A Tip-Toe Through the Forest

Price Prediction for Used Cars Using Ensemble Methods

Statistical Learning - DAT 402

R. Vincent Caldwell
School of Mathematical and Statistical Sciences
Arizona State University
Tempe, AZ
rvcaldw1@asu.edu

***Abstract*—The purpose of this project is to predict used car pricing from a supplied data set. This will be done by applying machine learning techniques that are well suited to this specific problem. We will use, for this project, data contained in the used car data set supplied by Dr. Robert McCulloch, which includes not just pricing, but data about the mechanical attributes for each vehicle. This is a final project for DAT 402: Statistical Learning taught by Dr. Robert McCulloch at Arizona State University.**

***Index Terms*—Regression, Tree, Random Forest, Boosting, XGBoosting, Machine Learning.**

I. Introduction

The used car market is a very important part of the global economy and it has a very large and growing footprint. The industry is so large, in fact, that its size and scope may be difficult to envision. For example, it is forecasted that the used car market will be worth a whopping $885 billion by 2026 [1], and, in 2018, more than 40.2 million used cars were sold in the US [2].

The market for used cars is huge, but not yet easily predictable. This has a lot to do with the annual differences in variance when it comes to supply and demand. So, understandably, we should want to know what the price distribution would look like if we were able to predict the price for a particular vehicle using the available data, and, of course, machine learning methods. This project attempts to predict used car pricing from different mechanical attributes.

The related work in this domain is broad [3]. It includes the estimation of not just pricing, but a multitude of related and unrelated factors that contribute to a consumer's decision whether or not they desire a particular vehicle. However, the scope of this project will be fairly narrow and includes only the prediction of price for used cars by analyzing the past trends. Consequently, the results may be used not just by corporate stakeholders but by private citizens as well. We, therefore, want to predict as accurately as possible to extract the most useful information from the small amount of data we have available.

Having already defined our business understanding. We will also pursue the data science life cycle in the following order [4]:

Data Collection - in which we gather and scrape the data necessary for the project.
Preprocessing - correct the inconsistencies within the data and handle any missing or erroneous values.
Initial Exploration - here we will investigate the data set and report on any interesting findings as well as an initial visualization of the data.
Feature Engineering - for this step, we will select important features and construct more meaningful ones using the raw data that we have. This also includes processing categorical variables using one-hot encoding.
Predictive Modeling - finally, we train machine learning models, evaluate their performance, and use them to make predictions.
Data Visualization - then, communicate the findings with key stakeholders using plots and interactive visualizations.

II. Data Set

Our data set contains a variety of structured data.

Data Collection The data was obtained directly from Dr. Robert McCulloch's website: http://www.rob-mcculloch.org/data/index.html. Attempting to locate other data via scraping online webpages, scouring databases, or any other means of available data mining was unnecessary. The following, *Table I*, is a description of the variables included within the data set.

Preprocessing Preprocessing normally takes the most amount of time. However, the available data set was already semi-processed. Luckily there was no missing or 'NA'

TABLE I
VARIABLE EXPLANATION

Variable name:	Definition:
Price	this variable contains the price of a used car, and its collection of values ranges from 599.00 to 79,999.00.
Trim	this variable contains the trim level that makes each model identifiable by particular features. These may involve both interior and exterior options. The unique values within the trim packages include '320', '420', '430', '55 AMG', '500', '550', '63 AMG', '350', '400', and '65 AMG'.
isOneOwner	this variable contains a binomial response of either a 1 or a 0 - corresponding to whether or not the vehicle has been previously owned by only one owner.
Mileage	this variable contains the mileage of a used car, and its collection of values ranges from 8 to 488525.
Year	this variable contains the year of the car, and its collection of values ranges from 1994 to 2014.
Color	this variable contains the exterior color of the car. The values included are 'Black', 'White', 'Silver', 'Gray', 'Blue', 'other', and 'unsp' (unspecified).
Displacement	this variable commonly measures engine size, and can be an indicator for the general power an engine may produce. The range of values varies from 3.2 to 6.3.
Fuel	this variable contains the vehicle fuel type. Values include 'Gasoline', 'Diesel', and 'Hybrid'.
Region	this variable contains whether or not the car was made in a certain region. The values include 'SoA' (Super output Areas), 'Mid' (Midwest), 'ESC' (East South Central, 'ENC' (East North Central), 'WSC' (West South Central), 'New' (New England), 'Mtn' (Mountain), 'Pac' (Pacific), and 'WNC' (West North Central).
soundSystem	this variable contains the sound system available in the given vehicle. The values contained are 'Premium', 'Bose', 'Harmin Kardon', 'Bang Olufsen', and 'unsp' (unspecified).
wheelType	this variable contains the type of wheels on the vehicle. The values included are 'Alloy', 'Premium', 'other', and 'unsp' (unspecified).

values. This alone exponentially sped up the preprocessing. Had there been missing or other values within the data set that needed to be addressed, we would need to invoke several methods in order to reach a satisfactory resolution. For instance, had the data represented what one would normally expect in the wild, we would preform the following steps before continuing with the initial exploration of the data:

- Installing and loading all necessary packages and libraries.
- Remove all of the white space and non-numeric elements. This includes dollar signs ($) and commas (,) from variables like currency, for instance.
- Check for missing values, and, if there are any, either remove the entry or replace the missing values with correct information or median values for all similar entries.
- Remove all 'NA' values with median values. Using the median will give you a middle value that is not effected by outliers unlike the mean value. In this regard, the median is preferred to the mean only because mean values are more sensitive to potential outliers.
- Then, we can rename columns for clearer reference.
- Finally, add any new columns that are not already in the dataframe.

We must bear in mind that, commonly, it is only a personal judgment call when we replace or remove NA, empty, or other values and rows from a data frame. In general, there are no hard and fast rules for replacement and removal.

Outlier Detection It is imperative that we perform outlier detection on the data set. Without performing outlier detection there is a chance that the results we get from our predictive modeling will be inaccurate.

It is important that the data we are using for our predictive modeling is accurate otherwise it could lead to false conclusions and affect the decisions we make. One of the most important steps in any analytical model is outlier detection or outliers in general. Outliers can have a large impact on univariate or multivariate analysis and can change, distort, enhance, or decrease predictive power when present in a dataset.

For our purposes, we will use an ensemble method called Isolation Forest to perform our outlier detection. Isolation Forest is a classification and regression algorithm, *see Fig. 1* [5]. It belongs to a family of machine learning algorithms called ensemble methods. Isolation forest is an improvement on the original ID3 algorithm for induction of decision trees.

The aim of isolation forest is to learn a set of decision trees so that each tree is accurate in its own right and independent from the other trees in the ensemble. This makes isolation forest a kind of 'noisy decision forests'. Discarding the rules learned by individual trees, Isolation Forest can generate ensemble rules that are efficient, accurate, and interpretable. The algorithm classifies each data point as either an outlier or not. If the result is -1, then it is an outlier. Otherwise, the result will return 1. We can simply sort our dataframe and remove all entries that are marked with the outlier designation.

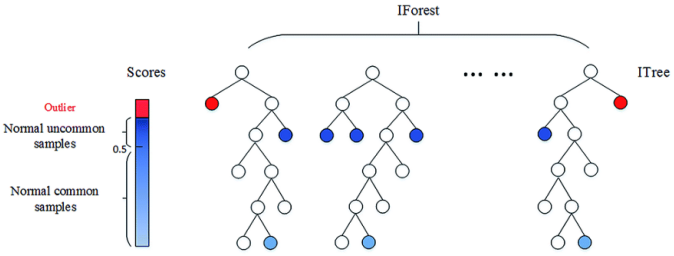

Fig. 1. Presentation of an Isolation Forest

[6]

III. Initial Exploration

Our initial exploration begins by producing a pairplot to visualize potential relationships within the data. Pairplots are a convenient way to visualize and explore how two variables interact with each other. In general, pairplots can be used to explore many different kinds of relationships, such as the relationship between price and mileage or mileage and year, *see Fig. 2.*

An interesting visualization we can use if we did not already have a specific purpose is a word cloud. This can be helpful to get a sense of how many occurrences of a word appear in the data and what the most common words are.

For example, we can use the size of the word to potentially infer its importance - since the size of the word is specifically tied to how many times it occurs within the data set. This is what we see here with the words "Gasoline" and "Alloy". It makes sense these two words are large, because they are also the most commonly found attributes of cars on the road today. Since word clouds are not interactive, it can be difficult to read that text. However, our data set is reasonable enough that a word cloud can be produced, *see Fig. 3.*

Finally, we can use a correlation matrix to show the correlation between all of the variables. The matrix does not show relationships between two variables directly, but rather how they relate to each other. We see that the columns and rows are ordered by value from highest to lowest. The closer two values in the matrix are to 1, the more strongly they are correlated, *see Fig. 4.*

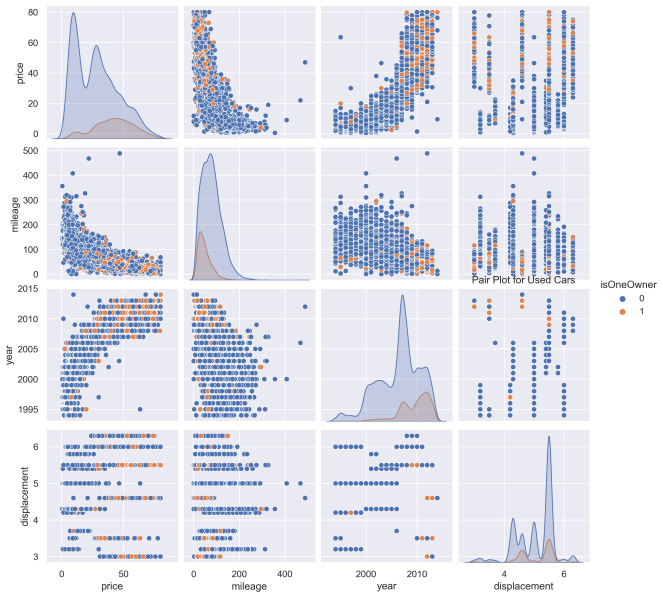

Fig. 2. Most Common Words in the Used Car Data Set

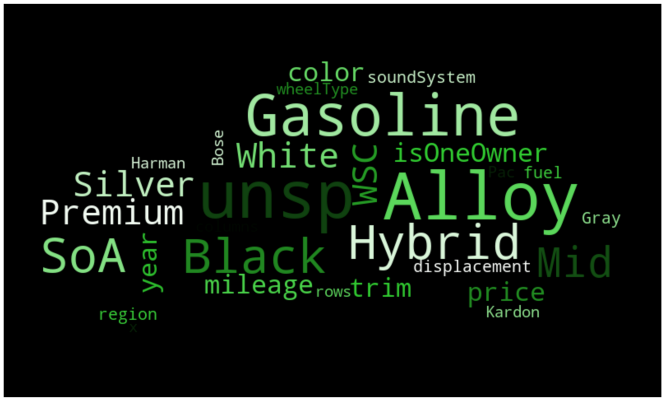

Fig. 3. Most Common Words in the Used Car Data Set

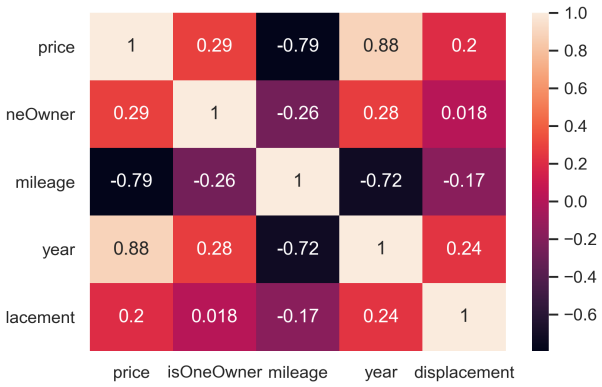

Fig. 4. Most Common Words in the Used Car Data Set

IV. Feature Engineering

As we move on to feature engineering, we need to make sure that our features are both useful and necessary. We must dummy-up our categorical variables and determine whether we will be using every column, or if we can apply dimensionality reduction. This is where one of the main concepts for feature engineering comes into play: feature selection. In a nutshell, feature selection attempts to reduce the number of features so that we have more control over our model.

Our goal is to find a subset of columns in our dataset which has maximum predictive value for our target variable. This can be done by using various statistical measures, developed over time for this very purpose. Selection can be done on different subsets independently based on their value and pull features derived from them. For instance, we may use LASSO regression as a means of dimensionality reduction and an initial baseline to compare accuracy to future ensemble methods, *see Fig. 5* [5].

After computing the LASSO regression, we are left with 5 of the original 55 possible features, *see Fig.6*. These are:

$$\begin{aligned} \textit{'mileage'}: &\ -0.10990378584660783 \\ \textit{'year'}: &\ 2.344278526269592 \\ \textit{'trim\_430'}: &\ -1.7309638957334854 \\ \textit{'trim\_63\ AMG'}: &\ 5.816693107028769 \\ \textit{'displacement\_4.6'}: &\ 5.935140458040483 \end{aligned}$$

We can test the overall accuracy of the LASSO regression and error by using the Mean Squared Error (MSE) and r^2 score, *see Fig. 7*. This is because LASSO identifies the model with the most optimal log-likelihood. The goal of LASSO

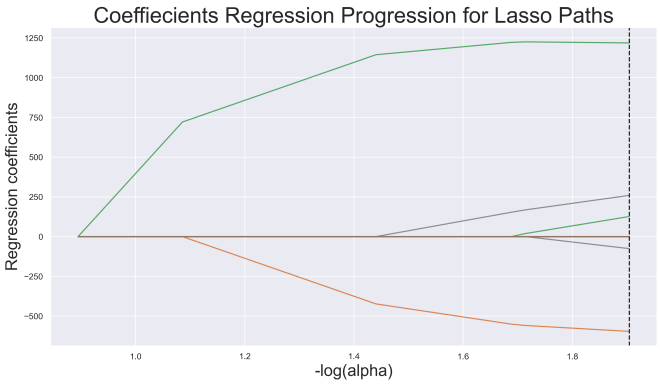

Fig. 5. Lasso Dimensionality Reduction

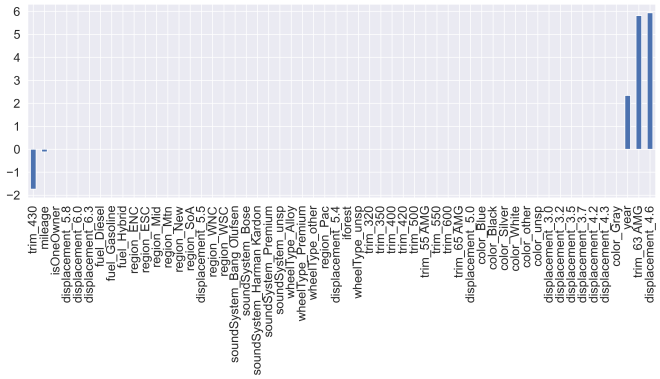

Fig. 6. Visualizing Coefficient Importance

regression is to select the subset of variables that have the greatest possible predictive power. Below are the calculated MSE and r^2 scores for both the train and test data - this is for comparison only, in-sample testing is an inappropriate measure of success.

We can examine the accuracy of the predicted variables and actual values in our predicted quantities from our LASSO regression model. The mean squared error (MSE) is given as follows:

$$\begin{aligned} \text{MSE } \textit{[Training data]}: &\ 49.36165366103956 \\ \text{MSE } \textit{[Test data]}: &\ 48.786511998326034 \\ r^2 \textit{ [Training data]}: &\ 0.8515505827794099 \\ r^2 \textit{ [Test data]}: &\ 0.8546526934199903 \end{aligned}$$

LASSO is more flexible than ordinary least squares because it does not require the constraint of equal slopes for all variables. When the goal is prediction accuracy, then LASSO has a more straightforward interpretation. LASSO regression attempts to estimate linear models, which are easier to interpret.

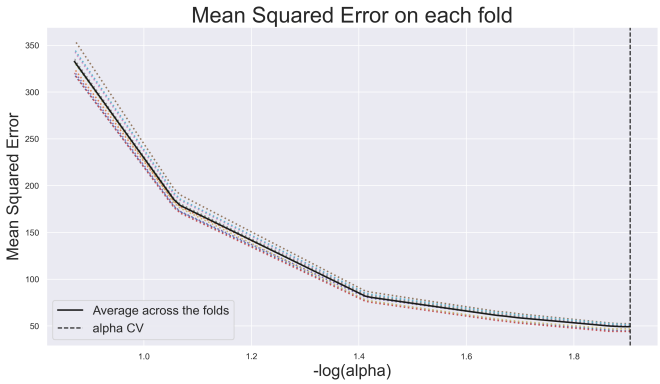

Fig. 7. MSE Per Fold

V. Predictive Modeling

Decision Tree Initially, we chose to model a single decision tree with no hyperparameter tuning to set a baseline for all future iterations of alternative ensemble methods. This decision tree was able to achieve accuracy above 88%. This particular algorithm was chosen because it requires less effort for preparation. Not only does a decision tree not require normalization, but it also does not require scaling nor is it sensitive to missing data. However, with every advantage there is a disadvantage.

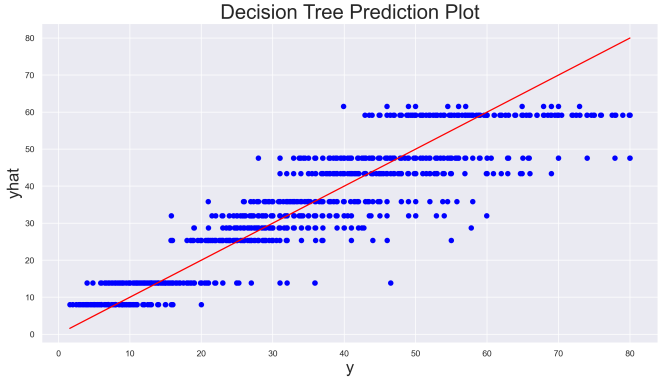

Fig. 8. Visualizing Decision Tree Prediction

Decision trees are quite sensitive to any change in the data causing instability. Additionally, decision trees can require more time to train a model, being both computationally expensive and greedy [5].

However, below you will find the calculated measures of accuracy and error.

Measures of accuracy and error

Root Mean Squared Error: 6.1678
Mean Squared Error: 38.0415
Mean Approximate Error: 4.3639
r^2: 0.8882
Model Accuracy*[Single Tree]:* 88.82%

We were able to plot a single decision tree diagram using the graphviz package from python, *see Fig. 9.*

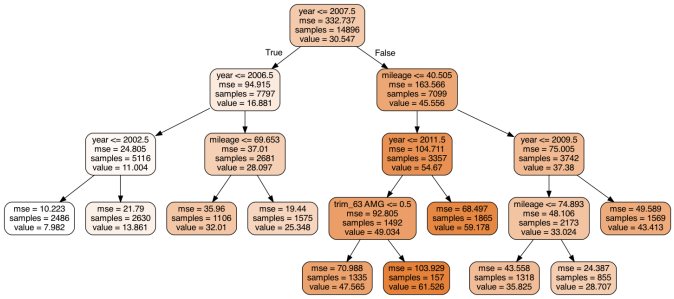

Fig. 9. Visualizing Sample Decision Tree

Random Forest The Random Forest algorithm is a very powerful and popular method for prediction using categorical and numeric input variables [5]. As a mainstay of the ensemble methods, it is particularly well suited to make fairly accurate predictions out of the box. However, we used GridSearch for hyperparameter optimization and cross validation to achieve a score of 91.53%. The best parameters included: n_estimators=41, min_samples_split=2, min_samples_leaf=2, max_features='sqrt', max_depth=4, bootstrap=True.

A few of the advantages of Random Forest include error handling on imbalanced data sets, and insensitivity to outliers. Random Forest also preforms error handling quite well. Due to this particular feature, individual error is minimized and overall variance is reduced. Although, while features need to be relevant, the algorithm can be difficult to understand - earning the black box label.

Black box aside, we were able to successfully improve our accuracy score from the Decision Tree, below are the calculated measures of accuracy and error.

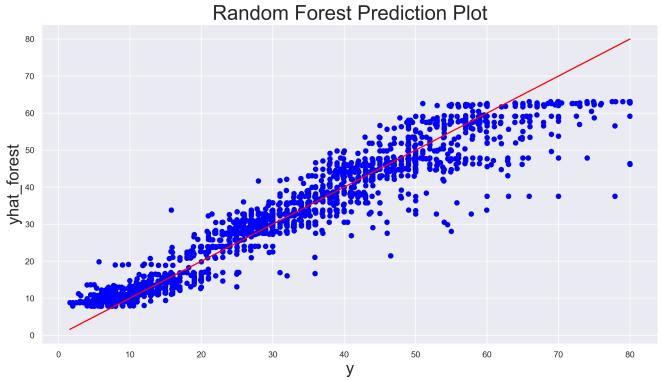

Fig. 10. Visualizing Random Forest Prediction

Measures of accuracy and error

Root Mean Squared Error: 5.3691
Mean Squared Error: 28.8267
Mean Approximate Error: 3.6636
r^2: 0.9153
Model Accuracy*[Random Forest]:* 91.53%

Like the decision tree, we were able to plot a single Random Forest tree diagram, *see Fig. 11.*

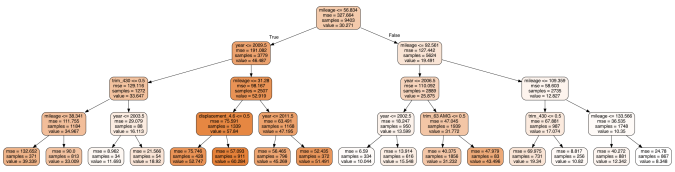

Fig. 11. Visualizing Sample Random Forest Tree

Boosting Boosting is an ensemble algorithm that sequentially builds models and combines them to form a one strong learner [5]. Boosting methods are shown to be consistent and powerful when learning from complex regions of the function space, which contain most of the data variance in a typical distribution. Our interest in boosting methods, however, lies in their ability to generally outpreform Random Forest. We were not disappointed. Using RandomizedSeachCV, we were able to achieve an accuracy score of 93.01%. We were also able to determine that the best parameters were: n_estimators=505, warm_start=True, min_samples_split=2, max_depth=2, and a learning_rate=0.021. We are confident that with further tuning the accuracy score could still be improved.

Boosting is preferred over Decision Trees and Random Forest because of its clone method - specifically bagging.

Bagging is the creation of multiple copies of the exact same model and in turn a final prediction is made using the average of all of the results. Bagging provides better generalization initially by preventing over-fitting, and, due to this, improves overall accuracy on unseen data. When prediction accuracy is measured through cross validation, the bagging algorithm provides an average of the accuracy scores of multiple trees with different bootstrap samples. This will produce different scores for each tree, so at the end we can average them to obtain a final score. For this reason, bagging is very useful in situations where data variations are very high, and samples are potentially low. It also provides an advantage over other algorithms that don't use this K-fold method when trying to predict unseen data. This is because they may not work as well on improving performance with new observations - unlike bagging [5].

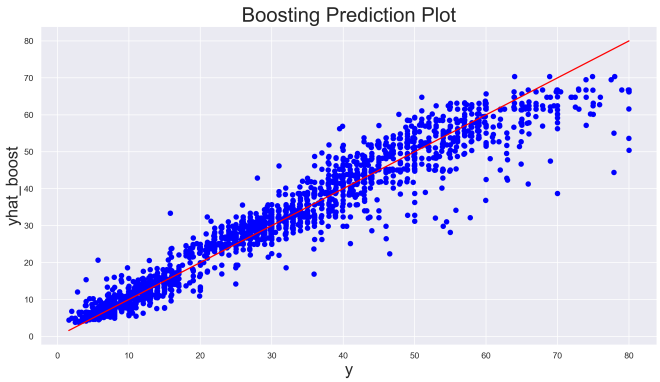

Fig. 12. Visualizing Boosting Prediction

To show the improvement in both accuracy and error, the results of the Boosting algorithm are supplied below.

Measures of accuracy and error

Root Mean Squared Error: 4.8795
Mean Squared Error: 23.8096
Mean Approximate Error: 3.2735
r^2: 0.9301
Model Accuracy*[Boosting]:* 93.01%

Below you will also find a plot for a Boosting tree diagram, *see Fig. 13.*

XGBoosting XGBoost stands for Extreme Gradient Boosting, and is a framework that allows people to train complex machine learning algorithms on large datasets [5]. Though there are many advantages to using XGBoost, one of the major benefits is that experts can use it to quickly

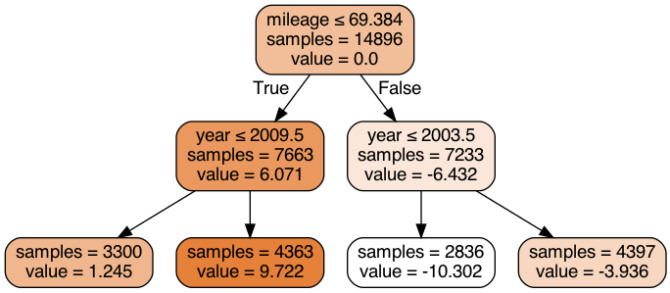

Fig. 13. Visualizing Sample Boosting Tree

and efficiently process data sets that would take months with slower methods. What's more, because XGBoost uses both stochastic gradient descent and gradient boosting algorithms, it does not need to spend time preprocessing data - which other systems often need. XGBoost has been gaining popularity over the years, and now it is one of the most popular frameworks because of its ability to train machine learning models on "big data" and winning many Kaggle competitions.

We often don't know what the optimal way to solve a problem is out of the gate. Instead, we pick an algorithm that's good at solving the problem in our data set. Generally speaking, there aren't many rules that can help you pick an algorithm for each individual data set. If you're not sure which algorithm will be best for your data set, it might seem impossible to choose one from a list of many; however, with XGBoost, just like Random Forest, you can quickly find an ideal combination.

After fitting our data to the XGBoost model and running a RandomizedSearchCV, it was determined that the best parameters for achieving the highest accuracy were setting n_estimators=753, max_depth=2, and the learning_rate=0.041. This combination triumphed and gave us our best accuracy score of 93.15%. Similar to our Boosting algorithm, I feel confident that with further optimization and hyperparameter tuning we would be able to increase that accuracy score.

Unlike some of the previous ensemble methods, overfitting is possible if the hyperparameters are not properly tuned, and generally is a much more complicated beast to handle. However, with excellent model performance, stability in the face of outliers, and swift interpretation, XGBoost is a powerhouse in the ensemble learning space.

As previously mentioned, XGBoost is very popular among Kaggle competitions. This is due to the particularly noticeable improvements in achievable model accuracy. Though many ensemble methods are renowned for having high accuracy, the fact that XGBoost can score 93% on a target validation set is remarkable. For this reason, many experts turn to XGBoost when choosing a method for their machine learning projects - usually after establishing a baseline with something like Random Forest.

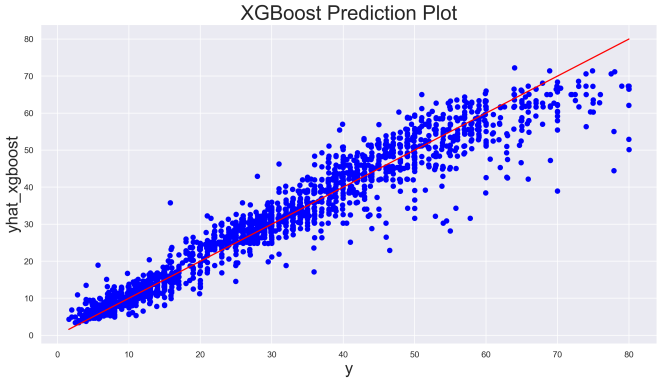

Fig. 14. Visualizing XGBoosting Prediction

The improvement to our accuracy and error is shown below in the error plot. It clearly shows that even at the beginning predictions, our accuracy increases by a large margin. This is comparable to using the GradientBoostingRegressor. In addition, while many ensemble methods suffer from unstable training; meaning they may perform poorly for certain training data sets or feature combinations, XGBoost can be awfully resilient to this problem.

Measures of accuracy and error

Root Mean Squared Error: 5.595
Mean Squared Error: 31.3044
Mean Approximate Error: 3.6897
r^2: 0.908
Model Accuracy*[XGBoosting]:* 93.15%

Finally, we come to the plot of the XGBoost tree diagram, *see Fig. 15*. This diagram looks different from the rest due to the required processing and necessary visualization differences required by XGBoost. There is, unfortunately, no way to currently use graphviz.

There are a few improvements we could make to help increase the accuracy of our XGBoost model. Namely, we can increase the range of potential hyperparameter values and use the computationally expensive GridSearchCV to test all potential values instead of RandomizedSearchCV. Additionally, we could use Bayesian optimization techniques such as Sequential Model-Based Optimization for General Algorithm Configuration (SMAC) or an Autoencoder to improve overall accuracy and efficiency.

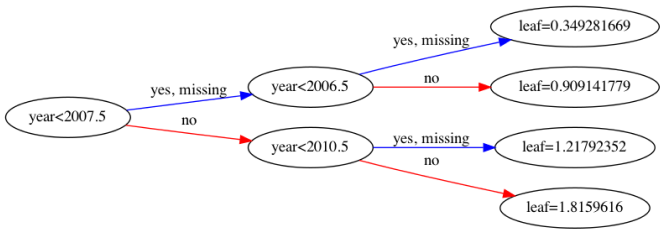

Fig. 15. Visualizing Sample XGBoosting Tree

VI. Future Directions

While the journey began by learning to crawl, currently we are still learning to walk through our deep, dark forest. Achieving success in mastering the many ensemble methods requires developing many different skills, and it would be an absolute tragedy if we did not even attempt to develop some of them. Ideally, in the near-term, we will continue developing the existing systems — specifically Isolation Forests, Random Forests, XGBoosting, and other potential alternatives. These methods will continue to be developed in the near-term to better understand why they are successful and what can be done to reduce errors to further improve scoring accuracy. In the long term, we could move into neural networks - specifically incorporating softmax to classify only those forecasts that have a score above a certain threshold - this advancement is predicted to significantly improve our accuracy.

One final avenue worth pursuing is to develop ensemble models that leverage the power of other machine learning algorithms - such as introducing an autoencoder to the XGBoost algorithm.

VII. Conclusion

The purpose of this project was to identify appropriate methods which can predict with reasonable accuracy the price of used automobiles from a representative data set. Employing several well-known machine learning methods, namely LASSO, Isolated Forests, Decision Trees, Random Forests, Boosting, and XGBoost we have achieved an impressive 93.1520% score on an out-of-sample test set.

By applying each of the models to our data set, we have identified that the model which consistently achieved the highest accuracy was the XGBoosting model.

To accomplish this, we first started by applying a wide variety of outlier detection and feature engineering techniques to our dataset in order to derive as much relevant information as possible from the attributes provided. We then used LASSO to preform dimensionality reduction. XGBoost was then used on top of this final step to achieve near perfect scores on our test set — showing promising evidence that it will be ideal for use in an actual working environment.

References

[1] L. Wood. United states used car market volume report 2020: Market is forecasted to be more than $885 billion by the end of 2026. [Online]. Available: https://bit.ly/3vvMBD0

[2] T. James. Used vehicle market poised for record sales in 2019, according to new report from edmunds. [Online]. Available: https://www.edmunds.com/industry/press/used-vehicle-market-poised-for-record-sales-in-2019-according-to-new-report-from-edmunds.html

[3] M. Bentenrieder and S. Coccorullo. A better approach to residual value. [Online]. Available: https://www.oliverwyman.com/our-expertise/insights/2019/jun/a-better-approach-to-residual-value.html

[4] S. Agarwal. Understanding the data science lifecycle. [Online]. Available: http://sudeep.co/data-science/Understanding-the-Data-Science-Lifecycle/

[5] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc, 2019.

[6] Miro. Isolation forest image. [Online]. Available: https://miro.medium.com/max/875/0*0GuMixLdSZo3V3Nh.