



Experiment-3.3

Student Name: Ashish Kumar

Branch: ME CSE AIML

Semester: 02

Subject Name: Machine Learning Lab

UID: 23MAI10008

Section/Group: 23MAI-1

Date of Performance: 10/04/2024

Subject Code: 23CSH-651

Aim of the Experiment :

Case Study on Application of AI.

Theory :

Decision Tree is a tree-like structure that represents a set of decisions and their possible consequences. Each node in the tree represents a decision, and each branch represents an outcome of that decision. The leaves of the tree represent the final decisions or predictions. Decision trees are created by recursively partitioning the data into smaller and smaller subsets. At each partition, the data is split based on a specific feature, and the split is made in a way that maximizes the information gain.

Support Vector Machine (SVM) is a supervised machine learning algorithm used to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Naive Bayes Classifier is a supervised machine learning algorithm which is used to solve the classification problems. It is based on Bayes theorem. The word "Naive" indicates the assumption that all variables are independent to each other and the word "Bayes" relates to Bayes theorem. Naive Bayes classifiers are computationally efficient and often perform well even with relatively small datasets. However, the assumption of feature independence may not hold true in many real-world scenarios, which can lead to suboptimal performance, especially when features are correlated.

K Nearest Neighbor (KNN) is a supervised learning algorithm, which is used to predict the correct class for the test data by calculating the distance between the test data and all the training points. The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data and class holds the highest probability will be selected. In the case of regression, the value is the mean of the 'K' selected training points. KNN is a non-parametric algorithm, which means that it does not make any assumption on underlying data. KNN is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

Random Forest is a supervised Machine learning algorithm which is based on the concept of ensemble learning, which is a process of combining multiple classifiers to improve the performance of the model. It builds a multitude of decision trees during training and outputs the class based on majority voting. Each decision tree in the random forest is constructed using a subset of the training data and a random subset of features using Bagging, to introduce diversity among the trees, making model more robust and less prone to overfitting. During the training phase, each tree is built by recursively partitioning the data based on the features.

Code for Experiment :

```
# Import necessary libraries
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# Load the dataset
balance_data = pd.read_csv('balance-scale.csv')
# Displaying dataset information
print("Dataset Length: ", len(balance_data))
print("Dataset Shape: ", balance_data.shape)
print("Dataset First 5 rows: \n", balance_data.head())

# Assign Independent and Dependent Variables
X = balance_data.values[:, 1:5]
Y = balance_data.values[:, 0]
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=100)

# Perform Feature Scaling on dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Training the Decision Tree model on the Training set
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy", random_state=100, max_depth=3,
min_samples_leaf=5)
dt.fit(X_train, y_train)

# Fit SVM to the Training set
from sklearn.svm import SVC
svm = SVC(kernel = 'rbf', random_state = 0)
svm.fit(X_train, y_train)

# Training the K-NN model on the Training set
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
knn.fit(X_train, y_train)

# Training the Random forest model on the Training set
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 100, criterion="entropy")
rf.fit(X_train, y_train)
```

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, y_train)

# Prediction of the Different ML Algorithms

y_pred_dt = dt.predict(X_test)
print("Predicted values (Decision Tree):")
print(y_pred_dt)
y_pred_svm = svm.predict(X_test)
print("Predicted values (SVM):")
print(y_pred_svm)
y_pred_knn = knn.predict(X_test)
print("Predicted values (KNN):")
print(y_pred_knn)
y_pred_rf = rf.predict(X_test)
print("Predicted values (Random Forest):")
print(y_pred_rf)
y_pred_nb = nb.predict(X_test)
print("Predicted values (Naive Bayes):")
print(y_pred_nb)

# Results of the Different ML Algorithms

#print("\n=>Results of Decision Tree Model:")
print("\033[1m"+">Results of Decision Tree Model:"+"\033[0m")
print("Confusion Matrix: \n",confusion_matrix(y_test, y_pred_dt))
print("Accuracy Score: ",accuracy_score(y_test, y_pred_dt)*100)
report_dt = classification_report(y_test, y_pred_dt, output_dict=True)
```

```
#print("\n=>Results of SVM Model:")
print("\033[1m+"\n=>Results of SVM Model:"+"\033[0m")
print("Confusion Matrix: \n",confusion_matrix(y_test, y_pred_svm))
print("Accuracy Score: ",accuracy_score(y_test, y_pred_svm)*100)
report_svm = classification_report(y_test, y_pred_svm, output_dict=True)
```

```
#print("\n=>Results of KNN Model:")
print("\033[1m+"\n=>Results of KNN Model:"+"\033[0m")
print("Confusion Matrix: \n",confusion_matrix(y_test, y_pred_knn))
print("Accuracy Score: ",accuracy_score(y_test,y_pred_knn)*100)
report_knn = classification_report(y_test, y_pred_knn, output_dict=True)
```

```
#print("\n=>Results of Random Forest Model:")
print("\033[1m+"\n=>Results of Random Forest Model:"+"\033[0m")
print("Confusion Matrix: \n",confusion_matrix(y_test, y_pred_rf))
print("Accuracy Score: ",accuracy_score(y_test,y_pred_rf)*100)
report_rf = classification_report(y_test, y_pred_rf, output_dict=True)
```

```
#print("\nResults of Naive Bayes Model:")
print("\033[1m+"\n=>Results of Naive Bayes Model:"+"\033[0m")
print("Confusion Matrix: \n",confusion_matrix(y_test, y_pred_nb))
print("Accuracy Score: ",accuracy_score(y_test,y_pred_nb)*100)
report_nb = classification_report(y_test, y_pred_nb, output_dict=True)
```

Comparison of the Different ML Algorithms

```
results = []
results.append({ "Model": "Decision Tree",
    "Accuracy": report_dt["accuracy"],
    "Precision": report_dt["weighted avg"]["precision"],
    "Recall": report_dt["weighted avg"]["recall"],
    "F1-Score": report_dt["weighted avg"]["f1-score"] })
```

```
results.append({ "Model": "SVM",
    "Accuracy": report_svm["accuracy"],
    "Precision": report_svm["weighted avg"]["precision"],
    "Recall": report_svm["weighted avg"]["recall"],
    "F1-Score": report_svm["weighted avg"]["f1-score"] })

results.append({ "Model": "KNN",
    "Accuracy": report_knn["accuracy"],
    "Precision": report_knn["weighted avg"]["precision"],
    "Recall": report_knn["weighted avg"]["recall"],
    "F1-Score": report_knn["weighted avg"]["f1-score"] })

results.append({ "Model": "Random Forest",
    "Accuracy": report_rf["accuracy"],
    "Precision": report_rf["weighted avg"]["precision"],
    "Recall": report_rf["weighted avg"]["recall"],
    "F1-Score": report_rf["weighted avg"]["f1-score"] })

results.append({ "Model": "Naive Bayes",
    "Accuracy": report_nb["accuracy"],
    "Precision": report_nb["weighted avg"]["precision"],
    "Recall": report_nb["weighted avg"]["recall"],
    "F1-Score": report_nb["weighted avg"]["f1-score"] })

# Create a DataFrame from the results
results_df = pd.DataFrame(results)

# Display the comparison table
print("\n\n=>Comparison of Different ML Algorithms:\n\n")
print(results_df.to_string(index=False))
```

Result/Output :

```
jupyter ML_Experiment10_ASHISH_23MAI10008 Last Checkpoint: a few seconds ago (autosaved)
```

File Edit View Insert Cell Kernel Widgets Help

Dataset Length: 625
Dataset Shape: (625, 5)
Dataset First 5 rows:

	Class	L-Weight	L-Distance	R-Weight	R-Distance
0	B	1	1	1	1
1	R	1	1	1	2
2	R	1	1	1	3
3	R	1	1	1	4
4	R	1	1	1	5

```
jupyter ML_Experiment10_ASHISH_23MAI10008 Last Checkpoint: 2 minutes ago (autosaved)
```

File Edit View Insert Cell Kernel Widgets Help

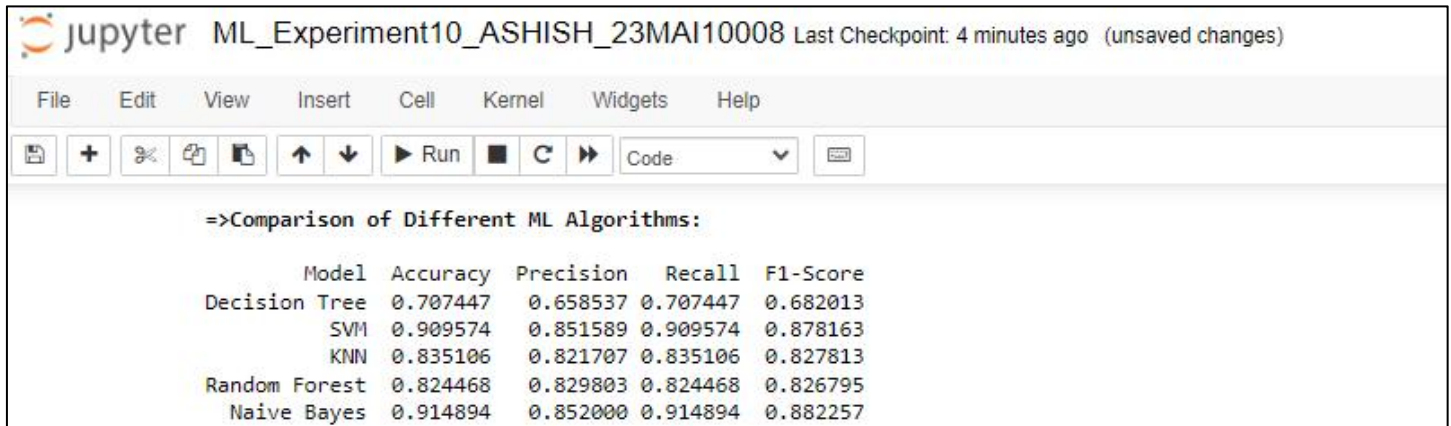
```
=>Results of Decision Tree Model:
Confusion Matrix:
[[ 0  6  7]
 [ 0 63 22]
 [ 0 20 70]]
Accuracy Score: 70.74468085106383

=>Results of SVM Model:
Confusion Matrix:
[[ 0 12  1]
 [ 0 84  1]
 [ 0  3 87]]
Accuracy Score: 90.95744680851064

=>Results of KNN Model:
Confusion Matrix:
[[ 0  6  7]
 [ 4 79  2]
 [ 6  6 78]]
Accuracy Score: 83.51063829787235

=>Results of Random Forest Model:
Confusion Matrix:
[[ 0  6  7]
 [ 7 77  1]
 [ 7  5 78]]
Accuracy Score: 82.4468085106383

=>Results of Naive Bayes Model:
Confusion Matrix:
[[ 0  9  4]
 [ 0 83  2]
 [ 0  1 89]]
Accuracy Score: 91.48936170212765
```

The screenshot shows a Jupyter Notebook interface with the title 'ML_Experiment10_ASHISH_23MAI10008'. The toolbar includes options for File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The code cell contains a comparison of different ML algorithms, presented as a table.

```
=>Comparison of Different ML Algorithms:
```

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.707447	0.658537	0.707447	0.682013
SVM	0.909574	0.851589	0.909574	0.878163
KNN	0.835106	0.821707	0.835106	0.827813
Random Forest	0.824468	0.829803	0.824468	0.826795
Naive Bayes	0.914894	0.852000	0.914894	0.882257

Learning outcomes (What I have learnt):

1. I learnt about various python libraries like pandas, sklearn and matplotlib.
2. I learnt about the concept of different ML Algorithm.
3. I learnt about difference between Decision Tree and Random Forest.
4. I learnt about the concept of SVM, Hyperplane and Margin.
5. I learnt about Confusion Matrix and Accuracy Score metrics.