import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from sklearn.metrics import r2\_score,mean\_squared\_error,mean\_absolute\_error,accuracy\_score

import pickle

data=pd.read\_csv('evdataset.csv')

data.head()

data.columns

data.info()

data.shape

data.isnull().sum()

data.dtypes

data.nunique()

#Finding correlation

corr\_matrix = data.corr()

corr\_matrix["Electric Range"].sort\_values(ascending=False)

#Checking for null values

data.nunique()

data.shape

#Converting categorical data into numerical data

data.replace({'Drive':{'Rear':2,'Front':0,'AWD':1}},inplace=True)

#Dropping few features

cols\_to\_use=['City - Cold Weather','Highway - Cold Weather','Combined - Cold Weather','City - Mild Weather','Highway - Mild Weather','Combined - Mild Weather','Acceleration 0 - 100 km/h','Top Speed','Electric Range','Total Power','Total Torque','Drive','Battery Capacity','Charge Power','Charge Speed','Fastcharge Speed','Gross Vehicle Weight (GVWR)','Max. Payload','Cargo Volume','Width','Length']

data=data[cols\_to\_use]

data.head()

data.isna().sum()

x=data.drop(['Electric Range','City - Cold Weather','Highway - Cold Weather','Combined - Cold Weather','City - Mild Weather','Highway - Mild Weather',

'Combined - Mild Weather'],axis=1)

y=data[['Electric Range','City - Cold Weather','Highway - Cold Weather','Combined - Cold Weather','City - Mild Weather','Highway - Mild Weather',

'Combined - Mild Weather']]

x.shape

#Normalizing the given data

from sklearn import preprocessing

x=preprocessing.normalize(x)

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=1)

#Using Linear Regression

from sklearn.linear\_model import LinearRegression

reg=LinearRegression().fit(X\_train,Y\_train)

#Model Evaluation (Regression Metrics)

# Calculate the score for Training Data

reg.score(X\_train, Y\_train)

print("R2 for Training Data: ", reg.score(X\_train, Y\_train))

reg.score(X\_test, Y\_test)

print("R2 for testing: ",reg.score(X\_test,Y\_test))

import matplotlib.pyplot as plt

import numpy as np

from sklearn.linear\_model import LinearRegression

# Assuming reg is your trained linear regression model

# Fit the linear regression model

reg = LinearRegression().fit(X\_train, Y\_train)

# Predictions on the training and testing sets

y\_train\_pred = reg.predict(X\_train)

y\_test\_pred = reg.predict(X\_test)

# Scatter plot for Training Data

plt.scatter(Y\_train, y\_train\_pred)

plt.title('Linear Regression: Actual vs. Predicted (Training Data)')

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.show()

# Scatter plot for Testing Data

plt.scatter(Y\_test, y\_test\_pred)

plt.title('Linear Regression: Actual vs. Predicted (Testing Data)')

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.show()

# Scatter plot for Training Data with Regression Line

plt.scatter(X\_train, Y\_train, color='blue')

plt.plot(X\_train, y\_train\_pred, color='red', linewidth=2)

plt.title('Linear Regression: Actual vs. Predicted (Training Data)')

plt.xlabel('X Values')

plt.ylabel('Y Values')

plt.show()

# Assuming X\_train and Y\_train are your training data

from sklearn.linear\_model import LinearRegression

# Create and fit the linear regression model

reg = LinearRegression().fit(X\_train, Y\_train)

# Get the coefficients

coefficients = reg.coef\_

# Print the coefficients

print("Coefficients:", coefficients[0])

pickle.dump(reg, open('model.pkl','wb'))

testing\_data\_prediction=reg.predict(X\_test)

plt.scatter(Y\_test,testing\_data\_prediction)

plt.xlabel("Actual Range")

plt.ylabel("Predicted Range")

plt.title("Actual range vs Predicted range(Testing)")

plt.show()

mae = mean\_absolute\_error(Y\_test,testing\_data\_prediction)

from sklearn.metrics import mean\_squared\_error

# Assuming Y\_test and testing\_data\_prediction are your actual and predicted values

mse = mean\_squared\_error(Y\_test, testing\_data\_prediction)

print("Mean Squared Error:", mse)

from sklearn.metrics import mean\_squared\_error

import math

# Assuming Y\_test and testing\_data\_prediction are your actual and predicted values

mse = mean\_squared\_error(Y\_test, testing\_data\_prediction)

rmse = math.sqrt(mse)

print("Root Mean Squared Error:", rmse)

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import RandomizedSearchCV

from scipy.stats import randint

param\_distribs={

'n\_estimators':randint(low=1,high=200),

'max\_features':randint(low=1,high=15),

}

forest\_reg=RandomForestRegressor(random\_state=42)

rnd\_search=RandomizedSearchCV(forest\_reg,param\_distributions=param\_distribs,n\_iter=10,cv=5,

scoring="neg\_mean\_squared\_error", random\_state=42)

rnd\_search.fit(X\_train,Y\_train)

testing\_data\_prediction1=rnd\_search.predict(X\_test)

plt.scatter(Y\_test,testing\_data\_prediction1)

plt.xlabel("Actual Range")

plt.ylabel("Predicted Range")

plt.title("Actual range vs Predicted range(Testing)")

plt.show()

mae1 = mean\_absolute\_error(Y\_test,testing\_data\_prediction1)

mae1

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

param\_grid=[

{'n\_estimators':[3,10,30],'max\_features':[2,4,6,8]},

{'bootstrap':[False],'n\_estimators':[3,10],'max\_features':[2,3,4]},

]

rfc=RandomForestClassifier()

grid\_search=GridSearchCV(rfc,param\_grid,cv=5,scoring="neg\_mean\_squared\_error", return\_train\_score=True)

grid\_search.fit(X\_train,Y\_train)

rfc\_clf=grid\_search.best\_estimator\_

rfc\_clf\_predictions=rfc\_clf.predict(X\_test)

testing\_data\_prediction2=rfc\_clf.predict(X\_test)

plt.scatter(Y\_test,testing\_data\_prediction2)

plt.xlabel("Actual Range")

plt.ylabel("Predicted Range")

plt.title("Actual range vs Predicted range(Testing)")

plt.show()

mae2 = mean\_absolute\_error(Y\_test,rfc\_clf\_predictions)

from sklearn.tree import DecisionTreeRegressor

tree\_reg=DecisionTreeRegressor(random\_state=42)

tree\_reg.fit(X\_train,Y\_train)

from sklearn.metrics import r2\_score

# Assuming tree\_reg is your trained Decision Tree Regressor

# Training set

y\_train\_pred = tree\_reg.predict(X\_train)

r2\_train = r2\_score(Y\_train, y\_train\_pred)

print("R2 Score for Training Data:", r2\_train)

# Testing set

y\_test\_pred = tree\_reg.predict(X\_test)

r2\_test = r2\_score(Y\_test, y\_test\_pred)

print("R2 Score for Testing Data:", r2\_test)

from sklearn.metrics import r2\_score, mean\_squared\_error

import numpy as np

# Assuming tree\_reg is your trained Decision Tree Regressor

y\_pred = tree\_reg.predict(X\_test)

# Mean Squared Error (MSE)

mse = mean\_squared\_error(Y\_test, y\_pred)

print("Mean Squared Error:", mse)

# Root Mean Squared Error (RMSE)

rmse = np.sqrt(mse)

print("Root Mean Squared Error:", rmse)

testing\_data\_prediction3=tree\_reg.predict(X\_test)

mae3 = mean\_absolute\_error(Y\_test,testing\_data\_prediction3)

mae3

from sklearn.ensemble import RandomForestRegressor

forest\_reg=RandomForestRegressor(n\_estimators=100,random\_state=42)

forest\_reg.fit(X\_train,Y\_train)

testing\_data\_prediction4=forest\_reg.predict(X\_test)

mae4 = mean\_absolute\_error(Y\_test,testing\_data\_prediction4)

from sklearn.metrics import r2\_score

# Assuming forest\_reg is your trained RandomForestRegressor

# Training set

y\_train\_pred = forest\_reg.predict(X\_train)

r2\_train = r2\_score(Y\_train, y\_train\_pred)

print("R2 Score for Training Data:", r2\_train)

# Testing set

y\_test\_pred = forest\_reg.predict(X\_test)

r2\_test = r2\_score(Y\_test, y\_test\_pred)

print("R2 Score for Testing Data:", r2\_test)

from sklearn.metrics import mean\_squared\_error

import numpy as np

# Assuming forest\_reg is your trained RandomForestRegressor

# Testing set

y\_test\_pred = forest\_reg.predict(X\_test)

mse\_test = mean\_squared\_error(Y\_test, y\_test\_pred)

rmse\_test = np.sqrt(mse\_test)

print("Mean Squared Error (MSE) for Testing Data:", mse\_test)

print("Root Mean Squared Error (RMSE) for Testing Data:", rmse\_test)

Explanation:  
  
This Jupyter Notebook file appears to be a data analysis and machine learning pipeline for electric vehicle (EV) data.

1. Importing Libraries and Data: This section imports necessary libraries such as NumPy, Pandas, Matplotlib, Seaborn, and scikit-learn. It loads the dataset ‘evdataset.csv’ using Pandas and displays some basic information about the dataset such as columns, data types, shape, and null values.

2. Data Preprocessing: It involves data cleaning and preprocessing steps. It checks for null values in the dataset, replaces categorical values with numerical values using ordinal encoding, and drops some unnecessary features.

3. Normalization: It normalizes the data using the ‘preprocessing.normalize()’ function from scikit-learn.

4. Splitting Data: It splits the data into training and testing sets using the ‘train\_test\_split()’ function from scikit-learn.

5. Linear Regression Model: It trains a linear regression model using the training data and evaluates its performance using R-squared score on both training and testing data.

6. Visualization of Model Performance: It visualizes the actual vs. predicted values of the target variable (electric vehicle range) using scatter plots.

7. Model Evaluation: It calculates mean absolute error (MAE) and mean squared error (MSE) for the linear regression model.

8. Random Forest Regression Model: It trains a random forest regression model using the training data and evaluates its performance using R-squared score on both training and testing data. It also visualizes the actual vs. predicted values of the target variable.

9. Grid Search for Hyperparameter Tuning: It performs a grid search for hyperparameter tuning on a random forest classifier.

10. Decision Tree Regression Model: It trains a decision tree regression model and evaluates its performance using R-squared score and mean absolute error.

11. Comparison of Models: It compares the performance of different models (linear regression, random forest regression, decision tree regression) based on R-squared score, MAE, and MSE.

Overall, this notebook performs data analysis, data preprocessing, and model training and evaluation for predicting electric vehicle range using various machine learning algorithms such as linear regression, decision tree regression, and random forest regression. It also performs hyperparameter tuning using grid search and compares the performance of different models.