

Assignment

3

ML Data Product

Adv MLA 11/23

Group 15

Student Last Name	Student First Name	Student ID	Group Allocation
Vimalasri	Chanthru	25048240	Student A
Gupta	Tarun	24607855	Student B
Chen	Zhicong	12636736	Student C
Shrestha	Nipesh	24605646	Student D

Github	Project + Streamlit Repo: https://github.com/ChanthruV/data_product_with_ml_rf_model.joblib google drive link (too large for Github please download for testing of Streamlit): https://drive.google.com/drive/folders/1TwNR-UYZLnFx1KI46KrE9Wy4ZNHQLvpz?usp=sharing
--------	--

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

Table of Contents

1. Executive Summary	2
2. Business Understanding	3
3. Data Understanding	4
4. Data Preparation	6
5. Modeling	8
6. Evaluation	10
a. Evaluation Metrics	10
b. Results and Analysis	10
c. Business Impact and Benefits	12
d. Data Privacy and Ethical Concerns	12
7. Deployment	13
a. Streamlit	13
b. Model Serving	14
c. Web App	15
8. Collaboration	17
a. Individual Contributions	17
b. Group Dynamic	17
c. Ways of Working Together	17
d. Issues Faced	18
9. Conclusion	20
10. Appendix	21



1. Executive Summary

Overview

Our project developed a data product to estimate US airfares, aiming to make travel planning easier. We created a straightforward app that predicts flight costs based on user input, using machine learning to provide more accurate estimates.

Problem Statement and Context

We noticed that travelers often struggle with finding current and specific fare information, which can make budgeting for trips difficult. Our app attempts to address this by using updated data to give users a better idea of what they might pay for their flights. It allows users to customize their inputs to deliver more accurate estimations of fares for their trips.

Achieved Outcomes and Results

We've built a Streamlit app that uses four machine learning models to estimate airfares. It's easy to use and gives quick predictions. This shows how machine learning can be applied to help with everyday tasks like planning a trip.

In short, our project offers a practical tool that helps predict the cost of flying, which could make it easier for travelers to budget and plan their journeys.



2. Business Understanding

a. Business Use Cases

The project serves the travel and aviation industry by providing accurate fare predictions, essential for airlines, travel agents, and customers. It addresses the need for reliable pricing forecasts for airline ticketing—a tool that assists in making informed decisions about pricing strategies, inventory management, and purchasing tickets.

The impetus for the project came from the complex and ever-changing nature of airline pricing. Prices fluctuate due to various factors, including seasonality, demand shifts, and competitive pricing. Machine learning algorithms are particularly suited for this context as they can analyze historical pricing data to forecast future fares. The objective was to minimize the unpredictability of ticket pricing, aiding travelers in planning and businesses in maximizing revenue.

b. Key Objectives

The project had several clear goals:

1. To employ machine learning techniques (e.g., XGBoost, Random Forest etc.) to develop dependable models for predicting airline ticket prices.
2. To provide travelers with accurate fare predictions to inform their booking decisions.
3. To assist airlines and travel companies in refining inventory management and pricing strategies in response to forecasted fare changes.

Stakeholders and Requirements:

The stakeholders include passengers seeking cost-effective travel options, and travel agencies and airlines needing to predict fare fluctuations for better inventory and pricing decisions. Passengers require transparency and reliable information on fare trends, while businesses need predictive insights for strategic planning.

Project Approach:

To address these requirements, the project leverages machine learning algorithms using historical flight data. This approach allows for detailed fare predictions. For airlines and travel agencies, this translates into the ability to meet demand and adjust pricing strategies proactively. Meanwhile, travelers benefit from making more informed decisions on when to book flights, leading to savings and improved travel experiences. Overall, the project aims to bridge the gap between the need for pricing insights among stakeholders and enable smarter decision-making through machine learning.



3. Data Understanding

Insights into the Dataset

The dataset from Expedia is an extensive collection of flight data points that include price, dates, airports, and flight characteristics. The data is structured to reflect the variety of information a potential traveler would encounter when searching for flights, making it a comprehensive source for price prediction.

Data Sources and Collection Methods

Data was sourced directly from Expedia's listings, ensuring a high level of relevance as it reflects actual market offerings. It was made available via a google drive where a zip folder was downloaded.


Data Limitations

Whilst data was thorough and organised by airport, the various zip files created additional resistance in sourcing and cleaning the data. Zip file functions in python were used to overcome this but could have been averted through the availability of a more collated data source. Some data was missing e.g., 'totalTravelDistance' field, which is critical for fare estimation. Also, the high dimensionality due to numerous categorical variables like airport codes and airline names presented challenges for modeling. These challenges were investigated and dealt as part of data preparation.

Variables/Features Significance

Key variables include:

- legId: Useful for data tracking but not for modeling.
- flightDate and searchDate: Essential for capturing seasonal fare variations.
- startingAirport and destinationAirport: Critical for understanding route-specific price dynamics.
- fareBasisCode: Could indicate the fare class and its pricing structure.
- travelDuration and elapsedDays: Time-related factors that could influence pricing.
- isBasicEconomy, isRefundable, isNonStop: Service-related attributes that are likely to affect fare.
- totalFare: The target variable for our prediction model.

- 
- segments...: Detailed itinerary information that could help in fine-tuning the fare prediction but are complex to integrate directly into the models.

Exploratory Data Analysis

During the EDA process, we conducted a number of checks including:

- A range check for dates to understand the timeframe covered, to be potentially used for time series analysis.
- A data type assessment to plan necessary transformations.
- A search for missing values to strategize imputation methods.
- An evaluation of unique values in categorical variables to prepare for encoding by factoring in cardinality.



4. Data Preparation



Step 1: Unzipping and Loading Data

Objective: Streamline the process of getting data ready for use.

- Located the source of zip files and prepared a destination for the extracted CSVs.
- Wrote a script to automatically unzip files and compile the data into a single location for easy access.

Step 2: Combining Data into One DataFrame

Objective: Create a single view of all data points for analysis.

- Initialized a list to collect individual DataFrames created from each CSV.
- Combined these DataFrames into one master DataFrame for a unified dataset.

Step 3: Cleaning the Combined Dataset

Objective: Ensure data quality and relevance for modeling.

- Removed columns unrelated to the fare estimation to declutter the dataset. Focused on what inputs would be easiest for the user, what features were most vital to improve model performance, and what features could also be calculated if not available so they could be used within our Streamlit app.
- Filled in missing 'totalTravelDistance' values with the average to avoid data gaps.
- Deleted any duplicate entries to prevent data redundancy.

Step 4: Converting and Encoding Features

Objective: Transform data into a machine-learning-friendly format.

- Changed the 'travelDuration' from text to seconds for numerical consistency.
- Applied hashing to turn complex categorical data into a simpler numeric array, facilitating easier processing by algorithms.

Step 5: Normalizing the Data

Objective: Balance all features on a common scale to aid in model accuracy.

- Used StandardScaler to normalize features, negating the influence of outlier values.

Step 6: Splitting the Data for Training and Validation

Objective: Reserve a portion of data to test the model's predictions.

- Divided the dataset into training (80%) and validation (20%) sets to ensure the model can be assessed fairly.

Step 7: Establishing a Baseline for Comparison

Objective: Set a simple standard to measure the improvement of machine learning models.

- Determined the most common fare value in the training set to use as a constant prediction for initial comparison.

Step 8: Saving Preprocessing Artifacts

Objective: Document and save the steps and tools used for preprocessing for future consistency.

- Kept a record of which columns were removed and stored this list for reference.
- Saved the configured StandardScaler to apply the same transformation to new data in the future.

The focus in this portion of code was on maintaining data integrity, ensuring simplicity in the modeling process, and setting up robust standards for model evaluation.

5. Modeling

In our project, we explored various machine learning approaches to develop a model capable of predicting airfares based on user inputs. Our choice of algorithms was driven by the need for accuracy, computational efficiency, and the ability to handle large datasets. Most data preparation was handled in #4 above.

Random Forest (Approach 1)

Algorithm Selection and Hyperparameters

We chose the Random Forest algorithm for its robustness against overfitting and its proficiency with large, complex datasets. The default hyperparameters were initially employed to establish a performance benchmark.

Data Preparation

Random Forest's inherent handling of raw features meant that no additional feature engineering was necessary. However, due to the algorithm's computational demands, we trained the model on a 20% data subset.

Training and Challenges

The training process was straightforward but computationally taxing. We evaluated the model using the Mean Squared Error (MSE) metric. Despite its better performance compared to baseline models, the long training times led us to deprioritize Random Forest for this application.

XGBoost (Approach 2)

Algorithm Selection and Hyperparameters

XGBoost was selected for its speed and performance, particularly with tabular data. The hyperparameters were fine-tuned manually, although the default parameters provided the best results in our initial tests.

Training and Challenges

XGBoost was trained on the full dataset and showed promising results. However, the extensive manual tuning did not significantly improve performance. Given the dataset's size, a comprehensive grid search for hyperparameter optimization was not feasible without substantial computational resources.



Linear Regression (Approach 3)

Algorithm Selection and Hyperparameters

Linear Regression was tested due to its computational efficiency and interpretability. The hyperparameter `fit_intercept` was optimized using grid search, providing insight into whether the model should calculate the intercept term.

Training and Challenges

The model was trained on the entire dataset, proving to be a quick solution for airfare estimation, though not as accurate as more complex models.

Elastic Net Regression (Approach 4)

Algorithm Selection and Hyperparameters

Elastic Net Regression was employed for its capability to manage feature correlation and perform selection through L1 and L2 regularization.

Training and Challenges

We initially attempted VARMAX, which is tailored for time series data but was computationally infeasible. Elastic Net offered a pragmatic approach, albeit without the time series component. The focus shifted to Elastic Net due to its ability to handle our dataset's complexity within our computational constraints.



6. Evaluation

a. Evaluation Metrics

For our project, we used Mean Squared Error (MSE) as the evaluation metric to assess the performance of our predictive models. MSE measures the average squared difference between the estimated values and the actual value, providing a clear quantification of the model's prediction accuracy.

This metric was chosen because it directly corresponds to our project's goal of minimizing the discrepancy between predicted and actual airfares. A lower MSE indicates a more accurate model that can provide reliable fare estimates, which is essential for users planning their travel budget. Additionally, MSE is particularly useful for regression problems and is a standard measure for evaluating continuous numerical predictions, making it a fitting choice for our airfare estimation models.

b. Results and Analysis

Model	Train MSE	Val MSE
Base Model	44268.19	N/A
Random Forest	2032.21	10008.04
XGBoost	14534.06	14687.35
Linear Regression (Grid Search)	26187.99	26304.28
Elastic Regression	27711.33	27828.64

Analyzing the performance metrics provided, we can draw several conclusions about the effectiveness of each model used in the project.

Performance Analysis:

- The Base Model serves as a benchmark with a training Mean Squared Error (MSE) of 44268.19. It performs the worst of all but this aligned with expectations, being calculated from the mode.
- The Random Forest model shows a substantial improvement over the Base Model, with a training MSE of 2032.21 and a validation MSE of 10008.04. The discrepancy between the training and validation MSE suggests some overfitting to the training data, but it still outperforms the Base Model considerably on unseen data. It is our best performing model.

- XGBoost provides a consistent performance on both training and validation sets, with MSEs of 14534.06 and 14687.35, respectively. Despite having a higher MSE compared to Random Forest on the training set, its similar training and validation scores indicate good generalization to unseen data.
- The Linear Regression (Grid Search) model's performance, with MSEs of 26187.99 (training) and 26304.28 (validation), suggests a good balance between bias and variance. However, it doesn't capture the complexity of the data as effectively as the Random Forest and XGBoost models.
- Elastic Regression exhibits a higher MSE than the Linear Regression model in both training (27711.33) and validation (27828.64), indicating it is the least effective among the advanced models at capturing the relationship between features and airfares.


Key Insights:

- The Random Forest and XGBoost models' superior performance suggests that the relationships within the dataset are non-linear and complex, and that these models can capture such complexities more effectively than linear models.
- The consistent performance of XGBoost on both training and validation suggests that it is the most robust model among those tested, likely due to its ability to prevent overfitting through its regularization techniques.
- The Linear and Elastic Regression models, while performing worse than their ensemble counterparts, still offer valuable insights. Their relative underperformance could be due to the linear nature of these models, which assumes a linear relationship between the features and the target variable—a condition not perfectly met in this case.

Implications and Further Improvements:

The insights suggest that while ensemble methods have strong predictive power, there's a trade-off with computational efficiency and model interpretability. For further improvement:

- Additional feature engineering could help linear models perform better.
- Exploring other ensemble methods that may provide a better balance between accuracy and overfitting, such as Gradient Boosting or Bagging, could be beneficial.
- Considering the application of dimensionality reduction techniques might improve the efficiency of the models without significantly sacrificing performance.
- Further hyperparameter tuning, potentially using more sophisticated methods like Bayesian optimization, could enhance model performance.
- Exploring time-series relationships (e.g., seasonal trends) could be useful, however this is probably best on a smaller dataset that covers a wider period of time.



These insights will guide the next steps in the project, with a focus on refining models to better meet the project's goals while also considering computational efficiency and model interpretability.

c. Business Impact and Benefits

The deployment of our final model has the potential to significantly benefit businesses within the travel industry. By providing accurate fare predictions, the model addresses the challenge of unpredictable pricing, enabling travel agencies and airlines to offer better budgeting tools to their customers. This capability could translate into increased customer satisfaction and loyalty, as travelers can plan and budget with greater confidence.

In terms of quantifiable improvements, the model's high accuracy in fare prediction reduces the risk of over- or underestimating prices, which could lead to more competitive pricing strategies and potentially higher revenues. The precise fare estimation could also streamline inventory management, as it allows for a more data-driven approach to pricing decisions.

d. Data Privacy and Ethical Concerns

Data privacy is a cornerstone of this project. Sensitive information, such as personal identifiers, was not utilized in the model development. The data used was restricted to publicly available information and aggregated statistics that do not infringe on individual privacy.

Ethical considerations were paramount, especially in ensuring that the model does not perpetuate biases or lead to unfair pricing practices. To this end, the model was rigorously tested for fairness and consistency across different scenarios.

Steps to safeguard privacy and ethics could include implementing access controls and encrypting data in transit and at rest. By taking these measures, we can aim to maintain high standards of data privacy and ethical integrity throughout the project lifecycle.



7. Deployment

a. Streamlit

Creating the Streamlit app involved designing a user-friendly interface that allows users to input their travel details and receive estimated airfares. Here's how the app was constructed and the logic behind it:

User Interface Components

- Title: Introduced the app with a title "Local Travel Airfare Estimator" using Streamlit's `st.title` function.
- Origin Airport Dropdown: Implemented a dropdown menu using `st.selectbox`, populated with a list of starting airports provided by the `get_starting_airports` function.
- Destination Airport Dropdown: Similar to the origin, another dropdown for destination airports enables users to select their end point.
- Flight Date Input: A date input control allows users to select their flight date through `st.date_input`.
- Fare Type Checkboxes: Checkboxes for basic economy, refundable tickets, and non-stop flights capture user preferences using `st.checkbox`.
- Cabin Type Dropdown: Another dropdown lets users choose the cabin type, ranging from coach to first class.
- Airline Dropdown: Allows users to select an airline from a list returned by the `get_airline_names` function.

Back-End Processing

Upon clicking the "Estimate Fare" button, the following occurs:

- Capture Current Date: The current date is obtained to factor in the booking lead time.
- Preprocess User Inputs: The function `preprocess_user_input` takes the details provided by the user and prepares the data for the prediction model. This includes:
 - Converting booleans to integers for model readability.
 - Transforming dates to UNIX timestamps to handle them as continuous variables.
 - Encoding categorical data like airports and airlines using a `FeatureHasher` for dimensionality reduction.
 - Fetching average travel durations and distances for the selected routes from a pre-processed dataset, which aids in providing contextual predictions.
- Model Prediction: Each trained model's filename is listed in `model_filenames`. For each model, the app:

- Loads the trained model using joblib.
- Passes the preprocessed user input to the model's predict function.
- Collects the fare predictions from each model.

User Output

- Predicted Fares: The app displays a simple list of predicted fares from each model, formatted to two decimal places to represent currency.

Summary

The Streamlit app serves as an interactive front end to a set of machine learning models. Users can specify their travel preferences, and the app processes these inputs, applies the trained models, and outputs fare estimates. The app makes the machine learning models accessible to non-technical users, encapsulating complex data transformations and predictions within a clean interface.

b. Model Serving

Deployment Process:

- Selected the best machine learning model based on performance metrics.
- Utilized joblib to serialize the trained model, saving its state for later use.
- Prepared a deployment environment, like a cloud server, ensuring it has the necessary resources.
- Deployed the serialized model onto the server, ready to be called by the application.

Integration Considerations/ Challenges:

- Ensured that the model could be integrated seamlessly with the Streamlit web app.
 - This was probably one of the **greatest challenges** for us due to the group nature of this assessment. For this to work well and seamlessly, we all needed to ensure we had the same inputs for our respective models when training, so this aligned with expected inputs for the Streamlit app. In instances where this was achieved, the size of the dataset made certain models computationally inefficient too so there was a lot of trial and error involved.
- Established a reliable data pipeline for the model to receive and process input data.
 - Imputation of certain values that the user shouldn't need to input (e.g. travelDistance) posed a challenge and required creative yet simple solutions such as deriving the mean for a given origin and destination airport.

- Implemented error handling and logging to address any runtime issues that may arise.
 - Introduced a 'Predicting Fares...' loader to give users a greater sense of security and a better indication of processing times.

c. Web App

Purpose and Functionalities:

The web application acts as a user-friendly interface for the fare estimation model, allowing users to input travel details and receive fare predictions.

It includes dropdowns for airport selection, date pickers for flight dates, checkboxes for flight preferences, and a prediction button to initiate the fare estimation.

Setup and Launch Instructions:

- Ensure all dependencies, such as Streamlit and joblib, are installed in the deployment environment.
- Load the web application files onto the server, including app.py and features.py.
- Start the application using Streamlit, which will host the app on a local port that can be accessed via a web browser.

Potential Users and Use Cases:


- Travelers looking to estimate airfares for budgeting purposes.
- Airlines and travel agencies to provide quick fare estimates to their clients.
- Benefits include real-time fare estimates, personalized travel options, and the convenience of planning from anywhere.

Commercialization Potential:

- The application could be integrated into travel booking websites as a value-added service.
- Subscription-based access for frequent users or travel agencies.
- Data collected from user interactions could be utilized for market analysis and strategic planning.

Current Limitations and Improvements:

- The current version might not handle extremely high user loads, necessitating scaling solutions.
- Integration with live fare databases could enhance accuracy.
- User feedback mechanisms could guide iterative improvements and feature additions.
- Training on larger datasets would be useful in improving model accuracy.



The `app.py` and `features.py` files outline the backbone of the web application, detailing the workflow from user input to fare prediction. They highlight the use of preprocessing for user inputs, model loading, and prediction execution, all facilitated through an interactive Streamlit interface. The setup is designed to be straightforward, ensuring that users can get fare estimates with minimal clicks and waiting time.

■ ■ ■

8. Collaboration

a. Individual Contributions

Member	Contribution
Chanthru Vimalasri	<ul style="list-style-type: none">• Data Preparation Notebook (consolidated all pre-processing steps, creating a base for us to create our models from)• XGBoost Model (Hyperparameter Tuning)• Streamlit App – wrote code for app.py and features.py.• Final Report Writing
Tarun Gupta	<ul style="list-style-type: none">• Random Forest Model• Light GBM Model• Final Report Writing
Nipesh Shreshta	<ul style="list-style-type: none">• WideDeepModel• Linear Regression• Final Report Writing
Zhicong Chen	<ul style="list-style-type: none">• ARIMA Time Series Model• Elastic Net Regression• Final Report Writing

b. Group Dynamic

Group 15 established a healthy group dynamic. Communication was established early, first via email and then WhatsApp to streamline discussion about the project. Base data cleaning code was created and shared and to optimize time and resources, we individually constructed our machine learning models in line with our interests and overall goals. We then re-convened to discuss the integration of the Streamlit app and writing of the final written report. Questions were relayed via WhatsApp and group members did their best to answer these doubts.

c. Ways of Working Together

Our team follows agile project management methodologies to effectively oversee our projects. While it's challenging to conduct regular in-person meetings due to scheduling constraints, we maintain constant communication through online collaboration tools. This enables us to engage in real-time discussions on project-related issues and ideas, ensuring that information flows smoothly and rapidly among team members.

In terms of tracking project progress and assigning tasks, we have a well-structured approach. Although we don't rely on specific project management tools, our team members are diligent about understanding and completing their tasks. This level of commitment ensures that everyone is clear about their responsibilities and objectives. Additionally, we utilize GitHub as a code version

control and collaboration platform, simplifying the sharing and management of code among team members.

When it comes to teamwork, our core principles include open communication and collaboration. We had weekly check ins. Every team member was encouraged to contribute ideas, make suggestions, and participate in the decision-making process. This transparent communication and collaborative approach strengthen team cohesion and enhance work efficiency, allowing us to achieve our project's common goals effectively.

d. Issues Faced

Throughout the Local Travel Airfare Estimator project, we encountered a series of challenges:

Large Dataset Handling:

The considerable size of our dataset led to long training times and computational burdens. To mitigate this, we employed strategies such as feature selection to reduce dimensionality, data sampling to decrease training time, and parallel computing when possible. This also resulted in `rf_model.joblib` being very large and unable to upload to Github. As a workaround, a google drive link has been provided for model testing but more robust issues such as cloud storage would be a better course of action in the future.

Code Adjustment for Model Integration:

Whilst it was more efficient for us to individually train our respective models, some miscommunication led to disparities in the code. Not all models utilized the split train/val data from data preparation. This led to inconsistent feature inputs for models whilst the Streamlit app would require a common set of inputs amongst all models. Upon realizing this, we set up a meeting to discuss the problems in more detail. Where possible, code adjustments were made to standardize inputs. In some cases, such as the **Wide Deep Model (Appendix A)**, we collectively decided using a simpler model such as Linear Regression would be the wisest course of action as well as being computationally efficient. In cases where ARIMA/VARMAX time series were impractical, we utilised models with similar benefits (without time components) such as Elastic Net Regression.

Managing Busy Schedules:

Coordinating efforts amidst our busy schedules was a logistical challenge. We addressed this by setting clear milestones and regular check-ins to ensure accountability.



Streamlit App Development:

The Streamlit app was a new endeavor for all of us. To navigate this, we allocated time for research and learning at the project's onset. Members shared tutorials, documentation, and best practices as they learned, creating a knowledge base for the team. We conducted frequent brainstorming sessions to decide on the app's design and functionality, ensuring we fully leveraged Streamlit's capabilities whilst trying to make the user experience as pleasant as possible (e.g., with backend computation of travel distance and duration)

Lessons and Recommendations:

From these experiences, we learned the importance of effective communication, proactive problem-solving, and the allocation of time for learning and development. For future collaborations, we recommend:

- Starting with a clear division of labor and flexibility to adjust as the project evolves.
- Scheduling regular meetings and using collaborative tools to stay aligned. Ask more clarifying questions to avoid miscommunication.



9. Conclusion

The Local Travel Airfare Estimator project aimed to provide a reliable estimate of flight costs, and it has done so effectively, demonstrating the practical use of machine learning models in everyday decision-making tools. We've built a web application that simplifies the complexity behind airfare predictions into a user-friendly interface, empowering users to estimate travel costs with ease.

Project Goals and Stakeholder Satisfaction:

The web app has met its goal to offer a straightforward tool for airfare estimation, aligning with user expectations for simplicity and efficiency. We've provided stakeholders with a functional product that aligns with their needs for accessible travel information.

Reflection on Success:

While we set out to create a helpful tool, and by many measures, we succeeded, the true value of the project lies in its potential for growth and improvement.

Recommendations for Future Development:

Future enhancements could include:

- Updating the data feeding the predictive models to reflect current trends.
- Expanding the app's capacity to handle more simultaneous users.
- Refining the models with new data to improve prediction accuracy.
- Gathering user feedback for iterative development, ensuring the tool remains aligned with user needs.

Ultimately, the project has laid down a solid groundwork for a potentially more robust system in the future, and with continued attention and development, it can become an even more valuable resource for travelers and industry professionals alike.





10. Appendix

- Appendix A – please refer to 'Appendix A_WideDeepModel' in /reports of Github. This provides details on the initial experimental approach take with the Wide Deep Model, later replaced with Linear Regression for Streamlit model cross-compatibility.