

# Financial Forecasting using Exchange Rates

Vijay Kumar

9<sup>th</sup> June 2024

GitHub Link : [https://github.com/VijayKumar203/Forecasting\\_Project](https://github.com/VijayKumar203/Forecasting_Project)

## Table of Contents

<b>INTRODUCTION.....</b>	<b>2</b>
<b>DATA LOADING AND PREPROCESSING .....</b>	<b>2</b>
Loading and Cleaning Data .....	2
Exploratory Data Analysis.....	2
<b>FEATURE ENGINEERING.....</b>	<b>4</b>
<b>MODEL TRAINING .....</b>	<b>5</b>
Train-Test Split .....	5
Multilayer Perceptron Model.....	5
<b>MODEL EVALUATION.....</b>	<b>6</b>
Predictions and Metrics .....	6
Visualization of results .....	7
<b>CONCLUSION .....</b>	<b>9</b>
<b>REFERENCES .....</b>	<b>9</b>
<b>APPENDIX .....</b>	<b>10</b>
Complete Code .....	10

## **Introduction**

This project's aim is to predict future exchange rates of Euro to US Dollar (EUR/USD) using historical exchange rate data. The key steps in this project include data cleaning, feature engineering, model training, and evaluation. A Multilayer Perceptron (MLP) model is used for this purpose, with the `neuralnet` package in R. This project would be able to assist financial analysts and investors in making informed decisions based on accurate exchange rate predictions.

## **Data Loading and Preprocessing**

### **Loading and Cleaning Data**

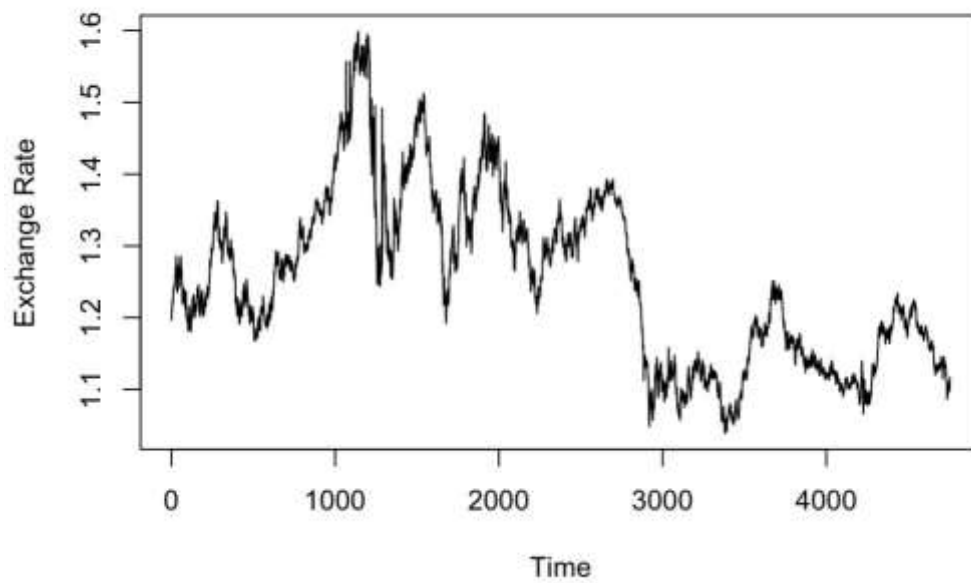
The historical data of EUR/USD exchange rates is loaded from a CSV file downloaded from a dataset from Kaggle.com.

This data includes various features, with the closing exchange rate being the primary variable of interest. The data is then cleaned by converting the exchange rates to numeric values and removing any missing or invalid entries. This ensures that the dataset is ready for analysis and modeling.

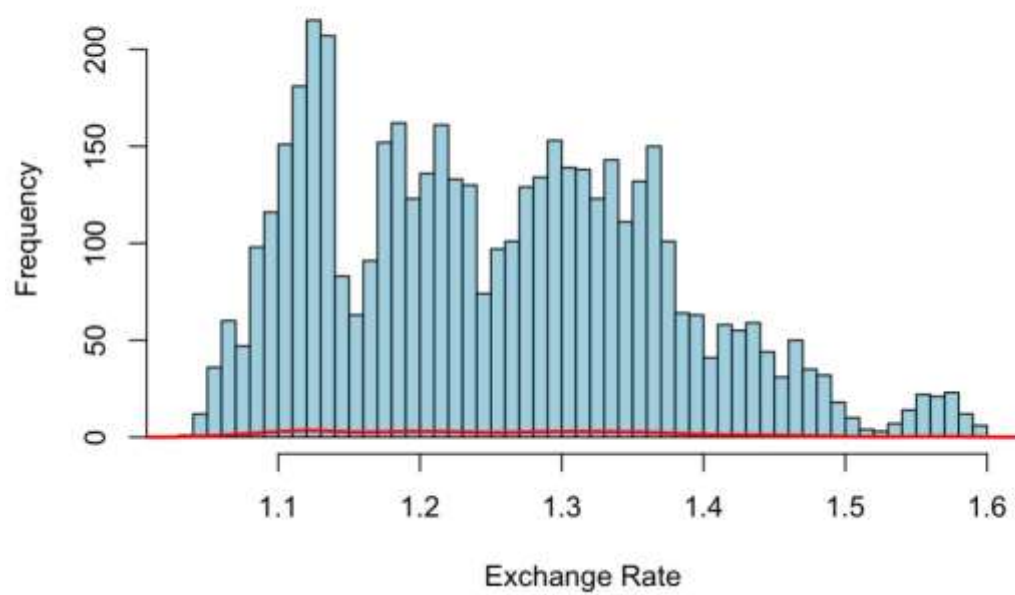
### **Exploratory Data Analysis**

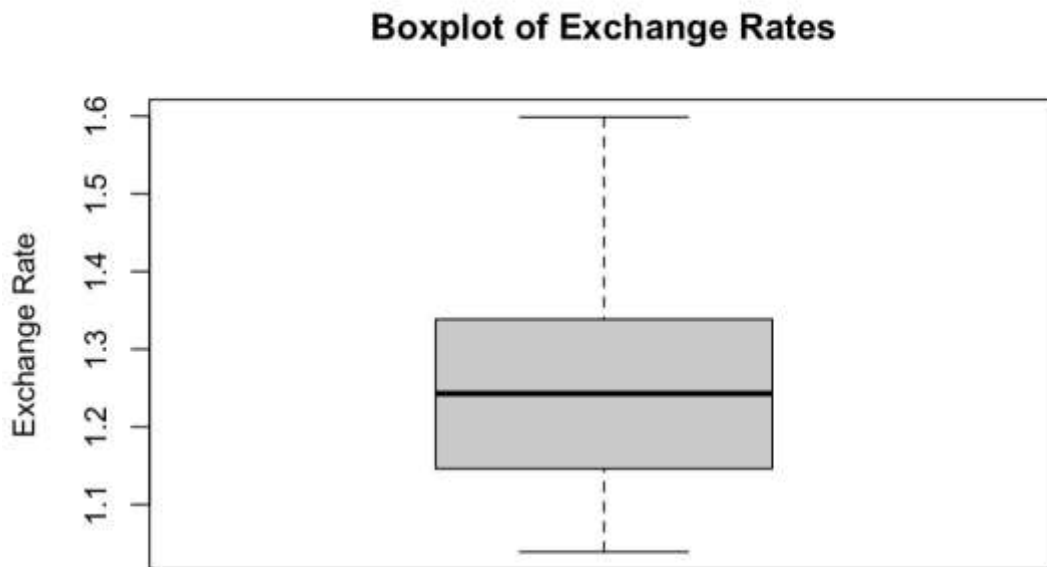
The plots below show the variation of the exchange rate data.

**Time Series Data**



**Histogram of Exchange Rates**



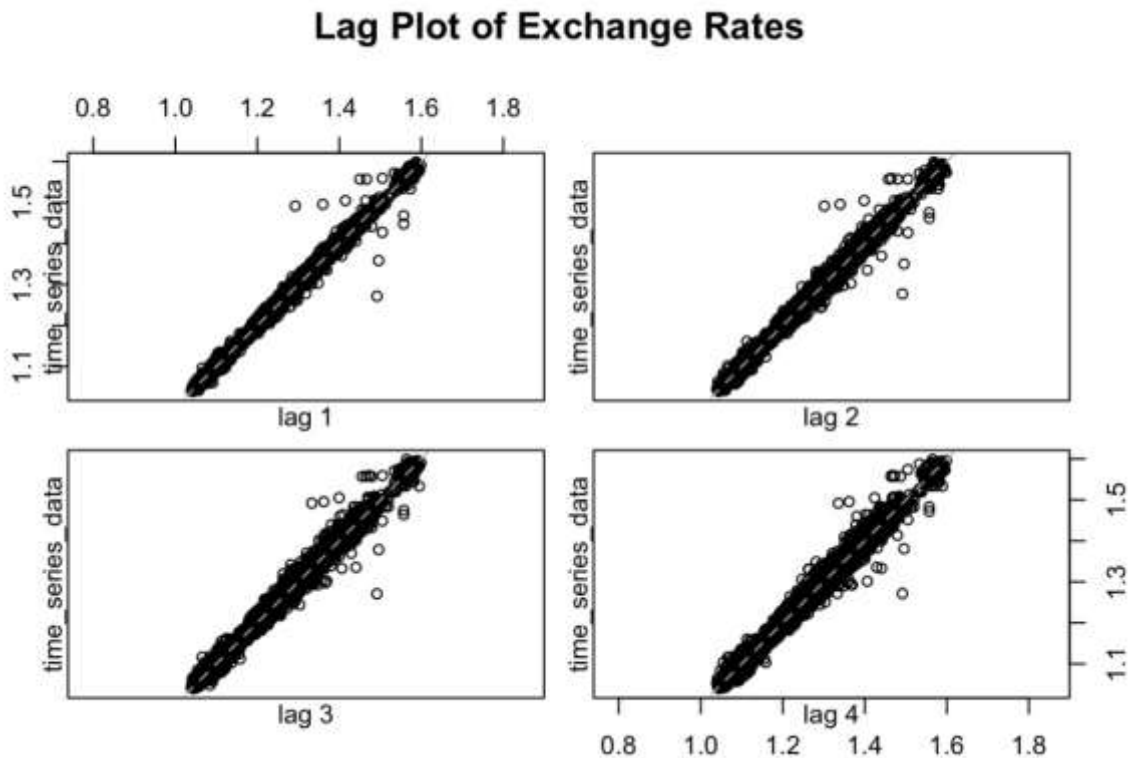


The boxplot shows that there are no outliers in the dataset under the closing rate column.

## Feature Engineering

To improve the predictive power of the model, feature engineering is performed. This involves creating lagged versions of the exchange rate data, which serve as input features for the model.

Exchange rates from previous days (lags) are used to predict the future exchange rate. These lagged features help the model capture temporal dependencies in the data in time series forecasting. They were lagged upto a t-4 level. Then all the NA containing rows were removed too.



## Model Training

### Train-Test Split

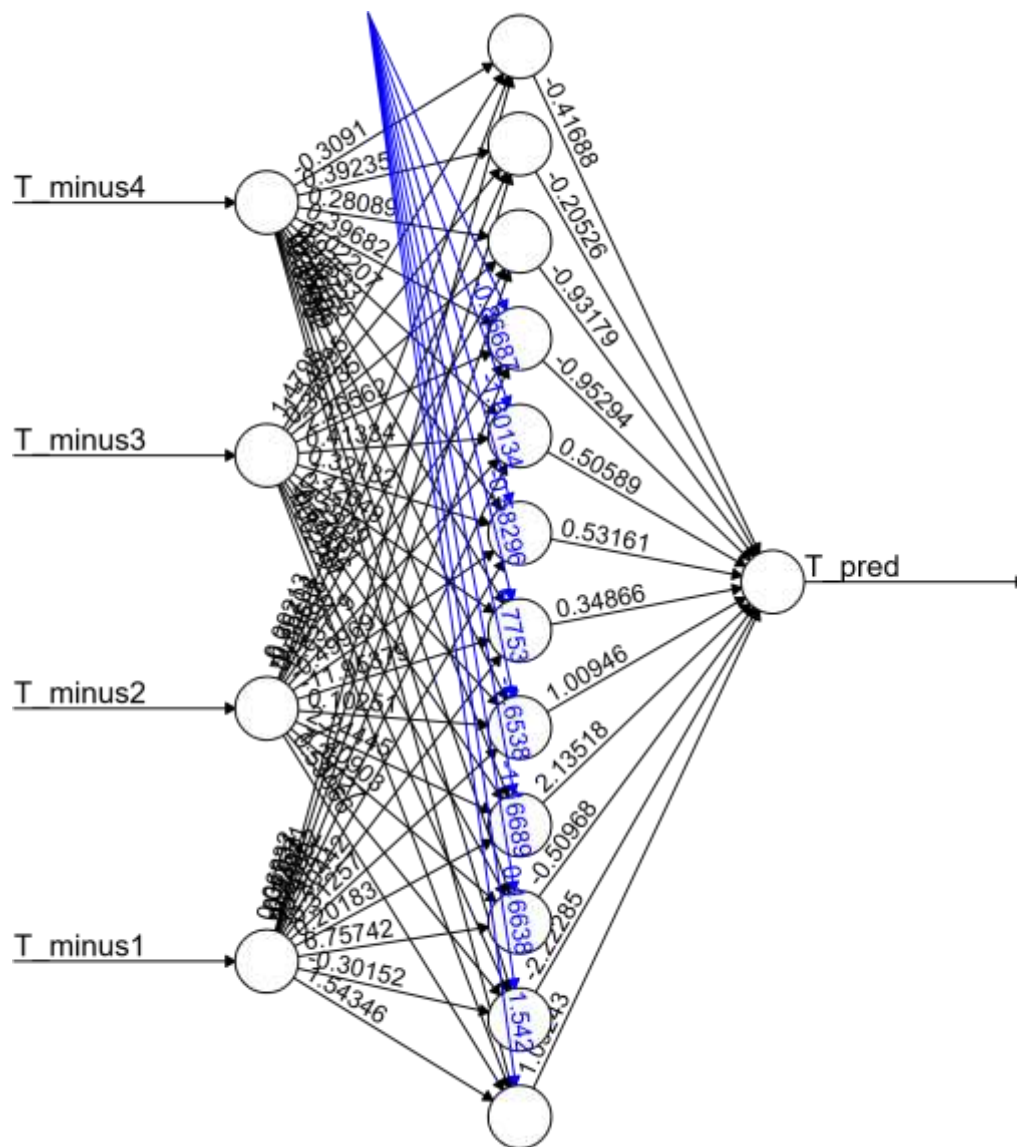
The dataset is divided into training and testing sets using an 80:20 split. The training set is used to train the model, while the testing set is used to evaluate its performance.

This split ensures that the model is evaluated on data it has not seen during training, so it can provide an unbiased performance.

### Multilayer Perceptron Model

An MLP model is constructed using the `neuralnet` package in R. Several models were tested varying the inputs, activation functions, hidden layers and number of nodes in each. Then using the statistical measures given below the best model was identified as well as the model which has least parameters to be trained to make the model more efficient.

The best model is designed with one hidden layer containing 12 neurons, and a logistic activation function is used. The model is trained to minimize the error in predicting the future exchange rate based on the lagged features. The training process involves optimizing the model weights through backpropagation.



## Model Evaluation

### Predictions and Metrics

The trained model is used to predict the exchange rates on the testing set.

Various statistical indices are computed to evaluate the model's performance, including Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Symmetric Mean Absolute Percentage Error (sMAPE). The model with the least errors was selected as the best model.

These metrics provide insights into the accuracy and reliability of the model's predictions.

RMSE: 0.004591427  
MAE: 0.003445648  
MAPE: 0.003008078 sMAPE:  
0.003006853

## Other models

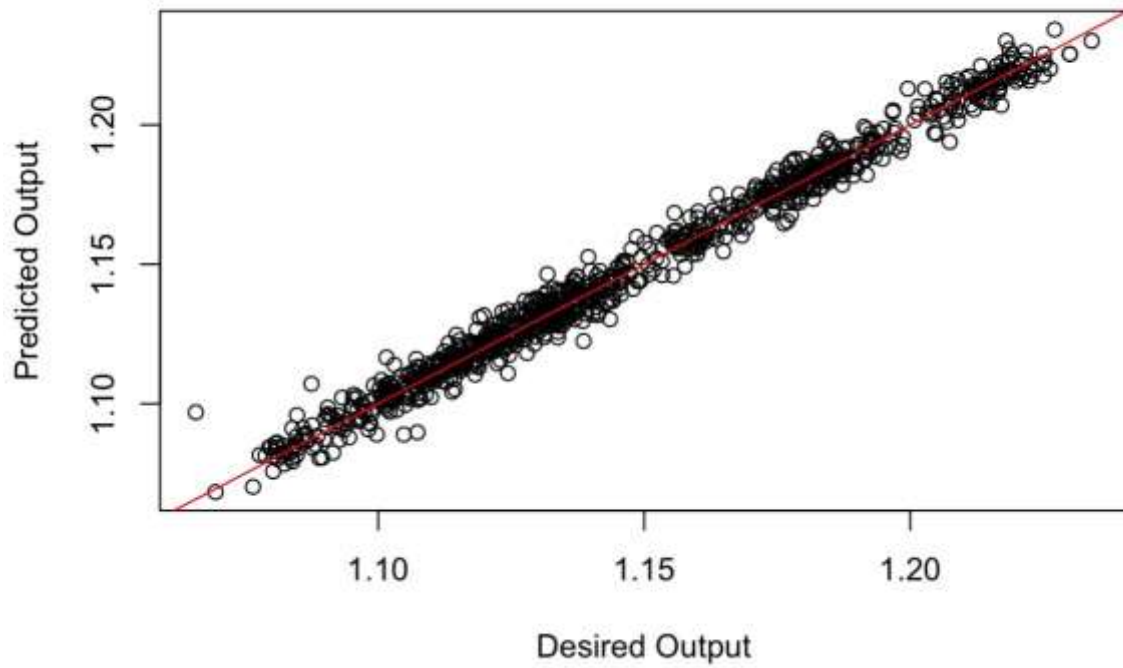
Model	Input	Hidden Layers	Hidden nodes	Linear/Non linear	Activation Function
dataset_model1	T-1, T-2, T-3, T-4	1	12	Linear	Logistic
dataset_model2	T-1, T-2, T-3, T-4	1	8	Non-linear	Logistic
dataset_model3	T-1, T-2, T-3	1	6	Linear	Logistic
dataset_model4	T-1, T-2, T-3	1	4	Non-linear	tanh
dataset_model5	T-1, T-2	1	12	Linear	Logistic
dataset_model6	T-1, T-2	1	8	Linear	Logistic
dataset_model7	T-1, T-2, T-3	1	12	Linear	tanh
dataset_model8	T-1, T-2, T-3, T-4	2	3,5	Linear	Logistic
dataset_model9	T-1, T-2, T-3, T-4	2	2,2	Non-linear	Logistic
dataset_model10	T-1, T-2, T-3	2	2,2	Linear	Logistic
dataset_model11	T-1, T-2, T-3	2	6,8	Non-linear	tanh
dataset_model12	T-1	2	5,12	Linear	Logistic
dataset_model13	T-1	2	4,8	Linear	Logistic
dataset_model14	T-1, T-2	2	5,12	Non-linear	Logistic
dataset_model15	T-1, T-2	2	4,8	Non-linear	tanh

## Visualization of results

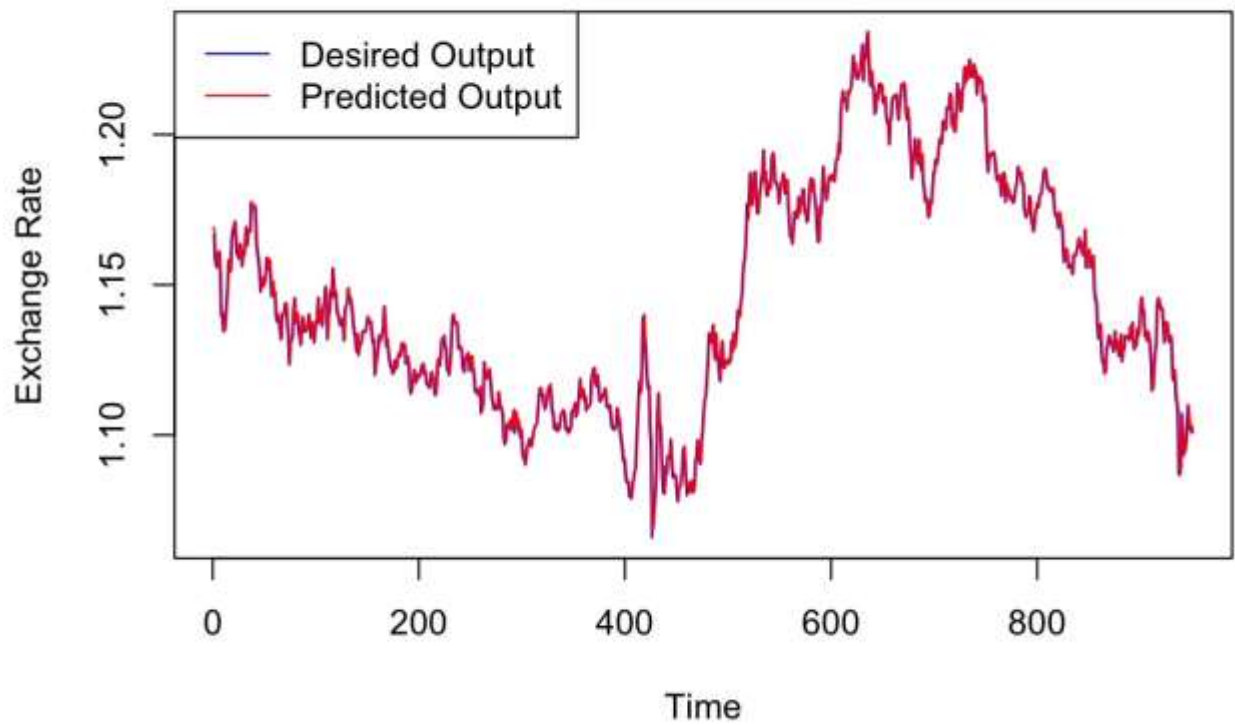
The results of the model's predictions are visualized to compare them with the actual exchange rates.

Scatter plot and time series plot were created to show the relationship between the predicted and actual values. These visualizations help identify patterns in the interpretation of the model's performance.

**Predicted vs. Desired Output**



**Predicted vs. Desired Output**





## Conclusion

In this project, a Multilayer Perceptron (MLP) model was successfully developed to predict future EUR/USD exchange rates using historical data. The model's performance was evaluated using several statistical indices, and visualizations were created to compare the predicted and actual values.

## Potential Impact

The model holds value for financial analysts and investors, providing them with accurate predictions of exchange rates. This can help them in making informed decisions, optimizing strategies, and reducing risks associated with currency fluctuations.

## Limitations

1. The model performance heavily depends on the quality and relevance of historical data used.
2. It assumes that historical patterns will continue to hold in the future, which may not always be the case.
3. External factors such as geopolitical events and economic policies, not considered in this model, can significantly impact exchange rates.

## Future Work

### 1. Enhanced Feature Engineering

Explore additional features such as technical indicators or sentiment analysis.

### 2. Real-Time Prediction

Develop a pipeline for real-time prediction using streaming data.

## References

Chowdhury. M. S. et al. (2024). Deep Learning Models for Stock Market Forecasting: A Comprehensive Comparative Analysis.  
10.32996/jbms.2024.6.2.9

Nagadia. M. (2022). Currency Rate Euro€ to USD\$ (2003-2022).  
<https://www.kaggle.com/datasets/meetnagadia/currency-rate-euro-to-usd>

Tang. P and Zhang. W. (2024). PDMLP: Patch-based Decomposed MLP for Long-Term Time Series Forecasting.  
[https://www.researchgate.net/publication/380821515\\_PDMLP\\_Patchbased\\_Decomposed\\_MLP\\_for\\_Long-Term\\_Time\\_Series\\_Forecastin](https://www.researchgate.net/publication/380821515_PDMLP_Patchbased_Decomposed_MLP_for_Long-Term_Time_Series_Forecastin)

## Appendix

### Complete Code

```
# Financial Forecasting
```

```
# Required libraires
```

```
packages <- c("neuralnet", "caret", "Metrics", "dplyr")
installed_packages <- installed.packages() for (pkg in
packages) { if (!pkg %in% installed_packages[,
"Package"]) {
  install.packages(pkg, dependencies = TRUE)
}
}
library(neuralnet)
library(caret) library(Metrics)
library(dplyr)
```

```
# Loading and preprocessing data
```

```
exchange_rate_data <- read.csv("EURUSDx.csv") dim(exchange_rate_data)
```

```
# Cleaning data time_series_data <-
```

```
exchange_rate_data$Close time_series_data <-
```

```
as.numeric(time_series_data) time_series_data <-
```

```
na.omit(time_series_data)
```

```
head(time_series_data)
```

```
plot(time_series_data, type = "l", main = "Time Series Data", xlab = "Time",
ylab = "Exchange Rate")
```

```
# Statistical summary
```

```
summary(time_series_data)
```

```
# Histogram and Density Plot
```

```

hist(time_series_data, breaks = 50, main = "Histogram of Exchange Rates", xlab
= "Exchange Rate", col = "lightblue")
lines(density(time_series_data), col = "red", lwd = 2)

# Boxplot
boxplot(time_series_data, main = "Boxplot of Exchange Rates", ylab =
"Exchange Rate")

# Autocorrelation plot
acf(time_series_data, main = "ACF of Exchange Rates")

dataset_min <- min(time_series_data)
dataset_max <- max(time_series_data)

# Time lags
lagged_data <- data.frame(T_minus4 = lag(time_series_data,4),
                          T_minus3 = lag(time_series_data,3),
                          T_minus2 = lag(time_series_data,2),
                          T_minus1 = lag(time_series_data,1),
                          T_pred = time_series_data)

head(lagged_data)

lagged_data <- na.omit(lagged_data)
head(lagged_data) str(lagged_data)

# Lag plot
lag.plot(time_series_data, lags = 4, do.lines = FALSE, main = "Lag Plot of
Exchange Rates")

# Normalisation function normalize
<- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

lagged_data_normalized <- as.data.frame(lapply(lagged_data, normalize))
head(lagged_data_normalized)

# Train test split
training_data <- lagged_data_normalized[1:3800, ]

```

```

testing_data <- lagged_data_normalized[3801:nrow(lagged_data_normalized)-
4, ]
head(training_data)
head(testing_data)
plot(training_data) plot(testing_data)

original_training_data <- time_series_data[1:3800]
head(original_training_data)
original_testing_data <- time_series_data[3801:4751]
head(original_testing_data)

dataset_min <- min(time_series_data) dataset_min
dataset_max <- max(time_series_data) dataset_max

# Denormalisation function denormalize
<- function(x, min, max) { return(
(max - min) * x + min )
}

set.seed(123)

# MLP Model dataset_model1
<- neuralnet(
  formula = T_pred ~ T_minus4 + T_minus3 + T_minus2 + T_minus1,
  data = training_data, hidden = 12, linear.output = TRUE, act.fct =
'logistic'
)
plot(dataset_model1)

# Model predictions
predicted_results_model1 <- predict(dataset_model1, testing_data)
head(predicted_results_model1)

predicted_T <- predicted_results_model1 head(predicted_T)
dim(predicted_T)

target_prediction <- denormalize(predicted_T, dataset_min, dataset_max)
head(target_prediction) dim(target_prediction)

```

```

# Statistical indices rmse_val <- rmse(unlist(original_testing_data),
unlist(target_prediction)) mae_val <- mae(unlist(original_testing_data),
unlist(target_prediction)) mape_val <- mape(unlist(original_testing_data),
unlist(target_prediction)) smape_val <- smape(unlist(original_testing_data),
unlist(target_prediction))

# Output statistics cat("RMSE:",
rmse_val, "\n") cat("MAE:",
mae_val, "\n") cat("MAPE:",
mape_val, "\n")
cat("sMAPE:", smape_val, "\n")

# Model evaluation plot(unlist(original_testing_data),
unlist(target_prediction), xlab = "Desired Output", ylab
= "Predicted Output", main = "Predicted vs. Desired
Output")
abline(0, 1, col = "red")

time_indices <- seq_along(unlist(original_testing_data))

plot(time_indices, unlist(original_testing_data), type = "l", col = "blue",
xlab = "Time", ylab = "Exchange Rate",
main = "Predicted vs. Desired Output")

lines(time_indices, unlist(target_prediction), col = "red")

legend("topleft", legend = c("Desired Output", "Predicted Output"),
col = c("blue", "red"), lty = 1)

```