Developing an Algorithmic Trading Strategy for Bitcoin

Submitted by:
Manish Kumar
1009645840

Introduction

- Algorithmic Trading:
  - Algorithmic trading involves the use of algorithms to automate and execute trading strategies.
  - This project focuses on the development of a robust algorithmic trading strategy tailored specifically for the cryptocurrency market, with Bitcoin as the primary asset.

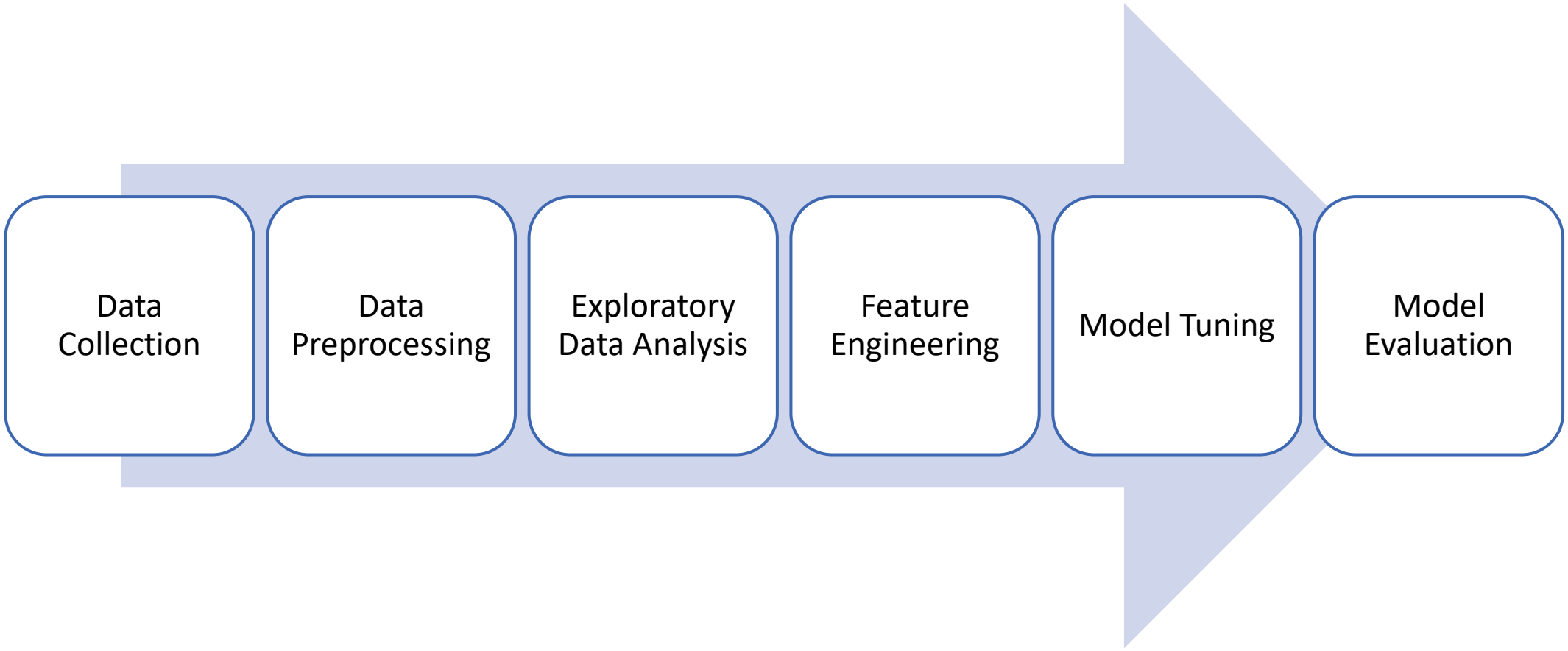
- Project Goals:
  - Design and Test Trading Strategies: Utilize historical data to create and refine trading strategies based on technical indicators.
  - Performance Evaluation: Backtest these strategies to assess their effectiveness, aiming to maximize returns and minimize risk.

Introduction

- Approach:
  - Data-driven insights: Leverage statistical analysis and data visualization to understand market behaviors.
  - Iterative Optimization: Continuously refine strategies based on backtesting results.

- Expected Outcomes:
  - A well-defined trading strategy that can outperform the market baseline.
  - Insightful data that could pave the way for applying machine learning techniques to improve strategy predictions.

Workflow Overview



Introduction to Data Collection

- Objective: Collect comprehensive data to fuel algorithmic trading strategies.

- Sources: Two primary sources:
  - Bitcoin Info Charts for crypto-specialized indicators.
  - Google News for sentiment analysis based on news headlines.

- Data:
  - Bitcoin OHLCV data.
  - Specialized crypto indicators for Bitcoin.
  - Historical news headlines related to Bitcoin.

Historical Bitcoin Indicators Data Collection

- Methodology: Use of requests and BeautifulSoup to scrape varied Bitcoin-related metrics.

- Data Points:
  - Number of transactions in blockchain per day
  - Average block size
  - Number of unique (from) addresses per day
  - Average mining difficulty per day
  - Average hash rate (hash/s) per day
  - Average price per day (USD)
  - Sent coins in USD per day
  - Average transaction fee (USD)
  - Median transaction fee (USD)
  - Average block time (minutes) (USD)
  - Average Transaction Value (USD)
  - Median Transaction Value (USD)
  - Number of unique (from or to) addresses per day.

Historical Bitcoin Data Collection

- Process:
  - Automated web scraping from multiple URLs.
  - Extraction of JavaScript-rendered data using regex parsing.

- Challenges:
  - Manual inspecting HTML files to extract details.
  - Handling large datasets.
  - Ensuring data integrity and accuracy.

- Code Implementation:
  - Implementation of the web scraping script is given in Python script named, webscrapper_bitcoinfocharts.ipynb
  - Bitcoin indicators data is stored in bitcoin_data_bitcoininfocharts.csv for easy access and manipulation.

Sentiment Data Collection from News Headlines

- Methodology:
  - Utilizing snscrape and guidelines from newscatcher to gather news headlines from Google News.

- Data Points:
  - Collect headlines related to Bitcoin for specified time frames (time frame is matched with Bitcoin OHLCV dataset, i.e. past 10 years).

- Process:
  - Scraping financial news headlines using dynamically generated queries by dates.
  - Aggregation of news headlines into a single dataset.
  - Links to full news articles are also collected but are not used for analysis.

Sentiment Data Collection from News Headlines

- Challenges:
  - Manual inspecting HTML files to extract details.
  - Dealing with non-standardized data.
  - Time efficiency in processing large volumes of news data.
  - Ensuring data integrity and accuracy.

- Code Implementation:
  - Implementation of the web scraping script is given in Python script named, webscrapper_googlenews.ipynb
  - News headlines data is stored in bitcoin_news_data.csv for easy access and manipulation.

Data Preprocessing

- Objective:
  - To clean and prepare raw data for exploratory data analysis, feature engineering, and modeling.

- Techniques Applied:
  - Data cleaning through inspection
  - Removing duplicates
  - Handling missing data using Imputation (mean, median, mode, constant)
    - *Imputation parameter i.e. mean, median, mode, or constant is chosen based on data inspection.*
  - Data deletion if required.

- Tools:
  - Python's Pandas and Scikit-learn libraries are used for data manipulation and preprocessing along with Microsoft Excel.

Introduction to Exploratory Data Analysis (EDA)

- Purpose:
  - To uncover patterns, spot anomalies, and check assumptions.

- Python Libraries Used:
  - Pandas and Numpy for data manipulation
  - Statsmodels for time series analysis
  - Matplotlib/Seaborn for visualization).

- Datasets:
  - Historical Bitcoin data
  - Bitcoin Indicators data
  - Google News sentiment data.

EDA – Bitcoin Time Series Analysis

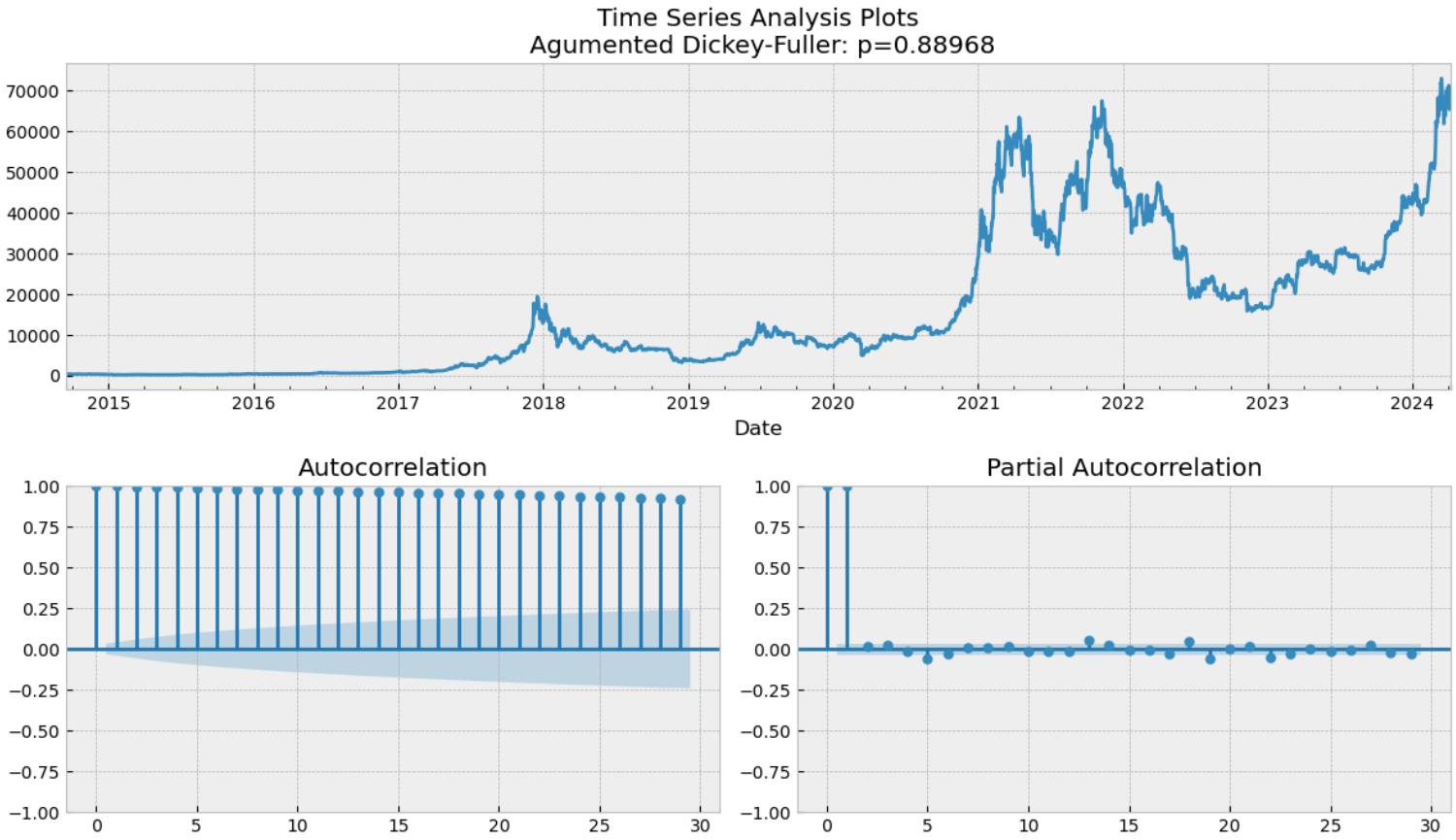
- Purpose:
  - To extract meaningful statistics from data indexed over time intervals.
  - To inspect autocorrelation and partial autocorrelation factors in the time series data.

- Techniques Used:
  - Autocorrelation and Partial Autocorrelation plots.

- Usage:
  - Partial Autocorrelation plot is used to get the number of window features required.
    - *Lookback window of 5 is chosen through initial inspection, (figure is given on next page).*

- Code Implementation:
  - Implementation of the web scraping script is given in Python script named, bitcoin_tsa.ipynb.

EDA - Time Series Analysis



EDA – News Sentiment Analysis

- Purpose:
  - Create a market sentiment signal based on daily financial news sentiment analysis.
  - To check the correlation between news sentiment and daily market returns.

- Libraries Used:
  - Vader-lexicon from NLTK library.
  - Bert-based transformer and Hugging Face's Transformers library.

EDA – Sentiment Analysis using Vader

- Data :
  - Data includes headlines along with their publication dates.
  - Data is stored in bitcoin_news_data.csv as described in the Step 1.

- Methodology:
  - Utilized the VADER SentimentIntensityAnalyzer to compute sentiment scores for each headline.
  - Scores aggregated by date to understand daily sentiment trends.

- Code Implementation:
  - Implementation of the sentiment classification script using Vader is given in Python script named, news_sentiment_analysis_vader.ipynb.
  - The results of this analysis are stored in bitcoin_news_data_sentiment_vader.csv for easy access and manipulation.

EDA – Sentiment Analysis using Vader

- Key Results:
  - The results obtained using the VADER sentiment analysis are not reliable, as it frequently misclassifies headlines that are relatively straightforward to interpret.
    - Conclusions are made based on manual inspection of the dataset stored in bitcoin_news_data_sentiment_vader.csv (see relevant row number 132456 for example)
  - Sentiment analysis reveals fluctuating public perception and potential predictors for market movements.
  - Headlines might not be enough to access the sentiment of the news for two reasons:
    - Headlines are misleading sometimes
    - Headlines are too short to make any reliable predictions.
  - Based on this analysis, a more sophisticated state-of-the-art Bert-based transformer is used for the task of Sentiment Classification.

EDA – Sentiment Analysis using Bert transformer

- Purpose:
  - To evaluate the sentiment of Bitcoin-related news headlines using the state-of-the-art BERT-based transformer specifically tuned for financial text.

- Data Overview:
  - Analyzed over 130,000 news headlines from a range of publications over a period spanning from 2014 to 2024.
  - Data includes headlines along with their publication dates.
  - Data is stored in bitcoin_news_data.csv as described in the Step 1.

- Methodology:
  - Used a BERT model pre-trained on financial texts, which is ideal for understanding complexities in financial news.
  - Used transformers library from Hugging Face for model deployment and sentiment classification.

EDA – Sentiment Analysis using Bert transformer

- Sentiment Classification:
  - Each headline was tokenized and fed into the BERT model to classify sentiments: positive, negative, and neutral.
  - The sentiment scores were aggregated to observe overall sentiment trends over a daily time period.

- Code Implementation:
  - Implementation of the sentiment classification script is given in Python script named, news_sentiment_analysis_bert.ipynb.
  - News sentiment data for Fin-Bert transformer is stored in bitcoin_news_data_sentiment_finbert.csv for easy access and manipulation.

EDA – Sentiment Analysis using Bert transformer

- Key Findings:
  - Unlike VADER, which assigns intensity scores to sentiments, the BERT-based model categorizes news as positive, neutral, or negative.
  - Sentiment classifications derived from the BERT model are found to be more consistent and reliable compared to those from VADER when analyzing the same set of news headlines.
    - Conclusions are made based on manual inspection of the classifications made by Vader and Bert transformer on the same news headlines.
  - Based on its enhanced reliability, the BERT model's classifications are chosen for subsequent in-depth analysis and research activities.

EDA – Correlation between market returns and news sentiment

- Purpose:
  - To analyze the relation between Bitcoin price movements and the sentiment derived from Bitcoin-related news headlines.

- Data Description:
  - Utilized a dataset of Bitcoin prices and matched news headlines spanning a period of 10 years.
  - Sentiment classifications for news headlines calculated using the Bert-based sentiment analysis model are used.

- Methodology:
  - Bitcoin price data and news sentiment classification were synchronized by date.
  - An intuitive simple strategy is devised for this task.

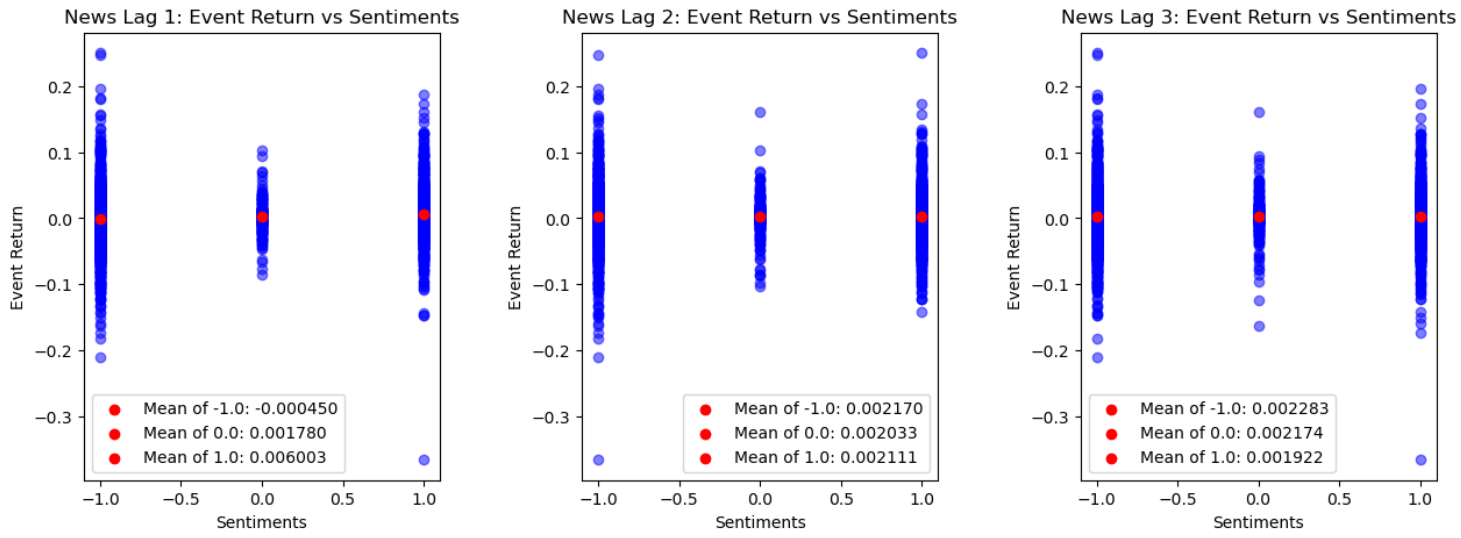
EDA – Correlation between market returns and news sentiment

- Code Implementation:
  - Implementation of the code is given in Python script named, news_btc_correlation.ipynb.
  - The results are stored in btc_usd_sentiment.csv for easy access and manipulation.

- Key Insights:
  - A small linear relation is found between the current-day return and 1-day lagged news sentiment, as can be seen in the graph on the next slide.
    - Detailed results and analysis is given in news_btc_correlation.ipynb.
  - This can be a useful feature for predictive analytics.
    - This sentiment feature is included in the dataset for Machine Learning and its significance will be discussed in Step 5.

EDA – Correlation between market returns and news sentiment

- As can be seen in the figure, there is a small relationship between the current-day return and 1-day lagged news sentiment.
- No conclusion is made on 2-day and 3-day lagged news sentiment.
- This is not the final conclusion, as the relationship will be further discussed in Step 5.



Introduction to Feature Engineering

- Objective:
  - To improve the predictive power of the machine learning models by creating new features from existing data.
  - Feature engineering can often be more beneficial than fine-tuning model parameters.

- Data Overview:
  - Daily BTC-USD dataset.
  - Daily Bitcoin indicators from Bitcoin Info Charts.
  - Daily news sentiment classification data from Sentiment Analysis.

- Methodology:
  - Bitcoin OHLCV data is used to derive various technical indicators.
  - The implementation details of these indicators are taken from Investopedia.
  - These indicators are chosen based on the guide to best NASDAQ indicators
    - I couldn't find any reliable sources for technical indicators based on OHLCV data for cryptocurrencies.

Introduction to Feature Engineering

- Code Implementation:
  - Implementation of the feature engineering code is given in Python script named, bitcoin_feature_engineering.ipynb.
  - The results are stored in bitcoin_techindicators_filtered_normalized.csv for easy access and manipulation.
- Final Set of Indicators:
  - 'sentiment',
  - 'btc_trans_blockchain',
  - 'avg_block_size',
  - 'unique_sentbyaddress',
  - 'avg_hashrate',
  - 'sent_in_usd',
  - 'median_trans_fee',
  - 'avg_block_confirm_time_min',
  - 'median_trans_value_usd',
  - 'sma_20',
  - 'ema_20',
  - 'ema_12',
  - 'ema_26',
  - 'macd',
  - 'signal_line',
  - 'rsi',
  - '%k',
  - '%d',
  - 'middle_band',
  - 'upper_band',
  - 'lower_band'

Feature Engineering Workflow

Data Loading

- Relevant CSV files are loaded as described in the presentation.
- The datasets have been appropriately concatenated to form a compre-hensive set for detailed analysis.

Data Cleaning

- Irrelevant indicators are removed based on correlation metrics available on Bitcoininfo.com. From groups of similar indicators, only one is retained (e.g., choosing between avg_trans_fee and median_trans_fee).
- Imputation techniques such as mean, median, mode, or a constant value are applied to fill missing data, tailored based on a thorough inspection of the dataset.

Technical Indicators

- A variety of technical indicators are calculated using OHLCV (Open, High, Low, Close, Volume) data, including but not limited to the Relative Strength Index (RSI), Simple Moving Average (SMA), and Exponential Moving Average (EMA).

Data Cleaning

- Missing values are addressed through imputation, utilizing methods such as mean, median, mode, or a specified constant, chosen based on detailed data inspection.
- Where necessary, data points with insufficient recoverable information are selectively deleted to maintain the integrity of the dataset.

Normalizing Data

- Volume-like and price-like indicators are normalized separately, using scaling criteria appropriate for each type as discussed in the course materials.
- The processed and normalized data is then saved into a CSV file, setting the foundation for developing algorithmic trading strategies.

Model Evaluation and Tuning Workflow

- Objective:
  - To adjust the model's hyperparameters to maximize predictive accuracy and ensure optimal performance across various market conditions.
  - Through systematic testing using cross-validation and reality checks, the aim is to affirm the model's efficacy in real-world scenarios.

- Code Implementation:
  - Implementation of the model evaluation and testing workflow is implemented in Python script named, main.ipynb.
  - The data used for model training and evaluations is stored in main.csv for easy access and manipulation.

Model Evaluation and Tuning Workflow

- Step 1: Data Loading and Preprocessing
  - The dataset bitcoin_techindicators_filtered_normalized.csv is loaded, containing Bitcoin technical indicators.
  - No additional preprocessing is required as the data comes pre-cleaned and normalized from the feature engineering phase.

- Step 2: Feature Engineering and Target Definition
  - Utilizing insights from the partial autocorrelation analysis, 5 window features are crafted to capture the momentum in opening prices, optimizing for the quantity of data available.
  - The **target**, labeled **target_returns**, represents the daily returns based on opening prices.

Model Evaluation and Tuning Workflow

- Step 3: Preparing Data for Modeling
  - The dataset is split into an 80% training set and a 20% test set
  - To preserve the chronological sequence vital for time-series analysis, the data is split sequentially without shuffling.

- Step 4: Evaluation Matric Customization
  - An information coefficient, specifically Spearman's rho, is employed as the custom scorer for both optimization and evaluation phases.
  - Custom scoring aligned with financial metrics is integral to the development of an effective algorithmic trading strategy.

Model Evaluation and Tuning Workflow

- Step 5: Defining Pipelines:
  - Separate preprocessing pipelines are established for numeric (imputation and scaling) and categorical (imputation and one-hot encoding) features to prepare the data for modeling.
  - A ColumnTransformer integrates these pipelines, applying appropriate transformations to each feature type and allowing for efficient data processing within a unified framework.

- Step 6: Parameter Grid:
  - Parameter grids are established for various regression models, including Ridge, Random Forest, Gradient Boosting, Extra Trees, AdaBoost, and MLP Regressor.
  - Each model's search parameters are meticulously defined to explore a wide range of hyperparameter combinations during optimization.
    - Ridge Tuning: Alpha
    - Random Forest: Number of estimators, max features, max tree depth, min samples split, min samples leaf, and bootstrap.
    - Gradient Boosting: Number of estimators, learning rate, tree depth, min samples split, min samples leaf, and sub-sampling rates.
    - Extra Trees: Number of estimators, max features, max tree depth, min samples split, min samples leaf, and bootstrap.
    - AdaBoost: Number of estimators, learning rate.
    - MLP Architecture: Hidden layers, alpha, learning rate, max iteration, initial learning rate, and momentum.

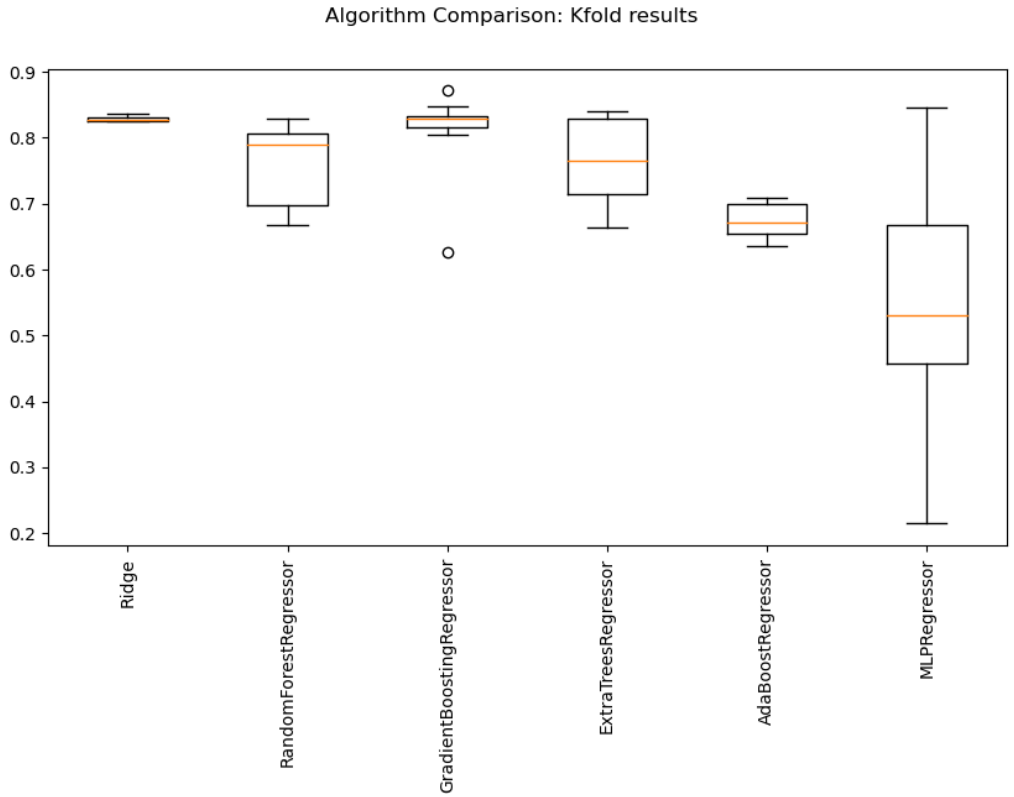
Model Evaluation and Tuning Workflow

- Step 7: Model Optimisation
  - Each model (defined in Step 6) undergoes a RandomizedSearchCV process, integrating preprocessing steps, to identify the best hyperparameters based on the Spearman scoring function.
  - The optimal parameters and the best cross-validation score for each model are captured and stored.

- Step 8: Performance Plots
  - A boxplot is utilized to compare the cross-validation results across different models, providing a visual representation of each model's performance consistency and variance. (*shown next slide*)
  - A bar chart contrasts the train and test errors for each model, aiding in the detection of underfitting or overfitting and ensuring the selection of a model with a good generalization performance. (*shown next slide*)

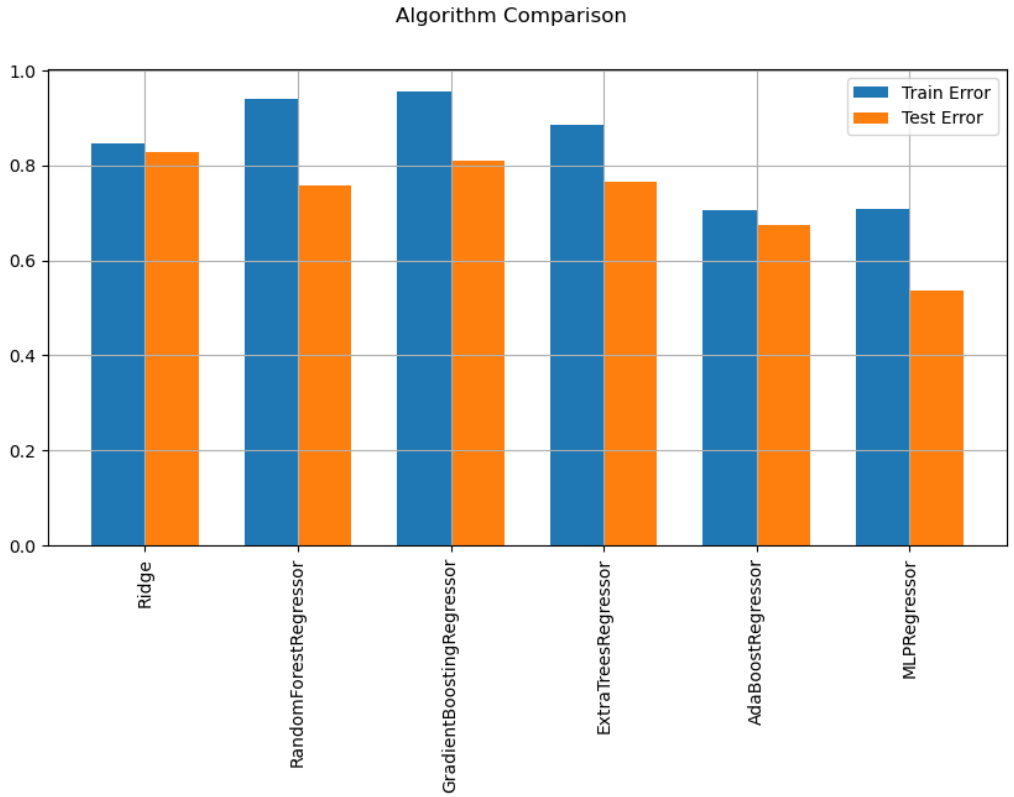
Model Evaluation and Tuning Workflow

- Ridge regression performs the best with stable results across 5 folds, exhibiting minimal variance in its performance.



Model Evaluation and Tuning Workflow

- The performance gap between training and testing sets for Ridge regression is minimal, indicating strong generalization capabilities and a well-fitted model.

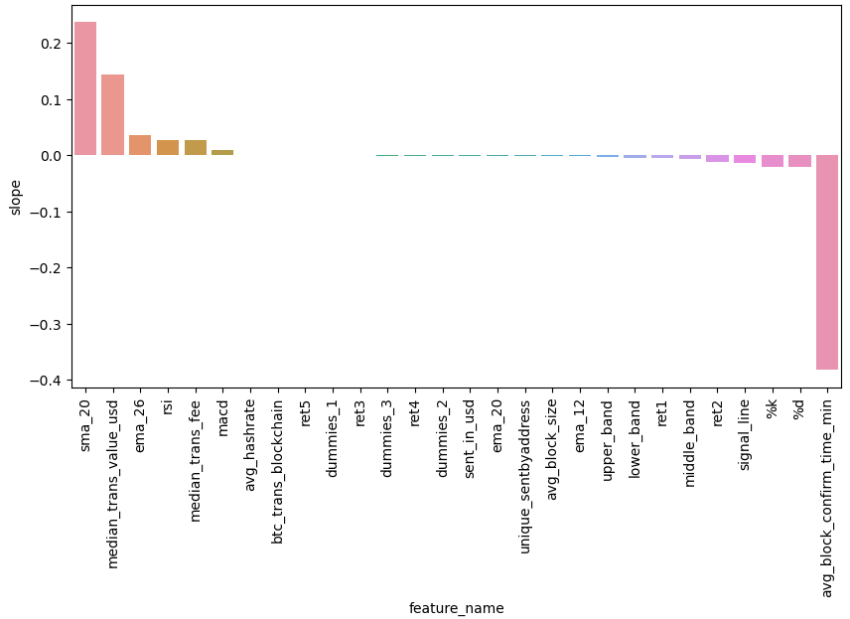


Model Evaluation and Tuning Workflow

- Step 9: Model Finalization:
  - Based on the above analysis **Ridge Regression** model is selected with **alpha=0.113**.
  - Cumulative returns for the trading algorithm are plotted against time.

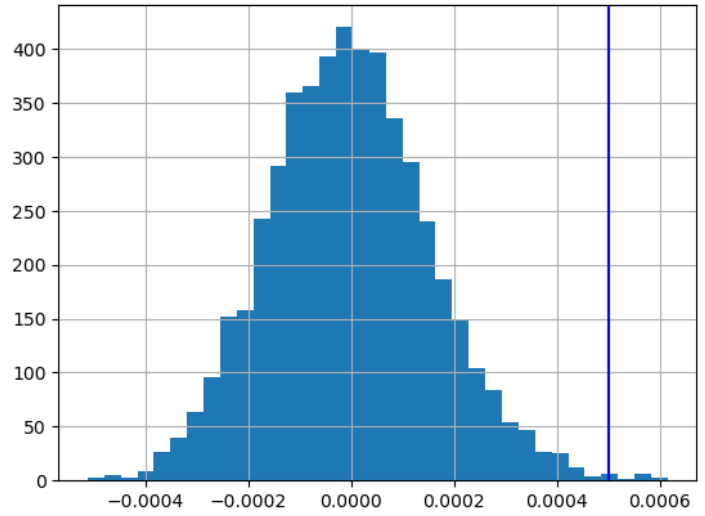
Model Evaluation and Tuning continued

- Step 10: Performance Evaluation and Feature Importance:
  - Out-sample CAGR and Sharpe Ratio are calculated for performance evaluation.
    - CAGR = 0.88
    - Sharpe ratio = 1.42
  - Feature Importance plot is shown below



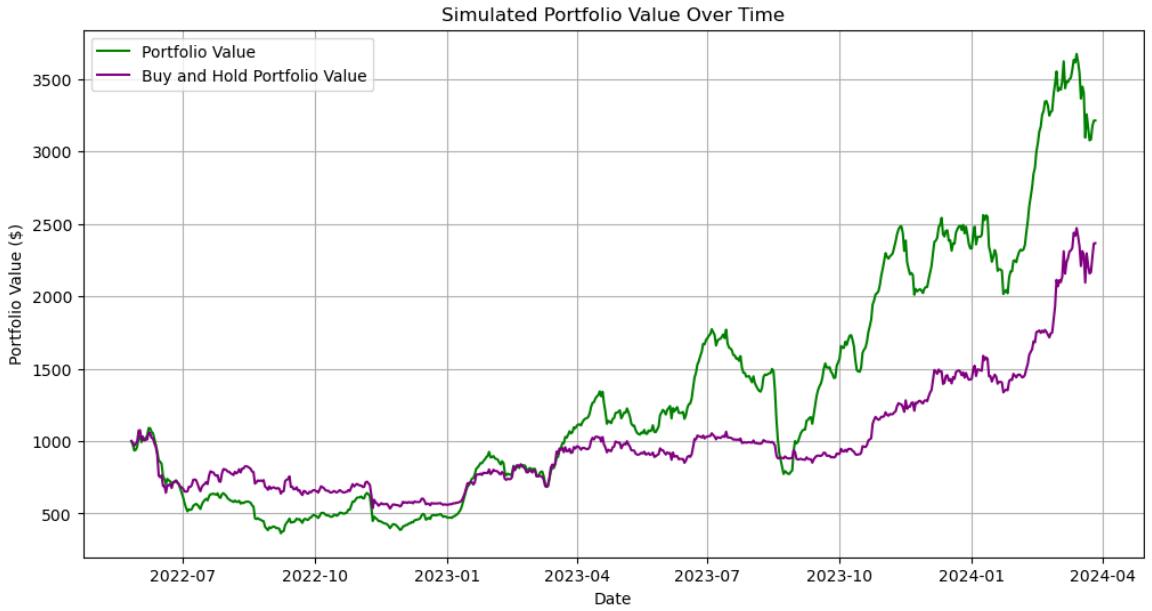
White Reality Check

- The open prices from the dataset are detrended to neutralize time-dependent effects, and the returns are tested with the White Reality Check to statistically validate the predictive power of the trading strategy beyond a random chance.
- The adjusted returns are subjected to the White Reality Check, employing a bootstrap method to rigorously assess the significance of the trading model's performance, ensuring robustness and reliability in the algorithm's predictive capabilities.



Comparison: Derived Trading Strategy with Buy-Hold Strategy

- Trading strategies are simulated with an initial portfolio value of 1000 over only test set (out-of-sample data).
- The results are shown below, concluding the better performance of our strategy compared to the simple buy and hold strategy.



Conclusions

- **Effective Model Identification**: Ridge regression was pinpointed as the optimal model for this dataset, demonstrating minimal performance variance across validation folds and robust generalization capabilities between training and testing datasets.
- **Strategic Hyperparameter Tuning**: Careful selection and tuning of hyperparameters for various models, including Random Forest and Gradient Boosting, highlighted the critical role of methodical optimization in enhancing model performance.
- **Comprehensive Performance Metrics**: The application of key financial performance metrics such as CAGR and Sharpe ratio provided a deep understanding of the model's risk-adjusted return capabilities.
- **Statistical Validation**: The use of detrending techniques and the White Reality Check for statistical tests affirmed the model's predictive strength and reliability, underscoring the importance of rigorous statistical validation in financial modeling.
- **Visual Insights and Risk Evaluation**: Plots illustrating cumulative returns and the comparative analysis of train versus test errors facilitated straightforward visual insights into model performance and helped identify the potential for overfitting or underfitting.

Future Research Directions

- **Enhanced Feature Engineering**: Explore additional technical indicators and alternative data sources (e.g., sentiment analysis from news articles from finviz) to enrich the model's predictive accuracy.
  - I have incorporated a web scraping script for finviz to gather news articles associated with various stocks, aiming for less noisy data. However, finviz's limitation of storing only the latest 100 news articles per stock makes it challenging to amass a substantial dataset.
- **Model Complexity**: Investigate more complex models or ensemble techniques that may capture nonlinear patterns and interactions more effectively than current models.
- **Deep Learning Approaches**: Consider implementing deep learning architectures such as LSTM (Long Short-Term Memory) networks to better handle sequence prediction problems inherent in financial time series.
- **High-Frequency Trading Data**: Utilize higher frequency data (minute-by-minute or second-by-second) to potentially uncover more granular insights and improve short-term trading strategies.
- **Deep Learning Endeavors**: I am currently working on implementing transformer-based methods for algorithmic trading, more specifically Temporal Fusion Transformers (TFT).

Thank You