Machine Learning and AI in Finance

Machine Learning Project:

Predicting cross-exchange cryptocurrency price differences using Machine Learning techniques.

Janek Edwards – 15909840

Table of Contents

Machine Learning and AI in Finance ***1***

Machine Learning Project: **1**

1. Business Case Development **4**

2. Data Description **5**

3. Methodology **6**

3.1 Data Preprocessing 6

3.2 Machine Learning Approach 7

4. Implementation **9**

4.1 LSTM Model 9

4.2 XGBoost Model 12

5. Results and Visualization **13**

5.1 LSTM Model 13

5.2 XGBoost Model 22

5.3 Results Analysis and Comparison 24

6. Business Impact Analysis **25**

6.1 Interpretation of Results in Business Context 25

6.2 Economic Implications 25

6.3 Practical Implementation Considerations 25

6.4 Limitations and Potential Improvements 26

7. Technical Documentation **26**

7.1 Dependencies 26

7.2 Code Structure Overview 27

7.3 Instructions for Reproduction 30

8. References **30**

Appendix **31**

LSTM Model Single Feature Optimized 31

LSTM Model Single Feature 36

LSTM Model Five Feature 38

LSTM Model Eleven Feature 40

XGBoost Model 41

Table of Figures

Figure 1: Forward Arbitrage over time 4

Figure 2: Top 5 Features 10

Figure 3: Top 11 Features 11

Figure 4: Folds 1-5 Single Feature 13

Figure 5: Folds 1-5 Five Feature 15

Figure 6: Correlation of Input Variables 5 Feature 17

Figure 7: Folds 1-5 Eleven Feature 18

Figure 8: Correlation of Input Variables 11 Feature 20
Figure 9: Folds 1-5 Single Optimized Feature 21
Figure 10: Folds 1-5 XGBoost 23

Table of Tables

Table 1: Dataset Columns 5
Table 2: Dataset Statistics 5
Table 3: Standard LSTM Model Characteristics 9
Table 4: Hyperparameters LSTM 11
Table 5: XGBoost Hyperparameters 12
Table 6: Hyperparameters XGBoost Results 12
Table 7: LSTM Single Feature Results 13
Table 8: LSTM 5 Feature Results 15
Table 9: LSTM 11 Feature Results 17
Table 10: LSTM Single Optimized Feature Results 20
Table 11: XGBoost Results 22

1. Business Case Development

Cryptocurrencies exhibit price differences across exchanges due to multiple factors, such as liquidity diferences, macroeconomic conditions, and forex fluctuations. These price discrepancies create opportunities where traders can buy cryptocurrency assets for a lower price on one exchange and sell at a higher price on another for profit.

In this project Bitcoin will be selected as a cryptocurrency, and its price differences between an international exchange, Kraken, and a South African exchange, Luno, will be analyzed. Currently, the Bitcoin/ZAR price pair often trades on Luno at a premium when compared to the Bitcoin/USD price pair on kraken. To take advantage of this, traders can buy Bitcoin using USD on Kraken, send the Bitcoin to their Luno wallet, and sell it on the exchange for ZAR and generate a profit.

The graph below shows the percentage price differences between the two exchanges over time accounting for real time forex rates. This does not take into account transaction fees. It should also be mentioned that although this percentage price difference will be referred to as “forward arbitrage”, this is not a pure arbitrage opportunity as it is not a completely risk-free strategy.

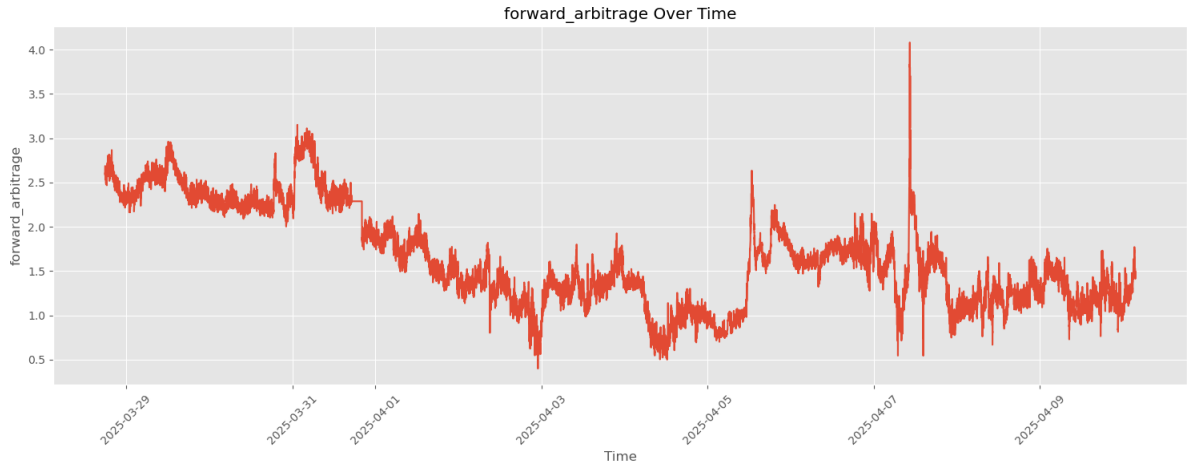


Figure 1: Forward Arbitrage over time

Due to the fluctuation of aforementioned market factors, this price discrepency fluctuates over time. Additionally, Bitcoin can take up to an hour to transfer between exchanges. Thus developing a Machine Learning model to predict these price differences ahead of time would be useful in choosing the best times to use the strategy and maximize profit.

Considering the transfer time of bitcoin, this project will aim to predict the one hour future price of bitcoin using two machine learning techniques: LSTM and XGBoost.

2. Data Description

In order to access Bitcoin market data, an API was used, accessing CCXT API *(CCXT, 2025)*. Through the CCXT API real time data from both Kraken and Luno exchanges were captured. Additionally, USD/ZAR real time exchange rates were captured using the Alpha Vantage API *(Alpha Vantage, 2025)*. The following data was fetched every minute from 28/03/2025 17:51:00 to 10/04/2025 04:35:00.

Table 1: Dataset Columns

Timestamp	Time at which values were fetched
Kraken BTC/USD Ask price	Most recent ask price
Kraken BTC/USD Bid price	Most recent bid price
Kraken BTC/USD Ask Volume	Volume of top 20 asks
Kraken BTC/USD Bid Volume	Volume of top 20 bids
Kraken BTC/ZAR Ask price	Most recent ask price
Kraken BTC/ZAR Bid price	Most recent bid price
Kraken BTC/ZAR Ask Volume	Volume of top 20 asks
Kraken BTC/ZAR Bid Volume	Volume of top 20 bids
USD/ZAR Exchange Rate	Most recent price

After retaining only the relevant columns, the Initial dataset statistics are shown below.

Table 2: Dataset Statistics

	btc_usd_ ask	**btc_usd_ ask_volume**	**btc_usd_ bid_volume**	**btc_zar_ bid**	**btc_zar_ ask_volume**	**btc_zar_ bid_volume**	**usd_zar_rate**	**forward_ arbitrage**
count	17718	17718	17718	17718	17718	17718	17718	17718
mean	82013,9	22,3	19,74	1574666,75	1,174	0,889	18,90	1,637
std	2564,6	16,9	16,31	32367,95	0,623	0,491	0,46	0,554
min	74472,8	0,084	0,075	1450007	0,106	0,150	18,21	0,399
25%	81555,1	8,72	5,9535	1552245	0,808	0,578	18,415	1,211
50%	82766,7	19,43	15,878	1573074,5	1,055	0,772	18,80	1,531
75%	83606,9	31,76	30,06	1599538,75	1,4	1,060	19,25	2,116
max	88446	109,84	157,49	1677796	11,98	5,767	19,93	4,081

The time period for which data was fetched was 17925 minutes long, for which 17718 data points were retrieved representing a strong coverage of 98.845%. Missing data points were due to API failures. There are no other limitations present in the data. The "forward_arbitrage" values were calculated as follows:

$$forward\_arbitrage = \frac{\frac{btc\_zar\_bid}{usd\_zar\_rate} - btc\_usd\_ask}{btc\_usd\_ask} * 100$$

The target variable “target_1hr” was created by shifting “forward_arbitrage” values by one hour. This is what we aim to predict.

3. Methodology

3.1 Data Preprocessing

3.1.1 *Missing values*

As mentioned previously, the data had a coverage of 98.845%, so there were few missing values. To handle the missing values, each datapoint was aligned to the closest minute and then missing values were treated using forward fill. The resulting coverage was 100%. Forward fill is often use in financial time series data and was thus deemed appropriate.

3.1.2 Outliers

Outliers were handled through Winsorization. Values were clipped at 0.01 and 0.99 percentiles. As arbitrage is being predicted, extreme values often represent actual trading opportunities. Thus, Winsorization was selected as it allows these signals while preventing them from dominating the model.

3.1.3 Feature Engineering

Temporal Features:
Day of week and hour of day features were created to capture any patterns that may be related to time.

Technical Features:
Moving average, RSI, volatility, price range, rate of change, momentum, and Bollinger band features were created for all price-based variables. This includes BTC/ZAR bid price, BTC/USD ask price, USD/ZAR rate, and forward arbitrage. These were all created for various periods.

Volume Features:
BTC/ZAR and BTC/USD ask depth and bid depth features were created to gauge relative volume-based demand. Moving average, volatility, and momentum features were then created from these for various periods.

Cross-Market Features:
Price spread, relative strength, and volatility ratio features were created between BTC/USD and BTC/ZAR markets for various periods.

3.1.3 Scaling Data

LSTM Model:
RobustScaler was used to scale the data for the LSTM model. It was ensured that the data was split into training and test datasets before scaling to prevent data leakage. RobustScaler was selected for its suitability for financial data and its lower sensitivity to outliers.

XGBoost:
StandardScaler was used for the XGBoost model. XGBoost is inherently robust to outliers due to its tee structure, so StandardScaler was selected. It was ensured that the data was split into training and test datasets before scaling to prevent data leakage.

3.1.3 Feature Selection

LSTM Model:
Due to the extensive feature engineering conducted in this project, features had to be analyzed and trimmed down. To do this, a mutual information regression was performed to measure the dependency between each feature and the target variable. The most important features were iteratively selected while ensuring that selected features did not share a correlation greater than 0.8. The most important non highly correlated features were selected in groups of 5 and 11. Additionally, a single feature dataset containing only the “forward_arbitrage” variable was used as a reference.

XGBoost:
For the XGBoost feature selection, a basic XGBoost model was used to retrieve feature importance. It uses 'gain' importance type, which measures the improvement in accuracy brought by a feature. Because XGBoost is robust to having many features, the maximum number of features were selected while ensuring the correlation was no greater than 0.8. This resulted in 62 features being used for XGBoost. The list of features can be found in the appendix.

3.2 Machine Learning Approach

3.2.1 Overview of Models

Long Short-Term Memory (LSTM) Model:

Arbitrage opportunities often follow patterns over time, making LSTM appropriate as it has the ability to capture long term patterns. The target variable is a future value, requiring temporal understanding, which suits LSTM. LSTM has the ability to capture momentum and market conditions that persist over time as is the case with the financial data that has been selected for this project. LSTM has the ability to learn

complex patterns in price movement. Additionally, LSTM can learn interactions between different market features and cross-market relationships.

XGBoost Model:

Finanical markets often have non-linear relationships. XGBoost is strong at capturing these relationships through tree-based splitting, feature interactions, and non-linear transformations. XGBoost provides good analysis of feature importance, allowing understanding for which factors drive arbitrage opportunities. Additionally, XGBoost is robust to outliers, different feature scales, and noisy data, all of which is present in financial data.

3.2.2 Model Comparison

Different Learning Approaches:

- LSTM: Sequential, temporal patterns
- XGBoost: Feature-based, non-linear relationships

Different Strengths:

- LSTM: Better at capturing long-term dependencies
- XGBoost: Better at handling feature interactions and importance

Different Weaknesses:

- LSTM: Can be harder to train and interpret
- XGBoost: Less effective at capturing pure temporal patterns

3.2.3 Hyperparameter Optimization Strategy

LSTM:
Due to the extensive training time required for the optimization of numerous hyperparameters for the LSTM model, the following strategy was implemented.
A standardized variant of each model was trained and tested using 3 different combinations of features that were relevant to each model. The best performing features were identified and then the final models were trained and tested on these features in combination with a hyperparameter optimization strategy outlined below.
For the LSTM hyperparameter optimization, Bayesian optimization was used with Gaussian Process. This was chosen because it is more efficient than a grid search strategy.

XGBoost:
For XGBoost, we only had one set of features to optimize for. As with LSTM, Bayesian optimization was used with Gaussian Process. This was chosen because it is more efficient than a grid search strategy.

3.2.4 Cross-Validation Approach

Cross-validation was implemented for both the LSTM and XGBoost models. The TimeSeriesSplit function was used was used. This method maintains temporal order and splits the data into n folds. Each fold uses all previous data for training, and the test set is always after the training set. This prevents data leakage and gives a more realistic evaluation of model performance.

3.2.5 Evaluation Metrics

For both XGBoost and LSTM, the following evaluation metrics were used:

- RMSE (Root Mean Square Error) - Measures the square root of the average squared differences between predictions and actual value.
- MAE (Mean Absolute Error) – Measure the average absolute differenes between predictions and actual values.
- R-squared Score – Measures how well the model's predictions match the actual values.
- A variety of graphical illustrations of model performance

4. Implementation

4.1 LSTM Model

As mentioned in Section 3.2.3, A standard model was selected and trained on 3 sets of different feature combinations.

4.1.1 Standard Model Characteristics

The standard model characteristics are shown in the table below.

Table 3: Standard LSTM Model Characteristics

Variable	**Details**
Input Shape	(90, number of features) → 90 time steps per sample
1st LSTM Layer	64 units, activation='tanh', return_sequences=True, recurrent_dropout=0.1
2nd LSTM Layer	32 units, activation='tanh', recurrent_dropout=0.1
Dense Layer 1	16 units, activation='relu'
Output Layer	1 unit (regression output)
Loss Function	Huber loss
Optimizer	Adam
Callbacks	ReduceLROnPlateau, EarlyStopping
Epochs	100 (with early stopping)
Batch Size	45

Number of Splits	5
Sequence Length	90 time steps
Validation Split	10% of training fold during model training

4.1.2 LSTM Model with single feature

The standard model was trained and tested using only “forward_arbitrage” as a feature. This is essentially the target variable lagged by one hour.

4.1.3 LSTM Model with 5 features

The standard model was trained and tested using the following features:

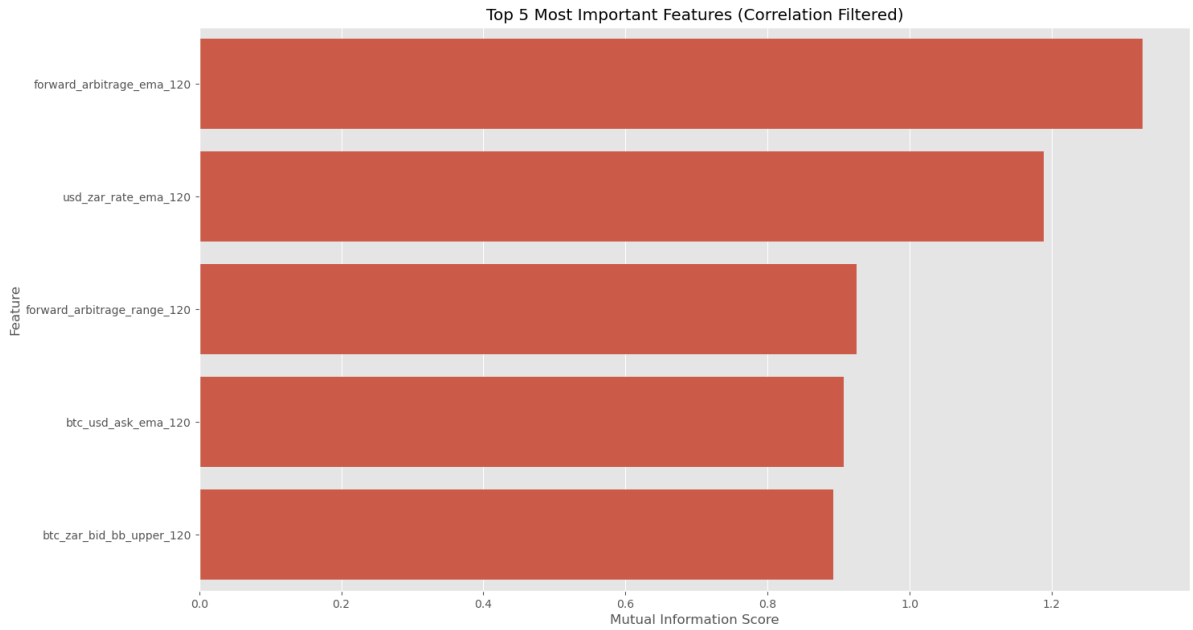


Figure 2: Top 5 Features

4.1.4 LSTM Model with 11 features

The standard model was trained and tested using the following features:

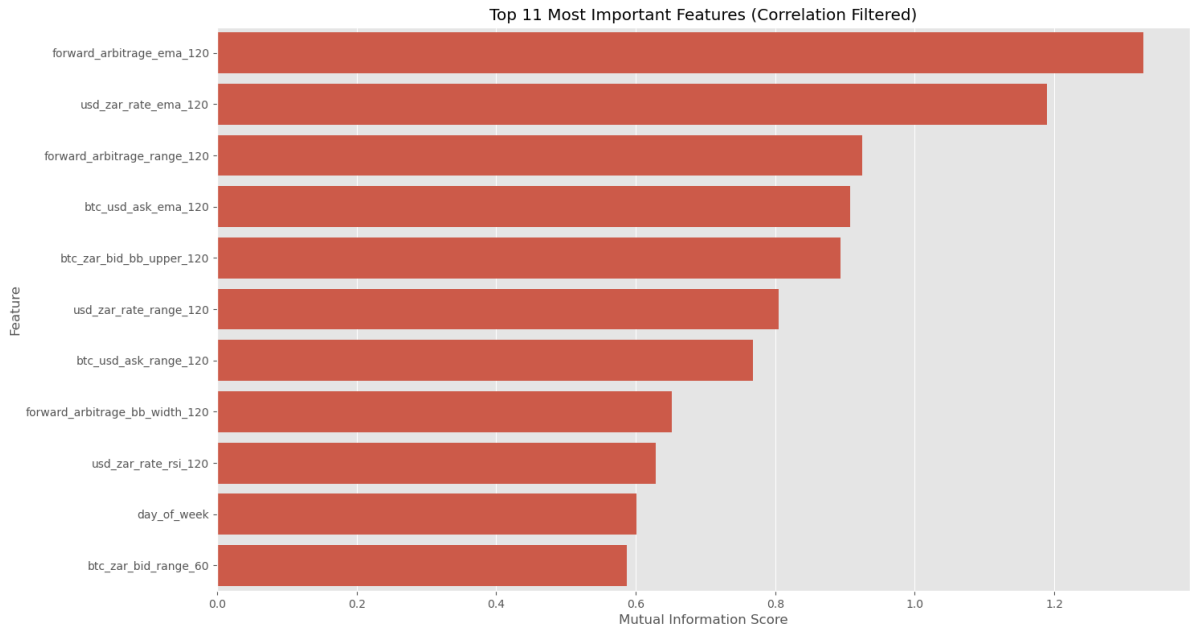


Figure 3: Top 11 Features

4.1.5 LSTM Hyperparameter Tuning

As mentioned in section 3.2.3, due to the extensive training time required for hyperparameter tuning, the only best performing features based on the standard model were selected for hyperparameter tuning.

The single feature model had the best performance, so it was chosen for hyperparameter optimization. The following parameters were optimized:

Table 4: Hyperparameters LSTM

Parameter	**Optimization range**
Layer 1 units	[32,128]
Layer 2 units	[16, 64]
Dropout rate	[0.1, 0.3]
Learning rate	[0.0001, 0.001]
Batch size	[32, 64]
Sequence length	60, 90

The results of the hyperparameter optimization are shown below.

Table 5: Hyperparameters LSTM Results

Parameter	Optimal Value
Layer 1 units	104
Layer 2 units	64
Dropout rate	0.2336
Learning rate	0.00014
Batch size	45
Sequence length	90

4.1.6 LSTM Model with single feature (optimized)

The standard model with the adjusted hyperparameters shown in Table 5 was then trained and tested using only "forward_arbitrage" as a feature.

4.2 XGBoost Model

The XGBoost model was trained and tested using 62 features, which can be found in the appendix.

4.2.1 XGBoost Hyperparameter Tuning

The following parameters were optimized.

Table 5: XGBoost Hyperparameters

Parameter	Optimization range
Number of estimators	[50,300]
Learning rate	[0.01, 0.3]
Max depth	[3, 10]
Minimum child weight	[1, 10]
Subsample	[0.6, 1]
Column subsample	[0.6, 1]
Gamma	[0, 0,5]
Alpha	[0, 1]
Lamda	[0, 1]

The results of the hyperparameter optimization are shown below.

Table 6: Hyperparameters XGBoost Results

Parameter	Optimal Value
Number of estimators	50
Learning rate	0.2999
Max depth	3
Minimum child weight	1
Subsample	1
Column subsample	0.6
Gamma	0

Alpha	1
Lamda	1

5. Results and Visualization

5.1 LSTM Model

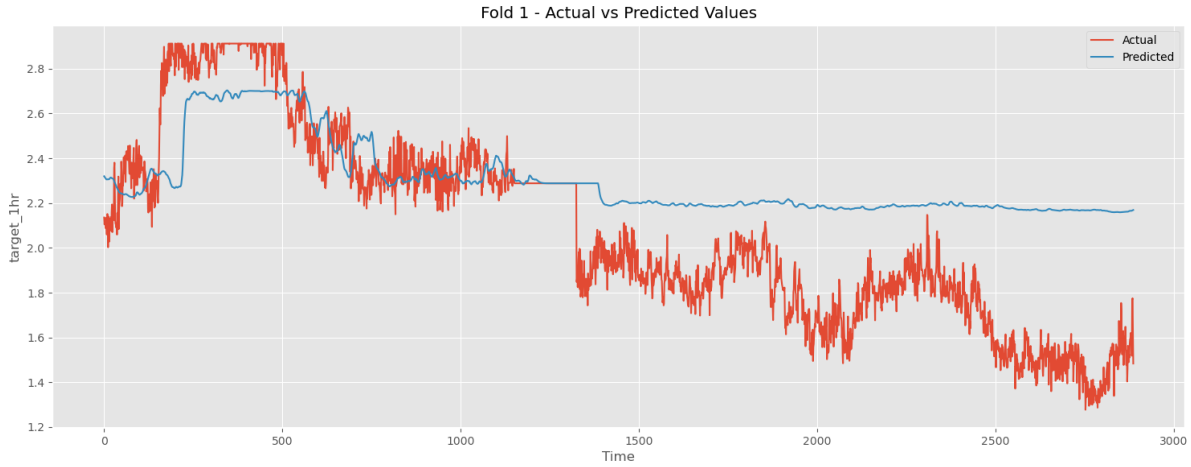
5.1.1 LSTM Model with single feature

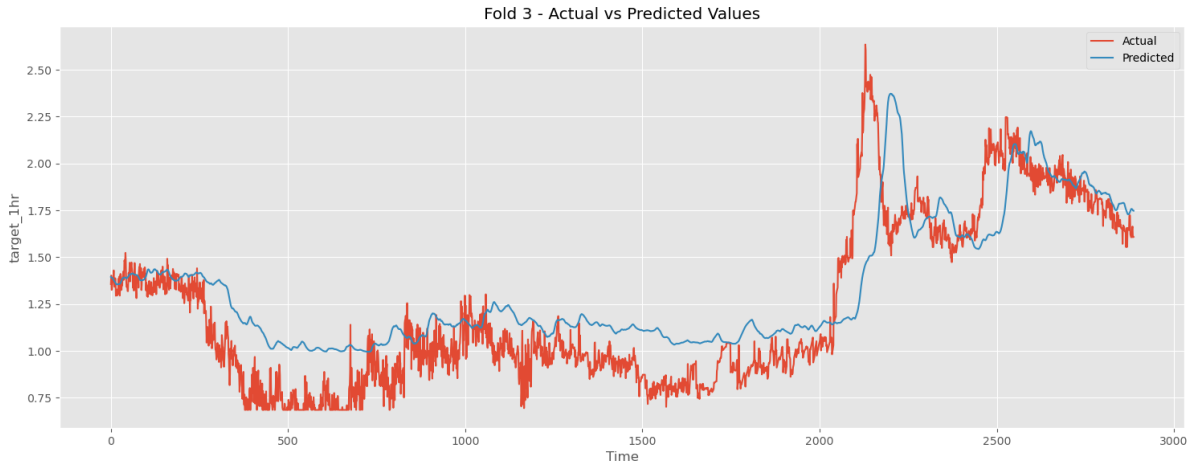
The result metrics for this model are as follows:

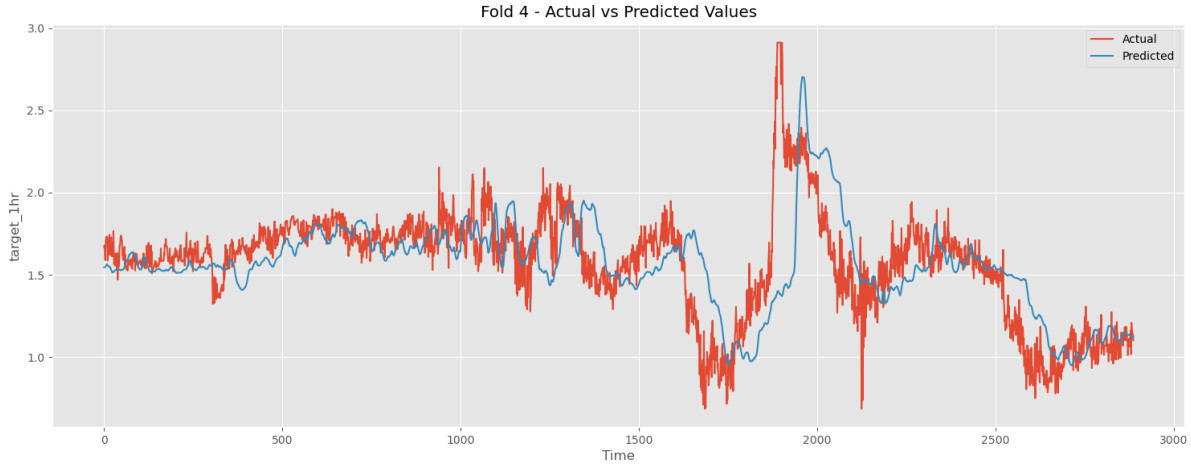
Table 7: LSTM Single Feature Results

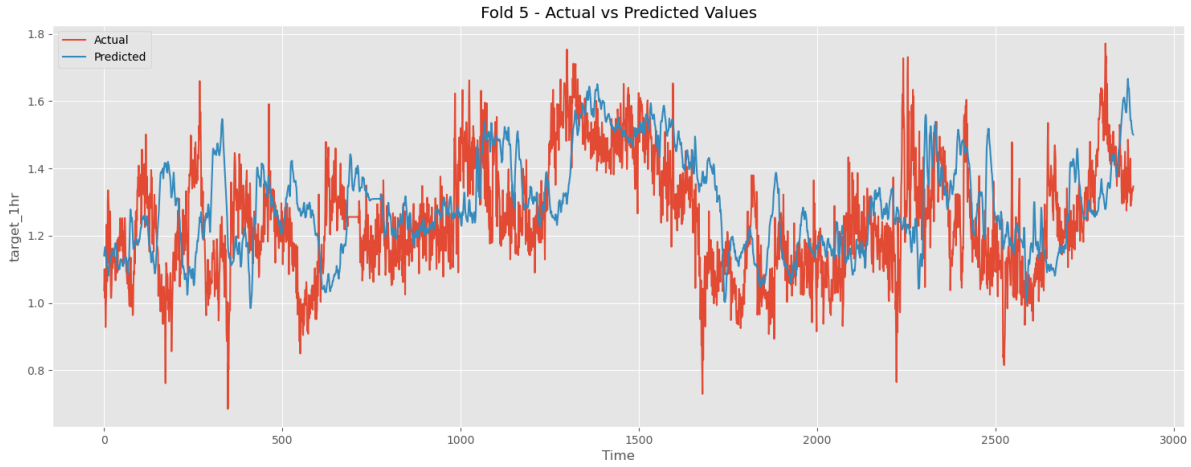
Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
RMSE	0.3699	0.3195	0.2518	0.2649	0.1756	**0.2763**
MAE	0.2968	0.2718	0.1960	0.1804	0.1410	**0.2172**
R2	0.2542	-1.4593	0.6682	0.3295	-0.0993	**-0.0614**

Figure 4: Folds 1-5 Single Feature









Analysis:

The model shows good approximation of actual values, picking up relevant trends. However, in fold 2 the model failured to capture trends accurately. Additionally, the model failed to correctly predict trends in fold 1 after the period for which forward fill

was used. The model can predict arbitrage values within roughly 0.25 units. The negative R-squared indicates that the model doesn't perform better than just using the average as a predicted value. This was the best performing model out of the baseline LSTM models.

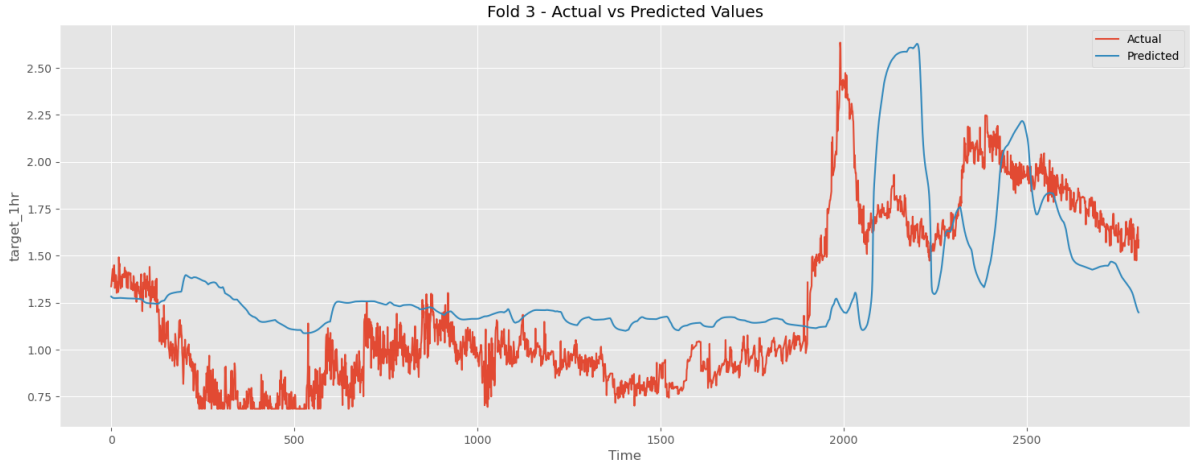
5.1.2 LSTM Model with 5 features

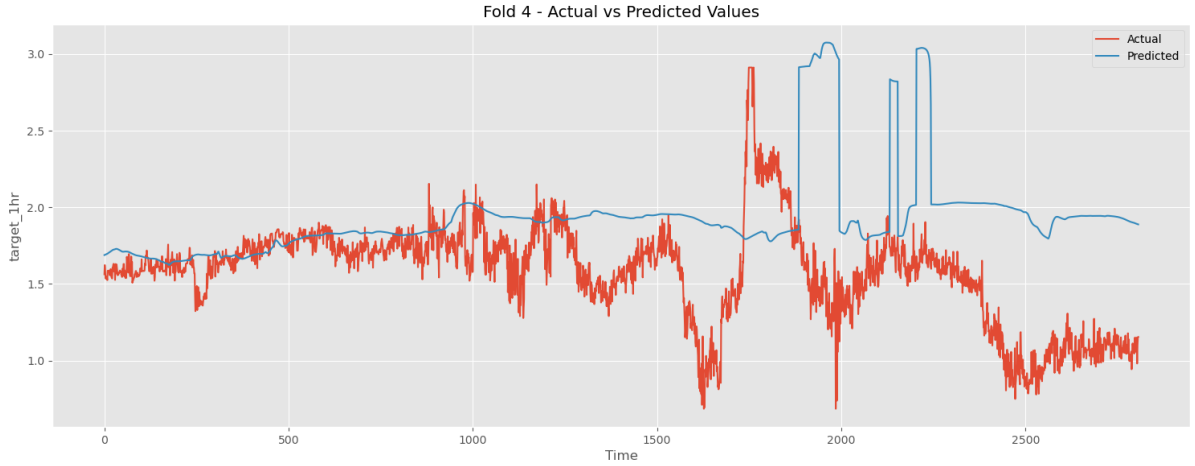
The result metrics for this model are as follows:

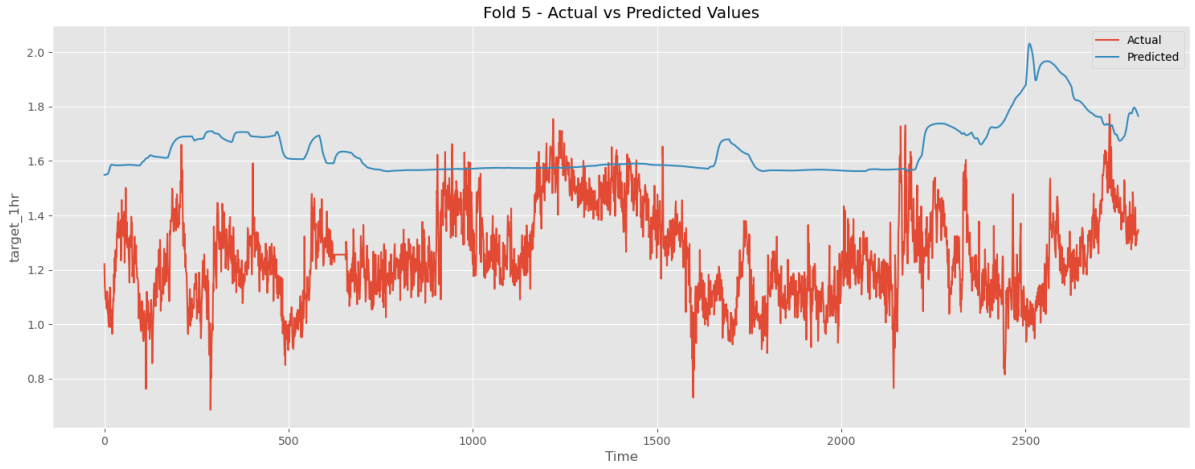
Table 8: LSTM 5 Feature Results

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
RMSE	0.4955	1.1297	0.3831	0.5822	0.4373	**0.6056**
MAE	0.4097	1.1017	0.3136	0.4174	0.3911	**0.5267**
R2	-0.2164	-30.0696	0.2595	-2.2431	-5.7331	**-7.6006**

Figure 5: Folds 1-5 Five Feature


Fold 3 - Actual vs Predicted Values
Actual
Predicted
target_1hr
Time


Fold 4 - Actual vs Predicted Values
Actual
Predicted
target_1hr
Time


Fold 5 - Actual vs Predicted Values
Actual
Predicted
target_1hr
Time

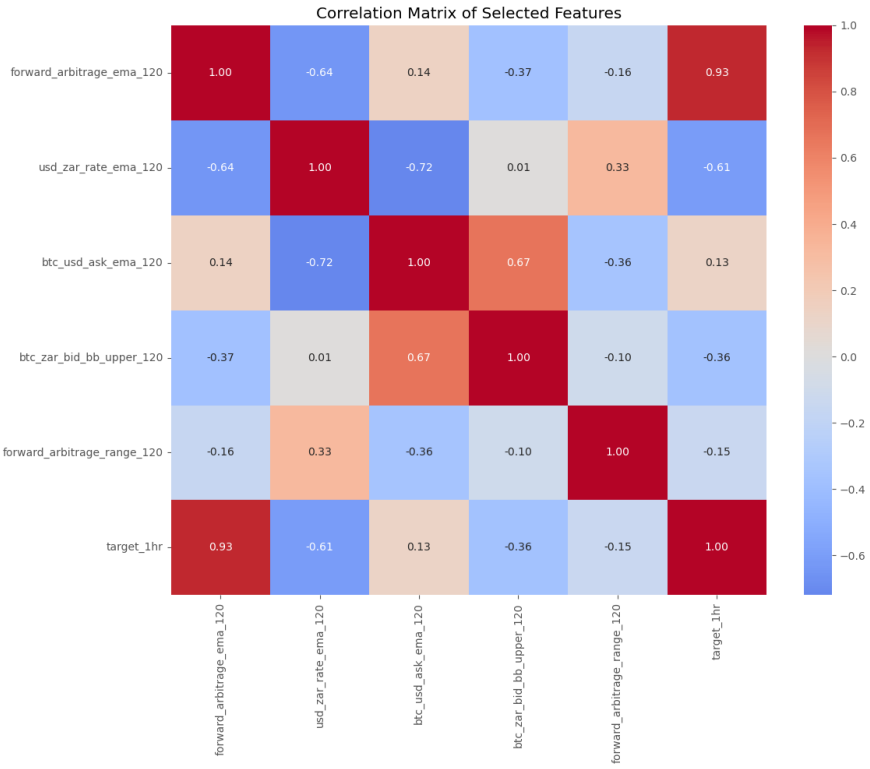


Figure 6: Correlation of Input Variables 5 Feature

Analysis:

The model shows a poor approximation of actual values, failing to pick up relevant trends. Where trends are correctly predicted, there is a large bias in the model. The model can predict arbitrage values within roughly 0.5 units. The strong negative R-squared indicates that the model doesn't perform better than just using the average as a predicted value. Overall the model performance is poor.

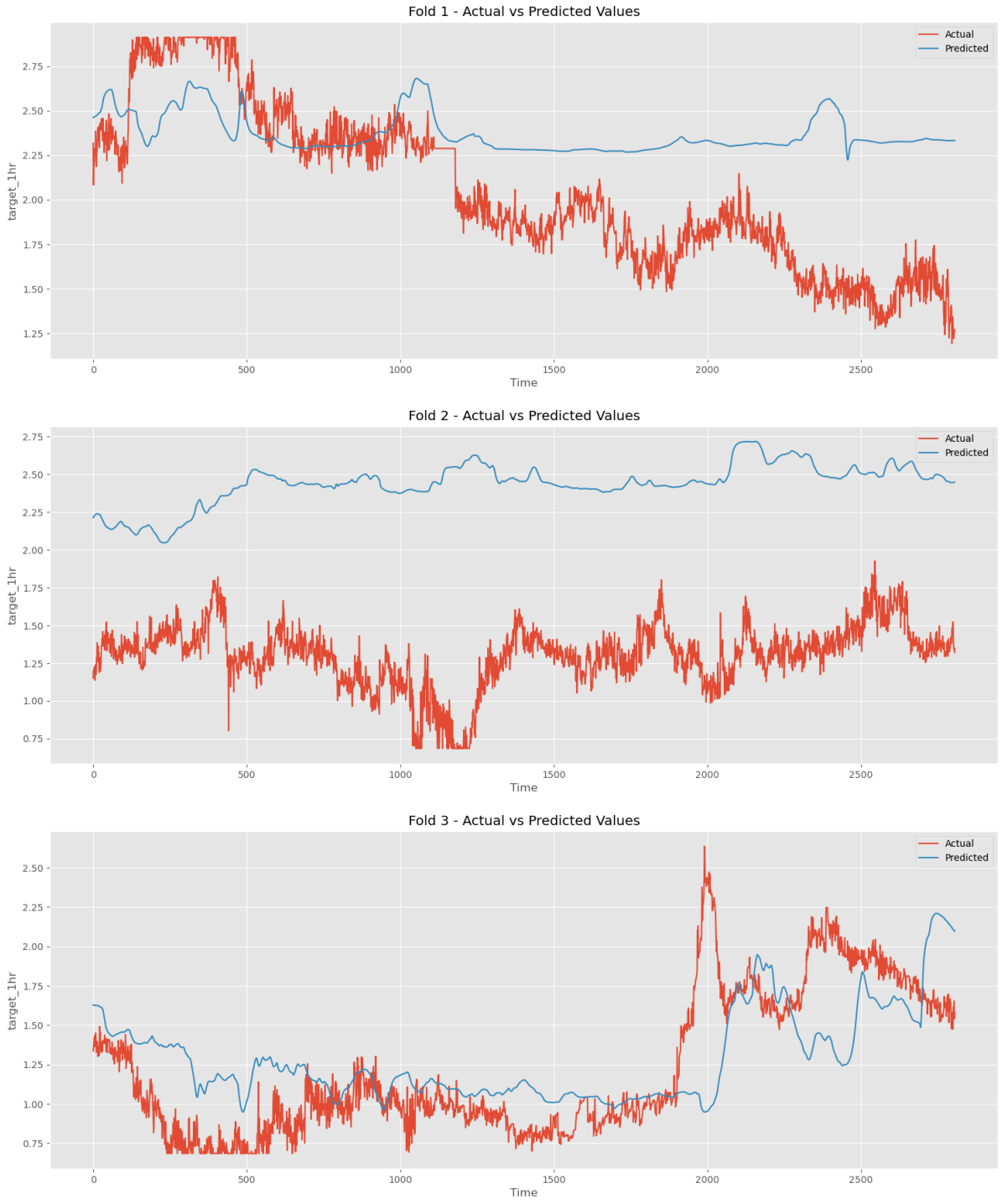
5.1.3 LSTM Model with 11 features

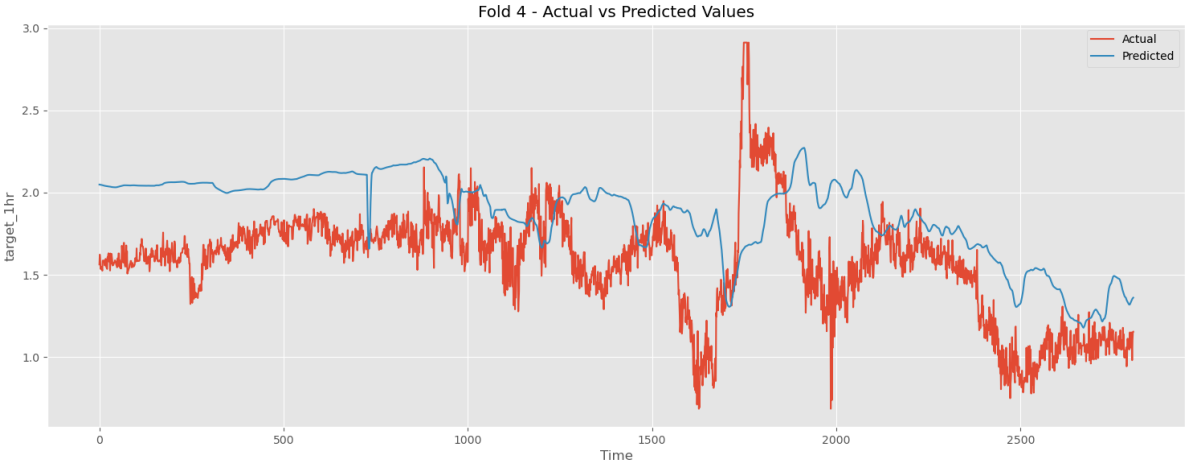
The result metrics for this model are as follows:

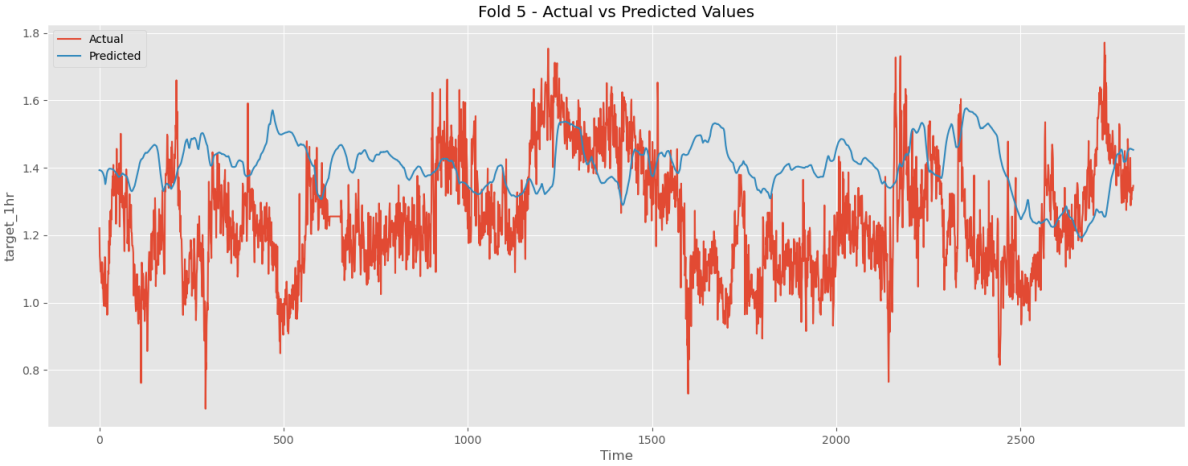
Table 9: LSTM 11 Feature Results

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
RMSE	0.5165	1.1664	0.3772	0.4131	0.2478	**0.5442**
MAE	0.4344	1.1376	0.2822	0.3586	0.2055	**0.4836**
R2	-0.3220	-32.1208	0.2821	-0.6325	1.1615	**-6.7909**

Figure 7: Folds 1-5 Eleven Feature




Fold 4 - Actual vs Predicted Values
Actual
Predicted
target_1hr
Time
3.0
2.5
2.0
1.5
1.0
0
500
1000
1500
2000
2500


Fold 5 - Actual vs Predicted Values
Actual
Predicted
target_1hr
Time
1.8
1.6
1.4
1.2
1.0
0.8
0
500
1000
1500
2000
2500

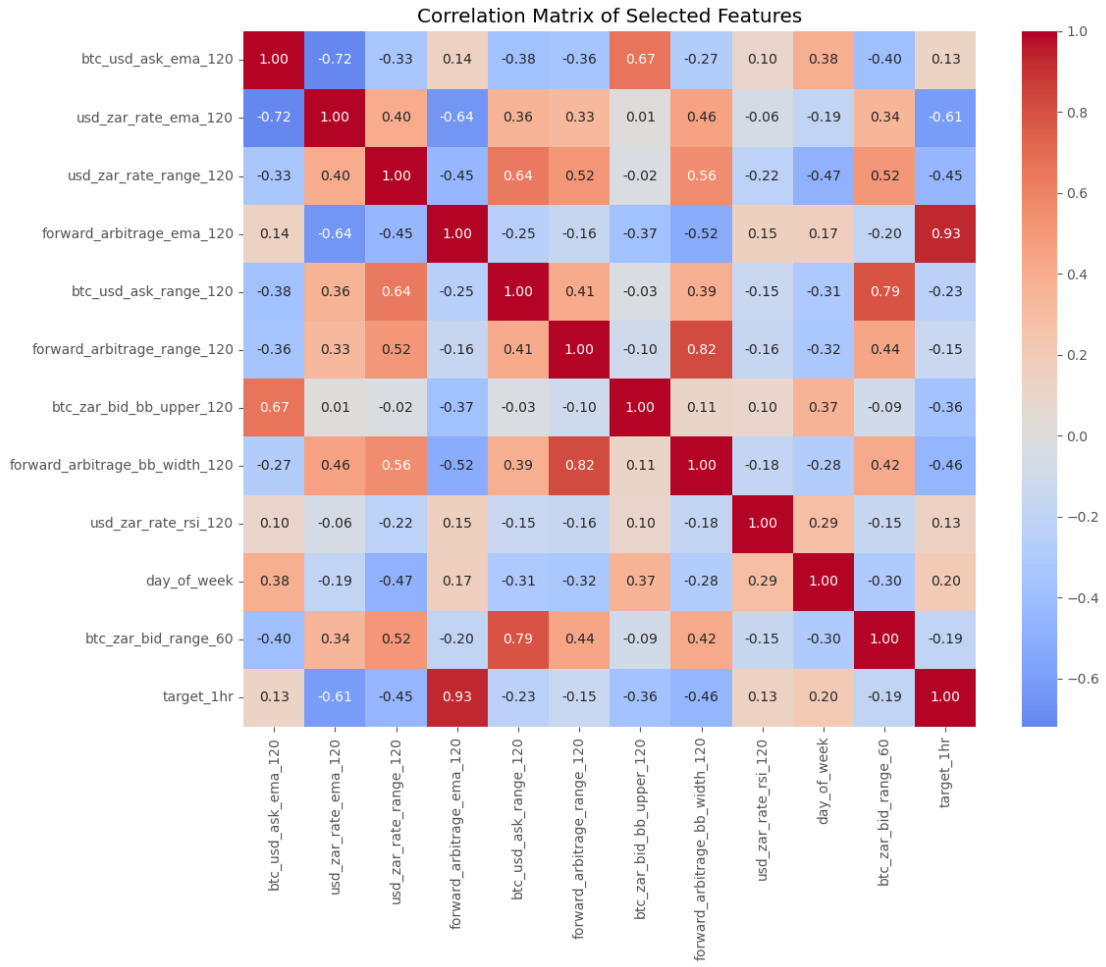


Figure 8: Correlation of Input Variables 11 Feature

Analysis:

The model shows a poor approximation of actual values, failing to pick up relevant trends. Where trends are correctly predicted, there is a large bias in the model. The model can predict arbitrage values within roughly 0.5 units. The strong negative R-squared indicates that the model doesn't perform better than just using the average as a predicted value. Overall the model performance is poor.

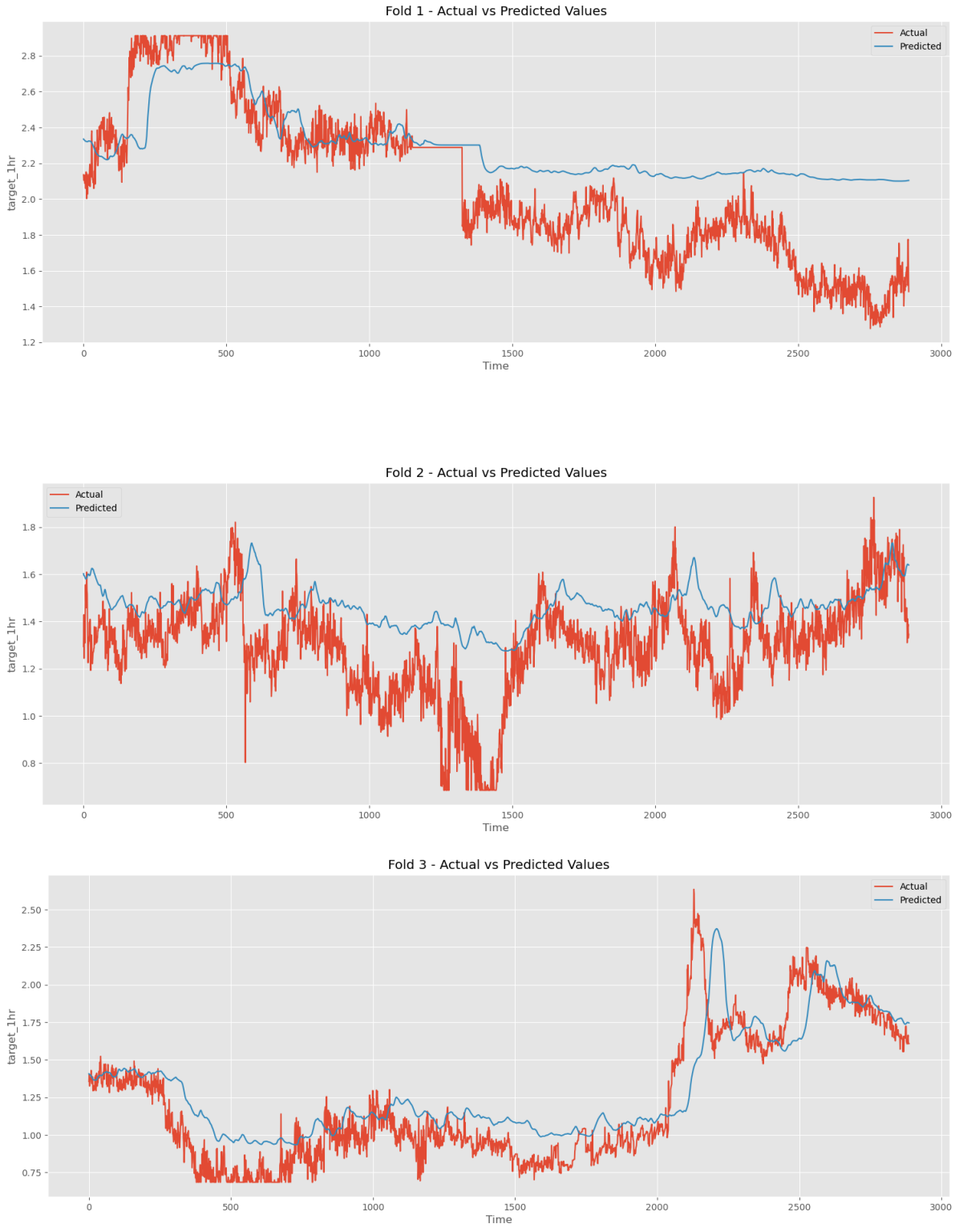
5.1.4 LSTM Model with single feature (optimized)

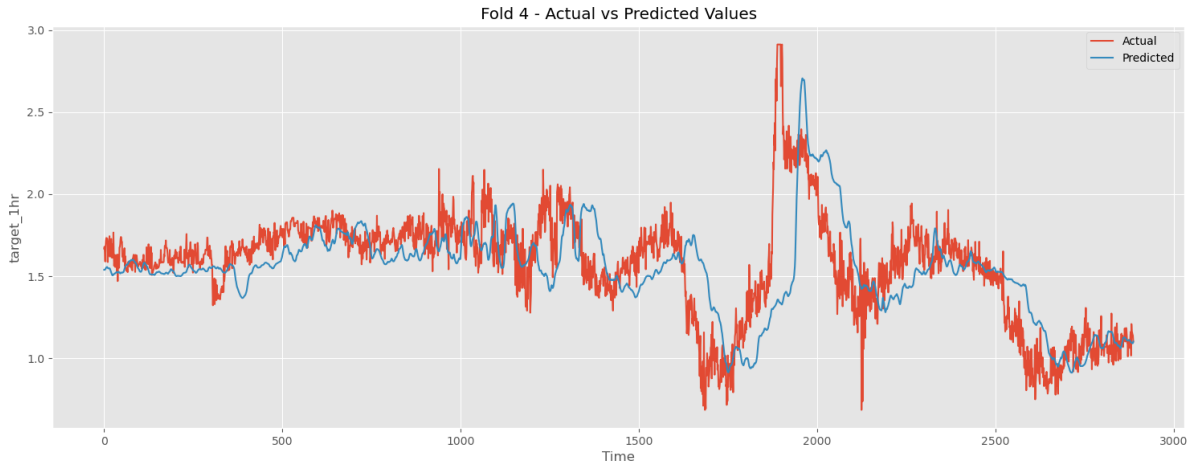
The result metrics for this model are as follows:

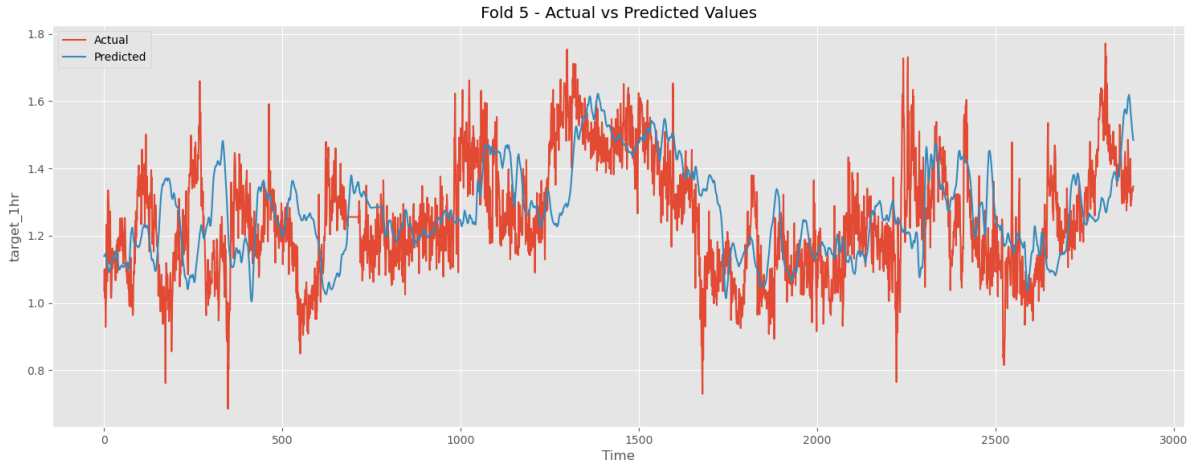
Table 10: LSTM Single Optimized Feature Results

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
RMSE	0.3390	0.2423	0.2401	0.2699	0.1664	**0.2515**
MAE	0.2720	0.1963	0.1815	0.1862	0.1327	**0.1937**
R2	0.3738	-0.4139	0.6983	0.3042	0.0130	**0.1951**

Figure 9: Folds 1-5 Single Optimized Feature







Analysis:

The model shows good approximation of actual values, picking up relevant trends. However, the model failed to correctly predict trends in fold 1 after the period for which forward fill was used. The model can predict arbitrage values within roughly 0.2 units. The positive R-squared indicates that the model performs better than just using the average as a predicted value. This was the best performing model out of all models.

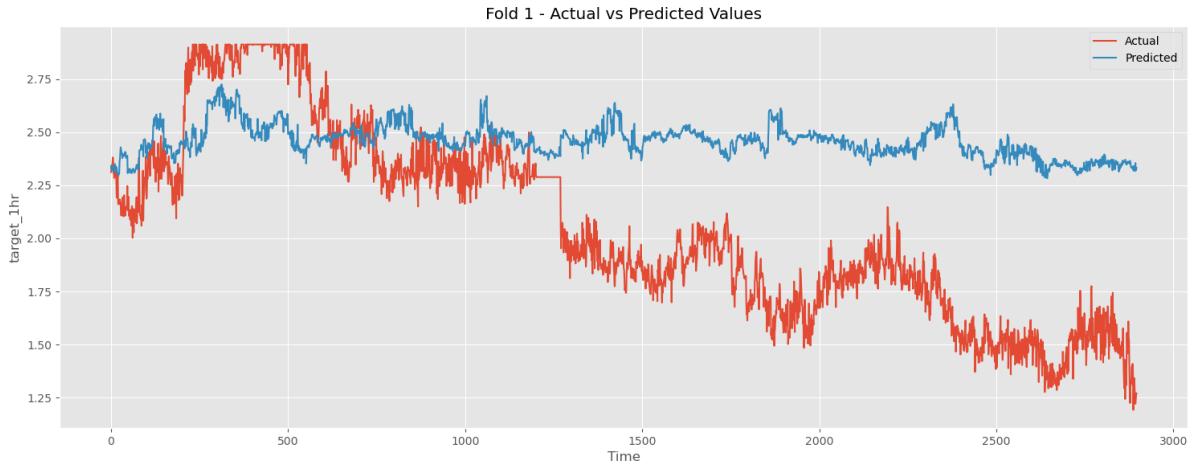
5.2 XGBoost Model

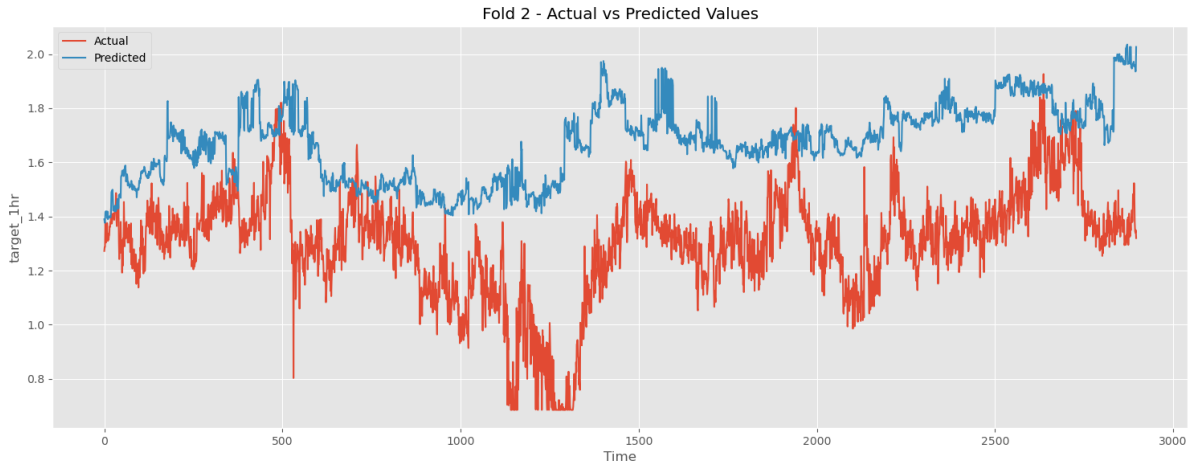
The result metrics for this model are as follows:

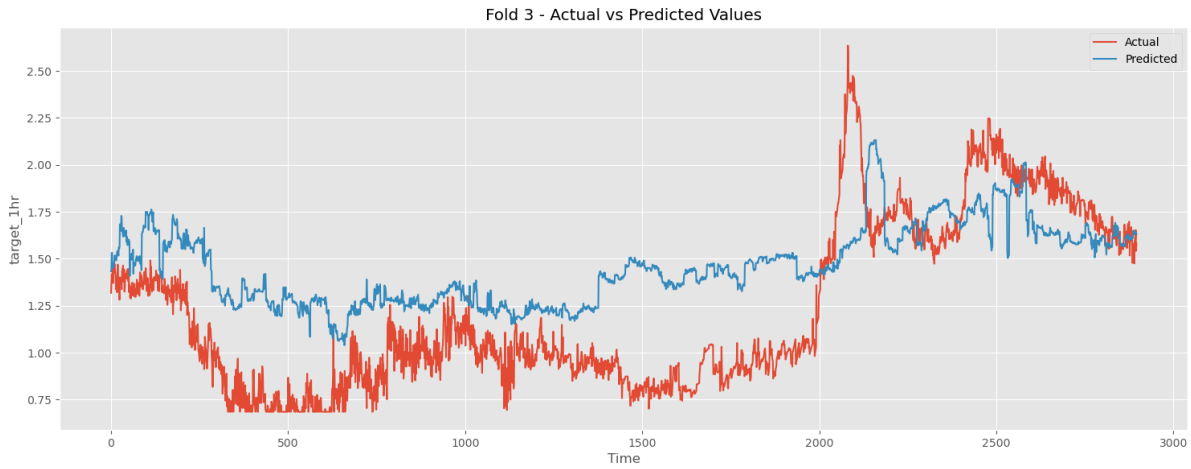
Table 11: XGBoost Results

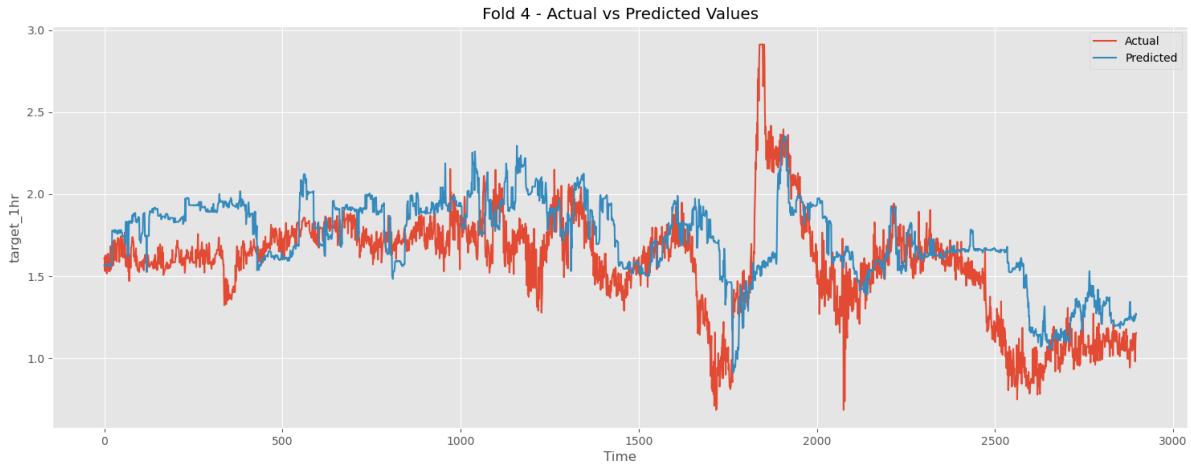
Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
RMSE	0.5729	0.4165	0.3889	0.3059	0.2309	**0.3830**
MAE	0.4876	0.3719	0.3399	0.2312	0.1898	**0.3241**
R2	-0.6726	-3.3408	0.2157	0.0784	-0.8840	**-0.9207**

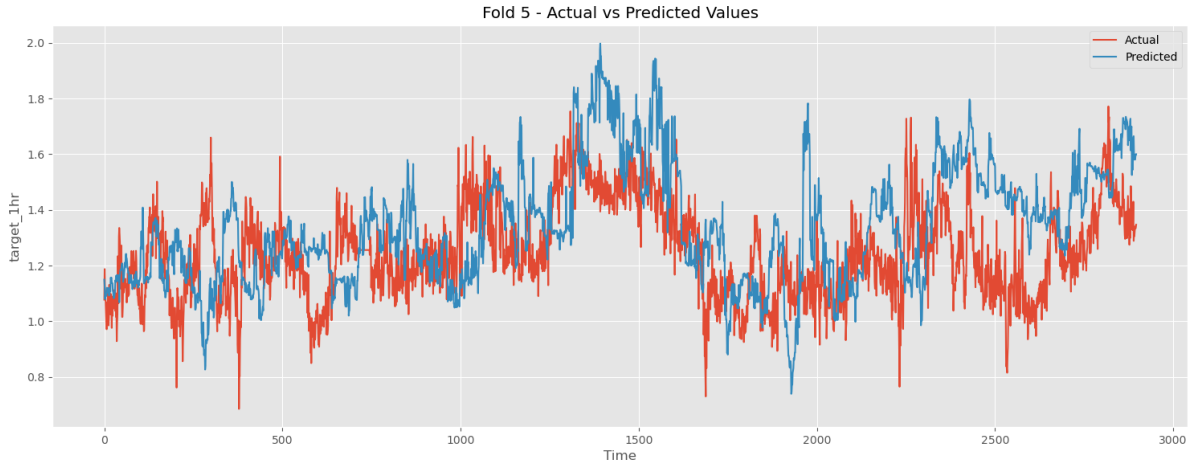
Figure 10: Folds 1-5 XGBoost











Analysis:
The model shows a poor approximation of actual values, failing to capture trends. Fold 4 shows promising performance but aside from that the model fails to generalise arbitrage movement. The model can predict arbitrage values within roughly 0.35 units. The negative R-squared indicates that the model doesn't perform better than just using the average as a predicted value. Overall the model performance is poor

5.3 Results Analysis and Comparison

Overall, the general model performance was quite poor. Aside from the single feature optimized LSTM model, all models had a negative R-squared and failed to predict arbitrage values with consistent accuracy. The single feature optimized LSTM model is the only model that displayed a positive R-squared and could be considered for deployment in predicting arbitrage values. A larger dataset and further analysis of model structure, features, and parameters is required to improve the poor performing models. The single feature model could also benefit from further research into optimal characteristics for prediction. The optimized single feature model's better performance does make sense considering that the single feature it uses, the "forward_arbitrage" values, already capture the information of bid, ask, and USD/ZAR exchange rate. LSTM

models are good at learning temporal patterns, and it is possible that the additional features only added extra noise and didn't assist with prediction.

6. Business Impact Analysis

6.1 Interpretation of Results in Business Context

This project focused on forecasting one hour ahead Bitcoin price differences (forward arbitrage) between Kraken BTC/USD and Luno BTC/ZAR, using LSTM and XGBoost models. These predictions were intended to help traders exploit cross-exchange arbitrage opportunities. The best-performing model was an LSTM using a single optimized feature (past forward arbitrage values), achieving an average RMSE of roughly 0.25, MAE of roughly 0.19, and a positive R-squared of 0.1951. All other models, including the feature-rich LSTM and XGBoost variants, performed poorly, with negative R-squared values, suggesting they often underperformed relative to a mean prediction.

Only the optimized single feature LSTM model provides potentially actionable prediction for real-time arbitrage. The rest are not reliable enough for deployment due to inconsistency and prediction errors.

6.2 Economic Implications

Profit potential: Accurate predictions of price differences one hour in advance could yield consistent profits via arbitrage, especially during high spread windows.

Risk management: Since arbitrage is not risk-free, due to factors like transfer delays or volatility, having a prediction model helps optimize trade timing and avoid unprofitable windows.

Cost consideration: Given that transaction fees, slippage, and withdrawal times were excluded from the model, actual profit margins might be significantly lower than predicted. These will need to be included before deployment.

If integrated into a trading system, even a modest average arbitrage prediction accuracy could generate profit.

6.3 Practical Implementation Considerations

If the model were to be practically implemented, the following should be considered:

Real time data: Maintaining live API connections to Kraken, Luno, and Forex feeds with robust error handling to avoid missing data.

Latency and execution: Bitcoin transfers can take up to 1 hour, so systems must predict ahead of time accurately and initiate trades within short execution windows.

Fees: A fee calulation function should be implemented to subtract trading, withdrawal, and currency conversion fees from predicted arbitrage spreads.

Cloud deployment: Host on a cloud platform with scheduled jobs and GPU acceleration for real time performance.

6.4 Limitations and Potential Improvements

Limitations:

Data window: Only two weeks of minute level data was used.

Poor feature performance: Adding more features decreased model performance, implying either noise or poor feature relevance.

Evaluation metrics: Even the best model had a low R-squared, meaning much of the price variation is still unexplained.

No fee modeling: Omitting trading and withdrawal fees may lead to overestimated profit potential.

Single cryptocurrency: Only BTC was considered. No ETH, LTC, or altcoins were included.

Potential Improvements:

Expand dataset: Incorporate at least 3 months of historical minute-level data to enhance model robustness.

Feature engineering refinement: Apply PCA, autoencoders, or attention mechanisms to improve feature signal quality.

Include fees & transfer delays: Model realistic slippage, time delays, and exchange limits to filter out impractical trades.

Alternative models: Experiment with Transformers, GRU, or Ensemble Models to better capture both time-dependencies and nonlinearities.

Multi-coin arbitrage: Include other cryptocurrencies and even triangular arbitrage routes to broaden strategy scope.

7. Technical Documentation

7.1 Dependencies

The required packages are in the “requirements.txt” file in the accompanying zip file. The required dataset “realtime_crypto_data.db” is in the accompanying zip file.

7.2 Code Structure Overview

data_preparation.py

1. Imports and Initial Setup

2. Data Loading and Initial Processing

- Load data from SQLite database

- Convert timestamp to datetime index

- Select relevant columns

- Create data description CSV

3. Data Preprocessing

- Forward fill missing values

- Create target variable (1-hour forward arbitrage)

- Create depth features

- Winsorize data (handle outliers)

4. Feature Engineering Functions

def create_technical_features():

  - Create time-based features (hour, day_of_week)

  - Generate technical indicators for multiple timeframes:

    * Moving Averages (SMA, EMA)

    * Bollinger Bands

    * RSI

    * Momentum

    * Volatility

    * Price Range

    * Rate of Change

  - Create cross-market features

- Create volume-based features

5. Feature Selection Functions

def analyze_feature_importance():

- Calculate mutual information scores
- Handle correlated features
- Create visualizations
- Return selected features

def analyze_xgboost_features():

- Train initial XGBoost model
- Calculate feature importance
- Handle correlated features
- Create visualizations
- Return selected features

6. Final Dataset Creation

- Create datasets for different models:
  * df_lstm
  * df_xgboost
  * df_1v

models.py

1. Imports

2. LSTM Model Functions

def simple_lstm():

- Time series cross-validation

- Sequence creation
- Model training
- Performance evaluation
- Visualization

def lstm_optimized():
- Uses optimized hyperparameters
- Similar structure to simple_lstm
- Enhanced performance tracking

# 3. XGBoost Model Functions
def xgboost_model():
- Time series cross-validation
- Feature scaling
- Model training
- Performance evaluation
- Feature importance analysis

# 4. Hyperparameter Optimization Functions
def optimize_lstm_hyperparameters():
- Bayesian optimization setup
- Parameter space definition
- Cross-validation
- Results storage and visualization

def optimize_xgboost_hyperparameters():
- Bayesian optimization setup
- Parameter space definition

- Cross-validation

- Results storage and visualization

5. Main Execution

if __name__ == "__main__":

- Run hyperparameter optimization

- Train and evaluate models

7.3 Instructions for Reproduction

- Ensure import directories are correct
- Run code, it is setup to reproduce all elements

8. References

1. CCXT. (2024). *CCXT: Cryptocurrency trading library* [Python library]. https://github.com/ccxt/ccxt
2. Alpha Vantage. (2024). *Alpha Vantage API* [Data set]. https://www.alphavantage.co

Appendix

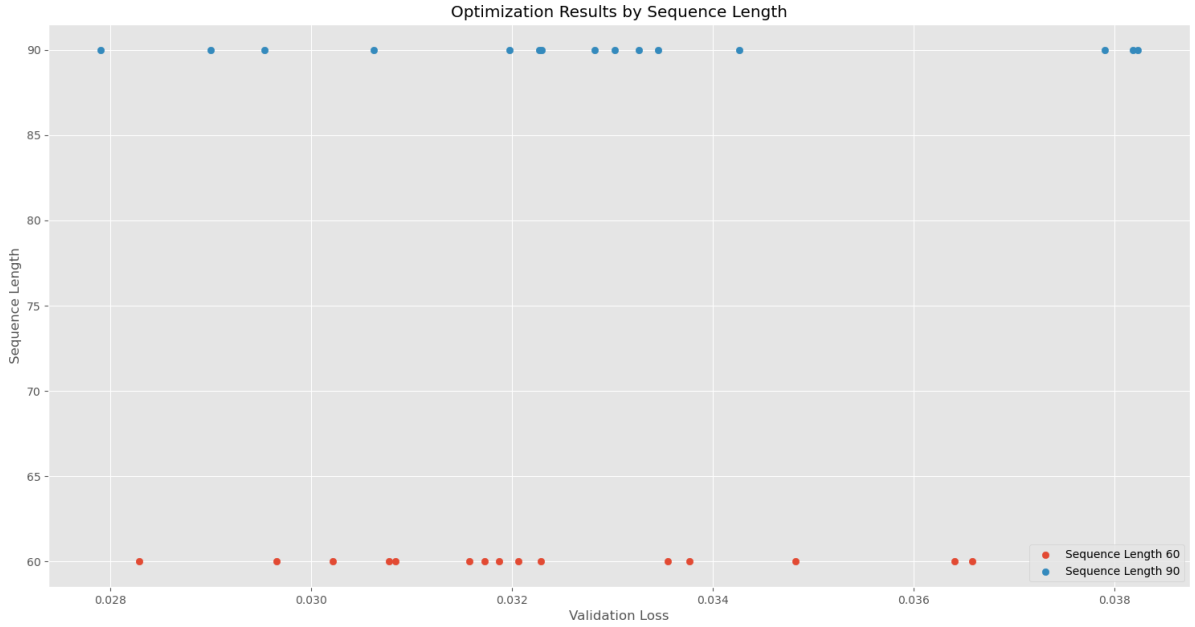
LSTM Model Single Feature Optimized

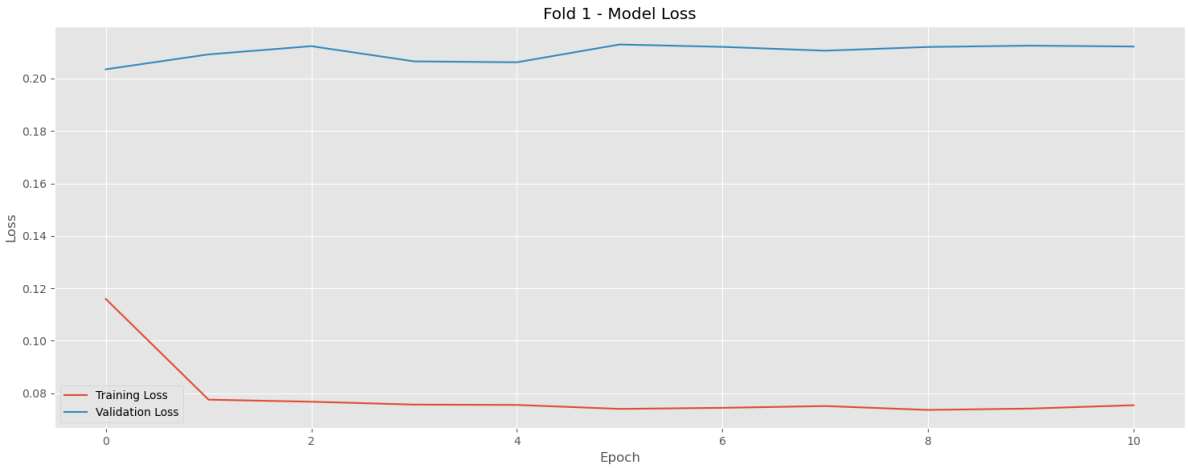
Optimization Results:

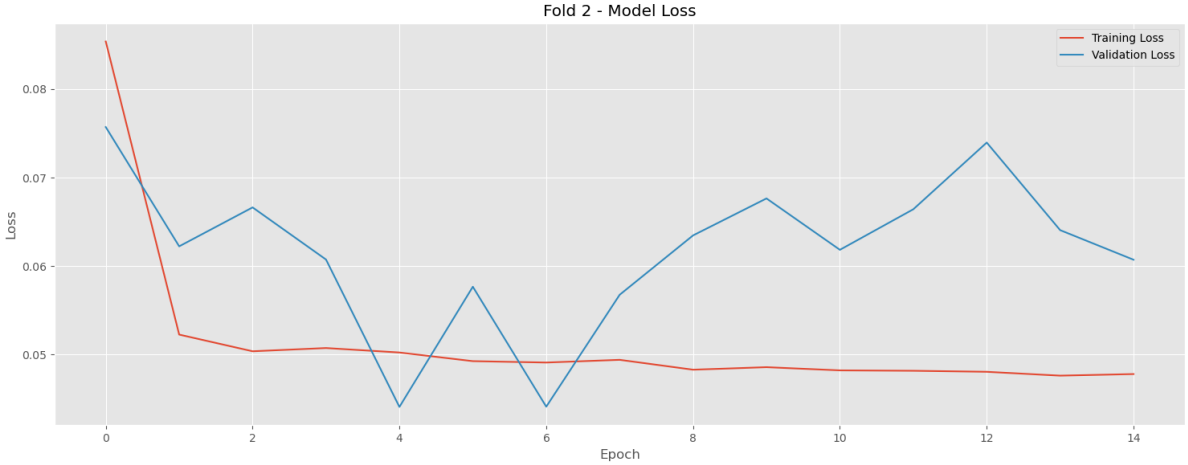
sequence_length	units_layer1	units_layer2	dropout_rate	learning_rate	batch_size	score
60	108	25	0.2559382000545539	0.0006371651421518385	46	0.031729172294338547
60	42	38	0.1667417222780437	0.0002285801361297467 3	53	0.034821250166550223
60	37	51	0.2877105418031 5003	0.0001007008892569129	64	0.036408221349120 14
60	91	45	0.1014132610439 4349	0.0001207561825372741 9	49	0.031575820719202 36
60	70	18	0.2947511037682 9183	0.0003094942063872738 6	35	0.030840690558155377
60	91	34	0.2966461771613 5766	0.0005200866039231821	60	0.028283731391032536
60	97	38	0.1026529922319 7331	0.0009479815801163677	50	0.033771288891633354
60	69	17	0.1461787651244 2982	0.0003169229194234106	54	0.036583917215 4665
60	91	56	0.1346729307015 5442	0.0004519545468159168	38	0.029651685307423275
60	105	36	0.1415883325736378	0.0006109302950379925	33	0.031869007274508476
60	105	64	0.2941089600530977	0.0001874478853392958 4	52	0.030214040850599606

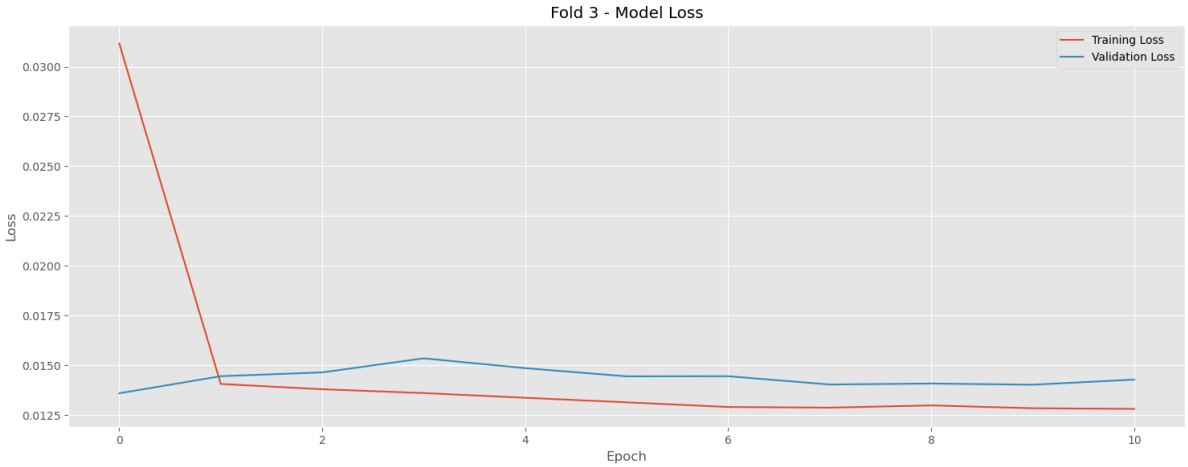
60	127	64	0.11454286584956724	0.0004151541560760305	44	0.032061267644166946
60	75	62	0.2796644300645029	0.0008547077334033816	41	0.0322896763 6823654
60	92	28	0.2904701750 4539263	0.0003485034127476289	32	0.0335510640 0946776
60	102	23	0.3	0.0005517454999001015	64	0.0307746790 3494835
90	108	25	0.2559382000545539	0.0006371651421518385	46	0.0322721184 5417818
90	42	38	0.1667417222780437	0.0002285801361 2974673	53	0.0382375537 10738816
90	37	51	0.2877105418 0315003	0.0001007008892569129	64	0.0322983749 2108345
90	91	45	0.1014132610 4394349	0.0001207561825372 7419	49	0.0332671149 32338394
90	70	18	0.2947511037 6829183	0.0003094942063872 7386	35	0.0328243014 7131284
90	91	34	0.2966461771 6135766	0.0005200866039231821	60	0.0334545690 56630135
90	97	38	0.1026529922 3197331	0.0009479815801163677	50	0.0295371655 3747654
90	69	17	0.1461787651 2442982	0.0003169229194234106	54	0.0379084746 04288735
90	91	56	0.1346729307 0155442	0.0004519545468159168	38	0.0319769854 3469111
90	105	36	0.1415883325736378	0.0006109302950379925	33	0.0342680122 7033138

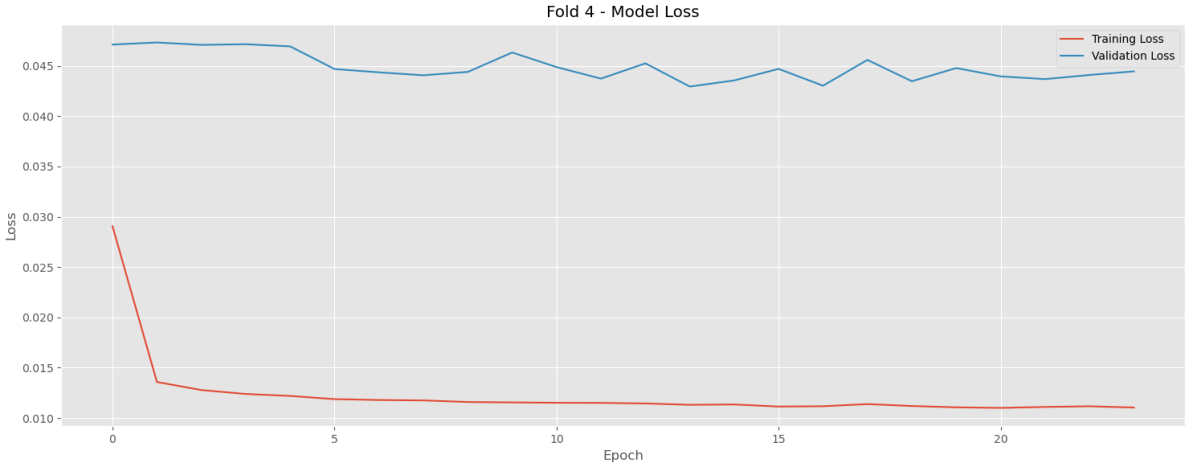
90	55	61	0.21561842415864504	0.0009992460702945634	43	0.03302802766362826
90	104	64	0.23360052491561492	0.00014128439258985422	45	0.02790195805331071
90	37	18	0.22160350425790078	0.00010520745815899579	56	0.03818903677165508
90	107	60	0.27164570860284720	0.00099763213608041480	64	0.02899479928116003 8
90	76	46	0.18521752480261863	0.00099988065 1142288	57	0.03062007141609 9865

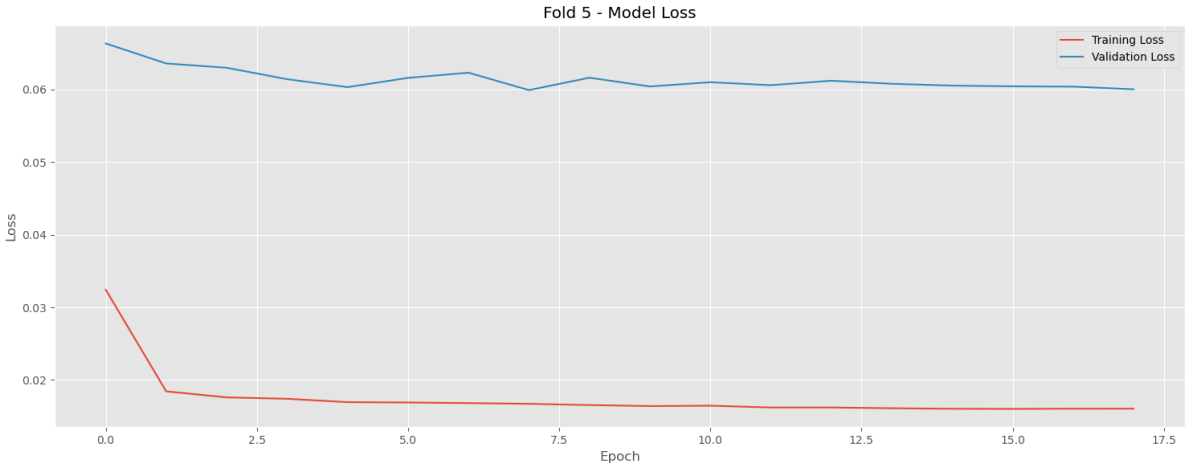



Fold 1 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0
2
4
6
8
10

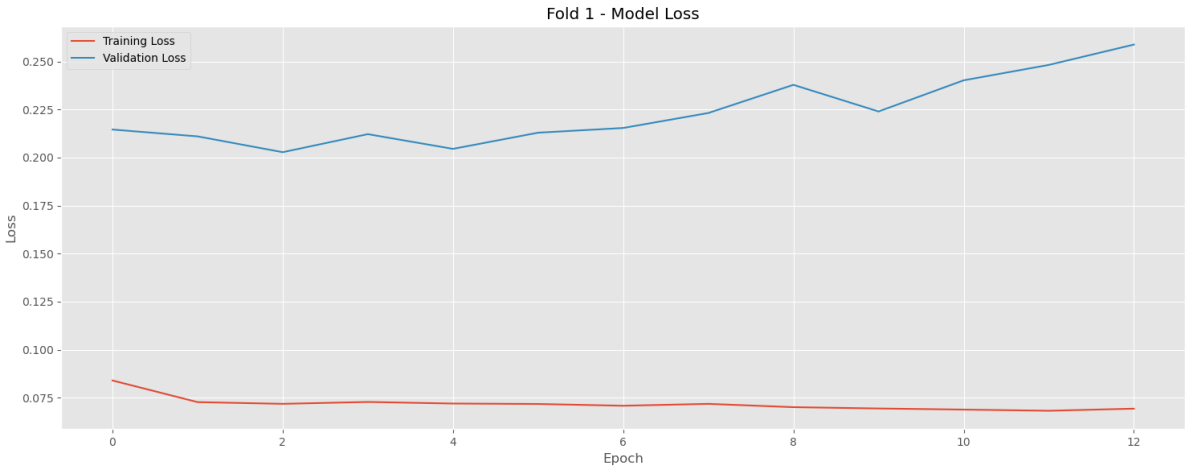

Fold 2 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.08
0.07
0.06
0.05
0
2
4
6
8
10
12
14

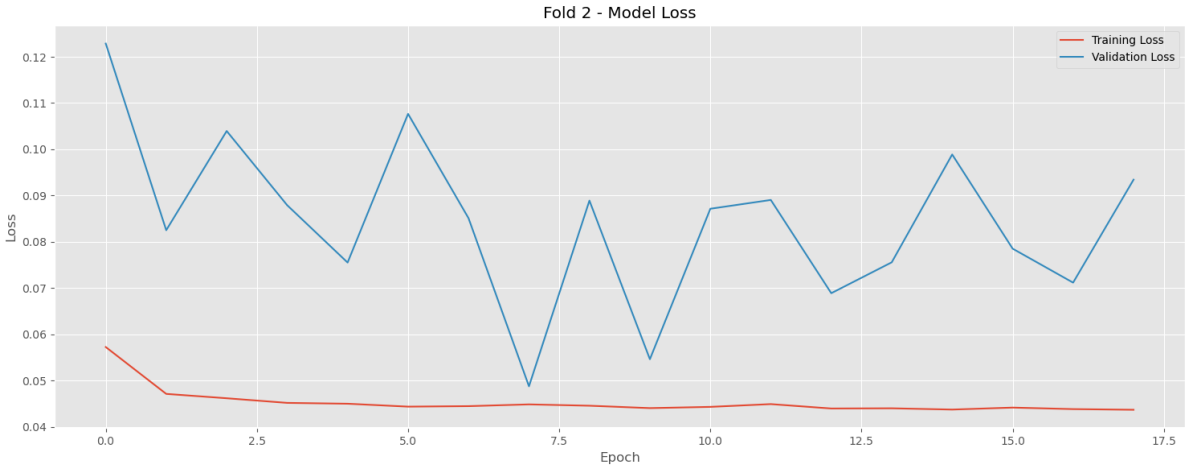

Fold 3 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0.0150
0.0125
0
2
4
6
8
10

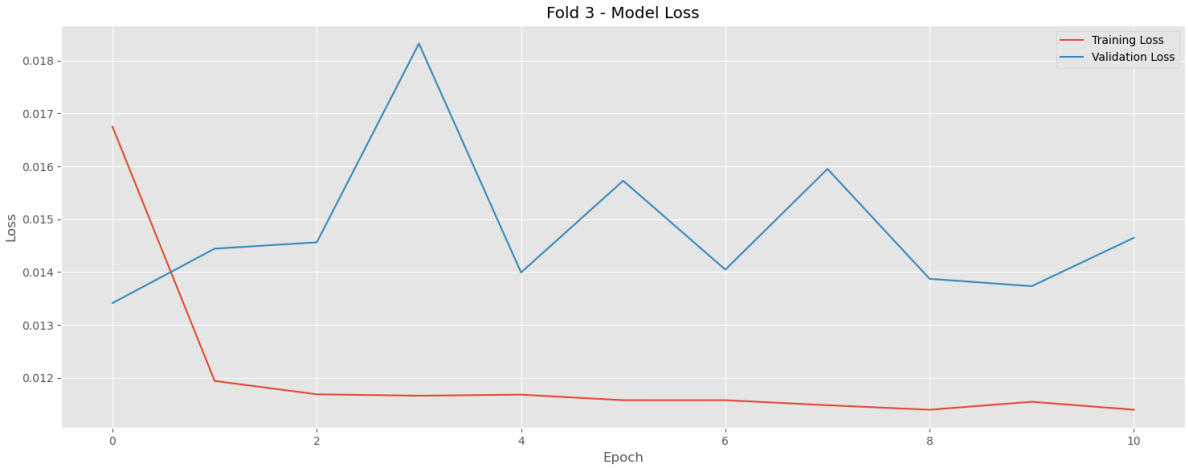

Fold 4 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0
5
10
15
20

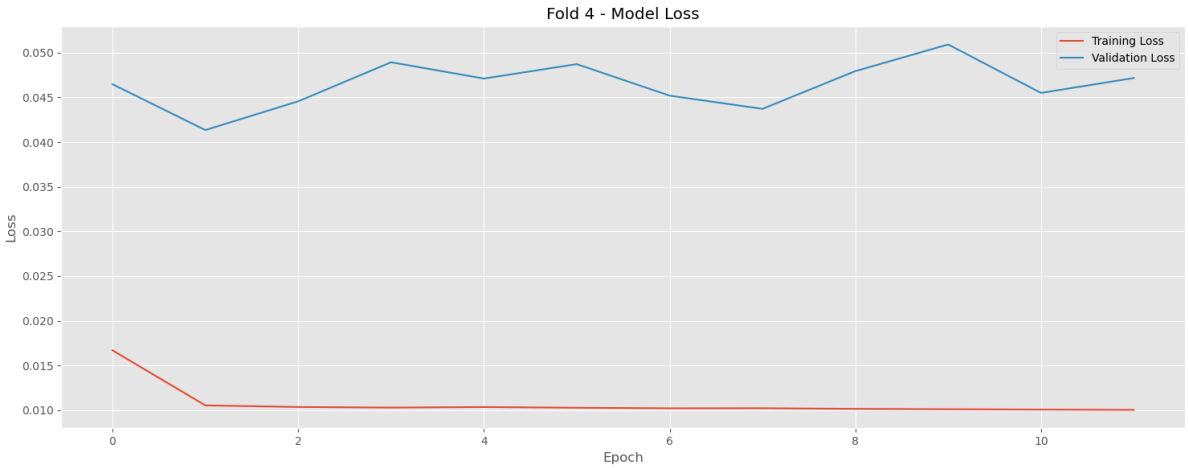

Fold 5 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.06
0.05
0.04
0.03
0.02
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

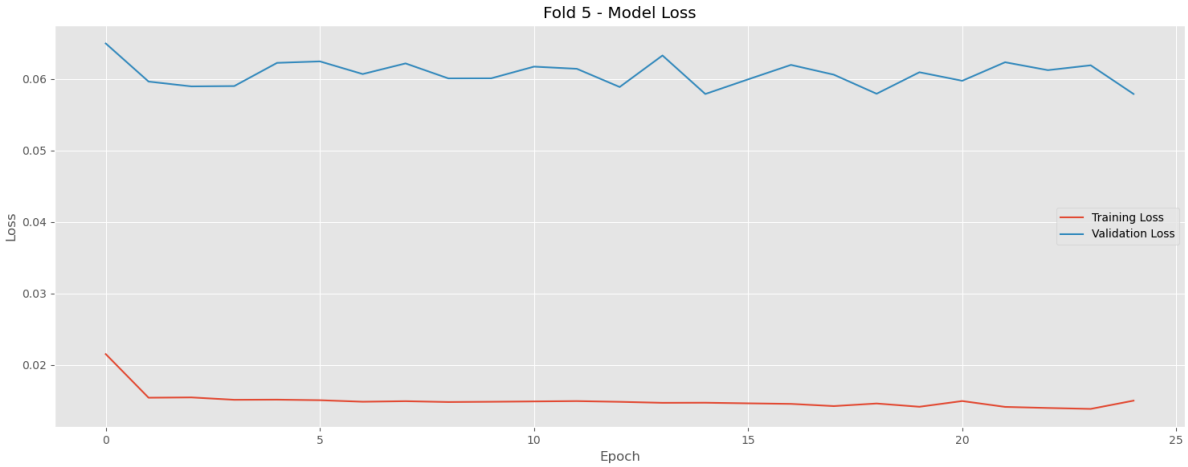
LSTM Model Single Feature



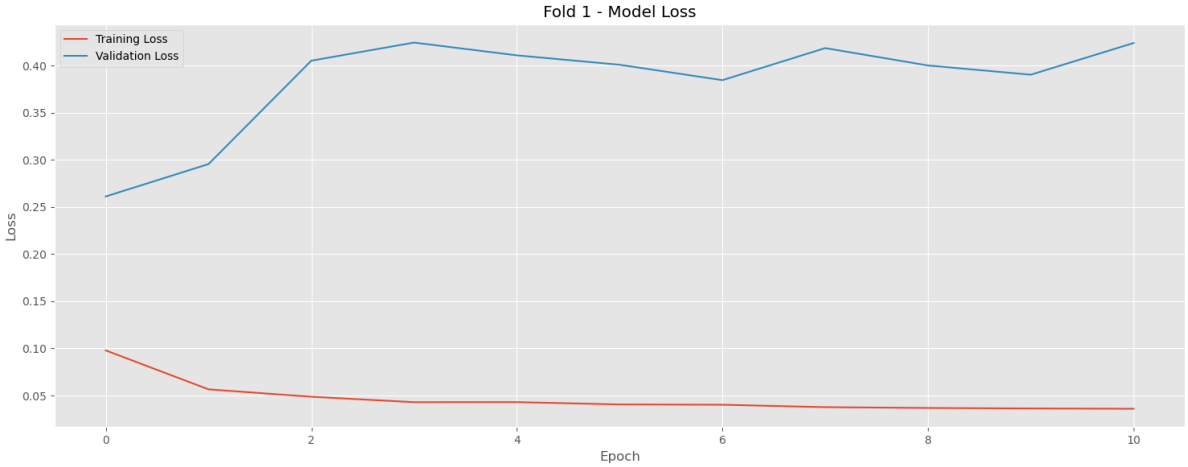


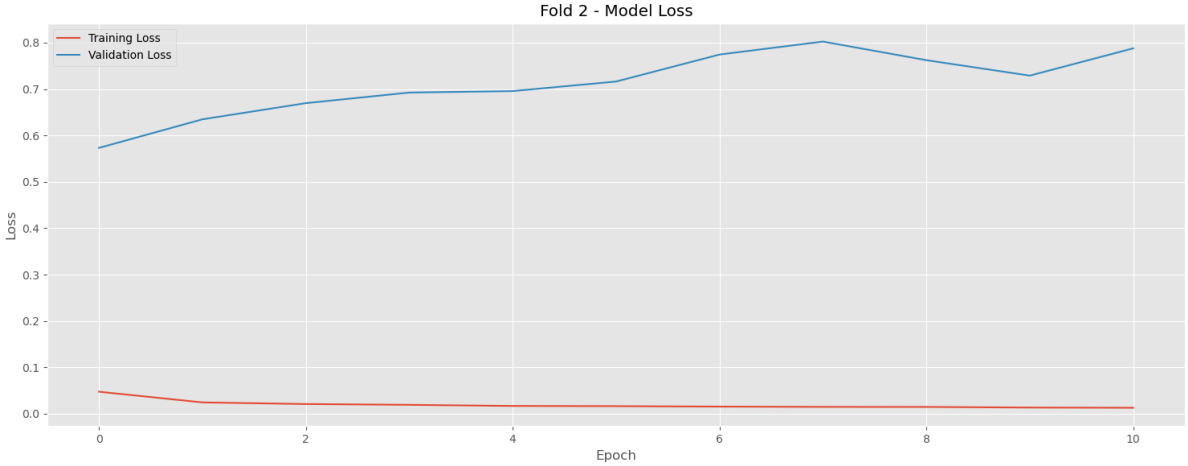


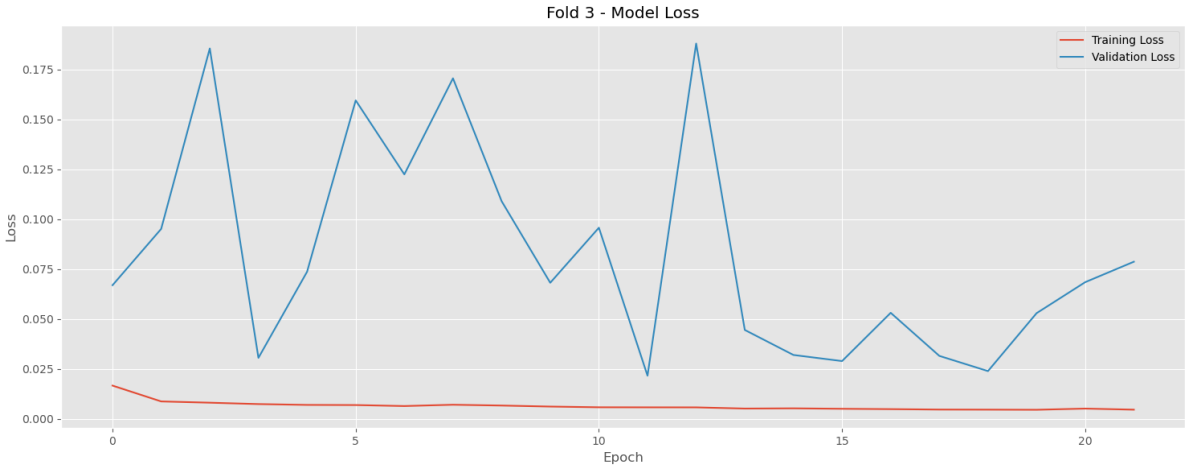

Fold 4 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.050
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0
2
4
6
8
10

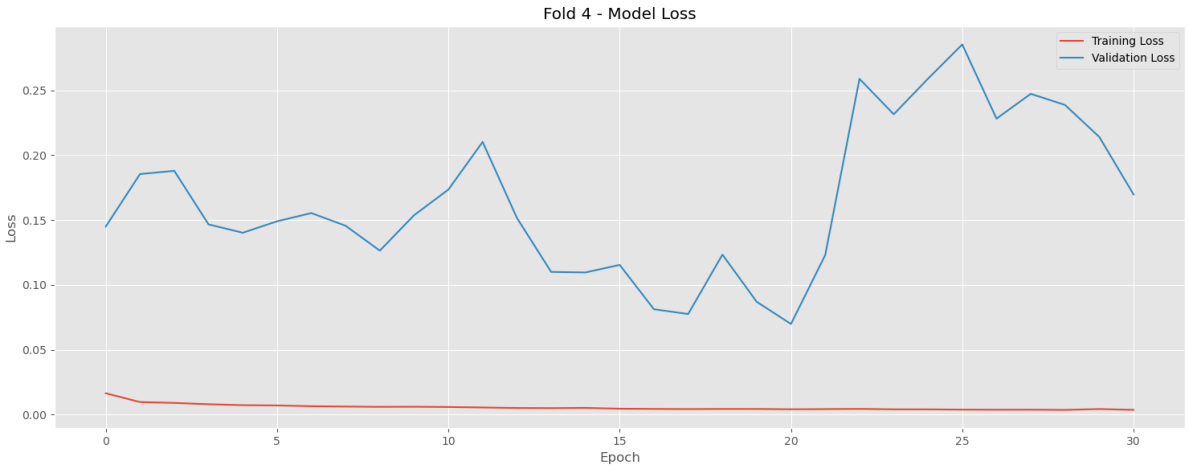

Fold 5 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.06
0.05
0.04
0.03
0.02
0
5
10
15
20
25

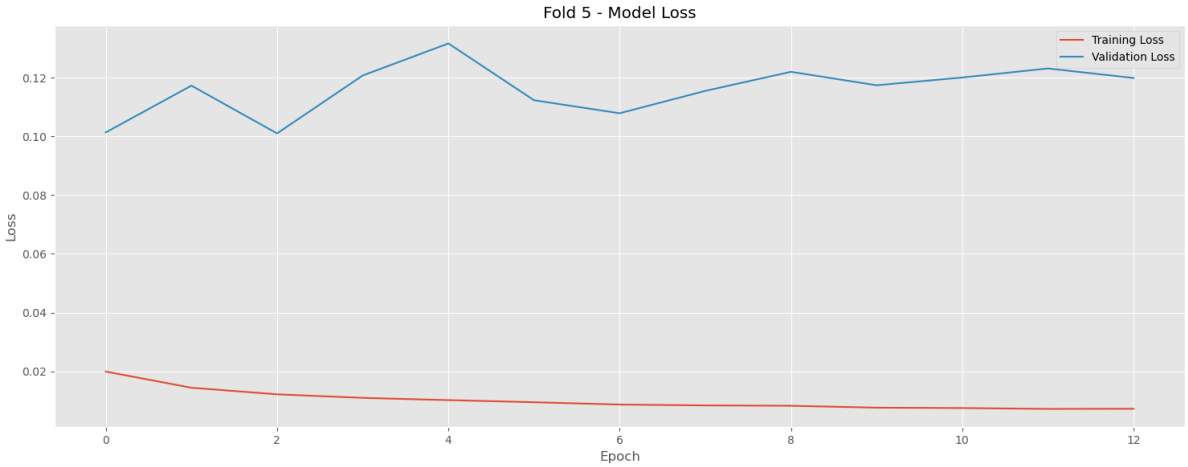
LSTM Model Five Feature



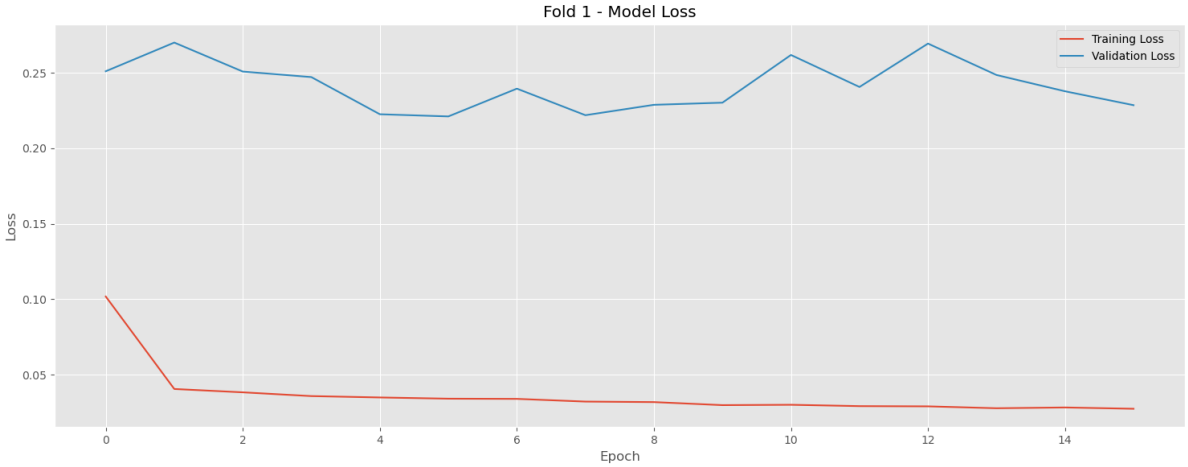


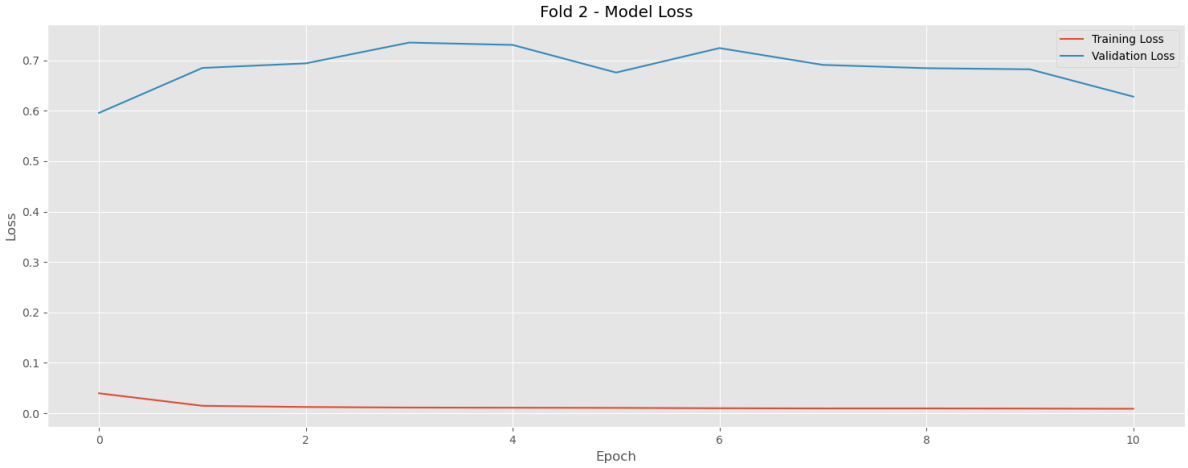


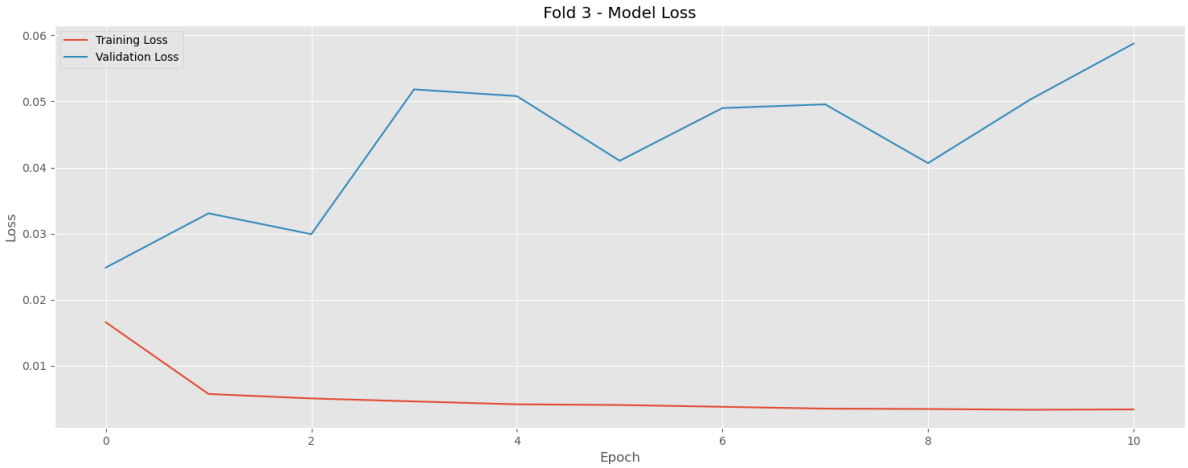

Fold 4 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
30

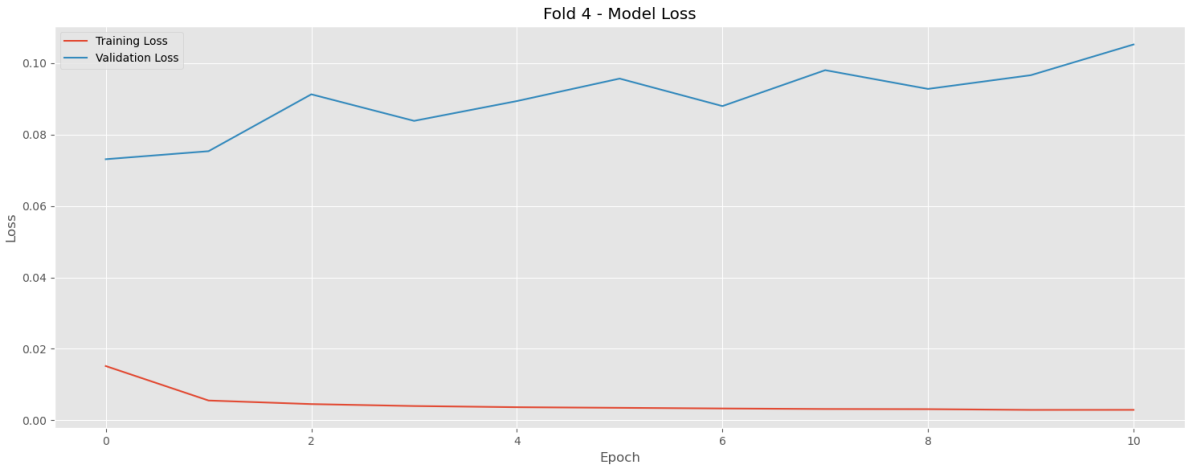

Fold 5 - Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.12
0.10
0.08
0.06
0.04
0.02
0
2
4
6
8
10
12

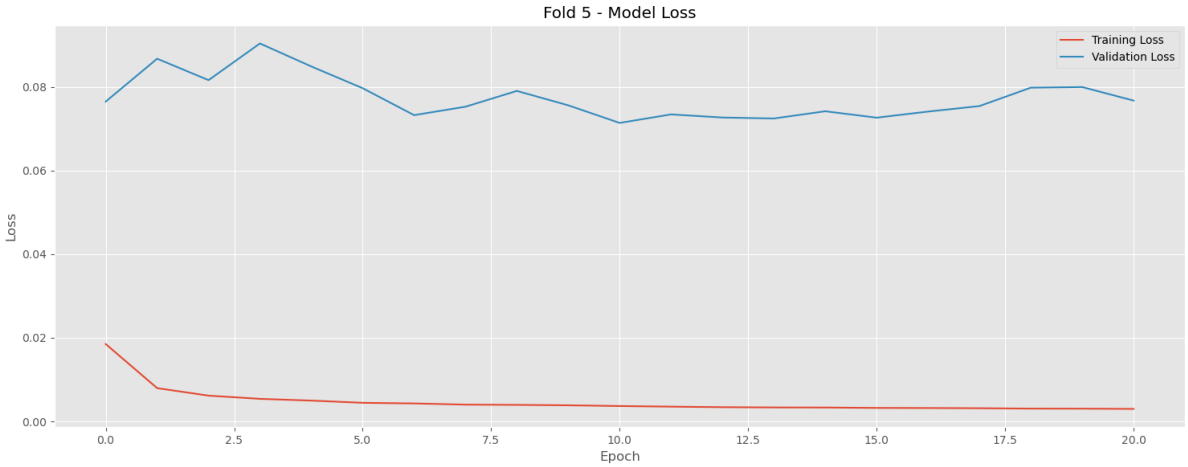
LSTM Model Eleven Feature











XGBoost Model

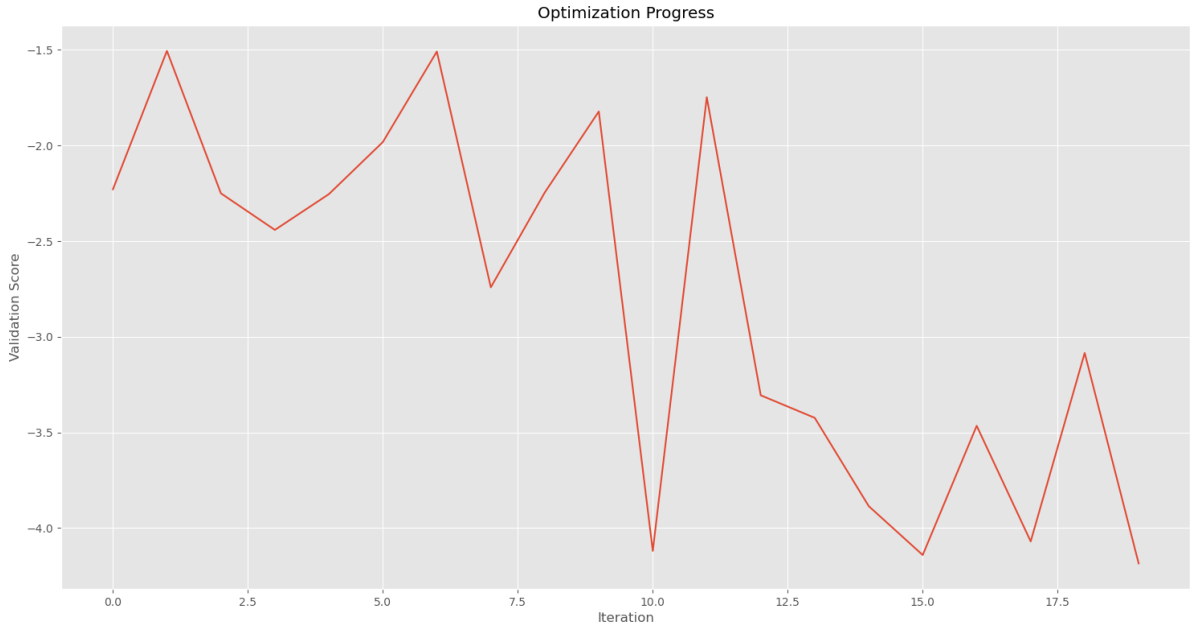
Optimization Results:

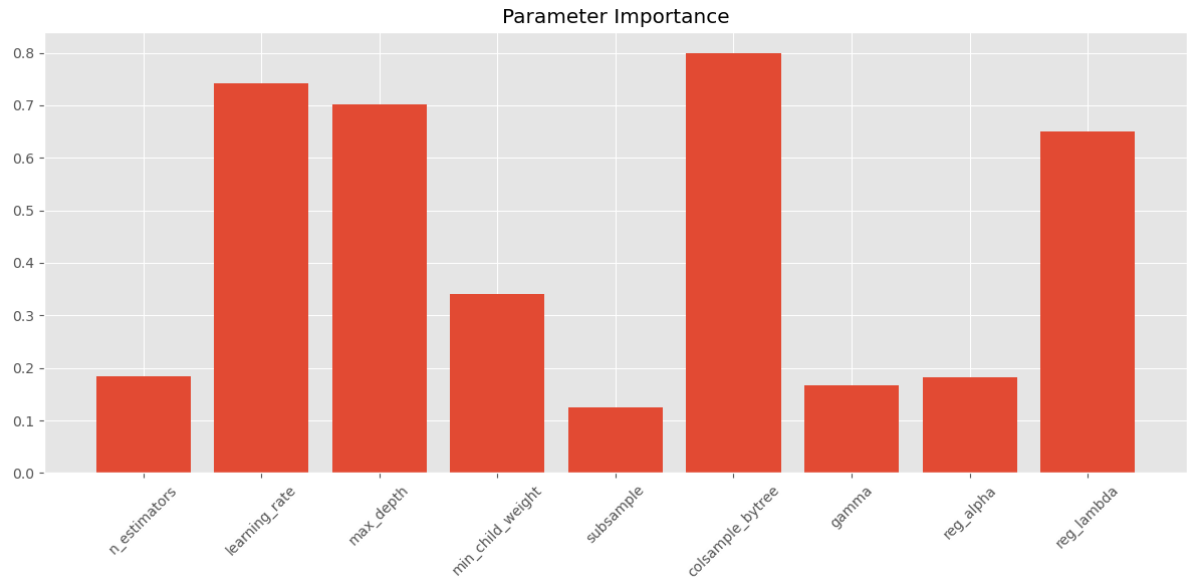
n_estimators	learning_rate	max_depth	min_child_weight	subsample	colsample_bytree	gamma	reg_alpha	reg_lambda	score
249	0.018661881481387678	8	6	0.7783331011414365	0.6399899663272012	0.2296244459829363	0.3337086111390219	0.1428668179219408	-2.2289877197361503
213	0.012115092723815247	8	9	0.6003115063364057	0.9968846237164871	0.3087407548138583	0.6116531604882281	0.0070663052197408	-1.5054551575957476

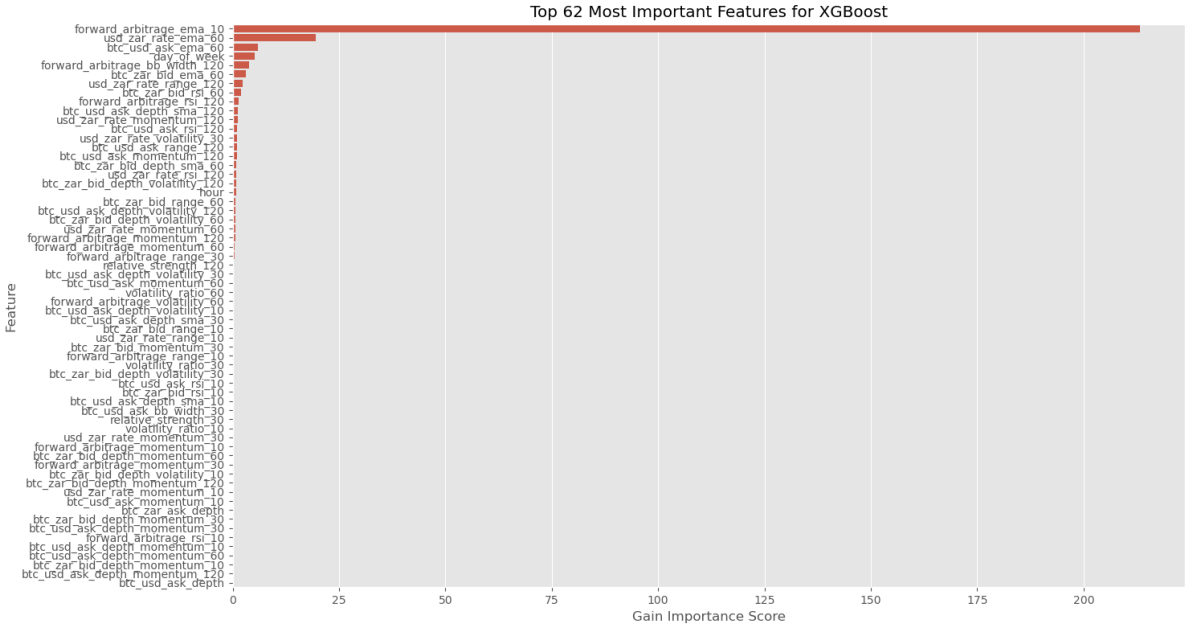
56	0.05958 7587766 28971	6	1	0.9895 022075 365838	0.6931 085361 721216	0.04530 321726 641041	0.61838 600933 30874	0.38246 1991267 16285	-2.2499 216152 61877
296	0.04891 7628051 31221	9	7	0.7801 997007 878172	0.6053 059844 639466	0.47110 087784 24265	0.56328 821784 55394	0.38541 6502539 9162	-2.4412 223211 343
54	0.02193 0986997 479973	5	7	0.8439 986631 130484	0.9332 779646 944658	0.08668 232675 388605	0.39106 060757 32409	0.18223 6087788 06236	-2.2542 505238 415123
239	0.04246 2619176 491884	4	6	0.6125 253169 822235	0.9369 139098 379995	0.22487 706668 48829	0.39515 023600 181454	0.92665 8865793 7944	-1.9818 067415 71887
232	0.03036 2694519 01499	7	6	0.9844 688097 397397	0.9378 135394 712607	0.37366 005506 86905	0.53969 213238 90799	0.58675 1165663 8483	-1.5088 653107 284455
291	0.07882 4856646 94613	5	4	0.6661 067756 25201	0.6062 545626 964776	0.21170 074035 318487	0.39488 151817 55698	0.29348 8174718 0382	-2.7413 215109 087314
54	0.01966 5923318 21354	8	8	0.8423 839899 124046	0.9705 203514 053397	0.32553 851275 09723	0.91495 967554 3781	0.85003 8577789 7995	-2.2451 891791 423306
162	0.01383 3507535 062179	6	7	0.8663 689426 469987	0.8365 191150 830908	0.13736 089649 50321	0.56124 342584 77013	0.38292 6874753 78995	-1.8221 701569 069786

300	0.2999999999999993	3	1	0.6	0.6	0.0	0.0840980029620168	1.0	-4.119191152074432
300	0.2999999999999993	10	10	0.6	0.9772958056050136	0.0	0.34294535282676375	1.0	-1.747673813855086
300	0.01	3	10	0.6	0.6	0.0	0.0	1.0	-3.305737589093088
300	0.2999999999999993	3	1	0.6	0.6	0.0	0.0	1.0	-3.423628289963452
166	0.2999999999999993	3	6	0.602785403384209	0.6	0.4721288345937615	0.7222025616209528	1.0	-3.885094494231472 7
50	0.2999999999999993	3	9	0.9415171630836012	0.6	0.12188005483361712	0.5059274092346265	1.0	-4.140725269754384
50	0.2999999999999993	3	10	0.6	0.6	0.5	0.0	1.0	-3.465218763370346
179	0.2999999999999993	3	4	0.6	0.6	0.23032475948636633	0.1268204472347371	1.0	-4.069543576106077

50	0.29999999999999993	10	1	0.9650122610070425	0.6	0.11872676814184584	0.9297968524554201	1.0	-3.0844407845480006
50	0.29999999999999993	3	1	1.0	0.6	0.0	1.0	1.0	-4.184700178960916






Top 62 Most Important Features for XGBoost
forward_arbitrage_ema_10
usd_zar_rate_ema_60
btc_usd_ask_ema_60
day_of_week
forward_arbitrage_bb_width_120
btc_zar_bid_ema_60
usd_zar_rate_range_120
btc_zar_bid_rsi_60
forward_arbitrage_rsi_120
btc_usd_ask_depth_sma_120
usd_zar_rate_momentum_120
btc_usd_ask_rsi_120
usd_zar_rate_volatility_30
btc_usd_ask_range_120
btc_usd_ask_momentum_120
btc_zar_bid_depth_sma_60
usd_zar_rate_rsi_120
btc_zar_bid_depth_volatility_120
hour
btc_zar_bid_range_60
btc_usd_ask_depth_volatility_120
btc_zar_bid_depth_volatility_60
usd_zar_rate_momentum_60
forward_arbitrage_momentum_120
forward_arbitrage_momentum_60
forward_arbitrage_range_30
relative_strength_120
btc_usd_ask_depth_volatility_30
btc_usd_ask_momentum_60
volatility_ratio_60
forward_arbitrage_volatility_60
btc_usd_ask_depth_volatility_10
btc_usd_ask_depth_sma_30
btc_zar_bid_range_10
usd_zar_rate_range_10
btc_zar_bid_momentum_30
forward_arbitrage_range_10
volatility_ratio_30
btc_zar_bid_depth_volatility_30
btc_usd_ask_rsi_10
btc_zar_bid_rsi_10
btc_usd_ask_depth_sma_10
btc_usd_ask_bb_width_30
relative_strength_30
volatility_ratio_10
usd_zar_rate_momentum_30
forward_arbitrage_momentum_10
btc_zar_bid_depth_momentum_60
forward_arbitrage_momentum_30
btc_zar_bid_depth_volatility_10
btc_zar_bid_depth_momentum_120
usd_zar_rate_momentum_10
btc_usd_ask_momentum_10
btc_zar_ask_depth
btc_zar_bid_depth_momentum_30
btc_usd_ask_depth_momentum_30
forward_arbitrage_rsi_10
btc_usd_ask_depth_momentum_10
btc_usd_ask_depth_momentum_60
btc_zar_bid_depth_momentum_10
btc_usd_ask_depth_momentum_120
btc_usd_ask_depth
Feature
0
25
50
75
100
125
150
175
200
Gain Importance Score