Internship Project-Predicting Used Car Prices

Introduction

Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

Dataset and Pre-Processing

For this project, we are using the dataset on used car sales from all over the United States from TrueCar. The features available in this dataset are Mileage, Make, Model, Year, State, and City.

Pruning: A histogram of the dataset indicated many outliers, for example, cars with really low mileages that were unrepresentative of a “used" car. Therefore, we pruned our dataset to three standard deviations around the mean.

One-Hot Vectorization: We converted the Make, Model, and State into one-hot vectors. Since we had over 2OOO unique cities in the dataset, we replaced the string representing the city with a boolean which was set if the population of the city was above a certain a threshold i.e. a major city.

Linearity: To analyze the degree to which our features are linearly related to price, we plotted the Price against Mileage and Year for a particular Make and Model. There seemed to be a fair degree of linearity for these two features.

Methodology

We utilized several classic and state-of-the-art methods, including ensemble learning techniques, with a **9O-1O split** on test and training data. To reduce the time required for training, we used 5OOk examples from our dataset (of almost a million examples).

1. **Linear Regression**: Quick to train and test as a baseline algorithm.
2. **Gradient Boost**: To account for non-linear relationships, by splitting the data into 1OO regions.
3. **Random Forest**: To account for a large number of features in the dataset and compare a bagging technique with the Gradient Boost method.
4. **XGBoost**: To improve performance compared to standard Gradient Boosting using regularization, second order gradients and added support for parallel compute.
5. **KMeans + Linear Regression Ensemble**: In order to capitalize on the linear regression results using ensemble learning,
6. **Light GBM**: To improve performance compared to Gradient Boosting with a leaf-wise-tree growth approach and improved speed compared to XGBoost.

Results

Learning Algorithm	Test Data R2 Score	Training Data R2 Score
Linear Regression	O.87	O.87
Gradient Boost	O.64	O.64
Random Forest	O.88	O.98
Light GBM	O.81	O.82
XGBoost	O.78	O.81
KMeans + LinReg	O.88	O.89

Compared to Linear Regression, most Decision-Tree based methods did not perform comparably well. This can be attributed to the apparent linearity of the dataset. It can also be attributed to the dificulty in tuning most gradient boost methods. The exception to this is the Random Forest method which marginally outperforms Linear Regression. However Random Forests tend to overfit the dataset. Building up from the relatively good performance of Linear Regression, the KMeans + Linear Regression Ensemble Learning Method (with K = 3) produced the best R2 Score on test data without high variance as it fits linear relationships categorically.

Future Work and References

We plan to utilize **Deep Neural Networks** to improve our prediction performance while avoiding overfitting. In addition to this, we shall **tune Decision-Tree parameters** like the number of trees, depth, etc. and sub-sample datapoint.