**Final Report: Blockchain Capstone Project**

**By Jimmy Cheng**

**Problem:**

How can I create a multi-factor model to predict future ETH and BTC prices over certain economic regimes given historical prices, blockchain metrics and economic indicators. The model’s performance will be measured by the accuracy of the prediction with a margin of error of within 15%.

**Data:**

We will use CoinMetrics Python API to collect 5 years of historical daily metrics data on BTC and ETH. Including fundamental metrics on the blockchains will create a multifactor model that is not simply limited to time series analysis on Price only. The model can also be further improved by joining data from traditional financial assets (Yahoo Finance) and Macro economic data (FRED StLouis FED). The goal is to create a multi-factor model for predicting future BTC and ETH prices.

CoinMetrics is founded in 2017 and is becoming the leading platform for analyzing and monitoring crypto networks and assets.

CoinMetrics python API: <https://pypi.org/project/coinmetrics-api-client/>

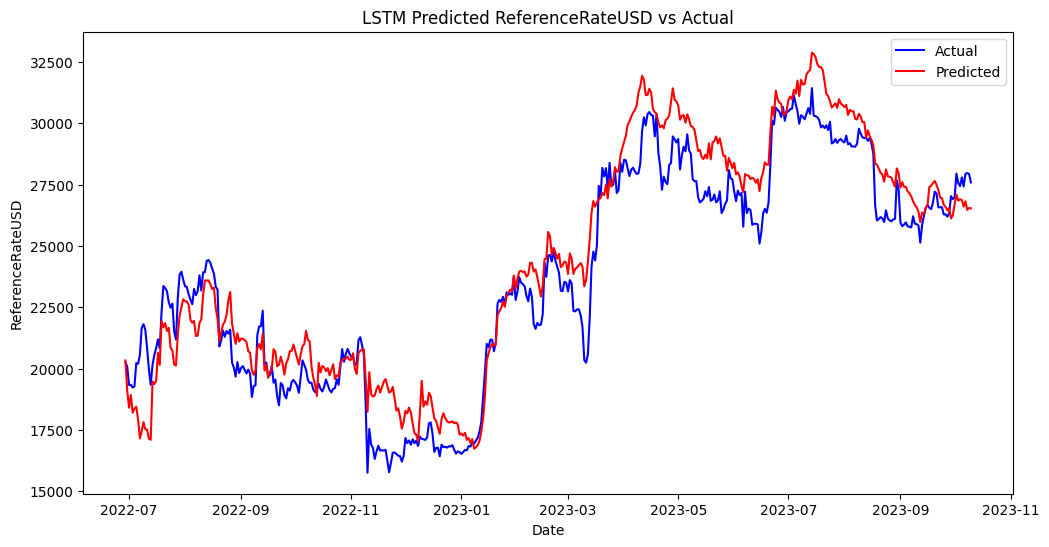
https://github.com/coinmetrics/api-client-python/tree/master/docs

US Economic data: <https://fred.stlouisfed.org/>

Yahoo Finance stock market data: <https://pypi.org/project/yfinance/>

**Summary of Findings:**

I achieved a model that predicts within a reasonable margin of error using a combination of XGBoost and LSTM neural network. I started by examining creating a percentage change series of the target variable (price) and checking whether a simple moving average or ARIMA model could show me anything about the series. However, a 1 factor ARIMA model based on price only showed that the problem at hand was much greater than that. Using XGBoost, I was able to convert the problem into a time-invariant problem and determined the most important features that affect percentage change in price over the training data set time frame. I then used the top 6 features that explained the most variance and build a 1-layer LSTM neural network. The final model uses a sequence length of 40, 30 neurons, ‘relu’ activation function, ‘adam’ optimizer, batch\_size = 7. The model is trained over first 80% of the dataset and tested over the final 20% and offered a directional accuracy of 70% on predicting daily percentage changes. By taking the integral of daily percentage changes, I am able to create a prediction series of bitcoin prices to within 15% accuracy of the actual prices.



**Data Wrangling:**

Please refer to the Asset Metrics file on github here (<https://github.com/jcheng93/Blockchain-Capstone-Project/blob/6052a3935447d9f64387c330ae7c08c4b6a332b8/Asset%20Metrics>) for the complete data series loaded for this project.

Data is loaded using public APIs, converted from string to float, and NA values are filled with previous day data. Forward fill is being used because future data will not be known. Blockchain data exists daily, stock market indices data are only on business days and Macro data only updates every month/ quarter. We use the blockchain index as the uniform index across all data sources and use forward fill on dates where there are no updates on stock market and macro data.

After cleaning and combining data from all APIs, the data is stored into a Pandas Dataframe and looks like this with 2472 daily observations across 30 different features.

<class 'pandas.core.frame.DataFrame'>

Index: 2472 entries, 2017-01-03 to 2023-10-10

Data columns (total 31 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 AdrActCnt 2472 non-null int64

1 AdrBalNtv0.1Cnt 2472 non-null int64

2 CapMVRVCur 2472 non-null float64

3 DiffLast 2472 non-null float64

4 FeeMedNtv 2472 non-null float64

5 FeeTotNtv 2472 non-null float64

6 FlowInExNtv 2472 non-null float64

7 FlowOutExNtv 2472 non-null float64

8 HashRate 2472 non-null float64

9 ReferenceRateUSD 2472 non-null float64

10 SplyFF 2472 non-null float64

11 SplyMiner1HopAllNtv 2472 non-null float64

12 TxCnt 2472 non-null int64

13 TxTfrValAdjNtv 2472 non-null float64

14 Volume 2472 non-null int64

15 DBC 2472 non-null float64

16 GLD 2472 non-null float64

17 USO 2472 non-null float64

18 UUP 2472 non-null float64

19 IXIC 2472 non-null float64

20 SPX 2472 non-null float64

21 2Y Yield 2472 non-null float64

22 10Y Yield 2472 non-null float64

23 M2 YoY 2472 non-null float64

24 M2 Velocity 2472 non-null float64

25 Real PCE YoY 2472 non-null float64

26 NonFarm Payrolls YoY 2472 non-null float64

27 Industrial Production Index YoY 2472 non-null float64

28 Headline CPI YoY 2472 non-null float64

29 Real GDP YoY 2472 non-null float64

30 Unemployment Rate 2472 non-null float64

dtypes: float64(27), int64(4)

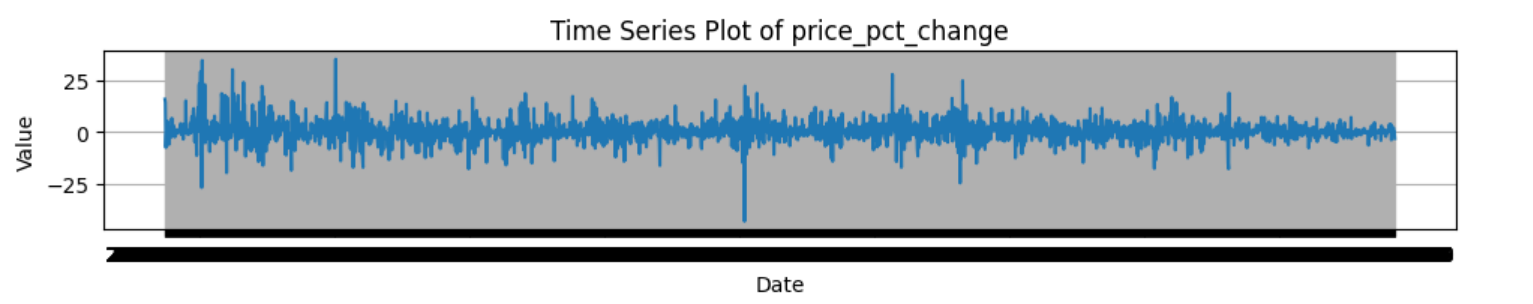
memory usage: 618.0+ KB

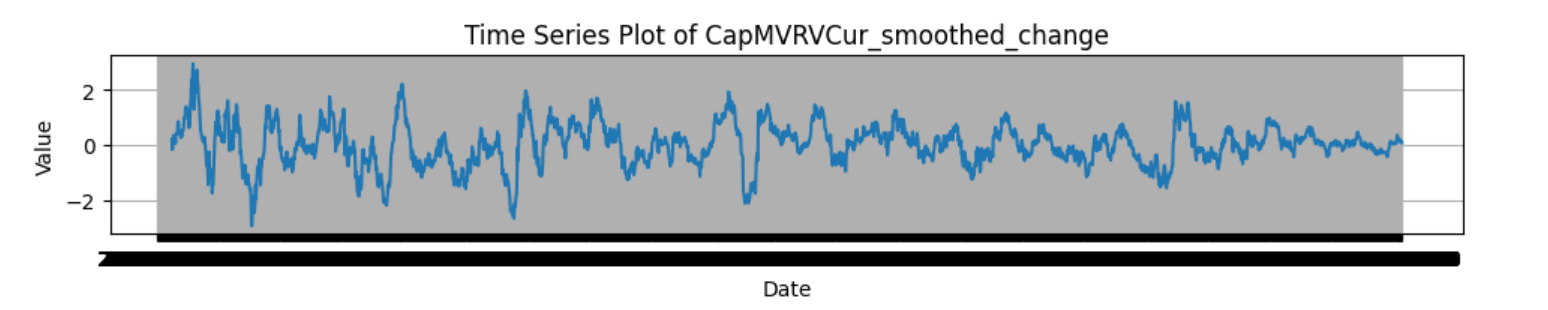
None

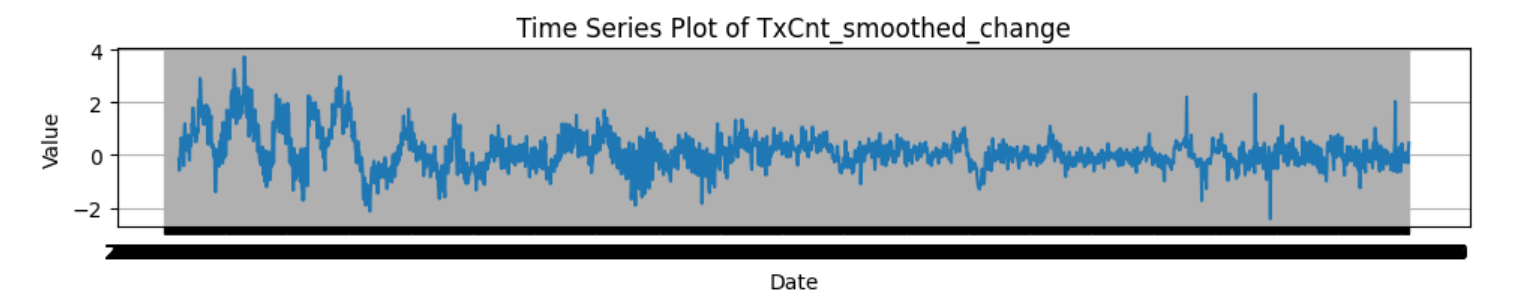
An important step in time series analysis to look for trends and create stationary variables. We used the ADFULLER test to check which columns are non-stationary, and then transform the non-stationary variables to stationary variables using the following techniques:

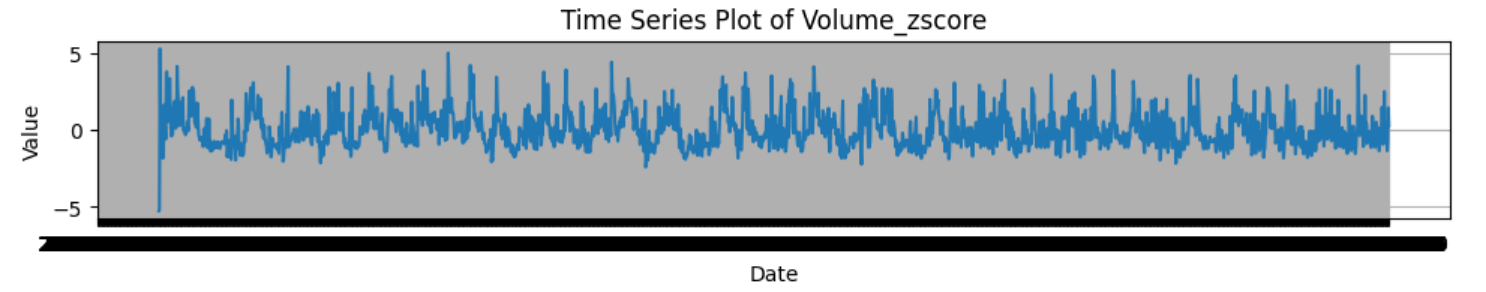
1. Day over day percentage change
2. 30 Day Z-score (series value – rolling mean of series) / (rolling std of series)
3. Month over month rate of change (used for Macro data series which are updated every month)

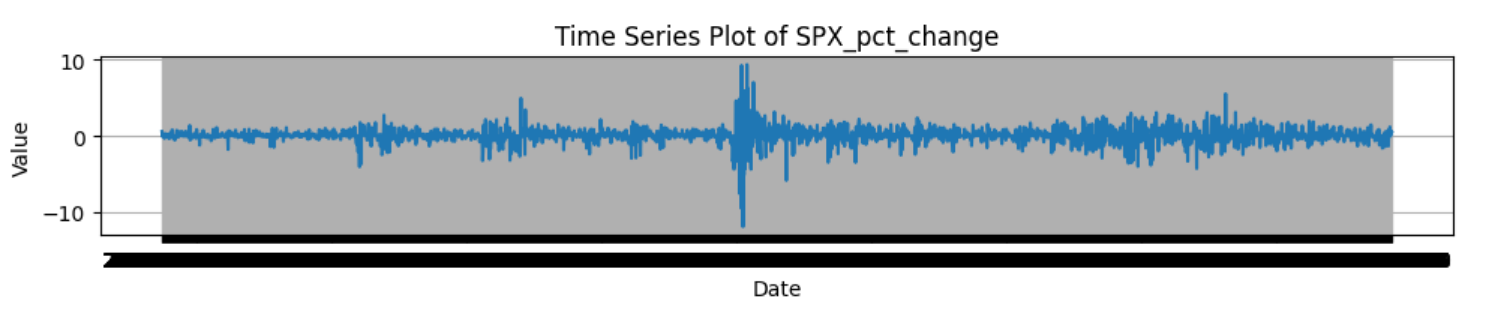
We created over 30 more stationary variables for the dataset which will be used as features later for machine learning. Below are some plots of the transformed variables.

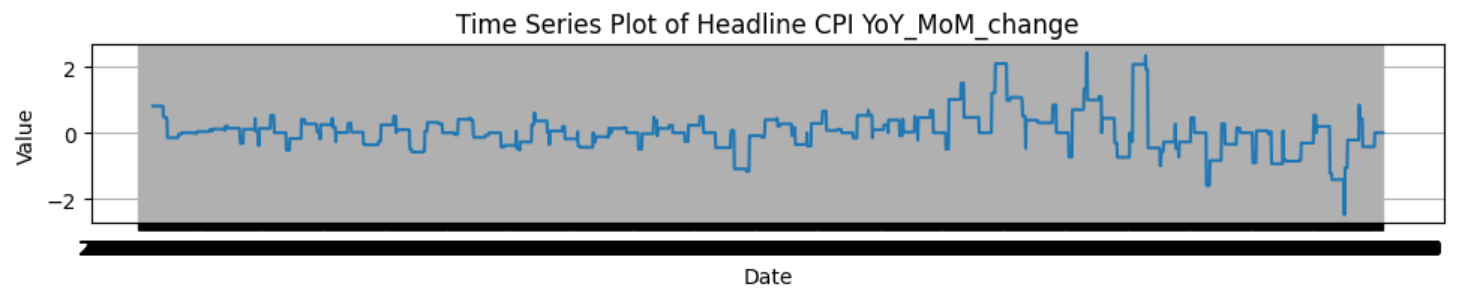






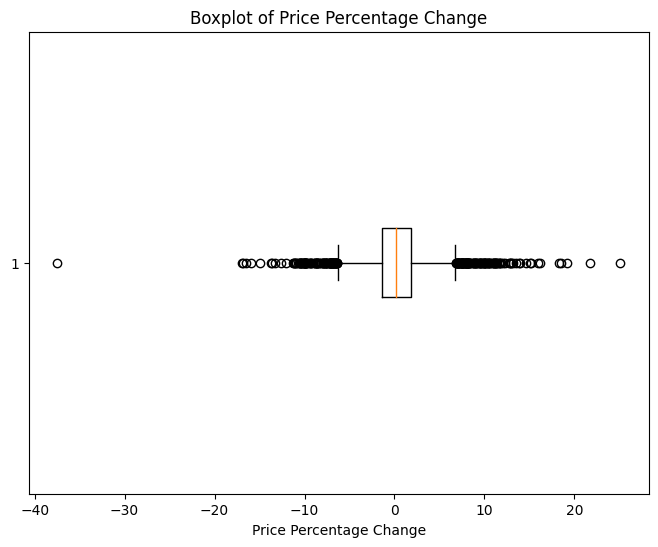




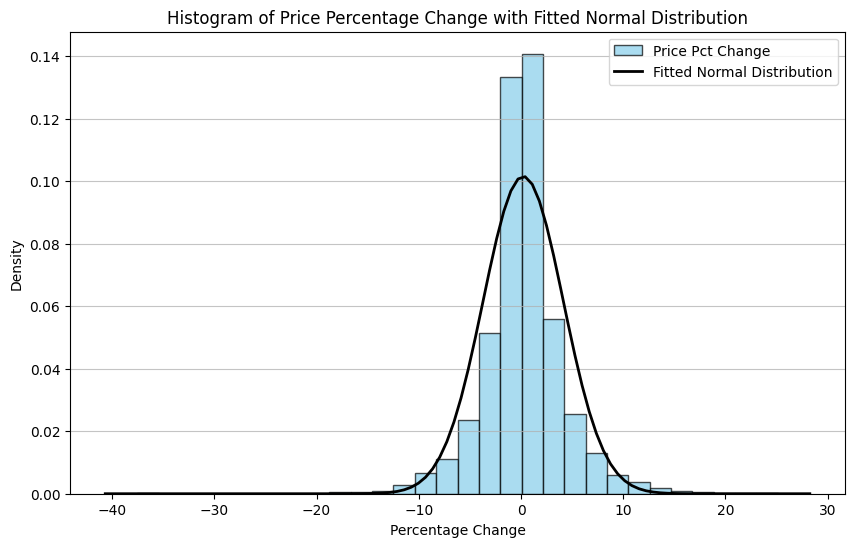


**EDA:**

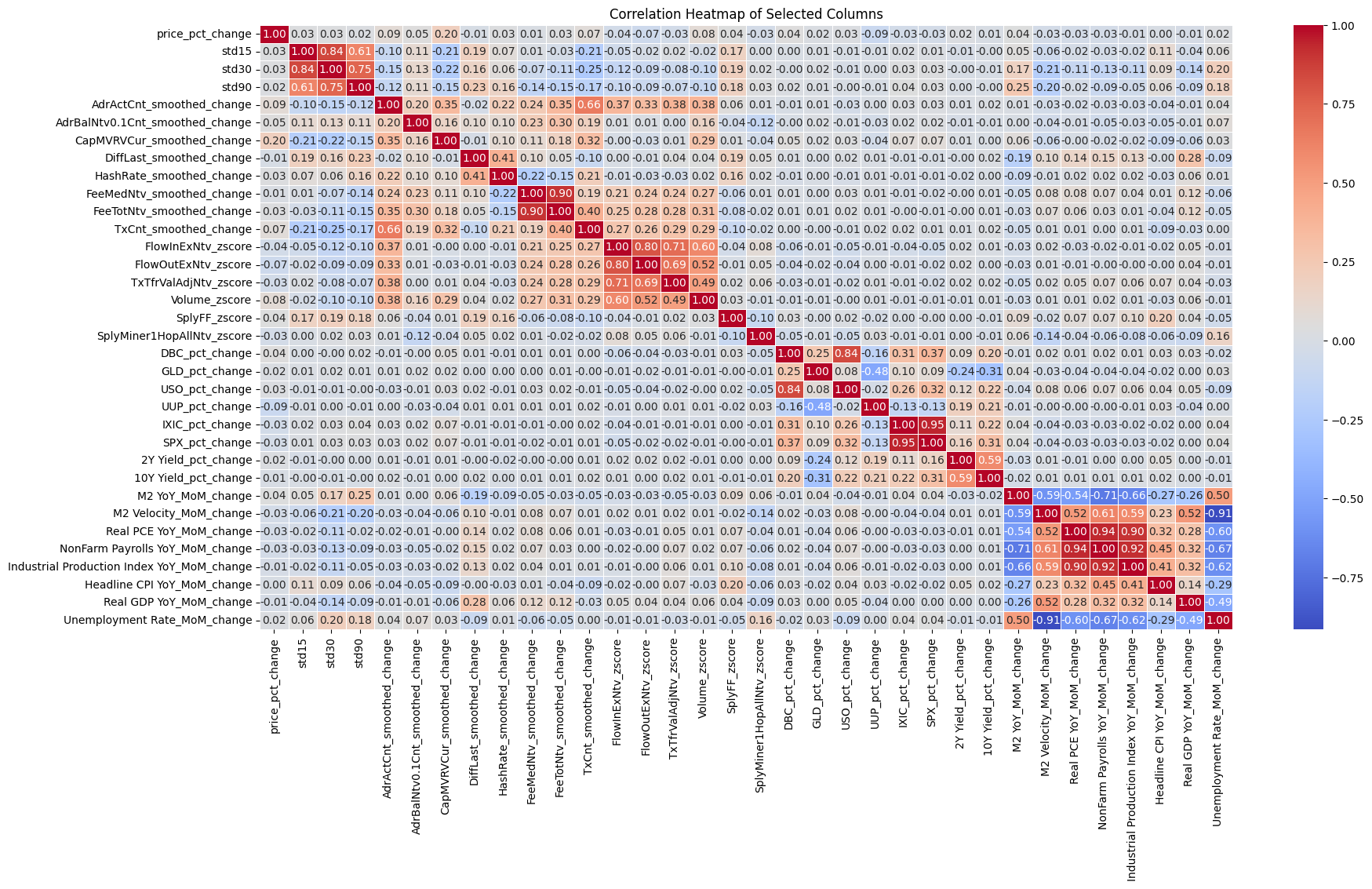
I now perform EDA on this new dataset with transformed variables. Our goal is to find features that may affect the percentage change of price. First, we look at the boxplot of our target feature, which shows that on most days, the price change is between plus/minus 10%.



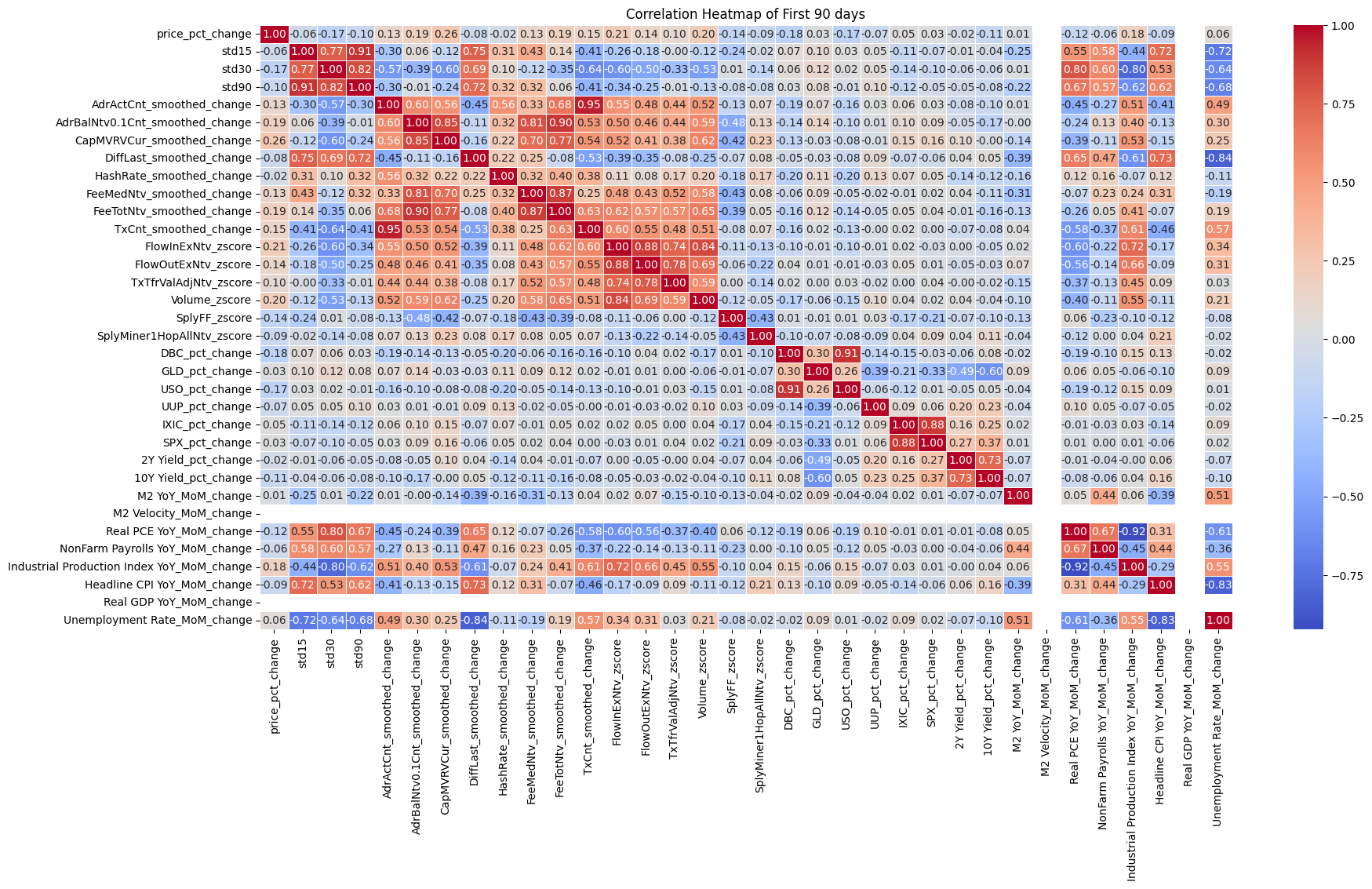
Overlaying it over a normal distribution, we can see that the series follows a normally distribution with a higher level of kurtosis and longer tails.

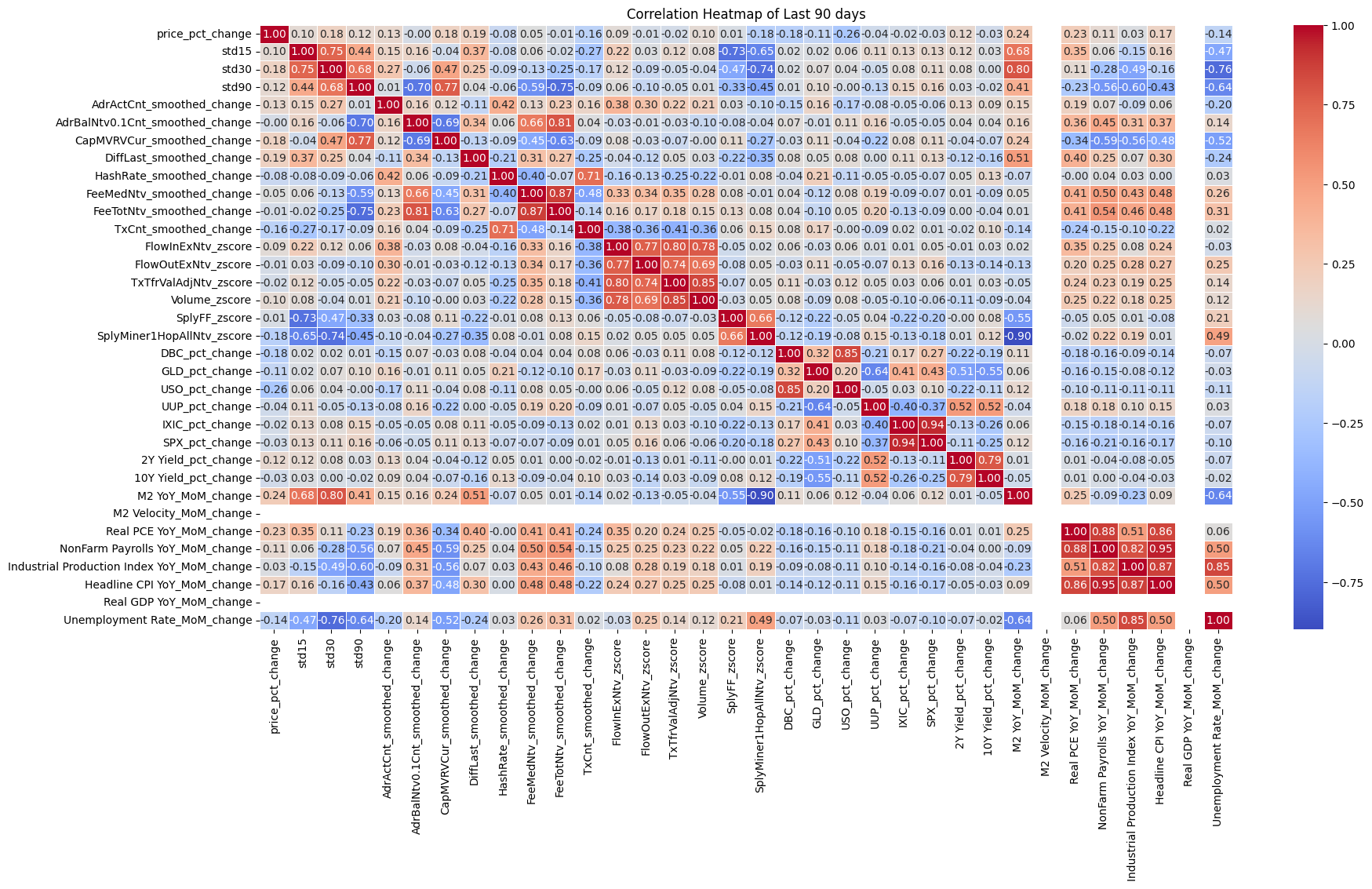


Next, we plot the correlation of all stationary variables, which shows that over the entire time series, price\_pct\_change only shows meaningful correlation with the MVRV\_change value. We also noticed a few highly correlated variables that could be eliminated in the machine learning step.



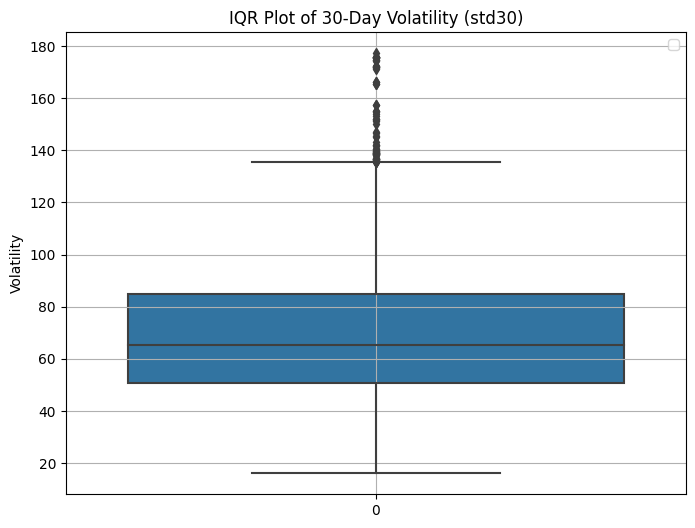
This prompt us to examine the correlation of features over different duration and we examine how the correlation plots change from the first 90 data points to last 90 data points. A noticeable change is in the first 90 days, price change has a higher correlation with blockchain native features such as Transaction Counts, Number of Addresses above 0.1 native units. Over the last 90 days, price change has a higher correlation with Macro data features such as M2 Supply, Real PCE MoM change. This is a phenomenon caused by the adoption curve of the blockchain and the market participants transitions from retail and blockchain users to institutions that seek an uncorrelated source of return.

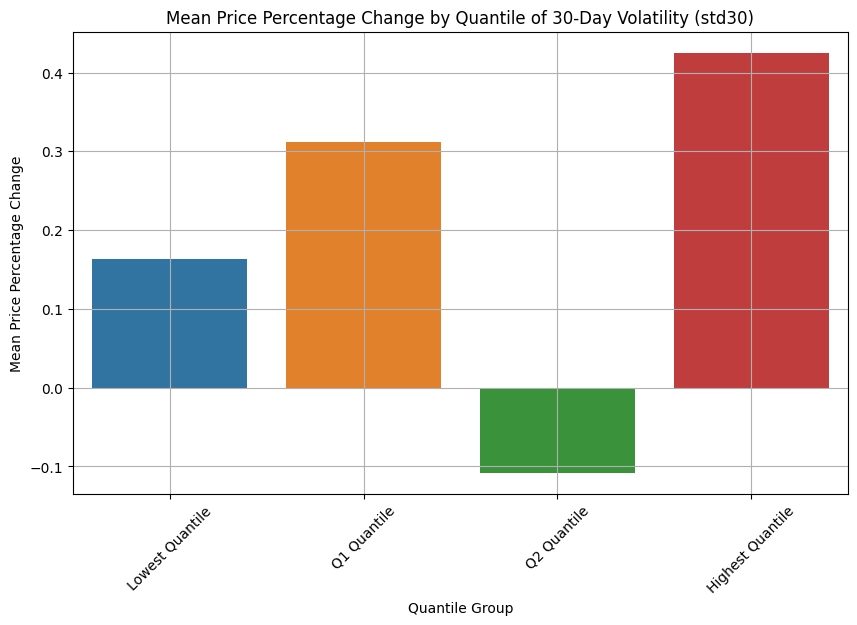


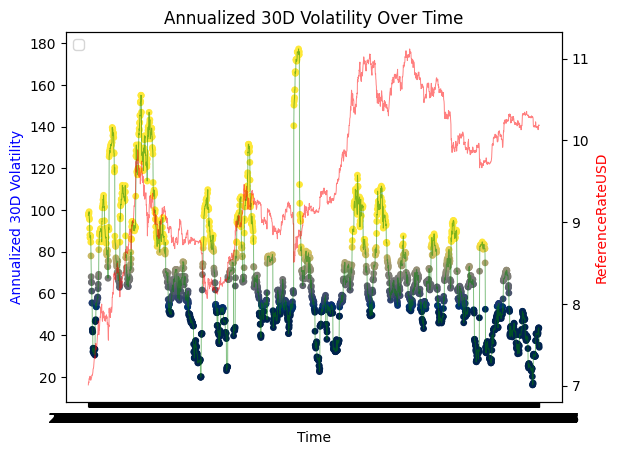


Finally, we need to look at the volatility of price returns since this is extremely in capital allocation models. Through examining the correlation matrices of negative and positive outliers, we can see the 15 day, 30 day and 90 day standard deviations of price returns have a high impact on price return series. Similarly, volume related metrics such as 30 day Z-score, Volume in and Volume out of exchanges also have a impact on outlier days.

We plot the IQR of 30 day volatility and divide the series into 4 buckets of volatility to examine the average price return of each volatility bucket.





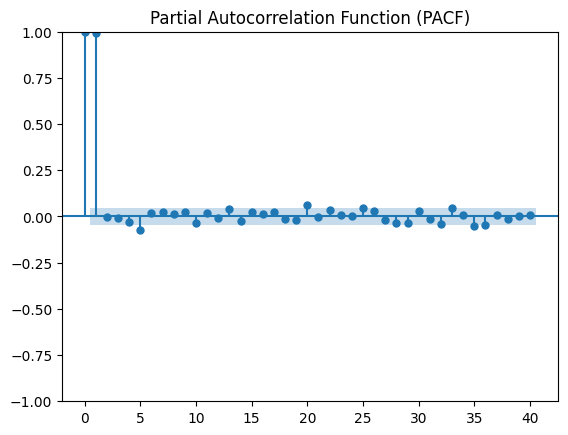
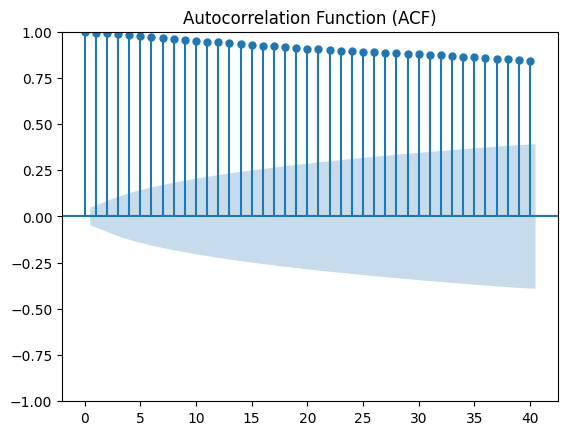


**Model Selection:**

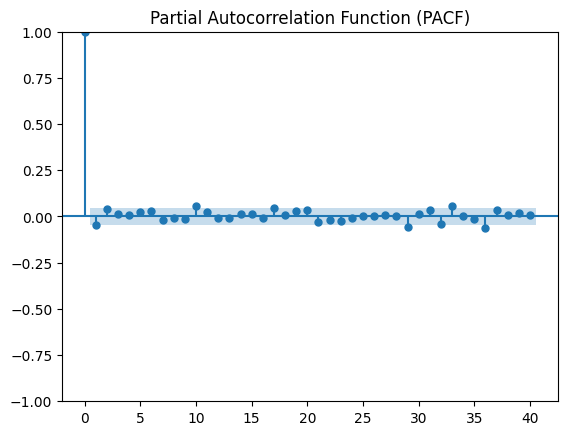
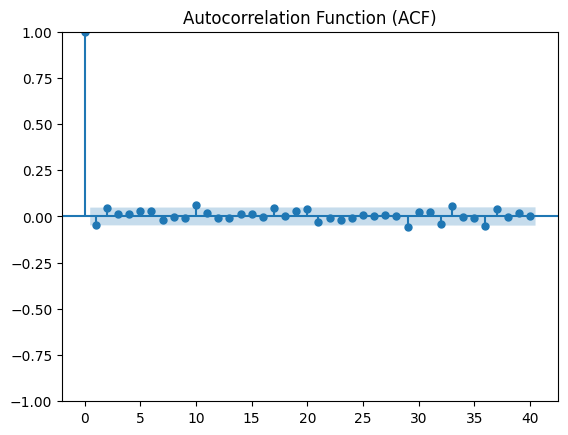
Just to recap, our data for bitcoin now has 2382 observations with 59 numerical variables for machine learning. Our preliminary EDA shows price returns may have correlations with volatility, volume and perhaps some blockchain native metrics that measures network adoption. However, the exact relationship is not clear and a model will need to be fitted and examined to understand these intricate relationships better.

Like all time-series analysis, we break down the target feature itself and see if there are any seasonal trends or autocorrelation within the series.

We look at the ACF and PACF plots of Price itself.

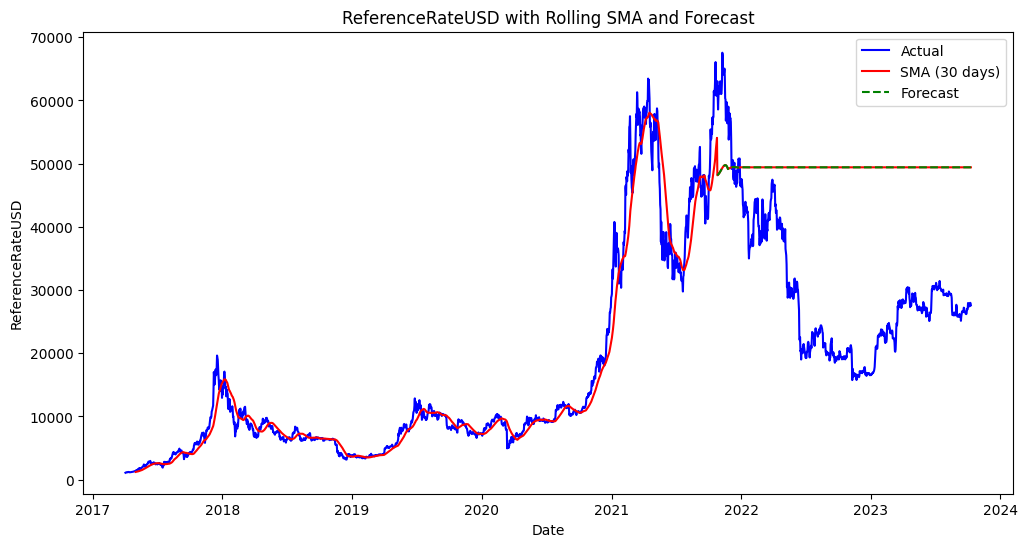


We then plot the ACF and PACF of price\_pct\_change series.

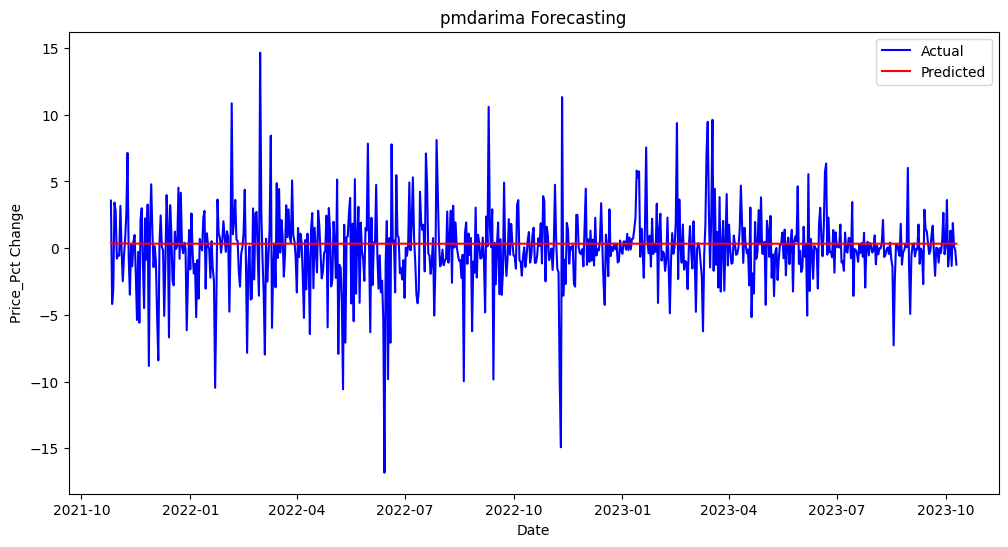


This leads us to the conclusion that our target variable should be the daily percentage change in price since it’s a stationary feature.

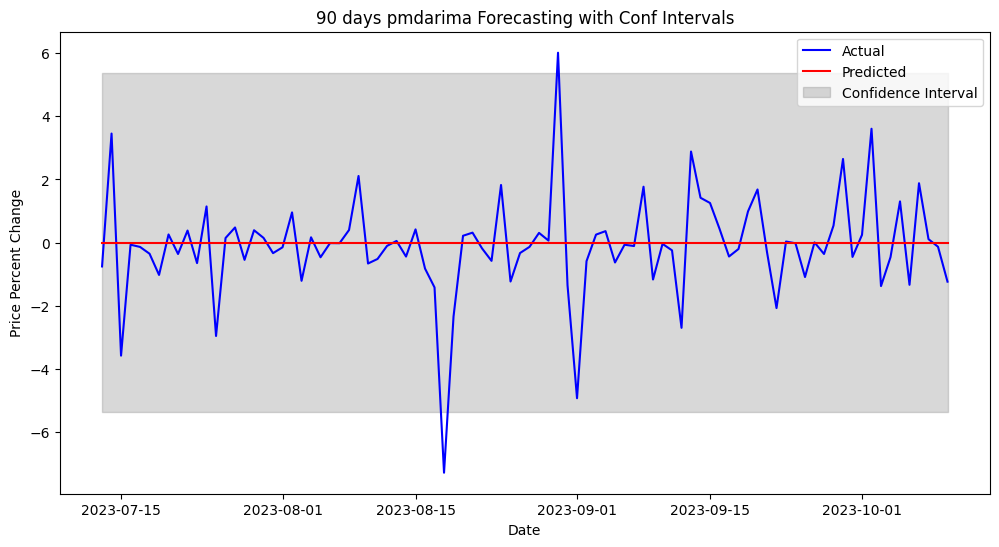
1st Model: Simple MA Model. This serves as our baseline for comparison later. SMA assumes future values are constant and will be the same as average of historical values.



2nd Model: 1 Factor ARIMA Model. We use pmdarima auto\_arima to obtain the best fitting model , which is order= (2,0,1)

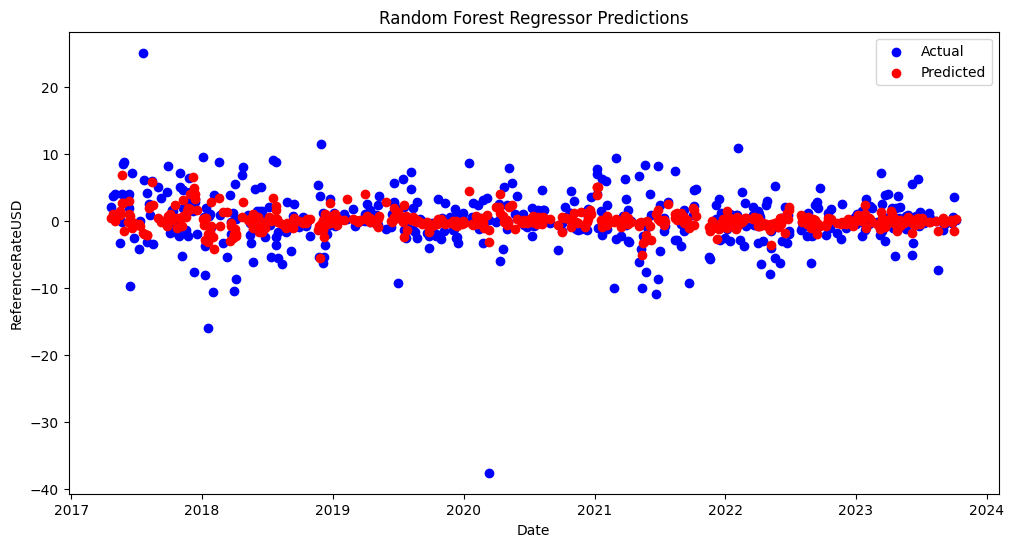


The predicted price changes look like a straight line around 0! It certainly looks like an error with the model. However, we zoomed into a portion of the predictions and also plot the confidence intervals.



Noticeably, most actual percentage change values are within the confidence interval, which shows that the model produced a mean close to 0 with large standard deviations. This 1 factor price model is clearly insufficient in predicting any meaningful patterns.

3rd Model: RF Regression. The intuition behind using a Random Forest Regression model here is to convert the problem into a time-invariant problem and determine if we can predict daily price changes based on the features of our dataset. The RF model is fitted on 80% of the data (as training data) and tested on 20% of the data. The features being used are all the stationary rate of change columns which was computed during the EDA step.



Predictions are noticeably more accurate than 1 factor ARIMA model. We calculated a few metrics of the model below. Directional Accuracy is if the predicted move is in the same direction as the actual move.

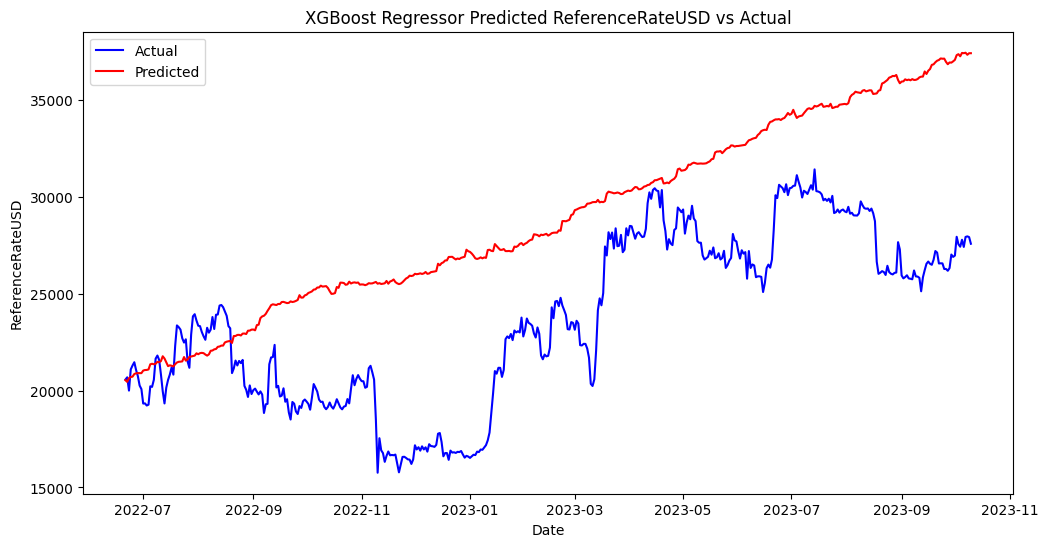
RMSE: 4.071003155759007

MAE: 2.6860502314798094

MAPE: 181.61195436278098%

Directional Accuracy: 53.03983228511531%

4th Model: XGBoost. We perform XGBoost on the same training data and tested it over the test data. The predicted percentage change data is then transformed into price data by taking the integral over time.



The model has the following error metrics when predicting percentage change in price:

RMSE: 4.012288653899718

MAE: 2.594338290197032

MAPE: 127.73048329243043%

Directional Accuracy: 56.39412997903563%

We perform Hyperparameter tuning of the XGBoost model which slightly improved the error metrics and produced a better price prediction chart.

RMSE: 3.9694614558340757

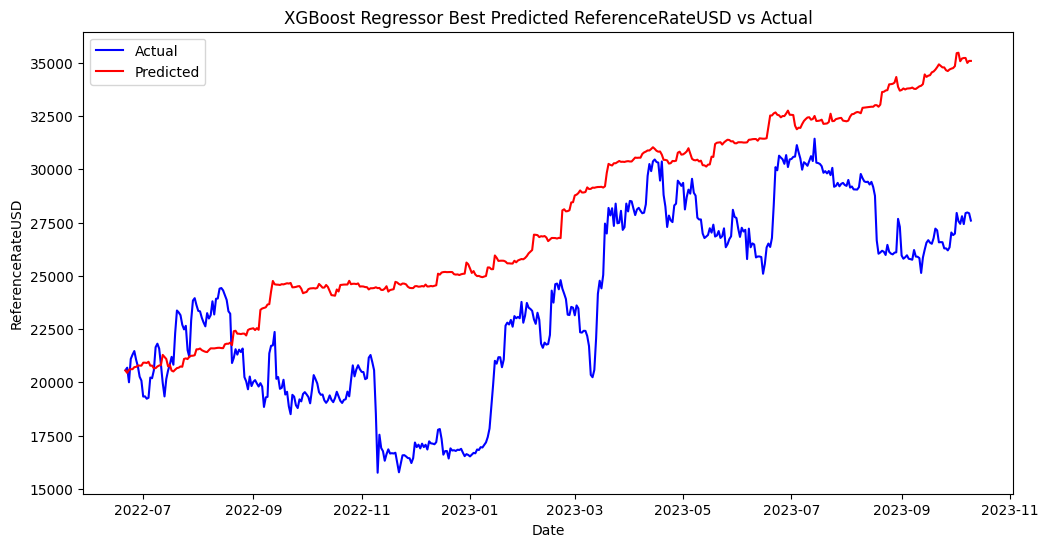
MAE: 2.5844950214713904

MAPE: 143.47828880998352%

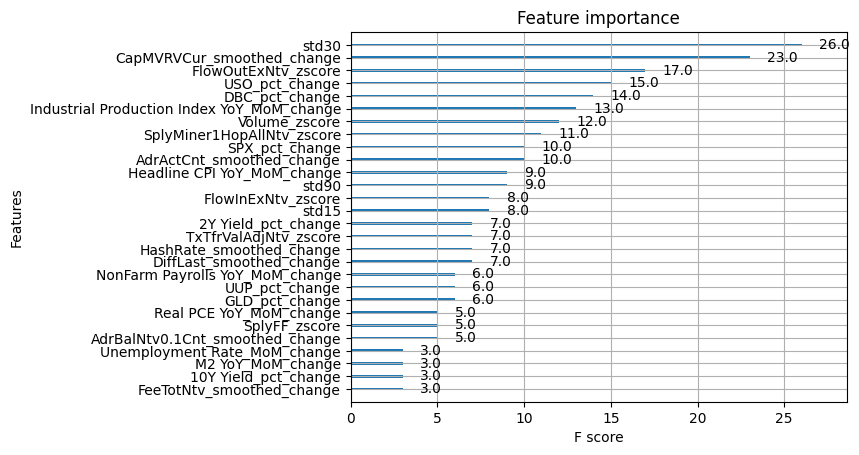
Directional Accuracy: 56.60377358490566%

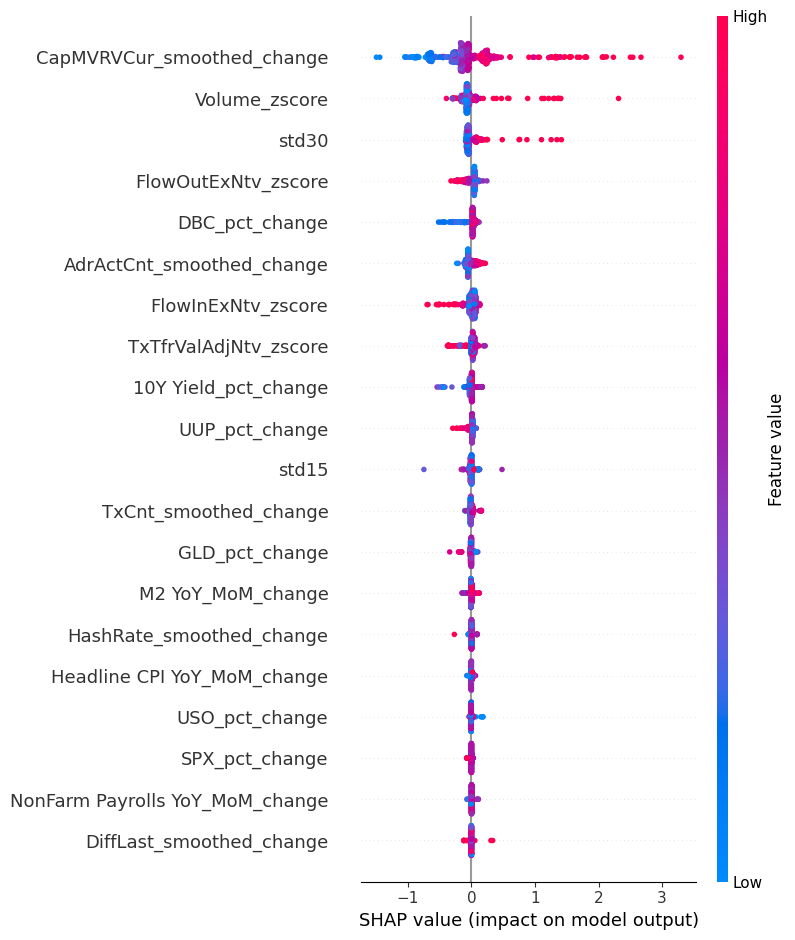
Best Hyperparameters: {'alpha': 10, 'colsample\_bytree': 0.7, 'learning\_rate': 0.01, 'max\_depth': 3, 'n\_estimators': 200, 'objective': 'reg:squarederror'}

Mean Squared Error: 15.756624249352381



XGBoost provided a time-invariant model which we can study to understand which features has the highest importance on our target feature. We perform looked at a simple feature importance chart of the XGBoost model and also looked at the SHAP Values.

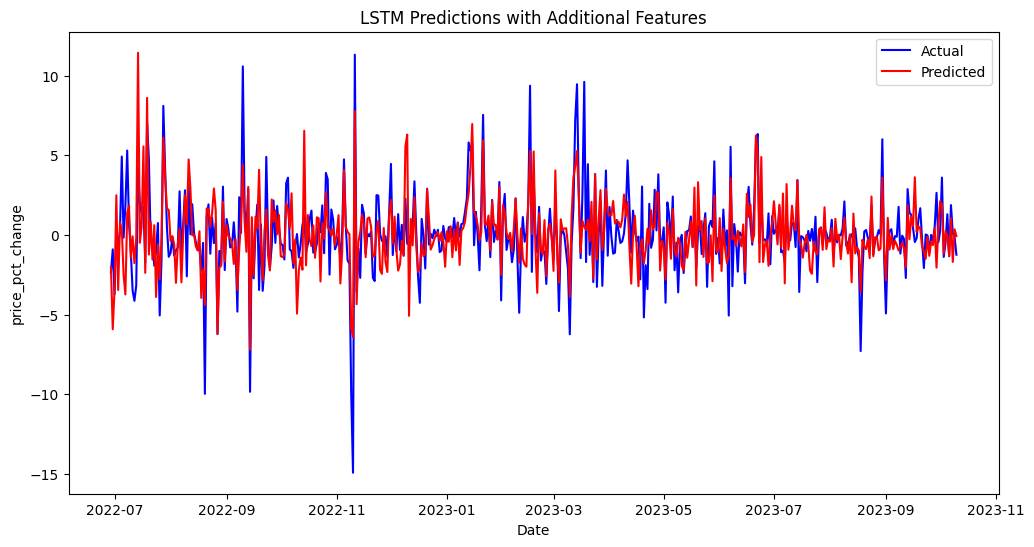




The feature importance analysis offered us similar insights to what was discovered during the EDA step! Std30, CapMVRVCur\_smoothed\_change and Volume\_zscore are return standard deviations and volume metrics and they offer the highest impact in predicting daily percentage change regardless of time! Given what we learned about our data series, I decided to input the top ranked features from XGBoost and the target feature into an LSTM Model.

**Final Model and Hyperparameter Tuning:**

Long Short Term Memory (LSTM) Neural Network: We used the top ranking features from XGBoost and feed it into a LSTM Model. The idea is perhaps there is some memory within our data structure that a 1 factor ARIMA model was not able to capture. We will explore using neural networks with a memory gate.



Our Error metrics showed a high much directional accuracy than XGBoost model of close to 70%. The MAE is also reduced to 1.4 from 2.58.

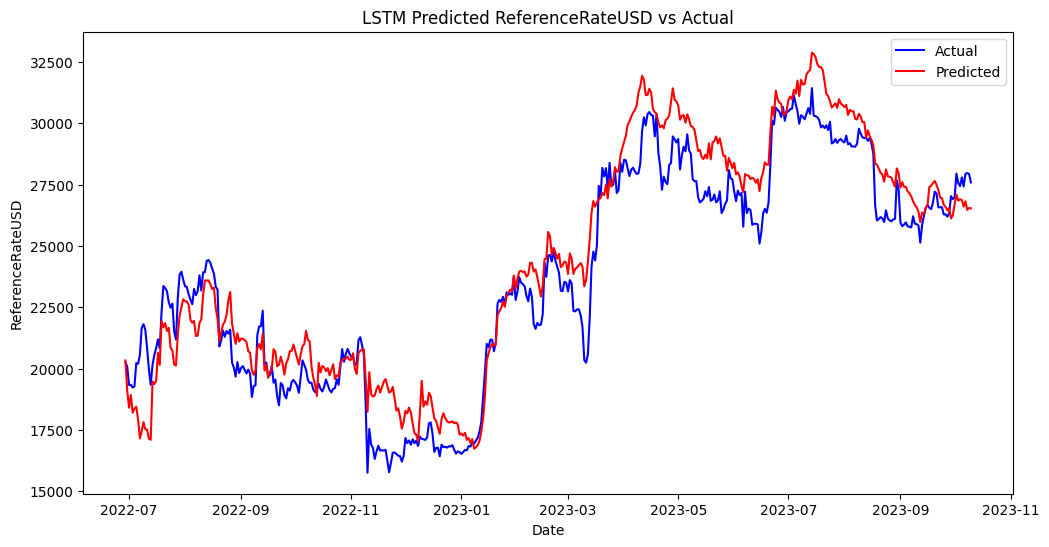
RMSE: 1.9241533507242965

MAE: 1.4003633892795562

MAPE: 929.500761124015%

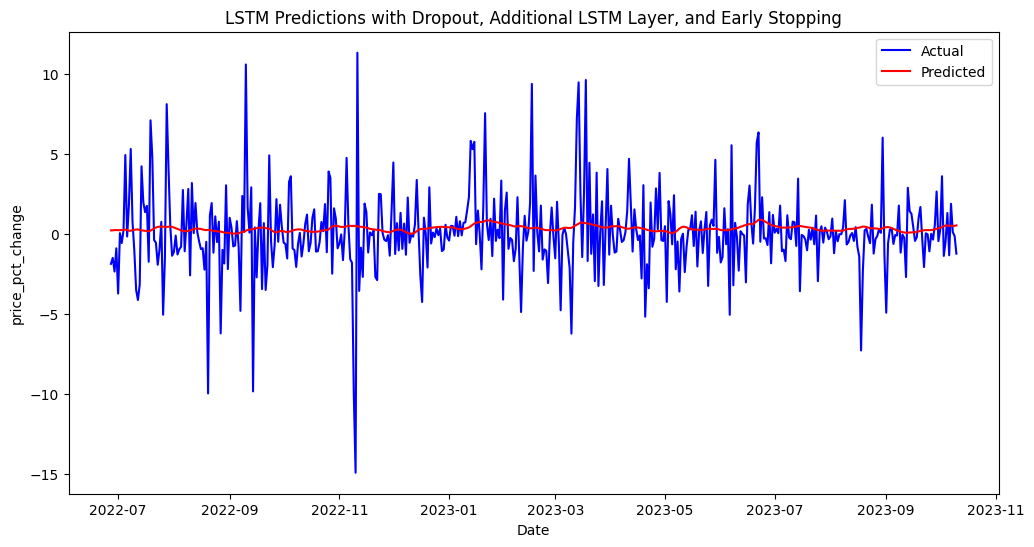
Directional Accuracy: 69.7228144989339%

Finally, taking the cumulative sum of percentage change predictions:



This is significantly better at predicting the bitcoin price than any previous models!

A 2 layer LSTM with a dropout layer of 0.2 is also being tested but the residuals showed that 2 layers smoothed the results too much and actually reduced the accuracy.



RMSE: 2.5590940892870346

MAE: 1.7140922997629073

MAPE: 529.0043199658106%

Directional Accuracy: 46.92144373673036%

We also tuned the sequence length used for LSTM and the best sequence length found is 40.

**Conclusion:**

The best prediction model for daily percentage price of bitcoin was found to be using a LSTM neural network with inputs being determined by XGBoost model. This model produced the least error and offers a prediction price that closely tracks the actual price.

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMSE | MAE | Directional Accuracy |
| LSTM with XGBoost | 1.924 | 1.4 | 69.72% |
| XGBoost | 3.969 | 2.585 | 56.6% |
| Random Forest | 4.07 | 2.68 | 53.04% |
| LSTM 2 Layer | 2.559 | 1.714 | 46.92% |