In [1]:

```python
# Generic inputs for ML task
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import tree

from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestClassifier

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

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

import plotly.io as pio
pio.renderers.default='notebook'

from IPython.display import display, HTML
```

In [2]:

```python
airline_data = pd.read_csv('Detailed_Statistics_Arrivals.csv')
airline_data
```

Out[2]:

	Carrier Code	Date (MM/DD/YYYY)	Flight Number	Tail Number	Origin Airport	Scheduled Arrival Time	Actual Arrival Time	Scheduled Elapsed Time (Minutes)	Actual Elapsed Time (Minutes)	Arrival Delay (Minutes)	Wheels-on Time	Taxi-In time (Minutes)	Delay Carrier (Minutes)	(
0	UA	1/1/2000	356	N306UA	ORD	17:03	16:49	105	91	-14	16:44	5	0	
1	UA	1/1/2000	1498	N976UA	ORD	19:25	19:15	101	92	-10	19:09	6	0	
2	UA	1/1/2001	356	N981ä1	ORD	17:00	16:56	109	104	-4	16:50	6	0	
3	UA	1/1/2001	1498	N985ä1	ORD	23:32	0:13	107	108	41	0:06	7	0	
4	UA	1/1/2001	1620	N991ä1	ORD	9:03	8:57	103	101	-6	8:51	6	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5339	UA	12/31/2019	1460	N838UA	EWR	18:15	18:14	75	60	-1	18:08	6	0	
5340	UA	12/31/2021	467	N872UA	IAD	18:38	18:32	78	68	-6	18:27	5	0	
5341	UA	12/31/2022	604	N801UA	DEN	14:58	14:46	193	178	-12	14:39	7	0	
5342	UA	12/31/2022	1998	N23707	ORD	21:08	20:44	113	98	-24	20:40	4	0	
5343	UA	12/31/2022	2488	N37427	EWR	23:14	0:46	75	73	92	0:40	6	92	

5344 rows × 17 columns

In [3]:

```python
# printing the number of rows in the data frame
print("Number of rows in the data frame:", len(airline_data))

print("(Rows,Columns) = ", airline_data.shape)
print("\n")

# check for NaN values
print("NaN values in the file are:\n", airline_data.isna().any())

print("\n Count of NaN values in each column (feature):\n", airline_data.isna().sum())
print("\nCount of total NaN values in entire file:", airline_data.isna().sum().sum())
```

```
Number of rows in the data frame: 5344
(Rows,Columns) =  (5344, 17)


NaN values in the file are:
 Carrier Code                                False
Date (MM/DD/YYYY)                           False
Flight Number                               False
Tail Number                                  True
Origin Airport                              False
Scheduled Arrival Time                      False
Actual Arrival Time                         False
Scheduled Elapsed Time (Minutes)            False
Actual Elapsed Time (Minutes)               False
Arrival Delay (Minutes)                     False
Wheels-on Time                              False
Taxi-In time (Minutes)                      False
Delay Carrier (Minutes)                     False
Delay Weather (Minutes)                     False
Delay National Aviation System (Minutes)    False
Delay Security (Minutes)                    False
Delay Late Aircraft Arrival (Minutes)       False
dtype: bool

 Count of NaN values in each column (feature):
 Carrier Code                                 0
Date (MM/DD/YYYY)                            0
Flight Number                                0
Tail Number                                 63
Origin Airport                               0
Scheduled Arrival Time                       0
Actual Arrival Time                          0
Scheduled Elapsed Time (Minutes)             0
Actual Elapsed Time (Minutes)                0
Arrival Delay (Minutes)                      0
Wheels-on Time                               0
Taxi-In time (Minutes)                       0
Delay Carrier (Minutes)                      0
Delay Weather (Minutes)                      0
Delay National Aviation System (Minutes)     0
Delay Security (Minutes)                     0
Delay Late Aircraft Arrival (Minutes)        0
dtype: int64

Count of total NaN values in entire file: 63
```

In [4]:

```
airline_data.dropna(inplace=True)
print("\nCount of total NaN values in entire file:", airline_data.isna().sum().sum())
airline_data.isna().any()
```

```
Count of total NaN values in entire file: 0
```

Out[4]:

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

In [5]:

```
airline_data.dtypes
```

Out[5]:
```
Carrier Code                              object
Date (MM/DD/YYYY)                         object
Flight Number                              int64
Tail Number                               object
Origin Airport                            object
Scheduled Arrival Time                    object
Actual Arrival Time                       object
Scheduled Elapsed Time (Minutes)           int64
Actual Elapsed Time (Minutes)              int64
Arrival Delay (Minutes)                    int64
Wheels-on Time                            object
Taxi-In time (Minutes)                     int64
Delay Carrier (Minutes)                    int64
Delay Weather (Minutes)                    int64
Delay National Aviation System (Minutes)   int64
Delay Security (Minutes)                   int64
Delay Late Aircraft Arrival (Minutes)      int64
dtype: object
```

In [6]:
```
#parsing the Timestsamp column as a date
airline_data['Date'] = pd.to_datetime(airline_data['Date (MM/DD/YYYY)'])
airline_data.insert(2, 'Date', airline_data.pop('Date'))
airline_data = airline_data.sort_values(by='Date', ascending=False)

airline_data.columns
```

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

In [7]:
```
airline_data = airline_data.drop(['Date (MM/DD/YYYY)'], axis = 1)
airline_data.head()
```

Out[7]:

	Carrier Code	Date	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)
454	UA	2023-01-31	1998	N808UA	ORD	21:17	20:52	113	97	-25	20:48	4	0	0
455	UA	2023-01-31	2617	N68807	EWR	23:12	22:59	74	66	-13	22:55	4	0	0
453	UA	2023-01-31	604	N851UA	DEN	14:59	14:47	193	175	-12	14:40	7	0	0
438	UA	2023-01-30	604	N882UA	DEN	14:59	14:35	193	172	-24	14:30	5	0	0
439	UA	2023-01-30	1998	N836UA	ORD	21:17	21:21	113	103	4	21:16	5	0	0

In [8]:
```
airline_data['Actual Arrival Time'] = airline_data['Actual Arrival Time'].str.replace('24:00:00', '00:00:00')
airline_data['Wheels-on Time'] = airline_data['Wheels-on Time'].str.replace('24:00:00', '00:00:00')

# convert time column to datetime format
airline_data['Scheduled Arrival Time'] = pd.to_datetime(airline_data['Scheduled Arrival Time'])
airline_data['Actual Arrival Time'] = pd.to_datetime(airline_data['Actual Arrival Time'])
airline_data['Wheels-on Time'] = pd.to_datetime(airline_data['Wheels-on Time'])

# convert time to AM/PM format
airline_data['Scheduled Arrival Time'] = airline_data['Scheduled Arrival Time'].dt.strftime('%I:%M %p')
airline_data['Actual Arrival Time'] = airline_data['Actual Arrival Time'].dt.strftime('%I:%M %p')
airline_data['Wheels-on Time'] = airline_data['Wheels-on Time'].dt.strftime('%I:%M %p')

print(airline_data)
```

```
    Carrier Code       Date  Flight Number Tail Number Origin Airport  \
454           UA 2023-01-31           1998      N808UA            ORD   
455           UA 2023-01-31           2617      N68807            EWR   
453           UA 2023-01-31            604      N851UA            DEN   
438           UA 2023-01-30            604      N882UA            DEN   
439           UA 2023-01-30           1998      N836UA            ORD   
..           ...        ...            ...         ...            ...   
14            UA 2000-01-02            356      N361UA            ORD   
15            UA 2000-01-02           1498      N994UA            ORD   
16            UA 2000-01-02           1620      N994UA            ORD   
1             UA 2000-01-01           1498      N976UA            ORD   
0             UA 2000-01-01            356      N306UA            ORD   

    Scheduled Arrival Time Actual Arrival Time  \
454               09:17 PM            08:52 PM   
455               11:12 PM            10:59 PM   
```

```
453               02:59 PM            02:47 PM
438               02:59 PM            02:35 PM
439               09:17 PM            09:21 PM
..                     ...                 ...
14                05:03 PM            04:59 PM
15                07:26 PM            08:22 PM
16                09:25 AM            09:10 AM
1                 07:25 PM            07:15 PM
0                 05:03 PM            04:49 PM

     Scheduled Elapsed Time (Minutes)  Actual Elapsed Time (Minutes)  \
454                               113                             97
455                                74                             66
453                               193                            175
438                               193                            172
439                               113                            103
..                                ...                            ...
14                                105                             81
15                                101                             92
16                                 95                             82
1                                 101                             92
0                                 105                             91

     Arrival Delay (Minutes) Wheels-on Time  Taxi-In time (Minutes)  \
454                      -25       08:48 PM                       4
455                      -13       10:55 PM                       4
453                      -12       02:40 PM                       7
438                      -24       02:30 PM                       5
439                        4       09:16 PM                       5
..                       ...            ...                     ...
14                        -4       04:56 PM                       3
15                        56       08:17 PM                       5
16                       -15       09:06 AM                       4
1                        -10       07:09 PM                       6
0                        -14       04:44 PM                       5

     Delay Carrier (Minutes)  Delay Weather (Minutes)  \
454                        0                        0
455                        0                        0
453                        0                        0
438                        0                        0
439                        0                        0
..                       ...                      ...
14                         0                        0
15                         0                        0
16                         0                        0
1                          0                        0
0                          0                        0

     Delay National Aviation System (Minutes)  Delay Security (Minutes)  \
454                                         0                         0
455                                         0                         0
453                                         0                         0
438                                         0                         0
439                                         0                         0
..                                        ...                       ...
14                                          0                         0
15                                          0                         0
16                                          0                         0
1                                           0                         0
0                                           0                         0

     Delay Late Aircraft Arrival (Minutes)
454                                      0
455                                      0
453                                      0
438                                      0
439                                      0
..                                     ...
14                                       0
15                                       0
16                                       0
1                                        0
0                                        0

[5281 rows x 17 columns]
```

In [9]:

```
airline_data.dtypes

# Select columns with float data type
float_columns = airline_data.select_dtypes(include=['float'])

# Print the resulting float columns
print(float_columns)
```

Out[9]:

```
Carrier Code                                          object
Date                                          datetime64[ns]
Flight Number                                          int64
Tail Number                                           object
Origin Airport                                        object
Scheduled Arrival Time                                object
Actual Arrival Time                                   object
Scheduled Elapsed Time (Minutes)                       int64
Actual Elapsed Time (Minutes)                          int64
Arrival Delay (Minutes)                                int64
Wheels-on Time                                        object
Taxi-In time (Minutes)                                 int64
Delay Carrier (Minutes)                                int64
Delay Weather (Minutes)                                int64
Delay National Aviation System (Minutes)               int64
Delay Security (Minutes)                               int64
Delay Late Aircraft Arrival (Minutes)                  int64
dtype: object
Empty DataFrame
Columns: []
Index: [454, 455, 453, 438, 439, 440, 425, 426, 410, 411, 409, 394, 395, 396, 381, 380, 379, 366, 365, 364, 350
, 351, 349, 334, 335, 336, 322, 321, 320, 305, 307, 306, 291, 290, 292, 277, 276, 275, 262, 261, 260, 247, 246,
245, 230, 232, 231, 218, 217, 216, 203, 202, 201, 188, 187, 186, 173, 172, 171, 156, 157, 158, 141, 142, 143, 1
40, 127, 126, 125, 113, 112, 99, 98, 84, 85, 71, 70, 55, 56, 42, 41, 28, 27, 26, 13, 11, 12, 5343, 5341, 5342,
5331, 5329, 5330, 5317, 5318, 5319, 5307, 5306, 5305, 5293, ...]

[5281 rows x 0 columns]
```

In [10]:

```
airline_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5281 entries, 454 to 0
Data columns (total 17 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   Carrier Code                              5281 non-null   object
 1   Date                                      5281 non-null   datetime64[ns]
 2   Flight Number                             5281 non-null   int64
 3   Tail Number                               5281 non-null   object
 4   Origin Airport                            5281 non-null   object
 5   Scheduled Arrival Time                    5281 non-null   object
 6   Actual Arrival Time                       5281 non-null   object
 7   Scheduled Elapsed Time (Minutes)          5281 non-null   int64
 8   Actual Elapsed Time (Minutes)             5281 non-null   int64
 9   Arrival Delay (Minutes)                   5281 non-null   int64
 10  Wheels-on Time                            5281 non-null   object
 11  Taxi-In time (Minutes)                    5281 non-null   int64
 12  Delay Carrier (Minutes)                   5281 non-null   int64
 13  Delay Weather (Minutes)                   5281 non-null   int64
 14  Delay National Aviation System (Minutes)  5281 non-null   int64
 15  Delay Security (Minutes)                  5281 non-null   int64
 16  Delay Late Aircraft Arrival (Minutes)     5281 non-null   int64
dtypes: datetime64[ns](1), int64(10), object(6)
memory usage: 742.6+ KB
```

In [11]:

```
# removing unnecessary features
airline_data = airline_data.drop(['Carrier Code', 'Tail Number'], axis = 1)
airline_data
```

Out[11]:

	Date	Flight 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)	Del Securi (Minute
454	2023-01-31	1998	ORD	09:17 PM	08:52 PM	113	97	-25	08:48 PM	4	0	0	0	
455	2023-01-31	2617	EWR	11:12 PM	10:59 PM	74	66	-13	10:55 PM	4	0	0	0	
453	2023-01-31	604	DEN	02:59 PM	02:47 PM	193	175	-12	02:40 PM	7	0	0	0	
438	2023-01-30	604	DEN	02:59 PM	02:35 PM	193	172	-24	02:30 PM	5	0	0	0	
439	2023-01-30	1998	ORD	09:17 PM	09:21 PM	113	103	4	09:16 PM	5	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
14	2000-01-02	356	ORD	05:03 PM	04:59 PM	105	81	-4	04:56 PM	3	0	0	0	
15	2000-01-02	1498	ORD	07:26 PM	08:22 PM	101	92	56	08:17 PM	5	0	0	0	
16	2000-01-02	1620	ORD	09:25 AM	09:10 AM	95	82	-15	09:06 AM	4	0	0	0	
1	2000-01-01	1498	ORD	07:25 PM	07:15 PM	101	92	-10	07:09 PM	6	0	0	0	
0	2000-01-01	356	ORD	05:03 PM	04:49 PM	105	91	-14	04:44 PM	5	0	0	0	

5281 rows × 15 columns

In [12]:

```
airline_data['Status'] = pd.cut(airline_data['Arrival Delay (Minutes)'],
                                bins=[float('-inf'), -10, 10, 30, float('inf')],
                                labels=['Early', 'On-time', 'Late', 'Severely Late'])
```

In [13]:

```
airline_data.columns
airline_data.isna().sum().sum()
```

Out[13]:

```
Index(['Date', 'Flight 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)',
       'Status'],
      dtype='object')
```

Out[13]:

```
0
```

In [14]:

```
airline_data['Delay Weather (Minutes)'].unique()
```

Out[14]:

```
array([  0,  11, 985,  30,  82,  85,  41, 162, 115,  38,  34,  42,  24,
        92,  22, 143,   2,   4, 594, 134,  18,  15,  20,   1,  66,  19,
         3,  43,  59,  21,   7], dtype=int64)
```

In [15]:

```
airline_data['Weather_Delay'] = np.where(airline_data['Delay Weather (Minutes)'] > 0, 1, 0)
```

In [16]:

```
airline_data['Weather_Delay'].unique()
```

Out[16]:

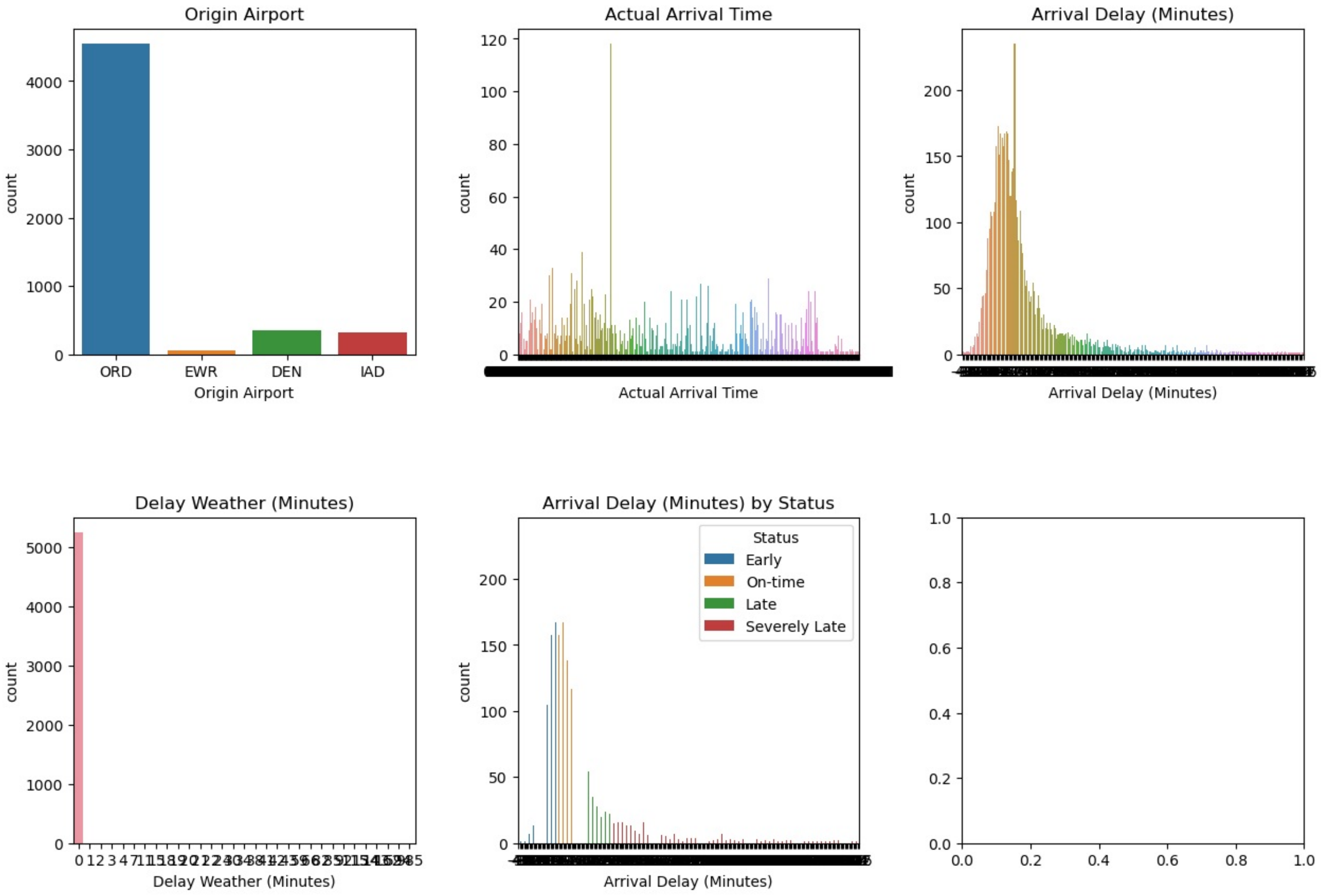
```
array([0, 1])
```

In [17]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# create subplots
fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))

# plot bar plots for each feature
sns.countplot(x='Origin Airport', data=airline_data, ax=axs[0, 0])
sns.countplot(x='Actual Arrival Time', data=airline_data, ax=axs[0, 1])
sns.countplot(x='Arrival Delay (Minutes)', data=airline_data, ax=axs[0, 2])
sns.countplot(x='Delay Weather (Minutes)', data=airline_data, ax=axs[1, 0])
sns.countplot(x='Arrival Delay (Minutes)', hue='Status', data=airline_data, ax=axs[1, 1])

# add titles to each plot
axs[0, 0].set_title('Origin Airport')
axs[0, 1].set_title('Actual Arrival Time')
axs[0, 2].set_title('Arrival Delay (Minutes)')
axs[1, 0].set_title('Delay Weather (Minutes)')
axs[1, 1].set_title('Arrival Delay (Minutes) by Status')

# adjust spacing between subplots
```

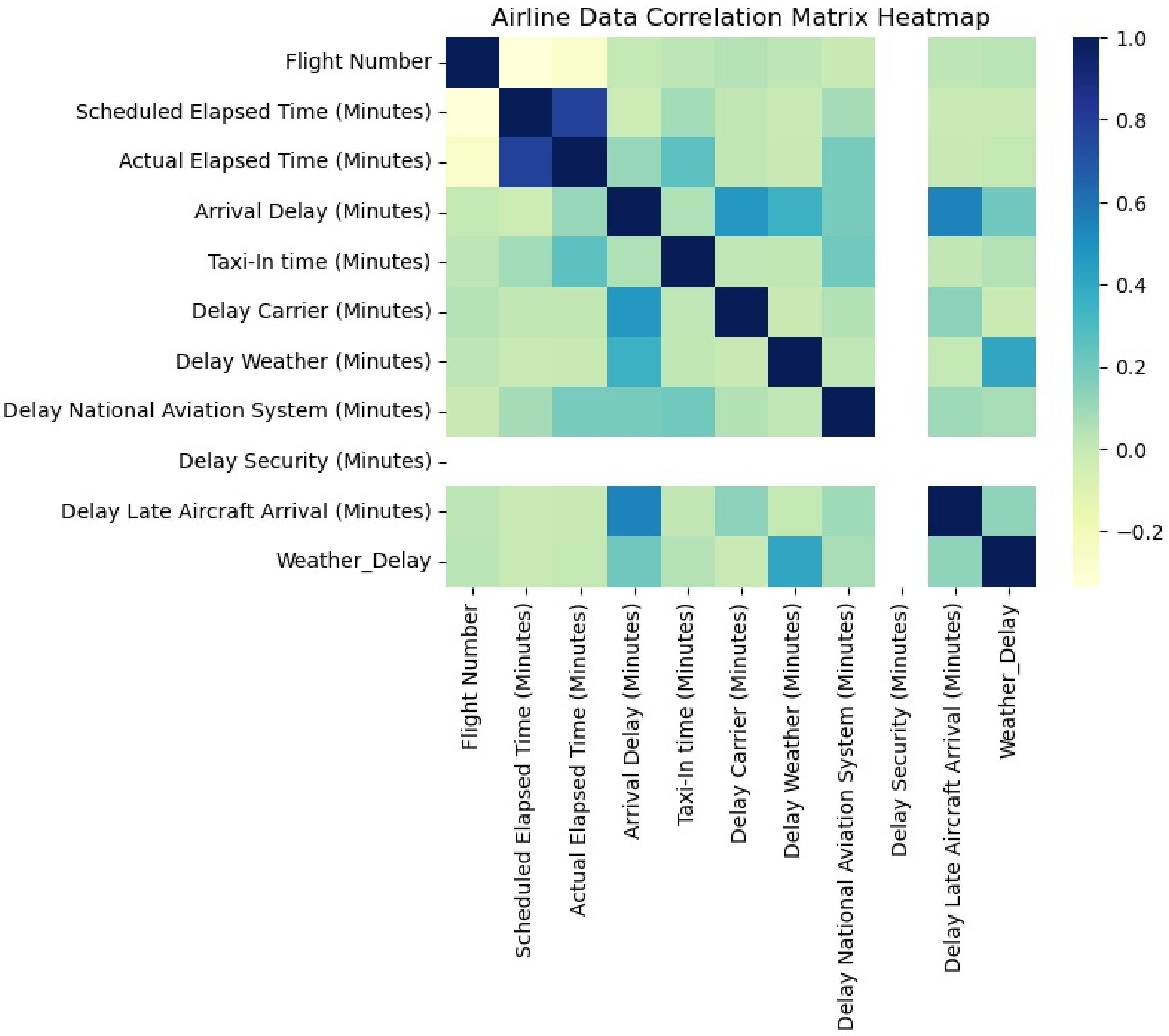
```
plt.subplots_adjust(wspace=0.3, hspace=0.5)

# show the plots
plt.show()
```

```
Out[17]: <AxesSubplot:xlabel='Origin Airport', ylabel='count'>
Out[17]: <AxesSubplot:xlabel='Actual Arrival Time', ylabel='count'>
Out[17]: <AxesSubplot:xlabel='Arrival Delay (Minutes)', ylabel='count'>
Out[17]: <AxesSubplot:xlabel='Delay Weather (Minutes)', ylabel='count'>
Out[17]: <AxesSubplot:xlabel='Arrival Delay (Minutes)', ylabel='count'>
Out[17]: Text(0.5, 1.0, 'Origin Airport')
Out[17]: Text(0.5, 1.0, 'Actual Arrival Time')
Out[17]: Text(0.5, 1.0, 'Arrival Delay (Minutes)')
Out[17]: Text(0.5, 1.0, 'Delay Weather (Minutes)')
Out[17]: Text(0.5, 1.0, 'Arrival Delay (Minutes) by Status')
```



```
In [18]: import seaborn as sns
         import matplotlib.pyplot as plt

         correl = airline_data.corr()
         sns.heatmap(correl, cmap="YlGnBu")
         plt.title("Airline Data Correlation Matrix Heatmap")
         plt.show()
```

```
Out[18]: <AxesSubplot:>
Out[18]: Text(0.5, 1.0, 'Airline Data Correlation Matrix Heatmap')
```



In [19]:

```
airline_data.drop(columns=['Actual Arrival Time', 'Arrival Delay (Minutes)',
       'Scheduled Elapsed Time (Minutes)', 'Actual Elapsed Time (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)'],inplace=True)
airline_data.head()
```

Out[19]:

	Date	Flight Number	Origin Airport	Scheduled Arrival Time	Status	Weather_Delay
454	2023-01-31	1998	ORD	09:17 PM	Early	0
455	2023-01-31	2617	EWR	11:12 PM	Early	0
453	2023-01-31	604	DEN	02:59 PM	Early	0
438	2023-01-30	604	DEN	02:59 PM	Early	0
439	2023-01-30	1998	ORD	09:17 PM	On-time	0

In [20]:

```
status_map = {'Early': 0, 'Severely Late': 1, 'Late': 2, 'On-time': 3}
airline_data['Status'] = airline_data['Status'].map(status_map)
```

In [21]:

```
airline_data['Scheduled Arrival Hour']= pd.to_datetime(airline_data['Scheduled Arrival Time']).dt.hour
airline_data['Scheduled Arrival Minutes']= pd.to_datetime(airline_data['Scheduled Arrival Time']).dt.minute

airline_data.drop(columns=['Scheduled Arrival Time','Date'],inplace=True)
```

In [22]:

```
airline_data.head()
airline_data.columns
```

Out[22]:

	Flight Number	Origin Airport	Status	Weather_Delay	Scheduled Arrival Hour	Scheduled Arrival Minutes
454	1998	ORD	0	0	21	17
455	2617	EWR	0	0	23	12
453	604	DEN	0	0	14	59
438	604	DEN	0	0	14	59
439	1998	ORD	3	0	21	17

Out[22]:

```
Index(['Flight Number', 'Origin Airport', 'Status', 'Weather_Delay',
       'Scheduled Arrival Hour', 'Scheduled Arrival Minutes'],
      dtype='object')
```

In [23]:

```
#dummy variables (one-hot encoding)
cat_cols = airline_data.select_dtypes(include=['object']).columns.tolist()
airline_data = pd.get_dummies(airline_data, columns=cat_cols, drop_first=True)

airline_data.head()
```

Out[23]:

	Flight Number	Status	Weather_Delay	Scheduled Arrival Hour	Scheduled Arrival Minutes	Origin Airport_EWR	Origin Airport_IAD	Origin Airport_ORD
454	1998	0	0	21	17	0	0	1
455	2617	0	0	23	12	1	0	0
453	604	0	0	14	59	0	0	0
438	604	0	0	14	59	0	0	0
439	1998	3	0	21	17	0	0	1

In [24]:

```
# Separate the features and the target variable
X = airline_data.drop(columns=["Status"])
y = airline_data["Status"]

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
airline_data = pd.DataFrame(sc.fit_transform(airline_data), columns = airline_data.columns, index = airline_dat
airline_data.head()
```

Out[24]:

	Flight Number	Status	Weather_Delay	Scheduled Arrival Hour	Scheduled Arrival Minutes	Origin Airport_EWR	Origin Airport_IAD	Origin Airport_ORD
454	1.18	-1.31	-0.09	0.91	-0.69	-0.11	-0.26	0.40
455	2.18	-1.31	-0.09	1.32	-0.93	9.17	-0.26	-2.48
453	-1.08	-1.31	-0.09	-0.55	1.34	-0.11	-0.26	-2.48
438	-1.08	-1.31	-0.09	-0.55	1.34	-0.11	-0.26	-2.48
439	1.18	0.98	-0.09	0.91	-0.69	-0.11	-0.26	0.40

In [25]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=50)

# print the length of the train and test data
print("Length of train data:", len(X_train))
print("Length of test data:", len(X_test))

print("\n")

X_train
X_test
y_train
y_test
```

```
Length of train data: 4224
Length of test data: 1057
```

Out[25]:

	Flight Number	Weather_Delay	Scheduled Arrival Hour	Scheduled Arrival Minutes	Origin Airport_EWR	Origin Airport_IAD	Origin Airport_ORD
590	1498	0	20	10	0	0	1
1135	607	0	21	50	0	0	1
4268	1730	0	16	54	0	0	1
5276	1498	0	9	44	0	0	1
4937	342	0	15	58	0	0	1
...	...	...	...	...	...	...	...
4395	1094	0	15	54	0	0	1
112	604	0	15	2	0	0	0
5152	2488	0	23	14	1	0	0
3387	1500	0	16	50	0	0	1
4450	1260	0	16	37	0	0	1

4224 rows × 7 columns

Out[25]:

	Flight Number	Weather_Delay	Scheduled Arrival Hour	Scheduled Arrival Minutes	Origin Airport_EWR	Origin Airport_IAD	Origin Airport_ORD
2463	356	0	15	58	0	0	1
4529	1620	0	8	48	0	0	1
4825	1498	0	23	23	0	0	1
3645	1498	0	21	1	0	0	1
722	356	0	20	32	0	0	1
...	...	...	...	...	...	...	...
4610	342	0	16	3	0	0	1
3686	1730	0	16	49	0	0	1
829	1498	0	20	41	0	0	1
541	1620	0	9	4	0	0	1
693	1620	0	9	13	0	0	1

1057 rows × 7 columns

Out[25]:

```
590     0
1135    3
4268    3
5276    3
4937    3
       ..
4395    0
112     2
5152    2
3387    0
4450    3
Name: Status, Length: 4224, dtype: category
Categories (4, int64): [0 < 3 < 2 < 1]
```

Out[25]:

```
2463    0
4529    3
4825    3
3645    1
722     3
       ..
4610    3
3686    2
829     2
541     3
693     0
Name: Status, Length: 1057, dtype: category
Categories (4, int64): [0 < 3 < 2 < 1]
```

In [26]:

```
from sklearn.ensemble import RandomForestClassifier

# create a Random Forest Classifier object
rf_clf = RandomForestClassifier(n_estimators=1000,
                                 random_state=50,
                                 max_depth=10,
                                 min_samples_split=5,
                                 min_samples_leaf=3,
                                 max_features=7)

# fit the model to the training data
rf_clf.fit(X_train, y_train)

# make predictions on the testing data
y_pred = rf_clf.predict(X_test)
```

Out[26]:

```
RandomForestClassifier(max_depth=10, max_features=7, min_samples_leaf=3,
                       min_samples_split=5, n_estimators=1000, random_state=50)
```

In [27]:

```python
test_output = pd.DataFrame(rf_clf.predict(X_test), index = X_test.index, columns = ['pred_Status'])
test_output.head()
```

Out[27]:

	pred_Status
2463	0
4529	3
4825	3
3645	3
722	3

In [28]:

```python
test_output = test_output.merge(y_test, left_index = True, right_index = True)
test_output.head()
```

Out[28]:

	pred_Status	Status
2463	0	0
4529	3	3
4825	3	3
3645	3	1
722	3	3

In [29]:

```python
april_test_data = pd.read_csv('project csv(Apr 21-24).csv')
Output_data = pd.read_csv('project csv(Apr 21-24).csv')
april_test_data.head()
```

Out[29]:

	Date	Day	Origin Airport	Flight Number	Arrival Time	Status (Early, On-time, Late, Severly Late)
0	4/21/2023	Friday	ORD	UA 3839	10:00 AM	NaN
1	4/21/2023	Friday	ORD	UA 3524	4:50 PM	NaN
2	4/21/2023	Friday	ORD	UA 538	9:34 PM	NaN
3	4/22/2023	Saturday	ORD	UA 3839	10:00 AM	NaN
4	4/22/2023	Saturday	ORD	UA 3524	4:50 PM	NaN

In [30]:

```python
april_test_data = april_test_data.drop(columns = 'Status (Early, On-time, Late, Severly Late)')
april_test_data.head()
```

Out[30]:

	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

In [31]:

```python
april_test_data.columns
april_test_data.dtypes
```

Out[31]:

```
Index(['Date', 'Day', 'Origin Airport', 'Flight Number', 'Arrival Time'], dtype='object')
```

Out[31]:

```
Date              object
Day               object
Origin Airport    object
Flight Number     object
Arrival Time      object
dtype: object
```

In [32]:

```python
april_test_data.dropna(inplace = True)
april_test_data.isna().any()
april_test_data.isna().sum().sum()
```

Out[32]:

```
Date              False
Day               False
Origin Airport    False
Flight Number     False
Arrival Time      False
dtype: bool
```

Out[32]:

```
0
```

In [33]:

```python
airline_data.columns
april_test_data.columns
```

Out[33]:
```
Index(['Flight Number', 'Status', 'Weather_Delay', 'Scheduled Arrival Hour',
       'Scheduled Arrival Minutes', 'Origin Airport_EWR', 'Origin Airport_IAD',
       'Origin Airport_ORD'],
      dtype='object')
```

Out[33]:
```
Index(['Date', 'Day', 'Origin Airport', 'Flight Number', 'Arrival Time'], dtype='object')
```

In [34]:
```
import datetime
april_test_data['Scheduled Arrival Time'] = april_test_data['Arrival Time'].str.strip().apply(lambda x: datetim
april_test_data.head()
```

Out[34]:

	Date	Day	Origin Airport	Flight Number	Arrival Time	Scheduled Arrival Time
0	4/21/2023	Friday	ORD	UA 3839	10:00 AM	10:00
1	4/21/2023	Friday	ORD	UA 3524	4:50 PM	16:50
2	4/21/2023	Friday	ORD	UA 538	9:34 PM	21:34
3	4/22/2023	Saturday	ORD	UA 3839	10:00 AM	10:00
4	4/22/2023	Saturday	ORD	UA 3524	4:50 PM	16:50

In [35]:
```
april_test_data.drop(columns=['Date','Day','Arrival Time'],inplace=True)
```

In [36]:
```
weather_delay_list = [0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
april_test_data['Weather_Delay'] = weather_delay_list
```

In [37]:
```
april_test_data['Scheduled Arrival Hour']= pd.to_datetime(april_test_data['Scheduled Arrival Time']).dt.hour
april_test_data['Scheduled Arrival Minutes']= pd.to_datetime(april_test_data['Scheduled Arrival Time']).dt.minu
april_test_data.drop(columns=['Scheduled Arrival Time'],inplace=True)
```

In [38]:
```
april_test_data['Flight Number'] = april_test_data['Flight Number'].str.extract('(\d+)', expand=False).astype(i
```

In [39]:
```
#dummy variables (one-hot encoding)
cat_cols = april_test_data.select_dtypes(include=['object']).columns.tolist()
april_test_data = pd.get_dummies(april_test_data, columns=cat_cols, drop_first=True)

april_test_data.head()
```

Out[39]:

	Flight Number	Weather_Delay	Scheduled Arrival Hour	Scheduled Arrival Minutes	Origin Airport_EWR	Origin Airport_IAD	Origin Airport_ORD
0	3839	0	10	0	0	0	1
1	3524	1	16	50	0	0	1
2	538	0	21	34	0	0	1
3	3839	0	10	0	0	0	1
4	3524	0	16	50	0	0	1

In [40]:
```
april_test_data.head()
airline_data.head()
```

Out[40]:

	Flight Number	Weather_Delay	Scheduled Arrival Hour	Scheduled Arrival Minutes	Origin Airport_EWR	Origin Airport_IAD	Origin Airport_ORD
0	3839	0	10	0	0	0	1
1	3524	1	16	50	0	0	1
2	538	0	21	34	0	0	1
3	3839	0	10	0	0	0	1
4	3524	0	16	50	0	0	1

Out[40]:

	Flight Number	Status	Weather_Delay	Scheduled Arrival Hour	Scheduled Arrival Minutes	Origin Airport_EWR	Origin Airport_IAD	Origin Airport_ORD
454	1.18	-1.31	-0.09	0.91	-0.69	-0.11	-0.26	0.40
455	2.18	-1.31	-0.09	1.32	-0.93	9.17	-0.26	-2.48
453	-1.08	-1.31	-0.09	-0.55	1.34	-0.11	-0.26	-2.48
438	-1.08	-1.31	-0.09	-0.55	1.34	-0.11	-0.26	-2.48
439	1.18	0.98	-0.09	0.91	-0.69	-0.11	-0.26	0.40

In [41]:
```
rf_clf.predict(april_test_data)
```

Out[41]:
```
array([0, 1, 3, 0, 0, 3, 1, 0, 1, 0, 0, 3, 3, 1, 3, 3, 0, 2, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0], dtype=int64)
```

In [42]:
```
test_output = pd.DataFrame(rf_clf.predict(april_test_data), index = april_test_data.index, columns = ['Status']
test_output.head()
test_data = test_output.merge(april_test_data, left_index = True, right_index = True)
test_data = test_data.drop(columns = ['Weather_Delay'])
```

Out[42]:

	Status
0	0
1	1
2	3
3	0
4	0

In [43]:

```
test_data['Status'].replace(0,"Early",inplace=True)
test_data['Status'].replace(1,"Severely Late",inplace=True)
test_data['Status'].replace(2,"Late",inplace=True)
test_data['Status'].replace(3,"On-time",inplace=True)
```

In [44]:

```
# make predictions and store them in a list or series
Output_data['Status (Early, On-time, Late, Severly Late)'] = test_data['Status']

# write the updated dataframe to the CSV file, overwriting the existing file
Output_data.to_csv('Output.csv', index=False)
```