

# BT4016 Assignment 2

## CREDIT RISK ANALYSIS

Lim Jin Ming, Jeremy Denzel (A0172720M)

Sung Zheng Jie (A0168188M)

## BT4016 Assignment 2

### ***Our Approach:***

This mainly address the use of validation set and test set and our approach to it.

We define the validation set as a means to not only benchmark whether the model is overfitting but also to finetune the model performance.

On the other hand, we see the test set as something that will always remain unobserved and untouched. Hence, the test set will act as a baseline to see how our model performs on unseen instances.

	Uses
Validation Set	- Check for overfitting, Finetune model performance (find best lambda regularisation in question 5; hyperparameter tuning in question 8)
Test Set	- Evaluate model performance on unseen data (no data leakage)

### Full model

1. (3 points) Apply three different techniques (logistic regression, tree classification, XGBoost) to all attributes in the dataset (FULL-MODEL) to predict loan default.

Compute the accuracy and MCC of each measure to compare how good each technique is at using these attributes at predicting default, using a test set.

### Cross Validation Set (Average)

	Logistic Regression	Tree Classification	XGBoost
<b>Accuracy</b>	87.00%	87.76%	88.93%
<b>MCC</b>	64.44%	65.89%	63.43%

### Test Set

	Logistic Regression	Tree Classification	XGBoost
<b>Accuracy</b>	86.48%	88.15%	90.32%
<b>MCC</b>	63.20%	64.16%	67.72%

2. (2 points) Describe the strategy you took in terms of training the data and selecting the appropriate parameters to best avoid overfitting. This should also be applied in the above and subsequent questions for consistency. \*Hint - you may further divide the training data\*

### Data used for Training

#### 1. SMOTE Oversampling instead of Resampling on existing dataset

SMOTE addresses the unbalanced training set and reduces information leakage by creating synthetic data points which are similar to the existing ones using nearest neighbor classification instead of duplicating existing data points. This, therefore, helps to reduce overfitting.

#### 2. Oversample only on the training set rather than on the entire dataset

If oversampling is done before splitting into the training and test sets, there will be information leakage from the test set into the training set. SMOTE algorithm works by using nearest neighbors of observations to create synthetic data.

When SMOTE is applied to the entire data, there is a chance that the nearest neighbors of minority class observations in the training set are in the test set. This means that information in the test set is partially captured by the synthetic data in the training set.

As a result, the model will be able to better predict the test set values than a completely unseen data, causing an overfitting problem.

#### 3. Stratified Cross-Validation

The goal of cross validation is to assess how the model will generalise to an independent data set. **This can be evaluated by looking at the deviation between the validation and test error.** If there is a large deviation, then there is a high chance of overfitting.

Specifically, in k-fold cross validation, every data point gets to be in a validation set exactly once, and gets to be in a training set k-1 times. This significantly reduces underfitting and as we are using most of the data for fitting, and also significantly reduces overfitting as most of the data is also being used in the validation set.

Stratification is also factor into the cross validation. Stratification is a technique where we rearrange the data in a way that each fold has a good representation of the whole dataset. This approach ensures that one class of data is not overrepresented especially when the target variable is unbalanced.

#### 4. Oversample during Cross-Validation instead of before Cross-Validation

SMOTE oversampling is done on all training folds during the process of cross validation rather than on the entire training set before cross validation. This is achieved by creating a SMOTE pipeline at each iteration. This will give us a more representative cross validation results when we compare it to the test

results. This is because the test set does not contain any SMOTE oversampled data points and only data from the original dataset. Therefore, our validation set should not contain SMOTE oversampled data points as well.

**Note: We will be applying all these concepts explained above to qns 8 as well.**

Select the best parameters for each model

The best parameters to prevent overfitting for each model are determined by intuition. From there, we repeatedly test multiple set of values and find the best set that greatly reduces the validation error and overfitting.

*Logistic Regression*

Parameters	Argument	Explanation
penalty	l2	L2 Regularisation is used to reduce overfitting by penalising the weights of the least important features to values close to 0.  Note: 'L1' is not used as the question stated to create a full model and L1 Regularisation results in a reduced model by penalising the coefficients of the least important features to 0.
C	1	C is the inverse value of regularization factor $\lambda$ i.e. $C = \frac{1}{\lambda}$ The smaller the value of C, the larger the value of $\lambda$ . This will increase the regularisation strength, which in turn adds a stronger penalty term in the error function to reduce overfitting.  We set C = 1 because it is a value that is neither too high nor too low, thus ensuring that our model does not suffer from either an overfitting or underfitting problem.

*Tree Classifier*

Parameters	Argument	Explanation
criterion	Entropy	Entropy is used to measure the information gain from a split  Information gain basically measures how much "information" a feature gives us about a class.. Given a random variable, the expected amount of information generated by its distribution can be expressed as $H = - \sum_i p_i \log(p_i)$
max_depth	7	The max depth parameter is used to determine the maximum

		<p>depth of the decision tree. In this case it is used to prevent overfitting which could happen when the decision tree expands fully and tries to maximise accuracy or any other metrics used.</p> <p>In this case, the maximum depth of the fully grown decision tree without pruning is 14. After trying a few values, we have arrived at the value of 7 which gives us a good average accuracy and mcc from the cross-validation and test results.</p>
min_samples_split	30	<p>This parameter controls the minimum number of samples required to split an internal node. It controls for overfitting by preventing a split from occurring when the number of samples is too small.</p> <p>We have set a value of 30 after multiple trials and find that it gives a good accuracy and mcc in the validation and test set while preventing overfitting.</p>
min_samples_leaf	10	<p>This parameter sets the minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This also controls for overfitting by preventing the leaf node from having too little data points.</p> <p>We have set a value of 30 after multiple trials and find that it gives a good accuracy and mcc in the validation and test set while preventing overfitting.</p>
min_impurity_decrease	0.005	<p>A node will be split if this split induces a decrease of the impurity greater than or equal to this value.</p> <p>The weighted impurity decrease equation is the following:  <math display="block">N_t / N * (impurity - N_{t_R} / N_t * right\_impurity - N_{t_L} / N_t * left\_impurity)</math> </p> <p>where N is the total number of samples, N_t is the number of samples at the current node, N_t_L is the number of samples in the left child, and N_t_R is the number of samples in the right child.</p> <p>N, N_t, N_t_R and N_t_L all refer to the weighted sum, if sample_weight is passed.</p> <p>This criteria prevents overfitting by preventing unnecessary splits which does not result in a significant information gain.</p> <p>We have set a value of 30 after multiple trials and find that it gives a good accuracy and mcc in the validation and test set while preventing overfitting.</p>

--	--	--

### XGBoost

Parameters	Argument	Explanation
max_depth	7	<p>It controls the depth of the tree. Larger the depth, the more complex the model; higher chances of overfitting.</p> <p>We referenced the max_depth from the decision tree and after trying a few values, we have arrived at the value of 7, which gives us a good average accuracy and mcc from the cross-validation and test results.</p>
reg_alpha	1	<p>reg_alpha is the L2 regularization on the weights. Regularization adds a penalty term to the objective function and controls for model complexity to prevent overfitting.</p> <p>The greater the alpha regularisation parameter, the greater the penalty to the weights. After multiple trails, we find that a reg_alpha value of 1 gives a good accuracy and mcc score while preventing overfitting.</p>
reg_lambda	2	<p>reg_lambda is the L1 regularization on the weights. Likewise, this adds a penalty and controls for model complexity.</p> <p>The greater the lambda regularisation parameter, the greater the penalty to the weights. After multiple trails, we find that a reg_lambda value of 2 gives a good accuracy and mcc score while preventing overfitting.</p>
gamma	1	<p>Gamma determines the minimum loss reduction. It controls regularization and prevents overfitting. The higher the value, the higher the regularization.</p> <p>We set the gamma = 1 so that regularisation is enabled and this value gives a good average accuracy and mcc from the cross-validation and test results.</p>
min_child_weight	5	<p>Min_child_weight refers to the minimum sum of instance weight. If the leaf node has a minimum sum of instance weight lower than the min_child_weight, the tree splitting stops.</p> <p>We set the min_child_weight = 5 to prevent overfitting by ensuring that there are not too many splits in the tree.</p>

### Reduced Model

3. (2 points) Which loan attributes *do you believe* are the most informative? Please use your **knowledge and intuition** to choose 10 of the attributes available, and explain why you chose these attributes. Let's call this the REDUCED-MODEL.

### 10 Selected Features

"int\_rate", "total\_pymnt", "total\_pymnt\_inv", "revol\_bal", "dti", "out\_prncp", "pub\_rec\_bankruptcies", "chargeoff\_within\_12\_mths", "tot\_cur\_bal", "tot\_hi\_cred\_lim"

Features	Definition	Reasons
int_rate	Interest Rate on the loan	A bank will charge a higher interest rate if it thinks there is a lower chance that the loan will be repaid. The riskier the loan, the higher the interest rate. Hence, we included interest rate because a high interest rate also means that there could be more bad loans.
total_pymnt	Payments received to date for total amount funded	We believe that if a customer have a higher total_pymnt they will have a lower debt and there will be less tendency that they will default on a loan given that most of the amount of their loan are already paid off.
total_pymnt_inv	Payments received to date for total amount funded by investors	Similar to the explanation above, the higher the total payment paid for by the investors, the lower the current debt of the customers, and hence, the less likely they would default on their loan.  It can also be inferred that the greater the funds received from investors, the more confidence these investors have on the customer in bringing future value for them. Hence, these customers are less likely to default.
revol_bal	Total credit revolving balance	The total revolving balance is the amount of credit card spending that goes unpaid at the end of a billing cycle. We believe that customers with credit revolving balance or a high amount of credit revolving balance will tend to default. It could act as an early sign to potential customers that will not repay their loans.
dti	Debt-to-income ratio	Intuitively, the higher the debt-to-income ratio of the customers - meaning those with more debt in relation to their income, the more likely they will have trouble making their loan payments. Hence, a high debt-to-income could mean more bad loans.

out_prncp	Remaining outstanding principal for total amount funded	Basically, we believe the higher the amount of outstanding principal, the more likely that a customer will default on the loan as they face greater difficulty in paying it back.
pub_rec_bankruptcies	Number of public record bankruptcies	If a customer has a long history of bankruptcies, then it is very likely that the customer would continue the same pattern down the road. Hence, there is a high chance of defaulting.
chargeoff_within_12_mths	Number of charge-offs within 12 months	A charge-off is the declaration by a creditor that an amount of debt is unlikely to be collected. It occurs when a consumer becomes severely delinquent on a debt. We believe this is a good sign that the customer will likely default on a loan.
tot_cur_bal	Total current balance of all accounts	Intuitively, customers with a higher total current balance in their bank accounts should have the financial capability to pay off their loan. Therefore, if a customer has a low balance in their accounts, it could mean that they are facing some sort of financial issue and may not be able to pay off their loan.
tot_hi_cred_lim	Total high credit/credit limit	Banks normally give high risk lenders lower credit limit because of fear that they will not be able to pay off their loan. Therefore, this is a good indicator to identify customers with high risk of defaulting.



4. (2 points) Let's use the attributes you've selected for the reduced model to predict default, and check the performance of the resulting models. Apply the same three techniques (logistic regression, tree classification, and xgboost) to the REDUCED-MODEL (i.e. only using the attributes you selected) to predict loan default. Again, compute their accuracy and MCC.

Cross Validation Set (Average)

	Logistic Regression	Tree Classification	XGBoost
<b>Accuracy</b>	85.83%	86.88%	88.26%
<b>MCC</b>	65.06%	67.36%	64.58%

Test Set

	Logistic Regression	Tree Classification	XGBoost
<b>Accuracy</b>	84.64%	85.81%	87.15%
<b>MCC</b>	62.99%	64.37%	59.18%

Lasso-reduced model

5. (a) (2 points) Calculate the accuracy and MCC of the LASSO approach, and compare it to previous methods. Apply the LASSO to the default response variable (1 or 0), and the entire set of 70 attributes, and constrain the model to have at most ten attributes in it. Find the best lambda. Let's call this the LASSO-MODEL.

Which attributes are those?

inq_fi	int_rate	mths_since_rcnt_il	out_prncp	out_prncp_inv
tot_cur_bal	total_acc	total_pymnt	total_pymnt_inv	total_rec_int

Compute the method's accuracy and MCC using 5-fold cross validation

Cross Validation Set (Average)

	<b>LASSO</b>
<b>Accuracy</b>	85.46%
<b>MCC</b>	64.47%

Test Set

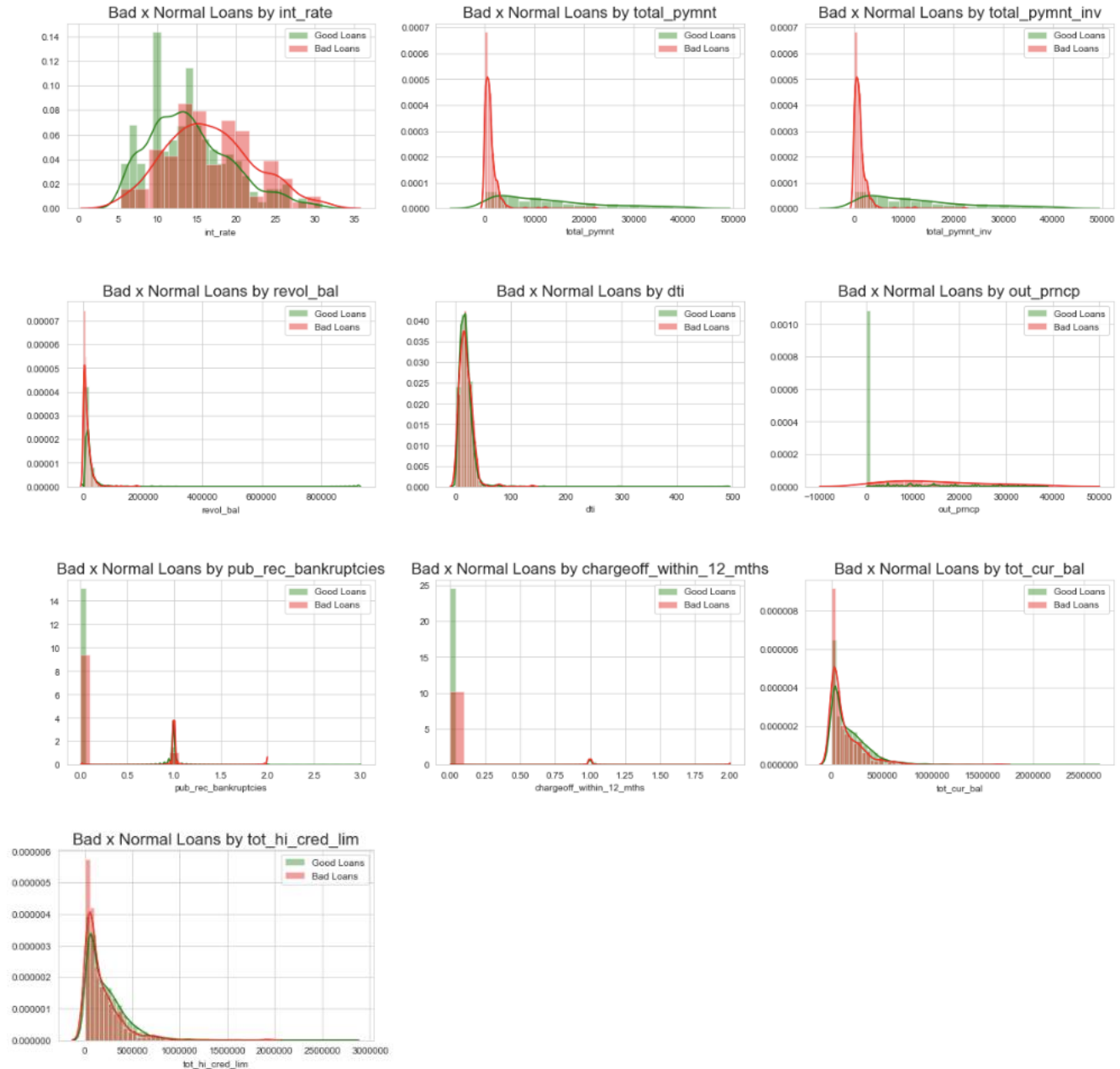
	<b>LASSO</b>
<b>Accuracy</b>	86.48%
<b>MCC</b>	51.72%

(b) (2 points) Did your intuition about the correct set of attributes in the REDUCED-MODEL match the results that you obtained from the LASSO-MODEL? Discuss.

<b>Our Selected Features</b>	<b>Lasso Selected Features</b>	<b>Do they match?</b>
int_rate	int_rate	Yes
total_pymnt	total_pymnt	Yes
total_pymnt_inv	total_pymnt_inv	Yes
revol_bal	inq-fi	No
dti	mths_since_rcnt_il	No
out_prncp	out_prncp	No
pub_rec_bankruptcies	out_prncp_inv	No
chargeoff_within_12_mths	total_acc	No
tot_cur_bal	tot_cur_bal	Yes
tot_hi_cred_lim	total_rec_int	No

Only 4 of our selected features in the REDUCED-MODEL match with the features in the LASSO-MODEL. The reason for the discrepancy can be attributed to the fact that the data does not exhibit the separation of customers (bad and normal loans) for each feature as described by our intuition, in particular for those wrongly identified features.

### Our Selected Features in the REDUCED-MODEL:



For example, we expected a high dti to have more bad loans and a low dti to have less bad loans - but this is generally not reflected in the data. We can instead see that there is no distinction in the customers in the `dti` distribution. When we look at the 4 features that are included in the LASSO Model, we can see that `total\_pymnt` and `total\_pymnt\_inv` shows a very different distribution for both

types of customers, while `int\_rate` and `tot\_cur\_bal` shows some differences especially at low values which could be used for identification.

All in all, the lasso model could probably identify some form of correlation and distribution of the features to distinguish the type of loans which if based on just our observation and intuition, cannot be identified.

### Understanding Model Performance

6. (1 point) What other metric might you use to compare the performance of the different models? Explain what this metric means and how it might be computed. For this question, you will not have to compute the metric.

In this specific business context, we would use the F1 score. The F1 score is calculated as  $2 * ((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$ . It is a balance between precision and recall. Precision is calculated as  $TP / (TP + FP)$  and recall is calculated as  $TP / (TP + FN)$  also known as the true positive rate.

Positive Label	Bad Loan
Negative Label	Normal Loan

A good F1 score indicates that there are **low false positives and false negatives**. In this scenario, it is important to have **low false positives** because a high false positive rate means that we wrongly classified good customers as loan defaulters. Doing so would result in a loss of revenue from not issuing the loan to these customers. It is even more crucial to have **low false negatives**. This is because we do not want to wrongly classify potential loan defaulters as good customers. If the lending club were to issue loans to these customers, they would suffer a much heavier loss as these customers will have a higher risk of defaulting.

Hence, when selecting the model, we would base it on the highest F1 score as another metric of evaluation.

7. (2 points) How do the results using the full set of attributes FULL-MODEL and LASSO-model compare to those that you personally selected REDUCED-MODEL to predict if a loan will default? What do you think is more important in this particular example, the set of attributes, or the classification techniques? Discuss.

### **Comparison of Results:**

When comparing the models, we will be primarily using the **Test Results** as a basis of comparison because this allows us to see how the models perform on a **truly unseen data**.

#### *(1) Full Model vs Reduced Model*

<b>Observation</b>	The Full Model performed better on the test set both in Accuracy (86.48% against 84.64%) and MCC (63.20% against 62.99%) compared to our personally selected reduced model.
<b>Explanation</b>	<p>This is because the Full Model considers all features and then uses 'l2' regularization to penalise the weak features by means of statistical measures (e.g. does this one feature lead to a clear separation of class?). The different weights assigned to each feature helps to minimise overfitting of the Full Model and increases its predictive power.</p> <p>On the other hand, our hand-picked features were based off domain knowledge and not through a thorough understanding of the data. As such, we could have excluded the significant features from our reduced model. This reduces the predictive power of our reduced model.</p>

#### *(2) Lasso Model vs Reduced Model*

<b>Observation</b>	<ul style="list-style-type: none"> <li>- Looking at the Test results, the Lasso Model performed slightly better than the Reduced Model in Accuracy (86.48% against 84.64%) but worse than the Reduced Model in MCC (51.72% against 62.99%).</li> <li>- Looking at the CV results, the Lasso Model performed fairly similar both in Accuracy (85.46% against 85.83%) and MCC (64.47% against 65.06%) compared to the Reduced Model.</li> </ul>
<b>Explanation</b>	<p>For the first observation, we do not have a conclusive explanation.</p> <p>However, for the second observation, we believe the similar CV performance could be because we managed to identify strong features such as 'total_pymnt' and 'out_prncp' that are able to separate the classes well. In addition, we are using 'l2' regularization for our Reduced Model, which will lower the weight of the coefficient of those poor features we have chosen.</p>

--	--

*(3) Full Model and Lasso Model vs Reduced Model*

<b>Observation</b>	When comparing the Test Accuracy, the Lasso Model and the Full Model perform equally well, while our Reduced Model has the lowest accuracy.
<b>Explanation</b>	<p>Both the Full Model and Lasso Model have similar performance results because they both consider the full feature set and then apply a penalty to the weak features - (reduce the coefficient of weak features close to 0 for `l2` in Full Model; penalise the coefficient of weak features to equal to 0 for `l1` in Lasso Model. This minimises overfitting and allow them to have a much better predictive performance.</p> <p>As mentioned above, we could have accidentally excluded some significant features in our reduced model, which explains the relatively poor performance in test accuracy.</p>

***Set of Attributes or Classification Techniques:***

In our opinion, we feel the classification technique is more important. While the set of attributes used contributes largely to the performance of the classifier, the classifier has its own inbuilt parameters to select the best set of attributes eg `l1` and `l2` regularization/ Max Features. If we understand the parameters of a model well, we could utilise it to select the best set of attributes for us. In addition, it is evident from the results of the Full Models and Reduced Models, that classification methods are more important, as given the same set of attributes, the XGBoost always outperforms the other 2 classifiers.

### Open Ended Tuning Problem

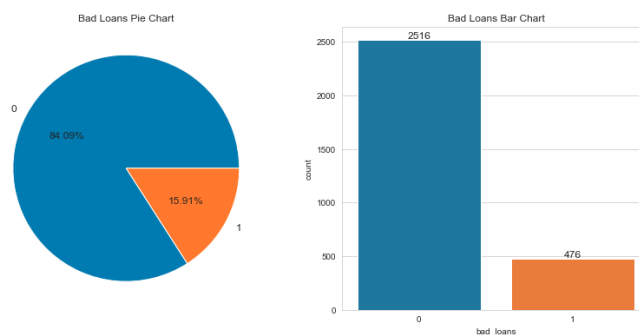
8. Your manager has given you a task of tuning a model that could be operationalized for predicting loan defaults. He has asked you to use the given full dataset and has left it up to you to decide how to use it. Your job is therefore to come out with a model that can generalize well, hence overfitting to data is of concern as the dataset is relatively small.

(a) (3 points) Describe your validation and testing procedure, noting down what techniques worked for you and what did not. Tabulate any results across the different tuning attempts and note down the various techniques you tried (e.g. feature engineering, different base models etc) in a separate column.

### 1. Exploratory Data Analysis

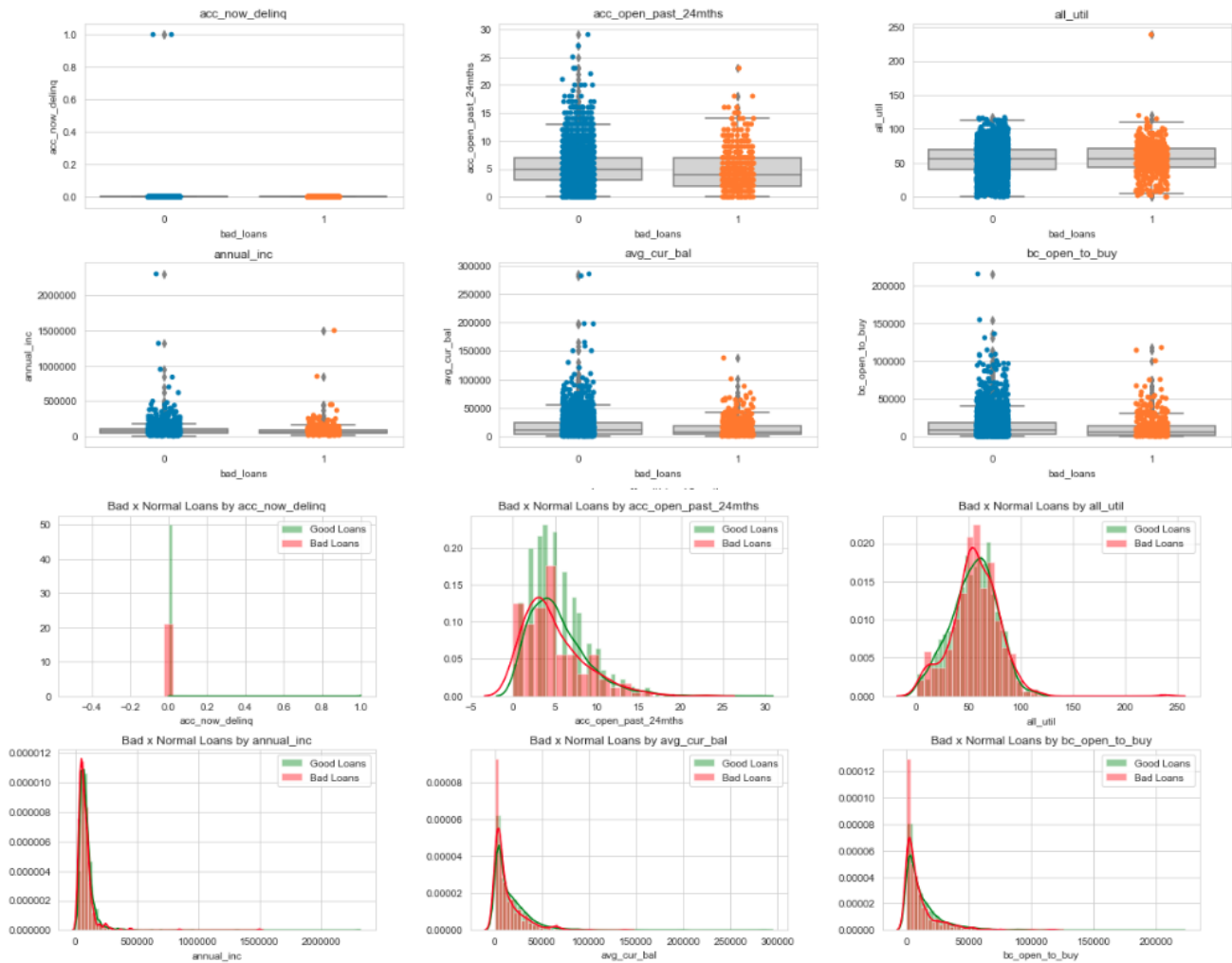
We perform some Exploratory Data Analysis to gather some insights from the data which will be useful for Modelling later.

#### Overall Bad Loans Rate



The breakdown of the classes of loan type in the dataset shows that it is an unbalanced dataset with more records of good loans compared to bad loans.

## Analyse the distribution of the variable by class



We then plotted the distribution of each variable by class. We realise there are some features which distinctly separates the class (bad and normal loans). We believe these features will be more statistically significant. The full visualisations can be found in the notebook.

## Analysis by Income Category

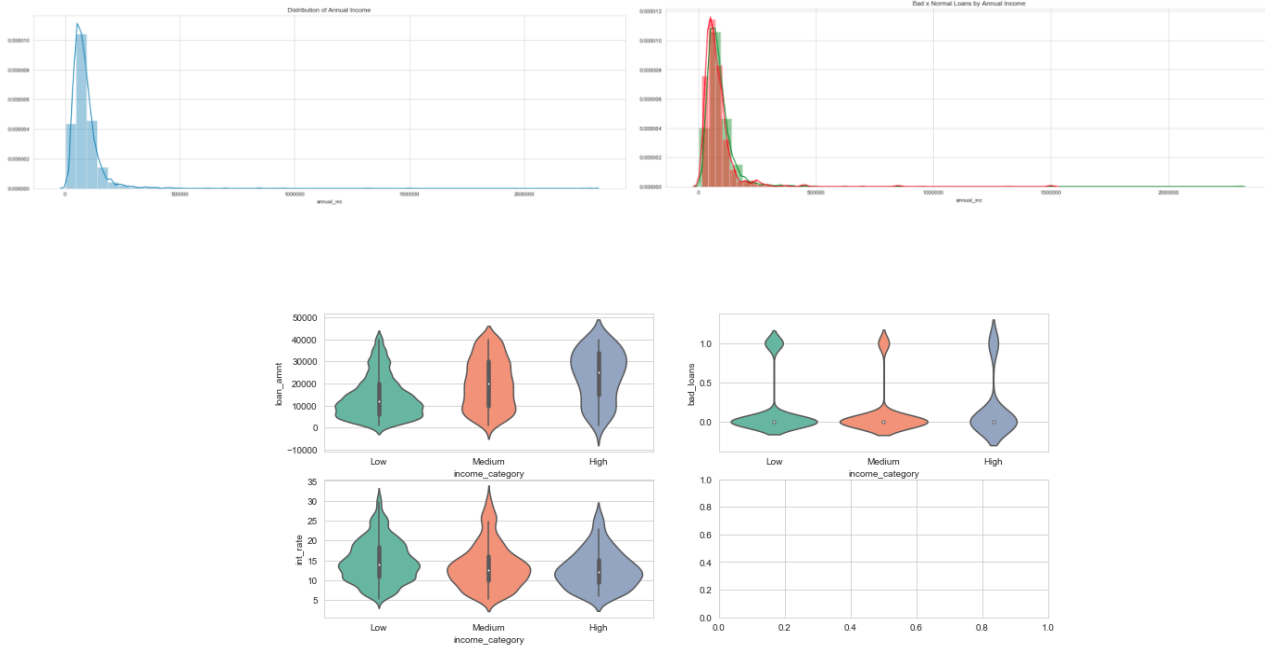
We then perform some analysis based on income category that we defined..

- **Low income category:** Borrowers that have an annual income lower or equal to 100,000 usd.
- **Medium income category:** Borrowers that have an annual income higher than 100,000 usd but lower or equal to 200,000 usd.
- **High income category:** Borrowers that have an annual income higher than 200,000 usd.

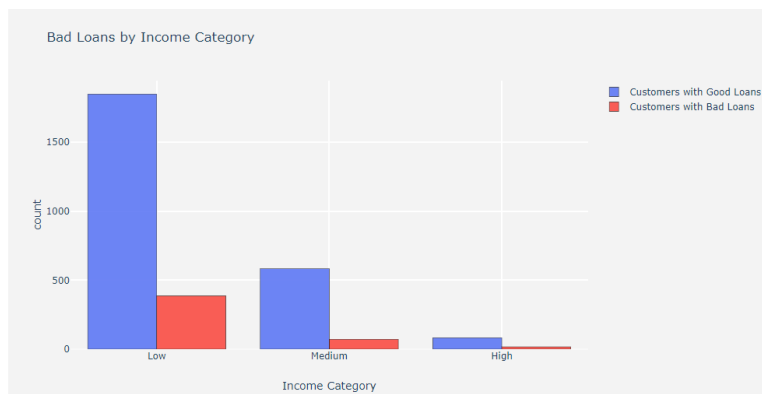


## Summary:

- Borrowers that made part of the **high income category** took higher loan amounts than people from **low** and **medium income categories**. People with higher annual incomes are more likely to pay loans with a higher amount.
- Borrowers with a lower income had on average **higher interest rates** while people with a higher annual income had **lower interest rates** on their loans.



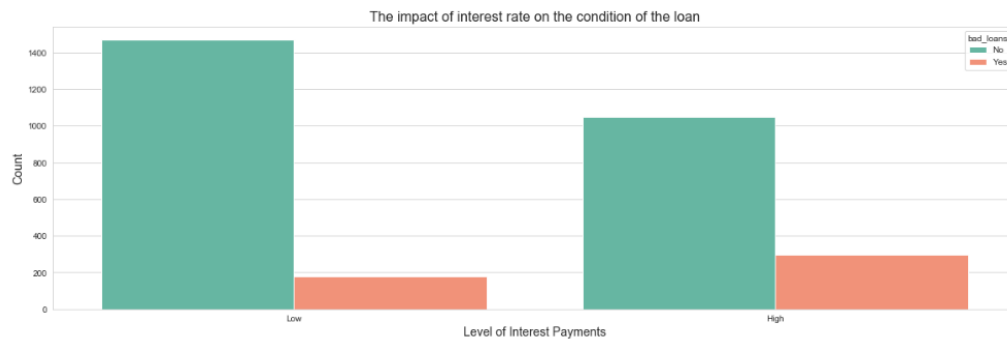
The higher the income, the higher the loan amount. The higher the income, the lower the interest rate.



There are more customers with bad loans in the low income category as compared to the other categories. However, this may not be reflected when comparing the proportion of the bad loans in each income category.



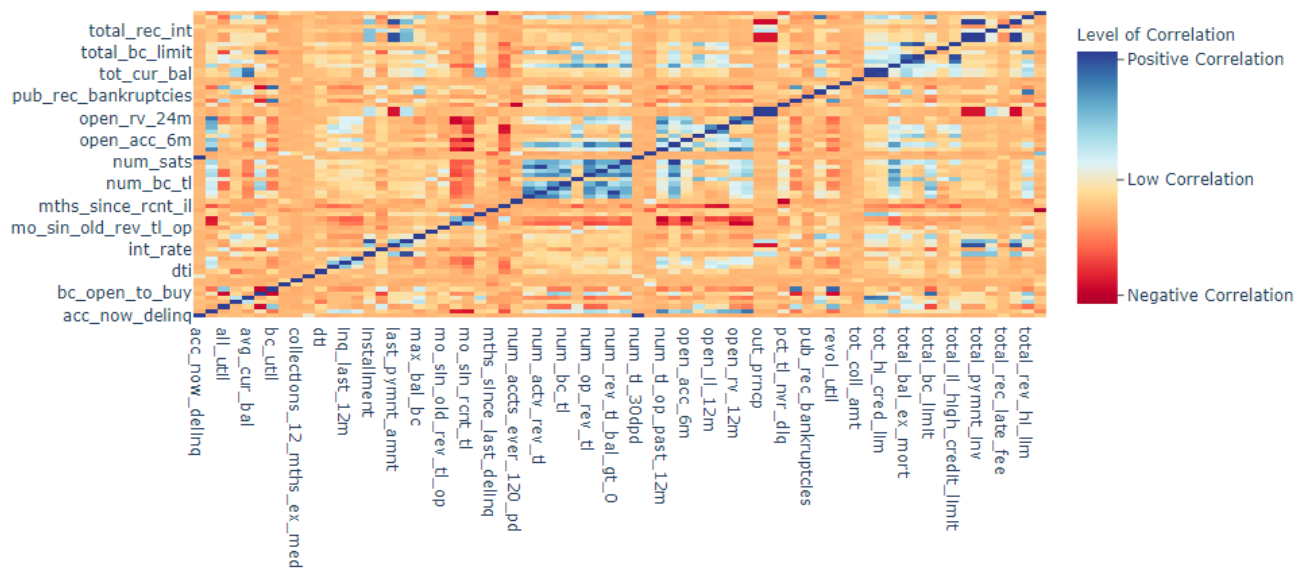
There is no clear pairwise correlation among the loan\_amount, int\_rate and annual\_income. Nonetheless, we can see a clear separation in classes (bad and normal loans) when analysing the relation between loan\_amnt and annual\_inc as well as int\_rate and annual\_inc. This is due to the fact that there are more bad loans in the lower income group.



Loans that have a **high interest rate** are more likely to default.

## Analysis by Correlation of Numerical Variables

Correlation Heatmap



It is difficult to analyse the correlation of all 70 variables. Instead, we will look at the correlation heatmap of the more significant features later on in the notebook.

## 2. Dataset Preparation

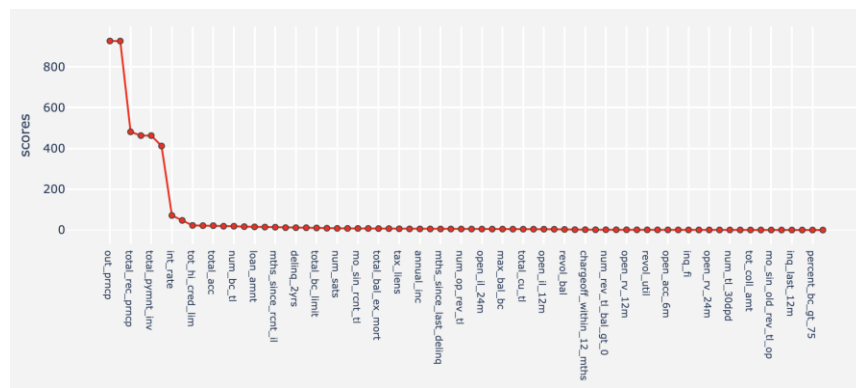
### - Train-Test Split

Stratified sampling is done to reduce variability between the train and test set, thereby maintaining the data integrity of the original dataset which helps create a more representative model.

### - Standardisation

StandardScaler is used. The scaler is first fitted on the train set before it is used to scale the test set. This ensures consistency and makes it possible to evaluate if the model can generalize well.

## 3. Feature Selection using $f_{\text{classif}}$



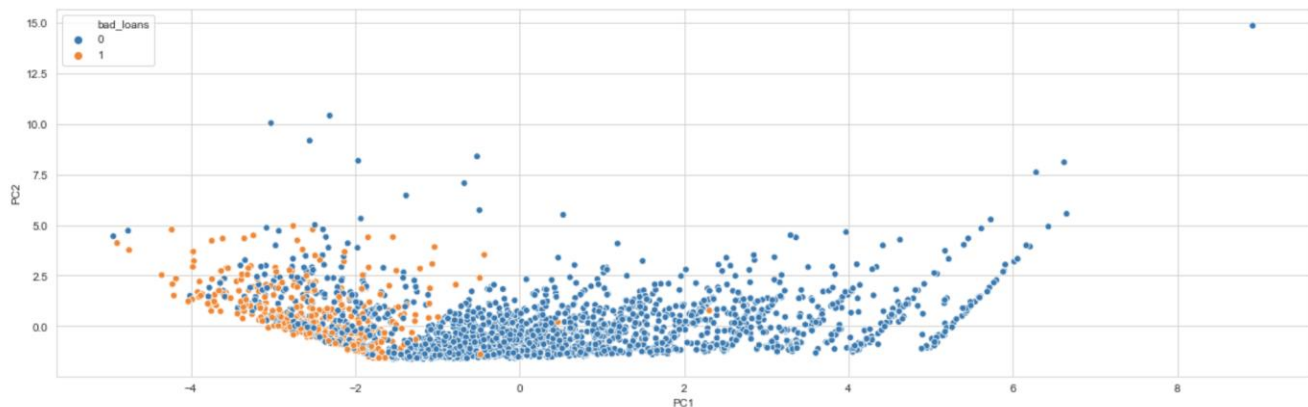
Feature selection is done after Train-Test split and using only the train set as it prevents information leakage (if performed with the test set as well) and gives a more representative and fair result when we perform our final prediction on the test set.

To select important features, we use the anova  $f_{\text{value}}$  scoring function. We evaluated the significance of each feature at the 5% significance level. The higher the  $f_{\text{value}}$  score, the more significant the feature is. Statistically significant features are those that have p-values less than 0.05 at the 5% significance level. We then take the top 10 out of those significant features.

## 4. Dimensionality Reduction

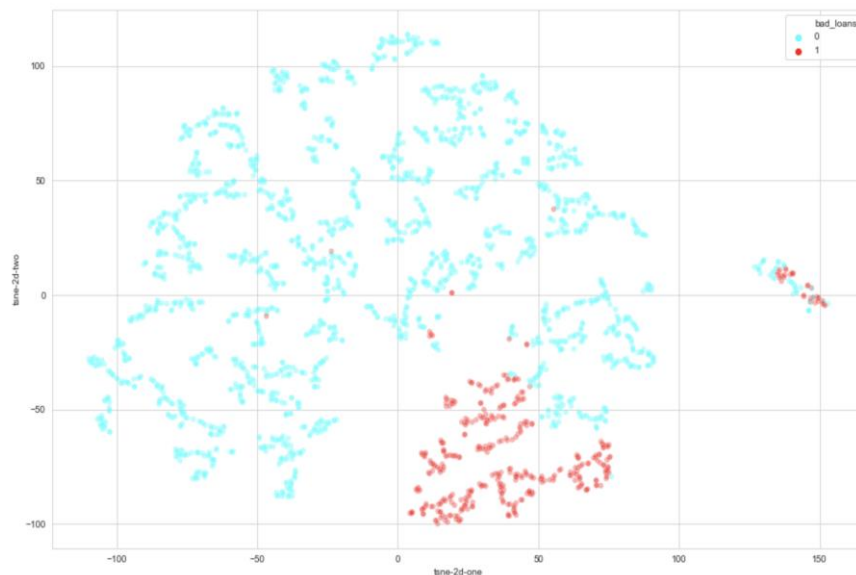
The objective of this section is to see whether the class is separable (bad and normal loans) when a given set of features (top 10 significant features) is used. In this section, we employed two statistical tools: (1) Principal Component Analysis (PCA) and (2) t-SNE.

### - Principal Component Analysis (PCA)



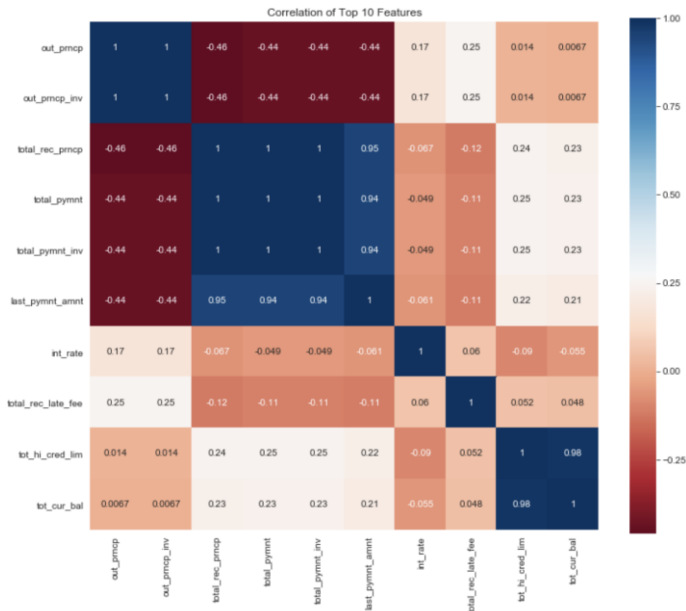
PC1 and PC2 accounts for a combined variation of 70%. The explained variation as well as the features breakdown/correlation of each principal component can be found in the notebook. From the graph, we can see that the given set of features can distinctly separate the 2 classes well.

### - t-SNE (t-Distributed Stochastic Neighbor Embedding)



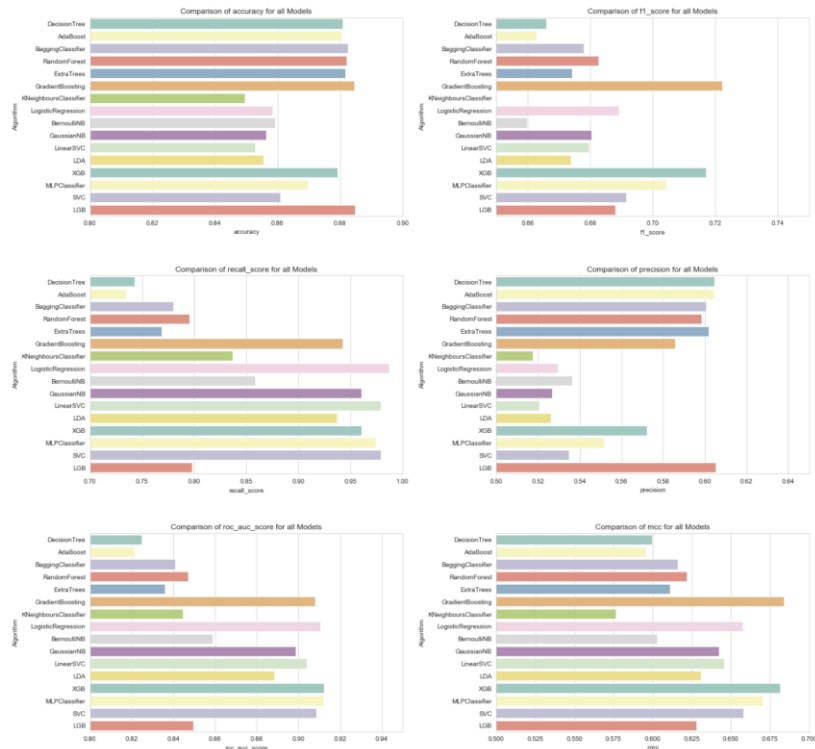
t-SNE is a non-linear technique for dimensionality reduction which is another popular tool used for visualizing high dimensional data. t-SNE algorithm works by constructing a probability distribution over pairs of high-dimensional objects where similar objects will have a high probability of being picked while dissimilar points have a smaller probability of being picked resulting in the different clusters. Good parameters must be chosen in order to general distinct cluster. In our case, we are able to show 2 distinct clusters with minor overlaps, this shows that the 2 classes can be distinctly separated with the correct parameters/features used.

## 5. Correlation of top 10 features



The correlation coefficient of the top 10 features is plotted out. A few features were found to be strongly correlated with each other and have a perfect correlation of 1. They are 'out\_prncp' and 'out\_prncp\_inv'. Another 4 features that were found to be strongly correlated are 'total\_rec\_prncp', 'total\_pymnt', 'total\_pymnt\_inv' and 'last\_pymnt\_amnt'. After removing the correlated features, we are left with the top 5 features - 'out\_prncp', 'total\_rec\_prncp', 'int\_rate', 'total\_rec\_late\_fee', 'tot\_hi\_cred\_lim'.

## 6. Modelling



A total of 16 different Machine Learning Models is then tested:

Decision Tree	Adaboost	BaggingClassifier	Random Forest
Extra Trees	Gradient Boosting	K Neighbours Classifier	Logistic Regression
Bernoulli Naive Bayes	Gaussian Naive Bayes	Linear Support Vector Classifier	Linear Discriminant Analysis
XGB	MLPClassifier	Support Vector Classifier	Linear Gradient Boosting

The parameters of these models are unchanged when they are fitted with the training dataset. A 5-fold cross validation is then done and their Accuracy, F1 Score, Recall, Precision, ROC AUC and MCC is printed and graphed out for comparison. This models serve as a baseline guide for us to choose the best models to work on/tune further.

#### - Cross Validation

Likewise, SMOTE oversampling is done on all training folds during the process of cross validation rather than on the entire training set before cross validation. This is achieved by creating a SMOTE pipeline at each iteration. This will give us a more representative cross validation results when we compare it to the test results. This is because the test set does not contain any SMOTE oversampled data points and only data from the original dataset. Therefore, our validation set should not contain SMOTE oversampled data points as well.

Similarly, stratification is also factor into the cross validation to ensure that one class of data is not overrepresented especially when the target variable is unbalanced.

## 7. SMOTE Oversampling on Train-Set

The SMOTE oversampling is then done on the entire training set. This oversampled training set will be used solely to fit the model and then evaluate it on the test set

## 8. Validation Curve

After choosing the top models to tune and improve on. We work on identifying the parameters in each model which has the most impact/effect on the model. This is done through plotting the cross validation score from tuning each specific parameter in each model. The plots of the cross validation curve can be found in the next section.

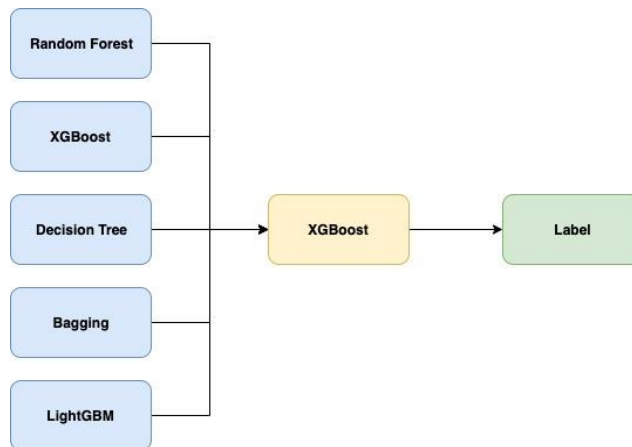
Note: For the cross-validation, we will be using the non-oversampled training set. It is only during each fold where the training set gets oversampled and the validation set does not. This minimises data leakage and any potential for overfitting.

## 9. Bayesian Optimisation Tuning

After identifying and scoping the important parameters to be tuned and its effective range based on the validation curve in the previous step, we utilise Bayesian Optimization to tune our model. Bayes Optimization focuses on finding the minimum/maximum of an objective function, therefore it tends to require fewer iterations to find the most optimal set of parameter as compared to GridSearch and RandomSearch and is shown to be equally effective. We then proceeded to perform several rounds of Bayesian Optimization on each model, each time improving and scoping the range of the parameters based on the results from the previous attempts.

Likewise, Bayesian Optimization utilises cross validation to compute the validation results. For cross validation, it will take in the non-oversampled training set, and then during each fold, the training set gets oversampled by SMOTE but the validation set does not. This minimises data leakage and any potential for overfitting.

## 10. Stacking



After performing Bayesian Optimization on each of the model we collate the results into the table below and selected the top 5 models to form our base model for stacking. The stacking infrastructure is shown in the picture. Our top 5 base/L1 models are - Random Forest, XGBoost, Decision Tree, Bagging and LightGBM. This 5 models are chosen solely based on their F1 cross validation score without looking at their F1 test score in order to preserve the integrity and accuracy of the final result.

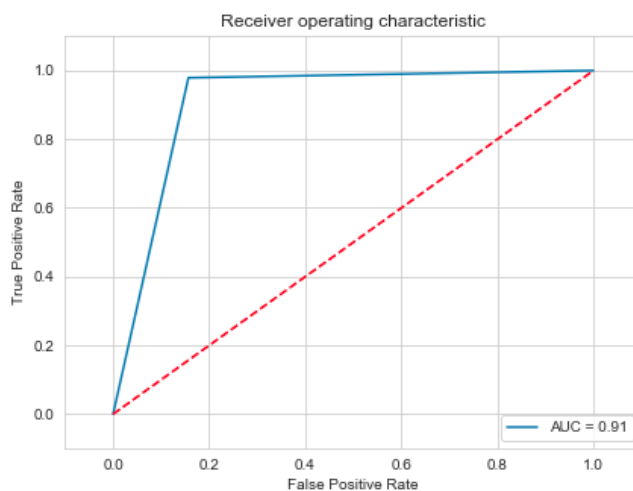
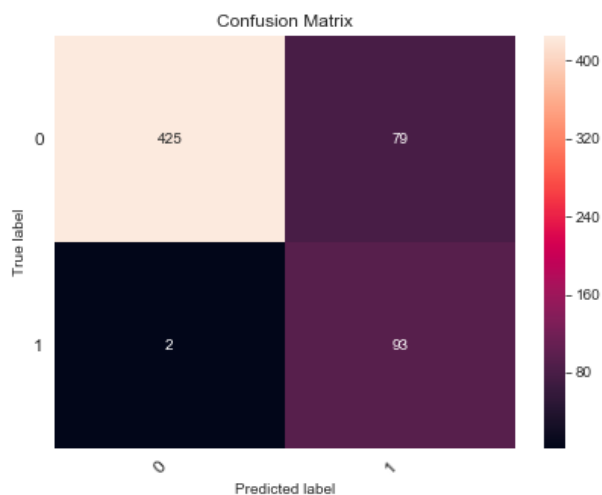
For our L2 Model, we have chosen to use XGBoost which has consistently good result. The final L2 Model model is also chosen solely based on the F1 cross validation score without looking at the F1 test score in order to preserve the integrity and accuracy of the final result.

The prediction results of each L1 model will then be used as train and test set for the L2 model. Likewise, we applied SMOTE oversampling on the entire training set. This oversampled training set will be used to fit the model to test its performance on the test set while the non-oversampled training set will be used for cross-validation.

We also finetuned the L2 model using Validation Curve and Bayesian Optimisation.



## 11. Final Results



Accuracy: 0.8647746243739566  
 Sensitivity: 0.9789473684210527  
 Precision: 0.5406976744186046  
 f1 Score: 0.696629213483146  
 ROC\_AUC\_Score: 0.9111006683375104  
 MCC: 0.6638608969649519

## L1 Base Models:

Random Forest:

Attempt	Parameters Tune	Best Parameter	CV Results (F1)	Test Results (F1)
0	NA	Base Parameters	68.27%	
1	{                     "max_depth": scope.int(hp.quniform('max_depth', 5, 15, 1)),                     "max_features": scope.int(hp.quniform('max_features', 3, 5, 1)),                     "min_samples_split": scope.int(hp.quniform("min_samples_split", 10, 50, 10)),                     "min_samples_leaf": scope.int(hp.quniform("min_samples_leaf",	{                     'bootstrap': True,                     'criterion': 'gini',                     'max_depth': 14.0,                     'max_features': 5.0,                     'min_samples_leaf': 10.0,                     'min_samples_split': 40.0,                     'n_estimators': 100.0                 }	71.58%	68.97%

	<pre> 10,30,10)), "bootstrap": hp.choice("bootstrap", [False,True]), "n_estimators" : scope.int(hp.quniform("n_estimators", 50,300,50)), "criterion": hp.choice("criterion", ["gini","entropy"])} </pre>			
2	<pre> {"max_depth": scope.int(hp.quniform('max_depth', 5, 15, 1)), "max_features": scope.int(hp.quniform('max_features', 3, 5, 1)), "min_samples_split": scope.int(hp.quniform("min_samples_split", 10, 50, 5)), "min_samples_leaf": scope.int(hp.quniform("min_samples_leaf", 10,50,5)), "bootstrap": hp.choice("bootstrap", [False,True]), "n_estimators" : scope.int(hp.quniform("n_estimators", 50,200,25)), "criterion": hp.choice("criterion", ["gini","entropy"])} </pre>	<pre> {'bootstrap': True, 'criterion': 'gini, 'max_depth': 9.0, 'max_features': 5.0, 'min_samples_leaf': 10.0, 'min_samples_split': 40.0, 'n_estimators': 150.0} </pre>	71.51%	69.47%
3	<pre> {"max_depth": scope.int(hp.quniform('max_depth', 5, 15, 1)), "max_features": scope.int(hp.quniform('max_features', 3, 5, 1)), "min_samples_split": scope.int(hp.quniform("min_samples_split", 5, 40, 5)), "min_samples_leaf": scope.int(hp.quniform("min_samples_leaf", 5,60,5)), "bootstrap": hp.choice("bootstrap", [True]), "n_estimators" : scope.int(hp.quniform("n_estimators", 100,250,10)), </pre>	<pre> {'bootstrap': True, 'criterion': 'gini, 'max_depth': 10.0, 'max_features': 5.0, 'min_samples_leaf': 10.0, 'min_samples_split': 40.0, 'n_estimators': 210.0} </pre>	71.63%	68.97%

	"criterion": hp.choice("criterion", ["gini"])}
--	--

## Logistic Regression

Attempt	Parameters Tune	Best Parameter	CV Results (F1)	Test Results (F1)
0	NA	Base Parameters	68.92%	
1	{ "penalty": hp.choice("penalty", ['l1','l2']), "fit_intercept" : hp.choice("fit_intercept", [True,False]), "C": hp.choice("C", list(np.logspace(-3,5,500))), "max_iter":hp.choice("max_iter", [50,100,200,500]), }	{ 'C':22.94676367231936 , 'fit_intercept': True, 'max_iter': 500, 'penalty': 'l2'} 	69.09%	67.90%
2	{ "penalty": hp.choice("penalty", ['l1','l2']), "fit_intercept" : hp.choice("fit_intercept", [True,False]), "C": hp.choice("C", list(np.logspace(-1,5,500))), "max_iter":hp.choice("max_iter", [100,200,500,700,1000]), "n_jobs":hp.choice("n_jobs", [3]) }	{ 'C':24.03042894406969 , 'fit_intercept': True, 'max_iter': 100, 'penalty': 'l2'} 	69.09%	67.90%
3	"penalty": hp.choice("penalty", ['l1','l2']), "fit_intercept" : hp.choice("fit_intercept", [True,False]), "C": hp.choice("C", list(np.logspace(-1,3,500))), "max_iter":hp.choice("max_iter", [50,75,100,200,500,700,1000]), "n_jobs":hp.choice("n_jobs", [3])	{ 'C':32.286282025367306 , 'fit_intercept': True, 'max_iter': 700, 'penalty': 'l2'} 	69.09%	67.90%

## XGBoost

Attempt	Parameters Tune	Best Parameter	CV Results (F1)	Test Results (F1)
0	NA	Base Parameters		
1	<pre>{   "learning_rate": hp.choice("learning_rate", [0.05, 0.07, 0.1, 0.15, 0.2]),   "min_child_weight": scope.int(hp.quniform("min_child_weight", 10, 20, 1)),   "gamma": scope.int(hp.quniform("gamma", 0, 4, 1)),   "subsample": hp.uniform("subsample", 0.6, 1.0),   "colsample_bytree": hp.uniform("colsample_bytree", 0.4, 0.7),   "reg_alpha": hp.choice("reg_alpha", [0.1, 1, 100]),   "reg_lambda": hp.choice("reg_lambda", [1e-5, 1e-2, 0.1]),   "max_depth": scope.int(hp.quniform("max_depth", 7, 11, 1)),   "n_estimators": hp.choice("n_estimators", [50, 75, 100, 150, 200]) }</pre>	<pre>{'colsample_bytree': 0.6970729260476218,  'gamma': 3.0,  'learning_rate': 0.15,  'max_depth': 7.0,  'min_child_weight': 18.0,  'n_estimators': 100,  'reg_alpha': 1,  'reg_lambda': 1e-5,  'subsample': 0.7863244971678923}</pre>	71.87%	70.12%
2	<pre>{   "learning_rate": hp.choice("learning_rate", [0.1, 0.15, 0.2, 0.25, 0.3]),   "min_child_weight": scope.int(hp.quniform("min_child_weight", 18, 25, 1)),   "gamma": scope.int(hp.quniform("gamma", 3, 7, 1)),   "subsample": hp.uniform("subsample", 0.7, 1.0),   "colsample_bytree": hp.uniform("colsample_bytree", 0.6, 0.7),   "reg_alpha": hp.choice("reg_alpha", [0.1, 0.5, 1, 1.5]),   "reg_lambda": hp.choice("reg_lambda", [1e-10, 1e-7, 1e-5, 1e-2]),   "max_depth": scope.int(hp.quniform("max_depth", 3, 7, 1)),   "n_estimators": hp.choice("n_estimators", [50, 75, 100, 150, 200]) }</pre>	<pre>{'colsample_bytree': 0.6653997176343666,  'gamma': 5.0,  'learning_rate': 0.3,  'max_depth': 5.0,  'min_child_weight': 19.0,  'n_estimators': 50,  'reg_alpha': 1.5,  'reg_lambda': 1e-2,  'subsample': 0.7680129070567101}</pre>	71.98%	70.04%

3	<pre>{   "learning_rate": hp.choice("learning_rate",     [0.3, 0.35, 0.4, 0.45, 0.5]),   "min_child_weight": scope.int(hp.quniform("min_child_weight", 19,25,1)),   "gamma": scope.int(hp.quniform("gamma", 5,10,1)),   "subsample": hp.uniform("subsample", 0.7, 1.0),   "colsample_bytree": hp.uniform("colsample_bytree", 0.6, 0.8),   "reg_alpha": hp.choice("reg_alpha", [1.5, 2, 2.5, 3]),   "reg_lambda": hp.choice("reg_lambda", [1e-2, 0.1, 1]),   "max_depth": scope.int(hp.quniform("max_depth", 5, 10, 1)),   "n_estimators": hp.choice("n_estimators", [50, 75, 100, 150, 200]) }</pre>	<pre>{'colsample_bytree': 0.7465937617317032, 'gamma': 7.0, 'learning_rate': 0.4, 'max_depth': 6.0, 'min_child_weight': 24.0, 'n_estimators': 200, 'reg_alpha': 1.5, 'reg_lambda': 1, 'subsample': 0.7069236585610036}</pre>	71.99%	68.25%
---	--	--	--------	--------

SVM:

Attempt	Parameters Tune	Best Parameter	CV Result s (F1)	Test Results (F1)
0	NA	Base Estimator	69.15%	
1	<pre>{ "gamma" : hp.choice("gamma", [0.05,0.07,0.08,0.1,0.5,1]),   "C": hp.choice("C", [0.1, 1, 10,50,100]),   "kernel": hp.choice("kernel", ["linear", "rbf"])} </pre>	<pre>{'C': 100, 'gamma': 0.08, 'kernel': 'rbf'} </pre>	70.57%	67.42%
2	<pre>{"gamma" : hp.choice("gamma", [0.07,0.075,0.08,0.09]),   "C": hp.choice("C", [100,125,150,200]),   "kernel": hp.choice("kernel", ["linear", "rbf"])} </pre>	<pre>{'C': 125, 'gamma': 0.075, 'kernel': 'rbf'} </pre>	70.71%	67.18%
3	<pre>{ "gamma" : hp.choice("gamma", [0.07,0.072,0.075,0.078,0.08,0.85]),   "C": hp.choice("C", [10,30,50,120,125,150,300]),   "kernel": hp.choice("kernel", ["rbf"])} </pre>	<pre>{'C': 125, 'gamma': 0.075, 'kernel': 'rbf'} </pre>	70.71%	67.18%

MLP:

Attempt	Parameters Tune	Best Parameter	CV Results (F1)	Test Results (F1)
0	NA	Base Estimator	70.43%	
1	{ "hidden_layer_sizes": hp.choice("hidden_layer_sizes", [(50,50,50), (50,100,50), (100,)]), "activation" : hp.choice("activation", ['logistic', 'tanh', 'relu']), "solver": hp.choice("solver", ['lbfgs', 'sgd', 'adam']), "learning_rate": hp.choice("learning_rate", ['constant','adaptive']), "alpha": hp.choice('alpha', [0.0001, 0.001, 0.01, 0.05]), "batch_size": hp.choice("batch_size", [32,64,128,256,512])}	{ hidden_layer_size s:(50,100,50), 'activation': 'logistic', 'solver': 'lbfgs', 'learning_rate': 'con stant', 'alpha': 0.01, 'batch_size': 128}	72.02%	66.3%
2	{ "hidden_layer_sizes": hp.choice("hidden_layer_sizes", [(50,75,50), (50,100,50), (75,100,75),(100,)]), "activation" : hp.choice("activation", ['logistic', 'tanh', 'relu']), "solver": hp.choice("solver", ['lbfgs', 'sgd', 'adam']), "learning_rate": hp.choice("learning_rate", ['constant','adaptive']), "alpha": hp.choice('alpha', [0.001,0.005,0.01,0.05]), "batch_size": hp.choice("batch_size", [64,128,256]) }	{ hidden_layer_size s:(75,100,75), 'activation': 'tanh', 's olver': 'adam', 'learning_rate': 'con stant', 'alpha': 0.05, 'batch_size': 256}	71.63%	67.68%
3	{ "hidden_layer_sizes": hp.choice("hidden_layer_sizes", [(50,100,50), (75,100,75)]), "activation" : hp.choice("activation", ['logistic', 'tanh', 'relu']), "solver": hp.choice("solver", ['lbfgs', 'sgd', 'adam']), "learning_rate": hp.choice("learning_rate", ['constant','adaptive']), "alpha": hp.choice('alpha', [0.01,0.05,0.075,0.1]),	{ hidden_layer_size s:(50,100,50), 'activation': 'logistic', 'solver': 'lbfgs', 'learning_rate': 'con stant', 'alpha': 0.01, 'batch_size': 128}	72.02%	66.3%

	"batch_size": hp.choice("batch_size", [128,256,512])}			
--	---	--	--	--

Decision Tree:

Attempt	Parameters Tune	Best Parameter	CV Results (F1)	Test Results (F1)
0	NA	Base Parameters	66.60%	
1	{ "max_depth": scope.int(hp.quniform('max_depth', 3, 10, 1)), "max_features": scope.int(hp.quniform('max_features', 3, 5, 1)), "criterion": hp.choice("criterion", ["gini","entropy"]), "min_samples_split": scope.int(hp.quniform("min_samples_split", 5, 50, 10)), "min_samples_leaf": scope.int(hp.quniform("min_samples_leaf", 30,60,5)), "min_impurity_decrease": hp.choice("min_impurity_decrease",[0,1e-05,0.001,0.005,0.01])}	{ 'criterion': "entropy", 'max_depth': 7.0, 'max_features': 3.0, 'min_impurity_decrease': 0.001, 'min_samples_leaf': 30.0, 'min_samples_split': 10.0}	71.62%	68.44%
2	{ "max_depth": scope.int(hp.quniform('max_depth', 5, 15, 1)), "max_features": scope.int(hp.quniform('max_features', 3, 5, 1)), "criterion": hp.choice("criterion", ["gini","entropy"]), "min_samples_split": scope.int(hp.quniform("min_samples_split", 5, 50, 10)), "min_samples_leaf": scope.int(hp.quniform("min_samples_leaf", 5,50,5)), "min_impurity_decrease": hp.choice("min_impurity_decrease",[0,1e-05,0.001,0.005,0.01])}	{ 'criterion': "gini", 'max_depth': 5.0, 'max_features': 3.0, 'min_impurity_decrease': 1e-05, 'min_samples_leaf': 40.0, 'min_samples_split': 30.0}	71.70%	69.14%

3	<pre>{   "max_depth":     scope.int(hp.quniform('max_depth', 3, 15, 1)),   "max_features":     scope.int(hp.quniform('max_features', 3, 5, 1)),   "criterion": hp.choice("criterion",     ["gini", "entropy"]),   "min_samples_split":     scope.int(hp.quniform("min_samples_split", 5, 50, 10)),   "min_samples_leaf":     scope.int(hp.quniform("min_samples_leaf", 5, 50, 5)),   "min_impurity_decrease":     hp.choice("min_impurity_decrease", [0, 1e-07, 1e-05, 0.001, 0.005]) }</pre>	<pre>{   'criterion': "gini",   'max_depth': 7.0,   'max_features': 4.0,   'min_impurity_decrease': 1e-07,   'min_samples_leaf': 25.0,   'min_samples_split': 20.0 }</pre>	71.81%	68.31%
4	<pre>{   "max_depth":     scope.int(hp.quniform('max_depth', 3, 15, 1)),   "max_features":     scope.int(hp.quniform('max_features', 3, 5, 1)),   "criterion": hp.choice("criterion", ["gini"]),   "min_samples_split":     scope.int(hp.quniform("min_samples_split", 5, 80, 10)),   "min_samples_leaf":     scope.int(hp.quniform("min_samples_leaf", 5, 50, 5)),   "min_impurity_decrease":     hp.choice("min_impurity_decrease", [0, 1e-07, 1e-06, 1e-05, 1e-03, 0.001]) }</pre>	<pre>{   'criterion': "gini",   'max_depth': 11.0,   'max_features': 3.0,   'min_impurity_decrease': 1e-07,   'min_samples_leaf': 20.0,   'min_samples_split': 80.0 }</pre>	72.06%	67.98%
5	<pre>{   "max_depth":     scope.int(hp.quniform('max_depth', 3, 15, 1)),   "max_features":     scope.int(hp.quniform('max_features', 3, 5, 1)),   "criterion": hp.choice("criterion", ["gini"]),   "min_samples_split":     scope.int(hp.quniform("min_samples_split", 5, 100, 10)),   "min_samples_leaf":     scope.int(hp.quniform("min_samples_leaf", </pre>	<pre>{   'criterion': 'gini',   'max_depth': 10.0,   'max_features': 3.0,   'min_impurity_decrease': 1e-04,   'min_samples_leaf': 15.0,   'min_samples_split': 40.0 }</pre>	71.82%	67.47%



	5,50,5)), "min_impurity_decrease": hp.choice("min_impurity_decrease",[0,1e-08,1e-07,1e-06,1e-05,1e-04,1e-03])}			
--	--	--	--	--

### BaggingClassifier

Attempt	Parameters Tune	Best Parameter	CV Results (F1)	Test Results (F1)
0	NA	Base Parameters	67.78%	
1	bagging_param_space1 = {"bootstrap": hp.choice("bootstrap", [False,True]), "n_estimators" : hp.choice("n_estimators", [10,15,20,25,30])}	{ 'bootstrap': False, 'n_estimators': 10}	71.22%	69.47%
2	bagging_param_space1 = {"bootstrap": hp.choice("bootstrap", [False,True]), "n_estimators" : hp.choice("n_estimators", [3,4,5,10,15,20,50,100])}	{ 'bootstrap': False, 'n_estimators': 5}	71.24%	69.47%

### Lightgbm

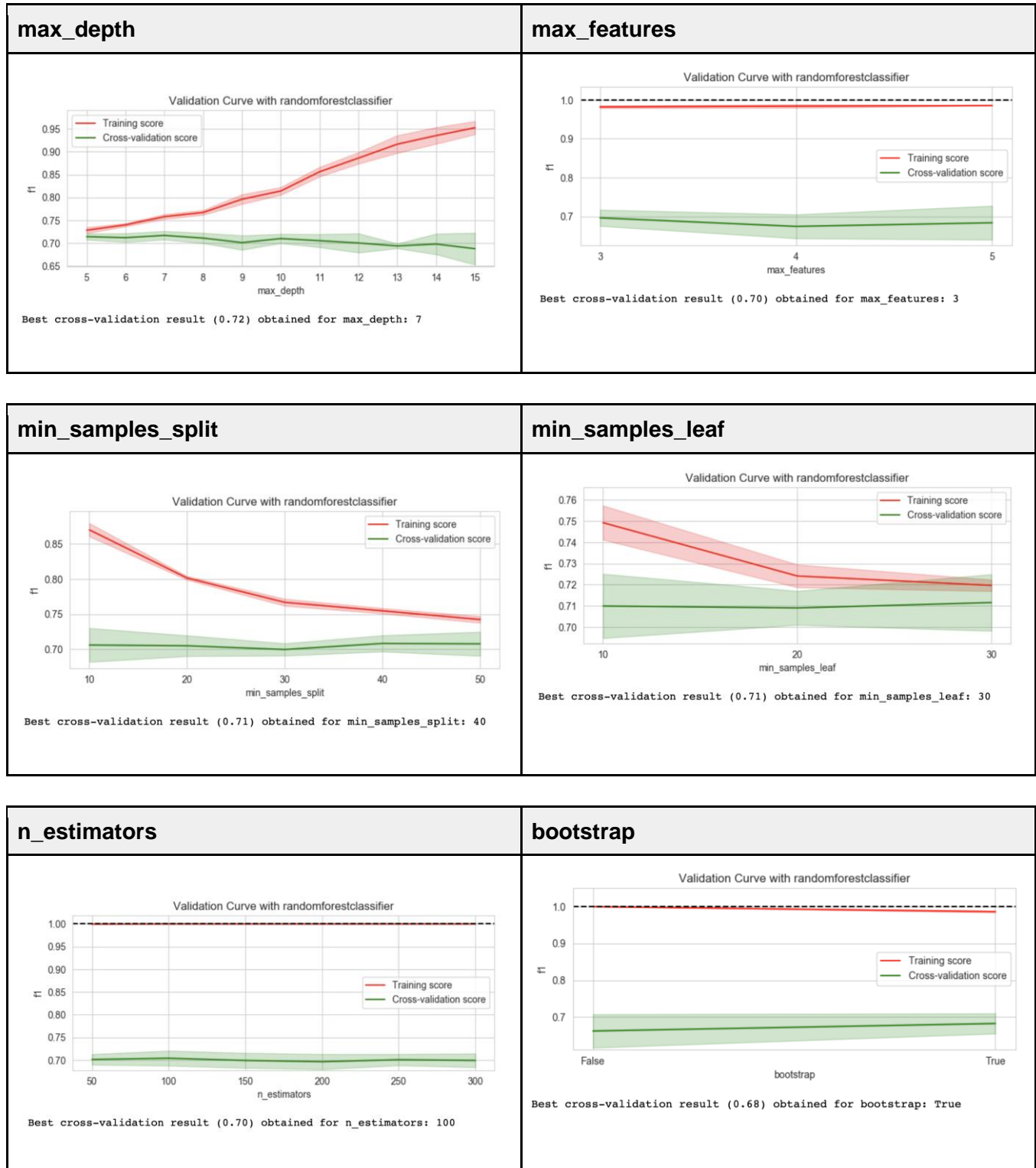
Attempt	Parameters Tune	Best Parameter	CV Results (F1)	Test Results (F1)
0	NA	Base Parameters		
1	{"boosting_type": hp.choice("boosting_type",["gbdt","dart","goss"]), 'num_leaves': scope.int(hp.quniform('num_leaves', 30, 50, 2)), "learning_rate":hp.choice("learning_rate", [0.01,0.05,0.1,0.2,0.3]), "gamma":scope.int(hp.quniform('gamma', 1, 5, 1)), 'colsample_bytree':hp.uniform('colsample_by_tree', 0.5, 1.0),	{'boosting_type': 'gbdt', 'colsample_by_tree': 0.8309554549705769, 'gamma': 5.0, 'learning_rate': 0.05, 'max_depth': 6.0, 'min_child_samples': 25.0, 'min_child_weight': 6.0, 'min_split_gain': 0.5,	72.10%	70.04%

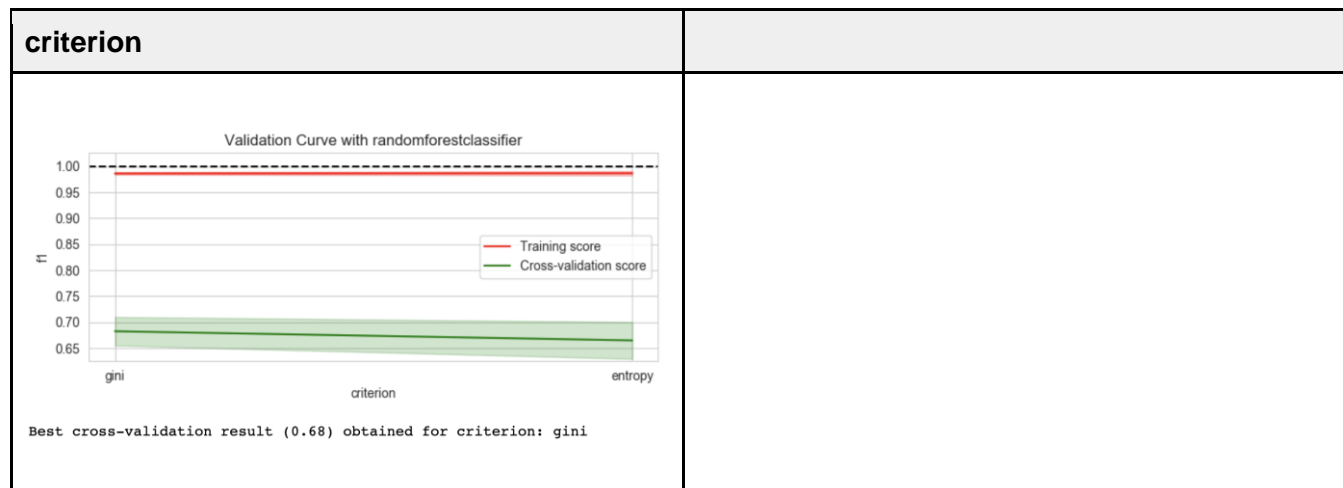
	<pre>'reg_alpha': hp.choice('reg_alpha', [1e-2, 0.1, 1, 100,200,500]), 'reg_lambda':hp.choice('reg_lambda', [1e-2, 0.1, 1, 100,200,500]), "max_depth": scope.int(hp.quniform('max_depth', 5, 10, 1)), 'min_child_weight': scope.int(hp.quniform('min_child_weight', 1, 7, 1)), "n_estimators" : hp.choice("n_estimators", [5,10,20,50,100]), "min_split_gain": hp.choice('min_split_gain',[0.01,0.05,0.1,0.5,1]), "min_child_samples":scope.int(hp.quniform('min_child_samples', 5, 30, 5)),}</pre>	<pre>'n_estimators': 20, 'num_leaves': 42.0, 'reg_alpha': 1, 'reg_lambda': 1e-02}</pre>		
2	<pre>{"boosting_type": hp.choice("boosting_type",["gbdt","dart","goss"]), 'num_leaves': scope.int(hp.quniform('num_leaves', 30, 50, 2)), "learning_rate":hp.choice("learning_rate", [0.01,0.025,0.05,0.075,0.1,0.2]), "gamma":scope.int(hp.quniform('gamma', 1, 5, 1)), 'colsample_bytree': hp.uniform('colsample_by_tree', 0.5, 1.0), 'reg_alpha': hp.choice('reg_alpha', [1e-05,1e-03,1e-2, 0.1, 1, 10]), 'reg_lambda':hp.choice('reg_lambda', [1e-05,1e-03,1e-2, 0.1, 1, 10]), "max_depth": scope.int(hp.quniform('max_depth', 5, 10, 1)), 'min_child_weight': scope.int(hp.quniform('min_child_weight', 1, 7, 1)), "n_estimators" : hp.choice("n_estimators", [10,20,40,50,100]), "min_split_gain": hp.choice('min_split_gain',[0.01,0.025,0.05,0.075,0.1,0.5]), "min_child_samples":scope.int(hp.quniform('min_child_samples', 10, 30, 5)),}</pre>	<pre>{'boosting_type': 'gbdt, 'colsample_by_tree': 0.805684643533219 1, 'gamma': 3.0, 'learning_rate': 0.075, 'max_depth': 6.0, 'min_child_samples': 30.0, 'min_child_weight': 7.0, 'min_split_gain': 0.05, 'n_estimators': 20, 'num_leaves': 32.0, 'reg_alpha': 1, 'reg_lambda': 1e-05}</pre>	71.97%	69.64%
3	<pre>lgb_param_space3 = {"boosting_type": hp.choice("boosting_type",["gbdt","dart","goss"]), 'num_leaves': scope.int(hp.quniform('num_leaves', 30, 50,</pre>	<pre>{'boosting_type': 'dart, 'colsample_by_tree': 0.859172501439123 9</pre>	72.03%	68.97%

<pre> 2)), "learning_rate":hp.choice("learning_rate", [0.01,0.025,0.05,0.075,0.1,0.2]), "gamma":scope.int(hp.quniform('gamma', 1, 5, 1)), 'colsample_bytree': hp.uniform('colsample_by_tree', 0.5, 1.0), 'reg_alpha': hp.choice('reg_alpha', [1e-2, 0.1, 1, 10]), 'reg_lambda':hp.choice('reg_lambda', [0,1e- 05,1e-03,1e-2, 0.1, 1]), "max_depth": scope.int(hp.quniform('max_depth', 5, 10, 1)), 'min_child_weight': scope.int(hp.quniform('min_child_weight', 1, 7, 1)), "n_estimators" : hp.choice("n_estimators", [15,20,25,30,40]), "min_split_gain": hp.choice('min_split_gain',[0.01,0.025,0.05,0. 075,0.1,0.5]), "min_child_samples":scope.int(hp.quniform(' min_child_samples', 10, 30, 5)),}</pre>	<pre> , 'gamma': 2.0, 'learning_rate': 0.075, 'max_depth': 6.0, 'min_child_samples': 30.0, 'min_child_weight': 2.0, 'min_split_gain': 0.5, 'n_estimators': 20, 'num_leaves': 30, 'reg_alpha': 1e-2, 'reg_lambda': 1}</pre>		
---	--	--	--

(b) (3 points) Also discuss any insights you may uncovered while tuning the models (what parameters made the most difference for which models, etc)

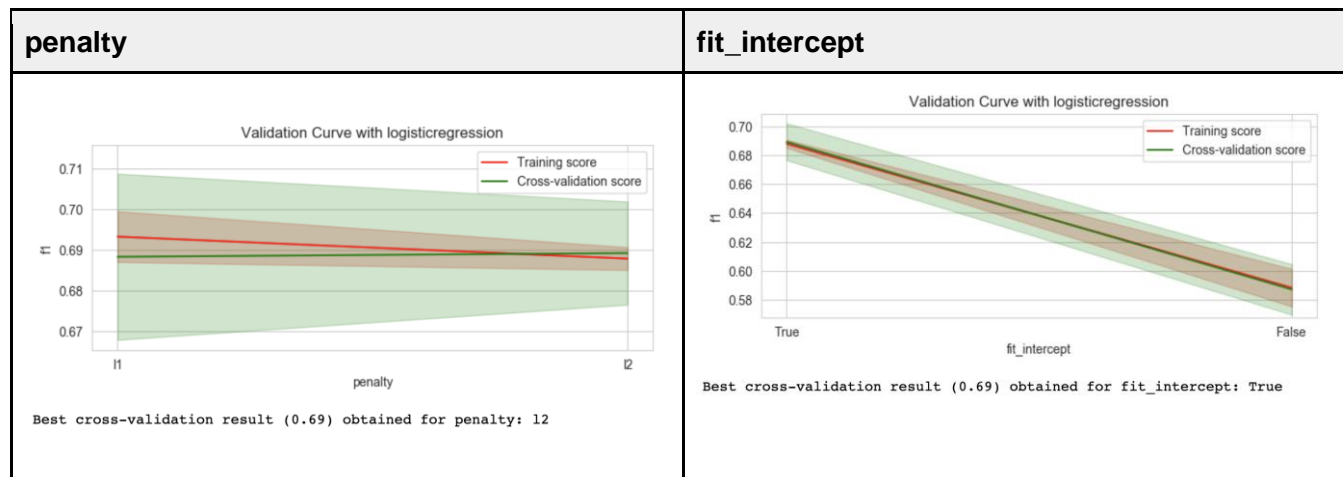
### Random Forest Hyperparameters:

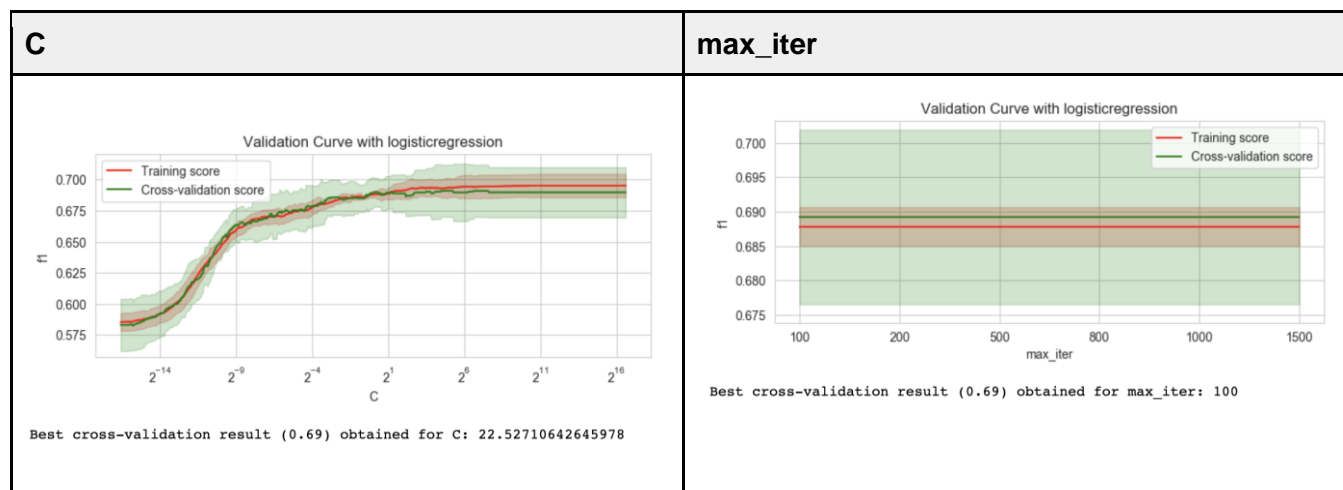




For Random Forest, max\_depth,max\_features, min\_sample\_split and min\_sample\_leaf seems to have the most impact on the model as tuning it allows the model to achieve a higher validation score. Among these 4 features, max\_depth made the most difference as it has the highest validation curve score of 0.72.

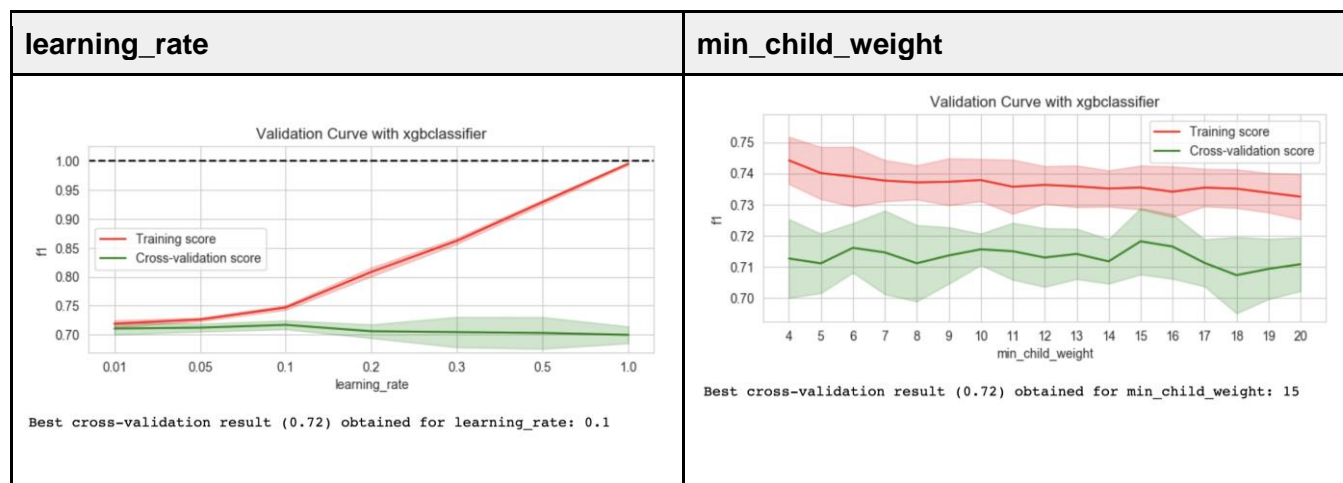
### Logistic Regression Hyperparameters:

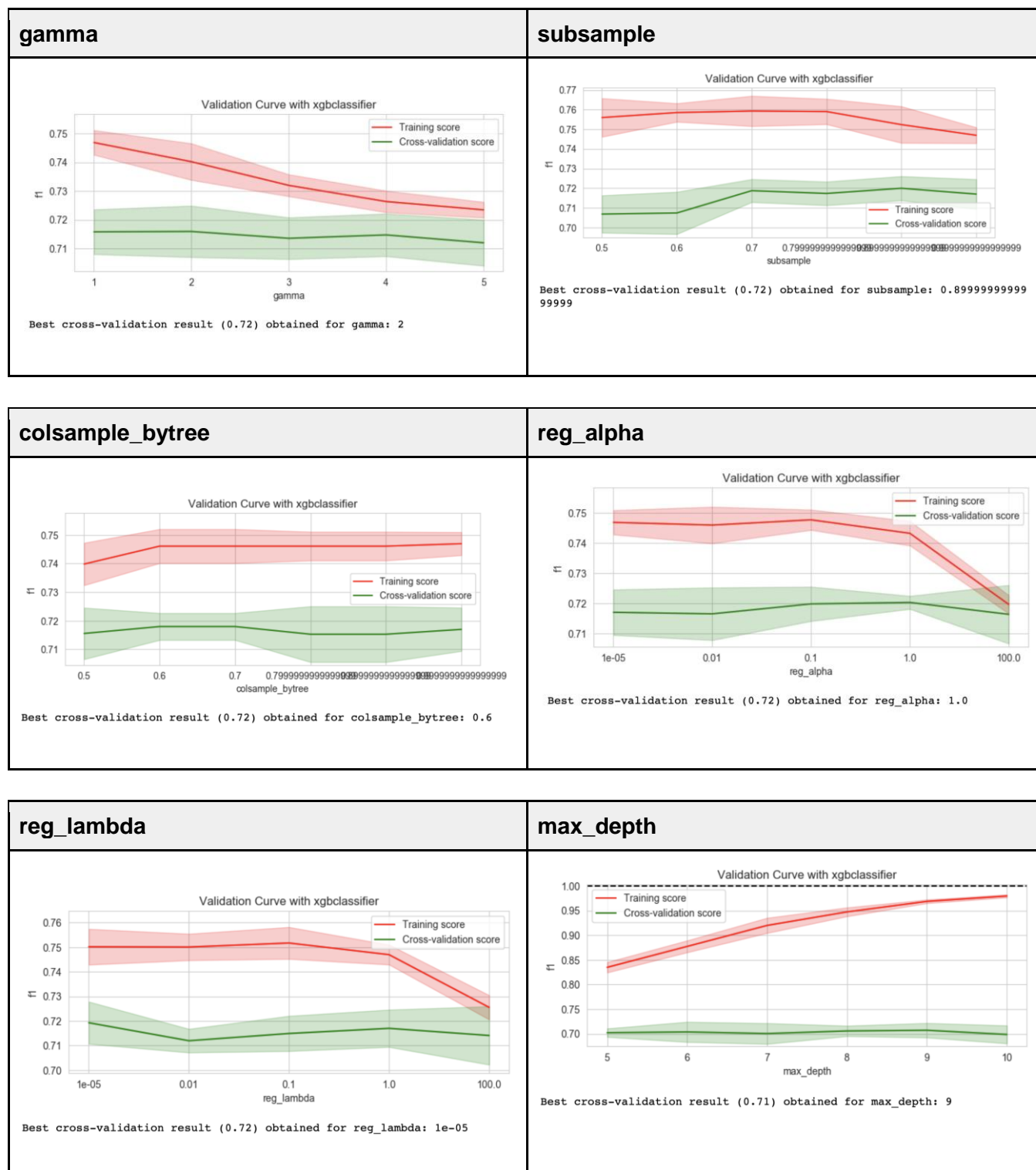


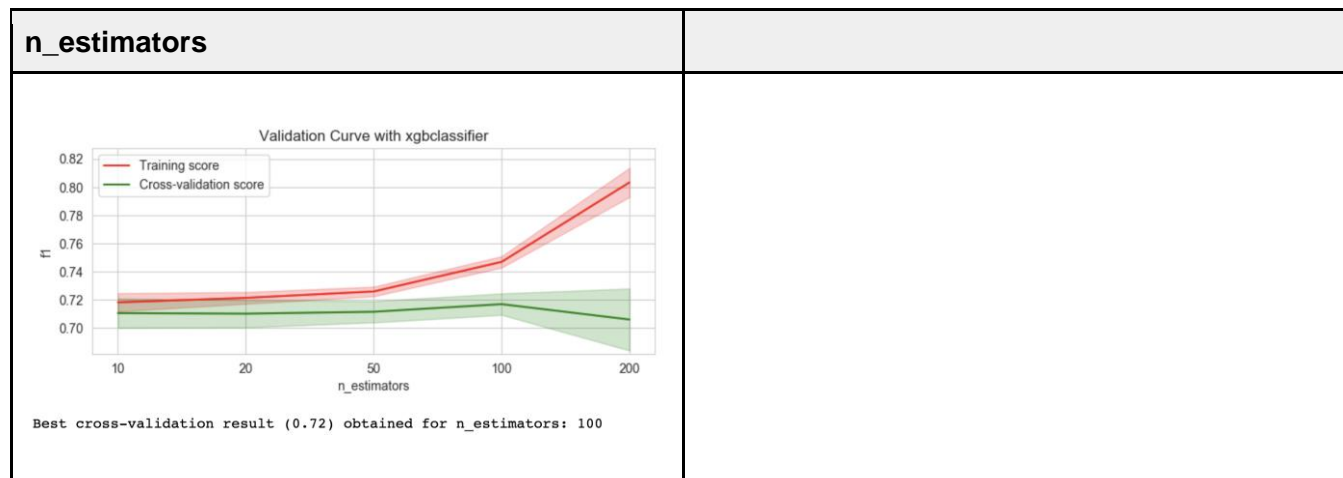


For Logistic Regression, the `fit_intercept` parameter seems to have the most impact. There is a sharp drop in validation score when `fit_intercept=False`. The `C` value which is the inverse of the regularization strength also has a huge impact on the performance of the model. A higher `C` results in a higher validation score. The other parameters seems to give similar results across different values.

### XGBoost Hyperparameters:

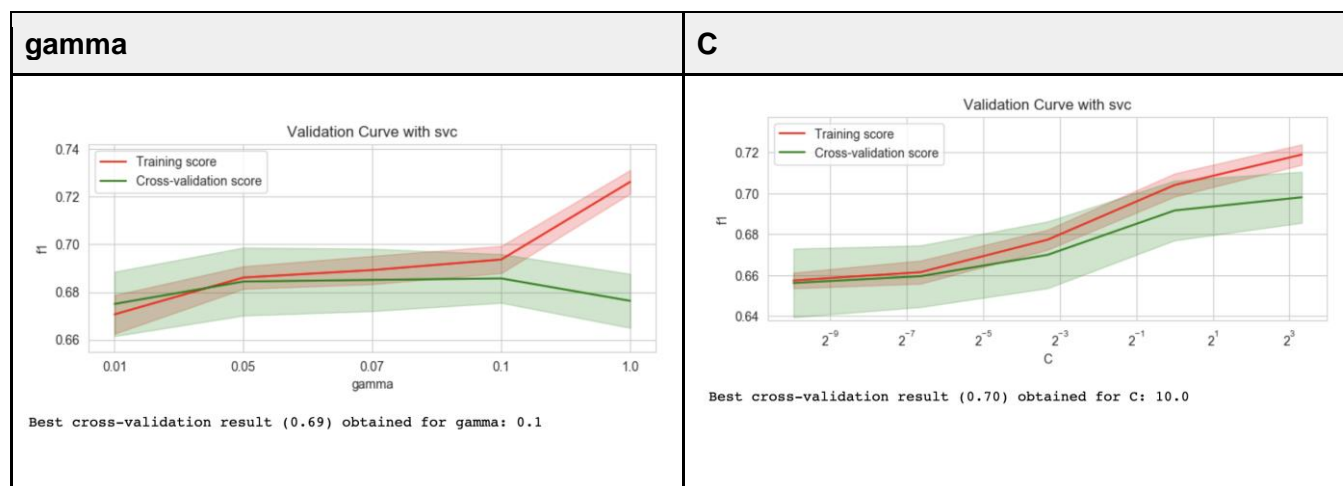




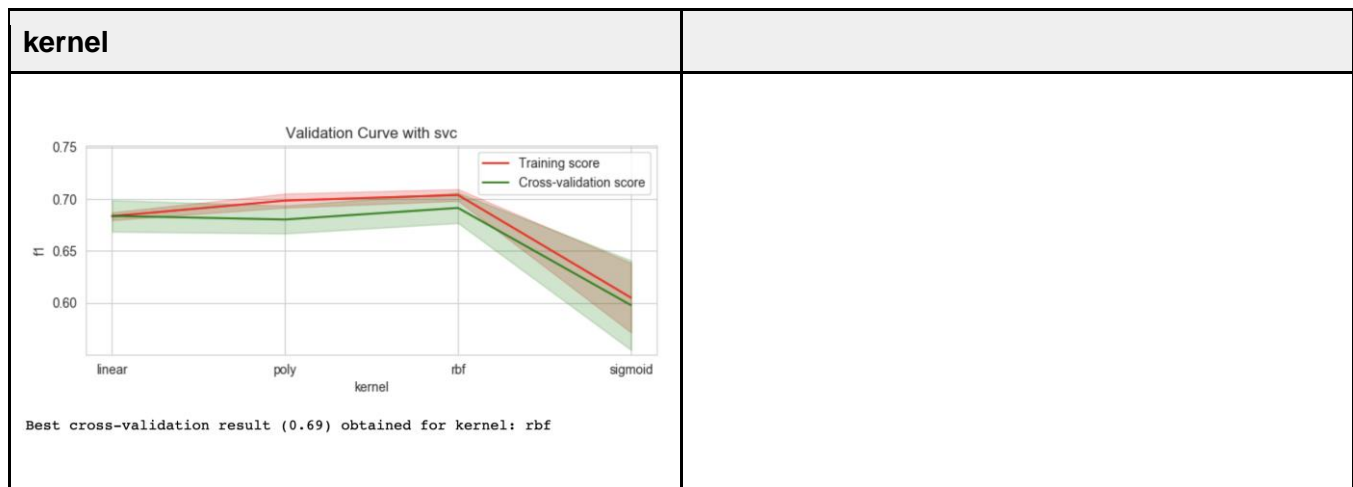


For XGBoost, the parameter min\_child\_weight, subsample, n\_estimators, reg\_alpha seems to make the most difference to the model as the plot of its validation curve shows the most variant at different values. However, tuning most of the features allow the model to reach a validation score of 0.72 which is the highest. Therefore, we should consider all these parameters when tuning our model.

### SVM Hyperparameters:

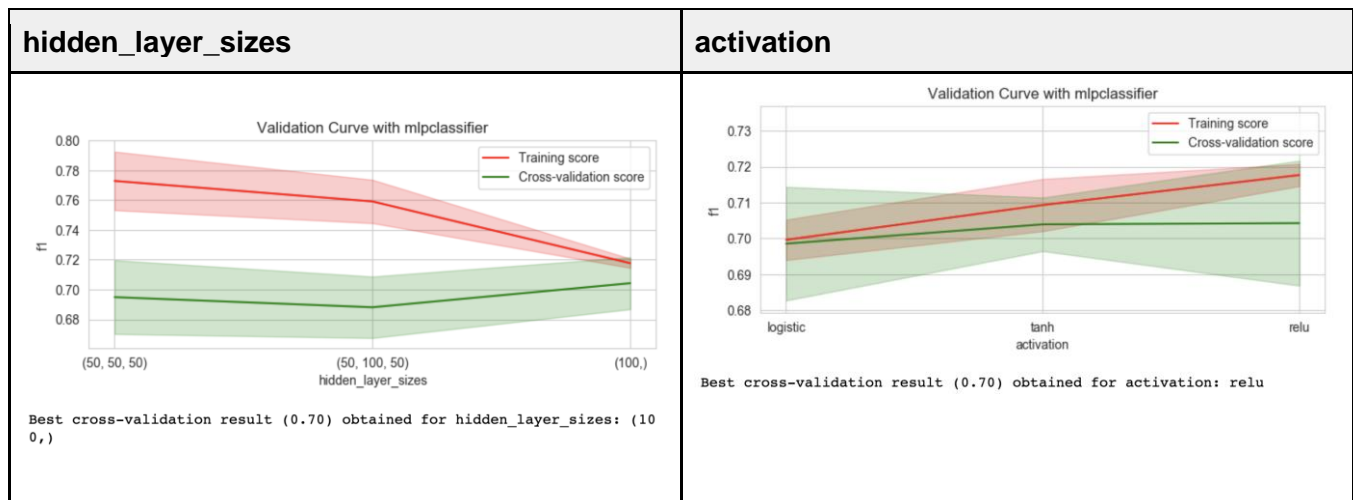


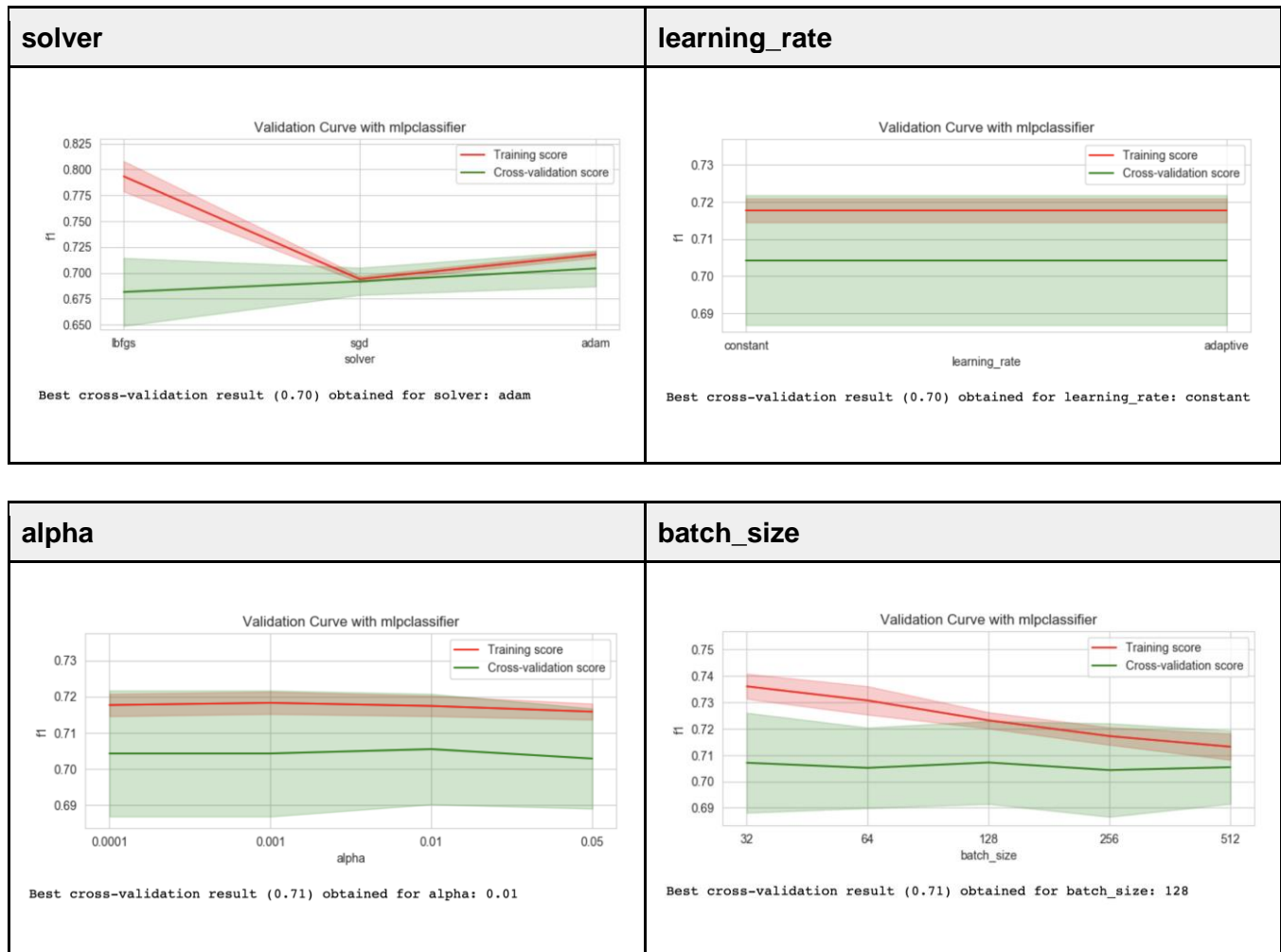




For SVM, all three parameters seem to be important as different value results in very different result as seen in the value of C which is the penalty parameter of C, a higher C results in a higher cross validation score. The kernel used also has a significant impact on the validation score. The sigmoid kernel seems to perform particularly worse than the other kernel. All 3 parameters should be tuned vigorously to achieve the best result.

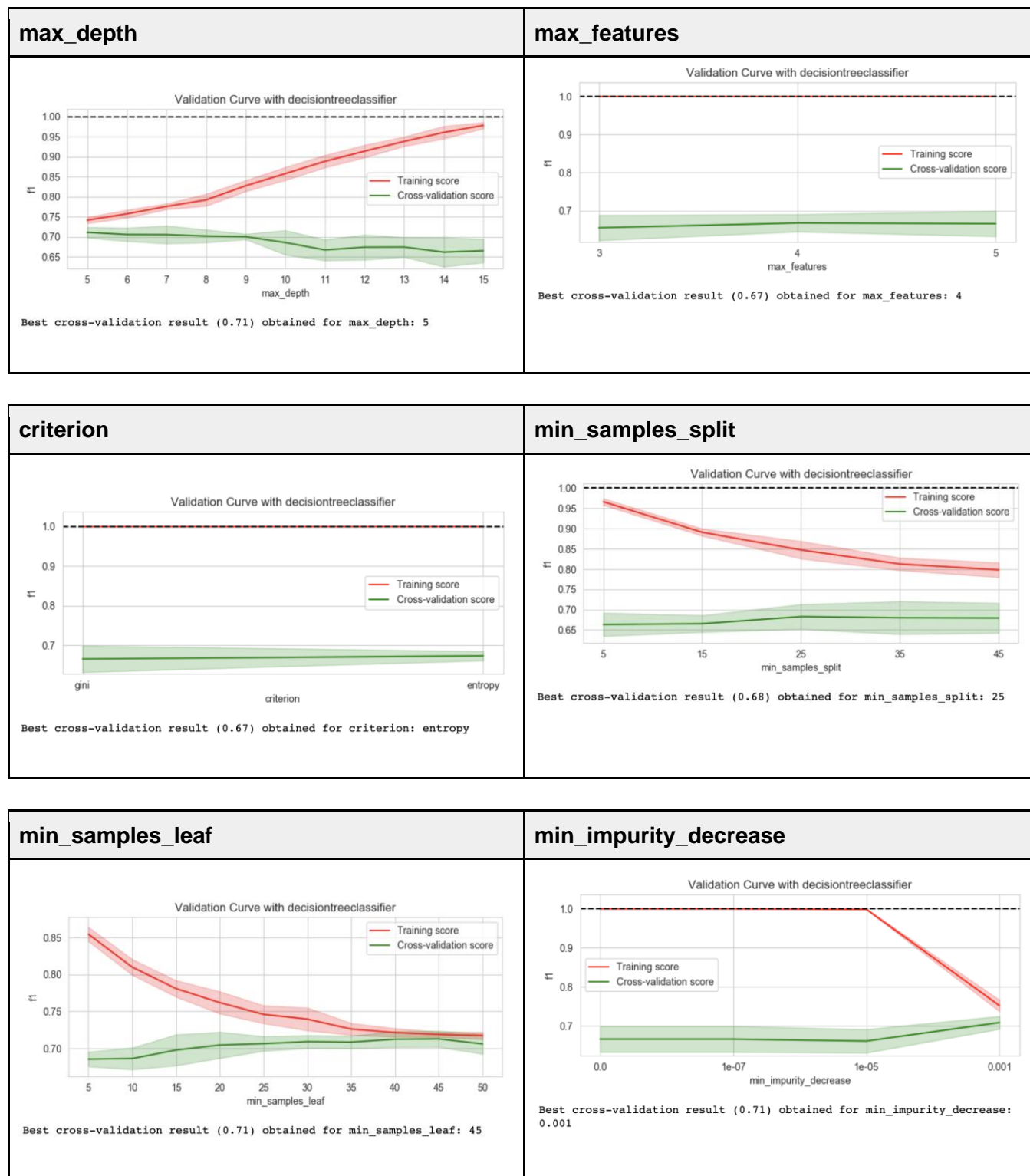
### MLP Hyperparameters:





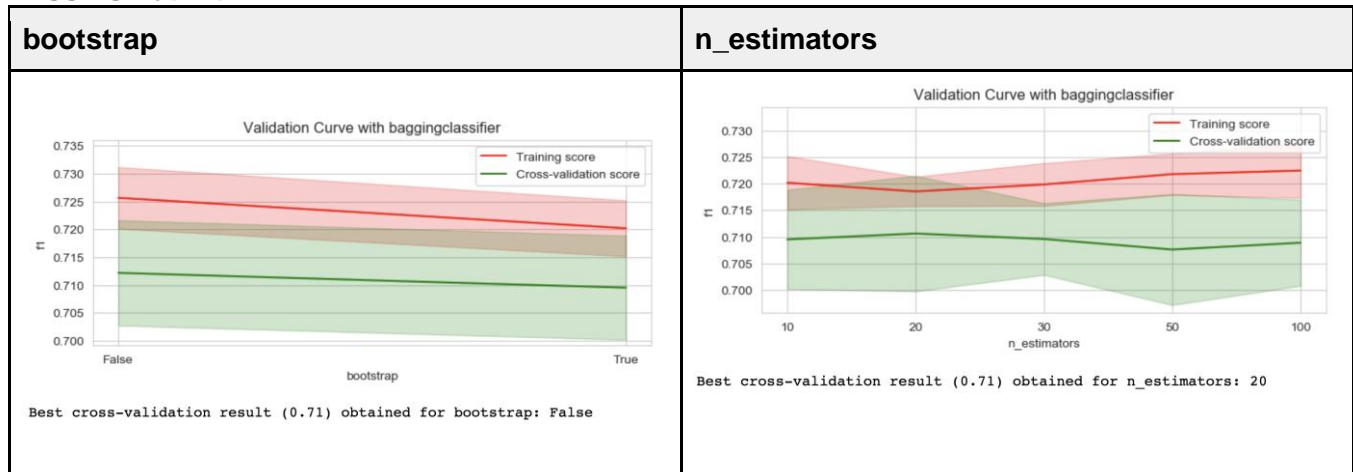
For MLP, the hidden\_layer\_size, activation, solver and batch\_size seems to have the most impact on the model as there is a huge variation in the validation curve across different value. In fact, all this features in MLP are equally important, as MLP is actually a Neural Network Model and each parameters are interlinked, there are more permutations to tune. Especially for hidden\_layer\_size, we can have different layers with different number of neurons that can have very different results.

## Decision Tree Hyperparameters:



For decision tree, max\_depth, min\_samples\_leaf, min\_impurity\_decrease, min\_samples\_split seems to have the most impact on the model. In fact, these features are for pruning the decision tree to prevent overfitting, therefore they are important in making the model generalizable to new unseen data.

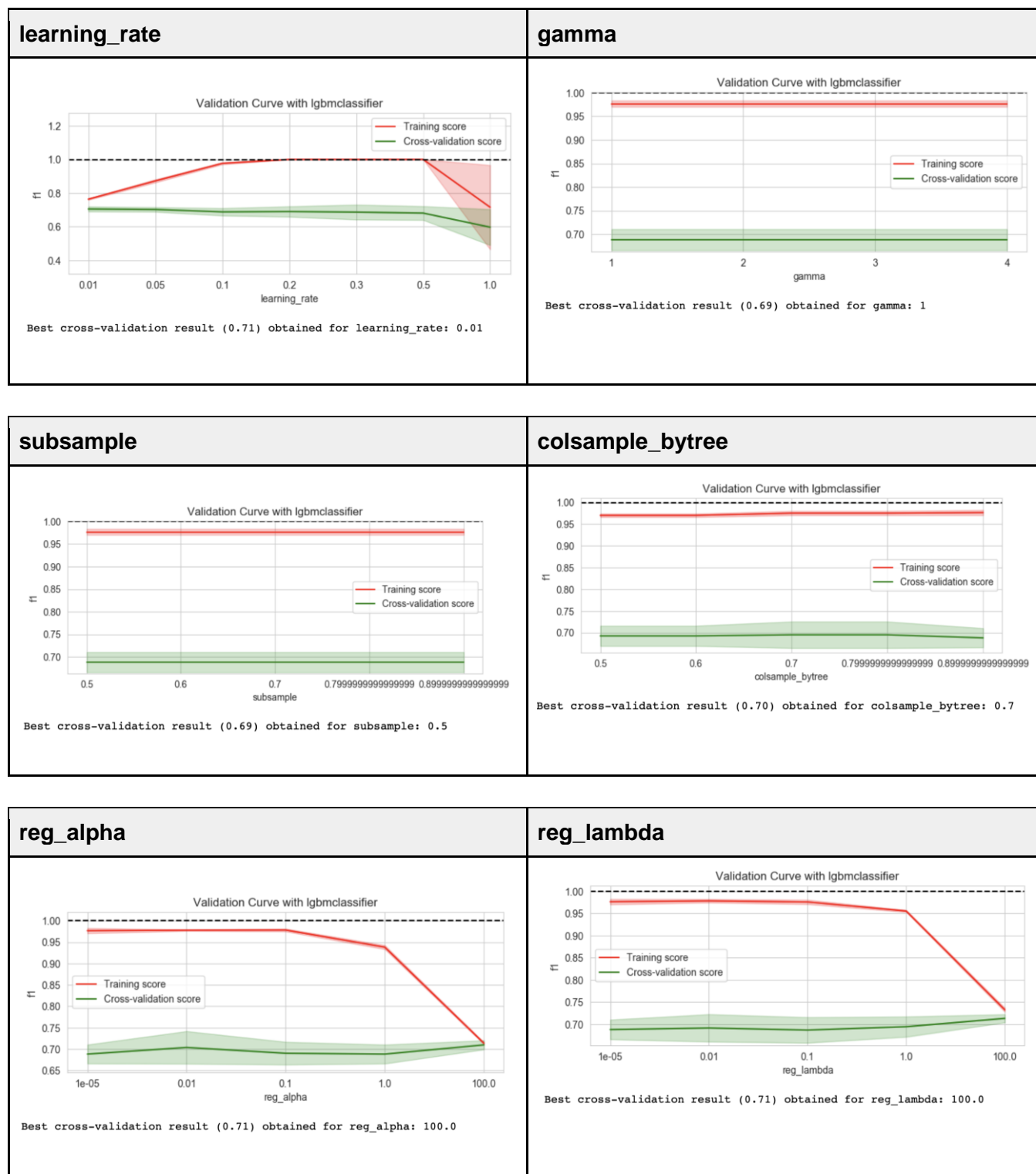
### Bagging Hyperparameters:

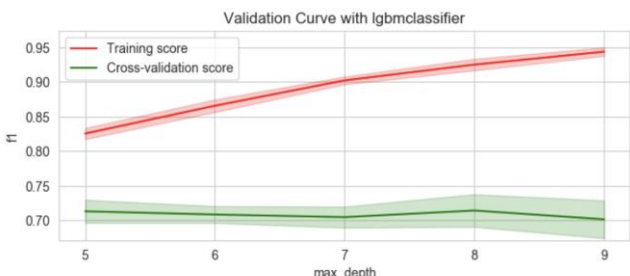
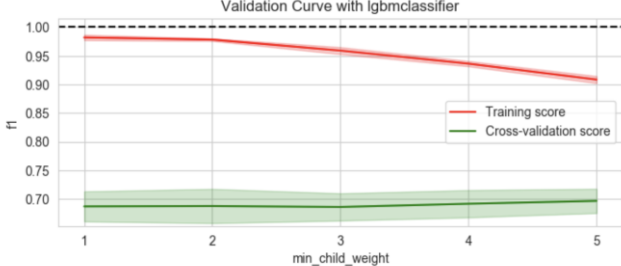
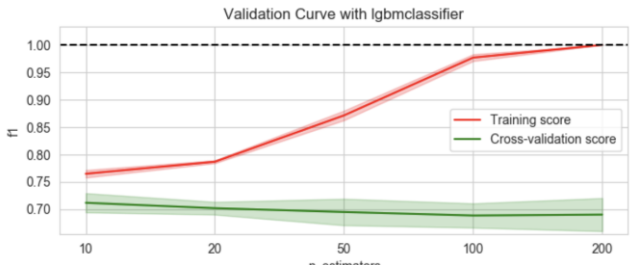
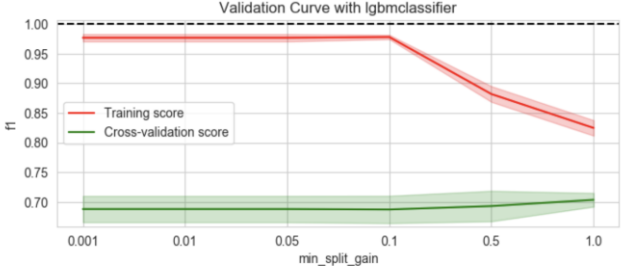
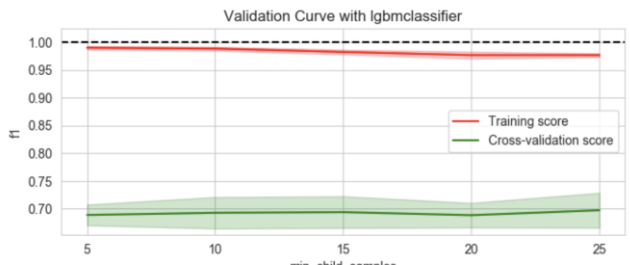


The Bagging Classifier is actually using the previous decision tree model as its base estimator. Therefore, the base estimator should have the most impact on the bagging model. However, the bootstrap and n\_estimators parameters also seem to help to improve the model.

### LightGBM Hyperparameters:





max_depth	min_child_weight
 <p>Best cross-validation result (0.71) obtained for max_depth: 8</p>	 <p>Best cross-validation result (0.70) obtained for min_child_weight: 5</p>
n_estimators	min_split_gain
 <p>Best cross-validation result (0.71) obtained for n_estimators: 10</p>	 <p>Best cross-validation result (0.70) obtained for min_split_gain: 1.0</p>
min_child_samples	
 <p>Best cross-validation result (0.70) obtained for min_child_samples: 25</p>	

Done by: Lim Jin Ming, Jeremy Denzel (A0172720M), Sung Zheng Jie (A0168188M)

For LGM, learning\_rate, reg\_alpha, reg\_lambda and max\_depth seems to make the most difference. As tuning this parameters result in the highest cross validation result. In addition, these features also show the most variation when tuned. Therefore, they should have the most impact on the lgb model.

Nonetheless, for all models all their respective parameters are tuned together to get the most optimal model.