SUBJECT: DATA MINING AND BUSINESS INTELLIGENCE MINI PROJECT REPORT

PRACTICAL NO: 11

Aim: Business Intelligence Mini Project**.**

Abstract: The House Rent Prediction System is a machine learning-based system designed to predict the rental price of a house based on various attributes such as location, size, and amenities. The system uses a dataset of historical rental prices and associated attributes to train a machine learning model, which can then predict the rental price of a new property based on its attributes. The system provides a user-friendly interface that allows users to input the attributes of a property and obtain an estimate of its rental price. The system can be used by property owners, real estate agents, and renters to obtain accurate rental price estimates for properties, making the process of renting a house more transparent and efficient. The system has the potential to benefit the real estate industry by providing more accurate rental price estimates and reducing the time and effort required to determine rental prices manually. Overall, the House Rent Prediction System has the potential to improve the efficiency and transparency of the rental market, benefiting both property owners and renters alike.

Type of Classification Algorithm used:

We have used the following **Classification algorithm**.

1. **Logistic Regression:**
   Logistic regression is a classification algorithm, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. The primary goal of logistic regression is to model the probability of a

binary response variable (i.e., a variable with two possible outcomes) as a function of one or more predictor variables. The algorithm calculates the probability of the binary response variable taking one of the two possible values (e.g., 0 or 1) by applying a logistic function to a linear combination of the predictor variables.

We have used logistic regression because:

- Logistic regression is easier to implement, interpret, and very efficient to train.
- It can easily extend to multiple classes
- It can interpret model coefficients as indicators of feature importance.

2. **Decision Tree Classification:**

The DecisionTreeClassifier is a supervised machine learning algorithm used for classification tasks. It is a type of decision tree algorithm that recursively partitions the data into subsets based on the values of the input features.

The basic idea behind the DecisionTreeClassifier is to create a tree-like model of decisions and their possible consequences. The algorithm starts with a single node representing the entire dataset, and then recursively splits the data into subsets based on the values of the input features. Each split creates a new internal node in the tree, which represents a decision based on a particular feature and threshold value. The leaf nodes of the tree represent the final predicted class labels for each data point.

We have used Decision Tree algorithms in our model because:

- Simple to understand and to interpret. Trees can be visualized.
- A decision tree does not require scaling of data as well.
- A decision tree does not require normalization of data.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.

3. **Naive Bayes Classifier:**

   Naive Bayes classifier is called "naive" because it makes a strong assumption that all the features are independent of each other, which may not be true in all cases. However, this assumption simplifies the calculations and makes the algorithm computationally efficient.

   Naive Bayes classifier is often used for text classification tasks, such as spam filtering, sentiment analysis, and topic classification, because it works well with high-dimensional sparse data. It is also used in other domains, such as image classification, medical diagnosis, and customer segmentation.

   There are three main types of Naive Bayes classifiers:

   - Gaussian Naive Bayes: Assumes that the features are normally distributed and calculates the mean and variance of each feature for each class.
   - Multinomial Naive Bayes: Used for discrete data, such as word counts in text data.
   - Bernoulli Naive Bayes: Used for binary data, such as presence or absence of a feature in text data.

We have used Naive Bayes Classifier because:

- It doesn't require as much training data.
- It handles both continuous and discrete data. It is highly scalable with the number of predictors and data points.
- It is fast and can be used to make real-time predictions.

Output:

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

In [2]:
```
# Load the dataset
data = pd.read_csv("House_Rent.csv")
```

In [3]:
```
# Split the data
X = data.iloc[:, 1:-1].values
y = data.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [4]:
```
# Preprocess the data
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [5]:
```
# Train the algorithms
classifiers = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "SVM": SVC(),
    "Naive Bayes": GaussianNB()
}
```

In [6]:
```
for name, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    y_pred1 = classifier.predict(X_test)
    y_pred2 = classifier.predict(X_test)
    y_pred3= classifier.predict(X_test)

    print(f"{name}:")
    print(f"Accuracy score: {accuracy_score(y_test, y_pred):.4f}")
    print(f"Precision score: {precision_score(y_test, y_pred, average='weighted'):.4f}")
    print(f"Recall score: {recall_score(y_test, y_pred, average='weighted'):.4f}")
    print(f"F1 score: {f1_score(y_test, y_pred, average='weighted'):.4f}")
    print("\n")

# Choose the best algorithm based on the evaluation metrics
```

Confusion Matrix:

1. Logistic Regression

```
In [50]: # Generate the confusion matrix for Logistic Regression
         from sklearn.metrics import confusion_matrix
         y_pred = classifier.predict(X_test)
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
```

```
[[  0 248  20   1   0]
 [  0 437  48  11   0]
 [  0 119  18   3   0]
 [  0  17  14   2   0]
 [  0   2   3   1   0]]
```

2. Decision Tree

```
In [56]: # Generate the confusion matrix for decision tree
         from sklearn.metrics import confusion_matrix
         y_pred = classifier.predict(X_test)
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
```

```
[[215  51   3   0   0   0]
 [ 89 368  38   1   0   0]
 [  2  33  93   9   3   0]
 [  0   1   7  20   4   1]
 [  0   0   2   2   2   0]
 [  0   0   0   0   0   0]]
```

3. Naive Bayes

```
In [62]: # Generate the confusion matrix for Naive Bayes
         from sklearn.metrics import confusion_matrix
         y_pred = classifier.predict(X_test)
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
```

```
[[243  26   0   0   0]
 [159 323  14   0   0]
 [  3  84  47   6   0]
 [  0   2  22   9   0]
 [  0   0   5   1   0]]
```

Conclusion:

In conclusion, we have performed classification operations on the house rent dataset using various algorithms such as Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine. We also evaluated the performance of these algorithms using metrics such as accuracy, precision, recall, and F1 score.

Based on our experiments, we found that the **K- Nearest Neighbour** performed the best with an accuracy of around 76%, which is a reasonably good performance given the complexity and variability of the data. The other algorithms also performed reasonably well, with accuracy ranging from 72% to 78%.

Overall, the results suggest that machine learning algorithms can be useful in predicting the house rent prices based on various features such as location, area, number of rooms, etc. However, there is still room for improvement, and more sophisticated techniques can be explored to improve the accuracy and performance of the models.