In our project, we are trying to predict the total number of bike rentals for a particular day and time given the date and weather conditions.

For our project, we used a dataset provided by Hadi Fanaee Tork from the company Capital Bikeshare.

Now, as for the data description.

The data is collected from Washington, D.C.

In our dataset, there are 9 attributes and 3 labels for each data point. The attribute contains the date and weather information; The labels are the number of casual users, the number of registered users, and the total number of bike rentals; The total number of bike rentals is just the sum of the number of casual and registered users. For the purpose of our project, we are only interested in predicting the total count.

The data contains both discrete and continuous attributes. The discrete attributes contains season, holiday, workday, and weather. The continuous data contains the date, the measured temperature, the temperature that it feels like, humidity, and wind speed.

For our model evaluation we decided to use R M S E and R square value. R squared value, or ccoefficient of determination, is a statistical measure of how well the regression line approximates the real data points. It the R squared value is 1, it means the model fits the data perfectly. A low R squared value means the model is performing poorly.

Since we are predicting continuous data, a regression model would be the most suitable choice. For all of our models, we did a 20, 80 split between the testing and training data. For any models that required tuning of hyper-parameters, we did a 20, 16, 64 split into testing, validation, and training set.

The first naive attempt was to use linear regression. The R\_squared value for linear regression was 0.32, and had an R\_M\_S\_E of 146.6 on the testing data. This was clearly not a good model, but it serves as a base line model for future comparisons.

The next model we attempted was kernel ridge regression. Using kernel ridge regression with polynomial kernel, the R squared value on the testing data was 0.54. In attempts to improve the performance, we started tuning the hyper-parameters associated with this model. After testing the gamma and alpha values, it appears that they don’t contribute to decrease the error. The degree of the polynomial on the other hand, has a significant influenced the model performance. The default degree of the polynomial was 3. To tune the hyper-parameter of our model, we did a 64/16/20 split of the data into training, validation and testing sets. We plotted the R squared score against the degree of the polynomial shown in the figure. The kernel ridge regression model provided by s\_k learn ran into runtime errors and yields inaccurate results when the degree of the polynomial exceeds 3. Therefore, no further investigation was done for polynomial kernels. Other kernel functions were tested, and the R squared values were 0.21, and 0 for r\_b\_f, and sigmoid respectively.

The next model we attempted was MLP regression. With max iteration set to 100000, and all other parameters set to default, we tested the relative performance of each activation function. It was observed the R squared value of relu was 0.45, logistic was 0.54, identity was 0.32, and tan\_h was 0.61. Since tan\_h performed significantly better than all the other activation functions, we decided to use tan\_h as the activation function for this model. There were two hyper-parameters we had to tune, the number of hidden layer, and the size of the layer. We generated 9 plots where the number of hidden layers ranged from 1 to 9. For each plot, we plotted the R-squared value against the number of neurons at each layer. It was observed that the overall performance got better as the number hidden layers increased. After the number of hidden layers got to 6, there was no significant increase in the performance. Comparing each graph, we see that the performance against the number of neurons was unpredictable. For our final model, we selected the model with a hidden layer of size 8 with 100 neurons in each layer. The reason we chose the model with 8 layers was because its performance appears to be less susceptible to changes in the number of neurons. For this model, the R-squared value and RMSE were 0.61 and 115.5 respectively.

Afterwards, we attempted regression trees. In attempts to find the optimal height of the regression tree, we again plotted the R squared value against the maximum height of the regression tree. In the figure, we see that the performance of the regression tree model has the best performance when the maximum height is set to 10. Therefore, our final regression tree model had a maximum height of 10. The R squared value of the regression tree on the testing set was 0.79, with an R\_M\_S\_E of 85.2.

To improve the performance of regression trees, we attempted using adaBoost on regression trees. The R squared value was plotted against the maximum height of the regression tree in Figure 2. From the figure, we see that the R squared value showed no significant improvement when the maximum height is greater than 8. Therefore, we decided to use 8 as the maximum height of each regression tree. The model had a similar performance on the testing data compared to the validation set. The R squared value was recorded to be 0.83 on the testing set, with an R\_M\_S\_E of 77.1.

Then to compare the performance of AdaBoost versus just plain regression tree, we plotted the R square value against the maximum height of tree. As we can see from the graph, AdaBoost still has a good performance when the maximum height of each tree gets large, but for a plain regression tree, the performance gets worse when the height is greater than 10. The decrease in performance of a plain regression tree may be due to the effect of over fitting. The interesting observations is that AdaBoost is less prone to over fit.

     Out of all the models, it seems that regression trees with adaBoost has the best result. This was as expected. Since out data contains both continuous and discrete attributes, a regression tree model would be best suited because it is good at dealing with both discrete and continuous data simultaneously. When there are discrete attributes in the data, Kernel ridge regression with a polynomial kernel on the other hand, is trying to fit a continuous function over an n-dimensional space, so it would have trouble fitting discrete attributes.