# **PROJECT REPORT**

# **Topic**: Air Traffic Delay Predictions

# 

# **Author:** Harshada Kailas Bagal, Sahithi Muppavaram

# 

# **Appendix**

Introduction……………………………………………………………………………………3

Description of Dataset………………………………………………….……………………...3

Data Source…………………………………………………………….……………………...5

Methodology…………………………………………………………...……………………...5

Data cleaning and Pre-processing……………………………………...……………………...5

Feature Engineering…………………………………………………………………………...7

Exploratory Data Analysis…………………………………………….………………………9

Modelling strategies………………………………………………………………………….14

Result and analysis……………………………………………………………………………22

Conclusion…………………………………………………………….……………………...23

References…………………………………………………………….……………………...24

# **Air Traffic Delay Predictions**

# **Introduction**

Modern transportation relies heavily on-air travel to link people and commerce worldwide. On the other hand, flight cancellations and delays can cause major problems for travelers and airlines, interfering with schedules and creating delays. Understanding the various factors impacting these delays is crucial for customers wanting a smooth trip and airlines looking to improve their operational efficacy. With recent advancements, Machine learning techniques can analyze past flight data to find trends and patterns linked to delays. This analysis, which makes use of data-driven and statistical approaches, will evaluate the performance of the airlines, and keep passengers well informed about the flight schedule. In addition to fulfilling passengers' expectations for a smooth travel experience, this also advances the industry's overall objective of continually enhancing customer satisfaction and operational efficiency.

The primary objective of this project is to provide travelers with well-informed choices and to provide airlines with important information to improve customer happiness and operational efficiency. Our aim is to develop a predictive system that can predict flight delays by using machine learning models. In order to help passengers and airlines navigate and enhance the flying experience, our model attempts to not only assess the probability of delays but also estimate the expected duration of any potential delays.

Our project's main goal is to identify patterns and trends related to aircraft delays and cancellations in the United States by analyzing a large dataset on these events. Our objective is to offer significant insights that enable people to make more informed decisions about their travel plans by exploring the fundamental causes of airline delays. In particular, our goal is to create a predictive model that can anticipate the probability of a flight delay and, in the event that it occurs, the estimated length of the delay after takeoff.

# **Description of Dataset**

The dataset comprises three distinct files containing data of flights, airlines, and airports, each providing valuable information in the context of air traffic delay predictions:

1. Flight Dataset

* The flight data gives certain operational details about every flight.
* It contains important information such as flight numbers, which are unique identifiers, and timing details like scheduled and actual departure timings, delays in departure
* It also records important flight milestones such as the taxiing time (Taxi\_out), takeoff (Wheels\_off), and landing (Wheels\_on) timings, as well as the airports involved (Origin airport, Destination airport).

2. Airport Dataset

* Airport data is primarily used to identify an airport using its IATA Code.
* It also provides geographic information about each airport, such as its name (Airport), city, state or area, and nation.
* This makes it easier to trace aircraft paths and conduct operational analyses at various geographic scales.
* The addition of latitude (and possibly longitude, although it is not stated) enables the execution of geographical analyses.

3. Airlines Dataset

* The airline’s data set enables the study of flight data by carrier by giving each airline a unique identity (IATA\_CODE) and name.
* Stakeholders can analyze competitiveness, network tactics, and the general state of the aviation business by using this dataset, which is essential for more comprehensive industry analysis.

The files are as follows:-

1. Flight.csv – This file contains flights information. The dataset contains 31 columns and around 57,000,00 entries
2. Flight number: Unique identifier for a specific flight.
3. Year: The calendar year in which the flight occurred.
4. Month: The specific month within the year when the flight took place.
5. Day: The day of the month when the flight departed or arrived.
6. Day of week: The day of the week on which the flight operated
7. Airline: The name or code of the airline carrier operating the flight.
8. Flight number: The unique identifier assigned to the flight by the airline carrier.
9. Tail number: The registration number or unique identifier assigned to the aircraft operating the flight.
10. Origin airport: The airport where the flight departs from.
11. Destination airport: The airport where the flight arrives.
12. Scheduled Departure: Planned departure time of the flight.
13. Departure Time: Actual departure time of the flight.
14. Departure Delay: Delay in departure time compared to the scheduled departure time.
15. Taxi\_out: Time spent taxiing out from the departure gate.
16. Wheels\_on: Time when the aircraft's wheels touch the runway upon arrival.
17. Wheels\_off: Time when the aircraft's wheels leave the ground during takeoff.
18. Air\_Time: Duration of the flight in the air.
19. Airport.csv – This file contains information about the airports. The dataset contains

7 columns and 322 entries.

1. IATA Code: International Air Transport Association code assigned to the airport.
2. Airport: Name of the airport.
3. City: City where the airport is located.
4. State: State or region where the airport is situated.
5. Country: Country where the airport is located.
6. Latitude: Geographic coordinate indicating the north-south position
7. Longitude: Geographic coordinate indicating the east-west position
8. Airline.csv – This file contains information about the airlines and the associated codes
9. IATA\_CODE: International Air Transport Association code assigned to the airline.
10. Airline: Name of the airline.

# **Data Source**

<https://www.kaggle.com/datasets/usdot/flight-delays>

# **Methodology**

To predict aircraft delays, we need to create certain rules and look at patterns. Clearly coding every one of these guidelines and actions will become quite time-consuming. To do this, we'll employ machine learning techniques. In order to examine the USA flight dataset, we will automate the learning process. We will have the ability to forecast whether a flight will be delayed or not based on the conditions and contributing factors. Since flight delays can be influenced by various factors, such as departure time, airline, airport, and weather conditions, we'll extract relevant features from the data to understand patterns and trends. These features will include:

Flight Time Features: Departure time, arrival time, and duration of the flight.

Airline Features: Airline carrier operating the flight.

Airport Features: Origin and destination airports.

Weather Features: Weather conditions at departure and arrival airports.

We'll predict the probability of a flight being delayed based on the extracted features. If the probability of delay exceeds a certain threshold (e.g., 15), we'll classify the flight as delayed; otherwise, it will be classified as on time. If using regression models, we will predict by what time our flight is going to get delayed

Our methodology for building the flight delay prediction model will involve the following steps:

1. Data cleaning and pre-processing
2. Feature engineering
3. Exploratory Data Analysis
4. ML algorithms

# **Data cleaning & Pre-processing**

We will start by cleaning and processing the data to make it suitable for addressing the current issue now that we have a dataset with an abundant amount of information on flight delays in the aviation industry. We will look for and confirm the following anomalies in the data

# **a. Finding Missing Values**

Finding missing values in the dataset is the first step in the procedure. In order to do this, we examined for any null or NaN (Not a Number) values that can affect our dataset's quality and, in turn, the prediction model's accuracy.

A screenshot of a computer

Description automatically generated

**b. Filling Missing Values**

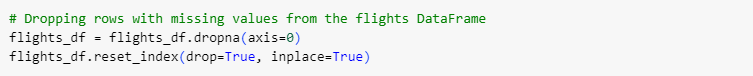
Zeros are used to fill in missing values for delay-related columns including "WEATHER\_DELAY," "AIRLINE\_DELAY," "AIR\_SYSTEM\_DELAY," "SECURITY\_DELAY," "LATE\_AIRCRAFT\_DELAY," and "SECURITY\_DELAY." This phase assumes that there was no delay caused by these causes if there are no recorded delays in these columns.

A screenshot of a computer

Description automatically generated

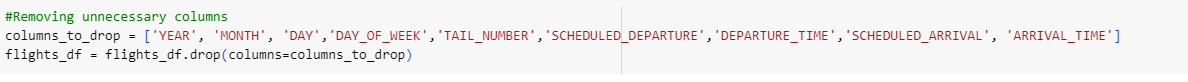
**c. Dropping Rows with Missing Values**

We started this procedure by removing rows that still have missing values after addressing particular columns. In order to improve the model’s capacity for precise prediction-making, it is imperative that it be trained on complete records.



**d. Eliminating Unnecessary Columns**

A few columns are found to be redundant for the predictive model and are eliminated from the dataset. 'Year', 'Month', 'Day', 'Day\_of\_week', 'Tail\_Number',' Scheduled\_Departure', 'Departure\_Time,’ ‘Scheduled\_Arrival', and 'Arrival\_Time' are some of these columns. The model can be better focused on the most important characteristics for forecasting flight delays by removing these columns.



# **Feature Engineering**

During feature engineering, our objective was to generate several new features that enhance our comprehension of the data in a more simplified and precise manner. In flight delay analysis, feature engineering plays a critical role in extracting relevant information from the dataset to improve the predictive performance of machine learning models. Here are some common techniques used in feature engineering for flight delay analysis:

**The steps for Feature Creation are as follows:**

1. Creating Date column as there is year, month, and day
2. Merged all the datasets i.e flights, airlines, and airports
3. Mapping cancellation reasons to numerical values dataset
4. Converting ‘Wheels\_On’ and ‘Wheels\_Off’ values to minutes
5. Converting ‘Scheduled\_Departure’ column values to standard date time format
6. Converting ‘Departure\_Time’ column values to standard date time format
7. Converting ‘Scheduled\_Arrival’ column values to standard date time format
8. Converting ‘Arrival\_Time’ column values to standard date time format

a. Time Conversion for 'SCHEDULED\_DEPARTURE' and Related Time Variables:

The project starts with the conversion of flight times, which are initially recorded in a HHMM format as floats, into a more usable datetime format. This involves a custom function that parses these floats, accommodating special cases like '2400' (representing midnight) by adjusting it to '0000', thereby preventing format errors. This conversion facilitates easier manipulation and analysis of time data, allowing the model to better understand patterns related to specific times of the day that might influence flight delays.

A screenshot of a computer code

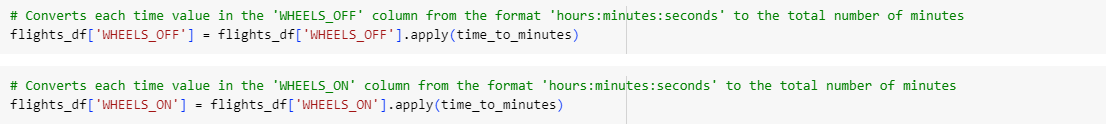
Description automatically generated

b. Converting Wheels\_On and Wheels\_Off values to minutes:

Converting "wheels\_on" and "wheels\_off" values to minutes can enhance the usability, efficiency, and interpretability of the flight dataset for various analytical tasks. Time operations in numerical format are often faster and more efficient than operations involving string or timestamp formats.

A screenshot of a computer

Description automatically generated



A white table with numbers and letters

Description automatically generated

In order for the prediction algorithms to identify patterns across various times of day, days of the week, and months—all of which have a substantial impact on flight delays—this combined datetime information is essential for modeling.

c. Handling Missing Values with Domain-Specific Knowledge:

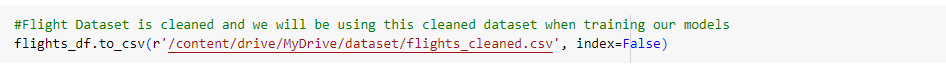
In relation to flight delay columns (such as 'AIRLINE\_DELAY' and 'SECURITY\_DELAY'), we substituted zeros for any missing values, presuming that the absence of a recorded value signifies no delay resulting from that reason. This method reflects domain-specific knowledge, in which missing data is not indicative of a delay, but rather an instructive lack of information. We found that the latter step is essential for preserving the dataset's integrity and making sure the model doesn't mistakenly interpret the lack of delay records as missing data, which could distort the predicted accuracy.

d. Feature Selection and Removal of Redundant Variables:

The variables that are ineffective for forecasting flight delays were found and eliminated. This comprises identifiers and time variables that become superfluous after being transformed into more useful characteristics. For instance, once their data is included in the newly formed datetime features, raw time variables are eliminated. By removing noise from the dataset, we were able to narrow down the model to the most important characteristics, which helped to reduce the problem's dimensionality and maybe improve the model's performance.

e. Exporting the Engineered Dataset for Modeling:

We cleaned and feature-engineered it, then exported it as a new file so that this curated dataset would be used in all future modeling endeavors. This is an important step in the process of moving from data preparation to modeling since it is at this point that the emphasis moves from data manipulation to the selection, training, and assessment of algorithms.



Now that our data is devoid of errors, we can proceed with preprocessing to create a functional dataset suitable for effectively training machine learning models. Furthermore, performing data analysis is pivotal for extracting valuable insights. Hence, we will merge multiple files and extract relevant fields according to our specific requirements.

1. Combine the 'flights\_df' DataFrame with the 'airlines\_df' DataFrame using the 'AIRLINE' column in 'flights\_df' and the 'IATA\_CODE' column in 'airlines\_df'. Retain only the matching records ('inner' join) to create an integrated dataset.
2. Eliminate the 'AIRLINE\_x' and 'IATA\_CODE' columns from the 'flights\_df' DataFrame.
3. Rename the 'AIRLINE\_y' column to 'AIRLINE' in the 'flights\_df' DataFrame.
4. Merge the 'flights\_df' DataFrame with the 'airports\_df' DataFrame based on the 'ORIGIN\_AIRPORT' column in 'flights\_df' and the 'IATA\_CODE' column in 'airports\_df', retaining only matching records ('inner' join).
5. Merge the 'flights\_df' DataFrame with the 'airports\_df' DataFrame based on the 'DESTINATION\_AIRPORT' column in 'flights\_df' and the 'IATA\_CODE' column in 'airports\_df', retaining only matching records ('inner' join).

# **Exploratory Data Analysis**

EDA is a crucial step in our project because it emphasizes how crucial it is to take time-based features—like months—out of flight data and analyze them. This study offers important insights that guide the feature engineering and predictive modeling stages of the project by analyzing how flight volumes fluctuate over the course of the year and spotting possible seasonal patterns. In order to create more precise predictive models, it is necessary to identify the intricate dynamics underlying flight delays using variables and visualization approaches.

We started the EDA process by obtaining the current month from the dataset’s date column. This is done because one important time-based factor that affects flight delays is the month. Dataset has a ‘DATE’ column, from which the ‘MONTH’ is extracted and added to a new column. This stage is crucial for examining the variations in flight delays between months, which may be brought on by seasonal travel trends or meteorological circumstances.

We found Seasonal trends in flight volumes in the depicted data, with some months perhaps exhibiting spikes because of holidays or vacation times. Understanding when flight delays are most likely to happen and organizing additional research on the reasons behind these delays depend on the results of our project.

Theories concerning the probable reasons for delays during these times can be developed by looking at the months with high flight volumes. For instance, inclement weather can cause more delays throughout the winter, while summertime travel spikes might cause delays during the summer.

We visualized the number of flights every month coming next after extracting the month. To do this, count the number of flights in each month by grouping the data by the just created 'MONTH' column. The resulting graphic offers a clear picture of the annual changes in flight volumes. The key variables in this study are 'flights\_df', which is a dataframe representing flight data, and 'flight\_counts', which is a collection of monthly flight counts that have been aggregated. This process assists in identifying months with greater flight volumes that may be more prone to delays because of higher air traffic and operational difficulties.

**Q1. Which months have the highest and lowest count of flights in 2015?**

A graph with a line

Description automatically generated

Here we can see that February, September and November and December have the lowest count of flights in 2015.

**Q2. Which days of the month have the highest and lowest daily count of flights?**

A graph showing the number of flights

Description automatically generated

We can see here that the 31st Day of the month has the lowest count of flights as not every month has 31st day.

**Q3. Is there any difference of flight count between daily and monthly counts for each month?**

A graph of a tall tower

Description automatically generated with medium confidence

Here we are analyzing data monthly based on weekdays for each month. All in all , we can see that there is no big difference between the showcased data

**Q4. Which origin city had the highest count of flights in 2015?**

A graph of different states

Description automatically generated

Here we can see that Chicago has the highest number of flights in the year 2015. Atlanta being the second highest Origin city

**Q5. Which airlines has the highest share of flights in 2015?**

A pie chart with different colored circles

Description automatically generated

Delta airlines had the highest amount of flights in 2015 with a share of 22% and Spirit aiines being the second highest.

**Q6. Which airline has the highest arrival delay in 2015?**

A screenshot of a graph

Description automatically generated

American Airline has the highest arrival delay.

**Q7. Which airline has the arrival delay greater than 15mins, on time and large delay?**

A graph of flight number

Description automatically generated with medium confidence

**Q8. Which airline have delayed and not delayed according to the counts of airline?**

A graph of different colored bars

Description automatically generated

Atlantic Southeast Airlines and American Airlines are the most delayed flights

# **Modelling strategies**

Here are some significant points that will influence the strategy for designing and training machine learning models:

1. Linear regression
2. Random Forest Regression
3. XGBoost Model
4. KNN regression

**Linear Regression**

In order to forecast the continuous variable ARRIVAL\_DELAY, we choose to use linear regression. This approach works well for comprehending how several independent factors relate to a continuous dependent variable. Because of its clarity and interpretability, it's a fantastic place to start when dealing with regression issues.

**a. Feature Selection and Preprocessing:** We have chosen several features, such as taxi times, distances, and different kinds of delays (such as weather delays, airline delays, etc.), which suggests a thorough method to record the variables influencing delays. Our feature variable is X and target variable is Y i.e Arrival Delay

A screen shot of a computer

Description automatically generated

**b. Model Training:** Using a subset of the data, we fitted the Linear Regression model to learn how each chosen feature affects the arrival delay.

A close-up of a text

Description automatically generated

**c. Assessment:** To assess the model's performance and comprehend the variance, R2 and Adjusted R2 were used.

A computer screen shot of text

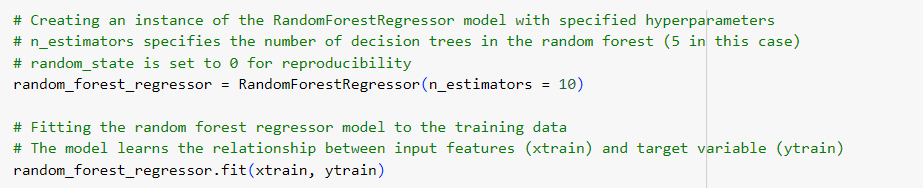
Description automatically generated

A screenshot of a computer

Description automatically generated

# **Random Forest Regression Model**

Rather than using a straightforward linear model, the Random Forest model—an ensemble of decision trees—was selected to handle the non-linear correlations and interactions between features. It's especially helpful for intricate datasets where several features influence the result.  
  
**a. Training:** The Random Forest Regressor, which is less prone to overfit than a single decision tree and can automatically handle categorical data, was trained after preprocessing.



**b. Prediction and Evaluation:** R2, MAE, MSE, and RMSE were used to evaluate the model's predictions. Higher-quality measures that capture more intricate patterns in the data than Linear Regression would make its application more justified.

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

# **XGBoost Model**

XGBoost is a strong, quick, and effective gradient boosting system. It is employed because of its effectiveness in handling sparse data and high performance. The selection of XGBoost may have been made with the intention to lessen overfitting.

**a. Training:** To control and enhance the process, the model's complexity was trained using parameters such as Colsample\_bytree, learning\_rate, and max\_depth are examples of hyperparameters that we have chosen with the intention of striking a balance between overfitting and model correctness.

A screen shot of a computer

Description automatically generated

**b. Prediction and Evaluation:** The model's predictive strength and capacity to generalize to previously untested data were demonstrated by applying evaluation metrics akin to those employed for Random Forest.

A computer code with text

Description automatically generated with medium confidence

A screenshot of a computer screen

Description automatically generated

# **KNN Regression Model**

KNN, or K-Nearest Neighbors, is a simple yet effective algorithm used for classification and regression tasks. It works by finding the K nearest data points to a given input point and making predictions based on the majority class or average value of those neighbors.

**a. Training:** In the training phase, the model learns by storing all the training data points in memory. No explicit training process is involved in KNN, as it is a lazy learner. However, hyperparameters like the number of neighbors (K) can be used to tune the data.

A screenshot of a computer program

Description automatically generated

**b. Prediction and Evaluation:** After the model is trained, predictions are made by finding the K nearest neighbors to the input data point and determining the majority class or average value among them. Evaluation metrics such as R2 score, MAE, MSE, RMSE can be used to assess the model's performance on a test dataset, like other regression algorithms.

Just like XGBoost, the selection of KNN might be driven by its simplicity, ease of implementation, and effectiveness in handling certain types of datasets. However, it's important to consider its limitations, such as high computational cost during prediction for large datasets and sensitivity to the choice of distance metric and value of K.

A screen shot of a computer

Description automatically generated

**Reasons for Model Selection and Tuning:** From setting a baseline (Linear Regression) to capturing intricate data correlations (Random Forest, XGBoost), each model has a specific function. Hyperparameter tuning is used to optimize for the best possible predictions. By comparing simpler and more sophisticated models, it is possible to determine which model is best suited for the features of our dataset. This diversity of models guarantees a robust approach to flight delay prediction. Hyperparameter tuning adjusts important parameters in the models in accordance with validation scores to maximize performance.

# **Hyperparameter Tuning**

The goal of hyperparameter tuning was to maximize the model parameters for the Random Forest and XGBoost. To get the greatest outcomes and enhance model performance, this procedure is essential. The process of fine-tuning a model's hyperparameters is essential to maximizing its performance. The max\_depth, min\_samples\_leaf, and other parameters have a big impact on how successfully the XGBoost generalizes to new data. Hyperparameter tuning helps strike a balance between bias and variance in the model. By fine-tuning the hyperparameters, we can adjust the model's complexity to minimize both bias (underfitting) and variance (overfitting), thereby achieving better generalization performance. Hyperparameter tuning can have a significant impact on the model's performance and generalization ability. By systematically searching for the optimal hyperparameters, we can improve the model's accuracy and robustness, leading to more reliable predictions of flight delays.

GridSearchCV for Comprehensive Lookup: In order to ascertain which tune provides the optimum performance, this method methodically cycles through several combinations of parameter tunes, cross-validating along the way. We used it to change a variety of settings, including min\_samples\_leaf, which specifies the bare minimum of samples needed to be at a leaf node, and max\_depth, which is the maximum depth of the tree.  
Relevance of min\_samples\_leaf and max\_depth: max\_depth determines the tree's maximum depth. Although deeper trees have a higher chance of overfitting, they can also catch more intricate patterns. Finding the ideal balance is aided by adjusting this parameter.

Each leaf node, or the decision node at the base of a tree, must have a minimal number of samples, which is ensured by min\_samples\_leaf. By imposing this restriction, the model is unable to learn too specialized patterns from the training set, which could lead to poor generalization.

# **Random Forest Regression Model Tuning**

We focused on the number of trees, maximum depth, and minimum samples per leaf and split while using GridSearchCV to search over a specified parameter grid to discover the optimal combination of parameters for the Random Forest model.

Number of Trees (n\_estimators): This parameter determines the number of decision trees to be used in the random forest ensemble. Increasing the number of trees can improve the model's performance up to a certain point, as it allows the model to capture more complex patterns in the data. However, beyond a certain threshold, adding more trees may lead to diminishing returns or overfitting.

Maximum Depth (max\_depth): It specifies the maximum depth of each decision tree in the forest. Deeper trees can potentially capture more intricate relationships in the data, but they also increase the risk of overfitting. Setting an appropriate maximum depth helps control the complexity of individual trees and the overall ensemble.

Minimum Samples per Leaf (min\_samples\_leaf): This parameter determines the minimum number of samples required to be at a leaf node. Setting a higher value for this parameter can prevent the model from creating nodes with very few samples, which might lead to overfitting. It encourages the model to generalize better by ensuring that each leaf contains a minimum amount of information.

Minimum Samples per Split (min\_samples\_split): It specifies the minimum number of samples required to split an internal node. Similar to min\_samples\_leaf, setting a higher value for this parameter can help prevent the model from splitting nodes that have insufficient data, thereby controlling overfitting.

By systematically exploring different combinations of these parameters using GridSearchCV, we can efficiently search over a specified parameter grid to find the optimal combination that maximizes the model's performance metrics, such as accuracy, precision, recall, or F1-score, depending on the task at hand. This process helps in fine-tuning the Random Forest model and improving its predictive capabilities while avoiding overfitting

A screenshot of a computer code

Description automatically generated

A computer screen shot of a computer screen

Description automatically generated

# **XGBoost Model Tuning**

In a similar vein, GridSearchCV was employed to optimize the model configuration for the XGBoost model by fine-tuning parameters like learning rate, max depth, and n\_estimators, alpha.

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

# **Result and Analysis**

We engaged in feature engineering to extract crucial insights from the provided dataset. Subsequently, we utilized this enriched dataset to train our machine learning models, incorporating hyperparameter tuning to enhance predictive performance. Notably, we applied weight adjustments to the minority class during hyperparameter tuning to address class imbalance.

Starting with a baseline model (Linear Regression) and progressing to more sophisticated algorithms (Random Forest, XGBoost), each model serves a distinct purpose in capturing intricate data correlations. Through hyperparameter tuning, we aimed to optimize predictions by adjusting key parameters based on validation scores. By comparing the performance of simpler and more complex models, we sought to identify the most suitable model for our dataset's specific features. This diversified approach ensures a robust strategy for flight delay prediction. Hyperparameter tuning involves fine-tuning essential model parameters to maximize performance, guided by validation scores. We meticulously compared the performance metrics of all models and scrutinized their significance. Following a thorough examination of these models, we determined 'Random Forest Regression' as the most effective model for our use case, as evidenced by its accuracy score of 0.95 and the lowest RMSE among all models.

|  |  |  |
| --- | --- | --- |
| **ML model** | **R2** | **RMSE** |
| Linear Regression | 0.94 | 9.58 |
| Random Forest Regression | 0.95 | 8.65 |
| XGBoost Model | 0.79 | 18.05 |
| KNN Regression | 0.91 | 11.32 |
| Random Forest Hyperparameter Tuning | 0.92 | 10.55 |
| XGBoost Hyperparameter Tuning | 0.93 | 10.62 |

**A graph of different regression algorithms

Description automatically generated**

# **Conclusion**

We evaluated multiple machine learning models to ensure robustness and reliability in our predictions.

Our Linear Regression model, often the first model to consider for its simplicity and interpretability, performed admirably, obtaining an R2 score of 0.94. This score implies that the model was able to explain 94% of the variability in the delay data, which is impressive for such a straightforward approach. However, its RMSE of 9.58 suggests there was still a significant average discrepancy between the predicted and actual delay times.

We then moved to more complex models, such as the KNN Regression, which produced an R2 of 0.91. While this model was quite effective, capturing a good deal of variance in the data, its RMSE was higher at 11.32, indicating less accuracy in the actual delay predictions compared to the Linear Regression model.

The initial XGBoost Model, prior to tuning, yielded the lowest R2 score of 0.79 amongst all models we tested. This suggested that almost a fifth of the variability in our delay data couldn't be accounted for by the model. Furthermore, its RMSE of 18.05 was considerably higher than the other models.

Seeking improvement, we applied hyperparameter tuning to our Random Forest and XGBoost models. Hyperparameter tuning is an optimization technique intended to configure the model settings to achieve better performance. Post-tuning, the XGBoost model's R2 improved to 0.93 and the Random Forest models to 0.92. This was a significant improvement for the XGBoost model, but both tuned models had higher RMSEs compared to their untuned counterparts, indicating that while they fit the training data better, they were possibly overfitting and not predicting as accurately on unseen data.

Our top performer was the standard Random Forest Regression model. With an R2 score of 0.95, it not only had the highest score, signifying its excellent capability to explain the variance in our dataset, but it also had a relatively low RMSE of 8.65. This balance indicates that the Random Forest model was not only good at understanding the structure of the training data but was also exceptional at accurately predicting delays on new, unseen data, avoiding the pitfalls of overfitting that often come with complex models and hyperparameter tuning.

Thus, we concluded that the standard Random Forest Regression model was the best model for our project on predicting air traffic delays. Its superior R2 score demonstrated its ability to capture the data's variance and its lower RMSE pointed to its accuracy, making it the most reliable model among all those we evaluated.

# **References**

[1] <https://www.kaggle.com/code/sidduboppana/starter-2015-flight-delays-and-45a4bcdc-f>

[2] <https://www.kaggle.com/code/fiftythirtyfour/usa-airline-delays-in-2015>

[3] <https://www.kaggle.com/code/silvia46/flight-cancellation-prediction-and-eda>

[4] <https://www.kaggle.com/code/abhishek211119/2015-flight-delays-and-cancellation-prediction/notebook>

[5] <https://github.com/JaHerbas/Predicting_Flight_Delays>

[6] <https://github.com/tomeryosef/Data-Science-FLIGHT-DELAY-PREDICTION-HIT>

[7] <https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#Algorithm>.

[8]<https://github.com/pranaykmr/FlightDelayPrediction/blob/master/Flight_Delay_Prediction_Final.ipynb>