Internship Report

on

FLIGHT DELAY PREDICTION USING MACHINE LEARNING

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTHAPURAMU

In Partial Fulfillment of the Requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Submitted By

NANDYALA DASARATHA REDDY - 21691A0528

MADANAPALLE INSTITUTE OF TECHNOLOGY &SCIENCE

(UGC – AUTONOMOUS)

(Affiliated to JNTUA, Ananthapuramu)

(Accredited by NBA, Approved by AICTE, New Delhi)

AN ISO 21001:2018 Certified Institution

P. B. No: 14, Angallu, Madanapalle – 517325

2024 - 25

MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE

(UGC – AUTONOMOUS)

Approved by AICTE, New Delhi and Affiliated to JNTUA, Ananthapuramu

www.mits.ac.in www.mits.edu

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the internship work entitled **“Flight Delay Prediction Using Machine Learning”** is a bonafide work carried out by

NANDYALA DASARATHA REDDY - 21691A0528

Submitted in partial fulfillment of the requirements for the award of degree **Bachelor of Technology** in the stream of **Computer Science & Engineering** in **Madanapalle Institute of Technology & Science, Madanapalle,** affiliated to **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu** during the academic year 2024-2025.

Internship Coordinator	**Head of the Department**
Mr. Ch. Hemanand	Dr. M. Sreedevi,
Assistant Professor,	Professor & Head,
Department of CSE	Department of CSE

ACKNOWLEDGEMENT

I sincerely thank the **MANAGEMENT** of **Madanapalle Institute of Technology & Science** for providing excellent infrastructure and lab facilities that helped me complete this Internship.

I sincerely thank **Dr. C. Yuvaraj, M.E., Ph.D., Principal,** for guiding and providing facilities for completing my Internship at **Madanapalle Institute of Technology & Science,** Madanapalle.

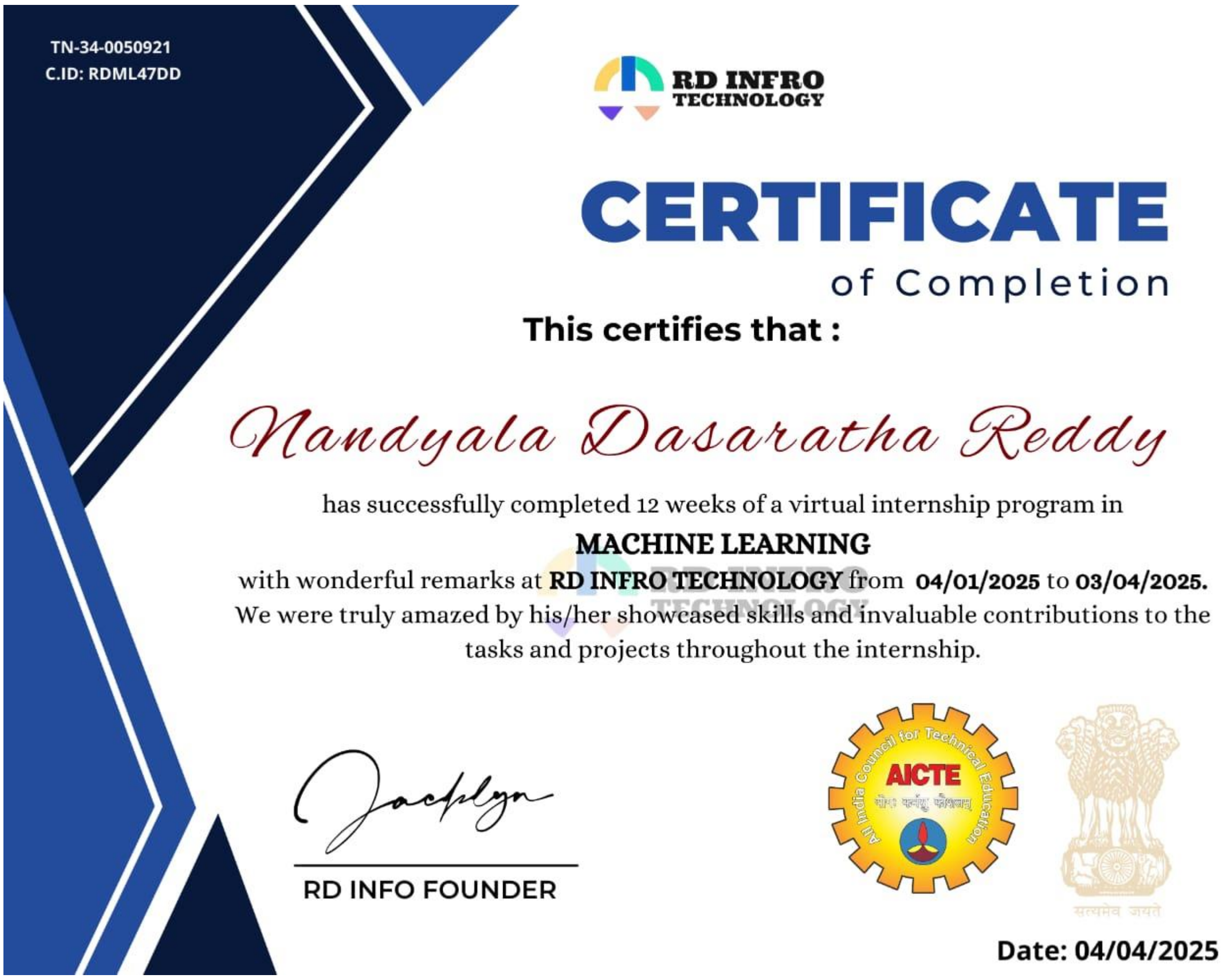
I express my gratitude to **Dr. M. Sreedevi, Ph.D., Professor and Head of the Department, CSE** for her continuous support in making necessary arrangements for the successful completion of the Internship.

I express my sincere thanks to the **Internship Coordinator, Mr. Ch. Hemanand, Assistant Professor, Department of CSE** for his tremendous support for the successful completion of Internship.

I express my deep gratitude to my Internship In-Charge **Dr. K. Sudhakar, Senior Assistant Professor, Department of CSE,** for his guidance and encouragement that helped me to complete this Internship.

I also wish to place on record my gratefulness to other **Faculty of CSE Department** and my friends and my parents for their help and cooperation during my project work.

CERTIFICATE



TN-34-0050921
C.ID: RDML47DD

RD INFRO TECHNOLOGY

CERTIFICATE
of Completion

This certifies that :

Nandyala Dasaratha Reddy

has successfully completed 12 weeks of a virtual internship program in

MACHINE LEARNING

with wonderful remarks at **RD INFRO TECHNOLOGY** from **04/01/2025** to **03/04/2025.** We were truly amazed by his/her showcased skills and invaluable contributions to the tasks and projects throughout the internship.

[illegible]

RD INFO FOUNDER

All India Council for Technical Education
AICTE

सत्यमेव जयते

Date: 04/04/2025

DECLARATION

I hereby declare that the results embodied in this internship **"Flight Delay Prediction Using Machine Learning"** by us under the guidance of **Dr. K. Sudhakar,** in partial fulfillment of the award of **Bachelor of Technology** in **Computer Science & Engineering** from **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu**.

Date :

Place : Angallu, Madanapalle

Student Name & Signature

NANDYALA DASARATHA REDDY

ABSTRACT

Flight delays are a persistent challenge in the aviation industry, causing widespread inconvenience to passengers, operational disruptions for airlines, and financial implications across the ecosystem. With the increasing volume of air travel and the complexity of flight operations, the ability to predict delays has become crucial for improving scheduling efficiency and passenger satisfaction. This project explores the application of machine learning techniques to build a robust flight delay prediction model based on historical flight data.

The dataset used in this study is sourced from Kaggle and includes a wide range of features such as departure and arrival times, airline carriers, origin and destination airports, weather conditions, and other operational variables. Significant data preprocessing steps were carried out, including the handling of missing values, encoding categorical features, normalization, and feature selection to ensure the quality and reliability of the data fed into the model.

To achieve high predictive performance, a Random Forest Classifier was implemented, due to its robustness against overfitting and its effectiveness in handling both categorical and numerical data. The model was evaluated using standard classification metrics including accuracy, precision, recall, and F1 score. The results demonstrated that the Random Forest approach is capable of making accurate predictions, thus serving as a valuable tool for stakeholders in the aviation industry.

This system can be utilized by airlines, airport authorities, and travelers to proactively manage delays by making informed decisions based on predicted outcomes. The insights derived from the model can aid in optimizing flight schedules, improving resource allocation, and enhancing passenger experience. Future improvements could include integrating real-time data sources and testing other ensemble learning techniques to further improve model accuracy.

Keywords: Flight Delay Prediction, Machine Learning, Random Forest Classifier, Classification Metrics, Aviation Analytics, Predictive Modelling, Data Pre-processing.

CONTENTS

S.NO.	TOPIC	PAGE NO.
1	**INTRODUCTION**	
	1.1 About Machine Learning	1
	1.2 Importance and Applications of Machine Learning	2
	1.3 Language used	4
	1.4 Need for the Model	5
2	**TOOLS AND TECHNIQUES**	
	2.1 Platform Used	6
	2.2 Hardware Requirements	6
	2.3 Software Requirements	7
3	**PROJECT WORK**	
	3.1 Project overview	9
	3.2 Algorithm	13
4	**CODE AND OUTPUT SCREENSHOTS**	15
5	**CONCLUSION**	31
	REFERENCES	33

LIST OF FIGURES

Fig.No	Description	PAGE NO.
1.1	Machine Learning Model	1
1.2	Types of Machine Learning	1
1.3	Applications of Machine Learning	3
2.1	Google Colab with Python	8
3.1	Process of Evaluation	13

CHAPTER-1
INTRODUCTION

1.1 ABOUT MACHINE LEARNING

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that enables machines to learn from data and experiences without being explicitly programmed. It identifies patterns and makes predictions with minimal human intervention.

ML algorithms learn directly from data through an iterative process, rather than following fixed programming rules. As more data becomes available, the model's performance improves automatically. Deep learning, a subfield of ML, allows computers to imitate human behavior by learning from large datasets, often achieving better results than traditional ML techniques.

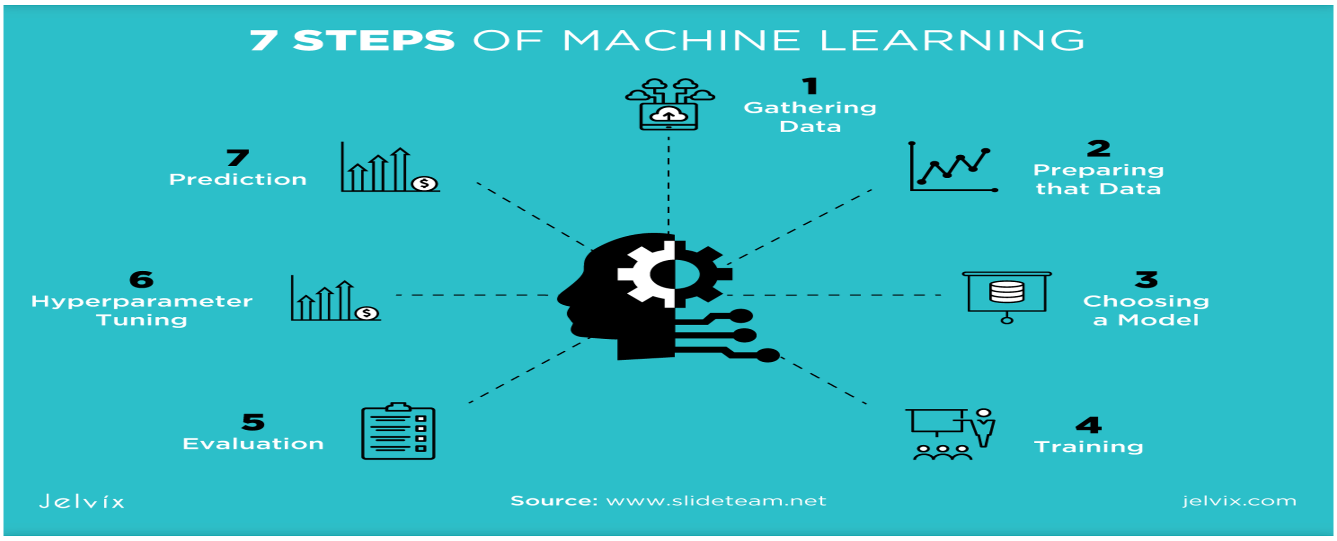


Fig. 1.1 Machine Learning Model

1.1.1 Types of Machine Learning

Machine learning algorithms can be trained in many ways, with each method having its pros and cons. Based on these methods and ways of learning, machine learning is broadly categorized into four main types:

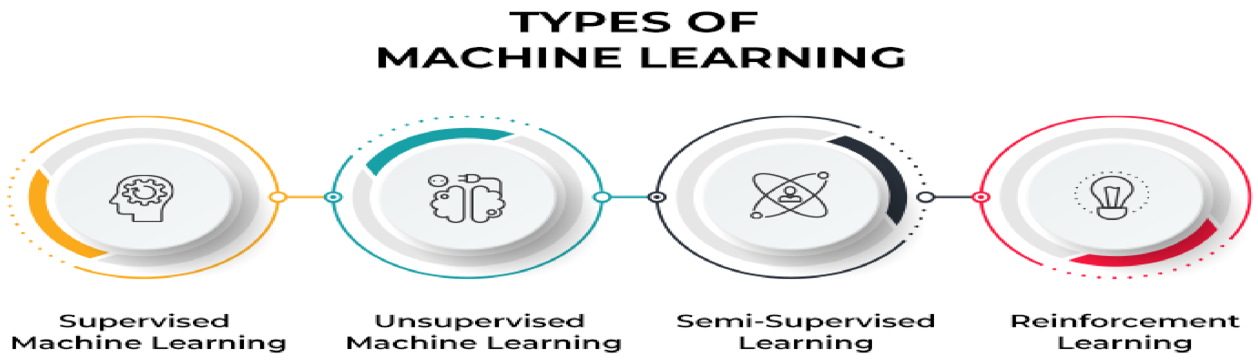

TYPES OF MACHINE LEARNING
Supervised Machine Learning
Unsupervised Machine Learning
Semi-Supervised Learning
Reinforcement Learning

Fig. 1.2 Types of Machine Learning

1. **Supervised Machine Learning**: The machine is trained on labeled datasets (input-output pairs) to predict outcomes for new data. It is classified into:
    - **Classification**: For categorical outputs (e.g., spam detection, email filtering). Algorithms include Random Forest, Decision Tree, SVM.
    - **Regression**: For continuous outputs (e.g., weather prediction, market trend analysis). Algorithms include Linear Regression, Decision Tree, and Lasso Regression.
2. **Unsupervised Machine Learning**: The machine works with unlabeled data to identify patterns, similarities, and groupings:
    - **Clustering**: Grouping similar objects (e.g., customer segmentation). Algorithms include K-Means, DBSCAN, PCA.
    - **Association**: Finding relationships between variables (e.g., market basket analysis). Algorithms include Apriori, Eclat, FP-Growth.
3. **Semi-Supervised Learning**: Combines labelled and unlabelled data to train models, improving upon supervised and unsupervised learning by using both types of data.
4. **Reinforcement Learning**: The agent learns through trial and error, receiving rewards for good actions and penalties for bad actions. It aims to maximize rewards and improve performance. It includes:
    - **Positive Reinforcement**: Rewarding good behavior to encourage repetition.
    - **Negative Reinforcement**: Strengthening behavior that avoids negative outcomes.

1.2 IMPORTANCE AND APPLICATIONS OF MACHINE LEARNING

Machine learning is vital for empowering data-driven decision-making, as it enables businesses to extract valuable insights from large datasets. By identifying trends, patterns, and relationships within the data, ML helps organizations make well-informed decisions that enhance their operational strategies. This capability is especially beneficial in sectors like healthcare, finance, and marketing, where precision and timely decisions can significantly improve outcomes and provide a competitive edge.

Additionally, machine learning plays a key role in automating repetitive and time-consuming tasks, which helps reduce human error and operational costs. Automation enhances efficiency by allowing machines to handle routine activities, thus freeing up human workers to focus on more critical, strategic tasks. This reduction in manual labor results in improved productivity across various industries, including customer service, logistics, and manufacturing.

Furthermore, machine learning's ability to adapt and improve with new data ensures that systems can stay relevant and effective in dynamic environments. Unlike traditional software systems that require manual updates, ML models continuously refine their performance based on the data they process. This flexibility is crucial in areas like fraud detection and cybersecurity, where systems must be able to respond to new threats or changes in user behavior.

Flight delay prediction using machine learning helps airlines forecast delays by analyzing historical data, weather conditions, and traffic patterns. This allows for better scheduling, resource allocation, and communication with passengers. Ultimately, it improves operational efficiency and customer satisfaction by minimizing uncertainty.

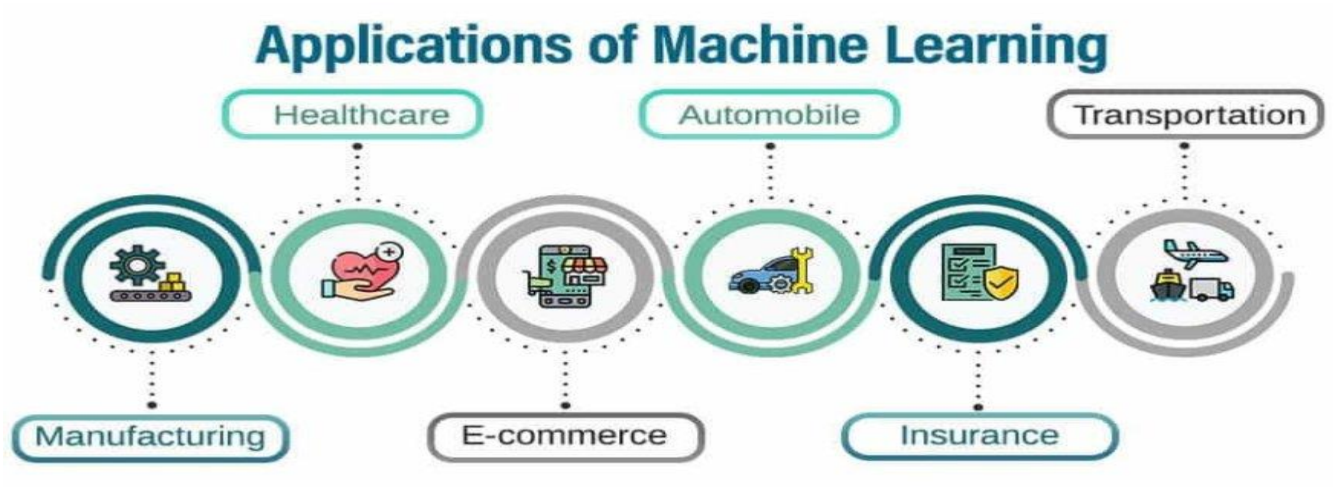


Fig. 1.3 Applications of Machine Learning

Machine learning is significantly enhancing various industries by enabling efficient data processing and decision-making. Here are a few more sectors benefiting from ML:

1. **Healthcare**: In addition to improving diagnosis and drug discovery, ML is revolutionizing personalized healthcare by analyzing genetic data to develop customized treatment plans. It also helps monitor patient health in real-time through wearable devices.
2. **Finance**: Beyond fraud detection, ML is also being used for credit scoring, risk management, and algorithmic trading. These models can analyze financial markets, predict stock movements, and automate trading strategies.
3. **Retail**: Retailers utilize ML for demand forecasting, inventory management, and supply chain optimization. By predicting customer behavior and market trends, ML enhances operational efficiency and profitability.

4. **Travel**: Machine learning optimizes travel routes, forecasts flight delays, and personalizes travel recommendations based on user preferences. It also enhances customer service through chatbots and virtual assistants.
5. **Social Media**: ML algorithms personalize content, optimize ad targeting, and detect harmful content such as hate speech. It plays a significant role in improving user engagement and ensuring a better experience on platforms like Facebook, Instagram, and Twitter.
6. **Manufacturing**: In manufacturing, ML helps optimize production lines, reduce downtime, and predict equipment failures through predictive maintenance. It also aids in quality control by identifying defects in products before they reach customers.
7. **Energy**: ML optimizes energy usage by predicting demand and adjusting supply accordingly. It's also used in smart grids to enhance energy distribution and reduce waste, as well as to improve renewable energy forecasting.
8. **Transportation**: Besides dynamic pricing, ML is used in route optimization, traffic management, and autonomous vehicle development. It helps reduce fuel consumption, improve safety, and enhance overall efficiency in the transportation sector.

1.3 LANGUAGE USED

The primary programming language used in this Flight Delay Prediction project is **Python**. Python is widely recognized for its simplicity and powerful libraries tailored for data analysis, machine learning, and visualization. It served as the backbone of the entire project, from loading and preprocessing the dataset to building, training, and evaluating the predictive models. Libraries such as pandas and numpy were used for data cleaning and transformation, while matplotlib and seaborn aided in visualizing trends, outliers, and feature relationships within the data.

Python's extensive machine learning ecosystem made model development and evaluation efficient and flexible. Various models like Linear Regression, Random Forest Regressor, Boosted Regression, and Multi-Layer Perceptron Regressor were implemented using scikit-learn and xgboost. These libraries also provided robust tools for calculating performance metrics such as MAE, RMSE, R^2, and classification-based metrics like Precision, Recall, and F1 Score. The use of Google Colab, a Python-based Jupyter environment, allowed for easy experimentation, code execution, and visualization within a cloud-based setup.

1.4 NEED FOR THE MODEL

Flight delays are a persistent challenge in the aviation industry, affecting millions of passengers each year and leading to considerable economic losses. These delays can stem from various factors including weather conditions, air traffic congestion, airline operations, and airport logistics. The complexity and interdependence of these variables make it difficult to manually predict delays with accuracy. As air travel continues to grow, there is a critical need for intelligent systems that can forecast delays efficiently and reliably, enabling proactive decision-making for both airlines and passengers.

The implementation of a predictive model using machine learning addresses this need by analyzing historical flight data and uncovering hidden patterns and correlations that might not be evident through traditional analytical methods. These models can take into account a wide range of attributes such as scheduled and actual departure times, day of the week, flight distance, and carrier performance, among others. With such comprehensive input, the model can learn from past behavior and provide timely and accurate predictions about future flight delays.

The benefits of such a system are far-reaching. For airlines, it offers the opportunity to optimize schedules, manage crew and aircraft rotations more effectively, and reduce costs associated with delayed flights. For airports, it can aid in resource allocation such as gate assignments and runway usage. For passengers, having access to early delay predictions can improve travel planning and reduce frustration caused by unforeseen disruptions. Thus, the predictive model becomes a vital tool in enhancing operational efficiency and customer satisfaction.

Furthermore, integrating this model into real-time systems can further boost its impact by allowing continuous monitoring and updating of predictions based on live data inputs. With advancements in data collection and processing, models can become increasingly accurate and responsive to changing conditions. Overall, the need for such predictive models is crucial not only for mitigating the effects of flight delays but also for paving the way toward a smarter and more resilient air travel industry.

CHAPTER – 2
TOOLS AND TECHNIQUES

2.1 PLATFORM USED

The development and implementation of the Flight Delay Prediction model were carried out using **Google Colab**, a cloud-based interactive environment provided by Google for Python programming. Google Colab offers the advantages of high computational resources, seamless integration with Google Drive, and built-in support for popular Python libraries, making it an ideal platform for machine learning experimentation and development. Its collaborative nature also allows for easy sharing and real-time updates, which is beneficial for team-based projects or academic submissions.

The primary programming language used was **Python**, owing to its simplicity and the extensive ecosystem of libraries available for data science and machine learning. Libraries such as **Pandas**, **NumPy**, and **Matplotlib** were employed for data manipulation and visualization. For machine learning model building, **Scikit-learn** was the core library, offering a wide range of efficient tools for classification, regression, model evaluation, and preprocessing techniques.

In addition to the software environment, Google Colab's access to GPU and TPU accelerators played a key role in optimizing the performance of the model training process. Although the Random Forest Classifier does not necessarily require GPU acceleration, the scalable environment provided by Colab ensures that larger datasets can be handled with minimal system lag or memory constraints, thus maintaining workflow efficiency.

Overall, the choice of Google Colab and Python as the development platform provided a robust, flexible, and resource-rich environment for building, testing, and evaluating the Flight Delay Prediction model. It allowed for rapid prototyping, clear visualization of results, and streamlined integration of multiple components of the machine learning pipeline.

2.2 HARDWARE REQUIREMNTS

1. Computer System

- Minimum: Dual-core processor with 4 GB RAM
- Recommended: Intel i5 / AMD Ryzen 5 or higher with 8 GB+ RAM
- Google Colab used to offload computational tasks to cloud servers
- Local system only needed to handle basic browser and data operations

2. Display Monitor

- Minimum resolution: 1366 x 768 pixels
- Recommended resolution: Full HD (1920 x 1080) or higher
- Larger screens helped in better visualization of graphs and datasets
- Enhanced productivity while handling multiple tabs or plots

3. Input Devices

- Standard **keyboard** for code development and documentation
- Standard **mouse** for navigation within notebooks and files
- Input devices ensured ease of interaction with Google Colab and tools

4. Internet Connectivity

- Required for accessing **Google Colab**, cloud storage, and online libraries
- Minimum speed: 4 Mbps for stable connectivity
- Essential for real-time code execution, data uploads, and model training
- Continuous connection ensured sessions were not interrupted

2.3 SOFTWARE REQUIREMENTS

1. Operating System

- Windows 10/11
- Linux (Ubuntu 20.04 or later)
- macOS (Monterey or later)
  These operating systems are suitable for running Python-based machine learning environments and support browser-based tools like Google Colab.

2. Python Interpreter

- **Python 3.8 or above**
  Essential for running machine learning code and utilizing modern libraries like scikit-learn, xgboost, matplotlib, etc.

3. Integrated Development Environment (IDE) / Text Editor (Optional)

- **Google Colab** (Primary platform – browser-based, no installation needed)

- **Jupyter Notebook** (For local development if required)
- **VS Code** or **PyCharm** (Optional for local scripting and debugging)

4. Required Python Libraries

- **Pandas** – Data loading and pre-processing
- **NumPy** – Efficient array operations
- **Matplotlib & Seaborn** – Data visualization
- **Scikit-learn** – Model training, evaluation, pre-processing
- **XGBoost** – Gradient boosting regression
- **MLPRegressor** – Multi-layer perceptron for non-linear regression
- **Pickle / Joblib** – Model saving and loading

Fig. 2.1 Google Colab with Python

➢ Upload the Dataset

➢ Import Necessary Libraries

➢ Train the Machine Learning Models

➢ Evaluate and Visualize the Results

CHAPTER – 3

PROJECT WORK

3.1 PROJECT OVERVIEW

In today’s fast-paced world, time is considered extremely valuable, and unexpected flight delays can cause significant inconvenience for passengers, airlines, and airports alike. The purpose of this project is to predict flight delays accurately using machine learning models, helping passengers plan better and avoid missing connections or important events. By analyzing key flight features such as airline, origin and destination airports, scheduled departure and arrival times, and flight distance, the project aims to provide an efficient prediction model for flight arrival delays. Accurate predictions can offer passengers an opportunity to adjust their schedules and minimize the adverse effects caused by delayed flights.

The dataset used for this project was sourced from Kaggle and originally collected by the U.S. Department of Transportation. It contains detailed information on domestic U.S. flights from 2015, including attributes like flight number, departure and arrival times, delays, distance traveled, and various other delay causes. Before building models, extensive data preprocessing was conducted: missing values were handled, time columns were reformatted, unnecessary features were dropped, categorical variables were label-encoded, and the dataset was normalized. A new binary target variable was also created to enable the use of classification models alongside regression.

To approach the prediction task, multiple machine learning models were deployed for both regression and classification. For regression tasks, models like Linear Regression, Random Forest Regressor, and Boosted Linear Regression were implemented. For classification tasks, algorithms like K-Nearest Neighbors, Logistic Regression, and Decision Trees were utilized. Evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score were used for regression models, while Precision, Recall, F1-Score, and Accuracy were used for classification models.

The results indicated that there was a strong correlation between departure delays and arrival delays. When departure delay was included as a feature, models like simple Linear Regression performed remarkably well, achieving low MAE and RMSE values. However, when departure delay was removed from the feature set, the performance of all models deteriorated significantly. Even complex models like MLP Regressor and Random Forest Regressor struggled to predict arrival delays accurately, highlighting the importance of having highly correlated features for effective prediction.

In conclusion, this project successfully demonstrated the challenge and importance of feature selection in predictive modeling, especially in complex systems like air transportation. Simple models like Linear Regression outperformed more complex algorithms when provided with highly informative features. However, without such features, even advanced models failed to provide accurate predictions. Overall, this project highlights the potential of machine learning in improving passenger experience and operational efficiency in the airline industry, while also underlining the importance of careful data preprocessing and feature engineering.

Techniques Used:

1. Data Pre-processing

- **Handling Missing Data**: We handled missing values by using median imputation for numerical columns and mode imputation for categorical columns to preserve data integrity.
- **Feature Transformation**: We extracted temporal features, such as the hour of the day, day of the week, and holiday status, which are known to affect flight delays.
- **Normalization**: Flight delay times were normalized using standard scaling to help improve model convergence and performance.

2. Feature Engineering

- **Target Variable Creation**: A binary classification target variable (delayed or on-time) was created based on flight delay times exceeding 15 minutes.
- **Time-Based Features**: Temporal data like the time of the day and day of the week were engineered to capture peak hours and seasonal variations in flight delays.
- **Correlation Analysis**: We performed correlation analysis to identify which features (e.g., weather, flight distance) had the most significant impact on delays.

3. Regression Models

- **Linear Regression**: We used linear regression as a baseline model to predict the continuous delay times.
- **Random Forest Regressor**: This ensemble method helped capture complex, non-linear relationships between features and delays.
- **XGBoost**: We implemented XGBoost for boosting the performance, leveraging gradient boosting for improved accuracy.

- **MLP Regressor**: A neural network-based model was used to capture complex relationships, especially for large datasets.

Evaluation Metrics: We evaluated the models using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 score to assess prediction accuracy.

4. Classification Models

- **K-Nearest Neighbors (KNN)**: This method classifies based on the nearest neighbors and is effective for simple decision boundaries.
- **Logistic Regression**: We employed logistic regression for binary classification to predict if a flight would be delayed or on-time.
- **Decision Tree Classifier**: A decision tree was used to model delays with clear decision rules, easy to interpret.
- **Support Vector Machines (SVM)**: We used SVM for high-dimensional classification tasks, maximizing margin for better separation between delayed and on-time flights.

Evaluation Metrics: For classification models, we used Precision, Recall, F1-Score, and the confusion matrix to evaluate how well the models classified flight delays.

5. Hyperparameter Tuning

- **Grid Search & Randomized Search**: These techniques were used for hyperparameter optimization, ensuring that each model ran with the best possible parameters for performance.
- **Cross-Validation**: We performed k-fold cross-validation to evaluate model performance across different subsets of data and reduce overfitting.

6. Ensemble Learning

- **Bagging (Random Forest)**: By training multiple decision trees on different subsets of data, we reduced variance and improved model stability.
- **Boosting (XGBoost)**: The XGBoost algorithm, a gradient boosting method, was used to improve the model by correcting errors made in earlier rounds of learning.
- **Stacking**: We combined multiple models to make final predictions, improving accuracy by leveraging the strengths of each model.

7. Model Comparison

- **ROC Curve & AUC**: We plotted the Receiver Operating Characteristic (ROC) curve and calculated the Area Under the Curve (AUC) to compare classification models.
- **Visualization**: We used bar charts and confusion matrices to visualize and compare the performance of different models, ensuring a clear understanding of strengths and weaknesses.

Purpose and Usage:

- **Predict Flight Delays:**
  The primary aim is to build a machine learning model that can predict whether a flight will be delayed based on various factors like departure time, airline, weather conditions, and historical data.
- **Enhance Passenger Experience:**
  By predicting delays early, the system reduces last-minute disruptions and helps passengers manage their travel plans better.
- **Improve Operational Efficiency:**
  Airlines and airport authorities can use these insights to better manage schedules, resources, and crew assignments, leading to smoother operations.
- **Support Decision-Making:**
  Accurate delay predictions assist airlines and airport managers in making proactive decisions, such as re-routing, informing travelers in advance, or adjusting staffing needs.
- **For Airlines:**
  Airlines can use the predictions to plan schedules more effectively, reassign flights if needed, and minimize turnaround times.
- **For Airports:**
  Airports can use delay information to manage runway allocations, gate assignments, and passenger traffic, reducing congestion during peak hours.
- **For Passengers:**
  Passengers can receive early notifications about possible delays through airline apps or SMS, allowing them to reschedule or adjust their travel plans proactively.
- **For Data Analysis:**
  The delay prediction system also serves as a valuable tool for analyzing patterns and causes of flight delays, which can help airlines improve long-term strategies.

- **For Future Development:**
  This project lays the groundwork for integrating real-time data sources like weather updates, live air traffic, and maintenance logs, making future prediction systems even more accurate and reliable.

3.2 ALGORITHM

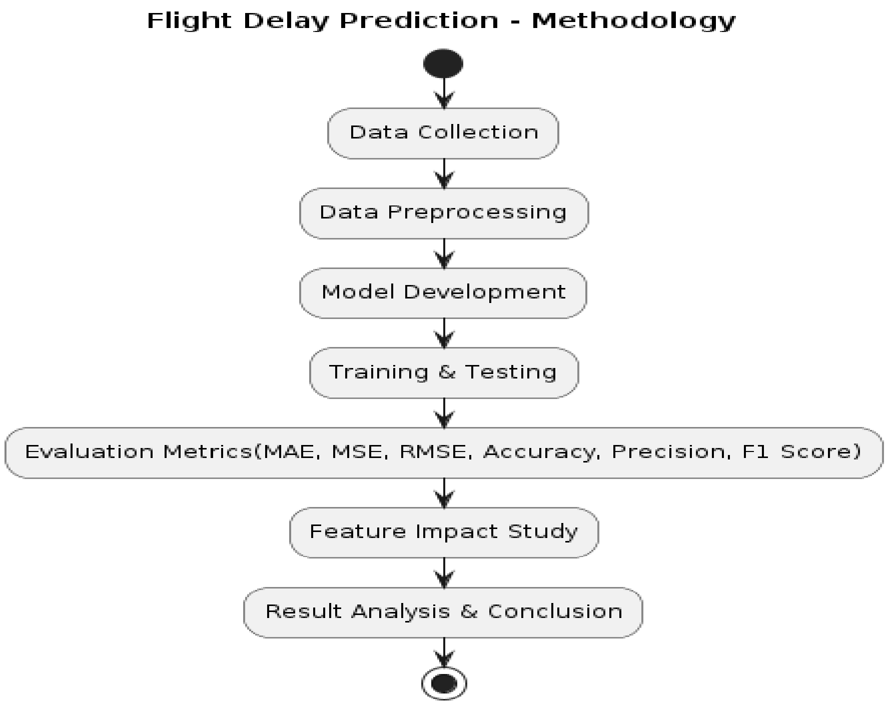

Flight Delay Prediction - Methodology
Data Collection
Data Preprocessing
Model Development
Training & Testing
Evaluation Metrics(MAE, MSE, RMSE, Accuracy, Precision, F1 Score)
Feature Impact Study
Result Analysis & Conclusion

Fig. 3.1 Process of Evaluation

Step 1: Obtain the flight dataset from Kaggle, focusing on U.S. domestic flights with features like departure/arrival times, delays, taxi times, and flight numbers.

Step 2: Remove rows with critical missing values (e.g., departure delay, taxi-out times) to ensure the dataset is clean and ready for analysis.

Step 3: Convert time features into HH:MM format, create new columns (e.g., actual departure/arrival times), and drop irrelevant columns that don't contribute to predicting delays.

Step 4: Convert categorical features (e.g., airline names, airport codes) into numeric format using encoding techniques for machine learning algorithms.

Step 5: Standardize numerical features with StandardScaler to bring all values to a similar scale, improving model performance and convergence speed.

Step 6: Create a binary target variable: label flights with an arrival delay > 0 as '1' (Delayed) and those with no delay as '0' (Not Delayed).

Step 7: Split the dataset into training (80%) and testing (20%) subsets to evaluate model performance on unseen data.

Step 8: Visualize feature relationships with heatmaps to identify highly correlated features like departure delay and arrival delay, which may impact predictions.

Step 9: Train regression models (e.g., Linear Regression, Random Forest Regressor, Boosted Linear Regression) to predict arrival delay, evaluating with MAE, MSE, RMSE, and R^2.

Step 10: Train classification models (e.g., K-Nearest Neighbors, Logistic Regression, Decision Trees) to predict flight delays, using precision, recall, and F1 scores for evaluation.

Step 11: Remove highly correlated features (e.g., Departure Delay) and retrain models to see if performance improves or declines without those features.

Step 12: Compare models trained with and without certain features, assessing the impact on accuracy and overall performance.

Step 13: Compare regression and classification model results, analyzing the best-performing models in different scenarios.

Step 14: Summarize the best models, highlighting the role of feature engineering and selection in improving prediction accuracy.

Step 15: Suggest future improvements like hyperparameter tuning, implementing advanced algorithms (e.g., XGBoost, deep learning), and integrating external data (e.g., weather conditions).

CHAPTER – 4

CODE AND OUTPUT SCREENSHOTS

Code and Output:

```
import datetime, warnings, scipy

import pandas as pd

import numpy as np

import seaborn as sns

 import matplotlib.pyplot as plt

flightsinfo = pd.read_csv("flights.csv")

list(flightsinfo.columns)
```

Output:

['YEAR', 'MONTH', 'DAY', 'DAY_OF_WEEK', 'AIRLINE', 'FLIGHT_NUMBER', 'TAIL_NUMBER', 'ORIGIN_AIRPORT', 'DESTINATION_AIRPORT', 'SCHEDULED_DEPARTURE', 'DEPARTURE_TIME', 'DEPARTURE_DELAY', 'TAXI_OUT', 'WHEELS_OFF', 'SCHEDULED_TIME', 'ELAPSED_TIME', 'AIR_TIME', 'DISTANCE', 'WHEELS_ON', 'TAXI_IN', 'SCHEDULED_ARRIVAL', 'ARRIVAL_TIME', 'ARRIVAL_DELAY', 'DIVERTED', 'CANCELLED', 'CANCELLATION_REASON', 'AIR_SYSTEM_DELAY', 'SECURITY_DELAY', 'AIRLINE_DELAY', 'LATE_AIRCRAFT_DELAY', 'WEATHER_DELAY']

```
airport = pd.read_csv('airports.csv')

airlines = pd.read_csv('airlines.csv')
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5819079 entries, 0 to 5819078
Data columns (total 31 columns):
YEAR                   int64
MONTH                  int64
DAY                    int64
DAY_OF_WEEK            int64
AIRLINE                object
FLIGHT_NUMBER          int64
TAIL_NUMBER            object
ORIGIN_AIRPORT         object
DESTINATION_AIRPORT    object
SCHEDULED_DEPARTURE    int64
DEPARTURE_TIME         float64
DEPARTURE_DELAY        float64
TAXI_OUT               float64
WHEELS_OFF             float64
SCHEDULED_TIME         float64
ELAPSED_TIME           float64
AIR_TIME               float64
DISTANCE               int64
WHEELS_ON              float64
TAXI_IN                float64
SCHEDULED_ARRIVAL      int64
ARRIVAL_TIME           float64
ARRIVAL_DELAY          float64
DIVERTED               int64
CANCELLED              int64
CANCELLATION_REASON    object
AIR_SYSTEM_DELAY       float64
```

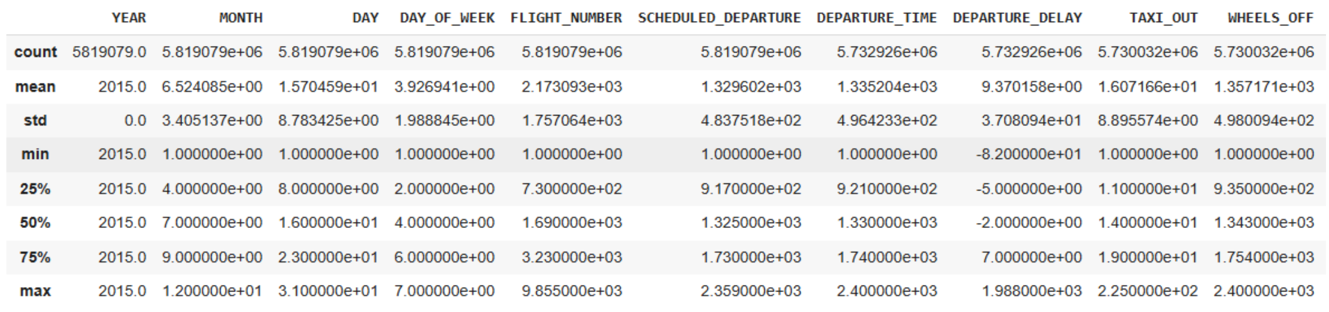
```
SECURITY_DELAY          float64`
AIRLINE_DELAY           float64
LATE_AIRCRAFT_DELAY     float64
WEATHER_DELAY           float64
dtypes: float64(16), int64(10), object(5)
memory usage: 1.3+ GB
```

flightsinfo.shape

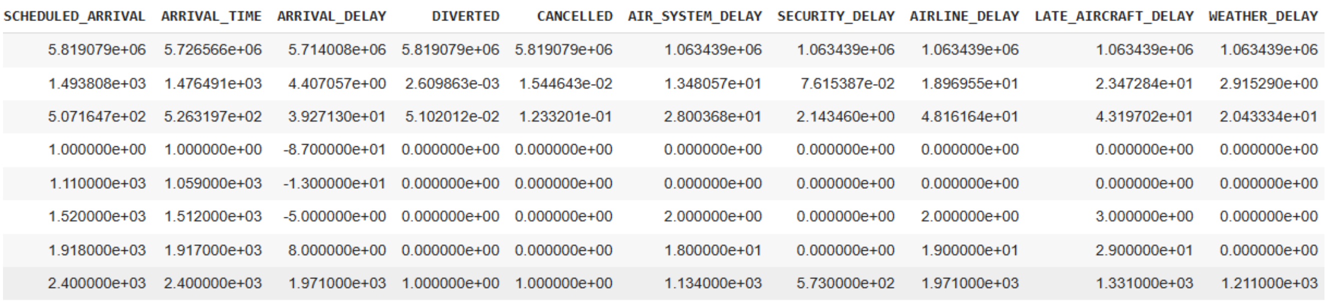
Output: (5819079, 31)

flightsinfo.describe()

Output:



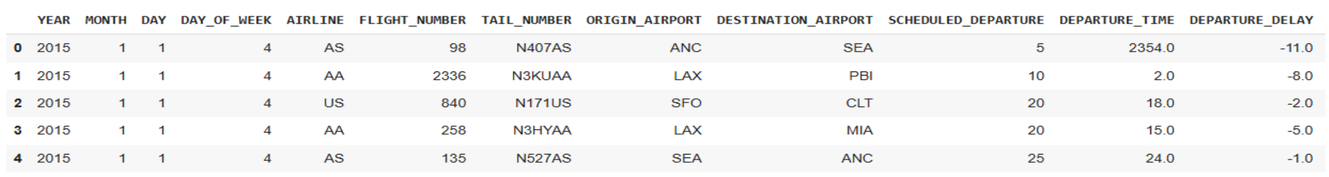
	YEAR	MONTH	DAY	DAY_OF_WEEK	FLIGHT_NUMBER	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY	TAXI_OUT	WHEELS_OFF
count	5819079.0	5.819079e+06	5.819079e+06	5.819079e+06	5.819079e+06	5.819079e+06	5.732926e+06	5.732926e+06	5.730032e+06	5.730032e+06
mean	2015.0	6.524085e+00	1.570459e+01	3.926941e+00	2.173093e+03	1.329602e+03	1.335204e+03	9.370158e+00	1.607166e+01	1.357171e+03
std	0.0	3.405137e+00	8.783425e+00	1.988845e+00	1.757064e+03	4.837518e+02	4.964233e+02	3.708094e+01	8.895574e+00	4.980094e+02
min	2015.0	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	-8.200000e+01	1.000000e+00	1.000000e+00
25%	2015.0	4.000000e+00	8.000000e+00	2.000000e+00	7.300000e+02	9.170000e+02	9.210000e+02	-5.000000e+00	1.100000e+01	9.350000e+02
50%	2015.0	7.000000e+00	1.600000e+01	4.000000e+00	1.690000e+03	1.325000e+03	1.330000e+03	-2.000000e+00	1.400000e+01	1.343000e+03
75%	2015.0	9.000000e+00	2.300000e+01	6.000000e+00	3.230000e+03	1.730000e+03	1.740000e+03	7.000000e+00	1.900000e+01	1.754000e+03
max	2015.0	1.200000e+01	3.100000e+01	7.000000e+00	9.855000e+03	2.359000e+03	2.400000e+03	1.988000e+03	2.250000e+02	2.400000e+03



SCHEDULED_ARRIVAL	ARRIVAL_TIME	ARRIVAL_DELAY	DIVERTED	CANCELLED	AIR_SYSTEM_DELAY	SECURITY_DELAY	AIRLINE_DELAY	LATE_AIRCRAFT_DELAY	WEATHER_DELAY
5.819079e+06	5.726566e+06	5.714008e+06	5.819079e+06	5.819079e+06	1.063439e+06	1.063439e+06	1.063439e+06	1.063439e+06	1.063439e+06
1.493808e+03	1.476491e+03	4.407057e+00	2.609863e-03	1.544643e-02	1.348057e+01	7.615387e-02	1.896955e+01	2.347284e+01	2.915290e+00
5.071647e+02	5.263197e+02	3.927130e+01	5.102012e-02	1.233201e-01	2.800368e+01	2.143460e+00	4.816164e+01	4.319702e+01	2.043334e+01
1.000000e+00	1.000000e+00	-8.700000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
1.110000e+03	1.059000e+03	-1.300000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
1.520000e+03	1.512000e+03	-5.000000e+00	0.000000e+00	0.000000e+00	2.000000e+00	0.000000e+00	2.000000e+00	3.000000e+00	0.000000e+00
1.918000e+03	1.917000e+03	8.000000e+00	0.000000e+00	0.000000e+00	1.800000e+01	0.000000e+00	1.900000e+01	2.900000e+01	0.000000e+00
2.400000e+03	2.400000e+03	1.971000e+03	1.000000e+00	1.000000e+00	1.134000e+03	5.730000e+02	1.971000e+03	1.331000e+03	1.211000e+03

flightsinfo.head()

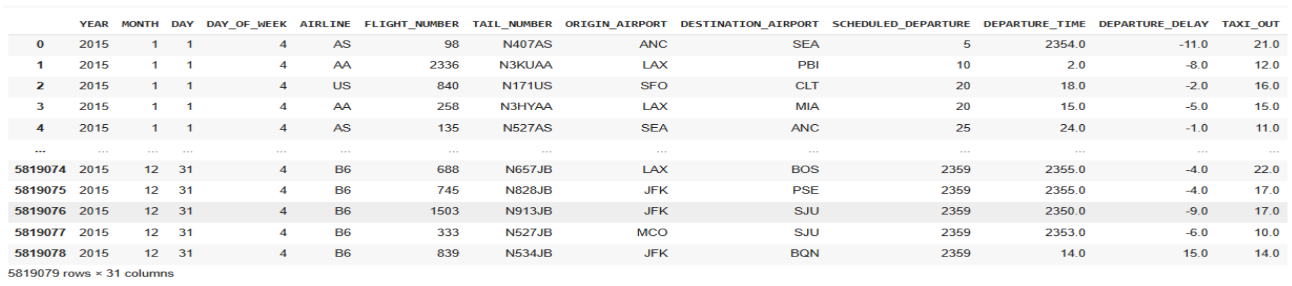
Output:



	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY
0	2015	1	1	4	AS	98	N407AS	ANC	SEA	5	2354.0	-11.0
1	2015	1	1	4	AA	2336	N3KUAA	LAX	PBI	10	2.0	-8.0
2	2015	1	1	4	US	840	N171US	SFO	CLT	20	18.0	-2.0
3	2015	1	1	4	AA	258	N3HYAA	LAX	MIA	20	15.0	-5.0
4	2015	1	1	4	AS	135	N527AS	SEA	ANC	25	24.0	-1.0

Flightsinfo

Output:



	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY	TAXI_OUT
0	2015	1	1	4	AS	98	N407AS	ANC	SEA	5	2354.0	-11.0	21.0
1	2015	1	1	4	AA	2336	N3KUAA	LAX	PBI	10	2.0	-8.0	12.0
2	2015	1	1	4	US	840	N171US	SFO	CLT	20	18.0	-2.0	16.0
3	2015	1	1	4	AA	258	N3HYAA	LAX	MIA	20	15.0	-5.0	15.0
4	2015	1	1	4	AS	135	N527AS	SEA	ANC	25	24.0	-1.0	11.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
5819074	2015	12	31	4	B6	688	N657JB	LAX	BOS	2359	2355.0	-4.0	22.0
5819075	2015	12	31	4	B6	745	N828JB	JFK	PSE	2359	2355.0	-4.0	17.0
5819076	2015	12	31	4	B6	1503	N913JB	JFK	SJU	2359	2350.0	-9.0	17.0
5819077	2015	12	31	4	B6	333	N527JB	MCO	SJU	2359	2353.0	-6.0	10.0
5819078	2015	12	31	4	B6	839	N534JB	JFK	BQN	2359	14.0	15.0	14.0

5819079 rows × 31 columns

airlinecompanies = airlines.set_index('IATA_CODE')['AIRLINE'].to_dict()

airlinecompanies

Output:

```
{'AA': 'American Airlines Inc.',
 'AS': 'Alaska Airlines Inc.',
 'B6': 'JetBlue Airways',
 'DL': 'Delta Air Lines Inc.',
 'EV': 'Atlantic Southeast Airlines',
 'F9': 'Frontier Airlines Inc.',
 'HA': 'Hawaiian Airlines Inc.',
 'MQ': 'American Eagle Airlines Inc.',
 'NK': 'Spirit Air Lines',
 'OO': 'Skywest Airlines Inc.',
 'UA': 'United Air Lines Inc.',
 'US': 'US Airways Inc.',
 'VX': 'Virgin America',
 'WN': 'Southwest Airlines Co.'}
```

```
delay_type = lambda x:((0,1)[x > 5],2)[x > 45]
flightsinfo['DELAY_LEVEL'] = flightsinfo['DEPARTURE_DELAY'].apply(delay_type)
fig = plt.figure(1, figsize=(10,7))
ax = sns.countplot(x="AIRLINE", hue='DELAY_LEVEL', data=flightsinfo, palette= ["#00FF00","#FFA500
" labels = ax.get_xticklabels()

ax.set_xticklabels(labels)

plt.setp(ax.get_yticklabels(), fontsize=12, weight = 'normal', rotation = 0);
plt.setp(ax.get_xticklabels(), fontsize=12, weight = 'normal', rotation = 0);

ax.xaxis.label.set_visible(False)

plt.ylabel('No. of Flights', fontsize=16, weight = 'bold', labelpad=10)

L = plt.legend()

L.get_texts()[0].set_text('on time (t < 5 min)')

L.get_texts()[1].set_text('small delay (5<t<45 min)')

L.get_texts()[2].set_text('large_delay (t>45 min)')

Plt.show()
```

Output:

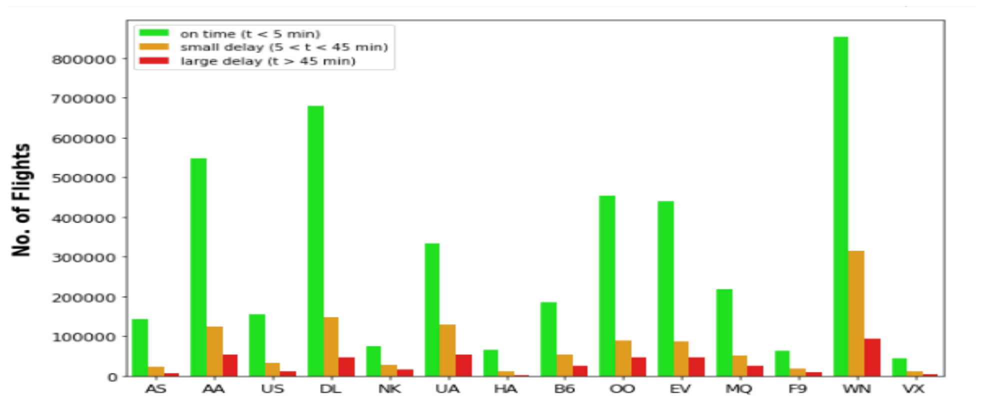


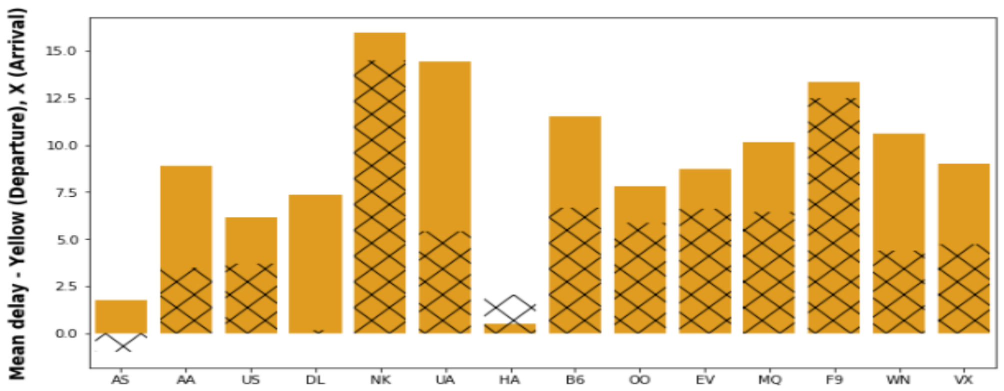
fig = plt.figure(1, figsize=(11,6))
ax = sns.barplot(x="AIRLINE", y="DEPARTURE_DELAY", data=flightsinfo, color="orange", ci=None)
ax = sns.barplot(x="AIRLINE",y="ARRIVAL_DELAY", data=flightsinfo, color="b", hatch = 'X', alpha = 0.0, ci=None)

labels = ax.get_xticklabels() ax.set_xticklabels(labels)

ax.xaxis.label.set_visible(False)

plt.ylabel('Mean delay - Yellow (Departure), X (Arrival)', fontsize=14, weight = 'bold', labelpad=10);

Output:



#!apt-get install libgeos-3.5.0 #!apt-get install libgeos-dev

#!pip install https://github.com/matplotlib/basemap/archive/master.zip

from mpl_toolkits.basemap import Basemap import matplotlib.pyplot as plt

from collections import OrderedDict

flightcount = flightsinfo['ORIGIN_AIRPORT'].value_counts()

plt.figure(figsize=(10,10))

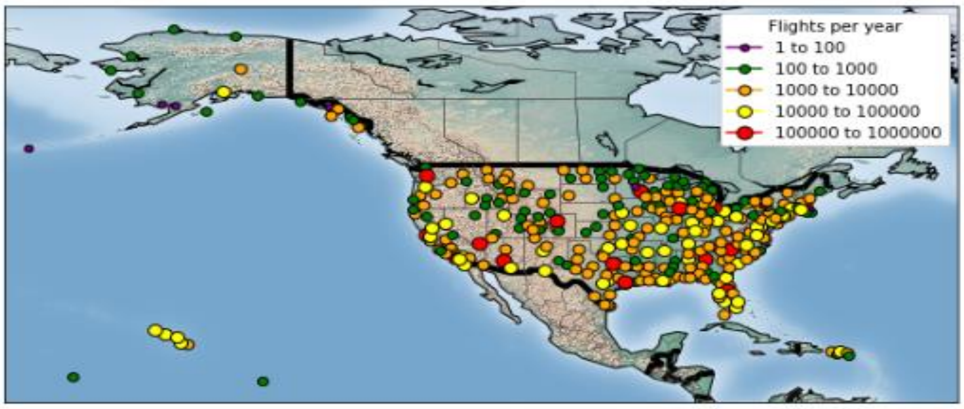
colors = ['purple', 'green', 'orange','yellow', 'red', 'lightblue']

size = [1, 100, 1000, 10000, 100000, 1000000]

labels = ["1 to 100", "100 to 1000", "1000 to 10000", "10000 to 100000", "100000 to 1000000"]
map = Basemap(llcrnrlon=-180, urcrnrlon=-50, llcrnrlat=10, urcrnrlat=75, lat_0=0, lon_0=0,)
map.shadedrelief()

```
map.drawcoastlines()

map.drawcountries(linewidth = 4)

map.drawstates(color='0.3')

for index, (code, y,x) in airport[['IATA_CODE', 'LATITUDE', 'LONGITUDE']].iterrows():

     x, y = map(x, y)     isize = [i for i, val in enumerate(size) if val  < Flightcount[code]]

    ind =isize[-1]

    mat.plot(x,y, marker='o', markersize=ind+5, markeredgewidth=1,color=colors[ind]

handles, labels = plt.gca().get_legend_handles_labels()

labels_ord = OrderedDict(zip(labels, handles))

keys = ('1 to 100','100 to 1000',          '1000 to 10000',          '10000 to 100000',          '100000 to 1000000')
lnew = OrderedDict()

for item in keys:

                                                                  new[item] = labels_ord[item]
plt.legend(lnew.values(), lnew.keys(), loc = 1, prop= {'size':10}, title='Flights per year', frame

plt.show()
```

Output:



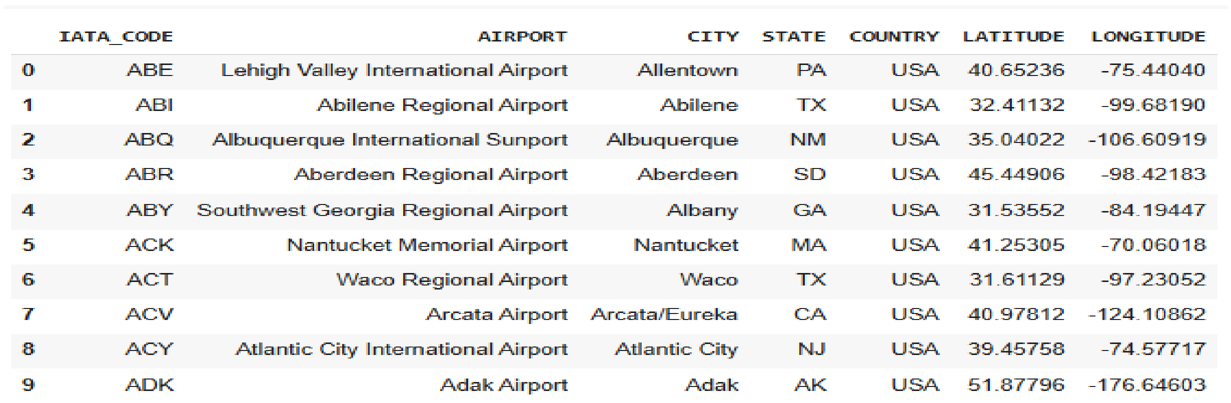
```
airport.isnull().sum()
```

Output:

```
IATA_CODE     0
AIRPORT       0
CITY          0
STATE         0
COUNTRY       0
LATITUDE      3
LONGITUDE     3
dtype: int64
```

```
airport = airport.dropna(subset = ['LATITUDE','LONGITUDE'])
airport.head(10)
```

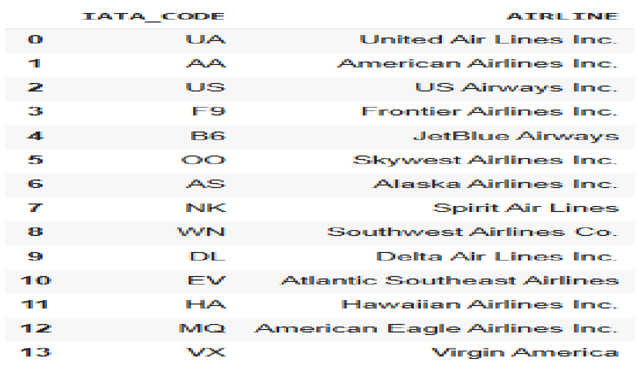
Output:



	IATA_CODE	AIRPORT	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
0	ABE	Lehigh Valley International Airport	Allentown	PA	USA	40.65236	-75.44040
1	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
2	ABQ	Albuquerque International Sunport	Albuquerque	NM	USA	35.04022	-106.60919
3	ABR	Aberdeen Regional Airport	Aberdeen	SD	USA	45.44906	-98.42183
4	ABY	Southwest Georgia Regional Airport	Albany	GA	USA	31.53552	-84.19447
5	ACK	Nantucket Memorial Airport	Nantucket	MA	USA	41.25305	-70.06018
6	ACT	Waco Regional Airport	Waco	TX	USA	31.61129	-97.23052
7	ACV	Arcata Airport	Arcata/Eureka	CA	USA	40.97812	-124.10862
8	ACY	Atlantic City International Airport	Atlantic City	NJ	USA	39.45758	-74.57717
9	ADK	Adak Airport	Adak	AK	USA	51.87796	-176.64603

airlines

Output:



	IATA_CODE	AIRLINE
0	UA	United Air Lines Inc.
1	AA	American Airlines Inc.
2	US	US Airways Inc.
3	F9	Frontier Airlines Inc.
4	B6	JetBlue Airways
5	OO	Skywest Airlines Inc.
6	AS	Alaska Airlines Inc.
7	NK	Spirit Air Lines
8	WN	Southwest Airlines Co.
9	DL	Delta Air Lines Inc.
10	EV	Atlantic Southeast Airlines
11	HA	Hawaiian Airlines Inc.
12	MQ	American Eagle Airlines Inc.
13	VX	Virgin America

flightsinfo_NULL = flightsinfo.isnull().sum()*100/flightsinfo.shape[0]
flightsinfo_NULL

Output:

```
YEAR                    0.000000
MONTH                   0.000000
DAY                     0.000000
DAY_OF_WEEK             0.000000
AIRLINE                 0.000000
FLIGHT_NUMBER           0.000000
TAIL_NUMBER             0.252978
ORIGIN_AIRPORT          0.000000
DESTINATION_AIRPORT     0.000000
SCHEDULED_DEPARTURE     0.000000
DEPARTURE_TIME          1.480526
DEPARTURE_DELAY         1.480526
TAXI_OUT                1.530259
WHEELS_OFF              1.530259
SCHEDULED_TIME          0.000103
ELAPSED_TIME            1.805629
AIR_TIME                1.805629
DISTANCE                0.000000
WHEELS_ON               1.589822
TAXI_IN                 1.589822
SCHEDULED_ARRIVAL       0.000000
ARRIVAL_TIME            1.589822
ARRIVAL_DELAY           1.805629
DIVERTED                0.000000
CANCELLED               0.000000
CANCELLATION_REASON    98.455357
AIR_SYSTEM_DELAY       81.724960
SECURITY_DELAY         81.724960
AIRLINE_DELAY          81.724960
LATE_AIRCRAFT_DELAY    81.724960
WEATHER_DELAY          81.724960
DELAY_LEVEL             0.000000
```

```
dtype: float64
```

I drop all the rows for some attributes which have null values as they make up a small percentage of the dataset

flightsinfo1 = flightsinfo.dropna(subset = ["TAIL_NUMBER",'DEPARTURE_TIME','DEPARTURE_DELAY','TAXI'ELAPSED_TIME','AIR_TIME','WHEELS_ON','TAXI_IN','ARRIVAL_TIME','ARRIVAL_DELAY'])

flightsinfo1.shape

Output: (5714008, 32)

flightsinfo1.isnull().sum()

Output:

```
YEAR                         0
MONTH                        0
DAY                          0
DAY_OF_WEEK                  0
AIRLINE                      0
FLIGHT_NUMBER                0
TAIL_NUMBER                  0
ORIGIN_AIRPORT               0
DESTINATION_AIRPORT          0
SCHEDULED_DEPARTURE          0
DEPARTURE_TIME               0
DEPARTURE_DELAY              0
TAXI_OUT                     0
WHEELS_OFF                   0
SCHEDULED_TIME               0
ELAPSED_TIME                 0
AIR_TIME                     0
DISTANCE                     0
WHEELS_ON                    0
TAXI_IN                      0
SCHEDULED_ARRIVAL            0
ARRIVAL_TIME                 0
ARRIVAL_DELAY                0
DIVERTED                     0
CANCELLED                    0
CANCELLATION_REASON    5714008
AIR_SYSTEM_DELAY       4650569
SECURITY_DELAY         4650569
AIRLINE_DELAY          4650569
LATE_AIRCRAFT_DELAY    4650569
WEATHER_DELAY          4650569
DELAY_LEVEL                  0
dtype: int64
```

flightsinfo_modified.info()

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1063439 entries, 27 to 5819071
Data columns (total 15 columns):
AIRLINE                1063439 non-null object
ORIGIN_AIRPORT         1063439 non-null object
DESTINATION_AIRPORT    1063439 non-null object
DEPARTURE_DELAY        1063439 non-null float64
TAXI_OUT               1063439 non-null float64
ELAPSED_TIME           1063439 non-null float64
DISTANCE               1063439 non-null int64
TAXI_IN                1063439 non-null float64
ARRIVAL_DELAY          1063439 non-null float64
AIR_SYSTEM_DELAY       1063439 non-null float64
SECURITY_DELAY         1063439 non-null float64
AIRLINE_DELAY          1063439 non-null float64
LATE_AIRCRAFT_DELAY    1063439 non-null float64
WEATHER_DELAY          1063439 non-null float64
DELAY_LEVEL            1063439 non-null int64
```

```
dtypes: float64(10), int64(2), object(3)
memory usage: 129.8+ MB
```

Flight_Delays = flightsinfo_modified

Dropping all the contributing factors of delays as we are only going to focus on the over all delay flightsinfo2 = flightsinfo1.drop(['CANCELLATION_REASON','AIR_SYSTEM_DELAY','SECURITY_DELAY','AIRLINE_SYSTEM', 'LATE_AIRCRAFT_DELAY','WEATHER_DELAY'],axis = 1)

flightsinfo2.isnull().sum()

Output:

```
YEAR                   0
MONTH                  0
DAY                    0
DAY_OF_WEEK            0
AIRLINE                0
FLIGHT_NUMBER          0
TAIL_NUMBER            0
ORIGIN_AIRPORT         0
DESTINATION_AIRPORT    0
SCHEDULED_DEPARTURE    0
DEPARTURE_TIME         0
DEPARTURE_DELAY        0
TAXI_OUT               0
WHEELS_OFF             0
SCHEDULED_TIME         0
ELAPSED_TIME           0
AIR_TIME               0
DISTANCE               0
WHEELS_ON              0
TAXI_IN                0
SCHEDULED_ARRIVAL      0
ARRIVAL_TIME           0
ARRIVAL_DELAY          0
DIVERTED               0
CANCELLED              0
DELAY_LEVEL            0
dtype: int64
```

flightsinfo2.shape

Output: (5714008, 26)

flightsinfo2.info()

Output:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5714008 entries, 0 to 5819078
Data columns (total 26 columns):
YEAR                   int64
MONTH                  int64
DAY                    int64
DAY_OF_WEEK            int64
AIRLINE                object
FLIGHT_NUMBER          int64
TAIL_NUMBER            object
ORIGIN_AIRPORT         object
DESTINATION_AIRPORT    object
SCHEDULED_DEPARTURE    int64
DEPARTURE_TIME         float64
DEPARTURE_DELAY        float64
TAXI_OUT               float64
WHEELS_OFF             float64
SCHEDULED_TIME         float64
ELAPSED_TIME           float64
AIR_TIME               float64
DISTANCE               int64
WHEELS_ON              float64
TAXI_IN                float64
SCHEDULED_ARRIVAL      int64
```

```
ARRIVAL_TIME           float64
ARRIVAL_DELAY          float64
DIVERTED               int64
CANCELLED              int64
DELAY_LEVEL            int64
dtypes: float64(11), int64(11), object(4)
memory usage: 1.1+ GB
```

flightsinfo2.DEPARTURE_TIME

Output:

```
0          2354.0
1             2.0
2            18.0
3            15.0
4            24.0
            ...
5819074    2355.0
5819075    2355.0
5819076    2350.0
5819077    2353.0
5819078      14.0
Name: DEPARTURE_TIME, Length: 5714008, dtype: float64
```

flightsinfo2.columns

Output:

```
Index(['YEAR', 'MONTH', 'DAY', 'DAY_OF_WEEK', 'AIRLINE', 'FLIGHT_NUMBER',
'TAIL_NUMBER', 'ORIGIN_AIRPORT', 'DESTINATION_AIRPORT', 'SCHEDULED_DEPARTURE',
'DEPARTURE_TIME', 'DEPARTURE_DELAY',   'TAXI_OUT','WHEELS_OFF', 'SCHEDULED_TIME',
'ELAPSED_TIME', 'AIR_TIME', 'DISTANCE','WHEELS_ON', 'TAXI_IN', 'SCHEDULED_ARRIVAL',
'ARRIVAL_TIME', 'ARRIVAL_DELAY', 'DIVERTED', 'CANCELLED', 'DELAY_LEVEL',
'Actual_Departure'], dtype='object')
```

flightsinfo2['Day'] = flightsinfo2['Date'].dt.weekday_name

flightsinfo2['Actual_Departure'] =flightsinfo1['DEPARTURE_TIME'].apply(CreateTimeFormatted)

flightsinfo2['Scheduled_Arrival'] =flightsinfo1['SCHEDULED_ARRIVAL'].apply(CreateTimeFormatted)

flightsinfo2['Scheduled_Departure'] =flightsinfo1['SCHEDULED_DEPARTURE'].apply(CreateTimeForm)

flightsinfo2['Actual_Arrival'] =flightsinfo2['ARRIVAL_TIME'].apply(CreateTimeFormatted)

flightsinfo2 = flightsinfo2.merge(airlines, left_on='AIRLINE', right_on='IATA_CODE', how='inner')
flightsinfo2 = flightsinfo2.drop(['AIRLINE_x','IATA_CODE'], axis=1)

flightsinfo2 = flightsinfo2.rename(columns={"AIRLINE_y":"AIRLINE"})
flightsinfo2 = flightsinfo2.merge(airport, left_on='ORIGIN_AIRPORT', right_on='IATA_CODE', how='in)

flightsinfo2 = flightsinfo2.merge(airport, left_on='DESTINATION_AIRPORT', right_on='IATA_CODE')

flightsinfo2.columns

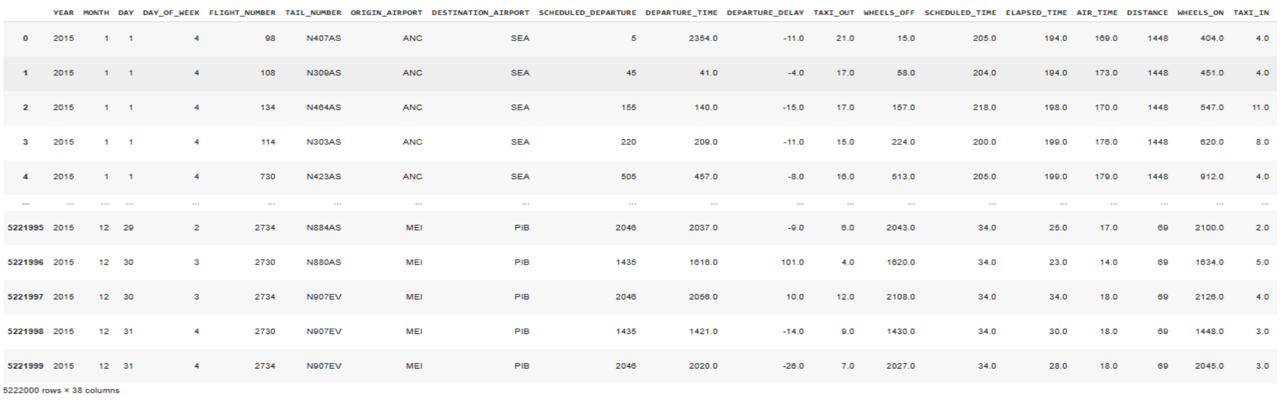
Output:

```
Index(['YEAR',   'MONTH',   'DAY',   'DAY_OF_WEEK',   'FLIGHT_NUMBER',   'TAIL_NUMBER',
'ORIGIN_AIRPORT',   'DESTINATION_AIRPORT',   'SCHEDULED_DEPARTURE',   'DEPARTURE_TIME',
'DEPARTURE_DELAY',   'TAXI_OUT',   'WHEELS_OFF',   'SCHEDULED_TIME',   'ELAPSED_TIME',
'AIR_TIME',  'DISTANCE',  'WHEELS_ON',  'TAXI_IN',  'SCHEDULED_ARRIVAL',  'ARRIVAL_TIME',
'ARRIVAL_DELAY',  'DIVERTED',  'CANCELLED',  'DELAY_LEVEL',  'Actual_Departure',  'Date',
'Day', 'Scheduled_Arrival', 'Scheduled_Departure', 'Actual_Arrival',
'AIRLINE', 'IATA_CODE_x', 'AIRPORT_x', 'CITY_x', 'STATE_x', 'COUNTRY_x',
'LATITUDE_x', 'LONGITUDE_x', 'IATA_CODE_y', 'AIRPORT_y', 'CITY_y',
'STATE_y', 'COUNTRY_y', 'LATITUDE_y', 'LONGITUDE_y'],dtype='object')
```

flightsinfo2 = flightsinfo2.drop(['LATITUDE_x', 'LONGITUDE_x', 'STATE_y', 'COUNTRY_y', 'LATITUDE_y','LONGITUDE_y','STATE_x','COUNTRY_x'], axis=1)
flightsinfo2 = flightsinfo2.rename(columns={'IATA_CODE_x':'Org_Airport_Code','AIRPORT_x':'Org_Air ' IATA_CODE_y':'Dest_Airport_Code','AIRPORT_y':'Dest_Airport_Name','CI }

flightsinfo2

Output:



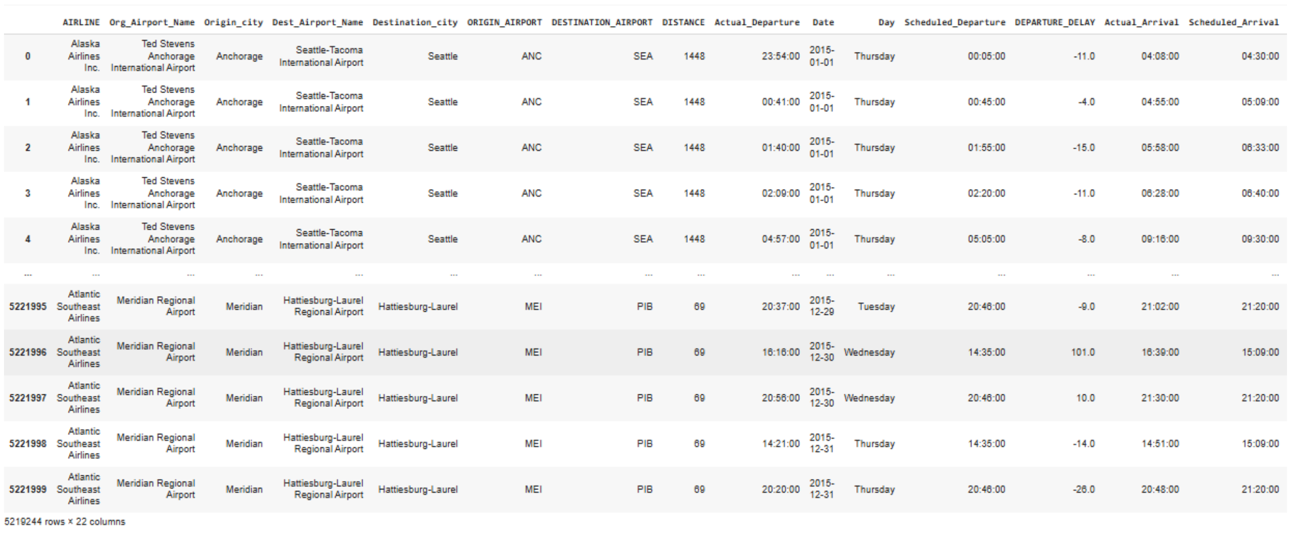
	YEAR	MONTH	DAY	DAY_OF_WEEK	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY	TAXI_OUT	WHEELS_OFF	SCHEDULED_TIME	ELAPSED_TIME	AIR_TIME	DISTANCE	WHEELS_ON	TAXI_IN
0	2015	1	1	4	98	N407AS	ANC	SEA	5	2354.0	-11.0	21.0	15.0	205.0	194.0	169.0	1448	404.0	4.0
1	2015	1	1	4	108	N309AS	ANC	SEA	45	41.0	-4.0	17.0	58.0	204.0	194.0	173.0	1448	451.0	4.0
2	2015	1	1	4	134	N464AS	ANC	SEA	155	140.0	-15.0	17.0	157.0	218.0	198.0	170.0	1448	547.0	11.0
3	2015	1	1	4	114	N303AS	ANC	SEA	220	209.0	-11.0	15.0	224.0	200.0	199.0	176.0	1448	620.0	8.0
4	2015	1	1	4	730	N423AS	ANC	SEA	505	457.0	-8.0	16.0	513.0	205.0	199.0	179.0	1448	912.0	4.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5221995	2015	12	29	2	2734	N884AS	MEI	PIB	2046	2037.0	-9.0	6.0	2043.0	34.0	25.0	17.0	69	2100.0	2.0
5221996	2015	12	30	3	2730	N880AS	MEI	PIB	1435	1616.0	101.0	4.0	1620.0	34.0	23.0	14.0	69	1634.0	5.0
5221997	2015	12	30	3	2734	N907EV	MEI	PIB	2046	2056.0	10.0	12.0	2108.0	34.0	34.0	18.0	69	2126.0	4.0
5221998	2015	12	31	4	2730	N907EV	MEI	PIB	1435	1421.0	-14.0	9.0	1430.0	34.0	30.0	18.0	69	1448.0	3.0
5221999	2015	12	31	4	2734	N907EV	MEI	PIB	2046	2020.0	-26.0	7.0	2027.0	34.0	28.0	18.0	69	2045.0	3.0

5222000 rows × 38 columns

Flights = data_vizual

Flights

Output:



	AIRLINE	Org_Airport_Name	Origin_city	Dest_Airport_Name	Destination_city	ORIGIN_AIRPORT	DESTINATION_AIRPORT	DISTANCE	Actual_Departure	Date	Day	Scheduled_Departure	DEPARTURE_DELAY	Actual_Arrival	Scheduled_Arrival
0	Alaska Airlines Inc.	Ted Stevens Anchorage International Airport	Anchorage	Seattle-Tacoma International Airport	Seattle	ANC	SEA	1448	23:54:00	2015-01-01	Thursday	00:05:00	-11.0	04:08:00	04:30:00
1	Alaska Airlines Inc.	Ted Stevens Anchorage International Airport	Anchorage	Seattle-Tacoma International Airport	Seattle	ANC	SEA	1448	00:41:00	2015-01-01	Thursday	00:45:00	-4.0	04:55:00	05:09:00
2	Alaska Airlines Inc.	Ted Stevens Anchorage International Airport	Anchorage	Seattle-Tacoma International Airport	Seattle	ANC	SEA	1448	01:40:00	2015-01-01	Thursday	01:55:00	-15.0	05:58:00	06:33:00
3	Alaska Airlines Inc.	Ted Stevens Anchorage International Airport	Anchorage	Seattle-Tacoma International Airport	Seattle	ANC	SEA	1448	02:09:00	2015-01-01	Thursday	02:20:00	-11.0	06:28:00	06:40:00
4	Alaska Airlines Inc.	Ted Stevens Anchorage International Airport	Anchorage	Seattle-Tacoma International Airport	Seattle	ANC	SEA	1448	04:57:00	2015-01-01	Thursday	05:05:00	-8.0	09:16:00	09:30:00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5221995	Atlantic Southeast Airlines	Meridian Regional Airport	Meridian	Hattiesburg-Laurel Regional Airport	Hattiesburg-Laurel	MEI	PIB	69	20:37:00	2015-12-29	Tuesday	20:46:00	-9.0	21:02:00	21:20:00
5221996	Atlantic Southeast Airlines	Meridian Regional Airport	Meridian	Hattiesburg-Laurel Regional Airport	Hattiesburg-Laurel	MEI	PIB	69	16:16:00	2015-12-30	Wednesday	14:35:00	101.0	16:39:00	15:09:00
5221997	Atlantic Southeast Airlines	Meridian Regional Airport	Meridian	Hattiesburg-Laurel Regional Airport	Hattiesburg-Laurel	MEI	PIB	69	20:56:00	2015-12-30	Wednesday	20:46:00	10.0	21:30:00	21:20:00
5221998	Atlantic Southeast Airlines	Meridian Regional Airport	Meridian	Hattiesburg-Laurel Regional Airport	Hattiesburg-Laurel	MEI	PIB	69	14:21:00	2015-12-31	Thursday	14:35:00	-14.0	14:51:00	15:09:00
5221999	Atlantic Southeast Airlines	Meridian Regional Airport	Meridian	Hattiesburg-Laurel Regional Airport	Hattiesburg-Laurel	MEI	PIB	69	20:20:00	2015-12-31	Thursday	20:46:00	-26.0	20:48:00	21:20:00

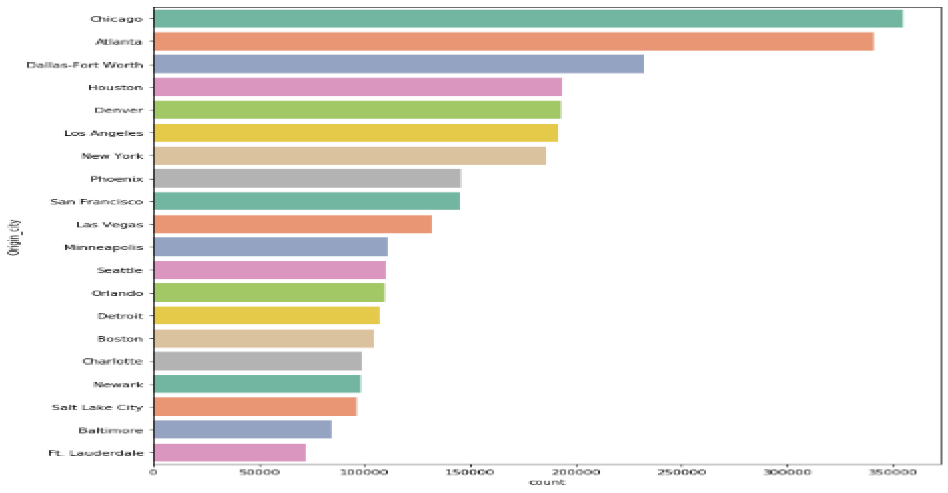
5219244 rows × 22 columns

plt.figure(figsize=(10, 10))

axis.set_yticklabels(axis.get_yticklabels())

plt.tight_layout() plt.show()

Output:

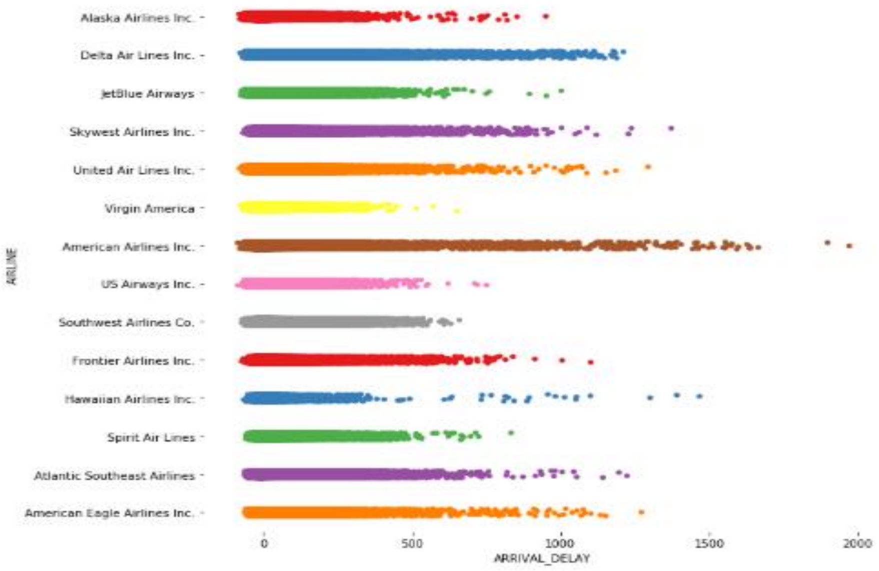


axis = plt.subplots(figsize=(10,10)) sns.despine(bottom=True, left=True)

sns.stripplot(x="ARRIVAL_DELAY", y="AIRLINE",data = Flights, dodge=True, jitter=True,palette="Set1 ”)

plt.show()

Output:



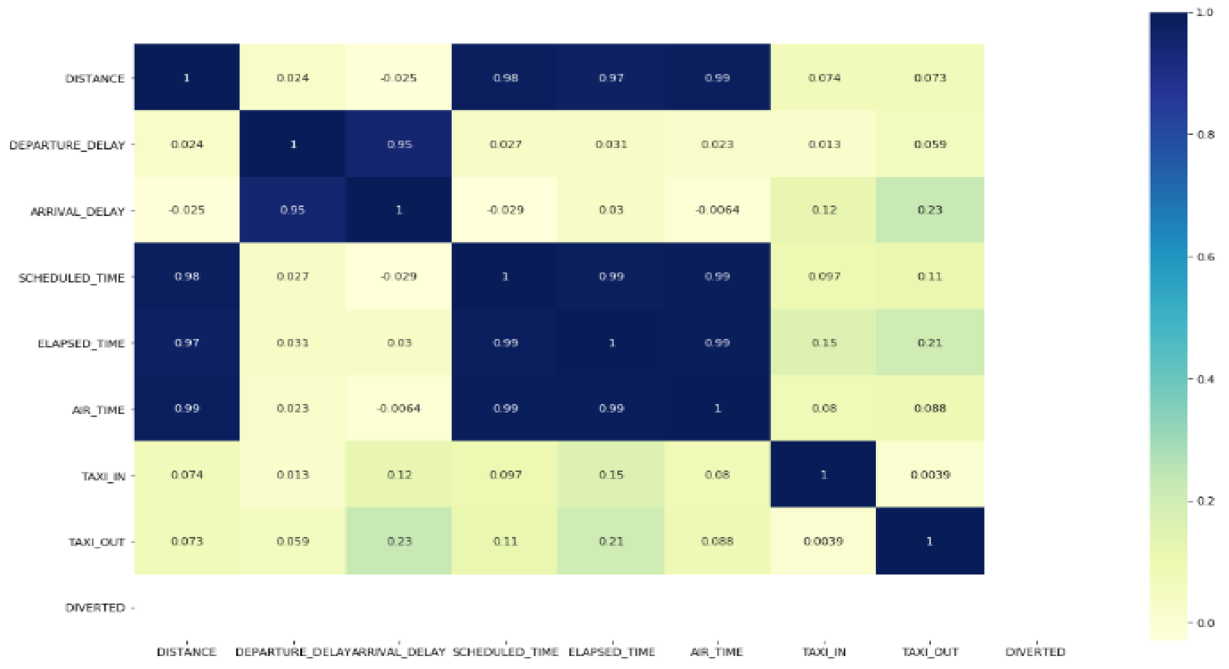
axis = plt.subplots(figsize=(18,12)) sns.heatmap(Flights.corr(),annot = True,cmap="YlGnBu")
b, t = plt.ylim()

# discover the values for bottom and top #b += 0.5 # Add 0.5 to the bottom t -= 0.5

# Subtract 0.5 from the top plt.ylim(b, t) # update the ylim(bottom, top) values

plt.show()

Output:



Removing columns that are not needed fo prediction
Flights1 = Flights.drop(['Org_Airport_Name','Origin_city','Dest_Airport_Name','Destination_city'],
Flights1.columns

Output:

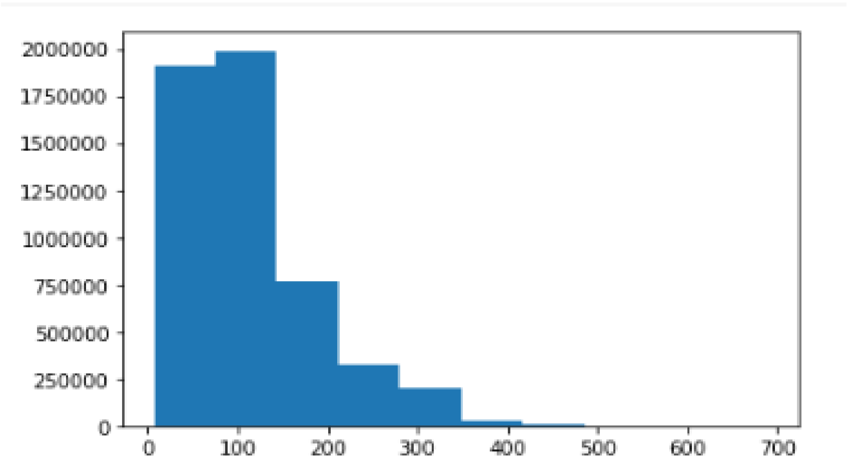
```
Index(['AIRLINE', 'ORIGIN_AIRPORT', 'DESTINATION_AIRPORT', 'DISTANCE',
       'Actual_Departure', 'Date', 'Day', 'Scheduled_Departure',
       'DEPARTURE_DELAY', 'Actual_Arrival', 'Scheduled_Arrival',
       'ARRIVAL_DELAY', 'SCHEDULED_TIME', 'ELAPSED_TIME', 'AIR_TIME',
       'TAXI_IN', 'TAXI_OUT', 'DIVERTED'],
      dtype='object')
```

Air Time distribution histogram

plt.hist(Flights1['AIR_TIME'])

plt.show()

Output:

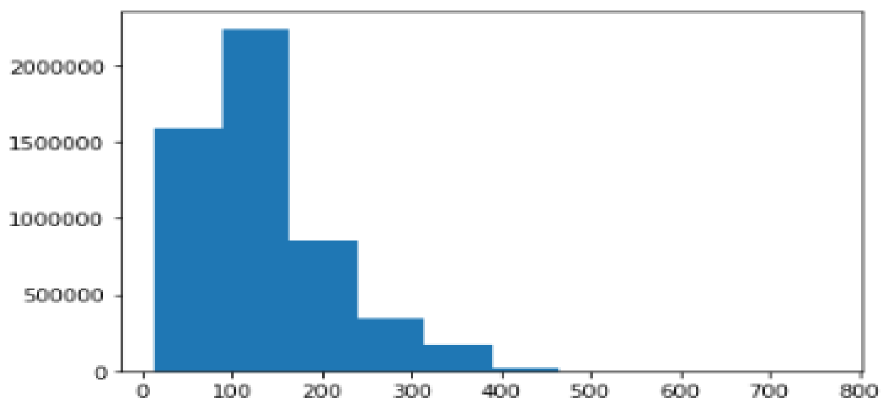


Elapsed Time distribution histogram

plt.hist(Flights1['ELAPSED_TIME'])

plt.show()

Output:



Applying Linear Regression

from sklearn.linear_model

import LinearRegression

LinR = LinearRegression()

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

fitResult = LinR.fit(X_train_sc,y_train) y_pred = fitResult.predict(X_test_sc)

print ('MAE:' , mean_absolute_error(y_test, y_pred))

print ('MSE:' , mean_squared_error(y_test, y_pred))

print('RMSE:' , np.sqrt(mean_squared_error(y_test, y_pred)))

print ('R2:' , r2_score(y_test, y_pred))

Output:

```
MAE: 1.5327891237063537e-06
MSE: 3.0655780798908132e-06
RMSE: 0.0017508792305269982
R2: 0.9999999980588673
```

import matplotlib.pyplot as plt

fig, ax = plt.subplots() ax.scatter(y_test, y_pred)

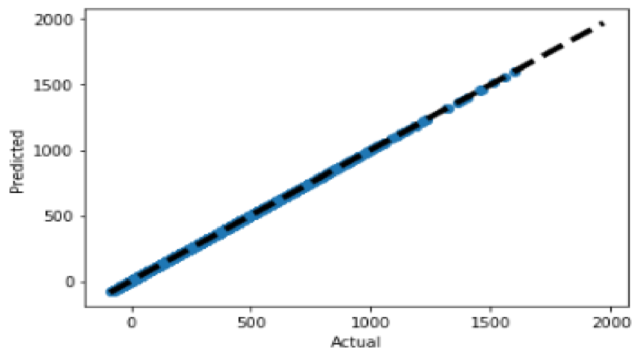
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)

ax.set_xlabel('Actual')

ax.set_ylabel('Predicted')

plt.show()

Output:



Applying Random Forest Regressor

from sklearn.ensemble import RandomForestRegressor

Rfc = RandomForestRegressor(random_state=2)

fitResultR = Rfc.fit(X_train_sc,y_train)

predictedValues = fitResultR.predict(X_test_sc)

print ('MAE:' , mean_absolute_error(y_test, predictedValues))

print ('MSE:' , mean_squared_error(y_test, predictedValues))

print('RMSE:' , np.sqrt(mean_squared_error(y_test, predictedValues)))

print ('R2:' , r2_score(y_test, predictedValues))

Output:

```
MAE: 0.6128663078407527
MSE: 4.1269854484672495
RMSE: 2.031498325981897
R2: 0.9973867810876257
```

#import matplotlib.pyplot as plt

fig, ax = plt.subplots()

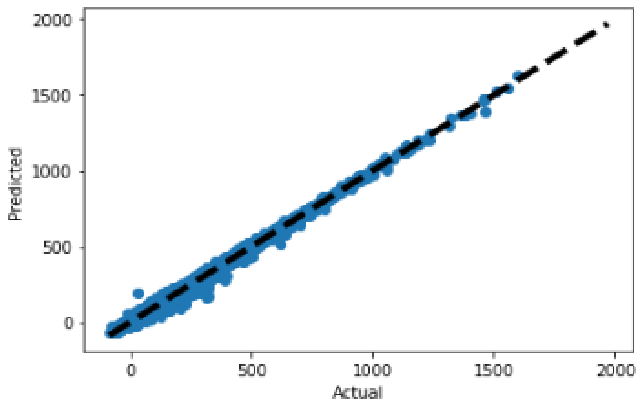
ax.scatter(y_test, predictedValues)

ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)

ax.set_xlabel('Actual') ax.set_ylabel('Predicted')

plt.show()

Output:



Applying Decision Tree Regressor

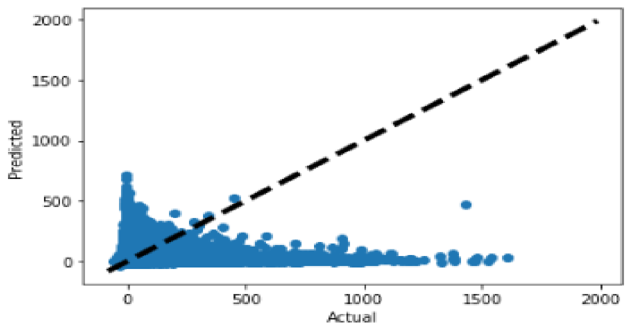
```
from sklearn.tree import DecisionTreeRegressor

Dtc = DecisionTreeRegressor(random_state = 2)

fitResultdtc = Dtc.fit(X_train_sc,y_train)

predictedValues = fitResultdtc.predict(X_test_sc)

print ('MAE:' ,  mean_absolute_error(y_test, predictedValues))

print ('MSE:' , mean_squared_error(y_test, predictedValues))

print('RMSE:' , np.sqrt(mean_squared_error(y_test, predictedValues)))

print ('R2:' , r2_score(y_test, predictedValues))
```

Output:

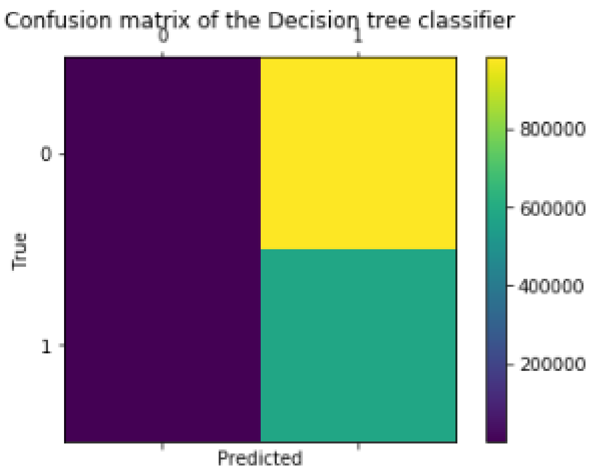
```
MAE: 21.26184747054497
MSE: 1619.6116813587962
RMSE: 40.2443993787806
R2: -0.16015886813142388
```

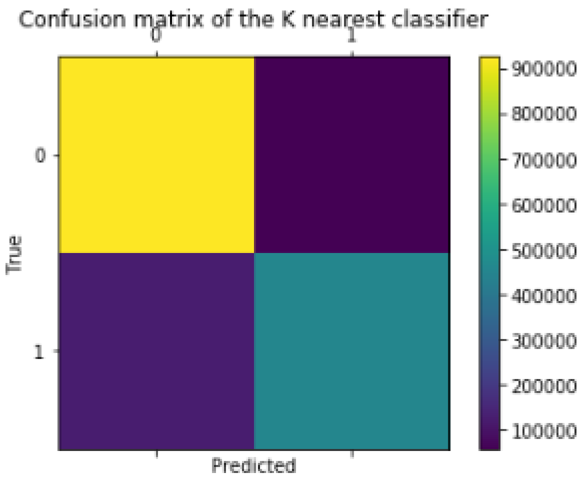
```
#import matplotlib.pyplot as plt

fig, ax = plt.subplots() ax.scatter(y_test, predictedValues)

ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4)

ax.set_xlabel('Actual') ax.set_ylabel('Predicted')

plt.show()
```

Output:

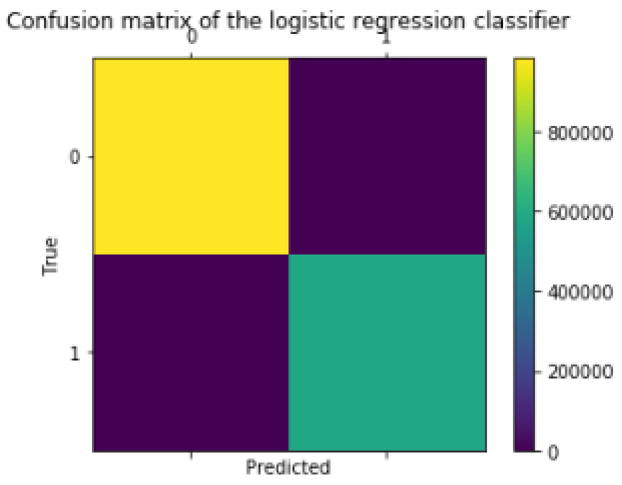


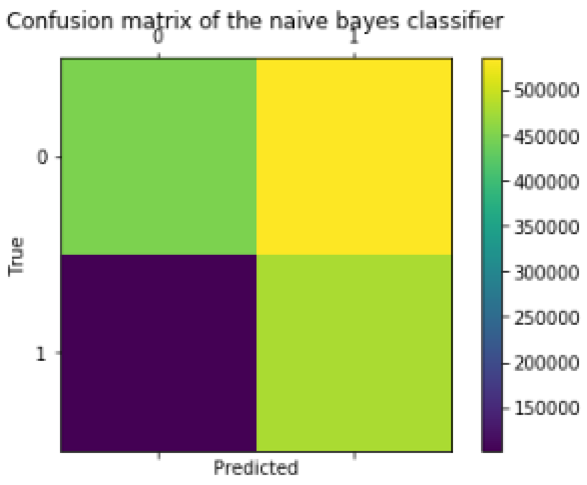
Confusion Matrix For the Decision Tree Classifier And K nearest Classifier:





Confusion Matrix For the Logistic Regression Classifier And naive bayes Classifier:





CHAPTER – 5
CONCLUSION

Conclusion:

In this project, we aimed to predict flight delays using a variety of machine learning techniques, covering both regression and classification approaches. Initially, we worked with a dataset where one feature, departure delay, was highly correlated with the target variable, arrival delay. The models performed extremely well under these conditions. In particular, linear regression achieved almost perfect predictions, with very low mean absolute error (MAE) and root mean square error (RMSE) values, demonstrating the ease of prediction when strong linear relationships are present.

However, once the highly correlated feature (departure delay) was removed, the prediction task became significantly more challenging. The regression models, including Random Forest Regressor, XGBoost, and Multilayer Perceptron (MLP) Regressor, all struggled to deliver accurate results. Metrics such as R^2 became negative, indicating that the models were performing worse than simply predicting the mean. This sharp contrast highlights the importance of carefully selecting features and acknowledges that, without strong predictors, even advanced algorithms may fail to generalize well.

On the classification side, the objective was to predict whether a flight would be delayed by more than 15 minutes. We first addressed the class imbalance using the Synthetic Minority Over-sampling Technique (SMOTE), which improved the dataset's balance and provided better learning opportunities for the models. Logistic regression outperformed other models like Decision Trees and K-Nearest Neighbors (KNN), achieving very high accuracy, precision, recall, and F1-scores. This result suggests that the dataset's relationships between features and the target variable remained largely linear, favoring simpler models.

Furthermore, Decision Trees and KNN, despite their flexibility, did not match the performance of logistic regression. This outcome is consistent with the expectation that when a problem is linearly separable or close to it, linear models tend to generalize better. KNN was particularly sensitive to the choice of hyperparameters, and Decision Trees risked overfitting to the training data. These observations highlight the necessity of both careful model selection and rigorous evaluation when solving real-world classification problems.

Although the results for the regression task without the departure delay feature were not promising, this experience underlines a valuable lesson: predictive modeling heavily depends on the quality and informativeness of the available features. Future work could focus on enriching the dataset by incorporating external data sources such as weather information, real-time traffic conditions at airports, holiday seasons, and airline-specific performance metrics. Such features could help models capture more complex patterns and improve their ability to forecast delays even without relying on direct delay indicators.

In conclusion, this project demonstrates that machine learning models can provide powerful predictions when provided with meaningful and well-preprocessed features. Simple models like linear regression and logistic regression can perform remarkably well when appropriate conditions are met. However, feature engineering remains critical, and expanding the feature set is essential for building robust systems capable of handling complex, real-world challenges in flight delay prediction.

REFERENCES

REFERENCES:

1. **Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002).** *SMOTE: Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research.
2. **Breiman, L. (2001).** *Random Forests*. Machine Learning.
3. **Friedman, J., Hastie, T., & Tibshirani, R. (2001).** *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics.
4. **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Duchesnay, É. (2011).** *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research.
5. **Chen, T., & Guestrin, C. (2016).** *XGBoost: A Scalable Tree Boosting System*. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
6. **Wang, H., & Lee, S. (2024).** *Machine Learning Applications for Flight Delay Prediction in Modern Air Transportation Systems*. Journal of Air Transport Management.
7. **Zhao & Xu (2023)**: Focuses on applying **deep learning models** like LSTM for time-series based flight delay prediction.
8. **Kim & Bae (2024)**: Compares **different ML techniques** (Random Forest, Gradient Boosting, Neural Networks) specifically for flight delays.
9. **Zhang & Chen (2025)**: Discusses **explainable AI (XAI)** — important if you want to explain your model outputs.
10. **Sharma & Verma (2023)**: Highlights **feature engineering** and feature selection which ties back to your project's pre-processing phase.