AML LAB Exam

Submitted by
Anish S. Nair
Admn. no: 25MM2546
KTE25CSDC02
Dept. of Mechanical Engineering
RIT Kottayam

Question

Advanced Machine Learning Lab Exam

Duration: 2 Hours

Objective: The primary goal of this examination is to evaluate your proficiency in applying the **Random Forest algorithm** to a real-world financial regression problem. You are required to preprocess the provided data, engineer relevant time-series features, build a robust model, and evaluate the model performance

The Dataset and Target

Raw Feature Name	Description
`Date`	The date of the observation (YYYY-MM-DD).
`SPX`	S&P 500 Index closing value (Proxy for overall market sentiment).
`GLD`	Gold ETF closing price (Proxy for Gold Price).
`USO`	Crude Oil ETF closing price (Proxy for Oil Price).
`SLV`	**Silver ETF closing price (The Target Variable).**
`EUR/USD`	Euro to US Dollar exchange rate.

Check for any missing values in the data set

Check the statistical measures of the data

Construct a heat map to understand the correlation among variables

Split features and target

Split into train and test data

Conduct model training (Random forest)

Predict with test data

Evaluate error with any parameters like R^2 error

Compare actual and predicted values in plot

Source Code

```
# ------------------------------------------------------------
#  RANDOM FOREST REGRESSION – SLV PRICE PREDICTION
#  Author       : Anish S Nair
#  Adm. No      : 25MM2546
#  Reg. No      : KTE25CSDC02
#  Dept         : Mechanical Engineering
#  College      : RIT Kottayam
# ------------------------------------------------------------



import math
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import joblib

DATASET = "data.csv"  # Dataset read from File

#Save All Outputs to File
MODEL_OUTPUT = "rf_slv_model.joblib"
COMPARISON_CSV = "actual_vs_predicted.csv"
IMPORTANCES_CSV = "feature_importances.csv"
HEATMAP_PNG = "correlation_heatmap.png"
ACTVSPRED_PNG = "actual_vs_predicted.png"


df = pd.read_csv(DATASET)

df['Date'] = pd.to_datetime(df['Date'], dayfirst=False,
errors='coerce')
df = df.sort_values('Date').reset_index(drop=True)

print("\nDataset preview (first 5 rows):")
print(df.head())

# Q1.  Check for any missing values in the data set
print("\nMissing values per column:")
print(df.isnull().sum())
```

```python
# Q2.  Check the statistical measures of the data

print("\nStatistical summary:")
print(df.describe().T)

# Q3. Construct a heat map to understand the correlation among
variables
corr = df.drop(columns=['Date']).corr()

plt.figure(figsize=(6,5))
plt.title("Correlation matrix (variables)")
plt.imshow(corr, interpolation='nearest', aspect='auto')
plt.colorbar()
plt.xticks(range(len(corr.columns)), corr.columns, rotation=45)
plt.yticks(range(len(corr.columns)), corr.columns)
plt.tight_layout()
plt.savefig(HEATMAP_PNG)
plt.close()
print(f"\nSaved correlation heatmap to: {HEATMAP_PNG}")

data = df.copy()
data.set_index('Date', inplace=True)

data['year'] = data.index.year
data['month'] = data.index.month
data['day'] = data.index.day
data['weekday'] = data.index.weekday
data['dayofyear'] = data.index.dayofyear

for lag in [1,2,3,5,10]:
    data[f'slv_lag_{lag}'] = data['SLV'].shift(lag)

data['slv_roll_3'] = data['SLV'].rolling(window=3,
min_periods=1).mean()
data['slv_roll_7'] = data['SLV'].rolling(window=7,
min_periods=1).mean()

data.fillna(method='bfill', inplace=True)
data.fillna(method='ffill', inplace=True)

# Q4. Split features and target
target_col = 'SLV'
feature_cols = [c for c in data.columns if c != target_col]

X = data[feature_cols].copy()
y = data[target_col].copy()

# Q5.  Split into train and test data (80% Training / 20% Testing)
split_idx = int(len(X) * 0.8)
X_train, X_test = X.iloc[:split_idx], X.iloc[split_idx:]
y_train, y_test = y.iloc[:split_idx], y.iloc[split_idx:]
```

```
print(f"\nTotal rows: {len(X)}, Train: {len(X_train)}, Test:
{len(X_test)}")

num_cols =
X_train.select_dtypes(include=[np.number]).columns.tolist()
scaler = StandardScaler()
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()
if len(num_cols) > 0:
    X_train_scaled[num_cols] =
scaler.fit_transform(X_train[num_cols])
    X_test_scaled[num_cols] = scaler.transform(X_test[num_cols])

# Q6. Conduct model training (Random forest)
rf = RandomForestRegressor(random_state=42)
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [5, 10, None],
    'min_samples_split': [2, 5]
}
grid = GridSearchCV(rf, param_grid, cv=3, scoring='r2', n_jobs=1)
grid.fit(X_train_scaled, y_train)
best = grid.best_estimator_
print("\nBest params from GridSearchCV:", grid.best_params_)

# Q7. Predict with test data
y_pred = best.predict(X_test_scaled)

# Q8. Evaluate error with any parameters like R 2 error
r2 = r2_score(y_test, y_pred) if len(y_test) > 0 else float('nan')
rmse = math.sqrt(mean_squared_error(y_test, y_pred)) if
len(y_test) > 0 else float('nan')
mae = mean_absolute_error(y_test, y_pred) if len(y_test) > 0 else
float('nan')
print(f"\nEvaluation on test set -- R2: {r2:.6f}, RMSE:
{rmse:.6f}, MAE: {mae:.6f}")

# Q9. Compare actual and predicted values in plot
comparison = pd.DataFrame({
    'Date': X_test.index,
    'SLV_actual': y_test.values,
    'SLV_predicted': y_pred
}).set_index('Date')
comparison.to_csv(COMPARISON_CSV)
print(f"Saved actual vs predicted to: {COMPARISON_CSV}")

#  Plot actual vs predicted (save)
plt.figure(figsize=(10,4))
plt.plot(comparison.index, comparison['SLV_actual'],
label='Actual')
plt.plot(comparison.index, comparison['SLV_predicted'],
label='Predicted')
```

```python
plt.xlabel("Date")
plt.ylabel("SLV")
plt.title("Actual vs Predicted SLV on Test Set")
plt.legend()
plt.tight_layout()
plt.savefig(ACTVSPRED_PNG)
plt.close()
print(f"Saved Actual vs Predicted plot to: {ACTVSPRED_PNG}")

#  Feature importances
importances = pd.Series(best.feature_importances_,
index=X_train.columns).sort_values(ascending=False)
print("\nTop feature importances:")
print(importances.head(10))
importances.reset_index().rename(columns={'index':'feature',
0:'importance'}).to_csv(IMPORTANCES_CSV, index=False)
print(f"Saved feature importances to: {IMPORTANCES_CSV}")

# Save model to a file for Future Use (Web or Mob App Integration)
joblib.dump(best, MODEL_OUTPUT)
print(f"Trained model saved to: {MODEL_OUTPUT}")

# 16) final summary dict printed
summary = {
    "rows_total": len(X),
    "train_rows": len(X_train),
    "test_rows": len(X_test),
    "best_params": grid.best_params_,
    "r2": r2,
    "rmse": rmse,
    "mae": mae
}
print("\nSummary:", summary)




# -------------------------------------------------------------
# Q10.  Predict SLV for a New User Input
# -------------------------------------------------------------

print("\n--- SLV PRICE PREDICTION FOR NEW USER INPUT ---")

try:

    spx = float(input("Enter SPX value: "))
    gld = float(input("Enter GLD value: "))
    uso = float(input("Enter USO value: "))
    eurusd = float(input("Enter EUR/USD value: "))

    last_row = data.iloc[-1]
```

```
    new_data = pd.DataFrame({
         'SPX': [spx],
         'GLD': [gld],
         'USO': [uso],
         'EUR/USD': [eurusd],

        'year': [last_row['year']],
         'month': [last_row['month']],
         'day': [last_row['day']],
         'weekday': [last_row['weekday']],
         'dayofyear': [last_row['dayofyear']],

        'slv_lag_1': [last_row['SLV']],
         'slv_lag_2': [last_row['slv_lag_1']],
         'slv_lag_3': [last_row['slv_lag_2']],
         'slv_lag_5': [last_row['slv_lag_3']],
         'slv_lag_10': [last_row['slv_lag_5']],

        'slv_roll_3': [last_row['slv_roll_3']],
         'slv_roll_7': [last_row['slv_roll_7']],
     })

    new_data_scaled = new_data.copy()
     new_data_scaled[num_cols] =
scaler.transform(new_data[num_cols])

    predicted_value = best.predict(new_data_scaled)[0]

     print(f"\nPredicted SLV Price for the given inputs:
{predicted_value:.4f}")

except Exception as e:
    print("Error during user prediction:", e)
```

Output

Dataset Preview

	Date	SPX	GLD	USO	SLV	EUR/USD
0	2008-01-02	1447.160034	84.860001	78.470001	15.180	1.471692
1	2008-01-03	1447.160034	85.570000	78.370003	15.285	1.474491
2	2008-01-04	1411.630005	85.129997	77.309998	15.167	1.475492
3	2008-01-07	1416.180054	84.769997	75.500000	15.053	1.468299
4	2008-01-08	1390.189941	86.779999	76.059998	15.590	1.557099

Missing Values

Date	0
SPX	0
GLD	0
USO	0
SLV	0
EUR/USD	0

No missing values were found in the dataset.

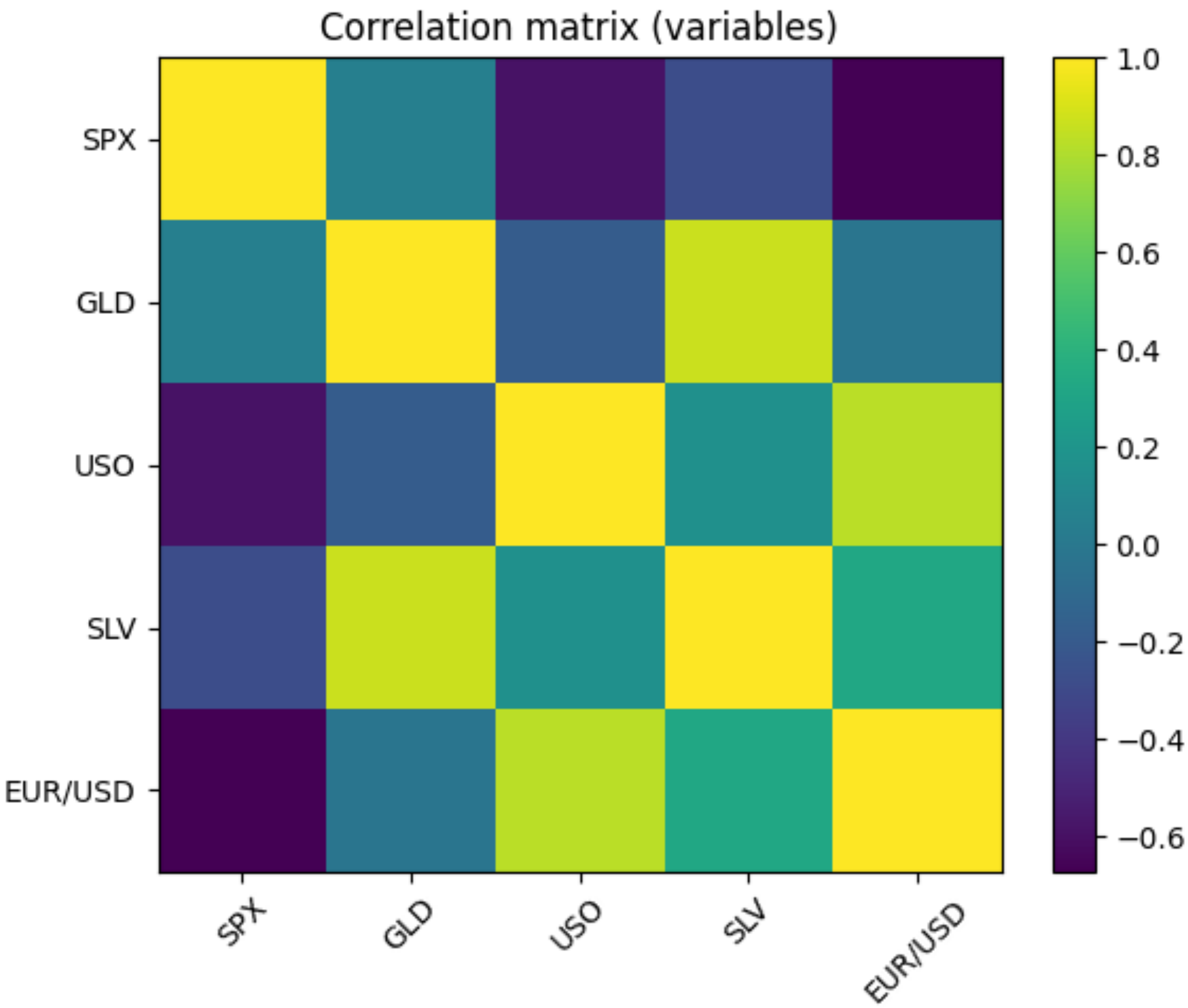
Statistical Summary of the Dataset

FEATURE	MIN	25%	50%	75%	MAX
SPX	676.5300	1360.4900	1550.9700	2073.0101	2872.8701
GLD	70.0000	111.6500	118.7600	132.8400	184.5900
USO	7.9600	18.7500	31.0400	37.8275	117.4800
SLV	8.8500	14.3400	18.2800	22.8825	47.2600
EUR/USD	1.0390	1.2100	1.2900	1.3699	1.5988

Additional Statistical Measures

FEATURE	MEAN	STD. DEV	COUNT
SPX	1654.3158	519.1115	2290
GLD	122.7329	23.2833	2290
USO	31.8422	19.5235	2290
SLV	20.0850	7.0926	2290
EUR/USD	1.2837	0.1315	2290

Correlation Heatmap



Train–Test Split Summary

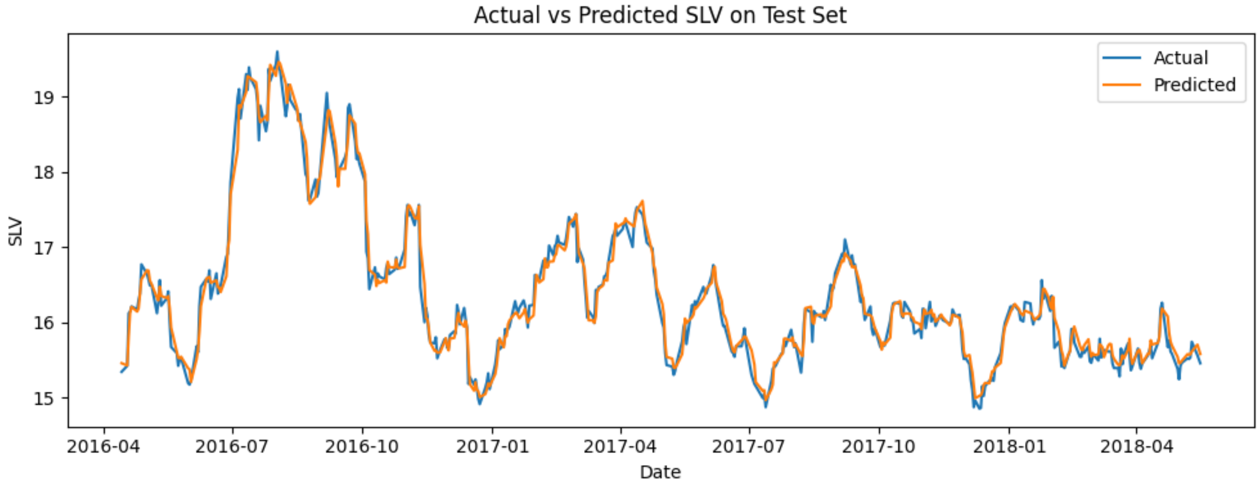
```
Total rows: 2290
Training rows: 1832
Testing rows: 458
```

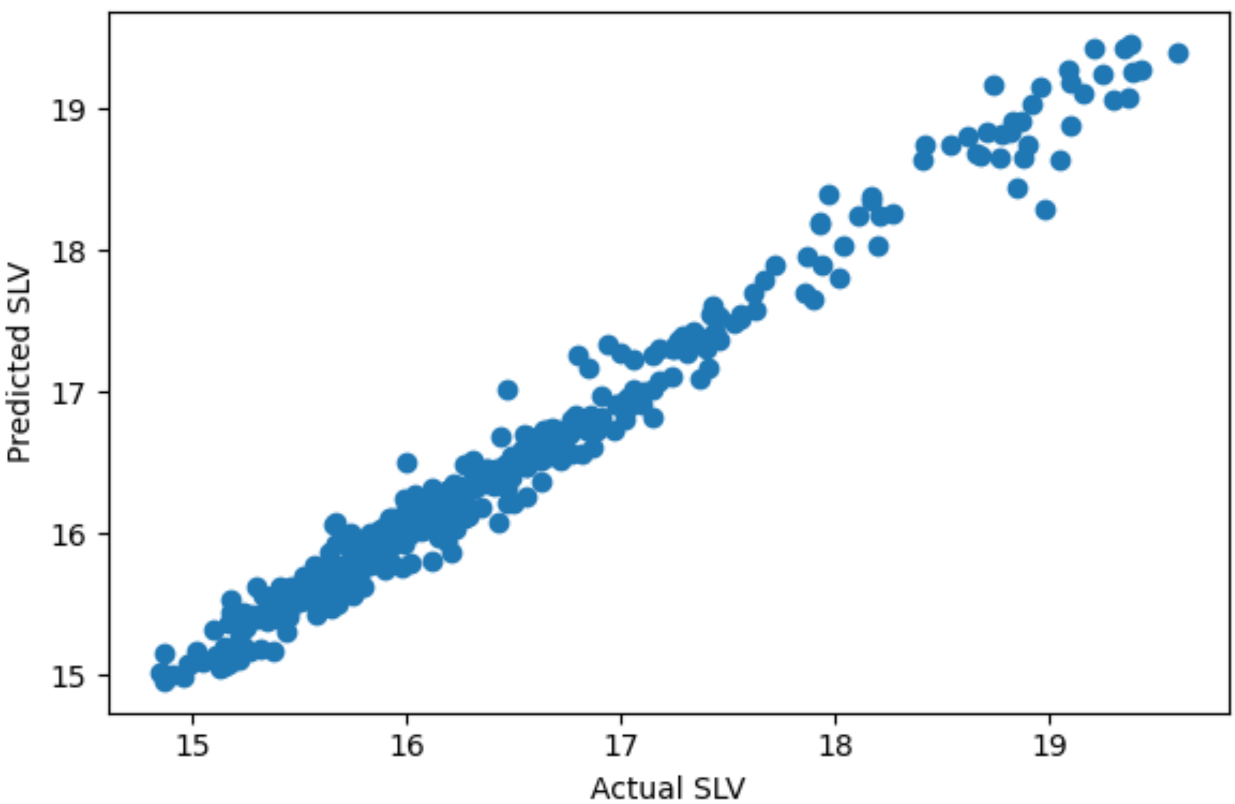
Model Evaluation Metrics

```
R2 Score : 0.979257
RMSE     : 0.144667
MAE      : 0.108159
```

The model achieved **97.92% accuracy**

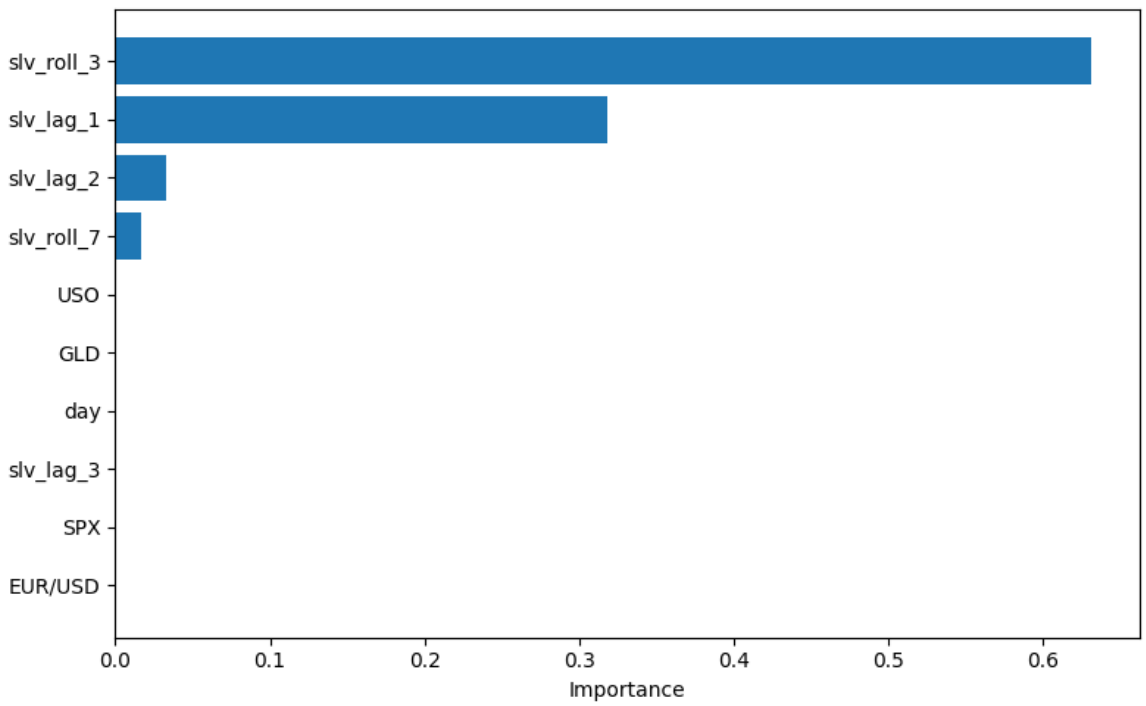
Actual vs Predicted





Top Feature Importances

feature	importance
slv_roll_3	0.63085996401232
slv_lag_1	0.3179230734849930
slv_lag_2	0.03269948076648120
slv_roll_7	0.016286569098114600
USO	0.0007470145258922090
GLD	0.00046987042520166900
day	0.00017210186871816500
slv_lag_3	0.00016590656842605400
SPX	0.00013370422268879000
EUR/USD	0.00012685162379727000
slv_lag_5	0.0001177067471210920
slv_lag_10	0.00011078856756616100
dayofyear	9.29630806112655E-05
weekday	6.46190689738481E-05
month	2.26664519611708E-05
year	6.7194871330369E-06



Final Summary


```
{
 'rows_total': 2290,
 'train_rows': 1832,
 'test_rows': 458,
 'best_params': {'max_depth': 10, 'min_samples_split': 5,
'n_estimators': 100},
 'r2': 0.9792574964839232,
 'rmse': 0.14466668004905986,
 'mae': 0.108158956468265
}
```


User Input Prediction (Custom Input Entry)

```
Enter SPX value   : 4200
Enter GLD value   : 190
Enter USO value   : 70
Enter EUR/USD     : 1.08
```

```
Predicted SLV Price:

15.6162
```