Team ID Team- 592401

Project Name -Time Series Analysis For Bitcoin Price Prediction Using Fb Prophet

INTRODUCTION

Crypto Price Prediction using FbProphet

The growing interest in cryptocurrency markets and the need for reliable price prediction tools. The volatile and dynamic nature of the cryptocurrency market has led to an increased interest in leveraging advanced technologies to forecast price movements. Cryptocurrency traders, investors, and analysts seek accurate and timely predictions to make informed decisions in this rapidly evolving landscape. This project, titled "Crypto Price Prediction using FbProphet," aims to explore the application of the FbProphet time series forecasting model in predicting cryptocurrency prices.

Purpose:

1.Trading Decision Support: Cryptocurrency traders can use price predictions to make informed decisions about when to buy or sell assets. By having an estimate of future prices, traders can implement trading strategies, set stop-loss orders, or identify potential entry and exit points.

2.Risk Management: Price predictions can help traders and investors better assess the risks associated with their cryptocurrency holdings. Understanding potential price movements allows them to implement risk management strategies and mitigate losses.

3.Portfolio Diversification: Investors can use price predictions to diversify their cryptocurrency portfolios. By having insights into which cryptocurrencies are expected to perform well, they can allocate their investments more effectively.

4.Market Analysis: Cryptocurrency analysts and researchers can use price predictions to gain insights into market trends and dynamics. They can analyze the projected price movements of different cryptocurrencies to understand market sentiment and make recommendations.

5.Hedging Strategies: Institutional investors and businesses can use price predictions to develop hedging strategies to protect against adverse price movements in cryptocurrencies. This can be especially valuable in managing exposure to cryptocurrencies.

6.Investor Confidence: Reliable price predictions can enhance investor confidence. When traders and investors have access to accurate forecasts, they may be more likely to participate in the cryptocurrency market, leading to increased liquidity and market stability.

LITERATURE SURVEY

Existing problem:-

1.High Volatility: Cryptocurrency markets are notoriously volatile, and price movements can be abrupt and unpredictable. FbProphet and other time series models may struggle to capture extreme price fluctuations.

2.Lack of Historical Data: Cryptocurrencies, particularly newer ones, may have limited historical data available, making it challenging to build accurate forecasting models. Short data histories can lead to less reliable predictions.

3.Non-Stationarity: Cryptocurrency price data often exhibit non-stationary behavior, which means that statistical properties change over time. FbProphet assumes stationarity, which can result in suboptimal predictions for non-stationary time series.

4.Market Sentiment: Cryptocurrency prices are influenced by market sentiment, news, and social media trends. FbProphet and traditional time series models do not inherently consider these external factors, which are crucial in cryptocurrency markets.

5.Regulatory Uncertainty: Changes in cryptocurrency regulations or government policies can have a significant impact on prices. Predicting such events and their consequences is a complex challenge.

References:-

1. Prophet Documentation

   https://facebook.github.io/prophet/docs/quick_start.html

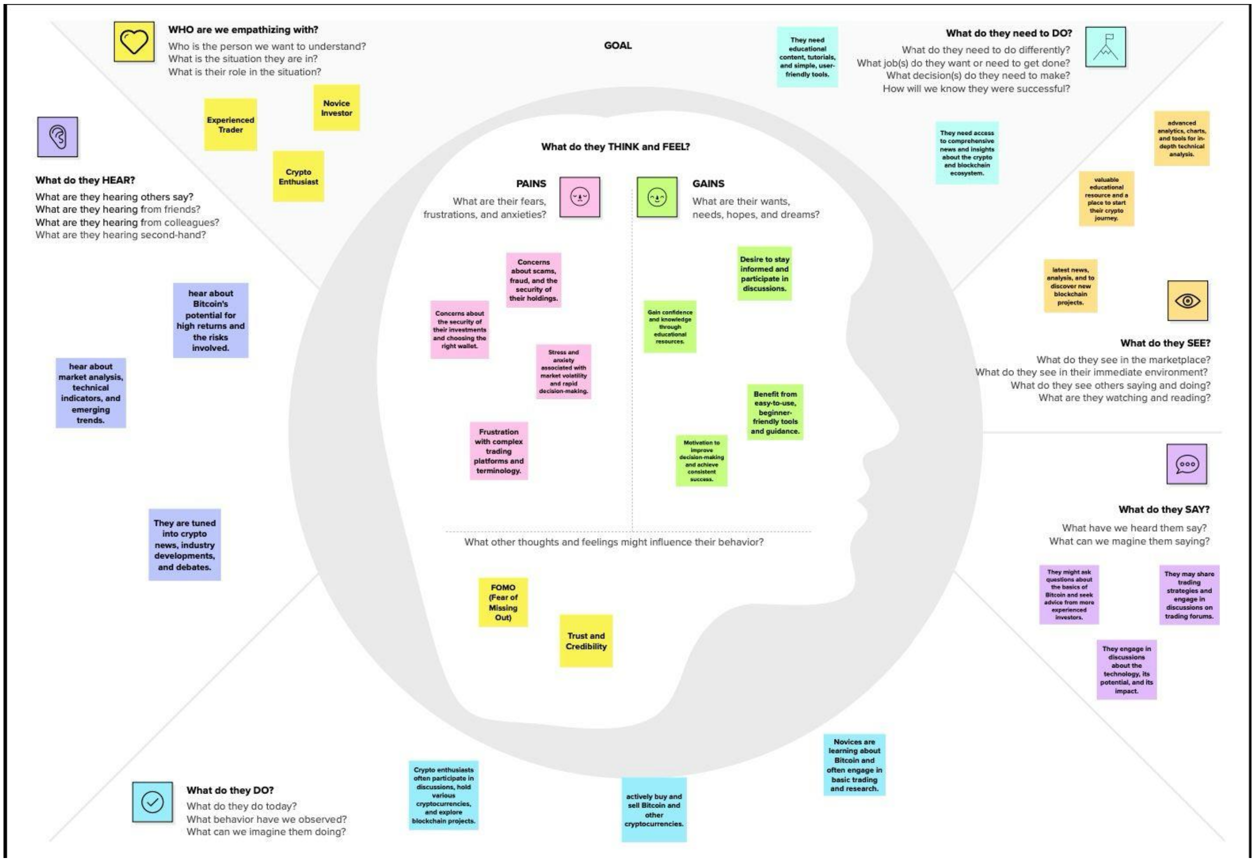
2. Prophet cross-validation and hyperparameter tuning
   https://facebook.github.io/prophet/docs/diagnostics.html

3. Prophet change point detection
   https://facebook.github.io/prophet/docs/trend_changepoints.html

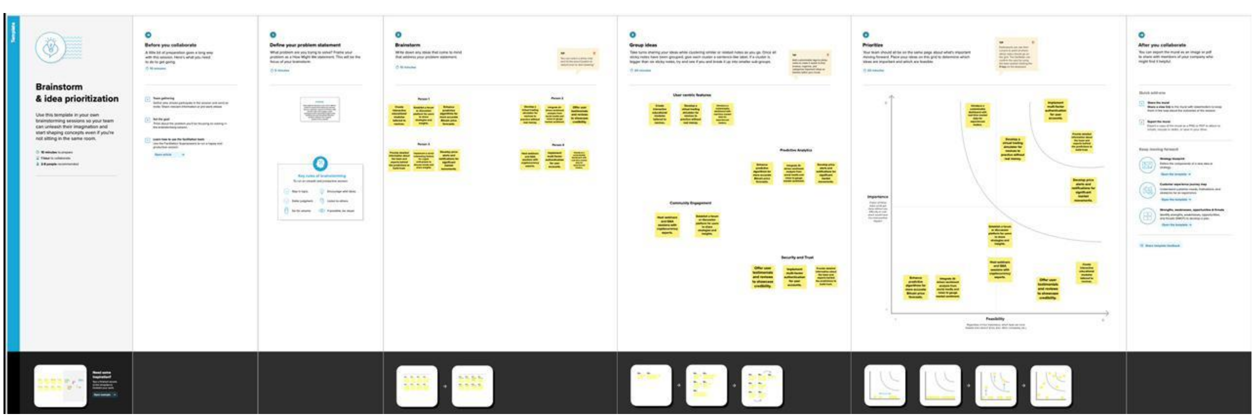
Problem statement:-

Cryptocurrency markets are highly volatile and complex, making it challenging for traders, investors, and analysts to make informed decisions regarding the buying and selling of cryptocurrencies. The lack of reliable forecasting tools for cryptocurrency price movements poses a significant obstacle for market participants who seek to manage risk, optimize trading strategies, and allocate investments effectively. This project aims to develop and evaluate a cryptocurrency price prediction model using FbProphet to provide accurate short-term price forecasts, thereby enabling market participants to make more informed decisions in this dynamic and rapidly evolving market.

Empathy Map Canvas:-



Ideation & Brainstorming:-



Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR - 1	**Data Acquisition and Preprocessing**	• **Data Collection** • **Data Cleaning** • **Feature Engineering**
FR - 2	**Model Selection and Customization**	• **Prophet Integration** • **Parameter Tuning** • **Model Validation**
FR - 3	**Model Development and Training**	• **Data Preparation** • **Model Training** • **Model Evaluation** • **Visualizations**
FR - 4	**User Interface and Accessibility**	• **User Interface Design** • **User Access Levels** • **Real-time Updates**

Non-functional Requirements:

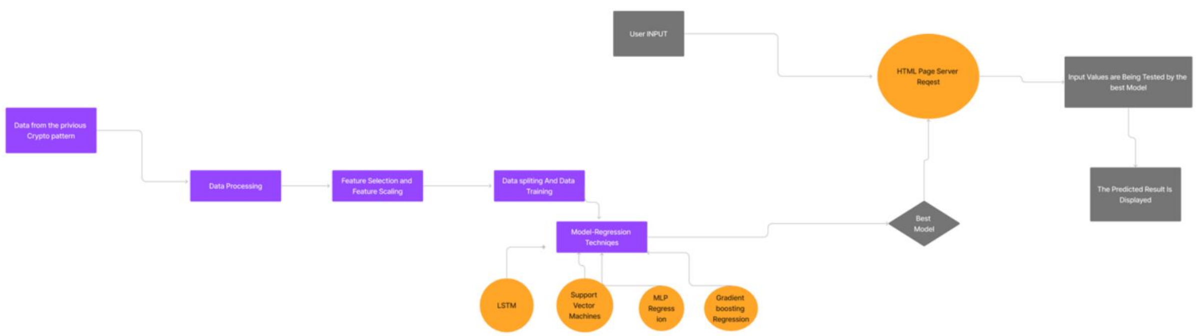
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR - 1	**Usability**	The user interface should be intuitive and user-friendly, allowing users of varying expertise to easily interact with the forecasting model.
NFR - 2	**Security**	Robust security measures should be in place to protect user data and maintain the privacy and integrity of the system.
NFR - 3	**Reliability**	The forecasting model must consistently deliver accurate predictions, with a focus on minimizing errors and ensuring dependable performance.
NFR - 4	**Performance**	The system should provide timely and responsive forecasts, efficiently handling data processing and model training to meet user expectations.
NFR - 5	**Availability**	The system should ensure high availability, minimizing downtime and

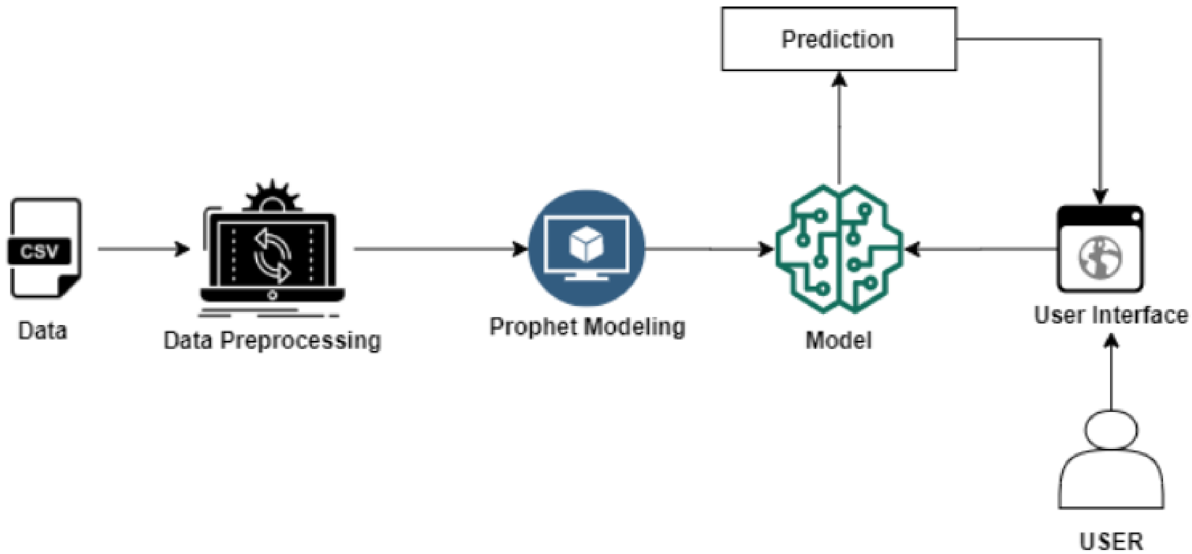
		ensuring users can access the model when needed.
NFR - 6	**Scalability**	The solution should be designed to scale efficiently, accommodating growing data volumes and user demands without compromising performance.

PROJECT DESIGN

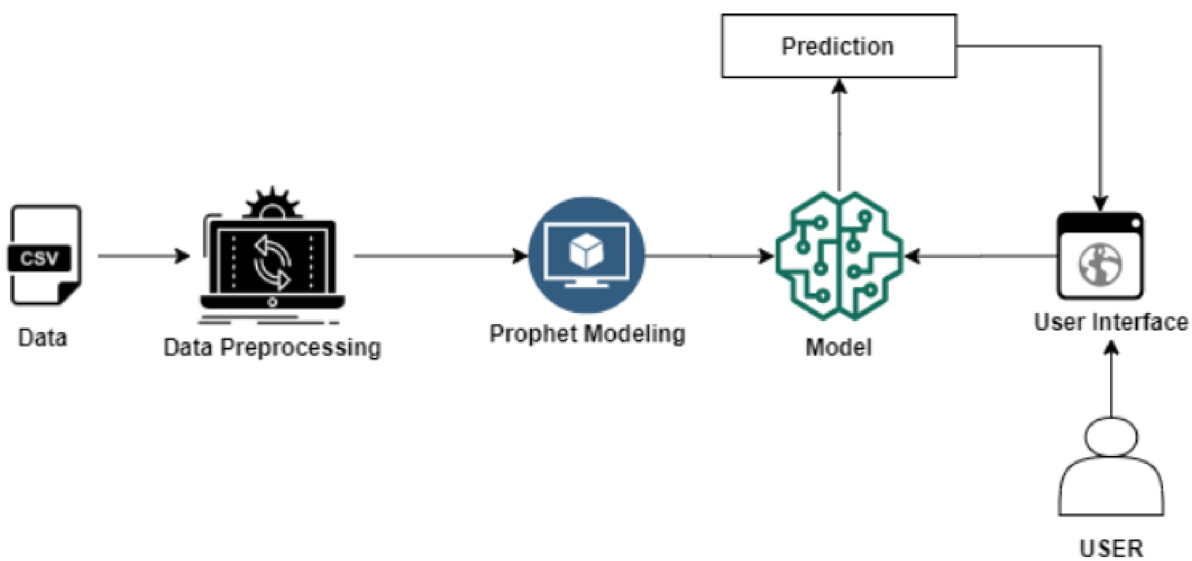
5.1 Data Flow Diagrams & User Stories



Solution Architecture



Technical Architecture



Sprint Planning & Estimation:-

Ideation phase :- We are planning for this is from 13-10-2023

Project Design phase :- 29-10-2023

Project Planning Phase :- 24-10-2023

Project development Phase:- 28-10-2023

Performance & Final Submission Phase :- 07-11-2023

Sprint Delivery Schedule:-

Ideation phase :- We are planning for this is from 18-10-2023

Project Design phase :- 23-10-2023

Project Planning Phase :- 27-10-2023

Project development Phase:- 06-10-2023

Performance & Final Submission Phase :- 09-11-2023

Coding & Solutioning

Feature 1

Real-Time Prediction:

Description: Implementing real-time prediction capabilities allows users to receive up-to-the-minute forecasts for cryptocurrency prices. This feature involves continuously updating the prediction model as new data becomes available, enabling users to make timely decisions in a fast-paced market.

Technical Implementation: Configure the system to fetch and process real-time cryptocurrency price data. Integrate a mechanism to trigger model updates at regular intervals or in response to significant market events. Implement an efficient data streaming or updating mechanism to ensure the latest information is fed into the FbProphet model.

Feature 2

Dynamic Model Adjustments:

Description: This feature involves the dynamic adjustment of the prediction model based on changing market conditions. The system should be able to adapt to different phases of market trends, sudden volatility, or external factors that may impact cryptocurrency prices.

Technical Implementation: Develop algorithms or rules that automatically adjust hyperparameters or retrain the FbProphet model when specific conditions are met. This could involve monitoring model performance metrics, detecting shifts in market behavior, or incorporating external indicators. Implement a feedback loop that continuously evaluates the model's accuracy and triggers adjustments when necessary.

Database Schema

Input Database Schema

Date Date

Open float

High float

Low float

Close float

Adj float

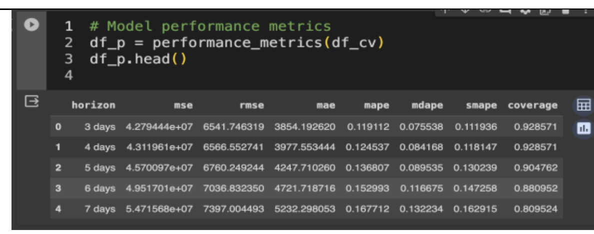
Close float

Volume int

Performance Testing

Performance Metrics

Model Performance Testing:

<table>
<tr><th>S.No.</th><th>Parameter</th><th>Values</th><th>Screenshot</th></tr>
<tr>
<td>1.</td>
<td>Metrics</td>
<td>Regression Model:
MAE - 4247.710260(5 days) ,
MSE - 4.570097e+07(5 days) ,
RMSE – 6760.249244 (5 days) ,
R2 score - 0.06465503430782771</td>
<td>


```
1 # Model performance metrics
2 df_p = performance_metrics(df_cv)
3 df_p.head()
4
```

	horizon	mse	rmse	mae	mape	mdape	smape	coverage
0	3 days	4.279444e+07	6541.746319	3854.192620	0.119112	0.075538	0.111936	0.928571
1	4 days	4.311961e+07	6566.552741	3977.553444	0.124537	0.084168	0.118147	0.928571
2	5 days	4.570097e+07	6760.249244	4247.710260	0.136807	0.089535	0.130239	0.904762
3	6 days	4.951701e+07	7036.832350	4721.718716	0.152993	0.116675	0.147258	0.880952
4	7 days	5.471568e+07	7397.004493	5232.298053	0.167712	0.132234	0.162915	0.809524

```
[27] 1 r2_score(df_cv.y, df_cv.yhat)
0.06465503430782771
```
</td>
</tr>
</table>

2.	Tune the Model		

Hyperparameter Tuning –

Create the prophet model with confidence internal of 95%

m = Prophet(interval_width=0.95, n_changepoints=7)

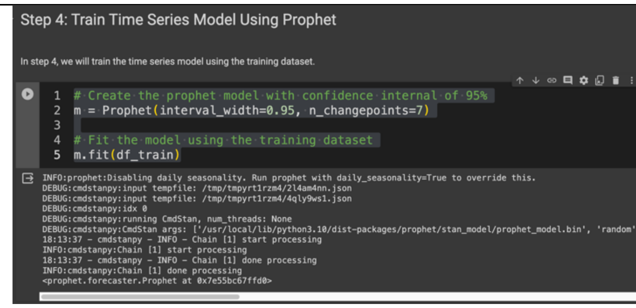
Fit the model using the training dataset

m.fit(df_train)

Validation Method –

In step 8, we will do cross-validation for the time series model. Prophet has a cross_validation function to automate the comparison between the actual and the predicted values.

- `m' is the trained model.
- `initial='500 days" means the initial model will be trained on the first 500 days of data.
- `period='60 days" means 60 days will be added to the training dataset for each additional model.
- horizon = '30 days' means that the model forecasts the next 30 days. When only horizon is given, Prophet defaults initial to be triple the horizon, and period to be half of the horizon.

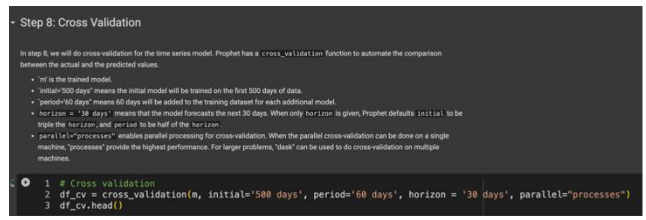


Step 4: Train Time Series Model Using Prophet

In step 4, we will train the time series model using the training dataset.

```
1 # Create the prophet model with confidence internal of 95%
2 m = Prophet(interval_width=0.95, n_changepoints=7)
3
4 # Fit the model using the training dataset
5 m.fit(df_train)
```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/2l4am4nn.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/4qly9ws1.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random',
18:13:37 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
18:13:37 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7e55bc67ffd0>
```

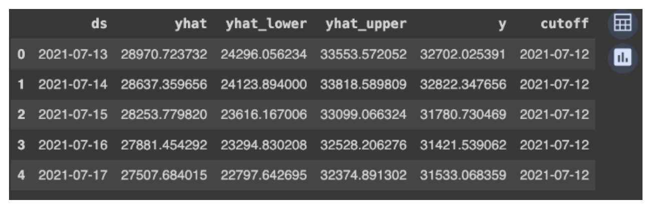


Step 8: Cross Validation

In step 8, we will do cross-validation for the time series model. Prophet has a `cross_validation` function to automate the comparison between the actual and the predicted values.

- `m' is the trained model.
- `initial='500 days" means the initial model will be trained on the first 500 days of data.
- `period='60 days" means 60 days will be added to the training dataset for each additional model.
- `horizon = '30 days'` means that the model forecasts the next 30 days. When only `horizon` is given, Prophet defaults `initial` to be triple the `horizon`, and `period` to be half of the `horizon`.
- `parallel="processes"` enables parallel processing for cross-validation. When the parallel cross-validation can be done on a single machine, "processes" provide the highest performance. For larger problems, "dask" can be used to do cross-validation on multiple machines.

```
1 # Cross validation
2 df_cv = cross_validation(m, initial='500 days', period='60 days', horizon = '30 days', parallel="processes")
3 df_cv.head()
```

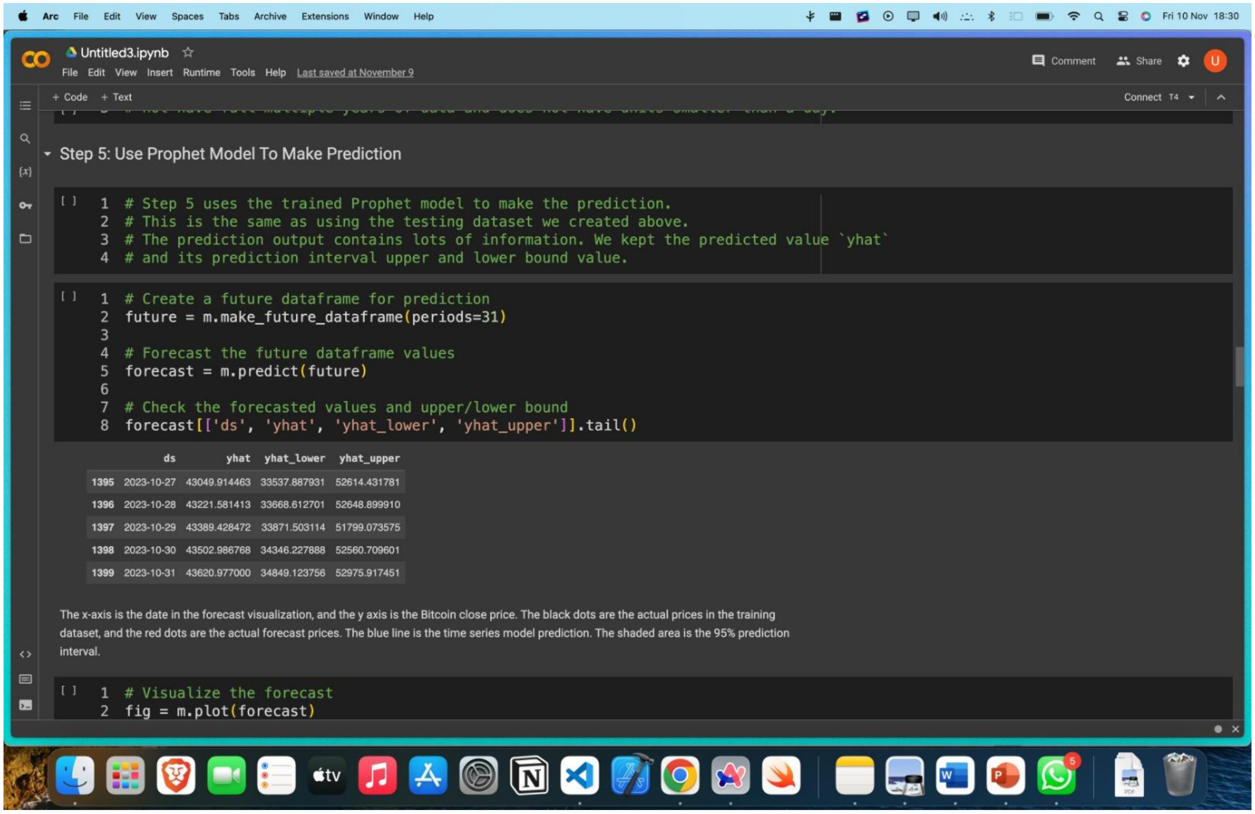


	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2021-07-13	28970.723732	24296.056234	33553.572052	32702.025391	2021-07-12
1	2021-07-14	28637.359656	24123.894000	33818.589809	32822.347656	2021-07-12
2	2021-07-15	28253.779820	23616.167006	33099.066324	31780.730469	2021-07-12
3	2021-07-16	27881.454292	23294.830208	32528.206276	31421.539062	2021-07-12
4	2021-07-17	27507.684015	22797.642695	32374.891302	31533.068359	2021-07-12

		• parallel="processes" enables parallel processing for cross-validation. When the parallel cross-validation can be done on a single machine, "processes" provide the highest performance. For larger problems, "dask" can be used to do cross-validation on multiple machines.	

Results

Output Screenshots



Untitled3.ipynb

Step 5: Use Prophet Model To Make Prediction

```
# Step 5 uses the trained Prophet model to make the prediction.
# This is the same as using the testing dataset we created above.
# The prediction output contains lots of information. We kept the predicted value `yhat`
# and its prediction interval upper and lower bound value.
```

```
# Create a future dataframe for prediction
future = m.make_future_dataframe(periods=31)

# Forecast the future dataframe values
forecast = m.predict(future)

# Check the forecasted values and upper/lower bound
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
1395	2023-10-27	43049.914463	33537.887931	52614.431781
1396	2023-10-28	43221.581413	33668.612701	52648.899910
1397	2023-10-29	43389.428472	33871.503114	51799.073575
1398	2023-10-30	43502.986768	34346.227888	52560.709601
1399	2023-10-31	43620.977000	34849.123756	52975.917451

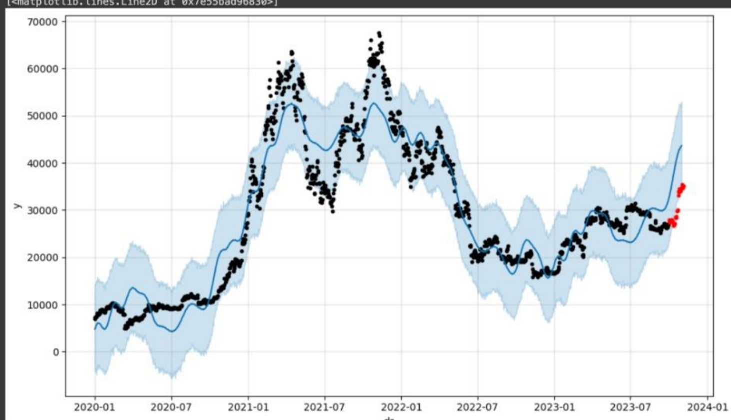
The x-axis is the date in the forecast visualization, and the y axis is the Bitcoin close price. The black dots are the actual prices in the training dataset, and the red dots are the actual forecast prices. The blue line is the time series model prediction. The shaded area is the 95% prediction interval.

```
# Visualize the forecast
fig = m.plot(forecast)
```

The x-axis is the date in the forecast visualization, and the y axis is the Bitcoin close price. The black dots are the actual prices in the training dataset, and the red dots are the actual forecast prices. The blue line is the time series model prediction. The shaded area is the 95% prediction interval.

```
# Visualize the forecast
fig = m.plot(forecast)
ax = fig.gca()
ax.plot( df_test["ds"], df_test["y"], 'r.')
```

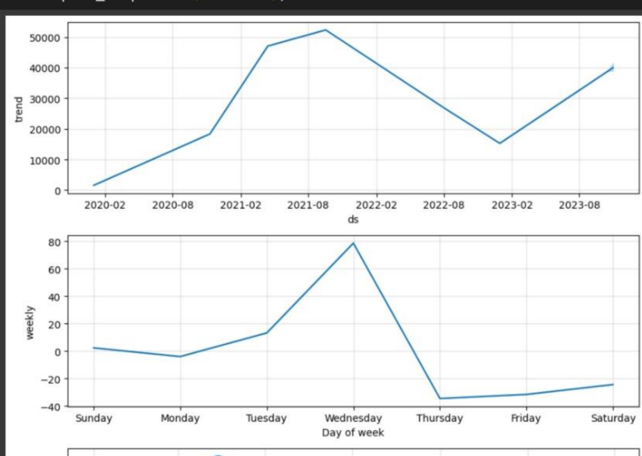
```
[<matplotlib.lines.Line2D at 0x7e55bad96830>]
```

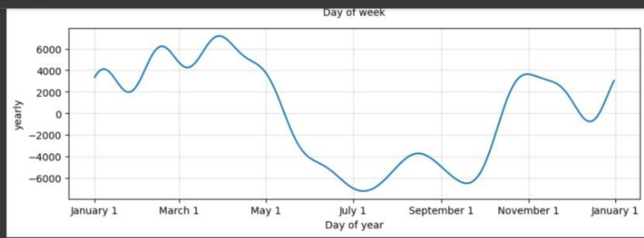


Step 6: Time Series Decomposition

In step 6, we will decompose the time series forecast.

```
# Visualize the components
m.plot_components(forecast);
```





Step 7: Identify Change Points

Prophet automatically identifies the change points in time series data following the steps below:

1. Specify the percentage of data used for identifying change points. The default is 80%.
2. Identify a large number of uniformly distributed dates with possible trajectories change.
3. Apply a sparse prior on the magnitudes of the change rate, which is similar to L1 regularization.

We can list the dates corresponding to the changepoints using `.changepoints`.

```python
# Default change points
print(f'There are {len(m.changepoints)} change points. \nThe change points dates are \n{df.loc[df["ds"].isin(m.changepoints)]}'
```

```
There are 7 change points.
The change points dates are
             ds             y
156  2020-06-05   9665.533203
313  2020-11-09  15332.315430
469  2021-04-14  63109.695312
625  2021-09-17  47267.519531
781  2022-02-20  38431.378906
938  2022-07-27  22930.548828
```

machine, "processes" provide the highest performance. For larger problems, "dask" can be used to do cross-validation on multiple machines.

```python
# Cross validation
df_cv = cross_validation(m, initial='500 days', period='60 days', horizon = '30 days', parallel="processes")
df_cv.head()
```

```
INFO:cmdstanpy:Chain [1] start processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/33gxf2v0.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/4hftlgzf.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=93950', 'data', 'file=/tmp/tmpyrt1rzm4/33gxf2v0.json', 'init=/tmp/tmpyrt1rzm4/4hft
18:13:44 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
18:13:44 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
18:13:44 - cmdstanpy - ERROR - Chain [1] error: error during processing Operation not permitted
ERROR:cmdstanpy:Chain [1] error: error during processing Operation not permitted
WARNING:prophet.models:Optimization terminated abnormally. Falling back to Newton.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/5t8b0xbn.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/fh3ekxh0.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=79606', 'data', 'file=/tmp/tmpyrt1rzm4/5t8b0xbn.json', 'init=/tmp/tmpyrt1rzm4/fh3e
18:13:44 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
18:13:45 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
18:13:45 - cmdstanpy - ERROR - Chain [1] error: error during processing Operation not permitted
ERROR:cmdstanpy:Chain [1] error: error during processing Operation not permitted
WARNING:prophet.models:Optimization terminated abnormally. Falling back to Newton.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/iw19grm3.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/y5vog0si.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=45358', 'data', 'file=/tmp/tmpyrt1rzm4/iw19grm3.json', 'init=/tmp/tmpyrt1rzm4/y5vo
18:13:45 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
18:13:47 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/vadiv4sw.json
18:13:47 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/4j37tjos.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=32546', 'data', 'file=/tmp/tmpyrt1rzm4/vadiv4sw.json', 'init=/tmp/tmpyrt1rzm4/4j37
18:13:48 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
18:13:48 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

INFO:cmdstanpy:Chain [1] done processing

	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2021-07-13	28970.723732	24296.056234	33553.572052	32702.025391	2021-07-12
1	2021-07-14	28637.359656	24123.894000	33818.589809	32822.347656	2021-07-12
2	2021-07-15	28253.779820	23616.167006	33099.066324	31780.730469	2021-07-12
3	2021-07-16	27881.454292	23294.830208	32528.206276	31421.539062	2021-07-12
4	2021-07-17	27507.684015	22797.642695	32374.891302	31533.068359	2021-07-12

Step 9: Prophet Model Performance Evaluation

Step 9 evaluates the cross-validation model performance.

- MSE (Mean Squared Error) sums up the squared difference between actual and prediction and is divided by the number of predictions.
- RMSE (Root Mean Square Error) takes the square root of MSE.
- MAE (Mean Absolute Error) sums up the absolute difference between actual and prediction and is divided by the number of predictions.
- MAPE (Mean Absolute Percentage Error) sums up the absolute percentage difference between actual and prediction and is divided by the number of predictions. MAPE is independent of the magnitude of data, so it can be used to compare different forecasts. But it's undefined when the actual value is zero.
- MDAPE (Median Absolute Percentage Error) is similar to MAPE. The difference is that it calculates the median instead of taking the average of the absolute percentage difference.
- SMAPE (Symmetric Mean Absolute Percentage Error) is similar to MAPE. The difference is that when calculating absolute percentage error, the denominator is the actual value for MAPE and the average of the actual and predicted value for SMAPE.

```python
# Model performance metrics
df_p = performance_metrics(df_cv)
df_p.head()
```

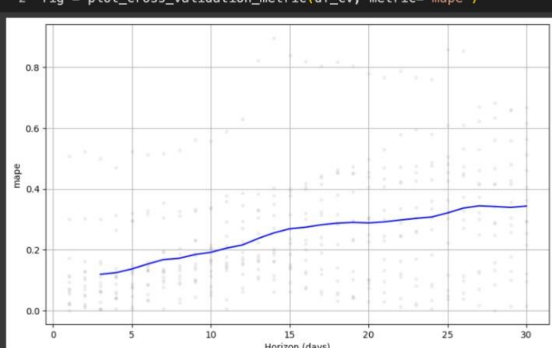
	horizon	mse	rmse	mae	mape	mdape	smape	coverage
0	3 days	4.279444e+07	6541.746319	3854.192620	0.119112	0.075538	0.111936	0.928571
1	4 days	4.311961e+07	6566.552741	3977.553444	0.124537	0.084168	0.118147	0.928571
2	5 days	4.570097e+07	6760.249244	4247.710260	0.136807	0.089535	0.130239	0.904762
3	6 days	4.951701e+07	7036.832350	4721.718716	0.152993	0.116675	0.147258	0.880952
4	7 days	5.471568e+07	7397.004493	5232.298053	0.167712	0.132234	0.162915	0.809524

	horizon	mse	rmse	mae	mape	mdape	smape	coverage
3	6 days	4.951701e+07	7036.832350	4721.718716	0.152993	0.116675	0.147258	0.880952
4	7 days	5.471568e+07	7397.004493	5232.298053	0.167712	0.132234	0.162915	0.809524

`plot_cross_validation_metric` method from Prophet helps us to plot the cross-validation performance results.

- The x-axis is the horizon. Because we set the horizon to be 30 days, the x-axis has a value up to 30.
- The y-axis is the metric we are interested in. We use `mape` as an example in this visualization.
- On each day, we can see three dots. This is because there are three models in the cross-validation, and each dot represents the MAPE from one model.
- The line is the aggregated performance across all the models. We can see that MAPE value increases with days, which is expected because time series tend to make better predictions for the near future than the far future.

```python
# Visualize the performance metrics
fig = plot_cross_validation_metric(df_cv, metric='mape')
```



Advantage

Ease of Use:

FbProphet is designed to be user-friendly and requires minimal configuration compared to more complex forecasting models. This makes it accessible to a broader audience, including individuals with limited expertise in machine learning.

Handling Missing Data:

FbProphet is robust in handling missing data and outliers. It employs a Bayesian approach to fill in gaps in the time series data, making it suitable for datasets with irregularities.

Automatic Seasonality Detection:

FbProphet automatically detects and incorporates seasonal patterns in the data, which is crucial for capturing recurring trends in cryptocurrency prices. This simplifies the modeling process and enhances the accuracy of predictions.

Flexibility for Customization:

While providing simplicity, FbProphet also allows users to incorporate custom seasonality, holidays, and special events that might influence cryptocurrency prices. This flexibility enables users to tailor the model to the specific characteristics of the market.

Handling Trends and Holidays:

FbProphet is well-suited for time series data with both strong trends and holiday effects. It can effectively capture long-term trends and adjust for holidays or significant events that might impact crypto prices.

Scalability:

FbProphet is designed to scale well with large datasets, making it suitable for applications with a considerable amount of historical price data. This scalability is important when dealing with the extensive and continuously growing cryptocurrency market.

Disadvantages

Limited Complexity:

FbProphet is designed to be a simple and user-friendly forecasting tool. While this is an advantage for many users, it also means that the model might struggle to capture complex relationships and patterns present in highly dynamic and unpredictable cryptocurrency markets.

Assumption of Additive Components:

FbProphet assumes that various components (trend, seasonality, holidays) are additive, which may not always hold true for all types of time series data. Cryptocurrency prices, influenced by various factors, might exhibit non-additive behavior, limiting the model's flexibility in certain scenarios.

Limited Feature Engineering:

The simplicity of FbProphet comes at the cost of limited support for advanced feature engineering. Users looking to incorporate intricate features or complex external factors into their predictions may find the model restrictive.

Fixed Seasonality Patterns:

FbProphet assumes that seasonal patterns are fixed over time, which may not capture changes in the market behavior. In cryptocurrency markets, seasonality might evolve or shift, and FbProphet's fixed patterns might not adapt effectively.

Sensitivity to Hyperparameters:

FbProphet's performance can be sensitive to the selection of hyperparameters, and finding the optimal values may require some experimentation. In some cases, model performance might be suboptimal if hyperparameters are not properly tuned.

Uncertainty Estimation Challenges:

While FbProphet provides uncertainty intervals for predictions, accurately estimating uncertainty can be challenging, especially during periods of high volatility. Users should interpret uncertainty intervals cautiously in rapidly changing market conditions.

Conclusion

In conclusion, the project on "Crypto Price Prediction using FbProphet" represents a significant effort to leverage time series forecasting techniques for predicting cryptocurrency prices. Throughout the course of this project, various aspects were considered, ranging from data collection and preprocessing to the application of the FbProphet model and the evaluation of its performance. The following key points summarize the findings and implications of the project:

Methodology:

The project utilized the FbProphet time series forecasting model, a user-friendly and accessible tool developed by Facebook. The model was applied to historical and real-time cryptocurrency price data to generate predictions.

Data Preprocessing:

Extensive data preprocessing steps were undertaken to handle missing values, outliers, and ensure the quality of the input data. The robustness of the FbProphet model in dealing with such preprocessing challenges was demonstrated.

Future Scope

The future of the "Crypto Price Prediction using FbProphet" project holds several possibilities for further development and enhancements. Here are some potential directions and considerations for the future of the project:

Integration of Additional Models:

Explore the integration of additional time series forecasting models and machine learning techniques to complement FbProphet. Ensemble methods or deep learning models could be considered to capture more intricate patterns in cryptocurrency price data.

Feature Engineering and External Factors:

Enhance the predictive power of the model by incorporating advanced feature engineering techniques and external factors that might influence cryptocurrency prices. This could include sentiment analysis of news, social media trends, or macroeconomic indicators.

APPENDIX

Source Code

```
!pip install yfinance prophet
```

```
# Data processing
import numpy as np
import pandas as pd

# Get time series data
import yfinance as yf

# Prophet model for time series forecast
from prophet import Prophet
from prophet.plot import add_changepoints_to_plot, plot_cross_validation_metric
from prophet.diagnostics import cross_validation, performance_metrics

# Visualization
import plotly.graph_objs as go
```

```
# Download Bitcoin data
data = yf.download(tickers='BTC-USD', start='2020-01-01', end='2023-11-06',
interval = '1d')

# Reset index and have date as a column
data.reset_index(inplace=True)

# Change date to datetime format
data['Date'] = pd.to_datetime(data['Date'])

# Take a look at the data
data.head()
```

```
 Declare a figure
```

```
fig = go.Figure()

# Candlestick chart
fig.add_trace(go.Candlestick(x=data.Date,
                open=data['Open'],
                high=data['High'],
                low=data['Low'],
                close=data['Close'],
                name = 'Bitcoin Data'))
```

```
# Keep only date and close price
df = data.drop(['Open', 'High', 'Low', 'Adj Close', 'Volume'], axis=1)

# Rename date to ds and close price to y
df.rename (columns={'Date': 'ds', 'Close': 'y'}, inplace=True)

# Take a look at the data
df.head ()
```

```
# Data information
df.info()
```

```
# Train test split
df_train = df[df['ds']<='2023-09-30']
df_test = df[df['ds']>'2023-09-30']

# Print the number of records and date range for training and testing dataset.
print('The training dataset has', len(df_train), 'records, ranging from',
df_train['ds'].min(), 'to', df_train['ds'].max())
print('The testing dataset has', len(df_test), 'records, ranging from',
df_test['ds'].min(), 'to', df_test['ds'].max())
```

```
# Create the prophet model with confidence internal of 95%
m = Prophet(interval_width=0.95, n_changepoints=7)

# Fit the model using the training dataset
m.fit(df_train)
```

```
# Create a future dataframe for prediction
```

```python
future = m.make_future_dataframe(periods=31)

# Forecast the future dataframe values
forecast = m.predict(future)

# Check the forecasted values and upper/lower bound
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

```python
# Visualize the forecast
fig = m.plot(forecast)
ax = fig.gca()
ax.plot( df_test["ds"], df_test["y"], 'r.')
```

```python
# Visualize the components
m.plot_components(forecast);
```

```python
# Default change points
print(f'There are {len(m.changepoints)} change points. \nThe change points dates
are \n{df.loc[df["ds"].isin(m.changepoints)]}')
```

```python
# Change points to plot
fig = m.plot(forecast)
a = add_changepoints_to_plot(fig.gca(), m, forecast)
```

```python
# Cross validation
df_cv = cross_validation(m, initial='500 days', period='60 days', horizon = '30
days', parallel="processes")
df_cv.head()
```

```python
# Model performance metrics
df_p = performance_metrics(df_cv)
df_p.head()
```

```python
# Visualize the performance metrics
fig = plot_cross_validation_metric(df_cv, metric='mape')
```

```python
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

```python
r2_score(df_cv.y, df_cv.yhat)
```

```python
mean_squared_error(df_cv.y, df_cv.yhat)
```

```python
mean_absolute_error(df_cv.y, df_cv.yhat)
```

GitHub

https://github.com/smartinternz02/SI-GuidedProject-601585-1697638773

Project Demo Link

https://drive.google.com/file/d/1-0LTyXHy0b11rBxglZxHrqojpFUZ3NAz/view?usp=sharing