SURUTHI S

225229141

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv("train_loan.csv")
```

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

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

In [4]:
```
df.shape
```

Out[4]:
```
(614, 13)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

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

Out[6]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [8]:
```
df.Loan_Status.value_counts()
```

Out[8]:
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [9]:
```
df.Dependents.unique()
```

Out[9]:
```
array(['0', '1', '2', '3+', nan], dtype=object)
```

STEP 2

In [10]:
```
type(df.Dependents[0])
```

Out[10]:
```
str
```

In [11]:
```
df.Dependents.unique()
```

Out[11]:
```
array(['0', '1', '2', '3+', nan], dtype=object)
```

In [12]:
```
from sklearn.preprocessing import LabelEncoder
```

In [14]:
```
label_encoder = LabelEncoder()
```

In [15]:
```
df1 = df.astype("str").apply(label_encoder.fit_transform)

df2 = df1.where(~df.isna(),df)

print(label_encoder.classes_)
```

```
['N' 'Y']
```

In [16]:
```
df2
```

Out[16]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
0	0	1	0	0	0	0	423	
1	1	1	1	1	0	0	352	
2	2	1	1	0	0	1	180	
3	3	1	1	0	1	0	131	
4	4	1	0	0	0	0	428	
...	...	...	...	...	...	...	...	...
609	609	0	0	0	0	0	166	
610	610	1	1	3	0	0	320	
611	611	1	1	1	0	0	481	
612	612	1	1	2	0	0	472	
613	613	0	0	0	0	1	352	

614 rows × 13 columns

https://stackoverflow.com/questions/55745402/label-encoding-without-nan-value (https://stackoverflow.com/questions/55745402/label-encoding-without-nan-value)

In [17]:
```
# replacing null with mode
for i in df2.columns:
    df2[i] = df2[i].fillna(df2[i].mode()[0])
```

In [18]:
```
df2.isnull().sum()
```

Out[18]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

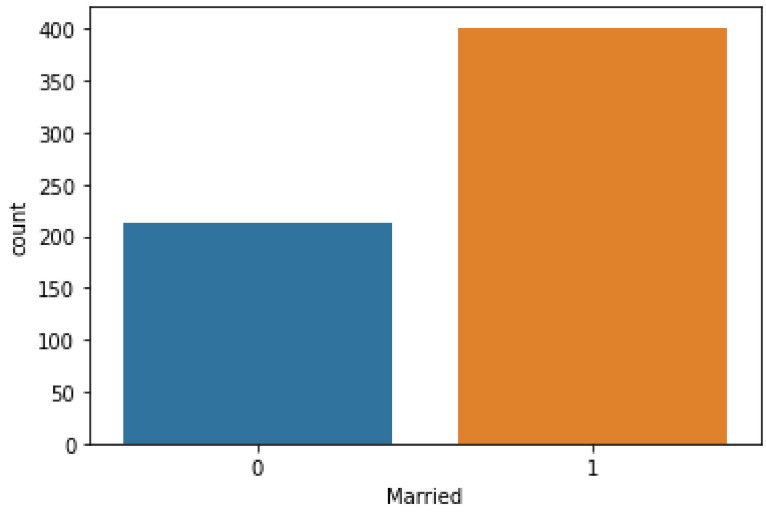
STEP 3 EDA

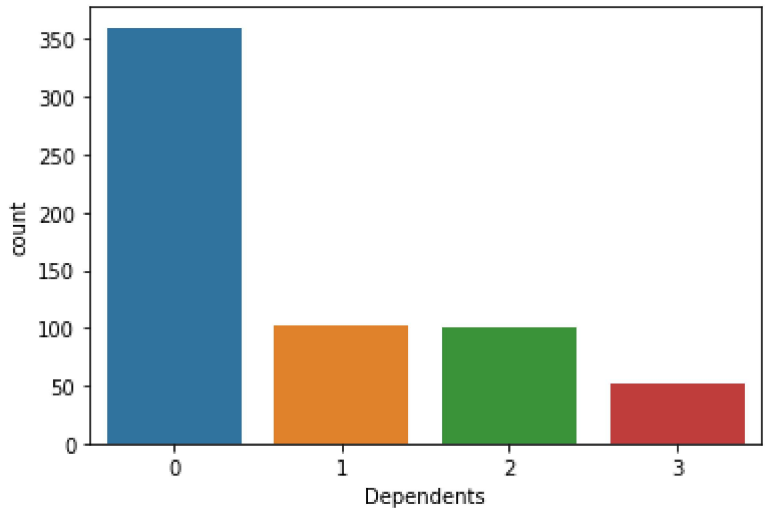
In [21]:
```
df2.columns
```

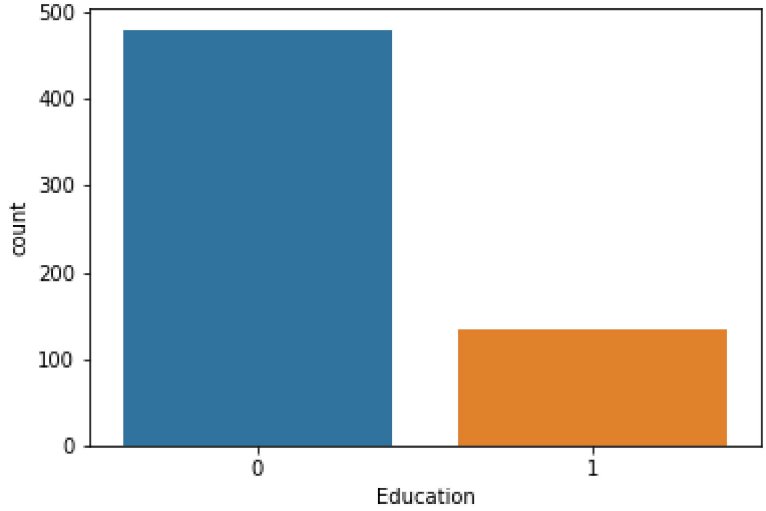
Out[21]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

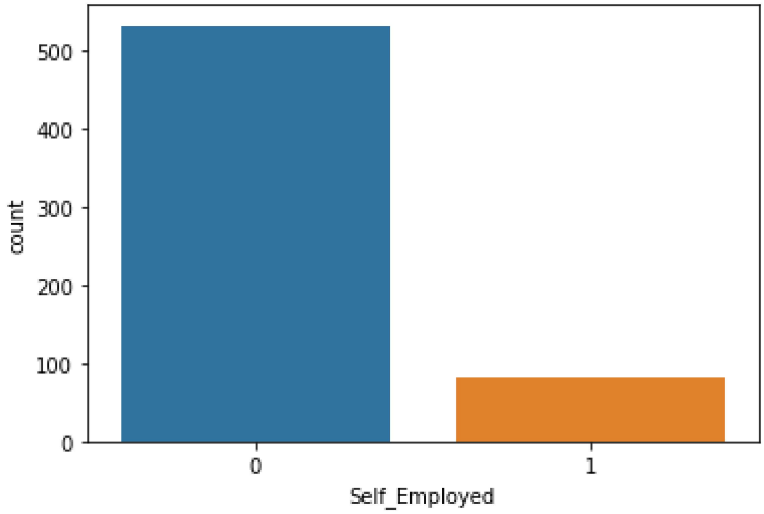
In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [23]:
```
cols = ['Married','Dependents','Education','Self_Employed']
for i in cols:
    sns.countplot(data=df2,x =i )
    plt.show()
```









STEP 4

In [24]:
```
feat = df2.drop("Loan_Status",axis=1)
```

In [25]:
```
label = df2[['Loan_Status']]
```

STEP 5

In [146]:
```
feat = pd.get_dummies(feat)
```

In [147]:
```
feat
```

Out[147]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
0	0	1	0	0	0	0	423	
1	1	1	1	1	0	0	352	
2	2	1	1	0	0	1	180	
3	3	1	1	0	1	0	131	
4	4	1	0	0	0	0	428	
...	...	...	...	...	...	...	...	
609	609	0	0	0	0	0	166	
610	610	1	1	3	0	0	320	
611	611	1	1	1	0	0	481	
612	612	1	1	2	0	0	472	
613	613	0	0	0	0	1	352	

614 rows × 12 columns

STEP 5

In [59]:
```
feat.shape
```

Out[59]: (614, 12)

In [58]:
```
from sklearn.model_selection import train_test_split
```

In [86]:
```
xtrain,xtest,ytrain,ytest = train_test_split(feat,label,test_size=0.20,random_sta
```

In [87]:
```
xtrain.shape
```

Out[87]: (491, 12)

In [88]:
```
xtest.shape
```

Out[88]: (123, 12)

In [89]:
```
from sklearn.svm import LinearSVC
```

In [90]:
```
svc = LinearSVC()
```

In [148]:
```
svc.fit(xtrain,ytrain)
```

```
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\utils
\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  return f(*args, **kwargs)
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\svm\_b
ase.py:986: ConvergenceWarning: Liblinear failed to converge, increase the numb
er of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[148]:
```
LinearSVC()
```

In [149]:
```
ypred = svc.predict(xtest)
```

In [150]:
```
from sklearn.metrics import classification_report
```

In [151]:
```
print(classification_report(ytest,ypred))
```

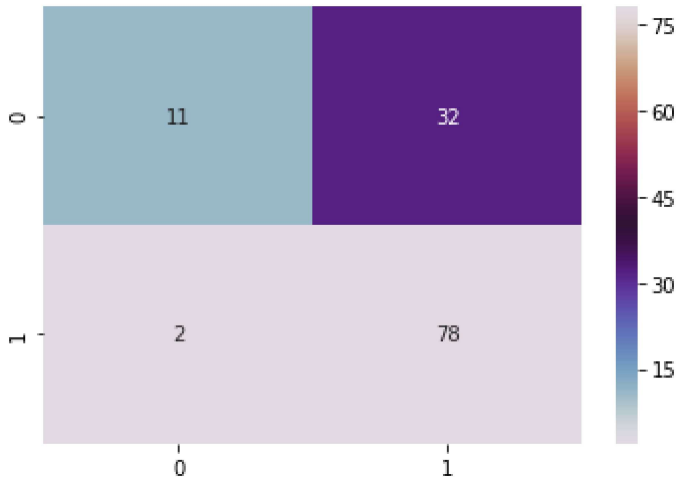
```
              precision    recall  f1-score   support

           0       1.00      0.07      0.13        43
           1       0.67      1.00      0.80        80

    accuracy                           0.67       123
   macro avg       0.83      0.53      0.47       123
weighted avg       0.78      0.67      0.57       123
```

In [95]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(ytest,ypred)
cm
```

Out[95]:
```
array([[11, 32],
       [ 2, 78]], dtype=int64)
```

In [98]:
```
sns.heatmap(cm,annot=True,cmap='twilight')
```

Out[98]: <AxesSubplot:>



STEP 7

In [99]:
```
from sklearn.linear_model import LogisticRegression
```

In [100]:
```
log = LogisticRegression()
```

In [101]:
```
log.fit(xtrain,ytrain)
```

```
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\utils
\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  return f(*args, **kwargs)
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\linear
_model\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=
1):
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 (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-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[101]: LogisticRegression()

In [102]:
```
ypred_log = log.predict(xtest)
```

In [103]:
```
print(classification_report(ytest,ypred_log))
```

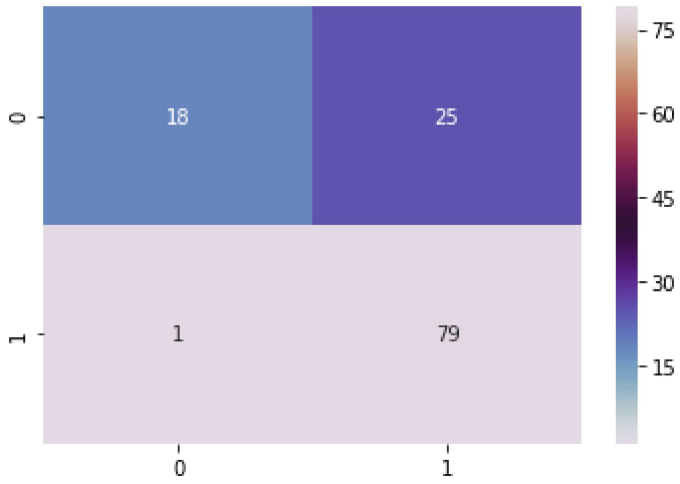
```
              precision    recall  f1-score   support

           0       0.95      0.42      0.58        43
           1       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123
```

In [104]:
```
cm_log = confusion_matrix(ytest,ypred_log)
cm_log
```

Out[104]:
```
array([[18, 25],
       [ 1, 79]], dtype=int64)
```

In [105]:
```
sns.heatmap(cm_log,annot=True,cmap='twilight')
```

Out[105]:
```
<AxesSubplot:>
```



In [106]:
```
from sklearn.linear_model import SGDClassifier
```

In [107]:
```
sgd = SGDClassifier()
```

In [108]:
```
sgd.fit(xtrain,ytrain)
```

```
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\utils
\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  return f(*args, **kwargs)
```

Out[108]:
```
SGDClassifier()
```

In [109]:
```
ypred_sgd = sgd.predict(xtest)
```

In [110]:
```
print(classification_report(ytest,ypred_sgd))
```

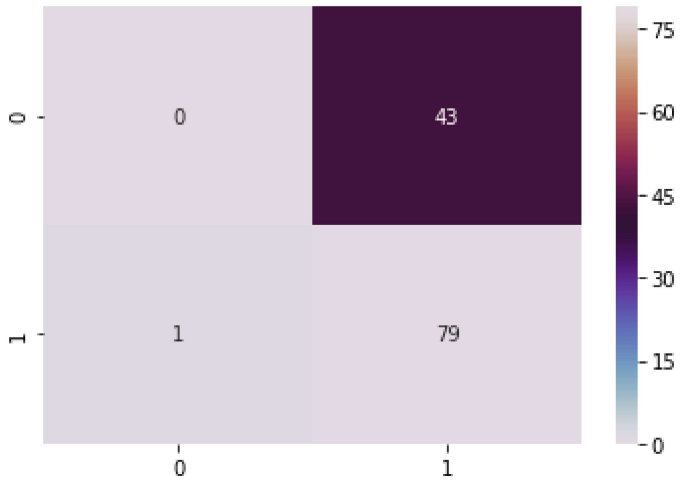
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        43
           1       0.65      0.99      0.78        80

    accuracy                           0.64       123
   macro avg       0.32      0.49      0.39       123
weighted avg       0.42      0.64      0.51       123
```

In [111]:
```
cm_sgd = confusion_matrix(ytest,ypred_sgd)
cm_sgd
```

Out[111]:
```
array([[ 0, 43],
       [ 1, 79]], dtype=int64)
```

In [112]:
```
sns.heatmap(cm_sgd,annot=True,cmap='twilight')
```

Out[112]: <AxesSubplot:>



linear kernel

In [113]:
```
from sklearn.svm import SVC
```

In [114]:
```
svc_linear = SVC(kernel='linear')
```

In [115]:
```
svc_linear.fit(xtrain,ytrain)
```

```
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\utils
\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  return f(*args, **kwargs)
```

Out[115]:
```
SVC(kernel='linear')
```

In [116]:
```
ypred_linear = svc_linear.predict(xtest)
```

In [117]:
```
print(classification_report(ytest,ypred_linear))
```

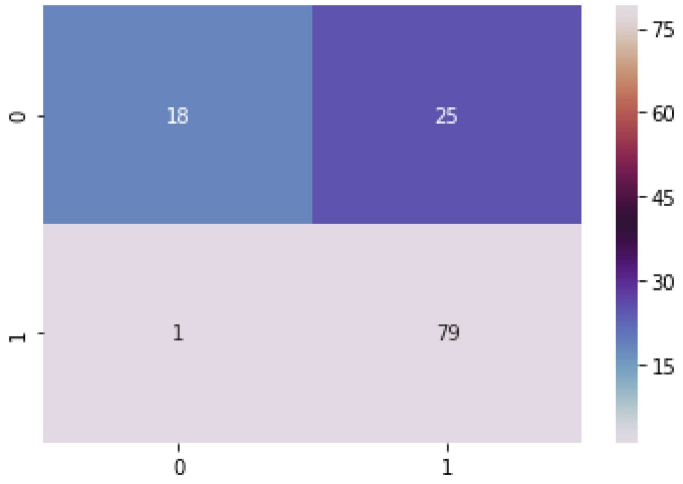
```
              precision    recall  f1-score   support

           0       0.95      0.42      0.58        43
           1       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123
```

In [118]:
```
from sklearn.metrics import confusion_matrix

cm_linear = confusion_matrix(ytest,ypred_linear)
cm_linear
```

Out[118]:
```
array([[18, 25],
       [ 1, 79]], dtype=int64)
```

In [120]:
```
sns.heatmap(cm_linear,annot=True,cmap='twilight')
```

Out[120]: <AxesSubplot:>



poly kernel

In [122]:
```
svc_poly = SVC(kernel='poly')
```

In [123]:
```
svc_poly.fit(xtrain,ytrain)
```

```
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\utils
\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  return f(*args, **kwargs)
```

Out[123]:
```
SVC(kernel='poly')
```

In [124]:
```
ypred_poly = svc.predict(xtest)
```

In [125]:
```
from sklearn.metrics import classification_report
```

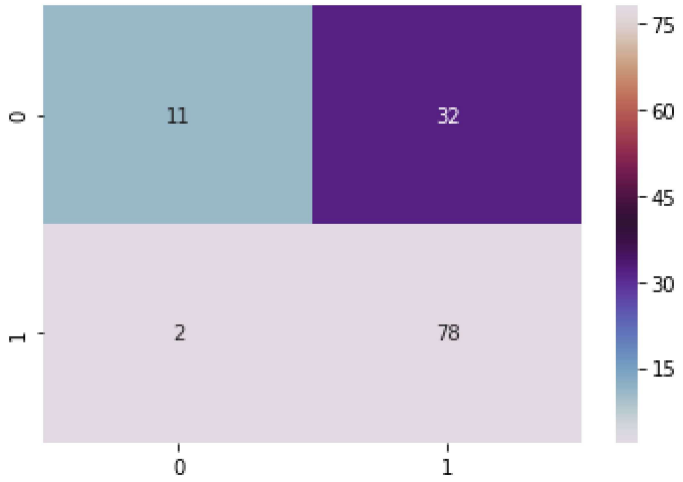
In [126]:
```
print(classification_report(ytest,ypred_poly))
```

```
              precision    recall  f1-score   support

           0       0.85      0.26      0.39        43
           1       0.71      0.97      0.82        80

    accuracy                           0.72       123
   macro avg       0.78      0.62      0.61       123
weighted avg       0.76      0.72      0.67       123
```

In [127]:
```
cm_poly = confusion_matrix(ytest,ypred_poly)
```

In [128]:
```
sns.heatmap(cm_poly,annot=True,cmap='twilight')
```

Out[128]:
```
<AxesSubplot:>
```



RBF KERNEL

In [129]:
```
svc_rbf = SVC(kernel='rbf')
```

In [130]:
```
svc_rbf.fit(xtrain,ytrain)
```

```
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\utils
\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  return f(*args, **kwargs)
```

Out[130]:
```
SVC()
```

In [131]:
```
ypred_rbf = svc_rbf.predict(xtest)
```

In [132]:
```
print(classification_report(ytest,ypred_rbf))
```

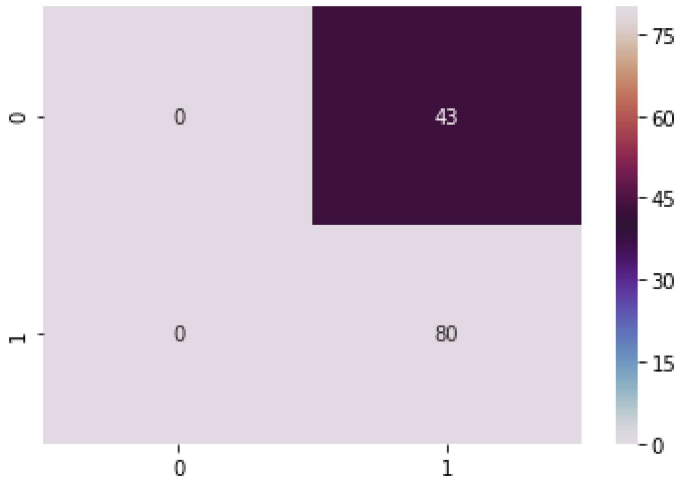
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        43
           1       0.65      1.00      0.79        80

    accuracy                           0.65       123
   macro avg       0.33      0.50      0.39       123
weighted avg       0.42      0.65      0.51       123


C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\metric
s\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\metric
s\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\metric
s\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [133]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(ytest,ypred_rbf)
cm
```

Out[133]:
```
array([[ 0, 43],
       [ 0, 80]], dtype=int64)
```

In [134]:
```
sns.heatmap(cm,annot=True,cmap='twilight')
```

Out[134]: <AxesSubplot:>



kernel : sigmoid

In [ ]:
```
```

In [135]:
```
svc_sigmoid = SVC(kernel='sigmoid')
```

In [136]:
```
svc_sigmoid.fit(xtrain,ytrain)
```

```
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\utils
\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  return f(*args, **kwargs)
```

Out[136]: SVC(kernel='sigmoid')

In [141]:
```
ypred_sigmoid = svc_rbf.predict(xtest)
```

In [142]:
```
print(classification_report(ytest,ypred_sigmoid))
```

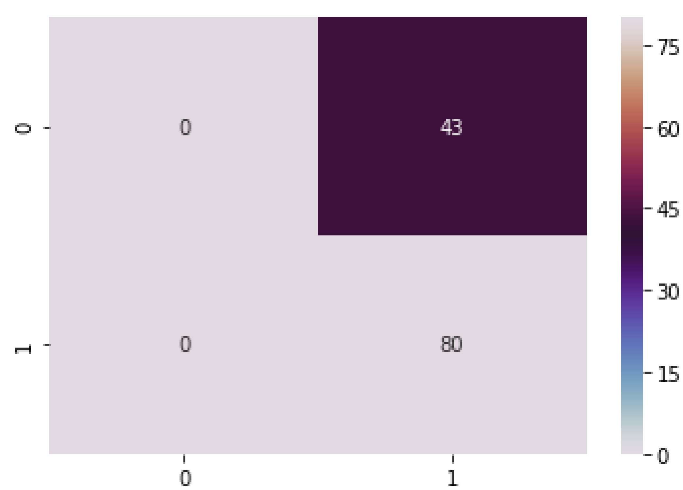
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        43
           1       0.65      1.00      0.79        80

    accuracy                           0.65       123
   macro avg       0.33      0.50      0.39       123
weighted avg       0.42      0.65      0.51       123


C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\metric
s\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\metric
s\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\metric
s\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are il
l-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [143]:
```
from sklearn.metrics import confusion_matrix

cm_sigmoid = confusion_matrix(ytest,ypred_sigmoid)
cm_sigmoid
```

Out[143]:
```
array([[ 0, 43],
       [ 0, 80]], dtype=int64)
```

In [144]:
```
sns.heatmap(cm_sigmoid,annot=True,cmap='twilight')
```

Out[144]: <AxesSubplot:>



interperting the results

Logistic regression and LinearSVC has highest accuracy score and seems to be the best models

In [ ]:

In [ ]: