Project 3:

By: Abdoulaye Kane

Olagoke Michael Kupolati

Executive summary

- Our project’s goal was to build a powerful tool that would perform technical analysis and generate a trade signal using a moving average crossover strategy
- Portfolio Prediction-stocks, cryptos, and futures
- Perform backtesting to evaluate the performance of different strategies using appropriate metrics
- Build a smart contract to get the latest prices for crypto BTC & ETH using provable API and verify via MetaMask if the wallet has a fund to purchase gas
- Crypto Crowdsale (in progress)
- Building users front-end using dApps for the crypto transaction (In progress)

Presentation

- Step 1: Algorithmic Trading
  - Import data
  - Prepare the data
  - Analysis and generate trade signals using average crossover strategy
  - Predict stock, crypto and futures.
  - Create a Logistic Regression Model with the original Data
  - Predict a Logistic Regression Model with Resample Training data
- Step 2: Smart Contract
  - Get crypto BTC and ETH latest prices
  - Crypto Crowdsale.

Algorithmic Trading

Portfolio Optimization & ML PREDICTIVE MODELS with focus on Trade Signals GENERATION using Moving Average (MA) Crossover Strategy (A Python implementation)

INTRODUCTION

Moving averages are commonly used in technical analysis of stocks to predict the future price trends. In this project, we have developed a Python script to generate buy/sell signals using simple moving average (SMA) and exponential moving average (EMA or EWMA) as well as evaluate train test split model validation procedure to reveal how the model performs on new data

Indicators such as Moving averages (MAs), Bollinger bands, Relative Strength Index (RSI) are mathematical technical analysis tools that traders and investors use to analyze the past and anticipate future price trends and patterns. Where fundamental analysts may track economic data, annual reports, or various other measures, quantitative traders and analysts rely on the charts and indicators to help interpret price moves.

The goal when using indicators is to identify trading opportunities and invest in a maximizing way with a view to optimizing the value of the portfolio.

PRIMARY GOALS

Building a powerful tool to perform technical analysis and generate trade signals using moving average crossover strategy.

Portfolio Prediction- stocks, crypto & futures

Performing back testing to evaluate the performance of different strategies using appropriate metrics

Dataframe & SIGNALS - INDEX_SP100.csv

Date	Close	Actual Returns	SMA_Fast	SMA_Slow	SMAFastRatio	SMASlowRatio	Diff	ClosePChge	Signal	Strategy Returns	FutureSlope
2014-03-09 16:00:00	2000.72	-0.000779	0.000627	0.000582	1.018781	1.063769	83.05220	-0.000779	0.0	0.000779	-0.001534
2014-04-09 16:00:00	1997.65	-0.001534	0.000562	0.000550	1.016647	1.061552	83.12025	-0.001534	1.0	0.001534	0.005036
2014-05-09 16:00:00	2007.71	0.005036	0.001568	0.000557	1.020167	1.066305	85.15420	0.005036	0.0	-0.005036	-0.003073
2014-08-09 16:00:00	2001.54	-0.003073	0.001553	0.000558	1.015455	1.062435	87.15975	-0.003073	0.0	-0.003073	-0.006545
2014-09-09 16:00:00	1988.44	-0.006545	0.001004	0.000532	1.007798	1.054920	88.13490	-0.006545	1.0	0.006545	0.003646

Date	Close	Actual Returns	SMA_Fast	SMA_Slow	SMAFastRatio	SMASlowRatio	Diff	ClosePChge	Signal	Strategy Returns	FutureSlope
2023-01-19 16:00:00	3898.85	-0.007638	-0.000943	-0.000814	1.002673	0.981603	-83.46480	-0.007638	1.0	0.007638	0.018918
2023-01-20 16:00:00	3972.61	0.018918	-0.000484	-0.000768	1.022136	1.000942	-82.29625	0.018918	1.0	-0.018918	0.011881
2023-01-23 16:00:00	4019.81	0.011881	0.000252	-0.000637	1.034020	1.013479	-78.79010	0.011881	0.0	0.011881	-0.000711
2023-01-24 16:00:00	4016.95	-0.000711	0.001247	-0.000585	1.031997	1.013351	-71.62110	-0.000711	0.0	-0.000711	-0.000182
2023-01-25 16:00:00	4016.22	-0.000182	0.001684	-0.000610	1.030075	1.013786	-62.64675	-0.000182	1.0	0.000182	0.011008

Dataframe & SIGNALS - INDEX_NASDAQ.csv

Date	Close	Actual Returns	SMA_Fast	SMA_Slow	SMAFastRatio	SMASlowRatio	Diff	ClosePChge	Signal	Strategy Returns	FutureSlope
2013-11-13 16:00:00	3965.58	0.011648	0.002962	0.001177	1.017203	1.132260	396.15630	0.011648	-1.0	0.011648	0.001806
2013-11-14 16:00:00	3972.74	0.001806	0.002175	0.001186	1.016828	1.132961	400.48285	0.001806	1.0	0.001806	0.003330
2013-11-15 16:00:00	3985.97	0.003330	0.001987	0.001151	1.018191	1.135428	404.21250	0.003330	1.0	0.003330	-0.009257
2013-11-18 16:00:00	3949.07	-0.009257	0.001367	0.001165	1.007388	1.123607	405.47500	-0.009257	1.0	-0.009257	-0.004436
2013-11-19 16:00:00	3931.55	-0.004436	0.001403	0.001081	1.001513	1.117414	407.17675	-0.004436	1.0	0.004436	-0.002615

Date	Close	Actual Returns	SMA_Fast	SMA_Slow	SMAFastRatio	SMASlowRatio	Diff	ClosePChge	Signal	Strategy Returns	FutureSlope
2023-01-19 16:00:00	10852.27	-0.009559	-0.001089	-0.001469	1.015194	0.936839	-894.07315	-0.009559	1.0	0.009559	0.026553
2023-01-20 16:00:00	11140.43	0.026553	-0.000435	-0.001464	1.042605	0.963125	-881.76775	0.026553	NaN	-0.026553	0.020105
2023-01-23 16:00:00	11364.41	0.020105	0.000724	-0.001228	1.062797	0.983697	-859.82815	0.020105	NaN	0.020105	-0.002652
2023-01-24 16:00:00	11334.27	-0.002652	0.001959	-0.001106	1.057905	0.982174	-826.10580	-0.002652	NaN	-0.002652	-0.001845
2023-01-25 16:00:00	11313.36	-0.001845	0.002270	-0.001120	1.053562	0.981461	-788.86810	-0.001845	NaN	0.001845	NaN

Dataframe & SIGNALS - INDEX_TSX.csv

Date	Close	Actual Returns	SMA_Fast	SMA_Slow	SMAFastRatio	SMASlowRatio	Diff	ClosePChge	Signal	Strategy Returns	FutureSlope
2013-11-15 16:00:00	13482.57	0.003811	0.001774	0.000272	1.015048	1.064773	620.30240	0.003811	-1.0	0.003811	-0.001818
2013-11-18 16:00:00	13458.06	-0.001818	0.001704	0.000305	1.011479	1.062513	639.07630	-0.001818	-1.0	-0.001818	-0.001136
2013-11-19 16:00:00	13442.77	-0.001136	0.001537	0.000266	1.008779	1.061024	656.15900	-0.001136	1.0	0.001136	-0.000949
2013-11-20 16:00:00	13430.01	-0.000949	0.001419	0.000281	1.006393	1.059719	671.50905	-0.000949	1.0	0.000949	0.003375
2013-11-21 16:00:00	13475.33	0.003375	0.001316	0.000288	1.008462	1.062989	685.41945	0.003375	-1.0	-0.003375	0.000223

Date	Close	Actual Returns	SMA_Fast	SMA_Slow	SMAFastRatio	SMASlowRatio	Diff	ClosePChge	Signal	Strategy Returns	FutureSlope
2023-01-19 16:00:00	20341.44	-0.001707	0.000651	-0.000407	1.028757	1.026968	-34.45960	-0.001707	1.0	0.001707	0.007953
2023-01-20 16:00:00	20503.21	0.007953	0.000971	-0.000399	1.035933	1.035548	-7.35765	0.007953	-1.0	-0.007953	0.006261
2023-01-23 16:00:00	20631.58	0.006261	0.001495	-0.000328	1.040863	1.042374	28.73580	0.006261	-1.0	0.006261	-0.000098
2023-01-24 16:00:00	20629.55	-0.000098	0.002076	-0.000293	1.038604	1.042577	75.68785	-0.000098	1.0	-0.000098	-0.001452
2023-01-25 16:00:00	20599.60	-0.001452	0.002329	-0.000312	1.034687	1.041388	128.11710	-0.001452	NaN	0.001452	NaN

CODE snippets

```
# Set the short window(faster moving average) and long window(shorter moving average)
short_window = 25
long_window = 200

# Generate the fast and slow simple moving averages (25 and 200 days, respectively)
signals_df['SMA_Fast'] = signals_df['Close'].rolling(window=short_window).mean()
signals_df['SMA_Slow'] = signals_df['Close'].rolling(window=long_window).mean()

# Generate the SMAFastRatio, SMALowRatio, Difference,pct_change of SMA_Fast, SMA_Slow and ClosePChge(Closing Price Changes) a

signals_df['SMAFastRatio']=signals_df['Close']/signals_df['SMA_Fast']
signals_df['SMASlowRatio']=signals_df['Close']/signals_df['SMA_Slow']
signals_df['Diff']=signals_df['SMA_Fast']-signals_df['SMA_Slow']
signals_df['SMA_Fast']=signals_df['SMA_Fast'].pct_change()
signals_df['SMA_Slow']=signals_df['SMA_Slow'].pct_change()
signals_df['ClosePChge']=signals_df['Close'].pct_change()

signals_df = signals_df.dropna()

# Review the DataFrame
display(signals_df.head())
display(signals_df.tail())
```

```
# Initialize the new Signal column
signals_df['Signal'] = 0.0

# When Actual Returns are greater than or equal to 0, generate signal to buy stock long
signals_df.loc[(signals_df['Actual Returns'] >= 0), 'Signal'] = 1

# When Actual Returns are less than 0, generate signal to sell stock short
signals_df.loc[(signals_df['Actual Returns'] < 0), 'Signal'] = -1

# Review the DataFrame
display(signals_df.head())
display(signals_df.tail())
```

CODE SNIPPETS

```
for index, row in signals_df.iterrows():
    if row["Actual Returns"] >= 0:
        signals_df.loc[index, "Signal"] = 1.0
    if row["Actual Returns"] < 0:
        signals_df.loc[index,"Signal"] = 0.0
# Calculate the points in time at which a position should be taken, 1 or -1
#ema_signals_df['Crossover']= ema_signals_df['Signal'].diff()
#ema_signals_df=ema_signals_df.dropna()
#ema_signals_df['Crossover']=ema_signals_df['Crossover'].shift(-8)
signals_df["Signal"]=signals_df["Signal"].shift(-1)
signals_df['FutureSlope']=signals_df["Actual Returns"].shift(-1)

'''ema_signals_df['Crossover']=ema_signals_df['Signal'].diff()
# Review the DataFrame
#ema_signals_df=ema_signals_df['Crossover'].dropna()
ema_signals_df=ema_signals_df.dropna()'''
signals_df.tail(10)
```

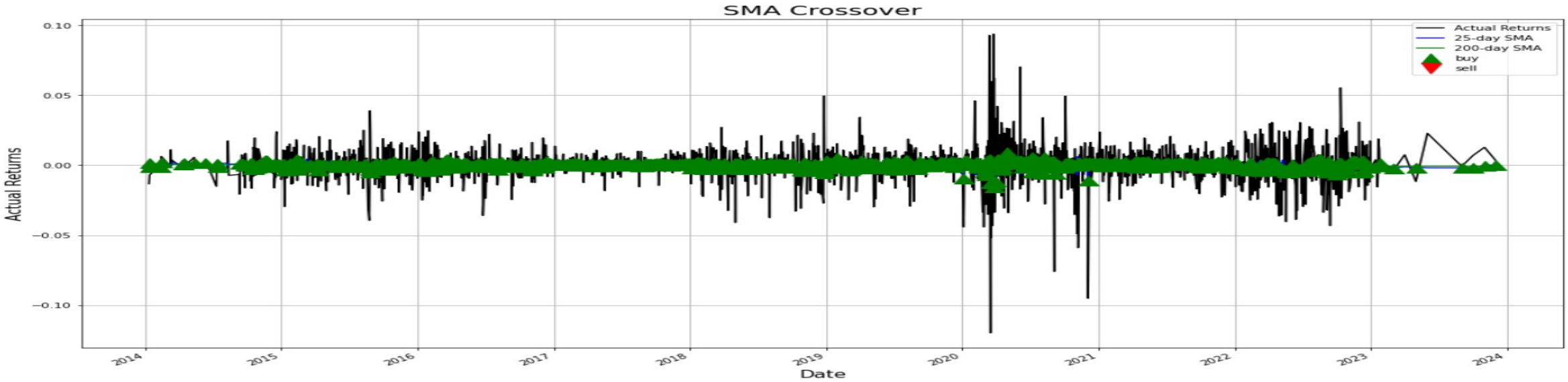
```
#display(df.hvplot.scatter(y='Futureslope',x='Closing Price/SMA_Fast/SMA_Slow/SMAFastRatio/Strategy Returns')

lstColumns=signals_df.columns

display(lstColumns)

signals_dfopt=pd.DataFrame()

for column in lstColumns:
    display(signals_df.hvplot.scatter(y='FutureSlope',x=column))
    dfopt=signals_df[['FutureSlope',column]]
    display(signals_dfopt.corr())
```

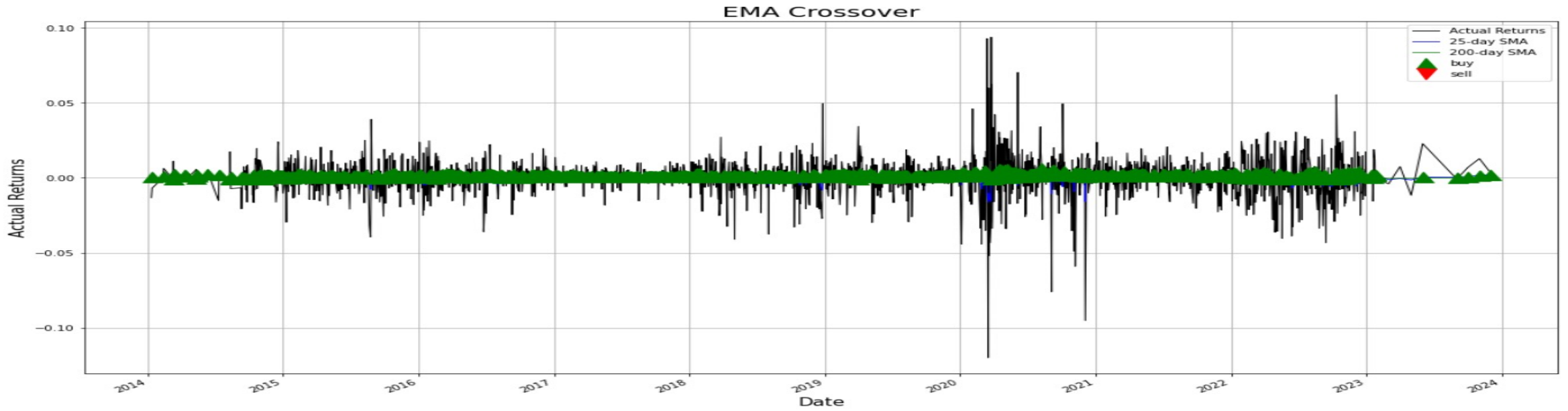
CODE SNIPPETS

```
plt.figure(figsize = (20,10))
# plot close price, short-term and long-term moving averages
signals_df['Actual Returns'].plot(color = 'k', label= 'Actual Returns')
signals_df['SMA_Fast'].plot(color = 'b',label = '25-day SMA')
signals_df['SMA_Slow'].plot(color = 'g', label = '200-day SMA')
# plot 'buy' signals
plt.plot(signals_df[signals_df['Signal'] == 1].index,
         signals_df['SMA_Fast'][signals_df['Signal'] == 1],
         '^', markersize = 15, color = 'g', label = 'buy')
# plot 'sell' signals
plt.plot(signals_df[signals_df['Signal'] == -1].index,
         signals_df['SMA_Fast'][signals_df['Signal'] == -1],
         'v', markersize = 15, color = 'r', label = 'sell')
plt.ylabel('Actual Returns', fontsize = 15 )
plt.xlabel('Date', fontsize = 15 )
plt.title('SMA Crossover', fontsize = 20)
plt.legend()
plt.grid()
plt.show()
```

```
# Create 25 days exponential moving average column
signals_df['SMA_Fast'] = signals_df['Actual Returns'].ewm(span = 25, adjust = False).mean()
# Create 200 days exponential moving average column
signals_df['SMA_Slow'] = signals_df['Actual Returns'].ewm(span = 200, adjust = False).mean()
# create a new column 'Signal' such that if 25-day EMA is greater   # than 200-day EMA then set Signal as 1 else 0
signals_df['Signal'] = 0.0
signals_df['Signal'] = np.where(signals_df['SMA_Fast'] > signals_df['SMA_Slow'], 1.0, 0.0)
# create a new column 'Signal' which is a day-to-day difference of # the 'Signal' column
# ultratech_df['Position'] = ultratech_df['Signal'].diff()
plt.figure(figsize = (20,10))
# plot close price, short-term and long-term moving averages
signals_df['Actual Returns'].plot(color = 'k', lw = 1, label = 'Actual Returns')
signals_df['SMA_Fast'].plot(color = 'b', lw = 1, label = '25-day SMA')
signals_df['SMA_Slow'].plot(color = 'g', lw = 1, label = '200-day SMA')
# plot 'buy' and 'sell' signals
plt.plot(signals_df[signals_df['Signal'] == 1].index,
         signals_df['SMA_Fast'][signals_df['Signal'] == 1],
         '^', markersize = 15, color = 'g', label = 'buy')
plt.plot(signals_df[signals_df['Signal'] == -1].index,
         signals_df['SMA_Fast'][signals_df['Signal'] == -1],
         'v', markersize = 15, color = 'r', label = 'sell')
plt.ylabel('Actual Returns', fontsize = 15 )
plt.xlabel('Date', fontsize = 15 )
plt.title('EMA Crossover', fontsize = 20)
plt.legend()
plt.grid()
```

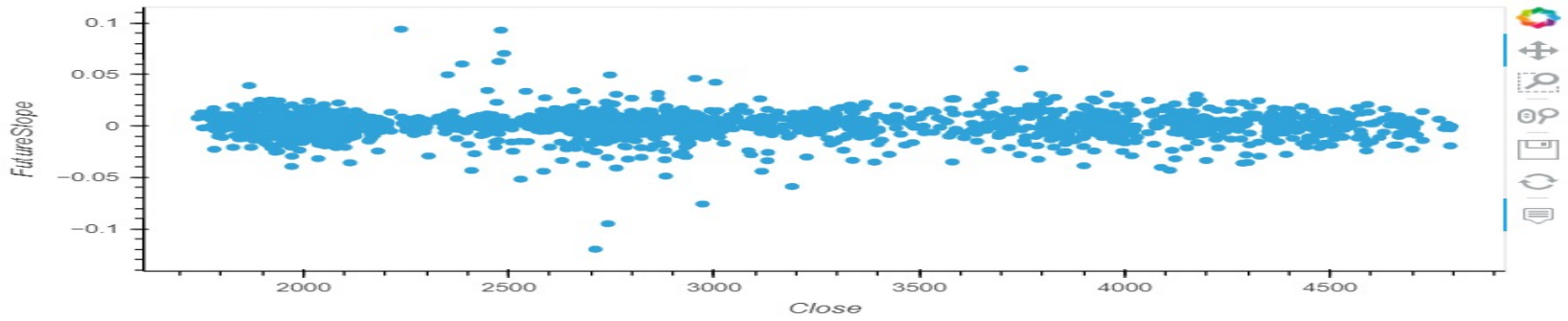
SMA/EMA CROSSOVER –INDEX_SP100

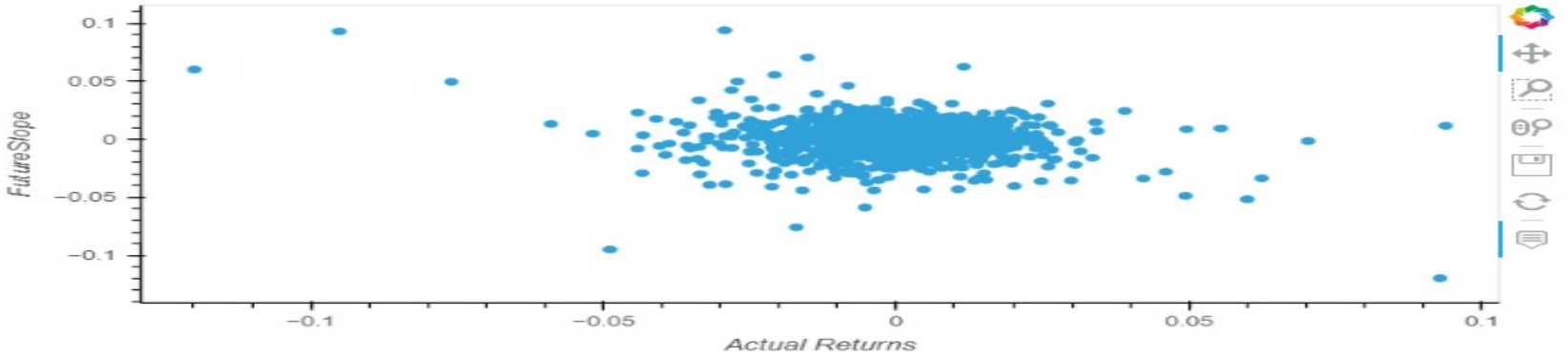


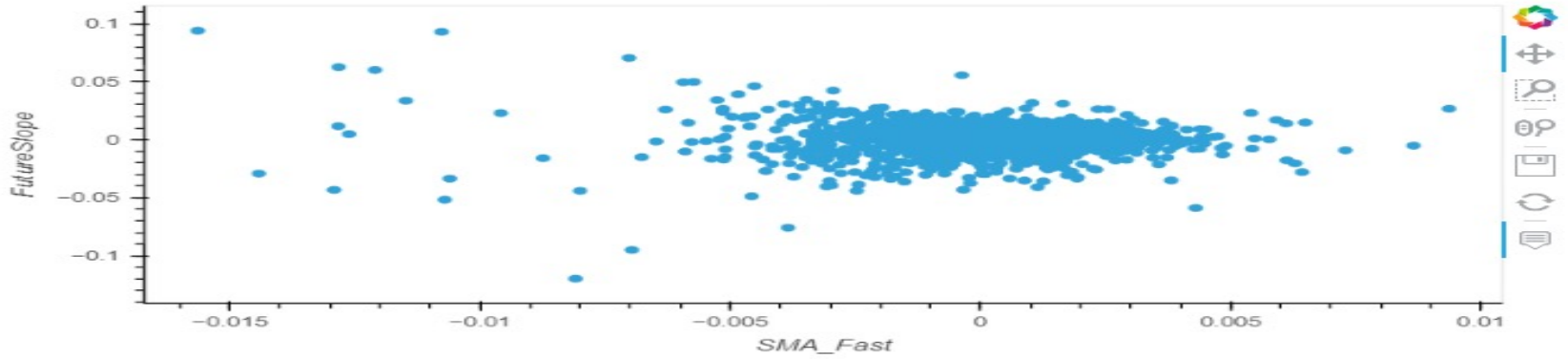


hvplot.scatter - INDEX_SP100

```
Index(['Close', 'Actual Returns', 'SMA_Fast', 'SMA_Slow', 'SMAFastRatio',
       'SMASlowRatio', 'Diff', 'ClosePChge', 'Signal', 'Strategy Returns',
       'FutureSlope'],
      dtype='object')
```







Model Validation Procedure
Train test split

```
y= signals_df['Signal']
```

```
# Generate the X_train and y_train DataFrames
X_train = X.loc[training_begin:training_end]
y_train = y.loc[training_begin:training_end]
```

```
# Generate the X_test and y_test DataFrames
X_test = X.loc[training_end+DateOffset(months=3):]
y_test = y.loc[training_end+DateOffset(months=3):]
```

```
# Scale the features DataFrames

# Create a StandardScaler instance
scaler = StandardScaler()

# Apply the scaler model to fit the X-train data
X_scaler = scaler.fit(X_train)

# Transform the X_train and X_test DataFrames using the X_scaler
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

```
# From SVM, instantiate SVC classifier model instance
svm_model = svm.SVC()

# Fit the model to the data using the training data
svm_model = svm_model.fit(X_train_scaled, y_train)

# Use the testing data to make the model predictions
svm_pred = svm_model.predict(X_test_scaled)

# Review the model's predicted values
svm_pred
```

Support vector machines (SVMs)

Introduction Support vector machines (SVMs) are powerful yet flexible supervised machine learning methods used for classification, regression, and, outliers' detection. SVMs are very efficient in high dimensional spaces and generally are used in classification problems. SVMs are popular and memory efficient because they use a subset of training points in the decision function.

The main goal of SVMs is to divide the datasets into number of classes in order to find a maximum marginal hyperplane (MMH) which can be done in the following two steps −

Support Vector Machines will first generate hyperplanes iteratively that separates the classes in the best way.

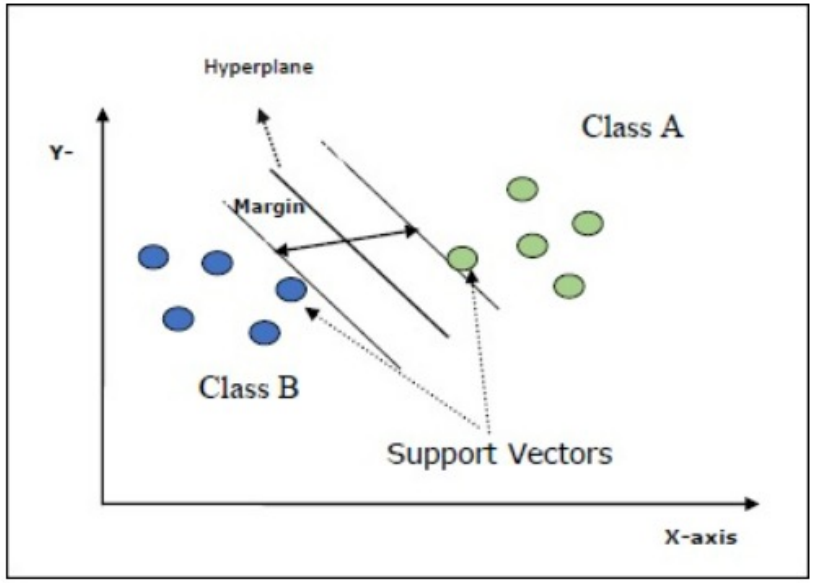
After that it will choose the hyperplane that segregate the classes correctly.

Some important concepts in SVM are as follows −

Support Vectors − They may be defined as the datapoints which are closest to the hyperplane. Support vectors help in deciding the separating line.

Hyperplane − The decision plane or space that divides set of objects having different classes.

Margin − The gap between two lines on the closet data points of different classes is called margin.



Review the classification report associated with the SVC model predictions

```
# From SVM, instantiate SVC classifier model instance
svm_model = svm.SVC(probability=True)
# Fit the model to the data using the training data
svm_model = svm_model.fit(X_train_scaled, y_train)
# Use the testing data to make the model predictions
svm_pred = svm_model.predict(X_test_scaled)

# Review the model's predicted values
svm_pred

# Use a classification report to evaluate the model using the predictions and testing data
svm_testing_report = classification_report(y_test, svm_pred2)

# Print the classification report
print(svm_testing_report)
```

```
y_probs = svm_model.predict_proba(X_test)[:, 1]

precision, recall, thresholds = precision_recall_curve(y_test, y_probs2)

# Plot the precision-recall curve
plt.plot(recall, precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title ('Precision-Recall Curve Line Plot')
#Precision-Recall Curve Line Plot for Logistic Regression Model for Imbalanced Classification
plt.show()

# Find the threshold with the highest F1 score
f1_scores = 2 * (precision * recall) / (precision + recall)
best_threshold_index = np.argmax(precision)
best_threshold = thresholds[best_threshold_index]

# Make predictions with the best threshold
y_pred = (y_probs <= best_threshold).astype(int)
```

svm_testing_report

INDEX_SP100

	precision	recall	f1-score	support
0.0	0.46	1.00	0.63	1066
1.0	0.00	0.00	0.00	1231
accuracy			0.46	2297
macro avg	0.23	0.50	0.32	2297
weighted avg	0.22	0.46	0.29	2297

INDEX_NASDAQ

	precision	recall	f1-score	support
-1.0	0.00	0.00	0.00	1026
1.0	0.55	1.00	0.71	1272
accuracy			0.55	2298
macro avg	0.28	0.50	0.36	2298
weighted avg	0.31	0.55	0.39	2298

INDEX_TSX

	precision	recall	f1-score	support
-1.0	0.45	1.00	0.62	1025
1.0	0.50	0.00	0.00	1263
accuracy			0.45	2288
macro avg	0.47	0.50	0.31	2288
weighted avg	0.48	0.45	0.28	2288

Evaluate a New Machine Learning Classifiers

Using AdaBoost¶

```
# Import a new classifier from SKLearn
from sklearn.ensemble import AdaBoostClassifier

# Initiate the model instance
abc = AdaBoostClassifier()
```

```
# Use a classification report to evaluate the model using the predictions and testing data
abc_testing_report = classification_report(y_test, abc_pred2)

# Print the classification report
print(abc_testing_report)
```

	precision	recall	f1-score	support
0.0	0.47	0.17	0.25	1066
1.0	0.54	0.83	0.65	1231
accuracy			0.52	2297
macro avg	0.50	0.50	0.45	2297
weighted avg	0.50	0.52	0.47	2297

Using Decision Trees

```
# Import a new classifier from SKLearn
from sklearn import tree

# Initiate the model instance
dtc = tree.DecisionTreeClassifier()
```

```
# Use a classification report to evaluate the model using the predictions and testing data
dtc_testing_report = classification_report(y_test, dtc_pred2)

# Print the classification report
print(dtc_testing_report)
```

	precision	recall	f1-score	support
0.0	0.47	0.51	0.49	1066
1.0	0.54	0.50	0.52	1231
accuracy			0.51	2297
macro avg	0.51	0.51	0.51	2297
weighted avg	0.51	0.51	0.51	2297

Using LogisticRegression

```
# Import LogisticRegression from sklearn
from sklearn.linear_model import LogisticRegression

# Create an instance of the LogisticRegression model
lr = LogisticRegression()
```

```
# Use a classification report to evaluate the model using the predictions and testing data
lr_testing_report = classification_report(y_test, lr_pred2)

# Print the classification report
print(lr_testing_report)
```

	precision	recall	f1-score	support
0.0	0.46	0.31	0.37	1066
1.0	0.54	0.69	0.60	1231
accuracy			0.51	2297
macro avg	0.50	0.50	0.49	2297
weighted avg	0.50	0.51	0.50	2297

Performing the Multiple Linear Regression of crypto currencies for optimization

```
Intercept:
 -1.4790124634302246e-09
Coefficients:
 [ 0.87960439 -0.06621048]
                            OLS Regression Results
==============================================================================
Dep. Variable:                    BTC_R   R-squared:                       0.868
Model:                              OLS   Adj. R-squared:                  0.868
Method:                   Least Squares   F-statistic:                 3.312e+06
Date:                  Mon, 10 Oct 2022   Prob (F-statistic):               0.00
Time:                          02:52:22   Log-Likelihood:              1.0751e+07
No. Observations:               1008392   AIC:                         -2.150e+07
Df Residuals:                   1008389   BIC:                         -2.150e+07
Df Model:                             2
Covariance Type:              nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.479e-09   5.65e-09     -0.262      0.793   -1.25e-08    9.59e-09
XRP_R          0.8796      0.000   2539.057      0.000       0.879       0.880
XLM_R         -0.0662      0.000   -206.648      0.000      -0.067      -0.066
==============================================================================
Omnibus:                  5170483.000   Durbin-Watson:                   2.169
Prob(Omnibus):                  0.000   Jarque-Bera (JB):   1021713650335421.500
Skew:                        -234.929   Prob(JB):                         0.00
Kurtosis:                  155941.572   Cond. No.                     6.62e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.62e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The first part shows the output generated by sklearn:

This output includes the intercept and coefficients. This information can be used to build the multiple linear regression equation as follows:

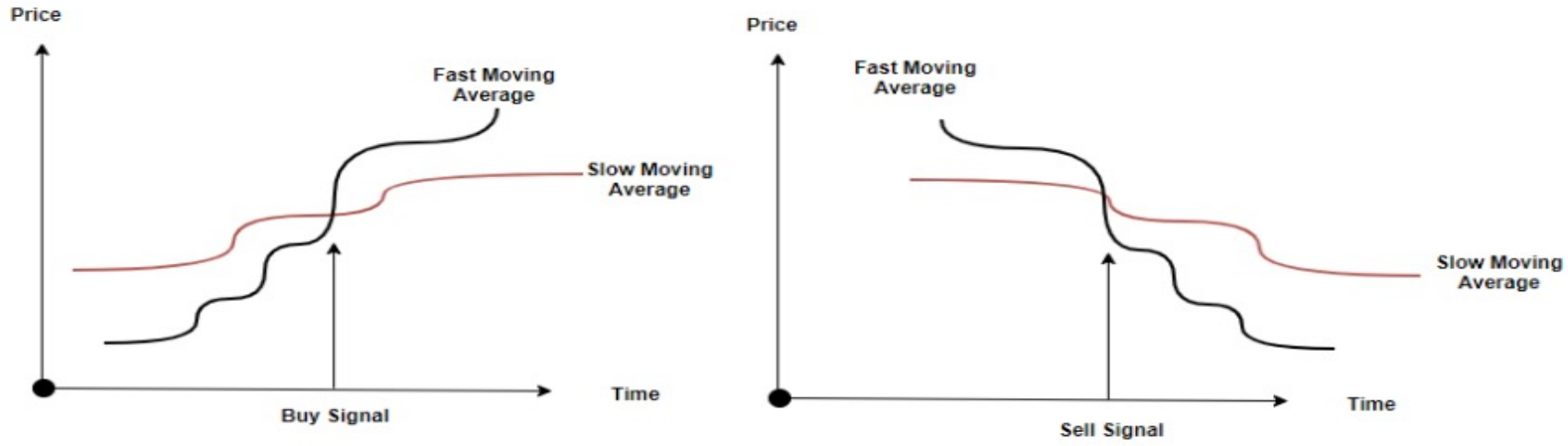
BTC_R = 1.4790124634302246e-09 + 0.87960439*XRP_R + (-0.06621048)*XLM_R

And once the numbers are plugged in, we can determine the return on BITCOIN(BTC_R):

Y = (1.4790124634302246e-09) + (0.87960439)*X1 + (-0.06621048)*X2 Whereas Y is the return on BITCOIN(BTC_R), the X1 & X2 are the returns on XRP and XLM respectively.

findings

Whenever the Fast Average Crosses from below the Slow Average to above it, you buy. Whenever the Fast Average Crosses from above the Slow Average to below, you sell.



r2 = r2_score(y_test, mymodel(X_test)). The result, R-squared, r2= 0.809 shows that the model fits the testing set as well, and we are confident that we can use the model to predict future values.

Best_threshold at 0.757772942247354 i.e threshold that results in the best balance of precision and recall(same as optimizing the F-measure that summarizes the harmonic mean of both measures)

We carried out a portfolio analysis of the combine index, sp100

CONCLUSION

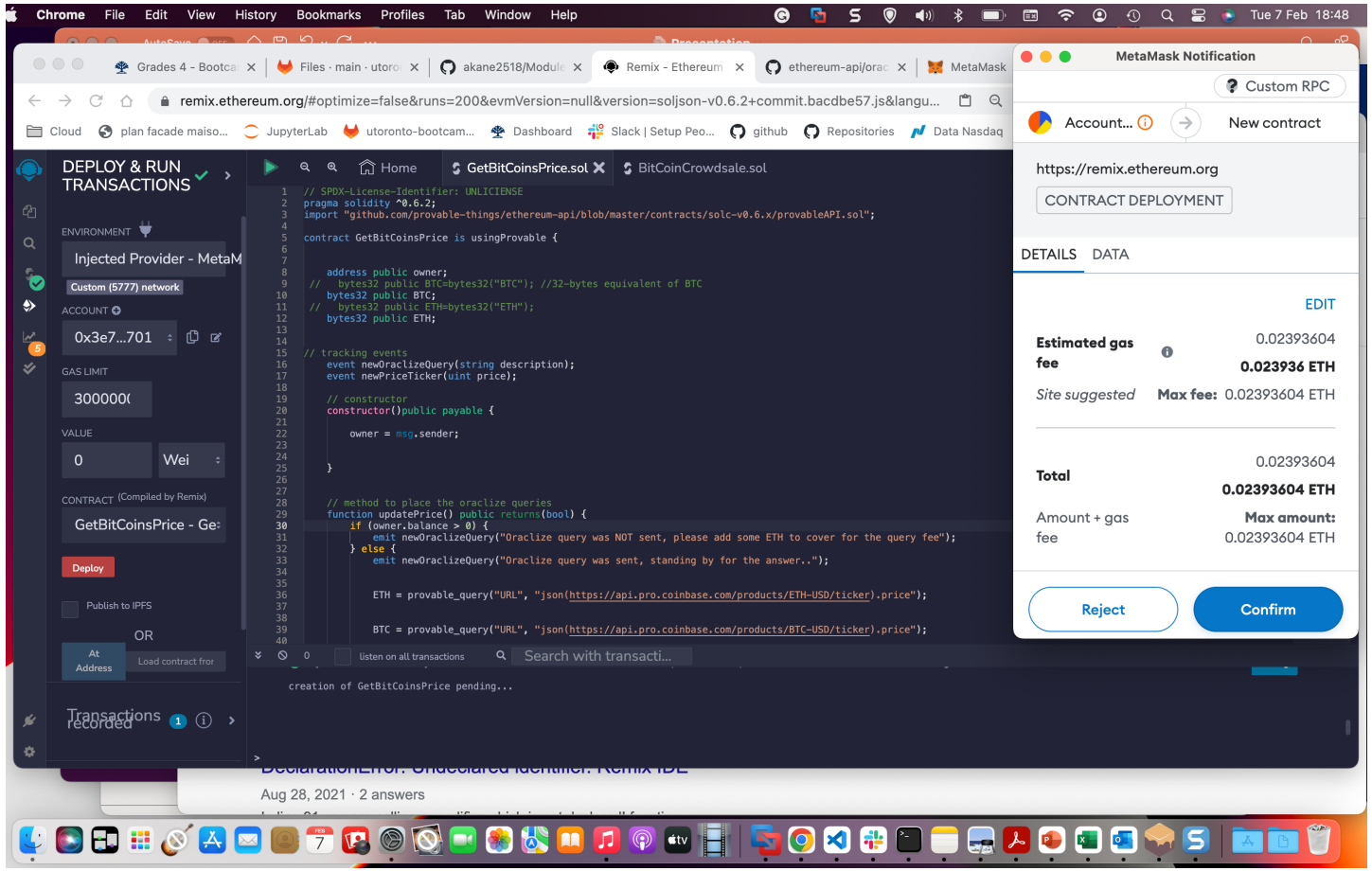
I hope this worksheet has provided a framework to test more (and hopefully better) ideas utilizing ML Models.

FUTURE WORK

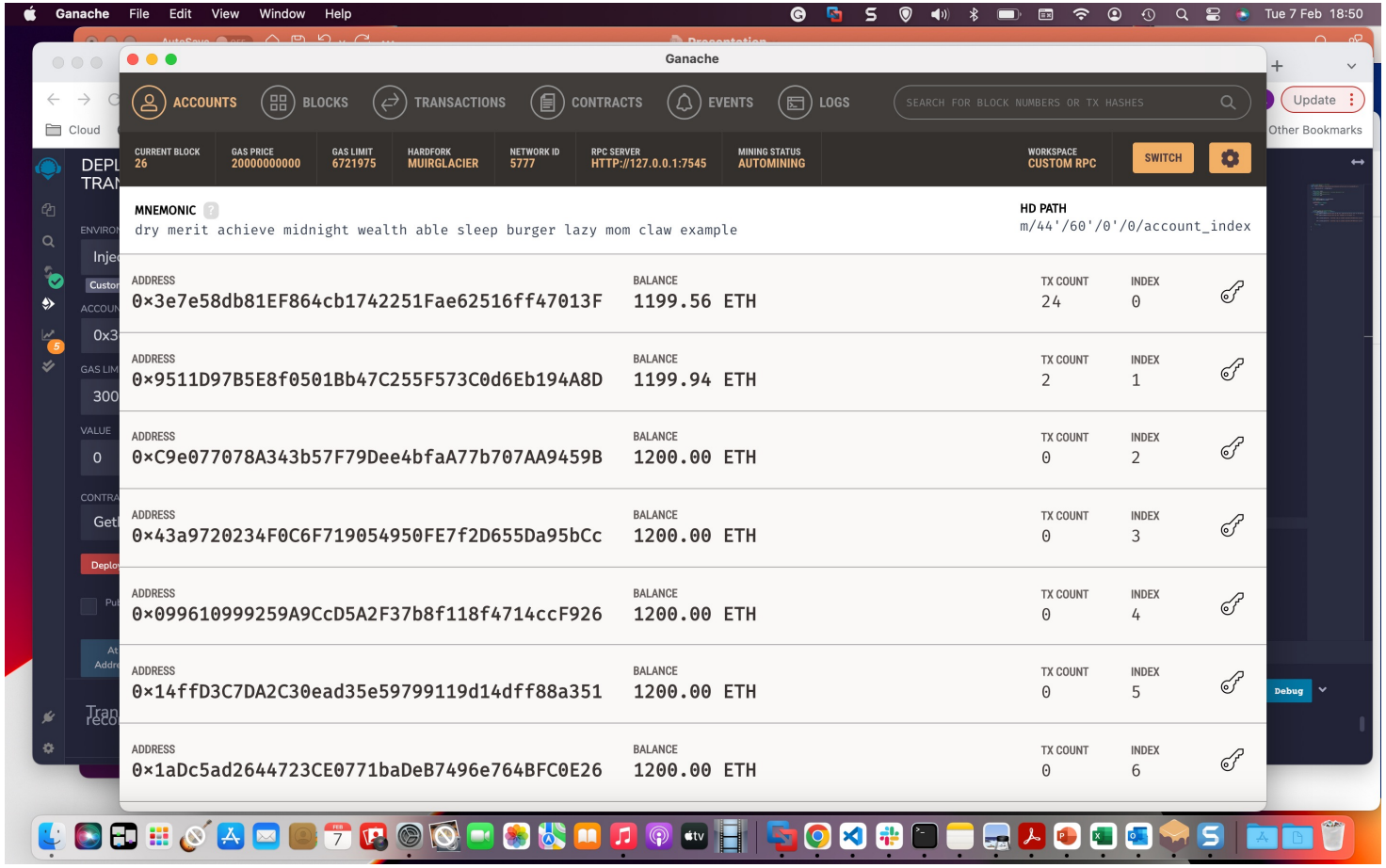
- Can test over different granularities such as seconds, hours, days & months or see how it does with intraday trading.
- Make a more robust model that possibly forecast movement vs. trying to capture a very noise next day indicator.
- Try to predict a target further in the future than a day.
- Find a better Slow/Fast Window optimization method that could consider a region of values vs. just the max value.
- Run this strategy over a portfolio to have less dead time for our money.

Smart Contract

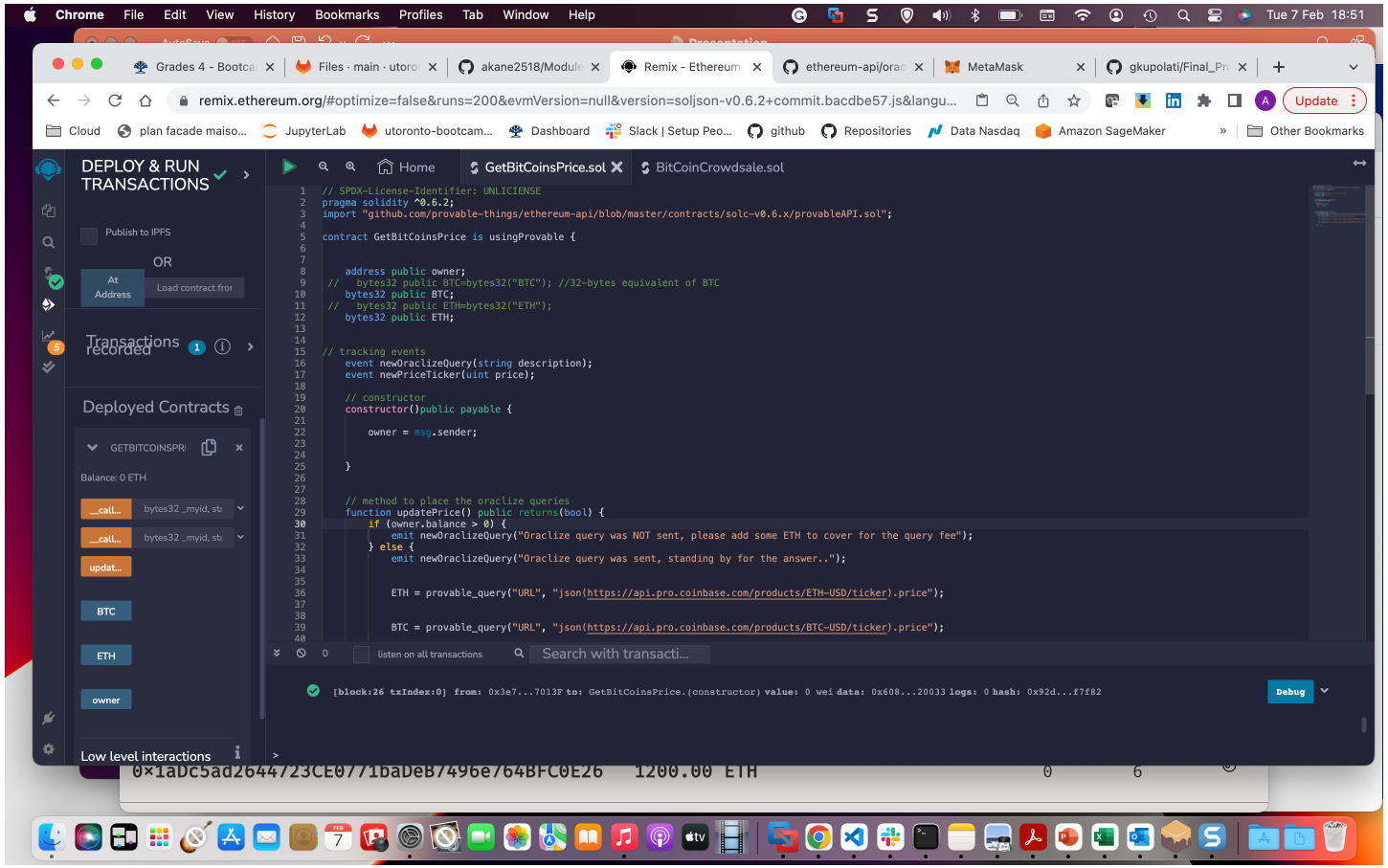
Get Bit Coins Prices



Wallet in Ganache was debited



Display Crypto (BTC and ETH) and additional functions available.



MEET OUR TEAM

Olagoke Michael Kupolati

Chief Executive Officer

Abdoulaye Kane

President

THANK YOU

Olagoke Michael Kupolati

gkupolati@gmail.com

https://github.com/gkupolati/Final_Project_AlgorithmicTrading_SmartContract.git