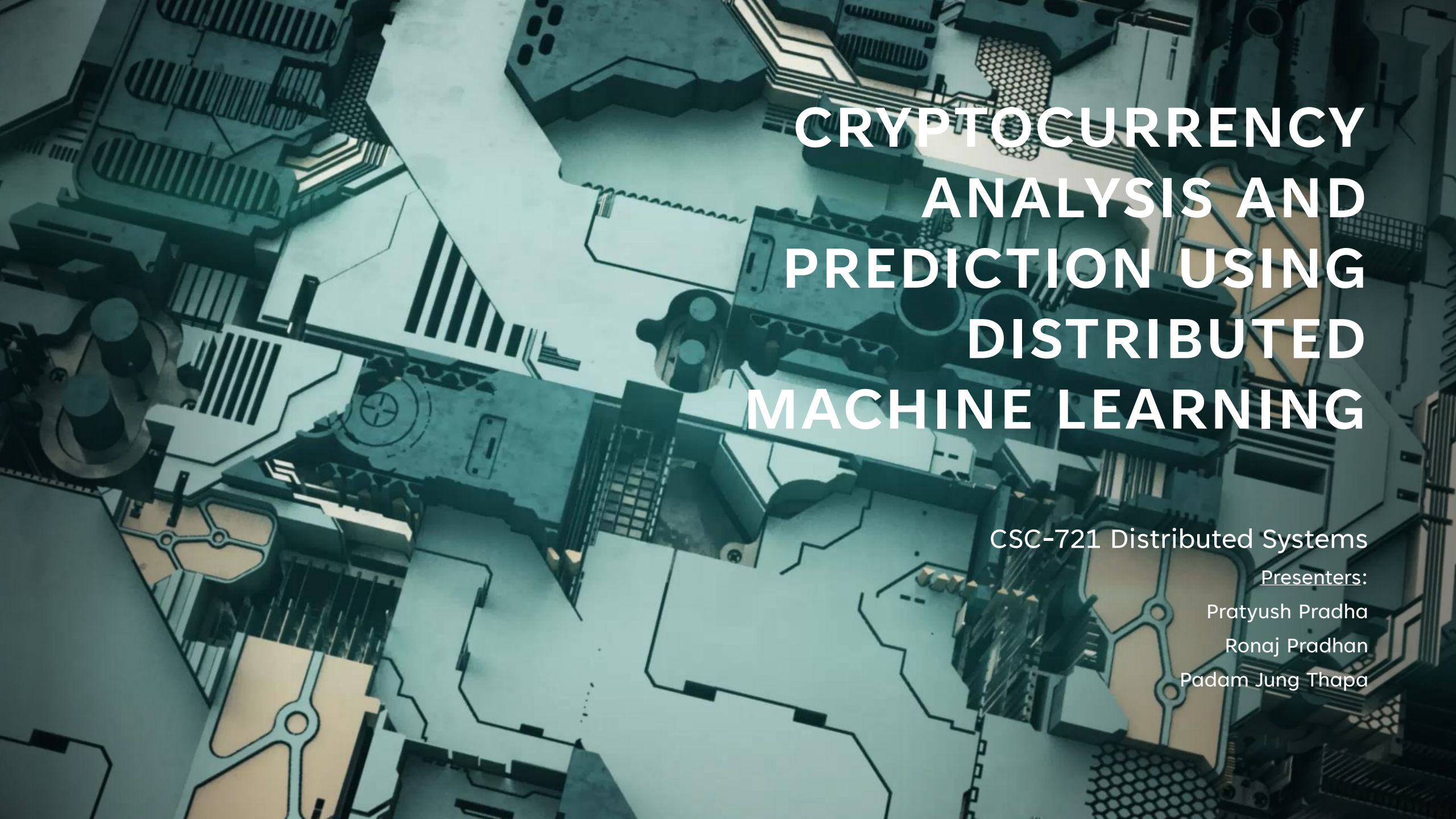


CRYPTOCURRENCY ANALYSIS AND PREDICTION USING DISTRIBUTED MACHINE LEARNING

CSC-721 Distributed Systems

Presenters:

Pratyush Pradha

Ronaj Pradhan

Padam Jung Thapa

AGENDA

Introduction

Motivation / Uniqueness

Materials and Methods

Methodology

Results and Analysis

Possible Future Improvements

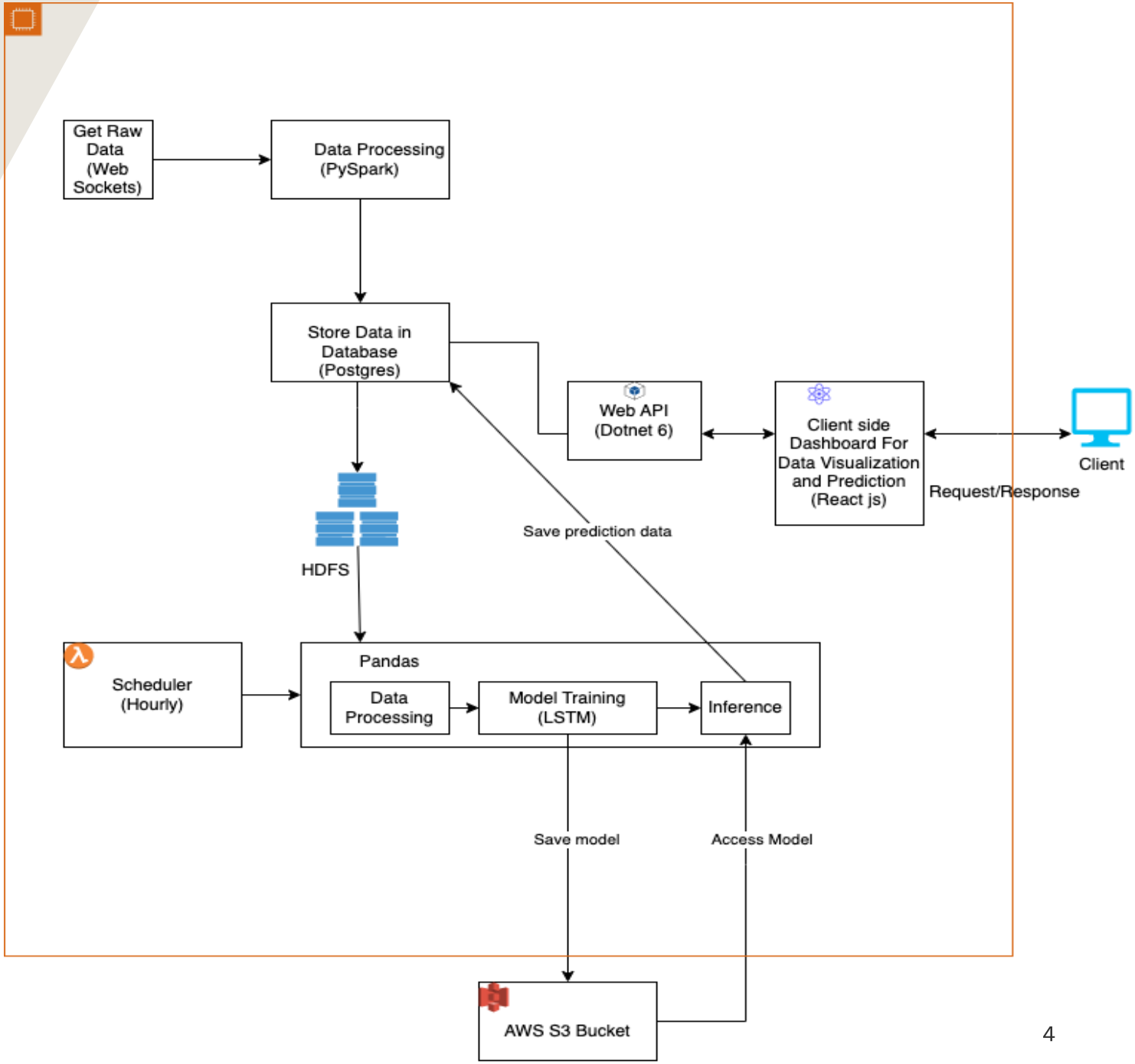
INTRODUCTION

Analysis of Ten most popular cryptocurrencies including Bitcoin, and Ethereum. We use the websockets provided by Binance to gather data and then perform ETL operations and analysis using PySpark which is later saved in PostGres DB and HDFS. After that .Net 6 web APIs are used as backend which utilizes Entity Framework as ORM and performs more granual data transformation and exposes the Data as APIs. ML model then uses the LSTM model to produce a prediction which is finally displayed in the Dashboard using React, Material UI and Apex Chart. The ML model is then saved in an Amazon S3 platform.

Aim:

1)Predict Hourly Closing Prices of the Cryptocurrencies: Bitcoin, Ethereum, Ripple, Binance Coin, DogeCoin, Cardano, Polygon, Polkadot, Solana, USDC.

METHODOLOGY

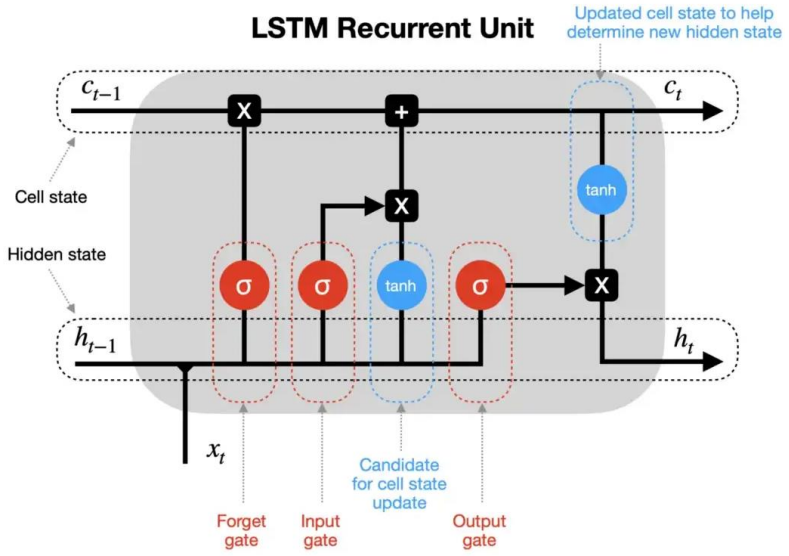


MOTIVATION/UNIQUENESS

- Decoupled, Deployable and Scalable System Architecture.
- Utilization of Multiple Servers/Clusters.
- Reusability across different areas of business
- Distributed and Cloud Computing architecture
- Improved fault isolation
- Cohesive Heterogeneous Solution, which Utilizes all the aspects of the Web
- All the modules of the system are OS independent

MATERIALS AND METHODS

- Datasets – Continuous real time data fetched through web sockets from Binance.com
- LSTMs – Time Series Analysis Model.
- PySpark and Pandas for ETL Operations. Spark is 100 times faster than hadoop mapreduce.
- PySpark – Apache Spark Interface.
- Why Postgres? Structured Data – Websockets.




```
def train_keras_model(X_train, y_train, epochs, batch_size,forecast,lookback, shuffle=False):

# Initializing the Neural Network based on LSTM
    model = Sequential()
    model.add(LSTM(units=50,return_sequences=True,input_shape=(lookback,1)))
    model.add(LSTM(units=50))
    model.add(Dense(forecast))
    model.compile(optimizer='adam',loss='mean_squared_error')

    model.fit(X_train, y_train, shuffle=shuffle, validation_split = 0.1, epochs=epochs, verbose=2, batch_size=batch_size)



    return model
```

RESULTS AND ANALYSIS

Dashboard

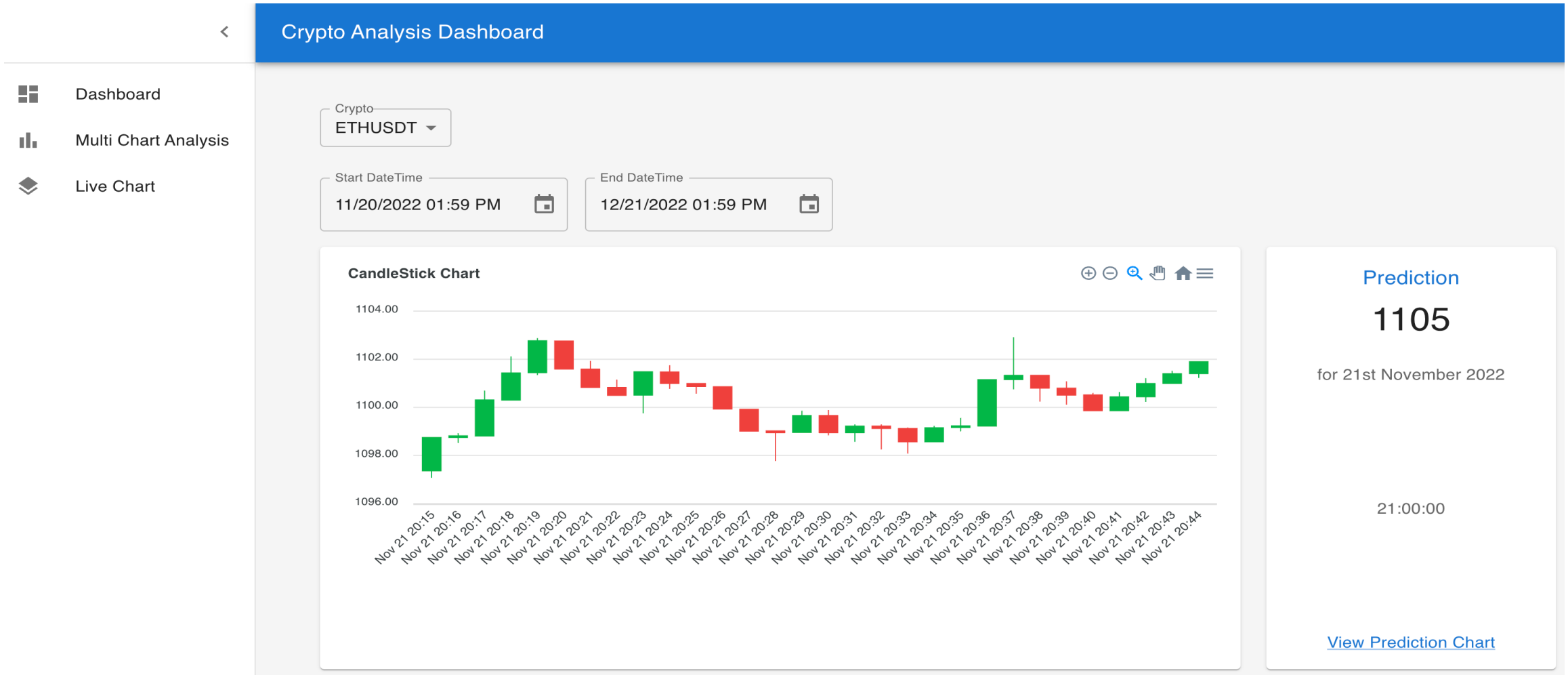
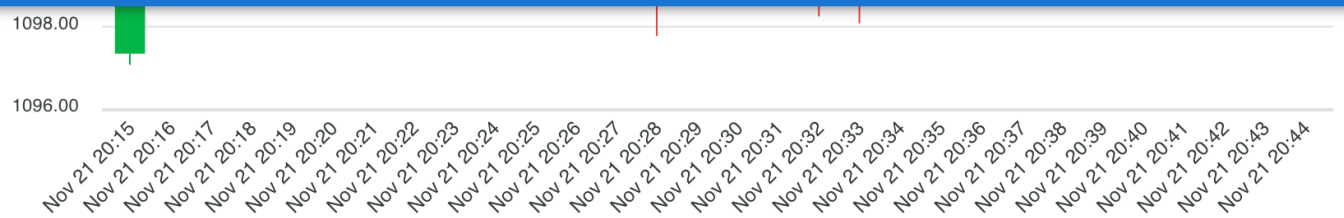


Fig: Candlestick Chart of Cryptocurrency Data (.Net 6, React, Material UI, and ApexChart)

Crypto Analysis Dashboard

- Dashboard
- Multi Chart Analysis
- Live Chart



21:00:00

View Prediction Chart

Symbol	Date	Total Number of Trades	Total Base Volume	Total Quote Volume
ADAUSDT	2022-11-21T00:00:00	1588	1656323.8	505331.43651
BNBUSDT	2022-11-21T00:00:00	3866	4439.7789999999995	1137825.1675
BTCUSDT	2022-11-21T00:00:00	103011	3618.61689	57204642.481162496
DOGEUSDT	2022-11-21T00:00:00	5376	42041196	3152763.94626
DOTUSDT	2022-11-21T00:00:00	640	69999.66	361916.8359
ETHUSDT	2022-11-21T00:00:00	9716	7342.498	8077432.107655

1–9 of 9

```python
def input_and_output_sequence(data,n_lookback,n_forecast):

    X = []
    Y = []

    for i in range(n_lookback, len(data) - n_forecast + 1):
        X.append(data[i - n_lookback: i])
        Y.append(data[i: i + n_forecast])

    return np.array(X), np.array(Y)
```

	Actual	Predicted
0	3729.92	3618.018555
1	3729.12	3609.883301
2	3747.72	3602.238281
3	3733.96	3597.714844
4	3745.54	3593.635254
...	...	...
9074	1563.69	1543.020264
9075	1564.79	1540.242798
9076	1557.71	1538.135010
9077	1565.83	1535.784424
9078	1572.69	1534.450439

9079 rows × 2 columns

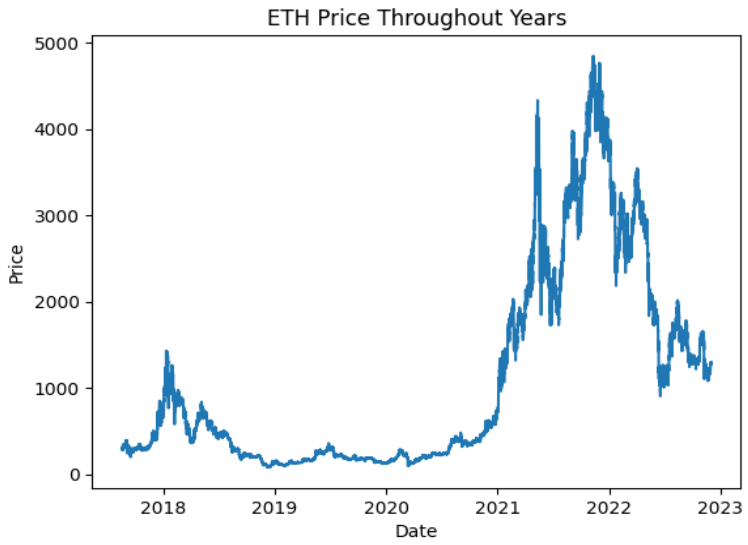
```python
from sklearn.metrics import mean_absolute_error

rmse = mean_squared_error(y_test,predictions_test,squared=False)
mape = mean_absolute_error(y_test,predictions_test)

print(f"rmse: {rmse}")
print(f"mape: {mape}")
```

```
rmse: 0.049191636035688845
mape: 0.03465241977875644
```

Fig: Evaluation Metrics



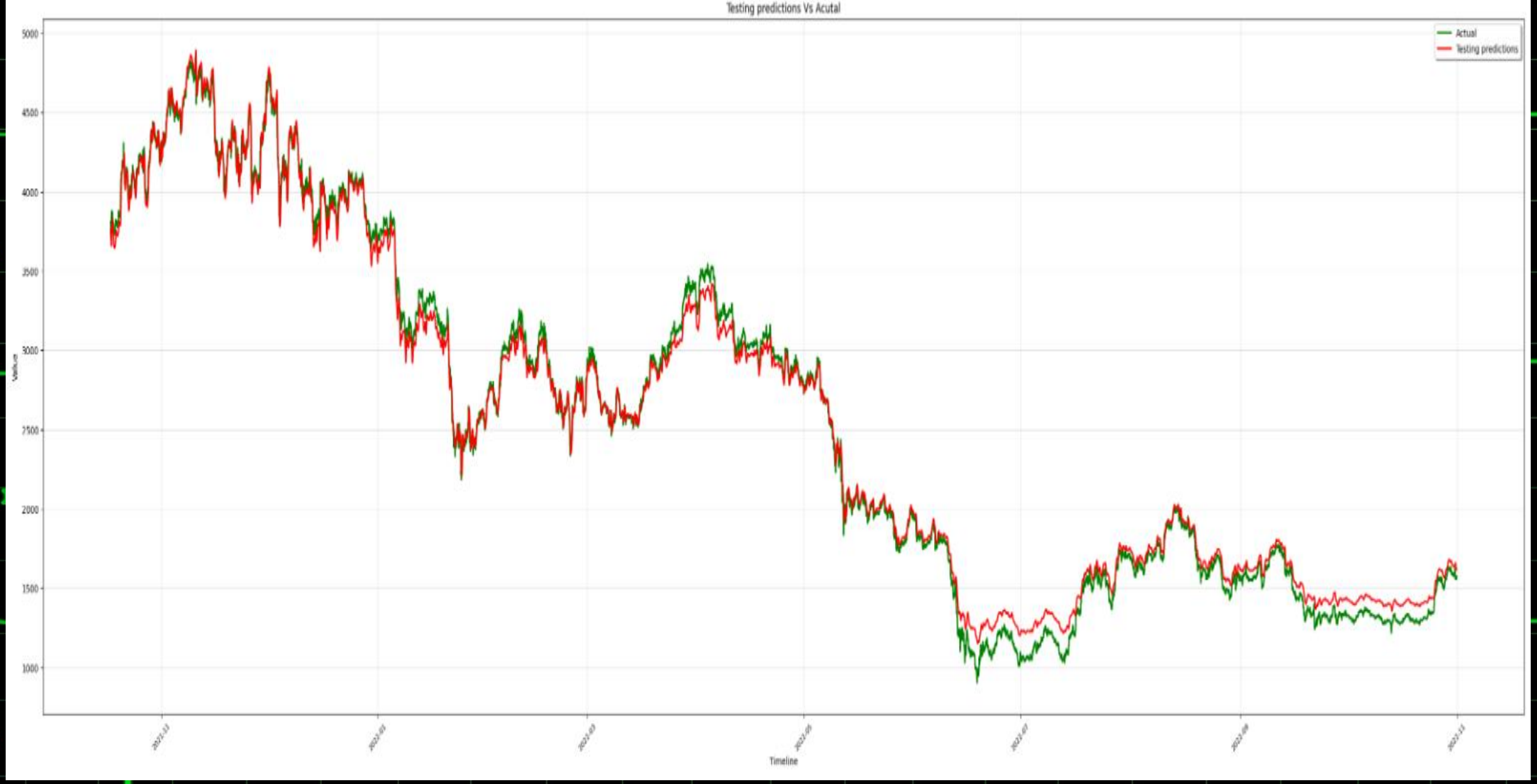


Fig: Line Graph of Prediction on Test Dataset

POSSIBLE FUTURE IMPROVEMENTS

1) Use AWS kinesis to perform data transformation and analysis in **real time**. Currently the Pyspark performs bulk operations on a chunk of data every 10mins and the LSTM model predicts the data every hour. Real-time data ingestion and analytics would help improve this.

2) Making the system even more scalable by using Load balancers (Elastic Load balancer by AWS). Load balancing becomes very important as the number of users increase.

3) While HDFS is a good option for data storage, it favors write once read multiple times approach. This is not suitable for real-time data. Utilization of database like Postgres or AWS Dynamo DB would be a better option in this scenario.

4) MultiChart analysis and Live data

MEET OUR TEAM

Pratyush Pradhan

Ronaj Pradhan

Padam Jung Thapa

THANK YOU