Artificial Intelligence – Semester Project

Data Analysis and Application of a Selected Machine Learning Method

Anhelina Rudzenka rudzeanh@fel.cvut.cz

January 12, 2026

Google Colab Link

GitHub Link

1 Introduction

This project focuses on predicting flight prices using machine learning. The goal is to build a model that can estimate how much a flight will cost based on features like source and departure cities, airline and flight class.

I chose to use Random Forest Regression as machine learning method, because it can handle different types of data (numbers and categories) and usually gives good results without needing complicated setup.

I used a dataset of 300153 flight records from the India flight market in 2022. This dataset includes information about 6 airlines, 6 cities, and different flight classes (economy and business). I chose this dataset because:

- It's large enough to train a model.
- It contains real flight data.
- The features are easy to understand

This documentation is organized as follows:

- Section 2 - Dataset Description: Details about the data, including features and basic statistics.
- Section 3 - Problem Definition: Explanation of the prediction task and chosen machine learning method.
- Section 4 - Data Preprocessing: How I cleaned and prepared the data for machine learning.
- Section 5 - Machine Learning Method: Description of the Random Forest algorithm and its parameters.
- Section 6 - Experiments and Results: Describes machine learning models I built, how I evaluated them, and the final results.
- Section 7 - Discussion: What worked well, what didn't work, and ideas for future improvements.
- Section 8 - Conclusion: Summary of the project and key findings
- Appendices: Additional plots and data profiles used in the analysis.

The goal of this project is to demonstrate practical machine learning skills by solving a real-world prediction problem from start to finish.

2 Dataset Description

Dataset Name: Flight Price Prediction
Source: Kaggle
URL: Kaggle Dataset Link

This dataset contains information about flight booking options from the website "Ease My Trip" for travel between India's top 6 metro cities. The data shows different flight options with their prices, airlines, departure times, and other details. Data was collected for 50 days, from February 11th to March 31st, 2022.

The original dataset includes 3 CSV files:

- **business.csv** – 93487 business class flight records
- **economy.csv** – 206774 economy class flight records
- **Clean_Dataset.csv** – 300153 flight records (combined and cleaned dataset)

For this project, I used the **Clean_Dataset.csv** file because it combines both business and economy class flight records into a single dataset and has already been cleaned, with 108 invalid rows removed.

The dataset consists of approximately 69% economy class flights and 31% business class flights, indicating a moderate imbalance toward economy class bookings that reflects real-world travel patterns.

The dataset has 11 columns that describe each flight:

- **airline** – Airline company name (6 airlines: SpiceJet, AirAsia, Vistara, GO_FIRST, Indigo, Air_India).
- **flight** – Flight code or flight number (1561 unique flight codes).
- **source_city** – Departure city (6 cities: Delhi, Mumbai, Bangalore, Kolkata, Hyderabad, Chennai).
- **departure_time** – Time of day when the flight departs (6 time periods: Early_Morning, Morning, Afternoon, Evening, Night, Late_Night).
- **stops** – Number of stops during the flight (3 categories: zero, one, two_or_more).
- **arrival_time** – Time of day when the flight arrives (Same six time periods as the departure time).
- **destination_city** – City where the flight arrives (Same six cities as the source city).
- **class** – Type of ticket (2 classes: economy, business).
- **duration** – Total flight duration in hours (Range: 0.83 to 49.83 hours, with 476 distinct values).
- **days_left** – Number of days remaining until departure (Range: 1 to 49 days).
- **price** – Flight ticket price in Indian Rupees (INR).

There are three numerical features (`duration`, `days_left`, `price`), while all other features are categorical text data.

During the exploratory analysis, I found the following key points:

- The mean ticket price is 20889.66, while the median price is 7425.00. This shows that the price distribution is highly skewed, with some very expensive tickets that increase the average price.
- Most passengers travel in the economy class.
- Most flights have one stop.
- There are very few late-night flights.

During the identification of dataset issues, I found the following:

- The dataset is fully complete, with no missing values in any of the columns.
- The data do not contain significant outliers. Less than 1% of the observations are outliers.
- None of the numerical features (`duration`, `days_left`, `price`) are highly correlated with each other.
- There is a strong class imbalance in the `flight` and `departure_time` columns. The imbalance ratios are approximately 54.48:1 for `departure_time` and 3235:1 for `flight`.
- For the `departure_time` column, morning flights account for 24% of the data, while late-night flights represent only 0.4%. This suggests that people tend to avoid late flights.
- For the `flight` column, this likely means that some flights operate much more often than others.

The plots used for the exploratory analysis and dataset issues identification are provided in the appendix. I also generated a data profile using `ydata-profiling`, which is also included in the appendix.

3 Problem Definition

The problem can be formulated as follows: given information about a flight, we want to predict the ticket price in Indian Rupees (INR).

This is a supervised learning problem, because the dataset contains labeled data (flights with known prices). More specifically, this is a regression task, because we predict the price, which is a continuous numerical value.

For this project, I chose to use Random Forest Regression as the main prediction algorithm. Random Forest Regression builds many decision trees and combines their predictions to make a strong model. This choice is justified by several reasons.

First, works well with both numerical and categorical features, which matches the dataset.

Second, Random Forest can capture non-linear relationships. The relationship between features and price is not always linear.

Fourth, Random Forest handles feature interactions automatically. The price may depend on combinations of features, and the model discovers these interactions during training without manual specification.

Finally, the model is less prone to overfitting compared to a single decision tree.

4 Data Preprocessing

During the exploratory data analysis in the previous chapter, I checked the dataset for common problems. Here, I describe the issues found and how they were addressed.

Outliers
Issue: Some flights had unusually long durations or very high prices.
Action: I kept all outliers in the dataset. This is because Random Forest models handle outliers well, and these extreme values represent real scenarios, such as multi-stop journeys or business class tickets peak times. Removing them could discard useful information about expensive flight options.

Irrelevant Features
Issue: The `Unnamed: 0` column was only an index. The `flight` column contained flight codes, which had too many unique values and mostly reflected historical data, not useful for learning general price patterns.
Action: Both columns were removed. This reduced noise and made the data simpler for the model.

I applied the following preprocessing steps:

1. **Feature Selection:** The `Unnamed: 0` and `flight` columns were removed as they were irrelevant.
2. **Separating Features and Target Variable:** The dataset was split into features (X) and the target variable (y). The target variable was the `price` column, while X contained all other columns used for prediction.
3. **Encoding Categorical Data:** All categorical features were converted from text labels to numbers using label encoding. Machine learning algorithms, including Random Forest, require numerical input.
4. **Train-Test Split:** The dataset was divided into training and testing sets. The training set contained 80% of the data for model learning, and the test set contained 20% to evaluate model performance on unseen data.

Each preprocessing step was chosen to ensure that the data would be suitable for Random Forest regression, which works best with clean, relevant, and numerical data.

5 Machine Learning Method

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. Instead of relying on a single decision tree, it builds many trees and averages their results to get a final prediction.

The algorithm creates multiple random subsets of the training data. Each subset is created by randomly selecting samples with replacement (some samples can appear multiple times, others not at all).

For each data subset, a decision tree is built. However, when splitting nodes in the tree, only a random subset of features is considered at each split point. This makes the trees different from each other.

When we want to predict a value for new data, all trees make their own predictions. The final prediction is the average of all tree predictions.

Mathematically, if we have N decision trees in the forest, and each tree i makes a prediction $\hat{y}_i(x)$ for an input x, the final prediction $\hat{y}(x)$ is given by:

$$\hat{y}(x) = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i(x)$$

This averaging helps to reduce overfitting and improves generalization compared to a single decision tree.

Random Forest Regression has several important parameters that control how the model is built:

`n_estimators` This parameter defines the number of trees in the forest. A larger number usually improves performance, but it also increases training time. In practice, values between 100 and 300 are often used.

`max_depth` This parameter sets the maximum depth of each tree. Limiting the depth can prevent overfitting. However, if set too low, it may lead to underfitting.

`min_samples_split` This is the minimum number of samples required to split an internal node. Higher values prevent the tree from learning too specific patterns.

`min_samples_leaf` This parameter defines the minimum number of samples in a leaf node. It helps to smooth the predictions and reduce noise.

`max_features` This defines how many features are randomly selected at each split.

`random_state` This parameter is used to make the results reproducible. Setting it to a fixed number ensures that the same random choices are made each time the model is trained.

Random Forest structure is defined by a collection of independent decision trees. Each tree has a hierarchical structure with internal decision nodes and leaf nodes that store prediction values.

6 Experiments and Results

First, I created a baseline model to see how well Random Forest works with simple settings. The baseline model used the parameters shown in Table 1.

Table 1: Random Forest Baseline Model Parameters

Parameter	**Value**	**Description**
`n_estimators`	100	Number of decision trees
`max_depth`	None	Maximum depth of the trees
`min_samples_split`	2	Minimum samples required to split a node
`min_samples_leaf`	1	Minimum samples allowed in a leaf node
`max_features`	`sqrt`	Number of features considered at each split
`random_state`	42	Ensures reproducible results
`n_jobs`	-1	Uses all available CPU cores

The baseline model achieved strong performance on the test dataset:

- **Test R^2:** 0.9841 (98.41%)
- **Test MAE:** 1221.10 INR
- **Test RMSE:** 2859.63 INR

These results were already very good, but I wanted to see if I could improve them. I manually adjusted the model parameters to try to improve performance. The final tuned model used the parameters shown in Table 2.

I manually tested different parameters and compared the results. Then I compared the baseline and tuned models on the testing set, as shown in Table 3.

The tuned model is better on all four metrics, and therefore it was selected as the final model.

Important note: The training scores for the tuned model appear worse than those of the baseline model, which may seem suspicious. However, the

Table 2: Random Forest Tuned Model Parameters

Parameter	Baseline	Tuned	Why I changed it
`n_estimators`	100	200	More trees can make better predictions
`max_depth`	None	None	Kept unlimited (worked well)
`min_samples_split`	2	5	Prevent trees from splitting too easily
`min_samples_leaf`	1	2	Make predictions smoother
`max_features`	`'sqrt'`	0.5	Use 50% of features for more variety

Table 3: Comparison of Baseline and Tuned Models on the Testing Set

Metric	Baseline	Tuned	Improvement
R^2	0.9841	0.9856	+0.0014
Adjusted R^2	0.9841	0.9856	+0.0014
MAE (INR)	1221.10	1201.68	–19.42
RMSE (INR)	2859.63	2727.88	–131.75

testing performance is the key evaluation criterion. Since the testing scores improved, the tuned model generalizes better to unseen data, which is the primary objective.

I used five different metrics to measure how well the model works:

R^2 (R-squared)
Final value: 0.9856 (98.56%)
What it means: The model explains 98.56% of the differences in flight prices.
Interpretation: Good result. Anything above 90% is considered very good.

Adjusted R^2
Final value: 0.9856 (98.56%)
What it means: Same as R^2, but adjusted for the number of features.
Interpretation: The value is the same as R^2, which confirms the model is truly good and not just inflated by having many features.

MAE (Mean Absolute Error)
Final value: INR 1201.68
What it means: On average, predictions are off by INR 1201.68.
Interpretation: The average flight price is INR 20894, so an error of INR 1201.68 is 5.75% of the average price. This is acceptable for price prediction.

RMSE (Root Mean Squared Error)
Final value: INR 2727.88

What it means: Similar to MAE, but gives more weight to large errors.
Interpretation: This value is higher than MAE because it penalizes large mistakes more strongly. It indicates that some predictions are further off, but overall the model remains reliable.

MSE (Mean Squared Error)
Final value: INR 7441302.56
What it means: Average of squared errors.
Interpretation: This metric is difficult to interpret directly because it is expressed in squared rupees. However, lower values indicate better performance. The RMSE, which is the square root of the MSE, is easier to interpret.

After training the model, I also checked which features are most important for predicting flight prices. The resulting feature importance ranking is shown in Table 4.

Table 4: Feature Importance for Flight Price Prediction

Rank	Feature	Importance	Percentage
1	class	0.8238	82.38%
2	airline	0.0631	6.31%
3	duration	0.0514	5.14%
4	stops	0.0171	1.71%
5	days_left	0.0166	1.66%
6	source_city	0.0097	0.97%
7	destination_city	0.0096	0.96%
8	arrival_time	0.0047	0.47%
9	departure_time	0.0040	0.40%

Key findings:

- **Class** is by far the most important feature (82%), which is expected since business class tickets are significantly more expensive than economy tickets.
- **Airline** plays a notable role (6%), as different airlines apply different pricing strategies.
- **Duration** has a meaningful impact (5%), with longer flights generally costing more.
- **Departure time** and **arrival time** have minimal influence (less than 1%) on ticket prices in this dataset.

This ranking helps to identify the primary factors driving flight prices in the Indian market.

The plot comparing actual and predicted flight prices is provided in Appendix 8 (Figure 13).

What this plot shows:

- Each dot represents one flight
- X-axis: actual price of the flight
- Y-axis: price predicted by the model
- Red dashed line: perfect predictions (predicted price equals actual price)

Interpretation: Most points lie close to the red dashed line, indicating that the predictions are accurate. The points form a tight cluster around the diagonal, showing that the model successfully learned the underlying patterns in the data. There are some points that spread out (especially for expensive flights), but overall the fit is good.

The distribution of prediction errors is shown in Appendix 8 (Figure 14).

What this plot shows:

- How often the model makes different sized errors
- X-axis: prediction error (actual price minus predicted price)
- Y-axis: number of flights with a given error
- Green line: average error (approximately zero)

Interpretation: The histogram has a distribution centered around zero, which is the desired outcome. Most predictions have very small errors, as shown by the high bar near zero. Errors are balanced on both sides, indicating that the model does not systematically over-predict or under-predict.

The residual plot is included in Appendix 8 (Figure 15).

What this plot shows:

- Residuals (differences between actual and predicted prices)
- X-axis: predicted price
- Y-axis: residual (error)
- Red line: zero error
- Orange dotted lines: ±1 standard deviation (±INR 2728)

Interpretation: The plot shows mostly random scatter around the zero line, which is good. Some diagonal striping is visible. This is normal for Random Forest models because they make predictions by combining many trees, which creates discrete levels in the predictions. Errors are slightly larger for expensive flights. This is normal for price data because expensive flights naturally have more variation.

7 Discussion

Strengths of the Model

High Accuracy. The final model achieved an R^2 score of 98.56%, which means it can explain more than 98% of the variation in flight prices. This is a good result.

No Overfitting. The difference between training performance ($R^2 = 99.32\%$) and testing performance ($R^2 = 98.56\%$) is very small. This shows that the model learned real patterns instead of just memorizing the training data.

Weaknesses of the Model

One Feature Dominates. The class feature has 82% importance, which means the model relies heavily on just one piece of information. I expected departure and arrival times and cities to be more important. The dataset might not have enough variation in prices by time. Business class is expensive regardless of whether the flight departs in the morning or evening.

Larger Errors for Expensive Flights. Looking at the residual plot, I noticed that errors are bigger for expensive flights. The model is less accurate when predicting high prices. This happened because there are fewer expensive

flights in the dataset (most flights are economy class). The model has less training data for expensive flights, so it makes bigger errors for them.

What Worked Well and Why

Random Forest Algorithm. Random Forest worked well for this problem. It handled different types of features (numbers and categories) without problems and didn't need much tuning. It worked because Random Forest can capture complex relationships (for example, that business class flights to certain cities are more expensive), it is robust and doesn't overfit easily, and it works well with categorical features like airline and city names.

Label Encoding. I used Label Encoding to convert categorical features (like airline names) into numbers. This worked well and Random Forest could handle this type of encoding.

Possible Alternatives

Instead of one model for all flights, I could create two separate models: one for economy class flights and one for business class flights. This might help because each model could focus on the price variations within that class.

8 Conclusion

Overall, I'm satisfied with the results of this project. The Random Forest model achieved good accuracy (98.56% R^2) and works reliably on new data. The most important finding is that flight class dominates price prediction in the Indian market and this one factor explains 82% of price variation.

The main limitation is that the model relies too heavily on this one feature, which means it might not be very useful for comparing flights within the same class. If I continue this project, I would focus on collecting more diverse data (especially expensive flights) and creating separate models for each class.

I learned that sometimes simple approaches work best. I spent time on hyperparameter tuning and got only small improvements, while the baseline model was already very good. I also learned the importance of visualizations - the residual plot helped me understand where the model struggles (expensive flights).

This project gave me practical experience with:

- Real-world data preprocessing.
- Feature engineering.
- Model evaluation.
- Interpreting visualizations to understand model behavior.

Appendix: Dataset Description Plots

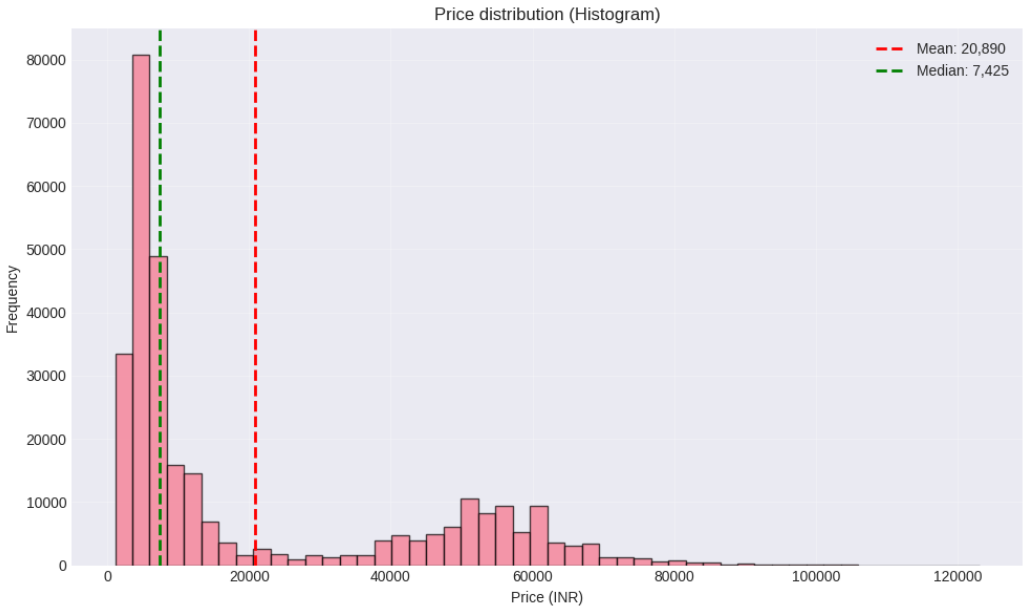

Figure 1: Price distribution of airline tickets.

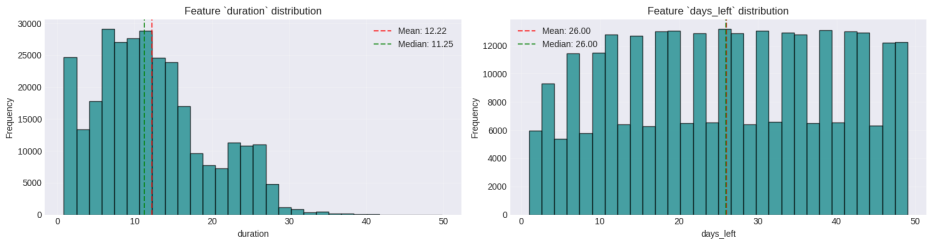

Figure 2: Other numerical attributes distribution.

Figure 3: Categorical attributes distribution.

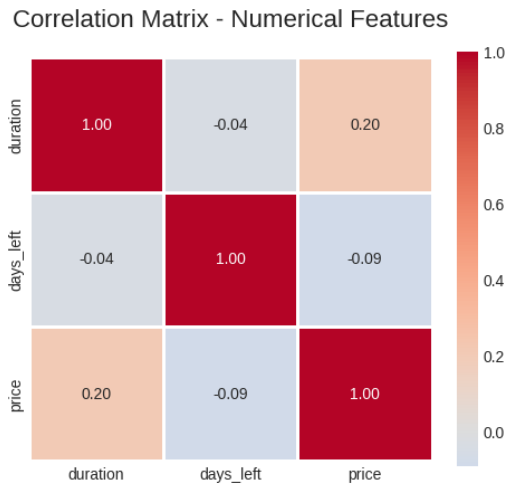

Figure 4: Correlation matrix.

Appendix: Data Profile

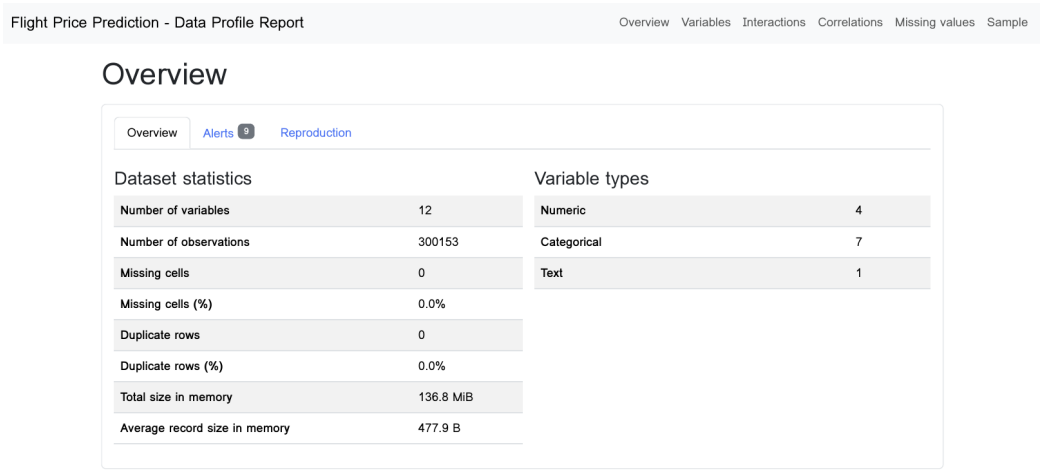

Figure 5: Data Profile Report - part 1.

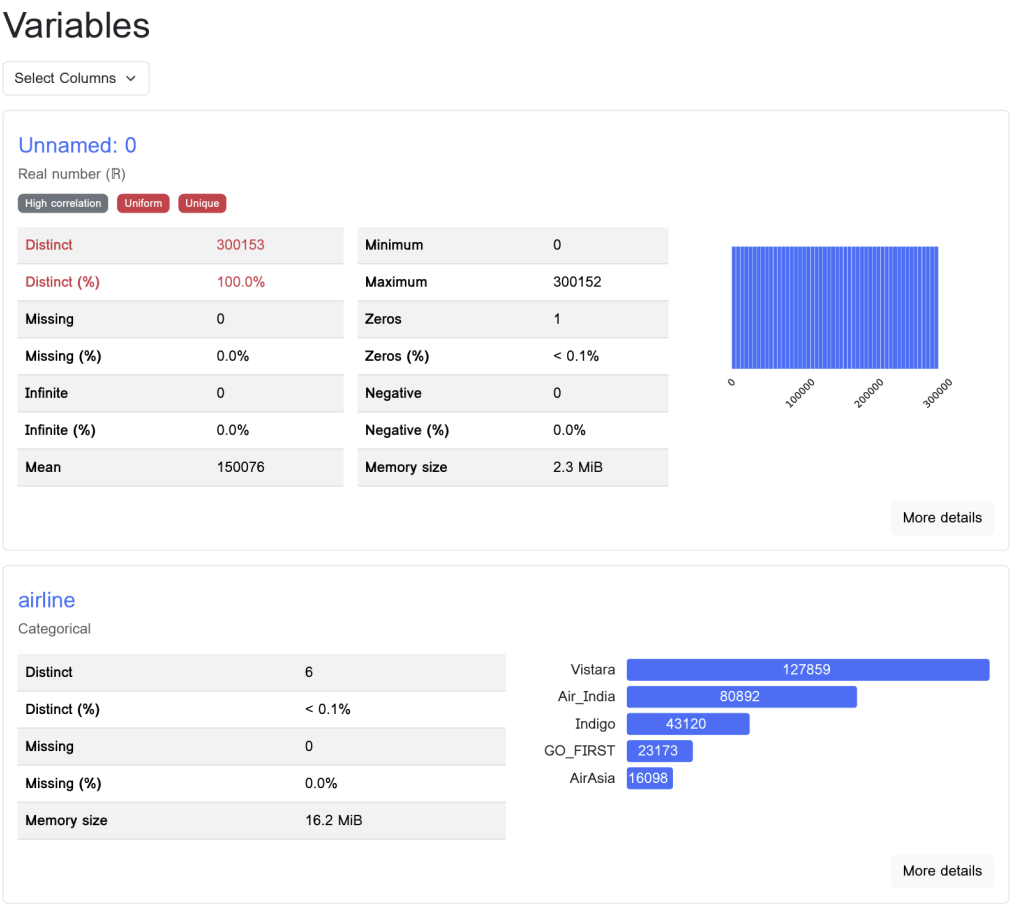

Figure 6: Data Profile Report - part 2.

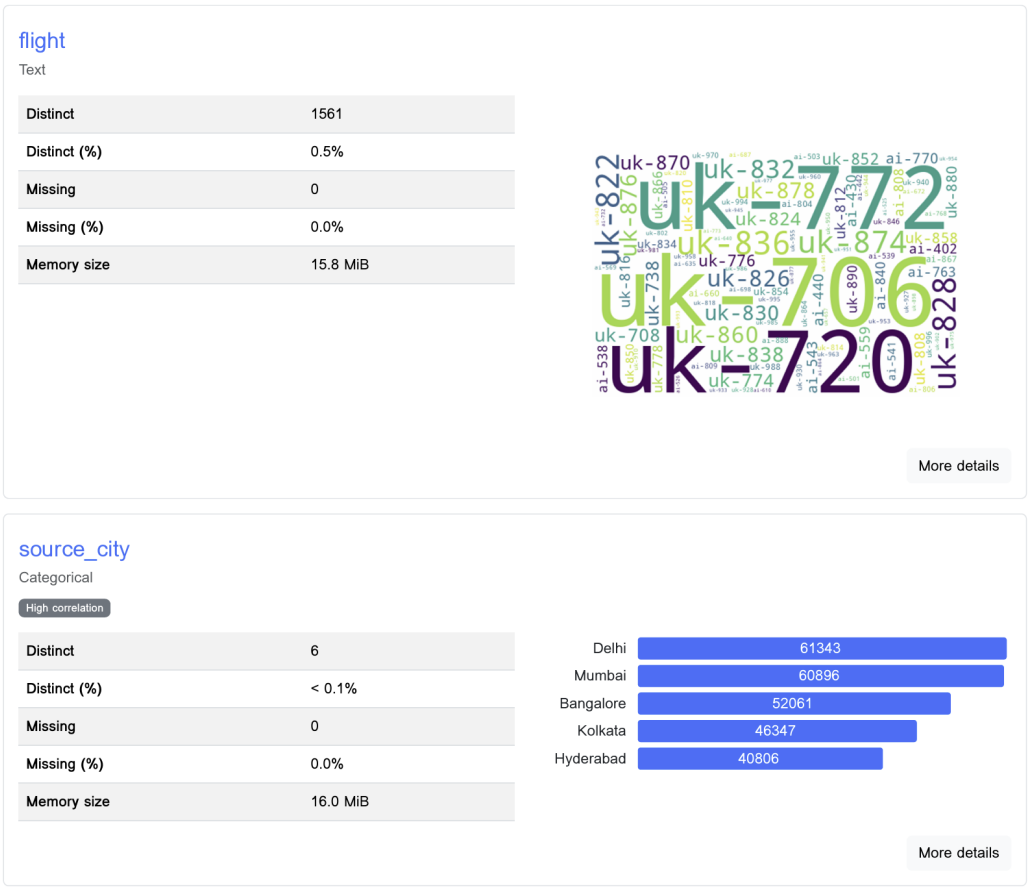

Figure 7: Data Profile Report - part 3.

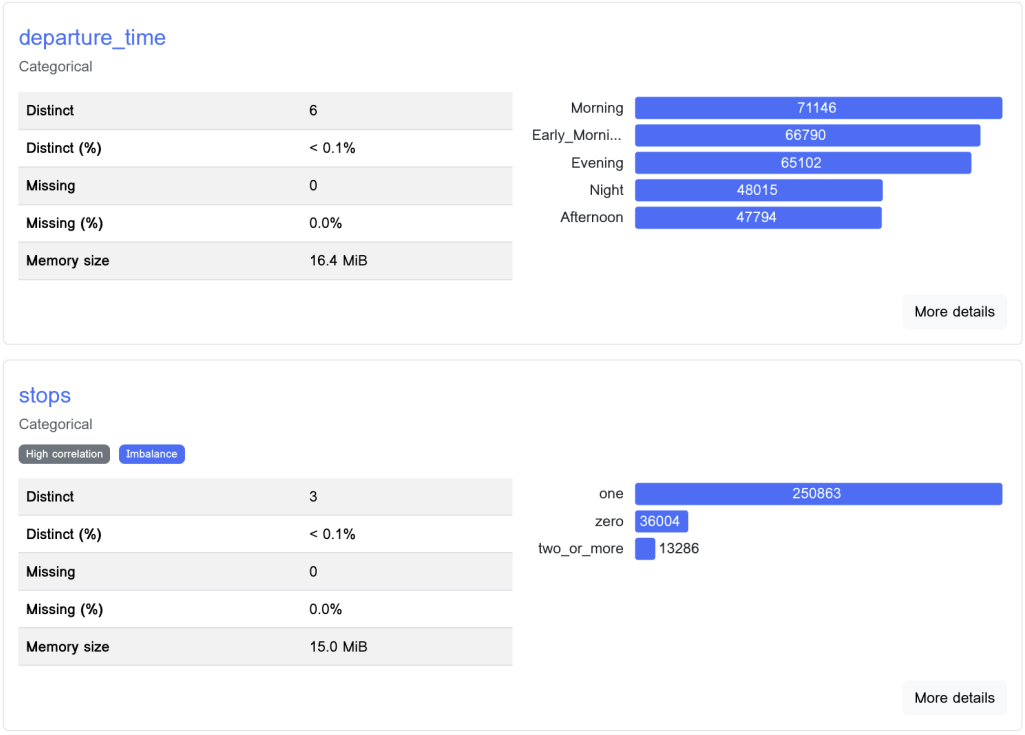

Figure 8: Data Profile Report - part 4.

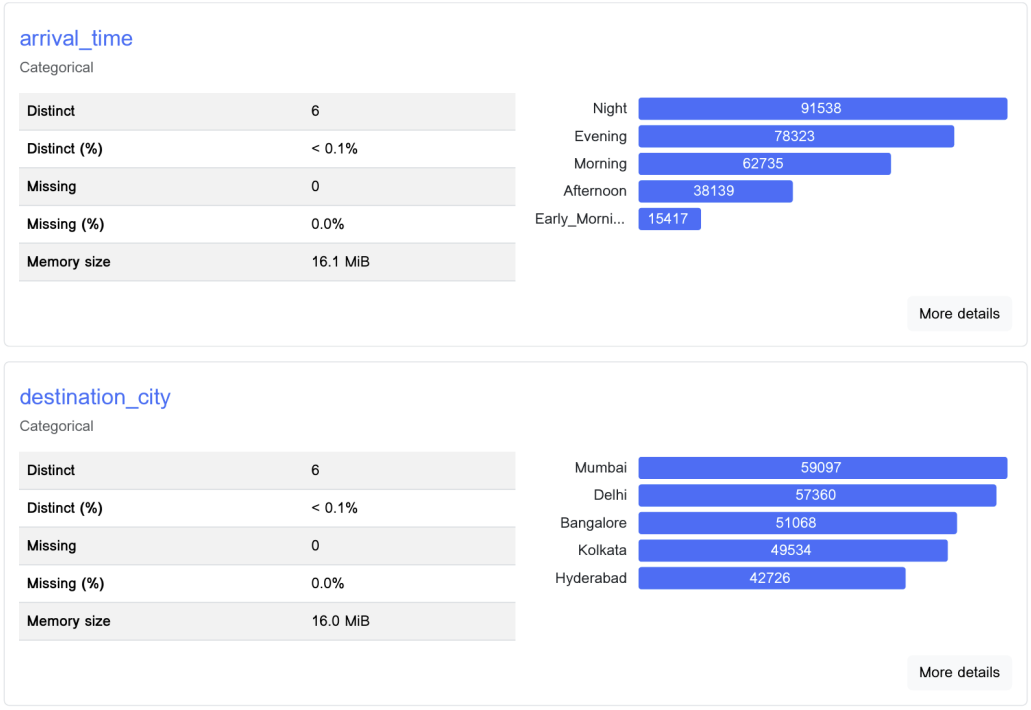

Figure 9: Data Profile Report - part 5.

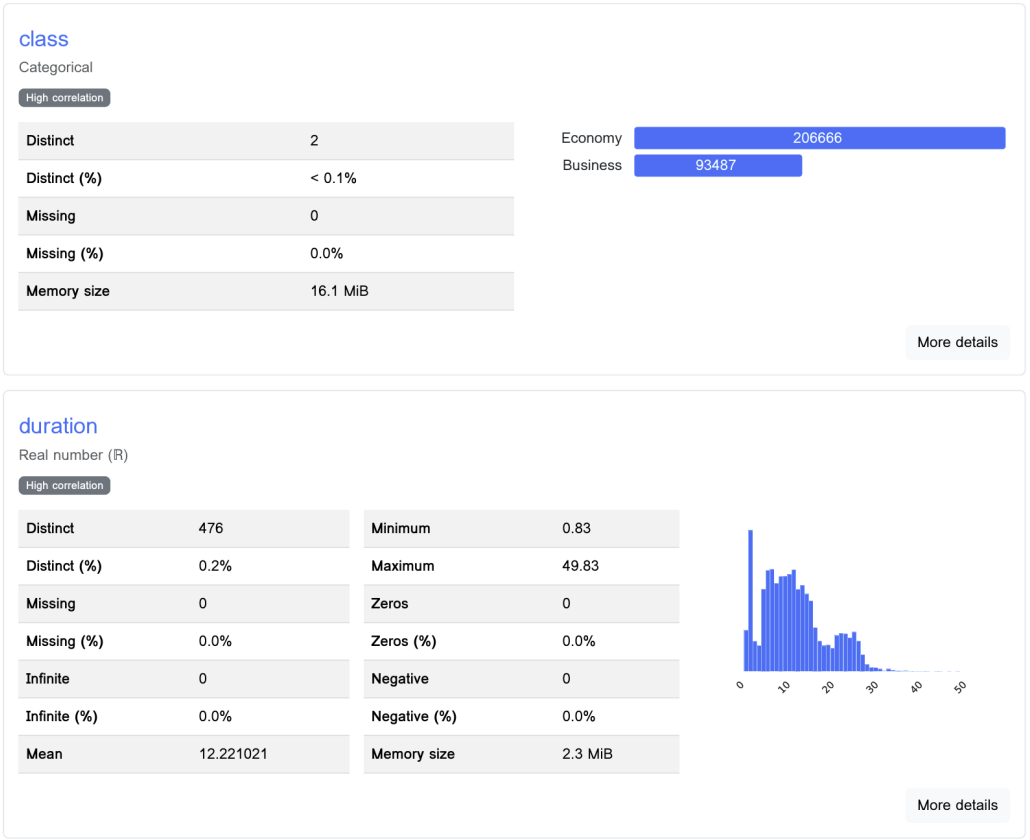

Figure 10: Data Profile Report - part 6.

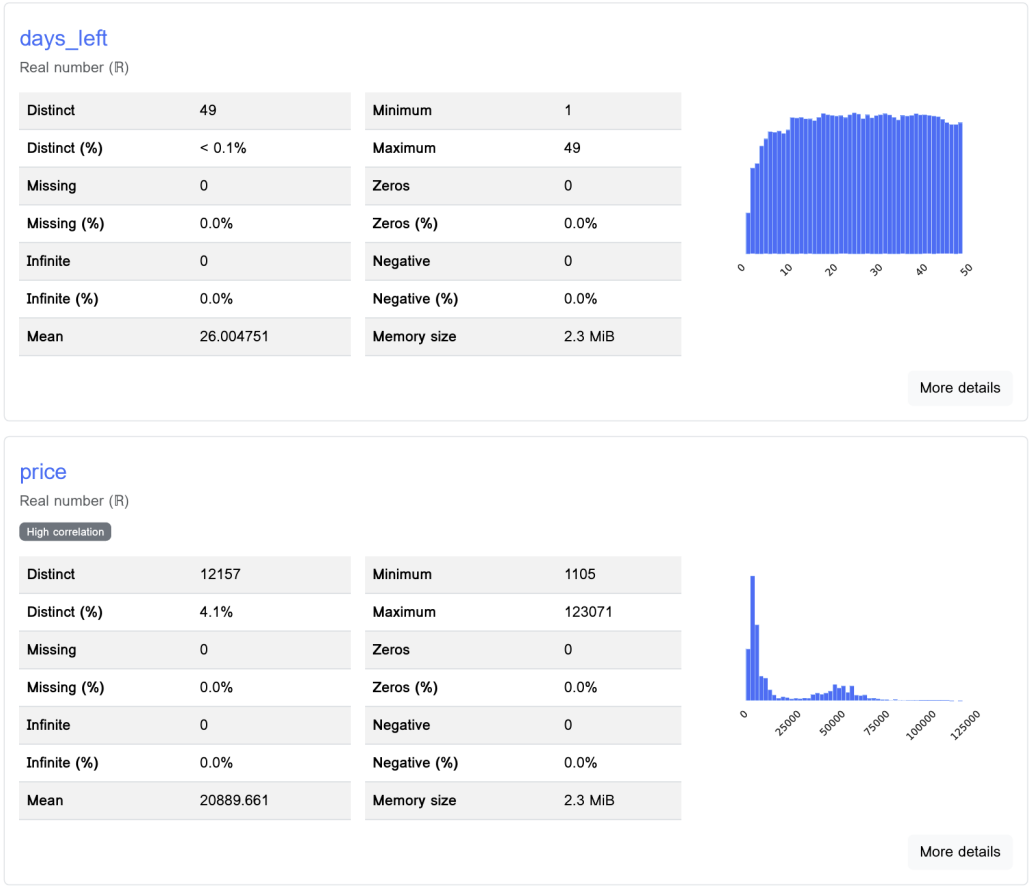

Figure 11: Data Profile Report - part 7.

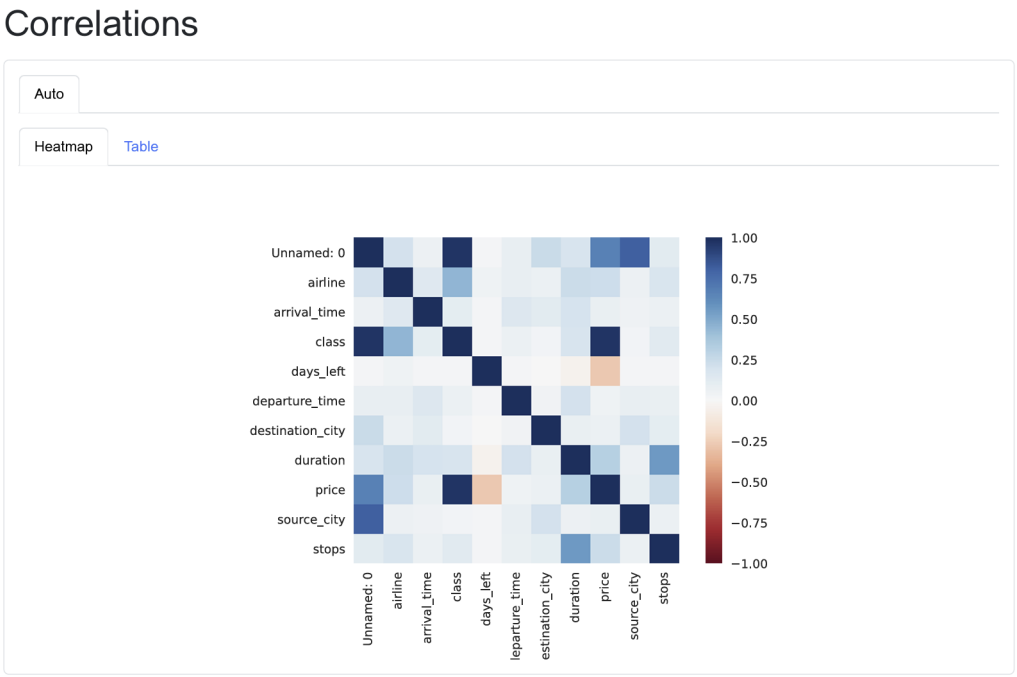

Figure 12: Data Profile Report - part 8.

Appendix: Plots

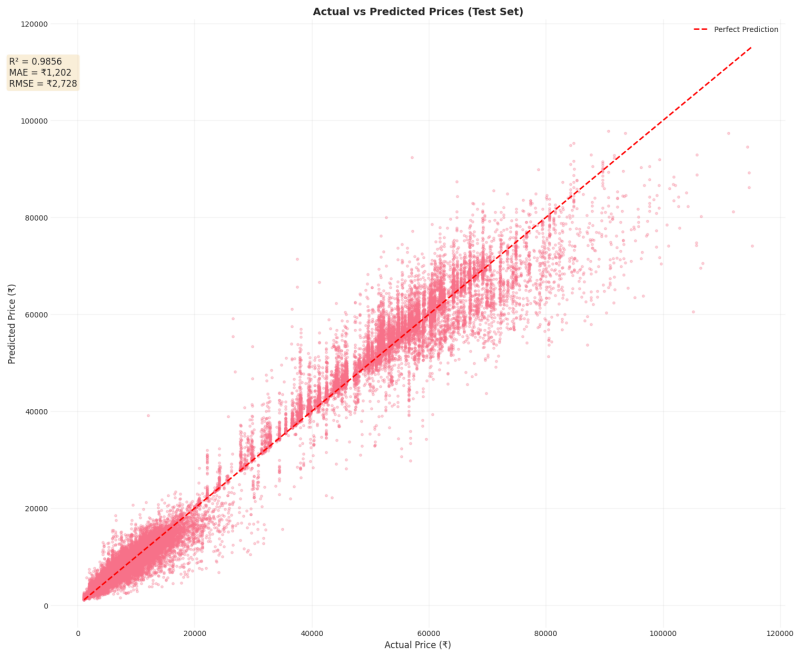

Figure 13: Actual vs Predicted Flight Prices

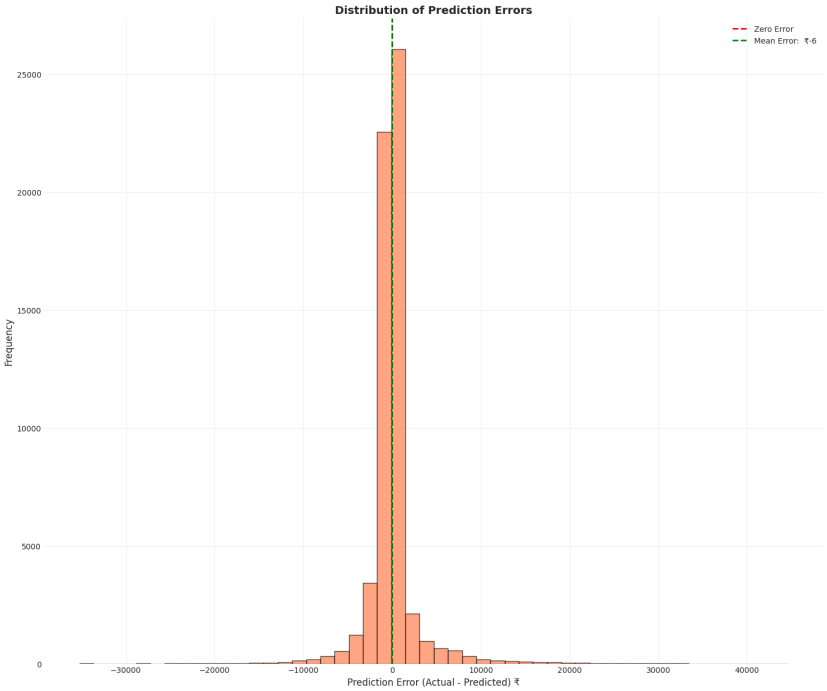

Figure 14: Distribution of Prediction Errors

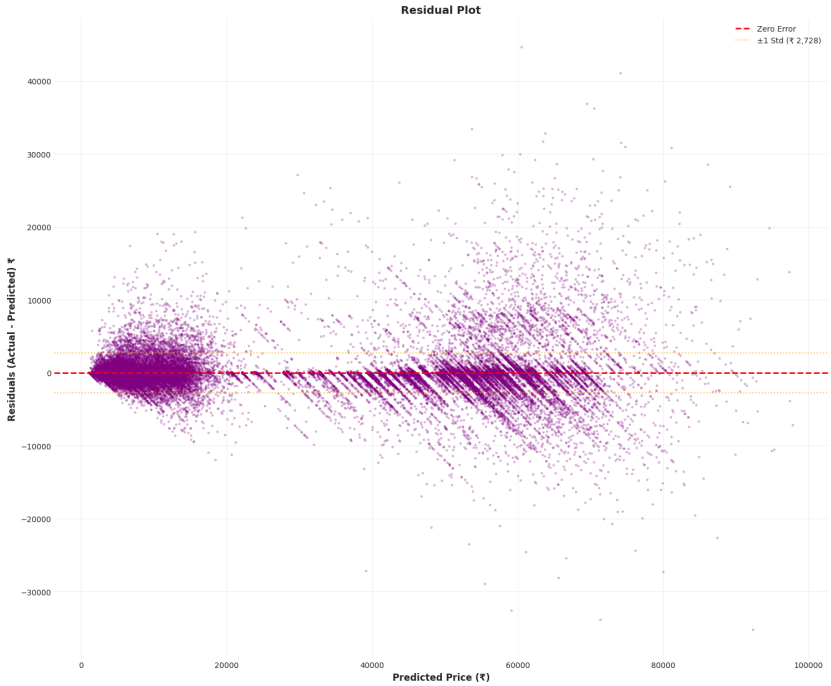

Figure 15: Residual Plot

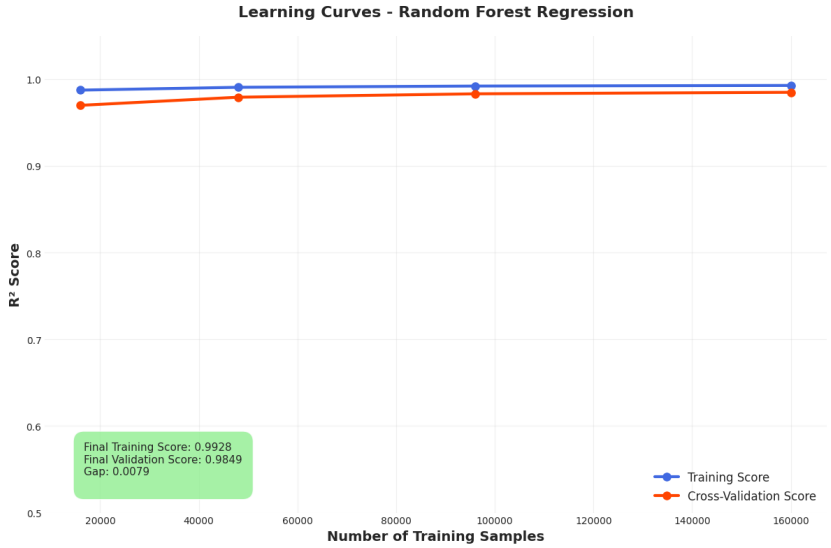

Figure 16: Learning Curves

Declaration of Generative AI Usage

As part of this semester project, I used generative AI tools (GitHub Copilot and Claude Sonnet 4.5) for the following purposes:

- brainstorming which topic to choose,
- consulting theoretical explanations and verifying correctness,
- drafting or reviewing parts of the code,
- language proofreading of the text.

I fully understand all methods, code, and interpretations used in this work and can independently explain them during the oral exam. I acknowledge that I bear full responsibility for the correctness of all content, calculations, code, and conclusions presented in this project.