This notebook contains the ML Model for predicting flight delay

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

import necessary libraries

```
!pip install catboost
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting catboost
  Downloading catboost-1.2-cp310-cp310-manylinux2014_x86_64.whl (98.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.6/98.6 MB 8.4 MB/s eta 0:00:00
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.10.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.22.4)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.5.3)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.13.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.0.7)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.39.3)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (8.4.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.4)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.2)
Installing collected packages: catboost
Successfully installed catboost-1.2
```

```
import pandas as pd
from sklearn.inspection import permutation_importance
from matplotlib import pyplot as plt
#from matplotlib import pyplot
import numpy as np
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from catboost import CatBoostClassifier, Pool
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.exceptions import DataConversionWarning
```

```
df=pd.read_csv('/content/drive/MyDrive/ML Project/Result.csv')
```

```
df
```

	Unnamed: 0	Unnamed: 0_x	Date (MM/DD/YYYY)	Flight Number	Destination Airport	Scheduled elapsed time	Actual elapsed time	Departure delay	Wheels-off	Taxi-Out time	...	winddir	seal

```
df.columns
```

```
Index(['Unnamed: 0', 'Unnamed: 0_x', 'Date (MM/DD/YYYY)', 'Flight Number',
       'Destination Airport', 'Scheduled elapsed time (Minutes)',
       'Actual elapsed time (Minutes)', 'Departure delay (Minutes)',
       'Wheels-off time', 'Taxi-Out time (Minutes)',
       'Delay Carrier (Minutes)_x', 'Delay Weather (Minutes)_x',
       'Delay National Aviation System (Minutes)_x',
       'Delay Security (Minutes)_x', 'Delay Late Aircraft Arrival (Minutes)_x',
       'Origin Airport', 'Arrival Delay (Minutes)', 'Wheels-on Time',
       'Taxi-In time (Minutes)', 'month', 'day', 'year', 'Unnamed: 0_y',
       'tempmax', 'tempmin', 'temp', 'feelslikemax', 'feelslikemin',
       'feelslike', 'dew', 'humidity', 'precip', 'precipprob', 'precipcover',
       'snow', 'snowdepth', 'windgust', 'windspeed', 'winddir',
       'sealevelpressure', 'cloudcover', 'visibility', 'solarradiation',
       'solarenergy', 'uvindex', 'severerisk', 'moonphase', 'icon'],
      dtype='object')
```

1011	1011	1020	10/28/22	2198.0	0	71.0	63.0	-13.0	22:25	18.0	...	30.9

drop unecessary columns

1012 rows x 48 columns

```
df=df.drop(columns=['Unnamed: 0', 'Unnamed: 0_x','Unnamed: 0_y'])
```

Rename the Delay column to Target

```
df1 = df.rename(columns={'Arrival Delay (Minutes)':'TARGET'})
```

```
target = pd.cut(df1.TARGET,bins=[-500,-10,10,30,1000],labels=['0','1','2','3'])
```

```
df1.insert(45,'TARGET1',target)
```

```
df1.columns
```

```
Index(['Date (MM/DD/YYYY)', 'Flight Number', 'Destination Airport',
       'Scheduled elapsed time (Minutes)', 'Actual elapsed time (Minutes)',
       'Departure delay (Minutes)', 'Wheels-off time',
       'Taxi-Out time (Minutes)', 'Delay Carrier (Minutes)_x',
       'Delay Weather (Minutes)_x',
       'Delay National Aviation System (Minutes)_x',
       'Delay Security (Minutes)_x', 'Delay Late Aircraft Arrival (Minutes)_x',
       'Origin Airport', 'TARGET', 'Wheels-on Time', 'Taxi-In time (Minutes)',
       'month', 'day', 'year', 'tempmax', 'tempmin', 'temp', 'feelslikemax',
       'feelslikemin', 'feelslike', 'dew', 'humidity', 'precip', 'precipprob',
       'precipcover', 'snow', 'snowdepth', 'windgust', 'windspeed', 'winddir',
       'sealevelpressure', 'cloudcover', 'visibility', 'solarradiation',
       'solarenergy', 'uvindex', 'severerisk', 'moonphase', 'icon', 'TARGET1'],
      dtype='object')
```

Define Features and Labels for training testing data

```
X1=df1[['Origin Airport', 'tempmax', 'tempmin', 'temp',
       'feelslikemax', 'feelslikemin', 'feelslike', 'dew', 'humidity',
       'precip', 'precipprob', 'precipcover', 'snow', 'snowdepth', 'windgust',
       'windspeed', 'winddir', 'sealevelpressure', 'cloudcover', 'visibility',
       'solarradiation', 'solarenergy', 'uvindex', 'severerisk', 'moonphase',
       'icon', 'day', 'month', 'year', 'Flight Number']]

y = df1['TARGET1']
```

```
X1_train, X1_test, y_train, y_test = train_test_split(X1,y, test_size=0.2, random_state=20)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X1_train = pd.DataFrame(sc.fit_transform(X1_train), columns = X1_train.columns, index = X1_train.index)
X1_test = pd.DataFrame(sc.transform(X1_test), columns = X1_test.columns, index = X1_test.index)
```

```
X1_train
```

	Origin Airport	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	precip	...	solarradiati
206	0.982572	0.949510	1.590662	1.272104	1.312042	1.460623	1.322399	1.738199	1.313543	0.255621	...	-1.1682
893	0.196126	0.780303	1.386212	1.086312	0.786974	1.293867	1.039039	1.347958	0.720380	0.640197	...	-0.9985
150	0.982572	0.667498	1.230440	0.959191	0.651473	1.166814	0.922361	1.443139	1.332678	1.045286	...	-0.8301
895	0.196126	0.930709	0.918897	1.017862	0.914007	0.912709	0.980700	1.081452	0.190997	-0.374755	...	-0.5304
438	-1.376767	-0.723761	-1.008778	-0.849840	-0.601914	-0.961315	-0.852807	-1.041077	-0.746585	-0.374755	...	1.6184
...	...	...	...	...	...	...	...	...	...	...	...	
924	0.196126	1.297325	1.434891	1.409004	1.549169	1.333571	1.439077	1.623982	0.694867	0.503740	...	-0.9711
223	0.982572	0.789703	0.918897	0.900520	0.710755	0.912709	0.864022	0.843501	0.006032	-0.374755	...	-0.2279
271	0.982572	0.827305	0.782597	0.841848	0.803912	0.801538	0.830686	1.052898	0.592817	-0.374755	...	-0.5728
474	-1.376767	0.554693	-0.658292	0.088900	0.549847	-0.572217	0.097283	-1.060113	-1.607629	-0.374755	...	2.0372

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=20).fit(X1_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
lr.score(X1_train,y_train)
```

```
0.49135802469135803
```

```
lr.score(X1_test,y_test)
```

```
0.458128078817734
```

```
importance = lr.coef_[0]
# summarize feature importance
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

```
Feature: 0, Score: 0.52421
Feature: 1, Score: -0.09852
Feature: 2, Score: -0.32440
Feature: 3, Score: -0.27824
Feature: 4, Score: 0.42868
Feature: 5, Score: 0.15797
Feature: 6, Score: 0.23195
Feature: 7, Score: -0.12714
Feature: 8, Score: -0.06830
Feature: 9, Score: -0.05962
Feature: 10, Score: 0.00200
Feature: 11, Score: 0.01525
Feature: 12, Score: 0.13324
Feature: 13, Score: -0.02470
Feature: 14, Score: 0.08387
Feature: 15, Score: -0.07289
Feature: 16, Score: -0.08252
Feature: 17, Score: 0.12474
Feature: 18, Score: 0.16798
Feature: 19, Score: 0.01411
Feature: 20, Score: 0.16845
```

```
test_output = pd.DataFrame(lr.predict(X1_test), index = X1_test.index, columns = ['pred'])
```

```
test_output
```

	pred
320	0
346	0
832	1
471	1
935	1
...	...
36	0
137	0
132	0
677	1
296	0

203 rows × 1 columns

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

	pred	TARGET1
320	0	0
346	0	2
832	1	1
471	1	1
935	1	3
...	...	...
36	0	1
137	0	1
132	0	0
677	1	1
296	0	0

203 rows × 2 columns

```
y_pred1=lr.predict(X1_test)
```

```
from sklearn.metrics import *
```

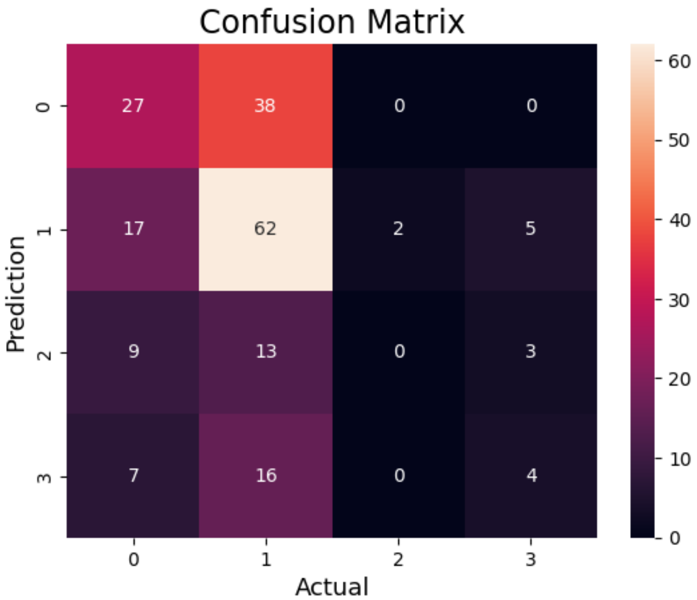
```
cm1 = confusion_matrix(y_test, y_pred1)
print('Classification Report:\n',classification_report(y_test, y_pred1))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.42      0.43        65
           1       0.48      0.72      0.58        86
           2       0.00      0.00      0.00        25
           3       0.33      0.15      0.21        27

    accuracy                           0.46       203
   macro avg       0.32      0.32      0.30       203
weighted avg       0.39      0.46      0.41       203
```

```
sns.heatmap(cm1,
            annot=True,
            fmt='g')
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```




```
catboost = CatBoostClassifier(random_state=20)
catboost.fit(X1_train, y_train, verbose=False)
```

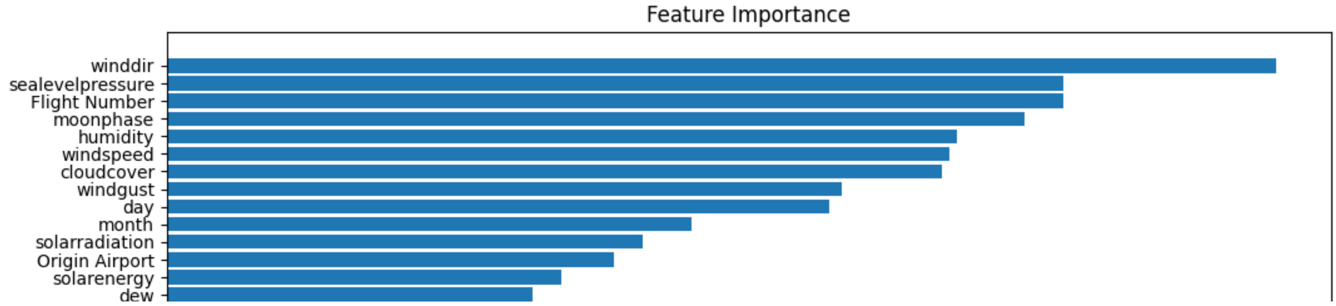
```
<catboost.core.CatBoostClassifier at 0x7f7494535fc0>
```

```
catboost.score(X1_test , y_test)
```

```
0.42857142857142855
```

```
feature_importance = catboost.feature_importances_
sorted_idx = np.argsort(feature_importance)
fig = plt.figure(figsize=(12, 6))
plt.barh(range(len(sorted_idx)), feature_importance[sorted_idx], align='center')
plt.yticks(range(len(sorted_idx)), np.array(X1_test.columns)[sorted_idx])
plt.title('Feature Importance')
```

```
Text(0.5, 1.0, 'Feature Importance')
```



Double-click (or enter) to edit


```
y_pred2=catboost.predict(X1_test)
```

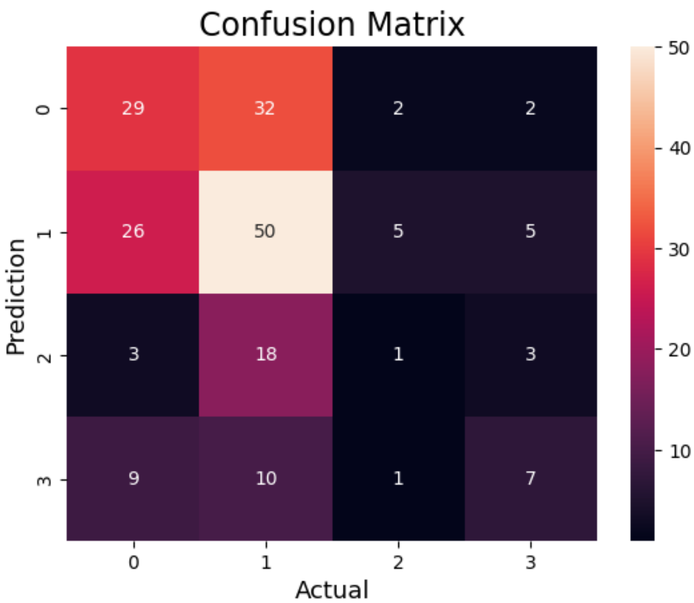

```
cm2 = confusion_matrix(y_test, y_pred2)
print('Classification Report:\n',classification_report(y_test, y_pred2))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.43      0.45      0.44        65
           1       0.45      0.58      0.51        86
           2       0.11      0.04      0.06        25
           3       0.41      0.26      0.32        27

    accuracy                           0.43       203
   macro avg       0.35      0.33      0.33       203
weighted avg       0.40      0.43      0.41       203
```

```
sns.heatmap(cm2,
            annot=True,
            fmt='g')
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```




```
rf = RandomForestClassifier(random_state=10)
rf.fit(X1_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=10)
```

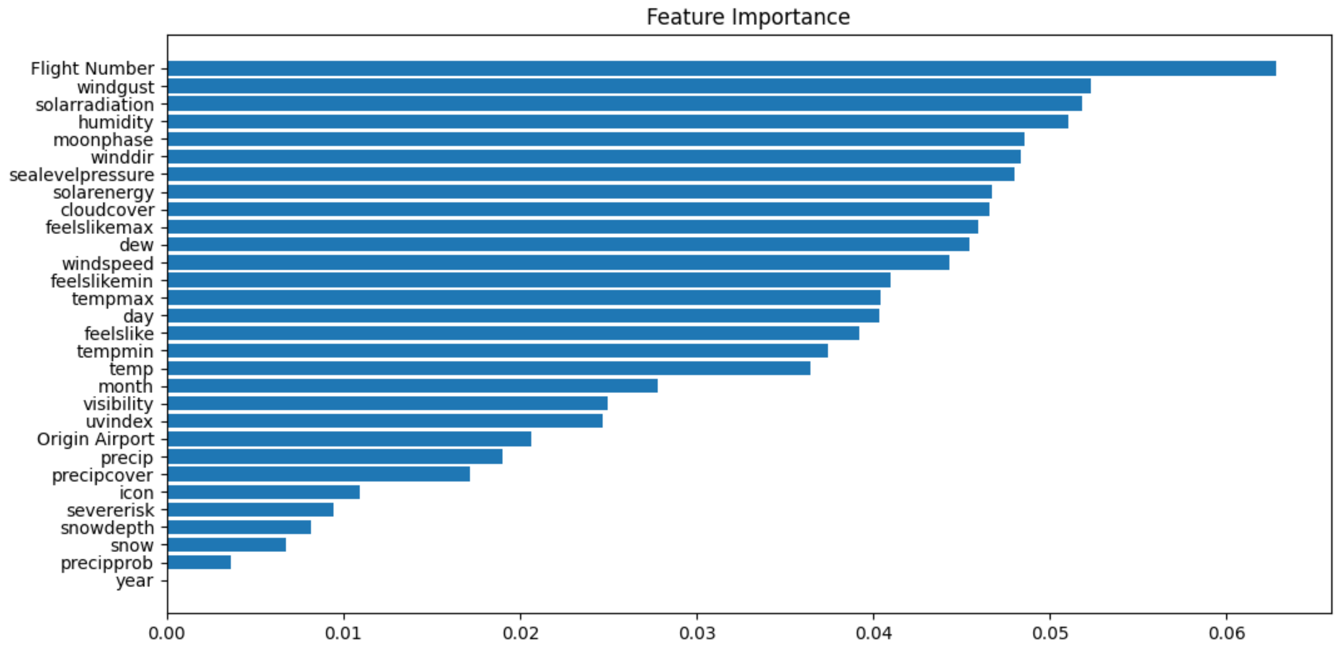
```
rf.score(X1_test,y_test)
```

```
0.43349753694581283
```

```
y_pred3=rf.predict(X1_test)
```

```
feature_importance = rf.feature_importances_
sorted_idx = np.argsort(feature_importance)
fig = plt.figure(figsize=(12, 6))
plt.barh(range(len(sorted_idx)), feature_importance[sorted_idx], align='center')
plt.yticks(range(len(sorted_idx)), np.array(X1_test.columns)[sorted_idx])
plt.title('Feature Importance')
```

```
Text(0.5, 1.0, 'Feature Importance')
```



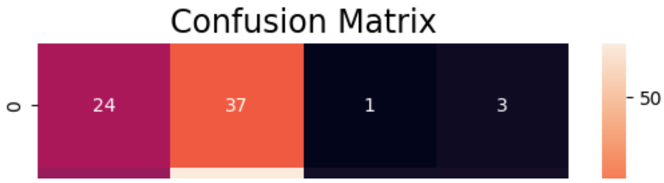

```
cm3 = confusion_matrix(y_test, y_pred3)
print('Classification Report:\n',classification_report(y_test, y_pred3))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.41      0.37      0.39        65
           1       0.46      0.65      0.54        86
           2       0.33      0.12      0.18        25
           3       0.33      0.19      0.24        27

    accuracy                           0.43       203
   macro avg       0.39      0.33      0.34       203
weighted avg       0.41      0.43      0.41       203
```

```
sns.heatmap(cm3,
            annot=True,
            fmt='g')
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```




```
##FORECASTING
```

```
f1=pd.read_csv('/content/drive/MyDrive/ML Project/WeatherDataF21-24.csv')
```

```
f1
```

	Unnamed: 0	Origin Airport	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	...	visibility	solarradiatio
0	0	ORD	16.6	7.1	11.2	16.6	5.4	10.2	2.0	55.1	...	16.0	115.
1	1	ORD	8.7	3.9	5.9	6.6	-1.2	2.6	-1.2	60.5	...	15.8	69.
2	2	ORD	5.7	2.3	4.3	3.7	-1.5	1.2	-2.2	63.2	...	16.0	35.
3	3	ORD	11.8	2.2	7.1	11.8	-0.8	5.2	-3.8	47.6	...	16.0	91.
4	0	DEN	8.8	-3.4	2.7	6.2	-7.4	-0.7	-12.5	36.0	...	14.7	127.
5	1	DEN	1.4	-5.2	-1.7	1.3	-10.3	-5.5	-4.4	82.5	...	9.5	105.
6	2	DEN	12.6	-3.6	4.3	12.6	-7.3	2.3	-4.8	57.0	...	15.9	295.
7	3	DEN	16.5	2.6	9.7	16.5	-0.6	8.6	-4.2	41.7	...	15.9	266.
8	0	EWR	20.0	10.7	14.9	20.0	10.7	14.9	8.9	68.2	...	16.0	255.
9	1	EWR	20.5	11.6	15.5	20.5	11.6	15.5	11.8	79.3	...	13.4	156.
10	2	EWR	20.4	12.7	16.2	20.4	12.7	16.2	8.5	65.6	...	13.1	189.
11	3	EWR	16.6	7.7	12.3	16.6	5.0	11.6	-0.1	45.3	...	16.0	240.
12	0	IAD	31.1	14.1	22.0	29.3	14.1	21.8	12.9	60.6	...	15.4	221.
13	1	IAD	22.8	14.8	18.9	22.8	14.8	18.9	12.5	68.5	...	15.3	60.
14	2	IAD	17.8	9.9	14.2	17.8	6.7	14.1	5.3	57.2	...	16.0	117.
15	3	IAD	16.0	8.1	11.6	16.0	5.5	10.8	-1.3	41.5	...	16.0	205.

16 rows × 30 columns

```
f1['day'] = f1['day'].astype(str)
f1['month'] = f1['month'].astype(str)
f1['year'] = f1['year'].astype(str)
```

```
f1.dtypes
```

```
Unnamed: 0          int64
Origin Airport     object
tempmax           float64
tempmin           float64
temp              float64
feelslikemax      float64
feelslikemin      float64
feelslike         float64
dew               float64
humidity          float64
precip            float64
```

```
precipprob            int64
precipcover         float64
snow                float64
snowdepth           float64
windgust            float64
windspeed           float64
winddir             float64
sealevelpressure    float64
cloudcover          float64
visibility          float64
solarradiation      float64
solarenergy         float64
uvindex               int64
severerisk            int64
moonphase           float64
icon                 object
day                  object
month                object
year                 object
dtype: object
```

```
f2=pd.read_csv('/content/drive/MyDrive/ML Project/project csv(Apr 21-24).csv')
```

```
f2
```

	Date	Day	Origin Airport	Flight Number	Arrival Time	Status (Early, On-time, Late, Severly Late)
0	4/21/23	Friday	ORD	3839	10:00 AM	NaN
1	4/21/23	Friday	ORD	3524	4:50 PM	NaN
2	4/21/23	Friday	ORD	538	9:34 PM	NaN
3	4/22/23	Saturday	ORD	3839	10:00 AM	NaN
4	4/22/23	Saturday	ORD	3524	4:50 PM	NaN
5	4/22/23	Saturday	ORD	538	9:34 PM	NaN
6	4/23/23	Sunday	ORD	3839	10:00 AM	NaN
7	4/23/23	Sunday	ORD	3524	4:55 PM	NaN
8	4/23/23	Sunday	ORD	538	9:34 PM	NaN
9	4/24/23	Monday	ORD	3839	10:00 AM	NaN
10	4/24/23	Monday	ORD	3524	4:50 PM	NaN
11	4/24/23	Monday	ORD	538	9:34 PM	NaN
12	4/21/23	Friday	DEN	604	3:12 PM	NaN
13	4/22/23	Saturday	DEN	604	3:12 PM	NaN
14	4/23/23	Sunday	DEN	604	3:12 PM	NaN
15	4/24/23	Monday	DEN	604	3:12 PM	NaN
16	4/21/23	Friday	EWR	4189	10:46 AM	NaN
17	4/21/23	Friday	EWR	1412	11:42 PM	NaN
18	4/22/23	Saturday	EWR	4189	10:46 AM	NaN
19	4/22/23	Saturday	EWR	1412	11:17 PM	NaN
20	4/23/23	Sunday	EWR	4189	10:46 AM	NaN
21	4/23/23	Sunday	EWR	1412	11:42 PM	NaN
22	4/24/23	Monday	EWR	4189	10:46 AM	NaN
23	4/24/23	Monday	EWR	1412	11:42 PM	NaN
24	4/21/23	Friday	IAD	4490	1:57 PM	NaN
25	4/21/23	Friday	IAD	4165	6:59 PM	NaN
26	4/22/23	Saturday	IAD	3805	1:58 PM	NaN
27	4/22/23	Saturday	IAD	4165	6:59 PM	NaN
28	4/23/23	Sunday	IAD	4490	1:57 PM	NaN
29	4/23/23	Sunday	IAD	4165	6:59 PM	NaN
30	4/24/23	Monday	IAD	4490	1:57 PM	NaN
31	4/24/23	Monday	IAD	4165	6:59 PM	NaN

```
f2[["month", "day", "year"]] = f2["Date"].str.split("/", expand = True)
```

```
f2
```

	Date	Day	Origin Airport	Flight Number	Arrival Time	Status (Early, On-time, Late, Severly Late)	month	day	year
0	4/21/23	Friday	ORD	3839	10:00 AM	NaN	4	21	23
1	4/21/23	Friday	ORD	3524	4:50 PM	NaN	4	21	23
2	4/21/23	Friday	ORD	538	9:34 PM	NaN	4	21	23
3	4/22/23	Saturday	ORD	3839	10:00 AM	NaN	4	22	23
4	4/22/23	Saturday	ORD	3524	4:50 PM	NaN	4	22	23
5	4/22/23	Saturday	ORD	538	9:34 PM	NaN	4	22	23
6	4/23/23	Sunday	ORD	3839	10:00 AM	NaN	4	23	23
7	4/23/23	Sunday	ORD	3524	4:55 PM	NaN	4	23	23
8	4/23/23	Sunday	ORD	538	9:34 PM	NaN	4	23	23
9	4/24/23	Monday	ORD	3839	10:00 AM	NaN	4	24	23
10	4/24/23	Monday	ORD	3524	4:50 PM	NaN	4	24	23
11	4/24/23	Monday	ORD	538	9:34 PM	NaN	4	24	23
12	4/21/23	Friday	DEN	604	3:12 PM	NaN	4	21	23
13	4/22/23	Saturday	DEN	604	3:12 PM	NaN	4	22	23
14	4/23/23	Sunday	DEN	604	3:12 PM	NaN	4	23	23
15	4/24/23	Monday	DEN	604	3:12 PM	NaN	4	24	23
16	4/21/23	Friday	EWR	4189	10:46 AM	NaN	4	21	23
17	4/21/23	Friday	EWR	1412	11:42 PM	NaN	4	21	23
18	4/22/23	Saturday	EWR	4189	10:46 AM	NaN	4	22	23
19	4/22/23	Saturday	EWR	1412	11:17 PM	NaN	4	22	23
20	4/23/23	Sunday	EWR	4189	10:46 AM	NaN	4	23	23
21	4/23/23	Sunday	EWR	1412	11:42 PM	NaN	4	23	23
22	4/24/23	Monday	EWR	4189	10:46 AM	NaN	4	24	23
23	4/24/23	Monday	EWR	1412	11:42 PM	NaN	4	24	23
24	4/21/23	Friday	IAD	4490	1:57 PM	NaN	4	21	23
25	4/21/23	Friday	IAD	4165	6:59 PM	NaN	4	21	23
26	4/22/23	Saturday	IAD	3805	1:58 PM	NaN	4	22	23
27	4/22/23	Saturday	IAD	4165	6:59 PM	NaN	4	22	23
28	4/23/23	Sunday	IAD	4490	1:57 PM	NaN	4	23	23
29	4/23/23	Sunday	IAD	4165	6:59 PM	NaN	4	23	23
30	4/24/23	Monday	IAD	4490	1:57 PM	NaN	4	24	23
31	4/24/23	Monday	IAD	4165	6:59 PM	NaN	4	24	23

```
f2.columns
```

```
Index(['Date', 'Day', 'Origin Airport', 'Flight Number', 'Arrival Time',
       'Status (Early, On-time, Late, Severly Late)', 'month', 'day', 'year'],
      dtype='object')
```

drop unnecessary columns

```
f2=f2.drop(columns=['Date', 'Day','Arrival Time','Status (Early, On-time, Late, Severly Late)'])
```

```
f2
```

	Origin Airport	Flight Number	month	day	year
0	ORD	3839	4	21	23
1	ORD	3524	4	21	23
2	ORD	538	4	21	23
3	ORD	3839	4	22	23
4	ORD	3524	4	22	23
5	ORD	538	4	22	23
6	ORD	3839	4	23	23
7	ORD	3524	4	23	23
8	ORD	538	4	23	23
9	ORD	3839	4	24	23
10	ORD	3524	4	24	23
11	ORD	538	4	24	23
12	DEN	604	4	21	23
13	DEN	604	4	22	23
14	DEN	604	4	23	23
15	DEN	604	4	24	23
16	EWR	4189	4	21	23
17	EWR	1412	4	21	23
18	EWR	4189	4	22	23
19	EWR	1412	4	22	23
20	EWR	4189	4	23	23
21	EWR	1412	4	23	23
22	EWR	4189	4	24	23
23	EWR	1412	4	24	23
24	IAD	4490	4	21	23
25	IAD	4165	4	21	23
26	IAD	3805	4	22	23

merge the files on origin airport and date

```
ff=pd.merge(f1,f2,on=['Origin Airport','day','month','year'])
```

```
ff
```

	Unnamed: 0	Origin Airport	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	...	solarradiation	solarer
0	0	ORD	16.6	7.1	11.2	16.6	5.4	10.2	2.0	55.1	...	115.2	
1	0	ORD	16.6	7.1	11.2	16.6	5.4	10.2	2.0	55.1	...	115.2	
2	0	ORD	16.6	7.1	11.2	16.6	5.4	10.2	2.0	55.1	...	115.2	
3	1	ORD	8.7	3.9	5.9	6.6	-1.2	2.6	-1.2	60.5	...	69.4	
4	1	ORD	8.7	3.9	5.9	6.6	-1.2	2.6	-1.2	60.5	...	69.4	
5	1	ORD	8.7	3.9	5.9	6.6	-1.2	2.6	-1.2	60.5	...	69.4	
6	2	ORD	5.7	2.3	4.3	3.7	-1.5	1.2	-2.2	63.2	...	35.6	
7	2	ORD	5.7	2.3	4.3	3.7	-1.5	1.2	-2.2	63.2	...	35.6	
8	2	ORD	5.7	2.3	4.3	3.7	-1.5	1.2	-2.2	63.2	...	35.6	
9	3	ORD	11.8	2.2	7.1	11.8	-0.8	5.2	-3.8	47.6	...	91.1	
10	3	ORD	11.8	2.2	7.1	11.8	-0.8	5.2	-3.8	47.6	...	91.1	
11	3	ORD	11.8	2.2	7.1	11.8	-0.8	5.2	-3.8	47.6	...	91.1	
12	0	DEN	8.8	-3.4	2.7	6.2	-7.4	-0.7	-12.5	36.0	...	127.2	
13	1	DEN	1.4	-5.2	-1.7	1.3	-10.3	-5.5	-4.4	82.5	...	105.5	
14	2	DEN	12.6	-3.6	4.3	12.6	-7.3	2.3	-4.8	57.0	...	295.9	
15	3	DEN	16.5	2.6	9.7	16.5	-0.6	8.6	-4.2	41.7	...	266.5	
16	0	EWR	20.0	10.7	14.9	20.0	10.7	14.9	8.9	68.2	...	255.1	
17	0	EWR	20.0	10.7	14.9	20.0	10.7	14.9	8.9	68.2	...	255.1	
18	1	EWR	20.5	11.6	15.5	20.5	11.6	15.5	11.8	79.3	...	156.0	
19	1	EWR	20.5	11.6	15.5	20.5	11.6	15.5	11.8	79.3	...	156.0	
20	2	EWR	20.4	12.7	16.2	20.4	12.7	16.2	8.5	65.6	...	189.6	
21	2	EWR	20.4	12.7	16.2	20.4	12.7	16.2	8.5	65.6	...	189.6	
22	3	EWR	16.6	7.7	12.3	16.6	5.0	11.6	-0.1	45.3	...	240.6	

perform label encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
ff['Origin Airport'] = le.fit_transform(ff['Origin Airport'])
ff['icon'] = le.fit_transform(ff['icon'])
```

```
ff
```

	Unnamed: 0	Origin Airport	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	...	solarradiation	solarer
0	0	3	16.6	7.1	11.2	16.6	5.4	10.2	2.0	55.1	...	115.2	
1	0	3	16.6	7.1	11.2	16.6	5.4	10.2	2.0	55.1	...	115.2	
2	0	3	16.6	7.1	11.2	16.6	5.4	10.2	2.0	55.1	...	115.2	
3	1	3	8.7	3.9	5.9	6.6	-1.2	2.6	-1.2	60.5	...	69.4	
4	1	3	8.7	3.9	5.9	6.6	-1.2	2.6	-1.2	60.5	...	69.4	
5	1	3	8.7	3.9	5.9	6.6	-1.2	2.6	-1.2	60.5	...	69.4	
6	2	3	5.7	2.3	4.3	3.7	-1.5	1.2	-2.2	63.2	...	35.6	
7	2	3	5.7	2.3	4.3	3.7	-1.5	1.2	-2.2	63.2	...	35.6	
8	2	3	5.7	2.3	4.3	3.7	-1.5	1.2	-2.2	63.2	...	35.6	
9	3	3	11.8	2.2	7.1	11.8	-0.8	5.2	-3.8	47.6	...	91.1	
10	3	3	11.8	2.2	7.1	11.8	-0.8	5.2	-3.8	47.6	...	91.1	
11	3	3	11.8	2.2	7.1	11.8	-0.8	5.2	-3.8	47.6	...	91.1	
12	0	0	8.8	-3.4	2.7	6.2	-7.4	-0.7	-12.5	36.0	...	127.2	
13	1	0	1.4	-5.2	-1.7	1.3	-10.3	-5.5	-4.4	82.5	...	105.5	
14	2	0	12.6	-3.6	4.3	12.6	-7.3	2.3	-4.8	57.0	...	295.9	
15	3	0	16.5	2.6	9.7	16.5	-0.6	8.6	-4.2	41.7	...	266.5	
16	0	1	20.0	10.7	14.9	20.0	10.7	14.9	8.9	68.2	...	255.1	
17	0	1	20.0	10.7	14.9	20.0	10.7	14.9	8.9	68.2	...	255.1	
18	1	1	20.5	11.6	15.5	20.5	11.6	15.5	11.8	79.3	...	156.0	
19	1	1	20.5	11.6	15.5	20.5	11.6	15.5	11.8	79.3	...	156.0	
20	2	1	20.4	12.7	16.2	20.4	12.7	16.2	8.5	65.6	...	189.6	
21	2	1	20.4	12.7	16.2	20.4	12.7	16.2	8.5	65.6	...	189.6	
22	3	1	16.6	7.7	12.3	16.6	5.0	11.6	-0.1	45.3	...	240.6	
23	3	1	16.6	7.7	12.3	16.6	5.0	11.6	-0.1	45.3	...	240.6	
24	0	2	31.1	14.1	22.0	29.3	14.1	21.8	12.9	60.6	...	221.4	
25	0	2	31.1	14.1	22.0	29.3	14.1	21.8	12.9	60.6	...	221.4	
26	1	2	22.8	14.8	18.9	22.8	14.8	18.9	12.5	68.5	...	60.5	
27	1	2	22.8	14.8	18.9	22.8	14.8	18.9	12.5	68.5	...	60.5	
28	2	2	17.8	9.9	14.2	17.8	6.7	14.1	5.3	57.2	...	117.7	
29	2	2	17.8	9.9	14.2	17.8	6.7	14.1	5.3	57.2	...	117.7	
30	3	2	16.0	8.1	11.6	16.0	5.5	10.8	-1.3	41.5	...	205.9	

```
ff.columns
```

```
Index(['Unnamed: 0', 'Origin Airport', 'tempmax', 'tempmin', 'temp',
       'feelslikemax', 'feelslikemin', 'feelslike', 'dew', 'humidity',
       'precip', 'precipprob', 'precipcover', 'snow', 'snowdepth', 'windgust',
       'windspeed', 'winddir', 'sealevelpressure', 'cloudcover', 'visibility',
       'solarradiation', 'solarenergy', 'uvindex', 'severerisk', 'moonphase',
       'icon', 'day', 'month', 'year', 'Flight Number'],
      dtype='object')
```

```
ff=ff.drop(columns=['Unnamed: 0'])
```

Fit the data into our pretrained model

```
ff=pd.DataFrame(sc.transform(ff),columns=ff.columns)
```

```
ff
```

	Origin Airport	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	precip	...	solarradiat
0	0.982572	-0.263142	-0.181241	-0.272906	-0.186942	-0.119592	-0.211080	-0.232041	-0.121530	-0.374755	...	0.00
1	0.982572	-0.263142	-0.181241	-0.272906	-0.186942	-0.119592	-0.211080	-0.232041	-0.121530	-0.374755	...	0.00
2	0.982572	-0.263142	-0.181241	-0.272906	-0.186942	-0.119592	-0.211080	-0.232041	-0.121530	-0.374755	...	0.00
3	0.982572	-1.005773	-0.492784	-0.791169	-1.033825	-0.643684	-0.844473	-0.536619	0.222887	-0.215823	...	-0.62
4	0.982572	-1.005773	-0.492784	-0.791169	-1.033825	-0.643684	-0.844473	-0.536619	0.222887	-0.215823	...	-0.62
5	0.982572	-1.005773	-0.492784	-0.791169	-1.033825	-0.643684	-0.844473	-0.536619	0.222887	-0.215823	...	-0.62
6	0.982572	-1.287785	-0.648556	-0.947626	-1.279421	-0.667506	-0.961151	-0.631800	0.395096	-0.374755	...	-1.08
7	0.982572	-1.287785	-0.648556	-0.947626	-1.279421	-0.667506	-0.961151	-0.631800	0.395096	-0.374755	...	-1.08
8	0.982572	-1.287785	-0.648556	-0.947626	-1.279421	-0.667506	-0.961151	-0.631800	0.395096	-0.374755	...	-1.08
9	0.982572	-0.714361	-0.658292	-0.673826	-0.593446	-0.611920	-0.627786	-0.784089	-0.599888	-0.374755	...	-0.32
10	0.982572	-0.714361	-0.658292	-0.673826	-0.593446	-0.611920	-0.627786	-0.784089	-0.599888	-0.374755	...	-0.32
11	0.982572	-0.714361	-0.658292	-0.673826	-0.593446	-0.611920	-0.627786	-0.784089	-0.599888	-0.374755	...	-0.32
12	-1.376767	-0.996373	-1.203493	-1.104082	-1.067700	-1.136012	-1.119499	-1.612161	-1.339749	-0.145722	...	0.16
13	-1.376767	-1.692003	-1.378736	-1.534338	-1.482673	-1.366294	-1.519537	-0.841198	1.626071	0.140748	...	-0.13
14	-1.376767	-0.639158	-1.222964	-0.947626	-0.525695	-1.128071	-0.869476	-0.879270	-0.000346	-0.374755	...	2.47
15	-1.376767	-0.272542	-0.619349	-0.419584	-0.195410	-0.596039	-0.344426	-0.822161	-0.976197	-0.374755	...	2.07
16	-0.590320	0.056472	0.169245	0.088900	0.100999	0.301269	0.180624	0.424706	0.714001	-0.374755	...	1.91
17	-0.590320	0.056472	0.169245	0.088900	0.100999	0.301269	0.180624	0.424706	0.714001	-0.374755	...	1.91
18	-0.590320	0.103474	0.256867	0.147572	0.143343	0.372736	0.230629	0.700730	1.421971	2.910015	...	0.55
19	-0.590320	0.103474	0.256867	0.147572	0.143343	0.372736	0.230629	0.700730	1.421971	2.910015	...	0.55
20	-0.590320	0.094074	0.363960	0.216021	0.134874	0.460085	0.288968	0.386633	0.548171	4.996285	...	1.01
21	-0.590320	0.094074	0.363960	0.216021	0.134874	0.460085	0.288968	0.386633	0.548171	4.996285	...	1.01
22	-0.590320	-0.263142	-0.122826	-0.165342	-0.186942	-0.151355	-0.094402	-0.431921	-0.746585	-0.374755	...	1.71
23	-0.590320	-0.263142	-0.122826	-0.165342	-0.186942	-0.151355	-0.094402	-0.431921	-0.746585	-0.374755	...	1.71
24	0.196126	1.099917	0.500260	0.783177	0.888600	0.571256	0.755679	0.805428	0.229265	-0.374755	...	1.45
25	0.196126	1.099917	0.500260	0.783177	0.888600	0.571256	0.755679	0.805428	0.229265	-0.374755	...	1.45
26	0.196126	0.319683	0.568411	0.480042	0.338126	0.626841	0.513989	0.767356	0.733136	1.478201	...	-0.74
27	0.196126	0.319683	0.568411	0.480042	0.338126	0.626841	0.513989	0.767356	0.733136	1.478201	...	-0.74
28	0.196126	-0.150337	0.091360	0.020450	-0.085316	-0.016362	0.113951	0.082055	0.012410	-0.374755	...	0.03
29	0.196126	-0.150337	0.091360	0.020450	-0.085316	-0.016362	0.113951	0.082055	0.012410	-0.374755	...	0.03

```
test1=catboost.predict(ff)
```

```
test1
```

```
array([['1'],
       ['1'],
       ['0'],
       ['1'],
       ['1'],
       ['1'],
       ['0'],
       ['0'],
       ['1'],
       ['1'],
       ['1'],
       ['0'],
       ['1'],
       ['3'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['1'],
       ['3'],
```