Module 2

Bitcoin Price Prediction Analysis

Mojo Dojo Casa House

Table of Contents

Model Description and Justification

Goals and Objectives

Predict bitcoin price and provide valuable insights for investors, traders, and policymakers

Goal

Objective

Evaluate and select the best model based on performance metrics (MAE, RMSE) to predict next 10 days' close prices

2024-01-20

Prediction

2022-01-01

2023-10-31

2023-11-01

2024-01-30

Training Data

Testing Data

Comprehensive Model Description

Linear Regression

Lasso Regression
Ridge Regression

Time Series

Long Short-Term Memory (LSTM)
Autoregressive Integrated Moving Average (ARIMA)

Non-linear Regression

AdaBoost
LightGBM
XGBoost
Decision Tree Regressor (CART)
Random Forest

Rationale for Model Selection

Linear Regression

✅Simplicity and Interpretability
❎Assumption of Linearity
❎Prone to Outliers

Time Series

✅Temporal Dynamics
✅Forecasting Capability
✅Model Flexibility
❎Assumption of Stationarity

Non-linear Regression

✅Capturing complex pattern (nonlinearity and dependencies)
✅Dataset is non-stationarity, nonlinear model can adapt to these changing patterns
✅Robustness to Outliers
❎Overfitting

Tuning Parameters (Hyperparameters)

Perform Grid search

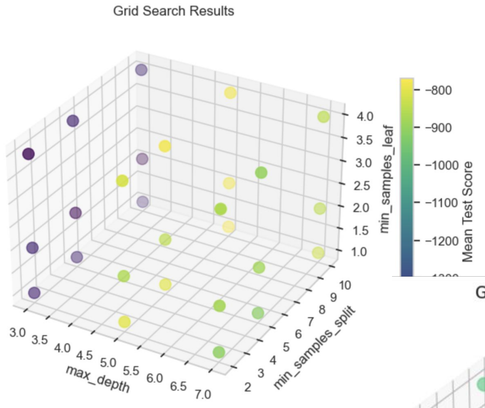
Find the optimal combination of parameters which has the **lowest MAE**

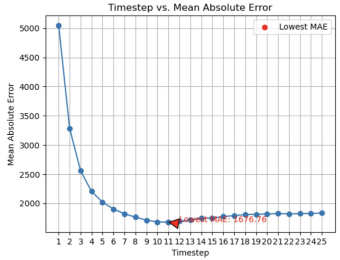
```
# Define the parameter grid
param_grid = {
    'min_child_weight': [1, 5, 10],
    'gamma': [0.5, 1, 1.5, 2, 5],
    'max_depth': [3, 5, 7, 12],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.3, 0.8]
}
# Initialize the XGBoost regressor
mymodel_xgb = XGBRegressor()

# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5, scoring='neg_mean_abso
grid_search.fit(X_train_xgboost, y_train_xgboost)

# Get the results of the grid search
results = grid_search.cv_results_
params = results['params']
mean_scores = -results['mean_test_score']

# Extract the parameter values and scores
n_estimators = [param['n_estimators'] for param in params]
learning_rate = [param['learning_rate'] for param in params]
max_depth = [param['max_depth'] for param in params]
# Print the best parameter
best_params = grid_search.best_params_
print(grid_search)
print("Best Parameter:", best_params)
```

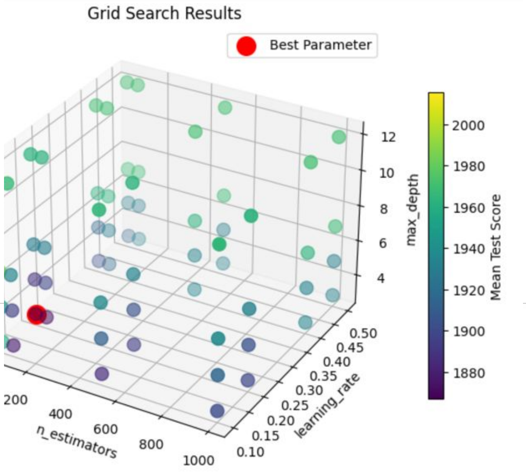
Tuning Parameters (Hyperparameters)

Tuning Plot Examples

```
Best Parameter: {'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 10}
```







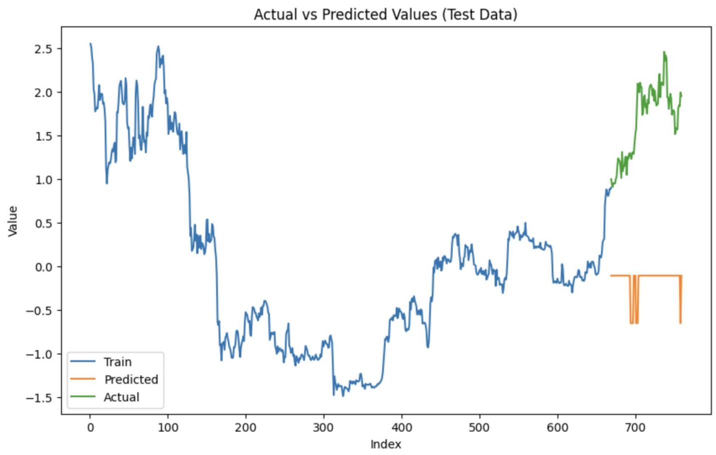

```
Performing stepwise search to minimize aic
 ARIMA(2,2,2)(0,0,0)[0] intercept   : AIC=12487.381, Time=0.07 sec
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=12782.275, Time=0.01 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=12645.669, Time=0.03 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=12548.096, Time=0.03 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=12780.278, Time=0.01 sec
 ARIMA(1,2,2)(0,0,0)[0] intercept   : AIC=12506.125, Time=0.04 sec
 ARIMA(2,2,1)(0,0,0)[0] intercept   : AIC=12530.056, Time=0.04 sec
 ARIMA(3,2,2)(0,0,0)[0] intercept   : AIC=12485.584, Time=0.05 sec
 ARIMA(3,2,1)(0,0,0)[0] intercept   : AIC=12536.734, Time=0.05 sec
 ARIMA(4,2,2)(0,0,0)[0] intercept   : AIC=12479.034, Time=0.12 sec
 ARIMA(4,2,1)(0,0,0)[0] intercept   : AIC=12524.601, Time=0.06 sec
 ARIMA(5,2,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.09 sec
 ARIMA(4,2,3)(0,0,0)[0] intercept   : AIC=12529.102, Time=0.09 sec
 ARIMA(3,2,3)(0,0,0)[0] intercept   : AIC=12492.950, Time=0.08 sec
 ARIMA(5,2,1)(0,0,0)[0] intercept   : AIC=12522.248, Time=0.15 sec
 ARIMA(5,2,3)(0,0,0)[0] intercept   : AIC=12524.392, Time=0.11 sec
 ARIMA(4,2,2)(0,0,0)[0]             : AIC=12464.300, Time=0.06 sec
 ARIMA(3,2,2)(0,0,0)[0]             : AIC=12481.333, Time=0.04 sec
 ARIMA(4,2,1)(0,0,0)[0]             : AIC=12522.173, Time=0.03 sec
 ARIMA(5,2,2)(0,0,0)[0]             : AIC=inf, Time=0.09 sec
 ARIMA(4,2,3)(0,0,0)[0]             : AIC=12527.138, Time=0.13 sec
 ARIMA(3,2,1)(0,0,0)[0]             : AIC=12534.160, Time=0.04 sec
 ARIMA(3,2,3)(0,0,0)[0]             : AIC=12491.040, Time=0.04 sec
 ARIMA(5,2,1)(0,0,0)[0]             : AIC=12519.444, Time=0.04 sec
 ARIMA(5,2,3)(0,0,0)[0]             : AIC=12522.429, Time=0.05 sec

Best model:  ARIMA(4,2,2)(0,0,0)[0]
Total fit time: 1.590 seconds
```

```
           param_grid={'colsample_bytree': [0.6, 0.8, 1.0],
                       'gamma': [0.5, 1, 1.5, 2, 5],
                       'learning_rate': [0.01, 0.3, 0.8],
                       'max_depth': [3, 5, 7, 12],
                       'min_child_weight': [1, 5, 10],
                       'n_estimators': [50, 100, 200],
                       'subsample': [0.6, 0.8, 1.0]},
           scoring='neg_mean_absolute_error')
Best Parameter: {'colsample_bytree': 0.8, 'gamma': 0.5, 'learning_rate': 0.3, 'max_depth': 3, 'min_child_weight': 1, 'n_estimator
s': 50, 'subsample': 1.0}
```

Consideration of Alternatives

Alternative Model

Modeling Using Keyword



Model	MAE	RMSE
Xgboost	13600.20	14434.23
CART	14530.80	15077.12
Lasso Regression	12161.39	12544.01
Ridge Regression	12544.55	12595.07
Random Forest	12202.54	12595.07

Model Validation, Performance, and Limits

Validation Techniques – New Approach: Using Timestep

Dataset for close price: x1,x2,x3,x4,x5,x6, x7,x8,....x760 (total 760 data points)

Training: x1, x2,x3, x4, x5, **x6**,**x7**,........x664,x665,x666,x667,x668, **x669** (total 669 data points for training)

Test: x670,x671,x672….x760(applied the same transformation as training set)

Example of **Training Matrix**:

$$X = \begin{pmatrix} \boxed{x1,x2,x3,x4,x5,} \\ \boxed{X2,x3,x4,x5,x6,} \\ \ldots\ldots \\ \boxed{x664,x665,x666,x667,x668} \end{pmatrix} \qquad Y = \begin{bmatrix} \mathbf{x6}, \\ \mathbf{x7}, \\ \cdots \\ \mathbf{x669} \end{bmatrix}$$

Dimension: 664 X 5

Dimension: 664 X 1

Validation Techniques – New Approach: Using Timestep

x1, x2, x3, x4, x5, p1, p2, p3, p4, p5, p6, p7, p8 …. p10, p11,p12,p13,p14, p15

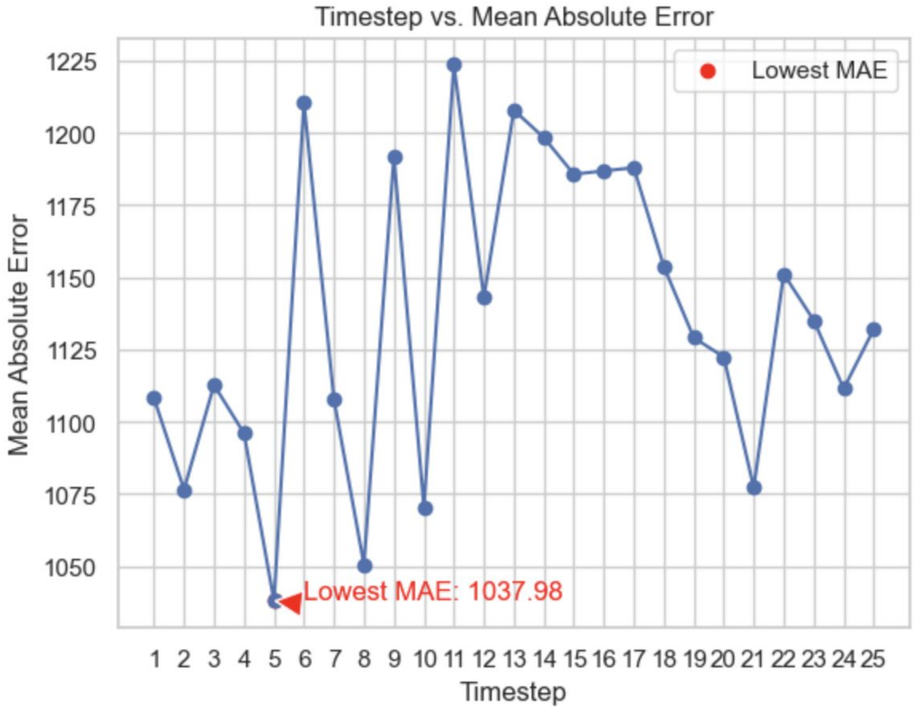
Note:

p: predicted

p1, p2, p3, p4, p5 are predicted from x1-x5

p6, p7, p8 …. p15 are predictions for 10 days

Validation Techniques – New Approach: Using Timestep



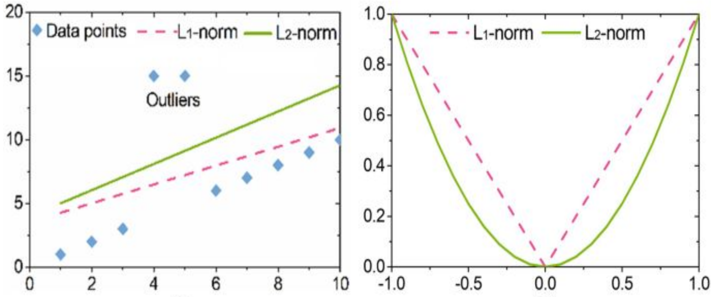
- Timestep Validation
- Validation to the model itself

Performance Metrics

R^2:
– Assume stationarity
– Designed for linear regression (not fit for our dataset)

MAE:
– Suitable for Bitcoin price prediction due to cryptocurrency market volatility
– MAE (L1) trends to produce a sparse solution many weights that are not important are driven towards 0

RMSE:
– More sensitive to outliers, like bitcoin price
– MSE (L2) limits the gigsk amount of weight evenly, so less important factors can continue to have less influence but still some influence



(a) (b)

Minimize MAE

Why choose MAE instead of R^2 or MSE?

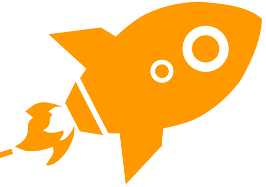
Performance Metrics

Model	MAE (test)	MAE (prediction)	RMSE (test)	RMSE(prediction)
AdaBoost	1113.39	1953.24	1420.74	2490.17
LightGBM	996.65	1252.23	1278.11	1676.29
XGBoost	1066.80	1297.40	1360.77	1894.29
CART	1681.11	1234.49	2397.72	1880.64
Lasso Regression	775.70	12578.83	1072.85	14198.81
Ridge Regression	766.89	12543.25	1065.08	14151.40
Random Forest	1066.38	12890.56	1352.11	14476.68

Model	MAE (prediction)	RMSE(prediction)
ARIMA	1496.24	1864.45
LSTM	1303.13	1579.77

Since **LSTM, XGBoost, and Lightgbm** has smaller MAE on test dataset, we will evaluate their prediction result

Interpretation of Results – LSTM

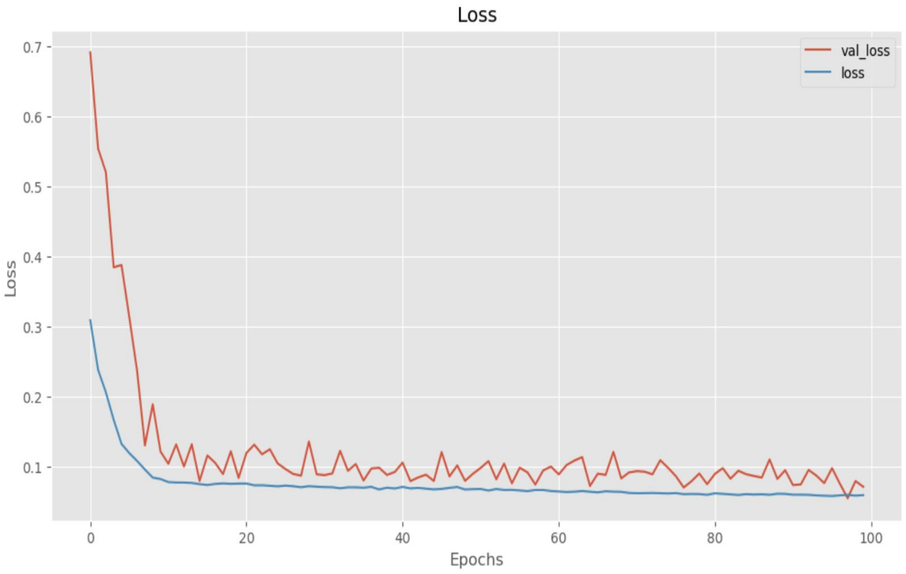


Why LSTM？

- Long-Term Dependencies (remember information for long periods) and Sequential Data Handling
- Market Volatility and Adaptability (learning from the most recent data)

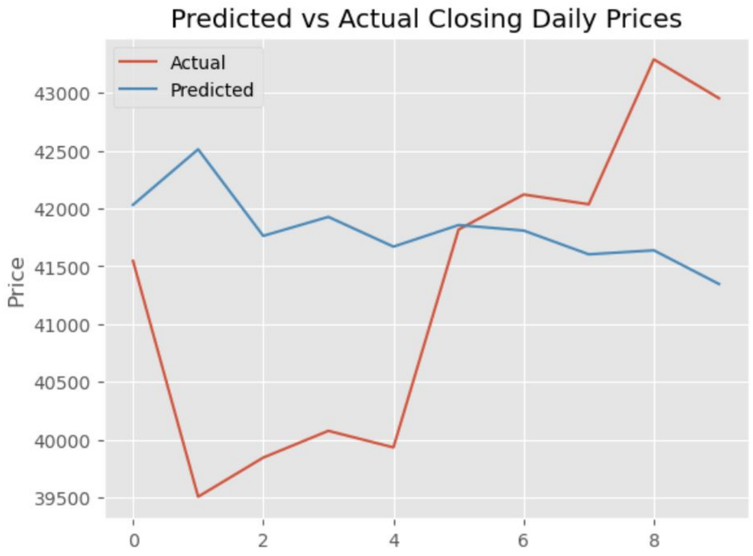
Interpretation of Results – LSTM

Result of Hyperparameter Tuning



Training for epoch=100 (when MAE stops reducing)

Prediction on next 10 days' close prices



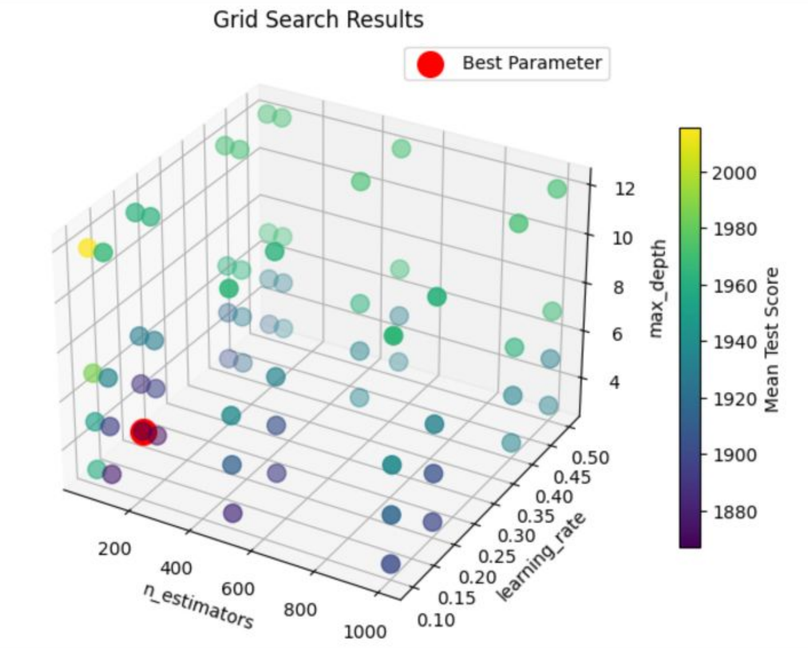
Interpretation of Results – XGBoost

Why XGBoost？

- Capture complex pattern (nonlinearity and dependencies)
- Dataset is non-stationarity, nonlinear model can adapt to these changing behaviors

Interpretation of Results – XGBoost

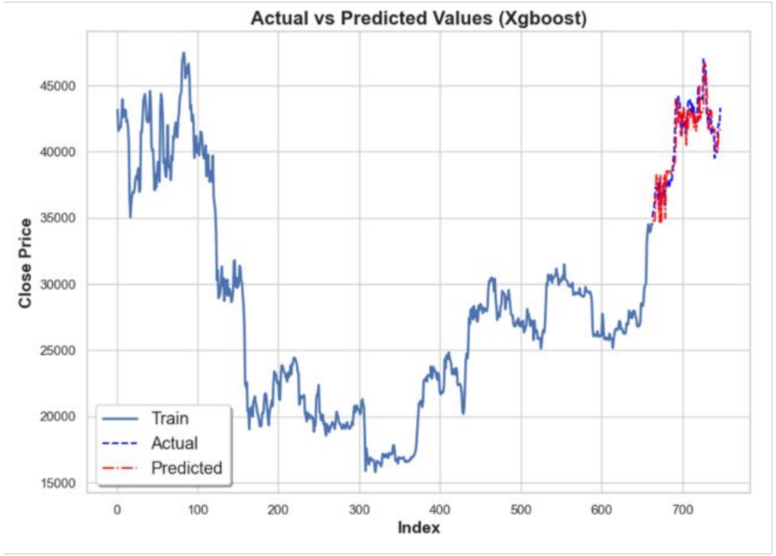
Hyperparameter Tuning Results



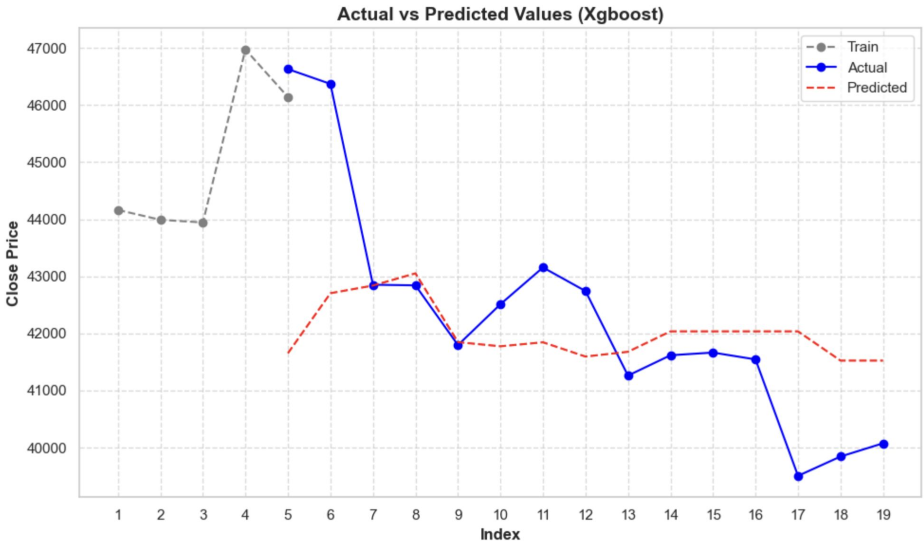

```
param_grid={'colsample_bytree': [0.6, 0.8, 1.0],
            'gamma': [0.5, 1, 1.5, 2, 5],
            'learning_rate': [0.01, 0.3, 0.8],
            'max_depth': [3, 5, 7, 12],
            'min_child_weight': [1, 5, 10],
            'n_estimators': [50, 100, 200],
            'subsample': [0.6, 0.8, 1.0]},
scoring='neg_mean_absolute_error')
Best Parameter: {'colsample_bytree': 0.8, 'gamma': 0.5, 'learning_rate': 0.3, 'max_depth': 3, 'min_child_weight': 1, 'n_estimator
s': 50, 'subsample': 1.0}
```

Interpretation of Results – XGBoost

Prediction on test dataset



Prediction on next 10 days' close prices



Interpretation of Results – XGBoost

One of the model limitation

Training XGBoost with large datasets may require significant memory and computational resources

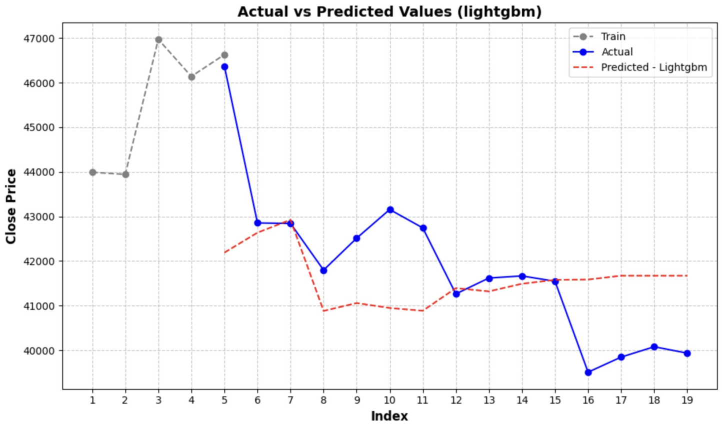
LightGBM: address some limitations of XGBoost

Pros:

- Improved training speed and reduced memory usage
- Achieve similar or even better performance

Cons:

- Sensitivity to hyperparameters -> making it harder to train



Interpretation of Results – Conclusion

Model Conclusion

- ❖ Linear regression models perform bad on MAE and RMSE
- ❖ Based on factors such as MAE, RMSE, computation cost, model stability -> want to choose xgboost as our final model for prediction

Model	MAE	RMSE	Computation Cost and Memory Usage	Model Stability
LSTM	1303.13	1579.77	High	Robust
XGBoost	1066.80	1297.40	Low	Robust
LightGBM	996.65	1252.23	Lower	Not robust

Limitations and Challenges

Discussion of Limitations and Challenges

Model Limitation: XGBoost

Potential Overfitting

Require significant computational resources

Approach Limitation

Focus on short term prediction

Only using tabular data for prediction

Biggest Challenge

Hard to combine keyword and timestep for prediction.

Addressing Challenges – Solutions and Approaches

Use Vector Autoregressive Model (VAR)

Add more features (including economic indicators i.e. VIX and interest rates)

NLP & TF-IDF to monitor market sentiment

Advanced feature engineering:
(e.g. using Quantile transformer & log transformer)

Application, Relevance, and Ethical Considerations

Real-World Relevance

Volatility Challenge

Bitcoin is known to be a highly volatile instrument and detecting rapid changes in price would mitigate risk.

Leveraged Trading Challenge

Detecting changes in bitcoin prices can prevent the occurrence of a margin closeout that forcefully clears out an investor's position.

Impact and Applicability

Enhance trading strategies, portfolio management, and risk mitigation techniques in the cryptocurrency market

Financial Markets

Collateralized Loans and Lending Protocols

Application:
provide insights for trading firms

– Predictive models facilitate quantitative analysis and backtesting of trading strategies
– Trading firms can simulate trading scenarios using historical data and price forecasts.
– Firms can optimize their trading algorithms based on the results of backtesting.

Accurate Bitcoin price predictions are essential for determining loan-to-value ratios, managing collateral requirements, and assessing the risk of liquidation events

Ethical Considerations

Data Privacy

All data used for our project is sourced legally from open-sources.

Transparency of model

Acknowledging the limitations of the model as well as detailing our approach and open source code.

Regulatory Compliance

Researched relevant SEC laws and ensured that our model is compliant. Traders can legally use our model for predictions.

Thank you