# 

# Project Report

## Cryptocurrency Portfolio Manager with Price Prediction using Machine Learning

**Name:** Ahmed Javed

**Roll Number:** 055

**Course Title:** Artificial Intelligence

**Submitted By:** Sir Rasik

**Institution Name:** Superior University Lahore

## Abstract

This project aims to provide an integrated platform to manage cryptocurrency investments and predict future prices using machine learning. It utilizes real-time data from the CoinGecko API and employs a linear regression model to forecast cryptocurrency prices for the next day. The tool allows users to track their portfolio's value, calculate profit/loss for each asset, and visualize historical price trends alongside predicted prices.

## Objectives

1. Portfolio Management: Enable users to input and track their cryptocurrency holdings, including quantity and purchase price.

2. Real-Time Valuation: Calculate the total portfolio value and individual asset profit/loss based on real-time prices.

3. Price Prediction: Implement a machine learning model to predict the next day's cryptocurrency price using historical data.

4. Visualization: Provide clear and intuitive graphs for historical prices and predictions to help users make informed decisions.

## Technologies and Libraries Used

1. Programming Language: Python

2. APIs: CoinGecko API for fetching historical and real-time cryptocurrency data.

3. Libraries:

- Data Manipulation: pandas, numpy

- Visualization: matplotlib

- Machine Learning: scikit-learn

- Date and Time Handling: datetime, pandas

## Project Features

1. User-Friendly Input:

- Users can enter details of their cryptocurrency portfolio interactively.

- Options to add multiple cryptocurrencies with their respective quantities and purchase prices.

2. Real-Time Price Fetching:

- Fetches the current price of cryptocurrencies using the CoinGecko API.

- Calculates the profit/loss for each asset and the overall portfolio.

3. Historical Data Collection:

- Retrieves the last 60 days of historical price data for selected cryptocurrencies.

4. Price Prediction:

- Uses Linear Regression to model historical price trends.

- Predicts the price for the next day based on the regression model.

5. Visualization:

- Plots historical prices and highlights the predicted price for the next day.

- Enhances data understanding with intuitive charts.

## Implementation Details

1. **Portfolio Management:**

- A dictionary structure is used to store user input for cryptocurrencies, their quantities, and purchase prices. - Calculation of real-time portfolio value and profit/loss is performed by multiplying current prices with quantities and comparing against purchase prices.

2. **Data Handling:**

- pandas is used to handle API responses and transform historical data into a DataFrame for analysis. - Dates are converted into Julian dates for regression modeling.

3. **Machine Learning Model:**

- Algorithm: Linear Regression from scikit-learn. - Training: The model is trained using the historical price data (date as the feature and price as the target). - Prediction: The trained model forecasts the price for a specified future date.

4. **Visualization:**

- matplotlib generates a graph displaying actual historical prices and the predicted price for the next day.

## Results and Output

1. **Portfolio Analysis:**

- Displays total portfolio value and detailed profit/loss for each cryptocurrency.

2. **Price Prediction:**

- Predicts the price for the selected cryptocurrency for the next day.

- Demonstrates the capability of using machine learning for financial forecasting.

3. **Graphical Representation:**

- Provides users with a visual comparison of historical prices and predicted prices.

## Challenges and Limitations

1. **Data Dependency:**

- The accuracy of predictions depends on the quality and availability of historical data.

2. **Model Limitation:**

- Linear Regression may not capture complex price patterns, especially in volatile markets like cryptocurrencies.

3. **API Constraints:**

- The CoinGecko API has rate limits, which could affect data retrieval for multiple cryptocurrencies.

4. **Assumptions:**

- The model assumes linear trends, which may not always reflect actual market behavior.

## Future Enhancements

1. **Advanced Modeling:**

- Implement more sophisticated machine learning algorithms like LSTM (Long Short-Term Memory) for better time-series predictions.

2. **User Interface:**

- Develop a graphical user interface (GUI) for improved usability.

3. **Portfolio Optimization:**

- Introduce features to recommend buy/sell strategies based on market trends.

4. **Multi-Currency Support:**

- Extend support to manage portfolios across multiple fiat and crypto pairs.

## Conclusion

This project successfully integrates real-time data retrieval, portfolio management, and machine learning to deliver a tool that aids cryptocurrency investors. Despite its limitations, it demonstrates how machine learning can be used for financial forecasting and decision-making. Future improvements could enhance accuracy and functionality, making it a more robust investment tool.

## Appendices

1. **Code:**
2. import requests
3. import pandas as pd
4. import numpy as np
5. import matplotlib.pyplot as plt
6. from sklearn.linear\_model import LinearRegression
7. from datetime import datetime
8. def get\_historical\_data(crypto\_id):
9. url = f"https://api.coingecko.com/api/v3/coins/{crypto\_id}/market\_chart?vs\_currency=usd&days=60"
10. response = requests.get(url)
11. if response.status\_code == 200:
12. data = response.json()
13. if 'prices' in data and data['prices']:
14. prices = pd.DataFrame(data['prices'], columns=['timestamp', 'price'])
15. prices['date'] = pd.to\_datetime(prices['timestamp'], unit='ms')
16. return prices[['date', 'price']]
17. else:
18. print(f"Error: 'prices' key not found or empty for {crypto\_id}.")
19. return pd.DataFrame()
20. else:
21. print(f"Error: API request failed for {crypto\_id}.")
22. return pd.DataFrame()
23. def get\_real\_time\_price(crypto\_id):
24. url = f'https://api.coingecko.com/api/v3/simple/price?ids={crypto\_id}&vs\_currencies=usd'
25. response = requests.get(url)
26. data = response.json()
27. return data[crypto\_id]['usd']
28. def calculate\_portfolio\_value(portfolio):
29. total\_value = 0
30. profit\_loss = {}
31. for crypto, data in portfolio.items():
32. current\_price = get\_real\_time\_price(crypto.lower())
33. value = data['quantity'] \* current\_price
34. pnl = (current\_price - data['purchase\_price']) \* data['quantity']
35. pnl\_type = "Profit" if pnl > 0 else "Loss"
36. percentage\_pnl = (pnl / (data['purchase\_price'] \* data['quantity'])) \* 100
37. profit\_loss[crypto] = {"type": pnl\_type, "amount": pnl, "percentage": percentage\_pnl}
38. total\_value += value
39. return total\_value, profit\_loss
40. def prepare\_data\_for\_prediction(prices):
41. prices['date'] = prices['date'].map(pd.Timestamp.to\_julian\_date)
42. X = prices['date'].values.reshape(-1, 1)
43. y = prices['price'].values
44. return X, y
45. def train\_price\_prediction\_model(crypto\_id):
46. prices = get\_historical\_data(crypto\_id)
47. X, y = prepare\_data\_for\_prediction(prices)
48. model = LinearRegression()
49. model.fit(X, y)
50. return model
51. def predict\_next\_day\_price(model, future\_date):
52. future\_date\_julian = np.array([future\_date.to\_julian\_date()]).reshape(-1, 1)
53. predicted\_price = model.predict(future\_date\_julian)
54. return predicted\_price[0]
55. def plot\_portfolio\_and\_prediction(prices, predicted\_price, future\_date, crypto\_id):
56. plt.figure(figsize=(10, 5))
57. plt.plot(prices['date'], prices['price'], label='Actual Price')
58. plt.scatter(future\_date, predicted\_price, color='red', label='Predicted Price for Next Day')
59. plt.xlabel('Date')
60. plt.ylabel('Price (USD)')
61. plt.title(f'{crypto\_id.capitalize()} Price Prediction')
62. plt.legend()
63. plt.show()
64. def get\_user\_portfolio():
65. portfolio = {}
66. while True:
67. crypto = input("Enter cryptocurrency name (e.g., bitcoin, ethereum) or type 'done' to finish: ").lower()
68. if crypto == 'done':
69. break
70. try:
71. quantity = float(input(f"Enter quantity of {crypto.capitalize()}: "))
72. purchase\_price = float(input(f"Enter purchase price of {crypto.capitalize()}: "))
73. portfolio[crypto] = {'quantity': quantity, 'purchase\_price': purchase\_price}
74. except ValueError:
75. print("Invalid input! Please enter numeric values for quantity and purchase price.")
76. return portfolio
77. def run\_project():
78. portfolio = get\_user\_portfolio()
79. if not portfolio:
80. print("No portfolio added. Exiting...")
81. return
82. total\_value, profit\_loss = calculate\_portfolio\_value(portfolio)
83. print(f"\nTotal Portfolio Value: ${total\_value:.2f}")
84. for crypto, pnl in profit\_loss.items():
85. print(f"{crypto.capitalize()}: {pnl['type']} = ${pnl['amount']:.2f} ({pnl['percentage']:.2f}%)")
86. if len(portfolio) == 1:
87. selected\_crypto = list(portfolio.keys())[0]
88. print(f"\nShowing price prediction for {selected\_crypto.capitalize()}...")
89. model = train\_price\_prediction\_model(selected\_crypto)
90. future\_date = pd.to\_datetime("today") + pd.DateOffset(days=1)
91. predicted\_price = predict\_next\_day\_price(model, future\_date)
92. print(f"\nPredicted {selected\_crypto.capitalize()} price for tomorrow: ${predicted\_price:.2f}")
93. prices = get\_historical\_data(selected\_crypto)
94. plot\_portfolio\_and\_prediction(prices, predicted\_price, future\_date, selected\_crypto)
95. return
96. print("\nSelect a cryptocurrency from your portfolio to see the price prediction:")
97. for idx, crypto in enumerate(portfolio.keys(), start=1):
98. print(f"{idx}. {crypto.capitalize()}")
99. choice = input("Enter the number of the cryptocurrency to see the chart: ").lower()
100. try:
101. choice = int(choice) - 1
102. selected\_crypto = list(portfolio.keys())[choice]
103. print(f"\nShowing price prediction for {selected\_crypto.capitalize()}...")
104. model = train\_price\_prediction\_model(selected\_crypto)
105. future\_date = pd.to\_datetime("today") + pd.DateOffset(days=1)
106. predicted\_price = predict\_next\_day\_price(model, future\_date)
107. print(f"\nPredicted {selected\_crypto.capitalize()} price for tomorrow: ${predicted\_price:.2f}")
108. prices = get\_historical\_data(selected\_crypto)
109. plot\_portfolio\_and\_prediction(prices, predicted\_price, future\_date, selected\_crypto)
110. except (ValueError, IndexError):
111. print("Invalid selection. Exiting program.")
112. if \_\_name\_\_ == "\_\_main\_\_":
113. run\_project()

2. **Data Sources:**

- CoinGecko API documentation: [https://www.coingecko.com/en/api](https://www.coingecko.com/en/api)