Arab International University

The faculty of Business Administration

Management information system

Acrux AI FZ-LLC (Spike)
An online platform that makes AI accessible for all businesses, governments and other enterprises

Supervisor:
Dr. Jaafar Salman

By:
MHD Tarek Chikh Alshabab

Submitted to the faculty of the Business Administration/ Management Information Technology department of the Arab International University in partial fulfillment of the requirements for the Degree of Bachelor of Business Administration

Semester 2019-2020/1 - Submission date: 11/02/2020

Acknowledgment

First and foremost, I have to thank my parents for their love and support throughout my life. Thank you both for giving me strength to reach for the stars and chase my dreams. My brother, little sisters, deserve my wholehearted thanks as well.

I would like to sincerely thank my supervisor, Dr. Jaafar Salman, for his guidance and support throughout this study, and especially for his confidence in me. I would like to thank Dr. Kinaz Al-Aytouni and Prof. Sulaiman Mouselli, in a special way, I express my heartfelt gratefulness for her guide and support that I believed I learned from the best.

I would like to express my special gratitude and thanks to all AIU’s Drs, Teachers and Staffs for giving us such attention and time throughout our bachelor’s degree years.

To all my friends, thank you for your understanding and encouragement in my many, many moments of crisis. Your friendship makes my life a wonderful experience. I cannot list all the names here, but you are always on my mind.

Thank you, Lord, for always being there for me.

This thesis is only the beginning of my journey.

Introduction

Every business needs artificial intelligence.

But not every business has the capability to build, maintain, and generate value from AI in-house.
Spike makes AI accessible for all businesses, governments and other enterprises.

By harnessing the power of your data to assist – not displace – humans at work, we improve business efficiency and productivity.

Spike is an artificial intelligence company. We help our customers do great things with data become AI-driven and put machine learning and artificial intelligence at the heart of their company.

one thing that businesses don't always understand it's their data that is their biggest asset they need to look after that data and harness it to its full potential and that's where AI and machine learning can come in it enables you to turn what is a hugely valuable asset into something that generates revenues profits and sustainable growth for your company to see examples of companies that are using AI amazingly

you don't have to look much further than Netflix and Amazon Amazon for example attribute 35% of their sales to their algorithms which generate recommendations and personalized product offerings Netflix say as much as 75% of all content is consumed by virtue of their AI-driven recommendation systems

Spikes uniquely positioned to help businesses become AI-driven we give them the technology and the skills they need to do this or with a business model that works for them so first and foremost

Spike built an artificial intelligence system that becomes a central system of intelligence within a company's business it streams any kind of data from anywhere into itself and interprets that data and then rapidly deploy machine learning and artificial intelligence the insights and the outputs of those algorithms can all be taken back into your other business systems and power automated intelligent AI-driven decisions

we don't just give companies the technology we take full ownership of the problem we give them the skills in the form of a managed service and a commercial model that works for them so businesses subscribe to Spike and we take ownership of delivering value from their data using machine learning in AI and

That's the difference with Spike it's full service its technology its people and it's a business model that our customers want to engage with.

What is Spike?

Spike, it’s an online platform that makes AI accessible for all businesses, governments and other enterprises.

An easy-to-use online platform that provides all of this service for the customer via a subscription model

1. mine data warehouse
2. Interactive Analytics
3. Prediction and forecasting report.

All of these futures without neet to any:

- No AI or machine learning experience required.
- No infrastructure need.
- 256-bit military encryption we ensure you a 100% encryption for your data.
- 50% more accurate in forecasts by reducing forecasting time from months to hours.
- Messy data? No problem, our team can help to sort it out.
- Simple commercial model our subscription models are simple and transparent.
- Easily integrate with your systems for any CRM, ERP or any other tools.
- We always here to help easy onboarding and ongoing support from our team of experts

Data Warehouse

A data warehouse is a central repository of information and data.
Data flows into a data warehouse from transactional systems, relational databases, and other sources, typically on a regular cadence.

Interactive Analytics

You can start querying data instantly, get results in seconds and pay only for the queries you run. And we make it easy to build stunning visualizations and rich dashboards that can be accessed from any browser or mobile device

AI and ML to predict the future

50% more accurate time-series forecasts with machine learning based on the same technology used at large enterprises with no machine learning experience required. With hours not months.

Why use Spike?

Improving customer experiences
Improving customer experiences and personalization is the principal reason why marketers are adopting AI and machine learning

Boost profits
Large automotive OEMs can boost their operating profits by up to 16% by deploying AI at scale across supply chain operations and production centers.

Positive ROI in AI
McKinsey found that 82% of enterprises adopting machine learning and AI have gained a financial return from their investments

Grow revenues and profits
Spike fuels competitive advantage. It adds an intelligence layer to existing systems and uses AI to generate insights, make predictions, and optimize
business processes.

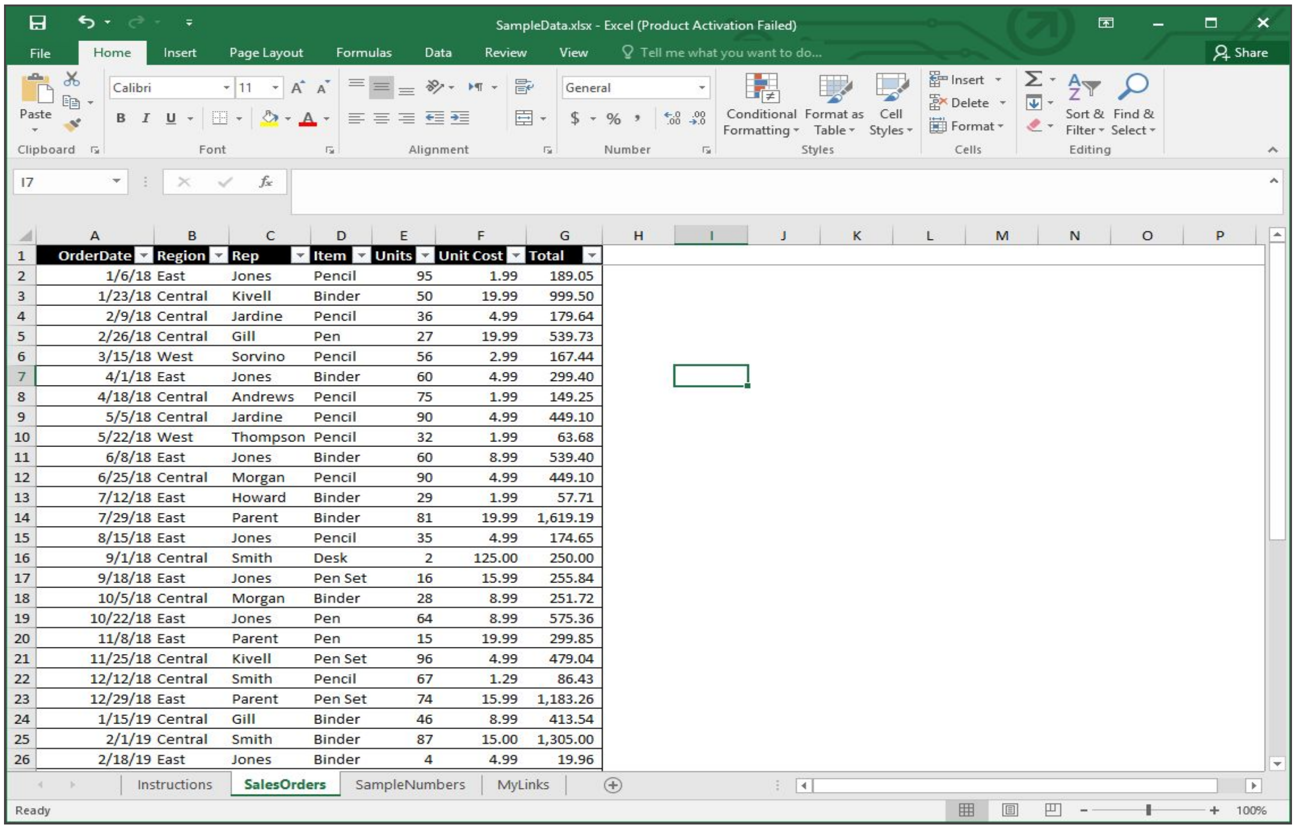
AI-powered businesses grow:

- 30% faster
- 50% higher

How to use Spike?

1- Import Your Data

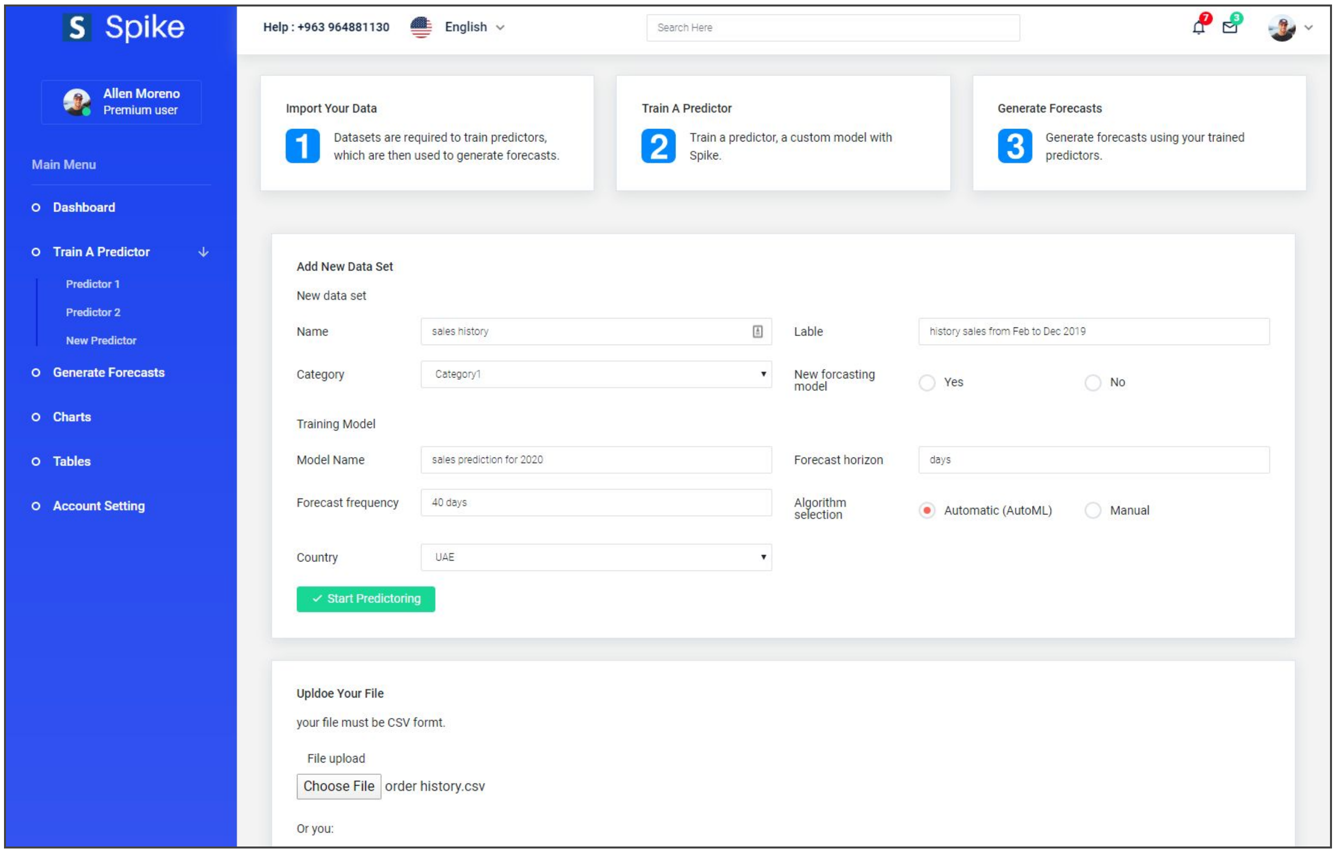
Datasets are required to train predictors, which are then used to generate forecasts.



OrderDate	Region	Rep	Item	Units	Unit Cost	Total
1/6/18	East	Jones	Pencil	95	1.99	189.05
1/23/18	Central	Kivell	Binder	50	19.99	999.50
2/9/18	Central	Jardine	Pencil	36	4.99	179.64
2/26/18	Central	Gill	Pen	27	19.99	539.73
3/15/18	West	Sorvino	Pencil	56	2.99	167.44
4/1/18	East	Jones	Binder	60	4.99	299.40
4/18/18	Central	Andrews	Pencil	75	1.99	149.25
5/5/18	Central	Jardine	Pencil	90	4.99	449.10
5/22/18	West	Thompson	Pencil	32	1.99	63.68
6/8/18	East	Jones	Binder	60	8.99	539.40
6/25/18	Central	Morgan	Pencil	90	4.99	449.10
7/12/18	East	Howard	Binder	29	1.99	57.71
7/29/18	East	Parent	Binder	81	19.99	1,619.19
8/15/18	East	Jones	Pencil	35	4.99	174.65
9/1/18	Central	Smith	Desk	2	125.00	250.00
9/18/18	East	Jones	Pen Set	16	15.99	255.84
10/5/18	Central	Morgan	Binder	28	8.99	251.72
10/22/18	East	Jones	Pen	64	8.99	575.36
11/8/18	East	Parent	Pen	15	19.99	299.85
11/25/18	Central	Kivell	Pen Set	96	4.99	479.04
12/12/18	Central	Smith	Pencil	67	1.29	86.43
12/29/18	East	Parent	Pen Set	74	15.99	1,183.26
1/15/19	Central	Gill	Binder	46	8.99	413.54
2/1/19	Central	Smith	Binder	87	15.00	1,305.00
2/18/19	East	Jones	Binder	4	4.99	19.96

2- Train A Predictor

Train a predictor, a custom model with the power of Spike AI and ML.



Import Your Data

1 Datasets are required to train predictors, which are then used to generate forecasts.

Train A Predictor

2 Train a predictor, a custom model with Spike.

Generate Forecasts

3 Generate forecasts using your trained predictors.

Add New Data Set

New data set

Name	sales history	Lable	history sales from Feb to Dec 2019
Category	Category1	New forcasting model	Yes No

Training Model

Model Name	sales prediction for 2020	Forecast horizon	days
Forecast frequency	40 days	Algorithm selection	[✓ Automatic (AutoML)] Manual
Country	UAE		

✓ Start Predictoring

Upldoe Your File

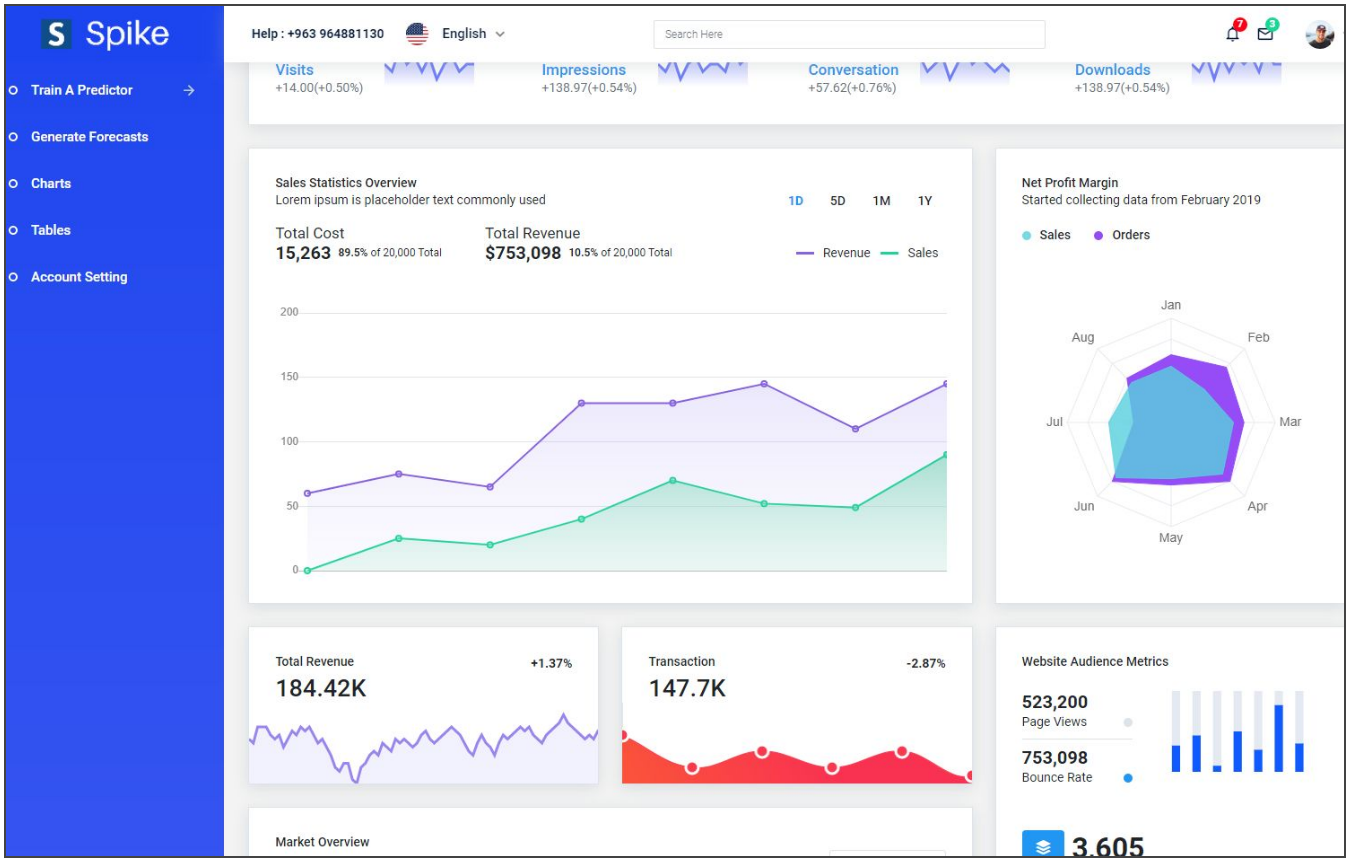
your file must be CSV formt.

File upload

Choose File order history.csv

Or you:

3- Generate Forecasts
Generate forecasts using your trained predictors.

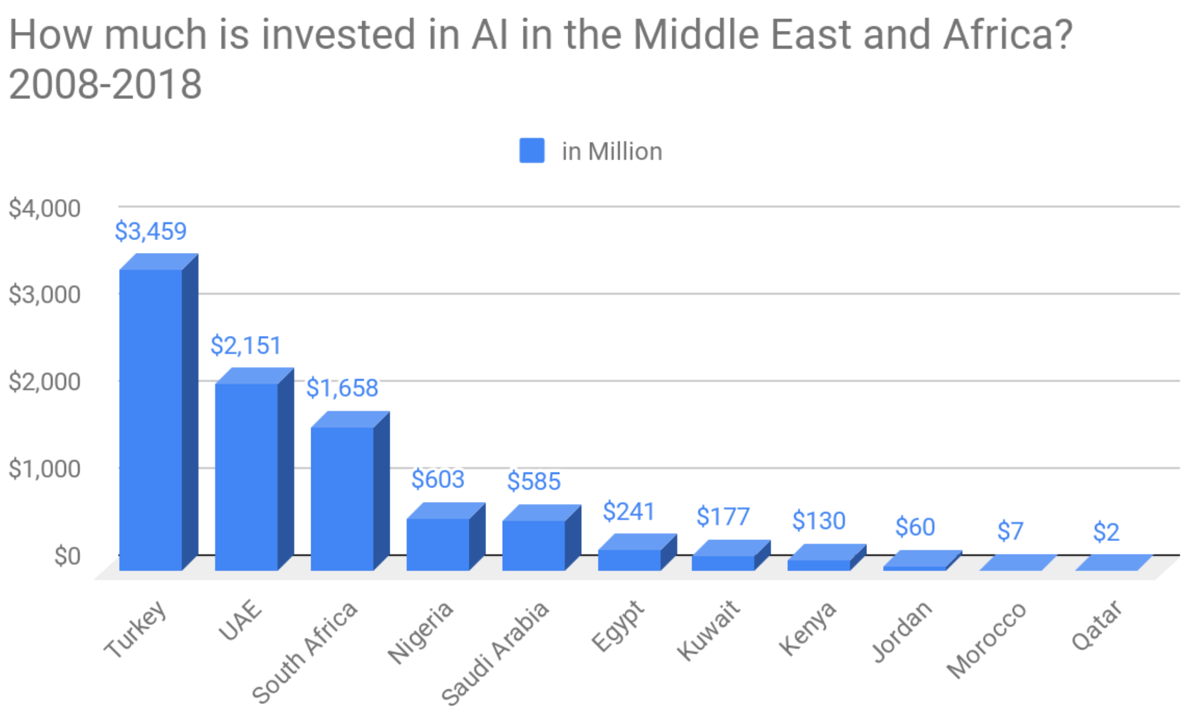


Market Size:

AI and machine learning have the potential to create an additional $2.6T in value by 2020 in Marketing and Sales, and up to $2T in manufacturing and supply chain planning.

IDC predicts worldwide spending on cognitive and Artificial Intelligence systems will reach $77.6B in 2022.

374,000,000$ Expected spending on AI systems in MEA for 2020



Spike Target Market:

Retail / E-Commerce

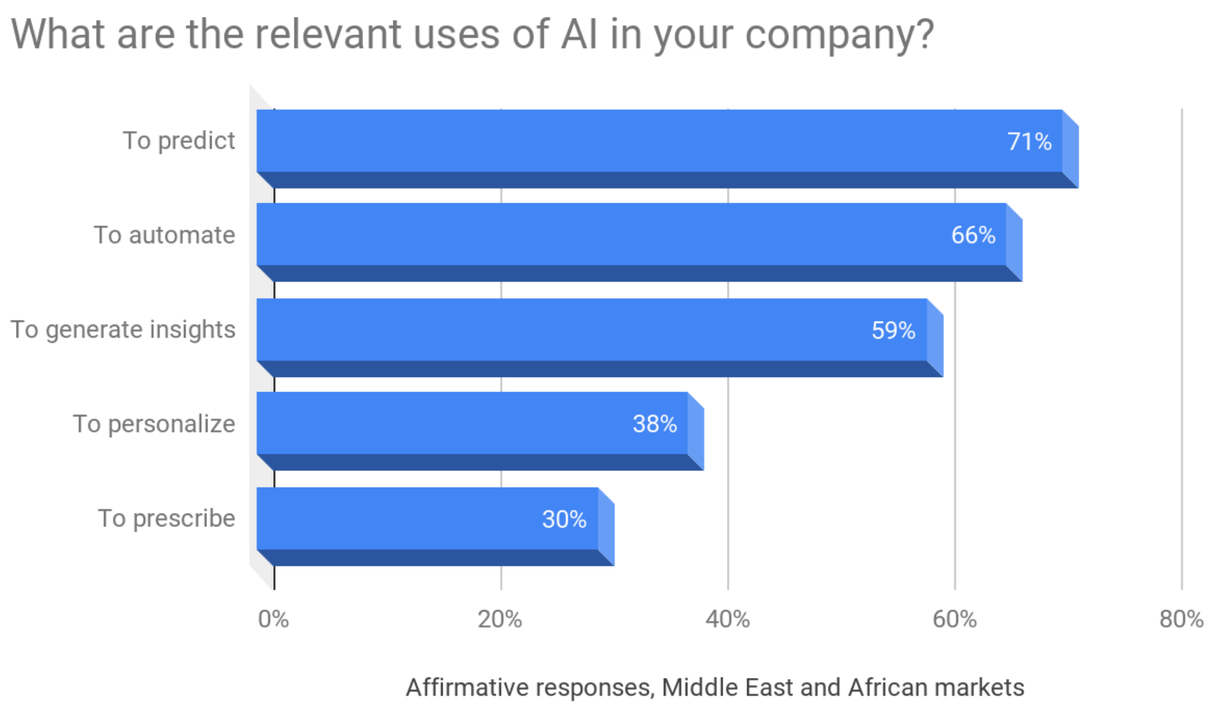
Increase forecasting and rebuying accuracy to drive profitability, avoid stockouts, optimize inventory sell-through, and reduce wastage.

Manufacturing

Reduce costs through better operational foresight, enabling more optimized warehouse operations and logistics processes.

Customer Service

Develop personal, one-to-one relationships with your customers to increase sales, reduce churn, and drive brand loyalty, with unique recommendations and personalized experience.



Operating Model:

Our Business Model:

With no up-front costs or consultancy fees, our simple subscription-based pricing structure is based on two factors:

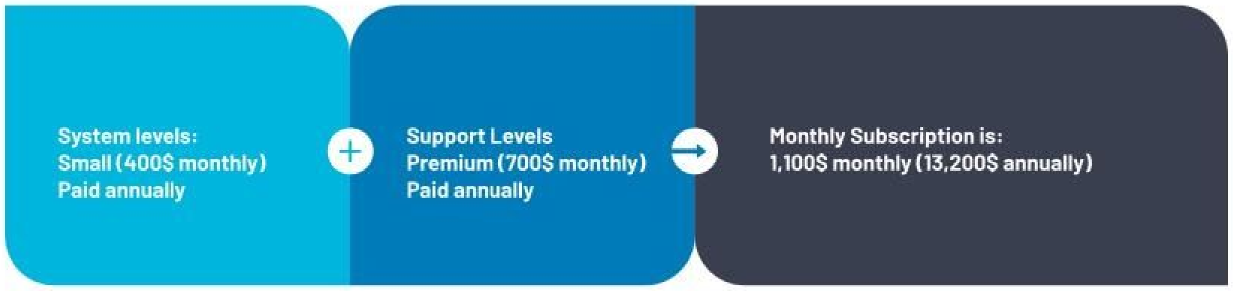
- **System levels**
- **Support Levels**

System levels			
	Small **400$ monthly**	**Medium** **700$ monthly**	**Large** **1000$ monthly**
Maximum data storage	**25 GB**	**50 GB**	**100 GB**
Ingestion frequency	**Daily**	**Hourly**	**Live**
Support Levels			
	Standard **400$ monthly**	**Premium** **700$ monthly**	**Premium+** **1000$ monthly**
Dedicated Customer Success Manager	**Yes**	**Yes**	**Yes**
Tailored solution enhancements	**12 days per year**	**24 days per year**	**24 days per year**
Expert support	**Working hours**	**Working hours**	**24/7**

Example pricing:

System levels: Small (300$ monthly) + Support Levels: Premium (600$ monthly) = Monthly Subscription is: 900$ monthly

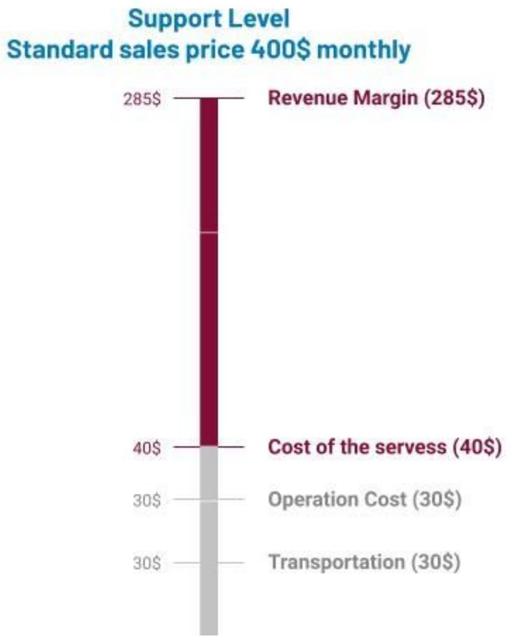
The customer at any point can change the plan and scale up or down in it the way that fits his needs.

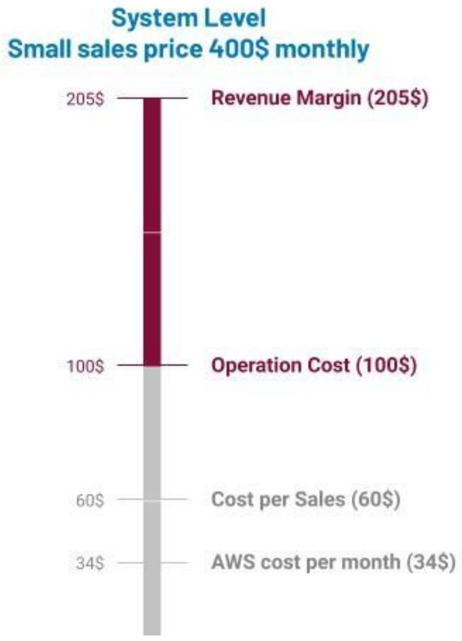


Fixed cost per month:

System Levels			
	Cost of good sold	Sales Price	Gross Profit
System Levels (Small)	194.5	400	195.225
System Level (Medium)	289.95	700	389.5475
System Level (Large)	466.7	1000	506.635
Support Levels			
	Cost of good sold	Sales Price	Gross Profit
Support Level (Standard)	100	400	285
Support Level (Premium)	170	700	503.5
Support Level (Premium+)	210	1000	750.5

Example:


Support Level
Standard sales price 400$ monthly
285$
Revenue Margin (285$)
40$
Cost of the servess (40$)
30$
Operation Cost (30$)
30$
Transportation (30$)


System Level
Small sales price 400$ monthly
205$
Revenue Margin (205$)
100$
Operation Cost (100$)
60$
Cost per Sales (60$)
34$
AWS cost per month (34$)

The projection for 6 Months

	month 1	month 2	month 3	month 4	month 5	month 6
REVENUES						
Revenues model 1	$0.00	$400.00	$800.00	$1,500.00	$2,600.00	$3,600.00
Revenues model 2	$0.00	$400.00	$1,100.00	$1,800.00	$2,900.00	$3,600.00
TOTAL REVENUES	$0.00	$800.00	$1,900.00	$3,300.00	$5,500.00	$7,200.00
COST OF GOODS SOLD						
Model 1	$0.00	$194.50	$389.00	$678.95	$1,163.40	$1,630.10
Model 2	$0.00	$100.00	$200.00	$370.00	$640.00	$850.00
TOTAL COST OF GOODS SOLD	$0.00	$294.50	$589.00	$1,048.95	$1,803.40	$2,480.10
GROSS PROFIT	$0.00	$505.50	$1,311.00	$2,251.05	$3,696.60	$4,719.90

OPERATION EXPENSE	month 1	month 2	month 3	month 4	month 5	month 6
CEO Salary	$1,900.00	$1,900.00	$1,900.00	$1,900.00	$1,900.00	$1,900.00
CTO Salary	$500.00	$500.00	$500.00	$800.00	$800.00	$800.00
Tech, Developer	$400.00	$400.00	$400.00	$500.00	$500.00	$500.00
Designer Salary	$0.00	$0.00	$500.00	$500.00	$500.00	$500.00
Customer Support Salary	$0.00	$0.00	$0.00	$300.00	$300.00	$300.00
Target Ads	$0.00	$200.00	$300.00	$400.00	$500.00	$600.00
Conferences Sponsor/Attending	$0.00	$0.00	$300.00	$0.00	$0.00	$300.00
Rant	$500.00	$500.00	$500.00	$500.00	$500.00	$500.00
License	$1,000.00	$0.00	$0.00	$0.00	$0.00	$0.00
Utility/Web Hosting/Other Expenses	$250.00	$250.00	$250.00	$250.00	$250.00	$250.00
TOTAL OPERATING EXPENSES	$4,550.00	$3,750.00	$4,650.00	$5,150.00	$5,250.00	$5,650.00
NET INCOME (LOSS)	-$4,550.00	-$3,244.50	-$3,339.00	-$2,898.95	-$1,553.40	-$930.10

Our Competitor

With our simple, easy to use platform and great support team our customers do not need any AI or machine learning experience required.

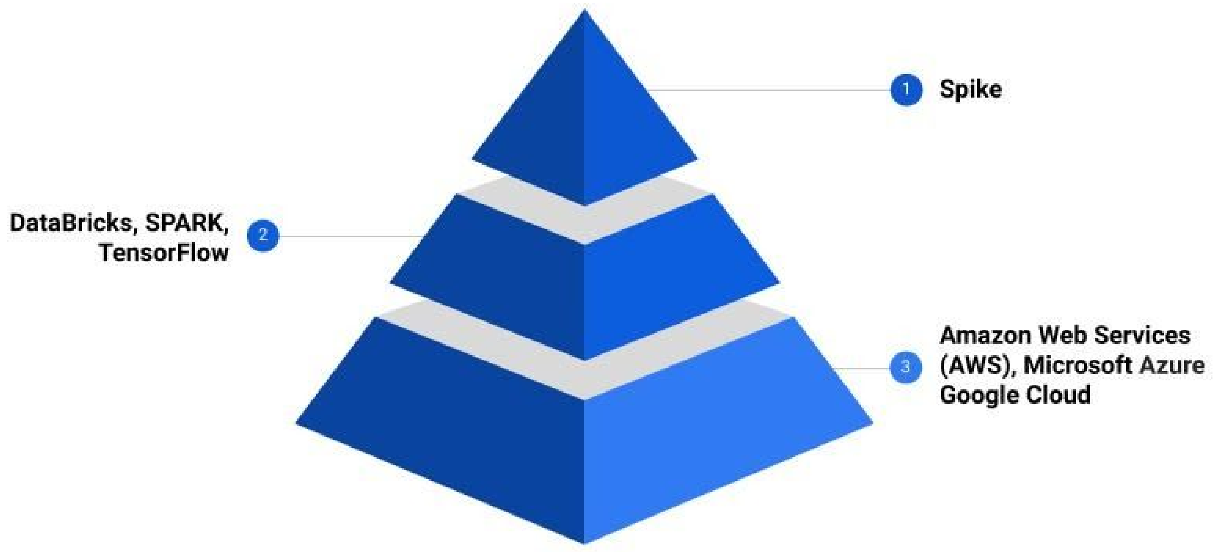
Thanks to our partnership with Amazon AWS, Microsoft Azure and Google Cloud there is no need for any infrastructure for our customers. We will provide the needed infrastructure and manage that for them.

There are two types of customers for this market:

1. Big Corporation which has their own teams and tools, they buy and investment in tech tools directly like AWS, Apache Spark and other tools. These tools it's used on a large scale and for a big company with a team, infrastructure-ready, Usually, they are technology companies. Here we don't target this type of customer.

2. The second type of customer which we are targeting here in Spike is non-technical and small business companies like Nike, Adidas and some e-consumers. The company makes a third-party deal with other partners to build their AI and ML models.

And here Spike comes to help this company to use AI and ML in their business to better decision making, more efficiency…


1 Spike
2 DataBricks, SPARK, TensorFlow
3 Amazon Web Services (AWS), Microsoft Azure Google Cloud

	SPIKE	SAP/ORACLE/ Microsoft BI	ANALYTICS Platform (ZOHO, Qlik, Tableau..)	DATABRICKS/ APACHE SPARK	SnowFlake / DATA management platform
EASY TO USE	✓	✓	✓		✓
Data Management	✓	✓		✓	✓
MENA REGEN	✓	✓	✓		
AI/ML for forecasting	✓			✓	
PRICE	✓				✓

Go-To-Market Plan

1. **Identify opportunities**
   We take the time to understand your business objectives and help you identify and prioritize your AI opportunities.

2. **Quantify opportunities**
   Leveraging a sample of your data, our team of AI consultants can build an evidence-based case for AI.

3. **Business case**
   Working closely with you, we'll agree with the exact scope and pricing build a business case for sign off.

4. **Kick-off**
   Spike team of industry and AI experts help you to configure our AI System and solutions to your individual business needs.

Marketing Channel:

- **Blog/Case Study**
  Publish reports for many topics for the public to build awareness about data with ML and AI.

- **Confreasses/Workshop**
  Attending this kind of event can build for us a good audience base for our brand and services.

- **AWS Partner Network in MENA**
  Joining the Amazon Web Server partner network can help us to build a good reputation.

- **Direct Marketing for a potential customer**
  Sending a unique case study/report for our potential customer.

- **Target Ads in the web**
  That will help us to collect and know potential customers early and in good volume.

Why the United Arab Emirates? Why an In5 tech innovation incubator?

- Be in UAE that will give us more: speed to test our solution and development
- A great market is open and ready for new technology
- The hub for AI and ML in MEA Reagan with easy to grow in the neighboring market.

In5 tech innovation:

Started in 2013, in5 was created to enable opportunities for entrepreneurs in the technology sector in the UAE and the region. Since then, it has engaged with 4,000 technopreneurs that have secured over AED 58 million in funding.

in5 offers startups state of the art workspaces, training and meeting rooms, unparalleled exposure to local, regional and international investors as well as a constant stream of mentorship and sustainability focused activities such as seminars, training sessions and workshops.

The innovation centers are designed to offer 5 key benefits to support the growing number of students, entrepreneurs and start-ups seeking to get new ideas off the ground in Dubai’s competitive markets.

- In5 will be our home for the next 6 months to try, learn, and launch our dream.
- A Creative space will help us to get our startup to the next level.
- A great opportunity to open our business in the UAE.

How to apply for In5 tech innovation?

1. SUBMISSION

Founders and entrepreneurs apply to the incubation program submitting an online application, including relevant material and information about their concept and business model.

2. VALIDATION

The in5 team shortlists candidates based on various eligibility criteria & invites selected candidates to present.

3. PRESENTATION

Shortlisted start-ups present their venture to a committee consisting of various industry experts and seasoned entrepreneurs, and answer the committee’s questions.

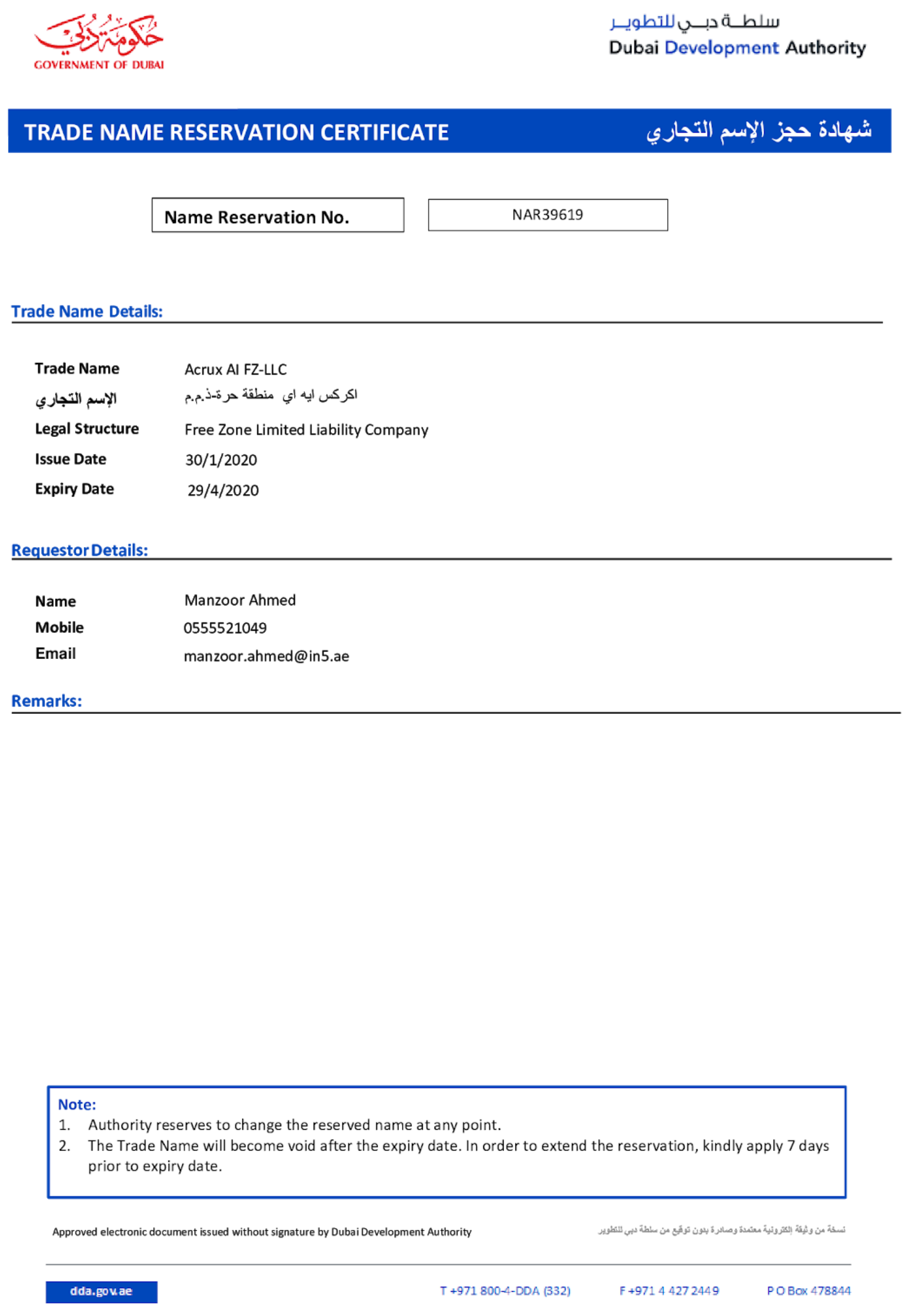
4. INCUBATION

The steering committee selects start-ups and admits them to the incubator. During their incubation period, startups access to various benefits that span the incubation value chain.

5. GRADUATION

On completion of the incubation phase, start-ups graduate and become part of the in5 alumni network.

Acrux AI FZ-LLC (Spike) Name Reservation Certificate with In5 incubator at Dubai City of interne



حكومة دبي
GOVERNMENT OF DUBAI

سلطة دبي للتطوير
Dubai Development Authority

TRADE NAME RESERVATION CERTIFICATE شهادة حجز الإسم التجاري

Name Reservation No.	NAR39619

Trade Name Details:

Trade Name	Acrux AI FZ-LLC
الإسم التجاري	اكركس ايه اي منطقة حرة-ذ.م.م
Legal Structure	Free Zone Limited Liability Company
Issue Date	30/1/2020
Expiry Date	29/4/2020

Requestor Details:

Name	Manzoor Ahmed
Mobile	0555521049
Email	manzoor.ahmed@in5.ae

Remarks:

Note:
1. Authority reserves to change the reserved name at any point.
2. The Trade Name will become void after the expiry date. In order to extend the reservation, kindly apply 7 days prior to expiry date.

Approved electronic document issued without signature by Dubai Development Authority

نسخة من وثيقة إلكترونية معتمدة وصادرة بدون توقيع من سلطة دبي للتطوير

dda.gov.ae T +971 800-4-DDA (332) F +971 4 427 2449 P O Box 478844

The used programming language in Spike

Python as programing language:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

R (programming language)

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, data mining surveys, and studies of scholarly literature databases show substantial increases in popularity; as of January 2020, R ranks 18th in the TIOBE index, a measure of popularity of programming languages.

A GNU package, source code for the R software environment is written primarily in C, Fortran, and R itself and is freely available under the GNU General Public License. Pre-compiled binary versions are provided for various operating systems. Although R has a command line interface, there are several graphical user interfaces, such as RStudio, an integrated development environment.

Why Python for Data Analysis?

For many people, the Python programming language has strong appeal. Since its first appearance in 1991, Python has become one of the most popular interpreted pro- gramming languages, along with Perl, Ruby, and others. Python and Ruby have become especially popular since 2005 or so for building websites using their numer- ous web frameworks, like Rails (Ruby) and Django (Python). Such languages are often called scripting languages, as they can be used to quickly write small programs, or scripts to automate other tasks. I don't like the term "scripting language," as it car- ries a connotation that they cannot be used for building serious software. Among interpreted languages, for various historical and cultural reasons, Python has devel- oped a large and active scientific computing and data analysis community. In the last 10 years, Python has gone from a bleeding-edge or "at your own risk" scientific com- puting language to one of the most important languages for data science, machine learning, and general software development in academia and industry.

For data analysis and interactive computing and data visualization, Python will inevi- tably draw comparisons with other open source and commercial programming lan- guages and tools in wide use, such as R, MATLAB, SAS, Stata, and others. In recent years, Python's improved support for libraries (such as pandas and scikit-learn) has made it a popular choice for data analysis tasks. Combined with Python's overall strength for general-purpose software engineering, it is an excellent option as a pri- mary language for building data applications

Why Not Python?

While Python is an excellent environment for building many kinds of analytical applications and general-purpose systems, there are a number of uses for which Python may be less suitable.

As Python is an interpreted programming language, in general most Python code will run substantially slower than code written in a compiled language like Java or C++. As programmer time is often more valuable than CPU time, many are happy to make this trade-off. However, in an application with very low latency or demanding resource utilization requirements (e.g., a high-frequency trading system), the time spent programming in a lower-level (but also lower-productivity) language like C++ to achieve the maximum possible performance might be time well spent.

Python can be a challenging language for building highly concurrent, multithreaded applications, particularly applications with many CPU-bound threads. The reason for this is that it has what is known as the global interpreter lock (GIL), a mechanism that prevents the interpreter from executing more than one Python instruction at a time. While it is true that in many big data processing applications, a cluster of computers may be required to process a dataset in a reasonable amount of time, there are still situations where a single-process, multithreaded system is desirable.

This is not to say that Python cannot execute truly multithreaded, parallel code. Python C extensions that use native multithreading (in C or C++) can run code in parallel without being impacted by the GIL, so long as they do not need to regularly interact with Python objects.

JavaScript

JavaScript, often abbreviated as JS, is an interpreted programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). However, the language itself does not include any input/output (I/O), such as networking, storage, or graphics facilities, as the host environment (usually a web browser) provides those APIs.

HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `<img />` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

PHP

PHP is a general-purpose programming language originally designed for web development. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.

PHP code may be executed with a command line interface (CLI), embedded into HTML code, or used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in a web server or as a Common Gateway Interface (CGI) executable. The web server outputs the results of the interpreted and executed PHP code, which may be any type of data, such as generated HTML code or binary image data. PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control.

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

Tools/Library used to build the platform

Apache HTTP Server

The Apache HTTP Server, colloquially called Apache, is free and open-source cross-platform web server software, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

The vast majority of Apache HTTP Server instances run on a Linux distribution, but current versions also run on Microsoft Windows and a wide variety of Unix-like systems. Past versions also ran on OpenVMS, NetWare, OS/2 and other operating systems, including ports to mainframes.

Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache played a key role in the initial growth of the World Wide Web, quickly overtaking NCSA HTTPd as the dominant HTTP server, and has remained most popular since April 1996. In 2009, it became the first web server software to serve more than 100 million websites. As of August 2019, it was estimated to serve 29% of all active websites, ranked 2nd after nginx at 32%, and 32% of the top million websites, ranked 2nd after "Other" with 33%.

PHP Laravel Framework

Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern and based on Symfony. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.

Features

The following features serve as Laravel's key design points (where not specifically noted, descriptions refer to the features of Laravel 3):

- Bundles provide a modular packaging system since the release of Laravel 3, with bundled features already available for easy addition to applications. Furthermore, Laravel 4 uses Composer as a dependency manager to add framework-agnostic and Laravel-specific PHP packages available from the Packagist repository.

- Eloquent ORM (object-relational mapping) is an advanced PHP implementation of the active record pattern, providing at the same time internal methods for enforcing constraints on the relationships between database objects. Following the active record pattern, Eloquent ORM presents database tables as classes, with their object instances tied to single table rows.

- Query builder, available since Laravel 3, provides a more direct database access alternative to the Eloquent ORM. Instead of requiring SQL queries to be written directly, Laravel's query builder provides a set of classes and methods capable of building queries programmatically. It also allows selectable caching of the results of executed queries.

- Application logic is an integral part of developed applications, implemented either by using controllers or as part of the route declarations. The syntax used to define application logic is similar to the one used by Sinatra framework.

- Reverse routing defines a relationship between the links and routes, making it possible for later changes to routes to be automatically propagated into relevant links. When the links are created by using names of existing routes, the appropriate uniform resource identifiers (URIs) are automatically created by Laravel.

- Restful controllers provide an optional way for separating the logic behind serving HTTP GET and POST requests.
- Class auto loading provides automated loading of PHP classes without the need for manual maintenance of inclusion paths. On-demand loading prevents inclusion of unnecessary components, so only the actually used components are loaded.

- View composers serve as customizable logical code units that can be executed when a view is loaded.

- Blade templating engine combines one or more templates with a data model to produce resulting views, doing that by transpiling the templates into cached PHP code for improved performance. Blade also provides a set of its own control structures such as conditional statements and loops, which are internally mapped to their PHP counterparts. Furthermore, Laravel services may be called from Blade templates, and the templating engine itself can be extended with custom directives.

- IoC containers make it possible for new objects to be generated by following the inversion of control (IoC) principle, in which the framework calls into the application- or task-specific code, with optional instantiating and referencing of new objects as singletons.

- Migrations provide a version control system for database schemas, making it possible to associate changes in the application's codebase and required changes in the database layout. As a result, this feature simplifies the deployment and updating of Laravel-based applications.

- Database seeding provides a way to populate database tables with selected default data that can be used for application testing or be performed as part of the initial application setup.

- Unit testing is provided as an integral part of Laravel,which itself contains unit tests that detect and prevent regressions in the framework. Unit tests can be run through the provided artisan command-line utility.

- Automatic pagination simplifies the task of implementing pagination, replacing the usual manual implementation approaches with automated methods integrated into Laravel.

- Form request is a feature of Laravel 5 that serves as the base for form input validation by internally binding event listeners, resulting in automated invoking of the form validation methods and generation of the actual form.

- Homestead - a Vagrant virtual machine that provides Laravel developers with all the tools necessary to develop Laravel straight out of the box, including, Ubuntu, Gulp, Bower and other development tools that are useful in developing full scale web applications.

- Canvas - a Laravel-powered publishing platform that helps visualize monthly trends, see where readers are coming from and what time of day they prefer to read content. Features like: Publication Statistics, Distraction-free writing, Unsplash Integration, Custom Social Data.

- Lazy Collection - This feature of the PHP framework Laravel 6, primarily enables you to deal with heavy loads of data, while keeping the memory usage low. Moreover, when you switch from all (_ to cursor ( ), just one expressive model is moved within the memory at a time as cursor ( ) makes use of the LazyCollection instance.

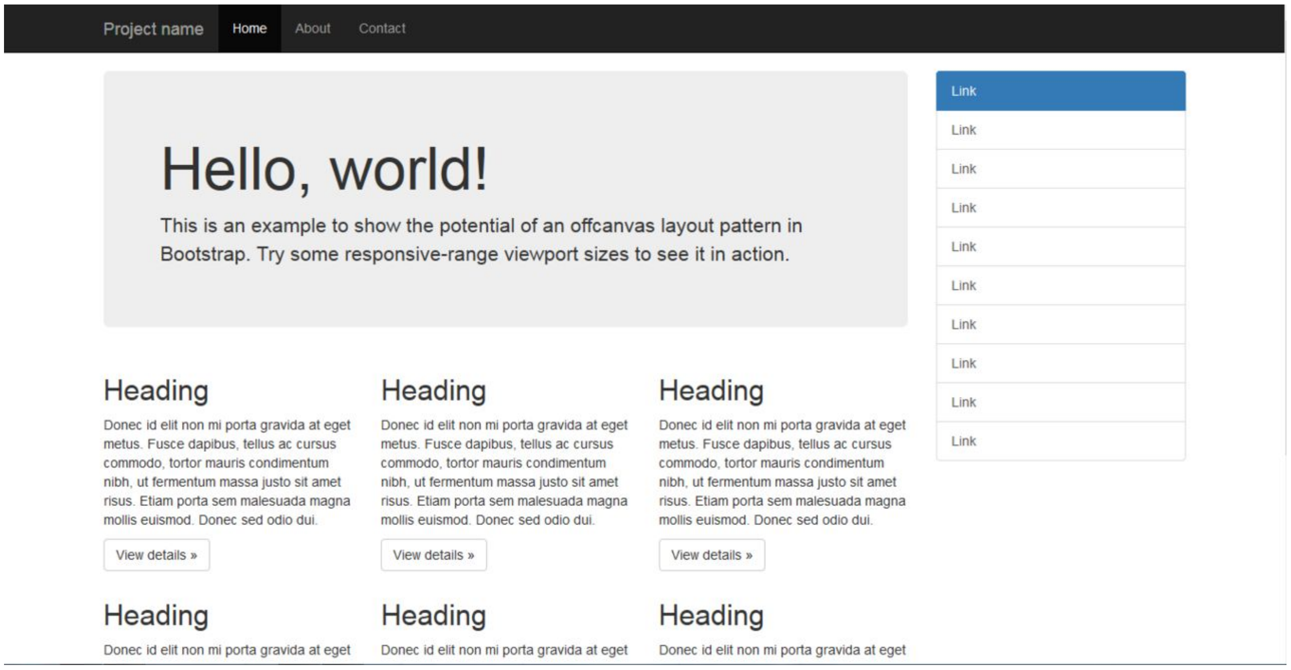
Bootstrap (front-end framework)

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation and other interface components.

Bootstrap is the sixth-most-starred project on GitHub, with more than 135,000 stars, behind freeCodeCamp (almost 307,000 stars) and marginally behind Vue.js framework.

Bootstrap is a web framework that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

Bootstrap also comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields.


Project name Home About Contact

Link
Link
Link
Link
Link
Link
Link
Link
Link
Link

Hello, world!

This is an example to show the potential of an offcanvas layout pattern in Bootstrap. Try some responsive-range viewport sizes to see it in action.

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

View details »

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

View details »

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

View details »

Heading

Donec id elit non mi porta gravida at eget

Heading

Donec id elit non mi porta gravida at eget

Heading

Donec id elit non mi porta gravida at eget

The most prominent components of Bootstrap are its layout components, as they affect an entire web page. The basic layout component is called "Container", as every other element in the page is placed in it. Developers can choose between a fixed-width container and a fluid-width container. While the latter always fills the width of the web page, the former uses one of the four predefined fixed widths, depending on the size of the screen showing the page:

- Smaller than 576 pixels
- 576–768 pixels
- 768–992 pixels
- 992–1200 pixels
- Larger than 1200 pixels

Once a container is in place, other Bootstrap layout components implement a CSS grid layout through defining rows and columns.

A precompiled version of Bootstrap is available in the form of one CSS file and three JavaScript files that can be readily added to any project. The raw form of Bootstrap, however, enables developers to implement further customization and size optimizations. This raw form is modular, meaning that the developer can remove unneeded components, apply a theme and modify the uncompiled Sass files.

jQuery

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License. As of May 2019, jQuery is used by 73% of the 10 million most popular websites. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin, having 3 to 4 times more usage than any other JavaScript library.

jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

The set of jQuery core features—DOM element selections, traversal and manipulation—enabled by its selector engine (named "Sizzle" from v1.3), created a new "programming style", fusing algorithms and DOM data structures. This style influenced the architecture of other JavaScript frameworks like YUI v3 and Dojo, later stimulating the creation of the standard Selectors API. Later, this style has been enhanced with a deeper algorithm-data fusion in an heir of jQuery, the D3.js framework.

Microsoft and Nokia bundle jQuery on their platforms Microsoft includes it with Visual Studio for use within Microsoft's ASP.NET AJAX and ASP.NET MVC frameworks while Nokia has integrated it into the Web Run-Time widget development platform.

Chart.js

Chart.js is a community maintained open-source library (it's available on GitHub) that helps you easily visualize data using JavaScript. It's similar to Chartist and Google Charts. It supports 8 different chart types (including bars, lines, & pies), and they're all responsive. In other words, you set up your chart once, and Chart.js will do the heavy-lifting for you and make sure that it's always legible (for example by removing some uncritical details if the chart gets smaller).

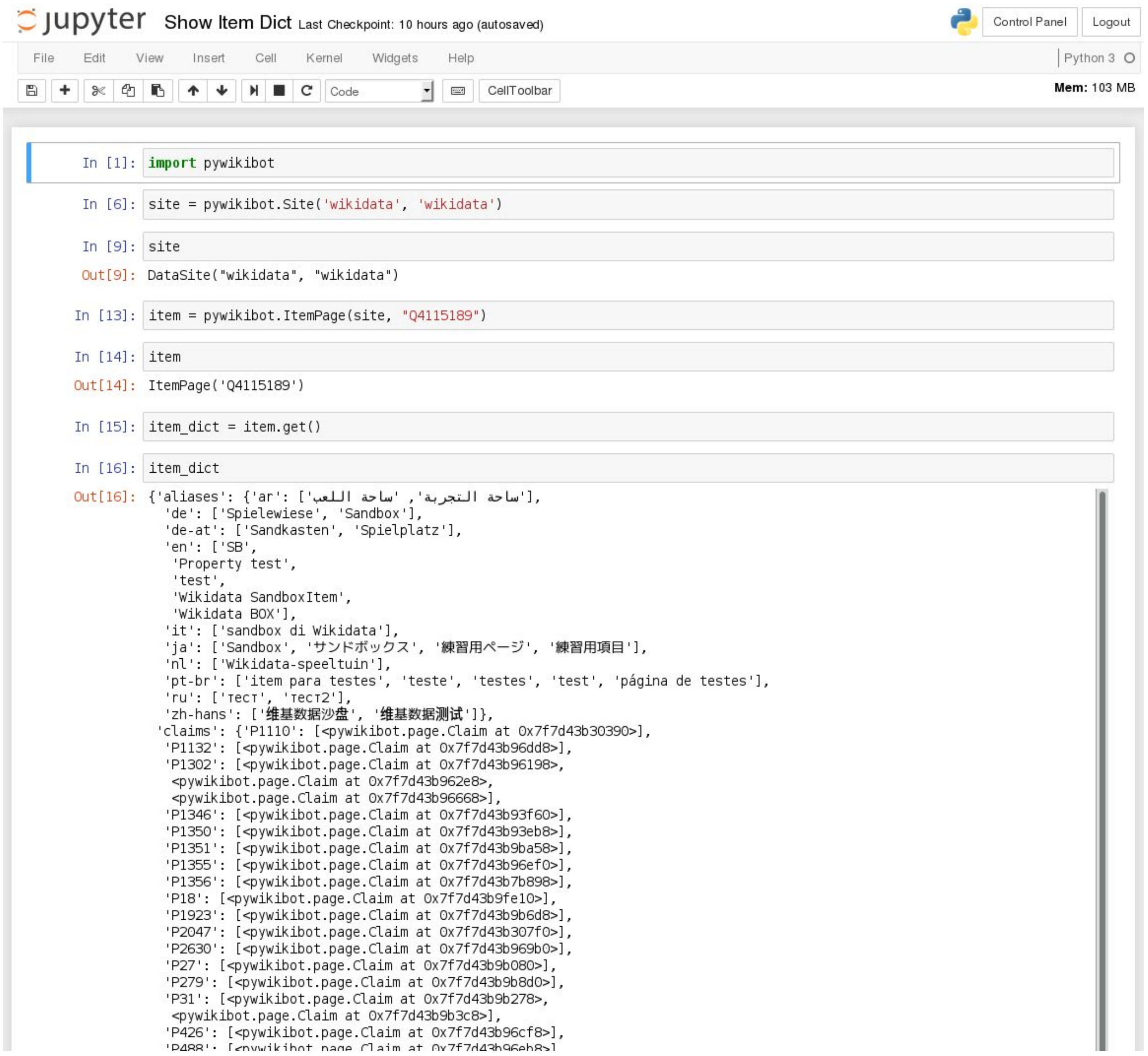
In a gist, this is what you need to do to draw a chart with Chart.js:

1. Define where on your page to draw the graph.
2. Define what type of graph you want to draw.
3. Supply Chart.js with data, labels, and other options.

And you'll get a beautiful, responsive, graph! Although we won't dig too deep into changing the design of our graph in this tutorial, Chart.js graphs are highly customizable. As a rule, whatever you see you can affect, and although the charts look good without much customization, you'll likely be able to realize all your (or someone else's) design visions with some extra effort.

IPython and Jupyter

The IPython project began in 2001 as Fernando Pérez's side project to make a better interactive Python interpreter. In the subsequent 16 years it has become one of the most important tools in the modern Python data stack. While it does not provide any computational or data analytical tools by itself, IPython is designed from the ground up to maximize your productivity in both interactive computing and software development. It encourages an execute-explore workflow instead of the typical edit-compile run workflow of many other programming languages. It also provides easy access to your operating system's shell and filesystem. Since much of data analysis coding involves exploration, trial and error, and iteration, IPython can help you get the job done faster.



```
In [1]: import pywikibot

In [6]: site = pywikibot.Site('wikidata', 'wikidata')

In [9]: site
Out[9]: DataSite("wikidata", "wikidata")

In [13]: item = pywikibot.ItemPage(site, "Q4115189")

In [14]: item
Out[14]: ItemPage('Q4115189')

In [15]: item_dict = item.get()

In [16]: item_dict
Out[16]: {'aliases': {'ar': ['ساحة التجربة', 'ساحة اللعب'],
           'de': ['Spielewiese', 'Sandbox'],
           'de-at': ['Sandkasten', 'Spielplatz'],
           'en': ['SB',
            'Property test',
            'test',
            'Wikidata SandboxItem',
            'Wikidata BOX'],
           'it': ['sandbox di Wikidata'],
           'ja': ['Sandbox', 'サンドボックス', '練習用ページ', '練習用項目'],
           'nl': ['Wikidata-speeltuin'],
           'pt-br': ['item para testes', 'teste', 'testes', 'test', 'página de testes'],
           'ru': ['тест', 'тест2'],
           'zh-hans': ['维基数据沙盘', '维基数据测试']},
          'claims': {'P1110': [<pywikibot.page.Claim at 0x7f7d43b30390>],
           'P1132': [<pywikibot.page.Claim at 0x7f7d43b96dd8>],
           'P1302': [<pywikibot.page.Claim at 0x7f7d43b96198>,
            <pywikibot.page.Claim at 0x7f7d43b962e8>,
            <pywikibot.page.Claim at 0x7f7d43b96668>],
           'P1346': [<pywikibot.page.Claim at 0x7f7d43b93f60>],
           'P1350': [<pywikibot.page.Claim at 0x7f7d43b93eb8>],
           'P1351': [<pywikibot.page.Claim at 0x7f7d43b9ba58>],
           'P1355': [<pywikibot.page.Claim at 0x7f7d43b96ef0>],
           'P1356': [<pywikibot.page.Claim at 0x7f7d43b7b898>],
           'P18': [<pywikibot.page.Claim at 0x7f7d43b9fe10>],
           'P1923': [<pywikibot.page.Claim at 0x7f7d43b9b6d8>],
           'P2047': [<pywikibot.page.Claim at 0x7f7d43b307f0>],
           'P2630': [<pywikibot.page.Claim at 0x7f7d43b969b0>],
           'P27': [<pywikibot.page.Claim at 0x7f7d43b9b080>],
           'P279': [<pywikibot.page.Claim at 0x7f7d43b9b8d0>],
           'P31': [<pywikibot.page.Claim at 0x7f7d43b9b278>,
            <pywikibot.page.Claim at 0x7f7d43b9b3c8>],
           'P426': [<pywikibot.page.Claim at 0x7f7d43b96cf8>],
           'P488': [<pywikibot.page.Claim at 0x7f7d43b96eb8>]
```

In 2014, Fernando and the IPython team announced the Jupyter project, a broader initiative to design language-agnostic interactive computing tools. The IPython web notebook became the Jupyter notebook, with support now for over 40 programming languages. The IPython system can now be used as a kernel (a programming language mode) for using Python with Jupyter.

IPython itself has become a component of the much broader Jupyter open source project, which provides a productive environment for interactive and exploratory computing. Its oldest and simplest "mode" is as an enhanced Python shell designed to accelerate the writing, testing, and debugging of Python code. You can also use the IPython system through the Jupyter Notebook, an interactive web-based code "note- book" offering support for dozens of programming languages.

The IPython shell and Jupyter notebooks are especially useful for data exploration and visualization. The Jupyter notebook system also allows you to author content in Markdown and HTML, providing you a means to create rich documents with code and text. Other programming languages have also implemented kernels for Jupyter to enable you to use languages other than Python in Jupyter.

For me personally, IPython is usually involved with the majority of my Python work, including running, debugging, and testing code.

NumPy

NumPy, short for Numerical Python, has long been a cornerstone of numerical com-
puting in Python. It provides the data structures, algorithms, and library glue needed
for most scientific applications involving numerical data in Python. NumPy contains,
among other things:

- A fast and efficient multidimensional array object ndarray
- Functions for performing element-wise computations with arrays or mathemati-
- cal operations between arrays
- Tools for reading and writing array-based datasets to disk
- Linear algebra operations, Fourier transform, and random number generation
- A mature C API to enable Python extensions and native C or C++ code to access NumPy's data structures and computational facilities

Beyond the fast array-processing capabilities that NumPy adds to Python, one of its
primary uses in data analysis is as a container for data to be passed between algo-
rithms and libraries. For numerical data, NumPy arrays are more efficient for storing
and manipulating data than the other built-in Python data structures. Also, libraries
written in a lower-level language, such as C or Fortran, can operate on the data stored
in a NumPy array without copying data into some other memory representation.

Thus, many numerical computing tools for Python either assume NumPy arrays as a primary data structure or else target seamless interoperability with NumPy

Pandas

pandas provides high-level data structures and functions designed to make working with structured or tabular data fast, easy, and expressive. Since its emergence in 2010, it has helped enable Python to be a powerful and productive data analysis environment. The primary objects in pandas that will be used in this study are the **DataFrame**, a tabular, column-oriented data structure with both row and column labels, and the **Series**, a one-dimensional labeled array object.

pandas blends the high-performance, array-computing ideas of NumPy with the flexible data manipulation capabilities of spreadsheets and relational databases (such as SQL). It provides sophisticated indexing functionality to make it easy to reshape, slice and dice, perform aggregations, and select subsets of data. Since data manipulation,

preparation, and cleaning is such an important skill in data analysis.

As a bit of background, I started building pandas in early 2008 during my tenure at AQR Capital Management, a quantitative investment management firm. At the time, I had a distinct set of requirements that were not well addressed by any single tool at my disposal:

-
- Data structures with labeled axes supporting automatic or explicit data alignment
- —this prevents common errors resulting from misaligned data and working with
- differently indexed data coming from different sources
- Integrated time series functionality
- The same data structures handle both time series data and non–time series data
- Arithmetic operations and reductions that preserve metadata
- Flexible handling of missing data
- Merge and other relational operations found in popular databases (SQL-based,
- for example)

I wanted to be able to do all of these things in one place, preferably in a language well suited to general-purpose software development. Python was a good candidate language for this, but at that time there was not an integrated set of data structures and tools providing this functionality. As a result of having been built initially to solve finance and business analytics problems, pandas features especially deep time series functionality and tools well suited for working with time-indexed data generated by business processes.

For users of the R language for statistical computing, the DataFrame name will be familiar, as the object was named after the similar R data.frame object. Unlike Python, data frames are built into the R programming language and its standard library. As a result, many features found in pandas are typically either part of the R core implementation or provided by add-on packages.

The pandas name itself is derived from panel data, an econometrics term for multidimensional structured datasets, and a play on the phrase Python data analysis itself.

System Design

Agile software development

Agile software development comprises various approaches to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s). It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.[further explanation needed]

The term agile (sometimes written Agile) was popularized, in this context, by the Manifesto for Agile Software Development. The values and principles espoused in this manifesto were derived from and underpin a broad range of software development frameworks, including Scrum and Kanban.

While there is much anecdotal evidence that adopting agile practices and values improves the agility of software professionals, teams and organizations, some empirical studies have disputed that evidence.

The Manifesto for Agile Software Development

Agile software development values

Based on their combined experience of developing software and helping others do that, the seventeen signatories to the manifesto proclaimed that they value:

- Individuals and Interactions over processes and tools
- Working Software over comprehensive documentation
- Customer Collaboration over contract negotiation
- Responding to Change over following a plan

That is to say, the items on the left are valued more than the items on the right.
As Scott Ambler elucidated:

- Tools and processes are important, but it is more important to have competent people working together effectively.
- Good documentation is useful in helping people to understand how the software is built and how to use it, but the main point of development is to create software, not documentation.
- A contract is important but is no substitute for working closely with customers to discover what they need.

- A project plan is important, but it must not be too rigid to accommodate changes in technology or the environment, stakeholders' priorities, and people's understanding of the problem and its solution.

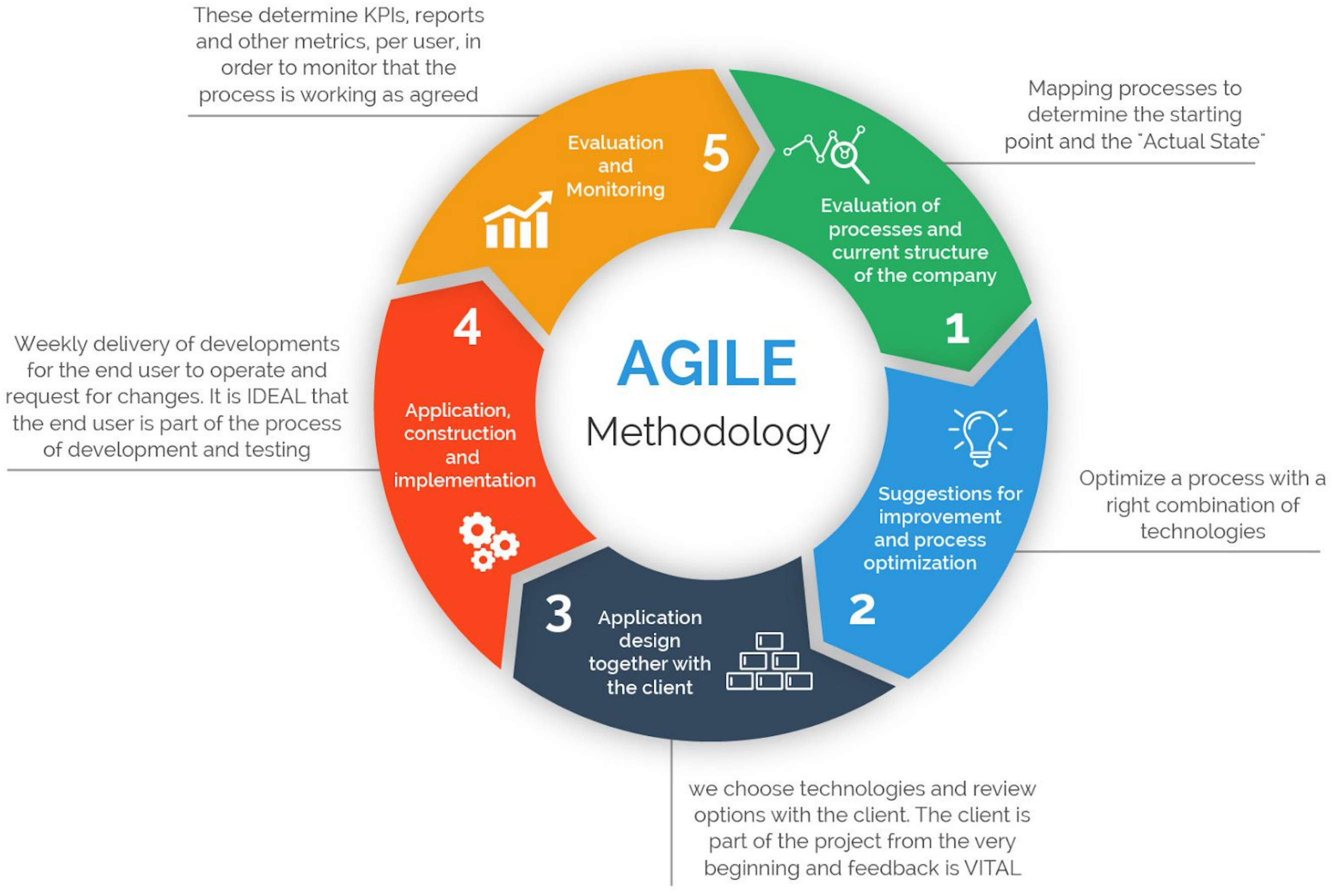
Some of the authors formed the Agile Alliance, a non-profit organization that promotes software development according to the manifesto's values and principles. Introducing the manifesto on behalf of the Agile Alliance, Jim Highsmith said,

The Agile movement is not anti-methodology, in fact many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment. Those who would brand proponents of XP or SCRUM or any of the other Agile Methodologies as "hackers" are ignorant of both the methodologies and the original definition of the term hacker.
— Jim Highsmith, History: The Agile Manifesto

Agile software development principles

The Manifesto for Agile Software Development is based on twelve principles:

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly


These determine KPIs, reports and other metrics, per user, in order to monitor that the process is working as agreed
Mapping processes to determine the starting point and the "Actual State"
Weekly delivery of developments for the end user to operate and request for changes. It is IDEAL that the end user is part of the process of development and testing
Optimize a process with a right combination of technologies
we choose technologies and review options with the client. The client is part of the project from the very beginning and feedback is VITAL
AGILE
Methodology
1
Evaluation of processes and current structure of the company
2
Suggestions for improvement and process optimization
3
Application design together with the client
4
Application, construction and implementation
5
Evaluation and Monitoring

System Design Cheatsheet

Basic Steps

1. **Clarify and agree on the scope of the system**

- User cases (description of sequences of events that, taken together, lead to a system doing something useful)
    - Who is going to use it?
    - How are they going to use it?
- Constraints
    - Mainly identify traffic and data handling constraints at scale.
    - Scale of the system such as requests per second, requests types, data written per second, data read per second)
    - Special system requirements such as multi-threading, read or write oriented.

2. **High level architecture design (Abstract design)**

- Sketch the important components and connections between them, but don't go into some details.
    - Application service layer (serves the requests)
    - List different services required.
    - Data Storage layer
    - eg. Usually a scalable system includes webserver (load balancer), service (service partition), database (master/slave database cluster) and caching systems.

3. **Component Design**

- Component + specific APIs required for each of them.
- Object oriented design for functionalities.
    - Map features to modules: One scenario for one module.
    - Consider the relationships among modules:
        - Certain functions must have unique instance (Singletons)
        - Core object can be made up of many other objects (composition).
        - One object is another object (inheritance)
- Database schema design.

4. **Understanding Bottlenecks**
   Perhaps your system needs a load balancer and many machines behind it to handle the user requests. * Or maybe the data is so huge that you need to distribute your database on multiple machines. What are some of the downsides that occur from doing that?
   Is the database too slow and does it need some in-memory caching?

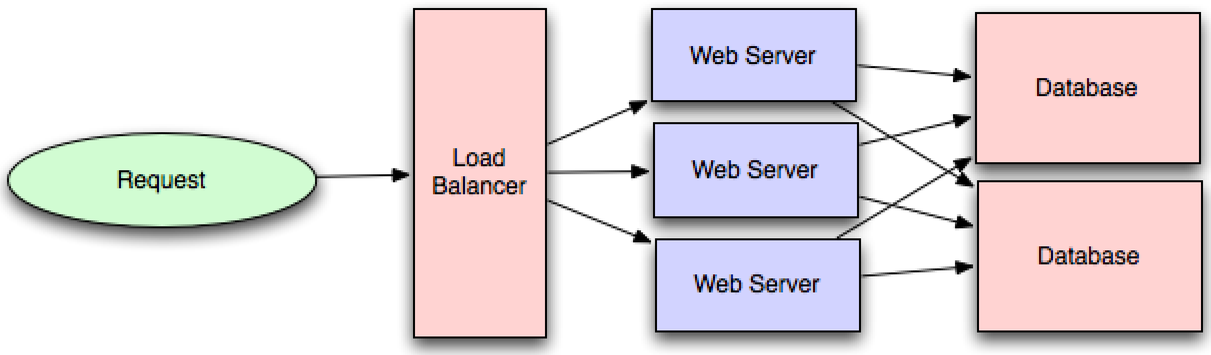
5. Scaling your abstract design

- **Vertical scaling**
    - You scale by adding more power (CPU, RAM) to your existing machine.

- **Horizontal scaling**
    - You scale by adding more machines into your pool of resources.

- **Caching**
    - Load balancing helps you scale horizontally across an ever-increasing number of servers, but caching will enable you to make vastly better use of the resources you already have, as well as making otherwise unattainable product requirements feasible.
    - Application caching requires explicit integration in the application code itself. Usually it will check if a value is in the cache; if not, retrieve the value from the database.
    - Database caching tends to be "free". When you flip your database on, you're going to get some level of default configuration which will provide some degree of caching and performance. Those initial settings will be optimized for a generic usecase, and by tweaking them to your system's access patterns you can generally squeeze a great deal of performance improvement.
    - In-memory caches are most potent in terms of raw performance. This is because they store their entire set of data in memory and accesses to RAM are orders of magnitude faster than those to disk. eg. Memcached or Redis.
    - eg. Precalculating results (e.g. the number of visits from each referring domain for the previous day),
    - eg. Pre-generating expensive indexes (e.g. suggested stories based on a user's click history)
    - eg. Storing copies of frequently accessed data in a faster backend (e.g. Memcache instead of PostgreSQL.

- **Load balancing**
    - Public servers of a scalable web service are hidden behind a load balancer. This load balancer evenly distributes load (requests from your users) onto your group/cluster of application servers.
    - Types: Smart client (hard to get it perfect), Hardware load balancers ($$$ but reliable), Software load balancers (hybrid - works for most systems)

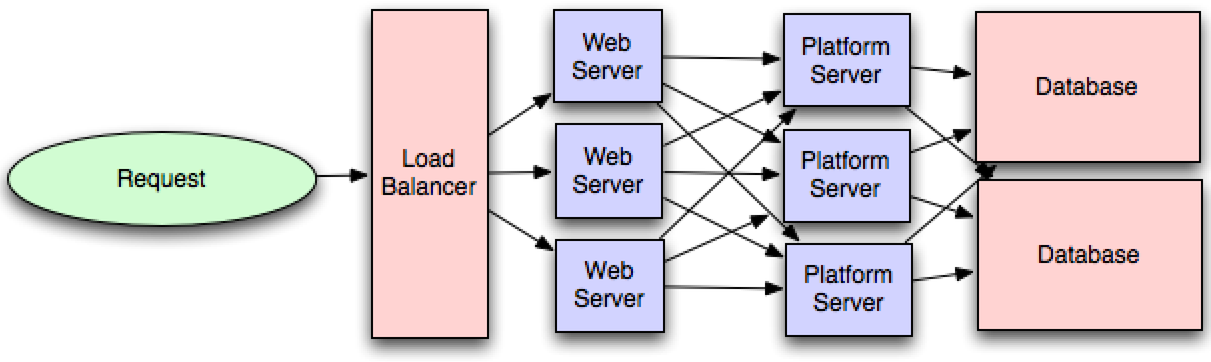


- **Database replication**
    - Database replication is the frequent electronic copying data from a database in one computer or server to a database in another so that all users share the same level of information. The result is a distributed database in which users can access data relevant to their tasks without interfering with the work of others. The implementation of database replication for the purpose of eliminating data ambiguity or inconsistency among users is known as normalization.

- **Database partitioning**
    - Partitioning of relational data usually refers to decomposing your tables either row-wise (horizontally) or column-wise (vertically).

- **Map-Reduce**
    - For sufficiently small systems you can often get away with adhoc queries on a SQL database, but that approach may not scale up trivially once the quantity of data stored or write-load requires sharding your database, and will usually require dedicated slaves for the purpose of performing these queries (at which point, maybe you'd rather use a system designed for analyzing large quantities of data, rather than fighting your database).
    - Adding a map-reduce layer makes it possible to perform data and/or processing intensive operations in a reasonable amount of time. You might use it for calculating suggested users in a social graph, or for generating analytics reports. eg. Hadoop, and maybe Hive or HBase.

- **Platform Layer (Services)**
    - Separating the platform and web application allow you to scale the pieces independently. If you add a new API, you can add platform servers without adding unnecessary capacity for your web application tier.
    - Adding a platform layer can be a way to reuse your infrastructure for multiple products or interfaces (a web application, an API, an iPhone app, etc) without writing too much redundant boilerplate code for dealing with caches, databases, etc.



Key topics for designing a system

1. **Concurrency**
   Do you understand threads, deadlock, and starvation? Do you know how to parallelize algorithms? Do you understand consistency and coherence?

2. **Networking**
   Do you roughly understand IPC and TCP/IP? Do you know the difference between throughput and latency, and when each is the relevant factor?

3. **Abstraction**
   You should understand the systems you're building upon. Do you know roughly how an OS, file system, and database work? Do you know about the various levels of caching in a modern OS?

4. **Real-World Performance**
   You should be familiar with the speed of everything your computer can do, including the relative performance of RAM, disk, SSD and your network.

5. **Estimation**
   Estimation, especially in the form of a back-of-the-envelope calculation, is important because it helps you narrow down the list of possible solutions to only the ones that are feasible. Then you have only a few prototypes or micro-benchmarks to write.

6. **Availability & Reliability**
   Are you thinking about how things can fail, especially in a distributed environment? Do know how to design a system to cope with network failures? Do you understand durability?

Web App System design considerations:

- Security (CORS)
- Using CDN
    - A content delivery network (CDN) is a system of distributed servers (network) that deliver webpages and other Web content to a user based on the geographic locations of the user, the origin of the webpage and a content delivery server.
    - This service is effective in speeding the delivery of content of websites with high traffic and websites that have global reach. The closer the CDN server is to the user geographically, the faster the content will be delivered to the user.
    - CDNs also provide protection from large surges in traffic.
- Full Text Search
    - Using Sphinx/Lucene/Solr - which achieve fast search responses because, instead of searching the text directly, it searches an index instead.
- Offline support/Progressive enhancement
    - Service Workers
- Web Workers
- Server Side rendering
- Asynchronous loading of assets (Lazy load items)
- Minimizing network requests (Http2 + bundling/sprites etc)
- Developer productivity/Tooling
- Accessibility
- Internationalization
- Responsive design
- Browser compatibility

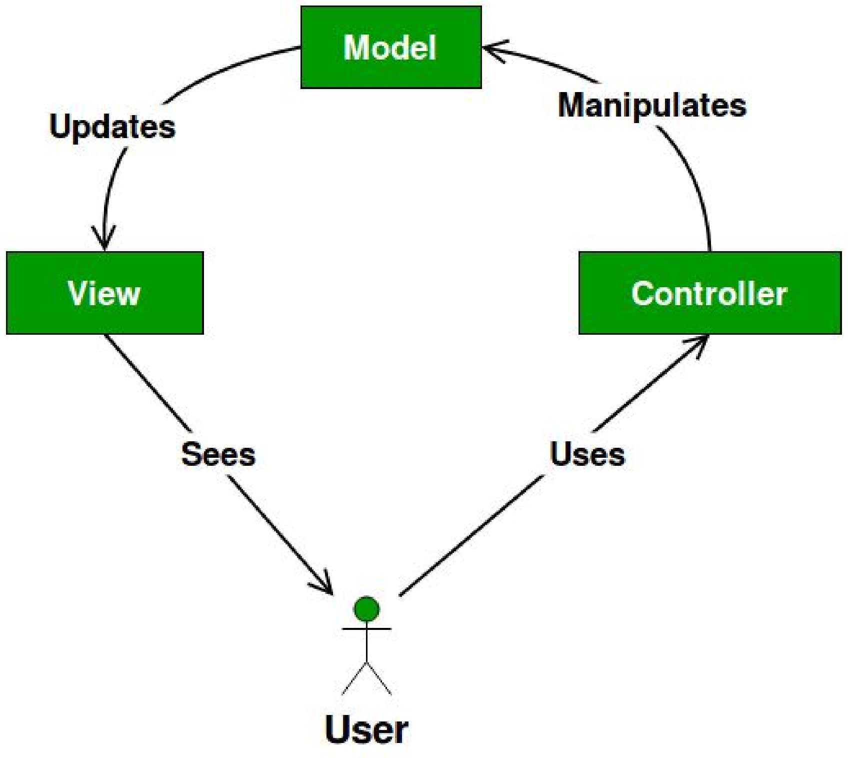
Working Components of Front-end Architecture

- **Code**
    - HTML5/WAI-ARIA
    - CSS/Sass Code standards and organization
    - Object-Oriented approach (how do objects break down and get put together)
    - JS frameworks/organization/performance optimization techniques
    - Asset Delivery - Front-end Ops
- **Documentation**
    - Onboarding Docs
    - Styleguide/Pattern Library
    - Architecture Diagrams (code flow, tool chain)
- **Testing**
    - Performance Testing
    - Visual Regression
    - Unit Testing
    - End-to-End Testing

- **Process**
    - Git Workflow
    - Dependency Management (npm, Bundler, Bower)
    - Build Systems (Grunt/Gulp)
    - Deploy Process
    - Continuous Integration (Travis CI, Jenkins)

Model–view–controller

Model–view–controller (usually known as MVC) is a software design pattern commonly used for developing user interfaces which divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. This kind of pattern is used for designing the layout of the page.



Traditionally used for desktop graphical user interfaces (GUIs), this pattern has become popular for designing web applications. Popular programming languages like JavaScript, Python, Ruby, PHP, Java, C#, and Swift have MVC frameworks that are used for web or mobile application development straight out of the box.

Model

The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

View

Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

Controller

Accepts input and converts it to commands for the model or view.
In addition to dividing the application into these components, the model–view–controller design defines the interactions between them.

- The model is responsible for managing the data of the application. It receives user input from the controller.
- The view means presentation of the model in a particular format.
- The controller responds to the user input and performs interactions on the data model objects. The controller receives the input, optionally validates it and then passes the input to the model.

As with other software patterns, MVC expresses the "core of the solution" to a problem while allowing it to be adapted for each system. Particular MVC designs can vary significantly from the traditional description here.

Simultaneous development

Because MVC decouples the various components of an application, developers are able to work in parallel on different components without affecting or blocking one another. For example, a team might divide their developers between the front-end and the back-end. The back-end developers can design the structure of the data and how the user interacts with it without requiring the user interface to be completed. Conversely, the front-end developers are able to design and test the layout of the application prior to the data structure being available.

Code reuse

The same (or similar) view for one application can be refactored for another application with different data because the view is simply handling how the data is being displayed to the user. Unfortunately this does not work when that code is also useful for handling user input. For example, DOM code (including the application's custom abstractions to it) is useful for both graphics display and user input. (Note that, despite the name Document Object Model, the DOM is actually not an MVC model, because it is the application's interface to the user).
To address these problems, MVC (and patterns like it) are often combined with a component architecture that provides a set of UI elements. Each UI element is a single higher-level component that combines the 3 required MVC components into a single package. By creating these higher-level components that are independent of each other, developers are able to reuse components quickly and easily in other applications.

Data Cleaning and Preparation

Data cleaning and preparation is the most critical first step in any AI project. As evidence shows, most data scientists spend most of their time — up to 70% — on cleaning data.

In this section, we'll guide you through these initial steps of data cleaning and preprocessing in Python, starting from importing the most popular libraries to actual encoding of features.

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

Step 1. Loading the data set

Importing libraries

The absolutely first thing you need to do is to import libraries for data preprocessing. There are lots of libraries available, but the most popular and important Python libraries for working on data are Numpy, Matplotlib, and Pandas. Numpy is the library used for all mathematical things. Pandas is the best tool available for importing and managing datasets. Matplotlib (Matplotlib.pyplot) is the library to make charts.

To make it easier for future use, you can import these libraries with a shortcut alias:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Loading data into pandas

Once you downloaded your data set and named it as a .csv file, you need to load it into a pandas DataFrame to explore it and perform some basic cleaning tasks removing information you don't need that will make data processing slower.
Usually, such tasks include:

- Removing the first line: it contains extraneous text instead of the column titles. This text prevents the data set from being parsed properly by the pandas library:

```
my_dataset = pd.read_csv('data/my_dataset.csv', skiprows=1,
low_memory=False)
```

- Removing columns with text explanations that we won't need, url columns and other unnecessary columns:

```
my_dataset = my_dataset.drop(['url'],axis=1)
```

- Removing all columns with only one value, or have more than 50% missing values to work faster (if your data set is large enough that it will still be meaningful):

```
my_dataset = my_dataset.dropna(thresh=half_count,axis=1)
```

It's also a good practice to name the filtered data set differently to keep it separate from the raw data. This makes sure you still have the original data in case you need to go back to it.

Step 2. Exploring the data set

Understanding the data

Now you have got your data set up, but you still should spend some time exploring it and understanding what feature each column represents. Such manual review of the data set is important, to avoid mistakes in the data analysis and the modelling process.
To make the process easier, you can create a DataFrame with the names of the columns, data types, the first row's values, and description from the data dictionary.
As you explore the features, you can pay attention to any column that:

- is formatted poorly,
- requires more data or a lot of pre-processing to turn into useful a feature, or
- contains redundant information,

since these things can hurt your analysis if handled incorrectly.
You should also pay attention to data leakage, which can cause the model to overfit. This is because the model will be also learning from features that won't be available when we're using it to make predictions. We need to be sure our model is trained using only the data it would have at the point of a loan application.

Deciding on a target column

With a filtered data set explored, you need to create a matrix of dependent variables and a vector of independent variables. At first you should decide on the appropriate column to use as a target column for modelling based on the question you want to answer. For example, if you want to predict the development of cancer, or the chance the credit will be approved, you need to find a column with the status of the disease or loan granting ad use it as the target column.
For example, if the target column is the last one, you can create the matrix of dependent variables by typing:

```
X = dataset.iloc[:, :-1].values
```

That first colon (:) means that we want to take all the lines in our dataset. : -1 means that we want to take all of the columns of data except the last one. The `.values` on the end means that we want all of the values.
To have a vector of independent variables with only the data from the last column, you can type

```
y = dataset.iloc[:, -1].values
```

Step 3. Preparing the Features for Machine Learning

Finally, it’s time to do the preparatory work to feed the features for ML algorithms. To clean the data set, you need to handle missing values and categorical features, because the mathematics underlying most machine learning models assumes that the data is numerical and contains no missing values. Moreover, the scikit-learn library returns an error if you try to train a model like linear regression and logistic regression using data that contain missing or non-numeric values.

Dealing with Missing Values

Missing data is perhaps the most common trait of unclean data. These values usually take the form of NaN or None.
here are several causes of missing values: sometimes values are missing because they do not exist, or because of improper collection of data or poor data entry. For example, if someone is under age, and the question applies to people over 18, then the question will contain a missing value. In such cases, it would be wrong to fill in a value for that question.
There are several ways to fill up missing values:

- you can remove the lines with the data if you have your data set is big enough and the percentage of missing values is high (over 50%, for example);
- you can fill all null variables with 0 is dealing with numerical values;
- you can use the Imputer class from the scikit-learn library to fill in missing values with the data’s (mean, median, most_frequent)
- you can also decide to fill up missing values with whatever value comes directly after it in the same column.

These decisions depend on the type of data, what you want to do with the data, and the cause of values missing. In reality, just because something is popular doesn’t necessarily make it the right choice. The most common strategy is to use the mean value, but depending on your data you may come up with a totally different approach.

Handling categorical data

Machine learning uses only numeric values (float or int data type). However, data sets often contain the object data type than needs to be transformed into numeric. In most cases, categorical values are discrete and can be encoded as dummy variables, assigning a number for each category. The simplest way is to use One Hot Encoder, specifying the index of the column you want to work on:

```
from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder(categorical_features = [0])
X = onehotencoder.fit_transform(X).toarray()
```

Dealing with inconsistent data entry

Inconsistency occurs, for example, when there are different unique values in a column which are meant to be the same. You can think of different approaches to capitalization, simple misprints and inconsistent formats to form an idea. One of the ways to remove data inconsistencies is by to remove whitespaces before or after entry names and by converting all cases to lower cases.
If there is a large number of inconsistent unique entries, however, it is impossible to manually check for the closest matches. You can use the Fuzzy Wuzzy package to identify which strings are most likely to be the same. It takes in two strings and returns a ratio. The closer the ratio is to 100, the more likely you will unify the strings.

Handling Dates and Times

A specific type of data inconsistency is inconsistent format of dates, such as dd/mm/yy and mm/dd/yy in the same columns. Your date values might not be in the right data type, and this will not allow you effectively perform manipulations and get insight from it. This time you can use the datetime package to fix the type of the date.

Scaling and Normalization

Scaling is important if you need to specify that a change in one quantity is not equal to another change in another. With the help of scaling you ensure that just because some features are big they won’t be used as a main predictor. For example, if you use the age and the salary of a person in prediction, some algorithms will pay attention to the salary more because it is bigger, which does not make any sense.
Normalization involves transforming or converting your dataset into a normal distribution. Some algorithms like SVM converge far faster on normalized data, so it makes sense to normalize your data to get better results.
There are many ways to perform feature scaling. In a nutshell, we put all of our features into the same scale so that none are dominated by another. For example, you can use the StandardScaler class from the sklearn.preprocessing package to fit and transform your data set:

from sklearn.preprocessing import StandardScaler

```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

#As you don't need to fit it to your test set, you can just apply
transformation.

sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)
```

Save to CSV

To be sure that you still have the raw data, it is a good practice to store the final output of each section or stage of your workflow in a separate csv file. In this way, you'll be able to make changes in your data processing flow without having to recalculate everything.
As we did previously, you can store your DataFrame as a .csv using the pandas to_csv() function.

```
my_dataset.to_csv("processed_data/cleaned_dataset.csv",index=False)
```

Data wrangling

Data wrangling, sometimes referred to as data munging, is the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics. A data wrangler is a person who performs these transformation operations.

This may include further munging, data visualization, data aggregation, training a statistical model, as well as many other potential uses. Data munging as a process typically follows a set of general steps which begin with extracting the data in a raw form from the data source, "munging" the raw data using algorithms (e.g. sorting) or parsing the data into predefined data structures, and finally depositing the resulting content into a data sink for storage and future use.

The data transformations are typically applied to distinct entities (e.g. fields, rows, columns, data values etc.) within a data set, and could include such actions as extractions, parsing, joining, standardizing, augmenting, cleansing, consolidating and filtering to create desired wrangling outputs that can be leveraged downstream.

The recipients could be individuals, such as data architects or data scientists who will investigate the data further, business users who will consume the data directly in reports, or systems that will further process the data and write it into targets such as data warehouses, data lakes or downstream applications.

Plotting and Visualization

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.
Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.
To get a little overview here are a few popular plotting libraries:

- Matplotlib: low level, provides lots of freedom
- Pandas Visualization: easy to use interface, built on Matplotlib
- Seaborn: high-level interface, great default styles
- ggplot: based on R’s ggplot2, uses Grammar of Graphics
- Plotly: can create interactive plots

In this article, we will learn how to create basic plots using Matplotlib, Pandas visualization and Seaborn as well as how to use some specific features of each library. This article will focus on the syntax and not on interpreting the graphs, which I will cover in another blog post.
In further articles, I will go over interactive plotting tools like Plotly, which is built on D3 and can also be used with JavaScript.

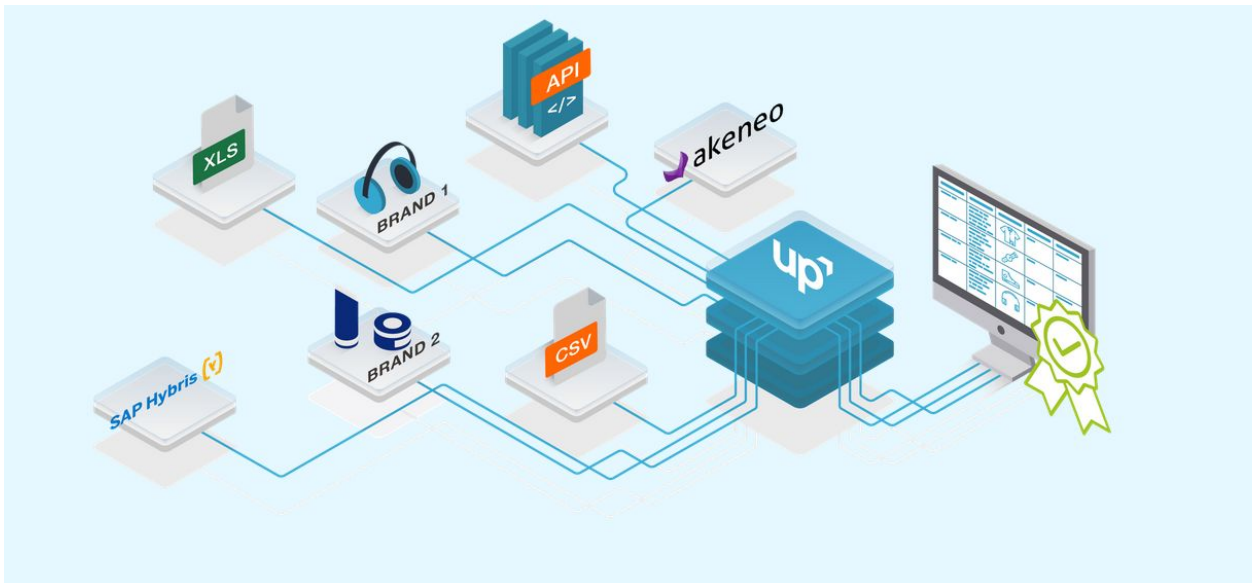
Data Aggregation

Both marketers and agencies who have ever tried to manage marketing campaigns across many different platforms like Facebook, Adwords, and more know how overwhelming it can be at times to aggregate all the data from several places.
The entire process of data aggregation can take over 20 hours every week. Luckily, there is a solution to this problem. Platforms like improvado are now available to help marketers save time and effort.
Platforms like these automate the data aggregation process, letting you create the reports you need to view all of your campaign data in one place, in real time. Let's take a closer look at what data aggregation is, how it can help your company, and what some of the best data aggregation platforms are.

What is Data Aggregation?

Data aggregation is any process in which data is brought together and conveyed in a summary form. It is typically used prior to the performance of a statistical analysis.
The information drawn from the data aggregation and statistical analysis can then be used to tell you all kinds of information about the data you are looking at.
In marketing, data aggregation usually comes from your marketing campaigns and the different channels you use to market to your customers.
You can aggregate your data from one specific campaign, looking at how it performed over time and with particular cohorts.
Ideally, though, you are aggregating the data from each specific campaign to compare them to each other - one grand data aggregation that tells you how your product is being received across channels, populations, and cohorts.
Doing so can take a lot of time and effort. Luckily, there are now tons of software available that can do the work for you, so that you aren't wasting half of your week simply picking through the data.

Data Aggregation in the Context of Middleware



When it comes to middleware for marketing analytics, there are three different activities and functions.

For short, we refer to them as ETV (Extract, Transform, Visualize). Together, this is the workflow of extracting and preparing data from SaaS applications for analysis.

For each of these three steps there is a software layer, meaning there are companies whose sole focus is to help marketers during each of these steps.

1. Extract - Data extraction layer
2. Transform - Data preparation layer
3. Visualize/Analyze - Visualization and analytics layer

Here at Improvado, we believe it's important to provide marketers the freedom to work with tools that allow them to integrate and analyze their data without involving engineers.
In this article, we are discussing the extraction phase of middleware for analytics - taking all of the data that is stored in your many marketing databases and funneling it into one place for analysis.

Data Visualization

Pandas Visualization

Pandas is an open source high-performance, easy-to-use library providing data structures, such as dataframes, and data analysis tools like the visualization tools we will use in this article.
Pandas Visualization makes it really easy to create plots out of a pandas dataframe and series. It also has a higher level API than Matplotlib and therefore we need less code for the same results.
Pandas can be installed using either pip or conda.

```
pip install pandas
or
conda install pandas
```

Scatter Plot

To create a scatter plot in Pandas we can call `<dataset>.plot.scatter()` and pass it two arguments, the name of the x-column as well as the name of the y-column. Optionally we can also pass it a title.

```
iris.plot.scatter(x='sepal_length', y='sepal_width', title='Iris Dataset')
```

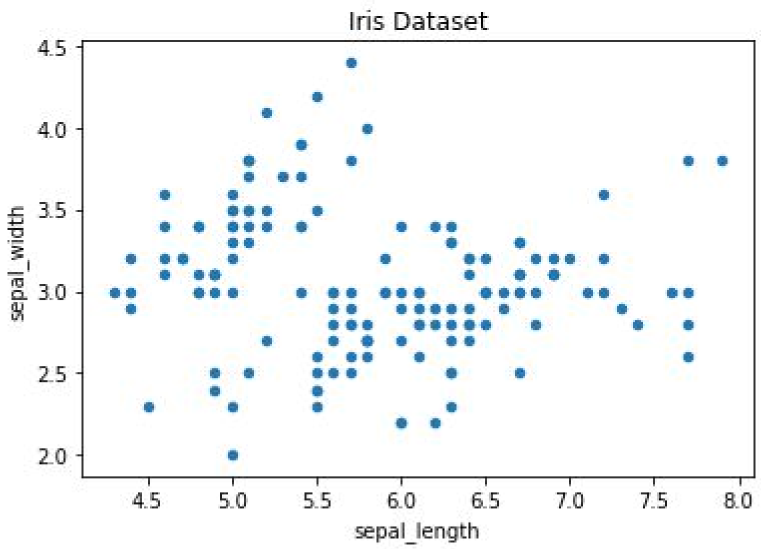


Figure: Scatter Plot

As you can see in the image it is automatically setting the x and y label to the column names.

Line Chart

To create a line-chart in Pandas we can call `<dataframe>.plot.line().` Whilst in Matplotlib we needed to loop-through each column we wanted to plot, in Pandas we don't need to do this because it automatically plots all available numeric columns (at least if we don't specify a specific column/s).

```
iris.drop(['class'], axis=1).plot.line(title='Iris Dataset')
```

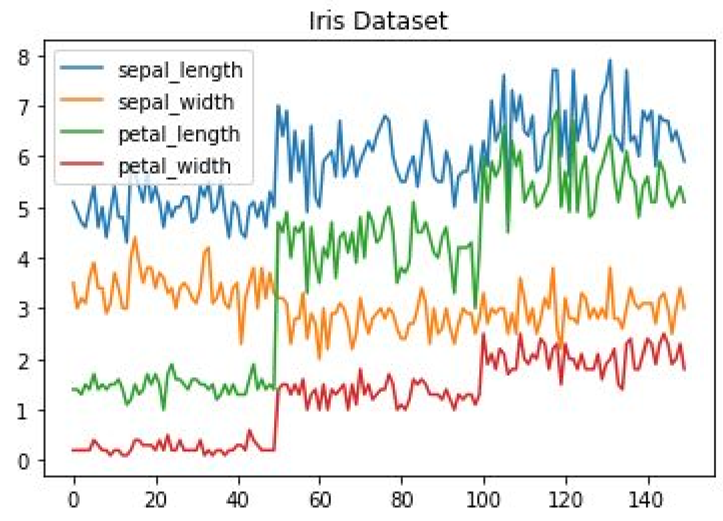


Figure: Line Chart

If we have more than one feature Pandas automatically creates a legend for us, as can be seen in the image above.

Histogram

In Pandas, we can create a Histogram with the `plot.hist` method. There aren't any required arguments but we can optionally pass some like the bin size.

```
wine_reviews['points'].plot.hist()
```

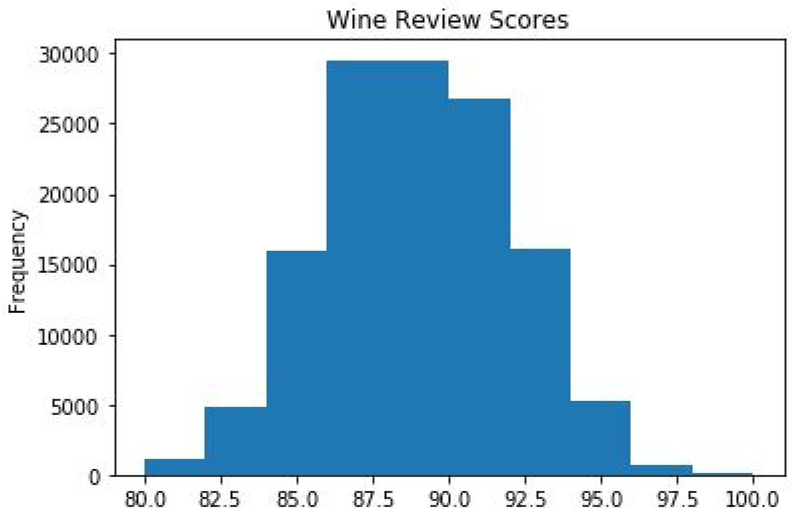


Figure: Histogram

It's also really easy to create multiple histograms.

```
iris.plot.hist(subplots=True, layout=(2,2), figsize=(10, 10), bins=20)
```

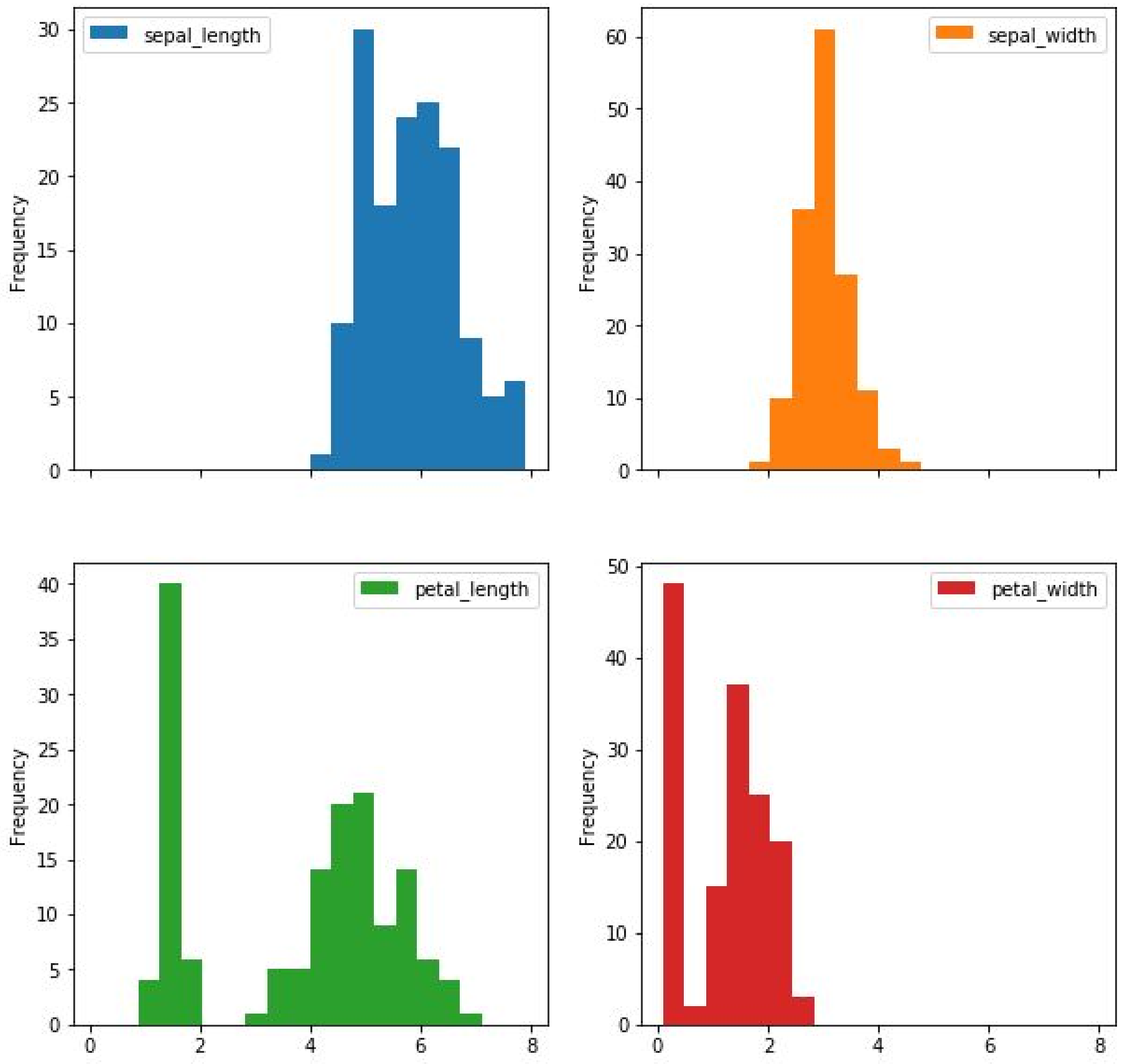


Figure 12: Multiple Histograms

The `subplots` argument specifies that we want a separate plot for each feature and the `layout` specifies the number of plots per row and column.

Bar Chart

To plot a bar-chart we can use the `plot.bar()` method, but before we can call this we need to get our data. For this we will first count the occurrences using the `value_count()` method and then sort the occurrences from smallest to largest using the `sort_index()` method.

```
wine_reviews['points'].value_counts().sort_index().plot.bar()
```

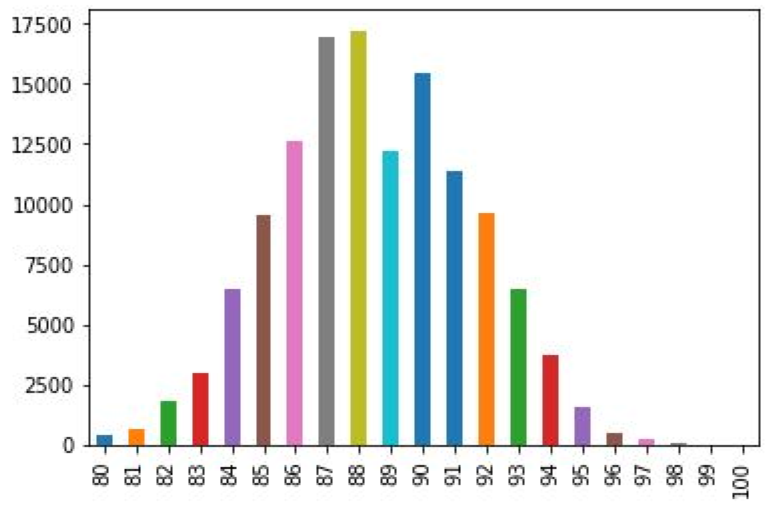


Figure: Vertical Bar-Chart

It’s also really simple to make a horizontal bar-chart using the plot.barh() method.

```
wine_reviews['points'].value_counts().sort_index().plot.barh()
```

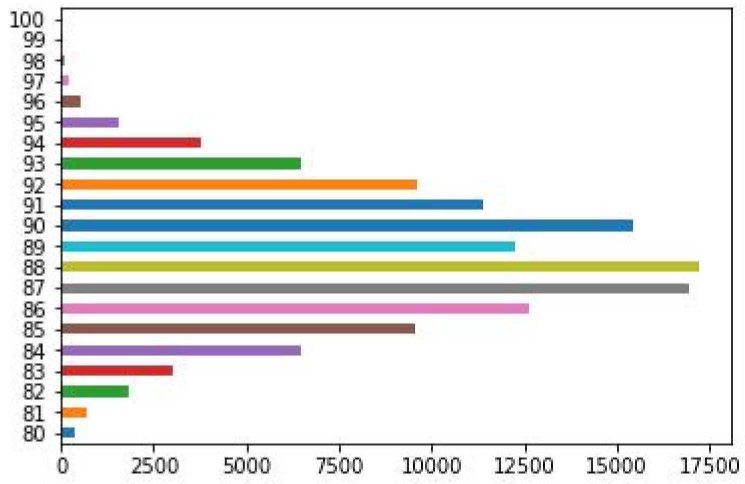


Figure: Horizontal Bar-Chart

We can also plot other data then the number of occurrences.

```
wine_reviews.groupby("country").price.mean().sort_values(ascending=False)[:
5].plot.bar()
```

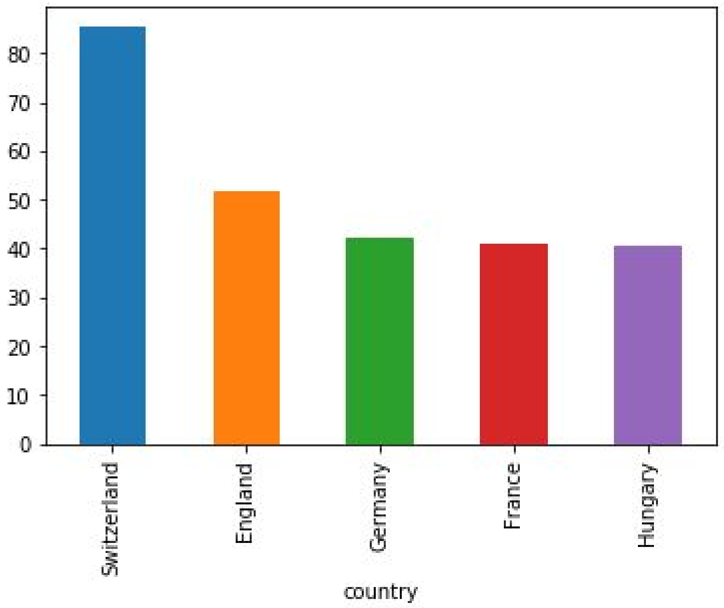
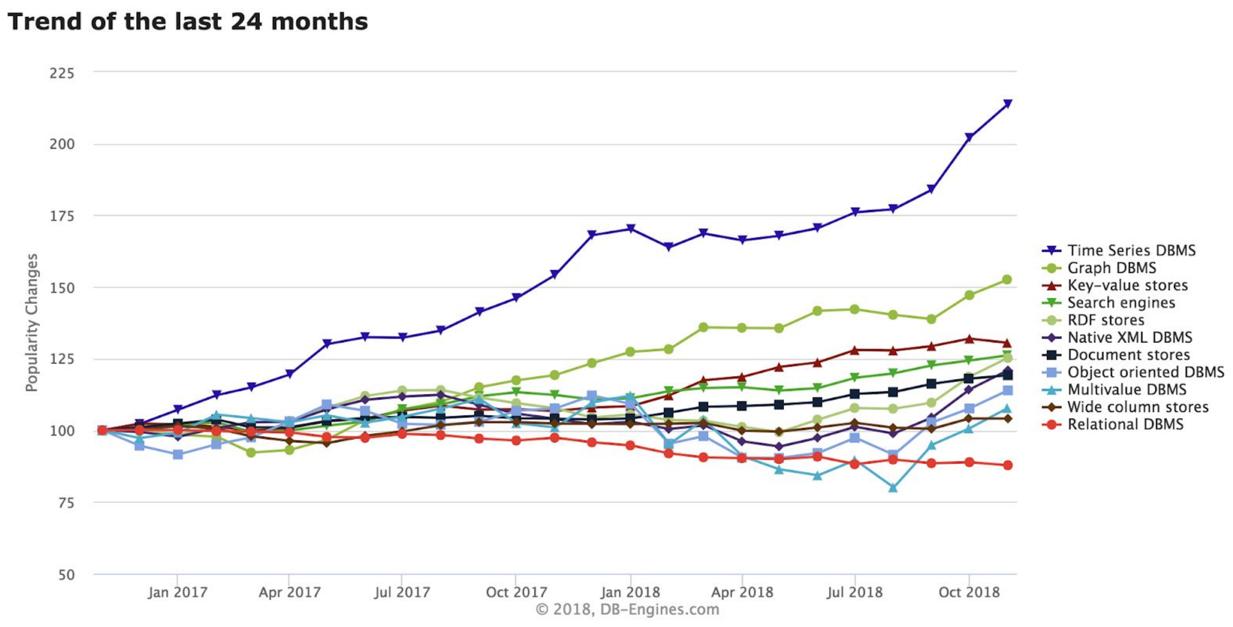


Figure: Countries with the most expensive wine(on average)

In the example above we grouped the data by country and then took the mean of the wine prices, ordered it, and plotted the 5 countries with the highest average wine price.

Time series

A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average.



Time series are very frequently plotted via line charts. Time series are used in statistics, signal processing, pattern recognition, econometrics, mathematical finance, weather forecasting, earthquake prediction, electroencephalography, control engineering, astronomy, communications engineering, and largely in any domain of applied science and engineering which involves temporal measurements.

Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. While regression analysis is often employed in such a way as to test theories that the current values of one or more independent time series affect the current value of another time series, this type of analysis of time series is not called "time series analysis", which focuses on comparing values of a single time series or multiple dependent time series at different points in time. Interrupted time series analysis is the analysis of interventions on a single time series.

Time series data have a natural temporal ordering. This makes time series analysis distinct from cross-sectional studies, in which there is no natural ordering of the observations (e.g. explaining people's wages by reference to their respective education levels, where the individuals' data could be entered in any order). Time series analysis is also distinct from spatial data analysis where the observations typically relate to geographical locations (e.g. accounting for house prices by the location as well as the intrinsic characteristics of the houses). A stochastic model for a time series will generally reflect the fact that observations close together in time will be more closely related than observations further apart. In addition, time series models will often make use of the natural one-way ordering of time so that values for a given period will be expressed as deriving in some way from past values, rather than from future values (see time reversibility.)

Time series analysis can be applied to real-valued, continuous data, discrete numeric data, or discrete symbolic data (i.e. sequences of characters, such as letters and words in the English language).

Forecasting

Forecasting has fascinated people for thousands of years, sometimes being considered a sign of divine inspiration, and sometimes being seen as a criminal activity.

One hundred years later, in ancient Babylon, forecasters would foretell the future based on the distribution of maggots in a rotten sheep's liver. By 300 BC, people wanting forecasts would journey to Delphi in Greece to consult the Oracle, who would provide her predictions while intoxicated by ethylene vapours. Forecasters had a tougher time under the emperor Constantine, who issued a decree in AD357 forbidding anyone "to consult a soothsayer, a mathematician, or a forecaster "May curiosity to foretell the future be silenced forever." A similar ban on forecasting occurred in England in 1736 when it became an offence to defraud by charging money for predictions. The punishment was three months' imprisonment with hard labour!

The varying fortunes of forecasters arise because good forecasts can seem almost magical, while bad forecasts may be dangerous. Consider the following famous predictions about computing.

I think there is a world market for maybe five computers. (Chairman of IBM, 1943)
Computers in the future may weigh no more than 1.5 tons. (Popular Mechanics, 1949)
There is no reason anyone would want a computer in their home. (President, DEC, 1977)
The last of these was made only three years before IBM produced the first personal computer. Not surprisingly, you can no longer buy a DEC computer. Forecasting is obviously a difficult activity, and businesses that do it well have a big advantage over those whose forecasts fail.

What can be forecast?

Forecasting is required in many situations: deciding whether to build another power generation plant in the next five years requires forecasts of future demand; scheduling staff in a call centre next week requires forecasts of call volumes; stocking an inventory requires forecasts of stock requirements. Forecasts can be required several years in advance (for the case of capital investments), or only a few minutes beforehand (for telecommunication routing). Whatever the circumstances or time horizons involved, forecasting is an important aid to effective and efficient planning.

Some things are easier to forecast than others. The time of the sunrise tomorrow morning can be forecast precisely. On the other hand, tomorrow's lotto numbers cannot be forecast with any accuracy. The predictability of an event or a quantity depends on several factors including:

1. how well we understand the factors that contribute to it;
2. how much data is available;
3. whether the forecasts can affect the thing we are trying to forecast.

For example, forecasts of electricity demand can be highly accurate because all three conditions are usually satisfied. We have a good idea of the contributing factors: electricity demand is driven largely by temperatures, with smaller effects for calendar variation such as holidays, and economic conditions. Provided there is a sufficient history of data on electricity demand and weather conditions, and we have the skills to develop a good model linking electricity demand and the key driver variables, the forecasts can be remarkably accurate.

On the other hand, when forecasting currency exchange rates, only one of the conditions is satisfied: there is plenty of available data. However, we have a limited understanding of the factors that affect exchange rates, and forecasts of the exchange rate have a direct effect on the rates themselves. If there are well-publicised forecasts that the exchange rate will increase, then people will immediately adjust the price they are willing to pay and so the forecasts are self-fulfilling. In a sense, the exchange rates become their own forecasts. This is an example of the "efficient market hypothesis". Consequently, forecasting whether the exchange rate will rise or fall tomorrow is about as predictable as forecasting whether a tossed coin will come down as a head or a tail. In both situations, you will be correct about 50% of the time, whatever you forecast. In situations like this, forecasters need to be aware of their own limitations, and not claim more than is possible.

Often in forecasting, a key step is knowing when something can be forecast accurately, and when forecasts will be no better than tossing a coin. Good forecasts capture the genuine patterns and relationships which exist in the historical data, but do not replicate past events that will not occur again.

Many people wrongly assume that forecasts are not possible in a changing environment. Every environment is changing, and a good forecasting model captures the way in which things are changing. Forecasts rarely assume that the environment is unchanging. What is normally assumed is that the way in which the environment is changing will continue into the future. That is, a highly volatile environment will continue to be highly volatile; a business with fluctuating sales will continue to have fluctuating sales; and an economy that has gone through booms and busts will continue to go through booms and busts. A forecasting model is intended to capture the way things move, not just where things are. As Abraham Lincoln said, "If we could first know where we are and whither we are tending, we could better judge what to do and how to do it". Forecasting situations vary widely in their time horizons, factors determining actual outcomes, types of data patterns, and many other aspects. Forecasting methods can be simple, such as using the most recent observation as a forecast (which is called the naïve method), or highly complex, such as neural nets and econometric systems of simultaneous equations. Sometimes, there will be no data available at all. For example, we may wish to forecast the sales of a new product in its first year, but there are obviously no data to work with. In situations like this, we use judgmental forecasting. The choice of method depends on what data are available and the predictability of the quantity to be forecast.

Forecasting, planning and goals

Forecasting is a common statistical task in business, where it helps to inform decisions about the scheduling of production, transportation and personnel, and provides a guide to long-term strategic planning. However, business forecasting is often done poorly, and is frequently confused with planning and goals. They are three different things.

Forecasting
is about predicting the future as accurately as possible, given all of the information available, including historical data and knowledge of any future events that might impact the forecasts.

Goals
are what you would like to have happen. Goals should be linked to forecasts and plans, but this does not always occur. Too often, goals are set without any plan for how to achieve them, and no forecasts for whether they are realistic.

Planning
is a response to forecasts and goals. Planning involves determining the appropriate actions that are required to make your forecasts match your goals.

Forecasting should be an integral part of the decision-making activities of management, as it can play an important role in many areas of a company. Modern organisations require short-term, medium-term and long-term forecasts, depending on the specific application.

Short-term forecasts
are needed for the scheduling of personnel, production and transportation. As part of the scheduling process, forecasts of demand are often also required.
Medium-term forecasts
are needed to determine future resource requirements, in order to purchase raw materials, hire personnel, or buy machinery and equipment.

Long-term forecasts

are used in strategic planning. Such decisions must take account of market opportunities, environmental factors and internal resources.

An organisation needs to develop a forecasting system that involves several approaches to predicting uncertain events. Such forecasting systems require the development of expertise in identifying forecasting problems, applying a range of forecasting methods, selecting appropriate methods for each problem, and evaluating and refining forecasting methods over time. It is also important to have strong organisational support for the use of formal forecasting methods if they are to be used successfully.

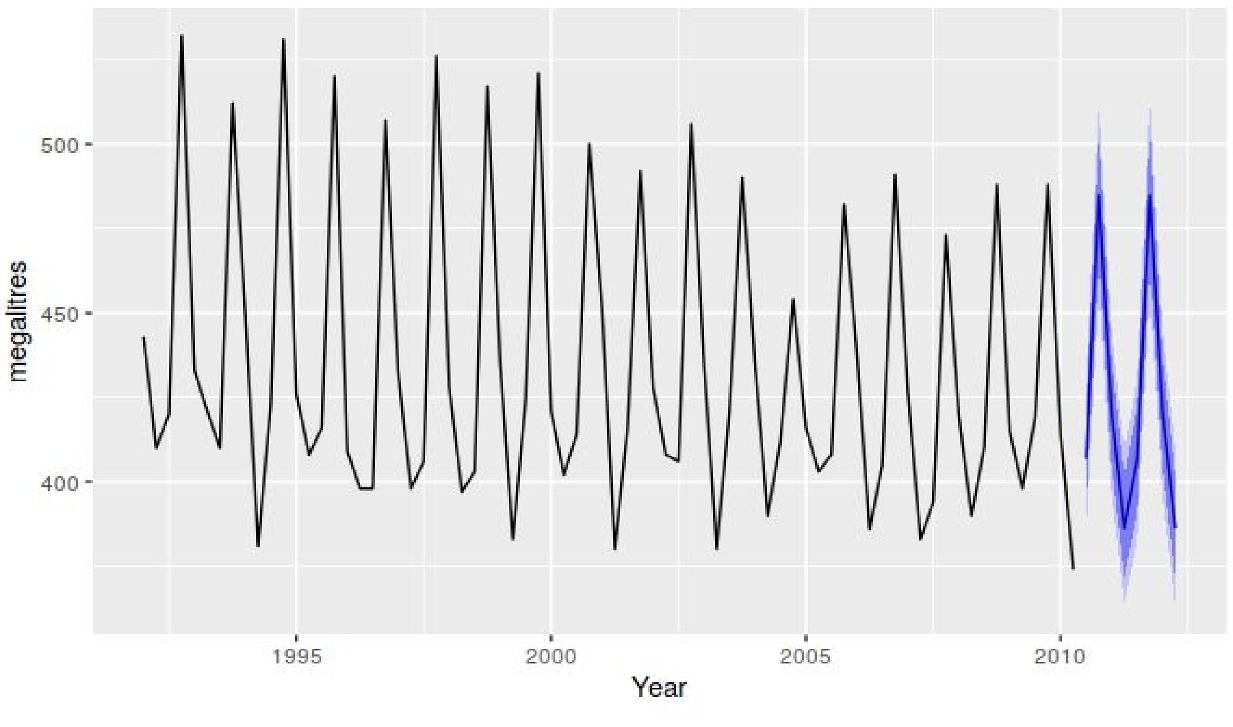
Time series forecasting

Examples of time series data include:

- Daily IBM stock prices
- Monthly rainfall
- Quarterly sales results for Amazon
- Annual Google profits

Anything that is observed sequentially over time is a time series. In this study, we will only consider time series that are observed at regular intervals of time (e.g., hourly, daily, weekly, monthly, quarterly, annually). Irregularly spaced time series can also occur, but are beyond the scope of this study.

When forecasting time series data, the aim is to estimate how the sequence of observations will continue into the future. Next figure shows the quarterly Australian beer production from 1992 to the second quarter of 2010.



The blue lines show forecasts for the next two years. Notice how the forecasts have captured the seasonal pattern seen in the historical data and replicated it for the next two years. The dark shaded region shows 80% prediction intervals. That is, each future value is expected to lie in the dark shaded region with a probability of 80%. The light shaded region shows 95% prediction intervals. These prediction intervals are a useful way of displaying the uncertainty in forecasts. In this case the forecasts are expected to be accurate, and hence the prediction intervals are quite narrow.

The simplest time series forecasting methods use only information on the variable to be forecast, and make no attempt to discover the factors that affect its behaviour. Therefore they will extrapolate trend and seasonal patterns, but they ignore all other information such as marketing initiatives, competitor activity, changes in economic conditions, and so on.
Time series models used for forecasting include decomposition models, exponential smoothing models and ARIMA models.

Predictor variables and time series forecasting

Predictor variables are often useful in time series forecasting. For example, suppose we wish to forecast the hourly electricity demand (ED) of a hot region during the summer period. A model with predictor variables might be of the form

$$\begin{aligned}\text{ED}=f(&\text{current temperature, strength of economy, population,}\\ &\text{time of day, day of week, error}).\end{aligned}$$

The relationship is not exact — there will always be changes in electricity demand that cannot be accounted for by the predictor variables. The “error” term on the right allows for random variation and the effects of relevant variables that are not included in the model. We call this an explanatory model because it helps explain what causes the variation in electricity demand.

Because the electricity demand data form a time series, we could also use a time series model for forecasting. In this case, a suitable time series forecasting equation is of the form

$$\text{ED}_{t+1} = f(\text{ED}_t, \text{ED}_{t-1}, \text{ED}_{t-2}, \text{ED}_{t-3}, \dots, \text{error}),$$

where t is the present hour, $t+1$ is the next hour, $t-1$ is the previous hour, $t-2$ is two hours ago, and so on. Here, prediction of the future is based on past values of a variable, but not on external variables which may affect the system. Again, the “error” term on the right allows for random variation and the effects of relevant variables that are not included in the model.

There is also a third type of model which combines the features of the above two models. For example, it might be given by

$$\mathrm{ED}_{t+1} = f(\mathrm{ED}_t, \text{current temperature, time of day, day of week, error}).$$

These types of “mixed models” have been given various names in different disciplines. They are known as dynamic regression models, panel data models, longitudinal models, transfer function models, and linear system models (assuming that f is linear).

An explanatory model is useful because it incorporates information about other variables, rather than only historical values of the variable to be forecast. However, there are several reasons a forecaster might select a time series model rather than an explanatory or mixed model. First, the system may not be understood, and even if it was understood it may be extremely difficult to measure the relationships that are assumed to govern its behaviour. Second, it is necessary to know or forecast the future values of the various predictors in order to be able to forecast the variable of interest, and this may be too difficult. Third, the main concern may be only to predict what will happen, not to know why it happens. Finally, the time series model may give more accurate forecasts than an explanatory or mixed model.

The model to be used in forecasting depends on the resources and data available, the accuracy of the competing models, and the way in which the forecasting model is to be used.

Some case studies

The following four cases are from our consulting practice and demonstrate different types of forecasting situations and the associated problems that often arise.

Case 1

The client was a large company manufacturing disposable tableware such as napkins and paper plates. They needed forecasts of each of hundreds of items every month. The time series data showed a range of patterns, some with trends, some seasonal, and some with neither. At the time, they were using their own software, written in-house, but it often produced forecasts that did not seem sensible. The methods that were being used were the following:

1. average of the last 12 months data;
2. average of the last 6 months data;
3. prediction from a straight line regression over the last 12 months;
4. prediction from a straight line regression over the last 6 months;
5. prediction obtained by a straight line through the last observation with slope equal to the average slope of the lines connecting last year’s and this year’s values;
6. prediction obtained by a straight line through the last observation with slope equal to the average slope of the lines connecting last year’s and this year’s values, where the average is taken only over the last 6 months.

They required us to tell them what was going wrong and to modify the software to provide more accurate forecasts. The software was written in COBOL, making it difficult to do any sophisticated numerical computation.

Case 2

In this case, the client was the Australian federal government, who needed to forecast the annual budget for the Pharmaceutical Benefit Scheme (PBS). The PBS provides a subsidy for many pharmaceutical products sold in Australia, and the expenditure depends on what people purchase during the year. The total expenditure was around A$7 billion in 2009, and had been underestimated by nearly $1 billion in each of the two years before we were asked to assist in developing a more accurate forecasting approach.

In order to forecast the total expenditure, it is necessary to forecast the sales volumes of hundreds of groups of pharmaceutical products using monthly data. Almost all of the groups have trends and seasonal patterns. The sales volumes for many groups have sudden jumps up or down due to changes in what drugs are subsidised. The expenditures for many groups also have sudden changes due to cheaper competitor drugs becoming available.

Thus we needed to find a forecasting method that allowed for trend and seasonality if they were present, and at the same time was robust to sudden changes in the underlying patterns. It also needed to be able to be applied automatically to a large number of time series.

Case 3

A large car fleet company asked us to help them forecast vehicle re-sale values. They purchase new vehicles, lease them out for three years, and then sell them. Better forecasts of vehicle sales values would mean better control of profits; understanding what affects resale values may allow leasing and sales policies to be developed in order to maximise profits.

At the time, the resale values were being forecast by a group of specialists. Unfortunately, they saw any statistical model as a threat to their jobs, and were uncooperative in providing information. Nevertheless, the company provided a large amount of data on previous vehicles and their eventual resale values.

Case 4

In this project, we needed to develop a model for forecasting weekly air passenger traffic on major domestic routes for one of Australia's leading airlines. The company required forecasts of passenger numbers for each major domestic route and for each class of passenger (economy class, business class and first class). The company provided weekly traffic data from the previous six years.

Air passenger numbers are affected by school holidays, major sporting events, advertising campaigns, competition behaviour, etc. School holidays often do not coincide in different

Australian cities, and sporting events sometimes move from one city to another. During the period of the historical data, there was a major pilots' strike during which there was no traffic for several months. A new cut-price airline also launched and folded. Towards the end of the historical data, the airline had trialled a redistribution of some economy class seats to business class, and some business class seats to first class. After several months, however, the seat classifications reverted to the original distribution.

The basic steps in a forecasting task

A forecasting task usually involves five basic steps.

Step 1: Problem definition.

Often this is the most difficult part of forecasting. Defining the problem carefully requires an understanding of the way the forecasts will be used, who requires the forecasts, and how the forecasting function fits within the organisation requiring the forecasts. A forecaster needs to spend time talking to everyone who will be involved in collecting data, maintaining databases, and using the forecasts for future planning.

Step 2: Gathering information.

There are always at least two kinds of information required: (a) statistical data, and (b) the accumulated expertise of the people who collect the data and use the forecasts. Often, it will be difficult to obtain enough historical data to be able to fit a good statistical model. Occasionally, old data will be less useful due to structural changes in the system being forecast; then we may choose to use only the most recent data. However, remember that good statistical models will handle evolutionary changes in the system; don't throw away good data unnecessarily.

Step 3: Preliminary (exploratory) analysis.

Always start by graphing the data. Are there consistent patterns? Is there a significant trend? Is seasonality important? Is there evidence of the presence of business cycles? Are there any outliers in the data that need to be explained by those with expert knowledge? How strong are the relationships among the variables available for analysis? Various tools have been developed to help with this analysis.

Step 4: Choosing and fitting models.

The best model to use depends on the availability of historical data, the strength of relationships between the forecast variable and any explanatory variables, and the way in which the forecasts are to be used. It is common to compare two or three potential models. Each model is itself an artificial construct that is based on a set of assumptions (explicit and implicit) and usually involves one or more parameters which must be estimated using the known historical data. We will discuss regression models , exponential smoothing methods , Box-Jenkins ARIMA models, Dynamic regression models , Hierarchical forecasting , and several advanced methods including neural networks and vector autoregression.

Step 5: Using and evaluating a forecasting model.

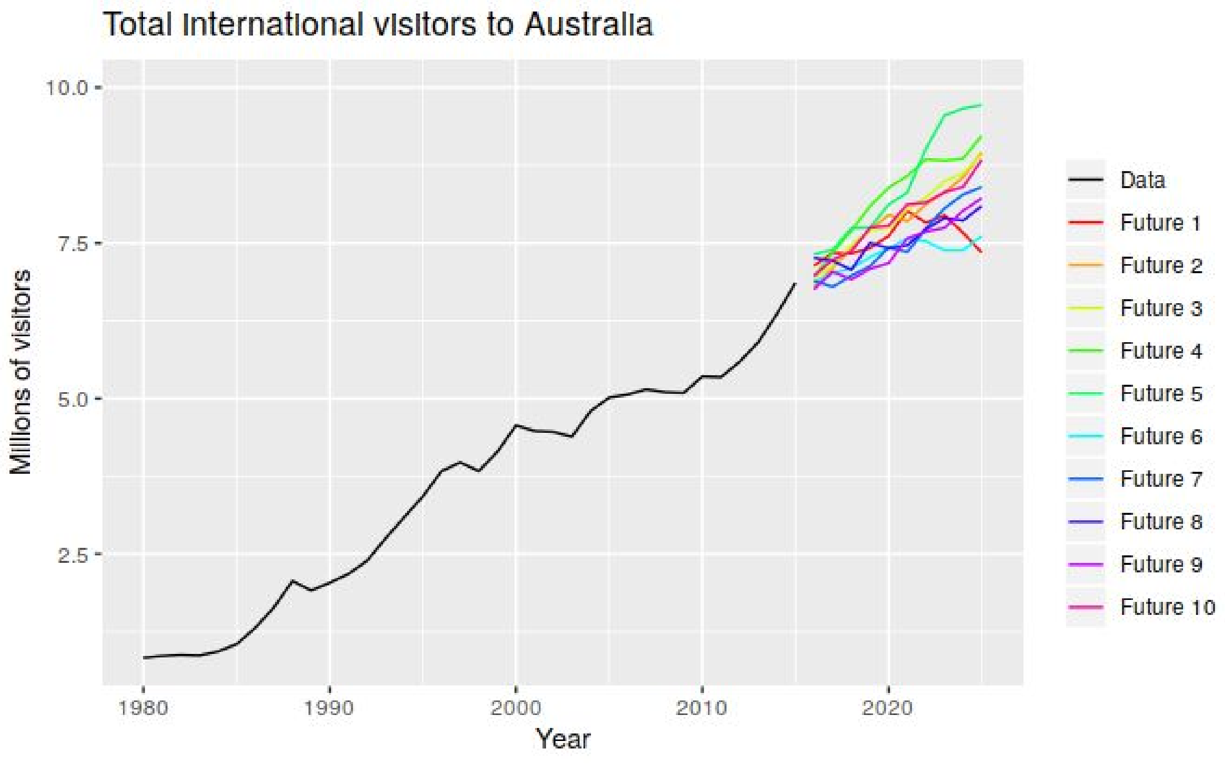
Once a model has been selected and its parameters estimated, the model is used to make forecasts. The performance of the model can only be properly evaluated after the data for the forecast period have become available. A number of methods have been developed to help in assessing the accuracy of forecasts. There are also organisational issues in using and acting on the forecasts. When using a forecasting model in practice, numerous practical issues arise such as how to handle missing values and outliers, or how to deal with short time series.

The statistical forecasting perspective

The thing we are trying to forecast is unknown (or we would not be forecasting it), and so we can think of it as a random variable. For example, the total sales for next month could take a range of possible values, and until we add up the actual sales at the end of the month, we don't know what the value will be. So until we know the sales for next month, it is a random quantity.

Because next month is relatively close, we usually have a good idea what the likely sales values could be. On the other hand, if we are forecasting the sales for the same month next year, the possible values it could take are much more variable. In most forecasting situations, the variation associated with the thing we are forecasting will shrink as the event approaches. In other words, the further ahead we forecast, the more uncertain we are.

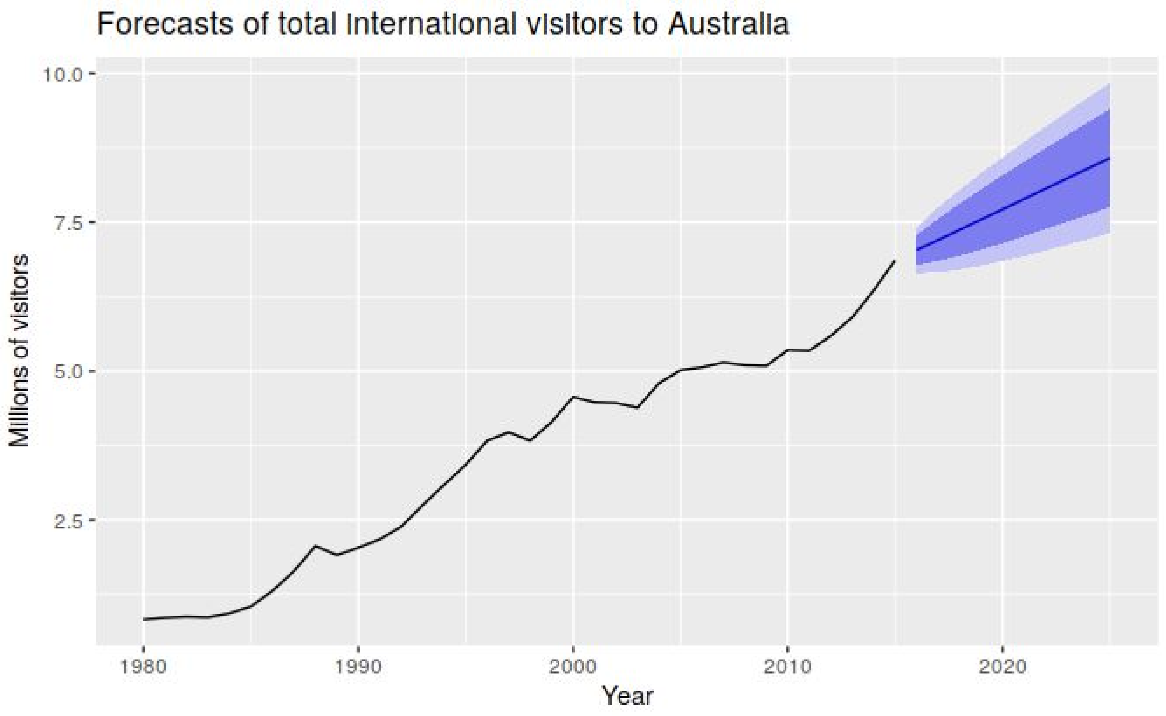
We can imagine many possible futures, each yielding a different value for the thing we wish to forecast. Plotted in black in Figure 1.2 are the total international visitors to Australia from 1980 to 2015. Also shown are ten possible futures from 2016–2025.



When we obtain a forecast, we are estimating the middle of the range of possible values the random variable could take. Often, a forecast is accompanied by a prediction interval giving a range of values the random variable could take with relatively high probability. For example, a

95% prediction interval contains a range of values which should include the actual future value with probability 95%.

Instead of plotting individual possible futures as shown in the next figure, we usually show these prediction intervals instead. The plot below shows 80% and 95% intervals for the future Australian international visitors. The blue line is the average of the possible future values, which we call the point forecasts.



We will use the subscript t for time. For example, yt will denote the observation at time t. Suppose we denote all the information we have observed as I and we want to forecast yt. We then write yt|I meaning “the random variable Yt given what we know in I”. The set of values that this random variable could take, along with their relative probabilities, is known as the “probability distribution” of yt|I. In forecasting, we call this the forecast distribution.

When we talk about the “forecast”, we usually mean the average value of the forecast distribution, and we put a “hat” over Y to show this. Thus, we write the forecast of Yt As y^t, meaning the average of the possible values that yt could take given everything we know. Occasionally, we will use y^t to refer to the median (or middle value) of the forecast distribution instead.

It is often useful to specify exactly what information we have used in calculating the forecast. Then we will write, for example, y^t|t−1 to mean the forecast of yt taking account of all previous observations (y1,…,yt−1). Similarly, y^T+h|T means the forecast of yT+h taking account of y1,…,yT (i.e., an h-step forecast taking account of all observations up to time T).

Some simple forecasting methods

Some forecasting methods are extremely simple and surprisingly effective. We will use the following four forecasting methods as benchmarks throughout this study.

Average method

Here, the forecasts of all future values are equal to the average (or “mean”) of the historical data. If we let the historical data be denoted by y1,…,yT , then we can write the forecasts as

$$\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T.$$

The notation y^T+h|T is a short-hand for the estimate of yT+h based on the data y1,…,yT.

```
meanf(y, h)
# y contains the time series
# h is the forecast horizon
```

Naïve method

For naïve forecasts, we simply set all forecasts to be the value of the last observation. That is,

$$\hat{y}_{T+h|T} = y_T.$$

This method works remarkably well for many economic and financial time series.

```
naive(y, h)
rwf(y, h) # Equivalent alternative
```

Because a naïve forecast is optimal when data follow a random walk (see Section 8.1), these are also called **random walk forecasts**.

Seasonal naïve method

A similar method is useful for highly seasonal data. In this case, we set each forecast to be equal to the last observed value from the same season of the year (e.g., the same month of the previous year). Formally, the forecast for time T+h is written as

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)},$$

where m= the seasonal period, and k is the integer part of (h−1)/m (i.e., the number of complete years in the forecast period prior to time T+h). This looks more complicated than it really is. For example, with monthly data, the forecast for all future February values is equal to the last observed February value. With quarterly data, the forecast of all future Q2 values is equal to the last observed Q2 value (where Q2 means the second quarter). Similar rules apply for other months and quarters, and for other seasonal periods.

```
snaive(y, h)
```

Drift method

A variation on the naïve method is to allow the forecasts to increase or decrease over time, where the amount of change over time (called the drift) is set to be the average change seen in the historical data. Thus the forecast for time T+h is given by

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right).$$

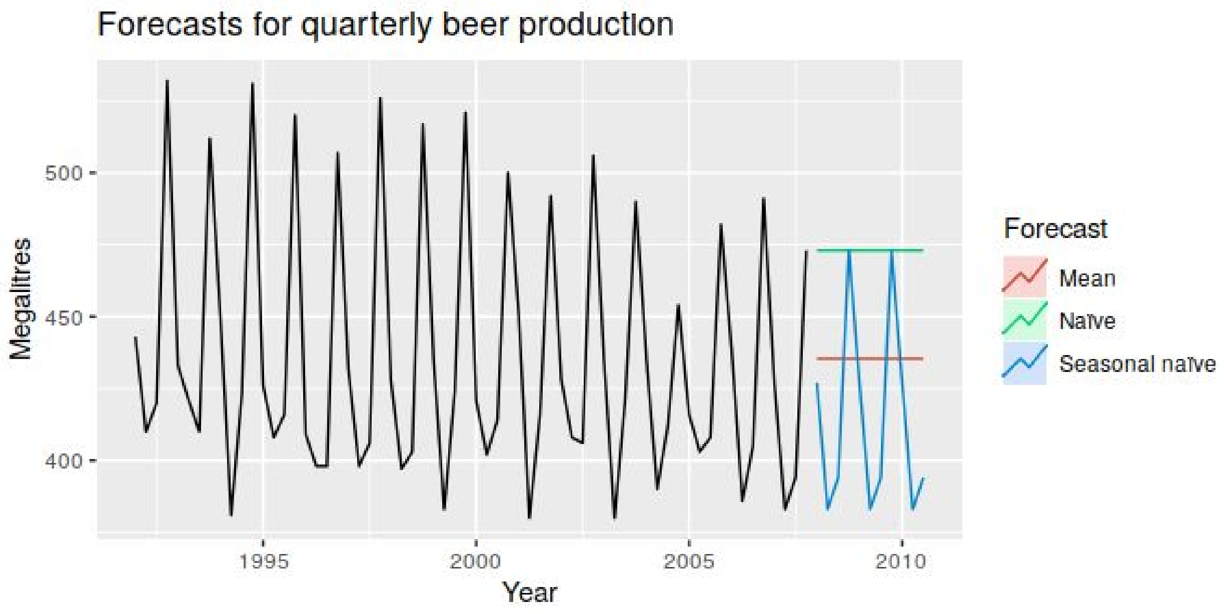
This is equivalent to drawing a line between the first and last observations, and extrapolating it into the future.

```
rwf(y, h, drift=TRUE)
```

Examples of using Average method, Naïve method, Drift method

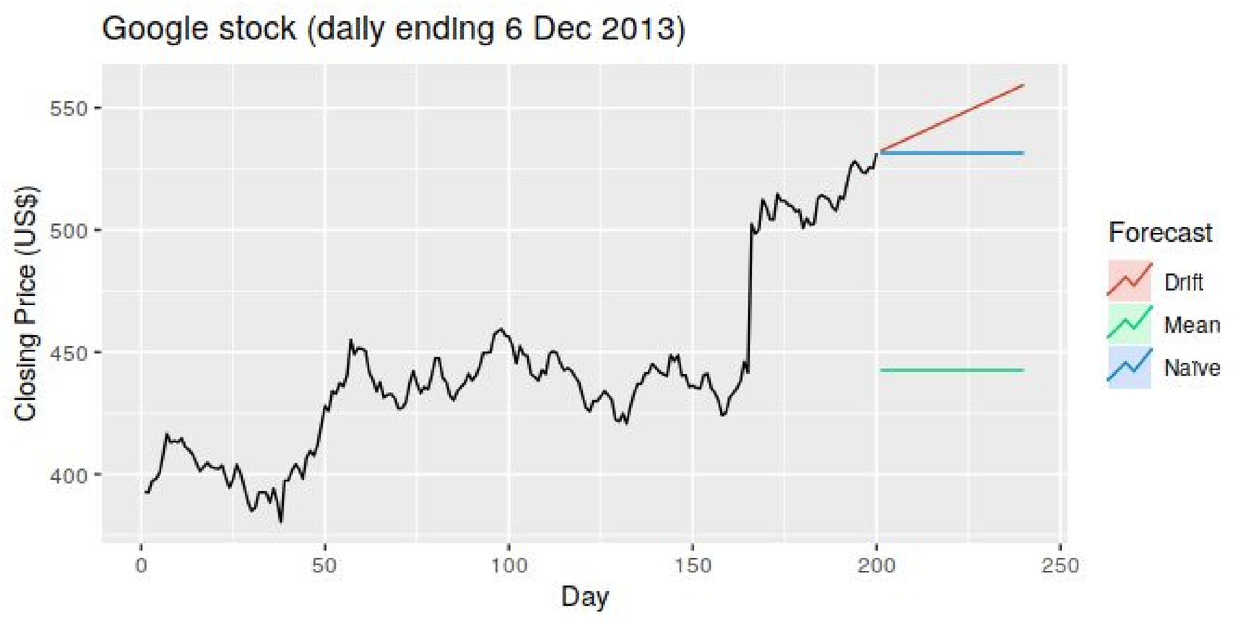
The next Figure shows the first three methods applied to the quarterly beer production data.

```r
# Set training data from 1992 to 2007
beer2 <- window(ausbeer,start=1992,end=c(2007,4))
# Plot some forecasts
autoplot(beer2) +
  autolayer(meanf(beer2, h=11),
    series="Mean", PI=FALSE) +
  autolayer(naive(beer2, h=11),
    series="Naïve", PI=FALSE) +
  autolayer(snaive(beer2, h=11),
    series="Seasonal naïve", PI=FALSE) +
  ggtitle("Forecasts for quarterly beer production") +
  xlab("Year") + ylab("Megalitres") +
  guides(colour=guide_legend(title="Forecast"))
```



In Figure, the non-seasonal methods are applied to a series of 200 days of the Google daily closing stock price.

```
autoplot(goog200) +
  autolayer(meanf(goog200, h=40),
    series="Mean", PI=FALSE) +
  autolayer(rwf(goog200, h=40),
    series="Naïve", PI=FALSE) +
  autolayer(rwf(goog200, drift=TRUE, h=40),
    series="Drift", PI=FALSE) +
  ggtitle("Google stock (daily ending 6 Dec 2013)") +
  xlab("Day") + ylab("Closing Price (US$)") +
  guides(colour=guide_legend(title="Forecast"))
```



Sometimes one of these simple methods will be the best forecasting method available; but in many cases, these methods will serve as benchmarks rather than the method of choice. That is, any forecasting methods we develop will be compared to these simple methods to ensure that the new method is better than these simple alternatives. If not, the new method is not worth considering.

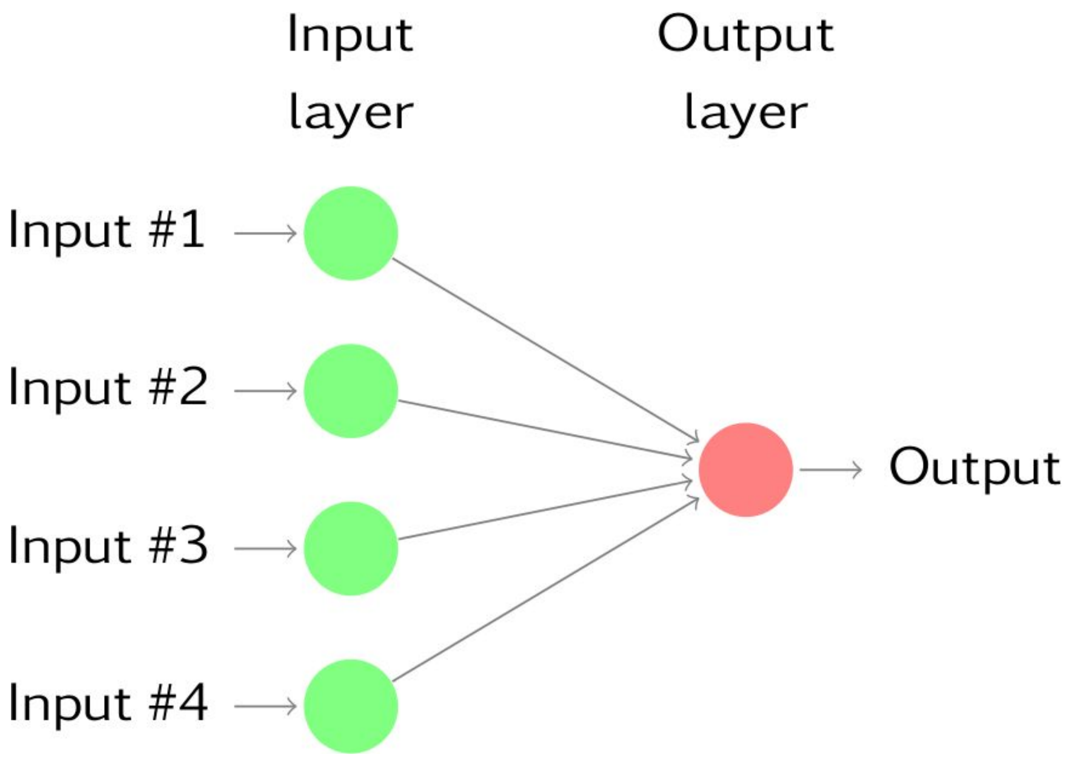
Neural network models in time forecasting

Artificial neural networks are forecasting methods that are based on simple mathematical models of the brain. They allow complex nonlinear relationships between the response variable and its predictors.

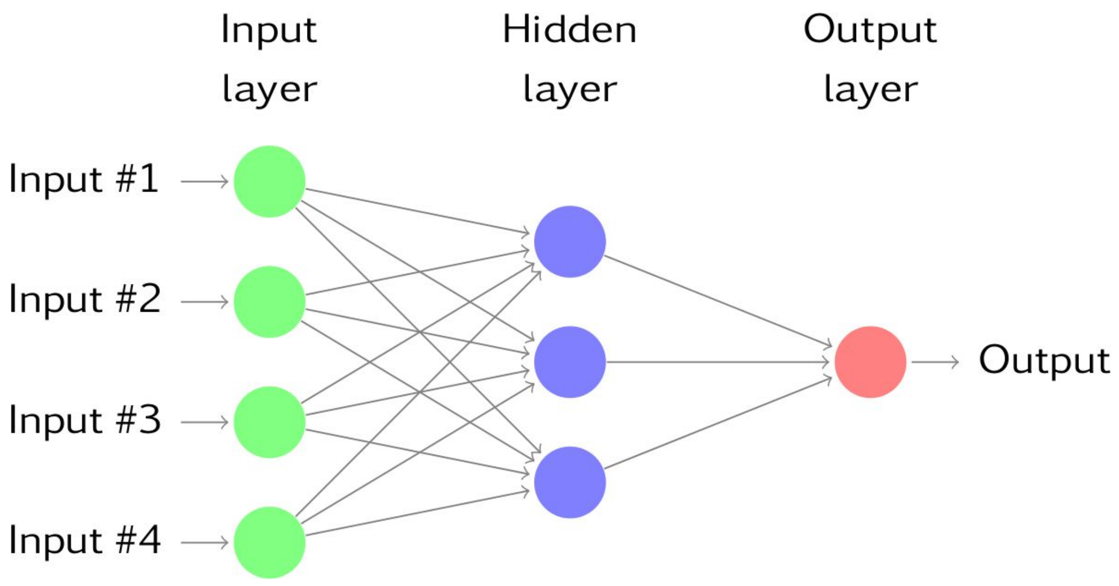
Neural network architecture

A neural network can be thought of as a network of “neurons” which are organised in layers. The predictors (or inputs) form the bottom layer, and the forecasts (or outputs) form the top layer. There may also be intermediate layers containing “hidden neurons”.

The simplest networks contain no hidden layers and are equivalent to linear regressions. Next Figure shows the neural network version of a linear regression with four predictors. The coefficients attached to these predictors are called “weights”. The forecasts are obtained by a linear combination of the inputs. The weights are selected in the neural network framework using a “learning algorithm” that minimises a “cost function” such as the MSE. Of course, in this simple example, we can use linear regression which is a much more efficient method of training the model.



Once we add an intermediate layer with hidden neurons, the neural network becomes non-linear. A simple example is shown in next Figure.



This is known as a multilayer feed-forward network, where each layer of nodes receives inputs from the previous layers. The outputs of the nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. The result is then modified by a nonlinear function before being output. For example, the inputs into hidden neuron j
in last Figure are combined linearly to give

$$z_j = b_j + \sum_{i=1}^{4} w_{i,j} x_i.$$

In the hidden layer, this is then modified using a nonlinear function such as a sigmoid,

$$s(z) = \frac{1}{1 + e^{-z}},$$

to give the input for the next layer. This tends to reduce the effect of extreme input values, thus making the network somewhat robust to outliers.

The parameters b1,b2,b3 and w1,1,…,w4,3 are “learned” from the data. The values of the weights are often restricted to prevent them from becoming too large. The parameter that restricts the weights is known as the “decay parameter”, and is often set to be equal to 0.1.

The weights take random values to begin with, and these are then updated using the observed data. Consequently, there is an element of randomness in the predictions produced by a neural network. Therefore, the network is usually trained several times using different random starting points, and the results are averaged.

The number of hidden layers, and the number of nodes in each hidden layer, must be specified in advance.

Neural network autoregression

With time series data, lagged values of the time series can be used as inputs to a neural network, just as we used lagged values in a linear autoregression model. We call this a neural network autoregression or NNAR model.

In this stuyd, we only consider feed-forward networks with one hidden layer, and we use the notation NNAR(p,k) to indicate there are p lagged inputs and k nodes in the hidden layer. For example, a NNAR(9,5) model is a neural network with the last nine observations (yt−1,yt−2,…,yt−9) used as inputs for forecasting the output yt, and with five neurons in the hidden layer. A NNAR(p,0) model is equivalent to an ARIMA(p,0,0) model, but without the restrictions on the parameters to ensure stationarity.

With seasonal data, it is useful to also add the last observed values from the same season as inputs. For example, an NNAR(3,1,2) 12 model has inputs yt−1, yt−2, yt−3 and yt−12, and two neurons in the hidden layer. More generally, an NNAR(p,P,k)m model has inputs

(yt−1,yt−2,…,yt−p,yt−m,yt−2m,yt−Pm) and k neurons in the hidden layer. A NNAR(p,P,0)m model is equivalent to an ARIMA(p,0,0)(P,0,0)m model but without the restrictions on the parameters that ensure stationarity.

The **nnetar()** function fits an NNAR(p,P,k)m model. If the values of p and P are not specified, they are selected automatically. For non-seasonal time series, the default is the optimal number of lags (according to the AIC) for a linear AR (p) model. For seasonal time series, the default values are P=1 and p is chosen from the optimal linear model fitted to the seasonally adjusted data. If k is not specified, it is set to k=(p+P+1)/2 (rounded to the nearest integer).

When it comes to forecasting, the network is applied iteratively. For forecasting one step ahead, we simply use the available historical inputs. For forecasting two steps ahead, we use the one-step forecast as an input, along with the historical data. This process proceeds until we have computed all the required forecasts.

Example of Time forecasting with Neural network: sunspots

The surface of the sun contains magnetic regions that appear as dark spots. These affect the propagation of radio waves, and so telecommunication companies like to predict sunspot activity in order to plan for any future difficulties. Sunspots follow a cycle of length between 9 and 14 years. In next Figure forecasts from an NNAR(10,6) are shown for the next 30 years. We have set a Box-Cox transformation with lambda=0 to ensure the forecasts stay positive.

```
fit <- nnetar(sunspotarea, lambda=0)
autoplot(forecast(fit,h=30))
```

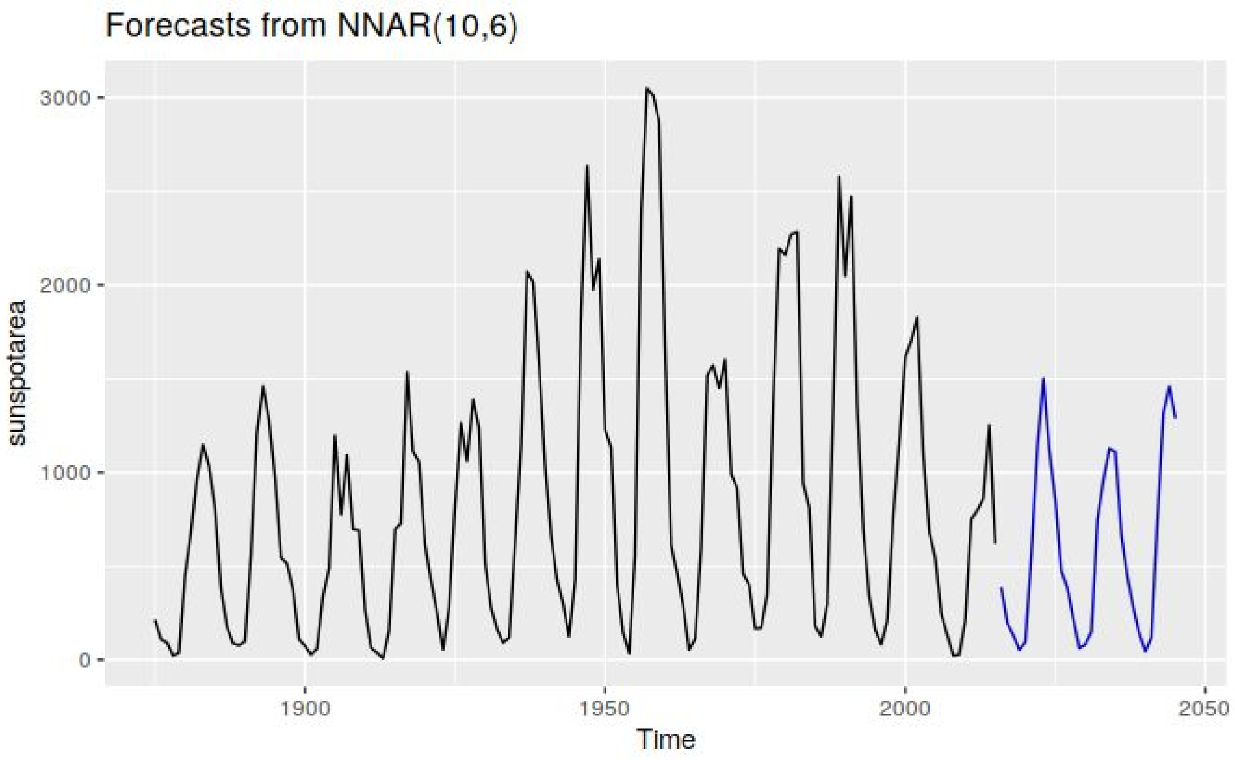


Figure: Forecasts from a neural network with ten lagged inputs and one hidden layer containing six neurons.

Here, the last 10 observations are used as predictors, and there are 6 neurons in the hidden layer. The cyclicity in the data has been modelled well. We can also see the asymmetry of the cycles has been captured by the model, where the increasing part of the cycle is steeper than the decreasing part of the cycle. This is one difference between a NNAR model and a linear AR model — while linear AR models can model cyclicity, the modelled cycles are always symmetric.

Prediction intervals

Unlike most of the methods considered in this study, neural networks are not based on a well-defined stochastic model, and so it is not straightforward to derive prediction intervals for the resultant forecasts. However, we can still compute prediction intervals using simulation where future sample paths are generated using bootstrapped residuals (as described in Section 3.5).

The neural network fitted to the sunspot data can be written as

$$y_t = f(\boldsymbol{y}_{t-1}) + \varepsilon_t$$

where yt−1=(yt−1,yt−2,…,yt−10)′ is a vector containing lagged values of the series, and f is a neural network with 6 hidden nodes in a single layer. The error series {εt} is assumed to be homoscedastic (and possibly also normally distributed).

We can simulate future sample paths of this model iteratively, by randomly generating a value for εt, either from a normal distribution, or by resampling from the historical values. So if εT+1∗is a random draw from the distribution of errors at time T+1, then

$$y^*_{T+1} = f(\boldsymbol{y}_T) + \varepsilon^*_{T+1}$$

is one possible draw from the forecast distribution for yT+1. Setting yT+1∗=(yT+1∗,yT,…,yT−6)′, we can then repeat the process to get

$$y^*_{T+2} = f(\boldsymbol{y}^*_{T+1}) + \varepsilon^*_{T+2}.$$

In this way, we can iteratively simulate a future sample path. By repeatedly simulating sample paths, we build up knowledge of the distribution for all future values based on the fitted neural network.

Here is a simulation of 9 possible future sample paths for the sunspot data. Each sample path covers the next 30 years after the observed data.

```r
sim <- ts(matrix(0, nrow=30L, ncol=9L),
  start=end(sunspotarea)[1L]+1L)
for(i in seq(9))
  sim[,i] <- simulate(fit, nsim=30L)
autoplot(sunspotarea) + autolayer(sim)
```

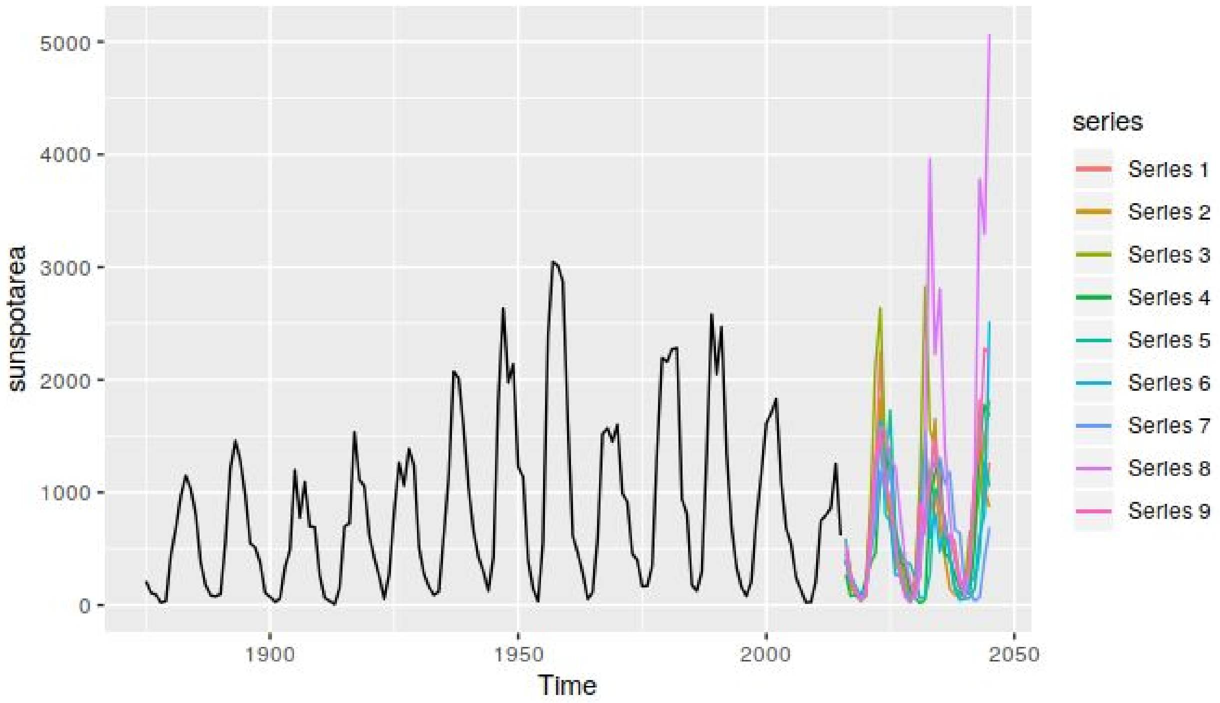


Figure: Future sample paths for the annual sunspot data.

If we do this a few hundred or thousand times, we can get a good picture of the forecast distributions. This is how the forecast() function produces prediction intervals for NNAR models:

```r
fcast <- forecast(fit, PI=TRUE, h=30)
autoplot(fcast)
```

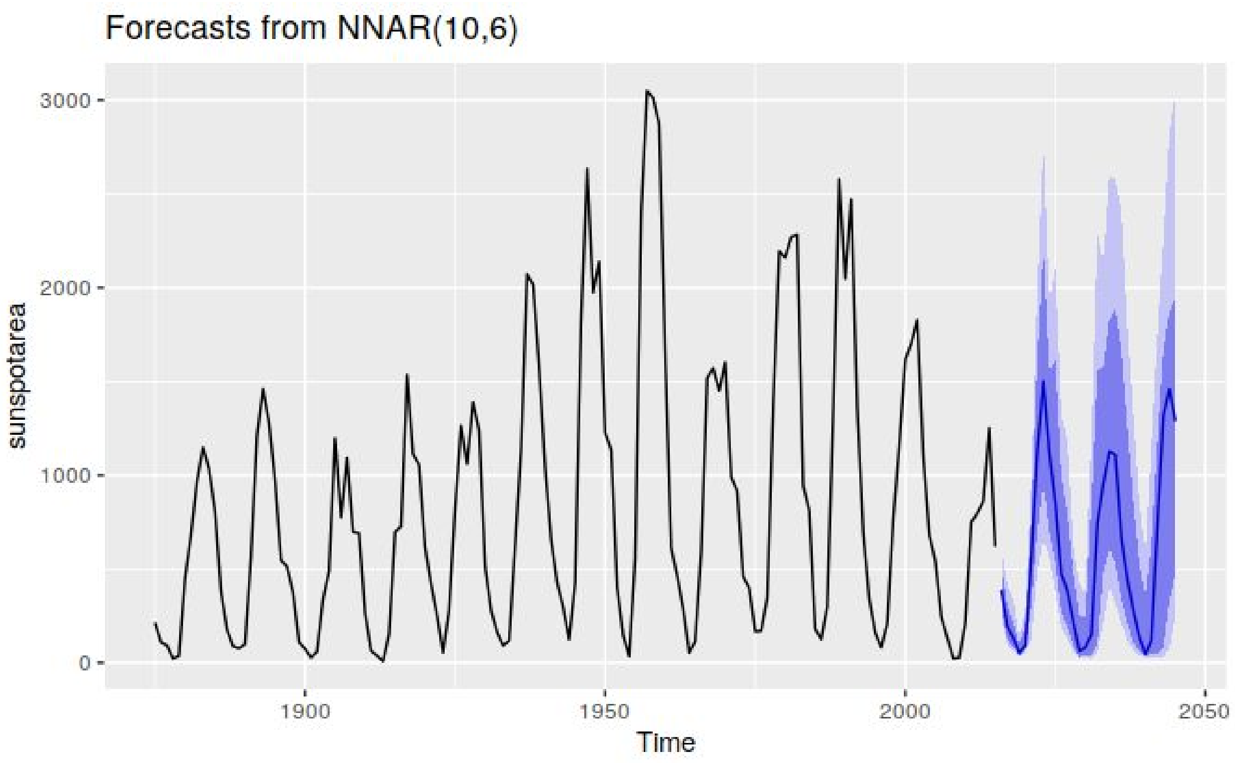


Figure: Forecasts with prediction intervals for the annual sunspot data. Prediction intervals are computed using simulated future sample paths.

Because it is a little slow, PI=FALSE is the default, so prediction intervals are not computed unless requested. The npaths argument in forecast() controls how many simulations are done (default 1000). By default, the errors are drawn from a normal distribution. The bootstrap argument allows the errors to be “bootstrapped” (i.e., randomly drawn from the historical errors).

Amazon Web Services (AWS)

Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis. In aggregate, these cloud computing web services provide a set of primitive abstract technical infrastructure and distributed computing building blocks and tools. One of these services is Amazon Elastic Compute Cloud, which allows users to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer, including hardware central processing units (CPUs) and graphics processing units (GPUs) for processing; local/RAM memory; hard-disk/SSD storage; a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, and customer relationship management (CRM).

The AWS technology is implemented at server farms throughout the world, and maintained by the Amazon subsidiary. Fees are based on a combination of usage (known as a "Pay-as-you-go" model), the hardware/OS/software/networking features chosen by the subscriber, required availability, redundancy, security, and service options. Subscribers can pay for a single virtual AWS computer, a dedicated physical computer, or clusters of either. As part of the subscription agreement, Amazon provides security for subscribers' system. AWS operates from many global geographical regions including 6 in North America.

In 2020, AWS comprised more than 212 services spanning a wide range including computing, storage, networking, database, analytics, application services, deployment, management, mobile, developer tools, and tools for the Internet of Things. The most popular include Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (Amazon S3). Most services are not exposed directly to end users, but instead offer functionality through APIs for developers to use in their applications. Amazon Web Services' offerings are accessed over HTTP, using the REST architectural style and SOAP protocol for older APIs and exclusively JSON for newer ones.

Amazon markets AWS to subscribers as a way of obtaining large scale computing capacity more quickly and cheaply than building an actual physical server farm. All services are billed based on usage, but each service measures usage in varying ways. As of 2017, AWS owns a dominant 34% of all cloud (IaaS, PaaS) while the next three competitors Microsoft, Google, and IBM have 11%, 8%, 6% respectively according to Synergy Group.

Amazon Web Services (AWS) Forecast

Amazon Forecast is a fully managed service for time-series forecasting. By providing Amazon Forecast with historical time-series data, you can predict future points in the series. Time-series forecasting is useful in multiple domains, including retail, financial planning, supply chain, and healthcare. You can also use Amazon Forecast to forecast operational metrics for inventory management, and workforce and resource planning and management.

For example, you can use Amazon Forecast to forecast the following:

- Retail product demand, such as the demand for products selling on a website or at a particular store or location
- Supply chain demand including the quantity of raw goods, services, or other inputs needed by manufacturing
- Resource requirements, such as the number of call center agents, contract workers, IT staff, and energy needed to meet demand
- Operational metrics, such as web traffic to servers, AWS usage, or IoT sensor usage
- Business metrics, such as cash flow, sales, profits, and expenses on a per-region or per-service basis

Amazon Forecast greatly simplifies building machine learning models. In addition to providing a set of predefined algorithms, Forecast provides an AutoML option for model training. AutoML automates complex machine learning tasks, such as algorithm selection, hyperparameter tuning, iterative modeling, and model assessment. Developers with no machine learning expertise can use the Amazon Forecast APIs, AWS Command Line Interface (AWS CLI), or Amazon Forecast console to import training data into one or more Amazon Forecast datasets, train predictors, and generate forecasts.

Amazon Forecast provides the following additional advantages:

- **Accuracy** – Amazon Forecast uses deep neural net and traditional statistical methods for forecasting. When you have many related time series, forecasts made using the Amazon Forecast deep learning algorithms, such as DeepAR+, tend to be more accurate than forecasts made with traditional methods, such as exponential smoothing.

- **Usability** – You can use theAmazon Forecast console to look up and visualize forecasts for any time series at different granularities. You can also see metrics for the accuracy of your forecasts.

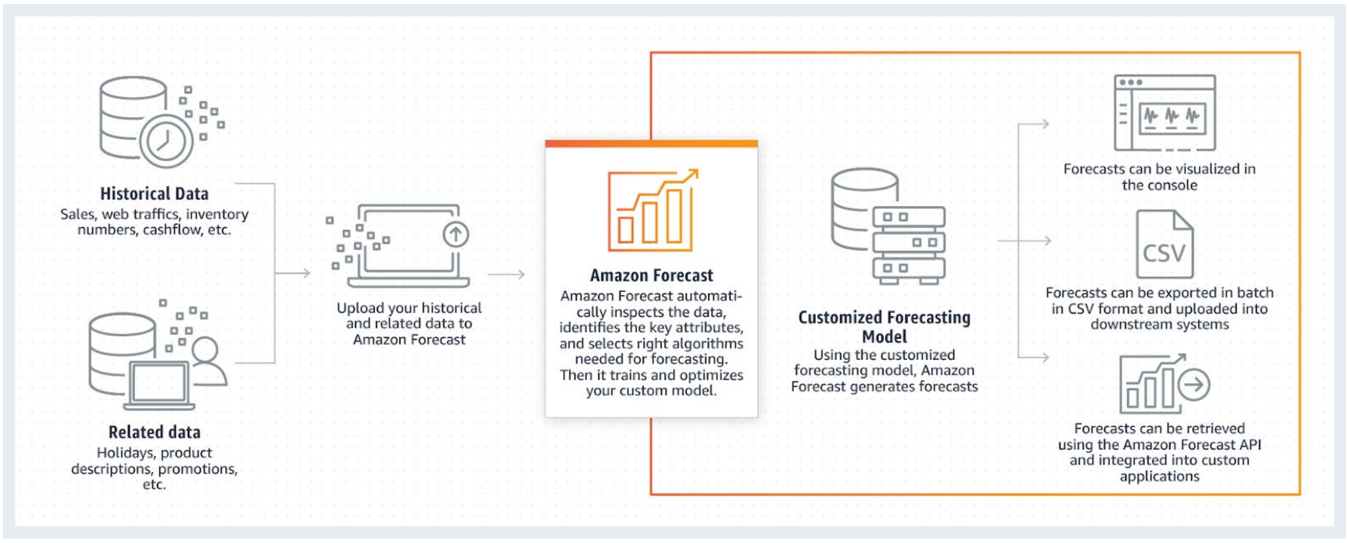
How Amazon Forecast Works

When creating forecasting projects in Amazon Forecast, you work with the following resources:

- **Datasets and Dataset Groups** – Datasets are collections of your input data. Dataset groups are collections of datasets that contain complimentary information. Forecast algorithms use your dataset groups to train custom forecasting models, called predictors.

- **Predictors** – Predictors are custom models trained on your data. You can train a predictor by choosing a prebuilt algorithm,or by choosing the AutoML option to have Amazon Forecast pick the best algorithm for you.

- **Forecasts** – You can generate forecasts for your time-series data, query them using the QueryForecast API, or visualize them in the console.


Historical Data
Sales, web traffics, inventory numbers, cashflow, etc.
Related data
Holidays, product descriptions, promotions, etc.
Upload your historical and related data to Amazon Forecast
Amazon Forecast
Amazon Forecast automatically inspects the data, identifies the key attributes, and selects right algorithms needed for forecasting. Then it trains and optimizes your custom model.
Customized Forecasting Model
Using the customized forecasting model, Amazon Forecast generates forecasts
Forecasts can be visualized in the console
CSV
Forecasts can be exported in batch in CSV format and uploaded into downstream systems
Forecasts can be retrieved using the Amazon Forecast API and integrated into custom applications

Datasets and Dataset Groups

Datasets contain the data used to train a predictor. You create one or more Amazon Forecast datasets and import your training data into them. A dataset group is a collection of complimentary datasets that detail a set of changing parameters over a series of time. After creating a dataset group, you use it to train a predictor.

Each dataset group can have up to three datasets, one of each dataset type: target time series, related time series, and item metadata.

To create and manage Forecast datasets and dataset groups, you can use the Forecast console, AWS Command Line Interface (AWS CLI), or AWS SDK.

Datasets

To create and manage Forecast datasets, you can use the Forecast APIs, including the CreateDataset and DescribeDataset operations.

When creating a dataset, you provide information, such as the following:

- The frequency/interval at which you recorded your data. For example, you might aggregate and record retail item sales every week. In the our test, you use the average electricity used per hour.

- The prediction format (the domain) and dataset type (within the domain). A dataset domain specifies which type of forecast you'd like to perform, while a dataset type helps you organize your training data into Forecast-friendly categories.

- The dataset schema. A schema maps the column headers of your dataset. For instance, when monitoring demand, you might have collected hourly data on the sales of an item at multiple stores. In this case, your schema would define the order, from left to right, in which timestamp, location, and hourly sales appear in your training data file. Schemas also define each column's data type, such as string or integer.

Each column in your Forecast dataset represents either a forecast dimension or feature. Forecast dimensions describe the aspects of your data that do not change over time, such a store or location. Forecast features include any parameters in your data that vary across time, such as price or promotion. Some dimensions, like timestamp or itemId, are required in target time series and related time series datasets.

Dataset Domains and Dataset Types

When you create a Forecast dataset, you choose a domain and a dataset type. Forecast provides domains for a number of use cases, such as forecasting retail demand or web traffic. You can also create a custom domain. For a complete list of Forecast domains, see Predefined Dataset Domains and Dataset Types.
Within each domain, Forecast users can specify the following types of datasets:

- Target time series dataset (required) – Use this dataset type when your training data is a time series and it includes the field that you want to generate a forecast for. This field is called the target field.

- Related time series dataset (optional) – Choose this dataset type when your training data is a time series, but it doesn't include the target field. For instance, if you're forecasting item demand, a related time series dataset might have price as a field, but not demand.

- Item metadata dataset (optional) – Choose this dataset type when your training data isn't time-series data, but includes metadata information about the items in the target time series or related time series datasets. For instance, if you're forecasting item demand, an item metadata dataset might color or brand as dimensions. Forecast only considers the data provided by an item metadata dataset type when you use the DeepAR+ algorithm.

Depending on the information in your training data and what you want to forecast, you might create more than one dataset.

For example, suppose that you want to generate a forecast for the demand of retail items, such as shoes and socks. You might create the following datasets in the RETAIL domain:

- Target time series dataset – Includes the historical time-series demand data for the retail items (item_id, timestamp, and the target field demand). Because it designates the target field that you want to forecast, you must have at least one target time series dataset in a dataset group.
  You can also add up to ten other dimensions to a target time series dataset. If you include only a target time series dataset in your dataset group, you can create forecasts at either the item level or the forecast dimension level of granularity only.
-
- Related time series dataset – Includes historical time-series data other than the target field, such as price or revenue. Because related time series data must be mappable to target time series data, each related time series dataset must contain the same identifying fields. In the RETAIL domain, these would be item_id and timestamp.
  A related time series dataset might contain data that refines the forecasts made off of your target time series dataset. For example, you might include price data in your related time series dataset on the future dates that you want to generate a forecast for. This way, Forecast can make predictions with an additional dimension of context.
- Item metadata dataset – Includes metadata for the retail items. Examples of metadata include brand, category,color, and genre.
-

Example Dataset with a Forecast Dimension
Continuing with the preceding example, imagine that you want to forecast the demand for shoes and socks based on a store's previous sales. In the following target time series dataset, store is a time-series forecast dimension, while demand is the target field. Socks are sold in two store locations (NYC and SFO), and shoes are sold only in ORD.

The first three rows of this table contain the first available sales data for the NYC, SFO, and ORD stores. The last three rows contain the last recorded sales data for each store. The ... row represents all of the item sales data recorded between the first and last entries.

timestamp	item_id	store	demand
2019-01-01	socks	NYC	25
2019-01-05	socks	SFO	45
2019-02-01	shoes	ORD	10
...			
2019-06-01	socks	NYC	100
2019-06-05	socks	SFO	5
2019-07-01	shoes	ORD	50

Dataset Schema

Each dataset requires a schema, a user-provided JSON mapping of the fields in your training data. This is where you list both the required and optional dimensions and features that you want to include in your dataset.

Some domains have optional dimensions that we recommend including. Optional dimensions are listed in the descriptions of each domain later in this guide. For an example, see RETAIL Domain. All optional dimensions take the data type string.

A schema is required for every dataset. The following is the accompanying schema for the example target time series dataset above.

```
{
     "attributes": [
        {
           "AttributeName": "timestamp",
           "AttributeType": "timestamp"
        },
        {
           "AttributeName": "item_id",
           "AttributeType": "string"
        },
        {
           "AttributeName": "store",
           "AttributeType": "string"
        },
        {
           "AttributeName": "demand",
           "AttributeType": "float"
        }
    ]
}
```

When you upload your training data to the dataset that uses this schema, Forecast assumes that the timestamp field is column 1, the item_id field is column 2, the store field is column 3, and the demand field, the target field, is column 4.

For the related time series dataset type, all related features must have a float or integer attribute type. For the item metadata dataset type, all features must have a string attribute type. For more information.

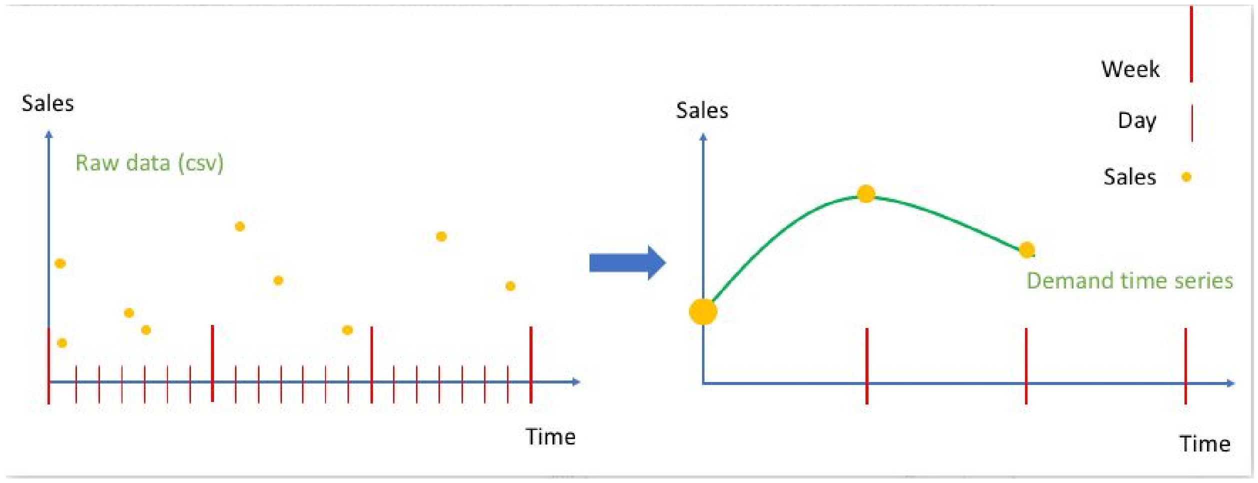
Note
An attributeName and attributeType pair is required for every column in the dataset. Forecast reserves a number of names that can't be used as the name of a schema attribute. For the list of reserved names,

Time Boundaries

The following table lists the time alignment boundaries Forecast uses when aggregating data.

Frequency	**Boundary**
Year	First day of the year (January 1)
Month	First day of the month
Week	Most recent Monday
Hour	Last top of the hour (09:00:00, 13:00:00)
Minute	Last top of the minute (45:00, 06:00)

The following figure shows how Forecast transforms data to fit the weekly boundary:



Choosing an Amazon Forecast Algorithm

Every Amazon Forecast predictor uses an algorithm to train a model, then uses the model to make a forecast using an input dataset group. To help you get started, Amazon Forecast provides the following predefined algorithms:

- Autoregressive Integrated Moving Average (ARIMA) Algorithm

  `arn:aws:forecast:::algorithm/ARIMA`

- DeepAR+ Algorithm

  `arn:aws:forecast:::algorithm/Deep_AR_Plus`

- Exponential Smoothing (ETS) Algorithm

  `arn:aws:forecast:::algorithm/ETS`

- Non-Parametric Time Series (NPTS) Algorithm

  `arn:aws:forecast:::algorithm/NPTS`

- Prophet Algorithm

  `arn:aws:forecast:::algorithm/Prophet`

Autoregressive Integrated Moving Average (ARIMA) Algorithm

Autoregressive Integrated Moving Average (ARIMA) is a commonly-used local statistical algorithm for time-series forecasting. ARIMA captures standard temporal structures (patterned organizations of time) in the input dataset. The Amazon Forecast ARIMA algorithm calls the Arima function in the Package 'forecast' of the Comprehensive R Archive Network (CRAN).

How ARIMA Works

The ARIMA algorithm is especially useful for datasets that can be mapped to stationary time series. The statistical properties of stationary time series, such as autocorrelations, are independent of time. Datasets with stationary time series usually contain a combination of signal and noise. The signal may exhibit a pattern of sinusoidal oscillation or have a seasonal component. ARIMA acts like a filter to separate the signal from the noise, and then extrapolates the signal in the future to make predictions.

ARIMA Hyperparameters and Tuning

Amazon Forecast converts the DataFrequency parameter specified in the CreateDataset operation to the frequency parameter of the R ts function using the following table:

DataFrequency (string)	**R ts frequency (integer)**
Y	1
M	12
W	52
D	7
H	24
30min	2
15min	4
10min	6
5min	12
1min	60

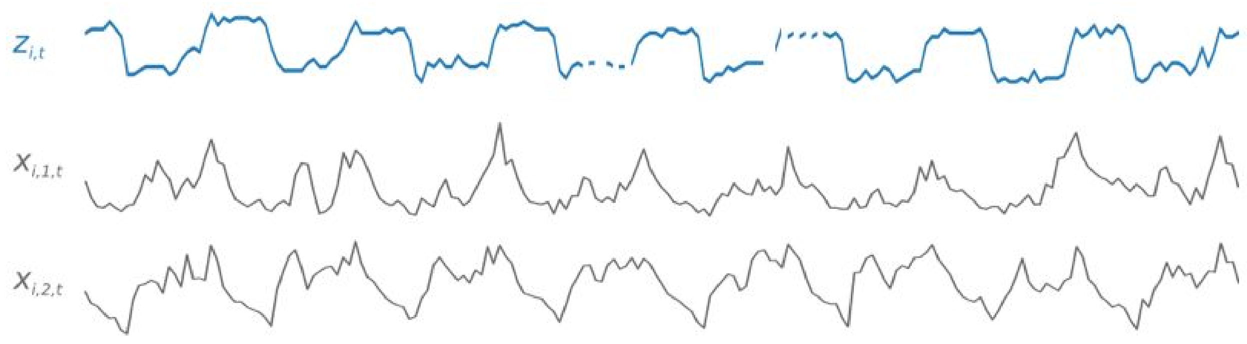
For frequencies less than 24 or short time series, the hyperparameters are set using the auto.arima function of the Package 'forecast' of CRAN. For frequencies greater than or equal to 24 and long time series, we use a Fourier series with K = 4,
Supported data frequencies that aren't in the table default to a ts frequency of 1.

DeepAR+ Algorithm

Amazon Forecast DeepAR+ is a supervised learning algorithm for forecasting scalar (one-dimensional) time series using recurrent neural networks (RNNs). Classical forecasting methods, such as autoregressive integrated moving average (ARIMA) or exponential smoothing (ETS), fit a single model to each individual time series, and then use that model to extrapolate the time series into the future. In many applications, however, you have many similar time series across a set of cross-sectional units. These time-series groupings demand different products, server loads, and requests for web pages. In this case, it can be beneficial to train a single model jointly over all of the time series. DeepAR+ takes this approach. When your dataset contains hundreds of feature time series, the DeepAR+ algorithm outperforms the standard ARIMA and ETS methods. You can also use the trained model for generating forecasts for new time series that are similar to the ones it has been trained on.

How DeepAR+ Works

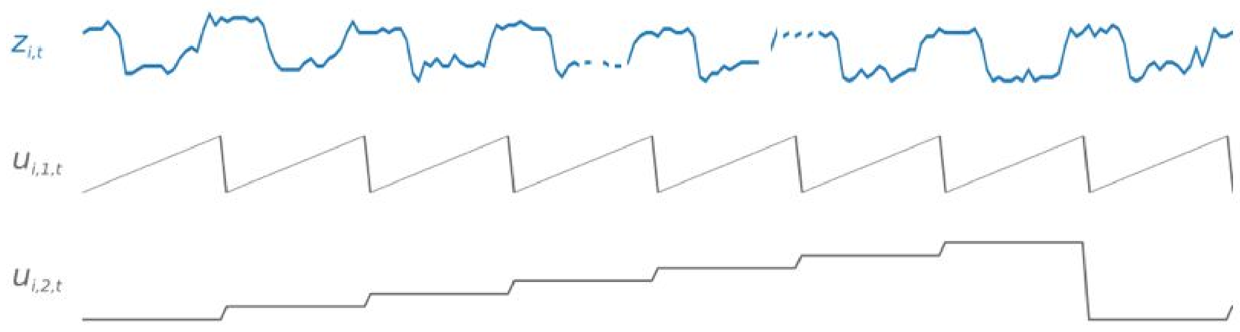
During training, DeepAR+ uses a training dataset and an optional testing dataset. It uses the testing dataset to evaluate the trained model. In general, the training and testing datasets don't have to contain the same set of time series. You can use a model trained on a given training set to generate forecasts for the future of the time series in the training set, and for other time series. Both the training and the testing datasets consist of (preferably more than one) target time series. Optionally, they can be associated with a vector of feature time series and a vector of categorical features (for details, see DeepAR Input/Output Interface in the Amazon SageMaker Developer Guide). The following example shows how this works for an element of a training dataset indexed by i. The training dataset consists of a target time series, zi,t, and two associated feature time series, xi,1,t and xi,2,t.



The target time series might contain missing values (denoted in the graphs by breaks in the time series). DeepAR+ supports only feature time series that are known in the future. This allows you to run counterfactual "what-if" scenarios. For example, "What happens if I change the price of a product in some way?"

Each target time series can also be associated with a number of categorical features. You can use these to encode that a time series belongs to certain groupings. Using categorical features allows the model to learn typical behavior for those groupings, which can increase accuracy. A model implements this by learning an embedding vector for each group that captures the common properties of all time series in the group.

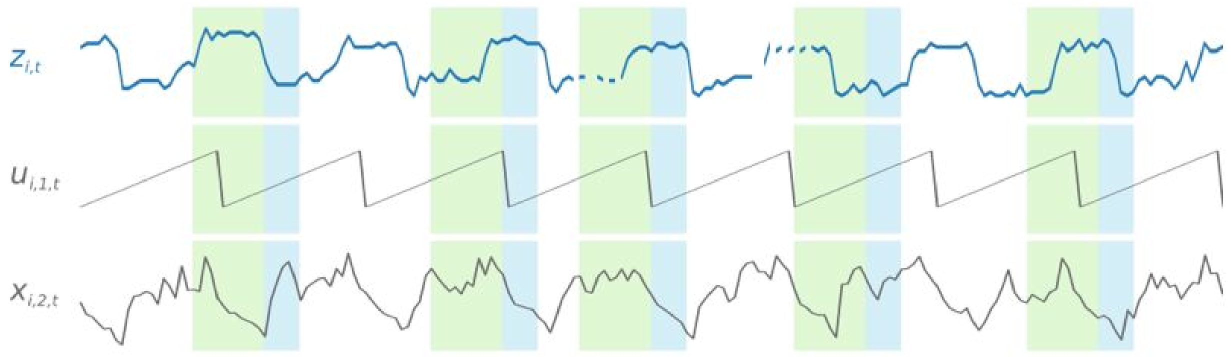
To facilitate learning time-dependent patterns, such as spikes during weekends, DeepAR+ automatically creates feature time series based on time-series granularity. For example, DeepAR+ creates two feature time series (day of the month and day of the year) at a weekly time-series frequency. It uses these derived feature time series along with the custom feature time series that you provide during training and inference. The following example shows two derived time-series features: ui,1,t represents the hour of the day, and ui,2,t the day of the week.



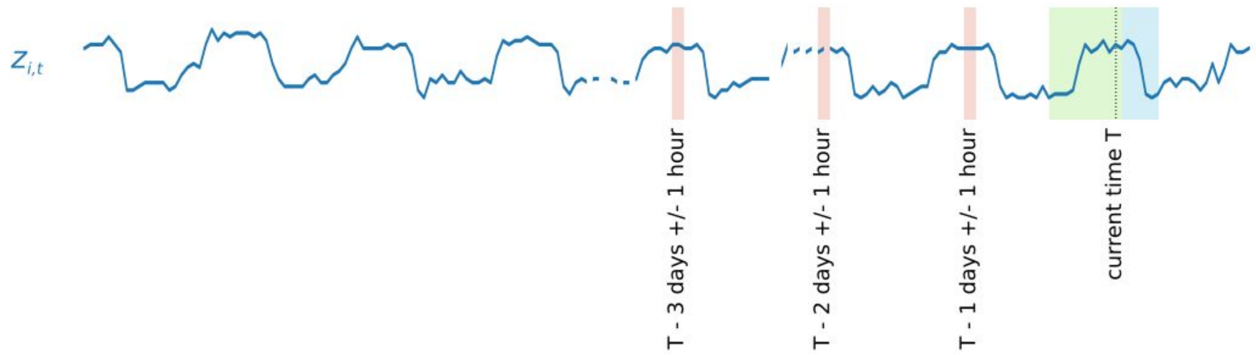
DeepAR+ automatically includes these feature time series based on the data frequency and the size of training data. The following table lists the features that can be derived for each supported basic time frequency.

Frequency of the Time Series	Derived Features
Minute	minute-of-hour, hour-of-day, day-of-week, day-of-month, day-of-year
Hour	hour-of-day, day-of-week, day-of-month, day-of-year
Day	day-of-week, day-of-month, day-of-year
Week	day-of-month, week-of-year
Month	month-of-year

A DeepAR+ model is trained by randomly sampling several training examples from each of the time series in the training dataset. Each training example consists of a pair of adjacent context and prediction windows with fixed predefined lengths. The context_length hyperparameter controls how far in the past the network can see, and the prediction_length parameter controls how far in the future predictions can be made. During training, Amazon Forecast ignores elements in the training dataset with time series shorter than the specified prediction length. The following example shows five samples, with a context length (highlighted in green) of 12 hours and a prediction length (highlighted in blue) of 6 hours, drawn from element i. For the sake of brevity, we've excluded the feature time series xi,1,t and ui,2,t.



To capture seasonality patterns, DeepAR+ also automatically feeds lagged (past period) values from the target time series. In our example with samples taken at an hourly frequency, for each time index t = T, the model exposes the zi,t values which occurred approximately one, two, and three days in the past (highlighted in pink).



For inference, the trained model takes as input the target time series, which might or might not have been used during training, and forecasts a probability distribution for the next prediction_length values. Because DeepAR+ is trained on the entire dataset, the forecast takes into account learned patterns from similar time series.

Exponential Smoothing (ETS) Algorithm

Exponential Smoothing (ETS) is a commonly-used local statistical algorithm for time-series forecasting. The Amazon Forecast ETS algorithm calls the ets function in the Package 'forecast' of the Comprehensive R Archive Network (CRAN).

How ETS Works

The ETS algorithm is especially useful for datasets with seasonality and other prior assumptions about the data. ETS computes a weighted average over all observations in the input time series dataset as its prediction. The weights are exponentially decreasing over time, rather than the constant weights in simple moving average methods. The weights are dependent on a constant parameter, which is known as the smoothing parameter.

Non-Parametric Time Series (NPTS) Algorithm

The Amazon Forecast Non-Parametric Time Series (NPTS) algorithm is a scalable, probabilistic baseline forecaster. It predicts the future value distribution of a given time series by sampling from past observations. The predictions are bounded by the observed values. NPTS is especially useful when the time series is intermittent (or sparse, containing many 0s) and bursty. For example, forecasting demand for individual items where the time series has many low counts. Amazon Forecast provides variants of NPTS that differ in which of the past observations are sampled and how they are sampled. To use an NPTS variant, you choose a hyperparameter setting.

How NPTS Works

Similar to classical forecasting methods, such as exponential smoothing (ETS) and autoregressive integrated moving average (ARIMA), NPTS generates predictions for each time series individually. The time series in the dataset can have different lengths. The time points where the observations are available are called the training range and the time points where the prediction is desired are called the prediction range.

Amazon Forecast NPTS forecasters have the following variants: NPTS, seasonal NPTS, climatological forecaster, and seasonal climatological forecaster.

Prophet Algorithm

Prophet is a popular local Bayesian structural time series model. The Amazon Forecast Prophet algorithm uses the Prophet class of the Python implementation of Prophet.

How Prophet Works

Prophet is especially useful for datasets that:

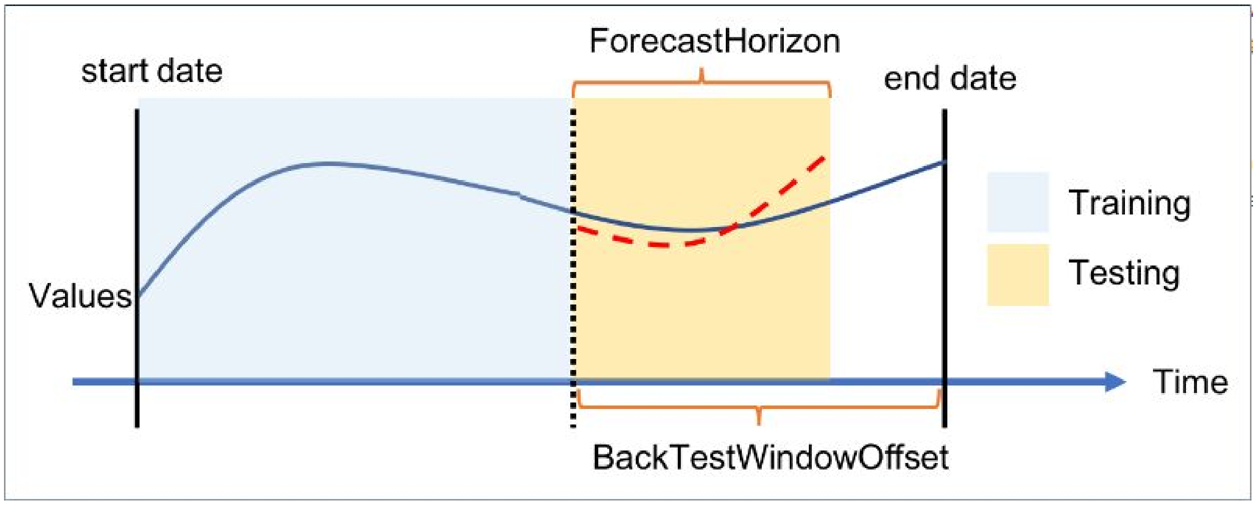
- Contain an extended time period (months or years) of detailed historical observations (hourly, daily, or weekly)
- Have multiple strong seasonalities
- Include previously known important, but irregular, events
- Have missing data points or large outliers
- Have non-linear growth trends that are approaching a limit

Prophet is an additive regression model with a piecewise linear or logistic growth curve trend. It includes a yearly seasonal component modeled using Fourier series and a weekly seasonal component modeled using dummy variables.

Evaluating Predictor Accuracy

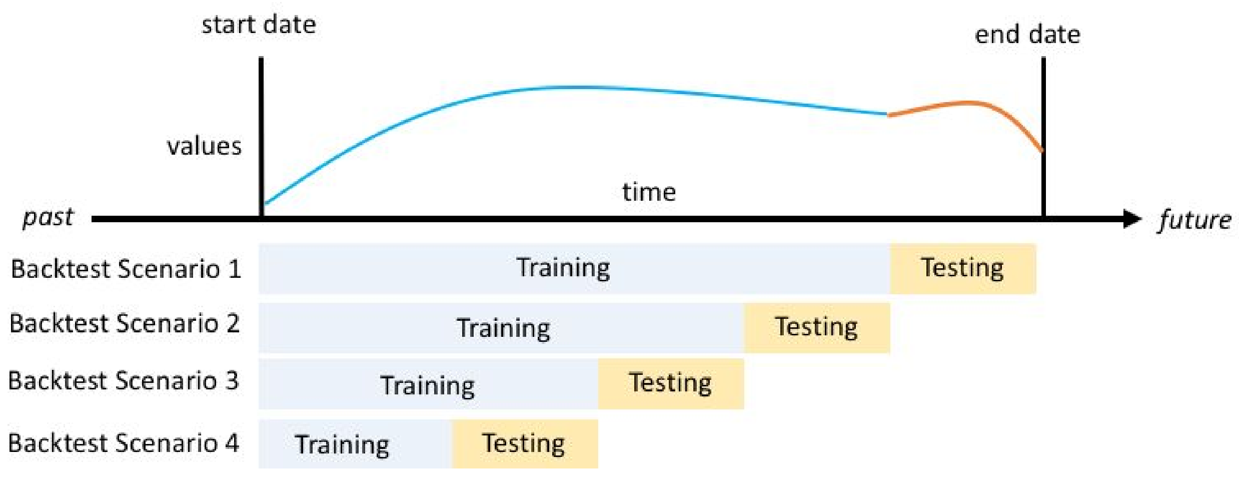
To evaluate the accuracy of an algorithm for various forecasting scenarios and to tune the predictor, use predictor metrics. Amazon Forecast uses backtesting to produce metrics.

Forecast automatically splits your input data into two datasets, training and test, as shown in the following figure. Forecast decides how to split the input data by using the BackTestWindowOffset parameter that you specify in the CreatePredictor operation, or if not specified, it uses the default value of the ForecastHorizon parameter. For more information, see EvaluationParameters.



To evaluate the metrics in multiple backtest scenarios with different virtual forecast start dates, as shown in the following figure, use the NumberOfBacktestWindows parameter in the CreatePredictor operation. The default for the NumberOfBacktestWindows parameter is

1. If you use the default, Forecast uses the simple splitting method shown in the preceding figure.



After training, Amazon Forecast calculates the root mean square error (RMSE) and weighted quantile losses to determine how well the model predicted the test data in each backtest window and the average value over all the backtest windows. These metrics measure the difference between the values predicted by the model and the actual values in the test dataset. To retrieve the metrics, you use the GetAccuracyMetrics operation.

Root Mean Square Error

RMSE is the square of the error term, which is the difference between the actual target value, yi,t, and the predicted (forecasted) value, ŷi,t, where i denotes the item index ranging from 1 to the total number of items, n, and t denotes the time index of the time series ranging from 1 to the final time in the evaluation period, T.

$$\text{RMSE} = \sqrt{\frac{1}{nT}\sum_{i,t}(\hat{y}_{i,t} - y_{i,t})^2},$$
$$i = 1, \ldots, n$$
$$t = 1, \ldots T$$

The RMSE metric favors a model whose individual errors are of consistent magnitude because large variations in error increase the RMSE. Because of the squared error, a few poorly predicted values in an otherwise good forecast can increase the RMSE.

Prediction Quantiles and MAPE

Prediction quantiles (intervals) express the uncertainty in the forecasts. By calculating prediction quantiles, the model shows how much uncertainty is associated with each forecast. Without accompanying prediction quantiles, point forecasts have limited value.

Predicting forecasts at different quantiles is particularly useful when the costs of under and over predicting differ. Amazon Forecast provides probabilistic predictions at three distinct quantiles—10%, 50%, and 90%—and calculates the associated loss (error) at each quantile. The weighted quantile loss (wQuantileLoss) calculates how far off the forecast is from actual demand in either direction. This is calculated as a percentage of demand on average in each quantile. This metric helps capture the bias inherent in each quantile, which can't be captured by a calculation like MAPE (Mean Absolute Percentage Error), where the weights are equal. As with MAPE and RMSE, lower wQuantileLoss errors indicate better overall forecast accuracy.

The weighted quantile loss is calculated as follows:

$$\mathrm{wQuantileLoss}[\tau] = 2\frac{\sum_{i,t}[\tau \max(y_{i,t} - q_{i,t}^{(\tau)}, 0) + (1-\tau)\max(q_{i,t}^{(\tau)} - y_{i,t}, 0)]}{\sum_{i,t}|y_{i,t}|}$$

q**i,t(τ) is the τ-quantile that the model predicts. τ is in the set {0.1, 0.2, ..., 0.9}.
Amazon Forecast calculates the weighted P10, P50, and P90 quantile losses, where τ is in the set {0.1, 0.5, 0.9}, respectively. This covers the standard 80% confidence interval. For RMSE, Amazon Forecast uses the P50 forecast to represent the predicted value, for example, ŷi,t = qi,t(0.5).

When the sum of the exact target over all items and all time is approximately zero in a given backtest window, the weighted quantile loss expression is undefined. In this case, Amazon Forecast outputs the unweighted quantile loss, which is the numerator in the above wQuantileLoss expression.

wQuantileLoss[0.1]: For the P10 prediction, the true value is expected to be lower than the predicted value 10% of the time.

For example, suppose that you're a retailer and you want to forecast product demand for winter gloves that sell well only during the fall and winter. If you don't have a lot of storage space and the cost of invested capital is high, or if the price of being overstocked on winter gloves concerns you, you might use the P10 quantile to order a relatively low number of winter gloves.

You know that the P10 forecast overestimates the demand for your winter gloves only 10% of the time, so 90% of the time you'll be sold out of your winter gloves.

wQuantileLoss[0.5]: For the P50 prediction, the true value is expected to be lower than the predicted value 50% of the time. In most cases, the point forecasts that you generate internally or with other forecasting tools should match the P50 forecasts. If τ = 0.5, both weights are equal and the wQuantileLoss[0.5] reduces to the commonly used Mean Absolute Percentage Error (MAPE):

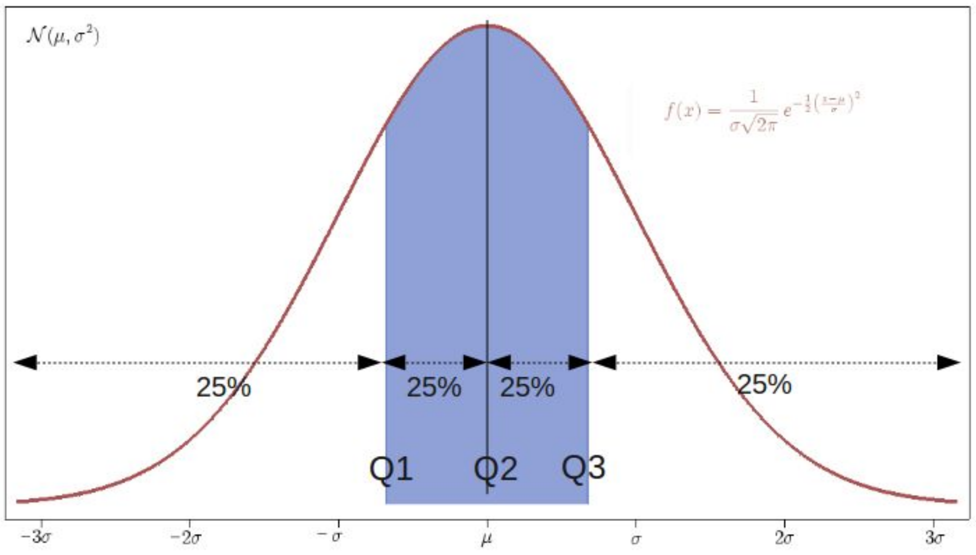
$$\text{MAPE} = \frac{\sum_{i,t} |\hat{y}_{i,t} - y_{i,t}|}{\sum_{i,t} |y_{i,t}|}$$

where `ŷi,t = qi,t(0.5)`.Forecast uses the scaling factor of 2 in the wQuantileLoss formula to cancel the 0.5 factor to obtain the exact MAPE expression.

Continuing the winter gloves example, if you know that there'll be a moderate amount of demand for the gloves and aren't concerned about being overstocked, you might choose to use the P50 quantile to order gloves.
wQuantileLoss[0.9]: For the P90 prediction, the true value is expected to be lower than the predicted value 90% of the time.

If you determine that being understocked on gloves will result in huge amounts of lost revenue—for example, the cost of not selling gloves is extremely high or the cost of invested capital is low—you might choose to use the P90 quantile to order gloves.

The following figure of a forecast that has a Gaussian distribution, shows the quantiles that divide the forecast into four regions of equal probability. For information about the quantiles of a distribution.



Steps to use Amazon Forcast

In this exercise, you use the Amazon Forecast console to import time-series data of electricity usage, create an Amazon Forecast predictor based on the input dataset, and make predictions of future electricity usage based on the input time interval.

For this exercise, we use the individual household electric power consumption dataset. (Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.) We aggregate the usage data hourly.

Prerequisites

- An AWS account. If you don't already have an AWS account
- Training data in your Amazon Simple Storage Service (Amazon S3) bucket.
- An AWS Identity and Access Management (IAM) role that allows Amazon Forecast to read and write to your S3 buckets.

Step 1: Import Training Data

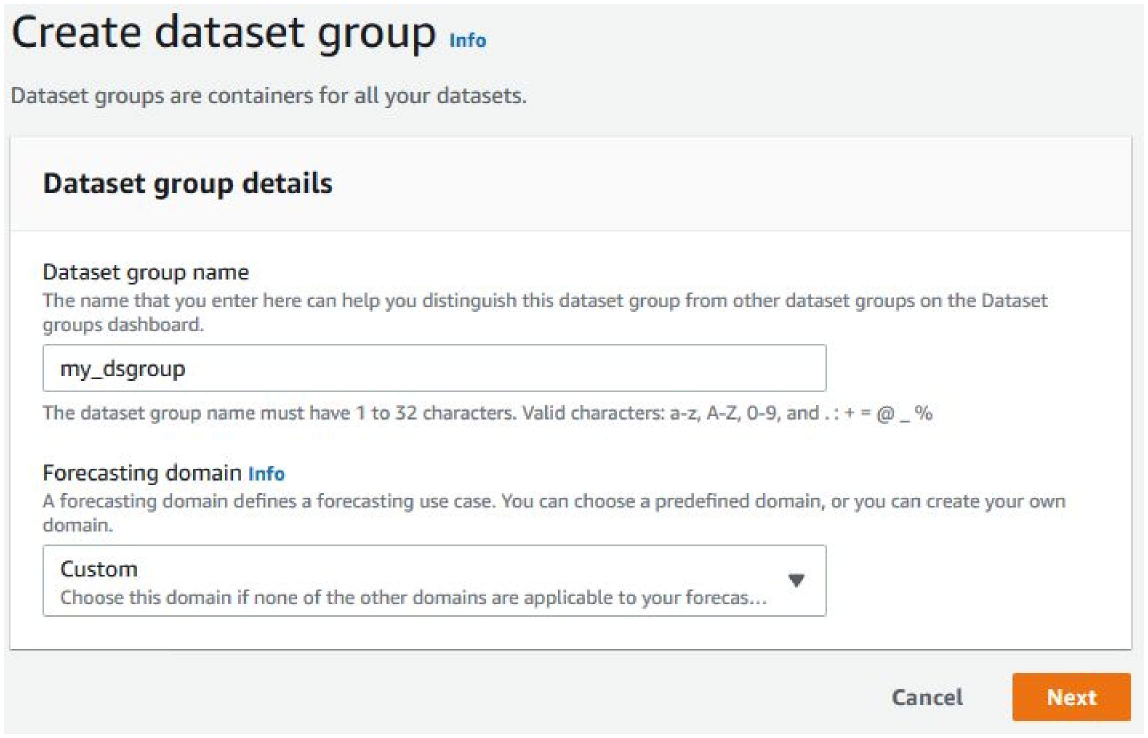
To import time-series data into Amazon Forecast, create a dataset group, choose a domain for your dataset group, specify the details of your data, and point Amazon Forecast to the S3 location of your data. You use a time series of historical electricity usage as an example for the target time series data.

Note
This exercise assumes that you haven't created any dataset groups. If you previously created a dataset group, what you see will vary slightly from the following screenshots and instructions.

To import time-series data for forecasting

1. Sign in to the AWS Management Console and open the Amazon Forecast console at https://console.aws.amazon.com/forecast/.
2. On the Amazon Forecast home page, choose Create dataset group.
3. On the Create dataset group page, for Dataset group details, provide the following information:
    - Dataset group name – Enter a name for your dataset group.
    - Forecasting domain – From the drop-down menu, choose Custom.

- similar to the following:



Create dataset group

Dataset groups are containers for all your datasets.

Dataset group details

Dataset group name

The name that you enter here can help you distinguish this dataset group from other dataset groups on the Dataset groups dashboard.

my_dsgroup

The dataset group name must have 1 to 32 characters. Valid characters: a-z, A-Z, 0-9, and . : + = @ _ %

Forecasting domain

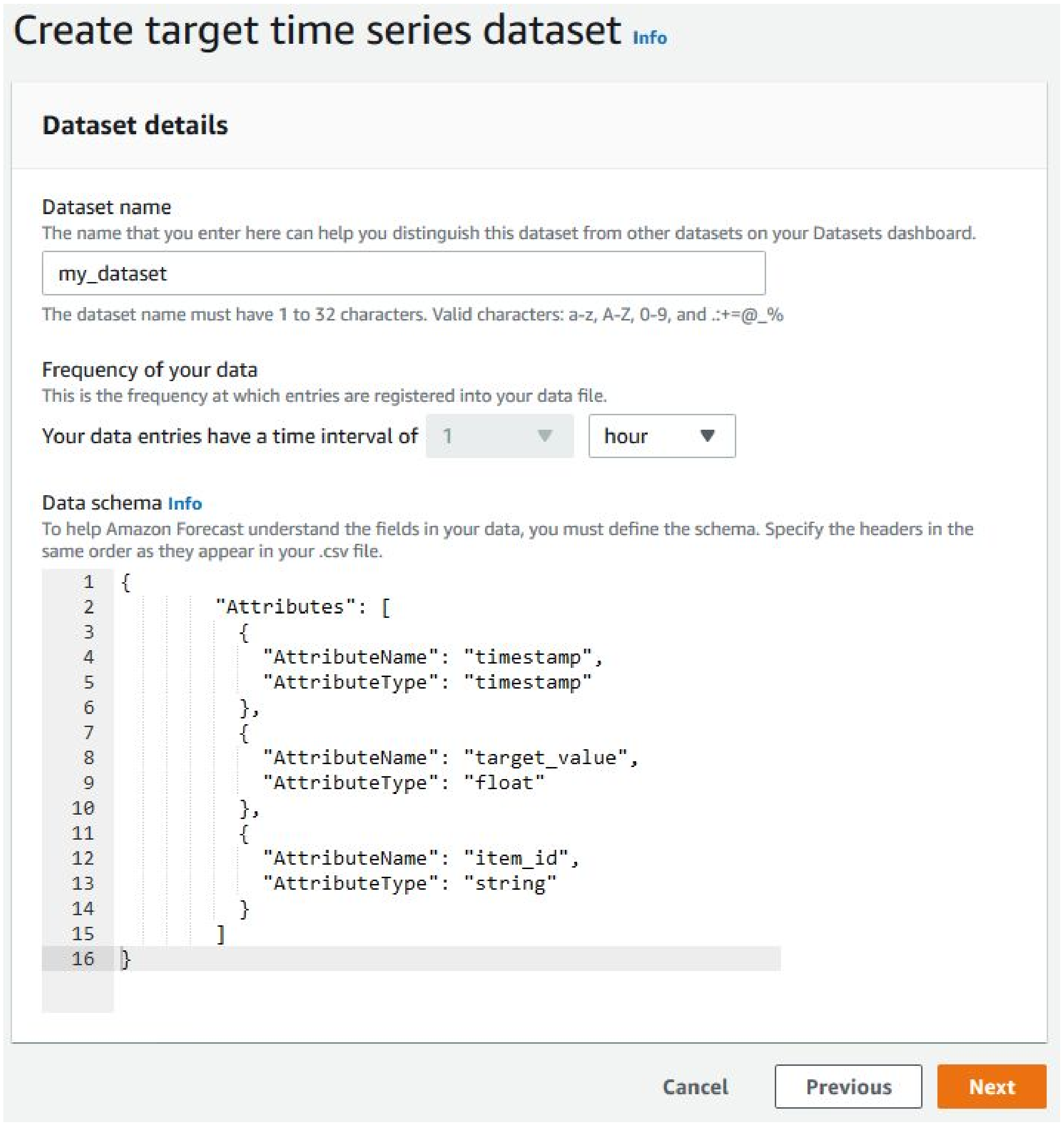
A forecasting domain defines a forecasting use case. You can choose a predefined domain, or you can create your own domain.

Custom

Choose this domain if none of the other domains are applicable to your forecas...

4. Choose Next.
5. On the Create target time series dataset page, for Dataset details, provide the following information:
    - Dataset name – Enter a name for your dataset.
    - Frequency of your data – Keep the default value of 1, and choose hour from the drop-down menu. This setting must be consistent with the input time series data. The time interval in the sample electricity-usage data is an hour.
    - Data schema – Update the schema to match the columns of the time-series data in data types and order. For the electricity usage input data, the columns correspond to: a timestamp, the electricity usage at the specified time (target_value), and the ID of the customer charged for the electricity usage (string), in that order.

6. Your screen should look similar to the following:



Create target time series dataset Info

Dataset details

Dataset name

The name that you enter here can help you distinguish this dataset from other datasets on your Datasets dashboard.

my_dataset

The dataset name must have 1 to 32 characters. Valid characters: a-z, A-Z, 0-9, and .:+=@_%

Frequency of your data

This is the frequency at which entries are registered into your data file.

Your data entries have a time interval of 1 hour

Data schema Info

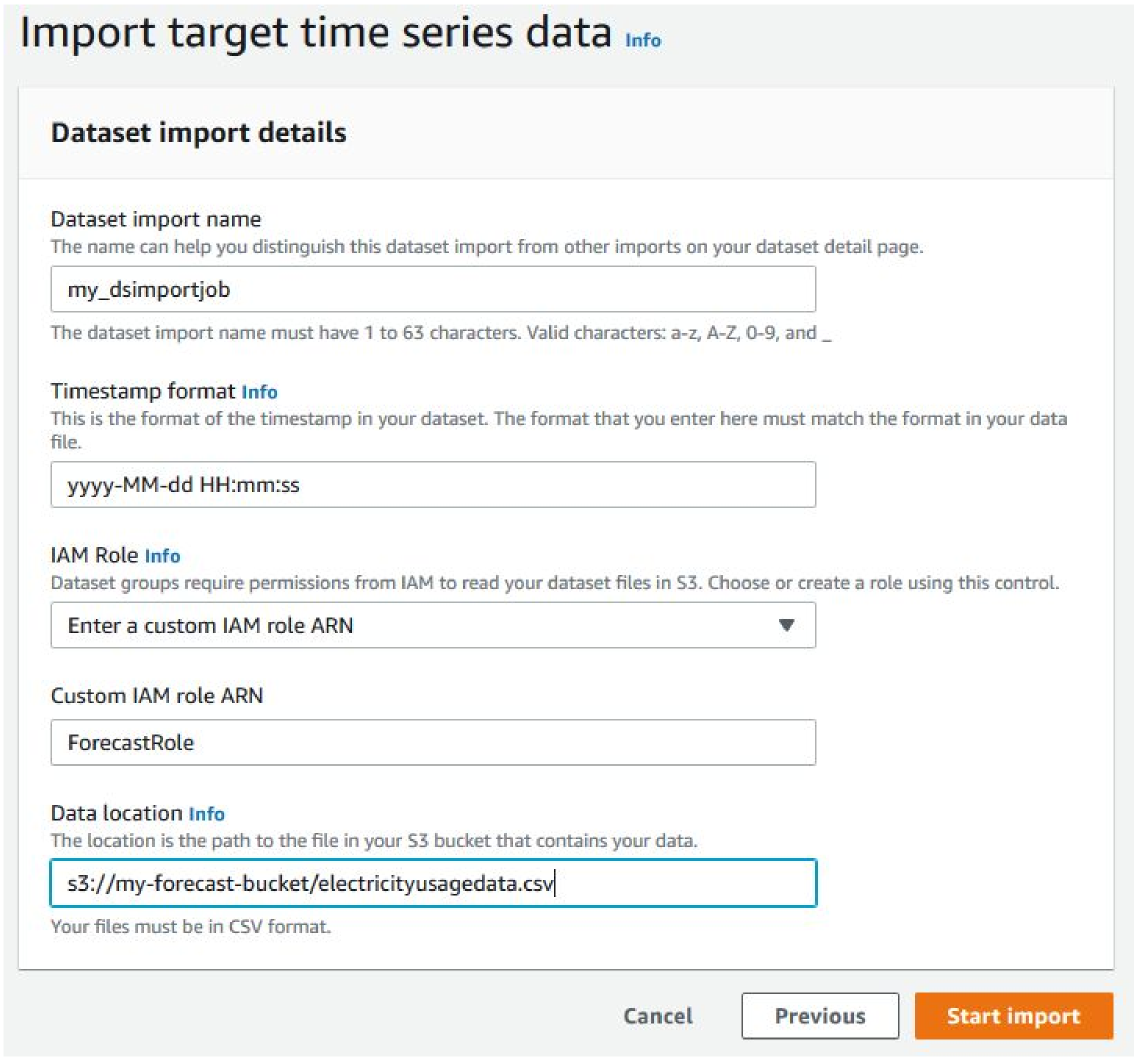
To help Amazon Forecast understand the fields in your data, you must define the schema. Specify the headers in the same order as they appear in your .csv file.

```
{
        "Attributes": [
          {
            "AttributeName": "timestamp",
            "AttributeType": "timestamp"
          },
          {
            "AttributeName": "target_value",
            "AttributeType": "float"
          },
          {
            "AttributeName": "item_id",
            "AttributeType": "string"
          }
        ]
}
```

7. Choose Next.
8. On the Import target time series data page, for Dataset import job details, provide the following information:
    - Dataset import job name – Enter a name for your dataset.
    - Timestamp format – Leave the default (yyyy-MM-dd HH:mm:ss). The format must be consistent with the input time series data.
    - IAM role – Keep the default Enter a custom IAM role ARN.
      Alternatively, you can have Amazon Forecast create the required IAM role for you by choosing Create a new role from the drop-down menu and following the on-screen instructions.
    - Custom IAM role ARN – Enter the Amazon Resource Name (ARN) of the IAM role that you created in Create an IAM Role for Amazon Forecast (IAM Console).

    ○ Data location – Use the following format to enter the location of your .csv file on Amazon S3:
      s3:////<filename.csv>

9. Your screen should look similar to the following:



Import target time series data Info

Dataset import details

Dataset import name
The name can help you distinguish this dataset import from other imports on your dataset detail page.
my_dsimportjob
The dataset import name must have 1 to 63 characters. Valid characters: a-z, A-Z, 0-9, and _

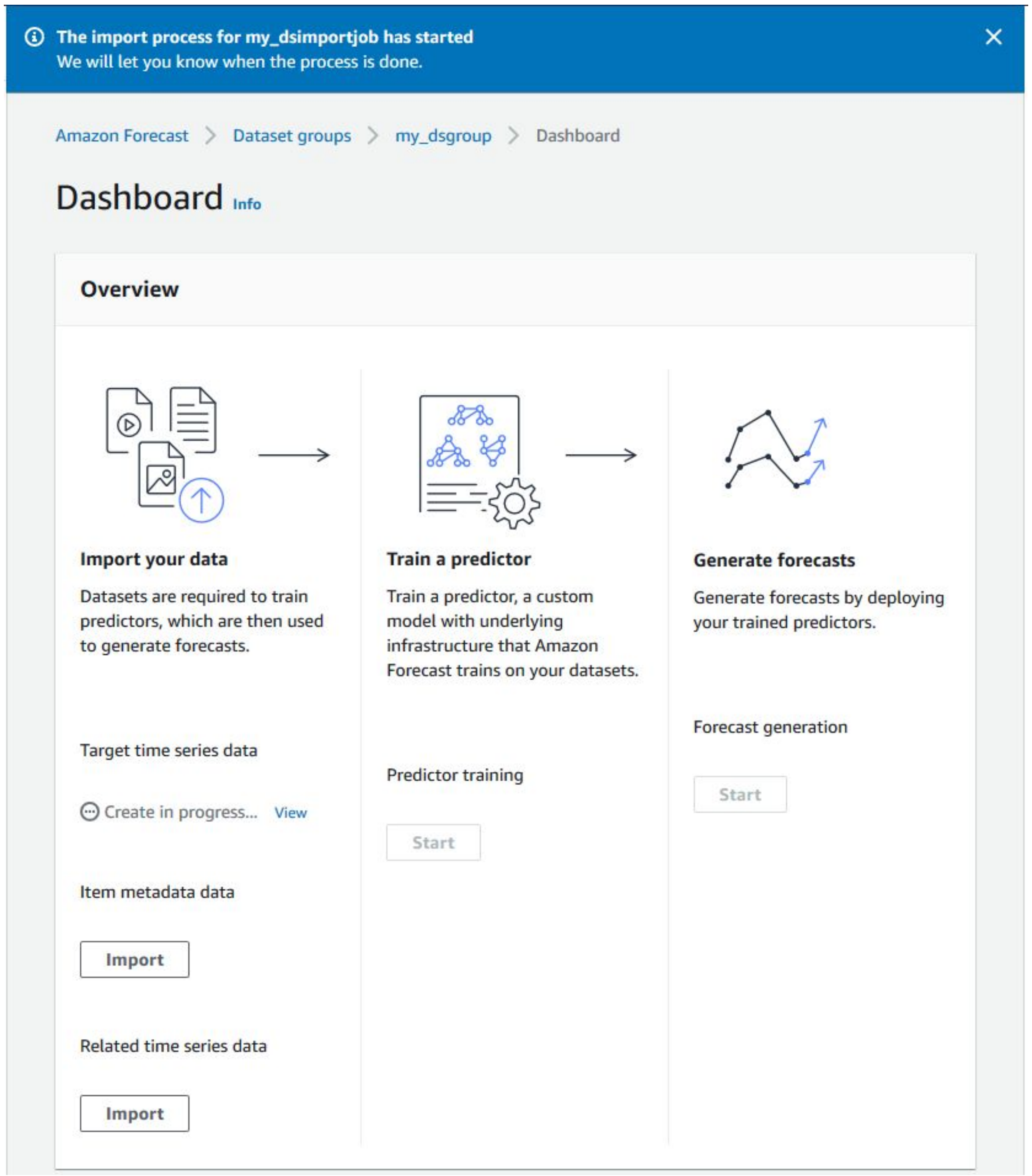
Timestamp format Info
This is the format of the timestamp in your dataset. The format that you enter here must match the format in your data file.
yyyy-MM-dd HH:mm:ss

IAM Role Info
Dataset groups require permissions from IAM to read your dataset files in S3. Choose or create a role using this control.
Enter a custom IAM role ARN

Custom IAM role ARN
ForecastRole

Data location Info
The location is the path to the file in your S3 bucket that contains your data.
s3://my-forecast-bucket/electricityusagedata.csv
Your files must be in CSV format.

10. Choose Start import.
11. The dataset group's Dashboard page is displayed. Your screen should look similar to the following:



The import process for my_dsimportjob has started
We will let you know when the process is done.

Amazon Forecast > Dataset groups > my_dsgroup > Dashboard

Dashboard Info

Overview

Import your data
Datasets are required to train predictors, which are then used to generate forecasts.

Target time series data
Create in progress... View

Item metadata data
Import

Related time series data
Import

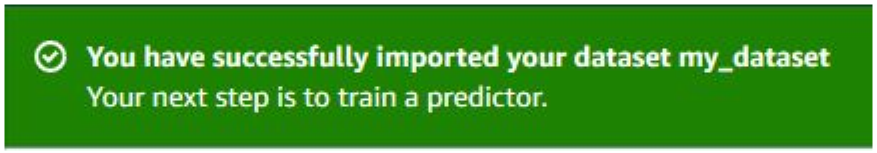
Train a predictor
Train a predictor, a custom model with underlying infrastructure that Amazon Forecast trains on your datasets.

Predictor training
Start

Generate forecasts
Generate forecasts by deploying your trained predictors.

Forecast generation
Start

Under Target time series data, you will see the status of the import job. Wait for Amazon Forecast to finish importing your time-series data. The process can take several minutes or longer. When your dataset has been imported, the status transitions to Active. Additionally, the banner at the top of the dashboard, changes to display the following message:



You have successfully imported your dataset my_dataset
Your next step is to train a predictor.

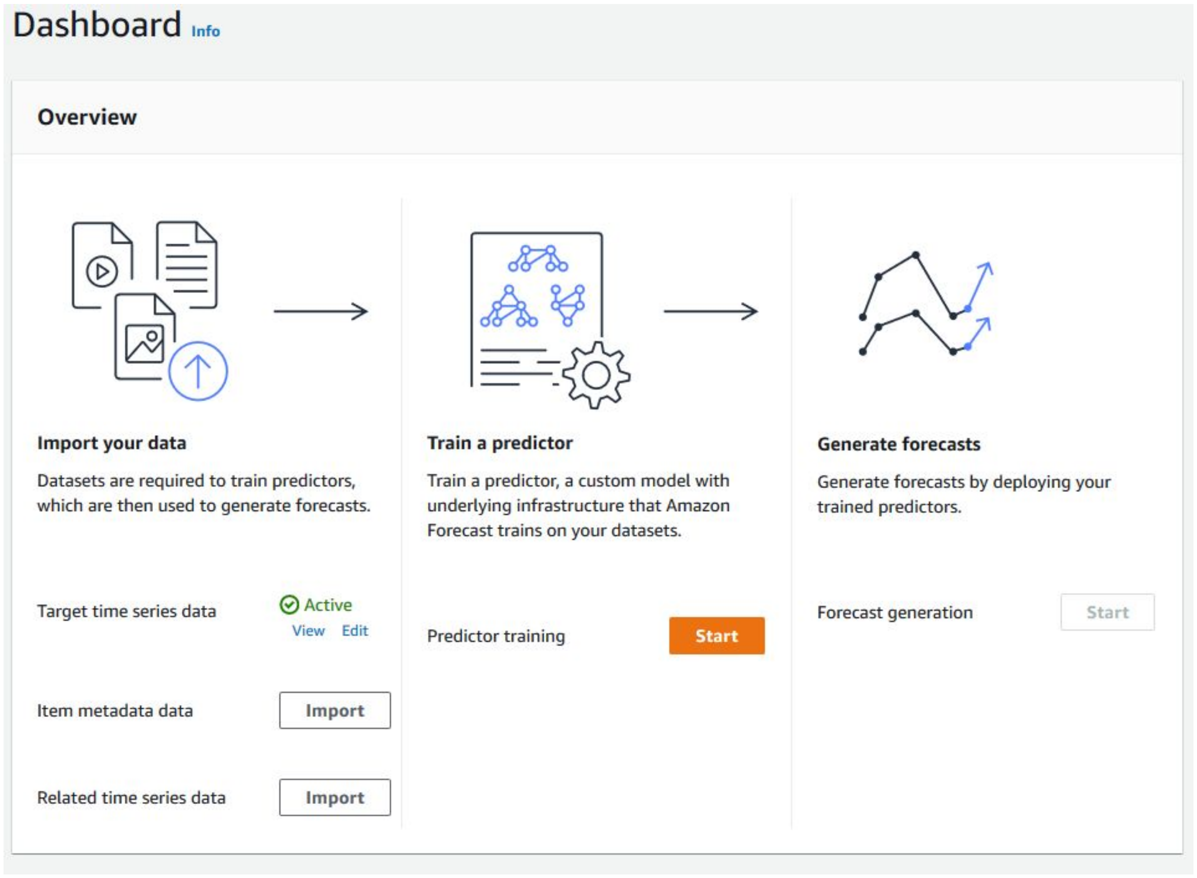
Now that your target time series dataset has been imported, you can train a predictor.

Step 2: Train a Predictor

To create a predictor, which is a trained model, choose an algorithm and the number (length times frequency) of predictions to make. You can choose a particular algorithm, or you can choose AutoML to have Amazon Forecast process your data and choose an algorithm to best suit your dataset group. For information about algorithms, see Choosing an Amazon Forecast Algorithm.

To train a predictor

1. After your target time series dataset has finished importing, your dataset group's Dashboard should look similar to the following:



Dashboard Info

Overview

Import your data

Datasets are required to train predictors, which are then used to generate forecasts.

Target time series data — Active — View Edit

Item metadata data — Import

Related time series data — Import

Train a predictor

Train a predictor, a custom model with underlying infrastructure that Amazon Forecast trains on your datasets.

Predictor training — Start

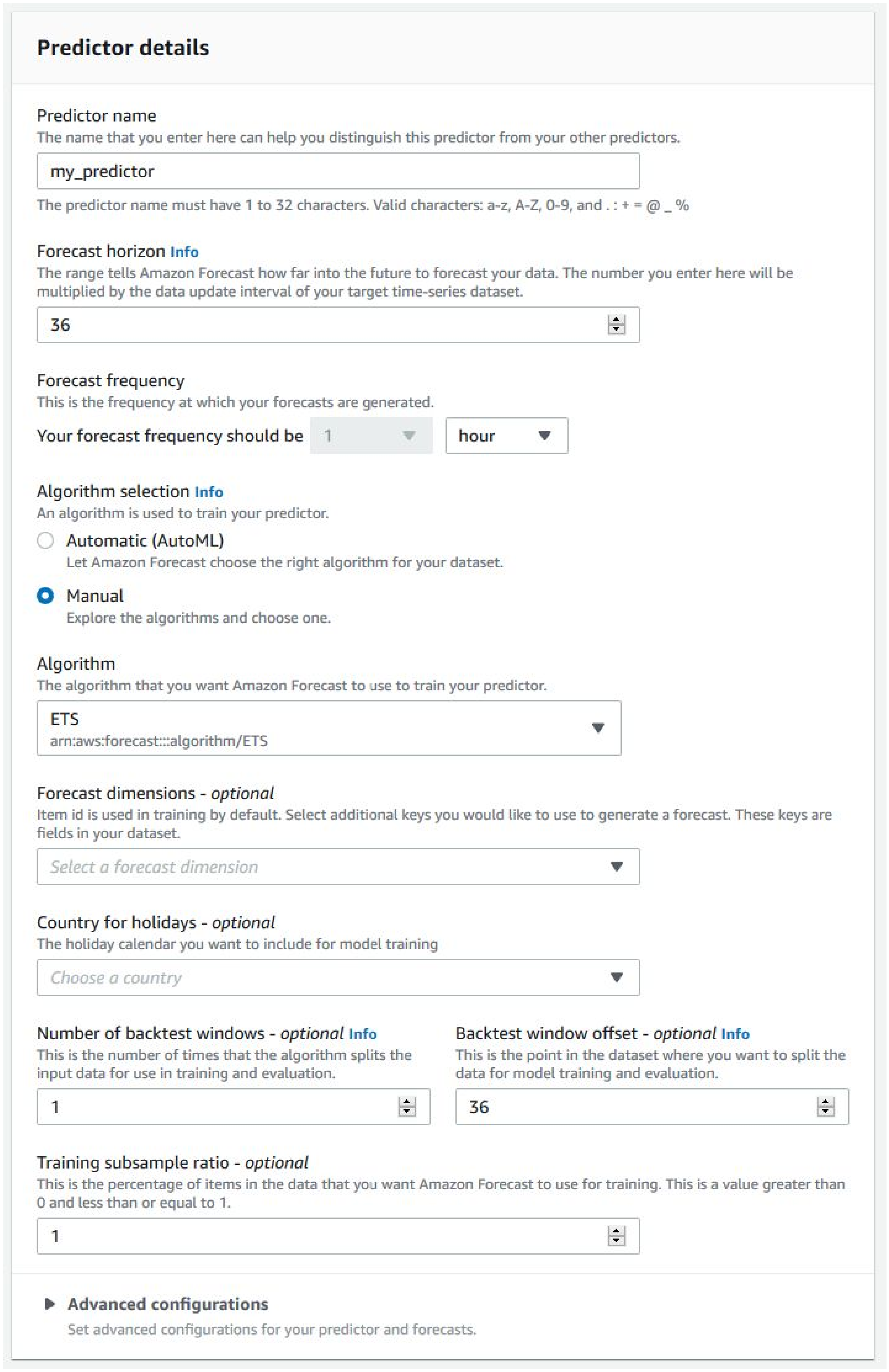
Generate forecasts

Generate forecasts by deploying your trained predictors.

Forecast generation — Start

Under Train a predictor, choose Start. The Train predictor page is displayed. Note
The Status of the Target time series data must be Active, which signifies that the import successfully finished, before you can train the predictor.

2. On the Train predictor page, for Predictor details, provide the following information:
    - Predictor name – Enter a name for your predictor.
    - Forecast horizon – Choose how far into the future to make predictions. This number multiplied by the data entry frequency (hourly) that you specified in Step 1: Import the Training Data determines how far into the future to make predictions. For this exercise, set the number to 36, to provide predictions for 36 hours.
    - Forecast frequency – Keep the default value of 1. From the drop-down menu, choose hour. This setting must be consistent with the input time series data. The time interval in the sample electricity-usage data is an hour.
    - Algorithm selection – Keep the default value Manual. From the drop-down menu, choose the ETS algorithm The remaining settings are optional, so leave the default values. Your screen should look similar to the following:



Predictor details

Predictor name
The name that you enter here can help you distinguish this predictor from your other predictors.
my_predictor
The predictor name must have 1 to 32 characters. Valid characters: a-z, A-Z, 0-9, and . : + = @ _ %

Forecast horizon Info
The range tells Amazon Forecast how far into the future to forecast your data. The number you enter here will be multiplied by the data update interval of your target time-series dataset.
36

Forecast frequency
This is the frequency at which your forecasts are generated.
Your forecast frequency should be 1 hour

Algorithm selection Info
An algorithm is used to train your predictor.
○ Automatic (AutoML)
Let Amazon Forecast choose the right algorithm for your dataset.
● Manual
Explore the algorithms and choose one.

Algorithm
The algorithm that you want Amazon Forecast to use to train your predictor.
ETS
arn:aws:forecast:::algorithm/ETS

Forecast dimensions - *optional*
Item id is used in training by default. Select additional keys you would like to use to generate a forecast. These keys are fields in your dataset.
Select a forecast dimension

Country for holidays - *optional*
The holiday calendar you want to include for model training
Choose a country

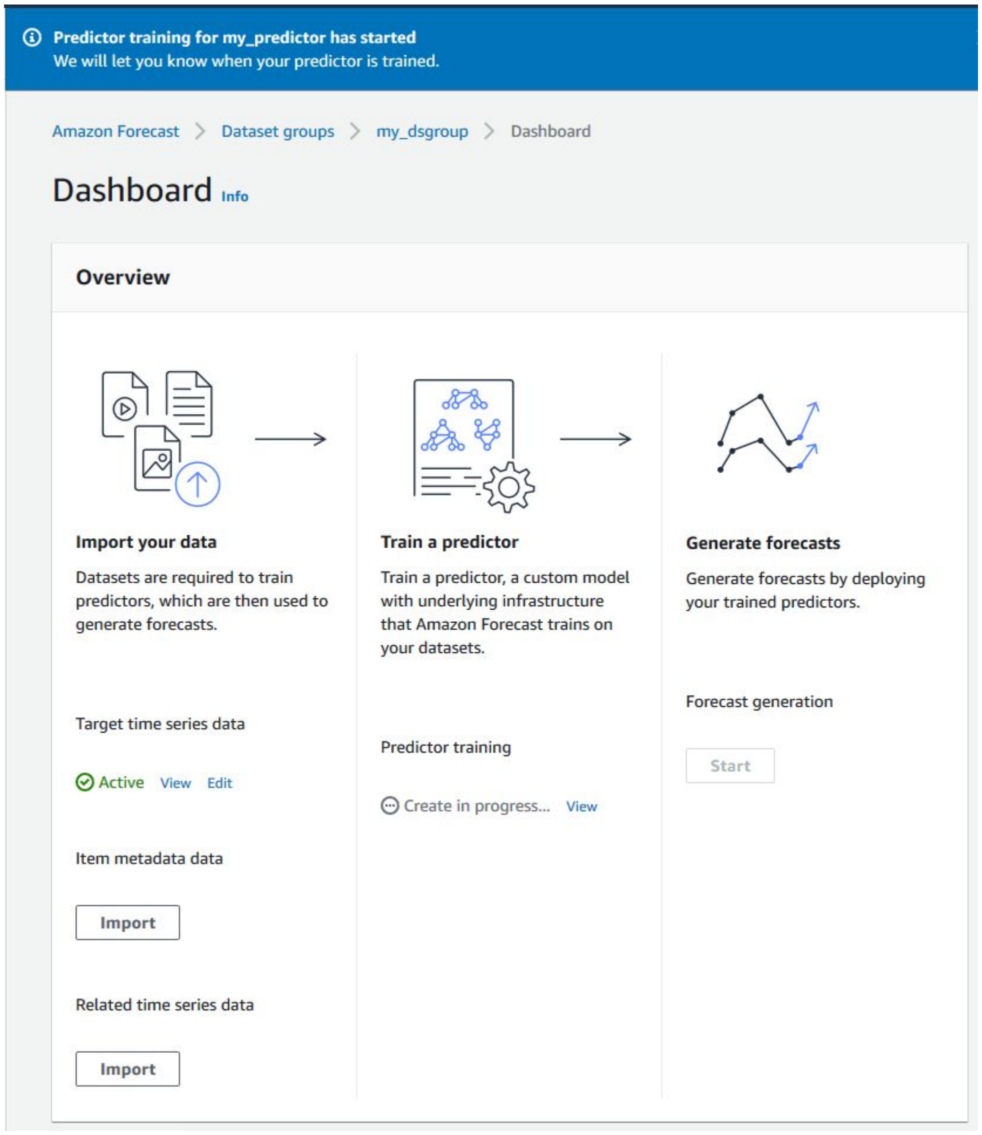
Number of backtest windows - *optional* Info
This is the number of times that the algorithm splits the input data for use in training and evaluation.
1

Backtest window offset - *optional* Info
This is the point in the dataset where you want to split the data for model training and evaluation.
36

Training subsample ratio - *optional*
This is the percentage of items in the data that you want Amazon Forecast to use for training. This is a value greater than 0 and less than or equal to 1.
1

▶ Advanced configurations
Set advanced configurations for your predictor and forecasts.

3. Choose Train predictor. Your dataset group's Dashboard page is displayed. Your screen should look similar to the following:



Predictor training for my_predictor has started
We will let you know when your predictor is trained.

Amazon Forecast > Dataset groups > my_dsgroup > Dashboard

Dashboard Info

Overview

Import your data
Datasets are required to train predictors, which are then used to generate forecasts.

Target time series data
Active View Edit

Item metadata data
Import

Related time series data
Import

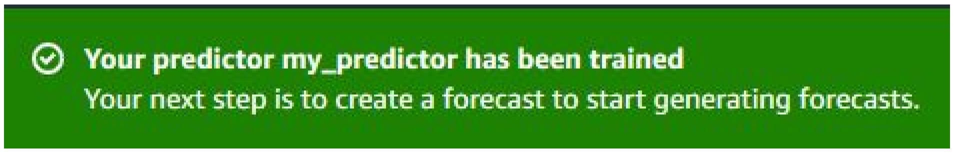
Train a predictor
Train a predictor, a custom model with underlying infrastructure that Amazon Forecast trains on your datasets.

Predictor training
Create in progress... View

Generate forecasts
Generate forecasts by deploying your trained predictors.

Forecast generation
Start

Under Predictor training, you will see the training status. Wait for Amazon Forecast to finish training the predictor. The process can take several minutes or longer. When your predictor has been trained, the status transitions to Active. Additionally, the banner at the top of the dashboard changes to display the following message:



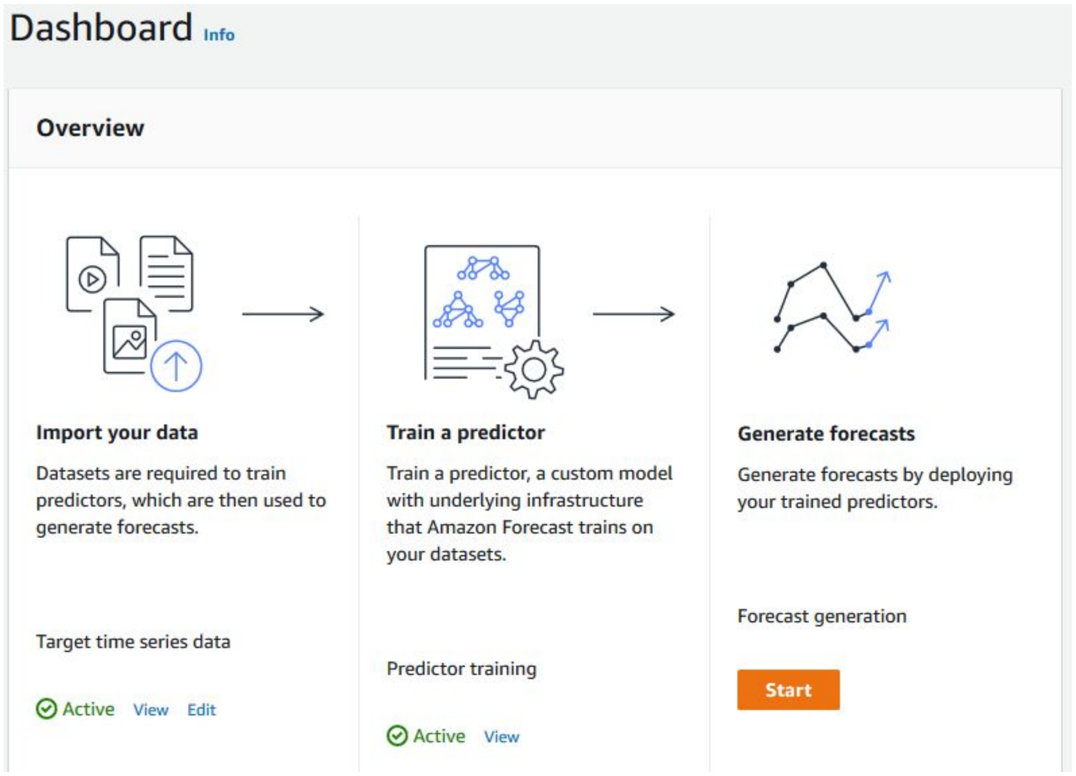
Your predictor my_predictor has been trained
Your next step is to create a forecast to start generating forecasts.

Now that your predictor has been trained, you can create a forecast.

Step 3: Create a Forecast

To make predictions (inferences), you use a predictor to create a forecast. A forecast is a group of predictions, one for every item in the target dataset. To retrieve the prediction for a single item, you query the forecast. To retrieve the complete forecast, you create an export job.
To get and view your forecast

1. After your predictor has finished training, your dataset group's Dashboard should look similar to the following:



Dashboard Info

Overview

Import your data

Datasets are required to train predictors, which are then used to generate forecasts.

Target time series data

✓ Active View Edit

Train a predictor

Train a predictor, a custom model with underlying infrastructure that Amazon Forecast trains on your datasets.

Predictor training

✓ Active View

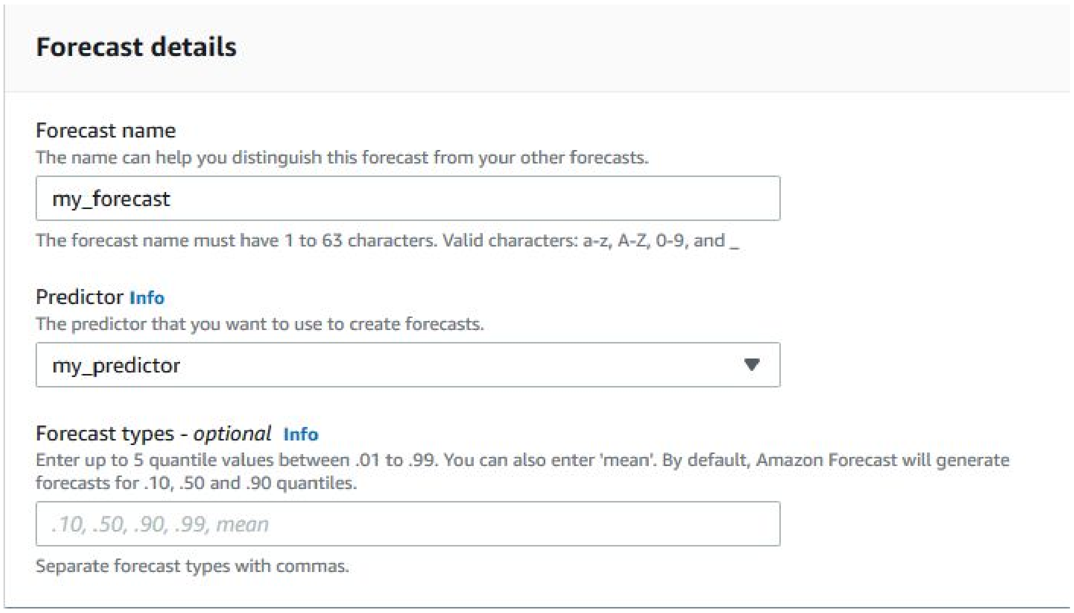
Generate forecasts

Generate forecasts by deploying your trained predictors.

Forecast generation

   Under Forecast generation, choose Start. The Create a forecast page is displayed. Note The Status of Predictor training must be Active before you can generate a forecast.
2. On the Create a forecast page, for Forecast details, provide the following information:
   - Forecast name – Enter a name for your forecast.
   - Predictor – From the drop-down menu, choose the predictor that you created in Step 2: Train a Predictor.

3. The remaining setting is optional, so leave the default value. Your screen should look similar to the following:



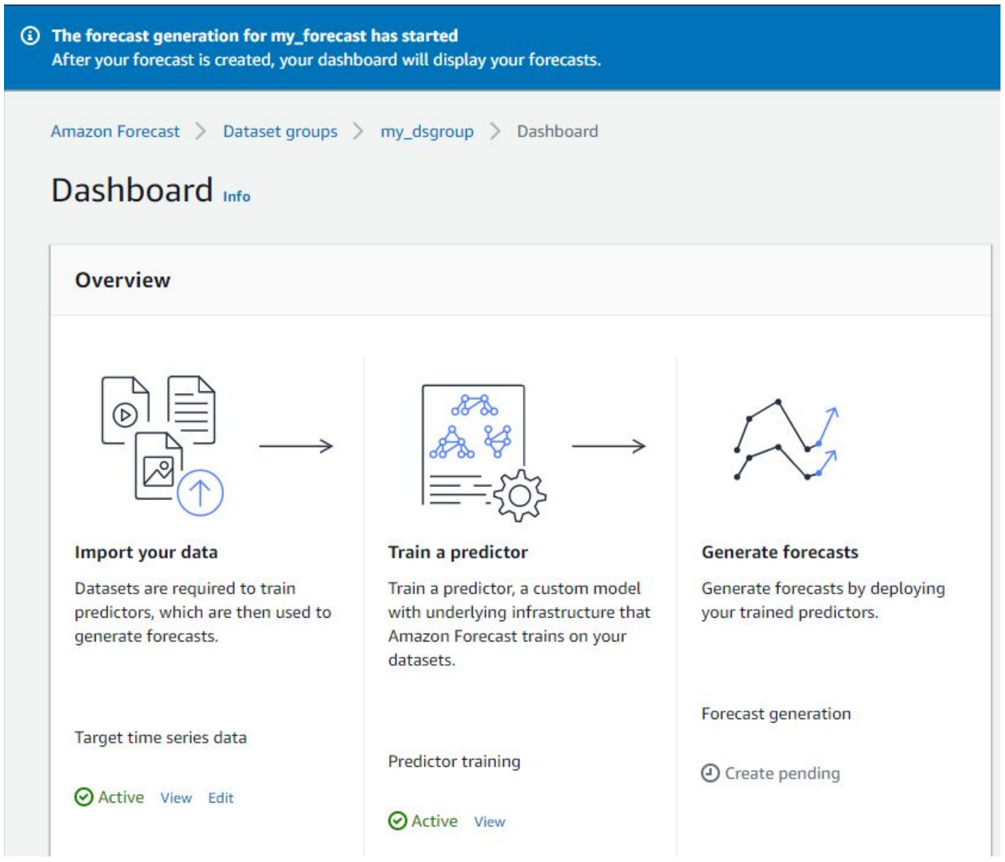
Forecast details

Forecast name
The name can help you distinguish this forecast from your other forecasts.
my_forecast
The forecast name must have 1 to 63 characters. Valid characters: a-z, A-Z, 0-9, and _

Predictor Info
The predictor that you want to use to create forecasts.
my_predictor

Forecast types - *optional* Info
Enter up to 5 quantile values between .01 to .99. You can also enter 'mean'. By default, Amazon Forecast will generate forecasts for .10, .50 and .90 quantiles.
.10, .50, .90, .99, mean
Separate forecast types with commas.

4. Choose Create a forecast. The dataset group's Dashboard page is displayed. Your screen should look similar to the following:



The forecast generation for my_forecast has started
After your forecast is created, your dashboard will display your forecasts.

Amazon Forecast > Dataset groups > my_dsgroup > Dashboard

Dashboard Info

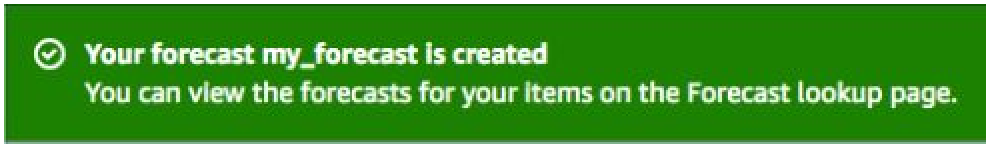
Overview

Import your data
Datasets are required to train predictors, which are then used to generate forecasts.
Target time series data
✓ Active View Edit

Train a predictor
Train a predictor, a custom model with underlying infrastructure that Amazon Forecast trains on your datasets.
Predictor training
✓ Active View

Generate forecasts
Generate forecasts by deploying your trained predictors.
Forecast generation
Create pending

Under Forecast generation, you should see the status of forecast generation. Wait for Amazon Forecast to finish creating the forecast. The process can take several minutes or longer. When your forecast has been created, the progress transitions to Active. Additionally, the banner at the top of the dashboard changes to display the following message:



Your forecast my_forecast is created
You can view the forecasts for your items on the Forecast lookup page.

Now that your forecast has been created, you can query or export the forecast.

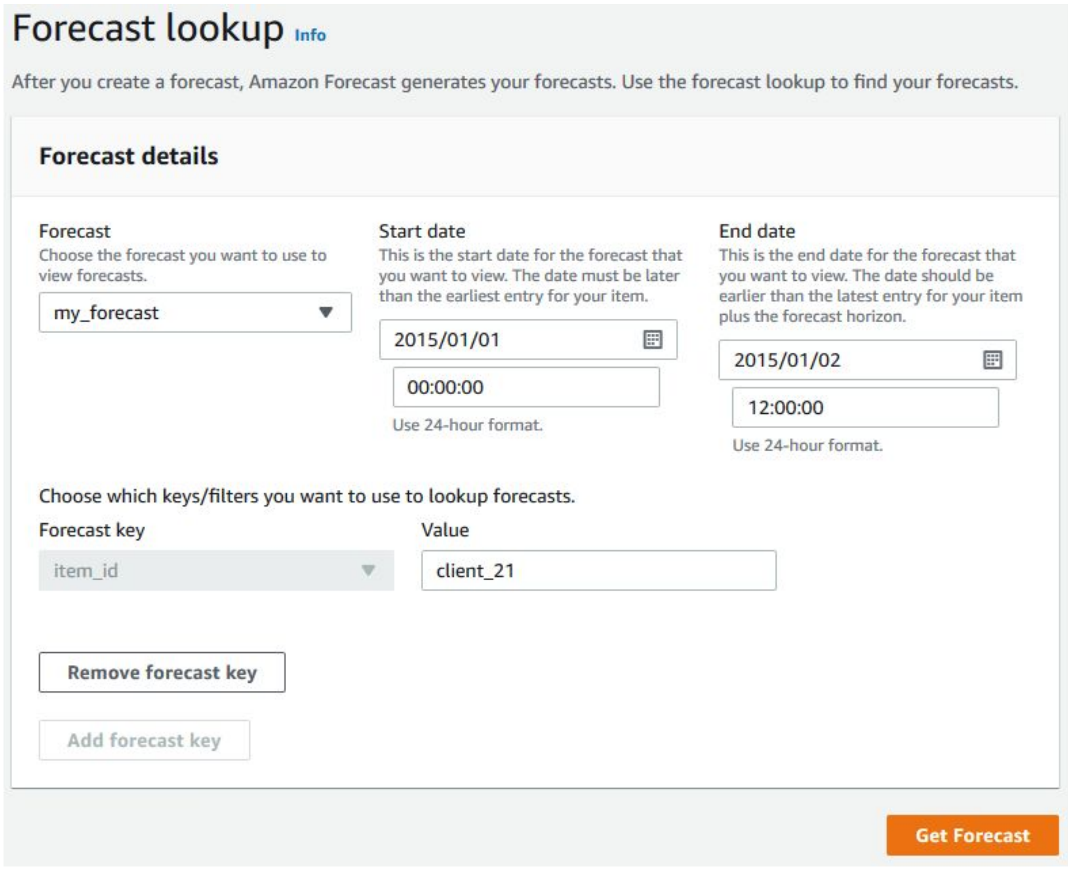
Step 4: Retrieve a Forecast

After the forecast has been created, you can query for a single item or export the complete forecast.

To query for a single item

1. If the dashboard is not displayed, in the navigation pane, under your dataset group, choose Dashboard.
2. In the Dashboard, under Generate forecasts, choose Lookup forecast. The Forecast lookup page is displayed.
3. On the Forecast lookup page, for Forecast details, provide the following information.
    - Forecast – From the drop-down menu, choose the forecast that you created in Step 3: Create a Forecast.
    - Start date – Enter 2015/01/01. Keep the default time of 00:00:00.
    - End date – Enter 2015/01/02. Change the time to 12:00:00.
      The date range of 36 hours corresponds to the Forecast horizon that you specified in Step 2: Train a Predictor.
    - Choose which keys/filters – Choose Add forecast key.
    - Forecast key – From the drop-down menu, choose item_id.
    - Value – Enter a value from the item_id column of the input time series of the electricity usage data. An item_id (for example, client_21) identifies a particular client who is included in the dataset.

4. Your screen should look similar to the following:



Forecast lookup

After you create a forecast, Amazon Forecast generates your forecasts. Use the forecast lookup to find your forecasts.

Forecast details

Forecast
Choose the forecast you want to use to view forecasts.
my_forecast

Start date
This is the start date for the forecast that you want to view. The date must be later than the earliest entry for your item.
2015/01/01
00:00:00
Use 24-hour format.

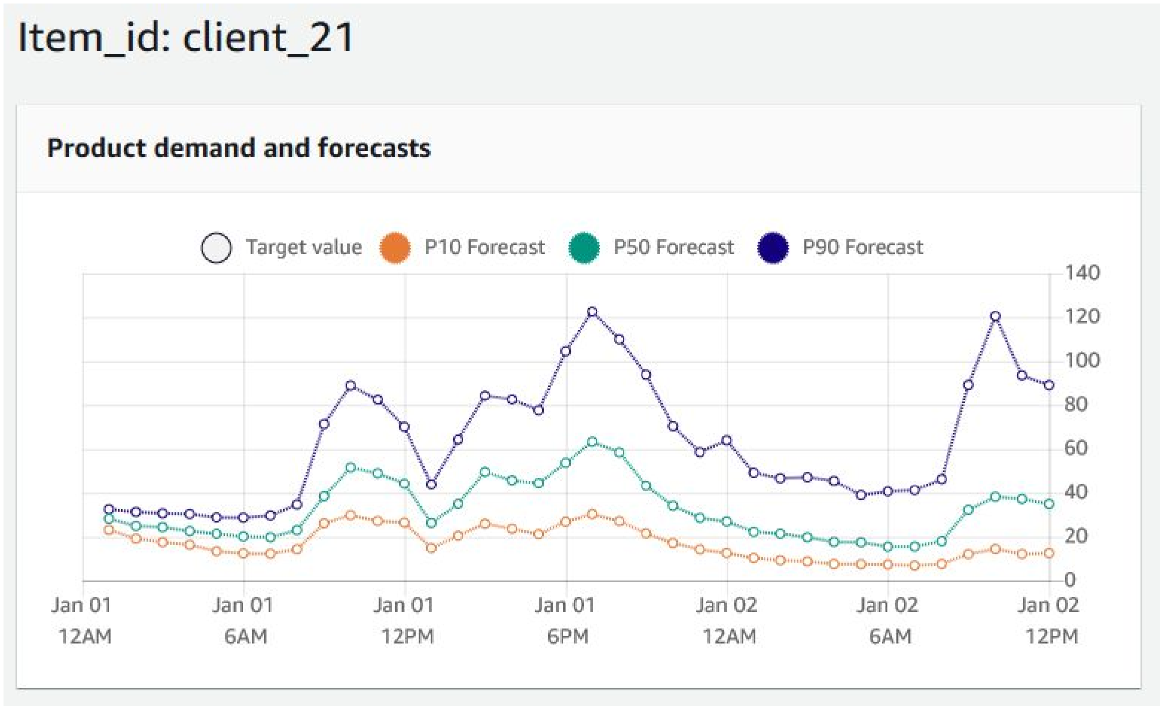
End date
This is the end date for the forecast that you want to view. The date should be earlier than the latest entry for your item plus the forecast horizon.
2015/01/02
12:00:00
Use 24-hour format.

Choose which keys/filters you want to use to lookup forecasts.

Forecast key	Value
item_id	client_21

5. Choose Get Forecast. When the forecast is displayed, review the forecast for electricity usage demand by client_21.

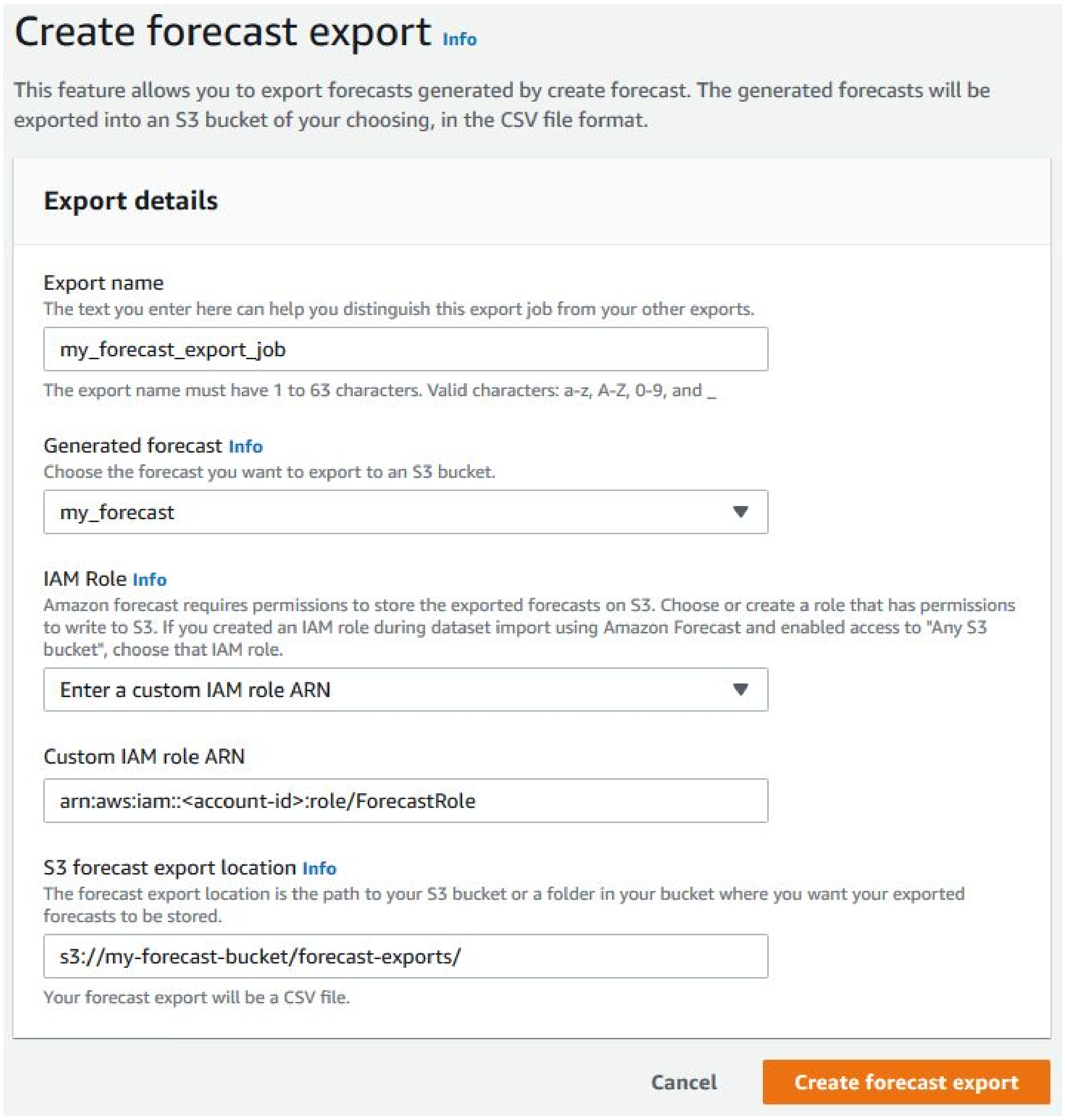
The forecast should look similar to the following:



To export the complete forecast

1. In the navigation pane, under your dataset group, choose Forecasts.
2. Choose the radio button next to the forecast that you created in Step 3: Create a Forecast.
3. Choose Create forecast export. The Create forecast export page is displayed.
4. On the Create forecast export page, for Export details, provide the following information.
    - Export name – Enter a name for your forecast export job.
    - Generated forecast – From the drop-down menu, choose the forecast that you created in Step 3: Create a Forecast.
    - IAM role – Keep the default Enter a custom IAM role ARN.
      Alternatively, you can have Amazon Forecast create the required IAM role for you by choosing Create a new role from the drop-down menu and following the on-screen instructions.
    - Custom IAM role ARN – Enter the Amazon Resource Name (ARN) of the IAM role that you created in Create an IAM Role for Amazon Forecast (IAM Console).
    - S3 forecast export location – Use the following format to enter the location of your Amazon Simple Storage Service (Amazon S3) bucket or folder in the bucket:
      s3:////

5. Your screen should look similar to the following:



Create forecast export Info

This feature allows you to export forecasts generated by create forecast. The generated forecasts will be exported into an S3 bucket of your choosing, in the CSV file format.

Export details

Export name
The text you enter here can help you distinguish this export job from your other exports.
my_forecast_export_job
The export name must have 1 to 63 characters. Valid characters: a-z, A-Z, 0-9, and _

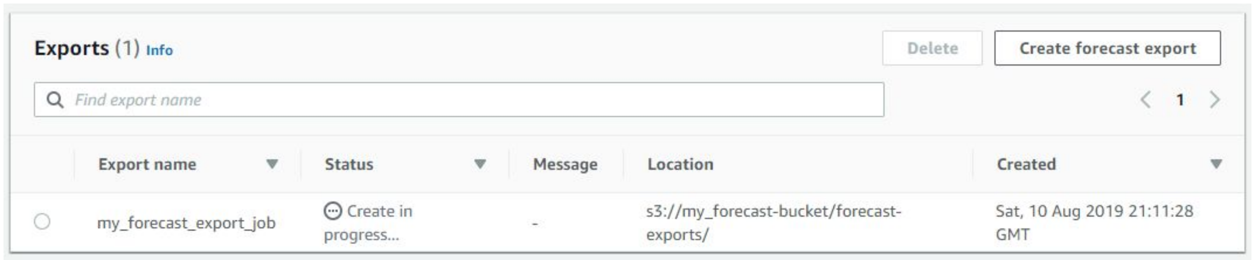
Generated forecast Info
Choose the forecast you want to export to an S3 bucket.
my_forecast

IAM Role Info
Amazon forecast requires permissions to store the exported forecasts on S3. Choose or create a role that has permissions to write to S3. If you created an IAM role during dataset import using Amazon Forecast and enabled access to "Any S3 bucket", choose that IAM role.
Enter a custom IAM role ARN

Custom IAM role ARN
arn:aws:iam::<account-id>:role/ForecastRole

S3 forecast export location Info
The forecast export location is the path to your S3 bucket or a folder in your bucket where you want your exported forecasts to be stored.
s3://my-forecast-bucket/forecast-exports/
Your forecast export will be a CSV file.

6. Choose Create forecast export. The my_forecast page is displayed.
Your screen should look similar to the following:



	Export name	Status	Message	Location	Created
○	my_forecast_export_job	Create in progress...	-	s3://my_forecast-bucket/forecast-exports/	Sat, 10 Aug 2019 21:11:28 GMT

You should see the status progress. Wait for Amazon Forecast to finish exporting the forecast. The process can take several minutes or longer. When your forecast has been exported, the status transitions to Active and you can find the forecast files in your S3 bucket.

Reference

- https://en.wikipedia.org/wiki/Python_(programming_language)
- https://en.wikipedia.org/wiki/R_(programming_language)
- https://en.wikipedia.org/wiki/JavaScript
- https://en.wikipedia.org/wiki/HTML
- https://en.wikipedia.org/wiki/Cascading_Style_Sheets
- https://en.wikipedia.org/wiki/PHP
- https://en.wikipedia.org/wiki/Apache_HTTP_Server
- https://en.wikipedia.org/wiki/Laravel
- https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)
- https://en.wikipedia.org/wiki/JQuery
- https://tobiasahlin.com/blog/introduction-to-chartjs/
- https://en.wikipedia.org/wiki/Project_Jupyter
- https://en.wikipedia.org/wiki/Agile_software_development
- https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- https://medium.com/sciforce/data-cleaning-and-preprocessing-for-beginners-25748ee00743
- https://improvado.io/blog/what-is-data-aggregation
- https://productsup.io/wp-content/uploads/2018/09/content-aggregation.png
- https://en.wikipedia.org/wiki/Time_series
- https://lh6.googleusercontent.com/3ilNt7AbjM0ahQd10tFn41s8ux_9rRIiDtVOUSUCKPqcvwc-5RZfKhA7SVD6RRIXVc2eMjncTtYIiL1wIXHy1LNBN0HsDM4kSCHjbDXvvbgI7inygKEazHrf68gsEx-2eSzmqLXT
- Python for Data Analysis, 2nd Edition book - https://www.oreilly.com/library/view/python-for-data/9781491957653
- https://en.wikipedia.org/wiki/Amazon_Web_Services
- Hyndman, R.J., & Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2. Accessed on <current date> - https://otexts.com/fpp2/
- Amazon Forecast Documentation - https://docs.aws.amazon.com/forecast/