Real-Time Cryptocurrency Analytics Dashboard and Trend Prediction

Course: Advanced Big Data Analytics

Syed Muhammad Abid Raza

24K-8004

FAST-NUCES

1. Introduction

Cryptocurrencies represent a rapidly evolving asset class characterized by high volatility and 24/7 trading activity. Real-time analytics and predictive modeling are crucial for both individual traders and institutional investors to gain insights and make informed decisions. This project presents a real-time cryptocurrency analytics dashboard developed using Dash, with data processing and machine learning capabilities powered by Apache Spark and PySpark MLlib.

2. Objectives

- Collect and visualize real-time cryptocurrency price data.
- Perform real-time data processing using Spark.
- Build and integrate predictive models for trend classification.
- Enable interactive exploration of historical and current trends via a web-based dashboard.

3. Tools and Technologies

- Dash (Plotly) - Web-based dashboard interface
- SQLite - Lightweight local database for data storage
- Apache Spark (PySpark) - Distributed data processing
- MLlib - Spark's machine learning library for classification
- Python - Core language used throughout the project
- Google Colab - Development environment for hosting the project

4. System Architecture

Components:

- Data Collection: Cryptocurrency price data is fetched using a public API and inserted into an SQLite database at regular intervals.
- Data Storage: Prices along with timestamps are stored persistently in SQLite.
- Data Processing: Spark ingests data from SQLite and performs ETL operations.
- Machine Learning Models: Trend Classification using Random Forest.
- Visualization: Real-time graphs, historical trends, and ML outputs

5. Features Implemented

- Real-Time Data Handling
- Real-Time Graphs
- Machine Learning Integration
- Interactive Dashboard (Tabs: Overview, Trend, Prediction, Database View)

6. Machine Learning Models

Price Trend Classification:

- Input: current_price, previous_price
- Feature: Price difference
- Label: 1 (uptrend), 0 (downtrend)
- Model: RandomForestClassifier

7. Methodology

Spark session creation:

```
spark = SparkSession.builder.appName("CoinGeckoStreamingApp").getOrCreate()
# Create a StreamingContext with a batch interval of 1 second
ssc = StreamingContext(spark.sparkContext, 60)
```

Schema for database:

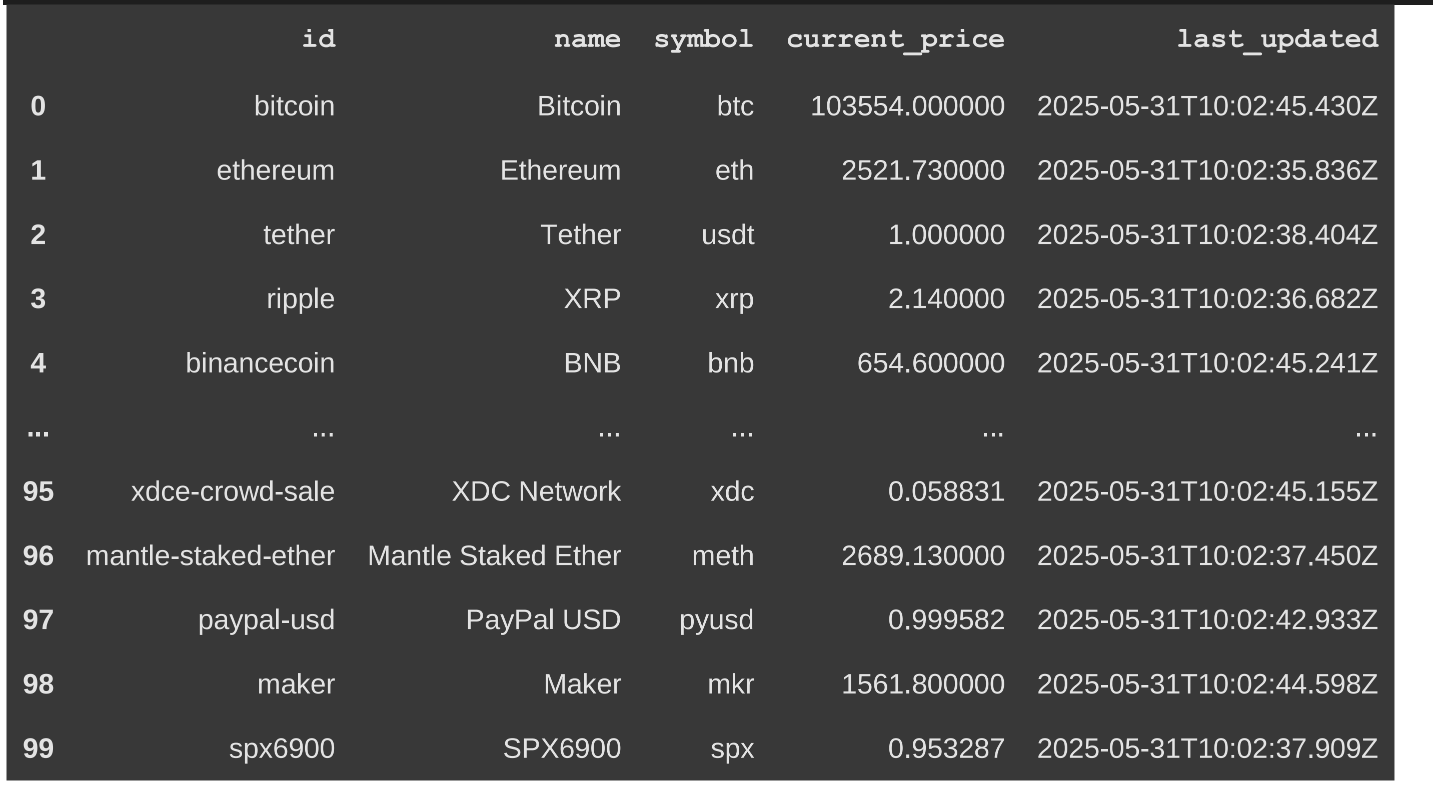
```
schema = StructType([
    StructField("id", StringType(), True),
    StructField("name", StringType(), True),
    StructField("symbol", StringType(), True),
    StructField("current_price", StringType(), True),
    StructField("last_updated", StringType(), True)
])
```

Api url:

```
api_url = "https://api.coingecko.com/api/v3/coins/markets"
```

Converting data from database to dataframe:

```
# Path to your database file
db_path = "/content/coingecko_data.db"

# Connect to the database
conn = sqlite3.connect(db_path)

# Query the table into a Pandas DataFrame
df = pd.read_sql_query("SELECT * FROM coingecko_market", conn)

# Show the first few rows
Df
```



	id	name	symbol	current_price	last_updated
0	bitcoin	Bitcoin	btc	103554.000000	2025-05-31T10:02:45.430Z
1	ethereum	Ethereum	eth	2521.730000	2025-05-31T10:02:35.836Z
2	tether	Tether	usdt	1.000000	2025-05-31T10:02:38.404Z
3	ripple	XRP	xrp	2.140000	2025-05-31T10:02:36.682Z
4	binancecoin	BNB	bnb	654.600000	2025-05-31T10:02:45.241Z
...	...	...	...	...	...
95	xdce-crowd-sale	XDC Network	xdc	0.058831	2025-05-31T10:02:45.155Z
96	mantle-staked-ether	Mantle Staked Ether	meth	2689.130000	2025-05-31T10:02:37.450Z
97	paypal-usd	PayPal USD	pyusd	0.999582	2025-05-31T10:02:42.933Z
98	maker	Maker	mkr	1561.800000	2025-05-31T10:02:44.598Z
99	spx6900	SPX6900	spx	0.953287	2025-05-31T10:02:37.909Z

Dash app creation:

```
app = JupyterDash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])
```

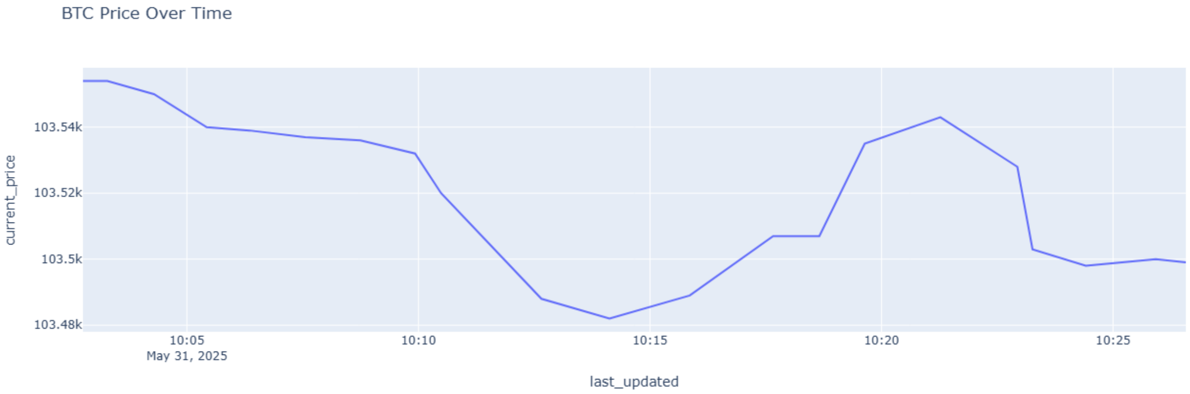
```
app.layout = dbc.Container([
    html.H1("Real-Time Crypto Analytics Dashboard"),
    dcc.Tabs(id="tabs", value='overview', children=[
        dcc.Tab(label='Overview', value='overview'),
        dcc.Tab(label='Trends', value='trends'),
        dcc.Tab(label='Alerts', value='alerts'),
        dcc.Tab(label='Trend Prediction', value='trend'),
    ]),
    html.Br(),
    dcc.Dropdown(id='symbol-select', options=[{'label': s, 'value': s} for s in
symbols], value='btc', clearable=False),
    html.Div(id='tab-content'),
    dcc.Interval(id='refresh', interval=60000, n_intervals=0)
])
```

```
@app.callback(
    Output('tab-content', 'children'),
    Input('tabs', 'value'),
    Input('symbol-select', 'value'),
    Input('refresh', 'n_intervals')
)
def update_price_chart(tab, symbol, n):
    conn = sqlite3.connect('/content/coingecko_data.db')
    df = pd.read_sql_query("SELECT * FROM coingecko_market WHERE symbol=?", conn,
params=(symbol,))
    conn.close()
    # ... Fetching and processing data ...
     # Convert current_price to float for plotting
    df['current_price'] = pd.to_numeric(df['current_price'], errors='coerce')
    df['last_updated'] = pd.to_datetime(df['last_updated'])
    df = add_indicators(df)

    if tab == 'overview':
        fig = px.line(df, x='last_updated', y='current_price',
title=f"{symbol.upper()} Price Over Time")
        return dcc.Graph(figure=fig)

    elif tab == 'trends':
        fig = px.line(df, x='last_updated', y='moving_avg', title=f"{symbol.upper()}
5-Point Moving Average")
        return dcc.Graph(figure=fig)

    elif tab == 'alerts':
        alert_df = df[df['z_score'].abs() > 2]
        if alert_df.empty:
            return html.Div("No price anomalies detected.")
        fig = px.scatter(alert_df, x='last_updated', y='current_price',
color='z_score',
                         title=f"Anomalies in {symbol.upper()} Price (Z-Score > 2)")
        return dcc.Graph(figure=fig)

    elif tab == 'trend':
        sdf = spark.createDataFrame(df[['last_updated', 'current_price']].dropna())
        sdf = sdf.withColumn("price_lag", lag("current_price",
1).over(Window.orderBy("last_updated")))
        sdf = sdf.withColumn("price_change", col("current_price") -
col("price_lag"))
        sdf = sdf.withColumn("label", when(col("price_change") > 0.01, "Rise")
                                          .when(col("price_change") < -0.01, "Drop")
                                          .otherwise("Stable"))

        label_indexer = StringIndexer(inputCol="label", outputCol="label_index",
handleInvalid='keep')
        assembler = VectorAssembler(inputCols=["price_change"],
outputCol="features", handleInvalid='skip')
        classifier = RandomForestClassifier(labelCol="label_index",
featuresCol="features", numTrees=10)
        pipeline = Pipeline(stages=[label_indexer, assembler, classifier])

        train, test = sdf.randomSplit([0.8, 0.2], seed=42)
        model = pipeline.fit(train)
        predictions = model.transform(test).toPandas()

        fig = px.scatter(predictions, x='last_updated', y='price_change',
color='label',
                         title=f"{symbol.upper()} Trend Classification
(Rise/Drop/Stable)")
        return dcc.Graph(figure=fig)
```

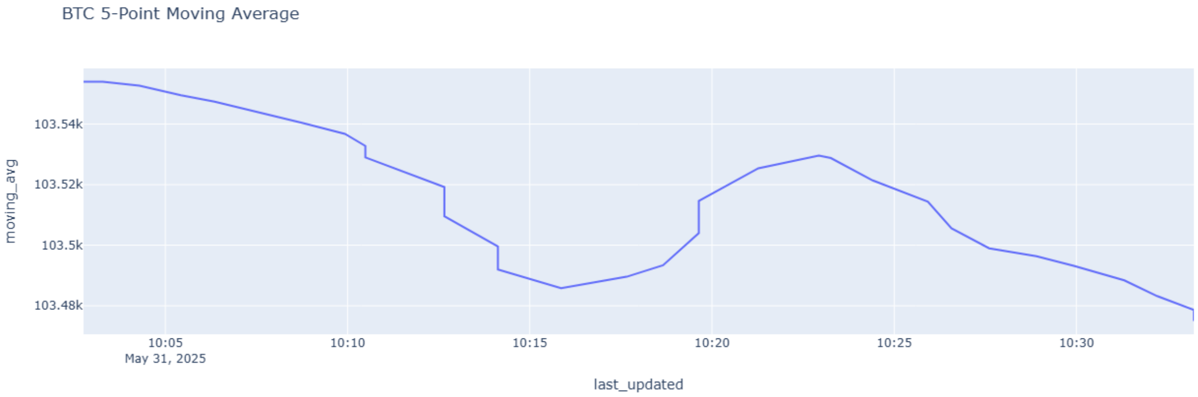
```
app.run(mode='inline', debug=True, port=8050)
```

8. Results

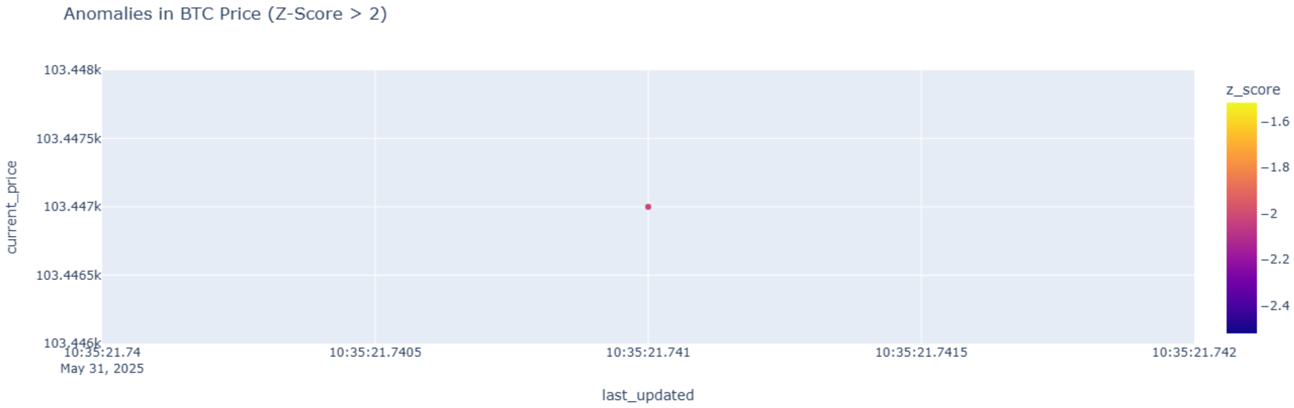
Bitcoin (btc) real-time evaluation:



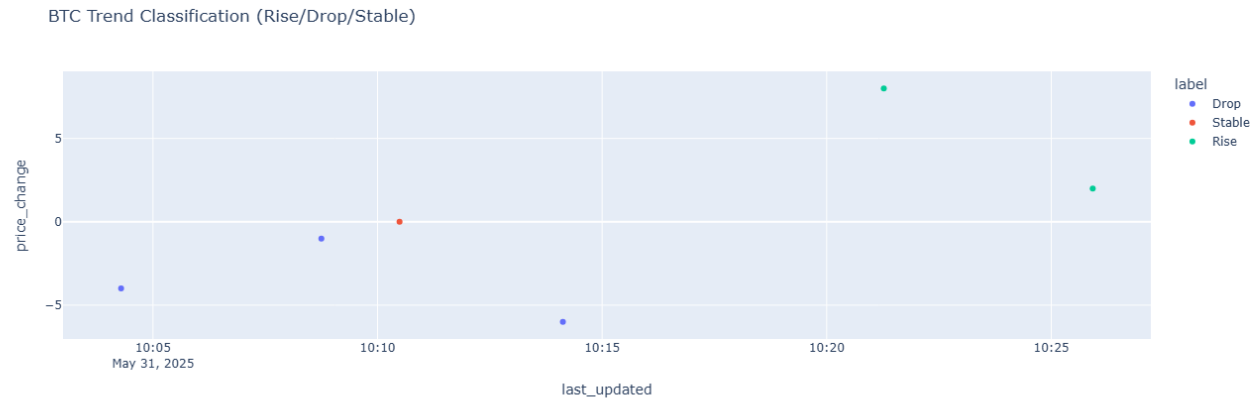
Bitcoin (btc) trends evaluation:



Bitcoin (btc) anamoly alert messages:



Bitcoin (btc) trend prediction:



9. Future Work

- Add filtering options by date/time range
- Implement anomaly detection
- Extend models using LSTM
- Deploy with backend support (PostgreSQL + Docker)

10. Conclusion

This project demonstrates the integration of real-time data ingestion, processing, and machine learning within a user-friendly dashboard. By leveraging Apache Spark and Dash, the system provides a powerful framework for cryptocurrency analytics.

11. References

- Apache Spark Documentation: https://spark.apache.org/docs/latest/
- Plotly Dash: https://dash.plotly.com/
- CoinGecko API: https://www.coingecko.com/en/api
- SQLite Documentation: https://sqlite.org/docs.html