PROJECT REPORT
GROUP – 8

Report submitted by :

Deepak Shinde

PROJECT 1: IPL data Analysis and extract various insights using different graphs.

PROJECT 2 : Advanced Financial fraud detection model with dashboard .

PROJECT 3 : Time series Analysis with cryptocurrency.

1. Introduction

The Indian Premier League (IPL) is one of the most celebrated cricket tournaments in the world. With an immense volume of data generated over the years, analyzing IPL data provides valuable insights into team performance, player statistics, and match outcomes. This project focuses on performing IPL data analysis by connecting PostgreSQL to Power BI and leveraging Python and advanced libraries for data visualization.

The IPL Data Analysis project focuses on extracting valuable insights from IPL match data using various data analytics techniques. By analyzing historical match outcomes, player performances, team comparisons, and venue statistics, the project visualizes trends and patterns through graphs like bar charts, line graphs, and scatter plots. It also explores player career insights, auction analysis, boundary counts, and more. With Python's powerful libraries like Pandas and Matplotlib, this project provides a comprehensive overview of IPL data for in-depth analysis and decision-making.

2. Objective

The primary objective of this project is to extract IPL data stored in a PostgreSQL database, analyze it using Python, and visualize it in Power BI. The key goals include:

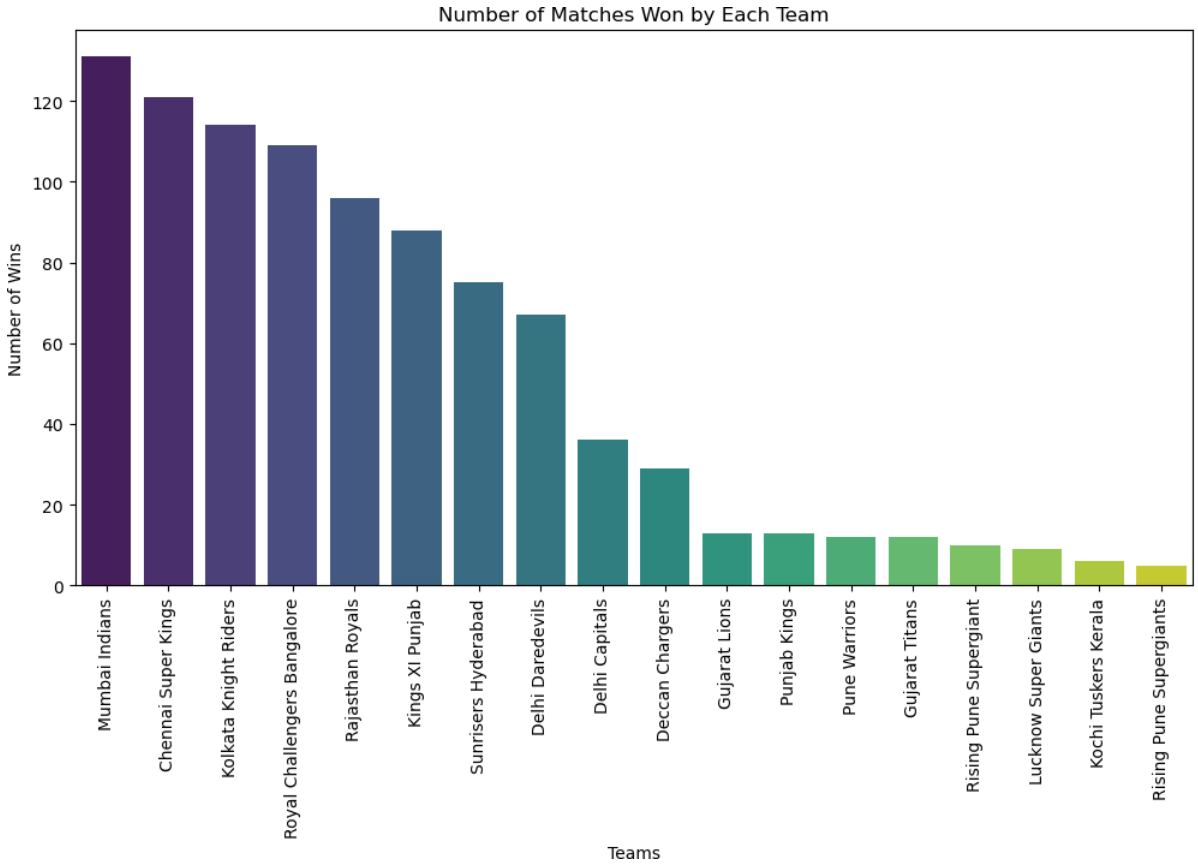
- Efficient data extraction from PostgreSQL into Power BI.
- Utilizing Python for advanced data manipulation and visualization.
- Generating insights through visualizations such as histograms, bar graphs, and pie charts.

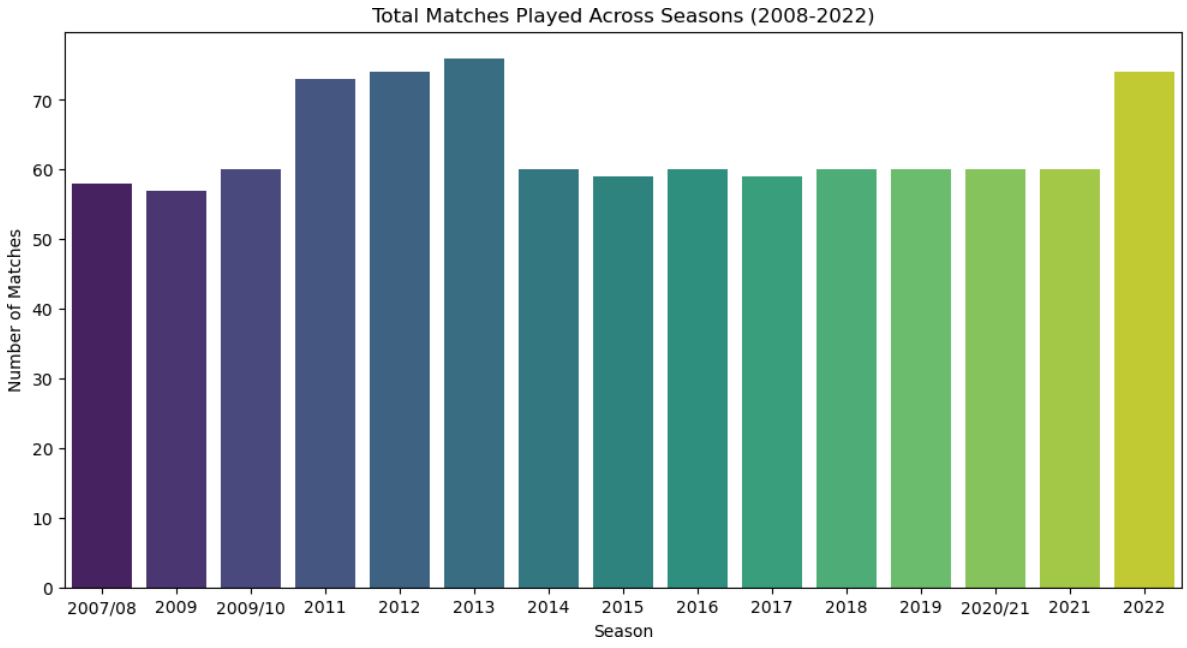
3. Features

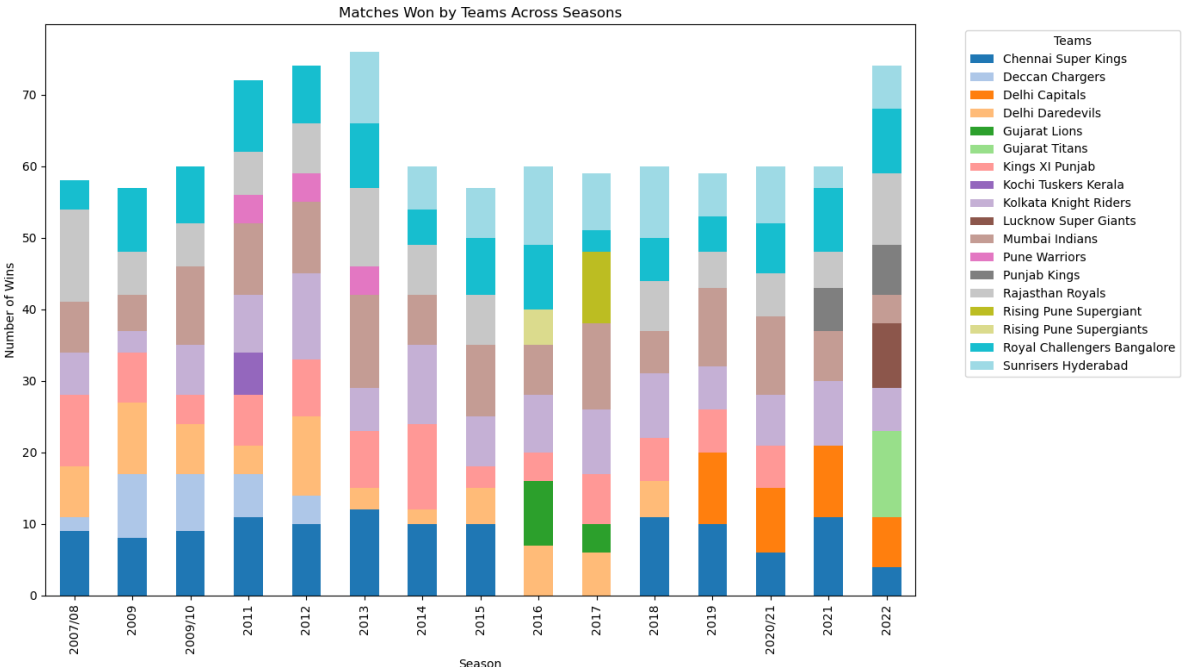
(A) Match Outcome Analysis

* Visualize match outcomes (Win/Loss) across different years .

* Analyze team performance based on historical data and seasonal trends.



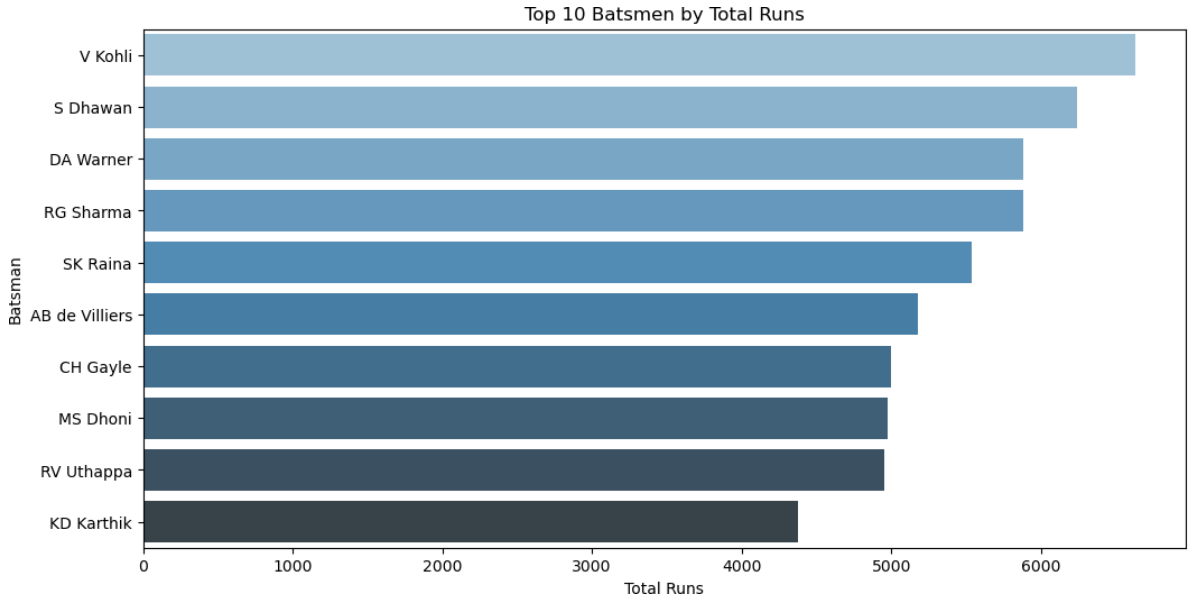


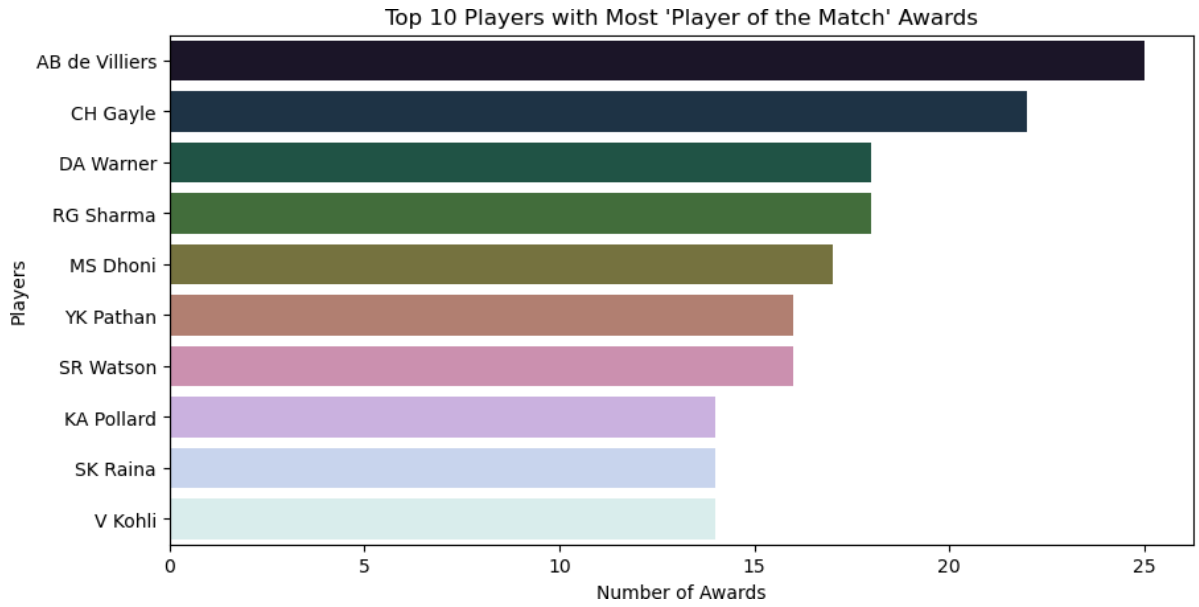


(B) Player Performance

* Track individual player statistics like runs, wickets, and strike rates.

* Use bar graphs and scatter plots to visualize player contributions.

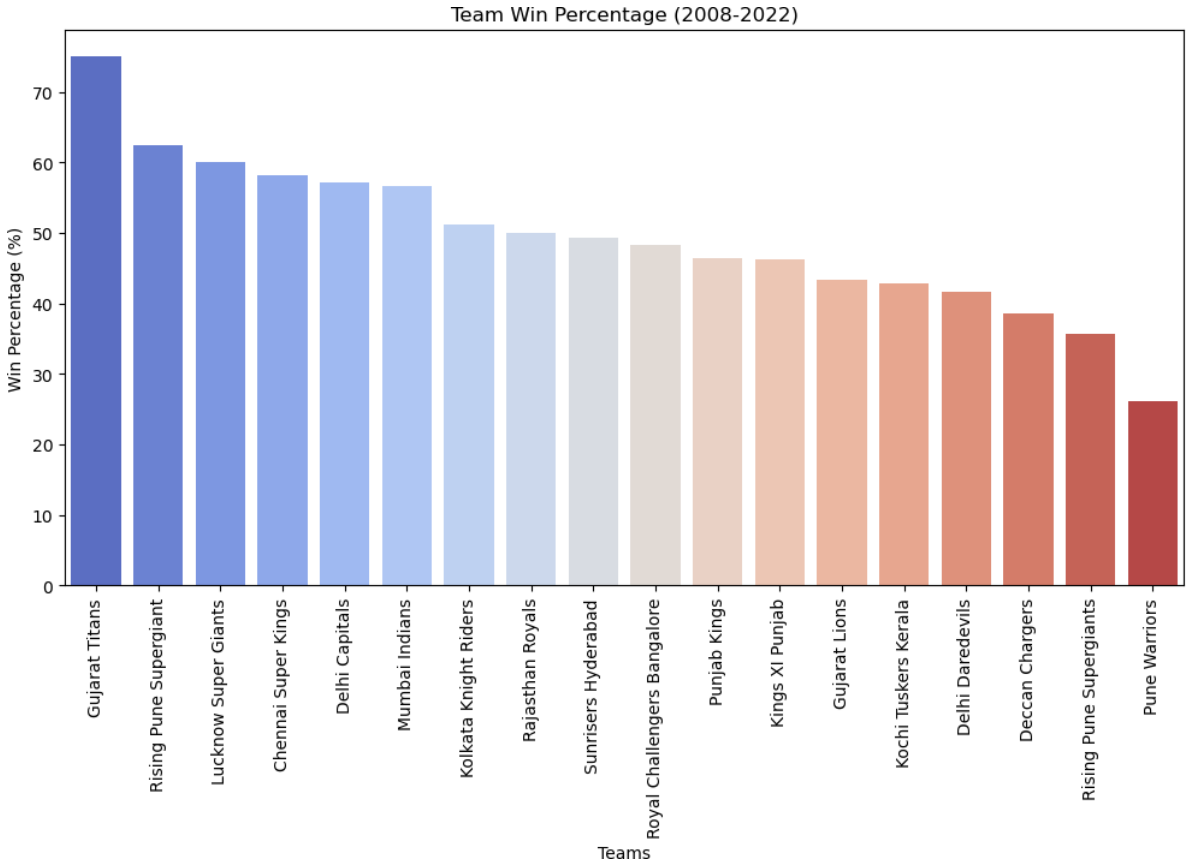




(c) Team Comparison

* Compare team performance using line and pie charts for win percentage.

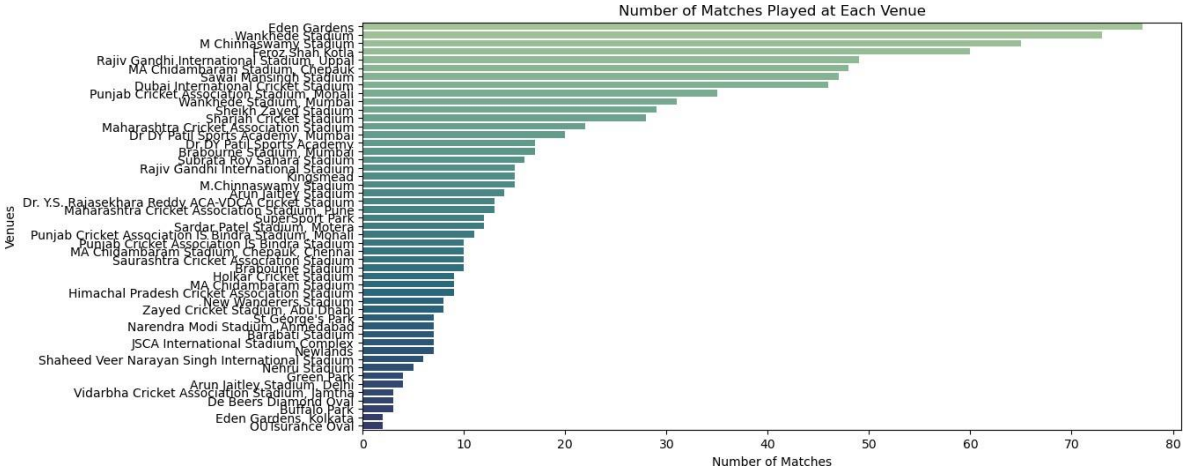
* Analyze the impact of team changes, including player transfers or injuries.



(d) Venue Performance

* Evaluate match outcomes across different IPL venues.

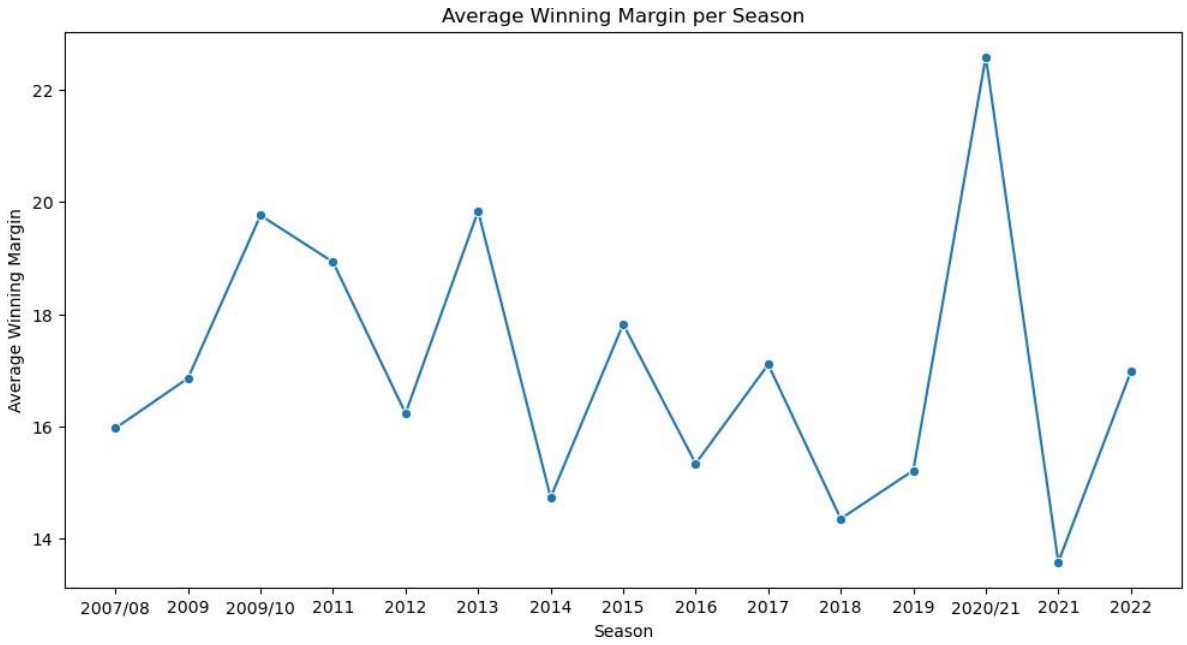
* Present venue-based performance using heatmaps or bar charts.

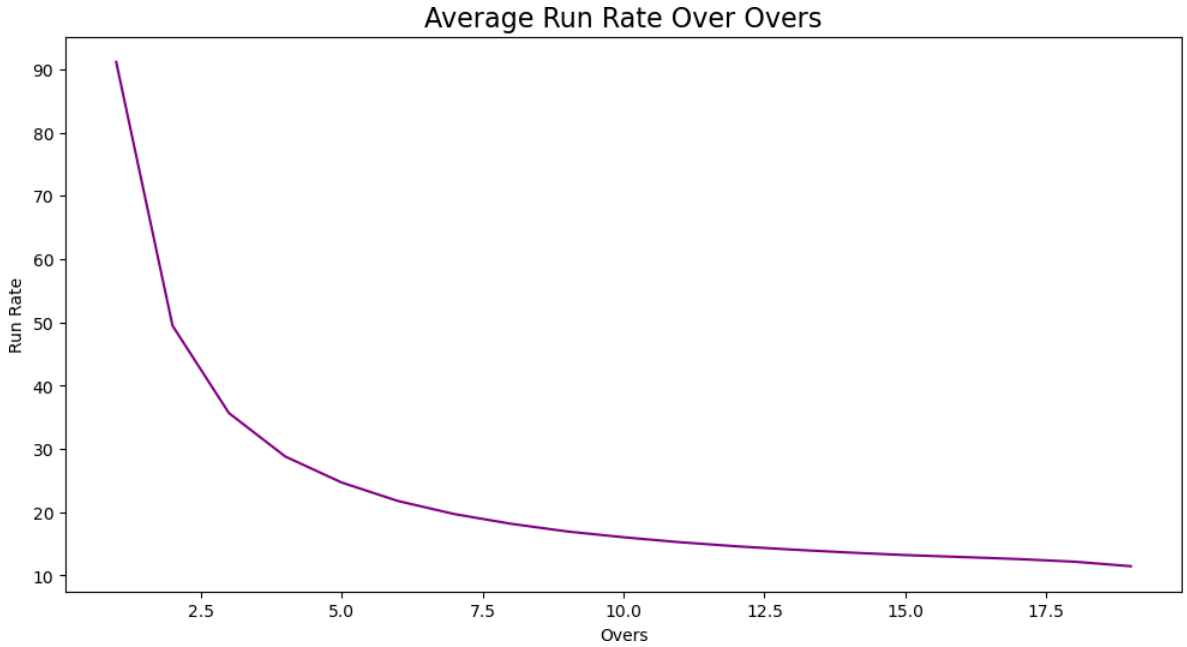


(e) Run Rate & Scoring Analysis

* Visualize average run rates for different teams and players .

* Use line graphs to track scoring trends over the years.





(f) Best Batting Partnership

* Identify and visualize top batting pairs using histograms .

* Analyze successful partnerships based on runs scored and boundary rates.

```
partnerships = deliveries.groupby(['id', 'batter', 'non_striker'])['batsman_run'].sum()
top_partnerships = partnerships.sort_values(ascending=False).head(10).reset_index()
print("Top 10 Partnerships:")
print(top_partnerships)
```

```
Top 10 Partnerships:
       id         batter non_striker  batsman_run
0  1304112       Q de Kock    KL Rahul          140
1   829795  AB de Villiers     V Kohli          133
2   980987  AB de Villiers     V Kohli          129
3   548372       CH Gayle     V Kohli          127
4   598027       CH Gayle  TM Dilshan          127
5  1175366     JM Bairstow   DA Warner          114
6   335994    AC Gilchrist  VVS Laxman          109
7   548363      RG Sharma    HH Gibbs          109
8   548329      DA Warner     NV Ojha          109
9  1216527      MA Agarwal    KL Rahul          106
```

4. Tools and Technologies

- **Database**: PostgreSQL
- **Data Analysis and Visualization**: Python (pandas, matplotlib, seaborn)
- **BI Tool**: Power BI

5. Methodology

5.1 Data Collection

The IPL dataset includes match data, player statistics, and team information. The dataset was loaded into a PostgreSQL database for structured storage and querying.

5.2 Data Extraction

Data was extracted from PostgreSQL using SQL queries. The extracted data was exported to Power BI and Python for analysis.

5.3 Data Analysis in Python

Python was used for advanced data manipulation and visualization. Libraries such as pandas, matplotlib, and seaborn were used to:

- Clean and preprocess data.
- Calculate key statistics.
- Create insightful visualizations.

5.4 Visualization in Power BI

Power BI was connected to PostgreSQL to create dynamic dashboards and visual reports. The following visualizations were generated:

- **Bar Graphs**: Team-wise and player-wise performance.
- **Pie Charts**: Win percentages and contribution of players.
- **Histograms**: Distribution of runs, wickets, and other key metrics.

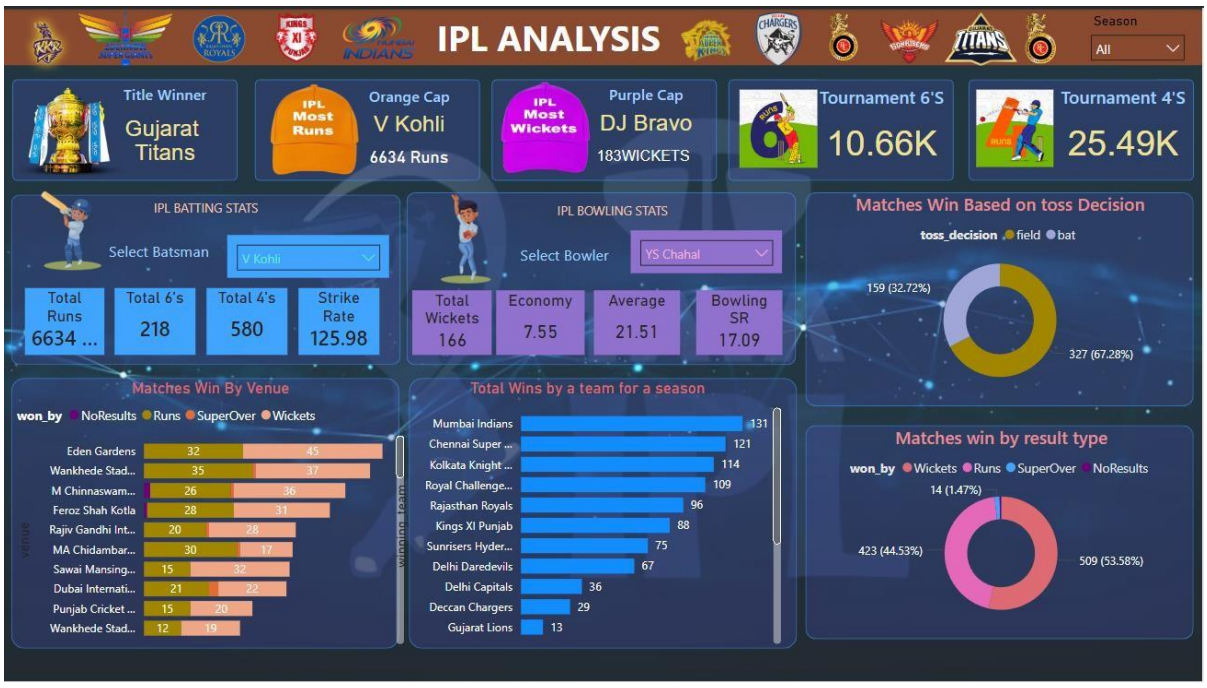
6. Implementation

6.1 Database Connection

PostgreSQL was hosted locally or on a cloud platform. The connection string details, including the database name, username, and password, were used to connect Python and Power BI to the database.

6.2 power BI Connection

Power BI was connected to PostgreSQL using the native PostgreSQL connector. SQL queries were executed directly in Power BI to create tables and perform transformations and created a dashboard.



6.3 Visualization

- **Histograms**: Used matplotlib and seaborn to show the distribution of runs and other metrics.
- **Bar Graphs**: Created in Power BI to depict top-performing players and teams.
- **Pie Charts**: Highlighted the proportion of wins by each team.

6. Results

The analysis provided the following insights:

- Team-wise and player-wise performance trends over the seasons.
- Top-performing players based on metrics such as runs scored, wickets taken, and strike rate.
- Distribution of match outcomes and high-scoring games.

Conclusion

The project successfully demonstrated how PostgreSQL, Python, and Power BI can be integrated to analyze and visualize IPL data. The analysis and visualizations provide valuable insights for fans, analysts, and stakeholders. Future work could include predictive modeling for match outcomes and player performance.

PROJECT 2 : Advanced Financial fraud detection model with dashboard .

Project Overview

Financial fraud has become increasingly sophisticated, requiring advanced detection mechanisms that leverage machine learning and artificial intelligence. This project focuses on building an **Advanced Financial Fraud Detection Model** that identifies suspicious transactions and fraudulent activities using a combination of data analysis, statistical techniques, and machine learning algorithms. Additionally, a **real-time interactive dashboard** will be developed to visualize fraud trends, monitor transactions, and assist analysts in decision-making.

Key Objectives

1. **Develop an AI-driven fraud detection model** that can analyze financial transactions and flag potential fraudulent activities.

2. **Implement machine learning algorithms** such as logistic regression, decision trees, random forests, gradient boosting, and deep learning models to improve detection accuracy.

3. **Design an interactive dashboard** that provides real-time monitoring, alerts, and insights into fraudulent activities.

4. **Enhance the explainability of the model** by incorporating SHAP (SHapley Additive exPlanations) values, feature

importance analysis, and interpretability techniques.

5. **Optimize model performance** by reducing false positives while ensuring a high fraud detection rate.

6. **Ensure data security and compliance** with financial regulations and privacy policies.

Technical Approach

1. Data Collection and Preprocessing

- **Data Sources**:

- Transaction data from banks and financial institutions.

- Customer profile data, past fraud records, and account behavior.

- External sources such as credit scores and cybersecurity databases.

- **Data Cleaning & Transformation**:

  - Handling missing values and outliers.

  - Encoding categorical variables.

  - Feature scaling and normalization.

- **Feature Engineering**:

- Creating new features like transaction velocity, location mismatch, and anomalous spending patterns.

- Using historical fraud data to generate fraud probability scores.

2. Machine Learning & Deep Learning Models

- **Supervised Learning Models**:

  - Logistic Regression, Decision Trees, Random Forest, XGBoost, and Support Vector Machines (SVM).

- **Unsupervised Learning Models** (for anomaly detection):

  - Autoencoders, Isolation Forests, and

One-Class SVM.

- **Deep Learning Models**:
    - LSTM (Long Short-Term Memory) networks for time-series fraud detection.
    - Graph Neural Networks for detecting fraud rings and collusion.
- **Model Training & Evaluation**:
    - Splitting dataset into training, validation, and test sets.
    - Using metrics such as precision, recall, F1-score, AUC-ROC, and confusion matrix to evaluate performance.

3. Fraud Detection & Alerting System

- **Real-time fraud detection pipeline** using streaming data (Kafka, Spark, or Flink).
- **Threshold-based fraud alert system** for high-risk transactions.
- **Automated reporting & investigation workflow** for flagged transactions.

4. Interactive Dashboard Development

- **Technology Stack**:
    - Frontend: React.js / Angular / Vue.js for UI.

- Backend: Flask / Django / FastAPI for API development.

- Database: PostgreSQL / MongoDB for storing transaction records.

- Visualization: Power BI, Tableau, or D3.js for fraud analysis.

- **Dashboard Features**:

  - **Live Transaction Monitoring**: Shows real-time transactions with fraud probability scores.

  - **Fraud Heatmaps**: Highlights fraud hotspots based on geographic locations.

- **Customer Risk Analysis**: Displays user risk scores based on transaction history.

- **Anomaly Detection Graphs**: Visual representation of unusual spending patterns.

- **Case Management System**: Allows investigators to analyze flagged transactions.

Deployment & Scalability

- **Cloud-based deployment** using AWS, Azure, or Google Cloud.

- **Microservices architecture** for modularity and scalability.

- **Containerization** using Docker and Kubernetes for deployment in production.

- **CI/CD pipeline** for continuous updates and model retraining.

Security & Compliance

- **Data Encryption** to protect sensitive customer information.

- **Role-Based Access Control (RBAC)** for secure dashboard access.

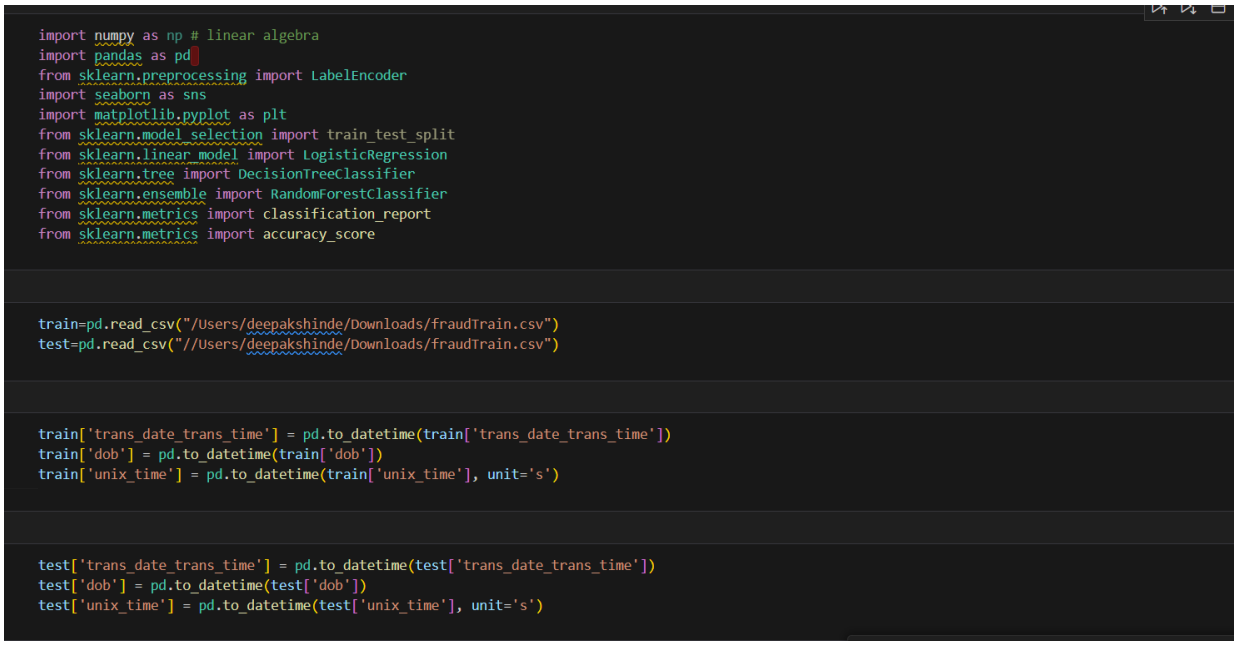
- **Compliance with Financial Regulations** such as GDPR, PCI DSS, and AML guidelines.

Expected Outcomes

- **Higher fraud detection rates** with minimal false positives.
- **Improved operational efficiency** for fraud investigators.
- **Real-time fraud tracking and reporting** with intuitive dashboards.
- **Scalable and adaptive fraud detection system** that evolves with changing fraud

tactics.

This project provides a **cutting-edge solution** to financial fraud detection by integrating **AI, machine learning, real-time monitoring, and advanced visualization** into a single platform.



```
import numpy as np # linear algebra
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

```
train=pd.read_csv("/Users/deepakshinde/Downloads/fraudTrain.csv")
test=pd.read_csv("//Users/deepakshinde/Downloads/fraudTrain.csv")
```

```
train['trans_date_trans_time'] = pd.to_datetime(train['trans_date_trans_time'])
train['dob'] = pd.to_datetime(train['dob'])
train['unix_time'] = pd.to_datetime(train['unix_time'], unit='s')
```

```
test['trans_date_trans_time'] = pd.to_datetime(test['trans_date_trans_time'])
test['dob'] = pd.to_datetime(test['dob'])
test['unix_time'] = pd.to_datetime(test['unix_time'], unit='s')
```

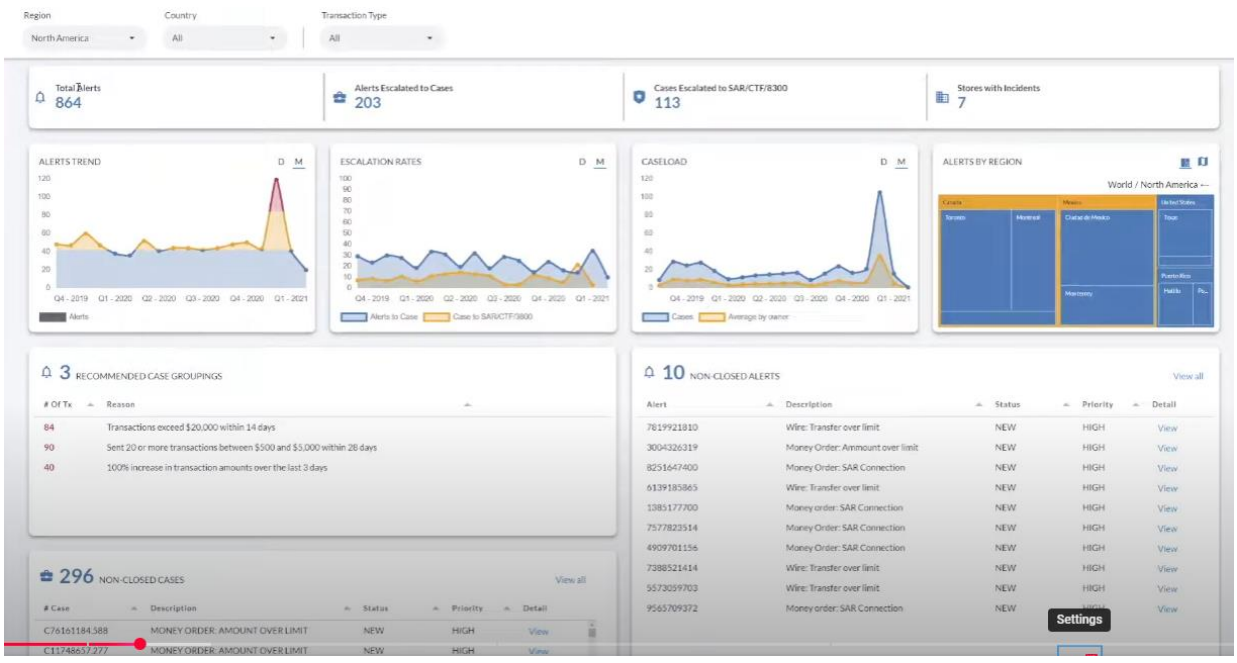
```python
train['trans_date_trans_time'] = pd.to_datetime(train['trans_date_trans_time'])
test['trans_date_trans_time'] = pd.to_datetime(test['trans_date_trans_time'])

# Extract useful features like year, month, day, hour, minute
train['year'] = train['trans_date_trans_time'].dt.year
train['month'] = train['trans_date_trans_time'].dt.month
train['day'] = train['trans_date_trans_time'].dt.day
train['hour'] = train['trans_date_trans_time'].dt.hour
train['minute'] = train['trans_date_trans_time'].dt.minute

test['year'] = test['trans_date_trans_time'].dt.year
test['month'] = test['trans_date_trans_time'].dt.month
test['day'] = test['trans_date_trans_time'].dt.day
test['hour'] = test['trans_date_trans_time'].dt.hour
test['minute'] = test['trans_date_trans_time'].dt.minute

# Drop the original 'trans_date_trans_time' column
train = train.drop(columns=['trans_date_trans_time'])
test = test.drop(columns=['trans_date_trans_time'])
```

```python
categorical_cols = [col for col in train.columns if train[col].dtype == 'object']
```



PROJECT 3 : Time series Analysis with cryptocurrency.

Project Overview

Cryptocurrency markets are highly volatile, with price fluctuations driven by a combination of market sentiment, macroeconomic factors, and speculative trading. This project focuses on applying **time series analysis techniques** to cryptocurrency data to understand trends, detect anomalies, and build predictive models for price forecasting.

The project involves collecting historical cryptocurrency price data, preprocessing it, and applying various **statistical and machine learning models** to analyze trends, volatility, seasonality, and anomalies. Additionally, an **interactive dashboard** will be developed to

visualize insights, forecast trends, and provide actionable insights for traders and investors.

Key Objectives

1. **Analyze historical cryptocurrency price data** using time series techniques.

2. **Identify trends, seasonality, and anomalies** in the market.

3. **Build predictive models** for cryptocurrency price forecasting.

4. **Develop an interactive dashboard** to visualize price movements, volatility, and forecasts.

5. **Evaluate and compare traditional statistical models with deep learning approaches.**

6. **Enhance risk management strategies** by analyzing volatility and market crashes.

Technical Approach

1. Data Collection & Preprocessing

- **Data Sources**:
    - Historical price data from sources like CoinGecko, Binance API, and CoinMarketCap.

- Market indicators such as trading volume, market capitalization, and order book data.

- External factors like global financial news, social media sentiment (Twitter, Reddit), and Google Trends.

- **Preprocessing Steps**:

  - Handling missing values and outliers.

  - Converting timestamps into a structured format.

  - Normalizing and scaling numerical features.

  - Feature engineering (e.g., moving

averages, RSI, MACD, Bollinger Bands).

---

2. Exploratory Data Analysis (EDA)

- **Trend Analysis**: Identifying long-term upward or downward trends.
- **Seasonality Detection**: Examining periodic fluctuations in prices.
- **Volatility Analysis**: Measuring market fluctuations using metrics like standard deviation, ATR, and GARCH models.
- **Correlation Analysis**: Understanding relationships between different cryptocurrencies (e.g., BTC-ETH

correlation).

- **Anomaly Detection**: Identifying unusual price spikes or crashes using statistical methods like Z-score and IQR.

---

3. Time Series Modeling & Forecasting

Traditional Time Series Models

- **Moving Averages (MA)**: Smoothing price trends for better visualization.

- **Autoregressive Integrated Moving Average (ARIMA)**: Predicting future prices based on past values.

- **Seasonal Decomposition of Time Series (STL)**: Separating trends, seasonality, and residuals.

- **GARCH (Generalized Autoregressive Conditional Heteroskedasticity)**: Modeling volatility and risk in cryptocurrency prices.

Machine Learning Approaches

- **Random Forest & Gradient Boosting Models**: Capturing complex relationships in price movements.

- **Support Vector Regression (SVR)**: Effective for non-linear price prediction.

Deep Learning Models

- **Long Short-Term Memory (LSTM) Networks**: Capturing long-term dependencies in price sequences.

- **Transformers (e.g., Temporal Fusion Transformer)**: More advanced sequence modeling techniques.

- **Convolutional Neural Networks (CNN) for Time Series**: Detecting patterns in cryptocurrency charts.

Reinforcement Learning for Crypto Trading

- Developing an agent-based model to optimize trading strategies.

- Using deep Q-learning and policy gradient methods for portfolio management.

---

4. Interactive Dashboard Development

- **Technology Stack**:
  - **Frontend**: React.js, Angular, or Vue.js for a dynamic UI.
  - **Backend**: Flask, FastAPI, or Django for API development.
  - **Database**: PostgreSQL or MongoDB to store historical and real-time data.
  - **Visualization**: D3.js, Power BI, or Tableau for interactive charts.

- **Dashboard Features**:
    - **Live Crypto Price Tracking**: Real-time price updates with market cap and volume.
    - **Historical Data Visualization**: Time series plots of cryptocurrency prices.
    - **Trend & Volatility Indicators**: Displaying moving averages, Bollinger Bands, and RSI.
    - **Price Prediction Module**: Integrating machine learning forecasts.
    - **Anomaly Detection Alerts**: Highlighting sudden market movements.

---

5. Deployment & Scalability

- **Cloud Deployment**: AWS, GCP, or Azure for scalable infrastructure.

- **Real-Time Data Processing**: Using Kafka, Spark Streaming, or WebSockets for live updates.

- **Model Retraining Pipeline**: Automating model updates with new data.

---

6. Security & Compliance

- **Data Encryption**: Ensuring secure API communication.

- **Role-Based Access Control (RBAC)**: Restricting sensitive financial data access.

- **Compliance with Financial Regulations**: Following guidelines for responsible AI in finance.

Expected Outcomes

- **Improved accuracy in cryptocurrency price forecasting.**

- **Better understanding of crypto market volatility and trends.**

- **Enhanced decision-making for traders and investors.**

- **Real-time anomaly detection to flag market crashes.**

- **A robust interactive dashboard for market analysis and visualization.**

This project combines **time series analytics, machine learning, and deep learning** to build a powerful **cryptocurrency market prediction and analysis tool**. 🚀

```python
import pandas as pd
import numpy as np
import requests
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import streamlit as st

# Step 1: Fetch Cryptocurrency Data (Bitcoin Example)
def fetch_crypto_data(symbol='BTC', currency='USD', days=365):
    url = f"https://api.coingecko.com/api/v3/coins/{symbol}/market_chart?vs_currency={curr
    response = requests.get(url)
    data = response.json()
    df = pd.DataFrame(data['prices'], columns=['timestamp', 'price'])
    df['date'] = pd.to_datetime(df['timestamp'], unit='ms')
    df.set_index('date', inplace=True)
    df.drop(columns=['timestamp'], inplace=True)
    return df
```