Assignment 3

ML Data Product

November 10, 2023

Group 8

Student Last Name	Student First Name	Student ID	Group Allocation
Chauhan	Anika	14188775	Student A
Dhawale	Kritika	24587661	Student B
Kotak	Sahil	24707592	Student C
Chhetri	Varun	24711703	Student D

Github	Project Repo: https://github.com/Kritz23/adv_mla_at3
Streamlit	App: https://advmlaat3.streamlit.app/

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

Table of Contents

1. Executive Summary

a. Overview

The project aimed to develop a data-driven product designed to enhance the experience of travelers in the USA by providing accurate estimations of local airfare costs. Recognizing the complexity and variability of flight pricing, the project sought to leverage machine learning to deliver personalized fare predictions to users planning their trips.

b. Problem Statement

Travelers frequently encounter challenges when budgeting for air travel due to fluctuating prices and a lack of transparent fare breakdowns. This uncertainty can lead to suboptimal travel planning and unexpected costs. Our project addresses this issue by offering a user-friendly solution that provides clear, customized fare estimates based on a variety of travel parameters.

c. Context

In an age where cost efficiency and planning are paramount, the project was initiated to empower users with the ability to forecast travel expenses accurately. The solution was particularly relevant in the dynamic landscape of the travel industry, where prices are influenced by numerous factors such as demand, seasonality, and route specifics.

d. Results

Upon the app's deployment, users gained the ability to input their travel details and receive immediate fare predictions from the ensemble of models. The application's user interface is straightforward, allowing for quick input validation and fare estimation with minimal effort from the user. The models' predictive performance was quantitatively assessed using the Root Mean Squared Error (RMSE) metric, ensuring accuracy in fare predictions. The RMSE scores across the models were as follows:

- Model 1 (Tensorflow): RMSE of 147.72
- Model 2 (CatBoost Regressor): RMSE of 103.73
- Model 3 (GradientBoost): RMSE of 138.41
- Model 4 (XGBRegressor): RMSE of 136.42

These scores indicate a high level of precision in the fare estimates provided by the app, with lower RMSE values reflecting more accurate predictions. The app's aggregated predictions, which combine insights from all four models, have demonstrated a high degree of reliability, aligning closely with actual fare data.

2. Business Understanding

a. Business Use Cases

The primary business use case for this project was to empower the users in The USA with a tool that could assist them in estimating the local travel airfare. By providing details such as the origin airport, destination airport, departure date, departure time and, cabin type (coach, premium, etc.), the users can receive accurate predictions of what their airfare would be. This tool can enhance the users' decision-making process when it comes to budgeting for flights.

b. Challenges and Opportunities

The challenges and opportunities that this project was motivated by were deeply rooted in the ever-changing nature of airfare pricing. The fluctuation in flight prices can be due to a plethora of factors such as the demand, seasonality, and popularity of flight routes. When users get reliable estimates of the fares, they can plan their travel budget with more efficiency and have a much more seamless experience while coming up with travel plans. Addressing the pain point of uncertainty and variability in airfare costs is where the opportunity lies.

c. Key Objectives

Accurate Predictions and a User-friendly interface were our key objectives for the project.

- **Accurate Predictions**: The primary objective was to machine learning models that produced as accurate results as possible. This accuracy is crucial when it comes to providing users with reliable estimates of their air travel - ultimately leading to increased trust in the product/service.
- **User-Friendly Interface**: Another key aspect of the project was to deploy a user-friendly Streamlit app that would allow users to easily input their start and end points, time of flight, date of departure and cabin details, and deliver accurate airfare information.

d. Stakeholders and their requirements

There are three main stakeholders when it comes to the use of this app:

- **Users**: The primary stakeholders in this scenario are the users who intend to travel locally to The USA. They require accurate and timely airfare predictions so they can make informed decisions when it comes to travel.

- **Travel Agencies**: The app can be beneficial for the travel agencies by helping them better understand market trends and adjust their travel packages according to the predicted airfares.
- **Airlines**: The airlines can optimize their pricing strategies by using the app to understand what factors influence the rise (and fall) in demand.

e. Addressing Stakeholders' Requirements

The project leverages machine learning algorithms to generate accurate predictions, thereby aiming at aiding its users' plan and budget better for their air travel. The varied models ensure the robustness of predictions, and the user-friendly interface makes it easier for a broad spectrum of audience to access the tool. Additionally, the market dynamic insights can help airlines strengthen their strategic decision-making.

3. Data Understanding

Data Source and Collection:

The dataset for this project has been sourced from Expedia, a well-known travel booking website. The given data has a collection of helpful information regarding flight fare, flight type, duration, trip distance, non-stop flight and many more. The data was likely collected by aggregating flight details and pricing information from various airlines and other flight information systems.

Data Shape and Size:

After correctly merging all the raw data, the data had 13519999 (roughly 13.5 million) rows and 23 columns.

a. Features and Significance

The dataset contained several features that were significant for predicting the flight fares:

‘legID’: A unique identifier for each flight, crucial for distinguishing between different flight records

‘Searchdate’: The date on which the fare was searched. This can provide insights into seasonal and price changes according to demand patterns.

‘Flightdate’: The actual date of the flight provides further insights to justify seasonal trends.

‘Startingairport’ and ‘Destinationairport’: Three-digit airport code to understand the most famous airport routes.

‘Farebasiscode’: Provides insights into the restriction of fares for each class type.

‘Travelduration’: The total travel duration of flights. It is very important to understand the pricing strategies for short and long-duration flights.

‘Elapseddays’: The time between the search date and flight date, which can significantly affect the fares.

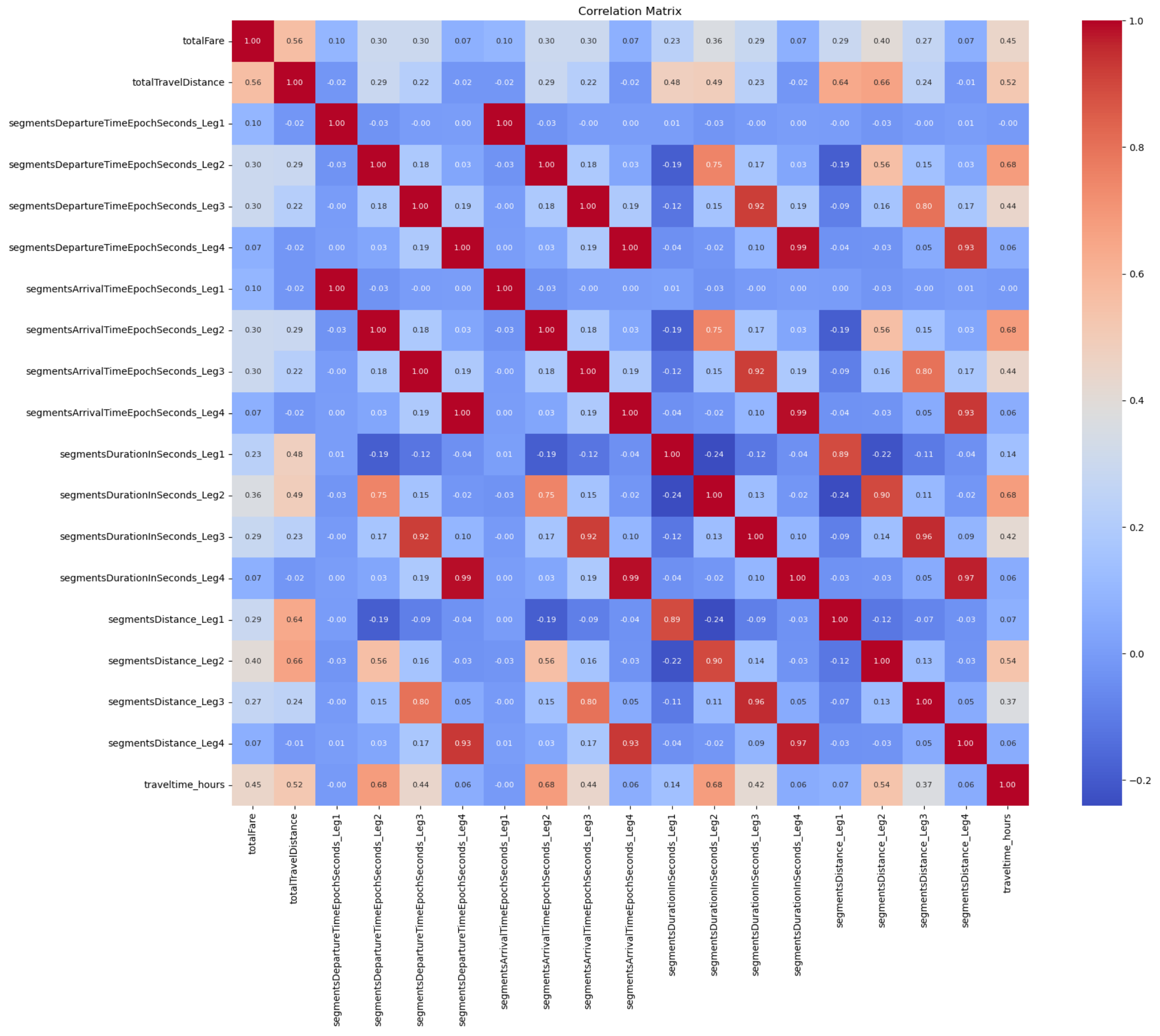
‘Isbasiceconomy’, ‘isrefundable’, ‘isnonstop’: Boolean features indicating respective values.

‘Basefare’ and ‘totalfare’: The target viable total fare which includes taxes and additional fees.

‘Totaltraveldistance’: Distance between the source and destination. Often the pricing is based on distance. Longer distances lead to higher fares.

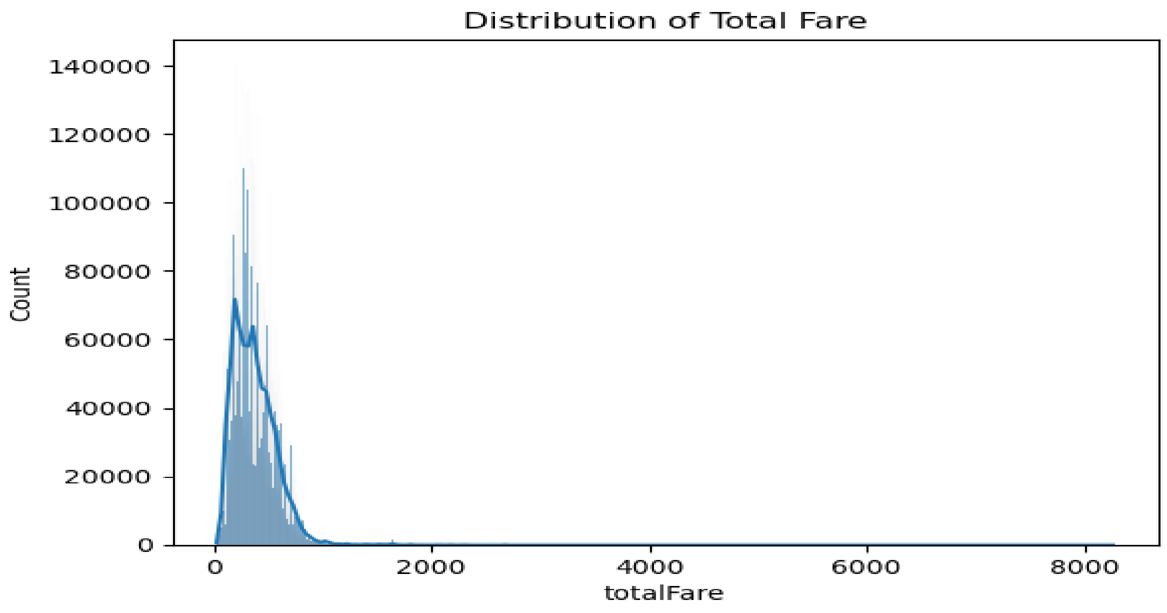
For the rest of the data provided information about the airline names, their departure and arrival times. They also provided information on equipment, each segment's duration and distance. Lastly the column ‘segmentcabincode’ provides information about the cabin class. This is one of the major components of airfare pricing.

A correlation matrix was plotted to understand the relationship of features with the target variable



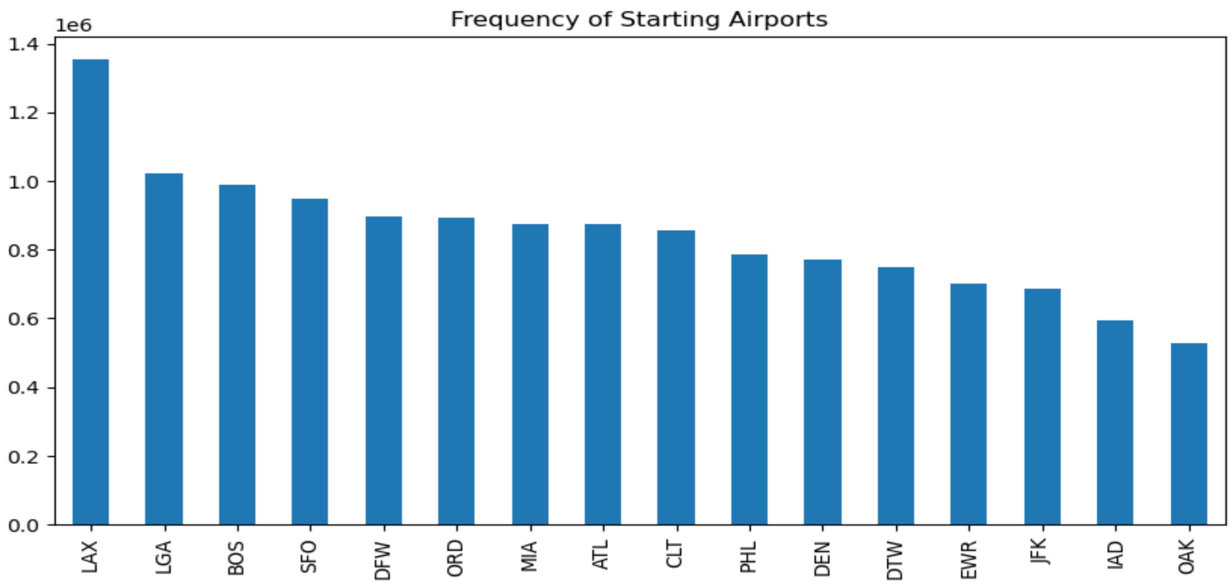
b. EDA

I. <u>Distribution of the target variable total fare:</u>



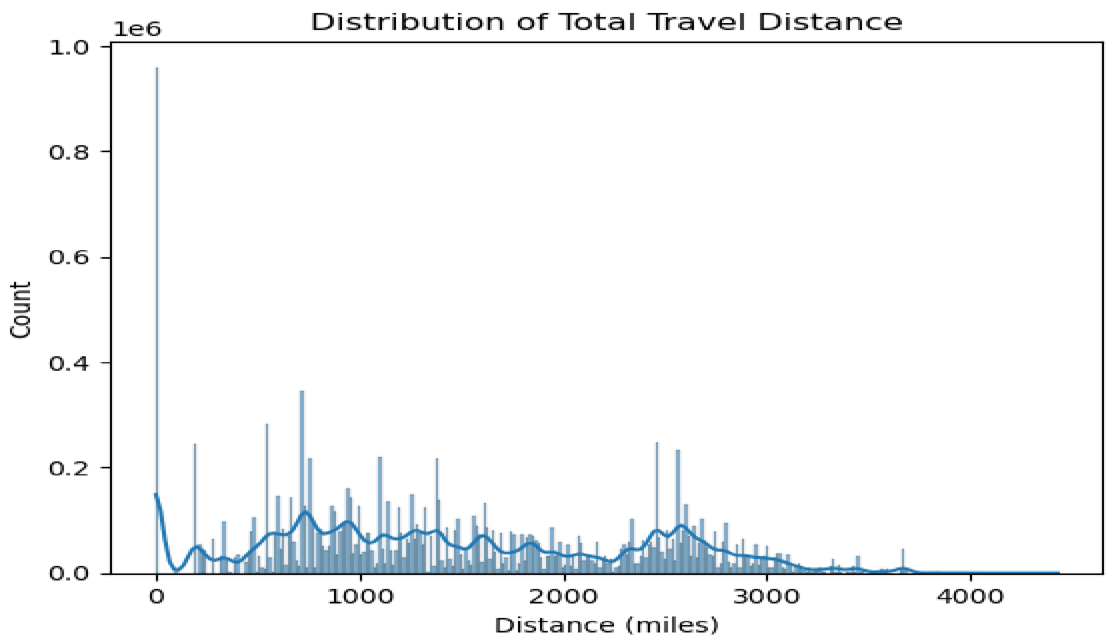
From the graph, we can clearly see that the fare distribution for most of the flights ranges from USD 500 - USD 1000.

II. <u>Frequency of starting airport:</u>



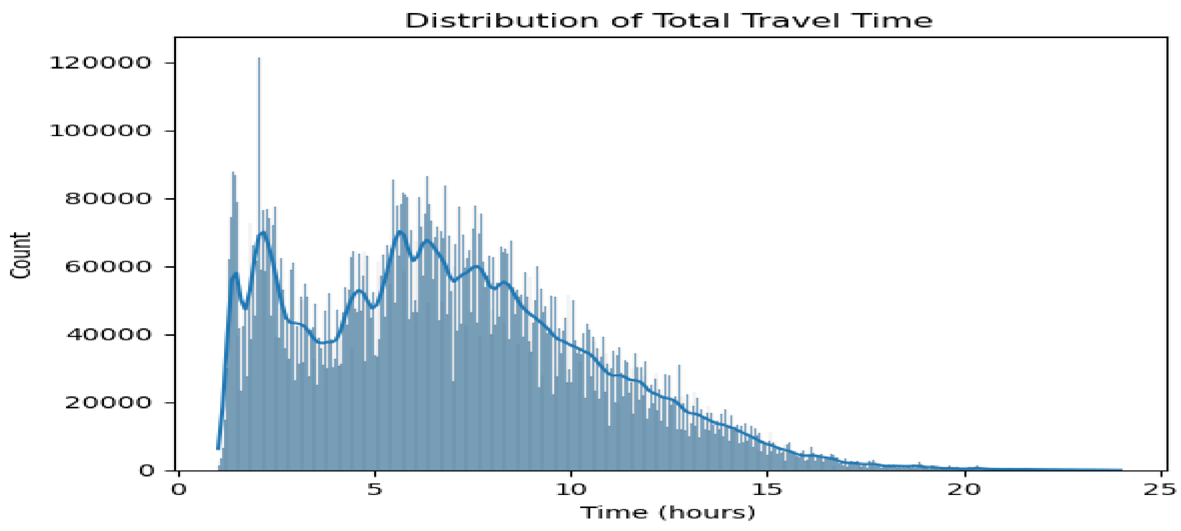
We can clearly see that LAX airport is the airport with the highest starting destination, followed by LGA and BOS.

III. Distribution of total travel distance:



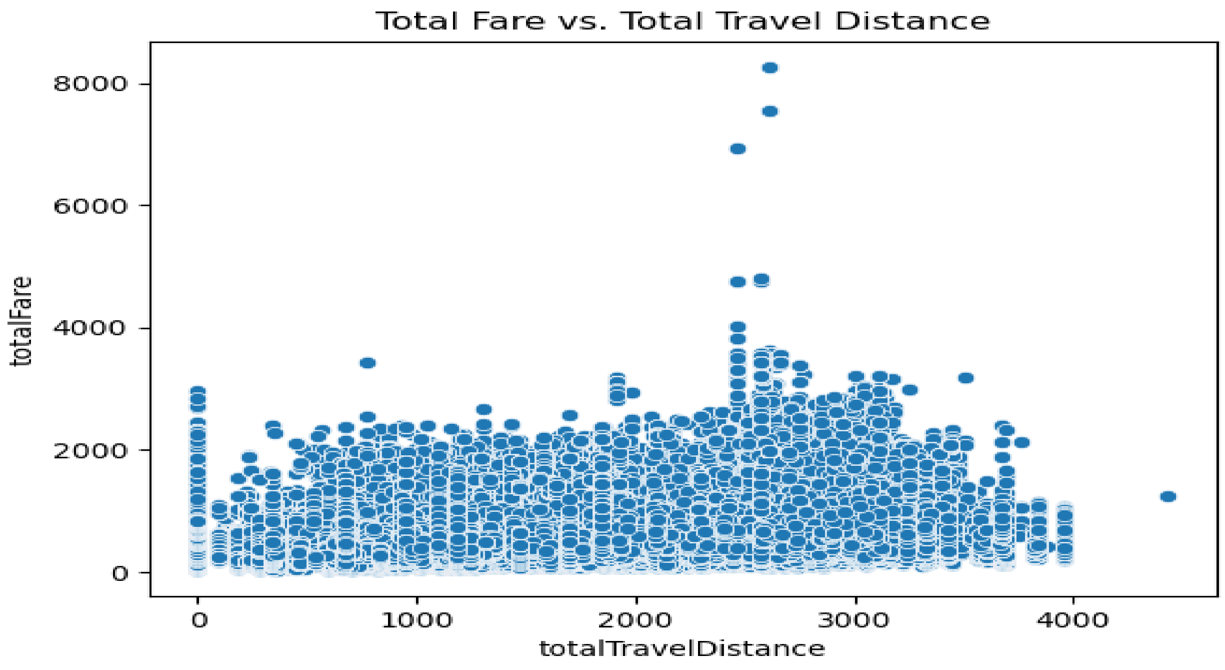
It can be observed that the travel distance distribution mostly ranges near 1000 miles and 2500 miles, indicating some flying behaviour.

IV. Distribution of Total Travel Time:



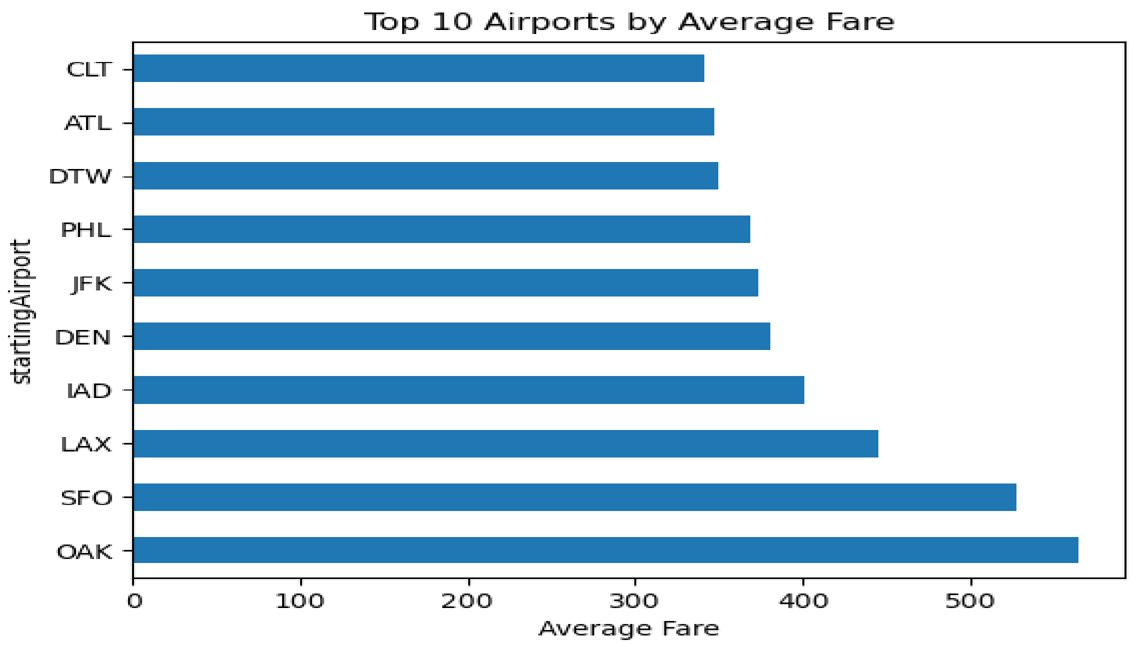
The total travel time mostly ranges from 2 hours to 15 hours, with higher frequencies in a 2-hour flight journey, followed by a 7-10 hour flight journey. This may mean that someone might be travelling to a 2-hour destination time, but since connecting flights would work cheaper, flyers would opt for them.

V. <u>Distribution of Fare According to trip distance:</u>



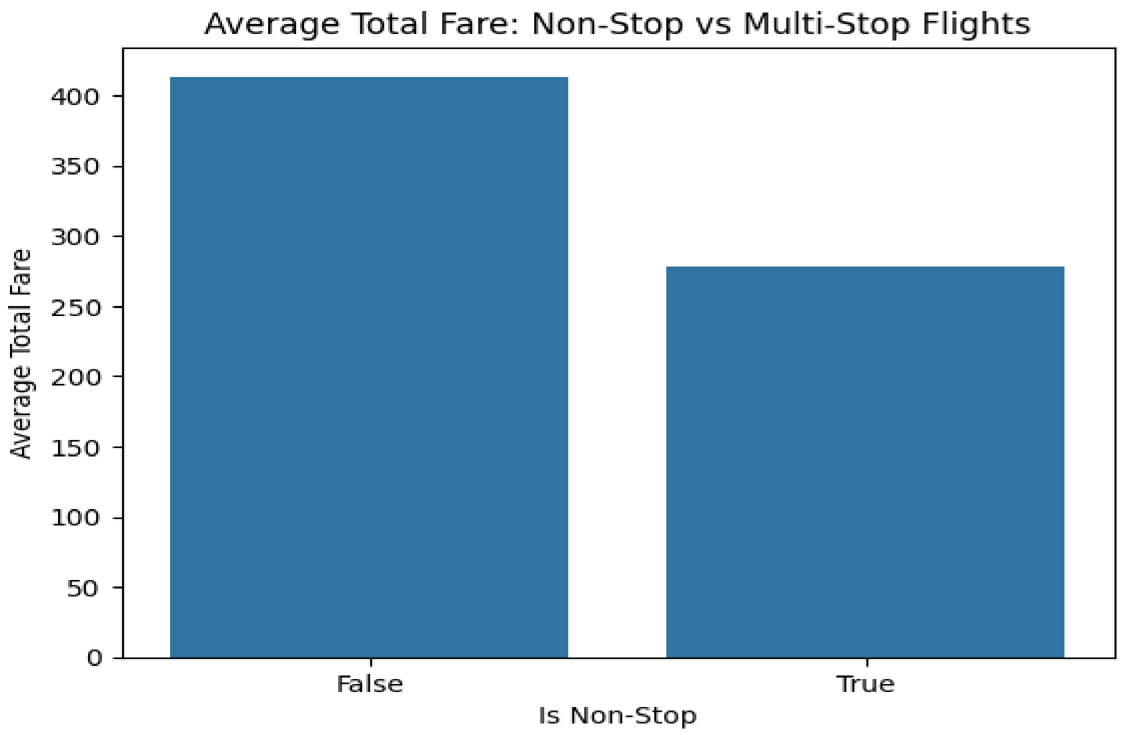
This graph shows us how fares are priced according to the distance and it can be confirmed that as distance increases, flight prices do increase with it.

VI. <u>Top Ten Airports by average fare:</u>



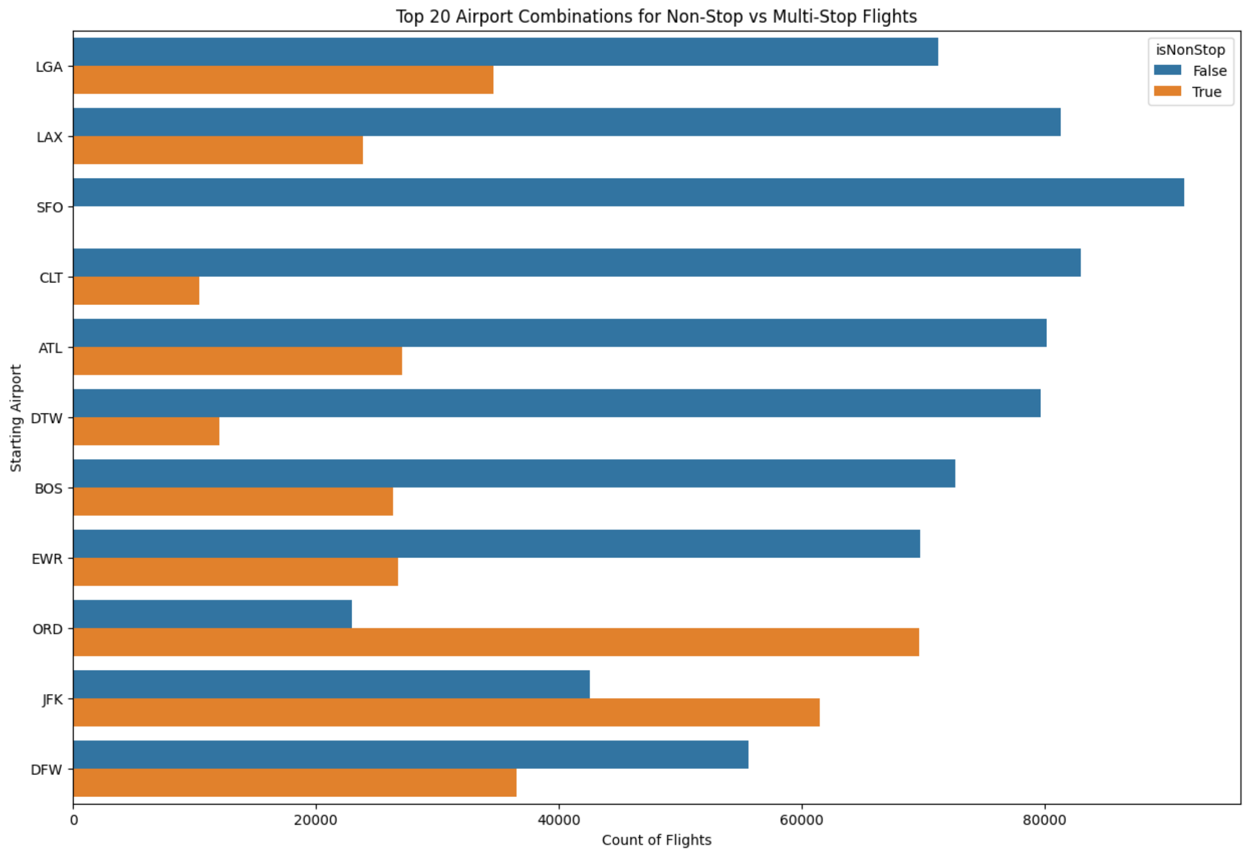
The given graph shows us the estimate of fares when travelling to or fro from these airports.

VII. <u>Average total fare for Non-stop and Multi-stop flights:</u>



The plotted graph shows that flights that are non-stop have lower fares than flights that have multi-stops.

VIII. Top twenty airport combinations for Non-stop vs multi-stop flights:



From the given graph we can make clear inferences that the majority of the flights are multi-stops.

4. Data Preparation

a. Data Cleaning & Transformation

The raw data provided was in different folders named by the starting airport of the flight, wherein there were numerous flight trip details in CSV files. In order to merge all the flight information in one file, we concatenated the data frames after extracting and reading them. We had around 13.6 million trip details and it became extremely difficult to handle this amount of data which lead to cleaning and transforming the dataset.

1. Removed 'legId': The 'legId' column, serving as an identifier for each trip, was removed from the dataset.
2. Downcasted Data Types: Data types within the dataset were optimized by downcasting to Pandas data types. This process was undertaken to conserve memory, enhancing the dataset's efficiency without compromising its integrity.
3. Separated Information by Each Leg of the Trip: The data was restructured to capture information specific to each leg of the trip, and the original columns were dropped. This transformation allows for a more granular analysis, potentially offering valuable insights into individual flight segments.
4. Replaced None Values with 0: Any missing or 'None' values were replaced with zeros, ensuring a standardized representation within the dataset.
5. Downcasted New Columns to Pandas Data Types: Following the creation of new columns, their data types were downcasted to Pandas data types, aligning with the optimization strategy to improve memory usage.
6. Converted 'TravelDuration' into Total Travel Time in Hours: The 'TravelDuration' column was transformed into a new column representing total travel time in hours and the original 'TravelDuration' column was dropped. This conversion allowed for a more unified representation of travel duration, which might better suit analysis and modeling needs.
7. Saved the dataframe in a pickle file: The preprocessed dataframe was saved in a .pkl file format. This choice enables flexibility and accessibility for further feature engineering according to diverse modeling requirements.

b. Data Preprocessing

Model 1 - Tensorflow

1. Separating target variable and features for model training.

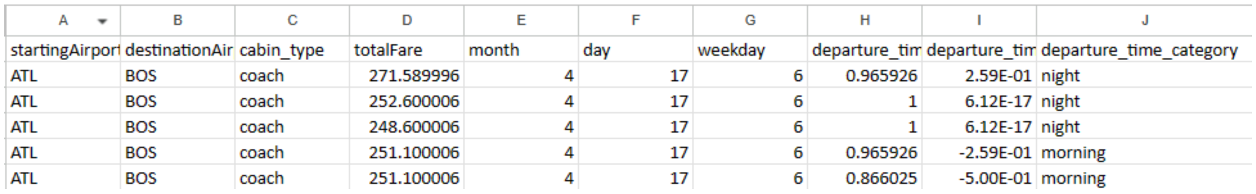
2. Label Encoding Categorical Data: To transform them into numerical representations. This encoding is crucial for machine learning algorithms, as they often require numerical inputs. The label encoder used for this transformation was saved as a joblib file, preserving the encoding scheme for future use.
3. Standard Scaling for Numeric Data: To normalize their values. Standardization ensures that numerical features are on a comparable scale, preventing dominance by features with larger magnitudes. The standard scaler employed for this purpose was saved as a joblib file, preserving the scaling parameters for consistency during predictions.
4. Splitting the dataset into training and validation sets to evaluate the model's performance.

c. Feature Engineering

Model 1 - Tensorflow

1. Converted Flight Date and Departure Time into Pandas DateTime Type
2. Grouped Data for Similar Journeys and Timings to Obtain Average Ticket Prices: This grouping allowed for the calculation of average ticket prices, considering factors like nonstop or refundable flights.
3. Dropped Null Values for 'TotalFare': Null values in the 'totalFare' column were dropped from the dataset which were created from grouping.
4. Flight Date Features Extraction: Features such as weekday, day, and month were extracted, providing temporal insights that may contribute to the understanding of how ticket prices vary over different days and months.
5. Departure Time Features: Features related to departure time were extracted to capture nuances in ticket pricing based on different times of the day. Sin and cosine values for time were computed, introducing cyclic features that account for the periodic nature of time. Additionally, the departure time was binned into categories such as morning, afternoon, etc., to further highlight patterns associated with specific time intervals.

This is an overview of the dataset after performing feature engineering.



A	B	C	D	E	F	G	H	I	J
startingAirpor	destinationAir	cabin_type	totalFare	month	day	weekday	departure_tim	departure_tim	departure_time_category
ATL	BOS	coach	271.589996	4	17	6	0.965926	2.59E-01	night
ATL	BOS	coach	252.600006	4	17	6	1	6.12E-17	night
ATL	BOS	coach	248.600006	4	17	6	1	6.12E-17	night
ATL	BOS	coach	251.100006	4	17	6	0.965926	-2.59E-01	morning
ATL	BOS	coach	251.100006	4	17	6	0.866025	-5.00E-01	morning

5. Modeling

a. Model 1 - Tensorflow

Neural networks were chosen for their capability to capture complex relationships in data, making them suitable for regression tasks like predicting airfare. The flexibility of neural networks allows them to learn intricate patterns present in the dataset.

1. Model Architectures

Several neural network architectures were experimented after the baseline results (Taking the average of total Fare):

1. A simple (trial) architecture with 3 dense layers: 1 input, 1 hidden and 1 output layer with rectified linear unit (ReLU) activation.
2. Added Dropout to 4 Layers Model: An extra hidden layer with regularization was added to increase learning of the model and assess the impact on performance by improving backpropagation.
3. Additional Hidden Layer Model: An extra hidden layer was introduced to explore if increased complexity improves predictive performance.
4. Simplify the Model: Removed 2 hidden layers, to check if model complexity was a problem.

2. Hyperparameter Tuning

Hyperparameters such as learning rate, batch size, and the number of epochs were tuned to find the optimal configuration. The tuning process involved a systematic exploration of hyperparameter spaces using WandB's run analysis.

3. Few Considerations

There were two ways to go about averaging the totalFare and making predictions:
Case 1: Take the average of total fare considering if the flight is nonstop or non-refundable. This would bind the model/app for additional input features by the user. To ensure there's no additional input required, we take predictions for both boolean values (ex: refundable = True and False), and then take an average of the two predictions.
Case 2: Take the average total fare for all flights (no matter if it is nonstop or non-refundable). This would train the model for all types of flight fares and makes more sense to deploy under given constraints.

b. Model 2 - Catboost

The second model employed in our project was the CatBoostRegressor, a powerful and efficient gradient-boosting framework designed for categorical data. We chose this model for its ability to handle categorical features natively. Key hyperparameters were set as follows:

- Iterations: 1000 (The maximum number of trees to be built)
- Learning rate: 0.1 (The step size for updating predictions)
- Depth: 10 (Depth of the tree)
- Loss function: RMSE (Root Mean Square Error, to measure the difference between predicted and actual values)
- Verbosity: True (To output the training progress)

Preprocessing and Feature Engineering:

- A critical preprocessing step was standardizing the numerical features to ensure that each feature contributed proportionally to the final prediction. This was achieved using the StandardScaler as part of a ColumnTransformer. Meanwhile, categorical features were left untouched to leverage CatBoost's native handling of categorical data.
- Feature engineering was focused on extracting meaningful insights from raw data. Notably, we established a hierarchy for cabin classes to quantify the level of comfort or luxury, ranging from 'coach' to 'first class.' We applied a function to determine the highest cabin class based on a ranking system, effectively capturing the premium nature of a flight segment.
- Another feature engineering task involved preprocessing the 'segmentsDistance' column. We split the distance string, replaced 'None' values with '0', and summed up the distances to obtain the total distance for a flight, accounting for possible multi-leg trips.

Training Process and Imbalanced Data Handling:

Training involved fitting the preprocessed data to the CatBoost model, with a clear demarcation of categorical features using their indices. This allowed CatBoost to internally handle categorical features without the need for explicit encoding, which can be advantageous when dealing with a dataset with numerous categorical levels.

We addressed imbalanced data by ensuring our metric of choice, RMSE, naturally accounts for the variance and gives a higher penalty to larger errors, thus diminishing the impact of imbalances in the fare distribution. Additionally, CatBoost has mechanisms to handle imbalanced datasets, which include built-in overfitting prevention features such as random subsampling of the data.

c. Model 3 - Gradient Boosting

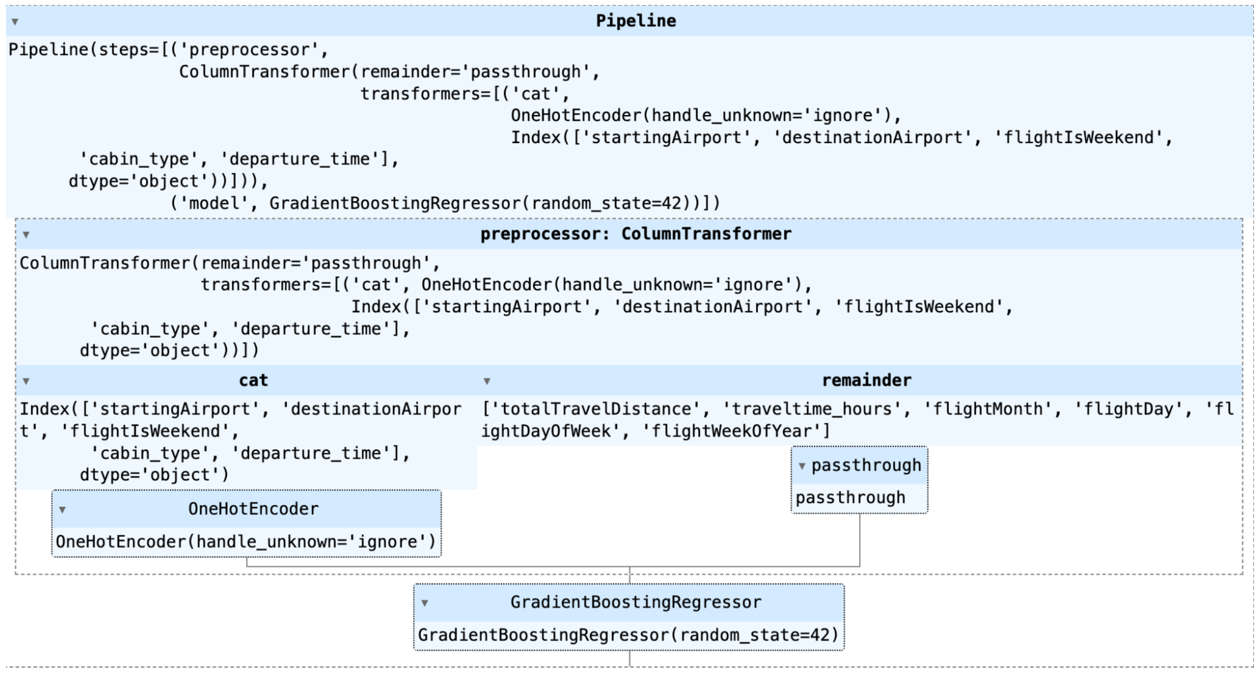
The third model used in our project is Gradient Boost. The gradient boost model is an ensemble machine learning model that builds a predictive model stage-wise. The core principle behind GBM is constructing new base learners that are maximally correlated with the negative gradient of loss function associated with the whole ensemble. Secondly, since the data is quite large, we need to train a mode that is computationally efficient and, at the same time, provides good results. The hyperparameters tuned for Gradient boost regressor are:

1. 'Loss': Optimises the loss function. Since we were doing a regression problem, it was set to 'squared_error'.
2. 'Learning_rate': It is the contribution of each tree, set to 0.1
3. 'N_estimators': The number of boosting stages was set to 100.
4. 'Criterion': To measure the quality of the split, it was set to friedmen_mse for calculating the mean squared error.
5. 'Min_sample_split' and 'min_sample_leaf: They were set to their default values of 2 and 1 respectively.
6. 'Max_depth': The maximum depth of the individual regression estimator was set to 3.
7. 'Sub_sample': To use all the samples, it was set to 1.

Preprocessing and Feature Engineering:

Data preparation: Once the data was correctly formatted and ingested, after thorough analysis, it was observed that some columns had a significant impact on the results. The missing values in the columns were replaced by 0. The rows that had 0 in columns for distance were dropped. After calculating the total distance by summing up each leg, all leg columns were dropped.

Feature Engineering: Converting the departure time column to date time, features like month, day, week and day of the week were calculated. Additionally, a boolean column is_weekend, was also added to check if the flight fares vary during weekdays and weekends. 'Flightweekoftheyear' column was created to understand the impact of seasonal trends. This provides information regarding during which weeks the flight prices peak and dip.



```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('cat',
                                                  OneHotEncoder(handle_unknown='ignore'),
                                                  Index(['startingAirport', 'destinationAirport', 'flightIsWeekend',
       'cabin_type', 'departure_time'],
      dtype='object'))])),
                ('model', GradientBoostingRegressor(random_state=42))])
```

```
ColumnTransformer(remainder='passthrough',
                  transformers=[('cat', OneHotEncoder(handle_unknown='ignore'),
                                 Index(['startingAirport', 'destinationAirport', 'flightIsWeekend',
       'cabin_type', 'departure_time'],
      dtype='object'))])
```

```
Index(['startingAirport', 'destinationAirpor
t', 'flightIsWeekend',
       'cabin_type', 'departure_time'],
      dtype='object')
```

```
['totalTravelDistance', 'traveltime_hours', 'flightMonth', 'flightDay', 'fl
ightDayOfWeek', 'flightWeekOfYear']
```

```
passthrough
```

```
OneHotEncoder(handle_unknown='ignore')
```

```
GradientBoostingRegressor(random_state=42)
```

Modelling: Before modelling the pre-processed data, it was necessary to fit a pipeline, that can handle the categorical variables. Although Gradient Boost can inherently handle categorical values, but encoding can sometimes help the model to interpret each category easily. Additionally, label encoding can tend to increase bias with a larger number of categories dominating the smaller categories. To ensure all categories are represented equally, I preferred using one-hot encoding.

d. Model 4a - XG Boost

XGBoost (eXtreme Gradient Boosting) is a powerful gradient boosting algorithm which is famous for its efficiency and performance. It was chosen due to the ease with which it handles complex relationships amongst various features alongside its superior performance when it comes to handling outliers.

To begin with, an XGBoost was trained with default hyperparameters. This was done to serve as a gradient boost baseline to all of the subsequently trained models.

Following that, another XGBoost Regressor was used whose hyperparameters were tuned using grid search. This was done as a systematic search over a hyperparameter grid can help pin-point the hyperparameter combinations which optimizes the model's performance.

The final XGBoost model was the one which used L1 and L2 regularization techniques. Regularization was introduced to prevent overfitting and to improve the model's overall generalized performance. It should be noted, that this came out to be the best performing model and was chosen only after training and testing two XGBoost algorithms and two LightGBM algorithms.

e. Model 4b - LightGBM

LightGBM (Light Gradient Boosting Machine) is another gradient boosting framework which is known for its speed and adaptability while handling large datasets. LightGBM was used alongside XGBoost to allow for their diversity to show through and understand which algorithm is better for the task at hand.

The first LightGBM model was trained with its default hyperparameters while the second one had its hyperparameters tuned via GridSearchCV. Just like in the case of XGBoost, grid search was performed to find the best hyperparameter combinations.

Implementation Details - Model 4a and 4b

Prior to training any of the XGBoost or LightGBM algorithms, the categorical features were first passed through a label encoder. This was done as both the algorithms require numeric input. Then, post train-test splitting the features were scaled using standard scaler. Also known as z-score normalization, standard scaling was applied to the features to avoid any undue influence of too large or too small values from any of the features on the trained models.

The label encoder and standard scaler were stored using joblib much like the 5 gradient boosting models. This was done so that they could be accessed by the deployed streamlit app.

6. Evaluation

a. Evaluation Metrics

Model 1 - Tensorflow

Specific Business Objective: Mitigating User Discontent

In addition to the primary business objective of providing accurate airfare predictions, there is a specific concern regarding user satisfaction. The aim is not only to minimize prediction errors but also to avoid situations where users might perceive the predicted price as significantly lower than the actual price, leading to dissatisfaction. Users may feel misled and frustrated, impacting their overall experience. To enhance user satisfaction, it is crucial to minimize instances where predicted prices are substantially lower than the true costs.

Reasons to choose specific parameters a/c to business objective:

- **Loss Function:** A loss function that penalizes underestimation more than overestimation - The Mean Squared Logarithmic Error (MSLE) is one such loss function that penalizes underestimation of higher values more than lower values. This ensures that the model is trained to prioritize avoiding substantial price underestimations, aligning with the goal of mitigating user discontent.
- **Optimizer**: An optimizer that balances speed and accuracy, such as Adam (Adaptive Moment Estimation), is commonly a good choice. Adam adapts learning rates for each parameter and combines the advantages of both AdaGrad and RMSProp.
- **Metrics**: For evaluation, while MSE (Mean Squared Error) is a standard metric for regression tasks, in my case, it's crucial to also monitor the Mean Absolute Percentage Error (MAPE) or the Percentage of Absolute Errors (PAE). These metrics show the percentage of the absolute difference between the predicted and actual values and could help ensure that the model doesn't make significant underestimations.

To compare with the baseline results, an additional train and test RMSE score was also observed for each experiment.

Model 2,3 & 4

For these ML models, the metric for evaluation was RMSE (Root Mean Squared Error). RMSE measures the average magnitude of errors between predicted and actual values. The smaller the RMSE value, the better is the model's performance. In the context of the regression task at hand, RMSE was chosen as it could easily quantify how well a model's predicting abilities are.

b. Results and Analysis

Model 1 - Tensorflow

Comparison of all 4 experiments performed (batch size=32, min learning rate=0.001):

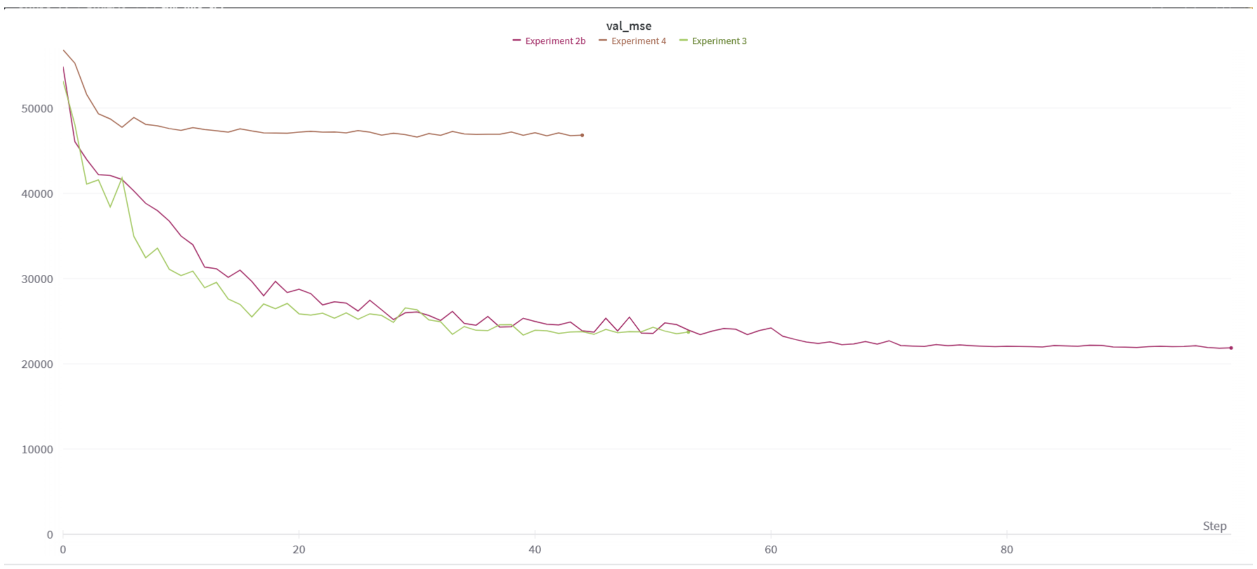
Experiment	Details	Val MAPE	Train RMSE	Test RMSE
1	three layers, no regularization	36.58	199.76857	200.37106
2a	four layers, dropout - 50 epochs	24.02	151.296	152.157
2b	four layers, dropout - 100 epochs	23.46	146.22525	147.725
3	five layers, dropout	24.09	152.09752	154.16371
4	three layers, dropout	41.73	216.26765	216.34686

Full experiment details of each run can be found here: https://wandb.ai/kritika_23/adv_mla_at3

Analysis and Comparison

- Validation MAPE: Models in experiments 2a and 2b with 4 layers and dropout as regularization achieved the lowest validation MAPE, indicating better accuracy in predicting airfare prices.
- Train/Test RMSE: 2a and 2b also performed well in terms of RMSE on both the training and test sets, with lower values compared to other models.

This graph shows validation MSE over each epoch for 3 experiments



The full report can be viewed at: https://api.wandb.ai/links/kritika_23/cjb35rqa

Key Insights

- Impact of Dropout: Introducing dropout layers in Models 2a, 2b, and 3 contributed to improved generalization and performance, preventing overfitting.
- Epoch Sensitivity: Model 2b achieved competitive results with 100 epochs, suggesting that further training helped to enhance predictive performance.
- Layer Complexity: Increasing the number of layers in Model 3 did not yield substantial improvements, highlighting the importance of finding the right balance between model complexity and interpretability.

Implications and Areas for Improvement

- Model Selection: Model 2b, with 4 layers and dropout, trained for 100 epochs, stands out as the best choice, considering both performance and training efficiency.
- Regularization Effectiveness: Regularization techniques explored in Experiments resulted in improved performance. Further investigation into different regularization strategies may be considered.
- Hyperparameter Tuning: Fine-tuning hyperparameters such as learning rates and batch sizes may provide additional insights and potential performance gains.

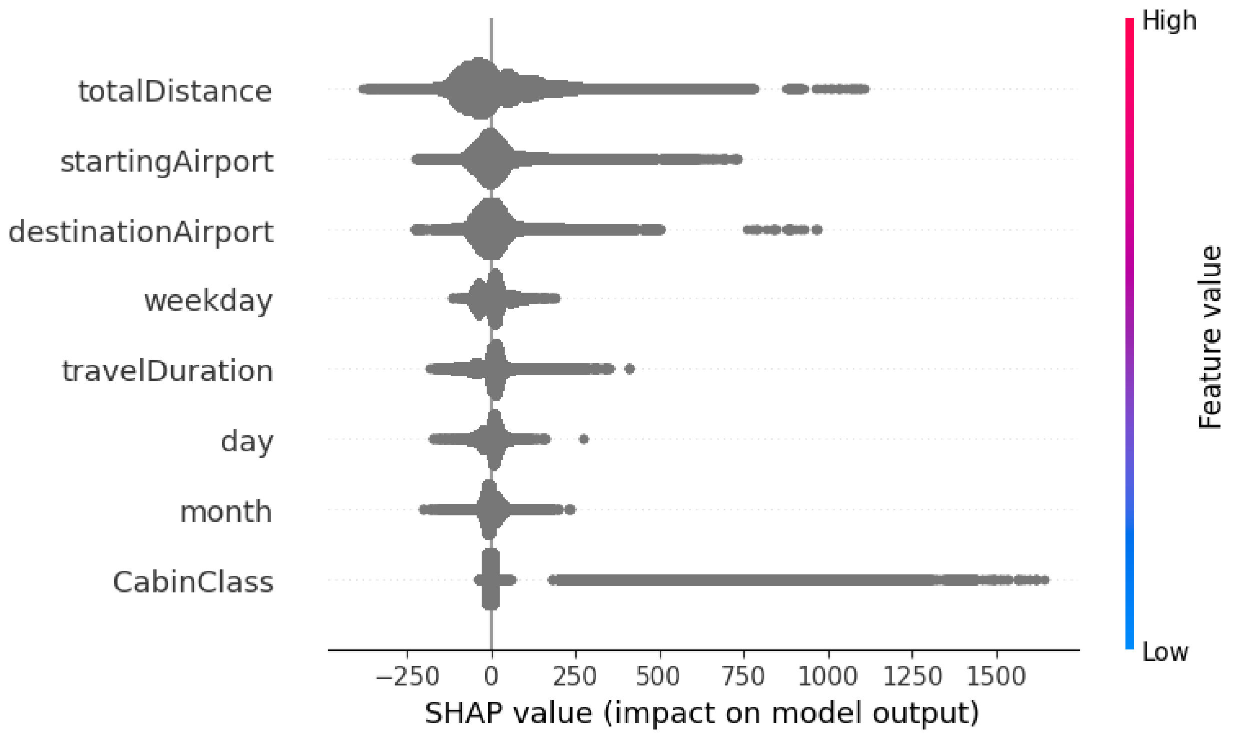
Model 2 - CatBoost Regressor

The CatBoost model's performance on the test dataset is summarized by the following metrics:

- RMSE: 103.73
- MAE: 69.80
- R2: 0.75

The RMSE and MAE indicate that, on average, the model's fare predictions are within approximately $103 and $70 of the actual fares, respectively. An R-squared value of 0.75 suggests that 75% of the variability in flight fares can be explained by the model, which is a substantial figure in the unpredictable domain of fare prices.

Feature Importance and SHAP Analysis:



The model's feature importance analysis revealed that **'totalDistance'** is the most significant predictor, followed by the 'startingAirport' and 'destinationAirport'. This aligns with intuitive understanding as longer distances generally correlate with higher fares, and specific airports have varying levels of taxes and operational costs affecting fare prices.

A SHAP summary plot further provided a granular view of feature impacts:

- Higher 'totalDistance' values consistently increased fare predictions.
- The specific 'startingAirport' and 'destinationAirport' have a complex influence, affecting fare predictions in both positive and negative directions.
- Temporal features like 'weekday', 'travelDuration', 'day', and 'month' demonstrated a more nuanced effect, indicating that while time-related aspects are relevant, they don't dominate the model's predictions.
- Interestingly, 'CabinClass' exhibited a relatively lower impact, suggesting that while service level matters, other factors might be more critical in determining fare prices.

These insights could inform potential improvements to the model, such as further feature engineering around airports and temporal data or exploring non-linear relationships between features and fare prices. The analysis also underscores the significance of distance as a predictor, which could be a focal point for future enhancements to the fare estimation algorithm.

Model 3: Gradient Boost

The model was evaluated based on the MSE, RMSE and MAE scores. A baseline model was also created to compare the score differences between the models.

Model Performance Scores:

Baseline:

1. MSE: 44,166.24
2. RMSE: 210.15
3. MAE: 156.83

Gradient Boost:

1. MSE: 19159.73
2. RMSE: 138.41
3. MAE: 95.99

Model Performance Metrics:

- Mean Squared Error (MSE): The average squared difference between the estimated values and the actual value. It gives a general idea of the error magnitude but is sensitive to outliers since the errors are squared.
- Root Mean Squared Error (RMSE): The square root of the MSE. It's in the same units as the target variable, making it more interpretable.
- Mean Absolute Error (MAE): The average absolute difference between the predicted values and the actual values. It provides a linear score, meaning all individual differences are weighted equally in the average.

Model Result Analysis:

- Comparison to Baseline: Both the MSE and RMSE are lower for the Gradient Boosting Model compared to the Baseline Model, which means that the Gradient Boosting Model has a better fit to the data and makes more accurate predictions. The MAE is also lower, indicating that, on average, the Gradient Boosting Model's predictions are closer to the actual values.
- Relative Improvement: The Gradient Boosting Model has reduced the MSE to almost half of what the Baseline Model's MSE is, and similar improvement is seen in RMSE and MAE. This is a substantial improvement in predictive performance.
- Error Magnitude: The RMSE of the Gradient Boosting Model is 138.42, which means that, on average, the model's predictions are off by this amount from the actual fare prices. This error may or may not be acceptable. For high-cost items (like expensive flights), this might be a reasonable error. For low-cost items, this might be too high.
- Model Selection: Given that the Gradient Boosting Model significantly outperforms the Baseline Model in all three metrics, it is likely a better model for the given data and can be selected for further optimisation and testing.
- In summary, the Gradient Boosting Model shows a significant improvement over the Baseline Model, indicating that it is learning from the features provided and is able to make more accurate predictions. Further model tuning, feature engineering, and possibly doing GridSearchCV or model stacking could lead to even better results. It is also important to note that the dataset is huge, and training models take a significant amount of time hence further tuning must be done where computational powers are high or the modelling is done in a small subset of data.

Model 4: XGBoost and LightGBM

Comparison of all experiments for Model 4

Experiment	RMSE Training	RMSE Testing
Exp 1: XGBoost with default args	131.593	136.422
Exp 2: XGBoost with Tuned Hyperparameters	141.409	143.483
Exp 3: LightGBM with default args	148.228	149.057
Exp 4: LightGBM with Tuned Hyperparameters	141.926	143.680
Exp 5: XGBoost with regularization	131.592	136.421

c. Business Impact and Benefits

Impact and Benefits:

1. Optimising Pricing Strategy: The given models can help the airlines and travel agencies understand the nature of their customers. They can add new customer management schemes and offers that help to attract new customers and, at the same time, maximise profits from the existing customers.
2. Customer Experience: Given the nature of the dataset, they can understand the challenges that customers are facing, like some routes have high customer counts, but due to the low frequency of planes, people choose a different mode of transport. They can then quickly add more planes for the given route to attract more customers.
3. Inventory Management: By understanding the fare trends across the seasons, the airlines can priorly predict the demands of seats, and they can adjust their inventory accordingly. This can prevent underbooking and overbooking.
4. Revenue Management: The airlines can better understand how their fair is getting distributed. One such way would be to sell the right seat to the right customers at the right time and for the right price to maximise profitability.

Challenges and Opportunities:

1. Dynamic Pricing: The models can be used to implement dynamic pricing, where the fare is adjusted in real-time according to the demand, competitor pricing, seasonal trends and other factors.
2. Demand Forecasting: By predicting the fares, the models can indirectly help forecast demands. The airlines can leverage this for marketing, route planning and operational adjustments. It is also possible that forecasts can be wrongly predicted, thereby causing losses of millions. Hence, before deploying the models in the real world, a thorough analysis covering all the aspects needs to be done.

Potential Negative Business Impacts:

1. Over Reliance on model predictions: It is important for the businesses to not rely too heavily on the predictions without understanding the limitations, it is possible that businesses may make poor strategic decisions based on flawed predictions.
2. Customer Trust: Incorrect predictions can directly lead to customer dissatisfaction by making them feel they are not getting the best prices.
3. Market Volatility: Dynamic pricing leads to sudden price changes over a short period of time. This can lead to customers getting irritated and not coming back again.

4. Regulatory Checks: If it is found that the model leads to price discrimination or unfair practices, it could come under regulatory scrutiny.

Recommendations to the Business:

1. Continuous model monitoring: If the model is not trained with the latest data, it may tend to produce flawed results and hence, the model should be regularly trained with the latest data to maintain accuracy over time.
2. Diverse Data Sources: Since the current data used is from limited sources, it is possible that the data may have some majority and minority class bias. Hence using data from diverse sources, negates this possibility, making it more fair and unbiased.
3. Scenario Planning: Using different planning scenarios, understand the potential impacts of different pricing strategies under various market conditions.
4. Customer Feedback Loop: Implement a technique in which the customer feedback is gathered on pricing and using that feedback to refine the pricing.

d. Data Privacy and Ethical Concerns

Data Privacy Implications: From the given data, it is possible that some personal information trends can be identified like; an individual's travel patterns and travel preferences. It could also reveal information regarding an individual's movements (recent location) and financial status. Hence, it is important that the data is collected from a trusted source to mitigate the possibility of data privacy.

Ethical considerations:

1. Bias in data: It is possible that while training the historical data, certain routes were more expensive due to different economic factors, which could lead to discriminatory pricing.
2. Impact of pricing: It is also important to consider that the trained models do not inaccurately predict prices that are unfairly disadvantageous to certain groups of flyers.

Steps taken to ensure data privacy:

Firstly, all the columns that provided any sensitive or personal information were dropped. The data provided and collected was from a trusted source. It was made sure that each team member handled the data securely and safely. A proactive check for biases during data modeling was done to ensure fair predictions. Only the features which help in predicting the fare price were used. Transparent communication was done on how the data was stored, handled and used.

By addressing these issues, it was made sure that the project follows all the data privacy and ethical guidelines for predicting flight fare prices.

7. Deployment

a. Model Serving

The deployment of our suite of predictive models represents a multifaceted approach to estimating flight fares. Each model—Neural Network using TensorFlow, CatBoost, GradientBoost, and XGBoost—brings unique strengths and required specific considerations for integration into our Streamlit application.

Neural Network (TensorFlow):

- Serialization**:** The TensorFlow model was saved using the HDF5 file format, which is ideal for storing large numerical models.
- Integration**:** The model's deployment was facilitated by TensorFlow's serving capabilities, which are well-suited for production environments.
- Scalability**:** Due to TensorFlow's broad ecosystem, we ensured that the model was scalable and could handle high request volumes efficiently.

CatBoost:

- Performance**:** CatBoost's handling of categorical features significantly reduced preprocessing needs, leading to a leaner pipeline.
- Efficiency**:** The model is inherently efficient with resource utilization, translating to faster predictions and lower operational costs.

GradientBoost:

- Adaptability**:** This model's adaptability to various data types allowed us to fine-tune the performance on our specific dataset.
- Deployment**:** Like CatBoost, we used 'joblib' for serialization, which streamlined the deployment process.

XGBoost:

- Accuracy**:** Known for its performance, XGBoost was chosen for its ability to drive precise predictions.
- Portability**:** The model's compatibility with numerous platforms ensured that deployment could be adapted to different environments if necessary.

Considerations:

Deploying a diverse set of models required us to adopt a modular and consistent approach to preprocessing, serialization, and prediction serving. We ensured that:

- Data provided for training was only from 2022-04-17 to 2022-07-17. We tried to incorporate the events from the calendar, but 4 months of data will not show any repetitions of events/holidays that the model will learn. Can add this feature when we have more historical data.
- All models were subjected to a rigorous preprocessing pipeline that standardized inputs and handled categorical variables effectively.
- Serialization formats were chosen based on the compatibility and efficiency of each model type.
- The web app was designed to load and serve predictions from any model dynamically, allowing for user selection and comparison.

Challenges:

- One of the primary challenges across all models was maintaining uniformity in the data preprocessing steps. Each model required inputs in a slightly different format, necessitating a flexible and robust preprocessing pipeline.
- During the deployment of our machine learning models to the Streamlit, we encountered several challenges related to library compatibility. One of the issues we faced was related to version conflict between the TensorFlow I/O GCS Filesystem package and our Python environment. Our project required a version of the TensorFlow I/O package that was compatible with the Python version in use. This issue was emblematic of a common problem in machine learning deployments: ensuring that all components of the project's stack are mutually compatible. The specificity of version requirements for TensorFlow, coupled with the constraints of the hosting environment, necessitated a careful balancing act.

Future Improvements

For future deployment endeavors, it is recommended to:

- Pull calendar events and merge them into training data, which will help the model learn the increased ticket prices during the public holidays or can even have a seasonal trend.
- Establish a more automated CI/CD pipeline for each model to streamline updates and version control.

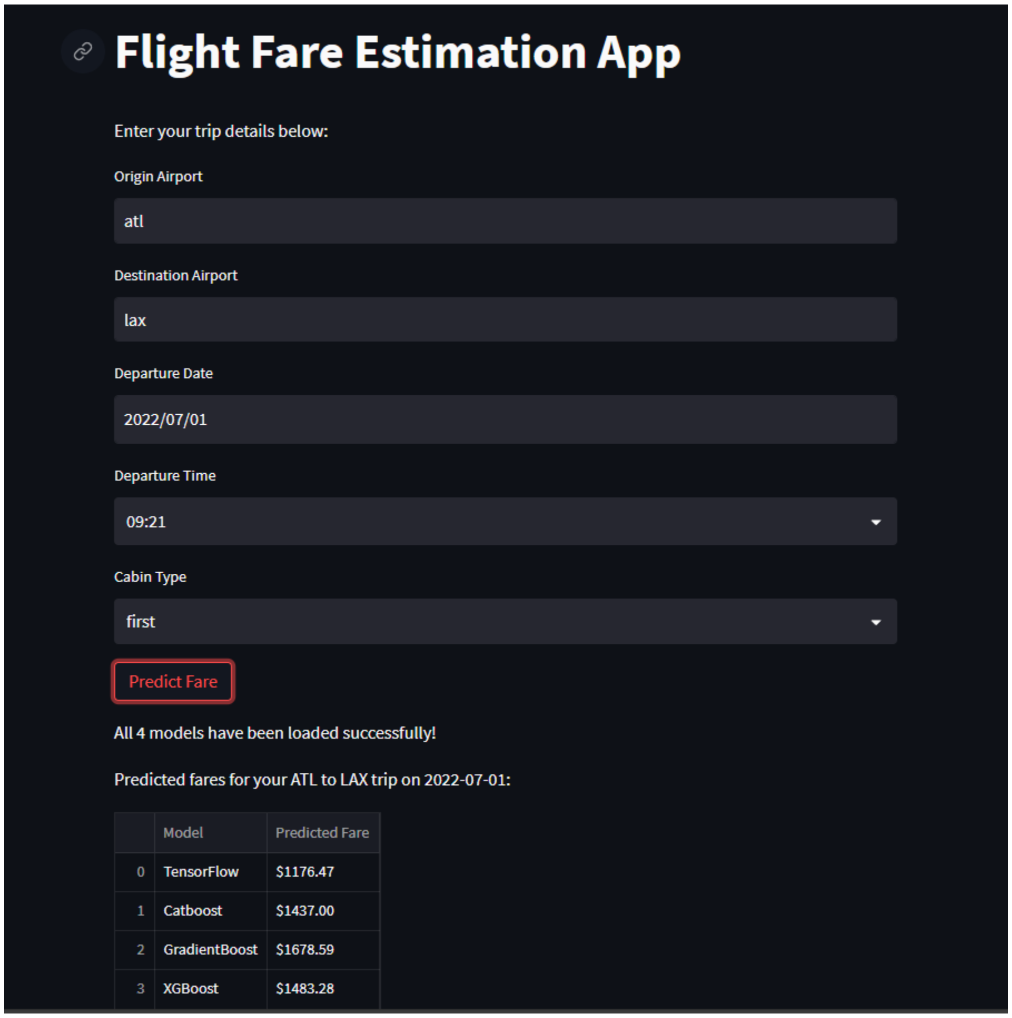
- Investigate the use of microservices architecture to isolate model dependencies and serve them independently.
- Consider the integration of MLOps practices to monitor model performance over time and retrain as needed.

b. Web App

Purpose and Functionality

The 'Flight Fare Estimation App' is designed to provide users with an instant estimate of airfares for domestic flights within the USA. By inputting basic trip details such as the origin airport, destination, departure date and time, and preferred cabin class, users can receive fare predictions from four distinct machine learning models. This facilitates a quick comparison to guide decision-making on travel expenses.

Instructions for Use



Flight Fare Estimation App

Enter your trip details below:

Origin Airport: atl

Destination Airport: lax

Departure Date: 2022/07/01

Departure Time: 09:21

Cabin Type: first

All 4 models have been loaded successfully!

Predicted fares for your ATL to LAX trip on 2022-07-01:

	Model	Predicted Fare
0	TensorFlow	$1176.47
1	Catboost	$1437.00
2	GradientBoost	$1678.59
3	XGBoost	$1483.28

To set up the app, users simply access the URL - https://advmlaat3.streamlit.app/, with no additional installation required.

The Flight Fare Estimation App is designed to provide users with an easy-to-use interface for estimating airfares. By inputting simple trip details, users can receive fare predictions powered by the team's sophisticated models.

Use Cases and Commercial Potential

The app caters to a wide array of users, including casual travelers, business professionals, and travel planners. By offering a reliable estimate of flight costs, the app enables better budgeting and decision-making for trip planning. Its commercial potential lies in its ability to be integrated with travel booking platforms, offering added value to customers and generating insights for the travel industry.

Limitations and Improvements

1. While the app currently fulfills its primary function effectively, there are limitations. For example, the travelDuration and totalDistance are estimated using a simplified model, which does not account for real-time variables such as weather or air traffic. Future improvements could include live data integration to enhance prediction accuracy.
2. Moreover, the app's reliance on third-party APIs, such as Airport Info API from RapidAPI (Airport info API Documentation (Active-api) | RapidAPI) for flight details, introduces a dependency that could affect performance and uptime. Exploring alternative data sources or developing proprietary data-gathering capabilities could be considered for long-term sustainability.

8. Collaboration

a. Individual Contributions

Student	Contribution
Anika	1. Explored and Built XGBoost and LightGBM model - Chauhan_Anika-14188775-xgboost.ipynb 2. Created a prediction script for best XGBoost model - model_xgb.py 3. Kept a track of meeting minutes to ensure clear task distribution and track progress 4. Report - Business Understanding, Conclusion
Kritika	1. Created and maintained Github, Google Drive, Poetry installation, and project requirements. 2. Data preprocessing, EDA - Data processing & EDA.ipynb Merged, cleaned and transformed data that was used for training by everyone. 3. Built tensorflow model - dhawale_kritika-24587661_neural_nets.ipynb 4. Built Streamlit app - streamlit_app.py (Created custom functions) 5. Created a prediction script for tf model - model_tf.py 6. Deployment on Streamlit by linking GitHub repository - Solved deployment issues for each team member. 7. Report - Data Preparation, Modeling tf, Evaluation, Collaboration
Sahil	1. EDA - Performed exploratory analysis of raw data. 2. Built Cat Boost model - kotak_sahil_24707592_catboost.ipynb 3. Created a prediction script for CatBoost model - model_catboost.py 4. Report - Executive Summary, Deployment, Modelling and Evaluation. 5. Api Integration with Streamlit.
Varun	1. EDA - Performed exploratory analysis on merged data. 2. Built Gradient Boost model - chhetri_varun_24711703_gradient_boost.ipynb 3. Created a prediction script for Gradient boost model - model_gb.py 4. Report - Data Understanding, EDA, Modelling and Evaluation.

b. Group Dynamic

Our primary channel for communication was a WhatsApp group. We used to regularly update on our progress on the group chat. Team meetings were held on Zoom and meetings were taken. Tasks were assigned according to each member's capabilities. GitHub was set up to track technical aspects of the project (version controlling for code) while Google Drive was used as a data storage bucket.

c. Issues Faced

The team faced a significant challenge related to time management as members juggled multiple assignments, leading to a last-minute crunch. The team emphasized the importance of early communication about potential time conflicts and that this led to stressful situations and overload on one of the team members.

The experience highlighted the critical need for effective time management strategies and improved communication within the team. Moving forward, it is recommended to implement project management tools that emphasize task timelines and allocate dedicated time for regular progress updates. Moreover, fostering a culture of transparency and open communication about workload and potential challenges can prevent last-minute bottlenecks, ultimately contributing to smoother collaboration and successful project outcomes.

9. Conclusion

The successful deployment of the app indicates a significant step towards achieving the primary business objective - accurate airfare predictions. The models ranging from gradient boosting to neural networks, helped immensely when it came to generalizing. The objective to accurately predict price aligned well with these models' capabilities. Neural Networks were a good choice due to their sensitivity to intricate patterns. This intricacy complemented the efficiency brought in by the ensemble approach of the gradient boosting frameworks.

Moving forward, we can look into deriving data from diverse sources to mitigate bias. A customer feedback loop can also be set up to refine the pricing model so that its ability to align with the users' expectations is enhanced and user dissatisfaction is kept at bay. For tweaking the deployed model further, seasonality can be an area worth exploring. Incorporating calendar events can help enhance the model's understanding of the pricing dynamics.

As the project moves forward, these recommendations will serve as a foundation for maintaining relevance in the dynamic domain of airfare while subsequently enhancing user experience.

10. References

[1] McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

[2] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

[3] Waskom, M., Botvinnik, Olga, O&#x27;Kane, Drew, Hobson, Paul, Lukauskas, Saulius, Gemperline, David C, … Qalieh, Adel. (2017). mwaskom/seaborn: v0.8.1 (September 2017). Zenodo. https://doi.org/10.5281/zenodo.883859

[4] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science &amp; Engineering, 9(3), 90–95.

[5] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[6] Van Rossum, G. (2020). The Python Library Reference, release 3.8.2. Python Software Foundation.

[7] Joblib Development Team (2023) Joblib/joblib: Computing with python functions., GitHub. Available at: https://github.com/joblib/joblib.

[8] Abadi, Mart&#x27;in, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … others. (2016). Tensorflow: A system for large-scale machine learning. In 12th $USENIX$ Symposium on Operating Systems Design and Implementation ($OSDI$ 16) (pp. 265–283).