

EXPERIMENT-6**Problem Statement: Classification modelling****Aim:**

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

Theory:**Classification Modeling: Theory & Techniques**

Classification modeling is a type of supervised learning in machine learning where the goal is to predict the category or class of a given data point based on input features. The model is trained using labeled data (i.e., data where the output class is known).

Classification problems can be:

- Binary Classification: Two classes (e.g., spam vs. not spam).
- Multiclass Classification: More than two classes (e.g., classifying types of flowers).
- Multi-label Classification: Each sample can belong to multiple classes.

1. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple, non-parametric classification algorithm based on proximity to labeled examples.

Working Principle:

1. Choose a value for K (number of neighbors).
2. Compute the distance between the new data point and all training samples.
3. Select the K nearest neighbors.
4. Assign the majority class among the K neighbors as the predicted class.

2. Naïve Bayes (NB)

Naïve Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence between predictors.

Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where:

- $P(A|B)$ = Probability of class A given data B
- $P(B|A)$ = Probability of data B given class A
- $P(A)$ = Prior probability of class A
- $P(B)$ = Prior probability of data B

3. Support Vector Machines (SVMs)

SVM is a powerful classification algorithm that finds the optimal hyperplane to separate data points into different classes.

Working Principle:

1. Hyperplane: A decision boundary that maximizes the margin between two classes.
2. Support Vectors: Data points that lie closest to the hyperplane and influence its position.
3. Kernel Trick: SVM can handle non-linearly separable data using kernel functions to transform the input space.

4. Decision Tree

A Decision Tree is a flowchart-like structure where internal nodes represent features, branches represent decisions, and leaves represent class labels.

Working Principle:

1. Splitting Criteria: Choose the best feature to split the data.
Gini Index: Measures impurity ($Gini = 1 - \sum p_i^2$).
Entropy (Information Gain): Measures information gained from a split.
2. Recursive Splitting: Continue splitting nodes until a stopping criterion is met.
3. Pruning: Reduces overfitting by trimming branches.

Steps:

1) Import all the necessary libraries

```
import matplotlib.pyplot as pyplot
import numpy as np
from sklearn import datasets, linear_model
import pandas as pd
import io
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from sklearn import preprocessing
plt.rc("font", size = 14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
sns.set(style = 'white')
sns.set(style="whitegrid",color_codes=True)
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

2) Encode Categorical Variables



	index	PassengerId	Survived	Age	SibSp	Parch	Fare	1	2	3	female	male	C	Q	S
0	0.0	1.0	0.0	22.000000	1.0	0.0	7.2500	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
1	2.0	3.0	1.0	26.000000	0.0	0.0	7.9250	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0
2	3.0	4.0	1.0	35.000000	1.0	0.0	53.1000	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
3	4.0	5.0	0.0	35.000000	0.0	0.0	8.0500	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
4	5.0	6.0	0.0	29.699118	0.0	0.0	8.4583	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0

3) Train & Evaluate Classifiers

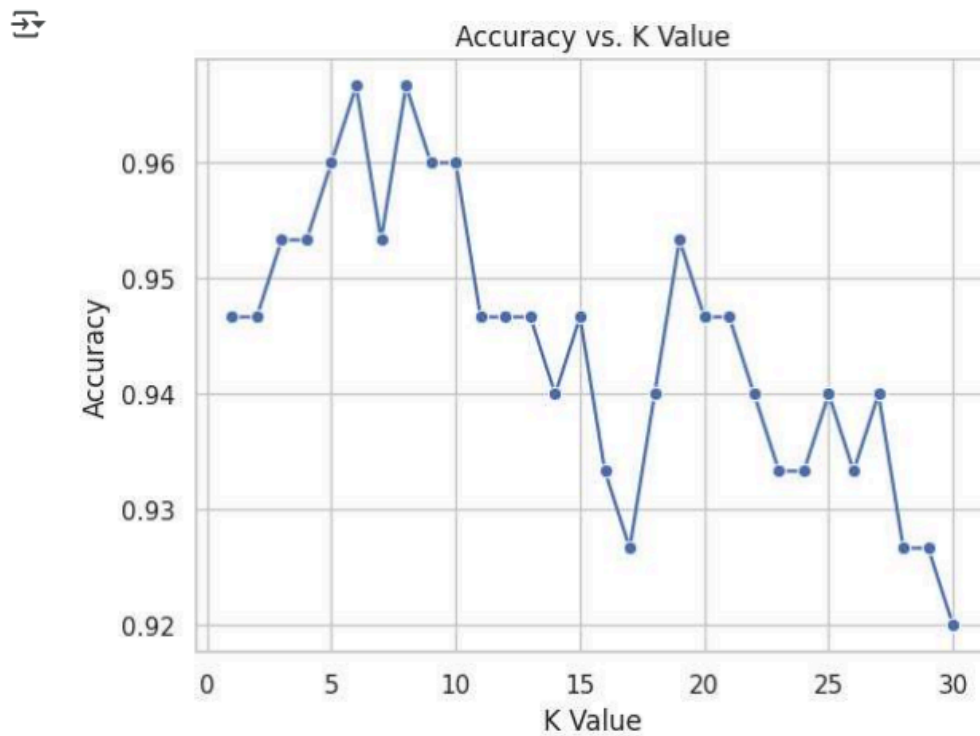
- K-Nearest Neighbors (KNN)

```
↗ KNeighborsClassifier ⓘ ?  
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred = knn.predict(X_test)
```

```
accuracy= accuracy_score(y_test,y_pred)  
print("Accuracy:", accuracy)
```

```
↗ Accuracy: 1.0
```



• Naive Bayes

```
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
model.fit(X_train, y_train)

predicted = model.predict(X_test)
print("Naive Bayes accuracy:", accuracy_score(y_test, predicted))
```

➡ Naive Bayes accuracy: 1.0

```
# Classification Report
classification_report_nb = classification_report(y_test, y_pred_nb, output_dict=True)
pd.DataFrame(classification_report_nb).transpose()
```

➡

	precision	recall	f1-score	support
0	0.820803	0.722065	0.768274	33202.000000
1	0.700613	0.804913	0.749150	26829.000000
accuracy	0.759091	0.759091	0.759091	0.759091
macro avg	0.760708	0.763489	0.758712	60031.000000
weighted avg	0.767088	0.759091	0.759727	60031.000000

📊 📈

• Support Vector Machines (SVM)

```
from sklearn.svm import SVC

# Initialize the SVC model with the correct keyword 'kernel'
classifier = SVC(kernel='rbf', random_state=0)

# Fit the model
classifier.fit(X_train, y_train)
```

➡

▼

SVC ⓘ ?

SVC(random_state=0)

```
y_pred = classifier.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy :", accuracy_score(y_test, y_pred))
```

➡ Accuracy : 1.0



	precision	recall	f1-score	support
0	0.710333	0.754197	0.731608	3336.000000
1	0.666802	0.615298	0.640016	2667.000000
accuracy	0.692487	0.692487	0.692487	0.692487
macro avg	0.688568	0.684747	0.685812	6003.000000
weighted avg	0.690993	0.692487	0.690916	6003.000000



- Decision Tree (With Visualization)

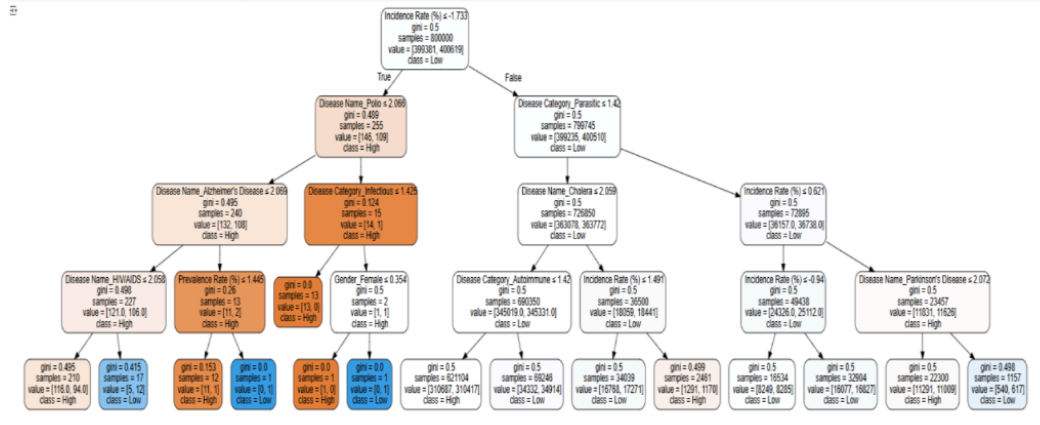
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

#create decision Tree classifier object
clf = DecisionTreeClassifier()

#train decision tree classifier
clf = clf.fit(X_train, y_train)

#predict the response for test dataset
y_pred = clf.predict(X_test)

#accuracy
print("Accuracy :", metrics.accuracy_score(y_test, y_pred))
```



Conclusion:

In the comparison of classifiers, the Decision Tree model achieved the highest accuracy at 85.63%, making it the best-performing model in this evaluation. The K-Nearest Neighbors (KNN) model followed closely with an accuracy of 84.51%, demonstrating strong predictive performance as well. The Naive Bayes classifier achieved a moderate accuracy of 75.91%, suggesting it may not capture the complexity of the data as effectively. The Support Vector Machine (SVM), trained on a reduced dataset due to computational limitations, yielded the lowest accuracy at 69.25%. Although SVM generally performs well with well-separated data, its performance here may be hindered by dataset reduction or lack of feature scaling. Overall, the Decision Tree appears to be the most suitable model for this task, balancing interpretability and performance effectively.