Predicting Red Hat Business Value

Machine Learning Engineer Nanodegree

Jacob Hsiao Sept 25th, 2016

I. Definition ¶

Project Overview

This project is a featured competition in Kaggle. (https://www.kaggle.com/c/predicting-red-hat-business-value)

Like most companies, Red Hat is able to gather a great deal of information over time about the behavior of individuals who interact with them. They're in search of better methods of using this behavioral data to predict which individuals they should approach—and even when and how to approach them.

In this project, I am challenged to create a classification algorithm that accurately identifies which customers have the most potential business value for Red Hat based on their characteristics and activities.

With an improved prediction model in place, Red Hat will be able to more efficiently prioritize resources to generate more business and better serve their customers.

Relative dataset:

- act_train.csv: labeled activities data for exploring features, training and cross-validating(labeled means that the outcome of each activity is provided)
- act_test.csv: unlabeled activities data for predicting and ranking the competitors by their AUC result on this set(the outcome of each activity is not provided)
- people.csv: one-to-more relations with act_train and act_test data, recording the id and features of people (one to more relation means that each people_id in people.csv matches several records in act_train.csv and act_test.csv)

Problem Statement

This problem is a supervised {0,1} classification one, which is about predicting whether a business activity has the potential business value(labeled 1) or not(labeled 0) using the model trained from training data.

The goal is to select and extract proper features and build an effective model to predict the outcome of each business activity.

To solve this problem, three main steps need to be followed.

The first one is making an EDA(exploratory data analysis) on features, find their relevance on label and explore hidden patterns inside the data by doing statistics and plotting graphs on features.

Then feature engineering is carried on based on the EDA in previous section. Feature engineering means that making some selection, transformation and extraction on features. Well-engineered features mean that all features are predictive and the performance of the classification model will be improved quite a lot.

Next, train several classifiers with different learning algorithms on training set to make cross validation on them. 3 best models will be selected based on their cross validation results and their parameters are further tuned in order to maximize their performance.

Finally, well-tuned models are trained with the whole training set and make prediction on whole test set. Then I will ensemble the predicted result from those classifiers to generate the final output.

Metrics

The metric I chosed to evaluate my model is AUC (Area Under receiver operating characteristic Curve).

The AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example, i.e. P(score(x+)>score(x−))

It is calculated this way.

$$AUC = \frac{\sum_{ins_i \in positiveclass} rank_{ins_i} - \frac{M \times (M+1)}{2}}{M \times N}.$$

Where M is the number of positive samples and N is the number of negatie samples.

There are 2 reasons why I choose this metric.

First, it is the metric used in this competition to rank the competators.

Second, the output of my model is probablities of outcome equal to one, in other words, a float number in range [0,1], not the actually prediction which is {0,1}. So the confidence of the model is also evaluated using this metric. For a positive sample, if the prediction is 0.8 instead of 0.7, then the AUC score would be increased a little as this prediction is correct and more certain.

The range of this value is [0,1] and the ideal value of this metric is 1 but 0.95 above is a good value.

II. Analysis

Data Exploration

Size of datasets:

- act train data: (2197291, 19)
- act test data: (498687, 18)
- people data: (189118, 45)

Act training/testing data sample:

people_id	ppl_100
activity_id	act2_1734928
date	2023-08-26 00:00:00
activity_category	type 4
char_1	NaN
char_2	NaN
char_3	NaN
char_4	NaN
char_5	NaN
char_6	NaN
char_7	NaN
char_8	NaN
char_9	NaN
char_10	type 76
outcome	0

People data sample:

people_id	ppl_100
char_1	type 2
group_1	group 17304
char_2	type 2
date	2021-06-29 00:00:00
char_3	type 5
char_4	type 5
char_5	type 5
char_6	type 3
char_7	type 11
char_8	type 2
char_9	type 2
char_10	True
char_11	False
char_12	False
char_13	True
char_14	True
char_15	False
char_16	True
char_17	False
char_18	False

char_19	False
char_20	False
char_21	True
char_22	False
char_23	False
char_24	False
char_25	False
char_26	False
char_27	True
char_28	True
char_29	False
char_30	True
char_31	True
char_32	False
char_33	False
char_34	True
char_35	True
char_36	True
char_37	False
char_38	36

From the samples, it is obvious that all the field names and categorial names are anonymous. Most of the fields are categorial except char_38 which is numeric.

Then the act training/testing data is combined with people data.

Here is field names and unique values in each fields of the combined data:

- people_id : 189118
- activity_id : 2695978
- date_x : 411
- activity_category : 7
- char_1_x : 52
- char_2_x : 33
- char_3_x : 12
- char_4_x : 8
- char_5_x : 8
- char_6_x : 6
- char_7_x : 9
- char_8_x : 19
- char_9_x : 20
- char_10_x : 6970
- year_x : 2
- month_x : 12
- day_x : 31
- char_1_y : 2
- group_1 : 34224
- char_2_y : 3
- date_y : 1196
- char_3_y : 43
- char_4_y : 25

- char_5_y : 9
- char_6_y : 7
- char_7_y : 25
- char_8_y : 8
- char_9_y : 9
- char_10_y to char_37: 2
- char_38 : 101
- year_y : 4
- month_y : 12
- day_y : 31

Most of the fields are {0,1} categorial fields. Most of the categorial fields do not have so many unique categories except “group_1” and “char_10_x” which have plenty of unique values(even much more than “char_38” which is a numeric field).

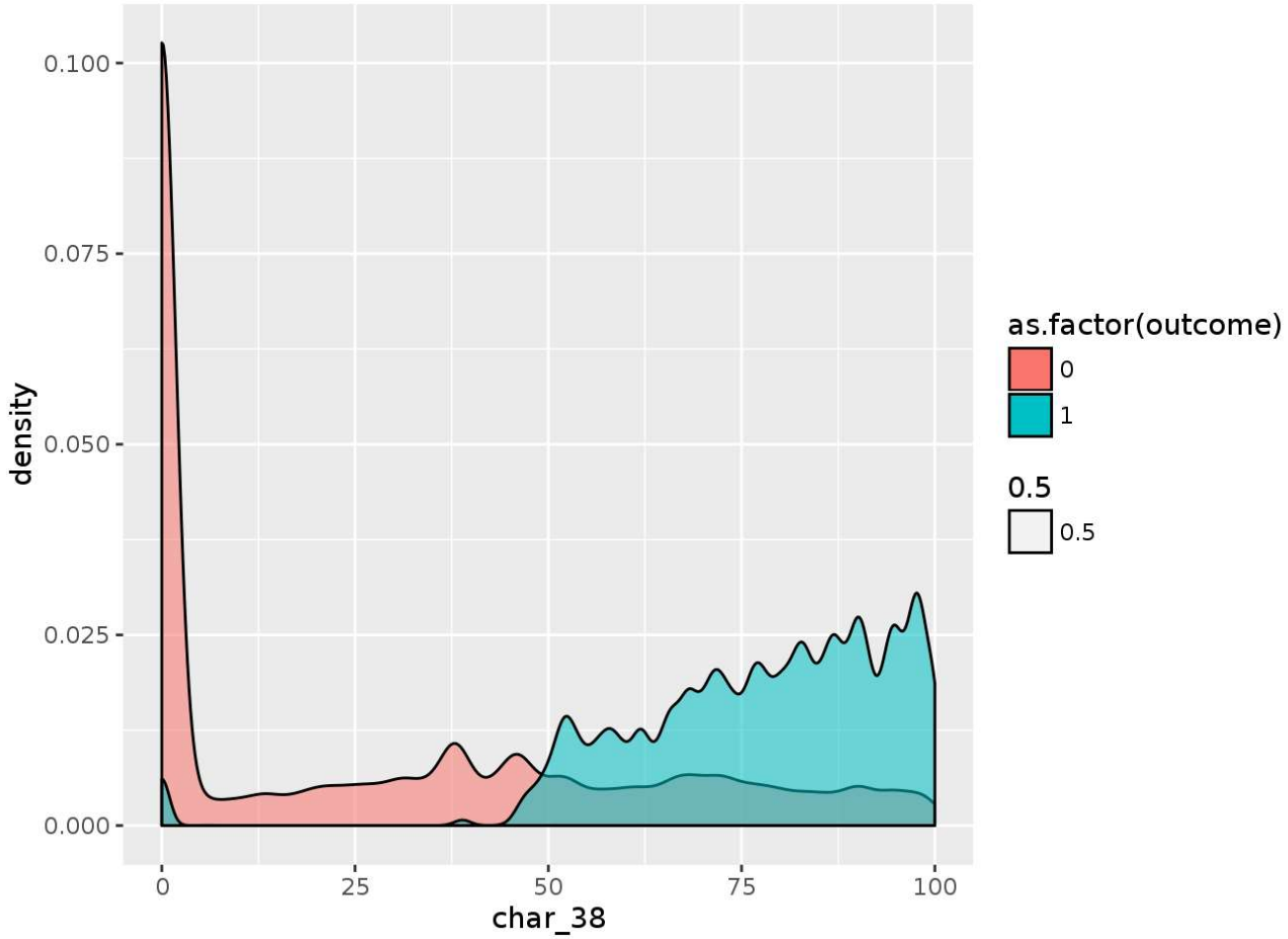
To explore how training and testing dataset is split, I made some interaction analysis on fields “people_id” and “group_id”. I found that training and testing data is split by people. Most of the groups are presented in both training and testing sets and some of them are “training only” or “testing only”.

Exploratory Visualization

Note: Most of the visualizations are not my work. They mainly came from sites (https://www.kaggle.com/j0shua/predicting-red-hat-business-value/redhat-hack-in-plain-english) and (https://www.kaggle.com/apapiu/predicting-red-hat-business-value/redhat-eda). Those are cited with (*).

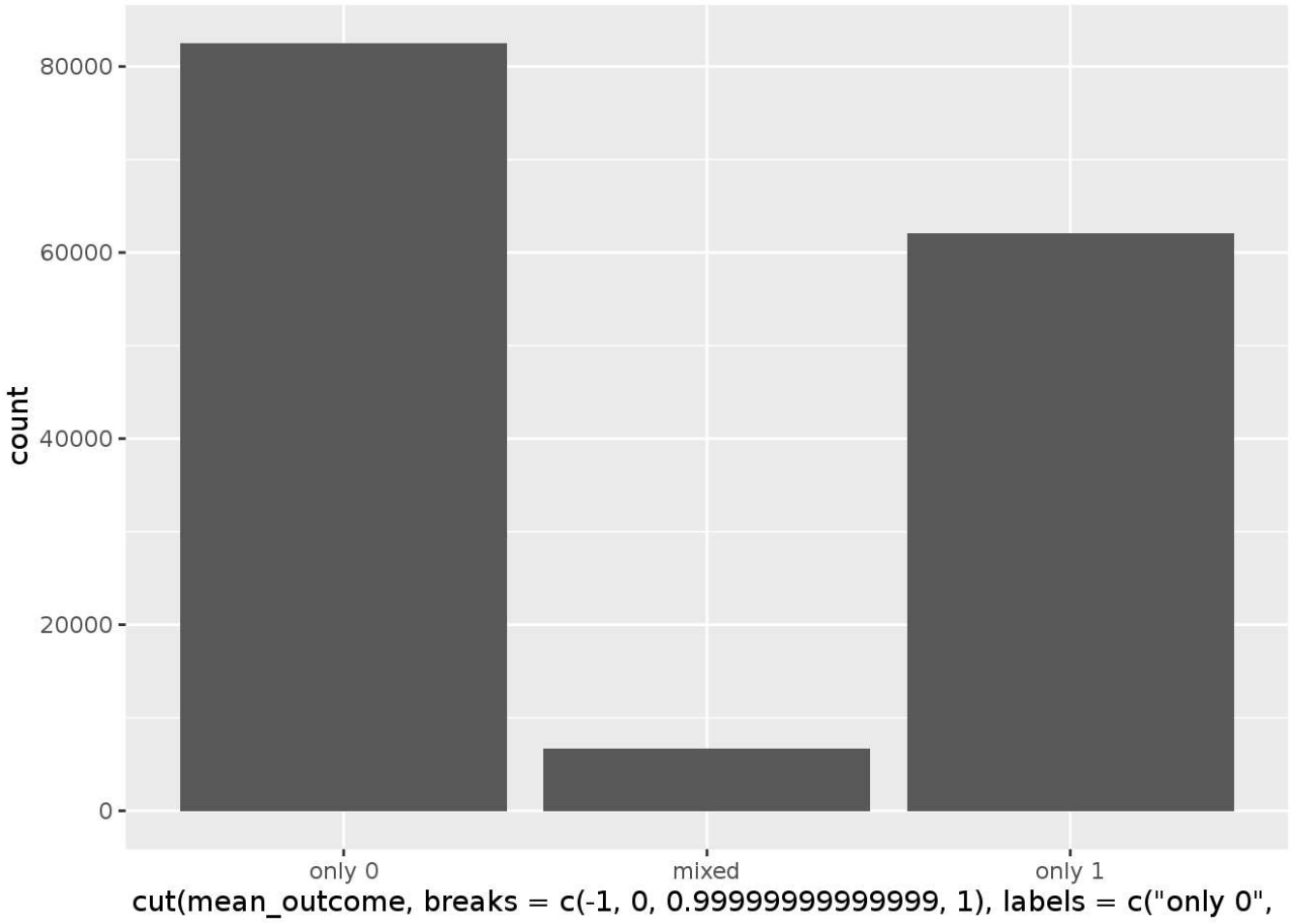
The first feature which is explored is char_38, the only numeric feature in this task.

What is distribution of char_38 in train set split by outcome?



(*)

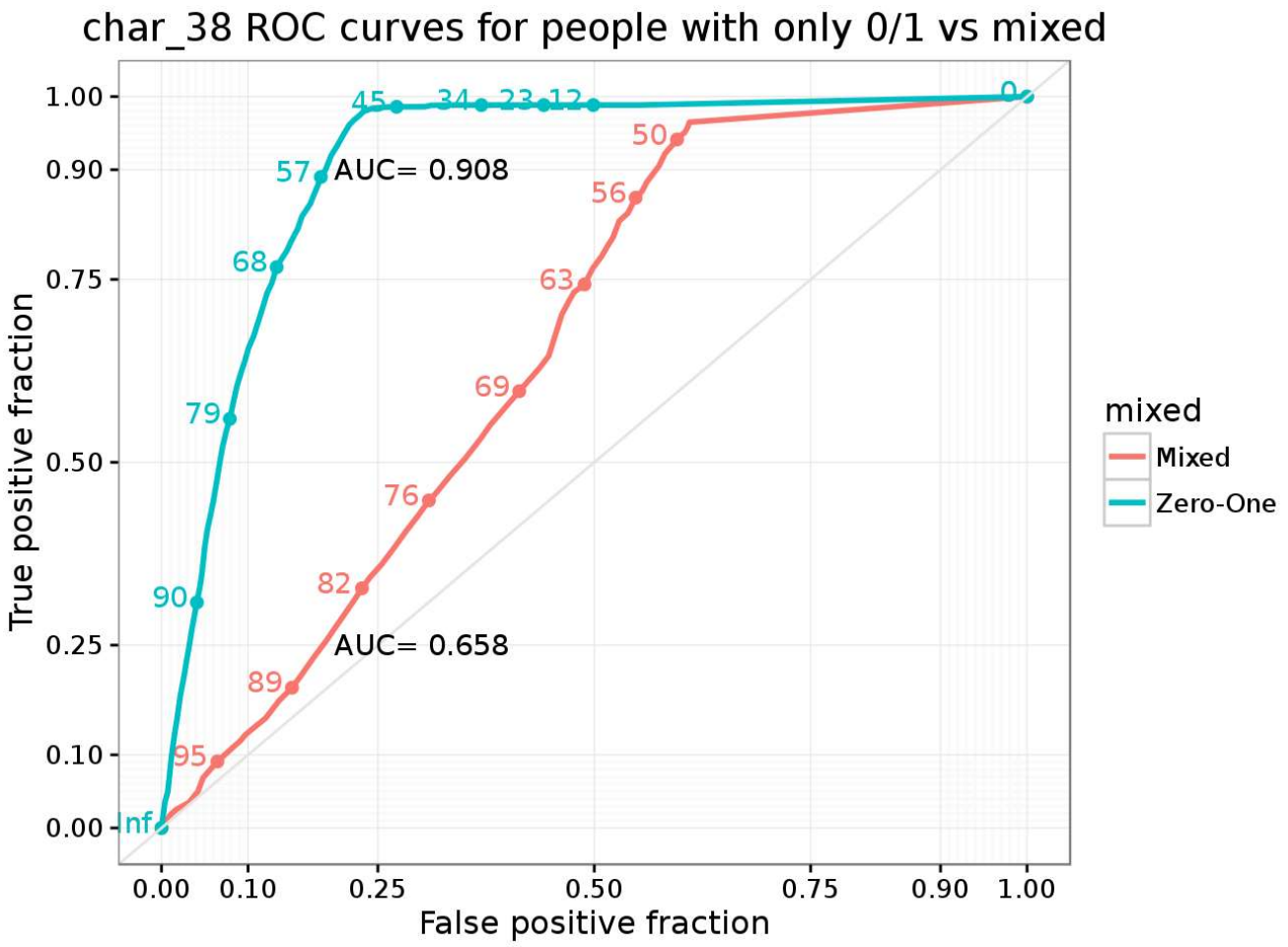
Then dive into “people_id” feature. I assume that most of people have steady outcome of 0 or 1 and a little amount of them changes over time. To prove this, let us see how many people have mean_outcome of 0, 1 or something in between.



(*)

My assumption is right.

How well is char_38 only for predicting the outcome for each of these categories(uniform or mixed)? What is the char_38 AUC for each category?

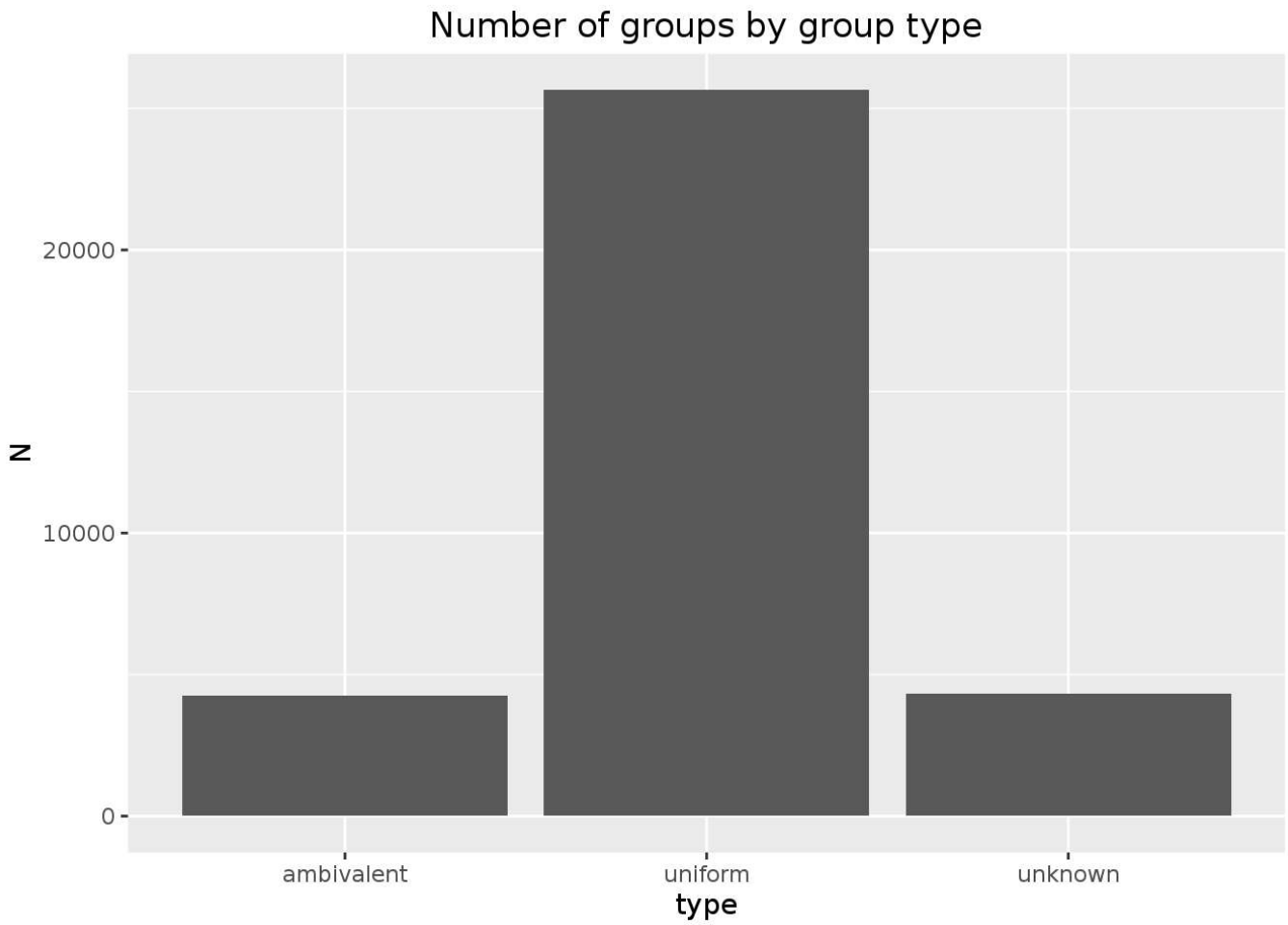


(*)

As one can see char_38 does particularly poor job at predicting outcomes for those people who “changed their outcome” during the course of actions. Therefore we will focus on those.

All people are belonging to their group. So next feature I am going to dive into is “group_1”.

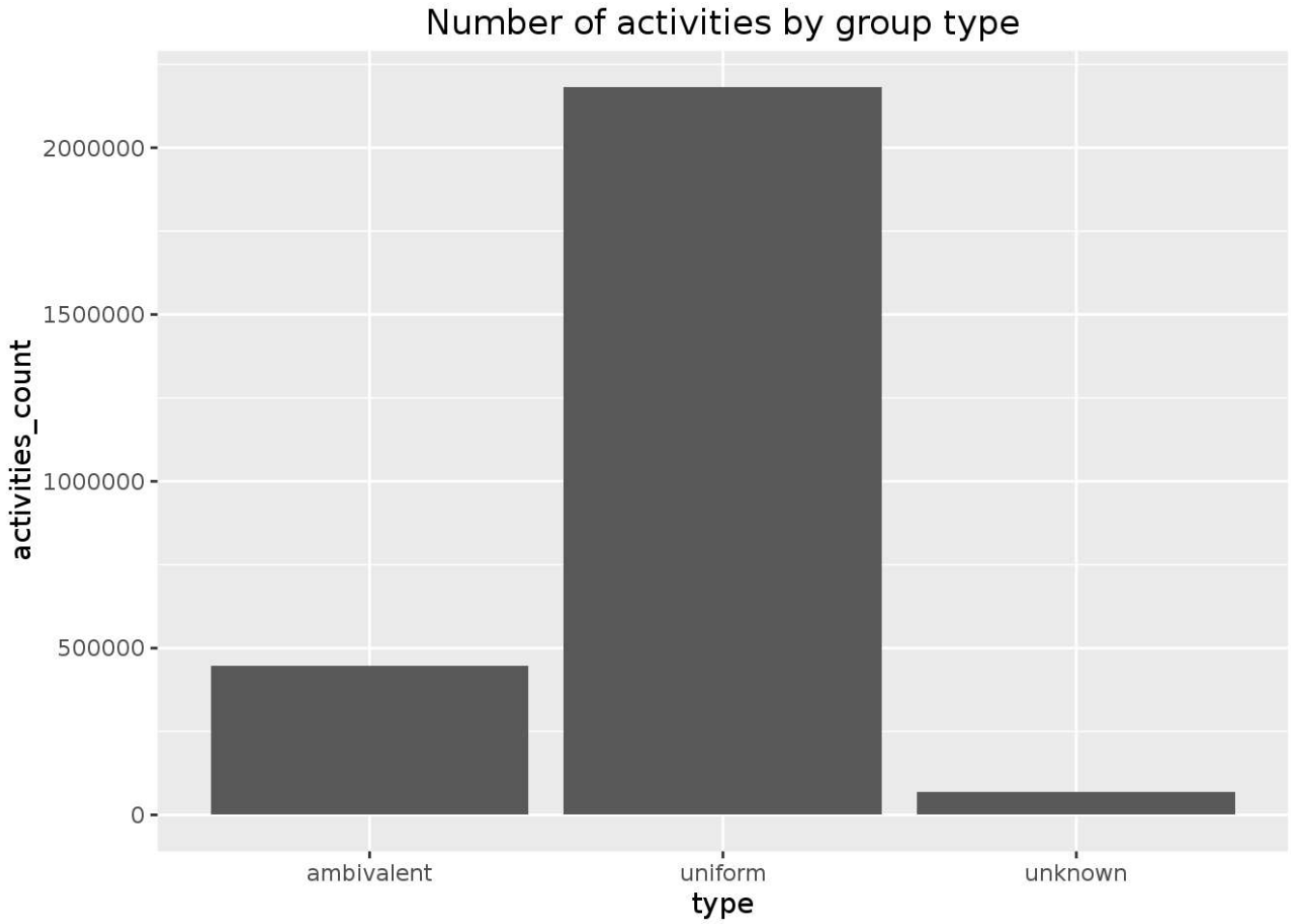
How many groups have only one outcome (“uniform group”)? How many groups exhibit both outcomes (“ambivalent group”)? How many groups presented on testing set have no presentation in training set(“unknown”, these may be ambivalent or uniform, we don’t know)?



(*)

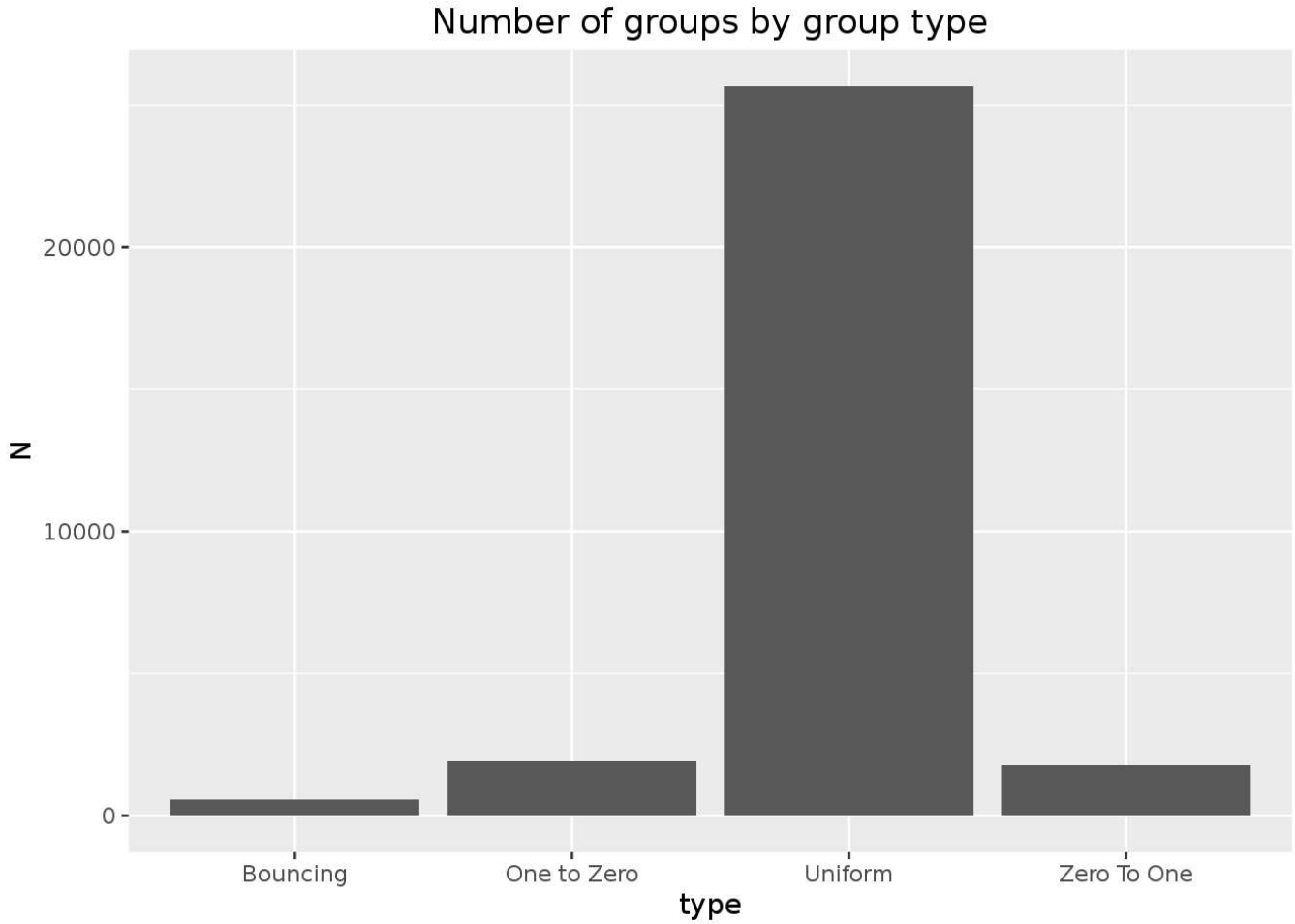
Well, most of the groups have uniform outcome, same as “people_id” field.

Then, how many activities are in each category of groups?



(*)

One more interesting questions. Among those ambivalent groups, how many are changing once (0->1 or 1->0) vs those who change labels several times.



(*)

What about the count of groups grouped by number of changes?

#of change	0	1	2	3
#of group	25646	3687	565	1

What what is the group id for that one group that changes the sign three times? it's Group 12187. Lets see this one and some other groups on the time chart (test set data points shown at outcome==0.5)

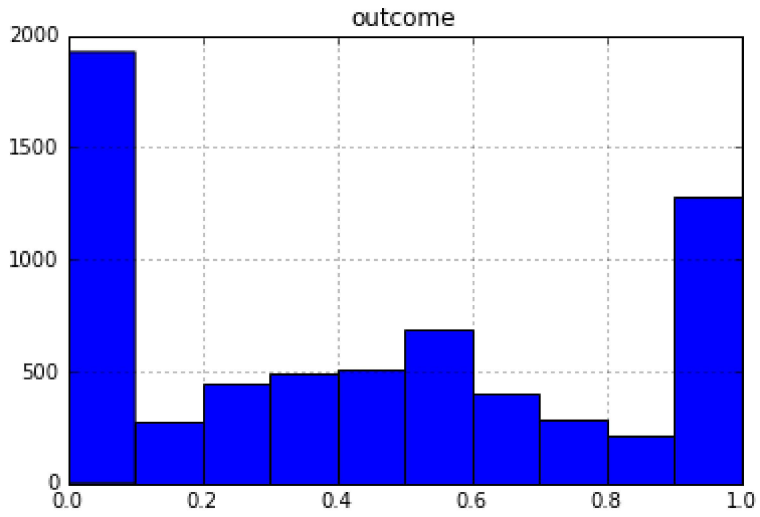
(*)

So in conclusion, there is a leakage(which means that the label can be directly filled using the data in training set) according to this pattern on "group_1" and "date_x" fields.

However, by interaction analysis on "group_1" field on training and testing dataset, there are still 70k rows on testing set that are unaffected by this leakage.

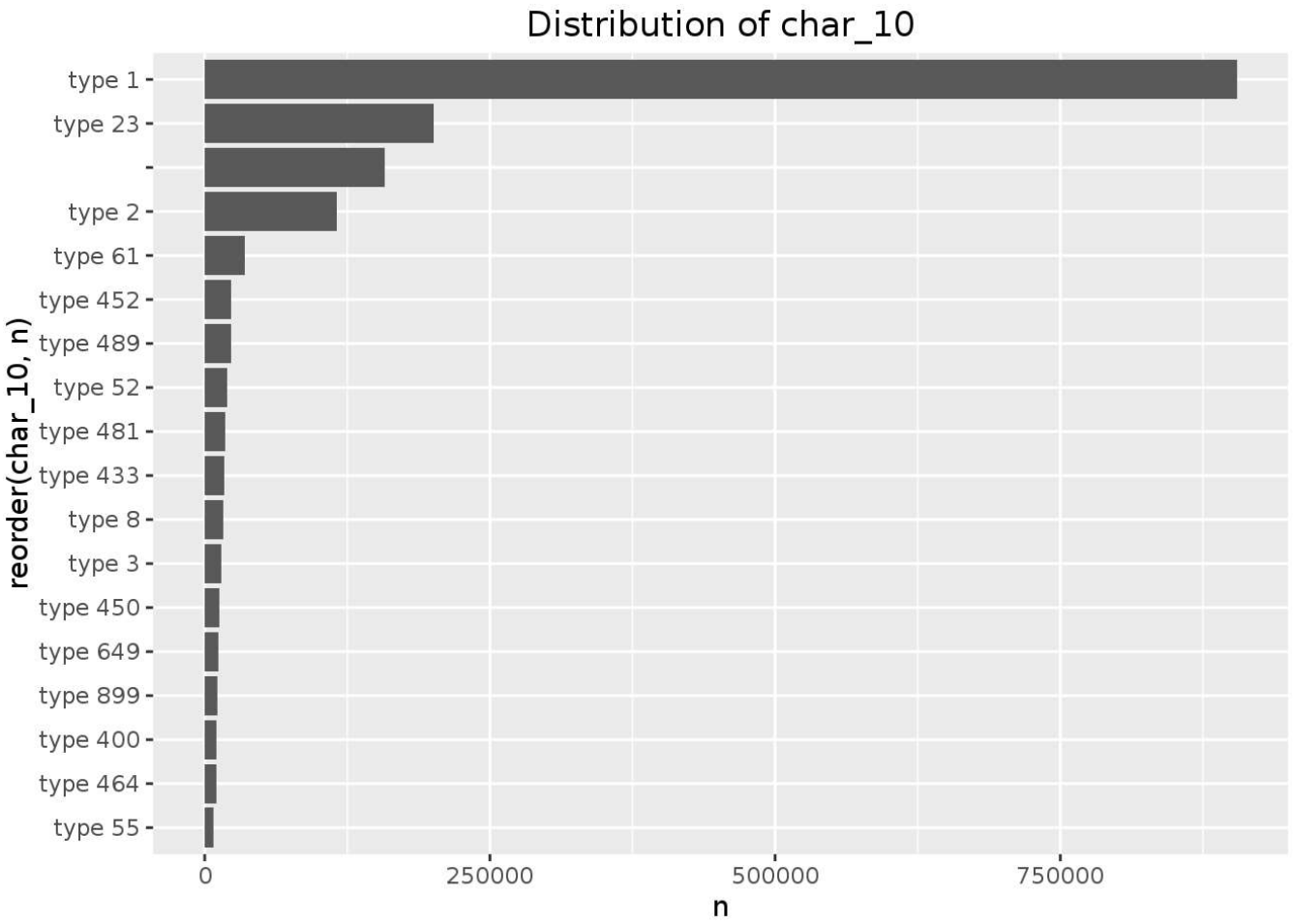
So a learning model is still needed to be built in order to predict the outcome of those rows and also ensemble the prediction of other rows with the leakage fill to give a more comprehensive prediction on testing set. The next feature I am interested in is "char_10_x"

First, plot a histogram on the average outcome group by char_10_x.



Little bit same as people_id and group_1 fields, which most of the categories has uniform outcome of 0 or 1 and some of them have some changes.

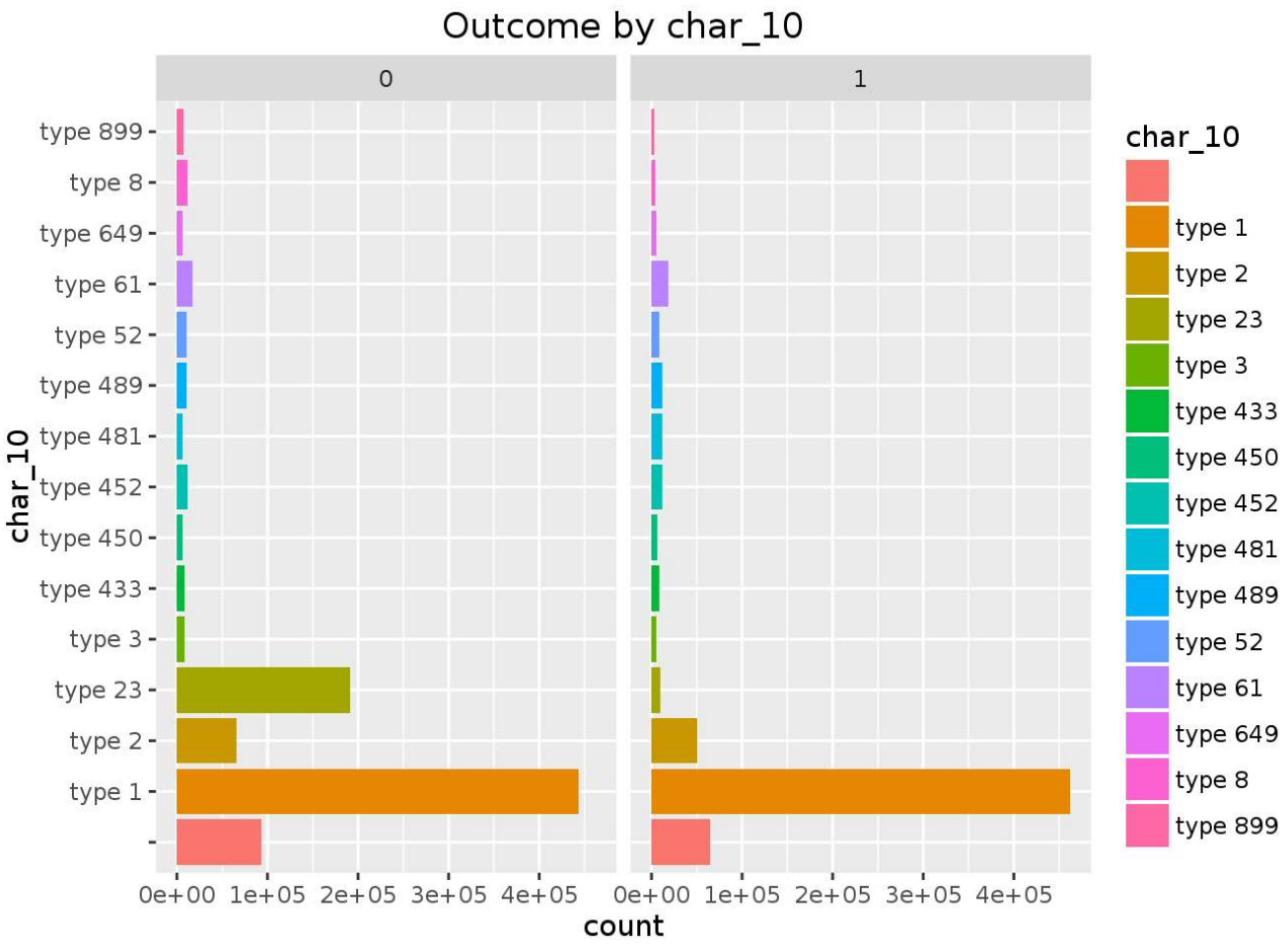
However, what about the distribution of char_10_x?



(*)

The majority is type_1 which is about 70% of the data.

And next, distribution grouped by outcome.



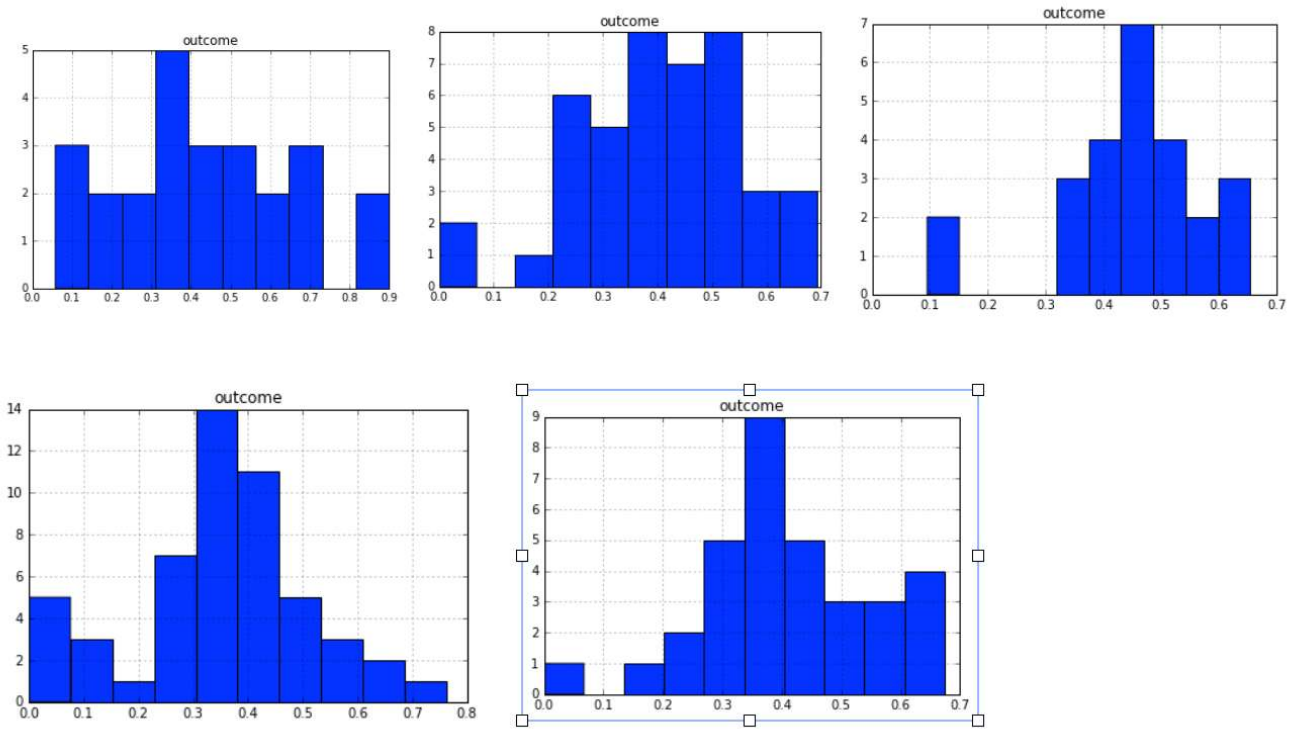
(*)

It shows that type_1 is distributed same on both outcomes.

Comparing to the histogram of average outcome on each category, the categories with uniform outcome are not the majority of data and the major category “type_1” has same probability on both “0” and “1”.

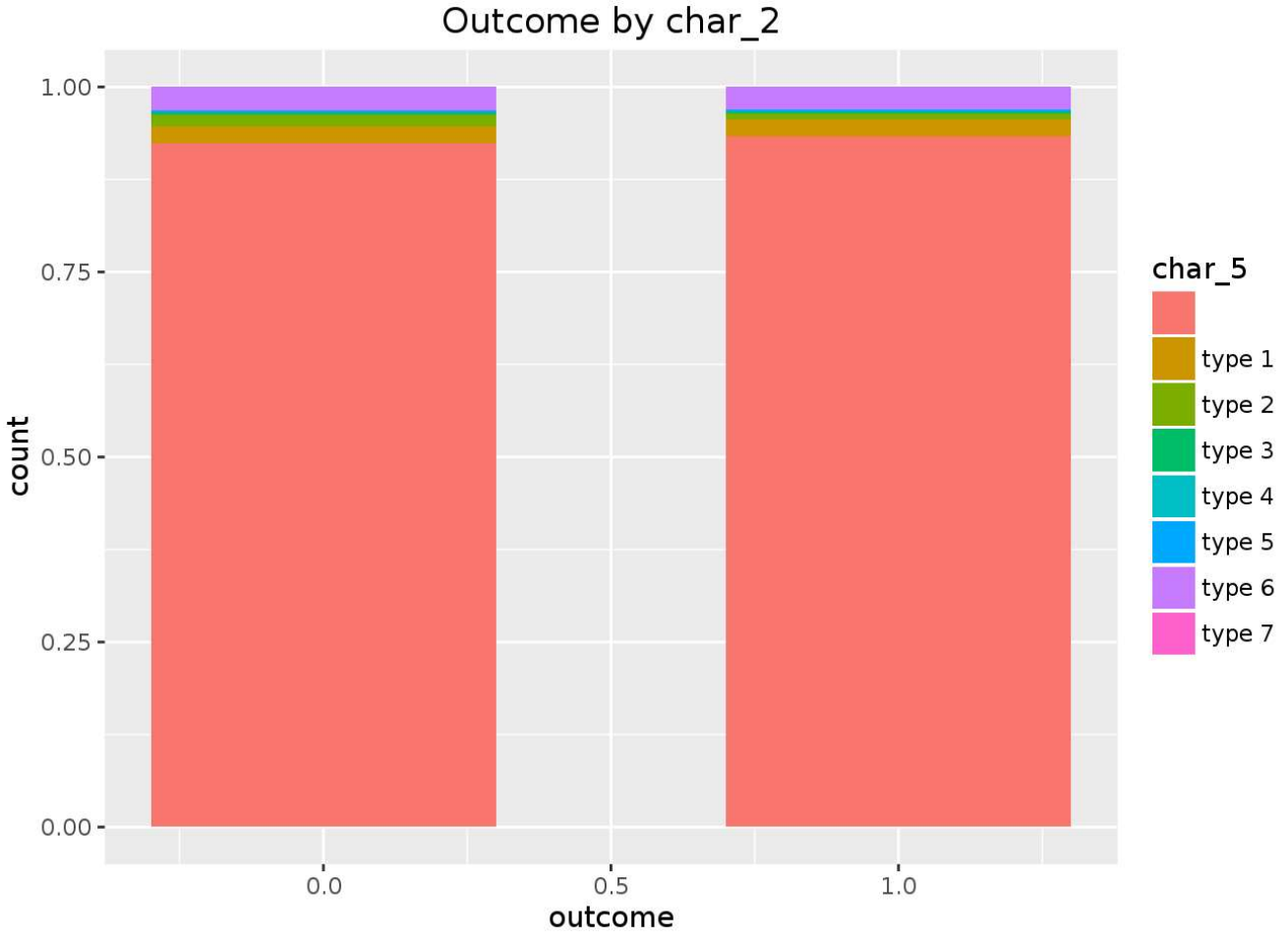
Therefore, this feature should be eliminated.

Similar explorations on fields “char_1_x”,"char_2_x","char_3_y","char_4_y","char_7_y" which has plenty of unique values and the results are shown below.



Then I found that they are all unrelevant with the outcome so they are all eliminated.

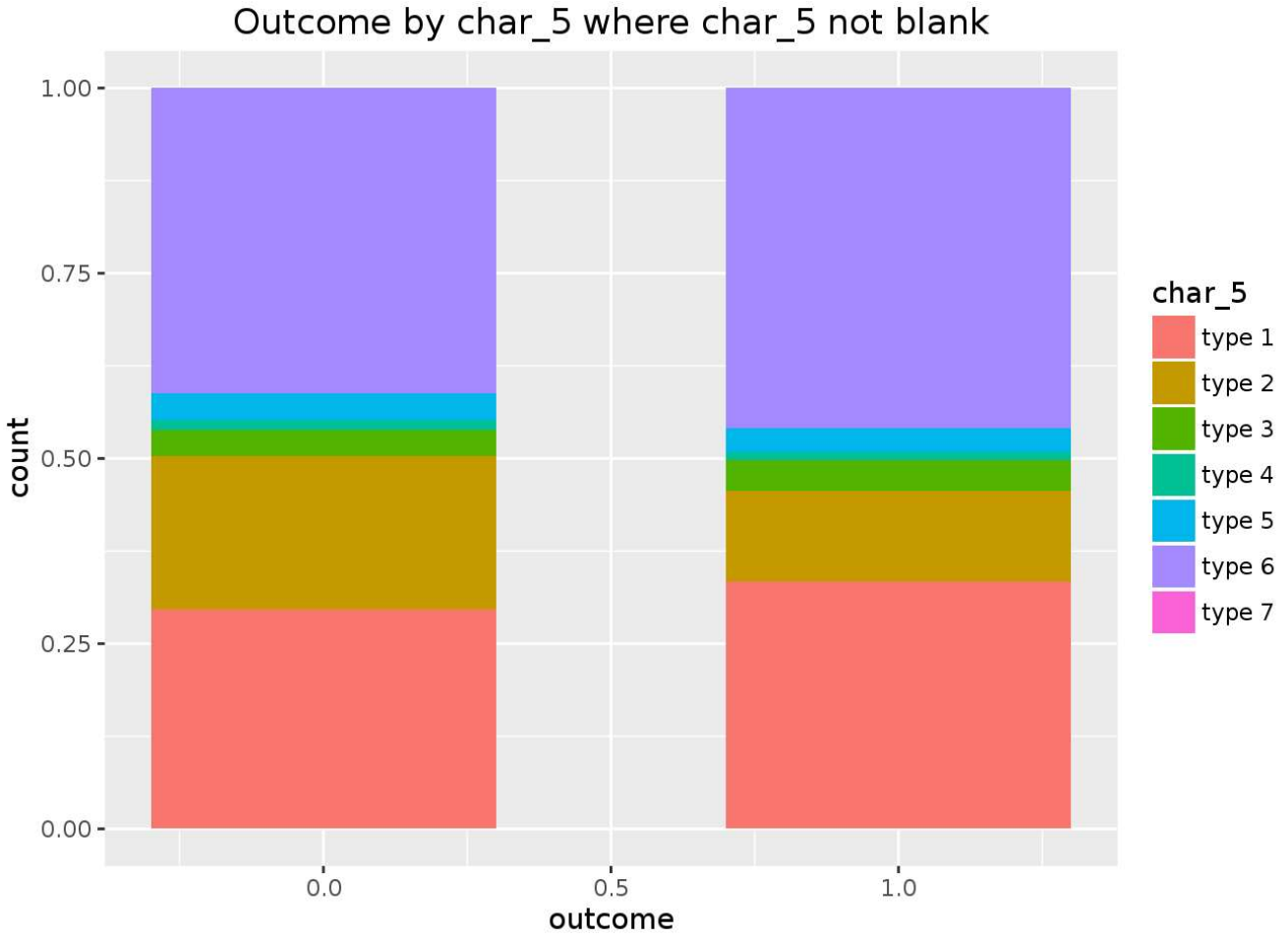
Finally, I found that most of the activity categorial fields have plenty of NA values as the samples illustrated in above section. Let us take char_5 as example. (Note that this graph's title is wrong, it should be "output by char_5")



(*)

Nearly 90% of the data are not provided.

What if we ignore the blank ones?



(*)

Even the blank is ignored, the rest data is still not predictive.

So most of the activity fields which have plenty of NA are eliminated.

They are char_1_x to char_9_x.

Algorithms and Techniques

From the exploratory stage, most of the test data are filled using group1-date leakage. However, there are still 70k of data which are not affected by this trick and they are going to be predicted using the model built by learning the training data.

Algorithm Selection:

In this stage, I am going to try several models in SimpleGridSearch, get 3 best based on CV results and do further GridSearch on them to find their best parameters.

The candidate algorithms are:

- Gaussian Naive Bayes(no parameter to tune)
  Pro: No parameters needed to be tuned
  Con: Assume that all features are independent
  Reason: No parameters needed to be tuned so faster to tune comparing to other algorithms

- K-Nearest Neighbours(tune n_neighbours)
  Pro: No input assumption, not sensitive to abnormality of data
  Con: High complexity
  Reason: The data points with the same output always locate very near in space
- Logistic Regression(tune C)
  Pro: Fast to train
  Con: Depend too much on kernel function when the data is not linearly seperateble
  Reason: Fast to tune and train with a good accuracy, pretty efficient
- Random Forest(tune n_estimators)
  Pro: The strength of ensembling is well used to increase the performance
  Con: Too slow to train
  Reason: Tree algorithms should always be a choice in classification case and this is picked rather than Decision tree as its ensembling feature would increase the performance a lot
- Gradient Boosting(tune n_estimators)
  Pro: No transformation needed to data(include categorial features)
  Con: Too slow to train
  Reason: A very popular algorithm in Kaggle. Similar to random forest which the advantage of ensembling is well used and a different approach of parameter update may cause a difference in performance

After selecting and tuning, 3 best models with their best parameters are selected and prepared for training and predicting.

They are going to be trained on whole training set, then make their predictions on whole test set, and finally, emsemble their results to give the final output.

Combining the results filled by leakage, I am going to try 2 methods: Fill those unaffected rows only, or ensemble the filled labels with the output generated by models.

Benchmark

On kaggle ranking leaderboard of this competition, to get a bronze medal(top 10% of competitors), the benchmark of AUC is 0.991

And this is chosen as my benchmark of model performance.

If my model's performance on testing set reach this benchmark, it would be able to correctly predict the outcomes of 99% activites in RedHat company and I will get a bronze medal(or better) in this competition.

III. Methodology

Data Preprocessing

Accroding the EDA done above, the features selected are:
activity_category, char_1_y, char_2_y
char_5_y, char_6_y, char_8_y
char_9_y, char_10_y, char_11
char_12, char_13, char_14
char_15, char_16, char_17
char_18, char_19, char_20
char_21, char_22, char_23
char_24, char_25, char_26
char_27, char_28, char_29

char_30, char_31, char_32
char_33, char_34, char_35
char_36, char_37, char_38

Most of them are {0,1} features. Some of them are categorial features that have distince categories no more than 15. Feature "char_38" is a numeric feature which is highly predictive.

Then the categorial features are going to be preprocessed using OneHotEncoder. This is about transforming the categorial fields into a sparse matrix where each column corresponds to one possible value of one feature.

Then outlier detection is carried on char_38, the only numeric feature. It turns out that no outlier presents in this field.

No features is extracted because I did not get an idea of how extracting them.

Finally, after selection and encoding, PCA is used to reduce the dimensions. After pca transformation, 95% of the variance are remained with 36 reduced features.

The selection and transformation processes are carried on both training and testing data sets.

Implementation

All coding process are documented in ipython notebook which is uploaded together with this report.

First, the leakage fill is generated by the code written by AlexIlchik in Kaggle Forum(in code file "script.py") and saved in file "Submission_testdt.csv". The unaffected rows are remained with NA in outcome.

As the approach mentioned in "Algorithm&Techniques" section, algorithm selection is carried on using the transformed training data.

Algorithm	TunedParameter	BestParameterValue	BestAUCScore	Selected
GaussianNB			0.667	No
KNearestNeighbours	n_neighbours	15	0.689	No
LogisticRegression	C	10000	0.849	Yes
RandomForest	n_estimators	500	0.854	Yes
GradientBoosting	n_estimators	500	0.861	Yes

Algorithms selected are LogisticRegression, RandomForest and GradientBoosting.

They are going to be tuned further in order to maximize their performances and ensembled in different weights with each other's prediction and the leakage fills.

Refinement

For refinement, the previous section has already done some simple tuning in learning algorithms.

After the previous section, best models are selected and prepared for tuning further.

Algorithm	TunedParameter	RangeOfValues	BestParameterValue	BestAUCScore
LogisticRegression	C	8000,10000,12000	10000	0.849
RandomForest	n_estimators	400,500,600	500	0.854

GradientBoosting	n_estimators	500,750,1000	750	0.872

Then the models with their best parameters are trained on whole training set and give their predictions on testing sets.

IV. Results

Model Evaluation and Validation

Evaluate my own model using K-Fold cross validation

First, my own model is evaluated using K-Fold cross validation on training data only.

The results shown below.

Fold	AUC Score
1	0.872
2	0.874
3	0.872
4	0.873
5	0.875
Average	0.873

The variance between each fold is pretty small. Maybe that is results from the huge amount of data in training set which is enough to train the model with sufficient data and test is with various input.

Ensemble with the leakage fill and evalutate them using kaggle leaderboard

Next, the final models included the ensemble of fills of the leakage data which is created by Alex's script from Kaggle forum which I referenced in my code and the predictions from my own final model. As the pre sections mentioned, there are 498k rows total in testing dataset, which 428k of them can be directly filled by leakage and 70k of them remain unaffected.

The way of evaluating the final models is to submit the prediction of the unlabeled test sets to Kaggle and it will then report the AUC score of the test data prediction and the ranking of this AUC score in all competitors on the LeaderBoard.

I believe this way of evaluating is reasonable enough as the test dataset is various enough to test my model in a variaty of inputs.

First, I submitted the prediction which most of the outcomes were filled directly from leakage data and only the unaffected rows were filled directly by my own model which is the ensembling of 3 selected algorithms. I name this model "M1". The M1 got **0.988962 AUC on LB**.

Then, with remaining the prediction for unaffected rows, for leakage-affected rows, the prediction of the whole testing set made by my own model was ensembled on 1:1 weight with the leakage fill. This model is named "M2". The M2 got **0.987350 AUC on LB**.

It turns out that ensemble the leakage fill with my prediction may decrease the performance on testing set a little bit. Maybe that is because that my prediction is more unsure than the leakage fill so this model tends to have less confidence and this leads to a decrease in AUC.

Finally I tried ensemble M1 and M2 on weight 2:1 and name it M3. The M3 got **0.989185 AUC on LB** which was the best results among all models.

I think this model is robust enough as its AUC score shows that it has enough ability to predict most of data correctly.

Change in training data will not affect the performance of this model as well because in selection process the K-Fold cross validation is used to evaluate my model on training set only. Model evaluated and selected this way would not be affected by the small perturbations training data.

Justification

The AUC score of the final model is 0.989, which is 0.002 below the benchmark.

However, I believe that my final solution is significant enough for solving this problem because an AUC result of 0.989 is high enough to correctly predict the outcomes of most of the activities.

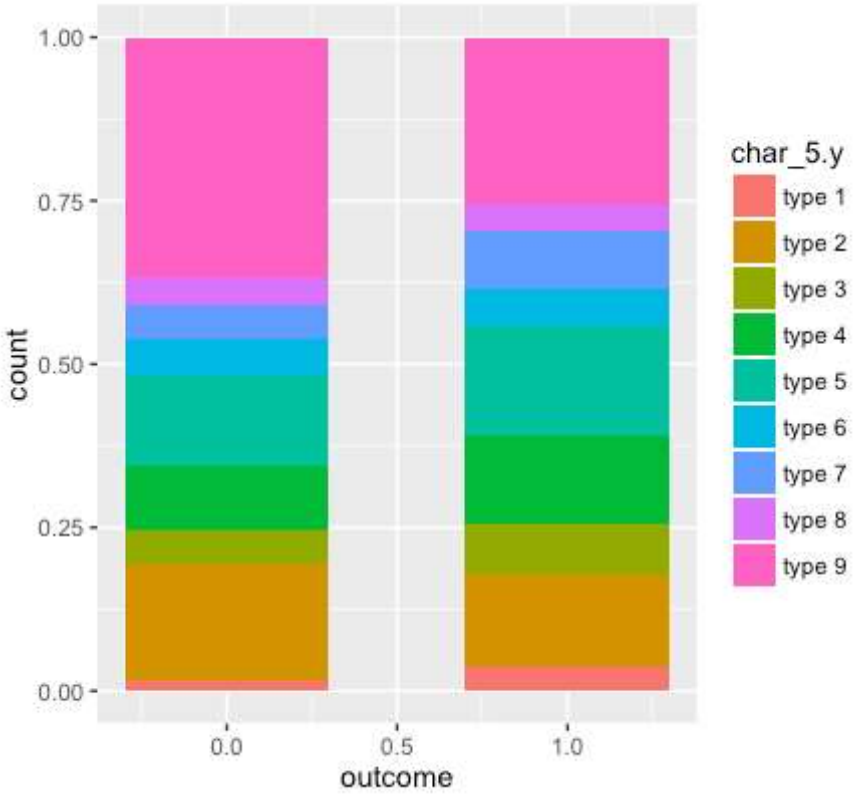
V. Conclusion

Free-Form Visualization

Note: In this section, the visualizations are all made by me using R and ggplot2.

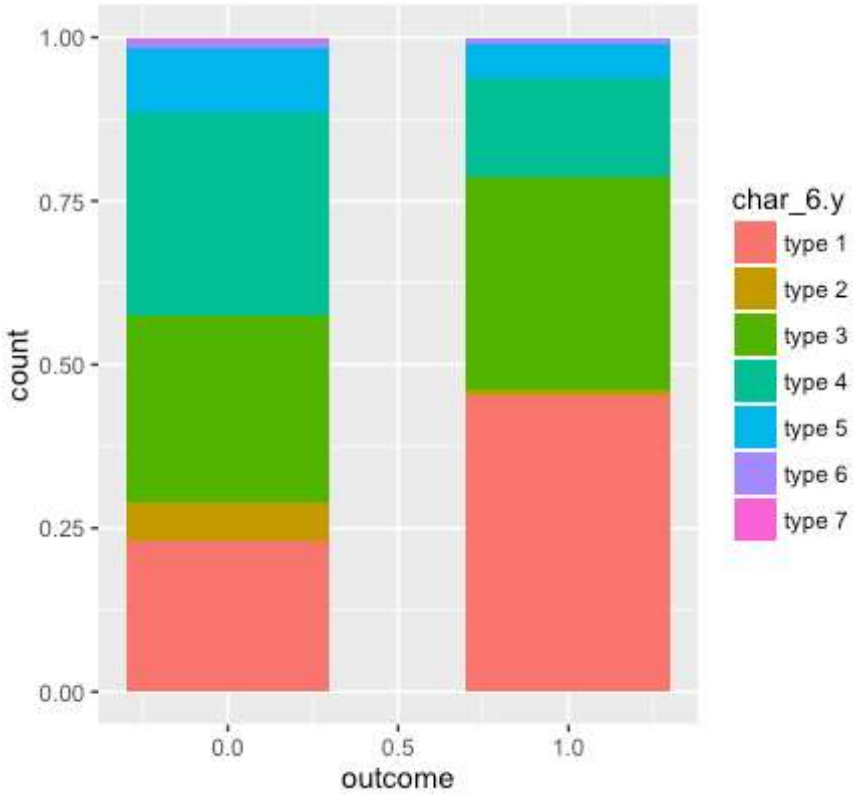
Some of the categorial features were not discussed in Explorative Visualization section. They are remained in both training and testing sets with OneHotEncode transformation. Here I am going to make a visualization on them to see whether they are predictive.

First is char_5_y:



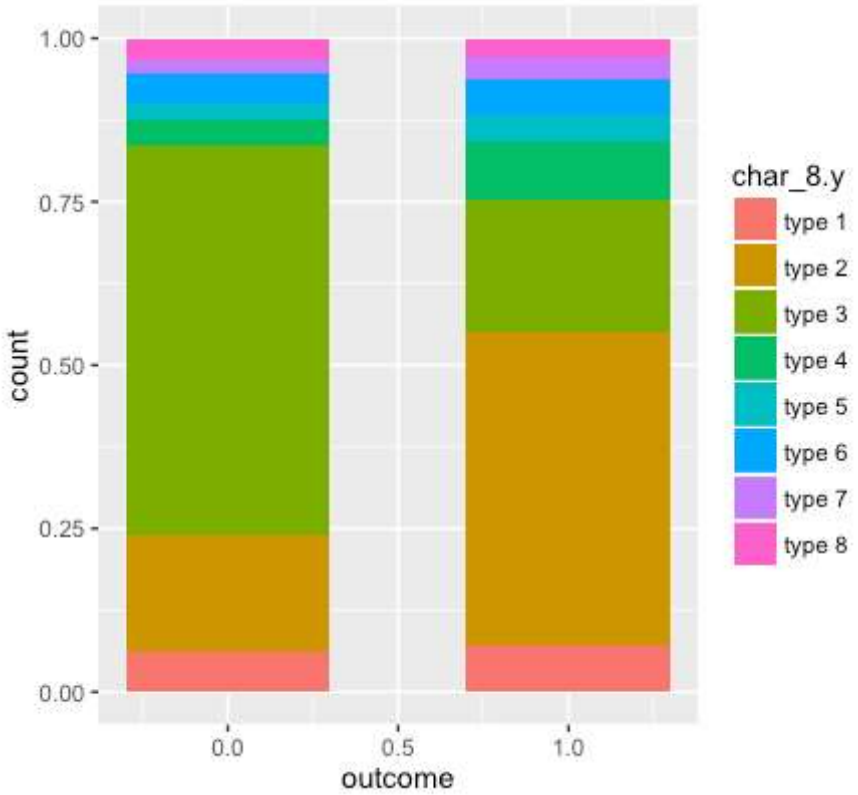
It turns out that this field is not that predictive. Most of the categories distributed the same on both outcomes.

Next one is char_6_y.



Some categories of this field is predictive such as type 2 and type 5.

Finally, let us look at char_8_y.



It is a highly predictive feature.

However, there may be some combined feature of those categorial fields which are highly predictive. For example, assume that when char_5_y is type 5 and char_15 is True, the outcome is always 1. And if char_5_y is type 6 and char_15 is False, the outcome is always 0. This is just my assumption which is not tested but the reason why I state it here is to illustrate that there must be some combined features on which a certain combination of several types in several fields would lead to a certain outcome 0 or 1. However, the combinations of types among those features is enormous so exploring which combination would work is a very tough process.

Reflection

In this project, I did some EDA on features and eliminated some that are unrelevant. Then I found the group_1 and date leakage on forum. It is a very interesting pattern and most of the test labels are correctly filled except 70k rows that are unaffected. Then I trained my own models, tuning them, ensembling them, and keep ensembling the predicted results with the leak fills.

There are several aspects I am here to share them. The leakage is the most interesting aspect in this project. My model performance was not good on CV(about 0.89 AUC) but by ensembling the leakage fills performance has increased to 0.989 AUC. What a huge improve!

I guess that the group_1 field is the company name. Instead of considering this pattern as leakge, I think it is true in real life that most of companies have steady outcome in their activities and some of them may change the outcome once during a period of time. This information is very useful if it is analyzed in depth.

The most difficult aspect of this project must be feature engineering. It is hard to decide which feature to remain and which to drop. It is also very hard to extract some relevant feature(so hard that I even failed to extract one).

For solving this problem, I think my final model is robust enough and it actually fits my expectation. However, it did not reach the benchmark I sat. Maybe the benchmark I sat was too high. If better feature engineering strategies was carried out and a better learning model is used, the performance would be increased.

Improvement

In this project, I think 2 main improvement can be made: better feature engineering and learning algorithm.

First, the learning algorithms I ended up using was Logistic Regression, Random Forest and Gradient Boosting. They are selected from a limited number of candidates. However, there may be some better algorithms for this project(such as xgboost) that I should use.

Second, feature engineering should be improved. I am not good at it and I need to learn more about it by reading books and learning from solutions.

After this competition is finished, #1 solution(made by Raddar) and #3 solution(made by Joshua Havelka) are posted in forum.

In raddar's solution, tf-idf is used in aggregating features and it was especially useful. He also considered leakage as a feature instead of an ensembling factor like what I considered. What's more, he did some cross validation strategic to make the CV results more reliable.

In Joshua's solution, he transformed group_1 variable to a continuous one where lower the value, higher probablity of positive outcomes. He also did some "leave one out" and "averaging outcomes" to char_10_x variable to make it much more predictable.

They are all good improvements that I could use to improve my model.

References:

1.https://www.kaggle.com/c/predicting-red-hat-business-value (https://www.kaggle.com/c/predicting-red-hat-business-value)

2.https://www.kaggle.com/dmi3kno/predicting-red-hat-business-value/redhat-hack-in-plain-english-eda (https://www.kaggle.com/dmi3kno/predicting-red-hat-business-value/redhat-hack-in-plain-english-eda)

3.https://www.kaggle.com/apapiu/predicting-red-hat-business-value/redhat-eda/notebook (https://www.kaggle.com/apapiu/predicting-red-hat-business-value/redhat-eda/notebook)

4.https://www.kaggle.com/c/predicting-red-hat-business-value/forums/t/23786/long-story-of-1-solution/136447#post136447 (https://www.kaggle.com/c/predicting-red-hat-business-value/forums/t/23786/long-story-of-1-solution/136447#post136447)

5.https://www.kaggle.com/c/predicting-red-hat-business-value/forums/t/23803/3-solution/136339#post136339 (https://www.kaggle.com/c/predicting-red-hat-business-value/forums/t/23803/3-solution/136339#post136339)

6.https://www.kaggle.com/ijkilchenko/predicting-red-hat-business-value/python-ver-of-group-1-and-date-trick (https://www.kaggle.com/ijkilchenko/predicting-red-hat-business-value/python-ver-of-group-1-and-date-trick)

7.http://scikit-learn.org/stable/documentation.html (http://scikit-learn.org/stable/documentation.html)

8.http://mlwiki.org/index.php/ROC_Analysis#AUC:_Area_Under_ROC_Curve (http://mlwiki.org/index.php/ROC_Analysis#AUC:_Area_Under_ROC_Curve)