EXPERIMENT 3

AIM

To implement and compare Decision Tree and Random Forest algorithms for classification on a real-world healthcare dataset in order to predict the occurrence of stroke and evaluate their performance using classification metrics.

OBJECTIVES

1. To understand the working of Decision Tree classification algorithm.
2. To implement Decision Tree for predicting stroke occurrence using healthcare data.
3. To understand the concept of ensemble learning and Random Forest algorithm.
4. To implement Random Forest classifier and improve prediction accuracy.
5. To evaluate and compare both models using metrics such as Accuracy, Confusion Matrix, Precision, Recall, and F1-score.
6. To analyze the effect of model complexity and overfitting in Decision Tree and Random Forest.

THEORY

Classification is a supervised machine learning technique used to assign data into predefined classes or categories based on input features. In this experiment, classification algorithms are used to predict whether a patient is likely to have a stroke (0 or 1).

Decision Tree Classification

Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It works by splitting the dataset into subsets based on the most significant feature, forming a tree-like structure.

Each internal node represents a feature, each branch represents a decision rule, and each leaf node represents an output class.

The algorithm selects the best feature for splitting using criteria such as:

- Gini Index
- Entropy (Information Gain)

The process continues recursively until a stopping condition is reached, such as maximum depth or minimum samples.

Advantages of Decision Tree:

- Easy to understand and interpret
- Handles both numerical and categorical data
- Requires little data preprocessing

Limitations:

- Prone to overfitting
- Sensitive to small variations in data
- Can create complex trees if not controlled

Random Forest Classification

Random Forest is an ensemble learning technique that combines multiple Decision Trees to improve performance and reduce overfitting.

It works by creating multiple Decision Trees using different subsets of the dataset and features. Each tree makes a prediction, and the final output is determined by majority voting.

Key concepts:

- Bagging (Bootstrap Aggregation): Random sampling of data with replacement
- Feature randomness: Selecting random features for each split

Advantages of Random Forest:

- Reduces overfitting compared to Decision Tree
- Provides higher accuracy and stability
- Handles large datasets efficiently

Limitations:

- More computationally expensive
- Less interpretable than a single Decision Tree

Dataset Description

The dataset used in this experiment is a healthcare stroke dataset containing patient information such as age, gender, hypertension, heart disease, average glucose level, BMI, smoking status, and other health indicators.

- The dataset includes both numerical and categorical features.
- The target variable is “stroke”, which is binary:
  0 → No Stroke
  1 → Stroke

Data preprocessing steps include:

- Removing unnecessary columns such as ID
- Handling missing values in BMI using mean imputation
- Encoding categorical variables into numerical format
- Splitting the dataset into training and testing sets

Evaluation Metrics

The performance of the classification models is evaluated using:

Accuracy:
The ratio of correctly predicted instances to total instances.

Confusion Matrix:
A table that shows correct and incorrect predictions in terms of True Positive, True Negative, False Positive, and False Negative.

Precision:
Measures how many predicted positive cases are actually correct.

Recall:
Measures how many actual positive cases are correctly predicted.

F1-Score:
The harmonic mean of precision and recall.

CODE

```
# ========================================
# STEP 1: Import Libraries
```

```
# ========================================

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import LabelEncoder


# ========================================
# STEP 2: Load Dataset
# ========================================

df = pd.read_csv("stroke.csv")
print(df.head())


# ========================================
# STEP 3: Data Preprocessing
# ========================================

# Drop ID column
df.drop("id", axis=1, inplace=True)

# Fill missing BMI values
df["bmi"].fillna(df["bmi"].mean(), inplace=True)

# Encode categorical columns
le = LabelEncoder()

cat_cols =
["gender","ever_married","work_type","Residence_type","smoking_status"]
```

```
for col in cat_cols:
    df[col] = le.fit_transform(df[col])

print("\nAfter Encoding:")
print(df.head())


# ========================================
# STEP 4: Define Features and Target
# ========================================

X = df.drop("stroke", axis=1)
y = df["stroke"]


# ========================================
# STEP 5: Train-Test Split
# ========================================

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)


# ========================================
# STEP 6: DECISION TREE CLASSIFIER
# ========================================

dt = DecisionTreeClassifier(max_depth=5, random_state=42)
dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)

print("\n===== DECISION TREE RESULTS =====")
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
print("\nClassification Report:\n", classification_report(y_test,
y_pred_dt))
```

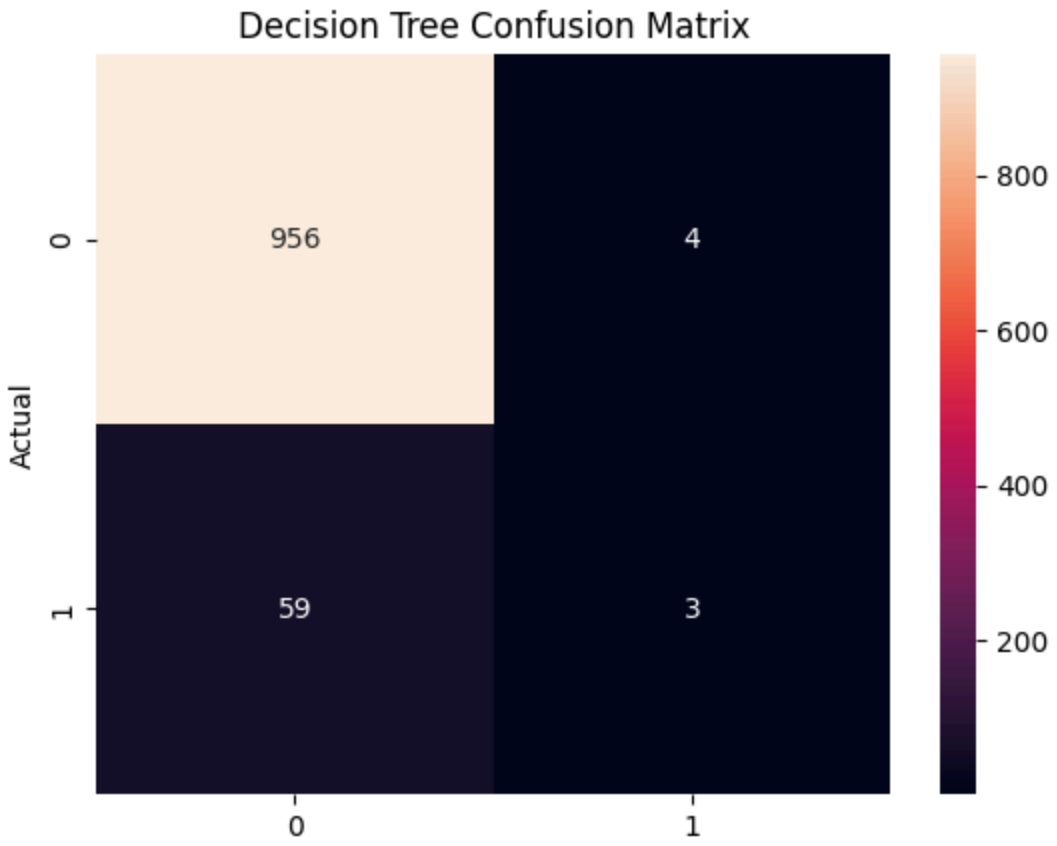
```
# Plot Confusion Matrix
sns.heatmap(confusion_matrix(y_test, y_pred_dt), annot=True, fmt="d")
plt.title("Decision Tree Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()


# ========================================
# STEP 7: RANDOM FOREST CLASSIFIER
# ========================================

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

y_pred_rf = rf.predict(X_test)

print("\n===== RANDOM FOREST RESULTS =====")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("\nClassification Report:\n", classification_report(y_test,
y_pred_rf))


# Plot Confusion Matrix
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt="d")
plt.title("Random Forest Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()


# ========================================
# STEP 8: MODEL COMPARISON
# ========================================

models = ["Decision Tree", "Random Forest"]
accuracies = [
    accuracy_score(y_test, y_pred_dt),
    accuracy_score(y_test, y_pred_rf)
```

```
]

plt.bar(models, accuracies)
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy")
plt.show()
```

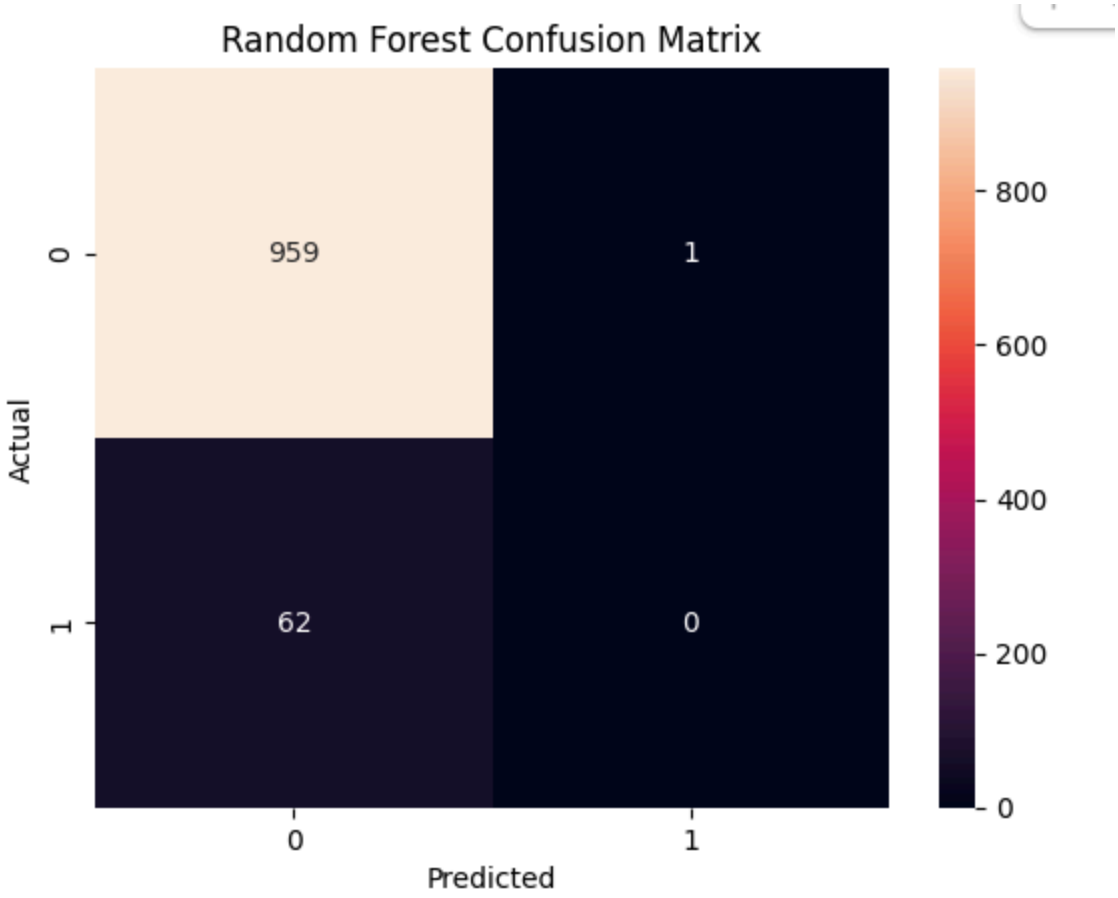
OUTPUT

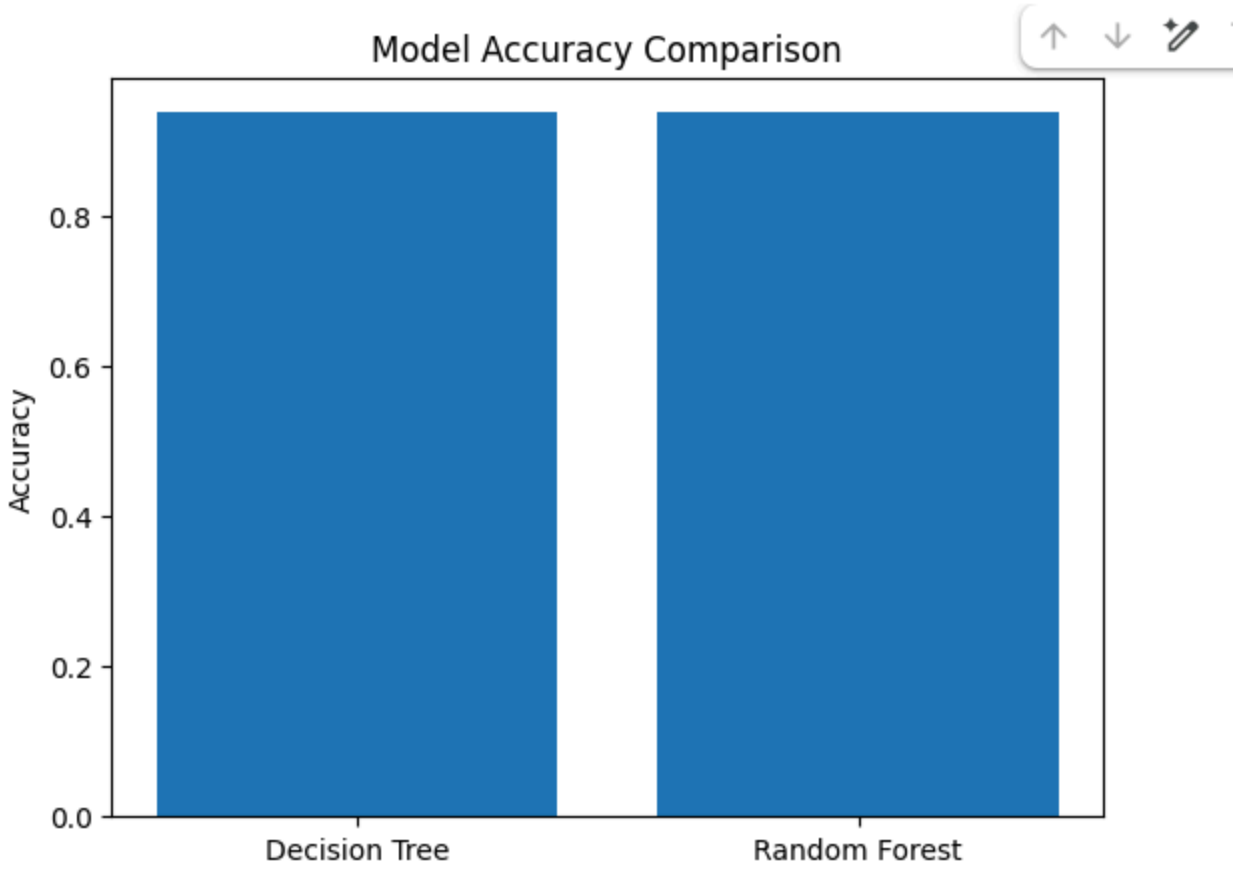
```
...        id  gender   age  hypertension  heart_disease ever_married  \
    0   9046    Male  67.0             0              1          Yes
    1  51676  Female  61.0             0              0          Yes
    2  31112    Male  80.0             0              1          Yes
    3  60182  Female  49.0             0              0          Yes
    4   1665  Female  79.0             1              0          Yes

          work_type Residence_type  avg_glucose_level   bmi   smoking_status  \
    0       Private          Urban             228.69  36.6  formerly smoked
    1  Self-employed          Rural             202.21   NaN     never smoked
    2       Private          Rural             105.92  32.5     never smoked
    3       Private          Urban             171.23  34.4           smokes
    4  Self-employed          Rural             174.12  24.0     never smoked


    stroke
0       1
1       1
2       1
3       1
4       1

After Encoding:
   gender   age  hypertension  heart_disease  ever_married  work_type  \
0       1  67.0             0              1             1          2
1       0  61.0             0              0             1          3
2       1  80.0             0              1             1          2
3       0  49.0             0              0             1          2
4       0  79.0             1              0             1          3
```

```
   Residence_type  avg_glucose_level        bmi  smoking_status  stroke
0               1             228.69  36.600000               1       1
1               0             202.21  28.893237               2       1
2               0             105.92  32.500000               2       1
3               1             171.23  34.400000               3       1
4               0             174.12  24.000000               2       1

===== DECISION TREE RESULTS =====
Accuracy: 0.9383561643835616

Confusion Matrix:
 [[956   4]
 [ 59   3]]


Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       960
           1       0.43      0.05      0.09        62

    accuracy                           0.94      1022
   macro avg       0.69      0.52      0.53      1022
weighted avg       0.91      0.94      0.91      1022

/tmp/ipython-input-4207125531.py:34: FutureWarning: A value is trying to be set on a cop
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  df["bmi"].fillna(df["bmi"].mean(), inplace=True)
```




```
===== RANDOM FOREST RESULTS =====
Accuracy: 0.9383561643835616

Confusion Matrix:
 [[959   1]
 [ 62   0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       960
           1       0.00      0.00      0.00        62

    accuracy                           0.94      1022
   macro avg       0.47      0.50      0.48      1022
weighted avg       0.88      0.94      0.91      1022
```


Random Forest Confusion Matrix
959
1
62
0
Actual
0
1
Predicted
0
1
800
600
400
200
0



CONCLUSION

In this experiment, Decision Tree and Random Forest algorithms were successfully implemented on a real-world healthcare dataset to predict stroke occurrence.

The Decision Tree model provided a simple and interpretable structure for classification but showed limitations such as overfitting and sensitivity to data variations.

The Random Forest model improved the overall performance by combining multiple decision trees using ensemble learning. It reduced overfitting and provided higher accuracy and better generalization compared to the Decision Tree model.

From the results, it can be concluded that Random Forest is more reliable and effective for classification tasks on real-world datasets due to its robustness and ability to handle complex data patterns.

Thus, ensemble methods like Random Forest are preferred over single models when accuracy and stability are important, especially in critical applications such as healthcare prediction systems.