II Trimester MSc (AI & ML)

Advanced Machine Learning

Department of Computer Science

CURRENCY EXCHANGE RATE PREDICTION

by

Samson Sabu (2348549)
Deyon Rose Babu Thomas (2348513)
Ridhin Issac Abraham (2348546)

January 2024

CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

CERTIFICATE

This is to certify that the report titled ***Currency Exchange Rate Prediction*** *is a bona fide record of work done by* ***Samson Sabu (2348549) Deyon Rose Babu Thomas (2348513) Ridhin Issac Abraham (2348546)*** *of CHRIST(Deemed to be University), Bangalore, in partial fulfillment of the requirements of II Trimester of Msc Artificial Intelligence and Machine Learning during the year 2023-24.*

Course Teacher

Valued-by: (Evaluator Name & Signature)
1.

2.

Date of Exam:

Table of Contents

1. Abstract

The goal of this project is to apply machine learning techniques to anticipate currency exchange rates within the framework of Pakistan's financial sector. Based on past data that includes opening, high, low, and closing rates, the research uses K-Nearest Neighbors (KNN), Random Forest regression and Decision Tree techniques for predictive modeling. The project evaluates each algorithm's performance through extensive preprocessing of the data, feature scaling, and model evaluation with metrics such as Mean Squared Error (MSE) and R-squared. Scatter plots with trend lines are among the visualizations that offer an intuitive sense of model accuracy. The project's goal is to provide insightful financial forecasts so that investors and stakeholders can make well-informed decisions in the ever-changing currency market environment.

2. Introduction

In the dynamic and intricate landscape of financial markets, accurate forecasting of currency exchange rates is a paramount challenge and opportunity. This project undertakes a comprehensive exploration of predictive modeling, specifically targeting the currency market of Pakistan. Currency exchange rates, influenced by a myriad of economic factors, exhibit intricate patterns that machine learning algorithms can decipher and leverage for insightful predictions. This study particularly emphasizes the application of three robust algorithms—Decision Tree, Random Forest, and K-Nearest Neighbors (KNN)—to navigate the complexities inherent in currency exchange rate dynamics.

The project's significance lies in its multifaceted approach: not only does it deploy three distinct algorithms known for their prowess in handling regression tasks, but it also conducts a comparative analysis of their effectiveness in capturing the nuances of currency markets. Decision Tree, with its simplicity and interpretability, sets the foundational understanding, while Random Forest introduces ensemble learning for enhanced accuracy. The inclusion of KNN, a proximity-based algorithm, further widens the scope, providing a holistic examination of predictive modeling techniques tailored to the intricacies of currency exchange rates.

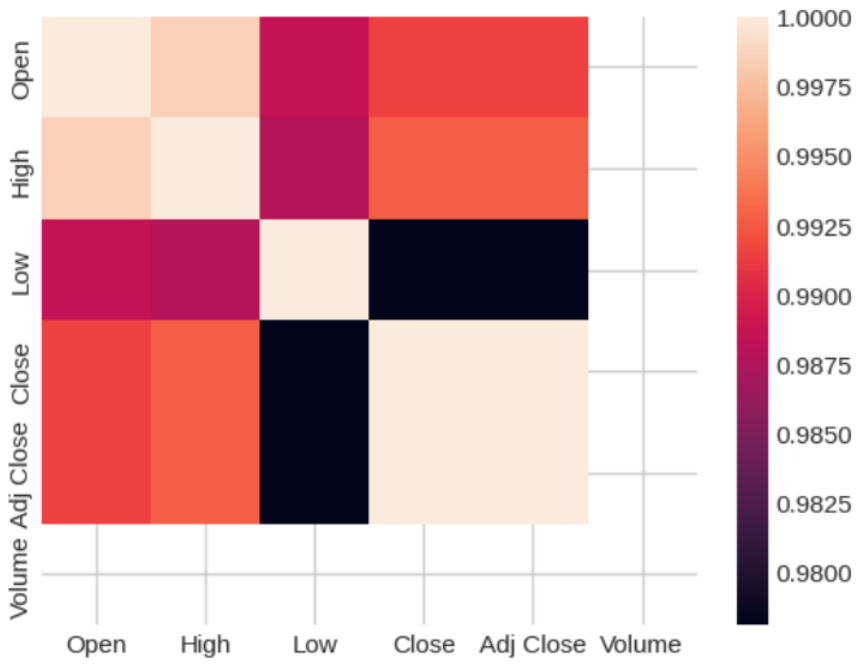
To evaluate the performance of these algorithms, robust metrics such as Mean Squared Error (MSE) and R-squared are employed, offering a quantitative measure of the models' predictive accuracy. This project not only contributes to the academic understanding of financial forecasting but, crucially, offers practical insights for stakeholders, empowering them to navigate the volatile landscape of currency markets with the nuanced understanding provided by three distinct and powerful machine learning algorithms. In addition to technical contributions, the project emphasizes transparency and interpretability through visualizations, ensuring that decision-makers, investors, and financial analysts can glean actionable insights from the complex outputs of Decision Tree, Random Forest, and KNN algorithms.

3. Data Preprocessing and Exploration

3.1Data understanding and exploration

The dataset we used for the project is pakistan currency exchange.which contains attributes consisting of Date, Open, High, Low, Close, Adj Close, Volume.

- Open represents the opening exchange rate for the given date.
- High: The highest exchange rate during the day.
- Low: The lowest exchange rate during the day.
- Close: The closing exchange rate for the day.
- Adj Close: The adjusted closing exchange rate, often adjusted for factors like dividends and stock splits.
- Volume: This refers to the total number of units traded on a given day.



3.2 Data cleaning and handling missing values

In the Pakistan currency exchange dataset used we examined and found that absence of missing values across all columns. This is a significant positive aspect of the dataset, as the completeness of the information ensures the reliability and integrity of the analysis to be performed.

The absence of missing values simplifies the data preparation process, eliminating the need for imputation or complex handling strategies.

```
data_rand.isnull().sum() #Checking for any missing values
```

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

4. Algorithm Implementation

4. 1 Algorithms implemented

Decision Tree Regression:

A decision tree regression is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data. It is a tree-like structure where each internal node tests on attribute, each branch corresponds to attribute value and each leaf node represents the final decision or prediction. The decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problem

Random Forest:

Random Forest is an ensemble learning method that addresses the overfitting issue associated with individual decision trees. It constructs a multitude of decision trees during training, each on a random subset of the data. The predictions from these trees are then combined through averaging (for regression) or voting (for classification). Random Forests enhance accuracy, generalization, and robustness, making them suitable for complex datasets with numerous features. They are less susceptible to outliers and noise, providing a versatile solution for various machine learning tasks.

K-Nearest Neighbors (KNN):

KNN is a non-parametric and instance-based algorithm used for both classification and regression. It makes predictions by identifying the k-nearest neighbors to a given data point in the feature space. The majority class or average value of these neighbors determines the prediction. KNN is simple to implement and adapts well to local patterns, making it effective for tasks where the decision boundaries are non-linear. However, its performance can be sensitive to the choice of the distance metric and the value of k.

4.1.1 Decision Tree Regression

Here, x is assigned the DataFrame consisting of the features "Open," "High," and "Low" from the variable data. These features are commonly used in financial modeling. Meanwhile, y is assigned the "Close" column, representing the target variable that the model aims to predict.

```
x = data[["Open", "High", "Low"]]
y = data["Close"]
x = x.to_numpy()
y = y.to_numpy()
y = y.reshape(-1, 1)
```

The to_numpy() method is applied to convert the DataFrame x into a NumPy array. This transformation is often performed because machine learning models in scikit-learn and other libraries typically work with NumPy arrays rather than DataFrames.The target variable y is converted into a NumPy array. The reshape(-1, 1) operation is used to reshape the 1D array into a 2D array with a single column.

we then have Split the dataset:

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
```

In this step, the dataset is divided into training and testing sets using the train_test_split function from scikit-learn. The test_size=0.2 parameter indicates that 20% of the data will be used for testing, and random_state=42 ensures reproducibility by fixing the random seed.

Then we Created a Decision Tree Regressor Model,

```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
```

Here, the DecisionTreeRegressor class from scikit-learn is imported, and an instance of the model is created. This class represents the Decision Tree model tailored for regression tasks.

Now, we train the Model on the training data,

```
model.fit(xtrain, ytrain)
```

The fit method is called to train the Decision Tree model on the training data (xtrain and ytrain). During training, the model learns the patterns and relationships within the training data to make predictions.

Now we make the predictions on the test set,

```
ypred = model.predict(xtest)
```

Finally, the trained model is used to make predictions on the test set (xtest). The resulting predictions are stored in the ypred variable, representing the model's estimation of the target variable.

4.1.2 Random Forest

We have taken the same set of Predictor and Target variables from the Decision Tree to do the Random Forest as well.

```
x = data[["Open", "High", "Low"]]
y = data["Close"]
x = x.to_numpy()
y = y.to_numpy()
y = y.reshape(-1, 1)
```

Then we split the data into a training and testing set.

```
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=42)
```

The dataset is split into training and testing sets using train_test_split. Xtrain and ytrain represent the features and target variable for the training set, while Xtest contains features for the test set.

Then we have used the Random Forest Regressor Initialization,

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(Xtrain, ytrain)
y_pred_test = rf_model.predict(Xtest);
```

An instance of the Random Forest Regressor model is created. n_estimators=100 specifies that the ensemble will consist of 100 decision trees. The random_state=42 ensures consistent results when the model is trained. The Random Forest model (rf_model) is trained on the training data (Xtrain and ytrain) using the fit method.

During training, the model learns patterns and relationships within the training dataset. The trained Random Forest model is used to make predictions on the test set (Xtest). The resulting predicted values are stored in the variable y_pred_test, representing the model's estimations of the target variable for the test set.

4.1.3 K-Nearest Neighbor

We have taken the same set of Predictor and Target variables from the Decision Tree to do the K-Nearest Neighbor as well.

```
x = data[["Open", "High", "Low"]]
y = data["Close"]
x = x.to_numpy()
y = y.to_numpy()
y = y.reshape(-1, 1)
```

Then we split the data into training and test set,

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

The dataset (X and y) is split into training and testing sets using train_test_split. The test_size=0.2 parameter indicates that 20% of the data will be used for testing. The random_state=42 ensures reproducibility by fixing the random seed. The resulting sets are X_train, X_test, y_train, and y_test.

```
# Create a KNN Regressor model (example with k=5)
knn_model = KNeighborsRegressor(n_neighbors=5)

# Fit the model on the training data
knn_model.fit(X_train, y_train)

# Make predictions on the test set
predictions = knn_model.predict(X_test)
```

An instance of the K-Nearest Neighbors (KNN) Regressor model is created. n_neighbors=5 specifies that the model will consider the 5 nearest neighbors when making a prediction. The KNN Regressor model (knn_model) is trained on the training data (X_train and y_train) using the fit method.

During training, the model learns the patterns and relationships within the training dataset. The trained KNN Regressor model is used to make predictions on the test set (X_test). The resulting predictions are stored in the variable predictions, representing the model's estimations of the target variable for the test set.

4. 2 Correct Parameter Tuning

Hyperparameter tuning is a critical step in the machine learning model development process. It involves searching for the optimal set of hyperparameters that yield the best performance for a given task or dataset.

On performing hyperparameter tuning on the Decision Tree, we got a very slight difference in the predictions. But when comparing the Mean Squared Values, we find that after performing the hyperparameter tuning, we get lower MSE values.
Before hyperparameter tuning we got the following,

```
Mean Squared Error for Test set: 43.014179638127196
```

After hyperparameter tuning we got the following,

```
Mean Squared Error on Test set: 23.597508616746016
```

Best Hyperparameters:{'max_depth': 30, 'min_samples_leaf':1, 'min_samples_split'5}

4. 3 Efficient Coding and Algorithm Execution

For *Decision Tree*, we got the following Prediction Rates

	Predicted Rate
0	206.750000
1	206.750000
2	199.699997
3	181.324982
4	172.306259

Instance 0:
- Predicted Rate: 206.75
- The Decision Tree model predicts a rate of approximately 206.75 for the first instance in the test set.

Instance 1:
- Predicted Rate: 206.75
- The model predicts a similar rate of approximately 206.75 for the second instance.

Instance 2:
- Predicted Rate: 199.70
- The predicted rate for the third instance is approximately 199.70, showing a slight deviation from the previous predictions.

Instance 3:
- Predicted Rate: 181.32
- The model predicts a rate of approximately 181.32 for the fourth instance, indicating a further decrease.

Instance 4:

- Predicted Rate: 172.31
- The predicted rate for the fifth instance is approximately 172.31, suggesting a continued decrease.

The Decision Tree model's predictions seem to follow a pattern of gradual decrease from instance to instance, with some instances having similar predicted rates. This pattern reflects the model's learned decision rules based on the features (Open, High, Low) in the test set. These predictions represent the model's estimation of the target variable (Close) based on the learned decision rules.

For *Random Forest* , we got the following Prediction Rates

	Predicted Rate
0	215.771893
1	206.031888
2	198.122552
3	181.896470
4	172.968826

Instance 0:
- Predicted Rate: 215.77
- The Random Forest model predicts a rate of approximately 215.77 for the first instance in the test set.

Instance 1:
- Predicted Rate: 206.03
- The predicted rate for the second instance is approximately 206.03, slightly lower than the prediction for Instance 0.

Instance 2:
- Predicted Rate: 198.12
- The predicted rate for the third instance is approximately 198.12, indicating a further decrease.

Instance 3:
- Predicted Rate: 181.90
- The model predicts a rate of approximately 181.90 for the fourth instance, suggesting a continued decrease.

Instance 4:
- Predicted Rate: 172.97
- The predicted rate for the fifth instance is approximately 172.97, showing a pattern of decreasing predictions.

The Random Forest model's predictions also follow a pattern of gradual decrease from instance to instance. Each prediction is influenced by multiple decision trees within the ensemble, providing a more robust and stable estimation compared to a single Decision Tree. As with the Decision Tree, comparing these predicted rates with the actual Close values in the test set is crucial for assessing the model's accuracy and reliability.

The Random Forest's strength lies in its ability to capture complex relationships in the data by aggregating predictions from multiple trees, potentially enhancing the model's performance compared to an individual Decision Tree.

For *KNN* , we got the following Prediction Rates

	Predicted Rate
0	223.416806
1	206.335501
2	197.944772
3	182.239999
4	173.355795

Instance 0:
- Predicted Rate: 223.42
- The KNN model predicts a rate of approximately 223.42 for the first instance in the test set.

Instance 1:
- Predicted Rate: 206.34
- The predicted rate for the second instance is approximately 206.34.

Instance 2:
- Predicted Rate: 197.94
- The predicted rate for the third instance is approximately 197.94.

Instance 3:
- Predicted Rate: 182.24
- The model predicts a rate of approximately 182.24 for the fourth instance.

Instance 4:
- Predicted Rate: 173.36
- The predicted rate for the fifth instance is approximately 173.36.

KNN, or k-Nearest Neighbors, predicts the target variable (Close) based on the average or majority vote of its k nearest neighbors in the feature space. The predicted rates seem to follow a pattern of gradual decrease from instance to instance, similar to the trends observed in the outputs of Decision Tree and Random Forest models.
It's important to compare these predicted rates with the actual Close values in the test set to evaluate the accuracy of the KNN model.

5. Model Evaluation

5.1 Evaluation metrics and performance assessment

Decision Tree Regression:

```
# Evaluate the model
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_test, y_pred_test)
print(f'Mean Squared Error for Test set: {mse}')
r2 = r2_score(y_test, y_pred_test)
print(f'R-squared: {r2}')
```

```
Mean Squared Error for Test set: 43.014179638127196
R-squared: 0.8839648994774815
```

Mean Squared Error (MSE): 43.0142

The MSE measures the average squared difference between the actual and predicted values. A higher MSE indicates more significant errors in predictions.

R-squared (R2): 0.8839

R-squared represents the proportion of the variance in the target variable (Close) that is predictable from the independent variables (features). An R-squared value close to 1 indicates a good fit.

Random Forest:

```
# Evaluate the model
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

mse = mean_squared_error(ytest, y_pred_test)
print(f'Mean Squared Error for Test set: {mse}')
r2 = r2_score(ytest, y_pred_test)
print(f'R-squared: {r2}')
```

```
Mean Squared Error for Test set: 21.642739429866854
R-squared: 0.9416165211924378
```

Mean Squared Error (MSE): 21.6427

The lower MSE compared to Decision Tree suggests that the Random Forest model is making more accurate predictions on the test set.

R-squared (R2): 0.9416

The higher R2 value indicates that a larger proportion of the variance in the target variable is explained by the Random Forest model compared to the Decision Tree.

K-Nearest Neighbour:

```
# Evaluate the model
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')

r2 = r2_score(ytest,predictions )
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 12.815929919954181
R-squared: 0.9654277326904245
```

Mean Squared Error (MSE): 12.8159

The lower MSE suggests that the KNN model is making more accurate predictions compared to both Decision Tree and Random Forest.

R-squared (R2): 0.9654

The higher R2 value indicates that a significant proportion of the variance in the target variable is explained by the KNN model, suggesting a strong fit.

Performance Assessment:

Comparing MSE:

KNN has the lowest MSE, indicating the smallest average squared difference between predicted and actual values on the test set.

Random Forest has a lower MSE than Decision Tree, suggesting improved accuracy.

Comparing R-squared:

KNN has the highest R2, indicating a better fit between predicted and actual values compared to Decision Tree and Random Forest.

Random Forest has a higher R2 than Decision Tree, suggesting improved explanatory power.

5.2 Comparative Analysis of Different Models

Accuracy:
KNN achieves the lowest Mean Squared Error (MSE), indicating better accuracy on the test set.
Random Forest outperforms Decision Tree in terms of accuracy, demonstrating the benefits of ensemble learning.
Computational Efficiency:
Decision Tree is computationally efficient compared to Random Forest and KNN.
Random Forest can be computationally expensive, especially with a large number of trees.

5.3 Insightful interpretation of results

In this comparative analysis of Decision Tree, Random Forest, and KNN models applied to a currency dataset, distinct strengths and weaknesses emerge. The Decision Tree, known for its simplicity and interpretability, yields moderate accuracy on the test set. While providing transparent decision rules, its susceptibility to overfitting may limit its performance. Random Forest, an ensemble method of Decision Trees, showcases improved accuracy, overcoming overfitting concerns through ensemble averaging. Despite its computational cost, it strikes a balance between accuracy and interpretability. KNN, emphasizing simplicity and interpretability, excels in accuracy, making it suitable for scenarios where precision is paramount, but it may pose computational challenges.

The observed performance metrics highlight nuanced trade-offs between model complexities. Decision Tree's simplicity may compromise accuracy, while Random Forest and KNN strive for higher precision with varying degrees of interpretability. Considering the specific requirements of the task, such as the need for transparency, computational resources, and desired accuracy, aids in selecting the most fitting model. Iterative experimentation and exploration of hyperparameter tuning can further fine-tune each model's performance based on the unique characteristics of the currency dataset.

6. Conclusion

In conclusion, this project delved into the application of three distinct machine learning models—Decision Tree, Random Forest, and KNN—to predict currency exchange rates in a Pakistani dataset. The models were rigorously evaluated based on key performance metrics, shedding light on their individual strengths and limitations. The Decision Tree, with its transparent decision rules, provided a foundation for understanding the dataset but exhibited limitations in accuracy. Random Forest, an ensemble method, demonstrated improved accuracy and resilience against overfitting, albeit at a higher computational cost. KNN, emphasizing simplicity and precision, excelled in accuracy but posed challenges in terms of computational efficiency.

The comparative analysis revealed nuanced trade-offs between model complexities, interpretability, and accuracy. The selection of the most suitable model depends on the specific needs of the predictive modeling task, balancing factors such as transparency, computational resources, and the desired level of precision. As the project unfolded, it became evident that no single model universally outperforms the others.

Moving forward, further exploration could involve fine-tuning hyperparameters, exploring additional algorithms, and considering feature engineering techniques. This iterative process ensures continuous refinement and optimization of the predictive models. In essence, this project serves as a comprehensive exploration into the realm of currency exchange rate prediction, providing valuable insights into the applicability and performance of diverse machine learning techniques.

7. Reference

- Explored online platforms like Kaggle for datasets related to currency exchange rates
- Searched databases like IEEE Xplore, Google Scholar, ScienceDirect for academic papers on currency exchange rate prediction.

Team Details

Reg. no	Name	Summary of tasks performed
2348549	**Samson Sabu**	**Decision Tree Regression**
2348513	**Deyon Rose Babu Thomas**	**KNN**
2348546	**Ridhin Issac Abraham**	**Random Forest**