In [1]:

```
# Generic inputs for most ML tasks
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# This is new
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor

pd.options.display.float_format = '{:,.2f}'.format

# setup interactive notebook mode
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from IPython.display import display, HTML
```

Fetching Flight data

In [2]:

```
# fetch data

mco_syr_sw_data = pd.read_csv('flight_data/mco_syr_sw_combined.csv')
mco_syr_jb_data = pd.read_csv('flight_data/mco_syr_jb_combined.csv')
jfk_syr_jb_data = pd.read_csv('flight_data/jfk_syr_jb_combined.csv')
jfk_syr_dl_data = pd.read_csv('flight_data/jfk_syr_dl_combined.csv')
ord_syr_ua_data = pd.read_csv('flight_data/ord_syr_ua_combined.csv')
ord_syr_aa_data = pd.read_csv('flight_data/ord_syr_aa_combined.csv')

ord_syr_ua_data['dep_order'] = 'early'
jfk_syr_jb_data['dep_order'] = 'early'
mco_syr_sw_data['dep_order'] = 'early'
ord_syr_aa_data['dep_order'] = 'latter'
jfk_syr_dl_data['dep_order'] = 'latter'
mco_syr_jb_data['dep_order'] = 'latter'
```

In [3]:
```
dfs = [ord_syr_aa_data, ord_syr_ua_data, jfk_syr_dl_data, jfk_syr_jb_data, mco_syr_jb_data, mco_syr_sw_dat
main_data = pd.concat(dfs,axis = 0)
main_data.head()
len(main_data)
```

Out[3]:

	Unnamed: 0	Carrier_Code	Date	Flight_Number	Tail_Number	Destination_Airport	Scheduled departure time	Actual departure time	Scheduled elapsed time (Minutes)	Actual elapsed time (Minutes)	...
0	0	MQ	2020-01-04	3,580.00	N240NN	SYR	7:55	8:21	112.00	87.00	...
1	1	MQ	2020-01-11	3,946.00	N247NN	SYR	15:00	15:09	108.00	132.00	...
2	2	MQ	2020-01-18	3,946.00	N265NN	SYR	15:00	16:27	108.00	147.00	...
3	3	MQ	2020-01-25	3,946.00	N281NN	SYR	15:00	14:55	108.00	126.00	...
4	4	MQ	2020-02-01	3,946.00	N283NN	SYR	15:00	14:57	108.00	99.00	...

5 rows × 36 columns

Out[3]: 8661

In [4]:
```
main_data.dtypes
```

Out[4]:
```
Unnamed: 0                                int64
Carrier_Code                             object
Date                                     object
Flight_Number                           float64
Tail_Number                              object
Destination_Airport                      object
Scheduled departure time                 object
Actual departure time                    object
Scheduled elapsed time (Minutes)        float64
Actual elapsed time (Minutes)           float64
Departure delay (Minutes)               float64
Wheels-off time                          object
Taxi-Out time (Minutes)                 float64
dep_Delay_Carrier                       float64
dep_Delay_Weather                       float64
dep_Delay_National_Aviation_System      float64
dep_Delay_Security                      float64
dep_Delay_Late_Aircraft_Arrival         float64
dep_hour                                  int64
dep_day                                   int64
dep_year                                  int64
dep_order                                object
Origin_Airport                           object
Scheduled Arrival Time                   object
Actual Arrival Time                      object
Arrival Delay (Minutes)                 float64
Wheels-on Time                           object
Taxi-In time (Minutes)                  float64
arr_Delay_Carrier                       float64
arr_Delay_Weather                       float64
arr_Delay_National_Aviation_System      float64
arr_Delay_Security                      float64
arr_Delay_Late_Aircraft_Arrival         float64
arr_hour                                  int64
arr_day                                   int64
arr_year                                  int64
dtype: object
```

In [5]:

```
main_data['Date'] = pd.to_datetime( main_data['Date'],format ="%Y-%m-%d")
main_data['Date'] = main_data['Date'].dt.strftime('%m/%d/%Y')
main_data['dep_min'] = main_data['Scheduled departure time'].str.split(":").str[1].astype('int64')
# main_data['dep_minutes'] = main_data['dep_min'].apply(round_to_nearest_quarter).astype('object')
main_data['dep_min'] = main_data['dep_min'].astype('object')
main_data['arr_min'] = main_data['Scheduled Arrival Time'].str.split(":").str[1].astype('int64')
# main_data['arr_minutes'] = main_data['arr_min'].apply(round_to_nearest_quarter).astype('object')
main_data['arr_min'] = main_data['arr_min'].astype('object')
main_data['Flight_Number'] = main_data['Flight_Number'].astype('object')
main_data['dep_hours'] = main_data['dep_hour'].astype('object')
main_data['dep_hour'] = main_data['dep_hour'].astype('object')
main_data['dep_day'] = main_data['dep_day'].astype('object')
main_data['arr_hours'] = main_data['arr_hour'].astype('object')
main_data['arr_hour'] = main_data['arr_hour'].astype('object')
main_data['arr_day'] = main_data['arr_day'].astype('object')
conditions = [
    (main_data['Arrival Delay (Minutes)'] > 5),
    (main_data['Arrival Delay (Minutes)'] >= -5) & (main_data['Arrival Delay (Minutes)'] <= 5),
    (main_data['Arrival Delay (Minutes)'] < -5)
]
conditions2 = [
    (main_data['Departure delay (Minutes)'] > 5),
    (main_data['Departure delay (Minutes)'] >=-5) & (main_data['Departure delay (Minutes)'] <= 5),
    (main_data['Departure delay (Minutes)'] < -5)
]
choices = [2, 1, 0]
main_data['dep_status'] = np.select(conditions2, choices)

main_data['arr_status'] = np.select(conditions, choices)
main_data.dtypes
main_data.head()
```

Out[5]:
```
Unnamed: 0                            int64
Carrier_Code                         object
Date                                 object
Flight_Number                        object
Tail_Number                          object
Destination_Airport                  object
Scheduled departure time             object
Actual departure time                object
Scheduled elapsed time (Minutes)    float64
Actual elapsed time (Minutes)       float64
Departure delay (Minutes)           float64
Wheels-off time                      object
Taxi-Out time (Minutes)             float64
dep_Delay_Carrier                   float64
dep_Delay_Weather                   float64
dep_Delay_National_Aviation_System  float64
dep_Delay_Security                  float64
dep_Delay_Late_Aircraft_Arrival     float64
dep_hour                             object
dep_day                              object
dep_year                              int64
dep_order                            object
Origin_Airport                       object
Scheduled Arrival Time               object
Actual Arrival Time                  object
Arrival Delay (Minutes)             float64
Wheels-on Time                       object
Taxi-In time (Minutes)              float64
arr_Delay_Carrier                   float64
arr_Delay_Weather                   float64
arr_Delay_National_Aviation_System  float64
arr_Delay_Security                  float64
arr_Delay_Late_Aircraft_Arrival     float64
arr_hour                             object
arr_day                              object
arr_year                              int64
dep_min                              object
arr_min                              object
dep_hours                            object
arr_hours                            object
dep_status                            int32
```

```
arr_status                                int32
dtype: object
```

Out[5]:

	Unnamed: 0	Carrier_Code	Date	Flight_Number	Tail_Number	Destination_Airport	Scheduled departure time	Actual departure time	Scheduled elapsed time (Minutes)	Actu elapse tin (Minute
0	0	MQ	01/04/2020	3,580.00	N240NN	SYR	7:55	8:21	112.00	87.(
1	1	MQ	01/11/2020	3,946.00	N247NN	SYR	15:00	15:09	108.00	132.(
2	2	MQ	01/18/2020	3,946.00	N265NN	SYR	15:00	16:27	108.00	147.(
3	3	MQ	01/25/2020	3,946.00	N281NN	SYR	15:00	14:55	108.00	126.(
4	4	MQ	02/01/2020	3,946.00	N283NN	SYR	15:00	14:57	108.00	99.(

5 rows × 42 columns

In [6]:

```
sub_data = main_data.drop(columns = ['Unnamed: 0','Destination_Airport','Actual departure time','Scheduled
                         'Departure delay (Minutes)','Wheels-off time','Taxi-Out time (Minutes)','dep_Dela
                         'dep_Delay_Weather','dep_Delay_National_Aviation_System','dep_Delay_Security','de
                         'arr_Delay_National_Aviation_System','arr_Delay_Security','arr_Delay_Late_Aircraf
```

In [7]:
```
sub_data.dtypes
```

Out[7]:
```
Carrier_Code                 object
Date                         object
Flight_Number                object
Tail_Number                  object
Scheduled departure time     object
dep_hour                     object
dep_day                      object
dep_order                    object
Origin_Airport               object
Scheduled Arrival Time       object
Arrival Delay (Minutes)     float64
arr_hour                     object
arr_day                      object
dep_min                      object
arr_min                      object
dep_hours                    object
arr_hours                    object
dep_status                    int32
arr_status                    int32
dtype: object
```

In [8]:
```
sub_data.head()
```

Out[8]:

	Carrier_Code	Date	Flight_Number	Tail_Number	Scheduled departure time	dep_hour	dep_day	dep_order	Origin_Airport	Scheduled Arrival Time	(M
0	MQ	01/04/2020	3,580.00	N240NN	7:55	7	5	latter	ORD	10:47	
1	MQ	01/11/2020	3,946.00	N247NN	15:00	15	5	latter	ORD	17:48	
2	MQ	01/18/2020	3,946.00	N265NN	15:00	15	5	latter	ORD	17:48	
3	MQ	01/25/2020	3,946.00	N281NN	15:00	15	5	latter	ORD	17:48	
4	MQ	02/01/2020	3,946.00	N283NN	15:00	15	5	latter	ORD	17:48	

Modifying flight data to have latter flights along with early flight arrival status

In [9]:
```
latter_data = sub_data[sub_data['dep_order'] == 'latter']
len(latter_data)
```

Out[9]: 5337

In [10]:
```
early_data = sub_data[sub_data['dep_order'] == 'early']
len(early_data)
```

Out[10]: 3324

In [11]:
```
merged_df = pd.merge(latter_data, early_data, on=['Date', 'Origin_Airport'], how = 'left')
merged_df.head()
len(merged_df)
```

Out[11]:

	Carrier_Code_x	Date	Flight_Number_x	Tail_Number_x	Scheduled departure time_x	dep_hour_x	dep_day_x	dep_order_x	Origin_Airport
0	MQ	01/04/2020	3,580.00	N240NN	7:55	7	5	latter	ORD
1	MQ	01/11/2020	3,946.00	N247NN	15:00	15	5	latter	ORD
2	MQ	01/18/2020	3,946.00	N265NN	15:00	15	5	latter	ORD
3	MQ	01/25/2020	3,946.00	N281NN	15:00	15	5	latter	ORD
4	MQ	02/01/2020	3,946.00	N283NN	15:00	15	5	latter	ORD

5 rows × 36 columns

Out[11]: 7251

In [12]:
```
merged_df.head(20)
```

Out[12]:

	Carrier_Code_x	Date	Flight_Number_x	Tail_Number_x	Scheduled departure time_x	dep_hour_x	dep_day_x	dep_order_x	Origin_Airport
0	MQ	01/04/2020	3,580.00	N240NN	7:55	7	5	latter	ORD
1	MQ	01/11/2020	3,946.00	N247NN	15:00	15	5	latter	ORD
2	MQ	01/18/2020	3,946.00	N265NN	15:00	15	5	latter	ORD
3	MQ	01/25/2020	3,946.00	N281NN	15:00	15	5	latter	ORD
4	MQ	02/01/2020	3,946.00	N283NN	15:00	15	5	latter	ORD
5	MQ	02/08/2020	3,946.00	N274NN	15:00	15	5	latter	ORD
6	MQ	04/06/2020	3,618.00	NaN	19:29	19	0	latter	ORD
7	MQ	04/07/2020	3,946.00	N287NN	15:06	15	1	latter	ORD
8	MQ	04/08/2020	3,946.00	NaN	15:06	15	2	latter	ORD
9	MQ	04/09/2020	3,946.00	NaN	15:06	15	3	latter	ORD
10	MQ	04/10/2020	3,946.00	NaN	15:06	15	4	latter	ORD
11	MQ	04/12/2020	3,946.00	NaN	15:06	15	6	latter	ORD
12	MQ	04/13/2020	3,946.00	N243NN	15:06	15	0	latter	ORD
13	MQ	04/14/2020	3,946.00	N234JW	15:06	15	1	latter	ORD
14	MQ	04/15/2020	3,946.00	NaN	15:06	15	2	latter	ORD
15	MQ	04/16/2020	3,946.00	N278NN	15:06	15	3	latter	ORD
16	MQ	04/17/2020	3,946.00	NaN	15:06	15	4	latter	ORD
17	MQ	04/19/2020	3,946.00	NaN	15:06	15	6	latter	ORD
18	MQ	04/20/2020	3,946.00	NaN	15:06	15	0	latter	ORD
19	MQ	04/21/2020	3,946.00	NaN	15:06	15	1	latter	ORD

20 rows × 36 columns

In [13]:
```
len(merged_df)
```

Out[13]:
```
7251
```

In [14]:
```
merged_df.columns
```

Out[14]:
```
Index(['Carrier_Code_x', 'Date', 'Flight_Number_x', 'Tail_Number_x',
       'Scheduled departure time_x', 'dep_hour_x', 'dep_day_x', 'dep_order_x',
       'Origin_Airport', 'Scheduled Arrival Time_x',
       'Arrival Delay (Minutes)_x', 'arr_hour_x', 'arr_day_x', 'dep_min_x',
       'arr_min_x', 'dep_hours_x', 'arr_hours_x', 'dep_status_x',
       'arr_status_x', 'Carrier_Code_y', 'Flight_Number_y', 'Tail_Number_y',
       'Scheduled departure time_y', 'dep_hour_y', 'dep_day_y', 'dep_order_y',
       'Scheduled Arrival Time_y', 'Arrival Delay (Minutes)_y', 'arr_hour_y',
       'arr_day_y', 'dep_min_y', 'arr_min_y', 'dep_hours_y', 'arr_hours_y',
       'dep_status_y', 'arr_status_y'],
      dtype='object')
```

Filtering flights with a max gap of 3hrs

In [15]:
```
merged_df['hour_diff']= merged_df['dep_hour_x'] - merged_df['dep_hour_y']
merged_df.to_csv('mergrrr.csv')
merged_df1 = merged_df[merged_df['hour_diff'] == 0 ]
merged_df1 = merged_df1[merged_df1['dep_min_x'] > merged_df1['dep_min_y']]
merged_df2 = merged_df[(merged_df['hour_diff'] >=1) & (merged_df['hour_diff'] <= 2)]
merged_df = pd.concat([merged_df1,merged_df2])
len(merged_df)
```

Out[15]:
```
893
```

In [16]:
```
merged_df.head()
```

Out[16]:

	Carrier_Code_x	Date	Flight_Number_x	Tail_Number_x	Scheduled departure time_x	dep_hour_x	dep_day_x	dep_order_x	Origin_Airport
869	MQ	07/04/2023	3,402.00	N298FR	18:30	18	1	latter	ORD
870	MQ	07/05/2023	3,402.00	N768RD	18:36	18	2	latter	ORD
873	MQ	07/06/2023	3,402.00	N634RW	18:36	18	3	latter	ORD
884	MQ	07/10/2023	3,402.00	N228NN	18:36	18	0	latter	ORD
887	MQ	07/11/2023	3,402.00	N449YX	18:36	18	1	latter	ORD

5 rows × 37 columns

correcting carrier codes

In [17]:
```
merged_df["Carrier_Code_x"] = np.where(merged_df["Carrier_Code_x"] == "MQ", "AA", merged_df["Carrier_Code_
merged_df["Carrier_Code_x"] = np.where(merged_df["Carrier_Code_x"] == "9E", "DL", merged_df["Carrier_Code_
```

In [18]:
```
merged_df.head()
```

Out[18]:

	Carrier_Code_x	Date	Flight_Number_x	Tail_Number_x	Scheduled departure time_x	dep_hour_x	dep_day_x	dep_order_x	Origin_Airport
869	AA	07/04/2023	3,402.00	N298FR	18:30	18	1	latter	ORD
870	AA	07/05/2023	3,402.00	N768RD	18:36	18	2	latter	ORD
873	AA	07/06/2023	3,402.00	N634RW	18:36	18	3	latter	ORD
884	AA	07/10/2023	3,402.00	N228NN	18:36	18	0	latter	ORD
887	AA	07/11/2023	3,402.00	N449YX	18:36	18	1	latter	ORD

5 rows × 37 columns

In [19]:

```
merged_df.columns
```

Out[19]:

```
Index(['Carrier_Code_x', 'Date', 'Flight_Number_x', 'Tail_Number_x',
       'Scheduled departure time_x', 'dep_hour_x', 'dep_day_x', 'dep_order_x',
       'Origin_Airport', 'Scheduled Arrival Time_x',
       'Arrival Delay (Minutes)_x', 'arr_hour_x', 'arr_day_x', 'dep_min_x',
       'arr_min_x', 'dep_hours_x', 'arr_hours_x', 'dep_status_x',
       'arr_status_x', 'Carrier_Code_y', 'Flight_Number_y', 'Tail_Number_y',
       'Scheduled departure time_y', 'dep_hour_y', 'dep_day_y', 'dep_order_y',
       'Scheduled Arrival Time_y', 'Arrival Delay (Minutes)_y', 'arr_hour_y',
       'arr_day_y', 'dep_min_y', 'arr_min_y', 'dep_hours_y', 'arr_hours_y',
       'dep_status_y', 'arr_status_y', 'hour_diff'],
      dtype='object')
```

In [20]:

```
merged_df.drop(columns=['Carrier_Code_y', 'Flight_Number_y',
       'Tail_Number_y', 'Scheduled departure time_y', 'dep_hour_y',
       'dep_day_y', 'dep_order_y', 'Scheduled Arrival Time_y',
       'Arrival Delay (Minutes)_y', 'arr_hour_y', 'arr_day_y', 'dep_min_y',
       'arr_min_y', 'dep_hours_y',
       'arr_hours_y','dep_status_y'], inplace=True)
```

In [21]:
```
merged_df.columns = merged_df.columns.str.replace('_x', '')
merged_df.rename(columns={'arr_status':'arr_status_x'}, inplace=True)

merged_df.head()
merged_df.columns
```

Out[21]:

	Carrier_Code	Date	Flight_Number	Tail_Number	Scheduled departure time	dep_hour	dep_day	dep_order	Origin_Airport	Scheduled Arrival Time
869	AA	07/04/2023	3,402.00	N298FR	18:30	18	1	latter	ORD	21:25
870	AA	07/05/2023	3,402.00	N768RD	18:36	18	2	latter	ORD	21:32
873	AA	07/06/2023	3,402.00	N634RW	18:36	18	3	latter	ORD	21:32
884	AA	07/10/2023	3,402.00	N228NN	18:36	18	0	latter	ORD	21:32
887	AA	07/11/2023	3,402.00	N449YX	18:36	18	1	latter	ORD	21:32

5 rows × 21 columns

Out[21]:
```
Index(['Carrier_Code', 'Date', 'Flight_Number', 'Tail_Number',
       'Scheduled departure time', 'dep_hour', 'dep_day', 'dep_order',
       'Origin_Airport', 'Scheduled Arrival Time', 'Arrival Delay (Minutes)',
       'arr_hour', 'arr_day', 'dep_min', 'arr_min', 'dep_hours', 'arr_hours',
       'dep_status', 'arr_status_x', 'arr_status_y', 'hour_diff'],
      dtype='object')
```

In [22]:
```
# merged_df.to_csv('flight_data\\all_lat_ear.csv')
```

Fetching weather data

In [23]:

```
# Read and process weather data files for each airport
jfk_weather_data = pd.read_csv('weather_data/JFK_weather_data_hourly_processed.csv')
syr_weather_data = pd.read_csv('weather_data/SYR_weather_data_hourly_processed.csv')
ord_weather_data = pd.read_csv('weather_data/ORD_weather_data_hourly_processed.csv')
mco_weather_data = pd.read_csv('weather_data/MCO_weather_data_hourly_processed.csv')

# Combine weather data for all airports
weather_dfs = [jfk_weather_data, ord_weather_data, mco_weather_data]
weather_data = pd.concat(weather_dfs, axis=0)
weather_data['dep_hours'] = weather_data['dep_hours'].astype('object')
syr_weather_data['arr_hours'] = syr_weather_data['arr_hours'].astype('object')
weather_data.head()
syr_weather_data.head()
weather_data.dtypes
syr_weather_data.dtypes
```

Out[23]:

	dep_azimuth	dep_clouds	dep_dewpt	dep_elev_angle	dep_h_angle	dep_precip	dep_pres	dep_revision_status	dep_rh	dep_sn
0	261.20	100	3.80	-26.20	NaN	0.00	1002	final	88	0
1	270.50	100	3.90	-37.50	NaN	0.25	1003	final	85	0
2	281.40	100	3.70	-48.80	NaN	0.00	1003	final	82	0
3	296.30	100	1.60	-59.60	NaN	0.00	1002	final	73	0
4	320.80	100	0.70	-68.60	NaN	0.00	1003	final	69	0

Out[23]:

	arr_azimuth	arr_clouds	arr_dewpt	arr_elev_angle	arr_h_angle	arr_precip	arr_pres	arr_revision_status	arr_rh	arr_snow	arr_t
0	260.90	100	-2.30	-24.90	NaN	0.00	987	final	78	0.00	
1	270.70	100	-3.00	-35.80	NaN	0.00	987	final	77	0.00	
2	282.10	100	-4.00	-46.60	NaN	0.00	986	final	71	0.00	
3	297.00	100	-4.40	-56.90	NaN	0.00	987	final	69	0.00	
4	319.80	100	-4.40	-65.60	NaN	0.00	986	final	69	0.00	

Out[23]:

```
dep_azimuth                float64
dep_clouds                   int64
dep_dewpt                  float64
dep_elev_angle             float64
dep_h_angle                float64
dep_precip                 float64
dep_pres                     int64
dep_revision_status         object
dep_rh                       int64
dep_snow                   float64
dep_temp                   float64
dep_vis                      int64
dep_weather.description     object
dep_weather.code             int64
dep_wind_dir                 int64
dep_wind_gust_spd          float64
dep_wind_spd               float64
Date                        object
dep_hours                   object
Origin_Airport              object
dtype: object
```

Out[23]:
```
arr_azimuth                 float64
arr_clouds                    int64
arr_dewpt                   float64
arr_elev_angle              float64
arr_h_angle                 float64
arr_precip                  float64
arr_pres                      int64
arr_revision_status          object
arr_rh                        int64
arr_snow                    float64
arr_temp                    float64
arr_vis                       int64
arr_weather.description      object
arr_weather.code              int64
arr_wind_dir                  int64
arr_wind_gust_spd           float64
arr_wind_spd                float64
Date                         object
arr_hours                    object
dtype: object
```

In [24]:
```
# Define merging logic based on airport code
merged_df = pd.merge(merged_df, syr_weather_data, how='left', on=['Date', 'arr_hours'])
merged_df.head()
```

Out[24]:

	Carrier_Code	Date	Flight_Number	Tail_Number	Scheduled departure time	dep_hour	dep_day	dep_order	Origin_Airport	Scheduled Arrival Time	...
0	AA	07/04/2023	3,402.00	N298FR	18:30	18	1	latter	ORD	21:25	...
1	AA	07/05/2023	3,402.00	N768RD	18:36	18	2	latter	ORD	21:32	...
2	AA	07/06/2023	3,402.00	N634RW	18:36	18	3	latter	ORD	21:32	...
3	AA	07/10/2023	3,402.00	N228NN	18:36	18	0	latter	ORD	21:32	...
4	AA	07/11/2023	3,402.00	N449YX	18:36	18	1	latter	ORD	21:32	...

5 rows × 38 columns

In [25]:
```
merged_df = pd.merge(merged_df, weather_data, how='left', on=['Origin_Airport', 'Date', 'dep_hours'])


merged_df.head()
```

Out[25]:

	Carrier_Code	Date	Flight_Number	Tail_Number	Scheduled departure time	dep_hour	dep_day	dep_order	Origin_Airport	Scheduled Arrival Time	...
0	AA	07/04/2023	3,402.00	N298FR	18:30	18	1	latter	ORD	21:25	...
1	AA	07/05/2023	3,402.00	N768RD	18:36	18	2	latter	ORD	21:32	...
2	AA	07/06/2023	3,402.00	N634RW	18:36	18	3	latter	ORD	21:32	...
3	AA	07/10/2023	3,402.00	N228NN	18:36	18	0	latter	ORD	21:32	...
4	AA	07/11/2023	3,402.00	N449YX	18:36	18	1	latter	ORD	21:32	...

5 rows × 55 columns

In [26]:
```
# merged_df.to_csv('flight_data\\all_lat_ear.csv')
```

In [27]:
```
merged_df.drop(columns= ['Date', 'Flight_Number', 'Tail_Number',
       'Scheduled departure time','Carrier_Code',
       'dep_order', 'Scheduled Arrival Time',
       'Arrival Delay (Minutes)','hour_diff','arr_weather.description','dep_weather.description'],inplace

merged_df.dtypes
```

Out[27]:
```
dep_hour                  object
dep_day                   object
Origin_Airport            object
arr_hour                  object
arr_day                   object
dep_min                   object
arr_min                   object
dep_hours                 object
arr_hours                 object
dep_status                 int32
arr_status_x               int32
arr_status_y             float64
arr_azimuth              float64
arr_clouds                 int64
arr_dewpt                float64
arr_elev_angle           float64
arr_h_angle              float64
arr_precip               float64
arr_pres                   int64
arr_revision_status       object
arr_rh                     int64
arr_snow                 float64
arr_temp                 float64
arr_vis                    int64
arr_weather.code           int64
arr_wind_dir               int64
arr_wind_gust_spd        float64
arr_wind_spd             float64
dep_azimuth              float64
dep_clouds                 int64
dep_dewpt                float64
dep_elev_angle           float64
dep_h_angle              float64
dep_precip               float64
dep_pres                   int64
dep_revision_status       object
dep_rh                     int64
dep_snow                 float64
dep_temp                 float64
dep_vis                    int64
dep_weather.code           int64
dep_wind_dir               int64
dep_wind_gust_spd        float64
```

```
dep_wind_spd           float64
dtype: object
```

In [28]:
```
merged_df['dep_status'] = merged_df['dep_status'].astype('int64').astype('object')
merged_df['arr_status_y'] = merged_df['arr_status_y'].astype('int64').astype('object')
merged_df.head()
merged_df.dtypes
su_data = merged_df
```

Out[28]:

	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	dep_hours	arr_hours	dep_status	...	dep_pres	dep_
0	18	1	ORD	21	1	30	25	18	21	0	...	989	
1	18	2	ORD	21	2	36	32	18	21	1	...	986	
2	18	3	ORD	21	3	36	32	18	21	1	...	989	
3	18	0	ORD	21	0	36	32	18	21	2	...	987	
4	18	1	ORD	21	1	36	32	18	21	2	...	989	

5 rows × 44 columns

Out[28]:
```
dep_hour                  object
dep_day                   object
Origin_Airport            object
arr_hour                  object
arr_day                   object
dep_min                   object
arr_min                   object
dep_hours                 object
arr_hours                 object
dep_status                object
arr_status_x               int32
arr_status_y              object
arr_azimuth              float64
arr_clouds                 int64
arr_dewpt                float64
arr_elev_angle           float64
arr_h_angle              float64
arr_precip               float64
arr_pres                   int64
arr_revision_status       object
arr_rh                     int64
arr_snow                 float64
arr_temp                 float64
arr_vis                    int64
arr_weather.code           int64
arr_wind_dir               int64
arr_wind_gust_spd        float64
arr_wind_spd             float64
dep_azimuth              float64
dep_clouds                 int64
dep_dewpt                float64
dep_elev_angle           float64
dep_h_angle              float64
dep_precip               float64
dep_pres                   int64
dep_revision_status       object
dep_rh                     int64
dep_snow                 float64
dep_temp                 float64
dep_vis                    int64
dep_weather.code           int64
dep_wind_dir               int64
dep_wind_gust_spd        float64
```

```
dep_wind_spd          float64
dtype: object
```

In [29]:

```
su_data.head()
```

Out[29]:

	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	dep_hours	arr_hours	dep_status	...	dep_pres	dep_
0	18	1	ORD	21	1	30	25	18	21	0	...	989	
1	18	2	ORD	21	2	36	32	18	21	1	...	986	
2	18	3	ORD	21	3	36	32	18	21	1	...	989	
3	18	0	ORD	21	0	36	32	18	21	2	...	987	
4	18	1	ORD	21	1	36	32	18	21	2	...	989	

5 rows × 44 columns

In [30]:
```
su_data.columns
su_data.head()
```

Out[30]:
```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_hours', 'arr_hours', 'dep_status',
       'arr_status_x', 'arr_status_y', 'arr_azimuth', 'arr_clouds',
       'arr_dewpt', 'arr_elev_angle', 'arr_h_angle', 'arr_precip', 'arr_pres',
       'arr_revision_status', 'arr_rh', 'arr_snow', 'arr_temp', 'arr_vis',
       'arr_weather.code', 'arr_wind_dir', 'arr_wind_gust_spd', 'arr_wind_spd',
       'dep_azimuth', 'dep_clouds', 'dep_dewpt', 'dep_elev_angle',
       'dep_h_angle', 'dep_precip', 'dep_pres', 'dep_revision_status',
       'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis', 'dep_weather.code',
       'dep_wind_dir', 'dep_wind_gust_spd', 'dep_wind_spd'],
      dtype='object')
```

Out[30]:

	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	dep_hours	arr_hours	dep_status	...	dep_pres	dep_
0	18	1	ORD	21	1	30	25	18	21	0	...	989	
1	18	2	ORD	21	2	36	32	18	21	1	...	986	
2	18	3	ORD	21	3	36	32	18	21	1	...	989	
3	18	0	ORD	21	0	36	32	18	21	2	...	987	
4	18	1	ORD	21	1	36	32	18	21	2	...	989	

5 rows × 44 columns

In [31]:

```
su_data['dep_hour'] = pd.Categorical(su_data['dep_hour'], categories=[i for i in range(24)])
su_data['dep_day'] = pd.Categorical(su_data['dep_day'], categories=[i for i in range(7)])
su_data['dep_min'] = pd.Categorical(su_data['dep_min'], categories=[i for i in range(60)])
su_data['arr_hour'] = pd.Categorical(su_data['arr_hour'], categories=[i for i in range(24)])
su_data['arr_day'] = pd.Categorical(su_data['arr_day'], categories=[i for i in range(7)])
su_data['arr_min'] = pd.Categorical(su_data['arr_min'], categories=[i for i in range(60)])
#su_data['Carrier_Code'] = pd.Categorical(su_data['Carrier_Code'], categories=['AA', 'DL', 'B6'])
su_data['Origin_Airport'] = pd.Categorical(su_data['Origin_Airport'], categories=['ORD', 'JFK', 'MCO'])
su_data['arr_weather.code'] = pd.Categorical(su_data['arr_weather.code'], categories=[200,201,202,230,231,
su_data['dep_weather.code'] = pd.Categorical(su_data['dep_weather.code'], categories=[200,201,202,230,231,
su_data['dep_status']= pd.Categorical(su_data['dep_status'], categories = [0,1,2])
su_data.drop(columns=['dep_hours', 'arr_hours'],inplace = True)
su_data.columns
```

Out[31]:

```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_status', 'arr_status_x', 'arr_status_y',
       'arr_azimuth', 'arr_clouds', 'arr_dewpt', 'arr_elev_angle',
       'arr_h_angle', 'arr_precip', 'arr_pres', 'arr_revision_status',
       'arr_rh', 'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.code',
       'arr_wind_dir', 'arr_wind_gust_spd', 'arr_wind_spd', 'dep_azimuth',
       'dep_clouds', 'dep_dewpt', 'dep_elev_angle', 'dep_h_angle',
       'dep_precip', 'dep_pres', 'dep_revision_status', 'dep_rh', 'dep_snow',
       'dep_temp', 'dep_vis', 'dep_weather.code', 'dep_wind_dir',
       'dep_wind_gust_spd', 'dep_wind_spd'],
      dtype='object')
```

In [32]:
```
su_data.isna().sum()
```

Out[32]:
```
dep_hour                0
dep_day                 0
Origin_Airport          0
arr_hour                0
arr_day                 0
dep_min                 0
arr_min                 0
dep_status              0
arr_status_x            0
arr_status_y            0
arr_azimuth             0
arr_clouds              0
arr_dewpt               0
arr_elev_angle          0
arr_h_angle           893
arr_precip              0
arr_pres                0
arr_revision_status     0
arr_rh                  0
arr_snow                0
arr_temp                0
arr_vis                 0
arr_weather.code        8
arr_wind_dir            0
arr_wind_gust_spd       0
arr_wind_spd            0
dep_azimuth             0
dep_clouds              0
dep_dewpt               0
dep_elev_angle          0
dep_h_angle           893
dep_precip              0
dep_pres                0
dep_revision_status     0
dep_rh                  0
dep_snow                0
dep_temp                0
dep_vis                 0
dep_weather.code        1
dep_wind_dir            0
dep_wind_gust_spd       0
```

```
dep_wind_spd             0
dtype: int64
```

In [33]:

```
su_data.drop(columns=['arr_h_angle','dep_h_angle', 'arr_azimuth', 'dep_azimuth', 'arr_elev_angle',
                      'dep_elev_angle', 'arr_revision_status', 'dep_revision_status'],inplace = True)
su_data.dropna(inplace=True)
```

In [34]:

```
su_data.columns
```

Out[34]:

```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_status', 'arr_status_x', 'arr_status_y',
       'arr_clouds', 'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh',
       'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.code', 'arr_wind_dir',
       'arr_wind_gust_spd', 'arr_wind_spd', 'dep_clouds', 'dep_dewpt',
       'dep_precip', 'dep_pres', 'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis',
       'dep_weather.code', 'dep_wind_dir', 'dep_wind_gust_spd',
       'dep_wind_spd'],
      dtype='object')
```

Training model to predict latter flight arrival status

In [35]:

```
su_data = pd.get_dummies(su_data, drop_first = True)
su_data.head()
su_data.dtypes
```

Out[35]:

	arr_status_x	arr_clouds	arr_dewpt	arr_precip	arr_pres	arr_rh	arr_snow	arr_temp	arr_vis	arr_wind_dir	...	dep_weather.code
0	1	50	18.10	0.00	999	68	0.00	24.40	13	325	...	
1	1	25	17.80	0.00	999	62	0.00	25.60	16	70	...	
2	0	87	18.80	0.00	995	62	0.00	26.70	16	300	...	
3	2	87	15.40	0.00	995	86	0.00	17.80	16	190	...	
4	2	87	17.00	0.00	996	61	0.00	25.00	16	320	...	

5 rows × 275 columns

Out[35]:

```
arr_status_x               int32
arr_clouds                 int64
arr_dewpt                float64
arr_precip               float64
arr_pres                   int64
                          ...
dep_weather.code_801        bool
dep_weather.code_802        bool
dep_weather.code_803        bool
dep_weather.code_804        bool
dep_weather.code_900        bool
Length: 275, dtype: object
```

In [36]:

```
X_train, X_test, y_train, y_test = train_test_split(su_data.drop(columns = ['arr_status_x']), su_data['arr
X_train
X_test
y_train.dtypes
y_test
```

Out[36]:

	arr_clouds	arr_dewpt	arr_precip	arr_pres	arr_rh	arr_snow	arr_temp	arr_vis	arr_wind_dir	arr_wind_gust_spd	...	dep_wea
333	50	-5.70	0.00	995	71	0.00	-1.10	16	90	2.80	...	
178	87	5.40	0.00	1009	49	0.00	16.10	16	300	7.20	...	
281	50	17.20	0.00	995	87	0.00	19.40	16	220	3.20	...	
297	87	13.80	0.00	998	83	0.00	16.70	16	190	6.00	...	
278	25	18.30	0.00	1006	81	0.00	21.70	16	35	2.80	...	
...	...	...	...	...	...	...	...	...	...	...	...	
506	50	-7.40	0.00	1007	49	0.00	2.20	16	90	4.40	...	
370	100	19.40	0.00	988	84	0.00	22.20	16	285	8.40	...	
440	100	11.70	0.00	985	100	0.00	11.70	10	250	7.60	...	
694	100	11.10	0.25	988	83	0.00	13.90	11	140	10.00	...	
86	100	5.50	0.50	998	79	0.00	8.90	16	280	9.30	...	

708 rows × 274 columns

Out[36]:

	arr_clouds	arr_dewpt	arr_precip	arr_pres	arr_rh	arr_snow	arr_temp	arr_vis	arr_wind_dir	arr_wind_gust_spd	...	dep_wea
711	100	8.10	0.00	1006	66	0.00	14.40	16	50	4.00	...	
668	25	8.90	0.00	1007	80	0.00	12.20	16	135	2.40	...	
821	100	1.00	0.00	999	85	0.00	3.30	10	300	9.20	...	
422	100	5.50	0.00	1004	92	0.00	6.70	8	80	7.60	...	
639	100	18.30	0.00	996	81	0.00	21.70	16	240	4.40	...	
...	...	...	...	...	...	...	...	...	...	...	...	
837	87	-12.30	0.00	1008	39	0.00	0.00	16	170	9.80	...	
644	87	11.10	0.00	998	72	0.00	16.10	16	340	8.00	...	
208	50	-5.00	0.00	1002	72	0.00	-0.60	16	210	9.30	...	
465	100	-4.40	0.00	1010	78	0.00	-1.10	16	160	2.80	...	
835	50	-13.50	0.00	1001	40	0.00	-1.70	16	250	18.50	...	

177 rows × 274 columns

Out[36]:

```
dtype('int32')
```

Out[36]:

```
711    1
668    2
821    1
422    1
639    0
      ..
837    1
644    0
208    2
465    0
835    2
Name: arr_status_x, Length: 177, dtype: int32
```

In [37]:
```
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
```

Out[37]:

	arr_clouds	arr_dewpt	arr_precip	arr_pres	arr_rh	arr_snow	arr_temp	arr_vis	arr_wind_dir	arr_wind_gust_spd	...	dep_wea
333	-0.77	-1.38	-0.20	-0.67	0.17	-0.08	-1.51	0.40	-1.21	-1.15	...	
178	0.39	-0.05	-0.20	1.10	-1.06	-0.08	0.45	0.40	1.17	0.17	...	
281	-0.77	1.37	-0.20	-0.67	1.06	-0.08	0.83	0.40	0.26	-1.03	...	
297	0.39	0.96	-0.20	-0.29	0.84	-0.08	0.52	0.40	-0.08	-0.19	...	
278	-1.56	1.50	-0.20	0.72	0.73	-0.08	1.09	0.40	-1.83	-1.15	...	
...	...	...	...	...	...	...	...	...	...	...	...	
506	-0.77	-1.58	-0.20	0.84	-1.06	-0.08	-1.14	0.40	-1.21	-0.67	...	
370	0.80	1.63	-0.20	-1.56	0.89	-0.08	1.15	0.40	1.00	0.53	...	
440	0.80	0.71	-0.20	-1.94	1.79	-0.08	-0.05	-1.59	0.60	0.29	...	
694	0.80	0.64	0.06	-1.56	0.84	-0.08	0.20	-1.26	-0.65	1.01	...	
86	0.80	-0.04	0.31	-0.29	0.61	-0.08	-0.37	0.40	0.94	0.80	...	

708 rows × 274 columns

Out[37]:

	arr_clouds	arr_dewpt	arr_precip	arr_pres	arr_rh	arr_snow	arr_temp	arr_vis	arr_wind_dir	arr_wind_gust_spd	...	dep_wea
711	0.80	0.28	-0.20	0.72	-0.11	-0.08	0.25	0.40	-1.66	-0.79	...	
668	-1.56	0.37	-0.20	0.84	0.67	-0.08	0.00	0.40	-0.70	-1.27	...	
821	0.80	-0.58	-0.20	-0.17	0.95	-0.08	-1.01	-1.59	1.17	0.77	...	
422	0.80	-0.04	-0.20	0.46	1.34	-0.08	-0.62	-2.26	-1.32	0.29	...	
639	0.80	1.50	-0.20	-0.55	0.73	-0.08	1.09	0.40	0.49	-0.67	...	
...	...	...	...	...	...	...	...	...	...	...	...	
837	0.39	-2.17	-0.20	0.97	-1.62	-0.08	-1.39	0.40	-0.31	0.95	...	
644	0.39	0.64	-0.20	-0.29	0.22	-0.08	0.45	0.40	1.62	0.41	...	
208	-0.77	-1.30	-0.20	0.21	0.22	-0.08	-1.46	0.40	0.15	0.80	...	
465	0.80	-1.22	-0.20	1.22	0.56	-0.08	-1.51	0.40	-0.42	-1.15	...	
835	-0.77	-2.32	-0.20	0.09	-1.56	-0.08	-1.58	0.40	0.60	3.55	...	

177 rows × 274 columns

Out[37]:

```
333    2
178    2
281    2
297    2
278    2
      ..
506    0
370    2
440    0
694    1
86     0
Name: arr_status_x, Length: 708, dtype: int32
```

Out[37]:
```
711    1
668    2
821    1
422    1
639    0
      ..
837    1
644    0
208    2
465    0
835    2
Name: arr_status_x, Length: 177, dtype: int32
```

In [38]:
```
arr_model = LogisticRegression(fit_intercept = True, solver='lbfgs', multi_class = 'multinomial', penalty


arr_model.fit(X_train, y_train)

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

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

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

```
         0.00000000e+00,  0.00000000e+00, -1.32600080e-01,
        -3.01124557e-01,  5.86985921e-02, -3.45515896e-02,
        -6.44265871e-02, -6.74655713e-02, -4.99330248e-01,
        -1.32062274e-01,  1.26724943e-01,  0.00000000e+00,
        -2.95476329e-01,  7.43587194e-01,  2.90824138e-01,
        -7.04065828e-02,  0.00000000e+00,  0.00000000e+00,
         2.50733914e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -4.22297364e-01,
         1.46573167e-01,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00, -2.93971401e-01, -3.81025792e-01,
         0.00000000e+00,  0.00000000e+00, -4.37724366e-01,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         1.32816035e-01,  1.46138449e-01, -1.63535785e+00,
        -1.33961222e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00, -7.96169001e-01, -6.85862430e-03,
         2.90824138e-01, -1.33638367e+00,  1.16339902e-01,
         3.38779158e-01,  0.00000000e+00, -3.15687786e-02,
         0.00000000e+00, -1.81473254e+00, -3.52862832e-02,
         0.00000000e+00,  3.07389014e-01,  2.19246224e-01,
```

In [39]:
```
arr_model.score(X_test,y_test)
```

Out[39]: 0.6045197740112994

In [40]:

```
from sklearn.tree import DecisionTreeClassifier
arr_clf = DecisionTreeClassifier(random_state=50, min_samples_leaf = 3)

arr_clf = arr_clf.fit(X_train, y_train)
arr_clf.score(X_train, y_train)

arr_clf.feature_importances_
arr_clf_output = pd.DataFrame(arr_clf.predict(X_test), index = X_test.index, columns = ['pred_arr_status']
arr_clf_output.head()
arr_clf.score(X_test, y_test)
```

Out[40]: 0.8940677966101694

Out[40]:
```
array([0.        , 0.02787978, 0.00563367, 0.01607327, 0.02078145,
       0.        , 0.02189077, 0.01564084, 0.05533383, 0.02991479,
       0.01404978, 0.00269427, 0.02301926, 0.013263  , 0.10437207,
       0.04703959, 0.        , 0.0254579 , 0.        , 0.05274309,
       0.0089314 , 0.07239871, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.00416413, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.01576733, 0.        ,
       0.        , 0.        , 0.01060994, 0.        , 0.        ,
       0.        , 0.        , 0.00729505, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.00475072, 0.        ,
       0.        , 0.        , 0.00145161, 0.        , 0.        ,
       0.00316715, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.00270812, 0.        , 0.        , 0.        ,
       0.        , 0.01057101, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.01219862, 0.        , 0.        , 0.        ,
       0.        , 0.00168825, 0.        , 0.        , 0.        ,
       0.00225133, 0.        , 0.        , 0.        , 0.00294306,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.00167006, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.00277125, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.01277225,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.02104347, 0.30541267, 0.00426347, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.00163433, 0.        ,
0.        , 0.00691294, 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.        , 0.        , 0.        , 0.        ,
0.        , 0.00683576, 0.        , 0.        ])
```

Out[40]:

	pred_arr_status
711	1
668	2
821	0
422	1
639	0

Out[40]: 0.5254237288135594

In [41]:

```
from sklearn.ensemble import RandomForestClassifier
arr_rf = RandomForestClassifier(random_state=50, min_samples_leaf = 4, max_features = "sqrt", n_estimators

arr_rf = arr_rf.fit(X_train, y_train)
arr_rf.score(X_train, y_train)

# rf.feature_importances_
feat_imp = pd.Series(arr_rf.feature_importances_, X_train.columns.values).sort_values(ascending=False)
feat_imp_table = pd.DataFrame(feat_imp)
feat_imp_table.head()

arr_rf_output = pd.DataFrame(arr_rf.predict(X_test), index = X_test.index, columns = ['pred_Y'])

arr_rf_output.head()
arr_rf_output = arr_rf_output.merge(y_test, left_index = True, right_index = True)
arr_rf_output.head()
print('Fraction of correct classification ')
arr_rf.score(X_test, y_test)
```

Out[41]: 0.8432203389830508

Out[41]:

	0
dep_status_2	0.17
dep_pres	0.05
dep_status_1	0.04
arr_dewpt	0.04
dep_dewpt	0.04

Out[41]:

	pred_Y
711	0
668	2
821	1
422	0
639	0

Out[41]:

	pred_Y	arr_status_x
711	0	1
668	2	2
821	1	1
422	0	1
639	0	0

```
Fraction of correct classification
```

Out[41]:
```
0.6779661016949152
```

In [42]:
```
from sklearn.ensemble import GradientBoostingClassifier
arr_gb = GradientBoostingClassifier(random_state=50, min_samples_split = 8, min_samples_leaf = 4, n_estima

arr_gb = arr_gb.fit(X_train, y_train)
arr_gb.score(X_train, y_train)
```

Out[42]:
```
1.0
```

In [43]:
```
arr_gb.score(X_test,y_test)
```

Out[43]:
```
0.632768361581921
```

preprocessing data to predict departure status

In [44]:
```
sub_data.columns
sub_data.head()
```

Out[44]:
```
Index(['Carrier_Code', 'Date', 'Flight_Number', 'Tail_Number',
       'Scheduled departure time', 'dep_hour', 'dep_day', 'dep_order',
       'Origin_Airport', 'Scheduled Arrival Time', 'Arrival Delay (Minutes)',
       'arr_hour', 'arr_day', 'dep_min', 'arr_min', 'dep_hours', 'arr_hours',
       'dep_status', 'arr_status'],
      dtype='object')
```

Out[44]:

	Carrier_Code	Date	Flight_Number	Tail_Number	Scheduled departure time	dep_hour	dep_day	dep_order	Origin_Airport	Scheduled Arrival Time	(M
0	MQ	01/04/2020	3,580.00	N240NN	7:55	7	5	latter	ORD	10:47	
1	MQ	01/11/2020	3,946.00	N247NN	15:00	15	5	latter	ORD	17:48	
2	MQ	01/18/2020	3,946.00	N265NN	15:00	15	5	latter	ORD	17:48	
3	MQ	01/25/2020	3,946.00	N281NN	15:00	15	5	latter	ORD	17:48	
4	MQ	02/01/2020	3,946.00	N283NN	15:00	15	5	latter	ORD	17:48	

In [45]:
```
sub_data.dtypes
```

Out[45]:
```
Carrier_Code                 object
Date                         object
Flight_Number                object
Tail_Number                  object
Scheduled departure time     object
dep_hour                     object
dep_day                      object
dep_order                    object
Origin_Airport               object
Scheduled Arrival Time       object
Arrival Delay (Minutes)     float64
arr_hour                     object
arr_day                      object
dep_min                      object
arr_min                      object
dep_hours                    object
arr_hours                    object
dep_status                    int32
arr_status                    int32
dtype: object
```

In [46]:
```
s_data = sub_data.drop(columns=['Carrier_Code', 'Flight_Number', 'Tail_Number',
       'Scheduled departure time', 'dep_order', 'Scheduled Arrival Time', 'Arrival Delay (Minutes)',
        'arr_status'])
```

In [47]:
```
s_data.columns
```

Out[47]:
```
Index(['Date', 'dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_hours', 'arr_hours', 'dep_status'],
      dtype='object')
```

In [48]:
```
s_data = pd.merge(s_data, weather_data, how='left', on=['Origin_Airport', 'Date', 'dep_hours'])


s_data.head()
```

Out[48]:

	Date	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	dep_hours	arr_hours	...	dep_revision_sta
0	01/04/2020	7	5	ORD	10	5	55	47	7	10	...	fi
1	01/11/2020	15	5	ORD	17	5	0	48	15	17	...	fi
2	01/18/2020	15	5	ORD	17	5	0	48	15	17	...	fi
3	01/25/2020	15	5	ORD	17	5	0	48	15	17	...	fi
4	02/01/2020	15	5	ORD	17	5	0	48	15	17	...	fi

5 rows × 28 columns

In [49]:
```
# Define merging logic based on airport code
s_data = pd.merge(s_data, syr_weather_data, how='left', on=['Date', 'arr_hours'])
s_data.head()
```

Out[49]:

	Date	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	dep_hours	arr_hours	...	arr_revision_statu
0	01/04/2020	7	5	ORD	10	5	55	47	7	10	...	fir
1	01/11/2020	15	5	ORD	17	5	0	48	15	17	...	fir
2	01/18/2020	15	5	ORD	17	5	0	48	15	17	...	fir
3	01/25/2020	15	5	ORD	17	5	0	48	15	17	...	fir
4	02/01/2020	15	5	ORD	17	5	0	48	15	17	...	fir

5 rows × 45 columns

In [50]:
```
s_data.columns
```

Out[50]:
```
Index(['Date', 'dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_hours', 'arr_hours', 'dep_status',
       'dep_azimuth', 'dep_clouds', 'dep_dewpt', 'dep_elev_angle',
       'dep_h_angle', 'dep_precip', 'dep_pres', 'dep_revision_status',
       'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis', 'dep_weather.description',
       'dep_weather.code', 'dep_wind_dir', 'dep_wind_gust_spd', 'dep_wind_spd',
       'arr_azimuth', 'arr_clouds', 'arr_dewpt', 'arr_elev_angle',
       'arr_h_angle', 'arr_precip', 'arr_pres', 'arr_revision_status',
       'arr_rh', 'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.description',
       'arr_weather.code', 'arr_wind_dir', 'arr_wind_gust_spd',
       'arr_wind_spd'],
      dtype='object')
```

In [51]:
```
s_data.drop(columns= ['Date','arr_azimuth',
                       'arr_hours', 'dep_hours',
                       'arr_weather.description', 'dep_weather.description'
                       ,'arr_elev_angle','arr_h_angle','arr_revision_status',
                      'dep_elev_angle','dep_h_angle','dep_revision_status','dep_azimuth'],inplace = True)
s_data.columns
```

Out[51]:
```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_status', 'dep_clouds', 'dep_dewpt',
       'dep_precip', 'dep_pres', 'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis',
       'dep_weather.code', 'dep_wind_dir', 'dep_wind_gust_spd', 'dep_wind_spd',
       'arr_clouds', 'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh',
       'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.code', 'arr_wind_dir',
       'arr_wind_gust_spd', 'arr_wind_spd'],
      dtype='object')
```

In [52]:

```
s_data.dtypes
s_data.columns

s_data.head()
s_data.isna().sum()
```

Out[52]:

```
dep_hour               object
dep_day                object
Origin_Airport         object
arr_hour               object
arr_day                object
dep_min                object
arr_min                object
dep_status              int32
dep_clouds              int64
dep_dewpt             float64
dep_precip            float64
dep_pres                int64
dep_rh                  int64
dep_snow              float64
dep_temp              float64
dep_vis                 int64
dep_weather.code        int64
dep_wind_dir            int64
dep_wind_gust_spd     float64
dep_wind_spd          float64
arr_clouds              int64
arr_dewpt             float64
arr_precip            float64
arr_pres                int64
arr_rh                  int64
arr_snow              float64
arr_temp              float64
arr_vis                 int64
arr_weather.code        int64
arr_wind_dir            int64
arr_wind_gust_spd     float64
arr_wind_spd          float64
dtype: object
```

Out[52]:
```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_status', 'dep_clouds', 'dep_dewpt',
       'dep_precip', 'dep_pres', 'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis',
       'dep_weather.code', 'dep_wind_dir', 'dep_wind_gust_spd', 'dep_wind_spd',
       'arr_clouds', 'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh',
       'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.code', 'arr_wind_dir',
       'arr_wind_gust_spd', 'arr_wind_spd'],
      dtype='object')
```

Out[52]:

	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	dep_status	dep_clouds	dep_dewpt	...	arr_precip	ar
0	7	5	ORD	10	5	55	47	2	100	-2.80	...	1.50	
1	15	5	ORD	17	5	0	48	2	100	-1.20	...	1.50	
2	15	5	ORD	17	5	0	48	2	100	-6.80	...	1.00	
3	15	5	ORD	17	5	0	48	1	100	-0.30	...	1.50	
4	15	5	ORD	17	5	0	48	1	100	-2.00	...	0.00	

5 rows × 32 columns

Out[52]:
```
dep_hour             0
dep_day              0
Origin_Airport       0
arr_hour             0
arr_day              0
dep_min              0
arr_min              0
dep_status           0
dep_clouds           0
dep_dewpt            0
dep_precip           0
dep_pres             0
dep_rh               0
dep_snow             0
dep_temp             0
dep_vis              0
dep_weather.code     0
dep_wind_dir         0
dep_wind_gust_spd    0
dep_wind_spd         0
arr_clouds           0
arr_dewpt            0
arr_precip           0
arr_pres             0
arr_rh               0
arr_snow             0
arr_temp             0
arr_vis              0
arr_weather.code     0
arr_wind_dir         0
arr_wind_gust_spd    0
arr_wind_spd         0
dtype: int64
```

In [53]:

```
su_data = s_data
su_data['dep_hour'] = pd.Categorical(su_data['dep_hour'], categories=[i for i in range(24)])
su_data['dep_day'] = pd.Categorical(su_data['dep_day'], categories=[i for i in range(7)])
su_data['dep_min'] = pd.Categorical(su_data['dep_min'], categories=[i for i in range(60)])
su_data['arr_hour'] = pd.Categorical(su_data['arr_hour'], categories=[i for i in range(24)])
su_data['arr_day'] = pd.Categorical(su_data['arr_day'], categories=[i for i in range(7)])
su_data['arr_min'] = pd.Categorical(su_data['arr_min'], categories=[i for i in range(60)])
su_data['Origin_Airport'] = pd.Categorical(su_data['Origin_Airport'], categories=['ORD', 'JFK', 'MCO'])
su_data['arr_weather.code'] = pd.Categorical(su_data['arr_weather.code'], categories=[200,201,202,230,231,
su_data['dep_weather.code'] = pd.Categorical(su_data['dep_weather.code'], categories=[200,201,202,230,231,

su_data.columns
```

Out[53]:

```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_status', 'dep_clouds', 'dep_dewpt',
       'dep_precip', 'dep_pres', 'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis',
       'dep_weather.code', 'dep_wind_dir', 'dep_wind_gust_spd', 'dep_wind_spd',
       'arr_clouds', 'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh',
       'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.code', 'arr_wind_dir',
       'arr_wind_gust_spd', 'arr_wind_spd'],
      dtype='object')
```

In [54]:

```
su_data['dep_precip'] = su_data['dep_precip']**2
su_data['arr_precip'] = su_data['arr_precip']**2
```

Traning model to predict depature status

In [55]:

```
dep_data = pd.get_dummies(su_data, drop_first = True)
dep_data.head()
```

Out[55]:

	dep_status	dep_clouds	dep_dewpt	dep_precip	dep_pres	dep_rh	dep_snow	dep_temp	dep_vis	dep_wind_dir	...	arr_weathe
0	2	100	-2.80	0.00	989	88	0.00	-1.10	11	320	...	
1	2	100	-1.20	0.25	985	96	8.50	-0.60	2	20	...	
2	2	100	-6.80	0.06	983	86	6.25	-4.80	14	250	...	
3	1	100	-0.30	0.25	985	92	4.00	0.80	6	255	...	
4	1	100	-2.00	0.00	984	73	0.00	2.40	16	255	...	

5 rows × 271 columns

In [56]:

```
X_train, X_test, y_train, y_test = train_test_split(dep_data.drop(columns = ['dep_status']), dep_data['dep
X_train
X_test
y_train.dtypes
y_test
```

Out[56]:

	dep_clouds	dep_dewpt	dep_precip	dep_pres	dep_rh	dep_snow	dep_temp	dep_vis	dep_wind_dir	dep_wind_gust_spd	...
940	87	18.20	0.00	984	66	0.00	25.00	16	250	2.80	...
8173	87	20.60	0.00	1007	63	0.00	28.30	16	60	9.80	...
7040	87	3.50	0.00	1002	44	0.00	15.70	16	295	13.90	...
5583	87	11.00	0.00	1007	74	0.00	15.60	16	170	10.80	...
425	0	-1.70	0.00	1000	63	0.00	4.70	16	210	4.00	...
...	...	...	...	...	...	...	...	...	...	...	...
5905	78	13.50	0.00	1021	57	0.00	22.50	16	245	4.80	...
6597	87	3.80	0.00	1005	28	0.00	23.30	16	215	10.00	...
7866	25	12.60	0.00	1015	46	0.00	25.00	16	230	8.80	...
1419	87	-3.80	0.00	985	42	0.00	8.30	16	185	6.40	...
1802	18	-2.20	0.00	1001	54	0.00	6.40	16	200	3.20	...

6928 rows × 270 columns

Out[56]:

	dep_clouds	dep_dewpt	dep_precip	dep_pres	dep_rh	dep_snow	dep_temp	dep_vis	dep_wind_dir	dep_wind_gust_spd	...
4600	25	17.70	0.00	1006	60	0.00	26.10	16	310	4.72	...
5412	50	5.60	0.06	1031	86	0.00	7.80	16	170	7.20	...
6801	87	16.80	0.00	1015	72	0.00	22.10	16	100	3.60	...
6567	31	6.10	0.00	1022	41	0.00	19.70	16	315	7.70	...
3051	100	10.40	4.00	1004	92	0.00	11.70	5	120	7.20	...
...	...	...	...	...	...	...	...	...	...	...	...
4135	96	11.90	0.00	1014	86	0.00	14.30	4	95	6.00	...
8534	50	24.20	0.00	1010	81	0.00	27.80	16	260	8.20	...
2115	25	-2.10	0.00	1000	22	0.00	20.30	16	45	7.60	...
7381	87	15.50	0.00	1012	66	0.00	22.20	16	140	4.00	...
2537	87	6.30	0.00	993	67	0.00	12.20	16	335	6.00	...

1733 rows × 270 columns

Out[56]:
```
dtype('int32')
```

Out[56]:
```
4600    2
5412    2
6801    2
6567    2
3051    2
       ..
4135    1
8534    1
2115    2
7381    0
2537    1
Name: dep_status, Length: 1733, dtype: int32
```

In [57]:
```
from sklearn.preprocessing import StandardScaler
sc1 = StandardScaler()
X_train = pd.DataFrame(sc1.fit_transform(X_train), columns = X_train.columns, index = X_train.index)
X_test = pd.DataFrame(sc1.transform(X_test), columns = X_test.columns, index = X_test.index)
X_train
X_test
y_train
y_test
```

Out[57]:

	dep_clouds	dep_dewpt	dep_precip	dep_pres	dep_rh	dep_snow	dep_temp	dep_vis	dep_wind_dir	dep_wind_gust_spd	...
940	0.63	1.02	-0.08	-1.77	0.16	-0.06	0.96	0.31	0.57	-1.45	...
8173	0.63	1.26	-0.08	-0.06	0.00	-0.06	1.30	0.31	-1.38	0.64	...
7040	0.63	-0.44	-0.08	-0.43	-1.00	-0.06	0.01	0.31	1.03	1.86	...
5583	0.63	0.31	-0.08	-0.06	0.58	-0.06	0.00	0.31	-0.25	0.93	...
425	-2.16	-0.95	-0.08	-0.58	0.00	-0.06	-1.12	0.31	0.16	-1.10	...
...	...	...	...	...	...	...	...	...	...	...	...
5905	0.34	0.56	-0.08	0.98	-0.32	-0.06	0.71	0.31	0.51	-0.86	...
6597	0.63	-0.41	-0.08	-0.21	-1.85	-0.06	0.79	0.31	0.21	0.70	...
7866	-1.36	0.47	-0.08	0.54	-0.90	-0.06	0.96	0.31	0.36	0.34	...
1419	0.63	-1.16	-0.08	-1.70	-1.11	-0.06	-0.75	0.31	-0.10	-0.38	...
1802	-1.58	-1.00	-0.08	-0.51	-0.47	-0.06	-0.94	0.31	0.05	-1.33	...

6928 rows × 270 columns

Out[57]:

	dep_clouds	dep_dewpt	dep_precip	dep_pres	dep_rh	dep_snow	dep_temp	dep_vis	dep_wind_dir	dep_wind_gust_spd	...
4600	-1.36	0.98	-0.08	-0.13	-0.16	-0.06	1.08	0.31	1.18	-0.88	...
5412	-0.56	-0.23	-0.07	1.73	1.22	-0.06	-0.80	0.31	-0.25	-0.14	...
6801	0.63	0.89	-0.08	0.54	0.48	-0.06	0.67	0.31	-0.97	-1.22	...
6567	-1.16	-0.18	-0.08	1.06	-1.16	-0.06	0.42	0.31	1.23	0.01	...
3051	1.04	0.25	0.25	-0.28	1.53	-0.06	-0.40	-3.46	-0.77	-0.14	...
...	...	...	...	...	...	...	...	...	...	...	...
4135	0.91	0.40	-0.08	0.46	1.22	-0.06	-0.13	-3.80	-1.02	-0.50	...
8534	-0.56	1.62	-0.08	0.17	0.95	-0.06	1.25	0.31	0.67	0.16	...
2115	-1.36	-0.99	-0.08	-0.58	-2.16	-0.06	0.48	0.31	-1.53	-0.02	...
7381	0.63	0.76	-0.08	0.31	0.16	-0.06	0.68	0.31	-0.56	-1.10	...
2537	0.63	-0.16	-0.08	-1.10	0.21	-0.06	-0.35	0.31	1.44	-0.50	...

1733 rows × 270 columns

Out[57]:

```
940     0
8173    1
7040    0
5583    0
425     2
       ..
5905    0
6597    2
7866    2
1419    1
1802    1
Name: dep_status, Length: 6928, dtype: int32
```

Out[57]:
```
4600    2
5412    2
6801    2
6567    2
3051    2
       ..
4135    1
8534    1
2115    2
7381    0
2537    1
Name: dep_status, Length: 1733, dtype: int32
```

In [58]:
```
dep_model = LogisticRegression(fit_intercept = True, solver='lbfgs', multi_class = 'ovr', penalty = None,


dep_model.fit(X_train, y_train)

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

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

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

Out[58]:
```
LogisticRegression
LogisticRegression(max_iter=1000, multi_class='ovr', penalty=None)
```

Out[58]: 0.5682736720554272

Out[58]:
```
array([[-2.12950348e-01,  2.74221788e-01, -1.59691385e-01,
         7.02226357e-02, -1.27390548e-01, -1.62686825e-01,
        -2.31168121e-01,  2.08375587e-02, -8.26508543e-02,
        -2.14765022e-01,  1.40362589e-01,  4.18705251e-01,
        -4.31960468e-02, -2.20665464e-03, -3.48607366e-02,
        -3.67809628e-02,  2.58951456e-02, -4.49678004e-02,
         1.02600835e-01, -1.40208958e-02, -1.52916027e-02,
         3.10537204e-02,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        -9.16226034e-01, -1.23326505e+00, -2.67662941e+00,
        -3.10503751e+00, -1.41281306e+00, -1.35203552e+00,
        -8.48159921e-01, -8.90070097e-01, -2.94075588e-01,
        -9.83136825e-02,  1.85040873e-01,  2.45850881e-01,
```

In [59]:
```
dep_model.score(X_test,y_test)
```

Out[59]: 0.5222158107328332

predicting outputs

In [60]:

```
pred_data2 = pd.read_csv('pred_data2.csv')
pred_data2.head()
pred_data2.dtypes
```

Out[60]:

	Unnamed: 0	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	arr_clouds	arr_dewpt	...	dep_precip	dep_
0	1	19	4	ORD	22	4	59	52	66	7.20	...	0.00	9
1	3	14	4	JFK	16	4	55	21	86	7.90	...	0.00	1,0
2	5	13	4	MCO	16	4	35	25	86	7.90	...	0.00	1,0
3	7	19	5	ORD	22	5	59	52	23	-2.40	...	0.00	9
4	9	14	5	JFK	16	5	55	21	58	-1.60	...	0.00	1,0

5 rows × 32 columns

Out[60]:
```
Unnamed: 0              int64
dep_hour                int64
dep_day                 int64
Origin_Airport         object
arr_hour                int64
arr_day                 int64
dep_min                 int64
arr_min                 int64
arr_clouds              int64
arr_dewpt             float64
arr_precip            float64
arr_pres              float64
arr_rh                  int64
arr_snow                int64
arr_temp              float64
arr_vis               float64
arr_weather.code        int64
arr_wind_dir            int64
arr_wind_gust_spd     float64
arr_wind_spd          float64
dep_clouds              int64
dep_dewpt             float64
dep_precip            float64
dep_pres              float64
dep_rh                  int64
dep_snow                int64
dep_temp              float64
dep_vis               float64
dep_weather.code        int64
dep_wind_dir            int64
dep_wind_gust_spd     float64
dep_wind_spd          float64
dtype: object
```

In [61]:

```
pred_data2['dep_min'] = pred_data2['dep_min'].astype('object')
pred_data2['arr_min'] = pred_data2['arr_min'].astype('object')
pred_data2['dep_hour'] = pred_data2['dep_hour'].astype('object')
pred_data2['dep_day'] = pred_data2['dep_day'].astype('object')
pred_data2['arr_hour'] = pred_data2['arr_hour'].astype('object')
pred_data2['arr_day'] = pred_data2['arr_day'].astype('object')
pred_data2['dep_weather.code'] = pred_data2['dep_weather.code'].astype('object')
pred_data2['arr_weather.code'] = pred_data2['arr_weather.code'].astype('object')
pred_data2.drop(columns=['Unnamed: 0'],inplace=True)

pred_data2.dtypes
```

Out[61]:
```
dep_hour             object
dep_day              object
Origin_Airport       object
arr_hour             object
arr_day              object
dep_min              object
arr_min              object
arr_clouds            int64
arr_dewpt           float64
arr_precip          float64
arr_pres            float64
arr_rh                int64
arr_snow              int64
arr_temp            float64
arr_vis             float64
arr_weather.code     object
arr_wind_dir          int64
arr_wind_gust_spd   float64
arr_wind_spd        float64
dep_clouds            int64
dep_dewpt           float64
dep_precip          float64
dep_pres            float64
dep_rh                int64
dep_snow              int64
dep_temp            float64
dep_vis             float64
dep_weather.code     object
dep_wind_dir          int64
dep_wind_gust_spd   float64
dep_wind_spd        float64
dtype: object
```

In [62]:

```
pred_data2['dep_hour'] = pd.Categorical(pred_data2['dep_hour'], categories=[i for i in range(24)])
pred_data2['dep_day'] = pd.Categorical(pred_data2['dep_day'], categories=[i for i in range(7)])
pred_data2['dep_min'] = pd.Categorical(pred_data2['dep_min'], categories=[i for i in range(60)])
pred_data2['arr_hour'] = pd.Categorical(pred_data2['arr_hour'], categories=[i for i in range(24)])
pred_data2['arr_day'] = pd.Categorical(pred_data2['arr_day'], categories=[i for i in range(7)])
pred_data2['arr_min'] = pd.Categorical(pred_data2['arr_min'], categories=[i for i in range(60)])
pred_data2['Origin_Airport'] = pd.Categorical(pred_data2['Origin_Airport'], categories=['ORD', 'JFK', 'MCC

pred_data2['arr_weather.code'] = pd.Categorical(pred_data2['arr_weather.code'], categories=[200,201,202,23
pred_data2['dep_weather.code'] = pd.Categorical(pred_data2['dep_weather.code'], categories=[200,201,202,23
```

In [63]:

```
pred_data = pred_data2
```

In [64]:

```
pred_data.columns
pred_data.head()
```

Out[64]:

```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'arr_clouds', 'arr_dewpt', 'arr_precip',
       'arr_pres', 'arr_rh', 'arr_snow', 'arr_temp', 'arr_vis',
       'arr_weather.code', 'arr_wind_dir', 'arr_wind_gust_spd', 'arr_wind_spd',
       'dep_clouds', 'dep_dewpt', 'dep_precip', 'dep_pres', 'dep_rh',
       'dep_snow', 'dep_temp', 'dep_vis', 'dep_weather.code', 'dep_wind_dir',
       'dep_wind_gust_spd', 'dep_wind_spd'],
      dtype='object')
```

Out[64]:

	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	arr_clouds	arr_dewpt	arr_precip	...	dep_precip	dep_
0	19	4	ORD	22	4	59	52	66	7.20	0.25	...	0.00	9
1	14	4	JFK	16	4	55	21	86	7.90	0.76	...	0.00	1,0
2	13	4	MCO	16	4	35	25	86	7.90	0.76	...	0.00	1,0
3	19	5	ORD	22	5	59	52	23	-2.40	0.00	...	0.00	9
4	14	5	JFK	16	5	55	21	58	-1.60	0.00	...	0.00	1,0

5 rows × 31 columns

In [65]:
```
pred_data = pred_data[['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_clouds', 'dep_dewpt',
       'dep_precip', 'dep_pres', 'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis',
       'dep_weather.code', 'dep_wind_dir', 'dep_wind_gust_spd', 'dep_wind_spd',
       'arr_clouds', 'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh',
       'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.code', 'arr_wind_dir',
       'arr_wind_gust_spd', 'arr_wind_spd']]
pred_data.columns
```

Out[65]:
```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_clouds', 'dep_dewpt', 'dep_precip',
       'dep_pres', 'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis',
       'dep_weather.code', 'dep_wind_dir', 'dep_wind_gust_spd', 'dep_wind_spd',
       'arr_clouds', 'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh',
       'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.code', 'arr_wind_dir',
       'arr_wind_gust_spd', 'arr_wind_spd'],
      dtype='object')
```

applying logistic regression model to predict departure status

In [66]:
```
dep_data1 = pd.get_dummies(pred_data, drop_first = True)
dep_data1.head()
```

Out[66]:

	dep_clouds	dep_dewpt	dep_precip	dep_pres	dep_rh	dep_snow	dep_temp	dep_vis	dep_wind_dir	dep_wind_gust_spd	...	a
0	50	-2.10	0.00	996.50	37	0	12.10	24.00	280	11.10	...	
1	85	4.50	0.00	1,019.00	59	0	12.30	24.00	120	7.30	...	
2	6	17.60	0.00	1,013.50	45	0	30.90	24.00	270	3.20	...	
3	71	-3.10	0.00	994.50	41	0	9.50	24.13	300	6.00	...	
4	65	3.60	0.00	1,014.00	42	0	16.50	24.00	270	6.00	...	

5 rows × 270 columns

In [67]:
```
X_test = pd.DataFrame(sc1.transform(dep_data1), columns = dep_data1.columns, index = dep_data1.index)
```

In [68]:
```
dep_model_output = pd.DataFrame(dep_model.predict(X_test), index = X_test.index, columns = ['dep_status'])
dep_model_output = dep_model_output.merge(pred_data2, left_index = True, right_index = True)
dep_model_output.head(20)
```

Out[68]:

	dep_status	dep_hour	dep_day	Origin_Airport	arr_hour	arr_day	dep_min	arr_min	arr_clouds	arr_dewpt	...	dep_precip	de
0	1	19	4	ORD	22	4	59	52	66	7.20	...	0.00	
1	1	14	4	JFK	16	4	55	21	86	7.90	...	0.00	1
2	1	13	4	MCO	16	4	35	25	86	7.90	...	0.00	1
3	1	19	5	ORD	22	5	59	52	23	-2.40	...	0.00	
4	1	14	5	JFK	16	5	55	21	58	-1.60	...	0.00	1
5	0	19	6	ORD	22	6	59	52	9	-2.90	...	0.00	
6	1	14	6	JFK	16	6	55	21	32	-3.20	...	0.00	1
7	1	13	6	MCO	16	6	35	25	32	-3.20	...	0.00	1
8	0	19	0	ORD	22	0	59	52	13	-4.30	...	0.00	
9	1	14	0	JFK	16	0	55	21	9	-4.60	...	0.00	1
10	2	13	0	MCO	16	0	34	25	9	-4.60	...	0.20	1

11 rows × 32 columns

In [69]:
```
len(pred_data2)
```

Out[69]: 11

In [70]:
```
my_data = {'arr_status_y': [0, 1, 2]}
df = pd.DataFrame(data = my_data)
```

In [71]:
```
df.head()
```

Out[71]:

	arr_status_y
0	0
1	1
2	2

In [72]:
```
dep_model_output =dep_model_output.merge(df,how = "cross")
len(dep_model_output)
```

Out[72]: 33

In [73]:
```
dep_model_output['arr_status_y'] = pd.Categorical(dep_model_output['arr_status_y'], categories=[0,1,2])
```

In [74]:
```
pred_data2 = dep_model_output[['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_status', 'arr_status_y',
       'arr_clouds', 'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh',
       'arr_snow', 'arr_temp', 'arr_vis', 'arr_weather.code', 'arr_wind_dir',
       'arr_wind_gust_spd', 'arr_wind_spd', 'dep_clouds', 'dep_dewpt',
       'dep_precip', 'dep_pres', 'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis',
       'dep_weather.code', 'dep_wind_dir', 'dep_wind_gust_spd',
       'dep_wind_spd']]
pred_data2.columns
```

Out[74]:
```
Index(['dep_hour', 'dep_day', 'Origin_Airport', 'arr_hour', 'arr_day',
       'dep_min', 'arr_min', 'dep_status', 'arr_status_y', 'arr_clouds',
       'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh', 'arr_snow', 'arr_temp',
       'arr_vis', 'arr_weather.code', 'arr_wind_dir', 'arr_wind_gust_spd',
       'arr_wind_spd', 'dep_clouds', 'dep_dewpt', 'dep_precip', 'dep_pres',
       'dep_rh', 'dep_snow', 'dep_temp', 'dep_vis', 'dep_weather.code',
       'dep_wind_dir', 'dep_wind_gust_spd', 'dep_wind_spd'],
      dtype='object')
```

In [75]:
```
pred_data2['dep_status'] = pred_data2['dep_status'].astype('int64').astype('object')
```

In [76]:
```
pred_data2['dep_status'] = pd.Categorical(pred_data2['dep_status'], categories=[0,1,2])
```

applying random forest model to predict arrival status

In [77]:
```
pred_data2 = pd.get_dummies(pred_data2, drop_first = True)
pred_data2.head()
pred_data2.dtypes
pred_data2.columns
```

Out[77]:

	arr_clouds	arr_dewpt	arr_precip	arr_pres	arr_rh	arr_snow	arr_temp	arr_vis	arr_wind_dir	arr_wind_gust_spd	...	dep_weath
0	66	7.20	0.25	999.00	73	0	11.90	19.49	260	8.10	...	
1	66	7.20	0.25	999.00	73	0	11.90	19.49	260	8.10	...	
2	66	7.20	0.25	999.00	73	0	11.90	19.49	260	8.10	...	
3	86	7.90	0.76	996.00	59	0	15.90	20.80	180	13.20	...	
4	86	7.90	0.76	996.00	59	0	15.90	20.80	180	13.20	...	

5 rows × 274 columns

Out[77]:
```
arr_clouds                 int64
arr_dewpt                float64
arr_precip               float64
arr_pres                 float64
arr_rh                     int64
                          ...
dep_weather.code_801        bool
dep_weather.code_802        bool
dep_weather.code_803        bool
dep_weather.code_804        bool
dep_weather.code_900        bool
Length: 274, dtype: object
```

Out[77]:
```
Index(['arr_clouds', 'arr_dewpt', 'arr_precip', 'arr_pres', 'arr_rh',
       'arr_snow', 'arr_temp', 'arr_vis', 'arr_wind_dir', 'arr_wind_gust_spd',
       ...
       'dep_weather.code_721', 'dep_weather.code_731', 'dep_weather.code_741',
       'dep_weather.code_751', 'dep_weather.code_800', 'dep_weather.code_801',
       'dep_weather.code_802', 'dep_weather.code_803', 'dep_weather.code_804',
       'dep_weather.code_900'],
      dtype='object', length=274)
```

In [78]:
```
from sklearn.preprocessing import StandardScaler
X_test = pd.DataFrame(sc.transform(pred_data2), columns = pred_data2.columns, index = pred_data2.index)
```

In [79]:

```
arr_rf_output = pd.DataFrame(arr_rf.predict(X_test), index = X_test.index, columns = ['pred_arr_status'])
arr_rf_output.head(50)
```

Out[79]:

	pred_arr_status
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0

	pred_arr_status
26	0
27	0
28	0
29	0
30	2
31	2
32	2

In [ ]:

```

```