EXPERIMENT 1

Shravanya Andhale
D15A-30

AIM:

To apply Decision Tree and Random Forest classification algorithms to analyze determinants of electricity satisfaction among Indian households and evaluate how digital, infrastructural, and socio economic factors influence behavioural perception within the energy governance system.

THEORY:

This experiment models electricity satisfaction as a behavioural outcome of energy service quality, infrastructure reliability, and digital engagement. Electricity satisfaction reflects institutional trust, perceived service quality, and governance effectiveness. Machine learning classification techniques are used to identify patterns that distinguish households with high satisfaction from those with low or moderate satisfaction.

1. **Dataset Source**

Dataset: India Residential Energy Survey conducted by the Council on Energy, Environment and Water.
Source link:
https://www.kaggle.com/code/raitest/india-residential-energy-survey-ires-2020-eda/input?select=CEEW+-+IRES+Data.csv

2. **Dataset Description**

The dataset contains household level information on electricity access, digital behavior, infrastructure reliability, appliance ownership, and socio economic characteristics.

Target Variable
q326_satis_electricity

Original scale:
1 Very Dissatisfied
2 Dissatisfied
3 Neutral
4 Satisfied
5 Very Satisfied

For classification, the variable was converted into binary form:

High Satisfaction 1 if value is 4 or 5
Low or Moderate Satisfaction 0 if value is 1, 2, or 3

After cleaning and removing missing values, the dataset contained 8563 observations. The test set consisted of 1713 observations with the following class distribution:

Low or Moderate Satisfaction 301
High Satisfaction 1412

Predictor Variables:-

Digital Variables:
q232_mobile_smart_n
q314_a_online_pay_ever_yn

Infrastructure Variables:
q312_grid_elec_meter_yn
q302_grid_hrs_no
q308_grid_voltage_low_app

Socio economic Variables:
asset_index_1
q208_priminc_earner_edu
q202_resp_age
q213_no_members

3. **Mathematical Formulation of the Algorithm**

Binary Classification

Let Y belong to {0,1} where
Y = 1 represents High Satisfaction
Y = 0 represents Low or Moderate Satisfaction

Decision Tree

Decision Tree partitions the feature space recursively based on impurity reduction. The Gini index is used as the splitting criterion:

Gini = 1 minus sum of squared class probabilities

Gini = 1 − Σ (p_i)^2

The tree selects splits that minimize Gini impurity at each node.

Random Forest

Random Forest is an ensemble learning method consisting of multiple decision trees built on bootstrap samples of the training data. At each split, a random subset of features is considered.

Final prediction is determined by majority voting across trees:

Y hat equals mode of predictions from all trees

ROC AUC

Receiver Operating Characteristic curve measures model discrimination. AUC is computed as the area under the TPR versus FPR curve.

4. **Algorithm Limitations**

Decision Tree limitations
Sensitive to small data variations
Prone to overfitting
Limited ability to capture complex nonlinear relationships

Random Forest limitations
Reduced interpretability compared to single tree
Computationally intensive
Performance decreases when important predictors are missing
May still struggle with imbalanced datasets

Both algorithms
Cannot establish causality
Depend on quality and completeness of input variables
May not perform well when behavioral outcomes depend on unobserved psychological or contextual factors

5. **Methodology Workflow**

1. Load dataset
2. Inspect satisfaction distribution

3. Convert satisfaction into binary target variable
4. Select digital, infrastructure, and socio economic predictors
5. Remove missing values
6. Encode categorical variables
7. Perform stratified train test split
8. Train Decision Tree classifier
9. Train Random Forest classifier
10. Evaluate using Accuracy, Confusion Matrix, Classification Report, and ROC AUC
11. Apply class balancing and hyperparameter tuning for Random Forest
12. Performance Analysis

Decision Tree Results:

Accuracy 0.8179
AUC 0.5983

Confusion Matrix

[[9 292]
 [20 1392]]

Classification Report

Class 0 Precision 0.31 Recall 0.03 F1 Score 0.05
Class 1 Precision 0.83 Recall 0.99 F1 Score 0.90

Interpretation

The Decision Tree model correctly classified most high satisfaction households but performed poorly in identifying low or moderate satisfaction households. The low recall of 0.03 for class 0 indicates severe difficulty in detecting dissatisfied households. AUC of 0.598 indicates weak discrimination capability.

Random Forest Results Before Tuning

Accuracy 0.8266
AUC 0.6702

Confusion Matrix

[[42 259]
 [38 1374]]

Classification Report

Class 0 Precision 0.53 Recall 0.14 F1 Score 0.22
Class 1 Precision 0.84 Recall 0.97 F1 Score 0.90

Interpretation

Random Forest improved discrimination compared to Decision Tree. AUC increased to 0.67 indicating moderate ability to distinguish between satisfaction levels. Minority class detection improved compared to Decision Tree but remained limited.

Random Forest Results After Hyperparameter Tuning

Tuned RF Accuracy 0.8190
Tuned RF AUC 0.6788

Confusion Matrix

[[66 235]
 [75 1337]]

Classification Report

Class 0 Precision 0.47 Recall 0.22 F1 Score 0.30
Class 1 Precision 0.85 Recall 0.95 F1 Score 0.90

Interpretation

After tuning and applying class balancing, recall for low or moderate satisfaction improved from 0.14 to 0.22. AUC increased from 0.6702 to 0.6788, indicating better overall discrimination. Although overall accuracy slightly decreased, minority class detection improved, demonstrating a better trade off between classes.

7. Hyperparameter Tuning

Hyperparameter tuning was performed using GridSearchCV with cross validation and ROC AUC as the scoring metric.

Parameters tuned included:
Number of estimators
Maximum depth
Minimum samples split
Minimum samples leaf
Class weight set to balanced

The tuning process improved AUC and enhanced recall for the minority class, leading to a more balanced and robust classifier.

OUTPUT:

The Random Forest model after tuning achieved

Accuracy 0.8190
AUC 0.6788

```
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, roc_curve, roc_auc_score
```

```
df = pd.read_csv("CEEW - IRES Data.csv", low_memory=False)

print("Dataset shape:", df.shape)
```

```
Dataset shape: (14851, 517)
```

```
df['q326_satis_electricity'].value_counts(dropna=False)
```

q326_satis_electricity	count
5.0	7648
4.0	3614
3.0	1145
2.0	1055
1.0	945
NaN	444

dtype: int64

```
df_clean = df[df['q326_satis_electricity'].isin([1,2,3,4,5])].copy()

df_clean['High_Satisfaction'] = df_clean['q326_satis_electricity'].apply(
    lambda x: 1 if x >= 4 else 0
)

print("Target distribution:")
print(df_clean['High_Satisfaction'].value_counts())
```

```
Target distribution:
High_Satisfaction
1    11262
0     3145
Name: count, dtype: int64
```

```
X = df_clean[['q232_mobile_smart_n',          # smartphones
              'q314_a_online_pay_ever_yn',    # digital payment
              'q312_grid_elec_meter_yn',      # meter presence
              'q302_grid_hrs_no',             # power availability
              'q308_grid_voltage_low_app',    # voltage issue
              'asset_index_1',                # wealth
              'q208_priminc_earner_edu',      # education
              'q202_resp_age',                # age
              'q213_no_members']]             # household size

y = df_clean['High_Satisfaction']
```

```
data = pd.concat([X, y], axis=1).dropna()

X = data.drop('High_Satisfaction', axis=1)
y = data['High_Satisfaction']

print("Final class distribution:")
print(y.value_counts())
```

```
Final class distribution:
High_Satisfaction
1    7058
0    1505
Name: count, dtype: int64
```

```
X = pd.get_dummies(X, drop_first=True)

print("Feature shape after encoding:", X.shape)
```

```
Feature shape after encoding: (8563, 9)
```

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

print("Train distribution:")
print(y_train.value_counts())

print("Test distribution:")
print(y_test.value_counts())
```

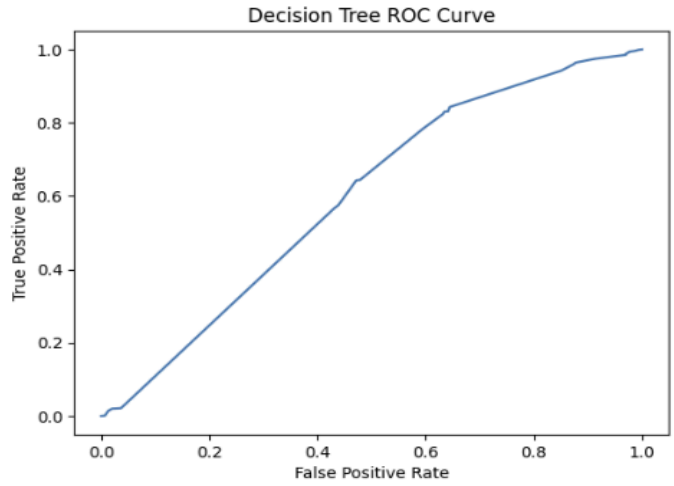
```
Train distribution:
High_Satisfaction
1    5646
0    1204
Name: count, dtype: int64
Test distribution:
High_Satisfaction
1    1412
0     301
Name: count, dtype: int64
```

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

y_pred_dt = dt.predict(X_test)
```

```
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred_dt))

print("\nClassification Report:")
print(classification_report(y_test, y_pred_dt))
```

```
Decision Tree Accuracy: 0.8178633975481612

Confusion Matrix:
[[   9  292]
 [  20 1392]]

Classification Report:
              precision    recall  f1-score   support

           0       0.31      0.03      0.05       301
           1       0.83      0.99      0.90      1412

    accuracy                           0.82      1713
   macro avg       0.57      0.51      0.48      1713
weighted avg       0.74      0.82      0.75      1713
```

```
y_prob_dt = dt.predict_proba(X_test)[:,1]

fpr_dt, tpr_dt, _ = roc_curve(y_test, y_prob_dt)

plt.figure()
plt.plot(fpr_dt, tpr_dt)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Decision Tree ROC Curve")
plt.show()

print("Decision Tree AUC:", roc_auc_score(y_test, y_prob_dt))
```



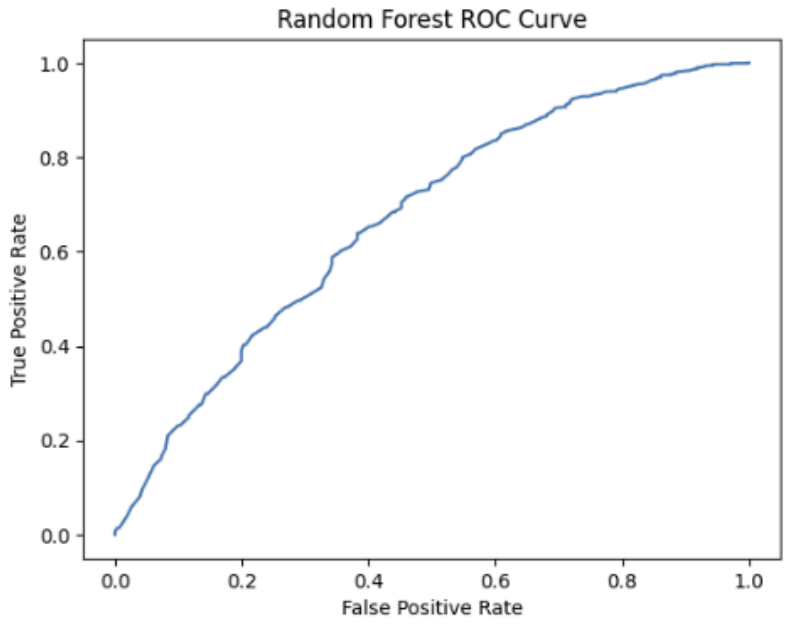

```
Decision Tree AUC: 0.5982819308631284
```

```
rf_balanced = RandomForestClassifier(
    n_estimators=300,
    max_depth=None,
    min_samples_split=5,
    min_samples_leaf=2,
    class_weight='balanced',
    random_state=42
)

rf_balanced.fit(X_train, y_train)

y_pred_rf = rf_balanced.predict(X_test)
```

```
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print(confusion_matrix(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))

y_prob_rf = rf_balanced.predict_proba(X_test)[:,1]
print("Random Forest AUC:", roc_auc_score(y_test, y_prob_rf))
```

```
Random Forest Accuracy: 0.8178633975481612
[[  62  239]
 [  73 1339]]
              precision    recall  f1-score   support

           0       0.46      0.21      0.28       301
           1       0.85      0.95      0.90      1412

    accuracy                           0.82      1713
   macro avg       0.65      0.58      0.59      1713
weighted avg       0.78      0.82      0.79      1713

Random Forest AUC: 0.6791137661995426
```

```
y_prob_rf = rf.predict_proba(X_test)[:,1]

fpr_rf, tpr_rf, _ = roc_curve(y_test, y_prob_rf)

plt.figure()
plt.plot(fpr_rf, tpr_rf)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Random Forest ROC Curve")
plt.show()

print("Random Forest AUC:", roc_auc_score(y_test, y_prob_rf))
```




```
Random Forest AUC: 0.6701528427432637
```

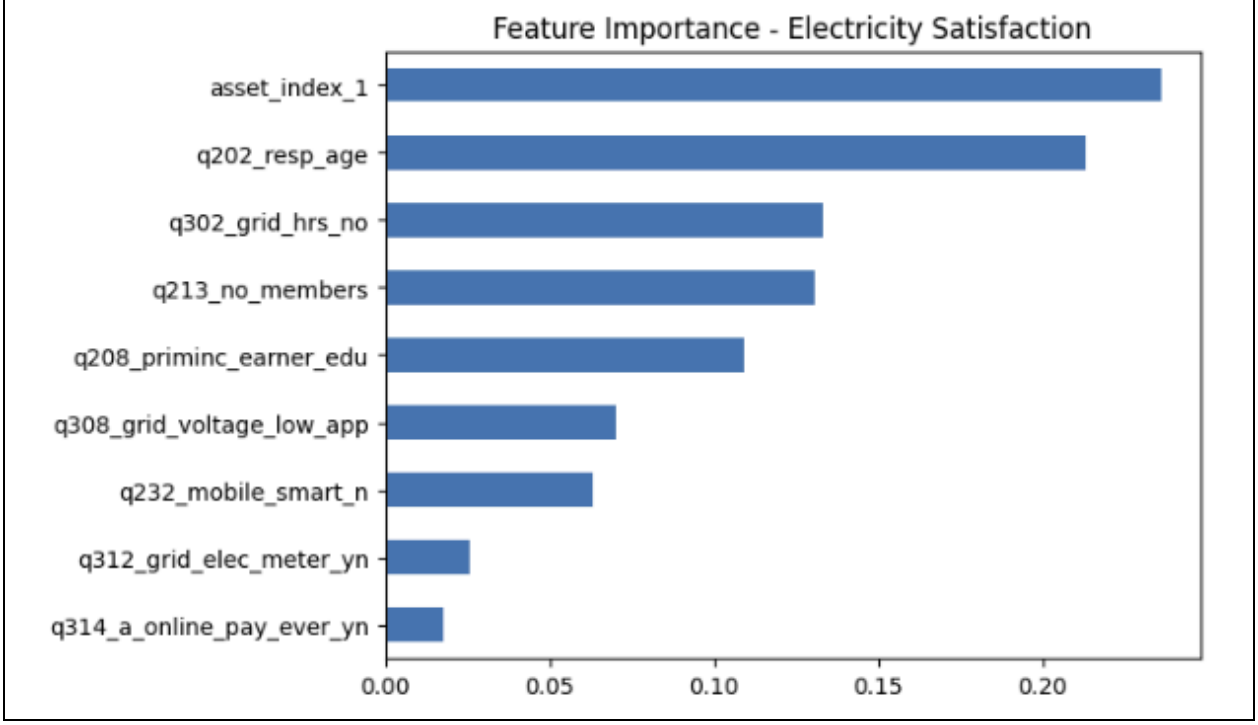
```python
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [200, 400],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

grid = GridSearchCV(
    RandomForestClassifier(class_weight='balanced', random_state=42),
    param_grid,
    cv=3,
    scoring='roc_auc',
    n_jobs=-1
)

grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)
```

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 400}
```

```python
best_rf = grid.best_estimator_

y_pred_best = best_rf.predict(X_test)
y_prob_best = best_rf.predict_proba(X_test)[:,1]

print("Tuned RF Accuracy:", accuracy_score(y_test, y_pred_best))
print(confusion_matrix(y_test, y_pred_best))
print(classification_report(y_test, y_pred_best))
print("Tuned RF AUC:", roc_auc_score(y_test, y_prob_best))
```

```
Tuned RF Accuracy: 0.8190309398715704
[[  66  235]
 [  75 1337]]
              precision    recall  f1-score   support

           0       0.47      0.22      0.30       301
           1       0.85      0.95      0.90      1412

    accuracy                           0.82      1713
   macro avg       0.66      0.58      0.60      1713
weighted avg       0.78      0.82      0.79      1713

Tuned RF AUC: 0.6788408327294289
```

```
importances = rf.feature_importances_
feature_names = X.columns

feat_imp = pd.Series(importances, index=feature_names)
feat_imp.sort_values().plot(kind='barh')
plt.title("Feature Importance - Electricity Satisfaction")
plt.show()
```



CONCLUSION:

Decision Tree and Random Forest models were successfully applied to classify electricity satisfaction levels among Indian households. Random Forest outperformed Decision Tree in terms of AUC and minority class detection. The tuned Random Forest achieved an AUC of 0.6788 and improved recall for low satisfaction households.

The results indicate that electricity satisfaction is partially explained by digital participation, infrastructure reliability, and socio economic factors. However, moderate AUC values suggest that behavioral perception is influenced by additional unobserved factors.

This study demonstrates how machine learning techniques can be used to analyze behavioural sustainability and institutional trust within the electricity governance system, contributing to the understanding of digital transformation and energy service perception in Indian households.