Student: Duncan Ferguson
Student Id: 871641260
Class: Comp 4431-1
Assignment: Exercise 10
Date: 11/20/2021
Group: Name: Broken Toe
Group Members: Emma Bright, Mike Santoro

Exercise 10

Part 1:PCA

Principle Component Analysis (PCA) can be used to reduce the time needed to build machine learning models. You are to apply PCA the weather prediction data set using GausianNP in sklearn ash show in the reading for today

First create a table (or chart) of the explained_variance_ratio_+ for the data set. Then run your models for 1,2,3,4,5,6,7 and all components. Create a table (or chart) showing the accuracy as a function of the number of components.

What to turn in: A pdf file containing:

- Your table/chart for explained_variance_ratio_
- Your Table/chart for accuracy for the various number of components, including for the full set of components
- A written answer to: Based on this, how many components do you recommend using for constructing the GaussianNB model for this data set?

178
```
# importing Necessary Libraries
import pandas as pd
from sklearn.naive_bayes import *
from sklearn.metrics import *
from sklearn.model_selection import train_test_split
from sklearn.inspection import permutation_importance
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
```

Create a function to perform the Gaussian NB machine learning algorithm

179
```
def runGaussianNB(x_train, x_test, y_train, y_test):
        global gaussianNB_predicted
        model = GaussianNB()
        model.fit(x_train, y_train)
```

```
        gaussianNB_predicted = model.predict(x_test)
        accuracy = accuracy_score(y_test, gaussianNB_predicted)
        imps = permutation_importance(model, x_test, y_test)
```

Reading in the weather data

180
```
df = pd.read_csv('cleanInfile.csv')
df.head()
```

180

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	W
0	2	13.4	22.9	0.6	13	44.0	13	14
1	2	7.4	25.1	0.0	14	44.0	6	15
2	2	12.9	25.7	0.0	15	46.0	13	15
3	2	9.2	28.0	0.0	4	24.0	9	0
4	2	17.5	32.3	1.0	13	41.0	1	7

181
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142193 entries, 0 to 142192
Data columns (total 20 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Location       142193 non-null  int64
 1   MinTemp        142193 non-null  float64
 2   MaxTemp        142193 non-null  float64
 3   Rainfall       142193 non-null  float64
 4   WindGustDir    142193 non-null  int64
 5   WindGustSpeed  142193 non-null  float64
 6   WindDir9am     142193 non-null  int64
 7   WindDir3pm     142193 non-null  int64
 8   WindSpeed9am   142193 non-null  float64
 9   WindSpeed3pm   142193 non-null  float64
 10  Humidity9am    142193 non-null  float64
 11  Humidity3pm    142193 non-null  float64
 12  Pressure9am    142193 non-null  float64
 13  Pressure3pm    142193 non-null  float64
 14  Cloud9am       142193 non-null  float64
 15  Cloud3pm       142193 non-null  float64
 16  Temp9am        142193 non-null  float64
 17  Temp3pm        142193 non-null  float64
 18  RainToday      142193 non-null  float64
 19  RainTomorrow   142193 non-null  int64
dtypes: float64(15), int64(5)
memory usage: 21.7 MB
```

Create a table of the Explained variance ratio for the data Set

182
```
Y = df['RainTomorrow']
df2 = df.drop('RainTomorrow', axis=1)

x_train, x_test, y_train, y_test = train_test_split(df2, Y, test_size=0.20, random_state=1)
```

```
s_x_train, s_x_test, s_y_train, s_y_test = x_train, x_test, y_train, y_test

runGaussianNB(x_train, x_test, y_train, y_test)

pca = PCA()
pca_x_train = pca.fit_transform(x_train)
pca_x_test = pca.fit_transform(x_test)
explained_variance = pca.explained_variance_ratio_
explained_variance_table = pd.DataFrame(explained_variance, columns=["Explained variance Ratio"])
explained_variance_table
```

182

	Explained variance Ratio
0	0.407655
1	0.169948
2	0.116532
3	0.082357
4	0.065879
5	0.038231
6	0.032713
7	0.023525
8	0.020696
9	0.017119
10	0.009312
11	0.005776
12	0.003771
13	0.002942
14	0.001183
15	0.000945
16	0.000707
17	0.000648
18	0.000061

183
```
total = explained_variance_table["Explained variance Ratio"].sum()
total
```

183
```
1.0
```

Run the Gaussian NB model for 1,2,3,4,5,6,7,8 and all components

184
```
accuracy_list = []
```

```
x_train, x_test = s_x_train, s_x_test

# Run Gaussian with all components
runGaussianNB(x_train, x_test, y_train, y_test)
accuracy_list.append(['All', accuracy_score(y_test, gaussianNB_predicted)])

# Run with 1,2,3,4,5,6,7,8
for i in range(1,9):
        pca= PCA(n_components=i, random_state=42)
        x_train = pca.fit_transform(s_x_train)
        x_test = pca.fit_transform(s_x_test)
        runGaussianNB(x_train, x_test, y_train, y_test)
        accuracy_list.append([str(i), accuracy_score(y_test, gaussianNB_predicted)])

result = pd.DataFrame(accuracy_list, columns=["Number of Components", "Accuracy"])
result
```

184

	Number of Components	Accuracy
0	All	0.812581
1	1	0.798375
2	2	0.824994
3	3	0.825381
4	4	0.830092
5	5	0.829882
6	6	0.817645
7	7	0.813355
8	8	0.810964

Based on this, how many components do you recommend using for constructing the GausianNB model for this data set?

For this data set I recommend using 4 components for the Gaussian Naive Bayes model. This is because it has the highest level of accuracy

185
```
result.sort_values('Accuracy', ascending=False).head(1)
```

185

	Number of Components	Accuracy
4	4	0.830092

Part 2: Clustering Quality

Apply kmeans and dbmeans clustering to these labeled data sets: outfile1.csv
outfile2.csv

outfile3.csv

The first data set contains data from cluster 1, the second from cluster 2, and the third from cluster 3

Calculate the following three metrics on the clustering for kmeans and dbscan:

- homogeneity
- completeness
- adjusted_mutual_info_score

What to turn in (A pdf file containing):

- a Table of your metrics
- a description of what these metrics tell you about the clustering?

186
```
numClusters = 3

outfile = pd.read_csv('outfile.csv')
labels1 = pd.read_csv('outfile1.csv')
labels2 = pd.read_csv('outfile2.csv')
labels3 = pd.read_csv('outfile3.csv')

kmeans = KMeans(n_clusters=numClusters, random_state=0).fit(outfile)
db = DBSCAN(eps=1.5, min_samples=4).fit(outfile)
```

187
```
outfile['predKM'] = kmeans.labels_
outfile['predDB'] = db.labels_
outfile.head()
```

187

	a1	a2	predKM	predDB
0	18.900508	17.738116	0	0
1	9.355166	32.029783	2	1
2	20.111226	10.921999	0	0
3	30.752270	21.326611	1	2
4	9.816053	31.332579	2	1

188
```
labels1['actual'] = 0
labels2['actual'] = 1
labels3['actual'] = 2
labels = labels1.append(labels2, ignore_index = True)
labels = labels.append(labels3, ignore_index = True)
labels.head()
```

188

	a1	a2	actual
0	9.355166	32.029783	0

	a1	a2	actual
1	9.816053	31.332579	0
2	9.125756	4.129040	0
3	8.632544	36.134807	0
4	9.616329	27.776006	0

189
```
df = pd.merge(labels, outfile,  how='left', left_on=['a1','a2'], right_on = ['a1','a2'])
df
```

189

	a1	a2	actual	predKM	predDB
0	9.355166	32.029783	0	2	1
1	9.816053	31.332579	0	2	1
2	9.125756	4.129040	0	0	1
3	8.632544	36.134807	0	2	1
4	9.616329	27.776006	0	2	1
...	...	...	...	...	...
995	29.860049	19.466960	2	1	2
996	28.471657	5.143157	2	0	2
997	30.043580	12.106201	2	0	2
998	28.667936	17.360923	2	0	2
999	30.433842	29.338134	2	1	2

1000 rows × 5 columns

190
```
print('K-Means')
print(f'Homogenity Score: {homogeneity_score(df.actual.to_list(),df.predKM.to_list())}')
print(f'Completeness Score: {completeness_score(df.actual.to_list(), df.predKM.to_list())}')
print(f'Adjusted Mutual Info Score: {adjusted_mutual_info_score(df.actual.to_list(), df.predKM.to_list())
```

```
K-Means
Homogenity Score: 0.3511815902654452
Completeness Score: 0.36120342674835665
Adjusted Mutual Info Score: 0.35492679063609317
```

191
```
print('DB Scan')
print(f"Homogenity Score: {homogeneity_score(df.actual.to_list(),df.predDB.to_list())}")
print(f'Completeness Score: {completeness_score(df.actual.to_list(), df.predDB.to_list())}')
print(f'Adjusted Mutual Info Score: {adjusted_mutual_info_score(df.actual.to_list(), df.predDB.to_list())
```

```
DB Scan
```

```
Homogenity Score: 1.0
Completeness Score: 0.99387940769096
Adjusted Mutual Info Score: 0.9969216318008042
```

A description of what these metrics tell you about the clustering?

- Homogenity score: The measure that each cluster contains only members of a single class
- Completeness Score: The Measure that all members of a given class are assigned to the same cluster.
- Adjusted Mutual Info Score: The combination of the homogenity score and the completeness score. Because they are both bounded by 0.0 and above by 1.0 (higher is better)