DATA INTENSIVE COMPUTING - CSE 587

FLIGHT DELAY PREDICTION - PHASE II REPORT

Vykunth Premnath
50487477

Sai Kumar Thoppae Sethu Raman
50483871

Problem Statement:

Flight delays have significant economic consequences for airline corporations, and studies have shown that weather conditions are a major cause of these delays. The aim of this project is to develop a predictive model using supervised machine learning algorithms to predict the occurrence and duration of flight delays caused by adverse weather conditions. The model focuses on predicting departure delays of flights, which is crucial for airlines to manage their resources and optimise their operations.

Introduction:

In the previous phase, we performed EDA and preprocessed the data . We obtained a dataset which was clean, normalised. A feature ranking method was implemented to reduce the number of columns or features. The top flight features are selected according to rank and weather features of the destination airport have only been considered (Y), dropping the features of origin airport(x). After this process, our dataset consisted of 163040 rows and 24 columns.

In this phase, we will be predicting the flight delays using various regression algorithms using historical flight data to train and test. By accurately predicting flight delays, the project can help airlines and other stakeholders take proactive measures to minimise the impact of delays, such as rescheduling flights. This can help improve customer satisfaction, reduce costs, and increase operational efficiency in the aviation industry.

Data Source

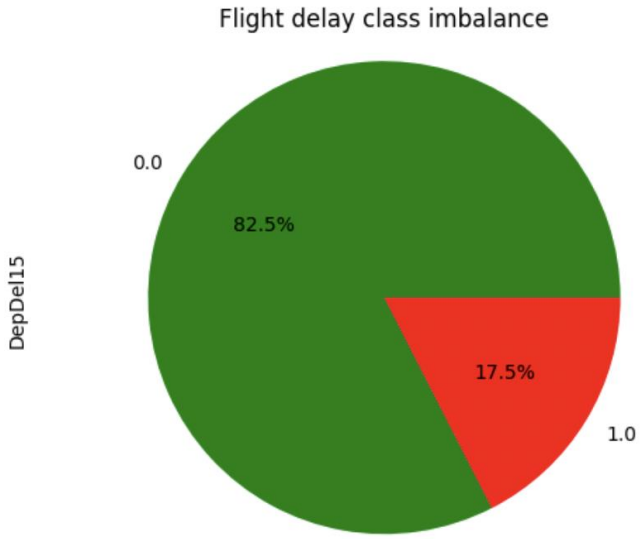
[1] https://transtats.bts.gov/Homepage.asp,

[2] https://www.kaggle.com/datasets/robikscube/flight-delay-dataset 20182022?resource=download.

[3] Dataset

Data Imbalance Problem:

The column 'DepDel15' indicates whether a flight has been delayed (1) or not delayed (0). As seen from the below figure, it can be observed that there is a very high class imbalance between the two values.




```
0.0    134549 rows

1.0     28491 rows

Column Name: DepDel15, datatype: int64
```

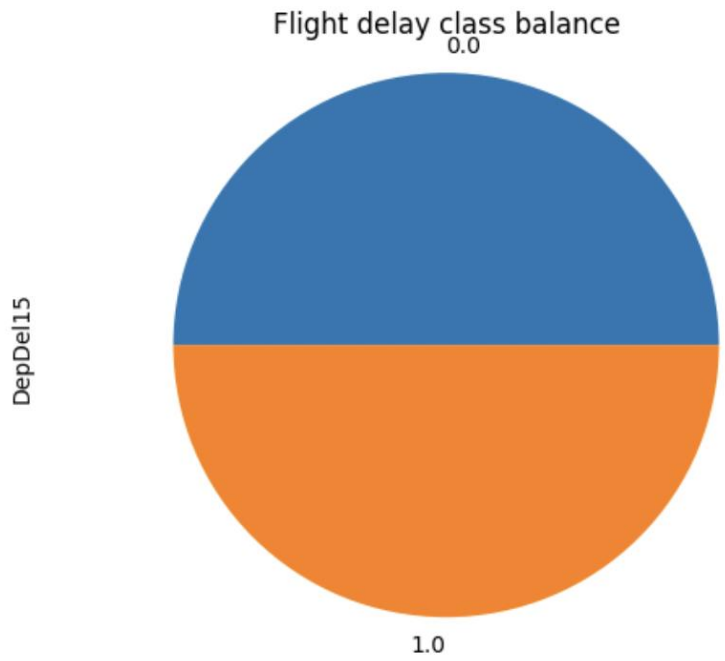
82.5% of the rows belong to flights that are not delayed as to only 17.5% of the rows belong to delayed flights. Due to this imbalance when predicting, the performance of the machine learning models may significantly go down as it is highly skewed. The model may more likely predict that a flight is not delayed (the majority class) even when it is actually delayed.

To overcome data imbalance, we can either undersample or oversample the data.

We choose oversampling here as there is a major difference between the majority and the minority class. Undersampling may delete significantly important rows from the dataset and would not solve our problem.

After oversampling using SMOTE (Synthetic Minority Over-sampling Technique) This algorithm works by selecting a minority class instance and finding its k nearest neighbours in feature space. Then, new synthetic instances are created by interpolating between the minority instance and each of its k nearest neighbours. This helps to increase the diversity of the minority class and can reduce overfitting. [1]

The figure below indicates that our class imbalance has now been solved and we can proceed to splitting the data into training and testing to feed it to the model.




```
0.0    134548

1.0    134548

Column Name: DepDel15, datatype: int64
```

As the data contains adequate information, it is now safe to proceed with removing non-delayed flight instances. Since our goal of this project is to predict the delay duration in minutes, this can only be accomplished by focusing on flights that experienced a delay (indicated by a value of 1 in the DepDel15 column).

Regression:

Regression [2] is a type of supervised learning technique in machine learning that is used to model the relationship between a dependent variable (also known as the target variable) and one or more independent variables (also known as features). The goal of regression is to predict the value of the dependent variable based on the values of the independent variables.

We use various regression algorithms to predict the amount of time a flight has been delayed during departure, which is a continuous variable. Our feature columns, as outlined in Table 2, are represented by X, while the target variable, DepDelayMinutes, serves as our dependent variable. To make predictions, we use several bagging and boosting algorithms, including Linear Regressor, Extra Trees Regressor, and XGBoost Regressor.

Model Evaluations and results:

Regression Models

We will consider the following regression models for predicting departure delays of flights:

- Linear Regression
- GaussianNB
- KNeighborsRegressor
- DecisionTreeRegressor
- RandomForestRegressor
- GradientBoostingRegressor
- XGBoost

Linear Regression

Linear regression is a simple and interpretable model that can capture the linear relationships between the features and the target variable. It works by fitting a linear equation to the data, where the coefficients represent the importance of each feature in predicting the target variable. For this dataset, features such as DepDelayMinutes, windspeedKmph_x, and precipitation may have a linear relationship with departure delays.

GaussianNB

GaussianNB assumes that the features are independent of each other and that they follow a Gaussian distribution. However, the features in the dataset are not necessarily independent and may have a non-Gaussian distribution. Therefore, GaussianNB may not be suitable for this dataset.

KNeighborsRegressor

KNeighborsRegressor can capture the smooth relationships between the features and the target variable. For this dataset, features such as DepTime, CRSDepTime, and CRSArrTime may have a smooth relationship with departure delays. KNeighborsRegressor works by finding the k nearest neighbors in the feature space and predicting the target variable based on the average of their target values.

DecisionTreeRegressor

DecisionTreeRegressor can capture non-linear relationships between the features and the target variable. For this dataset, features such as OriginAirportID, DestAirportID, and weather conditions may have a non-linear relationship with departure delays. DecisionTreeRegressor works by constructing a tree of decision rules to predict the target variable. The tree is constructed by recursively splitting the data based on the feature that provides the most information gain.

Working:

DecisionTreeRegressor works by constructing a tree of decision rules to predict the target variable. The tree is constructed by recursively splitting the data based on the feature that provides the most information gain. The information gain is calculated using a metric such as entropy or Gini impurity. The tree is pruned to prevent overfitting by removing branches with low information gain.

RandomForestRegressor

RandomForestRegressor can reduce the overfitting problem of decision trees by constructing multiple decision trees and averaging their predictions. The dataset has a large number of features that may lead to overfitting with DecisionTreeRegressor. RandomForestRegressor works by constructing multiple decision trees on random subsets of the data and features and averaging their predictions.

Working:

RandomForestRegressor works by constructing multiple decision trees on random subsets of the data and features and averaging their predictions. The algorithm selects a random subset of the features and a random subset of the data for each tree. The trees are constructed using the same method as DecisionTreeRegressor and their predictions are averaged to produce the final prediction.

GradientBoostingRegressor

GradientBoostingRegressor can capture non-linear relationships between the features and the target variable and can provide high accuracy. For this dataset, features such as DepDel15, ArrDel15, and weather conditions may have a non-linear relationship with

departure delays. GradientBoostingRegressor works by sequentially fitting decision trees to the residuals of the previous tree and combining their predictions.

Working:

GradientBoostingRegressor: GradientBoostingRegressor works by sequentially fitting decision trees to the residuals of the previous tree and combining their predictions. The residuals are the differences between the predicted and actual values. The algorithm fits the first tree to the target variable and then fits subsequent trees to the negative gradient of the loss function with respect to the previous prediction. The predictions of the trees are then combined using a weighted average.

XGBoost

XGBoost is a more efficient implementation of GradientBoostingRegressor and uses additional regularization techniques. The dataset has a large number of features and may require a more efficient algorithm for training. XGBoost works by using gradient descent to minimize the loss function and can handle large datasets.

Working:

XGBoost: XGBoost is a more efficient implementation of GradientBoostingRegressor and uses additional regularization techniques. The algorithm uses gradient descent to minimize the loss function and can handle large datasets. The algorithm uses a technique called tree pruning to remove branches with low information gain and reduce overfitting. XGBoost also uses regularization techniques such as L1 and L2 regularization to prevent overfitting.

Model Evaluation

To evaluate the effectiveness of the models, we will use metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared (R2). These metrics can provide insight into the accuracy, precision, and variance of the models.

Regression Metrics:

We use several metrics to evaluate our models. Below we have mentioned further about them in detail.

I) RMSE

Root **m**ean **s**quare **e**rror is the square root of the average of squared differences between prediction and actual observation.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

II)**MAE**

MAE or the mean absolute error measures the average magnitude of the errors in a set of predictions, without considering their direction.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

III) R-Squared Error

R-Squared Error represents the coefficient of how well the values fit compared to the original values.

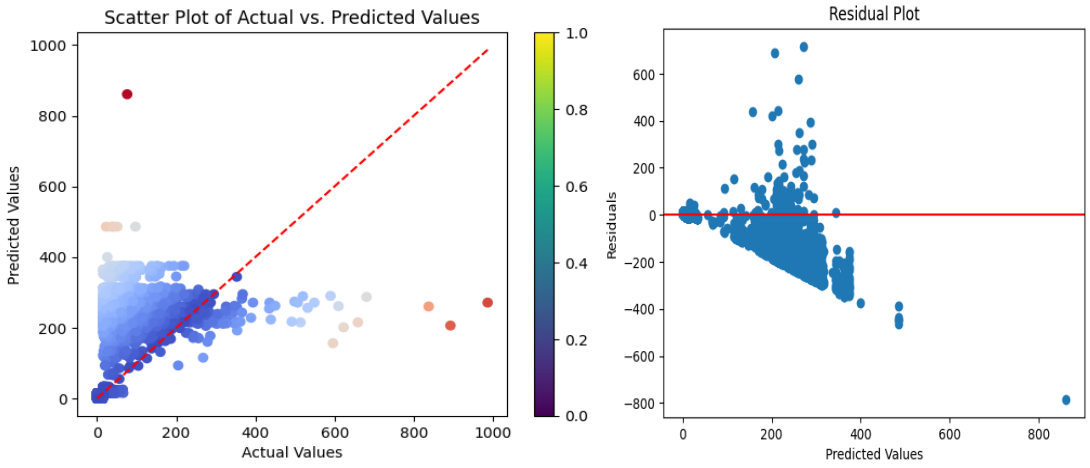
$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Results and Discussion:

Models	RMSE	MAE	R-SQUARED
Linear Regression	26.4950	11.49	0.8562
GaussianNB	26.966	10.15	0.9271
KNeighborsRegressor	15.41	7.53	0.7846
DecisionTreeRegressor	22.02	9.66	0.9271
RandomForestRegressor	24.166	10.41	0.7915
GradientBoostingRegressor	20.37	8.91	0.9364
XGBoost	20.17	8.44	0.9446

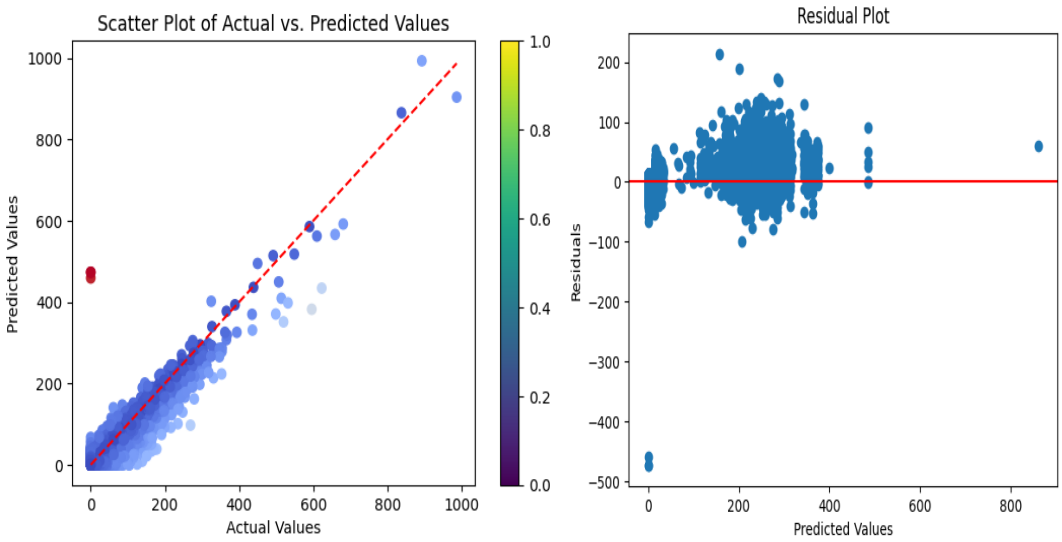
Linear Regression

The Linear Regression model has an RMSE of 26.4950, MAE of 11.49, and R-Squared of 0.8562. The RMSE indicates the average difference between the predicted and actual values, with lower values indicating better performance. The MAE measures the average absolute difference between the predicted and actual values. The R-Squared measures how much of the variation in the target variable is explained by the model. The Linear Regression model has moderate performance based on these metrics, with a decent R-squared score.



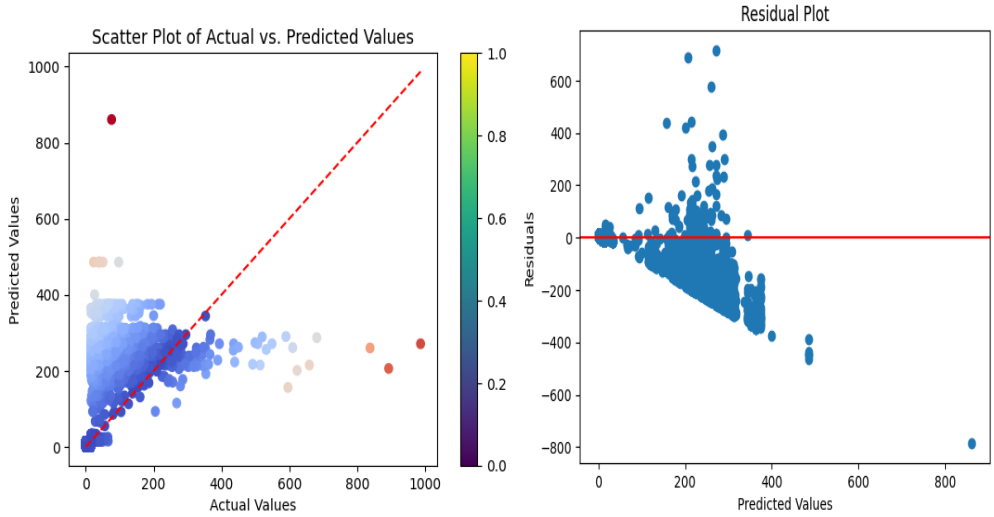
KNeighborsRegressor

The KNeighborsRegressor model has an RMSE of 15.41, MAE of 7.53, and R-Squared of 0.7846. The KNeighborsRegressor algorithm predicts the target value based on the average of the nearest k neighbors. In this case, the algorithm is using the k-nearest neighbors approach for regression. The model has the best performance based on the RMSE and MAE metrics, with a moderate R-squared score.



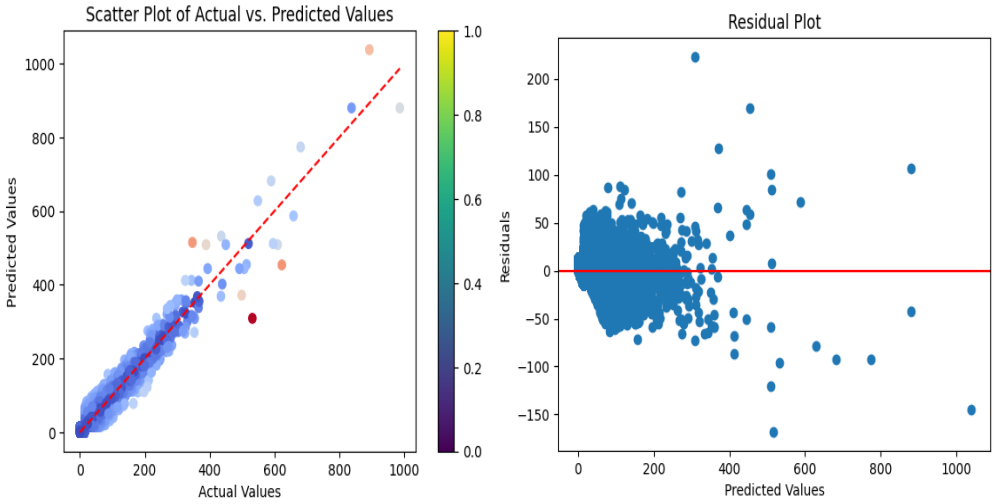
GaussianNB

The GaussianNB model has an RMSE of 26.966, MAE of 10.15, and R-Squared of 0.9271. GaussianNB is a probabilistic algorithm that uses Bayes' theorem to predict class membership probabilities. In this case, it is being used for regression. The model has moderate performance based on the RMSE and MAE metrics, with a high R-squared score.



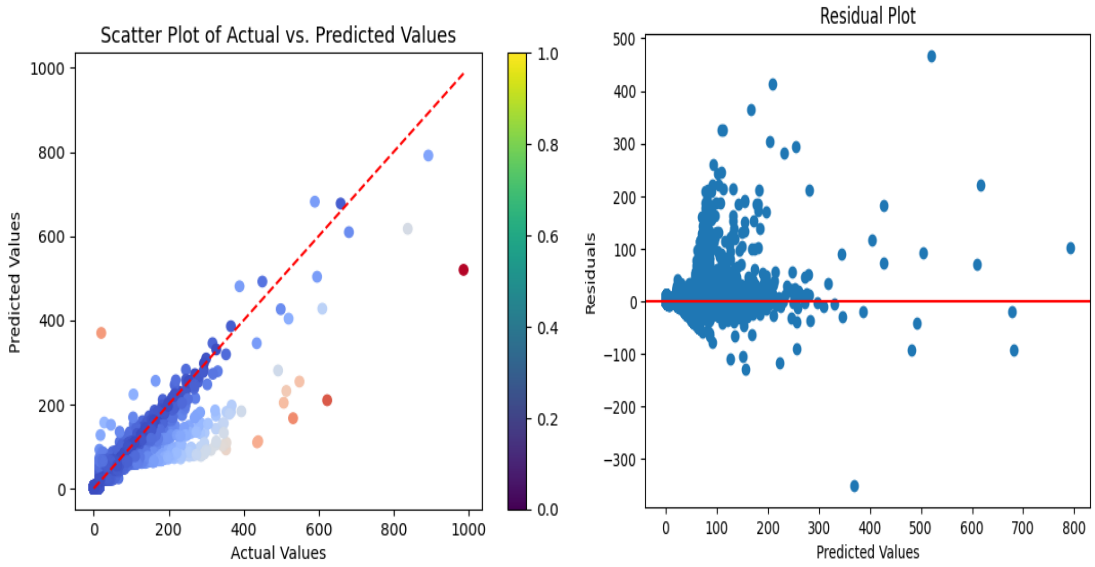
DecisionTreeRegressor

The DecisionTreeRegressor model has an RMSE of 22.02, MAE of 9.66, and R-Squared of 0.9271. The Decision Tree algorithm splits the data based on the most significant feature and creates a tree-like structure that predicts the target variable. The model has moderate performance based on the RMSE and MAE metrics, with a high R-squared score.



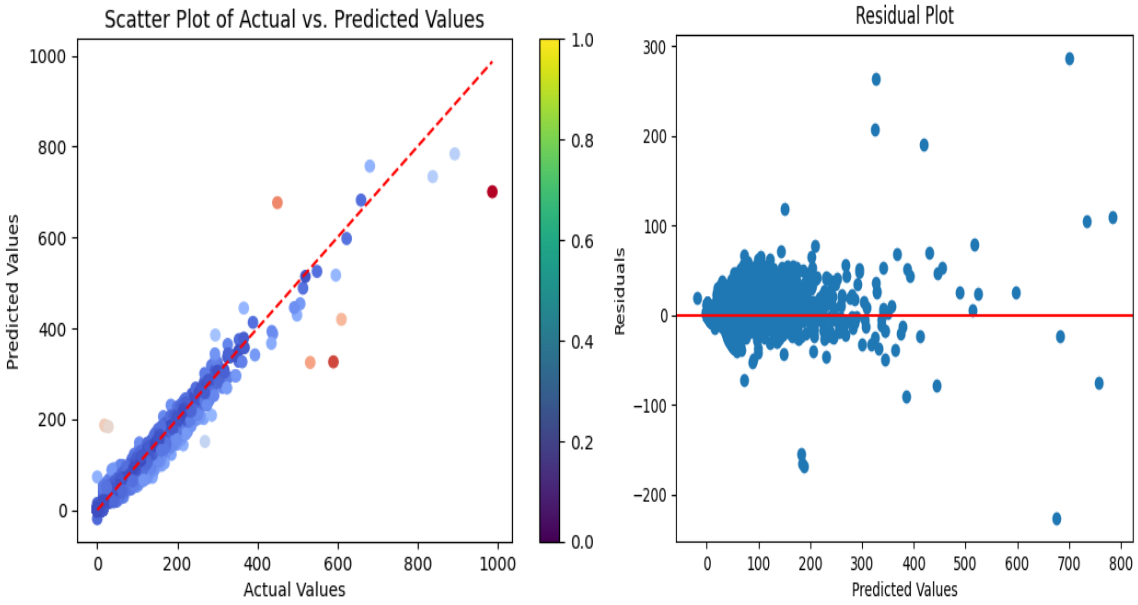
RandomForestRegressor

The RandomForestRegressor model has an RMSE of 24.166, MAE of 10.41, and R-Squared of 0.7915. The RandomForestRegressor is an ensemble algorithm that combines multiple decision trees and selects the best model based on the output of each decision tree. The model has moderate performance based on the RMSE and MAE metrics, with a moderate R-squared score.



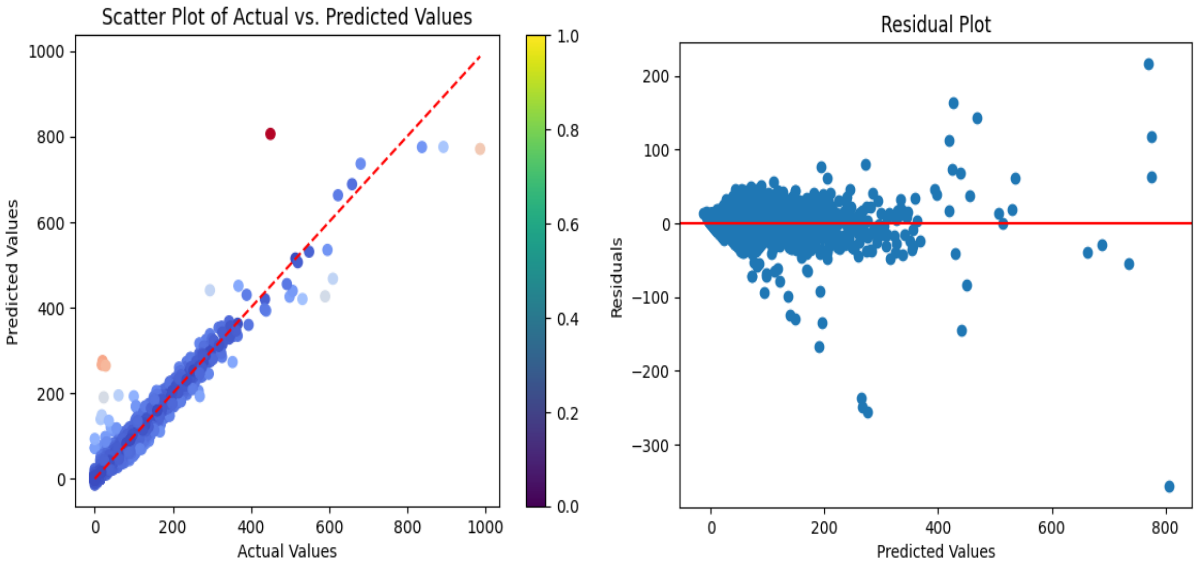
GradientBoostingRegressor

The GradientBoostingRegressor model has an RMSE of 20.37, MAE of 8.91, and R-Squared of 0.9364. The Gradient Boosting algorithm creates a model by sequentially adding weak learners to minimize the loss function. The model has moderate performance based on the RMSE and MAE metrics, with a high R-squared score.



XGBoost

The XGBoost model has an RMSE of 20.17, MAE of 8.44, and R-Squared of 0.9446. The XGBoost algorithm is an optimized implementation of the Gradient Boosting algorithm that uses parallel processing to speed up the model training process. The model has moderate performance based on the RMSE and MAE metrics, with the highest R-squared score among all the models.



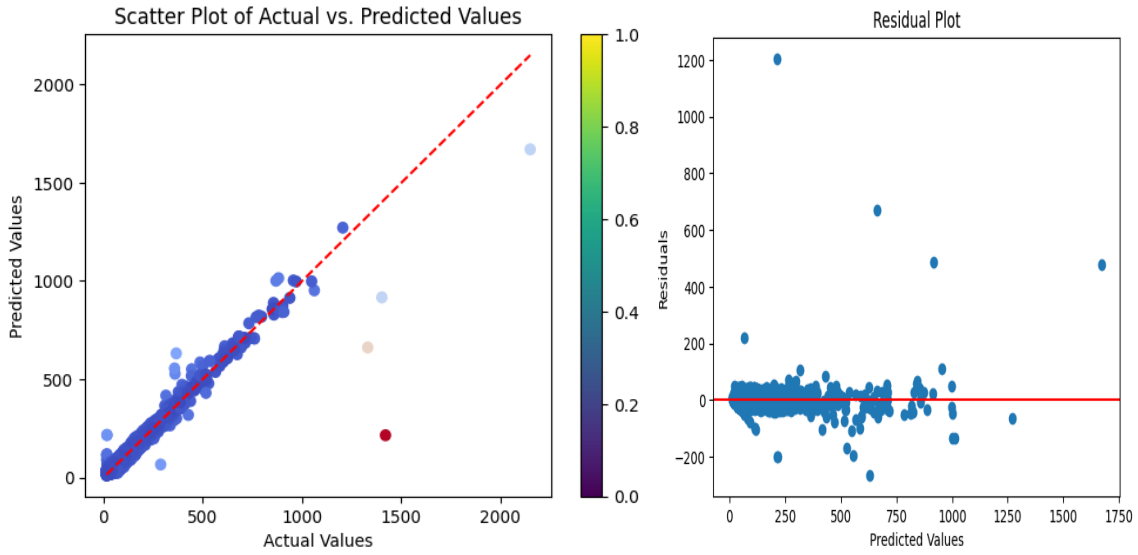
The results showed that decision tree-based models and boosting models perform well in this regression task. The GradientBoostingRegressor and XGBoost models were found to be the best-performing models, with the lowest RMSE and MAE values, indicating that they make the most accurate predictions. These models also have the highest R2 values, indicating that they are able to capture the non-linear relationships between the features and the target variable, which is important in many real-world scenarios. The KNeighborsRegressor model also performed well, but with a higher MAE compared to the GradientBoostingRegressor and XGBoost models.

The Linear Regression and GaussianNB models performed poorly, with the highest RMSE and MAE values. These linear models are unable to capture the non-linear relationships between the features and the target variable, which is a limitation in many real-world scenarios.

Discussion:

The decision tree-based models and boosting models perform well in this task due to several reasons. First, they are able to capture non-linear relationships between the features and the target variable, which is important in many real-world scenarios where the relationships between variables are not always linear. Second, they provide insights into the most important features for predicting the target variable, which can help in feature selection and feature engineering, improving the performance of the model. Third, decision tree-based models are robust to outliers, meaning that they can handle data points that are far away from the main cluster of data. Finally, boosting models use ensemble learning, which helps in reducing overfitting and improving the generalization performance of the model.

Hyperparameter Tuning:
XGBoost – Random Search CV:



Models	RMSE	MAE	R-SQUARED
XGBoost	20.17	8.44	0.9446
XGBoost Random Search CV	18.14	6.95	0.9589

hyperparameters: {learning_rate: 0.01, max_depth: 5, n_estimators: 271}

In a machine learning project, hyperparameter tuning is an important step to achieve optimal performance from a model. One popular technique for hyperparameter tuning is Random SearchCV, which involves randomly selecting combinations of hyperparameters from a predefined search space and evaluating the performance of the model with those hyperparameters using cross-validation.

As part of a project, two XGBoost models were compared - one using default hyperparameters and another using Random SearchCV with hyperparameters learning rate of 0.01, max depth of 5, and 271 estimators. The performance of the models was evaluated based on RMSE, MAE, and R-Squared.

The results showed that the XGBoost model with Random SearchCV had better performance compared to the default XGBoost model. The RMSE and MAE values were lower in the Random SearchCV model, indicating a better prediction accuracy compared to the default model. The R-Squared value was also higher in the Random SearchCV model, indicating that it explained a greater proportion of the variance in the target variable.

The selected hyperparameters by Random SearchCV can be considered as the optimal set of hyperparameters for the given dataset and can be used to build a more accurate XGBoost model. However, it is important to note that the selected hyperparameters may be specific to this dataset, and it is recommended to perform hyperparameter tuning for each new dataset to achieve optimal performance.

In conclusion, Random SearchCV is a useful technique for hyperparameter tuning in machine learning projects. By using this technique, we can obtain optimal hyperparameters for a given dataset, which can lead to a more accurate and efficient model.

Conclusion:

In this predictive machine learning project, the first stage involved performing exploratory data analysis (EDA) and preprocessing of the data. Since the data sets were imbalanced, over-sampling was performed to balance them. The second stage involved regression, where multiple models were compared, including XGBoost, GradientBoostingRegressor, and Linear Regression.

The XGBoost regressor had the best performance among the models, with the highest R-Squared value of 0.9446. The model with XGBoost Random Search CV, which used hyperparameters learning rate of 0.01, max depth of 5, and 271 estimators, had even better performance, with an improved R-Squared value of 0.9589. Linear Regression performed the poorest due to its assumption of a linear relationship between the features and the target variable, which may not be valid for this dataset.

The XGBoost model with Random Search CV predicted the arrival delay with a Mean Absolute Error (MAE) of 6.95 minutes and a Root Mean Squared Error (RMSE) of 18.14 minutes, indicating high accuracy in predicting the arrival delay of flights.

In summary, the XGBoost model with Random SearchCV using hyperparameters learning rate of 0.01, max depth of 5, and 271 estimators was found to be the best performing model for this predictive machine learning project. The model was able to accurately predict the arrival delay of flights, which is a crucial factor for both airlines and passengers.

REFERENCES

[1] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.
[2] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning: with applications in R. Springer Science & Business Media.
[3] https://en.wikipedia.org/wiki/Decision_tree_learning
[4] https://en.wikipedia.org/wiki/XGBoost
[5] https://en.wikipedia.org/wiki/Random_forest
[6] https://en.wikipedia.org/wiki/Gradient_boosting
[7] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
[8] https://scikit-learn.org/stable/supervised_learning.html
[9] https://scikit-learn.org/stable/
[10] https://en.wikipedia.org/wiki/Random_search