MATH 4334: Mathematical Modeling (HW #2a)

Reece Iriye

September 28, 2023

Exercise 1

The data sets braking-distance-1.xlsx and braking-distance-2.xlsx contain two sets of measurements on the braking distance of a car as a function of speed. The experimental setup consists of a driver on a closed course. At a random, unexpected time, an experimenter simultaneously records the car's position, and causes the sound of a bell to be played inside the car. At that point, the driver must apply the brakes and bring the car to a stop as so on as possible, and the distance traveled from the moment the bell rings is recorded. The experiment is conducted for two sets of conditions.

For each data set,

(a) Find a polynomial model that fits each data set reasonably well

(b) justify the order of, and terms in, the polynomial you chose

(c) record the polynomial coefficients you obtained (for future use)

(d) Use your model to predict the braking distance for each case if the speed were 100 mph

(e) Discuss whether your prediction seems reasonable, and any limitations that should be considered

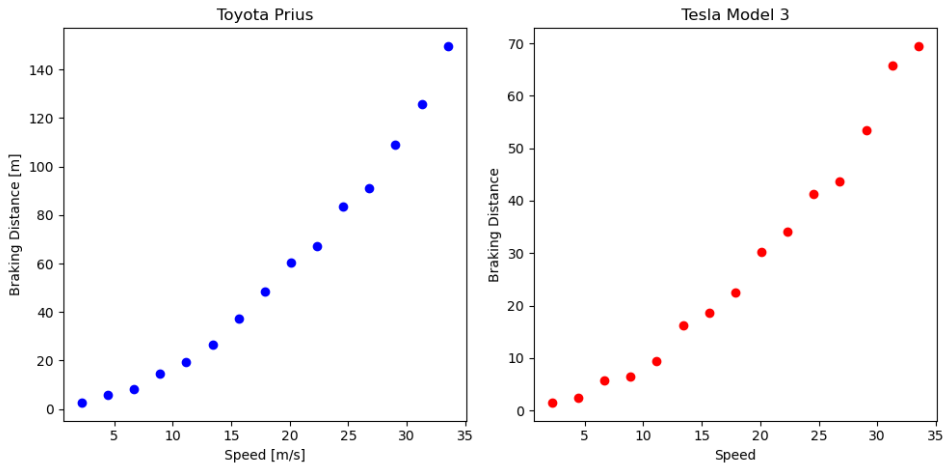

Figure 1: *Raw Data for Braking Distance of Both Cars*

In Figure 1, the plots of raw data recorded depicting braking distance vs. driving speed for

both the Toyota Prius and the Tesla Model 3 can be seen. All of the models below record speed as a measure of m/s instead of mph.

Toyota Prius Analysis:

For the Toyota Prius, the graph absolutely looks like it should depict some sort of polynomial fit, and before any analysis, I hypothesize that a fit of degree 2 will correspond with it well.

For the Toyota Prius, I first test polynomial fits from 1 degree up to 7 degrees. Starting with a degree 1 polynomial, I start trying to create a best fit polynomial below.

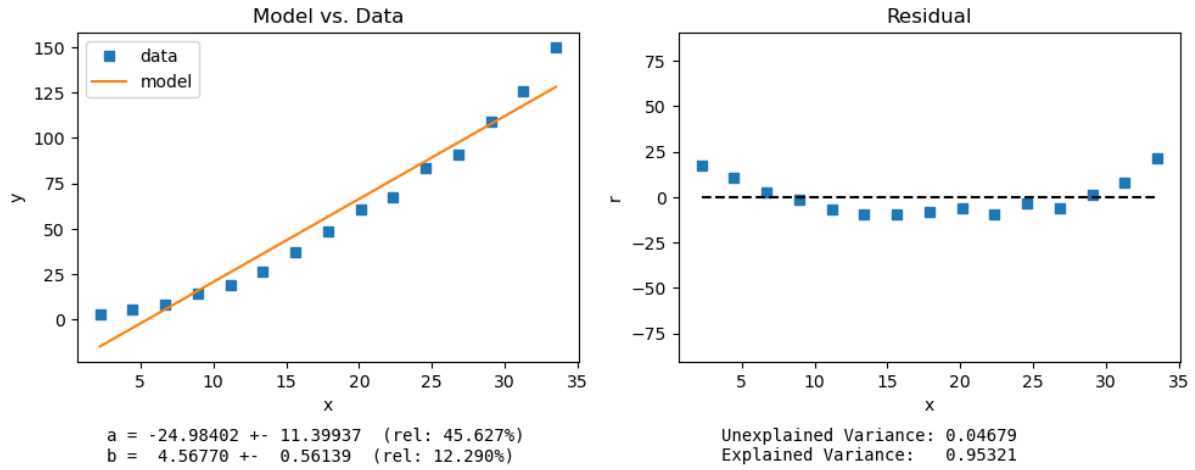

Figure 2: *Degree 1 Model Fit for Toyota Prius*

Figure 2 depicts the fit of a degree 1 polynomial in predicting braking distance based on car speed. I can tell that the linear fit under-fits the data because of the upward facing bowl shape in the residual plot. There is clearly some sort of curvature in this model, so I will keep increasing the order polynomial fit to check.

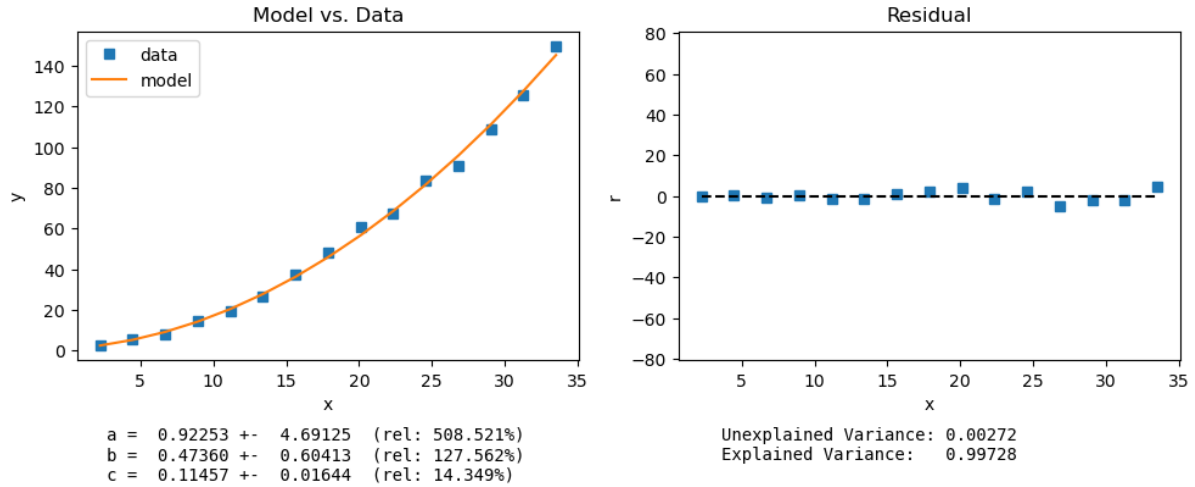

Figure 3: *Degree 2 Model Fit for Toyota Prius*

Figure 3 depicts the fit of a degree 2 polynomial in predicting braking distance based on car speed. The explained variance in the model is much stronger than the Figure 2, and the pattern

in the residual plot is now not nearly as profound. One issue with this plot is that the margin of error in the bias term and the linear coefficients are high. I will continue to investigate higher order polynomial fits while keeping this phenomenon in the back of my head.

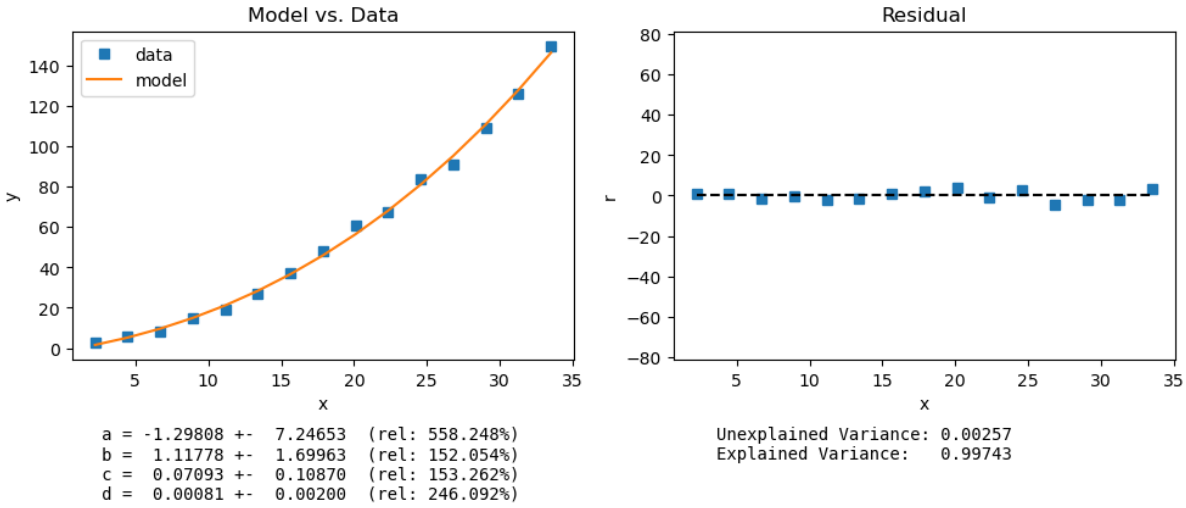

Figure 4: *Degree 3 Model Fit for Toyota Prius*

Figure 4 depicts the fit of a degree 3 polynomial in predicting braking distance based on car speed. The explained variance is stronger in this polynomial, but it is not nearly as profound of an increase from 3 degrees to 2 degrees as it was from 1 degree to 2 degrees. Because of the significant increases in the margin of errors in the coefficients, I do not feel nearly as comfortable choosing this model as I would choosing the 2-degree polynomial featured in Figure 3.

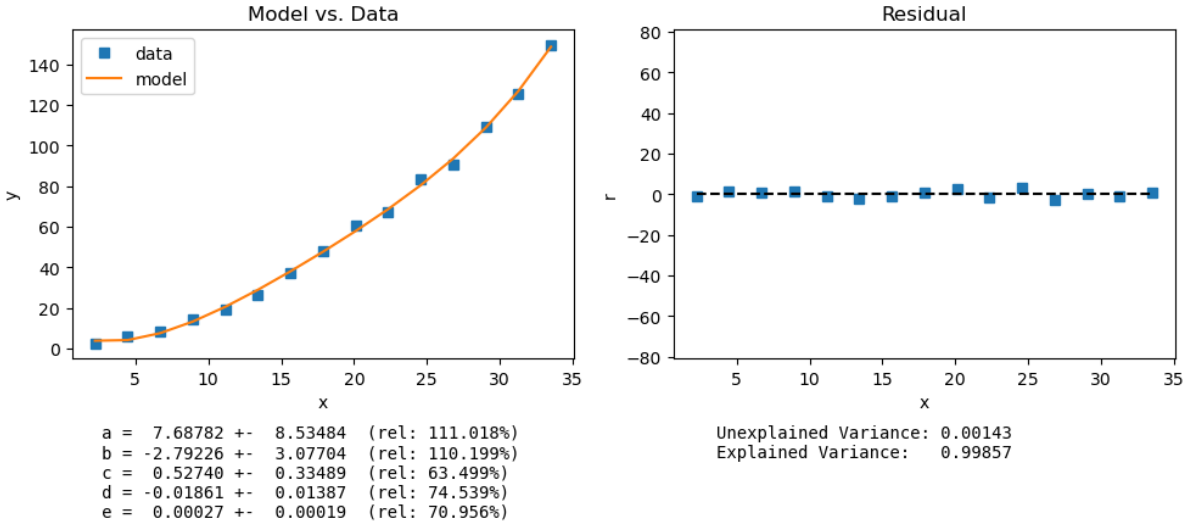

Figure 5: *Degree 4 Model Fit for Toyota Prius*

Figure 5 depicts the fit of a degree 4 polynomial in predicting braking distance based on car speed. The explained variance is slightly enhanced in comparison to the degree 3 polynomial, but the enhancement isn't by any practically significant margin. Moreover, the coefficient

for the 4th-degree variable is not large by any means, suggesting the model was already performing pretty well and did not need any difference.

Out of all the polynomials that we have observed so far, I believe that the degree 2 polynomial best fits the data. My reasoning for this observation is because of both the lack of significant drop-off in the explained variance in the degree 2 polynomial in comparison to the degree 3 polynomial and also the fact that margins of error in all of the coefficients rise significantly in the degree 3 polynomial. In Figure 6, the UVR dropoff can be seen, and it showcases how the UVR drop-off after hitting the second degree is not really significant.

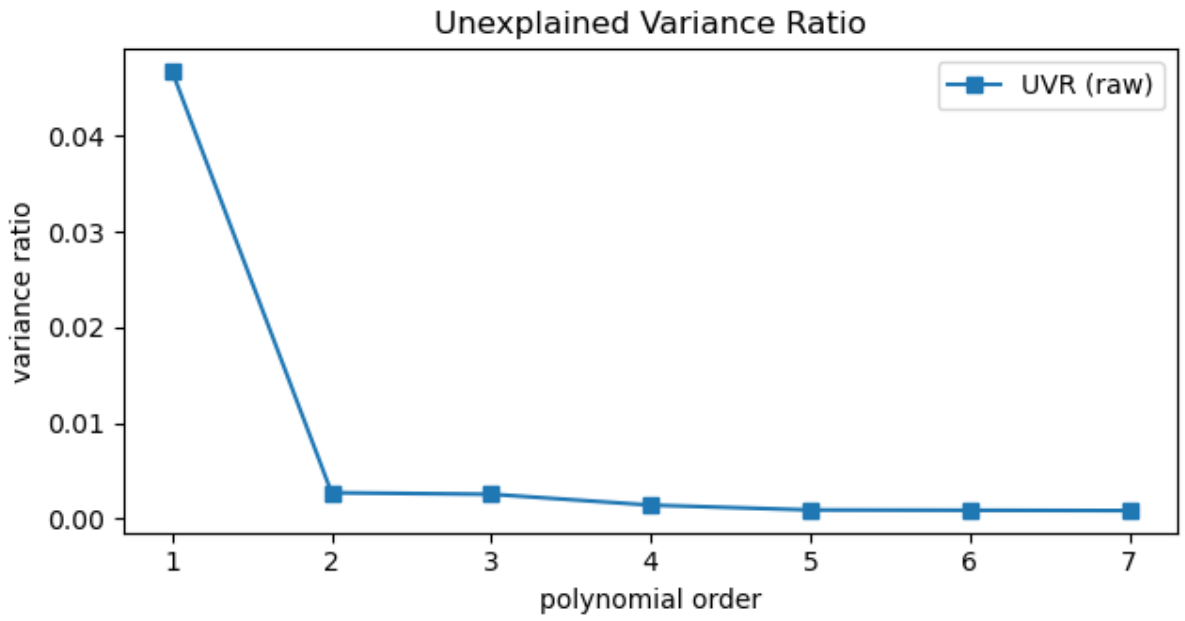

Figure 6: *Toyota Polynomial 1-7 UVR Drop-Off*

On the topic of margins of error, however, the degree 2 polynomial with all coefficients existing in the model does have some coefficients that I believe we should have removed. The margins of error for both the bias term and linear term are large, and it also does not make any conceptual sense to have a bias term other than 0, because if a car is traveling at 0 m/s, the braking distance should be 0 m every single time. Conceptually, if the goal is to predict the exact braking distance of the car, I should make the bias term equal to 0. If the goal of our model is to highlight braking performance, keeping the bias term equal to 0 is correct, so I will proceed with that value equal to 0. Hypothetically, however, if the goal of our model was to be precautious in our modeling and advise a driver on the distance where they should come to a full stop to avoid a serious accident, having a bias term equal to a value like 10 would be smart, because over-predicting and having negative residuals throughout the range of our model would be much more important. Nevertheless, I will continue testing without a bias term.

The EVR for the degree-2 model portrayed in Figure 7 without the bias term is barely smaller than the EVR in the degree-2 model with all coefficients, but not by a significant margin whatsoever. By running a Fisher test comparing the polynomial with the bias term against the polynomial without the bias term, I was able to infer that the 2-degree polynomial without the bias term performed better, concluding that the inclusion of the bias term in the polynomial might have been accommodating for some of the dataset's random noise, rather than genuine

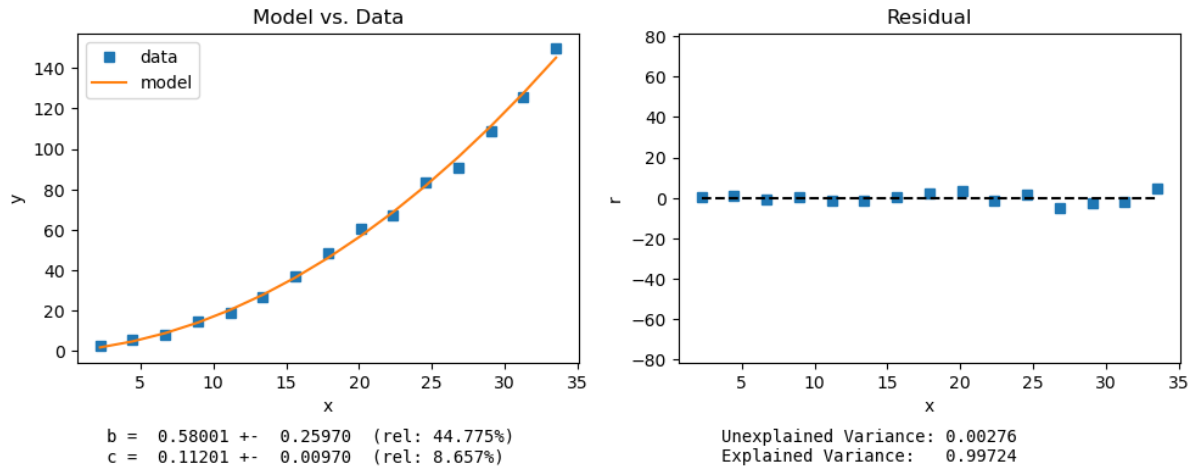

Figure 7: *Degree 2 Model Fit for Toyota Prius without Bias Term*

underlying patterns. Hence, I will prefer the 2-degree polynomial lacking the bias term.

Moreover, by conducting the same study and evaluating whether the 2-degree polynomial without the bias term *and* without the linear term performs better than the polynomial without just the bias term, the Fisher test concludes that the noise captured by the polynomial with both the linear and quadratic terms likely does not capture random noise, but instead significant and genuine underlying patterns in the data instead.

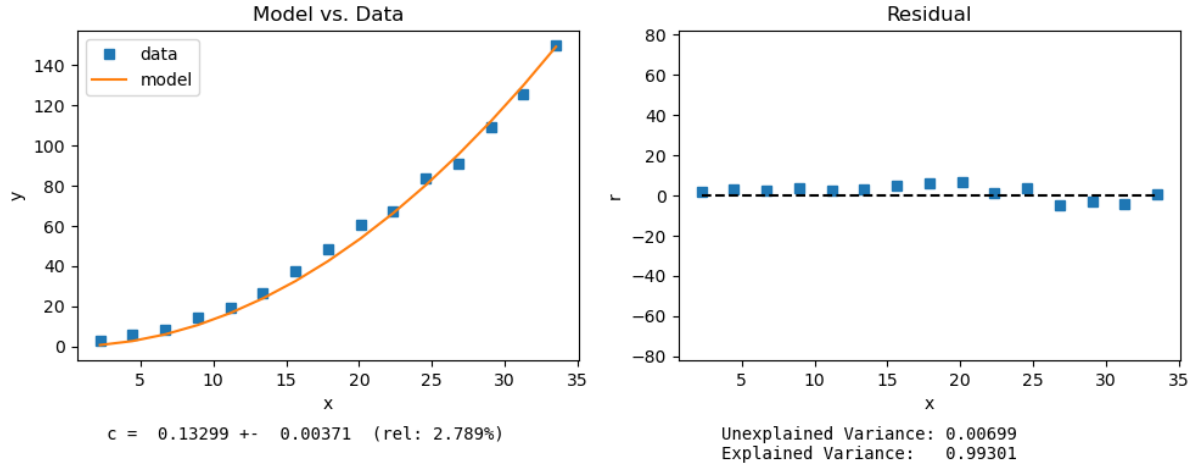

Figure 8: *Degree 2 Model Fit for Toyota Prius without Bias and Linear Terms*

Figure 8 showcases a plot of the polynomial fit, and while the EVR is strong, it has a somewhat noticeable drop-off in comparison to the model in Figure 7, which leads me to believe that the model with the linear and quadratic terms is the best model fit to the data—assuming the goal is to display braking distance based on car speeds for competitive purposes. The equation for this model is:

$$\hat{braking\ distance} = 0.58001 \cdot (speed) + 0.11201 \cdot (speed)^2$$

With this equation, if a Toyota Prius were to slam on the breaks going 100 mph (44.704 m/s),

the braking distance would be 249.78 m (0.1552 miles). There are considerations I am taking into fact to investigate reliability of this extrapolated prediction.

The quadratic increase makes sense, because the Toyota Prius is dissipating much more kinetic energy going 100 mph than can be seen in the measured data points.

At highway speeds around 70 mph, cars brake around 107 m (obviously with a big margin of error depending on the type of car we are talking about) when not taking into account a reaction time. Reaction time is not something to take into account due to the way that this experiment was conducted. The braking distance of 249.78 m is likely a little bit of an overestimation, but due to the quadratic nature of the data, the braking distance is probably still very large nevertheless. Extrapolating predictions from our polynomial model is definitely less reliable in these cases, because the data our model was trained on did not include cars going at speeds of 100 mph. Real-world conditions may likely kick in at this point too, such as the actual suitability of a the car's braking system to be able to come to such a harsh stop while remaining completely in tact.

Tesla Model 3

The data for the braking distance vs. speed graph for the Tesla Model 3 according to Figure 1 follows a similar trend to the Toyota, but there seems to be a little bit more noise and less of a smooth curvature that can be seen. Knowing what we know now about the Toyota dataset, because the patterns in both datasets are similar, I will first explore only the 2nd and 3rd degree polynomial curve fits to the data, because we certainly already know that the function of best fit is not linear based on the pattern.

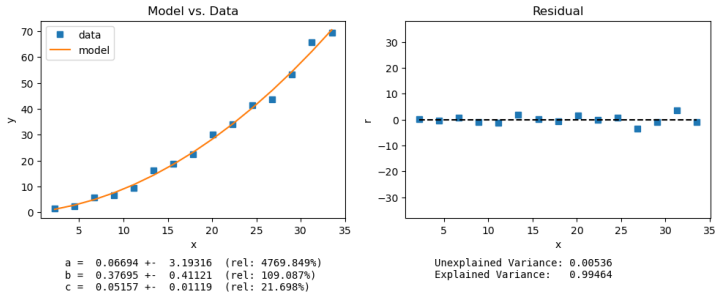

Figure 9: *Degree 2 Model Fit for Tesla Model 3*

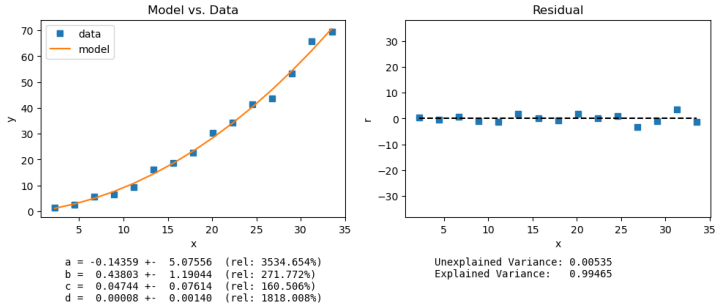

Figure 10: *Degree 3 Model Fit for Tesla Model 3*

Figure 9 showcases the degree-2 polynomial fit for the Tesla Model 3 braking distances based off time, whereas Figure 10 showcases the degree-3 polynomial fit. Its distances are around half

of what the Toyota Prius braking distances are. Both models appear to fit the data considerably well, but the UVR decrease between the second degree polynomial and the third degree polynomial is not large at all. Using the Fisher Test, we can conclude that the inclusion of a 3rd-degree term in the polynomial of best fit might have been accommodating for some of the dataset's random noise, rather than genuine underlying patterns, meaning that I would prefer the 2nd-degree polynomial for more general purposes.

The bias and linear terms have a large margin of error, and conceptually speaking, it does not make sense to include the bias term if the point of this model is to compare braking distances between the Tesla Model 3 and other cars. So, I will remove that going forward, but I want to compare the performances of the 2nd-degree polynomial fit with and without the linear term.

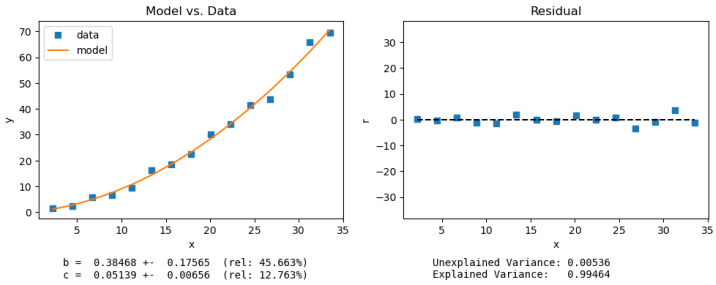

Figure 11: *Degree 2 Model Fit for Tesla Model 3 Without Bias Term*

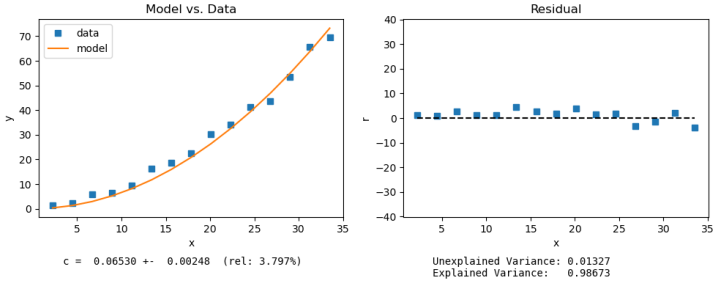

Figure 12: *Degree 2 Model Fit for Tesla Model 3 Without Bias and Linear Terms*

According to a Fisher test, it is stated that the complex model (the quadratic with the linear term) captures patterns in our data that our not captured by our simple model (the quadratic without the linear term), with an F-score of 19.18 and a P-Value of 0.0007. I am noticing some noise in the data that may be a result of time-recording human error with the braking systems for these cars in the collected data where the car is moving faster in Figure 12, but I will acknowledge that this may be true, note that it may cause extrapolation errors due to the residual plot in Figure 12, and accept the model incorporating both a linear and quadratic term as the knowledge of truth–at least within this domain of data. The equation can be denoted as the following:

$$\hat{braking\ distance} = 0.38468(speed) + 0.05139(speed)^2$$

Using this model, when a Tesla Model 3 is traveling at 100 mph (44.704 m/s), the resulting speed would be 119.89 meters (or 0.074 miles). However, it's uncertain if this solution actually

is valid, because of conditions that may appear in the circumstance that a Tesla Model 3 is traveling at 100 mph. Estimating braking distance at high speeds is challenging due to the risks and uncertainties associated with extrapolation beyond the range of observed data. Real-world braking distance is influenced by various factors, including road conditions, tire grip, and the braking system's efficiency, which may not be fully accounted for in our model. Other variables may be important to take into consideration at these speeds that we haven't seen as impactful within the domain of our collected data, so it's uncertain that we can trust the data at this level.

Exercise 2

One fundamental way of analyzing computer algorithms is to determine their characteristic run time as a function of their input size N. For example, you might ask "how long do es it take to sort a list of N strings?" Frequently-encountered runtimes include things like:

Notation	Description	Example
$O(\log(N))$	Logarithmic Time	Phone book lookup (best case)
$O(N)$	Linear Time	Vector-vector product (typical case)
$O(N^2)$	Quadratic Time	Matrix-vector product (typical case)
$O(N^p)$	Polynomial Time	Matrix-matrix product (typical case)
$O(N!)$	Factorial Time	Traveling salesman (worst case)

The data sets algorithm-runtime-1.xlsx and algorithm-runtime-2.xlsx contain measurements of the runtime of two different algorithms for various input sizes. For each data set:

- find a polynomial model that fits the data reasonably well

- justify the order of, and terms in, the polynomial you chose

- record the polynomial coefficients you obtained (for future use)

- if you had to decide which algorithm to use to perform calculations on an input of size 10,000, how would you approach this decision?

Dataset 1:

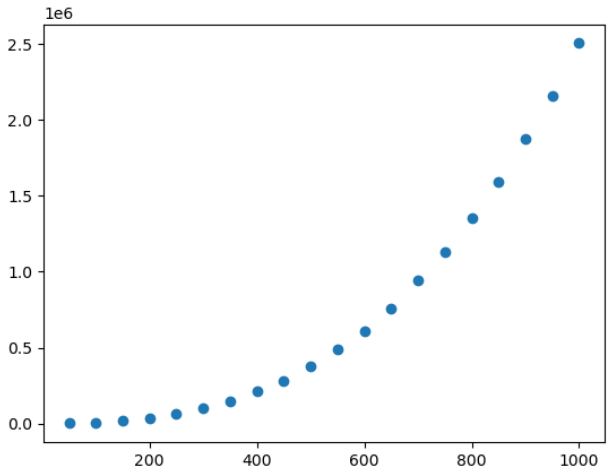

Figure 13: *Plot of Algorithm 1 Runtimes*

The runtimes in the first algorithm have a curved pattern associated with them according to Figure 13, so I will start checking best fit models at the second degree polynomial.

Figure 14 showcases a strong sinusoidal pattern in the residual. This presents a possible issue of underfitting that is apparent enough both in the residual and the actual fit plot to the point where I do not trust the degree 2 polynomial as an accurate model representing the fit of the data and the runtime of the algorithm of interest, even though our EVR is high at 0.99772.

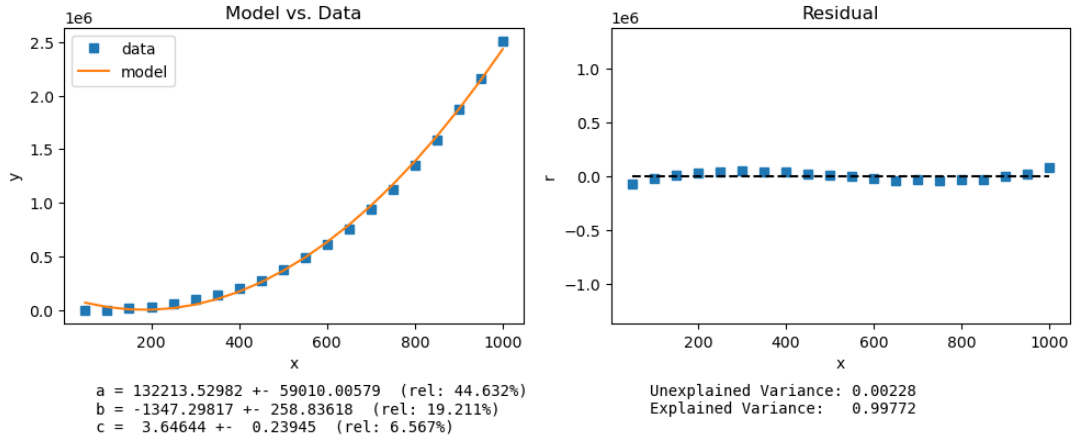

Figure 14: *Algorithm 1: Degree 2 Polynomial Fit and Residual*

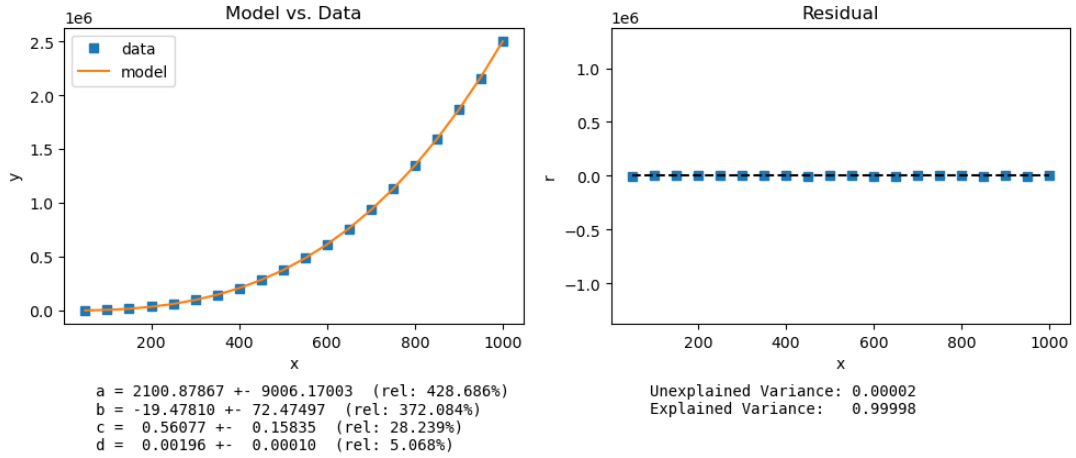

Figure 15: *Algorithm 1: Degree 3 Polynomial Fit and Residual*

Observing Figure 15, I notice that the issue with a strong pattern appearing in the residual has decreased significantly with the degree 3 polynomial in comparison to Figure 14. This alone leads me to believe that the 3-degree polynomial is a stronger model for runtime prediction as opposed to the degree-2 polynomial from Figure 14. However, I am going to check the 4-degree polynomial and also conduct a Fisher test to see which one I will pick and if the increase in EVR for degree 4 is significantly bigger to the point where it captures patterns in the data that the 3-degree polynomial does not catch.

The EVR for the 4-degree polynomial as seen in Figure 16 rounds to the identical 0.99998 value at 5 digits in comparison to the 3-degree polynomial in Figure 15 at 0.99998. Using the Fisher test to compare the performance of these 2 models, it is revealed through the F-number of 0.6227 that the degree 4 polynomial only captures insignificant and likely random patterns in the data and does not capture any additional significant pattern in the data that the 3 degree polynomial model does not capture.

However, I believe that the 3 degree polynomial model can be improved, as there appears to be a large margin of error in the bias term and linear coefficient.

Figure 17 showcases the degree 3 polynomial without the bias term fitted onto the same piece

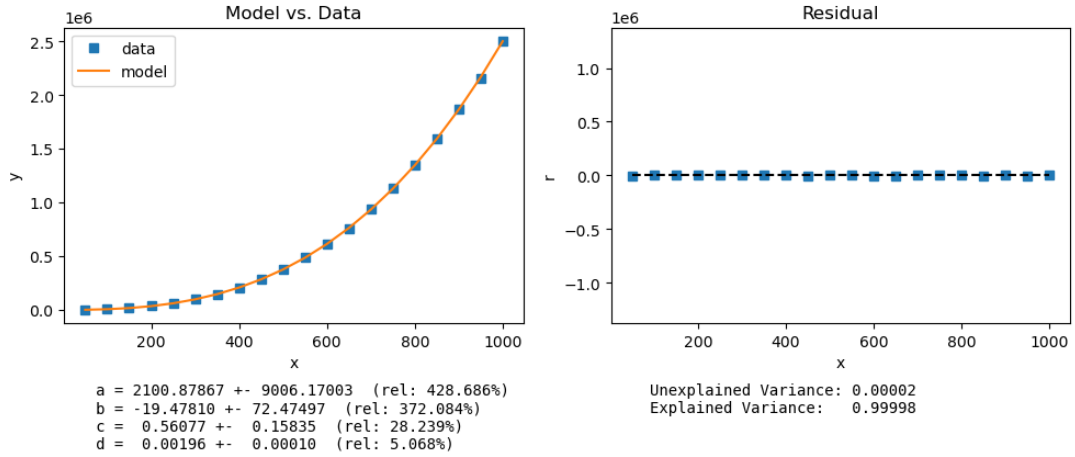

Figure 16: *Algorithm 1: Degree 4 Polynomial Fit and Residual*

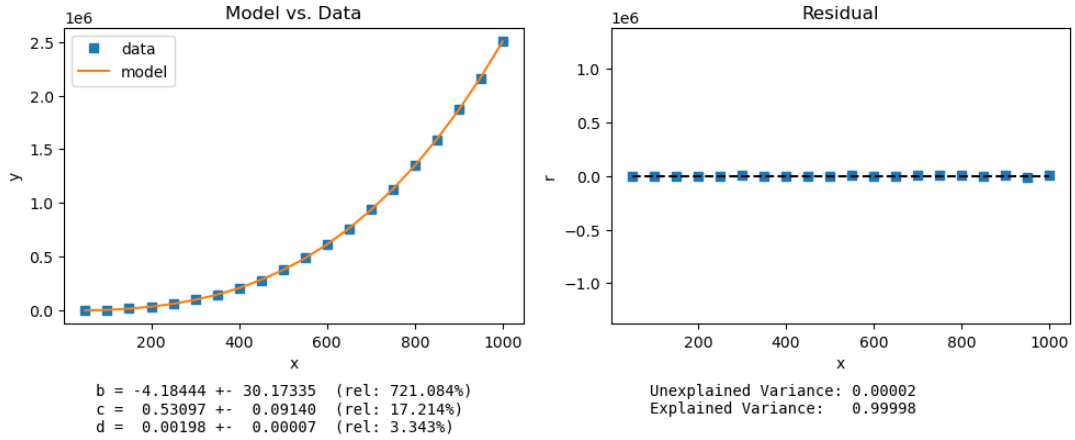

Figure 17: *Algorithm 1: Degree 3 Polynomial Fit and Residual without Bias Term*

of data. We can see that the EVR rounded to 5 decimal points also is 0.99998. The residual has no obvious structure to it and appears to fit to the model very well in Figure 17. The F-value is 0.2177 and the P-value is 0.6471 when comparing these two models using a Fisher Test, showcasing that we cannot conclude that the model with the bias term captures significant patterns in the data that the model without the bias term does not capture.

It does make conceptual sense to have a bias term on an algorithm runtime graph, because there may be a constant runtime that appears on every iteration of the program getting booted up and starting. Since we are uncertain about what that value may be and it may fluctuate on each run, we should proceed without it with creating a model that showcases the performance of the algorithm at various inputs ignoring a bias term to better understand what the true values of every other coefficient are.

The margin of error for the linear term is still extremely large, so I want to evaluate a model without it and check its performance.

The EVR of the model without the bias and linear terms in Figure 18 appears to yield the same 5-digit rounded EVR of 0.99998 in comparison to the model in Figure 17, and it is more simple because it involves less terms. The residual plot showcases an extremely close fit, and the

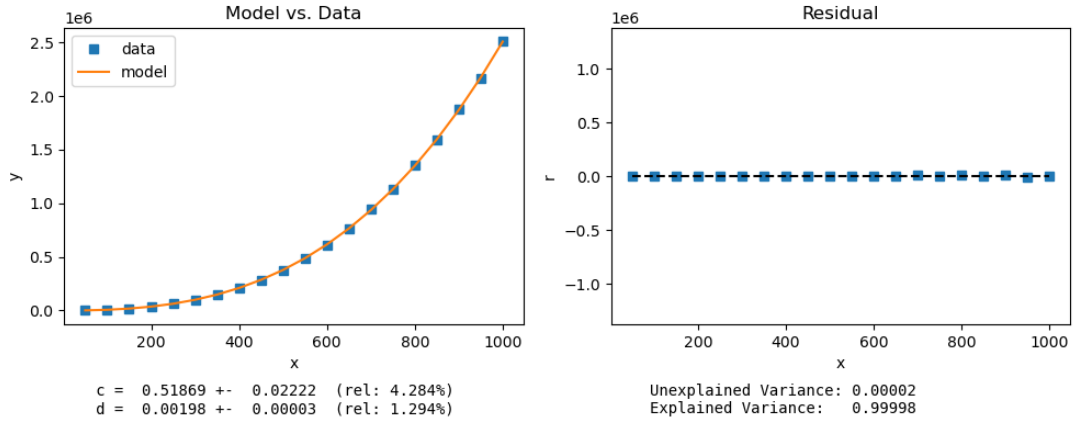

Figure 18: *Algorithm 1: Degree 3 Polynomial Fit and Residual without Bias and Linear Terms*

plot with the polynomial fit in Figure 18 shows how the model is strong and explains most of the data variance. The F-value is 0.0769 and the P-value is 0.7848 when comparing the more simple model in Figure 18 to the model including the linear term in Figure 17, showcasing that we cannot conclude that the model in Figure 17 captures true patterns in the data that the model in Figure 18 does not catch.

The margin of error for both terms is small, but I still do want to check if the polynomial containing just the cubic term performs better than the model including both the cubic and quadratic terms.

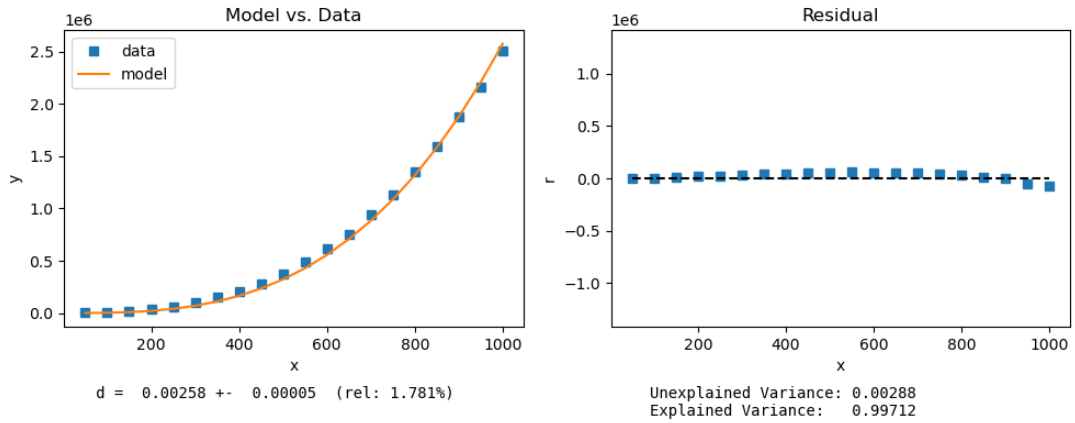

Figure 19: *Algorithm 1: Degree 3 Polynomial Fit and Residual with just Cubic Term*

The model with just the cubic term in Figure 19 does not appear to perform nearly as well as the model in Figure 18, with the residual showcasing a defined curved pattern and the data deviating from the model in the left figure in Figure 19. The model here also had a smaller EVR of 0.99712 as opposed to 0.99998, and the Fisher test comparing the simpler Figure 19 model with the more complex Figure 18 model yields an F-value of 2179.3253 and a P-Value of approximately 0, allowing me to conclude that the model with the cubic and quadratic terms performs better in predicting the runtime of this algorithm in comparison to the model with just the cubic term. Noting that runtime is in milliseconds, the equation for this model I choose

is:

$$\hat{runtime} = 0.51869(input\ count)^2 + 0.00198(input\ count)^3$$

Hence, the algorithm's big-Oh notation is $O(n^3)$.

Dataset 2:

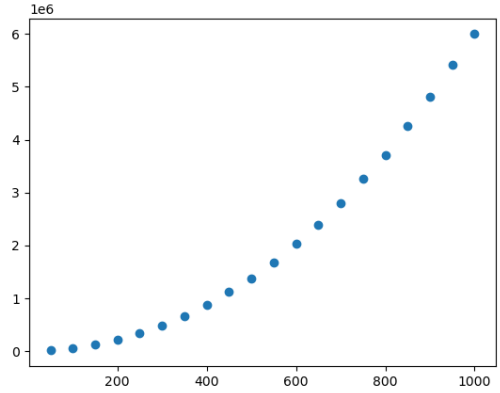

Figure 20: *Algorithm 2: Raw Data Scatterplot*

The raw data appears to be curved to some extent in Figure 20, so I will first test quadratic and cubic polynomials to see how well they perform in accurately predicting the data. I will not test a linear function, because the data is clearly not linear.

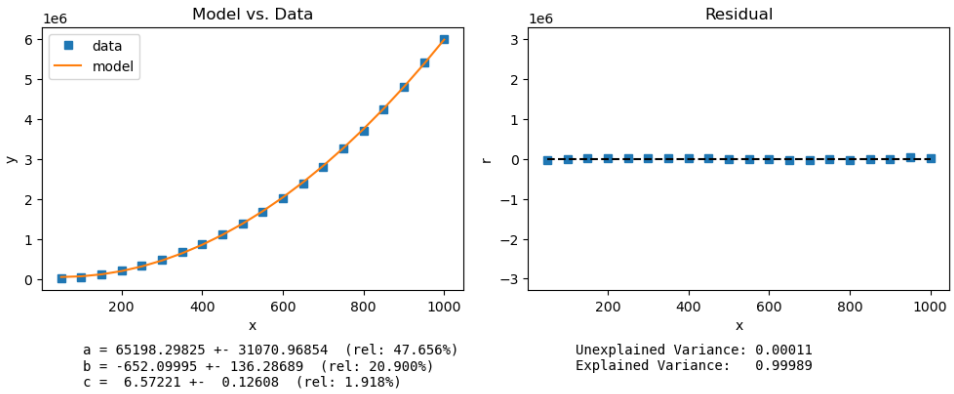

Figure 21: *Algorithm 2: Quadratic Polynomial Model Fit to Data and Residual Plot*

The model performs well in predicting the data with an EVR of 0.99989 as we can see in Figure 21. Additionally, a somewhat sinusoidal structure appears in the residual, so it gives me an excuse to check whether or not a cubic would perform better on the prediction task.

Figure 22 showcases the fit of a cubic polynomial with an EVR of 0.99997, which outperforms the quadratic. The fit appears to be strong, so I will conduct a Fisher test to see if the performance boost is significant or just potentially random. The Fisher test returns an F-statistic of 49.7398 and a P-Value of approximately 0, allowing me to conclude that the cubic likely captures patterns in our data that are not captured in the quadratic that do not appear to be due to random chance.

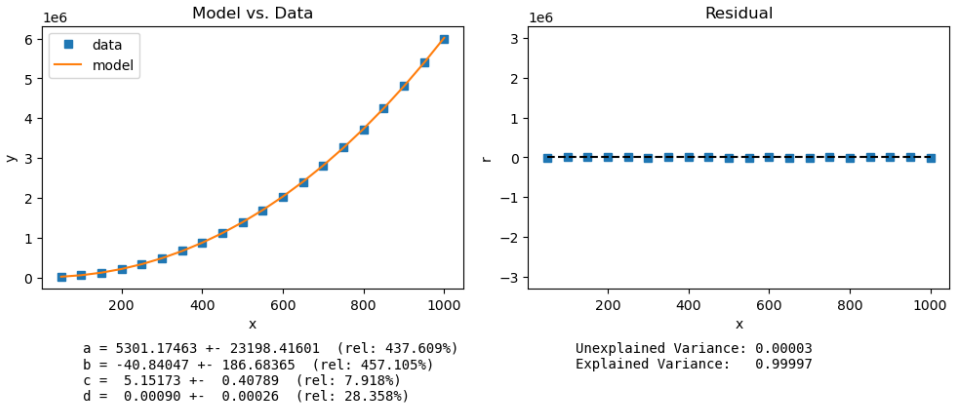

Figure 22: *Algorithm 2: Cubic Polynomial Model Fit to Data and Residual Plot*

Because of this significant performance boost, I will check how a 4-degree polynomial compares to the cubic.

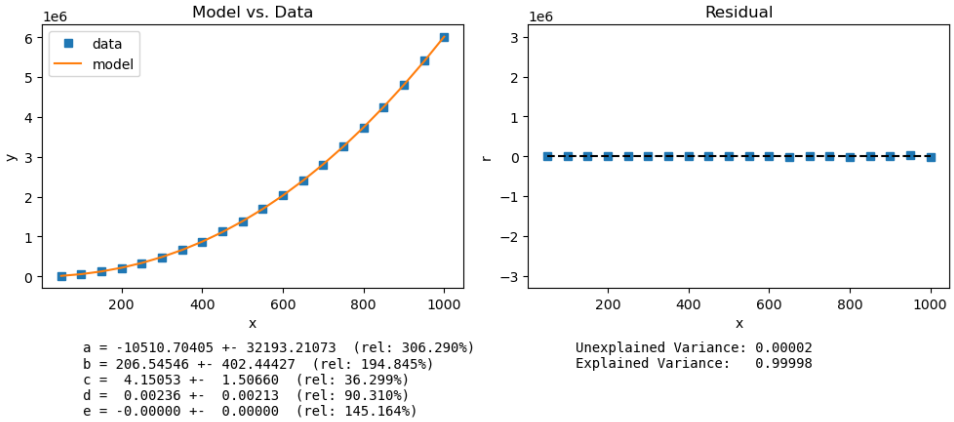

Figure 23: *Algorithm 2: 4-Degree Polynomial Model Fit to Data and Residual Plot*

The EVR increases by 0.00001 for the 4-degree polynomial model from Figure 23 in comparison to the 3-degree polynomial in Figure 22. The model fits to the data extremely well, but I will check whether or not this boost in performance is significant through a Fisher test. The Fisher test comparing these two models yields an F-statistic of 1.8982 and a P-Value of 0.1885 greater than 0.05, signifying to me that the cubic polynomial is preferred in my opinion, because it is likely that the 4-degree polynomial's increase in EVR may just be due to random chance and we could not conclude otherwise.

In the cubic model, however, there seems to be values that have high margins of error in their coefficients, specifically the bias term and linear term. I will test out some different model fits without these values to evaluate their performance and check whether or not they are needed in this model, as 0 is included in both of their ranges.

The model without the bias term, as ca be concluded from Figure 24, showcases that the model has an EVR of 0.99997, still performing very well. A Fisher test yields an F-statistic of 0.2089 and a P-value of 0.6589, allowing me to not be able to conclude that the model with the bias term performs better than the model without the bias term. Hence, I will remove that term and compare the model that has no bias term with other cubics without further terms moving

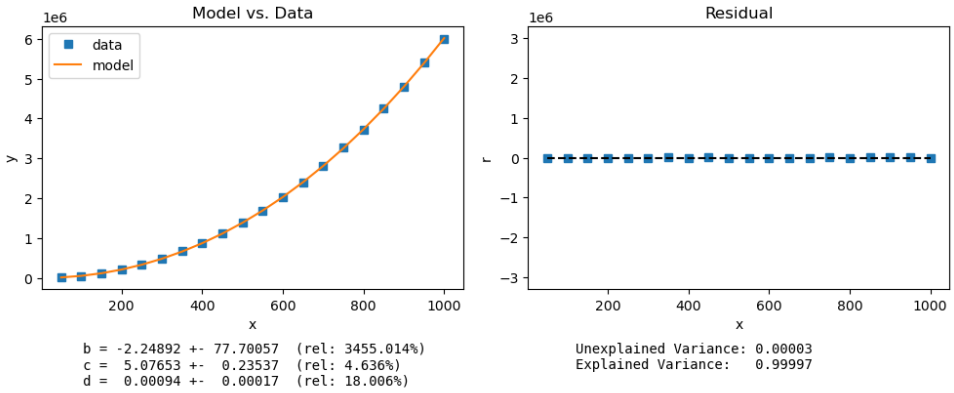

Figure 24: *Algorithm 2: Cubic Polynomial without Bias Term Model Fit to Data and Residual Plot*

forward. I will now try removing the linear term, because the margin of error is excessively large.

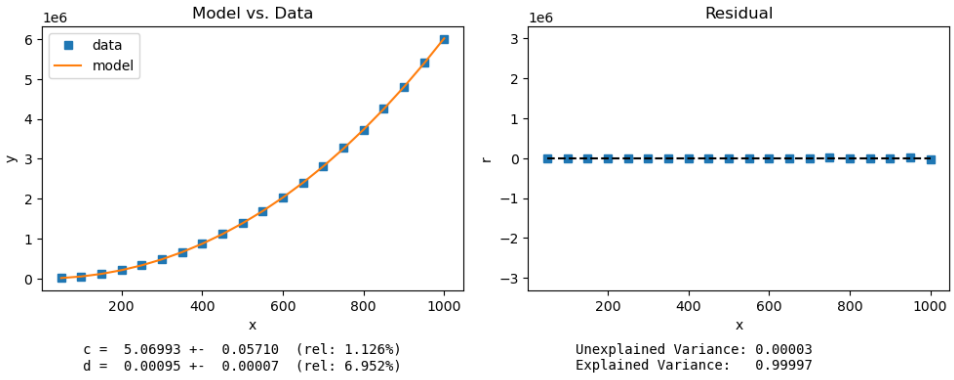

Figure 25: *Algorithm 2: Cubic Polynomial without Bias Term Model Fit to Data and Residual Plot*

The cubic without both the linear and bias terms in Figure 25 performs well with an EVR of 0.99997, and it's not a drop-off rounded to the fifth digit when compared to the EVR of the model in Figure 24. Conducting a Fisher test, we obtain an F-statistic of 0.0034 and a P-Value of 0.9545, allowing me to not be able to conclude that the more complex term is better than the more simple model. Hence, I will proceed with the cubic model that contains only the quadratic and cubic terms, as it is more simple and likely is strong because of its ability to capture patterns in the data instead of just being able to capture that and noise significantly.

The cubic model with just the cubic term in Figure 26 performs noticeably worse than the cubic model with the quadratic and cubic terms from Figure 25. From visualizations alone, since there's a strong pattern in the residual, we can conclude immediately that this model does not fit the sample data nearly as well, and because of that, it likely does not predict the performance of the algorithm well at all. Hence, I will proceed with the cubic model containing the quadratic term from Figure 25 as my model of choice for predicting algorithm runtime based on input count.

The model for the second algorithm that I am choosing is as follows:

$$\hat{runtime} = 5.06993(input\ count)^2 + 0.00095(input\ count)^3$$

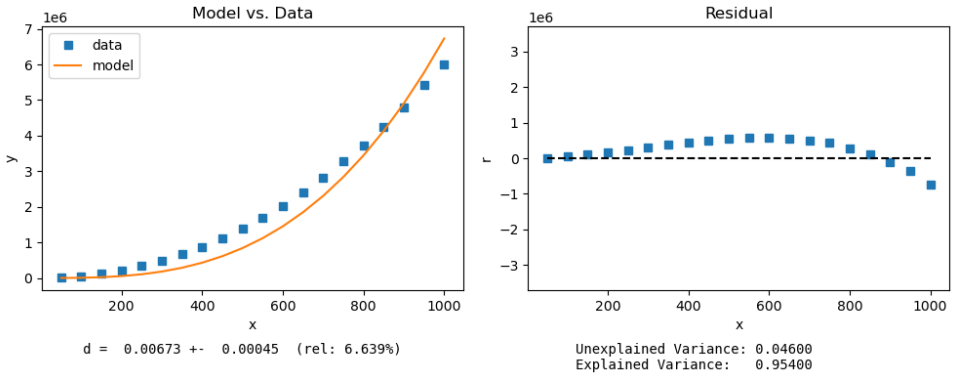

Figure 26: *Algorithm 2: Cubic Polynomial without Bias Term Model Fit to Data and Residual Plot*

Based on my model of choice, the second algorithm has a time complexity of $O(n^3)$.

Comparing Algorithm Performances on 10,000 Inputs:

My first way of analyzing these algorithms to see which performs better would be to default to their big-Oh notations. Both algorithms are $O(n^3)$, so we would need to dive deeper into the $T(n)$ functions to see which algorithm performs better. We do actually some sort of resemblance of a $T(n)$ approximation explicitly calculating runtime as dependent variables that were modeled through minimizing the sum of squares with certain terms in our function, and while these functions aren't derived from a proof of algorithm time complexity but instead through regression modeling of based on input counts and algorithm runtimes, we are still able to calculate runtime approximations through extrapolation of a regression model.

Extrapolation isn't nearly as bad in this scenario as it was in our car example in Problem 1. There are not as many confounding variables that could also contribute to an increase in runtime as there are in the car scenario for braking distance at higher speeds. We have no clue if the algorithms are supposed to behave differently with more inputs, but assuming they do not, we can obtain the performances pretty well using this methodology.

Algorithm 1 Runtime: 2,036,386.879 seconds

Algorithm 2 Runtime: 1,456,198.270 seconds

Based on the polynomial regression models we constructed to estimate algorithm runtimes for both algorithm 1 and algorithm 2, it is clear that algorithm 2 outperforms algorithm 1 when considering an input size of 10,000. Our chosen cubic polynomial model for Algorithm 2 predicts a runtime of approximately 1,456,198.270 seconds for 10,000 inputs, while the equivalent model for Algorithm 1 predicts a longer runtime of approximately 2,036,386.879 seconds.

However, it's important to note the limitations of this conclusion. My models are derived from observed data and are based on the assumption that the algorithms behave consistently as the input size increases. While these models provide reasonable estimates, they are not guaranteed to hold true for all scenarios, especially when dealing with significantly larger input sizes or different computational environments. Additionally, the models do not take into account other factors that could impact runtime, such as hardware specifications, software optimizations, and algorithm-specific intricacies.

Algorithm 2 is likely to perform better than Algorithm 1 for an input size of 10,000 based on the polynomial regression models we constructed. However, this conclusion should not be taken at face value, and any practical decision regarding algorithm selection should consider additional factors and possibly involve empirical testing and validation on the specific hardware and software environment where these algorithms will be deployed. $T(n)$ functions derived from the intricacies of each algorithm would likely allow for a better analysis.