Least Squares

Reid Vogel

November 2025

1 Abstract

Being able to read and analyze data is one of the most important skills in today's world. Almost everything has some data information associated that can be parsed, analyzed, and applied to solving some problem. Without utilizing data, society would not be able to make educated decisions, drive innovation, and improve efficiency on goods already created.

In this scenario, we will be looking at a data set from Kaggle titled 'Analyzing Student Academic Trends'. Inside of this dataset the two columns, hours studied and exam score, are used to solve the question 'Is there a linear or non-linear relationship between a student time spent studying and their exam score'. By running this data set through some python code, we used 4 different line fitting methods to best analyze the correlation between time studied and exam scores. Each of these 4 different methods resulted in lines of best fit to represent the data. In general, it was found that the more time spent studying, the better the exam scores were across the board.

2 Introduction

To answer our hypothesis, there is a linear relationship between a student time spent studying and their exam score, we used different ways to solve least square problems to create the line of best fit. These methods include Cholesky Decomposition, QR decomposition, LSQR Check, and a non-linear analysis using a quadratic poly 2 fit. By using these four graphs, we are able to visualize the relationship between the data and possibly predict possible outcomes with unknown future data.

Ever since the start of school, students will always make excuses to not study for exams claiming they already know everything or that it will not have a positive impact on their final exam result. This experimental question 'Is there a linear or non-linear relationship between a student time spent studying and their exam score' is aimed to see if students who study are able to produce better exam scores. To ensure a linear relationship, for every hour of studying, there needs to be some sort of increase in the exam score. In contrast, a non-linear relationship would show after a set number of hours students are unable to increase their scores and the data points would plateau out around a common grade.

3 Methods and Computations

3.1 Dataset

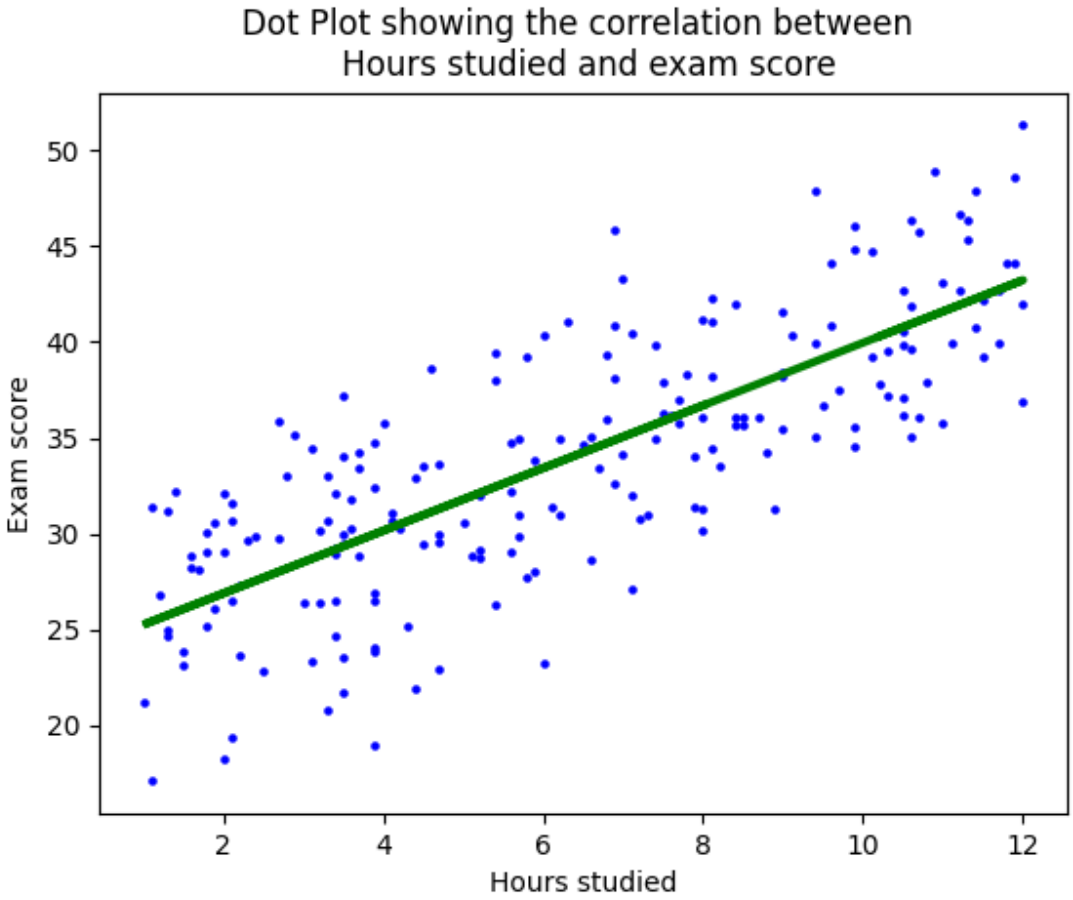
The data set 'Analyzing Student Academic Trends' was downloaded from Kaggle with this link

https://www.kaggle.com/datasets/saadaliyaseen/analyzing-student-academic-trends.
Around two months ago, Saad Ali Yaseen published this data for public use with the goal of understanding learning achievements. The students data set collected information on 200 students, covering study hours, sleep duration, attendance, past scores, and final exam results. The average study time was 6.3 hours, slept 6.6 hours, and attended class 74.8 percent. The exam scores ranged from 17.1 to 51.3 with a mean of 34. Looking at the provided bar graph on Kaggle, the student exam scores form a fairly standard bell curve. There is no information regarding the age/grade these students are attending during this study. There is also no information on how they selected these students, random sampling or volunteers, for the data set so these results can't really be generalized to all students. By only looking at this data set, it's hard to determine direct correlation between our two variables, hours studied and exam scores, as there my be confounding variables such as previous classes taken, attendance rate, or even time spent not paying attention in class. After plotting all of this data, there is a clear visualization that there are no outliers in the data that will affect the line of best fit. By having extreme outliers, you are either going to get a line of best fit that either under or over represents that data trying to compensate just a few data points. Sometimes, it would almost be better to exclude the extreme outliers, as it would allow the line of best fit to focus more on the majority of that data.

3.2 Comparison of Linear Algorithms

Below are the resulting graphs and output created by the python file LeastSquares.py. Each subsection will include an explanation of the python code and math. For the comparison, each of these methods produced very similar results for the line of best fit. Cholesky, QR, and LSQR all produced a slope roughly around 1.63410989. However, each result was slightly different after getting to around the 13th decimal place. While mathematically these should produce the same result since they all are fundamentally solving the same least squares problem, in practice there is a slight rounding error since computers aren't able to compute such finite decimals. This is the same problem that we run into when looking at the intercept for each of these methods.

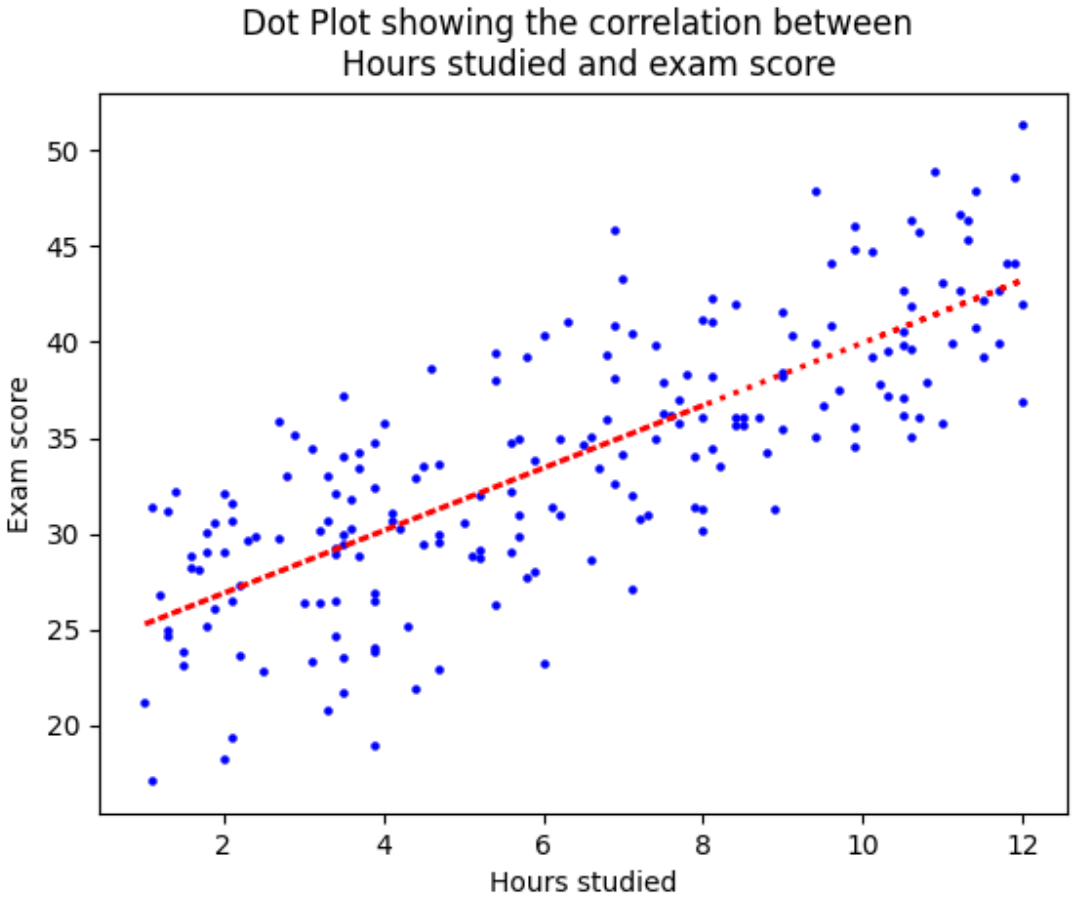
3.2.1 Cholesky Decomposition




```
----Cholesky Decomposition----
 Intercept Chol: 23.618437852039314 slope: 1.6341098961284777
```

When first starting solving with Cholesky Decomposition we take the x values and create a nx2 (n = length of x) matrix where one column is filled with all ones while the other is the x values. From there, we are able to solve for the normal equations and finally use the cholesky function to find the lower triangular matrix. After finding the lower triangular matrix, we are able to use both backwards and forwards substitution to find the slope and intercept for this equation. Cholesky is super useful since we are able to break down this matrix into two simpler triangular matrices allowing for easier computations. Looking at the graph, this line of best fit supports our hypothesis since we have a positive slope supporting more the student studies the better exam score they are receiving.

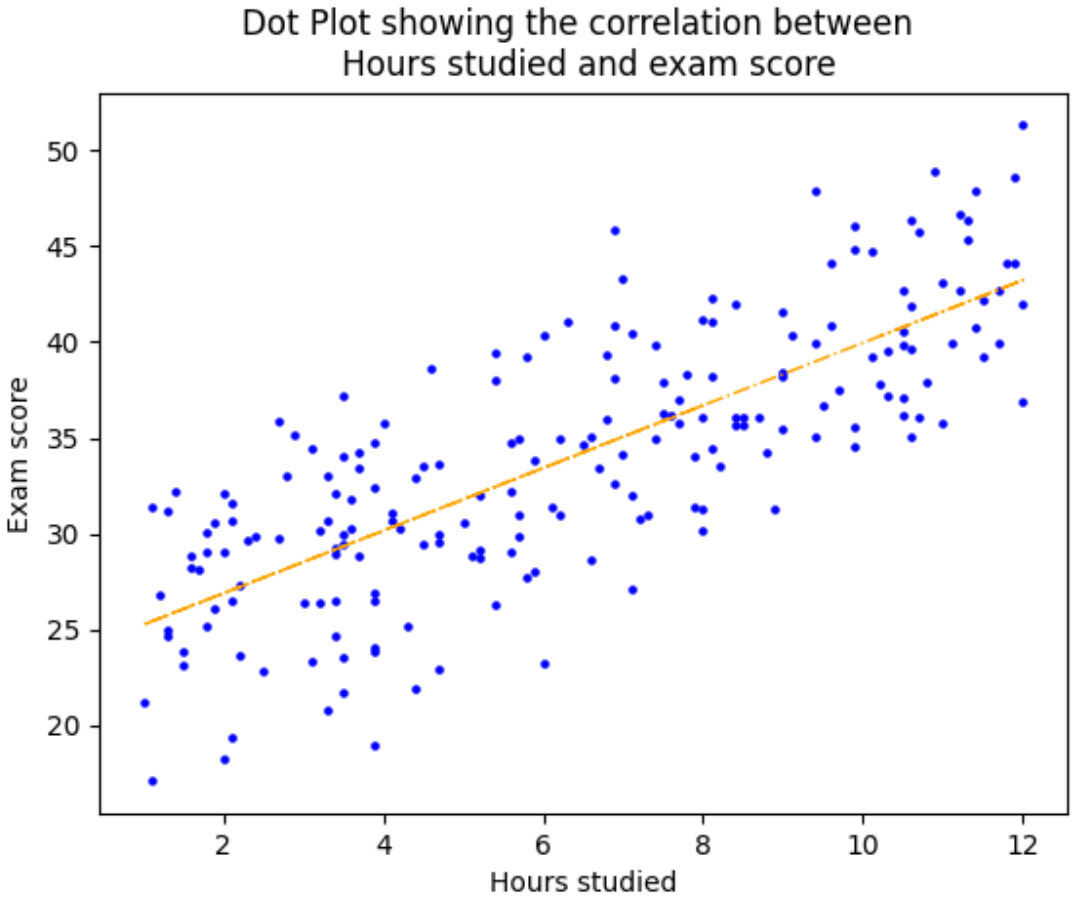
3.2.2 QR Decomposition




```
----QR Decomposition----
 Intercept QR: 23.618437852039314 slope: 1.6341098961284781
```

When solving the line of best fit with QR decomposition we first start by using the method qr in python to find the Q and R matrices. Then we multiply both sides by Q^T to simplify the left side by the identity matrix. From there we are able to back substitute with the upper triangular matrix to solve the β[intercept, slope].

3.2.3 LSQR




```
----LSQR----
 Intercept lsqr: 23.618437852039335 slope: 1.6341098961284772
```

This is equivalent to the built in solver in Matlab. Inside of the Numpy library they have the lstsq method that will solve for everything needed to find the line of best fit. Numpy documentation goes in depth about how each of these are results are obtained. These results found in the built in least square method are almost exactly the same for what we found when using the normal equations and QR decomposition to find the line of best fit.

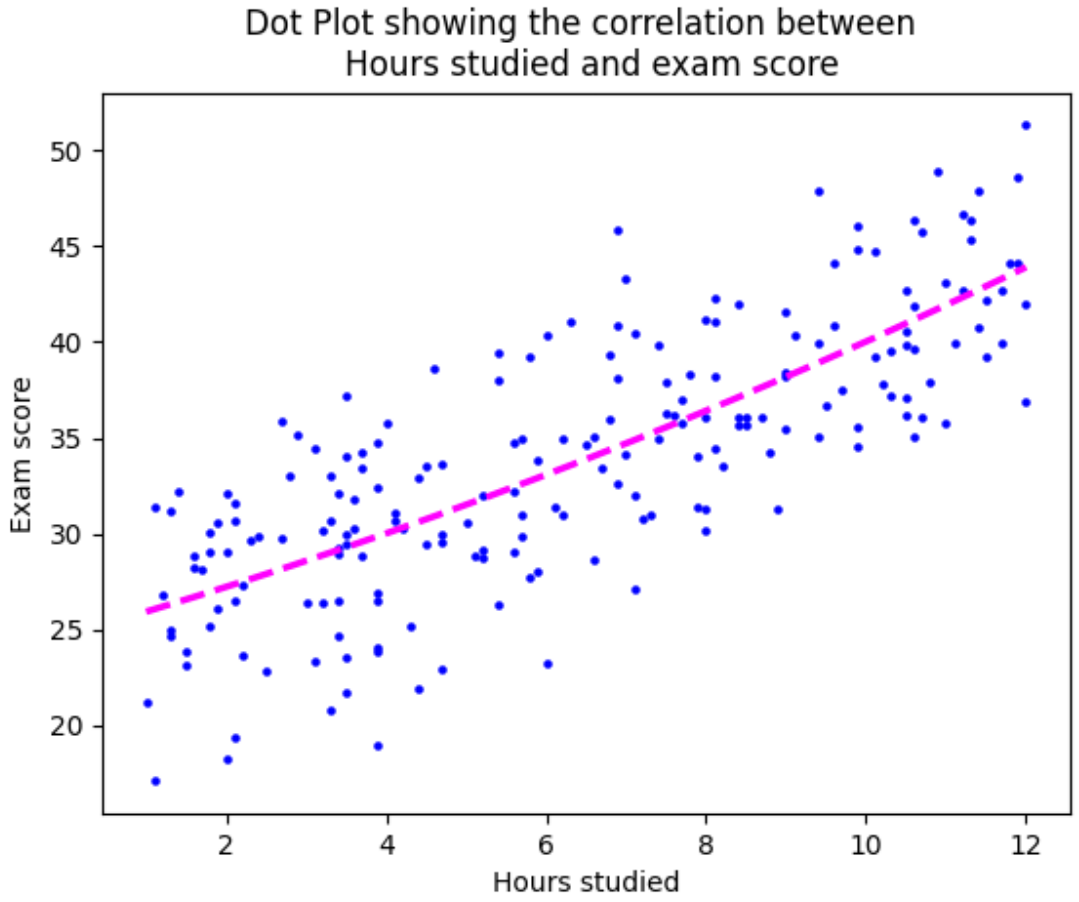
3.3 Condition Number

```
Condition Number: 15.895971314278766
```

The condition number is shown in the image above by using the Numpy python method cond(). The condition number shows how the output value is going to effected depending on how large the input value changes. With our given condition number we can say that our matrix is well conditioned since our value is very close to 1. With a smaller condition number this proves that our matrix is closer to being non-singular compared to singular. This shows that there is a high correlation between the columns and rows of the matrix.

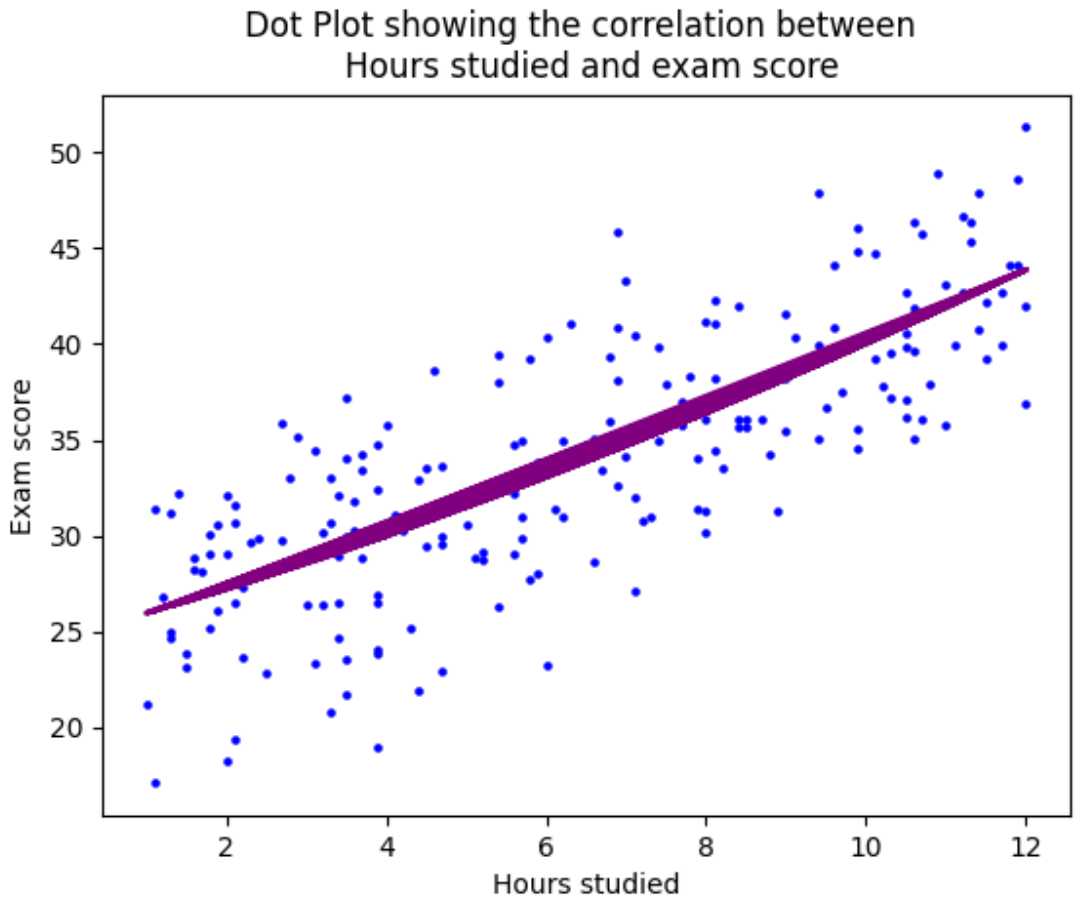
3.4 Other Curve Fits

3.4.1 Poly2 fit



For the other line of best fit used was a quadratic equation. The equation $Y = p1 * x^2 + p2 * x + p3$ was used to create this line of best fit. In the python code, the Numpy library has a polyfit function that will solve the line of best fit. After solving for p1,p2, and p3 there is some code that will smooth out the line for plotting on the graph. Looking at the graph, the line of best quadratic fit is almost linear. This proves that our hypothesis is correct because if the students exams were to level out after a certain amount of time studied would have a negative slope.

3.4.2 Third Polynomial



In this line of best fit we are doing something very similar as above but instead taking the third polynomial. Here we are creating a non-linear graph to try and represent the data better than the previous linear slopes. However, with such linear data the curve is minimal. In the code we use the Numpy polyfit function and then the polyval function to evaluate the polynomial at specific x values. The result is the line of best fit created above. This line of best fit makes sense for our data as it covers the majority of the data while including the slight outliers above and below.

Overall the two polynomial methods above are super useful because it is able to cover some of the data that is missed using a linear best line of fit. By adding a slight curve to our graph we are able to represent the outliers while still getting across that there is a positive corelation between the students time spent studying and their exam scores.

4 Summary of Results

After creating and analyzing the different graphs the hypothesis, there is a linear correlation between student hours studied and exam scores, can be confirmed. For each of the lines, they all created almost exactly the same line of best fit to show the positive linear correlation. Another thing to back up the evidence was how the poly 2 graph almost created a positive linear slope.

There isn't necessarily one correct way to find the line of best fit for this data set since it is so obvious that there is a positive corelation between the hours studied and the students exam scores. Since they all solve the least squares problem it would come down to which would solve for the line of best fit the quickest and computationally the cheapest. As mentioned in the data set section, it isn't necessarily possible to fully suggest that hours studied is the only reason that students got a better exam score. There could be other confounding factors that influenced the students scores. Putting in some personal thought about the data set this makes perfect sense. The more time that a student studies for an exam should definitely increase their exam scores.

5 Conclusion

Curve fitting is the process of finding the line of best fit between a multiple variable data set. By applying methods of linear, non-linear, or quadratic equations we can visually represent a correlation between the two sets of data, and make predictions, model trends, and understand real world problems. While the line of best fit won't always produce a perfect representation of the data it can lower the error between the the graph and real observed data.

When interpreting graphs from the news and websites, it is crucial to understand the basics of curve fitting. Even if you didn't take the data and create the 100 percent accurate line of best fit, being able to look and create a basic line in your head gives another whole interpretation on that data. It's important to think if the line of best fit over or under represents that data due to outliers within the data set. By keeping this in mind, it allows to verify that the line created is truly representing the whole data set and can draw an accurate conclusion.

Overall, understanding curve fitting will help build statistical literacy. By looking beyond the dotted points on the graphs allows users to think about the sample size, outliers in the data, and how other factors are effecting the claim.

6 Appendix - python code

Below is python code that was run to generate all the graphs above. Some lines are wrapped in order to fit onto the overleaf page. Indentation may need to be adjusted if the code is directly copied into an IDE. Currently all lines of best fit are printed to the same graph, but commenting them out individually would allow them to be shown on the data set by themselves.

```
#Question: Is there a linear or non-linear relationship between a student time
    spent studying and their exam score
#Link to dataset:
    http://kaggle.com/datasets/saadaliyaseen/analyzing-student-academic-trends


import csv
import matplotlib.pyplot as plt
import numpy as np


if __name__ == '__main__':

    #create x and y axis arrays
    #Hours studied
```

```
x = []
#Exam Score
y = []

#read in the file
with open('student_exam_scores.csv', 'r', newline='') as csvfile:
    reader = csv.reader(csvfile)
    header = next(reader)
    for row in reader:
        hours_studied = float(row[1])  # hours studied
        score = float(row[5])  # score on exam
        x.append(hours_studied)
        y.append(score)


#convert into numpy array
x = np.array(x)
y = np.array(y)

#------linear fit code using cholesky decomposition--------
#create Matrix A
A = np.column_stack((np.ones_like(x), x))

#form the normal equations
ATA = A.T @ A
ATy = A.T @ y

#cholesky decomposition
L = np.linalg.cholesky(ATA)

#Solve L * y' = b(forward substitution)
y_prime = np.linalg.solve(L, ATy)

#Solve L.T * B = y'(backward substitution)
beta = np.linalg.solve(L.T, y_prime)

#getting the intercept and slope from beta which was solved in the back sub
    above
intercept_chol, slope_chol = beta

#------QR decomposition and solving for x using back substitution---------
#back_substitution function
def back_substitution(R, y):
    n = len(y)
    x = np.zeros_like(y)
    for i in reversed(range(n)):
        x[i] = (y[i] - np.dot(R[i, i + 1:], x[i + 1:])) / R[i, i]
    return x

#qr decomposition
Q, R = np.linalg.qr(A)
y_qr = Q.T @ y

beta_qr = back_substitution(R, y_qr)
intercept_qr, slope_qr = beta_qr

#-----LSQR check--------

#this is the equivalent to matlab lsqr method.
```

```
beta_lsqr, residuals, rank, s = np.linalg.lstsq(A, y, rcond=None)
intercept_lsqr, slope_lsqr = beta_lsqr

#-----Condition number-------
condition_number = np.linalg.cond(A)
print("Condition Number: {}".format(condition_number))
#not a super large condition number but not considered well-conditioned

#other curve fit
# Fit a quadratic curve: Y = p1*x^2 + p2*x + p3
p1, p2, p3 = np.polyfit(x, y, 2)

#smoothig out the line
x_fit = np.linspace(min(x), max(x), 200)
y_fit = p1 * x_fit ** 2 + p2 * x_fit + p3

#solving for the 3rd polynomial equation
coefficients = np.polyfit(x, y, 3)
third_fit = np.polyval(coefficients, x)


# create graph and line of best fit
plt.scatter(x, y, marker='o', color='blue', s=5)
#plotting cholesky
plt.plot(x, intercept_chol + slope_chol * x, color = 'green', linewidth = 3)
#plotting qr in dotted red
plt.plot(x, intercept_qr + slope_qr * x, color='red', linestyle='dotted',
    linewidth = 2)
#plotting the lsqr method
plt.plot(x, intercept_lsqr + slope_lsqr * x, color='orange',
    linestyle='dashdot', linewidth = 1)
#Plotting the quadratic (poly2) fit
plt.plot(x_fit, y_fit, color='magenta', linestyle='--', linewidth=2.5)
#plotting cubic polynomial
plt.plot(x, third_fit, color='purple', linestyle='-', linewidth=2)



#hard to tell, but when both lines are plotted they are overlapping each
    other on the graph.
#Both the chol and qr decompositions are giving similar answers

#uncomment the two lines below to get the exact values above each point
#for i, txt in enumerate(y):
    #plt.text(x[i], y[i], str(txt), ha='center', va='bottom')

# Add labels and title
plt.xlabel("Hours studied")
plt.ylabel("Exam score")
plt.title("Dot Plot showing the correlation between \n Hours studied and exam
    score")

# Display the plot
plt.show()

#print out all the slopes and intercepts of the three methods
print("----Cholesky Decomposition----\n Intercept Chol:
    {}".format(intercept_chol)+" slope: {}".format(slope_chol))
```

```
print("----QR Decomposition----\n Intercept QR: {}".format(intercept_qr) + "
    slope: {}".format(slope_qr))
print("----LSQR----\n Intercept lsqr: {}".format(intercept_lsqr) + " slope:
    {}".format(slope_lsqr))
```