# Problem Statement

Cryptocurrencies are one of the most emerging technological advancements in the financial sector of the past decade, and the leading cryptocurrency is Bitcoin. Cryptocurrencies, by nature, are extremely volatile, and Bitcoin is no different. The ability to predict the future price of bitcoin and forecast it is an extremely valuable asset, as it could guide investors on when to invest into said cryptocurrency. It also is beneficial to gather information about the world economy as a whole; since bitcoin is the biggest cryptocurrency, it shows and dictates patterns to other cryptocurrencies and other economical factors.

The aim of this project is to develop a data mining model utilizing the KDD process in RapidMiner Studio to preprocess, mine, and extract valuable information from its historical price data, and compare which model and data mining technique yields the most accurate results, ultimately leading to a model that is able to accurately predict and forecast the future price of Bitcoin.

This project addresses the challenges of predicting the future price of such a volatile concept through historical data in relation to it. Applying data preprocessing, model training and evaluation techniques in RapidMiner Studio to come up with the most accurate model to predict the Bitcoin price in the future.

By the end of the project, we expect to find the most accurate model throughout our testing, that can effectively predict future prices of Bitcoin, and gain more knowledge on what are the factors that affect its pricing overall.

# Selection of Data Set

The dataset that we’ve chosen is the Bitcoin dataset from Kaggle from the article “Cryptocurrency Historical Prices.” Which contains historical Bitcoin statistics collected over several years, namely from 2009 to 2017, which encompasses numerous values such as market performance and blockchain activity.

We have chosen this dataset for our project for a multitude of reasons, including but not limited to:

* It’s directly related to our topic in the financial technology sector
* It has more than sufficient data and is structured well enough to come up with predictive models and forecast bitcoin prices
* It has great depth for modeling as it covers many concepts such as blockchain and economic features

The dataset consists of 24 attributes, 23 of them being quantitative (float64) and 1 date attribute. There are over 1500 instances/records, more specifically, 1584 instances.

The dataset also has 478 missing values in the btc\_trade\_volume attribute, which is a highly important attribute for our analysis and modeling, so we would have to fill those in during preprocessing in Task 3.

The dataset also contains some redundant attributes, with some attributes having a correlation value of >0.99, alongside multiple transaction related attributes such as btc\_n\_transactions, btc\_n\_transactions\_excluding\_popular, and btc\_n\_transactions\_total, this will also be addressed in task 3 with either feature selection or applying PCA.

The following are some visualizations regarding the dataset to further understand the distribution and summary of the data.

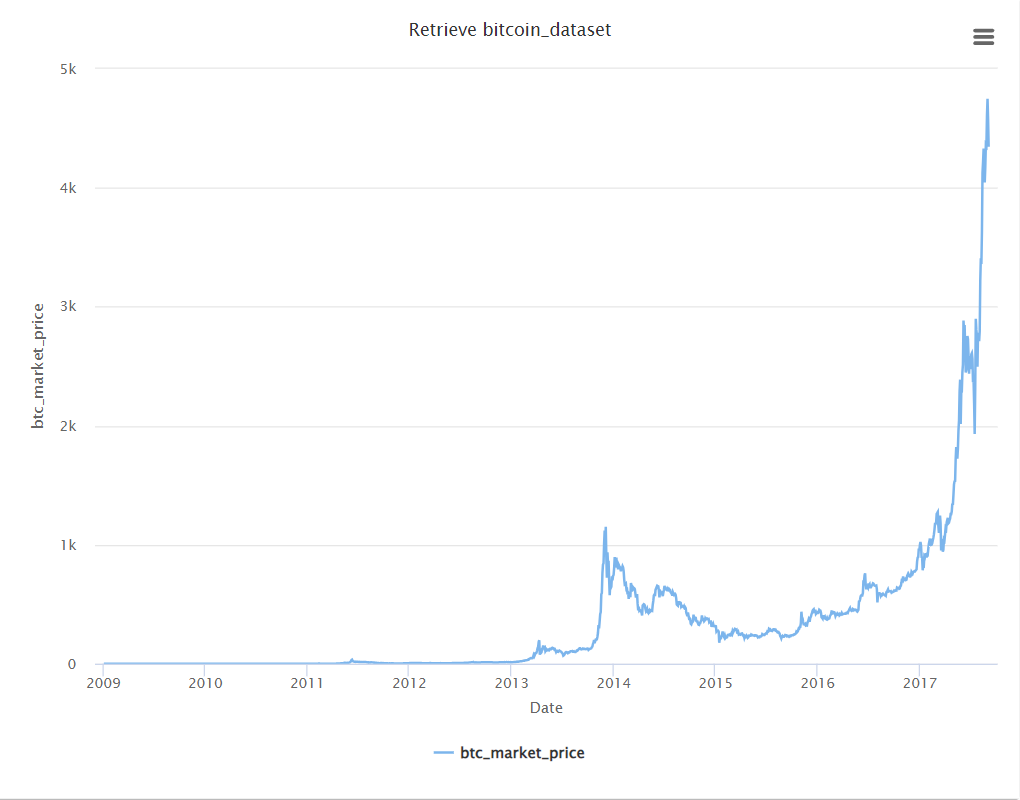


Figure 1 - The market price of bitcoin throughout the years

Figure 1 shows how volatile the price of bitcoin can be, and how it has had exponential growth in the later years.

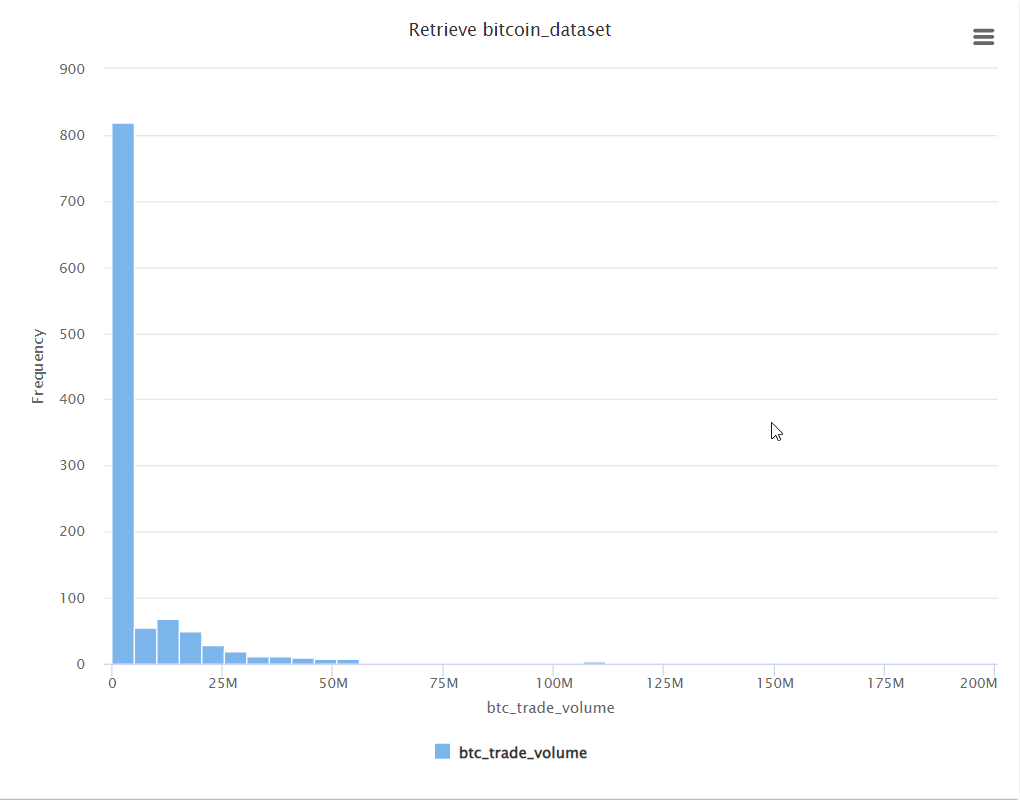


Figure 2 - A Histogram showcasing btc\_trade\_volume

As we can see from the histogram in figure 2, the btc\_trade\_volume attribute is extremely rightly skewed, and there are significant outliers in the data, showcasing the need for proper outlier treatment and normalizing the values.

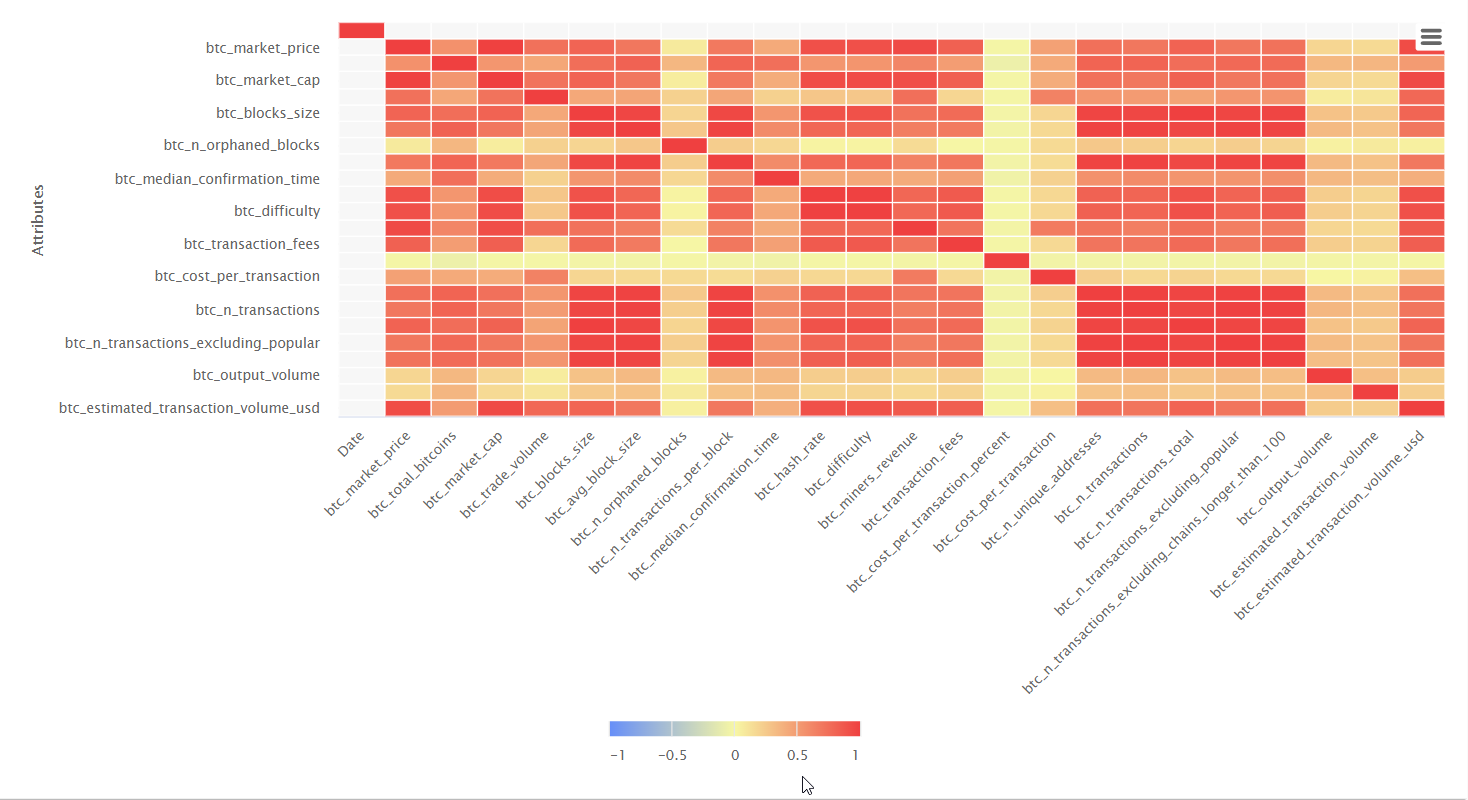


Figure 3 - Heatmap correlation Matrix

Figure 3 displays just how extremely highly correlated some attributes are, reinforcing the point we discussed earlier about redundant attributes, and the need for a dimenstionality reduction technique such as PCA.

A graph with blue lines

AI-generated content may be incorrect.

Figure 4 - Normalized Box plot of difficulty, hash rate, and market price

Figure 4 shows a normalized box plot of the 3 attributes to showcase the outliers and how the large scale attributes can severely skew the data.

This dataset is a time-series dataset, showcasing the evolution and volatility of the bitcoin cryptocurrency over time. Numerous attributes (such as market\_price and difficulty) display exponential growth in the later years, which goes hand in hand with the evolution in the financial technology sector. Within the attributes, there are large scale attributes, which are magnitudes bigger than others, portraying the need to normalize the data for better results.

# Preparation and pre-processing of Data

A diagram of data processing

AI-generated content may be incorrect.

Figure 5 - KDD process diagram

The Knowledge Discovery in Databases process comprises of numerous stages. It starts off with Data Selection, then goes to Data Cleaning and transformation and reduction, the step we are currently at.

Missing values, inconsistent values, redundant data and outliers are all challenges to be faced during the process of KDD, and all are obstacles that must be addressed to get proper outcomes and being able to extract desired data efficiently and effectively.

## Data Cleaning

The first part of it is data cleaning, the process of dealing with inaccurate, inconsistent, noisy, or incomplete data. The first step was to deal with the missing values, the dataset had significant gaps in the btc\_trade\_volume attribute, with seven significant gaps, ranging from 3 day gaps all the way to a gap that is 367 days. Based on this, a 2 step plan has been implemented to deal with the missing values, while keeping respect to the time-series nature of the data.

For the smaller gaps of less than 22 days, we applied a moving average using the moving average operator in RapidMiner, with a window width of 22 (for 22 rows, which equates to 22 days in our dataset)

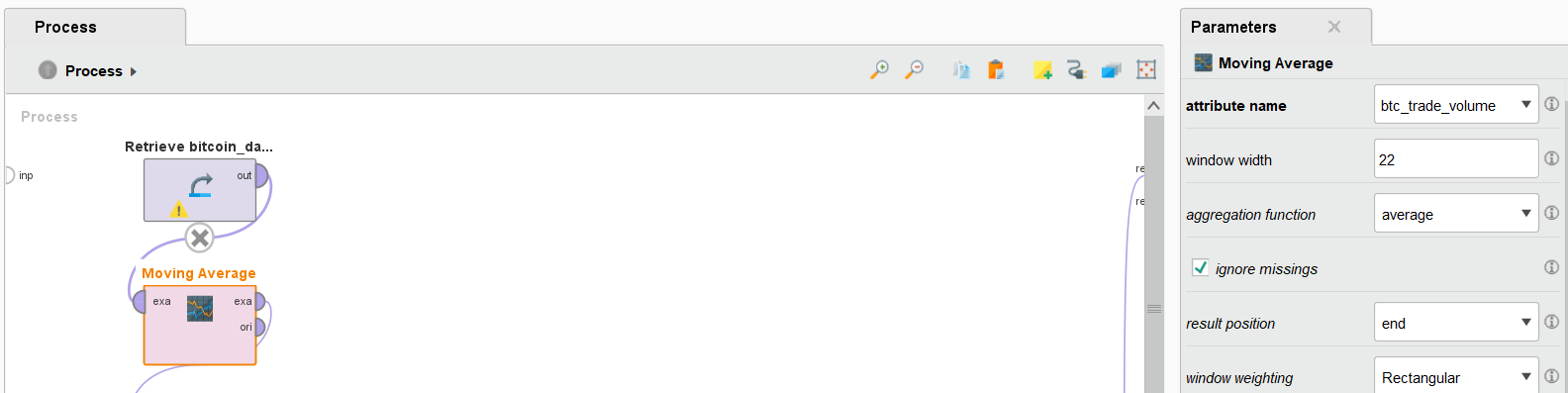


Figure 6 - Parameters of the moving average operator

The moving average operator takes in the values consisting of the current value and x amount of neighboring values (in our case 22) and takes in the average of it, providing a way to smoothen the gaps without sacrificing the integrity of the data.

Consecutively applied is the Generate Attributes operator, used to create a new column called btc\_trade\_volume\_RMV that is conditioned to fill in the missing values in the original attribute of gaps<22 days with the average, skipping over non-missing rows.

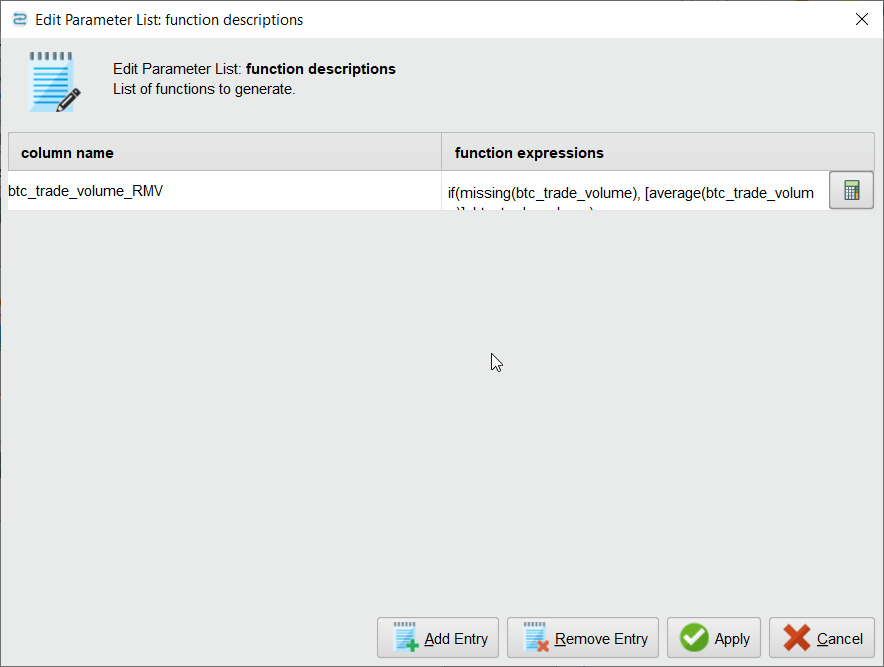


Figure 7 - using the if(missing(btc\_trade\_volume), [average(btc\_trade\_volume)], btc\_trade\_volume) expression in the generate attributes operator

Secondly, for our large gap of 367 rows (i.e 367 days), the forward fill method has been used as averages of a gap this big could not be easily calculated and would not be feasible.

Utilizing the Replace Missing Values (Series) operator set with “previous value” to achieve forward fill to logically fill the aforementioned gap in respect to the time series nature of the dataset.

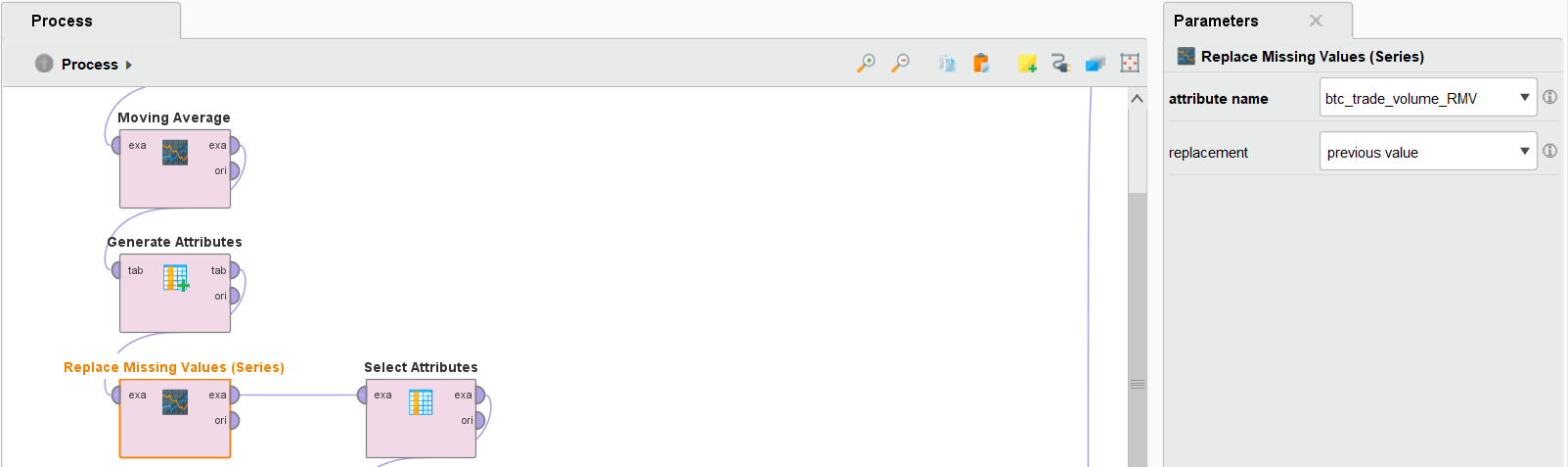


Figure 8 - The set up of the operator to forward fill the gap

The combination of the moving average and the forward fill methods allows us to fill all the missing values in the dataset ensures readiness for analysis and mining in the next steps, while maintaining the integrity, realism and completion of all attributes within the dataset.

## Selection and Normalization of Attributes

After ensuring that the dataset is now complete, we’ve selected the most relevant attributes to keep for the next steps of preprocessing and mining, as those attributes encompass the most needed and most relevant parts of the dataset. We’ve utilized the select attributes operator to achieve this.

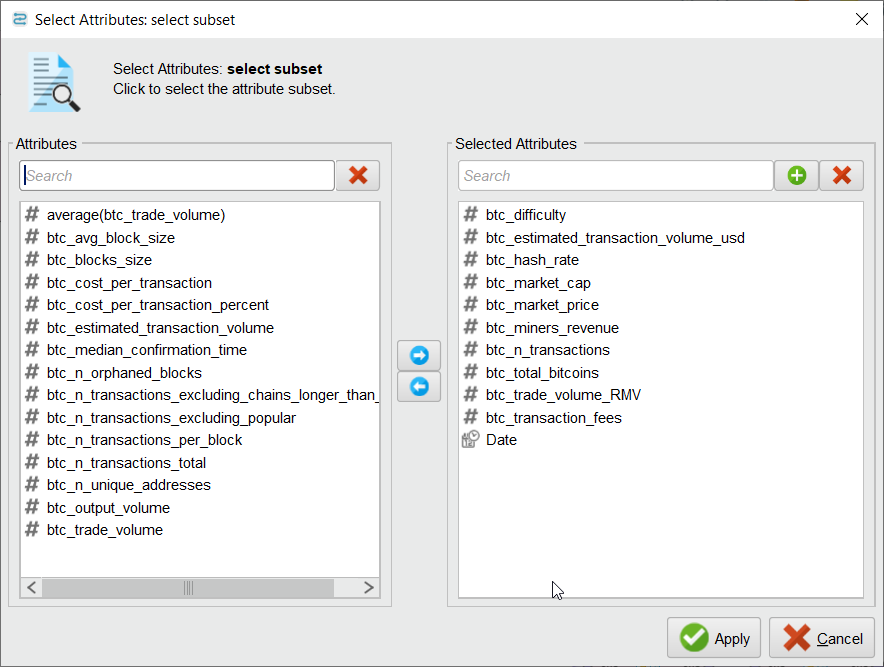


Figure 9 - Using the select attributes operator to only include a subset of attributes

Alongside that, we’ve also used the rename operator to change the btc\_trade\_volume\_RMV attribute to its original name, as that is the column with the complete data.

We’ve added the Generate ID attribute to the process to have a unique identifier and keep respect to the time series regardless of which models we use.

Next comes normalization, an extremely important step in data pre-processing to ensure all values are taken into account equally, eliminating outliers, preventing numerical instability, and massively improve performance for a multitude of algorithms such as nueral networks and distance based algorithms such as KNN.

We have implemented the Normalize operator using Z-Transformation (or statistical normalization) to the selected attributes, it bases each attribute around a mean of zero, achieving consistent feature scaling across the whole dataset.

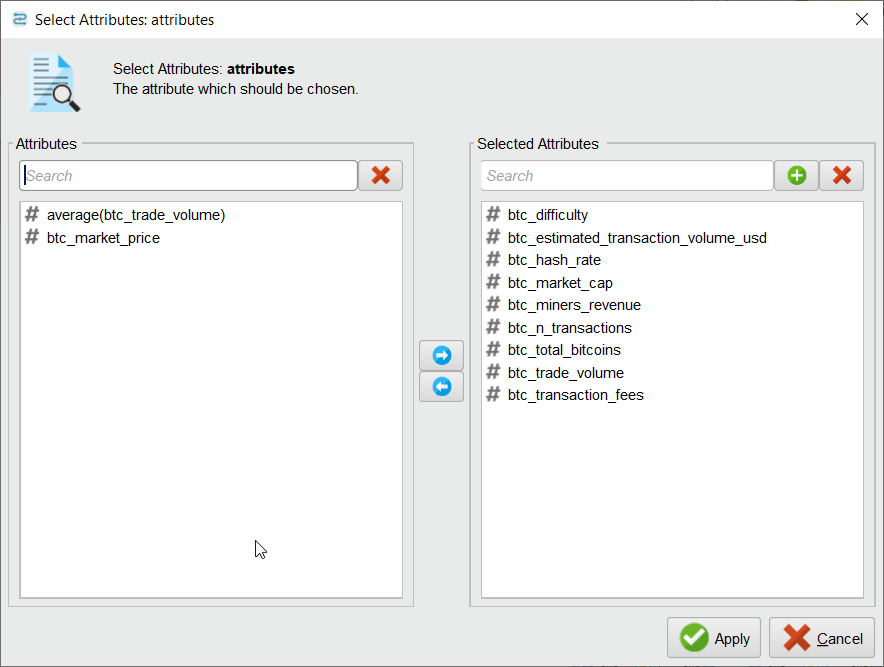


Figure 10 - The configuration of the normalize operator

The btc\_market\_price was intentionally left out as preserving its original value is highly important for the analysis and the ability to predict future values.

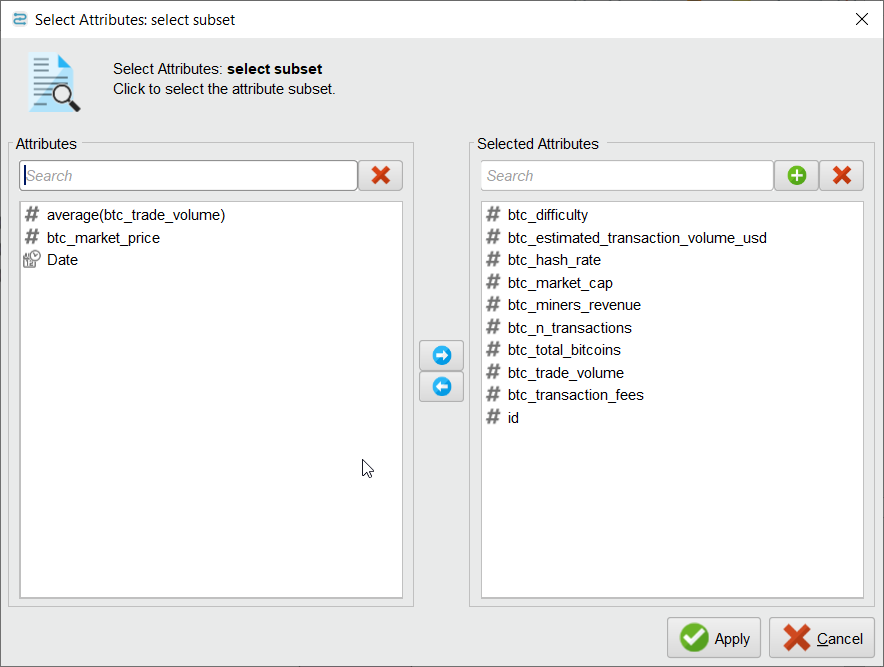
We have then added the multiply operator to duplicate our data, split into 2 different select attributes.   


Figure 11 - Selection of PCA Attributes

The first select attributes feeding into the PCA operator, leaving out the trade volume and market price attributes to keep them as is.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 12 - Flow of Process and configuration of PCA

We have made the variance threshold at 0.95 to retain accurate data, but not to go too aggressive into dimensionality reduction and lose too much of the data.

The second split of the multiply leads to another select attribute wherein we take the btc market price, date and ID, to then join them with the attributes that have went through PCA.

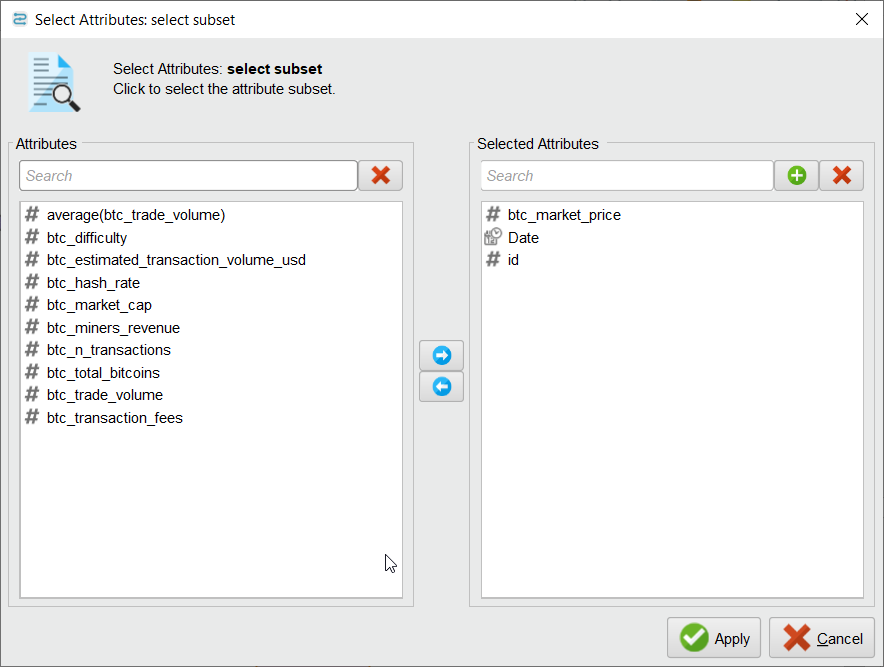


Figure 13 - Second select attributes to join the date, market price, and ID into the PCA'd dataset

## Visualization differences between raw dataset and preprocessed dataset

Task 4

Task 4 Began with the retrieval of the preprocessed dataset, the type of models used were mainly supervised, because they perform more efficiently with Time-Series data.

Linear regrettion-h2

The first model used was Linear Regression, a set role as label was applied to the (btc\_market\_price) attribute to identify it as the target of prediction, allowing rapid miner to distinguish between it and the other attributes. a split of 70:30 was Selected to make sure the model had a sufficient portion of the data to learn from while keeping enough data for testing.

For the performance, the Performance (regression) operator was used because it is specifically designed to evaluate regression models by providing essential metrics such as RMSE, Absolute Error, Correlation and **R²**.

Linear Regression showed strong predictive performance, proving to be an appropriate choice for predicting patterns in Time-Series data.

A computer screen shot of a diagram

AI-generated content may be incorrect.

Figure 14- Process Flow of the Linear Regression Model Including Parameter Configuration.

Random forest-h2

The next model used was Random Forest. Initially, its results were comparatively weaker, this happened because the algorithm isn’t well suited with time series data and is better at handling regular datasets with features that don’t depend on each other. The model treats each row separately and doesn't remember what came before or after.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 15 - Process Flow of the Random Forest Model Including Parameter Configuration.

To mitigate this issue, multiple tuning strategies were implemented. These methods include adjusting the number of decision trees and enabling the random split option. This was done to increase the models’ diversity and make training quicker.

Before tuning, the model showed poor alignment with the target’s trend. However, after adjustments, the performance improved across all evaluation metrics. While it did not match or even get close to the performance results of the Linear Regression model. Tuning the Random Forest model has shown enough improvement to be considered for Ensemble model testing.

A graph of a person and person

AI-generated content may be incorrect.A graph of a person and person

AI-generated content may be incorrect.

Figure 16 - Random Forest Prediction Plot (After tuning)

Figure 17 - Random Forest Prediction Plot (Before tuning)

The original untuned model gives a flat prediction line that does not match the real price trend, which is a clear case of underfitting. After tuning the model tracks the overall trend somewhat better, especially in the early years. But it starts struggling in the following years because of the rapid jump in bitcoins price.

Neural net-h2

The Objective is to build models which can predict the price of bitcoin based on its historical data, so a supervised, time-series-respecting regression algorithm such as neural net is bound to be fitting for our business problem.

Given the nature of bitcoin pricing and factors affecting, including the possibility of non-linear relationships between attributes, and neural network’s ability to capture said relationships, it has been chosen as one of the core models.

Neural Network learns by a feed-forward and back propagation algorithm.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 18 - Outside Flow of process

The btc\_market\_price attribute, our goal attribute to be predicted, is set as label with the set role operator, then applied is the optimize parameter grid to automatically get the best

configuration for the neural network operator itselfA screenshot of a computer

AI-generated content may be incorrect.

Figure 19 - Configuration wizard for selecting the parameters

The following parameters were used inside the optimize parameters(grid) option to use the different combinations of training cycles, learning rate, and momentum to get the best configuration for the neural net.

Training cycles: 200, 500, 1000

Learning rate: 0.01, 0.05, 0.1

Momentum:0.2,0.5,0.7

Which would produce 27 different combinations of these 3 parameters to get what would be the best output.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 20 - Configuration inside the optimize parameters operator

Cross Validation with 5 folds and linear sampling was chosen to optimize the best possible results in relation to our time-series dataset.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 21 - Configuration of Cross Validation to get the best input

A screenshot of a computer

AI-generated content may be incorrect.

Figure 22 - Results from the optimize parameters grid operator

The lowest RMSE from all 27 combinations was the combination of 200 training cycles, a learning rate of 0.05, and a momentum of 0.7.

After finding the optimum parameters for the neural net model, we’ve extracted it into another model.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 23 - Flow of Final Model with optimized parameters

Splitting the model 70:30 training/testing to ensure its accurate and get the correct performance numbers by means of performance(regression), evaluated on RMSE, Absolute Error, Correlation and Squared Correlation. Where it performed quite well, especially compared to random forest.

Ensemble Model -h2

Lastly comes the ensemble modeling, wherein we put together the different models that we have built upon our dataset. There are multiple ways to build an ensemble model, we have chosen to use voting for our ensemble technique. Voting is a powerful ensemble technique that balances bias and variance, where LR has low variance but high bias, and NN has low bias but high variance. It also mitigates risks of overfitting, especially coming from a complex model like neural network.

The ensemble model’s flow looks similar to the flow of each model separately as its trained and tested in the same way, with the difference being that the model being applied is the output from the results of the voting operator.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 24 - Ensemble Model's flow with the split of the data consistent with our separate models' configuration

A screenshot of a computer

AI-generated content may be incorrect.

Figure 25 - Configuration inside the Vote Operator

Inside the Vote Operator we placed the models with the configuration that we have previously optimized. Several iterations have been ran, including a combination of all 3 as shown in the picture, a combination of LR and NN, and a combination of LR and RF.

Thereafter, we feed the testing data into the apply model to get the prediction output from the voting ensemble, and feed the result into the performance vector as well, to be able to evaluate its performance.

Task 5, Model Evaluation

The different models have been evaluated been on 4 different performance criterion, since all of our models are classified under regression models, so the 4 criterion are Root Mean Squared Error(RMSE), Absolute Error(AE), Correlation(R) and Squared Correlation(R2).

A graph showing a line

AI-generated content may be incorrect.A black text with black text

AI-generated content may be incorrect.The first model’s prediction that was tested on the dataset is Linear Regression, it was the best performing model, having the lowest RMSE at 153.348, absolute error at 94.657, R of 0.992 and R2 of 0.983. It is expected for LR to be the best performing model as the linear, time-series nature of bitcoin prices works best with a simple model like linear regression, especially when other attributes such as btc\_transaction\_volume or btc\_hash\_rate move up or down with the btc\_market\_price attribute, our target attribute.

Figure 26 - Visualization between btc\_market\_price and LRs predicted values

Figure 27 - Linear Regression's Performance Vector

As it can be seen here, due to the aforementioned reasons, LR is within very close reach of the actual price of bitcoin throughout the dataset.

A graph with green and blue lines

AI-generated content may be incorrect.A black text with numbers and a black text

AI-generated content may be incorrect.Next up to be evaluated is the Random Forest model, with a RMSE of 889.614, absolute error of 428.586, R of 0.783 and R2 of 0.612. It was the least accurate model out of the 3 models we have used due to the nature of the bitcoin dataset. RF excels at handling noisy, nonlinear, high featured datasets since it creates a lot of decision trees which splits data into separated regions, which is not ideal for a (mostly) smooth, continuous attribute such as bitcoin’s market price. RF predictions are based on averages of leaves, which is counterintuitive for continuous data.

Figure 28 - Random Forest's prediction against actual market price

Figure 29 - Random Forest's performance vector

Due to the nature of random forest as a classification algorithm, it was not able to keep up with bitcoin price’s occasional volatility and spikes.

A screenshot of a computer code

AI-generated content may be incorrect.The last model to be tested is the neural net, which performed right in between LR and RF, showing RMSE of 623.387, AE of 310.208, R of 0.953 and R2 of 0.909. In general, NN performed adequately as it can capture interactions between attributes, it has very flexible learning capability due to the complexity of the algorithm and can approximate continuous functions, thus leading to good predictions. On the other hand, neural networks perform very well for huge datasets, larger than our dataset of 1584 examples, and due to the linearity of bitcoin’s market price, linear regression simply performs better.

Figure 30 - Performance Vector for Neural Net

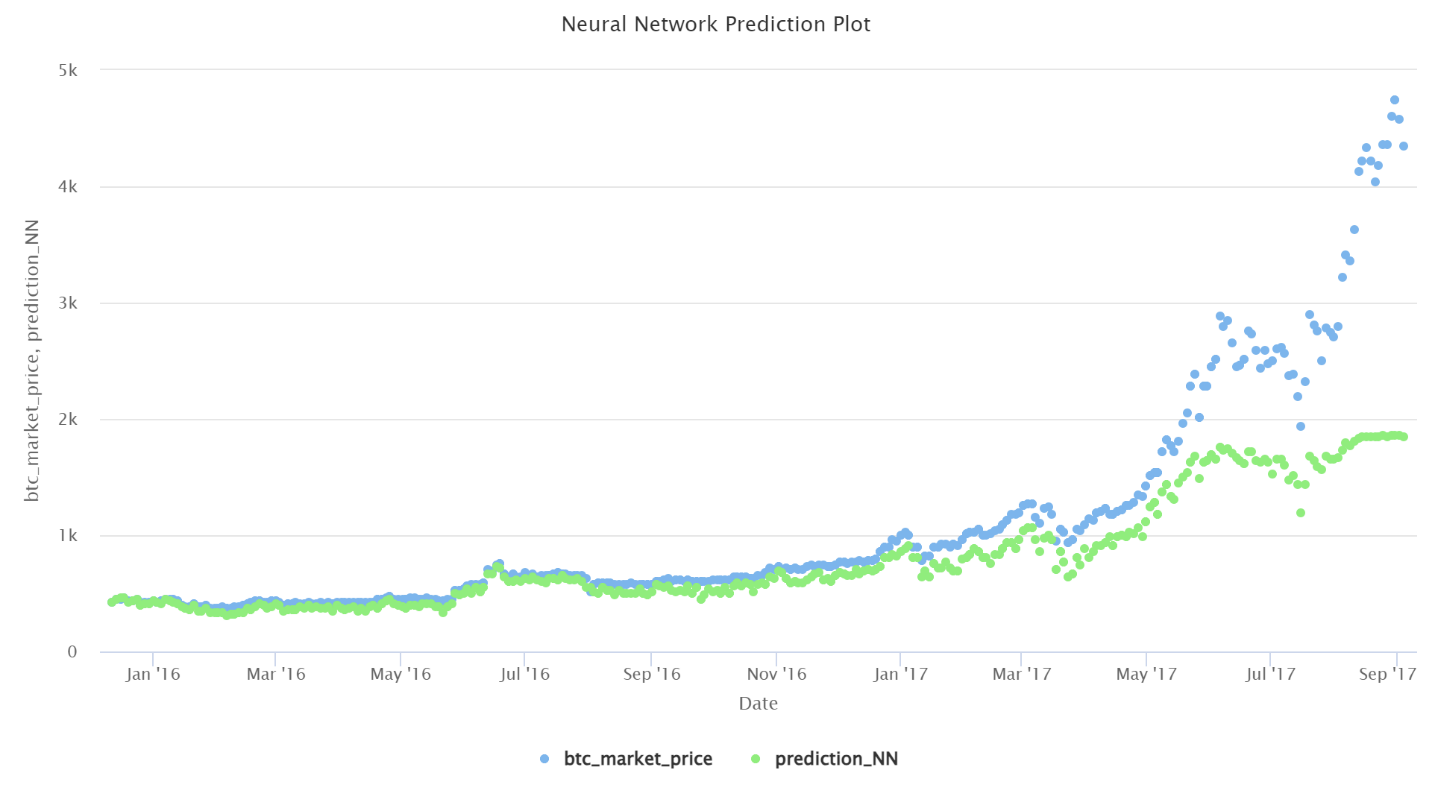


Figure 31 - Neural Net's prediction against actual market price

We’ve tried numerous ensemble models, including a combination of all 3, LR&RF, and a LR,NN combination.

The one to be highlighted is the combination of Linear Regression and Neural Network, unsurpsingly, it was our best performing ensemble model due to the fact that their individual respective models were the best performing out of the 3.

A black text with white text

AI-generated content may be incorrect.It produced an RMSE of 305.313, an absolute error of 144.503, an R of 0.983 and R2 of 0.967, performing better than our neural network model alone, but slightly worse than the linear regression model. Due to the LR model being very highly suited for the data, the complexity added by the slightly less accurate neural net, even after benefitting from the strengths of the neural net, the linear regression alone remained superior for predicting bitcoin’s price in our dataset.

Figure 32 - Performance Vector for LR NN Ensemble model

A graph showing a line of growth

AI-generated content may be incorrect.

Figure 33 - Scatter plot of the best performing ensemble mode

Overview of each model’s performance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **RMSE** | **MAE (Absolute Error)** | **Correlation** | **R² (Squared Correlation)** |
| **Linear Regression** | 153.348 | 94.657 | 0.992 | 0.983 |
| **Random Forest** | 889.614 | 428.586 | 0.783 | 0.612 |
| **Neural Network** | 623.387 | 310.208 | 0.953 | 0.909 |
| **LR + NN** | 305.313 | 144.503 | 0.983 | 0.967 |
| **LR + RF** | 429.867 | 217.240 | 0.989 | 0.979 |
| **LR + RF + NN** | 487.046 | 235.055 | 0.983 | 0.967 |

Some other visualizations encompassing all 3 individual models fared against each other, and all 3 ensembles fared against each other. A graph of a graph

AI-generated content may be incorrect.

Figure 34 - Scatter plot of all models individually against the market price

This scatter plot shows precisely how much better linear regression is compared to the other models individually.

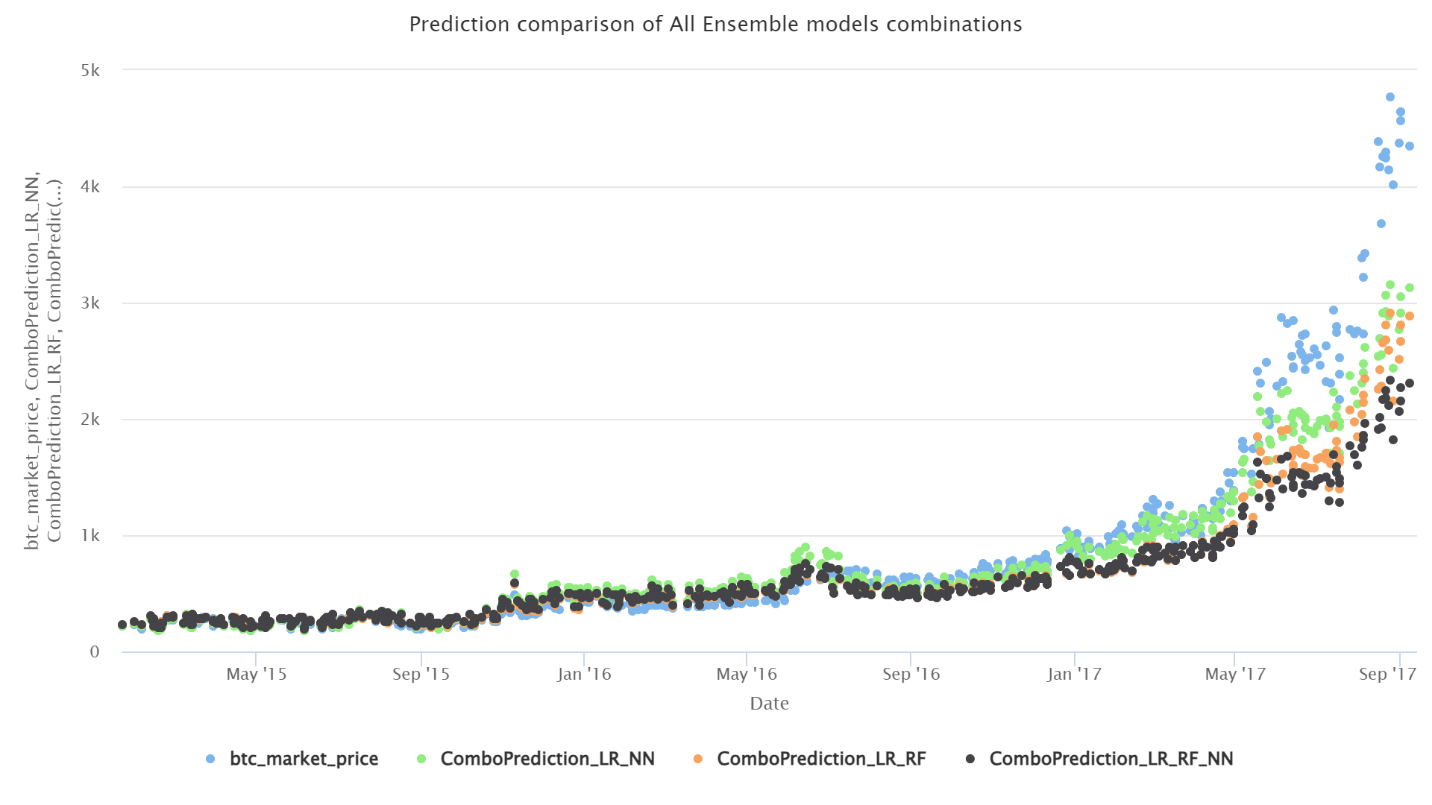


Figure 35 - Scatter plot of all ensemble models against the bitcoin market price

Task 6

Inferences and insights -h2

* Linear Regression Dominated
  + Outperformed Neural Net and Random Forest
  + Showcases that the relationship between the blockchain attributes is mostly linear, which is LRs strong area, as it excels in regression, time-series datasets.
* Neural Network’s complexity was not enough to outperform Linear Regression
  + NN captured non-linear interactions, but with the small size of the dataset, it was not able to utilize its full strength
  + The added complexity has lead to slight overfitting
* Random Forest was comparatively weaker
  + Random Forest excels in classification problems with noisy data, not data with mostly smooth and linear attributes
  + Tuning RF’s parameters has shown improvement in prediction, but was not enough to overtake the other 2 models.
* The 2 best individual models performed the greatest in an ensemble
  + Linear Regression and Neural Net were able to complement eachother and produce the best ensemble model out of all combos
  + LR has low variance and high bias while NN has high variance but low bias, providing a good trade off between generalization and adaptability

Recommendations -h2

There are multiple reccomendatins which could be done to improive upon this project, both technically and business wise.

On the technical end, External factors should be implemented into the predictive algorithms, as it is not just the blockchain features included into our dataset that affect bitcoin’s pricing, real-life events, interest rates and stocks all factor into it too.

Real time predicition would also be great improvement for the existing system, implementing real-time data piplines to feed into the model, and get accurate, up-to-date pricing predicitons.

On the business end, the created models could be infused with price risk analysis, or investment decisions, used to drive business decisions, and accelerate business growth.

Tying into the technical improvements, the real time updated predictions could be fed into automated dashboards using software such as Microsoft PowerBI to give potential investors easy access to real-time, interpretable data.

Time constraint was also a big factor into the project’s implementation, improvements that could’ve been made but time constraints affected are as follows:

Implementation of Deep Learning models, such as RNNs which are good for time-series sequential data such as bitcoin prices, with numerous hidden layers and a larger data set, a model such as RNN could prove to be very accurate.

Implemention of LSTM or ARIMA models which better fit time series models and could prove more accurate, but RapidMiner inferences and performance casting proved difficult to properly achieve within a reasonable timeframe.

Refelection -h2

Working on the Project has shown us extensively the KDD process, from data selection and cleaning all the way to interpretation and evaluation of data.

We were able to get the skills of working with time-series sequential (yet volatile) data in a predictive context

Find the balance between model complexity, parameter tuning and optimization to get the best possible results out numerous models which work differently and should be treated differently when optimizing.

See how ensemble modeling techniques offer advantage in balancing strengths of different models while reducing weaknesses.

How there are certain metrics which allow us to evaluate the accuracy of any given regression model/algorithm such as RMSE, AE, R and R2.

We also were able to learn how to interpret Performance of models through visualizations such as scatter plots, and make inferences and insights all throughout them.

We also learned on how to effectively and efficiently troubleshoot shortcomings both on model building, model evaluation, and results interpretation. Through making these mistakes and having to fix them, we’ve gathered a deeper understanding of how machine learning pipelines function on a fundamental level.