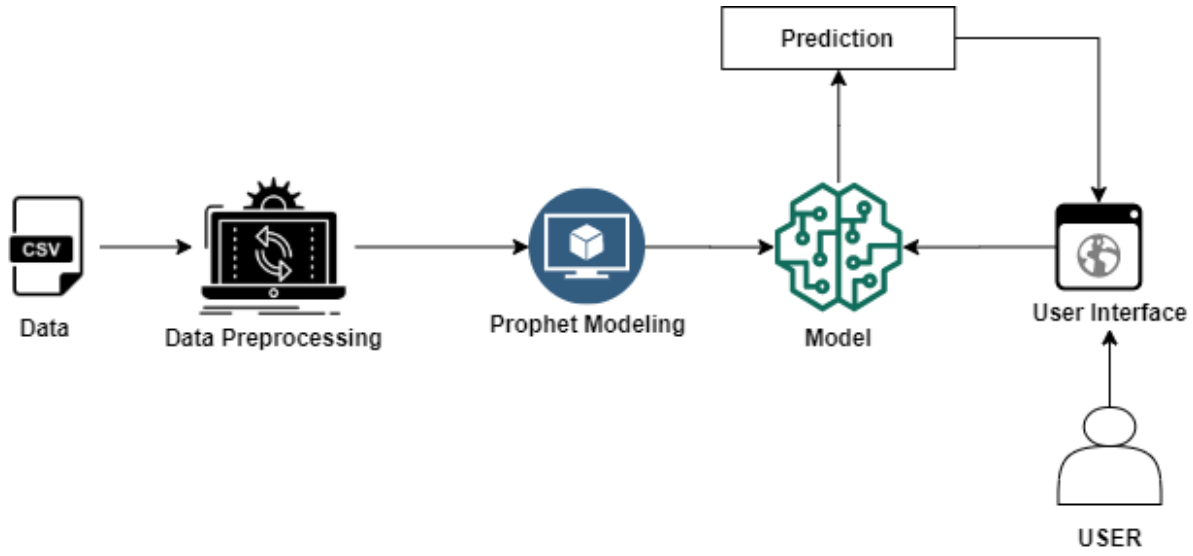
Crypto Price Prediction using FbProphet

Project Idea:

Bitcoin is a cryptocurrency that was created in January 2009. It is the world's most valuable crypto+currency and is traded on over 40 exchanges around the world, accepting over 30 different currencies. As a currency, Bitcoin offers a new opportunity for price forecasting as it has high volatility, which is much higher compared to traditional currencies. The price of Bitcoin in January 2017 was 1,000 USD and by the end of December 2017, its value went up to 16000 USD and its value as of June 2022 is 25711 USD. We can say that the crypto market is very volatile, and among all the cryptocurrencies in the market, Bitcoin is experienced by most investors due to its anonymity and transparency in the system.

This project aims to work on the prediction system for Bitcoin using FbProphet to predict the price. There are various factors affecting the price of Bitcoin. A FbProphet model is built that helps to define the price trend of Bitcoin in the future.

Architecture:



Learning Outcomes:

By the end of this project:

- You'll be able to know the fundamental concepts of time series forecasting.
- You will be able to analyze or get insights into data through visualization.
- You will be able to know how to build a web application using the Flask framework.

Project Flow:

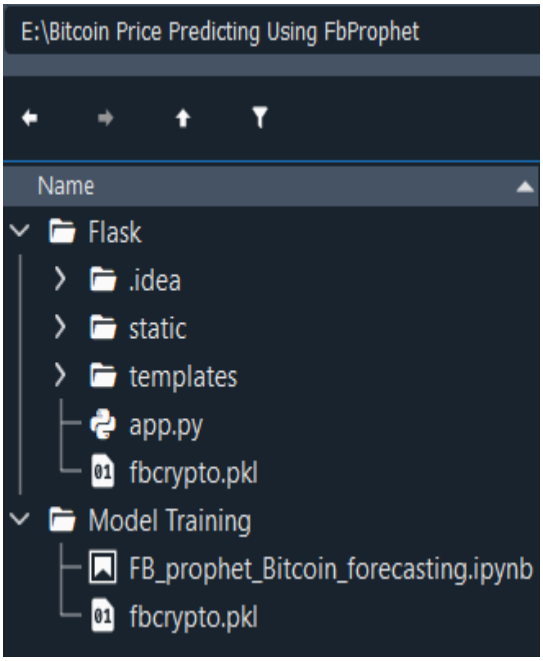
- The user interacts with the UI (User Interface) to select the date as input.
- Selected Date input values are analyzed by the model which is integrated
- Once the model is analyzed the input prediction is showcased on the UI

To accomplish this, complete all the milestones & activities listed below.

- Installation of Pre-requisites.
    - Installation of Anaconda IDE / Anaconda Navigator.
    - Installation of Python packages.
- Data Collection.
    - Create or Collect the dataset.
- Data Pre-processing.
    - Importing of Libraries.
    - Importing of Dataset.
    - Analyze the data.
    - Handling Missing Values, reset the index & renaming the column.
    - Visualizing the Time Series
- Model Building.
    - Fitting the prophet library.
    - Making Future Predictions
    - Evaluation of the model.
    - Save the model.
- Application Building
    - Create an HTML file
    - Build a Python Code

Project Structure:

Create a Project folder that contains files as shown below



- All the above files will be used to develop a flask application.
- The static folder contains a CSS file.
- We are building a Flask Application that needs HTML pages stored in the templates folder and a python script app.py for serverside scripting
- The Model training folder contains the training file FB_prophet_Bitcoin_forecasting.ipynb
- The fbcrypto.pkl is the saved model file

Milestone 1: Installation of Pre-requisites

To complete the project successfully, you need to install the following software & packages:

Activity 1: Install Anaconda IDE / Anaconda Navigator.

- In order to develop a solution to this problem statement, we need an environment to write and test the code.
- We use Anaconda IDE (Integrated Developing Environment).
- Refer to the below link to download & install Anaconda Navigator.

Link: https://www.youtube.com/watch?v=5mDYijMfSzs

Activity 2: Installation of Python Packages

- Follow the below steps to install the required libraries.

    - Open Anaconda Navigator as administrator.
    - Type “pip install pystan==2.19.1.1” and press enter.
    - Type “pip install yfinance” and press enter.
    - Type “conda install -y fbprophet -c conda-forge ” and press enter.

The above steps allow to install the required packages.

With this, we are done with the completion of milestone 1.

- To know more about the prophet library go through the reference links.

Link 1: https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/

Link2:https://towardsdatascience.com/a-quick-start-of-time-series-forecasting-with-a-practical-example-using-fb-prophet-31c4447a2274

Milestone 2: Data Collection

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc.

Activity1: Download The dataset

The dataset used for this project was obtained from Yahoo Finance. Please refer to the link given below to download the data set for the last 5 years.

Dataset Link: BTC-USD

Milestone 3: Data Pre-processing

Data preprocessing is a data mining technique that is used to transform the raw data into a useful and efficient format. So we need to clean the dataset properly in order to fetch good results.

Activity 1: Import Libraries

Import the necessary libraries for data pre-processing, forecasting using FbProphet, etc.

- It is important to import all the necessary libraries such as pandas,plotly, yahoo finance & Fbprophet.
- **Pandas**- It is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language.

- **Plotly-** The Plotly Python library is an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases. Built on top of the Plotly JavaScript library.
- **Yahoo Finance-** Download Market data from the yfinance module.

```
import pandas as pd
import yfinance as yf
from datetime import datetime
from datetime import timedelta
import plotly.graph_objects as go
from fbprophet import Prophet
from fbprophet.plot import plot_plotly, plot_components_plotly
import warnings

warnings.filterwarnings('ignore')

pd.options.display.float_format = '${:,.2f}'.format
```

Activity 2: Import Dataset

Download the real-time data from the Yahoo Finance library where we need to pass three parameters in the yahoo finance download function i.e. abbreviation name of the cryptocurrency, start date, and today date then we stored it into a variable called df.

```
today = datetime.today().strftime('%Y-%m-%d')
start_date = '2016-01-01'

df = yf.download('BTC-USD',start_date, today)
```

```
[**********100%**********]  1 of 1 completed
```

Check the entire dataset.

```
df
```

Date	Open	High	Low	Close	Adj Close	Volume
2016-01-01	$430.72	$436.25	$427.52	$434.33	$434.33	36278900
2016-01-02	$434.62	$436.06	$431.87	$433.44	$433.44	30096600
2016-01-03	$433.58	$433.74	$424.71	$430.01	$430.01	39633800
2016-01-04	$430.06	$434.52	$429.08	$433.09	$433.09	38477500
2016-01-05	$433.07	$434.18	$429.68	$431.96	$431.96	34522600
...	...	...	...	...	...	...
2022-06-09	$30,215.28	$30,609.31	$30,020.27	$30,112.00	$30,112.00	21692004719
2022-06-10	$30,110.33	$30,245.81	$28,978.15	$29,083.80	$29,083.80	29867476527
2022-06-11	$29,084.67	$29,401.92	$28,236.21	$28,360.81	$28,360.81	27246574439
2022-06-12	$28,373.51	$28,502.69	$26,762.65	$26,762.65	$26,762.65	34163220274
2022-06-13	$26,645.55	$26,787.33	$25,137.23	$25,423.14	$25,423.14	46797164544

2356 rows × 6 columns

- Bitcoin Dataset contains the following Columns

```
df.columns
```

```
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

1. Date:- Datewise Information related to the quote currency.
2. Open:- The opening price of the time interval in the quote currency (For BTC/USD, the price would be USD).
3. High: Highest price reached during the time interval, in the quote currency.
4. Low: Lowest price reached during the time interval, in the quote currency.
5. Close:- The closing price of the time interval, in the quote currency.
6. Adj Close:- Final prices of the time interval, in the quote currency.
7. Volume: Quantity of assets bought or sold, displayed in base currency.

Activity 3: Analyse the data

- the head() method is used to return the top n (5 by default) rows of a Data.

```
df.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2016-01-01	$430.72	$436.25	$427.52	$434.33	$434.33	36278900
2016-01-02	$434.62	$436.06	$431.87	$433.44	$433.44	30096600
2016-01-03	$433.58	$433.74	$424.71	$430.01	$430.01	39633800
2016-01-04	$430.06	$434.52	$429.08	$433.09	$433.09	38477500
2016-01-05	$433.07	$434.18	$429.68	$431.96	$431.96	34522600

- List the Last five-row of the dataset using the tail function.

```
df.tail()
```

Date	Open	High	Low	Close	Adj Close	Volume
2022-06-09	$30,215.28	$30,609.31	$30,020.27	$30,112.00	$30,112.00	21692004719
2022-06-10	$30,110.33	$30,245.81	$28,978.15	$29,083.80	$29,083.80	29867476527
2022-06-11	$29,084.67	$29,401.92	$28,236.21	$28,360.81	$28,360.81	27246574439
2022-06-12	$28,373.51	$28,502.69	$26,762.65	$26,762.65	$26,762.65	34163220274
2022-06-13	$26,645.55	$26,787.33	$25,137.23	$25,280.53	$25,280.53	45642170368

- describe() method computes a summary of statistics like count, mean, standard deviation, min, max, and quartile values.

```
df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	$2,356.00	$2,356.00	$2,356.00	$2,356.00	$2,356.00	$2,356.00
mean	$14,779.77	$15,164.79	$14,350.14	$14,788.36	$14,788.36	$18,367,261,110.34
std	$17,237.82	$17,682.56	$16,721.16	$17,232.59	$17,232.59	$20,508,300,401.19
min	$365.07	$374.95	$354.91	$364.33	$364.33	$28,514,000.00
25%	$3,298.22	$3,426.65	$3,227.00	$3,335.27	$3,335.27	$1,530,309,952.00
50%	$7,989.25	$8,190.30	$7,784.55	$7,987.76	$7,987.76	$13,150,143,267.00
75%	$16,334.51	$17,175.61	$15,588.38	$16,425.55	$16,425.55	$30,221,720,189.25
max	$67,549.73	$68,789.62	$66,382.06	$67,566.83	$67,566.83	$350,967,941,479.00

- info() method prints information about the DataFrame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2337 entries, 2016-01-01 to 2022-05-25
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       2337 non-null   float64
 1   High       2337 non-null   float64
 2   Low        2337 non-null   float64
 3   Close      2337 non-null   float64
 4   Adj Close  2337 non-null   float64
 5   Volume     2337 non-null   int64  
dtypes: float64(5), int64(1)
memory usage: 127.8 KB
```

Activity 4: Handling Missing Values, reset the index & renaming the column.

1. After loading it is important to check the complete information of data as it can indicate many of the hidden information such as null values in a column or a row

2. Check whether any null values are there or not. if it is present then the following can be done,

a.Imputing data using the Imputation method in sklearn

b.Filling NaN values with mean, median, and mode using fillna() method.

3. isnull()- Generate boolean mask indicating missing values.

4. We don't have any missing values present in our dataframe.

```
df.isnull().any()
```

```
Open         False
High         False
Low          False
Close        False
Adj Close    False
Volume       False
dtype: bool
```

5. Check the total no of missing values presented in the dataset

```
df.isnull().sum()
```

```
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

Now use the reset_index() function to generate a new DataFrame or Series with the index reset and it will add a date as a column.

```
df.reset_index(inplace=True)
df.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

Now check the first five rows of data using the head function.

```
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2016-01-01	$430.72	$436.25	$427.52	$434.33	$434.33	36278900
1	2016-01-02	$434.62	$436.06	$431.87	$433.44	$433.44	30096600
2	2016-01-03	$433.58	$433.74	$424.71	$430.01	$430.01	39633800
3	2016-01-04	$430.06	$434.52	$429.08	$433.09	$433.09	38477500
4	2016-01-05	$433.07	$434.18	$429.68	$431.96	$431.96	34522600

Create a new dataframe with the Date and Open column and store it into df1 variable then check the top 5 rows of data using the head function.

```
df1 = df[["Date", "Open"]]
```

```
df1.head()
```

	Date	Open
0	2016-01-01	$430.72
1	2016-01-02	$434.62
2	2016-01-03	$433.58
3	2016-01-04	$430.06
4	2016-01-05	$433.07

- Renaming all the column names accordingly to the prophet library integration for building the model.

```
new_names = {
    "Date": "ds",
    "Open": "y",
}

df1.rename(columns=new_names, inplace=True)
```

- List the first five rows of the dataset after the changes.

```
df1.head()
```

	ds	y
0	2016-01-01	$430.72
1	2016-01-02	$434.62
2	2016-01-03	$433.58
3	2016-01-04	$430.06
4	2016-01-05	$433.07

Activity 5: Visualize Time Series Plot

- Now, let's visualize the data using the Plotly library for Time Series plot of Bitcoin Open Price.

```python
x = df1["ds"]
y = df1["y"]

fig = go.Figure()

fig.add_trace(go.Scatter(x=x, y=y))

# Set title
fig.update_layout(
    title_text="Time series plot of Bitcoin Open Price",
)

fig.update_layout(
    xaxis=dict(
        rangeselector=dict(
            buttons=list(
                [
                    dict(count=1, label="1m", step="month", stepmode="backward"),
                    dict(count=6, label="6m", step="month", stepmode="backward"),
                    dict(count=1, label="YTD", step="year", stepmode="todate"),
                    dict(count=1, label="1y", step="year", stepmode="backward"),
                    dict(step="all"),
                ]
            )
        ),
        rangeslider=dict(visible=True),
        type="date",
    )
)
```

Milestone 4: Model Building

In this milestone, you will build the model using the prophet library.

Activity 1: Fitting the prophet library

- Create the instance of the prophet and fit it to the dataset.

By default Prophet fits additive seasonalities, meaning the effect of the seasonality is added to the trend to get the forecast. This time series of the price of Bitcoin where additive seasonality does not work. This time series has a clear yearly cycle, but the seasonality in the forecast is too large at the start of the time series and too small at the end. In this time series, the seasonality is not a constant additive factor as assumed by Prophet, rather it grows with the trend. This is multiplicative seasonality.

the prophet can model multiplicative seasonality by setting seasonality_mode='multiplicative' in the input arguments:

```
m = Prophet(
    seasonality_mode="multiplicative"
)

m.fit(df1)
```

```
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
<fbprophet.forecaster.Prophet at 0x7f7b86e76610>
```

Note: It will take a few minutes to fit the model.

Activity 2:Making Future Predictions

The next step is to prepare our model to make future predictions. This is achieved using the Prophet.make_future_dataframe method and passing the number of days we'd like to predict in the future. We use the periods attribute to specify this. This also includes the historical dates. We'll use these historical dates to compare the predictions with the actual values in the ds column.

```
future = m.make_future_dataframe(periods = 365)
future.tail()
```

	ds
2697	2023-05-21
2698	2023-05-22
2699	2023-05-23
2700	2023-05-24
2701	2023-05-25

Activity 3:Evaluate the model

We use the predict method to make future predictions. This will generate a dataframe with an **yhat** column that will contain the predictions.

If we check the head for our forecast dataframe we'll notice that it has very many columns. However, we are mainly interested in **ds, yhat, yhat_lower and yhat_upper**. **yhat** is our predicted forecast, **yhat_lower** is the lower bound for our predictions and **yhat_upper** is the upper bound for our predictions.

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

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	multiplicative_terms	multiplicative_terms_lower	multiplicative_terms_upper	weekly	weekly_lower
0	2016-01-01	$298.00	$-1,894.97	$2,870.90	$298.00	$298.00	$0.14	$0.14	$0.14	$-0.00	$-0.00
1	2016-01-02	$299.29	$-2,024.60	$2,659.02	$299.29	$299.29	$0.14	$0.14	$0.14	$-0.00	$-0.00
2	2016-01-03	$300.59	$-2,167.52	$2,765.47	$300.59	$300.59	$0.14	$0.14	$0.14	$-0.00	$-0.00
3	2016-01-04	$301.88	$-2,106.58	$2,783.96	$301.88	$301.88	$0.13	$0.13	$0.13	$-0.00	$-0.00
4	2016-01-05	$303.17	$-2,205.14	$2,831.11	$303.17	$303.17	$0.13	$0.13	$0.13	$0.00	$0.00
...	...	...	...	...	...	...	...	...	...	...	...
2716	2023-	$24,898.24	$-14,095.69	$43,876.75	$17,876.44	$59,156.53	$0.26	$0.26	$0.26	$-0.00	$-0.00

- Get the summary of the forecast.

```
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
2716	2023-06-09	$18,403.13	$-14,095.69	$43,876.75
2717	2023-06-10	$18,437.81	$-13,977.51	$44,551.99
2718	2023-06-11	$18,483.06	$-13,684.28	$44,031.22
2719	2023-06-12	$18,506.96	$-13,657.00	$44,534.01
2720	2023-06-13	$18,551.37	$-13,826.56	$44,260.30

- For predicting the next day's price we calculate the DateTime and stored it into the next_day variable then predict that value.

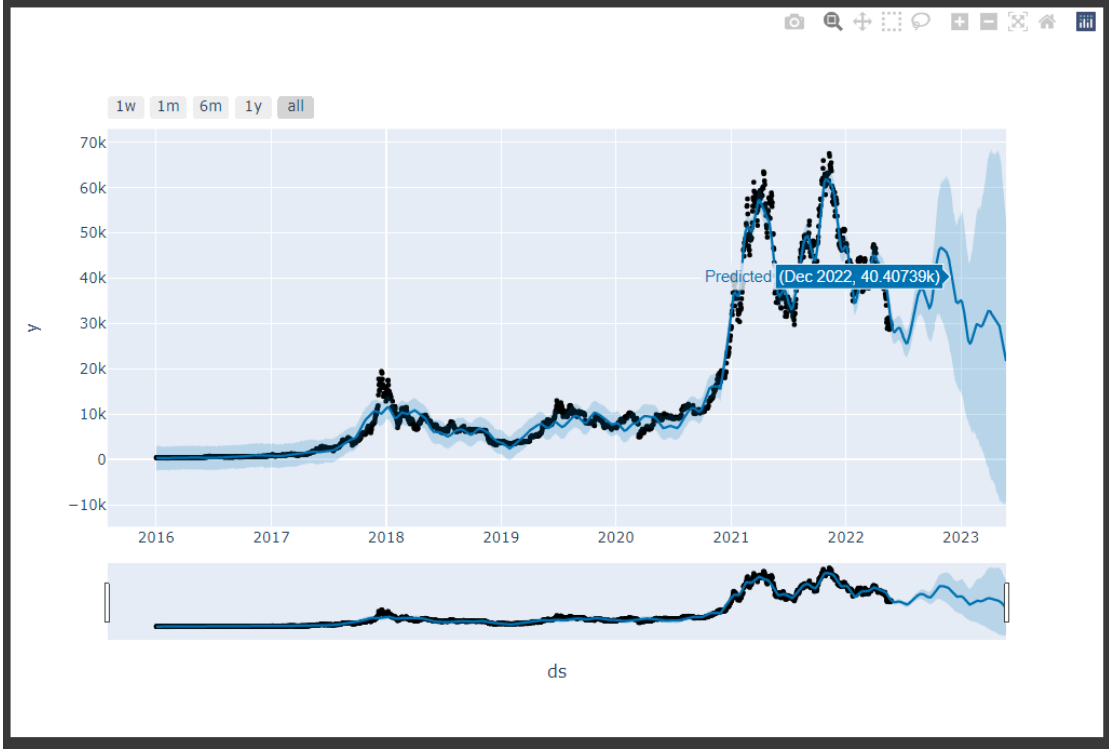
```
next_day = (datetime.today() + timedelta(days=1)).strftime('%Y-%m-%d')

forecast[forecast['ds'] == next_day]['yhat'].item()
```

```
28224.110755749032
```

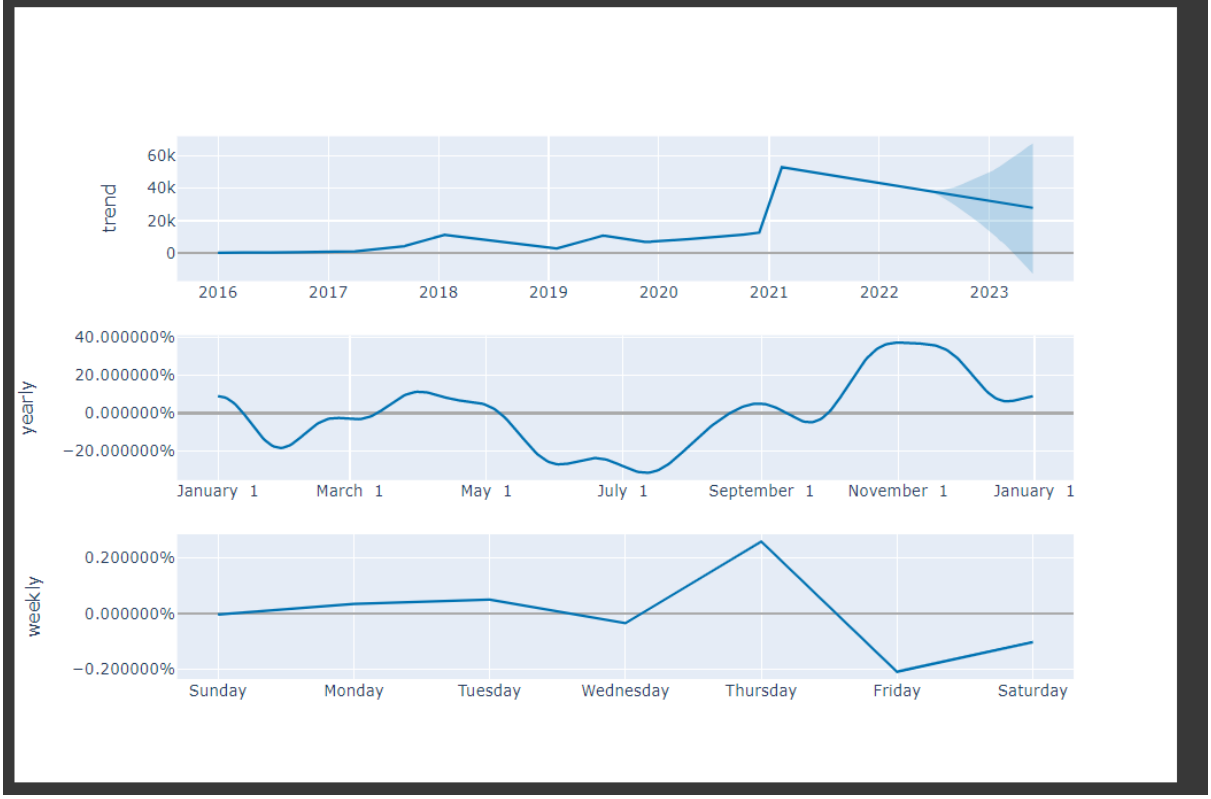
- Now, let's visualize the forecast value of the Bitcoin price till the next year.

```
plot_plotly(m, forecast)
```



- Visualize the components using plot_components_plotly which showcases the trend of the bitcoin price, Yearly growth in percentage, and weekly growth in percentage.

```
plot_components_plotly(m, forecast)
```



Activity 4: Save the model.

This is the final activity of this milestone, here you will be saving the model to integrate to the web application.

- Follow the commands to save your model.

```
import pickle
pickle.dump(m,open('fbcrypto.pkl','wb'))
```

Milestone 5: Application Building

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.

In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to predict the price of bitcoin on a selected date and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the "predict" button, the next page is opened where the user selects the date and predicts the output.

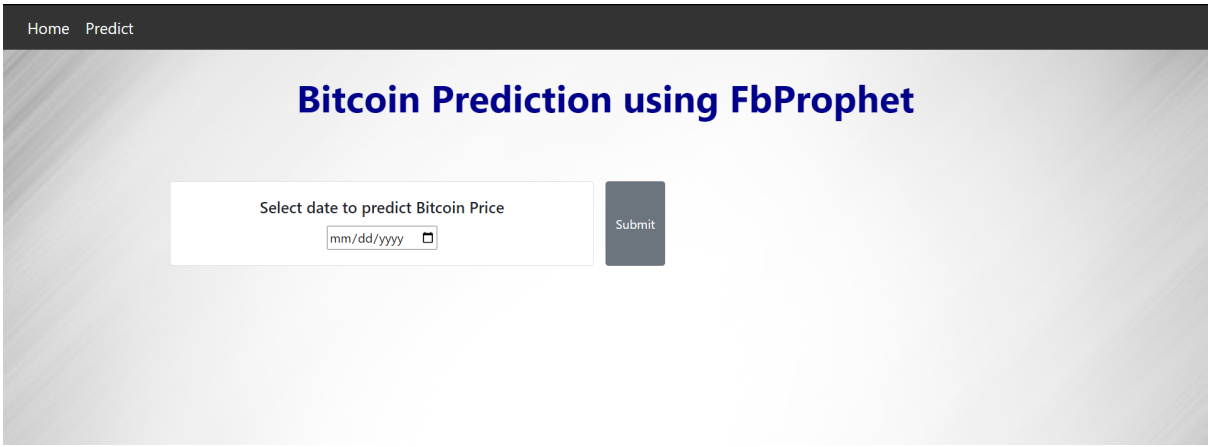
Activity 1 : Create HTML Pages

- We use HTML to create the front-end part of the web page.

- Here, we have created 2 HTML pages- predict.html & index.html.
- index.html displays the home page for an introduction to the project
- predict.html gives the prediction of bitcoin based on the selection date.
- For more information regarding HTML & CSS
  Reference Links : HTML & CSS

index.html looks like this

predict.html looks like this



Activity 2: Build python code

Task 1: Importing Libraries

The first step is usually importing the libraries that will be needed in the program.

```
import numpy as np
import pandas as pd
from flask import Flask, request, jsonify, render_template
import pickle
```

Importing the flask module into the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as an argument Pickle library to load the model file.

Task 2: Creating our flask application and loading our model by using pickle.load() method

```
#flask app
app = Flask(__name__)
#loading the saved model
m = pickle.load(open('fbcrypto.pkl', 'rb'))
```

Task 3: Routing to the HTML Pages

Here, the declared constructor is used to route to the HTML page created earlier.

```
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')#rendering html page

@app.route('/Bitcoin',methods=['POST','GET'])
def prediction(): # route which will take you to the prediction page
    return render_template('predict.html')
```

In the above code snippet, the '/' URL is bound with the index.html. Hence, when the home page of a web server is opened in the browser, the HTML page (index.html) will be rendered.

Task 4:Making Future Prediction

This step is to prepare our model to make future predictions. This is achieved using the Prophet.make_future_dataframe method and passing the number of days we'd like to predict in the future and storing the data into a forecast variable.

```
future = m.make_future_dataframe(periods = 365)
forecast = m.predict(future)
print(forecast)
```

Showcasing prediction on UI:

Firstly, we are rendering the index.html template and from there we are navigating to our prediction page that is predict.html. We select the date (year, month & day) these values are sent to the loaded model, and the resultant output is displayed on predict.html.

```
@app.route('/predict',methods=['POST'])
def y_predict():
    if request.method == "POST":
        ds = request.form["Date"]
        print(ds)
        ds=str(ds)
        print(ds)
        next_day=ds
        print(next_day)
        prediction=forecast[forecast['ds'] == next_day]['yhat'].item()
        prediction=round(prediction,2)
        print(prediction)
        return render_template('predict.html',prediction_text="Bitcoin Price on selected date is $ {} Cents".format(prediction))
    return render_template("predict.html")
```

Finally, Run the application

This is used to run the application on localhost.

```
if __name__ == "__main__":
    app.run(debug=False)
```

Activity 3: Running of flask Application

- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type the “python app.py” command.
- It will show the local host where your app is running on **http://127.0.0.1.5000/**
- Copy that local host URL and open that URL in the browser. It does navigate you to where you can view your web page.

```
(base) C:\Users\USER>E:

(base) E:\>cd E:\Bitcoin Price Predicting Using FbProphet\Flask

(base) E:\Bitcoin Price Predicting Using FbProphet\Flask>python app.py
```

Then it will run on localhost:5000

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

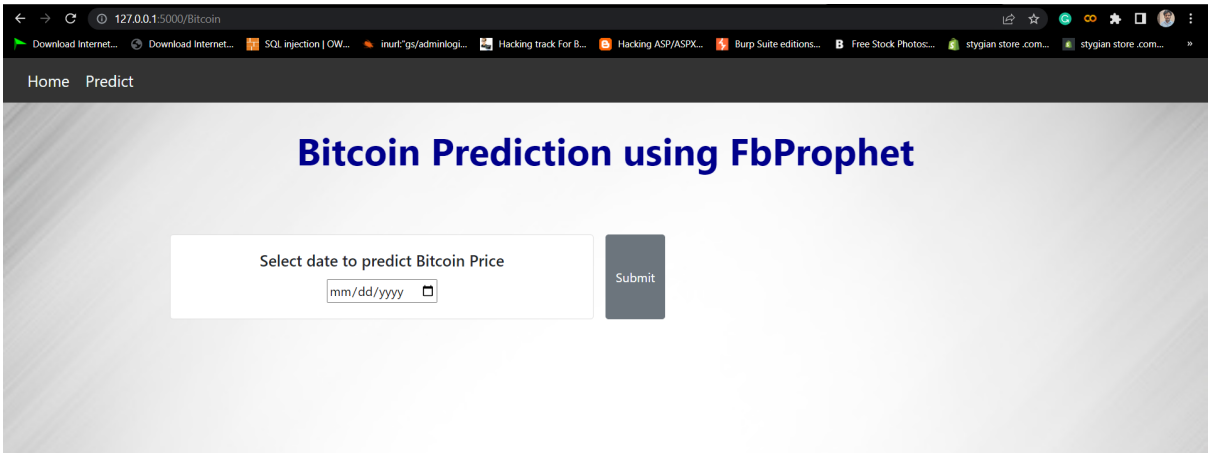
Navigate to the localhost (http://127.0.0.1:5000/) where you can view your web page.

Predicting the results:-

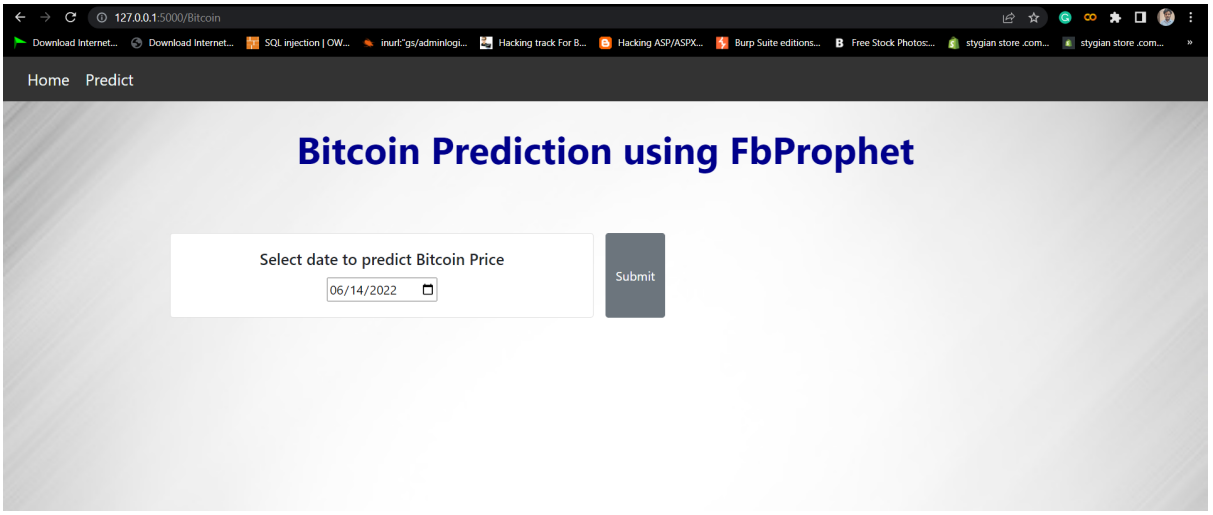
The home page is displayed when we write the localhost URL to the browser.

If we select to predict from the navigation bar, we will be redirected to the prediction page.

- Select the date, click on the submit button and the result/prediction will be reflected on the web page.

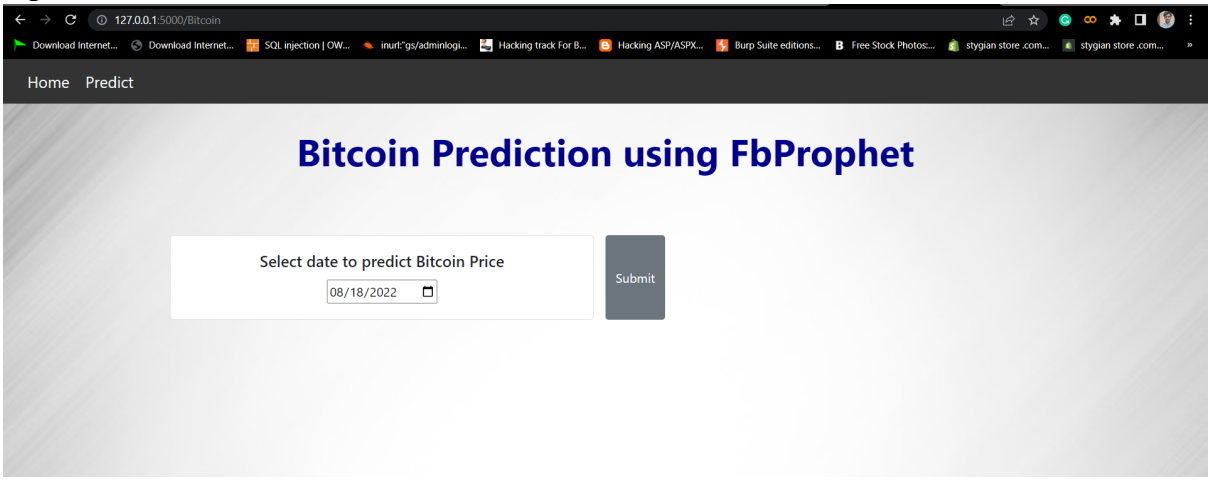


Input 1:- We will select the date.

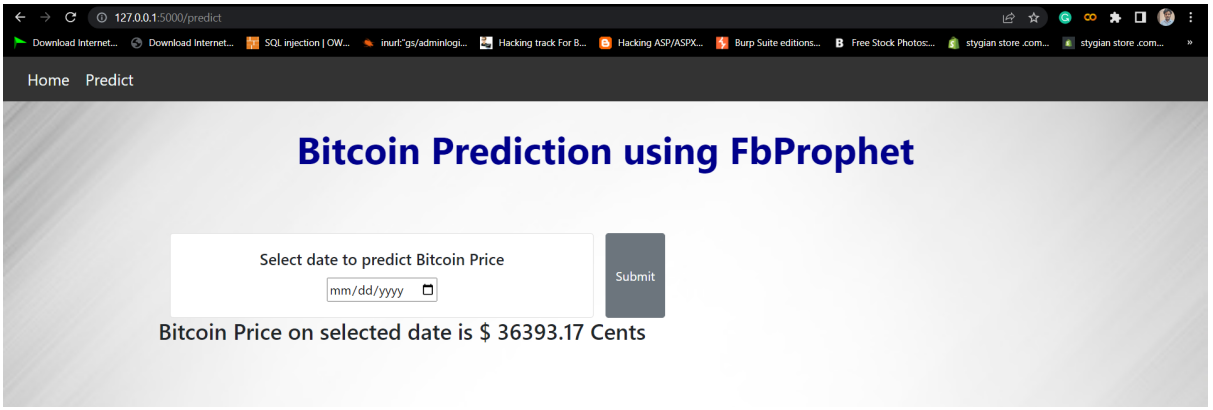


Based on the date selected it predicts the price of Bitcoin.

Input 2:- We will select a different date.



Based on the date selected it predicts the price of Bitcoin.


127.0.0.1:5000/predict
Download Internet...
Download Internet...
SQL injection | OW...
inurl:"gs/adminlogi...
Hacking track For B...
Hacking ASP/ASPX...
Burp Suite editions...
Free Stock Photos:...
stygian store .com...
stygian store .com...
Home
Predict
Bitcoin Prediction using FbProphet
Select date to predict Bitcoin Price
mm/dd/yyyy
Submit
Bitcoin Price on selected date is $ 36393.17 Cents