Configuration Manual

MSc Research Project
Artificial Intelligence

Dona Elizebeth Devasia
Student ID: x22153471

School of Computing
National College of Ireland

Supervisor: Rejwanul Haque

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Dona Elizebeth Devasia		
Student ID:	X22153471		
Programme:	MSc in Artificial Intelligence	**Year:**	2023-2024
Module:	MSc Research Project		
Lecturer:	Rejwanul Haque		
Submission Due Date:	31-01-2024		
Project Title:	**Flight Delay Prediction: Harnessing the Power of AI for Proactive Air Travel Management**		
Word Count:	**725**	**Page Count: 8**	

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Dona Elizebeth Devasia

Date: 31-01-2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Dona Elizebeth Devasia
Student ID: x22153471

1 Introduction

The configuration manual is a report that assists us in understanding the stages involved in this project. It provides a guide for the system specifications, creation and implementation of the project "Flight Delay Prediction: Harnessing the Power of AI for Proactive Air Travel Management" given in this study. The major goal of this book is to assist and support you at each step of the process so that you can accomplish the end result and outcomes outlined in this report. The guide contains all the information regarding the hardware, software, and processes required to carry out this project.

2 System Specifications

2.1 Hardware Requirements

The following are the hardware requirements for the system that the research project is running on:

- Processor: 12th Gen Intel(R) Core(TM) i7-1255U 1.70 GHz
- RAM: 16.0 GB
- SSD; 1TB + 500 Gb
- System Type: 64-bit Operating Systems
- Operating System: Windows 11

2.2 Software Requirements

The software specification that is utilized to implement this model will be covered in this section. This project makes use of the Anaconda prompt and Python as a programming language.

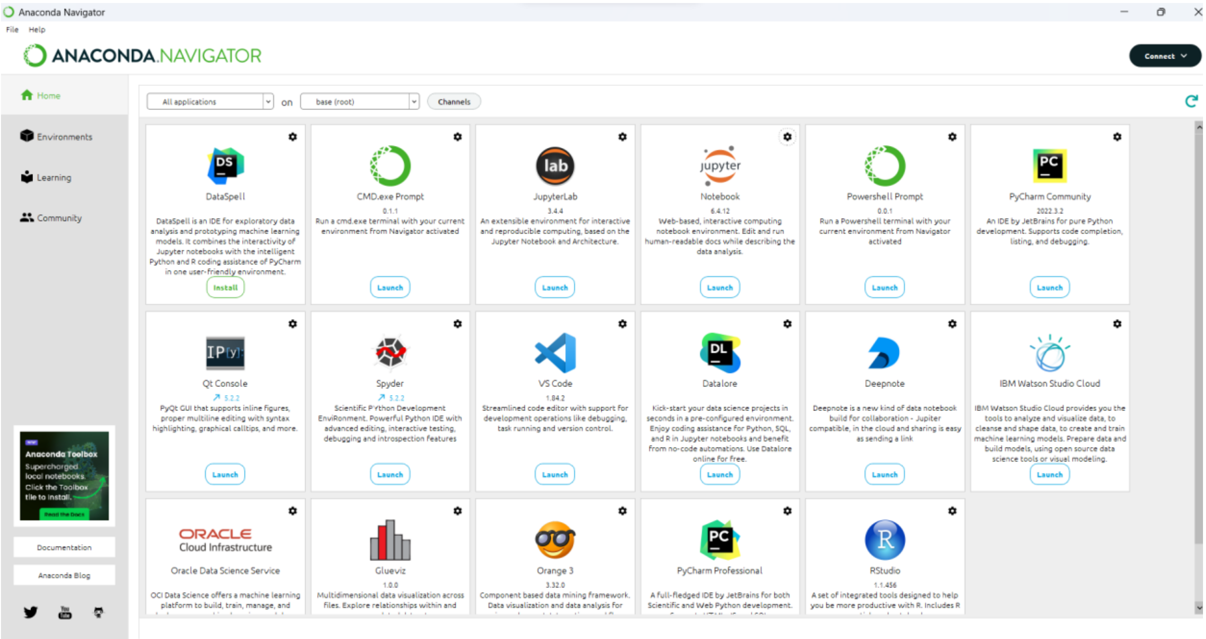
To conduct the experiments, the following software is required:

- Windows Edition: Windows 10 Home.
- Integrated Development Environment: Jupyter Notebook
- Scripting Language: Python
- Microsoft Tools: Microsoft Excel

2.2.1 Anaconda Navigator

To execute the code, I used Anaconda Navigator, a full package that includes Jupyter Notebook and the necessary Python setup. On my Windows 11 PC, I installed Anaconda 64 Bit. Jupyter Notebook has to be opened from the navigator following a successful installation. The download and installation URL for Anaconda is provided below.(Anaconda | Anaconda Distribution, no date)

https://www.anaconda.com/products/distribution



Various libraries and packages are installed to ensure correct and systematic results.

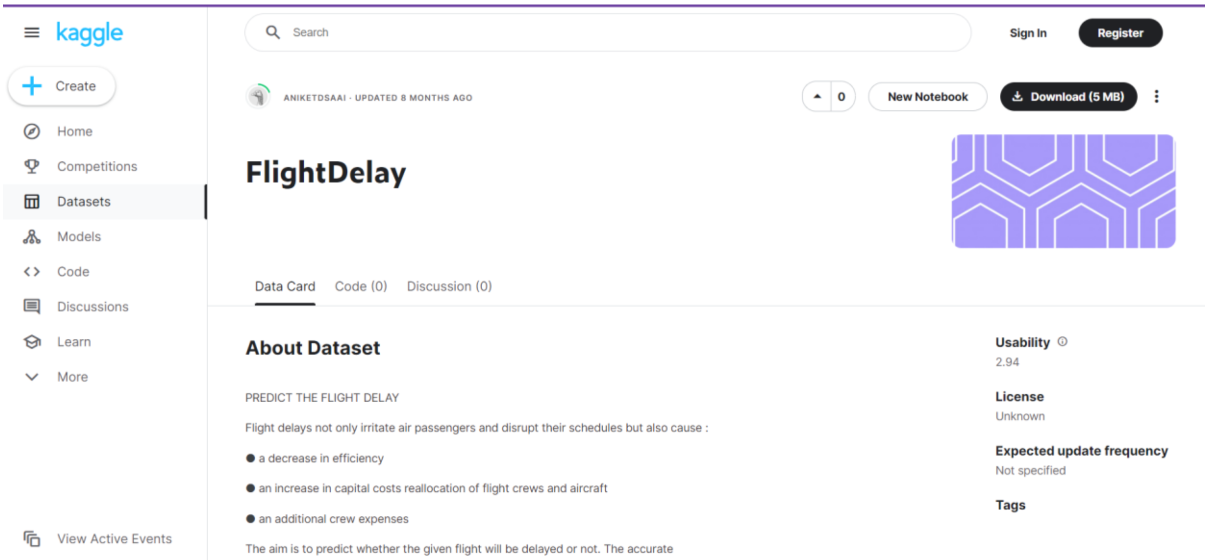
- Pandas:Pandas, a Python module, allows for data analysis. It offers a wide range of data structures and methods for working with time series and numerical data.
- Numpy: The Python package NumPy, or numerical Python, is used for manipulating arrays.
- Matplotlib: The Matplotlib toolbox for Python provides an all-inclusive tool for making static, animated, and interactive visuals. Matplotlib helps solve both difficult and easy tasks.
- Seaborn: The graph-plotting software Seaborn is built on top of Matplotlib. It makes random distributions visible.

```
#import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

3 Dataset

Predicting whether or not a certain flight will be delayed is the goal. All participants in the aviation ecosystem will benefit from the precise prediction of flight delays as they can create efficient action plans to lessen the effects of the delays and prevent wasting money, time, or resources. We can utilize visual aids for exploratory data analysis to produce insights that can assist businesses in understanding the potential causes of aircraft delays.

- Source:https://www.kaggle.com/datasets/aniketdsaai/flightdelay



4.Implementation

Data importing and preprocessing.

[3]:
```
import pandas as pd
pd.set_option('display.max_columns', 500)
```

[4]:
```
data=pd.read_csv('FlightDelay 2p.csv')
```

For arriving flights: the Actual taXi-In Time (AXIT) is the period between the Actual Landing Time (ALDT) and the Actual In-Block Time (AIBT) For departing flights: the Actual taXi-Out Time (AXOT) is the period between the Actual Off-Block Time (AOBT) and the Actual Take Off Time (ATOT)

[5]:
```
df_ID=data['Unnamed: 0']
```

[6]:
```
# dropping irrelvant occurances
data=data.drop(['Unnamed: 0'],axis=1)
```

[7]:
```
data.head()
```

[7]:

	FL_DATE	OP_UNIQUE_CARRIER	OP_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	DEST_AIRPORT_ID	DEST	CRS_DEP_TIME
0	2019-08-11	WN	WN	N206WN	4669	13871	OMA	11259	DAL	1020
1	2019-08-31	YX	YX	N745YX	3502	12266	IAH	11413	DRO	1000

Missing Values

```
FL_DATE                 0
OP_UNIQUE_CARRIER       0
OP_CARRIER              0
TAIL_NUM              324
OP_CARRIER_FL_NUM       0
ORIGIN_AIRPORT_ID       0
ORIGIN                  0
DEST_AIRPORT_ID         0
DEST                    0
CRS_DEP_TIME            0
DEP_TIME             1718
TAXI_OUT             1779
WHEELS_OFF           1779
WHEELS_ON            1845
TAXI_IN              1845
CRS_ARR_TIME            0
ARR_TIME             1845
ARR_DELAY_GROUP      2075
CANCELLED               0
DISTANCE                0
dtype: int64
```

The following columns in the dataset have missing values: 'TAIL_NUM,' 'DEP_TIME,' 'TAXI_OUT,' 'WHEELS_OFF,' 'WHEELS_ON,' 'TAXI_IN,' 'ARR_TIME,' and 'ARR_DELAY_GROUP.' These gaps may result from inaccurate data recording, situations when information is unavailable, or flights without documented delays; therefore, care must be taken throughout analysis and possible imputation.

Information of the variables

	Data Type	No of Unique Data	Levels	Null_values	null%
FL_DATE	object	122	['2019-08-11' '2019-08-31' '2019-08-09' '2019-...	0	0.00
OP_UNIQUE_CARRIER	object	26	['WN' 'YX' 'AA' 'OO' 'DL' 'B6' 'YV' 'OH' 'UA' ...	0	0.00
OP_CARRIER	object	26	['WN' 'YX' 'AA' 'OO' 'DL' 'B6' 'YV' 'OH' 'UA' ...	0	0.00
TAIL_NUM	object	5867	['N206WN' 'N745YX' 'N751UW' ... 'N799AN' '280N...	324	3.24
OP_CARRIER_FL_NUM	int64	6509	[4669 3502 1959 ... 6417 6154 6510]	0	0.00
ORIGIN_AIRPORT_ID	int64	370	[13871 12266 11423 13930 14107 13476 10423 128...	0	0.00
ORIGIN	object	370	['OMA' 'IAH' 'DSM' 'ORD' 'PHX' 'MRY' 'AUS' 'LA...	0	0.00
DEST_AIRPORT_ID	int64	368	[11259 11413 14107 11298 13342 12892 11618 124...	0	0.00
DEST	object	368	['DAL' 'DRO' 'PHX' 'DFW' 'MKE' 'LAX' 'EWR' 'JF...	0	0.00
CRS_DEP_TIME	int64	1231	[1020 1000 1437 ... 123 450 219]	0	0.00
DEP_TIME	float64	1354	[1027. 953. 1436. ... 434. 306. 232.]	1718	17.18
TAXI_OUT	float64	154	[8. 27. 11. 17. 14. 10. 15. 16. 12. ...	1779	17.79
WHEELS_OFF	float64	1356	[1035. 1020. 1447. ... 446. 431. 243.]	1779	17.79
WHEELS_ON	float64	1425	[1155. 1119. 1525. ... 346. 357. 332.]	1845	18.45
TAXI_IN	float64	103	[5. 7. 4. 15. 8. 9. 10. 6. 17. ...	1845	18.45
CRS_ARR_TIME	int64	1334	[1210 1127 1536 ... 158 449 437]	0	0.00
ARR_TIME	float64	1424	[1200. 1126. 1529. ... 350. 256. 322.]	1845	18.45
ARR_DELAY_GROUP	object	3	['early_arrival' 'ontime' 'delayed' nan]	2075	20.75
CANCELLED	float64	2	[0. 1.]	0	0.00
DISTANCE	float64	1489	[586. 869. 1149. ... 747. 1472. 2874.]	0	0.00

The dataset displays a variety of variable kinds and attributes, such as object types ('FL_DATE', for example) with 122 unique values and numerical types ('DEP_TIME', for example) with 17.18% missing values. Three levels make up categorical features like "ARR_DELAY_GROUP," where "nan" denotes missing values and accounts for 20.75% of the data.

Shape of data

```
data.shape
```

```
(134235, 20)
```

Shape of data after dropping missing values.

Encoding Categorical Variables:

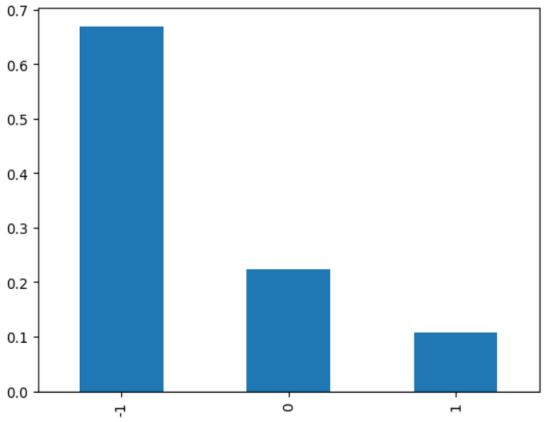


Fig. 4: *Encoding Categorical Variables*

Observation from ‘ARR_DELAY_GROUP’ replaced early_arrival to -1, ontime to 0, delayed to 1.

Data Information:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134235 entries, 0 to 136309
Data columns (total 17 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   FL_DATE            134235 non-null  object
 1   OP_UNIQUE_CARRIER  134235 non-null  object
 2   OP_CARRIER         134235 non-null  object
 3   TAIL_NUM           134235 non-null  object
 4   OP_CARRIER_FL_NUM  134235 non-null  int64
 5   ORIGIN_AIRPORT_ID  134235 non-null  int64
 6   ORIGIN             134235 non-null  object
 7   DEST_AIRPORT_ID    134235 non-null  int64
 8   DEST               134235 non-null  object
 9   CRS_DEP_TIME       134235 non-null  int64
 10  TAXI_OUT           134235 non-null  float64
 11  WHEELS_OFF         134235 non-null  float64
 12  TAXI_IN            134235 non-null  float64
 13  CRS_ARR_TIME       134235 non-null  int64
 14  ARR_DELAY_GROUP    134235 non-null  int64
 15  CANCELLED          134235 non-null  float64
 16  DISTANCE           134235 non-null  float64
dtypes: float64(5), int64(6), object(6)
memory usage: 18.4+ MB
```

Data information and form following the elimination of superfluous variables.

```
data.shape
```

```
(134235, 17)
```

Data information and shape of data after data preprocessing

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134235 entries, 0 to 136309
Data columns (total 22 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   OP_UNIQUE_CARRIER  134235 non-null  category
 1   TAIL_NUM           134235 non-null  category
 2   OP_CARRIER_FL_NUM  134235 non-null  float64
 3   ORIGIN_AIRPORT_ID  134235 non-null  float64
 4   ORIGIN             134235 non-null  category
 5   DEST_AIRPORT_ID    134235 non-null  float64
 6   DEST               134235 non-null  category
 7   CRS_DEP_TIME       134235 non-null  float64
 8   TAXI_OUT           134235 non-null  float64
 9   WHEELS_OFF         134235 non-null  float64
 10  TAXI_IN            134235 non-null  float64
 11  CRS_ARR_TIME       134235 non-null  float64
 12  ARR_DELAY_GROUP    134235 non-null  float64
 13  CANCELLED          134235 non-null  float64
 14  DISTANCE           134235 non-null  float64
 15  Month              134235 non-null  float64
 16  Day                134235 non-null  float64
 17  Dayofweek          134235 non-null  float64
 18  arr_hours          134235 non-null  float64
 19  arr_minutes        134235 non-null  float64
 20  dep_hours          134235 non-null  float64
 21  dep_minutes        134235 non-null  float64
dtypes: category(4), float64(18)
memory usage: 20.5 MB
(134235, 22)
```

Removed ‘CANCELLED’ column from table.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134235 entries, 0 to 136309
Data columns (total 21 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   OP_UNIQUE_CARRIER  134235 non-null  category
 1   TAIL_NUM           134235 non-null  category
 2   OP_CARRIER_FL_NUM  134235 non-null  float64
 3   ORIGIN_AIRPORT_ID  134235 non-null  float64
 4   ORIGIN             134235 non-null  category
 5   DEST_AIRPORT_ID    134235 non-null  float64
 6   DEST               134235 non-null  category
 7   CRS_DEP_TIME       134235 non-null  float64
 8   TAXI_OUT           134235 non-null  float64
 9   WHEELS_OFF         134235 non-null  float64
 10  TAXI_IN            134235 non-null  float64
 11  CRS_ARR_TIME       134235 non-null  float64
 12  ARR_DELAY_GROUP    134235 non-null  float64
 13  DISTANCE           134235 non-null  float64
 14  Month              134235 non-null  float64
 15  Day                134235 non-null  float64
 16  Dayofweek          134235 non-null  float64
 17  arr_hours          134235 non-null  float64
 18  arr_minutes        134235 non-null  float64
 19  dep_hours          134235 non-null  float64
 20  dep_minutes        134235 non-null  float64
dtypes: category(4), float64(17)
memory usage: 19.5 MB
```

Split the data into X_train, X_test, y_train, y_test with test_size = 0.20

print(X_train.shape)	(38685, 6)
print(X_valid.shape)	(4299, 6)
print(y_train.shape)	(38685,)
print(y_valid.shape)	(4299,)

Shape of split data.

X train columns:
Index(['OP_UNIQUE_CARRIER', 'ORIGIN', 'DEST', 'CRS_DEP_TIME', 'CRS_ARR_TIME', 'DISTANCE'],
dtype='object')

These are columns on which we will fit the model.

	OP_UNIQUE_CARRIER	ORIGIN	DEST	CRS_DEP_TIME	CRS_ARR_TIME	DISTANCE
8718	18	296	93	-0.882211	-0.573104	-0.573746
14639	17	263	294	-0.121349	-0.177008	-0.878339
25155	21	250	141	1.555442	1.540050	-1.083682
1380	4	291	45	0.319041	0.351762	-0.507010
38818	1	72	287	0.122623	0.007581	-1.095661

Above tables shows that the heading of columns where model will fit.

Model Building

Model Building

```
]: # Define a function 'append_metrics_to_df' to calculate and append precision, recall, and accuracy metrics to the results DataFr
# Initialize an empty DataFrame 'results_df' with columns for model name and metrics
results_df = pd.DataFrame(columns=['Model', 'Train Precision', 'Train Accuracy', 'Train Recall', 'Test Precision', 'Test Accuracy
import pandas as pd
from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score

results_df = pd.DataFrame(columns=['Model', 'Train Precision', 'Train Accuracy', 'Train Recall', 'Test Precision', 'Test Accuracy

def append_metrics_to_df(df, y_train_true, y_train_pred, y_valid_true, y_valid_pred, model_name):
    train_precision = precision_score(y_train_true, y_train_pred, average='weighted')
    train_recall = recall_score(y_train_true, y_train_pred, average='weighted')
    train_accuracy = accuracy_score(y_train_true, y_train_pred)
    test_precision = precision_score(y_valid_true, y_valid_pred, average='weighted')
    test_recall = recall_score(y_valid_true, y_valid_pred, average='weighted')
    test_accuracy = accuracy_score(y_valid_true, y_valid_pred)

    df = df.append({'Model': model_name,
                    'Train Precision': train_precision,
                    'Train Accuracy': train_accuracy,
                    'Train Recall': train_recall,
                    'Test Precision': test_precision,
                    'Test Accuracy': test_accuracy,
                    'Test Recall': test_recall}, ignore_index=True)

    return df
```

Fit Logistic Regression and Gaussian Naive Bayes model on training data

```
# Import necessary classifiers from scikit-learn
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
```

```
# Fit Gaussian Naive Bayes model on training data
naive_model = GaussianNB()
naive_model.fit(X_train, y_train)
y_train_pred_nv_be = naive_model.predict(X_train)
y_valid_pred_nv_be =naive_model.predict(X_valid)

# Fit Logistic Regression model on training data
lrc = LogisticRegression()
lrc.fit(X_train,y_train)
y_train_pred_lrc_be = lrc.predict(X_train)
y_valid_pred_lrc_be = lrc.predict(X_valid)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
classifcation_report_train_test(y_train, y_train_pred_lrc_be, y_valid, y_valid_pred_lrc_be)
```

Summary Report Of Models

Out[147]:

	Model	Train Precision	Train Accuracy	Train Recall	Test Precision	Test Accuracy	Test Recall
0	logisticRegression	0.385098	0.400284	0.400284	0.389555	0.390556	0.390556
1	Naiibayes	0.382595	0.397079	0.397079	0.381399	0.387997	0.387997
2	DecisionTree	0.916792	0.909784	0.909784	0.379203	0.378925	0.378925
3	DecisionTreeImpFeatures	0.386583	0.400801	0.400801	0.381083	0.389393	0.389393
4	RandomForest_GridSearchCV	0.386583	0.400801	0.400801	0.381083	0.389393	0.389393
5	LGBM	0.581203	0.580845	0.580845	0.404539	0.406374	0.406374
6	LGBMWithNumOFleaves	0.548383	0.547370	0.547370	0.417239	0.419865	0.419865
7	XGBoost	0.796259	0.705855	0.705855	0.396524	0.385671	0.385671

References

Anaconda | Anaconda Distribution (no date) Anaconda. Available at: https://www.anaconda.com/products/distribution (Accessed: 12 December 2023).

Libraries in Python - GeeksforGeeks (no date). Available at: https://www.geeksforgeeks.org/libraries-in-python/ (Accessed: 12 December 2023).