# Capstone Project Report
## Machine Learning Engineer Nanodegree

## "Using Machine Learning Models to Forecast Bitcoin Price Movements"

Author: René Javier Rivero Arrieta
Essen, Germany
31.12.2020

### Abstract

Since its release in 2008, Bitcoin has marveled the world as a financial asset and store of value, however given its relatively short life span, the factors affecting its price have not been properly identified, price forecasts of traders are highly biased with emotions and opinions and its high volatility can be rarely predicted to create profitable trades. In this project, we gathered a vast set of stock market indexes and commodities price data, interest rates, fundamental Bitcoin parameters (such as hashrate and transactions) and technical indicators (such as relative strength index and moving averages), used Principal Component Analysis to reduce its dimensionality and applied XGBoost and Sklearn's Neural Network models to forecast day-ahead Bitcoin prices (as a regression problem) and predict its price movements (classification problem, using buy, sell and keep out labels).

As a result, with the PCA model we could reduce the dimensionality of the original dataset from 27 to 6 features, preserving more than 90% of the variance. When forecasting Bitcoin prices, Sklearn's neural network performed better than XGBoost, both visually (similarity of the curve) and also on the root mean square error on the testing data set (0.10 vs 0.15). When predicting trading signals using multi-class classification, both models resulted in low accuracy (31-60%). This might be due to a natural bias of the feed-in data set, since in historical data Bitcoin shows a higher tendency to have positive daily returns. In future studies, shuffling the dataset before training and validating is recommended and could potentially improve results.

### Introduction

Since the 20th century, mathematicians, economists, business analysts and traders have used computers to come up with different technical and fundamental indicators to predict the price movements on the stock market. In one article, The Economist ('The Rise of the Financial Machines', 2019) mentioned that algorithmic funds "account for 35% of America's stock market, 60% of institutional equity assets and 60% of trading activity", which is an impressive number for the amount of money that means.

More recent academic research has also explored the use of different Machine Learning algorithms to predict stock price movements. While one may find a lot of academic sources, I would mention some recent studies like 'Prediction of Stock Market by Principal Component Analysis' (Waqar, Muhammad & Dawood, Hassan & Guo, Ping & Shahnawaz, Muhammad &

Ghazanfar, Mustansar ali., 2017), ‘Forecasting daily stock market return using dimensionality reduction’ (Zhong X. Enke D, 2017), ‘High-performance stock index trading via neural networks and trees’ (Chalvatzis, C., Hristu-Varsakelis, D., 2020) or ‘An innovative neural network approach for stock market prediction’ (Pang, X., Zhou, Y., Wang, P., Lin, W., Chang, V., 2020), all exploring different algorithms and approaches to stock market prediction.

**Problem statement**

Financial products have evolved, as algorithms and technology also have. With relatively short span of life, one the best performing financial asset is considered to be Bitcoin, created in 2008 by an author (or group of authors) with the pseudonym Satoshi Nakamoto (‘Bitcoin: A Peer-to-Peer Electronic Cash System’, 2008). However, due to its short and also accelerated growth, Bitcoin has not been studied as other more mature financial assets.

For this reason, our objective with this Capstone Project, will be to leverage on different machine learning algorithms to:

- Understand the principal components affecting the movements of Bitcoin prices
- Create a *price forecaster* for Bitcoin prices, using regression techniques
- Create a *trading signal predictor*, using multi-class classifier to predict buy, sell or keep out signals.

**Project design and methodology**

In order to achieve a solution to the above-mentioned problem, I’ve split the project into four notebooks containing all the necessary steps on the machine learning workflow, from data sourcing, to model selection, hyperparameter tuning and model deployment and testing. Each notebook is named and includes the following:

1. **Data sourcing and preparation**: Includes data sourcing and visualizations, data cleaning (for example, NA values), normalization of data sets and correlation analysis.

2. **Feature Engineering**: Includes Principal Component Analysis and feature assessment.

3. **XGBoost**: Includes price forecasting and multi-class trading signal predictor, and partial results analysis from the XGBoost model used.

4. **Neural Network:** Includes price forecasting and multi-class trading signal predictor, plus final discussion and results comparison of both models.

By the end of the project, we would have a better idea of the different variables affecting the price of this revolutionary asset, and a predictor of its price movements.

**Evaluation Metrics**

While in the project proposal I've originally stated that I would compare returns comparing the different strategies, after performing the analysis it was clearer that there were clearer metrics to take a look at, to evaluate results and also for comparing the XGBoost model results against the Neural Network. These metrics are namely:

- For performing the dimensionality reduction and feature engineering, we would look at the **explained variance to select the number of features** taken.

- For comparing **price forecasting, the Root Mean Square Error (RSME)** would be a more adequate comparison metric, since it measures the distance of the predicted data point to the actual measure.

- For comparing **multi-class classification, the accuracy** would be a better metric, since it compares the number of correct predictions to the total number of predictions.

**Data exploration and preprocessing**

The datasets to be used in this study mainly relied on stock market price data and fundamental Bitcoin data publicly available in different sources. To source this data, we used yfinance (Yahoo Finance) and Quandl libraries. Some of the characteristics of all these data sets are:

- They are all **time series data**, indexed with Datetime format.

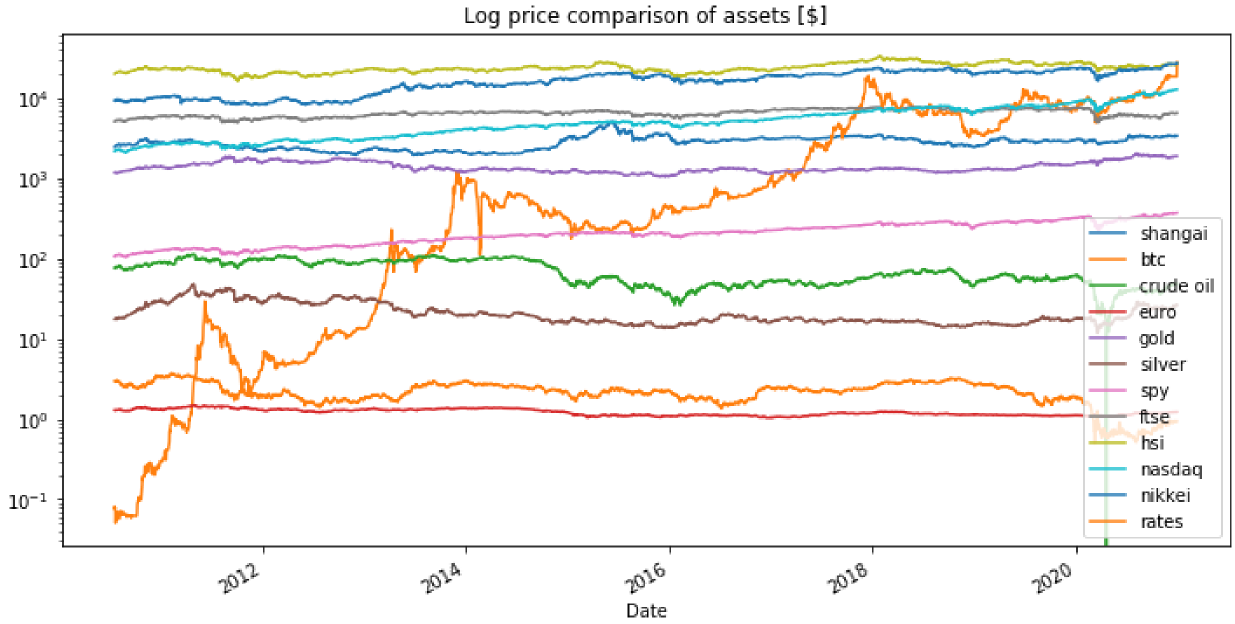
- The timeframe taken for the data points are **daily data points**.

- The start date was the beginning of Bitcoin in 2008, but we reduced the data set to a point where we have data from each of them (July 2010), since some important indexes were created afterwards.

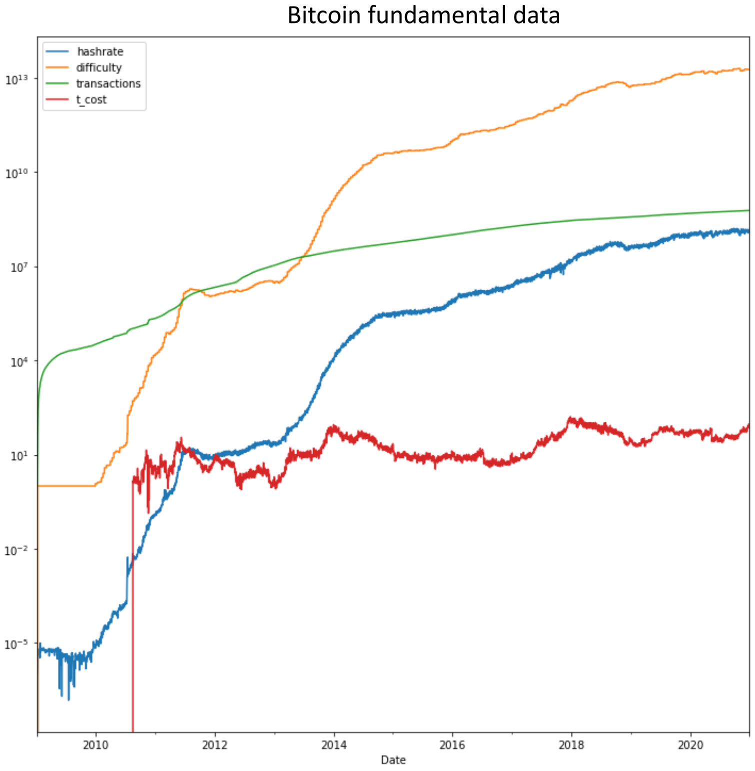
- **Price data usually in OHLC (open, high, low, close) feature format.** For our purposes we may focus on the 'closing' values if others are not considered relevant. Daily closing values will also be used for further calculations needed.

Some of the features considered in the input data sets are:

- **Stock market:** Bitcoin price and volume, SPX500, NASDAQ, Dow Jones, DAX30, gold price, silver price, oil price, interest rates, amongst others, depicting general market conditions. Below is a visualization of the stock market features selected for this exercise.



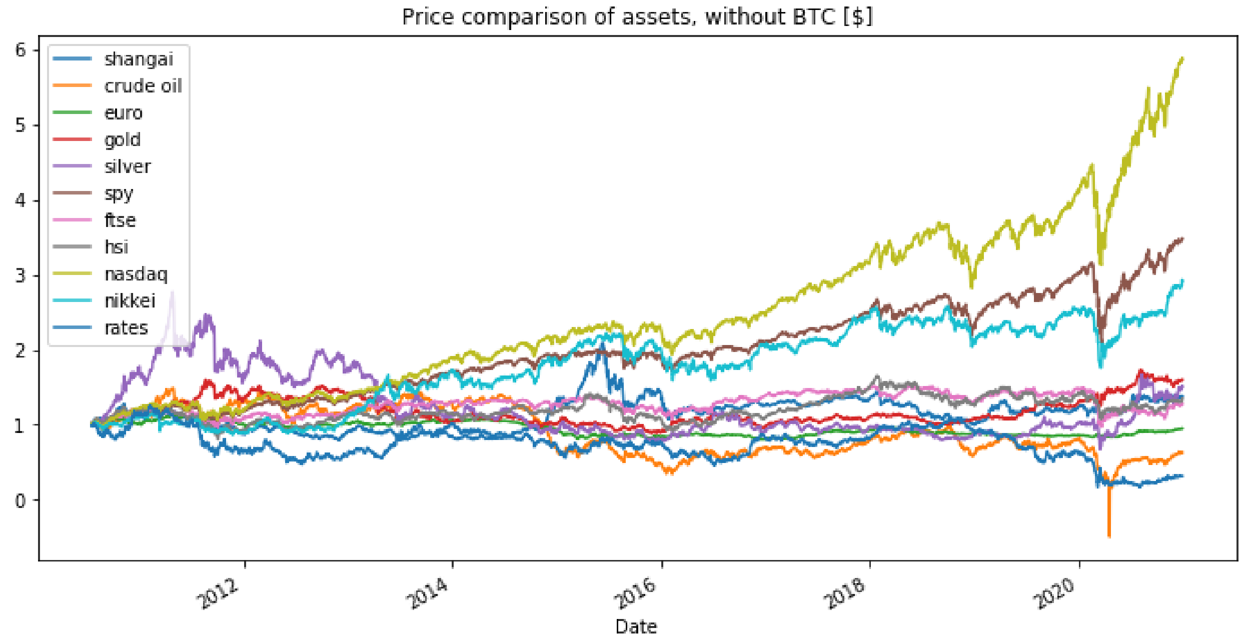
- **Bitcoin fundamental data:** hash-rate, block size, transaction fees. A visualization of the selected fundamental features can be seen in the image below.



Additional technical indicators, such as moving averages and relative strength index (RSI), were also calculated during the notebook and included in the feed in data set.

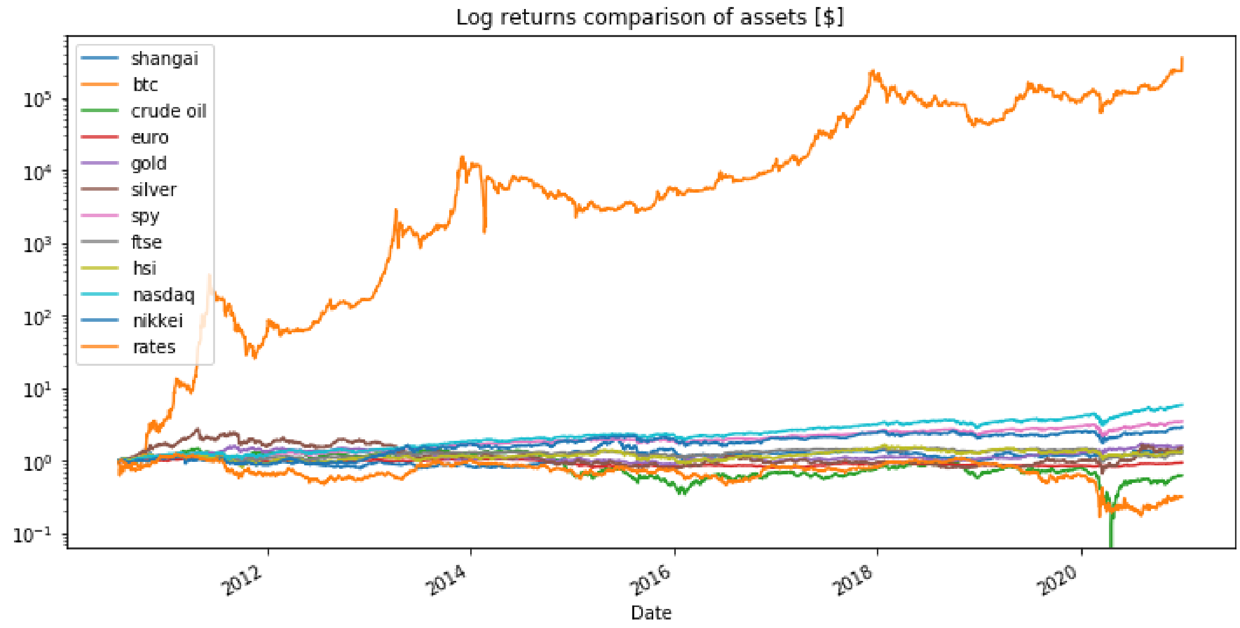
## Data Processing and Refinement

Since we could observe that stock market price data is very different from one another (for example, while Nikkei index trades around USD 10.000, S&P 500 index trades at around USD 5.000), we adjusted this data to a standard base of 1 and show daily returns. This gives us a **better idea of the price change, instead of the spot price**. The obtained data set can be appreciated in the image below.



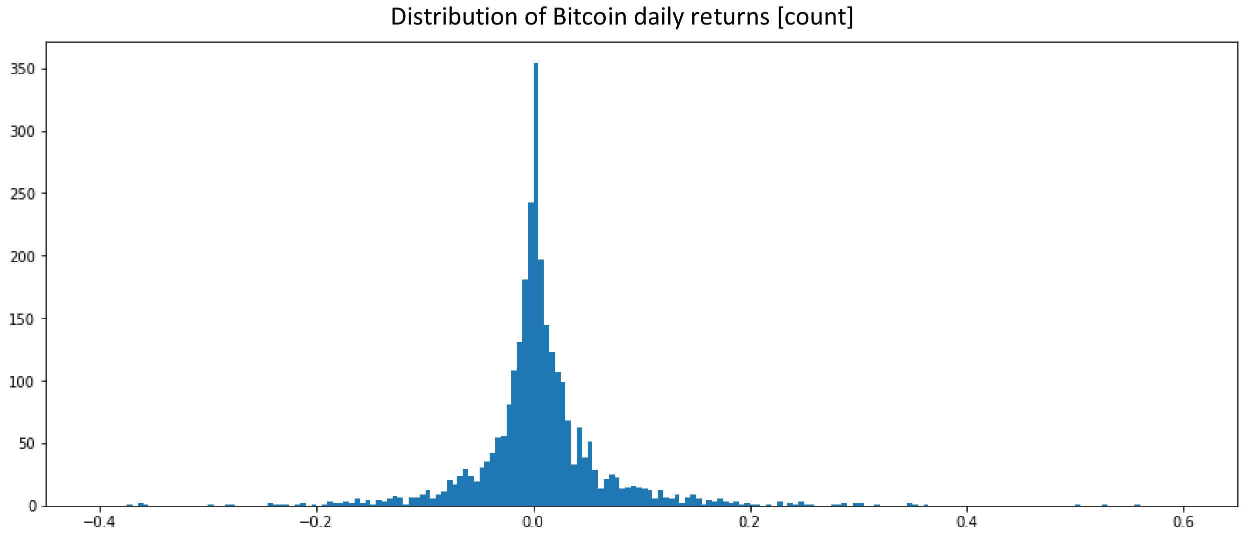
As mentioned before, **this is a better picture of how the asset value changes over time** and makes it more comparable to one another. We can see, for example, that the NASDAQ index gathering technology companies has grown 6x times its value from the start of the data set, whereas crude oil traded even at negative prices during March 2020.

This also brings up to an interesting point which is the massive price surge from Bitcoin, that can be appreciated in the image below. Since 2010, Bitcoin has increased its price 100.000x, which is not only astonishing, but also makes it the best performing asset evaluated in this data set.



## Labelling data for classification purposes

We were also interested on labelling and classifying the daily returns to create an appropriate training set. For this, we analyzed the distribution of the daily returns, which determined that 50% of the daily returns are always over +0.1%, while the top 25% are over +2.3% and the lower 25% of data points had daily returns of under -1.3%. The distribution can be seen in the image below. For our classification purposes, **we labelled days with returns on the lower 25% as 'sell' signal (0), the 50% range in between as 'keep out' (1) and the higher 25% as 'buy' signal (2).** Afterwards, labels were shifted so we would be actually predicting next-day movements.

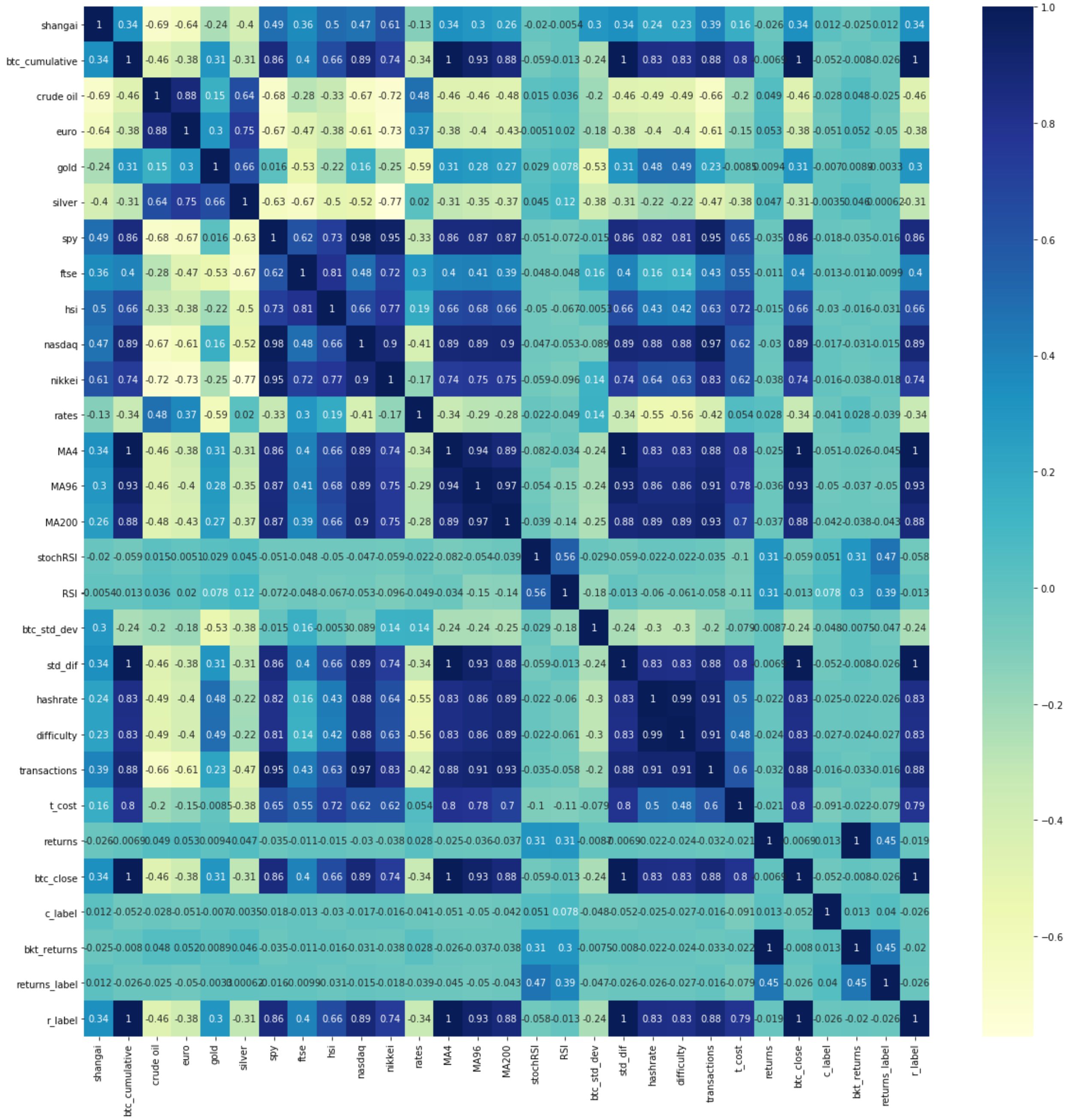


Some additional interesting data shows **that on its worst day, Bitcoin had a price decrease of -57%, while on its best day it made an astonishing +336%.**

## Correlations between initial features

In order to verify that feature engineering would be a good step afterwards, we made an initial correlation analysis using a heatmap, amongst all the initial data set features. The heatmap can be seen in the image below.

Correlation heatmap between initial dataset features



As expected, we can already see some groups of features that are highly correlated not only to Bitcoin, but also between each other. This is already a good sign that performing dimensionality reduction could be a good step towards decreasing the number of features to use to feed in our follow-up models, so that these can be trained faster and more efficiently, making sure as well that we do not lose the variance explained in the original data set.

**Data cleaning, normalization and split into training and testing data**

As final cleaning steps for the original data set, the following steps were taken:

- **Used a self-made min-max scaler function to normalize the data,** and also be able to bring it back to an interpretable state.

- **Got rid of NA values,** first by reducing the data set to where we had available data points on all features, resulting in a data set with more than 2700 rows and 29 columns. Afterwards we performed back- and forward filling as needed per step, to make sure we got rid of possible errors in our data.

- **Labels were shifted one step back,** so we actually predict day-ahead price and signals.

- **Split the data in training and test data sets** in the following way:
  - Training data would be from start date until 2017-12-31
  - Test data would be from 2018-01-01 onwards.

The reason for this specific timeframe split, is that I wanted to include all bullish (surging prices), bearish (decreasing prices) and stable scenarios both on the training and testing data sets, so we avoid biasing as much as possible. The selected timeframes included all three scenarios in the training and testing data sets.

**Algorithms and techniques used**

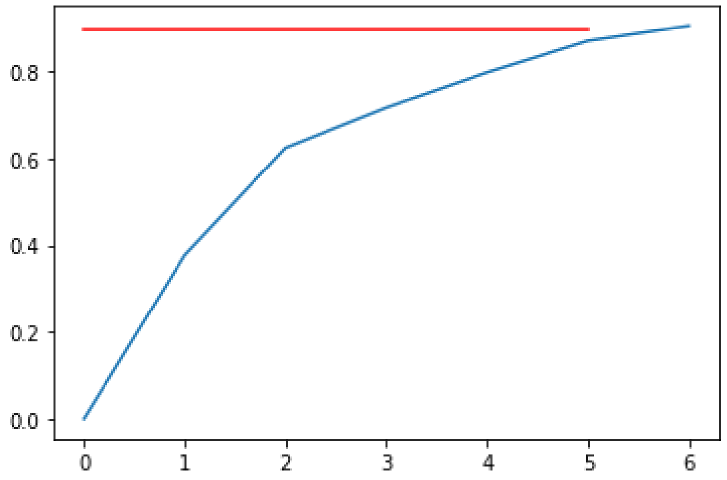
From the set of daily price and technical indicators, **we will use Principal Component Analysis to determine the most relevant data vectors affecting Bitcoin price** and select appropriate data inputs for our price estimator at a later stage.

Once the principal components affecting Bitcoin price have been identified using PCA and feature engineering, I will use them to train a model to either forecast, or estimate ‘buy’, ‘sell’ or ‘no trade’ signals optimizing intra-day returns as the optimization variable. For this, my **benchmark model will be XGBoost**, given its good and lightweight performance, and additionally **we will be evaluating a Neural Network against it**, since consulted bibliography have proven good results for price predictions of other assets in the past.

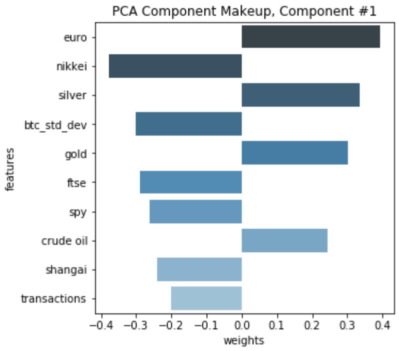
**Principal Component Analysis (PCA)**

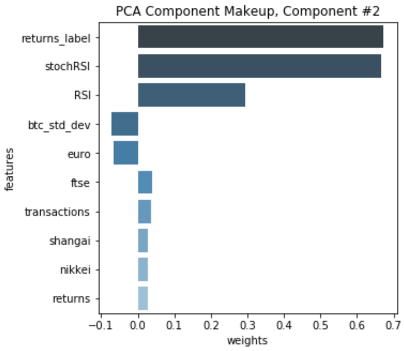
After training and deploying a PCA model using AWS Sagemaker, we observed that **0.91% of the total variance of the dataset could be explained using just 6 reduced components**. For this we used a visualization showing the explained variance against the total components used. As we use more components, more variance can be explained, and we aimed for a threshold of 0.9 (red line) which we could achieve with 6 components (out of a total 25). A graph depicting this analysis can be seen below.

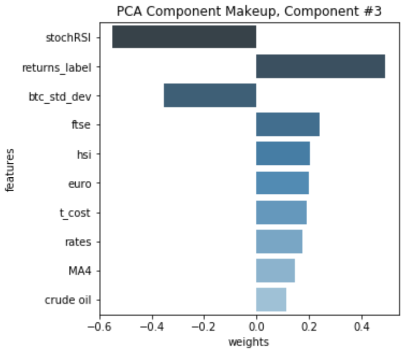
Explained variance vs. components used [#]

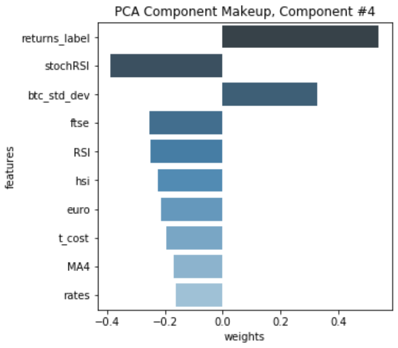


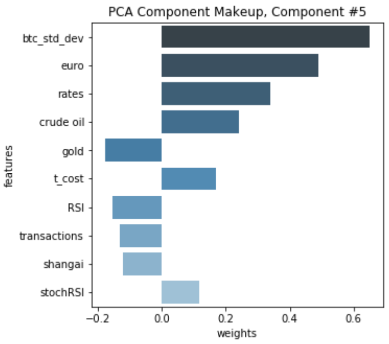
Furthermore, we can have a look at the chosen components in the images below. Interestingly, **we can observe that the model is weighting important market features in the newly created ones**, such as the most traded pair worldwide (EUR/USD), commodities (gold, silver, oil), top stock indexes (SPY, Nikkei...) and some of the technical indicators (relative strength index, moving averages, standard deviation, previous day labels and price).

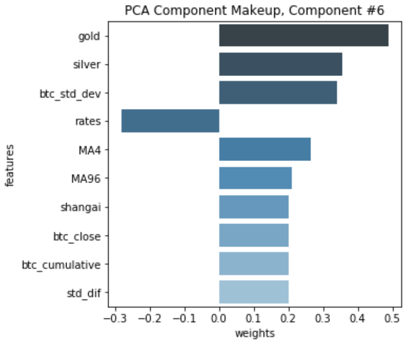












After analyzing these components, we deployed the model to predict the relevant vectors of the training and testing data sets and stored these datasets with reduced components to feed follow-up models.

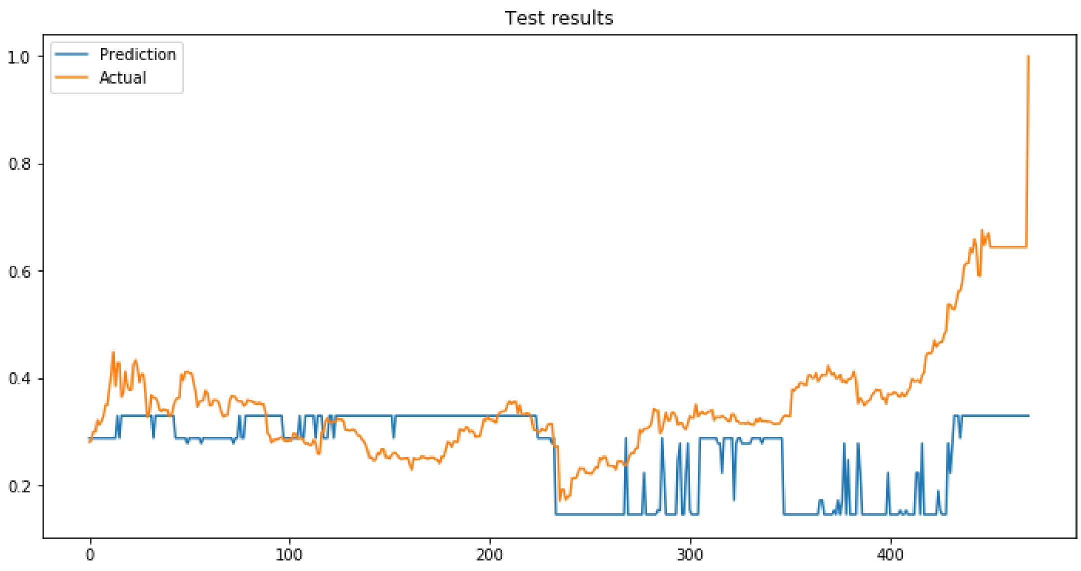
**XGBoost**

Given its reported lightweight and good performance for forecasting and classification applications, we will be using XGBoost as a benchmark model to create a Bitcoin price forecaster and a trading signal predictor.

XGBoost – Bitcoin Price Forecasting

Since predicting Bitcoin prices involves a forecasting problem in its essence, we used as a training objective ‘reg:squarederror’, which also in line with our evaluation metric (RSME). Besides training, we additionally tuned our hyperparameters using Sagemaker hyperparameter tuner, and used the recommended ranges given in Sagemaker documentation for this.

Given that using the Hyperparameter Tuner involves performing a number of training jobs iterating over the ranges of hyperparameters, the training of the model lasted around 10 minutes. As a result, we got an RSME on the test data of 0.15, with predictions having a stepped curve that can be seen below.



As we can see, neither the resulted curve seems optimal to solve this problem, nor the resulting RSME on the test data. Let's see how this compares to the results of the second model later.

### XGBoost – Trading Signal Predictor

For predicting trading signals, we will consider this to be a multi-class classification problem that given the feed data, will predict day-ahead price movements using three labels – buy (2), keep out (1) and sell (0). In order to achieve this, we defined as the training objective of XGBoost 'multi:softmax' and for hyperparameter tuning we will use maximize our model accuracy (also in line with our evaluation metrics), so Sagemaker's Hyperparameter Tuner can train a number of jobs and select the best one.

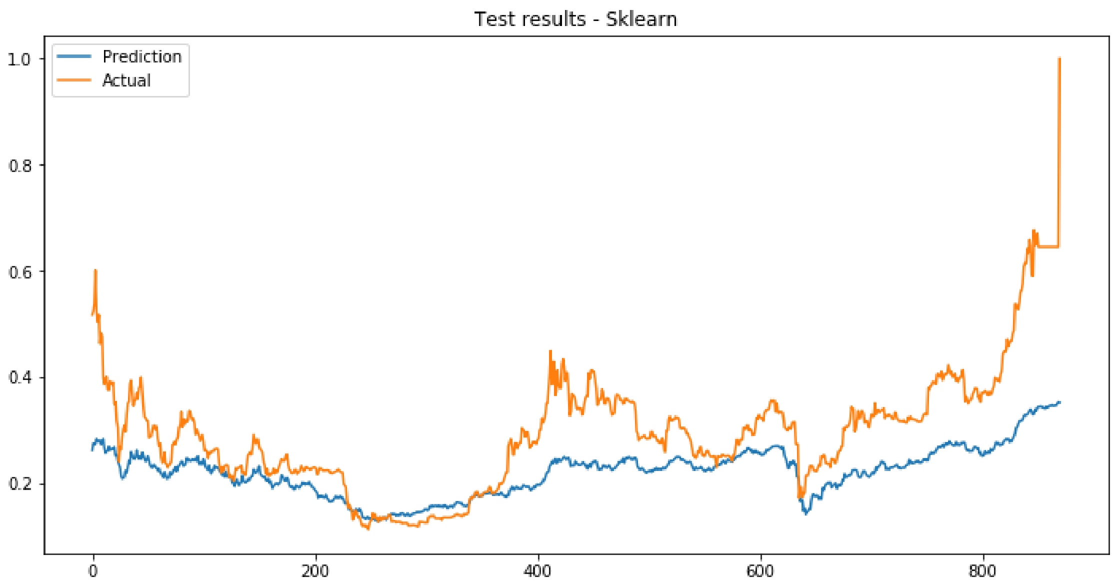
Given we used Hyperparameter Tuner, training and tuning this model lasted around half an hour, and **as a result we got a classifier with 60% of accuracy.** However, on the test data set we got a 'keep out' signal throughout the data set, which is definitely not optimal

### Neural Network

To deploy a model using a Neural Network, we used SKlearn given that it has two powerful yet simple algorithms that we could quickly use for the two tasks ahead – forecasting and multi-class classification. Namely, **we used Sklearn's MLPRegressor to forecast Bitcoin prices**, and the **MLPClassifier for predicting the trading signals** (buy, sell, stay out).

### Neural Network - Bitcoin Price Forecasting

The model was trained very quickly (only 1 minute) and the deployment lasted 8 minutes and 32 seconds using Sagemaker. Below you may find a graph depicting the predicted results against the actual ones.



Our comparison metric, **the RSME was computed to be 0.10 on the test data** using this approach.

### Neural Network - Trading Signal Predictor

This model was trained in 3 minutes and 43 seconds, and its deployment lasted 7 minutes 31 seconds using AWS Sagemaker resources. After deployment, we computed its accuracy, **achieving a 31.4% accuracy on the test data set**.

### Results Discussion, Evaluation and Comparison

We can summarize our results obtained during this project in the following table:

| | Principal Component Analysis | XGBoost | Neural Network (Sklearn) |
|---|---|---|---|
| **Evaluation Metric** | 90% explained variance retained using 6 components out of a total 27. | RSME of 0.15 on forecasting.<br>Accuracy of 60% on classification. | RSME of 0.10 on forecasting.<br>Accuracy of 31% on classification. |
| **Graphs and Interpretability** | Components identify and weight in key correlated features, preserving its essence. | Suboptimal stepped curve on forecasting. 'Keep out' signal throughout the whole test data set. | Good curve on forecasting. |
| **Observations** | Great results overall | Suboptimal results for forecasting. Classification is more accurate, but only returns one label. | Best for forecasting, classification could be improved. |

Given the summarized results above, we can conclude that the PCA model did a great job reducing the dimensionality of the data set, without compromising the explained variance of the original features. On the other hand, when comparing XGBoost against SKlearn's neural network, the neural network performed much better on forecasting Bitcoin prices with lower RSME and a better fit to the curve. Finally, when predicting price movements, XGBoost scored better in accuracy but it was mainly due to the fact that it recommended to keep out of any trade throughout the test dataset, which makes it suboptimal to rely on. This was not the case on SKlearn's neural network, which had a 31% accuracy, but its results seemed more logical. The suboptimal results on classification tasks might be due to a natural bias of the feed-in data set, since in historical data Bitcoin shows a higher tendency to have positive daily returns. In future studies, shuffling the dataset before training and validating is recommended and could potentially improve results.

**References**

Chalvatzis, C., Hristu-Varsakelis, D., 2020. *High-performance stock index trading via neural networks and trees.* Applied Soft Computing, Vol. 96, November 2020, 106567.

Pang, X., Zhou, Y., Wang, P., Lin, W., Chang, V., 2020. *An innovative neural network approach for stock market prediction.* The Journal of Supercomputing 76, 2098-2118(2020).

Satoshi Nakamoto, 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf

The Economist (2019) *The Rise of the Financial Machines.* The Economist Group Limited, London 2019. https://www.economist.com/leaders/2019/10/03/the-rise-of-the-financial-machines

Waqar, Muhammad & Dawood, Hassan & Guo, Ping & Shahnawaz, Muhammad & Ghazanfar, Mustansar ali., 2017. *Prediction of Stock Market by Principal Component Analysis*. 13th International Conference on Computational Intelligence and Security (CIS).

Zhong X. Enke D, 2017. *Forecasting daily stock market return using dimensionality reduction.* Expert Systems with Applications, Vol. 67, January 2017, Pages 126-139.