Flight Delay Prediction

1 Project: Flight Delay Predicition

[120]:
```
# Necessary imports

import ast
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import datetime as dt
from sklearn.model_selection import train_test_split

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

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from IPython.display import display, HTML

import plotly
plotly.offline.init_notebook_mode(connected=True)
from plotly.graph_objs import *
from plotly import tools
import plotly.graph_objects as go
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import classification_report
import xgboost as xgb


from weatherbit.api import Api
```

```
import requests
```

1.1 Data Extraction

[3]:
```
# Flight data downloaded from transtats.bts.gov
init_data = pd.read_csv("datasets/united_2022.csv")
init_data.head(30)
```

[3]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Tail Number	Origin Airport	Scheduled Arrival Time	Actual Arrival Time
0	UA	1/1/2022	1282.0	N4901U	IAD	23:10	0:01
1	UA	1/2/2022	1282.0	N4901U	IAD	23:10	23:27
2	UA	1/3/2022	1282.0	N893UA	IAD	23:10	23:31
3	UA	1/4/2022	1282.0	N814UA	IAD	23:44	1:59
4	UA	1/5/2022	1282.0	N827UA	IAD	23:44	23:30
5	UA	1/6/2022	1282.0	N485UA	IAD	23:44	23:45
6	UA	1/7/2022	1282.0	N423UA	IAD		
7	UA	1/8/2022	1282.0	N893UA	IAD		
8	UA	1/9/2022	1282.0	N854UA	IAD		
9	UA	1/10/2022	1282.0	N810UA	IAD		
10	UA	1/11/2022	1282.0	N880UA	IAD		
11	UA	1/12/2022	1282.0	N871UA	IAD		
12	UA	1/13/2022	1282.0	N845UA	IAD		
13	UA	1/14/2022	1282.0	N469UA	IAD		
14	UA	1/15/2022	1282.0	N818UA	IAD		
15	UA	1/16/2022	1282.0	NaN	IAD		
16	UA	1/17/2022	1282.0	N420UA	IAD		
17	UA	1/18/2022	1282.0	N833UA	IAD		
18	UA	1/19/2022	1282.0	N877UA	IAD		
19	UA	1/20/2022	1282.0	N825UA	IAD		
20	UA	1/21/2022	1282.0	N416UA	IAD		
21	UA	1/22/2022	1282.0	N803UA	IAD		
22	UA	1/23/2022	1282.0	N489UA	IAD		
23	UA	1/24/2022	1282.0	N449UA	IAD		
24	UA	1/25/2022	1282.0	N490UA	IAD		
25	UA	1/26/2022	1282.0	N827UA	IAD		
26	UA	1/27/2022	1282.0	N414UA	IAD		
27	UA	1/28/2022	1282.0	N421UA	IAD		
28	UA	1/29/2022	1282.0	N469UA	IAD		
29	UA	1/30/2022	1282.0	N828UA	IAD		

6	23:44	0:01
7	23:44	23:38
8	23:44	1:26
9	23:44	0:14
10	23:44	23:26
11	23:44	23:30
12	23:44	23:41
13	23:44	23:37
14	23:44	23:36
15	23:44	0:00
16	23:44	23:55
17	23:44	23:36
18	23:44	23:35
19	23:44	23:25
20	23:44	23:28
21	23:44	0:05
22	23:44	0:04
23	23:44	23:28
24	23:44	23:33
25	23:44	23:33
26	23:44	23:28
27	23:44	23:50
28	23:44	23:36
29	23:44	23:37

	Scheduled Elapsed Time (Minutes)	Actual Elapsed Time (Minutes)	\
0	70.0	76.0	
1	70.0	64.0	
2	70.0	68.0	
3	69.0	89.0	
4	69.0	61.0	
5	69.0	66.0	
6	69.0	78.0	
7	69.0	63.0	
8	69.0	72.0	
9	69.0	77.0	
10	69.0	59.0	
11	69.0	67.0	
12	69.0	60.0	
13	69.0	72.0	
14	69.0	64.0	
15	69.0	0.0	
16	69.0	75.0	
17	69.0	66.0	
18	69.0	68.0	
19	69.0	65.0	
20	69.0	63.0	

21	69.0	64.0
22	69.0	95.0
23	69.0	66.0
24	69.0	68.0
25	69.0	70.0
26	69.0	67.0
27	69.0	75.0
28	69.0	75.0
29	69.0	69.0

	Arrival Delay (Minutes)	Wheels-on Time	Taxi-In time (Minutes)	\
0	51.0	23:55	6.0	
1	17.0	23:19	8.0	
2	21.0	23:25	6.0	
3	135.0	1:55	4.0	
4	-14.0	23:25	5.0	
5	1.0	23:38	7.0	
6	17.0	23:52	9.0	
7	-6.0	23:33	5.0	
8	102.0	1:19	7.0	
9	30.0	0:08	6.0	
10	-18.0	23:18	8.0	
11	-14.0	23:24	6.0	
12	-3.0	23:37	4.0	
13	-7.0	23:24	13.0	
14	-8.0	23:31	5.0	
15	0.0	0:00	0.0	
16	11.0	23:50	5.0	
17	-8.0	23:30	6.0	
18	-9.0	23:30	5.0	
19	-19.0	23:20	5.0	
20	-16.0	23:23	5.0	
21	21.0	23:59	6.0	
22	20.0	23:31	33.0	
23	-16.0	23:23	5.0	
24	-11.0	23:24	9.0	
25	-11.0	23:28	5.0	
26	-16.0	23:22	6.0	
27	6.0	23:46	4.0	
28	-8.0	23:30	6.0	
29	-7.0	23:30	7.0	

	Delay Carrier (Minutes)	Delay Weather (Minutes)	\
0	23.0	0.0	
1	17.0	0.0	
2	21.0	0.0	
3	115.0	0.0	

4	0.0	0.0
5	0.0	0.0
6	8.0	0.0
7	0.0	0.0
8	99.0	0.0
9	0.0	22.0
10	0.0	0.0
11	0.0	0.0
12	0.0	0.0
13	0.0	0.0
14	0.0	0.0
15	0.0	0.0
16	0.0	0.0
17	0.0	0.0
18	0.0	0.0
19	0.0	0.0
20	0.0	0.0
21	21.0	0.0
22	0.0	0.0
23	0.0	0.0
24	0.0	0.0
25	0.0	0.0
26	0.0	0.0
27	0.0	0.0
28	0.0	0.0
29	0.0	0.0

	Delay National Aviation System (Minutes)	Delay Security (Minutes)	\
0	6.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	20.0	0.0	
4	0.0	0.0	
5	0.0	0.0	
6	9.0	0.0	
7	0.0	0.0	
8	3.0	0.0	
9	8.0	0.0	
10	0.0	0.0	
11	0.0	0.0	
12	0.0	0.0	
13	0.0	0.0	
14	0.0	0.0	
15	0.0	0.0	
16	0.0	0.0	
17	0.0	0.0	
18	0.0	0.0	

19	0.0	0.0
20	0.0	0.0
21	0.0	0.0
22	20.0	0.0
23	0.0	0.0
24	0.0	0.0
25	0.0	0.0
26	0.0	0.0
27	0.0	0.0
28	0.0	0.0
29	0.0	0.0

	Delay Late Aircraft Arrival (Minutes)
0	22.0
1	0.0
2	0.0
3	0.0
4	0.0
5	0.0
6	0.0
7	0.0
8	0.0
9	0.0
10	0.0
11	0.0
12	0.0
13	0.0
14	0.0
15	0.0
16	0.0
17	0.0
18	0.0
19	0.0
20	0.0
21	0.0
22	0.0
23	0.0
24	0.0
25	0.0
26	0.0
27	0.0
28	0.0
29	0.0

[4]:
```
init_data.dtypes
```

[4]:
```
Carrier Code                                  object
Date (MM/DD/YYYY)                             object
Flight Number                                float64
Tail Number                                   object
Origin Airport                                object
Scheduled Arrival Time                        object
Actual Arrival Time                           object
Scheduled Elapsed Time (Minutes)             float64
Actual Elapsed Time (Minutes)                float64
Arrival Delay (Minutes)                      float64
Wheels-on Time                                object
Taxi-In time (Minutes)                       float64
Delay Carrier (Minutes)                      float64
Delay Weather (Minutes)                      float64
Delay National Aviation System (Minutes)     float64
Delay Security (Minutes)                     float64
Delay Late Aircraft Arrival (Minutes)        float64
dtype: object
```

[5]:
```
init_data.shape
```

[5]:
```
(1040, 17)
```

[6]:
```
init_data.isna().sum()
```

[6]:
```
Carrier Code                                  1
Date (MM/DD/YYYY)                             2
Flight Number                                 2
Tail Number                                  18
Origin Airport                                2
Scheduled Arrival Time                        2
Actual Arrival Time                           2
Scheduled Elapsed Time (Minutes)              2
Actual Elapsed Time (Minutes)                 2
Arrival Delay (Minutes)                       2
Wheels-on Time                                2
Taxi-In time (Minutes)                        2
Delay Carrier (Minutes)                       2
Delay Weather (Minutes)                       2
Delay National Aviation System (Minutes)      2
Delay Security (Minutes)                      2
Delay Late Aircraft Arrival (Minutes)         2
dtype: int64
```

[7]:
```
init_data = init_data.dropna()
```

[8]:
```
init_data.shape
```

[8]: (1022, 17)

[9]:
```
set(init_data['Origin Airport'])
```

[9]: {'DEN', 'EWR', 'IAD', 'ORD'}

[5]:
```
# Fetching Weather data at origin airport from Weatherbit API and saving it to␣
 ↪csv file

url = f'https://api.weatherbit.io/v2.0/history/hourly?
 ↪city=Syracuse&start_date=2022-01-01&end_date=2022-12-31&key=ce8801187c3f4d559e2ce6dabf416f5
response = requests.get(url)
if response.status_code == 200:
    print("Data fetched...")
    data = response.json()

    raw_weather_df = pd.DataFrame(data['data'])
    raw_weather_df.to_csv('raw_weather_1.csv', index = False)

else:
    print('Error:', response.status_code)
```

Data fetched…

[10]:
```
raw_weather_df = pd.read_csv("raw_weather_1.csv")
raw_weather_df.head()
```

[10]:

	app_temp	azimuth	clouds	datetime	dewpt	dhi	dni	elev_angle	ghi	\
0	3.9	260.93	0	2022-01-01:00	4.3	0.0	0.0	-24.85	0.0	
1	1.3	270.74	25	2022-01-01:01	3.3	0.0	0.0	-35.75	0.0	
2	0.8	282.10	25	2022-01-01:02	1.6	0.0	0.0	-46.62	0.0	
3	-1.1	297.04	25	2022-01-01:03	1.7	0.0	0.0	-56.93	0.0	
4	-1.1	319.76	25	2022-01-01:04	1.7	0.0	0.0	-65.55	0.0	

	h_angle	…	temp	timestamp_local	timestamp_utc	ts	\
0	NaN	…	5.0	2021-12-31T19:00:00	2022-01-01T00:00:00	1640995200	
1	NaN	…	3.3	2021-12-31T20:00:00	2022-01-01T01:00:00	1640998800	
2	NaN	…	2.2	2021-12-31T21:00:00	2022-01-01T02:00:00	1641002400	
3	NaN	…	1.7	2021-12-31T22:00:00	2022-01-01T03:00:00	1641006000	
4	NaN	…	1.7	2021-12-31T23:00:00	2022-01-01T04:00:00	1641009600	

	uv	vis	weather	wind_dir	\
0	0.0	16	{'icon': 'c01n', 'description': 'Clear Sky', '…	140	
1	0.0	14	{'icon': 'a05n', 'description': 'Fog', 'code':…	110	
2	0.0	0	{'icon': 'a03n', 'description': 'Haze', 'code'…	90	
3	0.0	11	{'icon': 'a05n', 'description': 'Fog', 'code':…	110	
4	0.0	13	{'icon': 'a05n', 'description': 'Fog', 'code':…	80	

```
   wind_gust_spd wind_spd
0            1.6      1.5
1            2.2      2.1
2            1.6      1.5
3            2.8      2.6
4            2.8      2.6

[5 rows x 28 columns]
```

[11]:
```
raw_weather_df.columns
```

[11]:
```
Index(['app_temp', 'azimuth', 'clouds', 'datetime', 'dewpt', 'dhi', 'dni',
       'elev_angle', 'ghi', 'h_angle', 'pod', 'precip', 'pres',
       'revision_status', 'rh', 'slp', 'snow', 'solar_rad', 'temp',
       'timestamp_local', 'timestamp_utc', 'ts', 'uv', 'vis', 'weather',
       'wind_dir', 'wind_gust_spd', 'wind_spd'],
      dtype='object')
```

[12]:
```
raw_weather_df.dtypes
```

[12]:
```
app_temp           float64
azimuth            float64
clouds               int64
datetime            object
dewpt              float64
dhi                float64
dni                float64
elev_angle         float64
ghi                float64
h_angle            float64
pod                 object
precip             float64
pres               float64
revision_status     object
rh                   int64
slp                float64
snow               float64
solar_rad            int64
temp               float64
timestamp_local     object
timestamp_utc       object
ts                   int64
uv                 float64
vis                  int64
weather             object
wind_dir             int64
```

```
wind_gust_spd      float64
wind_spd           float64
dtype: object
```

[13]:
```
init_data.dtypes
```

[13]:
```
Carrier Code                                object
Date (MM/DD/YYYY)                           object
Flight Number                              float64
Tail Number                                 object
Origin Airport                              object
Scheduled Arrival Time                      object
Actual Arrival Time                         object
Scheduled Elapsed Time (Minutes)           float64
Actual Elapsed Time (Minutes)              float64
Arrival Delay (Minutes)                    float64
Wheels-on Time                              object
Taxi-In time (Minutes)                     float64
Delay Carrier (Minutes)                    float64
Delay Weather (Minutes)                    float64
Delay National Aviation System (Minutes)   float64
Delay Security (Minutes)                   float64
Delay Late Aircraft Arrival (Minutes)      float64
dtype: object
```

[15]:
```
# Making a copy of flight data and weather data
flight_data_copy = init_data.copy()
weather_data_copy = raw_weather_df.copy()
```

1.1.1 Flight data cleaning

[16]:
```
# Converting the Scheduled Arrival Time of flight to the nearest hour

flight_data_copy['nearest_hour'] = pd.to_datetime(flight_data_copy['Scheduled␣
 ↪Arrival Time'], format='%H:%M')
flight_data_copy['nearest_hour'] = flight_data_copy['nearest_hour'].dt.
 ↪round('H').dt.time

flight_data_copy.head()
```

[16]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Tail Number	Origin Airport	\
0	UA	1/1/2022	1282.0	N4901U	IAD	
1	UA	1/2/2022	1282.0	N4901U	IAD	
2	UA	1/3/2022	1282.0	N893UA	IAD	
3	UA	1/4/2022	1282.0	N814UA	IAD	
4	UA	1/5/2022	1282.0	N827UA	IAD	

```
  Scheduled Arrival Time Actual Arrival Time  \
0                  23:10                0:01
1                  23:10               23:27
2                  23:10               23:31
3                  23:44                1:59
4                  23:44               23:30

   Scheduled Elapsed Time (Minutes)  Actual Elapsed Time (Minutes)  \
0                              70.0                           76.0
1                              70.0                           64.0
2                              70.0                           68.0
3                              69.0                           89.0
4                              69.0                           61.0

   Arrival Delay (Minutes) Wheels-on Time  Taxi-In time (Minutes)  \
0                     51.0          23:55                    6.0
1                     17.0          23:19                    8.0
2                     21.0          23:25                    6.0
3                    135.0           1:55                    4.0
4                    -14.0          23:25                    5.0

   Delay Carrier (Minutes)  Delay Weather (Minutes)  \
0                     23.0                      0.0
1                     17.0                      0.0
2                     21.0                      0.0
3                    115.0                      0.0
4                      0.0                      0.0

   Delay National Aviation System (Minutes)  Delay Security (Minutes)  \
0                                       6.0                       0.0
1                                       0.0                       0.0
2                                       0.0                       0.0
3                                      20.0                       0.0
4                                       0.0                       0.0

   Delay Late Aircraft Arrival (Minutes) nearest_hour
0                                  22.0     23:00:00
1                                   0.0     23:00:00
2                                   0.0     23:00:00
3                                   0.0     00:00:00
4                                   0.0     00:00:00
```

[17]:
```
type(flight_data_copy['nearest_hour'][0])
```

[17]:
```
datetime.time
```

[18]:
```
# Converting the date of string datatype in (MM/DD/YYYY) format to date type in␣
↪(MM-DD-YYYY) format

flight_data_copy['date'] = pd.to_datetime(flight_data_copy['Date (MM/DD/
↪YYYY)'], format='%m/%d/%Y').dt.date
```

[19]:
```
flight_data_copy.head()
```

[19]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Tail Number	Origin Airport	\
0	UA	1/1/2022	1282.0	N4901U	IAD	
1	UA	1/2/2022	1282.0	N4901U	IAD	
2	UA	1/3/2022	1282.0	N893UA	IAD	
3	UA	1/4/2022	1282.0	N814UA	IAD	
4	UA	1/5/2022	1282.0	N827UA	IAD	

	Scheduled Arrival Time	Actual Arrival Time	\
0	23:10	0:01	
1	23:10	23:27	
2	23:10	23:31	
3	23:44	1:59	
4	23:44	23:30	

	Scheduled Elapsed Time (Minutes)	Actual Elapsed Time (Minutes)	\
0	70.0	76.0	
1	70.0	64.0	
2	70.0	68.0	
3	69.0	89.0	
4	69.0	61.0	

	Arrival Delay (Minutes)	Wheels-on Time	Taxi-In time (Minutes)	\
0	51.0	23:55	6.0	
1	17.0	23:19	8.0	
2	21.0	23:25	6.0	
3	135.0	1:55	4.0	
4	-14.0	23:25	5.0	

	Delay Carrier (Minutes)	Delay Weather (Minutes)	\
0	23.0	0.0	
1	17.0	0.0	
2	21.0	0.0	
3	115.0	0.0	
4	0.0	0.0	

	Delay National Aviation System (Minutes)	Delay Security (Minutes)	\
0	6.0	0.0	
1	0.0	0.0	
2	0.0	0.0	

```
3                                    20.0                         0.0
4                                     0.0                         0.0

   Delay Late Aircraft Arrival (Minutes) nearest_hour        date
0                                  22.0     23:00:00  2022-01-01
1                                   0.0     23:00:00  2022-01-02
2                                   0.0     23:00:00  2022-01-03
3                                   0.0     00:00:00  2022-01-04
4                                   0.0     00:00:00  2022-01-05
```

[20]:
```
type(flight_data_copy['date'][0])
```

[20]:
```
datetime.date
```

[21]:
```
# Merge the formatted date and nearest hour to datetime and converting it to␣
 ↪datetime datatype

flight_data_copy['datetime'] = flight_data_copy.apply(lambda row: dt.datetime.
 ↪combine(row['date'], row['nearest_hour']), axis=1)
```

[22]:
```
flight_data_copy.head()
```

[22]:
```
  Carrier Code Date (MM/DD/YYYY)  Flight Number Tail Number Origin Airport  \
0           UA          1/1/2022         1282.0      N4901U            IAD
1           UA          1/2/2022         1282.0      N4901U            IAD
2           UA          1/3/2022         1282.0      N893UA            IAD
3           UA          1/4/2022         1282.0      N814UA            IAD
4           UA          1/5/2022         1282.0      N827UA            IAD

  Scheduled Arrival Time Actual Arrival Time  \
0                  23:10                0:01
1                  23:10               23:27
2                  23:10               23:31
3                  23:44                1:59
4                  23:44               23:30

   Scheduled Elapsed Time (Minutes)  Actual Elapsed Time (Minutes)  \
0                              70.0                           76.0
1                              70.0                           64.0
2                              70.0                           68.0
3                              69.0                           89.0
4                              69.0                           61.0

   Arrival Delay (Minutes) Wheels-on Time  Taxi-In time (Minutes)  \
0                     51.0          23:55                     6.0
1                     17.0          23:19                     8.0
2                     21.0          23:25                     6.0
```

```
3                    135.0           1:55                     4.0
4                    -14.0          23:25                     5.0

   Delay Carrier (Minutes)  Delay Weather (Minutes)  \
0                     23.0                      0.0
1                     17.0                      0.0
2                     21.0                      0.0
3                    115.0                      0.0
4                      0.0                      0.0

   Delay National Aviation System (Minutes)  Delay Security (Minutes)  \
0                                       6.0                       0.0
1                                       0.0                       0.0
2                                       0.0                       0.0
3                                      20.0                       0.0
4                                       0.0                       0.0

   Delay Late Aircraft Arrival (Minutes) nearest_hour        date  \
0                                  22.0     23:00:00  2022-01-01
1                                   0.0     23:00:00  2022-01-02
2                                   0.0     23:00:00  2022-01-03
3                                   0.0     00:00:00  2022-01-04
4                                   0.0     00:00:00  2022-01-05

             datetime
0 2022-01-01 23:00:00
1 2022-01-02 23:00:00
2 2022-01-03 23:00:00
3 2022-01-04 00:00:00
4 2022-01-05 00:00:00
```

[32]:
```python
# Function to convert the timezones at 4 airport locations to UTC

def convert_to_utc(row):
    time_utc = None
    if row['Origin Airport'] == 'IAD':
        time_utc = row['datetime'].tz_localize(tz = 'US/Pacific').
 ↪tz_convert('UTC').tz_localize(None)
    elif row['Origin Airport'] == 'EWR':
        time_utc = row['datetime'].tz_localize(tz = 'US/Eastern').
 ↪tz_convert('UTC').tz_localize(None)
    elif row['Origin Airport'] == 'ORD':
        time_utc = row['datetime'].tz_localize(tz = 'US/Central').
 ↪tz_convert('UTC').tz_localize(None)
    elif row['Origin Airport'] == 'DEN':
        time_utc = row['datetime'].tz_localize(tz = 'US/Mountain').
 ↪tz_convert('UTC').tz_localize(None)
```

```
    return time_utc
```

[35]:
```
# Applying the convert_to_utc function on flight data

flight_data_copy['datetime_utc'] = flight_data_copy.apply(lambda row:␣
 ↪convert_to_utc(row), axis=1)
flight_data_copy.head()
```

[35]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Tail Number	Origin Airport	\
0	UA	1/1/2022	1282.0	N4901U	IAD	
1	UA	1/2/2022	1282.0	N4901U	IAD	
2	UA	1/3/2022	1282.0	N893UA	IAD	
3	UA	1/4/2022	1282.0	N814UA	IAD	
4	UA	1/5/2022	1282.0	N827UA	IAD	

	Scheduled Arrival Time	Actual Arrival Time	\
0	23:10	0:01	
1	23:10	23:27	
2	23:10	23:31	
3	23:44	1:59	
4	23:44	23:30	

	Scheduled Elapsed Time (Minutes)	Actual Elapsed Time (Minutes)	\
0	70.0	76.0	
1	70.0	64.0	
2	70.0	68.0	
3	69.0	89.0	
4	69.0	61.0	

	Arrival Delay (Minutes)	…	Taxi-In time (Minutes)	\
0	51.0	…	6.0	
1	17.0	…	8.0	
2	21.0	…	6.0	
3	135.0	…	4.0	
4	-14.0	…	5.0	

	Delay Carrier (Minutes)	Delay Weather (Minutes)	\
0	23.0	0.0	
1	17.0	0.0	
2	21.0	0.0	
3	115.0	0.0	
4	0.0	0.0	

	Delay National Aviation System (Minutes)	Delay Security (Minutes)	\
0	6.0	0.0	
1	0.0	0.0	

```
2                                        0.0                        0.0
3                                       20.0                        0.0
4                                        0.0                        0.0

   Delay Late Aircraft Arrival (Minutes)  nearest_hour        date  \
0                                   22.0      23:00:00  2022-01-01
1                                    0.0      23:00:00  2022-01-02
2                                    0.0      23:00:00  2022-01-03
3                                    0.0      00:00:00  2022-01-04
4                                    0.0      00:00:00  2022-01-05

             datetime        datetime_utc
0 2022-01-01 23:00:00 2022-01-02 07:00:00
1 2022-01-02 23:00:00 2022-01-03 07:00:00
2 2022-01-03 23:00:00 2022-01-04 07:00:00
3 2022-01-04 00:00:00 2022-01-04 08:00:00
4 2022-01-05 00:00:00 2022-01-05 08:00:00

[5 rows x 21 columns]
```

1.1.2 Weather data cleaning

[38]:
```
# Weather data

weather_data_copy.columns
```

[38]:
```
Index(['app_temp', 'azimuth', 'clouds', 'datetime', 'dewpt', 'dhi', 'dni',
       'elev_angle', 'ghi', 'h_angle', 'pod', 'precip', 'pres',
       'revision_status', 'rh', 'slp', 'snow', 'solar_rad', 'temp',
       'timestamp_local', 'timestamp_utc', 'ts', 'uv', 'vis', 'weather',
       'wind_dir', 'wind_gust_spd', 'wind_spd'],
      dtype='object')
```

[42]:
```
weather_data_copy['datetime_utc'] = pd.
 ↪to_datetime(weather_data_copy['timestamp_utc'])
```

[43]:
```
weather_data_copy.head()
```

[43]:
```
   app_temp  azimuth  clouds       datetime  dewpt  dhi  dni  elev_angle  ghi  \
0       3.9   260.93       0  2022-01-01:00    4.3  0.0  0.0      -24.85  0.0
1       1.3   270.74      25  2022-01-01:01    3.3  0.0  0.0      -35.75  0.0
2       0.8   282.10      25  2022-01-01:02    1.6  0.0  0.0      -46.62  0.0
3      -1.1   297.04      25  2022-01-01:03    1.7  0.0  0.0      -56.93  0.0
4      -1.1   319.76      25  2022-01-01:04    1.7  0.0  0.0      -65.55  0.0

   h_angle  …      timestamp_local        timestamp_utc          ts   uv  \
0      NaN  …  2021-12-31T19:00:00  2022-01-01T00:00:00  1640995200  0.0
```

```
1      NaN  …  2021-12-31T20:00:00  2022-01-01T01:00:00  1640998800  0.0
2      NaN  …  2021-12-31T21:00:00  2022-01-01T02:00:00  1641002400  0.0
3      NaN  …  2021-12-31T22:00:00  2022-01-01T03:00:00  1641006000  0.0
4      NaN  …  2021-12-31T23:00:00  2022-01-01T04:00:00  1641009600  0.0

   vis                                              weather  wind_dir  \
0   16  {'icon': 'c01n', 'description': 'Clear Sky', '…       140
1   14  {'icon': 'a05n', 'description': 'Fog', 'code':…       110
2    0  {'icon': 'a03n', 'description': 'Haze', 'code'…        90
3   11  {'icon': 'a05n', 'description': 'Fog', 'code':…       110
4   13  {'icon': 'a05n', 'description': 'Fog', 'code':…        80

   wind_gust_spd  wind_spd        datetime_utc
0            1.6       1.5 2022-01-01 00:00:00
1            2.2       2.1 2022-01-01 01:00:00
2            1.6       1.5 2022-01-01 02:00:00
3            2.8       2.6 2022-01-01 03:00:00
4            2.8       2.6 2022-01-01 04:00:00

[5 rows x 29 columns]
```

[44]:
```
flight_weather_df = pd.merge(flight_data_copy, weather_data_copy,␣
↪on='datetime_utc')
```

[46]:
```
flight_weather_df.iloc[0]
```

[46]:
```
Carrier Code
UA
Date (MM/DD/YYYY)
1/1/2022
Flight Number
1282.0
Tail Number
N4901U
Origin Airport
IAD
Scheduled Arrival Time
23:10
Actual Arrival Time
0:01
Scheduled Elapsed Time (Minutes)
70.0
Actual Elapsed Time (Minutes)
76.0
Arrival Delay (Minutes)
51.0
Wheels-on Time
```

```
23:55
Taxi-In time (Minutes)
6.0
Delay Carrier (Minutes)
23.0
Delay Weather (Minutes)
0.0
Delay National Aviation System (Minutes)
6.0
Delay Security (Minutes)
0.0
Delay Late Aircraft Arrival (Minutes)
22.0
nearest_hour
23:00:00
date
2022-01-01
datetime_x
2022-01-01 23:00:00
datetime_utc
2022-01-02 07:00:00
app_temp
-3.0
azimuth
57.89
clouds
100
datetime_y
2022-01-02:07
dewpt
0.5
dhi
0.0
dni
0.0
elev_angle
-59.34
ghi
0.0
h_angle
NaN
pod
n
precip
0.25
pres
990.3
```

```
revision_status
final
rh
96
slp
1005.2
snow
0.5
solar_rad
0
temp
1.1
timestamp_local
2022-01-02T02:00:00
timestamp_utc
2022-01-02T07:00:00
ts
1641106800
uv
0.0
vis
11
weather                                          {'icon': 's04n', 'description': 'Mix
snow/rain…
wind_dir
300
wind_gust_spd
4.5
wind_spd
4.1
Name: 0, dtype: object
```

[55]:
```python
# Categorize the Arrival Delay Time to 4 categories
def get_status(row):
    status = None
    if row['Arrival Delay (Minutes)'] < -10:
        status = 'early'
    elif row['Arrival Delay (Minutes)'] < 10:
        status = 'on-time'
    elif row['Arrival Delay (Minutes)'] < 30:
        status = 'late'
    else:
        status = 'severely-late'

    return status
```

[56]:
```
# Applying the get_status method

flight_weather_df['status'] = flight_weather_df.apply(lambda row:␣
↪get_status(row), axis=1)
```

[224]:
```
flight_weather_df.head()
```

[224]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Tail Number	Origin Airport
0	UA	1/1/2022	1,282.00	N4901U	IAD
1	UA	1/2/2022	1,282.00	N4901U	IAD
2	UA	1/3/2022	1,282.00	N893UA	IAD
3	UA	1/4/2022	1,282.00	N814UA	IAD
4	UA	1/5/2022	1,282.00	N827UA	IAD

	Scheduled Arrival Time	Actual Arrival Time
0	23:10	0:01
1	23:10	23:27
2	23:10	23:31
3	23:44	1:59
4	23:44	23:30

	Scheduled Elapsed Time (Minutes)	Actual Elapsed Time (Minutes)
0	70.00	76.00
1	70.00	64.00
2	70.00	68.00
3	69.00	89.00
4	69.00	61.00

	Arrival Delay (Minutes)	…	uv	vis	weather	wind_dir	wind_gust_spd
0	51.00	…	0.00	11	2	300	4.50
1	17.00	…	0.00	16	1	320	2.20
2	21.00	…	0.00	16	1	150	1.60
3	135.00	…	0.00	16	1	120	2.20
4	-14.00	…	0.00	16	1	140	4.50

	wind_spd	status	week	month	day
0	4.10	severely-late	52	1	2
1	2.10	late	1	1	3
2	1.50	late	1	1	4
3	2.10	severely-late	1	1	4
4	4.10	early	1	1	5

[5 rows x 53 columns]

[65]:
```
set(flight_weather_df['weather'])
```

[65]:
```
{"{'icon': 'a03d', 'description': 'Haze', 'code': 721}",
 "{'icon': 'a03n', 'description': 'Haze', 'code': 721}",
 "{'icon': 'a05d', 'description': 'Fog', 'code': 741}",
 "{'icon': 'a05n', 'description': 'Fog', 'code': 741}",
 "{'icon': 'c01d', 'description': 'Clear Sky', 'code': 800}",
 "{'icon': 'c01n', 'description': 'Clear Sky', 'code': 800}",
 "{'icon': 'c02d', 'description': 'Few clouds', 'code': 801}",
 "{'icon': 'c02d', 'description': 'Scattered clouds', 'code': 802}",
 "{'icon': 'c02n', 'description': 'Few clouds', 'code': 801}",
 "{'icon': 'c02n', 'description': 'Scattered clouds', 'code': 802}",
 "{'icon': 'c03d', 'description': 'Broken clouds', 'code': 803}",
 "{'icon': 'c03n', 'description': 'Broken clouds', 'code': 803}",
 "{'icon': 'c04d', 'description': 'Overcast clouds', 'code': 804}",
 "{'icon': 'c04n', 'description': 'Overcast clouds', 'code': 804}",
 "{'icon': 'f01n', 'description': 'Freezing rain', 'code': 511}",
 "{'icon': 'r01d', 'description': 'Light rain', 'code': 500}",
 "{'icon': 'r01n', 'description': 'Light rain', 'code': 500}",
 "{'icon': 'r02n', 'description': 'Moderate rain', 'code': 501}",
 "{'icon': 'r03d', 'description': 'Heavy rain', 'code': 502}",
 "{'icon': 'r03n', 'description': 'Heavy rain', 'code': 502}",
 "{'icon': 's01d', 'description': 'Light snow', 'code': 600}",
 "{'icon': 's01n', 'description': 'Light snow', 'code': 600}",
 "{'icon': 's02n', 'description': 'Snow', 'code': 601}",
 "{'icon': 's04d', 'description': 'Mix snow/rain', 'code': 610}",
 "{'icon': 's04n', 'description': 'Mix snow/rain', 'code': 610}",
 "{'icon': 's05n', 'description': 'Heavy sleet', 'code': 612}",
 "{'icon': 's06n', 'description': 'Flurries', 'code': 623}",
 "{'icon': 't03n', 'description': 'Thunderstorm with heavy rain', 'code': 202}"}
```

[70]:
```
# Extract the weather code from the weather dictionary values

def extract_weather_code(str_dict):
    dict_obj = ast.literal_eval(str_dict)
    return dict_obj['code']

flight_weather_df['weather'] = flight_weather_df['weather'].
 ↪apply(extract_weather_code)
```

[72]:
```
set(flight_weather_df['weather'])
```

[72]:
```
{202,
 500,
 501,
 502,
 511,
 600,
 601,
```

```
610,
612,
623,
721,
741,
800,
801,
802,
803,
804}
```

[73]:
```
# Replace weather values
flight_weather_df['weather'].replace([202, 500, 501, 502, 511, 600, 601, 610,␣
↪612, 623, 721, 741, 800, 801, 802, 803, 804],
                                     [3, 2, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2, 0, 1,␣
↪1, 1, 1], inplace=True)
```

[81]:
```
set(flight_weather_df['temp'])
```

[81]:
```
{-20.0,
 -19.4,
 -18.3,
 -17.2,
 -14.4,
 -13.9,
 -13.3,
 -12.8,
 -11.7,
 -10.6,
 -10.0,
 -9.4,
 -8.9,
 -8.3,
 -7.8,
 -7.2,
 -6.7,
 -6.1,
 -5.6,
 -5.0,
 -4.4,
 -3.9,
 -3.3,
 -2.8,
 -2.2,
 -1.7,
 -1.1,
 -0.6,
```

```
0.0,
0.3,
0.6,
1.1,
1.7,
2.2,
2.8,
3.3,
3.9,
4.4,
5.0,
5.6,
6.1,
6.4,
6.7,
7.2,
7.5,
7.8,
8.1,
8.3,
8.9,
9.4,
10.0,
10.3,
10.6,
11.1,
11.7,
11.9,
12.2,
12.3,
12.5,
12.8,
13.1,
13.3,
13.4,
13.6,
13.9,
14.2,
14.4,
14.7,
14.8,
15.0,
15.3,
15.6,
15.9,
16.1,
16.4,
```

```
16.6,
16.7,
16.9,
17.2,
17.3,
17.5,
17.8,
18.3,
18.6,
18.9,
19.1,
19.4,
19.7,
20.0,
20.3,
20.5,
20.6,
20.9,
21.1,
21.6,
21.7,
21.9,
22.0,
22.2,
22.5,
22.8,
23.0,
23.3,
23.4,
23.9,
24.0,
24.4,
24.5,
24.7,
25.0,
25.3,
25.6,
25.9,
26.1,
26.6,
26.7,
27.2,
27.3,
27.5,
27.8,
28.3,
28.9,
```

```
 29.1,
 29.4,
 30.0,
 30.6,
 31.1,
 31.7,
 32.2,
 32.8,
 33.3,
 33.9,
 34.4,
 35.0}
```

[82]:
```
# Adding the week, month and day of the year to the dataframe

flight_weather_df['week'] = flight_weather_df['datetime_utc'].dt.week
flight_weather_df['month'] = flight_weather_df['datetime_utc'].dt.month
flight_weather_df['day'] = flight_weather_df['datetime_utc'].dt.day
```

```
C:\Users\vidhe\AppData\Local\Temp\ipykernel_14620\2148686766.py:1:
FutureWarning: Series.dt.weekofyear and Series.dt.week have been deprecated.
Please use Series.dt.isocalendar().week instead.
  flight_weather_df['week'] = flight_weather_df['datetime_utc'].dt.week
```

[92]:
```
flight_weather_df.columns
```

[92]:
```
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)', 'nearest_hour', 'date',
       'datetime_x', 'datetime_utc', 'app_temp', 'azimuth', 'clouds',
       'datetime_y', 'dewpt', 'dhi', 'dni', 'elev_angle', 'ghi', 'h_angle',
       'pod', 'precip', 'pres', 'revision_status', 'rh', 'slp', 'snow',
       'solar_rad', 'temp', 'timestamp_local', 'timestamp_utc', 'ts', 'uv',
       'vis', 'weather', 'wind_dir', 'wind_gust_spd', 'wind_spd', 'status',
       'week', 'month', 'day'],
      dtype='object')
```

[93]:
```
# (flight_weather_df.loc[flight_weather_df['weather'] == 2]).
  ↪to_csv('late_flights_1.csv', index = False)
flight_weather_df.iloc[0]
```

[93]:
```
Carrier Code                                  UA
Date (MM/DD/YYYY)                       1/1/2022
```

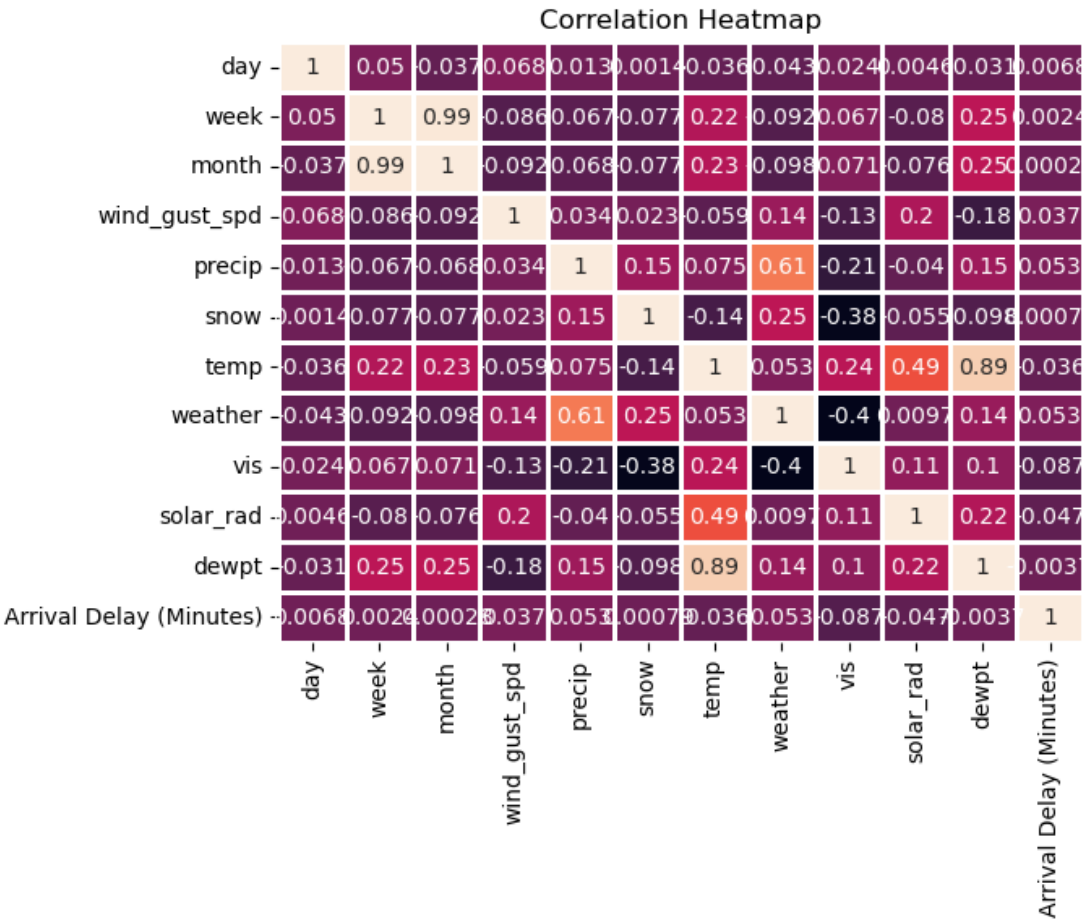
Flight Number	1282.0
Tail Number	N4901U
Origin Airport	IAD
Scheduled Arrival Time	23:10
Actual Arrival Time	0:01
Scheduled Elapsed Time (Minutes)	70.0
Actual Elapsed Time (Minutes)	76.0
Arrival Delay (Minutes)	51.0
Wheels-on Time	23:55
Taxi-In time (Minutes)	6.0
Delay Carrier (Minutes)	23.0
Delay Weather (Minutes)	0.0
Delay National Aviation System (Minutes)	6.0
Delay Security (Minutes)	0.0
Delay Late Aircraft Arrival (Minutes)	22.0
nearest_hour	23:00:00
date	2022-01-01
datetime_x	2022-01-01 23:00:00
datetime_utc	2022-01-02 07:00:00
app_temp	-3.0
azimuth	57.89
clouds	100
datetime_y	2022-01-02:07
dewpt	0.5
dhi	0.0
dni	0.0
elev_angle	-59.34
ghi	0.0
h_angle	NaN
pod	n
precip	0.25
pres	990.3
revision_status	final
rh	96
slp	1005.2
snow	0.5
solar_rad	0
temp	1.1
timestamp_local	2022-01-02T02:00:00
timestamp_utc	2022-01-02T07:00:00
ts	1641106800
uv	0.0
vis	11
weather	2
wind_dir	300
wind_gust_spd	4.5
wind_spd	4.1

```
status                                          severely-late
week                                                       52
month                                                       1
day                                                         2
Name: 0, dtype: object
```

[107]:
```
# Creating a subset data for correlation matrix

corr_subset_data = flight_weather_df[['day', 'week', 'month', 'wind_gust_spd',␣
 ↪'precip', 'snow', 'temp', 'weather', 'vis', 'solar_rad', 'dewpt', 'Arrival␣
 ↪Delay (Minutes)']]
```

[109]:
```
# Correlation matrix

correl = corr_subset_data.corr()
ax1 = sns.heatmap(correl, cbar=0, linewidths=1, annot=True)

plt.title("Correlation Heatmap")
plt.show()
```

[109]:
```
Text(0.5, 1.0, 'Correlation Heatmap')
```



1.2 Model Creation and Training

[ ]:
```
# Creating subset data with necessary columns of the raw flight and weather data

subset_data = flight_weather_df[['day', 'week', 'month', 'wind_gust_spd',␣
↪'precip', 'snow', 'temp', 'weather', 'vis', 'solar_rad', 'dewpt', 'status']]
```

[95]:
```
# Splitting the dataset into 80% training and 20% testing data.

X_train, X_test, y_train, y_test = train_test_split(subset_data.drop(columns =␣
↪['status']), subset_data['status'], test_size=0.2, stratify =␣
↪subset_data['status'], random_state=42)
```

[96]:
```
X_train, X_test, y_train, y_test
```

[96]:
```
(     day  week  month  wind_gust_spd  precip  snow  temp  weather  vis  \
 714   18    37      9            2.4    0.00   0.0  20.9        1   16
 906   16    46     11            5.1    0.00   0.0   0.0        1    3
```

```
520   25    30      7            5.1    9.00   0.0  23.3        3   11
835   20    42     10            5.8    0.00   0.0   5.6        1   16
56    17     7      2           13.0    0.00   0.0  10.6        1   16
..   …    …    …             …      …    …   …      …  …
506   21    29      7            7.8    3.25   0.0  28.3        2   16
697   12    37      9            3.3    0.00   0.0  19.4        1   16
837   21    42     10            8.0    0.00   0.0  16.7        1   16
728   23    38      9            3.9    0.25   0.0  10.6        1   16
834   21    42     10            2.8    0.00   0.0   7.2        1   16

     solar_rad  dewpt
714          0   14.3
906          0   -2.1
520          0   20.4
835          0    1.0
56           0    0.3
..         …    …
506          0   17.5
697          0   17.7
837        146   -1.9
728          0    7.7
834          0   -3.3

[813 rows x 11 columns],
     day  week  month  wind_gust_spd  precip  snow  temp  weather  vis  \
410   26    25      6            2.2    1.25   0.0  23.3        2   14
472   13    28      7            8.0    1.50   0.0  21.7        2    5
59    18     7      2           10.0    1.00   7.0   0.6        2    6
485   16    28      7            5.1    0.00   0.0  31.1        1   16
139   18    11      3            2.1    0.00   0.0  12.8        1   16
..   …    …    …             …      …    …   …      …  …
60    19     7      2            3.3    0.00   0.0  -8.3        1   16
38    10     6      2            2.8    0.00   0.0   6.1        1   16
736   26    39      9            2.8    0.00   0.0  15.0        1   16
965   12    50     12            2.5    0.00   0.0   0.6        1   10
536   30    30      7            3.6    0.00   0.0  20.0        1   16

     solar_rad  dewpt
410        155   21.6
472         94   17.7
59           0   -1.2
485        523   11.5
139          0    6.6
..         …    …
60           0  -13.9
38           0   -2.2
736          0   12.7
```

```
965          0   -0.7
536          0   13.5

[204 rows x 11 columns],
714            early
906            early
520    severely-late
835          on-time
56           on-time
           …
506          on-time
697          on-time
837          on-time
728    severely-late
834          on-time
Name: status, Length: 813, dtype: object,
410            early
472             late
59     severely-late
485          on-time
139             late
           …
60           on-time
38             early
736    severely-late
965    severely-late
536          on-time
Name: status, Length: 204, dtype: object)
```

[100]:
```
# Scaling
scaling = True
if scaling:
    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)
```

[101]:
```
# Logistic Regression Model

model = LogisticRegression(fit_intercept = True, solver='lbfgs', multi_class =␣
 ↪'ovr', penalty = 'none')

model.fit(X_train, y_train)
```

```
C:\Users\vidhe\anaconda3\envs\cis600\lib\site-
```

```
packages\sklearn\linear_model\_logistic.py:1173: FutureWarning:
`penalty='none'`has been deprecated in 1.2 and will be removed in 1.4. To keep
the past behaviour, set `penalty=None`.
  warnings.warn(
```

[101]:
```
LogisticRegression(multi_class='ovr', penalty='none')
```

[102]:
```
# The following gives the mean accuracy on the given data and labels
score = model.score(X_train, y_train)

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

# This is the coefficient Beta_0
intercept = model.intercept_

print("Model score: ", score)
print("Coefficients: ", coef)
print("Intercept: ", intercept)
```

```
Model score:  0.4661746617466175
Coefficients:  [[-3.05235200e-01  3.18279204e+00 -3.06002283e+00 -6.07547578e-02
  -1.00758663e-01 -1.11981419e+02  5.93879352e-01 -2.07863752e-01
  -1.28919894e-01 -2.37279388e-01 -6.18735590e-01]
 [ 1.57253320e-01 -1.00146548e+00  9.54744386e-01  1.63469309e-03
   8.60220310e-02 -3.78303822e-03 -4.62330119e-01 -9.44363117e-02
   7.11581451e-02  2.56414672e-01  4.25378922e-01]
 [ 1.42026795e-01 -1.56362748e+00  1.53699002e+00 -4.26152105e-02
  -2.07087952e-01  2.55286654e-01 -9.35881500e-02  3.20625598e-01
   4.07857784e-02  1.87252779e-01  1.27111791e-01]
 [-1.71607250e-01  1.43774611e+00 -1.57248505e+00  1.97722464e-01
   1.92088383e-01 -4.13351455e+00 -4.99396494e-01 -9.96475430e-02
   4.75987447e-02 -3.55199914e-01  5.39017656e-01]]
Intercept:  [-12.12893019  -1.97771574  -0.22767257  -2.33933417]
```

[103]:
```
# Percentage of correct predictions

test_output = pd.DataFrame(model.predict(X_test), index = X_test.index, columns␣
 ↪= ['pred_Status'])
test_output = test_output.merge(y_test, left_index = True, right_index = True)
print('Percentage of correct predictions is ')
print(round(model.score(X_test, y_test), 2))
```

```
Percentage of correct predictions is
0.42
```

[112]:
```
# Gradient Boosting

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)
gb.score(X_train, y_train)

feat_imp = pd.Series(gb.feature_importances_, X_train.columns.values).
 ↪sort_values(ascending=False)

feat_imp_table = pd.DataFrame(feat_imp)
feat_imp_table = feat_imp_table.reset_index()
feat_imp_table.columns = ['Features', 'Values']
feat_imp.plot(kind='bar', title='Feature Importances')
plt.ylabel('Feature Importance Score')
plt.figure(figsize=[40,20], dpi = 50)
feat_imp.head(12)

test_output = pd.DataFrame(gb.predict(X_test), index = X_test.index, columns =␣
 ↪['pred_Y'])

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

[112]:
```
0.8757687576875769
```

[112]:
```
<Axes: title={'center': 'Feature Importances'}>
```

[112]:
```
Text(0, 0.5, 'Feature Importance Score')
```

[112]:
```
<Figure size 2000x1000 with 0 Axes>
```

[112]:
```
dewpt           0.21
wind_gust_spd   0.17
week            0.15
temp            0.14
day             0.14
solar_rad       0.10
precip          0.02
vis             0.02
weather         0.02
month           0.02
snow            0.01
```

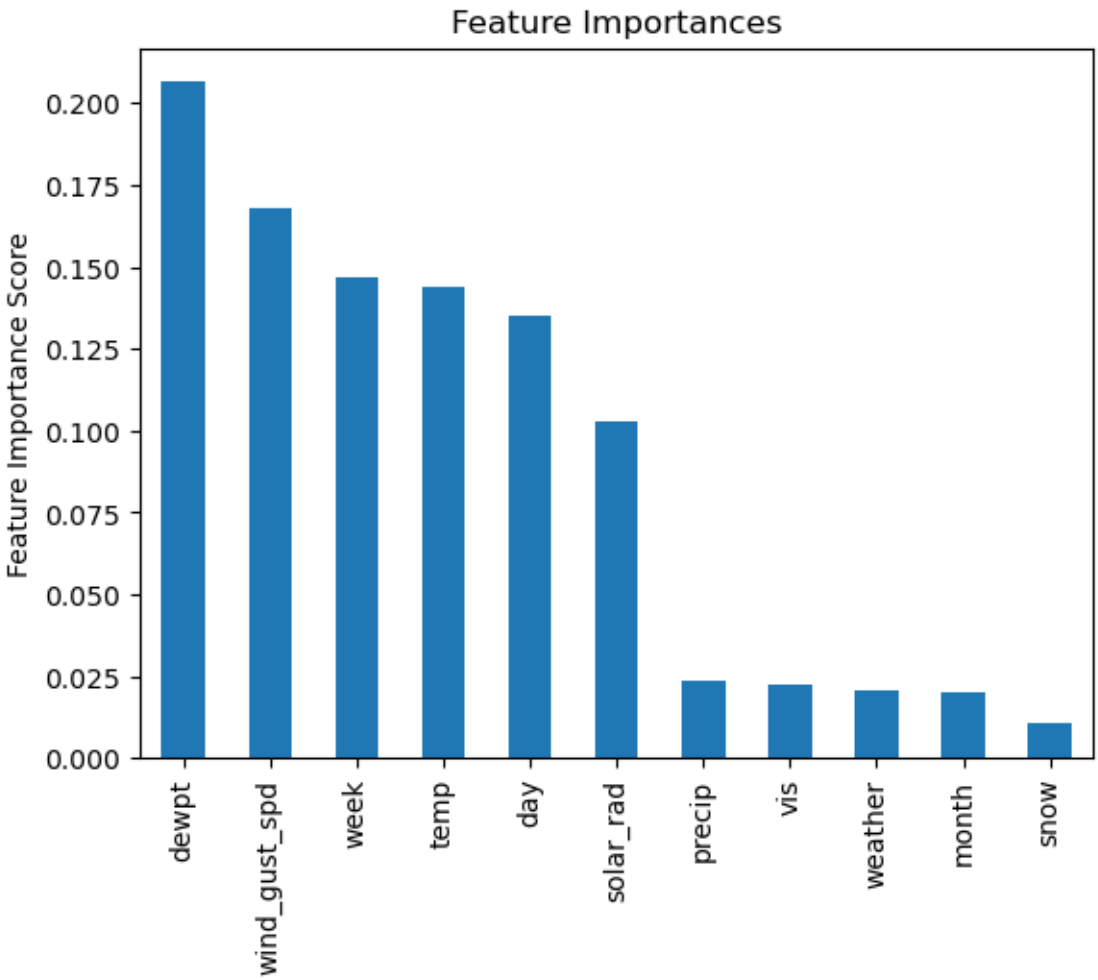
```
dtype: float64
```

[112]:
```
      pred_Y
410  on-time
472  on-time
59      late
485    early
139  on-time
```

[112]:
```
      pred_Y         status
410  on-time          early
472  on-time           late
59      late  severely-late
485    early        on-time
139  on-time           late
```

```
Fraction of correct classification
```

[112]: 0.43137254901960786



```
<Figure size 2000x1000 with 0 Axes>
```

[113]:
```
# Random Forests

rf = RandomForestClassifier(random_state=50, min_samples_leaf = 6, max_features␣
 ↪= "sqrt", n_estimators = 100)

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

# rf.feature_importances_
feat_imp = pd.Series(rf.feature_importances_, X_train.columns.values).
 ↪sort_values(ascending=False)

feat_imp_table = pd.DataFrame(feat_imp)
feat_imp_table = feat_imp_table.reset_index()
feat_imp_table.columns = ['Features', 'Values']
feat_imp.plot(kind='bar', title='Feature Importances')
plt.ylabel('Feature Importance Score')
plt.figure(figsize=[40,20], dpi = 50)
feat_imp.head(12)

test_output = pd.DataFrame(rf.predict(X_test), index = X_test.index, columns =␣
 ↪['pred_Y'])

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

[113]:
```
0.6371463714637147
```

[113]:
```
<Axes: title={'center': 'Feature Importances'}>
```

[113]:
```
Text(0, 0.5, 'Feature Importance Score')
```

[113]:
```
<Figure size 2000x1000 with 0 Axes>
```

[113]:
```
dewpt           0.18
temp            0.16
week            0.15
wind_gust_spd   0.14
day             0.14
solar_rad       0.09
```

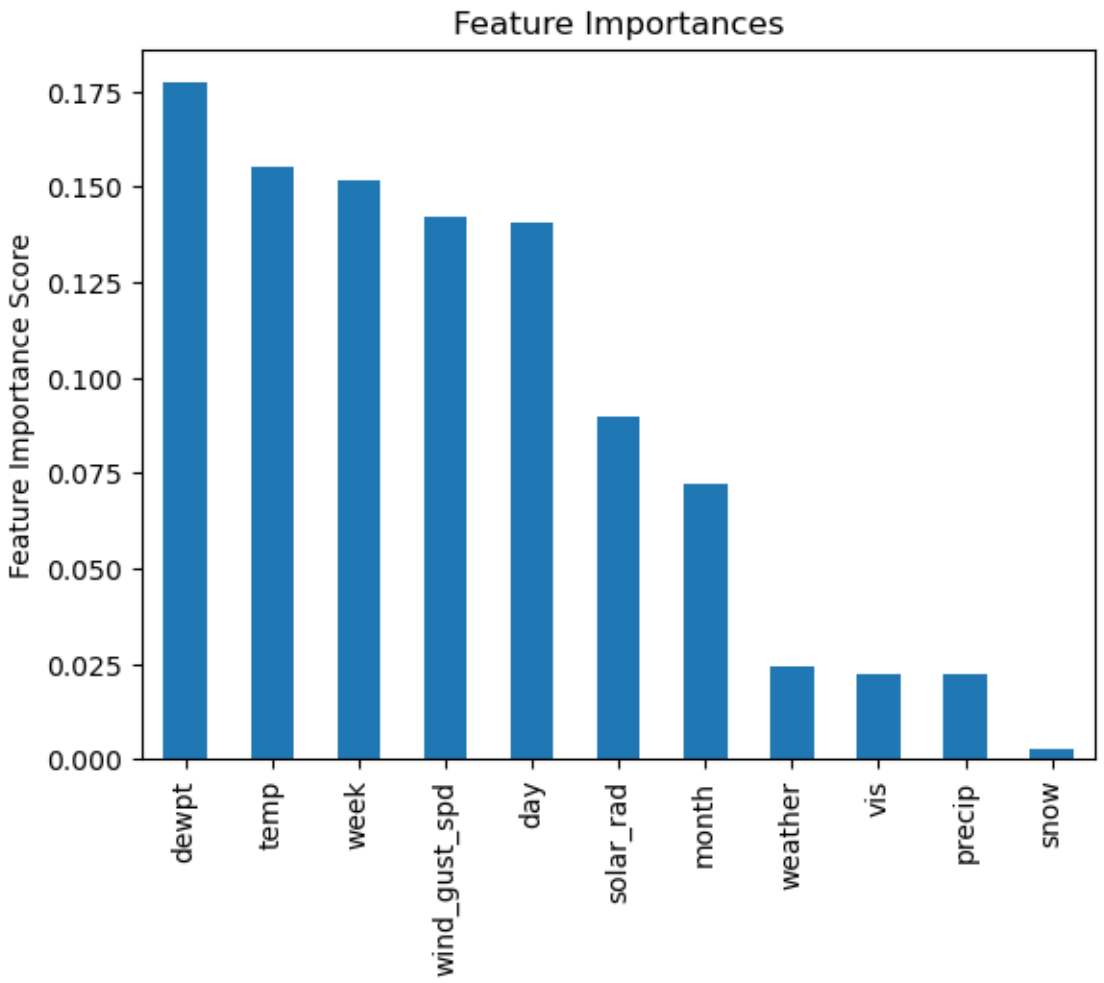
```
month           0.07
weather         0.02
vis             0.02
precip          0.02
snow            0.00
dtype: float64
```

[113]:
```
      pred_Y
410  on-time
472  on-time
59   on-time
485  on-time
139  on-time
```

[113]:
```
      pred_Y         status
410  on-time          early
472  on-time           late
59   on-time  severely-late
485  on-time        on-time
139  on-time           late
```

```
Fraction of correct classification
```

[113]: 0.47549019607843135



<Figure size 2000x1000 with 0 Axes>

[135]:
```
# Train an XGBoost model

le = LabelEncoder()
y_train = le.fit_transform(y_train)

params = {'objective': 'multi:softmax', 'num_class': 4}
dtrain = xgb.DMatrix(X_train, label=y_train)
model = xgb.train(params, dtrain)



# Make predictions on the test data
y_test = le.fit_transform(y_test)
```

```
dtest = xgb.DMatrix(X_test)
y_pred = model.predict(dtest)

# Print the classification report
target_names = le.inverse_transform(range(0, 4))

print(classification_report(y_test, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

       early       0.41      0.25      0.31        60
        late       0.22      0.08      0.12        25
     on-time       0.46      0.74      0.57        91
severely-late       0.25      0.11      0.15        28

    accuracy                           0.43       204
   macro avg       0.33      0.29      0.29       204
weighted avg       0.39      0.43      0.38       204
```

1.3 Predictions

[143]:
```
# Fetching predictions to be made to dataframe

data_1 = pd.read_csv('datasets/project csv(Apr 12-15).csv')
data_1.head()
```

[143]:

	Date	Day	Origin	Airport	Flight Number	Arrival Time	\
0	4/12/2023	Wednesday	ORD		UA 3839	10:00 AM	
1	4/12/2023	Wednesday	ORD		UA 3524	4:52 PM	
2	4/12/2023	Wednesday	ORD		UA 538	9:34 PM	
3	4/13/2023	Thursday	ORD		UA 3839	10:00 AM	
4	4/13/2023	Thursday	ORD		UA 3524	4:50 PM	

	Status (Early, On-time, Late, Severly Late)
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN

[236]:
```
data_2 = pd.read_csv('datasets/project csv(Apr 21-24).csv')
data_2.head()
```

[236]:

	Date	Day	Origin	Airport	Flight Number	Arrival Time	\
0	4/21/2023	Friday	ORD		UA 3839	10:00 AM	
1	4/21/2023	Friday	ORD		UA 3524	4:50 PM	

```
2  4/21/2023    Friday            ORD        UA 538      9:34 PM
3  4/22/2023  Saturday            ORD       UA 3839     10:00 AM
4  4/22/2023  Saturday            ORD       UA 3524      4:50 PM

   Status (Early, On-time, Late, Severly Late)
0                                          NaN
1                                          NaN
2                                          NaN
3                                          NaN
4                                          NaN
```

1.3.1 Apr 12 - 15

[145]:
```
# Dropping the status column

data_1 = data_1.drop(columns=['Status (Early, On-time, Late, Severly Late)'])
data_1 = data_1.dropna()
```

[148]:
```
data_1
```

[148]:

	Date	Day	Origin Airport	Flight Number	Arrival Time
0	4/12/2023	Wednesday	ORD	UA 3839	10:00 AM
1	4/12/2023	Wednesday	ORD	UA 3524	4:52 PM
2	4/12/2023	Wednesday	ORD	UA 538	9:34 PM
3	4/13/2023	Thursday	ORD	UA 3839	10:00 AM
4	4/13/2023	Thursday	ORD	UA 3524	4:50 PM
5	4/13/2023	Thursday	ORD	UA 538	9:34 PM
6	4/14/2023	Friday	ORD	UA 3839	10:00 AM
7	4/14/2023	Friday	ORD	UA 3524	4:50 PM
8	4/14/2023	Friday	ORD	UA 538	9:34 PM
9	4/15/2023	Saturday	ORD	UA 3839	10:00 AM
10	4/15/2023	Saturday	ORD	UA 3524	4:50 PM
11	4/15/2023	Saturday	ORD	UA 538	9:34 PM
12	4/12/2023	Wednesday	DEN	UA 604	3:12 PM
13	4/13/2023	Thursday	DEN	UA 604	3:12 PM
14	4/14/2023	Friday	DEN	UA 604	3:12 PM
15	4/15/2023	Saturday	DEN	UA 604	3:12 PM
16	4/12/2023	Wednesday	EWR	UA 4189	10:46 AM
17	4/12/2023	Wednesday	EWR	UA 1412	11:42 PM
18	4/13/2023	Thursday	EWR	UA 4189	10:46 AM
19	4/13/2023	Thursday	EWR	UA 1412	11:42 PM
20	4/14/2023	Friday	EWR	UA 4189	10:46 AM
21	4/14/2023	Friday	EWR	UA 1412	11:42 PM
22	4/15/2023	Saturday	EWR	UA 4189	10:46 AM
23	4/15/2023	Saturday	EWR	UA 1412	11:17 PM
24	4/12/2023	Wednesday	IAD	UA 4490	1:57 PM
25	4/12/2023	Wednesday	IAD	UA 4165	6:59 PM

26	4/13/2023	Thursday	IAD	UA 4490	1:57 PM
27	4/13/2023	Thursday	IAD	UA 4165	6:59 PM
28	4/14/2023	Friday	IAD	UA 4490	1:57 PM
29	4/14/2023	Friday	IAD	UA 4165	6:59 PM
30	4/15/2023	Saturday	IAD	UA 3805	1:58 PM
31	4/15/2023	Saturday	IAD	UA 4165	6:59 PM

[161]:
```
# Converting the arrival time to time at nearest hour

data_1['nearest_hour'] = pd.to_datetime(data_1['Arrival Time'], format='%I:%M␣
 ↪%p')
data_1['nearest_hour'] = data_1['nearest_hour'].dt.strftime('%H:%M')
data_1['nearest_hour'] = pd.to_datetime(data_1['nearest_hour']).dt.round('H').
 ↪dt.time
```

```
C:\Users\vidhe\AppData\Local\Temp\ipykernel_14620\2428816250.py:1:
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

C:\Users\vidhe\AppData\Local\Temp\ipykernel_14620\2428816250.py: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

C:\Users\vidhe\AppData\Local\Temp\ipykernel_14620\2428816250.py: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
```

[162]:
```
data_1
```

[162]:

	Date	Day	Origin Airport	Flight Number	Arrival Time	nearest_hour
0	4/12/2023	Wednesday	ORD	UA 3839	10:00 AM	10:00:00
1	4/12/2023	Wednesday	ORD	UA 3524	4:52 PM	17:00:00
2	4/12/2023	Wednesday	ORD	UA 538	9:34 PM	22:00:00
3	4/13/2023	Thursday	ORD	UA 3839	10:00 AM	10:00:00
4	4/13/2023	Thursday	ORD	UA 3524	4:50 PM	17:00:00
5	4/13/2023	Thursday	ORD	UA 538	9:34 PM	22:00:00
6	4/14/2023	Friday	ORD	UA 3839	10:00 AM	10:00:00
7	4/14/2023	Friday	ORD	UA 3524	4:50 PM	17:00:00
8	4/14/2023	Friday	ORD	UA 538	9:34 PM	22:00:00
9	4/15/2023	Saturday	ORD	UA 3839	10:00 AM	10:00:00
10	4/15/2023	Saturday	ORD	UA 3524	4:50 PM	17:00:00
11	4/15/2023	Saturday	ORD	UA 538	9:34 PM	22:00:00
12	4/12/2023	Wednesday	DEN	UA 604	3:12 PM	
13	4/13/2023	Thursday	DEN	UA 604	3:12 PM	
14	4/14/2023	Friday	DEN	UA 604	3:12 PM	
15	4/15/2023	Saturday	DEN	UA 604	3:12 PM	
16	4/12/2023	Wednesday	EWR	UA 4189	10:46 AM	
17	4/12/2023	Wednesday	EWR	UA 1412	11:42 PM	
18	4/13/2023	Thursday	EWR	UA 4189	10:46 AM	
19	4/13/2023	Thursday	EWR	UA 1412	11:42 PM	
20	4/14/2023	Friday	EWR	UA 4189	10:46 AM	
21	4/14/2023	Friday	EWR	UA 1412	11:42 PM	
22	4/15/2023	Saturday	EWR	UA 4189	10:46 AM	
23	4/15/2023	Saturday	EWR	UA 1412	11:17 PM	
24	4/12/2023	Wednesday	IAD	UA 4490	1:57 PM	
25	4/12/2023	Wednesday	IAD	UA 4165	6:59 PM	
26	4/13/2023	Thursday	IAD	UA 4490	1:57 PM	
27	4/13/2023	Thursday	IAD	UA 4165	6:59 PM	
28	4/14/2023	Friday	IAD	UA 4490	1:57 PM	
29	4/14/2023	Friday	IAD	UA 4165	6:59 PM	
30	4/15/2023	Saturday	IAD	UA 3805	1:58 PM	
31	4/15/2023	Saturday	IAD	UA 4165	6:59 PM	

```
12     15:00:00
13     15:00:00
14     15:00:00
15     15:00:00
16     11:00:00
17     00:00:00
18     11:00:00
19     00:00:00
20     11:00:00
21     00:00:00
22     11:00:00
23     23:00:00
24     14:00:00
25     19:00:00
26     14:00:00
27     19:00:00
28     14:00:00
29     19:00:00
30     14:00:00
31     19:00:00
```

[163]:
```
# Changing date format and extracting datetime using date and nearest hour

data_1['date'] = pd.to_datetime(data_1['Date'], format='%m/%d/%Y').dt.date
data_1['datetime'] = data_1.apply(lambda row: dt.datetime.combine(row['date'],␣
 ↪row['nearest_hour']), axis=1)
```

```
C:\Users\vidhe\AppData\Local\Temp\ipykernel_14620\667019764.py:1:
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

C:\Users\vidhe\AppData\Local\Temp\ipykernel_14620\667019764.py: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
```

[164]:
```
data_1
```

[164]:

	Date	Day	Origin Airport	Flight Number	Arrival Time
0	4/12/2023	Wednesday	ORD	UA 3839	10:00 AM
1	4/12/2023	Wednesday	ORD	UA 3524	4:52 PM
2	4/12/2023	Wednesday	ORD	UA 538	9:34 PM
3	4/13/2023	Thursday	ORD	UA 3839	10:00 AM
4	4/13/2023	Thursday	ORD	UA 3524	4:50 PM
5	4/13/2023	Thursday	ORD	UA 538	9:34 PM
6	4/14/2023	Friday	ORD	UA 3839	10:00 AM
7	4/14/2023	Friday	ORD	UA 3524	4:50 PM
8	4/14/2023	Friday	ORD	UA 538	9:34 PM
9	4/15/2023	Saturday	ORD	UA 3839	10:00 AM
10	4/15/2023	Saturday	ORD	UA 3524	4:50 PM
11	4/15/2023	Saturday	ORD	UA 538	9:34 PM
12	4/12/2023	Wednesday	DEN	UA 604	3:12 PM
13	4/13/2023	Thursday	DEN	UA 604	3:12 PM
14	4/14/2023	Friday	DEN	UA 604	3:12 PM
15	4/15/2023	Saturday	DEN	UA 604	3:12 PM
16	4/12/2023	Wednesday	EWR	UA 4189	10:46 AM
17	4/12/2023	Wednesday	EWR	UA 1412	11:42 PM
18	4/13/2023	Thursday	EWR	UA 4189	10:46 AM
19	4/13/2023	Thursday	EWR	UA 1412	11:42 PM
20	4/14/2023	Friday	EWR	UA 4189	10:46 AM
21	4/14/2023	Friday	EWR	UA 1412	11:42 PM
22	4/15/2023	Saturday	EWR	UA 4189	10:46 AM
23	4/15/2023	Saturday	EWR	UA 1412	11:17 PM
24	4/12/2023	Wednesday	IAD	UA 4490	1:57 PM
25	4/12/2023	Wednesday	IAD	UA 4165	6:59 PM
26	4/13/2023	Thursday	IAD	UA 4490	1:57 PM
27	4/13/2023	Thursday	IAD	UA 4165	6:59 PM
28	4/14/2023	Friday	IAD	UA 4490	1:57 PM
29	4/14/2023	Friday	IAD	UA 4165	6:59 PM
30	4/15/2023	Saturday	IAD	UA 3805	1:58 PM
31	4/15/2023	Saturday	IAD	UA 4165	6:59 PM

	nearest_hour	date	datetime
0	10:00:00	2023-04-12	2023-04-12 10:00:00
1	17:00:00	2023-04-12	2023-04-12 17:00:00
2	22:00:00	2023-04-12	2023-04-12 22:00:00
3	10:00:00	2023-04-13	2023-04-13 10:00:00
4	17:00:00	2023-04-13	2023-04-13 17:00:00
5	22:00:00	2023-04-13	2023-04-13 22:00:00
6	10:00:00	2023-04-14	2023-04-14 10:00:00
7	17:00:00	2023-04-14	2023-04-14 17:00:00
8	22:00:00	2023-04-14	2023-04-14 22:00:00
9	10:00:00	2023-04-15	2023-04-15 10:00:00

```
10      17:00:00  2023-04-15 2023-04-15 17:00:00
11      22:00:00  2023-04-15 2023-04-15 22:00:00
12      15:00:00  2023-04-12 2023-04-12 15:00:00
13      15:00:00  2023-04-13 2023-04-13 15:00:00
14      15:00:00  2023-04-14 2023-04-14 15:00:00
15      15:00:00  2023-04-15 2023-04-15 15:00:00
16      11:00:00  2023-04-12 2023-04-12 11:00:00
17      00:00:00  2023-04-12 2023-04-12 00:00:00
18      11:00:00  2023-04-13 2023-04-13 11:00:00
19      00:00:00  2023-04-13 2023-04-13 00:00:00
20      11:00:00  2023-04-14 2023-04-14 11:00:00
21      00:00:00  2023-04-14 2023-04-14 00:00:00
22      11:00:00  2023-04-15 2023-04-15 11:00:00
23      23:00:00  2023-04-15 2023-04-15 23:00:00
24      14:00:00  2023-04-12 2023-04-12 14:00:00
25      19:00:00  2023-04-12 2023-04-12 19:00:00
26      14:00:00  2023-04-13 2023-04-13 14:00:00
27      19:00:00  2023-04-13 2023-04-13 19:00:00
28      14:00:00  2023-04-14 2023-04-14 14:00:00
29      19:00:00  2023-04-14 2023-04-14 19:00:00
30      14:00:00  2023-04-15 2023-04-15 14:00:00
31      19:00:00  2023-04-15 2023-04-15 19:00:00
```

[167]:
```
# Retreiving April weather data

url = f'https://api.weatherbit.io/v2.0/history/hourly?
 ↪city=Syracuse&start_date=2023-04-11&end_date=2023-04-25&key=c1773a2891af4c87adeb01d743ee66e
response = requests.get(url)
if response.status_code == 200:
    print("Data fetched...")
    data = response.json()

    raw_weather_df = pd.DataFrame(data['data'])
    raw_weather_df.to_csv('raw_weather_2.csv', index = False)

else:
    print('Error:', response.status_code)
```

```
Data fetched...
```

[168]:
```
raw_weather_april_df = pd.read_csv("raw_weather_2.csv")
raw_weather_april_df.head()
```

[168]:

	app_temp	azimuth	clouds	datetime	dewpt	dhi	dni	elev_angle	ghi	\
0	16.70	285.50	37	2023-04-11:00	-3.90	0	0	-4.00	0	
1	13.30	296.40	50	2023-04-11:01	-1.80	0	0	-14.30	0	
2	12.20	308.70	25	2023-04-11:02	-2.40	0	0	-23.50	0	

```
3     11.10   323.00      25  2023-04-11:03  -2.30    0    0      -31.10    0
4     10.60   339.80      25  2023-04-11:04  -3.50    0    0      -36.30    0

   h_angle  …  temp      timestamp_local        timestamp_utc          ts  \
0      NaN  … 16.70  2023-04-10T20:00:00  2023-04-11T00:00:00  1681171200
1      NaN  … 13.30  2023-04-10T21:00:00  2023-04-11T01:00:00  1681174800
2      NaN  … 12.20  2023-04-10T22:00:00  2023-04-11T02:00:00  1681178400
3      NaN  … 11.10  2023-04-10T23:00:00  2023-04-11T03:00:00  1681182000
4      NaN  … 10.60  2023-04-11T00:00:00  2023-04-11T04:00:00  1681185600

    uv  vis                                              weather  wind_dir  \
0 0.00   16  {'description': 'Scattered clouds', 'code': 80…       250
1 0.00   16  {'description': 'Broken clouds', 'code': 803, …       180
2 0.00   16  {'description': 'Scattered clouds', 'code': 80…       110
3 0.00   16  {'description': 'Scattered clouds', 'code': 80…       130
4 0.00   16  {'description': 'Scattered clouds', 'code': 80…       130

   wind_gust_spd wind_spd
0           1.60     1.50
1           0.40     0.40
2           2.20     2.10
3           1.60     1.50
4           1.60     1.50

[5 rows x 28 columns]
```

[169]:
```
# Making a copy of weather data

april_weather_data_copy = raw_weather_april_df.copy()
```

[170]:
```
# Using the local time as datetime

april_weather_data_copy['datetime'] = pd.
 ↪to_datetime(april_weather_data_copy['timestamp_local'])
```

[172]:
```
april_weather_data_copy.iloc[0]
```

[172]:
```
app_temp                                    16.70
azimuth                                    285.50
clouds                                         37
datetime                    2023-04-11 00:00:00
dewpt                                      -3.90
dhi                                            0
dni                                            0
elev_angle                                 -4.00
ghi                                            0
h_angle                                      NaN
```

```
pod                                                                   n
precip                                                             0.00
pres                                                               1010
revision_status                                                   final
rh                                                                   24
slp                                                                1025
snow                                                               0.00
solar_rad                                                             0
temp                                                              16.70
timestamp_local                                     2023-04-10T20:00:00
timestamp_utc                                       2023-04-11T00:00:00
ts                                                           1681171200
uv                                                                 0.00
vis                                                                  16
weather           {'description': 'Scattered clouds', 'code': 80…
wind_dir                                                            250
wind_gust_spd                                                      1.60
wind_spd                                                           1.50
Name: 0, dtype: object
```

[173]:
```
# Merging flight and weather data with datetime as key

april_flight_weather_df = pd.merge(data_1, april_weather_data_copy,␣
 ↪on='datetime')
```

[225]:
```
april_flight_weather_df.head()
```

[225]:

	Date	Day	Origin Airport	Flight Number	Arrival Time	\
0	4/12/2023	Wednesday	ORD	UA 3839	10:00 AM	
1	4/12/2023	Wednesday	ORD	UA 3524	4:52 PM	
2	4/12/2023	Wednesday	ORD	UA 538	9:34 PM	
3	4/13/2023	Thursday	ORD	UA 3839	10:00 AM	
4	4/13/2023	Thursday	ORD	UA 3524	4:50 PM	

	nearest_hour	date	datetime	app_temp	azimuth	…	\
0	10:00:00	2023-04-12	2023-04-12 10:00:00	13.30	72.30	…	
1	17:00:00	2023-04-12	2023-04-12 17:00:00	20.80	177.80	…	
2	22:00:00	2023-04-12	2023-04-12 22:00:00	25.50	265.60	…	
3	10:00:00	2023-04-13	2023-04-13 10:00:00	16.70	72.10	…	
4	17:00:00	2023-04-13	2023-04-13 17:00:00	26.90	177.90	…	

	ts	uv	vis	weather	wind_dir	wind_gust_spd	wind_spd	week	\
0	1681293600	0.00	16	1	280	2.80	2.60	15	
1	1681318800	7.90	16	1	260	5.80	5.10	15	
2	1681336800	2.10	16	1	280	12.00	7.70	15	
3	1681380000	0.00	16	0	230	3.90	3.60	15	
4	1681405200	7.90	16	0	270	7.80	6.70	15	

```
   month  day
0      4   12
1      4   12
2      4   12
3      4   13
4      4   13

[5 rows x 38 columns]
```

[175]:
```
# Adding the respective week, month and day from datetime column

april_flight_weather_df['week'] = april_flight_weather_df['datetime'].dt.week
april_flight_weather_df['month'] = april_flight_weather_df['datetime'].dt.month
april_flight_weather_df['day'] = april_flight_weather_df['datetime'].dt.day
```

```
C:\Users\vidhe\AppData\Local\Temp\ipykernel_14620\3019203327.py:1:
FutureWarning:

Series.dt.weekofyear and Series.dt.week have been deprecated. Please use
Series.dt.isocalendar().week instead.
```

[179]:
```
# Extracting weather codes and replacing them with numeric data

april_flight_weather_df['weather'] = april_flight_weather_df['weather'].
 ↪apply(extract_weather_code)
april_flight_weather_df['weather'].replace([202, 500, 501, 502, 511, 600, 601,␣
 ↪610, 612, 623, 721, 741, 800, 801, 802, 803, 804],
                                           [3, 2, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2, 0, 1,␣
 ↪1, 1, 1], inplace=True)
```

[180]:
```
april_flight_weather_df.head()
```

[180]:
```
         Date        Day Origin Airport Flight Number Arrival Time  \
0  4/12/2023  Wednesday            ORD       UA 3839     10:00 AM
1  4/12/2023  Wednesday            ORD       UA 3524      4:52 PM
2  4/12/2023  Wednesday            ORD        UA 538      9:34 PM
3  4/13/2023   Thursday            ORD       UA 3839     10:00 AM
4  4/13/2023   Thursday            ORD       UA 3524      4:50 PM

  nearest_hour        date            datetime  app_temp  azimuth  …  \
0     10:00:00  2023-04-12 2023-04-12 10:00:00     13.30    72.30  …
1     17:00:00  2023-04-12 2023-04-12 17:00:00     20.80   177.80  …
2     22:00:00  2023-04-12 2023-04-12 22:00:00     25.50   265.60  …
3     10:00:00  2023-04-13 2023-04-13 10:00:00     16.70    72.10  …
4     17:00:00  2023-04-13 2023-04-13 17:00:00     26.90   177.90  …
```

```
           ts   uv  vis  weather  wind_dir  wind_gust_spd  wind_spd week  \
0  1681293600 0.00   16        1       280           2.80      2.60   15
1  1681318800 7.90   16        1       260           5.80      5.10   15
2  1681336800 2.10   16        1       280          12.00      7.70   15
3  1681380000 0.00   16        0       230           3.90      3.60   15
4  1681405200 7.90   16        0       270           7.80      6.70   15

   month  day
0      4   12
1      4   12
2      4   12
3      4   13
4      4   13

[5 rows x 38 columns]
```

[230]:
```
# Using the subset of columns for training

test_data = april_flight_weather_df[['day', 'week', 'month', 'wind_gust_spd',␣
 ↪'precip', 'snow', 'temp', 'weather', 'vis', 'solar_rad', 'dewpt']]
```

[250]:
```
test_data.head()
```

[250]:
```
   day  week  month  wind_gust_spd  precip  snow  temp  weather  vis  \
0   12    15      4           2.80    0.00  0.00 13.30        1   16
1   12    15      4           5.80    0.00  0.00 21.70        1   16
2   12    15      4          12.00    0.00  0.00 26.10        1   16
3   13    15      4           3.90    0.00  0.00 16.70        0   16
4   13    15      4           7.80    0.00  0.00 27.80        0   16

   solar_rad  dewpt
0          0   1.60
1        793   4.70
2        241   6.70
3          0   5.90
4        871   8.10
```

[251]:
```
# Testing using the XGBoost Model

dtest = xgb.DMatrix(test_data)
y_pred = model.predict(dtest)

test_output = pd.DataFrame(y_pred, index = test_data.index, columns =␣
 ↪['status'])
test_output['status'] = le.inverse_transform(test_output['status'].astype(int))
prediction = test_data.merge(test_output, left_index = True, right_index = True)
```

[254]:
```
prediction.head()
```

[254]:

	day	week	month	wind_gust_spd	precip	snow	temp	weather	vis	\
0	12	15	4	2.80	0	0	13.30	1	16	
1	12	15	4	5.80	0	0	21.70	1	16	
2	12	15	4	12.00	0	0	26.10	1	16	
3	13	15	4	3.90	0	0	16.70	0	16	
4	13	15	4	7.80	0	0	27.80	0	16	

	solar_rad	dewpt	status
0	0	1.60	On-time
1	793	4.70	On-time
2	241	6.70	On-time
3	0	5.90	On-time
4	871	8.10	On-time

[255]:
```
prediction
```

[255]:

	day	week	month	wind_gust_spd	precip	snow	temp	weather	vis	\
0	12	15	4	2.80	0	0	13.30	1	16	
1	12	15	4	5.80	0	0	21.70	1	16	
2	12	15	4	12.00	0	0	26.10	1	16	
3	13	15	4	3.90	0	0	16.70	0	16	
4	13	15	4	7.80	0	0	27.80	0	16	
5	13	15	4	5.80	0	0	29.40	0	16	
6	14	15	4	1.70	0	0	12.20	1	16	
7	14	15	4	0.40	0	0	27.80	1	16	
8	14	15	4	3.30	0	0	28.90	1	16	
9	15	15	4	2.80	0	0	11.70	1	16	
10	15	15	4	1.60	0	0	27.80	1	16	
11	15	15	4	3.90	0	0	27.20	1	16	
12	12	15	4	2.80	0	0	17.20	1	16	
13	13	15	4	8.00	0	0	25.00	0	16	
14	14	15	4	2.20	0	0	24.40	1	16	
15	15	15	4	2.10	0	0	25.60	1	16	
16	12	15	4	3.30	0	0	10.60	1	16	
17	12	15	4	9.20	0	0	20.00	1	16	
18	13	15	4	3.90	0	0	16.70	0	16	
19	13	15	4	6.50	0	0	22.80	1	16	
20	14	15	4	1.20	0	0	11.70	1	16	
21	14	15	4	4.50	0	0	25.00	1	16	
22	15	15	4	1.60	0	0	12.80	1	16	
23	15	15	4	2.80	0	0	26.70	1	16	
24	12	15	4	1.60	0	0	15.60	1	16	
25	12	15	4	12.00	0	0	25.60	1	16	
26	13	15	4	7.20	0	0	23.90	0	16	
27	13	15	4	10.00	0	0	29.40	0	16	

```
28   14    15      4           1.60       0      0 22.20        1    16
29   14    15      4           2.20       0      0 29.40        1    16
30   15    15      4           2.50       0      0 22.80        1    16
31   15    15      4           5.10       0      0 30.00        1    16
```

	solar_rad	dewpt	status
0	0	1.60	On-time
1	793	4.70	On-time
2	241	6.70	On-time
3	0	5.90	On-time
4	871	8.10	On-time
5	268	7.30	On-time
6	0	5.60	On-time
7	868	7.10	On-time
8	119	5.10	On-time
9	0	7.00	On-time
10	803	9.10	Late
11	242	10.80	On-time
12	560	0.80	On-time
13	744	6.70	On-time
14	654	5.70	On-time
15	658	9.00	On-time
16	58	2.20	On-time
17	0	2.80	On-time
18	57	5.90	On-time
19	0	6.60	On-time
20	65	5.60	On-time
21	0	3.60	On-time
22	68	7.00	On-time
23	79	12.00	On-time
24	263	1.00	On-time
25	753	7.70	On-time
26	603	6.70	On-time
27	763	8.50	On-time
28	267	7.60	On-time
29	337	5.50	On-time
30	608	9.90	On-time
31	703	9.00	Late

[234]:
```
prediction.to_csv('final prediction Apr 12-15.csv', index = False)
```

1.3.2 Apr 21 - 24

[237]:
```
data_2 = data_2.drop(columns=['Status (Early, On-time, Late, Severly Late)'])
data_2 = data_2.dropna()
```

[238]:
```
data_2['nearest_hour'] = pd.to_datetime(data_2['Arrival Time'], format='%I:%M␣
↪%p')
data_2['nearest_hour'] = data_2['nearest_hour'].dt.strftime('%H:%M')
data_2['nearest_hour'] = pd.to_datetime(data_2['nearest_hour']).dt.round('H').
↪dt.time
```

[239]:
```
data_2['date'] = pd.to_datetime(data_2['Date'], format='%m/%d/%Y').dt.date
data_2['datetime'] = data_2.apply(lambda row: dt.datetime.combine(row['date'],␣
↪row['nearest_hour']), axis=1)
```

[240]:
```
new_april_flight_weather_df = pd.merge(data_2, april_weather_data_copy,␣
↪on='datetime')
```

[241]:
```
new_april_flight_weather_df.head()
```

[241]:

	Date	Day	Origin Airport	Flight Number	Arrival Time	nearest_hour	\
0	4/21/2023	Friday	ORD	UA 3839	10:00 AM	10:00:00	
1	4/21/2023	Friday	ORD	UA 3524	4:50 PM	17:00:00	
2	4/21/2023	Friday	ORD	UA 538	9:34 PM	22:00:00	
3	4/22/2023	Saturday	ORD	UA 3839	10:00 AM	10:00:00	
4	4/22/2023	Saturday	ORD	UA 3524	4:50 PM	17:00:00	

	date	datetime	app_temp	azimuth	…	temp	\
0	2023-04-21	2023-04-21 10:00:00	10.60	70.40	…	10.60	
1	2023-04-21	2023-04-21 17:00:00	24.50	178.60	…	25.00	
2	2023-04-21	2023-04-21 22:00:00	27.80	268.40	…	29.40	
3	2023-04-22	2023-04-22 10:00:00	12.80	70.30	…	12.80	
4	2023-04-22	2023-04-22 17:00:00	20.90	178.70	…	21.70	

	timestamp_local	timestamp_utc	ts	uv	vis	\
0	2023-04-21T06:00:00	2023-04-21T10:00:00	1682071200	0.00	16	
1	2023-04-21T13:00:00	2023-04-21T17:00:00	1682096400	7.00	16	
2	2023-04-21T18:00:00	2023-04-21T22:00:00	1682114400	0.70	16	
3	2023-04-22T06:00:00	2023-04-22T10:00:00	1682157600	0.00	16	
4	2023-04-22T13:00:00	2023-04-22T17:00:00	1682182800	2.90	16	

	weather	wind_dir	wind_gust_spd	\
0	{'description': 'Overcast clouds', 'code': 804…	90	4.50	
1	{'description': 'Overcast clouds', 'code': 804…	100	2.80	
2	{'description': 'Broken clouds', 'code': 803, …	210	7.20	
3	{'description': 'Broken clouds', 'code': 803, …	70	4.50	
4	{'description': 'Overcast clouds', 'code': 804…	140	17.00	

	wind_spd
0	4.10
1	2.60
2	6.20

```
3      4.10
4     10.80

[5 rows x 35 columns]
```

[242]:
```
new_april_flight_weather_df['week'] = new_april_flight_weather_df['datetime'].
 ↪dt.week
new_april_flight_weather_df['month'] = new_april_flight_weather_df['datetime'].
 ↪dt.month
new_april_flight_weather_df['day'] = new_april_flight_weather_df['datetime'].dt.
 ↪day
```

```
C:\Users\vidhe\AppData\Local\Temp\ipykernel_14620\2857885748.py:1:
FutureWarning:

Series.dt.weekofyear and Series.dt.week have been deprecated. Please use
Series.dt.isocalendar().week instead.
```

[243]:
```
new_april_flight_weather_df['weather'] = new_april_flight_weather_df['weather'].
 ↪apply(extract_weather_code)

new_april_flight_weather_df['weather'].replace([202, 500, 501, 502, 511, 600,␣
 ↪601, 610, 612, 623, 721, 741, 800, 801, 802, 803, 804],
                                        [3, 2, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2, 0, 1,␣
 ↪1, 1, 1], inplace=True)
```

[244]:
```
new_april_flight_weather_df.head()
```

[244]:

	Date	Day	Origin Airport	Flight Number	Arrival Time	nearest_hour	\
0	4/21/2023	Friday	ORD	UA 3839	10:00 AM	10:00:00	
1	4/21/2023	Friday	ORD	UA 3524	4:50 PM	17:00:00	
2	4/21/2023	Friday	ORD	UA 538	9:34 PM	22:00:00	
3	4/22/2023	Saturday	ORD	UA 3839	10:00 AM	10:00:00	
4	4/22/2023	Saturday	ORD	UA 3524	4:50 PM	17:00:00	

	date	datetime	app_temp	azimuth	…	ts	uv	\
0	2023-04-21	2023-04-21 10:00:00	10.60	70.40	…	1682071200	0.00	
1	2023-04-21	2023-04-21 17:00:00	24.50	178.60	…	1682096400	7.00	
2	2023-04-21	2023-04-21 22:00:00	27.80	268.40	…	1682114400	0.70	
3	2023-04-22	2023-04-22 10:00:00	12.80	70.30	…	1682157600	0.00	
4	2023-04-22	2023-04-22 17:00:00	20.90	178.70	…	1682182800	2.90	

	vis	weather	wind_dir	wind_gust_spd	wind_spd	week	month	day
0	16	1	90	4.50	4.10	16	4	21
1	16	1	100	2.80	2.60	16	4	21
2	16	1	210	7.20	6.20	16	4	21

```
3   16        1        70           4.50      4.10   16      4   22
4   16        1       140          17.00     10.80   16      4   22

[5 rows x 38 columns]
```

[245]:
```
new_test_data = new_april_flight_weather_df[['day', 'week', 'month',␣
 ↪'wind_gust_spd', 'precip', 'snow', 'temp', 'weather', 'vis', 'solar_rad',␣
 ↪'dewpt']]
```

[246]:
```
new_test_data.head()
```

[246]:
```
   day  week  month  wind_gust_spd  precip  snow  temp  weather  vis  \
0   21    16      4           4.50    0.00  0.00 10.60        1   16
1   21    16      4           2.80    0.00  0.00 25.00        1   16
2   21    16      4           7.20    0.00  0.00 29.40        1   16
3   22    16      4           4.50    0.00  0.00 12.80        1   16
4   22    16      4          17.00    0.00  0.00 21.70        1   16

   solar_rad  dewpt
0          0   1.70
1        395   9.70
2        269   4.80
3          0   8.30
4        210   6.00
```

[247]:
```
dtest = xgb.DMatrix(new_test_data)
y_pred = model.predict(dtest)

test_output = pd.DataFrame(y_pred, index = new_test_data.index, columns =␣
 ↪['status'])
test_output['status'] = le.inverse_transform(test_output['status'].astype(int))
prediction = new_test_data.merge(test_output, left_index = True, right_index =␣
 ↪True)
```

[258]:
```
prediction.head()
```

[258]:
```
   day  week  month  wind_gust_spd  precip  snow  temp  weather  vis  \
0   21    16      4           4.50    0.00     0 10.60        1   16
1   21    16      4           2.80    0.00     0 25.00        1   16
2   21    16      4           7.20    0.00     0 29.40        1   16
3   22    16      4           4.50    0.00     0 12.80        1   16
4   22    16      4          17.00    0.00     0 21.70        1   16

   solar_rad  dewpt   status
0          0   1.70     Late
1        395   9.70  On-time
2        269   4.80     Late
```

```
3           0   8.30  On-time
4         210   6.00  On-time
```

[257]:
```
prediction
```

[257]:
```
     day  week  month  wind_gust_spd  precip  snow  temp  weather  vis  \
0     21    16      4           4.50    0.00     0 10.60        1   16
1     21    16      4           2.80    0.00     0 25.00        1   16
2     21    16      4           7.20    0.00     0 29.40        1   16
3     22    16      4           4.50    0.00     0 12.80        1   16
4     22    16      4          17.00    0.00     0 21.70        1   16
5     22    16      4          10.00    0.00     0 15.60        1   11
6     23    16      4           1.20    0.00     0  8.90        1   11
7     23    16      4           5.10    0.00     0 12.80        1   16
8     23    16      4           3.30    1.00     0  8.30        2   16
9     24    17      4           3.90    0.00     0  4.40        1   16
10    24    17      4          10.00    0.50     0  8.90        1   16
11    24    17      4           5.10    0.00     0  8.90        1   16
12    21    16      4           2.20    0.00     0 20.60        1   16
13    22    16      4          13.00    0.00     0 20.60        1   16
14    23    16      4           5.10    0.00     0 11.10        1   16
15    24    17      4           4.50    0.75     0  8.90        2   16
16    21    16      4           3.90    0.00     0 10.60        1   16
17    21    16      4           2.20    0.00     0 13.30        1   16
18    22    16      4           5.80    0.00     0 12.20        1   16
19    22    16      4           5.80    1.00     0 15.00        2   16
20    23    16      4           1.20    0.00     0  8.90        1   11
21    23    16      4           3.90    2.75     0 13.90        2    6
22    24    17      4           3.30    0.00     0  5.60        1   16
23    24    17      4           2.80    0.25     0  7.20        1   16
24    21    16      4           3.90    0.00     0 17.20        1   16
25    21    16      4           8.00    0.00     0 30.00        1   16
26    22    16      4           8.50    0.00     0 18.90        1   16
27    22    16      4          11.00    0.00     0 21.70        1   16
28    23    16      4           4.50    0.00     0 10.60        1   16
29    23    16      4           7.20    0.00     0 12.80        1   16
30    24    17      4           4.50    0.00     0  7.80        1   16
31    24    17      4           5.80    0.00     0 10.00        1   16

    solar_rad  dewpt   status
0           0   1.70     Late
1         395   9.70  On-time
2         269   4.80     Late
3           0   8.30  On-time
4         210   6.00  On-time
5          95  11.00  On-time
6           0   7.20     Late
```

```
7          398   3.20  On-time
8          107   0.50  On-time
9            0   0.40  On-time
10         211   1.10  On-time
11         133   1.60     Late
12         589   6.90     Late
13         191   9.20  On-time
14         345   4.30  On-time
15         193   1.60  On-time
16          84   2.20  On-time
17           0  -0.70  On-time
18          87   6.70  On-time
19          56  10.40     Late
20          89   7.20  On-time
21           0  10.50     Late
22          57   0.60  On-time
23           0   0.40  On-time
24         584   5.50  On-time
25         721   9.00  On-time
26         169   9.40     Late
27         193   7.20  On-time
28         284   6.00  On-time
29         349   0.90  On-time
30         170   1.60  On-time
31         350   0.60  On-time
```

[249]:
```
prediction.to_csv('final prediction Apr 21-24.csv', index = False)
```

[ ]:
```

```

[ ]:
```

```