```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df=pd.read_csv('/content/adult.csv')
```

```
df.shape
```

```
(32560, 15)
```

```
df.sample(10)
```

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male
6263	38	Private	63509	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male
12950	19	Private	304469	10th	6	Never-married	Farming-fishing	Own-child	White	Male
15815	35	Private	119992	Assoc-acdm	12	Married-civ-spouse	Craft-repair	Husband	White	Male
30055	18	Private	174394	HS-grad	9	Never-married	Other-service	Own-child	White	Female
4582	39	Private	82488	Some-college	10	Divorced	Sales	Unmarried	Asian-Pac-Islander	Female

```
df.columns=['Age','Workclass','Fnlwgt','Education','education_num','marital_status','occupation','relationship','race','sex','capital_gain','
```

```
df
```

	Age	Workclass	Fnlwgt	Education	education_num	marital_status	occupation	rel
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	N
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	
...	...	...	...	...	...	...	...	
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	

32560 rows × 15 columns

```
df.isnull()
```

	Age	Workclass	Fnlwgt	Education	education_num	marital_status	occupation	re
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
32555	False	False	False	False	False	False	False	
32556	False	False	False	False	False	False	False	
32557	False	False	False	False	False	False	False	
32558	False	False	False	False	False	False	False	
32559	False	False	False	False	False	False	False	

32560 rows × 15 columns

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['Workclass']= label_encoder.fit_transform(df['Workclass'])
```

```
df['marital_status']= label_encoder.fit_transform(df['marital_status'])
df['relationship']= label_encoder.fit_transform(df['relationship'])
df['race']= label_encoder.fit_transform(df['race'])
df['sex']= label_encoder.fit_transform(df['sex'])
df['native_country']= label_encoder.fit_transform(df['native_country'])
```

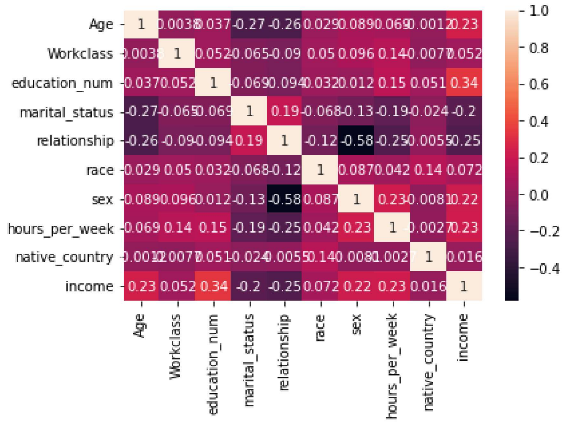
```
df['income']= label_encoder.fit_transform(df['income'])
```

```
df=df.drop(columns=['Fnlwgt','Education','occupation','capital_gain','capital_loss'])
```

```
corr=df.corr()
```

```
sns.heatmap(corr,annot=True)
```

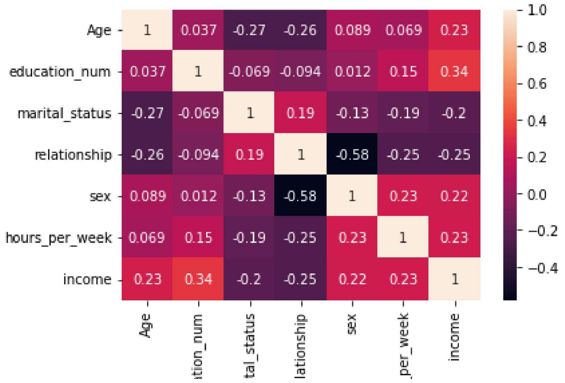
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc3e3639e80>
```



```
df=df.drop(columns=['Workclass','race','native_country'])
```

```
corr=df.corr()
sns.heatmap(corr,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc3f1fe04c0>
```



SPLITTING INPUT AND OUTPUT

```
X=df.drop(['income','marital_status','sex','Age','hours_per_week'],axis=1)
y=df['income']
```

a. DECISION TREE

```
from sklearn import tree
model = tree.DecisionTreeClassifier(criterion="entropy")
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 42)
```

```
model.fit(X_train, y_train)
```

```
     DecisionTreeClassifier(criterion='entropy')
```

```
model.score(X_train, y_train)
```

```
     0.8227009477009477
```

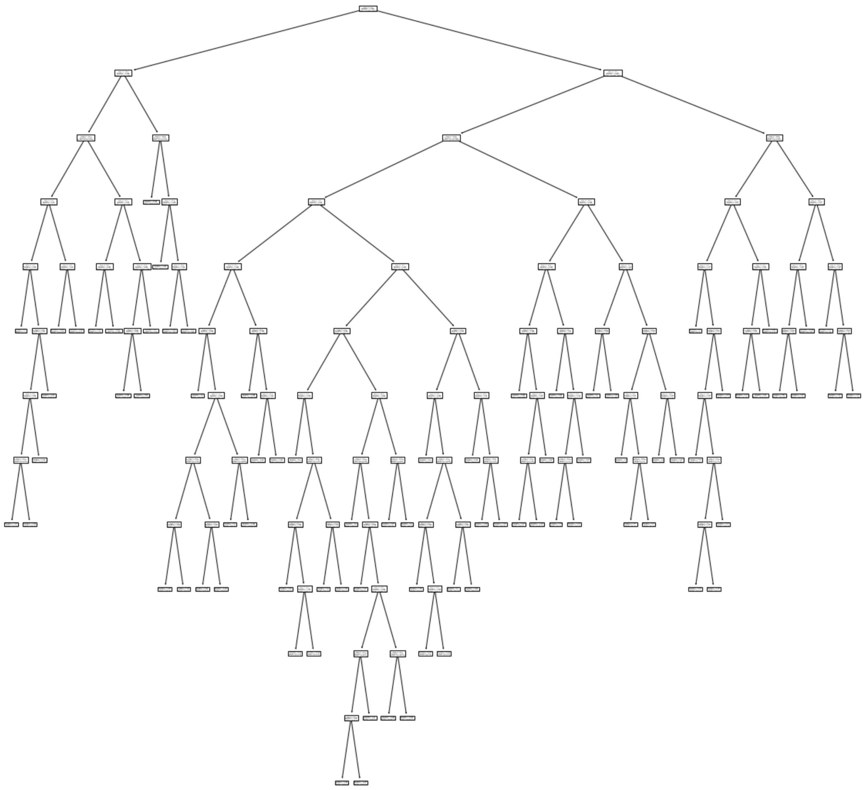
```
y_pred = model.predict(X_test)
```

```
model.score(X_test,y_test)
```

```
     0.8162366912366913
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(20,20))

from sklearn import tree

tree.plot_tree(model.fit(X_train, y_train))
```

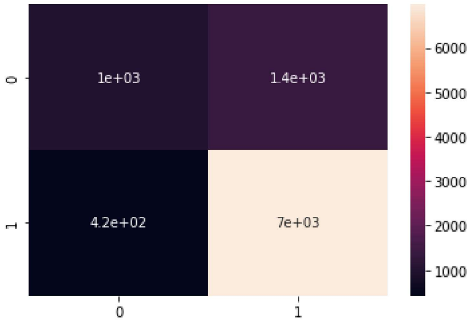
```
 Text(0.8810810810810811, 0.5, 'entropy = 0.93\nsamples = 350\nvalue = [229, 121]'),
 Text(0.8918918918918919, 0.5769230769230769, 'entropy = 0.982\nsamples = 221\nvalue
= [128, 93]'),
 Text(0.9459459459459459, 0.7307692307692307, 'X[0] <= 13.5\nentropy =
0.917\nsamples = 407\nvalue = [135, 272]'),
 Text(0.9243243243243243, 0.6538461538461539, 'X[0] <= 12.5\nentropy =
0.958\nsamples = 308\nvalue = [117, 191]'),
 Text(0.9135135135135135, 0.5769230769230769, 'X[0] <= 11.5\nentropy =
0.998\nsamples = 104\nvalue = [49, 55]'),
 Text(0.9027027027027027, 0.5, 'entropy = 0.981\nsamples = 55\nvalue = [23, 32]'),
 Text(0.9243243243243243, 0.5, 'entropy = 0.997\nsamples = 49\nvalue = [26, 23]'),
 Text(0.9351351351351351, 0.5769230769230769, 'entropy = 0.918\nsamples = 204\nvalue
= [68, 136]'),
 Text(0.9675675675675676, 0.6538461538461539, 'X[0] <= 14.5\nentropy =
0.684\nsamples = 99\nvalue = [18, 81]'),
 Text(0.9567567567567568, 0.5769230769230769, 'entropy = 0.77\nsamples = 71\nvalue =
[16, 55]'),
 Text(0.9783783783783784, 0.5769230769230769, 'X[0] <= 15.5\nentropy =
0.371\nsamples = 28\nvalue = [2, 26]'),
 Text(0.9675675675675676, 0.5, 'entropy = 0.337\nsamples = 16\nvalue = [1, 15]'),
 Text(0.9891891891891892, 0.5, 'entropy = 0.414\nsamples = 12\nvalue = [1, 11]')]
```



```
from sklearn.metrics import confusion_matrix

cf=confusion_matrix(y_test, y_pred, labels = [1,0])
```

```
sns.heatmap(cf,annot=True)
```

```
     <matplotlib.axes._subplots.AxesSubplot at 0x7fc3def1dfd0>
```



```
from sklearn.metrics import classification_report

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

```
                  precision    recall  f1-score   support

               0       0.84      0.94      0.89      7395
               1       0.70      0.42      0.53      2373

        accuracy                           0.82      9768
       macro avg       0.77      0.68      0.71      9768
    weighted avg       0.80      0.82      0.80      9768
```

b. RANDOM-FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
classifier_rf = RandomForestClassifier(random_state=42, n_jobs=-1, max_depth=5, n_estimators=100, oob_score=True)
```

```
classifier_rf.fit(X_train, y_train)
```

```
     RandomForestClassifier(max_depth=5, n_jobs=-1, oob_score=True, random_state=42)
```

```
classifier_rf.oob_score_
```

```
     0.819980694980695
```

```
classifier_rf.score(X_test, y_test)
```

```
     0.8158271908271908
```

```
rf = RandomForestClassifier(random_state=42, n_jobs=-1)
```

```
params = {
    'max_depth': [2,3,5,10,20],
    'min_samples_leaf': [5,10,20,50,100,200],
    'n_estimators': [10,25,30,50,100,200]
}
```

```
from sklearn.model_selection import GridSearchCV
```

```
grid_search = GridSearchCV(estimator=rf, param_grid=params, cv = 4, n_jobs=-1, verbose=1, scoring="accuracy")
```

```
grid_search.fit(X_train, y_train)
```

```
     Fitting 4 folds for each of 180 candidates, totalling 720 fits
     GridSearchCV(cv=4, estimator=RandomForestClassifier(n_jobs=-1, random_state=42),
                  n_jobs=-1,
                  param_grid={'max_depth': [2, 3, 5, 10, 20],
                              'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                              'n_estimators': [10, 25, 30, 50, 100, 200]},
                  scoring='accuracy', verbose=1)
```

```
grid_search.best_score_
```

```
0.821911196911197
```

```
rf_best = grid_search.best_estimator_
rf_best
```

```
RandomForestClassifier(max_depth=10, min_samples_leaf=10, n_estimators=50,
                       n_jobs=-1, random_state=42)
```

```
rf_best.feature_importances_
```

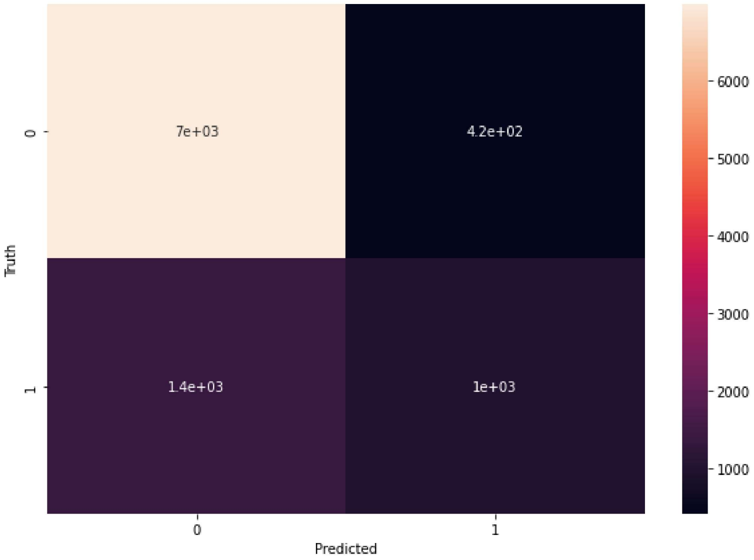
```
array([0.40102007, 0.59897993])
```

```
from sklearn.metrics import confusion_matrix

y_pred = rf_best.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[6971,  424],
       [1371, 1002]])
```

```
plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```



```
from scipy.stats import randint

rs_space={'max_depth':list(np.arange(10, 100, step=10)) + [None],
              'n_estimators':np.arange(10, 500, step=50),
              'max_features':randint(1,7),
              'criterion':['gini','entropy'],
              'min_samples_leaf':randint(1,4),
              'min_samples_split':np.arange(2, 10, step=2)
         }
```

```
from sklearn.model_selection import RandomizedSearchCV
rf = RandomForestClassifier(random_state=42, n_jobs=-1)
rf_random = RandomizedSearchCV(rf, rs_space, n_iter=50, scoring='accuracy', n_jobs=-1, cv=4)
```

```
%%time
model_random = rf_random.fit(X_train, y_train)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
124 fits failed out of a total of 200.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
124 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py", line 450, in fit
    trees = Parallel(
  File "/usr/local/lib/python3.8/dist-packages/joblib/parallel.py", line 1098, in __call__
    self.retrieve()
  File "/usr/local/lib/python3.8/dist-packages/joblib/parallel.py", line 975, in retrieve
    self._output.extend(job.get(timeout=self.timeout))
  File "/usr/lib/python3.8/multiprocessing/pool.py", line 771, in get
    raise self._value
  File "/usr/lib/python3.8/multiprocessing/pool.py", line 125, in worker
    result = (True, func(*args, **kwds))
  File "/usr/local/lib/python3.8/dist-packages/joblib/_parallel_backends.py", line 620, in __call__
    return self.func(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/joblib/parallel.py", line 288, in __call__
    return [func(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/joblib/parallel.py", line 288, in <listcomp>
    return [func(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/utils/fixes.py", line 216, in __call__
    return self.function(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py", line 185, in _parallel_build_trees
    tree.fit(X, y, sample_weight=curr_sample_weight, check_input=False)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/tree/_classes.py", line 937, in fit
    super().fit(
  File "/usr/local/lib/python3.8/dist-packages/sklearn/tree/_classes.py", line 308, in fit
    raise ValueError("max_features must be in (0, n_features]")
ValueError: max_features must be in (0, n_features]

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_search.py:969: UserWarning: One or more of the test scores are non-finit
 0.8215602         nan 0.82142857        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.82142857 0.82142857 0.8217357         nan        nan        nan
        nan 0.82160407        nan 0.8215602  0.82160407        nan
 0.82142857 0.8215602         nan        nan        nan        nan
 0.82160407        nan 0.82142857 0.82160407        nan 0.82164795
        nan        nan 0.82160407        nan 0.82160407        nan
        nan        nan]
  warnings.warn(
CPU times: user 6.87 s, sys: 475 ms, total: 7.34 s
Wall time: 1min 32s
```

```
model_random.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 50,
 'max_features': 2,
 'min_samples_leaf': 2,
 'min_samples_split': 6,
 'n_estimators': 460}
```

```
model_random.best_score_
```

```
0.8217356967356967
```

```
rf_best1 = model_random.best_estimator_
rf_best1
```

```
RandomForestClassifier(max_depth=50, max_features=2, min_samples_leaf=2,
                       min_samples_split=6, n_estimators=460, n_jobs=-1,
                       random_state=42)
```

```
from sklearn.metrics import confusion_matrix

y_pred1 = rf_best1.predict(X_test)

cm1 = confusion_matrix(y_test, y_pred)
cm1
```

```
     array([[6971,  424],
            [1371, 1002]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
                   precision    recall  f1-score   support

                0       0.84      0.94      0.89      7395
                1       0.70      0.42      0.53      2373

         accuracy                           0.82      9768
        macro avg       0.77      0.68      0.71      9768
     weighted avg       0.80      0.82      0.80      9768
```

c. LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
     LogisticRegression(random_state=0)
```

```
y_pred = classifier.predict(X_test)
```

```
print(classifier.score(X_train, y_train))
print(classifier.score(X_test, y_test))
```

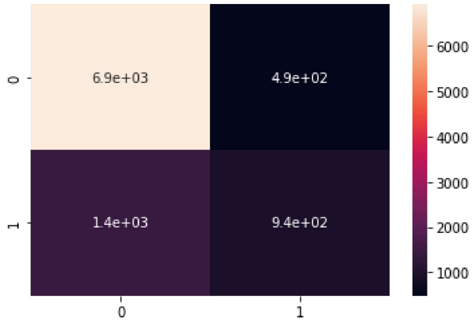
```
     0.8086170586170586
     0.8029279279279279
```

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

print ("Confusion Matrix : \n", cm)
```

```
     Confusion Matrix :
      [[6906  489]
      [1436  937]]
```

```
sns.heatmap(cm,annot=True)
```

```
     <matplotlib.axes._subplots.AxesSubplot at 0x7f85f22da640>
```



```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

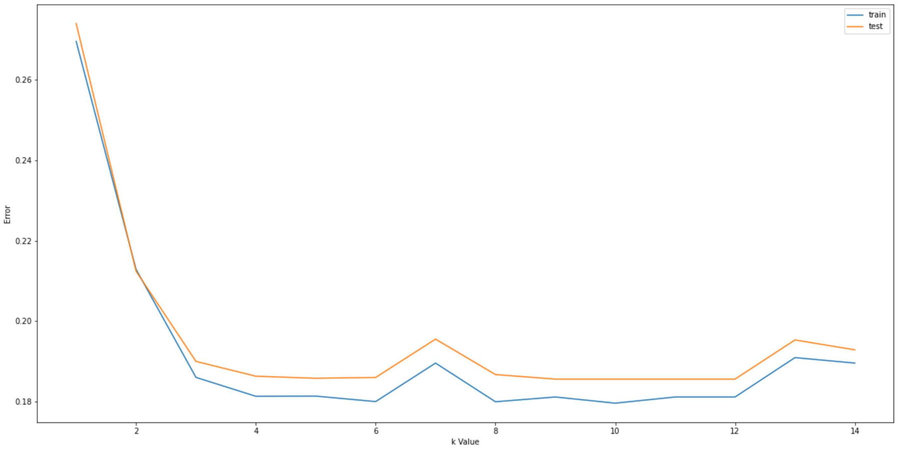
```
                   precision    recall  f1-score   support

                0       0.83      0.93      0.88      7395
                1       0.66      0.39      0.49      2373

         accuracy                           0.80      9768
```

```
       macro avg       0.74      0.66      0.69      9768
    weighted avg       0.79      0.80      0.78      9768
```

d. KNN CLASSIFIER

```
from sklearn.neighbors import KNeighborsClassifier

error1= []
error2= []
for k in range(1,15):
    knn= KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
    y_pred1= knn.predict(X_train)
    error1.append(np.mean(y_train!= y_pred1))
    y_pred2= knn.predict(X_test)
    error2.append(np.mean(y_test!= y_pred2))

plt.figure(figsize=(20,10))
plt.plot(range(1,15),error1,label="train")
plt.plot(range(1,15),error2,label="test")
plt.xlabel('k Value')
plt.ylabel('Error')
plt.legend()
```

```
     <matplotlib.legend.Legend at 0x7f85f9c59430>
```



```
knn = KNeighborsClassifier(n_neighbors=6)
```

```
knn.fit(X_train, y_train)
```

```
     KNeighborsClassifier(n_neighbors=6)
```

```
knn.score(X_test, y_test)
```

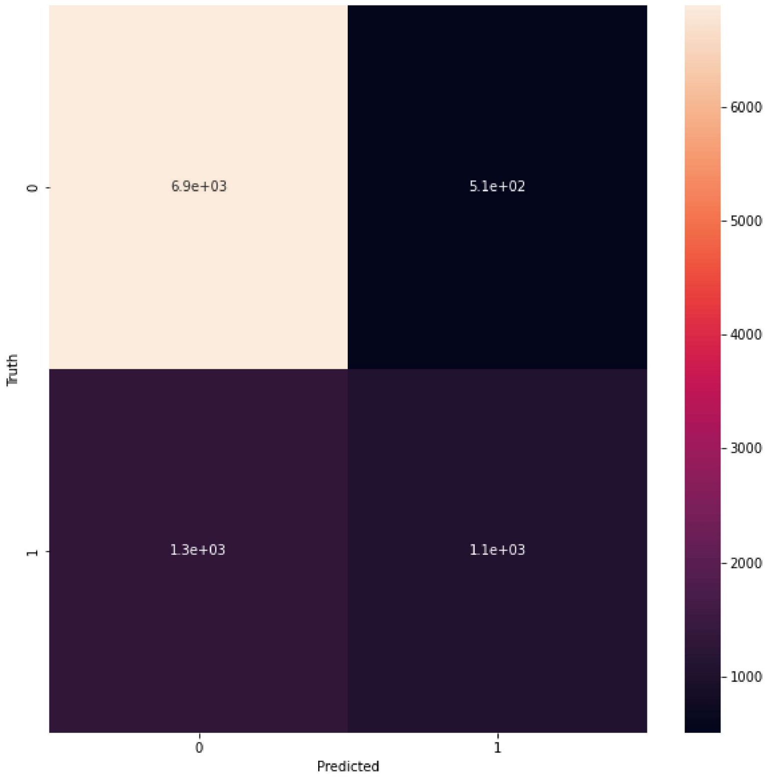
```
     0.813984438984439
```

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

```
array([[6888,  507],
       [1310, 1063]])
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,10))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```



```
from sklearn.metrics import classification_report

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

```
              precision    recall  f1-score   support

           0       0.84      0.93      0.88      7395
           1       0.68      0.45      0.54      2373

    accuracy                           0.81      9768
   macro avg       0.76      0.69      0.71      9768
weighted avg       0.80      0.81      0.80      9768
```

e.SVC

```
from sklearn.svm import SVC
svm_model = SVC(gamma='auto',kernel="linear").fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)

acc_train = svm_model.score(X_train, y_train)
acc_test = svm_model.score(X_test, y_test)
print(acc_train,"accuracy for train")
print(acc_test,"accuracy for test")
```

```
0.76009126009126 accuracy for train
0.7570638820638821 accuracy for test
```

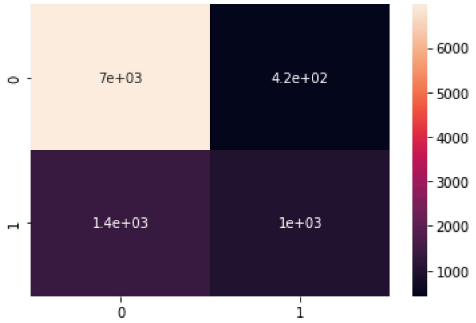
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

```
     array([[6971,  424],
            [1371, 1002]])
```

```
sns.heatmap(cm,annot=True)
```

```
     <matplotlib.axes._subplots.AxesSubplot at 0x7f85f079bee0>
```



```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      7395
           1       0.70      0.42      0.53      2373

    accuracy                           0.82      9768
   macro avg       0.77      0.68      0.71      9768
weighted avg       0.80      0.82      0.80      9768
```

✓ 0s completed at 10:19 PM