UCS 1625 - Foundations of Data Science
Assignment-2

Name:S.Nachammai
Class: CSE - B
Register Number: 185001112

BREAST CANCER CLASSIFICATION

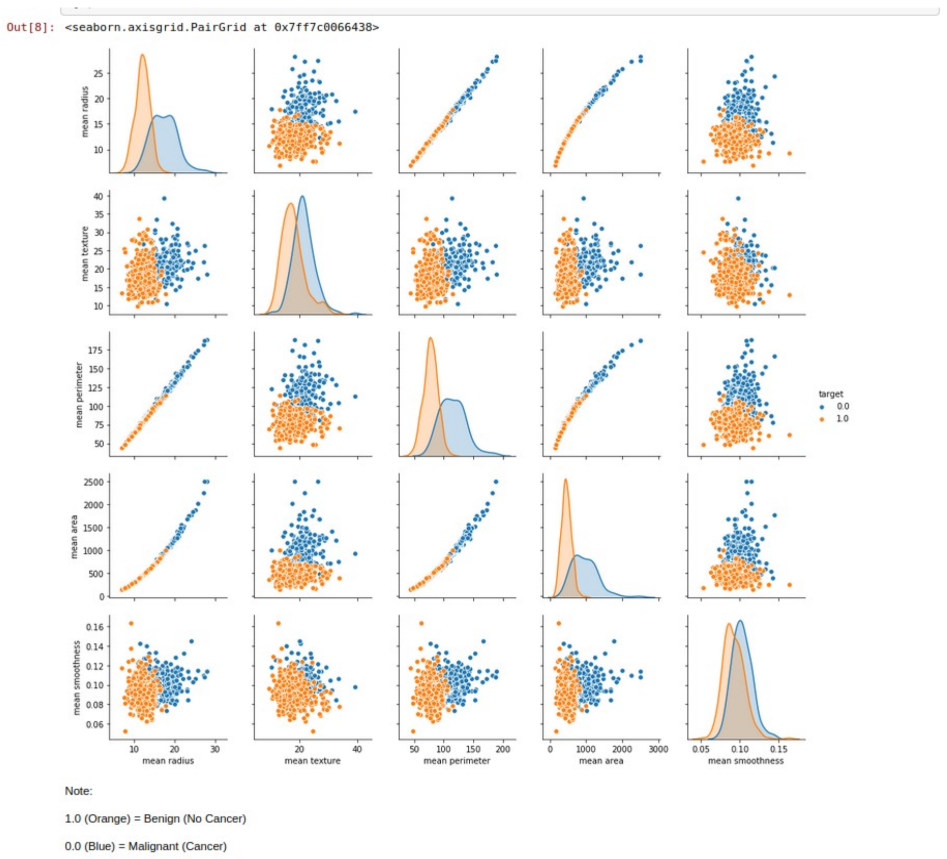
1) Implement Support Vector Machine in Python for a sample dataset

Breast Cancer Classification Using SVM

CODE:

#Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
df_cancer=pd.read_csv('data.csv')

# First 5 variables (features) plotted
sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean texture',
'mean perimeter','mean area','mean smoothness'] )
```



```
Out[8]: <seaborn.axisgrid.PairGrid at 0x7ff7c0066438>
```

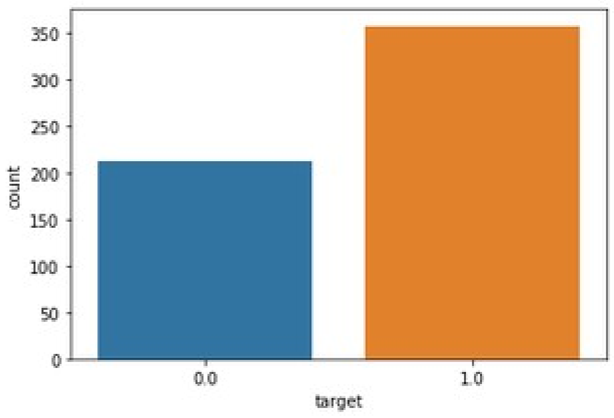
Note:

1.0 (Orange) = Benign (No Cancer)

0.0 (Blue) = Malignant (Cancer)

#Benign and malignant count in our dataset
df_cancer['target'].value_counts()
sns.countplot(df_cancer['target'], label = "Count")

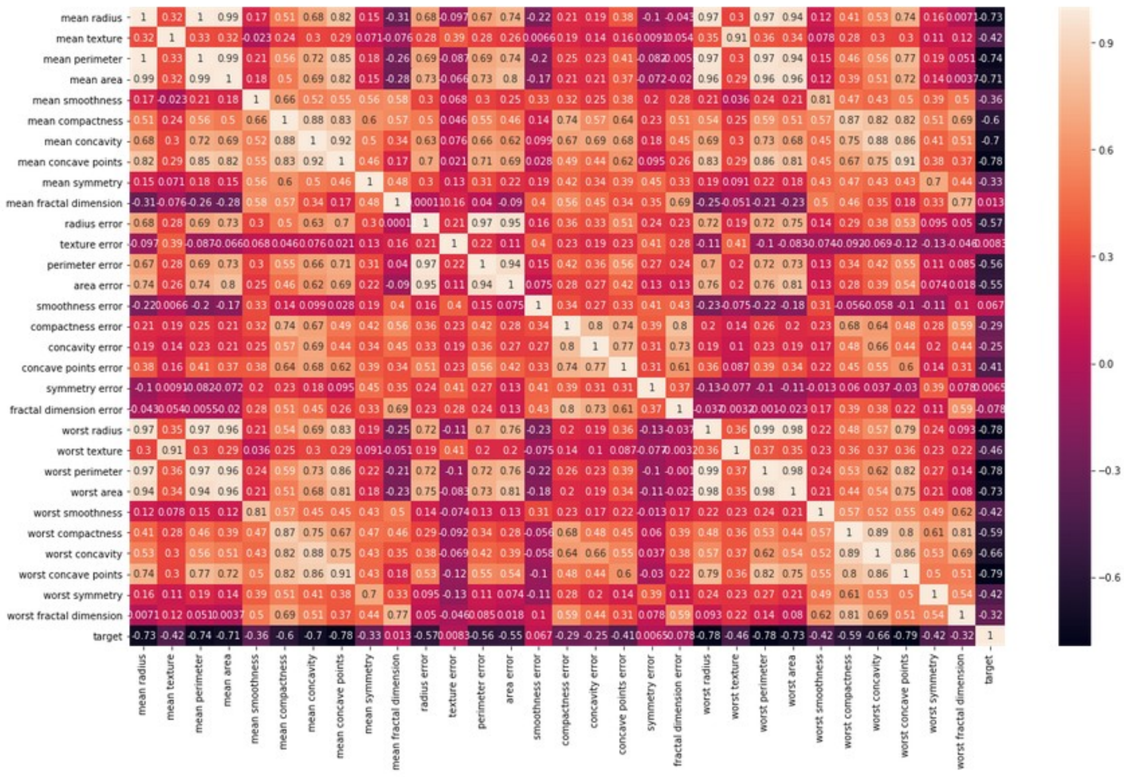
```
In [10]: sns.countplot(df_cancer['target'], label = "Count")

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff7bd944f28>
```



We have 212 - Malignant, and 357 - Benign

#Correlation between features
plt.figure(figsize=(20,12))
sns.heatmap(df_cancer.corr(), annot=True)

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff7be008cc0>
```



#Model Training
X = df_cancer.drop(['target'], axis = 1)
y = df_cancer['target']
#80%-Training & 20%-Testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 20)

#SVM model
from sklearn.svm import SVC
svc_model = SVC()
svc_model.fit(X_train, y_train)
#prediction
y_predict = svc_model.predict(X_test)

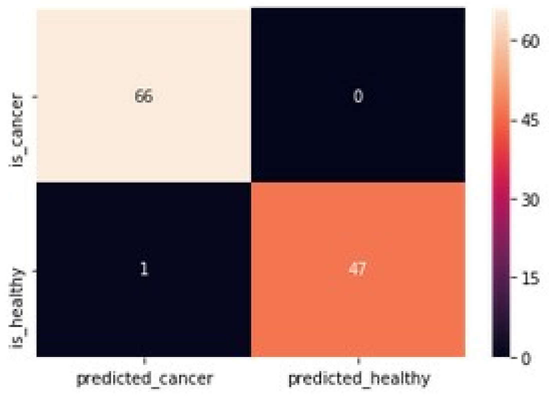
#Confusion Matrix
from sklearn.metrics import classification_report, confusion_matrix
cm = np.array(confusion_matrix(y_test, y_predict, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                         columns=['predicted_cancer','predicted_healthy'])
confusion

Out[30]:

	predicted_cancer	predicted_healthy
is_cancer	66	0
is_healthy	1	47

sns.heatmap(confusion, annot=True)

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff7b4c2b198>
```



#Classification Report
print(classification_report(y_test, y_predict))

```
In [32]: print(classification_report(y_test,y_predict))

              precision    recall  f1-score   support

         0.0       1.00      0.98      0.99        48
         1.0       0.99      1.00      0.99        66

    accuracy                           0.99       114
   macro avg       0.99      0.99      0.99       114
weighted avg       0.99      0.99      0.99       114
```

We have a very high accuracy of **0.99** with only one false prediction.

2)Implement K - Means Clustering for the sample dataset

Breast Cancer Classification Using K – Means Clustering

<u>CODE:</u>

#Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import Kmeans
from sklearn.metrics import classification_report, confusion_matrix

df=pd.read_csv('data.csv')
```

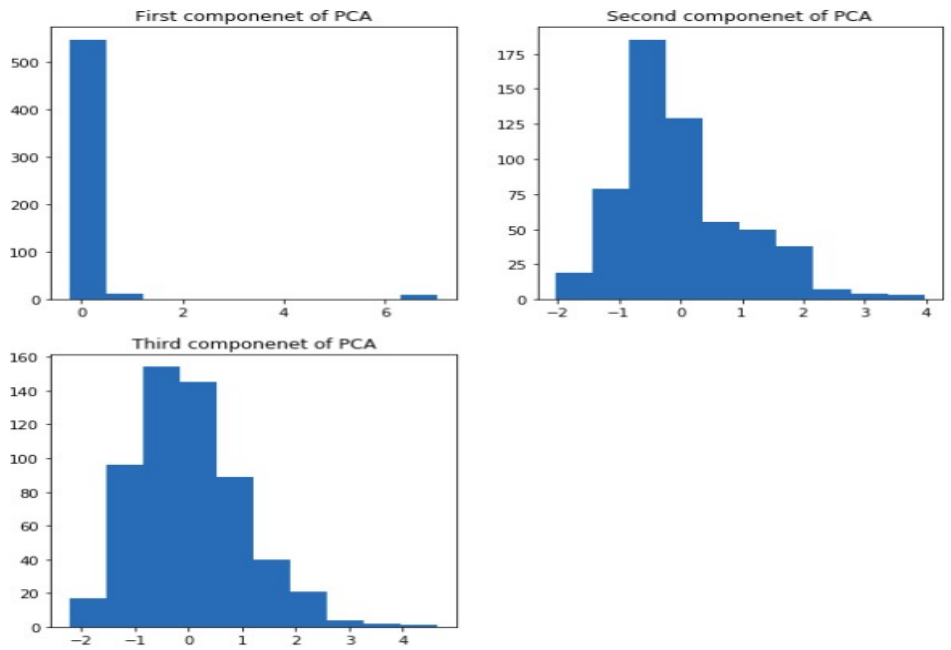
#Scaling the datasets

```
scaler=StandardScaler()
scaled_data=scaler.fit_transform(data)
```

#Visualizing the data

```
x=scaled_data[:,0]
y=scaled_data[:,1]
z=scaled_data[:,2]

plt.figure(figsize=(10,10))
plt.subplot(221)
plt.title('First componenet of PCA')
plt.hist(x)
plt.subplot(222)
plt.title('Second componenet of PCA')
plt.hist(y)
plt.subplot(223)
plt.title('Third componenet of PCA')
plt.hist(z)
```

```
Out[13]: (array([ 17.,  96., 154., 145.,  89.,  40.,  21.,   4.,   2.,   1.]),
          array([-2.22924851, -1.54113476, -0.85302101, -0.16490726,  0.5232064
         9,
                  1.21132023,  1.89943398,  2.58754773,  3.27566148,  3.9637752
         3,
                  4.65188898]),
          <a list of 10 Patch objects>)
```



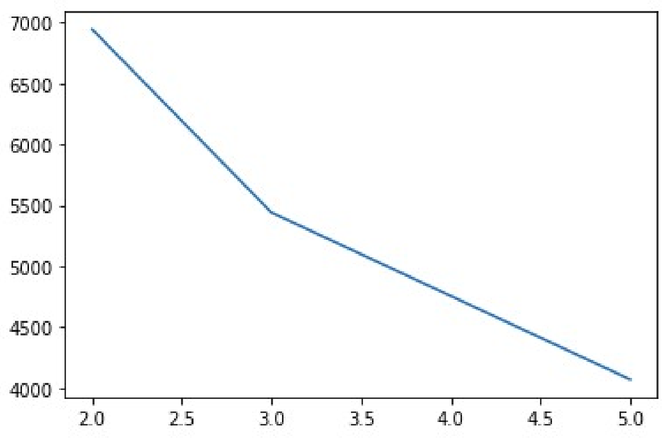
#K-Means Clustering

```
k_means=KMeans(n_clusters=2)
pred_clusters=k_means.fit_predict(trans_data)
k=[2,3,5]
error_list=[]
for k_val in k:
    k_means=KMeans(n_clusters=k_val)
    k_means.fit_predict(trans_data)
    error_list.append(k_means.inertia_)
pd.DataFrame(error_list,k)
```

#Plotting curve between SSE and K

```
plt.plot(k,error_list)
```

```
Out[18]: [<matplotlib.lines.Line2D at 0x7f1d059d4908>]
```



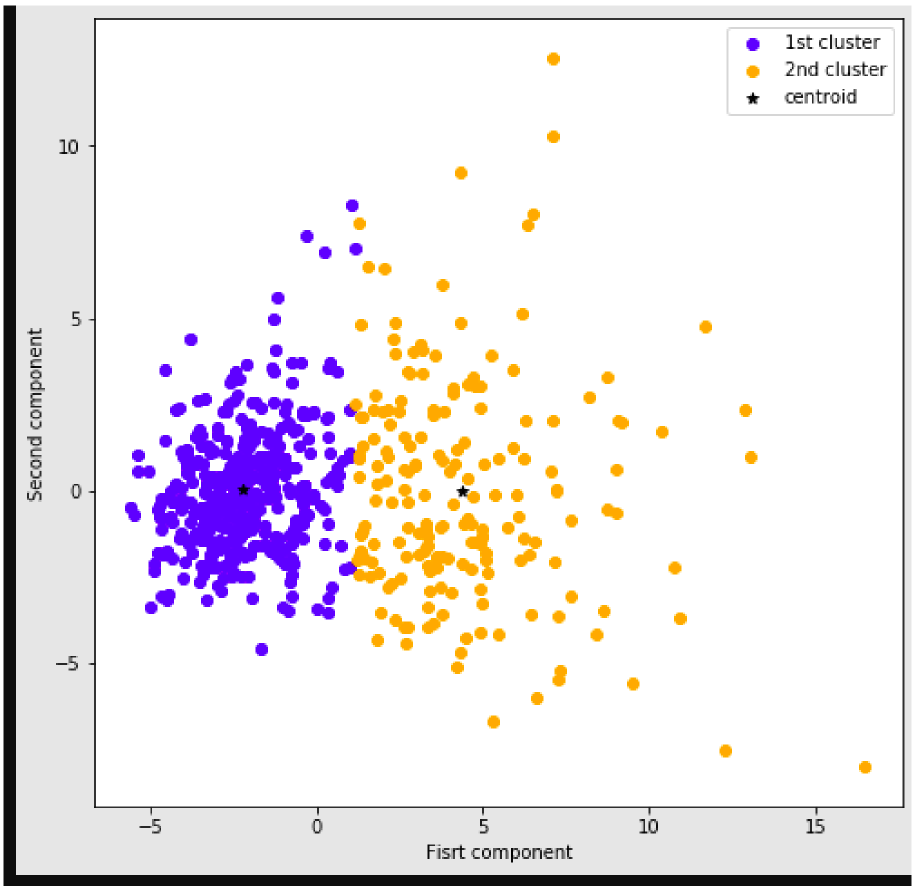
We can see that at K=3 the SSE error is less

#Visualize the data and the centroid using k=3

```
k_means=KMeans(n_clusters=2)
pred_clusters=k_means.fit_predict(trans_data)
centers=k_means.cluster_centers_


df_new=pd.DataFrame(trans_data,columns=['col1','col2','col3'])
df_new['clusters']=pred_clusters
data0=df_new[df_new['clusters']==0]
data1=df_new[df_new['clusters']==1]
```

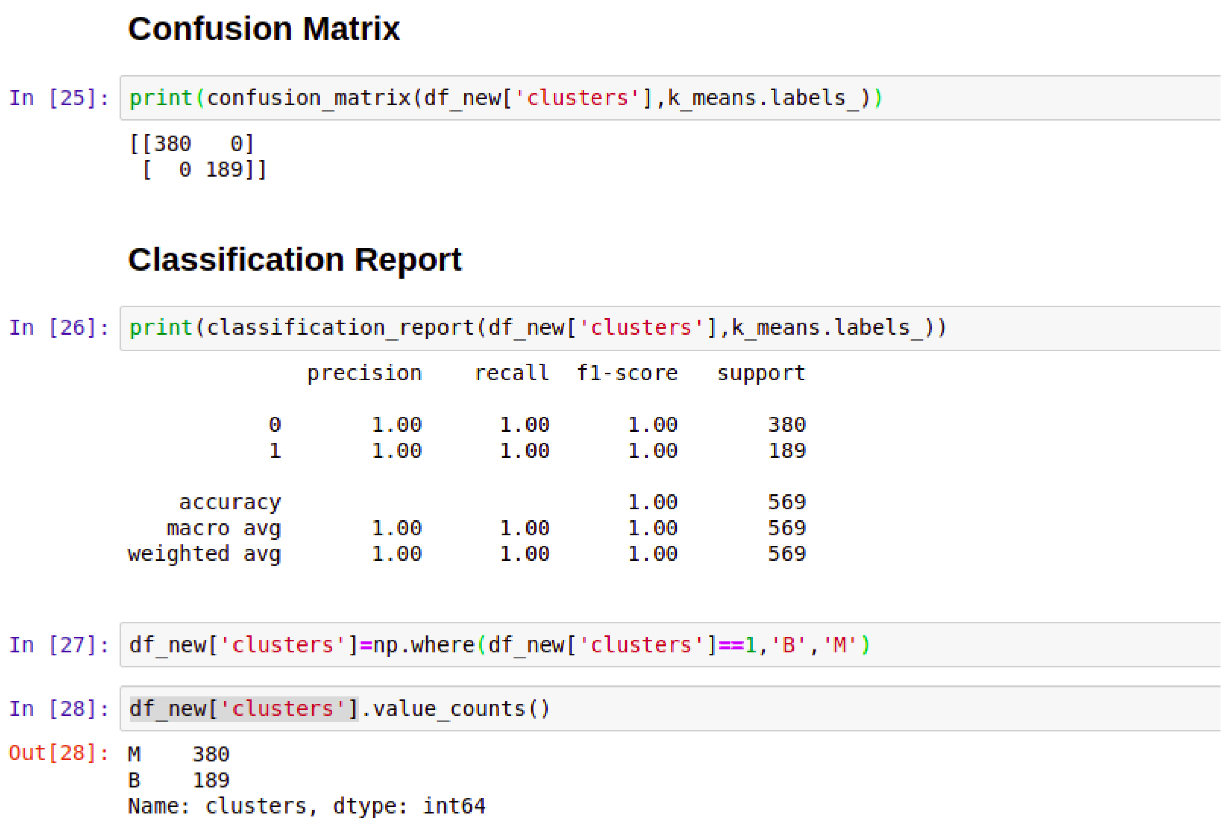
#Plotting clusters

```
plt.figure(figsize=(8,8))
plt.scatter(data0['col1'],data0['col2'],color='blue',label='1st cluster')
plt.scatter(data1['col1'],data1['col2'],color='orange',label='2nd cluster')
plt.scatter(centers[:,0],centers[:,1],marker='*',color='black',label='centroid')
plt.xlabel('Fisrt component')
plt.ylabel('Second component')
plt.legend()
```



```
print(confusion_matrix(df_new['clusters'],k_means.labels_))
print(classification_report(df_new['clusters'],k_means.labels_))
```

OUTPUT:



Confusion Matrix

```
In [25]: print(confusion_matrix(df_new['clusters'],k_means.labels_))

         [[380   0]
          [  0 189]]
```

Classification Report

```
In [26]: print(classification_report(df_new['clusters'],k_means.labels_))

                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00       380
                    1       1.00      1.00      1.00       189

             accuracy                           1.00       569
            macro avg       1.00      1.00      1.00       569
         weighted avg       1.00      1.00      1.00       569


In [27]: df_new['clusters']=np.where(df_new['clusters']==1,'B','M')

In [28]: df_new['clusters'].value_counts()

Out[28]: M    380
         B    189
         Name: clusters, dtype: int64
```

We have obtained a model with a very high accuracy

3)Write down the applications of Hidden Markov Model. Tabulate the differences and similarities of Bayesian and Markov Networks.

Applications of Hidden Markov Model:

-> Human identification using Gait
-> Human action recognition from Time Sequential Images
-> Facial expression identification from videos
-> Video analysis and tracking
-> Modeling and analyzing biological sequences
-> Speech and gesture recognition

Similarities between Bayesian and Markov Networks:

1. Probabilistic transitions between these states
2. Next state determined only by the current state (Markov property)
3. Finite number of states
4. Representation of dependencies

Differences between Bayesian and Markov Networks:

Hidden Markov Model	**Bayesian Network**
Markov network can represent cyclic dependencies	Bayesian network cannot represent cyclic dependencies
Markov network is an undirected graphical model	Bayesian network is a directed graphical model
The main weakness of is their inability to represent induced and non-transitive dependencies; two independent variables will be directly connected by an edge, merely because some other variable depends on both. As a result, many useful independencies go unrepresented in the network.	Bayesian networks use the richer language of directed graphs, where the directions of the arrows permit us to distinguish genuine dependencies from spurious dependencies induced by hypothetical observations.
The general idea of any Markov Process is that "given the present, future is independent of the past"	The general idea of any Bayesian method is that "given the prior, future is independent of the past