Department of 統計及精算學系
Statistics & Actuarial Science
THE UNIVERSITY OF HONG KONG

FICO

The University of Hong Kong

Academic Year 2020-2021 Semester One

Dec 6, 2020

STAT3612 Statistical Machine Learning

Final Report

Interpretable Machine Learning on HELOC

GUN Chun Ho 3035466320

Table of Content

Abstract

This project aimed at building a machine learning model for the prediction of credit risk based on the HELOC dataset. This project will adopt the approach of comparing the two models selected from White-Box models and Black-Box models respectively. Several methods will be used in feature selection to boost the accuracy of both methods. The accuracy of the two models will be considered in both normal and monotonicity basis. Several interpretation methods will then be used to compare their performance on their interpretability as well. My result shows that the selected White-Box model has a better performance on both accuracy and interpretability.

Introduction

Management of credit risk by controlling the credit standards for borrowers is one of the challenges facing in the banking industry. How to maximize the risk-adjusted discount rate of return while keeping the credit risk exposure at an acceptable level is the key to the success of financial institutes. The machine learning model may assist in prediction of credibility of an individual based on his/her historical behaviour.

In this project, a real data case study is constructed based on the data in Home Equity Line of Credit (HELOC) loans, which contains 23 raw features such as "Percent trades never delinquent", "Months since most recent inquiry", "Months since most recent delinquency". The target variable is the "RiskFlag", which indicates if the individual will pay back the loans or not.

In building a machine learning model, one of the most common problems faced by the data scientists is the balancing between model accuracy and interpretability. Black-Box models usually provide a higher accuracy compared to the White-Box models. However, due to the simplicity, the latter usually has higher interpretability, which is important for decision-makers. Therefore, one of the goals of our project is to compare the performance of models chosen from White-Box and Black-Box models in terms of accuracy and interpretability.

To build the machine learning model, the first step is a simple data exploration and imputation of missing data, followed by feature selection. Feature selection is essential in this project. Not only can it simplify the dataset but also reduce the chance of overfitting. We have adopted 3 methods in selecting the relevant features. Then, two models, chosen from White-Box models and Black-Box models, are built with and without monotonic constraints for the comparisons. The performances of the two models are evaluated by the prediction accuracy and interpretability.

Data Exploration

The dataset consists of 10459 individuals with 23 raw features. Each has a target variable “Risk Flag” to indicate whether the loan has been 90-day delinquent over a two-year period. It is a binary variable which takes good and bad as its values.

We noticed that there are several subsets in the features based on the data meaning. They are the x6 & x7 and x16 & x17 pairs. The table below shows the details.

Table 1.1: data meaning of different features

Subset	Variable	Description	Meaning
1	x6	Number Trades 60+ Ever	Number of trades which are more than 60 past due
	x7	Number Trades 90+ Ever	Number of trades which are more than 90 past due
2	x16	Number of Inq Last 6 Months	Application of new credit card for more credits
	x17	Number of Inq Last 6 Months excluding last 7 days.	The exclusion of “application last 7 days” probably due to the removal of inquiries for price comparison shopping

To better observe the relationship between features, especially the subsets, we have conducted the correlation matrix and the scatterplot on the dataset.

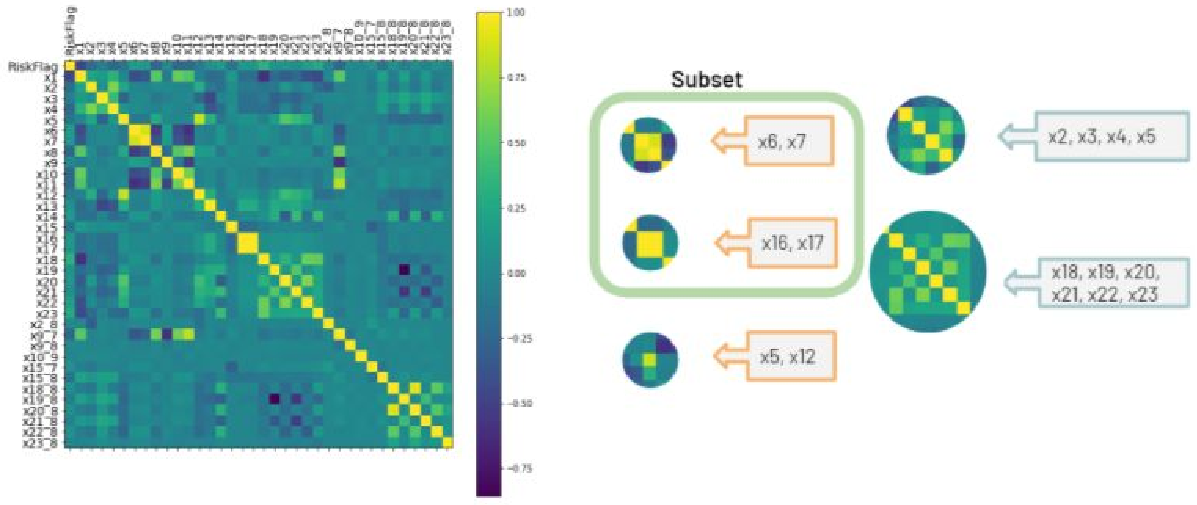


Fig.1.1 Scatter plot of the dataset with selected captures

The extra columns starting from x2_8 are the result of the data preprocessing as explained in the next section.

The scatter plot can demonstrate the superset-subset relationship in the two pairs (x6 & x7, x16 & x17) as a relatively high correlation (the two yellow squares) are shown. Apart from the subset relationship, 3 groups of variables also present a higher correlation among all. They are 1) x5, x12; 2) x2, x3, x4, x5; 3) x18, x19, x20, x21, x22, x23.

We also conduct the correlation matrix analysis on the features to further investigate their relationship.

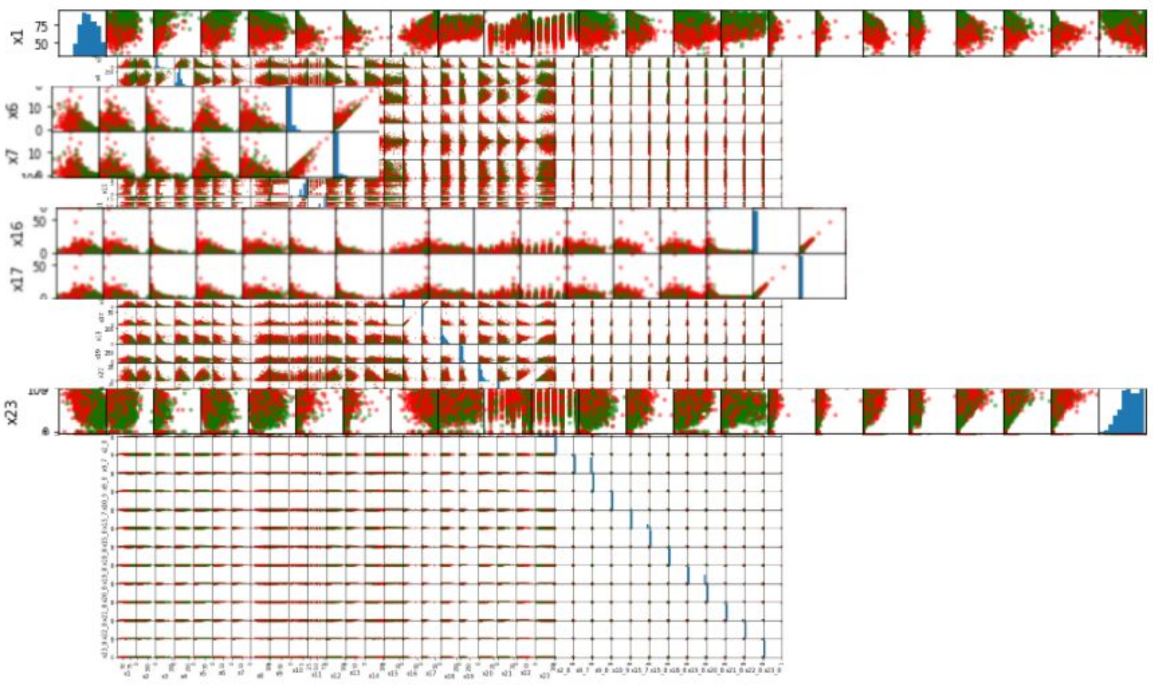


Fig.1.2 Correlation matrix of all features

In the correlation matrix, we can see a clear “separation line” in x1 and x23. It may imply that they are relatively more important features among all since it can separate the target variable in each comparison with good performance. For the subset relationship, we can also see a “triangular representation” in the x6,x7 pair and the x16,x17 pair which further proved their set-subset relationships.

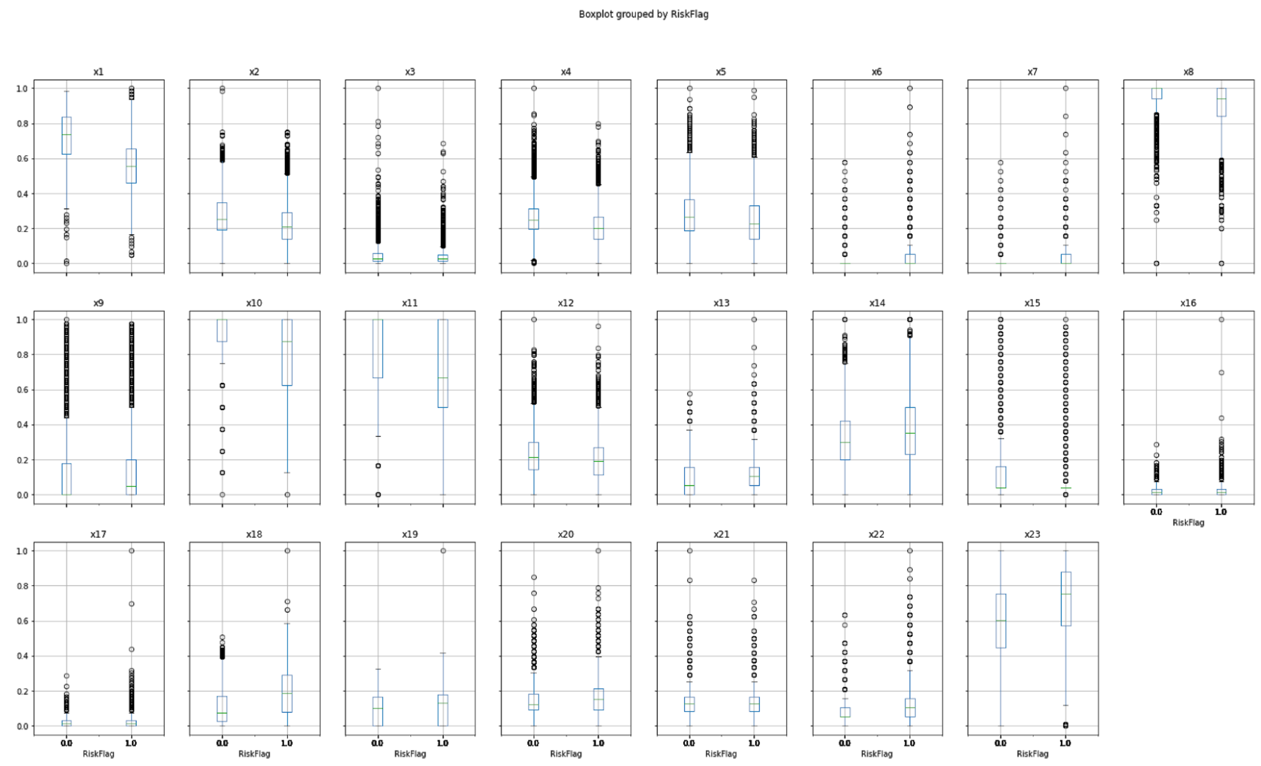


Fig.1.3 Boxplots of all features

From the boxplot, it is apparent that x1 has a strong effect on RiskFlag which has the same finding from its scatterplot and correlation matrix. A similar observation is also given by some other values such as x23but their effects are notably weaker.

Data preprocessing

In HelocData, missing values are represented by −7,−8 or −9. We plot bar charts for the distribution and correlation of each error code for analysis:

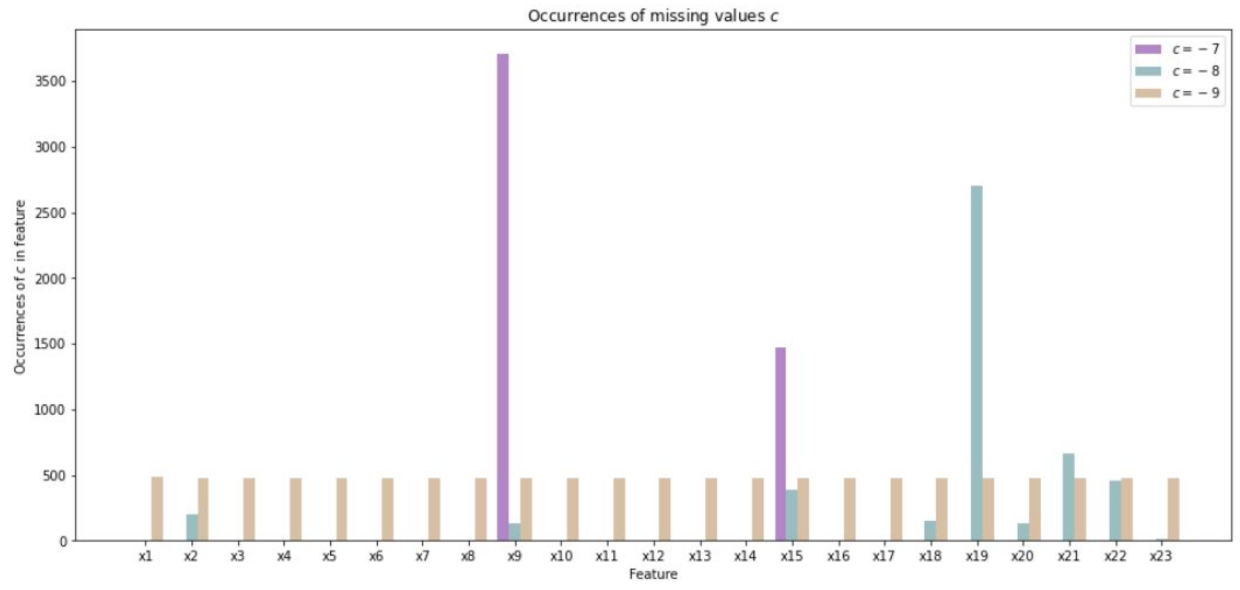


Fig.1.4 Distribution of occurrences of -7, -8, -9 in each feature

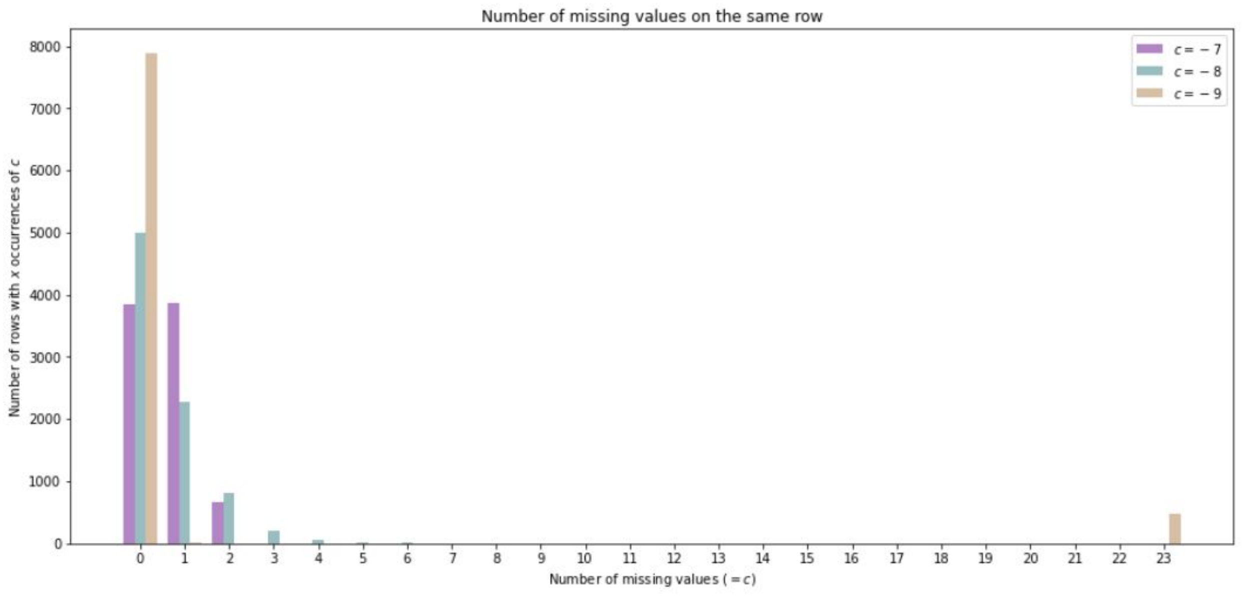


Fig.1.5 Number of missing values on the same row

Handling of -7

From the first, bar chart, −7 only occurs in x9 and x15. x9 and x15 are the numbers of months since the most recent delinquency/inquiry.

−7 implies that delinquency/inquiry never happened, so they should be imputed with a reasonable value that maximizes feature effect, such as the corresponding extrema in their columns.

This is achieved by adding the features x9_7 and x15_7, indicating that the x9/x15 column contains a "never happened" model, then imputing the missing cells with −1.

For i∈{x9, x15}, regression-based models can choose an appropriate coefficient βi_7 such that column i contributes to the result by βi_7 if it is a missing value, or by βix if it is not missing.

Handling of -8

−8 means "no valid trades".

In x18, x19 and x23, −8 is the placeholder of division-by-zero errors. In other columns such as x21, −8 implies lack of installment trades, etc.

Therefore, a similar technique as in handling −7 can be used for each column with −8 data. Note that such imputation is only performed on columns with at least one −7/−8 value. If they never occur on the training data, nothing can be inferred about their effects on testing data, and adding such a feature to the model contributes no positive effect on the model accuracy.

Handling of -9

−9 means data are uninvestigated.

From the second bar chart, it appears that occurrences of −9 are clustered, i.e. rows that have at least one −9 most likely have all cells −9. We call these rows Completely Uninvestigated Data (CUD) for brevity.

Let X be the random variable for our prediction for a CUD row (Bad = 1) and p be the proportion of RiskFlag = Bad among the CUD rows. The prediction accuracy $\Pr(X=1)p+\Pr(X=0)(1-p)$ is maximized when X is the constant $1\{p>0.5\}$. In our train-test partition, we measured $p>0.5$. The dataset is partitioned into CUD and non-CUD, where further investigation is only performed on the non-CUD subset.

Splitting nominal data

x10 and x11 are semi-nominal. Labels beyond the monotonic constraint ([0,7] and [2,8] respectively) are imputed as 0 and an indicator column is added for each such label.

Fixing RiskFlag

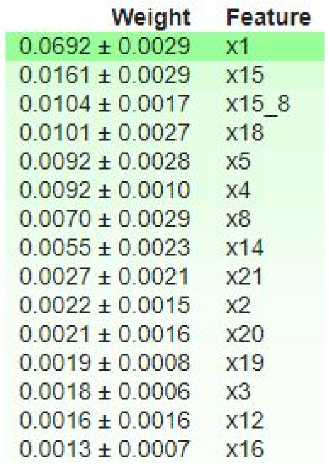
RiskFlag is converted to a 0/1 column where 0 implies Good and 1 implies Bad (consistent with the monotonicity constraint on Pr(Bad=1)).

Feature Selection

We have included three different approaches: permutation importance, backward selection and data interpretation when doing features selection. When there is collinearity, backward selection is usually preferred. However, if the model built is complicated, forward selection is also acceptable (Heinze, Wallisch & Dunkler, 2018). Since XGBoost has the highest performance in our preliminary test, we use the XGBoost model as a benchmark for features selection. We have included 3 methods for feature selection and we summarized the result for consideration.

Method 1: Top 15 features with highest Variable importance

Method 1 is obtained from permutation feature importance. We randomly shuffle a single column (i.e. one feature), make predictions using the re-ordered dataset and observe the changes of accuracy. The performance deterioration can be interpreted as the significance of that feature. The top 15 features with highest importance are selected by this method and they are listed below:



Weight	Feature
0.0692 ± 0.0029	x1
0.0161 ± 0.0029	x15
0.0104 ± 0.0017	x15_8
0.0101 ± 0.0027	x18
0.0092 ± 0.0028	x5
0.0092 ± 0.0010	x4
0.0070 ± 0.0029	x8
0.0055 ± 0.0023	x14
0.0027 ± 0.0021	x21
0.0022 ± 0.0015	x2
0.0021 ± 0.0016	x20
0.0019 ± 0.0008	x19
0.0018 ± 0.0006	x3
0.0016 ± 0.0016	x12
0.0013 ± 0.0007	x16

Figure 2.1: Permutation importance of the method 1

Stepwise selection method

We further analyze the feature importances using stepwise selection methods. We iteratively remove one feature from the set of features, retrain the model with the remaining features and select one to remove again. Method 2 and 3 are backward selection with different feature select criteria.

Method 2 : Remove most important features

In every step the most important feature will be removed and it will be added to a temporary list. This approach preserves the most important features.

Next, we want to determine the number of features selected to achieve highest accuracy. The plot below shows the number of features selected and their accuracy.

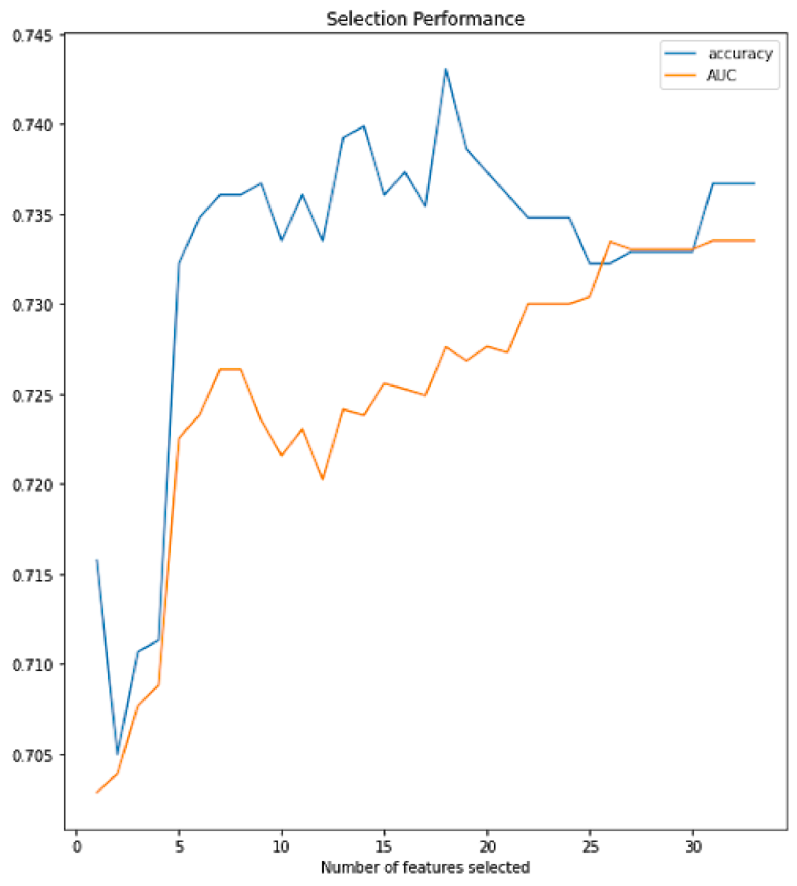


Figure 2.2: Number of features and accuracy for method 2

The plot indicated that selecting 17 features achieve highest accuracy. Note that the features selection starts from the beginning of the temporary list. Hence, the top 17 features in the temporary list will be selected, they are x1, x18, x22, x8, x4, x10, x11, x9_7, x9, x6, x20, x23, x15, x2, x5, x12 and x7.

Method 3: Remove least important features

In every step, the least important feature will be removed. The features last removed are used as the final selection.

This method is expected to remove collinearity. Suppose x_i and x_j are highly correlated, and x_i has higher importance than x_j . When x_j is removed, the retrained model must rely on x_i for the information previously provided by x_j , so it is expected that x_i takes the importances of both itself and x_j as in the previous method.

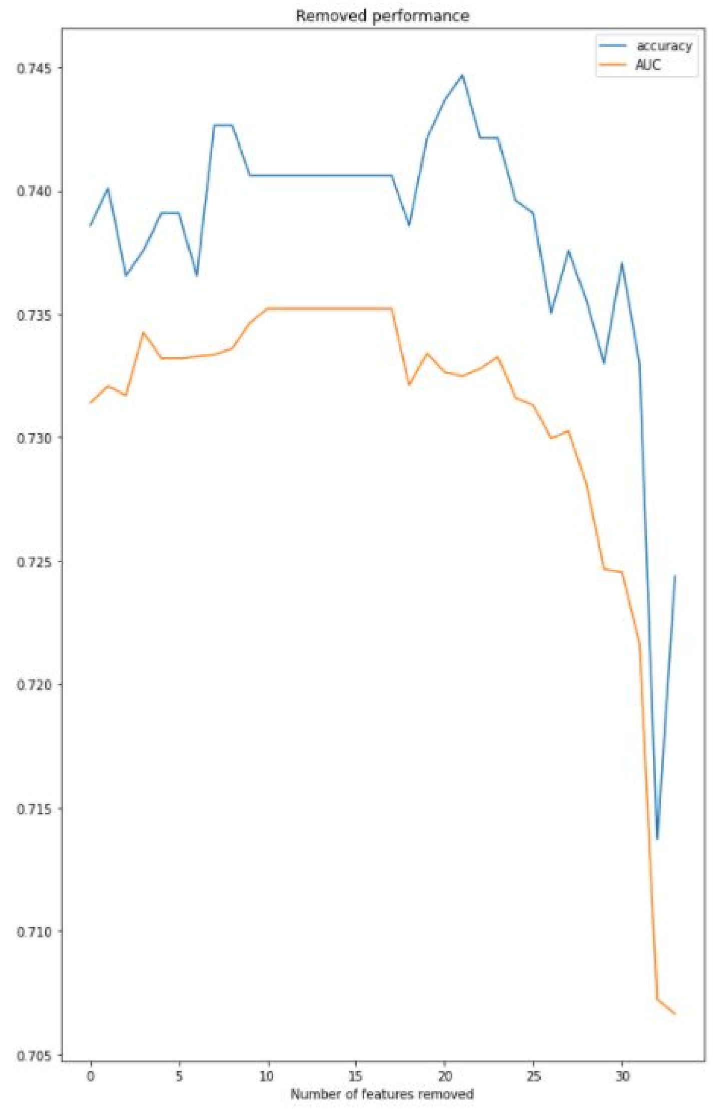


Figure 2.3: Number of features and accuracy for method 3

The plot indicates that removing 20 features for this method achieves highest accuracy. Since we have total 35 features at the beginning, the 15 features remaining will be selected and they are x1, x4, x5, x8, x10, x12, x14, x15, x15_8, x17, x18, x20, x21, x22 and x23.

Justification

We designed three methods due to the possible interaction effect of different pairs of features. It may affect the final result significantly. For example, the second method may select a bunch of highly correlated features. When one of the correlated pairs is being removed, the importance of the other one at the next cycle will increase. Therefore, we adopt three methods together to compensate for the effect.

Summary

We summarize the result generated by the above methods and select the features for our model. The table below shows the features selected by at least one method.

Table 2.1: features selected by at least one method

Variable	Description	Randomly shuffling	Remove the most important	Remove the least important
x1	Consolidated version of risk markers	Y	Y	Y
x4	Average Months in File	Y	Y	Y
x5	Number Satisfactory Trades	Y	Y	Y
x8	Percent Trades Never Delinquent	Y	Y	Y
x12	Number of Total Trades (total number of credit accounts)	Y	Y	Y
x15	Months Since Most Recent Inq excl 7 days	Y	Y	Y
x18	Net Fraction Revolving Burden. This is revolving balance divided by credit limit	Y	Y	Y
x20	Number Revolving Trades with Balance	Y	Y	Y
x2	Months Since Oldest Trade Open	Y	Y	
x10	Max Delq/Public Records Last 12 Months. See tab "MaxDelq" for each category		Y	Y

x14	Percent Installment Trades	Y		Y
x15_8	No Usable/Valid Trades or Inquiries	Y		Y
x21	Number Installment Trades with Balance	Y		Y
x22	Number Bank/Natl Trades w high utilization ratio		Y	Y
x23	Percent Trades with Balance		Y	Y
x3	Months Since Most Recent Trade Open	Y		
x6	Number Trades 60+ Ever		Y	
x7	Number Trades 90+ Ever		Y	
x9	Months Since Most Recent Delinquency		Y	
x9_7	Condition not Met (e.g. No Inquiries, No Delinquencies)		Y	
x11	Max Delinquency Ever.		Y	
x16	Number of Inq Last 6 Months	Y		
x17	Number of Inq Last 6 Months excl 7days. Excluding the last 7 days removes inquiries that are likely due to price comparison shopping.			Y
x19	Net Fraction Installment Burden. This is installment balance divided by the original loan amount.	Y		

If a feature is selected by at least two methods, it will be selected. For features only selected by one method, we will examine them one by one by our interpretation. The rest of the features will be dropped.

Table 2.2: Features selected by one method and interpretation

Variable	Description	Interpretation
x3	Months Since Most Recent Trade Open	Carrying an account does not necessarily reflect one's affordability. For example, the individual could be very rich and able to afford debts.
x6	Number Trades 60+ Ever	In method 3, it only ranks 21.
x7	Number Trades 90+ Ever	In method 3, it only ranks 34.
x9	Months Since Most Recent Delinquency	Comparing x9 with x9_7, the issue of "having delinquency or not" should be more important since this reflects one's ability in managing credit.
x9_7	Condition not Met (e.g. No Inquiries, No Delinquencies)	Cases x9_7 is included in variable "Max. Delq/Public Records Last 12 Months" (x10), which has been selected.
x11	Max Delinquency Ever.	In method 3, it is eliminated in the first 5 steps. Compared to x10, it shows less significance.
x16	Number of Inq Last 6 Months	Normally, credit card companies will conduct due diligence to control risk before approving extra credit card applications. For example, checking credit history. Therefore, people that successfully apply for a credit card usually are considered as "Good". In this case, x16 & x17 cannot contribute much to the prediction.
x17	Number of Inq Last 6 Months excl 7days. Excluding the last 7 days removes inquiries that are likely due to price comparison shopping.	
x19	Net Fraction Installment Burden. This is the installment balance divided by the original loan amount.	When comparing with x18: Net Fraction Revolving Burden, it is less significant.

In summary, we have selected 15 features for our model, they are x1, x2, x4, x5, x8, x10, x12, x14, x15, x15_8, x18, x20, x21, x22 and x23.

Note that a number of indicator columns have been removed while their corresponding features are still present. This implies one of two possibilities:

- Imputation with 0 is close enough to the optimal value.
- There is too little data with the error term for imputation value to matter.

To verify this, we try to replace the instances of −1 in each feature with their maximum value and evaluate the change in accuracy with respect to XGBoost and got the following results:

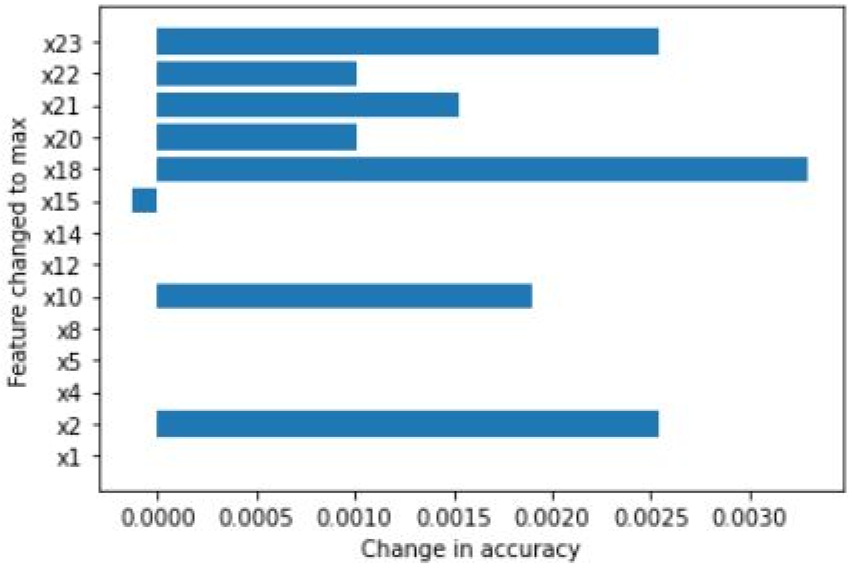


Figure 2.4: Change in accuracy caused by changing imputation to maximum

The feature with greatest improvement in accuracy is x18 with only 0.3%. We consider this to be insignificant and continue to use the model with only zeros for imputation.

Machine Learning Methods

Model Selection (Without monotonicity)

In this selection of algorithms and models, we have contracted different models like Logistics GAM, linear GAM, Logistic regression and XG boost etc. After we have compared their training and testing accuracy score, we decided to use Logistic GAM as our White-Box model and XG boost for our Black-Box model. After that, we use different feature engineering methods like Bagging, piecewise linear method, splining and grid search.

The table below shows the accuracy score of our models.

Table 3.1: Training and testing accuracy score on different models without monotonic constraint

Model	**Accuracy on training set**	**Accuracy on testing set**
Logistic GAM	0.749524172	0.754545455
Logistic Regression	0.734551453	0.74949495
Linear GAM	0.745844436	0.743939394
XGboost random search	0.756249207	0.742929293
XGBoost exhaustive grid search - Acc	0.756249207	0.742424242

After we constructed different models, we compared the testing accuracy score among them. We decided to use Logistic GAM and XGboost classifiers since they have a relatively higher testing accuracy score, with **0.754545455** and **0.743939394** respectively.

Feature Engineering

We wondered if we could further boost the accuracy score, so we tried different feature engineering methods like **Binning**, **Natural Cubic Spline Basis Expansion** as well as different classifiers and random search methods.

We trained the models by different methods and compared with their accuracy scores. The table below shows a summary of their accuracy score.

Table 3.2: Training and testing accuracy score on different models with feature engineering

Model	**Accuracy on training set**	**Accuracy on testing set**
Logistic GAM	0.749524172	0.754545455
Logistic GAM with equal width	0.725669331	0.74040404
Logistic GAM with IV	0.734551453	0.747979798
Logistic GAM with splines (spline=1) (RELU)	0.739753838	0.75454545
Logistic GAM with splines (spline=2)	0.739246289	0.755050505
Logistic GAM with splines (spline=3)	0.739753838	0.75404040
Logistic GAM with grid search	0.738992514	0.754545455
Logistic Regression	0.734551453	0.74949495
Linear GAM with grid search	0.735439665	0.74646464
Linear GAM	0.745844436	0.743939394
XGBoost exhaustive grid search - Acc	0.756249207	0.742424242

XGBoost First random search	0.7540921203	0.743939393
XGBoost Second random search	0.7540921203	0.743939393

From the above table, we realised that the binning techniques with equal width actually reduce the Accuracy on the testing set from **0.754545455** to **0.74040404.** With splines (splines degree = 2), the accuracy score increased to **0.755050505**

For the XGboost, we can see that the model with random search no matter if it has a second random search or not, it does not have much effect on the testing accuracy. It stays with **0.743939394.**

As a result, we choose Logistic GAM with degree 2 splines and XG boost random search as our White-Box and Black-Box model without monotonic constraints since their testing accuracy scores are the highest, with **0.755050505** and **0.743939394** respectively.

Model Selection (With monotonicity)

For the model with monotonicity, we conducted a similar method with the models without monotonicity. We re-trained the model by applying the monotonicity data. The results are listed below:

Table 3.3: Training and testing accuracy score on different models with monotonic constraint

Model	**Monotonic training accuracy**	**Monotonic testing accuracy**
Logistic GAM	0.740388276	0.755050505
Logistic GAM with equal width	0.720974496	0.738888889
Logistic GAM with splines (spline=1)	0.73772364	0.75555555

(RELU)		
Logistic GAM with splines (spline=2)	0.738738739	0.756060606
Logistic GAM with splines (spline=3)	0.740007613	0.757575757
Logistic GAM with grid search	0.662860043	0.667171717
XGboost classifier	0.749904834	0.750505051
XGBoost exhaustive grid search - Acc	0.749904834	0.749904834
XGBoost First random search	0.749524172	0.750505051
XGBoost Second random search	0.749524172	0.750505051

As we can see from the above table, we can see the Logistic GAM with the Natural Cubic Basis Expansion with the degree 2 has the highest which is **0.756060606** so we selected the Logistic GAM with the Natural Cubic Basis Expansion with the degree 2 as our White-Box model. For Black-Box model with monotonic constraints, we select XGBoost with the random search since it has the highest testing score which is **0.750505051**.

Results: White-Box model vs Black-Box model

Among different machine learning models we tried, we have selected the Logistic GAM and XGBoost models to be our candidates on our model comparison of the white and Black-Box. To maximize the accuracy score of the two models, we applied several Feature Engineering and model tuning methods, including Spline Expansion, Hyperparameter Tuning, Grid Search and Randomized Search on Cross-Validation. In terms of the monotonicity, it is worth mentioning that we applied the constraints individually based on the Data Dictionary on HELOC instead of applying them in one-cut.

With the internal comparison of each of the models, we have selected the highest accuracy one in each category to compare with their counterpart. The following table presents the details on the accuracy score comparison.

Table 4.1: Accuracy Score Comparison Table

Without the monotonic constraints	
Logistic GAM with Spline Degree = 2	XGBoost random search
0.7550505050505051 ✔	0.743939393939
With monotonic constraints	
Logistic GAM with Spline Degree = 2	XGBoost random search
0.7560606060606061 ✔	0.750505050505

Interestingly, we found that the two classes (White-Box and Black-Box) in both conditions (without the monotonicity, with the monotonicity) have higher performance on the accuracy score. Comparing each class across the condition of monotonicity, we can also observe a better performance on accuracy. When we further investigated the accuracy of the White-Box simple models, we noticed that the Logistic GAM with degree-2 splines with monotonic constraints

(0.75606) can beat that without monotonic constraint (0.75505). It may imply that both models without monotonic constraints have a little bit of an overfitting problem.

Interpretation on Final models

Global Interpretation on Logistic GAM

We conducted two partial dependence plots on the two selected models (Logistic GAM with spline order = 2 without monotonic constraint and Logistic GAM with spline order = 2 with the constraint).

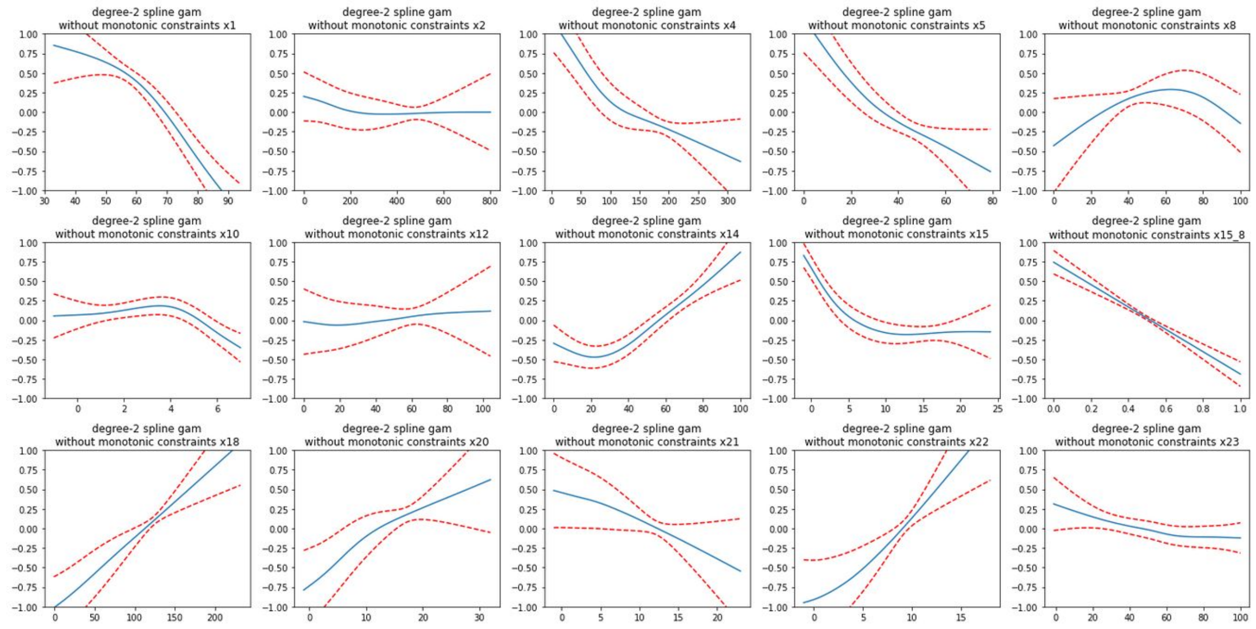


Fig. 3.1 Partial Dependence Plot on Logistic GAM with spline order = 2 without monotonicity

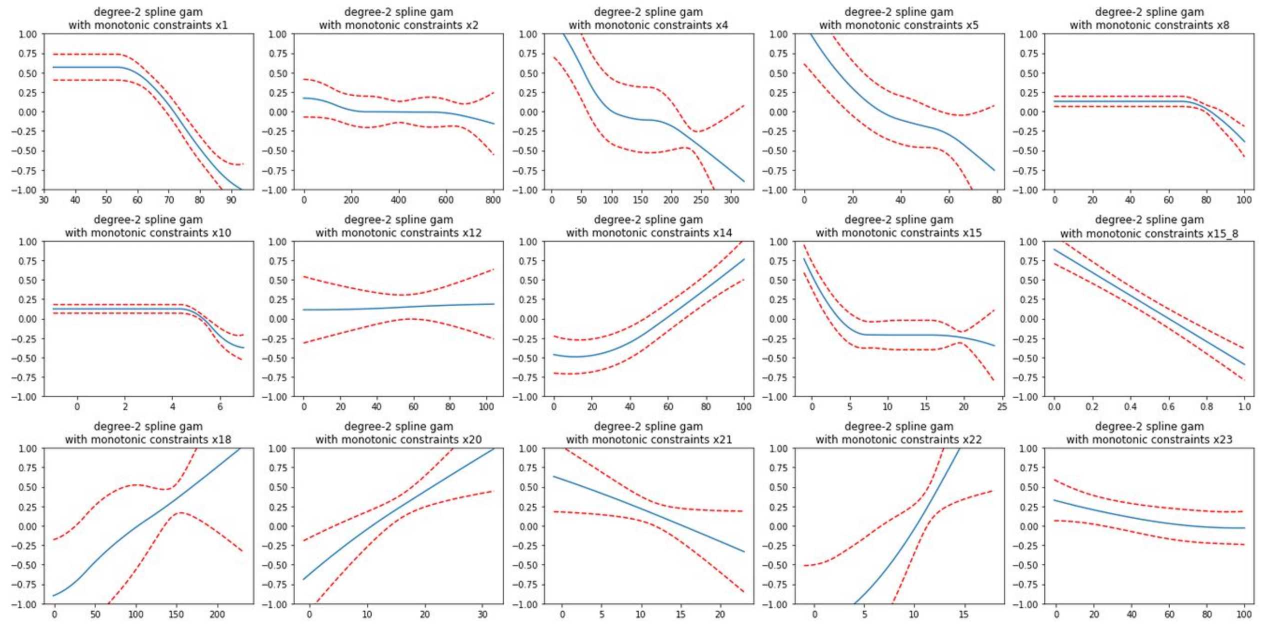


Fig. 3.2 Partial Dependence Plot on Logistic GAM with spline order = 2 with monotonicity

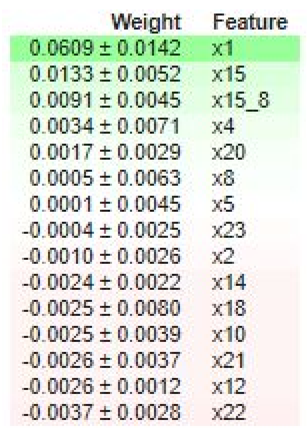
One big contrast can be observed in the comparison of the x8 in both conditions (with and without monotonic constraint). In the monotonicity condition, the x8 value remained constant until 80 whereas the x8 appeared as a concave line in the non-monotonicity condition. A similar contrast can be seen in the x10 with a less concave form in the non-monotonicity condition. Since the x10 itself is monotonically decreasing for 0-7, it obviously improved with the constraint.

We can also observe the linear relationship for some features in the PDPs. The x15_8, x18, x22 and x21 in both plots are the linear line. Similarly, the x1, x2,x4,x5,x12,x20,x22, x23 show a close-to-linear relationship in the both graphs.

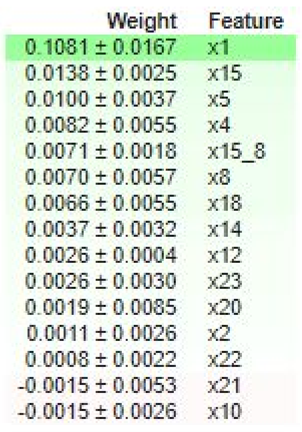
In those linear and close-to-linear relationships, we can conclude the effect of some features on the Risk Flag with the consideration of the two conditions. The low values of the feature “x1” ,“x4”, “x5”, “x15_8”, “x21” have a higher probability of predicting target as a “Bad” (RiskFlag = 1) whereas their high values have a higher chance of predicting target as a “Good” (RiskFlag = 0). On the other hand, the low values of the feature “x18”, “x20”, “x22” have a higher probability of predicting target as a “Good” (RiskFlag = 0) while their high values have a higher chance of predicting target as a “Bad” (RiskFlag = 1).

Generally, the confidence interval of the features in the monotonicity condition became narrower. It implies that the effect of most features were less fluctuated in the condition.

Global Interpretation on Permutation importance on XGBoost.



Weight	Feature
0.0609 ± 0.0142	x1
0.0133 ± 0.0052	x15
0.0091 ± 0.0045	x15_8
0.0034 ± 0.0071	x4
0.0017 ± 0.0029	x20
0.0005 ± 0.0063	x8
0.0001 ± 0.0045	x5
-0.0004 ± 0.0025	x23
-0.0010 ± 0.0026	x2
-0.0024 ± 0.0022	x14
-0.0025 ± 0.0080	x18
-0.0025 ± 0.0039	x10
-0.0026 ± 0.0037	x21
-0.0026 ± 0.0012	x12
-0.0037 ± 0.0028	x22



Weight	Feature
0.1081 ± 0.0167	x1
0.0138 ± 0.0025	x15
0.0100 ± 0.0037	x5
0.0082 ± 0.0055	x4
0.0071 ± 0.0018	x15_8
0.0070 ± 0.0057	x8
0.0066 ± 0.0055	x18
0.0037 ± 0.0032	x14
0.0026 ± 0.0004	x12
0.0026 ± 0.0030	x23
0.0019 ± 0.0085	x20
0.0011 ± 0.0026	x2
0.0008 ± 0.0022	x22
-0.0015 ± 0.0053	x21
-0.0015 ± 0.0026	x10

Fig. 3.3 Permutation importance on the XGBoost

without monotonicity (left) and with monotonicity (right)

In the condition of non-monotonicity, we observed that the top three are the x1, x15, x15_8. For the other condition, the top two are the same (x1 and x15) but the x15_8 is replaced by x5 which is in the middle in the non-monotonicity condition. The rank of x18 also jumped from last five (non-monotonicity) to the 7th (monotonicity).

In the non-monotonicity condition, x22, x12 and x21 are relatively less important. Similar results found in the monotonicity condition with x10,x21,x22 the bottom three.

Another big contrast between two conditions is the coefficients of the features. We can see there are plenty of features with the negative coefficient ranging from x23 to x22 (rank 8 to rank 15) while only the last two (x21,x10) have negative values in the monotonic condition.

Local Interpretation on SHAP values on XGBoost

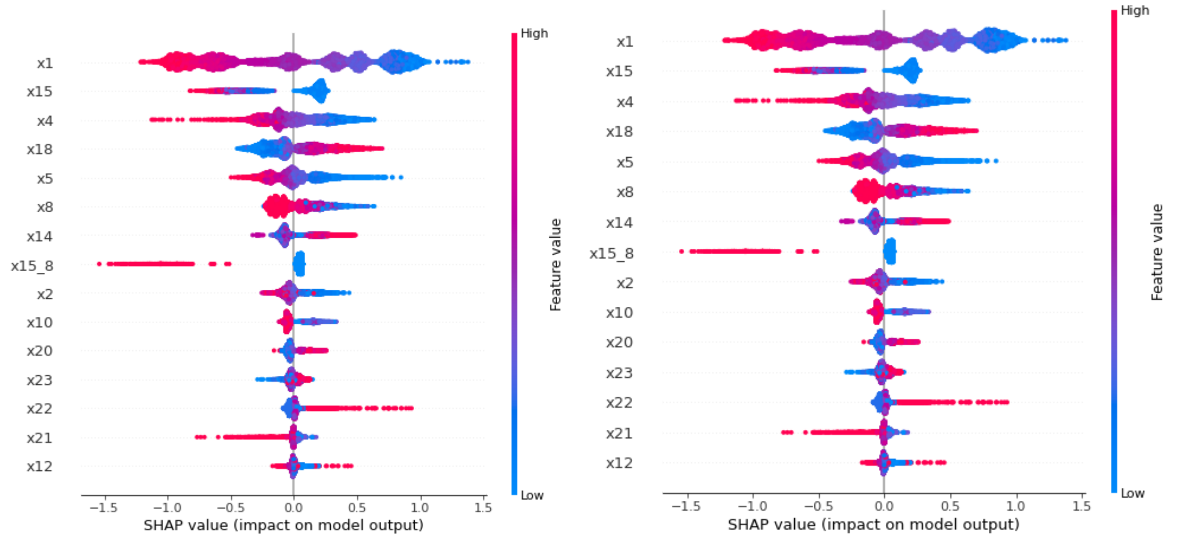


Fig. 3.4 SHAP on the XGBoost

without monotonicity (left) and with monotonicity (right)

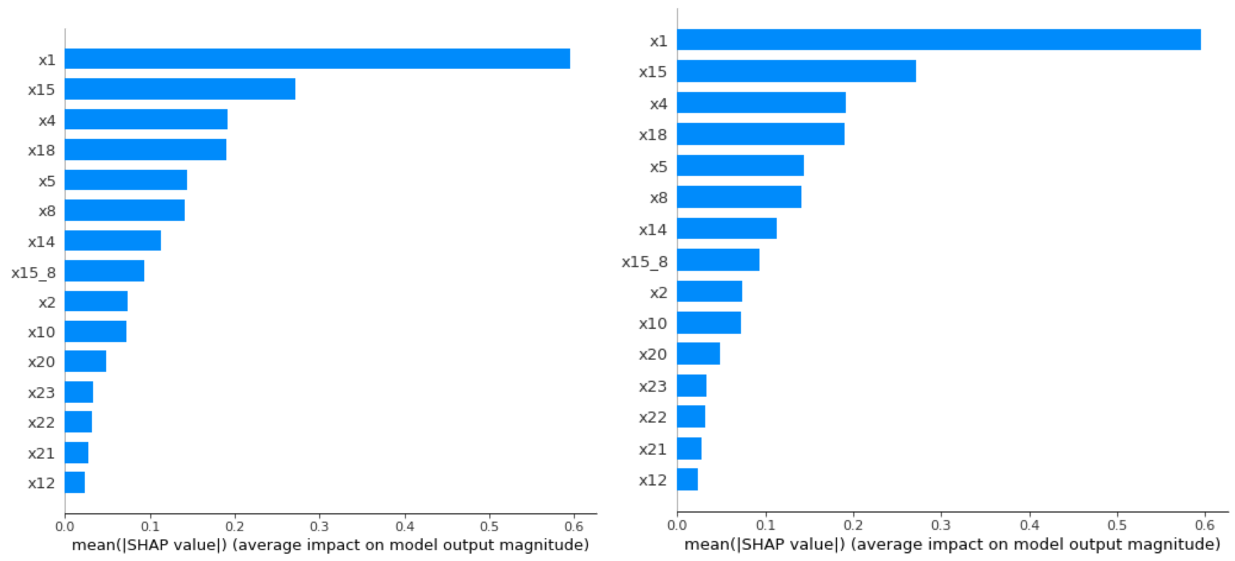
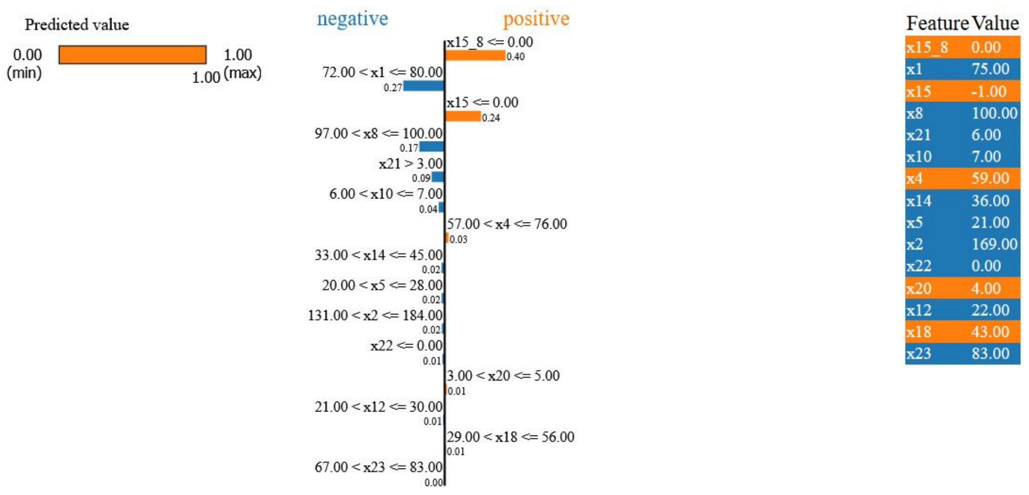


Fig. 3.5 SHAP bar charts on the XGBoost

without monotonicity (left) and with monotonicity (right)

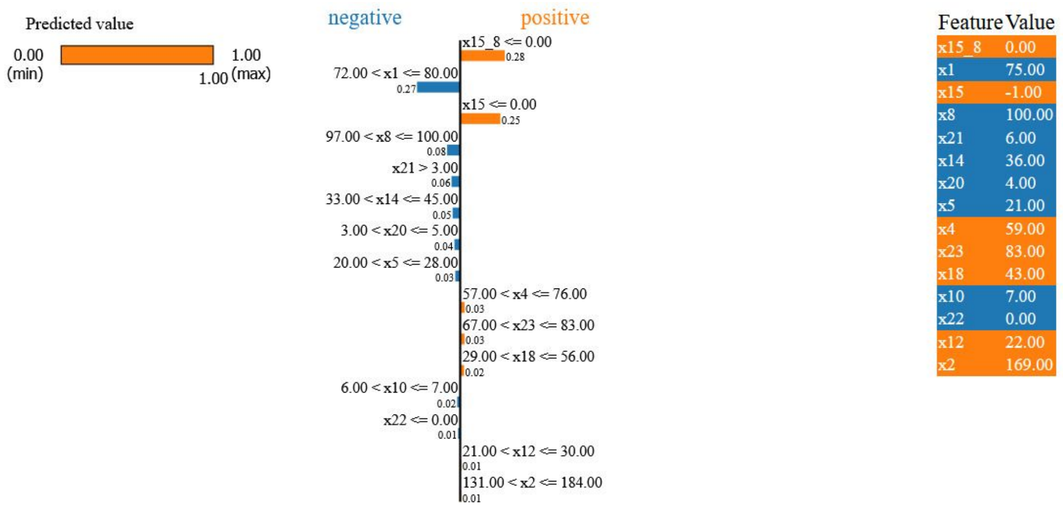
Both conditions have the same orders of importance on the SHAP values. The x1, x15, x4 are the top three while the x22, x21, x12 are those relatively less important.

Local Interpretation on LIME values on XGBoost



Feature	Value
x15_8	0.00
x1	75.00
x15	-1.00
x8	100.00
x21	6.00
x10	7.00
x4	59.00
x14	36.00
x5	21.00
x2	169.00
x22	0.00
x20	4.00
x12	22.00
x18	43.00
x23	83.00

Fig. 3.6 LIME value on the XGBoost without monotonicity



Feature	Value
x15_8	0.00
x1	75.00
x15	-1.00
x8	100.00
x21	6.00
x14	36.00
x20	4.00
x5	21.00
x4	59.00
x23	83.00
x18	43.00
x10	7.00
x22	0.00
x12	22.00
x2	169.00

Fig. 3.7 LIME value on the XGBoost with monotonicity

Similar rank results are found in the two conditions for the top three: x15_8,x1 and x15. However the bottom two in the non-monotonicity condition (x18, x23) are replaced by the x2,x12 in the monotonicity condition. We can observe that the x18, x23 jumped to a higher rank in the monotonicity condition.

Comparing with the result of the permutation importance method, LIME can give the similar result on the top three important features in XGBoost (permutation importance: x15_8, x15 and x1)

However, different from the result of permutation importance and SHAP interpretation, there is a significant raise in the rank of x15_8 and x15 pair, especially for x15_8. The effect of x15 and x15_8 are shown to be more significant in LIME, probably because of the collinearity between the two variables. This shows that LIME may consider the interaction effect and the collinearity more when computing the significance of features.

One thing worth mentioning x15_8's feature values dropped from 0.4 (non-monotonicity condition) to 0.28 (monotonicity condition). The x15_8 became less important in the monotonicity condition.

Comparison of GAM and XGBoost by local interpretation

	White-Box (Logistic GAM)	Black-Box (XGBoost)
Monotonic Constraint		
Accuracy	With constraint is larger than without constraint	
Individual Value (PDP vs LIME)		
x1	PDP shows a decreasing trend from around 70 to 80	A similar performance is found in LIME. $72 < x1 \leq 80$ has a more significant effect

x8	PDP shows a decreasing trend from around 70 to 80	$97 < x1 \leq 100$ has a more significant effect
x15	PDP shows a decreasing trend from 0 to around 10	≤ 0 has a more significant effect
x15_8	PDP shows a decreasing trend from 0 to 1	≤ 0 has a more significant effect

For the monotonic constraints accuracy comparison, we can see the models with monotonicity have a better performance than that without monotonicity in both white and black-model categories. We believe that the result can be explained with our PDP. In the White-Box model, multiple straight sections are observed. It may be indicated that the White-Box model is less affected by local fluctuations of data and thus had a less chance of overfitting. For the counterpart, the unfitted expectation of the performance before and after the constraint is found in the PDP. It means that the overfitting issue might exist.

We also have some interesting insights into some individual values. The above table has shown four examples of the comparison. Generally, the trend shown in the PDP covers the range mentioned in the LIME interpretation. However, the LIME interpretation of the Black-Box model gives a more specific range of the values that has a more significant effect. One obvious example is the x8 in the above table. It is noticed that the lowest values of x15 and x15_8 are both "0" which mean the range of ≤ 0 (i.e. = 0) also included in the range stated in the PDP. It may suggest that XGBoost may depend on the values with that specific range.

Conclusion

We found that the White-Box model (Logistic GAM with spline order = 2 in non-monotonicity and Logistic GAM with spline order = 2 in monotonicity) has a better performance on the accuracy score than the Black-Box model (XGBoost). Such an exciting finding can break people's stereotype on the "Best Accuracy Award" on the Black-Box model, more specifically is the XGBoost itself, which has always included the winning solutions in multiple machine learning competitions starting from 2015 (Nielsen, Didrik, 2016). With the proper feature engineering and model tuning techniques based on the understanding of the dataset distribution and the feature meaning in the real world, a simple model like the Logistic GAM with degree-2 splines can triumph among the two classes comparison by giving a higher accuracy score and its decent interpretation nature. We hope our exploration and the result can contribute some efforts in the further study in the Interpretable Machine Learning academic field.

Acknowledgement

We would like to thank the Lecturer Dr. Aijun Zang and the two tutors Mr. Yayang Xu, Mr. Yifeng Guo for your effort on teaching the course and the help on our project especially for the suggestion on the comparison between the White-Box and Black-Box model.

Reference

Didrik Nielsen (2016). Tree Boosting With XGBoost - Why Does XGBoost Win "Every" Machine Learning Competition?. NTNU Open. http://hdl.handle.net/11250/2433761.

Heinze, G., Wallisch, C., & Dunkler, D (2018, May). Variable selection - A review and recommendations for the practicing statistician. Biom J. 60(3): 431-449. Retrieved December 06, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5969114/

Kurzelewski, T., & Radzikowski, T. (2020, October 14). XAI Stories. Retrieved December 06, 2020, from https://pbiecek.github.io/xai_stories/story-heloc-credits.html