# Gold Project - Random Forest

Group Member: Enzo Goncalves Pereira

```
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split,
RandomizedSearchCV
from sklearn.metrics import mean_squared_error, r2_score,
mean_absolute_percentage_error
import ta

tickers = ['AAPL', 'GOOGL', 'MSFT', 'AMZN', 'META', 'TSLA', 'NFLX',
'NVDA', 'INTC', 'AMD']
all_data = []

df.head()
```


{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1475,\n  \"fields\": [\n    {\n      \"column\": [\n        \"Date\",\n        \"\"\n      ],\n      \"properties\": {\n        \"dtype\": \"date\",\n        \"min\": \"2018-02-20 00:00:00\",\n        \"max\": \"2023-12-28 00:00:00\",\n        \"num_unique_values\": 1475,\n        \"samples\": [\n          \"2021-07-09 00:00:00\",\n          \"2018-11-09 00:00:00\",\n          \"2022-12-30 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n        \"Close\",\n        \"AMD\"\n      ],\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 37.195295920341366,\n        \"min\": 9.529999732971191,\n        \"max\": 161.91000366210938,\n        \"num_unique_values\": 1371,\n        \"samples\": [\n          49.099998474121094,\n          86.0199966430664,\n          29.56999969482422\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n        \"High\",\n        \"AMD\"\n      ],\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 37.945022335224444,\n        \"min\": 9.770000457763672,\n        \"max\": 164.4600067138672,\n        \"num_unique_values\": 1357,\n        \"samples\": [\n          12.180000305175781,\n          63.529998779296875,\n          104.33999633789062\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n        \"Low\",\n        \"AMD\"\n      ],\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 36.42405288273248,\n        \"min\": 9.039999961853027,\n        \"max\": 156.10000610351562,\n

```
\"num_unique_values\": 1364,\n        \"samples\": [\n
44.91999816894531,\n          114.22000122070312,\n
89.83000183105469\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"Open\",\n        \"AMD\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 37.21015635219173,\n
\"min\": 9.079999923706055,\n        \"max\": 163.27999877929688,\n
\"num_unique_values\": 1384,\n        \"samples\": [\n
106.27999877929688,\n          23.889999389648438,\n
18.209999084472656\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"Volume\",\n        \"AMD\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 33823773,\n        \"min\":
16705900,\n        \"max\": 325058400,\n        \"num_unique_values\":
1472,\n        \"samples\": [\n          28526600,\n
85900700,\n          44220100\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": [\n        \"Ticker\",\n        \"\"\n      ],\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n        \"samples\": [\n          \"AMD\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": [\n        \"Return\",\n
\"\"\n      ],\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.03488613297614636,\n        \"min\": -
0.15445372164796256,\n        \"max\": 0.19948052740716316,\n
\"num_unique_values\": 1470,\n        \"samples\": [\n
0.027760246393346177\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"MA5\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 37.07948942101667,\n
\"min\": 9.696000099182129,\n        \"max\": 156.56199951171874,\n
\"num_unique_values\": 1466,\n        \"samples\": [\n
73.5759994506836\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"MA10\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 36.96819252294862,\n
\"min\": 9.767000007629395,\n        \"max\": 154.9550003051758,\n
\"num_unique_values\": 1465,\n        \"samples\": [\n
137.54400024414062\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"Volatility\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.5585104149719968,\n
\"min\": 0.0933809941627774,\n        \"max\": 9.110697152907985,\n
\"num_unique_values\": 1470,\n        \"samples\": [\n
3.381269726866524\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"Volume_Change\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.3569666475579771,\n
\"min\": -0.6812331237534243,\n        \"max\": 3.493063993228108,\n
```

```
\"num_unique_values\": 1475,\n        \"samples\": [\n          -
0.22759587069088205\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"RSI\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 12.424484308987319,\n
\"min\": 23.883045233409703,\n        \"max\": 88.04434947568875,\n
\"num_unique_values\": 1469,\n        \"samples\": [\n
49.243216832169715\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"MACD\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2.6921747614144373,\n
\"min\": -9.137739874598026,\n        \"max\": 10.38178643592832,\n
\"num_unique_values\": 1475,\n        \"samples\": [\n
3.063899806783084\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"MACD_Signal\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2.501166679010129,\n
\"min\": -7.029182310577328,\n        \"max\": 9.57048202746371,\n
\"num_unique_values\": 1475,\n        \"samples\": [\n
2.838123282100581\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"BB_High\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 40.69458465015693,\n
\"min\": 10.448555986873474,\n        \"max\": 166.41669060768362,\n
\"num_unique_values\": 1475,\n        \"samples\": [\n
96.43614531964627\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"BB_Low\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 33.21606997065296,\n
\"min\": 8.948179588500318,\n        \"max\": 139.17766907003724,\n
\"num_unique_values\": 1475,\n        \"samples\": [\n
77.41885422259006\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": [\n
\"Target\",\n        \"\"\n      ],\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 37.22129542218635,\n
\"min\": 9.529999732971191,\n        \"max\": 161.91000366210938,\n
\"num_unique_values\": 1371,\n        \"samples\": [\n
48.599998474121094\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
for ticker in tickers:
    df = yf.download(ticker, start='2018-01-01', end='2023-12-31')
    df['Ticker'] = ticker
    df = df.reset_index()

    df['Return'] = df['Close'].pct_change()
    df['MA5'] = df['Close'].rolling(window=5).mean()
    df['MA10'] = df['Close'].rolling(window=10).mean()
    df['Volatility'] = df['Close'].rolling(window=5).std()
```

```python
    df['Volume_Change'] = df['Volume'].pct_change()

    close_series = df['Close'].squeeze()
    df['RSI'] = ta.momentum.RSIIndicator(close=close_series).rsi()
    macd = ta.trend.MACD(close=close_series)
    df['MACD'] = macd.macd()
    df['MACD_Signal'] = macd.macd_signal()
    boll = ta.volatility.BollingerBands(close=close_series)
    df['BB_High'] = boll.bollinger_hband()
    df['BB_Low'] = boll.bollinger_lband()

    df['Target'] = df['Close'].shift(-1)

    df = df.dropna()
    all_data.append(df)

data = pd.concat(all_data, ignore_index=True)

data.head()
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

{"type":"dataframe","variable_name":"data"}

```python
# We select the features and the target
features = ['Close', 'MA5', 'MA10', 'Volatility', 'Volume_Change',
'RSI', 'MACD', 'MACD_Signal', 'BB_High', 'BB_Low']
X = data[features]
y = data['Target']

# I split my training and test, so I will train my model on 2018-2022
and test on 2023
train_data = data[data['Date'] < '2023-01-01']
test_data = data[data['Date'] >= '2023-01-01']

X_train = train_data[features]
y_train = train_data['Target']
X_test = test_data[features]
y_test = test_data['Target']

param_dist = {
    'n_estimators': [50],
```

```python
    'max_depth': [5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

random_search = RandomizedSearchCV(
    estimator=RandomForestRegressor(random_state=42),
    param_distributions=param_dist,
    n_iter=4,
    cv=2,
    scoring='neg_mean_squared_error',
    n_jobs=-1,
    verbose=1,
    random_state=42
)

random_search.fit(X_train, y_train)
best_model = random_search.best_estimator_
```

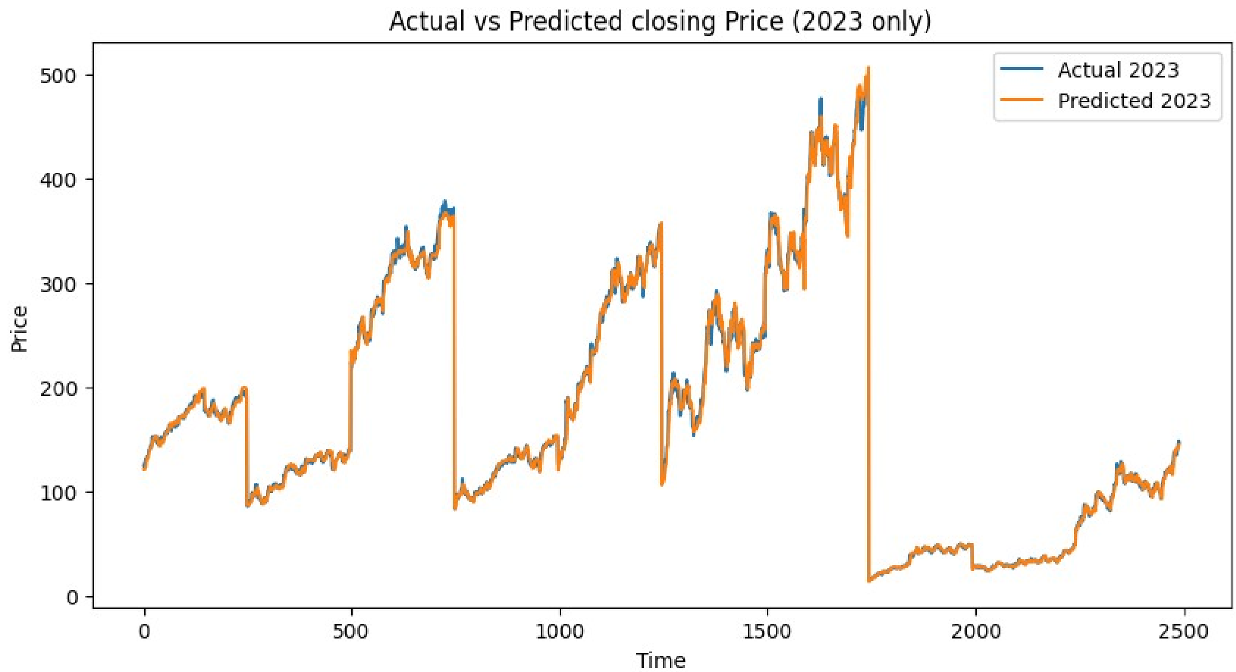
```
Fitting 2 folds for each of 4 candidates, totalling 8 fits
```

```python
# Predict and evaluate
y_pred = best_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)

print("Best params:", random_search.best_params_)
print("Evaluation on 2023 data:")
print(f"RMSE: {rmse:.2f}")
print(f"R² Score: {r2:.2f}")
print(f"mAPE: {mape:.2%}")
```

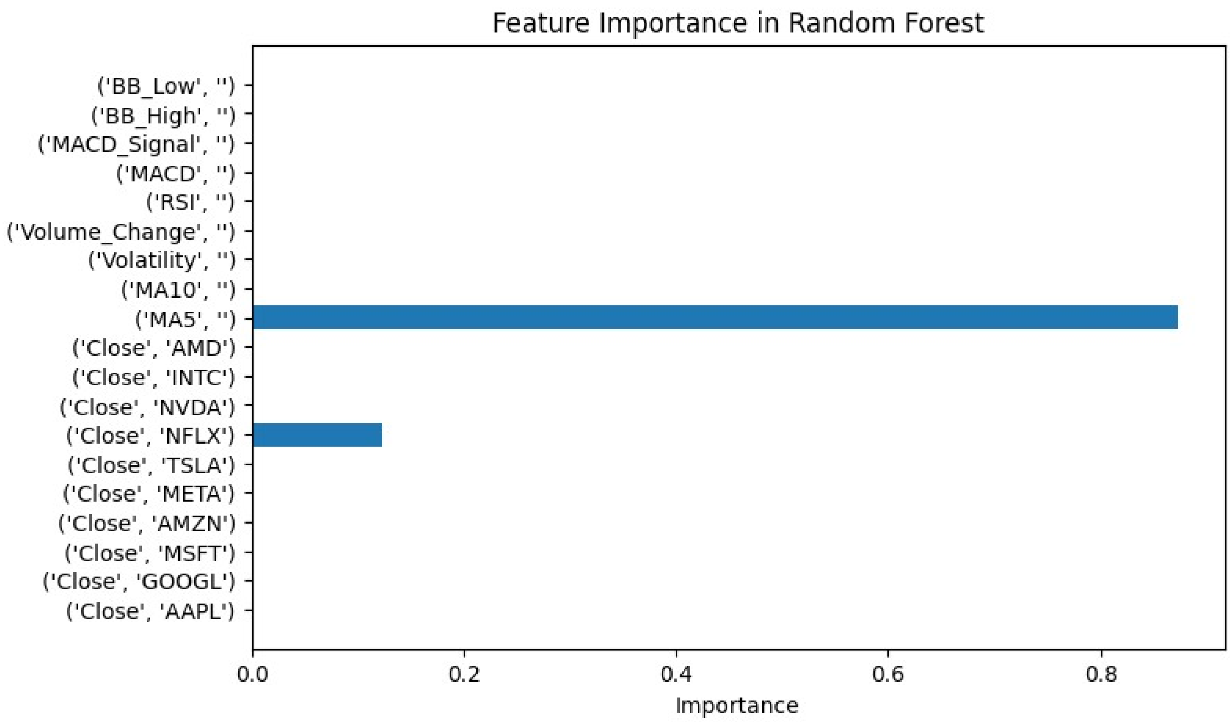
```
Best params: {'n_estimators': 50, 'min_samples_split': 5,
'min_samples_leaf': 1, 'max_depth': 10}
Evaluation on 2023 data:
RMSE: 6.07
R² Score: 1.00
mAPE: 2.37%
```

```python
# Plot predicted vs actual
plt.figure(figsize=(10,5))
plt.plot(y_test.values, label='Actual 2023')
plt.plot(y_pred, label='Predicted 2023')
plt.legend()
plt.title("Actual vs Predicted closing Price (2023 only)")
plt.xlabel("Time")
plt.ylabel("Price")
plt.show()
```




```
# Feature Importance
importances = best_model.feature_importances_
feature_names = [str(col) for col in X.columns]

plt.figure(figsize=(8,5))
plt.barh(feature_names, importances)
plt.xlabel("Importance")
plt.title("Feature Importance in Random Forest")
plt.show()
```


Feature Importance in Random Forest
('BB_Low', '')
('BB_High', '')
('MACD_Signal', '')
('MACD', '')
('RSI', '')
('Volume_Change', '')
('Volatility', '')
('MA10', '')
('MA5', '')
('Close', 'AMD')
('Close', 'INTC')
('Close', 'NVDA')
('Close', 'NFLX')
('Close', 'TSLA')
('Close', 'META')
('Close', 'AMZN')
('Close', 'MSFT')
('Close', 'GOOGL')
('Close', 'AAPL')
0.0
0.2
0.4
0.6
0.8
Importance