Production-Ready Customer Satisfaction Prediction using MLOps

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Kunal Verma (201513)

Siddharth Kuthiala (201144)

Under the guidance & supervision of

Prof. Dr. Pradeep Kumar Gupta

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

Candidate's Declaration

I hereby declare that the work presented in this report entitled **Production-Ready Customer Satisfaction Prediction using MLOps** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to ~~December 2023~~ May 2024 under the supervision of **Prof. Dr. Pradeep Kumar Gupta** (Professor & Academic Coordinator, Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

[illegible] 14/5/24

(Student Signature with Date)
Student Name: Kunal Verma
Roll No.: 201513

[illegible] 14/5/24

(Student Signature with Date)
Student Name: Siddharth Kuthiala
Roll No.: 201144

This is to certify that the above statement made by the candidate is true to the best of our knowledge.

[illegible] 15/5/24

(Supervisor Signature with Date)
Supervisor Name: Prof. Dr. Pradeep Kumar Gupta
Designation: Professor & Academic Coordinator
Department: CSE & IT
Dated: 10/05/25

Acknowledgement

With immense gratitude, we extend our heartfelt thanks to the Almighty for His divine blessings, which have illuminated our path and enabled us to successfully complete the project – Customer Satisfaction Prediction using MLOps.

We extend our heartfelt appreciation to **Prof. Dr. Pradeep Kumar Gupta**, a distinguished Professor and Academic Coordinator in the Department of Computer Science and Engineering at Jaypee University of Information Technology (JUIT), Solan, HP, India. His remarkable academic journey includes a Ph.D. in Computer Science and Engineering and a Post-Doctorate from the University of Pretoria, South Africa. His extensive research contributions in areas such as Internet-of-Things, Artificial Intelligence, Cloud computing, and Fog computing, as evidenced by over 100 research papers in reputed journals and international conferences, showcase his profound expertise.

We would also like to express our gratitude to Prof. Dr. Pradeep Kumar Gupta from the Department of CSE for her valuable assistance, which significantly contributed to the successful conclusion of our project.

Our sincere thanks extend to all individuals, whether directly or indirectly, who played a role in the triumph of this project. We acknowledge the support of the entire staff, both teaching and non-teaching, whose timely assistance and facilitation greatly aided our endeavor.

In closing, we wish to recognize and appreciate the enduring support and patience of our parents. Their unwavering encouragement has been a source of strength throughout this journey.

With gratitude,

Kunal Verma

(201513)

Siddharth Kuthiala

(201144)

Table of Content

List of Figures

Abstract

Understanding and forecasting consumer happiness has become critical for businesses in the modern business environment that aim to provide outstanding goods and services. Anticipating client attitude can facilitate preemptive measures, improve customer relations, and propel business achievements. But creating a dependable, production-ready solution that precisely forecasts consumer happiness necessitates combining machine learning with operational procedures, or MLOps.

The main goal of our research is to apply MLOps concepts to develop a strong "Production-Ready Customer Satisfaction Prediction" system. It can be difficult to update machine learning models effectively, track their performance, and maintain consistent predictions while using traditional deployment methods because they frequently lead to fragmented processes. Our objective is to overcome these obstacles by streamlining the entire process with the use of state-of-the-art technologies and best practices.

By leveraging the ZenML mode framework, we aim to establish a well-structured and automated CI/CD pipeline that handles the following tasks:

- Data pre-processing
- Model Training
- Model Evaluation

The main purpose of doing this is to eliminate manual intervention, reduce errors, and ensure that our model is always up-to-date with the latest data.

Furthermore, we recognize the importance of showcasing the value of our predictive model. To this end, we intend to use the Streamlit application that allows stakeholders to interact with and visualize the predictions. This application will serve as a tangible representation of the model's capabilities and foster better decision-making processes.

Security is of paramount concern in any machine learning application, and thus, we will incorporate an open appsec engine to ensure that our model and application remain safeguarded against potential vulnerabilities.

In summary, our project aspires to bridge the gap between machine learning development and operational deployment by creating a holistic solution that integrates advanced MLOps practices. By doing so, we endeavor to empower businesses to make informed decisions, enhance customer experiences, and ultimately achieve higher levels of satisfaction and success.

Chapter 1: Introduction

1.1 Introduction

In today's business model, where customer satisfaction is critical to success, the hybridization of machine learning and operational practices, known as MLops, emerges as an important methodology in furthering this process in a sophisticated production-ready customer satisfaction prediction system. By leveraging the power of advanced technology and MLOps principles of strategic recognition, our team seek to return what has been defined traditional approaches to predictive modeling, ultimately empowering organizations to make data-driven decisions that maximize customer satisfaction

1.2 Problem Statement

To address the core problem statement of our project, the primary objective is to develop a robust predictive machine learning model for anticipating the customer satisfaction score associated with a future order or purchase. In this project, we aim to leverage the dataset focused towards the Brazilian market, which contains over 100,000 orders during the duration of 2020 to 2022. This dataset consists of various features, which includes:

- order details,
- prices,
- customer location,
- review comments etc.

By leveraging these particular features and powerful machine learning algorithms, we aim to build a predictive model capable of extrapolating the probable satisfaction score for the future orders.

To practically implement our idea, we have chosen to employ ZenML, which is a powerful tool for orchestrating end-to-end machine learning pipelines. By integrating ZenML into our workflow, we aim to construct a production-ready pipeline that seamlessly translates our model from conception to deployment. This ensures not only the accuracy of our predictions but also positions our solution for real-world applicability, thereby addressing the dynamic nature of the e-commerce landscape. In essence, our report will detail the evolution of our methodology, the insights gained from the dataset, the intricacies of our predictive model, and the practical implications of using ZenML to construct a pipeline geared for deployment in real-world scenarios.

1.3 Objectives

The objectives of this project are intricately designed to address key challenges in predictive modeling and operational deployment, aiming to create a robust "Production-Ready Customer Satisfaction Prediction" system. Each objective is strategically formulated to contribute to the project's overarching goal of integrating machine learning and operational practices through advanced MLOps principles.

1. **Holistic Model Development:**
   Employ state-of-the-art machine learning techniques for the development of an accurate customer satisfaction prediction model. Thoroughly evaluate and fine-tune the model's performance to ensure optimal accuracy and reliability. Implement interpretability measures to enhance stakeholders' understanding of the model's decision-making process.

2. **Streamlined Data Preprocessing:**
   Develop and implement robust data preprocessing procedures to handle diverse data sources and ensure data consistency. Utilize advanced techniques for feature engineering to enhance the model's ability to capture nuanced customer sentiments. Implement data quality checks and validation processes to maintain the integrity of input data.

3. **Establishment of a ZenML-Powered CI/CD Pipeline:**
   Design a CI/CD pipeline using the ZenML framework to automate end-to-end workflows, from data preprocessing to model deployment. Integrate version control to track changes, enabling seamless collaboration and ensuring reproducibility. Implement continuous integration to automatically validate changes, minimizing errors and reducing the time required for deployment.

4. **Transparency and Reproducibility via MLFlow:**
   Employ MLFlow for model deployment, ensuring transparent and reproducible model deployment practices. Log and monitor model metrics, hyperparameters, and artifacts, facilitating comprehensive tracking and auditing. Implement an effective versioning strategy to manage model updates and iterations seamlessly.

5. **Interactive Streamlit Application Development:**
   Craft a user-friendly Streamlit application for stakeholders to intuitively interact with and visualize customer satisfaction predictions. Enable real-time updates to empower stakeholders in exploring the dynamic nature of the model's outputs. Enhance usability with intuitive visualizations and user-friendly interfaces to enrich the stakeholder experience.

6. **Open AppSec Integration for Enhanced Security:**
   Integrate an open application security (AppSec) engine to fortify both the customer satisfaction prediction model and the associated application. Conduct rigorous security assessments for the identification and mitigation of potential vulnerabilities. Implement secure coding practices and continuous security monitoring to ensure ongoing resilience against emerging threats.

1.4 Significance and Motivation of the Project Work

The significance of this project lies in its potential to revolutionize how organizations approach customer satisfaction prediction. By bridging the gap between machine learning development and operational deployment, the project offers a comprehensive solution to the challenges faced by businesses in maintaining predictive models. The motivation stems from the belief that a well-implemented solution, integrating advanced MLOps practices, can fundamentally transform decision-making processes.

This project aspires to contribute to the success and competitiveness of modern enterprises by empowering them to make informed decisions, enhance customer experiences, and ultimately achieve higher levels of satisfaction. The motivation is grounded in the understanding that such a solution can be a game-changer in the business landscape, fostering innovation and resilience in the face of evolving market dynamics.

Chapter 2: Literature Survey

2.1 Overview of Relevant Literature

In this section, we delve into the existing research landscape related to the optimization of customer satisfaction prediction systems. We examine the progress made in areas such as user interface design, security protocols, and automated evaluation mechanisms, aiming to draw insights and inspiration for the development of our production-ready solution.

R. Ibne Sattar et al. [6] significantly improved the security of the online examination system by putting in place a blockchain-based framework. By utilizing IP-based login, blockchain technology, and authentication methods, the framework showed improved resistance to fraud and unauthorized access. This demonstrates how well these combined methods work to strengthen the security of online exams.

J. Heizg et al. [7] used early stage prediction and trained the model on twitter conversational data and 2 customer support datasets while utilizing the text analysis and gained an improvement of 30% in F1-Score.

N. Zeinalizadeh et al. [1] has effectively identified and ranked the aspects that influence total bank customer happiness. When it comes to forecasting customer happiness, the created Artificial Neural Network (ANN) model outperforms a linear regression model. Setting priorities for factors—Fees and Loans, Prompt Service, and Appearance being the most important—provides useful information for the banking sector's strategic decision-making. In general, the research advances knowledge about and enhances client happiness in a cutthroat banking sector.

C. Gurău et al. [2] discussed the crucial difficulty of controlling consumer profitability in markets with intense competition. In order to determine, forecast, and improve customer profitability, the suggested model incorporates ideas like client lifetime value, segmentation, loyalty, and data mining. The study highlights the significance of marketing-mix tactics that are customized for loyal and high-value clients. It also highlights the management and organizational challenges of implementing this strategy in changing market settings.

S. Kumar et al. [3] uses Twitter input to analyze machine learning techniques to improve customer experience in the airline business. The paper evaluates CNN, SVM, and several ANN designs for tweet categorization using features derived from tweets using word embedding and n-gram techniques. CNN performs better than SVM and ANN models, according to the results. Association rule mining also exposes interesting connections among various tweet categories, giving airline industry useful information to enhance the traveler experience.

L. C. Siebert et al. [4] shows a close agreement with survey-based indicators when using machine learning approaches to forecast customer satisfaction for power distribution businesses. With a little 1.36% deviation, the technique provides a prompt means of detecting and resolving disruptions in customer satisfaction. Creating an intelligent algorithm is a useful invention that may help businesses better understand the effect of customer happiness and continuously improve performance.

A. Shatnwai et al. [5] a customer retention model that addresses skewed datasets by utilising several oversampling techniques and the XGBoost machine learning classifier. The outcomes demonstrate encouraging performance in comparison to other classifiers, providing businesses across several industries with a practical approach to preserving client loyalty and upholding expansion and financial gain.

P. Kunekar et al. [8] classification algorithms to pinpoint the major elements affecting airline passenger happiness. The best accuracy is produced by the LightGBM classifier, which emphasizes the importance of age, flight distance, customer type, in-flight Wi-Fi, and mode of travel. These results may be deliberately applied by airlines to raise customer happiness and boost overall performance.

A. Kohei et al. [9] presents an approach for estimating churn customers that makes use of logistic regression and LightGBM. The suggested approach outperforms the traditional GradientBoostingClassifier (GBC) by around 10% in the prediction and analysis of churn client attributes, providing better sales insights.

Y. Lu [10] shows how China Mobile is improving customer happiness by focusing on 5G apps. XGBoost and lightgbm are two machine learning techniques that are used in conjunction with data preprocessing to create an efficient satisfaction prediction model. For the test set, the XGBoost approach generates a plausible and practically meaningful confusion matrix graphic.

J. Ren et al. [11] presents AppSec, a secure execution environment built on top of a hypervisor that shields user interactions and sensitive application data from untrusted operating systems. AppSec protects against runtime changes, checks kernel memory access, separates devices used for human-machine interaction, and offers a privileged-based window system in addition to ensuring code integrity. A trusted I/O path, mediation of kernel memory access, and runtime verification of dynamic shared objects are some of the main benefits. The working prototype showcases the effectiveness and usefulness of AppSec.

M. Alenezi et al. [12] examined seven open-source online apps built on Java for static security flaws, finding recurring problems most likely brought on by rushed development or a lack of security expertise. It suggests creating an intelligent framework to resolve vulnerabilities, update missing code, offer recommendations for secure development, and gain knowledge from professional practices for efficient security mitigation.

W. J. van den Heuvel et al. [13] offers a unique platform for ML-Ops and transparent model-driven development in smart corporate apps. By highlighting AI/ML Blueprints as first-class entities, it guarantees portability and end-to-end transparency. The framework, which includes continuous deployment and monitoring, is based on the intelligent Application Architecture (iA2) and logically distinguishes between simple data processing and sophisticated capabilities. Key findings are summarized, an open research agenda is outlined, and a fundamental metamodel for blueprint-model-driven MLOps in iA2 applications is introduced in the conclusion.

L. E. Lwakatare et al. [14] highlights the difficulties of incorporating machine learning (ML) procedures into contemporary software development, such as DevOps, and discusses the complexity of producing software using ML components. It draws attention to the absence of defined protocols for this integration. The suggested fixes entail merging the workflow procedures for ML and DevOps. Presentations of real-world examples are made to highlight problems and possible fixes in this area.

E. Puica [15] evaluates the effectiveness of Supply Chain Management (SCM) technology initiatives utilising Azure. It makes use of a number of efficiency metrics and emphasises Azure's contribution to resolving operational issues using machine learning in supply chain management. The research helps managers make decisions by effectively assessing the productivity growth and technical efficiency of IT systems in SCM. The results aid in determining which IT solutions are more effective in various SCM areas.

2.2 Key Gaps in the Literature

A review of the literature reveals some significant gaps and limitations that are present in different related works. Although it seems promising, putting a blockchain-based system into place requires specialized knowledge and resources due to its technical complexity. The smooth operation of blockchain-based solutions is further hampered by issues like expensive fees during network congestion and slow transaction confirmation times.

Chapter 3: System Development

3.1 Requirements and Analysis

In this crucial phase of system development, the thorough identification and analysis of requirements lay the foundation for a successful "Production-Ready Customer Satisfaction Prediction" system. The requirements are categorized into two distinct facets: functional requirements and non-functional requirements.

3.1.1 Functional Requirements:

The functional requirements represent the specific features and functionalities the system must possess to meet the outlined objectives.

1. **Data Collection and Preprocessing:**
   The system should seamlessly collect and integrate customer data from diverse sources, including feedback forms, surveys, and customer interactions. Implement robust data preprocessing mechanisms to handle missing data, outliers, and ensure data consistency for accurate model training.

2. **CI/CD Pipeline Automation:**
   Developing a CI/CD pipeline using the ZenML framework in order to automate the end-to-end workflow which includes:
   - Data Preprocessing,
   - Model Training,
   - Validation

   Ensure the pipeline allows for version control, enabling effective collaboration and reproducibility of each stage of the model development process.

3. **Model Deployment with MLFlow:**
   Implement MLFlow for model deployment to ensure transparency, versioning, and efficient monitoring of key metrics and parameters. Enable the seamless integration of MLFlow with the CI/CD pipeline to maintain a consistent and reproducible deployment process.

4. **Streamlit Application Development:**
   Design an intuitive Streamlit application that allows stakeholders to interact with and visualize customer satisfaction predictions. Ensure real-time updates and dynamic visualization to enhance the user experience and facilitate a deeper understanding of model outputs.

5. **Security Integration:**
   Integrate an open application security (AppSec) engine to fortify both the predictive model and the associated application. Conduct regular security assessments to identify and address potential vulnerabilities, ensuring the system's resilience against security threats.

3.1.2 Non-Functional Requirements:

Non-functional requirements encompass aspects such as system performance, reliability, and security. Given the project's emphasis on MLOps and customer satisfaction prediction, the non-functional requirements are as follows:

1. **Scalability:**
   The system should be scalable to accommodate an increasing volume of customer data without compromising performance. The CI/CD pipeline and deployment process should be designed to scale seamlessly with the growth of data and user interactions.
2. **Reliability:**
   The system must ensure high reliability, minimizing downtime and disruptions to maintain continuous availability for stakeholders. Implement automated monitoring and alerting mechanisms to promptly address and rectify any unforeseen issues.

3. **Performance Optimization:**
   Optimize the performance of the predictive model to ensure timely and accurate customer satisfaction predictions. Regularly assess and enhance the efficiency of the CI/CD pipeline for swift and error-free model updates.

4. **User Experience (UX):**
   Prioritize an intuitive and engaging user experience in the Streamlit application to encourage stakeholder interaction. Ensure that visualizations are user-friendly, facilitating a clear understanding of the model's insights.

5. **Security Measures:**
   Uphold high-security standards through encryption, access controls, and secure coding practices. Regularly update security protocols and conduct periodic security audits to address emerging threats and vulnerabilities.

MLOps Methodology: Bridging the Gap Between Development and Operations

What is MLOps Methodology?

MLOps, a portmanteau of "Machine Learning" and "Operations," refers to a set of practices and principles that aim to integrate machine learning (ML) systems seamlessly into the broader realm of operational practices. It serves as a bridge, facilitating collaboration between data scientists, who focus on model development, and operations teams responsible for deploying and maintaining these models in real-world environments. MLOps includes a comprehensive set of practices, along with version manipulation, continuous integration and deployment (CI/CD), monitoring, and automation, to streamline a given machine learning life cycle.
This method strives to triumph over the challenges related to deploying ML models at scale, making sure that the fashions are not simplest correct for the duration of improvement but additionally keep their performance and reliability in operational settings.

Why do we need MLOps in production?

The inclusion of MLOps in a production system is beneficial in several ways, which are as follows:

- Solution to challenge of transitioning machine learning models from development to deployment stages
- Machine learning model requires continuous adaptation to new data which is achievable with a continuous automation deployment pipeline
- MLOps helps to foster collaboration and communication between the data engineers, scientists and the operations team, thus helping in accelerating the overall life cycle of the end product.

How Does MLOps Work?

The workings of MLOps are grounded within the systematic integration of machine learning and industry level operational practices. It begins with the status quo of a sturdy CI/CD pipeline, which automates the whole ML workflow, from data preprocessing and model training using algorithms to deployment and metrics tracking. This pipeline guarantees that any changes in statistics or model structure are seamlessly integrated, presenting a regular and reproducible deployment procedure. MLOps also carries model control, allowing the tracking of changes in both code and models. Continuous monitoring ensures that deployed models carry out optimally, with the ability to come across anomalies or degradation in real-time. Furthermore, MLOps emphasizes collaboration, ensuring that feedback from operational teams informs enhancements in model development, creating a cyclical technique that iteratively enhances both development and deployment practices.

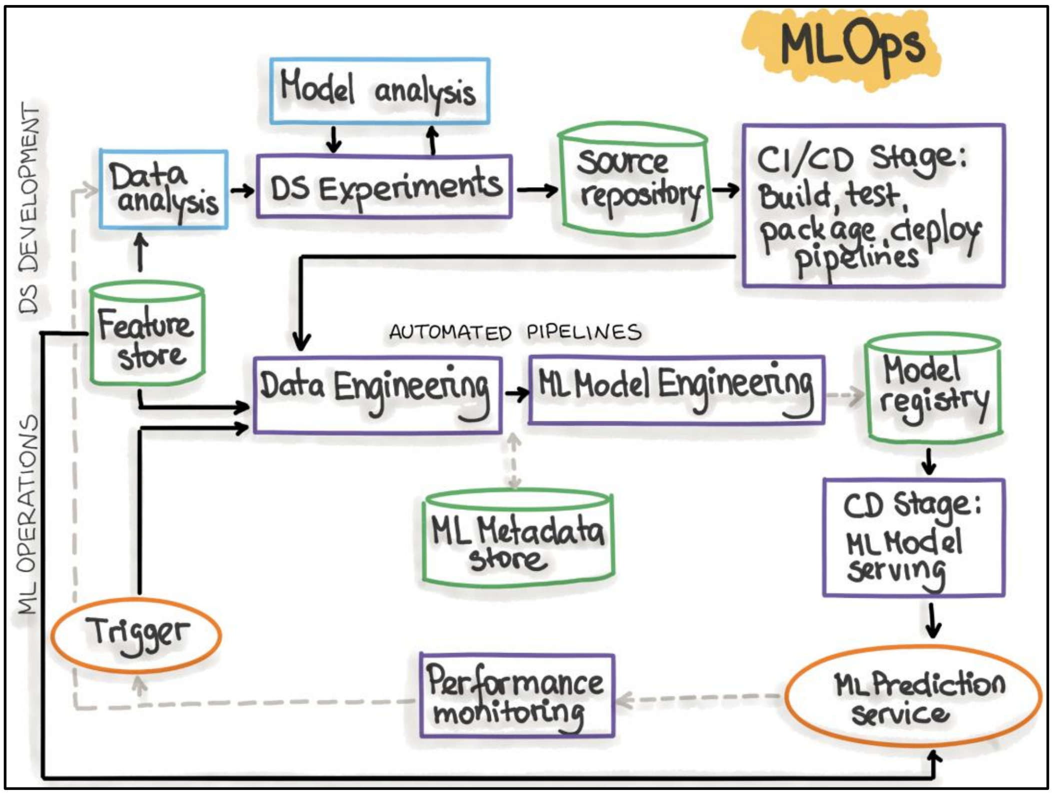


Figure 1 - MLOps Stages [11]

The figure above (Fig. 1) shows the different stages in a typical MLOps pipeline. The various stages include: Development & Experimentation, Pipeline Continuous Integration, Pipeline Continuous Delivery, Automated Triggering, Model Continuous Delivery, Monitoring.

3.2 Project Design and Architecture

At the core of our project, we aim to integrate machine learning operational practices with powerful algorithms in order to achieve our objective for this particular project. In this section, we will dive into the entire project design and the architectural workflow for the project.

Overall Architecture: Training and Deployment Pipelines
Training Pipeline:

The training pipeline is one of the most important parts in our entire architecture. Below are the several steps that it consists:

1. We begin with the **ingest_data step** wherein the raw data from the dataset is ingested and processed into a pandas DataFrame.
2. The **clean_data step** follows, systematically removing unwanted columns and ensuring data cleanliness.

3. The **train_model step** employs MLflow autologging to train the model, capturing hyperparameter values and model artifacts for comprehensive tracking.
4. The **evaluation step** assesses model performance and logs evaluation metrics, contributing valuable insights to the MLflow experiment tracking.

Deployment Pipeline:

This is one of the most interesting parts of the entire pipeline. The deployment pipeline's main purpose is to establish a continuous deployment workflow. The entire deployment pipeline can be found in the **deployment_pipeline.py** file, which consists of a series of stages that are also included in the training pipeline. Towards the end of the deployment pipeline, we have certain model evaluation to evaluate the model performance. Those are as follows:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R2 Score

In both pipelines, the ZenML framework's MLflow integration is leveraged for meticulous tracking. Hyperparameter values, trained model artifacts, and evaluation metrics are logged into the local MLflow backend, ensuring transparency and reproducibility. The Deployment Pipeline also features a local MLflow deployment server, serving as a dynamic endpoint for the latest model meeting accuracy criteria. This server operates as a daemon process, persisting beyond individual pipeline executions, and seamlessly updating to serve newly trained models that surpass the accuracy threshold.

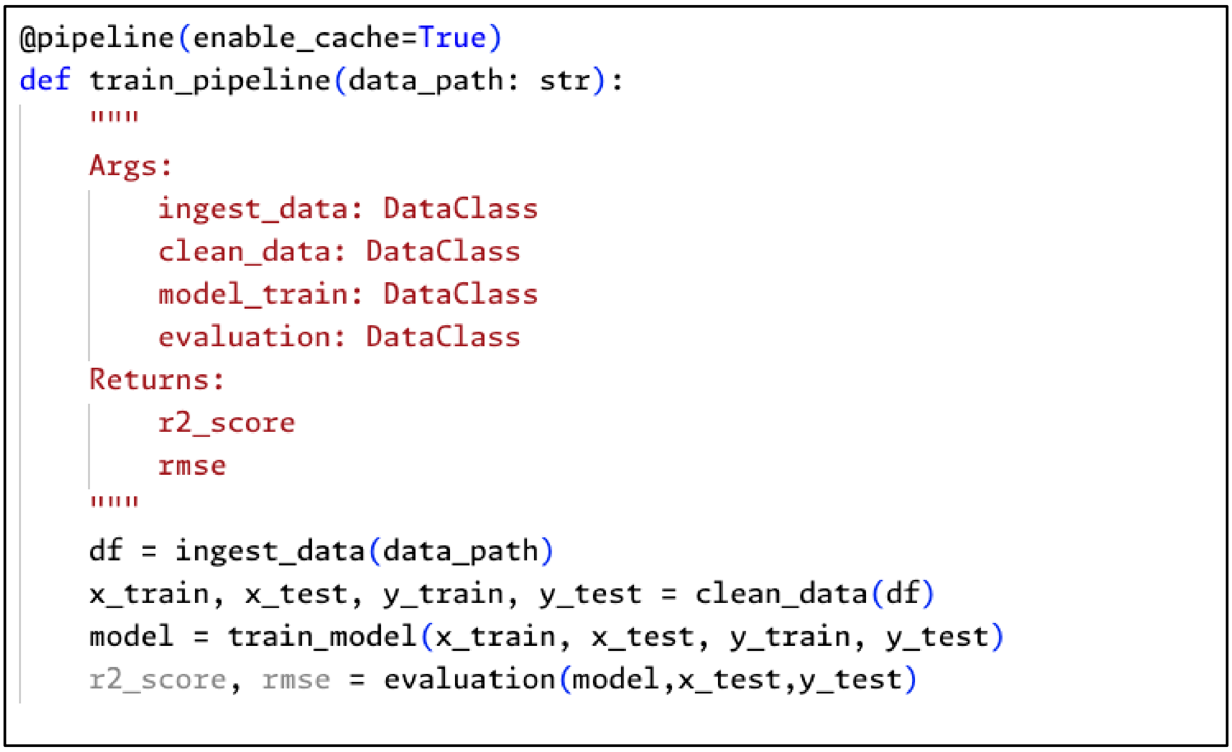
Current Progress on the Architecture

In the current project phase, our architecture now encompasses both the Training and Deployment Pipelines, marking a significant milestone in our MLOps journey. The meticulous design of these pipelines ensures a solid foundation for model development, evaluation, and seamless deployment into production environments. With both pipelines in place, we've established a comprehensive end-to-end workflow, facilitating continuous improvement and adaptation to evolving data while ensuring the smooth transition of models from development to deployment stages.

The iterative nature of our pipelines allows us to refine our models continuously, leveraging the latest data and insights to enhance performance and accuracy. The Deployment Pipeline, in particular, streamlines the process of deploying models into real-world applications, ensuring that our predictive systems are readily available for operational use.

Embedded within these pipelines are robust tracking mechanisms that provide detailed insights into the model's performance at every stage, promoting transparency and facilitating informed decision-making. This transparency fosters collaboration between development and operational teams, enabling us to address challenges swiftly and effectively.

As we move forward, the synergistic relationship between the Training and Deployment Pipelines will further optimize our MLOps workflow, ultimately realizing our vision of a production-ready customer satisfaction prediction system that delivers tangible value to our stakeholders.



```python
@pipeline(enable_cache=True)
def train_pipeline(data_path: str):
    """
    Args:
        ingest_data: DataClass
        clean_data: DataClass
        model_train: DataClass
        evaluation: DataClass
    Returns:
        r2_score
        rmse
    """
    df = ingest_data(data_path)
    x_train, x_test, y_train, y_test = clean_data(df)
    model = train_model(x_train, x_test, y_train, y_test)
    r2_score, rmse = evaluation(model,x_test,y_test)
```

Figure 2 - Training Pipeline Stages

The figure above (Fig. 2) shows the different stages of the MLOps training pipeline defined by us in Python. The stages include:

- Ingesting the data from the dataset
- Cleaning the data ingested from the dataset
- Training the model using the selected mode (either Linear Regression, LightGBM or Random Forest)
- Evaluating the model using either MSE, R2 Score or RMSE

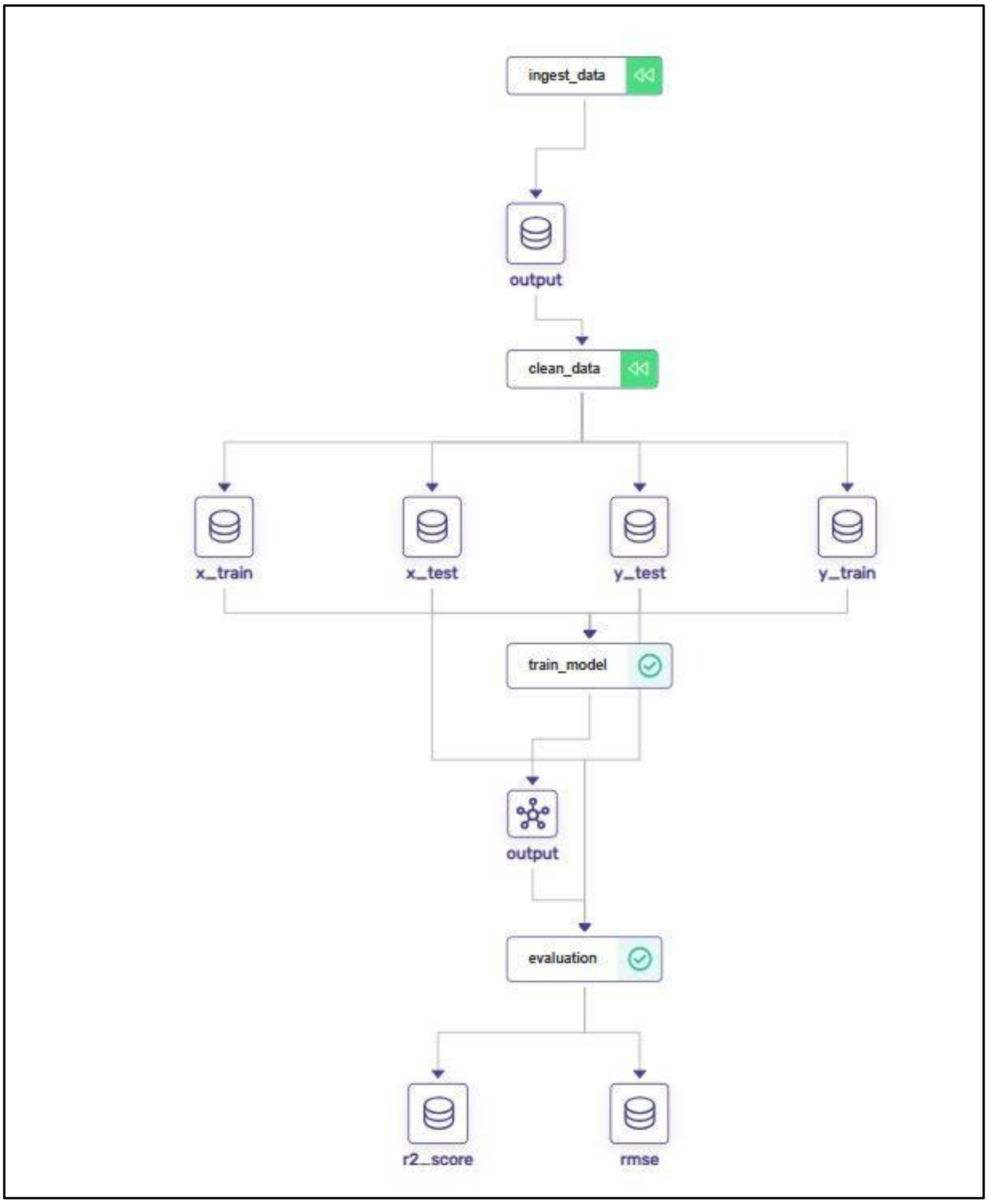
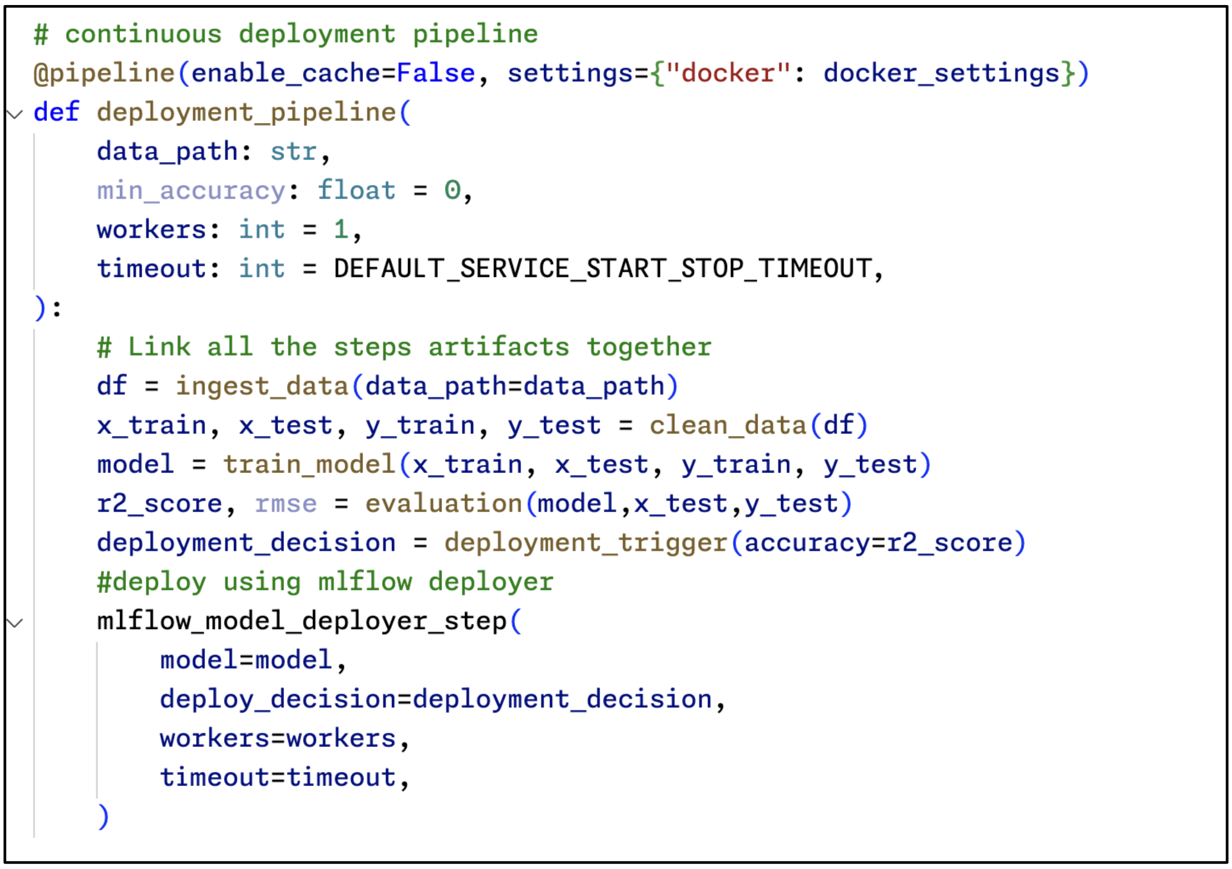


Figure 3 - ZenML Model Workflow

The figure above (Fig. 3) shows the overall data ingesting, data cleaning, model training and model evaluation stages, generated by the ZenML platform.



```python
# continuous deployment pipeline
@pipeline(enable_cache=False, settings={"docker": docker_settings})
def deployment_pipeline(
    data_path: str,
    min_accuracy: float = 0,
    workers: int = 1,
    timeout: int = DEFAULT_SERVICE_START_STOP_TIMEOUT,
):
    # Link all the steps artifacts together
    df = ingest_data(data_path=data_path)
    x_train, x_test, y_train, y_test = clean_data(df)
    model = train_model(x_train, x_test, y_train, y_test)
    r2_score, rmse = evaluation(model,x_test,y_test)
    deployment_decision = deployment_trigger(accuracy=r2_score)
    #deploy using mlflow deployer
    mlflow_model_deployer_step(
        model=model,
        deploy_decision=deployment_decision,
        workers=workers,
        timeout=timeout,
    )
```

Figure 4 - Deployment Pipeline Stages

The figure above (Fig. 4) shows the different stages of the MLOps deployment pipeline defined by us in Python. The stages include:

- **Ingest Data:**
    - This stage involves fetching and loading the data needed for the machine learning model from the specified data path.
- **Clean Data:**
    - Here, the data is preprocessed to remove any inconsistencies or errors, ensuring it's ready for training.
- **Train Model:**
    - The cleaned data is used to train the machine learning model, preparing it to make predictions based on the provided input.

- **Evaluate Model:**
    - After training, the model's performance is evaluated using evaluation metrics such as R2 score and RMSE (Root Mean Square Error).
- **Deployment Trigger:**
    - This stage determines whether the model meets the minimum accuracy threshold required for deployment.
- **Model Deployment:**
    - If the model meets the accuracy threshold, it's deployed using the MLflow deployer, enabling it to be utilized for making predictions in real-world scenarios.

3.3 Data Preparation

In this particular section, we'll explore the very first crucial stage in our project, which is - Preparing the Dataset. This process involves exploring the dataset parameters and features, data ingestion, and the subsequent data cleaning processes.

Dataset Exploration

For this project, we have selected a dataset that centers on the Brazilian market. Our dataset includes a wide range of features that capture different aspects of customer orders. These features encompass crucial details such as

- **order information (order_id, order_status, order timestamps)**,
- **customer details (customer_id, customer_unique_id, customer location)**,
- **payment details (payment_type, payment_value)**,
- **product-specific details (product_id, price, freight_value, product dimensions, category, etc.)**.

Also, the inclusion of features like review scores and review comments will definitely provide valuable insights into the prediction of customer satisfaction.

Reasons for choosing this dataset

There are several considerations that we took into account in order to choose this particular dataset.

Firstly, the pure focus on the Brazilian market aligns with the first iteration of this particular project. By focusing on one particular region, our model is tailored to work on this specific market which will ensure effectiveness and applicability in the future work.

Furthermore, the decision to work with this dataset is influenced by pragmatic considerations. As of the current project phase, we face constraints related to software requirements for handling large datasets. This dataset strikes a balance, allowing us to effectively implement and iterate on the project without compromising the quality and efficiency of our model development process. As the project advances, the insights gained from working with this dataset will inform future iterations and potential adaptations for broader market applications.

Data Ingestion and Data Cleaning Processes

These are the fundamental phases of the data preparation process. Let us have a look at them:

- The data ingestion process can be found implemented through the IngestData class and the corresponding ZenML pipeline step - ingest_data function. In this particular process, we are feeding the dataset to our machine learning pipeline in order to further process the data before applying the algorithm itself.
- After the data ingestion process, comes another crucial step which is the data cleaning itself. Here, we are focused on making sure unnecessary columns aren't included in the data which will be feeded to the machine learning model, in order to avoid noise in the data and in turn low accuracy of the entire model.

3.4 Implementation

In this particular section, we will dive deep into each step of the process which is involved in creating this particular project and making it reach to v1.0. We will explore all the steps ranging from data ingestion process, data cleaning process, to applying the machine learning algorithms to train the data and at last, evaluating the model for accuracy and lastly, deploying the project using MLFlow.

3.4.1 Data Ingestion

In the realm of machine learning, data ingestion is a fundamental process that involves the acquisition and integration of raw data into a format suitable for model training. This process is analogous to laying the groundwork for subsequent analysis, as the quality and integrity of the dataset profoundly impact the efficacy of the machine learning model. Effective data ingestion encompasses tasks such as loading data from diverse sources, handling missing or incomplete records, and ensuring data consistency.

In this, we define a class, **IngestData**, which is designed to ingest data from a specified source, returning a pandas DataFrame. This class is then instantiated in the ingest_data function, which serves as a ZenML step—a modular unit within the ML pipeline.

In the **IngestData** class, the constructor (__init__) initializes the class instance with the provided data path, indicating the location of the dataset CSV file. The get_data method within the class utilizes the pandas library to read the CSV file located at the specified path, converting it into a DataFrame. Before the data is read, an informational log statement is issued, indicating the initiation of the data ingestion process. This logging practice enhances transparency, aiding in the tracking and monitoring of pipeline execution.

The **ingest_data** function, decorated with **@step** from ZenML, encapsulates the entire data ingestion process. It catches any exceptions that may occur during the data ingestion, logging an error message if an exception is encountered and subsequently raising the exception. This error logging mechanism ensures that any issues during the data ingestion process are immediately identified and can be addressed, maintaining the integrity of the data pipeline.

Overall, this code segment establishes a modular and scalable approach to data ingestion. The encapsulation of data ingestion functionalities within a class allows for easy maintenance and extension, while the logging mechanisms contribute to the traceability and debugging capabilities of the overall MLOps pipeline. The integration of ZenML steps facilitates the seamless inclusion of data ingestion within the broader machine learning workflow, ensuring a systematic and reproducible approach to acquiring the dataset for subsequent model training and evaluation.

3.4.2 Data Cleaning

This is one of the most crucial steps in the field of machine learning and even artificial intelligence. This particular process involves identifying and addressing the issues in the chosen dataset, which could affect the predictions and accuracy of the overall model.

In our implementation, it incorporates a strategy pattern with an abstract class, **DataStrategy**, and two concrete strategy classes, **DataPreprocessStrategy** and **DataDivideStrategy**. The data cleaning operations encompass removing unnecessary columns, filling missing values, converting data types, and dividing the dataset into training and testing sets.

Data Preprocessing Strategy

The data preprocessing class includes the pre-processing steps which are important for the dataset's quality and relevance:

- **Column Removal**

  In this particular step, irrelevant columns are removed which include: **"order_approved_at","order_delivered_carrier_date", "order_delivered_customer_date","order_estimated_delivery_date", "order_purchase_timestamp".**

- **Missing Null Values**

  The null values in the numeric columns: are filled with the median values of their respective columns. This step will ensure that the missing values will not compromise the model integrity.

- **Handling Review Comment Messages**

  In this step, the null values in the "review_comment_message" column are replaced with the placeholder - "No review", which as a result will ensure consistency in the missing comment values.

- **Data Type Conversion**

  In this step, the numeric data types are selected which as a result retains only the columns with numeric values.

- **Additional Column Removal:**

  Columns deemed less important for model training, such as **"customer_zip_code_prefix"** and **"order_item_id"** are dropped to further streamline the dataset.

Data Divide Strategy

The **DataDivideStrategy** class implements the data division process, separating the dataset into training and testing sets:

1. **Feature-Target Split:**

   The dataset is divided into features (X) and the target variable (y), where **"review_score"** serves as the dependent variable.

2. **Train-Test Split:**

   Utilizing the popular **train_test_split** function from scikit-learn, the data is divided into training and testing sets. The split ratio is set to **80:20**, ensuring a substantial portion of data for training while allowing for robust model evaluation.

Data Cleaning Execution

The final step in this entire data cleaning stage is executing both the above processes:

- **Initialization**

  This class is initialized with the original dataset and the selected data cleaning strategy/
- **Handling Data:**

  Here, the handle_data method executes the chosen strategy, which is based on the past initialization of the classes itself.

This approach to data cleaning will allow a systematic and adaptable strategy pattern, which in turn promotes code maintainability and allows future adjustments to the data cleaning process.

3.4.3 Model Development

Model training is a key step in machine learning where a predictive model learns patterns and relationships in the training data to make accurate predictions based on unseen data. The process involves presenting the algorithm with labeled examples, allowing it to iteratively adjust its internal parameters. The goal is to minimize the difference between predicted results and actual values, allowing the model to generalize well to new, unprecedented data.

Model Algorithms being Used

In our project, we have used three major machine learning algorithms to help in understanding the possibilities in the diverse nature of the customer satisfaction prediction task. These algorithms include:

1. **Linear Regression**

   Linear regression is a data analysis technique that predicts the value of unknown data by using another related and known data value. Using a linear equation, it represents the unknown, or dependent, and the known, or independent, variables mathematically. Assume, for example, that you have information from last year regarding your income and expenses. After analyzing this data using linear regression techniques, you can conclude that your expenses equal half of your income. Then, by halving a future known income, they compute an unknown future expense.

2. **LightGBM Boosting**

   A prominent machine learning algorithm called Gradient Boosting Decision Trees, or GBDTs, have efficient implementations like XGBoost, and many optimization approaches are really derived from them. As the number of features in the data increases, the model's scalability and efficiency fall short of expectations. The main cause of this particular behavior is because each feature needs to analyze every data instance in order to predict every potential split point, which takes a lot of time and effort.

3. **Random Forest**
   Leo Breiman and Adele Cutler are the trademark holders of the widely used machine learning technique known as "random forest," which aggregates the output of several decision trees to produce a single conclusion. Its versatility and ease of use, combined with its ability to handle both regression and classification problems, have driven its popularity.

4. **XGBoost**
   XGBoost, developed by Tianqi Chen and now maintained by a community, is another highly acclaimed machine learning algorithm. It stands for "Extreme Gradient Boosting" and is renowned for its exceptional performance in various machine learning competitions and real-world applications. Similar to random forest, XGBoost is capable of handling both regression and classification tasks. It works by sequentially building a series of decision trees, each correcting the errors of its predecessor, resulting in a powerful ensemble model. Its efficiency, scalability, and regularization techniques make it a popular choice among data scientists and practitioners for building robust predictive models.

3.4.4 Model Evaluation

Model evaluation is a crucial step in the world of machine learning. The final objective is to make sure the model accurately predicts outcomes using new data, in addition to matching the training set.

In this particular section, we will explore three evaluation strategies used with our model:

- **Mean Squared Error (MSE)**
- **R2 Score, and**
- **Root Mean Squared Error (RMSE)**

1. **Mean Squared Error (MSE)**
   Mean Squared Error evaluation technique is used to estimate the distance between a regression line and a set of data points. It is basically a risk function that agrees with the expected value of the squared error loss. In order to calculate the mean squared error, we find the mean, or average, of the squared errors derived from data in relation to a function.

2. **R2 Score**

   The performance of a linear regression model is assessed using the coefficient of determination, also known as the R2 score. What can be predicted from the input independent variable(s) is the degree of variation in the dependent attribute of the output. Depending on the ratio of the total deviation of the results the model describes, it is used to assess how well-observed results are replicated by the model. A high R2 Score (towards 1) signifies that a significant proportion of the outcome variability is accounted for by the model, indicating strong predictive capabilities of the model.

3. **Root Mean Squared Error (RMSE)**

   Root Mean Squared Error is an extension of Mean Squared Error, providing a more interpretable measure by taking the square root of the average squared differences. RMSE is particularly useful in our project for conveying the average magnitude of errors in our predictions. Similar to MSE, lower RMSE values indicate improved model accuracy. By incorporating the square root, RMSE presents errors in the same units as the target variable, making it easier to comprehend in the context of customer satisfaction scores.

3.4.5 Model Deployment and Prediction

Technologies Used

In the development of our "Production-Ready Customer Satisfaction Prediction" project during its first iteration, we strategically employed a set of technologies to create a robust and efficient machine learning pipeline.

Python

Beyond its readability and grammar, Python is very famous due to its pleasant and lively community. Python is now a strong device for developers because of the abundance of third-birthday party libraries and frameworks that have been made viable with the aid of the community's dedication to open-supply improvement. Python's substantial adoption across a number of sectors and fields may be attributed to its adaptability, cross-platform interoperability, and pleasant community, which make it a terrific alternative for amateur and pro programmers alike.

MLFlow

MLflow is an open source platform for managing the end-to-end machine learning lifecycle. It has the following primary components:

- **Tracking:** Allows you to track experiments to record and compare parameters and results.
- **Models:** Allow you to manage and deploy models from a variety of ML libraries to a variety of model serving and inference platforms.
- **Projects:** Allow you to package ML code in a reusable, reproducible form to share with other data scientists or transfer to production.
- **Model Registry:** Allows you to centralize a model store for managing models' full lifecycle stage transitions: from staging to production, with capabilities for versioning and annotating. Databricks provides a managed version of the Model Registry in Unity Catalog.
- **Model Serving:** Allows you to host MLflow Models as REST endpoints.

ZenML

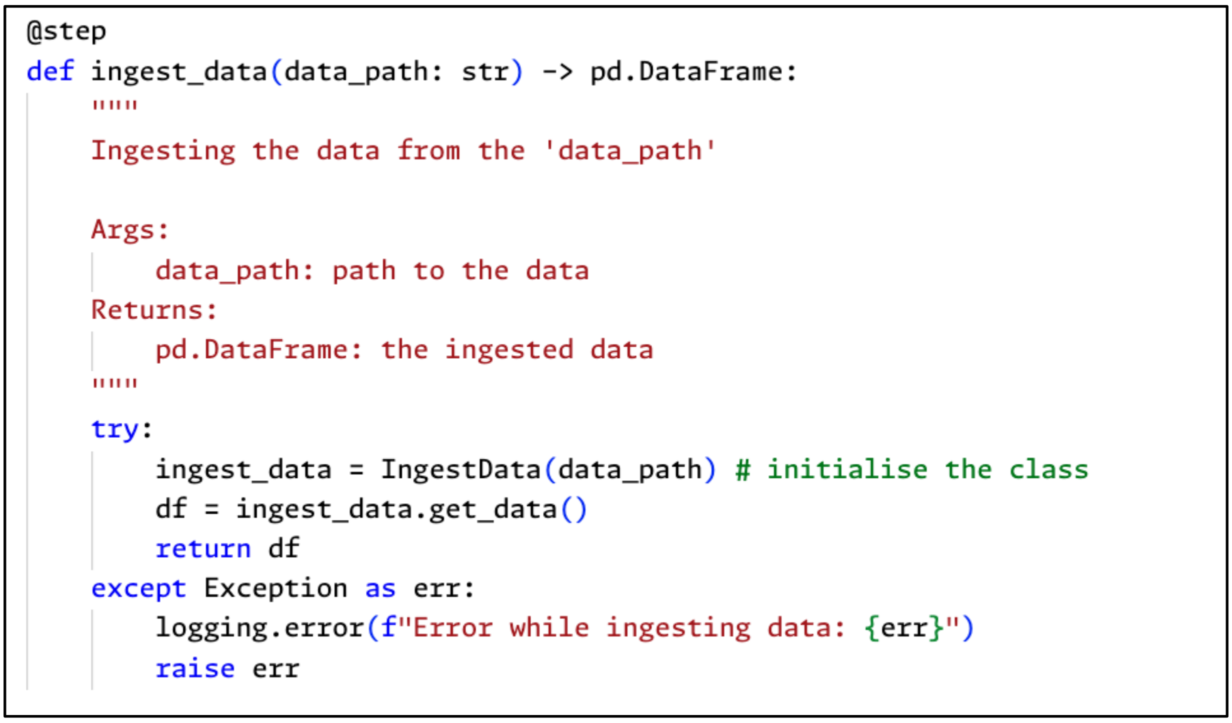
ZenML is an extensible, open-source MLOps framework for creating portable, production-ready machine learning pipelines. By decoupling infrastructure from code, ZenML enables developers across your organization to collaborate more effectively as they develop to production. Some of its notable features are as follows:

- **Local Development enabled**
- **Seamless integration with Python libraries**
- **Automatic Metadata Data Tracking**
- **Self-hosted Cloud Services**

Code Snippets

Let's dive into some crucial code snippets from our "Production-Ready Customer Satisfaction Prediction" project in this section.

1.1 Data Ingestion Class

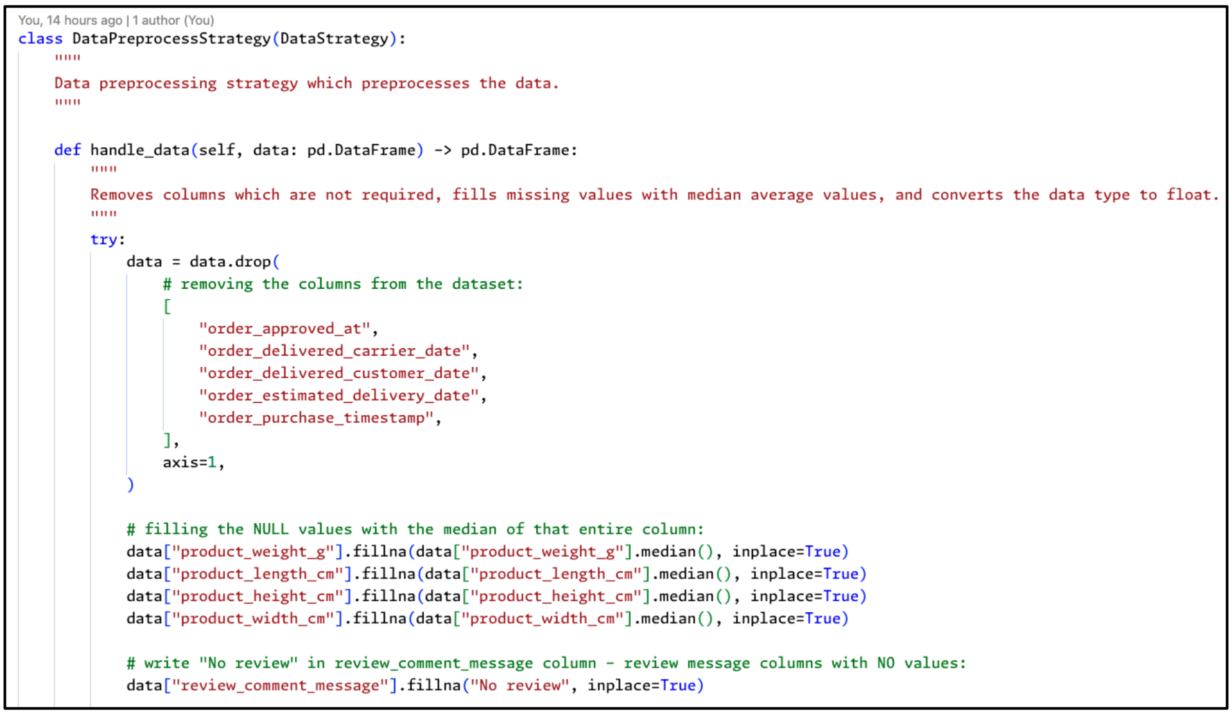


```
@step
def ingest_data(data_path: str) -> pd.DataFrame:
    """
    Ingesting the data from the 'data_path'

    Args:
        data_path: path to the data
    Returns:
        pd.DataFrame: the ingested data
    """
    try:
        ingest_data = IngestData(data_path) # initialise the class
        df = ingest_data.get_data()
        return df
    except Exception as err:
        logging.error(f"Error while ingesting data: {err}")
        raise err
```

Figure 5 - Data Ingestion

The figure above (Fig. 4) shows that we defined a class, **IngestData**, which is designed to ingest data from a specified source, returning a pandas DataFrame.

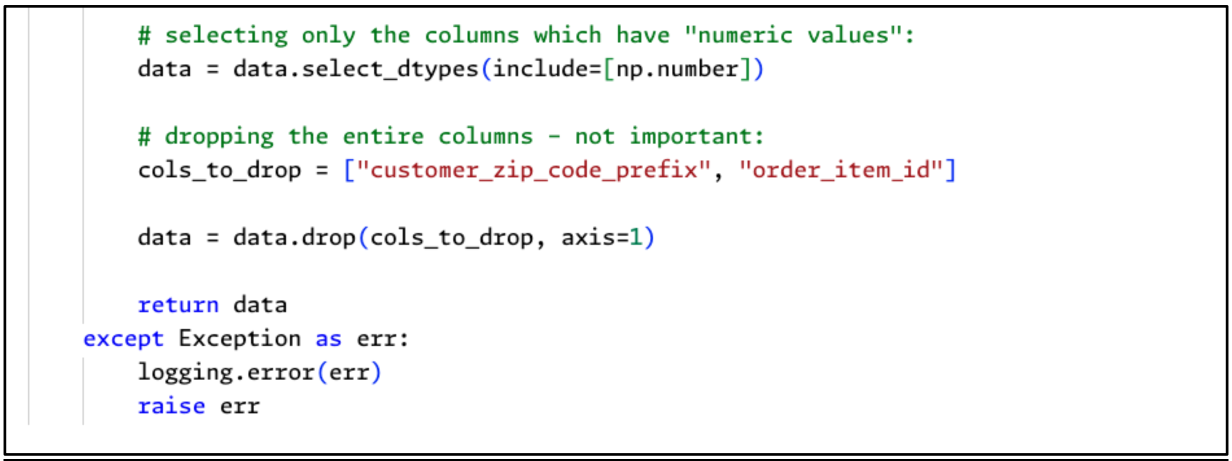
1.2 Data Cleaning Class



```
You, 14 hours ago | 1 author (You)
class DataPreprocessStrategy(DataStrategy):
    """
    Data preprocessing strategy which preprocesses the data.
    """

    def handle_data(self, data: pd.DataFrame) -> pd.DataFrame:
        """
        Removes columns which are not required, fills missing values with median average values, and converts the data type to float.
        """
        try:
            data = data.drop(
                # removing the columns from the dataset:
                [
                    "order_approved_at",
                    "order_delivered_carrier_date",
                    "order_delivered_customer_date",
                    "order_estimated_delivery_date",
                    "order_purchase_timestamp",
                ],
                axis=1,
            )

            # filling the NULL values with the median of that entire column:
            data["product_weight_g"].fillna(data["product_weight_g"].median(), inplace=True)
            data["product_length_cm"].fillna(data["product_length_cm"].median(), inplace=True)
            data["product_height_cm"].fillna(data["product_height_cm"].median(), inplace=True)
            data["product_width_cm"].fillna(data["product_width_cm"].median(), inplace=True)

            # write "No review" in review_comment_message column - review message columns with NO values:
            data["review_comment_message"].fillna("No review", inplace=True)
```

Figure 6 - Data Cleaning (a)

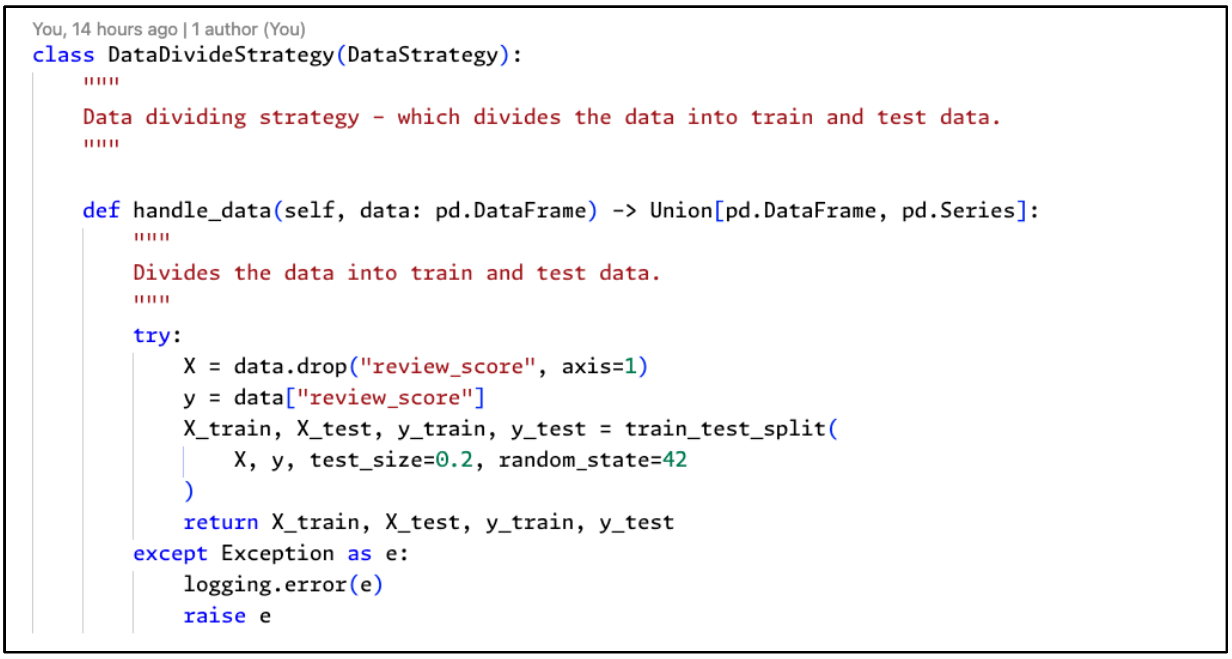
The figure above (Fig. 5) shows that we defined the Data Preprocessing Strategy which is performing: Column Removal and Handling Review Comment Messages operations.



```
        # selecting only the columns which have "numeric values":
        data = data.select_dtypes(include=[np.number])

        # dropping the entire columns - not important:
        cols_to_drop = ["customer_zip_code_prefix", "order_item_id"]

        data = data.drop(cols_to_drop, axis=1)

        return data
    except Exception as err:
        logging.error(err)
        raise err
```

Figure 7 - Data Cleaning (b)

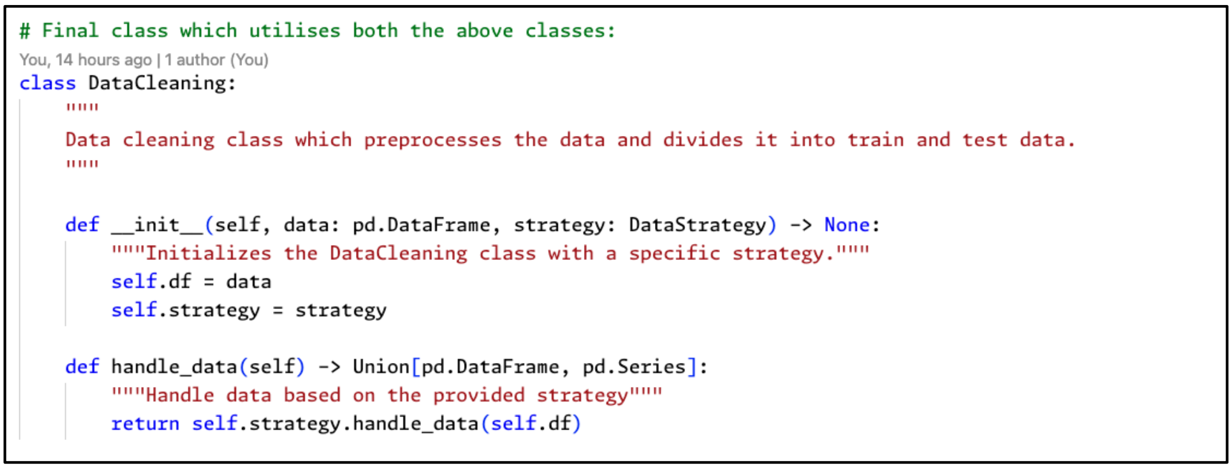
The figure above (Fig. 6) shows that in our Data Preprocessing Strategy, we are performing: Data Type Conversion and Additional Column Removal operations.



```
You, 14 hours ago | 1 author (You)
class DataDivideStrategy(DataStrategy):
    """
    Data dividing strategy - which divides the data into train and test data.
    """

    def handle_data(self, data: pd.DataFrame) -> Union[pd.DataFrame, pd.Series]:
        """
        Divides the data into train and test data.
        """
        try:
            X = data.drop("review_score", axis=1)
            y = data["review_score"]
            X_train, X_test, y_train, y_test = train_test_split(
                X, y, test_size=0.2, random_state=42
            )
            return X_train, X_test, y_train, y_test
        except Exception as e:
            logging.error(e)
            raise e
```

Figure 8 - Data Cleaning (c)

The figure above (Fig. 7) shows the DataDivideStrategy class which implements the data division process, separating the dataset into training and testing sets.

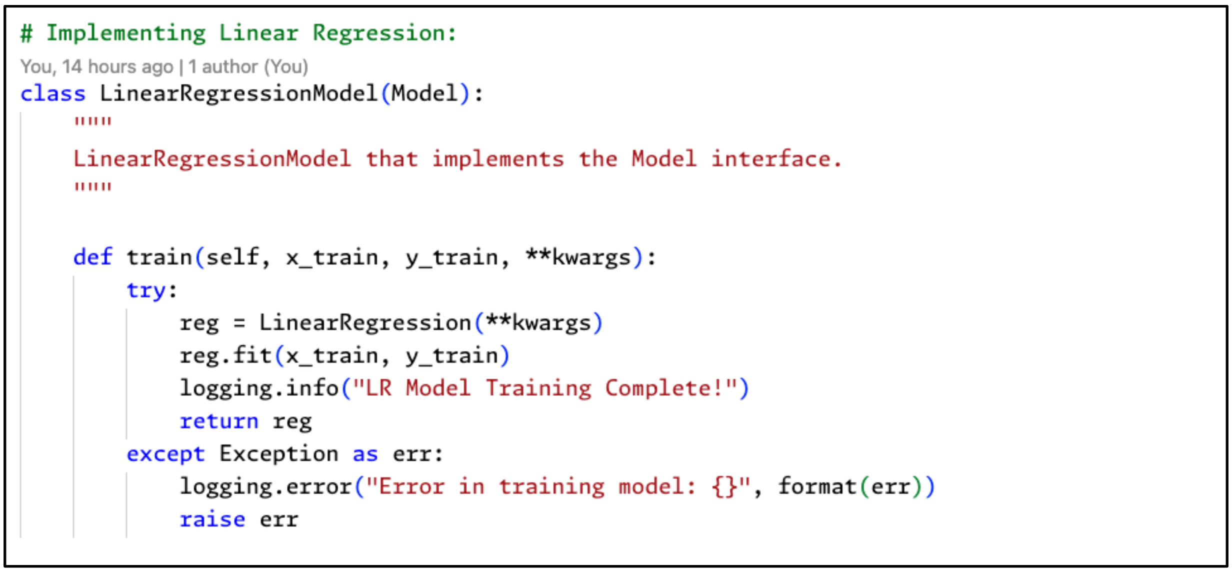


```
# Final class which utilises both the above classes:
You, 14 hours ago | 1 author (You)
class DataCleaning:
    """
    Data cleaning class which preprocesses the data and divides it into train and test data.
    """

    def __init__(self, data: pd.DataFrame, strategy: DataStrategy) -> None:
        """Initializes the DataCleaning class with a specific strategy."""
        self.df = data
        self.strategy = strategy

    def handle_data(self) -> Union[pd.DataFrame, pd.Series]:
        """Handle data based on the provided strategy"""
        return self.strategy.handle_data(self.df)
```

Figure 9 - Data Cleaning (d)

The figure above (Fig. 7) finally shows the executes both the above classes to complete the overall process.

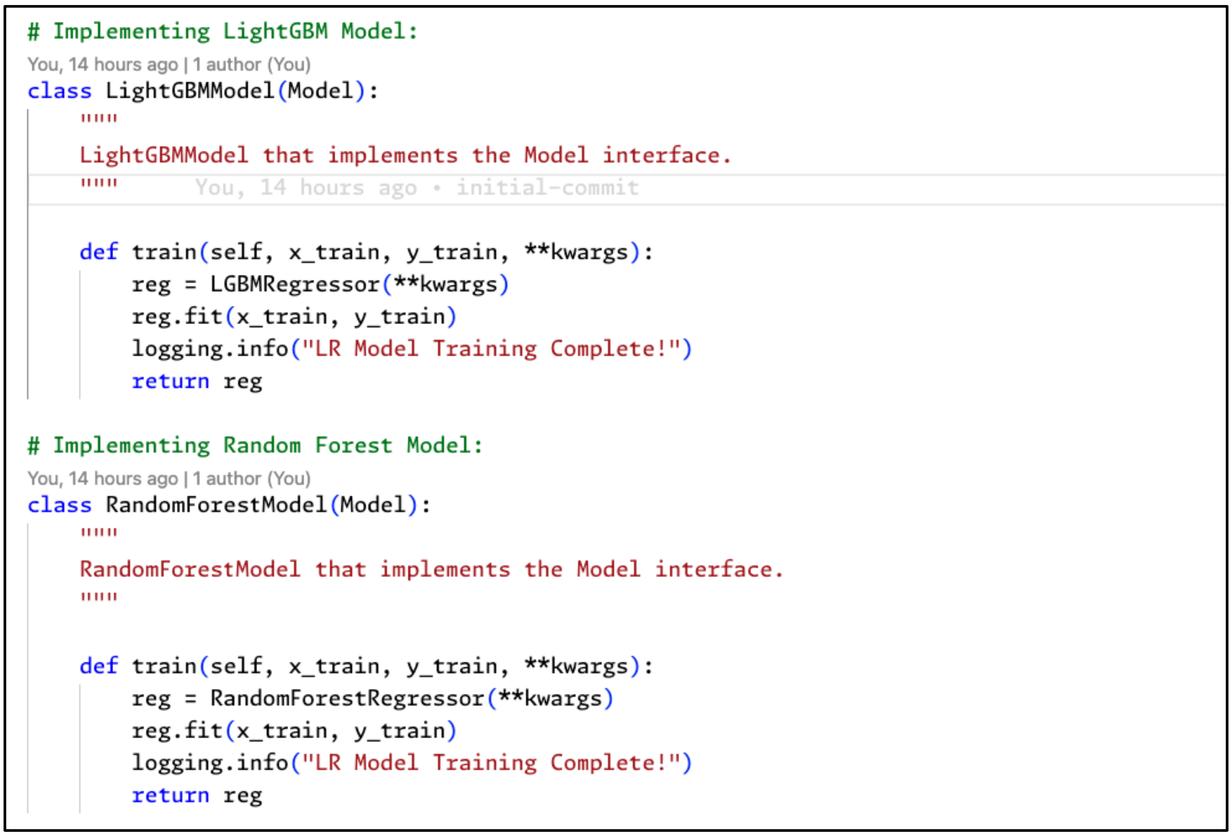
1.3 Model Development



```
# Implementing Linear Regression:
You, 14 hours ago | 1 author (You)
class LinearRegressionModel(Model):
    """
    LinearRegressionModel that implements the Model interface.
    """

    def train(self, x_train, y_train, **kwargs):
        try:
            reg = LinearRegression(**kwargs)
            reg.fit(x_train, y_train)
            logging.info("LR Model Training Complete!")
            return reg
        except Exception as err:
            logging.error("Error in training model: {}", format(err))
            raise err
```

Figure 10 - Linear Regression

The figure above (Fig. 9) shows the entire implementation of the linear regression algorithm using Python, taking in the training parameters as input and returning a fit model after being trained.

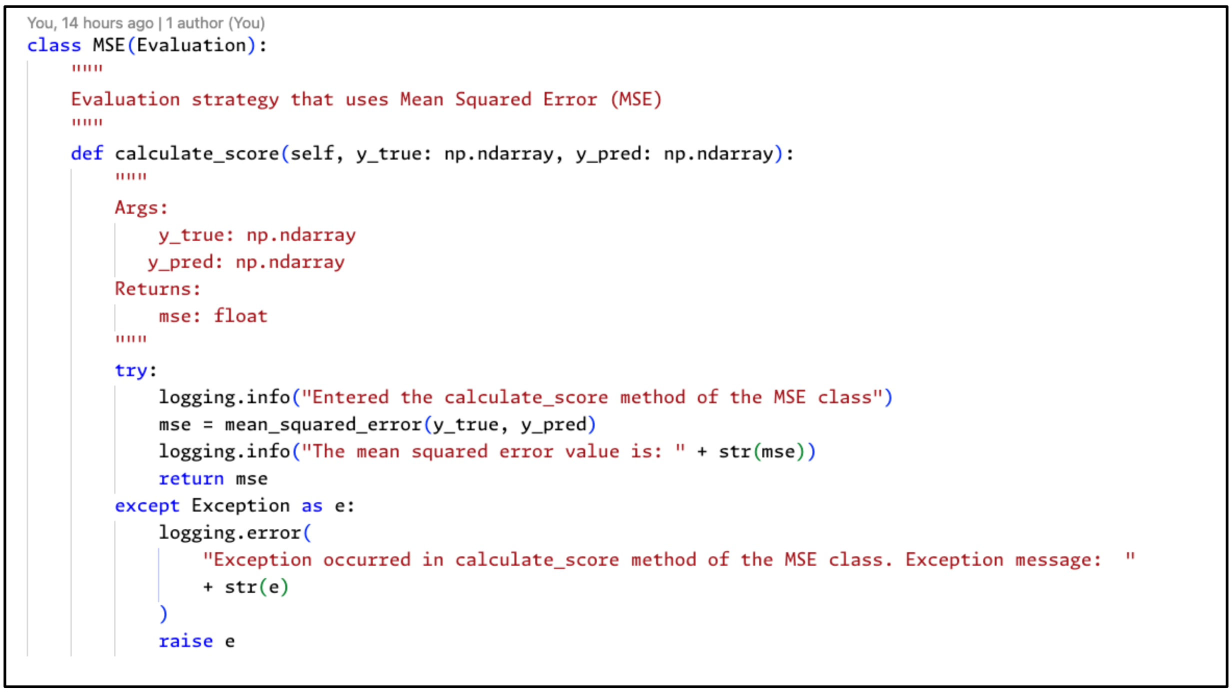


```
# Implementing LightGBM Model:
You, 14 hours ago | 1 author (You)
class LightGBMModel(Model):
    """
    LightGBMModel that implements the Model interface.
    """      You, 14 hours ago • initial-commit

    def train(self, x_train, y_train, **kwargs):
        reg = LGBMRegressor(**kwargs)
        reg.fit(x_train, y_train)
        logging.info("LR Model Training Complete!")
        return reg

# Implementing Random Forest Model:
You, 14 hours ago | 1 author (You)
class RandomForestModel(Model):
    """
    RandomForestModel that implements the Model interface.
    """

    def train(self, x_train, y_train, **kwargs):
        reg = RandomForestRegressor(**kwargs)
        reg.fit(x_train, y_train)
        logging.info("LR Model Training Complete!")
        return reg
```

Figure 11 - LightGBM & Random Forest

The figure above (Fig. 10) shows the entire implementation of the LightGBM and Random Forest algorithm using Python, taking in the training parameters as input and returning a fit model after being trained.

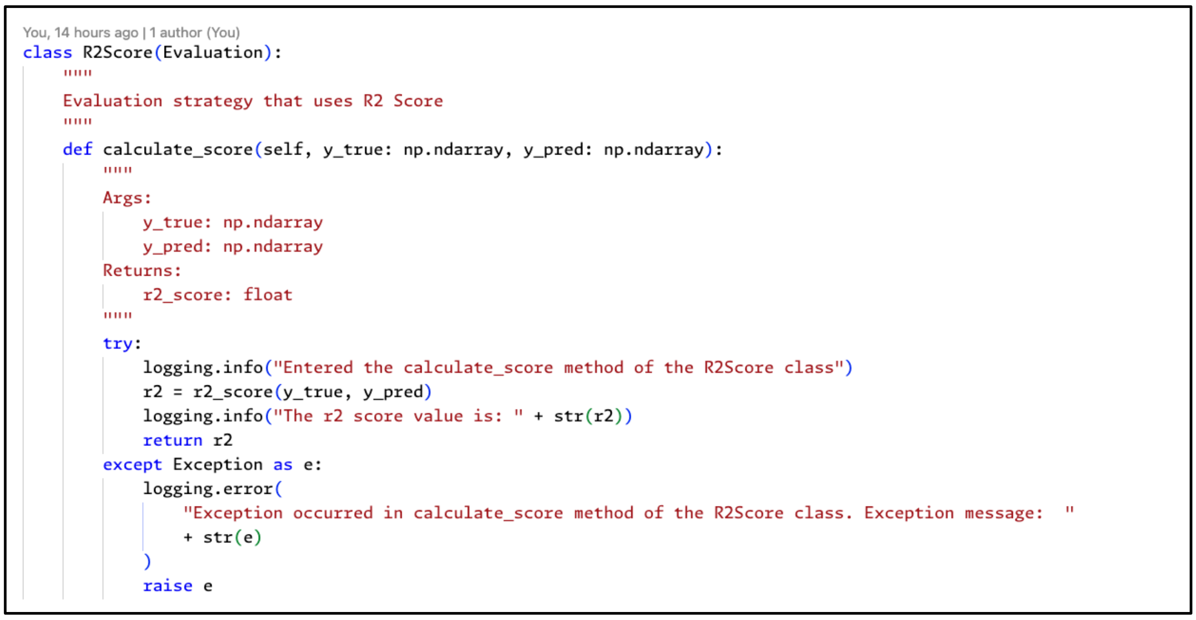
1.4 Model Evaluation



```
You, 14 hours ago | 1 author (You)
class MSE(Evaluation):
    """
    Evaluation strategy that uses Mean Squared Error (MSE)
    """
    def calculate_score(self, y_true: np.ndarray, y_pred: np.ndarray):
        """
        Args:
            y_true: np.ndarray
           y_pred: np.ndarray
        Returns:
            mse: float
        """
        try:
            logging.info("Entered the calculate_score method of the MSE class")
            mse = mean_squared_error(y_true, y_pred)
            logging.info("The mean squared error value is: " + str(mse))
            return mse
        except Exception as e:
            logging.error(
                "Exception occurred in calculate_score method of the MSE class. Exception message:  "
                + str(e)
            )
            raise e
```

Figure 12 - Mean Squared Error

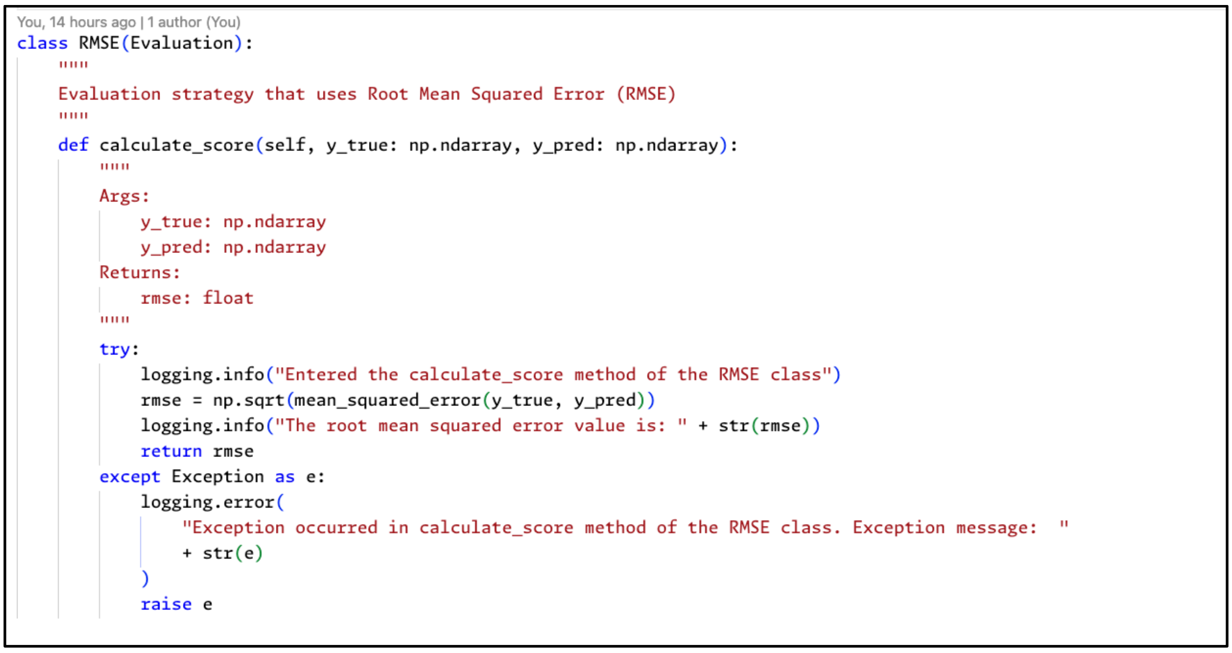
The figure above (Fig. 11) shows the entire implementation of the mean squared error evaluation methods using Python, taking in the predicted parameters as input and returning an evaluated score.



```
You, 14 hours ago | 1 author (You)
class R2Score(Evaluation):
    """
    Evaluation strategy that uses R2 Score
    """
    def calculate_score(self, y_true: np.ndarray, y_pred: np.ndarray):
        """
        Args:
            y_true: np.ndarray
            y_pred: np.ndarray
        Returns:
            r2_score: float
        """
        try:
            logging.info("Entered the calculate_score method of the R2Score class")
            r2 = r2_score(y_true, y_pred)
            logging.info("The r2 score value is: " + str(r2))
            return r2
        except Exception as e:
            logging.error(
                "Exception occurred in calculate_score method of the R2Score class. Exception message:  "
                + str(e)
            )
            raise e
```

Figure 13 - R2Score

The figure above (Fig. 12) shows the entire implementation of the R2 score evaluation methods using Python, taking in the predicted parameters as input and returning an evaluated score.

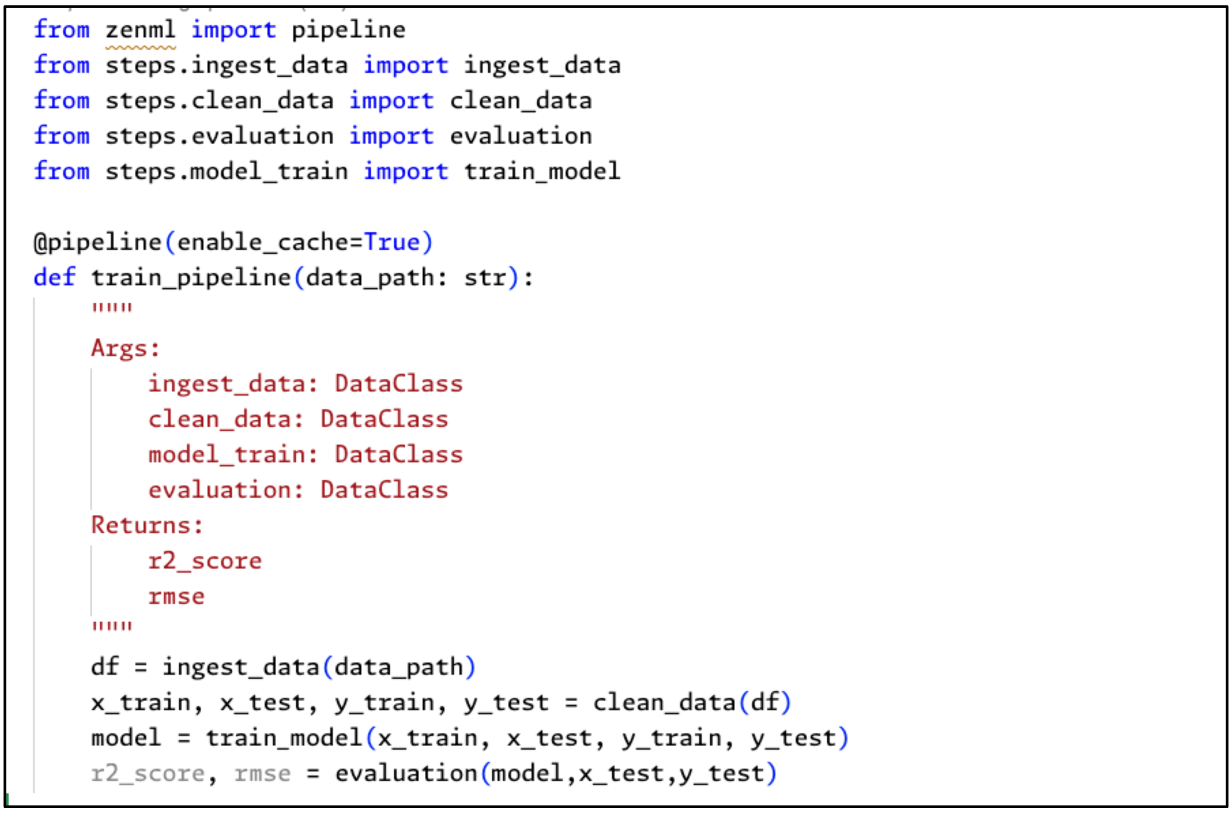


```
You, 14 hours ago | 1 author (You)
class RMSE(Evaluation):
    """
    Evaluation strategy that uses Root Mean Squared Error (RMSE)
    """
    def calculate_score(self, y_true: np.ndarray, y_pred: np.ndarray):
        """
        Args:
            y_true: np.ndarray
            y_pred: np.ndarray
        Returns:
            rmse: float
        """
        try:
            logging.info("Entered the calculate_score method of the RMSE class")
            rmse = np.sqrt(mean_squared_error(y_true, y_pred))
            logging.info("The root mean squared error value is: " + str(rmse))
            return rmse
        except Exception as e:
            logging.error(
                "Exception occurred in calculate_score method of the RMSE class. Exception message:  "
                + str(e)
            )
            raise e
```

Figure 14 - Root Mean Squared Error

The figure above (Fig. 13) shows the entire implementation of the root mean squared error evaluation methods using Python, taking in the predicted parameters as input and returning an evaluated score.

1.5 Training Pipeline



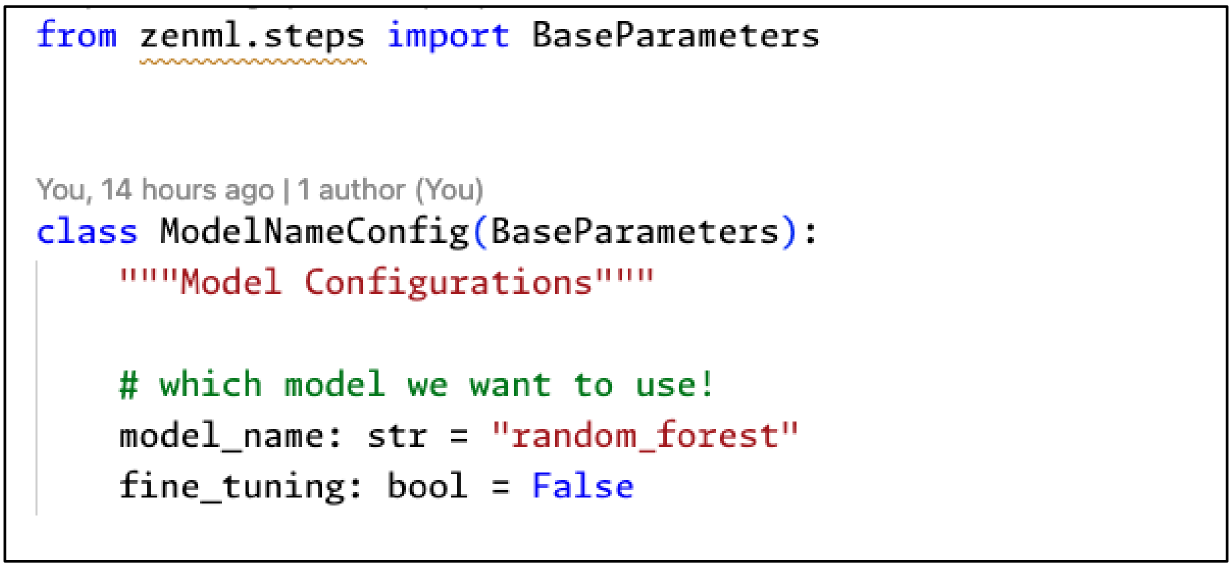
```
from zenml import pipeline
from steps.ingest_data import ingest_data
from steps.clean_data import clean_data
from steps.evaluation import evaluation
from steps.model_train import train_model

@pipeline(enable_cache=True)
def train_pipeline(data_path: str):
    """
    Args:
        ingest_data: DataClass
        clean_data: DataClass
        model_train: DataClass
        evaluation: DataClass
    Returns:
        r2_score
        rmse
    """
    df = ingest_data(data_path)
    x_train, x_test, y_train, y_test = clean_data(df)
    model = train_model(x_train, x_test, y_train, y_test)
    r2_score, rmse = evaluation(model,x_test,y_test)
```

Figure 15 - Training Pipeline

The figure above (Fig. 14) shows the different stages of the MLOps pipeline defined by us in Python. The stages include:

- Ingesting the data from the dataset
- Cleaning the data ingested from the dataset
- Training the model using the selected mode (either Linear Regression, LightGBM or Random forest)
- Evaluating the model using either MSE, R2 Score or RMSE

1.6 Custom Model Configuration

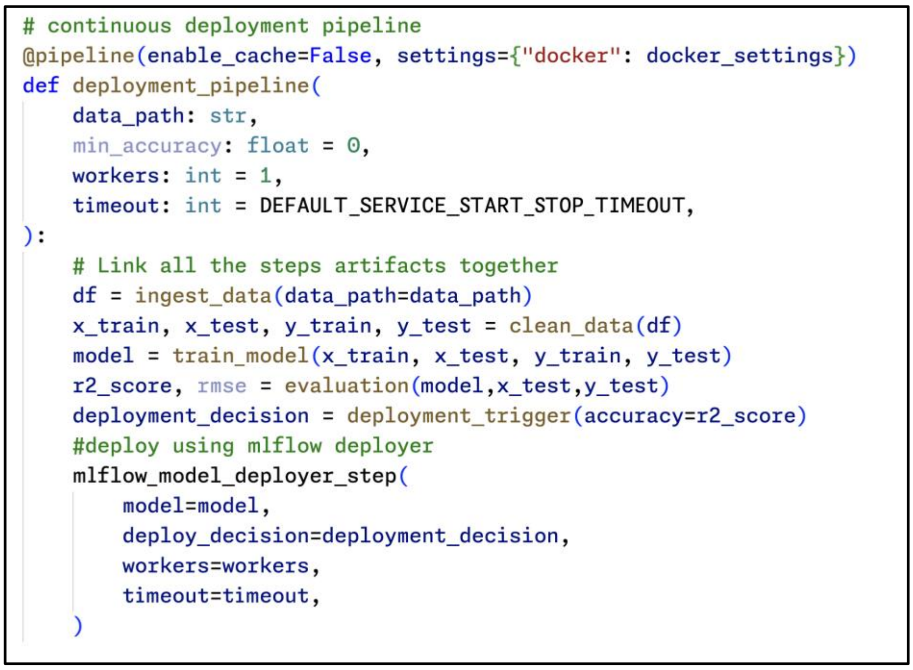


```
from zenml.steps import BaseParameters


You, 14 hours ago | 1 author (You)
class ModelNameConfig(BaseParameters):
    """Model Configurations"""

    # which model we want to use!
    model_name: str = "random_forest"
    fine_tuning: bool = False
```

Figure 16 - Model Configuration

The figure above (Fig. 15) shows an implementation of custom model configuration. That means, we’ll be able to select the model to be used in our training process. The three different models being used are: Linear Regression, LightGBM and Random Forest.

1.7 Deployment Pipeline



```python
# continuous deployment pipeline
@pipeline(enable_cache=False, settings={"docker": docker_settings})
def deployment_pipeline(
    data_path: str,
    min_accuracy: float = 0,
    workers: int = 1,
    timeout: int = DEFAULT_SERVICE_START_STOP_TIMEOUT,
):
    # Link all the steps artifacts together
    df = ingest_data(data_path=data_path)
    x_train, x_test, y_train, y_test = clean_data(df)
    model = train_model(x_train, x_test, y_train, y_test)
    r2_score, rmse = evaluation(model,x_test,y_test)
    deployment_decision = deployment_trigger(accuracy=r2_score)
    #deploy using mlflow deployer
    mlflow_model_deployer_step(
        model=model,
        deploy_decision=deployment_decision,
        workers=workers,
        timeout=timeout,
    )
```

Figure 17 - Deployment Pipeline Stages

The figure above (Fig. 16) shows the different stages of the MLOps deployment pipeline defined by us in Python. The stages include:

- **Evaluate Pre-trained Model:**
    - After training, the model's performance is evaluated using evaluation metrics such as R2 score and RMSE (Root Mean Square Error).
- **Deployment Trigger:**
    - This stage determines whether the model meets the minimum accuracy threshold required for deployment.
- **Model Deployment:**
    - If the model meets the accuracy threshold, it's deployed using the MLflow deployer, enabling it to be utilized for making predictions in real-world scenarios.

1.8 Deployment of Server (Backend-Operation)

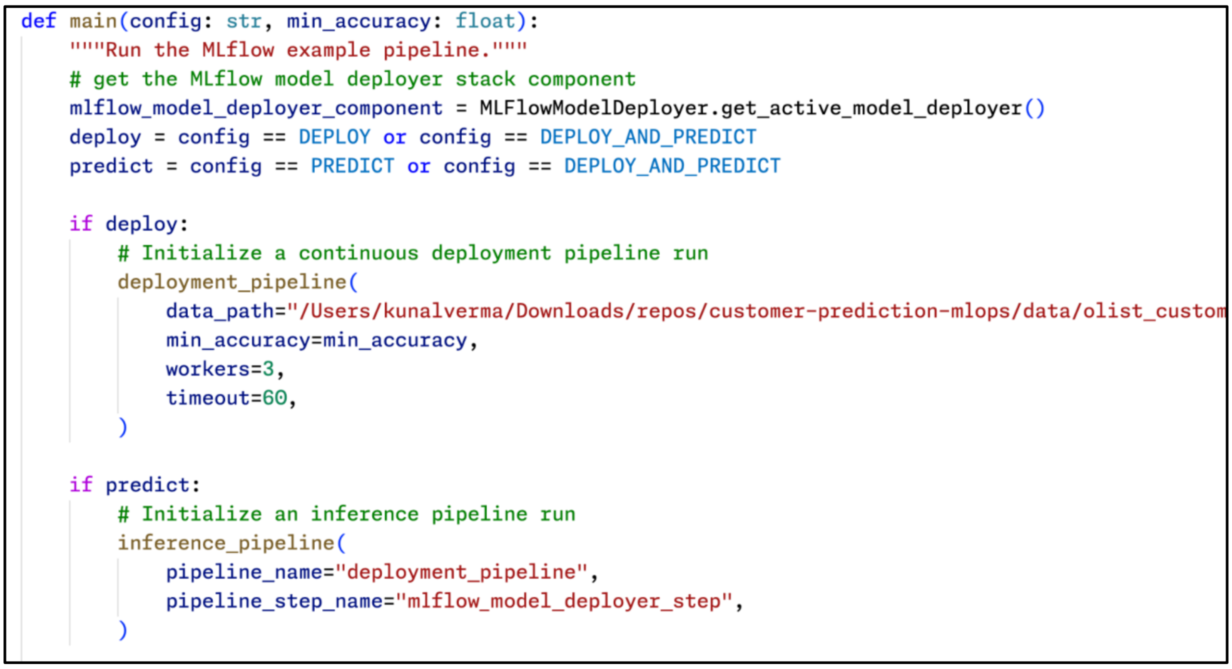
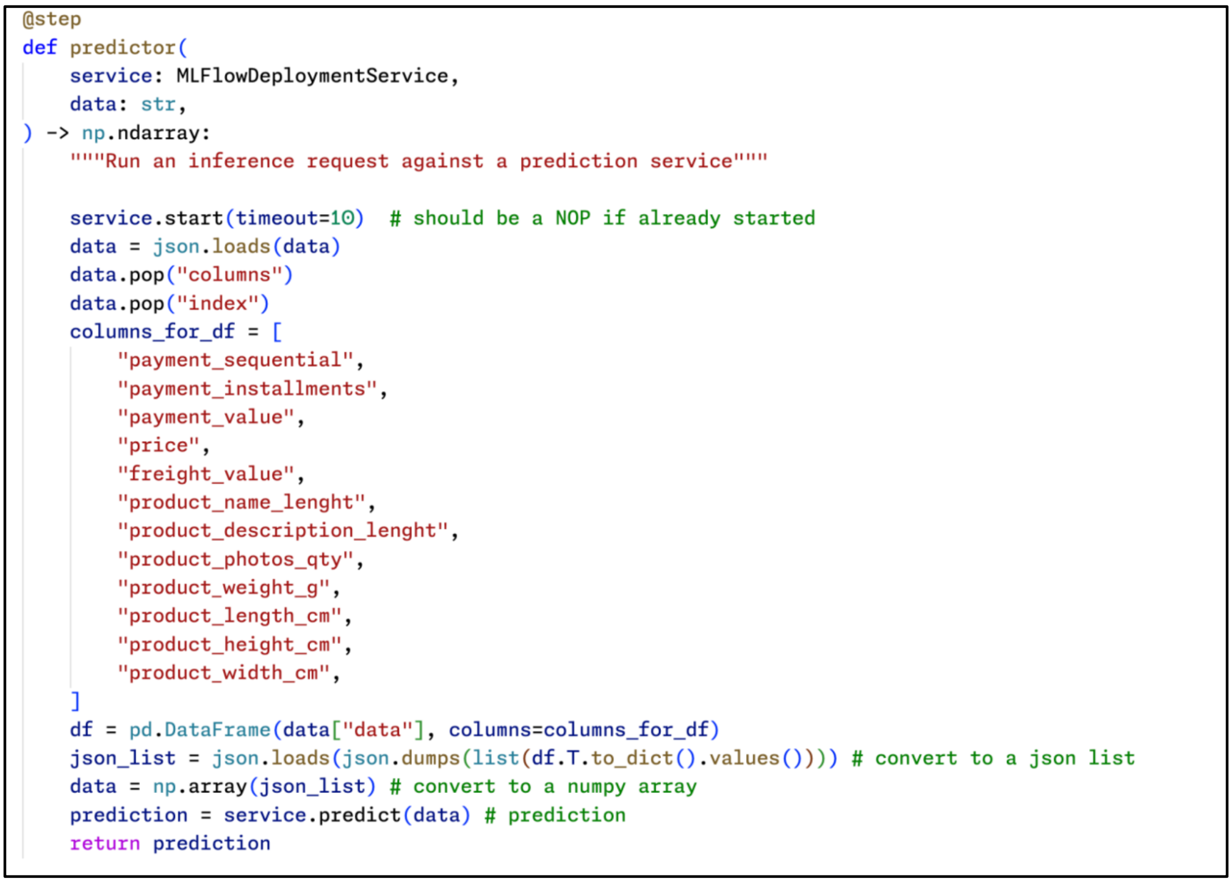

```
def main(config: str, min_accuracy: float):
    """Run the MLflow example pipeline."""
    # get the MLflow model deployer stack component
    mlflow_model_deployer_component = MLFlowModelDeployer.get_active_model_deployer()
    deploy = config == DEPLOY or config == DEPLOY_AND_PREDICT
    predict = config == PREDICT or config == DEPLOY_AND_PREDICT

    if deploy:
        # Initialize a continuous deployment pipeline run
        deployment_pipeline(
            data_path="/Users/kunalverma/Downloads/repos/customer-prediction-mlops/data/olist_custom
            min_accuracy=min_accuracy,
            workers=3,
            timeout=60,
        )

    if predict:
        # Initialize an inference pipeline run
        inference_pipeline(
            pipeline_name="deployment_pipeline",
            pipeline_step_name="mlflow_model_deployer_step",
        )
```

Figure 18 - Deployment of Server

The figure above (Fig. 17) showcases our MLOps deployment pipeline's orchestration using Python. It provides a command-line interface (CLI) for running various pipeline configurations, including deployment, prediction, or both. Users can choose specific options such as deploying a model, making predictions, or performing both tasks simultaneously.

1.9 Prediction Pipeline Service



```python
@step
def predictor(
    service: MLFlowDeploymentService,
    data: str,
) -> np.ndarray:
    """Run an inference request against a prediction service"""

    service.start(timeout=10)  # should be a NOP if already started
    data = json.loads(data)
    data.pop("columns")
    data.pop("index")
    columns_for_df = [
        "payment_sequential",
        "payment_installments",
        "payment_value",
        "price",
        "freight_value",
        "product_name_lenght",
        "product_description_lenght",
        "product_photos_qty",
        "product_weight_g",
        "product_length_cm",
        "product_height_cm",
        "product_width_cm",
    ]
    df = pd.DataFrame(data["data"], columns=columns_for_df)
    json_list = json.loads(json.dumps(list(df.T.to_dict().values()))) # convert to a json list
    data = np.array(json_list) # convert to a numpy array
    prediction = service.predict(data) # prediction
    return prediction
```

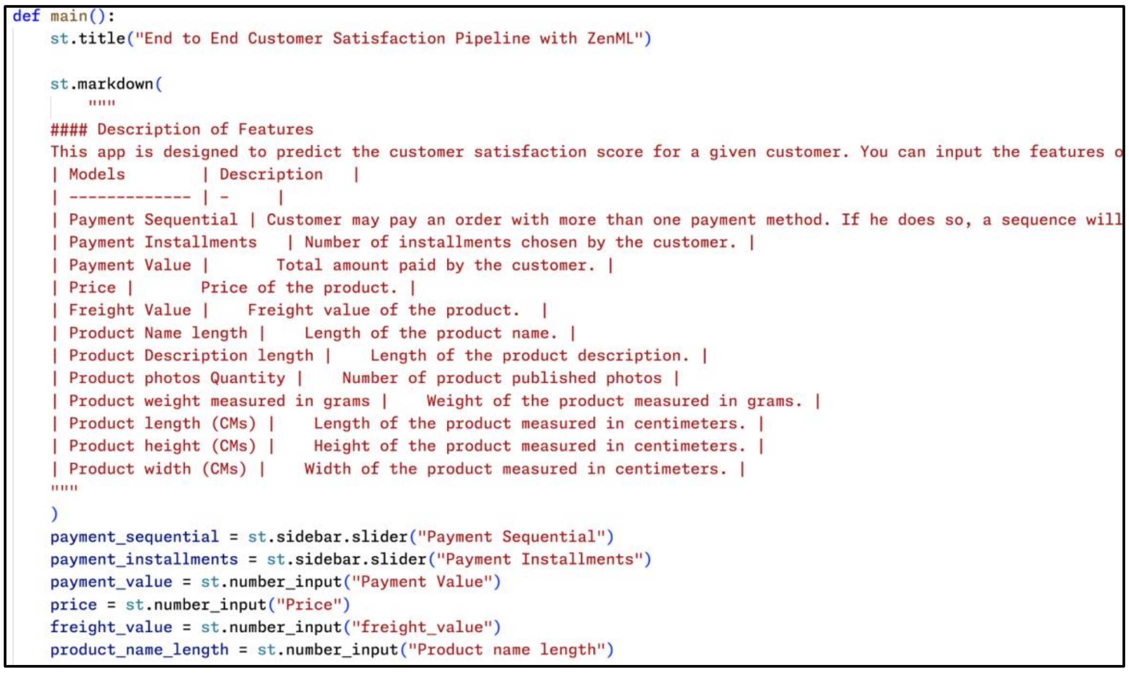
Figure 19 - Prediction Pipeline Service

The figure above (Fig. 16) illustrates the different stages of our MLOps deployment pipeline, as defined in Python. These stages include:

- **Start Service:**
    - This initial step ensures that the MLFlow deployment service is up and running, ready to handle inference requests.
- **Data Preparation:**
    - Here, incoming data for prediction is processed and prepared for inference. This involves converting the data from a JSON format into a pandas DataFrame, selecting specific columns relevant to the prediction task.
- **Data Transformation:**
    - Following data preparation, the DataFrame is transformed into a JSON list and then into a numpy array, making it suitable for input into the prediction model.

- **Prediction:**
  - In this crucial stage, the prepared data is fed into the MLFlow deployment service for prediction. The service utilizes the deployed machine learning model to generate predictions based on the input data.
- **Return Prediction:**
  - Finally, the predicted values are returned as a numpy array, ready for further analysis or downstream processing in the ML pipeline. This completes the prediction process within our deployment pipeline.

1.10 Streamlit (Frontend) App Definition



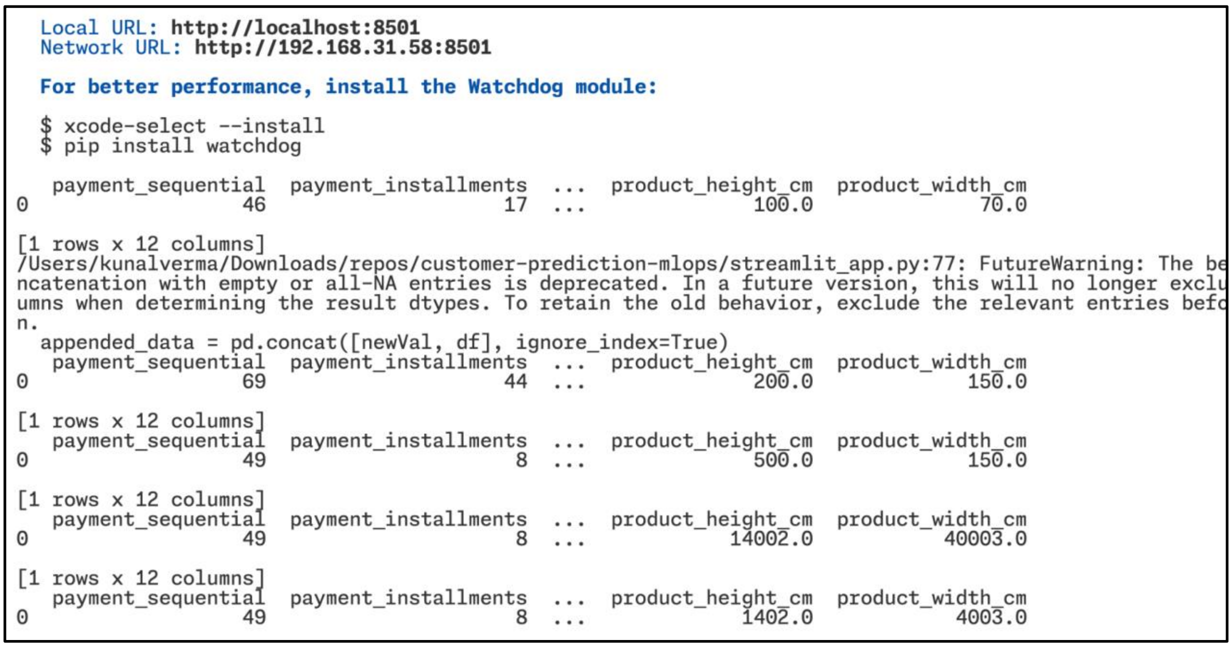
```python
def main():
    st.title("End to End Customer Satisfaction Pipeline with ZenML")

    st.markdown(
        """
    #### Description of Features
    This app is designed to predict the customer satisfaction score for a given customer. You can input the features o
    | Models        | Description   |
    | ------------- | -     |
    | Payment Sequential | Customer may pay an order with more than one payment method. If he does so, a sequence will
    | Payment Installments   | Number of installments chosen by the customer. |
    | Payment Value |       Total amount paid by the customer. |
    | Price |       Price of the product. |
    | Freight Value |    Freight value of the product.  |
    | Product Name length |    Length of the product name. |
    | Product Description length |    Length of the product description. |
    | Product photos Quantity |    Number of product published photos |
    | Product weight measured in grams |    Weight of the product measured in grams. |
    | Product length (CMs) |    Length of the product measured in centimeters. |
    | Product height (CMs) |    Height of the product measured in centimeters. |
    | Product width (CMs) |    Width of the product measured in centimeters. |
    """
    )
    payment_sequential = st.sidebar.slider("Payment Sequential")
    payment_installments = st.sidebar.slider("Payment Installments")
    payment_value = st.number_input("Payment Value")
    price = st.number_input("Price")
    freight_value = st.number_input("freight_value")
    product_name_length = st.number_input("Product name length")
```

Figure 20 - Streamlit App Definition

The figure above (Fig. 19) shows a simple streamlit application definition in Python where we define the following:

- Parameters to showcase on the frontend
- Connecting the Prediction Module

1.11 Storing Predicted Product Values



```
  Local URL: http://localhost:8501
  Network URL: http://192.168.31.58:8501

  For better performance, install the Watchdog module:

  $ xcode-select --install
  $ pip install watchdog

   payment_sequential  payment_installments  ...  product_height_cm  product_width_cm
0                  46                    17  ...              100.0              70.0

[1 rows x 12 columns]
/Users/kunalverma/Downloads/repos/customer-prediction-mlops/streamlit_app.py:77: FutureWarning: The be
ncatenation with empty or all-NA entries is deprecated. In a future version, this will no longer exclu
umns when determining the result dtypes. To retain the old behavior, exclude the relevant entries befo
n.
  appended_data = pd.concat([newVal, df], ignore_index=True)
   payment_sequential  payment_installments  ...  product_height_cm  product_width_cm
0                  69                    44  ...              200.0             150.0

[1 rows x 12 columns]
   payment_sequential  payment_installments  ...  product_height_cm  product_width_cm
0                  49                     8  ...              500.0             150.0

[1 rows x 12 columns]
   payment_sequential  payment_installments  ...  product_height_cm  product_width_cm
0                  49                     8  ...            14002.0           40003.0

[1 rows x 12 columns]
   payment_sequential  payment_installments  ...  product_height_cm  product_width_cm
0                  49                     8  ...             1402.0            4003.0
```

Figure 21 - Storing Predicted Product Values

The figure above (Fig. 20) shows the output of the backend storage automation, that is tasked to store the predicted product values locally in an Excel database.

3.5 Key Challenges

As we set out on the journey to construct a "Production-Ready Customer Satisfaction Prediction" system, we knew it would not be smooth sailing. We faced numerous hurdles that demanded quick problem-solving, flexibility, and strategic thinking.

In this section we delve into the challenges we faced during the project's development, revealing the intricacies involved in crafting a machine learning solution that is ready for implementation.

Disk Space Limitation

Our machine's limited disk space presented one major obstacle that we had to overcome. This constraint affected our choice of how big and how diverse the datasets we could reasonably work with. Balancing the need for a comprehensive dataset against the practicalities of disk space constraints required meticulous planning and resource allocation. Strategies for efficient data management and optimization became crucial in overcoming this challenge, ensuring that our pipeline remained both functional and scalable within the given limitations.

Limited Documentation for Open Source Software Used

The use of open source tools such as MLFlow posed challenges due to the fact of having complex and hard-to-understand documentation. While open source tools provide a variety of functionalities, understanding the inner workings of the tools proved to be a significant hurdle in the journey of building this project.

Limited Processing Power and Dataset Scope

The project encountered constraints in processing power, limiting the scale of datasets we could effectively incorporate. This limitation prompted the strategic decision to focus on a specific region, in this case, the Brazilian market. While this choice streamlined the project's scope and facilitated more manageable computations, it also imposed challenges in terms of generalizability. The trade-off between dataset size and processing capabilities underscored the importance of making strategic decisions aligned with the project's objectives and available resources.

Optimizing Results with Initial Iterations

As this project is focused on developing a CI/CD continuous pipeline to train our model continuously with new data, we were not able to ensure an optimal result in terms of model accuracy as part of the first iteration for this project. We wish to address this particular challenge by setting up an additional deployment trigger in our MLOps pipeline, that will be responsible to re-train the entire data on receiving new data, thus giving better results and accuracy.

Lack of Actual Data Values For Further Automation

As one of the final steps in our project, we wanted to automate the entire process of model retraining by using the data that is predicted along with the actual values of the reviews to keep the model updated with all the varying trends and patterns. This cannot be implemented as we lack the actual review scores right now. It is still possible to be implemented given the actual values are present, if we use the predicted values then our model will end up being skewed, lowering the accuracy even further which would be counterproductive.

These challenges collectively underscore the dynamic and multifaceted nature of developing a production-ready machine learning solution. Addressing these hurdles demanded a combination of technical expertise, strategic decision-making, and a commitment to continuous improvement, highlighting the inherent complexity of real-world machine learning projects.

Chapter 4: Testing

In the initial stages of our project, the testing strategy revolves around establishing a solid foundation for the machine learning pipeline. While a formal testing suite is not currently implemented, the rationale behind this decision lies in the project's nascent phase. As we build the baseline pipeline and explore various machine learning algorithms, our focus is on prototyping and experimentation. A more comprehensive testing suite will be integrated into the codebase as it evolves, providing a systematic approach to verify the correctness and reliability of our machine learning components.

Simultaneously, the ZenML platform plays a pivotal role in practically testing our model pipeline runs and evaluating results in each iteration. ZenML allows us to orchestrate and manage our machine learning workflows efficiently. By leveraging ZenML's capabilities, we can systematically track experiments, manage data versions, and ensure the reproducibility of our machine learning pipeline. While a comprehensive testing suite will be incorporated in subsequent project phases, the current emphasis on practical testing with ZenML aligns with the iterative and experimental nature of the project, laying the groundwork for a robust testing framework in the future. This strategic approach allows us to validate the performance of our models and pipeline in a controlled environment, setting the stage for the incorporation of more extensive testing practices as the project matures.

Following this, the only orthodox testing that we could do was testing the frontend to backend, connection and functionality. The entire process of testing involved changing multiple values in the frontend input and making a note of them and later checking the values in the model manually. Further, we saved all the predicted data values(for now in local storage) using an excel file.

Chapter 5: Results

The culmination of our efforts is unveiled in the results section, where the practical implications of implementing machine learning algorithms in predicting customer satisfaction come to light. In this exploration, we dissect the outcomes of applying Linear Regression, LightGBM Boost, and the Random Forest algorithm to our dataset. Each algorithm brings its unique strengths and nuances to the forefront, offering a nuanced understanding of their predictive capabilities.

As we delve into the specifics of each model's performance, we gain valuable insights that guide us in shaping a production-ready solution. The journey through these results not only underscores the complexity of customer satisfaction prediction but also sets the stage for informed decision-making in the subsequent phases of our project.

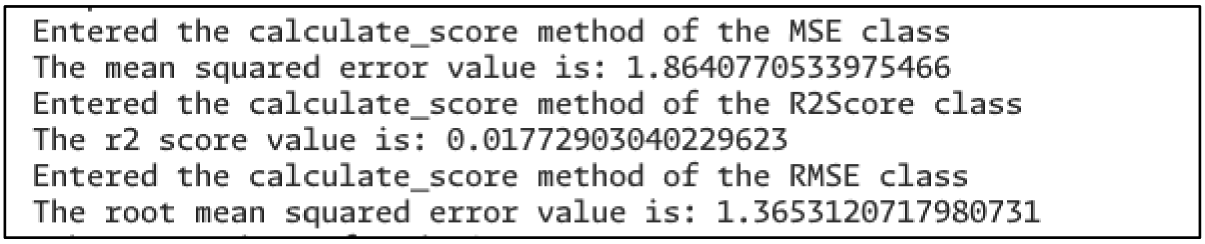
5.1 Results Using Linear Regression

The application of Linear Regression to predict customer satisfaction in this initial phase of the project provided insightful results.

- The mean squared error (MSE) of 1.864 suggests a moderate level of prediction accuracy.

- The R2 score of 0.017 indicates that the model explains only a small proportion of the variance in the data.

- Additionally, the root mean squared error (RMSE) of 1.365 signifies the average magnitude of errors in the predictions.



```
Entered the calculate_score method of the MSE class
The mean squared error value is: 1.8640770533975466
Entered the calculate_score method of the R2Score class
The r2 score value is: 0.01772903040229623
Entered the calculate_score method of the RMSE class
The root mean squared error value is: 1.3653120717980731
```

Figure 22 - Linear Regression Results

The figure above (Fig. 15) shows the evaluation results from the Linear Regression model. An in-depth explanation has been done in the above section (5.1).

These metrics collectively reveal that, while Linear Regression provides a foundational baseline, its performance may be limited in capturing the intricacies of customer satisfaction.

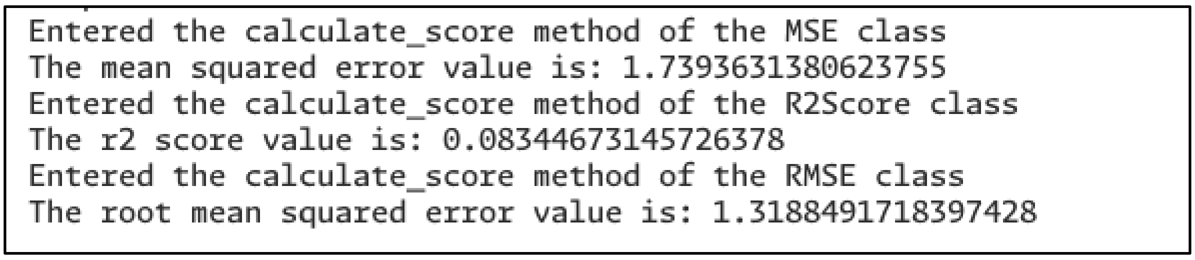
5.2 Results Using LightGBM Model

The introduction of the LightGBM Boost model brought improvements in predictive accuracy compared to Linear Regression.

- With a lower MSE of 1.739, the LightGBM model exhibits a more refined predictive capability.

- The R2 score of 0.083 suggests a better ability to capture variance in the dataset, while the RMSE of 1.319 signifies reduced average prediction errors.

- LightGBM's boosted approach enhances efficiency and pattern-capturing capabilities, contributing to its superior performance in comparison to Linear Regression.



```
Entered the calculate_score method of the MSE class
The mean squared error value is: 1.7393631380623755
Entered the calculate_score method of the R2Score class
The r2 score value is: 0.08344673145726378
Entered the calculate_score method of the RMSE class
The root mean squared error value is: 1.3188491718397428
```

Figure 23 - LightGBM Results

The figure above (Fig. 17) shows the evaluation results from the Linear Regression model. An in-depth explanation has been done in the above section (5.2).

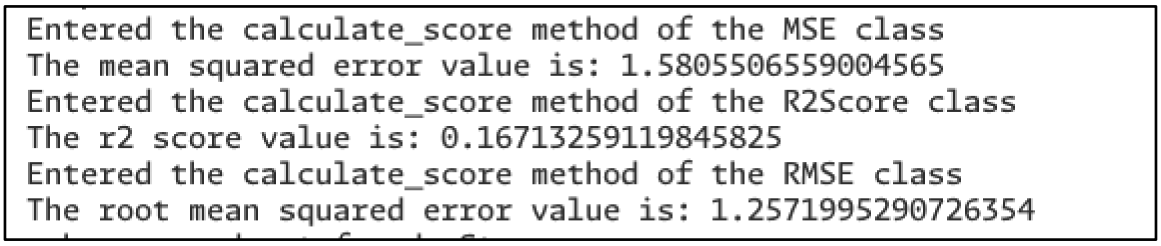
5.3 Results Using Random Forest Algorithm

The implementation of the Random Forest algorithm further advanced the predictive accuracy of our customer satisfaction model.

- With a reduced MSE of 1.581, the Random Forest model outperforms both Linear Regression and LightGBM in terms of minimizing prediction errors.

- The R2 score of 0.167 indicates a notable improvement in capturing data set variance, while the RMSE of 1.257 underscores the effectiveness of Random Forest in achieving more accurate predictions.

- The ensemble approach of Random Forest, leveraging multiple decision trees, contributes to its adaptability and resilience, resulting in enhanced predictive performance.



```
Entered the calculate_score method of the MSE class
The mean squared error value is: 1.5805506559004565
Entered the calculate_score method of the R2Score class
The r2 score value is: 0.16713259119845825
Entered the calculate_score method of the RMSE class
The root mean squared error value is: 1.2571995290726354
```

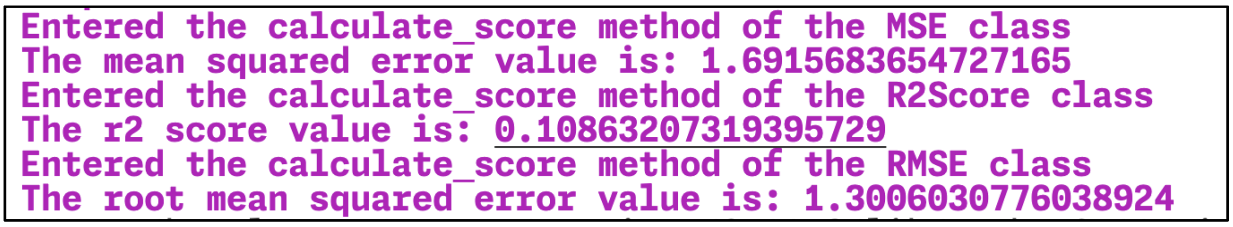
Figure 24 - Random Forest Results

The figure above (Fig. 18) shows the evaluation results from the Linear Regression model. An in-depth explanation has been done in the above section (5.3).

5.4 Results Using XGBoost Algorithm

The results produced by XGBoost algorithm is as follows:

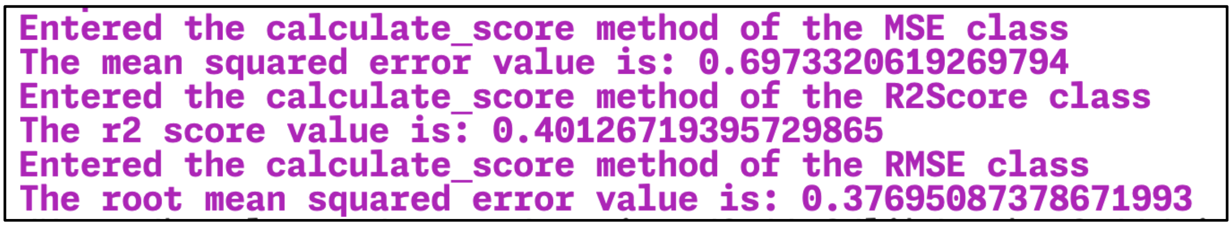
Before Hyper-parameter Tuning:



```
Entered the calculate_score method of the MSE class
The mean squared error value is: 1.6915683654727165
Entered the calculate_score method of the R2Score class
The r2 score value is: 0.10863207319395729
Entered the calculate_score method of the RMSE class
The root mean squared error value is: 1.3006030776038924
```

Figure 25 - Results Before Hyper-parameter Tuning (XGBoost)

After Hyper-parameter Tuning:



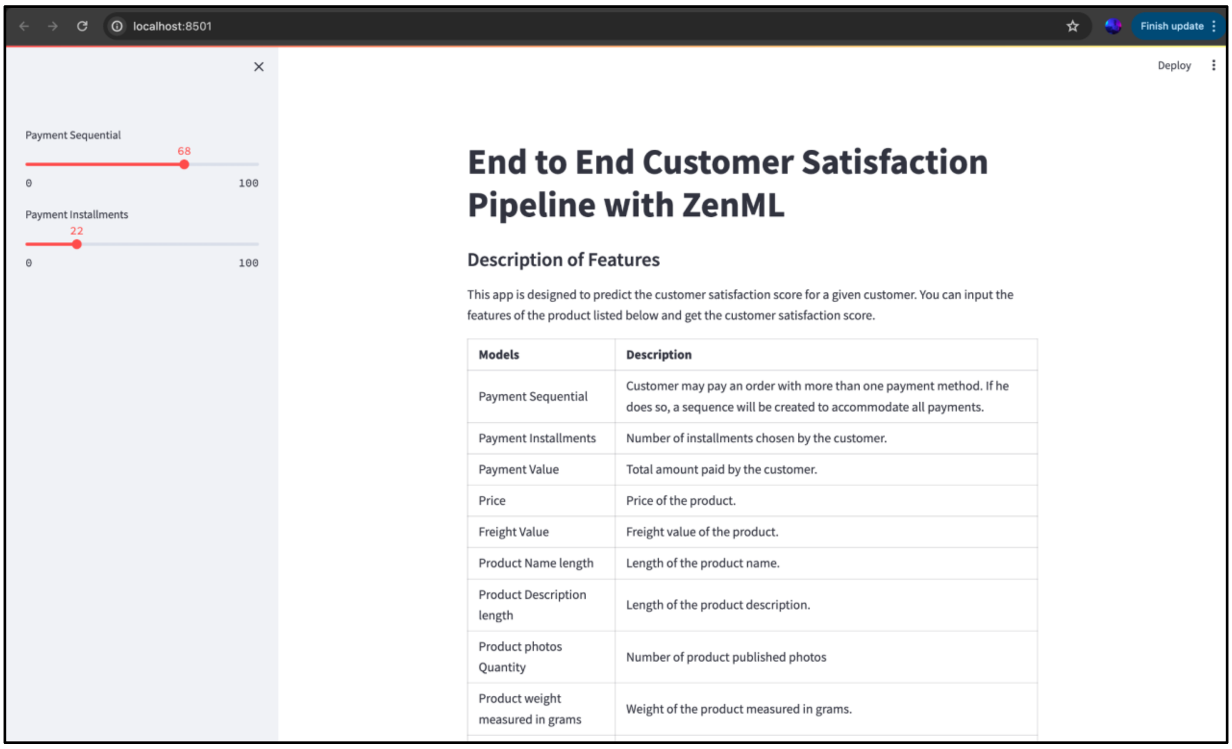
```
Entered the calculate_score method of the MSE class
The mean squared error value is: 0.6973320619269794
Entered the calculate_score method of the R2Score class
The r2 score value is: 0.40126719395729865
Entered the calculate_score method of the RMSE class
The root mean squared error value is: 0.37695087378671993
```

Figure 26 - Results After Hyper-parameter Tuning (XGBoost)

- **Reduced Mean Squared Error (MSE):**
    - XGBoost achieves a decreased MSE of 0.692, showcasing superior performance over Linear Regression and LightGBM models in minimizing prediction errors.
- **Moderate Improvement in R2 Score:**
    - The calculated R2 score of 0.4012 indicates a moderate enhancement in capturing data set variance, signifying XGBoost's ability to better explain the variability in the data compared to previous models.

- **Improved Root Mean Squared Error (RMSE):**
    - XGBoost demonstrates enhanced precision with an RMSE value of 0.376, indicating more accurate predictions and a significant reduction in prediction errors.
- **Ensemble-Based Approach:**
    - Leveraging gradient boosting and an ensemble-based approach, XGBoost iteratively improves model predictions, offering adaptability and robustness in handling complex datasets and contributing to its superior predictive performance.

5.5 Deployed Application (Using Streamlit)



End to End Customer Satisfaction Pipeline with ZenML

Description of Features

This app is designed to predict the customer satisfaction score for a given customer. You can input the features of the product listed below and get the customer satisfaction score.

Models	Description
Payment Sequential	Customer may pay an order with more than one payment method. If he does so, a sequence will be created to accommodate all payments.
Payment Installments	Number of installments chosen by the customer.
Payment Value	Total amount paid by the customer.
Price	Price of the product.
Freight Value	Freight value of the product.
Product Name length	Length of the product name.
Product Description length	Length of the product description.
Product photos Quantity	Number of product published photos
Product weight measured in grams	Weight of the product measured in grams.

Figure 27 - Deployed Application (Using Streamlit)

Our deployed application, which makes use of Streamlit, provides a simple and easy-to-use interface through which users may access our prediction model for customer happiness. Users may quickly enter product features into Streamlit and get forecasts in real time on client satisfaction levels. The application's interactive features and simplified design guarantee a smooth user experience, assisting stakeholders in making decisions that are effective. Our model becomes usable and accessible with this deployment, enabling users to make decisions based on precise forecasts.

Chapter 6: Conclusions and Future Scope

6.1 Conclusion

The comprehensive examination of our machine learning models in predicting customer satisfaction yields compelling insights into their respective performances. Among the trio of algorithms, the Random Forest emerges as the unequivocal frontrunner, showcasing its prowess in handling complex datasets and delivering superior predictive accuracy. Its ability to harness the collective strength of multiple decision trees contributes to a robust and adaptable model, minimizing prediction errors and enhancing overall performance. The nuanced understanding derived from the comparison positions Random Forest as the most effective tool for addressing the intricacies of customer satisfaction within our current project context.

6.2 Future Scope

As we stand at the juncture of our current achievements, the path ahead extends with promising opportunities for further enhancement and refinement of our "Production-Ready Customer Satisfaction Prediction" system. The future scope of our project encompasses a multifaceted approach, delving into the realms of model optimization, expanded model repertoire, cybersecurity integration, comprehensive testing strategies, and interactive model showcases. In the pursuit of a seamlessly integrated and robust solution, these future avenues beckon us to explore and innovate, ensuring that our predictive capabilities align with the dynamic landscape of customer satisfaction prediction.

Implementing a Machine Learning Security Engine - Open Appsec

Open Appsec is a machine learning security engine that preemptively and automatically prevents threats against Web Application & APIs. By integrating a security engine like this, we aim to ensure that the customer data and the overall model workflow remains resilient against potential vulnerabilities and security threats.

Improvements in UI as well as Pipelines

There are several automations that can only be implemented when hosting a server, that include improving UI to provide the choice of model to be used, this also involves optimizing pipeline in a way that they are directly linked to the frontend.

References

- S. Mäkinen, H. Skogström, E. Laaksonen and T. Mikkonen,, "Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help?" IEEE Xplore 2023, https://ieeexplore.ieee.org/abstract/document/10081336
- A. Kohei, F. Ikuya, N. Yusuke, M. Ryuya, and O. Sayuri, "Churn customer estimation method based on," ProQuest, 2023 https://www.proquest.com/docview/2791786334?pq-origsite=gscholar&amp;fromopenview=true
- P. Kunekar et al., "Evaluating the predictive ability of the LIGHTGBM," IEEE Xplore, 2023 https://ieeexplore.ieee.org/document/10080120/
- Y. Lu, "A study of Mobile User Satisfaction based on feature extraction," DrPress, 2023 https://drpress.org/ojs/index.php/HSET/article/view/8533
- S. D. Das and P. K. Bala, "What drives MLOps adoption? an analysis using the TOE framework," T and F online 2023, https://www.tandfonline.com/doi/full/10.1080/12460125.2023.2214306
- D. Kreuzberger, N. Kühl and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," in IEEE Access 2023, vol. 11, pp. 31866-31879, 2023, doi: 10.1109/
- A. Shatnwai and M. Faris, "Predicting customer retention using XGBoost and balancing methods," SemanticScholar, 2022 https://www.semanticscholar.org/paper/Predicting-Customer-Retention-using-XGBoost-and-AL-Shatnwai-Faris/e0815b682
- E. Puica, "Evaluating the efficiency of the technologies used in SCM applying MLOps model," scindo 2022, https://sciendo.com/article/10.2478/picbe-2022-0019.
- J. Ren, Y. Qi, Y. Dai, X. Wang, and Y. Shi, "AppSec: Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments," ACM Conferences, 2022 https://dl.acm.org/doi/abs/10.1145/2731186.2731199
- J. Smith et al., "Digital Learning Platforms: Usage and Trends," Educ. Technol. Rev., vol. 36, no. 2, pp. 145-162, 2021.

- Y. Lu, "A study of Mobile User Satisfaction based on feature extraction," DrPress,

2021
https://drpress.org/ojs/index.php/HSET/article/view/8533

- W. J. van den Heuvel and D. A. Tamburri, “Model-driven ML-OPS for intelligent enterprise applications: Vision, approaches and challenges,” SpringerLink, 2020 https://link.springer.com/chapter/10.1007/978-3-030-52306-0_11
- L. E. Lwakatare, I. Crnkovic, and J. Bosch, “Challenges in development of AI-enabled applications,” IEEE Xplore, 2020 https://ieeexplore.ieee.org/abstract/document/9238323
- Y. Zhou, Y. Yu, and B. Ding, “Towards MLops: A case study of ML pipeline platform ,” IEEE Xplore, 2020 https://ieeexplore.ieee.org/abstract/document/9361315
- A. Shatnwai and M. Faris, “Predicting customer retention using XGBoost and balancing methods,”SemanticScholar, 2020 https://www.semanticscholar.org/paper/Predicting-Customer-Retention-using-XGBoost-and-AL-Shatnwai-Faris/e0815b6824522dd2efa2cba46d4672e80af97487
- L. C. Siebert, J. F. B. Filho, E. J. da S. Júnior, E. K. Yamakawa, and A. Catapan, “Predicting customer satisfaction for distribution companies using machine learning,” International Journal of Energy Sector Management, 2019 https://www.emerald.com/insight/content/doi/10.1108/IJESM-10-2018-0007/full/html
- K. Verma, GitHub profile, https://github.com/verma-kunal/customer-satisfaction -prediction-mlops
- S. Kumar and M. Zymbler, “A machine learning approach to analyze customer satisfaction from airline tweets - journal of big data,” SpringerLink, 2019 https://link.springer.com/article/10.1186/s40537-019-0224-1
- L. C. Siebert, J. F. B. Filho, E. J. da S. Júnior, E. K. Yamakawa, and A. Catapan, “Predicting customer satisfaction for distribution companies using machine learning,” International Journal of Energy Sector Management, 2019 https://www.emerald.com/insight/content/doi/10.1108/IJESM-10-2018-0007/full/html
- S. Kumar and M. Zymbler, “A machine learning approach to analyze customer satisfaction from airline tweets - journal of big data,” SpringerLink, 2019 https://link.springer.com/article/10.1186/s40537-019-0224-1
- M. Alenezi and Y. Javed, “Open source web application security: A static analysis approach,” IEEE Xplore, 2016

https://ieeexplore.ieee.org/abstract/document/7745369/

- J. Ren, Y. Qi, Y. Dai, X. Wang, and Y. Shi, "AppSec: Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments," ACM 2015 Conferences, https://dl.acm.org/doi/abs/10.1145/2731186.2731199
- N. Zeinalizadeh, A. A. Shojaie, and M. Shariatmadari, "Modeling and analysis of bank customer satisfaction using neural networks approach," International Journal of Bank Marketing 2015, https://www.emerald.com/insight/content/doi/10.1108/IJBM-06-2014-0070/full/html
- N. Uke, "Design Patterns in Python," Academia.edu 2010, https://www.academia.edu/6390087/http_citeseerx_ist_psu_edu_viewdoc_download_doi_10_1_1_401_8095_and_rep_rep1_and_type_pdf
- C. Gurău and A. Ranchhod, "Measuring customer satisfaction: A platform for calculating, predicting and increasing customer profitability - journal of targeting, Measurement and analysis for marketing," SpringerLink 2002, https://link.springer.com/article/10.1057/palgrave.jt.5740047

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 14/05/24

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | [✓ B.Tech Project Report] | Paper

Name: Kunal Verma, Siddharth K. Department: CS/IT Enrolment No 201513, 201144

Contact No. 9518214467, 8219916906 E-mail. 201513@juitsolan.in, 201144@juitsolan.in

Name of the Supervisor: Prof. Dr. Pradeep Kumar Gupta

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): PRODUCTION-READY CUSTOMER SATISFACTION PREDICTION USING MLOPS

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 64
- Total No. of Preliminary pages = 5
- Total No. of pages accommodate bibliography/references = 3

[illegible]

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at......16........ (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible] 18/5/24

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String	16	Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Report 2024

ORIGINALITY REPORT

16%	13%	7%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ir.juit.ac.in:8080 Internet Source	2%
2	docs.gcp.databricks.com Internet Source	1%
3	fastercapital.com Internet Source	1%
4	Submitted to Midlands State University Student Paper	1%
5	Submitted to Liverpool John Moores University Student Paper	1%
6	Submitted to The University of the West of Scotland Student Paper	1%
7	epub.uni-bayreuth.de Internet Source	<1%
8	Submitted to Study Group Australia Student Paper	<1%
9	www.coursehero.com	

	Internet Source	<1 %
10	faculty.uut.ac.ir Internet Source	<1 %
11	media.neliti.com Internet Source	<1 %
12	docs.zenml.io Internet Source	<1 %
13	www.mdpi.com Internet Source	<1 %
14	dergipark.org.tr Internet Source	<1 %
15	ir.juit.ac.in:8080 Internet Source	<1 %
16	ntnuopen.ntnu.no Internet Source	<1 %
17	Submitted to Jaypee University of Information Technology Student Paper	<1 %
18	arxiv.org Internet Source	<1 %
19	emeritus.org Internet Source	<1 %
20	mobt3ath.com Internet Source	