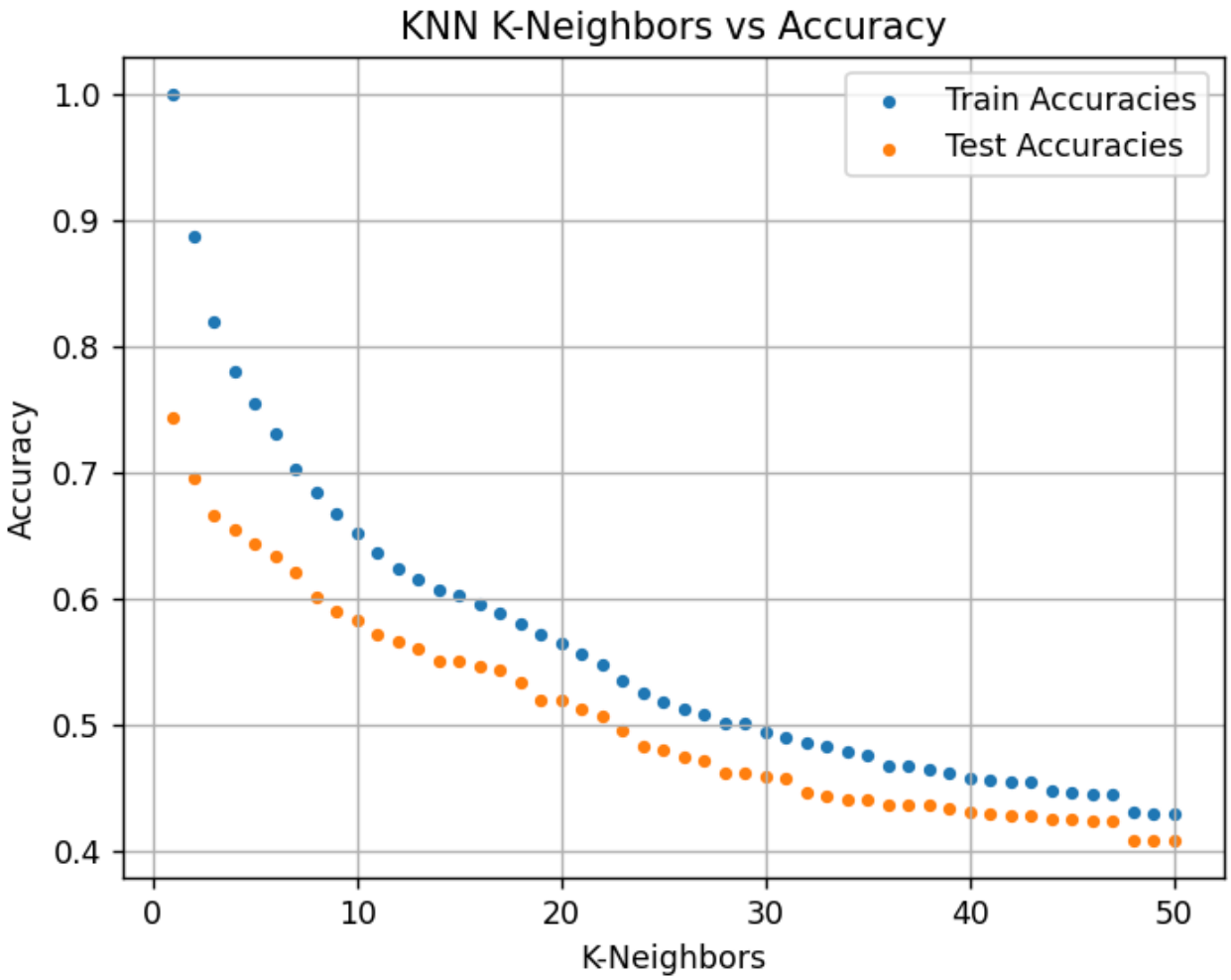
Team Name: Algo Warriors
Members: Rahul Basak, Aden Zhao, Arnav Kaul
PM: Samara Silverman
PM Meeting insights: Decision Tree Regression not working very well makes sense since most people use ensemble methods instead of one single decision tree.

Challenges that arose: It was pretty straightforward as we were just implementing models
What went well: We got a good understanding of how each model performs on our dataset
What do you want to do for the last week of your project? Use more advanced ML models

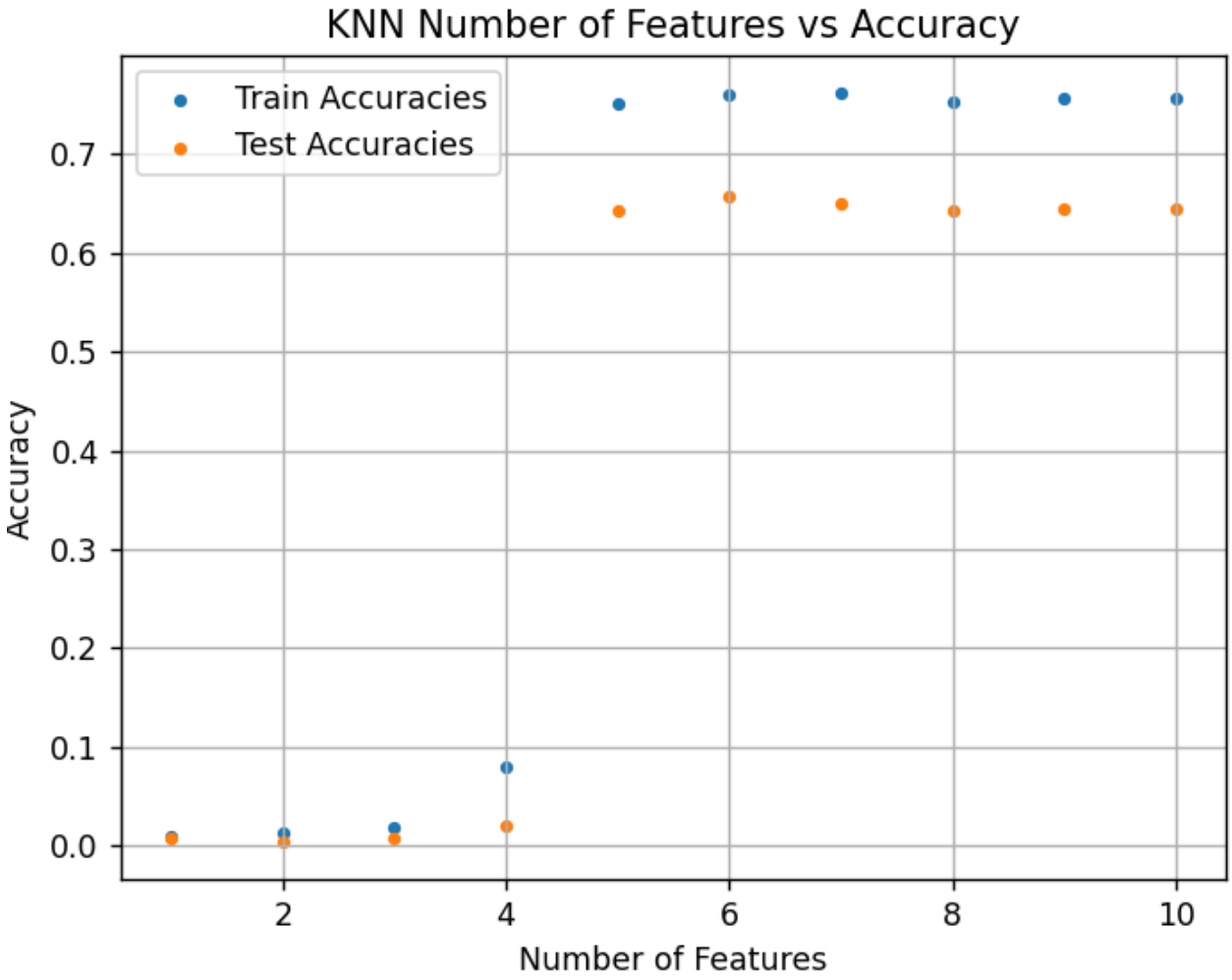
We chose to perform Supervised Regression since our core idea for the project was predicting data science salaries.

k-Nearest Neighbors Regression
KNN models simply classify points based on their nearest neighbors. When given a test point, the model determines the k closest points by some measure of distance (usually Euclidean or Manhattan). It then assigns the majority class as the prediction for the given point.

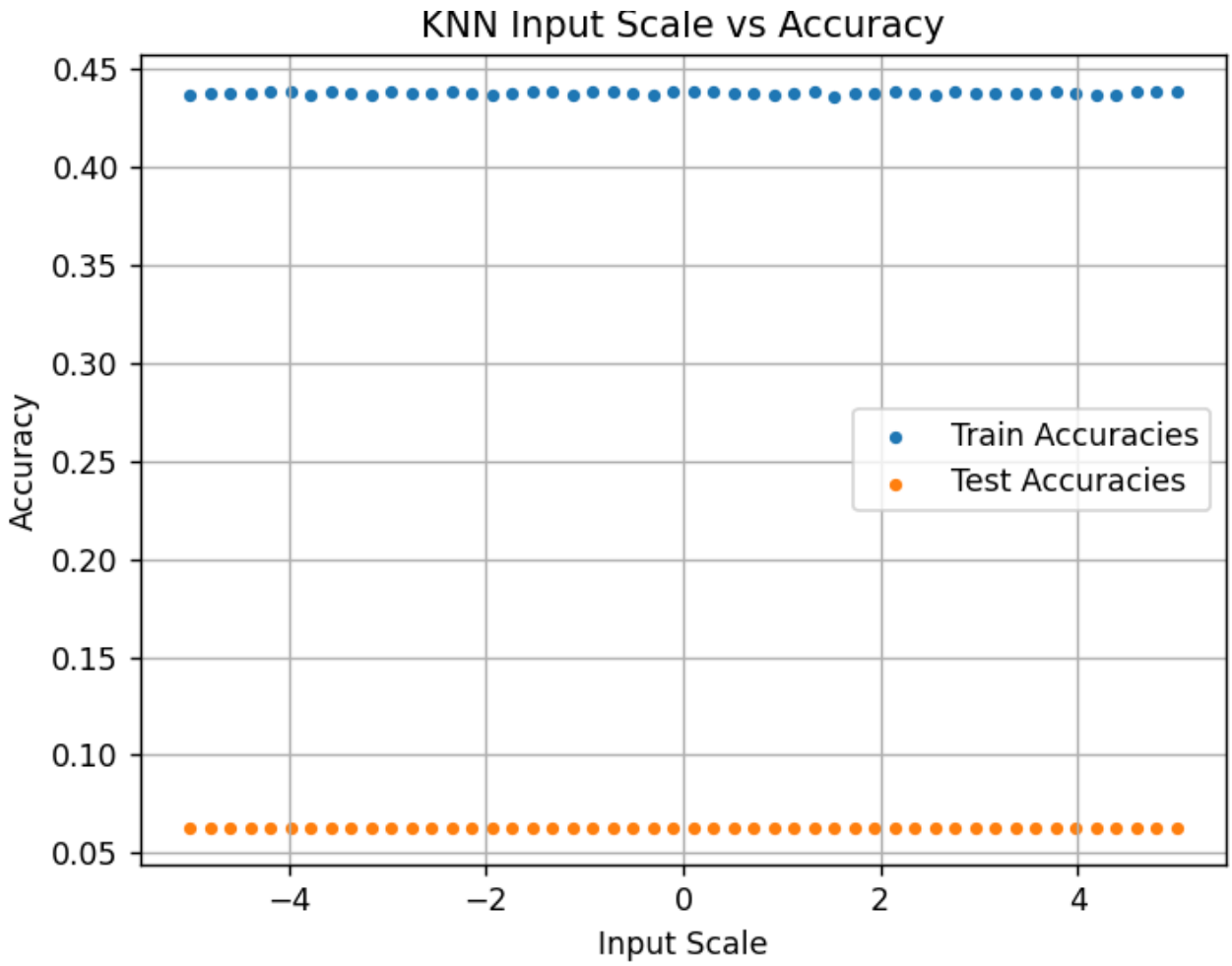
As k increases, the prediction becomes less dependent on the nearby points and simply converges to the class with the greatest number of datapoints in the test dataset.



When the number of input features decreases, the feature space "flattens" making the predictions less specific to a given points features.



Since all of our features are categorical, normalization and scaling significantly degrade the accuracy of our KNN model.



Support Vector Regression

i. How does an SVM work?

ii. How is this similar to or different from logistic regression? What do the different kernel types between linear, polynomial, and radial basis function (RBF) do?

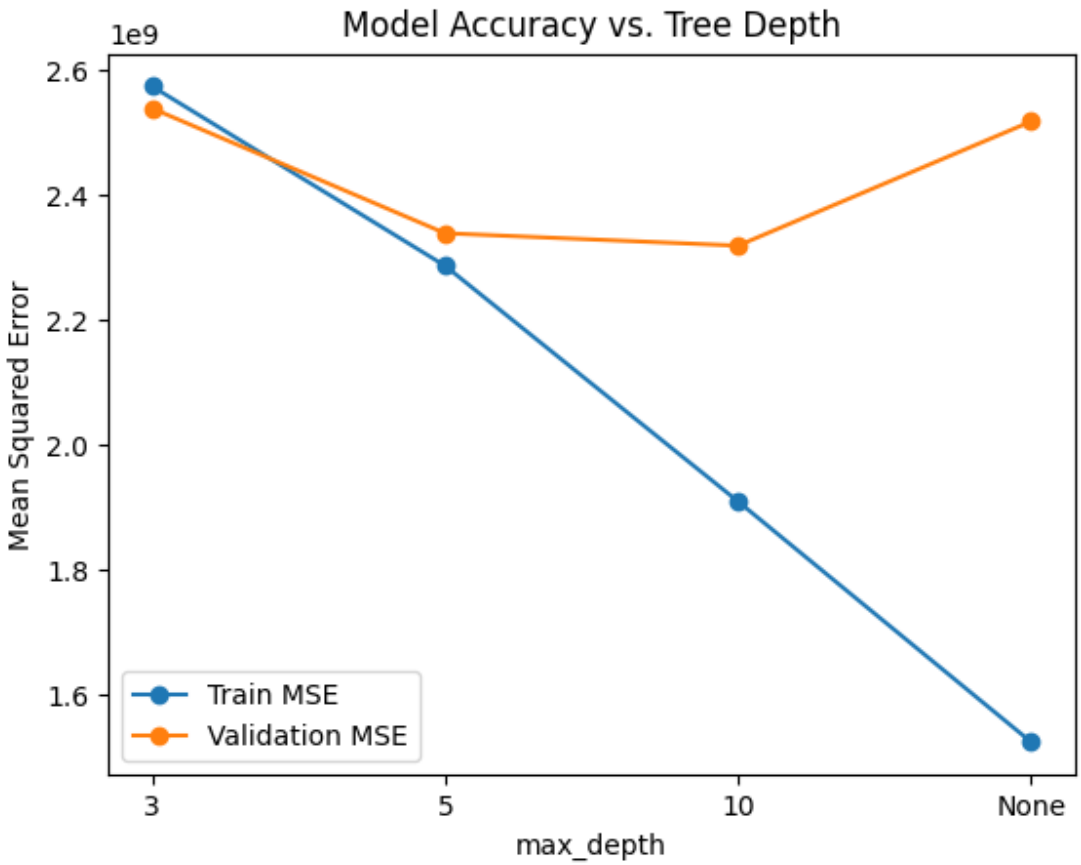
iii. For each kernel, what happens when you increase or decrease the magnitudes of hyperparameters C and gamma? Why? Justify with plots

Decision Tree Regression

i. How does a decision tree classifier work?

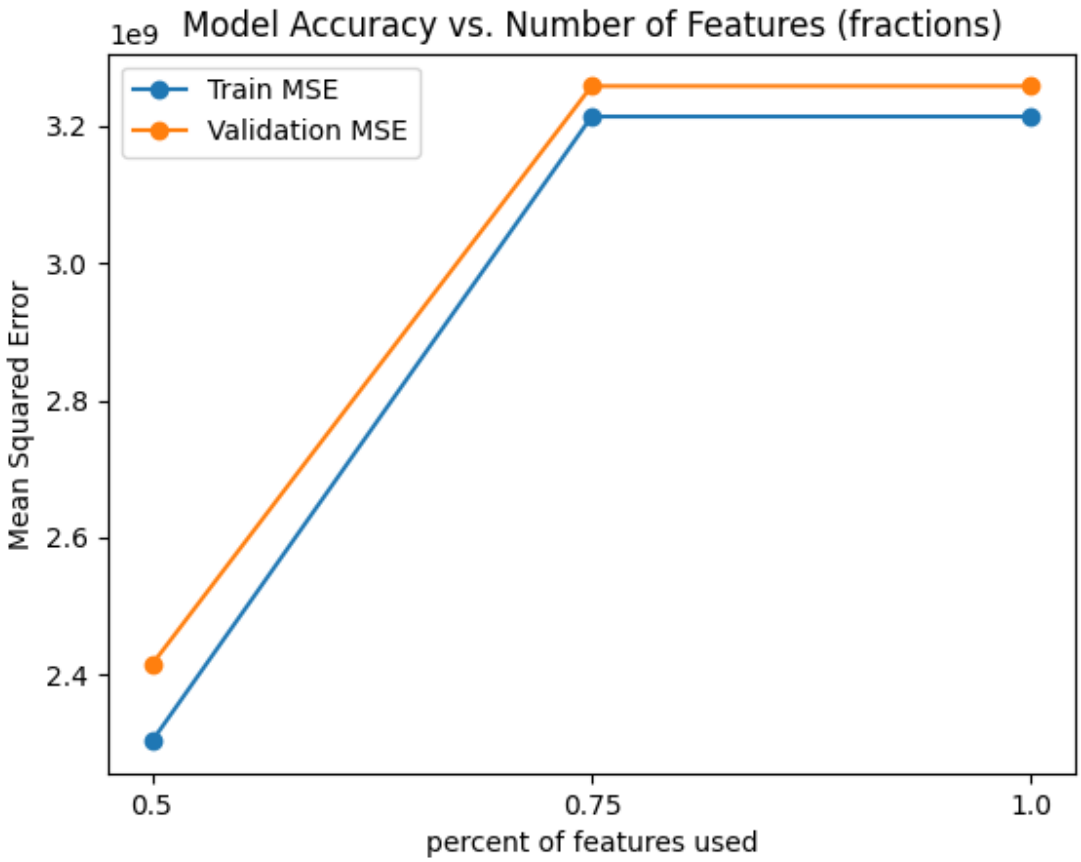
Decision trees try to predict the value of a target value by learning “rules” from the data features. It starts from a root question and continuously splits data by asking about certain features until it reaches a leaf node with a final prediction.

ii. What happens when you increase or decrease the maximum depth? Justify with a plot.



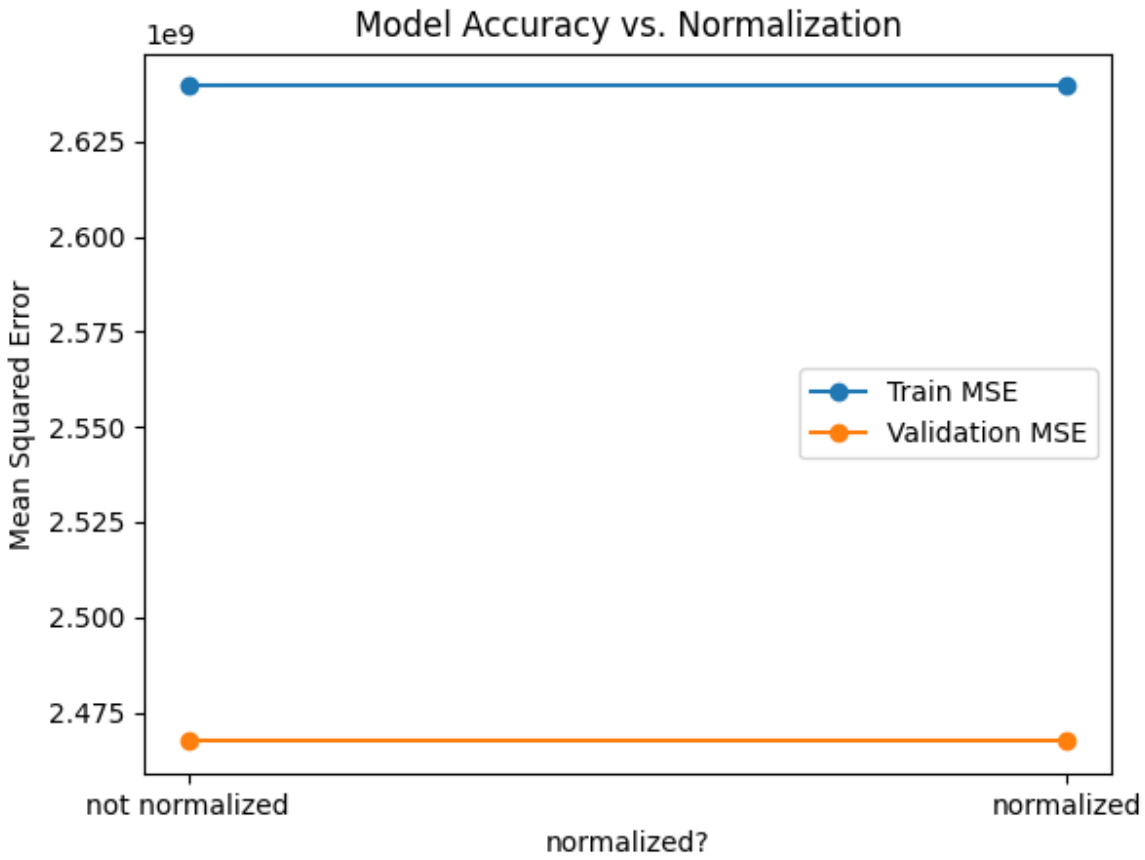
Increasing the depth means the decision tree regressor can “ask more questions” about the data to help it create better predictions, which is reflected in how increasing max depth from 3 to 10 decreases the validation MSE. However, when we set max_depth to None, our decision tree overfits on our training data and actually performs worse on the validation data.

iii. What happens when you reduce the number of features? Justify with a plot.



Strangely, the model gets worse when we use more features. I would have thought that using more features would give the model more information to create predictions from, but I guess not. I am not sure why this is the case. (maybe this is a red flag?)

iv. What happens when you normalize features? If features are already normalized, what happens when you scale them out of proportion? Justify with a plot.



Normalizing the data has no effect on the accuracy of the model. This makes intuitive sense since as long as we train our decision tree on the normalized data, the rules it learns should be the same (only with the scaling changed). Thus, the model will make the same decisions.

Explore and answer the questions.
i. Which k-NN, SVM, or DT performed the best? Justify with a plot.
Why do you think that model had low validation loss?
As seen above, the model that performed the best was KNN. We believe it had the lowest validation loss due to the heavily categorical nature of our feature set. Due to this, the nearby points shared similar feature classes, leading to similar prediction classes.
ii. Compare the tradeoffs between using a k-NN, SVM, or DT
classifier. Hypothesize in what settings each would outperform the
other two. Consider
1. Time it takes to train
2. Time it takes to test
3. Train loss
4. Test loss
Since our KNN model performed significantly well compared to the others, we expect to use this in almost all scenarios. Since KNN has an extremely minimal training time, it is especially useful

for scenarios that require a short initiation time. However, KNNs have a very intensive test classification time so it may not be as useful for quick prediction requirements.

Signatures: Arnav Kaul, Aden Zhao, Rahul Basak