Lab 6 - Predicting Breast Cancer - III-PART-B

Submitted By
Name: Harsha KG
Register Number: 19112005
Class: **5 BSc Data Science**

Lab Overview

Objectives

A) Compare and understand the difference SVM and anyother two classification algorithm result with respect to Breast cancer Dataset

B)Find a Dataset and Demonstrate SVM Algorithm on it.

Problem Definition

A) Compare and contrast the different evaluation metrices, effect of classification with respect to change in training-test split, random state, pre-processing labels, hyper parameters,alogrithm parameter of decision tree and random forest and interpret the change in result using visualization.List the advantages and Disadvantages of SVM

B) Enumerate your findings/observations on SVM Algorithm

Approach

Imported the Dataset using required libraries from Kaggle([https://www.kaggle.com/uciml/breast-cancer-wisconsin-data (https://www.kaggle.com/uciml/breast-cancer-wisconsin-data)](https://www.kaggle.com/uciml/breast-cancer-wisconsin-data)) to python.Did some pre-processing technique and then build the model using SVM, Decision tree Random forest and compare the difference in the classification metrics and other parameters and after than did hyperparameter tuning for checking the model which gives the highest accuarancy with all the parameter. At the end did some visualization on the results.Liest out the advantages and disadvantages of SVM.

After a thorough understanding in SVM took a dataset(Prima India Diabetics)dataset from kaggle and perfomed SVM.

Sections

1. Lab Overview
2. Imported the Required Libraraies
3. Load the Dataset
4. Basic Inference from Data (Data Wrangling)
    A. Finding the dimension of dataset
    B. Getting the Concise summary of Dataframe
    C. Checking the no of null values
    D. Finding the unique values of Dependent varaiable and counting them
    E. Drop unnamed columns
    F. Description of Datset.
    G. Getting to view the correlation on our data set
    H. Dividing the Columns into Dependent(Y) and Independent One(X)
    I. Plotting target column

1. Train-Test Split

1. SVM
    A. Classification Report
    B. Confusion Matrix
    C. Finding the Accuracy by changing the Parameter
    D. Histogram of Accuracy
    E. HyperParameter Tuning

1. Conclusion

References

1. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html)
2. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)
3. https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python (https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python)
4. https://www.kaggle.com/uciml/breast-cancer-wisconsin-data (https://www.kaggle.com/uciml/breast-cancer-wisconsin-data)
5. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
6. https://scikit-learn.org/stable/modules/tree.html (https://scikit-learn.org/stable/modules/tree.html)
7. https://www.kaggle.com/uciml/pima-indians-diabetes-database (https://www.kaggle.com/uciml/pima-indians-diabetes-database)

About The Datset

The datasets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome.

Independent variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Pregnancies-Number of times pregnant

Glucose-Plasma glucose concentration a 2 hours in an oral glucose tolerance test

BloodPressure-Diastolic blood pressure (mm Hg)

SkinThickness-Triceps skin fold thickness (mm)

Insulin-2-Hour serum insulin (mu U/ml)

BMI-Body mass index (weight in kg/(height in m)^2)

DiabetesPedigreeFunction-Diabetes pedigree function

Age-Age (years)

Outcome-Class variable (0 or 1) 268 of 768 are 1, the others are 0

Importing Required Libraries

In [49]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import seaborn as sns
import matplotlib.pyplot as plt
```

Loading the Dataset

In [11]:
```
df = pd.read_csv("diabetes.csv")
```

In [12]:
```
df
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outc
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	
...	...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	63	
764	2	122	70	27	0	36.8	0.340	27	
765	5	121	72	23	112	26.2	0.245	30	
766	1	126	60	0	0	30.1	0.349	47	
767	1	93	70	31	0	30.4	0.315	23	

768 rows × 9 columns

Basic Inference from Data (Data Wrangling)

Dimension of Dataset

In [38]:
```
df.shape
```

Out[38]: (768, 9)

Getting the Concise summary of Dataframe

In [39]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Checking the no of null values

In [41]:
```
df.isnull().sum()
df.isna().sum()
```

Out[41]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Finding the unique values of Dependent varaiable and counting them

In [43]:
```
classes = pd.unique(df['Outcome'])
print(classes)
```

```
[1 0]
```

In [45]:
```
df['Outcome'].value_counts()
```

Out[45]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

Description of Datset

In [47]:
```
#Getting a statistical decription of our data
df.describe()
```

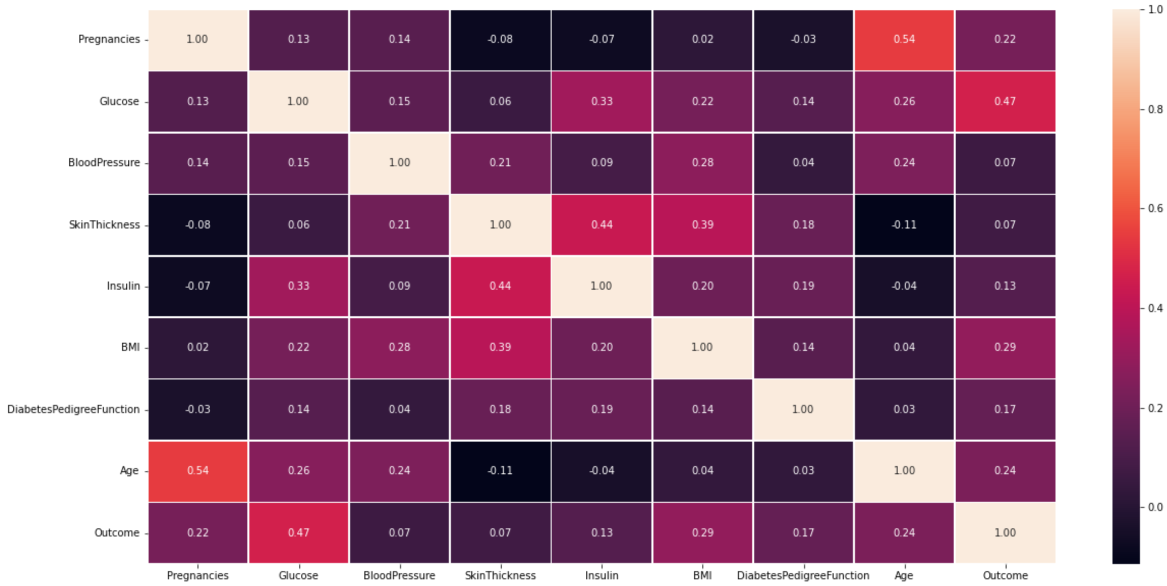
Out[47]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000C
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.4718
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.3313
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078C
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.2437
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.3725
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.6262
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420C

Getting to view the correlation on our data set

In [51]:
```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True, fmt=".2f",annot_kws={"size":10},linewidths=.7)
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x1d1a8915640>



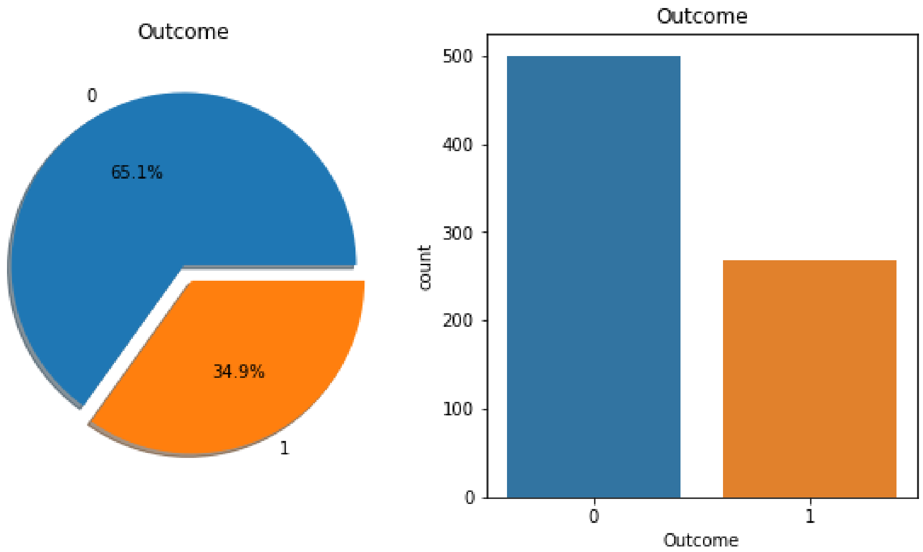
Dividing the Columns into Dependent(Y) and Independent One(X)

In [14]:
```
X = df.drop(['Outcome'], axis = 'columns')
Y = df.Outcome
```

Plotting target column

In [52]:
```
f,ax=plt.subplots(1,2,figsize=(10,5))
df['Outcome'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],shadow=Tr
ax[0].set_title('Outcome')
ax[0].set_ylabel('')
sns.countplot('Outcome',data=df,ax=ax[1])
ax[1].set_title('Outcome')
plt.show()
```

```
C:\Users\HP\anaconda3\anacondaorginal\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only val
id positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
```



Train-Test Split

In [15]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= 0.2)
```

In [53]:
```
# print(X_train.shape);
print(Y_train.shape);
print("\r\n");
print(X_test.shape);
print(Y_test.shape);
```

```
(614,)


(154, 8)
(154,)
```

SVM

In [16]:
```
model = SVC(kernel = 'linear', C = 1)
```

In [17]:
```
model.fit(X_train, Y_train)
```

Out[17]:
```
SVC(C=1, kernel='linear')
```

In [19]:
```
y_pred = model.predict(X_test)
```

Accuracy Score

In [20]:
```
from sklearn.metrics import accuracy_score
acc_score2 = accuracy_score(Y_test, y_pred)
print(acc_score2)
```

```
0.7727272727272727
```

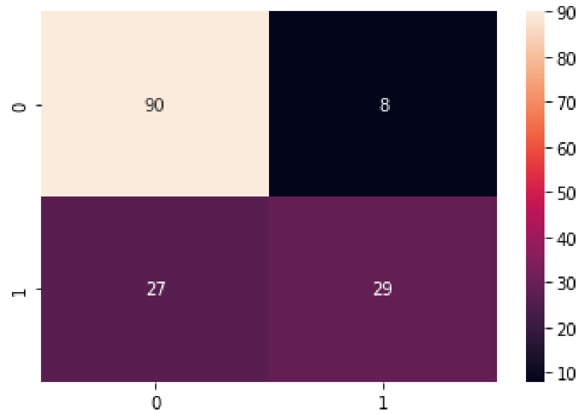
Confusion Matrix

In [21]:
```
from sklearn.metrics import confusion_matrix
Cm = confusion_matrix(Y_test, y_pred)
Cm
```

Out[21]:
```
array([[90,  8],
       [27, 29]], dtype=int64)
```

In [22]:
```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(Y_test, y_pred)
dataframe_conf_matrix = conf_matrix
sns.heatmap(dataframe_conf_matrix, annot=True)
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d1a8731190>
```



classification_report

In [54]:
```
from sklearn.metrics import classification_report
class_report = classification_report(Y_test, y_pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.77      0.92      0.84        98
           1       0.78      0.52      0.62        56

    accuracy                           0.77       154
   macro avg       0.78      0.72      0.73       154
weighted avg       0.77      0.77      0.76       154
```

In [23]:

```
Accuracy = (Cm[0][0] + Cm[1][1]) / (Cm[0][0] + Cm[1][1] + Cm[0][1] + Cm[1][0])
print("Accuracy",Accuracy)
Error_rate = (Cm[0][1] + Cm[1][0]) / (Cm[0][0] + Cm[1][1] + Cm[0][1] + Cm[1][0])
print("Error_rate",Error_rate)
Sensitivity = Cm[0][0]/(Cm[0][0] + Cm[1][0])
print("Sensitivity",Sensitivity)
Specificity = Cm[1][1]/(Cm[1][1] + Cm[0][1])
print("Specificity",Specificity)
Recall = Cm[0][0]/(Cm[0][0] + Cm[1][0])
print("Recall",Recall)
Precision = Cm[0][0]/(Cm[0][0] + Cm[0][1])
print("Precision",Precision)
F1Score = (2*(Precision*Recall))/(Precision + Recall)
print("F1Score",F1Score)
```

```
Accuracy 0.7727272727272727
Error_rate 0.22727272727272727
Sensitivity 0.7692307692307693
Specificity 0.7837837837837838
Recall 0.7692307692307693
Precision 0.9183673469387755
F1Score 0.8372093023255814
```

Finding the Accuracy by changing the Parameter

In [24]:

```
def doSVC(X, Y, test_size = 0.20, randomstate = None, c = 1.0, Kernel = "rbf"):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = test_size, rando
    cls1 = SVC(C = c, kernel = Kernel)
    cls1.fit(X_train,Y_train)
    pred = cls1.predict(X_test)
    acc_score1 = accuracy_score(pred,Y_test)
    return acc_score1
```

In [25]:

```
test_size = [0.30, 0.25, 0.20, 0.10]
random_states = [8, 27, 42]
Regularization_Parameter = [1.0,50.0,100.0]
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
```

In [26]:

```
df2 = pd.DataFrame(columns = ['Test_Size','Random_States','RegularizationParameter','Kerne
```

In [27]:

```
for t_size in test_size:
    for r_state in random_states:
        for RP in Regularization_Parameter:
            for ker in kernels:
                a1 = doSVC(X, Y, t_size, r_state, RP, ker)
                svd = {}
                svd['Test_Size'] = t_size
                svd['Random_States'] = r_state
                svd['Support_Vector_Machine_Accuracy'] = a1
                svd['RegularizationParameter'] = RP
                svd['Kernel'] = ker
                df2 = df2.append(svd, ignore_index = True)
```

In [58]:
```
df2
```

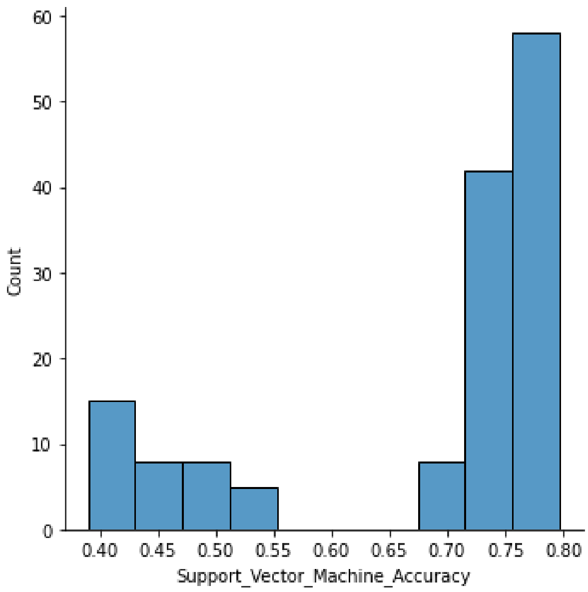
Out[58]:

	Test_Size	Random_States	RegularizationParameter	Kernel	Support_Vector_Machine_Accuracy
0	0.3	8	1.0	linear	0.796537
1	0.3	8	1.0	poly	0.757576
2	0.3	8	1.0	rbf	0.740260
3	0.3	8	1.0	sigmoid	0.519481
4	0.3	8	50.0	linear	0.779221
...	...	...	...	...	...
139	0.1	42	50.0	sigmoid	0.467532
140	0.1	42	100.0	linear	0.688312
141	0.1	42	100.0	poly	0.714286
142	0.1	42	100.0	rbf	0.714286
143	0.1	42	100.0	sigmoid	0.467532

144 rows × 5 columns

In [59]:
```
sns.displot(x = 'Support_Vector_Machine_Accuracy', data = df2)
```

Out[59]: <seaborn.axisgrid.FacetGrid at 0x1d1a92dafa0>



Improving Model

Hyper Paramter Tuning

In [56]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
oHe = OneHotEncoder()
```

In [35]:
```
from sklearn.model_selection import GridSearchCV
parameters = [{'C': [1.0,50.0,100.0],'kernel' : ['linear', 'poly', 'rbf', 'sigmoid']}]
oHe = OneHotEncoder()
grid_search = GridSearchCV(estimator = model,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 10,
                           n_jobs = -1)
grid_search.fit(X_train, Y_train)
best_accuracy_svc = grid_search.best_score_
best_parameters = grid_search.best_params_
print(best_accuracy_svc)
print(best_parameters)
```

```
0.7604442094130089
{'C': 1.0, 'kernel': 'linear'}
```

Conclusion

In this Lab, for Diabetics Dataset accuracy range is around 75 to 80 per(max)

When the Kernel was "sigmoid" the accuracy value range below 50 %

Without changing random state and test size max accuracy is around 76%

When all of the given paramter are changed the max accuracy is around 80%

In [ ]: