AIRLINE FLIGHTS Price Prediction

Hands-on Machine Learning and Data Science 2024

Veronika Rybak
Ruslan Tsibirov
Olga Ivanova

Research Objective

The primary objective of this research is to leverage supervised machine learning techniques such as Linear Regression, Decision Trees, Random Forest Regressor, etc., to build a robust and accurate model for price prediction, by analyzing historical flight data. More precisely, we are determined to answer the following questions:

- **Which features have the most significant impact on ticket prices?**
- **What is the best machine learning model for predicting ticket prices based on the given features?**

Significance of the study

Understanding and predicting airline flight prices has significant implications for both, customers and airline industries.

An accurate price prediction model can aid customers in identifying the best time for ticket purchase, potentially saving their money and improving their travel and planning experience, which simultaneously results in an increased customer satisfaction and loyalty.

Conversely, airline companies can utilize the model to refine their pricing strategies, guaranteeing competitive yet profitable pricing, enhancing their revenue control, and boosting operational effectiveness.

The Dataset

- **Dataset** contains information about flight booking options from the website Easemytrip for flight travel between India's top 6 metro cities.

- There are 300261 datapoints and 11 attributes in the cleaned dataset.

- Data was collected for 50 days, from February 11th to March 31st, 2022.

- Data was collected in two parts: one for economy class tickets and another for business class tickets.

Price

Target variable stores information about the ticket price in rupees.

Features

Airline

The name of the airline company is stored in the airline column; a categorical feature having 6 different airlines.

Flight

Flight stores information regarding the plane's flight code; a categorical feature.

Source City

City from which the flight takes off. It is a categorical feature having 6 unique cities.

Departure Time

This is a derived categorical feature obtained created by grouping time periods into bins. It stores information about the departure time and have 6 unique time labels.

Stops

A categorical feature with 3 distinct values that stores the number of stops between the source and destination cities.

Features

Arrival Time

This is a derived categorical feature created by grouping time intervals into bins. It has six distinct time labels and keeps information about the arrival time.

Destination City

City where the flight will land. It is a categorical feature having 6 unique cities.

Class

A categorical feature that contains information on seat class; it has two distinct values: Business and Economy.

Duration

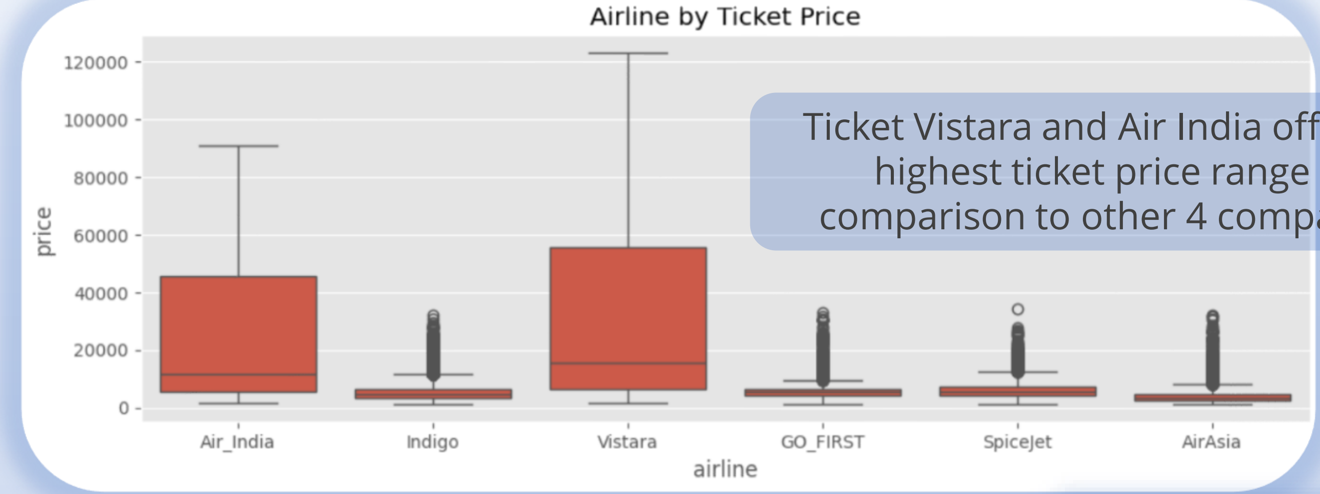
A continuous feature that displays the overall amount of time it takes to travel between cities in hours.

Days Left

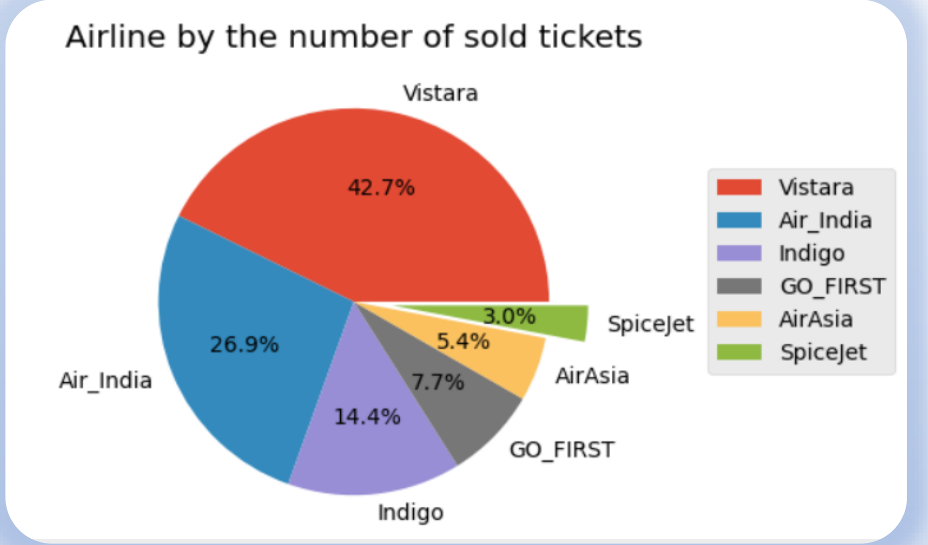
This is a derived characteristic that is calculated by subtracting the trip date by the booking date.

Data Visualization

Popularity of airlines

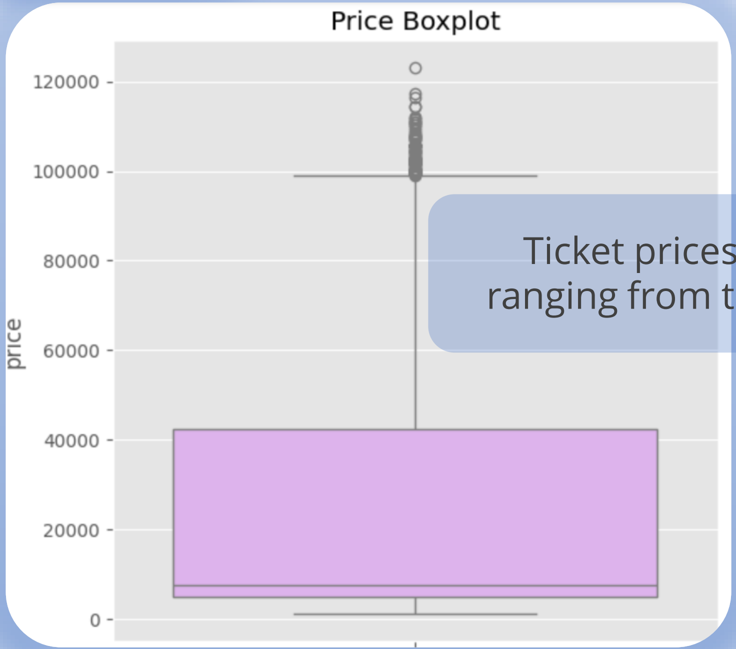


Ticket Vistara and Air India offer the highest ticket price range in comparison to other 4 companies



All 4 companies have numerous high-priced outliers, which may be related to the last-minute bookings for example

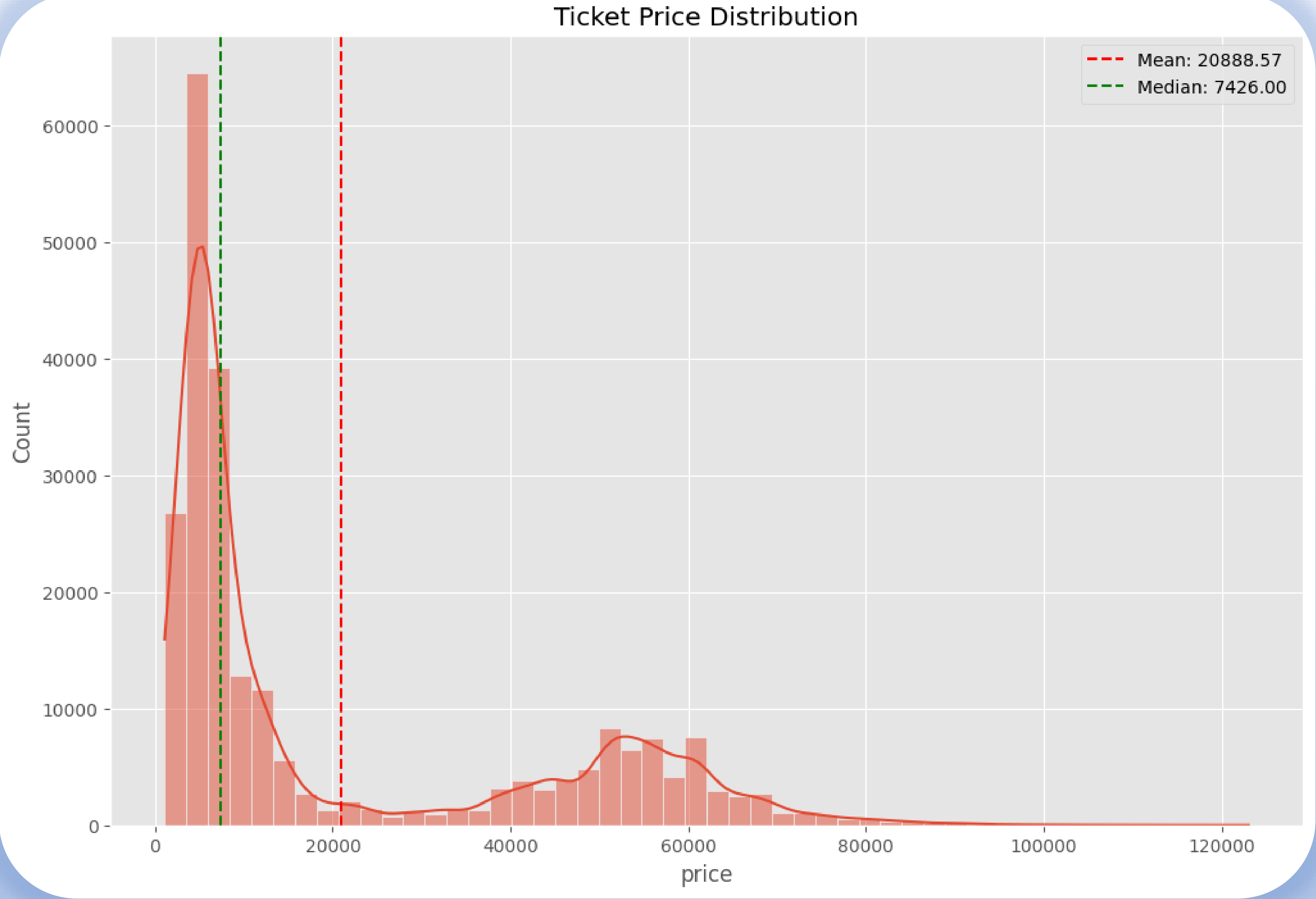
Vistara and Air India form approximately 70% of the entire market



Ticket prices are mainly distributed ranging from the 5000 to 40000 rupees

Data Visualization

Ticket Price Distribution



The distribution is heavily right-skewed, with most prices clustered at the lower end and a long tail extending to higher prices. This suggests that while there are many affordable tickets, there are also tickets that are significantly more expensive

Median ticket price is lower than the mean ticket price (which is common in right-skewed distributions), suggesting that higher-priced tickets are pulling the mean upwards

Data Visualization

Ticket price vs Duration of the flight

Most flights have a duration between 0 and 25 hours, with a high density of points in this range

Prices vary widely, especially for the flights with shorter duration

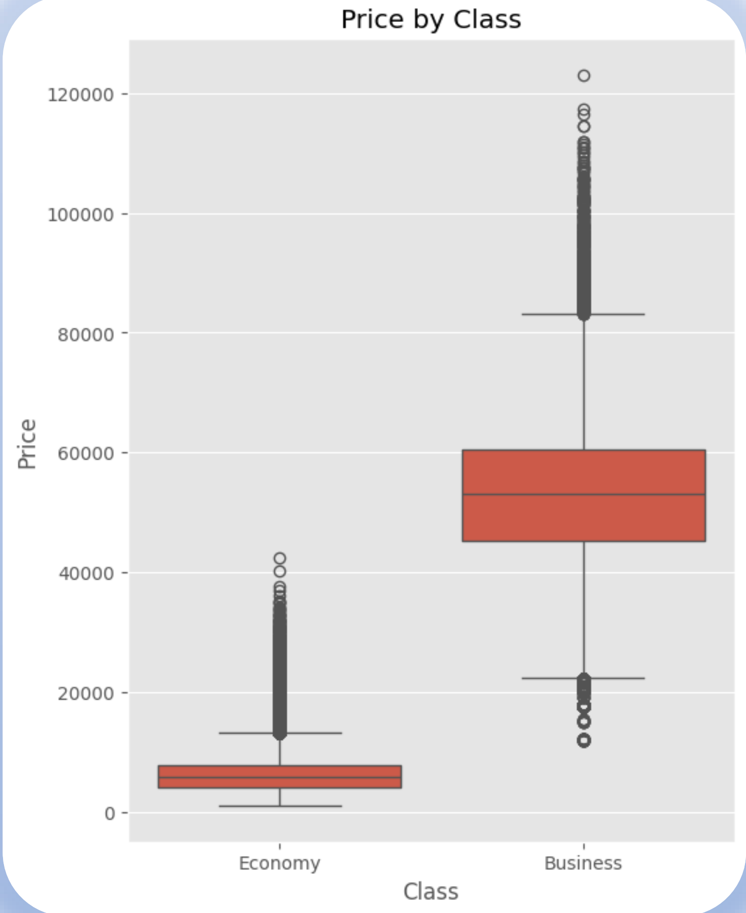
Some points indicate very high prices for both short and long flights, suggesting Business class or last-minute tickets

We can also observe some kind of gap on the plot in between 25000 and 40000 rupees (approximately), which is most probably due to the difference in pricing in different classes, Business and Economy. We shall investigate it further to prove our claim (see below).

Data Visualization

Flight Class Distribution

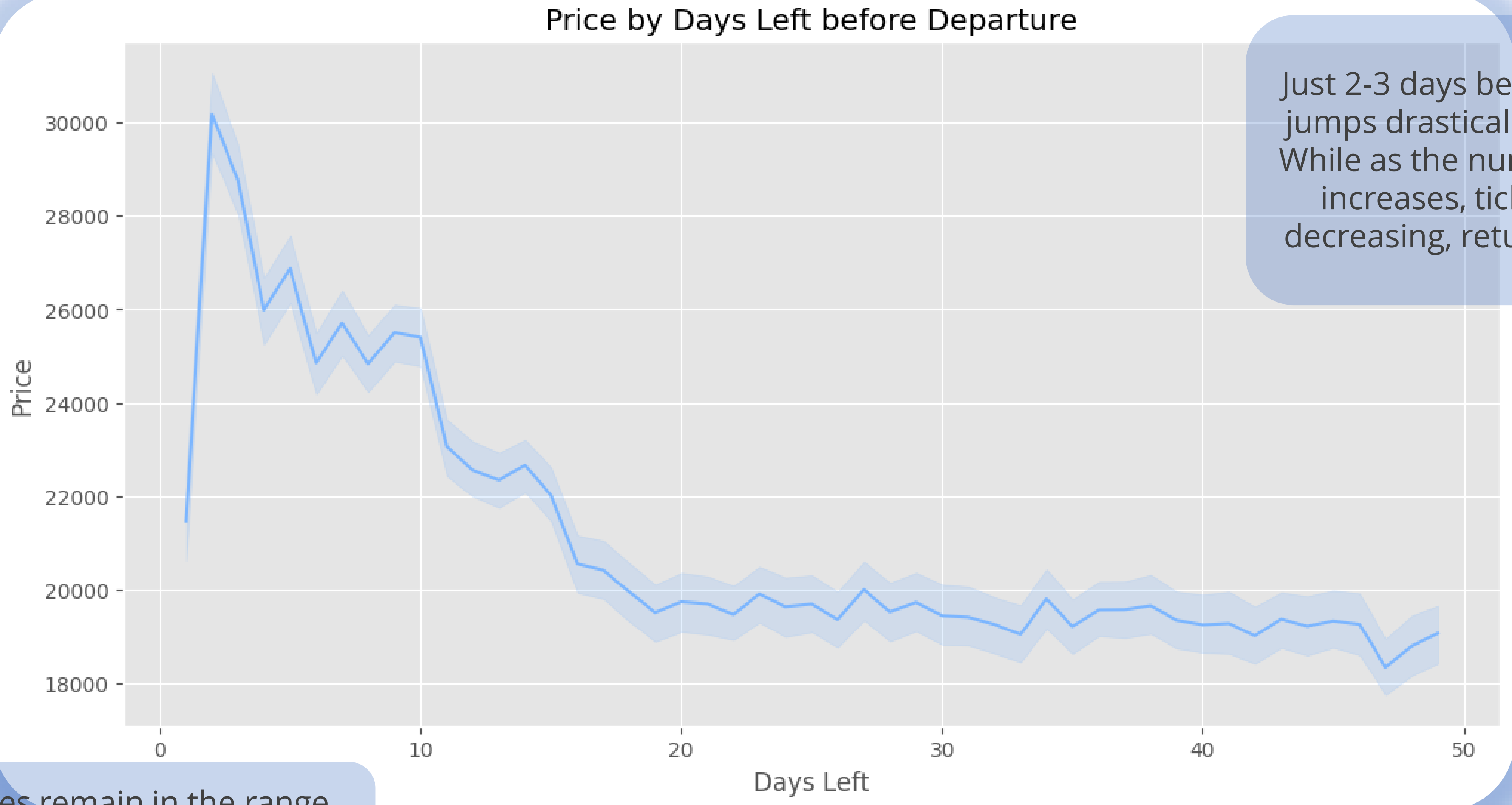
Indeed, the pie chart above demonstrates the correctness of our claim, that the low ticket density within the range between 25000 and 40000 rupees is most likely due to deliberate pricing strategy to create a clear distinction between Business and Economy class flights



Business class shows more price variability (wider box and longer whiskers), while economy class prices are more clustered

Data Visualization

Price by Days Left before Departure

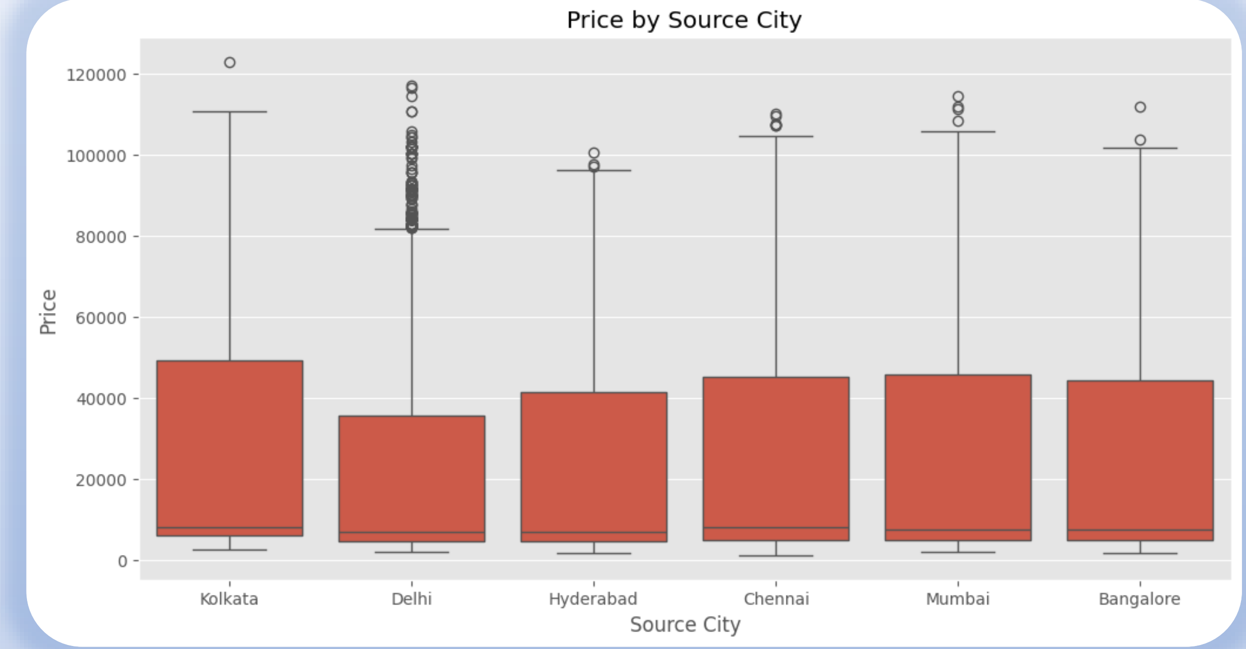


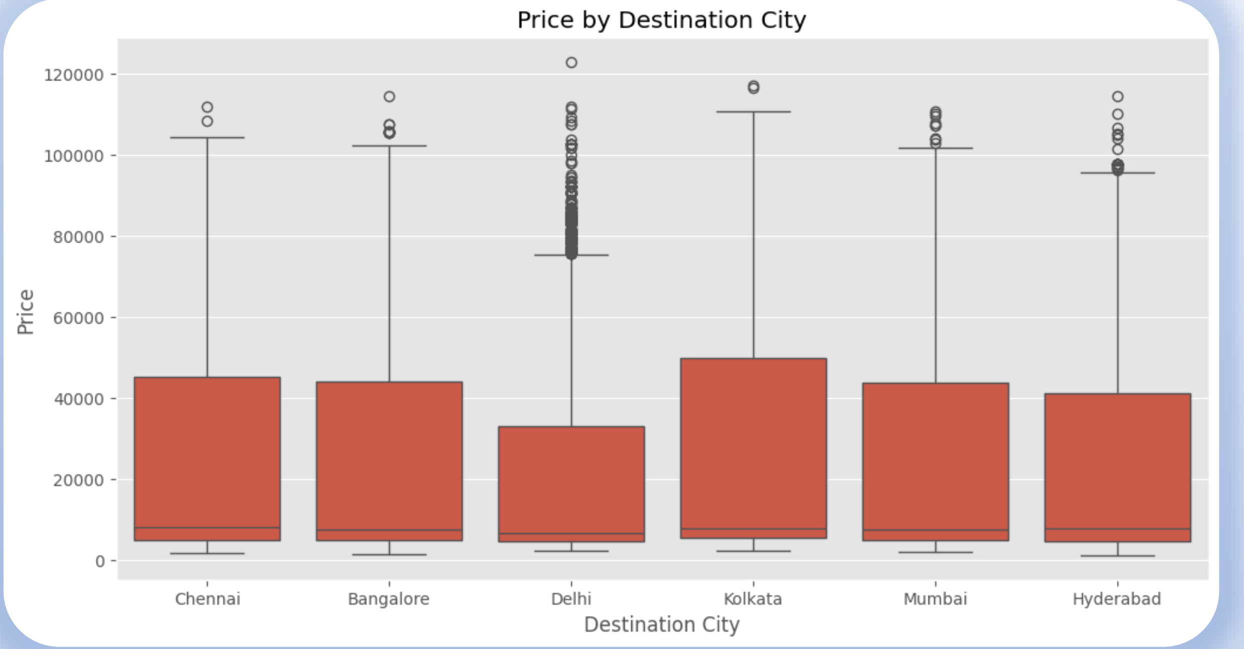
Just 2-3 days beforehand, prices jumps drastically, hitting 30000. While as the number of days left increases, ticket price keep decreasing, returning to regular

Noticeably, prices remain in the range 18000-20000 rupees with 20 days and more left before the departure

Data Visualization

Price by Source and Destination City





Most cities, both as source and destination, have median prices in the range of 6000-8000

Delhi stands out with a significant number of high-priced outliers both as source and destination city

The interquartile ranges are quite similar across all cities, indicating a consistent spread in the middle 50% of ticket prices. The overall price ranges from around 3000 to 50000 for most cities

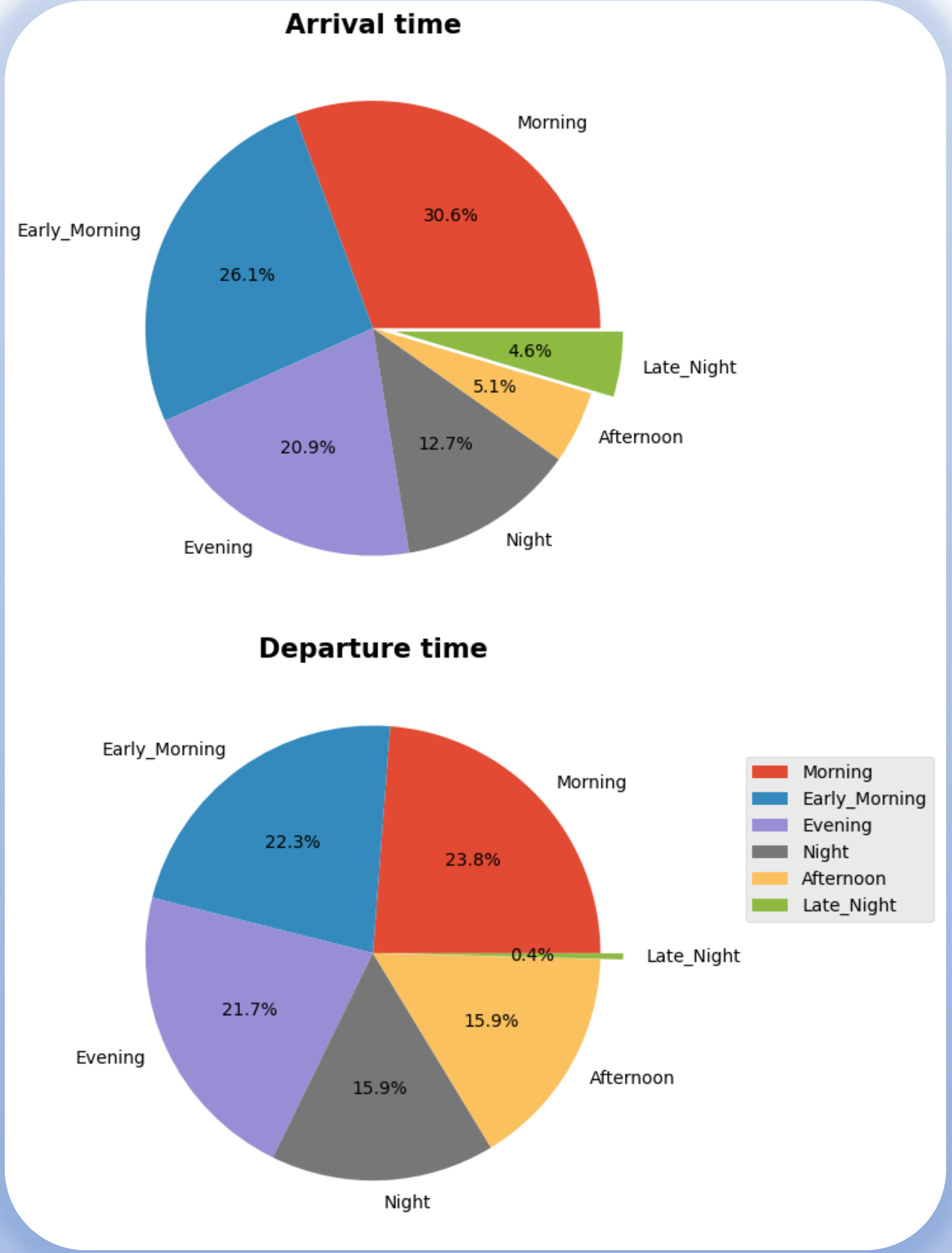
Data Visualization

Tickets Distribution by Arrival and Departure Time

Both departure and arrival times see the highest activity in the morning, with arrival slightly higher than departures

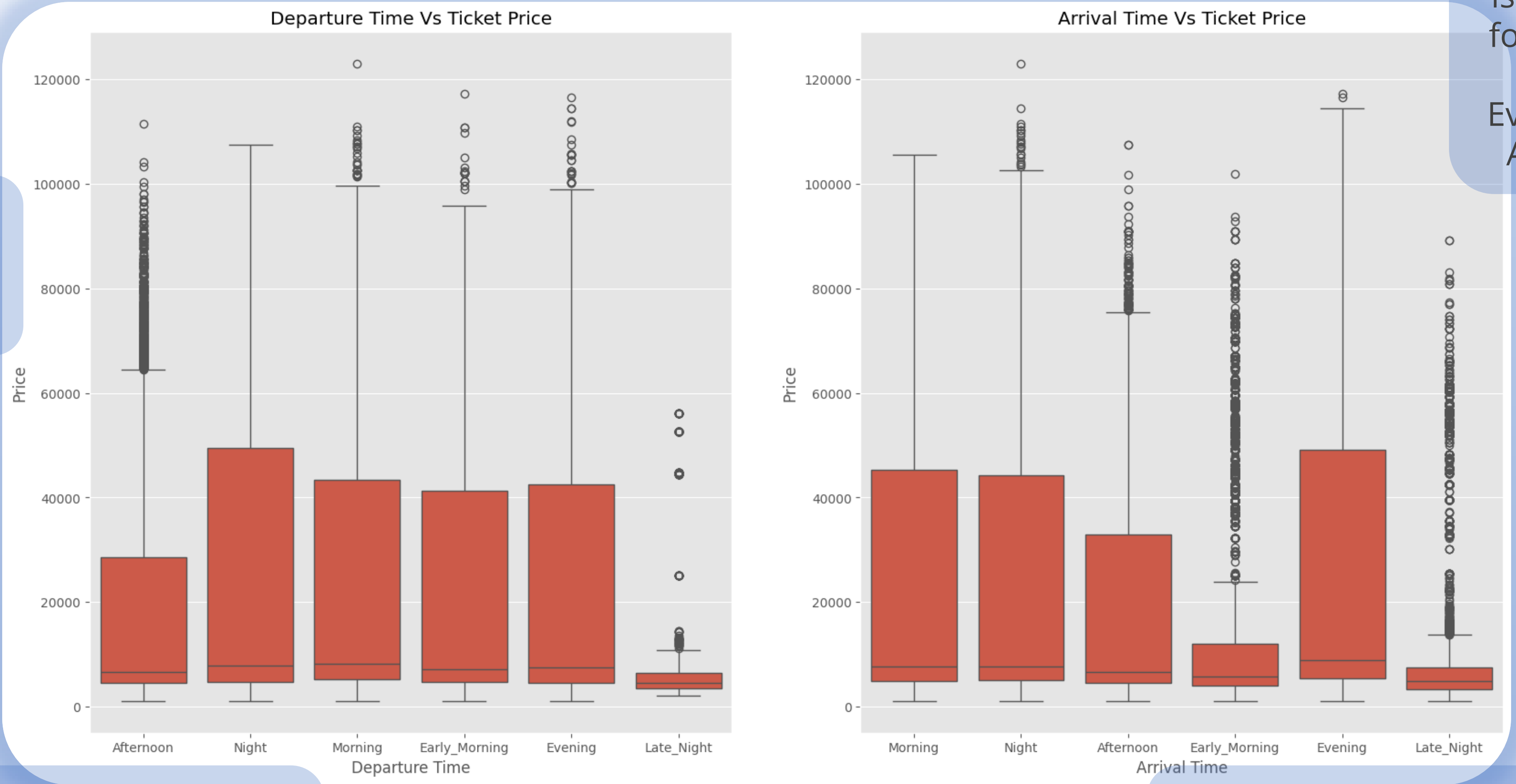
Both night and afternoon periods see moderate/low activity compared to other times

Very few flights are scheduled to depart late at night, while a small number arrive during this period



Data Visualization

Departure and Arrival Time vs Ticket Price



The variability in ticket prices is quite high for Morning, Night, Evening, and Afternoon

Morning and Night flights are generally more expensive

Flights departing and arriving in the Late Night tend to have significantly lower ticket prices in comparison to other times of the day

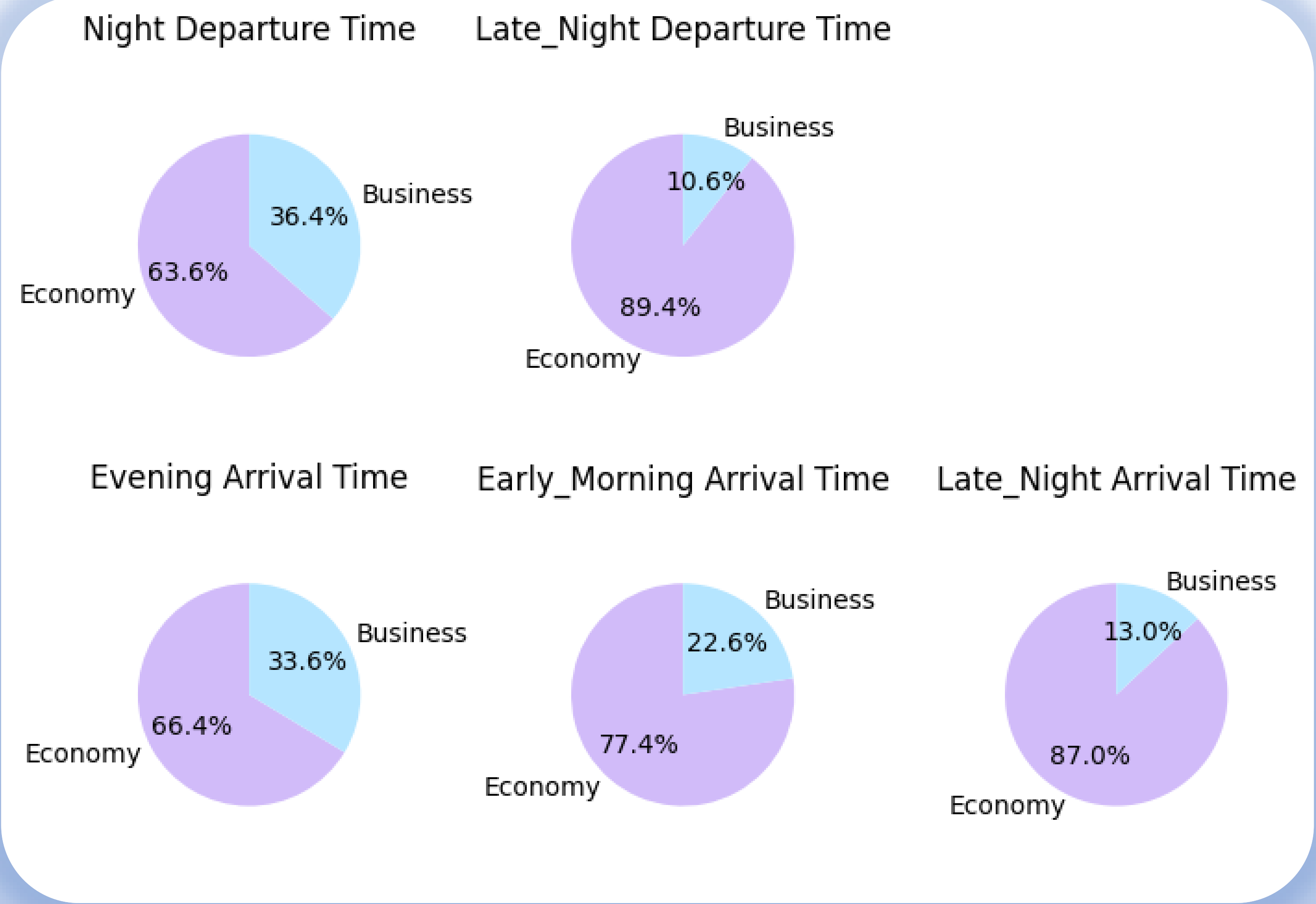
Early Morning and Late Night departures, as well as departures in Afternoon, have a significant number of outliers

Data Visualization

Class Distribution by Departure and Arrival Times

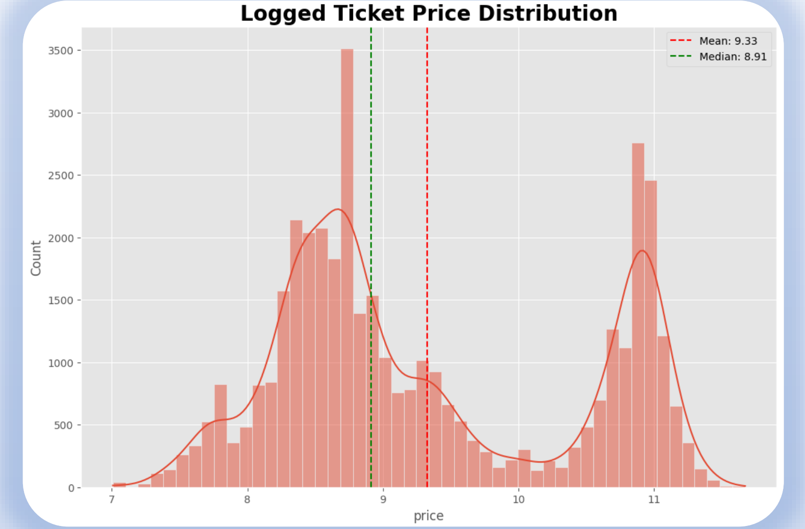
The reason why Late Night Departure is so cheap in comparison to other timings is that majority of the tickets are of the Economy Class, with Business class forming only 10.6% of tickets

Evening arrivals are formed by 33.6% of Business class, explaining why they are more expensive with higher variability in comparison to other timings. As for the least popular and quite cheap, in comparison to others, Late Night arrival, we see that there is quite a low amount of Business class tickets, around 13%



Data preparation

Logged Ticket Price Distribution



We apply log transformation to our target variable to reduce the gap between significantly more expensive tickets and the majority, which could aid some of our models in performing better

Feature Selection

We drop the column 'flight" with flight numbers: this information is completely useless since it has zero impact on customers' decision when purchasing tickets

Feature Interaction

By combining the features Combining Source and Destination Cities, we are able to create a unique identifier for each route. Such feature engineering, where we create new features from the existing ones will most likely simplify our analysis, handle the categorical variables, and potentially improve our model performance

One-Hot Encoding

Using One-Hot Encoding, we convert categorical variables into a numerical format suitable for machine learning algorithms, also ensuring that no ordinal relationship is implied between categories (i.e. category 2 > category 1). As a result, our dataset contains 55 columns

Model Evaluation & Feature Importance Analysis

Summarizing the model performances, highlighting the most important features, and suggesting next steps for fine-tuning the most promising models.

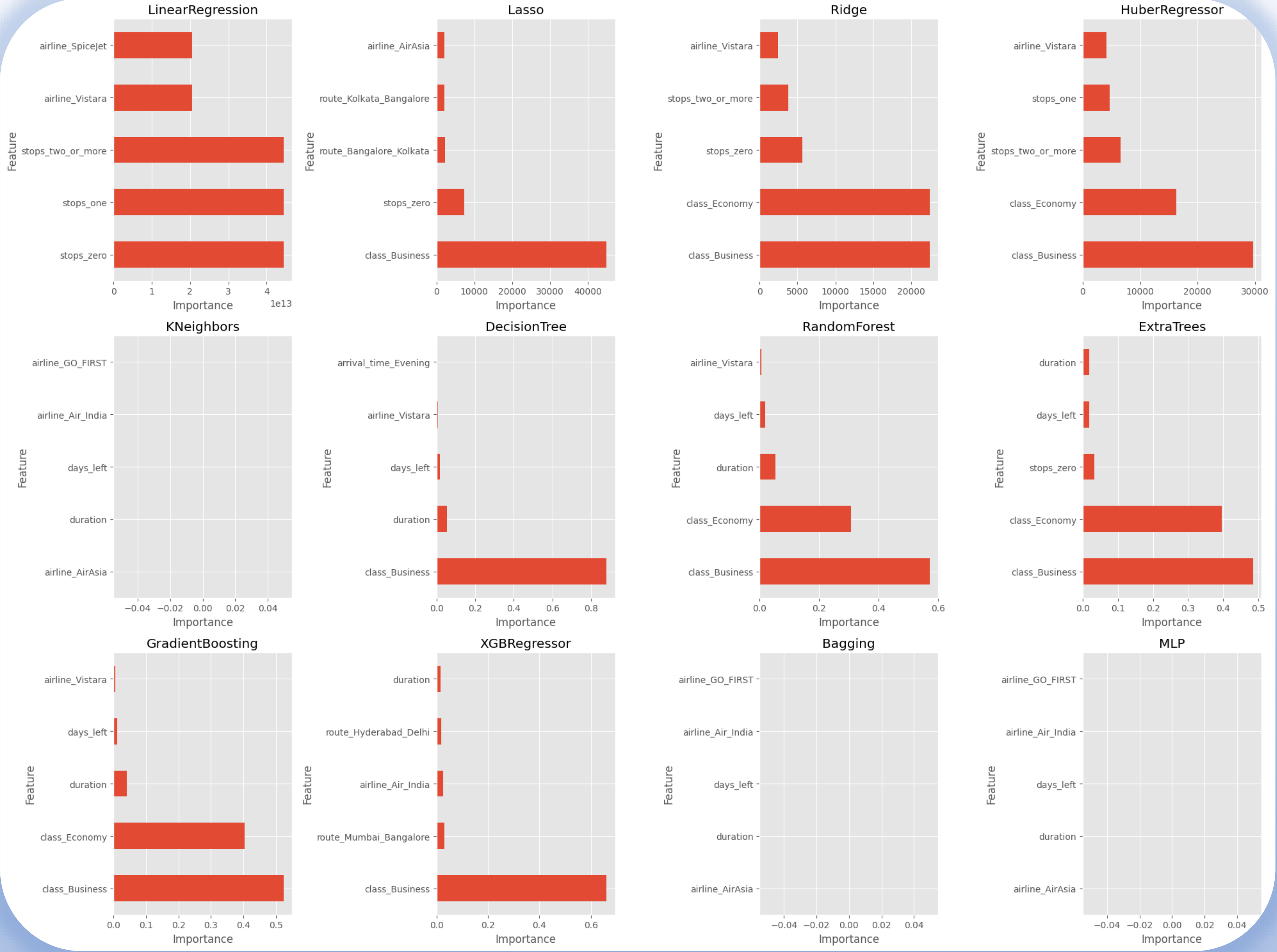
	Model	MAE	RMSE	R2
0	ExtraTrees	1519.348928	3368.160398	0.978130
1	RandomForest	1561.068588	3250.603931	0.979630
2	Bagging	1628.844087	3405.791053	0.977639
3	DecisionTree	1670.188500	4199.845198	0.965996
4	XGBRegressor	2051.714682	3596.166901	0.975069
5	GradientBoosting	2943.492004	5011.695285	0.951580
6	MLP	4165.960169	6234.110192	0.925078
7	HuberRegressor	4203.699501	7179.441857	0.900634
8	Lasso	4582.153911	6804.249546	0.910748
9	Ridge	4585.627956	6803.900272	0.910757
10	LinearRegression	4585.763735	6803.872851	0.910758
11	KNeighbors	4728.592420	7707.665266	0.885474

We will use R2, RMSE, MAE, MAPE metrics to evaluate our models.

Here we can compare performances of our "dirty" models and select couple of the most promising to fine-tune them.

We see that tree-based models give us the best performance, but in educational purposes let's also try to tune some linear models and MLPRegressor.

The most important features:

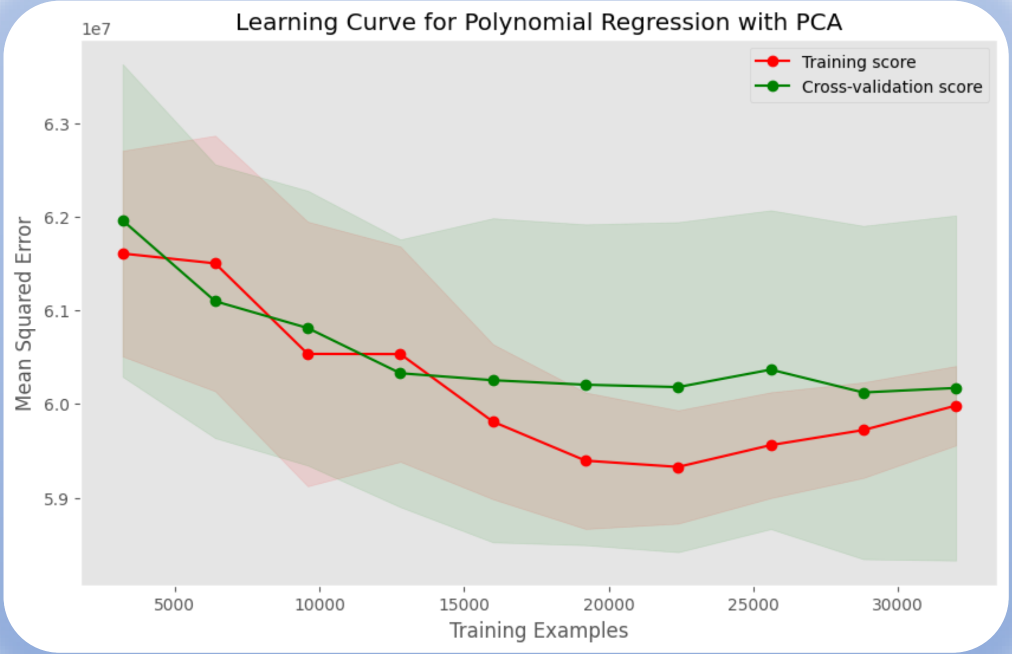


Fine-Tuning the System

Linear Regression with Polynomial Features

- **StandardScaler**
- **Hyperparameters**: 'regressor__poly__degree’ (polynomial degree) and 'regressor__pca__n_components’ (the number of components for PCA).
- **Elapsed time** for GridSearchCV (Polynomial): 70.29 seconds

Best parameters: {'regressor__pca__n_components': 40, 'regressor__poly__degree': 1}
Best cross-validation score: 4585.626467539792

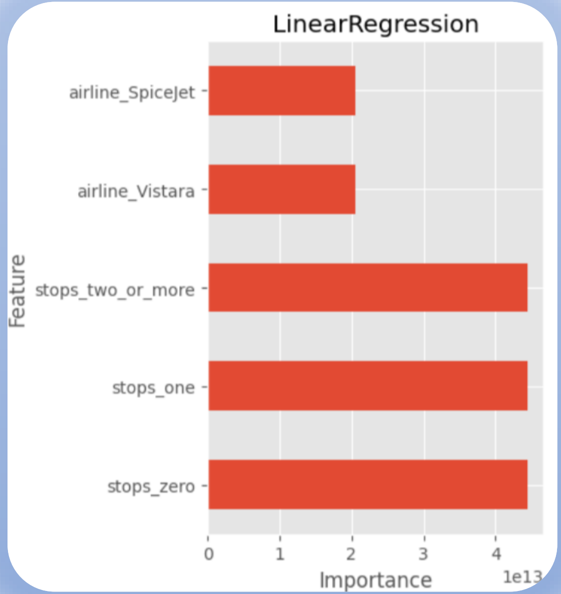


Metrics:

Train Set:
R2 Score: 0.88277
RMSE: 7746.40670
MAE: 4577.75621
MAPE: 0.26436

Test Set:
R2 Score: 0.88962
RMSE: 7567.01488
MAE: 4499.51610
MAPE: 0.26571

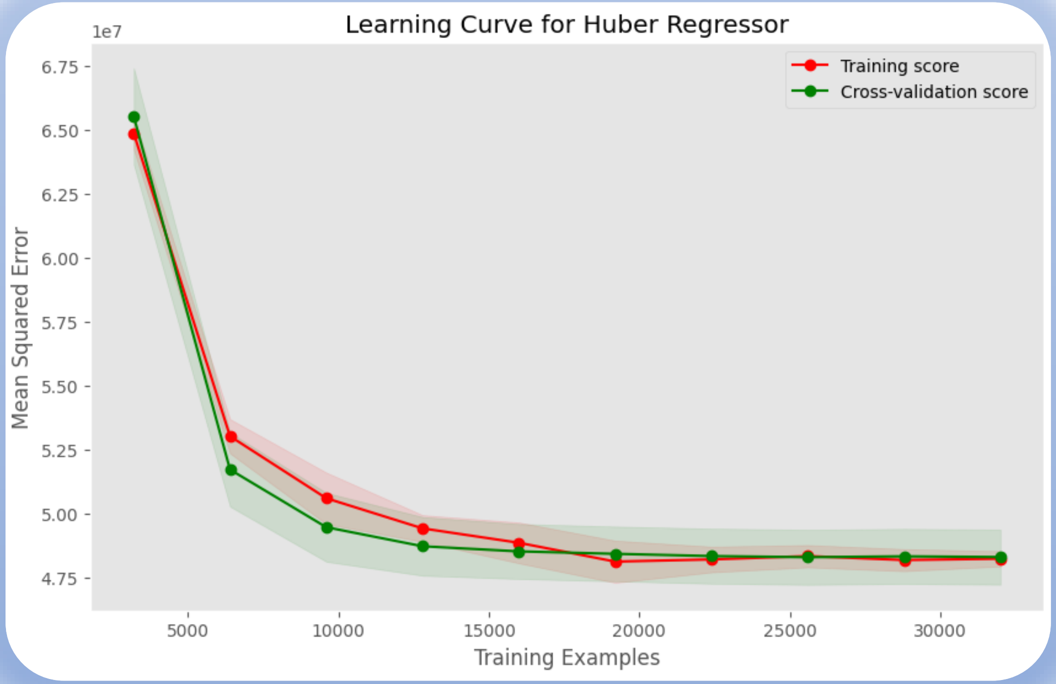
The most important features:



Huber Regression

- **StandardScaler**
- **Hyperparameters:** 'hub__alpha' (regularization strength parameters, ranging logarithmically from 10^{-4} to 10^{4}) and 'hub__epsilon' (parameters for the Huber loss function, controlling the point where the loss function changes from quadratic to linear).
- **Elapsed time** for GridSearchCV (Huber): 575.73 seconds

Best parameters: {'hub__alpha': 0.18420699693267145, 'hub__epsilon': 1.4}
Cross-validated RMSE: 6949.169195287733



Metrics:

Train Set:
R2 Score: 0.90575
RMSE: 6945.67623
MAE: 4117.67319
MAPE: 0.30099

Test Set:
R2 Score: 0.90156
RMSE: 7145.85867
MAE: 4216.04143
MAPE: 0.30701

The most important features:

Random Forest Regressor (GridSearch)

- **Hyperparameters:**
  'n_estimators': [100, 150] (Reduced the upper range to limit complexity),
  'max_features': ['auto', 'sqrt', 'log2'] (Added 'log2' to consider fewer features at each split),
  'max_depth': [5, 10, 15] (Reduced the range to prevent very deep trees),
  'min_samples_split': [2, 5, 10] (Added to control the minimum number of samples required to split a node),
  'min_samples_leaf': [1, 2, 4] (Added to ensure a minimum number of samples at the leaf nodes) and
  'bootstrap': [True, False] (Added to explore the effect of bootstrap sampling)

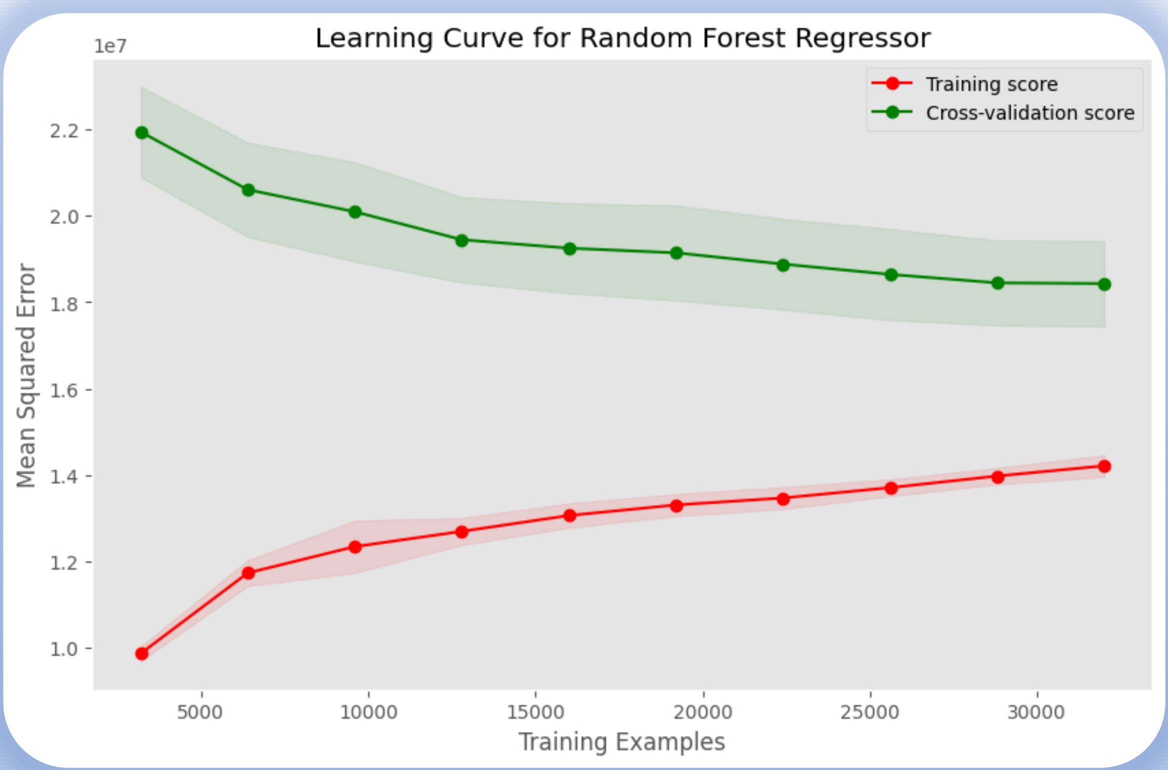
- **Elapsed time** for GridSearchCV (RandomForest):
  10171.34 seconds

Best parameters: {'bootstrap': True, 'max_depth': 15, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}
Best cross-validation score: 0.9641405109202088

Metrics:

Train Set:
R2 Score: 0.97172
RMSE: 3804.90621
MAE: 2047.27419
MAPE: 0.13247

Test Set:
R2 Score: 0.96385
RMSE: 4330.40679
MAE: 2336.73706
MAPE: 0.15239



Random Forest Regressor (Hyperopt)

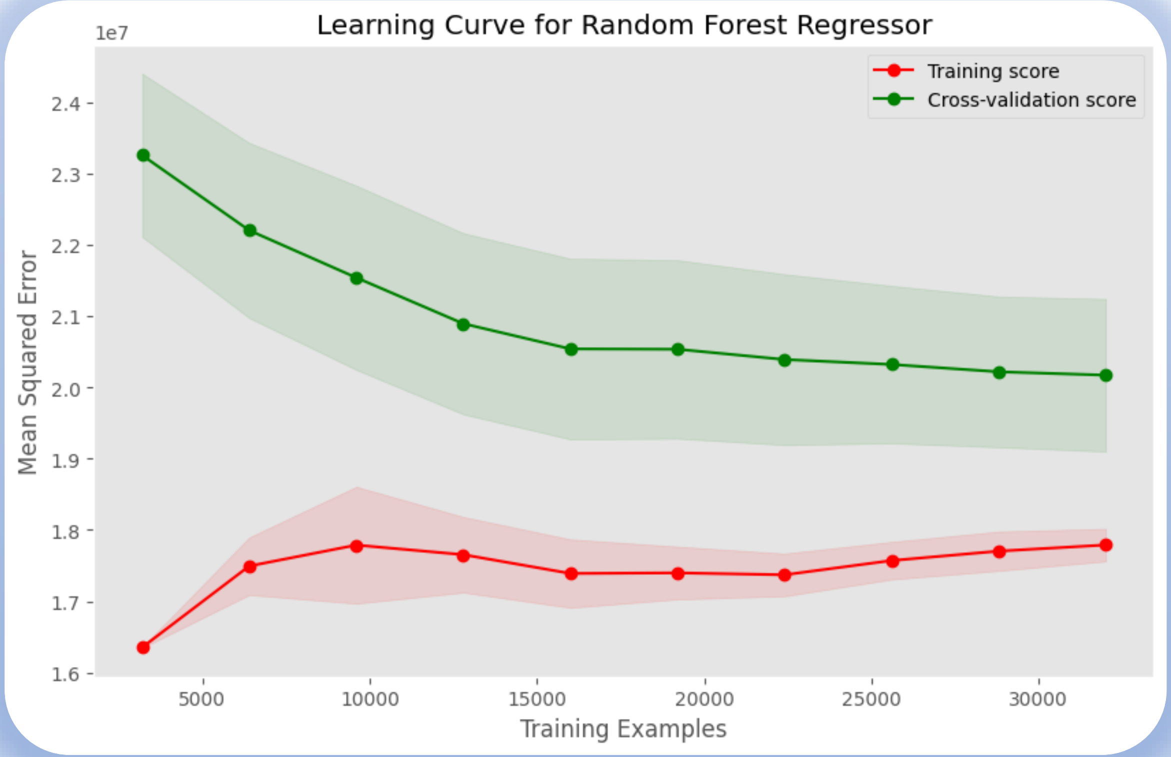
Since hyperparameter tuning with GridSearchCV would probably take a long time (because it searches the full space of available parameter values), we'll try to use another optimization library named hyperopt. Hyperopt uses a popular alternative to tune the model hyperparameters called Bayesian Optimization.

Best parameters: {'bootstrap': 0, 'max_depth': 13.0, 'max_features': 0, 'min_samples_leaf': 4.0, 'min_samples_split': 4.0, 'n_estimators': 150.0}
Best cross-validation score: 0.9641405109202088

Metrics:

Train Set:
R2 Score: 0.96495
RMSE: 4235.51196
MAE: 2342.95637
MAPE: 0.15166

Test Set:
R2 Score: 0.96062
RMSE: 4519.90641
MAE: 2485.52781
MAPE: 0.16249



Elapsed time for HyperOpt (RandomForest): 1123.86 seconds

Random Forest Regressor

GridSearch

- needed around 169 minutes to get the work done

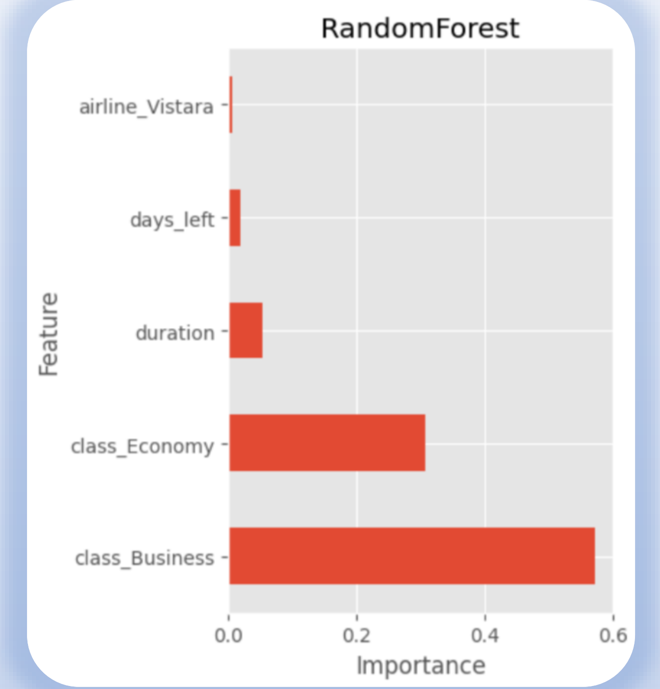
- give us better scores, but are overfitting a bit

Hyperop

- while Hyperopt have done it in around 18 minutes

- give us better generalization, which seem to be more robust estimation

The most important features:



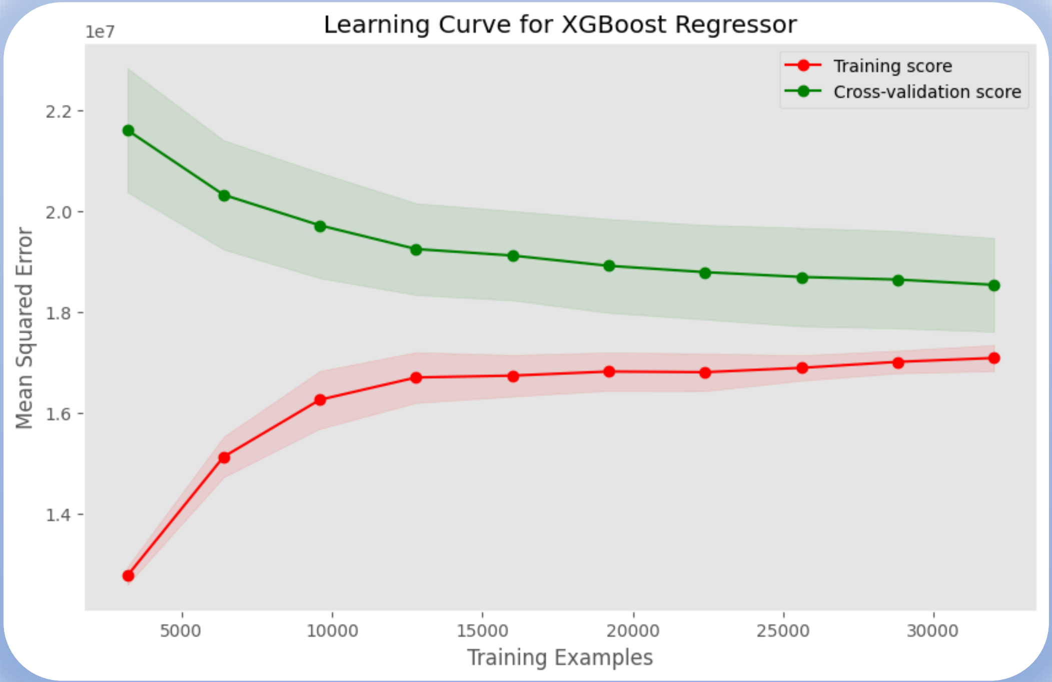
Comparison:
Yet, Random Forest model performs the best in terms of mean squared error compared to the linear and polynomial regression models. It has the lowest cross-validation error among the three, indicating better generalization.

The linear regression model showed the smallest gap between training and validation errors, indicating less overfitting.

XGBoost Regressor

- **Hyperparameters**: 'colsample_bytree': [0.8, 1.0] (Subsample ratio of columns when constructing each tree),
  'learning_rate': [0.01, 0.1] (Step size shrinkage),
  'max_depth': [3, 5] (Maximum depth of a tree),
  'n_estimators': [50, 100] (Number of boosted trees to fit) and
  'subsample': [0.8, 1.0] (Subsample ratio of the training instances)
- **Elapsed time** for GridSearchCV (XGBoost): 102.03 seconds

Best parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100, ‘subsample': 0.8}
Best cross-validation score: 0.9638011523402191

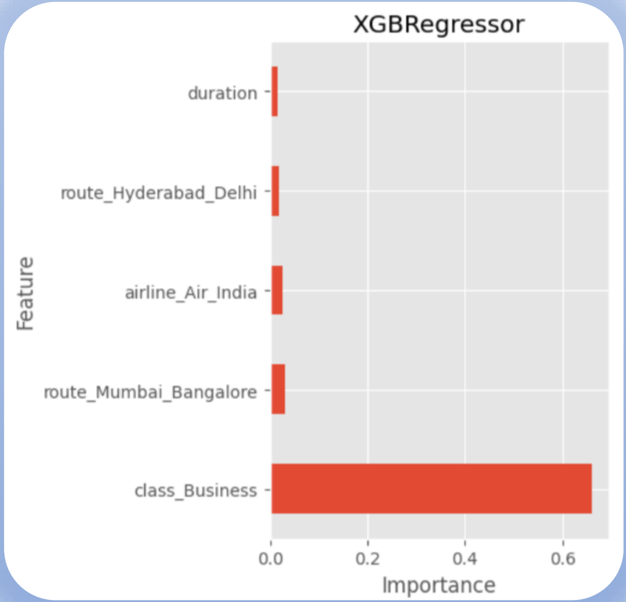


Metrics:

Train Set:
R2 Score: 0.96601
RMSE: 4171.33015
MAE: 2428.88379
MAPE: 0.17197

Test Set:
R2 Score: 0.96372
RMSE: 4338.10259
MAE: 2510.80365
MAPE: 0.17819

The most important features:



Extra Trees (GridSearch)

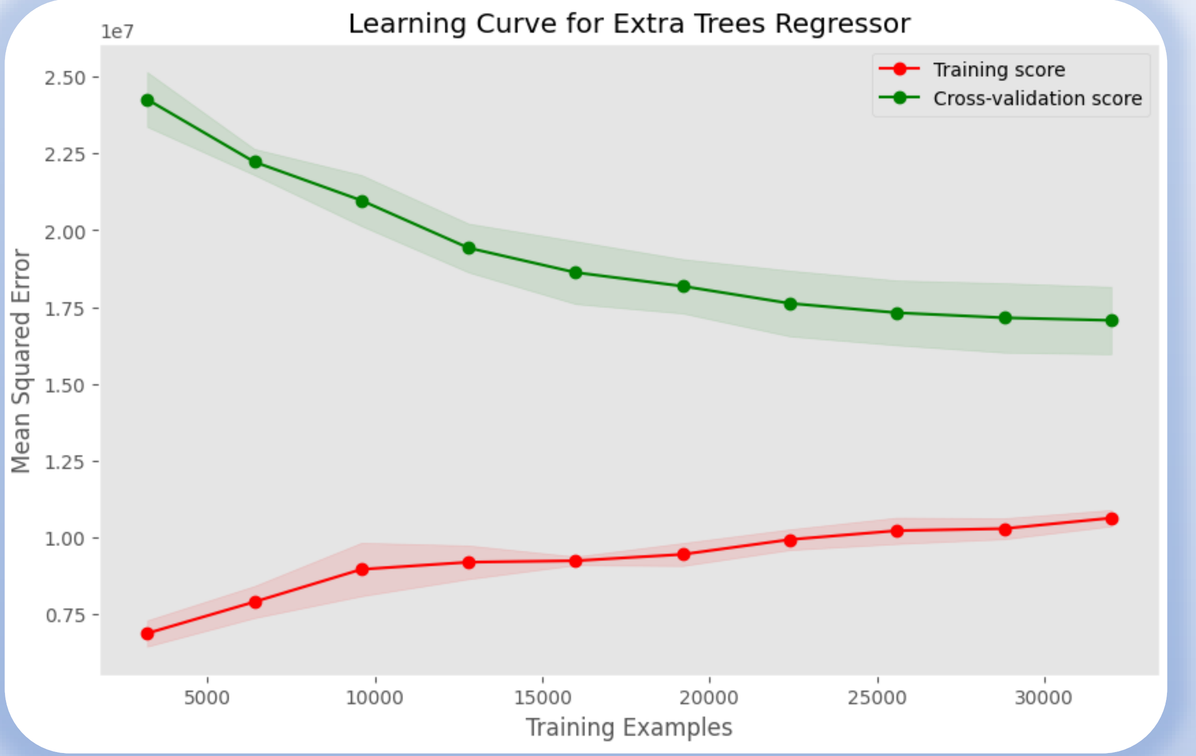
- **Hyperparameters:** 'n_estimators': [100, 150] (A narrower range to reduce fits),
  'max_depth': [10, 20] (Focused on practical depths) and
  'min_samples_split': [2, 10] (Critical to control overfitting and underfitting)
- **Elapsed time** for GridSearchCV (ExtraTrees): 1704.72 seconds

Best parameters: {'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 150}
Best cross-validation score: 0.9666677266665424

Metrics:

Train Set:
R2 Score: 0.97798
RMSE: 3356.88782
MAE: 1531.58871
MAPE: 0.09392

Test Set:
R2 Score: 0.96665
RMSE: 4158.99296
MAE: 2114.67365
MAPE: 0.13795



gap between two curves is quite large, which shows overfitting

Extra Trees (Hyperopt)

Now let us use HyperOpt and compare the results with GridSearch:

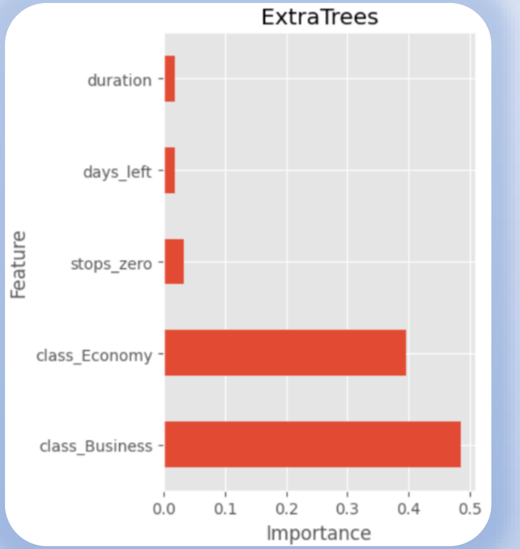
Best parameters: {'bootstrap': 0, 'max_depth': 14.0, 'max_features': 0, 'min_samples_leaf': 2.0, 'min_samples_split': 2.0, 'n_estimators': 120.0}

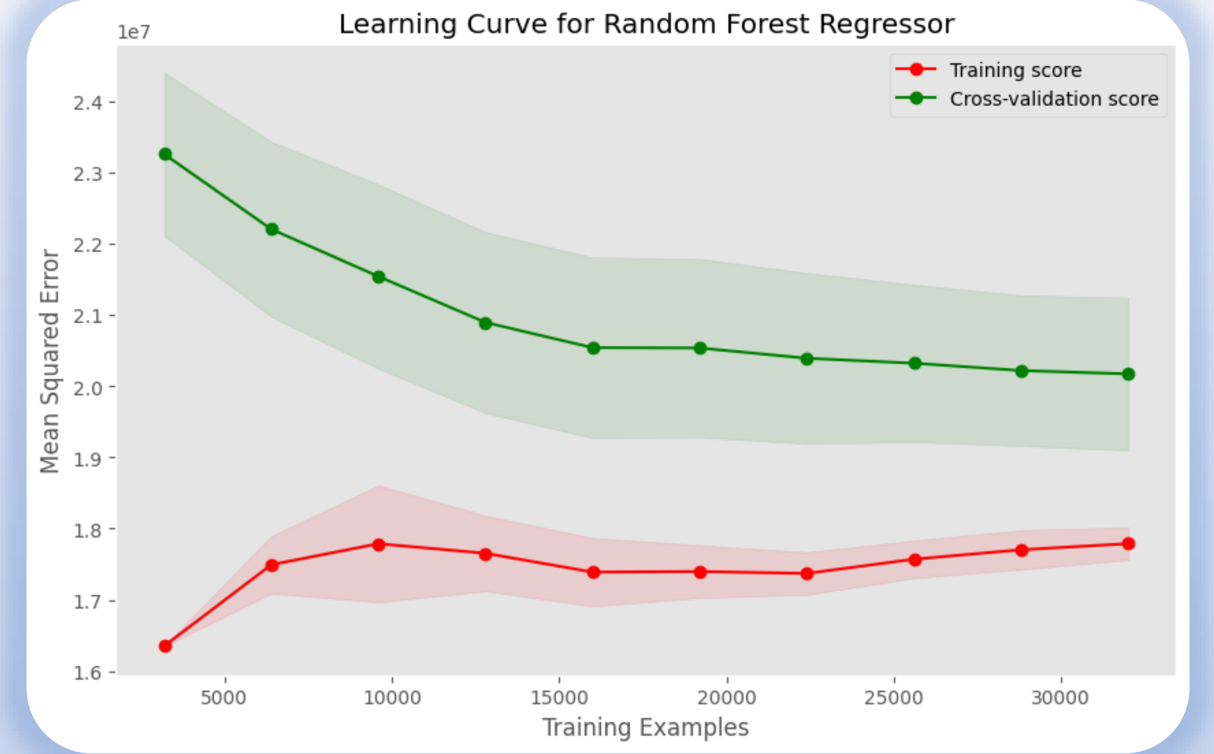
Metrics:

Train Set:
R2 Score: 0.96600
RMSE: 4171.52487
MAE: 2259.36910
MAPE: 0.14639

Test Set:
R2 Score: 0.96210
RMSE: 4434.16763
MAE: 2414.41154
MAPE: 0.15981

The most important features:





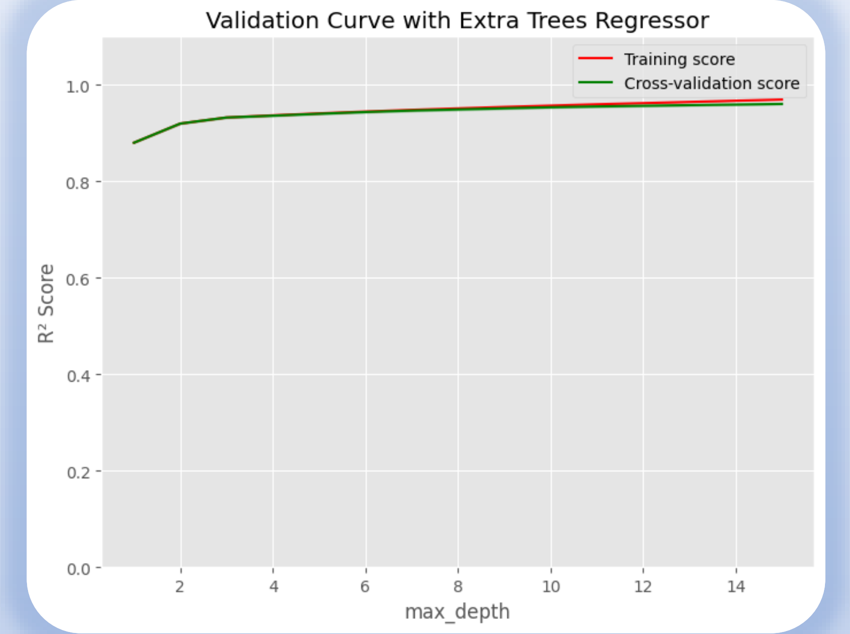
Elapsed time for HyperOpt (ExtraTrees): 574.99 seconds

Comparing this two models with different approaches to tuning, we can say that model hyperopt should be preferred because it generalizes better to unseen data, despite having a slightly lower test score compared to Model GridSearchCV

Extra Trees (Randomized Search)

After hyperparameter tuning our tree-based models performed significantly worse. That could be caused by excessively small value of max_depth. In order to solve this problem, let's fix other parameters to default values and investigate dependence of this particular hyperparameter on our model's performance using randomized search and validation curves

range from 1 to 15

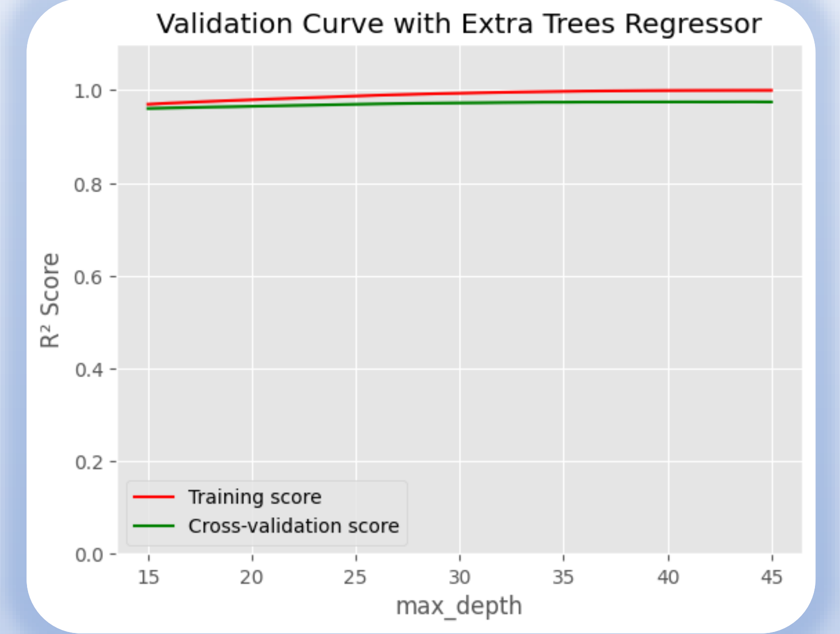


Best parameters found: {'max_depth': 15}
Elapsed time for RandomizedSearchCV (ExtraTrees): 493.89 seconds

Train Set:
R2 Score: 0.968356343
MAE: 2089.0691366367

Test Set:
R2 Score: 0.962027673
MAE: 2377.3180287230

range from 15 to 45



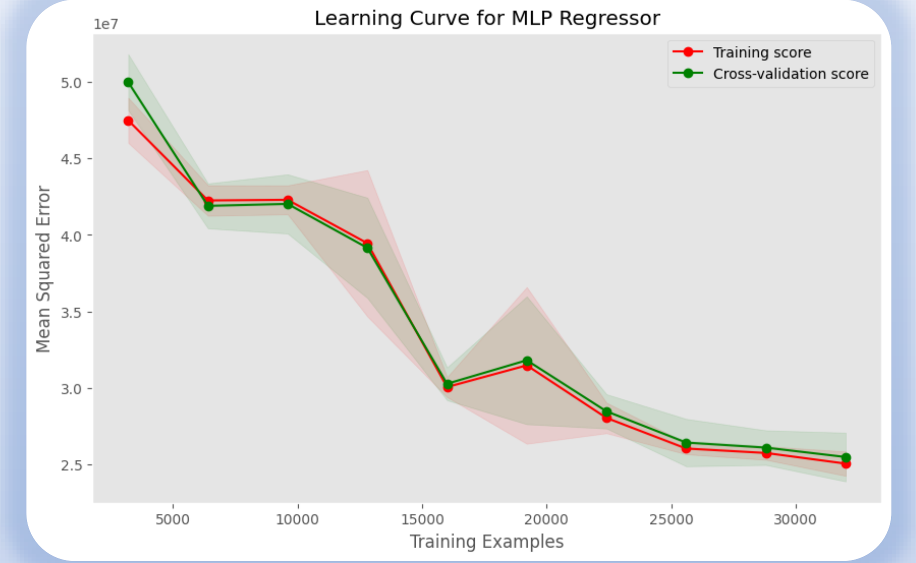
Best parameters found: {'max_depth': 43}
Elapsed time for RandomizedSearchCV (ExtraTrees): 1175.93 seconds

Train Set:
R2 Score: 0.999796924
MAE: 59.809206822365

Test Set:
R2 Score: 0.978280510
MAE: 1521.9535776855

MLPRegressor

- **StandardScaler**
- **Hyperparameters**: ' mlp__alpha' (Regularization parameter values), 'mlp__hidden_layer_sizes' (Specifies the sizes of hidden layers.) and 'mlp__learning_rate' (Learning rate schedule for weight updates)



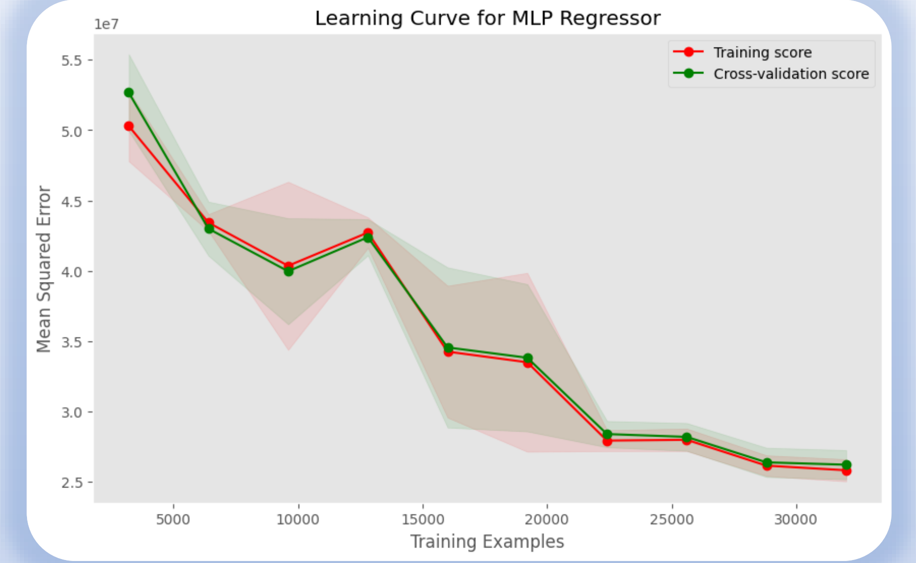
- **Elapsed time** for HyperOpt (MLP): 3139.07 seconds

Best parameters: {'alpha': 0.09459393336021493, 'hidden_layer_sizes': 160.0, 'learning_rate': 1}

Metrics:

Train Set:
R2 Score: 0.95184
RMSE: 4965.20238
MAE: 3037.64657
MAPE: 0.23875

Test Set:
R2 Score: 0.95116
RMSE: 5033.36032
MAE: 3068.09689
MAPE: 0.24246



- **Elapsed time** for GridSearch (MLP): 4456.03 seconds

Best parameters: {'mlp__alpha': 0.001, 'mlp__hidden_layer_sizes': 150, 'mlp__learning_rate': 'constant'}

Metrics:

Train Set:
R2 Score: 0.95320
RMSE: 4894.24894
MAE: 2986.88287
MAPE: 0.23828

Test Set:
R2 Score: 0.95243
RMSE: 4967.40218
MAE: 3021.79740
MAPE: 0.24178

Performance comparison

Let's compare metrics of our models after hyperparameter tuning

	Model	Train R2	Train RMSE	Train MAE	Train MAPE	Test R2	Test RMSE	Test MAE	Test MAPE
2	Random Forest GS	0.966672	4143.375294	2188.265412	0.138327	0.963187	4356.168140	2308.280980	0.145338
4	Extra Trees HO	0.962601	4389.176083	2351.634092	0.149540	0.959944	4544.017154	2430.530553	0.153473
3	XGB Regressor GS	0.964557	4272.810608	2473.447299	0.173663	0.963441	4341.129827	2493.006918	0.173411
5	MLP Regressor HO	0.960207	4527.444746	2666.242901	0.201741	0.958861	4605.038760	2694.572714	0.201859
1	Huber Regressor	0.904745	7004.787212	4147.580287	0.302286	0.904918	7000.943566	4117.637657	0.298852
0	Polynomial Regression	0.884536	7712.137047	4559.936490	0.263783	0.883653	7744.342738	4570.301101	0.263254

- After fine-tuning our system, we can point out that tree-based models performed significantly worse in comparison to no hyperparameters tuning.
- In contrast, MLP showed better scores with hyperparameters obtained after tuning.
- Huber and Polynomial Regressions showed approximately same scores as before, with a slight decrease of MAE.

Performance comparison

To get the best possible result, let's train tree-based models with default settings on a full training set.

	Model	Train R2	Train RMSE	Train MAE	Train MAPE	Test R2	Test RMSE	Test MAE	Test MAPE
0	Random Forest	0.997497	1135.482753	425.240167	0.027201	0.984850	2794.580479	1092.769480	0.070849
2	Extra Trees	0.999286	606.248974	58.646549	0.002689	0.982232	3026.398084	1152.873922	0.075952
1	XGB Regressor	0.978581	3321.656079	1895.060093	0.139592	0.976690	3466.417709	1957.925946	0.142162

- **Preferred Models:** Even though the tree-based models with tuned hyperparameters (Random Forest GS, Extra Trees HO, XGB Regressor GS) seem to have better balance between training and test performance, models with default settings are preferable, due to the significantly better performance.
- **Overfitting Models:** The "plain" models exhibit signs of overfitting. But their excellent performance in combination with significant amount of test data, give us reason to believe that they would be a better choice.