Capstone Project Report

Machine Learning Engineer Nanodegree

Zhishuang Zhu
24 September 2019

Prediction of Weight Increase over A Year

1. Definition

1.1 Project Overview

Overweight has become a common problem nowadays. It is estimated 40% of adults are overweighed (World Health Oraganisation, 2018). It will be interesting to quantify these factors that have impact to overweight. Hopefully the outcome of the analysis will help to bring up some solutions to prevent us from overweighting.

1.2 Problem and Solution Statement

Whether one is overweight or not, is actually really straightforward to measure. Instead of predicting overweight, this project is trying to predict whether one will increase their weight or not within a year, given the lifestyle they currently have. Inputs are personal demographic, dietary and laboratory data from Kaggle[1] originally produced by CDC[2].

I also want to explore what are the key contributors to weight increase? To be more specific, what are the factors that will help with reduce risk of weight gain? What are the factors that lead to weight gain?
Our solution is mainly focusing on using Logistic regression as benchmark model, and XGboost as challenger model.
Top variables from each model are then used as predictive factors to assess their impact to weight gain.

1.3 Evaluation Metrics

There are few metrics that will be used to evaluate mode performance. It includes:

- **AUC**: area under the curve. It is important and most popular metrics for checking classification model's performance. It tells us how well the model is able to discriminate between classes (overweight and not overweight), where 1 to be the best.
- **F1 score:** F1 score is the harmonic average of precision and recall, 1 to be the best and 0 to the worst. F1 score will be used here instead of predictive accuracy. The reason is predictive accuracy is not a useful metrics for imbalanced dataset (Afonja, 2017).

[1] https://www.kaggle.com/cdc/national-health-and-nutrition-examination-survey

[2] Centers for Disease Control and Prevention

2. Analysis

2.1 Data Exploration and Visualization

2.1.1 Target Variable Exploration

Final dataset has 9,813 observations with 1,816 features; out of which 1,657 are 1 and 8,156 are 0. With sample weight[3] applied, the weight gain rate is 20%.

Fig1 shows the weight gain rate without applying sample weight is 19.7%, which reflects the whole population weight gain rate, while it is only 16.7% without applying sample weight. It indicates the importance of taking sample weight into account, in order to ensure our population is represented by this sample.

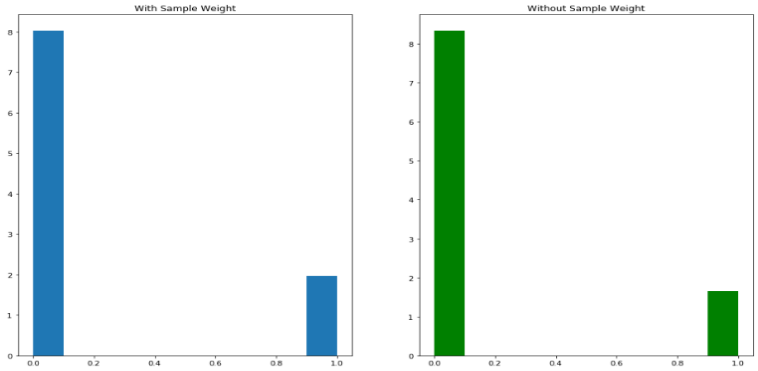


Fig 1 – Weight Gain rate with sample weight (Left) and without sample weight (Right)

2.1.2 Features in Nutshell

Broad Category	Number of Features	Features for modeling
Demographic (_g)	47	22
Examination (_e)	224	107
Diet (_d)	168	44
Lab (_l)	424	246
Questionnaire (_q)	953	354

Table 1 – Five categories of the data inputs. The 2nd column represents the original number of features, and the 3rd column displays the final features that are used for modeling after dimensionality reduction

In order to help reader gain intuition of how the raw data look like, a data snapshot is provided in *Table 2*, the 1st column is unique ID for each observation, 2nd column is sample weight and the 3rd column is our target to predict. The rest 5 columns are random representative of each category with special suffix to distinguish which category each comes from.

	ID	weight	weight_inc	MIAINTRP_g	OHX03CSC_e	DRD370IQ_d	LBXBPB_l	MCQ300C_q
0	73557	13281.237386	0	2.0	NaN	NaN	NaN	1.0
1	73558	23682.057386	1	2.0	NaN	NaN	1.69	1.0
2	73559	57214.803319	0	2.0	568.0	NaN	1.45	2.0
3	73560	55201.178592	0	2.0	NaN	NaN	0.37	NaN
4	73561	63709.667069	0	2.0	56.0	NaN	NaN	2.0

Table 2 - it is a sample of data with 5 rows and 8 columns

[3] **Sample Weights:** The 2-year sample weights (**WTINT2YR,** WTMEC2YR) should be used for all NHANES 2013-2014 analyses. Detailed instructions for combining datasets from previous NHANES cycles are provided in the NHANES Analytic Guidelines (https://wwwn.cdc.gov/nchs/nhanes/analyticguidelines.aspx). Note that medical dataset is not used here hence another sample weight (WTMEC2YR) will not be used.

2.1.3 Features Correlation

In the session of features exploration, main focus lies on features after dimensionality reduction (detailed in session 3.1.2). For simplicity, we just use one example to demonstrate the Pearson correlation[4] between features for examination inputs. For example, BMXRECUM_e (Recumbent Length (cm)) has negative Pearson correlation of -0.69 with MGXH1T1 (Muscle Strength Grip strength (kg), hand 1, test 1). It indicates our inputs still have certain level of correlation after dimensionality reduction.

This is one of the motivations to carry out further de-correlation exercise detailed in session 3.1.3.

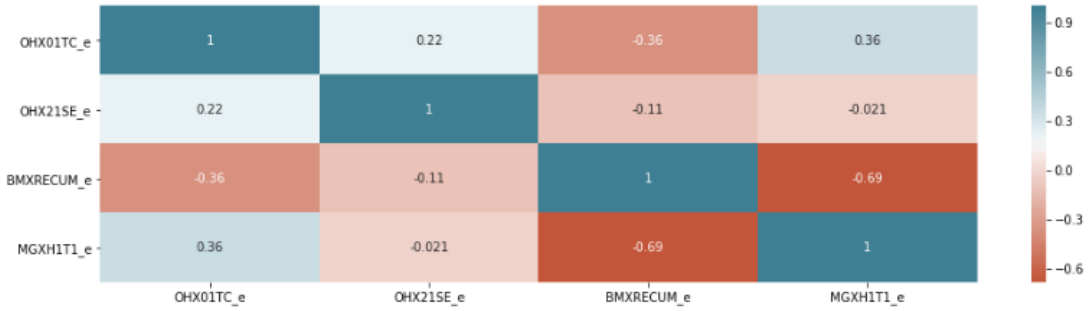


Fig 2 - Examinations inputs, including Blood pressure, body measures etc.

2.1.4 Other Properties

Missing: Many columns are having missing; the dataset will be left with no rows if we drop any observations that have missing. It is not recommended as missing also means information. In our case missing isreplaced by -99999. The reason why I go with extreme negative value is because no other value in the dataset will be the same as it, unlike replacing missing with 0. Worth mentioning that missing treatment is not required for XGboost but essential for our benchmark model.

Skewed value: Whether the feature is skewed or not has been looked into use statistic skewness. For normally distributed data, the skewness should be about 0. The *Fig3* indicates our dataset is considerably skewed. So standard normalization is applied.

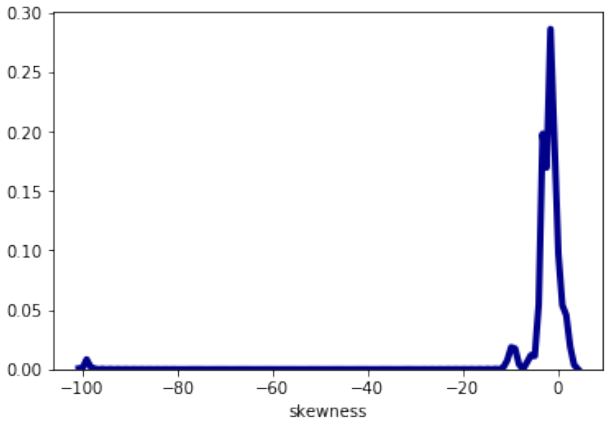


Fig3 – Skewness statistic distribution for top 100 features, 0 means normal distribution

[4] Standard correlation coefficient used as default method in python corr()

Outliers: Because XGboost is not sensitive to outliers (SYNCED, 2017), so this analysis is skipped.
Values that does not make sense: This has been further discussed in session 3.1.1 on categorical features.

2.2 Algorithm and Techniques

It is a classification problem. Target is defined as 1 if weight increase is larger than 5 pounds over a year. Information from the overall dataset will be used as inputs to predict how likely an individual will gain weight in the next one-year.

Logistic regression model is built as benchmark. Its strength lies on solving classification problems and very single features can be explained easily. It works well when features have linear relationship with target variable and are independent between each other.
The weakness of Logistic Regression is the fact it is not able to capture non-linearity and can perform poorly when there is high correlation between features. It also does not work too well with imbalanced dataset. The advantage and disadvantages of Logistic Regression makes it a good candidate as benchmark model.

XGboost is selected as the challenger model to be compared with benchmark model, because it has been widely recognized as the winner in many data mining challenges (Tianqi Chen, 2016). Tree boosting has been shown to give state-of-the-art results on many standard classification benchmarks (Li, 2010).

The strength of XGboost: Gradient boosting tree can learn non-linear relationship and fairly robust with outliers. What's more, it is designed to deal with missing values, so it is not always required to treat missing.

Weakenesses of XGboost: It can prone to over-fitting. Requires careful tuning of different hyper-parameters. It can be computationally intensive as each tree is built sequentially.

XGboost is extreme gradient boosting tree. Gradient Boosting Tree is a combination of many weak decision trees. Decision tree forms a model in a tree structure. It breaks down dataset into smaller and smaller subsets based on conditions on features. Final prediction is from the leaf node. Gradient Boosting is a way to weight more importance to the misclassified observations.
Gradient Boosting Tree builds decision trees sequentially. It starts with build a simple decision tree, which is just slightly better than random guess. Then second simple decision tree will be built on the error of first decision tree and repeat the same procedure for the rest of trees. The prediction is the majority vote from all decision trees.

The key advantage of XGboost over Gradient boosting tree are (Tianqi Chen, 2016):

- More scalable end-to-end tree boosting system, which means it can run much faster than existing GBM solution
- It can handle sparse data
- Parallel and distributed computing makes learning faster which enables quicker model exploration

2.3 Benchmark

Three logistic regression models are built. L2 regularization is used for all three models with C=2 which indicates less stronger regularization as compared to default setting (scikit learn).
Logistic Regression 1 is built based on all inputs of **807** features.
However, Logistic Regression requires variables to be independent from each other; while this is not a hard rule for tree-based approach. Because of it, a special treatment is done to remove features that are highlight correlated. They are then used to build **Logistic Regression 2**. We can see small performance uplift in Logistic Regression 2.

Library Sickit learn has limitation of interpret significance of features in the model. In order to overcome it, another library called statmodel is used to interpret feature importance.
From partial output below in *Fig4*, we can see not all features are statistically significant. So as a quick solution, only the features that have pvalue less than 0.05 are kept. The list of 10 features is listed in the *Appendix 6.3*, they are used to build **Logistic Regression 3**. However, the model performance has dropped significantly.
In order to fix this, stepwise approach shall be applied. But due to time constraint, this has not been looked into. This shall be considered for further improvement.

```
      Current function value: 0.306082
      Iterations: 35
                           Logit Regression Results
==============================================================================
Dep. Variable:             weight_inc   No. Observations:                 5887
Model:                          Logit   Df Residuals:                     5827
Method:                           MLE   Df Model:                           59
Date:                Tue, 17 Sep 2019   Pseudo R-squ.:                0.3342
Time:                        16:51:53   Log-Likelihood:               -1801.9
converged:                      False   LL-Null:                      -2706.2
                                        LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
SIAPROXY_g   -12.7149     59.979     -0.212      0.832    -130.271     104.841
DMDHRAGE_g    -0.4541      0.229     -1.986      0.047      -0.902      -0.006
DMDHREDU_g     0.3003      0.267      1.124      0.261      -0.223       0.824
DMDHRMAR_g    -0.3222      0.358     -0.899      0.369      -1.025       0.380
DMDHSEDU_g    -0.2633      0.087     -3.036      0.002      -0.433      -0.093
WTMEC2YR_g    -0.3334      0.254     -1.312      0.190      -0.832       0.165
INDHHIN2_g    -0.2028      0.389     -0.521      0.602      -0.965       0.560
PEASCTM1_e    -0.4276      0.222     -1.930      0.054      -0.862       0.007
BMXWT_e        0.0887      0.387      0.229      0.819      -0.669       0.847
OHX02CSC_e    -0.0028      0.130     -0.021      0.983      -0.258       0.252
OHX19CSC_e    -0.0111      0.120     -0.092      0.926      -0.247       0.225
DBD100_d       0.0734      0.096      0.768      0.443      -0.114       0.261
DR1TMOIS_d    -0.0062      0.189     -0.033      0.974      -0.377       0.365
PHAFSTHR.x_l  -0.2718      0.397     -0.685      0.494      -1.050       0.506
LBXHE1_l       0.2201      0.118      1.865      0.062      -0.011       0.451
PHAFSTHR.y_l   0.0524      0.082      0.640      0.522      -0.108       0.213
LBDPFUAL_l     0.0280      0.092      0.303      0.762      -0.153       0.209
LBDBMNSI_l    -0.1526      0.087     -1.754      0.079      -0.323       0.018
URDFLOW1_l    -0.2184      0.137     -1.590      0.112      -0.488       0.051
```

Fig 4 - Logistic regression output from statsmodel

From the AUC table (*Table 3*), F1 score table (*Table 4*), and ROC curve (*Fig 5*), Logistic Regression 2 will be used as our benchmark model as it achieves best model performance in terms of highest AUC and F1 Score in Test set.

	Train	Validation	Test
Logistic Regression 1	87.93	81.64	80.15
Logistic Regression 2	83.06	81.53	**82.02**
Logistic Regression 3	69.29	64.01	68.24

Table 3 - AUC Table for three logistic regressions

	Train	Validation	Test
Logistic Regression 1	0.58	0.5	0.49
Logistic Regression 2	0.52	0.49	**0.50**
Logistic Regression 3	0.42	0.34	0.41

Table 4 - F1 Score with 0.2 as threshold for three logistic regressions

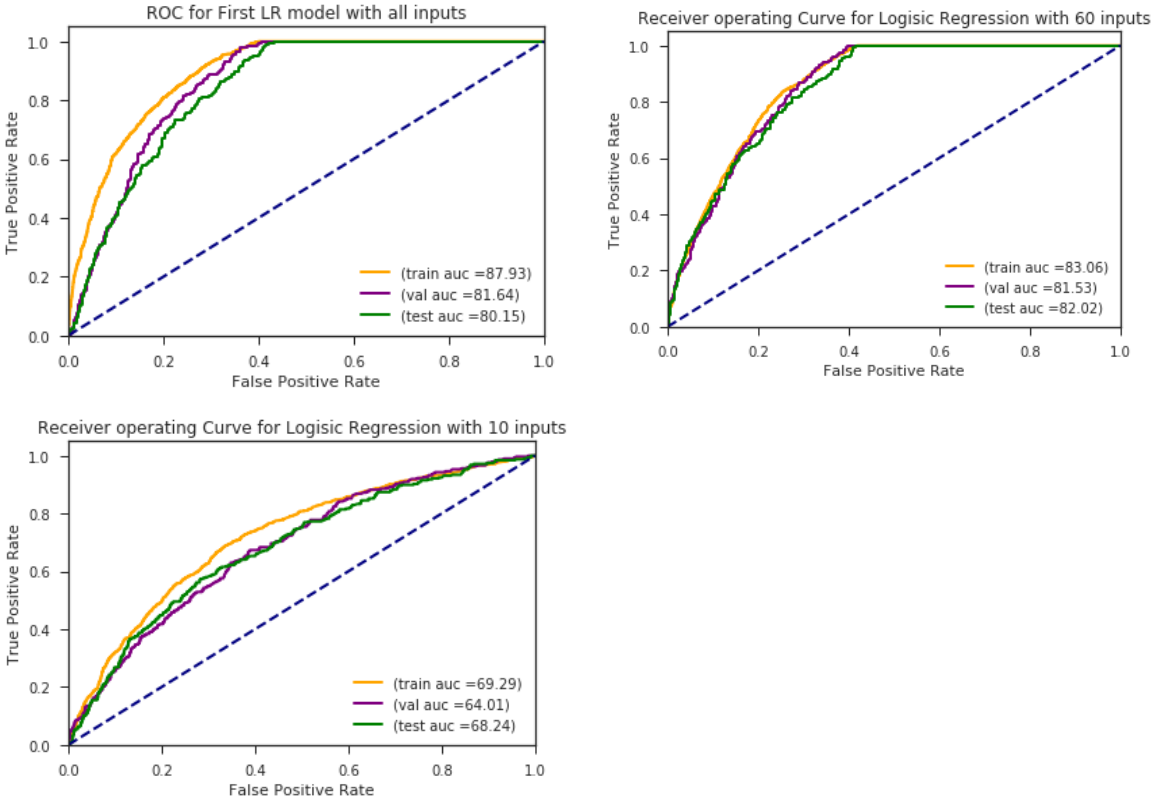


Fig 5 - AUC Curve for three Logistic Regression

3. Methodology

3.1 Data preprocessing

This phase is critical for modelling. The critical steps are treatment for missing values; converting categorical variables to numerical as most scikit learn machine-learning packages can't handle categorical variables. And lastly dimensionality reduction will be done in order to have useful variables as input for modelling.

3.1.1 Categorical Feature Conversion

There are 1,814 columns in total before dimensionality reduction, among them, 1,783 are numerical variables and 31 are categorical variables. Categorical variables are not predictive ones and don't seem to provide any predictive power to the model, so we are not going to use any of them (see Appendix 6.1). Hence there is no requirement to use LabelEncoder or OneHotEncoder for our case.

3.1.2 Dimensionality Reduction

In this process, we simply remove features that has univariate gini[5] less than or equal to 0.05. 1,006 out of 1,814 are dropped because of their low predictive power. As we can see from *Fig 6*, most features are having univariate gini between -0.05 to 0.3.

[5] https://en.wikipedia.org/wiki/Gini_coefficient

This might not be the best approach because some features, which have low gini by itself but can, add marginal predictive power to the model.

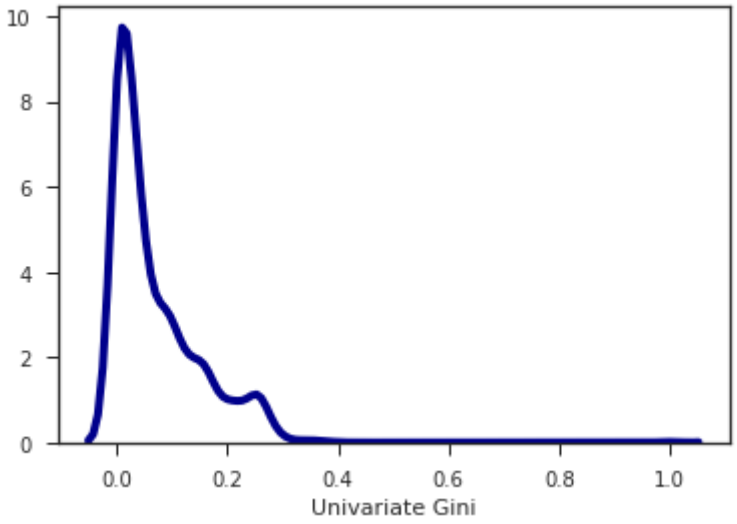


Fig 6 – Univaraite Gini distribution for all features

3.1.3 Remove highly correlated variables

This is the treatment especially for Logistic Regression as it requires variables to be independent from each other; while this is not a hard rule for tree-based approaches. Threshold of 0.5 is used to de-correlate inputs, and only 60 features are left. From *Fig 7* below, we can see all 60 variables have correlations less than 0.5.

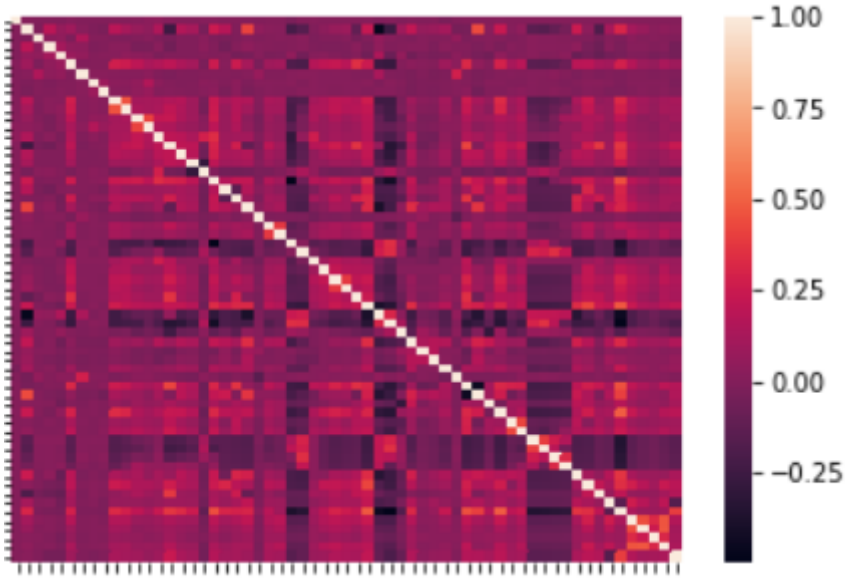


Fig 7 - Pearson correlation heat map

3.1.4 Normalization

From previous data exploration, the data is skewed. In order to ensure that each feature is treated equally when applying supervised learners, normalization is also applied. Applying a scaling to the data does not change the shape of each feature's distribution. All numerical inputs are normalized with mean equal to 0 and variance equal to 1.

3.2 Implementation

3.2.1 Default XGboost

I first of all tried XGboost with default hyper-parameters with random state to be 123456. From *Table 5 and Fig 8*, AUC for train dataset is 95, while it is only 85 for both validation and test dataset. It indicates the model is over fitting.

	Train	Validation	Test
AUC	95.30	85.05	85.74
F1 Score	0.73	0.52	**0.56**

Table 5 – AUC and F1 Score for defaut XGboost

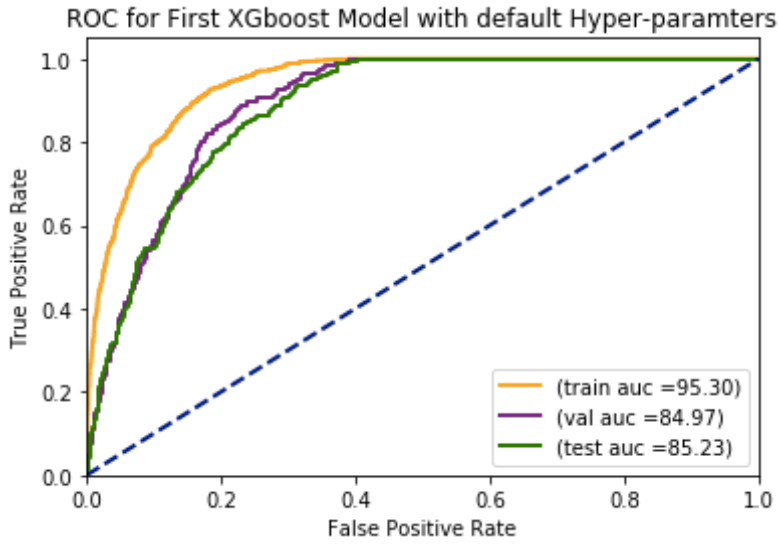


Fig 8- AUC curve for train, validation and test dataset

3.2.2 Manually Adjust Few hyper-parameters

In the 2nd trial, I adjusted some hyper-parameter settings in order to reduce over fitting. Taking some suggestions from Owen Zhang (Zhang, 2015) and XGboost Support Website (XBoost), the adjustment is made as show in *Table 6*:

Hyper-parameter	Reason
learning_rate = 0.01	Step size shrinkage used in update to prevents overfitting. The default value is 1, the smaller the more effective it is to prevent overfitting. However, also bear in mind, the smaller, the more time if will take to fit the model.
colsample_bytree = 0.4	It is the subsample ratio of columns when constructing each tree. Default value is 1, the smaller the value is, the less likely the tree is going to be overfitted.
subsample = 0.8	Subsample ratio of the training instances. Default value is 1, reducing it will take of part of the training set, which prevent overfitting.
n_estimators=1000	Number of trees in XGboost
reg_alpha = 0.3	L1 regularization term on weights. Default value is 0, Increasing this value will make model more conservative.
max_depth=4	The max depth of the tree. The larger this value, the more complex the tree is. Default value is 6, it is reduce to 4 in order to reduce the complexity.
gamma=10	Minimum loss reduction required making a further partition on a leaf node of the tree. The bigger gamma is, the more conservative the tree is.

Table 6 – Manually adjusted hyper-parameters, the other hyper-parameters are used as default

Table 7 shows the performance of XGboost model with reasonable inputs. As we can see, the adjusted XGboost Model is experiencing more severe over fitting issue and F1 score in Test dataset has drop slightly. However, the performance in test dataset does increase by 1 AUC point.

	Train	Validation	Test
AUC	98.51	85.13	**85.96**
F1 Score	0.73	0.51	**0.55**

Table 7 – AUC and F1 Score for altered XGboost

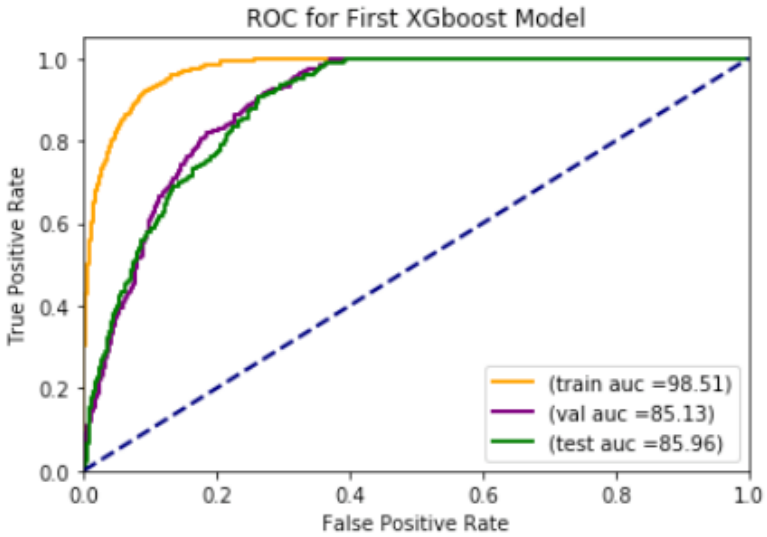


Fig 9- AUC curve for train, validation and test dataset

3.3 Refinement

3.3.1 Bayesian Optimization for Hyper-Parameter Tuning

Three approaches for hyper-parameter tuning have been looked into, including Grid Search and Random Search.

Grid Search is the most straightforward tuning approach. It added some automation to the manual trialing combinations of hyper-parameters. This approach will try every combination of the set of hyper parameters. But it is very time consuming and computationally intensive.

Random Search won't test sequentially all the combinations as Grid Search. The main advantage of Random Search over Grid Search is broader range of values or hyper parameters could be searched given same time or searching the same ones with much less time. However It does not take past trials into consideration for next search.

Bayesian optimization is used for hyper-parameter tuning here, because it builds a probability model of the objective function and use it to select the most promising hyper-parameters to evaluate in the true objective function. It updates objective function based on previous trials and that's why it find better hyper parameters in less time.

There are different libraries implemented for Bayesian Optimisation. Here I will only focus on **HyperOpt,** mainly for two reasons:

1). It is the most popular library for Bayesian optimisation

2). It can be twisted to incorporate with sample weights. Worth mentioning, I have also tried another library called Bayes Opt but it has issues with including sample weights.

HyperOpt use Tree Parzen Estimator (TPE) as Surrogate Model, which represents the distribution of a loss in terms of the value of a hyper parameter. It uses Expected Improvement (EI) as Acquisition Function that decide which the next points to explore are (maelfabien, 2019).

3.3.2 Initial Exploration of Number of Trees

The motivation of looking into number of trees in XGboost before dive into hyper-parameter tuning is because tuning job is computationally intensive. Number of trees plays a crucial role of time it will take to tune.

From the *Fig 10,* we can see the AUC in validation (amber coloured) stopped increasing after 150 trees. With this as a fact and also taking the running time into consideration, I changed the hyper-parameter of n_estimators to be 150 for my first hyper-parameter tuning.

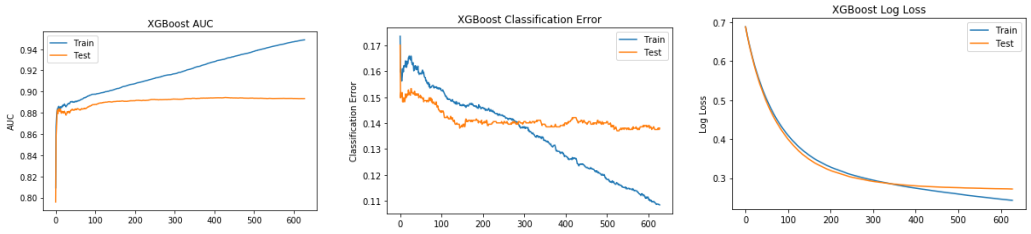


Fig 10 - Evaluation plot for train and validation dataset. From left to right, they are number of trees for AUC, classification error and log loss

3.3.3 First trial of Bayesian Optimization for hyper-parameter tuning

The setting of first Bayesian Optimization can be found in *Table 8*. The blue coded hyper-parameters are set to be tuned using Bayesian Optimisation. The amber coded ones are adjusted.

Hyper-parameter	Reason
learning_rate (0,1)	Step size shrinkage used in update to prevents overfitting. The default value is 1, the smaller the more effective it is to prevent overfitting. However, also bear in mind, the smaller, the more time if will take to fit the model.
colsample_bytree = 0.4	It is the subsample ratio of columns when constructing each tree. Default value is 1, the smaller the value is, the less likely the tree is going to be overfitted.
subsample = 0.8	Subsample ratio of the training instances. Default value is 1, reducing it will take of part of the training set, which prevent overfitting.
n_estimators=150	Number of trees in XGboost
reg_alpha = 0.3	L1 regularization term on weights. Default value is 0, Increasing this value will make model more conservative.
max_depth (1,8)	The max depth of the tree. The larger this value, the more complex the tree is. Default value is 6, it is reduce to 4 in order to reduce the complexity.
Gamma (0,20)	Minimum loss reduction required making a further partition on a leaf node of the tree. The bigger gamma is, the more conservative the tree is.

'tree_method': exact	Exact greedy algorithm is usually used for small to medium dataset like this one
min_child_weight (1,300)	Minimum sum of instance weight (hessian) needed in a child.

Table 8 - Setting of first Bayesian Optimization The rest of hyper-parameters are kept the same.

The best hyper-paramter tunred out to be {'eta': 0.04908639075639369, 'gamma': 11.945262328444612, 'max_depth': 5, 'min_child_weight': 173}

However, with this setting, the model turns out to be more over fitting with similar result in validation and test dataset, as can be seen in *Table 9* and *Fig 11.*

	Train	Validation	Test
AUC	99.9	84.7	85.27
F1 Score	0.89	0.52	**0.55**

Table 9 – AUC and F1 Score for XGboost with best hyper-parameters from Bayesian optimisation

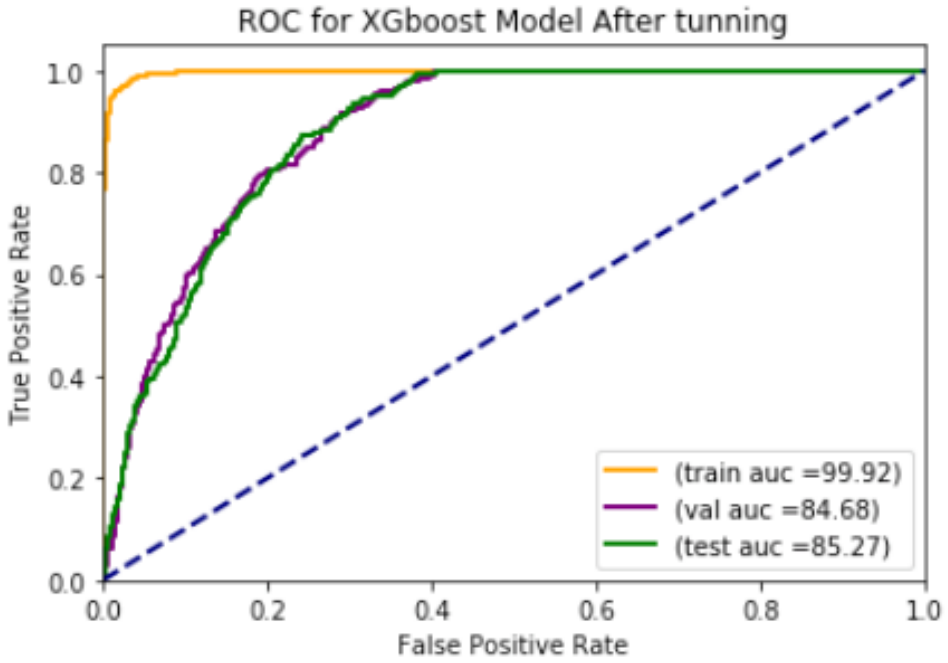


Fig 11- AUC curve for train, validation and test dataset

3.3.3 Final Trial of Bayesian Optimization for Hyper-parameter Tuning

I suspect the drop of evaluation result must because of reduction of number of trees, so I change it back to 1000. It is suggested to use **early stopping** to limit over fitting with XGboost (Brownlee, 2016). Hence train and validation dataset are both used to explore where is optimal early stopping for our XGboost. It is set to build 1000 trees and early stopping is set to be 200. It means if the validation auc does not increase after 200 trees, it will stop building more trees.

The new adjustment can be seen in *Table 10*:

Hyper-parameter	Reason
learning_rate (0,1)	Step size shrinkage used in update to prevents overfitting. The default value is 1, the smaller the more effective it is to prevent overfitting. However, also bear in mind, the smaller, the more time if will take to fit the model.
colsample_bytree = 0.4	It is the subsample ratio of columns when constructing each tree. Default value is 1, the smaller the value is, the less likely the tree is going to be overfitted.
subsample = 0.8	Subsample ratio of the training instances. Default value is 1, reducing it will take of part of the training set, which prevent overfitting.
n_estimators=1000	Number of trees in XGboost
reg_alpha = 0.3	L1 regularization term on weights. Default value is 0, Increasing this value will make model more conservative.
max_depth (1,8)	The max depth of the tree. The larger this value, the more complex the tree is. Default value is 6, it is reduce to 4 in order to reduce the complexity.
Gamma (0,20)	Minimum loss reduction required making a further partition on a leaf node of the tree. The bigger gamma is, the more conservative the tree is.
'tree_method': auto	Exact greedy algorithm is usually used for small to medium dataset like this one
min_child_weight (1,300)	Minimum sum of instance weight (hessian) needed in a child.
'early stopping parameter' 200	

Table 10 - The blue coded hyper-parameters are set to be tuned using Bayesian Optimisation. The rest of hyper-parameters are fixed

After running for 16 hours, with n_estimator = 1000, it recommended the best hyper-parameter to b e {'eta': 0.052064433112289046, 'gamma': 2.150605547313533, 'max_depth': 6, 'min_chil d_weight': 288}. However, from the ROC curve (*Fig 12*), the model is over fitted badly as trai n auc is 100.

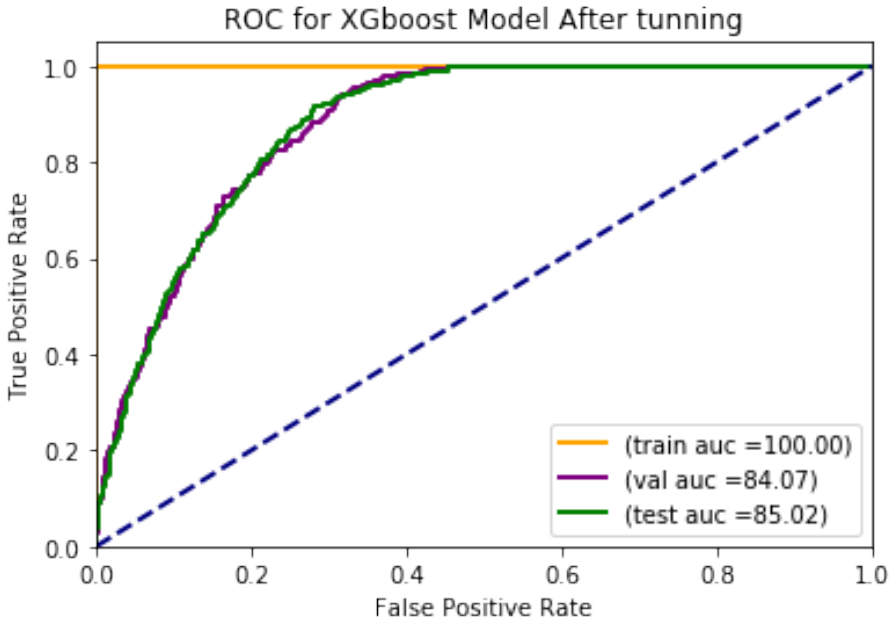


Fig 12- AUC curve for train, validation and test dataset

The evaluation curve is then plotted and further confirmed the number of trees should not be more than 200 and early stopping should not be too large either.

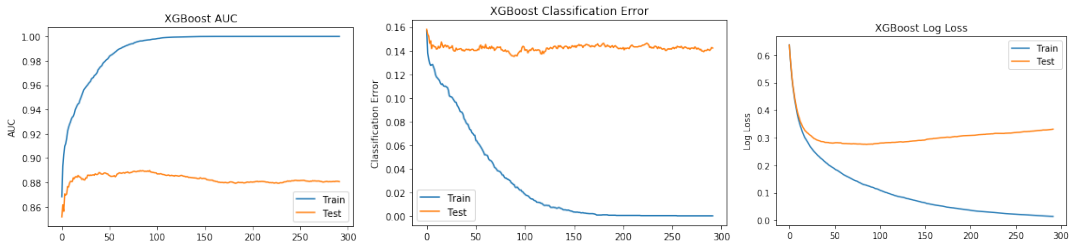


Fig 13 - Evaluation plot for train and validation dataset. From left to right, they are number of trees for AUC, classification error and log loss

I then manually keep number of tree to be 100 and change early stopping to be 10 with rest of the hyper-parameters stay the same. The final setting can been found in Table

Hyper-parameter	Reason
learning_rate	0.052064433112289046
colsample_bytree	0.4
subsample	0.8
n_estimators	**100**
reg_alpha	0.3
max_depth	6
Gamma	2.150605547313533
'tree_method'	auto
min_child_weight	288
'early stopping parameter'	10

Table 11 – Final Best setting for hyper-parameters for XGboost winner

	Train	Validation	Test
AUC	99.9	84.5	**86.2**
F1 Score	0.89	0.53	**0.58**

Table 12 – AUC and F1 Score for XGboost with best hyper-parameters from Bayesian optimisation

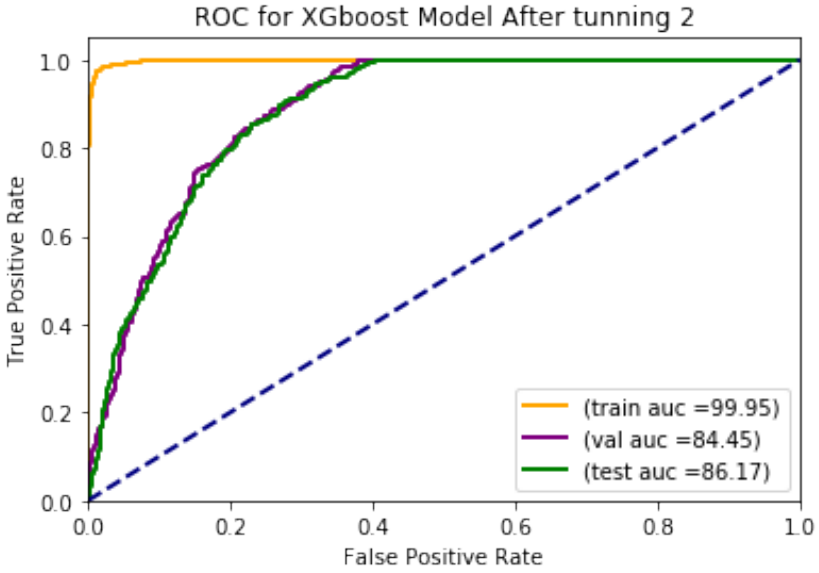


Fig 13- AUC curve for train, validation and test dataset

The challenge in this phase is it takes a lot of time to run. There was a time I had my tuning job run overnight and it only got 2% completed.
Also recording the results and version control can be also quite challenging, while I'm experimenting different settings.
It was also frustrating when after so many trials; the tuning does not improve the model performance much.

4. Results

4.1 Model Evaluation and Validation

Both benchmark and challenger model are quite robust as it achieves fairly good performance in validation and test dataset (See *Table 12 & 13*).

Though Machine-learning models are criticized as black box, there are still ways to unbox them. First of all we can look into feature importance as shown in the *Fig 14*. Feature importance indicates how valuable each feature is in construction of the boosted decision trees within the model.

It makes sense *During the past 12 months, {have you/has SP} tried to lose weight?* (WHQ070_g) is a strong predictor on whether one is going to gain weight or not in the next one year.
Job type is the second strongest predictor in the model.

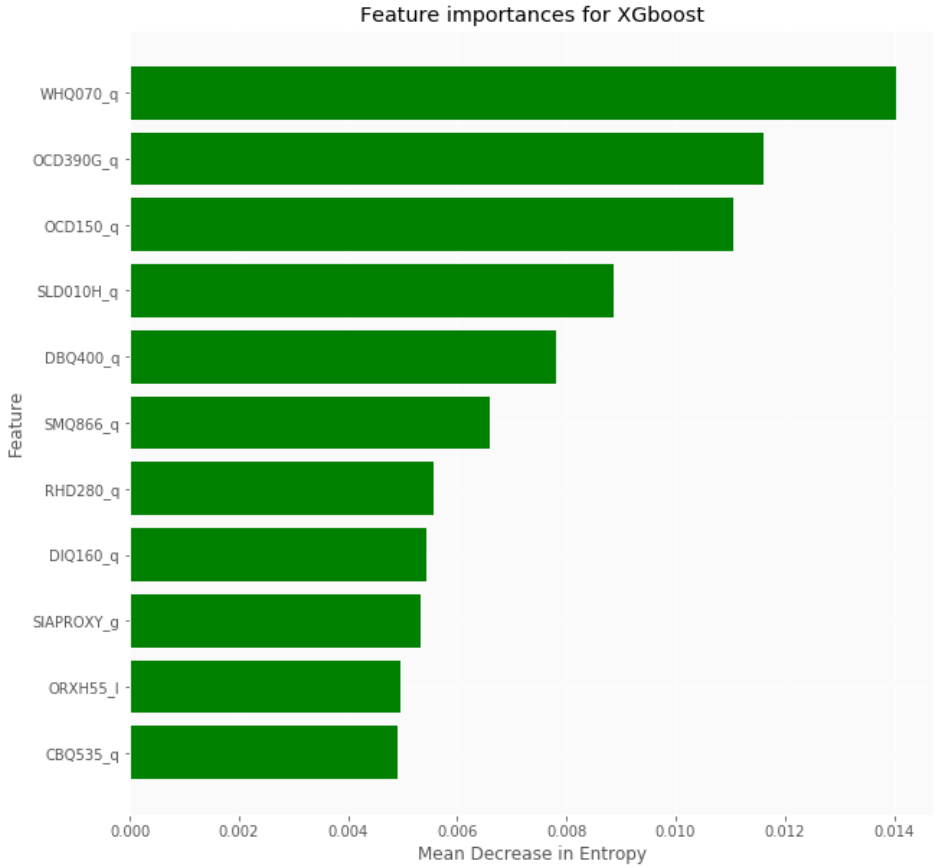


Fig 14- Feature importance of XGboost model

WHQ070_g	During the past 12 months, {have you/has SP} tried to lose weight?

OCD390G_g	Thinking of all the paid jobs or businesses {you/SP} ever had, what kind of work {were you/was s/he} doing the longest? (For example, electrical engineer, stock clerk, typist, farmer.)
OCD150_q	(SP Interview Version) In this part of the survey I will ask you questions about {your/SP's} work experience. Which of the following {were you/was SP} doing last week . . . (Family Interview Version) The next questions are about {your/NON-SP HEAD'S/NON- SP SPOUSE'S} current job or business. Which of the following {were you/was} {NON-SP HEAD/NON-SP SPOUSE} doing last week
SLD010H_q	The next set of questions is about your sleeping habits. How much sleep {do you/does SP} usually get at night on weekdays or workdays?
DBQ400_q	Does {your/SP's} school serve a complete breakfast that costs the same every day?
SMQ866_q	During the last 7 days, {did you/SP} spend time in a bar?
RHD280_q	{Have you/Has SP} had a hysterectomy that is, surgery to remove {your/her} uterus or womb?
DIQ160_q	{Have you/Has SP} ever been told by a doctor or other health professional that {you have/SP has} any of the following: prediabetes, impaired fasting glucose, impaired glucose tolerance, borderline diabetes or that {your/her/his} blood sugar is higher than normal but not high enough to be called diabetes or sugar diabetes?
SIAPROXY_g	Was a Proxy respondent used in conducting the Sample Person (SP) interview?
ORXH55_I	HPV Type 55
CBQ535_q	The last time when you ate out or bought food at a fast-food or pizza place, did you see nutrition or health information about any foods on the menu?
WH1060_q	[Over the last 2 weeks, how often have you been bothered by the following problems:] feeling bad about yourself - or that you are a failure or have let yourself or your family down?

Table 13 - XGboost Feature Dictionary

4.2 Justification

XGboost Model is marginally outperforming the benchmark model. It achieves 4 AUC point uplift and 0.08 uplift in F1 score in Test dataset.
Note Result in Train dataset is not the best to draw conclusion as our XGboost is still experiencing overfitting issue.

	Train	Validation	Test
AUC	99.9	84.5	**86.2**
F1 Score	0.89	0.53	**0.58**

Table 12 – AUC and F1 Score for XGboost with best hyper-parameters from Bayesian optimisation

	Train	Validation	Test
AUC	83.06	81.53	**82.02**
F1 Score	0.52	0.49	**0.50**

Table 13 – AUC and F1 Score for Best Benchmark Logistic Regression

5. Conclusion

5.1 Free form visualization – Unbox the Black Box

Apart from feature importance demonstrated in session 4.1, there are few other ways to unbox the black box.
The *Fig 15* shows how the feature value makes difference to the final output. It makes sense that the more often you are feeling bad, the more likely you are going to gain weight (WHQ060). However, it is interesting (or rather surprising) to see from SLD010H_q, the more you sleep, the more likely you are going to gain weight.

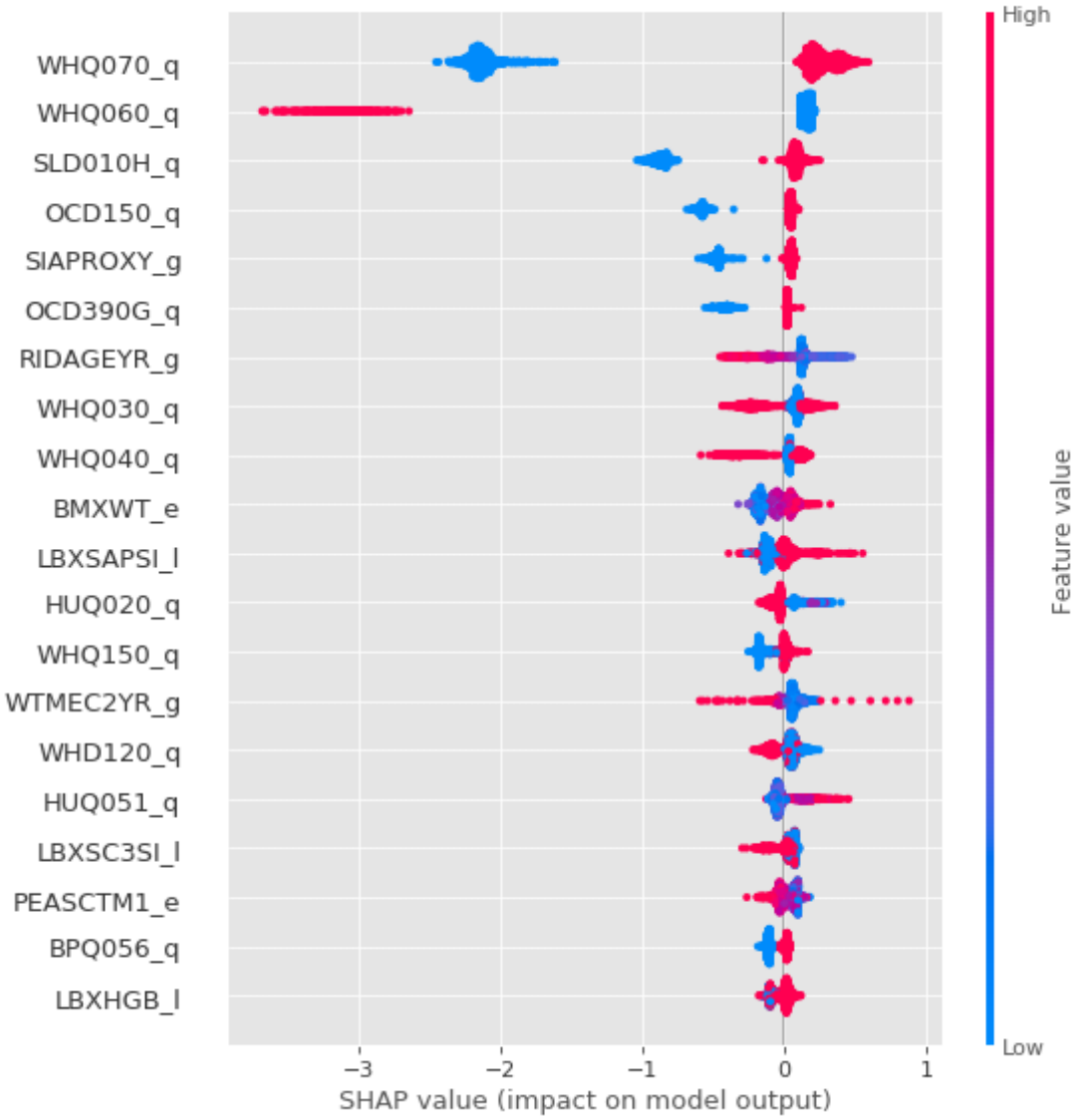


Fig 15 – shap value for XGboost model (the best attempt)

We can also look into individual level results. The individual are only 1% likely to gain 5 pound in 1 year. One important reason is she/he had not thought about losing weight in the past 1 year. It makes sense as that could serve as an indicator that losing weight was not really her/his concern.

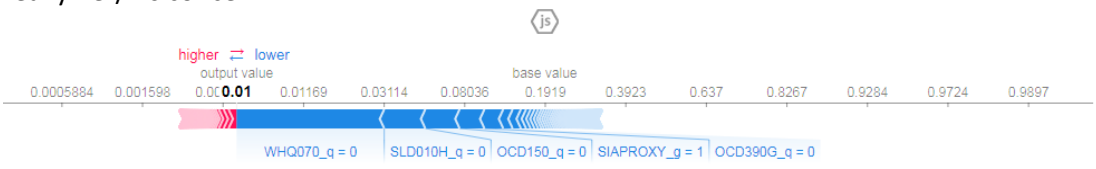


Fig 16 - SHAP value for individual

5.2 Reflection

It has been great journey to go through the whole end-to-end model development process.

Proposal: Coming up with a proposal was fun but definitely not easy. The main difficult was to ensure there are good quality data available to solve my problem statement.

Data Exploration and Understanding: This phase requires a lot of patience, as the data I use is massive. It initially starts with 1800 features and have five data dictionaries for them. In order to be able to navigate to right data dictionary more efficiently, I added suffix to each feature. Though I have spent considerably significant amount of time, I could have spend more time in exploring and plotting the features.

Data Pre-processing: I was lucky that the categorical features turn out to be useless features and I don't have to spend much time on categorical conversion using LabelEncoder or OneHotEncoder. The challenge mainly comes from dimensionality reduction. As I mentioned, the current solution is not perfect and it is the phase worth spending more time, in order to ensure we keep all beneficial features.

Benchmark Modeling: Refining benchmark model was fun. The challenge is to spend more time to apply stepwise to build Logistic Regression from scratch to ensure all features are statistically significant.

XGboost Model and Hyper-Parameter Tunning: Train a XGboost Model is easy, while tuning it is hard! This is the phase I spend majority of my time on, while the return is not satisfying. I have days running the Bayesian optimization overnight and hoping the next day the algorithm will give me magic combination of hyper- parameters. However, most of the time I m left with disappointment with the results. I have also tried innovative approach to visualize the hyper-parameters (as seen in session 6.4) and it really helped me to gain better understanding of hyper-parameters. **I'd really like some advice from reviewer on whether I have made mistake in tuning and what other better library I can try**.

In summary, I have learnt a lot through this project, especially have gained better understanding of XGboost model and Bayesian Optimization for hyper-parameter tuning. I wish I could have spent more time in data preprocessing stage and understand data more thoroughly.

5.3 Improvement

There are lot things to improve for this project.

1) Improve on dimensionality reduction: refine the current approach with combination of gini and correlation. At the moment they are looked into separately.
2) Refine benchmark model properly through stepwise algorithm.
3) Due to time constraint, I have not trialled out LGB and Neural Nets as planned. It will be interesting to see how these approaches perform as compared to XGboost.
4) Most importantly, I wanted to spend more time to understand why the hyper-parameter tuning is not giving better results. I have done very check on hyper-parameters (Appendix 6.4), but still could not figure out the root cause. I'd really like to get some feedback on that and happy to dive deeper into it.

6.Appendix

6.1 Categorical Variables

Variable	Meaning	Included?
OHX02CTC_e -OHX31CTC_e	Coronal Caries: Upper right xnd molar (2M) tooth code	Not required
CSXTSEQ_e	Sequence in which whole mouth taste tests were administered.	Not required
SMDUPCA_q	Cigarette 12-digit Universal Product Code	Not required
SMD100BR_q	BRAND OF CIGARETTES SMOKED BY SP (SUB-BRAND INCLUDED IF APPLICABLE AND AVAILABLE)	Not required

6.2 List with suspiciously high roc (roc > 1)

PHACOFHR_I	The time (in hours) since the examinee last drank coffee or tea with cream or sugar
WHD080U_q	How did {you/SP} try to lose weight?
DRQSDT7_d	What kind of diet are you on? (Is it a weight loss or low calorie diet: low fat or cholesterol diet; low salt or sodium diet; sugar free or low sugar diet; low fiber diet; high fiber diet; diabetic diet; or another type of diet?)
PAQ759T_q	In what school sports or physical activity clubs {do you/does SP} participate?
CSQ210_q	{Have you/Has SP} ever had any of the following? wisdom teeth removed.
URXVOL3_I	The volume of urine collection #3 (mL)
DRD370EQ_d	Number of times cod was eaten in the past 30 days
OSD050CA_q	Did that fracture occur as a result of . .
OSQ220_q	Was he . .
URDFLOW3_I	Urine #3 Flow Rate (mL/min)
OSQ090A_q	Was this fracture the result of severe trauma such as a car accident, being struck by a vehicle, a physical attack, or a hard fall such as falling off a ladder or down stairs?
CSXGRAOD_e	Forced Choice Odor Selection, Grape Scent
PHACOFMN_I	The time (in minutes) since the examinee last drank coffee or tea with cream or sugar.

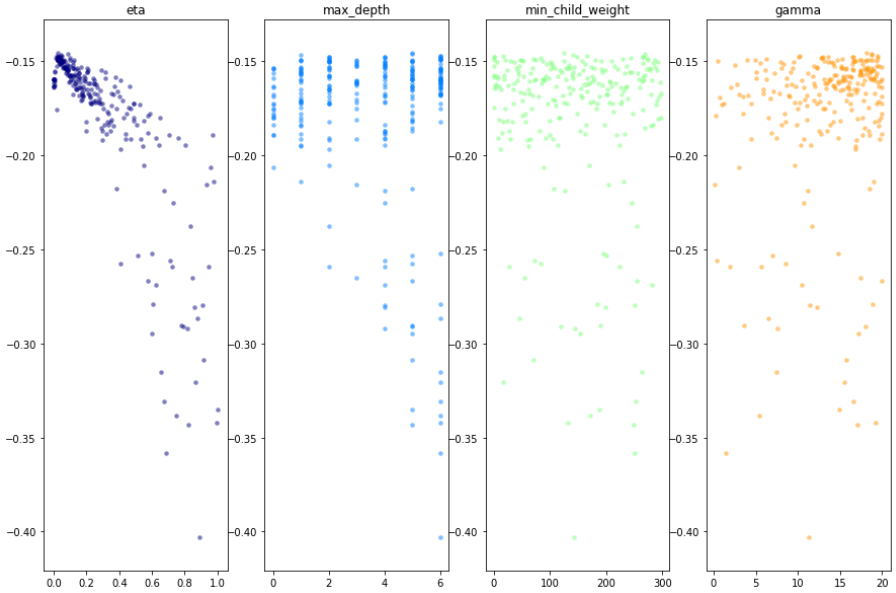
Further drop list, two variables that are used to create weight increase WHD020, WHD050

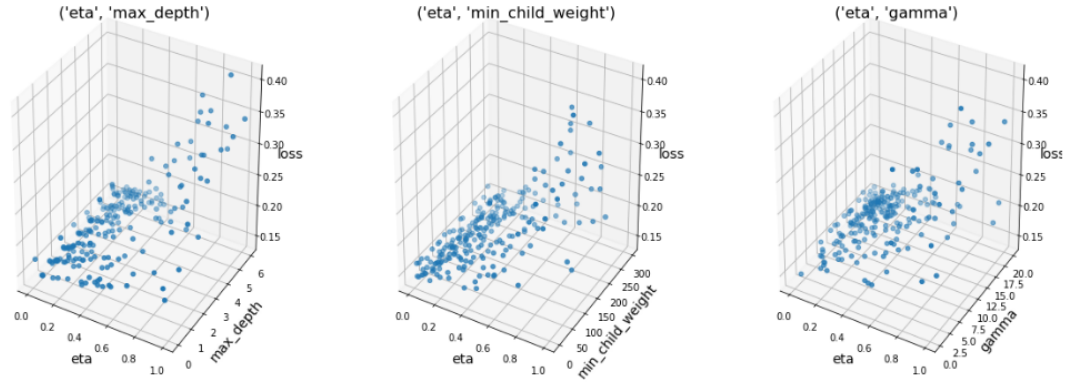
6.3 Inputs for Logistic Regression 3

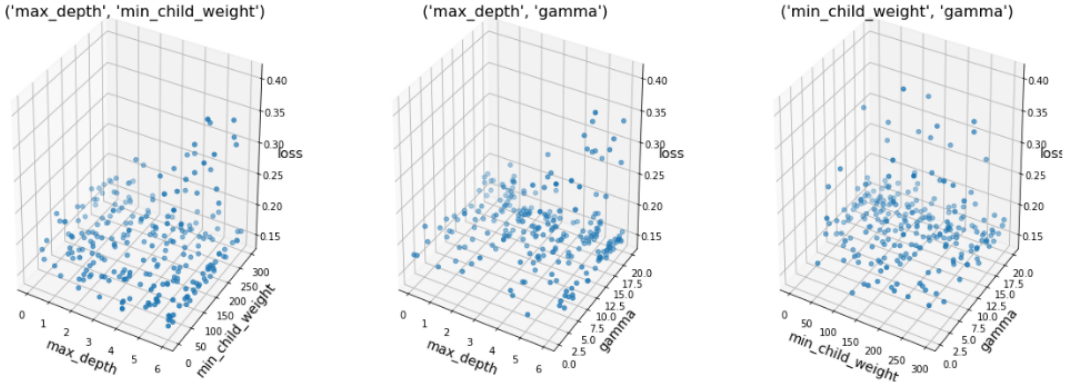
DMDHRAGE_g (-0.4541)	HH reference person's age in years
DMDHSEDU_g(0.3003)	HH reference person's spouse's education level
DIQ175A_q(0.2992)	Why {Do you/Does SP} think {you are/he is/she is} at risk for diabetes or prediabetes?
DBQ421_q (1.0488)	{Do you/Does SP} get these breakfasts free, at a reduced price, or {do you/does he/she} pay full price?
HUQ010_q (1.0372)	I have some general questions about {your/SP's} health. Would you say {your/SP's} health in general is . . .
HUD080_q(0.3289)	How many different times did {you/SP} stay in any hospital overnight or longer {during the past 12 months}? (Do not count total number of nights, just total number of hospital admissions for stays which lasted 1 or more nights.)

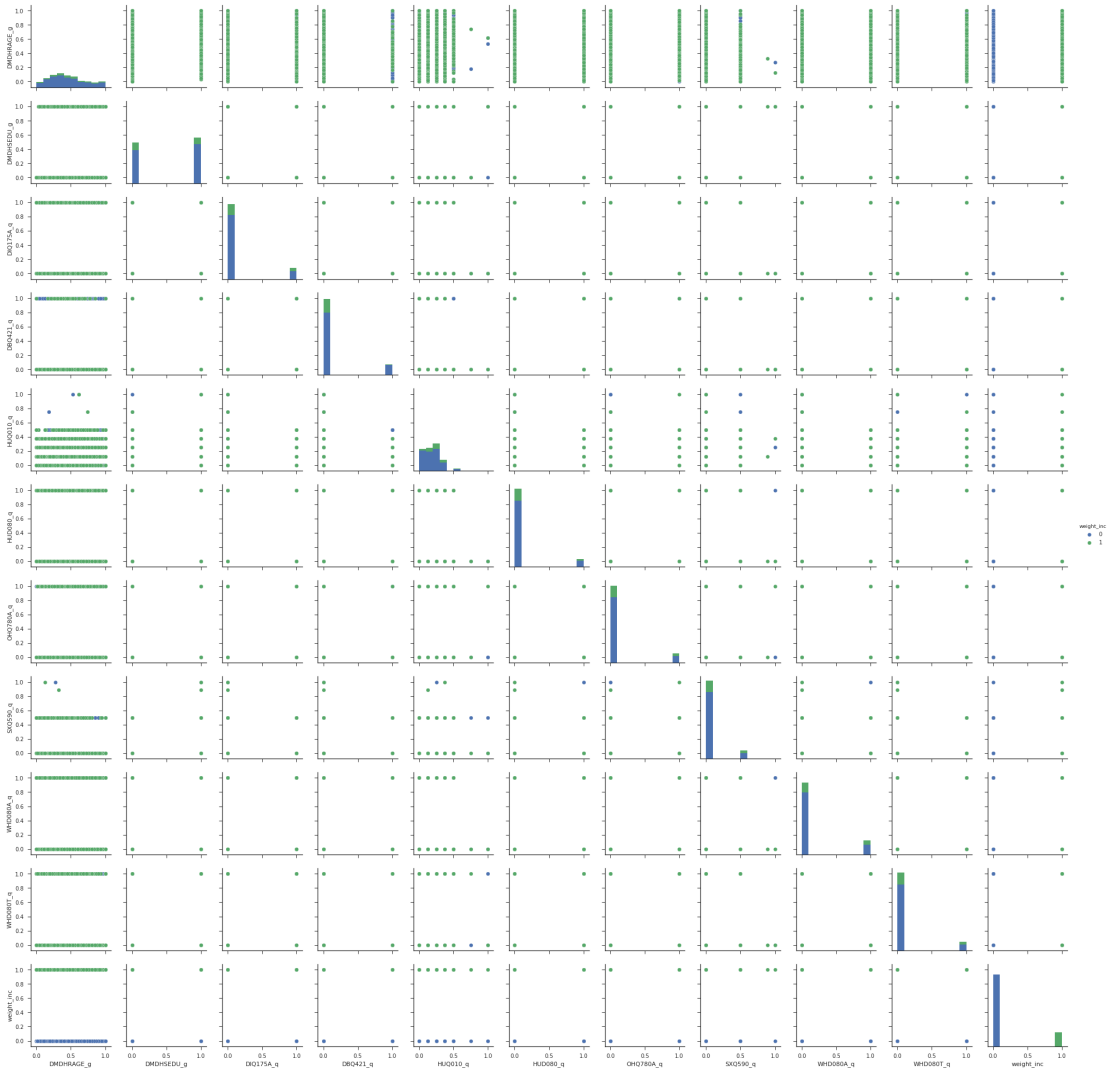
OHQ780A_q(0.4148)	What were the reasons that (you/SP) could not get the dental care (you/she/he) needed?
SXQ590_q(0.8474)	Of the persons you had any kind of sex with in the past 12 months, how many were five or more years older than you?
WHD080A_q (0.4713)	How did {you/SP} try to lose weight?
WHD080T_q(0.3897)	How did {you/SP} try to lose weight?

6.4 Free form visualization – From Hyper Parameter Tuning









Bibliography

Zhang, O. (2015, 01 01). *linkedin*. (O. D. Conference, Producer) Retrieved 01 01, 2015, from Slideshare: https://www.slideshare.net/odsc/owen-zhangopen-sourcetoolsanddscompetitions1

Wikipedia. (n.d.). *Gini coefficient*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Gini_coefficient

World Health Oraganisation. (2018, 2 16). *Obesity and overweight*. Retrieved from World Health Oraganisation: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight

XBoost. (n.d.). *XGboost Parameters*. Retrieved from XGboost: https://xgboost.readthedocs.io/en/latest/parameter.html

Afonja, T. (2017, 12 08). *Accuracy Paradox*. Retrieved from Towards Data Science: https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b

Brownlee, J. (2016, 09 02). *Avoid Overfitting By Early Stopping With XGBoost In Python*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/avoid-overfitting-by-early-stopping-with-xgboost-in-python/

Jimenez, M. P. (2018). *A demographic, clinical, and behavioral typology of obesity in the United States: an analysis of National Health and Nutrition Examination Survey 2011-2012.* Annals of epidemiology vol. 28,3.

Li, P. (2010). Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost. *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence* (pp. 302–311). UAI: eprint arXiv:1203.3491.

maelfabien. (2019, 07 26). *A Guide to Hyperparameter Optimization (HPO)*. Retrieved from github: https://maelfabien.github.io/machinelearning/Explorium4/#conclusion
SYNCED. (2017, 10 22). Retrieved from Tree Boosting With XGBoost — Why Does XGBoost Win "Every" Machine Learning Competition?: https://medium.com/syncedreview/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition-ca8034c0b283
scikit learn. (n.d.). *sklearn.linear_model.LogisticRegression*. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
Tianqi Chen, C. G. (2016). XGBoost: A Scalable Tree Boosting System. *KDD.* San Francisco: ACM CIGKDD Conference.