Code with Execution Procedure

Import Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.tree import export_text
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
```

- These lines import necessary libraries and modules for data manipulation, machine learning modeling, and evaluation.

Read Dataset

```
data = pd.read_csv("C:\\jupyter-notebook\\cocomo81.csv")
```

- Reads the CSV file named "cocomo81.csv" into a pandas DataFrame called `data`.

Obtain No. of rows and columns

```
num_rows, num_columns = data.shape
print("Number of rows:", num_rows)
print("Number of columns:", num_columns)
print("Column names:", data.columns)
```

- Obtains the number of rows and columns in the dataset and prints them, along with the column names.

Apply Normalization

scaler = MinMaxScaler()

data_scaled = scaler.fit_transform(data)

data_scaled_df = pd.DataFrame(data_scaled, columns=data.columns)

- Initializes a MinMaxScaler object to scale the data between 0 and 1. Fits the scaler to the data and transforms it. Finally, creates a DataFrame `data_scaled_df` with the scaled data.

Drop Effort

data_scaled_df.columns = data_scaled_df.columns.str.strip()

- Removes leading and trailing whitespaces from the column names.

target_column = 'Effort1'

- Defines the target column name.

X = data_scaled_df.drop(columns=['Effort1'])

Y = data_scaled_df['Effort1']

Split the Data

- Splits the DataFrame into features (X) and the target variable (Y).

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

- Splits the data into training and testing sets with 80% for training and 20% for testing, using a random state for reproducibility.

Initialize Decision Tree Regressor

dt_model = DecisionTreeRegressor(random_state=42)

dt_model.fit(X_train, Y_train)

Y_pred = dt_model.predict(X_test)

- Initializes a Decision Tree Regressor model, fits it to the training data, and predicts the target variable for the test data.

tree_regressor = DecisionTreeRegressor(max_depth=20, min_samples_split=2, random_state=42)

tree_regressor.fit(X_train, Y_train)

Y_pred = tree_regressor.predict(X_test)

Calculate MSE & RMSE for Decision Tree

mse = np.mean((Y_test - Y_pred)**2)

rmse = np.sqrt(np.mean((Y_test - Y_pred)**2))

print(f"MSE: {mse}")

print(f"RMSE: {rmse}")

- Calculates mean squared error (MSE) and root mean squared error (RMSE) between the predicted and actual values, then prints them.

Initialize AdaBoost Technique

base_regressor = DecisionTreeRegressor(max_depth=5, random_state=42)

adaboost_regressor = AdaBoostRegressor(base_regressor, n_estimators=1, random_state=42)

Y_train = Y_train.ravel()

adaboost_regressor.fit(X_train, Y_train)

y_pred_boosted = adaboost_regressor.predict(X_test)

- Initializes a base Decision Tree Regressor and an AdaBoost Regressor using it. Fits the AdaBoost model to the training data and predicts the target variable for the test data.

Calculate MSE & RMSE for AdaBoost

mse = np.mean((Y_test - y_pred_boosted)**2)

rmse = np.sqrt(np.mean((Y_test - y_pred_boosted)**2))

print(f"MSE: {mse}")

print(f"RMSE: {rmse}")

- Calculates MSE and RMSE between the predicted and actual values for the AdaBoost model, then prints them.

Initialize Pruned Decision Tree

pruned_dt_regressor = DecisionTreeRegressor(max_depth=2 ,random_state=42)

pruned_dt_regressor.fit(X_train, Y_train)

Y_pred_pruned = pruned_dt_regressor.predict(X_test)

Initializes and fits a pruned Decision Tree Regressor with specified hyperparameters to the training data and predicts the target variable for the test data.

Calculate MSE & RMSE for Pruned Decision Tree

```
mse = np.mean((Y_test - Y_pred_pruned)**2)
rmse = np.sqrt(np.mean((Y_test - Y_pred_pruned)**2))
 print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
```

Calculates MSE and RMSE between the predicted and actual values for the pruned Decision Tree model, then prints them.

The provided code performs regression analysis on software effort estimation data using decision tree-based models. It first preprocesses the data, scales it, and splits it into training and testing sets. Three decision tree regressors are trained: one with default parameters, one with boosted AdaBoost, and one pruned for simplicity. The code evaluates each model's performance using Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) metrics. Finally, it prints the MSE and RMSE values for each model. Overall, the code demonstrates the application of decision tree regressors and their variants in software effort estimation, comparing their performance.

ocalhost:8888/notebooks/Majorimplementation.ipynb

File Edit View Run Kernel Settings Help

Code

[2]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.tree import export_text
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
```

[3]:
```
data = pd.read_csv("C:\\jupyter-notebook\\cocomo81.csv")
num_rows, num_columns = data.shape
print("Number of rows:", num_rows)
print("Number of columns:", num_columns)
print("Column names:", data. columns)
```

```
Number of rows: 63
Number of columns: 17
Column names: Index(['Rely', 'Data', 'Cplx', 'Time', 'Stor', 'Virt', 'Turn', 'Acap', 'Aexp',
       'Pcap', 'Vexp', 'Lexp', 'Modp', 'Tool', 'Sced', 'Size', 'Effort1'],
      dtype='object')
```

[4]:
```
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
data_scaled_df = pd.DataFrame(data_scaled, columns=data.columns)
print(data_scaled_df)
data_scaled_df.columns = data_scaled_df.columns.str.strip()  # Corrected the variable name
target_column = 'Effort1'
```

```
       Rely  Data      Cplx      Time      Stor      Virt      Turn  \
0  0.200000   1.0  0.000000  0.000000  0.107143  0.651163  0.714286
1  0.200000   1.0  0.157895  0.000000  0.107143  0.302326  0.714286
2  0.384615   1.0  0.157895  0.000000  0.000000  0.000000  0.250000
3  0.000000   1.0  0.000000  0.000000  0.000000  0.000000  0.464286
4  0.200000   0.0  0.315789  0.000000  0.000000  0.000000  0.464286
```

```
4   0.200000   0.0  0.315789  0.000000  0.000000  0.000000  0.464286
..       ...   ...       ...       ...       ...       ...       ...
58  0.384615   0.0  0.473684  0.090909  0.107143  0.302326  0.000000
59  0.615385   0.0  0.631579  0.166667  0.107143  0.302326  0.464286
60  0.384615   0.0  0.473684  0.000000  0.000000  0.000000  0.000000
61  0.200000   0.0  0.631579  0.166667  0.375000  0.651163  0.464286
62  0.384615   0.0  0.473684  0.000000  0.000000  0.302326  0.000000

        Acap      Aexp      Pcap      Vexp      Lexp      Modp      Tool  \
0   0.640000  0.659574  0.652778  0.645161  0.263158  1.000000  0.658537
1   0.386667  0.191489  0.416667  0.000000  0.000000  0.666667  0.414634
2   0.200000  0.000000  0.222222  0.000000  0.000000  0.214286  0.195122
3   0.640000  0.191489  1.000000  0.322581  0.000000  1.000000  0.414634
4   0.386667  0.382979  0.222222  0.000000  0.000000  1.000000  0.414634
..       ...       ...       ...       ...       ...       ...       ...
58  0.386667  0.382979  0.416667  0.322581  0.263158  0.214286  0.414634
59  0.200000  0.659574  0.222222  0.645161  0.631579  0.666667  0.658537
60  0.200000  0.382979  0.222222  0.000000  0.263158  0.000000  0.414634
61  0.093333  0.000000  0.000000  1.000000  1.000000  0.214286  1.000000
62  0.000000  0.000000  0.222222  0.322581  0.263158  0.000000  0.414634

        Sced      Size   Effort1
0   0.173913  0.096706  0.178522
1   0.000000  0.253497  0.139906
2   0.000000  0.113256  0.020809
3   0.173913  0.050539  0.020546
4   0.000000  0.012212  0.002378
..       ...       ...       ...
58  0.000000  0.018310  0.005626
59  0.347826  0.004111  0.004485
60  0.000000  0.022665  0.003870
61  0.000000  0.006202  0.002817
62  0.000000  0.006986  0.000799

[63 rows x 17 columns]
```

[11]:

```
X = data_scaled_df.drop(columns=['Effort1'])
Y= data_scaled_df['Effort1']
#print(Y)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
0     0.178522
```

```
0     0.178522
1     0.139906
2     0.020809
3     0.020546
4     0.002378
        ...
58    0.005626
59    0.004485
60    0.003870
61    0.002817
62    0.000799
Name: Effort1, Length: 63, dtype: float64
```

[13]:
```
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, Y_train)
Y_pred = dt_model.predict(X_test)
#print(Y_pred)
```

[14]:
```
tree_regressor = DecisionTreeRegressor(max_depth=20, min_samples_split=2, random_state=42)
tree_regressor.fit(X_train, Y_train)
```

[14]:
```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=20, random_state=42)
```

[15]:
```
Y_pred = tree_regressor.predict(X_test)
print(Y_pred)
print(Y_test)
```

```
[0.00071089 0.05257984 0.02054572 0.04678737 0.00430925 0.01054054
 0.00299278 0.00676666 0.04678737 0.00562572 0.01492878 0.00264172
 0.00018431]
61    0.002817
57    0.010892
0     0.178522
43    0.007118
5     0.003256
36    0.003607
16    0.000272
```

```
60    0.003870
55    0.083561
9     0.027655
40    0.000009
Name: Effort1, dtype: float64
```

[18]:
```python
mse = np.mean((Y_test - Y_pred)**2)
rmse = np.sqrt(np.mean((Y_test - Y_pred)**2))
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
```

```
MSE: 0.0026035678653477047
RMSE: 0.05102516893992321
```

[21]:
```python
base_regressor = DecisionTreeRegressor(max_depth=5, random_state=42)
adaboost_regressor = AdaBoostRegressor(base_regressor, n_estimators=1, random_state=42)
Y_train = Y_train.ravel()
adaboost_regressor.fit(X_train, Y_train)
y_pred_boosted = adaboost_regressor.predict(X_test)
#print(y_pred_boosted)
```

[12]:
```python
mse = np.mean((Y_test - y_pred_boosted)**2)
rmse = np.sqrt(np.mean((Y_test - y_pred_boosted)**2))
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
```

```
MSE: 0.0018943267009220099
RMSE: 0.04352386357990304
```

[13]:
```python
pruned_dt_regressor = DecisionTreeRegressor(max_depth=2 ,random_state=42)
pruned_dt_regressor.fit(X_train, Y_train)
Y_pred_pruned = pruned_dt_regressor.predict(X_test)
```

[12]:
```python
mse = np.mean((Y_test - Y_pred_pruned)**2)
rmse = np.sqrt(np.mean((Y_test - Y_pred_pruned)**2))
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
```

```
MSE: 0.002504062444789234
RMSE: 0.05004060795782995
```