# Solution Design - Market Timing Strategies for Crypto currencies using Kalman Filters

By

Rakesh Sharma P. R

*WorldQuant University, Master of Science Financial Engineering*

[*rakeshsharma.pr@gmail.com*](mailto:rakeshsharma.pr@gmail.com)

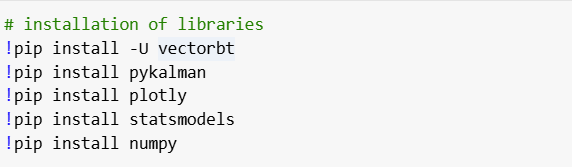
# Introduction

This document is a high-level design document explaining the solution implementation for the research paper – on market timing strategies using Kalman Filter. In this document we will discuss the methodology used in the research process and critical code implementations, design and validation techniques that lead to our research conclusion. All the code samples shared here can be seen in the google colab Jupyter notebook attached in the reference section.

# Pre-Requisites

The code implementation is done in python as the python has become the de-facto language for research and data science projects. There are several libraries that we used in this research project and it is easy to replicate and reproduce in a Jupyter Notebook environment. As a pre-requisite a user of this project should have access to Google Colab Notebook or a local Jupyter Notebook environment. Pip is an environment management tool for python and will help replicate the current settings in any user environment. The below diagram shows the important libraries and their latest version installed.

**Fig. 1. Pip installation of libraries**



As part of the research, we are exploring 2 new python implementations specific to this research. They are ***pykalman*** and ***Vectorbt***. The documentation for both is given in the below sites

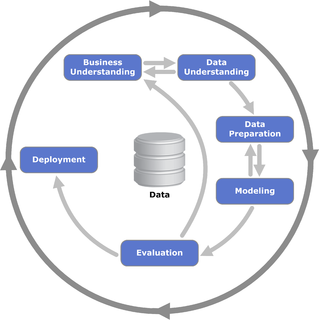
<https://github.com/pykalman/pykalman>

<https://vectorbt.dev/>

The reason for using these libraries is that they are easy to implement and there are numerous examples and resources to get started for users.

# Research Methodology

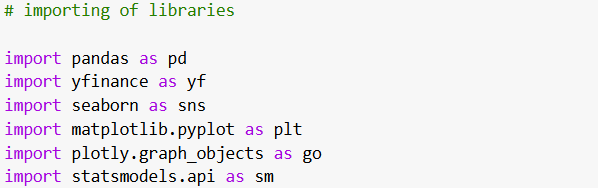
In this research we have followed an industry standard of model life cycle of our research implementation which is called CRISP-DM and has the following stages [2]. In the below section we have highlighted the various stages and cited critical code for various stages. This is nicely depicted in the below diagram and is an iterative process.



**Fig. 2. CRISP-DM cycle [1]**

# Data Collection

We are doing research on crypto data namely Ethereum and Bitcoin and we are extracting it from the website <https://finance.yahoo.com/>. There is a python library to extract these data from the website and can be done using this below code section.



**Fig. 3. Importing of libraries**

Here we import the implementation as ***yfinance*** and download the data using the ticker name for the time period of our interest.

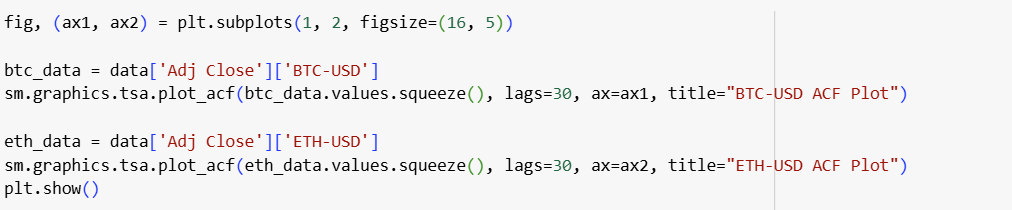
**Fig. 4. Downloading of data using yfinance**



# Data Understanding

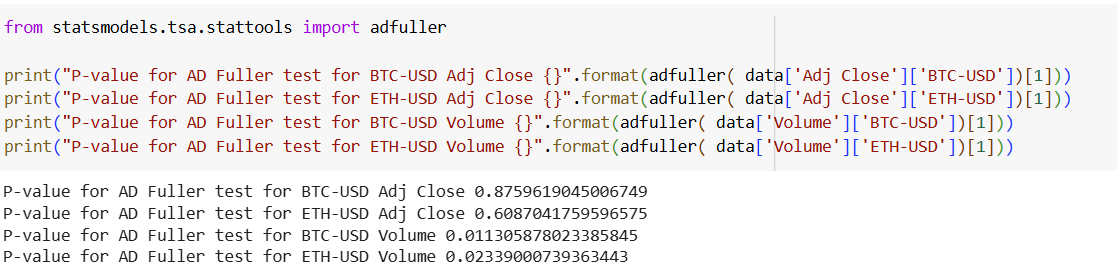
We use visualization libraries such as ***matplotlib*** and ***plotly*** for various understanding the trend, noise and seasonality by computing their rolling means for various time periods. During exploration we also understand the correlation between two asset values, and observe their autocorrelations.

**Fig. 5. Calling ACF and PACF plots to understand autocorrelation**



We have used ***statsmodels*** for implementing AD fuller test for stationarity. The below section of code shows the implementation of this test.

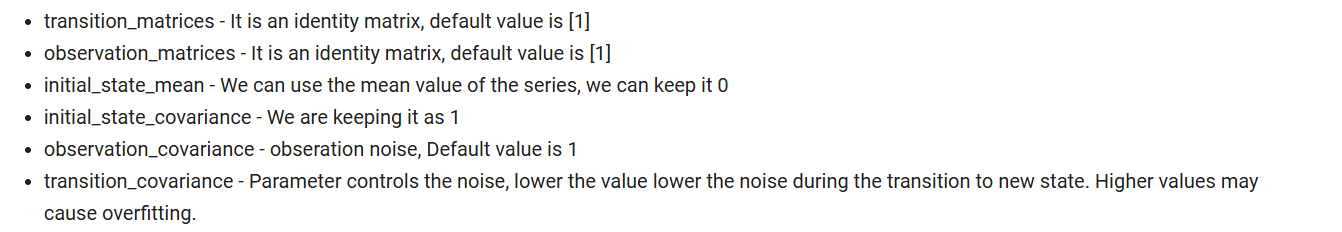
**Fig. 6. Calling ADF test for stationarity**



# Modeling - Kalman Filter Implementation

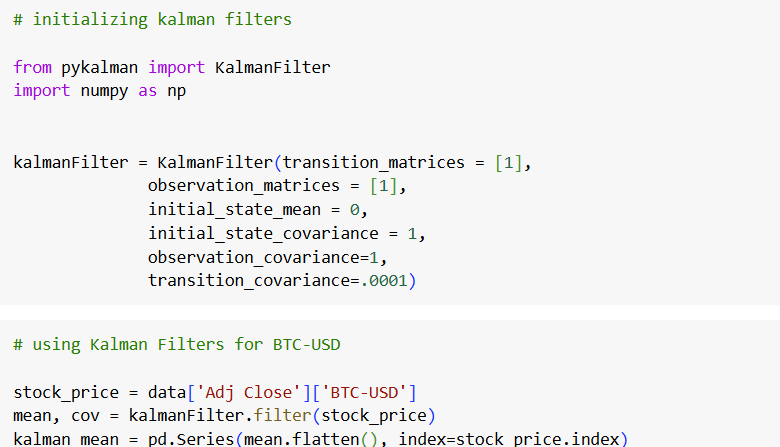
We use Kalman Filter class implemented in pykalman library and it takes these parameters. Transition covariance is the hyperparameter tuned for fitting the model on different datasets.

**Fig. 7. Definition of Kalman Filter Parameters**



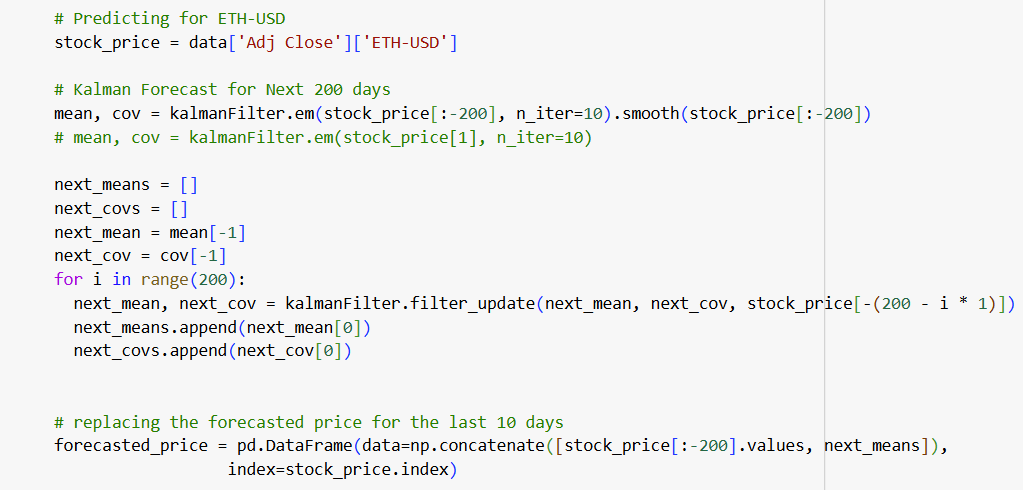
The code for implementing the filter is as shown below

**Fig. 8. Defining Kalman Filter object and calling filter method**



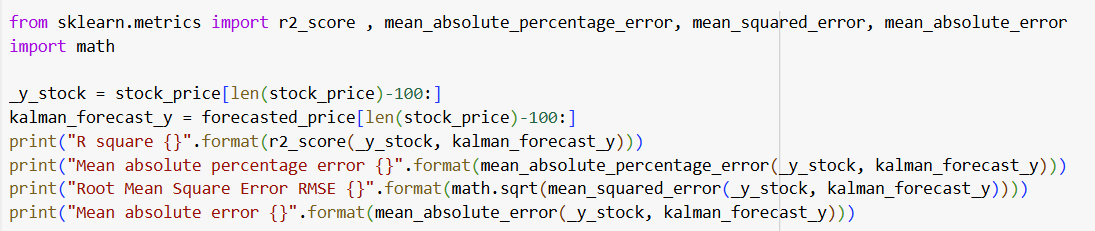
We can use the filter update method to get the forecast for the validation data set.

**Fig. 9. Calling filter update method for the forecast period**



The forecasted values are tested for model performance metrics implemented in ***sklearn*** package as shown below. This is same for output for linear regression predictions.

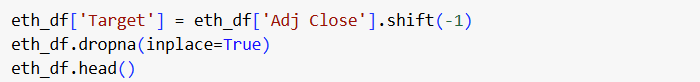
**Fig. 10. Calling model metrics for Kalman Filter forecast**



# Modeling - Linear Regression Implementation

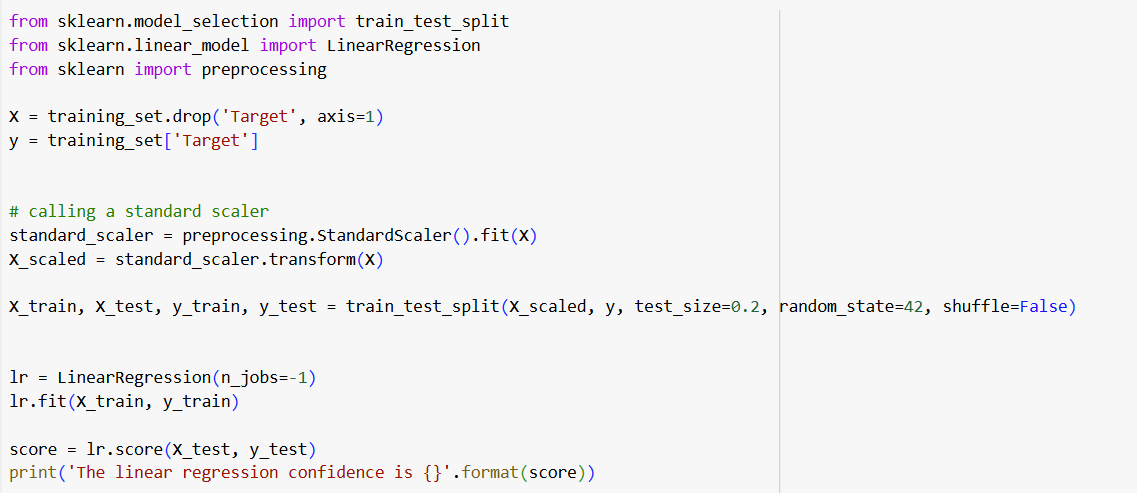
In this modeling we first define the target variable, before splitting the data into training and forecast set. The forecast set is checked for performance metrics.

**Fig. 11. Defining the target variable, which is Adj Close Price for the next day**



The training and forecast set is scaled before calling the fit function.

**Fig. 12. Defining the linear regression model for prediction**

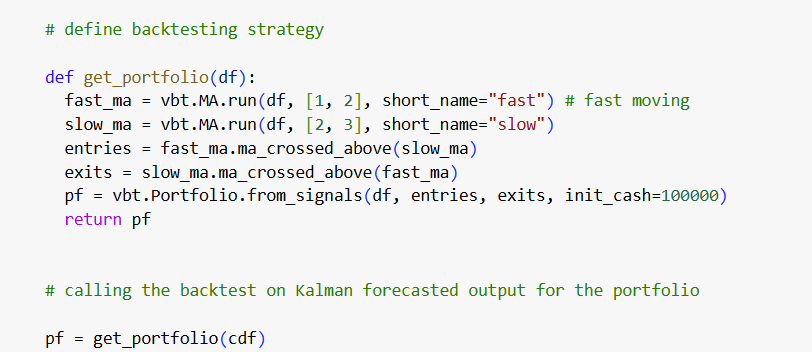


# Evaluation

We use ***Vectorbt*** to define trading strategy. We are using Double Moving Average Crossover implementation in the below function. Our portfolio is the predicted values of BTC-USD and ETH-USD using Kalman Filter and Linear Regression models.

We define slow moving and fast-moving time periods for our cross over, we have used short time periods such 1 and 2 days for out portfolio. These steps defined can be seen in the below code section.

**Fig. 13. Defining the DMAC trading strategy for our portfolio**



# References

1. https://en.wikipedia.org/wiki/Cross-industry\_standard\_process\_for\_data\_mining
2. <https://www.ibm.com/docs/sr/spss-modeler/saas?topic=dm-crisp-help-overview>
3. <https://colab.research.google.com/drive/1DHlMEGF2ZAOw8_1gXE1x8jITpTMfp4Hq#scrollTo=SsIFH2jcHVlR>
4. https://github.com/rakeshsharma14/WorldQuant-Capstone