**How much is your car worth?**

****

**By: Ruben Valdovinos Jr, Natalia Walls, and Samuel Flusche**

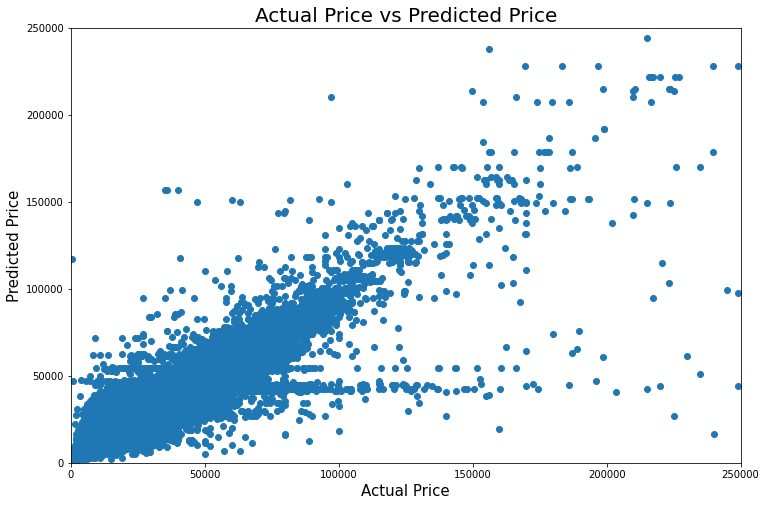
Introduction:

The dataset that we used came from Kaggle.com and is sourced from cargurus.com. The dataset contained the price that the car sold for and then several descriptive factors about the car. The first issue we ran into with this dataset was its size. The dataset had several million rows and 66 different columns, making the dataset 10 GB large, which was far too big to effectively work with so it was necessary to pair the dataset down a little bit. To do this we took a random sample of 400,000 rows and get rid of useless or redundant columns, which allowed us to narrow the dataset down to 32 columns. The next issue we ran into was with a very large amount of Nan values in almost all of the descriptive columns. The descriptive columns kind of split into 3 different categories. There are numerical columns, which have numerical stats about the car. Things such as mileage, length, height, number of owners, etc. The next type was categorical columns, which have non numerical stats about the car. This would constitute factors like the brand, model, engine type, etc. The final category was boolean columns which just had true or false values for the car of things like, whether or not the car was new and whether or not the car was damaged. Many of the boolean columns had almost half of their values missing so the most effective way to fill the Nan values for these was to just label them as ‘unknown’ and make it a third option for those columns. For categorical variables it was appropriate to do the same. For numerical columns it was a little bit more complicated, but ultimately the decision we came to was to just use the average value of each respective column to fill the Nans with. From there we had a dataset we could actually work with to analyze.

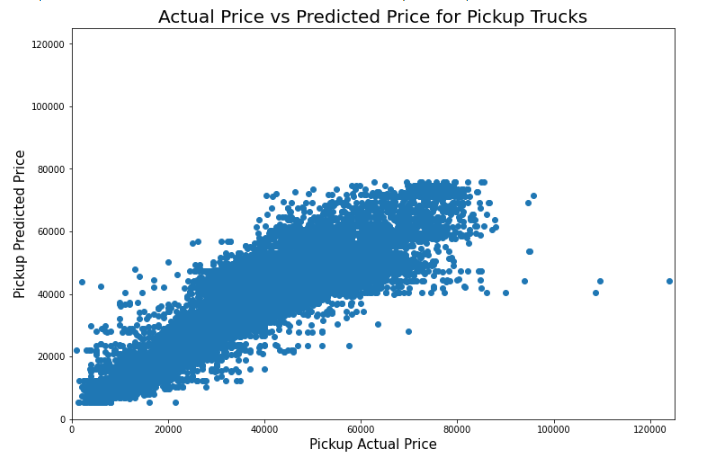
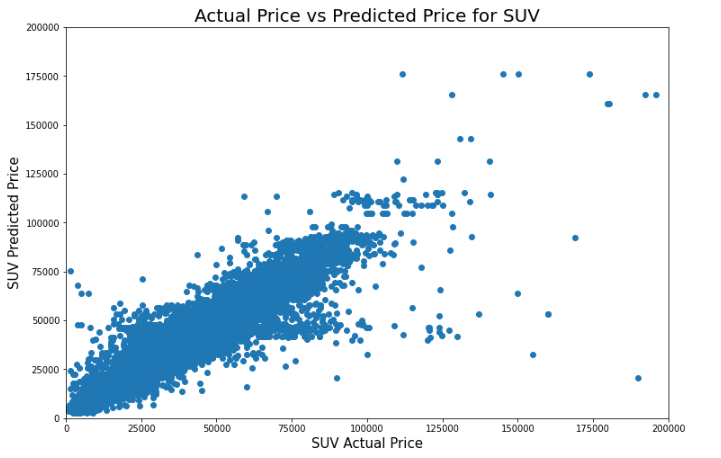
Our goal for this analysis is to find what model is the most accurate when predicting the price of cars. Since price is a continuous value, we need to use a regression model. The way we went about this was to first just do a basic linear regression model, but we soon found out that it was not accurate in predicting prices, so we quickly tried a decision tree regression and a K Nearest Neighbor regression model as well. The last thing we attempted was ensembling the two regressors. The columns used for all our models are mileage, owner count, front legroom, back legroom, length, width, height, new, horsepower, damaged, and brand.

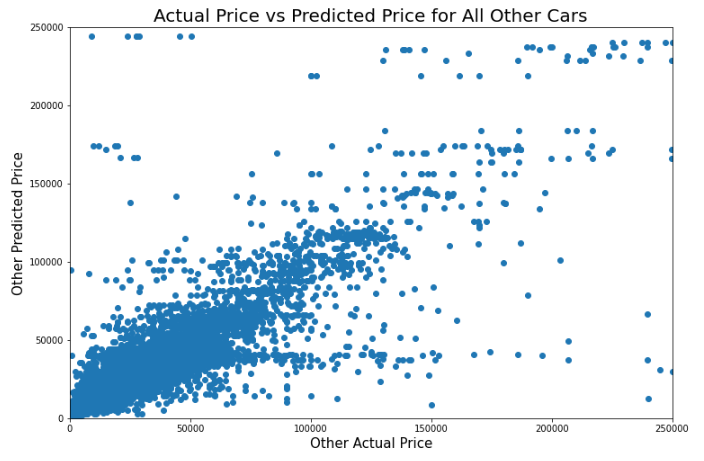
## Decision Tree Regressions:

When first making our decision tree regressions we attempted to use every car in our data set all at once. So we took all of the different types of cars and used a model to predict the price. We made a model that had an R^2 of 79.05% when predicting the prices for every type of car in our data set.



Next, we decided to split up our model into three different data sets, one only for SUVs, another for pickup trucks, and the last one has all other cars. After we split up the data set we did the same regression, on those three individual data sets. Doing this for SUVs we got an R^2 of 87.14%. For pickup trucks, we got an R^2 of 75.42%, and for the rest of the cars, the R^2 was at 59.52%. Below are the scatter plots of the predicted values vs the true values for all three data sets.

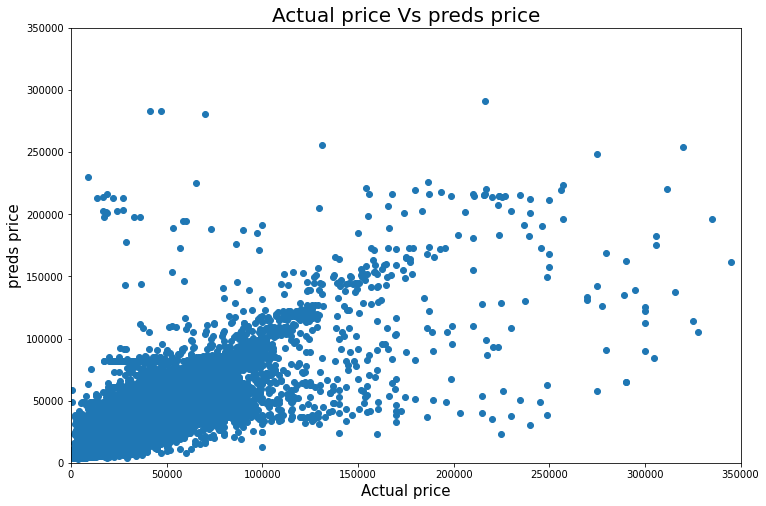




## K Nearest Neighbors Regressions:

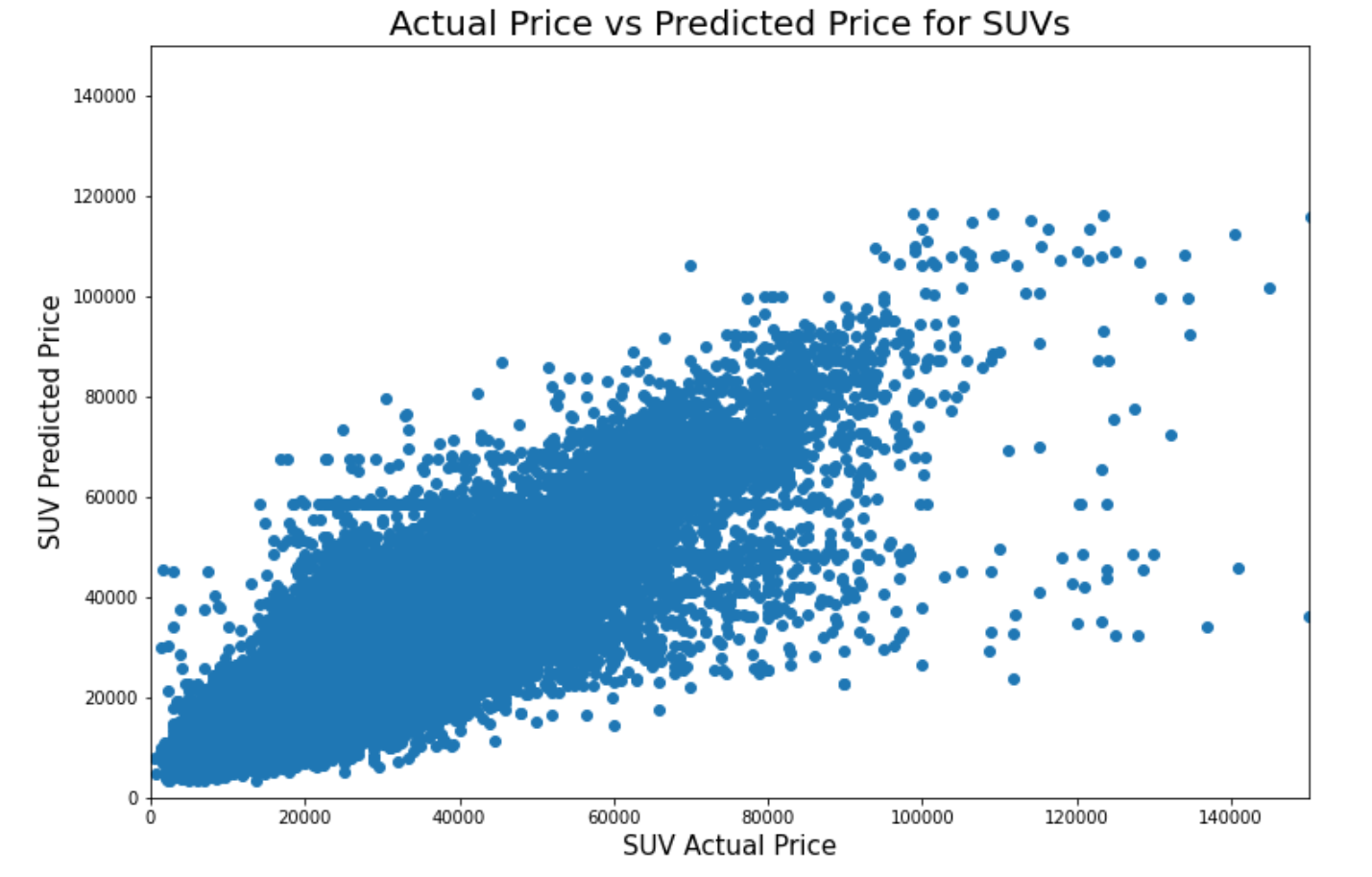
Next, we wanted to see if we could get a more accurate model with K nearest neighbor regression models, so we did the same thing as we previously did. Where we did a KNN regression model on every car in the respective columns, then split it up into SUVs, pickup trucks, and the remainder of the cars.

When we did the regression for all cars the KNN regression had a R^2 of 68.76% which is decent but still needs improvement. The best parameters for the model were the metric distance: euclidean and nearest neighbors: 8.

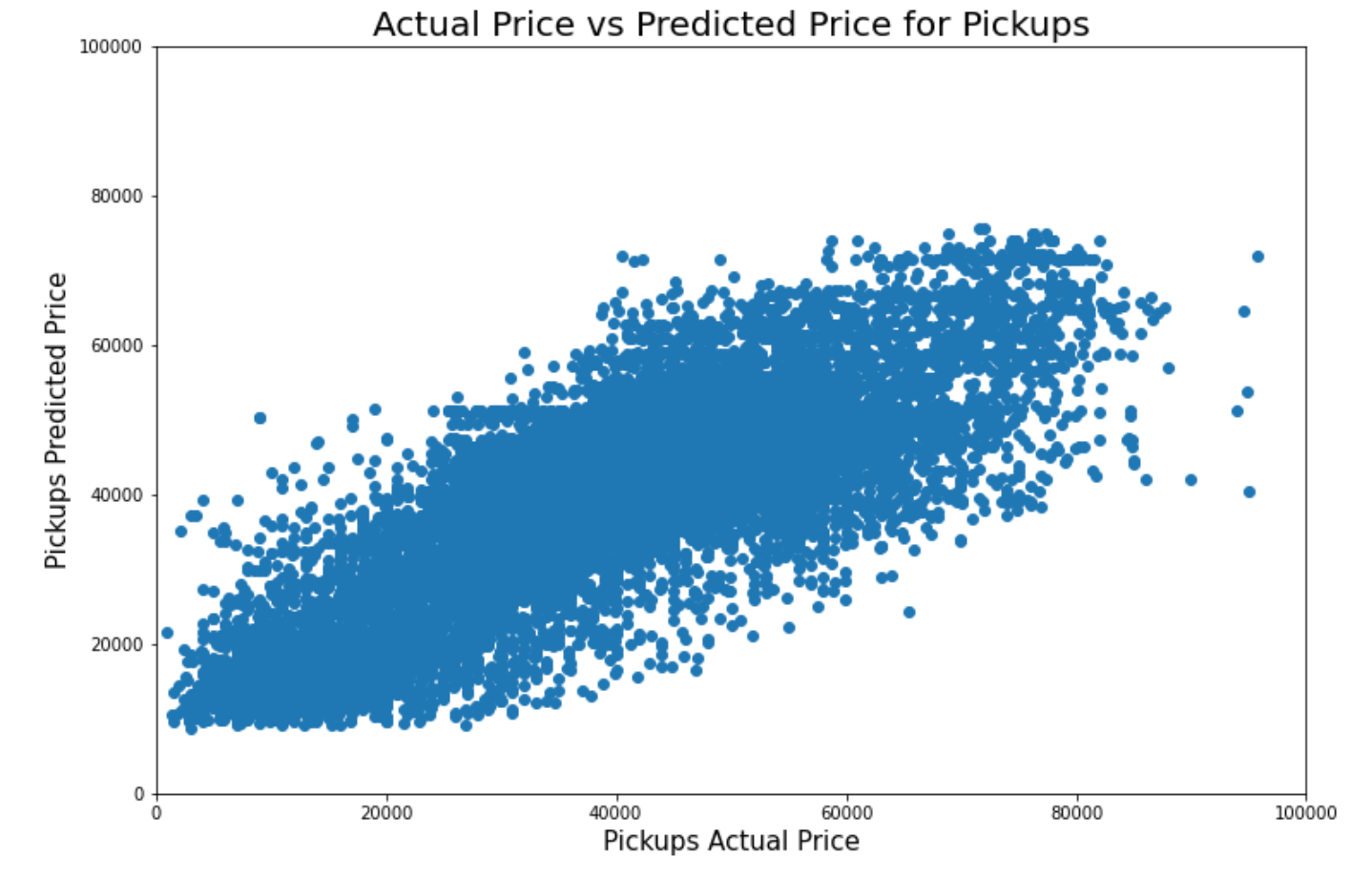


After we fit the model for the entire data set we once again split up the cars into the three categories and got different results.

For the model where we only fit SUV’s we got an R^2 of 72.42% which is better than what we were getting as a whole. The best parameters for this model ended up being nearest neighbors: 5 and the best metric: euclidean.

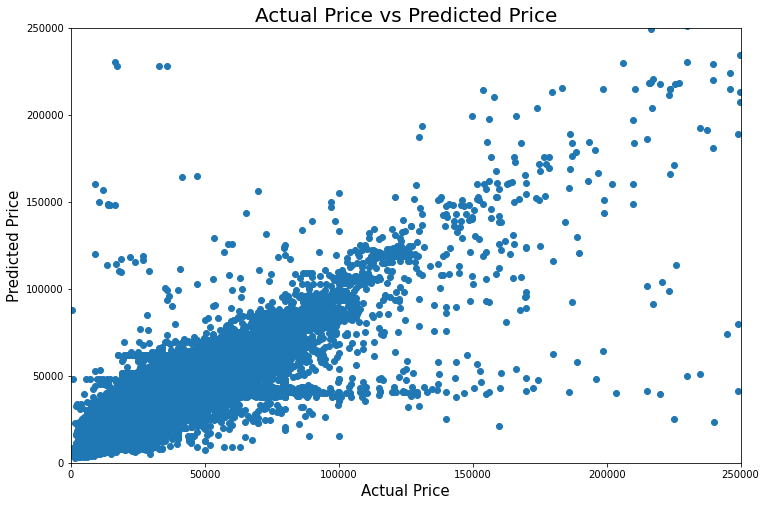


After looking at SUV’s we once again looked at the Pickup tucks thinking that we would see a better R^2 once we fit the model just for pickups. However, our model came out worse than we expected. The R^2 came out to be 62.78% and the best parameters were nearest neighbors: 23 and the best metric: euclidean.



Lastly, we followed the same procedure for the rest of the cars and the model gave us an R^2 of 41.45% with the best parameters being nearest neighbors: 12 and the best metric: euclidean.

Once we had all of those models we decided to ensemble the decision tree model and the nearest neighbors model to see if we could make a model that was even better. We used the voting regressor and fed the models we had made into it and the model gave us back an R^2 of 77.34%. The graph below shows the voting model predictions.



## Conclusion:

After looking at all the models there were some that were definitely better than others. The nearest neighbors model was not good at predicting prices because of the fact that cars had similar dimensions but were different brands making columns like front legroom, back legroom, length, width, and height useless. For example, you could have a sports car and a regular car be the same dimensions but since the vast majority of the cars that have those dimensions are cheaper the model will also predict that the sports car to have the value of a regular car. Yet the decision tree models did very well overall. The models we made were very good with SUV’s and the overall data. Our best models were the decision tree models that had R^2 values of 79.05%(all cars) and 87.14%(SUV).