# Prediction of On-time Flight Performance

MOHAMMED ZOHER GUNIEM, University of Stavanger, Norway

Since the very first successful flight performed by the Wright brothers at the beginning of the last century, aviation has continued to shorten the distances between the different corners of our world. This has brought together a wide variety of advantages along with lots of interesting challenges to be solved. One of these challenges is the on-time performance of commercial flights. This research paper aims to use existing data from on-time performance measurements for building a robust regression model to estimate a delay or a possible early departure of future flights. Being able to predict flight delays gives any airline a great advantage to prepare for possible delays and therefore be able to prevent them. This also saves the time of both passengers and airline companies and reduce the economic and social impact of flight delays on modern societies.

CCS Concepts: • **Flight On-time Performance**; • **Supervised Machine Learning**; • **Flight Delay Analysis**;

Additional Key Words and Phrases: XGBoost, Regression, Delay Prediction

**ACM Reference Format:**
Mohammed Zoher Guniem. 2024. Prediction of On-time Flight Performance. In *Proceedings of (UiS)*. Stavanger, Norway, 9 pages.

## 1 Introduction

Building an accurate estimator of flight delays and early departures requires a dataset with good quality and integrity. These properties are to be found in the dataset provided by the Bureau of Transportation Statistics in the US [1]. The dataset includes various details about the on-time performance of domestic commercial flights over the united states. The most important of these details are information about the origin and destination of a flight, scheduled departure and arrival times, airline, aircraft types and various columns that contain the registered delay of a historic flight both in the number of minutes and with many types of time intervals. Such information about on-time flight performance makes this dataset a good foundation for further analysis and model building in this research.

The aim of this research is to use the historic information about on-time flight performance to build a supervised machine learning model that acts as a regressor and helps estimate the delay of a future flight. For this purpose data from 2022 and 2023 are downloaded to be used in this research; the first dataset from 2022 is used for tuning, evaluating and training the regressor, and the 2023 dataset is used for final testing and outcome analysis.

The project is kicked-off by checking that the downloaded data meets the quality requirements in this research, where data points with bad quality are cleaned away from the datasets. This startup step gives the confidence to start visualizing the distributions of recorded delays across the different features in the dataset. Which highlights the importance of each feature in estimating a certain delay or early departure value.

Furthermore, the categorical features in the dataset is mapped and encoded to suit the need of the regressor in having numerical values to work with. This regressor uses the great capabilities of XGBoost library that builds on the gradient boosting framework, which is known for its efficiently and scalability for regression tasks [2]. The regressor models in this research are firstly tuned and evaluated before being used to build various types of on-time performance estimators in an experimental environment. The results of these experiments will determine which regressor is the most efficient to predict the on-time performance of future flights based on evaluation metrics such as the Huber Loss, Mean Absolute Error and Mean Squared Error.

On the final stage of this project, the experimental models is trained on the training dataset from 2022 by using its previously tuned parameters. The model is then used to estimate delays of the future delays in the 2023 dataset before comparing these predictions to the ground known truth of delays in the year of 2023 to be able to analyze the errors of this estimators and how good it is in anticipating the number of minutes a certain flight might be delayed or leave earlier than scheduled.

## 2 Data Processing

Although the dataset is considered to be of high quality, it is still important to run a control check according to the need of this research. Selecting the right relevant features from the dataset and understanding such features constitutes the first step toward building an efficient and more accurate estimation model for flight on-time performance. The dataset is presented in tabular format and contains feature columns about origins, destinations, departure and arrival times along with information about operating airlines and used airplanes. While on the target side, this dataset contains information about departure delays and/or early departure in many formats such as delays in the number of minutes and in 15 minutes intervals.

Many of these features are useful for delay estimation model. However, information such as flight number is unique for every flight and non-informative for predicting on-time performance. The dataset also contains already encoded feature columns about the destination and origin, the encoded feature column is useful for the algorithm of model building, while the categorical feature us useful for visualization to suit the analytical human eye.

The dataset also contains a small number of rows with non-identified information in any feature or on-time performance measures. Due to the much bigger size of the overall dataset rows (about 6.5 millions rows for each of the 2022 and 2023 datasets), it is best to remove such rows to avoid bugs and missing information during model building. in total 177352 rows are removed from the 2022 training dataset and 84531 rows are removed from the testing

Author's Contact Information: Mohammed Zoher Guniem, mghunime@yahoo.no, University of Stavanger, Stavanger, Norway.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*UiS, Prediction of On-time Flight Performance, Stavanger, Norway*

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

2023 dataset due to missing information in the departure delay and scheduled elapsed time columns.

Furthermore, Some requirements are established according to each feature and target columns; two examples of such requirements is the expected data type and ranges of values a column cell can have. Table 1 shows the data quality requirement for both the Quarter and ORIGIN columns. The overall view of all requirements and the actual checking of each requirement is available in the preprocessing notebooks of this project.

Table 1. Data Quality Requirements

| Feature | Data Quality Requirement |
|---|---|
| QUARTER | Values must range from 1 to 4, by 1 step<br>Value is required; If Nan, delete row<br>Value type is int |
| ORIGIN | Value is required; If Nan, delete row<br>Value type is category |

## 3 Flight Delay Analysis

Understanding the relations between features in the dataset and the targeted on-time performance is crucial in selecting the right set of features for the desired purpose of a machine learning model. Many useful visualizations has been generated in this project aiming to gain an initial insight into the training dataset and build more knowledge about the distribution of recorded delays and early departures across these features.

An increasing accumulation of departure delays is noticed towards 6 pm. O'clock in Figure 1 before dropping down to minimal values during after midnight hours. This clearly shows the rush hours and correlation between the hour a flight is scheduled in and its probability to be delayed.

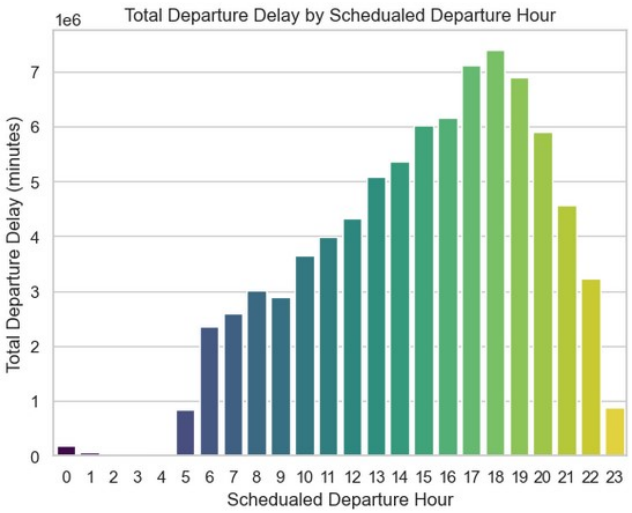


Fig. 1. Total Departure Delay by Scheduled Departure Hour

Another interesting notice is the distribution of departure delays across the 17 operating airlines in the training dataset, where 6 airlines stands for about 74% of all recorded departure delays. Although this might be connected to the volume of flights an airline operate, it still represents a correlation key feature to be taken into consideration when building a model to estimate on-time flight performance. Figure 2 shows this distribution of departure delays across the airlines in the training dataset. Similar trends has also been noticed among origin and destination features such as airport, states and cities, where busy airports and big cities stands for a greater amount of recorded departure delays.

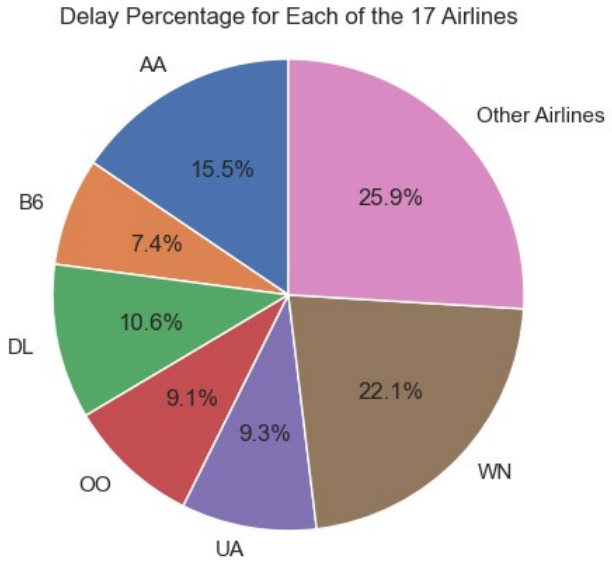


Fig. 2. Total Departure Delay among the different airlines in the dataset

The correlation value between numerical continuous features and the number of minutes in departure delay seen in Figure 3 shows that the scheduled departure and arrival times are highly correlated to departure delays than all other numerical features, while features like the day of week, scheduled elapsed time and distance to fly are in the middle range of correlation. We also notice that the quarter, month and day of month are of lower correlation to the values of departure delays. Although the maximum correlation does not exceed 0.10 we later will see how to use these combination of features in a machine learning model to achieve good results of predicting departure delays for the entire year of 2023.

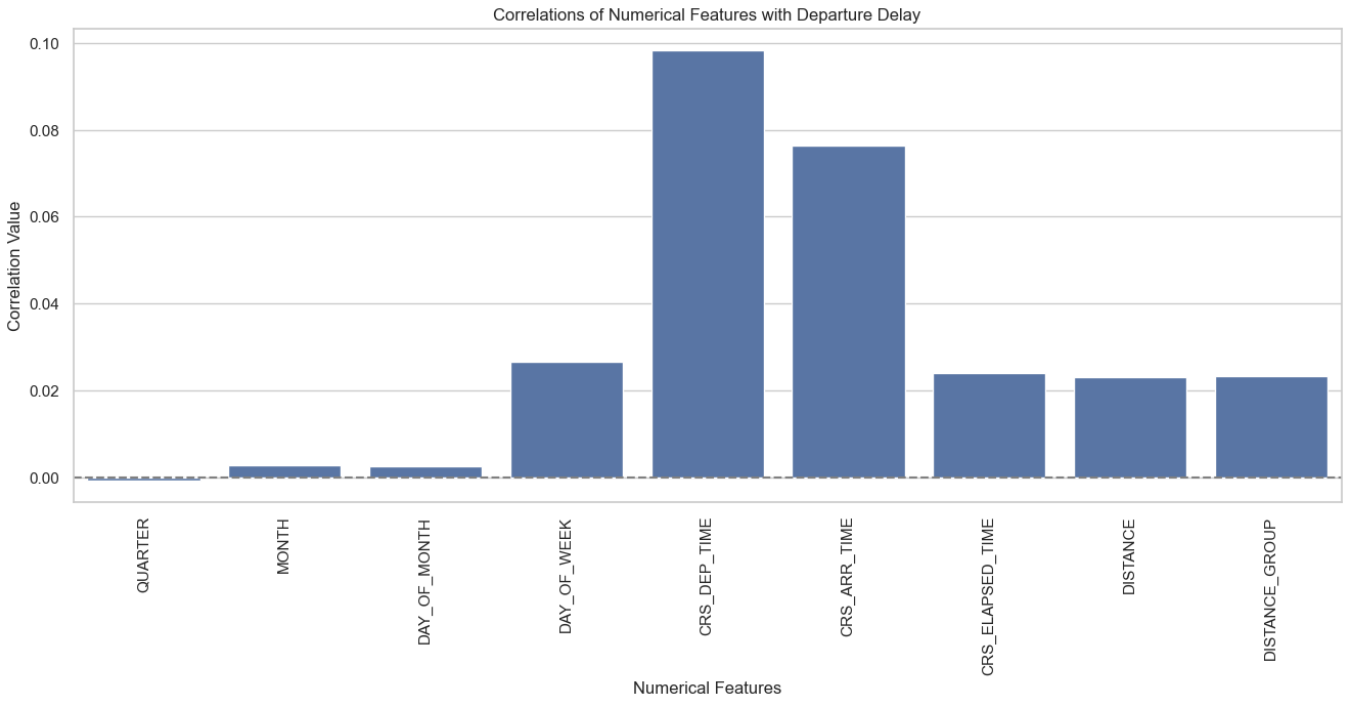


Fig. 3. Correlation Values Between Numerical Features and Departure Delays.

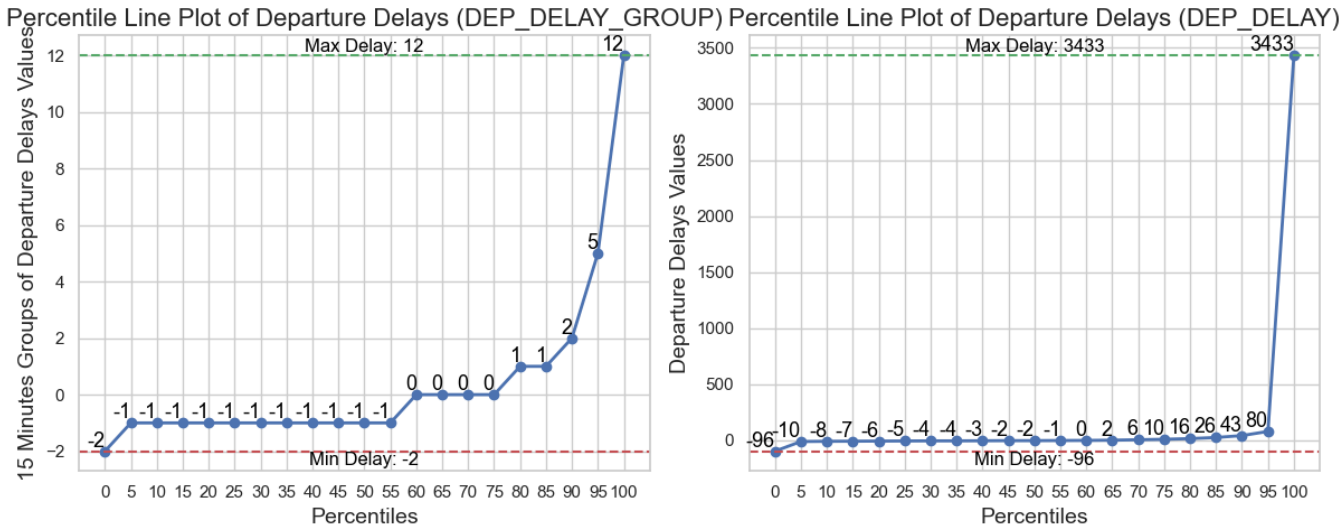


Fig. 4. Percentiles vs. Their values of departure delays with 15 minutes intervals on the left, and number of minutes on the right.

Lastly and before we can dive into model building, it is important to have an idea on the range of departure delays, which is shown in Figure 4 in both 15 minutes intervals (keeping in mind that delays larger than 179 minutes are registered in one category denoted as group 12 and early departures less than -15 minutes are also registered in one category denoted as group -2 for the purpose of limiting the range of values), This range limitation does not apply for the plot on the right where we study the actual numerical value of delays or early departures in minutes.

The right plot in Figure 4 shows that 35% of values lays between 0 and 80 minutes, and another 5% include large outliers for delays up to 3433 minutes. This observation is somehow minimal but still visible in the left plot where 15 minutes intervals are used, as both plots shows that 90% of delays or early departures values are located in the space of -10 minutes to around 80 minutes with small number of large outliers at the right departure delay end, and some moderate outliers on the left early departure end.

Such outliers might be harder to predict but removing them from the dataset might cause us to loose hidden key information about some flights and routes. Later in the experimental part in this project we will keep an eye on such outliers and their total effect on the performance of the built machine learning model under evaluation.

Many more insightful visualizations are also available in the visualization notebook within the source code of this project.

## 4 Preparing the datasets

Although the dataset includes lots of categorical features, it also includes already encoded columns that correspond to those features and has an attached selection of lookup tables that makes the interpretation between encoding and decoding more agile. However, this is not the case for the tail number that identify an aircraft used under a certain flight, or the departure and arrival time blocks which shows the hour interval when this flight were scheduled in a categorical format. To deal with this obstacle, the values of these 3 features are mapped and label encoded then stored in a separate CSV file containing all the encoded and numerical features of the dataset and ready to be used for model building.

Later on, standardization and min/max scaling has been applied on the training dataset but without registering any noticeable improvement in term of either model performance or speed of convergence, while adding more complexity to the dataset made it hard to interpret. Based on these hard consequences our dataset should work fine with the XGBoost model without scaling and it was therefore removed from the dataset preparing steps.

The final result of preparation is now a csv file for each of the training and testing datasets, where numerical values are kept as they are, while categorical values are encoded into numerical categories using label encoding. These data will then be used under model building for tuning and evaluation along with the different experiments and final testing.

## 5 The Algorithm of Choice

At the early stages of this research many machine learning algorithms have been tried out on the training dataset such us Random Forest, Support Vector Machines and even deep learning algorithms like Long-Short Term Memory and multi-layer neural networks, the winning algorithm that gave the best combination of performance and running time was the gradient boosting algorithm XGBoost which is also known as Gradient Boosted Decision Trees (GBDT) with a configuration similar to the Random Forest algorithm but with better optimization in term of performance and speed. XGBoost also include features such as regularization, parallelization and early stopping, which help improve both efficiency and model accuracy in prediction or estimation tasks [2].

## 6 Model Tuning and Evaluation

An important step of building any machine learning model is adjusting its parameters to suit the training dataset for optimal or better performance. This can be achieved by many techniques each has its pros. and cons., the list below shows the top three most popular techniques to search model parameters and highlights their benefits and drawbacks.

- Grid Search
  This is the most extensive technique where every possible combination of the targeted tuning parameters is tried out by building and evaluating a model with each possible combination. The advantages of this searching technique is the guarantee of finding the optimal solution, however in practice and especially with large datasets this technique requires lots of computation power and running time.

- Early Stopping
  The idea behind this technique is to specify a number of tries (n) where the search is terminated if no improvement of performance is recorded after (n) trials of models, although this might give good results with some datasets, we cannot be confident of how far a certain model lays in term of performance from the optimal tuned model. This is because the optimal model might be discovered in the try number (n+1) and more.

- Random Search
  Random Search is a trade-off between the methods of Grid Search and Early Stopping. where randomly generated combinations within the range of each targeted model parameter is tried out and the best model is picked out as the winner. This technique proves to be an efficient way to search within the entire range of each parameter without requiring to try every possible combination with other parameters, and although it does not guarantee to find the optimal tuned model, it proves in practice that a model very close to the optimal model is often found using this technique. For this reason and to save time and computation resources the Random Search technique is adopted as the searching strategy of this research.

It is also important to not train and test a model with the same data, which can lead to misleading results as the model would often perform better with data it has seen before. This is why k-fold validation is used in this project, where k is set to 2 folds and the training dataset is randomly partitioned into 2 smaller datasets with

equal size, then a model is trained on the first dataset and evaluated using the second dataset, before repeating the same process but with using the second dataset as the training dataset and the first one as the evaluation dataset. Keeping track of errors in this process makes it possible to calculate a total evaluation result at the end of evaluation. The reason the training dataset is split in two parts during k-fold validation is because our future testing dataset from the year of 2023 and the training dataset from the year of 2022 are roughly of the same size, and therefore splitting the training dataset in half offers the same relative size between training and testing dataset as its going to be in the final testing stage, which puts us on the right track when evaluating different models under the way to use the tuned model parameters that are more likely to give a better performance.

Choosing the right evaluation metric is also crucial in identifying the capability of a model. To determine which evaluation model to use in this regression task, we can take a look at the distribution of departure delay values across the training dataset in Figure 4 where we notice a small number of high outliers at the beginning and end of the targeted delay values compared to the normal range of values between -10 and +80 minutes of delay or early departure. This tells us that using the Mean Squared Error (MSE) would be unfair for our dataset as it is very sensitive to large outliers in errors, using the Mean Absolute Error (MAE) is also unfair to the performance when predicting the outliers as it distributes the errors across all data points. Which highlight the need of having an evaluation metric that combines both of MSE and MAE, and this is where the Huber Loss evaluation metric comes to the rescue as it is an alternating function that behaves somehow similar to MSE for errors smaller than a value delta, and switches to behave in a way similar to MAE for errors larger than a value delta. This delta value can be chosen using various techniques, but the most usual way of choosing delta is to have it equal to the percentile value at which the outliers starts appearing in the dataset and for the training dataset used in this research it is observed at the 95% percentile, so the delta value is set to be 80 error in minutes if the target is `DEP_DELAY` and group 5 if the target is `DEP_DELAY_GROUP` as shown in Figure 4.

The Huber loss function is defined as:

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \le \delta, \\ \delta \cdot (|y - \hat{y}| - \frac{1}{2}\delta) & \text{for } |y - \hat{y}| > \delta, \end{cases}$$

where $y$ is the true value, $\hat{y}$ is the predicted value, and $\delta$ is a threshold parameter.

Now that we have established the tuning strategies of searching parameters, k-fold validation and evaluation metrics, it is time to put all these pieces toghether and run a tuning job that gives us the better combination of XGBoost model parameters to be used in model experimenting and testing later in this research. XGBoost has lots of parameters to play with, but most of them works fine with their default values, and to narrow the focus on the most important parameters, we will be tuning the most important parameters explained in the following list:

- `max_depth`
  Controls the maximum depth of each tree estimator in the model. It has a typical range from 3 to 10, where larger values allow deeper trees, which may capture more information but risk overfitting, while smaller values lead to simpler models.

- learning_rate
  Controls the step size at each iteration while moving toward a minimum. It has a typical range from 0.01 to 0.3, where smaller values generally improve performance but increase training time and may need more boosting rounds. Larger values can speed up learning but risk overfitting.

- subsample
  Specifies the fraction of training data used to train each tree, which introduces randomness and helps prevent overfitting. It has a range from 0.1 to 1.0, where a value less than 1.0 reduces the amount of training data for each tree, which can help generalization. A value too small may lead to underfitting.

- colsample_bytree
  Similar to subsample this parameter specifies the fraction of features to be randomly selected when building each tree. It has a range from 0.1 to 1.0, where value less than 1.0 reduces the number of features used for each tree, which can help prevent overfitting and reduce model complexity. A smaller value can also speed up training.

- min_child_weight
  Specifies the minimum sum of instance weights in a child and therefore controls the minimum size of a leaf and helps prevent overfitting by requiring more data to create a split. it has a typical range from 1 to 10 where higher values make the model more conservative by requiring more instances in a leaf node before it can split. On the other hand, lower values allow for more splits, potentially capturing more specific patterns but at the risk of overfitting.

- n_estimators
  Refers to the number of boosting rounds, or the total number of trees that will be built during the training process. It has a typical range from 100 to 1000 estimators depending on the size and complexity of the training dataset. A higher value of n_estimators allows the model to train more trees, potentially improving performance, but can also lead to overfitting if the model becomes too complex. It's important to balance it with the learning rate since a smaller learning rate may require more trees to reach a good solution.

The results of randomly searching these parameters with 100 different random combinations across their ranges and using 2-fold validation are shown in Table 2. This tuning has also been performed using two targets separately; the first target `DEP_DELAY` uses the actual minute value of early delay or departure, and the second target uses the `DEP_DELAY_GROUP` where delays has been grouped

into 15 categories with 15 minutes intervals between each category -1 to 11, and where category -2 is dedicated to early departures lower than -15 minutes and category 14 is dedicated to departures higher than 180 minutes. Table 2 also shows the calculated Huber Loss score for each of the two searches performed

Table 2. Best Found Values of XGboost Parameters using DEP_DELAY and DEP_DELAY_GROUP respectively

| Regression Target | DEP_DELAY | DEP_DELAY_GROUP |
|---|---|---|
| Parameter | - | - |
| max_depth | 8 | 9 |
| learning_rate | 0.22 | 0.25 |
| subsample | 0.81 | 0.48 |
| colsample_bytree | 0.79 | 0.87 |
| min_child_weight | 5 | 5 |
| Huber Loss Score | 629 | 2.02 |

When it comes to tuning the number of estimators in a model, a different approach is used due to the limitation of computing power, as training an XGBoost model with higher number of estimators than the default 100 often requires more computational resources than what most modern computers can provide. The solution here is to try every number of estimators from 100 to 1000 with an interval of 50 using the tuned values of parameters from the previous random search. This improves the Huber Loss score of the model using DEP_DELAY as a target with a value of 950 estimators, but does not seem to improve the performance of the model using DEP_DELAY_GROUP as target. The optimal values of n_estimators and corresponding Huber Loss score for each model is shown in Table 3

Table 3. Best Found Values of n_estimators using DEP_DELAY and DEP_DELAY_GROUP respectively

| Regression Target | DEP_DELAY | DEP_DELAY_GROUP |
|---|---|---|
| Parameter | - | - |
| n_estimators | 950 | 100 |
| Huber Loss Score | 615 | 2.02 |

The reason the model using DEP_DELAY_GROUP as target has a very low score is not because it is much more accurate than the model using DEP_DELAY, but rather because the grouped target shrinks the target space into 15 values which gives such low Huber Loss score value. As we would see later, building a regression model using DEP_DELAY_GROUP requires us to multiply the predicted value by the interval step of 15 minutes to get the number of minutes a delay or early departure is predicted to be.

More details and result graphs is available for more analysis inside the tuning and evaluation notebooks of the source code in this project, and later on in this report we would be using the achieved and documented values of parameters in this section to build and evaluate more models in different experiments on the way for a more accurate regression model to predict departure delays and early departures of commercial flights.

## 7 Experimentation

Now that we have tuned and evaluated the base regression models of XGBoost on the two targets DEP_DELAY and DEP_DELAY_GROUP we can put these models through different experiments that aims to improve their performance and capabilities. In the following two subsections we will explain the ideas behind each experiment and view their respective implementation setups before we discuss the achieved results of each experiment.

### 7.1 Hybrid XGboost Regressor Experiment

Noticing the high number of large outliers in DEP_DELAY relative to the rest of the dataset, we might want to take an advantage of the categorization done in the DEP_DELAY_GROUP column since it sets a gathering category of -2 for departures earlier than -15 minutes, and an another gathering category of 12 for all departures with delays larger or equal to 180 minutes. and as seen in Figure 4, most of the outliers might exist at those ends of the delay distribution.

As an experiment we would than train a tuned model of XGBoost regressor using the DEP_DELAY_GROUP as a target, which predicts a continuous float value, this predicted value is then multiplied with the category interval of 15 minutes to get the number of minutes a new flight is predicted to be delayed. then using the DEP_DELAY group as an evaluation column to see if this is better than training the model by directly using the DEP_DELAY as both training and evaluation target.

K-fold cross validation is also used here to make sure not to evaluate the model based on data it has already seen during training. And the reason why this model is given the name of Hybrid XGBoost Regressor is because it uses the DEP_DELAY_GROUP column as target in training, but then utilizes the column DEP_DELAY for evaluation of performance. Table 4 shows the final Huber Loss Score achieved under evaluation, which is unfortunately Higher than the tuned regressor model that used the DEP_DELAY under both training and evaluation. However, the difference between a score of 670 and a score of 615 is not that big, and perhaps future experiments that might build on the idea of this experiment can give better results if a smaller or larger group intervals is used than the 15 minutes window.

### 7.2 Mean For Outliers Regressor Experiment

The idea behind this experiment is the assumption that predicting the mean of the outliers in delays and early departures is satisfying enough. The reason for this is that airline companies are often considerate about large delays as most of them are obligated by customer low to provide accommodation, food and transport services if the delay is 3 hours or above. For this reason these operators companies are mostly interested in knowing if the delay is likely to last three or more hours without needing to know the exact number of minutes a flight might be delayed.

One way to build a model that take this into consideration is to edit the training dataset, to set each delay that has a value equal or higher than 180 minutes to the mean of all delay values higher than

179 minutes in the entire dataset. Also on the other early departures end, departures earlier than -15 minutes from scheduled time of departure are all set to the mean of all early departures for more than 15 minutes before the scheduled time to depart. While all other delay values in between -15 and +180 minutes are kept as they are. This also requires us to edit the evaluation and testing datasets as we are know interested in predicting the actual delay value if between -15 and 180 minutes and the mean of all delay values if lower than -15 or from 180 and above.

Implementing this setup and running an evaluation using the training dataset gives some promising results as it reduces the Mean Squared Error by around 30% from the previous Hybrid Regressor Experiment, while keeping the Huber Loss score equal to the tuned regressor using the DEP_DELAY as a target. A decrease of the Mean Absolute Error does also show improvement in this experiment.

Table 4 summarizes the results of each experiment, while more details about the distribution of errors across evaluation data are plotted in the notebooks of each experiment in the source code of this project.

Table 4. Experimentation Results using the tuned XGBoost regressor models of DEP_DELAY and DEP_DELAY_GROUP

| Regression Target | Hybrid | Mean For Outliers |
|---|---|---|
| Evaluation Metric | - | - |
| Mean Absolute Error | 19 | 16 |
| Mean Squared Error | 2916 | 2067 |
| Huber Loss Score | 670 | 615 |

In the next section, these experiments are put into testing using new data about flights from the year 2023 and study their effect on the distribution of errors for predicting on-time performance of future flights.

## 8 Test Analysis

After evaluating our experiments on the training dataset from the year 2022 using the 2-fold cross validation, it is time to see how these regression models would perform on new unseen data in the future. In this section, two regression models are trained on the entire dataset from the year of 2022 by implementing the two experimental setups of hybrid and mean for outliers respectively to each experiment testing.

Figure 5 show the distribution of absolute prediction errors from ground truth for the entire year of 2023 made by the regressor from the hybrid experiment; the percentiles on the x-axis show that the model was $\pm$10 minutes off from the ground truth in 65% of the predictions, and between +10 and +79 minutes in 25% of these predictions. This leaves 5% for relatively large errors in under estimation at -161 minutes at minimum and another 5% for relatively large errors in over estimation at +4425 minutes in errors at maximum. Although the regressor model performs good in estimation for most of the predictions keeping the predicted value within one hour of the ground truth, it still suffers from large outliers in the dataset which can be seen through large errors at the start and end 5% of the errors.

The Mean For Outliers experiment attempt to find a solution for this outlier problem by setting or delay value that is larger than or equal to 180 minutes to the mean value of those outliers delays equal to or larger than 3 hours, and similarly setting the early departures lower than -15 minutes to the mean values of those early departures smaller than -15 minutes. This off course builds on the idea that outliers are extreme values that occurs randomly with no clear reason known, and that setting their values to the mean of the outliers is fair and satisfying enough which is more practical for aviation purposes where these regression models are to be used. Figure 6 shows how the maximum error drops from 4425 in the hybrid experiment testing to 367 for the mean For outliers experiment testing, while we notice a little increase of errors on the the early departures end that is less important for the purpose of this on-time performance model. Otherwise the error picture in between outliers is as good as it was in the Hybrid Experiment with some small relocation of percentile values.

Furthermore, and in Table 5 we can view the scores of Mean Absolute Error, Mean Squared Error and Huber Loss to make a general comparison on the estimation accuracy of each model. The Mean For Outliers model gives us a better handling for outliers which is observed by the low Mean Squared Error compared to the other two models, while still give the same Mean Absolute Error as a model trained directly on the continuous value of delay in minutes. These two models also share the same Huber Loss Score. while the model trained on the 15 minutes interval seem to be outperformed by these two models.

Table 5. Testing Results of Hybrid and Mean For Outliers Experiments vs. the direct DEP_DELAY based XGBoost regressor model

| Model | Hybrid | Mean for Outliers | DEP_DELAY based |
|---|---|---|---|
| Metric | - | - | - |
| Mean Absolute Error | 20 | 18 | 18 |
| Mean Squared Error | 3298 | 2323 | 3160 |
| Huber Loss Score | 714 | 668 | 667 |

Choosing which model to use in production depends on the purpose of estimation, for example; if it is satisfying enough for us to estimate the mean of long delays (higher than 3 hours) or early departures (lower than -15 minutes), then going for the Mean for Outliers model is the most logical choice, but if we have a need to predict the exact number of minutes in delays or early departures then a model based on the DEP_DELAY target column is the most beneficial in this case.

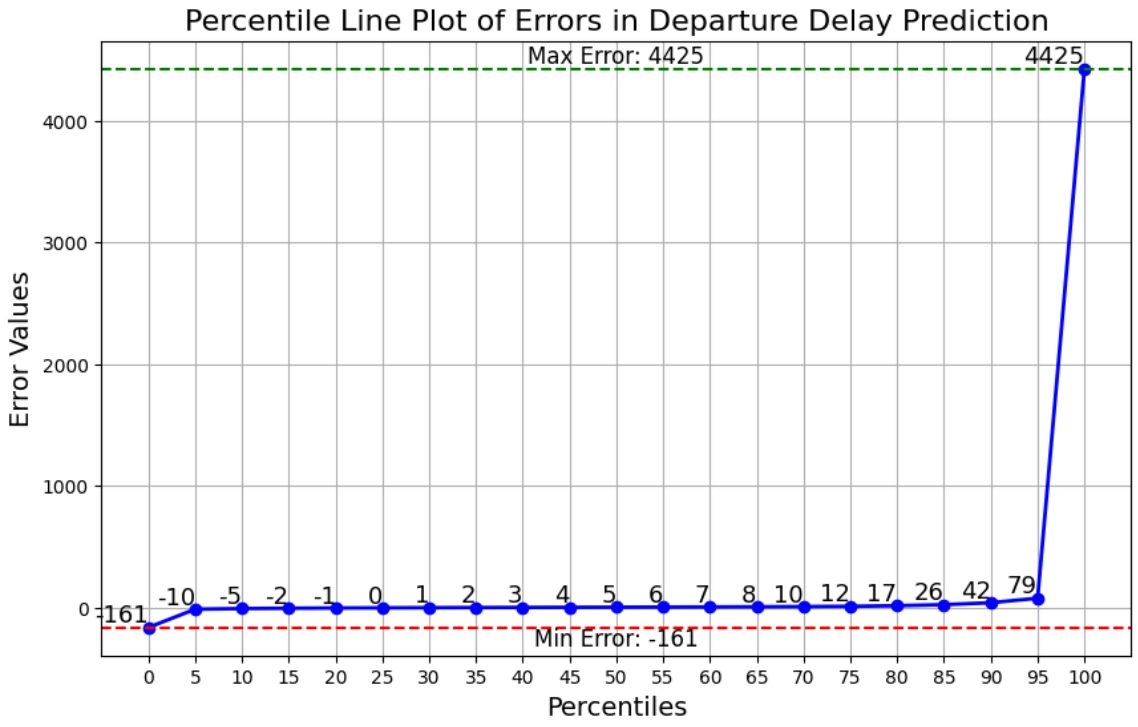


Fig. 5. Prediction errors percentiles vs. their values using The Regression Model From the Hybrid XGBoost experiment

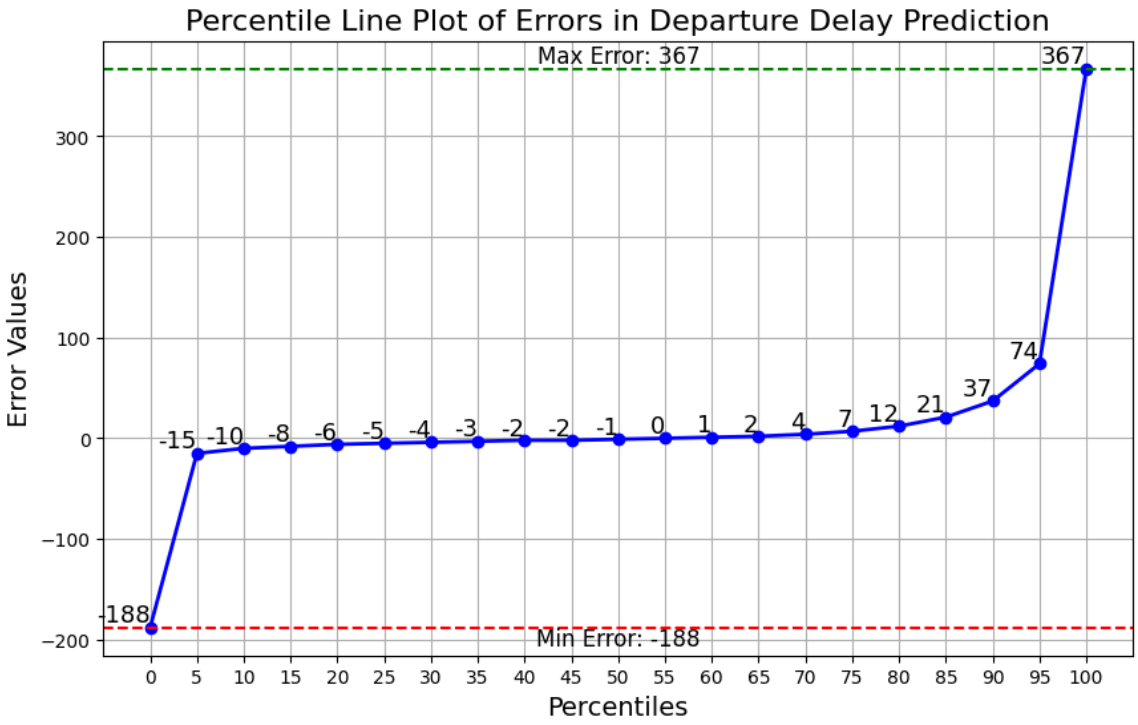


Fig. 6. Prediction errors percentiles vs. their values using The Regression Model From the Mean For Outliers XGBoost experiment.

## 9 Source Code

This research project is based entirely on jupyter notebooks using the Python Programming Language [4] with some of its popular frameworks such as pandas for data handling [3] and matplotlib [5] with seaborn [6] for visualizations. Viewing these notebooks offers a deeper insight into the actual implementation of experiments and their evaluation along with other steps such as preprocessing, visualization and encoding. The notebooks along with setup instructions on local environment are available on GitHub in the repository referenced in [7].

The reason this project is based on notebooks is because of its nature as a research project. If considering to run a production environment or testing the solution over a longer period of time, using frameworks such as MLflow might be a better way of implementing such solution.

## 10 Future Improvements

The previous experiments and analysis in this research shows that this problem is rich with interesting challenges to deal with, and a great potential for improvements for more accurate prediction of on-time performance of future flights. Some of these are about trying another machine learning algorithm or even another deep learning algorithm from for example a popular framework such as tensorflow to see if that might give significant improvements than XGBoost can give in this research.

Another measure that might improve the performance of this model is feeding more data into this model, and carefully watching its performance, and since the data source [1] offers data from as early as 1987, this might be very beneficial to improve the model performance. However it is important to keep the running time of model training and most importantly prediction under control by using tuning and removing noisy data or data that does not meet the requirements of this research if found, or also using any additional data from other sources such as weather data if such data improve the errors of the regression model.

A more experimental improvement might be to build a classification model that tries to isolate the outliers in the data, and after that building two models; one regressor for estimating outliers, and another one to predict the most central data across the distribution. Each model should then be trained on the part of data points that offers information to serve the purpose of each model. This means that the model for outliers should only be trained on those data points in the dataset that is classified as outliers. This idea might sound complicated but it sure worth being put into an experiment that evaluate its performance in the future.

## 11 Acknowledgments

Great thanks to the University of Stavanger for offering the highly relevant course named "Artificial Intelligence for Engineers" to engineers across many industries, and many thanks to the professor in Computer Science at the University of Stavanger (UiS) Mina Farmanbar for her valuable teaching and sharing of knowledge throughout the course.

## References

[1] US Bureau of Transportation Statistics. (2024, November 22). Reporting Carrier On-Time Performance (1987-present). TranStats. Retrieved from https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=FGJ
[2] XGBoost Documentation. (2024, November 22). Retrieved from https://xgboost.readthedocs.io/en/stable/
[3] pandas documentation. (2024, November 22). Retrieved from https://pandas.pydata.org/docs/
[4] Python 3.13.0 documentation. (2024, November 22). Retrieved from https://docs.python.org/3/
[5] Matplotlib 3.9.2 documentation. (2024, November 24). Retrieved from https://matplotlib.org/stable/index.html
[6] seaborn: statistical data visualization. (2024, November 24). Retrieved from https://seaborn.pydata.org/
[7] Prediction-of-On-time-Flight-Performance. (2024, November 24). Retrieved from https://github.com/MohammedGuniem/Prediction-of-On-time-Flight-Performance