Assignment 3

ML Data Product

10-Nov-2023

Group 10

Student Last Name	Student First Name	Student ID	Group Allocation
Britto	Rohan	24610990	Student A
Khatri	Smit	24712248	Student B
Mohiuddin	Mahjabeen	24610507	Student C
Murgudkar	Archit	14190286	Student D

Github	Project Repo: https://github.com/rohanbrit/adv_mla_asgn3 Streamlit App Repo: https://github.com/rohanbrit/adv_mla_asgn3_streamlit

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

Table of Contents

1. Executive Summary

Project Overview:

This project's primary goal was to develop a data product that caters to the unique needs of travelers in the USA by providing them with a tool to estimate local airfare accurately. To achieve this, we created a Streamlit app, allowing users to input specific travel details. These inputs, including origin and destination airports, departure date, departure time, and cabin class, are processed by our predictive model to provide users with precise flight fare predictions.

Problem Statement and Context:

The problem at hand revolves around the challenge travelers face when trying to estimate airfare costs for local trips. The aviation industry's dynamic pricing strategies, coupled with a wide array of ticket types and options, often leave travelers perplexed and struggling to accurately budget their travel expenses. Considering this, our project sought to address this issue by harnessing a substantial dataset of historical flight information to create a predictive model capable of estimating local airfares.

Achieved Outcomes and Results:

A substantial dataset of over 13.5 million records from diverse airports was successfully consolidated and cleaned during this project. Notably, the project retained certain outliers in the fare data, which often represent premium ticket prices. This decision was made to reflect the authenticity of these fares, ensuring that all possibilities were covered. Moreover, the project enhanced the dataset by extracting key temporal information from the flight date, including the flight year, month, day, and week. Additionally, the time gap between the date of the user's search and the flight date, a known influential factor in fare pricing, was incorporated as a feature in the dataset.

The project's culmination is the creation of a Streamlit app that empowers users to input their specific travel details. The app, equipped with a predictive model, processes these inputs, and returns accurate airfare predictions.

2. Business Understanding

a. Business Use Cases

We worked on 4 different business use cases, i.e., 1 per group member. For the origin airport, destination airport, departure date, departure time and cabin type selected, our models would predict:

- The average fare. (Developed by Smit Khatri)
- The fare specific to airlines that run on the given route at the given hour. (Developed by Rohan Britto)
- The fare for a flight with no layover and 1-3 layovers. (Developed by Mahjabeen Mohiuddin)
- The fare for a refundable ticket vs a non-refundable one. (Developed by Archit Murgudkar)

This would give the customer a good understanding of the total price of a ticket for various options and would allow him/her to make an informed choice.

b. Key Objectives

- The key objectives of this project would be to provide a holistic view of the options available along with the price for each of the options.
- The key stakeholders would involve airline companies and travel agencies.
- These models would allow airline companies to predict the average fare for multiple options and competitor airline companies to provide a better price range to customers, thus improving their sales and profits.
- Currently, most, if not all travel agencies depend on airline company's data to provide flight costs. Some companies do not release this data well in advance. While booking the flight directly based on predicted costs might not be feasible, stakeholders could leverage these models to provide customers with a price range based on predictions to give users better insights.

3. Data Understanding

The dataset used in this project for estimating local travel airfare consists of flight-related information. The data dictionary can be found in the Appendix section which provides a detailed explanation of each of the features in the dataset.

An initial analysis of the dataset revealed the following:

The dataset is extensive, with over 13 million rows and 23 columns. Most columns are in object (string) format, including date and time-related columns. Missing data was observed in the totalTravelDistance and segmentsEquipmentDescription columns. An outlier check revealed a small number of records with total fares exceeding $4000, which are likely related to first-class fares or last-minute bookings.

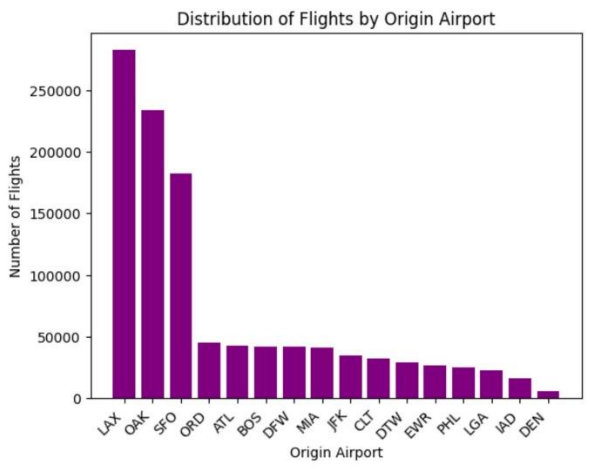

Fig. 1: Origin airport bar chart

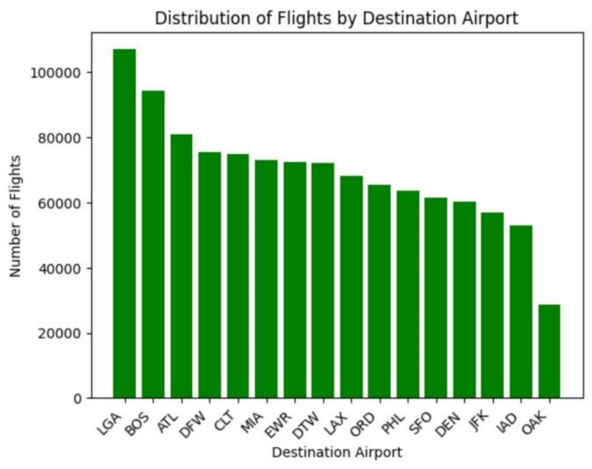

Fig. 2: Destination airport bar chart

The above bar charts gives evidence that the Origin Airport "LAX" has more than 250000 number of flights and the destination airport "LGA" has more number of flights with the count of 110000.

4. Data Preparation

The dataset provided consisted of 16 folders with multiple zip files in each of them.

Data Extraction: The data appears to have been bulk-extracted from these sources. The zip files in each of the subfolders, such as ATL, BOS, etc., were extracted using applications like 7zip or Winrar.

Combining Data: The data from all the directories was combined into a single dataset. This involved reading individual CSV files from each subfolder and concatenating them to create a unified dataset.

Approach 1: The average fare

- Before feeding the data into the model, it's crucial to prepare and preprocess it appropriately. This process is often just as important as the choice of the machine learning algorithm itself. The dataset included a mix of categorical and numerical features, making it necessary to employ a data preprocessing pipeline.
- The pipeline was set up to include data preprocessing steps, with a specific focus on handling categorical features. Categorical features, such as the starting and destination airports and cabin codes, needed to be transformed into a numerical format that the model could work with. To achieve this, one-hot encoding was employed, allowing each category within a categorical feature to be represented as a binary (0 or 1) value.

Approach 2: The fare specific to airlines that run on the given route at the given hour

- For this model, departure hour was used instead of the exact time to reduce the chances of overfitting.
- Apart from this, it was important to fetch the airline names from segmentsAirlineName column. I created a function to split all airlines in segmentsAirlineName and convert it to columns. For e.g. [Delta, Delta] would be converted to 2 in Delta column and [Delta, United] would be converted to 1 in Delta and United each.
- Cabin codes have a hierarchy or order, i.e., coach is the least expensive cabin type and first is the most expensive. Hence, I have ranked them accordingly. I noticed that there are some cases where the traveler has searched for separate travel coaches in a layover flight. This would affect the prices of the flights and it was important to handle them. I created a function to calculate the cabin weight. As per my logic, [coach, coach] will be

converted to (1+1)/2, which is like a single coach flight, but if the cabin code is [first, coach], it will calculate it as (4+1)/2, which will inform the system that the price difference might be due to a higher cabin code.

- Flight timetable with origin airport, destination airport, departure hour, and airline names were stored in a separate file for making predictions.
- As we were dealing with time series data, we had to perform the split based on the date, i.e., older data was used for training and validation, and most recent data was supposed to be used for testing. I decided to use data from flight weeks 15-23 (approximately 71%) as training data, weeks 24-25 (approximately 18%) as validation data and weeks 26-28 (approximately 11%) as testing data.
- Standard scaling was performed on numerical data and Target encoding was used on remaining categorical data using Pipelines.

Approach 3: The fare for a flight with no layover and 1-3 layovers

- The approach used for data preparation is based on the US residents' perspectives and what key factors they would focus on while booking airline tickets to minimise the fare amount.
- The customers who are more concerned with cutting the cost of travel and saving money will mostly prefer more layover flights, which will be cheaper than one layover.

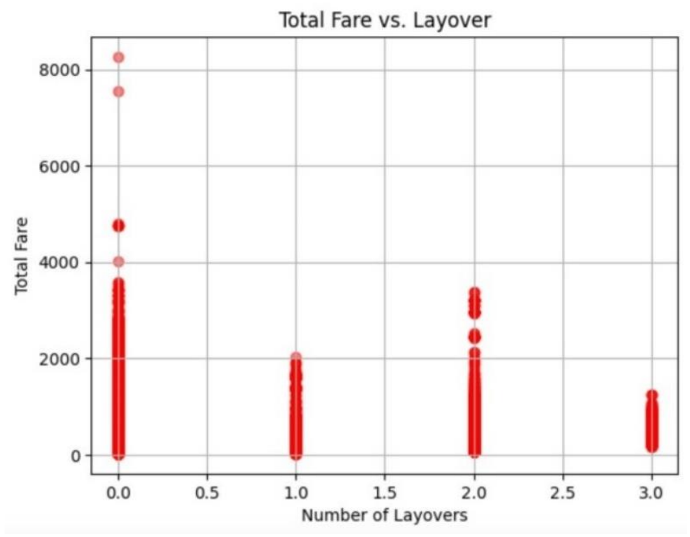

Fig. 3: Fare of flight based on number of layovers

The scatter plot shows that with 0 layover the fare price is high.

- Another scenario could be that if it is a long journey, the customer must take a flight that includes two to three layovers between the origin and destination airports.
- After merging various files to form a dataset, the columns such as startingAirport, destinationAirport, isNonStops, totalFare, segmentDepartureRawTime, segmentCabinCode, segmentsAirlineCode, flightdate were considered, and the rest of the columns were ignored. There are 13519999 observations in the dataset. There were some duplicate legIds present, and they were dropped from the dataset.
- The column **segmentsAirlineCode** contains the connecting flight code that helps in analyzing the **number of layovers** that the customer will come across during their travels from the origin airport to the destination airport.
- The data in the column name flight date, which contains the information about the flight, is split into Departure_Hours and departure minutes using the split function.
- The dataset is split into train, validation, and test sets in the ratios of 60%, 20%, and 20% to analyse the performance of the model. Initially, the model is trained with 60% of the data, and therefore the model has learned all the patterns from the previous data. 20% of the new data is assigned to the validation set to make predictions on unseen data. The remaining 20% of data is used to evaluate the model.
- The standard scaler is inserted into the stage of a pipeline to scale the numerical data, and the transformer encoder is inserted, on the other hand, to pass the object data types.
- The numerical columns and categorical columns are ingested into the columnTransformer. The preprocessor and model are added to the pipeline.

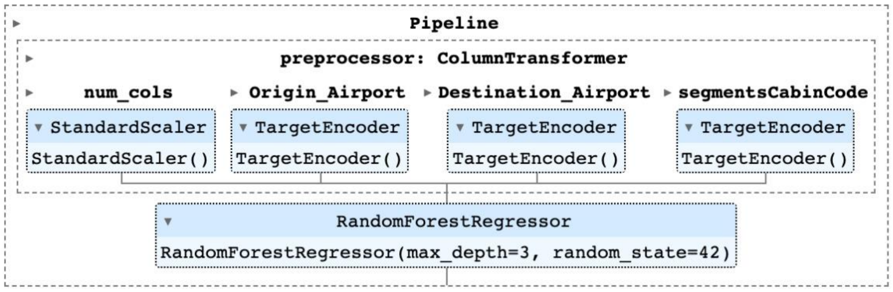

Fig. 4: Pipeline Process

Approach 4: The fare for a refundable ticket vs a non-refundable one

- Users based in the USA will be able to get the flight ticket price for both cases, refundable ticket price and non-refundable ticket price. This will better help users in decision making for which type of flight to board based on their personal interests and situations and the flight prices in both the cases.

- The crucial feature to make predictions in this specific use case is 'isRefundable' which i converted from boolean to int, indicating refundable as 1 and non-refundable as 0. It was surprising to see how few people had booked for refundable flight tickets as the number was just 191 out of total bookings of 13519999.

- I split data into training, validation and testing. As we have a lot of data to build a machine learning model, I decided to split the data into training and test in 7:3 ratio and further the training data divided into training and validation in 4:1 ratio.
- I decided to build a pipeline using the 'Pipeline' library from sklearn. I created a list of numerical and categorical columns which will be used in predicting the target feature. By creating a numeric transformer named num_transformer, I scaled all the numeric values to the same scale. Scaling numeric values to the same scale brings them into one range and thus the model performs with greater accuracy.

5. Modeling

Approach 1: The average fare

- The Decision Tree Regressor was chosen as the primary algorithm due to its versatility and ability to capture complex data relationships, making it an ideal choice for predicting airline fares.
- One of the strengths of decision trees is their ability to handle both categorical and numerical features, making them well-suited for the diverse dataset commonly found in airline fare prediction models. To implement the Decision Tree Regressor, the scikit-learn library, a powerful and popular machine learning tool, was utilized.
- The Decision Tree Regressor was chosen for several reasons. Firstly, it offers transparency and interpretability, which is crucial for understanding how the model makes predictions. Secondly, decision trees can capture complex interactions and non-linear relationships within the data. This ability is vital in predicting airline fares since prices are influenced by various factors, and these relationships are often non-linear.

Approach 2: The fare specific to airlines that run on the given route at the given hour

- For the model that predicted flight ticket costs of various airlines, we built an airline timetable to find the unique flights that run between the selected airports at the selected hour. This information was stored in a csv file for making predictions. For this model, we used various algorithms like linear regression, random forest, and gradient boosting. They were compared against each other and, also against a dummy baseline model that predicted the mean value of the fare.
- The initial step involved implementing a dummy model that predicted the mean fare for all instances. This simple algorithm serves as a baseline for performance comparison with more sophisticated models. The dummy model requires minimal preprocessing, as its purpose is to establish a straightforward benchmark.
- Three advanced algorithms were employed: linear regression, random forest, and gradient boosting. For linear regression, no specific hyperparameters were altered. In contrast, hyperparameters such as n_estimators, max_depth, and min_samples_leaf was adjusted for random forest and gradient boosting to enhance performance, address overfitting, and manage computational resources.

Approach 3: The fare for a flight with no layover and 1-3 layovers

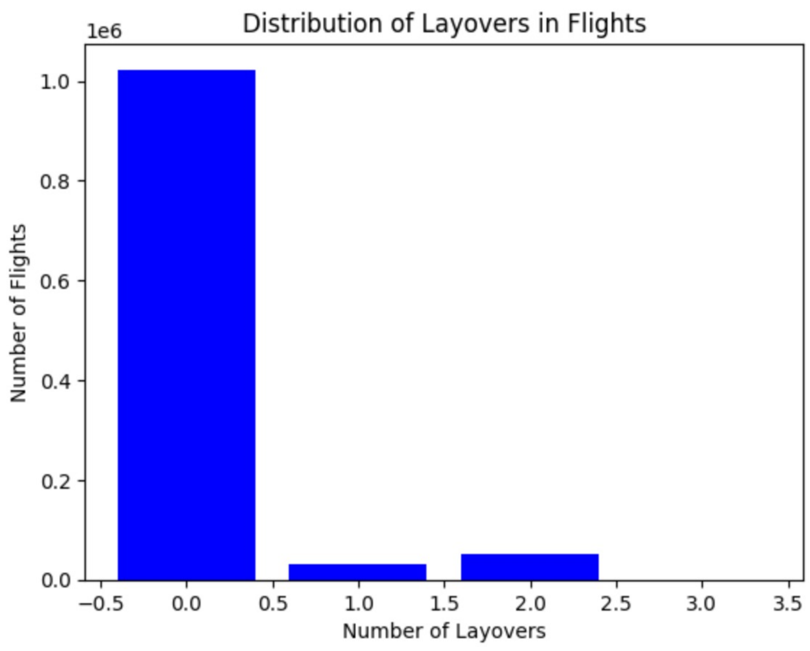

Fig. 5: Insights into with one layover count of flights

The bar chart shows that there are more number of flights with one layover

- The Airlines dataset has a huge number of observations, and this can be easily dealt with supervised machine learning algorithms such as Neural Network, and Forest forest regressions.
- **Model 1: The Random Forest Regressor** algorithm has gained popularity for dealing with large, complex datasets and its user-friendly nature. It also avoids overfitting of data to the maximum extent. Each tree is created from different samples of rows, and at each node, a different set of samples of features is selected for splitting. It ensembles multiple models, and the combination of these multiple models makes a single prediction.
- **Baseline Model :**To test the accuracy of the prediction, initially a baseline model is tested by multiplying the total airfare prices with its own mean. This baseline model score is compared with the model prediction scores.
- **Model 2:** Initially, many models on a **Forward propagation neural network** with Adam as an optimizer with the input of various hidden layers were tested. The model was underfitting the data. The model was producing high errors, and as a result, all those models will produce inaccurate results, which may incur losses to the airline business in strategizing the fare amount.

- The **Random Forest Regressor** model was tested with hyperparameters such as:
    - **N_estimators:** The number of decision trees that will be run in the model.
    - **Max_depth:** The maximum possible depth of each tree is set.
    - **Min_sample_leaf:** The minimum number of samples required to be a leaf node.
    - **Min_samples_split**: The minimum number of samples required to split an internal node.
    - **Random_state:** to get the same set of data.
    - This model gave a desired result that neither overfit nor underfit the data.

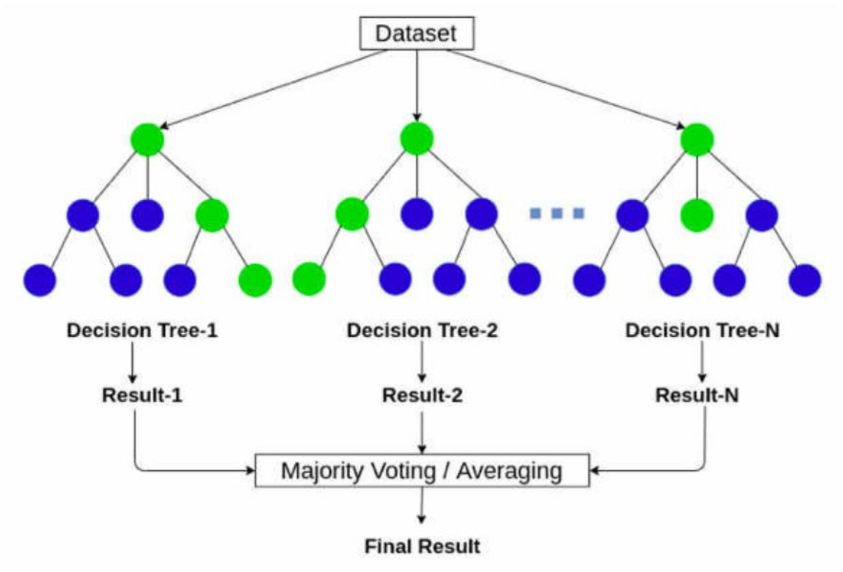

Fig. 6: Random Forest Regressor

Approach 4: The fare for a refundable ticket vs a non-refundable one

- The first essential step in modelling is to check the baseline performance of the model. I imported the accuracy metrics ‘Mean Sqaured Error’ (mse) and ‘Mean Absolute Error’ (mae). The mse and the mae achieved for the baseline performance was 207.481 and 154.9437 respectively. The baseline model acts as a benchmark and can be compared against the accuracy scores of the models developed using machine learning algorithms.

- Further, I decided to implement modeling using sklearn pipeline. I created a num_transformer which scaled all the numeric columns. Scaling all the numeric features ensures all the features are scaled on a single range, thus making machine learning models perform better. Then, I developed a cat_tranformer which encoded all categorical columns using target encoding. Later, I created a preprocessor which consisted of all the transformers defined.

- In the modeling pipeline, I defined the steps, first being the preprocessor which will convert all the data into numeric which will then be fed to the ml model. I decided to implement an XGBoost regressor. The reason behind this is XGBoost is a computationally efficient algorithm and performs efficiently on large datasets. After defining the model in the pipeline, I fit this pipeline on X_train, y_train.

6. Evaluation

a. Evaluation Metrics

- Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were chosen as the evaluation metrics for this assignment.
- MAE is easy to understand and interpret. It provides a straightforward measure of the average magnitude of errors without considering their direction. Each absolute error is treated equally in the calculation, making MAE robust to outliers. For flight ticket prediction, MAE gives a clear indication of the average discrepancy between predicted and actual fares.
- RMSE penalizes larger errors more heavily than smaller errors due to the squared term. This makes RMSE sensitive to outliers and large errors, providing a more nuanced view of the overall model performance. The use of the square root ensures that RMSE is in the same units as the target variable, making it directly interpretable. RMSE is commonly chosen when larger errors need to be emphasized, which is valuable in scenarios where the impact of large errors is significant, such as predicting ticket costs.

b. Results and Analysis

Model 1: The average fare

As the model was developed using the Decision Tree Regressor, evaluation metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) were computed to assess its performance. Here are the results:
Root Mean Squared Fare Prediction: 138.97 USD
Mean Absolute Fare Prediction: 97.54 USD

Model 2: The fare specific to airlines that run on the given route at the given hour

The following best scores were recorded for each of the algorithms used:

Algorithm	Training MAE	Validation MAE	Training RMSE	Validation RMSE
Base Model	158.9194	NA	231.7206	NA
Linear Regression	109.1267	113.6353	161.9104	157.8313

Random Forest	95.9027	101.8221	137.0676	142.0360
Gradient Boosting	76.7069	81.9992	112.9231	117.8925

Based on these scores, it was clear that Gradient Boosting algorithm with n_estimators=50, max_depth=8 and min_samples_leaf=3 performed best among all though there was slight overfitting that was noticed. We checked its performance on testing data, and it seemed to provide pretty good results and hence, we decided to go ahead with it.

Model 3: The fare for a flight with no layover and 1-3 layovers

Baseline Model Score:

The mse score of **baseline** model is: 226.827

The mae score of **baseline** model is: 168.3565

Model	**Training score**	**Validation score**	**Test set score**
1.Random Forest Regressor	Mse : 183.0577 Mae: 134.1100	Mse: 183.218 Mae: 134.297	Mse: 184.1858 Mse: 134.6509
2. Neural Network: 3 hidden layers, L2 loss function	Mse: 30401.3145	----	Evaluation: 30924.662109375 Prediction: 496.92752 Y_test iloc[0] : 488.6
3.Neural Newtork: 4 hidden layers-128 neurons	Mse: 30743.9707	----	Evaluation:32112.923828125 test prediction :553.176 y_test iloc [0] 594.7
4.Neural network: 7 hidden layers	Mse : 29818.6641	---	Evaluation: 30741.080078125 Prediction: 30741.080078125 y_test.iloc[0] 594.7

Model 1: Random Forest Regressor: Though the validation set, test set slightly overfits the data, but compared to all the model tested Random forest gives a better prediction results.

Model 2: Neural Network: with three hidden layers and regularization to manage loss

Sequential( ) # First layer

layer1 = (Dense(70, activation='relu', input_shape=(13,) , kernel_regularizer=l2(0.01)))

# Additional hidden layers

layer2 = (Dense(64, activation='relu', kernel_regularizer=l2(0.01)))

layer3 = (Dense(32, activation='relu', kernel_regularizer=l2(0.01)))

# Output layer

top_layer =(Dense(1))

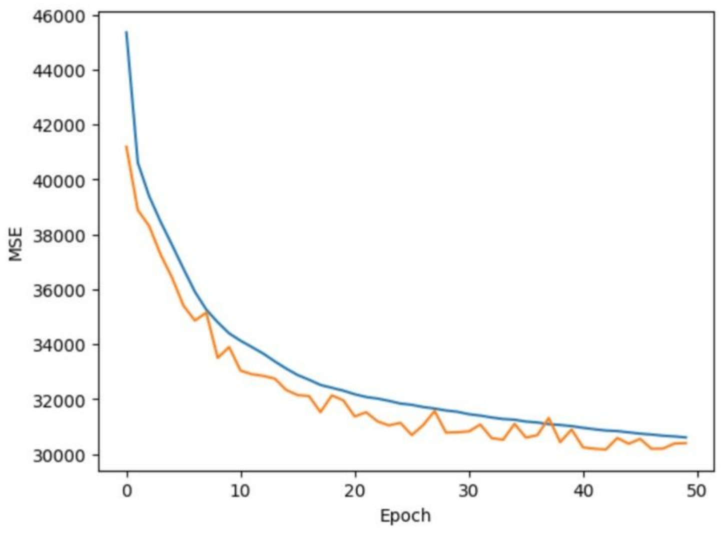

Fig. 7: Neural Network mse matrix analysis on 3 hidden layers

Analysis:

Model 3: Neural network with 4 hidden layers and 128 highest neurons and least are 38

Model.sequential()

layer1 = (Dense(128, activation='relu', input_shape=(15,)))

layer2 =(Dense(70, activation='relu'))

layer3 =(Dense(50, activation='relu'))

layer4 =(Dense(38, activation='relu'))# Add the input layer with the desired input shape

top_layer=(Dense(1)) # Output layer

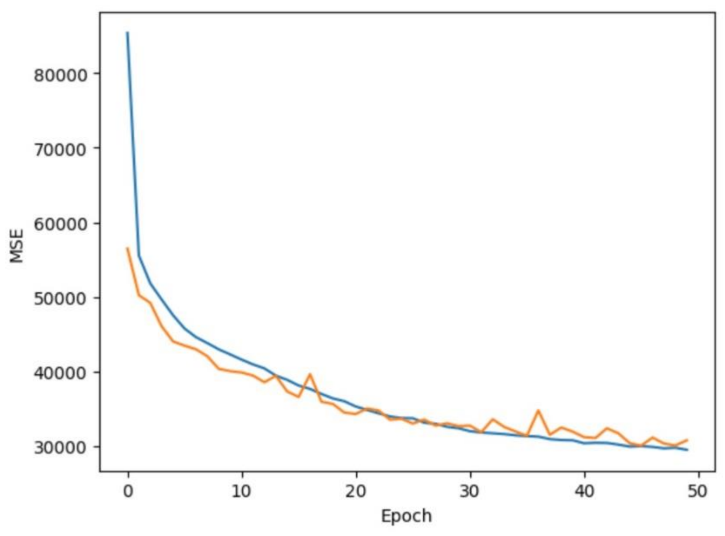

Fig. 8: Neural network matrix analysis on 4 hidden layer

Analysis: Model works well between epoch 20 to epoch 30, model overfits the data.

Model 4: Neural networkwith 7 hidden layers, maximum of 140 neurons and minimum of 5

Model.sequential()

layer1 = (Dense(140, activation='relu', input_shape=(15,)))

layer2 = (Dense(120, activation='relu')) # A hidden layer with 250 units and ReLU activation

layer3 = (Dense(80, activation='relu'))

layer4 = (Dense(50, activation='relu'))

layer5 = (Dense(30, activation='relu'))

layer6= (Dense(20, activation='relu'))

layer7= (Dense(5, activation='relu'))# Add your output layer

top_layer =(Dense(1)) # Output layer

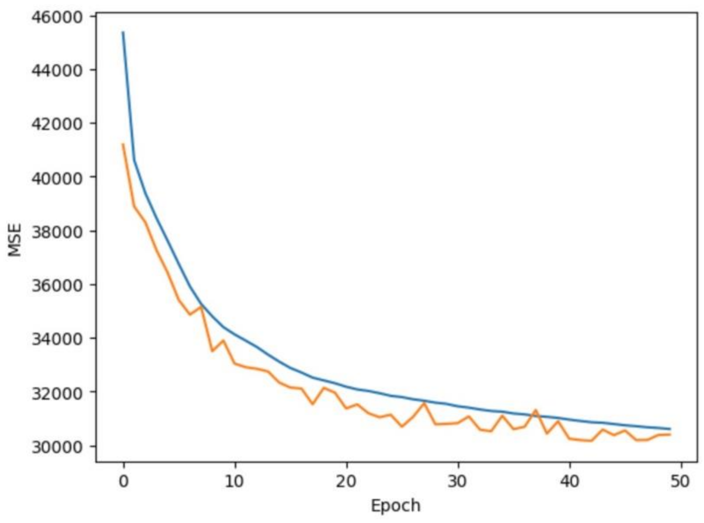

Fig. 9: Neural network mse score analysis with 7 hidden layers

Analysis: Model worked well at epoch 7,25 ,38, and data is over fitted.

Model 4: The fare for a refundable ticket vs a non-refundable one

The following evaluation metric scores were achieved from modelling:

Algorithm	Training MAE	Validation MAE	Training MSE	Validation MSE
Baseline Model	154.9437	NA	207.4813	NA
XGBRegressor	89.8906	89.9654	129.5864	129.6141

After testing the model on the unseen test data, following results were obtained:

MAE on testing dataset = 89.9323

MSE on testing dataset = 129.8985

c. Business Impact and Benefits

The final model revolutionizes fare predictions, enabling businesses to:

- **Optimize Pricing Strategies:** Achieve precise average fare predictions, enhancing competitiveness and revenue management.
- **Tailor Offerings:** Strategically price routes and times, catering to diverse traveler preferences and maximizing revenue potential.

- **Adapt to Market Dynamics:** Dynamically adjust prices in response to market changes, improving competitiveness and customer experience.
- **Enhance Customer Satisfaction:** Offer accurate fare predictions for layover options and refundable tickets, leading to improved customer satisfaction and loyalty.

d. Data Privacy and Ethical Concerns

- As the project deals exclusively with airline-related data, the potential impact on individual privacy is less direct. However, it's essential to consider the sensitivity of certain airline data, such as pricing strategies and operational details.
- To address ethical concerns in model deployment, collaboration with industry stakeholders can be emphasized. Ensuring that the model's outputs contribute positively to the airline industry and do not unduly impact market dynamics.
- Ensure that the project adheres to industry guidelines and standards related to the use of airline data. Collaboration with industry bodies can help establish and uphold ethical norms.
- Maintain transparent communication with stakeholders, including airlines, about the project's goals, methodologies, and the ethical principles guiding the use of the data. This transparency fosters trust within the industry.
- Conduct regular audits and reviews of the project's practices to ensure ongoing adherence to ethical principles. This includes reviewing data usage, model outputs, and any potential impact on the airline industry.

7. Deployment

a. Model Serving

- Deploying the trained models involved integrating them into a Streamlit application, providing a user-friendly interface for end-users to input origin airport, destination airport, departure date and time, and cabin type.
- For model 2, the application dynamically retrieved airline information from the timetable file for the airline-specific prediction model.
- The predictions were then displayed in organized tabs within the Streamlit interface.

b. Web App

- This web application serves as a comprehensive solution for accurate flight ticket fare predictions, designed with a user-friendly interface using Streamlit.
- Users, including travel agencies, airlines, and individual travelers, can input travel details to receive precise predictions for various scenarios such as average fares, route and time-specific fares, layover analysis, and refundable vs. non-refundable ticket fares.
- To set up, users need to install dependencies, download the application files, and run the Streamlit command.
- The potential benefits include improved revenue management, enhanced customer satisfaction, and streamlined pricing strategies.
- For future commercialization, the application could be marketed as a subscription service for businesses or explore licensing models for integration with existing travel platforms.
- Current limitations involve real-time data updates and diverse data sources, with potential improvements focusing on automating updates and incorporating more advanced machine learning models to further enhance prediction accuracy.

8. Collaboration

a. Individual Contributions

In our collaborative group project, each member played a distinct and vital role, contributing their expertise to achieve a cohesive and successful outcome. Rohan took charge of merging datasets, ensuring the foundation for comprehensive analysis. Mahjabeen demonstrated proficiency in data cleaning and analysis, enhancing the quality and reliability of our dataset. Each group member independently crafted models tailored to their specific use cases, showcasing their analytical skills and domain knowledge. Rohan led the development of the Streamlit application and seamlessly integrated the diverse models with the help of the respective team members. All team members contributed in the final report. This collaborative effort resulted in a synergistic combination of skills, with each member's contributions contributing to the project's overall success.

b. Group Dynamic

Our team's collaboration was marked by open communication and mutual respect. Regular Zoom meetings and Whatsapp group messages facilitated seamless coordination and task sharing, ensuring each member's strengths were leveraged. Embracing a transparent workflow and utilizing collaborative tools contributed to our project's success.

c. Ways of Working Together

Setting a deadline for each task and meeting frequently to discuss the progress were the key steps for project success. Weekly virtual meetings were held to discuss progress, track milestones, and make collaborative decisions.

d. Issues Faced

Challenges like data inconsistencies and model complexities were addressed through clear communication, ad-hoc issue resolution, and flexible adaptation. Lessons learned highlight the importance of early issue identification and continuous communication for smoother collaborations in the future.

9. Conclusion

In conclusion, this project has demonstrated the pivotal role of accurate fare predictions in transforming the airline industry's pricing strategies. The positive outcomes, including revenue optimization and heightened customer satisfaction, underscore the potential for leveraging advanced predictive models. As we move forward, ongoing experimentation and refinement remain critical, offering avenues for continuous improvement and adaptation to dynamic market conditions. The success of this project sets the stage for a data-driven approach to pricing, emphasizing the importance of precision in meeting customer expectations and sustaining the industry's competitiveness.

10. References

- Dey, P. (2022, May 13). *Secret hacks to book cheap flight tickets!*. Times of India Travel. https://timesofindia.indiatimes.com/travel/things-to-do/secret-hacks-to-book-cheap-flight-tickets/photostory/91543242.cms
- Structure of random forest regressor algorithm the random forest ... (n.d.). https://www.researchgate.net/figure/Structure-of-random-forest-regressor-algorithm-The-Random-Forest-Regressor-model-is_fig1_371581286
- *Sklearn.ensemble.randomforestregressor*. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- Brownlee, J. (2021, January 12). *Gentle introduction to the adam optimization algorithm for deep learning*. MachineLearningMastery.com. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
- Oppermann, A. (2021, December 21). *Activation functions in deep learning: Sigmoid, Tanh, relu*. KI Tutorials. https://artemoppermann.com/activation-functions-in-deep-learning-sigmoid-tanh-relu/
- *1.10. decision trees*. scikit. (n.d.-a). https://scikit-learn.org/stable/modules/tree.html#:~:text=Decision%20Trees%20(DTs)%20are%20a,as%20a%20piecewise%20constant%20approximation.
- *Gradient boosting regression*. scikit. (n.d.-b). https://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_regression.html
- *Sklearn.linear_model.linearregression*. scikit. (n.d.-d). https://scikit-learn.org/stable/modulesgenerated/sklearn.linear_model.LinearRegression.html

11. Appendix:

Package:

List of class and functions:

1. NullRegressor class
2. drop_nan_values
3. replace_null_with_Zero
4. mean_null
5. median_null
6. drop_target
7. random_split_train_val_sets
8. save_train_val_sets
9. load_train_val_sets
10. split_train_val_test_random
11. save_train_val_test_sets
12. load_train_val_test_sets
13. print_regressor_scores
14. assess_regressor_set
15. fit_assess_regressor
16. test

Data Dictionary:

- 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.
- fareBasisCode: The fare basis code.
- travelDuration: The travel duration in hours and minutes.
- elapsedDays: The number of elapsed days (usually 0).
- 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.
- baseFare: The price of the ticket (in USD).
- totalFare: The price of the ticket (in USD) including taxes and other fees.
- seatsRemaining: Integer for the number of seats remaining.
- 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. The entries for each of the legs are separated by '||'.
- segmentsDepartureTimeRaw: String containing the departure time (ISO 8601 format: YYYY-MM-DDThh:mm:ss.000±[hh]:00) for each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsArrivalTimeEpochSeconds: String containing the arrival time (Unix time) for each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsArrivalTimeRaw: String containing the arrival time (ISO 8601 format: YYYY-MM-DDThh:mm:ss.000±[hh]:00) for each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsArrivalAirportCode: String containing the IATA airport code for the arrival location for each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsDepartureAirportCode: String containing the IATA airport code for the departure location for each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsAirlineName: String containing the name of the airline that services each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsAirlineCode: String containing the two-letter airline code that services each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsEquipmentDescription: String containing the type of airplane used for each leg of the trip (e.g. "Airbus A321" or "Boeing 737-800"). The entries for each of the legs are separated by '||'.
- segmentsDurationInSeconds: String containing the duration of the flight (in seconds) for each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsDistance: String containing the distance traveled (in miles) for each leg of the trip. The entries for each of the legs are separated by '||'.
- segmentsCabinCode: String containing the cabin for each leg of the trip (e.g. "coach"). The entries for each of the legs are separated by '||'.