Experiment-6

Aim: Classification modelling
a. Choose a classifier for classification problems.
b. Evaluate the performance of the
classifier. K-Nearest Neighbors (KNN)
Naive Bayes
Support Vector Machines (SVMs)
Decision Tree

Theory:
Decision Tree: The Decision Tree classifier builds a model by recursively splitting the data based on feature values, creating a tree where each node represents a decision rule and each leaf a class label. This approach is highly interpretable, as the decision rules can be easily visualized and understood

K-Nearest Neighbors (KNN): KNN classifies a new instance by finding the k closest training examples based on a distance metric (typically Euclidean distance) and assigning the majority class among these neighbors. It is a non-parametric and intuitive method that performs well when features are properly scaled.

Naive Bayes: Naive Bayes uses Bayes' theorem with the strong assumption that all features are conditionally independent given the class label. This probabilistic classifier is computationally efficient and performs robustly in high-dimensional settings, despite its simplicity.

Support Vector Machines (SVM): SVM finds the optimal hyperplane that separates classes by maximizing the margin between them, and it can handle nonlinear boundaries through the use of kernel functions. It is especially effective in high-dimensional spaces and tends to offer robust performance with appropriate parameter tuning.
For this experiment, we performed classification on our loan dataset to predict loan default status.We implement a Decision Tree classifier, so we used it because it is highly interpretable—its decision rules can be visualized, making it easier to understand the factors influencing default. Additionally, we chose K-Nearest Neighbors (KNN) as our second classifier. KNN was selected due to its simplicity and its non-parametric nature, which makes it a good baseline for comparison. Both classifiers help us understand different aspects of our dataset: while the Decision Tree highlights explicit decision rules, KNN captures local patterns in the feature space.

Experiment-6

Data Description:

The **Global Health Statistics** dataset contains **1,000,000 records** with **22 attributes** related to disease prevalence, healthcare access, and socio-economic factors across different countries and years. It includes details on diseases, their categories, prevalence, incidence, mortality rates, and affected demographics (age, gender, and population). Healthcare factors such as access percentage, doctors and hospital beds per 1,000 people, treatment types, and recovery rates are also recorded. Economic indicators like per capita income, education index, and urbanization rate provide additional context. This dataset is useful for epidemiological studies, healthcare planning, and analyzing global health trends.

Implementation :

1. Data Preparation:

The dataset requires handling missing values, encoding categorical variables (e.g., Gender, Disease Category), and normalizing numerical features for better model performance.

Models like Decision Trees and Naïve Bayes classify diseases based on attributes such as prevalence rate, mortality rate, and healthcare access, evaluated using accuracy, precision, and recall.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import graphviz


df=pd.read_csv('/content/Global Health Statistics.csv')
df.head()
```

	Country	Year	Disease Name	Disease Category	Prevalence Rate (%)	Incidence Rate (%)	Mortality Rate (%)	Age Group	Gender	Population Affected	...	Hospital Beds per 1000	Treatment Type	Average Treatment Cost (USD)	Availability of Vaccines/Treatment	Recovery Rate (%)	DALYs	Improvement in 5 Years (%)	Per Capita Income (USD)	Education Index	Urbanization Rate (%)
0	Italy	2013	Malaria	Respiratory	0.95	1.55	8.42	0-18	Male	471007	...	7.58	Medication	21064	No	91.82	4493	2.16	16886	0.79	86.02
1	France	2002	Ebola	Parasitic	12.46	8.63	8.75	61+	Male	634318	...	5.11	Surgery	47851	Yes	76.65	2366	4.82	80639	0.74	45.52
2	Turkey	2015	COVID-19	Genetic	0.91	2.35	6.22	36-60	Male	154878	...	3.49	Vaccination	27834	Yes	98.55	41	5.81	12245	0.41	40.20
3	Indonesia	2011	Parkinson's Disease	Autoimmune	4.68	6.29	3.99	0-18	Other	446224	...	8.44	Surgery	144	Yes	67.35	3201	2.22	49336	0.49	58.47
4	Italy	2013	Tuberculosis	Genetic	0.83	13.59	7.01	61+	Male	472908	...	5.90	Medication	8908	Yes	50.06	2832	6.93	47701	0.50	48.14

5 rows × 22 columns

Experiment-6

2. Data Splitting:

The dataset is split into training (80%) and testing (20%) sets using `train_test_split()`, ensuring the model is trained on one portion and evaluated on another for unbiased performance assessment.

`stratify=y` maintains the class distribution in both sets, preventing data imbalance, while `random_state=42` ensures reproducibility of the split.

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=39)
```

3. Classifier Setup and Training:

The Decision Tree model's accuracy, precision, recall, and F1-score are computed to assess classification performance, using a weighted average to handle class imbalances.

A heatmap is plotted to analyze true positives, false positives, true negatives, and false negatives, aiding in error pattern interpretation.

```
# Predictions on the test set
y_pred_dt = dt_classifier.predict(X_test)

# Compute performance metrics
accuracy_dt = accuracy_score(y_test, y_pred_dt)
precision_dt = precision_score(y_test, y_pred_dt, average='weighted')
recall_dt = recall_score(y_test, y_pred_dt, average='weighted')
f1_dt = f1_score(y_test, y_pred_dt, average='weighted')

print("Decision Tree Performance:")
print("Accuracy:", accuracy_dt)
print("Precision:", precision_dt)
print("Recall:", recall_dt)
print("F1 Score:", f1_dt)

# Plot confusion matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)
plt.figure(figsize=(8,6))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Blues',
            xticklabels=target_le.classes_, yticklabels=target_le.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Decision Tree Confusion Matrix')
plt.show()
```

```
Decision Tree Performance:
Accuracy: 0.50001
Precision: 0.5003881497242001
Recall: 0.50001
F1 Score: 0.47869836290236084
```

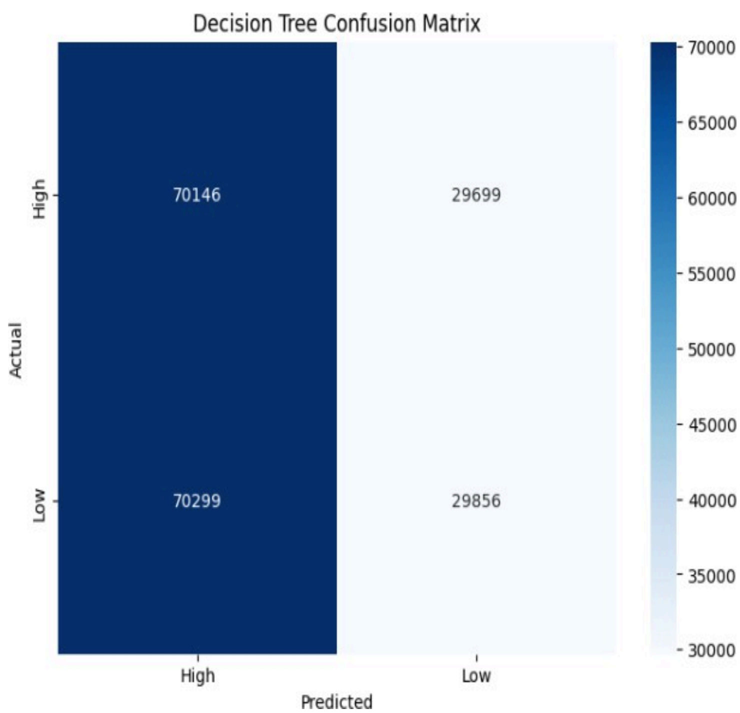
Experiment-6

4. Model Evaluation:

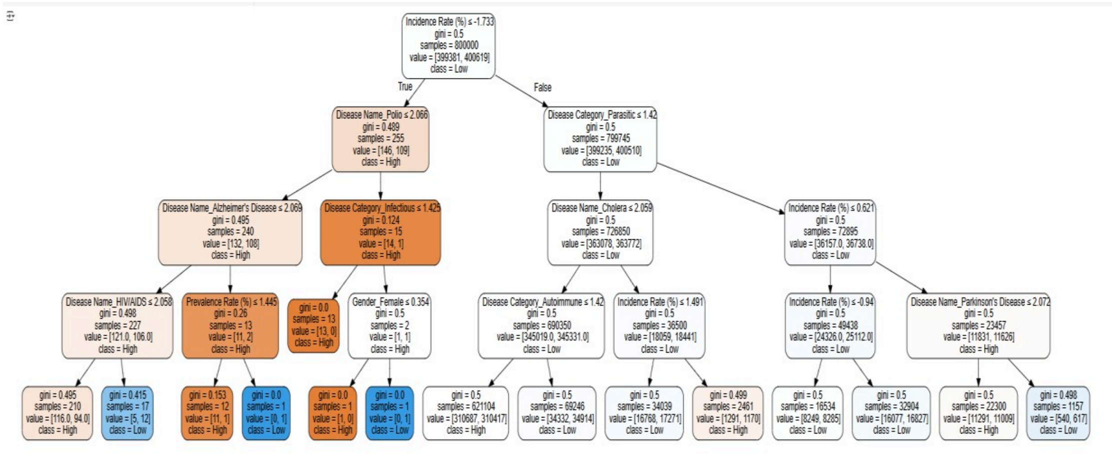
The heatmap visualizes classification performance, where diagonal values represent correct predictions, and off-diagonal values indicate misclassifications.

The model struggles with distinguishing between classes, as seen in the significant misclassification of "High" and "Low" categories.

```
# Plot confusion matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)
plt.figure(figsize=(8,6))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Blues',
            xticklabels=target_le.classes_, yticklabels=target_le.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Decision Tree Confusion Matrix')
plt.show()
```



Experiment-6



5. Naive Bayes

Naive Bayes Classifier: A probabilistic algorithm based on Bayes' theorem, assuming feature independence, commonly used for text classification and spam detection.

Advantages: Fast, efficient with small datasets, and performs well with high-dimensional data despite its strong independence assumption.

Experiment-6

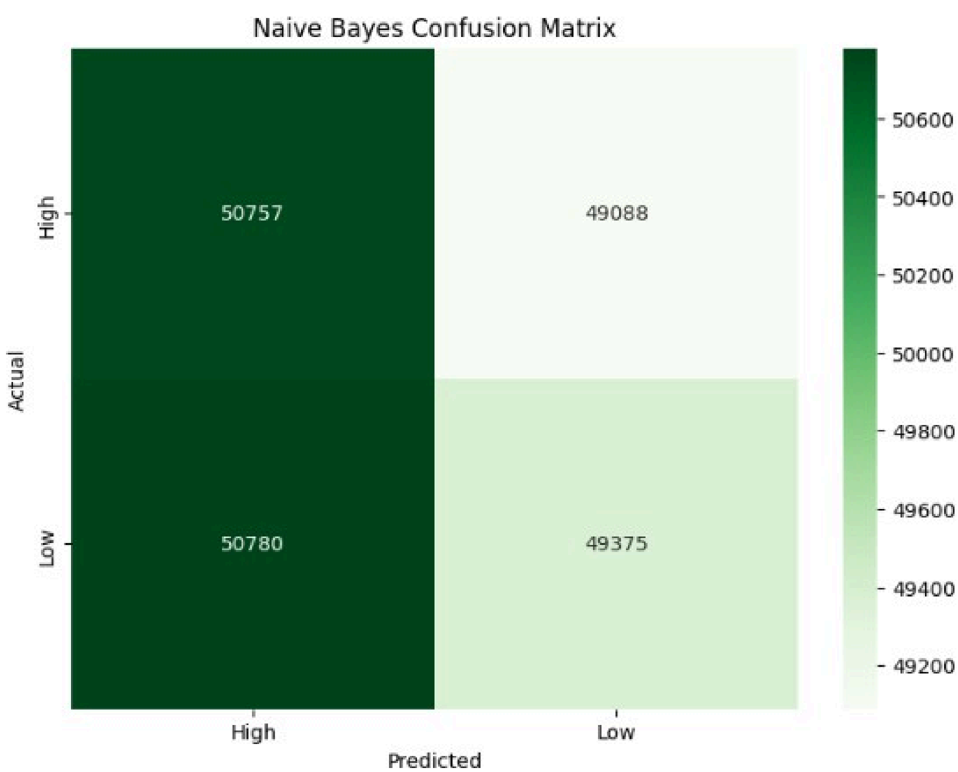
```python
# Predictions on the test set
y_pred_nb = nb_classifier.predict(X_test)

# Compute performance metrics for Naive Bayes
accuracy_nb = accuracy_score(y_test, y_pred_nb)
precision_nb = precision_score(y_test, y_pred_nb, average='weighted')
recall_nb = recall_score(y_test, y_pred_nb, average='weighted')
f1_nb = f1_score(y_test, y_pred_nb, average='weighted')

print("Naive Bayes Performance:")
print("Accuracy:", accuracy_nb)
print("Precision:", precision_nb)
print("Recall:", recall_nb)
print("F1 Score:", f1_nb)

# Plot confusion matrix for Naive Bayes
cm_nb = confusion_matrix(y_test, y_pred_nb)
plt.figure(figsize=(8,6))
sns.heatmap(cm_nb, annot=True, fmt='d', cmap='Greens',
            xticklabels=target_le.classes_, yticklabels=target_le.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Naive Bayes Confusion Matrix')
plt.show()
```

```
Naive Bayes Performance:
Accuracy: 0.50066
Precision: 0.5006732877789277
Recall: 0.50066
F1 Score: 0.5006308078888082
```



Experiment-6

Conclusion :

In our classification experiments, we used the Decision Tree classifier to predict The Decision Tree model, with a maximum depth set to 3 for clarity, achieved an accuracy of about

However, while it performed well on the majority class, its performance on the minority class was suboptimal, as observed in the classification report and confusion matrix.