



FLIGHT PRICE PREDICTION

ĐẶNG THỊ BÍCH PHƯƠNG – NHÓM 4

AGENDA

Data Overview

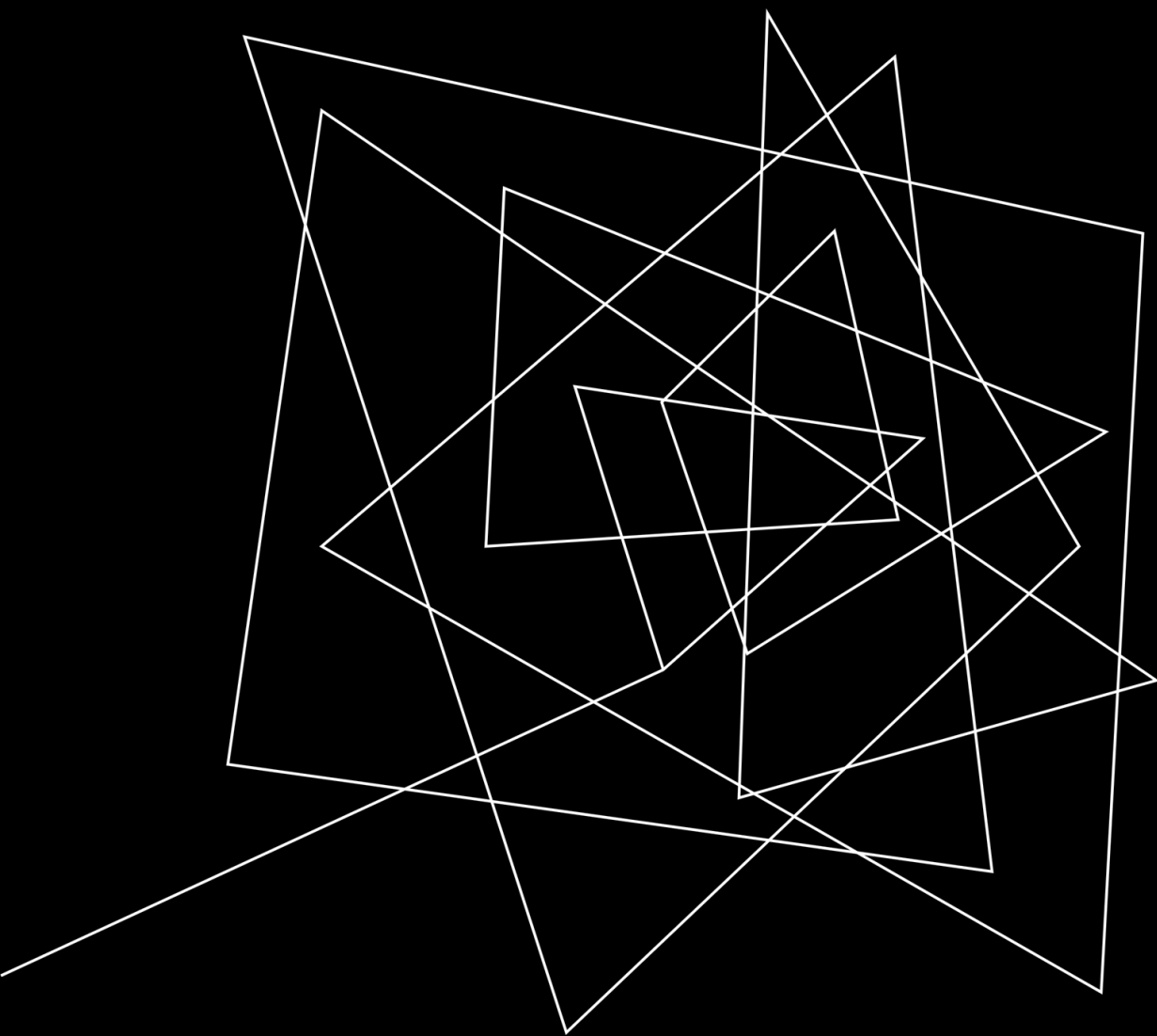
Data Cleaning

Data Exploration

Data Preprocessing

Build Models

Conclusion



DATA OVERVIEW

DATA OVERVIEW

The dataset used in this project was sourced from Kaggle, specifically from the dataset titled "Flight Price Prediction". It contains comprehensive information about flight booking options from the "Ease My Trip" website for flight travel between India's top 6 metro cities.

The dataset consists of 5000 distinct flight booking options and spans a 50-day period, from February 11th to March 31st, 2022. It was curated for the purpose of conducting data analysis and building predictive models to forecast flight prices accurately.

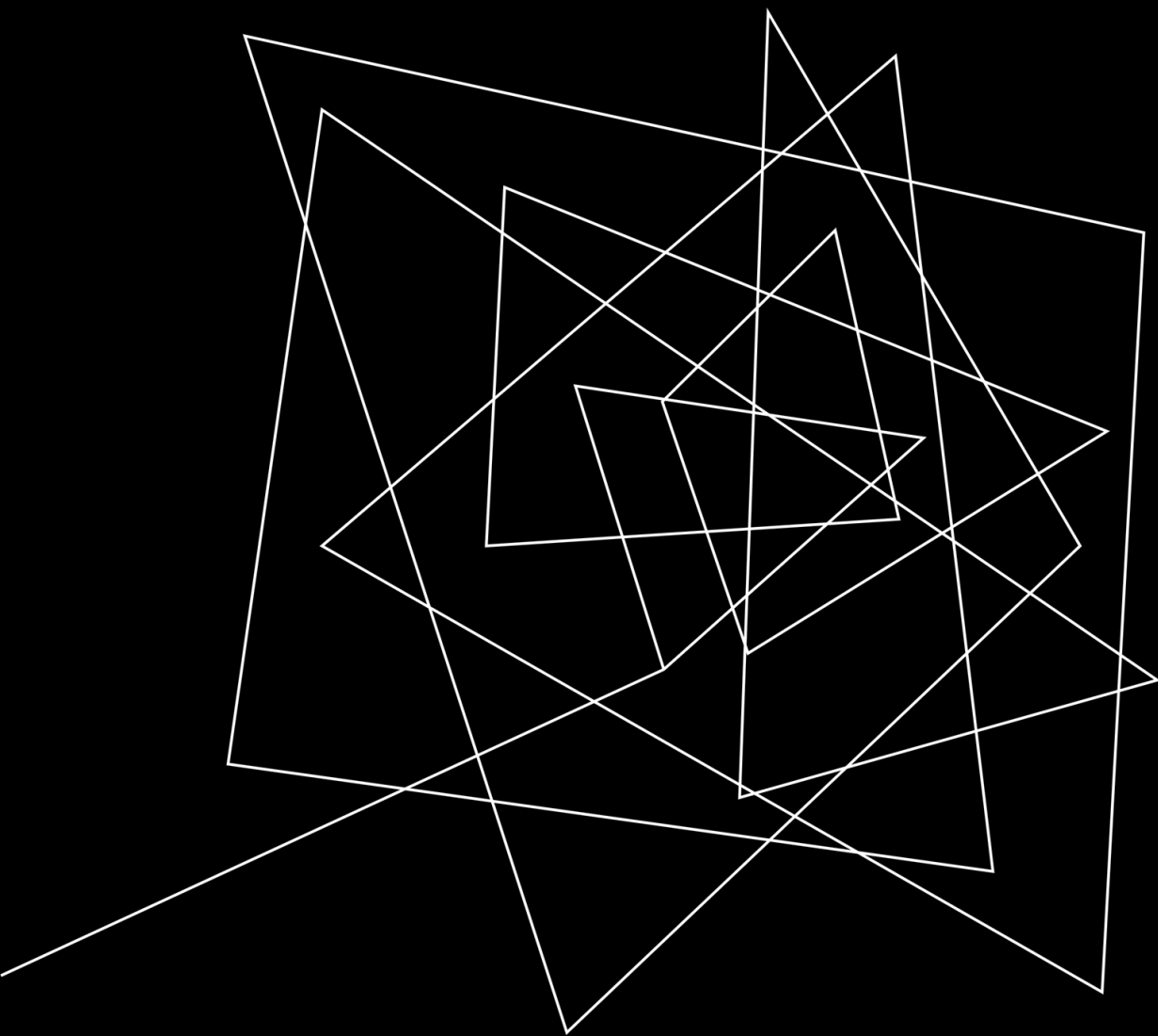


INFORMATION ABOUT DATA FEATURES

- **Airline:** Name of the airline
- **Flight:** The plane's flight code
- **Source City:** City from which the flight takes off
- **Departure Time:** The departure time of plan
- **Stops:** The number of stops between the source and destination cities
- **Arrival Time:** The arrival time of plan
- **Destination City:** City where the flight will land
- **Class:** Seat class (Business and Economy)
- **Duration:** The overall amount of time it takes to travel between cities in hours
- **Days Left:** Number of days subtract the trip date by the booking date
- **Price:** The ticket price

OBJECTIVE OF THE PROJECT

The project's goal is to analyze existing data sets to accurately predict flight prices and reveal factors that influence airfare prices. Thereby helping travelers search and choose flights that suit their needs.

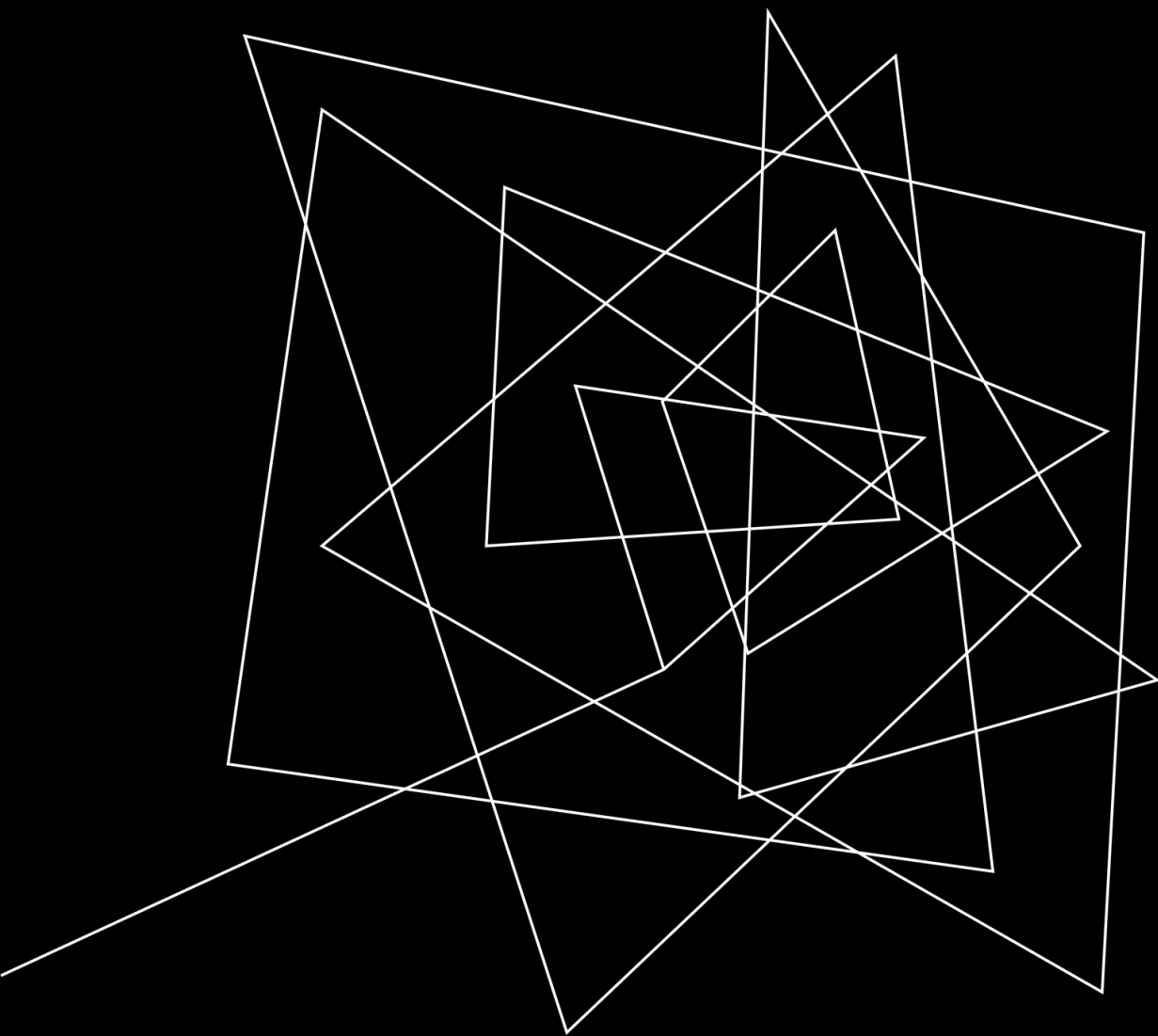


DATA CLEANING

REMOVE COLUMNS

```
[ ] # Remove redundant columns
    sampled_df = sampled_df.drop(columns=["Unnamed: 0", "flight"])
```

In the dataset, there are two columns "*Unnamed: 0*" and "*flight*" that do not affect the dependent variable "*Price*" so will be omitted.

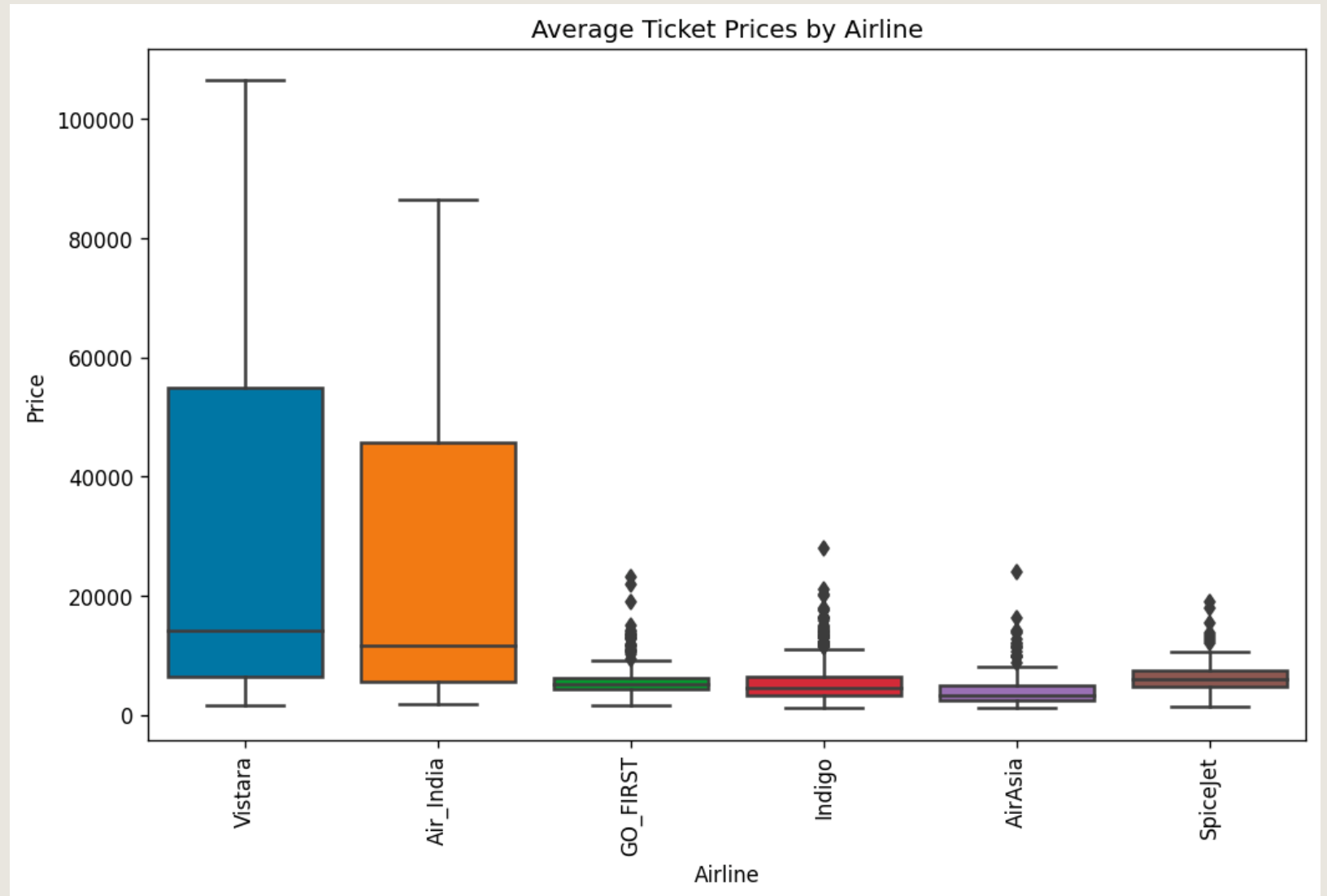


DATA EXPLORATION

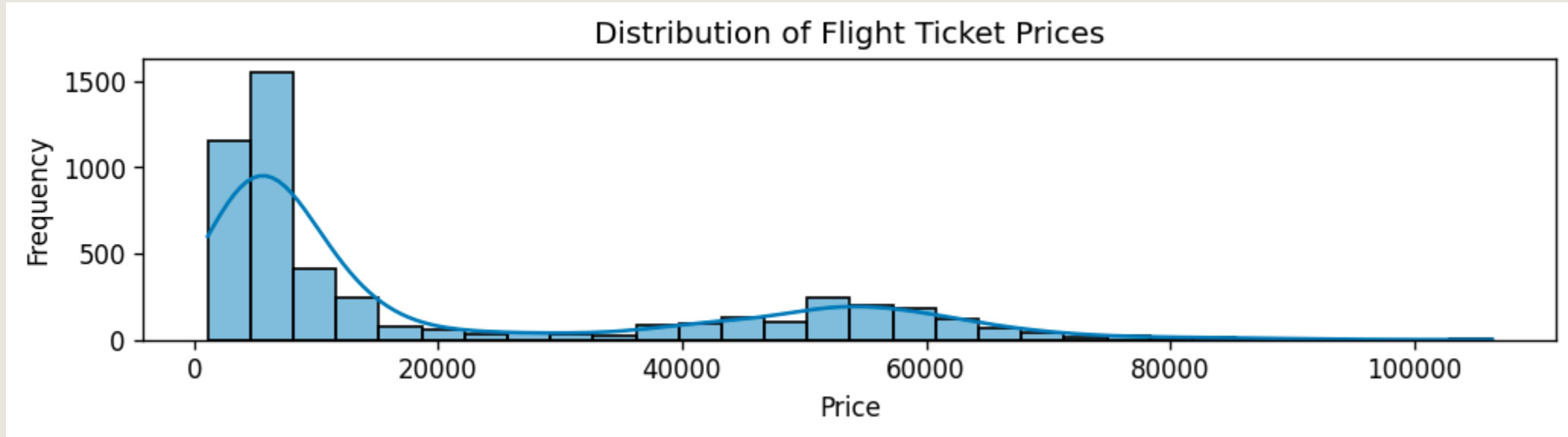
AIRLINE VARIABLE

High airfare prices will belong to two airlines **Vistara** and **Air_India**.

Airline ticket prices are divided into two distinct segments, low and high.

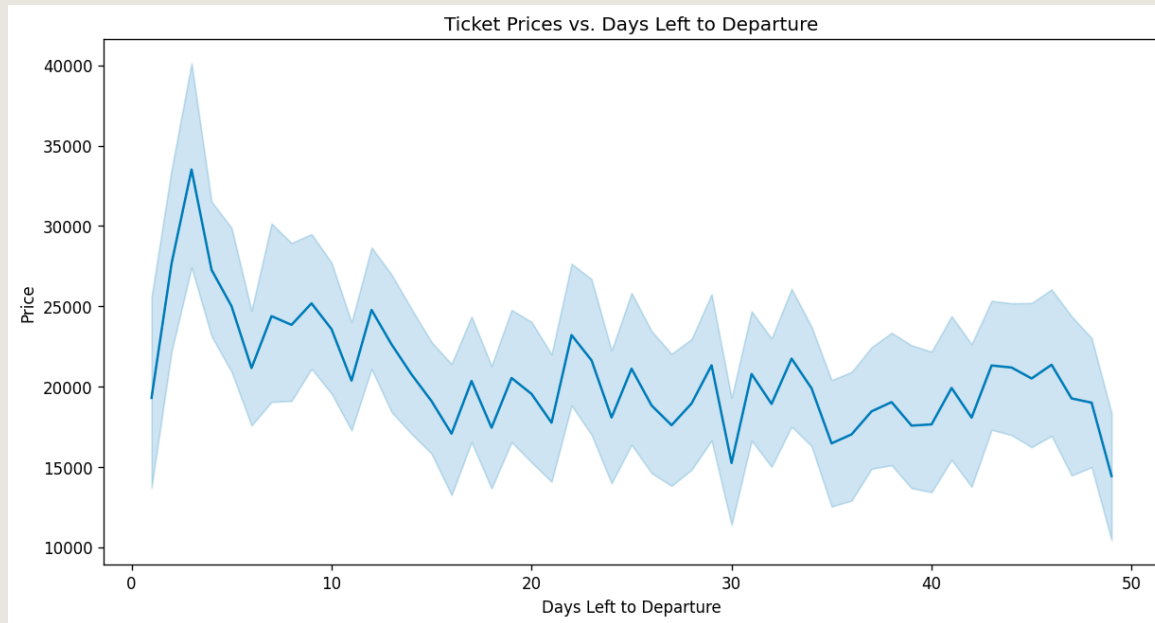


PRICE VARIABLE

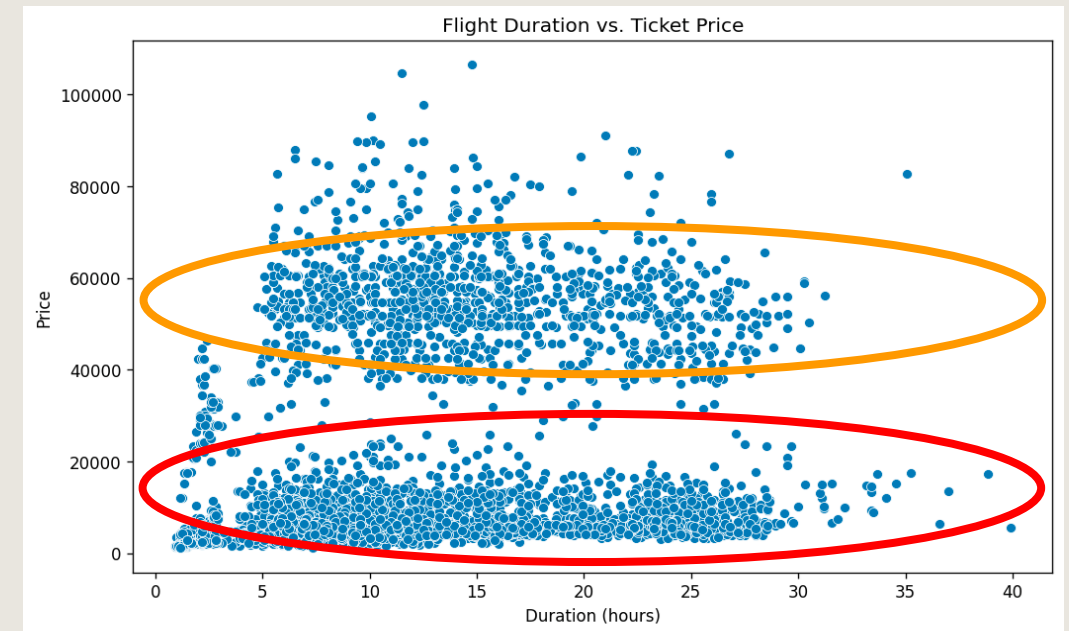


Airline ticket prices are being divided into two distinct segments: **low price** (0-20,000) and **high price** (40,000-70,000).

GROUP VARIABLE ABOUT TIME

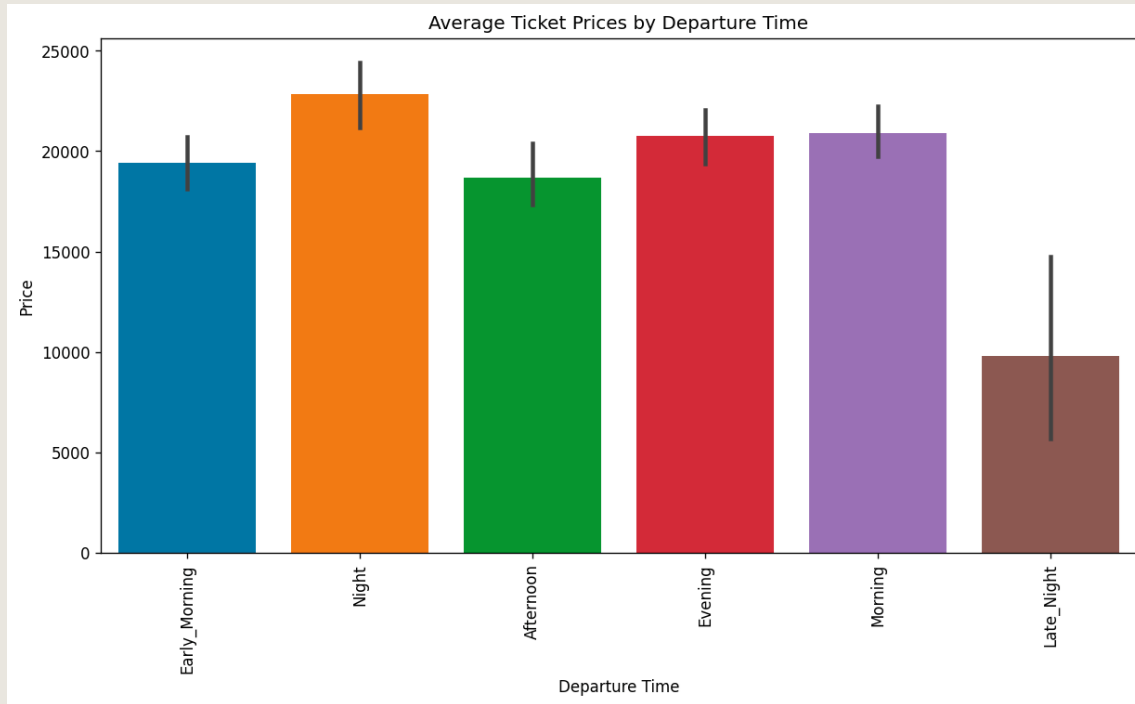


Airline ticket prices tend to **decrease** with the number of days booked before the flight. The closer to the date tourists book tickets, the higher the airfare will be and vice versa.

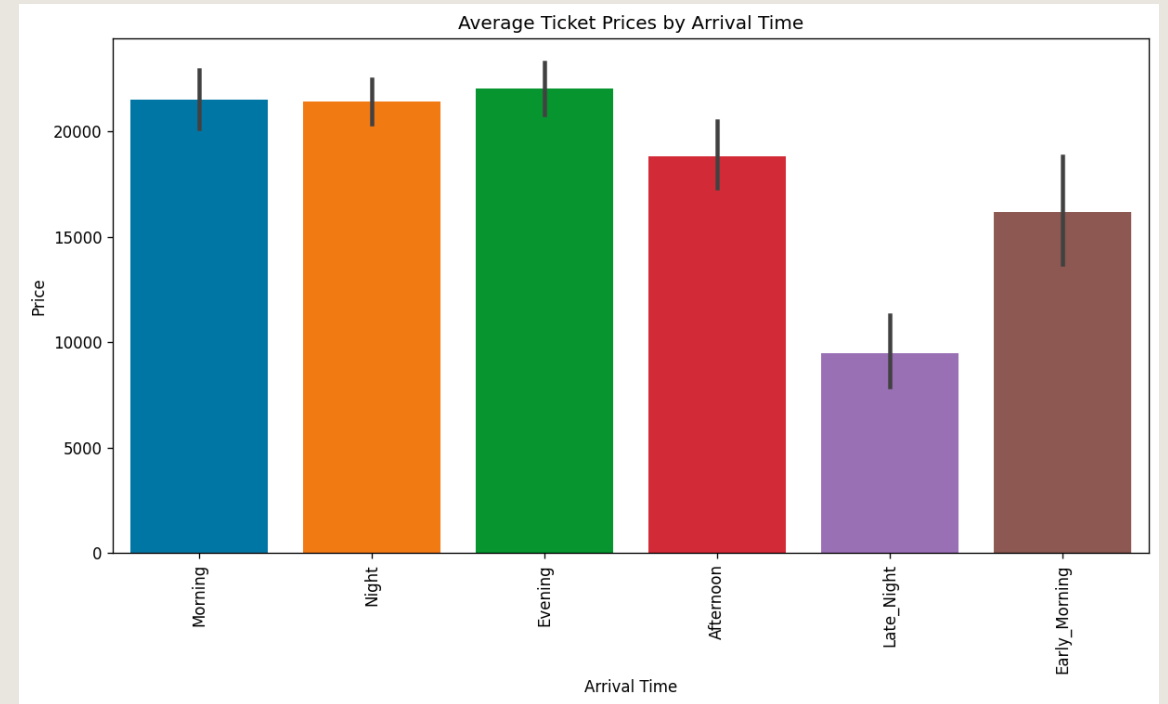


The duration feature tends to be distributed into **two groups**: **low price** and **high price**.

GROUP VARIABLE ABOUT TIME



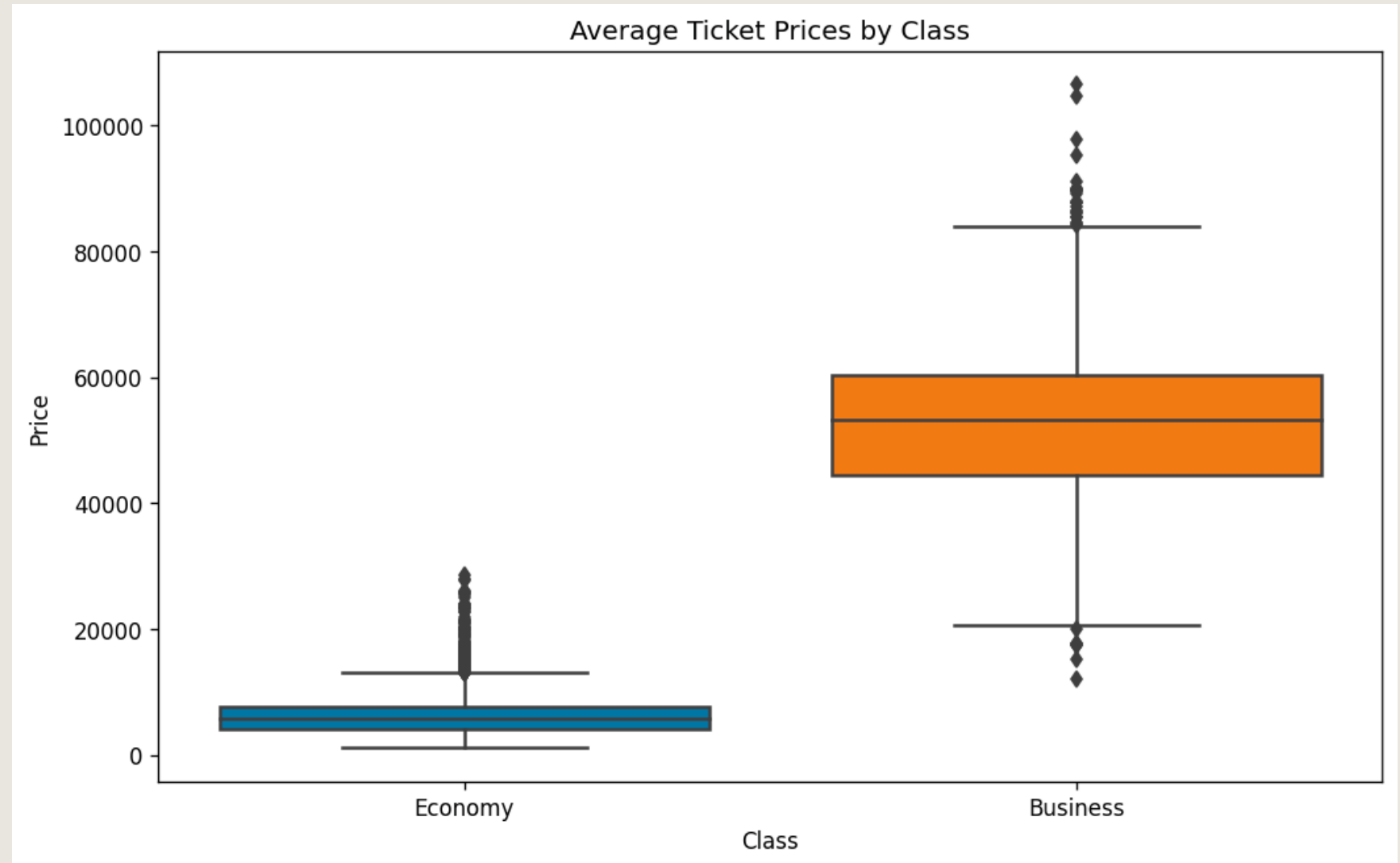
Arrival time at Night will have the highest ticket price and Late_Night will have the cheapest ticket price. However, there is no significant difference in airfare between arrival times.

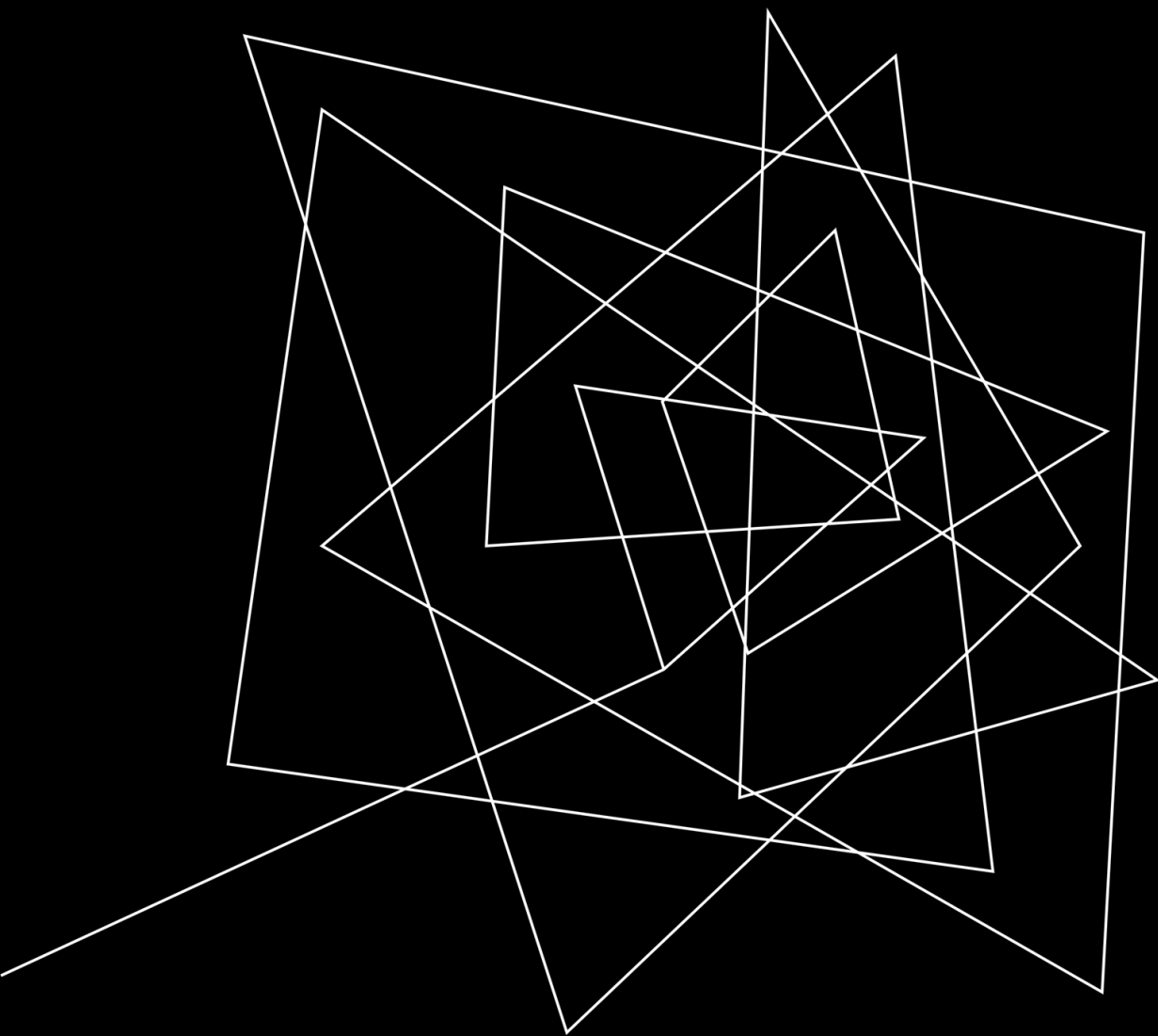


Evening departure times will have the highest ticket prices and Late_Night will have the cheapest prices. There is also not much difference in ticket prices between departure times.

CLASS VARIABLE

The average airline ticket price for **Business** class is **much higher** than the price for **Economy** class.





DATA PREPROCESSING

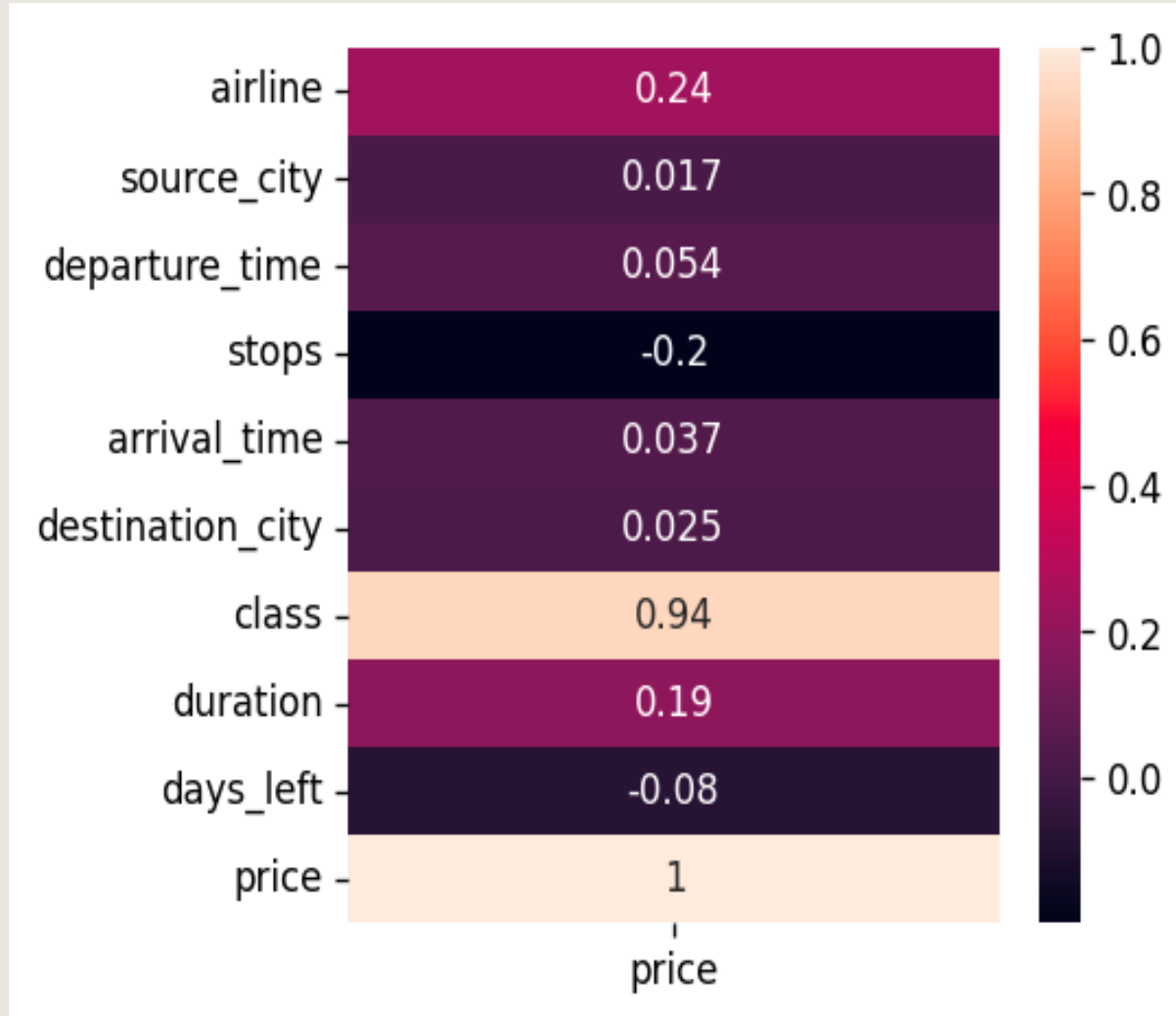
ENCODING

```
▶ #encoder str feature
from sklearn.preprocessing import LabelEncoder
lec = LabelEncoder()
for col, dtype in list(sampled_df.dtypes.items()):
    if (dtype == 'object' and col != 'class'):
        sampled_df[col] = lec.fit_transform(sampled_df[col])

sampled_df
```

Encoding data in the columns airline, flight, source_city, departure_time, stops, arrival_time, destination_city, class

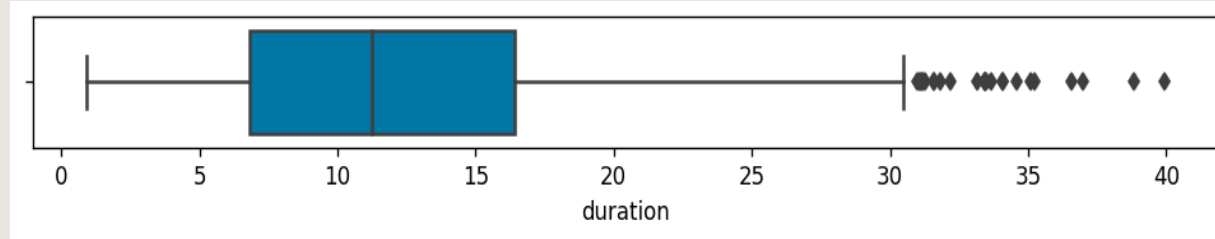
CORRELATION ANALYSIS



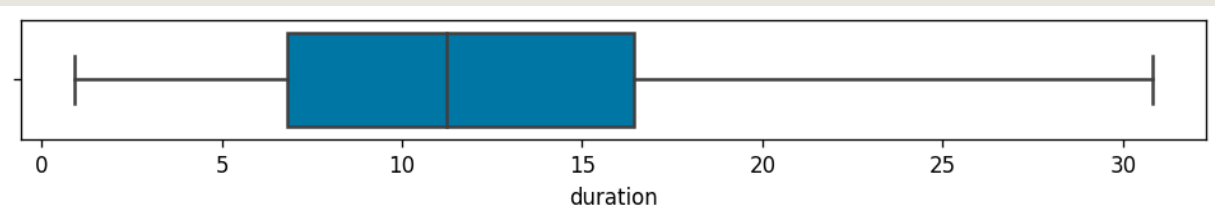
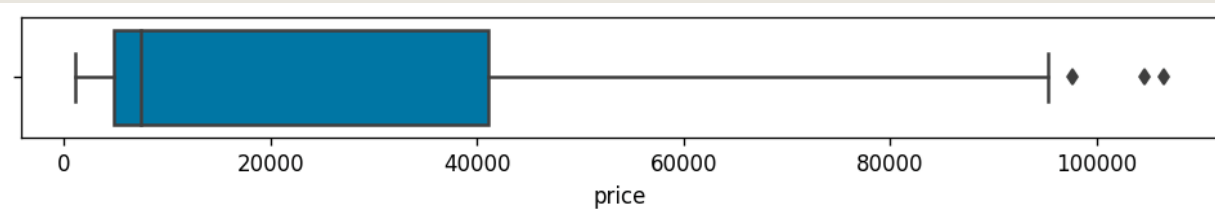
Generate a heatmap of variables against the dependent variable (price) to assess the correlation between the variables and the dependent variable.

→ Based on the correlation analysis results depicted on the heatmap, it is observed that none of the variables exhibit exceptionally low correlation, suggesting that there is no immediate need to eliminate any variables from the dataset.

OUTLIER HANDLING



→ The Box plot constructed for the variables reveals the potential presence of outliers in the duration and price variables.



→ Apply the Interquartile Range (IQR) method to handle outliers.



DATA PREPARATION

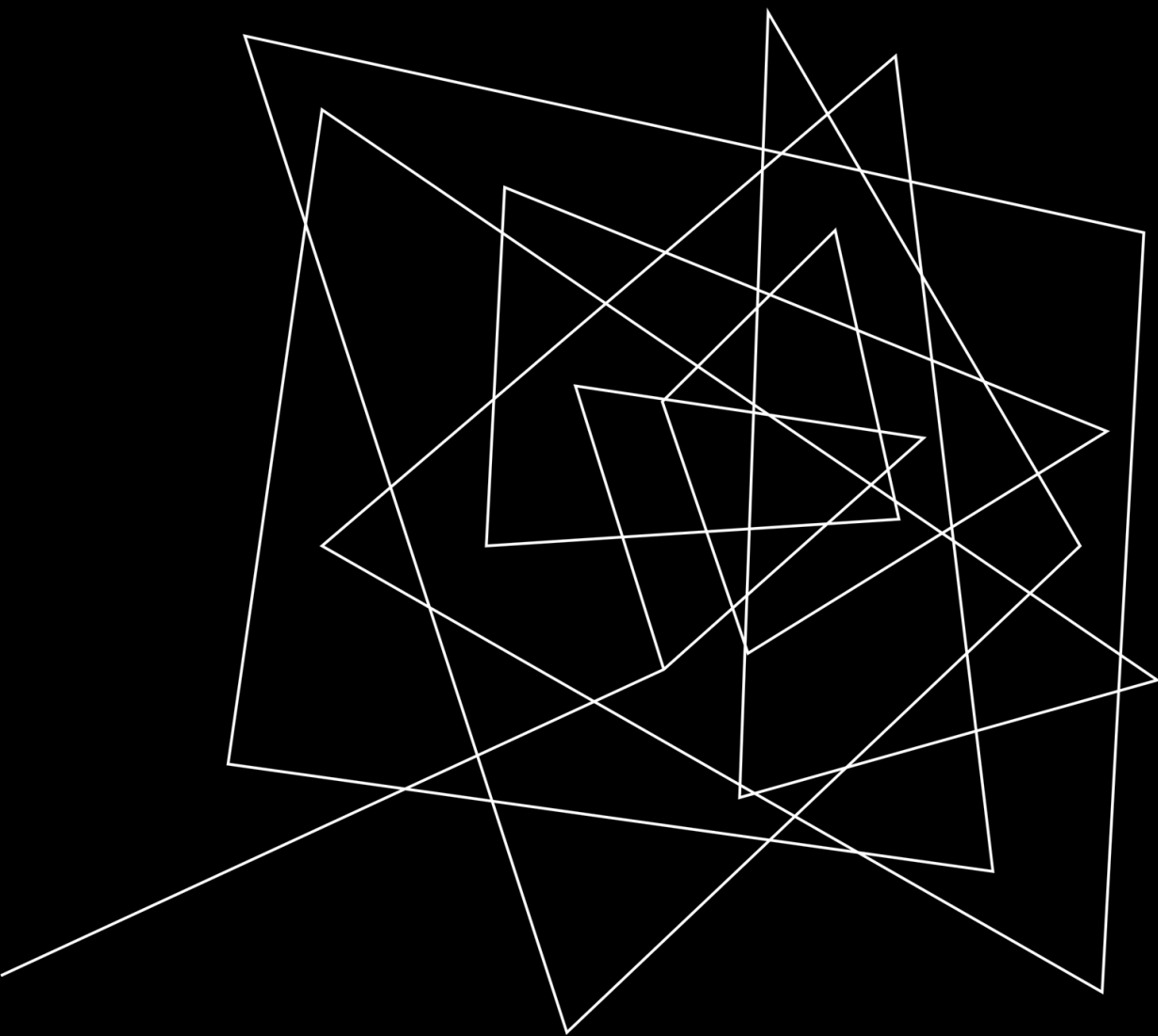
```
▶ #Split data
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.3, random_state=42
)
```

Split train and test files to build models

```
▶ from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Normalize data of continuous variables on sets X_train and X_test



BUILD MODELS

- 1. Linear Regression**
- 2. Support Vector Regressor**
- 3. Random Forest**
- 4. Decision Tree**

LINEAR REGRESSION

```
▶ y_test_pred = model_multiple.predict(X_test)
  from sklearn.metrics import r2_score
  acc = r2_score(y_test, y_test_pred)

  print('R2_score:', acc)

  mse = mean_squared_error(y_test, y_test_pred)
  rmse = math.sqrt(mean_squared_error(y_test, y_test_pred))

  print('MSE:', mse)
  print('RMSE:', rmse)
```

r2_score	MSE	RMSE
0.905	47187670.926	6869.328

The result is yielded by the Linear Regression model

SUPPORT VECTOR REGRESSION

```
from sklearn.svm import SVR
model_svr = SVR(kernel='linear', C=0.1) # set kernel and epsilon
model_svr.fit(X_train, y_train.values.ravel())
y_svr_pred = model_svr.predict(X_test)
```

```
#Tính toán độ chính xác
from sklearn.metrics import r2_score
Acc_svr = r2_score(y_test.values.ravel(), y_svr_pred)
print('The accuracy of this model is:R-square=', "{:.2f}".format(Acc_svr))
```

```
mse_svr = mean_squared_error(y_test, y_svr_pred)
rmse_svr = math.sqrt(mean_squared_error(y_test, y_svr_pred))

print('MSE:', mse_svr)
print('RMSE:', rmse_svr)
```

The Support Vector Regression model gives the following results:

r2_score	MSE	RMSE
-0.33	658171147.722	25654.846

➔ The model gave poor results, **r2_score** was low, **MSE** and **RMSE** were high even though **GridSearchCV** was run.

RANDOM FOREST

```
y_test_rf_pred = model_rf.predict(X_test)

acc_test_rf = r2_score(y_test.values.ravel(), y_test_rf_pred)

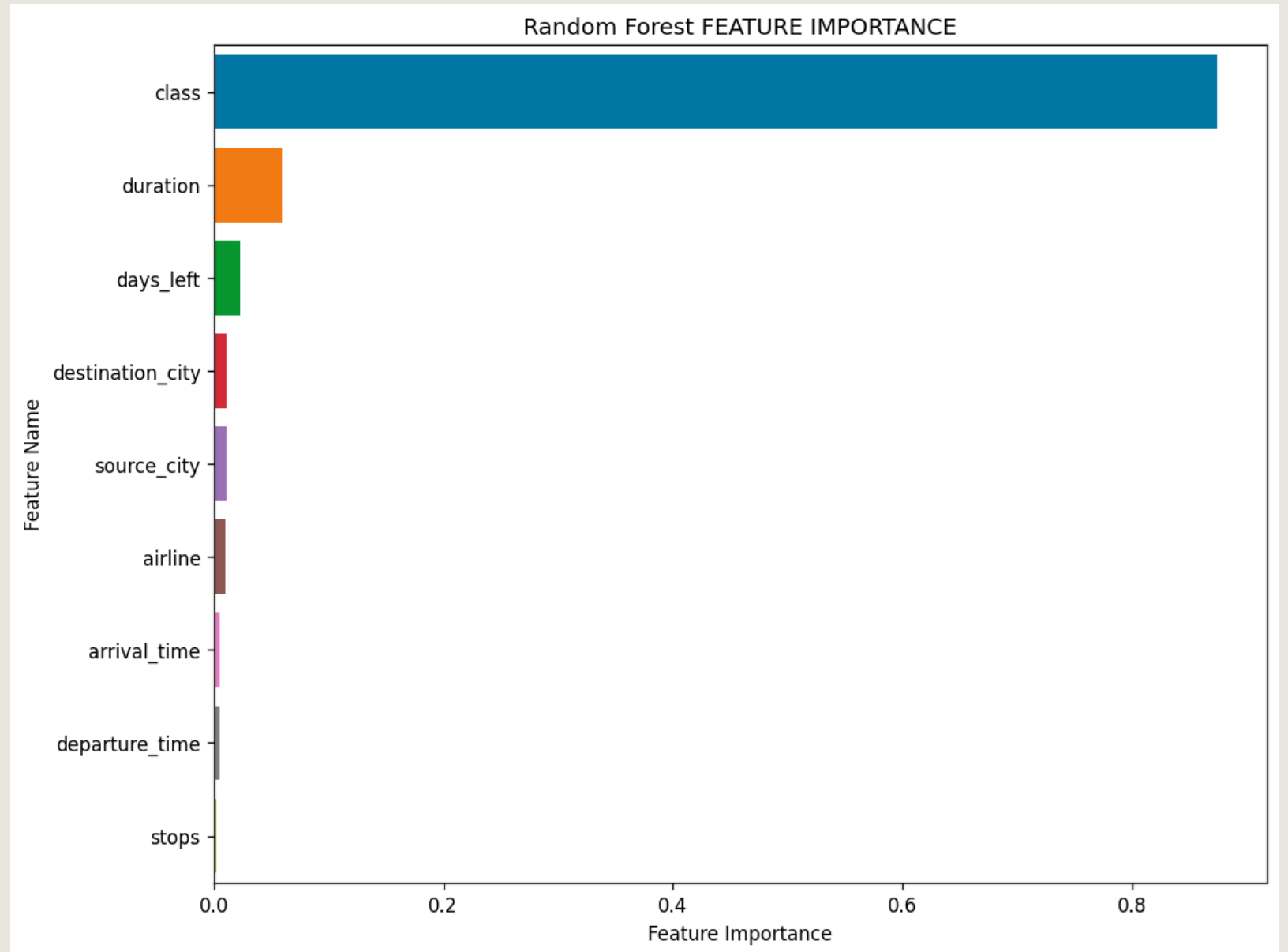
print('R2_score_rf:', acc_test_rf)

mse_test_rf = mean_squared_error(y_test.values.ravel(), y_test_rf_pred)
rmse_test_rf = math.sqrt(mean_squared_error(y_test.values.ravel(), y_test_rf_pred))
print(mse_test_rf)
print(rmse_test_rf)
```

r2_score	MSE	RMSE
0.946	26826894.29	5179.47

The result is yielded by the Random Forest Regression model

The important features that most affect the **Random Forest** model are **class**, **duration** and **days left**



DECISION TREE

```
y_test_tree_pred = model_tree.predict(X_test)

acc_test_tree = r2_score(y_test.values.ravel(), y_test_tree_pred)

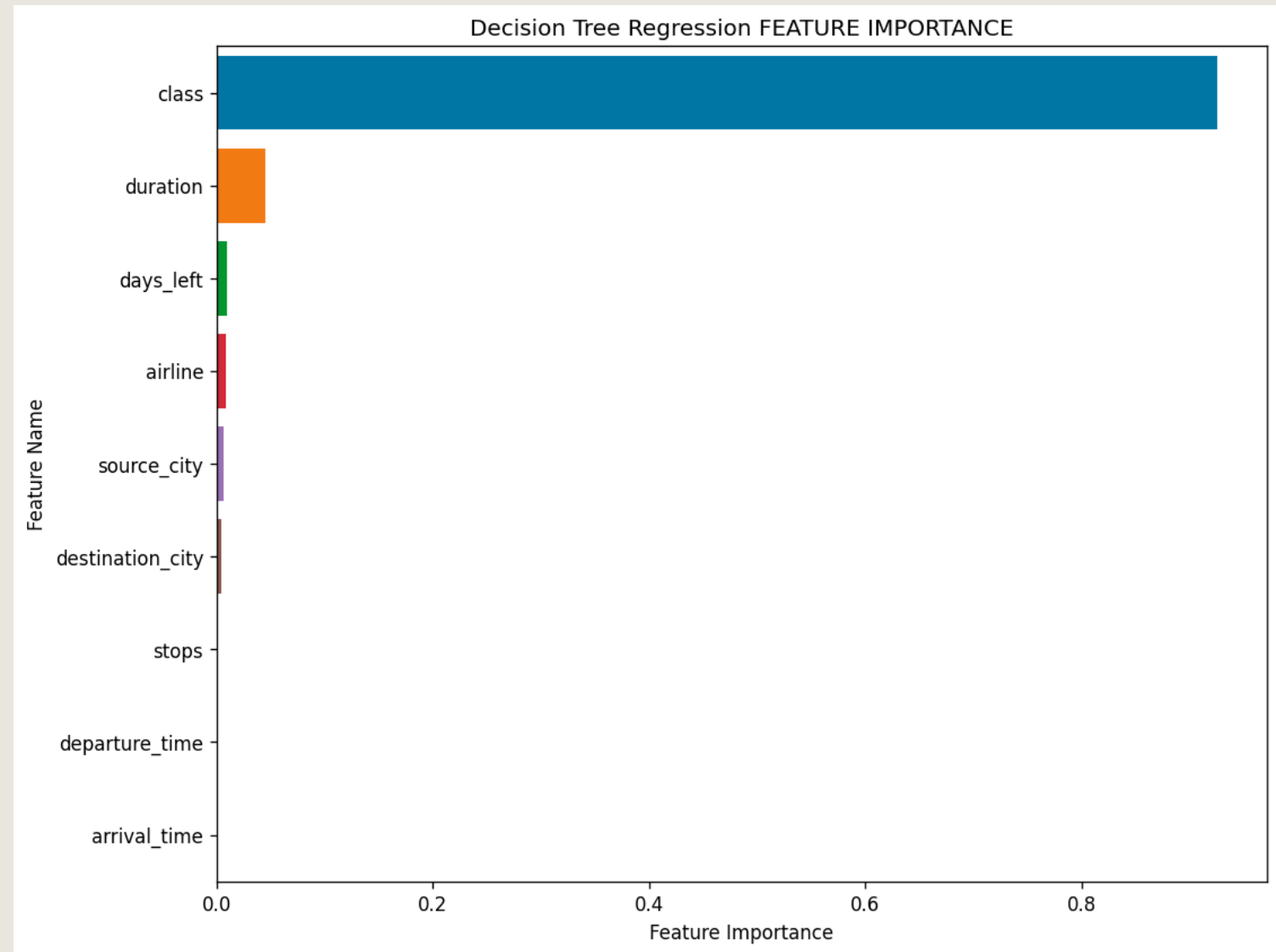
print('R2_score_tree_test:', acc_test_tree)

mse_test_tree = mean_squared_error(y_test.values.ravel(), y_test_tree_pred)
rmse_test_tree = math.sqrt(mean_squared_error(y_test.values.ravel(), y_test_tree_pred))
print(mse_test_tree)
print(rmse_test_tree)
```

r2_score	MSE	RMSE
0.938	29743151.871	5453.728

The result is yielded by the Decision Tree model

The important attributes that most affect the Decision Tree model are **class**, **duration** and **days left**



COMPARE MODELS

Model	r2_score	MSE	RMSE
Linear Regression	0.905	47187670.926	6869.328
Support Vector Regressor	-0.33	658171147.722	25654.846
Random Forest	0.946	26826894.29	5179.47
Decision Tree	0.938	29743151.871	5453.728

Based on the following statistical table, it can be seen that the **r2_score** of the **Random Forest** model is the highest at **0.946**. Besides, the **MSE** and **RMSE** of this model are also at the lowest level.

➡ **Random Forest is the best model for research.**

FINE-TUNING

```
# fine-tuning model
from sklearn.model_selection import RandomizedSearchCV
space = {
    'n_estimators': [50, 100, 150, 250, 1000],
    # 'learning_rate': [0.01, 0.1, 1],
}
rand_search_rf = RandomizedSearchCV(
    estimator=model_rf,
    param_distributions=space,
    n_iter=10,
    cv=3,
    verbose=3
)

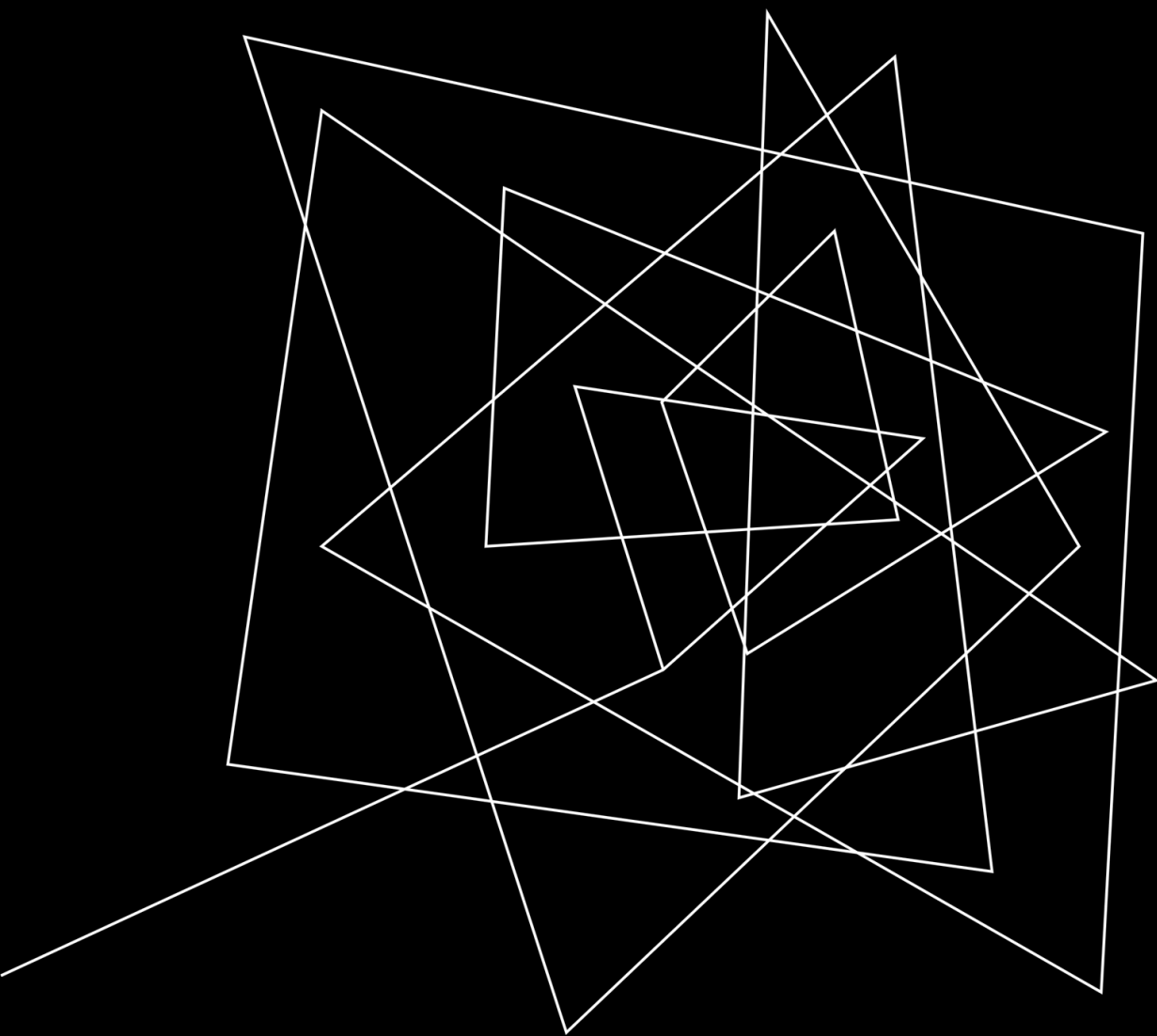
rand_search_rf.fit(X_train, y_train)

rand_search_rf.best_estimator_
```

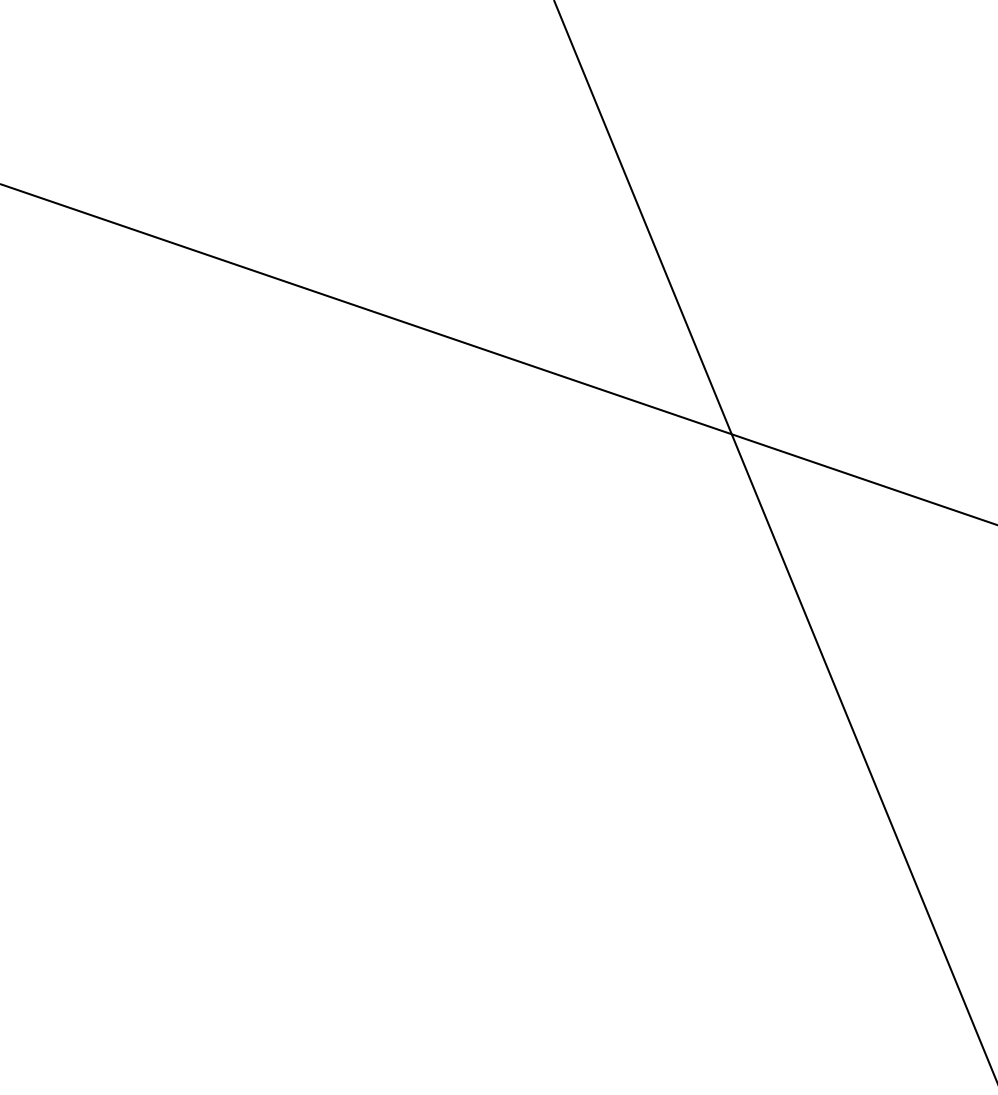
```
# evaluate fine-tuned
from sklearn.metrics import r2_score, mean_squared_error
y_pred_rf_ft = rand_search_rf.predict(X_test)
print(r2_score(y_test.values.ravel(), y_pred_rf_ft))
print(mean_squared_error(y_test.values.ravel(), y_pred_rf_ft))
print(math.sqrt(mean_squared_error(y_test.values.ravel(), y_pred_rf_ft)))
```

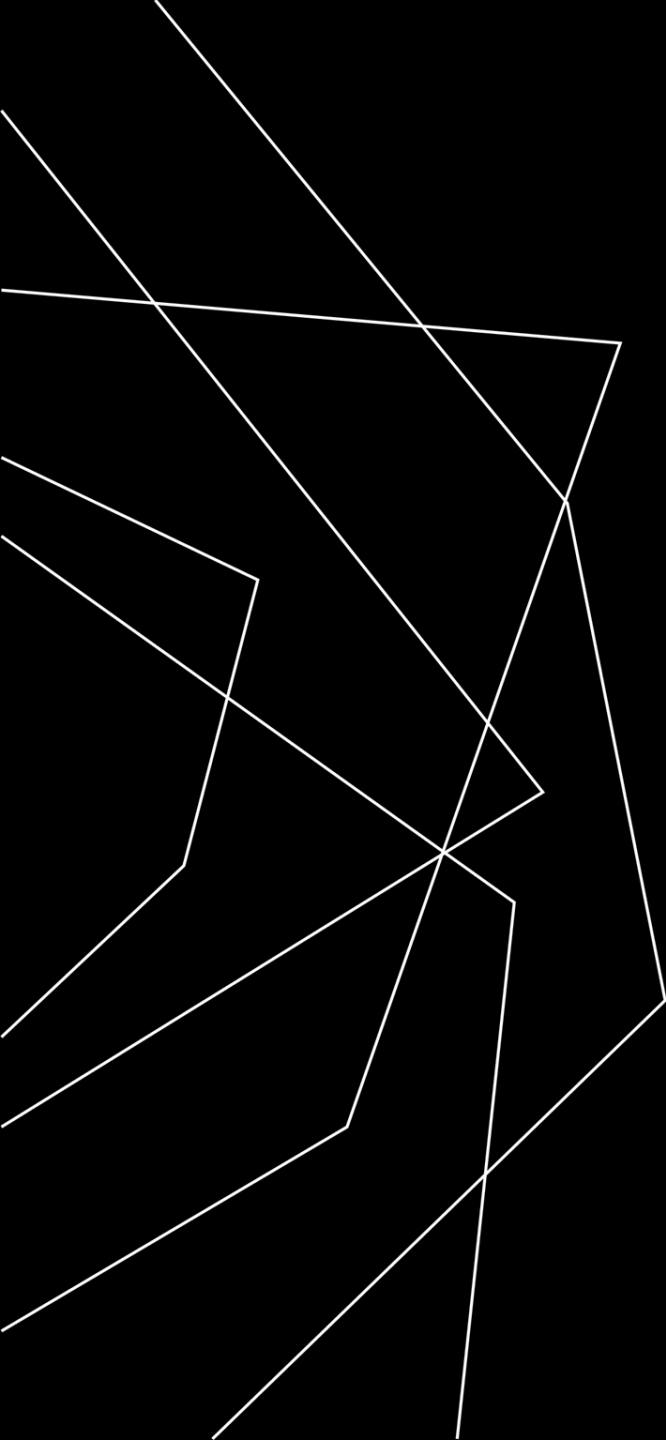
r2_score	MSE	RMSE
0.95	24608451.03	4960.69

Optimize the **Random Forest** model to achieve the most optimal result by metrics **RandomizedSearchCV**.



CONCLUSION

- 
- Based on the analysis => **Random Forest** is the most optimal model to predict air ticket prices.
 - The variables that most affect airfare prices are **class**, **duration** and **days_left** => Travelers need to pay attention to these attributes to choose to buy air tickets at the best price.
 - Based on the price distribution chart, it can be seen that airfare prices are divided into 2 segments: **low price** and **high price** => build a classification model for better evaluation.



THANK YOU