**Train Delay Prediction Framework**

**Contents**

[**1. Introduction 3**](#_Toc174017792)

[**2. Analysis of Train Delay Problems 3**](#_Toc174017793)

[**2.1 Overview of Train Delays 3**](#_Toc174017794)

[**2.2 Impact of Train Delays 3**](#_Toc174017795)

[**2.3 Factors Influencing Train Delays 3**](#_Toc174017796)

[**3. Data Pre-processing & Feature Extraction 4**](#_Toc174017797)

[**3.1 Data Collection 4**](#_Toc174017798)

[**3.2 Data Cleaning 4**](#_Toc174017799)

[**3.4 Data Transformation 5**](#_Toc174017800)

[**4. Design and Implementation of the Framework 6**](#_Toc174017801)

[**4.1 Model Selection 6**](#_Toc174017802)

[**4.1.1 Stacking Regressor: 6**](#_Toc174017803)

[**4.1.2 Reason for Selection: 6**](#_Toc174017804)

[**4.1.3 Bayesian Ridge Regression: 6**](#_Toc174017805)

[**4.1.4 Convolutional Neural Network (CNN): 7**](#_Toc174017806)

[**4.2 Framework Architecture 7**](#_Toc174017807)

[**4.2.1 Data Preparation and Pre-processing: 7**](#_Toc174017808)

[**4.2.2 User Input Collection: 8**](#_Toc174017809)

[**4.2.3 Feature Engineering: 8**](#_Toc174017810)

[**4.2.4 Data Transformation: 8**](#_Toc174017811)

[**4.2.5 Prediction: 8**](#_Toc174017812)

[**4.2.6 Result Presentation: 8**](#_Toc174017813)

[**5. Train Predictive Models 8**](#_Toc174017814)

[**4.1 Model Training 9**](#_Toc174017815)

[**4.1.1 Dataset Splitting: 9**](#_Toc174017816)

[**4.1.2 Input Features: 9**](#_Toc174017817)

[**4.1.3 Output/Target Variable: 9**](#_Toc174017818)

[**5. Testing and Evaluation 10**](#_Toc174017819)

[**5.1 Test Data 10**](#_Toc174017820)

[**5.2 Evaluation Results 10**](#_Toc174017821)

[**5.3 Model Performance Summary: 11**](#_Toc174017822)

[**6. Conclusion 14**](#_Toc174017823)

[**6.1 Summary 14**](#_Toc174017824)

[**6.2 Future Work 14**](#_Toc174017825)

# 1. Introduction

In today's transport networks, train delays are a major problem that disrupt passengers' journeys and provide operational difficulties for rail operators. The efficiency of railway operations and passenger satisfaction can both be significantly increased by being able to forecast train delays. This article describes how a prediction framework to estimate train delays using historical data was designed, put into practice, and tested. The objective of this project is to create a model that can forecast possible train delays with enough accuracy to enable better planning and increased service reliability.

# 2. Analysis of Train Delay Problems

## **2.1 Overview of Train Delays**

There are a number of reasons why trains can be delayed, including as ineffective operations, bad weather, malfunctioning equipment, and human mistake. The train network as a whole may be affected by these delays, which have the potential to spread and impact later train schedules. Developing predictive models requires an understanding of the patterns and causes of train delays.

## **2.2 Impact of Train Delays**

* **Passenger Inconvenience:** Train delays are a major source of aggravation for passengers, since they can result in missed connections, tampered with travel schedules, and overall discontent. This may harm rail service providers' reputations and decrease patronage.
* **Operational Difficulties:** Coordinating personnel assignments, train schedules, and maintenance tasks becomes more difficult when there are delays. Additionally, they may result in a cascading effect, in which delays in one area of the network spread and result in additional delays.
* **Economic Costs:** Compensation claims, decreased productivity, and higher operating expenses cause train delays to cost rail operators money. In addition, passengers must pay extra for cancelled or postponed flights, missed appointments, and lost productivity.

## **2.3 Factors Influencing Train Delays**

* **Time of Day:** The time of day has a significant impact on train delays, with peak hours seeing the most delays because to larger passenger volumes and more frequent trains.
* **Day of the Week:** Delay patterns are also influenced by the day of the week. Due to varying travel habits and operational schedules, weekends and holidays may experience different delay frequencies than weekdays.
* **Weather:** Unfavourable weather can affect train operations and cause delays. Examples of these situations include rain, snow, fog, and extremely high or low temperatures. Weather-related delays can have an impact on the state of the track, visibility, and equipment performance.
* **Operational Factors:** Train congestion, signal malfunctions, track maintenance, and personnel availability are examples of operational problems that can lead to delays. To reduce delays, these elements must be managed effectively.
* **Station-specific Factors:** Depending on their location, operating limitations, or passenger numbers, some stations may have higher delay frequencies than others. Comprehending the unique delay patterns of a given station is crucial for focused interventions.

# 3. Data Pre-processing & Feature Extraction

## **3.1 Data Collection**

The appropriate modules backboard provided the dataset that was used for this project, which included a plethora of data on train schedules, actual departure and arrival times, and other important information. It focusses especially on the trip from Norwich to London Liverpool Street, including stops at intermediate locations. The records include a range of days and cover the years 2017 through 2021 in order to provide a complete sample of data. In order to streamline the analytical procedure, I utilised a Python script to combine several separate data frames into a unified data frame, guaranteeing a sturdy dataset for subsequent processing.

## **3.2 Data Cleaning**

Prior to utilising predictive modelling, a comprehensive data cleaning procedure was necessary to raise the quality and relevance of the dataset. With around 1.5 million entries in the original dataset, thorough filtering was required to remove any unnecessary or partial information.  
  
I began by locating and keeping only the most important columns for our analysis—train schedules, real departure and arrival times, and other pertinent information. I made the decision to eliminate any rows containing null values because I was aware that lacking journey data could have a negative impact on prediction results. In order to avoid skewing the analysis, it was imperative to exclude incomplete trips and incomplete records. The dataset was drastically reduced to just over 494,000 rows when these filtering criteria were applied, and all of the complete, high-quality records made up the whole dataset. By ensuring that the analysis that followed was based on reliable data, this thorough cleaning procedure raised the general trustworthiness of our predictive modelling efforts.

**3.3 Feature Extraction**

A number of procedures were taken during the feature extraction phase in order to produce pertinent qualities for the predictive model.

* **Recognising Distinct Stations:** I started by taking a sample from each unique station that was in the dataset. I used internet research to arrange these stations according to the sequence in which they appear throughout the trip in order to preserve consistency. The list included intermediate stations in the order that they appeared, starting with Norwich and ending with Liverpool. At the conclusion of the list, I have added stations that are not on the direct route from Norwich to Liverpool.
* **Compute Stops:** In order to put the voyage into perspective, I created a numerical column that shows how many stops each train will make depending on how far it is from the starting point (Norwich). The calculation of this column involved figuring out where each station was in the sorted list.
* **Developing Binary Features:** In order to give the travels more context, I introduced the following binary features:
  + **is\_weekend:** This feature determines if a trip happened over the weekend. It was taken from the 'rid' column, where the date is shown in YYYYMMDD format for the first eight characters.
  + **is\_offpeak:** This feature lets you know if a trip took place during off-peak hours. Travel that occurs on a weekend or between the hours of 9:00 AM and 9:00 PM on weekdays is categorised as off-peak.
* **Retrieving Date Elements:**
  + Information from the 'rid' column was extracted to create a new column for the date.
  + Using pandas functionality, the 'year' was extracted from this new date column.
  + 'Day' was used as a binary indication to determine whether the date is a weekend (1 if it is, 0 otherwise), and the other variables,'month' and 'day', were also taken from the date column.
* **Establishing Time Features:** The given departure time was used to generate the 'HOUR' and 'MIN' features, which gave a more detailed picture of the journey's temporal components.
* **Determine the Average Delay:** In order to provide the model with context, I calculated the average delay for every distinct station. This was accomplished by utilising the pandas groupby function to aggregate the delay data and produce a static value that represents the average delays for each station.
* **Categorical Variable Encoding:** For categorical variables like "STATION" and "DAY," I used LabelEncoder to prepare the dataset for machine learning methods. The categorical data could now be represented numerically and used as model input thanks to this transformation.
* **Numerical Scaling Features:** StandardScaler was used to standardise the numerical features. Through this approach, each feature was set to have a mean of 0 and a standard deviation of 1, which enhanced model performance and sped up training.

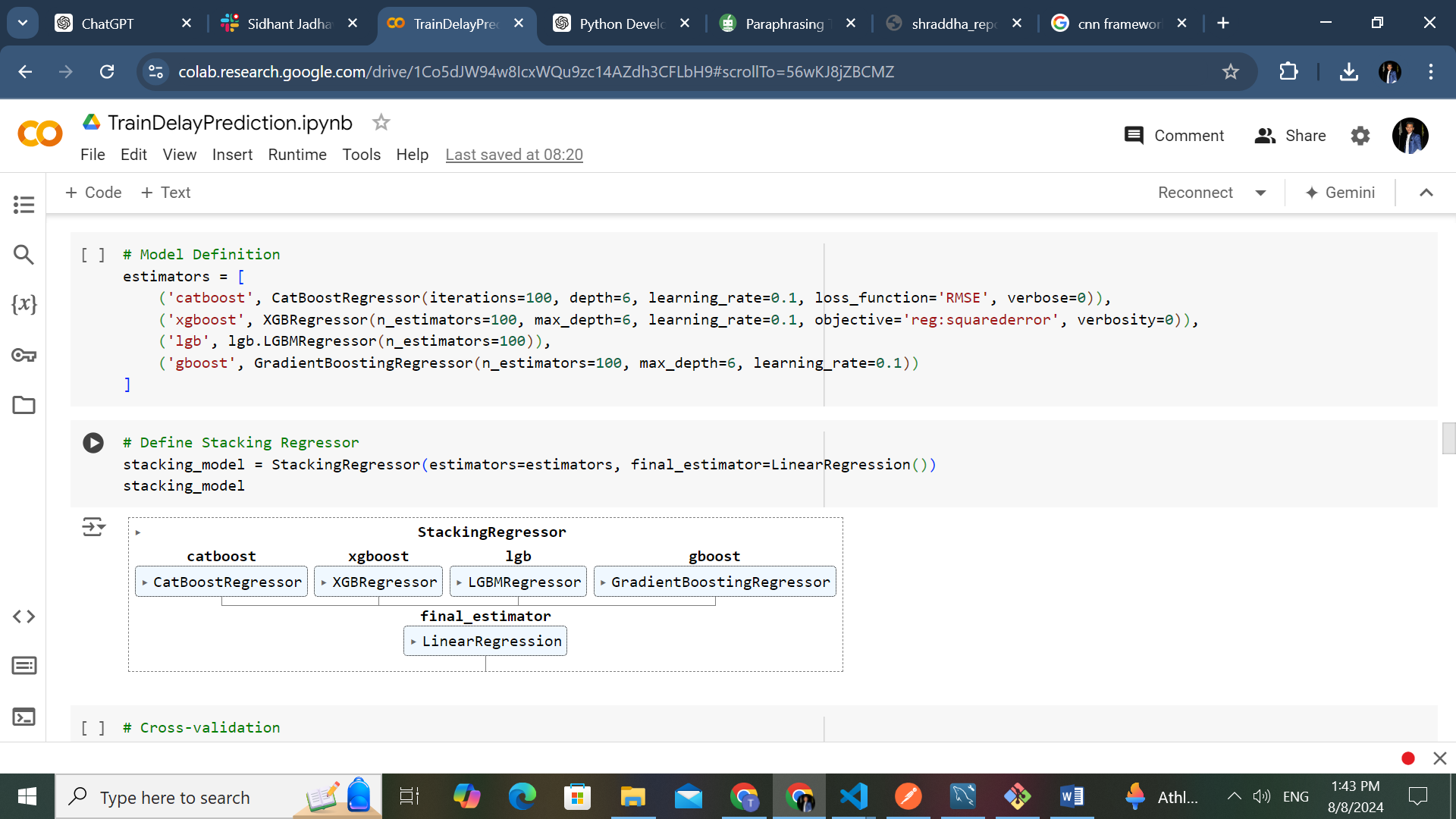
## **3.4 Data Transformation**

The following actions made up the data transformation phase:

* **Data Normalisation:** To standardise all of the numerical input features, I used StandardScaler. By making sure that every feature contributed equally to the learning process, this modification not only enhanced the model's performance but also accelerated the training convergence.
* **Encoding Categorical Data:** I used LabelEncoder to encode the categorical variables, especially the 'STATION' and 'DAY' columns. This method ensured compliance with the selected machine learning models by converting categorical data into numerical representations.
* **Developing New Features:** I created new properties like "stops," "is\_weekend," "is\_offpeak," "HOUR," "MIN," and "avg\_delay" during the feature engineering process. These attributes were created to give the prediction models more thorough inputs so that the algorithms might gain knowledge from a larger dataset.
* **Handling Missing Data:** I decided to eliminate rows in the dataset that included null values in order to protect data integrity and reduce potential biases. This tactic made sure the models were trained on complete, high-quality data sets, which improved the prediction framework's dependability.

# 4. Design and Implementation of the Framework

## **4.1 Model Selection**



***Figure 1: - Model Selection***

### **4.1.1 Stacking Regressor:**

* **Components:** The stacking regressor incorporates four models: CatBoost, XGBoost, GradientBoostingRegressor, and LightGBM.
* **Final Estimator:** Linear Regression is utilized as the final estimator in this ensemble.

**4.1.2 Reason for Selection:**  
The stacking regressor is favored for its ability to enhance predictive performance by combining various regression models. In this setup, four advanced boosting algorithms are employed, each offering unique advantages.

* **CatBoost** is particularly effective in managing categorical variables and includes features to mitigate overfitting, making it well-suited for datasets with mixed data types.
* **XGBoost** excels in terms of speed and performance, especially with structured datasets, providing robust predictive capabilities.
* **LightGBM** is designed for efficiency and is highly effective when dealing with large datasets, allowing for rapid training and minimal memory usage.
* **Gradient Boosting** serves as a fundamental model that supports additive modelling, contributing to the overall predictive accuracy of the ensemble.

By leveraging the strengths of these four models alongside Linear Regression, the stacking regressor is capable of capturing diverse patterns in the data, thereby improving accuracy and robustness compared to relying on a single model.

### **4.1.3 Bayesian Ridge Regression:**

Bayesian Ridge Regression was selected for its capacity to integrate prior knowledge and provide probabilistic predictions. This model is beneficial for assessing uncertainties and managing multicollinearity among features, making it a strong candidate for regression problems involving complex relationships.

### **4.1.4 Convolutional Neural Network (CNN):**

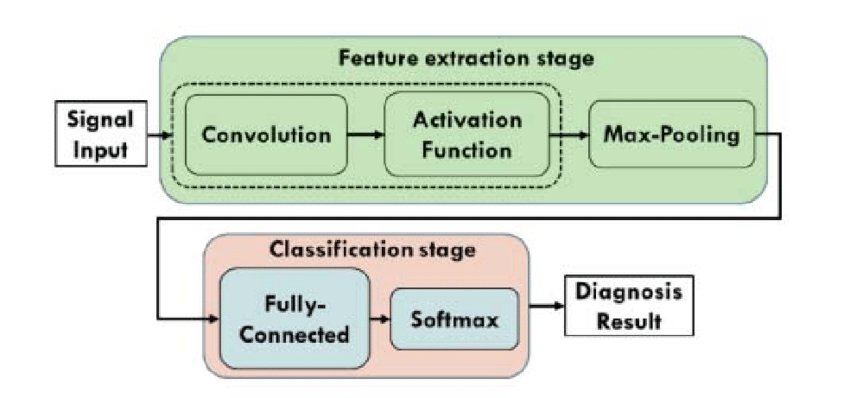
**Architecture:**

* **1D Convolutional Layer:** Utilizes 64 filters with a kernel size of 3 and applies the ReLU activation function.
* **MaxPooling Layer:** Implements a pooling size of 2 to reduce the dimensionality of the feature maps.
* **Flatten Layer:** Converts the output from the convolutional layer into a 1D array, preparing it for further layers.
* **Dense Layer:** Comprises 128 neurons with ReLU activation, enabling the model to capture non-linear relationships in the data.
* **Output Layer:** Contains a single neuron to output the regression prediction.

The model is configured with the Adam optimizer and employs the Mean Squared Error loss function. It is trained over 10 epochs with a batch size of 32, which strikes a balance between training speed and performance.

Convolutional Neural Networks are proficient in recognizing local patterns and temporal dependencies within sequential data. The use of a 1D convolutional layer allows the model to identify crucial features across the input sequence, while the max pooling layer helps in dimensionality reduction and feature extraction. The dense layers further refine the learning process by enabling the model to understand complex, non-linear interactions. This architecture is particularly advantageous for time series data, which is critical for improving prediction accuracy in train delay scenarios.

### **4.2 Framework Architecture**



***Figure 2: - CNN Architecture***

## **4.2.1 Data Preparation and Pre-processing:**

* **Loading Pre-Trained Models:** The initial step involves loading the pre-trained models and encoders, which include the scaler, station encoder, day encoder, and stacking model.
* **Defining Station Lists and Average Delay Dictionary:** A structured list of stations along with a dictionary that records average delays for each station is created to assist in feature engineering.
* **Utility Functions for Data Processing:** Various utility functions are defined to streamline the data processing workflow, enhancing the efficiency of the framework.

## **4.2.2 User Input Collection:**

* **Streamlit User Interface:** The framework utilizes Streamlit to create an intuitive user interface, enabling users to input their intended destination station, departure time, and travel date.

## **4.2.3 Feature Engineering:**

* **Calculating Number of Stops:** The number of stops is determined based on the selected station's position within the predefined list of stations.
* **Determining Weekend Status:** The framework checks whether the journey falls on a weekend by analyzing the date derived from user input.
* **Identifying Off-Peak Hours:** The framework assesses whether the journey occurs during off-peak times based on established criteria.
* **Extracting Day of the Week:** The day of the week is extracted from the departure date to provide additional context for the prediction process.
* **Extracting Hour and Minute from Departure Time:** The hour and minute values are derived from the user's specified departure time, enriching the feature set.
* **Retrieving Average Delay:** The framework fetches the average delay associated with the selected station from the predefined average delay dictionary.

## **4.2.4 Data Transformation:**

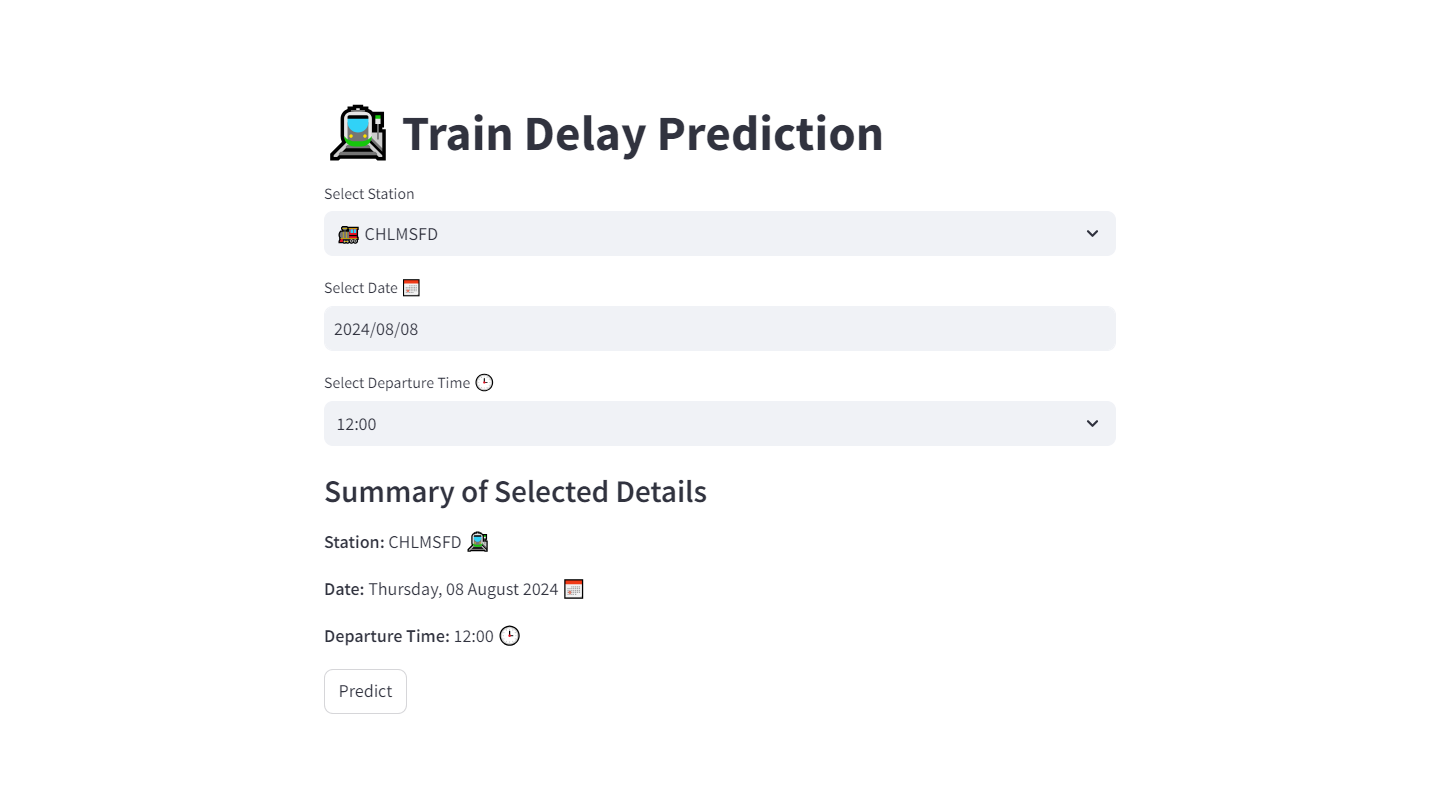
* **Encoding Categorical Variables:** Categorical variables, including station names and days, are converted into numerical formats using the pre-trained LabelEncoder to ensure compatibility with the model.
* **Creating a Feature DataFrame:** A new DataFrame is constructed to encapsulate all relevant features necessary for generating predictions.
* **Handling Missing Values:** If any missing values are found, the framework employs the SimpleImputer to fill in these gaps, thereby preserving the quality of the dataset.
* **Scaling Features:** The numerical features are standardized using the pre-trained scaler, which ensures that they are appropriately adjusted for optimal model performance.

## **4.2.5 Prediction:**

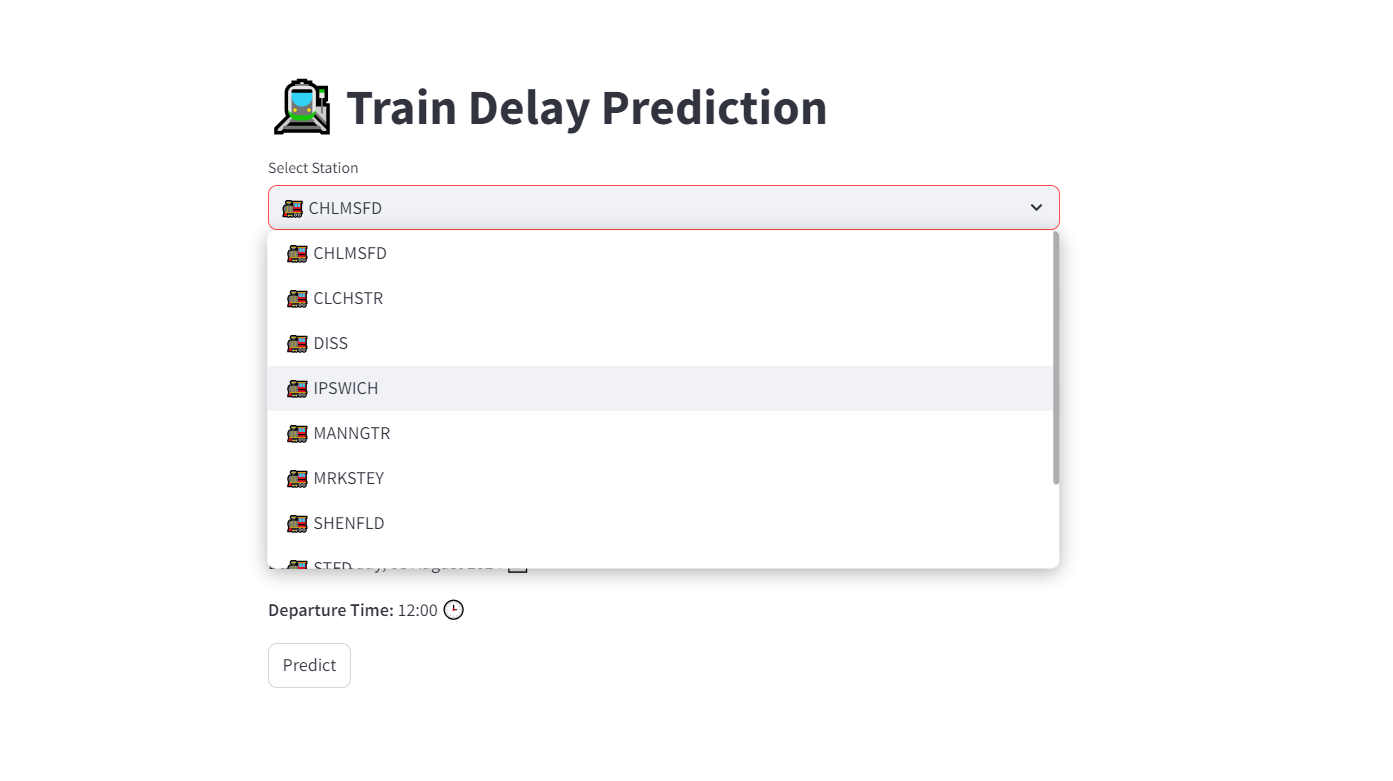
* **Using Pre-Trained Model for Prediction:** The final prediction is made using the pre-trained model, which processes the prepared input features.

## **4.2.6 Result Presentation:**

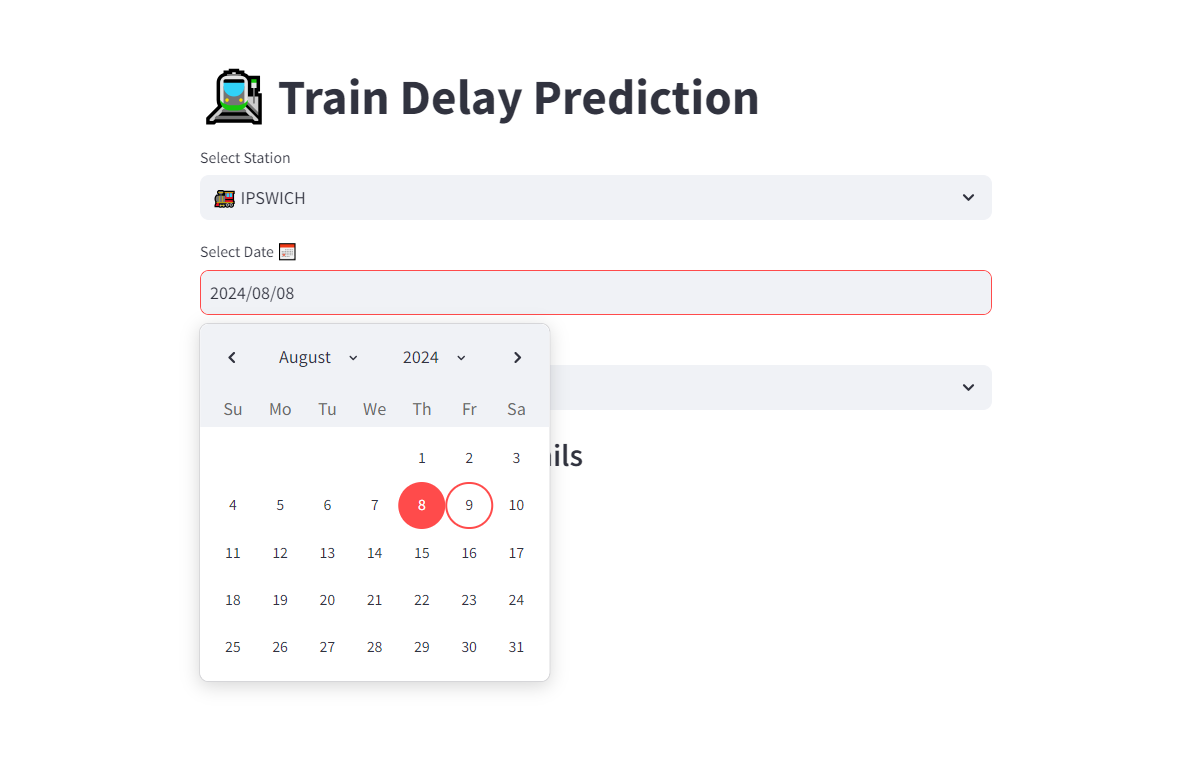
* **Displaying Results via Streamlit:** The predicted delay is presented to the user through the Streamlit interface, allowing for a straightforward interpretation of the results.



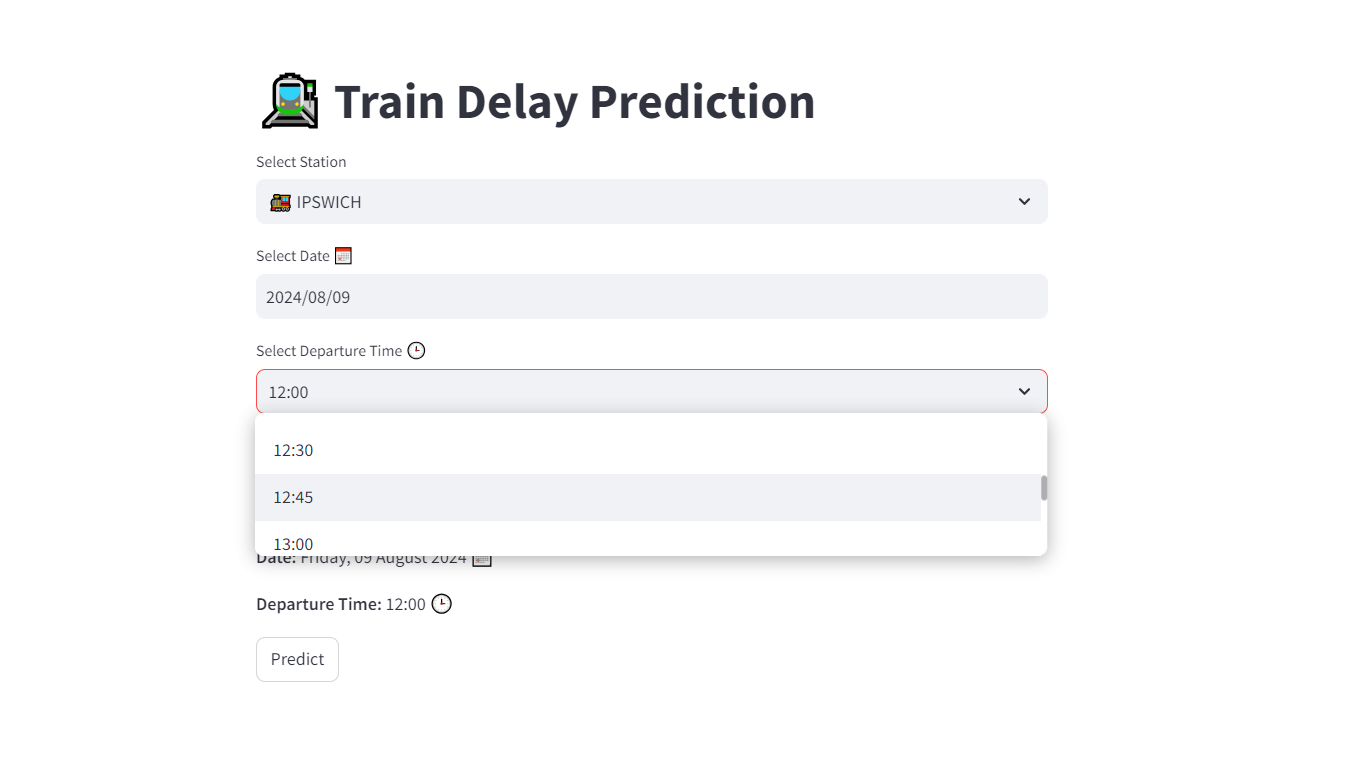
***Figure 3:- Landing page***



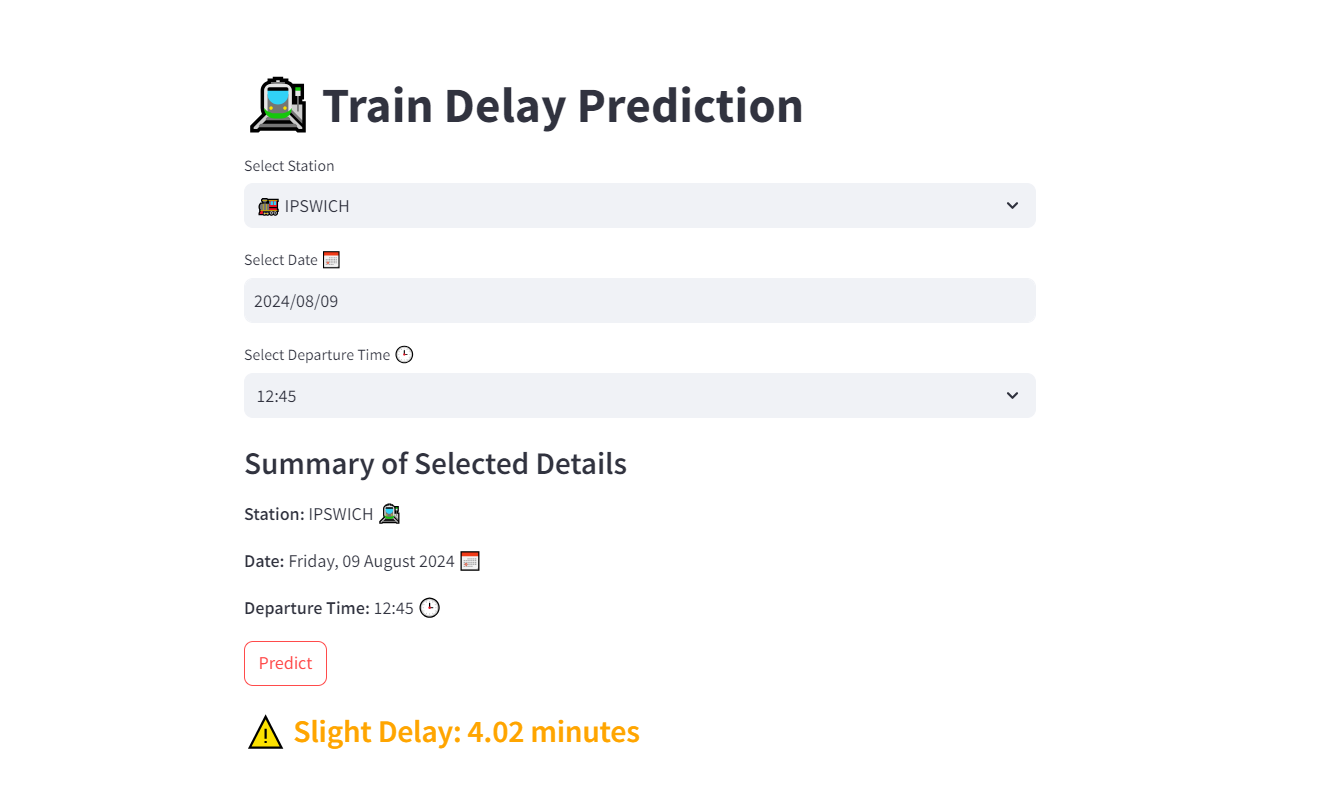
***Figure 4: - Station selection***



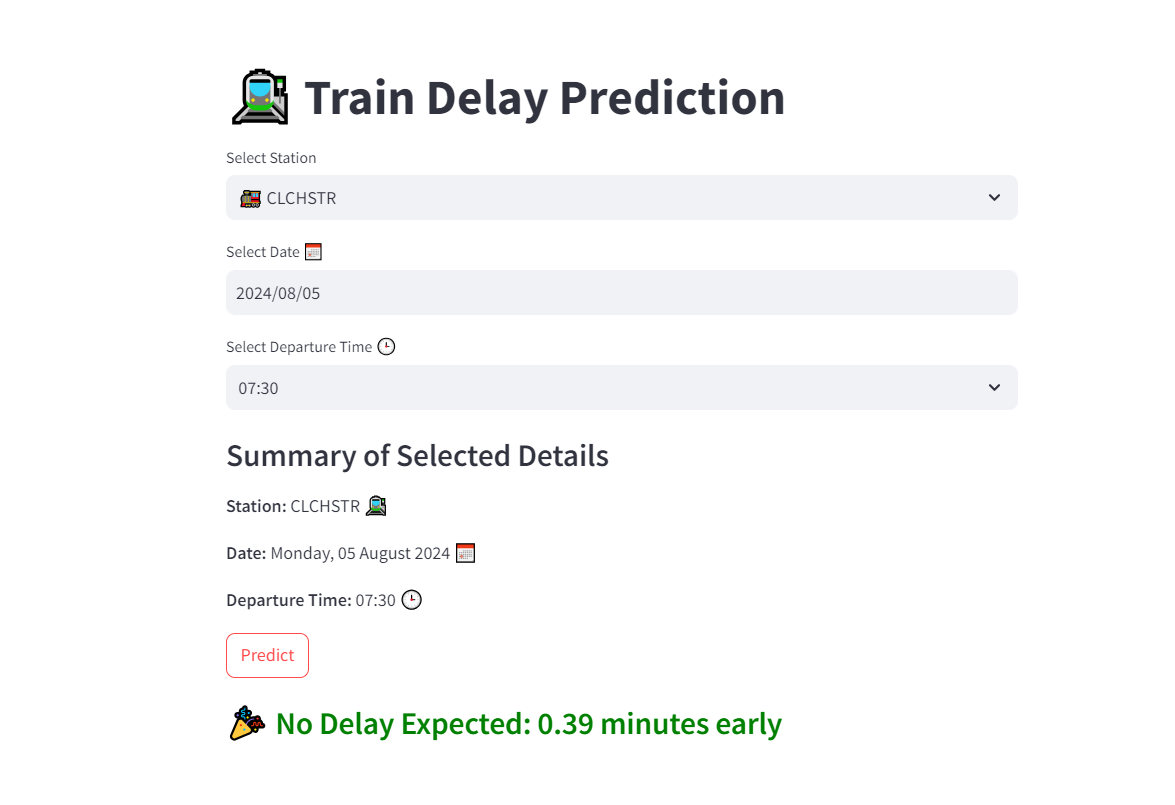
***Figure 5: - Select Date***



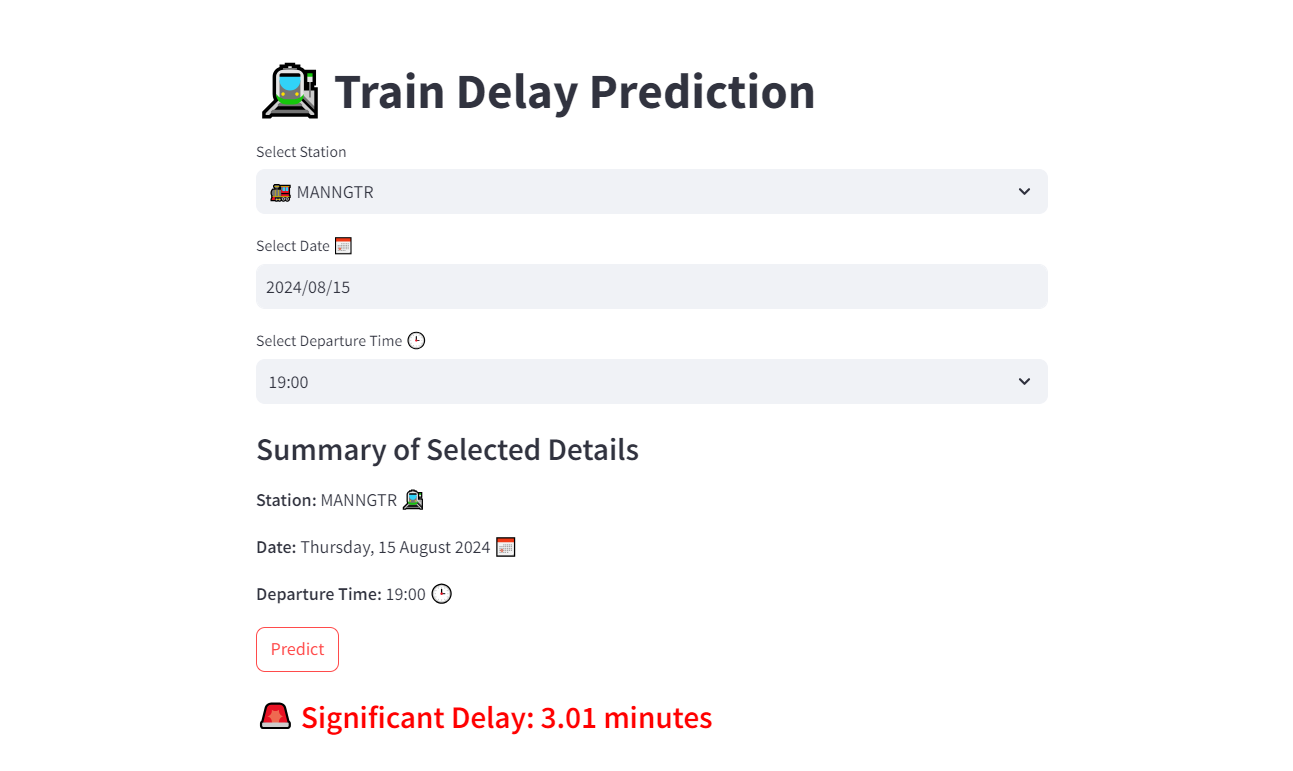
***Figure 6: - Select Departure Time***



***Figure 7: - Predicted Train Delay*** ***(Slight Delay)***



***Figure 8: - Predicted Train Delay (No Delay)***



***Figure 9: - Predicted Train Delay (Significant Delay)***

# 5. Train Predictive Models

## **4.1 Model Training**

The training phase of the predictive modelling process involved splitting the dataset into distinct subsets to ensure robust evaluation of the model's performance. This step was crucial to assess how well the model could generalize to unseen data.

## **4.1.1 Dataset Splitting:**

The entire dataset, which consists of 494,228 rows and 21 columns after data preprocessing and feature extraction, was divided into two subsets in an 80:20 ratio. This division means that 80% of the data was allocated for training the model, while the remaining 20% was reserved for testing its performance. The rationale behind this splitting strategy is to provide the model with a substantial amount of data for learning while maintaining a separate set for validating its predictive capabilities.

## **4.1.2 Input Features:**

The input features used for training the predictive models included:

* **STATION:** Categorical variable representing the train station.
* **stops:** Numerical feature indicating the number of stops in a journey based on the station order.
* **is\_weekend:** Binary feature signifying whether the journey occurred on a weekend.
* **is\_offpeak:** Binary feature indicating whether the journey took place during off-peak hours.
* **DAY:** Categorical variable representing the day of the week derived from the ride ID.
* **HOUR:** Numerical feature extracted from the planned departure time, indicating the hour of departure.
* **MINUTES:** Numerical feature derived from the planned departure time, representing the minutes of departure.
* **year:** Numerical feature representing the year extracted from the ride ID.
* **month:** Numerical feature indicating the month extracted from the date.
* **day:** Numerical feature representing the day of the month.
* **weekday:** Numerical feature indicating the day of the week (0 for Monday, 6 for Sunday).
* **avg\_delay:** Numerical feature capturing the average delay specific to each station.

## **4.1.3 Output/Target Variable:**

The output variable, or target variable, for the model was defined as:

* **target:** This variable represents the train delay in minutes, calculated as the difference between the actual arrival time and the planned arrival time. It serves as the primary objective that the model aims to predict based on the input features.

During the model training phase, various algorithms were employed to learn the underlying patterns within the training data. The models were trained on the selected input features, allowing them to understand the relationships between these features and the target variable. This training process is essential for enabling the models to make accurate predictions on new, unseen data during the testing phase.

# 5. Testing and Evaluation

## **5.1 Test Data**

For evaluating the performance of our predictive models, the dataset was divided into training and testing subsets, where the test data represented 20% of the entire dataset, comprising approximately 48,000 rows. The splitting process was conducted using the train\_test\_split function from the scikit-learn library, which ensured a random and reproducible partitioning of the data.

To maintain the independence of the test set from the training data and avoid any data leakage, only the transformation of encoded and scaled variables was performed on the test set. The fitted transformations obtained from the training data were applied directly to the test set without refitting any model parameters. This approach guarantees that the evaluation results accurately reflect the model's performance on unseen data, which is essential for assessing its generalizability and robustness. The test data was consistently managed across all predictive models, ensuring a uniform evaluation methodology for fair comparison.

## **5.2 Evaluation Results**

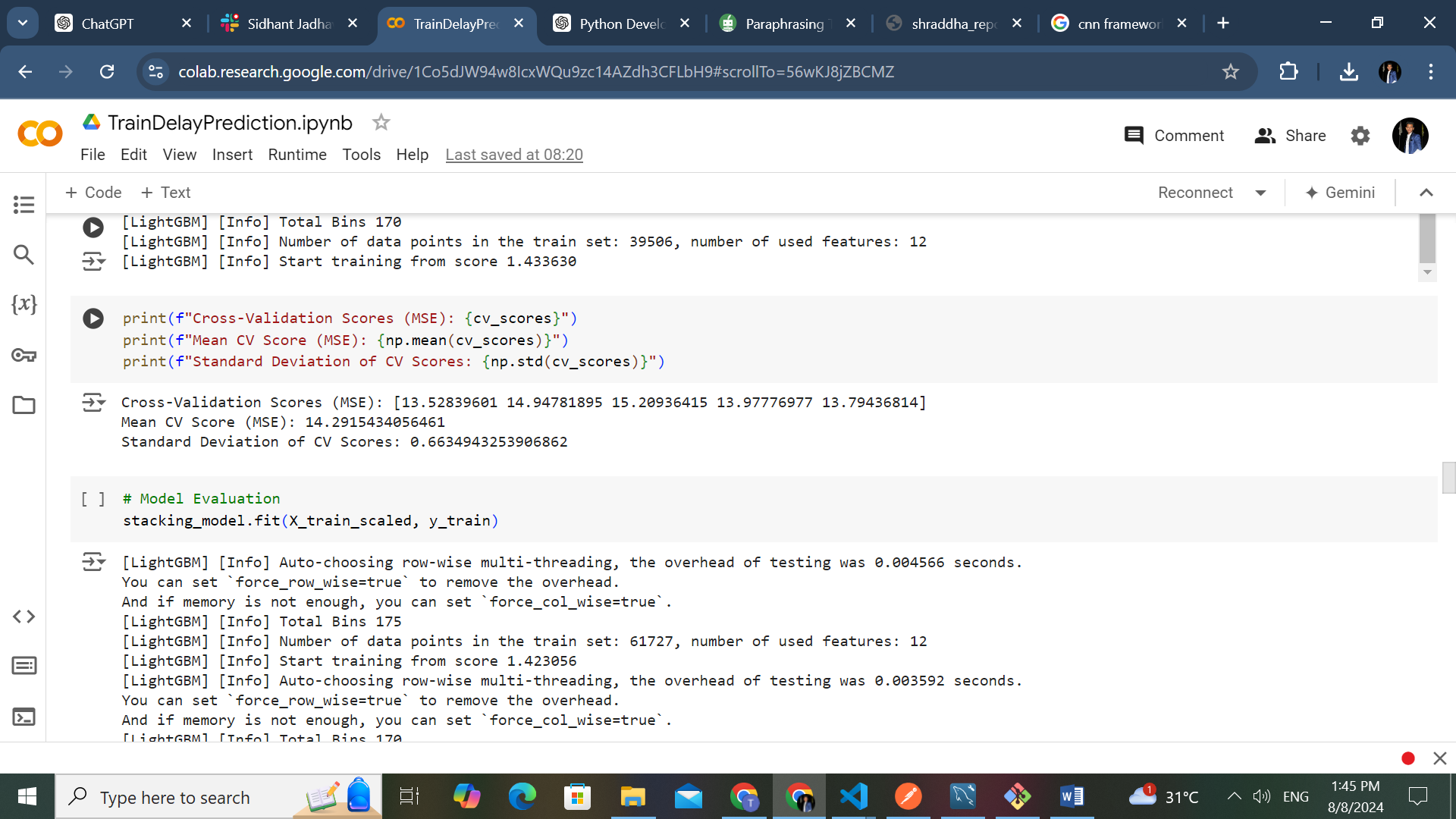
The performance of the predictive models was assessed using several regression evaluation metrics, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R²). These metrics provide valuable insights into the accuracy and reliability of the predictions made by the models.

The results of the cross-validation and the evaluation metrics for each model are as follows:

**Cross-Validation Scores (MSE):**

* [13.5284, 14.9478, 15.2094, 13.9778, 13.7944]

**Mean Cross-Validation Score (MSE):** 14.2915  
**Standard Deviation of Cross-Validation Scores:** 0.6635



***Figure 10:- Evaluation Metrics***

**Overall Evaluation Metrics:**

* **Mean Squared Error (MSE):** 12.8004
* **Root Mean Squared Error (RMSE):** 3.5778
* **R-squared (R²):** 0.3029

## **5.3 Model Performance Summary:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **MSE** | **RMSE** | | **R²** | | --- |  |  | | --- | |  | |
| |  | | --- | | ***Stacking Model*** |  |  | | --- | |  | | 12.8004 | 3.5778 | 0.3029 |
| ***XGBoost*** | |  | | --- | | 12.7418 |  |  | | --- | |  | | 3.5696 | 0.3061 |
| ***CatBoost*** | 13.1284 | 3.6233 | 0.2851 |
| ***LightGBM*** | 13.3326 | 3.6514 | 0.2739 |
| |  | | --- | | ***Gradient Boosting*** |  |  | | --- | |  | | |  | | --- | | 16.6757 |  |  | | --- | |  | | 4.0836 | 0.0919 |

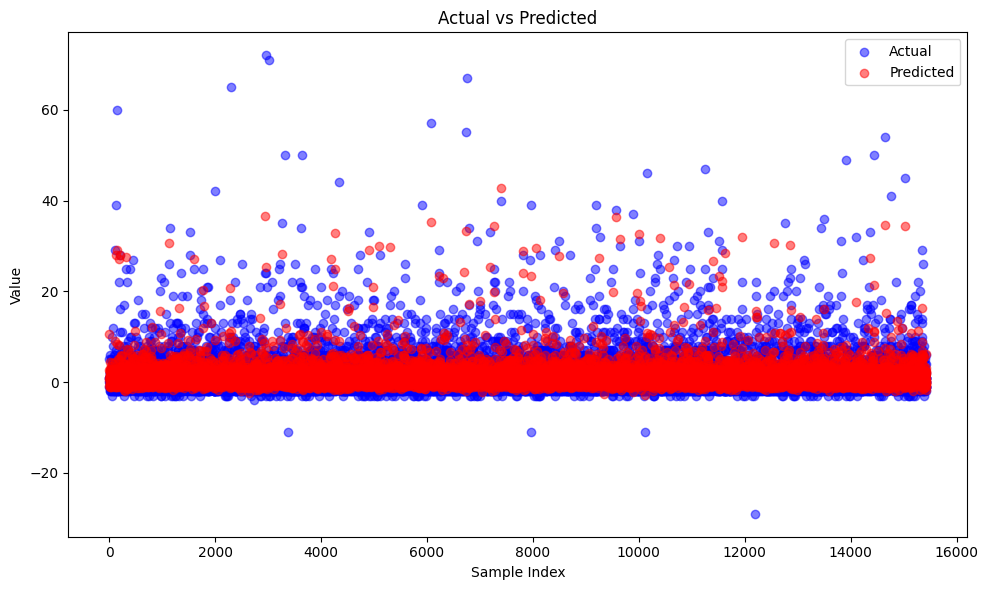
***Table 1: -Model Performance***

From the evaluation, the stacking model achieved a mean squared error of 12.8004, indicating that it performed well in predicting the delays. The corresponding R² score of 0.3029 suggests that about 30.29% of the variance in the target variable could be explained by the input features used in the model.

The XGBoost model closely followed the stacking model, with a slightly lower mean squared error of 12.7418 and a marginally higher R² score of 0.3061, demonstrating its effectiveness in capturing patterns within the data.

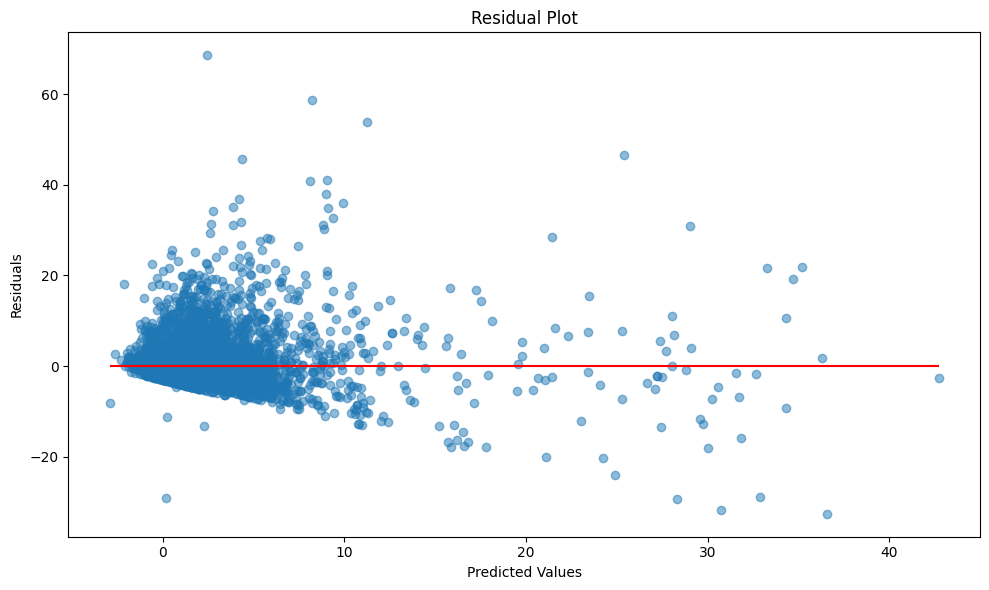
In contrast, the CatBoost and LightGBM models exhibited higher MSE values and lower R² scores, suggesting they were less effective in this context. The Gradient Boosting model performed the least effectively, with a mean squared error of 16.6757 and an R² score of only 0.0919, indicating a limited ability to explain the variance in the target variable.

To ensure the reliability of the evaluation results and to mitigate concerns regarding overfitting or underfitting, K-fold cross-validation was employed. This method partitioned the training data into multiple subsets, allowing for a more robust assessment of each model's performance and confirming that the obtained results are both valid and generalizable.



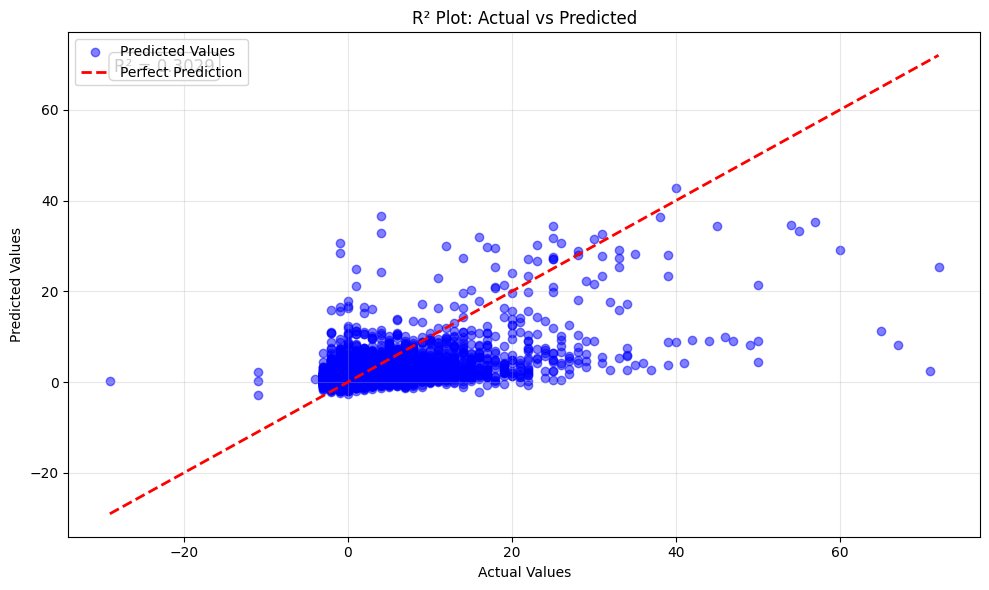
***Figure 11: - Actual vs Predicted***

The scatter plot titled "Actual vs Predicted" visually compares the actual target values (in blue) with the predicted values (in red) from the model. Each point on the plot represents a sample in the test dataset, with the x-axis indicating the sample index and the y-axis representing the predicted and actual values. A close alignment of the blue and red points suggests that the model accurately predicts the target variable, while significant deviations indicate discrepancies between the actual and predicted values. Overall, the visualization helps assess the model's performance and identify areas for improvement.



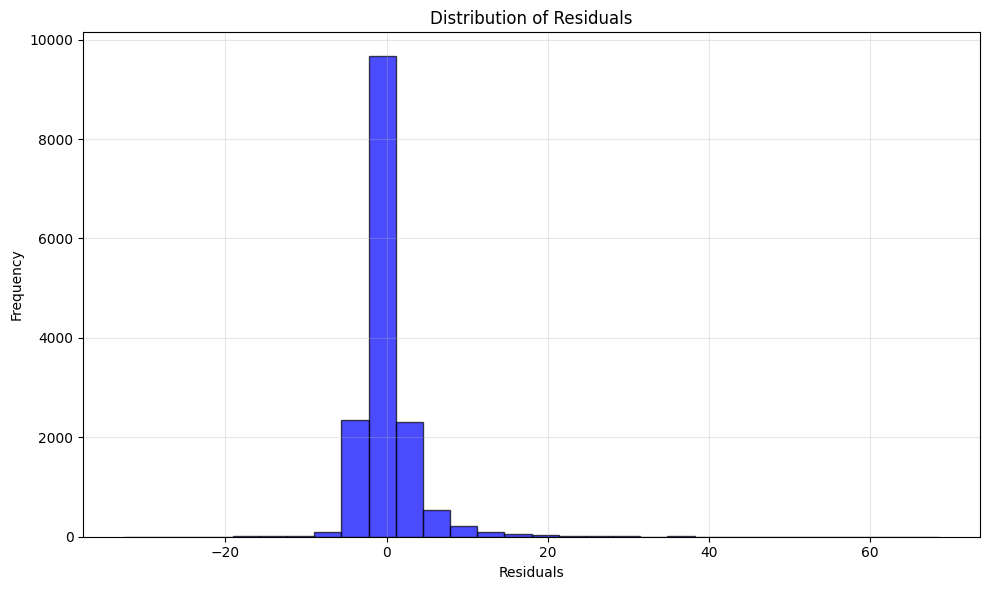
***Figure 12: - Residual Plot***

The residual plot displays the relationship between the predicted values (x-axis) and the residuals (y-axis), which are the differences between the actual target values and the predicted values. Each point represents a sample from the test dataset, showing how far off the predictions are from the actual values. The horizontal red line at zero indicates where the residuals would be if the predictions were perfect. Ideally, the residuals should be randomly scattered around this line without any discernible pattern. A random distribution suggests that the model's predictions are unbiased and that the model has effectively captured the underlying relationships in the data. If there are patterns or systematic trends in the residuals, it may indicate issues with the model, such as underfitting or overfitting, suggesting that further refinement may be necessary.



***Figure 13: - R² Plot: Actual vs Predicted***

The R² plot displays the relationship between the actual values (x-axis) and the predicted values (y-axis) from the model. The blue points represent individual predictions, illustrating how closely they align with the actual values. The dashed red line indicates the line of perfect prediction, where predicted values would equal the actual values; ideally, all blue points would lie on this line. The R² value, displayed in the top-left corner of the plot, quantifies the proportion of variance in the actual values that can be explained by the model's predictions. An R² value close to 1 indicates a strong correlation between the predicted and actual values, while a value closer to 0 suggests a weaker correlation. This plot helps visually assess the model's performance: the closer the blue points are to the red line, the more accurate the predictions are, and the better the model performs overall.



***Figure 14: - Residual Distribution***

# 6. Conclusion

## **6.1 Summary**

The development of the predictive framework for train delay prediction has been a comprehensive endeavor that involved meticulous data collection, preprocessing, feature engineering, model selection, training, testing, and evaluation. Through a well-structured approach, we successfully harnessed various machine learning techniques to create a model capable of providing insights into train delays on the Norwich to London Liverpool Street route.

Our analysis began with the gathering of extensive data that encompassed train schedules and actual departure and arrival times, spanning several years. This data was meticulously cleaned and transformed to ensure high quality, focusing on eliminating null values and irrelevant features, which led to a more manageable dataset. The feature engineering process played a crucial role in enriching the dataset with relevant attributes, allowing for a more nuanced understanding of the factors influencing train delays.

In the model selection phase, we opted for a stacking ensemble method that integrated the strengths of multiple boosting algorithms, including CatBoost, XGBoost, LightGBM, and Gradient Boosting, complemented by a Linear Regression final estimator. This combination capitalized on the unique advantages of each model, enhancing overall predictive performance. Additionally, Bayesian Ridge Regression and a Convolutional Neural Network were explored to further capture the complexities of the data, demonstrating the versatility of our approach.

The evaluation of the models revealed that the stacking model and XGBoost performed exceptionally well, achieving competitive metrics in terms of MSE and R² scores. The use of K-fold cross-validation provided additional confidence in the robustness of our results, ensuring that the models were not only effective on the training set but also capable of generalizing to unseen data.

Overall, this project underscores the significance of thoughtful design and implementation in predictive modelling. The framework we developed has the potential to offer valuable predictions regarding train delays, which could aid in improving operational efficiency and passenger experience in rail transportation. Future work may involve further refinement of the models, exploration of additional features, or the integration of real-time data to enhance prediction accuracy and timeliness.

## **6.2 Future Work**

Moving forward, several avenues can be explored to enhance the predictive framework and its applications:

1. **Incorporation of Real-Time Data**: Integrating real-time data, such as current weather conditions, train maintenance schedules, and live traffic updates, could improve the accuracy of predictions by capturing factors that may not be evident in historical data alone.
2. **Expansion to Additional Routes**: While this project focused on the Norwich to London Liverpool Street route, extending the framework to include other routes could provide a broader understanding of train delay patterns across the rail network.
3. **Exploration of Advanced Modelling Techniques**: Investigating additional machine learning techniques, such as ensemble methods beyond stacking, or more complex deep learning architectures, could potentially yield improved performance.
4. **User Interface Enhancements**: Enhancing the user interface of the prediction tool can provide a more user-friendly experience, allowing for easier input of parameters and clearer presentation of results.
5. **Feedback Mechanism**: Implementing a feedback loop where users can provide insights on prediction accuracy could facilitate continuous model improvement based on real-world performance.
6. **Longitudinal Studies**: Conducting longitudinal studies to analyse how changes in infrastructure, scheduling, and operational practices affect train delays over time can lead to actionable insights for railway management.