Cryptocurrency Trading Model

Group 6

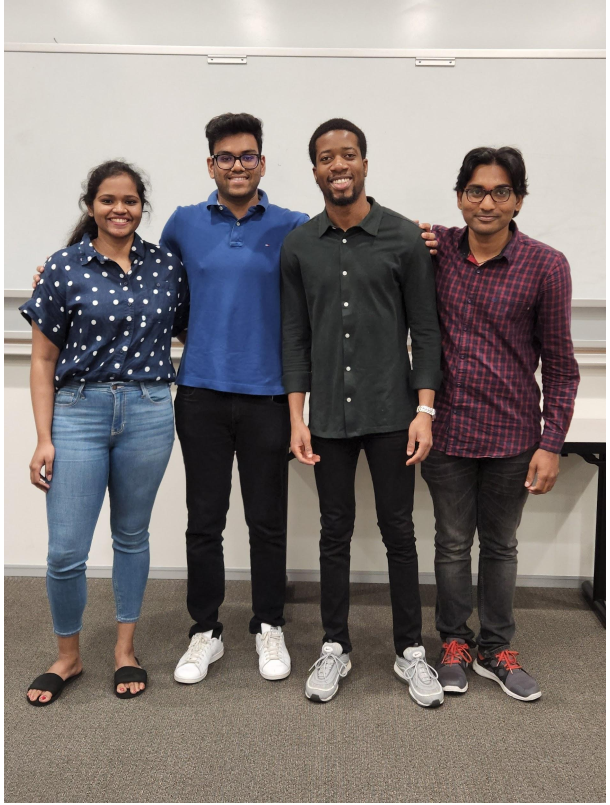
Members

Boluwarin Oluwapelumi Adaramaja

Devarya Raman

Swetha Eragamreddy

Yaswant Reddy Kadiyam



Model Context and Output

Purpose:

- Predict the trading decision to buy/sell/hold any cryptocurrency in the next 4 hours
- 'ATOM-USD', 'BTC-USD', 'DOGE-USD', 'DOT-USD', 'ETH-USD', 'ETC-USD', 'POLY-USD', 'SHIB-USD' pairs were considered for the model
- Train model based on stock indicators as independent variables

Output:

0: Stop Loss Reached first

1: Take Profit Reached first

2: Closed without reaching SL or TP

Model Strategy

- At time t, buy if prediction is 1 and check next 48 records for prediction 0 to sell at that time. Sell after holding period expires

- At time t, sell if prediction is 0 and we have volume of that crypto

- Do nothing if prediction is 2

Summary Statistics of Raw Data

Train data from 2021-06-01 00:00:00 to 2022-02-01 05:05:00

Test data from 2022-02-01 05:05:00 to 2022-04-06 00:00:00

```
[32] sample_df[['low', 'high', 'close', 'volume']].describe()
```

	low	high	close	volume
count	489994.000000	489994.000000	489994.000000	4.899940e+05
mean	6892.275263	6912.178023	6902.246884	3.098728e+09
std	16294.198471	16339.319864	16316.802931	2.377685e+10
min	0.000007	0.000007	0.000007	1.701243e-01
25%	0.492500	0.494300	0.493400	3.526577e+02
50%	31.367000	31.530000	31.450000	2.863190e+03
75%	2368.382500	2379.277500	2373.555000	3.876681e+04
max	68566.000000	69000.000000	68733.070000	1.629530e+12

```
[41] sample_df.coinpair.value_counts()/sample_df.shape[0]*100

ATOM-USD    13.795883
BTC-USD     13.795883
ETH-USD     13.795883
ETC-USD     13.795475
DOGE-USD    13.650983
DOT-USD     12.884443
POLY-USD    10.393393
SHIB-USD     7.888056
Name: coinpair, dtype: float64
```

```
sample_test_df[['low', 'high', 'close', 'volume']].describe()
```

	low	high	close	volume
count	124658.000000	124658.000000	124658.000000	1.246580e+05
mean	5674.262296	5689.606494	5682.059672	1.566566e+09
std	13831.219661	13867.513085	13849.670396	6.726316e+09
min	0.000021	0.000021	0.000021	1.784532e-01
25%	0.153200	0.154100	0.153700	3.260000e+02
50%	24.740000	24.890000	24.820000	2.107380e+03
75%	2521.525000	2529.742500	2525.917500	8.416265e+04
max	48137.200000	48240.000000	48203.430000	2.127464e+11

```
[40] sample_test_df.coinpair.value_counts()/sample_test_df.shape[0]*100

ATOM-USD    12.673074
BTC-USD     12.673074
DOGE-USD    12.673074
DOT-USD     12.673074
ETH-USD     12.673074
SHIB-USD    12.673074
ETC-USD     12.655425
POLY-USD    11.306134
Name: coinpair, dtype: float64
```

Output Variable

Stop Loss

- 1 ATR below entry price

Take Profit

- 1.5 ATR above entry price

Holding Period

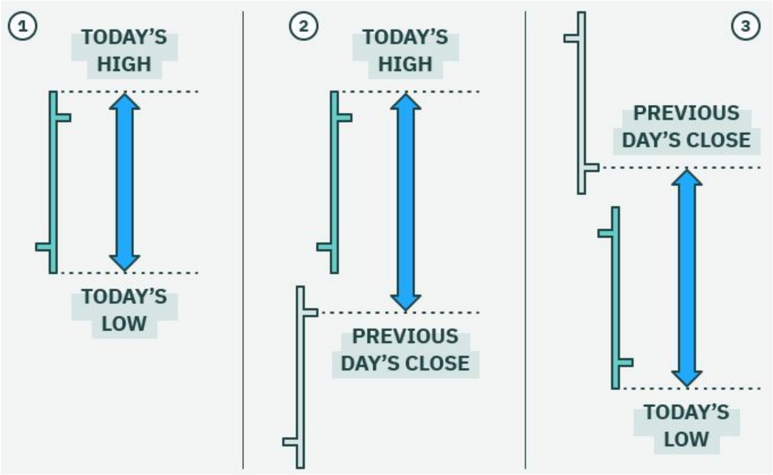
- 48 periods (4 hours)

Y Variable:

0: Stop Loss Reached first

1: Take Profit Reached first

2: Closed without reaching SL or TP



Summary Stats of Output

```
print(sample_test_df['y_4'].value_counts()/sample_test_df.shape[0]*100)
```

```
2.0    63.249852
0.0    29.423703
1.0     7.326445
Name: y_4, dtype: float64
```

```
print(sample_df['y_4'].value_counts()/sample_df.shape[0]*100)
```

```
2.0    67.830831
0.0    23.483757
1.0     8.685412
Name: y_4, dtype: float64
```

Pre-Processing

-One hot encoding : Coin-pair value with values 'ATOM-USD', 'BTC-USD', 'DOGE-USD', 'DOT-USD', 'ETH-USD', 'ETC-USD', 'POLY-USD', 'SHIB-USD'

```
transformer = make_column_transformer(
    (OneHotEncoder(), ['coinpair']),
    remainder='passthrough')
transformed = transformer.fit_transform(train_X)
```

- Normalization : Normalization done on all data in neural network sequence

```
normalizer = tf.keras.layers.Normalization(axis=-1)
normalizer.adapt(train_X)
```

-Outlier treatment : Outliers can be indicators of crypto movement. So, are important for prediction

-Missing value Imputation : There are no missing values in the raw data, but we removed missing values introduced by feature engineering

Final Data Summary Statistics

```
sample_df[['x_rsi','x_ATR', 'obv']].describe()
```

	x_rsi	x_ATR	obv
count	489994.000000	4.899940e+05	4.899940e+05
mean	49.968011	1.992899e+01	5.413829e+11
std	5.952184	4.980956e+01	2.250205e+12
min	10.001939	1.791667e-08	-7.200264e+11
25%	46.290494	2.269271e-03	-1.087676e+07
50%	49.884614	1.388542e-01	-4.884800e+05
75%	53.545492	7.876615e+00	-2.340940e+04
max	88.488519	7.994090e+02	2.275325e+13

```
sample_test_df[['x_rsi','x_ATR', 'obv']].describe()
```

	x_rsi	x_ATR	obv
count	124658.000000	1.246580e+05	1.246580e+05
mean	50.030907	1.533782e+01	-6.647833e+10
std	6.214252	3.991874e+01	3.242473e+11
min	23.928380	3.895833e-08	-1.646573e+12
25%	46.229433	7.854167e-04	-1.419723e+06
50%	50.067299	8.270833e-02	-2.184421e+05
75%	53.795820	4.641823e+00	-8.369863e+03
max	82.597462	3.059490e+02	1.257596e+12

```
for each in ['MA_1_3', 'MA_3_24', 'MA_24_216', 'x_bollinger',
             'MA_3_8', 'MA_8_24', 'MA_24_48', 'coinpair']:
  print(sample_df[each].value_counts()/sample_df.shape[0]*100)
```

```
0.0    71.510467
1.0    28.489533
Name: MA_1_3, dtype: float64
0.0    73.905395
1.0    26.094605
Name: MA_3_24, dtype: float64
0.0    75.315004
1.0    24.684996
Name: MA_24_216, dtype: float64
2.0    86.050237
1.0     7.156822
0.0     6.792940
Name: x_bollinger, dtype: float64
0.0    71.808226
1.0    28.191774
Name: MA_3_8, dtype: float64
0.0    71.8619
1.0    28.1381
Name: MA_8_24, dtype: float64
0.0    70.982094
1.0    29.017906
Name: MA_24_48, dtype: float64
ATOM-USD    13.795883
BTC-USD     13.795883
ETH-USD     13.795883
ETC-USD     13.795475
DOGE-USD    13.650983
DOT-USD     12.884443
POLY-USD    10.393393
SHIB-USD     7.888056
Name: coinpair, dtype: float64
```

```
for each in ['MA_1_3', 'MA_3_24', 'MA_24_216', 'x_bollinger',
             'MA_3_8', 'MA_8_24', 'MA_24_48', 'coinpair']:
  print(sample_test_df[each].value_counts()/sample_test_df.shape[0]*100)
```

```
0.0    71.020713
1.0    28.979287
Name: MA_1_3, dtype: float64
0.0    73.239584
1.0    26.760416
Name: MA_3_24, dtype: float64
0.0    72.637937
1.0    27.362063
Name: MA_24_216, dtype: float64
2.0    83.986587
0.0     8.285068
1.0     7.728345
Name: x_bollinger, dtype: float64
0.0    70.559451
1.0    29.440549
Name: MA_3_8, dtype: float64
0.0    71.0159
1.0    28.9841
Name: MA_8_24, dtype: float64
0.0    71.266184
1.0    28.733816
Name: MA_24_48, dtype: float64
ATOM-USD    12.673074
BTC-USD     12.673074
DOGE-USD    12.673074
DOT-USD     12.673074
ETH-USD     12.673074
SHIB-USD    12.673074
ETC-USD     12.655425
POLY-USD    11.306134
Name: coinpair, dtype: float64
```

Feature Engineering + Selection

Relative Strength Index - 48 observations (4 hours)

rs = up_ma/down_ma, rsi = 100 - (100 / (1 + rs))

Moving averages differences

- Close price between 1hr - 3hr, 3hr - 8hr, 8hr - 24hr, 3hr - 24hr, 24hr - 48hr(2 days), 24hr - 214hr(9 days)

Average True Range

- 48 true ranges from past 48 records (4 hours)

Bollinger Bands

upper_band = sma + 2 * rstd

lower_band = sma - 2 * rstd

- 48 records (4 hours)

Neural Network

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
{'units': 5, 'hidden_layers': 3, 'activation': 'relu'}	0.993234634	0.82319206	0.347238243	0.998551011	0.434314996	0.719306189	0.276933665	1
{'units': 10, 'hidden_layers': 3, 'activation': 'relu'}	0.34656477	0.82319206	0.384912103	0.998551011	0.161584929	0.542960975	0.314671367	7
{'units': 5, 'hidden_layers': 5, 'activation': 'relu'}	0.315656275	0.82319206	0.375503838	0.998551011	0.097216271	0.522023891	0.335349846	8
{'units': 10, 'hidden_layers': 5, 'activation': 'relu'}	0.457035273	0.82319206	0.347238243	0.998520374	0.088940591	0.542985308	0.327990533	6
{'units': 5, 'hidden_layers': 3, 'activation': 'sigmoid'}	0.315656275	0.82319206	0.347238243	0.998551011	0.75601542	0.648130602	0.270612454	3
{'units': 10, 'hidden_layers': 3, 'activation': 'sigmoid'}	0.315656275	0.82319206	0.347238243	0.998551011	0.75601542	0.648130602	0.270612454	3
{'units': 5, 'hidden_layers': 5, 'activation': 'sigmoid'}	0.315656275	0.82319206	0.347238243	0.998551011	0.75601542	0.648130602	0.270612454	3
{'units': 10, 'hidden_layers': 5, 'activation': 'sigmoid'}	0.448014766	0.82319206	0.347238243	0.998551011	0.75601542	0.6746023	0.241726671	2

Neural Network

```
[26] y_preb_probs = loaded_model.predict_proba(test_X)
     roc_auc_score(test_y, y_preb_probs, average="weighted", multi_class="ovr")

     0.9845597975367406

[27] y_preb_probs_train = loaded_model.predict_proba(train_X)
     roc_auc_score(train_y, y_preb_probs_train, average="weighted", multi_class="ovr")

     0.981405747089793

[28] y_pred_test = loaded_model.predict(test_X)
     accuracy_score(test_y, y_pred_test)

     0.925740826902405

[29] y_pred_train = loaded_model.predict(train_X)
     accuracy_score(train_y, y_pred_train)

     0.9110621762715462
```

XGB Model

mean_fit_time	std_fit_time	mean_score_time	std_score_time	params	rank_test_score
36.45749454	0.80301692	0.006001234	9.17E-07	{'n_estimators': 150, 'max_depth': 6, 'learning_rate': 0.001}	1
6.785929155	0.150901172	0.004600048	0.000801866	{'n_estimators': 50, 'max_depth': 4, 'learning_rate': 0.001}	2
19.83760018	0.343146875	0.004201174	0.000399947	{'n_estimators': 150, 'max_depth': 4, 'learning_rate': 0.01}	3
27.21397004	0.9752698	0.004399729	0.000799084	{'n_estimators': 200, 'max_depth': 4, 'learning_rate': 0.01}	4
52.23815255	2.649097523	0.004200506	0.000747551	{'n_estimators': 200, 'max_depth': 6, 'learning_rate': 0.001}	5
21.88410387	0.839022729	0.004002905	0.000633731	{'n_estimators': 150, 'max_depth': 4, 'learning_rate': 0.1}	6
8.587910414	0.201189417	0.00460043	0.002245881	{'n_estimators': 50, 'max_depth': 4, 'learning_rate': 0.1}	7
16.91393595	0.350118859	0.004001188	0.000632862	{'n_estimators': 100, 'max_depth': 4, 'learning_rate': 0.1}	8
34.40803094	0.900780048	0.004404116	0.000488317	{'n_estimators': 200, 'max_depth': 4, 'learning_rate': 0.001}	9
15.69593687	1.969237498	0.004209232	0.000395727	{'n_estimators': 100, 'max_depth': 4, 'learning_rate': 0.001}	10

XGB Model

```
y_preb_probs = random_search.predict_proba(test_X)
roc_auc_score(test_y, y_preb_probs, average="weighted", multi_class="ovr")
```

```
0.9844792416663627
```

```
y_preb_probs_train = random_search.predict_proba(train_X)
roc_auc_score(train_y, y_preb_probs_train, average="weighted", multi_class="ovr")
```

```
0.982827265220068
```

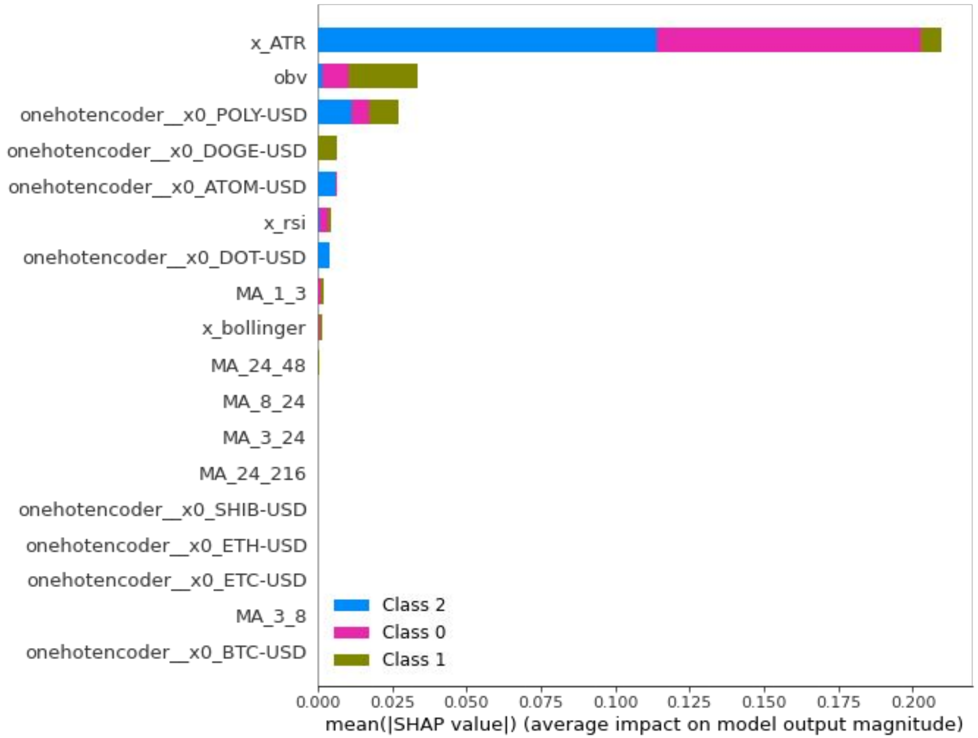
```
y_pred_test = random_search.predict(test_X)
accuracy_score(test_y, y_pred_test)
```

```
0.9132346098926664
```

```
y_pred_train = random_search.predict(train_X)
accuracy_score(train_y, y_pred_train)
```

```
0.9112417703073915
```

SHAP Analysis



Strategy Results

```
ATOM-USD
2.0999999999999996
0.70006
------------------------
BTC-USD
0
0
------------------------
DOGE-USD
0
0
------------------------
DOT-USD
-0.08900000000000041
0.24877000000000002
------------------------
ETC-USD
0.7600000000000016
0.497
------------------------
ETH-USD
0
0
------------------------
POLY-USD
17.28260000000001
1.410015999999999
------------------------
SHIB-USD
0
0
------------------------
```

```
ATOM-USD
0
0
------------------------
BTC-USD
0
0
------------------------
DOGE-USD
-0.06990000000000107
1.0928450000000025
------------------------
DOT-USD
0
0
------------------------
ETC-USD
-16.480000000000018
34.65460000000002
------------------------
ETH-USD
0
0
------------------------
POLY-USD
3.5757000000000656
62.900228999999285
------------------------
SHIB-USD
0
0
------------------------
```

```
ATOM-USD
0
0
------------------------
BTC-USD
0
0
------------------------
DOGE-USD
0
0
------------------------
DOT-USD
0
0
------------------------
ETC-USD
0
0
------------------------
ETH-USD
0
0
------------------------
POLY-USD
0
0
------------------------
SHIB-USD
0
0
------------------------
```

```
ATOM-USD
0
0
------------------------
BTC-USD
0
0
------------------------
DOGE-USD
0
0
------------------------
DOT-USD
0
0
------------------------
ETC-USD
0
0
------------------------
ETH-USD
0
0
------------------------
POLY-USD
0
0
------------------------
SHIB-USD
0
0
------------------------
```

Learnings

- Large datasets take a lot of time to process and model
- Use parallel computing if possible
- Perform EDA on raw data, engineered features and target variable
- Follow all preprocessing steps, overlooking some step may result in inaccurate model
- Handling data imbalance is important
- Do not take metrics to face value. Consider your strategy and see if the model works for your strategy