CIS 662 Intro to Machine Learning and Algorithms

Flight delay prediction project - group 10

Team Members:

Vikas Papana

Ajay Hemanshu Desai

Hemil Anip Shah

Rangel Anselm Koli

In [1]:
```
# Generic inputs for most ML tasks
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report

pd.options.display.float_format = '{:,.2f}'.format
from IPython.display import display, HTML
```

Flight Data

Fetching, reading and preprocessing flight data

In [2]:
```
# Reading Arrival flights data
Arrival_data = pd.read_csv(r"C:\Users\VIKAS\Documents\Intro to ML\Project\datasets\Arrival_data.csv",
                           parse_dates = ['Date (MM/DD/YYYY)', 'Scheduled Arrival Time'])

Arrival_data.head()
```

Out[2]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Tail Number	Origin Airport	Scheduled Arrival Time	Actual Arrival Time	Scheduled Elapsed Time (Minutes)	Actual Elapsed Time (Minutes)	Arrival Delay (Minutes)	Wheels-on Time	Taxi-In time (Minutes)	C (Mir
0	MQ	2023-01-01	3392	N283NN	ORD	2024-05-02 21:16:00	21:08	111	104	-8	21:02	6	
1	MQ	2023-01-01	3518	N213NN	ORD	2024-05-02 16:02:00	15:37	117	96	-25	15:33	4	
2	MQ	2023-01-02	3392	N248NN	ORD	2024-05-02 20:19:00	20:12	106	102	-7	20:08	4	
3	MQ	2023-01-02	3518	N263NN	ORD	2024-05-02 16:02:00	15:56	117	112	-6	15:52	4	
4	MQ	2023-01-03	3392	N276NN	ORD	2024-05-02 20:19:00	20:57	106	113	38	20:52	5	

In [3]:
```
Arrival_data.columns
```

Out[3]:
```
Index(['Carrier Code', 'Date (MM/DD/YYYY)', 'Flight Number', 'Tail Number',
       'Origin Airport', 'Scheduled Arrival Time', 'Actual Arrival Time',
       'Scheduled Elapsed Time (Minutes)', 'Actual Elapsed Time (Minutes)',
       'Arrival Delay (Minutes)', 'Wheels-on Time', 'Taxi-In time (Minutes)',
       'Delay Carrier (Minutes)', 'Delay Weather (Minutes)',
       'Delay National Aviation System (Minutes)', 'Delay Security (Minutes)',
       'Delay Late Aircraft Arrival (Minutes)'],
      dtype='object')
```

In [4]:
```
# Reading departure flights data
Departure_data = pd.read_csv(r"C:\Users\VIKAS\Documents\Intro to ML\Project\datasets\Departure_data.csv",
                             parse_dates = ['Date (MM/DD/YYYY)', 'Scheduled departure time'])

Departure_data.head()
```

Out[4]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Tail Number	Destination Airport	Scheduled departure time	Actual departure time	Scheduled elapsed time (Minutes)	Actual elapsed time (Minutes)	Departure delay (Minutes)	Wheels-off time	Taxi-O tin (Minute
0	MQ	2023-01-01	3305	N663AR	COU	2024-05-02 19:35:00	19:34	81	97	-1	20:08	
1	MQ	2023-01-01	3309	N939AE	MKE	2024-05-02 09:30:00	9:27	68	53	-3	9:53	
2	MQ	2023-01-01	3315	N689EC	ALO	2024-05-02 09:00:00	8:50	85	60	-10	9:06	
3	MQ	2023-01-01	3322	N902BC	COU	2024-05-02 12:55:00	12:52	85	80	-3	13:11	
4	MQ	2023-01-01	3324	N267NN	AUS	2024-05-02 19:15:00	19:08	172	167	-7	19:23	

In [5]:
```
Departure_data.columns
```

Out[5]:
```
Index(['Carrier Code', 'Date (MM/DD/YYYY)', 'Flight Number', 'Tail Number',
       'Destination Airport', 'Scheduled departure time',
       'Actual departure time', 'Scheduled elapsed time (Minutes)',
       'Actual elapsed time (Minutes)', 'Departure delay (Minutes)',
       'Wheels-off time', 'Taxi-Out time (Minutes)', 'Delay Carrier (Minutes)',
       'Delay Weather (Minutes)', 'Delay National Aviation System (Minutes)',
       'Delay Security (Minutes)', 'Delay Late Aircraft Arrival (Minutes)'],
      dtype='object')
```

In [6]:
```
# Filtering data - flights from Chicago, New York and Orlando airports
Arrival_data = Arrival_data[Arrival_data['Origin Airport'].isin({'ORD', 'JFK', 'MCO'})]
Arrival_data = Arrival_data[['Carrier Code', 'Date (MM/DD/YYYY)', 'Flight Number', 'Origin Airport',
                             'Scheduled Arrival Time', 'Arrival Delay (Minutes)']]

Arrival_data.head()
```

Out[6]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Origin Airport	Scheduled Arrival Time	Arrival Delay (Minutes)
0	MQ	2023-01-01	3392	ORD	2024-05-02 21:16:00	-8
1	MQ	2023-01-01	3518	ORD	2024-05-02 16:02:00	-25
2	MQ	2023-01-02	3392	ORD	2024-05-02 20:19:00	-7
3	MQ	2023-01-02	3518	ORD	2024-05-02 16:02:00	-6
4	MQ	2023-01-03	3392	ORD	2024-05-02 20:19:00	38

In [7]:
```
# Filtering data - Flights reaching syracuse airport
Departure_data = Departure_data[Departure_data['Destination Airport'].isin({'SYR'})]
Departure_data = Departure_data[['Carrier Code', 'Date (MM/DD/YYYY)', 'Flight Number',
                                 'Destination Airport', 'Scheduled departure time']]

Departure_data.head()
```

Out[7]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Destination Airport	Scheduled departure time
15	MQ	2023-01-01	3392	SYR	2024-05-02 18:25:00
29	MQ	2023-01-01	3518	SYR	2024-05-02 13:05:00
134	MQ	2023-01-02	3392	SYR	2024-05-02 17:33:00
150	MQ	2023-01-02	3518	SYR	2024-05-02 13:05:00
272	MQ	2023-01-03	3392	SYR	2024-05-02 17:33:00

In [8]:
```
# Merging Arrival data and departure data based on carrier code , date and flight number
Flight_data = pd.merge(Arrival_data, Departure_data, on = ['Carrier Code', 'Date (MM/DD/YYYY)',
                                                           'Flight Number'])

Flight_data.head()
```

Out[8]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Origin Airport	Scheduled Arrival Time	Arrival Delay (Minutes)	Destination Airport	Scheduled departure time
0	MQ	2023-01-01	3392	ORD	2024-05-02 21:16:00	-8	SYR	2024-05-02 18:25:00
1	MQ	2023-01-01	3518	ORD	2024-05-02 16:02:00	-25	SYR	2024-05-02 13:05:00
2	MQ	2023-01-02	3392	ORD	2024-05-02 20:19:00	-7	SYR	2024-05-02 17:33:00
3	MQ	2023-01-02	3518	ORD	2024-05-02 16:02:00	-6	SYR	2024-05-02 13:05:00
4	MQ	2023-01-03	3392	ORD	2024-05-02 20:19:00	38	SYR	2024-05-02 17:33:00

In [9]:
```
Flight_data = Flight_data[Flight_data['Flight Number'].isin({538, 3402, 116, 5340, 491, 56, 656})]
Flight_data['Flight Number'].replace(56,656,inplace=True)

Flight_data.head()
```

Out[9]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Origin Airport	Scheduled Arrival Time	Arrival Delay (Minutes)	Destination Airport	Scheduled departure time
71	MQ	2023-06-01	3402	ORD	2024-05-02 22:32:00	57	SYR	2024-05-02 19:42:00
73	MQ	2023-06-02	3402	ORD	2024-05-02 22:32:00	2	SYR	2024-05-02 19:42:00
75	MQ	2023-06-03	3402	ORD	2024-05-02 23:41:00	-15	SYR	2024-05-02 20:46:00
77	MQ	2023-06-04	3402	ORD	2024-05-02 22:28:00	0	SYR	2024-05-02 19:39:00
79	MQ	2023-06-05	3402	ORD	2024-05-02 22:28:00	-7	SYR	2024-05-02 19:39:00

In [10]:
```
Flight_data['Date'] = Flight_data['Date (MM/DD/YYYY)']
Flight_data['Date_MM'] = Flight_data['Date'].dt.month
Flight_data['Date_DD'] = Flight_data['Date'].dt.day

Flight_data['Carrier'] = Flight_data['Carrier Code']
Flight_data['Flight_num'] = Flight_data['Flight Number']
Flight_data['Origin'] = Flight_data['Origin Airport']
Flight_data['Dest'] = Flight_data['Destination Airport']

Flight_data['Arr_t_h'] = Flight_data['Scheduled Arrival Time'].dt.hour
Flight_data['Arr_t_m'] = Flight_data['Scheduled Arrival Time'].dt.minute
Flight_data['Depart_t_h'] = Flight_data['Scheduled departure time'].dt.hour
Flight_data['Depart_t_m'] = Flight_data['Scheduled departure time'].dt.minute


def categorize_arrival_delay(delay):
    if delay <= -5:
        return 'Early'
    elif delay <= 5:
        return 'On Time'
    else:
        return 'Late'
Flight_data['Arr_Status'] = Flight_data['Arrival Delay (Minutes)'].apply(categorize_arrival_delay)

Flight_data.drop(columns = ['Date (MM/DD/YYYY)', 'Carrier Code', 'Flight Number', 'Origin Airport',
                            'Destination Airport', 'Scheduled Arrival Time', 'Scheduled departure time',
                            'Arrival Delay (Minutes)'], inplace = True)

Flight_data.head()
```

Out[10]:

	Date	Date_MM	Date_DD	Carrier	Flight_num	Origin	Dest	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Arr_Status
71	2023-06-01	6	1	MQ	3402	ORD	SYR	22	32	19	42	Late
73	2023-06-02	6	2	MQ	3402	ORD	SYR	22	32	19	42	On Time
75	2023-06-03	6	3	MQ	3402	ORD	SYR	23	41	20	46	Early
77	2023-06-04	6	4	MQ	3402	ORD	SYR	22	28	19	39	On Time
79	2023-06-05	6	5	MQ	3402	ORD	SYR	22	28	19	39	Early

In [11]:
```
set(Flight_data['Carrier'])
```

Out[11]: {'9E', 'B6', 'MQ', 'UA', 'WN'}

In [12]:
```
set(Flight_data['Flight_num'])
```

Out[12]: {116, 491, 538, 656, 3402, 5340}

In [13]:
```
set(Flight_data['Origin'])
```

Out[13]: {'JFK', 'MCO', 'ORD'}

In [14]:
```
set(Flight_data['Dest'])
```

Out[14]: {'SYR'}

In [15]:
```
set(Flight_data['Arr_Status'])
```

Out[15]: {'Early', 'Late', 'On Time'}

In [16]:
```
Flight_data['Arr_Status'].replace("Early",0,inplace=True)
Flight_data['Arr_Status'].replace("On Time",1,inplace=True)
Flight_data['Arr_Status'].replace("Late",2,inplace=True)

Flight_data.head()
```

Out[16]:

	Date	Date_MM	Date_DD	Carrier	Flight_num	Origin	Dest	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Arr_Status
71	2023-06-01	6	1	MQ	3402	ORD	SYR	22	32	19	42	2
73	2023-06-02	6	2	MQ	3402	ORD	SYR	22	32	19	42	1
75	2023-06-03	6	3	MQ	3402	ORD	SYR	23	41	20	46	0
77	2023-06-04	6	4	MQ	3402	ORD	SYR	22	28	19	39	1
79	2023-06-05	6	5	MQ	3402	ORD	SYR	22	28	19	39	0

In [17]:
```
Flight_data.shape
```

Out[17]:
```
(940, 12)
```

In [18]:
```
Flight_data.isna().sum()
```

Out[18]:
```
Date          0
Date_MM       0
Date_DD       0
Carrier       0
Flight_num    0
Origin        0
Dest          0
Arr_t_h       0
Arr_t_m       0
Depart_t_h    0
Depart_t_m    0
Arr_Status    0
dtype: int64
```

Weather Data

Fetching, reading and preproessing Syracuse weather data

In [19]:
```
# Fetching syracuse weather data for the year 2023
SYR_weather = pd.read_excel(r"C:\Users\VIKAS\Documents\Intro to ML\Project\datasets\SYR_weather_2023.xlsx
                            parse_dates = ['datetime'])
SYR_weather.head()
```

Out[19]:

	datetime	clouds	temp	max_temp	min_temp	wind_dir	max_wind_dir	wind_spd	max_wind_spd	wind_gust_spd	...	snow_de
0	2023-01-01	100	5.10	10.00	3.00	274	274	4.00	10.30	10.80	...	0
1	2023-01-02	100	5.50	7.80	3.90	207	207	2.20	5.40	7.20	...	0
2	2023-01-03	99	4.30	5.00	2.20	86	86	2.30	5.80	5.60	...	0
3	2023-01-04	100	6.20	7.20	5.00	140	140	2.40	6.70	12.40	...	0
4	2023-01-05	82	6.00	8.90	2.30	217	217	2.40	8.90	10.80	...	0

5 rows × 21 columns

In [20]:
```
SYR_weather.shape
```

Out[20]: (364, 21)

In [21]:
```
SYR_weather.isna().sum()
```

Out[21]:
```
datetime         0
clouds           0
temp             0
max_temp         0
min_temp         0
wind_dir         0
max_wind_dir     0
wind_spd         0
max_wind_spd     0
wind_gust_spd    0
snow             0
snow_depth       0
precip           0
precip_gpm       0
pres             0
solar_rad        0
t_solar_rad      0
uv               0
dewpt            0
rh               0
slp              0
dtype: int64
```

In [22]:
```
SYR_weather.drop(columns = ['max_wind_dir', 'max_wind_spd', 'precip_gpm', 'solar_rad', 't_solar_rad'],
                 inplace = True)

SYR_weather.columns
```

Out[22]:
```
Index(['datetime', 'clouds', 'temp', 'max_temp', 'min_temp', 'wind_dir',
       'wind_spd', 'wind_gust_spd', 'snow', 'snow_depth', 'precip', 'pres',
       'uv', 'dewpt', 'rh', 'slp'],
      dtype='object')
```

In [23]:
```
SYR_weather.shape
```

Out[23]:
```
(364, 16)
```

Flight weather data - main dataset

In [24]:
```
# Merging flight data and weather data
Flight_weather = pd.merge(Flight_data, SYR_weather, left_on='Date', right_on='datetime')

Flight_weather.drop(columns = ['datetime', 'Date'], inplace=True)

Flight_weather.head()
```

Out[24]:

	Date_MM	Date_DD	Carrier	Flight_num	Origin	Dest	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	...	wind_spd	wind_gust_spd
0	6	1	MQ	3402	ORD	SYR	22	32	19	42	...	1.60	7.60
1	6	1	B6	116	JFK	SYR	14	42	13	30	...	1.60	7.60
2	6	1	B6	656	MCO	SYR	19	2	16	15	...	1.60	7.60
3	6	1	9E	5340	JFK	SYR	0	4	22	30	...	1.60	7.60
4	6	1	UA	538	ORD	SYR	21	9	18	18	...	1.60	7.60

5 rows × 26 columns

In [25]:
```
Flight_weather.shape
```

Out[25]:
```
(938, 26)
```

In [26]:
```
Flight_weather.columns
```

Out[26]:
```
Index(['Date_MM', 'Date_DD', 'Carrier', 'Flight_num', 'Origin', 'Dest',
       'Arr_t_h', 'Arr_t_m', 'Depart_t_h', 'Depart_t_m', 'Arr_Status',
       'clouds', 'temp', 'max_temp', 'min_temp', 'wind_dir', 'wind_spd',
       'wind_gust_spd', 'snow', 'snow_depth', 'precip', 'pres', 'uv', 'dewpt',
       'rh', 'slp'],
      dtype='object')
```

In [27]:
```
Flight_weather.isna().sum()
```

Out[27]:
```
Date_MM          0
Date_DD          0
Carrier          0
Flight_num       0
Origin           0
Dest             0
Arr_t_h          0
Arr_t_m          0
Depart_t_h       0
Depart_t_m       0
Arr_Status       0
clouds           0
temp             0
max_temp         0
min_temp         0
wind_dir         0
wind_spd         0
wind_gust_spd    0
snow             0
snow_depth       0
precip           0
pres             0
uv               0
dewpt            0
rh               0
slp              0
dtype: int64
```

In [28]:
```
Flight_weather.dtypes
```

Out[28]:
```
Date_MM           int64
Date_DD           int64
Carrier          object
Flight_num        int64
Origin           object
Dest             object
Arr_t_h           int64
Arr_t_m           int64
Depart_t_h        int64
Depart_t_m        int64
Arr_Status        int64
clouds            int64
temp            float64
max_temp        float64
min_temp        float64
wind_dir          int64
wind_spd        float64
wind_gust_spd   float64
snow            float64
snow_depth      float64
precip          float64
pres              int64
uv              float64
dewpt           float64
rh                int64
slp               int64
dtype: object
```

In [29]:
```
Flight_weather = pd.get_dummies(Flight_weather, columns = ['Carrier', 'Origin', 'Dest'],
                                drop_first = True)

Flight_weather.head()
```

Out[29]:

	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Arr_Status	clouds	temp	...	uv	dewpt	rh	s
0	6	1	3402	22	32	19	42	2	26	24.30	...	10.10	10.80	47	10
1	6	1	116	14	42	13	30	0	26	24.30	...	10.10	10.80	47	10
2	6	1	656	19	2	16	15	0	26	24.30	...	10.10	10.80	47	10
3	6	1	5340	0	4	22	30	0	26	24.30	...	10.10	10.80	47	10
4	6	1	538	21	9	18	18	0	26	24.30	...	10.10	10.80	47	10

5 rows × 29 columns

In [30]:
```
Flight_weather.shape
```

Out[30]:
```
(938, 29)
```

In [31]:
```
Flight_weather.columns
```

Out[31]:
```
Index(['Date_MM', 'Date_DD', 'Flight_num', 'Arr_t_h', 'Arr_t_m', 'Depart_t_h',
       'Depart_t_m', 'Arr_Status', 'clouds', 'temp', 'max_temp', 'min_temp',
       'wind_dir', 'wind_spd', 'wind_gust_spd', 'snow', 'snow_depth', 'precip',
       'pres', 'uv', 'dewpt', 'rh', 'slp', 'Carrier_B6', 'Carrier_MQ',
       'Carrier_UA', 'Carrier_WN', 'Origin_MCO', 'Origin_ORD'],
      dtype='object')
```

In [32]:
```
Flight_weather.dtypes
```

Out[32]:
```
Date_MM            int64
Date_DD            int64
Flight_num         int64
Arr_t_h            int64
Arr_t_m            int64
Depart_t_h         int64
Depart_t_m         int64
Arr_Status         int64
clouds             int64
temp             float64
max_temp         float64
min_temp         float64
wind_dir           int64
wind_spd         float64
wind_gust_spd    float64
snow             float64
snow_depth       float64
precip           float64
pres               int64
uv               float64
dewpt            float64
rh                 int64
slp                int64
Carrier_B6         uint8
Carrier_MQ         uint8
Carrier_UA         uint8
Carrier_WN         uint8
Origin_MCO         uint8
Origin_ORD         uint8
dtype: object
```

In [33]:
```
# Splitting the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(Flight_weather.drop(columns = ['Arr_Status']),
                                                    Flight_weather['Arr_Status'], test_size=0.2,
                                                    random_state=42)
```

In [34]:
```
X_train
```

Out[34]:

	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	clouds	temp	max_temp	...	uv	dewpt	rh	
865	12	3	538	21	1	18	10	100	6.60	9.40	...	0.70	4.90	89	1
616	5	27	116	14	42	13	30	16	16.70	26.70	...	10.00	3.60	46	1
2	6	1	656	19	2	16	15	26	24.30	32.80	...	10.10	10.80	47	1
101	7	11	656	19	35	16	49	44	24.60	31.10	...	9.00	15.60	59	1
332	3	17	116	14	43	13	30	98	5.40	9.40	...	1.80	1.90	79	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
106	7	14	3402	21	32	18	36	71	22.70	28.90	...	7.50	16.10	68	1
270	2	24	656	16	51	14	7	96	-5.70	0.00	...	1.30	-8.70	80	1
860	12	2	116	15	21	14	0	100	7.00	9.40	...	0.70	4.70	86	1
435	4	11	538	21	34	18	40	74	14.90	22.80	...	2.50	0.00	39	1
102	7	12	3402	21	32	18	36	84	23.20	27.80	...	5.40	15.70	64	1

750 rows × 28 columns

In [35]:
```
X_test
```

Out[35]:

	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	clouds	temp	max_temp	...	uv	dewpt	rh	
70	6	25	3402	22	28	19	39	72	23.20	30.00	...	6.20	18.90	78	10
331	3	17	656	16	51	14	7	98	5.40	9.40	...	1.80	1.90	79	10
858	12	1	656	19	14	16	24	88	8.30	12.20	...	0.70	1.10	63	10
495	4	26	538	21	34	18	40	77	7.10	11.70	...	4.90	0.20	63	10
209	1	21	656	18	59	16	18	91	-0.30	0.60	...	0.80	-3.30	80	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
208	1	21	116	14	29	13	16	91	-0.30	0.60	...	0.80	-3.30	80	10
468	4	20	656	16	0	13	15	81	8.40	15.00	...	7.30	-1.00	54	10
82	7	1	3402	23	41	20	46	81	25.30	30.00	...	4.40	18.70	67	10
310	3	11	5340	23	59	22	32	82	-2.10	0.60	...	1.70	-5.10	80	10
817	11	16	538	21	1	18	10	53	7.70	16.70	...	2.10	0.10	61	10

188 rows × 28 columns

In [36]:
```
y_train
```

Out[36]:
```
865    0
616    0
2      0
101    1
332    1
      ..
106    0
270    1
860    0
435    0
102    2
Name: Arr_Status, Length: 750, dtype: int64
```

In [37]:
```
y_test
```

Out[37]:
```
70     1
331    2
858    2
495    2
209    2
      ..
208    2
468    1
82     2
310    1
817    0
Name: Arr_Status, Length: 188, dtype: int64
```

In [38]:
```
if False:
    from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    X_train = pd.DataFrame(sc.fit_transform(X_train), columns = X_train.columns, index = X_train.index)
    X_test = pd.DataFrame(sc.transform(X_test), columns = X_test.columns, index = X_test.index)

    X_train
    X_test
    y_train
    y_test
```

Random Forest Classifier

In [39]:
```
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)

y_pred_rf = rf_classifier.predict(X_test)

rf_classifier.score(X_train, y_train)
```

Out[39]:
```
1.0
```

In [40]:
```
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf)}")
```

```
Accuracy: 0.5053191489361702
```

In [41]:
```
test_output = pd.DataFrame(rf_classifier.predict(X_test), index = X_test.index,
                           columns = ['pred_Type'])
test_output = test_output.merge(y_test, left_index = True, right_index = True)
test_output.head()
```

Out[41]:

	pred_Type	Arr_Status
70	2	1
331	1	2
858	0	2
495	0	2
209	0	2

In [42]:
```
print('Percentage of correct predictions is ')
print(rf_classifier.score(X_test, y_test))
```

```
Percentage of correct predictions is
0.5053191489361702
```

Gradient Boosting Classifier

In [43]:
```
gb = GradientBoostingClassifier(random_state=50, min_samples_split = 12, min_samples_leaf = 6,
                                max_depth = 4, n_estimators = 100)
gb = gb.fit(X_train, y_train)

y_pred_gb = gb.predict(X_test)

gb.score(X_train, y_train)
```

Out[43]: 0.928

In [44]:
```
print(f"Accuracy: {accuracy_score(y_test, y_pred_gb)}")
```

```
Accuracy: 0.5053191489361702
```

In [45]:
```
test_output = pd.DataFrame(gb.predict(X_test), index = X_test.index, columns = ['pred_Type'])
test_output = test_output.merge(y_test, left_index = True, right_index = True)
test_output.head()
```

Out[45]:

	pred_Type	Arr_Status
70	2	1
331	1	2
858	0	2
495	0	2
209	0	2

In [46]:
```
print('Percentage of correct predictions is ')
print(gb.score(X_test, y_test))
```

```
Percentage of correct predictions is
0.5053191489361702
```

XGBoost Classifier

In [47]:
```
xgb_classifier = xgb.XGBClassifier(objective='multi:softmax', num_class=3, random_state=42)
xgb_classifier.fit(X_train, y_train)

y_pred_xgb = xgb_classifier.predict(X_test)

xgb_classifier.score(X_train, y_train)
```

Out[47]: 1.0

In [48]:
```
print(f"Accuracy: {accuracy_score(y_test, y_pred_xgb)}")
```

```
Accuracy: 0.43617021276595747
```

In [49]:
```
test_output = pd.DataFrame(xgb_classifier.predict(X_test), index = X_test.index,
                           columns = ['pred_Type'])
test_output = test_output.merge(y_test, left_index = True, right_index = True)
test_output.head()
```

Out[49]:

	pred_Type	Arr_Status
70	2	1
331	1	2
858	0	2
495	0	2
209	0	2

In [50]:
```
print('Percentage of correct predictions is ')
print(xgb_classifier.score(X_test, y_test))
```

```
Percentage of correct predictions is
0.43617021276595747
```

Logistic Regression

In [51]:
```
Log_reg = LogisticRegression(fit_intercept = True, solver='newton-cg',
                             multi_class = 'multinomial', penalty = 'none', max_iter = 1000)

Log_reg.fit(X_train, y_train)

# The following gives the mean accuracy on the given data and labels
Log_reg.score(X_train, y_train)

# This is the coefficient Beta_1, ..., Beta_7
#model.coef_

# This is the coefficient Beta_0
#model.intercept_
```

```
C:\Users\VIKAS\anaconda3\lib\site-packages\scipy\optimize\linesearch.py:477: LineSearchWarning: The line
search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
C:\Users\VIKAS\anaconda3\lib\site-packages\scipy\optimize\linesearch.py:327: LineSearchWarning: The line
search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
C:\Users\VIKAS\anaconda3\lib\site-packages\scipy\optimize\linesearch.py:437: LineSearchWarning: Rounding
errors prevent the line search from converging
  warn(msg, LineSearchWarning)
C:\Users\VIKAS\anaconda3\lib\site-packages\sklearn\utils\optimize.py:204: UserWarning: Line Search faile
d
  warnings.warn('Line Search failed')
```

Out[51]: 0.5426666666666666

In [52]:
```
test_output = pd.DataFrame(Log_reg.predict(X_test), index = X_test.index, columns = ['pred_Type'])
test_output = test_output.merge(y_test, left_index = True, right_index = True)
test_output.head()
```

Out[52]:

	pred_Type	Arr_Status
70	2	1
331	2	2
858	0	2
495	2	2
209	0	2

In [53]:
```
print('Percentage of correct predictions is ')
print(Log_reg.score(X_test, y_test))
```

```
Percentage of correct predictions is
0.48936170212765956
```

Testing weather data

In [54]:
```
test_weather = pd.read_excel(r"C:\Users\VIKAS\Documents\Intro to ML\Project\datasets\test_weather.xlsx",
                             parse_dates = ['datetime'])
test_weather
```

Out[54]:

	datetime	clouds	temp	max_temp	min_temp	wind_dir	wind_spd	wind_gust_spd	snow	snow_depth	precip	pres	uv
0	2024-04-09	30	16.70	24.50	5.80	154	2.40	3.50	0	0	0.00	998.20	3.00
1	2024-04-10	79	15.00	17.60	11.90	196	3.50	5.30	0	0	6.80	997.40	2.20
2	2024-04-11	88	15.80	20.00	11.10	133	7.20	10.70	0	0	6.15	992.40	2.30
3	2024-04-12	79	14.60	18.00	7.70	203	10.80	16.40	0	0	11.50	975.90	3.50
4	2024-04-13	76	6.90	9.20	4.60	271	9.80	14.60	0	0	4.35	987.40	3.50
5	2024-04-14	65	9.80	15.00	5.50	232	5.40	7.90	0	0	11.35	991.20	4.30
6	2024-04-15	62	11.60	16.10	7.60	269	4.00	4.00	0	0	33.50	994.10	8.10
7	2024-04-16	21	12.70	17.00	8.10	250	2.80	2.80	0	0	29.50	1,006.30	8.10
8	2024-04-17	38	14.20	18.60	10.60	138	3.50	3.50	0	0	67.00	1,010.10	8.20
9	2024-04-18	94	13.50	17.40	9.80	223	3.60	3.60	0	0	64.00	1,006.90	8.30
10	2024-04-19	84	12.40	16.70	8.90	171	4.20	4.60	0	0	42.56	1,003.50	8.30
11	2024-04-20	49	10.10	19.80	4.60	227	3.40	6.50	0	0	25.00	1,007.50	8.40
12	2024-04-21	90	12.20	17.90	6.80	110	1.90	2.70	0	0	13.19	1,002.80	8.40
13	2024-04-22	72	12.40	18.60	8.50	131	5.50	10.90	0	0	17.38	999.50	8.50
14	2024-04-23	91	10.60	19.10	4.40	236	3.50	7.50	0	0	27.38	1,000.00	8.60
15	2024-04-24	37	7.10	15.30	2.20	301	3.60	6.20	0	0	24.25	1,006.80	8.60

Initial predictions

In [55]:
```
# Reading flight data for initial predictions
initial_flight = pd.read_csv(r"C:\Users\VIKAS\Documents\Intro to ML\Project\datasets\flight_initials.csv",
                             parse_dates = ['Date', 'Scheduled Arrival Time', 'Scheduled departure time'])

initial_flight
```

Out[55]:

	Date	Carrier	Flight_Num	Origin	Dest	Scheduled Arrival Time	Scheduled departure time
0	2024-04-10	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
1	2024-04-10	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
2	2024-04-10	B6	116	JFK	SYR	2024-05-02 14:50:00	2024-05-02 13:33:00
3	2024-04-10	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
4	2024-04-10	WN	491	MCO	SYR	2024-05-02 13:45:00	2024-05-02 11:05:00
5	2024-04-10	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
6	2024-04-11	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
7	2024-04-11	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
8	2024-04-11	B6	116	JFK	SYR	2024-05-02 14:50:00	2024-05-02 13:33:00
9	2024-04-11	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
10	2024-04-11	WN	491	MCO	SYR	2024-05-02 14:20:00	2024-05-02 11:35:00
11	2024-04-11	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
12	2024-04-12	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
13	2024-04-12	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
14	2024-04-12	B6	116	JFK	SYR	2024-05-02 14:50:00	2024-05-02 13:33:00
15	2024-04-12	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
16	2024-04-12	WN	491	MCO	SYR	2024-05-02 14:20:00	2024-05-02 11:35:00
17	2024-04-12	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
18	2024-04-13	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
19	2024-04-13	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
20	2024-04-13	B6	116	JFK	SYR	2024-05-02 14:30:00	2024-05-02 13:12:00
21	2024-04-13	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
22	2024-04-13	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00

In [56]:

```
initial_flight['Date_MM'] = initial_flight['Date'].dt.month
initial_flight['Date_DD'] = initial_flight['Date'].dt.day

initial_flight['Flight_num'] = initial_flight['Flight_Num']

initial_flight['Arr_t_h'] = initial_flight['Scheduled Arrival Time'].dt.hour
initial_flight['Arr_t_m'] = initial_flight['Scheduled Arrival Time'].dt.minute
initial_flight['Depart_t_h'] = initial_flight['Scheduled departure time'].dt.hour
initial_flight['Depart_t_m'] = initial_flight['Scheduled departure time'].dt.minute

initial_flight.drop(columns = ['Scheduled Arrival Time', 'Scheduled departure time', 'Flight_Num'],
                    inplace = True)

initial_flight
```

Out[56]:

	Date	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m
0	2024-04-10	UA	ORD	SYR	4	10	538	21	47	18	52
1	2024-04-10	MQ	ORD	SYR	4	10	3402	22	52	19	59
2	2024-04-10	B6	JFK	SYR	4	10	116	14	50	13	33
3	2024-04-10	9E	JFK	SYR	4	10	5340	16	21	14	55
4	2024-04-10	WN	MCO	SYR	4	10	491	13	45	11	5
5	2024-04-10	B6	MCO	SYR	4	10	656	16	25	13	35
6	2024-04-11	UA	ORD	SYR	4	11	538	21	47	18	52
7	2024-04-11	MQ	ORD	SYR	4	11	3402	22	52	19	59
8	2024-04-11	B6	JFK	SYR	4	11	116	14	50	13	33
9	2024-04-11	9E	JFK	SYR	4	11	5340	16	21	14	55
10	2024-04-11	WN	MCO	SYR	4	11	491	14	20	11	35
11	2024-04-11	B6	MCO	SYR	4	11	656	16	25	13	35
12	2024-04-12	UA	ORD	SYR	4	12	538	21	47	18	52
13	2024-04-12	MQ	ORD	SYR	4	12	3402	22	52	19	59
14	2024-04-12	B6	JFK	SYR	4	12	116	14	50	13	33
15	2024-04-12	9E	JFK	SYR	4	12	5340	16	21	14	55
16	2024-04-12	WN	MCO	SYR	4	12	491	14	20	11	35
17	2024-04-12	B6	MCO	SYR	4	12	656	16	25	13	35
18	2024-04-13	UA	ORD	SYR	4	13	538	21	47	18	52
19	2024-04-13	MQ	ORD	SYR	4	13	3402	22	52	19	59
20	2024-04-13	B6	JFK	SYR	4	13	116	14	30	13	12
21	2024-04-13	9E	JFK	SYR	4	13	5340	16	21	14	55
22	2024-04-13	B6	MCO	SYR	4	13	656	16	25	13	35

In [57]:

```
# Merging flight data and tesing weather data for initial predictions based on dates
test_initials = pd.merge(initial_flight, test_weather, left_on='Date', right_on='datetime')
test_initials.drop(columns = ['datetime', 'Date'], inplace=True)

test_initials
```

Out[57]:

	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	...	wind_spd	wind_gust_spc
0	UA	ORD	SYR	4	10	538	21	47	18	52	...	3.50	5.3(
1	MQ	ORD	SYR	4	10	3402	22	52	19	59	...	3.50	5.3(
2	B6	JFK	SYR	4	10	116	14	50	13	33	...	3.50	5.3(
3	9E	JFK	SYR	4	10	5340	16	21	14	55	...	3.50	5.3(
4	WN	MCO	SYR	4	10	491	13	45	11	5	...	3.50	5.3(
5	B6	MCO	SYR	4	10	656	16	25	13	35	...	3.50	5.3(
6	UA	ORD	SYR	4	11	538	21	47	18	52	...	7.20	10.7(
7	MQ	ORD	SYR	4	11	3402	22	52	19	59	...	7.20	10.7(
8	B6	JFK	SYR	4	11	116	14	50	13	33	...	7.20	10.7(
9	9E	JFK	SYR	4	11	5340	16	21	14	55	...	7.20	10.7(
10	WN	MCO	SYR	4	11	491	14	20	11	35	...	7.20	10.7(
11	B6	MCO	SYR	4	11	656	16	25	13	35	...	7.20	10.7(
12	UA	ORD	SYR	4	12	538	21	47	18	52	...	10.80	16.4(
13	MQ	ORD	SYR	4	12	3402	22	52	19	59	...	10.80	16.4(
14	B6	JFK	SYR	4	12	116	14	50	13	33	...	10.80	16.4(
15	9E	JFK	SYR	4	12	5340	16	21	14	55	...	10.80	16.4(
16	WN	MCO	SYR	4	12	491	14	20	11	35	...	10.80	16.4(
17	B6	MCO	SYR	4	12	656	16	25	13	35	...	10.80	16.4(
18	UA	ORD	SYR	4	13	538	21	47	18	52	...	9.80	14.6(
19	MQ	ORD	SYR	4	13	3402	22	52	19	59	...	9.80	14.6(
20	B6	JFK	SYR	4	13	116	14	30	13	12	...	9.80	14.6(
21	9E	JFK	SYR	4	13	5340	16	21	14	55	...	9.80	14.6(
22	B6	MCO	SYR	4	13	656	16	25	13	35	...	9.80	14.6(

23 rows × 25 columns

In [58]:

```
test_initials = pd.get_dummies(test_initials, columns = ['Carrier', 'Origin','Dest'],
                               drop_first = True)

test_initials
```

Out[58]:

	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	clouds	temp	max_temp	...	uv	dewpt	rh	
0	4	10	538	21	47	18	52	79	15.00	17.60	...	2.20	10.20	73	1,0
1	4	10	3402	22	52	19	59	79	15.00	17.60	...	2.20	10.20	73	1,0
2	4	10	116	14	50	13	33	79	15.00	17.60	...	2.20	10.20	73	1,0
3	4	10	5340	16	21	14	55	79	15.00	17.60	...	2.20	10.20	73	1,0
4	4	10	491	13	45	11	5	79	15.00	17.60	...	2.20	10.20	73	1,0
5	4	10	656	16	25	13	35	79	15.00	17.60	...	2.20	10.20	73	1,0
6	4	11	538	21	47	18	52	88	15.80	20.00	...	2.30	12.00	78	1,0
7	4	11	3402	22	52	19	59	88	15.80	20.00	...	2.30	12.00	78	1,0
8	4	11	116	14	50	13	33	88	15.80	20.00	...	2.30	12.00	78	1,0
9	4	11	5340	16	21	14	55	88	15.80	20.00	...	2.30	12.00	78	1,0
10	4	11	491	14	20	11	35	88	15.80	20.00	...	2.30	12.00	78	1,0
11	4	11	656	16	25	13	35	88	15.80	20.00	...	2.30	12.00	78	1,0
12	4	12	538	21	47	18	52	79	14.60	18.00	...	3.50	9.70	73	9
13	4	12	3402	22	52	19	59	79	14.60	18.00	...	3.50	9.70	73	9
14	4	12	116	14	50	13	33	79	14.60	18.00	...	3.50	9.70	73	9
15	4	12	5340	16	21	14	55	79	14.60	18.00	...	3.50	9.70	73	9
16	4	12	491	14	20	11	35	79	14.60	18.00	...	3.50	9.70	73	9
17	4	12	656	16	25	13	35	79	14.60	18.00	...	3.50	9.70	73	9
18	4	13	538	21	47	18	52	76	6.90	9.20	...	3.50	2.90	76	1,0
19	4	13	3402	22	52	19	59	76	6.90	9.20	...	3.50	2.90	76	1,0
20	4	13	116	14	30	13	12	76	6.90	9.20	...	3.50	2.90	76	1,0
21	4	13	5340	16	21	14	55	76	6.90	9.20	...	3.50	2.90	76	1,0
22	4	13	656	16	25	13	35	76	6.90	9.20	...	3.50	2.90	76	1,0

23 rows × 28 columns

Predicting initial test data using pre-trained model - Random Forest Classifier

In [59]:
```
y1_pred = rf_classifier.predict(test_initials)
#y1_pred = gb.predict(test_initials)
#y1_pred = xgb_classifier.predict(test_initials)
#y1_pred = Log_reg.predict(test_initials)
y1_pred
```

Out[59]:
```
array([0, 0, 2, 2, 0, 2, 0, 0, 2, 2, 2, 2, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0,
       2], dtype=int64)
```

In [60]:
```
label_mapping = {
    0: "Early",
    1: "On Time",
    2: "Late"
}
```

In [61]:
```
predictions = np.vectorize(label_mapping.get)(y1_pred)
predictions
```

Out[61]:
```
array(['Early', 'Early', 'Late', 'Late', 'Early', 'Late', 'Early',
       'Early', 'Late', 'Late', 'Late', 'Late', 'Early', 'Early', 'Late',
       'Early', 'Late', 'Late', 'Early', 'Early', 'Late', 'Early', 'Late'],
      dtype='<U5')
```

Final predictions

In [62]:
```
# Reading flight data for final predictions
final_flight = pd.read_csv(r"C:\Users\VIKAS\Documents\Intro to ML\Project\datasets\flight_finals.csv",
                           parse_dates = ['Date', 'Scheduled Arrival Time',
                                          'Scheduled departure time'])

final_flight
```

Out[62]:

	Date	Carrier	Flight_Num	Origin	Dest	Scheduled Arrival Time	Scheduled departure time
0	2024-04-19	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
1	2024-04-19	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
2	2024-04-19	B6	116	JFK	SYR	2024-05-02 14:51:00	2024-05-02 13:34:00
3	2024-04-19	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
4	2024-04-19	WN	491	MCO	SYR	2024-05-02 14:20:00	2024-05-02 11:35:00
5	2024-04-19	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
6	2024-04-20	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
7	2024-04-20	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
8	2024-04-20	B6	116	JFK	SYR	2024-05-02 14:41:00	2024-05-02 13:25:00
9	2024-04-20	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
10	2024-04-20	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
11	2024-04-21	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
12	2024-04-21	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
13	2024-04-21	B6	116	JFK	SYR	2024-05-02 14:51:00	2024-05-02 13:35:00
14	2024-04-21	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
15	2024-04-21	WN	491	MCO	SYR	2024-05-02 13:50:00	2024-05-02 11:05:00
16	2024-04-21	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
17	2024-04-22	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
18	2024-04-22	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
19	2024-04-22	B6	116	JFK	SYR	2024-05-02 14:51:00	2024-05-02 13:35:00
20	2024-04-22	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
21	2024-04-22	WN	491	MCO	SYR	2024-05-02 14:20:00	2024-05-02 11:35:00
22	2024-04-22	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:34:00

In [63]:

```
final_flight['Date_MM'] = final_flight['Date'].dt.month
final_flight['Date_DD'] = final_flight['Date'].dt.day

final_flight['Flight_num'] = final_flight['Flight_Num']


final_flight['Arr_t_h'] = final_flight['Scheduled Arrival Time'].dt.hour
final_flight['Arr_t_m'] = final_flight['Scheduled Arrival Time'].dt.minute
final_flight['Depart_t_h'] = final_flight['Scheduled departure time'].dt.hour
final_flight['Depart_t_m'] = final_flight['Scheduled departure time'].dt.minute

final_flight.drop(columns = ['Scheduled Arrival Time', 'Scheduled departure time', 'Flight_Num'],
                  inplace = True)

final_flight
```

Out[63]:

	Date	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m
0	2024-04-19	UA	ORD	SYR	4	19	538	21	47	18	52
1	2024-04-19	MQ	ORD	SYR	4	19	3402	22	52	19	59
2	2024-04-19	B6	JFK	SYR	4	19	116	14	51	13	34
3	2024-04-19	9E	JFK	SYR	4	19	5340	16	21	14	55
4	2024-04-19	WN	MCO	SYR	4	19	491	14	20	11	35
5	2024-04-19	B6	MCO	SYR	4	19	656	16	25	13	35
6	2024-04-20	UA	ORD	SYR	4	20	538	21	47	18	52
7	2024-04-20	MQ	ORD	SYR	4	20	3402	22	52	19	59
8	2024-04-20	B6	JFK	SYR	4	20	116	14	41	13	25
9	2024-04-20	9E	JFK	SYR	4	20	5340	16	21	14	55
10	2024-04-20	B6	MCO	SYR	4	20	656	16	25	13	35
11	2024-04-21	UA	ORD	SYR	4	21	538	21	47	18	52
12	2024-04-21	MQ	ORD	SYR	4	21	3402	22	52	19	59
13	2024-04-21	B6	JFK	SYR	4	21	116	14	51	13	35
14	2024-04-21	9E	JFK	SYR	4	21	5340	16	21	14	55
15	2024-04-21	WN	MCO	SYR	4	21	491	13	50	11	5
16	2024-04-21	B6	MCO	SYR	4	21	656	16	25	13	35
17	2024-04-22	UA	ORD	SYR	4	22	538	21	47	18	52
18	2024-04-22	MQ	ORD	SYR	4	22	3402	22	52	19	59
19	2024-04-22	B6	JFK	SYR	4	22	116	14	51	13	35
20	2024-04-22	9E	JFK	SYR	4	22	5340	16	21	14	55
21	2024-04-22	WN	MCO	SYR	4	22	491	14	20	11	35
22	2024-04-22	B6	MCO	SYR	4	22	656	16	25	13	34

In [64]:

```
# Merging flight data and tesing weather data for final predictions based on dates
test_finals = pd.merge(final_flight, test_weather, left_on='Date', right_on='datetime')
test_finals.drop(columns = ['datetime', 'Date'], inplace=True)

test_finals
```

Out[64]:

	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	...	wind_spd	wind_gust_spc
0	UA	ORD	SYR	4	19	538	21	47	18	52	...	4.20	4.6(
1	MQ	ORD	SYR	4	19	3402	22	52	19	59	...	4.20	4.6(
2	B6	JFK	SYR	4	19	116	14	51	13	34	...	4.20	4.6(
3	9E	JFK	SYR	4	19	5340	16	21	14	55	...	4.20	4.6(
4	WN	MCO	SYR	4	19	491	14	20	11	35	...	4.20	4.6(
5	B6	MCO	SYR	4	19	656	16	25	13	35	...	4.20	4.6(
6	UA	ORD	SYR	4	20	538	21	47	18	52	...	3.40	6.5(
7	MQ	ORD	SYR	4	20	3402	22	52	19	59	...	3.40	6.5(
8	B6	JFK	SYR	4	20	116	14	41	13	25	...	3.40	6.5(
9	9E	JFK	SYR	4	20	5340	16	21	14	55	...	3.40	6.5(
10	B6	MCO	SYR	4	20	656	16	25	13	35	...	3.40	6.5(
11	UA	ORD	SYR	4	21	538	21	47	18	52	...	1.90	2.7(
12	MQ	ORD	SYR	4	21	3402	22	52	19	59	...	1.90	2.7(
13	B6	JFK	SYR	4	21	116	14	51	13	35	...	1.90	2.7(
14	9E	JFK	SYR	4	21	5340	16	21	14	55	...	1.90	2.7(
15	WN	MCO	SYR	4	21	491	13	50	11	5	...	1.90	2.7(
16	B6	MCO	SYR	4	21	656	16	25	13	35	...	1.90	2.7(
17	UA	ORD	SYR	4	22	538	21	47	18	52	...	5.50	10.9(
18	MQ	ORD	SYR	4	22	3402	22	52	19	59	...	5.50	10.9(
19	B6	JFK	SYR	4	22	116	14	51	13	35	...	5.50	10.9(
20	9E	JFK	SYR	4	22	5340	16	21	14	55	...	5.50	10.9(
21	WN	MCO	SYR	4	22	491	14	20	11	35	...	5.50	10.9(
22	B6	MCO	SYR	4	22	656	16	25	13	34	...	5.50	10.9(

23 rows × 25 columns

In [65]:
```
test_finals = pd.get_dummies(test_finals, columns = ['Carrier', 'Origin','Dest'], drop_first = True)

test_finals
```

Out[65]:

	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	clouds	temp	max_temp	...	uv	dewpt	rh	
0	4	19	538	21	47	18	52	84	12.40	16.70	...	8.30	3.90	57	1,0
1	4	19	3402	22	52	19	59	84	12.40	16.70	...	8.30	3.90	57	1,0
2	4	19	116	14	51	13	34	84	12.40	16.70	...	8.30	3.90	57	1,0
3	4	19	5340	16	21	14	55	84	12.40	16.70	...	8.30	3.90	57	1,0
4	4	19	491	14	20	11	35	84	12.40	16.70	...	8.30	3.90	57	1,0
5	4	19	656	16	25	13	35	84	12.40	16.70	...	8.30	3.90	57	1,0
6	4	20	538	21	47	18	52	49	10.10	19.80	...	8.40	6.30	78	1,0
7	4	20	3402	22	52	19	59	49	10.10	19.80	...	8.40	6.30	78	1,0
8	4	20	116	14	41	13	25	49	10.10	19.80	...	8.40	6.30	78	1,0
9	4	20	5340	16	21	14	55	49	10.10	19.80	...	8.40	6.30	78	1,0
10	4	20	656	16	25	13	35	49	10.10	19.80	...	8.40	6.30	78	1,0
11	4	21	538	21	47	18	52	90	12.20	17.90	...	8.40	9.60	85	1,0
12	4	21	3402	22	52	19	59	90	12.20	17.90	...	8.40	9.60	85	1,0
13	4	21	116	14	51	13	35	90	12.20	17.90	...	8.40	9.60	85	1,0
14	4	21	5340	16	21	14	55	90	12.20	17.90	...	8.40	9.60	85	1,0
15	4	21	491	13	50	11	5	90	12.20	17.90	...	8.40	9.60	85	1,0
16	4	21	656	16	25	13	35	90	12.20	17.90	...	8.40	9.60	85	1,0
17	4	22	538	21	47	18	52	72	12.40	18.60	...	8.50	8.50	78	1,0
18	4	22	3402	22	52	19	59	72	12.40	18.60	...	8.50	8.50	78	1,0
19	4	22	116	14	51	13	35	72	12.40	18.60	...	8.50	8.50	78	1,0
20	4	22	5340	16	21	14	55	72	12.40	18.60	...	8.50	8.50	78	1,0
21	4	22	491	14	20	11	35	72	12.40	18.60	...	8.50	8.50	78	1,0
22	4	22	656	16	25	13	34	72	12.40	18.60	...	8.50	8.50	78	1,0

23 rows × 28 columns

Predicting final test data using pre-trained model - Random Forest Classifier

In [66]:
```
y2_pred = rf_classifier.predict(test_finals)
#y2_pred = gb.predict(test_finals)
#y2_pred = xgb_classifier.predict(test_finals)
#y2_pred = Log_reg.predict(test_finals)

y2_pred
```

Out[66]:
```
array([0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 2, 0, 2, 2, 2, 2, 2, 2,
       2], dtype=int64)
```

In [67]:
```
predictions = np.vectorize(label_mapping.get)(y2_pred)
predictions
```

Out[67]:
```
array(['Early', 'Early', 'Early', 'Late', 'Early', 'Early', 'Early',
       'Early', 'Early', 'Early', 'Late', 'Early', 'Early', 'Late',
       'Late', 'Early', 'Late', 'Late', 'Late', 'Late', 'Late', 'Late',
       'Late'], dtype='<U5')
```

Latter flights

In [68]:
```
# Splitting dataset into first and second flights according to given conditions
First_flights = Flight_data[Flight_data['Flight_num'].isin({538, 116, 491})]
First_flights['Carrier1'] = First_flights['Carrier']
First_flights['Flight_num1'] = First_flights['Flight_num']
First_flights['Arr_Status1'] = First_flights['Arr_Status']
First_flights = First_flights[['Date', 'Carrier1', 'Flight_num1', 'Origin', 'Arr_Status1']]

First_flights.head()
```

```
<ipython-input-68-95a2ae905f3c>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  First_flights['Carrier1'] = First_flights['Carrier']
<ipython-input-68-95a2ae905f3c>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  First_flights['Flight_num1'] = First_flights['Flight_num']
<ipython-input-68-95a2ae905f3c>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  First_flights['Arr_Status1'] = First_flights['Arr_Status']
```

Out[68]:

	Date	Carrier1	Flight_num1	Origin	Arr_Status1
555	2023-01-01	B6	116	JFK	2
558	2023-01-02	B6	116	JFK	2
561	2023-01-03	B6	116	JFK	2
564	2023-01-04	B6	116	JFK	2
567	2023-01-05	B6	116	JFK	0

In [69]:
```
Second_flights = Flight_data[Flight_data['Flight_num'].isin({3402, 5340, 656})]

Second_flights.head()
```

Out[69]:

	Date	Date_MM	Date_DD	Carrier	Flight_num	Origin	Dest	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Arr_Status
71	2023-06-01	6	1	MQ	3402	ORD	SYR	22	32	19	42	2
73	2023-06-02	6	2	MQ	3402	ORD	SYR	22	32	19	42	1
75	2023-06-03	6	3	MQ	3402	ORD	SYR	23	41	20	46	0
77	2023-06-04	6	4	MQ	3402	ORD	SYR	22	28	19	39	1
79	2023-06-05	6	5	MQ	3402	ORD	SYR	22	28	19	39	0

In [70]:
```
#merging first and second flights based on origin and date
Latter_flight = pd.merge(Second_flights, First_flights, on = ['Origin', 'Date'])

Latter_flight.head()
```

Out[70]:

	Date	Date_MM	Date_DD	Carrier	Flight_num	Origin	Dest	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Arr_Status	Carrier1	F
0	2023-06-01	6	1	MQ	3402	ORD	SYR	22	32	19	42	2	UA	
1	2023-06-05	6	5	MQ	3402	ORD	SYR	22	28	19	39	0	UA	
2	2023-06-06	6	6	MQ	3402	ORD	SYR	22	28	19	39	0	UA	
3	2023-06-07	6	7	MQ	3402	ORD	SYR	22	28	19	39	1	UA	
4	2023-06-08	6	8	MQ	3402	ORD	SYR	22	28	19	39	1	UA	

In [71]:
```
# merging latter flight data syracuse weather data based on date
Latter_Flight_weather = pd.merge(Latter_flight, SYR_weather, left_on='Date', right_on='datetime')

# arrival status of second flights as Arr_Status2
Latter_Flight_weather['Arr_Status2'] = Latter_Flight_weather['Arr_Status']
Latter_Flight_weather.drop(columns = ['datetime', 'Date', 'Arr_Status'], inplace=True)

Latter_Flight_weather.head()
```

Out[71]:

	Date_MM	Date_DD	Carrier	Flight_num	Origin	Dest	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	...	wind_gust_spd	snow	sn
0	6	1	MQ	3402	ORD	SYR	22	32	19	42	...	7.60	0.00	
1	6	1	9E	5340	JFK	SYR	0	4	22	30	...	7.60	0.00	
2	6	5	MQ	3402	ORD	SYR	22	28	19	39	...	12.40	0.00	
3	6	6	MQ	3402	ORD	SYR	22	28	19	39	...	15.60	0.00	
4	6	7	MQ	3402	ORD	SYR	22	28	19	39	...	12.80	0.00	

5 rows × 29 columns

In [72]:
```
Latter_Flight_weather.columns
```

Out[72]:
```
Index(['Date_MM', 'Date_DD', 'Carrier', 'Flight_num', 'Origin', 'Dest',
       'Arr_t_h', 'Arr_t_m', 'Depart_t_h', 'Depart_t_m', 'Carrier1',
       'Flight_num1', 'Arr_Status1', 'clouds', 'temp', 'max_temp', 'min_temp',
       'wind_dir', 'wind_spd', 'wind_gust_spd', 'snow', 'snow_depth', 'precip',
       'pres', 'uv', 'dewpt', 'rh', 'slp', 'Arr_Status2'],
      dtype='object')
```

In [73]:
```
Latter_Flight_weather.isna().sum()
```

Out[73]:
```
Date_MM          0
Date_DD          0
Carrier          0
Flight_num       0
Origin           0
Dest             0
Arr_t_h          0
Arr_t_m          0
Depart_t_h       0
Depart_t_m       0
Carrier1         0
Flight_num1      0
Arr_Status1      0
clouds           0
temp             0
max_temp         0
min_temp         0
wind_dir         0
wind_spd         0
wind_gust_spd    0
snow             0
snow_depth       0
precip           0
pres             0
uv               0
dewpt            0
rh               0
slp              0
Arr_Status2      0
dtype: int64
```

In [74]:
```
Latter_Flight_weather.dtypes
```

Out[74]:
```
Date_MM           int64
Date_DD           int64
Carrier          object
Flight_num        int64
Origin           object
Dest             object
Arr_t_h           int64
Arr_t_m           int64
Depart_t_h        int64
Depart_t_m        int64
Carrier1         object
Flight_num1       int64
Arr_Status1       int64
clouds            int64
temp            float64
max_temp        float64
min_temp        float64
wind_dir          int64
wind_spd        float64
wind_gust_spd   float64
snow            float64
snow_depth      float64
precip          float64
pres              int64
uv              float64
dewpt           float64
rh                int64
slp               int64
Arr_Status2       int64
dtype: object
```

In [75]:
```
Latter_Flight_weather = pd.get_dummies(Latter_Flight_weather,
                                       columns = ['Carrier', 'Origin','Carrier1', 'Dest'],
                                       drop_first = True)

Latter_Flight_weather.head()
```

Out[75]:

	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Flight_num1	Arr_Status1	clouds	...	dewpt	rh	
0	6	1	3402	22	32	19	42	538	0	26	...	10.80	47	1(
1	6	1	5340	0	4	22	30	116	0	26	...	10.80	47	1(
2	6	5	3402	22	28	19	39	538	0	80	...	5.10	50	1(
3	6	6	3402	22	28	19	39	538	2	94	...	6.90	54	1(
4	6	7	3402	22	28	19	39	538	1	99	...	5.20	56	1(

5 rows × 31 columns

In [76]:
```
X_train, X_test, y_train, y_test = train_test_split(Latter_Flight_weather.drop(columns = ['Arr_Status2']),
                                                    Latter_Flight_weather['Arr_Status2'], test_size=0.2,
                                                    random_state=42)


X_train
X_test
y_train
y_test
```

Out[76]:
```
100    2
10     0
4      1
81     0
97     2
65     2
30     0
33     0
93     1
11     1
47     2
42     2
0      2
78     2
18     2
64     2
67     0
79     0
55     2
44     0
12     0
80     1
Name: Arr_Status2, dtype: int64
```

In [77]:
```
if False:
    from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    X_train = pd.DataFrame(sc.fit_transform(X_train), columns = X_train.columns, index = X_train.index)
    X_test = pd.DataFrame(sc.transform(X_test), columns = X_test.columns, index = X_test.index)

    X_train
    X_test
    y_train
    y_test
```

Random Forest Classifier

In [78]:
```
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)

y_pred = rf_classifier.predict(X_test)

rf_classifier.score(X_train, y_train)

print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

```
Accuracy: 0.3181818181818182
```

Gradient Boosting classifier

In [79]:
```
gb = GradientBoostingClassifier(random_state=50, min_samples_split = 12, min_samples_leaf = 6,
                                max_depth = 4, n_estimators = 100)
gb = gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)

gb.score(X_train, y_train)

print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

```
Accuracy: 0.36363636363636365
```

XGBoost classifier

In [80]:
```
xgb_classifier = xgb.XGBClassifier(objective='multi:softmax', num_class=3, random_state=42)
xgb_classifier.fit(X_train, y_train)

y_pred = xgb_classifier.predict(X_test)

print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

```
Accuracy: 0.4090909090909091
```

Initial predictions for Latter flights

In [81]:
```
initial_flight = pd.read_csv(r"C:\Users\VIKAS\Documents\Intro to ML\Project\datasets\flight_initials.csv"
                             parse_dates = ['Date', 'Scheduled Arrival Time',
                                            'Scheduled departure time'])

initial_flight
```

Out[81]:

	Date	Carrier	Flight_Num	Origin	Dest	Scheduled Arrival Time	Scheduled departure time
0	2024-04-10	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
1	2024-04-10	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
2	2024-04-10	B6	116	JFK	SYR	2024-05-02 14:50:00	2024-05-02 13:33:00
3	2024-04-10	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
4	2024-04-10	WN	491	MCO	SYR	2024-05-02 13:45:00	2024-05-02 11:05:00
5	2024-04-10	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
6	2024-04-11	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
7	2024-04-11	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
8	2024-04-11	B6	116	JFK	SYR	2024-05-02 14:50:00	2024-05-02 13:33:00
9	2024-04-11	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
10	2024-04-11	WN	491	MCO	SYR	2024-05-02 14:20:00	2024-05-02 11:35:00
11	2024-04-11	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
12	2024-04-12	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
13	2024-04-12	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
14	2024-04-12	B6	116	JFK	SYR	2024-05-02 14:50:00	2024-05-02 13:33:00
15	2024-04-12	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
16	2024-04-12	WN	491	MCO	SYR	2024-05-02 14:20:00	2024-05-02 11:35:00
17	2024-04-12	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
18	2024-04-13	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
19	2024-04-13	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
20	2024-04-13	B6	116	JFK	SYR	2024-05-02 14:30:00	2024-05-02 13:12:00
21	2024-04-13	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
22	2024-04-13	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00

In [82]:

```
initial_flight['Date_MM'] = initial_flight['Date'].dt.month
initial_flight['Date_DD'] = initial_flight['Date'].dt.day

initial_flight['Flight_num'] = initial_flight['Flight_Num']

initial_flight['Arr_t_h'] = initial_flight['Scheduled Arrival Time'].dt.hour
initial_flight['Arr_t_m'] = initial_flight['Scheduled Arrival Time'].dt.minute
initial_flight['Depart_t_h'] = initial_flight['Scheduled departure time'].dt.hour
initial_flight['Depart_t_m'] = initial_flight['Scheduled departure time'].dt.minute

initial_flight.drop(columns = ['Scheduled Arrival Time', 'Scheduled departure time', 'Flight_Num'],
                    inplace = True)

initial_flight
```

Out[82]:

	Date	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m
0	2024-04-10	UA	ORD	SYR	4	10	538	21	47	18	52
1	2024-04-10	MQ	ORD	SYR	4	10	3402	22	52	19	59
2	2024-04-10	B6	JFK	SYR	4	10	116	14	50	13	33
3	2024-04-10	9E	JFK	SYR	4	10	5340	16	21	14	55
4	2024-04-10	WN	MCO	SYR	4	10	491	13	45	11	5
5	2024-04-10	B6	MCO	SYR	4	10	656	16	25	13	35
6	2024-04-11	UA	ORD	SYR	4	11	538	21	47	18	52
7	2024-04-11	MQ	ORD	SYR	4	11	3402	22	52	19	59
8	2024-04-11	B6	JFK	SYR	4	11	116	14	50	13	33
9	2024-04-11	9E	JFK	SYR	4	11	5340	16	21	14	55
10	2024-04-11	WN	MCO	SYR	4	11	491	14	20	11	35
11	2024-04-11	B6	MCO	SYR	4	11	656	16	25	13	35
12	2024-04-12	UA	ORD	SYR	4	12	538	21	47	18	52
13	2024-04-12	MQ	ORD	SYR	4	12	3402	22	52	19	59
14	2024-04-12	B6	JFK	SYR	4	12	116	14	50	13	33
15	2024-04-12	9E	JFK	SYR	4	12	5340	16	21	14	55
16	2024-04-12	WN	MCO	SYR	4	12	491	14	20	11	35
17	2024-04-12	B6	MCO	SYR	4	12	656	16	25	13	35
18	2024-04-13	UA	ORD	SYR	4	13	538	21	47	18	52
19	2024-04-13	MQ	ORD	SYR	4	13	3402	22	52	19	59
20	2024-04-13	B6	JFK	SYR	4	13	116	14	30	13	12
21	2024-04-13	9E	JFK	SYR	4	13	5340	16	21	14	55
22	2024-04-13	B6	MCO	SYR	4	13	656	16	25	13	35

In [83]:
```
First_flights = initial_flight[initial_flight['Flight_num'].isin({538, 116, 491})]
First_flights['Carrier1'] = First_flights['Carrier']
First_flights['Flight_num1'] = First_flights['Flight_num']
First_flights = First_flights[['Date', 'Carrier1', 'Flight_num1', 'Origin']]

First_flights
```

```
<ipython-input-83-f9cdab8bc466>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  First_flights['Carrier1'] = First_flights['Carrier']
<ipython-input-83-f9cdab8bc466>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  First_flights['Flight_num1'] = First_flights['Flight_num']
```

Out[83]:

	Date	Carrier1	Flight_num1	Origin
0	2024-04-10	UA	538	ORD
2	2024-04-10	B6	116	JFK
4	2024-04-10	WN	491	MCO
6	2024-04-11	UA	538	ORD
8	2024-04-11	B6	116	JFK
10	2024-04-11	WN	491	MCO
12	2024-04-12	UA	538	ORD
14	2024-04-12	B6	116	JFK
16	2024-04-12	WN	491	MCO
18	2024-04-13	UA	538	ORD
20	2024-04-13	B6	116	JFK

In [84]:
```
Second_flights = initial_flight[initial_flight['Flight_num'].isin({3402, 5340, 656})]

Second_flights
```

Out[84]:

	Date	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m
1	2024-04-10	MQ	ORD	SYR	4	10	3402	22	52	19	59
3	2024-04-10	9E	JFK	SYR	4	10	5340	16	21	14	55
5	2024-04-10	B6	MCO	SYR	4	10	656	16	25	13	35
7	2024-04-11	MQ	ORD	SYR	4	11	3402	22	52	19	59
9	2024-04-11	9E	JFK	SYR	4	11	5340	16	21	14	55
11	2024-04-11	B6	MCO	SYR	4	11	656	16	25	13	35
13	2024-04-12	MQ	ORD	SYR	4	12	3402	22	52	19	59
15	2024-04-12	9E	JFK	SYR	4	12	5340	16	21	14	55
17	2024-04-12	B6	MCO	SYR	4	12	656	16	25	13	35
19	2024-04-13	MQ	ORD	SYR	4	13	3402	22	52	19	59
21	2024-04-13	9E	JFK	SYR	4	13	5340	16	21	14	55
22	2024-04-13	B6	MCO	SYR	4	13	656	16	25	13	35

In [85]:
```
Latter_flight = pd.merge(Second_flights, First_flights, on = ['Origin', 'Date'])
Latter_flight['Arr_Status1'] = 0

Latter_flight
```

Out[85]:

	Date	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Carrier1	Flight_num1
0	2024-04-10	MQ	ORD	SYR	4	10	3402	22	52	19	59	UA	538
1	2024-04-10	9E	JFK	SYR	4	10	5340	16	21	14	55	B6	116
2	2024-04-10	B6	MCO	SYR	4	10	656	16	25	13	35	WN	491
3	2024-04-11	MQ	ORD	SYR	4	11	3402	22	52	19	59	UA	538
4	2024-04-11	9E	JFK	SYR	4	11	5340	16	21	14	55	B6	116
5	2024-04-11	B6	MCO	SYR	4	11	656	16	25	13	35	WN	491
6	2024-04-12	MQ	ORD	SYR	4	12	3402	22	52	19	59	UA	538
7	2024-04-12	9E	JFK	SYR	4	12	5340	16	21	14	55	B6	116
8	2024-04-12	B6	MCO	SYR	4	12	656	16	25	13	35	WN	491
9	2024-04-13	MQ	ORD	SYR	4	13	3402	22	52	19	59	UA	538
10	2024-04-13	9E	JFK	SYR	4	13	5340	16	21	14	55	B6	116

In [86]:
```
Latter_Flight_weather = pd.merge(Latter_flight, test_weather, left_on='Date', right_on='datetime')
Latter_Flight_weather.drop(columns = ['datetime', 'Date'], inplace=True)

Latter_Flight_weather
```

Out[86]:

	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	...	wind_spd	wind_gust_spd
0	MQ	ORD	SYR	4	10	3402	22	52	19	59	...	3.50	5.30
1	9E	JFK	SYR	4	10	5340	16	21	14	55	...	3.50	5.30
2	B6	MCO	SYR	4	10	656	16	25	13	35	...	3.50	5.30
3	MQ	ORD	SYR	4	11	3402	22	52	19	59	...	7.20	10.70
4	9E	JFK	SYR	4	11	5340	16	21	14	55	...	7.20	10.70
5	B6	MCO	SYR	4	11	656	16	25	13	35	...	7.20	10.70
6	MQ	ORD	SYR	4	12	3402	22	52	19	59	...	10.80	16.40
7	9E	JFK	SYR	4	12	5340	16	21	14	55	...	10.80	16.40
8	B6	MCO	SYR	4	12	656	16	25	13	35	...	10.80	16.40
9	MQ	ORD	SYR	4	13	3402	22	52	19	59	...	9.80	14.60
10	9E	JFK	SYR	4	13	5340	16	21	14	55	...	9.80	14.60

11 rows × 28 columns

In [87]:
```
Latter_Flight_weather = pd.get_dummies(Latter_Flight_weather,
                                       columns = ['Carrier', 'Origin','Carrier1', 'Dest'],
                                       drop_first = True)
Latter_Flight_weather
```

Out[87]:

	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Flight_num1	Arr_Status1	clouds	...	uv	dewpt
0	4	10	3402	22	52	19	59	538	0	79	...	2.20	10.20
1	4	10	5340	16	21	14	55	116	0	79	...	2.20	10.20
2	4	10	656	16	25	13	35	491	0	79	...	2.20	10.20
3	4	11	3402	22	52	19	59	538	0	88	...	2.30	12.00
4	4	11	5340	16	21	14	55	116	0	88	...	2.30	12.00
5	4	11	656	16	25	13	35	491	0	88	...	2.30	12.00
6	4	12	3402	22	52	19	59	538	0	79	...	3.50	9.70
7	4	12	5340	16	21	14	55	116	0	79	...	3.50	9.70
8	4	12	656	16	25	13	35	491	0	79	...	3.50	9.70
9	4	13	3402	22	52	19	59	538	0	76	...	3.50	2.90
10	4	13	5340	16	21	14	55	116	0	76	...	3.50	2.90

11 rows × 30 columns

Predicting initial test data of latter flights using pre-trained model - XGBoost classifier

In [88]:
```
Latter_Flight_weather['Arr_Status1'] = 0
y1_pred0 = xgb_classifier.predict(Latter_Flight_weather)

Latter_Flight_weather['Arr_Status1'] = 1
y1_pred1 = xgb_classifier.predict(Latter_Flight_weather)

Latter_Flight_weather['Arr_Status1'] = 2
y1_pred2 = xgb_classifier.predict(Latter_Flight_weather)

y1_pred = np.column_stack((y1_pred0, y1_pred1, y1_pred2))
y1_pred
```

Out[88]:
```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 2],
       [2, 2, 2],
       [2, 2, 2],
       [2, 2, 2],
       [1, 1, 1],
       [1, 1, 1],
       [2, 2, 2],
       [0, 0, 0],
       [0, 0, 0]])
```

In [89]:
```
predictions = np.vectorize(label_mapping.get)(y1_pred)
predictions
```

Out[89]:
```
array([['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'Late'],
       ['Late', 'Late', 'Late'],
       ['Late', 'Late', 'Late'],
       ['Late', 'Late', 'Late'],
       ['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'On Time'],
       ['Late', 'Late', 'Late'],
       ['Early', 'Early', 'Early'],
       ['Early', 'Early', 'Early']], dtype='<U7')
```

Final predictions for Latter flights

In [90]:
```
final_flight = pd.read_csv(r"C:\Users\VIKAS\Documents\Intro to ML\Project\datasets\flight_finals.csv",
                           parse_dates = ['Date', 'Scheduled Arrival Time', 'Scheduled departure time'])

final_flight
```

Out[90]:

	Date	Carrier	Flight_Num	Origin	Dest	Scheduled Arrival Time	Scheduled departure time
0	2024-04-19	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
1	2024-04-19	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
2	2024-04-19	B6	116	JFK	SYR	2024-05-02 14:51:00	2024-05-02 13:34:00
3	2024-04-19	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
4	2024-04-19	WN	491	MCO	SYR	2024-05-02 14:20:00	2024-05-02 11:35:00
5	2024-04-19	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
6	2024-04-20	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
7	2024-04-20	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
8	2024-04-20	B6	116	JFK	SYR	2024-05-02 14:41:00	2024-05-02 13:25:00
9	2024-04-20	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
10	2024-04-20	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
11	2024-04-21	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
12	2024-04-21	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
13	2024-04-21	B6	116	JFK	SYR	2024-05-02 14:51:00	2024-05-02 13:35:00
14	2024-04-21	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
15	2024-04-21	WN	491	MCO	SYR	2024-05-02 13:50:00	2024-05-02 11:05:00
16	2024-04-21	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:35:00
17	2024-04-22	UA	538	ORD	SYR	2024-05-02 21:47:00	2024-05-02 18:52:00
18	2024-04-22	MQ	3402	ORD	SYR	2024-05-02 22:52:00	2024-05-02 19:59:00
19	2024-04-22	B6	116	JFK	SYR	2024-05-02 14:51:00	2024-05-02 13:35:00
20	2024-04-22	9E	5340	JFK	SYR	2024-05-02 16:21:00	2024-05-02 14:55:00
21	2024-04-22	WN	491	MCO	SYR	2024-05-02 14:20:00	2024-05-02 11:35:00
22	2024-04-22	B6	656	MCO	SYR	2024-05-02 16:25:00	2024-05-02 13:34:00

In [91]:

```
final_flight['Date_MM'] = final_flight['Date'].dt.month
final_flight['Date_DD'] = final_flight['Date'].dt.day

final_flight['Flight_num'] = final_flight['Flight_Num']

final_flight['Arr_t_h'] = final_flight['Scheduled Arrival Time'].dt.hour
final_flight['Arr_t_m'] = final_flight['Scheduled Arrival Time'].dt.minute
final_flight['Depart_t_h'] = final_flight['Scheduled departure time'].dt.hour
final_flight['Depart_t_m'] = final_flight['Scheduled departure time'].dt.minute

final_flight.drop(columns = ['Scheduled Arrival Time', 'Scheduled departure time', 'Flight_Num'],
                  inplace = True)

final_flight
```

Out[91]:

	Date	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m
0	2024-04-19	UA	ORD	SYR	4	19	538	21	47	18	52
1	2024-04-19	MQ	ORD	SYR	4	19	3402	22	52	19	59
2	2024-04-19	B6	JFK	SYR	4	19	116	14	51	13	34
3	2024-04-19	9E	JFK	SYR	4	19	5340	16	21	14	55
4	2024-04-19	WN	MCO	SYR	4	19	491	14	20	11	35
5	2024-04-19	B6	MCO	SYR	4	19	656	16	25	13	35
6	2024-04-20	UA	ORD	SYR	4	20	538	21	47	18	52
7	2024-04-20	MQ	ORD	SYR	4	20	3402	22	52	19	59
8	2024-04-20	B6	JFK	SYR	4	20	116	14	41	13	25
9	2024-04-20	9E	JFK	SYR	4	20	5340	16	21	14	55
10	2024-04-20	B6	MCO	SYR	4	20	656	16	25	13	35
11	2024-04-21	UA	ORD	SYR	4	21	538	21	47	18	52
12	2024-04-21	MQ	ORD	SYR	4	21	3402	22	52	19	59
13	2024-04-21	B6	JFK	SYR	4	21	116	14	51	13	35
14	2024-04-21	9E	JFK	SYR	4	21	5340	16	21	14	55
15	2024-04-21	WN	MCO	SYR	4	21	491	13	50	11	5
16	2024-04-21	B6	MCO	SYR	4	21	656	16	25	13	35
17	2024-04-22	UA	ORD	SYR	4	22	538	21	47	18	52
18	2024-04-22	MQ	ORD	SYR	4	22	3402	22	52	19	59
19	2024-04-22	B6	JFK	SYR	4	22	116	14	51	13	35
20	2024-04-22	9E	JFK	SYR	4	22	5340	16	21	14	55
21	2024-04-22	WN	MCO	SYR	4	22	491	14	20	11	35
22	2024-04-22	B6	MCO	SYR	4	22	656	16	25	13	34

In [92]:
```
First_flights = final_flight[final_flight['Flight_num'].isin({538, 116, 491})]
First_flights['Carrier1'] = First_flights['Carrier']
First_flights['Flight_num1'] = First_flights['Flight_num']
First_flights = First_flights[['Date', 'Carrier1', 'Flight_num1', 'Origin']]

First_flights
```

```
<ipython-input-92-28f418c5ea33>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  First_flights['Carrier1'] = First_flights['Carrier']
<ipython-input-92-28f418c5ea33>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  First_flights['Flight_num1'] = First_flights['Flight_num']
```

Out[92]:

	Date	Carrier1	Flight_num1	Origin
0	2024-04-19	UA	538	ORD
2	2024-04-19	B6	116	JFK
4	2024-04-19	WN	491	MCO
6	2024-04-20	UA	538	ORD
8	2024-04-20	B6	116	JFK
11	2024-04-21	UA	538	ORD
13	2024-04-21	B6	116	JFK
15	2024-04-21	WN	491	MCO
17	2024-04-22	UA	538	ORD
19	2024-04-22	B6	116	JFK
21	2024-04-22	WN	491	MCO

In [93]:
```
Second_flights = final_flight[final_flight['Flight_num'].isin({3402, 5340, 656})]

Second_flights
```

Out[93]:

	Date	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m
1	2024-04-19	MQ	ORD	SYR	4	19	3402	22	52	19	59
3	2024-04-19	9E	JFK	SYR	4	19	5340	16	21	14	55
5	2024-04-19	B6	MCO	SYR	4	19	656	16	25	13	35
7	2024-04-20	MQ	ORD	SYR	4	20	3402	22	52	19	59
9	2024-04-20	9E	JFK	SYR	4	20	5340	16	21	14	55
10	2024-04-20	B6	MCO	SYR	4	20	656	16	25	13	35
12	2024-04-21	MQ	ORD	SYR	4	21	3402	22	52	19	59
14	2024-04-21	9E	JFK	SYR	4	21	5340	16	21	14	55
16	2024-04-21	B6	MCO	SYR	4	21	656	16	25	13	35
18	2024-04-22	MQ	ORD	SYR	4	22	3402	22	52	19	59
20	2024-04-22	9E	JFK	SYR	4	22	5340	16	21	14	55
22	2024-04-22	B6	MCO	SYR	4	22	656	16	25	13	34

In [94]:
```
Latter_flight = pd.merge(Second_flights, First_flights, on = ['Origin', 'Date'])
Latter_flight['Arr_Status1'] = 0

Latter_flight
```

Out[94]:

	Date	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Carrier1	Flight_num1
0	2024-04-19	MQ	ORD	SYR	4	19	3402	22	52	19	59	UA	538
1	2024-04-19	9E	JFK	SYR	4	19	5340	16	21	14	55	B6	116
2	2024-04-19	B6	MCO	SYR	4	19	656	16	25	13	35	WN	491
3	2024-04-20	MQ	ORD	SYR	4	20	3402	22	52	19	59	UA	538
4	2024-04-20	9E	JFK	SYR	4	20	5340	16	21	14	55	B6	116
5	2024-04-21	MQ	ORD	SYR	4	21	3402	22	52	19	59	UA	538
6	2024-04-21	9E	JFK	SYR	4	21	5340	16	21	14	55	B6	116
7	2024-04-21	B6	MCO	SYR	4	21	656	16	25	13	35	WN	491
8	2024-04-22	MQ	ORD	SYR	4	22	3402	22	52	19	59	UA	538
9	2024-04-22	9E	JFK	SYR	4	22	5340	16	21	14	55	B6	116
10	2024-04-22	B6	MCO	SYR	4	22	656	16	25	13	34	WN	491

In [95]:
```
Latter_Flight_weather = pd.merge(Latter_flight, test_weather, left_on='Date', right_on='datetime')
Latter_Flight_weather.drop(columns = ['datetime', 'Date'], inplace=True)

Latter_Flight_weather
```

Out[95]:

	Carrier	Origin	Dest	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	...	wind_spd	wind_gust_spc
0	MQ	ORD	SYR	4	19	3402	22	52	19	59	...	4.20	4.6(
1	9E	JFK	SYR	4	19	5340	16	21	14	55	...	4.20	4.6(
2	B6	MCO	SYR	4	19	656	16	25	13	35	...	4.20	4.6(
3	MQ	ORD	SYR	4	20	3402	22	52	19	59	...	3.40	6.5(
4	9E	JFK	SYR	4	20	5340	16	21	14	55	...	3.40	6.5(
5	MQ	ORD	SYR	4	21	3402	22	52	19	59	...	1.90	2.7(
6	9E	JFK	SYR	4	21	5340	16	21	14	55	...	1.90	2.7(
7	B6	MCO	SYR	4	21	656	16	25	13	35	...	1.90	2.7(
8	MQ	ORD	SYR	4	22	3402	22	52	19	59	...	5.50	10.9(
9	9E	JFK	SYR	4	22	5340	16	21	14	55	...	5.50	10.9(
10	B6	MCO	SYR	4	22	656	16	25	13	34	...	5.50	10.9(

11 rows × 28 columns

In [96]:
```
Latter_Flight_weather = pd.get_dummies(Latter_Flight_weather,
                                       columns = ['Carrier', 'Origin','Carrier1', 'Dest'],
                                       drop_first = True)

Latter_Flight_weather
```

Out[96]:

	Date_MM	Date_DD	Flight_num	Arr_t_h	Arr_t_m	Depart_t_h	Depart_t_m	Flight_num1	Arr_Status1	clouds	...	uv	dewpt
0	4	19	3402	22	52	19	59	538	0	84	...	8.30	3.90
1	4	19	5340	16	21	14	55	116	0	84	...	8.30	3.90
2	4	19	656	16	25	13	35	491	0	84	...	8.30	3.90
3	4	20	3402	22	52	19	59	538	0	49	...	8.40	6.30
4	4	20	5340	16	21	14	55	116	0	49	...	8.40	6.30
5	4	21	3402	22	52	19	59	538	0	90	...	8.40	9.60
6	4	21	5340	16	21	14	55	116	0	90	...	8.40	9.60
7	4	21	656	16	25	13	35	491	0	90	...	8.40	9.60
8	4	22	3402	22	52	19	59	538	0	72	...	8.50	8.50
9	4	22	5340	16	21	14	55	116	0	72	...	8.50	8.50
10	4	22	656	16	25	13	34	491	0	72	...	8.50	8.50

11 rows × 30 columns

Predicting final test data of latter flights using pre-trained model - XGBoost classifier

In [97]:
```
Latter_Flight_weather['Arr_Status1'] = 0
y2_pred0 = xgb_classifier.predict(Latter_Flight_weather)

Latter_Flight_weather['Arr_Status1'] = 1
y2_pred1 = xgb_classifier.predict(Latter_Flight_weather)

Latter_Flight_weather['Arr_Status1'] = 2
y2_pred2 = xgb_classifier.predict(Latter_Flight_weather)

y2_pred = np.column_stack((y1_pred0, y1_pred1, y1_pred2))
y2_pred
```

Out[97]:
```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 2],
       [2, 2, 2],
       [2, 2, 2],
       [2, 2, 2],
       [1, 1, 1],
       [1, 1, 1],
       [2, 2, 2],
       [0, 0, 0],
       [0, 0, 0]])
```

In [98]:
```
predictions = np.vectorize(label_mapping.get)(y2_pred)
predictions
```

Out[98]:
```
array([['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'Late'],
       ['Late', 'Late', 'Late'],
       ['Late', 'Late', 'Late'],
       ['Late', 'Late', 'Late'],
       ['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'On Time'],
       ['Late', 'Late', 'Late'],
       ['Early', 'Early', 'Early'],
       ['Early', 'Early', 'Early']], dtype='<U7')
```