Machine Learning Engineer Nanodegree

Capstone Project

George Seah July 15th, 2017

I. Definition

Project Overview

Domain Background of Machine Learining in manufacturing testing.

The proposal domain is in the manufacturing testing field. In most of the mass production manufacturing, testing are part of the manufacturing process which help to ensure product quality and reliability. At the same time, testing also involve higher cost to the manufacturer and time consuming. The proposed project is to examine the prediction of the testing time required based on the all the available features. The proposed project is based on the Kaggle competitions: [Mercedes-Benz Greener Manufacturing] (https://www.kaggle.com/c/mercedes-benz-greener-manufacturing/ (https://www.kaggle.com/c/mercedes-benz-greener-manufacturing/)). We will use the data from this competition to examine different machine learning method in making prediction.

Problem Statement

The problem we tried to solve is to predict the test time (the 'y') required based on all the featured provides (total 376 features).

Measurement

Based on the competition request, we found out that the prediction is scored based on R^2 value (Coefficient of Determination), so we will use the same metric to score and compare our model.

Anonimyzation of the dataset column name and data processing

As the dataset column name are anonymized, so we wouldn't be able to know the underlying meaning of each variable. It creates some issues for us to understand well the data, such as if we have done the Principal Component Analysis, we wouldn't be able to know what the first few component would actually mean. After some research in the community discussion, one of the way that many data science expert use in this competition are adding all the principal component and their original component as part of the features set and select the algorithm, such as gradient boosting or random forest or regularized regression like Lasso that would select through the features set that make best prediction.

Potential solution

The potential solution would be using gradient boosting regression tree. During the gradient boosting regression tree model building, I explore two different transformation --label encoding for all the multi-categorical variables. I also built a random forest as a benchmark model to compare the performance of gradient boosting. Besides, I also built a stacked model that combine lasso, gradient boosting and random forest for comparison. Apart from using the cross-validation R^2 score from our code, I also use Kaggle submission to cross-check the performance of the model.

Metrics

Selected Metric

Based on the competition request, the proposed metric is R^2. Based on the predicted value in test set to know the prediction capability of the model. Although the competition mandatory to use R^2, but I think it is worthwile to discuss about different choices of the measurement metrics and why R^2 is suitable.

Choices of measurement metrics

Based on the research in sklearn metrics[1], we can see that following are the list of metrics available:

1. Mean Absolute Error (MAE)
2. Mean Squared Error (MSE)
3. Median Absolute Erro(MedAE)
4. R^2
5. Explained Variance Score

For the coming discussion, I will focus the discussion on three of the most common metrics -- MAE , MSE, and R^2

1. Mean Absolute Error (MAE) -

   It takes absolute on the difference between predicted value and actual value. The key benefit of MAE would be interpretation as it measure the average eror across all the prediction. Shortfall of MAE is that it is using absolute which make it mathematically not a mathematically differentiable function as compared to Mean Squared Error (MSE)
2. Mean Squared Error (MSE) -

   It take squared value on the differences between predicted value and actual value. There are two key benefit would be the function is differentiable, which is very helpful if we use any Machine Learning Model utilized gradient descent to tune the model. Another feature of MSE is that it penalized on higher error since it take squared operation on the differences.
3. R^2$_{\text{2</sup> -}}$

   It capture the propotion of variance that explained by the model. One of the key feature of R^2 is that it could be negative value. It means that the total predicted error from the model are higher than the total variance of the data.

Since we are mainly focus on boosting or ensemble method, all 3 of them should be feasible for the modelling. If we plan to use any model that are using gradient descent algorithm such as Neural Network, I think MSE is more appropriate as it is mathematically differentiable.

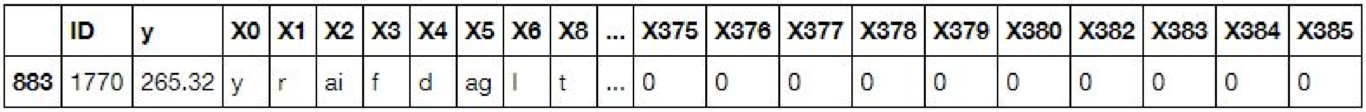
II. Analysis

(approx. 2-4 pages)

Data Exploration

The dataset provided from the competition has total 376 features. all of them are cateogorical. There are 8 variables are multi-categorical, where X4 has the smallest category count at 4 and X0 has the highest category count at 49.

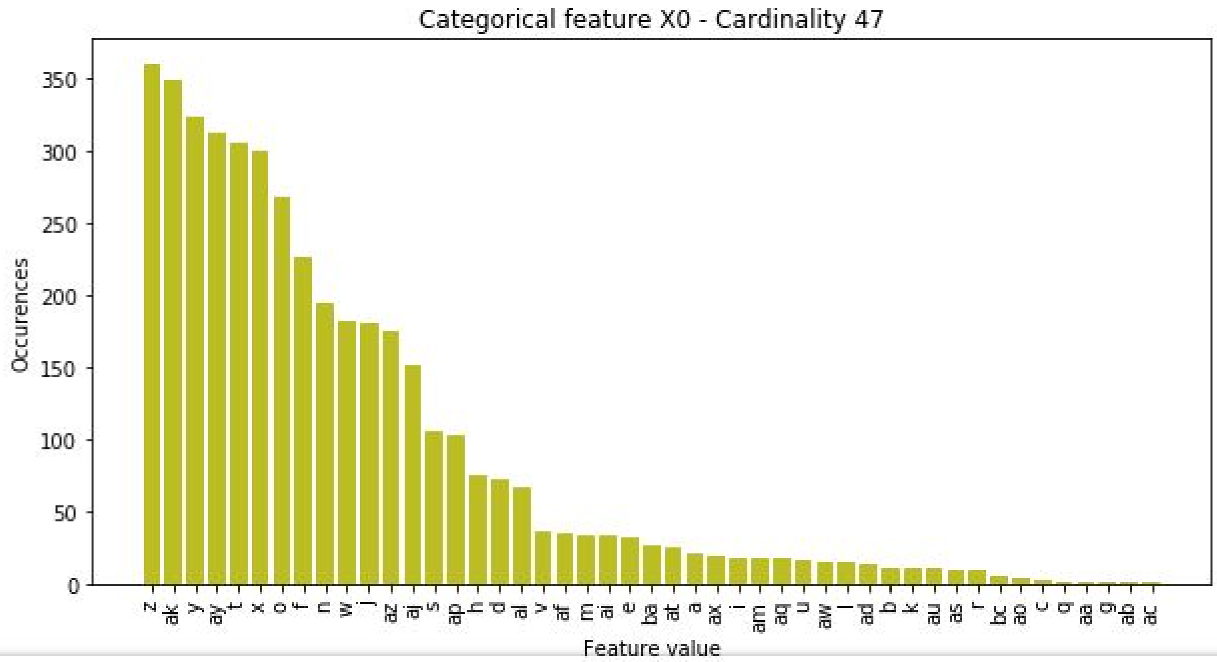
ColumnName	TestDataUniqueValue	TrainDataUniqueValue
X0	49	47
X1	27	27
X2	45	44
X3	7	7
X4	4	4
X5	32	29
X6	12	12
X8	25	25

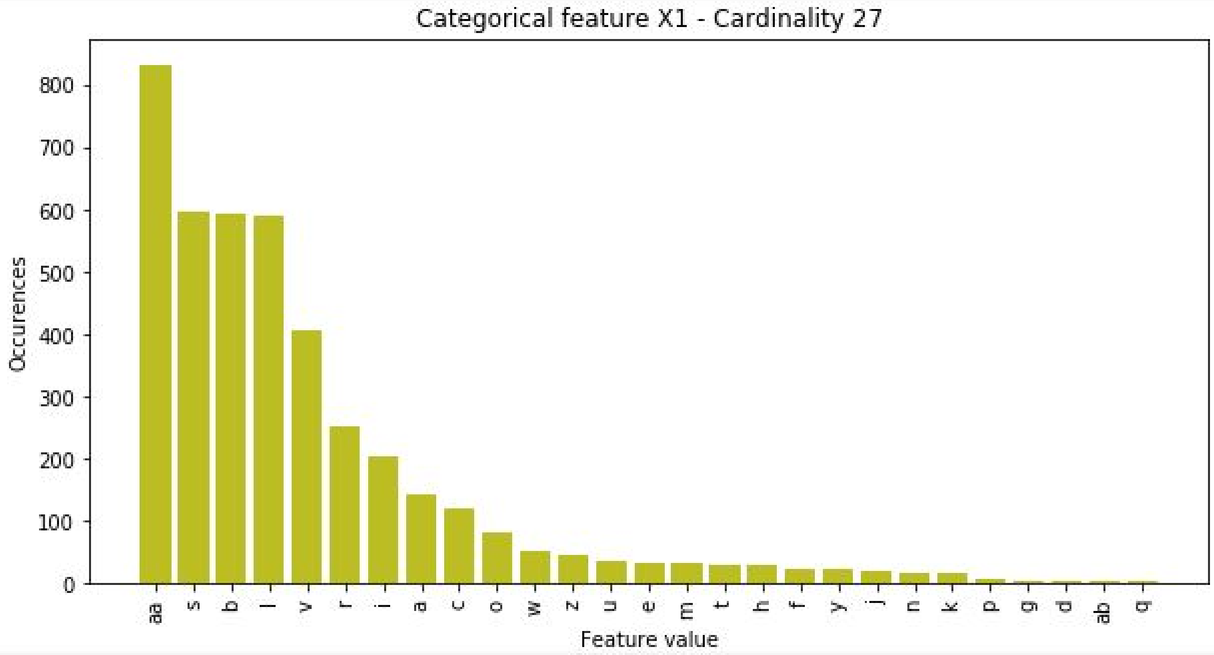


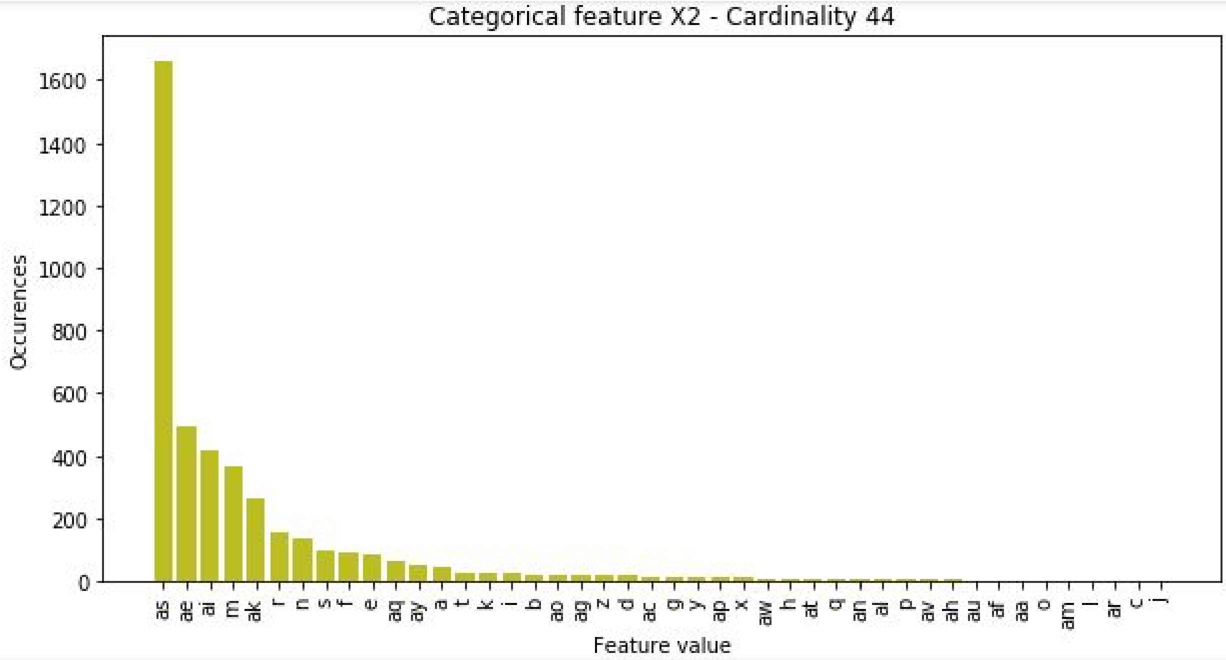
	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
883	1770	265.32	y	r	ai	f	d	ag	l	t	...	0	0	0	0	0	0	0	0	0	0

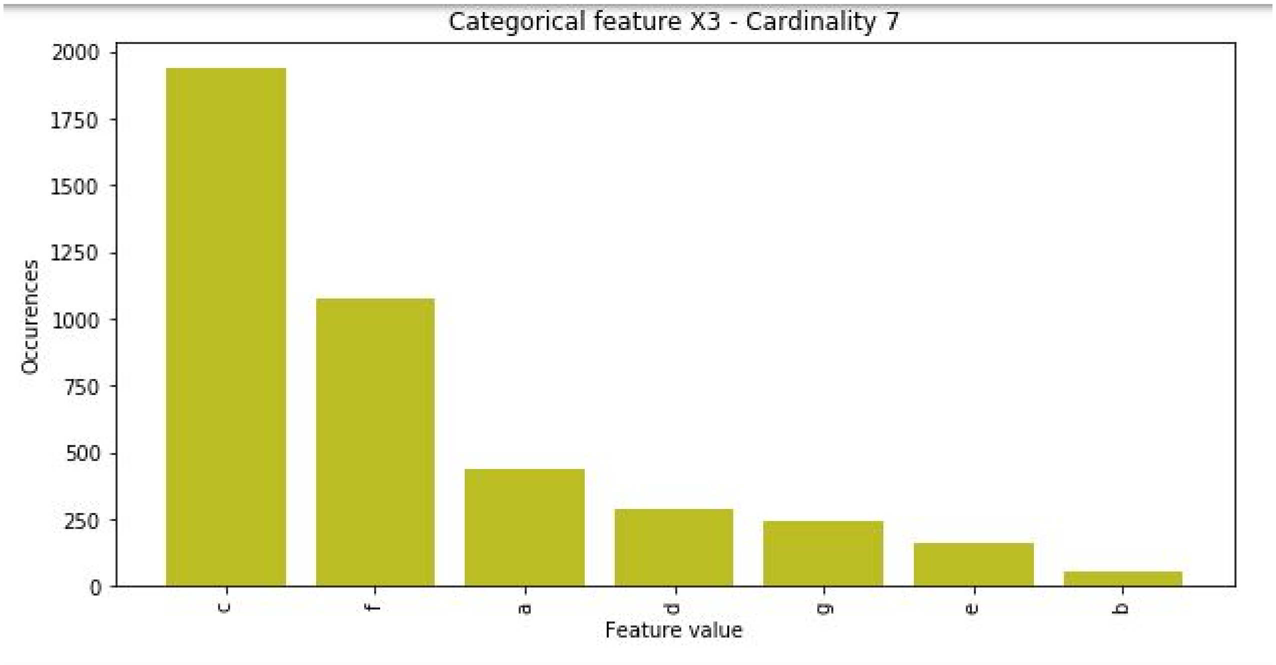
As we can see from table above, we can see that all features are anonymized.So, we wouldn't be able to know what would be the underlying meaning of each feature.

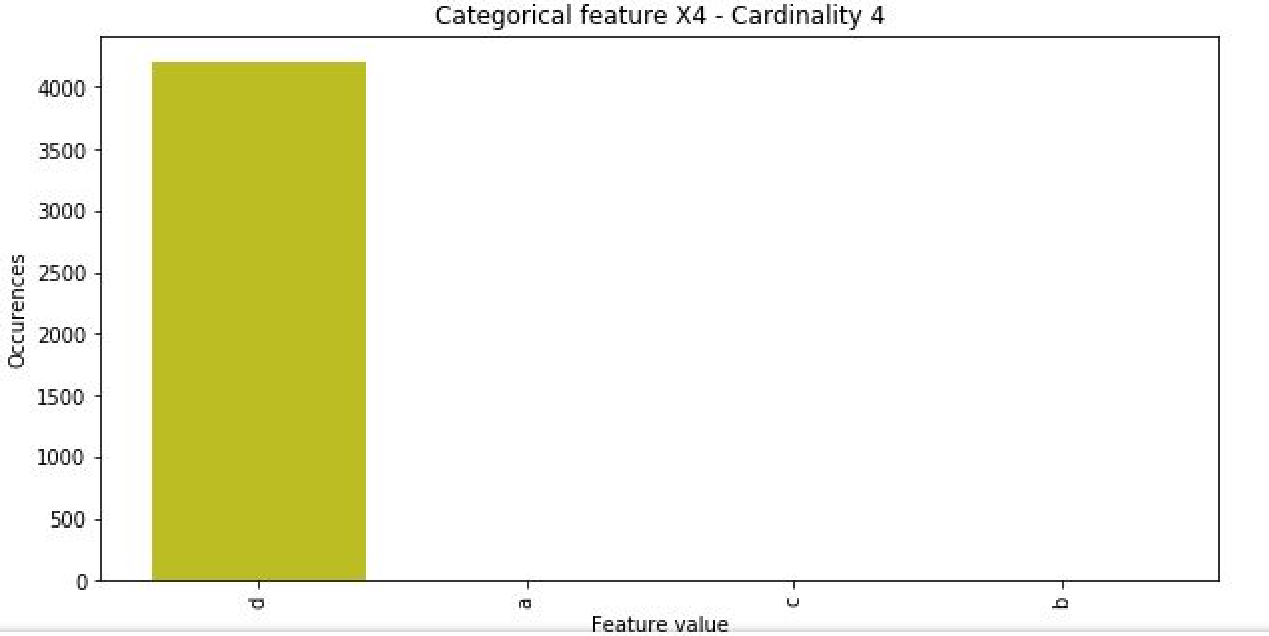
We plotted the multi-categorical variable frequency count, we can see that most of them are highly skewed towards few category with the extreme cases like X4, which has most of the sample with one value only.

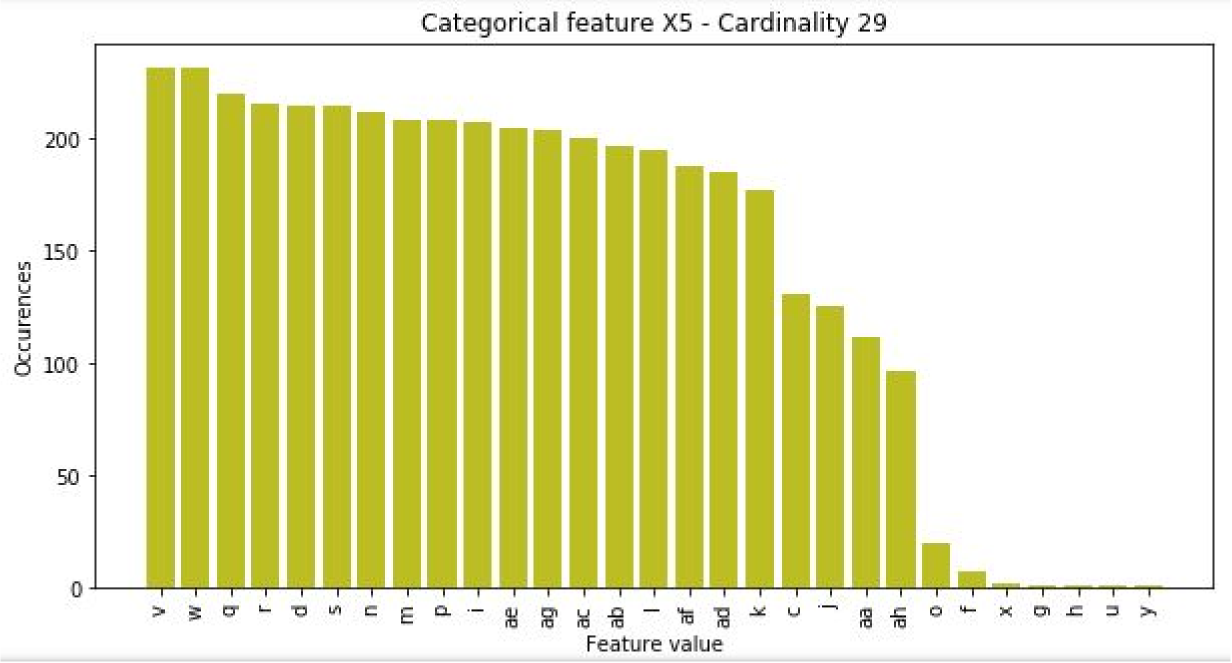


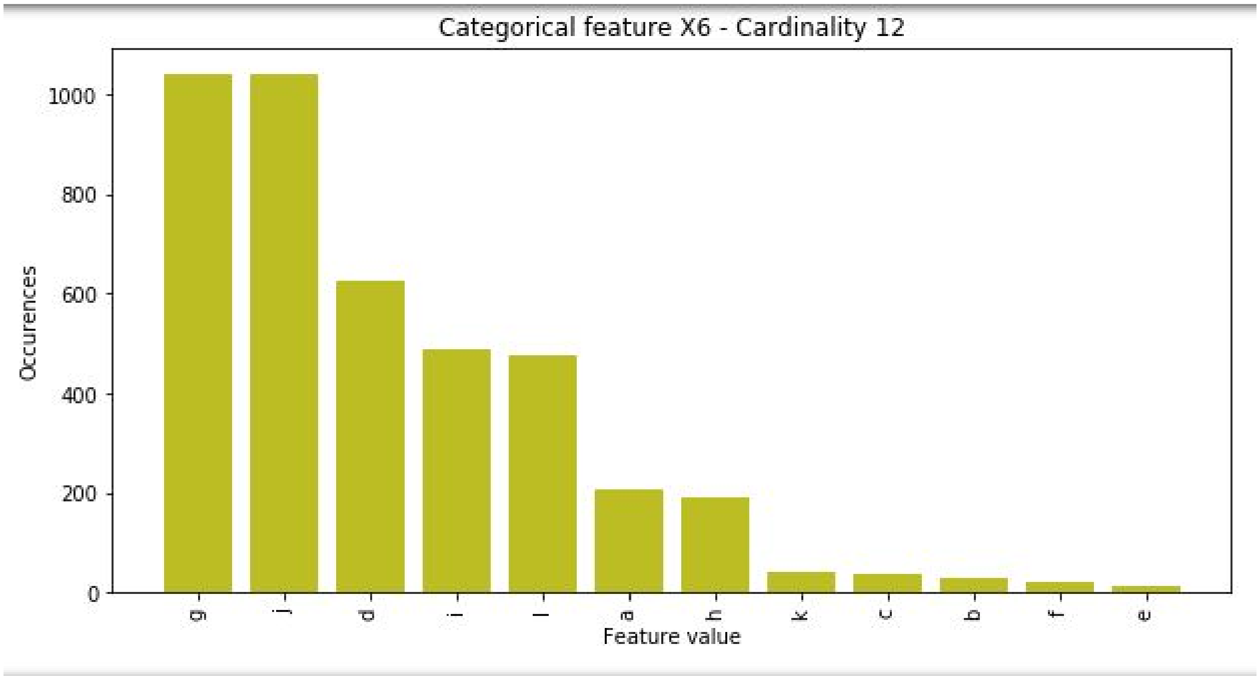

Categorical feature X1 - Cardinality 27
Occurences
800
700
600
500
400
300
200
100
0
aa
s
b
l
v
r
i
a
c
o
w
z
u
e
m
t
h
f
y
j
n
k
p
g
d
ab
q
Feature value

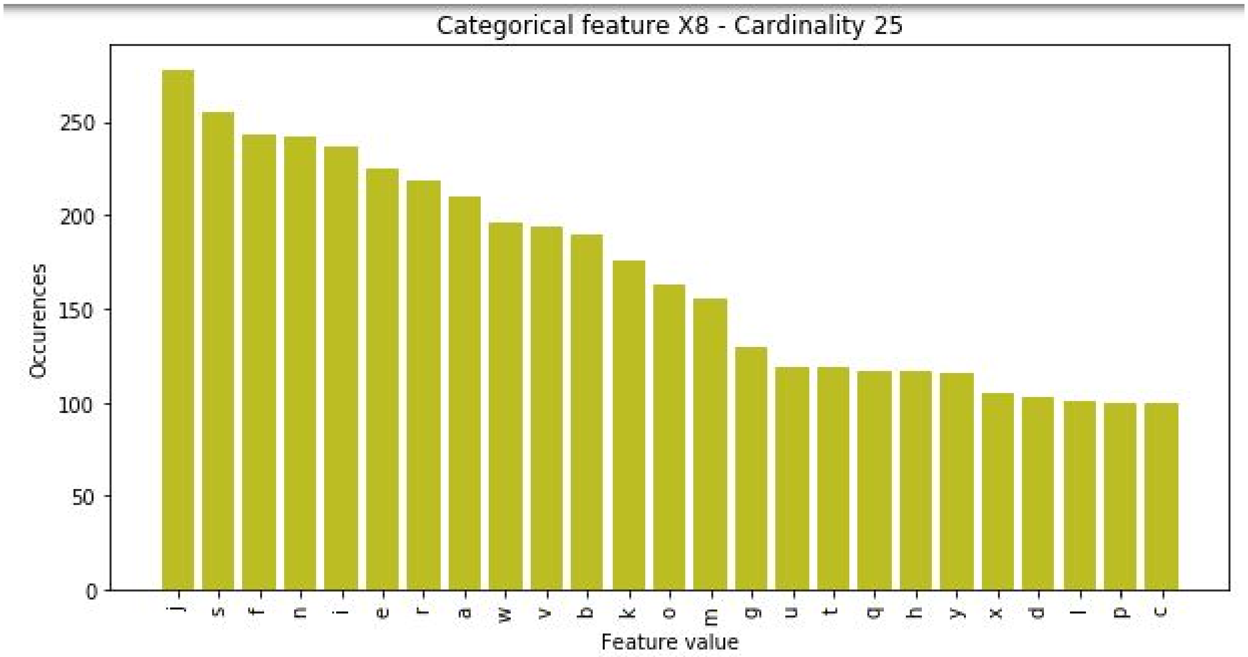

Categorical feature X2 - Cardinality 44
Occurences
1600
1400
1200
1000
800
600
400
200
0
as
ae
ai
m
ak
r
n
s
f
e
aq
ay
a
t
k
i
b
ao
ag
z
d
ac
g
y
ap
x
aw
h
at
q
an
al
p
av
ah
au
af
aa
o
am
l
ar
c
j
Feature value

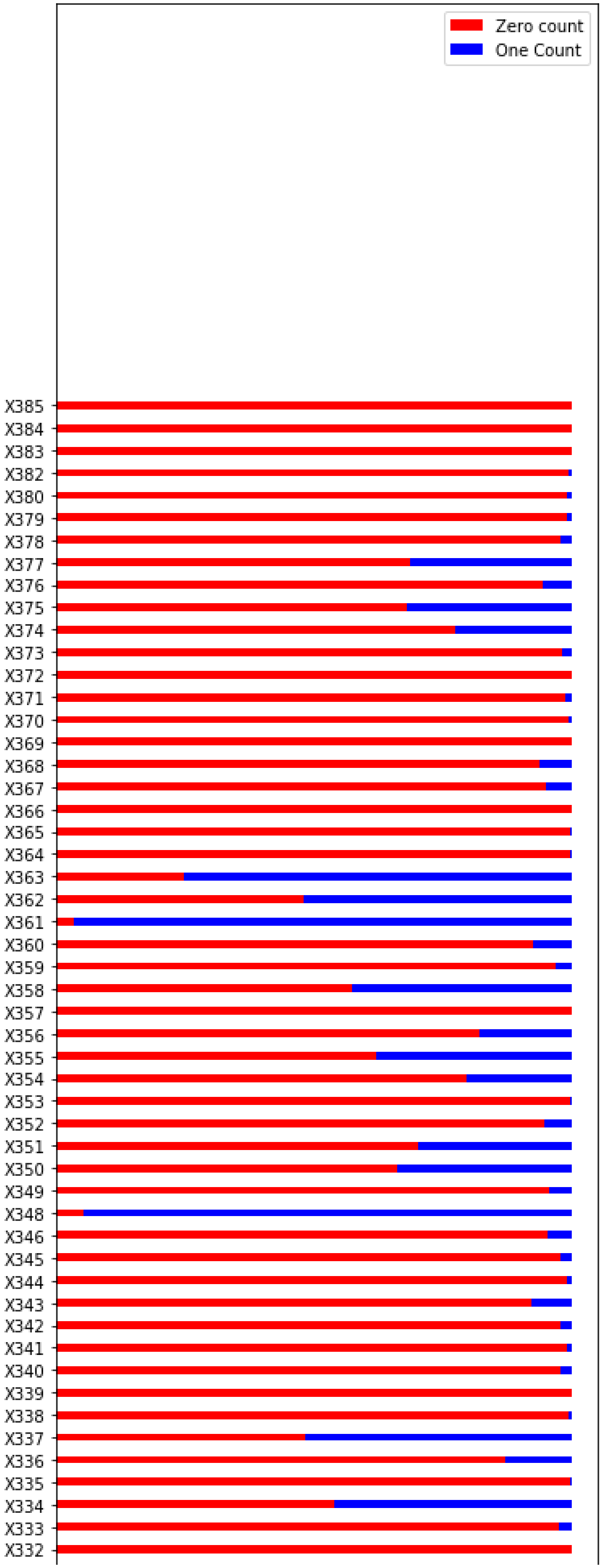

Categorical feature X3 - Cardinality 7
Occurences
0
250
500
750
1000
1250
1500
1750
2000
c
f
a
d
g
e
b
Feature value

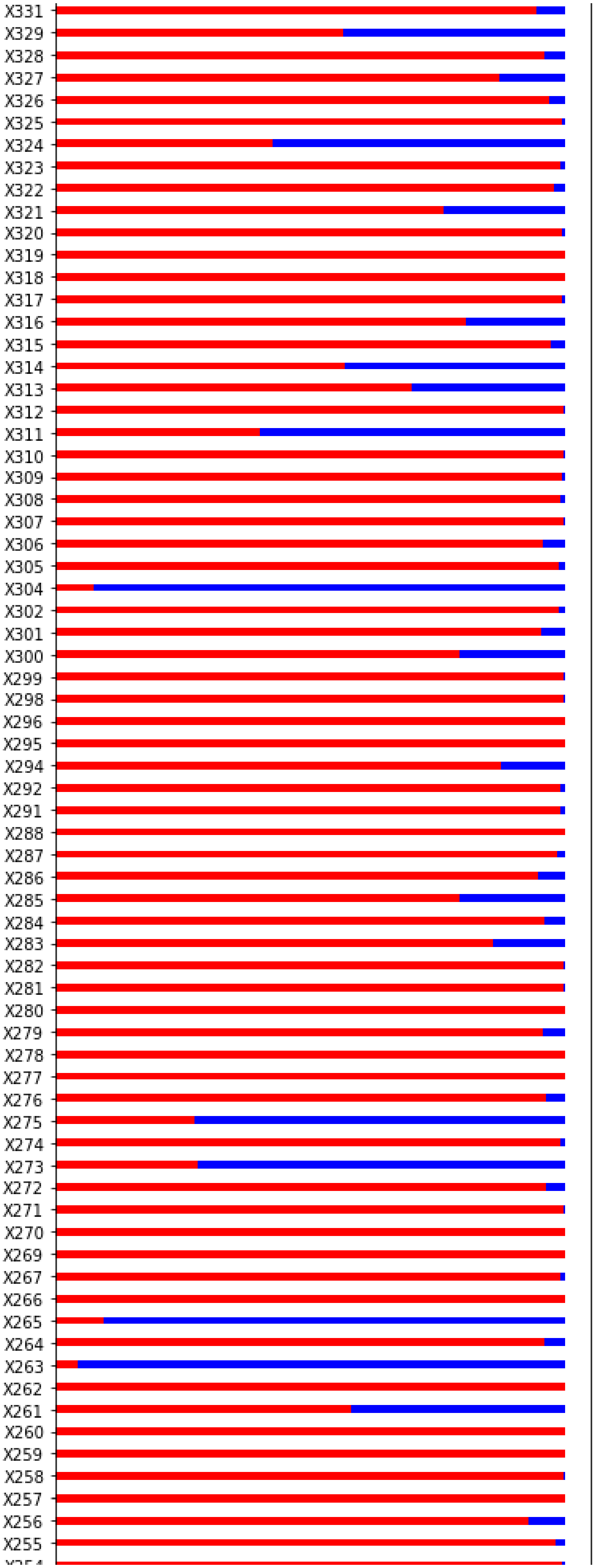

Categorical feature X4 - Cardinality 4
Occurences
0
500
1000
1500
2000
2500
3000
3500
4000
d
a
c
b
Feature value

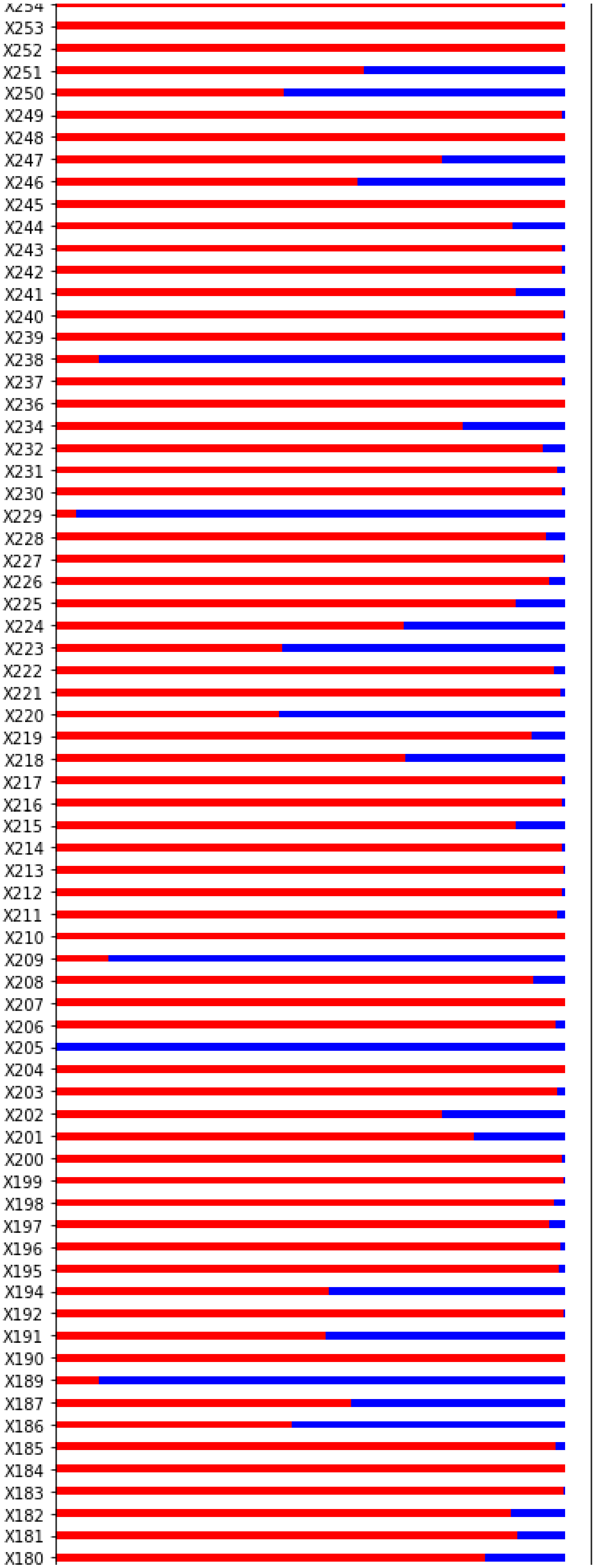

Categorical feature X5 - Cardinality 29
Occurences
0
50
100
150
200
Feature value
v
w
q
r
d
s
n
m
p
i
ae
ag
ac
ab
l
af
ad
k
c
j
aa
ah
o
f
x
g
h
u
y

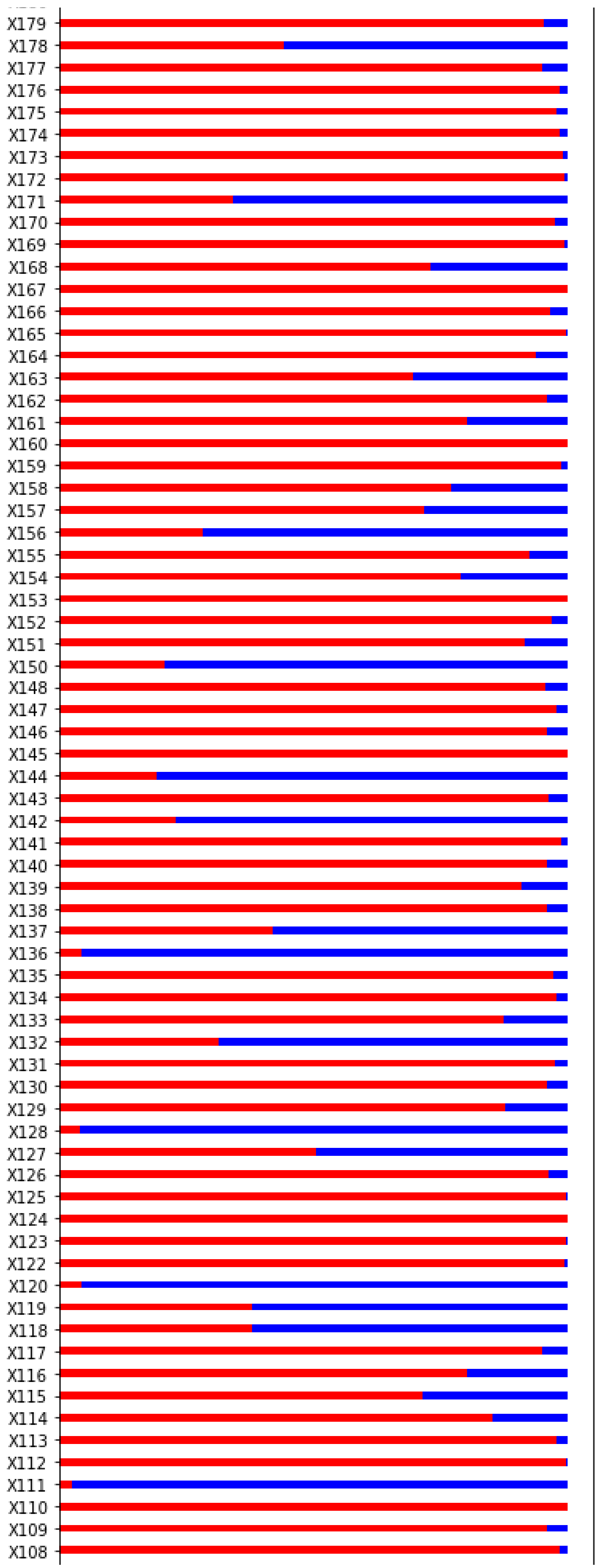

Categorical feature X6 - Cardinality 12
Occurences
0
200
400
600
800
1000
Feature value
g
j
d
i
l
a
h
k
c
b
f
e

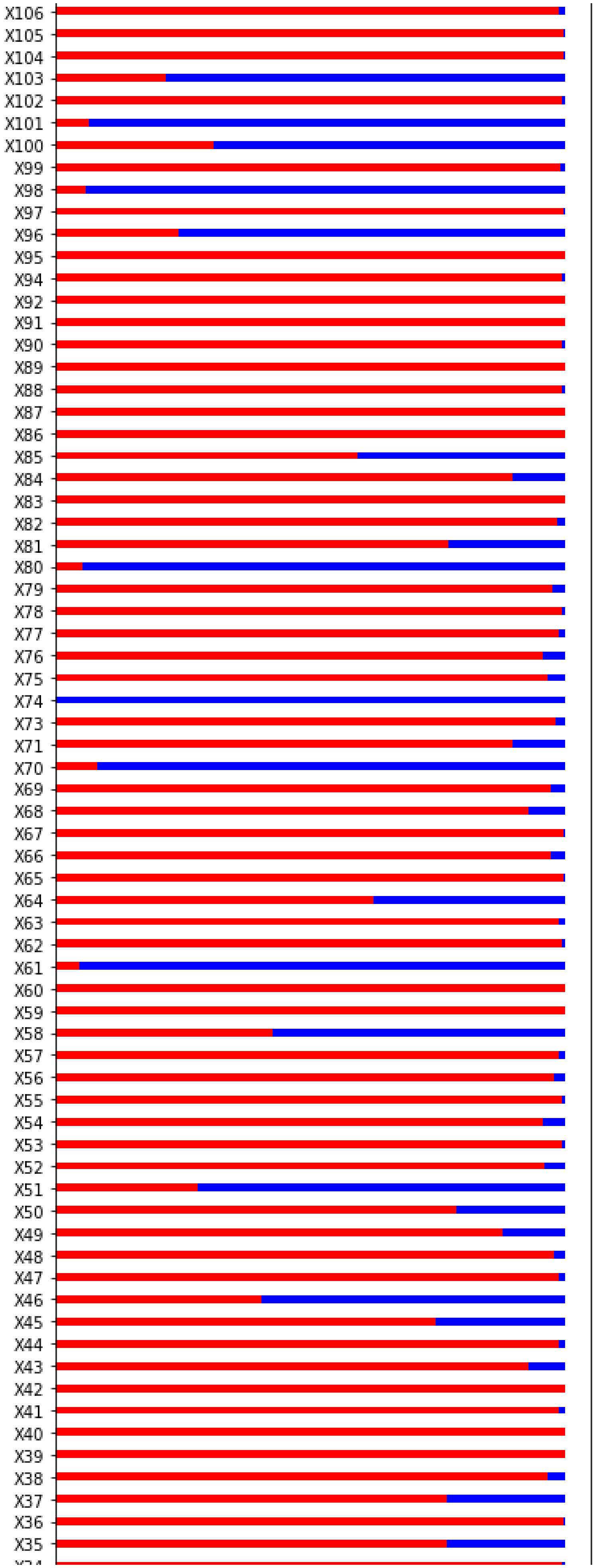

Categorical feature X8 - Cardinality 25
Occurences
0
50
100
150
200
250
j
s
f
n
i
e
r
a
w
v
b
k
o
m
g
u
t
q
h
y
x
d
l
p
c
Feature value

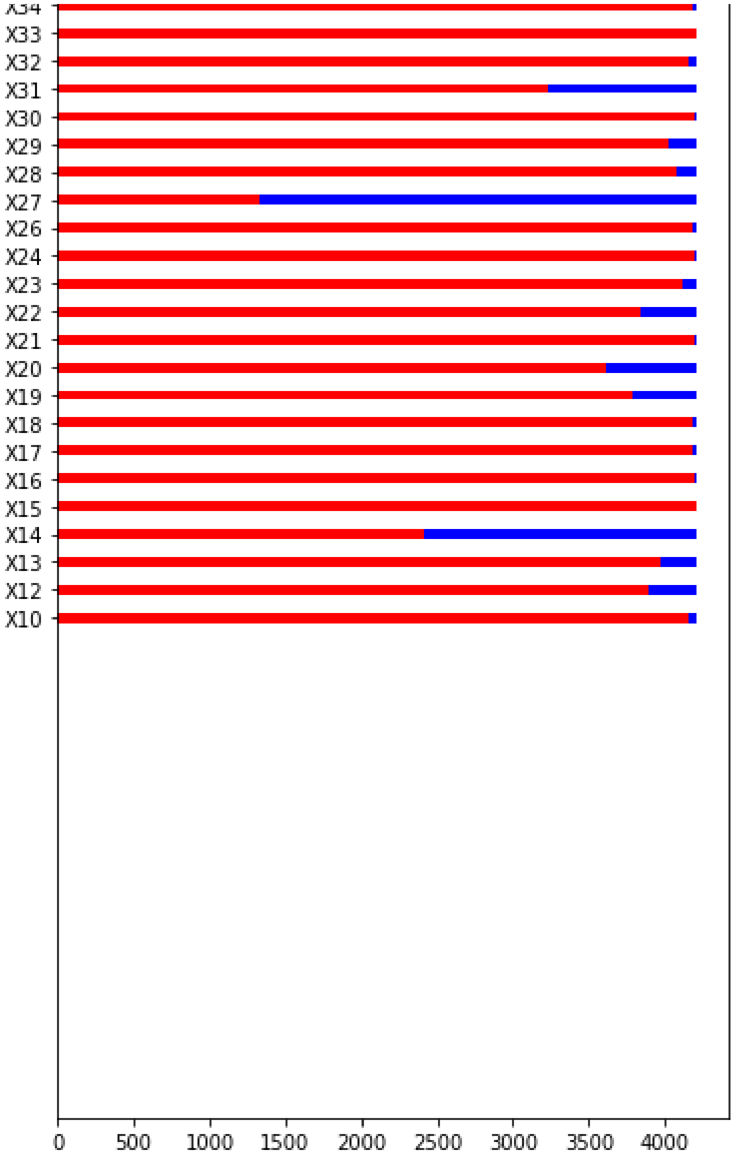

Zero count
One Count
X385
X384
X383
X382
X380
X379
X378
X377
X376
X375
X374
X373
X372
X371
X370
X369
X368
X367
X366
X365
X364
X363
X362
X361
X360
X359
X358
X357
X356
X355
X354
X353
X352
X351
X350
X349
X348
X346
X345
X344
X343
X342
X341
X340
X339
X338
X337
X336
X335
X334
X333
X332


X331
X329
X328
X327
X326
X325
X324
X323
X322
X321
X320
X319
X318
X317
X316
X315
X314
X313
X312
X311
X310
X309
X308
X307
X306
X305
X304
X302
X301
X300
X299
X298
X296
X295
X294
X292
X291
X288
X287
X286
X285
X284
X283
X282
X281
X280
X279
X278
X277
X276
X275
X274
X273
X272
X271
X270
X269
X267
X266
X265
X264
X263
X262
X261
X260
X259
X258
X257
X256
X255


X254
X253
X252
X251
X250
X249
X248
X247
X246
X245
X244
X243
X242
X241
X240
X239
X238
X237
X236
X234
X232
X231
X230
X229
X228
X227
X226
X225
X224
X223
X222
X221
X220
X219
X218
X217
X216
X215
X214
X213
X212
X211
X210
X209
X208
X207
X206
X205
X204
X203
X202
X201
X200
X199
X198
X197
X196
X195
X194
X192
X191
X190
X189
X187
X186
X185
X184
X183
X182
X181
X180


X179
X178
X177
X176
X175
X174
X173
X172
X171
X170
X169
X168
X167
X166
X165
X164
X163
X162
X161
X160
X159
X158
X157
X156
X155
X154
X153
X152
X151
X150
X148
X147
X146
X145
X144
X143
X142
X141
X140
X139
X138
X137
X136
X135
X134
X133
X132
X131
X130
X129
X128
X127
X126
X125
X124
X123
X122
X120
X119
X118
X117
X116
X115
X114
X113
X112
X111
X110
X109
X108


X106
X105
X104
X103
X102
X101
X100
X99
X98
X97
X96
X95
X94
X92
X91
X90
X89
X88
X87
X86
X85
X84
X83
X82
X81
X80
X79
X78
X77
X76
X75
X74
X73
X71
X70
X69
X68
X67
X66
X65
X64
X63
X62
X61
X60
X59
X58
X57
X56
X55
X54
X53
X52
X51
X50
X49
X48
X47
X46
X45
X44
X43
X42
X41
X40
X39
X38
X37
X36
X35



Count of binary value by each variable

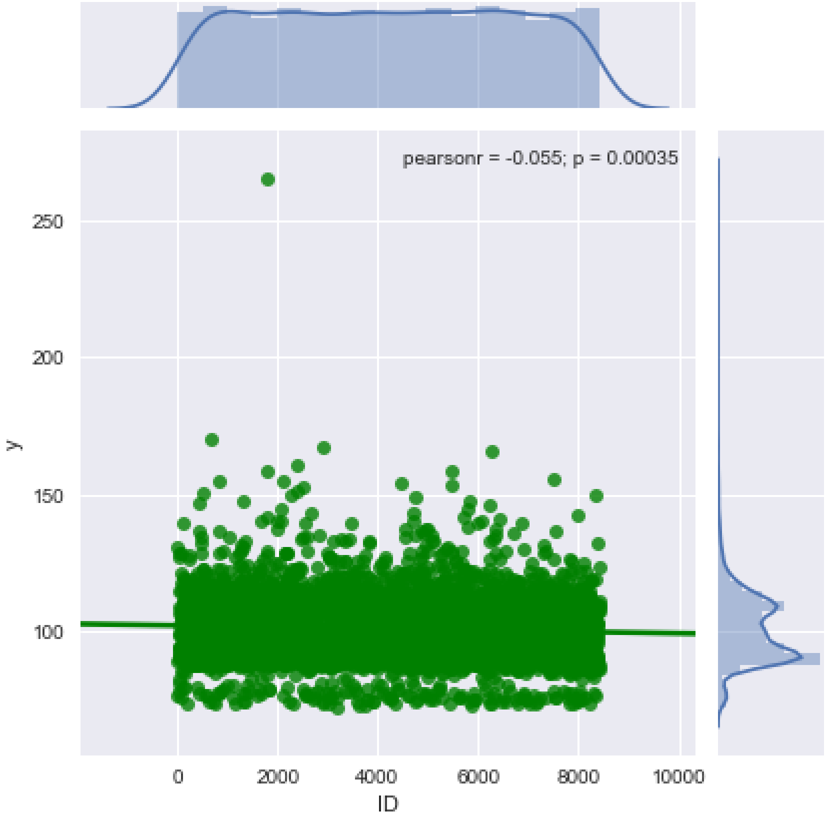
For rest of the binary variables, we can see that we have some cateogorical variable has only one value.

Finally, the datasets has split into two separate sets -- training set and test set. Eact set of data has total 4209 datapoint.

Exploratory Visualization and Analysis

Start from the "y"

We start by look at the y by plotting against the ID



Y vs ID joint plot

From the data, there is no obvious relationship between the ID and "Y". But, we can see that there is a downward trend as ID gets higher number.

Looking at the "x"

From the "x", we start by looking at the 8 multi-categorical variables. As we can see from the figure below, most of them are highly skewed towards few categories except X8.

We then look at rest of the binary variables, we also found some variables have only one value.

Our first intuition from such observation is that some of the variables that has only one categorical value or very small observations can be removed from the data to reduce the dimensionality.

As those variables least likely will have any value in training the models since they only has one value across the full spectrum of "y".

Additional feature engineering

Besides pre-processing the data set, we also applied various feature engineering technique to give more variable options to the model. Feature engineering techniqued used includes:

1. Principal Component Analysis (PCA)
2. Independent Component Analysis (ICA)
3. Gaussian Random Projection
4. Sparse Random Projection
5. Singular Value Decomposition

Algorithms and Techniques

Due to anonymized independent variable, the ensemble method such as xgboost or random forest would be ideal in such situation since we couldnt make any assumption (or bias) behind the data. So, we would need to find a algorithm that could help us select the important variables.

Testing various combination of data pre-processing and models

During the model building, we explore different approach as followings:

1. Xgboost with one hot encoding of the multi-cateogorical variables. In addition, we run principal component analysis(PCA) and independent component analysis (ICA) and append the new variables into the datasets.
2. Xgboost with label encoding of the multi-categorical variables. In addition, we run PCA and ICA and append the new variables into the datasets.
3. Stacked model with label encoding:
   A. Gradient Boosting Regression
   B. Lasso Regression
   C. Light GBM
   D. Random Forest In addition, we run PCA, ICA,Singular Value Decomposition, Gaussian Random Projection and Sparse Random Projection to enrich the datasets

1st approach -- Xgboost with one hot encoding

From the dataset, the first thought that I have is to one hot encoding all the multi-categorical variables. The key rationale comes from the observation above that most of the multi-categorical variables are highly skewed. Although one hot encoding would increase the dimensionality of the dataset, but we could also exclude some of the category that has no data at all. Most importantly, one hot encoding also has no asumption behind the ordinality of the category in each variables.

There are several hyperparameter that xgboost allow us to tuned,some of the key hyperparameter includes:

1. n_trees - number of trees
2. eta - learning rate
3. max_depth - maximum depth of a tree
4. subsample - Ratio of the instance are used for training. 0.5 means 50% of the training set's instances
5. objective - objective option tells the xgboost that whether we are working on regressin problem (reg:linear) or classification problem (reg:logistic). There are many other options available[3]

2nd approach -- Xgboost with label encoding

On the contrary of above methods, we do label encoding on the multi-categorical variables. It means that we assume some sort of ordinality inside the categorical variables. It would be a good model to run for comparing the result against the first approach

Same as above, we could tune several hyperparameter in Xgboost.

3rd approach -- Stacked model with label encoding

The 3rd approach is stacked model which we use a combination of different model to perform the prediction. There are several research paper showed that the stacked model could deliver a better prediction capability[4] [5].

Benchmark

Randomforest is used as the benchmark model to compare the model selected performed better or worse.

Randomforest is another popular decision tree ensemble method. It has better performance in preventing over-fitting compare to decision tree (single tree). It is relatively easy to tune the hyperparameter with python gridsearchCV.

The primary rationale behind using Randomforest as the benchmark model is that:

1. It gives relative good performance in prediction compare to decision tree
2. It is similarly flexible compare to our primary method Xgboost

III. Methodology

Data Preprocessing

When building the model, we tried several data-processing approach to see what would be the prediction results:

1. One hot encoding all the variables.
2. Label encoding all the multi-categorical varibles
3. Remove variables that has only one value (either has 0 or 1 only)
4. Include the ID or exclude the ID has one of the variable
5. performed Principal Component Analysis, Independent Component Analysis, Gaussian Random Projection, and Sparse Random Projection -- The purpose of performing above variables transformation technique is to find out new synthesized variable that may have significance predicton capability.

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
- *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
- *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

The gradient boosting implementation procedure as following:

1. Load the data into jupyter notebook
2. Pre-processing based on the different approach we want to test.
3. Import requested ML library, such as Xgboost
4. use gradient boosting cross validation method to find out the optimal number of tree based on R^2 score

5. Run the gradient boosting one more time based on the number of trees selected

The stacked model implementation procedure as following:

1. Load the data into jupyter notebook
2. Pre-processing based on the different approach we want to test.
3. Import requested ML library, such as Xgboost,Gradient Boosting Regressor, Lasso, RandomForestRegressor, lightgbm
4. Prepare helper function to implement the sub-models
5. Use xgboost as aggregator model to tune the final result of all the sub-model

We also run the benchmark model - Random Forest, the implementation as following:

1. Load the data into jupyter notebook
2. Pre-processing the data based on the final model approach selected
3. Import requested ML library, such as RandomForestRegressor
4. use gridsearchCV to tune the hyperparameter of Random Forest
5. Run the random forest one more time based on final hyper-parameter selected

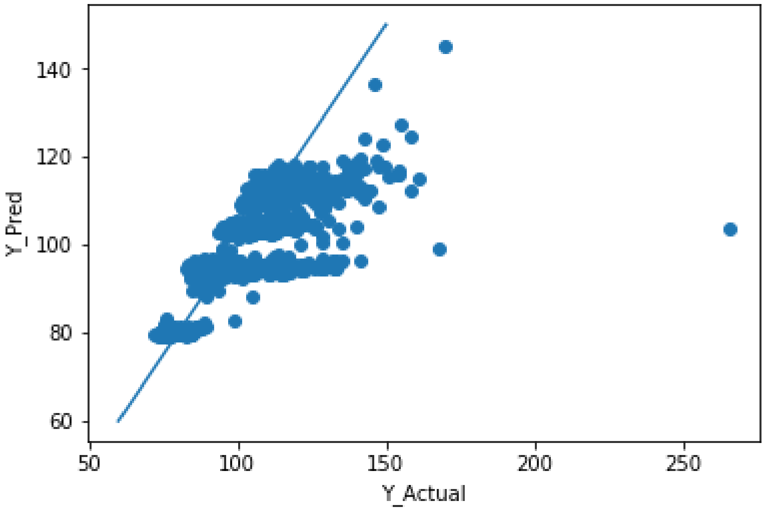
Refinement

Initial solution

My initial solution is to use xgboost with one-hot encoding for all the variables and excluding the ID. The rationale is that ID should be just an indexing of the each data row and should not be part of the predicting variable.

After running xgboost optimized with number of trees, the result is not good. Following are the score for the initial model:

Model Name	Cross Validation Score	Public Leaderboard	Private Leaderboard
Xgb with OHE	0.6365	-0.48733	-0.5714



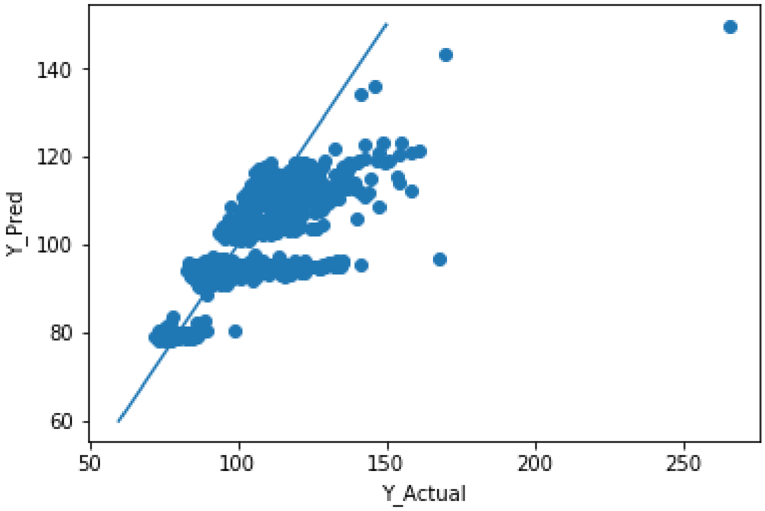
Y_Actual vs Y_Predicted in train data

Subsequent solution testing -- Xgboost with label encoding

Follow on the initial solution, I tried to use label encoding on the dataset with ICA and PCA.

The result as following:

Model Name	Cross Validation Score	Public Leaderboard	Private Leaderboard
Xgb with LE	0.6522	0.5648	0.5462



Y_Actual vs Y_Predicted in train data

As we can see from above, it significantly improve the R^2 score. From the above results, one of the possible reason that we would see such improvement is that the category within each variables seems to have ordinality and thus explain the improvement in predictability.

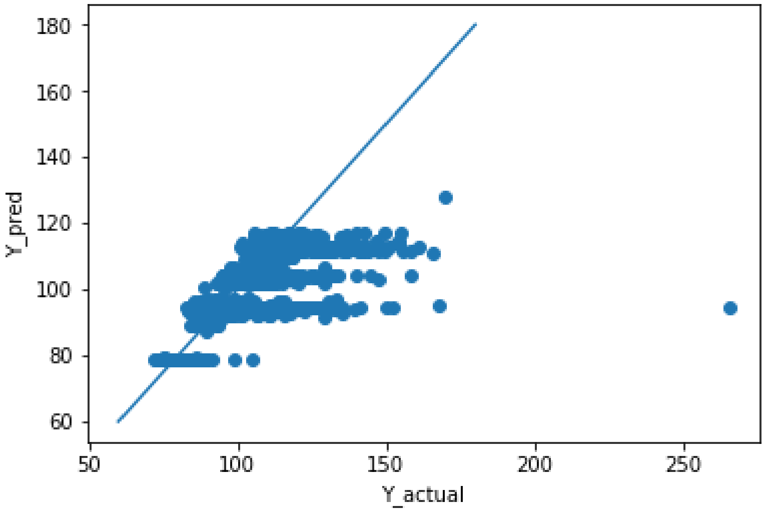
Subsequent solution testing -- Stacked model

The stacked model is combining various model for the predictions.

The result as following:

Model Name	Cross Validation Score	Public Leaderboard	Private Leaderboard
Stacked Model	0.65	0.5767	0.5458

One of the interesting observation of the stacked model is that we get better result in both cross validation and public leaderboard scores, but not the private leaderboard. From this observation, it may imply that we may have already over-fitted the data as compare to the xgboost.



IV. Results

Model Evaluation and Validation

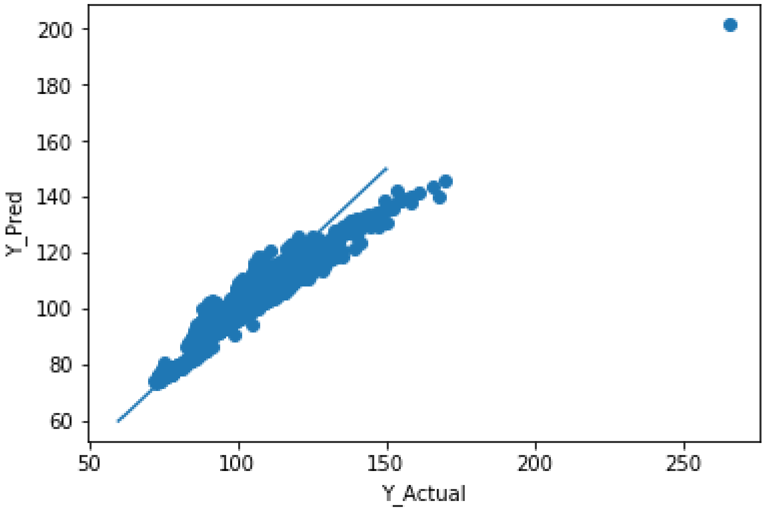
Based on the cross validation scores, we selected the xgboost with label encoding as the final model as it has the highest R^2 performance scores. It has the highest private leaderboard score, indicating that it has better generalization on the unseen data. Besides, it is generally more simpler to undersand compare to stacked model.

Justification

Benchmark model -- Random Forest

We run the benchmark model - Random Forest. The results shows that:

Model Name	Cross Validation Score	Public Leaderboard	Private Leaderboard
Random Forest	0.92	0.4963	0.4468



Y_Actual vs Y_Predicted in train data

Comparing to the benchmark model

Comparing our final model results as compare to the benchmark result, we can see that our final selected model performed relatively well compare to the benchmark model.

Based on the current private leaderboard No 1 team results at 0.5555, the final selected model is considered acceptable as a starting point model to know what would be the estimated test time would be given all the variables.

From the data, we've seen one outlier data point with extreme high test time. Further investigation in such outlier would help us to unveil insight in optimizing the model and the operational issue behind it.

One of the interesting observation is the Y_Actual vs Y_Predicted scatter plot for all 3 models vs benchmark model. We can see that our selected model has predicted lower value in "y" around 130 and above. Meanwhile, Random Forest seems do a better prediction on training set. But, the end validation score in private leaderboard shows that the result is lower compare to gradient boosting. It means that the random forest may already have overfitted the data.

V. Conclusion

Reflection

During this project, we've gone through the whole process of machine learning application :

1. Data visualization and understanding
2. Identify and implement data pre-processing
3. Identify models, test models, and implement them
4. Evaluate and select the final model

We managed to improve the result from negative R^2 value to better result by evaluating different model approach.

The interesting and difficult aspects of the projects is that I get to learn more about the modelling method I am interested in such as gradient boost (with XGboost) and Stacked model. Both of them are widely used in the kaggle community. The other interesting aspect of this project is that all the independent variable are anonymized. It means that we would need to rely on statistical method to understand more about the data. Method that we used Yes, the final model fit my expectatiosn in terms of the prediction outcome and the overall implementation process can be generalized for similar prediction problems.

This is also my first time participating the kaggle competition. It has been a fruitful experience on how much I learned during the competition.

Improvement

During the implementation and research, I found one of the new exploratory data analysis method called t-SNE(t-distributed Stochastic Neighbor Embedding) [6], which can help us to visualize high dimension data in lower dimension space. I believe we could make further improvement in data pre-processing if we able to extract some pattern from the data via this method.

Other than that, I would think we could run unsupervised learning such as k-mean clustering to see if have any natural clustering pattern for those datapoints with high test time(high "y").

Besides, Another approach that I think that may worth to test is deep learning. I believe it would be worthwhile compare its result versus xgboost.

Footnotes

1. [Scikit-learn Regresion Metrics]http://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics (http://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics))
2. Coefficient of Determination (https://en.wikipedia.org/wiki/Coefficient_of_determination)
3. [XGboost Parameters] (https://github.com/dmlc/xgboost/blob/master/doc/parameter.md (https://github.com/dmlc/xgboost/blob/master/doc/parameter.md))
4. [Why Stacked Model Perform Effective Collective Classfication] (https://kdl.cs.umass.edu/papers/fast-jensen-icdm2008.pdf (https://kdl.cs.umass.edu/papers/fast-jensen-icdm2008.pdf))
5. [Stacked Regression] (http://statistics.berkeley.edu/sites/default/files/tech-reports/367.pdf (http://statistics.berkeley.edu/sites/default/files/tech-reports/367.pdf))
6. [t-distributed Stochastic Neighbor Embedding Wikipedia] (https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding (https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding))

Besides above references, the coding and development has referred a lot of public sharing in kaggle such as :

1. [Kaggle - Mercedes-Benz Greener Manufacturing Kernels] (https://www.kaggle.com/c/mercedes-benz-greener-manufacturing/kernels (https://www.kaggle.com/c/mercedes-benz-greener-manufacturing/kernels))