Name: Ronit Sahoo
Class: D15C
Roll No: 41
Batch: B

MLDL EXPERIMENT NO: 4

Aim:

- Implement K-Nearest Neighbors (KNN) and evaluate model performance.

Dataset Description

- The Wine Quality Dataset (WineQT) is a well-known multivariate dataset widely used in machine learning for regression and classification tasks related to food quality assessment. It contains physicochemical properties of red wine samples and is often regarded as a benchmark dataset for studying feature influence, regression modeling, and quality prediction problems. Unlike small or synthetic datasets, WineQT reflects real-world variability in chemical composition, making it suitable for evaluating models that capture non-linear relationships between input features and sensory quality ratings.
- The dataset is particularly valuable for algorithms such as Linear Regression, Decision Trees, Random Forests, K-Nearest Neighbors (KNN), and Logistic Regression, as it includes correlated chemical attributes that influence wine quality. The primary objective of the dataset is to predict the quality score of wine (typically ranging from 0 to 10) based on various physicochemical measurements. **Dataset Size:** ~1,100 rows and 13 columns
- Some of the columns in the Wine Quality Dataset are listed below:

Column Name	Data Type	Description
Fixed Acidity	Numerical (Float)	Concentration of non-volatile acids in wine
Volatile Acidity	Numerical (Float)	Amount of acetic acid, affecting taste and aroma
Citric Acid	Numerical (Float)	Citric acid content contributing to freshness
Residual Sugar	Numerical (Float)	Remaining sugar after fermentation
Chlorides	Numerical (Float)	Salt concentration in wine
Free Sulfur Dioxide	Numerical (Integer)	Amount of free SO_2, preventing microbial growth
Total Sulfur Dioxide	Numerical (Integer)	Total SO_2 concentration
Density	Numerical (Float)	Density of wine, related to alcohol and sugar content
pH	Numerical (Float)	Acidity level of the wine
Sulphates	Numerical (Float)	Additive influencing wine preservation and taste
Alcohol	Numerical (Float)	Alcohol content by volume
Quality	Numerical (Integer)	Sensory quality score assigned by experts
Id	Numerical (Integer)	Unique identifier for each wine sample

Dataset Source: https://www.kaggle.com/datasets/yasserh/wine-quality-dataset

KNN (K-NEAREST NEIGHBORS)

Theory:

K-Nearest Neighbors (KNN) is a non-parametric, instance-based supervised learning algorithm that can be applied to both classification and regression problems. It is commonly referred to as a *lazy learning* technique because it does not build an explicit model during the training phase. Instead, the algorithm stores the entire training dataset and defers computation until a prediction is required. The core idea behind KNN is the similarity assumption, which states that data points with similar feature values are likely to have similar outcomes. When a new wine sample is introduced, KNN identifies the *k* closest samples from the training set and predicts the wine quality based on the majority class (classification) or the average value (regression) of those neighbors.

The notion of "closeness" between data points is defined using distance metrics. The most commonly used metric is Euclidean Distance, which measures the straight-line distance between two points in an *n*-dimensional feature space.

For two samples

$P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_n)$, the Euclidean distance is given by:

$$d(P, Q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

In cases where features are on different scales or robustness to outliers is required, Manhattan Distance may be preferred. It computes distance as the sum of absolute differences between corresponding feature values:

$$\boxed{d(P, Q) = \sum_{i=1}^{n} |p_i - q_i|}$$

These distance measures enable KNN to compare wine samples based on physicochemical properties such as alcohol content, acidity, sulphates, and pH.

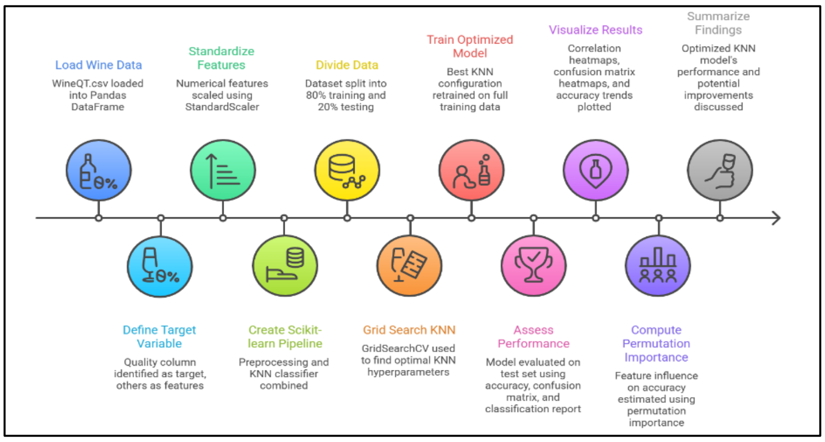
Limitations:

1. **High Computational Cost:**: Since KNN performs no learning during training, all distance calculations are carried out at prediction time. For large datasets, this leads to high computational overhead, as the algorithm must compute distances between the test sample and every training instance, making it unsuitable for real-time applications with large data volumes.
2. **Curse of Dimensionality:**: As the number of features increases, the effectiveness of distance-based comparison decreases. In high-dimensional spaces, data points become sparsely distributed, causing distances between neighbors to become nearly uniform. This reduces KNN's ability to distinguish meaningful patterns, especially when many correlated chemical attributes are present.

3. **Sensitivity to Feature Scaling and Outliers:**: KNN is highly sensitive to differences in feature scale. If one feature (e.g., sulfur dioxide) has a much larger numeric range than another (e.g., pH), it can dominate the distance calculation unless proper normalization is applied. Additionally, noisy data points or outliers in the training set can significantly affect predictions, particularly when a small value of *k* is used.

Workflow:

1. **Data Collection:** The Wine Quality dataset (WineQT.csv) is loaded into a Pandas DataFrame. It contains physicochemical attributes such as acidity levels, sulphates, alcohol content, pH, density, and sulfur dioxide concentrations used to predict wine quality.
2. **Target Identification:** The target variable is identified as Quality, representing the sensory score assigned to each wine sample. All remaining columns are treated as input features.
3. **Data Preprocessing:** Numerical features are standardized using StandardScaler to ensure equal contribution during distance computation in KNN. Since the dataset primarily contains numerical attributes, scaling is applied uniformly across features.
4. **Pipeline Construction:** A Scikit-learn Pipeline is constructed to combine preprocessing steps and the KNeighborsClassifier, ensuring consistent transformations during training and evaluation.
5. **Train–Test Split:** The dataset is divided into 80% training data and 20% testing data using a fixed random state to ensure reproducibility and fair evaluation.
6. **Hyperparameter Tuning with Grid Search:** GridSearchCV with 5-fold cross-validation is used to optimize KNN hyperparameters, including the number of neighbors (*k*), distance weighting strategy, and distance metric (Euclidean or Manhattan).
7. **Model Training:** The best-performing KNN configuration obtained from grid search is retrained on the full training dataset to build the final optimized model.
8. **Model Evaluation:** Model performance is evaluated on the test set using accuracy, a confusion matrix, and a detailed classification report including precision, recall, and F1-score.
9. **Visualization and Analysis:** Correlation heatmaps are generated to analyze relationships between numerical features and wine quality. Confusion matrix heatmaps visualize prediction outcomes, and line plots illustrate accuracy trends across different *k* values.
10. **Feature Importance Analysis:** Permutation importance is computed to estimate the influence of each feature on model accuracy, providing insight into which chemical properties most affect wine quality prediction.
11. **Conclusion:** The optimized KNN model demonstrates effective performance in predicting wine quality when proper feature scaling is applied. While KNN captures local similarity patterns well, further performance improvements can be achieved using ensemble techniques such as Random Forests or Gradient Boosting.


Load Wine Data
WineQT.csv loaded into Pandas DataFrame
Define Target Variable
Quality column identified as target, others as features
Standardize Features
Numerical features scaled using StandardScaler
Create Scikit-learn Pipeline
Preprocessing and KNN classifier combined
Divide Data
Dataset split into 80% training and 20% testing
Grid Search KNN
GridSearchCV used to find optimal KNN hyperparameters
Train Optimized Model
Best KNN configuration retrained on full training data
Assess Performance
Model evaluated on test set using accuracy, confusion matrix, and classification report
Visualize Results
Correlation heatmaps, confusion matrix heatmaps, and accuracy trends plotted
Compute Permutation Importance
Feature influence on accuracy estimated using permutation importance
Summarize Findings
Optimized KNN model's performance and potential improvements discussed

Performance Analysis:

The performance of the optimized K-Nearest Neighbors (KNN) model on the Wine Quality dataset is evaluated using Accuracy, a Confusion Matrix, and class-specific metrics such as Precision, Recall, and F1-score.

1. **Overall Accuracy:** The tuned KNN model achieves a final test accuracy of 77.29%, indicating that the model correctly classifies wine samples in approximately 77 out of every 100 cases. While lower than ensemble-based methods, this performance is reasonable for a distance-based algorithm applied to a dataset with overlapping feature distributions and multiple correlated chemical attributes.

2. **Class-Specific Performance (Precision & Recall):** To better understand model behavior across both quality classes, the classification report is analyzed:

   a. Class 0 (Lower Quality Wines):

   - Precision: **0.81**
   - Recall: **0.66**
   - F1-score: **0.73**

   The model shows strong precision for lower-quality wines, meaning most wines predicted as low quality are correctly classified. However, the lower recall indicates that some low-quality wines are misclassified as higher quality.

   b. Class 1 (Higher Quality Wines):

   - Precision: **0.75**
   - Recall: **0.87**
   - F1-score: **0.81**

   The model performs better at identifying higher-quality wines, with a high recall of 87%, indicating that most quality wines are successfully detected. This suggests the model favors sensitivity toward the positive class.

3. **Confusion Matrix Breakdown:** The confusion matrix provides insight into correct and incorrect predictions:

   - **True Negatives:** Lower-quality wines correctly classified as low quality.
   - **True Positives:** Higher-quality wines correctly identified.
   - **False Positives:** Lower-quality wines incorrectly predicted as high quality.
   - **False Negatives:** Higher-quality wines misclassified as low quality.

The relatively higher recall for quality wines indicates fewer false negatives, which is desirable when identifying better-quality products.

Hyperparameter Tuning:

K-Nearest Neighbors is a distance-based model whose performance heavily depends on hyperparameter selection. **GridSearchCV** is used to exhaustively evaluate different parameter combinations using **5-fold cross-validation**, ensuring reliable and unbiased performance estimates.

The following hyperparameters are tuned:

- **Number of Neighbors (n_neighbors):** Controls neighborhood size. A larger value smooths decision boundaries and reduces noise sensitivity. The optimal value obtained is **k = 15**.
- **Weighting Method (weights):** Distance-based weighting assigns higher influence to closer neighbors, improving prediction reliability.
- **Distance Metric (metric):** Manhattan distance is selected over Euclidean distance, offering improved stability in standardized, higher-dimensional feature space.

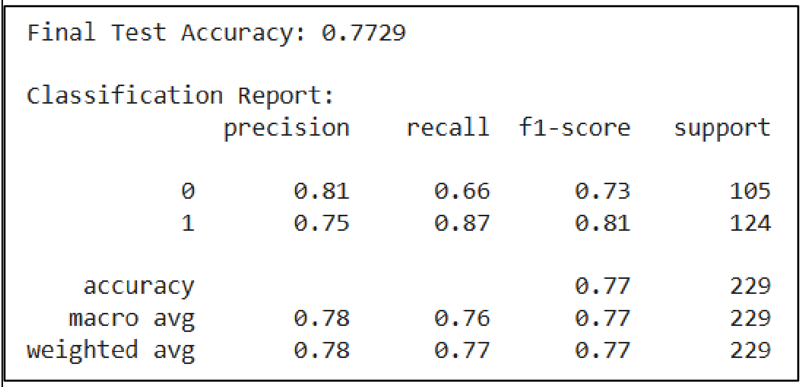
The optimal configuration (k = 15, distance-based weighting, Manhattan distance) achieves the best cross-validation accuracy of 76.04% and results in a final test accuracy of 77.29%, demonstrating effective generalization.

Code:

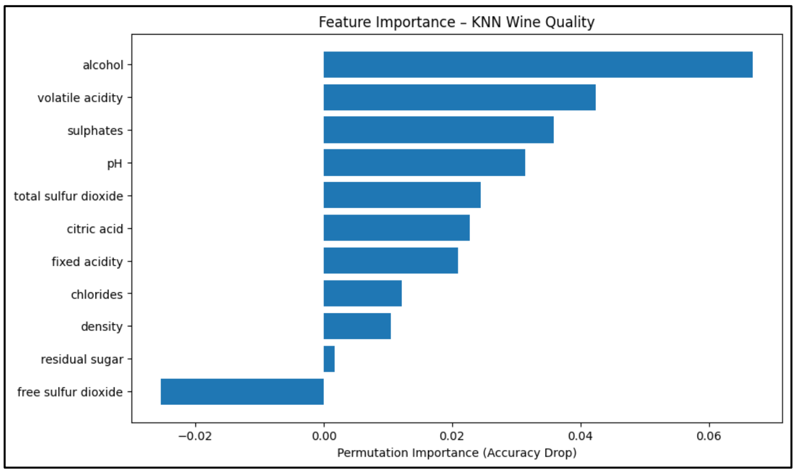
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import
train_test_split, GridSearchCV
from sklearn.preprocessing import
StandardScaler
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix
from sklearn.inspection import
permutation_importance

# ========================
# 1. Load Dataset
# ========================
df = pd.read_csv("dataset.csv")   # change
name if needed
df.columns = df.columns.str.strip()

print("Dataset Shape:", df.shape)
print(df.head())

# ========================
# 2. Drop ID column (NOT a feature)
# ========================
if 'Id' in df.columns:
    df = df.drop('Id', axis=1)

# ========================
# 3. Convert to Binary Classification
#    Good wine: quality >= 6
# ========================
df['quality_label'] = (df['quality'] >=
6).astype(int)

X = df.drop(['quality', 'quality_label'], axis=1)
y = df['quality_label']

# ========================
# 4. Train-Test Split
# ========================
X_train, X_test, y_train, y_test =
train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

# ========================
# 5. Feature Scaling (MANDATORY for KNN)
# ========================
scaler = StandardScaler()
```

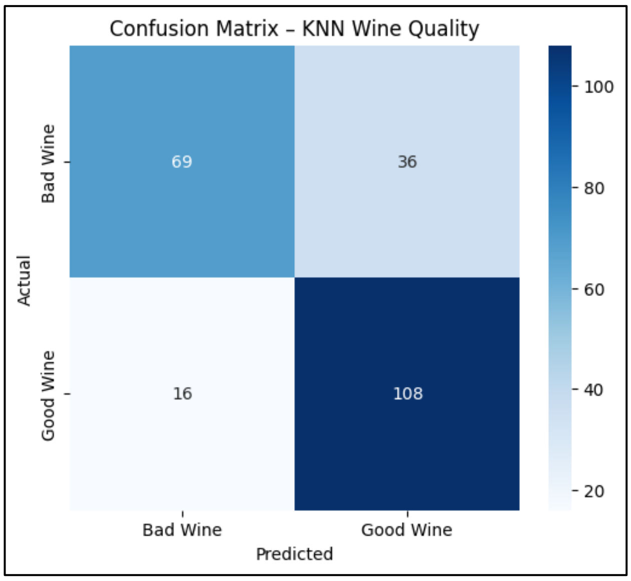
```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# =========================
# 6. KNN + Hyperparameter Tuning
# =========================
param_grid = {
   'n_neighbors': [3, 5, 7, 9, 11, 15],
   'weights': ['uniform', 'distance'],
   'metric': ['euclidean', 'manhattan']
}

knn = KNeighborsClassifier()

grid_search = GridSearchCV(
   knn,
   param_grid,
   cv=5,
   scoring='accuracy',
   n_jobs=-1
)

grid_search.fit(X_train_scaled, y_train)

best_model = grid_search.best_estimator_

print("\nBest Parameters:",
grid_search.best_params_)
print("Best Cross-Validation Accuracy:",
round(grid_search.best_score_, 4))

# =========================
# 7. Evaluation
# =========================
y_pred = best_model.predict(X_test_scaled)

print("\nFinal Test Accuracy:",
round(accuracy_score(y_test, y_pred), 4))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# =========================
# 8. Confusion Matrix
# =========================
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,5))
sns.heatmap(
   cm,
   annot=True,
   fmt='d',
   cmap='Blues',
   xticklabels=['Bad Wine', 'Good Wine'],
   yticklabels=['Bad Wine', 'Good Wine']
)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix – KNN Wine
Quality")
plt.show()

# =========================
# 9. Correlation Heatmap
# =========================
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), cmap='RdBu_r',
center=0)
plt.title("Correlation Heatmap – Wine Quality
Dataset")
plt.show()

# =========================
# 10. Accuracy vs Number of Neighbors (K)
# =========================
results_df =
pd.DataFrame(grid_search.cv_results_)
subset = results_df[results_df['param_metric']
== 'euclidean']

plt.figure(figsize=(10,6))
sns.lineplot(
   data=subset,
   x='param_n_neighbors',
   y='mean_test_score',
   hue='param_weights',
   marker='o'
)
plt.xlabel("Number of Neighbors (K)")
plt.ylabel("Mean Cross-Validation Accuracy")
plt.title("Hyperparameter Tuning: Accuracy vs
K")
plt.show()

# =========================
# 11. Permutation Feature Importance
# =========================
perm_importance =
permutation_importance(
   best_model,
   X_test_scaled,
   y_test,
    n_repeats=10,
    random_state=42
)

sorted_idx =
perm_importance.importances_mean.argsort
()

plt.figure(figsize=(10,6))
plt.barh(
    X.columns[sorted_idx],
    perm_importance.importances_mean[sorte
d_idx]
)
plt.xlabel("Permutation Importance (Accuracy
Drop)")
plt.title("Feature Importance – KNN Wine
Quality")
plt.show()
```

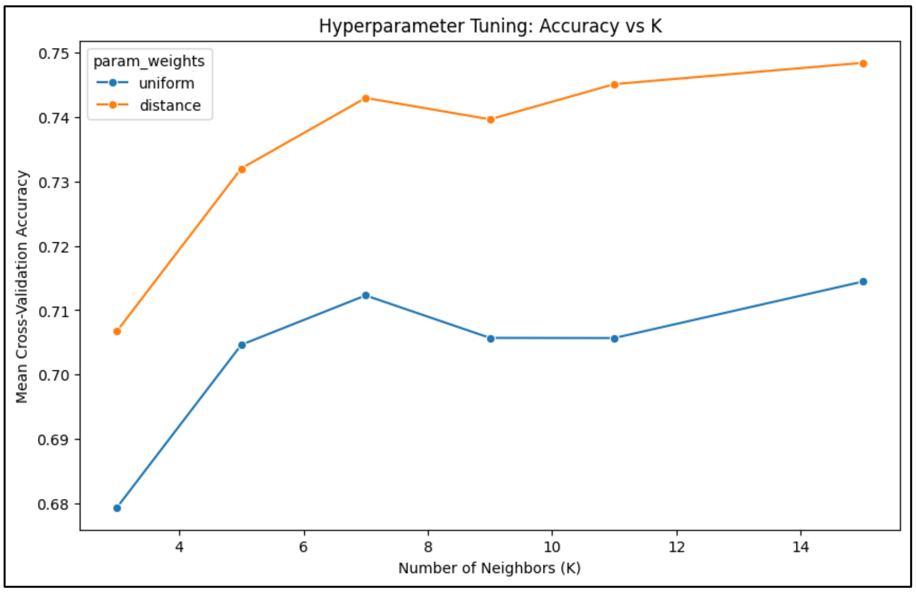
Output:

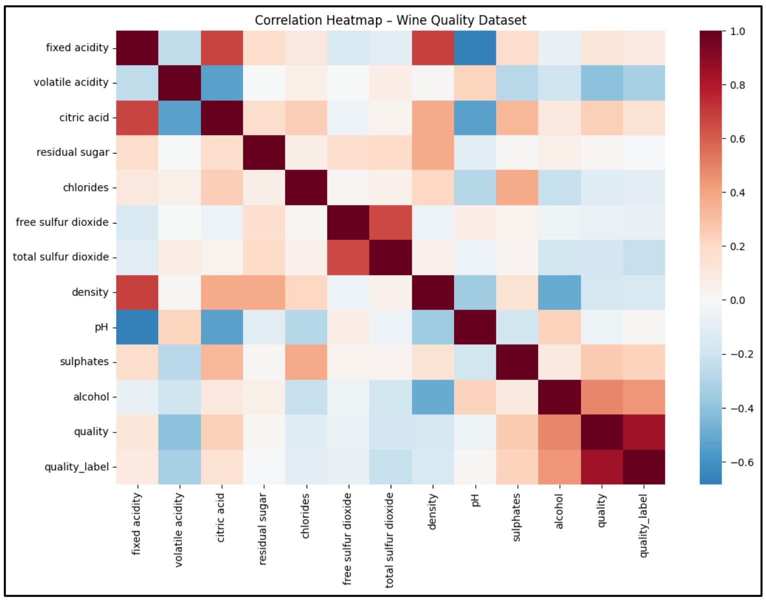


```
Final Test Accuracy: 0.7729

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.66      0.73       105
           1       0.75      0.87      0.81       124

    accuracy                           0.77       229
   macro avg       0.78      0.76      0.77       229
weighted avg       0.78      0.77      0.77       229
```









Conclusion:

- In this experiment, the K-Nearest Neighbors (KNN) algorithm was studied theoretically and implemented practically on the Wine Quality dataset.
- Proper data preprocessing and feature scaling were essential to ensure accurate distance-based comparisons between wine samples.
- Hyperparameter tuning using GridSearchCV identified the optimal configuration with k = 15, Manhattan distance, and distance-based weighting.
- The optimized KNN model achieved a test accuracy of 77.29%, demonstrating effective classification of wine quality levels.
- While KNN successfully captured local similarity patterns, its performance and scalability are limited, and ensemble models such as Random Forest can provide better robustness and accuracy.