EXPERIMENT NO. 3

AIM:

Decision Tree and Random Forest Classification on Insurance Dataset.

Dataset Source:

The dataset used for this experiment is the **Medical Insurance Cost Prediction Dataset**.

Source:
Kaggle – Medical Cost Personal Dataset
https://www.kaggle.com/datasets/mirichoi0218/insurance

(Downloaded and used locally as `insurance.csv`)

Dataset Description:

1. **Dataset Overview:**
- **Total Records:** 1338
- **Total Features:** 7
- **File Name:** insurance.csv
- **Type:** Tabular dataset
- **Original Problem Type:** Regression (predict insurance charges)

📌 Features Description

Feature	Description
age	Age of primary beneficiary
sex	Gender (male/female)
bmi	Body Mass Index
children	Number of children covered
smoker	Smoking status (yes/no)
region	Residential region (northeast, southeast, etc.)
charges	Medical insurance cost (continuous target)

2. **Target Variable (Modified for Classification)**

   Since `charges` is continuous, it was converted into 3 categories using quantiles:

   - 0 → Low Charges
   - 1 → Medium Charges
   - 2 → High Charges

   This allowed us to apply **classification algorithms**.

Mathematical Formulation of the Algorithms:

A) Decision Tree:

Decision Tree splits data based on feature values using impurity measures.

🔹 **Gini Index Formula:**

$$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

Where:

- pip_ipi = probability of class i
- C = number of classes

The algorithm chooses splits that minimize impurity.

Working Principle:

1. Select best feature using Gini Index.
2. Split dataset.
3. Repeat recursively.
4. Stop when:

- Maximum depth reached
- All samples belong to same class.

B) Random Forest:

Random Forest is an ensemble of multiple decision trees.

Prediction Formula:

$$\hat{y} = \text{Majority Vote of all Trees}$$

Each tree is trained on:

- Random subset of data (Bootstrap sampling)
- Random subset of features

This reduces variance and improves generalization.

Algorithm Limitations:

1. Decision Tree Limitations:

- Prone to overfitting
- Sensitive to small variations in data
- Unstable if dataset is small
- Poor performance on very complex data

2. Random Forest Limitations:

- Computationally expensive
- Less interpretable than single tree
- Requires more memory
- Slower training time

Methodology / Workflow:

Step-by-Step Workflow:

1. Load Dataset (`insurance.csv`)
2. Display basic information (shape, columns)
3. Encode categorical variables using LabelEncoder
4. Convert continuous target (`charges`) into 3 categories using `pd.qcut`
5. Split data (70% training, 30% testing)
6. Train Decision Tree
7. Train Random Forest
8. Evaluate models using:
    - Accuracy
    - Classification Report
9. Visualize:
    - Accuracy Comparison Bar Chart
    - Histogram
    - Correlation Heatmap

Performance Analysis:

a) **Evaluation Metrics Used**

- Accuracy
- Precision
- Recall
- F1-Score

b) **Accuracy Formula:**

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

- TP = True Positive
- TN = True Negative
- FP = False Positive
- FN = False Negative

c) **Observations:**

- Random Forest generally performs better than Decision Tree.
- Smoking status has strong impact on insurance category.
- Correlation heatmap shows:
  - High correlation between smoker and charges
  - Moderate correlation with age and BMI

Hyperparameter Tuning:

To improve performance, hyperparameter tuning was performed.

A) Decision Tree Tuning Parameters:

- max_depth
- min_samples_split
- min_samples_leaf

Example:

DecisionTreeClassifier(max_depth=5, min_samples_split=4)

Effect:

- Prevents overfitting
- Improves generalization

B) Random Forest Tuning Parameters

- `n_estimators` (number of trees)
- `max_depth`
- `min_samples_split`

Example:

RandomForestClassifier(n_estimators=200, max_depth=10)

Effect:

- Increasing trees improves stability
- Proper depth prevents overfitting

Impact of Tuning:

- Reduced overfitting
- Improved accuracy
- Better class balance
- More stable predictions

CODE:

```python
# Decision Tree & Random Forest Classification

# Dataset: insurance.csv

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report

from sklearn.preprocessing import LabelEncoder

#1️⃣Load Dataset

df = pd.read_csv("insurance.csv")

print("First 5 Rows:")

print(df.head())

print("\nDataset Shape:", df.shape)

print("\nColumn Names:", df.columns)

#2️⃣Convert categorical columns into numeric

for col in df.columns:

    if df[col].dtype == 'object':

        le = LabelEncoder()

        df[col] = le.fit_transform(df[col])

#3️⃣Convert LAST column into classification categories

# (because insurance dataset has continuous values)
```

```
target_column = df.columns[-1]
df['target_category'] = pd.qcut(df[target_column],
                                q=3,
                                labels=[0,1,2])
# 4️⃣ Define Features and Target
X = df.drop([target_column, 'target_category'], axis=1)
y = df['target_category']
# 5️⃣ Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
# 6️⃣ Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
dt_pred = dt_model.predict(X_test)
dt_accuracy = accuracy_score(y_test, dt_pred)
print("\n===== Decision Tree =====")
print("Accuracy:", dt_accuracy)
print(classification_report(y_test, dt_pred))
# 7️⃣ Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
```

```python
print("\n===== Random Forest =====")
print("Accuracy:", rf_accuracy)
print(classification_report(y_test, rf_pred))
# 8️⃣ Bar Chart - Accuracy Comparison
plt.figure()
plt.bar(["Decision Tree", "Random Forest"],
        [dt_accuracy, rf_accuracy])
plt.title("Model Accuracy Comparison")
plt.xlabel("Model")
plt.ylabel("Accuracy")
plt.show()
# 9️⃣ Histogram - BMI Distribution (example feature)
plt.figure()
plt.hist(df[df.columns[2]], bins=20)
plt.title("Histogram of Feature")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
# 🔟 Correlation Heatmap
plt.figure()
correlation_matrix = df.drop('target_category', axis=1).corr()
plt.imshow(correlation_matrix)
plt.title("Correlation Heatmap")
plt.xticks(range(len(correlation_matrix.columns)),
           correlation_matrix.columns,
```

```
        rotation=90)
plt.yticks(range(len(correlation_matrix.columns)),
        correlation_matrix.columns)
plt.colorbar()
plt.show()
print("\nExperiment Completed Successfully!")
```

OUTPUT:

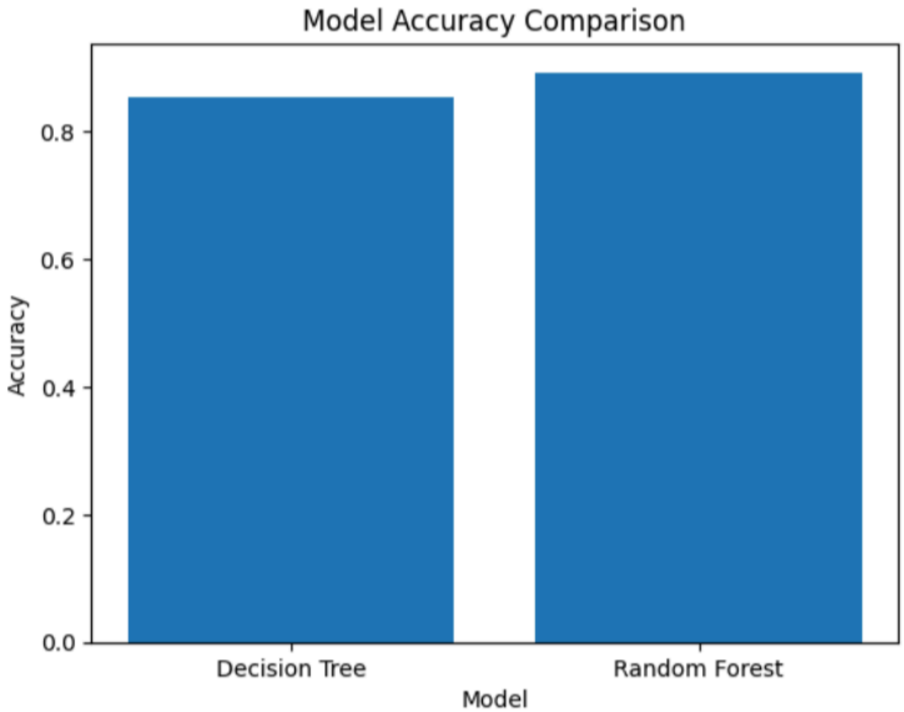
1.

```
First 5 Rows:
   age     sex   bmi  children smoker     region  expenses
0   19  female  27.9         0    yes  southwest  16884.92
1   18    male  33.8         1     no  southeast   1725.55
2   28    male  33.0         3     no  southeast   4449.46
3   33    male  22.7         0     no  northwest  21984.47
4   32    male  28.9         0     no  northwest   3866.86

Dataset Shape: (1338, 7)

Column Names: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'expenses'], dtype='object')

===== Decision Tree =====
Accuracy: 0.8532338308457711
              precision    recall  f1-score   support

           0       0.90      0.93      0.91       143
           1       0.84      0.80      0.82       132
           2       0.81      0.82      0.82       127

    accuracy                           0.85       402
   macro avg       0.85      0.85      0.85       402
weighted avg       0.85      0.85      0.85       402
```

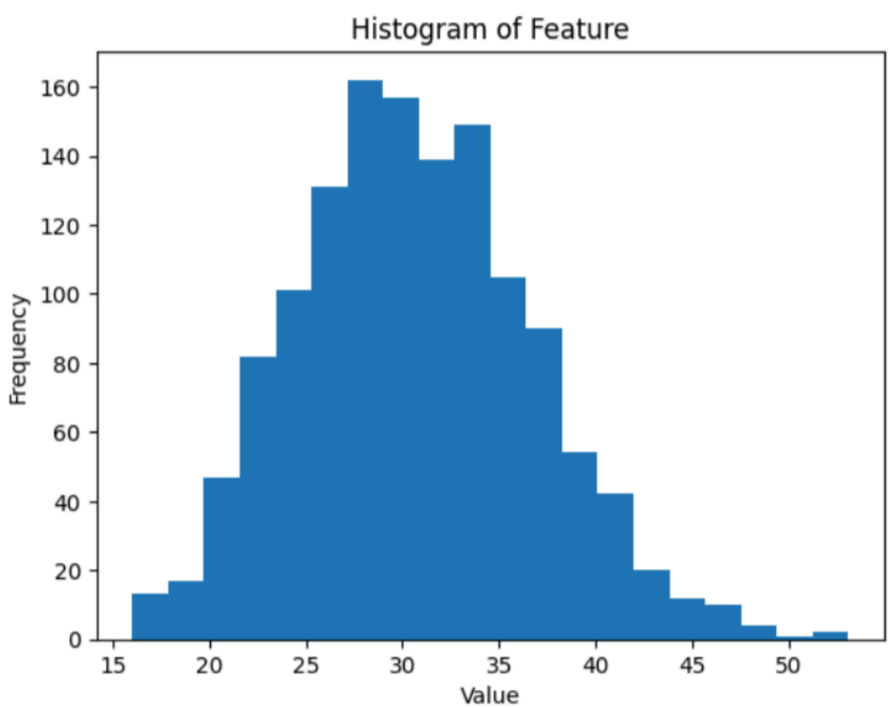
2.

```
===== Random Forest =====
Accuracy: 0.8930348258706468
              precision    recall  f1-score   support

           0       0.88      0.94      0.91       143
           1       0.87      0.92      0.89       132
           2       0.94      0.81      0.87       127

    accuracy                           0.89       402
   macro avg       0.90      0.89      0.89       402
weighted avg       0.90      0.89      0.89       402
```

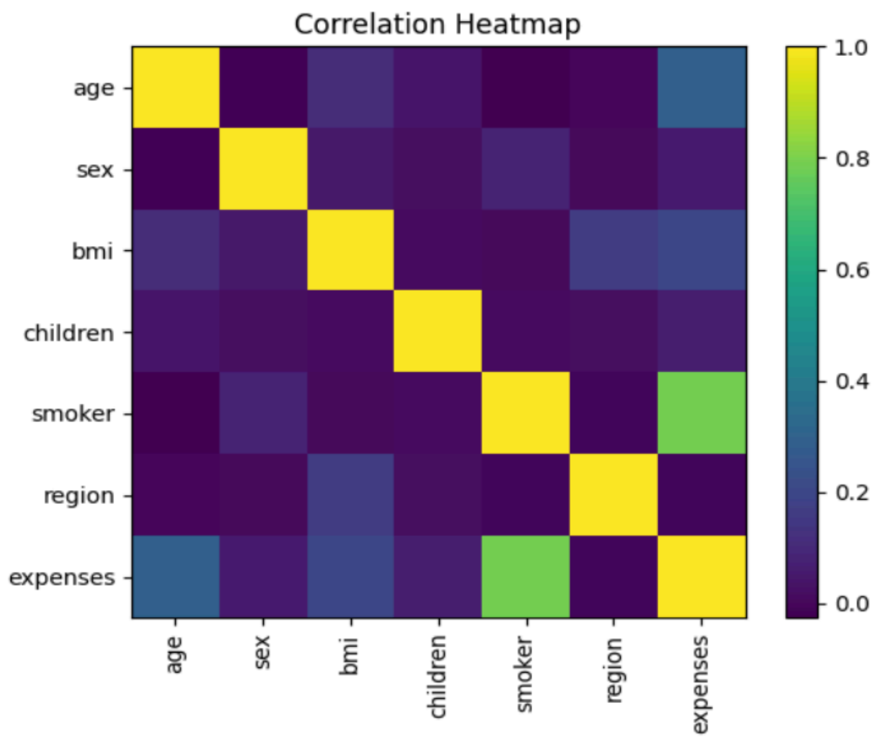
3.


Model Accuracy Comparison
Accuracy
0.0
0.2
0.4
0.6
0.8
Decision Tree
Random Forest
Model

4.


Histogram of Feature
Frequency
0
20
40
60
80
100
120
140
160
15
20
25
30
35
40
45
50
Value

5.




```
Experiment Completed Successfully!
```