Experiment No: 6

Problem Statement: Classification Modelling: a. Choose classifier for classification problem. b. Evaluate the performance of classifier.

Perform classification using the below 4 classifiers on the same dataset used in Experiment No. 5:

- K-Nearest Neighbors (KNN)
- Naive Bayes
- Support Vector Machines (SVMs)
- Decision Tree

Theory: Classification is a **supervised learning** technique where the goal is to assign a label to input data based on past training examples. In this experiment, we explore four fundamental classification algorithms: **K-Nearest Neighbors, Naive Bayes, Support Vector Machines, and Decision Trees**. Each algorithm has its strengths depending on the dataset and the nature of the classification problem.

---

1. K-Nearest Neighbors (KNN):

- **Instance-based learning** algorithm.
- Classifies a new data point based on the majority class among its **K closest neighbors** in the training dataset.

- Distance metrics (like Euclidean distance) are used to find the nearest neighbors.
- It is **non-parametric**, simple to implement, and effective for small datasets.
- Sensitive to the choice of K and the scale of data.

Steps in KNN:

1. Choose the number of neighbors (K).
2. Calculate distance between test data and all training data.
3. Identify the K-nearest neighbors.
4. Assign the most common class among neighbors to the test data.

---

2. Naive Bayes Classifier:

- A **probabilistic classifier** based on **Bayes' Theorem** with the “naive” assumption of **feature independence**.
- Best suited for **text classification** problems like spam detection.
- Fast, simple, and effective for large datasets.

Bayes’ Theorem: $P(C|X)=P(X|C)\cdot P(C)P(X)P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$ Where:

- $P(C|X)P(C|X)$: Posterior probability of class C given input X.

- P(X∣C)P(X|C): Likelihood of input X given class C.
- P(C)P(C): Prior probability of class C.
- P(X)P(X): Evidence or total probability of input X.

---

3. Support Vector Machines (SVM):

- A **margin-based classifier** that finds the optimal **hyperplane** that separates classes with maximum margin.
- Effective in **high-dimensional spaces** and can be used with **non-linear kernels** (e.g., RBF).
- Robust to overfitting especially in high-dimensional space.

Key Concepts:

- **Hyperplane**: Decision boundary.
- **Margin**: Distance between the hyperplane and nearest data points from each class.
- **Support Vectors**: Data points that lie closest to the decision boundary.

---

4. Decision Tree Classifier:

- A **tree-structured model** where internal nodes represent decision rules, branches represent outcomes, and leaf nodes represent class

labels.

- Splits the dataset based on feature values to maximize the information gain or Gini index.
- Easy to understand and visualize.
- Prone to overfitting on noisy data (can be controlled using pruning).

Key Terms:

- **Entropy**: Measure of impurity in the dataset.
- **Information Gain**: Reduction in entropy after splitting a dataset.

Implementation Summary:

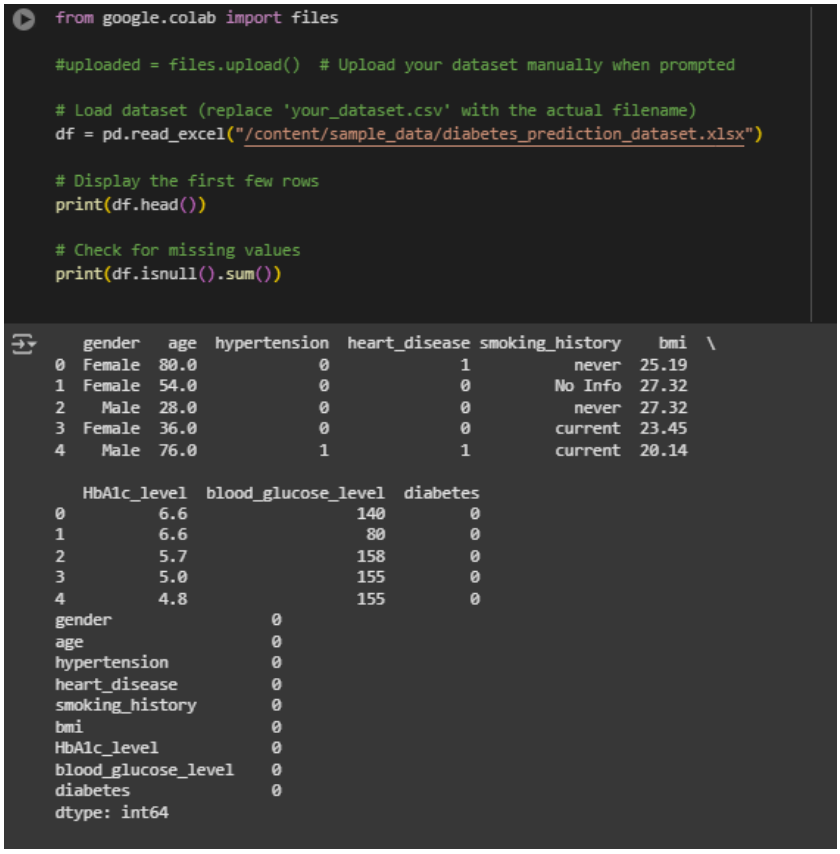
1. Load and preprocess the dataset (same as Experiment 5).
2. Apply each classifier (KNN, Naive Bayes, SVM, Decision Tree).
3. Split data into training and testing sets.
4. Train each model and make predictions.
5. Evaluate using metrics:
    - **Accuracy**
    - **Precision**

- **Recall**
- **F1-score**
- **Confusion Matrix**

6. Compare performance across models.

Space for Screenshots:

- Dataset preview and preprocessing steps



```
from google.colab import files

#uploaded = files.upload()  # Upload your dataset manually when prompted

# Load dataset (replace 'your_dataset.csv' with the actual filename)
df = pd.read_excel("/content/sample_data/diabetes_prediction_dataset.xlsx")

# Display the first few rows
print(df.head())

# Check for missing values
print(df.isnull().sum())
```

```
   gender   age  hypertension  heart_disease smoking_history    bmi  \
0  Female  80.0             0              1           never  25.19
1  Female  54.0             0              0         No Info  27.32
2    Male  28.0             0              0           never  27.32
3  Female  36.0             0              0         current  23.45
4    Male  76.0             1              1         current  20.14

   HbA1c_level  blood_glucose_level  diabetes
0          6.6                  140         0
1          6.6                   80         0
2          5.7                  158         0
3          5.0                  155         0
4          4.8                  155         0
gender                 0
age                    0
hypertension           0
heart_disease          0
smoking_history        0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
```

- Model training and prediction outputs

```
knn = KNeighborsClassifier(n_neighbors=5)  # Using k=5
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
```

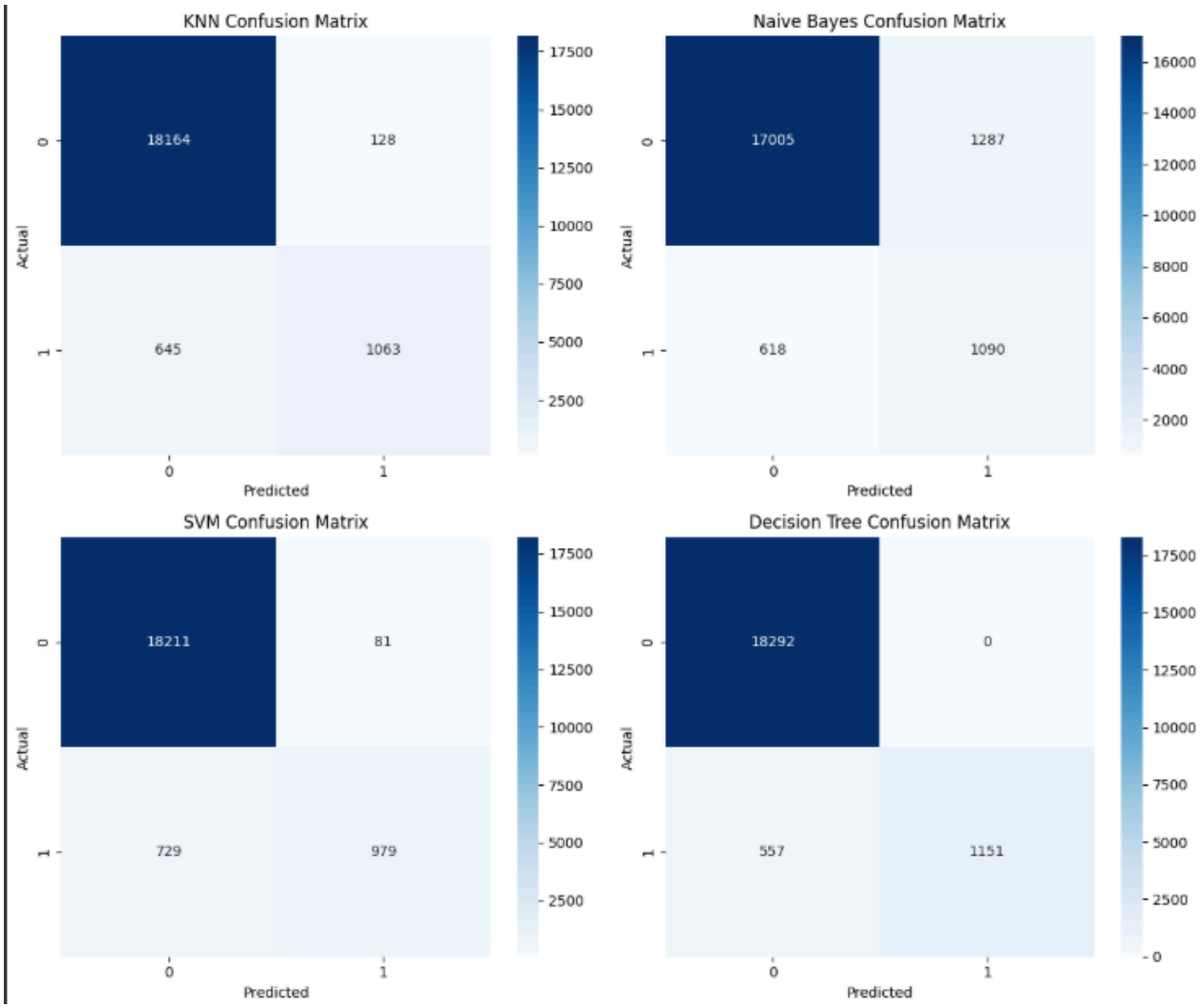
```
KNN Accuracy: 0.96135
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18292
           1       0.89      0.62      0.73      1708

    accuracy                           0.96     20000
   macro avg       0.93      0.81      0.86     20000
weighted avg       0.96      0.96      0.96     20000
```

```
Naive Bayes Accuracy: 0.90475
              precision    recall  f1-score   support

           0       0.96      0.93      0.95     18292
           1       0.46      0.64      0.53      1708

    accuracy                           0.90     20000
   macro avg       0.71      0.78      0.74     20000
weighted avg       0.92      0.90      0.91     20000
```

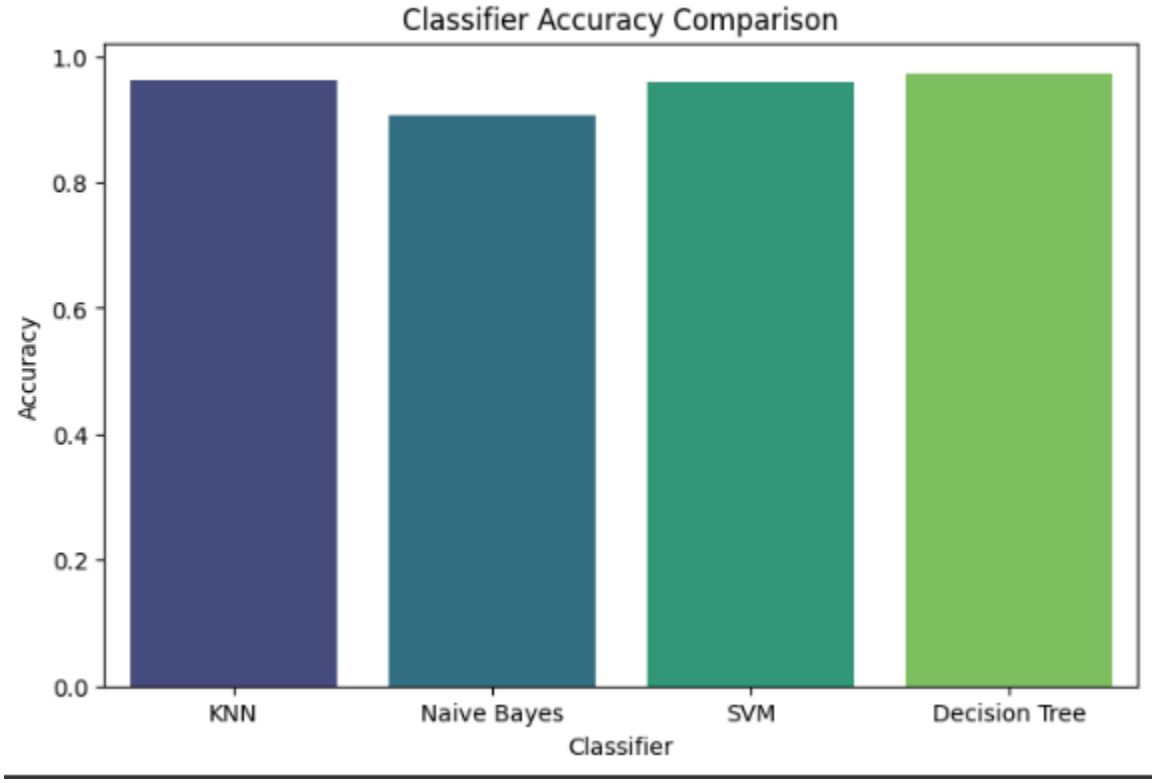
```
SVM Accuracy: 0.9595
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     18292
           1       0.92      0.57      0.71      1708

    accuracy                           0.96     20000
   macro avg       0.94      0.78      0.84     20000
weighted avg       0.96      0.96      0.96     20000
```

```
Decision Tree Accuracy: 0.97215
              precision    recall  f1-score   support

           0       0.97      1.00      0.99     18292
           1       1.00      0.67      0.81      1708

    accuracy                           0.97     20000
   macro avg       0.99      0.84      0.90     20000
weighted avg       0.97      0.97      0.97     20000
```

- Evaluation metrics and confusion matrix for each classifier



- Final performance comparison table



Conclusion: In this experiment, we successfully implemented and evaluated four different classifiers on the same dataset. Each model performed differently depending on the data characteristics. KNN worked well for clear boundaries, Naive Bayes was fast and effective for probabilistic outputs, SVM provided robust classification with margin maximization, and Decision Trees offered interpretable rules. Comparing the performance metrics helped us understand the trade-offs between these classifiers, and choose the best one for a given task.