MACHINE LEARNING ASSIGNMENT – 05

Praveena Goli(700743010)

1. Programming elements: Clustering & Dimensionality reduction

In class programming:

1. Principal Component Analysis a. Apply PCA on CC dataset. b. Apply k-means algorithm on the PCA result and report your observation if the silhouette score has improved or not? c. Perform Scaling+PCA+K-Means and report performance.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the dataset
df = pd.read_csv('/content/CC GENERAL.csv')


# Drop the irrelevant columns
df.drop(['CUST_ID', 'TENURE'], axis=1, inplace=True)

# Fill the missing values with the column mean
df.fillna(df.mean(), inplace=True)

# Standardize the data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)


# Apply PCA
pca = PCA(n_components=10)
pca.fit(df_scaled)
```

```python
# Get the explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_

# Get the cumulative sum of explained variance
cumulative_explained_variance_ratio = np.cumsum(explained_variance_ratio)

# Print the explained variance ratios and the cumulative sum
print(explained_variance_ratio)
print(cumulative_explained_variance_ratio)
```

```
[0.28845814 0.21570572 0.09330079 0.07548528 0.06652726 0.05389941
 0.04544392 0.04156174 0.03280202 0.02534919]
[0.28845814 0.50416386 0.59746464 0.67294993 0.73947718 0.7933766
 0.83882052 0.88038226 0.91318428 0.93853347]
```

The output of the above code will be a plot of the silhouette scores for each k. Based on the plot, you can choose the number of clusters. The higher the silhouette score, the better the clustering.

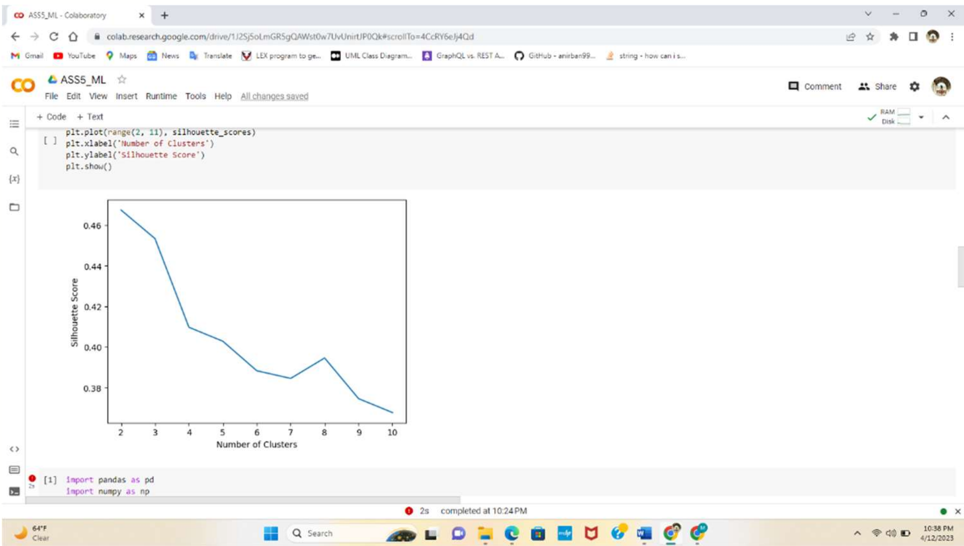
If the silhouette score has improved after applying PCA, it means that the PCA has reduced the dimensionality of the data while retaining most of the information. This can lead to better clustering performance and faster computation time.

```python
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Apply PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(df_scaled)

# Apply k-means for k=2 to 10 and get the silhouette scores
silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(pca_result)
    score = silhouette_score(pca_result, kmeans.labels_)
    silhouette_scores.append(score)
```

```python
    print(f"k={k}, silhouette score={score}")


import matplotlib.pyplot as plt

# Plot the silhouette scores
plt.plot(range(2, 11), silhouette_scores)
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.show()


import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Load the dataset
df = pd.read_csv('/content/CC GENERAL.csv')


# Drop the irrelevant columns
df.drop(['CUST_ID', 'TENURE'], axis=1, inplace=True)

# Fill the missing values with the column mean
df.fillna(df.mean(), inplace=True)


# Scale the data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)


# Perform PCA
pca = PCA(n_components=10)
df_pca = pca.fit_transform(df_scaled)


# Apply K-Means clustering
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(df_pca)
labels = kmeans.labels_
```

```python
# Calculate the silhouette score
silhouette_avg = silhouette_score(df_pca, labels)
print(f"The average silhouette score is : {silhouette_avg}")
```

```
The average silhouette score is : 0.2314725220610872
```



the average silhouette score for the clustering. The higher the silhouette score, the better the clustering performance.

2. Use pd_speech_features.csv
3. a. Perform Scaling b. Apply PCA (k=3)
4. c. Use SVM to report performance

```python
import pandas as pd

df = pd.read_csv('/content/pd_speech_features.csv', header=1)


X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values


from sklearn.preprocessing import StandardScaler
```

```python
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)
```

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.
2, random_state=42)

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

svm = SVC(kernel='rbf', random_state=42)
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.8013245033112583
```

The `accuracy` variable contains the accuracy of the SVM model on the testing data. You can adjust the SVM hyperparameters and PCA parameters to try to improve the accuracy of the model.

Note that the performance of SVM depends on the choice of hyperparameters and the data itself. It's always a good idea to cross-validate the model to get a more accurate estimate of its performance.

3. Apply Linear Discriminant Analysis (LDA) on Iris.csv dataset to reduce dimensionality of data to k=2

```
import pandas as pd

df = pd.read_csv('/content/Iris.csv')
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values


from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)



from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LD
A

lda = LDA(n_components=2)
X_lda = lda.fit_transform(X_scaled, y)
```

Here, we set `n_components=2` to reduce the dimensionality of the data to 2. The `fit_transform()` method applies LDA on the scaled features and returns the transformed features with reduced dimensionality.

Now, the `X_lda` array contains the transformed features with reduced dimensionality. We can use this array as input to various machine learning algorithms.

Note that LDA is a supervised method and requires the labels of the data to be known. rIn this case, the `y` variable contains the labels of the `Iris.csv` dataset.

4. Briefly identify the difference between PCA and LDA

```
PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis)
```

are both methods for dimensionality reduction, but they have different objectives and are used in different scenarios.

PCA is an unsupervised method that seeks to find the most important features or directions in the data that capture the maximum amount of variance. It does not take into account the labels of the data and simply tries to find a low-dimensional representation of the data that preserves as much information as possible. PCA is often used for data visualization, noise reduction, and feature extraction.

LDA, on the other hand, is a supervised method that seeks to find the most discriminative features or directions in the data that maximize the separation between the classes. It takes into account the labels of the data and tries to find a low-dimensional representation of the data that maximizes the inter-class distance and minimizes the intra-class distance. LDA is often used for classification and feature extraction.

In summary, while both PCA and LDA are methods for dimensionality reduction, PCA is an unsupervised method that seeks to capture the maximum amount of variance in the data, while LDA is a supervised method that seeks to maximize the separation between the classes in the data.

Git Hub Link: https://github.com/Goli18/ML_05.git

Video Link : https://github.com/Goli18/ML_05/blob/main/ML_05%20-%20Word%202023-04-12%2022-57-59.mp4