```
from google.colab import drive
drive.mount
```

```
<function google.colab.drive.mount(mountpoint, force_remount=False,
timeout_ms=120000, readonly=False)>
```

```
import numpy as np
import pandas as pd
import pandas as contact
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

```
dataset= pd.read_csv("/content/flightdata.csv")
dataset.head()
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_
0	2016	1	1	1	
1	2016	1	1	1	
2	2016	1	1	1	
3	2016	1	1	1	
4	2016	1	1	1	

5 rows × 26 columns

```
dataset.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
```

```
dataset = dataset[["FL_NUM", "MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "ORIGIN", "DEST", "CRS_ARR_TIME", "DEP_DEL15", "
dataset.isnull().sum()
```

```
FL_NUM            0
MONTH             0
DAY_OF_MONTH      0
DAY_OF_WEEK       0
ORIGIN            0
DEST              0
CRS_ARR_TIME      0
DEP_DEL15       107
ARR_DEL15       188
dtype: int64
```

```
dataset[dataset.isnull().any(axis=1)].head(10)
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK
177	2834	1	9	6
179	86	1	10	7
184	557	1	10	7
210	1096	1	10	7
478	1542	1	22	5
481	1795	1	22	5
491	2312	1	22	5
499	423	1	23	6
500	425	1	23	6
501	427	1	23	6

```
dataset['DEP_DEL15'].mode()
```

```
0    0.0
Name: DEP_DEL15, dtype: float64
```

```
dataset = dataset.fillna({'ARR_DEL15': 1})
dataset = dataset.fillna({'DEP_DEL15': 0})
dataset.iloc[177:185]
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK
177	2834	1	9	6
178	2839	1	9	6
179	86	1	10	7
180	87	1	10	7
181	423	1	10	7
182	440	1	10	7

```
import math
for index,row in dataset.iterrows():
  dataset.loc[index, 'CRS_ARR_TIME'] = math.floor(row['CRS_ARR_TIME']/100)
dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	C
0	1399	1	1	5	
1	1476	1	1	5	
2	1597	1	1	5	
3	1768	1	1	5	
4	1823	1	1	5	

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['DEST'] = le.fit_transform(dataset['DEST'])
dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

```
dataset.head(5)
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	C
0	1399	1	1	5	
1	1476	1	1	5	
2	1597	1	1	5	
3	1768	1	1	5	
4	1823	1	1	5	

```
dataset['ORIGIN'].unique()
```

```
     array([0, 1, 4, 3, 2])
```

```
dataset = pd.get_dummies(dataset, columns=['ORIGIN', 'DEST'])
dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	C
0	1399	1	1	5	
1	1476	1	1	5	

```
x=dataset.iloc[:, 0:8].values
y=dataset.iloc[:, 8:9].values
```

```
x
```

```
array([[1.399e+03, 1.000e+00, 1.000e+00, ..., 0.000e+00, 0.000e+00,
        1.000e+00],
       [1.476e+03, 1.000e+00, 1.000e+00, ..., 0.000e+00, 0.000e+00,
        0.000e+00],
       [1.597e+03, 1.000e+00, 1.000e+00, ..., 0.000e+00, 0.000e+00,
        1.000e+00],
       ...,
       [1.823e+03, 1.200e+01, 3.000e+01, ..., 0.000e+00, 0.000e+00,
        0.000e+00],
       [1.901e+03, 1.200e+01, 3.000e+01, ..., 0.000e+00, 0.000e+00,
        1.000e+00],
       [2.005e+03, 1.200e+01, 3.000e+01, ..., 0.000e+00, 0.000e+00,
        1.000e+00]])
```

```
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder()
z=oh.fit_transform(x[:,4:5]).toarray()
t=oh.fit_transform(x[:,5:6]).toarray()
```

```
z
```

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

```
t
```

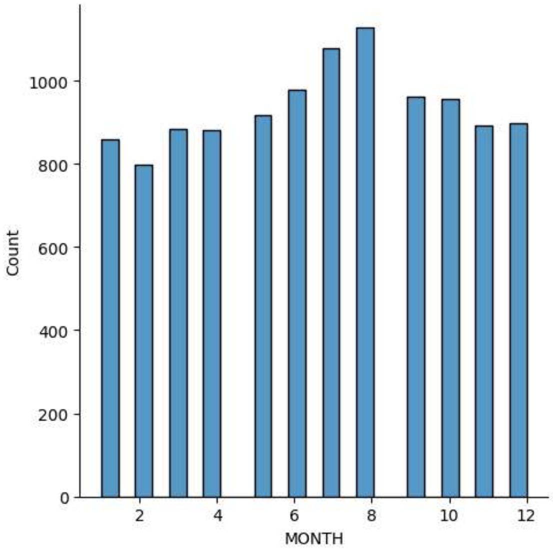
```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [1., 0.],
       [1., 0.],
       [1., 0.]])
```

```
dataset.describe()
```

	FL_NUM	MONTH	DAY_OF_MONT
count	11231.000000	11231.000000	11231.00000
mean	1334.325617	6.628973	15.79075
std	811.875227	3.354678	8.78205
min	7.000000	1.000000	1.00000

```
sns.displot(dataset.MONTH)
```

```
<seaborn.axisgrid.FacetGrid at
0x7f014efd21f0>
```



20CS07 Hariharadevi E
9:44 AM Yesterday

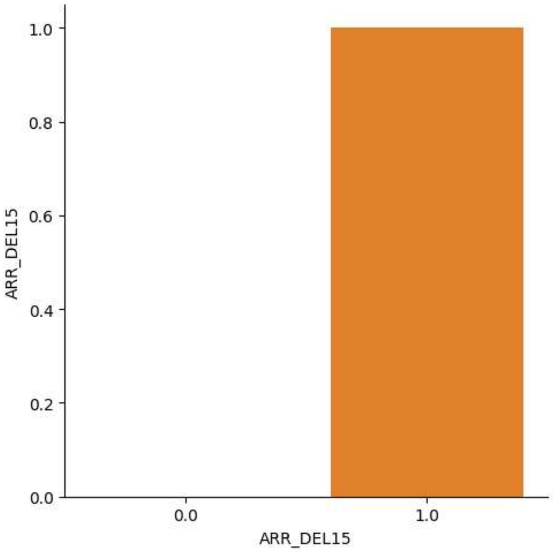
how to describe

```
sns.scatterplot(x='ARR_DEL15',y='DEP_DEL15',data=dataset)
```

```
    <Axes: xlabel='ARR_DEL15'
```

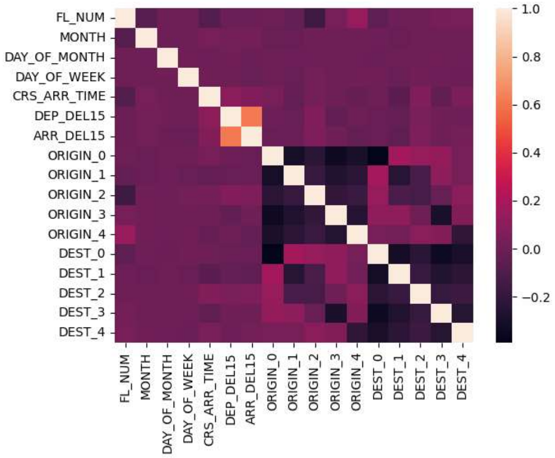
```
sns.catplot(x="ARR_DEL15",y="ARR_DEL15",kind='bar',data=dataset)
```

```
    <seaborn.axisgrid.FacetGrid at
    0x7f014c22db80>
```



```
sns.heatmap(dataset.corr())
```

```
    <Axes: >
```



```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

```
x_test.shape
```

```
(2247, 8)
```

```
x_train.shape
```

```
(8984, 8)
```

```
y_test.shape
```

```
(2247, 1)
```

```
y_train.shape
```

```
(8984, 1)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state = 0)
classifier.fit(x_train,y_train)
```

```
▾       DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

```
decisiontree = classifier.predict(x_test)
```

```
decisiontree
```

```
array([1, 0, 0, ..., 1, 0, 0], dtype=uint8)
```

```
from sklearn.metrics import accuracy_score
desacc = accuracy_score(y_test,decisiontree)
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')
```

```
rfc.fit(x_train,y_train)
```

```
<ipython-input-35-b87bb2ba9825>:1: DataConvers
  rfc.fit(x_train,y_train)
▾                   RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_
```

```
rfc.fit(x_train,y_train)
```

```
<ipython-input-36-b87bb2ba9825>:1: DataConvers
  rfc.fit(x_train,y_train)
▾                  RandomForestClassifier
RandomForestClassifier(criterion='entropy', n
```

```
y_predict = rfc.predict(x_test)
```

```
import tensorflow
from  tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
Classification = Sequential()
Classification.add(Dense(30,activation='relu'))
Classification.add(Dense(128,activation='relu'))
Classification.add(Dense(64,activation='relu'))
Classification.add(Dense(32,activation='relu'))
Classification.add(Dense(1,activation='sigmoid'))
```

```
Classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
Classification.fit(x_train,y_train,batch_size=4,validation_split=0.2,epochs=100)
```

```
1797/1797 [==============================] - 4s 2ms/step - loss: 0.0
Epoch 95/100
1797/1797 [==============================] - 6s 3ms/step - loss: 0.0
Epoch 96/100
1797/1797 [==============================] - 4s 2ms/step - loss: 0.0
Epoch 97/100
1797/1797 [==============================] - 4s 2ms/step - loss: 0.0
Epoch 98/100
1797/1797 [==============================] - 6s 3ms/step - loss: 0.0
Epoch 99/100
1797/1797 [==============================] - 4s 2ms/step - loss: 0.0
Epoch 100/100
1797/1797 [==============================] - 4s 2ms/step - loss: 0.0
```

```
y_pred = classifier.predict([[129,99,1,0,0,1,0,1]])

print(y_pred)
(y_pred)
```

```
[0]
array([0], dtype=uint8)
```

```
y_pred = rfc.predict([[129,99,1,0,0,1,0,1]])

print(y_pred)
(y_pred)
```

```
[0]
array([0], dtype=uint8)
```

```
Classification.save('flight.h5')
```

```
y_pred = Classification.predict(x_test)
```

```
71/71 [==============================] - 0s 1ms/step
```

```
y_pred
```

```
array([[0.9999329 ],
       [0.001969  ],
       [0.        ],
       ...,
       [0.1678162 ],
       [0.        ],
       [0.56437033]], dtype=float32)
```

```
y_pred = (y_pred > 0.5)
y_pred
```

```
array([[ True],
       [False],
       [False],
       ...,
       [False],
       [False],
       [ True]])
```

```
def predict_exit(sample_value):
  sample_value = np.array(sample_value)
  sample_value = sample_value.reshape(1,-1)
  sample_value = sc.transform(sample_value)
  return classifier.predict(sample_value)
```

```
    from sklearn import model_selection
    from sklearn.neural_network import MLPClassifier


    dfs = []
    models = [
               ('RF', RandomForestClassifier()),
               ('DecisionTree',DecisionTreeClassifier()),
               ('ANN',MLPClassifier())
             ]
    results = []
    names = []
    scoring = ['accuracy', 'precision_weighted', 'recall_weighted', 'f1_weighted', 'roc_auc']
    target_names = ['no delay', 'delay']
    for name, model in models:
           kfold = model_selection.KFold(n_splits=5, shuffle=True, random_state=90210)
           cv_results = model_selection.cross_validate(model,x_train,y_train,cv=kfold,scoring=scoring)
           clf = model.fit(x_train, y_train)
           y_pred = clf.predict(x_test)
           print(name)
           print(classification_report(y_test, y_pred, target_names=target_names))
           results.append(cv_results)
           names.append(name)
           this_df = pd.DataFrame(cv_results)
           this_df['model'] = name
           dfs.append(this_df)
    final = pd.contact(dfs, ignore_index=True)
    return final
```

```
/usr/local/lib/python3.9/dist-packages/sklearn
  estimator.fit(X_train, y_train, **fit_params
/usr/local/lib/python3.9/dist-packages/sklearn
  estimator.fit(X_train, y_train, **fit_params
/usr/local/lib/python3.9/dist-packages/sklearn
  estimator.fit(X_train, y_train, **fit_params
/usr/local/lib/python3.9/dist-packages/sklearn
  estimator.fit(X_train, y_train, **fit_params
/usr/local/lib/python3.9/dist-packages/sklearn
  estimator.fit(X_train, y_train, **fit_params
<ipython-input-64-c006436488cb>:14: DataConver
  clf = model.fit(x_train, y_train)
RF
              precision    recall  f1-score

    no delay       0.90      0.98      0.94
       delay       0.87      0.54      0.67

    accuracy                           0.89
   macro avg       0.89      0.76      0.80
weighted avg       0.89      0.89      0.88

DecisionTree
              precision    recall  f1-score

    no delay       0.99      1.00      0.99
       delay       0.98      0.97      0.98

    accuracy                           0.99
   macro avg       0.99      0.98      0.98
weighted avg       0.99      0.99      0.99

/usr/local/lib/python3.9/dist-packages/sklearn
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predict)
cm
```

```
array([[1773,   29],
       [ 217,  228]])
```

```
from sklearn.metrics import accuracy_score
desacc = accuracy_score(y_test,decisiontree)
```

```
    accuracy                           0.81
```

```
desacc
```

```
0.9893190921228304
```

```
  warnings.warn(
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,decisiontree)
```

```
Traceback (most recent call last)
```

```
cm

     array([[1790,   12],
            [  12,  433]])

     ignore_index=True)
```

```
from sklearn.metrics import accuracy_score,classification_report
score = accuracy_score(y_pred,y_test)
print('the accuracy for ANN model is: {}%'.format(score*100))

     the accuracy for ANN model is: 81.08589230084557%
         260
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm

     array([[1782,   20],
            [ 405,   40]])
```

```
parameters = {
                'n_estimators' : [1,20,30,55,68,74,90,120,115],
                'criterion':['gini','entropy'],
                'max_features' : ["auto", "sqrt", "log2"],
         'max_depth' : [2,5,8,10], 'verbose' : [1,2,3,4,6,8,9,10]
}
```

```
from flask import Flask,request,render_template
import numpy as np
import pandas as pd
import pickle
import os


model = pickle.load(open('flight.h5','rb'))

app = Flask(_name_)

     ---------------------------------------------
     ------------------------------
     UnpicklingError
     Traceback (most recent call last)
     <ipython-input-91-13bb0f5e2969> in <cell
     line: 1>()
     ----> 1 model =
     pickle.load(open('flight.h5','rb'))
           2
           3 app = Flask(_name_)
```

```
@app.route('/')
def home():
  return render_template("index.html")
@app.route('/prediction',methods =['POST'])

       File "<ipython-input-95-5c7befd58eb2>",
     line 4
         @app.route('/prediction',methods =
     ['POST'])

     ^
     SyntaxError: unexpected EOF while parsing
```

```
from flask.templating import render_template
def predict():
  name = request.form['name']
```

```
    month = request.form['month']
    dayofmonth = request.form['dayofmonth']
    origin = request.form['origin']
    if(origin == "msp"):
      origin1,origin2,origin3,origin4,origin5 = 0,0,0,0,1
    if(origin == "dtw"):
      origin1,origin2,origin3,origin4,origin5 = 1,0,0,0,0
    if(origin == "jfk"):
      origin1,origin2,origin3,origin4,origin5 = 0,0,1,0,0
    if(origin == "sea"):
      origin1,origin2,origin3,origin4,origin5 = 0,1,0,0,0
    if(origin == "alt"):
      origin1,origin2,origin3,origin4,origin5 = 0,0,0,1,0
    destination = request.form['destination']
    if(destination == "msp"):
      destination1,destination2,destination3,destination4,destination5 = 0,0,0,0,1
    if(destination == "dtw"):
      destination1,destination2,destination3,destination4,destination5 = 1,0,0,0,0
    if(destination == "jfk"):
      destination1,destination2,destination3,destination4,destination5 = 0,0,1,0,0
    if(destination == "alt"):
      destination1,destination2,destination3,destination4,destination5 = 0,0,0,1,0
  dept  = request.form['dept']
  arrtime = request.form['arrtime']
  actdept = request.form['actdept']
  dept15=int(dept)-int(actdept)
  total = [[name,month,dayofmonth,dayofweek,origin1,origin2,origin3,origin4,origin5,destination1,destination2,destinati
  y_pred = model.predict(total)
  print(y_pred)
  if(y_pred==[0.]):
    ans="the flight willbe on time"
  else:
    ans="the flight will be delayed"
  return render_template("index.html",showcase = ans)
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call
<ipython-input-100-cba71743fca1> in <cell line: 26>()
     24   if(destination == "alt"):
     25     destination1,destination2,destination3,destination4,destin
0,0,0,1,0
---> 26 dept  = request.form['dept']
     27 arrtime = request.form['arrtime']
     28 actdept = request.form['actdept']

                                  1 frames

/usr/local/lib/python3.9/dist-packages/werkzeug/local.py in _get_curre
    511                 obj = local.get()  # type: ignore[union-at
    512             except LookupError:
--> 513                 raise RuntimeError(unbound_message) from N
    514 
    515             return get_name(obj)

RuntimeError: Working outside of request context.

This typically means that you attempted to use functionality that need
an active HTTP request. Consult the documentation on testing for
information about how to avoid this problem.
```

SEARCH STACK OVERFLOW

```
  if_name_== '_main__':
    app.run(debug = True)
  File "<ipython-input-103-baffd0416b7d>",
line 1
    if_name_== '_main__':
                        ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

0s completed at 10:50 PM