# DA-AG-016: KNN & PCA – Complete Assignment Solution

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

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm. It does not build an explicit model during training; instead, it stores the training data and makes predictions based on similarity measures. In classification, KNN assigns a class label to a new data point based on the majority class among its K nearest neighbors. In regression, it predicts a continuous value by averaging the target values of the K nearest neighbors. Distance metrics such as Euclidean or Manhattan distance are commonly used to measure similarity.

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

The Curse of Dimensionality refers to the challenges that arise when working with high-dimensional data. As the number of features increases, the distance between data points becomes less meaningful. For KNN, this results in poor performance because all points appear similarly distant, reducing the algorithm's ability to identify true nearest neighbors. It also increases computational cost and sensitivity to noise.

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

Principal Component Analysis (PCA) is an unsupervised dimensionality reduction technique that transforms original features into a new set of orthogonal components that capture maximum variance. Unlike feature selection, which selects a subset of original features, PCA creates new features (principal components) by combining existing ones. PCA focuses on variance preservation, while feature selection focuses on relevance.

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

Eigenvectors represent the directions of maximum variance in the data, while eigenvalues indicate the amount of variance captured along each eigenvector. In PCA, eigenvectors define the principal components, and eigenvalues help determine their importance. Components with larger eigenvalues are retained to preserve most of the data's information.

## Question 5: How do KNN and PCA complement each other when applied in a single pipeline?

PCA reduces dimensionality and removes noise, addressing the Curse of Dimensionality. KNN benefits from PCA because distance calculations become more meaningful in lower dimensions. Together, PCA improves computational efficiency and generalization, while KNN provides flexible and accurate classification.

## Question 6: KNN with and without Feature Scaling

```
from sklearn.datasets import load_wine
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

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)
```

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
acc_unscaled = accuracy_score(y_test, knn.predict(X_test))

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

knn.fit(X_train_s, y_train)
acc_scaled = accuracy_score(y_test, knn.predict(X_test_s))

print("Accuracy without scaling:", acc_unscaled)
print("Accuracy with scaling:", acc_scaled)
```

## Question 7: PCA Explained Variance Ratio

```
from sklearn.decomposition import PCA
from sklearn.datasets import load_wine

X, y = load_wine(return_X_y=True)
pca = PCA()
pca.fit(X)

print("Explained Variance Ratio:", pca.explained_variance_ratio_)
```

## Question 8: KNN on PCA-Transformed Data

```
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_wine

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)

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_s)
X_test_pca = pca.transform(X_test_s)

knn = KNeighborsClassifier()
knn.fit(X_train_pca, y_train)
print("Accuracy with PCA:", accuracy_score(y_test, knn.predict(X_test_pca)))
```

## Question 9: KNN with Different Distance Metrics

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn_eu = KNeighborsClassifier(metric='euclidean')
knn_ma = KNeighborsClassifier(metric='manhattan')

knn_eu.fit(X_train_s, y_train)
knn_ma.fit(X_train_s, y_train)

print("Euclidean Accuracy:", accuracy_score(y_test, knn_eu.predict(X_test_s)))
print("Manhattan Accuracy:", accuracy_score(y_test, knn_ma.predict(X_test_s)))
```

## Question 10: PCA + KNN for Gene Expression Classification

PCA is applied to reduce thousands of gene features into a smaller set of components that capture maximum variance. The number of components is chosen using explained variance thresholds.

KNN is then trained on the reduced feature space. Model evaluation is performed using cross-validation, accuracy, precision, recall, and ROC-AUC. This pipeline reduces overfitting, improves interpretability, and provides a robust solution for high-dimensional biomedical data.