Machine Learning Engineer Nanodegree

Capstone Project

Wenbo Ma

10/21/2016

1. Definition

Project Overview

In this project, I will investigate dataset from Kaggle Competition: Predicting Red Hat Business Value

Like many other companies, Red Hat is able to gather a great deal of information over time about the behavior of individuals who interact with them. This information contains characteristics of Red Hat's customers and their activities. Based on these characteristics and activities, I will build binary classification models to predict whether an individual has potential business value for Red Hat. The outcome will be a binary variable taking value on 0 or 1.

Three dataset are provided by Red Hat as follows:

Name	Size	Description
people.csv	3.22MB	Customer's Characteristics
act_train.csv	17.07MB	Activities and Outcome – Training Set
act_test.csv	4.03MB	Activities - Test Set

Link to the Kaggle competition: https://www.kaggle.com/c/predicting-red-hat-business-value

Problem Statement

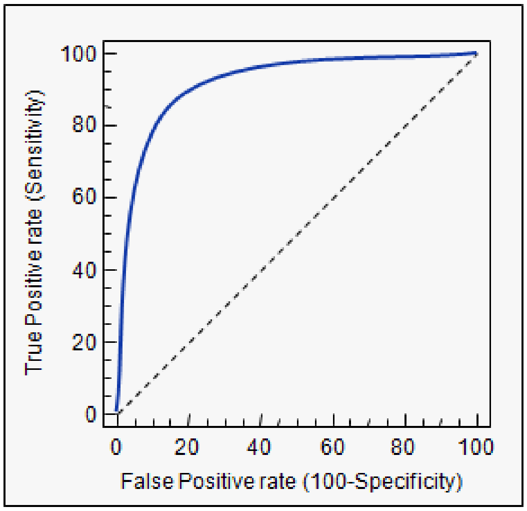
This is a binary classification problem. The inputs will be all the columns in the act_train.csv and the people.csv file (excluding the outcome column) and the output class will be in the 'outcome' column. Machine learning algorithms that specific to classification can be the solution such as decision tree, logistics regression, neural networks and ensemble learning.

Steps to solve this problem can be summarized as follows:

- Download the dataset and understand their structures and contents
- Figure out what are the dependent and independent variables and how to use and combine the three dataset
- Data preprocessing and exploratory analysis.
- Train a simple classifier and use it as an benchmark model
- Improve the benchmark model by feature engineering and parameters tuning
- Train the final model on entire training set and make prediction

Metrics

Area under ROC Curves (AUC) is used in this project for model comparison. Sample ROC plot is as follows:



Source: https://www.medcalc.org/manual/roc-curves.php

Simple Explanation to Confusion Matrix and True Positive Rate and False Positive Rate

Confusion Matrix		True Class	
		Positive	Negative
Predicted Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

True Positive Rate = TP/ (TP+FN)

False Positive Rate = FP/ (FP+FN)

ROC curve can be generated as follows:

- Set various thresholds for a fitted classifier in order to get corresponding pair of values for (False Positive Rate, True Positive Rate)
- Plot the (FPR,TPR) in the graph and connect them into a curve

AUC score is the area under the curve and it is usually between 0.5 and 1. The higher the AUC score is, the better the classifier is.

ROC and AUC is used here as ROC (AUC) is insensitive to imbalanced distribution of the binary outcome class and therefore the model performance can be trusted and generalized well in unseen data.

2. Analysis

Data Exploration

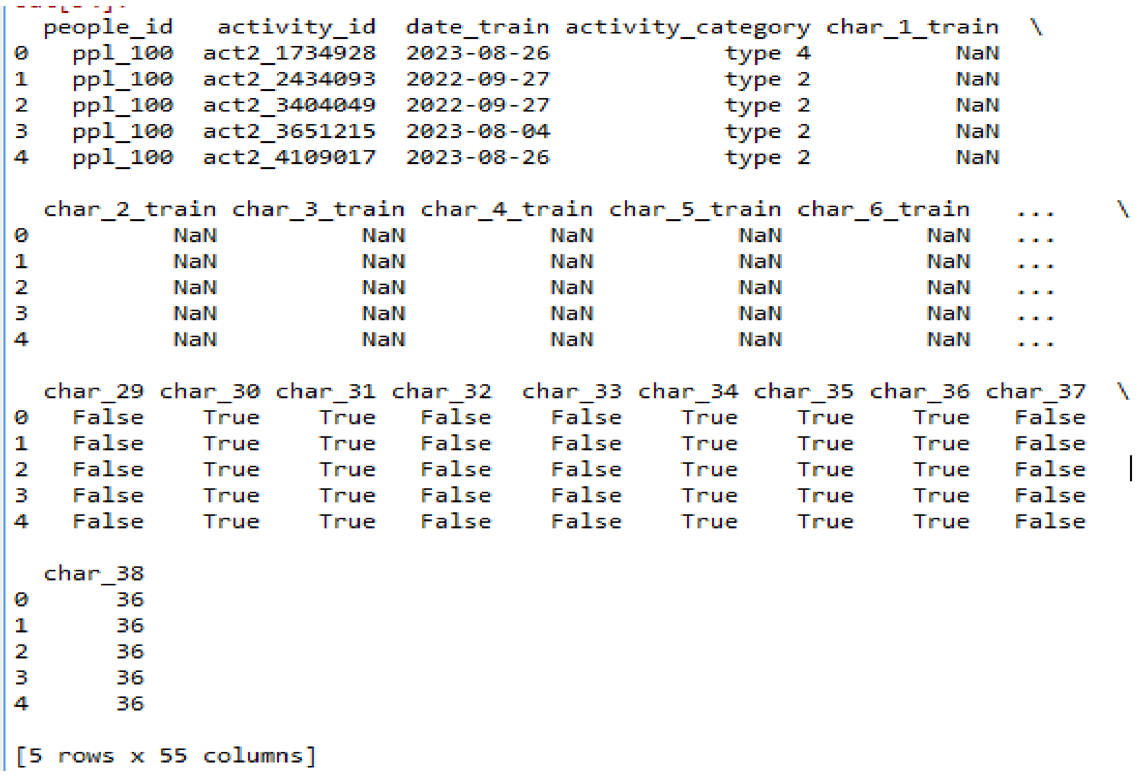
Dataset Name	Number of Samples	Number of Columns	Types of Features	Note
people.csv	189118	41	Categorical, Binary, Numeric	
act_train.csv	2197291	15	Categorical, Numeric	One column is the outcome which is binary
act_test.csv	498687	14	Categorical	

There is a common feature named people_id in both people.csv and act_train.csv. This field is used as a key to merge people.csv and act_train.csv. This process also be applied to act_test.csv. After this preprocessing, the new data is as follows:

Dataset Name	Number of Samples	Number of Features	Types of Features	Note
Training Set	2197291	54	categorical, numerical	
Test Set	498687	54	categorical, numerical	

An important characteristic of the dataset is that only one feature is numerical and all the rest are categorical including binary.

Sample Data (first 5 rows of the original training set)



```
  people_id   activity_id  date_train activity_category char_1_train  \
0   ppl_100  act2_1734928  2023-08-26            type 4          NaN
1   ppl_100  act2_2434093  2022-09-27            type 2          NaN
2   ppl_100  act2_3404049  2022-09-27            type 2          NaN
3   ppl_100  act2_3651215  2023-08-04            type 2          NaN
4   ppl_100  act2_4109017  2023-08-26            type 2          NaN

  char_2_train char_3_train char_4_train char_5_train char_6_train   ...    \
0          NaN          NaN          NaN          NaN          NaN   ...
1          NaN          NaN          NaN          NaN          NaN   ...
2          NaN          NaN          NaN          NaN          NaN   ...
3          NaN          NaN          NaN          NaN          NaN   ...
4          NaN          NaN          NaN          NaN          NaN   ...

  char_29 char_30 char_31 char_32  char_33 char_34 char_35 char_36 char_37  \
0   False    True    True   False    False    True    True    True   False
1   False    True    True   False    False    True    True    True   False
2   False    True    True   False    False    True    True    True   False
3   False    True    True   False    False    True    True    True   False
4   False    True    True   False    False    True    True    True   False

  char_38
0      36
1      36
2      36
3      36
4      36

[5 rows x 55 columns]
```

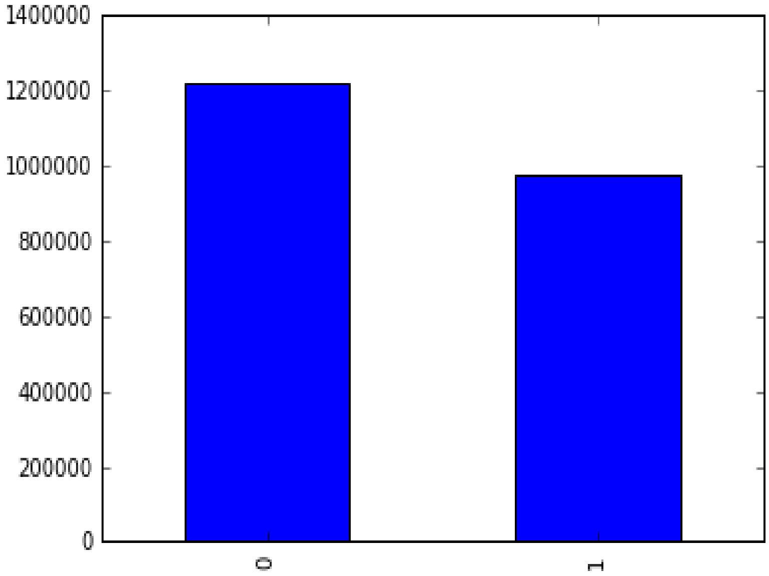
Following in red is added after the second review

Per the previous reviews, I understand that I was required to split the training set into a new small training set and test set as I don't have the access to the outcome on the real test set on Kaggle nor perform analysis on the test result (e.g. confusion matrix).

Therefore, I split the original training set into a new training set and test set by the proportion 4:1. This proportion is selected as it is the proportion between the original training set and test set. Furthermore, stratified sampling by the feature 'outcome' is performed for this split as it preserves the distribution of the feature 'outcome' in the original training set.

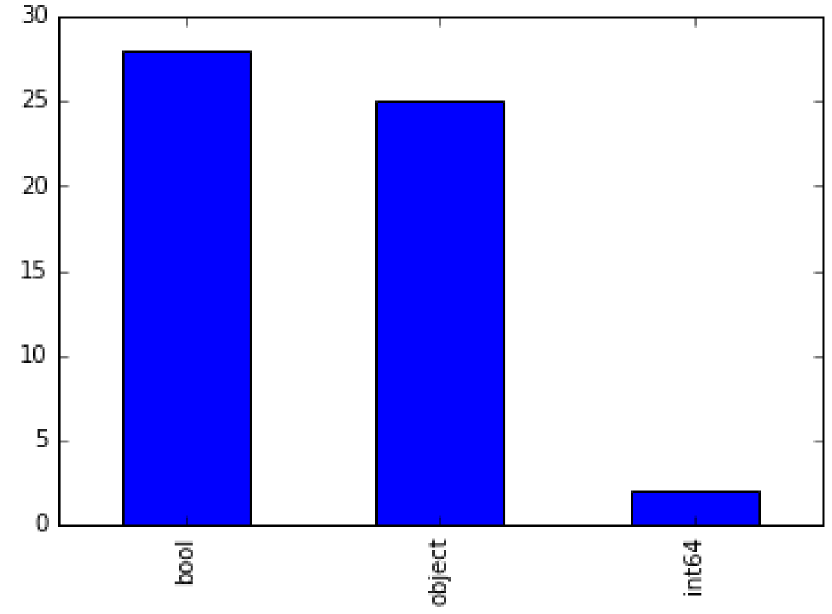
The confusion matrix in the conclusion section is achieved by training the final model in this new training set and applying it into the new test set. However, all cross validation is performed on the original training set as it was completed by the first review and took days to get the result. Therefore, no new cross validation is performed after the second review. The final model applied to the original test set on Kaggle is also trained on the original training set before the review. Please be kindly notice these differences.

Exploratory Visualization



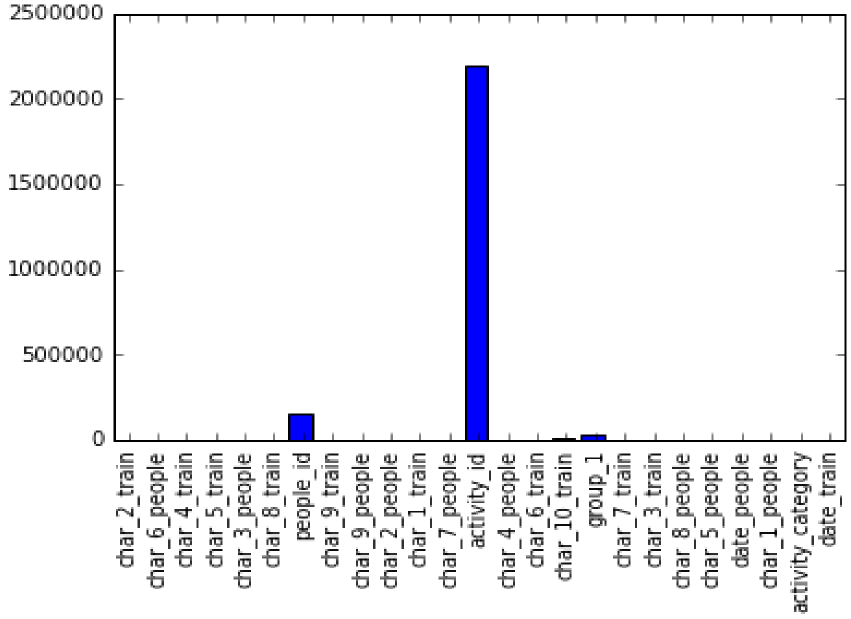
Distribution of Dependent Variable (binary) in the Training Set

From the plot above, we can see the dependent variable is nearly equally distributed so that a stratified sampling in cross-validation stage may not necessary.

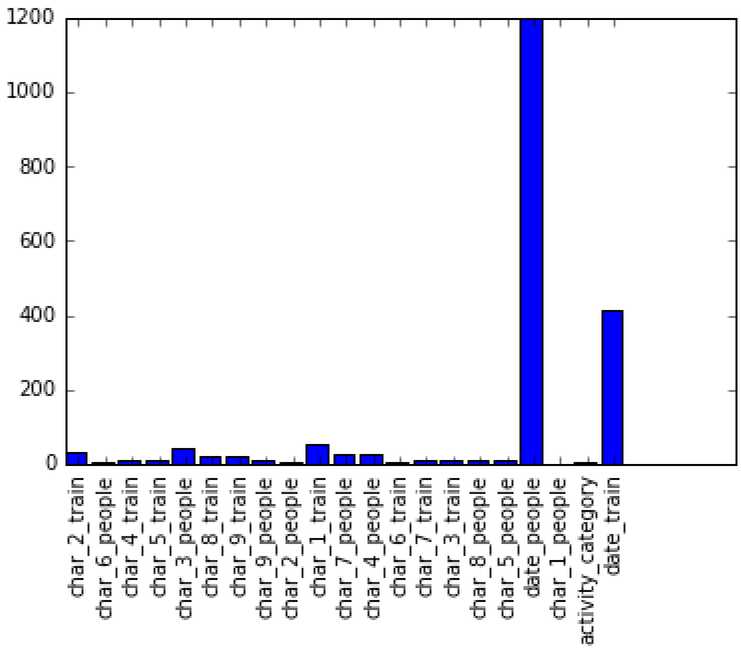


Distribution of Features' Type

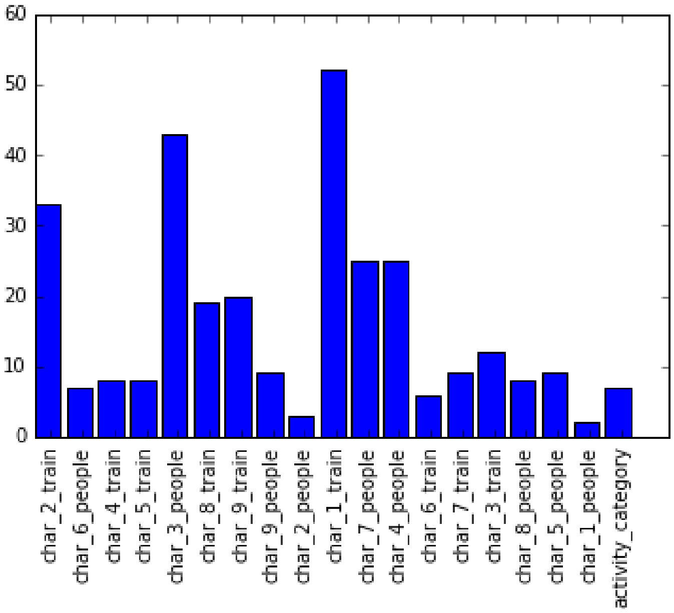
From the plot above, we can see most variables are categorical variables (binary or more category). This characteristics of data make tree-based models a good candidate.



Number of Categories for Each Categorical Variable

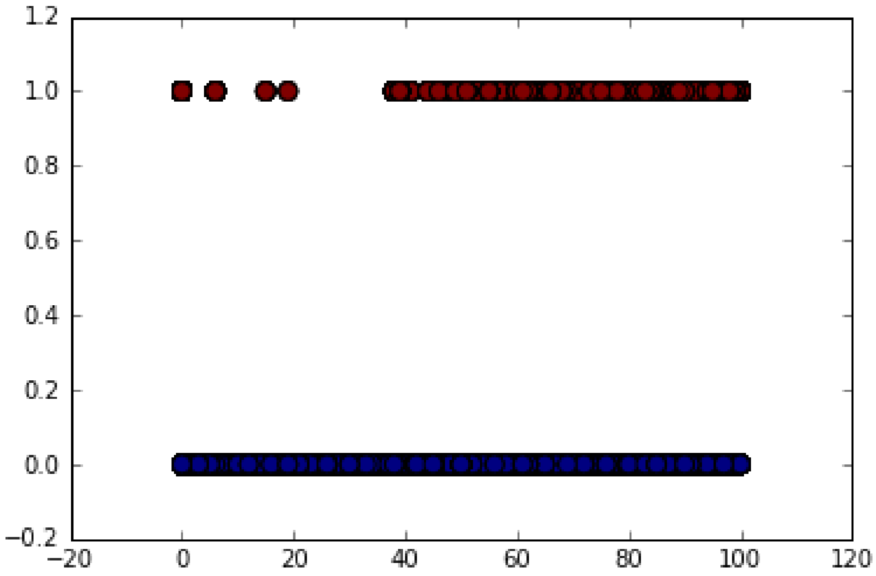


Number of Categories for Categorical Variables excluding "people_id","activity_id","char_10_train" and "group_1"



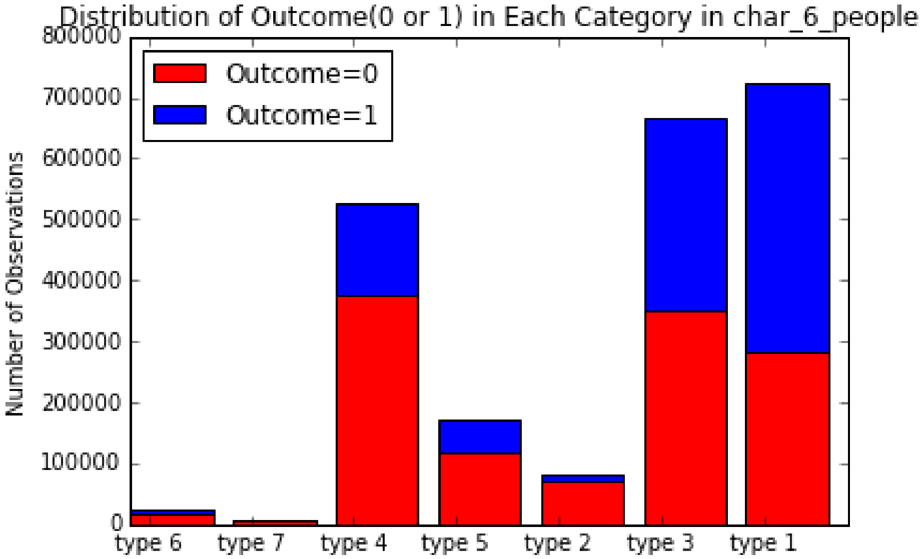
Number of Categories for Categorical Variables further excluding "date_train" and "date_people"

From the three plots above, we can see that there are several variables having large amount of categories. The initial model will remove these variables as they may cause memory error in one-hot encoding, model building and prediction process. These variables can potentially be added back later depending on whether the initial model suffers from high variance or high bias (underestimate)



Distribution of char_38 by outcome (0 or 1)

From the plot above, we can see if char_38 is less than 40, the observations are more likely to have outcome 0 than 1. The feature char_38 may have some predictive power.



From the plot above, we can see if an observation has type 6, 7, 4, 5 or 2 in feature char_6_people, it is more likely to have outcome 0 than outcome 1. This feature char_6_people has some predictive power.

Algorithms and Techniques

Since there are 53 categorical variables and only 1 numeric variables, tree-based models may be appropriate for this problem. In addition, as the number of categorical variables is large, I would expect a better performance from ensemble trees such as gradient boosting trees as such ensemble models are more able to capture the interactions among all the variables.

A high-level introduction to the decision tree and gradient boosting trees algorithm:

A single decision-tree model works as follows:

- Specify a criterion (e.g. Gini index or purity) to evaluate a split in a feature
- Grid search on an optimal split in each of the feature and compare among all the optimal splits by the specified criterion to get the best split (a feature and a split value)
- Split the data set by the best split and iterate from step 1 on the rest features
- Stop the split when a certain criterion meets (maximum number of nodes or a certain value in the criterion)

An ensemble-trees model (e.g. Gradient Boosting Trees) adds single decision-tree model in a forward addictive way and specifically work as follows:

- Fit an decision tree model to the dataset
- Fit a subsequent decision tree model to the negative gradient of the previous model
- Update the final model by add up the previous model and subsequent model
- Iterate from step 1 to add more single decision tree until a certain criterion is meet (e.g. the number of trees)

The most important two parameters need to be tuned are number of trees and maximum number of nodes in a tree. I would expect the training AUC increases as the two parameters increase. This is because the more complex the model is, the more likely it captures all the patterns in the data (this is similar to the situation in regression analysis where the training R^2 can be always improved by adding more variables or higher order terms). However, the AUC on validation and test set would be decrease if the model is too complex (too many number of tress and each tree has too many nodes) as the model will suffer from high variance.

5-fold Cross validation are used to choose the best parameters. 5-fold CV is selected as size ratio of the training set and the validation set is similar to that of the entire training set and the actual test set (~2,000,000:~500,000 = 4:1)

Benchmark

A single Decision Tree Classifier is used as the benchmark model. The average AUC score from 5-fold cross validation on the training set is used as the benchmark AUC.

Parameters: max_depth:25

Benchmark model cross validation performance

Validation	AUC Train	AUC Valid
1	0.9098	0.8952
2	0.9181	0.8627
3	0.9086	0.8955
4	0.9108	0.7958
5	0.9153	0.8781
Average	0.9125	0.8655

3. Methodology

Data Preprocessing

- Act_train.csv and act_test.csv file are merged with people.csv by feature “people_id”.
- Feature “outcome” is taken out of the act_train dataset as the dependent variable
- “people_id”,”activity_id”, “date_train”,”date_people” features are removed as they may not have predictive power (might be further preprocessed and added back in refinement)
- For all the categorical features including binary features, they are transformed to separate binary features with one-hot encoding. One-hot encoding is necessary as the numeric representation of the categorical variables implies the order of and the difference between each category is meaningful. However, there is no such meaning in these categories. Therefore, one-hot encoding is implemented here.

Implementation

- 5-fold cross validation is used in implementation. For each of the five validation, 4 folds are used as training set and the rest one is used as test set.
- 5-fold CV is selected as size ratio of the training set and the validation set is similar to that of the entire training set and the actual test set (~2,000,000:~500,000 = 4:1)
- Training set of transformed features (one-hot encoding) is put into Gradient Boosting Classifier from sklearn.ensemble library. For this initial model, parameters are set by default as follows:

Key Parameters	Default Value for Initial Model
N_estimators	100
Max_depth	3
Learning_rate	0.1

- Initial Model Performance

Validation	AUC on Training Set	AUC on Validation Set
1	0.8722	0.8760
2	0.8786	0.8499
3	0.8717	0.8787
4	0.8730	0.7824
5	0.8748	0.8640
Average	0.8741	0.8520

- Average AUC on validation set is used and reported to evaluate the model performance

Refinement:

- Grid search on parameters n_estimators and max_depth are implemented to get the best model based on validation AUC score.
- Parameter space:

n_estimators	max_depth
100	5,10,15,20,25
150	5,10,15,20,25
200	5,10,15,20,25

- These parameters are selected as I guess (before perform CV) the model with (100,5) would less than enough in complexity to capture the patterns in the data (underestimate) and (200,25) would be more than enough to capture the patterns (overestimate).So I put several points between (100,5) and (200,25) to achieve a good point in the bias-variance tradeoff.
- AUC score is averaged among 5 validation set on 5-fold cross validation.

4. Results

Model Evaluation and Validation

Result for Grid Search on (number of trees, maximum depth of a single tree)

Number of Trees	Max Depth of a Single Tree	Average AUC of 5 Fold CV on Training Set	Average AUC of 5 Fold CV on Validation Set
100	5	0.8834	0.8552
	10	0.9033	0.8629
	15	0.9206	0.8672
	20	0.9354	0.8709
	25	0.9479	0.8744
150	5	0.8857	0.8563
	10	0.9067	0.8642
	15	0.9237	0.8681
	20	0.9385	0.8716
	25	0.9506	0.8753
200	5	0.8879	0.8569
	10	0.9096	0.8645
	15	0.9266	0.8689
	20	0.9412	0.8728
	25	0.9529	0.8761
	30	0.9620	0.8786

The final model/parameters (200,30) is selected by the highest average AUC score.

The final model is robust since the AUC score on all the 5 cross-validation varies little. Therefore the model can be trusted.

From the result above, we can see the parameter Number of Tress does not impact the model performance by a lot. Instead, we do see the performance improves as the parameter Max Depth of a Single Tree increases. I should have included more values (50,75,100,…) to be tuned but the CV process is very time consuming at this point so that I will leave it to be completed after I master the distributed computing and parallel programming.

In addition, as the Max Depth of a Single Tree increase, the difference between training AUC and validation AUC increases, which signals that the model might be suffering from high variance (too many parameters to estimate). Feature engineering such as merge some less frequent categories in a categorical variable might be helpful. I would consider this as a future improvement for this project as well.

Since this is a completed Kaggle competition, I was able to train the model on the entire original training set and compare the predicted value on the test set to the real value. It turns out that the AUC on the unseen dataset is 0.89. This result further validate that the model can be generalized well.

Justification

The validation AUC score is 0.8786 for the final model and 0.8655 for the benchmark model. Therefore, the final model is stronger than the benchmark.

The AUC score for the final model is much higher than that from random guess which is 0.5. Therefore, the marketing team in RedHat can leverage the predicted outcome from the final model in their marketing campaign and they would get a much better result than randomly approaching potential or existing customers.

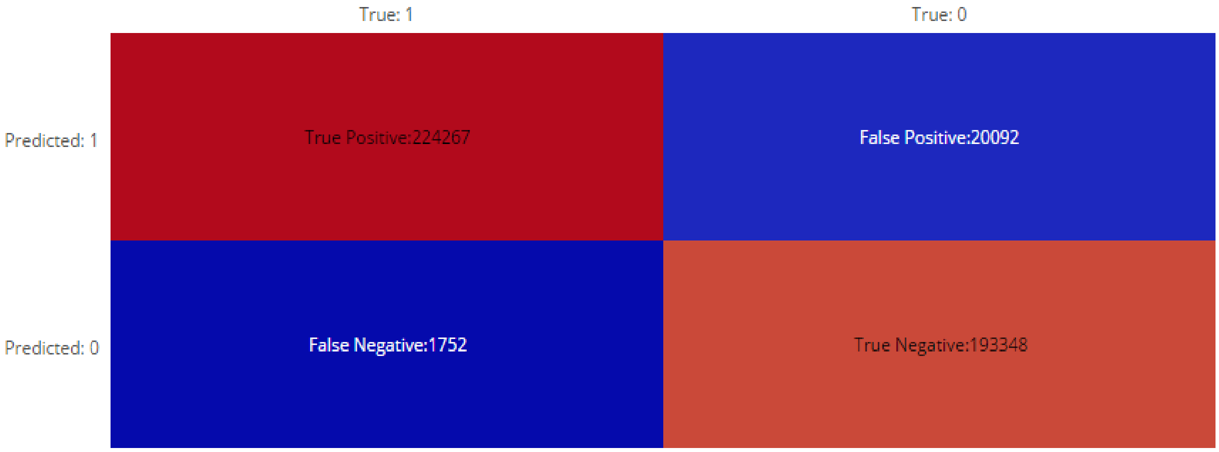
5. Conclusion

Visualization

AUC on the test set for the final tuned model and the benchmark model:

Model	AUC
Final Model: GBT	0.89
Benchmark Model: Decision Tree	0.87

Confusion Matrix on Test Set (Test set split from the original training set)



		True Class	
		Positive	Negative
Predicted Class	Positive	224267	20092
	Negative	1752	193348

Prediction Threshold is 0.5

From the confusion matrix above, we can see

- Size of test set is 439459
- Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.95
- True Positive Rate = TP/(TP+FN) = 0.992
- False Positive Rate = FP/(FP+TN) = 0.094

Reflection

- One difficult I had in the project is that there are so many categorical variables and some of them have number of categories up to 29899(e.g. feature: group_1). When making prediction with fitted Gradient Boosting Trees model, I got memory error. To resolve this problem, I split the dataset into chunks and process the data chunk by chunk, which turned out to be an effective way.
- I also revisited Andrew Ng's machine learning course at Coursera during the project where I learned how to evaluate whether the model is suffering from high bias, high variance or both and how to improve the model accordingly. This is useful when I firstly fit the model with some categorical variables removed. I know I had to add back more variables as the AUC for the training and test set is very close which means the model was more likely suffering from high bias. This technique keeps me from going into the wrong direction.
- In addition, this is the first time I deal with so many categorical variables and I learned the gradient boosting tree algorithm and was able to understand the mathematics behind it. (my own note on understanding the mathematics behind gradient boosting trees is attached in a separate file)

Improvement

- The “date” feature isn’t included in the model building process. If I were to include it and performed appropriate feature engineering, the final model performance would be better.
- The cross-validation and training time is very time consuming in a single machine. It takes days to get the final result. I am already learning Spark and distributed computing at edx and hopefully I could leverage these advanced computing technique in the near term to decrease the time.
- The winning solution at Kaggle competition uses xgboost algorithm. I should have implemented the same model. However, I couldn’t install it correctly on my machine. I hope a simple installation process for Python at Windows system will be released soon so that I can implement the xgboost model to see whether a better performance could be achieved.
- From coding perspective, I should embed some similar steps into a function and simply call the function when using it. But I don’t have enough time to refine it at this point so I will leave it to the future.