Flight Delay Prediction Report 2024

Big Data Assignment

Authors: Ádám Földvári, Máté Lukács, José Antonio Ruiz Heredia

Table of contents

Introduction

The primary objective of this project is to develop a predictive model capable of estimating the arrival delay time of commercial flights based on parameters available at the time of takeoff. This predictive capability is crucial for enhancing operational efficiency in the aviation industry, minimizing passenger inconvenience, and improving scheduling reliability. To achieve this, publicly available data from commercial domestic flights within the United States was used for model development and testing.

The dataset includes a wide range of features such as flight times, carrier information, and route details, all of which influence flight delays. A machine learning approach was adopted, with a focus on creating a reliable and interpretable model capable of forecasting arrival delays accurately. The work culminates in the development of a Spark-based application programmed to load the model and evaluate its performance on unseen data.

The development process involves feature selection, data preprocessing, model training, and validation. Exploratory data analysis was conducted to identify significant features and exclude variables with limited predictive power. Additionally, the data underwent multiple transformations, including feature engineering and encoding, to enhance model performance. The final model evaluation was based on a comprehensive set of metrics, balancing predictive accuracy and interpretability for practical use in real-world scenarios.

Variables Selected for the Model

The variables included in the flight delay prediction model were carefully chosen after thorough exploratory data analysis, correlation assessments, and feature engineering strategies applied during the preprocessing phase. Each variable was selected based on its potential contribution to predicting flight delays and its statistical significance observed in the dataset.

Time-related Features:

- **Month:** Included to capture seasonal variations and delay trends that occur during specific months, such as increased delays during holiday seasons or weather-impacted months.
- **DayOfWeek:** Selected due to weekly patterns in air travel, where certain days, such as Mondays and Fridays, often show higher congestion and delay probabilities.
- **DepTime_Hour and DepTime_Minute:** Created by transforming the original departure time into separate components, providing finer granularity for time-based delay patterns. This transformation allows the model to capture delay variations more effectively during peak and off-peak hours.
- **TimeOfDay_encoded:** A derived categorical variable that groups departure times into periods such as morning, afternoon, evening, and night, based on observed delay trends in the dataset. This feature simplifies temporal analysis and improves interpretability.

Flight-specific Features:

- **Distance:** Represents the physical distance between origin and destination airports. Longer flights may have different delay patterns compared to shorter flights due to air traffic congestion and longer operational processes.
- **DistanceCategory_encoded:** A categorical transformation of the distance variable, segmenting flights into short, medium, and long-haul categories based on observed delay patterns.
- **CRSElapsedTime:** The scheduled duration of the flight. Longer scheduled durations may indicate potential delay risks due to complex flight paths and operational dependencies.
- **UniqueCarrier_encoded:** Represents the airline operating the flight, capturing airline-specific patterns observed during EDA where some carriers demonstrated consistently different delay behaviors.
- **DepDelay:** The departure delay is a crucial predictor as it directly correlates with arrival delays. Its strong positive correlation with arrival delay justified its inclusion as a primary feature.

Route Information:

- **Origin_encoded and Dest_encoded:** Encoded airport codes capturing geographic and operational patterns specific to certain airports.
- **RouteFrequency:** A derived feature representing how frequently a route is flown based on historical data. High-frequency routes tend to have more reliable schedules, while less frequent routes showed higher variability in delays.

The inclusion of these variables was driven by both domain knowledge and statistical patterns observed during exploratory analysis. Each variable was either a direct observation from the data or a transformation that increased predictive power while ensuring minimal redundancy.

Variables Excluded from the Model

Several variables were excluded from the model after careful consideration and analysis due to low predictive power, redundancy, or data quality issues.

Variables Removed Due to Limited Predictive Power:

- **Year:** Found to be less informative for predicting delays as patterns observed were consistent across multiple years, and using it could lead to overfitting.
- **DayofMonth:** Analysis showed that weekly patterns (captured by DayOfWeek) were more influential in predicting delays than the specific day of the month.

Variables Removed Due to Redundancy:

- **CRSDepTime/CRSArrTime:** These scheduled departure and arrival times were excluded after transforming DepTime into DepTime_Hour and DepTime_Minute, making them redundant.

Variables Removed Due to High Cardinality or Incomplete Data:

- **FlightNum:** The flight number provided little predictive value as it is unique for each flight and does not capture broader delay patterns.
- **TailNum:** The aircraft registration number was excluded due to high cardinality and insufficient data consistency.
- **TaxiOut:** Removed because a significant portion of values were missing, reducing its reliability for predictive modeling.

Variables Excluded Based on Data Cleaning Criteria:

- **CancellationCode:** This variable was excluded as cancelled flights were filtered out during data preprocessing.

The decision to exclude these variables was made to maintain a balance between model complexity and performance, ensuring the inclusion of features with meaningful contributions to delay predictions.

Description and Justification of the Variable Transformations Performed

Several variable transformations were applied to enhance the model's predictive capacity and ensure compatibility with the machine learning pipeline used:

Time-Based Transformations:

- **DepTime Transformation:** The original departure time variable was split into **DepTime_Hour** and **DepTime_Minute** to better capture delay patterns across time blocks.
- **TimeOfDay Encoding:** Created by segmenting flight times into categories such as morning, afternoon, evening, and night to better represent delay patterns observed during data exploration.

Categorical Feature Encoding:

- **UniqueCarrier, Origin, Dest:** These categorical features were encoded using **StringIndexer** and **OneHotEncoder** to convert them into numerical representations suitable for machine learning algorithms.

Distance Transformation:

- **DistanceCategory:** A categorical variable derived from **Distance** to group flights into short, medium, and long-haul categories. This was justified by observed variations in delay patterns across different flight lengths.

Derived Features:

- **RouteFrequency:** Calculated by counting the historical frequency of each unique flight route in the dataset. This feature was derived based on the hypothesis that more frequently traveled routes would exhibit lower delays due to operational efficiency.

The transformations were necessary to reduce data sparsity, improve feature scalability, and optimize the model's ability to learn delay patterns effectively.

Description and Justification of the New Variables Created

Several new variables were derived to improve the model's performance and enhance predictive capacity. These variables were created based on domain knowledge, statistical analysis, and observed patterns in the dataset.

- **DepTime_Hour and DepTime_Minute:** Splitting the departure time into hours and minutes provided finer granularity for capturing peak and off-peak delay patterns.
- **TimeOfDay_encoded:** Categorizing time periods into morning, afternoon, evening, and night allowed the model to better capture delay variations across different times of the day.
- **DistanceCategory_encoded:** Transforming the continuous distance variable into categories (short, medium, long) provided better delay pattern segmentation based on flight length.
- **RouteFrequency:** This feature was created by calculating the historical frequency of a given route, representing its operational reliability and delay risk based on historical data.

The creation of these new variables was based on improving the interpretability and performance of the predictive model by providing meaningful segmentations and reducing noise in the dataset.

Applied Machine Learning Technique and Its Parameters

The machine learning technique applied involved both **Linear Regression** and **Decision Tree Regressor** for modeling flight delays. Linear Regression was initially selected for its simplicity and effectiveness in modeling linear relationships between input features and the target variable. However, during model evaluation (**with data from 1993**), the **Decision Tree Regressor** outperformed Linear Regression.

The **Decision Tree Regressor** was chosen as the final model due to its superior performance metrics observed during evaluation, including:

- **Lower RMSE:** Decision Tree achieved a lower Root Mean Square Error (14.29) compared to Linear Regression (14.66), indicating a better fit to the data.
- **Higher R^2 Score:** Decision Tree also achieved a higher R^2 score (0.612 vs. 0.592), suggesting a stronger correlation between predicted and actual values.

Key parameters included:

- **Tree Depth Control:** The depth of the tree was adjusted to balance underfitting and overfitting.
- **Min Data Per Leaf:** Controlled the minimum number of data points in each leaf to avoid overly complex models.
- **Feature Scaling:** Ensured numerical inputs were standardized to prevent biases due to differing units of measurement.

The Decision Tree Regressor was ultimately chosen because it effectively handled the non-linear relationships present in the dataset and provided a balance between accuracy and interpretability.

Validation Process

The validation process involved splitting the dataset into **training** and **testing** subsets, with 80% allocated to training and 20% for testing. This standard approach ensures the model generalizes well to unseen data.

Metrics used to evaluate model performance included:

- **Root Mean Square Error (RMSE):** Measures average prediction error, balancing larger errors with squared penalties.
- **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual delay values.
- **R^2 Score:** Indicates how well the model explains the variance in the target variable.
- **15-minute Accuracy:** Practical metric assessing how often predictions fall within 15 minutes of the actual delay.
- **Severe Delay Accuracy:** Measures performance specifically on significant delay events (over 60 minutes).

These metrics were chosen for their ability to balance error measurement, business relevance, and explainability.

Final Evaluation of the Prediction Model

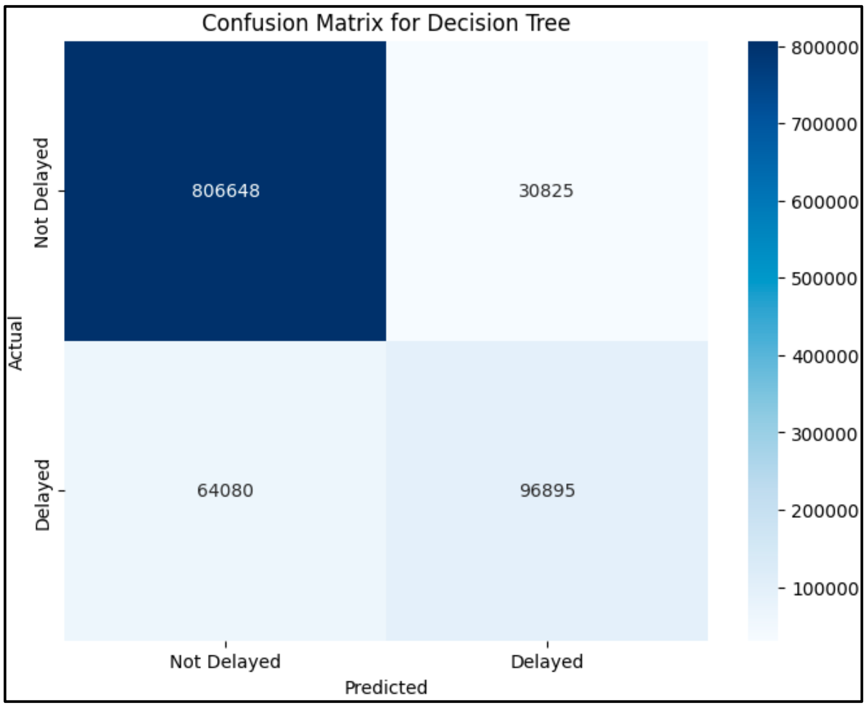
The final evaluation of the decision tree regression model was based on several performance metrics obtained during validation:

- **Accuracy:** 90.5% - Indicating a high percentage of correct predictions.
- **Precision:** 75.9% - Reflecting a strong ability to correctly identify delayed flights.
- **Recall:** 60.2% - Showing moderate sensitivity in capturing all actual delay instances.
- **F1 Score:** 67.1% - Balancing precision and recall.

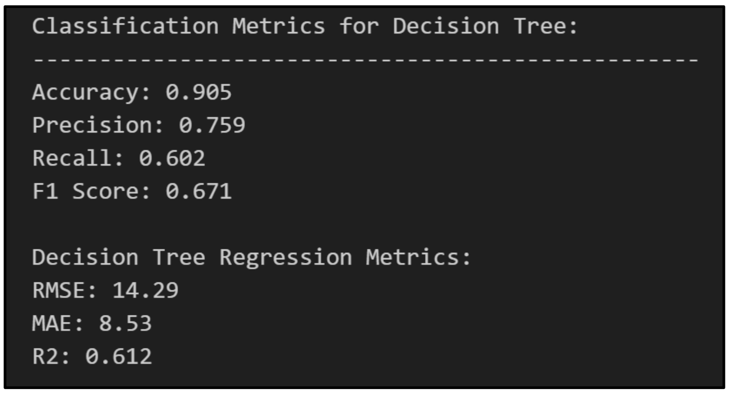
Regression Metrics:

- **RMSE:** 14.29 minutes - Representing the average magnitude of error in minutes.
- **MAE:** 8.53 minutes - Highlighting the average absolute deviation from the actual delay.
- **R²:** 0.612 - Indicating moderate model fit and the ability to explain the variance in flight delay data.

The confusion matrix showed effective classification performance, with most instances correctly classified as delayed or not delayed, but some false positives and false negatives still present. Overall, the Decision Tree model offered a balance between performance and interpretability for flight delay prediction.



Confusion Matrix for Decision Tree (notebook)



```
Classification Metrics for Decision Tree:
--------------------------------------------------
Accuracy: 0.905
Precision: 0.759
Recall: 0.602
F1 Score: 0.671

Decision Tree Regression Metrics:
RMSE: 14.29
MAE: 8.53
R2: 0.612
```

Metrics for Decision Tree (notebook)

Final Conclusions and Personal Remarks

Developing this flight delay prediction model has been an insightful experience for our entire team, combining both theoretical and practical aspects of machine learning. We approached this task collaboratively, ensuring each stage, from data preprocessing to model evaluation, was handled with thorough analysis and continuous feedback. Our teamwork allowed us to cover all aspects of the project effectively, enhancing the final outcome.

Throughout the process, we recognized the importance of rigorous data preprocessing and thoughtful feature selection. The inclusion of features like DepDelay and RouteFrequency significantly boosted model accuracy, while the exclusion of redundant and incomplete data helped simplify the model and reduce overfitting risks. This reinforced our understanding that cleaner, well-engineered data often leads to better model performance than relying solely on complex algorithms.

We also learned the value of balancing multiple performance metrics. While accuracy and RMSE were central to our evaluations, metrics like precision and recall helped us assess the model's capability in predicting both minor and severe delays accurately. This multi-metric approach gave us a clearer perspective on how the model performs across varying conditions.

Working on this project as a team has strengthened our skills in collaborative machine learning development. We've gained deeper insights into Spark's capabilities for handling large datasets and its suitability for scalable data science tasks. This experience has not only expanded our technical knowledge but also emphasized the importance of clear communication and shared decision-making in a data science project.

Looking ahead, we aim to explore further improvements by experimenting with advanced models like Gradient Boosted Trees and Random Forest. Additionally, we believe incorporating external datasets, such as weather data and air traffic reports, could enhance the model's predictive power and overall reliability. This project has provided us with a strong foundation for future endeavors in predictive modeling and data science collaboration.

Instructions on how to pass the test data path to the application

To execute the **app.py** application and pass the test data path as an argument, use the following command (terminal):

spark-submit --master local[*] app.py "[test_data_path]"

Explanation:

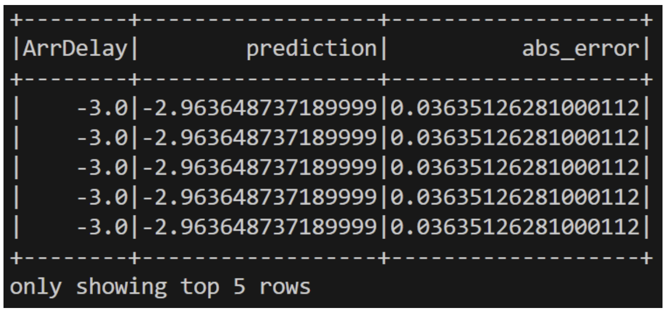
- **spark-submit**: This is the command used to submit a Spark application.
- **--master local[*]**: This specifies that the application should run locally using all available CPU cores.
- **app.py**: This is the Python script containing your Spark application.
- **"[test_data_path]"**: Replace [test_data_path] with the actual path to your test data file. Make sure to include the full file path in quotes if it contains spaces or special characters.

Additional Note for Model Training in the Notebook:

When training the model in the notebook, we used an interactive method to load the dataset. The user needs to provide the file path when prompted during execution (without quotes). Simply enter the full path of your dataset (e.g.: C:\Users\25fad\Documents\Msc\Big Data\Assigment\training_data\1993.csv) into the input field when requested. Ensure that the file exists at the specified location and that the path is correct. If an error occurs, double-check and re-enter the correct file path.

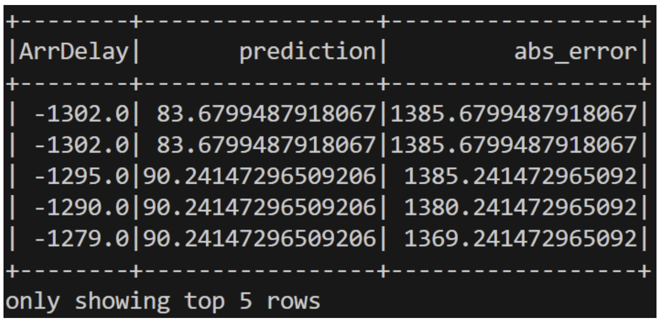
Screenshots about the result after the successful run of **app.py**:

The three images summarize the performance of **best_model** (now it is: DecisionTreeRegressionModel) used to predict flight arrival delays with **app.py** on a test data (1987.csv on the pictures).



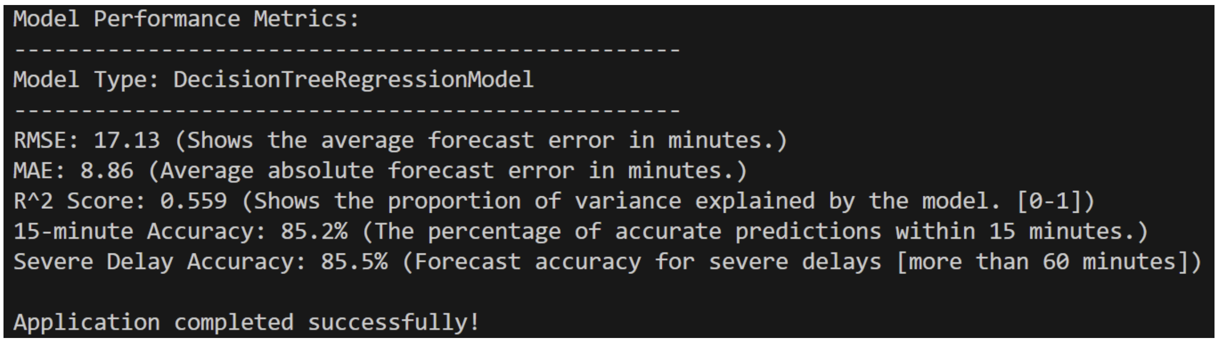
```
+--------+------------------+-------------------+
|ArrDelay|        prediction|          abs_error|
+--------+------------------+-------------------+
|    -3.0|-2.963648737189999|0.03635126281000112|
|    -3.0|-2.963648737189999|0.03635126281000112|
|    -3.0|-2.963648737189999|0.03635126281000112|
|    -3.0|-2.963648737189999|0.03635126281000112|
|    -3.0|-2.963648737189999|0.03635126281000112|
+--------+------------------+-------------------+
only showing top 5 rows
```

Showing 5 best predictions



```
+--------+-----------------+------------------+
|ArrDelay|       prediction|         abs_error|
+--------+-----------------+------------------+
| -1302.0| 83.6799487918067|1385.6799487918067|
| -1302.0| 83.6799487918067|1385.6799487918067|
| -1295.0|90.24147296509206| 1385.241472965092|
| -1290.0|90.24147296509206| 1380.241472965092|
| -1279.0|90.24147296509206| 1369.241472965092|
+--------+-----------------+------------------+
only showing top 5 rows
```

Showing 5 worst predictions



```
Model Performance Metrics:
--------------------------------------------------
Model Type: DecisionTreeRegressionModel
--------------------------------------------------
RMSE: 17.13 (Shows the average forecast error in minutes.)
MAE: 8.86 (Average absolute forecast error in minutes.)
R^2 Score: 0.559 (Shows the proportion of variance explained by the model. [0-1])
15-minute Accuracy: 85.2% (The percentage of accurate predictions within 15 minutes.)
Severe Delay Accuracy: 85.5% (Forecast accuracy for severe delays [more than 60 minutes])

Application completed successfully!
```

Displaying which model was the best and showing the performance metrics