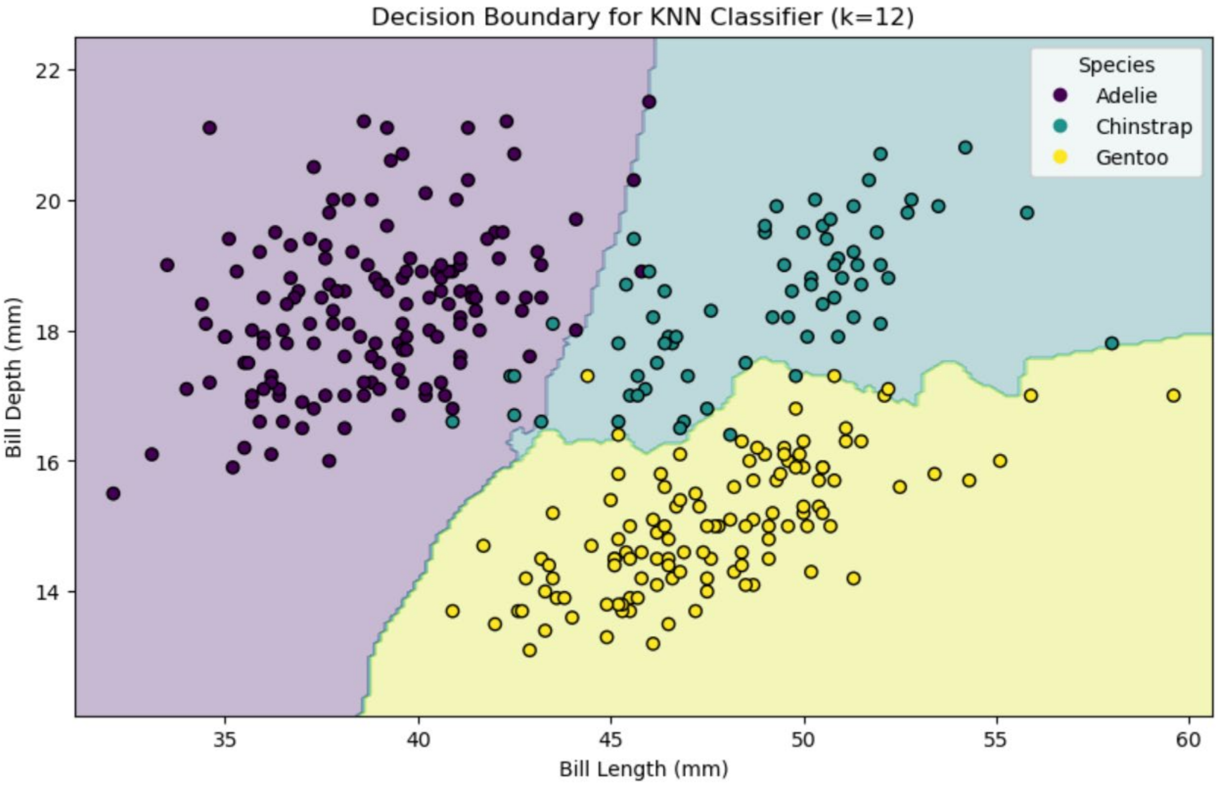
Problem #1

Using the seaborn Penguins dataset, set up a KNN to predict the species of penguin using
bill length and bill depth. Find the best k value, and plot a map (mesh graph) with this k
value.

The best value for k is: 12 with an accuracy of 0.99

As you can see, the KNN accurately predicts 99% of the points, with only a handful of points wrong.



#2. Import the tips dataset from seaborn. Find the most accurate polynomial (or linear) model to correlate total bill and size of party to tip as a percent of the bill. What is the underlying function for the model? Do the coefficients and degree make sense?

# Coefficient for total_bill: approximately -0.285

# Coefficient for size: approximately 0.916

# y intercept means the baseline tip percentage, hypothetically in this case, when the size of the party is 0 and the total bill is 0,

# the baseline tip is around 20% (19.53). which makes sense. I usually tip around 20% as well.

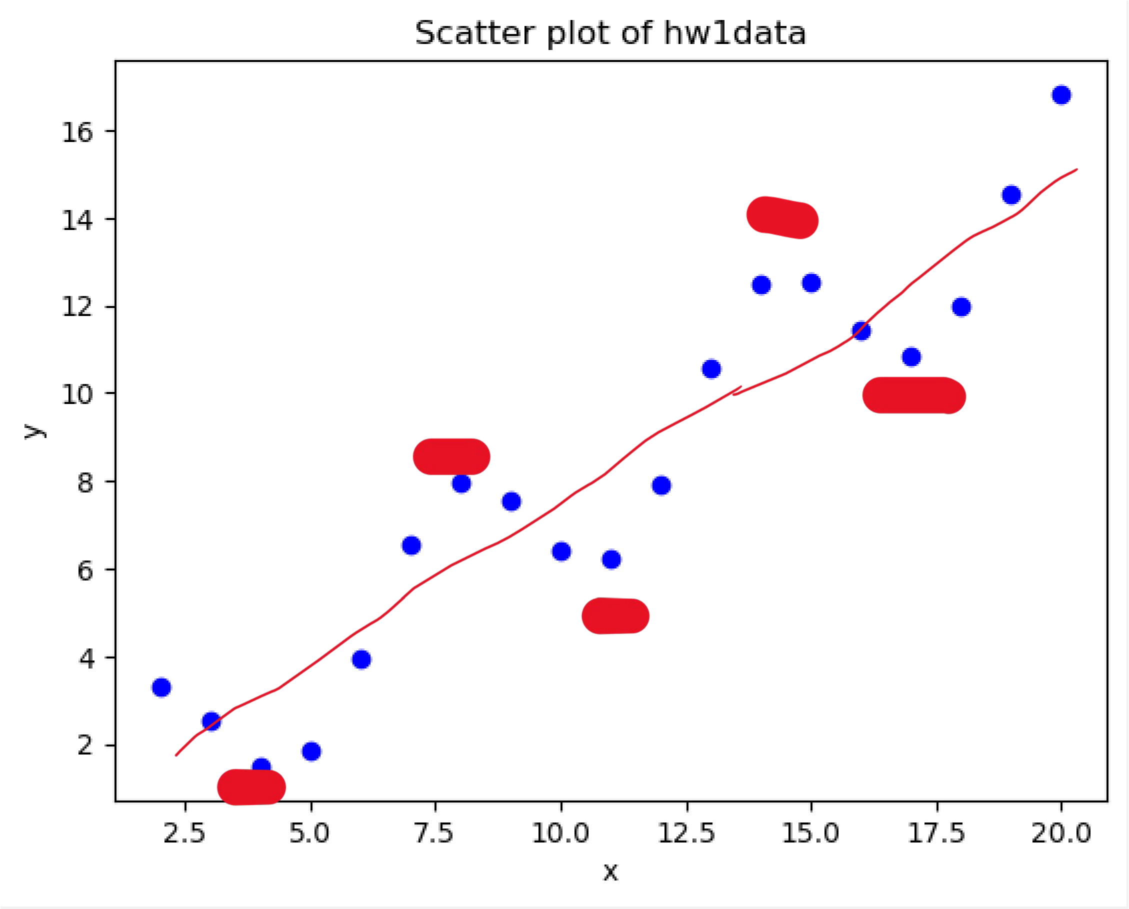
When the total bill goes up, people tend to tip less. This makes sense because people do not want to have to pay a fortune on top of their already expensive meals.

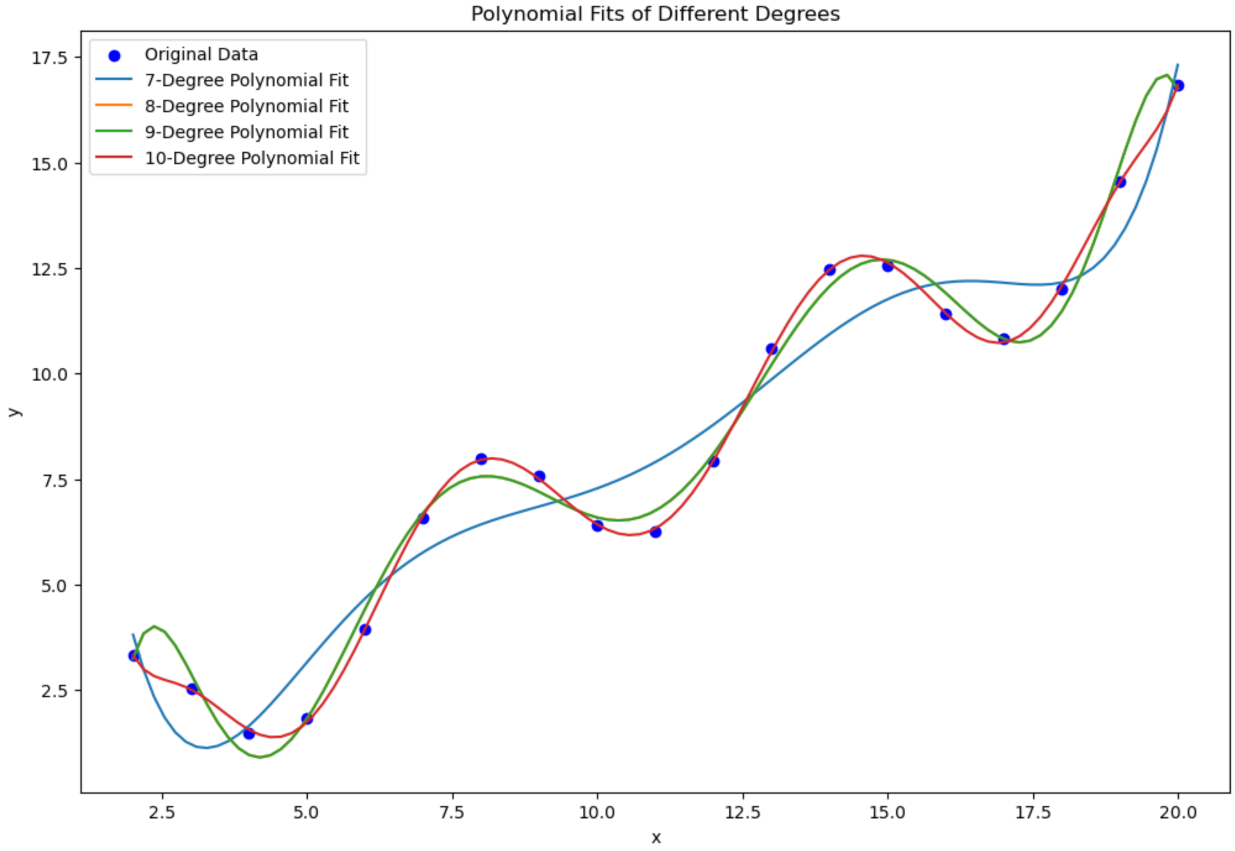
When the party size goes up, so does the tip. This makes sense because usually, more people are paying, or it's possible that the customer feels like they need to support the server/restraint more as they had to deal with a large party.

#3.

I tried doing a squareroot transformation of data, as well as log. Neither seemed to produce good results. I didn’t end up keeping this code in my python file.

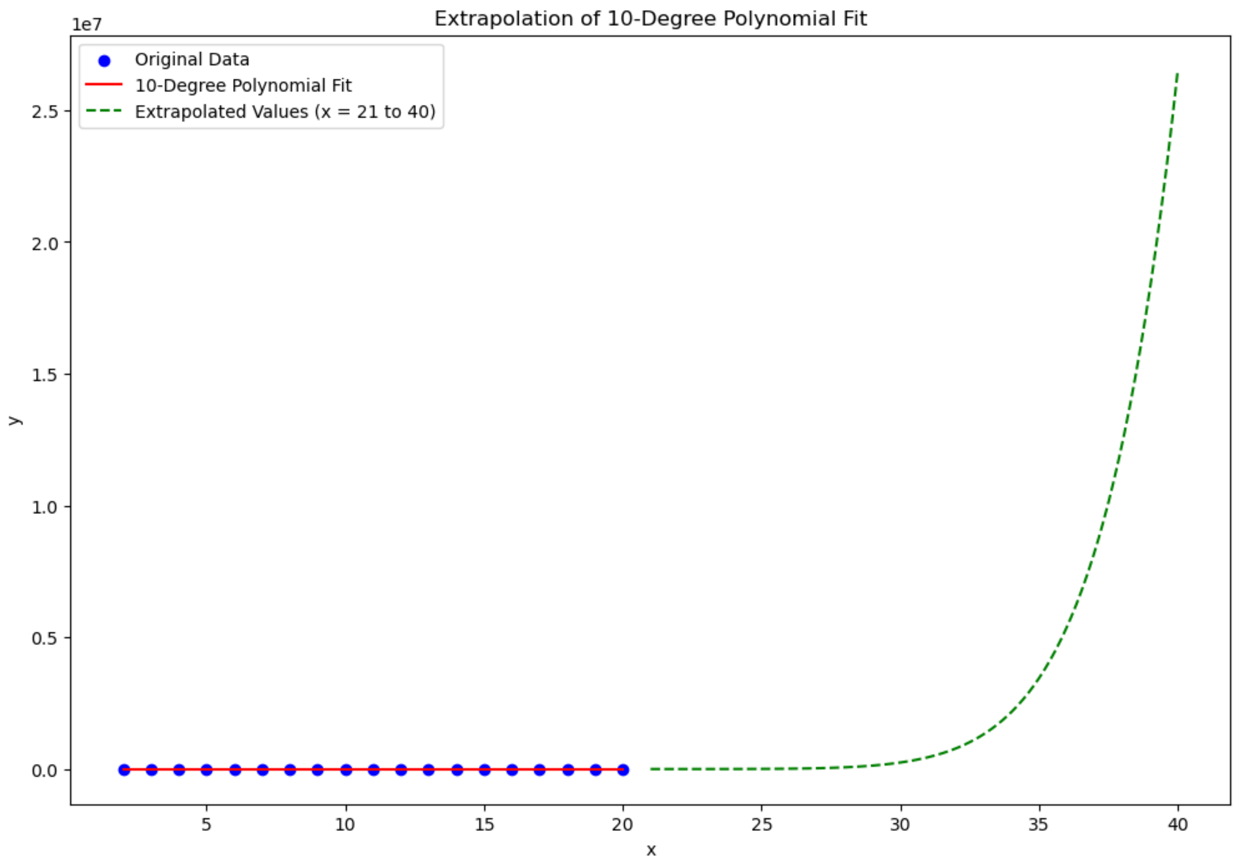
It seems the function is a high-degree polynomial just by looking at the local minimums and maximums.


Scatter plot of hw1data
16
14
12
10
8
6
4
2
y
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
x

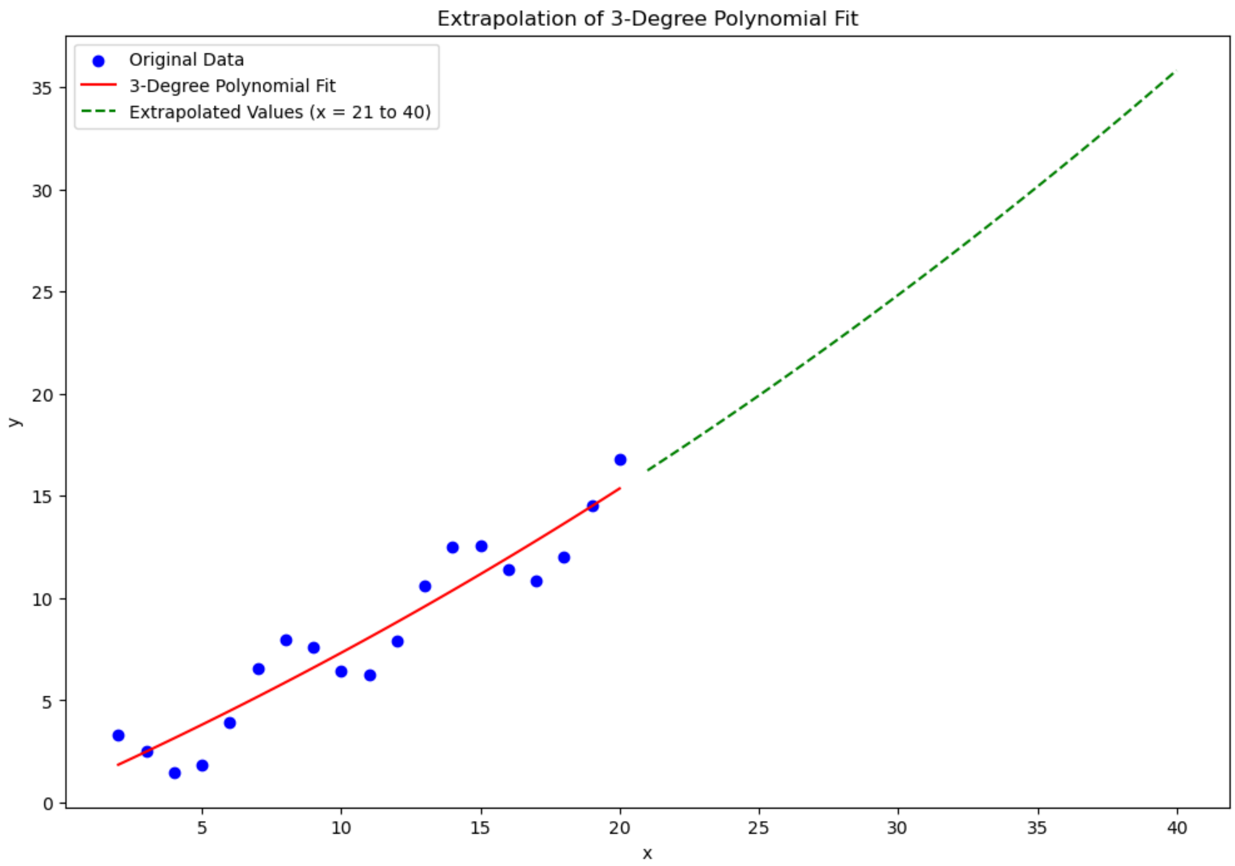


Note that the dictionary is off, these are for degrees of 7,9, and 11.

I just graphed the functions of degree +=2 from values of 7,11.



As you can see, this pattern does not follow the trend past about 28 on unseen data. It quickly goes into values of a million LOL.

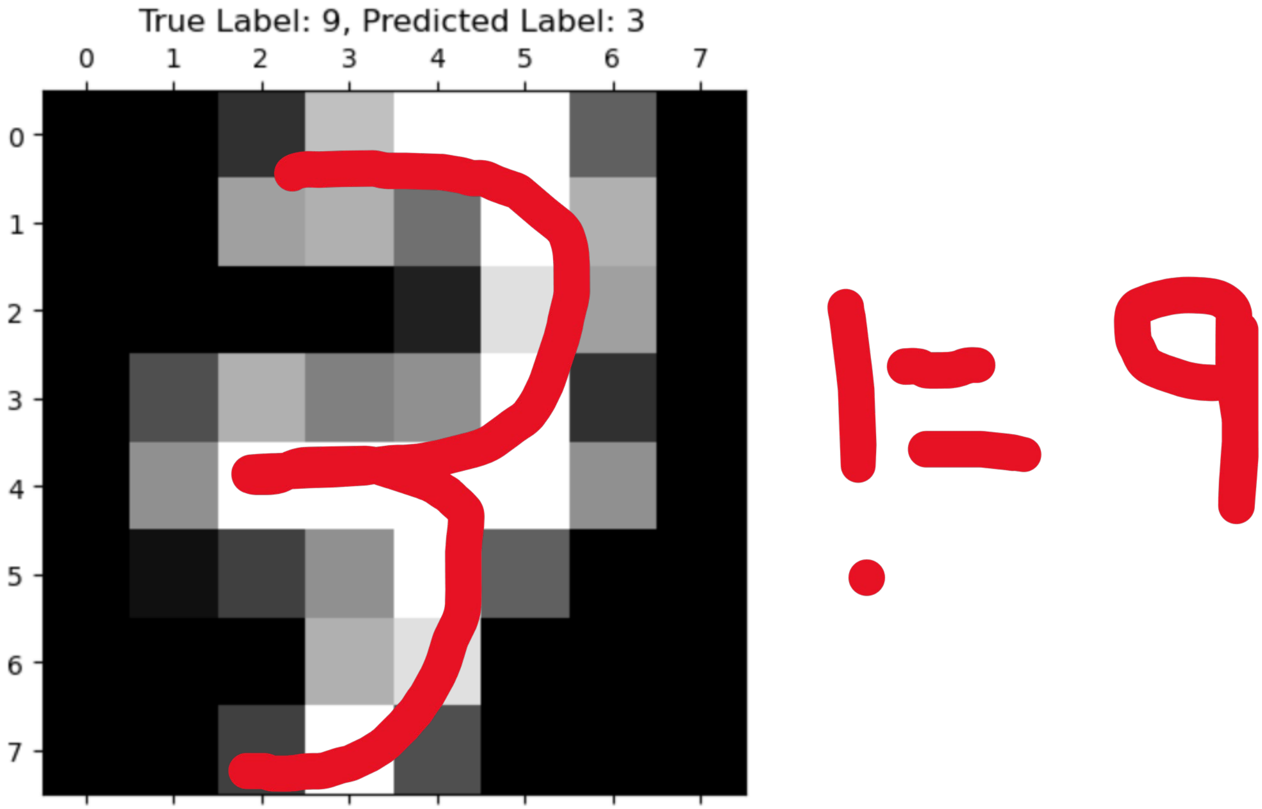


We can see that a third-degree polynomial fits the data a lot more in the sense of unseen data. It does not overfit the original data, so it does better on new data.

#4. N/A

#5.

Sklearn provides a dataset called load_digits which contains datapoints that, when
graphed in grayscale, feature images of numerals written by hand. Use plt.gray() and
plt.matshow(datapoints) to view these images.



```
Predicted Probabilities: [[3.64160501e-09 4.43298328e-10 1.28630643e-05 9.69996975e-01
  6.56319208e-11 4.71887956e-10 2.79530924e-10 2.72415005e-07
  1.89765578e-05 2.99709077e-02]]
```

It makes sense why the model predicted a 3, even though I think it’s a 3. That is the worst drawn 9 I have seen in 22 years of living.

The model was almost 97% sure it was a 3. I would say I'm 100% sure it’s a three. Did someone mislabel the data initially is what I'm wondering?

Also, it's interesting that the model gave it a 2.99 percent chance of being a 9.