

Airline Price Prediction: A Data-Driven Approach for Predicting Airline Prices

Data Detectives – Cumulative Progress Report 4

Team Members	Email
Swetha Tanikonda	swethatanikonda@loyalistcollege.com
Mohamed Maaz Rehan	mohamedmaazrehan@loyalistcollege.com
Devendra Singh Shekhawat	devendrasinghshek@loyalistcollege.com
Fatemi Sadikbhai Lokhandwala	fatemisadikbhailo@loyalistcollege.com
Gaurav Singh Rawat	gauravsinghrawat@loyalistcollege.com
Gaurav Singh	gauravsingh3@loyalistcollege.com
Jankiba Viralsinh Zala	jankibaviralsinhz@loyalistcollege.com
Comfort Iroha Onuoha	comfortirohaonuoh@loyalistcollege.com
Isha Savaliya	ishasavaliya@loyalistcollege.com
Tirth Patel	tirthpatel@loyalistcollege.com
Urjeet Parmar	urjeetparmar@loyalistcollege.com

Presentation Link	In-class presentation4
Website Link	SkyVista
GitHub Repository Link	Airline-Price-Prediction GItHub Link
GitHub Repository for our website	SkyVista GitHub Link

INDEX

1.Introduction	3
1.1 Project Overview.....	3
1.2 Project Objectives	3
1.3 Scope of the Project.....	3
2. Project Management.....	5
2.1 Roles and Responsibilities.....	5
2.4 Project Timeline.....	7
2.5 Accountability and Conflict Resolution.....	8
3. Technical Implementation.....	9

3.1 Data Collection and Preprocessing	9
3.2 Model Development.....	10
3.3 Model Performance Comparison	11
3.4 Feature Importance	11
3.5 Dashboard Development	12
3.5.1 Aesthetics.....	12
3.5.2 Accessibility.....	12
3.5.3 Interactivity	13
3.5.4 Insights.....	13
3.6 Deployment	14
3.6.1 Single Airline Prediction Page.....	15
3.6.2 Registration Page.....	16
3.6.3 Login page.....	17
3.6.4 Post-Login Page.....	18
3.6.5 List of Airline Prediction Page.....	19
4. Coding Review.....	20
4.1 Modular Structure:.....	20
4.2 Key Libraries and Technologies	23
4.3 Coding Standards and Collaboration.....	24
5. Testing.....	24
5.1 Unit Testing.....	24
5.2 Load Testing:	25
5.4 Speed & Latency Experiments.....	28
5.5 Error Correction & Optimization:	28
6. Results and Insights.....	29
6.1 Model Performance	29
6.2 Error Correction & Optimization	30
6.3 Collaboration and Consistency.....	31
6.4 Understanding and Contributions	31

7. Presentation Review	33
7.1 Presentation Effectiveness	33
7.2 Missed Points	34
8. Self-Assessment	34
8.1 Time Management and Collaboration	34
8.2 Conflict Resolution	35
8.3 Technical Implementation	35
8.4 Visualization and Presentation	35
8.5 Learning and Growth	35
9. Challenges & Solutions	35
10. Conclusion	36
11. References	37

1.Introduction

1.1 Project Overview

This report details the development of an airline price prediction project, a data-driven solution to help travelers find the best flight prices. Our project aimed to leverage machine learning techniques and data visualization to predict airfare based on various factors. The primary objective was to build a user-friendly and reliable system capable of providing accurate flight price estimations.

1.2 Project Objectives

The key objectives of the project are:

- To gather flight price data from various sources such as Kayak.
- To clean and preprocess the data for accurate and effective analysis.
- To build and evaluate predictive models and deploy them for user accessibility.
- To engineer relevant features that will enhance the performance of predictive models.
- To create an interactive dashboard for effective data visualization and user interaction.

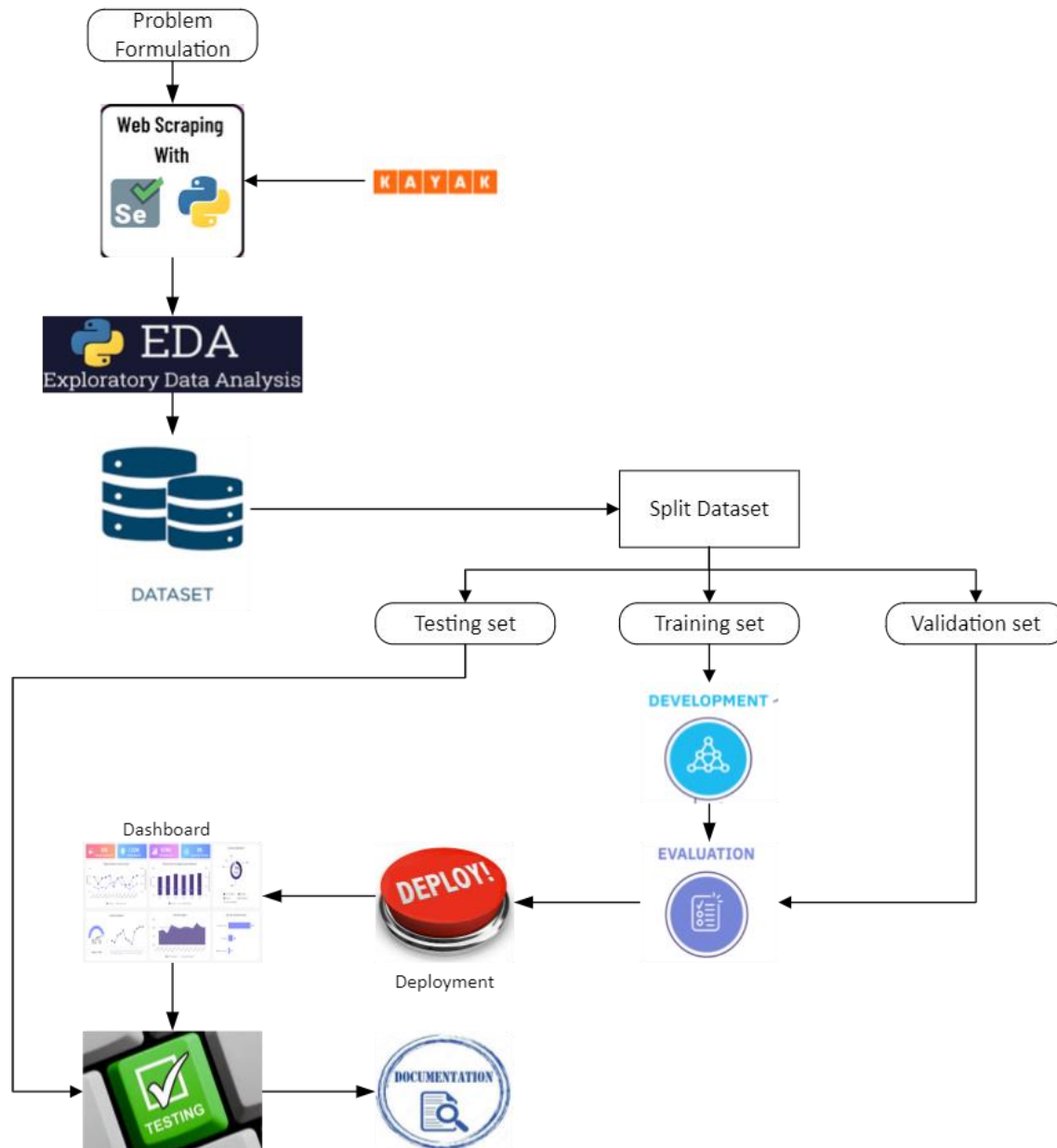
1.3 Scope of the Project

The project will analyze data such as past ticket prices, flight dates, number of stops, and other relevant factors to train the predictive models. The data will include features such as:

- Airline: The name of the airline operating the flight.

- Source: The departure city.
- Destination: The arrival city.
- Number of Stops: The number of stops or layovers between the source and destination.
- Class: The travel class (e.g., Economy, Business, First Class).
- Date: The date of departure.
- Total_Stopover_Time: The total duration of all stopovers combined in minutes.
- Price in CAD: The price of the ticket in Canadian Dollars (CAD)- (Target Variable).
- Days_left: The number of days left from the date of data collection until the departure date.
- Departure_24hr: The departure time in 24-hour format.
- Arrival_24hr: The arrival time in 24-hour format.
- Arrival_Day_Offset: The day offset for the arrival time relative to the departure date (e.g., 0 for same day).

This is the High-level architecture of this project:



2. Project Management

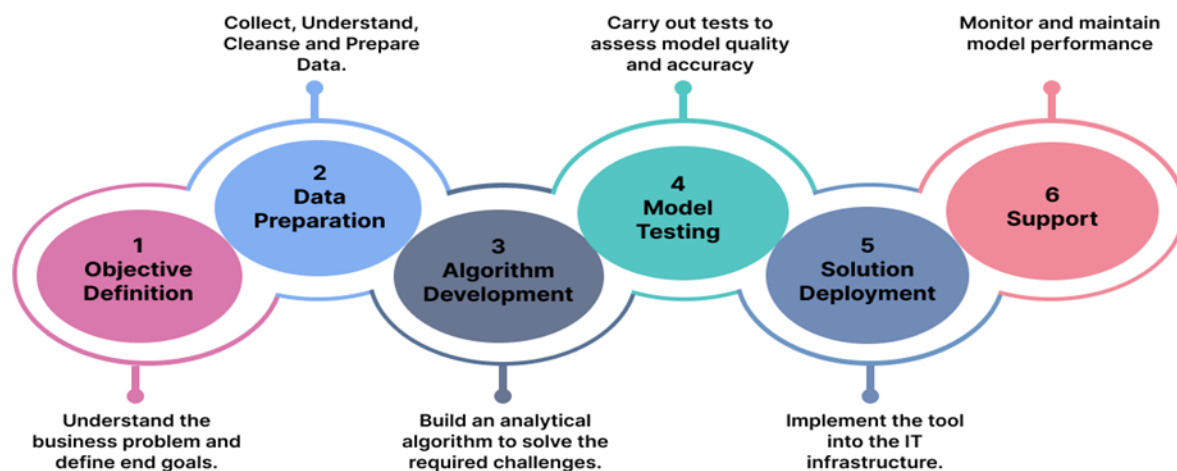
2.1 Roles and Responsibilities

This table shows what roles and responsibilities we have chosen and the reason behind the roles chosen.

Name	Role	Responsibilities	Reason
Swetha Tanikonda	Team Leader & Model Development	Lead the team, coordinate tasks, develop machine learning models, select algorithms, train models, and tune hyperparameters	Extensive experience in leadership and machine learning model development
Mohamed Maaz Rehan	Data Acquisition & Model Development	Collect and preprocess data, develop deep learning models	Proven expertise in data collection, preprocessing and deep learning development
Jankiba Viralsinh Zala	Testing	Conduct testing, document issues, validate fixes	Attention to detail and thorough understanding of testing processes
Devendra Singh Shekhawat	Testing	Test models, ensure quality, conduct testing	Strong background in quality assurance and testing
Gaurav Singh	Deployment	Assist with deployment, monitor and troubleshoot	Experience in model deployment and troubleshooting
Fatemi Sadikbhai Lokhandwala	Deployment	Deploy models, handle model deployment and integration	Expertise in python and web development and knowledge of deployment using streamlit
Urjeet Parmar	Deployment	Handle model deployment and also handles database for user	Expertise in model deployment and system integration
Isha Savaliya	Data Warehousing	Support warehouse design, maintain data integrity	Knowledgeable in data warehousing and ensuring data accuracy
Gaurav Singh Rawat	Data Warehousing	Manage data storage solutions, support warehouse design, maintain data integrity	Expertise in data management and warehousing
Comfort Iroha Onuoha	Dashboard Creation	Design dashboards, ensure visualization clarity, maintain dashboards, collaborate on requirements	Proficient in creating clear and effective data visualizations
Tirth Patel	Dashboard Creation	Developing and maintaining an interactive PowerBI dashboard with providing data visualizations to support decision making	Proven proficiency in developing clear and effective data visualizations and a strong track record in enhancing dashboard functionality and usability.

2.2 Project Lifecycle

Below image shows us the project lifecycle we are following for this project



2.3 Cost of work and number of hours worked

The table shows us the time invested till now for the project and how much it cost from our pocket for the project.

S.No	Student Name	Role	Total Hours Worked	Cost
------	--------------	------	--------------------	------

1	Swetha Tanikonda	Team Leader & Model Development	80hr	\$0
2	Mohamed Maaz Rehan	Data Acquisition	79hr	\$0
3	Devendra Singh Shekhawat	Testing	74hr	\$0
4	Fatemi Sadikbhai Lokhandwala	Deployment	76hr	\$0
5	Gaurav Singh Rawat	Data Warehousing	73hr	\$0
6	Gaurav Singh	Model Development	79hr	\$0
7	Jankiba Viralsinh Zala	Testing	70hr	\$0
8	Comfort Iroha Onuoha	Data Preprocessing	74hr	\$0
9	Isha Savaliya	Dashboard Creation	70hr	\$0
10	Tirth Patel	Dashboard Creation	75hr	\$0
11	Urjeet Parmar	Deployment	78hr	\$0

2.4 Project Timeline

We created our new timeline using Smartsheet which is less cluttered and has a better representation of what we have done till now and what we are going to do in the next weeks.

ID	Name	Apr, 24		May, 24				Jun, 24				Jul, 24				Aug, 24			
		21	28	05	12	19	26	02	09	16	23	30	07	14	21	28	04	11	
1	Week 1 - Research & Familiarization																		
2	Week 2 - Documentation Setup																		
3	Week 3 - Data Preprocessing																		
4	Week 4 - Data Acquisition																		
5	Week 5 - Feature Engineering Phase 1																		
6	Week 6 - Model Development Phase 1																		
7	Week 7 - Model Evaluation Phase 1	:																	
8	Week 8 - Study Break																		
9	Week 9 - Model Enhancement																		
10	Week 10 - Model Development Phase 2																		
11	Week 11 - Dashboard Creation																		
12	Week 12 - Final Testing & Integration																		
13	Week 13 - Presentation Preparation																		
14	Week 14 - Final Touches																		
15	Week 15 - Project Submission																		

The project timeline, as illustrated in the screenshots outlines the key phases, milestones, and tasks necessary for the successful completion of the project. It includes all phases, with specific start and end dates, dependencies, and critical milestones to ensure timely progress and completion.

2.5 Accountability and Conflict Resolution

Methods to Ensure Accountability:

- **Regular Meetings:** We have held weekly meetings to track progress and address any issues.
- **Task Management Tools:** Used tools like Microsoft teams and excel to assign tasks and monitor their completion.
- **Progress Reports:** Required team members to submit progress reports at the end of each week.
- Provided a clear definition of roles and responsibilities to avoid overlap and ensure each member is aware of their tasks.

Conflict Resolution Strategies:

- **Open Communication:** Frequent team meetings and discussions encouraged open communication to address issues promptly.
- **Mediation:** A designated team member facilitated discussions to find mutually acceptable solutions.

Mediation Example:

Scenario: Urjeet and Fatemi had a conflict over choosing the technology for the project website. Urjeet preferred AWS for its scalability, while Fatemi favored Django for its robust framework. Streamlit was also considered for its simplicity. Swetha, as the mediator, facilitated the discussion to find a solution.:

- **Identifying Concerns:** Swetha listened to both sides to understand their preferences and concerns regarding AWS, Django, and Streamlit.
- **Exploring Options:** Swetha encouraged discussion on how each option could meet the project's needs, focusing on scalability, development speed, and integration.
- **Finding Common Ground:** Swetha guided the team to consider the balance between development efficiency and functionality.
- **Reaching Consensus:** The team decided to use Django for the website due to its robust framework, while still considering AWS for deployment.

Outcome:

With Swetha's mediation, the team successfully chose Django for website development, aligning with project needs and team expertise.

2.6 Adaptations and Changes

- **Adjustment of Data Sources:** Initial data source is asking whether we are human or robot while fetching the data; so, we switched from using a Skyscanner to Kayak.
- **Algorithm Switch:** Based on preliminary results, switched from a basic regression model to a more complex ensemble method to improve predictions.
- **Timeline Adjustments:** Extended model development phase to allow for thorough testing and optimization.

Adjustment Example: Originally, the model development phase was planned for three weeks, followed by two weeks of testing. To enhance model reliability, the team extended the development phase by two weeks.

Reasoning:

- Extending the development phase allows for:

- More comprehensive testing to validate model performance.
- Fine-tuning of algorithms and parameters for better accuracy.
- Ensuring models meet project requirements before deployment.

Outcome: This adjustment ensures that the models are rigorously tested and optimized, improving their reliability and performance when deployed in operational settings.

2.7 Steps Taken to Execute the Project Plan

1. **Initial Planning:** Defined project scope, objectives, and team roles.
1. **Data Acquisition:** Maaz sourced live data from Kayak and demo data from Kaggle.
2. **Data Preprocessing:** Comfort cleaned the data, handled missing values, and encoded categorical variables.
3. **Feature Engineering:** Devendra created new features based on domain knowledge and selected the most relevant ones and is now working on testing.
4. **Data Warehousing:** Gaurav designed data storage solutions, ensuring accessibility and security. He also worked with Maaz to scrape data through the cloud due to the lengthy process of direct system scraping.
5. **Model Development:** Swetha and Gaurav Singh developed and trained multiple machine learning models.
6. **Model Evaluation:** Jankiba evaluated models using different metrics as necessary and is now focused on testing.
7. **Deployment:** Fatemi and Urjeet deployed the model by saving it as pickle files locally within the Django project's directory (models/pickles) and used views.py to call the model from the local directory.
8. **Dashboard:** Isha and Tirth developed interactive dashboards to visualize the model's predictions using Tableau and Power BI. They chose Power BI as its user-friendly, has collaborative environment, and seamlessly integrates with the dataset.

2.8 Decision-Making Process

- **Consensus-Based:** Decisions were made collectively, with input from all team members.
- **Task Delegation:** Tasks were delegated based on individual strengths and expertise.

3. Technical Implementation

3.1 Data Collection and Preprocessing

The data collection and preprocessing phase involved gathering historical flight price data exclusively from the Kayak website. The initial dataset required comprehensive cleaning and preprocessing to ensure consistency and suitability for analysis. This crucial step was essential to prepare the data for subsequent machine learning model development.

Preprocessing Steps

1. Data Cleaning:

- Removed duplicates and irrelevant data.
- Handled missing values through imputation techniques.

2. Data Transformation:

- Converted categorical variables into numerical format using encoding techniques.
- Normalized numerical features to ensure consistent scales across the dataset.

3. Feature Engineering:

- Extracted additional features like departure and arrival hours, days left until the flight, and total stopover time to enhance the model's predictive capabilities.

4. Data Splitting:

- Split the dataset into training and testing sets to evaluate the model's performance effectively.

3.2 Model Development

In the model development phase, we focused on creating and fine-tuning machine learning models to accurately predict airline ticket prices. This involved selecting appropriate algorithms, training the models on historical data, and evaluating their performance to ensure they met our accuracy requirements.

Key steps included:

1. **Algorithm Selection:** We experimented with various algorithms to find the best fit for our data. Algorithms considered included Linear Regression, Decision tree, Random Forest, ExtraTreeRegressor and XGBoost.
2. **Model Training:** Each chosen model was trained using scraped flight price data. This involved splitting the data into training and testing sets to evaluate model performance effectively.
3. **Hyperparameter Tuning:** We fine-tuned model parameters to optimize performance. This process was done using techniques such as Grid Search and Randomized Cross-Validation.

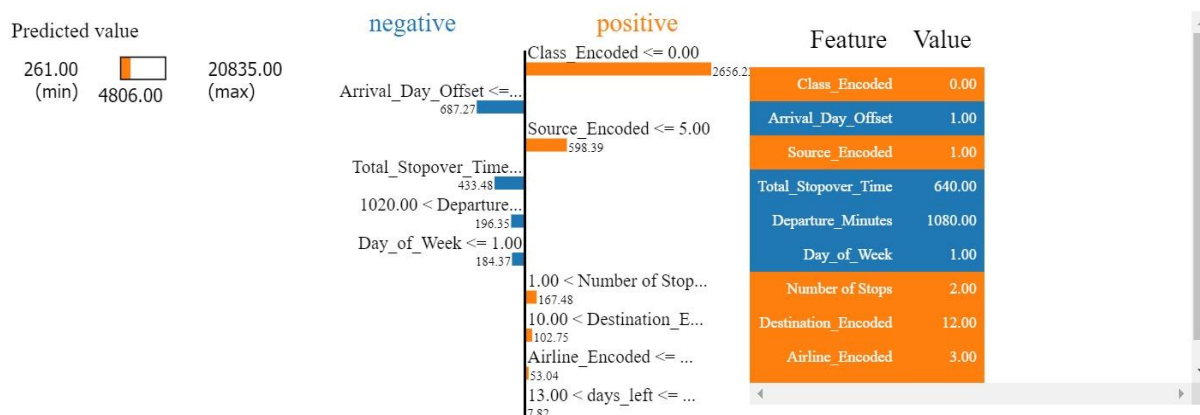
4. **Model Evaluation:** After training, we assessed the models' accuracy and robustness using performance metrics such as Mean Squared Error (MSE) and R-squared.

3.3 Model Performance Comparison

Here's a comparison of the performance of different models:

Models	Metrics				
	MSE	RMSE	MAE	R2	Size
Extratrees	404238.524	635.797	273.075	0.9089	1.51GB
Random Forest	425759.691	652.502	288.491	0.9041	1.39GB
Decision tree	600894.144	775.173	277.411	0.8646	18.7MB
Xgboost	1056905.448	1028.059	653.517	0.7619	477KB
Gradient boosting	1275499.236	1129.38	725.009	0.7127	186KB
Linear regression	3600841.786	1897.588	1411.61	0.1888	2kb

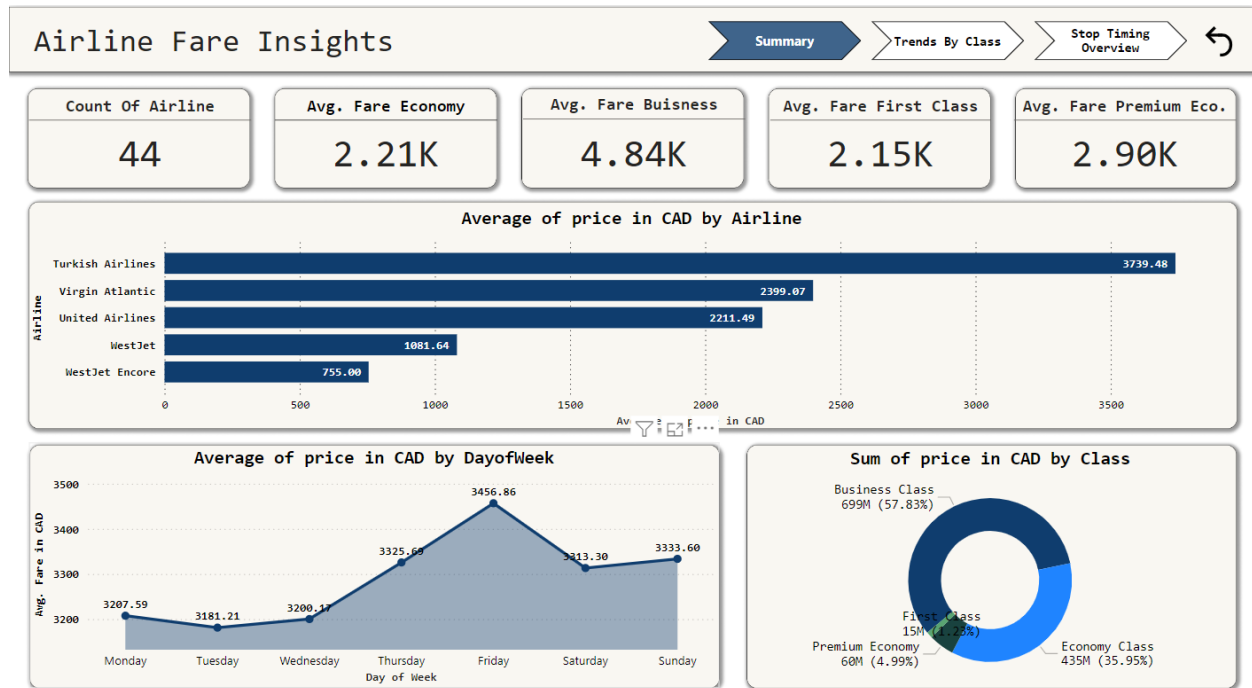
3.4 Feature Importance



This visual provides insights into the factors influencing flight predictions, such as durations or delays. It highlights both positive and negative influences on the predicted values. Key features like "Arrival Day Offset," "Source Encoded," and "Total Stopover Time" significantly impact the predictions. Understanding these influences can help optimize flight schedules, improve operational efficiency, and enhance passenger satisfaction by addressing the most impactful factors.

3.5 Dashboard Development

Our airline data dashboard is designed to offer an intuitive and interactive platform for exploring comprehensive flight information. Utilizing a detailed dataset, this dashboard empowers stakeholders with the tools they need for informed decision-making. In this section, we will outline how the dashboard addresses key factors to meet the diverse needs of its users.



Here's how our dashboard addresses aesthetics, accessibility, interactivity, and insights:

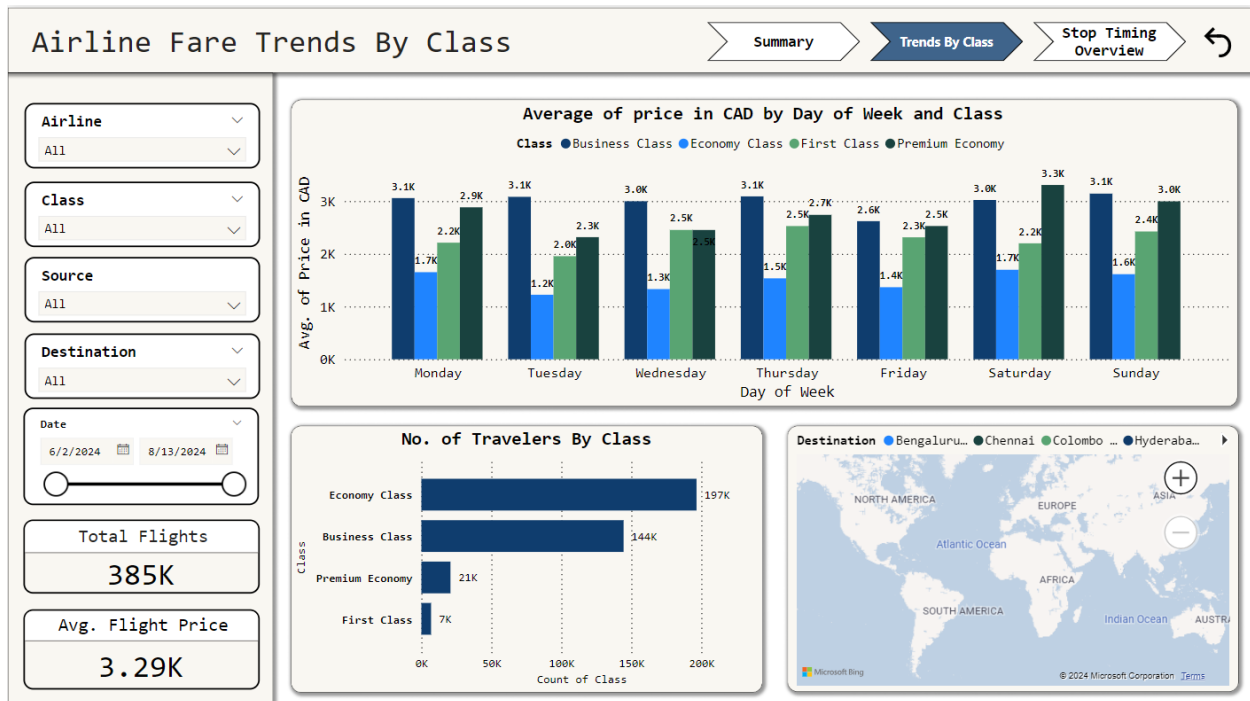
3.5.1 Aesthetics

Visual Appeal: The dashboard employs a clean and modern design with a consistent color palette, making it visually appealing. The use of blue hues for graphs and highlights ensures readability and a professional look.

Layout and Organization: The logical arrangement of elements, with key metrics at the top and detailed graphs below, provides a clear and organized view. This layout helps users quickly find the information they need.

3.5.2 Accessibility

User-Friendly Interface: The dashboard features a user-friendly interface with clear labels and a logical layout. The filters on the left side are straightforward, allowing users to easily refine their search based on airline, class, source, destination, and date range. This design ensures that users can navigate and interact with the dashboard effortlessly.



3.5.3 Interactivity

Filters and Controls: The left-side panel with filters (airline, class, source, destination, date range) allows users to customize their view and drill down into specific data points.

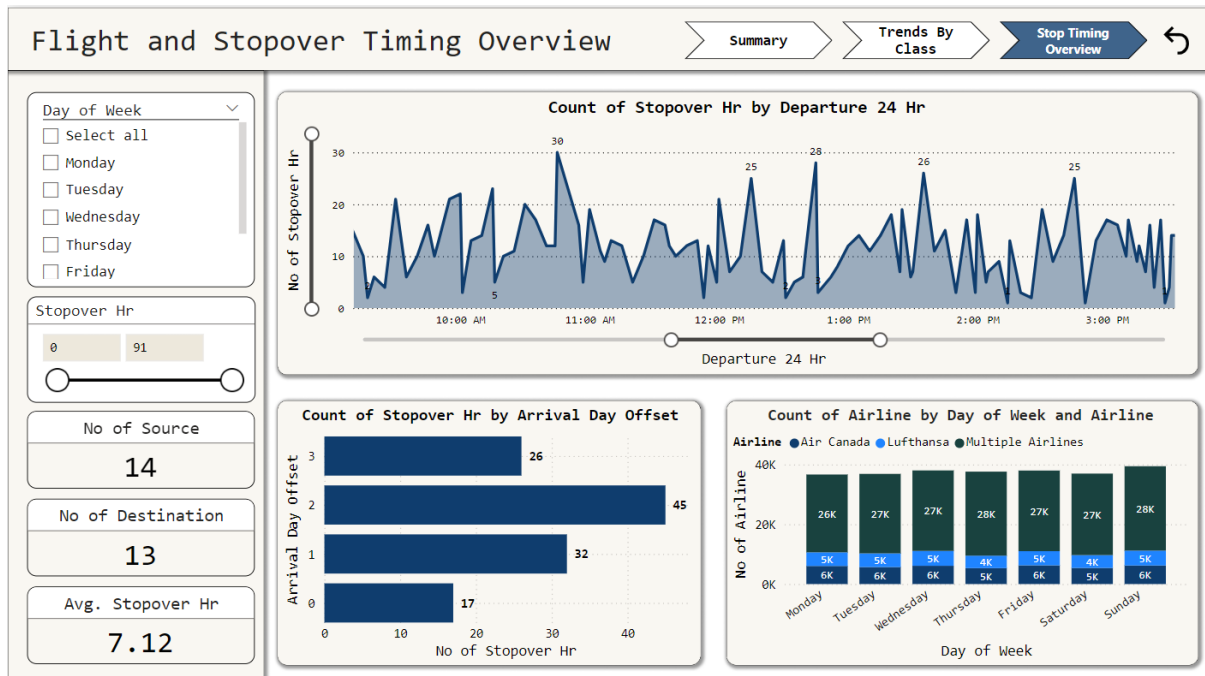
Dynamic Charts: Interactive features like customizable filters and dynamic charts let users explore data in different ways. For instance, hovering over data points in the graphs reveals detailed information, and the charts update in real-time based on the selected filters.

3.5.4 Insights

Key Metrics: The dashboard highlights key metrics like average fare prices (Economy, Business, First Class, Premium Economy) and total flights. These metrics provide a quick overview of the most important data.

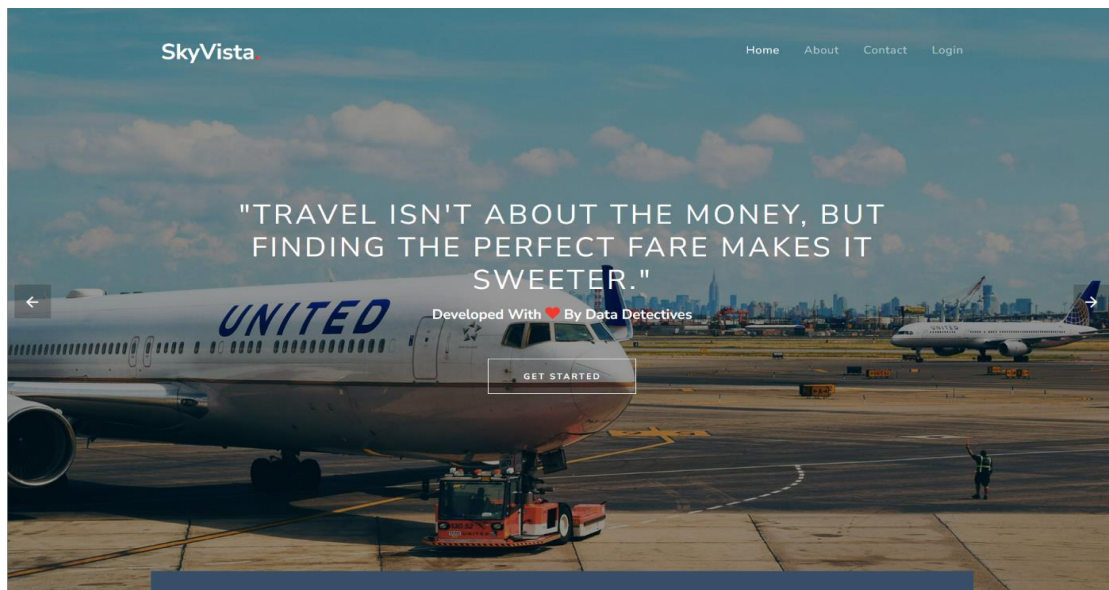
Detailed Analysis: Graphs such as the "Average of price in CAD by Airline" and "Average of price in CAD by Day of Week" offer deeper insights into fare trends. The "Sum of price in CAD by Class" donut chart provides a clear visual representation of fare distribution across different classes.

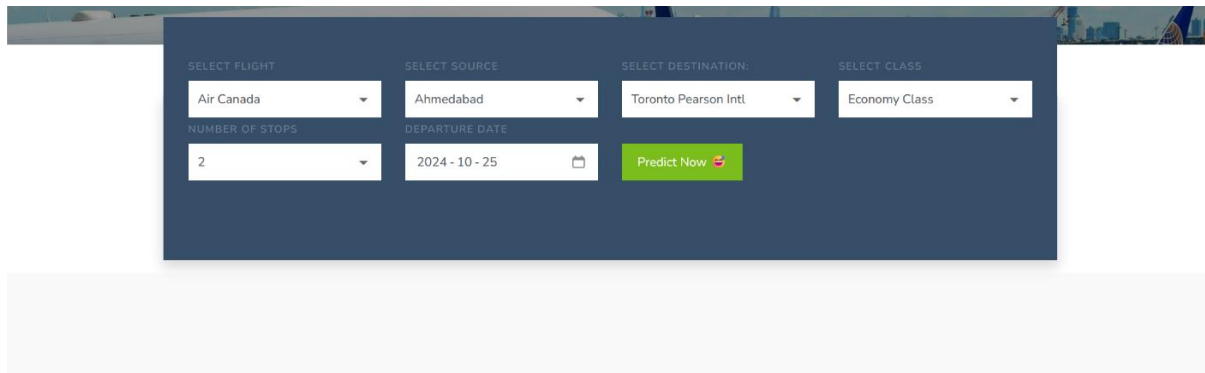
Comparative Analysis: The "Trends By Class" page allows users to compare average prices by day of the week across different classes, helping identify patterns and trends. The "Flight and Stopover Timing Overview" page provides insights into stopover durations and their distribution across different times and days.



3.6 Deployment

The project was deployed using a Django application on Render Cloud, ensuring robust performance and security. We implemented a secure HTTPS connection for data protection. The application includes user authentication with login and registration pages. It offers two main features: on the home page, users can predict prices for a single airline, while logged-in users can access price predictions for various airlines at a selected destination, displayed through interactive Bootstrap cards. This setup provides a seamless and secure experience for all of us.





A dark blue flight prediction form with white text and input fields. The form is divided into four main sections: SELECT FLIGHT, SELECT SOURCE, SELECT DESTINATION, and SELECT CLASS. Below these are NUMBER OF STOPS and DEPARTURE DATE. A green 'Predict Now' button is at the bottom right.

SELECT FLIGHT	SELECT SOURCE	SELECT DESTINATION	SELECT CLASS
Air Canada	Ahmedabad	Toronto Pearson Intl	Economy Class

NUMBER OF STOPS	DEPARTURE DATE
2	2024 - 10 - 25

Predict Now

Want more airlines rather than single price?

Well, our homepage model allows you to predict fare price for a single airline but if you want to see a list of airlines for a route the just sign up for our free service and get started on your journey to find a perfect fare.

Get Started

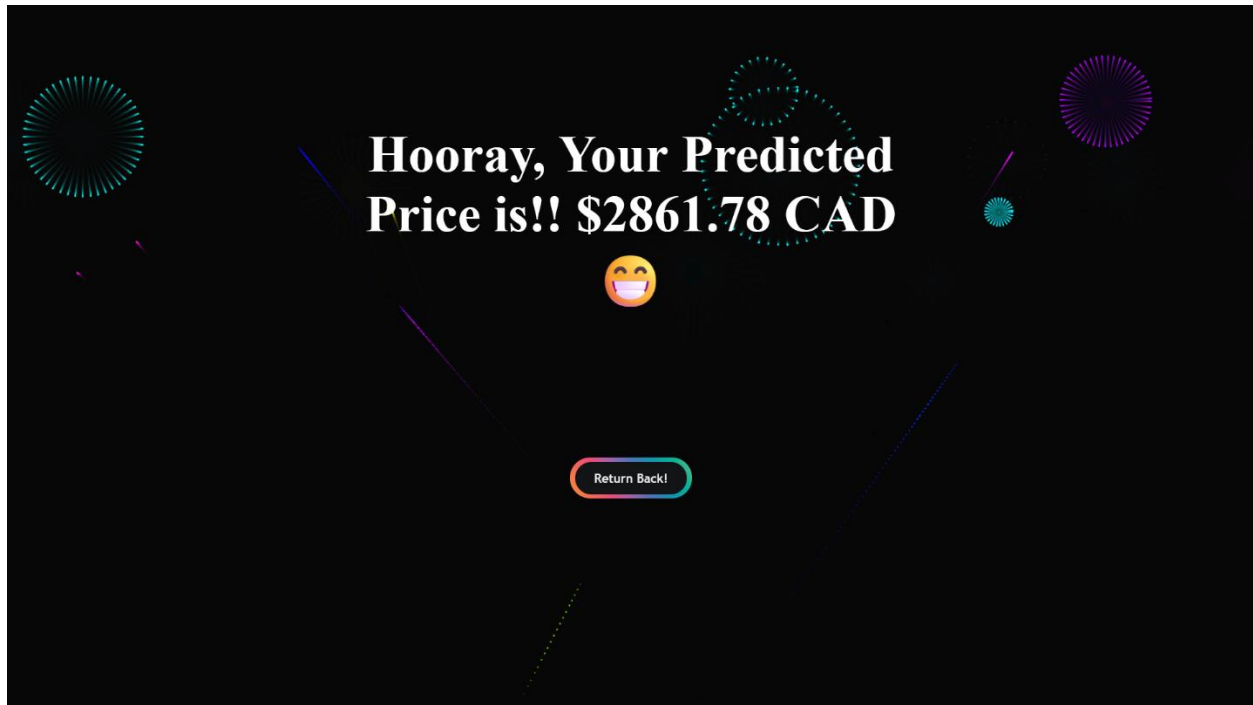
Header: See a List of Predicted Fare Prices.

Go Home

Airline	Source	Destination	Class	Price
Air Canada	Ahmedabad	Toronto Pearson Intl	Economy Class	\$1026.00 CAD
Air Canada	Ahmedabad	Toronto Pearson Intl	Economy Class	\$1026.00 CAD
Air Canada	Ahmedabad	Toronto Pearson Intl	Economy Class	\$1026.00 CAD
Air Canada	Ahmedabad	Toronto Pearson Intl	Economy Class	\$1026.00 CAD
Air Canada	Ahmedabad	Toronto Pearson Intl	Economy Class	\$1026.00 CAD

3.6.1 Single Airline Prediction Page

The Single Airline Prediction page enables users to swiftly obtain price predictions for specific airlines. By entering details such as the flight's source, destination, class, stops, and departure date, users receive an estimated price for their chosen airline. The interface is designed to be user-friendly, with clear input fields and an intuitive layout to ensure ease of use. This page focuses on providing accurate predictions efficiently, assisting users in making well-informed decisions about their flights.



3.6.2 Registration Page

The Registration page enables new users to create an account effortlessly by requiring basic information such as name, email, and password. It's simple and intuitive design guides users through the sign-up process with clear instructions. Upon registration, users gain access to additional features and personalized services on the website. This secure and user-friendly page ensures a seamless onboarding experience.

SkyVista Home About Contact Us

Sign Up

Enter your first name

Enter your last name

ubtecx@gmail.com

Enter your email

.....

Confirm password

☐ I accept all terms & condition

Register Now

Already have an account? [Login now](#)

3.6.3 Login page

The login page provides a straightforward way for returning users to access their accounts. Users simply enter their registered email and password to log in. The design focuses on ease of use, with clear prompts and a clean layout. This page ensures a quick and secure entry into the website, maintaining a seamless user experience.

SkyVista Home About Contact Us

Login

ubtecx@gmail.com

.....

Login

Don't have an account? [Sign Up now](#)



ABOUT SKYVISTA

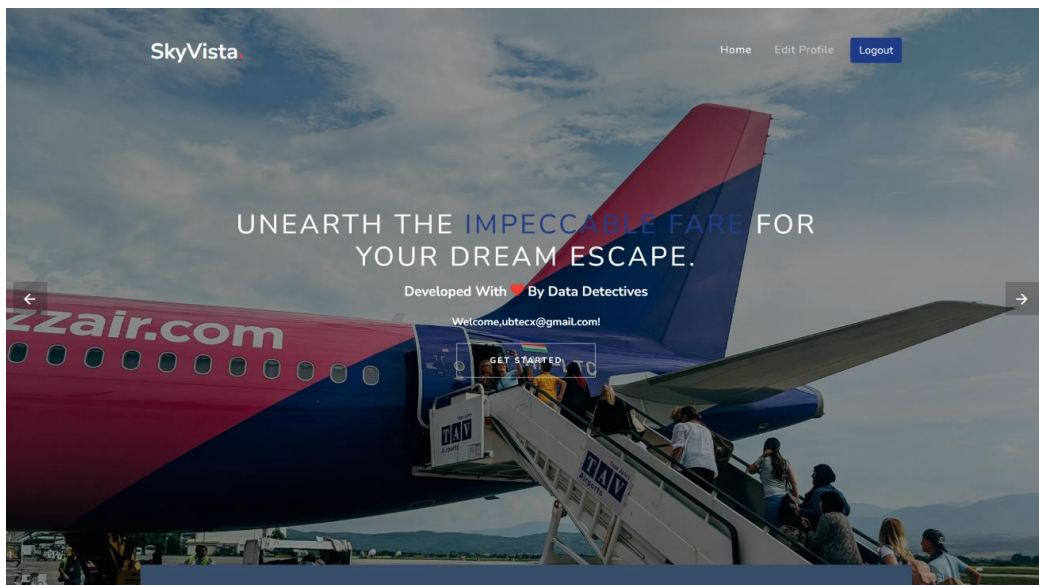
Discover Sky Vista: Your Flight Fare Prediction Experts

At Sky Vista, we use advanced machine learning to deliver accurate fare price predictions. Our platform helps you make informed decisions and optimize your travel budget with reliable insights into future flight costs. While we don't handle bookings, our data-driven solutions enhance your planning process and travel experience.

[Return Home](#)

3.6.4 Post-Login Page

After logging in, users are directed to a personalized dashboard where they can access advanced features. This page allows users to input specific travel details and select from a list of airlines to obtain price predictions for their chosen destination. The results are presented in an organized format, utilizing Bootstrap cards for a clean and interactive display. Users can easily compare prices across different airlines, enabling them to make well-informed decisions about their travel plans.



SELECT SOURCE

Ahmedabad

SELECT DESTINATION:

Toronto Pearson Intl

SELECT CLASS

Economy Class

NUMBER OF STOPS

2

DEPARTURE DATE

2024 - 10 - 26

Predict Now

WHY CHOOSE US?

Customers choose our airline ticket price prediction service for its exceptional accuracy derived from advanced machine learning models, ensuring they secure the best deals effortlessly. Our user-friendly interface and transparent predictions empower travelers to make informed decisions, providing a competitive edge in navigating the fluctuating airline ticket market.

3.6.5 List of Airline Prediction Page

On the "List of Airline Prediction" page, users can see predicted flight prices for multiple airlines based on their chosen destinations. After entering their travel details, users get a list of predictions displayed in easy-to-read cards, making it simple to compare prices and find the best options.

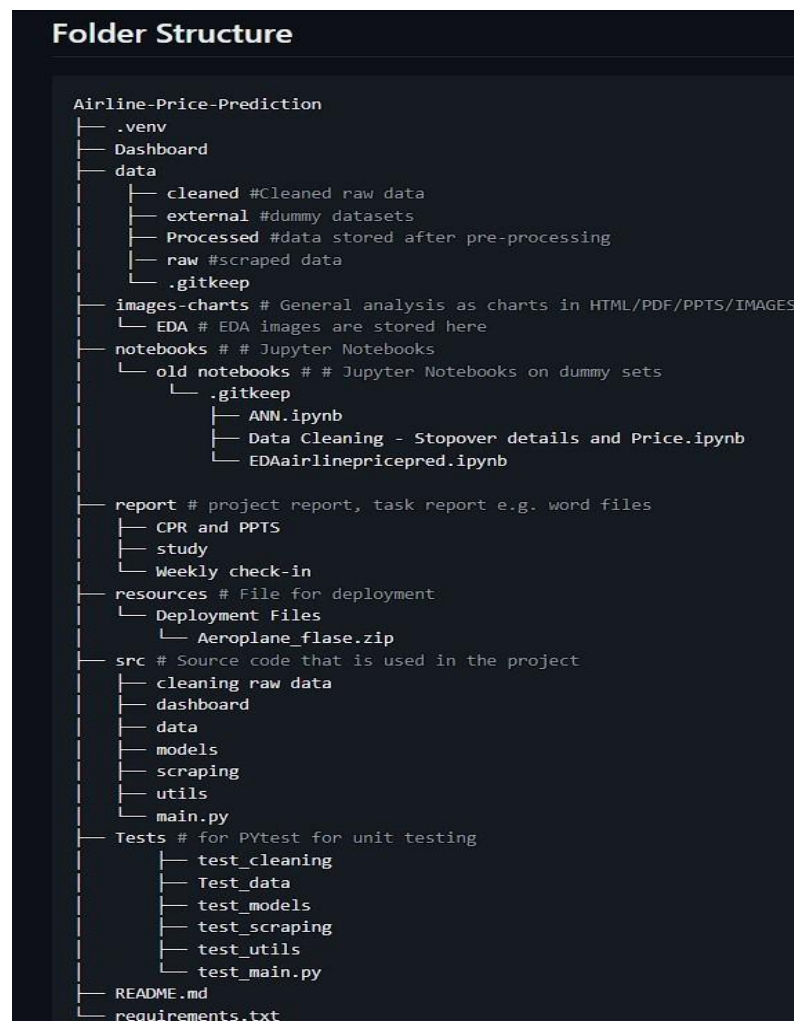
<div><div>✈️ Departure</div><div>Change</div><div>Flight date: 2024-10-26</div><div>2024-10-26</div><div>Ahmedabad → Toronto Pearson Intl</div></div>	→	<div><div>2024-10-26</div><div>Estimated price</div><div>\$2722.33 CAD</div></div>	<div>Royal Jordanian Economy</div>
<div><div>✈️ Departure</div><div>Change</div><div>Flight date: 2024-10-26</div><div>2024-10-26</div><div>Ahmedabad → Toronto Pearson Intl</div></div>	→	<div><div>2024-10-26</div><div>Estimated price</div><div>\$2661.72 CAD</div></div>	<div>SAUDIA Economy</div>
<div><div>✈️ Departure</div><div>Change</div><div>Flight date: 2024-10-26</div><div>2024-10-26</div><div>Ahmedabad → Toronto Pearson Intl</div></div>	→	<div><div>2024-10-26</div><div>Estimated price</div><div>\$2661.72 CAD</div></div>	<div>SWISS Economy</div>
<div><div>✈️ Departure</div><div>Change</div><div>Flight date: 2024-10-26</div><div>2024-10-26</div><div>Ahmedabad → Toronto Pearson Intl</div></div>	→	<div><div>2024-10-26</div><div>Estimated price</div><div>\$2661.72 CAD</div></div>	<div>Singapore Airlines Economy</div>

4. Coding Review

Our airline price prediction system is built upon a robust and well-structured Python codebase. We prioritize a modular design, as evidenced by distinct modules for data acquisition, preprocessing, model training, and deployment. Leveraging libraries like Selenium, Pandas, Scikit-learn, and Django, we've created an efficient and scalable solution. A user-friendly dashboard, built using PowerBI, allows for easy interaction with the system. Our commitment to clean code, PEP 8 style guidelines, and regular code reviews ensured maintainability and fostered seamless team collaboration. This section details the structure of our codebase, highlights key algorithms and libraries used, and describes our approach to maintainability and scalability.

4.1 Modular Structure:

We've designed our project with a modular architecture to improve code organization, readability, and collaboration.



```

class ModelTrainer:
    """Gaurav809"""
    def __init__(self, X, y, model_params=None):
        self.X = X
        self.y = y
        self.model_params = model_params if model_params else MODEL_PARAMS
        self.models = {name: cls(**self.model_params.get(name, {})) for name, cls in MODELS.items()}
        self.trained_models = {}

    2 usages 1 Gaurav809
    def train_model(self, model_type):
        if model_type not in self.models:
            raise ValueError('Model type {} is not supported'.format(model_type))

        model = self.models[model_type]
        X_train, X_test, y_train, y_test = train_test_split(self.X, self.y, test_size=0.2, random_state=42)

        model.fit(X_train, y_train)
        #joblib.dump(model, f'{MODEL_PATH}/{model_type}.pkl')
        self.trained_models[model_type] = model
        return model, X_train, y_train, X_test, y_test

```

```

# Define selected models
selected_models = ['random_forest', 'decision_tree', 'gradient_boosting', 'xgboost', 'linear_regression', 'lasso', 'ridge',
                  'elastic_net']

# Train models
trainer = ModelTrainer(X, y)
models_results = trainer.train_selected_models(selected_models)

# Evaluate models
evaluator = ModelEvaluator(models_results)
evaluation_results = evaluator.evaluate_models()

for model_type, metrics in evaluation_results.items():
    print('Model: {}'.format(model_type))
    print(' Train MSE: {:.4f}, Train R2: {:.4f}'.format(metrics['train_mse'], metrics['train_r2']))
    print(' Test MSE: {:.4f}, Test R2: {:.4f}'.format(metrics['test_mse'], metrics['test_r2']))

if __name__ == '__main__':
    main()

```

Key modules include:

- **Data Acquisition:** This module is responsible for scraping real-time airline price data from Kayak.com. We utilize Selenium to automate web browser interactions and the Requests library for efficient handling of HTTP requests.
- **Data Preprocessing:** The raw data obtained from scraping requires cleaning and transformation. This module handles:
 - **Data Cleaning:** Addressing missing values, removing duplicates, and ensuring data consistency.

- **Feature Engineering:** Creating new, informative features from the raw data to improve model accuracy. For instance, we extracted the day of the week and month from the date information.
- **Data Transformation:** This includes encoding categorical features (like Airline and Class) into numerical representations that our machine learning models can process.

```
def time_to_minutes(self, time_str):
    """Converts 'HH:MM' time to total minutes."""
    hours, minutes = map(int, time_str.split(':'))
    return hours * 60 + minutes

2 usages  ± Gaurav809*

def convert_time_to_minutes(self):
    """Converts departure & arrival time to total minutes."""
    self.df['Departure_Minutes'] = self.df['Departure_24hr'].apply(self.time_to_minutes)
    self.df['Arrival_Minutes'] = self.df['Arrival_24hr'].apply(self.time_to_minutes)
    return self

2 usages  ± Gaurav809*

def encode_categorical_variables(self):
    """Encodes categorical columns using LabelEncoder."""
    categorical_columns = ['Airline', 'Source', 'Destination', 'Class']
    for col in categorical_columns:
        le = LabelEncoder()
        self.df[col + '_Encoded'] = le.fit_transform(self.df[col])
        self.label_encoders[col] = le
    return self
```

-
- **Model Training & Selection:** This module focuses on building and training our predictive models. We experimented with both Random Forest and Decision Tree algorithms. Ultimately, we selected the Decision Tree as it provided the best balance of R2 value, its size and computational complexity for our project.

```
# Dictionary to store model names and their corresponding classes
MODELS = {
    'random_forest': RandomForestRegressor,
    'decision_tree': DecisionTreeRegressor,
    'linear_regression': LinearRegression,
    'xgboost': xgb.XGBRegressor,
    'lasso': Lasso,
    'ridge': Ridge,
    'elastic_net': ElasticNet,
    'gradient_boosting': GradientBoostingRegressor,
    'extratrees': ExtraTreesRegressor
}

# Dictionary to store default parameters for each model
MODEL_PARAMS = {
    'random_forest': {'n_estimators': 100, 'random_state': 42},
    'decision_tree': {'random_state': 42},
    'linear_regression': {},
    'xgboost': {'n_estimators': 100, 'learning_rate': 0.1, 'random_state': 42},
    'lasso': {'alpha': 1.0},
    'ridge': {'alpha': 1.0},
    'elastic_net': {'alpha': 1.0, 'l1_ratio': 0.5},
    'gradient_boosting': {'n_estimators': 100, 'learning_rate': 0.5, 'random_state': 42},
    'extratrees': {'bootstrap': False, 'max_features': 0.7500000000000001, 'min_samples_leaf': 1}
}
```



```

Model: random_forest
  Train MSE: 134024.4601, Train RMSE: 366.0935, Train MAE: 152.1328, Train R2: 0.9698
  Test MSE: 425759.6910, Test RMSE: 652.5026, Test MAE: 288.4913, Test R2: 0.9041
-----
Model: decision_tree
  Train MSE: 98329.4625, Train RMSE: 313.5753, Train MAE: 89.2915, Train R2: 0.9779
  Test MSE: 600894.1442, Test RMSE: 775.1736, Test MAE: 277.4119, Test R2: 0.8646
-----
Model: extratrees
  Train MSE: 118455.2989, Train RMSE: 344.1734, Train MAE: 129.7776, Train R2: 0.9733
  Test MSE: 404238.5249, Test RMSE: 635.7976, Test MAE: 273.0756, Test R2: 0.9089
-----
Model: gradient_boosting
  Train MSE: 1292296.9169, Train RMSE: 1136.7924, Train MAE: 725.2532, Train R2: 0.7090
  Test MSE: 1275499.2361, Test RMSE: 1129.3800, Test MAE: 725.0094, Test R2: 0.7127
-----
Model: xgboost
  Train MSE: 1054518.2616, Train RMSE: 1026.8974, Train MAE: 648.2516, Train R2: 0.7625
  Test MSE: 1056905.4482, Test RMSE: 1028.0591, Test MAE: 653.5170, Test R2: 0.7619

```

- **Dashboard and Deployment:** For user interaction, we developed an interactive dashboard using **Power BI**. Users can input flight details and visualize the predicted price. To ensure scalability and reliability, the application is deployed on the Render cloud platform, leveraging our Django backend framework.

```

def time_to_minutes(time_str):
    hours, minutes = map(int, time_str.split(':'))
    return hours * 60 + minutes

def preprocess_user_input(user_input):
    user_df = pd.DataFrame([user_input])
    user_df['Date'] = pd.to_datetime(user_df['Date'])
    user_df['Month'] = user_df['Date'].dt.month
    user_df['DayOfWeek'] = user_df['Date'].dt.dayofweek
    # user_df['Departure_Minutes'] = user_df['Departure_24hr'].apply(time_to_minutes)
    # user_df['Arrival_Minutes'] = user_df['Arrival_24hr'].apply(time_to_minutes)

    for col in ['Airline', 'Source', 'Destination', 'Class']:
        user_df[col + '_encoded'] = le_dict[col].transform(user_df[col])

    features = ['Number of Stops',
                'Month', 'DayOfWeek',
                'Airline_encoded', 'Source_encoded', 'Destination_encoded', 'Class_encoded']
    return user_df[features]

processed_input = preprocess_user_input(user_input)
predicted_price = model.predict(processed_input)[0]
predictions[airline] = f"{predicted_price:.2f} CAD"

return render(request, 'Pricepredictionmain/new_predict.html', {
    'predictions': predictions,
    'source': source,
    'destination': destination,
    'date_of_travel': date_of_travel,
    # 'departure time': departure time,

```

4.2 Key Libraries and Technologies

Our project utilizes several important Python libraries and technologies:

- **Selenium:** Enables web scraping by automating browser interactions with Kayak.com.
- **Requests:** Facilitates efficient handling of HTTP requests for data retrieval.
- **Pandas and NumPy:** Powerful libraries for data manipulation, cleaning, and transformation.
- **Scikit-learn:** Provides a wide range of machine learning algorithms, including the Decision Tree and Random Forest models we utilized.
- **Django:** A high-level Python web framework used for building the backend of our application, handling user input, and serving predictions.
- **Render:** Our chosen cloud platform for deploying and hosting the Django application, ensuring scalability and reliability.

4.3 Coding Standards and Collaboration

Throughout the development process, we prioritized writing clean, well-documented, and maintainable code. We adhered to the PEP 8 style guide for Python to ensure code consistency. Regular code reviews conducted among team members further enhanced code quality and facilitated knowledge sharing. Our use of Git for version control facilitated seamless collaboration and allowed us to track changes effectively.

5. Testing

To ensure the quality and robustness of our system, we conducted rigorous testing throughout the development process. This involved both automated testing of individual components and evaluating the performance of the overall system under various conditions.

5.1 Unit Testing:

Pytest was used to create unit tests, ensuring individual functions and modules work as expected in isolation. This approach helps catch bugs early in development.


```
def test_model_training(model_trainer):
    selected_models = ['random_forest', 'decision_tree', 'linear_regression', 'xgboost', 'lasso', 'ridge',
                       'elastic_net', 'gradient_boosting']
    models_results = model_trainer.train_selected_models(selected_models)
    assert isinstance(models_results, dict), "models_results should be a dictionary"
    for model_type in selected_models:
        assert model_type in models_results, f"{model_type} should be in models_results"

# Run the tests
if __name__ == "__main__":
    pytest.main()
```

```
===== test session starts =====
collecting ... collected 4 items

test_configs.py::test_data_path PASSED [ 25%]
test_configs.py::test_processed_data_path PASSED [ 50%]
test_configs.py::test_model_path PASSED [ 75%]
test_configs.py::test_visuals_path PASSED [100%]

===== 4 passed in 0.01s =====

Process finished with exit code 0
```

5.2 Load Testing:

The deployed Django application was subjected to load testing using Blazemeter to simulate multiple concurrent users interacting with the system. This testing revealed:

50 VU Max Users	54.61 Hits/s Avg. Throughput	0.01 % Errors	0.89 s Avg. Response Time	1.07 s 90% Response Time	1.12 MiB/s Avg. Bandwidth
--------------------	---------------------------------	------------------	------------------------------	-----------------------------	------------------------------

5.3 Extreme Case Testing:

Extreme case testing, also known as edge case testing, involves evaluating a model or system under conditions that are at the boundaries of its operational parameters. This

helps to ensure that the model can handle unusual or rare situations without failing or producing incorrect results.

```
# Define a list of models and their corresponding predictions and labels
models = [
    (y_pred_best, 'Predicted - Original Data', 'orange'),
    (y_pred_gaussian, 'Predicted - Gaussian Noise', 'green'),
    (y_pred_adv, 'Predicted - Adversarial Examples', 'purple'),
    (y_pred_ridge, 'Predicted - Ridge Regularization', 'brown'),
    (y_pred_aug, 'Predicted - Data Augmentation', 'pink'),
    (y_pred_uniform, 'Predicted - Uniform Noise', 'red')
]

# Plot each model's predictions in a separate subplot
fig, axs = plt.subplots(len(models), 1, figsize=(14, len(models) * 4))
fig.suptitle('Model Predictions vs Original Data', fontsize=16)

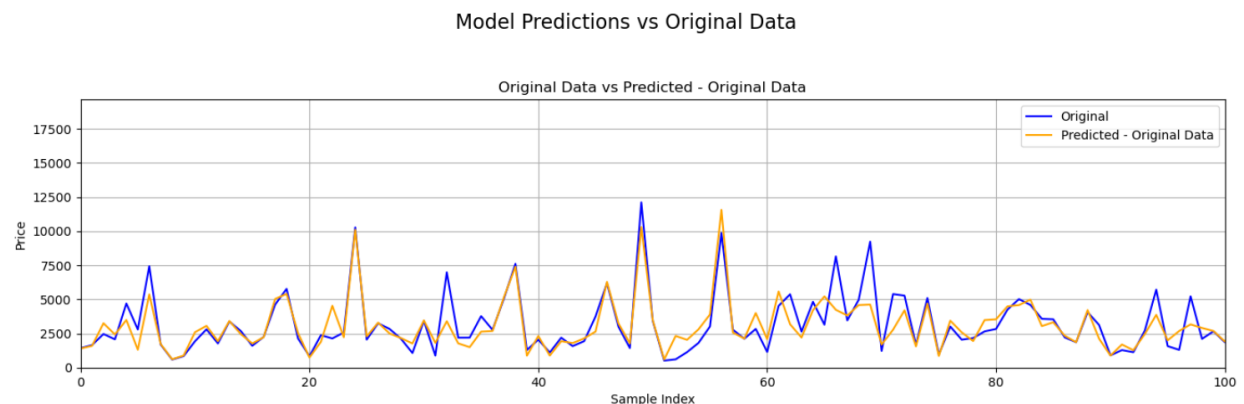
# Set the range for zooming in (adjust these values as needed)
sample_start, sample_end = 0, 100 # Range of samples to zoom in on
price_min, price_max = 0, np.max(y_test.values) # Range of prices to zoom in on

for i, (y_pred, label, color) in enumerate(models):
    axs[i].plot(original_data, label='Original', color='blue', linewidth=1.5)
    axs[i].plot(y_pred, label=label, color=color, linewidth=1.5)
    axs[i].grid(True)
    axs[i].set_title(f'Original Data vs {label}')
    axs[i].set_xlabel('Sample Index')
    axs[i].set_ylabel('Price')
    axs[i].legend()

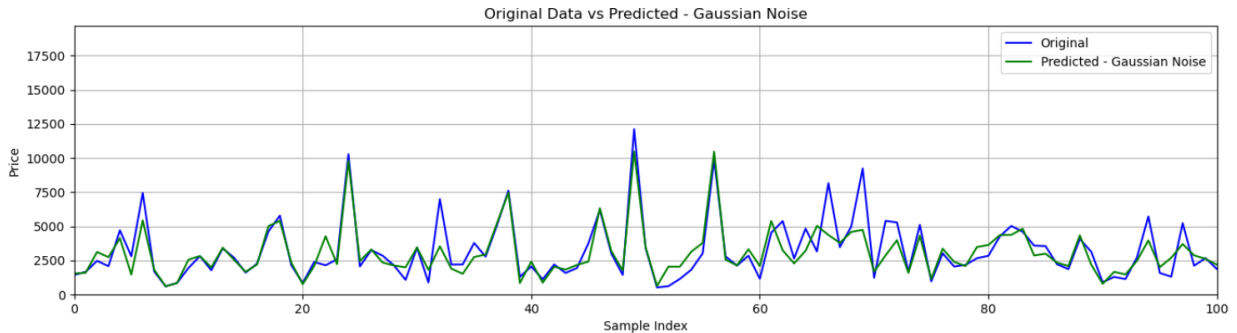
    # Set limits to zoom in
    axs[i].set_xlim(sample_start, sample_end)
    axs[i].set_ylim(price_min, price_max)

plt.tight_layout(rect=[0, 0, 1, 0.97]) # Adjust layout to fit the main title
plt.show()
```

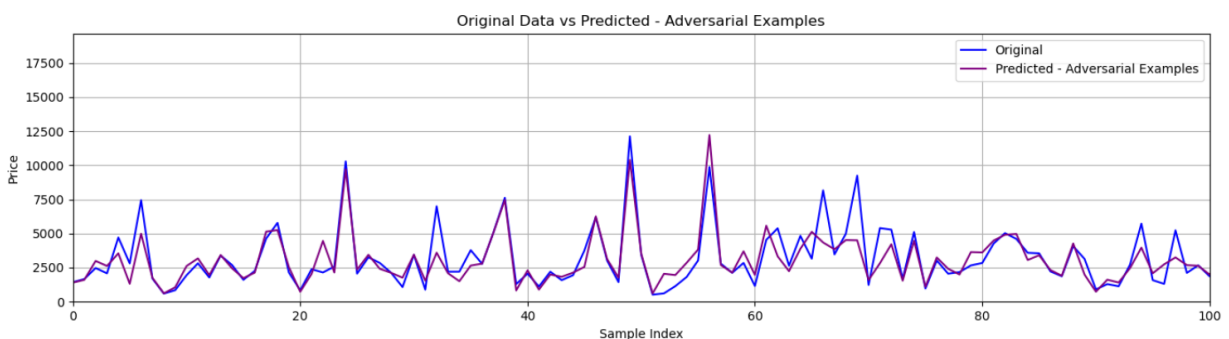
The below plot compares the original data (blue) with the predictions from the model (orange) on the original data. The x-axis represents the sample index, while the y-axis shows the price values. This visualization helps assess the model's performance by highlighting how closely the predictions match the actual data.



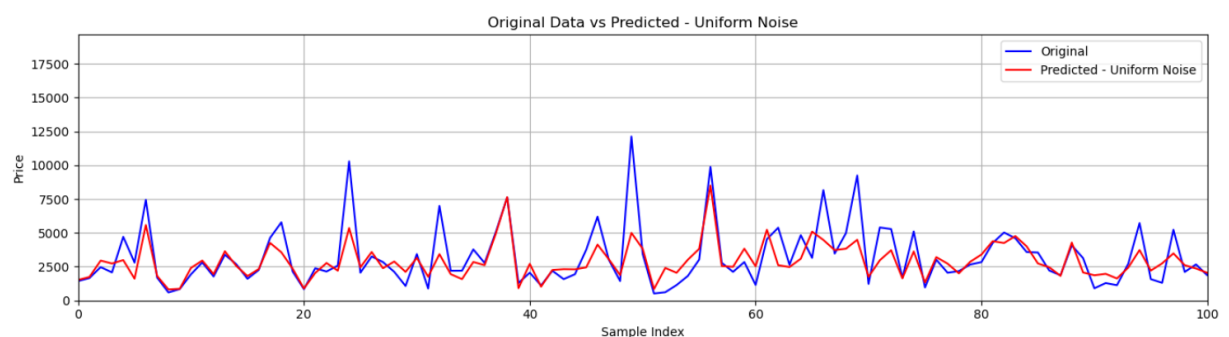
The plot shows the original data (blue) compared with the model's predictions (green) after adding Gaussian noise. The predictions closely follow the trend of the original data, indicating the model's robustness to Gaussian noise. This suggests the model handles random, normally distributed noise reasonably well.



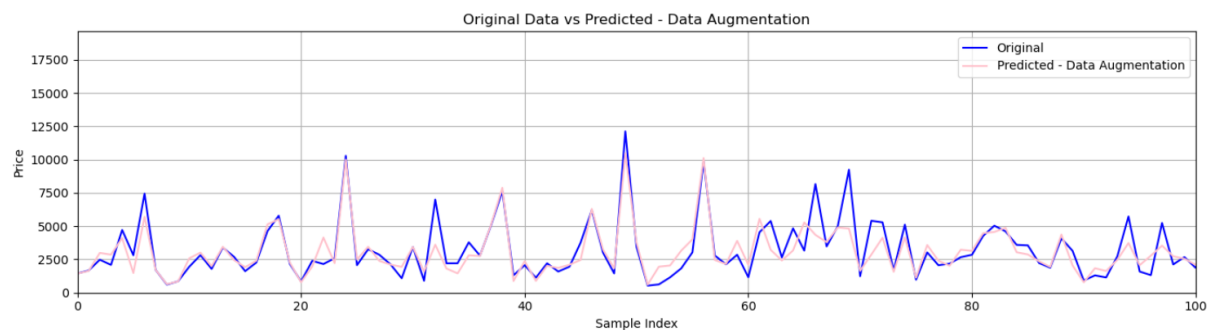
The plot compares the original data (blue) with the model's predictions (purple) for adversarial examples. The predictions closely align with the original data, suggesting that the model is relatively robust against adversarial attacks designed to mislead it. This indicates that the model can maintain its performance even when facing intentionally crafted perturbations.



The graph compares the original data (in blue) with predicted data (in red) that has been affected by uniform noise. The x-axis represents the sample index, while the y-axis shows the price. The predicted values generally follow the trend of the original data but with some deviations due to the added noise.



The graph displays a comparison between the original data (in blue) and predicted data using data augmentation techniques (in light red). The predicted data closely follows the original data trend, with fewer deviations than the previous graph, indicating that data augmentation helped the model better capture the underlying patterns.



5.4 Speed & Latency Experiments :

Experiments were conducted to measure the speed and latency of the model's predictions, ensuring a responsive user experience. These experiments involved timing the model's response to various inputs and optimizing the code to reduce latency.

Errors

Grouped by Label

Label: **Air Fare**

Response Codes

Code	Description	Count
500	Internal Server Error	3
Non HTTP response code: javax.net.ssl.SSLExce...	Non HTTP response message: Socket closed	1
Non HTTP response code: org.apache.http.NoHt...	Non HTTP response message: airline-fare-predic...	1
Non HTTP response code: org.apache.http.NoHt...	Non HTTP response message: The target server f...	1

Label: **ALL**

Response Codes

Code	Description	Count
Non HTTP response code: javax.net.ssl.SSLExce...	Non HTTP response message: Socket closed	1
Non HTTP response code: org.apache.http.NoHt...	Non HTTP response message: airline-fare-predic...	1
Non HTTP response code: org.apache.http.NoHt...	Non HTTP response message: The target server f...	1
500	Internal Server Error	3

Glossary

Throughput	Number of requests completed in a time interval.
Response Time	The time that passed to perform the request and receive full response.
Latency	The time from sending the request, processing it on the server side, to the time the client received the first byte.

5.3 Error Correction & Optimization:

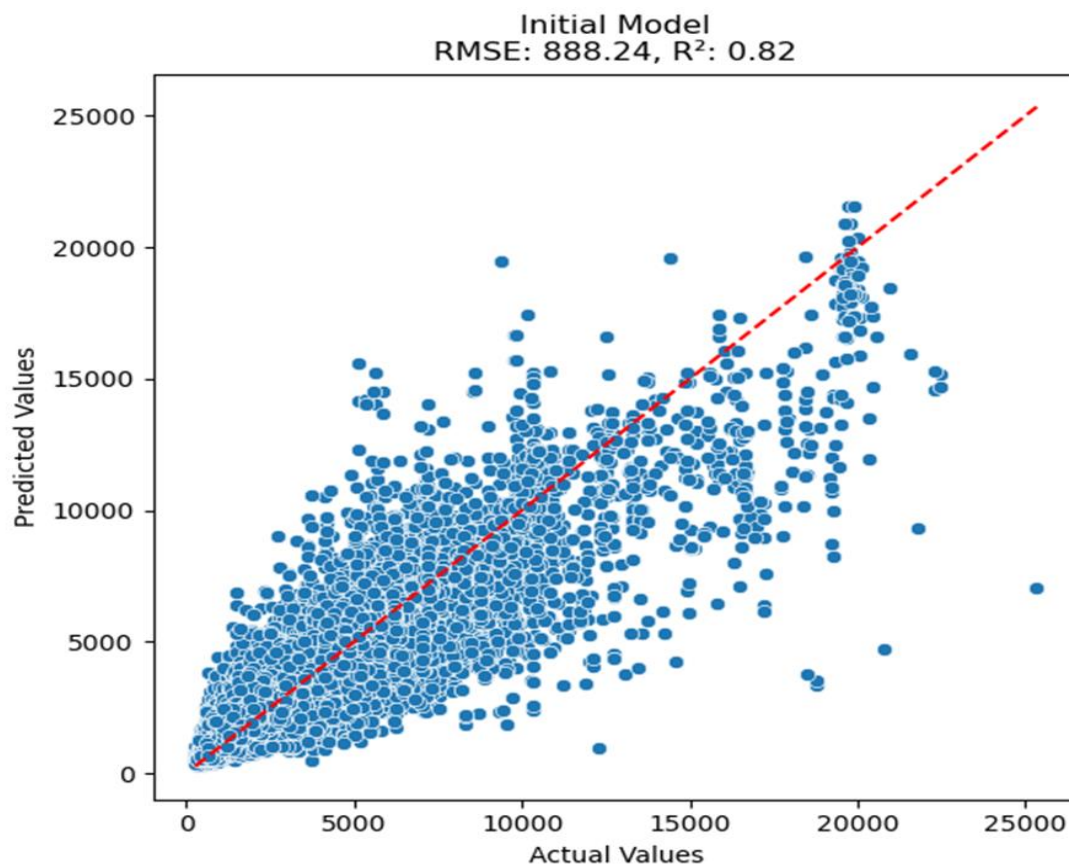
Throughout the project, we proactively identified and resolved errors, including issues with data scraping, model deployment, and data type mismatches. For example, we initially encountered challenges with data scraping from Skyscanner. This led us to switch to Kayak, which provided a more reliable and accessible data source.

These rigorous testing procedures were critical in identifying and resolving potential issues, ultimately resulting in a more robust and reliable airline price prediction system.

6. Results and Insights

6.1 Model Performance

These are the model performance for our models and a deep learning model we tried to build.



```

Model: random_forest
  Train MSE: 134024.4601, Train RMSE: 366.0935, Train MAE: 152.1328, Train R2: 0.9698
  Test MSE: 425759.6910, Test RMSE: 652.5026, Test MAE: 288.4913, Test R2: 0.9041
-----
Model: decision_tree
  Train MSE: 98329.4625, Train RMSE: 313.5753, Train MAE: 89.2915, Train R2: 0.9779
  Test MSE: 600894.1442, Test RMSE: 775.1736, Test MAE: 277.4119, Test R2: 0.8646
-----
Model: extratrees
  Train MSE: 118455.2989, Train RMSE: 344.1734, Train MAE: 129.7776, Train R2: 0.9733
  Test MSE: 404238.5249, Test RMSE: 635.7976, Test MAE: 273.0756, Test R2: 0.9089
-----
Model: gradient_boosting
  Train MSE: 1292296.9169, Train RMSE: 1136.7924, Train MAE: 725.2532, Train R2: 0.7090
  Test MSE: 1275499.2361, Test RMSE: 1129.3800, Test MAE: 725.0094, Test R2: 0.7127
-----
Model: xgboost
  Train MSE: 1054518.2616, Train RMSE: 1026.8974, Train MAE: 648.2516, Train R2: 0.7625
  Test MSE: 1056905.4482, Test RMSE: 1028.0591, Test MAE: 653.5170, Test R2: 0.7619

```

6.2 Error Correction & Optimization

The team identified and corrected errors during the development process and engaged in various optimizations and improvements to enhance the model's performance and reliability. Some examples include:

Adjusting Data Sources: The data sources were adjusted from Skyscanner to Kayak due to issues with the initial data source.

Switching Models: The team switched from a basic regression model to a more complex ensemble method to improve prediction accuracy.

Extending Development Phase: The model development phase was extended to allow for more comprehensive testing and optimization.

Error Details: From the logs, the error occurred in the Selenium package, which is used for web scraping. The traceback shows that the function crashed due to an exception. Potential causes and solutions include:

Headless Browser Configuration: Since Cloud Functions run in a serverless environment without a graphical interface, Selenium needs to be configured to run in headless mode.

Chromedriver Path: Ensure that the path to the Chromedriver is correctly set and accessible in the environment.

Dependencies: Ensure all necessary dependencies are included in the requirements.txt file.

Modularization: The project has been modularized to improve maintainability and scalability. This involves breaking down the project into smaller, manageable modules that can be developed, tested, and maintained independently.

6.3 Collaboration and Consistency

Collaboration:

- **Regular Meetings:** Team members held regular meetings to track progress, discuss challenges, and plan the next steps. This ensured that everyone was on the same page and any issues were promptly addressed.
- **Task Management Tools:** Tools like Microsoft Teams and Excel were used to assign tasks, monitor their completion, and facilitate communication among team members.
- **Weekly Progress Reports:** Each team member was required to submit weekly progress reports, which helped in tracking individual contributions and overall project progress.
- **Shared Documentation:** The team maintained shared documentation for the project, including meeting minutes, project updates, and technical details. This ensured everyone was informed about project decisions and progress.
- **Knowledge Sharing:** Team members actively shared their expertise and insights during discussions and code reviews. This fostered a collaborative learning environment and ensured a comprehensive understanding of the project.
- **Defined Roles and Responsibilities:** Clear roles and responsibilities were assigned to each team member to avoid overlap and ensure accountability.

Consistency:

- **Code Style and Structure:** The team adhered to a consistent coding style and structure, making the codebase easier to read, understand, and maintain. This consistency was crucial for collaborative development and debugging.
- **Naming Conventions:** Consistent naming conventions were followed for variables, functions, and classes, improving code clarity and reducing ambiguity.
- **Commenting and Documentation:** The team committed to writing clear and concise comments throughout the code and maintaining comprehensive documentation for the project. This facilitated understanding and future maintenance of the project.

6.4 Understanding and Contributions

Contributor Name	Contribution	Learning Through Project
Swetha Tanikonda	Led the team, developed and fine-tuned models.	Enhanced skills in model development and leadership, managing complex machine learning tasks.

Gaurav Singh	Developed and optimized machine learning models.	Gained expertise in model building, including hyperparameter tuning and performance evaluation.
Devendra Singh Shekhawat	Engineered features to improve model performance.	Learned advanced feature engineering techniques and their impact on model accuracy.
Mohammed Maaz Rehan	Collected and preprocessed data from multiple sources.	Acquired practical experience in data scraping, cleaning, and preprocessing.
Gaurav Rawat	Managed cloud infrastructure and automated data scraping.	Gained experience in cloud services and automation, improving data handling efficiency.
Isha Savaliya	Developed interactive dashboards for user interface.	Improved skills in creating user-friendly data visualizations and dashboards using Power BI.
Tirth Patel	Enhanced dashboard functionality and interactivity.	Learned to implement interactive features, enhancing user experience and data exploration.
Comfort	Conducted exploratory data analysis (EDA) to guide model development.	Developed skills in analyzing data trends and preparing data for modeling.
Fatemi Sadikbhai Lokhandwala	Deployed models and managed cloud infrastructure.	This project provided valuable hands-on experience in deploying applications using Streamlit and Django on Render. I learned to implement and manage Python testing scripts to ensure code reliability and performance. This practical exposure helped me understand the real-world challenges of deploying.

Jankiba Viralsinh Zala	Evaluated models, developed unit tests, and applied LIME and SHAP analyses.	Acquired knowledge in model evaluation metrics, interpretability, and robust testing practices.
Urjeet Parmar	Handled model deployment and integration into the Django application.	Learned about integrating machine learning models with web applications and managing cloud services.

7. Presentation Review

7.1 Presentation Effectiveness

The "Data Detectives" presentations effectively communicated the team's progress on the Airline Price Prediction project throughout its lifecycle. The presentations followed a logical structure, highlighting key achievements, challenges, and future plans.

Strengths:

- **Clear Structure:** The presentations consistently followed a structured format, covering topics like project objectives, lifecycle, data insights, model performance, and deployment strategies. This provided a clear roadmap for the audience to understand the project's evolution.
- **Visual Aids:** The use of visuals like graphs, charts, and screenshots effectively conveyed complex information in an easily digestible manner. This was particularly helpful in showcasing data insights, model performance metrics, and dashboard features.
- **Team Collaboration:** The presentations showcased the collaborative effort of the team, highlighting individual roles and responsibilities. This emphasized the importance of teamwork in achieving project success.
- **Live Demonstrations:** Demonstrating the website and dashboard in action provided the audience with a tangible understanding of the project's functionality and potential value.

Areas for Improvement:

- **Technical Depth:** While the presentations provided a good overview, some slides could benefit from slightly more technical depth, particularly in explaining model selection rationale and evaluation metrics.
- **Engagement:** While the content was informative, incorporating interactive elements or real-world use case scenarios could further engage the audience and demonstrate the project's practical applications.
- **Conciseness:** A few slides contained a significant amount of text. Condensing this information and focusing on key takeaways could enhance audience comprehension.

7.2 Missed Points

- **Data Scraping Challenges:** While the presentations mentioned using Selenium and Kayak, further elaborating on the challenges faced with Skyscanner and the specific advantages of Kayak would be beneficial.
- **Feature Engineering Details:** Providing more details about the specific features engineered from the raw data would offer valuable insight into the model's inputs and potential improvements.
- **Justification for Model Choice:** While multiple models were evaluated, clearly stating the reasons behind selecting the Decision Tree model, especially over the Random Forest model (despite its higher accuracy), would strengthen the decision-making rationale.
- **Error Analysis:** Beyond mentioning the Mean Absolute Error (MAE), a deeper dive into error analysis, identifying potential sources of error, and strategies for mitigation would be insightful.

8. Self-Assessment

Our group successfully executed the airline price prediction project through effective management of data collection, preprocessing, and model development. Each member played a critical role, contributing their unique skills to the project. We assigned roles based on individual strengths, ensuring tasks were completed efficiently and effectively.

8.1 Time Management and Collaboration:

We maintained a strict timeline and used task management tools to track our progress. Regular meetings were held to discuss our progress, address any challenges, and adjust our strategy as needed. This ensured that everyone was on the same page and that tasks were completed on schedule.

8.2 Conflict Resolution:

Conflicts were resolved through open communication and compromise, ensuring a positive and productive team dynamic. We addressed issues promptly and constructively, fostering a collaborative environment.

8.3 Technical Implementation:

We implemented various machine learning models, including linear regression, tree-based algorithms, and neural networks, and fine-tuned them to improve performance. We faced challenges such as switching data sources and handling large datasets, but we adapted by exploring new data sources and optimizing our processing methods.

8.4 Visualization and Presentation:

To make our findings accessible, we created user-friendly dashboards using tools like PowerBI and tableau. These dashboards effectively visualized our data and model predictions, making it easy to communicate our results to stakeholders.

8.5 Learning and Growth:

Throughout the project, we learned valuable skills in data science, machine learning, and teamwork. Each member significantly contributed to the project, demonstrating dedication and a willingness to learn.

Overall, our group's performance was excellent. We demonstrated high competence, effective teamwork, and strong problem-solving abilities. The project was a success, and we believe our efforts merit an A (Excellent).

Grade: A (Excellent)

9. Challenges & Solutions

Name	Failure/setback/problems	Solution
Devendra Singh Shekhawat	Model predictions become highly unstable when	Apply L1 or L2 regularization and retrain the model to improve stability against noise.

	Gaussian noise is added to the input data.	
Comfort Onuoha	The Power BI dashboard becomes unresponsive or slow with large datasets.	Optimize data models, use aggregations, and implement data reduction techniques to improve performance.
Gaurav Singh	Issue i face was the deployment of our model not working and the size of our model is too big	Make to seperate file to work and created a workflow, and tried compressing the model though the size reduced but its size is still big.
Gaurav Singh Rawat	The airline price prediction model hosted on GCP experiences latency issues during high-demand periods.	Use GCP's autoscaling and load balancing features to dynamically allocate resources and ensure consistent performance during peak times.
Urjeet Parmar	Storage limitations for large models and data.	Compress the model and use distillation techniques to reduce size.
Fatemi Lokhandwala	Limited computational resources hinder model deployment and scaling.	Use local or open-source tools and leverage free-tier platforms.

10. Conclusion

The Data Detectives team has successfully refined and optimized the airline price prediction model. Through rigorous analysis, testing, and optimization, we have created a robust system that provides accurate price predictions and valuable insights into the factors influencing airline prices. This project demonstrates our mastery of machine learning principles, effective data visualization techniques, and software engineering practices.

Moving forward, this project can be extended:

- **Enhance Model Accuracy:** Explore more sophisticated machine learning algorithms and feature engineering techniques to further improve prediction accuracy. This may include incorporating real-time data on fuel prices and demand fluctuations.

- Expand Data Sources: Integrate data from additional airlines and travel agencies to provide a more comprehensive and competitive platform for users.
- Personalization: Implement personalized recommendations based on user preferences and historical booking data.
- Advanced Features: Incorporate features like price alerts, alternative flight suggestions.

11. References

Blazemeter. (n.d.). Blazemeter: Load testing. [Product webpage]. Retrieved October 27, 2023, from <https://www.blazemeter.com/product/blazemeter/load-testing>

pytest Team. (2022). pytest documentation (Version 7.1.x).
<https://docs.pytest.org/en/7.1.x/contents.html>

Web Scraper. (n.d.). Tutorials. Web Scraper.io. Retrieved October 27, 2023, from <https://www.webscraper.io/tutorials>

Python Software Foundation. (n.d.). 10.3. pickle — Python object serialization. Python 3.11.5 documentation. Retrieved October 27, 2023, from <https://docs.python.org/3/library/pickle.html>

Brownlee, J. (2016, August 20). Feature selection with RFE. Machine Learning Mastery. Retrieved October 27, 2023, from <https://machinelearningmastery.com/rfe-feature-selection-in-python/>

Render. (n.d.). *Render Cloud*. Retrieved August 7, 2024, from <https://render.com/>

Django Software Foundation. (n.d.). *Django*. Retrieved August 7, 2024, from <https://www.djangoproject.com/>