EXPERIMENT REPORT

Student Name	Ronik Karki
Project Name	Assignment 3 (Group Project)
Date	10/11/2023
Deliverables	**Notebook Name:** notebook_ronik_at3.ipynb **Training models file:** Train_model_ronik.py **Model file name:** best_model_ronik.pb

---

1. EXPERIMENT BACKGROUND	
1.a. Business Objective	The primary objective of this project is to create a predictive model for estimating the total fare of domestic airplane tickets within the United States, based on the departure and arrival airports. The aim is to provide travelers with a tool that enables them to make informed decisions about when to purchase their tickets, ultimately saving money. Typically, ticket prices tend to increase as the departure date approaches, and by accurately predicting fares, travelers can secure cost-effective bookings. Given that the target variable (ticket fare) is a numerical value, the central focus of this project is to develop a regression model capable of forecasting the expected fare for a given time frame. To evaluate the model's performance, we will employ two key metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). Additionally, the R2-score will be used during the model development phase to assess how well it explains the variance and the goodness of fit. From a business perspective, the success of this project hinges on achieving low MAE and RMSE scores. High error values would present challenges for users relying on the model for fare predictions. For instance, if the error exceeds the expected fare range, it could result in inaccurate predictions, leading users to potentially overpay for their tickets. Therefore, a successful model should consistently produce MAE and RMSE scores that are within the range of typical total fares.
1.b. Hypothesis	The hypothesis for this experiment posits that boosting models such as XGBoost will outperform the linear regression model. This is based on the consideration that many features in the dataset do not adhere to the underlying assumptions that parametric models like linear regression rely on. In contrast, XGBoost does not impose strict underlying assumptions on its predictors. Additionally, it is expected that both the boosting and linear regression models, which can handle correlated predictors, will demonstrate superior performance compared to a naive model.

1.c. Experiment Objective	The objective of this experiment is to enhance model performance by leveraging correlated features and implementing feature engineering techniques to create new features. The ultimate goal is to identify and choose the most effective model for deployment in the Streamlit application, allowing users to make accurate predictions.

---

2. EXPERIMENT DETAILS	
2.a. Data Preparation	In the data preparation phase, the dataset was initially contained within zip files. To begin, all of these files were extracted and then consolidated into single files for each airport, grouping them according to their respective airport names. Given that some values within the dataset contained multiple values separated by "||," a pandas function was utilized to explode these values into separate rows. For example, the value "coach || coach" was transformed into two rows, with one for "coach" before the "||" delimiter and the other for "coach" after it. Subsequently, any duplicate records were removed, and the dataset was further consolidated by incorporating all of the relevant airport information into a single file. Additionally, to optimize memory usage, the dataset was downcasted. Following this, the data was ready for subsequent preprocessing and the application of various feature engineering techniques.
2.b. Feature Engineering	For this experiment, three types of feature engineering techniques were used: - **Data Imputation:** Missing values were identified in the "total travel distance" and "segment distance" features. To address these missing values, imputation was performed for each airport. The mode value was selected as the imputed distance, as it represents the most frequently occurring distance for a particular airport. This approach was chosen to ensure that the imputed values closely align with the typical distances associated with each airport, thereby enhancing the realism of the predictions. - **Mapping Categorical Features:** Three categorical features were considered for this model: the starting airport name, the destination airport name, and the cabin code type. Given the limited number of unique values for each of these features, mapping was employed. For the starting and destination airport names, where there were only 16 distinct airports, values ranging from 0 to 16 were created, with 0 representing the "unknown" category. Similarly, for the cabin code type, which had four categories, values were mapped from 0 to 4, with 0 signifying "unknown." - **Date-Time Feature Extraction:** The date of each flight was subjected to feature extraction to yield additional insights for the model. The date was transformed into separate features, including the hour, minute, day, month, and year. This extraction allows the

	model to better understand the temporal patterns and variations associated with each flight, potentially improving its predictive capabilities. - **Converting the Target Variable:** The total fare for airline tickets is influenced by various factors, including airline-specific pricing. Since the dataset contained the same input conditions but different total fare values due to the variability in airline prices, a transformation was applied to create a unified fare for a specific departure date and time based on the starting and destination airports. This transformation involved determining the most commonly occurring fare value for that particular time and route, using the mode value. This approach was chosen because the dataset lacked detailed user input for airline information, making it challenging to differentiate between airlines. Utilizing the mode value as the fare for a specific time and route was a logical choice, as it is likely to represent the prevailing fare for that period, given that most values in the dataset would be similar. This conversion helped to standardize the fare variable and make it more representative of the typical fare for a specific combination of departure time and airport route.
2.c. Modelling	A set of ten features was chosen for this experiment, and various machine learning models were trained using this feature set. Additionally, a cross-validation technique was employed with 5 different folds to assess the model's performance across different data splits. This allowed for the evaluation of the model's robustness and its ability to generalize to different subsets of the data. The top-performing model from this initial evaluation was then subjected to further optimization through hyperparameter tuning. The search space for hyperparameter optimization was defined using Hyperopt, a method for automated hyperparameter search. This step aimed to fine-tune the model's hyperparameters to decrease the loss of MAE score and enhance the overall performance of the model. In the hyperparameter tuning process, the following search space was defined using Hyperopt: - 'max_depth': This hyperparameter was selected from a range of values between 3 and 15, with increments of 1. It determines the maximum depth of the decision trees in the XGBoost model. - 'learning_rate': The learning rate was sampled from a log-uniform distribution in the range from -5 to 0. This hyperparameter controls the step size at each iteration while moving toward a minimum of the loss function. - 'n_estimators': The number of estimators (trees) was chosen from a range of values between 50 and 200, with increments of 10. It represents the number of boosting rounds, which is the number of decision trees to be created.

3. EXPERIMENT RESULTS

3.a. Technical Performance

Four models (i.e. Naive-Baseline Model, Linear regression model, XGBoost Regressor model and Tuned XGBRegressor Model) were run for this experiment. All of the fluctuations were noted and the following scores were achieved in 5 different splits.

1. **Baseline model:**

   This model was created using the average values as the prediction for the upcoming values. The following outputs were obtained on a 5-fold cross-validation

Metrics	Scores
RMSE	233.486
MAE	169.7098
R2	0

2. **Linear Regression Model:**

   The following outputs were obtained on a 5-fold cross-validation

Splits	Training	Testing
Metrics	**Scores**	**Score**
RMSE	193.463	193.375
MAE	134.580	134.556
R2	0.313	0.3136

3. **Xgboost Regressor Model:**

   The following outputs were obtained on a 5-fold cross-validation

Splits	Training	Testing
Metrics	**Scores**	**Score**
RMSE	144.894	144.912
MAE	94.540	94.575
R2	0.614	0.6145

4. **Xgboost Regressor Model Tuned:**

   The best hyperparameters obtained for the XGBoost model after hyperparameter tuning were:

   max_depth: 15
   learning_rate: 0.252
   n_estimators: 200

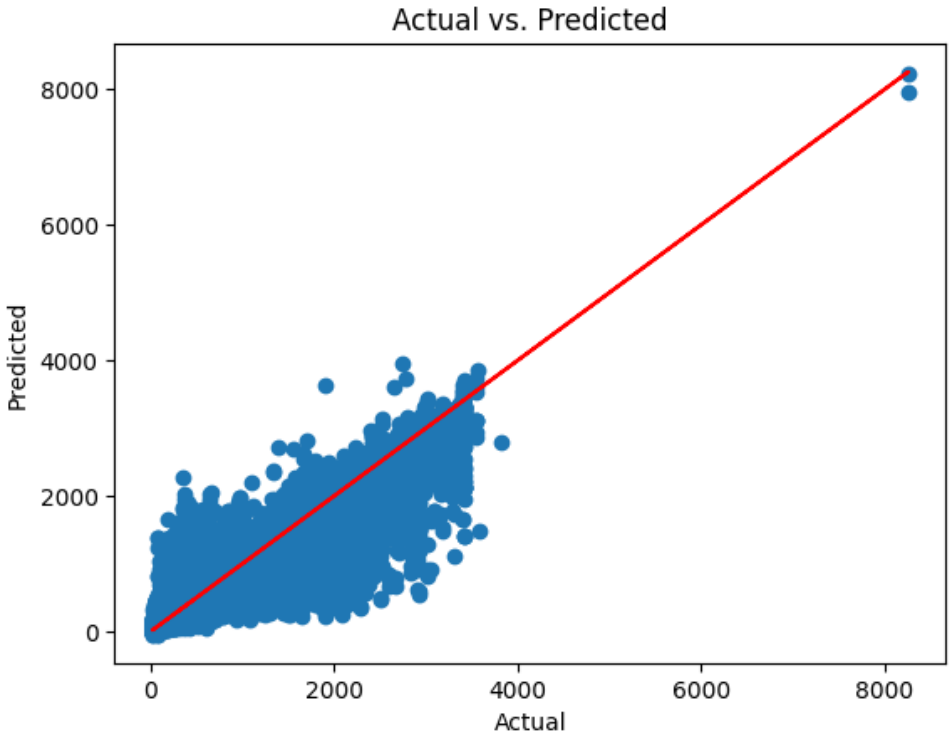
   These optimized hyperparameters were subsequently applied to the XGBoost model, leading to the following results.

Splits	**Training**	**Testing**
Metrics	**Scores**	**Score**
RMSE	62.424	66.846
MAE	40.251	42.263
R2	0.928	0.917

Based on the results obtained, it is evident that the XGBoost Regressor model(Table 4), after undergoing hyperparameter tuning, outperformed all other models. This outcome aligns with the initial hypothesis that this model would excel because it doesn't rely on strict underlying assumptions, making it more versatile and robust in handling the dataset.

In contrast, the Linear Regression model failed to meet expectations, likely due to its reliance on the underlying assumptions, which were not met by the dataset. This limitation prevented it from learning the data properly and achieving satisfactory performance.

While it is noted that the XGBoost-tuned model exhibited a slight performance difference between the training and test sets, this discrepancy was overshadowed by its overall strong performance and lower error. This indicates that the XGBoost-tuned model is a solid choice for accurate predictions and generalization to unseen data, making it the preferred model for this task.

3.b. Business Impact	 Fig 1: Actual vs predicted graph for XGB model With an MAE score of around 41 and RMSE scores of around 62, and the observation that the model's predictions tend to be below the actual rates, it's evident that the model performs reasonably well. The graphical representation in Figure 1 shows that the majority of the blue dots (representing predicted values) are close to the actual prices, with the red dot representing a perfect prediction. This indicates that the model has a decent ability to predict flight ticket prices, and its predictions are generally lower than the actual prices. Users can indeed benefit from this model by obtaining estimated ticket prices with minimal error. It offers a higher chance of purchasing tickets at a lower price, which aligns with the goal of helping users make cost-effective booking decisions. While the model might not be perfect, it appears to be a valuable tool for users seeking to save money on flight ticket purchases by providing reliable price estimates.
3.c. Encountered Issues	Dealing with a zipped dataset presented significant challenges, requiring substantial resources and time for the data concatenation process. Resource limitations posed constraints on both hyperparameter tuning and feature engineering, potentially hindering the model's optimization. With more abundant resources, there could have been greater exploration of hyperparameter spaces and the creation of more advanced features, which was hindered by memory issues. These limitations underscore the importance of efficiently managing and processing large datasets, as well as the potential for further model enhancement under more favorable resource conditions.

---

4. FUTURE EXPERIMENT

4.a. Key Learning	Preserving the dataset's values was a crucial aspect of this phase. Handling missing values, including their imputation, proved to be of utmost importance, especially since one of these imputed features turned out to be the most influential predictor for the project. Consequently, a careful approach to data handling was essential, with a minimal dropout of values, even in the worst-case scenarios. The downcasting process significantly aided in conserving memory and enhancing the model's performance when working with large datasets.
4.b. Suggestions / Recommendations	While the current model exhibits good performance, there are several opportunities for substantial improvement in future predictions when utilizing high-resource computing environments. Consider the following enhancements: 1. **Feature Engineering:** Expanding the feature set by incorporating more significant features and advanced feature engineering techniques, such as one-hot encoding, can provide the model with additional information to make more accurate predictions. 2. **Add Time Series Models:** Combining machine learning models with time series models can help capture temporal trends and patterns in the data. Models like ARIMA or Prophet might enhance the model's ability to forecast future prices precisely. 3. **Hyperparameter Tuning:** Utilizing techniques like grid search or Bayesian optimization for hyperparameter tuning can help identify the best set of parameters for the model. This fine-tuning can optimize the model's predictive capabilities and performance.