American University of Science and Technology

Faculty of Engineering

Department of Computer Science

Final Project

Fall 2021 – 2022

Monday Feburary 1st, 2022

Fady Saoudy (20210158)

Louay Khaddaj (12190509)

Mohammad Sabra (12190027)

Submitted to:

Dr. Charbel Boustany

TABLE OF CONTENTS

In the Machine Learning and Data Mining course, the most effective machine learning algorithms are showcased followed by the basic methodologies that automate the detection of patterns by computers alongside general concepts of data mining. Not only are students introduced to the theoretical approach, but also the practical approach to real life struggles faced in the field, and are taught what the best algorithms to use in each case are.

Chapter 1 – Introduction

1.1 - Introduction

Course code: CSI469

Course name: Machine Learning and Data Mining

Chapters covered in this course:

Chapter 1: Introduction to Data Mining and Machine Learning

- Descriptive, Predictive, and Prescriptive Analytics
- Apply predictive analytics using Data Mining and Machine learning
- Methods of Data Mining
- Data Mining Methodology

Chapter 2: Data Mining and Statistics

- Average, mean, Standard Deviation,
- Variance, Normal Distribution, Zscore, Percentile, Histogram, probability value (P value), hypothesis test in statistics, t test
- Verification if input is relevant to predicted output in Applied machine learning
    - chi-squared test
    - Confusion Matrix
- Understanding and writing the conclusion
- Case studies

Chapter 3: Unsupervised Learning

- **Clustering Algorithms in Data Mining**
  - Goal of Clustering
    - Clustering Algorithms
    - K-Means Clustering
    - Hierarchical Agglomerative Clustering
    - Density-Based Clustering
  - Application of Clustering
- Association Algorithms in data mining
  - Goal of Association
  - Association Algorithms
    - Apriori
    - Dynamic Item set counting
- Application of Association -market basket analysis

Chapter 4: Supervised Learning

- Regression Algorithms in Data Mining
  - Goal of Regression
  - Regression Algorithms
    - Linear Regression
    - Multiple regression
    - logistic regression
    - Time sequence analysis
      - Simple Moving Average “SMA”
      - Weighted Moving Average “WMA”
      - Exponential Triple Smoothing
    - Techniques to measure errors: MAD, MSE, MAPE
  - Case studies
- Classification Algorithms in data mining
  - Goal of classification
  - Classification algorithms:
    - Decision Tree
    - Naive Bayes
    - Bayesian networks
    - Neural Network
    - logistic regression
  - Application of Classification

Falling within the emphasis of data science automatically marks Machine Learning one of the most important courses to be taken by a computer scientist. Having its prerequisites be advanced math courses and object-oriented programming puts this course on a pedestal. It introduces students to several algorithms that are prevalent in the work field and gets them ready for hands-on real-life situations by keeping them up to date with current topics such as COVID and Crypto currency predictions for example.

With the rise in value in cryptocurrency naturally comes various studies on how and why it has become such a predominant topic in today's world of finance. Creating a cryptocurrency prediction system is a very tricky subject to tackle, because if it were to be predictable, coin holders would've bought and sold coins at the optimal time guaranteeing nonstop gains making everyone a millionaire. However, Cryptocurrency is controlled by myriad factors and variables that are subject to abrupt changes, consequently making it unpredictable. In this project, we rely on the heftiest factors that control cryptocurrency to form an estimation of the daily peak, the low, opening and closing prices of coins, which gives holders an idea as to whether to buy, sell or maintain.

1.2 - Type of the Project

This project is a hybrid between theory and practical work.

Theoretical approach: Within this project, arbitrary numbers are used to formulate intangible predictions that can sometimes be far from true.

Practical approach: The project is implemented using python using its vast variety of libraries that ease the coding process. Python is versatile and multipurpose, open source and is considered a core language in data science and machine learning, so it is only natural that python was the go-to language when it came to building this project.

1.3 - Learning outcomes

- Demonstrate knowledge of various predictive Association modeling techniques
- Use a data mining program to analyze data and develop regression predictive models
- Compare and evaluate the accuracy of predictive models
- Understand and use classification data mining methods including DT, NN, NT
- Evaluate results produced by various data mining methods
- Execute knowledge discovery projects
- Demonstrate knowledge of key principles and techniques of data mining
- Apply fundamental concepts of data mining
- Use statistical methods and visualization to explore and prepare data
- Describe the theoretical constructs and core processes of unsupervised clustering data mining.

Chapter 2 – Problem Statement

This chapter introduces the problem statement and the challenges faced throughout the development of this project.

2.1 - Problem Statement

Cryptocurrency is unpredictable.

Having someone announce a prediction leads to so much traffic in compliance with that prediction. If a coin was predicted to rise in price and someone with a huge audience announces that it is going to rise, the masses rush to purchase that coin, which naturally shoots the price of that coin up due to increase in demand. And usually for someone to announce a prediction like that, they have to be one of the top holders in the coin so that they can be the ones selling the coin to the amateurs who responded to his fraudulent prediction. This is where capitalism in the world of cryptocurrency reveals itself. Besides its volatility, cryptocurrency and mining are very demanding in terms of hardware finances and electricity, sometimes leading to bankruptcy.

2.2 - Challenges that were faced

Cryptocurrency being unpredictable forces us to broaden our margin of error which consequently weakens the accuracy of our estimations.

Law challenges: cryptocurrency accounts being free for multiple initiation by the same person without having to use their real name has led to speculations in regards to the its safety, and the process of trading is considered a hoax. Being anonymous has allowed many criminals to safely trade money without having to have an official bank account which required ID verification and real name authentication. [1]

Electricity bills: Not to mention how costly the hardware required for mining can be, mining is very demanding on electricity. More electricity required more fuel consumption, consequently leading to more damage to the environment. [1]

No data to trace back: Cryptocurrency has no available information from the previous years that can be used in order to formulate the merest predictions. And with cryptocurrency only joining the financial field in 2009, it is considered hard for investors to rely on history records to ensure the profitability of their investment. [1]

Despite the challenges faced, our team is committed to taking all possible variances into consideration to be able to come up with predictions that give investors a brief idea as to what to do.

Chapter 3 – Proposed Solution

This chapter tackles the proposed solution and a full documentation of the code.

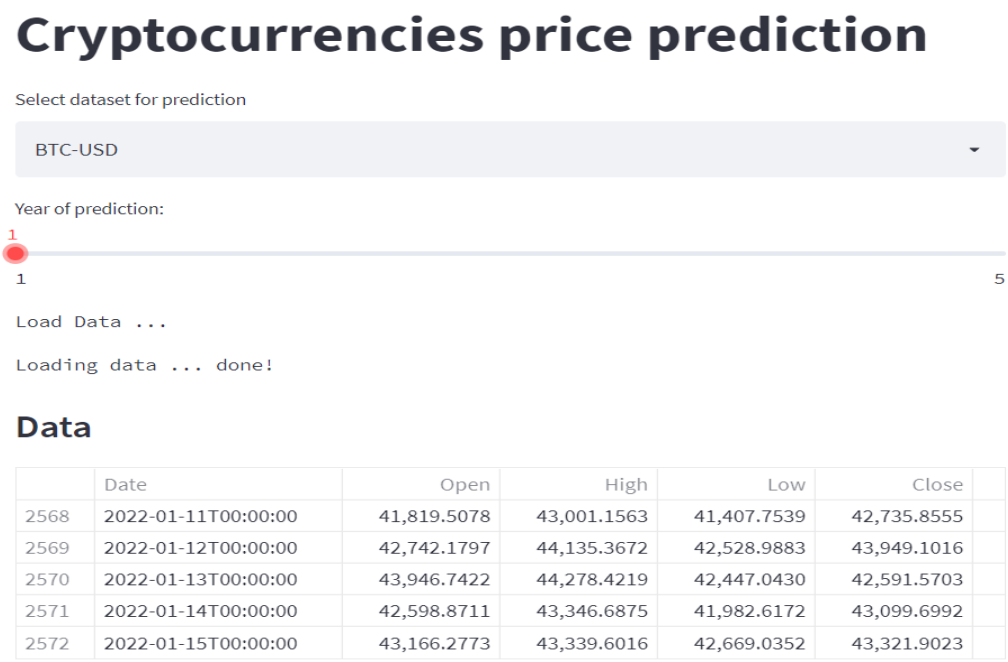
3.1 - Proposed Solution

Data Exploration:

The key advantage of this project is that the prophet library allows prediction of not only 1 coin but multiple alt coins such as BTC-USD","ETH-USD","ADA-USD","XRP-USD","SOL-USD","MATIC-USD","LINK-USD", "FTM-USD", "LTC-USD". This means multiple datasets will be used each time the user chooses a coin to achieve his desired goals of predictions. The import of yfinance package allowed the use of multiple datasets for each of the above mentioned coins. It's good to mention that streamlit library is an open-source library that is heavily used in machine learning and data science project and doesn't require front-end experience where scripts are directly transformed into a web app. So, the interface is a web app where all the logic is being implemented in python language.

First of all, the user will choose from a drop down list the alt coin that he wants to make the time series predictions in using the prophet library. Consequently, the correct dataset will be chosen based on his choice. Moreover, the user is allowed to choose the years of prediction he desires not exceeding 5 years. For example, he can choose to predict 5 years ahead from the current date.

Now, after the user made his decisions and chose the type of coin and the years of prediction. The data will be displayed accordingly and in the below figure a sample run of the explained above:



	Date	Open	High	Low	Close
2568	2022-01-11T00:00:00	41,819.5078	43,001.1563	41,407.7539	42,735.8555
2569	2022-01-12T00:00:00	42,742.1797	44,135.3672	42,528.9883	43,949.1016
2570	2022-01-13T00:00:00	43,946.7422	44,278.4219	42,447.0430	42,591.5703
2571	2022-01-14T00:00:00	42,598.8711	43,346.6875	41,982.6172	43,099.6992
2572	2022-01-15T00:00:00	43,166.2773	43,339.6016	42,669.0352	43,321.9023

In the above figure the user chose bitcoin as an alt coin to start his prediction with and he wants to forecast one year ahead. Accordingly, the data is being loaded successfully and displayed on the screen. The start date is from 2015-01-01 and the current date will be the date the user trying to forecast. So, the last 5 rows were displayed on the screen using the .tail method. As observed above, the dataset contains the opening and closing price of this coin. The low and high represents the highest and lowest prices this coin reached in that day. As well as the volume which refers to the sum of total trades taking place in that day. As for the adjusted close, it's the adjusted closing pricing meaning the closing price after paying dividends and stock splits to shareholders. So, the adjusted closing price will start after the closing price is settled.

Loading the datasets:

Now, to load the dataset a function load_data passing to it a positional argument is created. Then a variable called data is declared and is initialized to the datasets that are found in yfinance package where all the needed alt coins that were used in this project are found in this package. Moreover, the date column is needed for the prophet model, so to make sure it's one of the columns that is listed we did reset our indexes so surely the Date will be now from the columns. The good part about stream lit library is the caching mechanism it possesses. Since manipulation of large datasets is done here and loading of data from web so a number of huge computations is performed. So, to preserve the performance a decorator @st.cahce is used. So the mark of the function load_data with @st.cache tells the streamlit when this load_data function is detected to check for the following:

- The input parameters that you called the function with
- The value of any external variable used in the function
- The body of the function
- The body of any function used inside the cached function. [2]

These 4 components are checked in the exact order and combination by streamlit, which in turns runs the function and will store the results in the cache. But in the second time, when this function is called, streamlit will skip executing the function and, instead, return the output previously stored in the cache. [2]

A code snippet of this function is shown below:

```
@st.cache

def load_data(parg):
    data=yf.download(parg, START,TODAY)
    data.reset_index(inplace=True)
    return data
```

Dataset Columns:

Now checking the columns of the dataset after the implementation of reset_index method:

DataSet Column:

	0
0	Date
1	Open
2	High
3	Low
4	Close
5	Adj Close
6	Volume

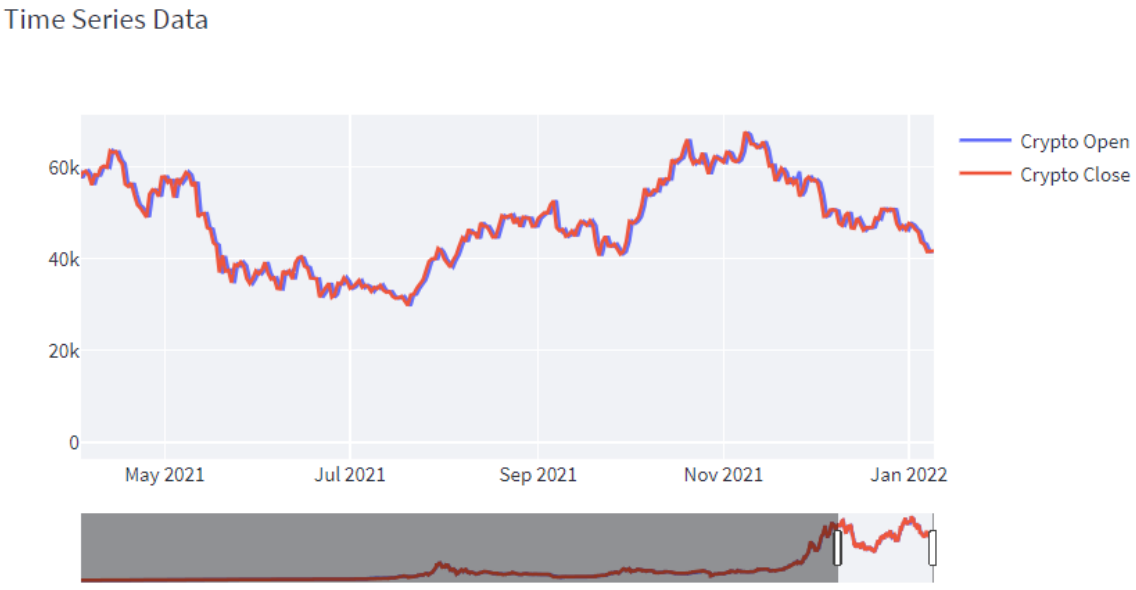
It appears that data column is in the list of indexed columns.

Visulization of the dataset:

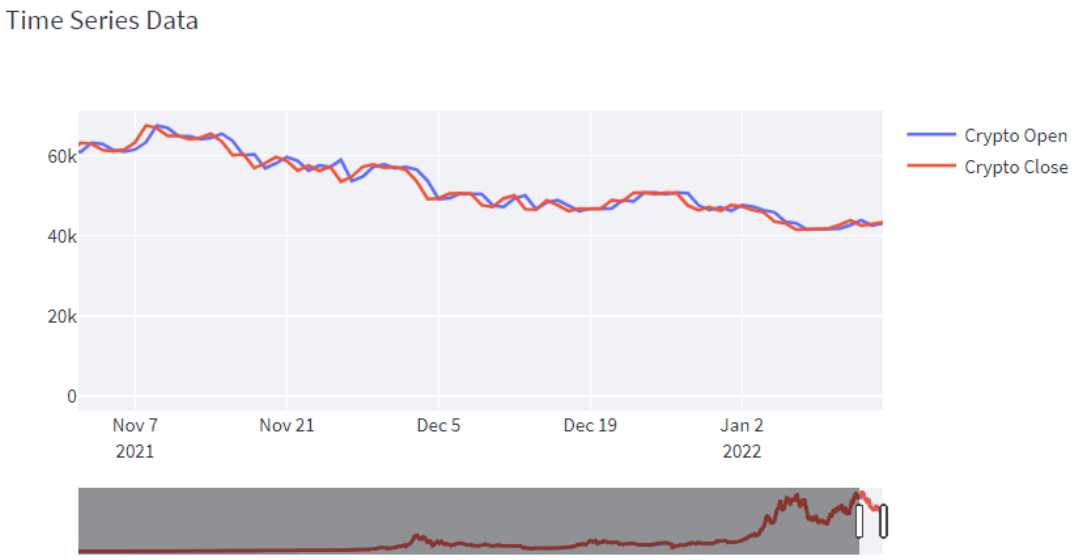
Visualizing the data would be considered a good practice to check and compare between the opening and closing of a certain bitcoin in terms of time. The below figure is plotted using the pyplot library.

A zoomed out version that shows the fluctutations over the years.

A zoomed in version that shows the fluctuations over the months:



And finally a more zoomed in figure over the recent days:



Preparing the Prophet Model:

The prophet has some limitations where it requires only to have 2 columns and in this case: the date and the close columns are the 2 columns needed. So, these 2 columns will be collected and put in a new data frame and the rename function is used to change the name of the columns following the correct name conventions. Again the .tail method is used to check the new data.

	ds	y
2568	2022-01-11T00:00:00	42,735.8555
2569	2022-01-12T00:00:00	43,949.1016
2570	2022-01-13T00:00:00	42,591.5703
2571	2022-01-14T00:00:00	43,099.6992
2572	2022-01-15T00:00:00	43,356.2969

```
df_train=data[['Date','Close']]
df_train=df_train.rename(columns={"Date":"ds","Close":"y"})
st.write(df_train.tail())
```

Prophet Model:

To forecast the values of the time series data in the future the prophet library should be applied now. A new Prophet object called m, a lot of arguments can be specified using prophet but the interval_width is used here that follows the concepts of confidence interval which is a set of estimates for an unknown parameter that's defined as an interval with a bottom and upper bound. The 95% confidence level is most common so 0.95 is used for the interval width. After the Prophet model has been used, fit method with the new data frame as input is being utilized. To get forecasts for this time series, the prophet shall have a new data frame with a ds column containing the dates. Prophet has a handy method called make future data frame which is a helper functions that completes the job. A period is passed which refers to the timestamp where in this case the number of years chosen by the user will be multiplied b 365 days so the make_future_dataframe will generate daily timestamps based on the user choice of prediction. The data frame with future dates is subsequently passed to the fitted model's forecast method.

A code snippet of the code is shown below:

```
m=Prophet(interval_width=0.95)
m.fit(df_train)
future=m.make_future_dataframe(periods=period)
forecast=m.predict(future)
st.write(forecast.tail())
```

The output would be:

	ds	trend	yhat_lower	yhat_upper	trend_lower
2933	2023-01-11T00:00:00	93,278.6589	80,376.4509	104,701.6973	85,055.0547
2934	2023-01-12T00:00:00	93,366.8396	80,700.9390	105,507.8837	85,114.1354
2935	2023-01-13T00:00:00	93,455.0203	81,241.1618	106,195.0080	85,173.2161
2936	2023-01-14T00:00:00	93,543.2009	81,237.2494	105,916.4209	85,232.2968
2937	2023-01-15T00:00:00	93,631.3816	81,623.7863	105,249.6610	85,291.3775

Definition:

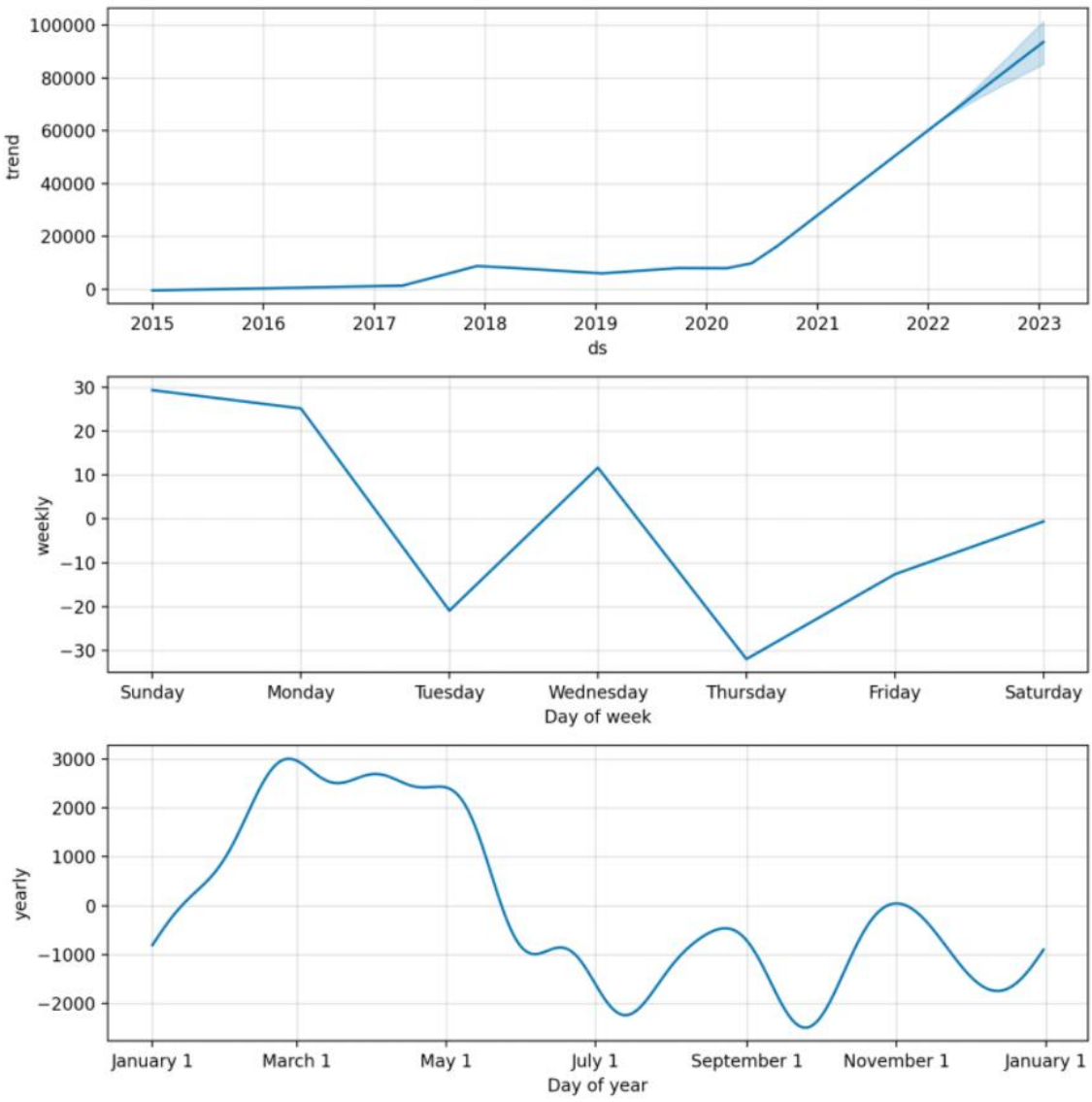
- ds: the datestamp of the forecasted value
- yhat: the forecasted value of the metric and in this case it’s the closing price
- yhat_lower: the lower bound of our forecasts
- yhat_upper: the upper bound of our forecasts
- trend: the forecasted value in terms of trend

Prophets Plot:

Observation:

- The black dots represent the observed value of the time series.
- The blue line represents the forecasted values.
- The uncertainty intervals of the forecasts are the blue shaded region.

Components of the prediction is one of Prophet's capabilities, and it can be used to demonstrate how daily, weekly, and annual patterns of the time series contribute to the overall forecasted values.



Observation:

- The first plot shows that the trend of coin price that simulates non-periodic variations in time series data and how its linearly increasing over time.
- The second plot highlights the fact that the weekly price of the coin peaks towards the end of the week and on Saturday. and go down on Tuesday and Thursday.
- The third plot shows that the Highest price occurs between January and May.

Adding Change Points to prophet:

Change points in data are points in time when the data drastically changes direction. To emphasize more, real time series have huge shifts in trajectories. By default, Prophet provides 25 potential change points that are evenly distributed across the first 80% of the time series. Vertical lines will be used to illustrate the locations of potential change points.

Now, the forecasted values are plotted again but with vertical lines drawn across the axis that illustrates the location of potential change points.

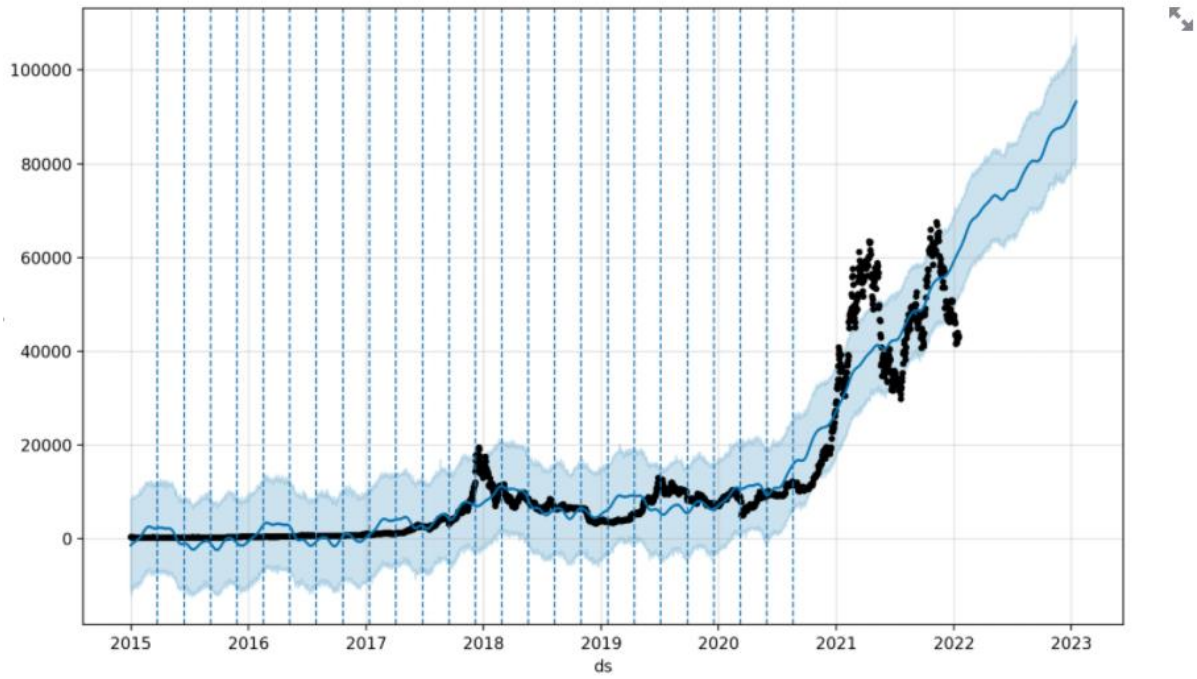
```
figure = m.plot(forecast)
for changepoint in m.changepoints:
    # axvline is used to add a vertical line across the axis
    plt.axvline(changepoint, ls='--', lw=1)
st.pyplot(figure)

# ---------------------------------------------------------

st.write("We can view the dates where the chagepoints occurred.")
st.write(m.changepoints)
```

This code snippet shows the way change points are detected and the way is plotted using the pyplot library.

Now it’s a good practice to show the exact dates as well since that’s not shown in the figure. And the figures are illustrated below:



Exact Dates:

	ds
82	2015-03-23T00:00:00
165	2015-06-14T00:00:00
247	2015-09-04T00:00:00
329	2015-11-25T00:00:00
412	2016-02-16T00:00:00
494	2016-05-08T00:00:00
576	2016-07-29T00:00:00
659	2016-10-20T00:00:00
741	2017-01-10T00:00:00
823	2017-04-02T00:00:00

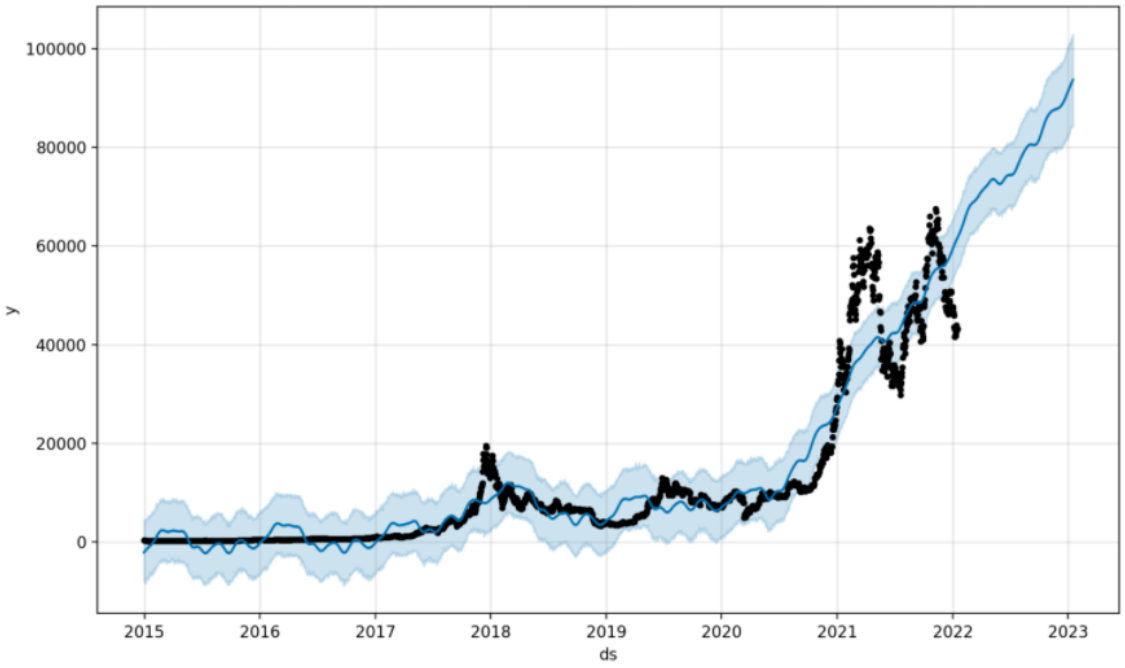
	ds
1317	2018-08-09T00:00:00
1399	2018-10-30T00:00:00
1482	2019-01-21T00:00:00
1564	2019-04-13T00:00:00
1646	2019-07-04T00:00:00
1729	2019-09-25T00:00:00
1811	2019-12-16T00:00:00
1893	2020-03-07T00:00:00
1976	2020-05-29T00:00:00
2058	2020-08-19T00:00:00

Adjusting Trends:

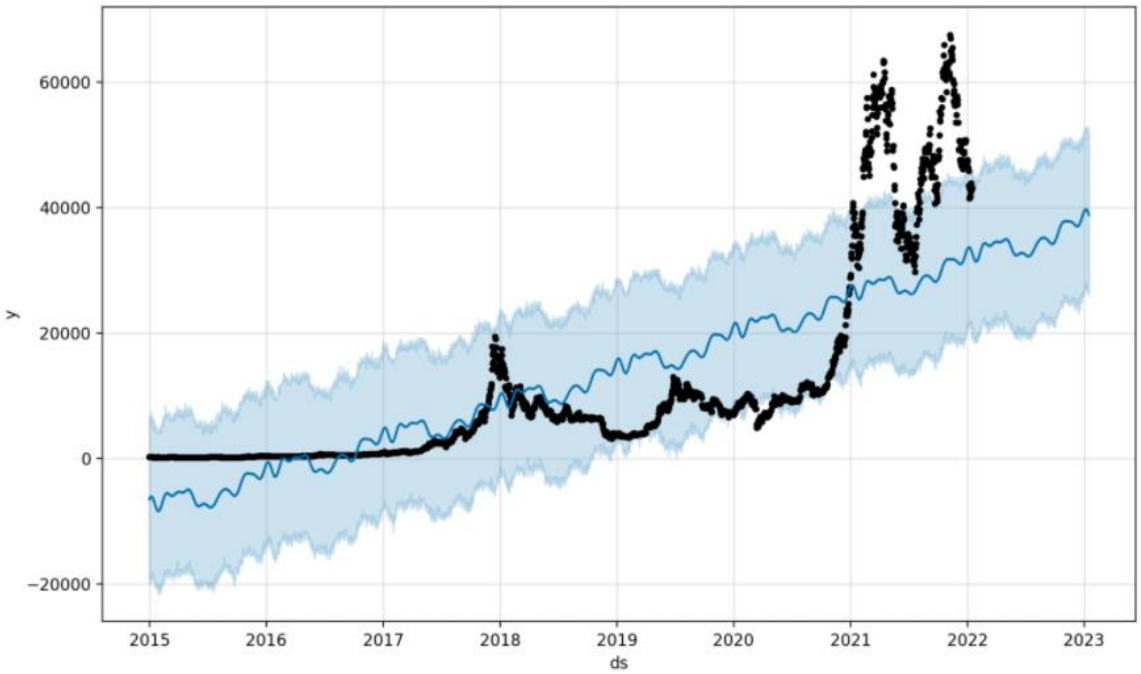
Trend adjustment helps prevent under fitting and overfitting, it allows for the adjustment of the strength of the trend. Changepoint_prior_scale is an input argument that is set to 0.05 by default, and increasing it would increase the flexibility. In this project it's been set to 0.08 for increased flexibility. Trend adjustment increases the accuracy of the actual yield in retrospect to the historical yield. If the flexibility is higher than needed, then trend changes might over fit and the

opposite is true where if the flexibility is low then trends changes might under fit. Below is a comparison between a high flexibility and a low one:

High Flexibility:



Low Flexibility:



Chapter 4 – Conclusion

4.1 – Conclusion

Cryptocurrency has solidified itself in the field of finance, it has proven that it is here to stay and only go up from here, for the future of finance profoundly relies on contemporary technologies. Taking into consideration interest rates, electric bills, and other variants, it is not an easy task to start investing in cryptocurrency and make profit out of it let alone a living. Overall, the multitude of factors that impact the prices of coins does not allow for their accurate prediction, however with prophet, we are able to come one step closer to safe decision making when it comes to selling and buying coins.

4.2 - Percentage of Completion

100% completion of the code. No errors occurred.

Source Code

```
from fbprophet.plot import add_changepoints_to_plot
import streamlit as st
import datetime as date
import yfinance as yf
from fbprophet import Prophet
from fbprophet.plot import plot_plotly
from plotly import graph_objects as go
import matplotlib.pyplot as plt



START = "2015-01-01"
TODAY = date.datetime.now()

st.title(" Cryptocurrencies price prediction")
crypto = ("BTC-USD", "ETH-USD", "ADA-USD", "XRP-USD", "SOL-USD",
          "MATIC-USD", "LINK-USD", "FTM-USD", "LTC-USD")
selected_crypto = st.selectbox("Select dataset for prediction", crypto)
n_year = st.slider("Year of prediction:", 1, 5)
period = n_year*365

# ------------------------------------------------------



@st.cache
def load_data(parg):
```

```
    data = yf.download(parg, START, TODAY)
    data.reset_index(inplace=True)
    return data


data_load_state = st.text("Load Data ...")
data = load_data(selected_crypto)
data_load_state = st.text("Loading data ... done!")

# ------------------------------------------------------

st.subheader("Data")
st.write(data.tail(5))

# ------------------------------------------------------

st.subheader("Exploring the Dataset")
st.write("Dataset Columns:")
st.write(data.columns)

# ------------------------------------------------------

st.subheader("Data Visualization")
st.write("It is considered a good practice to visualize the data at hand. So
let’s plot our time series data:")


def plot_raw_data():
    fig = go.Figure()
```

```
    fig.add_trace(go.Scatter(x=data["Date"],
                  y=data['Open'], name="Crypto Open"))
    fig.add_trace(go.Scatter(x=data["Date"],
                  y=data['Close'], name="Crypto Close"))


    fig.layout.update(title="Time Series Data",
xaxis_rangeslider_visible=True)
    st.plotly_chart(fig, use_container_width=True)



plot_raw_data()


# ----------------------------------------------------


st.subheader("Data Preparation")

st.write("The prophet has some limitations where it requires only to have 2
columns and in this case: the date and the close columns are the 2 columns
needed.")

st.write("So, we must rename the columns in our dataframe. The rename
function is used to change the name of the columns following the correct name
conventions")

st.write("The .tail method is used to check the new data.")

df_train = data[['Date', 'Close']]

df_train = df_train.rename(columns={"Date": "ds", "Close": "y"})

st.write(df_train.tail())


# ----------------------------------------------------


st.subheader('Prophet Model')

st.write("The Prophet model has been used here.")
```

```
st.write("A fit method with the new data frame as input is being utilized.")

st.write("After that, The make_future_dataframe which is a handy method in
prophet will generate daily timestamps based on the user choice of
prediction")

st.write("The data frame with future dates is subsequently passed to the
fitted model's forecast method.")

st.write("Again, the .tail method is used to check the new data.")


m = Prophet(interval_width=0.95)

m.fit(df_train)

future = m.make_future_dataframe(periods=period)

forecast = m.predict(future)

st.write(forecast.tail())


# ---------------------------------------------------------



st.subheader("Visulaization of the Prophet Model")

forecast_fig = plot_plotly(m, forecast)

st.plotly_chart(forecast_fig, use_container_width=True, uncertainty=True)

st.write("• The black dots represent the observed value of the time series.")

st.write("• The blue line represents the forecasted values.")

st.write("• The uncertainty intervals of the forecasts are the blue shaded
region")



# ---------------------------------------------------------



st.subheader("Plotting the Forecasted components")
```

```
st.write("Components of the prediction is one of Prophet's capabilities, and
it can be used to demonstrate how daily, weekly, and annual patterns of the
time series contribute to the overall forecasted values. ")

fig2 = m.plot_components(forecast)

st.write(fig2)



st.write("Wow!!!! The above plot provides interesting insights.")

st.write("•The first plot shows that the trend of coin prices that simulates
non-periodic variations in time series data and how its linearly increasing
over time.")

st.write("•The second plot highlights the fact that the weekly price of the
coin  peaks towards the end of the week and on Saturday. and go down on
Tuesday and Thursday")

st.write("•The third plot shows that the Highest price occurs between
January and May")




# -----------------------------------------------------


st.subheader("Adding ChangePoints to Prophet")



st.write("Changepoints are the datetime points where the time series have
abrupt changes in the trajectory.")

st.write("By default, Prophet adds 25 changepoints to the initial 80% of the
data-set.")

st.write("Let’s plot the vertical lines where the potential changepoints
occurred")



# another way of plotting

#fig = m.plot(forecast)

#a = add_changepoints_to_plot(fig.gca(), m, forecast)

# st.pyplot(fig)
```

```
figure = m.plot(forecast)
for changepoint in m.changepoints:
    # axvline is used to add a vertical line across the axis
    plt.axvline(changepoint, ls='--', lw=1)
st.pyplot(figure)

# ---------------------------------------------------------

st.write("We can view the dates where the chagepoints occurred.")
st.write(m.changepoints)

# ---------------------------------------------------------

st.subheader("Adjusting Trend ")
st.write("Prophet allows us to adjust the trend in case there is an overfit
or underfit.")
st.write("changepoint_prior_scale helps adjust the strength of the trend.")

# another way of visualizing the trend flexibilty
#pro_change = Prophet(yearly_seasonality=True, changepoint_prior_scale=0.08)
#forecast = pro_change.fit(df_train).predict(future)
#fig = pro_change.plot(forecast)
#a = add_changepoints_to_plot(fig.gca(), pro_change, forecast)
# st.pyplot(fig)

# Increasing Changepoint_prior_scale  will make the trend more flexible:
st.subheader("Higher Flexibilty:")
m = Prophet(changepoint_prior_scale=0.08)
```

```
forecast = m.fit(df_train).predict(future)

fig = m.plot(forecast)

st.pyplot(fig)

st.subheader("Lower Flexibilty:")

# Decreasing Changepoint_prior_scale  will make the trend less flexible:

n = Prophet(changepoint_prior_scale=0.001)

forecastn = n.fit(df_train).predict(future)

fign = n.plot(forecastn)

st.pyplot(fign)


# ----------------------------------------------------------
```

REFERENCES

[1]M. A. FAUZI, N. PAIMAN, and Z. OTHMAN, “Bitcoin and Cryptocurrency: Challenges, Opportunities and Future Works,” The Journal of Asian Finance, Economics and Business, vol. 7, no. 8, pp. 695–704, Aug. 2020.

[2] "Optimize performance with st.cache - Streamlit Docs", Docs.streamlit.io, 2022. [Online]. Available: https://docs.streamlit.io/library/advanced-features/caching.