# December Flight Delay Prediction

**Anonymous CogSci submission**

## Abstract

This project aims to develop a predictive model for flight delays utilizing a dataset spanning from 2018 to 2023. Leveraging various pre-processing techniques, we explore methods to enhance the quality and relevance of the data. Subsequently, we apply four distinct machine learning algorithms to analyze the processed datasets. Our objective is to assess the efficacy of each method in accurately predicting flight delays, thereby contributing to the advancement of predictive analytics in the aviation industry. Through comprehensive experimentation and analysis, we seek to identify the most effective approach for predicting flight delays, offering valuable insights for stakeholders to optimize operational efficiency and enhance passenger experience.

**Keywords:** Flight delay prediction; Data pre-processing; Machine learning

## 1. Introduction

The modern era of air travel is marked by both convenience and complexity, where millions of passengers rely on airlines to reach their destinations safely and on time. However, the reality often falls short of this expectation, as flight delays continue to plague the industry, causing frustration for passengers and operational headaches for airlines.

The landscape of air travel is characterized by its dynamic and interconnected nature, where a multitude of factors converge to determine the punctuality of flights. Within in this complex environment, December emerges as a particularly critical period, marked by increased travel demand driven by holidays, vacations, and family reunions. Against this backdrop, the challenge of predicting flight delays assumes heightened significance, as any disruptions during this time can have far-reaching consequences for passengers, airlines, and airport operations. Figure 1 and 2, sourced from the Bureau of Transportation Statistics, illustrates the flight delays nationally for December 2023 and December 2022 respectively, showcasing a notable improvement compared to the previous year. In December 2022, only 68.72% of flights were on time, reflecting a challenging period with a significant portion of flights experiencing delays. The top three reasons for delays in 2022 were identified as air carrier delay, air craft arriving late, and national aviation system delays. However, the following year, December 2023, exhibited marked progress, with 83.86% of flights that were on time. Despite the improvement seen in December 2023, the 16% flight delays still remains significant. This percentage translates into a substantial number of delayed flights given the high volume of air traffic during the holiday season.

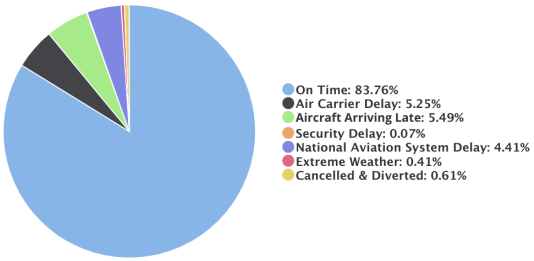


Figure 1: Flight Delays by Cause National (December, 2023).

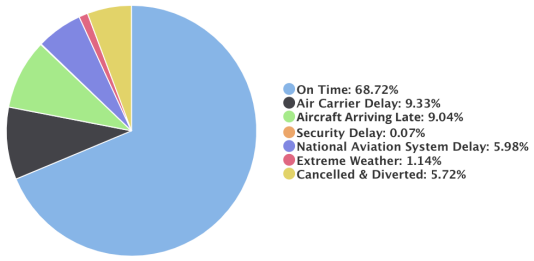


Figure 2: Flight Delays by Cause National (December, 2022).

Our project aims to tackle this pressing issue by developing a system capable of accurately predicting flight delays specifically for the December time frame, using the dataset spanning from 2018 to 2023. By leveraging a rich dataset provided by the Bureau of Transportation Statistics(BTS), encompassing over 4 million flight records, we seek to shed light on the dynamics of flight delays over this critical period.

### 1.1 Impact

The significance of this problem cannot be overstated. Flight delays not only inconvenience passengers but also incur substantial costs for airlines and impact the broader economy. By forecasting delays, travelers can better plan their journeys, reducing stress and potential financial losses. Airlines stand to benefit from improved operational efficiency, as they can adjust staffing, gate assignments, and schedules to minimize disruptions. Enhanced customer experiences result from proactive communication about delays, fostering loyalty and satisfaction. If the project succeeds in accurately predicting flight delays, it can make a substantial difference in several areas. Firstly, travelers will experience improved journey planning, leading to reduced stress and smoother travel experiences. This could translate into significant time and cost savings for passengers who can avoid missed connections and unnecessary expenses due to unexpected delays. Additionally, airlines can optimize their operations, leading to cost reduction and potentially lower ticket prices for customers. Overall, the project's success holds the potential to revolutionize air travel, benefiting both passengers and industry stakeholders alike. Furthermore, the project underscores the value of data-driven decision making, offering opportunities for research and development in aviation analytics. Also, the project can stimulate advancements in predictive modeling methodologies, encouraging researchers and practitioners to develop more robust algorithms capable of handling diverse and dynamic datasets inherent in the aviation domain.

## 2. Data Understanding

We initially load the data corresponding to the month December into Colab from Bureau of Transportation Statistics(BTS)[1] for the years 2018-2023. Then we concatenate this data and get an overview of the dataset's structure, including the number of rows and columns, column names, and data types. In the dataset, we see that there are 110 columns related to a flight and its journey. Among these columns, we see that a few of them have NaN values in them and so we ignore these columns. We shortlist the columns to FlightDate, DOT_ID_Reporting_Airline, OriginAirportID, OriginCityName, OriginState, DestAirportID, DestState, DepDelay, ArrDel15, Diverted, AirTime, Flights, Distance, SecurityDelay, DivDistance, DivArrDelay. Among these columns, we count for the number of null values in each column and drop the last three columns as they are completely null. Our target column is ArrDel15. We drop the rows in the dataset whose value in target column is NaN. We fill the NaN values in the columns DepDelay and AirTime with the mean of respective columns. We take a snapshot of dataset corresponding to (a few)columns of interest[AirTime in Hours, Distance in Miles, DepDelay and ArrDel15 in minutes] and it looks as follows:

[1]BTS Data for Airplanes

Table 1: First 4 rows, 6 columns of BTS Dataset

| DIRA | OS | AirTime | Distance | DepDelay | ArrDel15 |
|---|---|---|---|---|---|
| 20304 | TX | 30.0 | 134.0 | -10.0 | 0.0 |
| 20304 | CO | 35.0 | 125.0 | 1.0 | 0.0 |
| 20304 | AZ | 38.0 | 513.0 | 28.0 | 0.0 |
| 20304 | AL | 81.0 | 147.0 | -5.0 | 0.0 |

DIRA = DOT_ID_Reporting_Airline
OS = OriginState

### 2.1 Data Visualisation

**2.1.1 Outlier Analysis :** We perform outlier analysis for DepDelay, AirTime, Distance, Flights and draw a box plot.

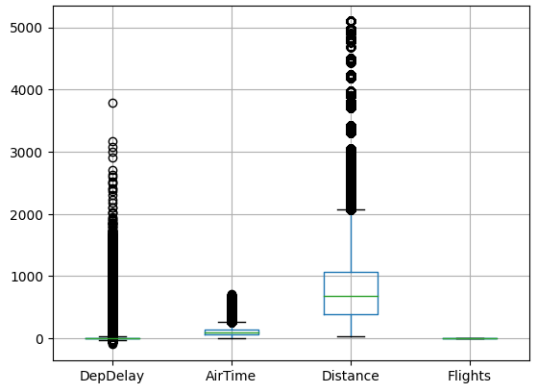


Figure 3: Box Plot of Outlier Analysis

**2.1.2 Counts vs Categorical Columns :** We perform bar plots for the count of respective categories in Categorical columns.

**2.1.3 Covid Impact :** We plot a Stacked Bar plot for the number of delayed flights for each year. We observe that the number of flights decreased during Covid due to lockdown restrictions and also the percentage of delay is minimal during this phase, probably due to less air traffic.

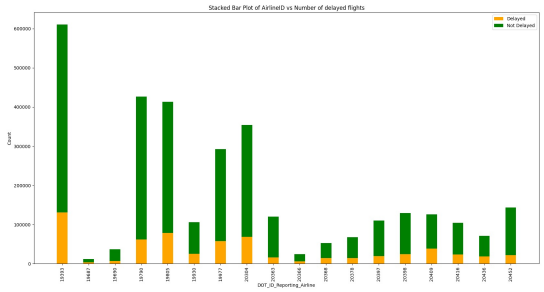


Figure 4: Stacked Bar Plot of DotIDReportingAirline vs Number of delayed flights

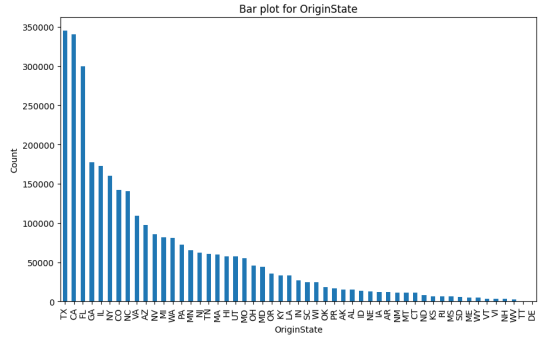


Figure 5: Bar Plot for Origin State


Figure 6: Stacked Bar Plot of Year vs Number of Delayed Flights

### 2.2 Correlation Analysis

**2.2.1 Continuous vs Target Variables :** We perform Point-Biserial Correlation[2] analysis between Continuous variables DepDelay, AirTime and Distance with respect to target variable ArrDel15. We obtain the following results.

Table 2: Point-Biserial Correlation analysis for ArrDel15

| Parameter | Coefficient | p-value |
|---|---|---|
| DepDelay | 0.5019741178870925 | 0.0 |
| AirTime | 0.04926971364956155 | 0.0 |
| Distance | 0.03069292284341516 | 0.0 |

**2.2.2 Categorical vs Target Variables :** We perform Chi-Square Correlation[3] analysis between Categorical variables FlightDate, OriginCityName, OriginState and DestState with respect to target variable ArrDel15. We obtain the following results.

Table 3: Chi-Square Test Correlation analysis for ArrDel15

| Parameter | Coefficient | p-value |
|---|---|---|
| FlightDate | 161389.49846069462 | 0.0 |
| OriginCityName | 17078.231050360017 | 0.0 |
| OriginState | 10390.590058479766 | 0.0 |
| DestState | 14572.122026900264 | 0.0 |

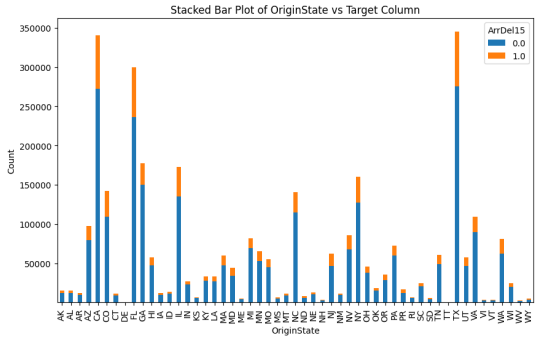


Figure 7: Stacked Bar Plot of OriginState vs ArrDel15

[2]Point-Biserial Correlation Coefficent - Wiki
[3]Chi Square Test - Wiki

### 2.3 Data Transformation

We standardise numerical variables like DepDelay, AirTime, Distance using StandardScaler() and normalize the data using MinMaxScaler().

### 2.4 Integration with Bird Strike Data[4]

We integrate our cleaned BTS dataset with Bird Strike Data downloaded from kaggle. We import Bird Strike dataset into Colab and make an inner join when altitude of flight in the BTS data matches the altitude of bird in Bird Stike data. A sample dataset from this merged dataset looks as follows :

Table 4: First 4 rows, 6 columns of Merged BTS-Bird Strike Dataset

| DIRA | DepDelay | ArrDel15 | Alt | SoB | W |
|---|---|---|---|---|---|
| 20304 | -10.0 | 0.0 | 400 | M | N |
| 20304 | 1.0 | 0.0 | 700 | L | Y |
| 20304 | 28.0 | 0.0 | 150 | S | N |
| 20304 | -5.0 | 0.0 | 530 | M | Y |

DIRA = DOT_ID_Reporting_Airline
Alt = Altitude(in ft)
SoB = Size of Bird(S = Small, M = Medium, L = Large)
W = Pilot being warned(Y = Yes, N = No)

Since flight delay is a continuous variable and bird size in bird strike is a categorical variable, it's not appropriate to calculate a correlation coefficient between them. Instead, we compare the average flight delay for different categories of bird sizes using a t-test[5].

Table 5: t-test for ArrDel15

| Bird Size | t-value | p-value |
|---|---|---|
| Small vs Medium | -0.8413 | 0.0013 |
| Medium vs Large | -1.0562 | 0.3217 |
| Large vs Small | 0.9761 | 0.0003 |

We see that as the t-test values are small($< 2$), we can conclude that the effect of bird size on ArrDel15 is insignificant.
So, we will not be integrating BTS dataset with Bird Strike dataset as a training data to our ML model to predict flight delay.

[4]Kaggle - Bird Strike Dataset
[5]Student's t-test - Wiki

### 2.5 Integration with Weather Data

We integrate our cleaned BTS dataset with Weather Data using Openmeteo API. We install Openmeteo API[6] in Colab and then we make an inner join when location in the BTS data matches the location of Weather data. A sample dataset from this merged dataset looks as follows :

Table 6: First 4 rows, 6 columns of Merged BTS-Weather Dataset

| DIRA | DepDelay | ArrDel15 | WC | RS | SS |
|---|---|---|---|---|---|
| 20304 | -10.0 | 0.0 | 71.0 | 0.5 | 0.91 |
| 20304 | 1.0 | 0.0 | 3.0 | 0.0 | 0.00 |
| 20304 | 28.0 | 0.0 | 53.0 | 1.1 | 0.00 |
| 20304 | -5.0 | 0.0 | 73.0 | 0.0 | 1.12 |

DIRA = DOT_ID_Reporting_Airline
WC = Weather_Code
RS = Rain_Sum
SS = Snowfall_Sum

We performed point-biserial correlation analysis for ArrDel15 with respect to Snowfall_Sum and Rain_Sum.

Table 7: Point-Biserial Correlation analysis for ArrDel15

| Parameter | Coefficient |
|---|---|
| Snowfall_Sum | -0.412 |
| Rain_Sum | 0.256 |

We see that the point-biserial correlation coefficient for snowfall is -0.412, indicating a moderate negative correlation between flight delay and snowfall. This suggests that higher snowfall is associated with shorter flight delays, or possibly even fewer delays.
We see that the point-biserial correlation coefficient for rainfall is 0.256, indicating a weak positive correlation between flight delay and rainfall. This suggests that higher rainfall is associated with slightly longer flight delays, although the correlation is not very strong.
We see that weather data can influence flight delay. So, we will be comparing our model trained on BTS dataset with model trained on BTS-weather data merged dataset while predicting flight delay.

[6]Open Meteo - Weather Data API

## 3. Methodology and Techniques

### 3.1 Compare tree-based methods on small dataset

In this experiment, we want to explore the performance of tree-based methods on flight delay prediction tasks on selected small dataset. This dataset includes the Pennsylvania (subset of all states' data) December flight delay data from 2018 to 2023. We are using 2018-2022 data as training set, and 2023 data as test set in this experiment. The selected features are all numeric: DayofMonth DayOfWeek OriginWac DestWac DepDelay DepDel15 Distance. The target class is ArrDel15, which is the arrival delay binary indicator. A Decision Tree is a flowchart-like tree structure where each internal node represents a "test" on an attribute (e.g., whether the departure delay is more than 15 minutes), each branch represents the outcome of the test, and each leaf node represents a class label (delayed or not delayed). (Swain & Hauska, 1977) The paths from root to leaf represent classification rules. Then we also tried the random forest classifier. A Random Forest is an ensemble of Decision Trees, usually trained with the "bagging" method to increase the overall result's accuracy. The basic idea is to build several Decision Trees independently and let them vote for the most popular class. (Parmar et al., 2019) This method is generally more accurate than a single decision tree. Moreover, since we observed that the dataset is unbalanced with less label 1 records for ArrDel15, the balanced random forest model might be helpful. A Balanced Random Forest is an adaptation of the standard Random Forest designed to handle imbalanced data sets. It adjusts the class distribution in the training dataset by undersampling the majority class, oversampling the minority class, or both during the tree training phase. Each tree is built on a different, balanced subset. (More & Rana, 2017) In short, we used the same dataset settings and preprocessing techniques aiming to compare the performance of different tree-based methods using metrics like accuracy and f1-score.

### 3.2 Evaluate data mining techniques

1. **KNN (K-Nearest Neighbors):** A simple, instance-based learning algorithm that classifies data points based on the majority class of their nearest neighbors. With its simple architecture, KNN (Guo et al., 2003) provided a significant speed boost, making it suitable for quick prototyping and initial exploration of the data. By classifying data points based on the majority class of their nearest neighbors, KNN offered a straightforward approach to classification tasks.

2. **XGBoost (Extreme Gradient Boosting):** A powerful ensemble learning algorithm that builds a series of decision trees sequentially, optimizing the overall model's performance. Known for its extreme gradient boosting technique, XGBoost (Chen & Guestrin, 2016) stood out for its powerful ensemble learning capabilities. By building a series of decision trees sequentially, XGBoost optimized the overall model's performance and handled complex relationships within the data effectively. This made it a strong contender for improving predictive accuracy.

3. **AdaBoost:** A boosting algorithm that combines multiple weak classifiers to create a strong classifier. AdaBoost (Hastie et al., 2009) offered a unique approach by combining multiple weak classifiers to create a strong classifier. Its focus on improving the performance of misclassified instances at each iteration made it a valuable addition to our model selection process.

4. **TabNet:** A deep learning model specifically designed for tabular data, which uses attention mechanisms to select informative features for decision making. TabNet (Arik & Pfister, 2021) captured our interest due to its attention mechanisms, which select informative features for decision-making. This adaptability to tabular data made TabNet a promising choice for extracting meaningful insights from our dataset.

By experimenting with these diverse techniques on both balanced subsets and the entire dataset, we aimed to leverage their respective strengths and assess their performance across different scenarios. This comprehensive approach allowed us to gain insights into the behavior of each model and select the most suitable ones for our specific task.

### 3.3 Implementation Steps and Model Voting

In sections 3.1 and 3.2, we experimented on a small subset (PA only) of the large dataset that includes flight delay data in all states, which helped us get the best performing models inspite of bias in the dataset. In our quest for maximizing predictive performance, we adopted a strategic approach by employing model voting. This technique allowed us to harness the collective wisdom of multiple models and achieve better generalization and robustness in our predictions.

#### Model Voting

Model voting (Polikar, 2006) involves combining the predictions from multiple individual models to

generate a final prediction. We utilized a **voting ensemble** approach, where each model gets an equal vote in the final prediction. This ensemble strategy helps in reducing the risk of over fitting and bias, as it aggregates diverse viewpoints from different models.

**Benefits of Model Voting**

1. **Improved Robustness**: By aggregating predictions from multiple models, we mitigate the impact of individual model weaknesses or biases, leading to more robust predictions.
2. **Enhanced Generalization**: Model voting leverages the diversity among models to capture different aspects of the data, thereby improving the generalization ability of the ensemble.
3. **Risk Mitigation**: In case of model failure or under performance, the ensemble remains resilient due to the combined strength of multiple models.

**Models Parameters for Voting**

To ensure effective integration of models in the voting ensemble, we carefully tuned their parameters to achieve optimal performance. Below is a summary table outlining the key parameters for each model:

- **TabNet:** Learning Rate = 0.01, Batch Size = 64
- **XGBoost:** Number of Estimators = 100, Learning Rate = 0.1
- **AdaBoost:** Number of Estimators = 50, Learning Rate = 1.0

**SMOTE in Parameter Experimentation**

In addition to tuning model parameters, we also experimented with Synthetic Minority Oversampling Technique (SMOTE) to address class imbalance issues. SMOTE artificially increases the number of minority class instances by creating synthetic samples. This technique helped in improving the performance of models, particularly in scenarios where class imbalance was prominent.

By carefully selecting and tuning these parameters, we were able to optimize the individual performance of each model before incorporating them into the voting ensemble. This ensured that each model contributed meaningfully to the final prediction, leading to enhanced overall performance.

In the subsequent stages of our experimentation, we combined the predictions from these tuned models using a suitable voting strategy to derive the final predictions for our target variable, which is our final ensemble architecture.

## 4. Results and Analysis

**Evaluation Metrics**

In our experiments, we mainly focus on four key metrics: Recall (Alaparthi et al., 2022), Precision(Alaparthi et al., 2022), F1-Score(Alaparthi et al., 2022), and Accuracy(Alaparthi et al., 2022). Recall measures the model's ability to correctly identify all positive instances, crucial in scenarios like disease diagnosis. Precision gauges the proportion of correctly identified positive instances out of all instances predicted as positive, significant in situations like spam detection. F1-Score provides a balance between Precision and Recall, especially valuable in the presence of class imbalance. Accuracy, while commonly used, may not reflect performance accurately in imbalanced datasets but offers a general indicator of correctness. By reporting these metrics, our research offers a nuanced understanding of model performance, aiding informed decision-making for specific applications.

### 4.1 Tree-based Model Comparison

Table 8: Tree-based Model Comparison

| Model | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| Decision Tree | 0.77 | 0.80 | 0.79 | 0.88 |
| Random Forest | 0.89 | 0.83 | 0.86 | 0.93 |
| Balanced Rand. F. | 0.78 | 0.85 | 0.81 | 0.88 |

From the results, we can see that overall the Random Forest method performs the best. For the macro average of precision and f1-score, random forest achieves 0.89 and 0.86. Notably, the Balanced Random Forest classifier achieves the highest macro average of recall among three tree-based models. For overall accuracy, the Random Forest model still performs the best, which is 0.93. This result shows that the Random Forest model is performing excellently for the flight delay prediction task in the selected small dataset of PA's December data. The results indicate the model that is both accurate and reliable across different types of predictions, making it well-suited for operational use.

### 4.2 Different models comparison

In this section we present the results of experiments on the large dataset with optimal columns mentioned in section-2. We compare the performance of various machine learning and deep learning algorithms like XG Boost, ADA Boost, Random Forest, KNN, Tabnet and a Voting based classifier which is a ensemble of XG Boost, ADA Boost and Tabnet. As Show in 9 the performance of all models is almost close due to the distribution of data and very less delays in general as shown in 6

Table 9: Evaluating performance on large dataset

| Model | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| XG Boost | 0.93 | 0.86 | 0.9 | 0.94 |
| ADA Boost | 0.91 | 0.84 | 0.87 | 0.92 |
| Random Forest | 0.92 | 0.86 | 0.89 | 0.93 |
| KNN | 0.77 | 0.80 | 0.79 | 0.88 |
| Tabnet | 0.91 | 0.85 | 0.88 | 0.93 |
| **Voting Classifier** | 0.94 | 0.87 | 0.91 | 0.94 |

### 4.3 Different sizes of data comparison

In section 3.1, we experimented on a small subset (PA only) of the large dataset that includes flight delay data in all states. For this experiment, we won't change other settings like model, features, and data preprocessing. Instead, we will change the size of the training set, and test the prediction on the large dataset as well. For the classifier model, we used Random Forest, which overall performs the best among tree-based methods. The features selected are the same as last experiments: DayofMonth DayOfWeek OriginWac DestWac DepDelay DepDel15 Distance. We preprocessed the data by dropping NaN ArrDel15 rows and using mean values to fill the missing values. For the small dataset, we gradually increased the size of the train set. Starting from using 2020-2022 data as training set, and 2023 data as test set. Then we added 2019 to the train set to observe the difference. After that, we also added 2018 data to the training set.

Table 10: Enlarged Dataset Comparison (Random Forest)

| Train Set | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| PA 2020-2022 | 0.88 | 0.83 | 0.85 | 0.92 |
| PA 2019-2022 | 0.89 | 0.83 | 0.86 | 0.93 |
| PA 2018-2022 | 0.89 | 0.83 | 0.86 | 0.93 |

From the results, we can notice that when we added 2019 data to the training set, the performance slightly improved. The macro avg f1-score and accuracy both increased by 0.01. However, after adding 2018 data, the performance doesn't change. The size of the training set might be stable and enough for our flight delay prediction task at this time.

Table 11: Performance Metrics on Large Datasets

| Train Set | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| PA 2018-2022 | 0.89 | 0.83 | 0.86 | 0.93 |
| ALL 2018-2022 | 0.88 | 0.83 | 0.85 | 0.93 |

For time considerations, we only trained the Random Forest classifier on the large dataset that includes all states' records (about 20 times larger than PA only) once to prove that the performance of the model on both small and large dataset are similar.

### 4.4 Weather and COVID influences

We have finally analysed the influence of Covid and adding Weather data on the performance of the models.

Table 12: Performance Metrics on Large Datasets

| Eval Set | F1-Score | Accuracy |
|---|---|---|
| 2020 | 0.89 | 0.94 |
| 2021 | 0.87 | 0.93 |
| 2023 | 0.89 | 0.94 |

For year 2020 Surprisingly we got good score which we are suspecting due to less air traffic at that time as seen in 6. Rest of the scores were as expected with a slight dip in year 2021, probably due to cases oscillating at that time. With incorporating weather data we observed a slight dip again in our F1 score, where the best model bench marked at **0.87**.

## 5. Discussion and Conclusion

The findings of this study hold significant implications for the aviation industry, particularly in the realm of flight delay management. The results from our experiments on the dataset highlight the effectiveness of ensemble methods, particularly model voting, in enhancing predictive accuracy for flight delay prediction. The voting classifier exhibits superior performance across precision, recall, and F1-score metrics. This underscores the potential of ensemble methods to mitigate individual model weaknesses and biases, ultimately leading to more robust predictions. One significant implication of this result is the confirmation of the utility of model voting in enhancing predictive accuracy and generalization. By successfully developing and validating a predictive model for flight delays using machine learning techniques, this project has demonstrated the potential to revolutionize how airlines and airports mitigate operational disruption. The accurate prediction of flight delays enables proactive decision-making, allowing airlines to optimize flight schedules, allocate resources efficiently, and minimize the impact on passengers. Furthermore, the data pre-processing techniques and implementation of different machine learning algorithms has

paved the way for more robust and reliable predictive models in the aviation industry.

In considering the scope of the project, opportunities for improvement abound, particularly in refining pre-processing techniques and incorporating additional features to enrich the predictive capabilities of the model. Exploring novel feature engineering methods and experimenting with ensemble learning techniques may yield further improvements in predictive accuracy. Additionally, the limited trials were conducted for each model setting due to time constraints. Future studies should consider multiple runs to ensure the stability of results.

In conclusion, by developing a predictive model that leverages historical flight data and machine learning techniques, this study has contributed to advancing predictive analytics in aviation and has the potential to drive positive change in operation efficiency and customer experience. As the aviation industry continues to evolve, the insights gained from this project will inform future endeavors aimed at enhacing the reliability and effectiveness of flight delay prediction models, ultimately benefiting stakeholders and passengers alike.

## References


Alaparthi, V. S., Pasam, T. R., Inagandla, D. A., Prakash, J., & Singh, P. K. (2022). Scser: Supervised contrastive learning for speech emotion recognition using transformers. *2022 15th international conference on human system interaction (HSI)*, 1–7.

Arik, S. Ö., & Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI conference on artificial intelligence*, *35*(8), 6679–6687.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). Knn model-based approach in classification. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, 986–996.

Hastie, T., Rosset, S., Zhu, J., & Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface*, *2*(3), 349–360.

More, A., & Rana, D. P. (2017). Review of random forest classification techniques to resolve data imbalance. *2017 1st International conference on intelligent systems and information management (ICISIM)*, 72–78.

Parmar, A., Katariya, R., & Patel, V. (2019). A review on random forest: An ensemble classifier. *International conference on intelligent data communication technologies and internet of things (ICICI) 2018*, 758–763.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, *6*(3), 21–45.

Swain, P. H., & Hauska, H. (1977). The decision tree classifier: Design and potential. *IEEE Transactions on Geoscience Electronics*, *15*(3), 142–147.