MSA 8200 PREDICTIVE ANALYTICS
FINAL PROJECT REPORT (GROUP 16)
BITCOIN PRICE PREDICTION

ABSTRACT

This report investigates the effectiveness of three statistical models, linear regression, polynomial regression, and SARIMAX, in predicting Bitcoin prices. The study uses historical Bitcoin price data from 2015 to 2022 and compares the accuracy of the models in predicting future prices. Linear regression and polynomial regression were used to fit a straight line and polynomial curve, respectively, to the Bitcoin price data, while SARIMAX model was employed to capture the time series dynamics and forecast Bitcoin prices. The study provides valuable insights into the potential of using statistical models to predict Bitcoin prices, which can inform investment decisions in the cryptocurrency market.

INTRODUCTION

Bitcoin, the world's first decentralized cryptocurrency, has gained massive popularity and adoption since its inception in 2009. With its unique characteristics, including limited supply, decentralization, and pseudonymity, Bitcoin has become a popular investment asset, with its price being subject to significant volatility. As a result, the ability to accurately predict Bitcoin prices has become a crucial concern for investors and traders in the cryptocurrency market.

Bitcoin price prediction refers to the process of forecasting the future price of Bitcoin using various techniques, including statistical models, machine learning algorithms, and sentiment analysis. The goal of Bitcoin price prediction is to provide investors and traders with insights into potential future price movements, which can inform investment decisions and trading strategies.

The field of Bitcoin price prediction has gained significant attention in recent years, with various studies investigating the effectiveness of different techniques in forecasting Bitcoin prices. These techniques include linear regression, polynomial regression, SARIMAX models, deep learning algorithms, and more. As the cryptocurrency market continues to evolve, the ability to predict Bitcoin prices accurately will become increasingly crucial for successful investment and trading in the market.

PROBLEM STATEMENT

The growing popularity and adoption of Bitcoin as an investment asset, mainly because they offer several potential benefits, such as increased privacy, security, and accessibility. Its price remains highly volatile, making it challenging for investors and traders to make informed investment decisions. Our aim is to predict the future prices of Bitcoin using historical data. We have collected data for Bitcoin prices and want to use this data to train models that can predict the future prices of Bitcoin. We compare the performance of Linear Regression, Polynomial Regression, and SARIMAX models to determine which method is most effective for predicting the future prices of Bitcoin.

CONTRIBUTIONS

Our contributions are as follows:

In this research study, we investigated the effectiveness of three different methods for Bitcoin price prediction: Linear Regression, Polynomial Regression, and SARIMAX. We utilized historical Bitcoin price data spanning from 2015 to 2022, and the goal was to determine which model produced the most accurate forecasts for future Bitcoin prices.

To evaluate the performance of these models, we used various metrics such as Root Mean Squared Error (RMSE), R-squared and Adjusted R-Squared. These metrics provided a quantitative assessment of the models' accuracy in predicting future Bitcoin prices. The R-squared value measures the goodness of fit of the model to the data, with higher values indicating a better fit. Adjusted R-squared, on the other hand, considers the number of variables in the model and penalizes the model for adding variables that do not improve its performance.

Our results showed that the Regression models outperformed SARIMAX in terms of accuracy. The Regression models had the lower RMSE values, indicating that it had the smallest average difference between predicted and actual Bitcoin prices. Additionally, the Regression models had higher R-squared and Adjusted R-Squared values, indicating that they had a better fit to the historical Bitcoin price data.

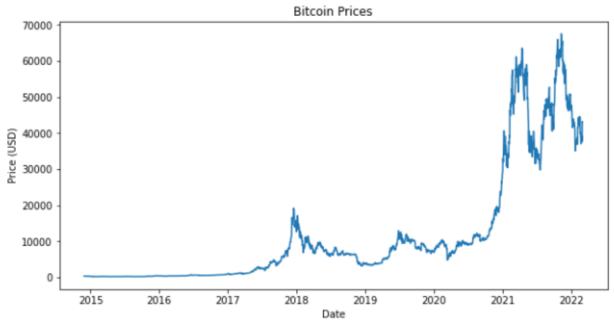
TECHNICAL SECTION

Data Collection and Preprocessing :

The historical data for Bitcoin prices from2015 to 2022, was collected from Kaggle, open source. The dataset consists of features like Unix, Date, opening price, closing price, highest price, and lowest price, Volume. For our analysis, we used the closing price as our target variable.

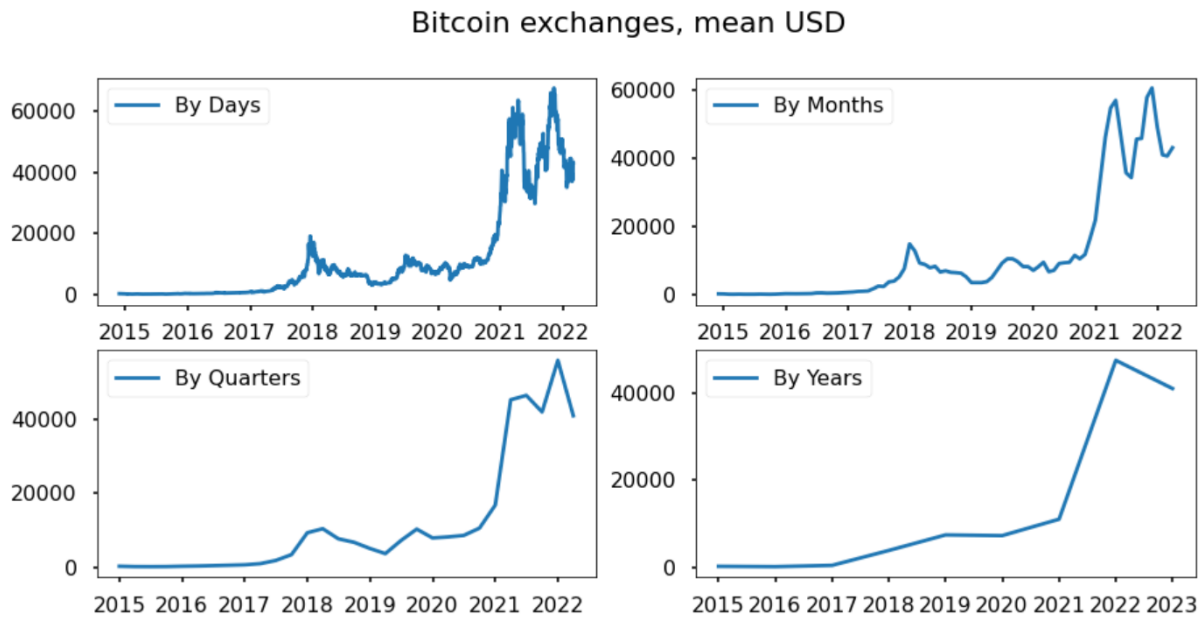
Before training our models, we preprocessed the data by removing any missing values and outliers. Coming to the features, we removed the 'high', 'low', 'Volume' features from the analysis as they are not relevant to gaining insights on Bitcoin prices during the given period.Then we checked for any null values in our dataset using 'isnull()' function.

```
# Visualizing the BTC prices from the dataset
plt.figure(figsize=(10,5))
plt.title('Bitcoin Prices')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.plot(df['close'])
plt.show()
```




```
data.isnull().sum()
```

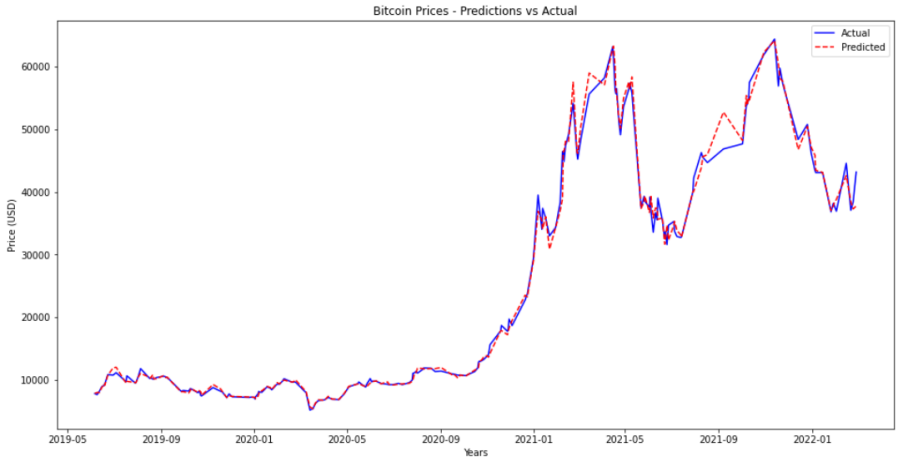
```
unix          0
date          0
symbol        0
open          0
high          0
low           0
close         0
Volume BTC    0
Volume USD    0
dtype: int64
```



Linear Regression :

Linear regression is a simple and commonly used machine learning model for prediction tasks. It is based on the idea of fitting a straight line through a set of data points that can be used to predict future values. In this study, we trained a linear regression model on the Bitcoin price data to predict future prices. The model calculates the relationship between the independent variables, such as the date, and the dependent variable, which is the Bitcoin price. Linear regression models have the advantage of being simple to understand and interpret, but they may not capture more complex relationships between variables. Coming to the implementation, we split the data into training and testing sets, of 80% and 20% respectively. The training set was used to fit the model, and the testing set was used to evaluate the model's performance.

The Linear Regression model gave an RMSE of 1305.35. The R-squared and Adjusted R-squared value was 0.99, indicating a good fit of the model on the data.



```
# Define input and output variables
X = df.drop(['close'], axis=1)
# X = df['unix']
y = df['close']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
# Create a linear regression model
lr_model = LinearRegression()

# Train the model using the training set
lr_model.fit(X_train, y_train)
```

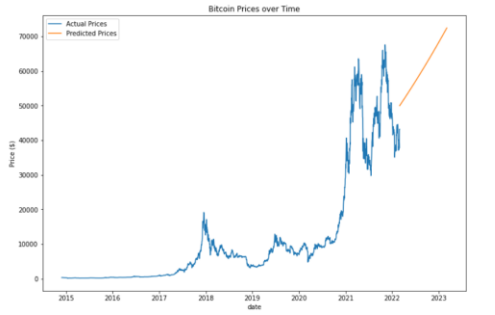
```
LinearRegression()
```

Polynomial Regression :

Polynomial regression is an extension of linear regression that allows for modeling non-linear relationships between variables. In this study, we trained a polynomial regression model with a degree of 3, which means the model includes a quadratic term. The polynomial regression model fits a curve through the data points instead of a straight line, which may better capture more complex relationships between variables. This model has the advantage of being able to capture non-linear trends, but it may also be more complex and prone to overfitting. Coming to the implementation, we split the data into training and testing sets, with 80:20 split. The training set was used to fit the model, and the testing set was used to evaluate the model's performance.

The Polynomial Regression model was used to predict the future prices of bitcoins for future dates ,the model gave an RMSE of 7357.25. The R-squared and Adjusted R-squared value was 0.78, indicating a better fit of the model for predicting the future prices.

```
# Plot the predicted prices
plt.figure(figsize=(12,8))
plt.plot(data['date'], data['close'], label='Actual Prices')
plt.plot(future_dates, future_prices, label='Predicted Prices')
plt.xlabel('date')
plt.ylabel('Price ($)')
plt.title('Bitcoin Prices over Time')
plt.legend()
plt.show()
```




```
# Split the data into training and test sets
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2, random_state=42)
```

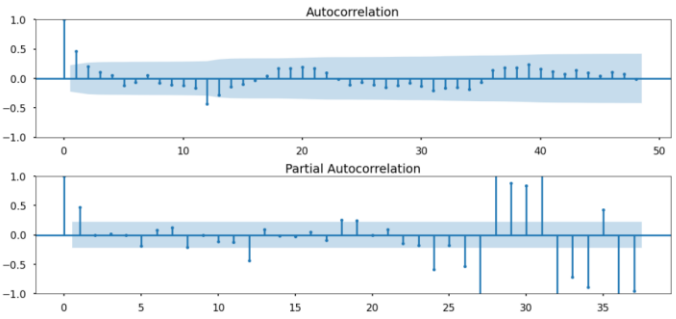
```
# Fit a polynomial regression model with degree 3
poly = PolynomialFeatures(degree=3)
X_train_poly = poly.fit_transform(X1_train)
X_test_poly = poly.transform(X1_test)
lr_model.fit(X_train_poly, y1_train)
```

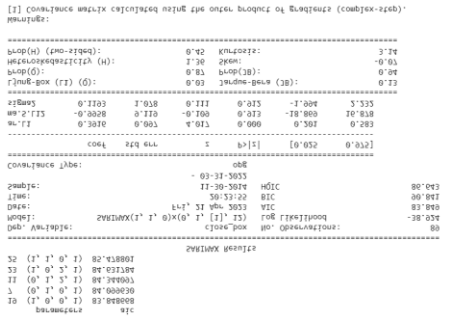
```
LinearRegression()
```

SARIMAX :

SARIMAX is a statistical model that is commonly used for time series analysis and forecasting. In this study, we trained a SARIMAX model on the Bitcoin price data to predict future prices. The model takes into

account the historical values of the time series, as well as external factors such as seasonality and trend. SARIMAX models have the advantage of being able to capture complex time series patterns, but they may also be more difficult to interpret and require more extensive parameter tuning. This was the last model we trained was the Seasonal Autoregressive Integrated Moving Average with Exogenous Regressors (SARIMAX) model. Coming to the implementation, we first used seasonal_decompose() function to elaborate a time series into its constituent components: trend, seasonal, and residual. We then ran a Dickey-Fuller test which is a statistical test used to determine whether a time series is stationary or non-stationary. In this case, the p-value of the Dickey-Fuller test is 0.995798. This suggests that there is very strong evidence that the time series is non-stationary, since the p-value is very close to 1. Then we get, the ACF plot that shows the correlation between a time series and its past values at different lags. Then the PACF plot that shows the correlation between a time series and its lagged values, controlling for the effects of intervening lags. We got an ACF plot shows a significant correlation at lag 1, and no significant correlation at higher lags, an AR(1) model may be appropriate. Then we use, a 'best_model' function is a function that selects the best ARIMA or SARIMA model based on a set of evaluation metrics such as AIC, BIC, or MSE.





EVALUATION SECTION

We used the Root Mean Squared Error (RMSE), R-squared and Adjusted R-squared values to evaluate the performance of the models. The RMSE measures the average squared difference between the predicted and actual values. The R-squared value measures the proportion of variance in the target variable.

For the SARIMAX model, we used the statsmodels library in Python to fit and evaluate the model. We used the Akaike Information Criterion (AIC) to select the best parameters for the SARIMAX model.

The results of the evaluation are summarized in the table below:

```python
# Calculating RMSE, R^2 and Adj R^2 of the model
from sklearn.metrics import mean_squared_error, r2_score

rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
n = X.shape[0]
p = X.shape[1]
adj_r_squared = 1 - (1 - r2) * (n - 1) / (n - p - 1)

print("RMSE: {:.2f}".format(rmse))
print("R-squared: {:.2f}".format(r2))
print("Adj R-squared: {:.2f}".format(adj_r_squared))
```

```
RMSE: 1305.35
R-squared: 0.99
Adj R-squared: 0.99
```

```python
# Calculating RMSE, R^2 and Adj R^2 of the model
from sklearn.metrics import mean_squared_error, r2_score

rmse = mean_squared_error(y1_test, y1_pred, squared=False)
r2 = r2_score(y1_test, y1_pred)
n = X.shape[0]
p = X.shape[1]
adj_r_squared = 1 - (1 - r2) * (n - 1) / (n - p - 1)

print("RMSE: {:.2f}".format(rmse))
print("R-squared: {:.2f}".format(r2))
print("Adj R-squared: {:.2f}".format(adj_r_squared))
```

```
RMSE: 7357.25
R-squared: 0.78
Adj R-squared: 0.78
```

```python
# Evaluate the model
from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt

# Predict test data using SARIMAX model
test_data['Predictions'] = result.predict(start=test_data.index[0], end=test_data.index[-1])

# Calculate RMSE
rmse = sqrt(mean_squared_error(test_data['close'], test_data['Predictions']))
print('RMSE:', rmse)

# Calculate R-squared
r2 = r2_score(test_data['close'], test_data['Predictions'])
print('R-squared:', r2)

# Calculate Adjusted R-squared
n = len(test_data)
p = 1  # only 1 feature used (Close)
adj_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
print('Adjusted R-squared:', adj_r2)
```

```
RMSE: 1813.3217892527884
R-squared: 0.9853004510008021
Adjusted R-squared: 0.9852726635735825
```

CONCLUSION

In conclusion, the Linear Regression model performs the best with 0.99 R^2 and Adjusted R^2 values with RMSE value of 1305.35. Meanwhile SARIMA performs the best for future price predictions with 0.98 R^2 and Adjusted R^2 values.

MEMBER CONTRIBUTION

Shashank Gampa worked on the time series predictions like Linear Regression, Polynomial Regression and SARIMAX models.

Sai Srinath Putta has helped the team to find the dataset and worked on the data preprocessing and data exploration part and helped the team in making the ppt.

Vitan Chopra helped in processing and summarized the results observed from various approaches followed and implemented.

Yannick Mbia helped in Data Preprocessing and Component Analysis.

SOURCE CODE

https://drive.google.com/drive/folders/1xYsJ5kicydAiBdov_F_XZtWPzc3sEt3q?usp=share_link