Name : Manaswi Kabadi
Class : D15C
Roll no. : 27
Batch : B

EXP - 4

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

Dataset Source : https://www.kaggle.com/datasets/henriqueyamahata/bank-marketing

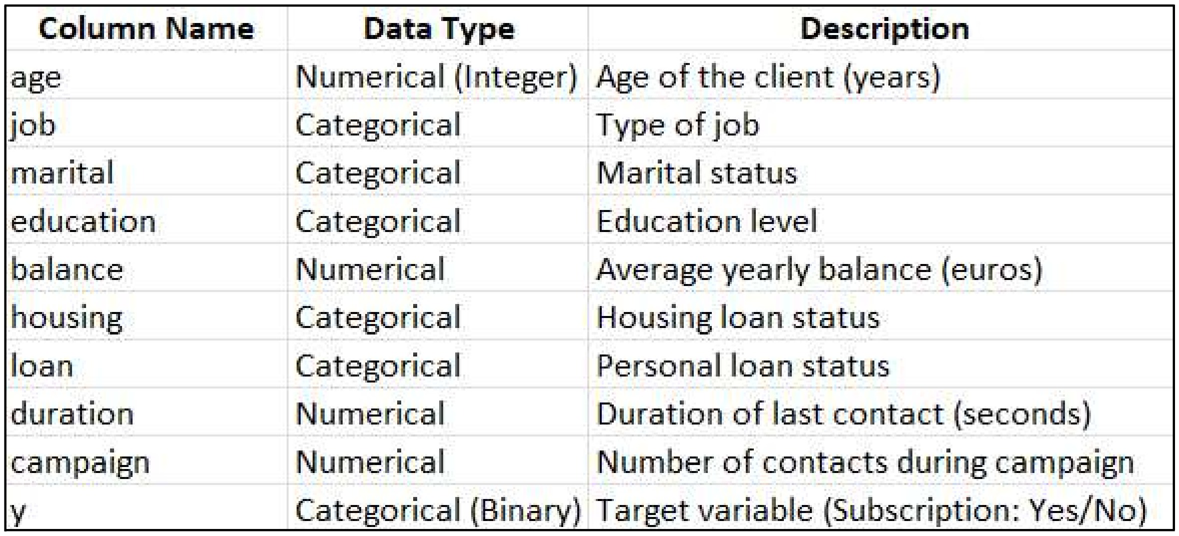
Dataset Description :

The Bank Marketing Dataset is a real-world dataset collected from a Portuguese banking institution and is commonly used to analyze customer behavior in marketing campaigns. The objective of the dataset is to predict whether a client will subscribe to a term deposit based on personal, economic, and campaign-related attributes.

The dataset contains approximately 45,000 records, making it a large dataset suitable for evaluating distance-based algorithms such as K-Nearest Neighbors (KNN). Each record represents a client contacted during a direct marketing campaign.

The dataset consists of a mixture of numerical and categorical attributes, which require preprocessing before applying the KNN algorithm.

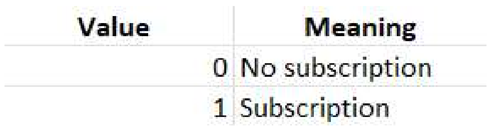
Dataset Attributes:



Column Name	Data Type	Description
age	Numerical (Integer)	Age of the client (years)
job	Categorical	Type of job
marital	Categorical	Marital status
education	Categorical	Education level
balance	Numerical	Average yearly balance (euros)
housing	Categorical	Housing loan status
loan	Categorical	Personal loan status
duration	Numerical	Duration of last contact (seconds)
campaign	Numerical	Number of contacts during campaign
y	Categorical (Binary)	Target variable (Subscription: Yes/No)

Target Variables Used in the Experiment

y – Indicates whether the client subscribed to a term deposit .This variable is used as the target for KNN classification.



Value	Meaning
0	No subscription
1	Subscription

Theory

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It is a non-parametric and instance-based learning algorithm, meaning it does not assume any underlying data distribution and does not build an explicit model during training.KNN is often referred to as a lazy learning algorithm because the training phase simply stores the dataset, and all computation is performed during the prediction phase. When a new data point is given, the algorithm identifies the K closest data points (neighbors) from the training dataset and assigns the class based on majority voting among those neighbors.
The fundamental assumption of KNN is that data points that are close to each other in feature space are likely to belong to the same class. Therefore, distance measurement plays a crucial role in the performance of the algorithm.
In this experiment, KNN is applied to the Bank Marketing dataset to classify whether a customer will subscribe to a term deposit (`y = 1`) or not (`y = 0`).

Mathematical Formulation of the Algorithm :

KNN classifies a test data point by computing the distance between the test point and all training data points using a distance metric.

Euclidean Distance (most commonly used):

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Where:

- x = test data point
- y = training data point
- n = number of features

Other distance metrics include:

Manhattan Distance

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Classification Rule

1. Compute distances between the test point and all training points.
2. Select the **K nearest neighbors**.
3. Assign the class that occurs most frequently among the K neighbors:

$$\hat{y} = \mathrm{mode}(y_1, y_2, \ldots, y_k)$$

Where y1,y2,…,yk are the class labels of the K nearest neighbors.
Since distance calculations are sensitive to feature scale, feature scaling using StandardScaler is mandatory for KNN.

Algorithm Limitations :

- **High Computational Cost**
  KNN requires calculating the distance between a test instance and all training instances, making prediction slow for large datasets.Condition of failure: Performance degrades when the dataset size is very large.
- **Sensitivity to Feature Scaling**
  Features with larger numeric ranges dominate distance calculations. Condition of failure: Without proper scaling, the model produces biased and incorrect predictions.
- **Curse of Dimensionality**
  As the number of features increases, distances between points become less meaningful.
  Condition of failure: Model accuracy decreases in high-dimensional datasets.
- **Sensitivity to Noise and Outliers**
  Noisy data or outliers can significantly affect neighbor selection. Condition of failure: Small values of K make the model unstable.
- **Memory Intensive**
  KNN stores the entire training dataset in memory.
  Condition of failure: High memory usage for large datasets.

Methodology / Workflow :

Data Collection

The Bank Marketing dataset was loaded from a CSV file. It contains customer demographic, financial, and campaign-related information along with the target variable indicating term deposit subscription.

Target Identification

The target variable `y` was identified, where `1` represents subscription and `0` represents no subscription. All remaining attributes were treated as input features.

Data Preprocessing

Categorical attributes were converted into numerical form using label encoding. Since KNN is a distance-based algorithm, feature scaling was performed using StandardScaler to ensure all features contribute equally to distance computation.

Train–Test Split

The dataset was divided into 80% training data and 20% testing data using a stratified split to preserve class distribution.

Model Training

Multiple KNN models were trained using different values of K to analyze the effect of the number of neighbors on model performance.

Hyperparameter Tuning

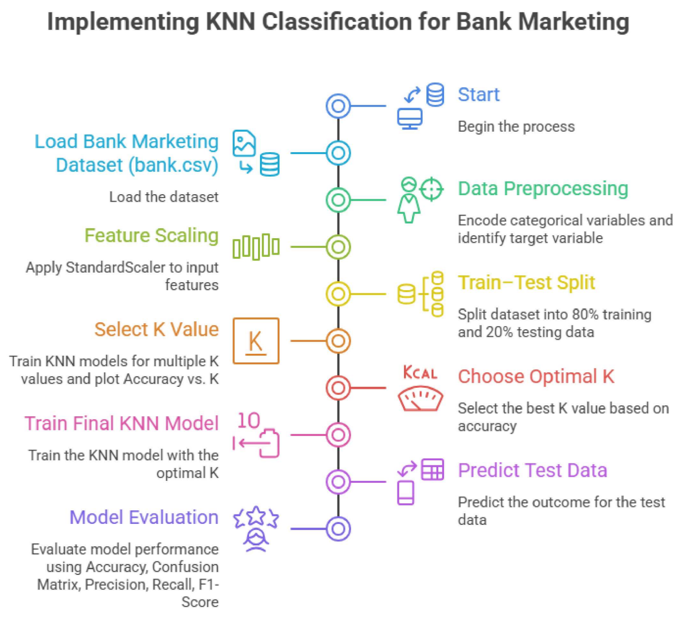
The optimal K value was selected based on the Accuracy vs. K analysis, choosing the value that produced the highest and most stable accuracy.

Prediction

The final KNN model with the selected K value was used to predict subscription outcomes on the test dataset.

Model Evaluation

Model performance was evaluated using accuracy, confusion matrix, precision, recall, and F1-score. Visualizations such as the confusion matrix and accuracy vs. K plot were used to analyze performance.


Implementing KNN Classification for Bank Marketing
Start
Begin the process
Load Bank Marketing Dataset (bank.csv)
Load the dataset
Data Preprocessing
Encode categorical variables and identify target variable
Feature Scaling
Apply StandardScaler to input features
Train–Test Split
Split dataset into 80% training and 20% testing data
Select K Value
K
Train KNN models for multiple K values and plot Accuracy vs. K
Choose Optimal K
KCAL
Select the best K value based on accuracy
Train Final KNN Model
10
Train the KNN model with the optimal K
Predict Test Data
Predict the outcome for the test data
Model Evaluation
Evaluate model performance using Accuracy, Confusion Matrix, Precision, Recall, F1-Score

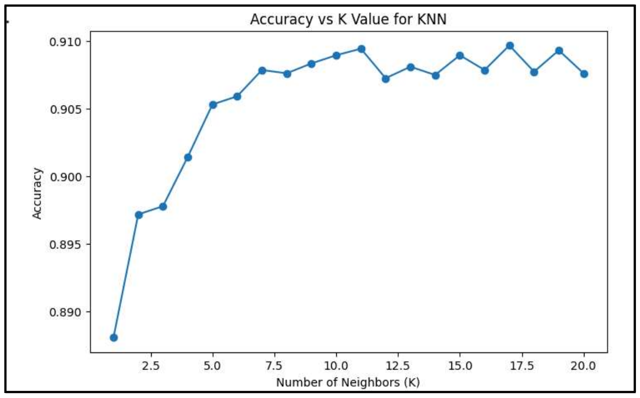
Code :

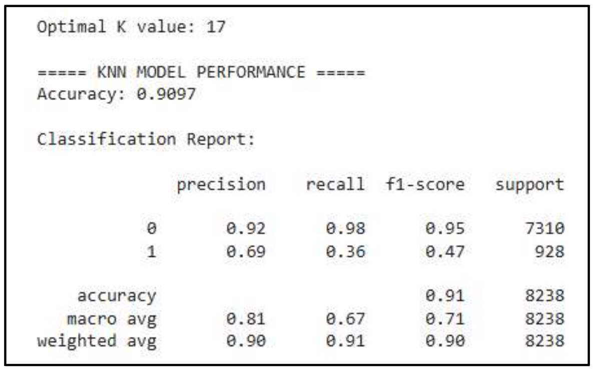
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection
import train_test_split
from sklearn.preprocessing import
LabelEncoder, StandardScaler
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import
accuracy_score,
classification_report,
confusion_matrix,
ConfusionMatrixDisplay
#Load Dataset
df = pd.read_csv("bank.csv",
sep=';')
# Data Preprocessing
le = LabelEncoder()
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] =
le.fit_transform(df[col])
# Features and Target
X = df.drop("y", axis=1)
y = df["y"]
# 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)
# Feature Scaling
scaler = StandardScaler()
X_train_scaled =
scaler.fit_transform(X_train)
X_test_scaled =
scaler.transform(X_test)
# Selecting Optimal K
accuracy_scores = []
k_range = range(1, 21)
for k in k_range:
    knn =
KNeighborsClassifier(n_neighbors=k
)
    knn.fit(X_train_scaled,
y_train)
    y_pred_k =
knn.predict(X_test_scaled)
accuracy_scores.append(accuracy_sc
ore(y_test, y_pred_k))
# Accuracy vs K Plot
plt.figure(figsize=(8, 5))
plt.plot(k_range, accuracy_scores,
marker='o')
plt.xlabel("Number of Neighbors
(K)")
plt.ylabel("Accuracy")
plt.title("Accuracy vs K Value for
KNN")
plt.show()
# Best K
best_k =
k_range[np.argmax(accuracy_scores)
]
print(f"Optimal K value:
{best_k}")
# Final KNN Model
knn_final =
KNeighborsClassifier(n_neighbors=b
est_k)
knn_final.fit(X_train_scaled,
y_train)
# Prediction
y_pred =
knn_final.predict(X_test_scaled)
# Model Evaluation
```

```
print("\n===== KNN MODEL
PERFORMANCE =====")
print(f"Accuracy:
{accuracy_score(y_test,
y_pred):.4f}")
print("\nClassification
Report:\n")
print(classification_report(y_test
, y_pred))
#Confusion Matrix
cm = confusion_matrix(y_test,
y_pred)
```

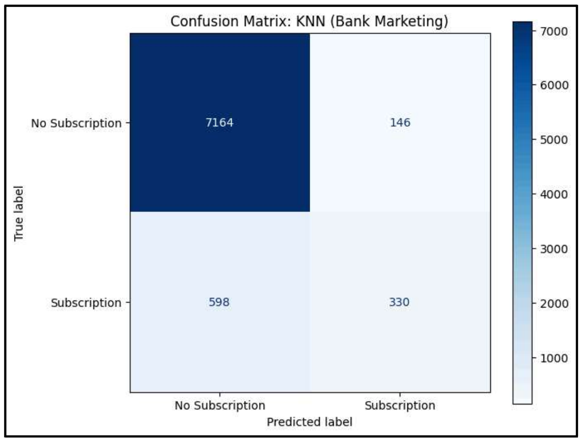
```
disp = ConfusionMatrixDisplay(
    confusion_matrix=cm,
    display_labels=["No
Subscription", "Subscription"])

fig, ax = plt.subplots(figsize=(7,
6))
disp.plot(cmap="Blues", ax=ax)
plt.title("Confusion Matrix: KNN
(Bank Marketing)")
plt.show()
```

OUTPUT :





```
Optimal K value: 17

===== KNN MODEL PERFORMANCE =====
Accuracy: 0.9097

Classification Report:

              precision    recall  f1-score   support

           0       0.92      0.98      0.95      7310
           1       0.69      0.36      0.47       928

    accuracy                           0.91      8238
   macro avg       0.81      0.67      0.71      8238
weighted avg       0.90      0.91      0.90      8238
```



Performance Analysis :

The performance of the K-Nearest Neighbors (KNN) classifier was evaluated using accuracy, confusion matrix, classification report, and the accuracy vs. K graph on the Bank Marketing dataset.

Accuracy vs. K Analysis
The Accuracy vs. K graph shows that model accuracy increases rapidly for small values of K and stabilizes for moderate values. The highest accuracy is achieved around K ≈ 11–17, indicating an optimal balance between bias and variance. Very small K values lead to noisy predictions, while larger K values slightly reduce sensitivity. The selected K produced a final accuracy of 90.97%.
Overall Accuracy
The KNN model achieved an accuracy of 90.97%, indicating strong overall classification performance. This shows that the model is effective in predicting customer subscription behavior for the majority of cases.

Confusion Matrix Analysis

The confusion matrix shows that:

- 7,164 non-subscription cases were correctly classified
- 330 subscription cases were correctly classified
- 598 subscription cases were misclassified as non-subscription

The model performs very well for the non-subscription (majority) class, while subscription cases are harder to predict due to class imbalance.

Classification Report Interpretation

- Class 0 (No Subscription):
  Precision = 0.92, Recall = 0.98, F1-score = 0.95
  → The model is highly reliable in identifying customers who do not subscribe.
- Class 1 (Subscription):
  Precision = 0.69, Recall = 0.36, F1-score = 0.47
  → The lower recall indicates that many actual subscribers are missed.

The macro-average F1-score of 0.71 highlights the impact of class imbalance, while the weighted F1-score of 0.90 confirms strong overall performance.

Hyperparameter Tuning :

The performance of KNN mainly depends on the number of neighbors (K). Different K values from 1 to 20 were tested, and accuracy increased initially before stabilizing. The optimal K was selected where maximum stable accuracy was achieved, resulting in a final accuracy of 90.97%. This tuning helped balance bias and variance in the model.

Conclusion :

In this experiment, the K-Nearest Neighbors (KNN) algorithm was implemented on the Bank Marketing dataset to predict customer subscription behavior. The model achieved an accuracy of 90.97%, demonstrating effective overall classification performance. Hyperparameter tuning helped identify an optimal K value, improving model stability. However, the results show that KNN performs better for the majority class, highlighting the impact of class imbalance. Overall, the experiment illustrates the importance of distance-based learning and proper parameter selection in classification tasks.