

Experiment No. 6

Aim:Classification modelling

- a. Choose classifier for classification problem.
- b. Evaluate the performance of classifier.

Theory:

Classification Modeling:

Classification is a **supervised machine learning** technique used to predict categorical labels based on input features. It is widely used in **medical diagnosis, spam detection, fraud detection, and more**.

Several classifiers can be used for classification tasks, each with its advantages and limitations:

1. K-Nearest Neighbors (KNN):

- A non-parametric, instance-based learning algorithm.
- Classifies data based on the majority label of its **K** nearest neighbors.
- Works well for structured and small datasets but **slower for large datasets**.

2. Naïve Bayes:

- A probabilistic classifier based on **Bayes' Theorem**.
- Assumes **independence** between features, making it efficient for large datasets.
- Performs well in **text classification** but struggles when features are highly correlated.

3. Support Vector Machine (SVM):

- A powerful classifier that finds the **optimal hyperplane** for separating classes.
- Works well for **high-dimensional** datasets.
- Sensitive to **noisy data** and computationally expensive for large datasets.

4. Decision Tree:

- A tree-structured model that splits data based on feature values.
- **Easy to interpret** and requires minimal data preprocessing.
- Prone to **overfitting**, which can reduce generalization performance.

1. Load the dataset -

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv("/content/Loan_default.csv")
df = df.head()
```

	LoanID	Age	Income	LoanAmount	CreditScore	MonthsEmployed	\
0	9	56	85994	50587	520	80	
1	8	69	50432	124440	458	15	
2	3	46	84208	129188	451	26	
3	12	32	31713	44799	743	0	
4	5	60	20437	9139	633	8	

	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Education	\
0	4	15.23	36	0.44	0	
1	1	4.81	60	0.68	2	
2	3	21.17	24	0.31	2	
3	3	7.07	24	0.23	1	
4	4	6.51	48	0.73	0	

	EmploymentType	MaritalStatus	HasMortgage	HasDependents	LoanPurpose	\
0	0	0	1	1	4	
1	0	1	0	0	4	
2	3	0	1	1	0	
3	0	1	0	0	1	
4	3	0	0	1	0	

	HasCoSigner	Default
0	1	0
1	1	0
2	0	1
3	0	0

2.Splitting data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

3.KNN

```
# Classifiers
def train_and_evaluate(model, name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n{name} Classifier:\n")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print("Classification Report:\n", classification_report(y_test, y_pred))

# KNN
knn = KNeighborsClassifier(n_neighbors=3)
train_and_evaluate(knn, "K-Nearest Neighbors")
```



K-Nearest Neighbors Classifier:

Accuracy: 0.80

Classification Report:

	precision	recall	f1-score	support
0	0.75	1.00	0.86	3
1	1.00	0.50	0.67	2
accuracy			0.80	5
macro avg	0.88	0.75	0.76	5
weighted avg	0.85	0.80	0.78	5

Confusion Matrix:

```
[[4 0]
 [1 0]]
```

4. Naive Bayes

```
# Naive Bayes
nb = GaussianNB()
train_and_evaluate(nb, "Naive Bayes")
```

Naive Bayes Classifier:

Accuracy: 0.60

Classification Report:

	precision	recall	f1-score	support
0	0.60	1.00	0.75	3
1	0.00	0.00	0.00	2
accuracy			0.60	5
macro avg	0.30	0.50	0.38	5
weighted avg	0.36	0.60	0.45	5

Confusion Matrix:

```
[[4 0]
 [0 1]]
```

5.Support Vector Machine

```
# SVM
svm = SVC(kernel='linear')
train_and_evaluate(svm, "Support Vector Machine")
```



Support Vector Machine Classifier:



Accuracy: 0.60

Classification Report:

	precision	recall	f1-score	support
0	0.60	1.00	0.75	3
1	0.00	0.00	0.00	2
accuracy			0.60	5
macro avg	0.30	0.50	0.38	5
weighted avg	0.36	0.60	0.45	5

Confusion Matrix:

```
[[4 0]
 [0 1]]
```

6.Decision Tree

```
# Decision Tree
dt = DecisionTreeClassifier()
train_and_evaluate(dt, "Decision Tree")
```



Decision Tree Classifier:



Accuracy: 0.80

Classification Report:

	precision	recall	f1-score	support
0	0.75	1.00	0.86	3
1	1.00	0.50	0.67	2
accuracy			0.80	5
macro avg	0.88	0.75	0.76	5
weighted avg	0.85	0.80	0.78	5

```
Confusion Matrix:  
[[4 0]  
[0 1]]
```

Conclusion :

Conclusion for Loan Default Prediction Models

- KNN and Decision Tree achieved 80% accuracy, making them effective for loan default prediction.
- Naïve Bayes and SVM had lower accuracy (60%), indicating weaker performance.
- KNN and Decision Tree balanced precision and recall well, making them suitable for further tuning.

Improvements can be made with class balancing, feature selection, and hyperparameter optimization.