CHANDRALEKHA CHILUMULA
101745900

ASSIGNMENT 4

TRADITIONAL MACHINE LEARNING CLASSIFICATION

DESCRIPTION:

In this assignment, we are required to build 3 total classifiers for the dataset. It is as follows

- k-Nearest Neighbors (KNN) Classifier
- Decision Tree Classifier
- Random Forest Classifier

INTRODUCTION TO CLASSIFIERS:

k-Nearest Nearest Neighbors (KNN)**:**

The k-nearest neighbors algorithm, sometimes referred to as KNN or k-NN, is a supervised learning classifier that employs proximity to produce classifications or predictions about the grouping of a single data point. Although it can be applied to classification or regression issues, it is commonly employed as a classification algorithm because it relies on the idea that comparable points can be discovered close to one another.

The KNN algorithm's "n neighbors" parameter, which represents the number of neighbors to take into account, is one of important parameters. The user typically sets the value of n neighbors, which specifies how many of the user's closest neighbors will be taken into account for prediction.

Decision Tree Classifier**:**

Another supervised learning method that is primarily employed for classification tasks is the decision tree classifier. As long as a user-defined stop condition is satisfied or each subset contains only one target variable, the dataset is recursively divided into subsets using the feature that provides the highest information gain. The decision tree model can be graphically depicted, with each leaf node representing a class label and each internal node of the tree signifying a choice based on a feature. The decision tree classifier can be visualized, understood, and used in many different ways. When dealing with complex data that has a large bias and variance or when the data is unbalanced, it might be vulnerable to overfitting and may not perform as expected.

Random Forest Classifier**:**

Random Forest is a well-known supervised learning technique that can be used for classification, regression, and other machine-learning problems. It is a machine learning technique that builds a large number of decision trees during training period and produces the class that is the mean of the predictions (regression) or classifications of the individual trees. By using numerous decision trees, each of which is trained using a random portion of the training data and a random subset of the features, the random forest approach aggregates their predictions. Random forest enhances the accuracy and robustness of the model by lowering overfitting and variance. Because of its excellent accuracy, scalability, and aptitude for handling noisy and large-scale data, random forest is a favorite among machine learning

algorithms. Many issues, including picture categorization, fraud detection, and financial forecasting, have effectively been used it.

CODE SNIPPET:

```
+ Code  + Text

# Load the heart disease dataset
data_frame = pd.read_csv('/content/sample_data/heart.csv')

# Split the dataset into training (75%)c and testing (25%) sets
train_df, test_df = train_test_split(data_frame, test_size=0.25, random_state=42)

# Split the features and target variable for the training and testing sets
X_train = train_df.drop('HeartDisease', axis=1)
y_train = train_df['HeartDisease']
X_test = test_df.drop('HeartDisease', axis=1)
y_test = test_df['HeartDisease']

# Define simulation parameters for each classifier
sim_params = {
    'k_nearest_neighbhor': {'k': 4},
    'decision_tree': {'maximun_depth': 4},
    'random_forest': {'n_estimators': 100, 'maximun_depth': 4}
}

# Define an empty list to store the results of each classifier
results = []

# Loop through each classifier and evaluate its performance
for classifier_name, classifier_params in sim_params.items():
    # Instantiate the classifier with the given parameters
    if classifier_name == 'k_nearest_neighbhor':
        clf = KNeighborsClassifier(n_neighbors=classifier_params['k'])
    elif classifier_name == 'decision_tree':
        clf = DecisionTreeClassifier(max_depth=classifier_params['max_depth'])
    elif classifier_name == 'random_forest':
        clf = RandomForestClassifier(n_estimators=classifier_params['n_estimators'],
                                     max_depth=classifier_params['max_depth'])
```

Figure 1

CODE OUTPUT:

```
print( F1 Score: , res[ F1 Score ])


Results for knn:

Confusion Matrix:
 [[87 25]
 [73 91]]
Accuracy: 0.644927536231884
Precision: 0.7844827586206896
Recall: 0.5548780487804879
F1 Score: 0.65

Results for decision_tree:

Confusion Matrix:
 [[ 93  19]
 [ 21 143]]
Accuracy: 0.855072463768116
Precision: 0.8827160493827161
Recall: 0.8719512195121951
F1 Score: 0.8773006134969324

Results for random_forest:

Confusion Matrix:
 [[ 97  15]
 [ 17 147]]
Accuracy: 0.8840579710144928
Precision: 0.9074074074074074
Recall: 0.8963414634146342
F1 Score: 0.901840490797546
```

CHANDRALEKHA CHILUMULA
101745900

SIMULATION PARAMETERS:

S.NO	CLASSIFIER	SIMULATION PARAMETERS
1	KNN	n_neigbors
2	Decision Tree	max_depth
3	Random Forest	Max_depth, n_estimators

DECISION TREE**:**

MAX DEPTH: This option regulates the maximum depth to which the decision tree can expand. A tree may overfit the training data if the maximum depth is set too high, whereas a tree may underfit the data if the maximum depth is set too low.

The split criterion specifies the measure that will be used to choose the best split at each node of the tree. Gini impurity, entropy, and information gain are a few typical split criteria**.**

RANDOM FOREST ALGORITHM**:**

n_estimators: The random forest's decision tree count is controlled by this option. The model's accuracy can be increased by adding more trees, although doing so raises the computational cost.

Max depth: This setting regulates the decision trees' maximum depth in the random forest. Higher accuracy may result from increasing the maximum depth, but overfitting may also rise as a result.

Criteria: This parameter establishes the metric by which the effectiveness of each split in the decision trees is measured.

K NEIGHBORS:

n_neighbors: The K-nearest neighbors (KNN) algorithm's "neighbors" option sets the number of closest neighbors to take into account when producing a prediction.

CONFUSION MATRIX:

KNN:

	Actually positive(1)	Actually negative(0)
Predicted positive(1)	87	25
Predicted negative(0)	73	91

DECISION TREE:

	Actually positive(1)	Actually negative(0)
Predicted positive(1)	93	19
Predicted negative(0)	21	143

RANDOM FOREST:

	Actually positive(1)	Actually negative(0)
Predicted positive(1)	97	15
Predicted negative(0)	17	147

PERFORMANCE ON TESTING SET:

S.NO	CLASSIFIER	ACCURACY	PRECISION	RECALL	F1 SCORE
1.	KNN	0.6449275362318884	0.7844827586206896	0.5548780487804879	0.65
2.	Decision Tree	0.8550724637681160	0.8827160493827161	0.8719512195121951	0.8773006134969324
3.	Random Forest	0.8913043478260869	0.9147286821705426	0.8939393939393939	0.9042145593869731

We have access to a lot of data from the simulation, including all of the model-specific simulation parameters. Also, we have three confusion matrices, one for each classifier, which might focus on how well each classifier is performing.

A table that lists each classifier's performance on the testing set has been created to assist us in better understanding how well it performed. Important matrices including accuracy, precision, recall, and f1 score are included in this table, which may be used to assess and compare how well each classifier performs.

The Parameters that Affected their Model's Performance:

The selection of parameters is a key component that significantly affects the efficacy of a machine learning model. k, the number of trees, and tree depth are the variables that regulate the behavior of the model and greatly affect its accuracy and resilience. The best results must be obtained by examining several values of each parameter and evaluating the model's performance for each structure. This necessitates an iterative process of changing parameter values, training the model, and evaluating its performance up until the perfect set of parameters is found. The process of parameter tweaking is important because it enhances the model's overall performance by enabling appropriate adaptation to novel and untested inputs. Thus, a key step in developing an accurate and trustworthy machine-learning model is selecting the appropriate set of parameter values.

k-Nearest Neighbors (k-NN) Classifier:

The k-nearest neighbor was providing the highest level of accuracy for this set of data on heart disease at value 5, which was 0.6869565217391305. After performing some cross-validation, I came to the conclusion that the ideal number for this data set to achieve maximum accuracy is 5, since the accuracy was rapidly declining as I increased the nearest neighbor value.

Decision Tree Classifier:

Max depth was the decision tree's most effective parameter. When I first tried to give the decision tree the maximum depth of 2, I received an accuracy of 0.8391304347826087;

however, when I tried to expand it further with more like 6, 7, 8, and so on, the overfitting worsened and the accuracy fell to 76.39 when the maximum depth was 10. I got the highest accuracy of 0.8869565217391304 when I gave the maximum depth of 4.

Random Forest Classifier:

When I assigned the value 50 for the n-estimators (number of trees) parameter for the random forest algorithm, I obtained an accuracy of 85.650.8826086956521739, and as I increased the value, the accuracy gradually improved. As I tried to increase the value of max depth together with n-estimators, the accuracy was reduced. I have tried to adjust the max depth value in conjunction with the n-estimators.

The accuracy was 0.8913043478260869 at value n_estimators 90, and as I increased the n-estimators value, the accuracy started to decline. The main cause of this decline in accuracy after increasing the n-estimators value to 100 was that as the number of trees increased, so did the model's training time.

CONCLUSION:

Because Random Forest combines many decision trees or KNN models to give a final prediction, overfitting (memorizing the data) is reduced, missing data is handled, and the model's accuracy is increased. As a result, Random Forest performs better than Decision Trees and KNN. Moreover, random forest has much higher accuracy than any other method. By building each decision tree using a subset of features chosen at random, Random Forest is able to handle high-dimensional data. In general, Random Forest is a more potent and reliable algorithm than KNN and Decision Trees.