```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score


data = pd.read_csv('Bank_Stock_Price_10Y.csv')


data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	2483.000000	2483.000000	2483.000000	2483.000000	2483.000000	2.483000e+03
mean	5219.973822	5265.847765	5173.628675	5219.887233	4886.148684	7.997496e+07
std	2223.156537	2240.113146	2206.459905	2223.903144	2276.934419	5.378122e+07
min	1970.000000	1980.000000	1940.000000	1965.000000	1691.382568	0.000000e+00
25%	2955.000000	2985.000000	2930.000000	2950.000000	2612.564454	5.153575e+07
50%	5170.000000	5235.000000	5120.000000	5180.000000	4736.543945	7.009800e+07
75%	6822.500000	6890.000000	6740.000000	6800.000000	6349.964111	9.651755e+07
max	9775.000000	9775.000000	9675.000000	9750.000000	9750.000000	1.062862e+09

```
data.isnull().sum()
data = data.dropna()


scaler = StandardScaler()
data[['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']] = scaler.fit_transform(data[['Open', 'High', 'Low', 'Close', 'Adj Close',


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


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


model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

```
y_pred = model.predict(X_test)


mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R-squared:', r2)
```

```
Mean Squared Error: 0.0002560900975424451
R-squared: 0.9997354910385263
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Load the data
data = pd.read_csv('/content/Bank_Stock_Price_10Y.csv')

# Convert the 'Date' column to Unix timestamps
data['Date'] = pd.to_datetime(data['Date']).astype(int)/10**9

# Split the data into features (X) and target (y)
X = data.drop(['Close'], axis=1)
y = data['Close']

# Define the number of models to train and the number of rows to select for each model
num_models = 10
num_rows_per_model = 50

# Initialize lists to store the model parameters, accuracy, and evaluation metrics
model_params = []
model_accuracies = []
model_metrics = []

# Train the models
for i in range(num_models):
    # Select a random subset of rows from the data with replacement
    random_rows = np.random.choice(len(X), num_rows_per_model, replace=True)
    X_train, X_test, y_train, y_test = train_test_split(X.iloc[random_rows], y.iloc[random_rows], test_size=0.2, random_state=i)

    # Train the decision tree model
    model = DecisionTreeRegressor(random_state=i)
    model.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = model.predict(X_test)

    # Calculate the accuracy and evaluation metrics
    accuracy = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)

    # Save the model parameters, accuracy, and evaluation metrics
    model_params.append(model.get_params())
    model_accuracies.append(accuracy)
    model_metrics.append({'MSE': mse})

    # Print the output of the current decision tree model
    print(f'Model {i+1}:')
    print(f'Accuracy: {accuracy}')
    print(f'MSE: {mse}')
```

```
Model 1:
Accuracy: 0.9939457416898689
MSE: 17332.5
Model 2:
Accuracy: 0.9811187130521283
MSE: 70077.5
Model 3:
Accuracy: 0.9922720347572718
MSE: 47645.0
Model 4:
Accuracy: 0.995859774317256
MSE: 27767.5
Model 5:
Accuracy: 0.9924102556535757
MSE: 20575.0
Model 6:
Accuracy: 0.995623249068554
MSE: 12430.0
Model 7:
Accuracy: 0.9987027322299215
MSE: 7685.0
Model 8:
Accuracy: 0.9948461888282342
MSE: 23935.0
Model 9:
Accuracy: 0.992332996655365
MSE: 47057.5
Model 10:
Accuracy: 0.9957510926647993
MSE: 13697.5
```