CMPE 255 – Fall 2024

Project Report: Flight Prices Classification

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

1. 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.

Objective

Our objective is to build a robust classification model that categorizes flight prices into three ranges: low, medium, and high. This categorization allows us to simplify the complex pricing dynamics and create a model capable of informing price-sensitive travel decisions and optimizing resource allocation for the airline industry.

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. Inspired by these studies, we selected the Random Forest, Decision Tree, and K-Nearest Neighbors.

2. System Design & Implementation Details

Algorithm Selection

Random Forest was chosen for its ability to handle complex datasets with mixed data types and provide feature importance insights. Decision Tree offers interpretability and computational efficiency for ranking feature importance, while KNN effectively captures non-linear relationships, serving as a reliable baseline.

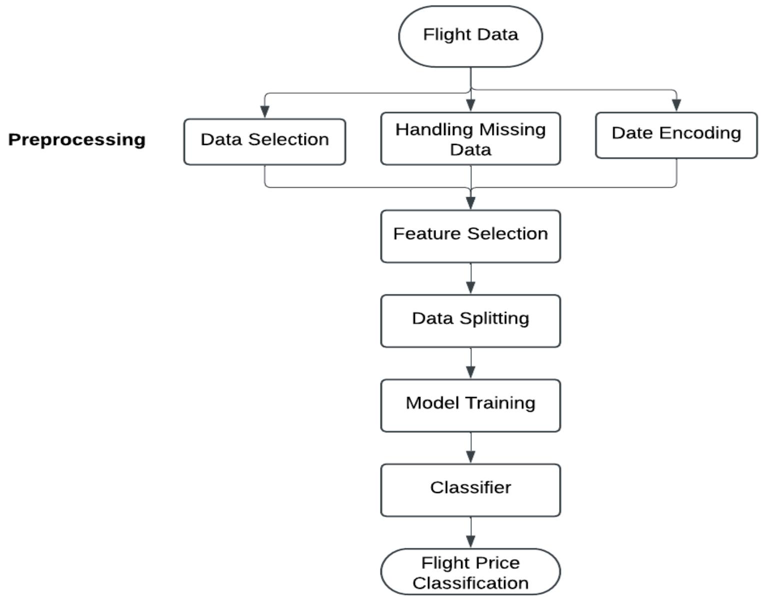
Technologies

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

Data Pipeline

Data was cleaned, engineered with new variables (e.g., weekend indicator), and split into 80-20 training and testing sets. Models were trained, evaluated, and visualized to analyze performance and feature importance.

Architecture Diagram



3. 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.

Kaggle Dataset: https://www.kaggle.com/datasets/dilwong/flightprices

Subset Extracted: https://drive.google.com/file/d/1fn4NaQ3xzocOg0mqfDkSqpNCcCMthjXW/view?usp=sharing

Preprocessing

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

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.

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
- **daysBeforeFligh**t: 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 converted a continuous price variable into a classification target.

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.

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

We selected Random Forest for its ability to handle mixed data types and its suitability for classification tasks. The model was configured with 100 trees and a maximum depth chosen based on cross-validation to balance accuracy and training time. Data was split 80-20 for training and testing, ensuring a sufficient sample for model evaluation.

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.

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 model.

Experimental Results

Evaluation Metrics

The classification performance was evaluated using the following metrics:

- **Accuracy**: Measures the overall correctness of the model.
- **Precision**: Indicates the accuracy of positive predictions.
- **Recall**: Reflects the model’s ability to capture actual positive cases.
- **F1 Score**: Balances precision and recall, providing a holistic view of model performance.

Model Performance

Random Forest

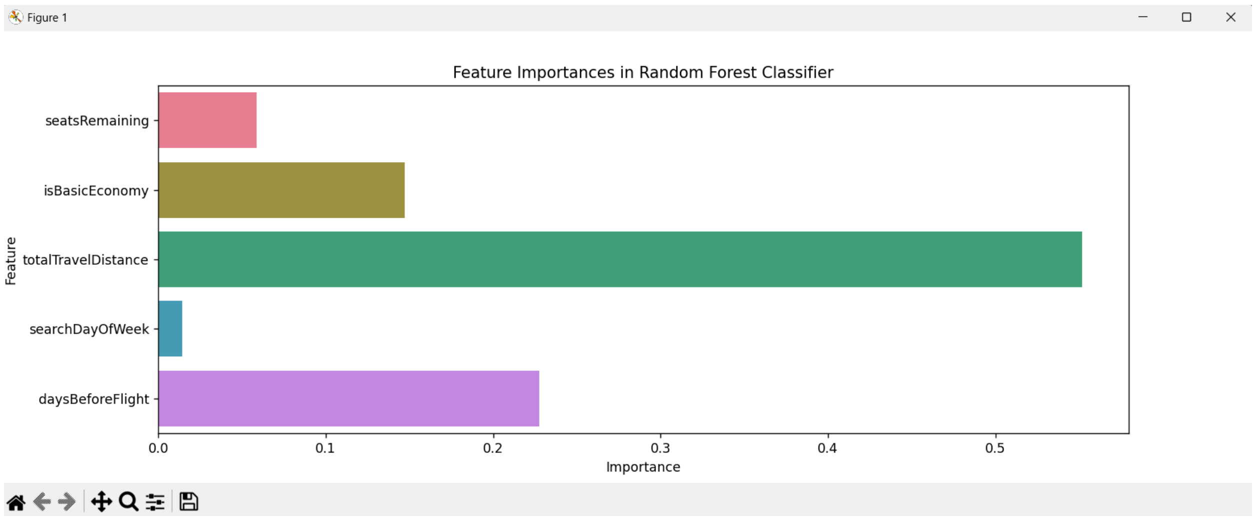
The Random Forest classifier achieved the following performance metrics on the test data:

- **Accuracy**:74%
- **Precision**: 74%
- **Recall**: 74%

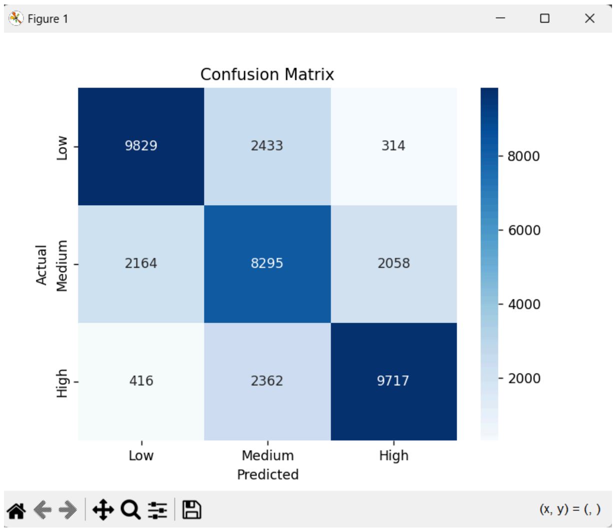
- **F1 Score**: 74%

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

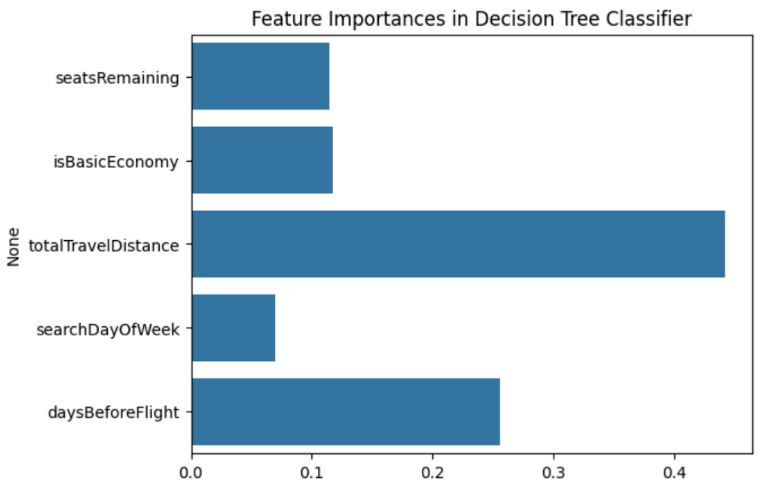
The confusion matrix shows that the Random Forest classifier performs well in distinguishing "Low" prices, “Medium” prices and “High” prices. It has the highest correct predictions for "High" (9,829 instances), but significant misclassifications occur between "Medium", "High" and

“Medium”, “Low” due to overlapping features. The "Low" and “High” categories have relatively fewer misclassifications, highlighting their distinctiveness.

Decision Tree

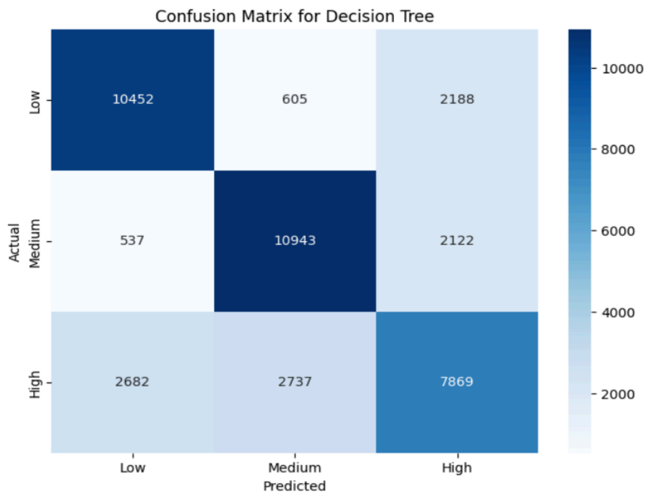
Decision tree classifier showed better results classifying flight prices with:

- **Accuracy** (73%)
- **Precision** (73%)
- **Recall** (73%)
- **F1 Score** (73%).



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

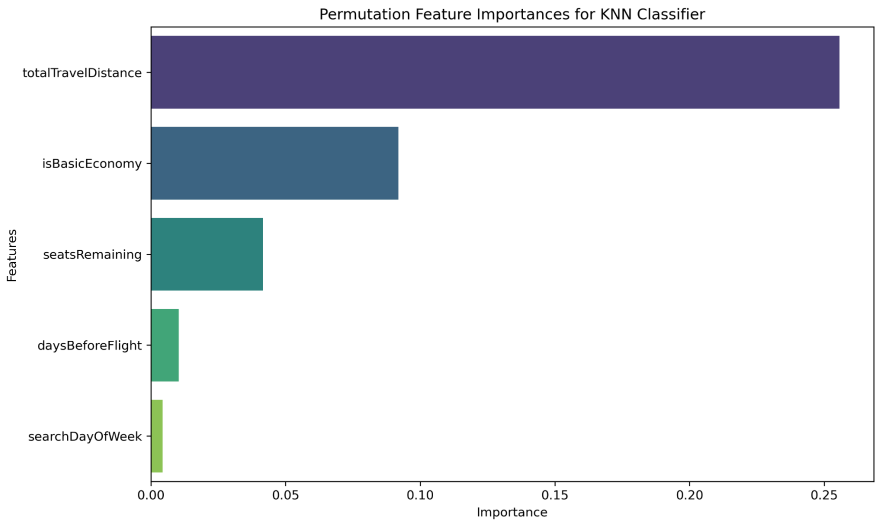
The confusion matrix shows that Decision Tree Classifier categorizes flight prices into Low and Medium well, but has some false positives for High category. Overall, this model classifies prices into three categories better than the other two models.

K-Nearest Neighbors

Here's the performance metrics for KNN:

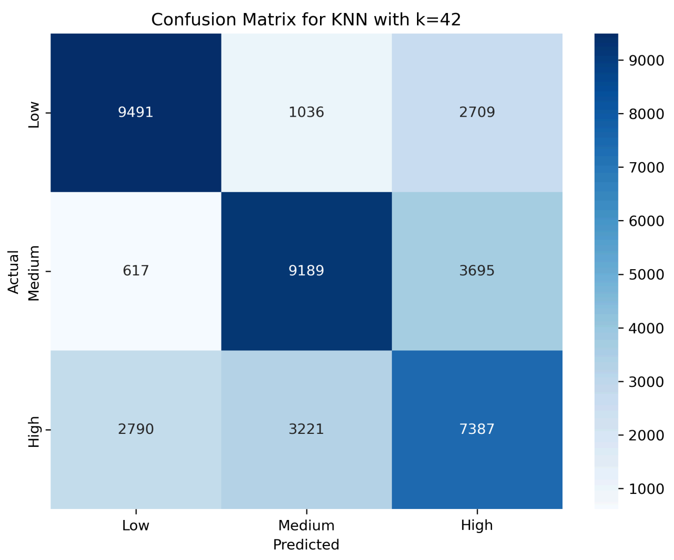
- **Accuracy**: 65%
- **Precision**: 65%
- **Recall**: 65%
- **F1 Score**: 65%

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.



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

4. Discussion

Key Insights and Decisions

Feature Engineering:
Including the **weekend indicator** was particularly beneficial for capturing variations in demand trends. Prices often fluctuate significantly during weekends, and this feature allowed the model to better account for these patterns. Similarly, the **seatsRemaining** and **totalTravelDistance** features provided key information related to supply-demand dynamics and the impact of travel distance on pricing.

Model Choices:

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

Challenges Faced:

- **Random Forest:** Faced data imbalance, overlapping features, and computational complexity, addressed through class balancing, feature engineering, and hyperparameter tuning.
- **Decision Tree:** Prone to overfitting, resolved by optimizing hyperparameters using grid search with 5-fold cross-validation.

- **KNN:** Switched from SVM due to long runtime; finding the optimal k required extensive GridSearchCV, with k=42 yielding the best F1 score.

Observations on Model Performance

Random Forest: Achieved 74% accuracy with precision and recall at 74% and 74%, respectively, highlighting `totalTravelDistance` and `isBasicEconomy` as key features. Misclassifications occur mainly between "Medium" and "High", “Medium” and “High” categories due to overlapping features and class imbalances. Optimizing hyperparameters and enhancing feature engineering could improve performance.

Decision Tree:

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.

Conclusion

All models selected for flight price prediction used the same features to categorize prices into three distinct categories. However, the Random Forest classification model achieved the best results, with an accuracy of 0.74 and precision, recall, and F1 scores also at 0.74. Random Forest excelled due to its ensemble approach, which combines multiple decision trees to reduce overfitting and improve generalization. This made it particularly effective in capturing patterns in the dataset while maintaining robustness against noise and variability. Compared to Decision Trees, which may be prone to overfitting on specific hierarchical patterns, and kNN, which can struggle with high-dimensional data or noise, Random Forest provided the optimal balance of accuracy, robustness, and predictive power for this problem. Its ability to aggregate diverse predictions led to the most reliable and consistent performance.

5. Project Plan and Task Distribution

We all worked on: data cleaning, feature engineering, data visualization, evaluation metrics, report documentation, presentation preparation

- **Vivek Ponnala**: Creating the github, random forest, report and slide presentation
- **Maggie Qin**: KNN, report and slide presentation
- **Hoai An Nguyen**: decision tree, report and slide presentation

Each member participated in group discussions, code reviews, and collaborative troubleshooting, ensuring a balanced workload distribution.

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

6. Bibliography

KNN

- K-Nearest Neighbors Algorithm - Scikit-learn Documentation
  https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- Understanding K-Nearest Neighbors (KNN) Algorithm – Towards Data Science
  https://towardsdatascience.com/understanding-k-nearest-neighbors-knn-682eeb84c912

Random Forest

- Random Forest Classifier - Scikit-learn Documentation
  https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- How Random Forest Works - Analytics Vidhya
  https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
- A Beginner’s Guide to Random Forest – GeeksforGeeks
  https://www.geeksforgeeks.org/random-forest-in-machine-learning/

Decision Tree

- Decision Tree Classifier - Scikit-learn Documentation
  https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- Decision Trees Explained - Towards Data Science
  https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052