Importing the required packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, precision_recall_fscore
import joblib
import pickle
```

Loading the dataset

In [2]:
```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your cr
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='PM77kN4FyWauT9b7yYQ1Mgxi4SbBI4_3cKxpFreNFXdY',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'flightdelayprediction-donotdelete-pr-3uvg7e82sbqe8r'
object_key = 'flightdata.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()
```

Out[2]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	C
0	2016	1	1	1	5	DL	N836DN	1399	
1	2016	1	1	1	5	DL	N964DN	1476	
2	2016	1	1	1	5	DL	N813DN	1597	
3	2016	1	1	1	5	DL	N587NW	1768	
4	2016	1	1	1	5	DL	N836DN	1823	

5 rows × 26 columns

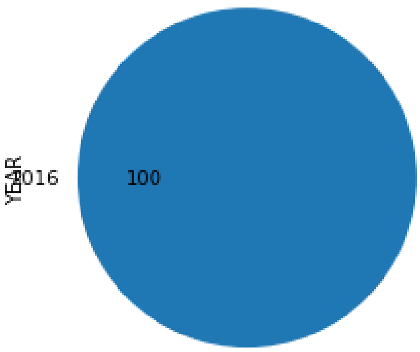
In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11231 entries, 0 to 11230
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   YEAR                 11231 non-null  int64  
 1   QUARTER              11231 non-null  int64  
 2   MONTH                11231 non-null  int64  
 3   DAY_OF_MONTH         11231 non-null  int64  
 4   DAY_OF_WEEK          11231 non-null  int64  
 5   UNIQUE_CARRIER       11231 non-null  object 
 6   TAIL_NUM             11231 non-null  object 
 7   FL_NUM               11231 non-null  int64  
 8   ORIGIN_AIRPORT_ID    11231 non-null  int64  
 9   ORIGIN               11231 non-null  object 
 10  DEST_AIRPORT_ID      11231 non-null  int64  
 11  DEST                 11231 non-null  object 
 12  CRS_DEP_TIME         11231 non-null  int64  
 13  DEP_TIME             11124 non-null  float64
 14  DEP_DELAY            11124 non-null  float64
 15  DEP_DEL15            11124 non-null  float64
 16  CRS_ARR_TIME         11231 non-null  int64  
 17  ARR_TIME             11116 non-null  float64
 18  ARR_DELAY            11043 non-null  float64
 19  ARR_DEL15            11043 non-null  float64
 20  CANCELLED            11231 non-null  float64
 21  DIVERTED             11231 non-null  float64
 22  CRS_ELAPSED_TIME     11231 non-null  float64
 23  ACTUAL_ELAPSED_TIME  11043 non-null  float64
 24  DISTANCE             11231 non-null  float64
 25  Unnamed: 25          0 non-null      float64
dtypes: float64(12), int64(10), object(4)
memory usage: 2.2+ MB
```

Performing Univariate Analysis

Using Pie Chart

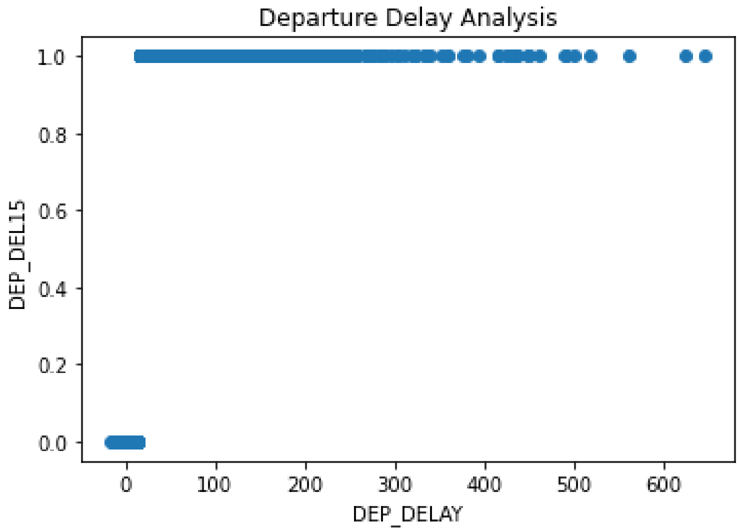
In [4]:
```
df['YEAR'].value_counts().plot(kind='pie', autopct='%.0f')
plt.show()
```



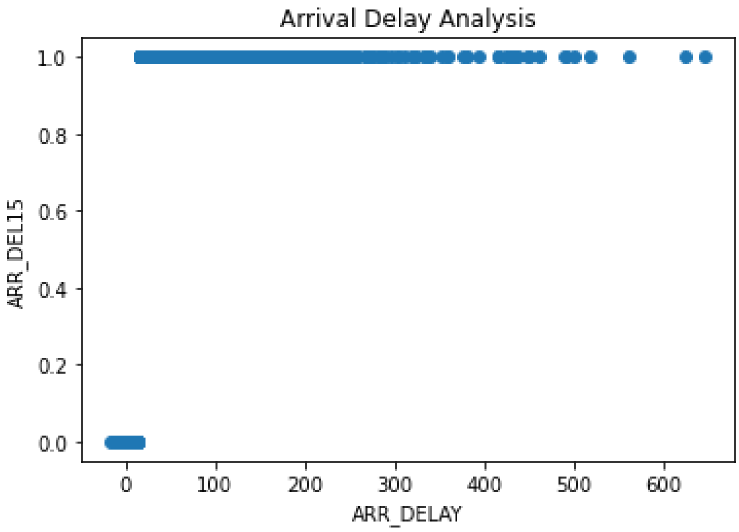
Performing Bivariate Analysis

Using scatterplot

In [5]:
```
plt.scatter(df.DEP_DELAY, df.DEP_DEL15)
plt.title('Departure Delay Analysis')
plt.xlabel('DEP_DELAY')
plt.ylabel('DEP_DEL15')
plt.show()
```

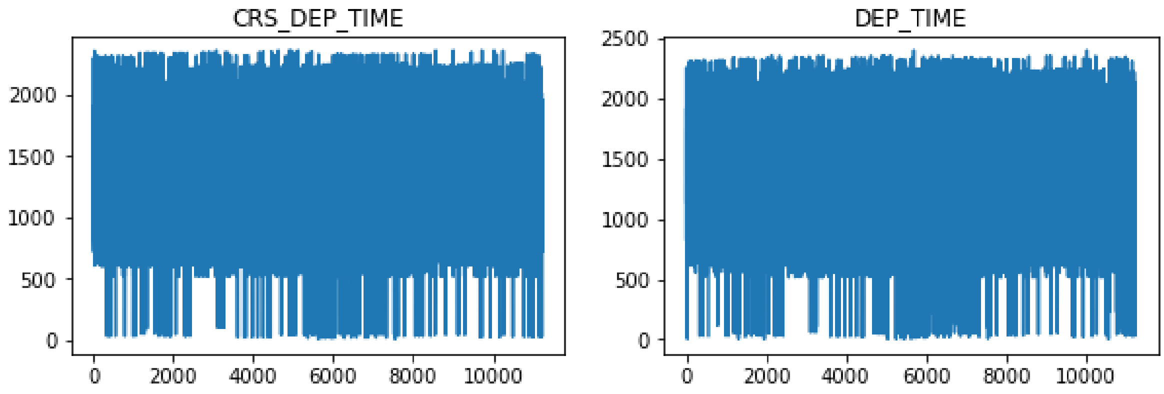


In [6]:
```
plt.scatter(df.DEP_DELAY, df.DEP_DEL15)
plt.title('Arrival Delay Analysis')
plt.xlabel('ARR_DELAY')
plt.ylabel('ARR_DEL15')
plt.show()
```

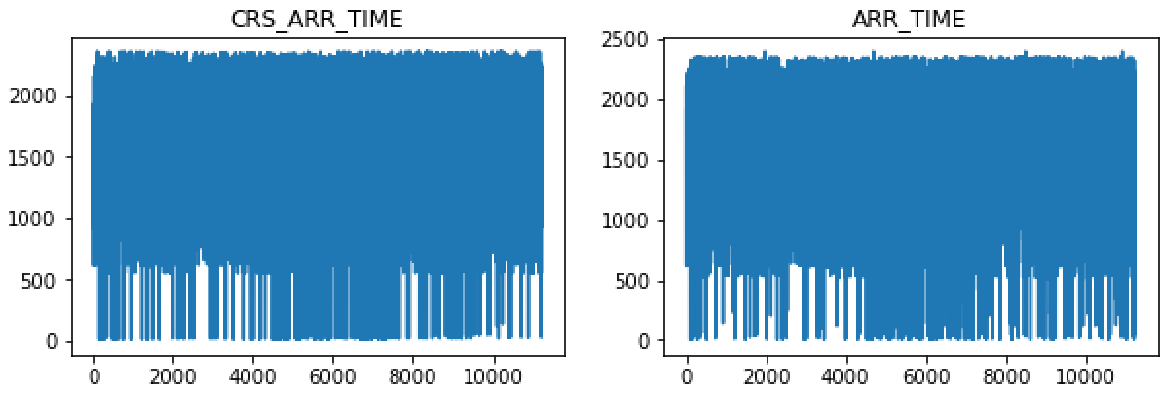


Using lineplots

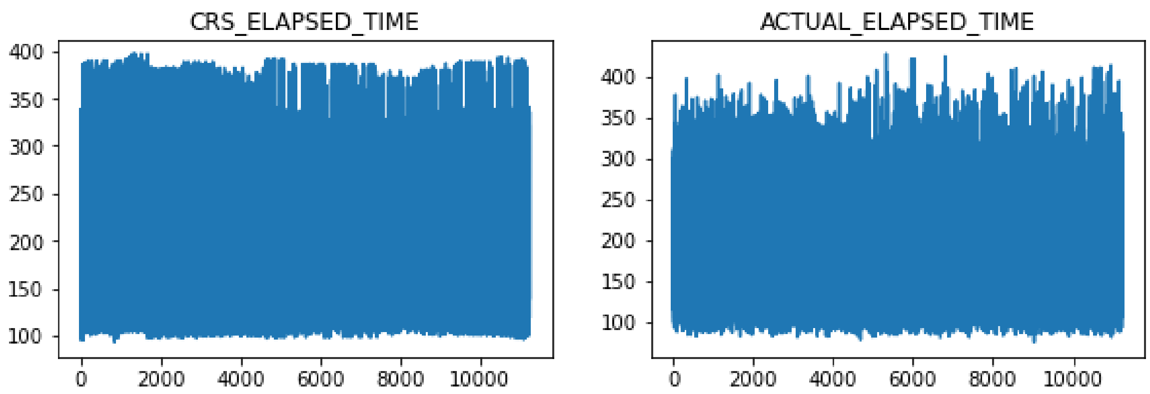
In [7]:
```
fig, ax = plt.subplots(figsize=(10, 3))
plt.subplot(1, 2, 1)
plt.title('CRS_DEP_TIME')
plt.plot(df.CRS_DEP_TIME)
plt.subplot(1, 2, 2)
plt.title('DEP_TIME')
plt.plot(df.DEP_TIME)
plt.show()
```



In [8]:
```
fig, ax = plt.subplots(figsize=(10, 3))
plt.subplot(1, 2, 1)
plt.title('CRS_ARR_TIME')
plt.plot(df.CRS_ARR_TIME)
plt.subplot(1, 2, 2)
plt.title('ARR_TIME')
plt.plot(df.ARR_TIME)
plt.show()
```



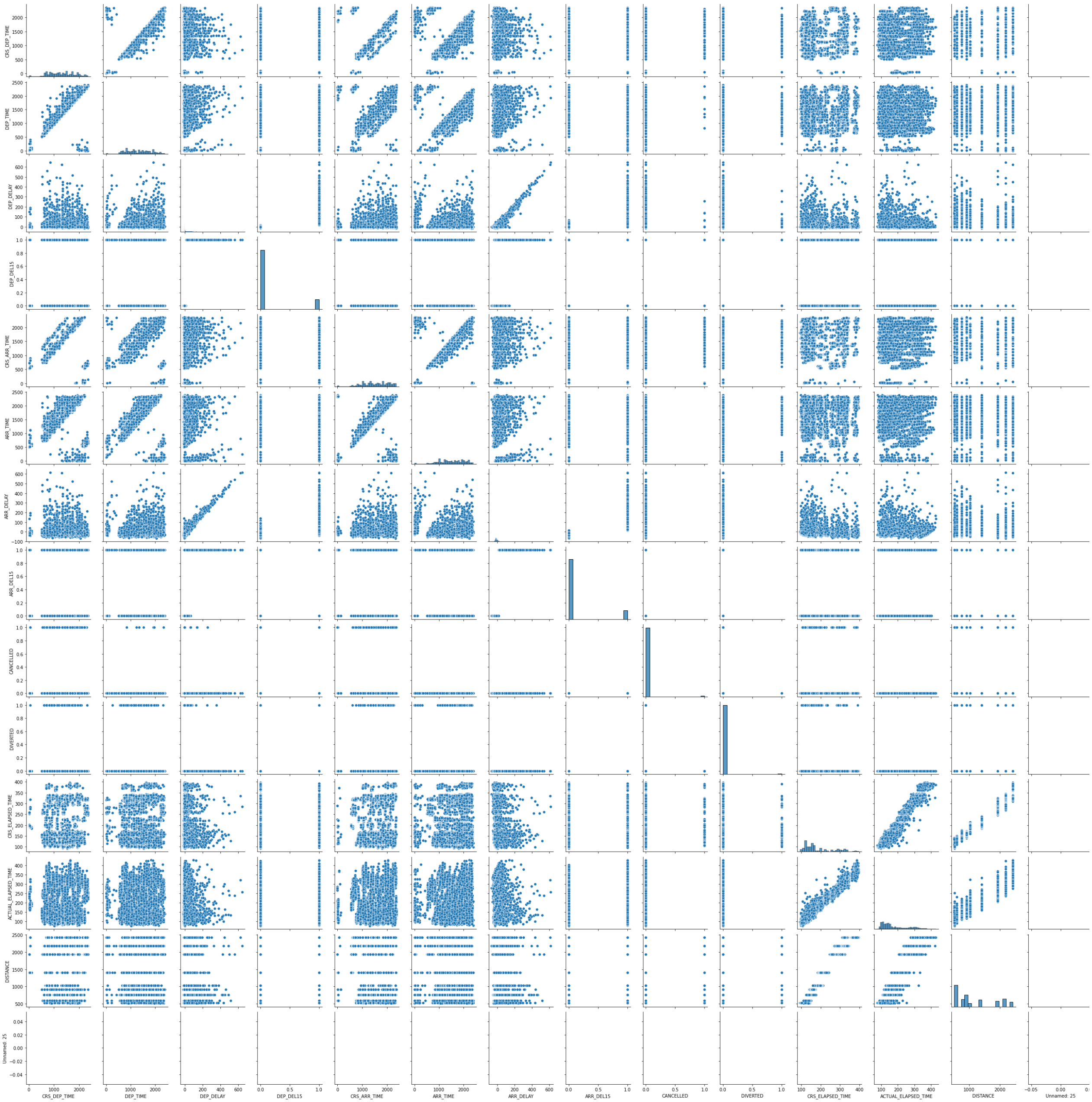
In [9]:
```
fig, ax = plt.subplots(figsize=(10, 3))
plt.subplot(1, 2, 1)
plt.title('CRS_ELAPSED_TIME')
plt.plot(df.CRS_ELAPSED_TIME)
plt.subplot(1, 2, 2)
plt.title('ACTUAL_ELAPSED_TIME')
plt.plot(df.ACTUAL_ELAPSED_TIME)
plt.show()
```



Performing Multivariate Analysis

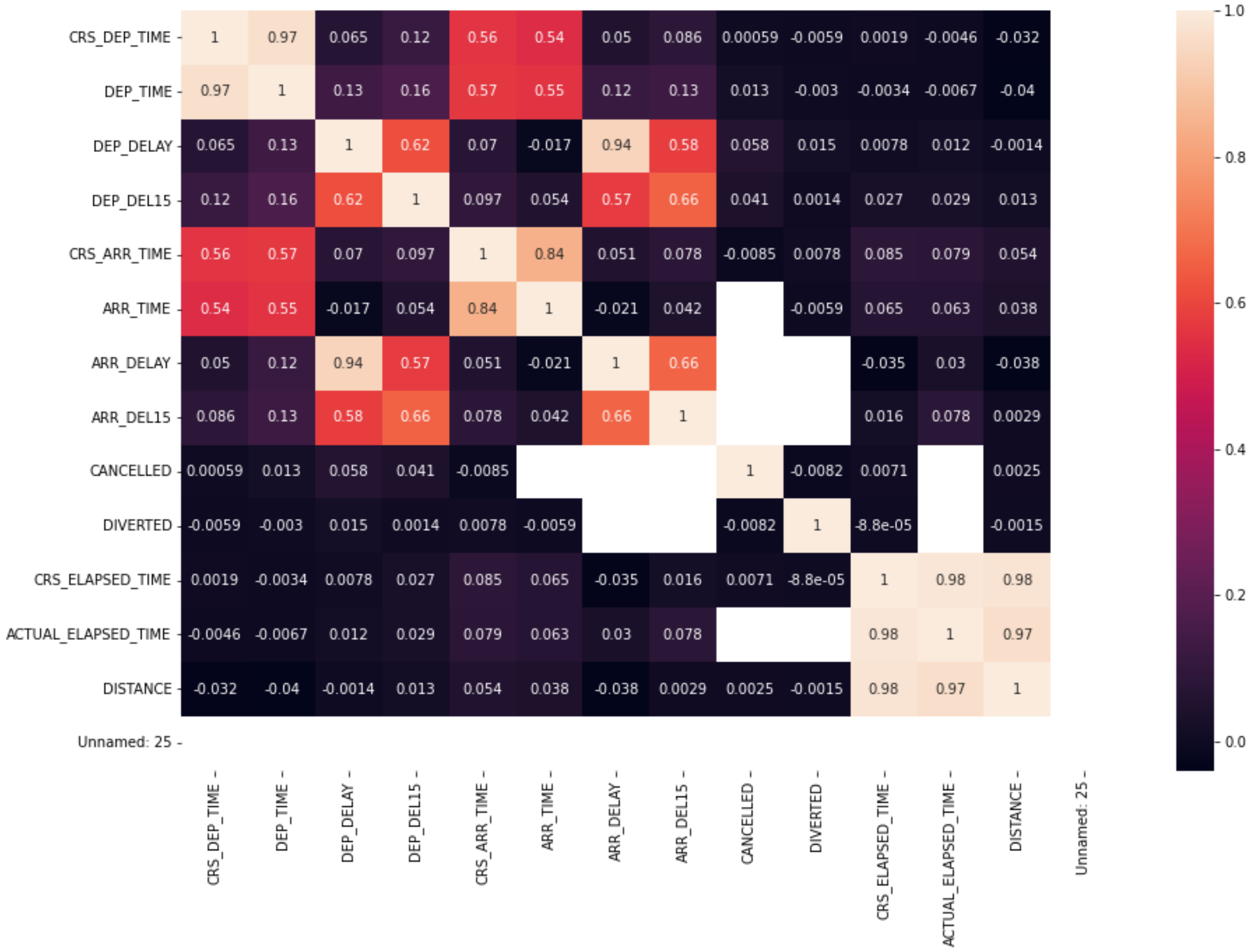
Using pairplot

In [10]:
```
sb.pairplot(df.iloc[:, 12:])
plt.show()
```



Using heatmap

In [11]:
```
fig, ax = plt.subplots(figsize=(15, 10))
sb.heatmap(df.iloc[:, 12:].corr(), annot=True, ax=ax)
plt.show()
```



Performing Descriptive Analysis

In [12]:
```
df.describe()
```

Out[12]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPC
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.(
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.!
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.(
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.(
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.(
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.(
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.(
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.(

8 rows × 22 columns

Dropping unnecessary columns

In [13]:
```
df = df[['FL_NUM', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK', 'ORIGIN', 'DEST', 'DEP_DEL15', 
df.head()
```

Out[13]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	DEP_DEL15	CRS_ARR_TIME	ARR_I
0	1399	1	1	5	ATL	SEA	0.0	2143	
1	1476	1	1	5	DTW	MSP	0.0	1435	
2	1597	1	1	5	ATL	SEA	0.0	1215	
3	1768	1	1	5	SEA	MSP	0.0	1335	
4	1823	1	1	5	SEA	DTW	0.0	607	

Handling Missing Values

Checking for null values

In [14]:
```
df.isnull().any()
```

Out[14]:
```
FL_NUM          False
MONTH           False
DAY_OF_MONTH    False
DAY_OF_WEEK     False
ORIGIN          False
DEST            False
DEP_DEL15        True
CRS_ARR_TIME    False
ARR_DEL15        True
dtype: bool
```

Replacing null values

In [15]:
```
df.fillna(df['DEP_DEL15'].mode()[0], inplace=True)
df.fillna(df['ARR_DEL15'].mode()[0], inplace=True)
```

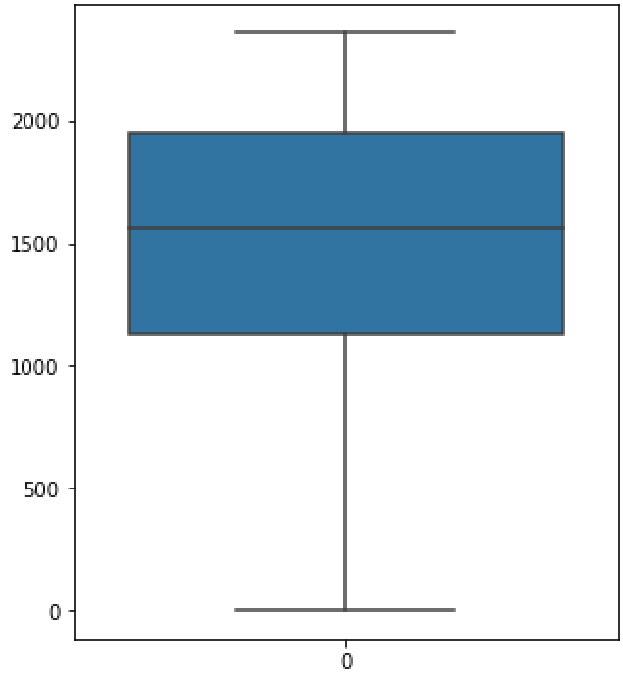
Checking if the replacement is made

In [16]:
```
df.isnull().any()
```

Out[16]:
```
FL_NUM          False
MONTH           False
DAY_OF_MONTH    False
DAY_OF_WEEK     False
ORIGIN          False
DEST            False
DEP_DEL15       False
CRS_ARR_TIME    False
ARR_DEL15       False
dtype: bool
```

Handling Outliers

In [17]:
```
fig, ax = plt.subplots(figsize=(5, 6))
sb.boxplot(data=df['CRS_ARR_TIME'])
plt.show()
```



There are no outliers

Encoding

One Hot Encoding

In [18]:
```
df = pd.get_dummies(df, columns=['ORIGIN', 'DEST'])
df.head()
```

Out[18]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	DEP_DEL15	CRS_ARR_TIME	ARR_DEL15	ORIGIN_A
0	1399	1	1	5	0.0	2143	0.0	
1	1476	1	1	5	0.0	1435	0.0	
2	1597	1	1	5	0.0	1215	0.0	
3	1768	1	1	5	0.0	1335	0.0	
4	1823	1	1	5	0.0	607	0.0	

In [19]:
```
df.columns
```

Out[19]:
```
Index(['FL_NUM', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK', 'DEP_DEL15',
       'CRS_ARR_TIME', 'ARR_DEL15', 'ORIGIN_ATL', 'ORIGIN_DTW', 'ORIGIN_JFK',
       'ORIGIN_MSP', 'ORIGIN_SEA', 'DEST_ATL', 'DEST_DTW', 'DEST_JFK',
       'DEST_MSP', 'DEST_SEA'],
      dtype='object')
```

Splitting dataset into Independent and Dependent Variables

In [20]:
```
X = df.drop(columns=['ARR_DEL15'])
Y = df[['ARR_DEL15']]
```

Scaling the Independent Variables

In [21]:
```
scale_crs = StandardScaler()
X[['CRS_ARR_TIME']] = scale_crs.fit_transform(X[['CRS_ARR_TIME']])
scale_flnum = StandardScaler()
X[['FL_NUM']] = scale_flnum.fit_transform(X[['FL_NUM']])
X.head()
```

Out[21]:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	DEP_DEL15	CRS_ARR_TIME	ORIGIN_ATL	ORIGIN_
0	0.079664	1	1	5	0.0	1.205371	1	
1	0.174510	1	1	5	0.0	-0.203612	0	
2	0.323555	1	1	5	0.0	-0.641431	1	
3	0.534188	1	1	5	0.0	-0.402620	0	
4	0.601935	1	1	5	0.0	-1.851405	0	

In [22]:
```
X.FL_NUM.value_counts()
```

Out[22]:
```
-0.549771    98
-0.918071    96
 0.808873    96
-0.919302    95
-0.532526    94
             ..
 1.865732     1
 0.242258     1
 0.195451     1
 0.212695     1
 1.608292     1
Name: FL_NUM, Length: 690, dtype: int64
```

Converting the Independent and Dependent Variables to 1D Arrays

In [23]:
```
X = X.values
Y = Y.values
```

Splitting dataset into Train and Test datasets

In [24]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

In [25]:
```
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[25]:
```
((8984, 16), (2247, 16), (8984, 1), (2247, 1))
```

Building the Machine Learning Models

Logistic Regression

In [26]:
```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(max_iter=200)
log_reg.fit(X_train, Y_train.ravel())
```

Out[26]:
```
LogisticRegression(max_iter=200)
```

Support Vector Machine (Classifier)

In [27]:
```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, Y_train.ravel())
```

Out[27]:
```
SVC()
```

KNN Classifier

In [28]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train.ravel())
```

Out[28]:
```
KNeighborsClassifier()
```

Random Forest Classifier

In [29]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=15, max_depth=3)
rf.fit(X_train, Y_train.ravel())
```

Out[29]:
```
RandomForestClassifier(max_depth=3, n_estimators=15)
```

Testing the Models

Logistic Regression

In [30]:
```
Y_pred_log_train = log_reg.predict(X_train)
Y_pred_log_test = log_reg.predict(X_test)
```

In [31]:
```
pd.DataFrame(Y_pred_log_train).value_counts()
```

Out[31]:
```
0.0    7692
1.0    1292
dtype: int64
```

In [32]:
```
pd.DataFrame(Y_pred_log_test).value_counts()
```

Out[32]:
```
0.0    1960
1.0     287
dtype: int64
```

Support Vector Machine (Classifier)

In [33]:
```
Y_pred_svc_train = svc.predict(X_train)
Y_pred_svc_test = svc.predict(X_test)
```

In [34]:
```
pd.DataFrame(Y_pred_svc_train).value_counts()
```

Out[34]:
```
0.0    7685
1.0    1299
dtype: int64
```

In [35]:
```
pd.DataFrame(Y_pred_svc_test).value_counts()
```
Out[35]:
```
0.0    1957
1.0     290
dtype: int64
```

KNN Classifier

In [36]:
```
Y_pred_knn_train = knn.predict(X_train)
Y_pred_knn_test = knn.predict(X_test)
```

In [37]:
```
pd.DataFrame(Y_pred_knn_train).value_counts()
```
Out[37]:
```
0.0    8586
1.0     398
dtype: int64
```

In [38]:
```
pd.DataFrame(Y_pred_knn_test).value_counts()
```
Out[38]:
```
0.0    2176
1.0      71
dtype: int64
```

Random Forest Classifier

In [39]:
```
Y_pred_rf_train = rf.predict(X_train)
Y_pred_rf_test = rf.predict(X_test)
```

In [40]:
```
pd.DataFrame(Y_pred_rf_train).value_counts()
```
Out[40]:
```
0.0    8976
1.0       8
dtype: int64
```

In [41]:
```
pd.DataFrame(Y_pred_rf_test).value_counts()
```
Out[41]:
```
0.0    2247
dtype: int64
```

Evaluating the ML Models using Metrics

Logistic Regression

Classification Report

In [42]:
```
print(classification_report(Y_test, Y_pred_log_test))
```
```
              precision    recall  f1-score   support

         0.0       0.96      0.94      0.95      1997
         1.0       0.61      0.70      0.65       250

    accuracy                           0.92      2247
   macro avg       0.78      0.82      0.80      2247
weighted avg       0.92      0.92      0.92      2247
```

Accuracy, Precision, Recall, F1 Score

In [43]:
```
acc_log = accuracy_score(Y_test, Y_pred_log_test)
prec_log, rec_log, f1_log, sup_log = precision_recall_fscore_support(Y_test, Y_pred_log_te
```

```
print('Accuracy Score =', acc_log)
print('Precision =', prec_log[0])
print('Recall =', rec_log[0])
print('F1 Score =', f1_log[0])
```

```
Accuracy Score = 0.9158878504672897
Precision = 0.9612244897959183
Recall = 0.943415122684026
F1 Score = 0.9522365428354814
```

Checking for Overfitting and Underfitting

In [44]:
```
log_train_acc = accuracy_score(Y_train, Y_pred_log_train)
log_test_acc = accuracy_score(Y_test, Y_pred_log_test)
print('Training Accuracy =', log_train_acc)
print('Testing Accuracy =', log_test_acc)
```

```
Training Accuracy = 0.9213045414069457
Testing Accuracy = 0.9158878504672897
```

There is no big variation in the training and testing accuracy. Therefore, the Logistic Regression model is not overfit or underfit.

Confusion Matrix

In [45]:
```
pd.crosstab(Y_test.ravel(), Y_pred_log_test)
```

Out[45]:

col_0	0.0	1.0
row_0		
0.0	1884	113
1.0	76	174

Support Vector Machine (Classifier)

Classification Report

In [46]:
```
print(classification_report(Y_test, Y_pred_svc_test))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.94      0.95      1997
         1.0       0.61      0.70      0.65       250

    accuracy                           0.92      2247
   macro avg       0.78      0.82      0.80      2247
weighted avg       0.92      0.92      0.92      2247
```

Accuracy, Precision, Recall, F1 Score

In [47]:
```
acc_svc = accuracy_score(Y_test, Y_pred_svc_test)
prec_svc, rec_svc, f1_svc, sup_svc = precision_recall_fscore_support(Y_test, Y_pred_svc_te
print('Accuracy Score =', acc_svc)
print('Precision =', prec_svc[0])
print('Recall =', rec_svc[0])
print('F1 Score =', f1_svc[0])
```

```
Accuracy Score = 0.9163328882955051
Precision = 0.9621870209504343
Recall = 0.942914371557336
F1 Score = 0.9524532119372786
```

Checking for Overfitting and Underfitting

In [48]:
```
svc_train_acc = accuracy_score(Y_train, Y_pred_svc_train)
svc_test_acc = accuracy_score(Y_test, Y_pred_svc_test)
print('Training Accuracy =', svc_train_acc)
print('Testing Accuracy =', svc_test_acc)
```

```
Training Accuracy = 0.9214158504007124
Testing Accuracy = 0.9163328882955051
```

There is no big variation in the training and testing accuracy. Therefore, the Support Vector Classifier model is not overfit or underfit.

Confusion Matrix

In [49]:
```
pd.crosstab(Y_test.ravel(), Y_pred_svc_test)
```

Out[49]:

col_0	0.0	1.0
row_0		
0.0	1883	114
1.0	74	176

KNN Classifier

Classification Report

In [50]:
```
print(classification_report(Y_test, Y_pred_knn_test))
```

```
              precision    recall  f1-score   support

         0.0       0.90      0.98      0.94      1997
         1.0       0.52      0.15      0.23       250

    accuracy                           0.89      2247
   macro avg       0.71      0.57      0.59      2247
weighted avg       0.86      0.89      0.86      2247
```

Accuracy, Precision, Recall, F1 Score

In [51]:
```
acc_knn = accuracy_score(Y_test, Y_pred_knn_test)
prec_knn, rec_knn, f1_knn, sup_knn = precision_recall_fscore_support(Y_test, Y_pred_knn_te
print('Accuracy Score =', acc_knn)
print('Precision =', prec_knn[0])
print('Recall =', rec_knn[0])
print('F1 Score =', f1_knn[0])
```

```
Accuracy Score = 0.8900756564307967
Precision = 0.9021139705882353
Recall = 0.9829744616925388
F1 Score = 0.940809968847352
```

Checking for Overfitting and Underfitting

In [52]:
```
knn_train_acc = accuracy_score(Y_train, Y_pred_knn_train)
```

```
knn_test_acc = accuracy_score(Y_test, Y_pred_knn_test)
print('Training Accuracy =', knn_train_acc)
print('Testing Accuracy =', knn_test_acc)
```

```
Training Accuracy = 0.9052760463045414
Testing Accuracy = 0.8900756564307967
```

There is no big variation in the training and testing accuracy. Therefore, the KNN Classifier model is not overfit or underfit.

Confusion Matrix

In [53]:
```
pd.crosstab(Y_test.ravel(), Y_pred_knn_test)
```

Out[53]:

col_0	0.0	1.0
row_0		
0.0	1963	34
1.0	213	37

Random Forest Classifier

Classification Report

In [54]:
```
print(classification_report(Y_test, Y_pred_rf_test))
```

```
              precision    recall  f1-score   support

         0.0       0.89      1.00      0.94      1997
         1.0       0.00      0.00      0.00       250

    accuracy                           0.89      2247
   macro avg       0.44      0.50      0.47      2247
weighted avg       0.79      0.89      0.84      2247
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/metrics/_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/metrics/_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/metrics/_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

Accuracy, Precision, Recall, F1 Score

In [55]:
```
acc_rf = accuracy_score(Y_test, Y_pred_rf_test)
prec_rf, rec_rf, f1_rf, sup_rf = precision_recall_fscore_support(Y_test, Y_pred_rf_test)
print('Accuracy Score =', acc_rf)
print('Precision =', prec_rf[0])
print('Recall =', rec_rf[0])
print('F1 Score =', f1_rf[0])
```

```
Accuracy Score = 0.8887405429461505
Precision = 0.8887405429461505
Recall = 1.0
F1 Score = 0.9410933081998115
```

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/metrics/_classification.py:
1318: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

Checking for Overfitting and Underfitting

In [56]:
```
rf_train_acc = accuracy_score(Y_train, Y_pred_rf_train)
rf_test_acc = accuracy_score(Y_test, Y_pred_rf_test)
print('Training Accuracy =', rf_train_acc)
print('Testing Accuracy =', rf_test_acc)
```

```
Training Accuracy = 0.8756678539626002
Testing Accuracy = 0.8887405429461505
```

There is no big variation in the training and testing accuracy. Therefore, the Random Forest Classifier model is not overfit or underfit.

Confusion Matrix

In [57]:
```
pd.crosstab(Y_test.ravel(), Y_pred_rf_test)
```

Out[57]:

col_0	**0.0**
row_0	
0.0	1997
1.0	250

On comparing the four models built, based on the performance metrics it is clear that Support Vector Classifier gives the highest performance. Hence, that model is chosen for deployment

Dumping the Chosen Model into pkl file

In [58]:
```
joblib.dump(svc, 'model.pkl')
```

Out[58]:
```
['model.pkl']
```

Dumping the Scaling Modules into pkl file

In [59]:
```
pickle.dump(scale_crs, open('crs_scale.pkl', 'wb'))
pickle.dump(scale_flnum, open('flnum_scale.pkl', 'wb'))
```

In [60]:
```
!pip install -U ibm-watson-machine-learning
```

```
Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.257)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.3.4)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.3.3)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.8.9)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.26.7)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (21.3)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2022.9.24)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (4.8.2)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.26.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (0.10.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (1.20.3)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (1.15.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (2.0.4)
Requirement already satisfied: zipp>=0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from importlib-metadata->ibm-watson-machine-learning) (3.6.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm-watson-machine-learning) (3.0.4)
```

In [61]:
```
from ibm_watson_machine_learning import APIClient
import json
```

In [62]:
```
wml_credentials = {
    "apikey":"G0-fTz5hk2xE6cxkk-PyyzQmio8NNqeMG_wTLTgCSqmU",
    "url":"https://us-south.ml.cloud.ibm.com"
}
```

In [63]:
```
wml_clients = APIClient(wml_credentials)
```

In [64]:
```
wml_clients.spaces.list()
```

```
Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
------------------------------------  ------  ------------------------
ID                                    NAME    CREATED
267b8ab3-7a6f-4efa-81fb-bc602a9f29e2  flight  2022-11-19T09:48:15.358Z
------------------------------------  ------  ------------------------
```

In [65]:
```
space_id = "267b8ab3-7a6f-4efa-81fb-bc602a9f29e2"
```

In [66]:
```
wml_clients.set.default_space(space_id)
```

Out[66]:
```
'SUCCESS'
```

In [67]:
```
wml_clients.software_specifications.list(500)
```

```
------------------------------  ------------------------------------  ----
NAME                            ASSET_ID                              TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6           0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                      0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6               10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-bf776828c4b7  base
autoai-kb_rt22.2-py3.10         125b6d9a-5b1f-5e8d-972a-b251688ccf40  base
runtime-22.1-py3.9              12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6         154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                    1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6          1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
kernel-spark3.3-r3.6            1c9e5454-f216-59dd-a20e-474a5cdf5988  base
pytorch-onnx_rt22.1-py3.9-edt   1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6            1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
spark-mllib_3.2                 20047f72-0a98-58c7-9ff5-a77b012eb8f5  base
tensorflow_2.4-py3.8-horovod    217c16f6-178f-56bf-824a-b19f20564c49  base
runtime-22.1-py3.9-cuda         26215f05-08c3-5a41-a1b0-da66306ce658  base
do_py3.8                        295addb5-9ef9-547e-9bf4-92ae3563e720  base
autoai-ts_3.8-py3.8             2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6           2b73a275-7cbf-420b-a912-eae7f436e0bc  base
kernel-spark3.3-py3.9           2b7961e2-e3b1-5a8c-a491-482c8368839a  base
pytorch_1.2-py3.6               2c8ef57d-2687-4b7d-acce-01f94976dac1  base
spark-mllib_2.3                 2e51f700-bca0-4b0d-88dc-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt      32983cea-3f32-4400-8965-dde874a8d67e  base
spark-mllib_3.0-py37            36507ebe-8770-55ba-ab2a-eafe787600e9  base
spark-mllib_2.4                 390d21f8-e58b-4fac-9c55-d7ceda621326  base
autoai-ts_rt22.2-py3.10         396b2e83-0953-5b86-9a55-7ce1628a406f  base
xgboost_0.82-py3.6              39e31acd-5f30-41dc-ae44-60233c80306e  base
pytorch-onnx_1.2-py3.6-edt      40589d0e-7019-4e28-8daa-fb03b6f4fe12  base
pytorch-onnx_rt22.2-py3.10      40e73f55-783a-5535-b3fa-0c8b94291431  base
default_r36py38                 41c247d3-45f8-5a71-b065-8580229facf0  base
autoai-ts_rt22.1-py3.9          4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
autoai-obm_3.0                  42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                    493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6           49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6              4ff8d6c2-1343-4c18-85e1-689c965304d3  base
pytorch-onnx_1.1-py3.6          50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
autoai-ts_3.9-py3.8             52c57136-80fa-572e-8728-a5e7cbb42cde  base
spark-mllib_2.4-scala_2.11      55a70f99-7320-4be5-9fb9-9edb5a443af5  base
spark-mllib_3.0                 5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base
autoai-obm_2.0                  5c2e37fa-80b8-5e77-840f-d912469614ee  base
spss-modeler_18.1               5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                      5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
runtime-22.2-py3.10-xc          5e8cddff-db4a-5a6a-b8aa-2d4af9864dab  base
autoai-kb_3.1-py3.7             632d4b22-10aa-5180-88f0-f52dfb6444d7  base
pytorch-onnx_1.7-py3.8          634d3cdc-b562-5bf9-a2d4-ea90a478456b  base
spark-mllib_2.3-r_3.6           6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c  base
tensorflow_2.4-py3.7            65e171d7-72d1-55d9-8ebb-f813d620c9bb  base
spss-modeler_18.2               687eddc9-028a-4117-b9dd-e57b36f1efa5  base
pytorch-onnx_1.2-py3.6          692a6a4d-2c4d-45ff-a1ed-b167ee55469a  base
spark-mllib_2.3-scala_2.11      7963efe5-bbec-417e-92cf-0574e21b4e8d  base
spark-mllib_2.4-py37            7abc992b-b685-532b-a122-a396a3cdbaab  base
caffe_1.0-py3.6                 7bb3dbe2-da6e-4145-918d-b6d84aa93b6b  base
pytorch-onnx_1.7-py3.7          812c6631-42b7-5613-982b-02098e6c909c  base
cuda-py3.6                      82c79ece-4d12-40e6-8787-a7b9e0f62770  base
tensorflow_1.15-py3.6-horovod   8964680e-d5e4-5bb8-919b-8342c6c0dfd8  base
hybrid_0.1                      8c1a58c6-62b5-4dc4-987a-df751c2756b6  base
pytorch-onnx_1.3-py3.7          8d5d8a87-a912-54cf-81ec-3914adaa988d  base
caffe-ibm_1.0-py3.6             8d863266-7927-4d1e-97d7-56a7f4c0a19b  base
```

```
runtime-22.2-py3.10-cuda          8ef391e4-ef58-5d46-b078-a82c211c1058   base
spss-modeler_17.1                 902d0051-84bd-4af6-ab6b-8f6aa6fdeabb   base
do_12.10                          9100fd72-8159-4eb9-8a0b-a87e12eefa36   base
do_py3.7                          9447fa8b-2051-4d24-9eef-5acb0e3c59f8   base
spark-mllib_3.0-r_3.6             94bb6052-c837-589d-83f1-f4142f219e32   base
cuda-py3.7-opence                 94e9652b-7f2d-59d5-ba5a-23a414ea488f   base
nlp-py3.8                         96e60351-99d4-5a1c-9cc0-473ac1b5a864   base
cuda-py3.7                        9a44990c-1aa1-4c7d-baf8-c4099011741c   base
hybrid_0.2                        9b3f9040-9cee-4ead-8d7a-780600f542f7   base
spark-mllib_3.0-py38              9f7a8fc1-4d3c-5e65-ab90-41fa8de2d418   base
autoai-kb_3.3-py3.7               a545cca3-02df-5c61-9e88-998b09dc79af   base
spark-mllib_3.0-py39              a6082a27-5acc-5163-b02c-6b96916eb5e0   base
runtime-22.1-py3.9-do             a7e7dbf1-1d03-5544-994d-e5ec845ce99a   base
default_py3.8                     ab9e1b80-f2ce-592c-a7d2-4f2344f77194   base
tensorflow_rt22.1-py3.9           acd9c798-6974-5d2f-a657-ce06e986df4d   base
kernel-spark3.2-py3.9             ad7033ee-794e-58cf-812e-a95f4b64b207   base
autoai-obm_2.0 with Spark 3.0     af10f35f-69fa-5d66-9bf5-acb58434263a   base
runtime-22.2-py3.10               b56101f1-309d-549b-a849-eaa63f77b2fb   base
default_py3.7_opence              c2057dd4-f42c-5f77-a02f-72bdbd3282c9   base
tensorflow_2.1-py3.7              c4032338-2a40-500a-beef-b01ab2667e27   base
do_py3.7_opence                   cc8f8976-b74a-551a-bb66-6377f8d865b4   base
spark-mllib_3.3                   d11f2434-4fc7-58b7-8a62-755da64fdaf8   base
autoai-kb_3.0-py3.6               d139f196-e04b-5d8b-9140-9a10ca1fa91a   base
spark-mllib_3.0-py36              d82546d5-dd78-5fbb-9131-2ec309bc56ed   base
autoai-kb_3.4-py3.8               da9b39c3-758c-5a4f-9cfd-457dd4d8c395   base
kernel-spark3.2-r3.6              db2fe4d6-d641-5d05-9972-73c654c60e0a   base
autoai-kb_rt22.1-py3.9            db6afe93-665f-5910-b117-d879897404d9   base
tensorflow_rt22.1-py3.9-horovod   dda170cc-ca67-5da7-9b7a-cf84c6987fae   base
autoai-ts_1.0-py3.7               deef04f0-0c42-5147-9711-89f9904299db   base
tensorflow_2.1-py3.7-horovod      e384fce5-fdd1-53f8-bc71-11326c9c635f   base
default_py3.7                     e4429883-c883-42b6-87a8-f419d64088cd   base
do_22.1                           e51999ba-6452-5f1f-8287-17228b88b652   base
autoai-obm_3.2                    eae86aab-da30-5229-a6a6-1d0d4e368983   base
runtime-22.2-r4.2                 ec0a3d28-08f7-556c-9674-ca7c2dba30bd   base
tensorflow_rt22.2-py3.10          f65bd165-f057-55de-b5cb-f97cf2c0f393   base
do_20.1                           f686cdd9-7904-5f9d-a732-01b0d6b10dc5   base
pytorch-onnx_rt22.2-py3.10-edt    f8a05d07-e7cd-57bb-a10b-23f1d4b837ac   base
scikit-learn_0.19-py3.6           f963fa9d-4bb7-5652-9c5d-8d9289ef6ad9   base
tensorflow_2.4-py3.8              fe185c44-9a99-5425-986b-59bd1d2eda46   base
-------------------------------   ------------------------------------   ----
```

In [68]:
```python
import sklearn
sklearn.__version__
'1.0.2'
```

Out[68]:
```
'1.0.2'
```

In [69]:
```python
MODEL_NAME="supportvectormachine"
DEPLOYMENT_NAME="svc__deployment"
DEMO_MODEL=svc
```

In [70]:
```python
soft_sepc_id=wml_clients.software_specifications.get_id_by_name("runtime-22.1-py3.9")
```

In [71]:
```python
model_props={
    wml_clients.repository.ModelMetaNames.NAME:MODEL_NAME,
    wml_clients.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",
    wml_clients.repository.ModelMetaNames.SOFTWARE_SPEC_UID: soft_sepc_id

}
```

In [72]:
```python
model_details=wml_clients.repository.store_model(model=DEMO_MODEL,meta_props=model_props,t
                                                  training_target=Y_train.ravel())
```

In [73]:
```python
model_details
```

Out[73]:


```
{'entity': {'hybrid_pipeline_software_specs': [],
  'label_column': 'l1',
  'schemas': {'input': [{'fields': [{'name': 'f0', 'type': 'float'},
      {'name': 'f1', 'type': 'float'},
      {'name': 'f2', 'type': 'float'},
      {'name': 'f3', 'type': 'float'},
      {'name': 'f4', 'type': 'float'},
      {'name': 'f5', 'type': 'float'},
      {'name': 'f6', 'type': 'float'},
      {'name': 'f7', 'type': 'float'},
      {'name': 'f8', 'type': 'float'},
      {'name': 'f9', 'type': 'float'},
      {'name': 'f10', 'type': 'float'},
      {'name': 'f11', 'type': 'float'},
      {'name': 'f12', 'type': 'float'},
      {'name': 'f13', 'type': 'float'},
      {'name': 'f14', 'type': 'float'},
      {'name': 'f15', 'type': 'float'}],
     'id': '1',
     'type': 'struct'}],
   'output': []},
  'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',
   'name': 'runtime-22.1-py3.9'},
  'type': 'scikit-learn_1.0'},
 'metadata': {'created_at': '2022-11-19T19:28:04.765Z',
  'id': 'e3e793cc-079e-4363-9775-0a67bb3665b3',
  'modified_at': '2022-11-19T19:28:07.620Z',
  'name': 'supportvectormachine',
  'owner': 'IBMid-661003XLH3',
  'resource_key': '2a8d6e41-3599-4d66-aa9d-aaa87b678c20',
  'space_id': '267b8ab3-7a6f-4efa-81fb-bc602a9f29e2'},
 'system': {'warnings': []}}
```


In [74]:

```
model_id = wml_clients.repository.get_model_id(model_details)
model_id
```

Out[74]:

```
'e3e793cc-079e-4363-9775-0a67bb3665b3'
```

In [75]:

```
deployment_props = {
    wml_clients.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_clients.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

In [76]:

```
deployment = wml_clients.deployments.create(
    artifact_uid=model_id,
    meta_props=deployment_props
)
```

```
#######################################################################################

Synchronous deployment creation for uid: 'e3e793cc-079e-4363-9775-0a67bb3665b3' started

#######################################################################################


initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls i
nstead.

ready


------------------------------------------------------------------------------------------
------
Successfully finished deployment creation, deployment_uid='486b7d6a-343e-4048-bee1-e73454d
27219'
------------------------------------------------------------------------------------------
------
```

In [ ]: