CMPE 255 Fall 2024

Project Report: Flight Prices Classification

Team Name: Data Mining Developers
Team Members: Vivek Ponnala (**SID** 017550819), Maggie Qin (**SID** 018207657), Hoai An Nguyen (**SID** 014719549)
Project Title: Flight Prices Classification: An Approach Using Random Forest, Decision Trees, and K-Nearest Neighbors

Introduction

Motivation

Frequent fluctuations in airline ticket prices affect both the consumer's ability to budget and the airline industry's revenue optimization. Understanding these fluctuations is critical for enabling better price predictions and decision-making. Our motivation is to provide insights into factors driving ticket prices and to identify patterns that can help predict future price trends, benefiting both consumers and industry professionals [1]-[6].

Objective

Our objective is to build a robust classification model that categorizes flight prices into three ranges via grouping each price by binning: low, medium, and high. Structuring the prices in such categories makes it possible to design a model that would help solve the problem of discerning the sensitivity of traveling to price changes and make relevant allocation of resources in the airline industry [7].

Literature Review

Several studies have explored airline pricing models, often using machine learning to uncover price determinants. Random Forest, Support Vector Machines, and Neural Networks are frequently applied due to their effectiveness in handling complex data [1]-[4]. Inspired by these studies, we selected the Random Forest, Decision Tree, and K-Nearest Neighbors.

System Design & Implementation Details

Algorithm Selection

Random Forest was selected due to its suitability when dealing with datasets that have both numerical and categorical variables, besides being capable of giving the feature importance [1]. Decision Tree is interpretably efficient for ranking the importance of features [2], whereas KNN can handle nonlinear interactions, providing an appropriate baseline [3].

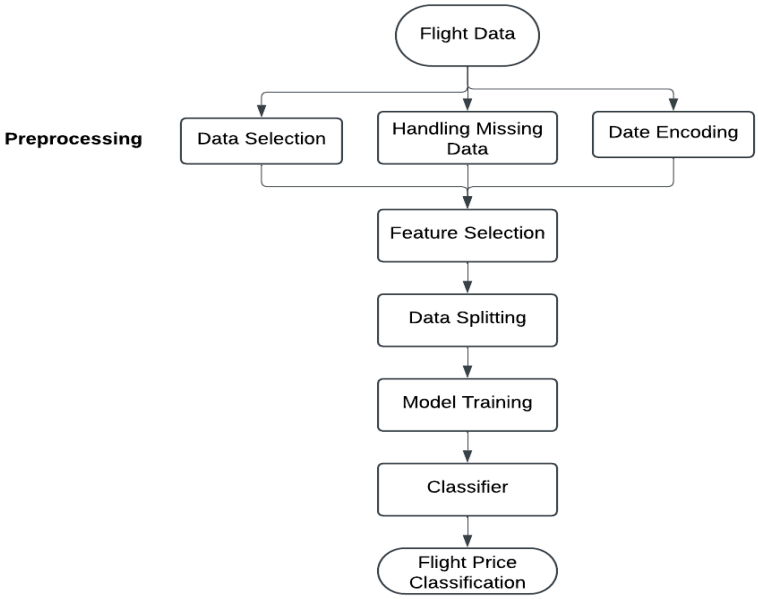
Technologies

Python, pandas, NumPy, and scikit-learn for data processing and modeling, with matplotlib and seaborn for visualization [5].

Data Pipeline

Variables were normalized and some created based on other variables like weekend indicator Variable names were edited for readability and then split the data 80/20 training and testing. Models were tested and accessed in order to determine their performance, as well as the importance of features in the process [6].

Architecture Diagram



Experiments: Dataset Description and Feature Engineering

Dataset Overview

The dataset was sourced from Kaggle and includes over 5 million records of flight ticket prices found on Expedia from April to October 2022. We used a subset focused on nonstop flights departing from different cities: DEN, DTW, EWR, IAD, JFK, LAX, LGA, MIA, OAK, ORD, PHL, SFO. Totaling to 206449 records with 25 columns [7].

Preprocessing

Data preprocessing involved filtering rows with missing or irrelevant values, converting date features, and encoding categorical variables [5].

Selected Features:

- **seatsRemaining**: Represents the number of seats left, highlighting supply-demand trends.
- **searchDate**: The date of the search, which can influence price based on seasonality.
- **totalTravelDistance**: Total distance of the flight, directly correlated with ticket prices.
- **isBasicEconomy**: A binary variable indicating whether the flight is a basic economy fare, a factor known to impact pricing [6].

Feature engineering:

- **isWeekend**: A derived feature from searchDate, indicating whether the search happened on a weekend. Created by extracting the day of the week from searchDate. Encoded as a binary variable (1 = Weekend, 0 = Weekday) to reflect demand spikes during weekends.
- **day**: Extracted from the date, capturing daily fluctuations in ticket prices.
- **month**: Reflects seasonal demand and pricing patterns.
- **year**: A higher-level indicator of the time frame (though it likely remains constant in this dataset subset)
- **daysBeforeFlight**: Number of days between search and flight date, to reflect the relationship between booking and departure date
- **searchDayOfWeek**: extracted from searchDate to capture patterns in pricing behavior based on day of the week. It converts days of the week into integers 0 = Monday, 6 = Sunday
- **daysBeforeFlight**: Represents the number of days between the date a flight ticket search was made and the scheduled departure date of the flight.

Price Quantiles:

- Flight prices were divided into three quantiles: low, medium, and high.
- This transformation converts a continuous price variable into a classification target, by dividing the values in the baseFare column into three equal-sized quantile-based bins.

Categorical Encoding: Binary encoding was applied to features like isWeekend and isBasicEconomy to enable them to work seamlessly with the models.

Feature Scaling: Continuous features are standardized with StandardScaler, ensuring that all numeric inputs have comparable scales and improving the performance of models sensitive to feature magnitudes [5].

Data Splitting

The preprocessed data was split into 80/20 Training Set and Testing Set to evaluate model performance effectively.

Model Selection and Training

Random Forest

In selecting the algorithm I was looking for one that can handle both numerical and categorical data for model building and Random Forest is known to be particularly effective for classification [1]. The model was fit with 100 trees and a maximum depth that was picked via cross-validation to ensure maximal accuracy without long training times. Data was divided with 80% of data used for training the model and the remaining 20% used for testing and validation to have a large enough sample.

Decision Tree

Decision tree was selected for this classification task due to its high interpretability. Because the decisions at each split are visualized, it's useful for determining which features and thresholds determine the price categories. Training was performed using default hyperparameters with random_state = 42 and dataset split into an 80/20 training/testing [2], [4].

K-Nearest Neighbors

To optimize the K-Nearest Neighbors (KNN) classifier, I use GridSearchCV with 5-fold cross-validation to determine the optimal number of neighbors k for classification. The dataset is split into 80% training and 20% testing. The GridSearchCV initially evaluates k values from 1 to 20, using the weighted F1 score as the evaluation metrics. Then, it turns out that the best k results in k = 18 with the highest F1 score of 0.642, then I extend the search range of k from 21 to 50 for a more thorough exploration. Then, I got the best k of 42 with a F1 score of 0.649, making it the final chosen mode [3].

Experimental Results

Evaluation Metrics

The classification performance was evaluated using the following metrics:

- **Accuracy**: Used to determine the level of correctness in the model being used.
- **Precision**: A performance metric used to point to the correctness of positive predictions.
- **Recall**: Measures how well positive cases that are really out there are caught by the model
- **F1 Score**: Compares the number of true positives with the number of true negatives while giving more of a general measure of a models' performance [1]-[4].

Model Performance

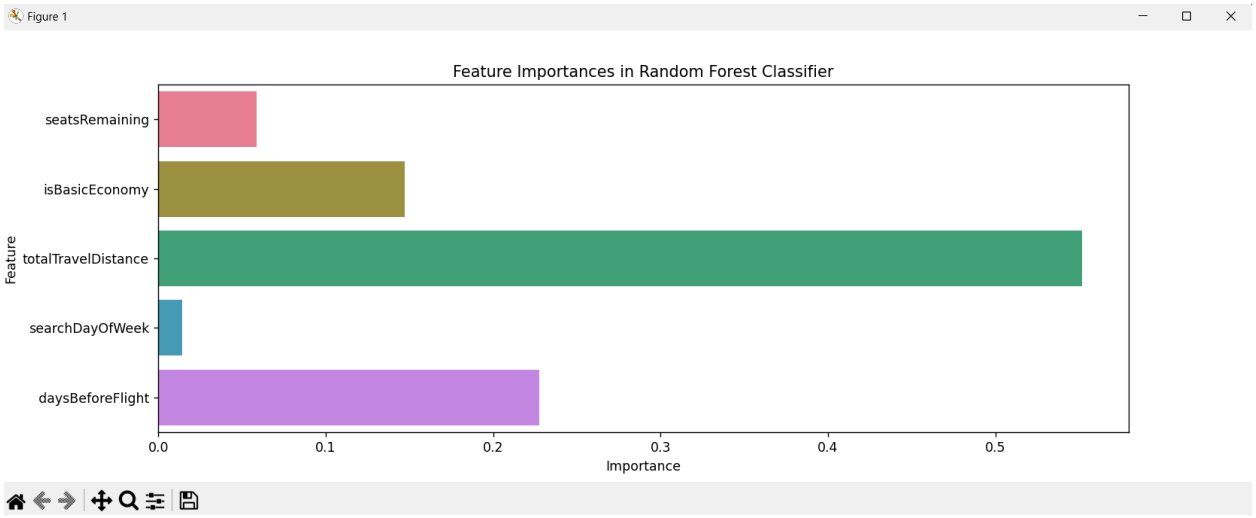
Random Forest

The Random Forest classifier achieved the following performance metrics on the test data:

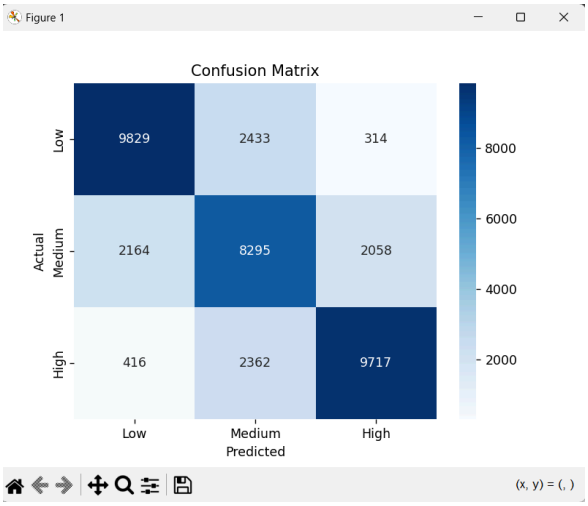
- **Accuracy**:74.07%
- **Precision**: 74.32%
- **Recall**: 74.07%
- **F1 Score**: 74.18%

These results indicate that the model performs moderately well in classifying flight prices into distinct ranges.

The feature importance plot reveals that “isBasicEconomy” and “totalTravelDistance” are the most influential factors, as expected. “DaysBeforeFlight” also contributes meaningfully to the model's decisions.



Feature importance graph for Random Forest Classifier



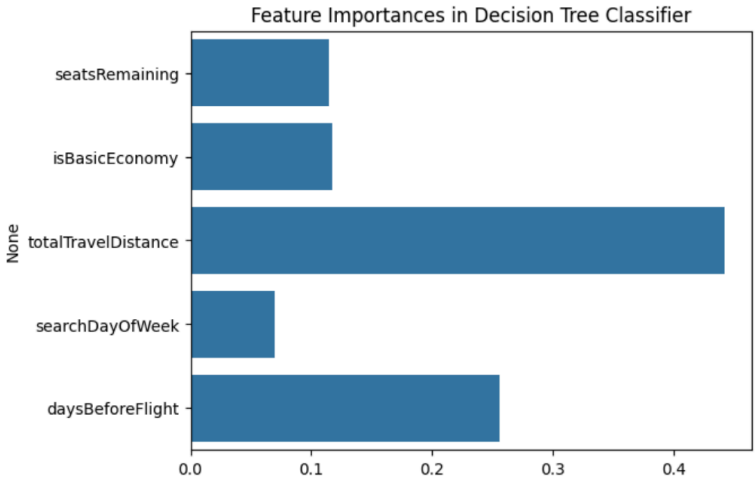
Confusion Matrix for Random Forest Classifier

The confusion matrix shows that the Random Forest classifier performs well in distinguishing "Low" prices, “Medium” prices and “High” prices. It has the highest correct predictions for "High" (9,829 instances), but significant misclassifications occur between "Medium", "High" and “Medium”, “Low” due to overlapping features. The "Low" and “High” categories have relatively fewer misclassifications, highlighting their distinctiveness [1].

Decision Tree

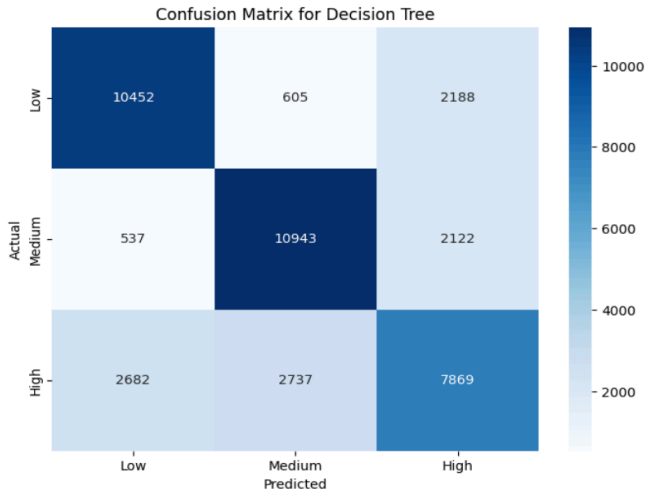
Decision tree classifier showed better results classifying flight prices with:

- **Accuracy** (72.99%)
- **Precision** (72.71%)
- **Recall** (72.99%)
- **F1 Score** (72.81%).



Feature importance for Decision Tree Classifier

The graph above shows that total travel distance and number of days between search and flight date are the features that are the most useful at predicting the target variable.



Confusion Matrix for Decision Tree Classifier

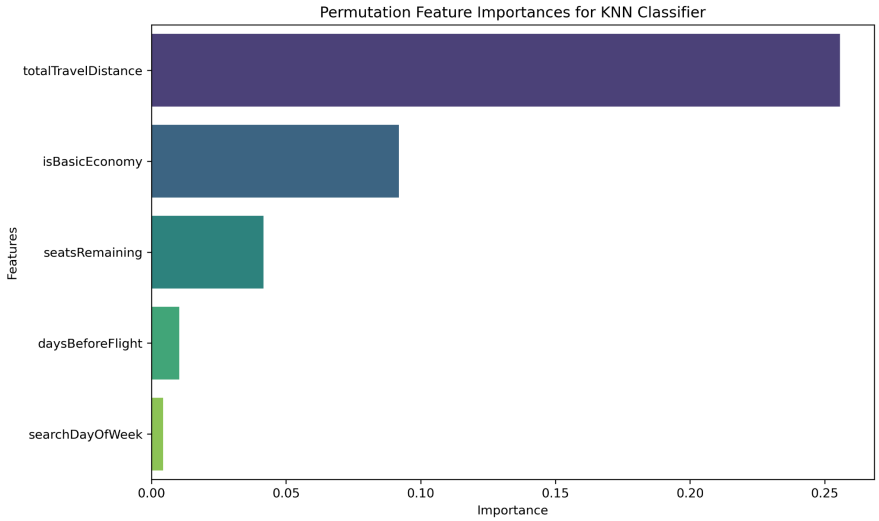
The confusion matrix shows that Decision Tree Classifier categorizes flight prices into Low and Medium well, but has some false positives for High category. Overall, this model classifies prices into three categories better than the other two models [2], [4].

K-Nearest Neighbors

Here's the performance metrics for KNN:

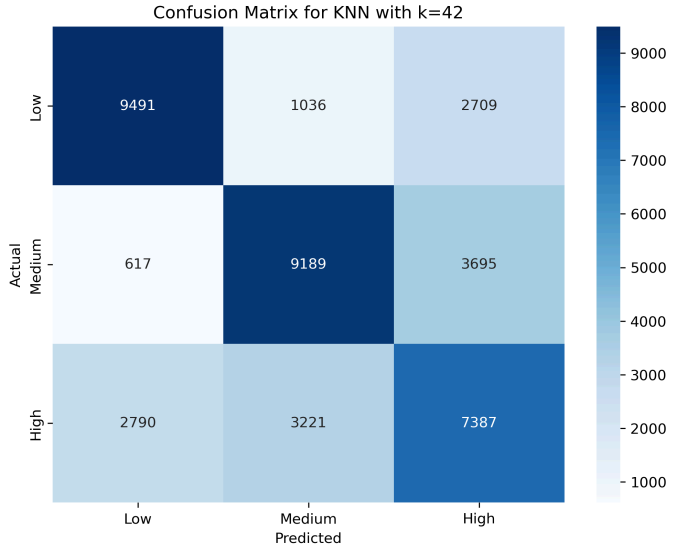
- **Accuracy**: 64.95%
- **Precision**: 65.14%
- **Recall**: 64.95%
- **F1 Score**: 65.03%

Since KNN doesn't inherently have feature importance attributes in its classifier library, I use permutation importance to get its feature importance for this classifier. Permutation importance evaluates the importance of each feature in a dataset by measuring the impact of shuffling its values on the model's performance. In this feature importance graph below, we could see that "totalTravelDistance" is its most important feature for flight price, then "isBasicEconomy", "seatReamining" also determines the flight price. Then, the "daysBeforeFlight" and "searchDayOfWeek" also have some contributing factors [3].



Feature Importance graph for KNN

The confusion matrix shows that the KNN model with k = 42 performs reasonably well for the “low” and “medium” prices but struggles with the “high” classes.



Confusion Matrix Graph for KNN, k = 42

Discussion

Key Insights and Decisions

Feature Engineering: The addition of the weekend indicator was especially helpful in exploring fluctuations in the demand patterns. This product attribute suggested that prices differ on different days of the week and especially on the weekend and this feature helped the model to

neutralize this. Similarly, the seatsRemaining and totalTravelDistance features were found to be related to the supply-demand dynamics and travel distance's effect on price, respectively.

Model Choices:

- **Random Forest:** Reduces overfitting, handles mixed data, and highlights key features like totalTravelDistance [1].
- **Decision Tree:** Highly interpretable and ranks feature importance effectively [2].
- **KNN:** Efficiently captures non-linear relationships with minimal training time and flexible tuning [3].

Challenges Faced:

- **Random Forest**: According to the problems of data imbalance, feature similarity, and computational cost, which were solved through class balancing, feature engineering, or hyperparameter tuning [1].
- **Decision Tree**: Suffers from overfitting, but the overfitting issue is addressed by applying the Grid search method of setting hyperparameters with 5- fold cross validation [2].
- **KNN**: Changed from SVM as it had a long execution time; to find optimal 'k', used GridSearchCV; best F1 score for k=42 [3].

Observations on Model Performance (Things that worked/didn't work)

Random Forest: Obtained 74% accuracy with precision and recall at 74% each: annotated 4 features, including totalTravelDistance and isBasicEconomy. The main types of errors are between "Medium" and "High", or "Medium" and "High" mainly due to the similarities in the features and due to class imbalance. Potential for better performance exists due to hyperparameter tuning and better feature selection and extraction [1].

Decision Tree: After performing GridSearch with 5-fold cross-validation, the best score that this classifier achieved was 73%. The model performance is effective, but not optimal since it has difficulty differentiating "High" prices. After comparing the training accuracy(78.12%) and validation accuracy(72.99%) it's clear that the Decision Tree classifier is overfitting. This could be due to class imbalance or features selected [2].

KNN: Price categories like "Medium" and "High" likely have overlapping feature distributions, making it difficult for KNN to draw clear decision boundaries. KNN relies on distances, so improperly scaled features like totalTravelDistance or seatsRemaining could disproportionately influence the results. If one price category (e.g., "Low") dominates, KNN may favor the majority class, leading to biased predictions [3].

Conclusion

All the models chosen for flight price prediction employed the same characteristics when assigning prices into three different groups. However, the best outcomes were obtained with the help of the Random Forest classification model with the accuracy of 0.74 and such measures as

precision, recall, and F1 equal to 0.74. Random Forest performed better than others because it uses multiple decision trees to minimize overfitting, which makes it more generalized. This makes the method especially suitable for finding relationships in the data while also being resistant to noise and variations. While Decision Tree overfits hierarchical patterns and kNN fails with high dimensionality or noise, Random Forest was the most accurate, robust, and stable for this issue. This makes it easier to aggregate multiple predictions and produce the most reliable and consistent results [3], [4], [5].

Project Plan and Task Distribution

We all worked on: pre-processing, feature extraction, data visualization, measurement of performance, report writing, presentation. **Vivek Ponnala** worked on creating the Github, implementing random forest, writing the report and preparing the slide presentation. **Maggie Qin** worked on Implementing KNN, report and slide presentation. **Hoai An Nguyen** worked on implementing decision trees, the report and slide. Peer discussions, code collaboration, technical problem solving were all integral activities where the group members actively contributed and achieved a fairly equal distribution of workload.

GitHub Repository Link: https://github.com/vipvivek15/CMPE_255_Final_Project

References

[1] "Random Forest Classifier - Scikit-learn Documentation," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: Dec. 2, 2024].

[2] "Decision Tree Classifier - Scikit-learn Documentation," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed: Dec. 2, 2024].

[3] "K-Nearest Neighbors Algorithm - Scikit-learn Documentation," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. [Accessed: Dec. 2, 2024].

[4] "Decision Trees Explained," Towards Data Science, [Online]. Available: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052. [Accessed: Dec. 2, 2024].

[5] W. McKinney, *Python for Data Analysis: Data Wrangling with pandas, NumPy, and IPython*. O'Reilly Media, 2017.

[6] J. VanderPlas, *Python Data Science Handbook*. O'Reilly Media, 2016.

[7] D. Wong, "Flight Prices Dataset," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/dilwong/flightprices. [Accessed: Dec. 2, 2024].