Predicting Future Euro Exchange Rates Using a Random Forest Regressor

CPSC 4383: ARTIFICIAL INTELLIGENCE
Aleena Zainab

Objective

Goal: To predict future Euro exchange rates using historical data and a Random Forest Regressor.

Deliverables:

1. EDA and Preprocessing: Show the handling of missing data, feature selection, and creation of lagged variables.
2. Model Training and Evaluation: Present the results of training the Random Forest Regressor and provide evaluation metrics (MAE, MSE, R^2).
3. Visualization: Plot of actual vs predicted Euro exchange rates.

Data and Tools Used

Dataset: Historical Euro exchange rate data.

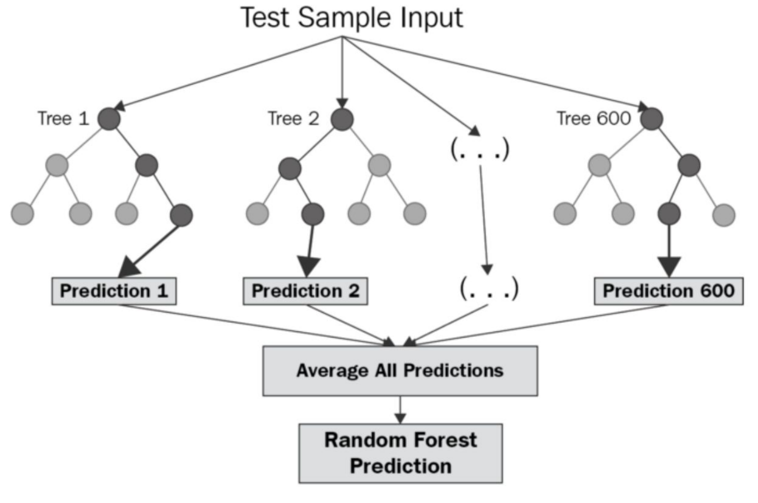
Tools:

- Python: Language used.
- Libraries: Pandas, NumPy, Scikit-Learn, Matplotlib
- Data Points: Daily exchange rates, used to predict the next day’s rate.

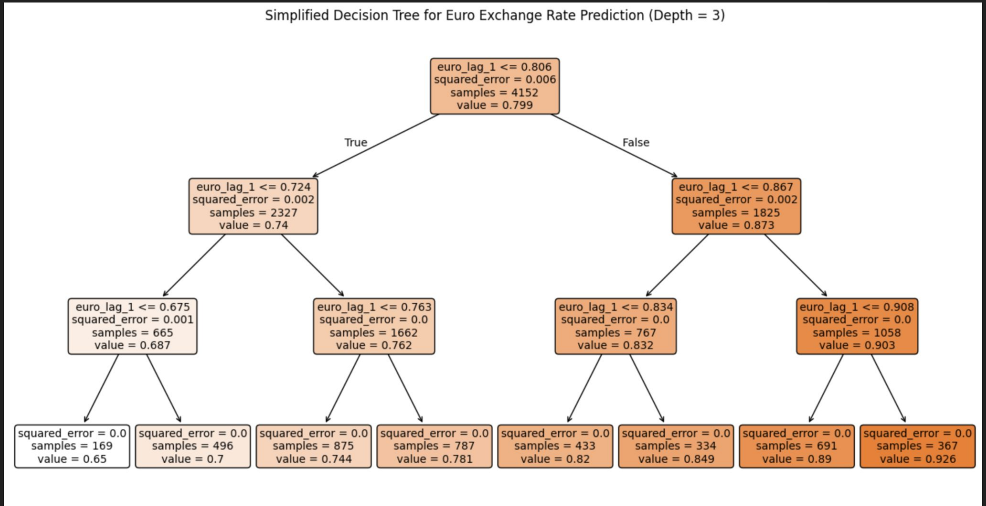
What is Random Forest Regressor?

- Ensemble Learning: Combines multiple decision trees to improve accuracy and reduce errors.
- Decision Trees: Each tree in the forest learns from different subsets of the data.
- Averaging Predictions: By averaging outputs from all trees, Random Forest reduces variance and enhances stability.
- Suitable for predicting trends in time series data, like exchange rates.

What is Random Forest Regressor?



https://www.keboola.com/blog/random-forest-regression



Setting Up the Environment and Loading Data

Code Overview: This code sets up the environment, mounts Google Drive, and loads the dataset for analysis.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import PartialDependenceDisplay
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

[3]
```
data = pd.read_csv('/content/drive/My Drive/exchange_rates.csv')

print("Initial Data Overview:")
print(data.head())
print(data.info())
```

```
Data columns (total 52 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         5220 non-null   object
 1   chinese_yuan       4845 non-null   float64
 2   euro               5152 non-null   float64
 3   japanese_yen       4935 non-null   float64
 4   uk_pound           5126 non-null   float64
 5   us_dollar          5220 non-null   float64
 6   algerian_dinar     3328 non-null   float64
 7   australian_dollar  4989 non-null   float64
```

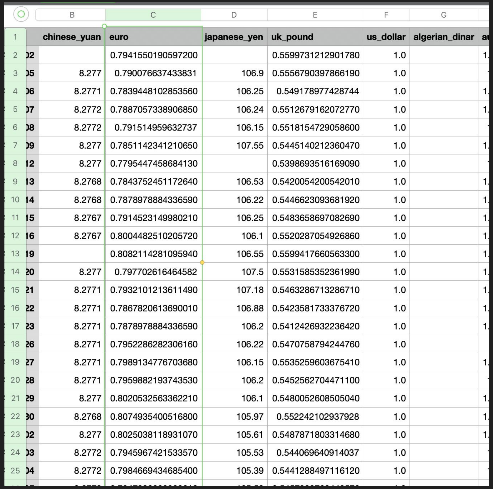
EDA and Preprocessing

- Handling Missing Data: Used forward fill to handle any gaps in the dataset.

```python
# Handle missing values with forward fill
data.fillna(method='ffill', inplace=True)
print("\nMissing values after forward-fill:")
print(data.isnull().sum())
```

- Feature Selection: Focused on the euro column to build our target variable.
- Creation of Lagged Variable:
  - Created euro_lag_1, a 1-day lagged value, as the main feature to capture short-term trends.

```python
euro_data['euro_lag_1'] = euro_data['euro'].shift(1)
```



		B	C	D	E	F	G
1		chinese_yuan	euro	japanese_yen	uk_pound	us_dollar	algerian_dinar
2	02		0.7941550190597200		0.5599731212901780	1.0	
3	05	8.277	0.790076637433831	106.9	0.5556790397866190	1.0	
4	06	8.2771	0.7839448102853560	106.25	0.549178977428744	1.0	
5	07	8.2772	0.7887057338906850	106.24	0.5512679162072770	1.0	
6	08	8.2772	0.791514959632737	106.15	0.5518154729058600	1.0	
7	09	8.277	0.7851142341210650	107.55	0.5445140212360470	1.0	
8	12	8.277	0.7795447458684130		0.5398693516169090	1.0	
9	13	8.2768	0.7843752451172640	106.53	0.5420054200542010	1.0	
10	14	8.2768	0.7878978884336590	106.22	0.5446623093681920	1.0	
11	15	8.2767	0.7914523149980210	106.25	0.5483658697082690	1.0	
12	16	8.2767	0.8004482510205720	106.1	0.5520287054926860	1.0	
13	19		0.8082114281095940	106.55	0.5599417660563300	1.0	
14	20	8.277	0.797702616464582	107.5	0.5531585352361990	1.0	
15	21	8.2771	0.7932101213611490	107.18	0.5463286713286710	1.0	
16	22	8.2771	0.7867820613690010	106.88	0.5423581733376720	1.0	
17	23	8.2771	0.7878978884336590	106.2	0.5412426932236420	1.0	
18	26	8.2771	0.7952286282306160	106.22	0.5470758794244760	1.0	
19	27	8.2771	0.7989134776703680	106.15	0.5535259603675410	1.0	
20	28	8.2771	0.7959882193743530	106.2	0.5452562704471100	1.0	
21	29	8.277	0.8020532563362210	106.1	0.5480052608505040	1.0	
22	30	8.2768	0.8074935400516800	105.97	0.552242102937928	1.0	
23	02	8.277	0.8025038118931070	105.61	0.5487871803314680	1.0	
24	03	8.2772	0.7945967421533570	105.53	0.544069640914037	1.0	
25	04	8.2772	0.7984669434685400	105.39	0.5441288497116120	1.0	

Model Training

Model Chosen: Random Forest Regressor

Reason: Random Forest is effective for time series prediction as it leverages multiple decision trees to capture trends.

Training Process:

- Split data into training (80%) and testing (20%) sets.
- Trained the model using only the 1-day lag as the primary feature.

```
# Define the feature (only 'euro_lag_1') and target
X = euro_data[['euro_lag_1']]  # Only the 1-day lag
y = euro_data['euro']

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

```
# Initialize and train the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42, oob_score=True)
rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(oob_score=True, random_state=42)
```

Model Evaluation Metrics

Out-of-Bag (OOB) Score: Used to assess the model’s generalization.

Mean Absolute Error (MAE): Indicates average error in prediction.

Mean Squared Error (MSE): Measures the variance of prediction errors.

R^2 Score: Measures the percentage of variance explained by the model.

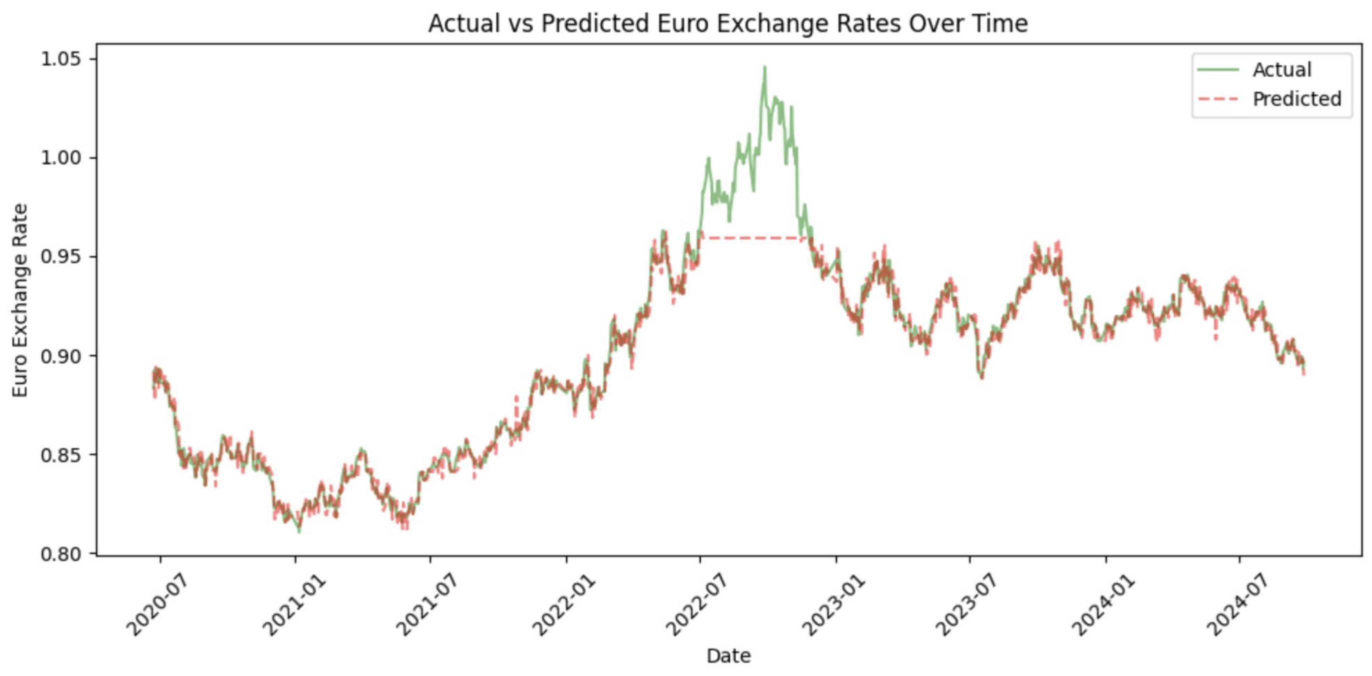
- Higher R^2 suggests better model performance.

```python
y_pred = rf_model.predict(X_test)

oob_score = rf_model.oob_score_
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Out-of-Bag Score: {oob_score}')
print("Mean Squared Error (MSE):", mse)
print('Mean Absolute Error (MAE):', mae)
print(f'R-squared Score: {r2}')
```

```
Out-of-Bag Score: 0.9948508067815014
Mean Squared Error (MSE): 0.0001995978640355459
Mean Absolute Error (MAE): 0.00704736401456525
R-squared Score: 0.9207432390481782
```

Visualization of Results

Actual vs. Predicted Plot:



Key Insights and Conclusion

Key Takeaways:

- The model captured short-term patterns well, leveraging recent values.
- High R^2 and low errors indicate accurate predictions for this task.

Conclusion:

- Random Forest is effective for predicting exchange rates with minimal feature engineering.
- This model could be enhanced with more features or by using an ensemble approach for better accuracy.

Resources:

https://www.keboola.com/blog/random-forest-regression

https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.RandomForestRegressor.html