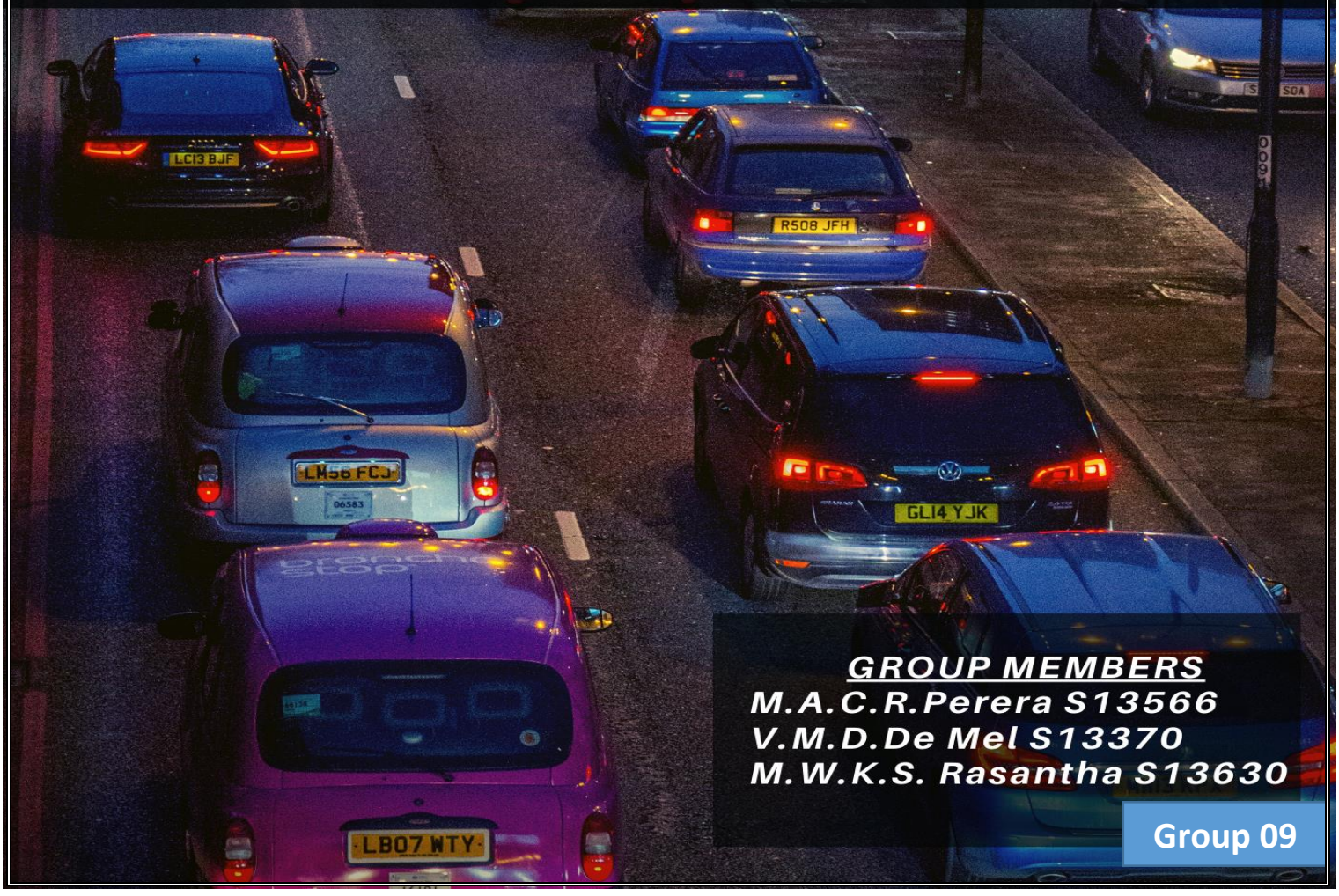
ST 4052-DATA ANALYSIS PROJECT-II

PRICE SUGGESTION FOR AUCTIONEXPORT.COM



GROUP MEMBERS

M.A.C.R.Perera S13566
V.M.D.De Mel S13370
M.W.K.S. Rasantha S13630

Group 09

Abstract

AUCTIONEXPORT.com is an online market place for selling and auctioning used vehicles. This data set contains details about 2499 cars under 13 variables. The prices of the cars in the data set are determined by the owners. This data set includes the price which is demanded by the owners based on the characteristics of the cars and their personal requirements. The purpose of this study is to suggest a price for a car when an owner publishes an advertisement. For a new host it will be beneficial as this will provide him with an idea about how his pricing should be in order to remain in the business. According to the suggestions from the descriptive analysis, Ridge Regression, Lasso Regression, Elastic Net Regression, Tree based methods and Boosting techniques were used to predict the price. Models fitted separately for clean titled vehicles and salvage insurance vehicles resulted in a lower test RMSE compared to the other models.

Table of Contents

List of Figures

List of Tables

Introduction

Auction Export, established in 2007, specializes in helping individuals around the world purchase and export the highly-desirable cars found in the USA and Canada. Auction Export's website (www.auctionexport.com) provides simple solutions for individuals to locate, purchase and pay for vehicles, as well as many other logistical issues such as transportation and shipping to virtually any port in the world. Auction Export's current client base extends up to 79 countries. This website

enables their customers to advertise their vehicles for a fixed price as well as for bidding. This data set has been obtained from the website mentioned above. This project focuses on 2499 cars that have been advertised in the website. The price of a car will be predicted using Ridge Regression, Lasso Regression, and Elastic Net Regression and Tree based methods as suggested from the descriptive analysis

The Question that will be answered

AUCTION EXPORT.com is a web portal to the world of North American wholesale auto auctions. This innovative technology allows international buyers to search, buy and export any vehicle of their choice. After going through the variables in the data set , our attention was drawn towards developing a model to provide a price suggestion for a vehicle that is going to be advertised. This will eventually be a helpful service for AUCTION EXPORT.com customers, when they advertise their vehicle and open them for bidding. It is clear that a realistic price will attract more bids from the buyers.

About the data set

There were 2499 observations under 13 variables.

No	Variable	Description
01	index	Index number of the vehicle
02	price	The sale price of the vehicle in the ad
03	brand	The brand of the car
04	model	Model of the vehicle
05	year	The vehicle registration year
06	title_status	This feature included binary classification, which are clean title vehicles and salvage insurance
07	mileage	Miles traveled by the vehicle
08	color	Color of the vehicle
09	vin	The vehicle identification number is a collection of 17 characters (digits and capital letters)
10	lot	A lot number is an identification number assigned to a particular quantity or lot of material from a single
11	state	The state in which the car is being available for purchase
12	country	The country in which the car is being available for purchase
13	condition	Time to the end of the auction from the snapshot date

Table 1: Description of the data set

Important Results of the descriptive analysis

1. There were no missing values in the dataset
2. Price = 0: 43 cars were found to have 0 price in the website since no customers have placed a bid. Those records were discarded.
3. The variable “country” had 2 countries namely USA and Canada. There were only 7 cars from Canada. Those entries were discarded and the study was restricted to USA.
4. The states of Nevada and Kentucky were found to have the most expensive cars.
5. Salvage insurance titled cars have lower price compared to other cars.

6. More than 95% of the cars are clean titled, leaving less than 5% to be salvage insurance titled. Hence we propose imbalance handling for this variable to avoid losing important information about price.
7. Most vehicles belong to 6 main colors namely, black, silver, blue, red, white and grey.
8. Continuous variables seemed to be correlated

Important Results of the advanced analysis

- An imbalance in a potentially important variable "title_status" was observed during the descriptive analysis. Hence we tried fitting all the models twice for the original dataset and the dataset where imbalance handling techniques have been applied.
- First, several models were fitted to the original data set and RMSE values were recorded.
- Since the values obtained were high, imbalance handling was applied to the variable "title_status" as an attempt to reduce the RMSE
- The imbalance handling techniques applied were over sampling, a combined approach of over and under sampling, and ROSE technique.
- The results obtained were not satisfactory and hence our attention was drawn to a different approach.
- Fitting models separately for salvage and clean titled cars was attempted. Here, the same test set was used. That is the train data sets for salvage and clean titled cars were built by the separation of the original train data set and the same was done for the test data set as well.
- Following is a summary of the results obtained.

- Step 1 : Comparison between the original dataset and the datasets which imbalance handling has been applied.

		Original Dataset	Over sampled Dataset	Mixed approach – Over sampling and Under Sampling	ROSE Technique
Ridge Regression	λ_{min}	7452.1420	7789.4227	8292.6353	8436.6555
	λ_{1sd}	7800.0665	8065.3373	8369.0073	8617.8437
Lasso Regression	λ_{min}	7426.8505	7736.8766	8235.7859	8398.2320
	λ_{1sd}	7800.0665	7936.1586	8369.0073	8572.6108
Elastic Net Regression	λ_{min}	7406.4093	7732.1624	8229.2003	8369.1592
	λ_{1sd}	7800.0665	7913.4603	8369.0073	8562.3913
Random Forest Regression		6983.6521	7025.3120	7567.5207	8086.4535
XG Boosting		7454.6258	12736.7833	12886.6050	12883.6526
Gradient boosting		7558.9567	7732.8941	8045.6396	8256.3149
SVR – Rbf Kernel		11497.2898	16789.5090	16785.4334	16462.6343
SVR – Linear Kernel		12409.5288	25660.4868	26759.5025	29367.0176

Table 2: Comparison of Original dataset with balanced data set

- Step 2 : Comparison between two separated models fitted to the two categories salvage and clean titled vehicles.

		Salvage Insurance	Clean Vehicle
Ridge Regression	λ_{min}	4388.0900	7429.6095
	λ_{1sd}	4612.5454	7783.6326
Lasso Regression	λ_{min}	4614.5275	7404.4674
	λ_{1sd}	4606.4972	7553.4651
Elastic Net Regression	λ_{min}	4517.7511	7403.2321
	λ_{1sd}	4600.1162	7555.4261
Random Forest Regression		4772.9874	7072.3451
XG Boosting		4905.5344	12437.2044
Gradient Boosting		4681.4695	7724.3343
SVR – Rbf Kernel		5155.7968	12118.7016
SVR – Linear Kernel		8017.0761	10807.7855

Table 3: RMSE Values for separate data sets

- The lowest RMSE value for Salvage insurance titled cars could be found from the Ridge Regression model, and that for clean vehicles could be found from the Random Forest Regression Model
- Hence the combined RMSE could be found as

$$\sqrt{\frac{(RMSE_{Salvage})^2 \times n_{Salvage} + (RMSE_{Clean})^2 \times n_{Clean}}{n_{Salvage} + n_{Clean}}} = 6977.4643$$

- Notice that this RMSE value is less than the RMSE value we obtained for the dataset which was not separated, which is 6983.6521.
- Hence the lowest RMSE was given by the combined models: Ridge Regression for Salvage Insurance vehicles and Random Forest Regression for clean titled vehicles.
  Ridge Regression Model for Salvage Insurance:

$$\lambda_{Min} = 2892.218$$

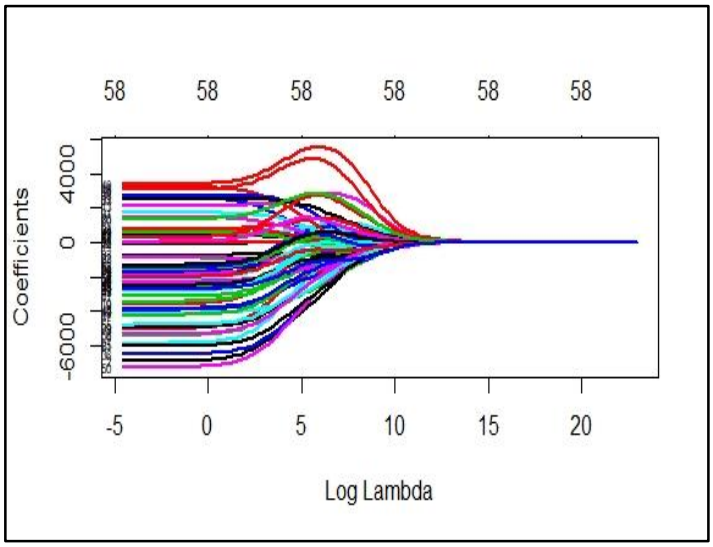


Figure 1:Variation of Coefficients with Log Lambda

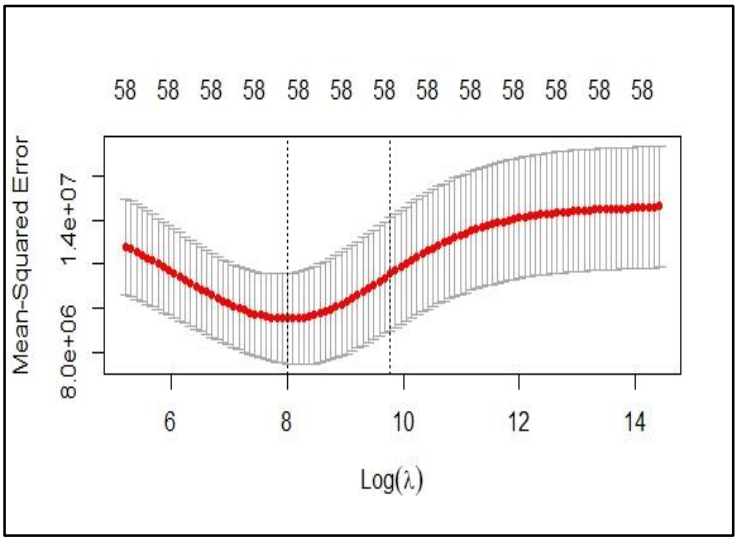


Figure 2: Varuation of MSE with Log Lambda

Model Coefficients:

Variable	Coefficient	Variable	Coefficient	Variable	Coefficient
(Intercept)	-2.5915e+05	year	1.3044e+02	stateindiana	-9.3075e+02
brandaudi	-2.3819e+03	mileage	-1.6964e-03	statekansas	-1.0525e+03
brandbmw	-5.1893e+02	colorbrown	-4.0776e+02	statekentucky	-1.1352e+03
brandchevrolet	-1.1972e+03	colorblue	-9.3887e+02	statemaryland	-8.0553e+02
brandchrysler	-1.6700e+03	colorgold	-6.5195e+02	statemassachusetts	2.9955e+03
branddodge	3.9632e+02	colorgreen	-3.2246e+01	statemichigan	-8.8757e+02
brandford	8.5195e+02	colorgrey	-5.1365e+02	stateminnesota	-7.7486e+02
brandgmc	-9.9176e+02	colorno_color	1.7040e+03	statemississipi	-2.4812e+02
brandjeep	-4.7614e+02	colororange	-1.0657e+03	statemontana	-8.2835e+02
brandnissan	-1.9521e+02	colorred	2.9359e+02	statenevada	-1.2060e+03
brandpeterbilt	-6.1929e+02	colorpurple	2.7593e+02	statenewjersey	1.0878e+03
modelcoupe	-4.3513e+02	colorsilver	-4.6761e+01	statenorthcarolina	-1.8194e+02
modelcutaway	5.6901e+02	colorwhite	7.5022e+02	stateohio	7.9281e+01
modeld	2.5123e+03	coloryellow	9.6021e+02	stateoklahoma	-1.0215e+03
modeldoor	-2.7883e+02	statecalifornia	4.9435e+02	stateoregon	-6.9830e+02
modeldr	7.9276e+03	statecolorado	-1.1905e+03	statepennsylvania	-2.1149e+03
modelpickup	-3.4370e+02	stateconnecticut	1.3646e+02	statesouthcarolina	4.2779e+02
modelpk	-8.3257e+02	stateflorida	4.1287e+02	statetennessee	-5.4845e+02
modeltruck	-6.1918e+02	stategeorgia	-6.4000e+02	statetexas	1.9536e+03
modelvan	-1.0246e+03	stateidaho	3.8795e+03	statevirginia	-9.1344e+02
modelvehicl	-1.1365e+03	stateillinois	-8.4849e+02		

Table 4: Model Coefficients -Salvage Insurance Category

Random Forest Regression Model for Clean titled vehicles:
Model Parameters

Parameter	Best Value
n_estimators	1600
min_samples_split	5
min_samples_leaf	1
max_features	‘sqrt’
max_depth	70
bootstrap	False

Table 5: Model Parameters : Clean vehicles

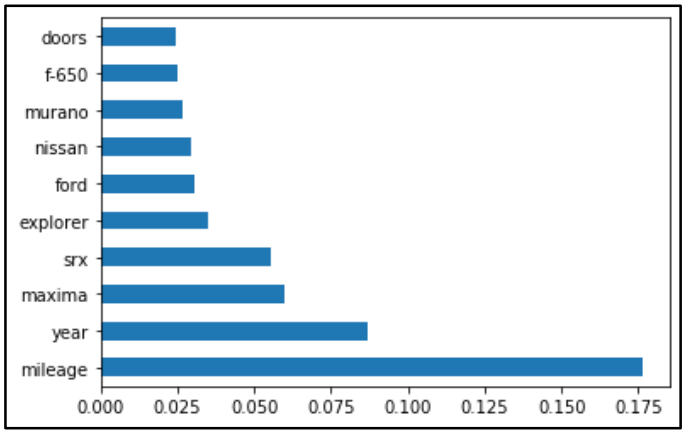


Figure 3: Feature Importance Plot

Feature Importance Plot

It was implied from the descriptive analysis that there is some association between mileage and price. Usually people prefer to buy a vehicle with low mileage. If a vehicle has low mileage, an extra value can be added to that vehicle. Also if a vehicle has high mileage, the price may be low for the vehicle. Hence mileage is a major variable that determines the price of vehicles. People also prefer buying a

new vehicle rather than an old one. That is, vehicles with a recent year of manufacture. Year is another important variable in determining prices of vehicles. A reasonable price can be added according to its year of manufacture. Addition to that, very old vehicles with antique value will also have high prices. Those two variable seem to play a major role in the price determination. Apart from those, the other variables that are among the first ten important features are either brands or models. Ford and Nissan are the brands that have been considered to be important and the other ones are models. It was noticed that the models Maxima, and Murano come from the brand Nissan while the models f-650 and Explorer come from the brand Ford. After a further research on this fact, we were able to figure out that both Nissan and Ford are among the top 10 famous brands in the USA. (Source: www.carlogos.org/popular-car-brands/). Hence the belonging to those brands will be an important fact that affects the price.

Issues Encountered and Proposed Solutions

- Problem: An imbalance could be observed in the predictor variable "Title_status" which says if the vehicle is in the clean condition or not. This variable was found to have an association with the response variable, "price".
  Solution: Apply imbalance handling techniques to handle the imbalance and check whether there is a reduction in the RMSE value.
- Problem : Very high test RMSE value was obtained for the model fitted using the complete dataset (both balanced and imbalanced)
  Solution : Fit separate models to each category of "title_status"
- Problem: The test RMSE remained high (though lower than the initial value) even when the dataset was split into categories by the variable "title_status".
  Solution: Try an advanced approach such as deep learning

Discussion and Conclusions

- Creating training set and test set: First the dataset was divided into training and test sets in the ratio of 80:20. The best model was found to have a very high test RMSE. Hence an attempt was made to fit separate models for the clean vehicles and salvage insurance vehicles separately. The same training set and test set were used. Training set was divided into two based on the "title_status" and the same was applied to the test set as well.
- Breaking down the dataset by "title_status" and fitting separate models resulted in a lower test RMSE than the best model fitted to the complete data set.
- It could be observed that although the models with the lowest test RMSE's were chosen, the test RMSE values were very high.
- The prices in this dataset are completely determined by the owners themselves and has no relation to the organization AUCTIONEXPORT.com. Hence the prices are more likely to be affected with the personal reasons of the owners. We suggest to use a deep learning approach to capture the patterns which couldn't be captured by the models used in this study.
- Imbalance handling techniques were applied to the predictor variable "title_status" to improve the performance of the model, but the results were contradictory to the expectations. This seems to be a consequence of the changes that occurred in the other predictors as the result of imbalance handling.

Appendix

- R was used for Imbalance handling techniques and Ridge, Lasso, Elastic Net Regression models.
- Python was used for other advanced modelling techniques.

Imbalance Handling in R

```
df= read.csv("D:\\University\\Year-4\\ST 4052\\Project 2\\color replaced.csv")
df=df[,-c(1,10,11)]
library(ROSE)
set.seed(10)
train=sample(1:nrow(df), 0.8*nrow(df))
test=(-train)
train_set = df[train,]
test_set = df[test,]
#check classes distribution
prop.table(table(train_set$title_status))
accuracy.meas(train_set$title_status)
library(rpart)
treeimb <- rpart(title_status ~ ., data = train_set)
pred.treeimb <- predict(treeimb, newdata = test_set)
accuracy.meas(test_set$title_status, pred.treeimb[,2])
roc.curve(test_set$title_status, pred.treeimb[,2], plotit = F)
#Over sampled
data_balanced_over <- ovun.sample(title_status ~ ., data = train_set, method =
"over",N = 3700)$data
table(data_balanced_over$title_status)
#both sampled
data_balanced_both <- ovun.sample(title_status ~ ., data = train_set, method =
"both", p=0.5,N=2500, seed = 10)$data
table(data_balanced_both$title_status)
#ROSE
data.rose <- ROSE(title_status ~ ., data = train_set, seed = 10)$data
table(data.rose$title_status)
```

Ridge, Lasso, Elastic Net Models

```
#Ridge Regression
grid=10^seq(10,-2,length =100)
ridge.mod=glmnet(x1,y1,alpha =0,lambda =grid)
plot(ridge.mod,xvar = "lambda",label = T,lw=2)
#cross validation
set.seed(123)
cv.out =cv.glmnet(x1,y1,alpha =0)
plot(cv.out)
bestlam=cv.out$lambda.min
cv.out$lambda.1se
#MSE for ridge
ridge.pred=predict(ridge.mod,s=cv.out$lambda.min,newx=x2)
mean((ridge.pred-y2)^2)
#coefficients of final model
out=glmnet(X,Y,alpha =0)
predict(out,type="coefficients",s=bestlam)[1:209,]
#Lasso Regression
lasso.mod =glmnet(x1,y1,alpha =1,lambda =grid)
plot(lasso.mod,xvar = "lambda",label = T,lw=2)
#cross validaton
```

```
set.seed(123)
cv.out =cv.glmnet(x1,y1,alpha =1)
plot(cv.out)
bestlam1=cv.out$lambda.min
bestlam1
cv.out$lambda.1se
#MSE for Lasso
lasso.pred=predict(lasso.mod,s=cv.out$lambda.1se,newx=x2)
mean((lasso.pred -y2)^2)
#coefficients of final model
out=glmnet(X,Y,alpha =1,lambda =grid)
lasso.coef=predict(out,type ="coefficients",s=bestlam1)[1:209,]
lasso.coef[lasso.coef!=0]
#Elastic net Regression
alph=seq(0.1,0.9,0.05)
en.mse=c()
en.best=c()
for(i in 1:length(alph)){

  en.mod=glmnet(x1,y1,alpha =alph[i],lambda =grid)
  set.seed(123)
  cv.en=cv.glmnet(x1,y1,alpha =alph[i])
  en.best[i]=cv.en$lambda.min
  en.pred=predict(en.mod,s=en.best[i],newx=x2)
  en.mse[i]=mean((en.pred-y2)^2)
}
en_df=data.frame(alph,en.mse,en.best)
en_df
bestlam=en_df$en.best[en.mse==min(en.mse)]
bestlam
#BEST FIT MODEL
out=glmnet(X,Y,alpha =0.1)
en.coef=predict(out,type="coefficients",s=bestlam)[1:209,]
en.coef
en.coef[en.coef!=0]
#en graphs
en.mod1=glmnet(x1,y1,alpha =0.1,lambda =grid)
plot(en.mod1,xvar = "lambda",label = T,lw=2)
set.seed(123)
cv.out =cv.glmnet(x1,y1,alpha =0.1)
plot(cv.out)
```

Random Forest Regression For Clean Titled Vehicles

```
from sklearn.ensemble import RandomForestRegressor
regr = RandomForestRegressor(random_state=10)
regr.fit(X_train_clean, y_train_clean)
#Grid Search for Parameter Tuning
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num =
10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
```

```
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
pprint(random_grid)
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf_clean = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random_clean = RandomizedSearchCV(estimator = rf_clean, param_distributions
= random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs=-1)
# Fit the random search model
rf_random_clean.fit(X_train_clean, y_train_clean)
#Best Parameters
rf_random_clean.best_params_
y_pred_clean=rf_random_clean.predict(X_test_clean)
math.sqrt(np.mean((np.array(y_test_clean)-y_pred_clean)**2))
#Feature Importance Plot
feat_importances = pd.Series(rf_clean.feature_importances_[1:204],
index=df_withDummies_clean.columns[1:204])
feat_importances.nlargest(10).plot(kind='barh')
```

Boosting Techniques for the original data set

```
original_train = pd.read_csv("D:/University/Year-4/ST 4052/Project
2/Balanced/imbalanced_train_set.csv")test_set =
pd.read_csv("D:/University/Year-4/ST 4052/Project 2/Balanced/test_set.csv")
full_data_Original =original_train.append(test_set, ignore_index=True)
# Handling Dummy Variables
Nbrand=pd.get_dummies(full_data_Original.brand)
Nmodel=pd.get_dummies(full_data_Original.model)
Ntitle=pd.get_dummies(full_data_Original.title_status)
Nstate=pd.get_dummies(full_data_Original.state)
Ncolor=pd.get_dummies(full_data_Original.color)
df_withDummies=pd.concat([full_data_Original,Nbrand,Nmodel,Ntitle,Nstate,Ncolor
], axis='columns')
df_withDummies=df_withDummies.drop(['brand','model','title_status','color','sta
te'],axis='columns')
#Train, test split
train = df_withDummies.iloc[0:1959,:]
test = df_withDummies.iloc[1959:2449,:]
X_train=train.drop(['price'],axis='columns')
y_train=train.price
```

```
X_test=test.drop(['price'],axis='columns')
y_test=test.price
#XG Boost
import xgboost as xgb
from sklearn.metrics import mean_squared_error
import pandas as pd
import numpy as np
xg_reg = xgb.XGBRegressor(objective ='reg:linear', colsample_bytree = 0.3,
learning_rate = 0.1,max_depth = 5, alpha = 10, n_estimators = 10)
test_df = train.loc[:,~train.columns.duplicated()]
test_df2 = test.loc[:,~test.columns.duplicated()]
xg_reg.fit(test_df[test_df.columns[1:214]],test_df[test_df.columns[0]])
data_dmatrix =
xgb.DMatrix(data=test_df[test_df.columns[1:214]],label=test_df[test_df.columns[
0]])
preds = xg_reg.predict(test_df2[test_df2.columns[1:214]])
rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE: %f" % (rmse))
# Gradient Boosting
# evaluate gradient boosting ensemble for regression
from numpy import mean
from numpy import std
from sklearn.datasets import make_regression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.ensemble import GradientBoostingRegressor
# define the model
model = GradientBoostingRegressor()
model.fit(X_train, y_train)
# make predictions
yhat = model.predict(X_test)
math.sqrt(np.mean((yhat-y_test)**2))
#Support Vector Regression
from sklearn.svm import SVR
regressor = SVR(kernel = 'linear')
regressor.fit(X_train, y_train)
y_pred_svr = regressor.predict(X_test)
math.sqrt(np.mean((y_pred_svr-y_test)**2))
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train, y_train)
y_pred_svr = regressor.predict(X_test)
math.sqrt(np.mean((y_pred_svr-y_test)**2))
#Same code was repeated for the separate datasets
```