COVER SHEET

Unit Name	Software Technology		
Unit Code	4483		
Semester	2	Year	2024
Assessment Name	Assessment 3		

Group Number	1

We declare that this assessment is solely our own work, except where due acknowledgements are made. We acknowledge that the assessor of this assessment may provide a copy of this assessment to another member of the University, and/or to a plagiarism checking service whilst assessing this assessment. We have read and understood the University Policies in respect of Student Academic Honesty.

Student ID		Date
1	u3285396	25/10/2024
2	u3280803	25/10/2024

Github Link

Introduction

By examining past flight data, the Flight Price Predictions system forecasts future flight ticket costs, assisting both passengers and travel agencies in making well-informed choices. Using machine learning algorithms, this Python-based object-oriented system analyses departure and arrival timings, seasonal variations, and airline pricing tactics to find trends in airfare. The system streamlines the management and study of airfare patterns by classifying and organising data into clearly defined types, such as airlines, flights, and pricing models. This enables users to effectively access and handle flight-related data. Future expansions are also supported by the scalable design, which makes it flexible enough to accommodate new features or data sources. The ultimate objective is to offer real-time fare fluctuation updates and personalised recommendations, which will improve the trip planning process, increase user satisfaction, and establish the system as a useful tool in the cutthroat travel sector, allowing for more intelligent and economical travel choices.

Background

The purpose of the Flight Price Predictions system is to address the intricacies of varying airfares, which are impacted by seasonality, demand, and airline pricing policies. This system uses predictive analytics to tell users when prices are likely to increase or decrease, assisting travellers in making the most of their spending. This is in contrast to general fare comparison tools, which just provide basic price lists. This strategy closes a market gap by offering practical advice on how to optimise savings in the erratic travel sector.

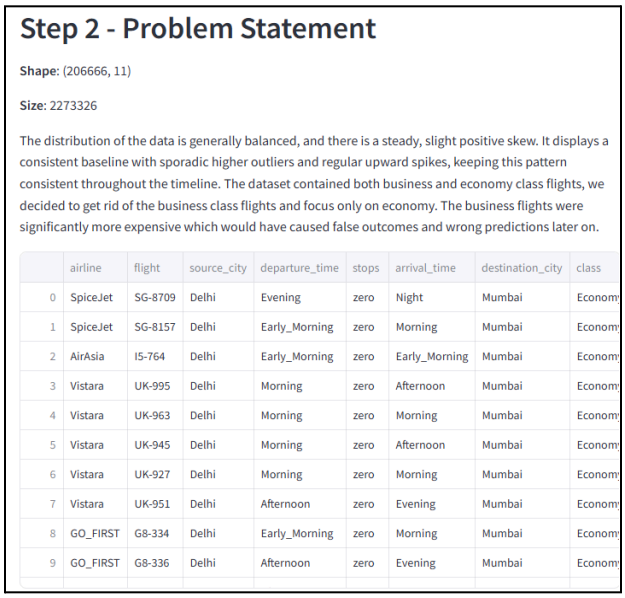
The present approach of manually tracking fares across several websites is time-consuming and might result in missing savings, which is why travellers frequently struggle with it. Price alerts are one example of an existing service that lacks the analytical depth required to detect patterns in real time or make precise predictions. The Flight pricing Predictions system, which focuses on machine learning techniques, distinguishes itself by providing advanced forecasting that uses historical data to predict future pricing fluctuations, giving consumers a distinct edge when it comes to trip planning.

The system has robust analytics, but it also has an easy-to-use interface that even non-technical users may use. Options for personalisation will improve the whole experience by enabling travellers to obtain recommendations that are specifically catered to their tastes and routines. In the end, the system hopes to transform flight booking by providing a more intelligent and effective method of navigating intricate price settings, making it a vital resource for astute tourists seeking to save costs and save time.

Problem statement

The problem of inefficient airfare selection, when customers lose out on savings because there aren't any trustworthy forecasting tools, is addressed by the Flight Price Predictions initiative. Existing solutions frequently rely on labour-intensive and irregular manual price tracking and lack real-time data and personalised insights. Without regular fare updates, travellers frequently miss opportunities to purchase cheaper tickets because they find it difficult to analyse flight data. Travel agencies and industry experts are also impacted by this gap since they lack the data-driven insights necessary to make wise judgements.

For scalability, flexibility, and access to libraries like Scikit-learn and Pandas for data processing and machine learning, the system makes use of Python's object-oriented programming (OOP). Modular architecture is supported by OOP concepts, enabling future expansions without requiring significant redesigns. Because the system is organised with classes for essential elements such as airlines, flights, and pricing models, users with different levels of technical ability can utilise it. This method makes trip planning more intelligent and effective by providing a sophisticated yet approachable solution that offers actionable insights.



Step 2 - Problem Statement

Shape: (206666, 11)

Size: 2273326

The distribution of the data is generally balanced, and there is a steady, slight positive skew. It displays a consistent baseline with sporadic higher outliers and regular upward spikes, keeping this pattern consistent throughout the timeline. The dataset contained both business and economy class flights, we decided to get rid of the business class flights and focus only on economy. The business flights were significantly more expensive which would have caused false outcomes and wrong predictions later on.

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy

Design

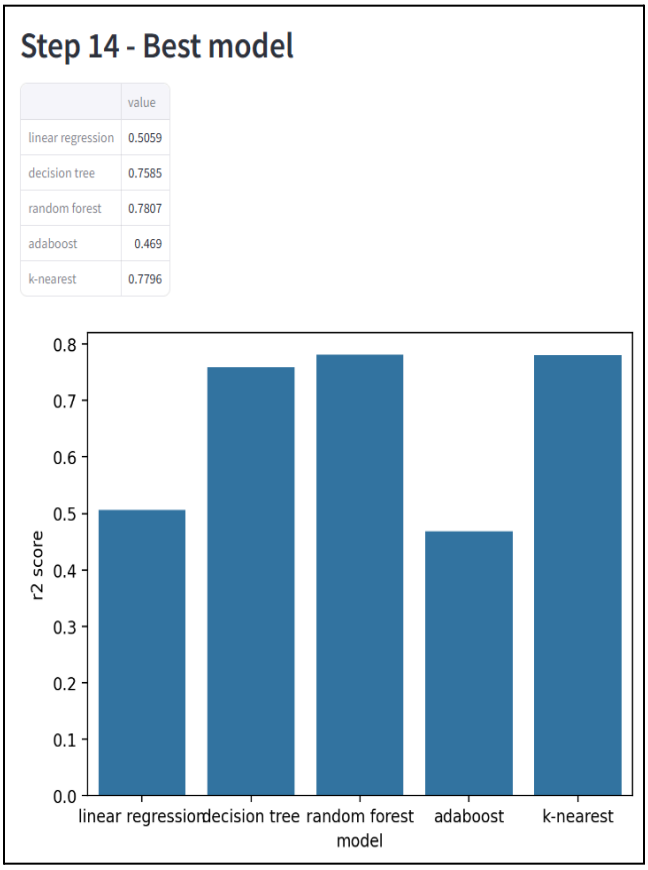
The modular architecture of the system uses components for data processing, model training, and user interaction to forecast flight prices. Important modules include data preprocessing, which scales and encodes raw data for machine learning; model training, which trains and stores several models (like Random Forest and Linear Regression) for comparison; and a Streamlit-created user interface that lets users upload datasets, examine model outcomes, and interactively visualise data insights.

The concepts of object-oriented design are used for encapsulation, modularity, and reusability. For ease of use and future scalability, core methods like train_models() and transform_scaler() handle particular tasks while hiding details from the user. Train_models()'s Factory Pattern adds versatility by managing various model instances. Even if there isn't a database schema in place yet, future integration might store user preferences or processed data, and UML class diagrams could be used to show how modules interact and improve scalability and customisation.

Development

Several Python technologies are used in the system's construction to manage user interaction, machine learning, and data processing. Pandas is a data manipulation tool that makes it possible to load, clean, and change datasets quickly. The machine learning framework is offered by Scikit-learn, which also includes preprocessing tools like StandardScaler and OneHotEncoder to get data ready for modelling, as well as models for predicting flight prices including Linear Regression, Random Forest, and K-Nearest Neighbours.

Streamlit is utilised as the web foundation to provide an interactive, user-friendly experience that lets users interact with visualisations, view forecasts, and upload data. Matplotlib and Seaborn, which offer crucial tools for visual data analysis and model performance comparison, are used to build visualisations. Users can better understand the data and model outputs thanks to these libraries' straightforward, graphical insights.



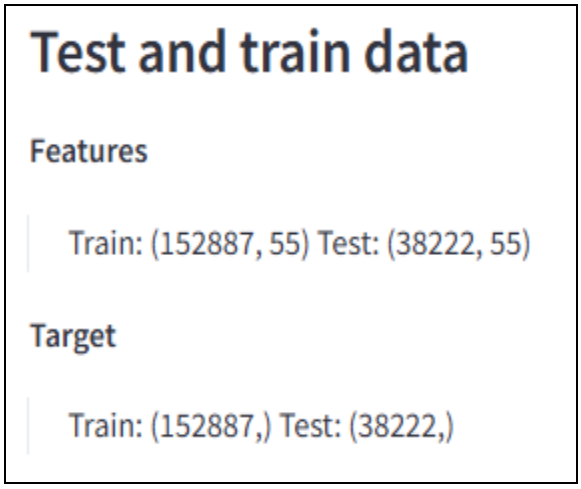
Joblib is used to save and reload trained models for effective model management, enabling model persistence without requiring retraining. The system is further improved with NumPy and SciPy, which provide statistical analysis and array operations. When combined, these technologies produce a robust and expandable flight price prediction system that is modular, strong, and easily accessible.

Testing

To guarantee dependability across all components, the system is tested using unit, integration, and system testing methodologies. Individual functions, like data preprocessing stages (transform_scaler() and transform_ohe()), are the focus of unit testing, which compares expected and actual outcomes to verify accuracy. The smooth operation of different modules, such as data processing and model training, is ensured via integration testing. To make sure the system works as intended for end users, system testing is carried out using the Streamlit interface to validate the entire end-to-end workflow, from data upload to model results.

To cover important features including data transformation, model training, and user interactions inside the Streamlit interface, a number of test cases were developed. For example, data transformation test cases ensure that numerical scaling and categorical encoding are implemented appropriately, while model training tests confirm that multiple models have been successfully trained and produce the desired performance metrics. In order to confirm that the primary components are operating correctly and delivering the anticipated outcomes, these test cases were carried out both manually and automatically. The automated tests were programmed to run following changes.

Any errors or problems were noted and examined during testing. Preprocessing data discrepancies and model evaluation errors brought on by unanticipated data formats were frequent problems. Error messages, log reviews, and inconsistent test case results were used to find these problems. Debugging through logging, adding validation procedures, and improving code to handle different data types were the remedies. Regular testing iterations and debugging techniques contributed to the system's resilience and dependability in a variety of scenarios.

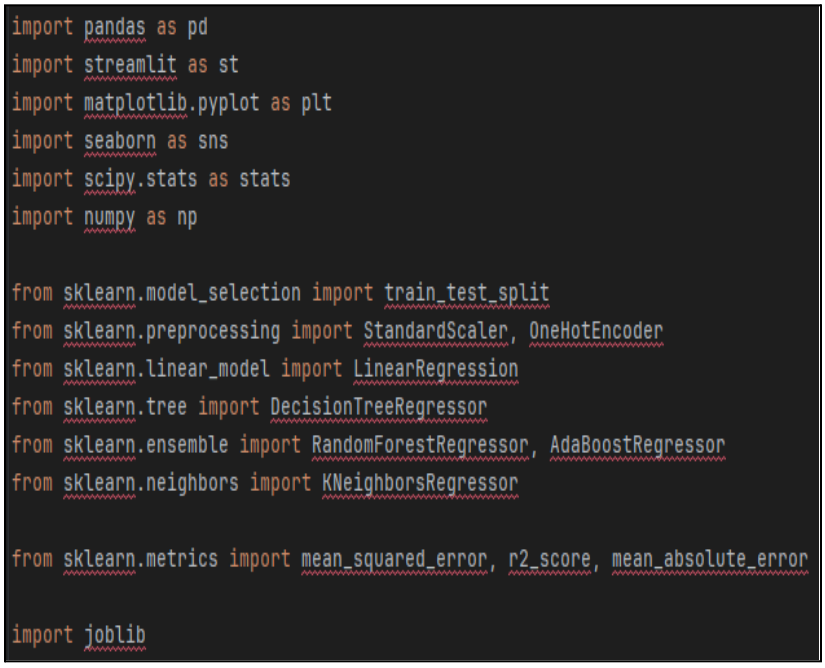


Integration

Data processing, machine learning, visualisation, and user interface components were all brought together by the system's integration. Although each module is made to function separately, they can all be readily integrated into the main Streamlit programme. Data preprocessing, for example, produces structured data that easily enters the machine learning module for model training, and the visualisation module receives the model training

results for analysis. These elements are coordinated by the Streamlit interface, which oversees the entire process from data upload to the presentation of forecasts and visual insights, giving users a seamless workflow.

Managing data flow and format compatibility between modules were among the integration problems, particularly as various functionalities called for distinct data structures or encodings. Aligning the computationally demanding model training and evaluation procedure with the real-time data updates in the Streamlit app presented another difficulty. Strict data validation checks between modules, improved data transformation procedures, and performance-optimised code—especially for model training and visualisation—were used to overcome these obstacles. Testing the system at various integration stages also aided in the early detection and resolution of data incompatibility problems.



```
import pandas as pd
import streamlit as st
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

import joblib
```

Although the system's functionality is mostly dependent on built-in Python libraries, third-party integrations are possible, such as connecting to an external API or database for real-time flight data. Although there aren't any external APIs or services directly integrated at the moment, the system's modular architecture makes it simple to add third-party services in the future for improved features like fare warnings, model changes, or dynamic data retrieval.

References/Bibliography

[1]bayerb, "Pandas dataframe - remove outliers," *Stack Overflow*, Sep. 15, 2017. https://stackoverflow.com/questions/46245035/pandas-dataframe-remove-outliers (accessed Oct. 20, 2024).

[2]"python - Detect and exclude outliers in a pandas DataFrame," *Stack Overflow*. https://stackoverflow.com/questions/23199796/detect-and-exclude-outliers-in-a-pandas-dataframe

[3]GeeksforGeeks, "Detect and Remove the Outliers using Python," *GeeksforGeeks*, Feb. 23, 2021. https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python (accessed Oct. 20, 2024).

[4]"Pandas Read CSV," *www.w3schools.com*. https://www.w3schools.com/python/pandas/pandas_csv.asp

[5]"Pandas Getting Started," *www.w3schools.com*. https://www.w3schools.com/python/pandas/pandas_getting_started.asp

[6]R. Bevans, "One-way ANOVA | When and How to Use It (With Examples)," *Scribbr*, Mar. 06, 2020. https://www.scribbr.com/statistics/one-way-anova/

[7]Collective Action, "Pandas: Get Dummies," *Stack Overflow*, Mar. 29, 2016. https://stackoverflow.com/questions/36285155/pandas-get-dummies (accessed Oct. 20, 2024).

[8]"DecisionTreeRegressor," *scikit-learn*, 2024. https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeRegressor.html (accessed Oct. 20, 2024).

[9]https://www.facebook.com/MachineLearningMastery, "Save and Load Machine Learning Models in Python with scikit-learn," *Machine Learning Mastery*, Jun. 07, 2016. https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/

[10]J. Wong, "One-Hot-Encode categorical variables and scale continuous ones simultaneouely," *Stack Overflow*, May 05, 2017. https://stackoverflow.com/questions/43798377/one-hot-encode-categorical-variables-an

d-scale-continuous-ones-simultaneouely (accessed Oct. 20, 2024).

[11]"Flight Price Prediction," *www.kaggle.com*. https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction

[12]Scikit-learn, "scikit-learn: machine learning in Python," *Scikit-learn.org*, 2019. https://scikit-learn.org/stable/

[13]Pandas, "Python Data Analysis Library," *Pydata.org*, 2018. https://pandas.pydata.org/

[14]Matplotlib, "Matplotlib: Python plotting — Matplotlib 3.1.1 documentation," *Matplotlib.org*, 2012. https://matplotlib.org/