Capstone Project

Yiying Wang

Machine Learning Engineer Nanodegree **September 5^{th}, 2016**

Definition

Project Overview

Auto dealerships who purchase used cars at auto auctions often face the risks of buying in cars with serious issues without being informed, such as, tampered odometers, mechanical issues, vehicles having titles that cannot be transferred, etc. These kinds of cars are called 'kicked' cars. Purchasing a kicked car is very costly for car dealers because they have to pay repair fees before they can resell the vehicles and there is a great chance that they couldn't resell them. To help dealers avoid loss like this, this project is aimed at developing a method that given some information and specifications about a car, it outputs the prediction for whether or not this car is a kicked car. The datasets used in this project are downloaded from Kaggle competition website (https://www.kaggle.com/c/DontGetKicked) which was provided by an online car dealership, Carvana.

Problem Statement

The problem that will be solved in this project is a supervised machine learning, binary classification problem. This is because the datasets contain past transactions of cars purchased from auctions with the outcome of whether or not those cars turned out to be kicked cars. It is binary classification problem since the target variable only contains two possible values: 0 (for good buy) and 1 (for bad buy). The datasets include both categorical data, such as vehicle's manufacturer, vehicle's color, vehicle's transmission type; and numerical data, such as acquisition price for the vehicle in different conditions. The approach for this project will expand categorical variables using one-hot encoding, meaning transforming categorical data into a matrix of 0s and 1s to indicate the category. For example, if, for color attribute, there were three possible colors: green, red and yellow, then this attribute would be transformed into three integers, with (1, 0, 0) indicating green, (0, 1, 0) indicating red and (0, 0, 1) indicating yellow. Besides, the frequencies of a car being a bad buy based on some categorical variables will be calculated and be used as additional features for prediction. Take the color of car as an example again, the frequencies of a green, red and yellow car being a bad buy will be calculated and a new feature called 'Color_Bad_Freq' will be created. The numerical attributes will be used after dropping multicollinearity variables.

After the data being transformed to numerical matrix form, the next step is to split the dataset into training set and validation set. The training set will be used for fitting selected models and validation set will be used to evaluate each model's performance. The training-validation separation is important for the models to have a better generalization to unseen data. The algorithms to be used are Logistic Regression, Neural Network, Random Forest and Gradient Boosted Trees. SVM and K-nearest neighbors (KNN) are excluded from consideration because the dataset has more than 70000 entries and this makes running SVM very slow, also, the dataset has 205 features after the transformation and KNN as a similarity-based machine learning algorithm doesn't perform well in high dimensions due to in high dimensions the examples tend to look alike [1]. For each algorithm mentioned above, hyperparameters will be tweaked to have models better fit the data. The model with the best performance will be chosen to predict the results for test dataset and the result will be submitted to Kaggle competition website for evaluation.

Metrics

Accuracy is not a good model performance indicator for this problem because the target variable is highly skewed. There are about 90% of data belonging to one class (0, a good buy) and 10% of data belonging to the other (1, a bad buy). Therefore, a prediction of all 0s would achieve an accuracy of 90% but it is not useful for the dealers. A more important indicator for this problem is sensitivity (recall) of the model. Sensitivity is also called true positive rate which represents the percent of positive samples that are actually identified as positives by the model. A binary classification model can make two kinds of mistakes, namely, false positive (identify a good car as a kicked car) and false negative (identify a kicked car as a good car). For our problem, the cost of making these two mistakes are different. Car dealerships could lose a lot of money from buying in a kicked car than not buying a good car. However, only focusing on sensitivity could lead to another issue. If increasing sensitivity is the only goal, the model that always predicts positive would be the 'best' model but this model is useless. Therefore, the area under ROC curve (AUC), which reflects a binary classifier's performance for a combination of true positive rate and false positive rate, will be used as the metric.

Analysis

Data Exploration and Exploratory Visualization

RefId	IsBadBuy	PurchDate	Auction	VehYear	VehicleAge	Make	Model	Trim	SubModel	Color	Transmissi	WheelType	WheelType	VehOdo	Nationality	Size	TopThree
1	0	12/7/09	ADESA	2006	3	MAZDA	MAZDA3	i	4D SEDAN	RED	AUTO	1	Alloy	89046	OTHER A	MEDIUM	OTHER
2	0	12/7/09	ADESA	2004	5	DODGE	1500 RAM	ST	QUAD CA	WHITE	AUTO	1	Alloy	93593	AMERICA	LARGE T	CHRYSLE
3	0	12/7/09	ADESA	2005	4	DODGE	STRATUS	SXT	4D SEDAN	MAROON	AUTO	2	Covers	73807	AMERICA	MEDIUM	CHRYSLE
4	0	12/7/09	ADESA	2004	5	DODGE	NEON	SXT	4D SEDAN	SILVER	AUTO	1	Alloy	65617	AMERICA	COMPAC	CHRYSLE
5	0	12/7/09	ADESA	2005	4	FORD	FOCUS	ZX3	2D COUPE	SILVER	MANUAL	2	Covers	69367	AMERICA	COMPAC	FORD
6	0	12/7/09	ADESA	2004	5	MITSUBIS	GALANT	ES	4D SEDAN	WHITE	AUTO	2	Covers	81054	OTHER A	MEDIUM	OTHER
7	0	12/7/09	ADESA	2004	5	KIA	SPECTRA	EX	4D SEDAN	BLACK	AUTO	2	Covers	65328	OTHER A	MEDIUM	OTHER
8	0	12/7/09	ADESA	2005	4	FORD	TAURUS	SE	4D SEDAN	WHITE	AUTO	2	Covers	65805	AMERICA	MEDIUM	FORD
9	0	12/7/09	ADESA	2007	2	KIA	SPECTRA	EX	4D SEDAN	BLACK	AUTO	2	Covers	49921	OTHER A	MEDIUM	OTHER
10	0	12/7/09	ADESA	2007	2	FORD	FIVE HUN	SEL	4D SEDAN	RED	AUTO	1	Alloy	84872	AMERICA	LARGE	FORD
11	0	12/14/09	ADESA	2005	4	GMC	1500 SIER	SLE	REG CAB	SILVER	AUTO	1	Alloy	80080	AMERICA	LARGE T	GM
12	0	12/14/09	ADESA	2001	8	FORD	F150 PICK	XL	REG CAB	WHITE	MANUAL	1	Alloy	75419	AMERICA	LARGE T	FORD
13	1	12/14/09	ADESA	2005	4	DODGE	CARAVAN	SE	MINIVAN	RED	AUTO	1	Alloy	79315	AMERICA	VAN	CHRYSLE
14	0	12/14/09	ADESA	2005	4	NISSAN	ALTIMA	Bas	4D SEDAN	WHITE	AUTO	2	Covers	71254	TOP LINE	MEDIUM	OTHER
15	0	12/14/09	ADESA	2006	3	DODGE	CARAVAN	SXT	MINIVAN	GOLD	AUTO	1	Alloy	74722	AMERICA	VAN	CHRYSLE
16	0	12/14/09	ADESA	2003	6	CHEVROI	CAVALIE	Bas	2D COUPE	WHITE	MANUAL	2	Covers	72132	AMERICA	COMPAC	GM
17	0	12/14/09	ADESA	2005	4	CHEVROI	TRAILBL	LS	4D SUV 4.	WHITE	AUTO	1	Alloy	80736	AMERICA	MEDIUM	GM
18	0	12/14/09	ADESA	2003	6	SATURN	VUE 2WD	4C	4D CUV 2.	RED	AUTO	2	Covers	75156	AMERICA	MEDIUM	GM
19	0	12/14/09	ADESA	2005	4	CHEVROI	IMPALA	Bas	4D SEDAN	GREY	AUTO	2	Covers	65925	AMERICA	LARGE	GM
20	0	12/14/09	ADESA	2004	5	GMC	ENVOY X	SLE	4D UTILI	BLACK	AUTO	1	Alloy	84498	AMERICA	MEDIUM	GM
21	0	12/14/09	ADESA	2002	7	CHRYSLE	VOYAGE	Bas	MINIVAN	BLUE	AUTO	2	Covers	54586	AMERICA	VAN	CHRYSLE
22	0	12/14/09	ADESA	2002	7	CHEVROI	MONTE C	SS	2D COUPE	WHITE	AUTO	1	Alloy	66536	AMERICA	LARGE	GM
23	0	12/21/09	ADESA	2004	5	CHEVROI	VENTURE	LS	PASSENG	SILVER	AUTO	2	Covers	98130	AMERICA	VAN	GM
24	0	12/21/09	ADESA	2006	3	CHEVROI	HHR	LS	4D SUV 2.	GREY	AUTO	2	Covers	59789	AMERICA	MEDIUM	GM
25	0	12/21/09	ADESA	2007	2	CHEVROI	HHR	LS	4D SUV 2.	GREY	AUTO	2	Covers	65663	AMERICA	MEDIUM	GM
26	1	12/21/09	ADESA	2004	5	MERCUR	SABLE	LS	4D SEDAN	WHITE	AUTO	1	Alloy	52106	AMERICA	MEDIUM	FORD
27	0	12/21/09	ADESA	2002	7	GMC	ENVOY X	SLE	4D UTILI	GOLD	AUTO	1	Alloy	88958	AMERICA	MEDIUM	GM
28	0	12/21/09	ADESA	2004	5	DODGE	1500 RAM	ST	QUAD CA	WHITE	AUTO	1	Alloy	76173	AMERICA	LARGE T	CHRYSLE
29	0	12/28/09	ADESA	2004	5	DODGE	DURANG	SLT	4D SUV 5.	RED	AUTO	1	Alloy	65393	AMERICA	MEDIUM	CHRYSLE
30	0	12/28/09	ADESA	2005	4	DODGE	CARAVAN	SXT	MINIVAN	GREY	AUTO	1	Alloy	80064	AMERICA	VAN	CHRYSLE

RefId	MMRAcqu	MMRAcqu	MMRAcqu	MMRAcqu	MMRCurr	MMRCurr	MMRCurr	MMRCurr	PRIMEUN	AUCGUAI	BYRNO	VNZIP1	VNST	VehBCost	IsOnlineSa	WarrantyCost
1	8155	9829	11636	13600	7451	8552	11597	12409	NULL	NULL	21973	33619	FL	7100	0	1113
2	6854	8383	10897	12572	7456	9222	11374	12791	NULL	NULL	19638	33619	FL	7600	0	1053
3	3202	4760	6943	8457	4035	5557	7146	8702	NULL	NULL	19638	33619	FL	4900	0	1389
4	1893	2675	4658	5690	1844	2646	4375	5518	NULL	NULL	19638	33619	FL	4100	0	630
5	3913	5054	7723	8707	3247	4384	6739	7911	NULL	NULL	19638	33619	FL	4000	0	1020
6	3901	4908	6706	8577	4709	5827	8149	9451	NULL	NULL	19638	33619	FL	5600	0	594
7	2966	4038	6240	8496	2980	4115	6230	8603	NULL	NULL	19638	33619	FL	4200	0	533
8	3313	4342	6667	7707	3713	4578	6942	8242	NULL	NULL	19638	33619	FL	4500	0	825
9	6196	7274	9687	10624	6417	7371	9637	10778	NULL	NULL	21973	33619	FL	5600	0	482
10	7845	9752	11734	13656	9167	10988	12580	14845	NULL	NULL	21973	33619	FL	7700	0	1633
11	5243	6627	8848	10458	5712	7552	9494	11663	NULL	NULL	5546	33619	FL	5500	0	1373
12	3168	4320	5826	6762	2871	3822	5734	6559	NULL	NULL	5546	33619	FL	5300	0	869
13	4225	5380	7910	10108	4807	6077	8523	10124	NULL	NULL	19638	33619	FL	5400	0	1623
14	5684	6921	8976	10789	5976	7190	9374	10780	NULL	NULL	19638	33619	FL	7800	0	686
15	6999	9730	10743	14066	7952	9910	12131	13836	NULL	NULL	19638	33619	FL	6900	0	1623
16	2228	3217	5256	7589	2128	2690	4855	5506	NULL	NULL	5546	33619	FL	3300	0	1455
17	6742	7951	11068	12885	7670	9159	12596	15041	NULL	NULL	19638	33619	FL	6800	0	1243
18	2756	3475	5835	6925	3513	4572	7195	9364	NULL	NULL	5546	33619	FL	4900	0	1923
19	3983	5244	7316	9194	4107	5191	7638	8800	NULL	NULL	19638	33619	FL	5700	0	1703
20	7347	8845	11092	13052	7436	8766	11367	12609	NULL	NULL	19638	33619	FL	7100	0	1243
21	2139	3106	5072	6424	3307	4388	6228	7413	NULL	NULL	19638	33619	FL	4800	0	1551
22	3522	4791	7689	10460	4403	5629	7512	8975	NULL	NULL	5546	33619	FL	5800	0	2003
23	2290	3818	5646	7427	2446	3542	5999	8687	NULL	NULL	5546	33619	FL	4100	0	5613
24	5693	6842	9586	11299	6062	7214	9721	11606	NULL	NULL	5546	33619	FL	7700	0	671
25	5830	6923	9644	10838	7038	7945	11189	11803	NULL	NULL	5546	33619	FL	7000	0	728
26	2619	3561	6502	7274	3160	4376	6600	8074	NULL	NULL	5546	33619	FL	4500	0	754
27	5367	6517	9005	10050	5510	6838	8522	10252	NULL	NULL	5546	33619	FL	8000	0	2452
28	6407	8051	10378	12443	7159	8934	11135	12560	NULL	NULL	19638	33619	FL	8800	0	920
29	7243	8582	11805	13250	7863	10109	12295	15807	NULL	NULL	19638	33619	FL	8400	0	1411
30	5439	7255	9587	11288	6175	7792	9768	11489	NULL	NULL	19638	33619	FL	5600	0	1763

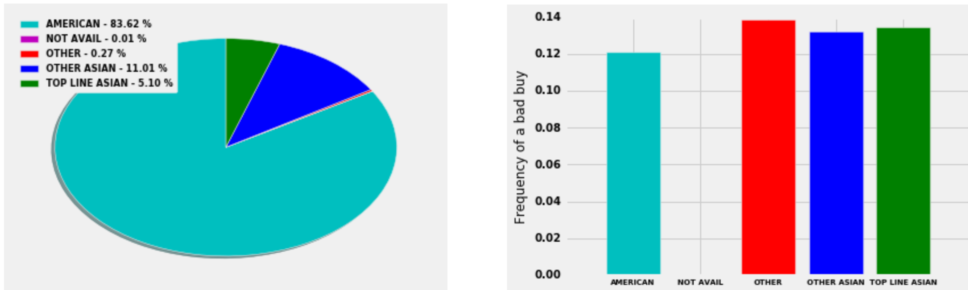
Figure 1. A sample of training dataset.

The raw training data downloaded from Kaggle is in the format of .csv files. It is a table with 72983 rows, each representing a past purchase, and 34 columns, of which 33 columns contain descriptive information for the purchase and 1 column ('IsBadBuy') being the binary result for whether it is a kicked car or not. A snippet of dataset is shown in figure 1 and a detailed description for all the columns is shown in Table 1. After a brief inspection of the data columns' description, four columns are selected to be dropped from analysis. 'RefId' and 'BYRNO' will be dropped because they are unique identifiers for the car and the buyer and they are not useful for prediction. 'VehYear' and 'WheelTypeID' will be dropped because they hold the same information with 'VehicleAge' and 'WheelType'. The remaining features will be roughly divided into two groups: categorical and numerical features and they will be inspected more in detail below. Later in the project, this training data file will be divided into training set and validation set for model fitting and selection. Kaggle also provides a separate file name 'test.csv'. This data will be used for evaluating the performance of the final model.

Table 1. Descriptions and data types for each column of the dataset

Field name	Description	Type
IsBadBuy	Identifies if the car is a kicked car	Target, Binary
RefId	Unique sequential number assigned to vehicles	To drop
BYRNO	Unique number assigned to the buyer	
VehYear	Manufacturer's year	
WheelTypeID	The type id of the vehicle wheel	
Nationality	The manufacturer's country	Categorical
Make	Manufacturer of the car	
Model	Model of the car	
Trim	Vehicle trim level	
SubModel	Vehicle submodel	
Color	Vehicle color	
Transmission	Vehicles transmission type	
VehicleAge	Years elapsed since the manufacturer's year	
WheelType	The vehicle wheel type description	
Size	The size category of the vehicle	
TopThreeAmericanName	Identifies if the manufacturer is top three	
PRIMEUNIT	If the vehicle has a higher demand	
AUCGUART	The level guarantee provided by auction	
VNZIP1	Zipcode where the car was purchased	
VNST	State where the car was purchased	
IsOnlineSale	If the vehicle was purchased online	
PurchDate	Date of the purchase	
Auction	Auction provider of the purchase	
MMRAcquisitionAuctionAveragePrice **MMRAcquisitionAuctionCleanPrice** **MMRAcquisitionRetailAveragePrice** **MMRAcquisitonRetailCleanPrice** **MMRCurrentAuctionAveragePrice** **MMRCurrentAuctionCleanPrice** **MMRCurrentRetailAveragePrice** **MMRCurrentRetailCleanPrice**	Prices for this vehicle in different conditions: average condition, above average condition, in the retail market in average condition, in the retail market in above average condition, in average condition as of current day, in the above condition as of current day, in the retail market in average condition as of current day, in the retail market in above average condition as of current day.	Numerical
VehBCost	Acquisition cost paid for the vehicle	
WarrantyCost	Warranty price	
VehOdo	The vehicles odometer reading	

Categorical Features



a **b**

Figure 2. Nationality of the cars in dataset.

The 'Nationality' column listed each car's manufacturer country. The majority (83.62%) of them are American cars and 16.11% of them are Asian cars of which 5.1% are classified as 'Top Line Asian'. All the other countries are grouped together called 'Other' (Figure 2, a). There are 5 samples in the dataset that miss values for this column and they are listed as 'Not Avail'. To further understand the impact of nationality on the likelihood of a car being kicked, a bar graph (Figure 2, b) showing the frequency of a bad buy grouped by nationality. The 'Not Avail' frequency is 0 mostly because there are only 5 samples in that group and they happen to be all good buys. As for other groups, the frequencies are fairly close to one another with the lowest as 0.121 from 'American' and the highest as 0.138 from 'Other'. Thus the frequency of being a bad buy based on nationality doesn't seem to be very informative.

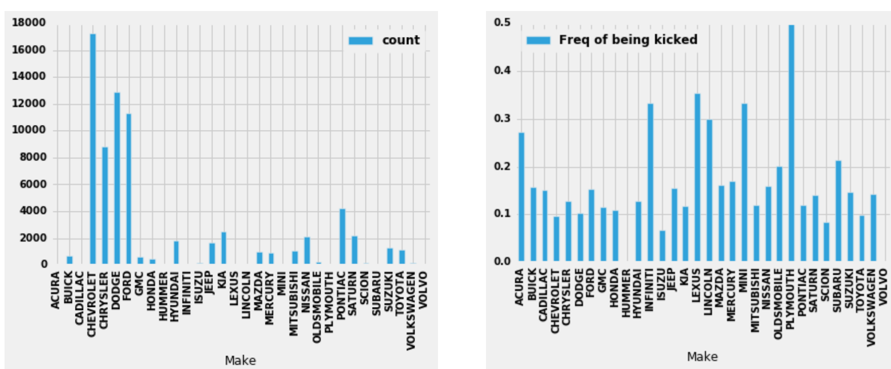


Figure 3. Count and frequency of being kicked of the cars based on their Manufacturer.

The 'Make' column has 32 unique values representing the manufacturer for cars in the dataset. The distribution of cars among different manufacturers is unbalanced with 'Chevrolet' being the most popular brand having 17248 records and several other brands just having a few records, e.g. 'Hummer', 'Plymouth'. This unbalanced distribution makes some of the frequencies based on manufacturer biased. For example, 'Hummer' has a 0% of being kicked but it mostly because there is only one Hummer car in the dataset and it happens to be a good buy. On the other hand, 'Plymouth' has the highest percent of being kicked but we cannot conclude that Plymouth cars are more likely to be kicked because the sample size is 2 and one of them is a kicked. Thus adjustments need to be made on the Make_Bad_Frequency table. The frequencies calculated from dataset (shown in figure 3) will be remained only if the count for that brand is larger than the median count, 649. For the manufacturers that have less than 1000 samples in dataset, their frequencies will be set to the median frequency, which is 0.14.

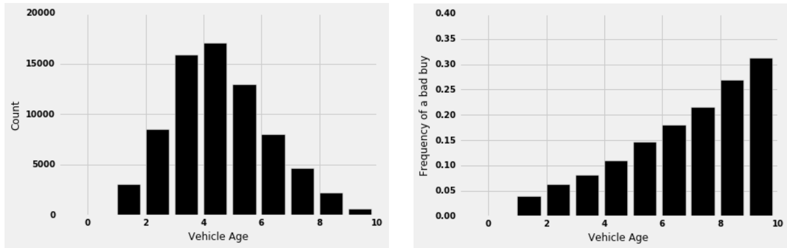


Figure 4. Count and frequency of being kicked cars based on their age.

A similar inspection is performed on the vehicle age. All the vehicles in the dataset have ages ranging from 0 year to 9 years and the distribution is bell-shaped with the majority of the vehicles have the middle ages and the newest and oldest cars are minorities. After plotting a bar graph for frequency of being kicked based on vehicle age, a strong correlation is observed. The frequency of being a bad buy monotonously increases as the vehicle age increases. It agrees with our intuition that when a car getting older, it tends to have more hidden mechanical issues and thus it is more likely to be a car sold overvalued at auction. Therefore, frequency of a bad buy based on vehicle age would possibly be a strong indicator for prediction and this will be included as an additional feature for future analysis. Except above three examples, other categorical variables are explored by a similar fashion to determine if frequency of a bad buy would be included in model building. The detailed description of categorical features treatment will be discussed later in 'Methodology – Data Preprocessing' section.

Numerical Features

There are, in total, 11 numerical features given in the dataset. Of which, 8 features are average vehicle prices for that car in different conditions (average or clean), in different time (the time of purchase or current), and in different purchasing places (auction or market). There are missing values for these features. It is, however, not a big concern here since the missing value transactions only account for 0.43% of all data points. Pairwise scatter plots for these 8 features has revealed that the prices for average cars and clean cars are highly correlated as shown in figure 5. To avoid this multicollinearity issue, the four clean prices features will be removed.

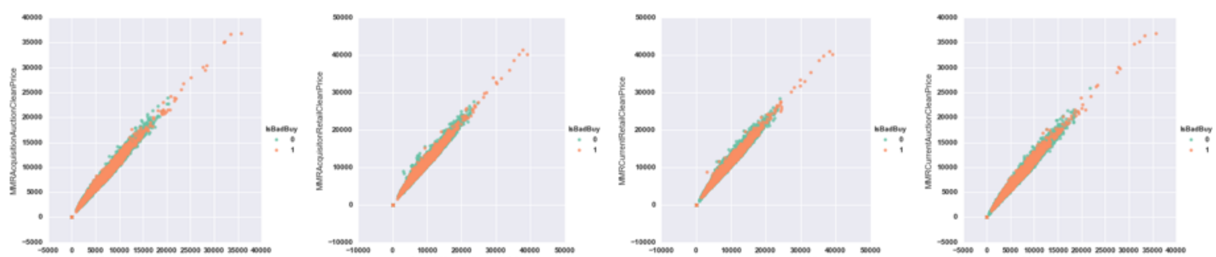


Figure 5. Average prices and clean prices are highly correlated for all conditions.

Figure 6 shows pairwise scatter plots for five prices, namely, acquisition auction average price, acquisition retail average price, current auction average price, current retail average price and vehicle cost. The plots are color coded with green indicating good buy and orange indicating bad buy. From these plots we can observe that these five prices are not very correlated. Two classes have slightly different price distributions and there are several points from the bad buy class spread out at spaces far away from the majority of points. However, they will not be removed from analysis as outliers because this characteristic might be an indication for being a bad buy.

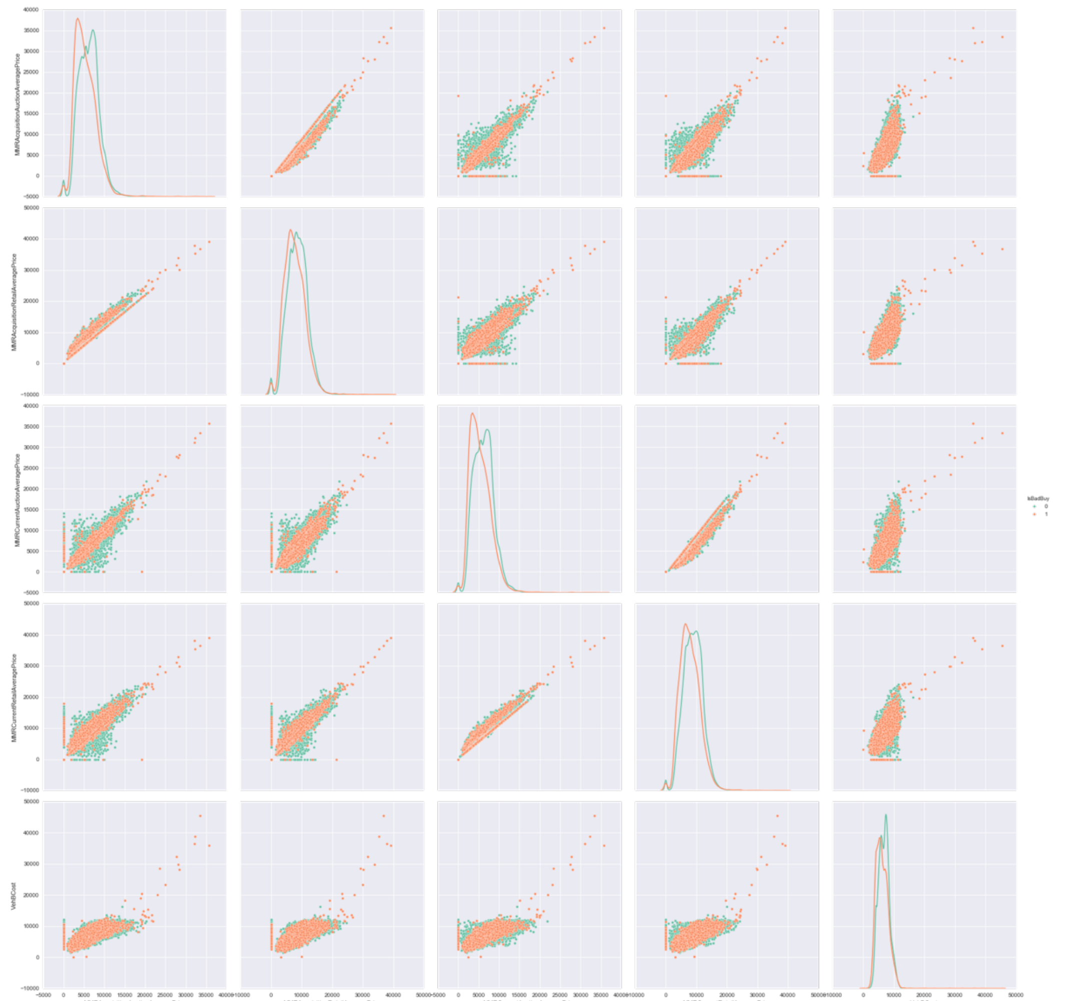


Figure 6. Pairwise scatter plot and kernel density plot for prices at different conditions.

Algorithms and Techniques

The machine learning algorithms to be applied in this project are Logistic Regression, Neural Network, Random Forest and Gradient Boosted Trees. These four algorithms are chosen because they are able to model a binary classifier based on large, complicated input variables which mixed with both categorical and numerical values.

Logistic regression is a natural first thought when dealing with binary classification problem because it can model the probability of samples belonging to one category and the other by linearly relating the feature matrix with the predicted log-odds. Then with a decision boundary, probabilities can be converted to class names. Parameters for the linear function will be optimized with data but some hyperparameters are subject to change, including 'penalty', 'C' and 'class_weight'. 'Penalty' and 'C' represent the penalty term in the loss function for regularization. Hyperparameter 'penalty' decides which norm to use: L1 (LASSO) or L2 (Ridge). 'C' decides the strength of regularization term. We can use these knobs to reduce overfitting. 'Class_weight' is a hyperparameter used to give different weights to different classes and it is important in the cases that classes are skewed.

Neural Network, unlike logistic regression, is a non-linear model. It is able to capture the non-linear effects that input data has on the response variable. This project will construct a neural network with 3 hidden layers. The input data will be taken a weighted sum to each node in the first hidden layer. Then this newly created values in hidden layer 1 nodes will run through an activation function and the result will be the taken a weighted sum to nodes in the next layer. After running through 3 hidden layers, results are taken a weighted sum again to produce output layer. In this case, output layer will be two nodes, each indicating a class. With number of layers determined, another hyperparameter that can be tuned is the number of nodes in each layer. More nodes will increase the complexity of the model and help capture the variances in the data but too many nodes might cause overfitting problem.

Random Forest is a large collection of decorrelated decision trees and it makes predictions by taking a majority vote from these decision trees. This explains the word 'forest' in its name. The other word 'random' is the mechanism of de-correlating the trees which is randomly bagging both samples and features when creating decision trees. Thus, each decision tree is created with a subset of data and a subset of variables. The main idea of random forest is to assume that most of the decision trees can make the correct prediction and they make mistakes in different places. By doing this, the noises and variances from each individual trees are averaged and the combined model would have low variance and a higher classification accuracy. The hyperparameters that will be tweaked in this project include the number of trees, max_features and min_samples_split. 'Class_weight" will be set to "balanced" for all models.

Gradient Boosted Trees is also a method of a combination of trees. It is different from random forest in the way that each tree is built to learn from the error of previous trees and it iteratively reweights training examples based on errors. Random forest tends to grow deep trees while gradient boosted trees tend to be shallow. It is applicable for heterogeneous data, especially works good for data in different scales and it can automatically detect non-linear feature interactions. However, it has the disadvantages including requires careful tuning, slow to train and it cannot extrapolate. The hyperparameters to be tuned include loss function, the number of trees, learning rate, max_depth and min_samples_split.

Benchmark

Suppose a car dealership didn't use any method to detect kicked cars from auction, they would assume all cars are good buys and this assumption gives an accuracy of about 90% and an AUC of 0.5. Thus a prediction of all samples belonging to class 0 which has an AUC value of 0.5 is used as benchmark.

Methodology

Data Preprocessing

A summary of treatments for categorical data is shown in table 2. The decisions were made based on the data exploration mentioned before. Most of the categorical variables are treated by both one-hot encoding and creating an additional frequency feature. Some variables, such as 'Model', 'SubModel', are treated with frequency features only because they contain too many categories and converting to one-hot would dramatically increase the dimensionality. Some variables, such as 'Nationality' and 'IsOnlineSale', are treated only with one-hot encoding because it is observed that the frequencies for different categories are all about the same. As for numerical variables, four prices attributes for clean conditions were dropped from feature matrix because they were found to be highly correlated with prices for average conditions. The remaining numerical variables were standardized to make all features centered around zero and have variance in the same order [2]. A detailed description of complete data preprocessing is shown in figure 7.

Table 2. Preprocessing treatments for categorical data

Field Name	Treatments
Nationality	5 categories, one-hot encoding
Make	32 categories, add frequency feature, set count < 629 category's frequency to 0.14, one-hot
Model	1063 categories, add frequency feature, set count < 8 category's frequency to 0.12
Trim	135 categories, add frequency feature, set count < 36 category's frequency to 0.15
SubModel	864 categories, add frequency feature, set count < 7 category's frequency to 0.1
Color	17 categories, add frequency feature, one-hot
Transmission	3 categories, add frequency feature, one-hot
VehicleAge	10 categories, add frequency feature, one-hot
WheelType	4 categories, add frequency feature, one-hot
Size	13 categories, add frequency feature, one-hot
TopThreeAmericanName	5 categories, add frequency feature, one-hot
PRIMEUNIT	3 categories, add frequency feature, one-hot
AUCGUART	3 categories, add frequency feature, one-hot
VNZIP1	153 categories, add frequency feature, set count < 176 category's frequency to 0.11
VNST	38 categories, add frequency feature, one-hot
IsOnlineSale	2 categories, one-hot
PurchDate	Extract month and date from the date string, add frequencies for month and day, one hot for both
Auction	3 categories, add frequency feature, one-hot

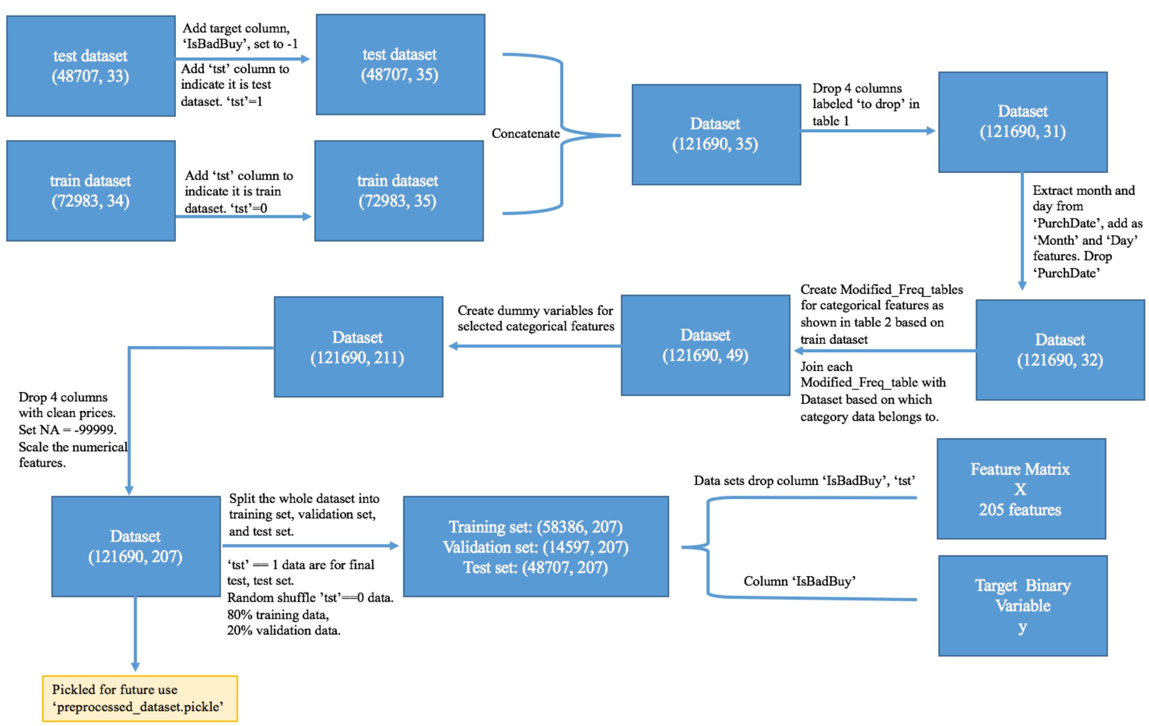


Figure 7. Data Preprocessing flowchart.

Implementation

Logistic Regression, Random Forest and Gradient Boosted Trees algorithms were implemented by directly applying scikit-learn machine learning package in Python. The models were trained using .fit method and the prediction probabilities were obtained using .predict_proba method. After completing the data processing step, the implementation of these three algorithms are fairly easy and out-of-the-box. The complications in the coding process mainly occurred in the implementation of neural network since scikit-learn 0.17 version does not provide a neural network classifier for direct use.

In order to implement neural network classifier in python, an open source software library for machine intelligence developed by Google – TensorFlow – was applied. TensorFlow provides a flexible environment for developing deep neural network architectures with a large-scale computation ability. Even though the tutorials for TensorFlow are well-written, it does take some efforts to understand how TensorFlow describes mathematical computation with data flow graphs in which nodes representing operations and edges representing multi-dimensional arrays (tensors). Building an algorithm from scratch brings more freedom and power but it is also very challenging compared to directly using scikit-learn package. Luckily, TensorFlow has already implemented a handful of useful functions such as various cost functions, optimizers and other common utility functions which simplifies the network building process as choosing and combining the correct pieces for a jigsaw puzzle. TensorFlow is able to construct very deep neural networks, convolutional neural network and recurrent neural network to solve complicated problems, such as image recognition and text analysis tasks. This project, however, used TensorFlow to build a three-layer traditional neural network. First, the three-layer structure neural network was constructed in which sigmoid function was applied as activation function. In the training process, corss_engropy_with_logits function was used as the cost function and AdamOptimizer was used to minimize the cost function.

For all these models, the AUC metrics was performed using scikit-learn metrics module. The false positive rate and true positive rate were obtained using roc_curve function and then an AUC plot was created from these values. The AUC score can be directly calculated using auc function. Cross validation method is used to tune the hyperparameters for the algorithms implemented in scikit-learn.

Refinement

Two hyperparameters, ‘penalty’ and ‘C’, were adjusted for logistic regression algorithm. Parameter ‘penalty’ has two options: L1 norm (least absolute errors) and L2 norm (least squares). Since L2 norm squares the error, it is more sensitive to outliers in the examples which leads a model less robust than using L1 norm. ‘C’ is a parameter that controls the regularization strength. The default is 1 in scikit-learn and four values (0.1, 1, 10, 100) were used to create different models. Models were trained on the training dataset and models’ performance were evaluated on the validation dataset. There are 8 models tested with hyperparameter sets (penalty, C) as (L1, 0.1), (L1, 1), (L1, 10), (L1, 100), (L2, 0.1), (L2, 1), (L2, 10) and (L2, 100). The validation AUC arranges from 0.767 to 0.786. For both penalty norms, the models with default C value (C = 1) have the highest AUC and since L1 norm models is more robust than L2 norm model, the logistic regression model is chosen to be the one with penalty being L1 norm and C being 1. The AUC curves for logistic regression model before and after tuning are shown in figure 8.

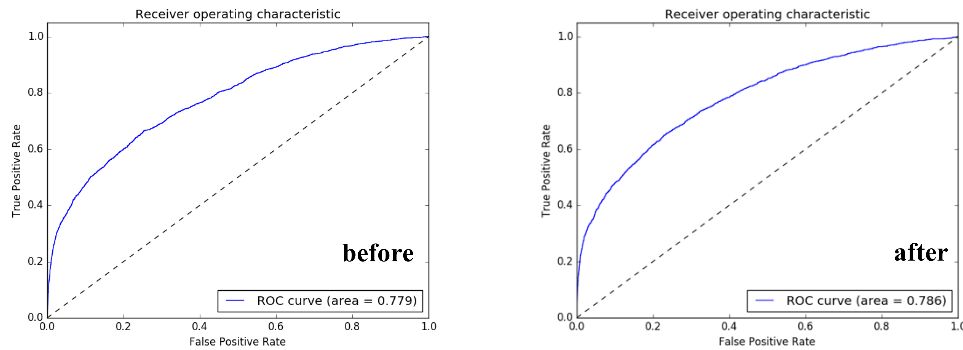


Figure 8. ROC curves for logistic regression model before and after refinement.

For the three-hidden-layer neural network models, there are three hyperparameters to choose: layer 1 node number, layer 2 node number and layer 3 node number. A variety of combinations have been tested, from as low as (5, 5, 5) to as high as (30, 30, 30). It is observed that the optimal choice for node numbers is (14, 13, 10) with an AUC of 0.766 compared to 0.736 for (5, 5, 5) and 0.672 for (30, 30, 30). This is possibly because the models suffer from either underfitting or overfitting for low node number models and high node number models. The neural network model parameters were trained by applying the feed forward and backpropagation cycle multiple times using training dataset and each cycle is called an epoch. To decide a best epoch number, 100 iterations were performed to record AUC value for training data and validation data as iteration increases, shown in figure 9 a. The training AUC is higher than validation AUC overall and the gap between them is increasing indicating overfitting after 60 iterations. Therefore, epoch number is chosen to be 60 for (14, 13, 10) model. Figure 9 b and c show the comparison between ROC curve after refinement and before refinement.

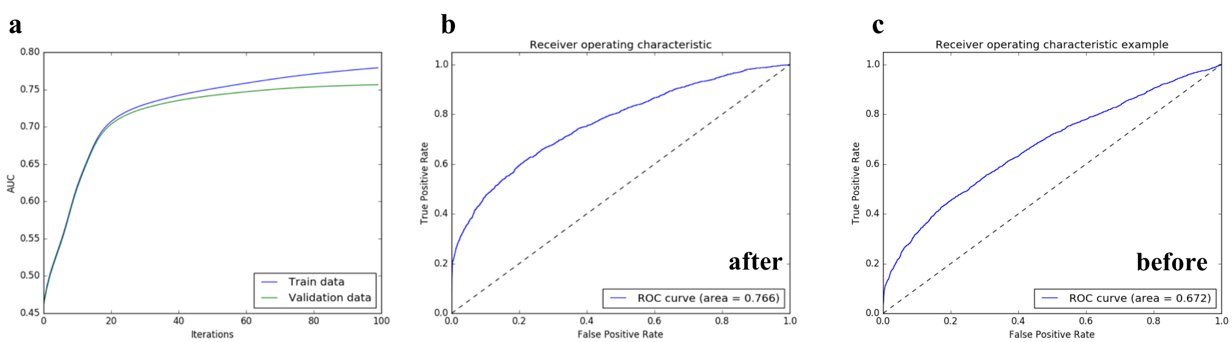


Figure 9. Three-layer neural network. a) training data AUC and validation data AUC as model with nodes number as (14, 13, 10) trained on different iterations of training data. b) the final neural network model's ROC curve. c) the ROC curve for the model before refinement.

For the random forest models, an initial model was built on the default hyperparameter values, namely number of trees is 10, max_features is 'auto' (square root of feature numbers) and min_samples_split is 2. The AUC for this model is 0.710. To search for an optimal parameter combination, a list of values for each parameter was passed to cross validation method with the metric for evaluating model being AUC score. The options for number of trees include 5, 10, 20, 30 and 50; the options for max_features include 'auto' and 'log2'; the options for min_samples_split include 2 and 4. After running a cross validation grid search, the model with the highest AUC (0.749) has parameters as number of trees is 20, max_features is 'auto', min_samples_split is 4. These parameters will be used for the final model of random forest. ROC curves for before and after refinement are shown in figure 10.

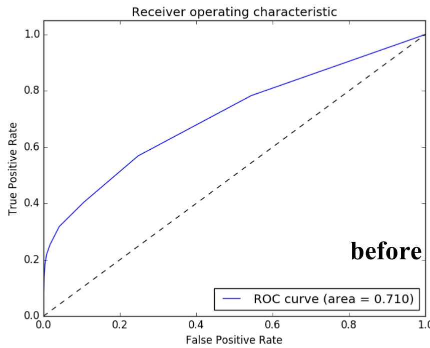


Figure 10. ROC curve for random forest model before and after refinement.

For the gradient boosted regression trees classifier, the hyperparameters to be tuned are loss function, the number of trees, learning rate, max_depth and min_samples_split. The first model trained has the default parameters, namely loss function is 'deviance', number of trees is 100, learning rate is 0.1, max_depth is 3 and min_samples_split is 2. The validation AUC for this model is 0.781. A variety of parameter values were tested using grid search. The options for loss function include 'deviance' and 'exponential'; the options for number of trees include 100, 300, 500 and 1000; the options for learning rate is 0.1, 1 and 10; the options for max_depth is 3, 5 and 7; the options for min_samples_split is 2 and 4. It is noticed that increasing learning_rate reduced AUC to a great deal. For example, with other parameters being the same, the model with learning_rate of 10 has AUC as low as 0.555, compared to the model with learning_rate of 0.1 has AUC as 0.779. The highest AUC among all models is 0.787 with hyperparameters being loss function is 'deviance', number of trees is 100, learning_rate is 0.1, max_depth is 5 and min_samples_split is 4. And this is selected as the gradient boosted trees model. Figure 11 shows the ROC curves for models before and after refinement. The default hyperparameter model has a fairly decent performance already but the one after refinement is chosen for final model.

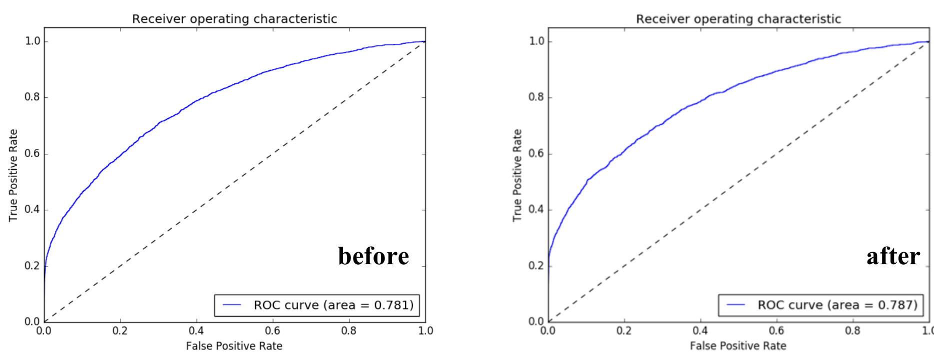


Figure 11. ROC curve for gradient boosted trees model before and after refinement.

Results

Model Evaluation and Validation

From the analysis above, gradient boosted trees model with the highest AUC is chosen to be the final model for this project. This model has 100 trees, uses 'deviance' loss function, has a learning rate of 0.1, maximum depth of 5 and minimum samples split of 4. To evaluate the robustness of this model, five individual runs were performed. For each run, the training dataset was assigned a unique random seed and randomly divided into two parts, one for training (80% of data) and the other for validation (20% of data). The model was trained on the training data and performance evaluated on the validation data and the AUC scores for five runs are summarized in table 3. All the AUC scores are close to or higher than 0.78, thus the robustness of model is confirmed.

Table 3. Model evaluation for five different runs.

Run	1	2	3	4	5
Random State for data splitting	1	2	3	4	5
AUC	0.774	0.789	0.792	0.791	0.786

Justification

As evaluated above, the average AUC for five individual runs is 0.786 which is a great improvement from the benchmark AUC 0.5. One way to interpret AUC is that an AUC of 0.786 means our model would predict a randomly chosen kicked car is more likely to be a bad buy than a randomly chosen non-kicked car with a probability of 78.6%. It is equivalent to say that given two randomly chosen cars, one kicked car and one non-kicked car, our model has a probability of 78.6% to successfully identify the kicked car, as compared to the flipping a coin guessing. To further justify the usefulness of this model, the prediction result for testing dataset was submitted to the Kaggle website for evaluation. Kaggle used Gini score as evaluation metrics for this problem. Gini is positively correlated to AUC and a Gini of 0 is equivalent to a random baseline. The prediction result obtained from this model has a Gini score of 0.23534. Even though this model did not beat the top one team which has a Gini score of 0.26720, it has a much improved Gini score of the benchmark (0.02358) and thus its usefulness is validated.

Conclusion

Free-Form Visualization

This project came up with a model that can predict the probability of a car sold in auction being a kicked car. As shown in figure 12, the majority of the cars are less likely to be kicked cars with probabilities being less than 0.4, while the model is also able to identify the few cars that are likely to be bad purchases. With this model in hand, car dealerships can have an estimate of risk for buying a second hand car by simply inputting the information and specifications of the car into the model. This useful model could potentially help them avoid financial losses and troubles. One difficult aspect for this problem is that the cut-off threshold for being a kicked car is somewhat subjective. Car dealers could have a probability of a car being a bad buy output from our model, but it is their judgement call as to buy the car or not when a probability is in the middle land, such as, 0.4-0.6.

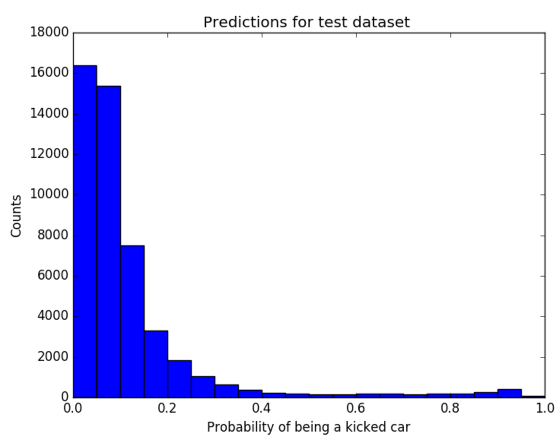


Figure 12. Histogram for test dataset predictions for being a kicked car.

Reflection

To solve the problem of predicting whether a car is a kicked car from an auction, this project undergoes the following steps:

1. Collecting relevant data and compiled the data into a well-formatted file (provided by Kaggle)
2. Exploring the data characteristics by visualization and making decisions for the usage of data
3. Preprocessing data, including convert categorical data to one-hot encoding, dropping extra features and scaling numerical data
4. Determining a benchmark for the classifier and proper metrics used for model evaluation
5. Implementing various machine learning algorithms to build the classifiers
6. Tuning hyperparameters for model selection
7. Evaluating the performance of the final model

In this process, I found the data wrangling part, steps 2-3, was difficult. This is because data exploration is a creative endeavor as there are various possible ways to investigate the data and it usually takes a lot of attempts before interesting information can be found and retrieved. Also, there are many subjective decisions need to be carefully made when it comes to how to deal with missing data or other abnormal data points. Another part I found difficult is to use TensorFlow to build a neural network piece by piece, instead of running machine learning algorithms from scikit-learn out of the box. However, this process helps me understand the algorithm more deeply and with this experience I appreciate more for the hard work and contributions of package developers to open source software libraries.

Improvement

The prediction results might be improved if we consider combining multiple algorithms and create a weighted ensemble model. For example, combining logistic regression, neural network and gradient boosted trees model and assigning different weights to these individual models. The final prediction is a decision made by their combined efforts. It is possible that different models would make mistakes at different places and majority of the models would predict correctly. In this way, the final model would be more robust and less likely to be influenced by flaws of any single model.

References:

[1] Pedro Domingos. A few useful things to know about machine learning. University of Washington.

[2] http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler