Capstone 2 Report

Problem Statement

Used car market has always been frequently discussed as the prices and depreciation of used cars are very tricky for both sellers and buyers and there is a considerable amount of transaction cost. This project is dedicated to provide some insights on the pricing mechanism and help to reduce the information cost.

For both dealers/sellers and buyers, this model would provide a good reference on pricing the vehicle and price transparency, which could improve the market efficiency by reducing the information cost and simplify the decision making process.

Data Source

https://www.kaggle.com/lepchenkov/usedcarscatalog/data
Author: Kirill Lepchenkov
The dataset is collected from various web resources in order to explore the used cars market and try to build a model that effectively predicts the price of the car based on its parameters (both numerical and categorical)
The data is scraped in Belarus (western Europe) on the 2nd of December 2019.

Data Cleaning

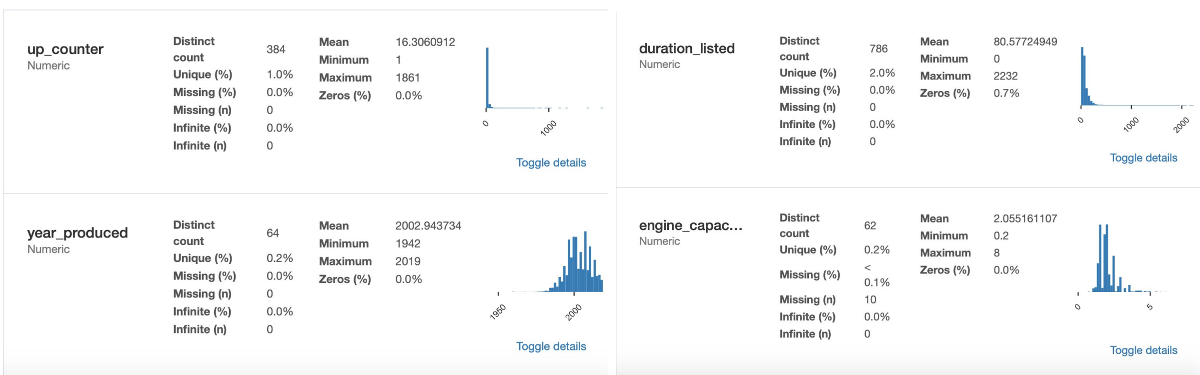
The original dataset has 38531 entries and 30 columns, all of them are non-null but 10 of them are nan and 2 columns are float64, 18 columns are int64, and 10 columns are object.

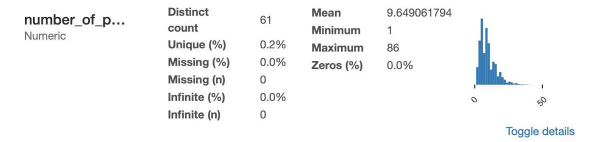
Data Wrangling

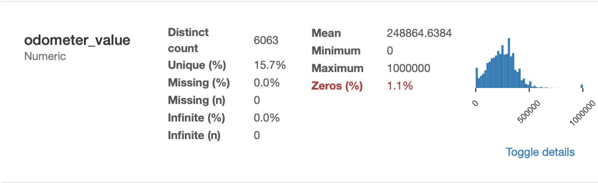
Boolean columns were then converted to numerical values. For categorical columns with less than 10 distinct values, one hot encoding was applied and then high correlation columns were dropped. For categorical columns with high cardinality, target encoding and hashing encoding were applied to generate 2 versions of the data, and missing values were filled with mean after splitting training and testing data.

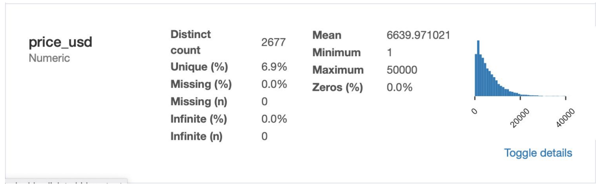
Exploratory Analysis

Some columns such as price and odometer value seem to have distribution patterns.

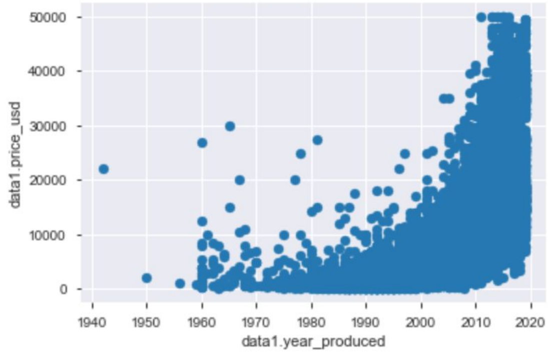


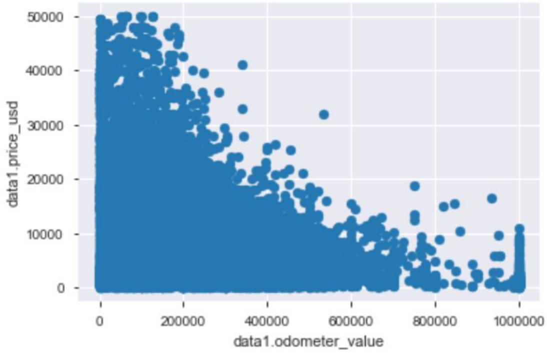




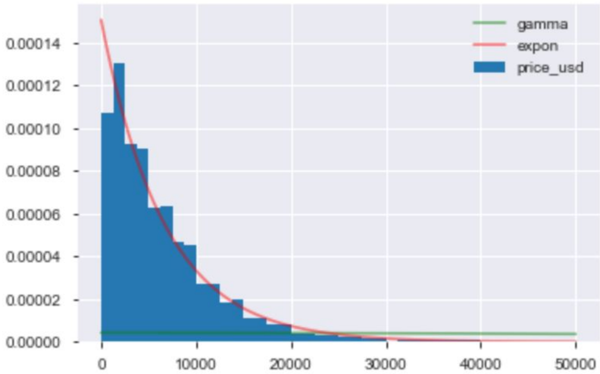


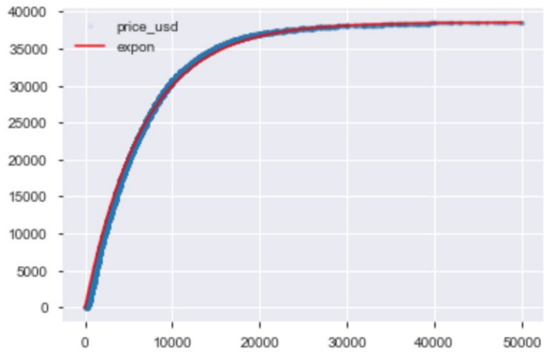
And some columns seem to be correlated: year produced and price seem to be positively correlated while odometer value and price seem to be negatively correlated, which is aligned with common sense on used cars.

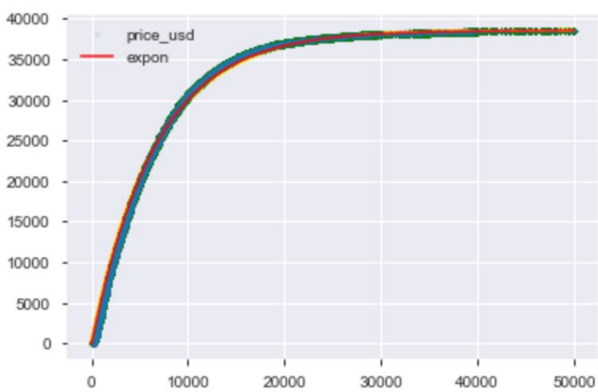




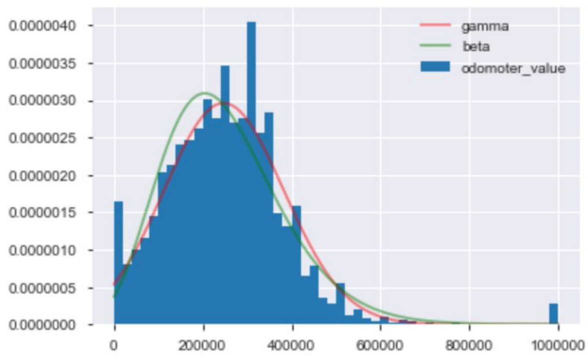
From price's histogram and empirical cumulative distribution it looks like price is exponentially distributed, and the bootstrap test (n=1000) suggests so as well.

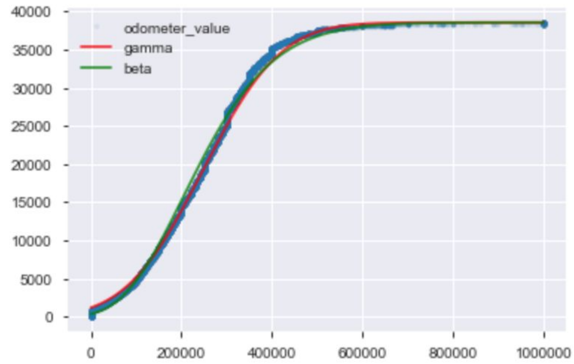




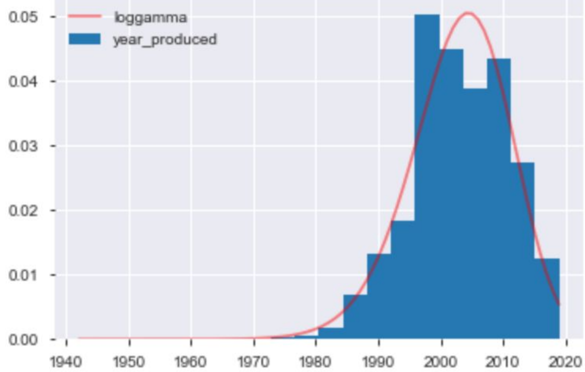


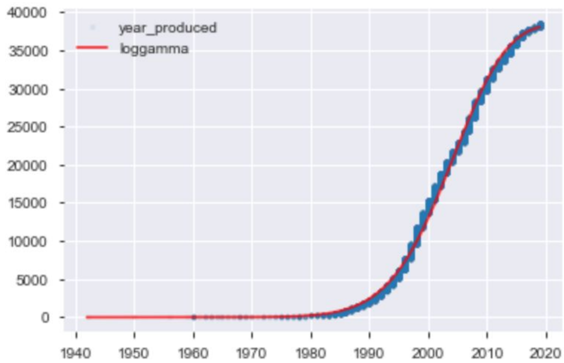
As for odometer value’s histogram and empirical cumulative distribution, it seems to be gamma distributed.

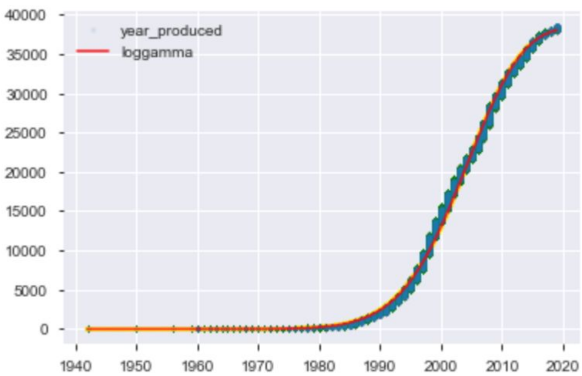




And year produced’s histogram, empirical cumulative distribution and the bootstrap test (n=1000) suggest it is log gamma distributed.







In-Depth Analysis

The 2 datasets (hashing encoding version and target encoding version) are split into training and test sets (test dataset size is 30% of the entire dataset) and fitted into various models which are then evaluated using 4 metrics: MSE, MAE, R2 score and explained variance score. The baseline for comparison is from fitting the data into a dummy model.

For target encoded data, I started with Bayesian models. Unfortunately I encountered some computation issues when trying ARD regression models and couldn’t get any results. Bayesian Ridge models have good performance in general.

For hashing encoded data, I tried ensemble models first. Gradient Boost, XG Boost and Random Forest models all have better performance compared to Bayesian models with target encoded data, Random Forest seem to have slightly better performance than the other two ensemble methods.

Then I tried a few more ensemble models on target encoded data for comparison purposes. I fitted target encoded data into the ensemble models and did the same tuning as well, the models' performance improved and Random Forest still seems to be the better method.

A few dense network models were tried on both target encoded and hashing encoded datasets. Models with up to 3 Dense layer and 100 nodes were trained and in general models trained with target encoded data have better performance compared to those with hashing encoded data. However, ensemble and Bayesian methods have better performance over the network models.

Results

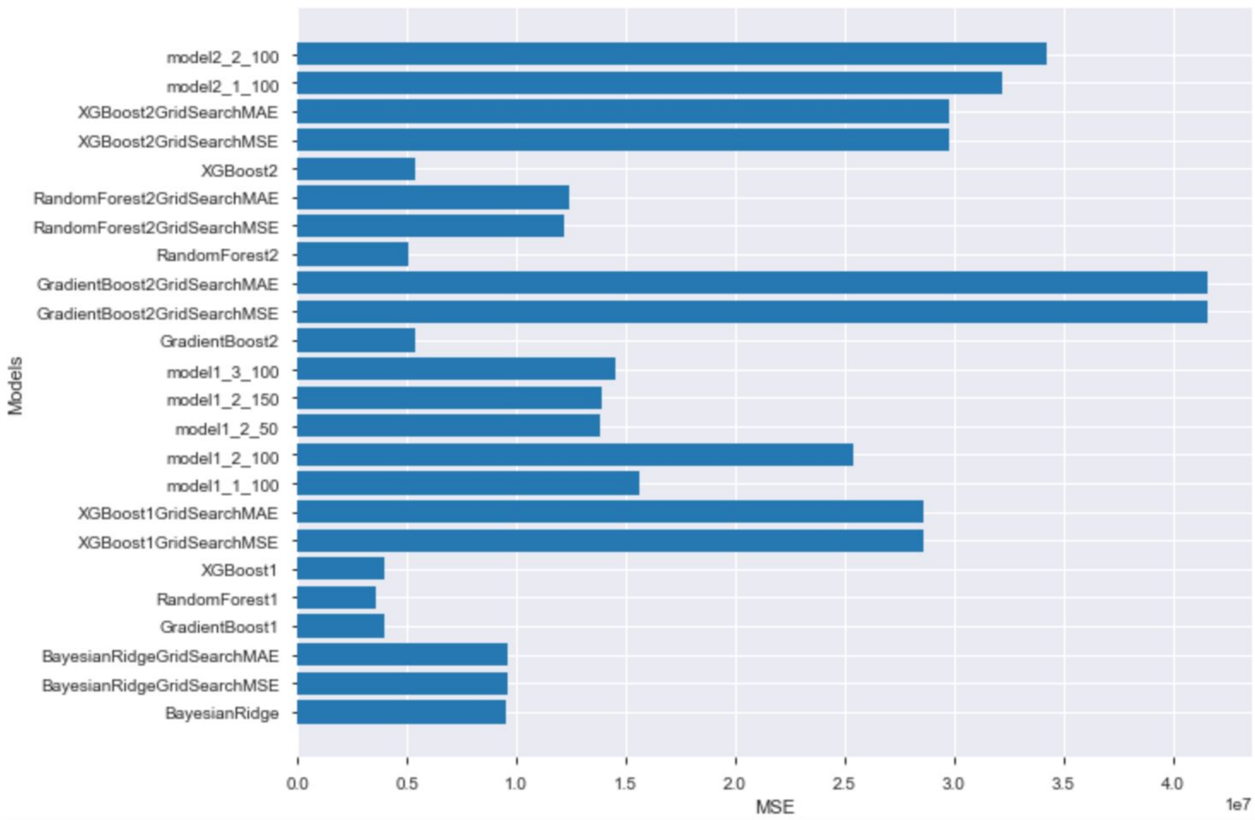
For target encoded data, all models perform better than the dummy model, Gradient Boost, Random Forest and XG Boost have the best performance out of all models listed, Bayesian Ridge and tuned Bayesian Ridge models are less accurate but still better than network models.

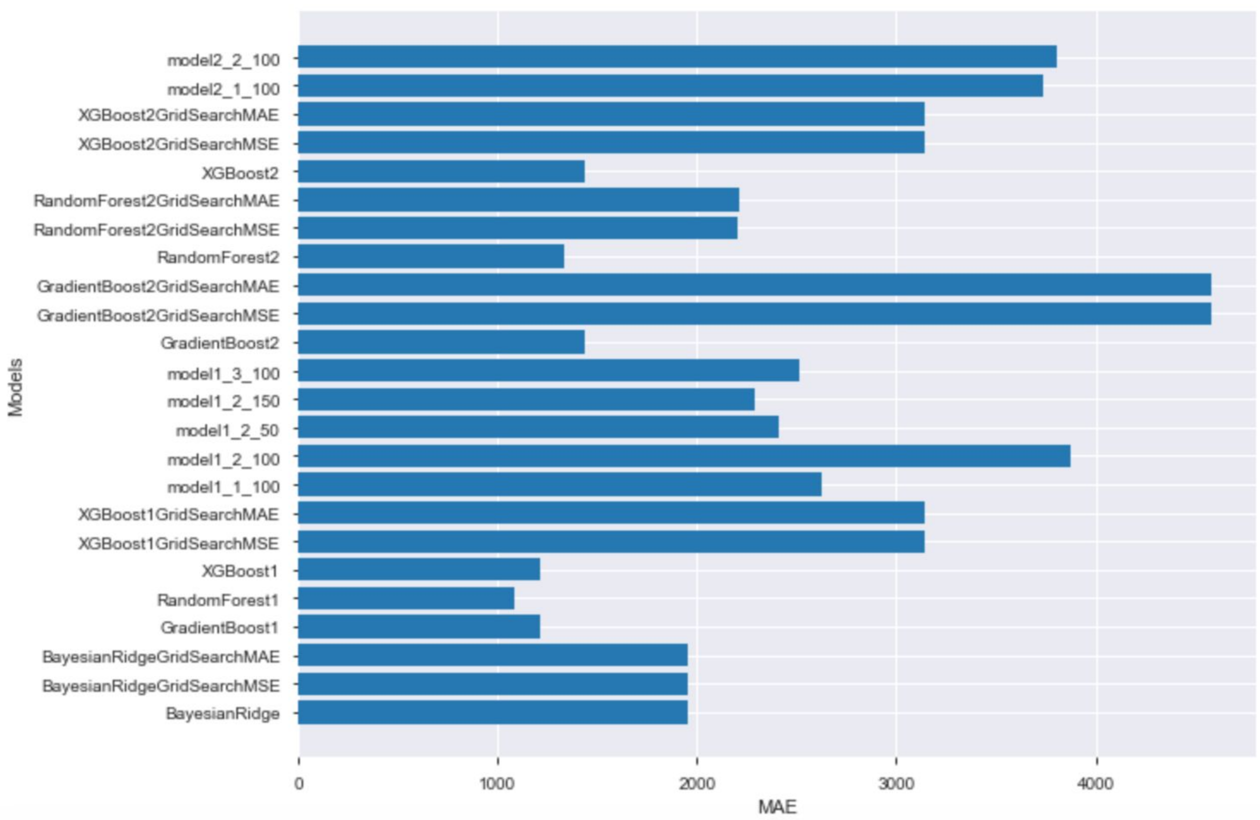
For hashing encoded data, one Gradient model with tuned hyperparameters has the same performance as the dummy model while other original ensemble models have the best performance out of all models listed. Bayesian models still seem to be better than network models.

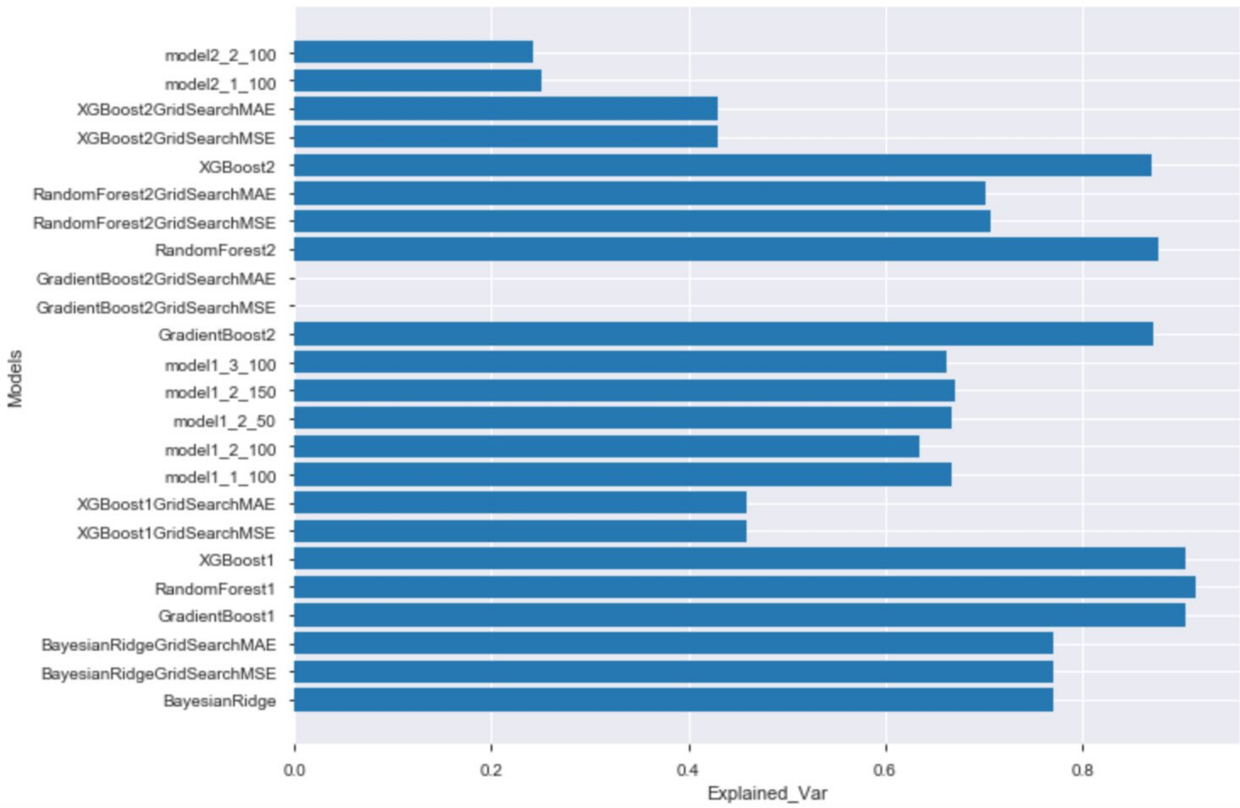
All scores are listed below:

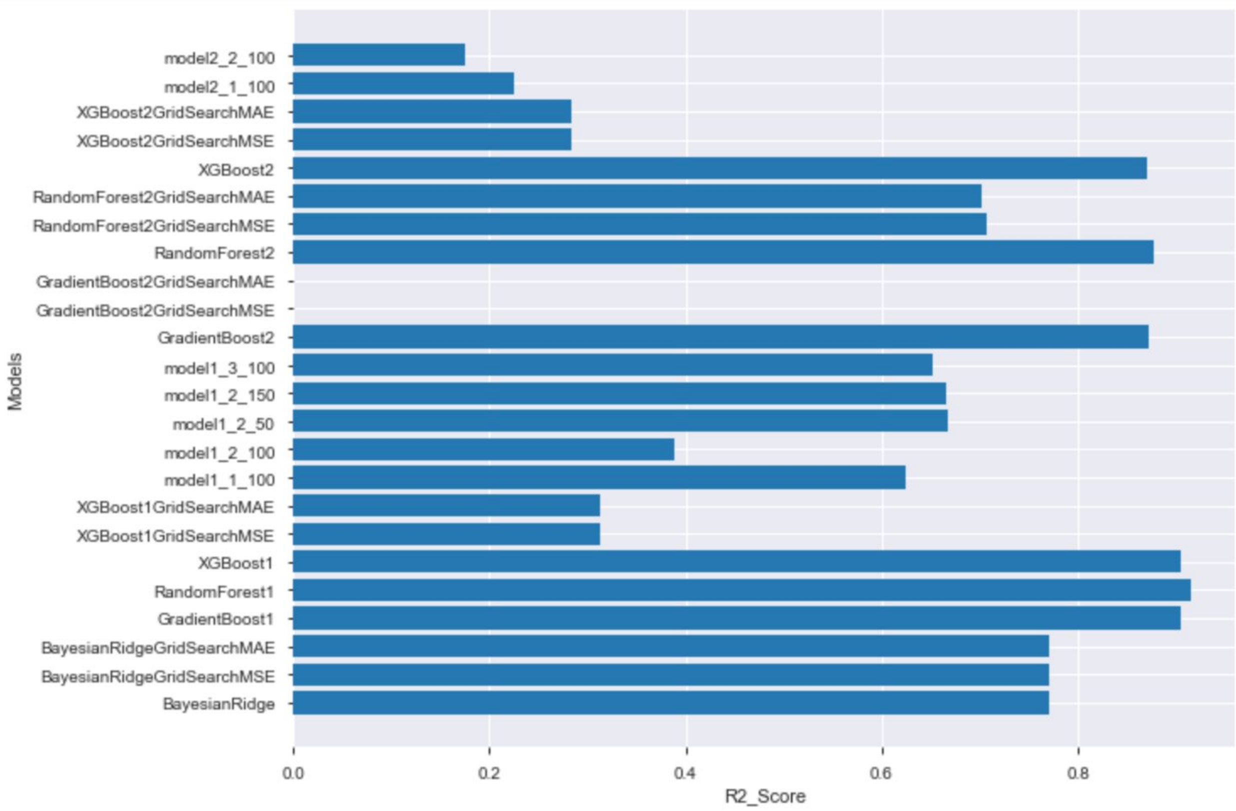
Model	Data	MSE	MAE	R2 Score	Explained Variance Score
Dummy	One hot encoding & target encoding	41555324.89972137	4578.682299877935	0.0	-9.359094948413471e-05
BayesianRidge	One hot encoding & target encoding	9546128.604215752	1952.2439200301233	0.7702596666848693	0.7702575528288137
BayesianRidgeGridSearchMSE	One hot encoding & target encoding	9562581.18542289	1953.7610644640974	0.7698645392022938	0.7698615958471383
BayesianRidgeGridSearchMAE	One hot encoding & target encoding	9562581.185404427	1953.7610642582426	0.7698645392010696	0.7698615958475826
GradientBoost	One hot encoding & target encoding	3983567.3209468764	1212.9329003864452	0.9041293884990883	0.9041292504291913
RandomForest	One hot encoding & target encoding	3677942.577723576	1090.9224569440696	0.911486179583577	0.9114845857002014
XGBoost	One hot encoding & target encoding	3999238.2390208724	1214.6457761973081	0.9037524323593867	0.9037521053877816

XGBoost1GridSearchMSE	One hot encoding & target encoding	28571334.853303522	3146.171982372033	0.45968659459907735	0.3123863442142871
XGBoost1GridSearchMAE	One hot encoding & target encoding	28571334.853303522	3146.171982372033	0.45968659459907735	0.3123863442142871
model1_1_100	One hot encoding & target encoding	14160433.22876375	2517.182053211087	0.6712547957214745	0.6592071280556949
model1_2_100	One hot encoding & target encoding	13795553.049855424	2340.3378508645424	0.6684166719481491	0.6679885376408998
model1_2_50	One hot encoding & target encoding	14333917.262186868	2565.760145237484	0.6647914154363566	0.6550319646951098
model1_2_150	One hot encoding & target encoding	14562651.905315619	2448.6211418614043	0.6683682057705136	0.6495271093926137
model1_3_100	One hot encoding & target encoding	14035659.975215843	2390.7809280997412	0.6627814780105654	0.6622099906610601
Dummy	One hot encoding & hashing encoding	41555324.89972137	4578.682299877935	0.0	-9.359094948413471e-05
GradientBoost	One hot encoding & hashing encoding	5354689.9285972575	1432.5466837692136	0.871151712801502	0.8711310501834681
GradientBoostGridSearchMSE	One hot encoding & hashing encoding	41548320.14531531	4578.231427053105	0.00016855414724414874	7.498937095795633e-05
GradientBoostGridSearchMAE	One hot encoding & hashing encoding	41548320.14531531	4578.231427053105	0.00016855414724414874	7.498937095795633e-05
RandomForest	One hot encoding & hashing encoding	5093579.896415042	1348.2142707057683	0.8774988873211101	0.8774150696285863
RandomForestGridSearchMSE	One hot encoding & hashing encoding	12289288.398387972	2195.62106019855	0.7042471317087462	0.7042391415729234
RandomForestGridSearchMAE	One hot encoding & hashing encoding	11529432.545138331	2114.3982644049665	0.7225287056527598	0.7225262556964256
XGBoost	One hot encoding & hashing encoding	5382670.103059661	1434.6297797936065	0.870478153997942	0.8704576637228636
XGBoost2GridSearchMSE	One hot encoding & hashing encoding	29749674.552090026	3139.7713408769123	0.42964813110609135	0.2840277648129351
XGBoost2GridSearchMAE	One hot encoding & hashing encoding	29749674.552090026	3139.7713408769123	0.42964813110609135	0.2840277648129351
model2_1_100	One hot encoding & hashing encoding	33135927.044444196	4297.972246364078	0.2546927614395935	0.20253232654810693
model2_2_100	One hot encoding & hashing encoding	32803207.80149969	4186.972484155358	0.23155047908681137	0.21053973313817298


model2_2_100
model2_1_100
XGBoost2GridSearchMAE
XGBoost2GridSearchMSE
XGBoost2
RandomForest2GridSearchMAE
RandomForest2GridSearchMSE
RandomForest2
GradientBoost2GridSearchMAE
GradientBoost2GridSearchMSE
GradientBoost2
model1_3_100
model1_2_150
model1_2_50
model1_2_100
model1_1_100
XGBoost1GridSearchMAE
XGBoost1GridSearchMSE
XGBoost1
RandomForest1
GradientBoost1
BayesianRidgeGridSearchMAE
BayesianRidgeGridSearchMSE
BayesianRidge
Models
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1e7
MSE


model2_2_100
model2_1_100
XGBoost2GridSearchMAE
XGBoost2GridSearchMSE
XGBoost2
RandomForest2GridSearchMAE
RandomForest2GridSearchMSE
RandomForest2
GradientBoost2GridSearchMAE
GradientBoost2GridSearchMSE
GradientBoost2
model1_3_100
model1_2_150
model1_2_50
model1_2_100
model1_1_100
XGBoost1GridSearchMAE
XGBoost1GridSearchMSE
XGBoost1
RandomForest1
GradientBoost1
BayesianRidgeGridSearchMAE
BayesianRidgeGridSearchMSE
BayesianRidge
Models
0
1000
2000
3000
4000
MAE


model2_2_100
model2_1_100
XGBoost2GridSearchMAE
XGBoost2GridSearchMSE
XGBoost2
RandomForest2GridSearchMAE
RandomForest2GridSearchMSE
RandomForest2
GradientBoost2GridSearchMAE
GradientBoost2GridSearchMSE
GradientBoost2
model1_3_100
model1_2_150
model1_2_50
model1_2_100
model1_1_100
XGBoost1GridSearchMAE
XGBoost1GridSearchMSE
XGBoost1
RandomForest1
GradientBoost1
BayesianRidgeGridSearchMAE
BayesianRidgeGridSearchMSE
BayesianRidge
Models
0.0
0.2
0.4
0.6
0.8
Explained_Var


model2_2_100
model2_1_100
XGBoost2GridSearchMAE
XGBoost2GridSearchMSE
XGBoost2
RandomForest2GridSearchMAE
RandomForest2GridSearchMSE
RandomForest2
GradientBoost2GridSearchMAE
GradientBoost2GridSearchMSE
GradientBoost2
model1_3_100
model1_2_150
model1_2_50
model1_2_100
model1_1_100
XGBoost1GridSearchMAE
XGBoost1GridSearchMSE
XGBoost1
RandomForest1
GradientBoost1
BayesianRidgeGridSearchMAE
BayesianRidgeGridSearchMSE
BayesianRidge
Models
0.0
0.2
0.4
0.6
0.8
R2_Score

From all models’ scores (dummy models excluded) above, it is clear that ensemble methods have better performance especially with target encoding, Bayesian Ridge is also a good alternative, and networks models I tried are not the best options. In general models with target encoding perform better. Due to time and computation cost concerns I didn’t tune each model very well and ARD models were not included in the results, it’s possible that there are better encoding methods such as different combinations of the encoding methods used here and models such as other Bayesian models and might give better predictions with more efforts spent on tuning the hyperparameters.

If given limited time and computation power, I would suggest using Random Forest or XG Boost with one hot encoded and target encoded data, and update the model at least once a year and when prediction’s scores fall close to a dummy model’s prediction score. As for data collection, I suggest dropping the redundant or similar features such as engine type and if engine has gas, and maybe adding some new features such as if had major accident, if modified, times of being traded.

I believe with good quality data, proper implementation, the suggested model would perform effectively on reducing information cost in the used car market and promote market efficiency. This model can be used on mobile apps and websites for consumers and combined with offers from dealers or customer reviews of dealers to help consumers make better decisions with less efforts. It can also be used for dealers to automate some processes during the transaction to cut the cost and make the price more transparent to consumers. There are many more applications for this model that would generate economic and social benefits, and I hope more people would be interested in using and improving the model.