AIML421 Assignment4 - Part 2: Performance Metrics in Classification

Dataset: adult.data, adult.test

In [1]:
```
import pandas as pd
adult_data =pd.read_csv("/Users/Jessie/Documents/JupyterNotebook/ass4data/pa
adult_data.head()
```

Out[1]:

	0	1	2	3	4	5	6	7	8	9	10
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0

Perform Initial Data Analysis:

In [2]:
```
adult_data.shape
```

Out[2]:
```
(32561, 15)
```

In [3]:
```
#check missing values
adult_data.isnull().sum()
```

Out[3]:
```
0        0
1     1836
2        0
3        0
4        0
5        0
6     1843
7        0
8        0
9        0
10       0
11       0
12       0
13     583
14       0
dtype: int64
```

In [4]:
```
#check duplicates
if any(adult_data.duplicated()):
    print("Hold on! There are duplications in the dataset")
```

```
    print(adult_data[adult_data.duplicated(keep='first')])
    print(len(adult_data[adult_data.duplicated(keep='first')]))
```

```
Hold on! There are duplications in the dataset
        0                  1       2             3   4                   5  \
4881   25            Private  308144     Bachelors  13       Never-married
5104   90            Private   52386  Some-college  10       Never-married
9171   21            Private  250051  Some-college  10       Never-married
11631  20            Private  107658  Some-college  10       Never-married
13084  25            Private  195994       1st-4th   2       Never-married
15059  21            Private  243368     Preschool   1       Never-married
17040  46            Private  173243       HS-grad   9  Married-civ-spouse
18555  30            Private  144593       HS-grad   9       Never-married
18698  19            Private   97261       HS-grad   9       Never-married
21318  19            Private  138153  Some-college  10       Never-married
21490  19            Private  146679  Some-college  10       Never-married
21875  49            Private   31267       7th-8th   4  Married-civ-spouse
22300  25            Private  195994       1st-4th   2       Never-married
22367  44            Private  367749     Bachelors  13       Never-married
22494  49   Self-emp-not-inc   43479  Some-college  10  Married-civ-spouse
25872  23            Private  240137       5th-6th   3       Never-married
26313  28            Private  274679       Masters  14       Never-married
28230  27            Private  255582       HS-grad   9       Never-married
28522  42            Private  204235  Some-college  10  Married-civ-spouse
28846  39            Private   30916       HS-grad   9  Married-civ-spouse
29157  38            Private  207202       HS-grad   9  Married-civ-spouse
30845  46            Private  133616  Some-college  10            Divorced
31993  19            Private  251579  Some-college  10       Never-married
32404  35            Private  379959       HS-grad   9            Divorced

                       6               7                    8       9  10  \
4881        Craft-repair   Not-in-family                White    Male   0
5104       Other-service   Not-in-family   Asian-Pac-Islander    Male   0
9171      Prof-specialty       Own-child                White  Female   0
11631       Tech-support   Not-in-family                White  Female   0
13084    Priv-house-serv   Not-in-family                White  Female   0
15059    Farming-fishing   Not-in-family                White    Male   0
17040       Craft-repair         Husband                White    Male   0
18555      Other-service   Not-in-family                Black    Male   0
18698    Farming-fishing   Not-in-family                White    Male   0
21318       Adm-clerical       Own-child                White  Female   0
21490    Exec-managerial       Own-child                Black    Male   0
21875       Craft-repair         Husband                White    Male   0
22300    Priv-house-serv   Not-in-family                White  Female   0
22367     Prof-specialty   Not-in-family                White  Female   0
22494       Craft-repair         Husband                White    Male   0
25872  Handlers-cleaners   Not-in-family                White    Male   0
26313     Prof-specialty   Not-in-family                White    Male   0
28230  Machine-op-inspct   Not-in-family                White  Female   0
28522     Prof-specialty         Husband                White    Male   0
28846       Craft-repair         Husband                White    Male   0
29157  Machine-op-inspct         Husband                White    Male   0
30845       Adm-clerical       Unmarried                White  Female   0
31993      Other-service       Own-child                White    Male   0
32404      Other-service   Not-in-family                White  Female   0

       11  12             13      14
4881    0  40         Mexico   <=50K
5104    0  35  United-States   <=50K
9171    0  10  United-States   <=50K
11631   0  10  United-States   <=50K
13084   0  40      Guatemala   <=50K
15059   0  50         Mexico   <=50K
17040   0  40  United-States   <=50K
18555   0  40            NaN   <=50K
```

```
18698    0   40   United-States   <=50K
21318    0   10   United-States   <=50K
21490    0   30   United-States   <=50K
21875    0   40   United-States   <=50K
22300    0   40       Guatemala   <=50K
22367    0   45          Mexico   <=50K
22494    0   40   United-States   <=50K
25872    0   55          Mexico   <=50K
26313    0   50   United-States   <=50K
28230    0   40   United-States   <=50K
28522    0   40   United-States    >50K
28846    0   40   United-States   <=50K
29157    0   48   United-States    >50K
30845    0   40   United-States   <=50K
31993    0   14   United-States   <=50K
32404    0   40   United-States   <=50K
24
```

In [5]:
```
adult_data.drop_duplicates(keep='first')
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female
...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female

32537 rows × 15 columns

In [6]:
```
adult_data.dtypes
```

Out[6]:
```
0      int64
1     object
2      int64
3     object
4      int64
5     object
6     object
7     object
8     object
9     object
10     int64
11     int64
12     int64
13    object
14    object
dtype: object
```

In [7]:
```
adult_data[1].unique()
```

Out[7]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', nan, ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

In [8]:
```
adult_data[3].unique()
```

Out[8]:
```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

In [9]:
```
print(adult_data[5].unique())
print(adult_data[6].unique())
print(adult_data[7].unique())
print(adult_data[8].unique())
print(adult_data[9].unique())
print(adult_data[13].unique())
print(adult_data[14].unique())
```

```
[' Never-married' ' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' ' Widowed']
[' Adm-clerical' ' Exec-managerial' ' Handlers-cleaners' ' Prof-specialty'
 ' Other-service' ' Sales' ' Craft-repair' ' Transport-moving'
 ' Farming-fishing' ' Machine-op-inspct' ' Tech-support' nan
 ' Protective-serv' ' Armed-Forces' ' Priv-house-serv']
[' Not-in-family' ' Husband' ' Wife' ' Own-child' ' Unmarried'
 ' Other-relative']
[' White' ' Black' ' Asian-Pac-Islander' ' Amer-Indian-Eskimo' ' Other']
[' Male' ' Female']
[' United-States' ' Cuba' ' Jamaica' ' India' nan ' Mexico' ' South'
 ' Puerto-Rico' ' Honduras' ' England' ' Canada' ' Germany' ' Iran'
 ' Philippines' ' Italy' ' Poland' ' Columbia' ' Cambodia' ' Thailand'
 ' Ecuador' ' Laos' ' Taiwan' ' Haiti' ' Portugal' ' Dominican-Republic'
 ' El-Salvador' ' France' ' Guatemala' ' China' ' Japan' ' Yugoslavia'
 ' Peru' ' Outlying-US(Guam-USVI-etc)' ' Scotland' ' Trinadad&Tobago'
 ' Greece' ' Nicaragua' ' Vietnam' ' Hong' ' Ireland' ' Hungary'
 ' Holand-Netherlands']
[' <=50K' ' >50K']
```

In [10]:
```
len(adult_data[2].unique())
```

Out[10]:
```
21648
```

Column 3(degrees) is ordinal - can use ordinal encoding however it looks like column 4 is the encoded value of column 3. so this column is dropped. others are nominal -

requirs one hot encoding Target variable has 2 unique value - can use label encoding

Next step - Initial Preproces the data - drop irelavent columns:

To prevent data leakage, we split the data into training and test set first

In [11]:
```
data=adult_data
data.columns=['F1','F2','F3','F4','F5','F6','F7','F8','F9','F10','F11','F12'
data.dropna(axis=0, inplace=True)
data.head()
```

Out[11]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0

In [12]:
```
data=data.drop(['F4'],axis=1)
data.head()
```

Out[12]:

	F1	F2	F3	F5	F6	F7	F8	F9	F10	F11	F12	F13
0	39	State-gov	77516	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40
1	50	Self-emp-not-inc	83311	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13
2	38	Private	215646	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40
3	53	Private	234721	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40
4	28	Private	338409	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40

In [13]:
```
from sklearn.model_selection import train_test_split
y_train=data['salary']
X_train=data.drop(['salary'], axis=1)
X_train.shape
```

Out[13]: (30162, 13)

In [14]:
```
y_train=y_train.str.replace(" ","")
```

```
y_train.head()
```

Out[14]:
```
0    <=50K
1    <=50K
2    <=50K
3    <=50K
4    <=50K
Name: salary, dtype: object
```

In [15]:
```
from sklearn.preprocessing import LabelEncoder
salary_encoder=LabelEncoder()
y_train=salary_encoder.fit_transform(y_train)
y_train
```

Out[15]:
```
array([0, 0, 0, ..., 0, 0, 1])
```

In [16]:
```
import pandas as pd
adult_test =pd.read_csv("/Users/Jessie/Documents/JupyterNotebook/ass4data/pa
```

In [17]:
```
test_data=adult_test
test_data.columns=['F1','F2','F3','F4','F5','F6','F7','F8','F9','F10','F11',
test_data.head()
```

Out[17]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F1
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	
4	18	NaN	103497	Some-college	10	Never-married	NaN	Own-child	White	Female	0	

In [18]:
```
test_data=test_data.drop(['F4'],axis=1)
```

In [19]:
```
test_data["salary"]=test_data["salary"].str.replace(".","")
test_data["salary"].head()
```

```
/var/folders/2f/n9n8cd5n1kz_kjfdg62nql0c0000gp/T/ipykernel_10633/799116048.p
y:1: FutureWarning: The default value of regex will change from True to Fals
e in a future version. In addition, single character regular expressions wil
l *not* be treated as literal strings when regex=True.
  test_data["salary"]=test_data["salary"].str.replace(".","")
```

Out[19]:
```
0    <=50K
1    <=50K
2     >50K
3     >50K
4    <=50K
Name: salary, dtype: object
```

In [20]:
```
y_test=test_data['salary']
X_test=test_data.drop(['salary'], axis=1)
X_test.shape
```

Out[20]: (16281, 13)

In [21]:
```
y_test=y_test.str.replace(" ","")
y_test=salary_encoder.transform(y_test)
y_test
```

Out[21]: array([0, 0, 1, ..., 0, 0, 1])

In [22]:
```
import numpy as np

from sklearn.compose import ColumnTransformer
from sklearn.datasets import fetch_openml
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder, OrdinalEnco
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_s
```

In [23]:
```
#code reference: https://scikit-learn.org/stable/auto_examples/compose/plot_

numeric_features = ["F1", "F3","F5","F11","F12","F13"]
numeric_transformer = Pipeline(
    steps=[("imputer", SimpleImputer(strategy="median")), ("scaler", Standar
)

categorical_features = ["F2", "F6","F7","F8","F9","F10","F14"]
categorical_transformer = OneHotEncoder(handle_unknown="ignore")

preprocessor = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, numeric_features),
        ("cat", categorical_transformer, categorical_features),
    ]
)
```

In [24]:
```
#1 KNN
from sklearn.neighbors import KNeighborsClassifier
clf = Pipeline(
    steps=[("preprocessor", preprocessor), ("classifier", KNeighborsClassifi
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

knn_metrics=['KNN',round(acc,2),round(precision,2),round(recall,2),round(f1,
print(knn_metrics)
```

```
['KNN', 0.84, 0.6, 0.67, 0.63, 0.86]
```

In [25]:
```
#2 GaussianNB
from sklearn.naive_bayes import GaussianNB
#!pip install mlxtend
from mlxtend.preprocessing import DenseTransformer
clf = Pipeline(
    steps=[("preprocessor", preprocessor),('to_dense', DenseTransformer()),
```

```
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

nb_metrics=['Naive Bayes',acc,precision,recall,f1,roc_auc]
print(nb_metrics)
```

```
['Naive Bayes', 0.5596707818930041, 0.9378575143005721, 0.34231754768909556,
0.5015643467983035, 0.814778104594228]
```

In [26]:

```
#3 svm
from sklearn.svm import SVC
clf = Pipeline(
    steps=[("preprocessor", preprocessor), ("classifier", SVC(probability=Tr
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

svm_metrics=['SVM',acc,precision,recall,f1,roc_auc]
print(svm_metrics)
```

```
['SVM', 0.8589767213316135, 0.5988039521580864, 0.7535994764397905, 0.667342
7991886409, 0.9001795817711276]
```

In [27]:

```
#4 DT
from sklearn.tree import DecisionTreeClassifier
clf = Pipeline(
    steps=[("preprocessor", preprocessor), ("classifier", DecisionTreeClassi
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

dt_metrics=['Decision Tree',acc,precision,recall,f1,roc_auc]
print(dt_metrics)
```

```
['Decision Tree', 0.8123579632700694, 0.6201248049921997, 0.599396833375219
9, 0.6095846645367411, 0.7459691174136712]
```

In [28]:

```
#5 RF
from sklearn.ensemble import RandomForestClassifier
clf = Pipeline(
```

```
    steps=[("preprocessor", preprocessor), ("classifier", RandomForestClassi
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

rf_metrics=['Random Forest',acc,precision,recall,f1,roc_auc]
print(rf_metrics)
```

```
['Random Forest', 0.8536330692217923, 0.6131045241809673, 0.724869351367968,
0.6643189181574869, 0.9021410763949658]
```

In [29]:

```
#6 AdaBoost
from sklearn.ensemble import AdaBoostClassifier
clf = Pipeline(
    steps=[("preprocessor", preprocessor), ("classifier", AdaBoostClassifier
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

ab_metrics=['AdaBoost',acc,precision,recall,f1,roc_auc]
print(ab_metrics)
```

```
['AdaBoost', 0.8611264664332657, 0.6180447217888716, 0.7500788892395077, 0.6
776906628652887, 0.9139321873638918]
```

In [30]:

```
#7 GradientBoosting
from sklearn.ensemble import GradientBoostingClassifier
clf = Pipeline(
    steps=[("preprocessor", preprocessor), ("classifier", GradientBoostingCl
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

gb_metrics=['GradientBoosting',acc,precision,recall,f1,roc_auc]
print(gb_metrics)
```

```
['GradientBoosting', 0.8700325532829679, 0.6105044201768071, 0.7916385704652
731, 0.6893716970052848, 0.9196526461782235]
```

In [31]:

```
#8 LINEAR
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
clf = Pipeline(
    steps=[("preprocessor", preprocessor),('to_dense', DenseTransformer()),
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

lda_metrics=['Linear Discriminati Analysis',acc,precision,recall,f1,roc_auc]
print(lda_metrics)
```

```
['Linear Discriminati Analysis', 0.8449112462379461, 0.5910036401456058, 0.7
048062015503876, 0.6429076509687455, 0.8927405660762016]
```

In [32]:

```
#9 MLP
from sklearn.neural_network import MLPClassifier
clf = Pipeline(
    steps=[("preprocessor", preprocessor), ("classifier", MLPClassifier(max_
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

mlp_metrics=['Multi-layer perceptron',acc,precision,recall,f1,roc_auc]
print(mlp_metrics)
```

```
['Multi-layer perceptron', 0.8335483078434985, 0.642225689027561, 0.64931650
893796, 0.645751633986928, 0.8852386335099564]
```

In [33]:

```
#10 Logistic Regression
from sklearn.linear_model import LogisticRegression
clf = Pipeline(
    steps=[("preprocessor", preprocessor), ("classifier", LogisticRegression
)
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)

acc=accuracy_score(y_pred, y_test)
precision=precision_score(y_pred, y_test)
recall=recall_score(y_pred, y_test)
f1=f1_score(y_pred, y_test)
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob[:,1], pos_label=1)
roc_auc = auc(fpr, tpr)

lr_metrics=['Logistic Regression',acc,precision,recall,f1,roc_auc]
print(lr_metrics)
```

```
['Logistic Regression', 0.8525889073152755, 0.6021840873634945, 0.7269303201
506592, 0.658703071672355, 0.9050184934618937]
```

In [34]:

```
metrics=[knn_metrics,nb_metrics,svm_metrics,dt_metrics,rf_metrics,ab_metrics
```

```
output=pd.DataFrame(columns=['Algorithm','ACC','Precision','recall','f1','au
pd.set_option('display.float_format', '{:.2f}'.format)
output
```

Out[34]:

	Algorithm	ACC	Precision	recall	f1	auc
0	KNN	0.84	0.60	0.67	0.63	0.86
1	Naive Bayes	0.56	0.94	0.34	0.50	0.81
2	SVM	0.86	0.60	0.75	0.67	0.90
3	Decision Tree	0.81	0.62	0.60	0.61	0.75
4	Random Forest	0.85	0.61	0.72	0.66	0.90
5	AdaBoost	0.86	0.62	0.75	0.68	0.91
6	GradientBoosting	0.87	0.61	0.79	0.69	0.92
7	Linear Discriminati Analysis	0.84	0.59	0.70	0.64	0.89
8	Multi-layer perceptron	0.83	0.64	0.65	0.65	0.89
9	Logistic Regression	0.85	0.60	0.73	0.66	0.91

In [ ]: