NAME: Dibyojyoti Dutta

EMAIL: dduttaoffice184@gmail.com

ASSIGNMENT NAME: KNN & PCA

DRIVE LINK: link

Assignment Code: DA-AG-016

KNN & PCA Assignment

Instructions: Carefully readeach question. UseGoogleDocs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 200

Question 1: What is K-Nearest Neighbors (KNN) and how does it work in both classification and regression problems?

Answer:

K-Nearest Neighbors is a supervised machine learning algorithm used for classification and regression tasks. Its called lazy learning because it doesn't explicitly build a model during training it just stores the data and makes predictions at query time.
*In classification, prediction is based on the majority class of neighbors.
*In regression, prediction is based on the average value of neighbors.

Question 2: What is the Curse of Dimensionality and how does it affect KNN performance?

Answer:

The curse of dimensionality refers to the problems that arise when working with high-dimensional data.
*Nearest neighbor becomes less meaningful
*Increased risk of overfitting
*Higher computational cost

Question 3: What is Principal Component Analysis (PCA)? How is it different from feature selection?

Answer:

Principal Component Analysis(PCA) is a dimensionally reduction technique that transforms the original features into a new set of features called principal components.
Feature selection- Selecting a subset of the original feature that are most relevant.
Keeps some of the original feature.
Remove irrelevant/redundant features.
Easy to interpret.

Question 4: What are eigenvalues and eigenvectors in PCA, and why are they important?

Answer:

$Av = \lambda v$
Here v = eigenvector
λ = eigenvalue (a scalar)

Importance:
*Eigenvectors define principal components
*Eigenvalues tell us the importance
*Dimensionally reduction

Question5: How do KNN and PCA complement each other when applied in a single pipeline?
Answer:

*PCA transforms high-dimensional data into a smaller set of informative features.
*KNN then classifies or predicts based on distance in this reduced space.
*Both complement each other because PCA reduces dimensionally and makes distance more meaningful, while KNN use those distance for prediction.
*The combo reduces curse of dimensionality, improves efficiency, and often increases accuracy.

Dataset:
Use the **Wine Dataset** from `sklearn.datasets.load_wine()`.

Question 6: Train a KNN Classifier on the Wine dataset with and without feature scaling. Compare model accuracy in both cases.
(***Include your Python code and output in the code box below.***
)

Answer:

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
X, y = load_wine(return_X_y=True)
X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, test_size=0.30, random_state=42, stratify=y
)
knn_plain = KNeighborsClassifier(n_neighbors=5)
knn_plain.fit(X_tr, y_tr)
acc_no = knn_plain.score(X_te, y_te)
knn_scaled = make_pipeline(StandardScaler(),
KNeighborsClassifier(n_neighbors=5))
knn_scaled.fit(X_tr, y_tr)
acc_yes = knn_scaled.score(X_te, y_te)
print(f"Accuracy without scaling: {acc_no:.3f}")
print(f"Accuracy with scaling : {acc_yes:.3f}")
```

```
Accuracy without scaling: 0.722
Accuracy with scaling : 0.944
```


Question 7:Train a PCA model on the Wine dataset and print the explained variance

ratio of each principal component.

(***Include your Python code and output in the code box below.***)

Answer:

```
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
wine = load_wine()
X, y = wine.data, wine.target
X_scaled = StandardScaler().fit_transform(X)
pca = PCA()
pca.fit(X_scaled)
print("Explained Variance Ratio for each Principal Component:")
for i, ratio in enumerate(pca.explained_variance_ratio_):
 print(f"PC{i+1}: {ratio:.4f}")
print("\nCumulative Explained Variance:")
print(pca.explained_variance_ratio_.cumsum())
```

```
Explained Variance Ratio for each Principal Component:
PC1: 0.3620
PC2: 0.1921
PC3: 0.1112
PC4: 0.0707
PC5: 0.0656
PC6: 0.0494
PC7: 0.0424
PC8: 0.0268
PC9: 0.0222
PC10: 0.0193
```



```
Explained Variance Ratio for each Principal Component:
PC1: 0.3620
PC2: 0.1921
PC3: 0.1112
PC4: 0.0707
PC5: 0.0656
PC6: 0.0494
PC7: 0.0424
PC8: 0.0268
PC9: 0.0222
PC10: 0.0193
PC11: 0.0174
PC12: 0.0130
PC13: 0.0080

Cumulative Explained Variance:
[0.36198848 0.55406338 0.66529969 0.73598999 0.80162293 0.85098116
 0.89336795 0.92017544 0.94239698 0.96169717 0.97906553 0.99204785
 1.        ]
```

Question 8: Train a KNN Classifier on the PCA-transformed dataset (retain top 2 components). Compare the accuracy with the original dataset.
(***Include your Python code and output in the code box below.***)

Answer:

```python
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
X, y = load_wine(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42,
    stratify=y
)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
y_pred_no_pca = knn.predict(X_test_scaled)
acc_no_pca = accuracy_score(y_test, y_pred_no_pca)
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
knn_pca = KNeighborsClassifier(n_neighbors=5)
```



```python
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
y_pred_no_pca = knn.predict(X_test_scaled)
acc_no_pca = accuracy_score(y_test, y_pred_no_pca)
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
knn_pca = KNeighborsClassifier(n_neighbors=5)
knn_pca.fit(X_train_pca, y_train)
y_pred_pca = knn_pca.predict(X_test_pca)
acc_pca = accuracy_score(y_test, y_pred_pca)
print(f"Accuracy with original scaled data:{acc_no_pca:.3f}")
print(f"Accuracy with PCA (2 components):{acc_pca:.3f}")
```

```
Accuracy with original scaled data:0.944
Accuracy with PCA (2 components):0.944
```

Question 9: Train a KNN Classifier with different distance metrics (`euclidean`, `manhattan`) on the scaled Wine dataset and compare the results. (***Include your Python codeandoutputinthecodeboxbelow.***)

Answer:

```python
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
wine = load_wine()
X, y = wine.data, wine.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42,
    stratify=y
)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
metrics = ['euclidean', 'manhattan']
result = {}
for metric in metrics:
  knn =  KNeighborsClassifier(n_neighbors=5,
                              metric=metric)
  knn.fit(X_train_scaled, y_train)
  y_pred = knn.predict(X_test_scaled)
  acc = accuracy_score(y_test, y_pred)
  result[metric] = acc
for metric, acc in result.items():
  print(f"KNN with {metric} distance Accuracy: {acc:.3f}")
```

```
[31] )
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
     metrics = ['euclidean', 'manhattan']
     result = {}
     for metric in metrics:
       knn =  KNeighborsClassifier(n_neighbors=5,
                                   metric=metric)
       knn.fit(X_train_scaled, y_train)
       y_pred = knn.predict(X_test_scaled)
       acc = accuracy_score(y_test, y_pred)
       result[metric] = acc
       for metric, acc in result.items():
         print(f"KNN with {metric} distance Accuracy: {acc:.3f}")

KNN with euclidean distance Accuracy: 0.944
KNN with euclidean distance Accuracy: 0.944
KNN with manhattan distance Accuracy: 0.981
```

Question 10: You are working with a high-dimensional gene expression dataset to classify patients with different types of cancer.
Due to the large number of features and a small number of samples, traditional models overfit.

Explain how you would:

- Use PCA to reduce dimensionality
- Decide how many components to keep
- Use KNN for classification post-dimensionality reduction
- Evaluate the model
- Justify this pipeline to your stakeholders as a robust solution for real-world biomedical data

(***Include your Python code and output in the code box below.***

Answer:


```
[34] import numpy as np
     from sklearn.datasets import make_classification
     from sklearn.decomposition import PCA
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.metrics import accuracy_score
     X, y = make_classification(n_samples=500,
                                n_features=2000,
                                n_informative=50, n_classes=5,
                                random_state=42)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
     pca = PCA(n_components=50)
     X_train_pca = pca.fit_transform(X_train_scaled)
     X_test_pca = pca.transform(X_test_scaled)
     knn = KNeighborsClassifier(n_neighbors=5)
     knn.fit(X_train_pca, y_train)
     y_pred = knn.predict(X_test_pca)
     print("Accuracy:", accuracy_score(y_test, y_pred))

Accuracy: 0.19
```