Project 1 - LSR Model of BTC-USD Data

Austin Phillips
ECE 4850
UVU Electrical Engineering
Orem, Utah
austin.phillips@uvu.edu

Angel Rodriguez
ECE 4850
UVU Electrical Engineering
Orem, Utah
10801309@uvu.edu

***Abstract*—This lab focuses on simple statistical analysis of closing price of Bitcoin between 1/1/2017 and 1/7/2023. A least-squares regression model of the data is explored and used to predict a future date.**

***Index Terms*—LSR, BTC, Statistics, Modeling, Prediction**

I. Statistical Modeling

Along with finding the mean (\$17615.19) and the variance (\$^{2}279202848.84) of the closing price of Bitcoin (BTC) between 1/1/2017 and 1/7/2023, we also plotted a histogram of the data along with the normalized histogram, as seen in Figure 1 and Figure 2 respectively. The histogram would suggest that, while it is possible for a day to end with a high value closing cost, because the histogram is skewed heavily to the right it is much more likely to experience a lower closing cost. The normalized histogram shows the same thing, but because the volume of the histogram totals to one, it is a rough generalization of the probability density function (PDF) of the data. Finally, the mean is plotted with the population in Figure 3, further demonstrating the skew towards the higher closing values.
It is worth noting that where summations are shown, it would be assumed for loops are used for calculation in Python, but a quicker means is used to perform the same calculations. An example can be seen between the formula for variance shown in Equation 5 and its improved form in Equation 6. Python's package numpy is ideal for matrix operations since hardware is used to execute operations in parallel, whereas, for loops would execute the same operations in series taking longer.

II. Least Squares Regression Model

Our LSR model is a fifth-order polynomial defined by Equation 1 and can be seen in Figure 4. The LSR model shows predictions for various values of x and is derived from Equation 2.

$$\mathcal{LSR}(x) = \alpha x^5 + \beta x^4 + \gamma x^3 + \delta x^2 + \epsilon x + \zeta \tag{1}$$

$$\hat{y} = \mathbf{X}\vec{B} \tag{2}$$

$$\vec{B} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \vec{y} \tag{3}$$

$$\mathbf{X} = \begin{bmatrix} x^{(1)5} & x^{(1)4} & x^{(1)3} & x^{(1)2} & x^{(1)1} & 1 \\ x^{(2)5} & x^{(2)4} & x^{(2)3} & x^{(2)2} & x^{(2)1} & 1 \\ x^{(3)5} & x^{(3)4} & x^{(3)3} & x^{(3)2} & x^{(3)1} & 1 \\ ... & ... & ... & ... & ... & ... \\ x^{(n)5} & x^{(n)4} & x^{(n)3} & x^{(n)2} & x^{(n)1} & 1 \end{bmatrix} \tag{4}$$

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n} \tag{5}$$

$$\sigma^2 = \frac{(\vec{y} - \mu)^\top \, (\vec{y} - \mu)}{n} \tag{6}$$

Where:
$\alpha = -7.065\,134\,68 \times 10^{-12}$,
$\beta = -2.707\,023\,76 \times 10^{-8}$,
$\gamma = 1.838\,857\,64 \times 10^{-4}$,
$\delta = -2.470\,223\,16 \times 10^{-1}$,
$\epsilon = 1.095\,792\,34 \times 10^{2}$,
$\zeta = -6.402\,158\,01 \times 10^{3}$

This produces a model that fairly accurately follows the given model data, even matching the mean of the original data. However, its biggest weakness is the fact that values outside of the original data's domain trend negative. This can be clearly seen when we attempt to predict the value on February 15th, 2023: the model predicts a closing value of -\$12,599.43. Since stocks and cryptocurrency cannot hold a negative value, this model severely fails to predict values outside of the original domain.

III. Repeating Experiment in Matlab

To test the accuracy of our model, the experiment is repeated in Matlab to see how the results differ. Differing results can show that outcome was calculated incorrectly on either. To start, the calculated average is 1.761518705954322e+04 and the calculated variance is 2.792028488419386e+08 which matches the earlier results obtained within Python. A histogram from Matlab is shown in Figure 5, the data compared with the average from Matlab is shown in Figure 6, and a 5-th degree polynomial prediction is shown in Figure 7, all of which match the earlier results of Python. In Matlab, the experiment will proceed with the prediction model being a

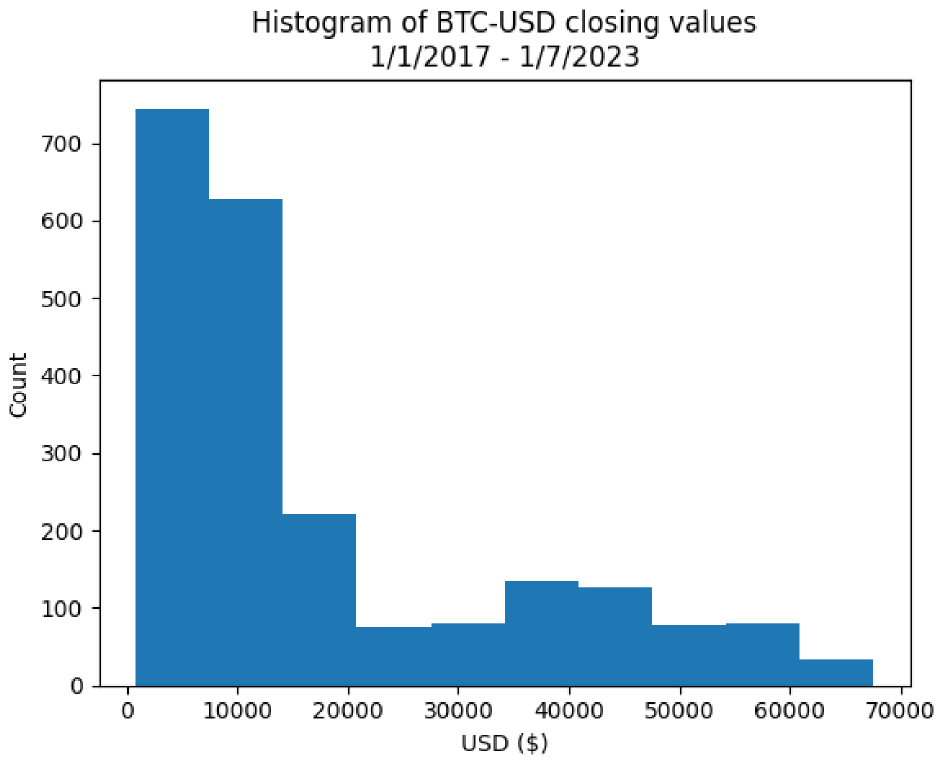

Fig. 1. Histogram of population

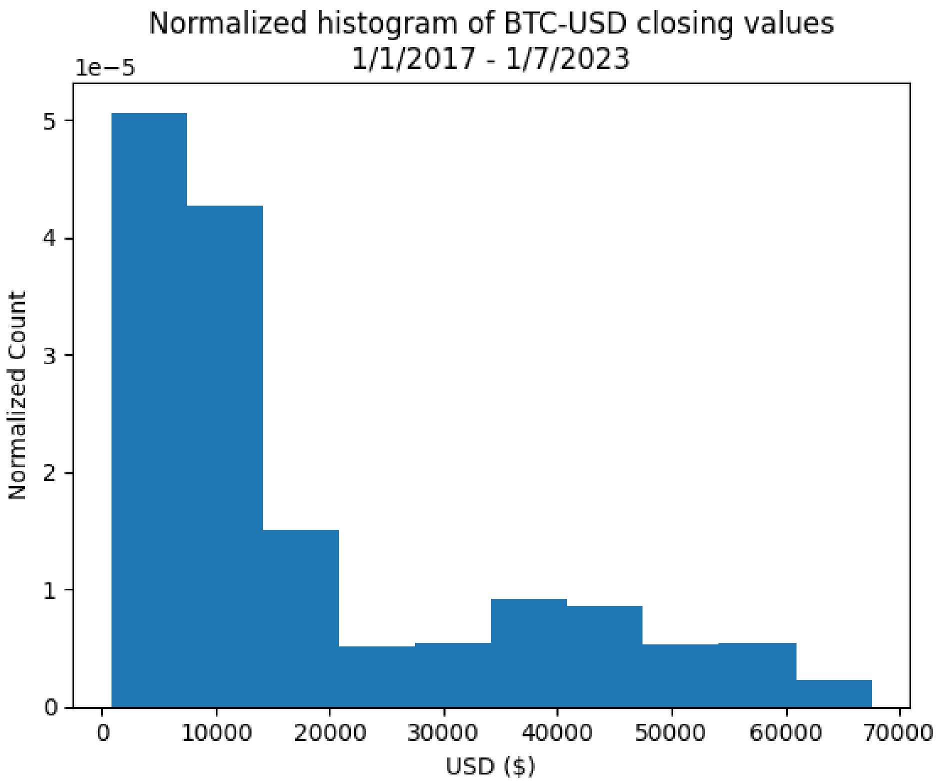

Fig. 2. Normalized histogram of population

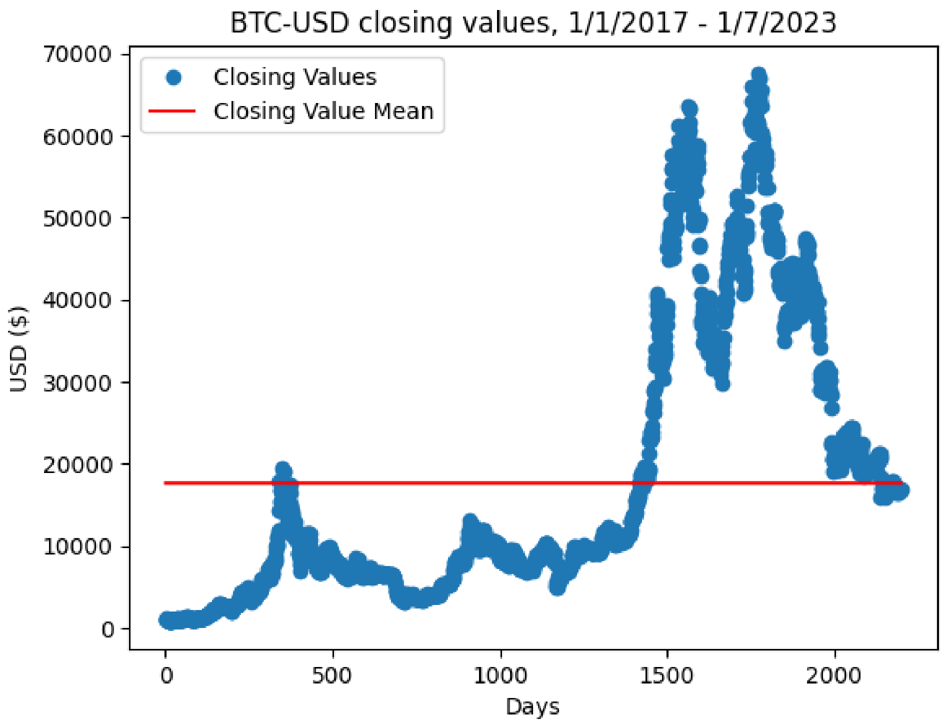

Fig. 3. Plot of population with the mean

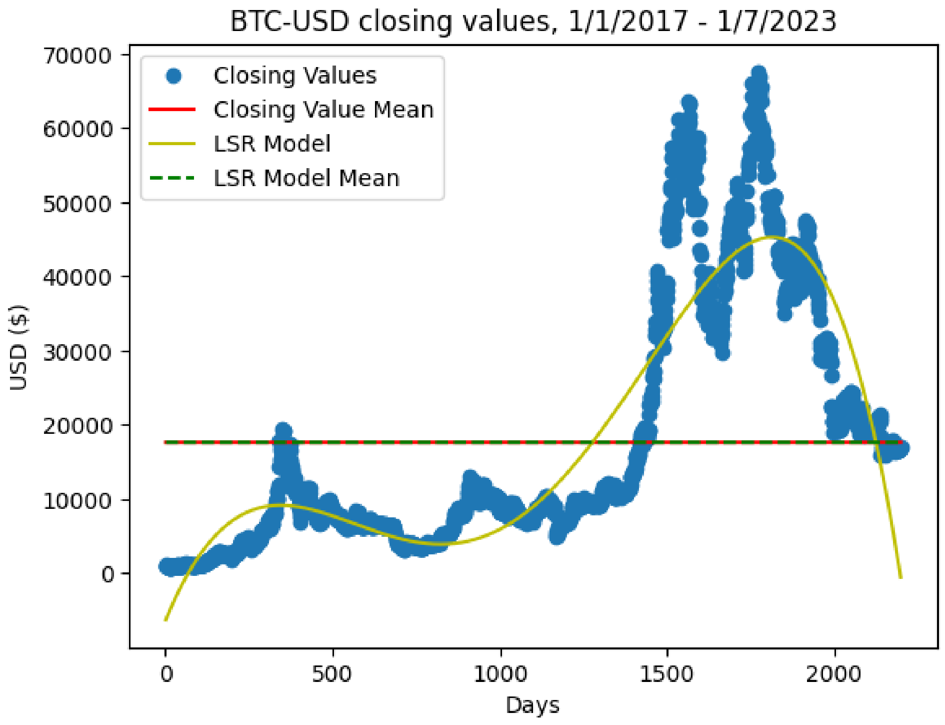

Fig. 4. Plot of population with the mean and our LSR model

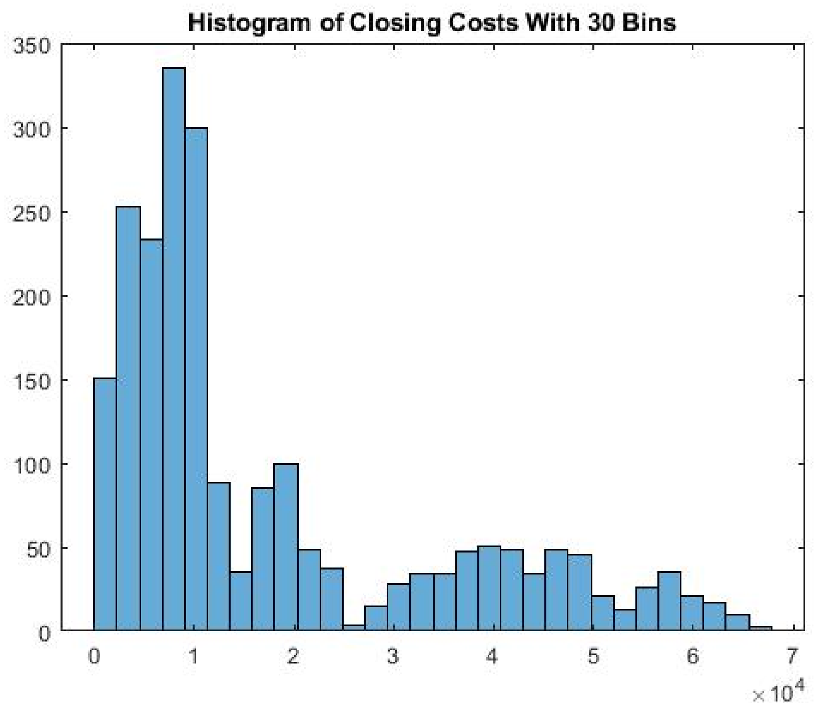

Fig. 5. Matlab Histogram of population

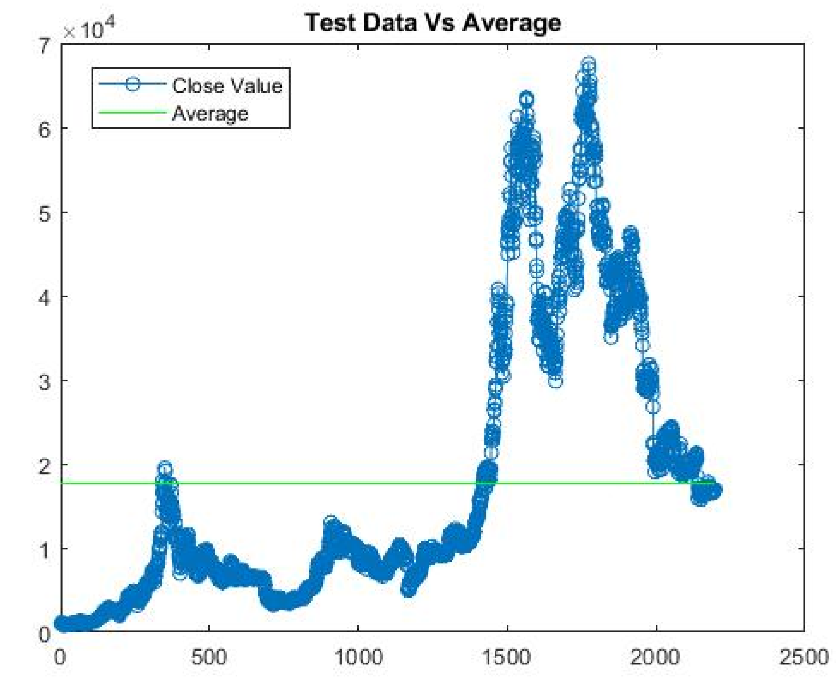

Fig. 6. Matlab Plot of population with the mean

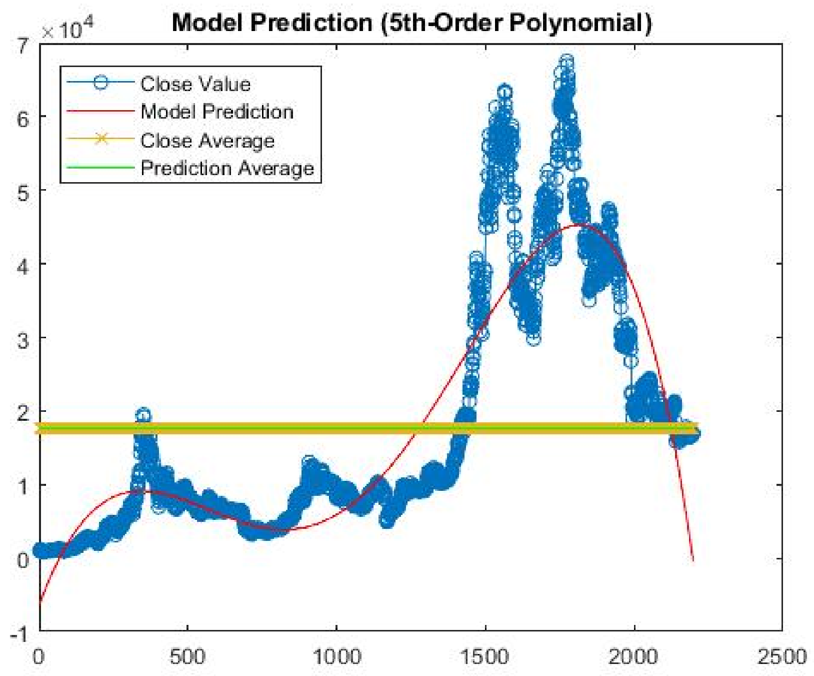

Fig. 7. Matlab Plot of population with the mean and our LSR model - 5th degree polynomial

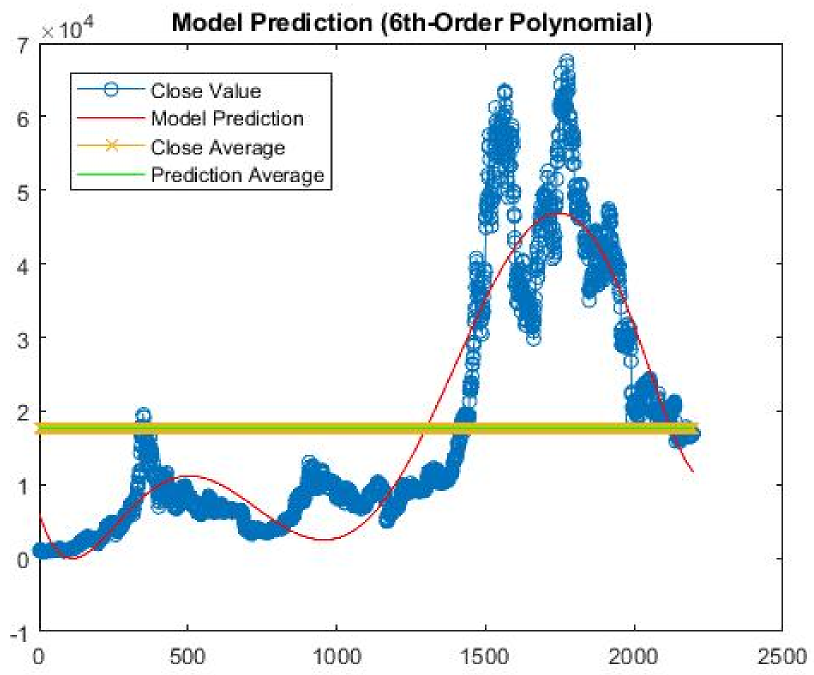

Fig. 8. Matlab Plot of population with the mean and our LSR model - 6th degree polynomial

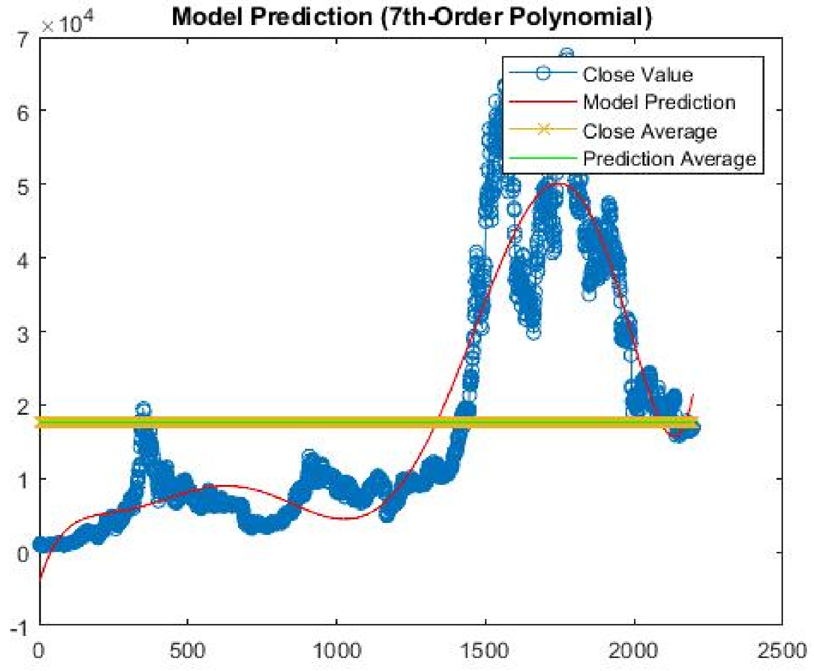

Fig. 9. Matlab Plot of population with the mean and our LSR model - 7th degree polynomial

6th-degree polynomial and a 7th-degree polynomial which the goal of not having a negative value for the target prediction since crypto currency prices cannot be negative. The results are shown in Figure 8 and Figure 9. The predicted price of 02/15/2023 using the 6-th degree polynomial is 10594.71 USD and 32718.60 USD for the 7th-degree polynomial model.

IV. Conclusion

Our model and findings in this project succeed in many ways but fail in others. It's possible to produce a better model, but we find the one produced to work well enough for project completion. The LSR model is a relatively simple introduction to the task of machine learning. With the experimentation in this project, we are prepared to dive deeper in the subject. Our coding was performed in a Jupyter Notebook and can be seen in Section V. Previously, it was hypothesised the higher-degree the polynomial, the more accurate, but these figures show differently. Figure 9 shows the model dipping more into the negative values than Figure 8, so it is concluded that the 6th-degree polynomial is the better fit than the other two models. In addition, the 5th-degree polynomial model appeared inadequate since where you can see the rate of change in the closing value stabilize, the model doubles down and continues dipping fast when it is clear the price will stagnate. Also, it is worth noting the 7th-degree polynomial shows the closing value jump at the end where the data shows it to stagnate. It is concluded that the 6-th degree polynomial model is the best fit of these. The predicted close price of Bitcoin on 02/15/2023 is 10594.71 USD.

V. Code

A. Part 1

```
import numpy as np
from openpyxl import load_workbook
import matplotlib.pyplot as plt

btc_workbook = load_workbook('BTC-USD.xlsx')
```

B. Part 2

```
btc_sheet = btc_workbook.active
closing_values = np.array([])
for value in btc_sheet.iter_rows(min_row=2,
                                min_col=6,
                                max_col=6,
                                values_only=True):
    closing_values = np.append(closing_values, value)
```

C. Part 3

```
%% Part 3-ish

x_hat = filter(a, b, x_echo_h);

figure;
subplot(3, 1, 1);
plot(0:1/Fs:1/Fs*(length(x)-1),x);
xlabel('t(s)');
ylabel('sound level');
title('x');
subplot(3, 1, 2);
plot(0:1/Fs:1/Fs*(length(x_echo_h)-1),x_echo_h);
xlabel('t(s)');
ylabel('sound level');
title('x echo h');
subplot(3, 1, 3);
plot(0:1/Fs:1/Fs*(length(x_hat)-1),x_hat);
xlabel('t(s)');
ylabel('sound level');
title('x hat');

soundsc(x_echo_h, Fs); pause(10);
soundsc(x_hat, Fs); pause(10);
```

D. Part 4

```
mean_cv = np.sum(closing_values) / closing_values.size
print(f"Mean of the closing values: {mean_cv}")
alpha = closing_values - mean_cv
variance = np.dot(alpha.T, alpha) / closing_values.size
print(f"Variance of the closing values: {variance}")
```

E. Part 5

```
num_bins = 10

counts_cv = np.zeros(num_bins)
bins_cv = np.linspace(np.amin(closing_values), np.amax(closing_values), num_bins+1)
for value in closing_values:
    for i in range(num_bins-1, -1, -1):
        if value >= bins_cv[i]:
            counts_cv[i] += 1
            break

plt.stairs(counts_cv, bins_cv, fill=True)
plt.title("Histogram of BTC-USD closing values\n1/1/2017 - 1/7/2023")
plt.xlabel('USD ($)')
plt.ylabel('Count')
plt.show()

plt.stairs((counts_cv / (np.sum(counts_cv) * np.diff(bins_cv))), bins_cv, fill=True)
plt.title("Normalized histogram of BTC-USD closing values\n1/1/2017 - 1/7/2023")
plt.xlabel('USD ($)')
plt.ylabel('Normalized Count')
plt.show()

# using the built in values
plt.hist(closing_values)
plt.title("Histogram of BTC-USD closing values\nusing built in function, 1/1/2017 - 1/7/2023")
plt.xlabel('USD ($)')
plt.ylabel('Count')
plt.show()

plt.hist(closing_values, density=True)
plt.title("Normalized histogram of BTC-USD closing values\nusing built in function, 1/1/2017 -
plt.xlabel('USD ($)')
plt.ylabel('Normalized Count')
plt.show()
```

F. Part 6

```
num_days = np.arange(1, closing_values.size + 1)
plt.plot(num_days, closing_values, 'o', label='Closing Values')
plt.xlabel('Days')
plt.ylabel('USD ($)')
plt.title('BTC-USD closing values, 1/1/2017 - 1/7/2023')
plt.plot(num_days, mean_cv*np.ones((closing_values.size, 1)), 'r', label='Closing Value Mean')
#plt.plot(num_days, np.median(closing_values)*np.ones((closing_values.size, 1)), 'g', label='C
plt.legend()
plt.show()
```

G. Part 7

```
x = num_days
X_aug = np.stack((x**5, x**4, x**3, x**2, x, np.ones((x.size))), axis=1)
Beta = np.dot(np.dot(np.linalg.inv(np.dot(X_aug.T, X_aug)), X_aug.T), closing_values)
```

H. Part 8

```
Y_hat = np.dot(X_aug, Beta)
mean_Y_hat = np.sum(Y_hat) / Y_hat.size
plt.plot(num_days, closing_values, 'o', label='Closing Values')
plt.xlabel('Days')
plt.ylabel('USD ($)')
plt.title('BTC-USD closing values, 1/1/2017 - 1/7/2023')
plt.plot(num_days, mean_cv*np.ones((closing_values.size, 1)), 'r', label='Closing Value Mean')
plt.plot(num_days, Y_hat, 'y', label='LSR Model')
plt.plot(num_days, mean_Y_hat*np.ones((Y_hat.size, 1)), 'g', linestyle='dashed', label='LSR Mo
plt.legend()
plt.show()
```

I. Part 9

```
# Feb 15th, 2023 corresponds to day 2237
x_day = 2237
equation_vector = np.array([x_day**5, x_day**4, x_day**3, x_day**2, x_day, 1])
y_hat_cv = np.dot(equation_vector, Beta)
print(f"The predicted closing value of BTC on Feb. 15th, 2023 is: ${np.around(y_hat_cv, decima
print(f"This seems fairly accurate, lol.")
```

J. Part 9 - Matlab code

```
clc, close all; clear

filename = 'BTC-USD.xlsx';
% closes = readtable(filename)
y = xlsread(filename, 'F:F');
x = [1:length(y)]';
average = sum(y) / length(y);
variance = ((y-average)'*(y-average))/length(y);
fprintf('The mean of the closing values is: %d', average)
fprintf('The variance of the closing values is: %d', variance)
X_aug = [x.^5,x.^4,x.^3,x.^2,x,ones(length(x),1)] ;
X_trans = X_aug';
B = ((X_trans*X_aug)^(-1))*X_trans*y;
y_hat = X_aug*B;
x_day = 2236
equation_vector = [x_day^5, x_day^4, x_day^3, x_day^2, x_day, 1];
y_hat_prediction = equation_vector*B;
nbins = 30;
histogram(y,nbins); title("Histogram of Closing Costs With 30 Bins")

plot(x,y,'-o'); title("Test Data Vs Average");
hold on;
plot(x,average*ones(length(x),1),'g');
legend('Close Value','Average')
hold off;

fprintf('Prediction for 1/6/2023 using 5th-degree polynomial is: %d', y_hat_prediction)
average_prediction = sum(y_hat) / length(y_hat);
plot(x,y,'-o'); title("Model Prediction (5th-Order Polynomial)");
hold on;
plot(x,y_hat,'r');
plot(x,average_prediction*ones(length(x),1),'-x');
plot(x,average*ones(length(x),1),'g');
legend('Close Value','Model Prediction', 'Close Average', 'Prediction Average')
```

```
hold off;


% 6th degree for fun

X_aug = [x.^6,x.^5,x.^4,x.^3,x.^2,x,ones(length(x),1)] ;
X_trans = X_aug';
B = ((X_trans*X_aug)^(-1))*X_trans*y;
y_hat = X_aug*B;
equation_vector = [x_day^6,x_day^5, x_day^4, x_day^3, x_day^2, x_day, 1];
y_hat_prediction = equation_vector*B;
fprintf('Prediction for 1/6/2023 using 6th-degree polynomial is: %d', y_hat_prediction)

average_prediction = sum(y_hat) / length(y_hat);
plot(x,y,'-o'); title("Model Prediction (6th-Order Polynomial)");
hold on;
plot(x,y_hat,'r');
plot(x,average_prediction*ones(length(x),1),'-x');
plot(x,average*ones(length(x),1),'g');
legend('Close Value','Model Prediction', 'Close Average', 'Prediction Average')
hold off;


% 7th degree for fun
X_aug = [x.^7,x.^6,x.^5,x.^4,x.^3,x.^2,x,ones(length(x),1)] ;
X_trans = X_aug';
B = ((X_trans*X_aug)^(-1))*X_trans*y;
y_hat = X_aug*B;
equation_vector = [x_day^6,x_day^5, x_day^4, x_day^3, x_day^2, x_day, 1];
y_hat_prediction = equation_vector*B;
fprintf('Prediction for 1/6/2023 using 7th-degree polynomial is: %d', y_hat_prediction)

average_prediction = sum(y_hat) / length(y_hat);
plot(x,y,'-o'); title("Model Prediction (7th-Order Polynomial)");
hold on;
plot(x,y_hat,'r');
plot(x,average_prediction*ones(length(x),1),'-x');
plot(x,average*ones(length(x),1),'g');
legend('Close Value','Model Prediction', 'Close Average', 'Prediction Average')
hold off;
```