Assignment 3

ML Data Product

5 November 2023

Group 19

Student Last Name	Student First Name	Student ID	Group Allocation
Makha	Panalee	14367914	Student A
Wijara	Wongwara	14191732	Student B
Tiamklang	Thirada	14337188	Student C
Phuawiriyakul	Thanchanok	24582239	Student D

Github	Project Repo: https://github.com/wongwara/flight-streamlit-at3

36120 - Advanced Machine Learning Application
Master of Data Science and Innovation
University of Technology of Sydney

Table of Contents

1. Executive Summary

Overview

The objective of this project is to create models for the development of a data product aimed at assisting users in the USA in improving their accuracy when estimating local travel airfare. Users will have the capability to input their trip specifics, and the application will provide predictions for their expected flight fares. The MSE and MAE measures will be used to evaluate model performance.

Problem Statement

The project aims to solve the challenge of accurately estimating local airfare for USA travellers in the context of the travel industry. Travellers struggle with predicting flight costs due to factors like dynamic pricing, fare classes, and seasonal variations, making it challenging to plan and budget trips effectively.

Achieved Outcomes and Results

All the models achieved impressive results, consistently maintaining Mean Squared Error (MSE) scores below 0.6. Hence, the K-Nearest Neighbors (KNN) model excelled in predicting 'total fare' and emerged as our preferred choice.

To make these models accessible and valuable to a broader audience, we deployed the best model from each team member on a Streamlit platform. Our Streamlit website (https://crybaby-fareprediction.streamlit.app/) features two distinct pages:

(a) Explore Page: This page provides insightful graphs illustrating the 'total fare' trends over each month. Users can quickly grasp how fares vary throughout the year.

(b) Prediction Page: On this page, users can interact with a form where they input specific flight details, namely departure date and time, starting point, departure airport, and cabin class. The form generates fare predictions using all four models, and users can compare the predictions from each model.

With this Streamlit platform, we aim to offer a valuable and user-friendly tool for fare prediction and exploration, enhancing the accessibility and utility of our project for a broader audience.

2. Business Understanding

a. Business Use Cases

The development of a data product aimed at assisting users in the United States with estimating airfare for local travel is driven by various business use cases and scenarios.

- **Personalized Travel Planning:** Users input travel details, and machine learning predicts flight fares, aiding budget planning.
- **Budget Optimization:** Users find cost-effective travel options, aligning plans with budgets.
- **Comparing Travel Options:** Users compare fare estimates for informed decision-making.
- **Historical Fare Analysis:** Users analyse airfare trends for optimal booking times.

Challenges and Opportunities

Challenge	Description
Data Complexity	Airfare prediction involves complex data, including airline routes, schedules, cabin classes, and demand fluctuations. Machine learning algorithms can effectively handle this complexity and provide valuable insights.
Large Dataset	The dataset used for training the machine learning model is extensive and combines diverse data types, necessitating significant data preprocessing and feature engineering.
Feature Engineering	To make the data relevant for machine learning, feature engineering is essential. This involves creating and transforming features that are informative for airfare prediction.
Privacy Concerns	Dealing with sensitive travel data requires careful consideration of data privacy and protection to ensure users' information is handled securely and ethically.

Table 1: Challenges faced

Opportunity	Description
User Engagement	Offering fare estimation and analysis tools can boost user engagement and retention for travel-related platforms, potentially generating revenue through advertising, affiliate marketing, or direct booking options.
User Interface Design	Creating an intuitive and user-friendly interface that effectively communicates airfare predictions to users is crucial for the project's success.
Competitive Advantage	Providing users with accurate airfare estimates and tools for budget optimization can give the business a competitive advantage in the travel industry, as users increasingly rely on data-driven decisions.

Table 2: Opportunity faced

b. Key Objectives

The key objective of this endeavour is to accurate Airfare Estimation by developing a data product that can accurately estimate local travel airfare for users in the USA, providing reliable fare predictions.

Stakeholders and Requirement

Stakeholder	Requirement
Users	Users seek accurate, personalised airfare estimates that help them plan and optimise their travel while staying within their budget.
Travel Platforms/ Agencies	Travel platforms offer travel planning tools to attract and retain users, increasing engagement and revenue.
Airlines	Airlines may benefit from more informed travellers who can make cost-effective booking decisions.

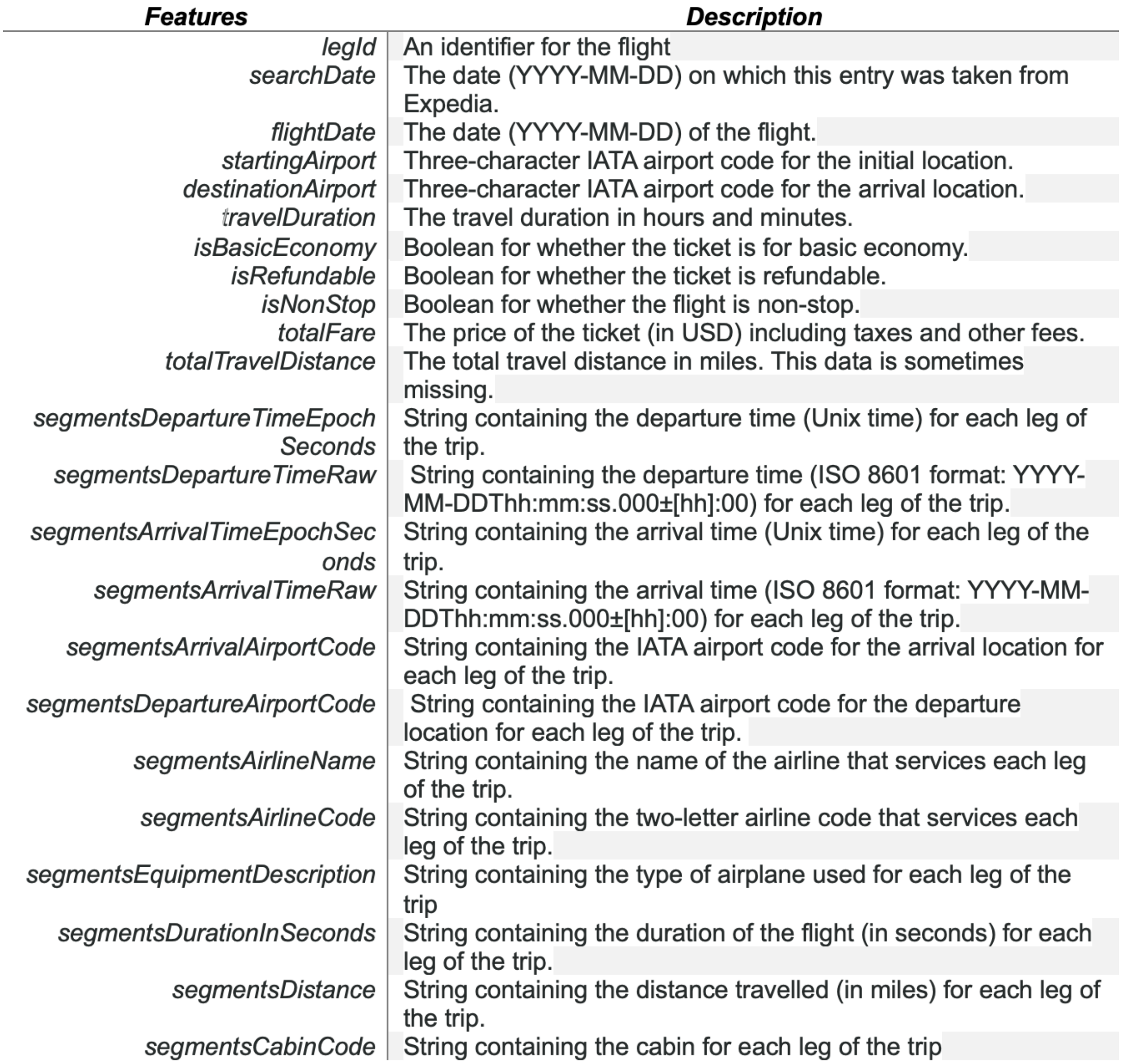
Table 3: Project's stakeholder and requirements

The project achieves the above requirements implementing machine learning algorithms. Provide personalised airfare estimates based on travel details provided by users, facilitate last-minute bookings with real-time fare information, improve travel platforms, and indirectly support airlines by assisting travellers make well-informed booking decisions.

3. Data Understanding

The dataset was scraped from Expedia between 2022-04-17 and 2022-07-17. It is stored in the "itineraries_csv.zip" file, which contains 16 folders, each corresponding to an airport in the USA. The merged dataset comprises 23 features, including 3 bool, 2 float, 1 integer and 18 object, and includes a total of 13,519,999 rows.

Data Description



Features	***Description***
legId	An identifier for the flight
searchDate	The date (YYYY-MM-DD) on which this entry was taken from Expedia.
flightDate	The date (YYYY-MM-DD) of the flight.
startingAirport	Three-character IATA airport code for the initial location.
destinationAirport	Three-character IATA airport code for the arrival location.
travelDuration	The travel duration in hours and minutes.
isBasicEconomy	Boolean for whether the ticket is for basic economy.
isRefundable	Boolean for whether the ticket is refundable.
isNonStop	Boolean for whether the flight is non-stop.
totalFare	The price of the ticket (in USD) including taxes and other fees.
totalTravelDistance	The total travel distance in miles. This data is sometimes missing.
segmentsDepartureTimeEpochSeconds	String containing the departure time (Unix time) for each leg of the trip.
segmentsDepartureTimeRaw	String containing the departure time (ISO 8601 format: YYYY-MM-DDThh:mm:ss.000±[hh]:00) for each leg of the trip.
segmentsArrivalTimeEpochSeconds	String containing the arrival time (Unix time) for each leg of the trip.
segmentsArrivalTimeRaw	String containing the arrival time (ISO 8601 format: YYYY-MM-DDThh:mm:ss.000±[hh]:00) for each leg of the trip.
segmentsArrivalAirportCode	String containing the IATA airport code for the arrival location for each leg of the trip.
segmentsDepartureAirportCode	String containing the IATA airport code for the departure location for each leg of the trip.
segmentsAirlineName	String containing the name of the airline that services each leg of the trip.
segmentsAirlineCode	String containing the two-letter airline code that services each leg of the trip.
segmentsEquipmentDescription	String containing the type of airplane used for each leg of the trip
segmentsDurationInSeconds	String containing the duration of the flight (in seconds) for each leg of the trip.
segmentsDistance	String containing the distance travelled (in miles) for each leg of the trip.
segmentsCabinCode	String containing the cabin for each leg of the trip

Figure 1: The table of data description

Exploratory Data Analysis

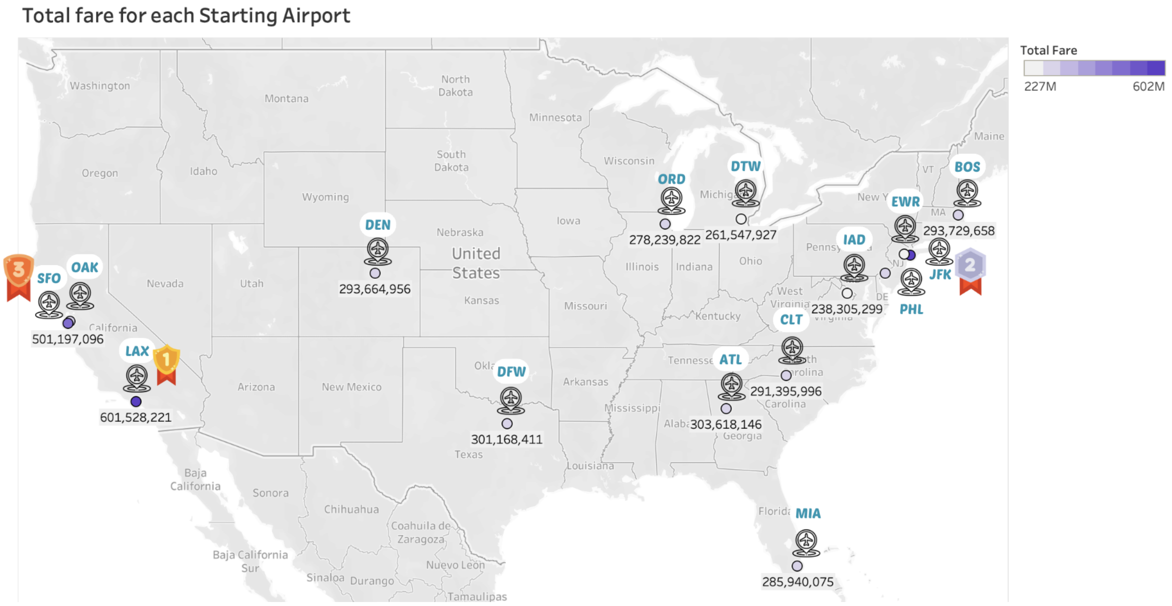


Figure 2: The map of total fare for each starting airport in the US

The map of the USA above represents the Starting Airport with their total fare. It can be seen that LAX, or Los Angeles International Airport, is the most popular airport from which passengers usually depart, with approximately 600 million dollars in total fares, followed by JFK and SFO airports, which have around 500 million dollars each.

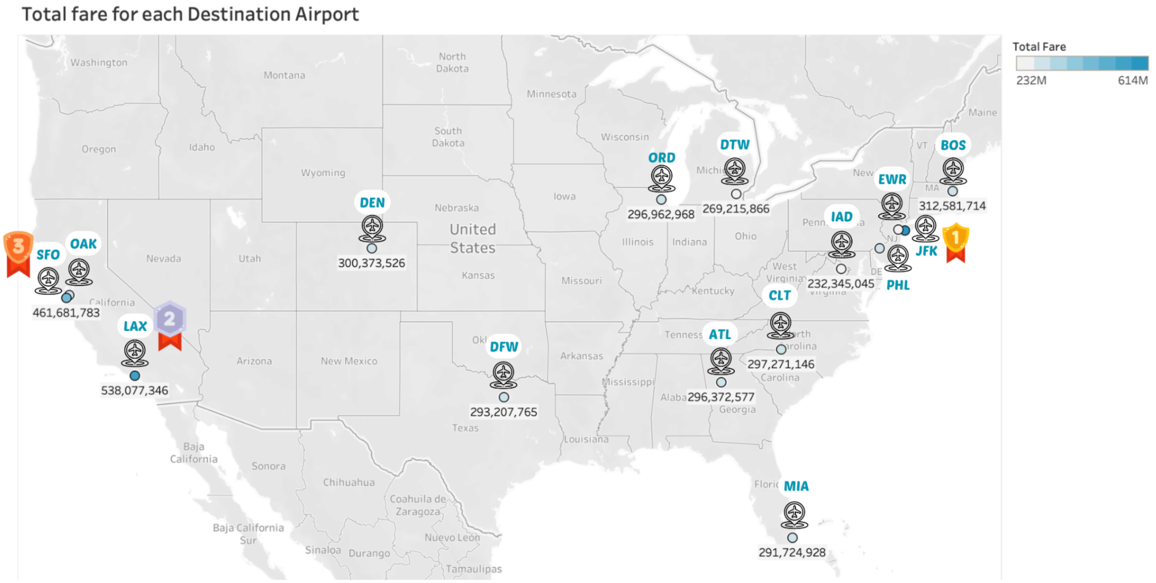


Figure 3: The map of total fare for each destination airport in the US

The map of the USA above represents the Destination Airport with their total fare. It can be seen that JFK or John F. Kennedy Airport is the most popular airport that passengers usually travel to, with approximately 614 million dollars in total fares, followed by LAX and SFO airports, which have around 538 million and 461 million dollars, respectively.

Correlation

Correlation analysis provides a valuable means to assess the relationships between features and the target variable. By measuring the correlation coefficients between individual features and the target variable, we can identify the attributes that show a high correlation, indicating their potential significance in predicting the target outcome. However, it is essential to avoid the issue of multicollinearity, where selected features are highly correlated with each other, because of model instability and unreliable coefficient estimates.

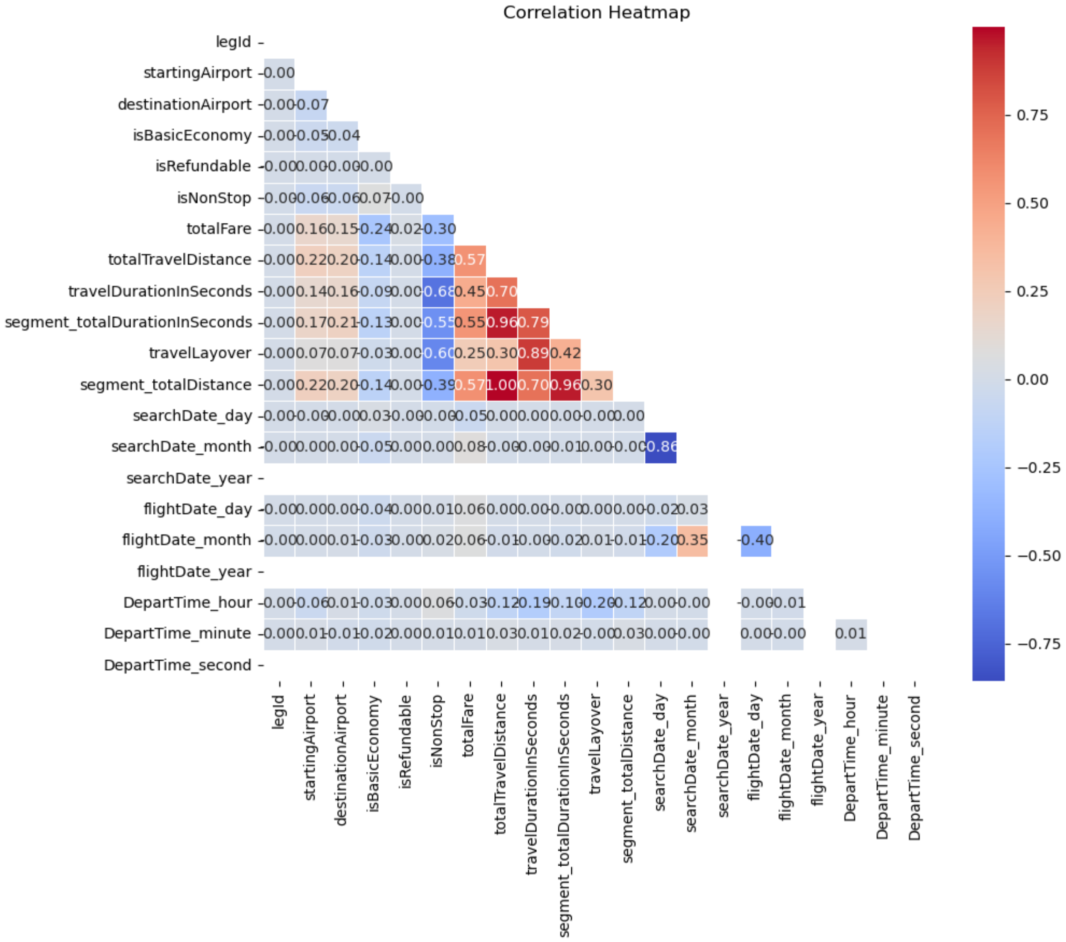
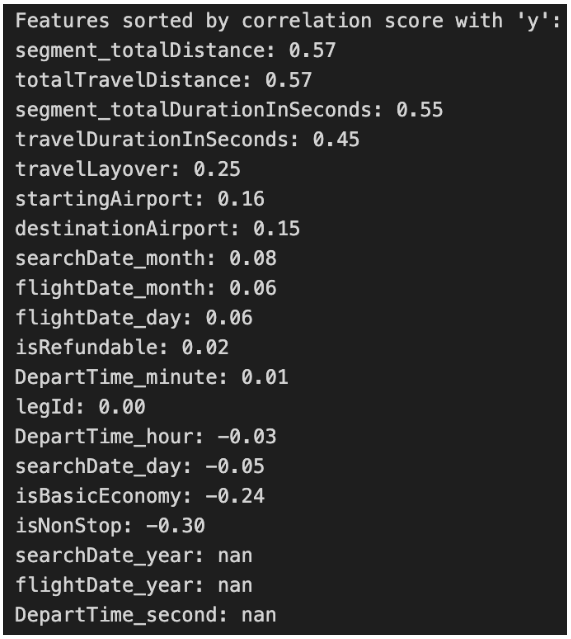


Figure 4: The correlation heatmap of all the features

4. Data Preparation

- **Data preprocessing**
  - **Merging Datasets:** Merge datasets from various airports into a single dataframe for exhaustive analysis.
  - **Converting Travel Duration:** Convert travel duration from a string format into seconds for consistency and easy analysis.
  - **Segment Splitting:** The segment columns are split into lists.
  - **Total Segment Duration:** Calculate the total duration of segments and create a new column named "totalDuration (segment)."
  - **Travel Layover:** A new column, "travelLayover," represents the difference between travel duration and the total segment duration.
  - **Transit Airport Code:** Extract and store the airport code of transit locations in a new column named "transitAirportcode."
  - **All Airport Codes:** We collect all airport codes, including departure, arrival, and transit, and store them in a new column labelled "All airport."
  - **Date Transformation:** The date column serachDate and flightDate is transformed into separate columns for day, month, and year.
  - **Departure Time Extraction:** Extract the departure time in Unix epoch seconds format and separate it into individual columns for hours, minutes, and seconds.
  - **Handling Null Values:** To address missing data, fill null values with the mean of the respective columns.
  - **Feature selection:**



```
Features sorted by correlation score with 'y':
segment_totalDistance: 0.57
totalTravelDistance: 0.57
segment_totalDurationInSeconds: 0.55
travelDurationInSeconds: 0.45
travelLayover: 0.25
startingAirport: 0.16
destinationAirport: 0.15
searchDate_month: 0.08
flightDate_month: 0.06
flightDate_day: 0.06
isRefundable: 0.02
DepartTime_minute: 0.01
legId: 0.00
DepartTime_hour: -0.03
searchDate_day: -0.05
isBasicEconomy: -0.24
isNonStop: -0.30
searchDate_year: nan
flightDate_year: nan
DepartTime_second: nan
```

Figure 5: features sorted by correlation score with the target (y)

We carried out feature selection by identifying variables that exhibited a correlation with the target variable (total fare) above |0.24|. In doing so, we maintained a critical consideration for multicollinearity levels between features, ensuring they remained below 0.7. Figure 4's heatmap visually represents these relationships.

The selected features, characterised by both high correlation with the target variable and low collinearity, encompass 'totalTravelDistance,' 'isNonStop,' 'isBasicEconomy,' 'startingAirport,' 'destinationAirport,' 'segmentsCabinCode,' 'flightDate_day,' 'flightDate_month,' 'flightDate_year,' 'DepartTime_hour,' 'DepartTime_minute,' 'DepartTime_second,' and 'totalFare.'

- **Feature Engineering**
  - **Factorising Segment Columns:** The segment columns were factorised to enhance their usability in modelling.
  - **Label Encoding 'SegmentCabinCode':** Applied label encoding to the 'segmentCabinCode' column, converting categorical data into a numerical format for analysis.
  - **Encoding Categorical Columns:** Other categorical columns were also encoded using label encoding, ensuring consistency in handling categorical data.

        - **Scaling Numeric Columns:** Numeric columns were scaled using Standard Scaler, making them compatible with the modelling process.

    - **Data splitting:**

The dataset was divided into training, testing, and validation sets with an 80:20 ratio. It is important to note that all the experiments used the same dataset after this split, allowing for a fair comparison of the best model later.

5. Modelling

a. Regression - Student A

- **Algorithm:** Linear Regression, Ridge, ElasticNetCV, K-Nearest Neighbors, and Gradient boosting

Model	***Rational***
Linear Regression	Linear regression is a fundamental algorithm for regression tasks and serves as a baseline model. It's selected for its simplicity and interpretability, making it a suitable starting point for airfare estimation.
Ridge Regression	Ridge regression is chosen to handle potential multicollinearity in the dataset. It helps prevent overfitting and is well-suited for scenarios where multiple features may be correlated.
ElasticNetCV	ElasticNetCV is a hybrid of L1 (Lasso) and L2 (Ridge) regularisation. It's selected to combine the feature selection capabilities of Lasso with the robustness of Ridge, providing a balanced approach to feature selection and regularisation.
K-Nearest Neighbors (KNN)	KNN is chosen for its ability to capture local patterns in data. It's useful when there might be geographical or airport-specific factors affecting airfares, and it can adapt to varying local conditions.
Gradient Boosting	Gradient boosting is a powerful ensemble method that can capture complex nonlinear relationships in the data. It's chosen to improve prediction accuracy by learning from the errors of the previous models.

These hyperparameters were chosen based on common practices and considerations for KNN models

- neighbours: [3, 5, 7, 9] - The range of neighbours provides flexibility in capturing local patterns
- weights: ['uniform'] - Treat all neighbours equally
- p: [1] - the Manhattan distance metric

- **Model Evaluation:**

The best parameters are neighbours = 9, weights = 'uniform', p = 1.

b. Regression - Student B

- **Algorithm:** Decision Tree, XGBoost and Adaboost

Model	***Rational***
Decision Tree	Decision trees are chosen for their simplicity and interpretability, which is critical when explaining fare predictions to passengers or stakeholders. They can capture non-linear relationships in tickets price factors and handle a mix of categorical and numerical features such as flight details, cabincode and travel layover. Their resistance to outliers is advantageous because ticket prices can vary.
XGBoost (default and with learning_rate: 0.2, max_depth: 5, n_estimators: 300	XGBoost was chosen for its high-performance capabilities, which can assist in modelling the complexity and noise in ticket price data. In real-world airline pricing datasets where data quality varies, the ability to handle missing values and automatically manage imbalanced data is critical. XGBoost can provide feature importances, which can help understand the main drivers of fare predictions.
Adaboost	AdaBoost improves the performance of the decision tree model. AdaBoost can help decision trees perform better in the context of airline ticket pricing by focusing on their weak points. This ensemble method improves predictive power while preserving interpretability, which is critical for ensuring transparent pricing strategies.

These hyperparameters were chosen based on common practices and considerations for XGBoost regression models

- n_estimators: [100, 200, 300] - The number of boosting rounds or trees in the ensemble.

- max_depth: [3, 4, 5] - The maximum depth of the individual trees.

- learning_rate: [0.1, 0.2, 0.01] - The learning rate, which controls the step size during optimization.

- **Model Evaluation:**

And the hyperparameters are: {'learning_rate': 0.2, 'max_depth': 5, 'n_estimators': 300}

c. TenserFlow - Student C

- **Algorithm:** TensorFlow Decision Forests (TF-DF) with Keras

Model	***Rational***
Random Forest	Random Forest is an ensemble learning method that uses multiple decision trees to make predictions. It's selected for its ability to handle complex relationships in the data and reduce overfitting. Random Forest can be especially effective when there are a large number of features and data points.
Gradient Boosted Trees	Gradient Boosted Trees is another ensemble method that builds an additive model by training decision trees sequentially. It's chosen to capture complex nonlinear relationships in the data and improve prediction accuracy by learning from the errors of previous models.

- **Convert to tensor flow datasets:**

  With batch size=100, using `from_tensor_slices` to convert NumPy arrays into tensors
- **Model fitting:**

  train_dataset, epochs=1, validation_data=validation_dataset
- **Model compile:**

  metrics=["mean_squared_error"]
- **Model Architecture:**

- RandomForestModel(task=tfdf.keras.Task.REGRESSION)
- RandomForestModel(task=tfdf.keras.Task.REGRESSION, num_trees=100, max_depth=10)
- GradientBoostedTreesModel(task=tfdf.keras.Task.REGRESSION)

- **Model Evaluation:**

  Evaluate train and test dataset and output loss as `MSE score`

d. Keras - Student D

- **Algorithm:** Keras and kerasTuner

Model	***Rational***
Keras	Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.
KerasTuner	KerasTuner is a user-friendly and scalable framework designed to simplify the process of hyperparameter optimization in machine learning. It addresses the challenges of searching for the best hyperparameters for models.

- **Model Architecture:**
  - **Keras:** defined keras with three different layers and compile the model
    - The input layer: 12 neurons, Input data shape (12,) and The ReLU activation function
    - The hidden layer: 8 neurons and The ReLU activation function
    - The output layer: 1 neuron
    - Compile the model with MSE score
  - **Keras Tuners:** defined function to create keras model and tuner with randomSearch
    - Create a function with tunable hyperparameters for the number of units in a hidden layer. The model uses mean squared error (MSE) as the loss function to optimise for prediction.
    - A random search using Keras Tuner to search for the best hyperparameters. Tuner with the objective is 'val_loss' or mse in our model tuner to minimise the validation loss and set max_trails with 5, as lower validation loss indicates better model generalisation

6. Evaluation

a. Evaluation Metrics

In the context of building a data product to estimate local travel airfare in the USA, the chosen evaluation metrics are:

Metric	**Description**
Mean Squared Error (MSE)	It measures the average squared difference between predicted and actual fares, helping assess prediction accuracy.
Mean Absolute Error (MAE)	MAE quantifies the average absolute difference between predicted and actual fares, offering insights into prediction accuracy.

Table 4: Evaluation metrics

b. Results and Analysis

- **Linear regression, Ridge, ElasticNetCV, K-Nearest Neighbors, Gradient Boost:**

The results of training five different regression models as in the bar chart below, with notable performance distinctions. The top-performing model is KNN4 or K-Nearest Neighbors with neighbours=9, weights='uniform', p=1, showcasing the lowest MSE on both the training and validation sets, approximately 0.206 and 0.222, respectively. Following closely is the Gradient Boost model, exhibiting slightly superior MSE scores compared to Linear regression, Ridge regression, and the ElasticNetCV model on both training and validation sets.

Notably, the KNN models, particularly KNN4, significantly outperform the linear regression-based models, delivering considerably lower MSE and MAE values. It's noteworthy that as the number of neighbours (K) increases from 3 to 9, the training MSE and MAE values decrease, signifying improved model fit. However, it's observed that the validation MSE and MAE values also decrease, but the improvement becomes less pronounced, hinting at the possibility of higher K values leading to over-smoothing.

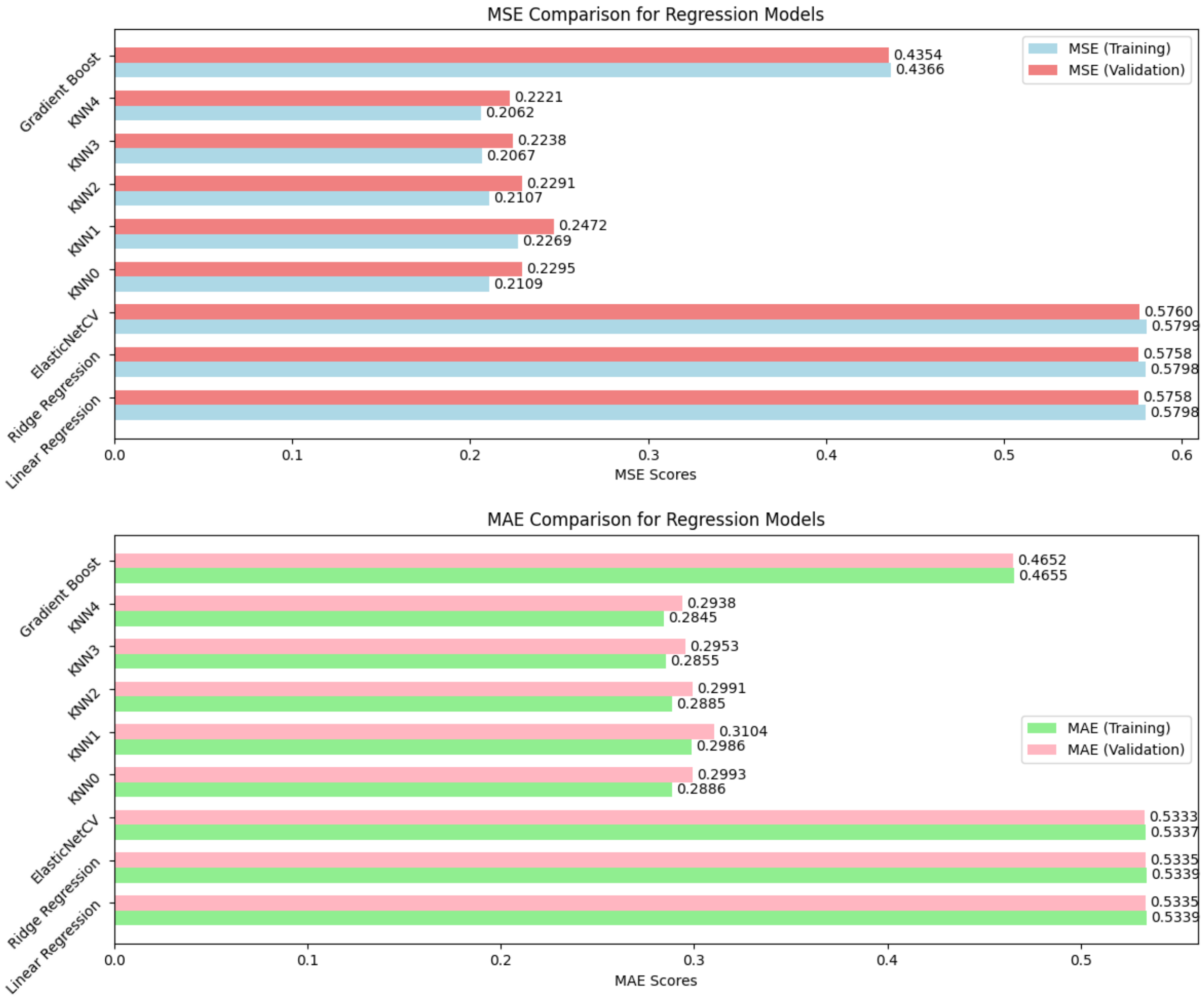


Figure 6: The bar chart of MSE and MAE score for regression models

- **Decision Tree, XGBoost Regressor and Adaboost regressor:**

The MSE and MAE scores reveal significant differences in model performance. The optimally tuned XGBoost model outperforms the default XGBoost by 0.2997 versus 0.3015 on the training set. This demonstrates the tuned XGBoost's improved predictive capability, outperforming the Decision Tree (MSE: 0.5146) and AdaBoost (MSE: 0.6438).

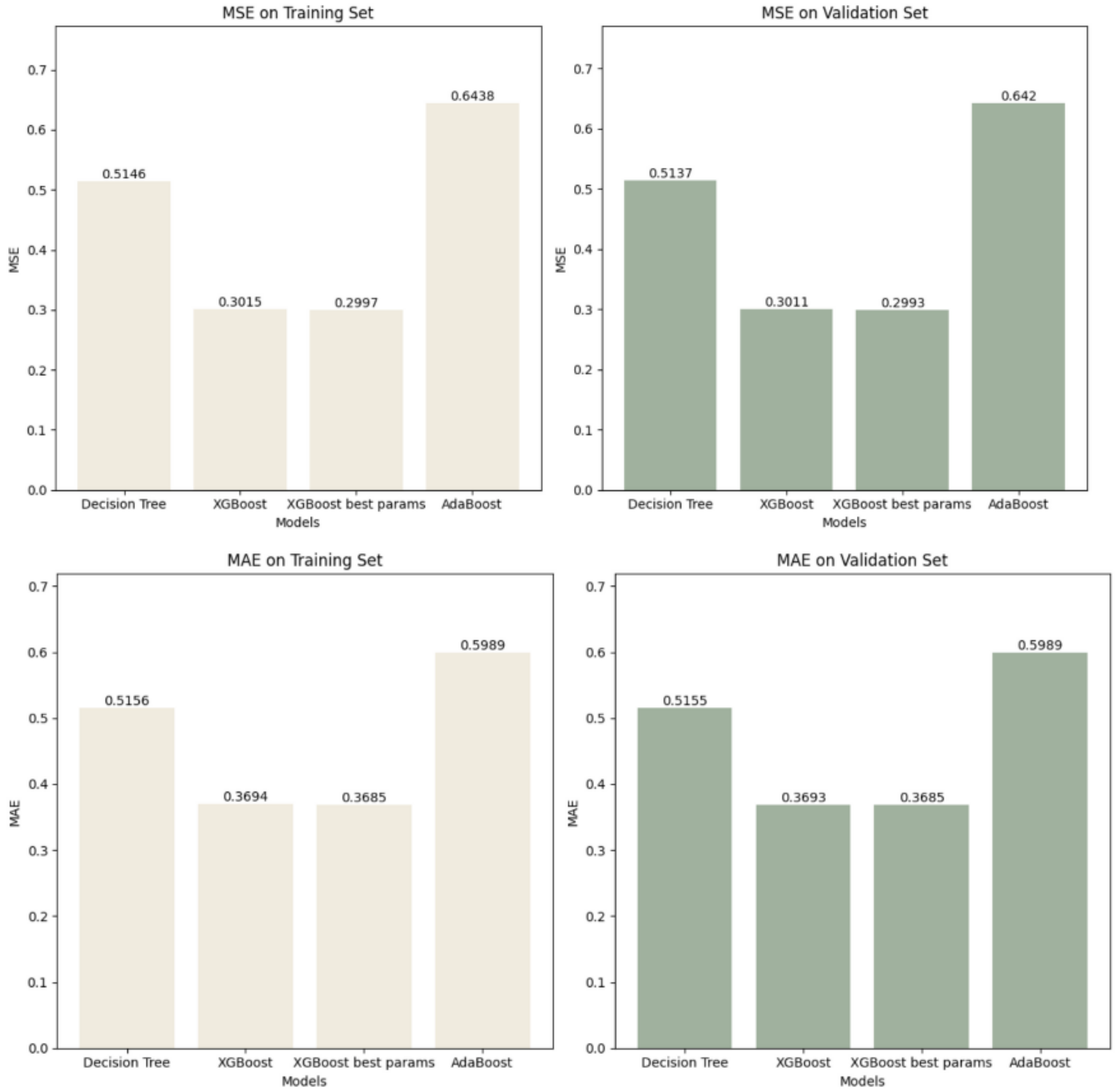


Figure 7: Bar chart of MSE and MAE score for Decision Tree, XGBoost and Adaboost

On the validation set, the optimally tuned XGBoost has a slightly higher MSE of 0.2993 compared to the default XGBoost's 0.3011, but it still outperforms the Decision Tree (MSE: 0.5137) and AdaBoost (MSE: 0.642).

In terms of MAE, the optimally tuned XGBoost (0.3685) slightly outperforms the default XGBoost (0.3694) on the training set, highlighting its improved predictive capability. XGBoost outperforms both the Decision Tree (MAE: 0.5156) and AdaBoost (MAE: 0.5989) once more.

On the validation set, the optimally tuned XGBoost has a slightly higher MAE of 0.3685 than the default XGBoost's 0.3693, but it still outperforms the Decision Tree (MAE: 0.5155) and AdaBoost (MAE: 0.5989).

The inability of Decision Tree to handle complex data relationships may result in higher errors. AdaBoost may be ineffective in improving the performance of a poor learner (Decision Tree) in this task. Both models may be overfitting because their errors on the validation set are higher than on the training set.

- **TensorFlow Decision Forests (TF-DF) :**

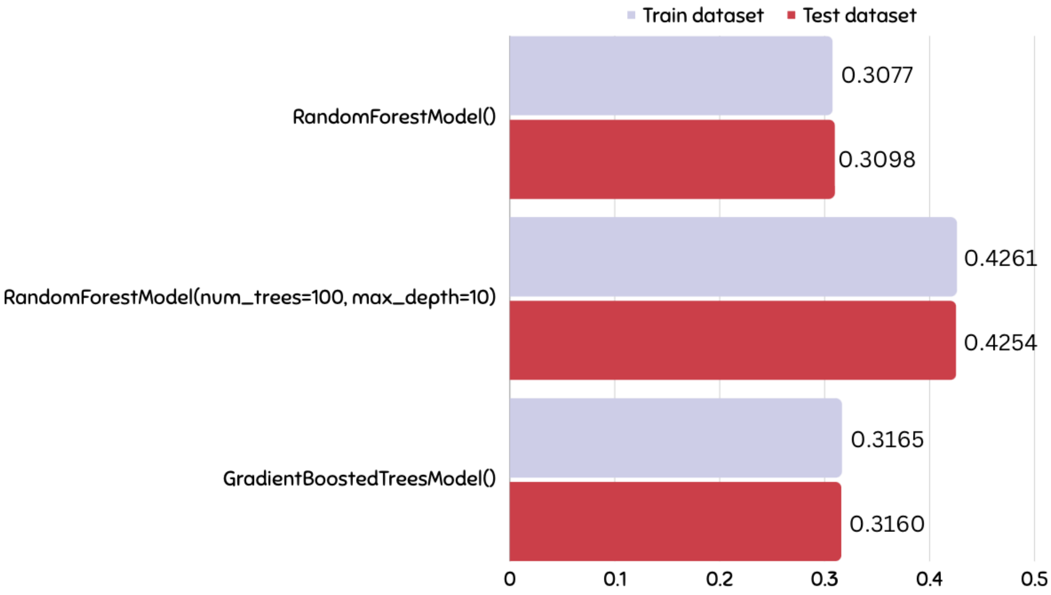


Figure 8: The bar chart of MSE score for TensorFlow Decision Forest (TF-DF)

The bar chart above represents the MSE score results on the training and testing sets for each model. It can be seen that the RandomForest model, with default hyperparameters, provided the lowest MSE score of 0.3098, which we will define as the best model among these three.

The key finding during the experiment with these three models is that, for the RandomForest model, we attempted to adjust the model architecture by tuning the num_trees and max_depth hyperparameters. However, it became apparent that the MSE score increased by approximately 0.12 points compared to the default settings, leading us to the decision not to make further adjustments. Instead, for TensorFlow Decision Forests (TF-DF), we explored another model, which is the Gradient Boosted Trees. The result was that RandomForest slightly outperformed Gradient Boosted Trees, with a lower MSE by 0.01.

- **Keras :**

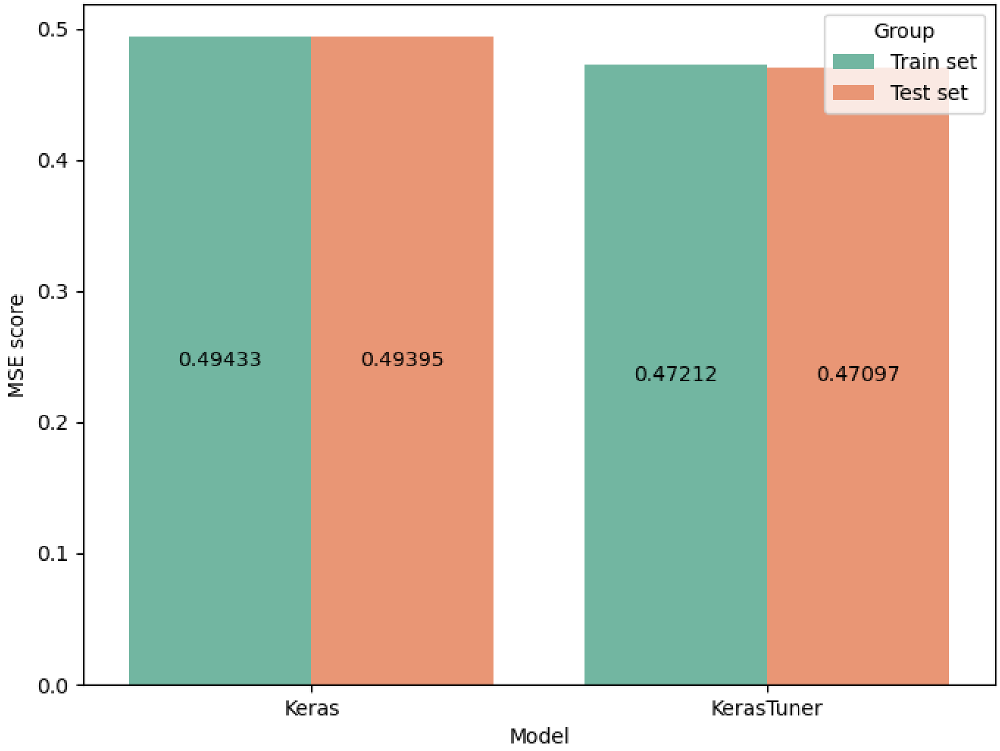


Figure 9: The bar chart of MSE score of Keras and kerasTuner

In this analysis, we assess the performance of two machine learning models, Keras and KerasTuner, focusing on their Mean Squared Error (MSE) scores. The Keras model demonstrates a test set MSE of 0.49395. On the other hand, the KerasTuner model achieves a test set MSE of 0.47097.

However, Both the Keras and kerasTuner model falls short of the other models in terms of performance. The reason behind this performance difference may be attributed to the limited amount of data available for training.

c. Business Impact and Benefits

The development of a data product for estimating local airfare with overall MSE scores less than 0.6 and MAE scores less than 0.6 has several significant business implications and benefits that show that this estimate is accurate.

Business use cases	*Impact*	*Benefits*	*Value Created*
1.Customised Travel Planning:	This enables users to make informed decisions based on their travel preferences and budget.	This aids in the avoidance of budget-related surprises and the alignment of travel plans with financial constraints.	The value created here is user satisfaction and confidence in their travel plans. This can lead to increased platform loyalty and engagement.
2.Budget Optimization:	The model assists users in locating cost-effective travel options. Budget constraints allow users to modify their plans.	By selecting flights that are within their budget, travellers can potentially save money and make travel more affordable.	The value is generated in terms of cost savings and financial planning for travellers. If users consistently find the best options on the platform, it can lead to increased bookings and customer loyalty.
3. Comparing Travel Options:	Users can easily compare fare estimates for multiple travel options, such as different airlines, travel dates, or routes.	Informed decision-making is facilitated, and users can choose the travel options that best suit their needs, whether it's based on cost, convenience, or other factors.	The value comes from providing users with the ability to make choices that align with their preferences and priorities. This can enhance user satisfaction and loyalty.
4. Historical Fare Analysis:	Users can analyze historical airfare trends to determine optimal booking times, which can be especially useful for planning travel in advance.	By analyzing historical fare data, users can identify trends and patterns that help them secure the best fares, potentially saving money on future bookings.	Value is generated by helping users make data-driven decisions, which can lead to cost savings and better travel experiences.

Table 5: Business use cases, Impact, Benefits and Value created

In summary, the models have a significant impact and benefit on users by providing accurate and reliable airfare estimates for their travel planning needs. These advantages include cost savings, time savings, and increased customer satisfaction, which can lead to increased customer retention and loyalty. The model helps to address the identified challenges in budget planning and fare analysis, while also leveraging opportunities for better-informed travel decisions.

d. Data Privacy and Ethical Concerns

Topic	***Description***
Personal Identifiers	The dataset may contain identifiers (e.g., legId) that could potentially lead to the identification of individuals, posing privacy risks.
Search and Flight Date	Information like searchDate and flightDate could reveal sensitive travel plans, which might breach user privacy expectations.
Location Data	Airport codes (startingAirport and destinationAirport) can be used to infer individuals' travel locations and patterns, raising privacy concerns.
Fare Information	Pricing data (baseFare and totalFare) can be sensitive and exploited for unethical practices, such as price discrimination.
Travel Duration and Distance	Data like travelDuration and totalTravelDistance can disclose the mode of transportation and potentially sensitive information about travelers.
Segment-Level Data	Segment-level details, including departure times, airline names, and equipment descriptions, can be used to track and profile individuals.
Non-Stop and Cabin Information	Columns like isNonStop and segmentsCabinCode may reveal traveler preferences without their consent.

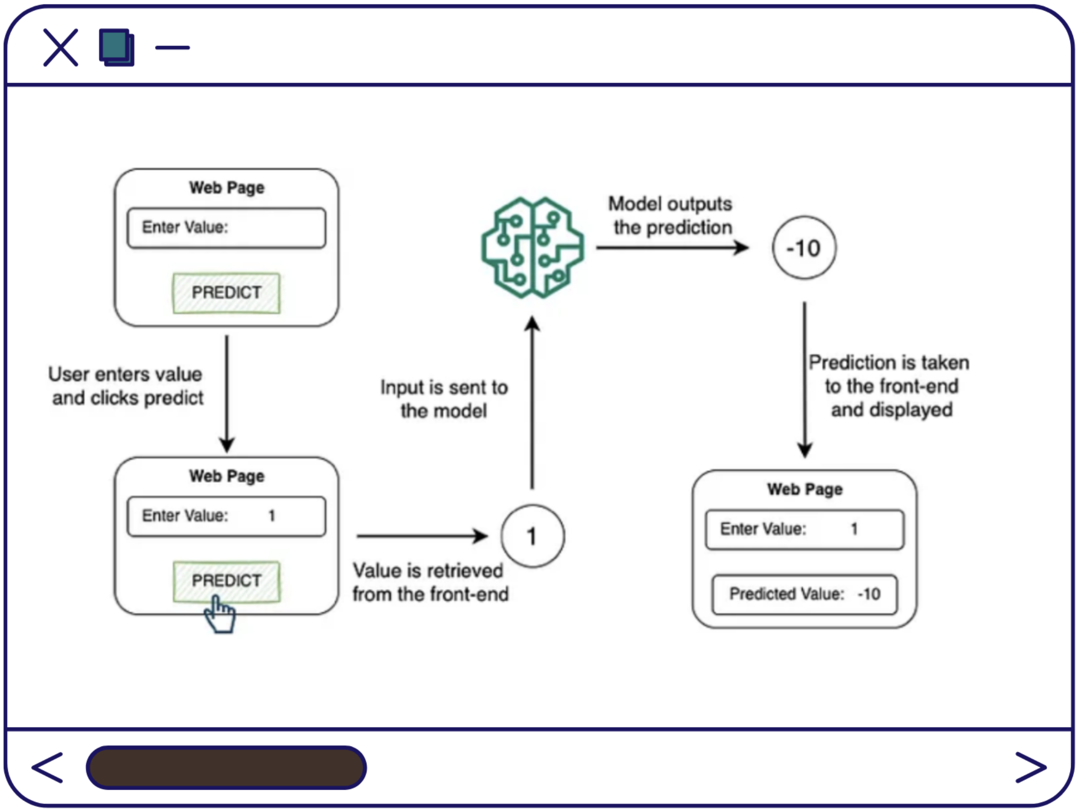
Table 6: Data Privacy and Ethical Concerns

To address these concerns, the dataset should be de-identified or anonymized, and strict access controls should be implemented. Ethical guidelines, including transparency, fairness, and obtaining user consent, should be established and followed to responsibly handle the data.

7. Deployment

a. Model Serving

With the use of web application such as Streamlit entails creating the front-end (or user interface) of the application for the user to interact with and provide input, then integrating it with the back end to pass the gathered data to the machine learning model for prediction.



Web Page
Enter Value:
PREDICT
User enters value and clicks predict
Web Page
Enter Value: 1
PREDICT
Value is retrieved from the front-end
1
Input is sent to the model
Model outputs the prediction
-10
Prediction is taken to the front-end and displayed
Web Page
Enter Value: 1
Predicted Value: -10

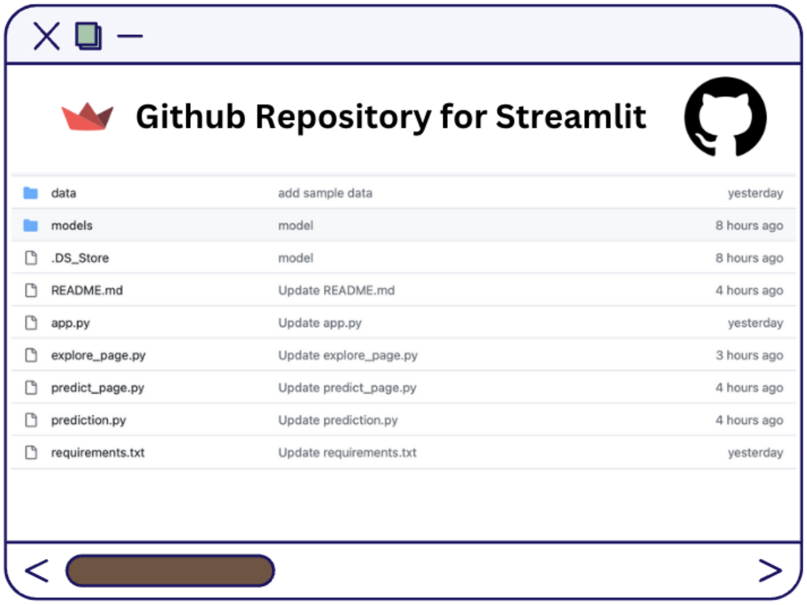
Figure 10: Web application workflow (Chawla, 2022)

Deployment Process

1. **Save trained models**

Train machine learning model that is ready for deployment. Save it in a format that can be easily loaded such as a joblib file, keras file and pb file.

2. Streamlit Files and Folders



Github Repository for Streamlit

data	add sample data	yesterday
models	model	8 hours ago
.DS_Store	model	8 hours ago
README.md	Update README.md	4 hours ago
app.py	Update app.py	yesterday
explore_page.py	Update explore_page.py	3 hours ago
predict_page.py	Update predict_page.py	4 hours ago
prediction.py	Update prediction.py	4 hours ago
requirements.txt	Update requirements.txt	yesterday

Figure 11: Show Streamlit files and folders

In the same repository we create 5 new files and 2 folders for the application

File name	Descriptions
2.1 app.py	The main script for the streamlit application.
2.2 predict_page.py	The predictions page, which makes use of the trained model.
2.3 explore_page.py	The sample data exploration page.
2.4 prediction.py	The file used to save untrained models for use in predict_page.py.
2.5 requirements.txt	A file that contains all of the requirements modules and specifies which libraries must be installed in the deployment environment
2.6 Data folder	A sample of our dataset for display on the explore page and for fitting a KNN model on the prediction page.
2.7 Models folder	Contains the best model from our model training, ready to be used without re-fitting in the predict page.

Table 7: file and script for Streamlit application

3. Version Control

To manage code changes and collaboration, we host our Streamlit application files and models on Github.

4. Streamlit Sharing (Streamlit.io)

We host our app on Streamlit Sharing (https://www.streamlit.io/sharing). This app is public and searchable through our subdomain https://crybaby-fareprediction.streamlit.app/.

Considerations for Real-World Implementation:

Considerations	Descriptions
1. Data Pipeline	Use data pipeline is robust and reliable, particularly in real-world scenarios where data quality and availability may vary.
2. Scalability	Prepared to scale application if user traffic increases.
3. Cost Management	Monitor hosting costs and weigh the financial implications of scaling or increasing usage.
4. Implement monitoring and logging	to track the performance and health of deployed application.
5. Error Handling	Have a solid error-handling strategy in place to handle unexpected errors or issues gracefully.

Table 8: Considerations

Challenges and Recommendations:

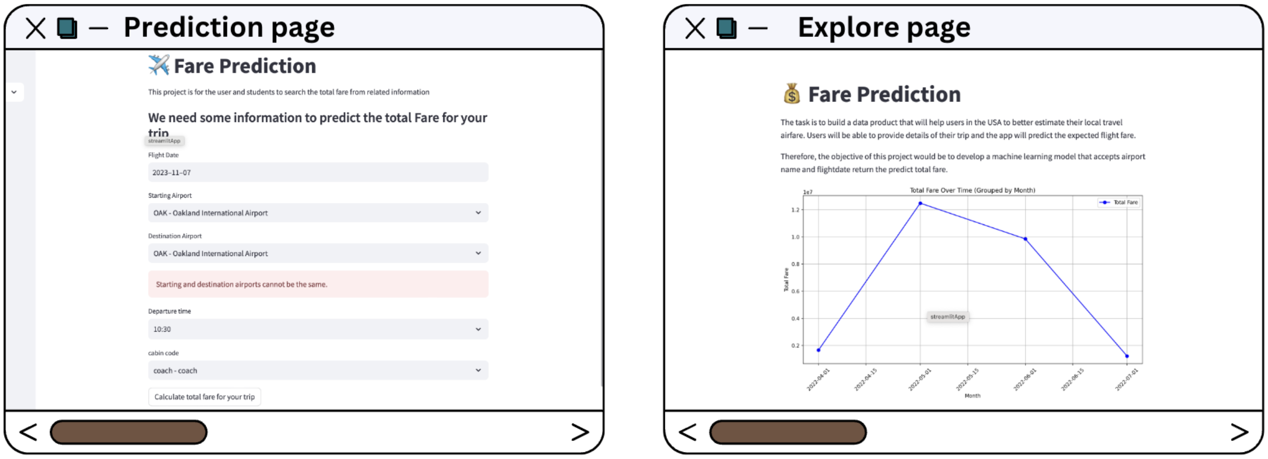
1. Plan for periodic data and model updates, and have a process in place for retraining and deploying updated models.

2. Take security precautions seriously, especially when dealing with user data. As needed, use authentication and authorisation.

3. Performance Optimisation: Improve the performance of the application to ensure responsiveness and a smooth user experience.

b. Web App

Streamlit is a tool for creating a visually appealing Web App that can hold graphics, images, text, tables, maps, and machine learning models.



Prediction page

✈️ Fare Prediction

This project is for the user and students to search the total fare from related information

We need some information to predict the total Fare for your trip

Flight Date: 2023-11-07

Starting Airport: OAK - Oakland International Airport

Destination Airport: OAK - Oakland International Airport

Starting and destination airports cannot be the same.

Departure time: 10:30

cabin code: coach - coach

Explore page

💰 Fare Prediction

The task is to build a data product that will help users in the USA to better estimate their local travel airfare. Users will be able to provide details of their trip and the app will predict the expected flight fare.

Therefore, the objective of this project would be to develop a machine learning model that accepts airport name and flightdate return the predict total fare.

Figure 12: shows web application.

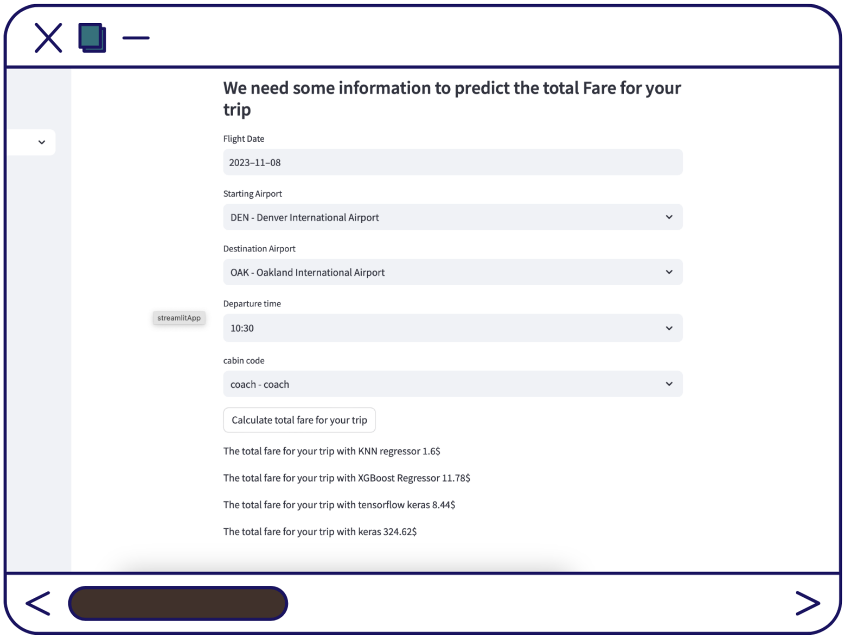
Purpose and Main Functionalities: Based on user inputs such as flight date, starting airport, destination airport, departure time, and cabin code, the web application predicts the total fare for a local trip. To provide users with fare estimates, the application employs four different prediction models. Its primary functions are as follows:

1. User Input: Users can enter their flight information, such as the date, starting airport, destination airport, departure time, and cabin code.

2. Fare Prediction: The application uses four distinct models to predict the total fare for the local trip based on the provided inputs.

Setting Up and Launching the Web Application:

The application is configured to accept five inputs: flight date, departure airport, destination airport, departure time, and cabin code. However, it requires 12 features for the internal application. We set totalTravelDistance to 1596 miles, which is the average totalTravelDistance from our dataset, and isNonStop to true when the user selects only coach, premium coach, business, and first, otherwise it returns false. Also, the isBasicEconomy property is set to always false because it is frequently false in our dataset.



We need some information to predict the total Fare for your trip

Flight Date: 2023–11–08

Starting Airport: DEN - Denver International Airport

Destination Airport: OAK - Oakland International Airport

Departure time: 10:30

cabin code: coach - coach

The total fare for your trip with KNN regressor 1.6$

The total fare for your trip with XGBoost Regressor 11.78$

The total fare for your trip with tensorflow keras 8.44$

The total fare for your trip with keras 324.62$

Figure 13: Web application input requirements.

Step to hold this application

1. Clone the application repository from GitHub.
2. Install the required dependencies and libraries.
3. Open workspace at share.streamlit.io, to create a new app.
4. Access the web application through a web browser and fix the error from the managed app.

Potential use, benefits and Commercialization

There are several use cases, benefits and commercialization as below table:

Potential Users and Use Cases	Potential Benefits	Potential Commercialization
1. Travelers and business travellers: Individuals planning local trips who want to estimate their total fare, People on business trips can assess fare options efficiently.	**1. Convenience**: Users can quickly obtain fare estimates for their local trips, aiding in travel planning.	**1.Subscription Model**: Offer premium features or ad-free versions for a subscription fee.
2. Travel Agencies: Agencies can use this tool to provide cost estimates to their clients.	**2. Cost Estimation**: It helps users make informed decisions about their travel expenses.	**2. Partnerships**: Collaborate with airlines or travel agencies for integration and referral agreements.

3. Airline Personnel: Airline employees may use it for internal pricing comparisons.	**3. Time Savings**: Avoid the need to manually calculate fares or compare different options.	**3. Data Monetization**: If the application collects data, anonymized and aggregated data could be sold to interested parties.

Table 9: Potential use case, benefits and commercialization of Web app

Current Limitations and Potential Improvements:

Limitations	**Potential Improvements**
Resource Limits	The application run into resource limits due to large files. To address this, consider optimizing the code or using more efficient data storage and retrieval methods.
Limited Cabin Code Options	The application only supports a specific set of cabin codes. Expanding this list to cover more cabin types would make it more versatile.
Data Accuracy	The accuracy of fare predictions depends on the underlying data quality. Continuously updating the dataset and improving data sources could enhance prediction accuracy.
User Authentication	The error pop-up related to reaching free-resource limits could be mitigated by implementing user authentication and session management to better control resource usage.
User Interface Enhancement	Improving the user interface design and user experience could make the application more user-friendly and visually appealing.
Model Selection and Tuning:	Regularly updating and fine-tuning the prediction models can lead to more accurate and reliable fare estimates.

Table 10: Limitations and Potential Improvement

8. Collaboration

a. Individual Contributions

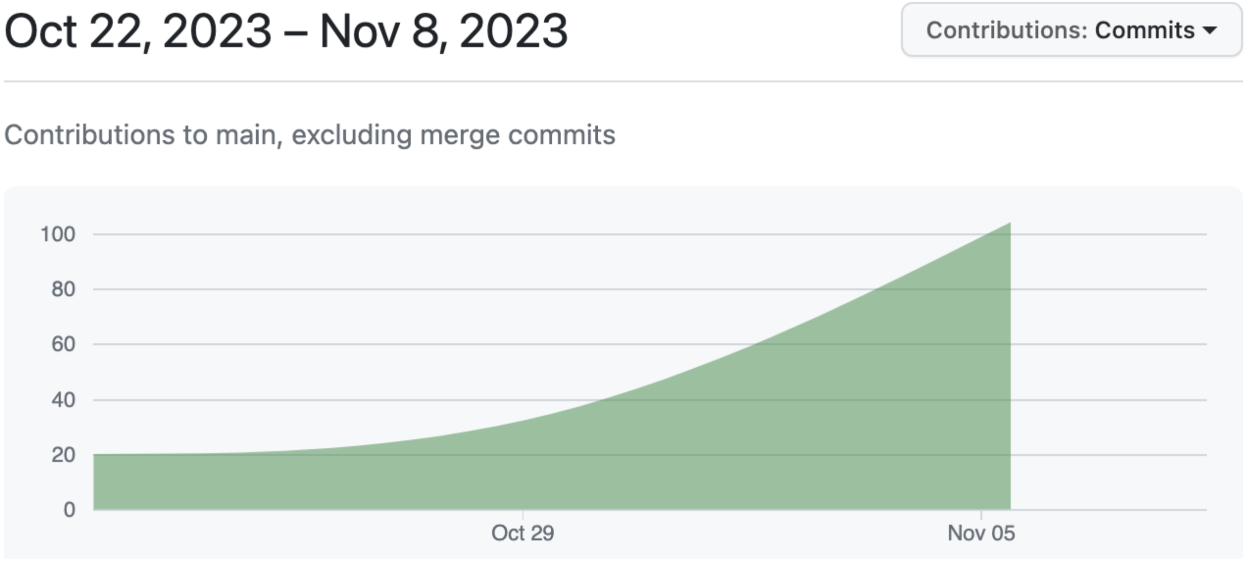
Team Members	***GitHub Username***	***Contribution***
Panalee Makha - Student A	panalee-mk	<PM_notebooks><Makha_Panalee-14367914-knn.ipynb>
Wongwara Wijara - Student B	wongwara	<WW_notebooks><Wijara_Wongwara-14191732-xgboost.ipynb> <WW_notebooks><EDA.ipynb>
Thirada Tiamklang - Student C	thirada2799	<TT_notebooks><Tiamklang_Thirada-14337188-tfdf.ipynb> <TT_notebooks><eda.ipynb>
Thanchanok Phuawiriyakul - Student D	bpthn	<TP_notebooks><Phuawiriyakul_Thanchanok-24582239-keras.ipynb> <TP_notebooks><clean.ipynb> <TP_notebooks><corr.ipynb>

Table 11: Individual contribution

b. Group Dynamic

Our team's collaboration was successful due to a combination of digital tools and in-person interactions. Google Drive handled large files, while GitHub facilitated code sharing and version control. Line chat supported real-time communication, while in-person meetings and co-location enhanced face-to-face interactions.

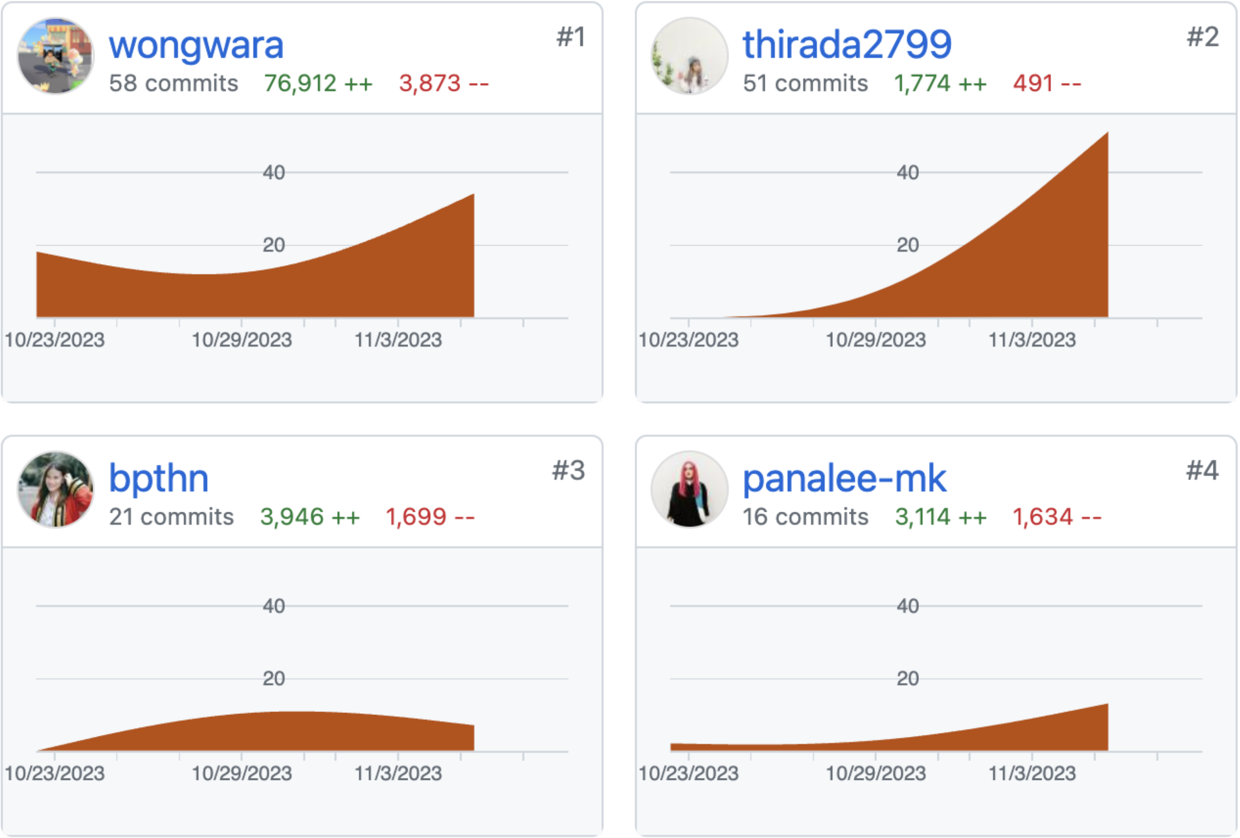
Our team's adaptability and flexibility enabled us to make changes as needed. In conclusion, our collaboration was a well-balanced mix of digital and in-person interactions that resulted in effective teamwork and successful project outcomes.


Oct 22, 2023 – Nov 8, 2023

Contributions: Commits

Contributions to main, excluding merge commits

Figure 14: Group Contributions on GitHub Repository


wongwara #1
58 commits 76,912 ++ 3,873 --

thirada2799 #2
51 commits 1,774 ++ 491 --

bpthn #3
21 commits 3,946 ++ 1,699 --

panalee-mk #4
16 commits 3,114 ++ 1,634 --

Figure 15: Individual Contributions on GitHub Repository

c. Ways of Working Together

Meeting 1:

Date	21 October 2023
Topic	Model selection
Details	In our project, we have chosen a different model for each member for training. We are now preparing the data through comprehensive data preprocessing steps, ensuring it is in optimal condition for training and model evaluation.
Attendees	Panalee Makha, Wongwara Wijara, Thirada Tiamklang, Thanchanok Phuawiriyakul

Meeting 2:

Date	25 October 2023
Topic	Feature selection and Correlation
Details	In our data analysis, we employed a feature selection process by assessing the correlation scores of each variable with the target variable, 'total fare.' Those features with a correlation score exceeding a threshold of |0.24| were carefully chosen for our model training. The selected features that met this criterion include: 'totalTravelDistance,' 'isNonStop,' 'isBasicEconomy,' 'startingAirport,' 'destinationAirport,' 'segmentsCabinCode,' 'flightDate_day,' 'flightDate_month,' 'flightDate_year,' 'DepartTime_hour,' 'DepartTime_minute,' 'DepartTime_second,' and 'totalFare.' These features were considered the most influential in predicting 'total fare'.
Attendees	Panalee Makha, Wongwara Wijara, Thirada Tiamklang, Thanchanok Phuawiriyakul

Meeting 3:

Date	31 October 2023
Topic	Compare models performance and discuss streamlit
Details	During our meeting, we conducted an in-depth analysis of several machine learning models to predict 'total fare.' Next, we identified K-Nearest Neighbors (KNN) as the best-performing model. For our Streamlit project, we have decided to integrate the top model selections of each team member. This approach enhances the application's versatility, enabling users to experience different modelling techniques.
Attendees	Panalee Makha, Wongwara Wijara, Thirada Tiamklang, Thanchanok Phuawiriyakul

d. Issues Faced

Technical issued

- Due to the large file size of the KNN model files, knn_fit.joblib and knn_fit.joblib.zip, they cannot be uploaded to GitHub. The best KNN model, which is saved in the models folder, does not include fitted data which can access the KNN model file, including the fitted data, through the following Google Drive link: https://drive.google.com/drive/folders/12ISspcn9g2bXIPBGeUGjVq1uTjT2XJup

The prediction result challenges:

Result Challenges	**Description**
Limited Data Range	Forecasting airfare prices for a date in 2023 becomes inherently difficult with a training dataset spanning the period from April 2022 to July 2022. Suggestion: Training dataset must be supplemented with historical data that extends into 2023, allowing the model to better understand evolving pricing trends.
External Factors	Flight prices are highly volatile due to a variety of external factors such as seasonality, holidays, economic conditions, and demand fluctuations. Suggestion: Includes features that capture the subtleties of seasonality and other external factors
Feature Importance	The feature "flightDate_year" is important in predictions, but its weighting within the model may not be proportional to its importance. If it is fixed in 2023, the model's predictive capability may be compromised. Suggestion: feature importance must be reevaluated, and the model's focus on the "flightDate_year" feature must be adjusted.

Table 12: Challenges in result

9. Conclusion

All models produced impressive results, with Mean Squared Error (MSE) values consistently below 0.6. The K-Nearest Neighbours (KNN) model performed well in predicting 'total fare' and was chosen as the best option. The project met its goal of providing accurate airfare predictions, as evidenced by the low MSE and MAE scores.

The project produced an easy-to-use Streamlit platform with an Explore Page and a Prediction Page. The Explore Page displays insightful monthly fare trends, whereas the Prediction Page allows users to enter flight information and receive fare predictions from four models. The use of the Streamlit platform made the models more accessible to a wider audience, achieving the goal of usability and accessibility.

To enhance prediction accuracy for future dates, we recommend the following actions:

1. **Collect and integrate historical data spanning 2023 and beyond into the training dataset**, allowing the model to accommodate future scenarios.

2. **Add variables capable of capturing seasonality, holiday effects**, and other external influences on airfare prices to the feature set.

3. **Ensure meticulous feature engineering and preprocessing** to effectively harness the underlying complexities of the data.

4. **Reevaluate the model's feature importance** and overall performance to fine-tune its parameters and align them with the problem's specific requirements.

Given the airline industry's multifaceted and dynamic nature, it is critical to recognise the inherent complexity of accurately predicting airfare prices. To produce reliable predictions, effective solutions frequently necessitate a combination of domain expertise and data-driven refinement.

10. References

Chawla, A. (2022, October 20). Deploying machine learning models with heroku. Medium. https://towardsdatascience.com/deploying-machine-learning-models-with- heroku-4dec1df87f71

How to label ENCODE while iter through the list of list in Python. (n.d.). Stack Overflow. https://stackoverflow.com/questions/59351829/how-to-label-encode-while-iter-through-the-list-of-list-in-python

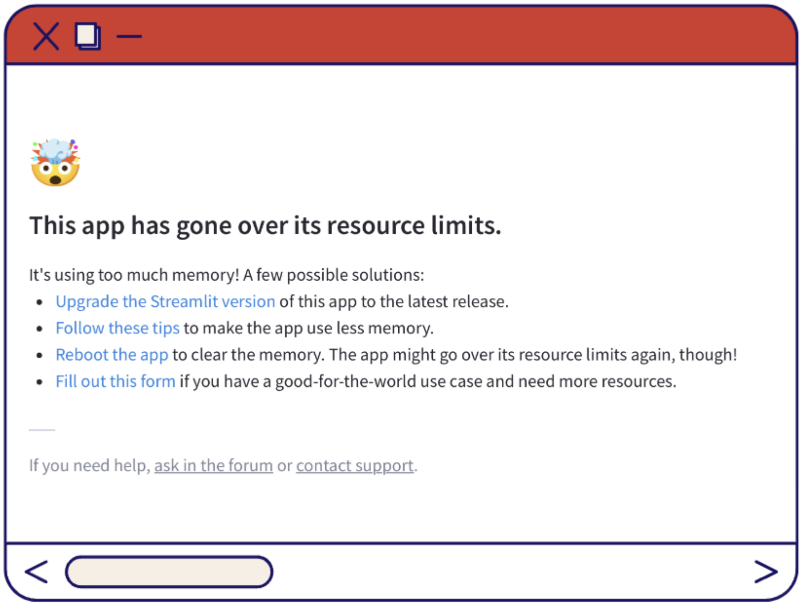
Keras Team. (2021, October 27). *Keras documentation: Getting started with KerasTuner*. Keras: Deep Learning for humans. https://keras.io/guides/keras_tuner/getting_started/

Keras Team. (n.d.). *Keras documentation: Regression losses*. Keras: Deep Learning for humans. https://keras.io/api/losses/regression_losses/

11. Appendix

Error might face during the model deployment

This occurs because our application reaches the free-resource limits from streamlit due the large size of files. When you access the application, you may see this error, which you can reboot using your github account (github collaborative).



🤯

This app has gone over its resource limits.

It's using too much memory! A few possible solutions:

- Upgrade the Streamlit version of this app to the latest release.
- Follow these tips to make the app use less memory.
- Reboot the app to clear the memory. The app might go over its resource limits again, though!
- Fill out this form if you have a good-for-the-world use case and need more resources.

If you need help, ask in the forum or contact support.

Figure 16: show error in the application

Log in to Streamlit.io with your github account, then go to manage the app, and finally reboot the application. However, when we finish the project, we test the web app and it works flawlessly.

Data Understanding:

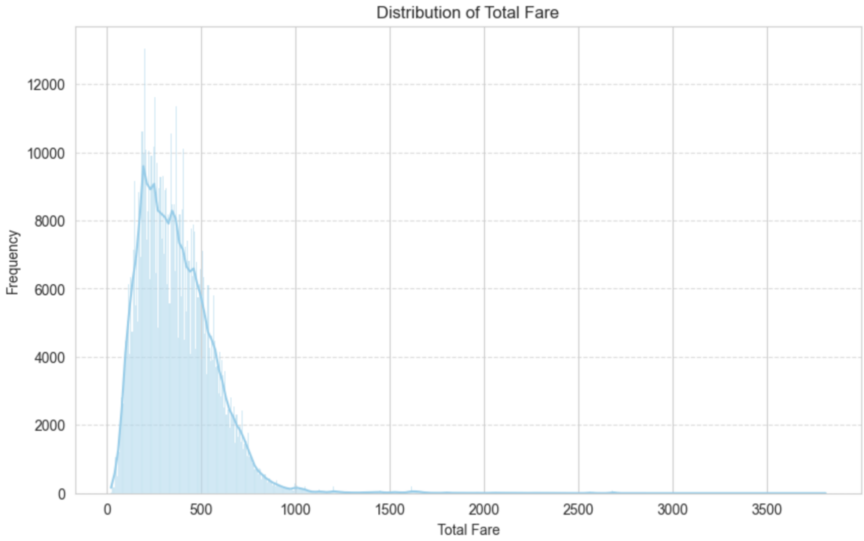


Figure 17: Distribution of Total fare

The total fare distribution is skewed to the left, with the most frequent occurrence of total fare values exceeding 10,000 times. Furthermore, a sizable proportion of total fare values fall between $300 and $500. This suggests that the majority of total fare values are concentrated in the lower range, with a long tail extending to the right, where a few extremely high values may exist. Lower fares are more common, according to the left-skewed distribution, while higher fares are less common but can have a significant impact on the overall distribution due to their high values.

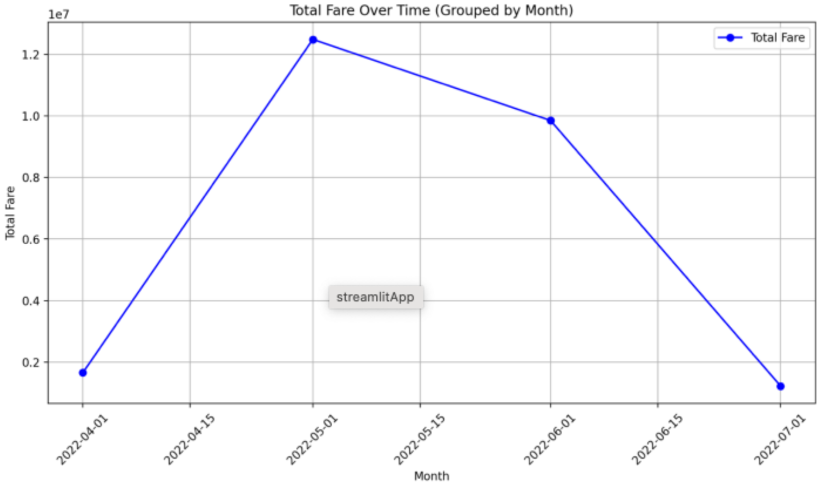


Figure 18: Total Fare over time

The average total fare in each month shows that the highest total fare is in May 2022, followed by a slight decrease in June and July 2022. According to this trend, May is the peak month for total fare, with the highest average. However, there is a slight decrease in average total fare in June and July, which could be influenced by seasonality, demand, or pricing strategies. Analysing these monthly fluctuations can provide insights into travel trends and assist businesses in making informed pricing and marketing decisions.

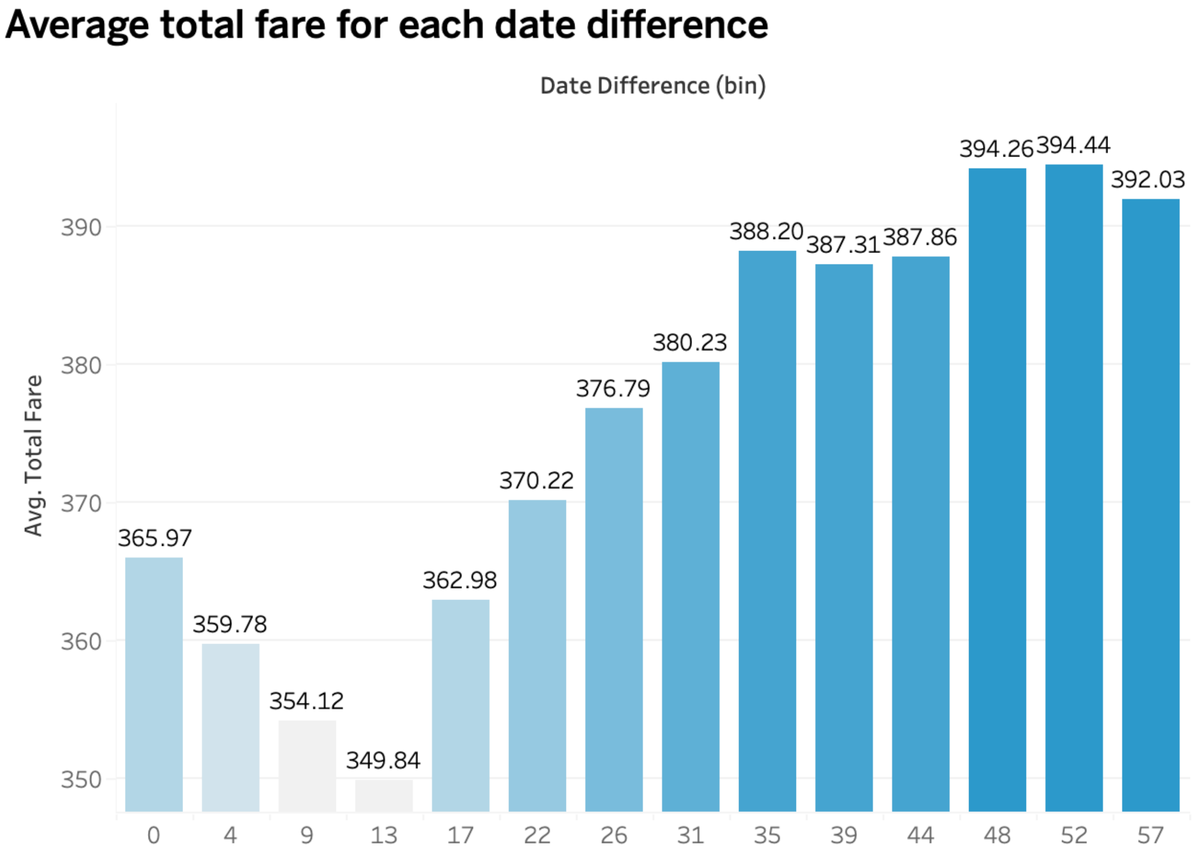


Figure 19: average total fare for each date difference

The bar chart above represents the average total fare for each date difference, which is the difference between the search date and the departure date. It can be seen that a 13-day difference represents the lowest total fare.