Capstone Project : NBFC (Foreclosure)

*Notes- **II***

Vikram Radhakrishnan

3rd January 2021

1). Model building and interpretation.

a. Build various models (You can choose to build models for either or all of descriptive, predictive or prescriptive purposes)

Descriptive analytics looks at data statistically to tell you what happened in the past. Descriptive analytics helps a business understand how it is performing by providing context to help stakeholders interpret information. This can be in the form of data visualizations like graphs, charts, reports, and dashboards.

Many descriptive models were built in Notes-1 showing past trends based on data and trying to interpret if there are any patterns that could be built upon particular features.

Predictive Analytics takes historical data and feeds it into a machine learning model that considers key trends and patterns. The model is then applied to current data to predict what will happen next. The machine learning models that we will be applying here largely cover this area of analytics on predicting future trends based on past data.

Prescriptive analytics takes predictive data to the next level. Now that you have an idea of what will likely happen in the future, what should you do? It suggests various courses of action and outlines what the potential implications would be for each. We will explore this too after building machine learning models.

CURRENT APPROACH: CLASSIFICATION BASED ON SUPERVISED LEARNING

For the current data-set we are dealing with, **Supervised Learning** algorithms would be appropriate. Here we model dependencies and relationships between a target prediction output and input features.

This problem also requires a **Classification** approach to solving. In classification, we create predictive models from training data which have features and class labels. These predictive models in-turn use the features learnt from training data on new, previously unseen data to predict their class labels. The output classes are discrete (not continuous).

In this exercise, we had earlier explored in NOTES-1, that many features can be eliminated through feature selection. We had used SCIKIT LEARN's **Feature Selection** to come up with the most important features required to model this problem.

Before this, we could also remove the highly correlated values by observing them through a **correlation heatmap**. This would reduce the number of features initially before ranking them through feature selection. These ideas were discussed and elaborated in Notes-1.

Alternatively, other methods such as PCA or Chi-square tests can be used to do feature selection and removal. Since any method can be used for this exercise, we do not discuss PCA or chi-square methods in this document further.

We further classified the features into **dependent** and **independent** variables ('X' and 'y' respectively).

We then perform a '**TEST-TRAIN-SPLIT'** dividing the rows or data into two segments on whether they will be used for training the model or for testing the 'trained' model.

For this problem, we use **80%** of the data for training and **20%** for testing. Given that the number of rows in our data is approximately 20,000 , this would mean that 16,000 rows are used for training and 4,000 records are used for testing.

For many of these models **Cross Validations** are performed. This means that an iterative process is performed to vary the test and train data within the dataset to get a more accurate model.

Other methods such as Tuning of parameters (e.g. through Grid Search), Boosting and Bagging techniques are also employed to improve model performance.

The following basic classification models are employed in this study:

1. **Decision Trees (CART)**
2. **Logistic Regression (Logit)**
3. **Linear Discriminant Analysis (LDA)**
4. **Support Vector Machine (SVM)**
5. **Artificial Neural Networks (ANN)**
6. **Gauss Naïve Bayes (NB)**
7. **K Nearest Neigbour (KNN)**

Other advanced models are used to further improve the performance and accuracy – these are :

8. **Random Forest with tuning (RF)**
9. **Bagging (with RF)**
10. **Boosting (e.g. XGBoost)**

These models are used along with this particular data to determine which of them is most appropriate to determine default using performance metrics such as **Accuracy, Support, F1 Score, Recall and Precision**.

Graphical means of comparing the test to train performances and understanding the applicability of the models can be achieved through tools such as the **ROC curve**.

b. Test your predictive model against the test set using various appropriate performance metrics

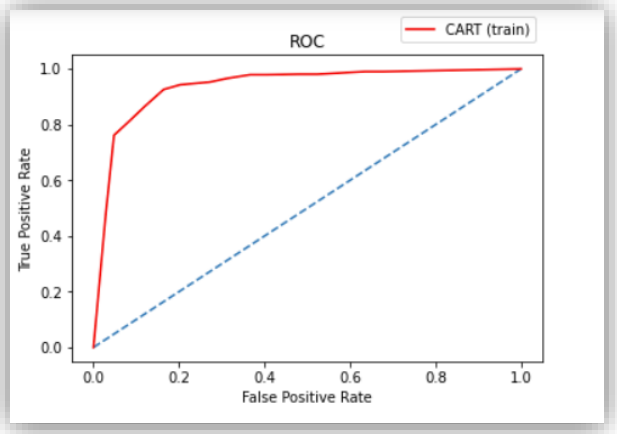
One of the first metrics we look at to understand how well our model behaves is the ROC curve. An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

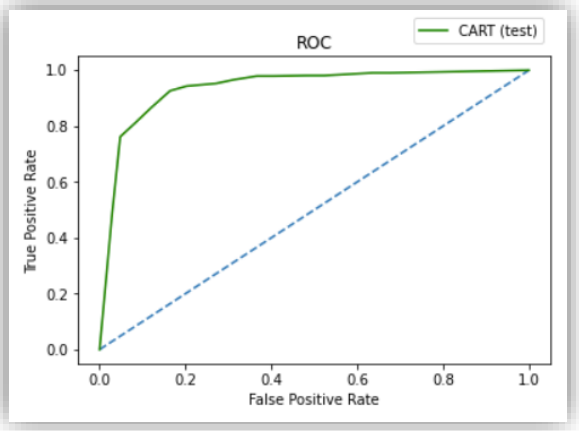
- True Positive Rate
- False Positive Rate

An ROC curve plots TPR vs. FPR at different classification thresholds.

Another metric associated with the ROC curve is the AUC. **AUC** stands for "Area under the ROC Curve." AUC measures the entire two-dimensional area underneath the entire ROC curve. AUC provides an aggregate measure of performance across all possible classification thresholds. AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

Let us take the example of the CART model – here we plot the TRAIN ROC vs TEST ROC.





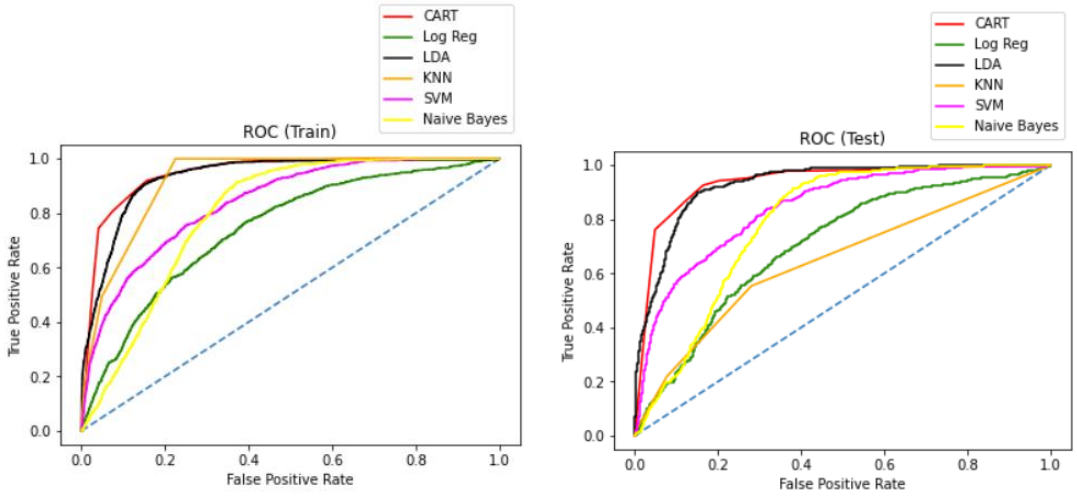
For the two curves plotted, the AUC scores are more or less the same, as they have approximately the same area under the curve.

	CART Train	CART Test
AUC	0.94	0.94

Similarly, we can plot ROC curves and get AUC values for other models individually as well.

For the purpose of this report, we do not plot these individually, but represent some of the relatively classic/simpler models in a single graph for Train and Test data.

In the following figure, we have plotted ROC curves in a single graph (each) to compare the model performances in either train or test data.



In the above example, we see that though some curves look similar in terms of shape (such as CART), others have quite different shapes between the Train and Test values such as K Nearest Neighbours (KNN).

	CART Train	CART Test	Logit Train	Logit Reg Test	LDA Train	LDA Test	KNN Train	KNN Test	SVM Train	SVM Test	NB Train	NB Test
AUC	0.94	0.94	0.91	0.91	0.92	0.92	0.92	0.90	0.91	0.91	0.80	0.79

For the models shown above, the AUC scores for test and train appear to be more or less very close to each other.

Similar for other advanced models such as Neural networks, Ensemble methods (RF) and models with tuning, boosting or bagging, we can plot the AUC scores. It is observed that some models 'OVERFIT' but they can be processed to prevent this from happening. Some ways to prevent overfitting are to work on feature selection, hyper-paramater tuning, cross-validation and other methods.

	CART Train	CART Test	Logit Train	Logit Reg Test	LDA Train	LDA Test	KNN Train	KNN Test	SVM Train	SVM Test	NB Train	NB Test	Neural Net Train	Neural Net Test
AUC	0.98	0.98	0.91	0.91	0.91	0.92	0.92	0.90	0.91	0.91	0.81	0.81	0.82	0.82
Accuracy	1.00	0.97	0.73	0.70	0.93	0.93	0.92	0.66	0.85	0.84	0.81	0.81	0.65	0.64
Recall	0.89	0.85	0.01	0.00	0.48	0.49	0.01	0.00	0.00	0.00	0.42	0.43	0.42	0.42
Precision	0.93	0.95	0.43	0.25	0.53	0.53	0.43	0.25	0.00	0.00	0.21	0.22	0.24	0.23
F1 Score	0.91	0.89	0.02	0.01	0.50	0.51	0.02	0.01	0.00	0.00	0.28	0.29	0.30	0.29

We observe that other than CART, the rest of the models do not perform too well in terms of other metrics such as recall and precision, though AUC and Accuracy may be somewhat acceptable. These models can be further bettered. However, we do not spend too much time doing this, as we have been able to get good results using other methods such as CART (and futher through RF and XGBoost which are discussed later).

Therefore we should proceed with Model Tuning.

c. Interpretation of the model(s)

In order to choose the right model, we need to check whether :

a) The accuracy score for test and train data are good enough (without overfitting scenario)
b) The other measures such as support, F1 Score, Precision and Recall are reasonable.

Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive. In our model, its important to have high precision.

Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive). With classification problems such as this, where False Negatives are a lot more expensive than False Positives, **we may want to have a model with a high recall rather than high precision.** In our model, we see poor recall scores for most of the models carried out except for Decision Trees (CART). This seems to be a good model for our problem. We can improve this model further by Tuning.

F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives). This seems to be healthy for the CART model.

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. In our case, CART seems to be the best model again here, since it shows a healthy support Score.

2. Model Tuning and business implication

a. Ensemble modelling, wherever applicable

We mentioned earlier in this document that Model Tuning can help make our models better in terms of metrics. One of the methods to do this is to employ Ensemble techniques.

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. Some examples of Ensemble methods are :

a) Random Forest (from extending CART or Decision Trees)
b) Bagging (**B**ootstrap **Agg**regat**ing**)
c) Bagging

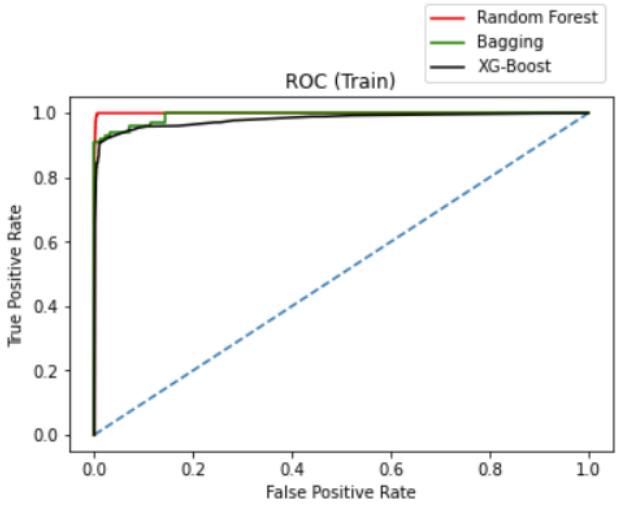
We have applied all the above three ensemble methods as individual models and evaluated their performance through metrics.

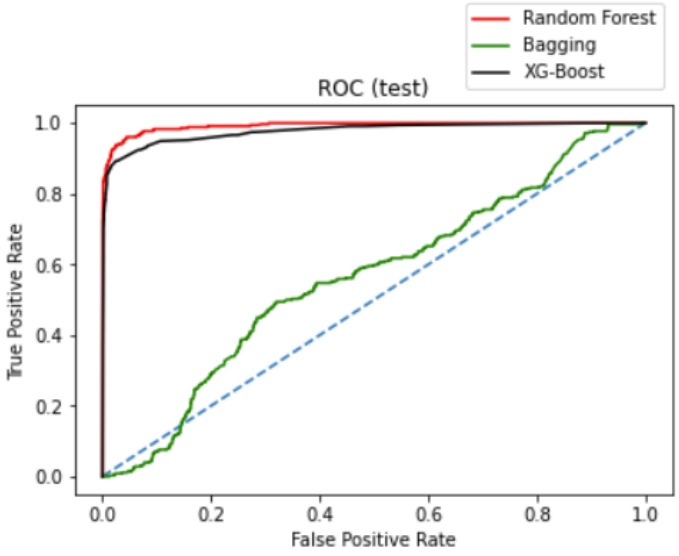
	Rand For (Tuned) Train	Rand For (Tuned) Test	Bagging(RF) Train	Bagging(RF) Test	XGBoost Train	XGBoost Test
AUC	0.99	0.98	0.94	0.09	0.98	0.98
Accuracy	1.00	0.99	0.99	0.56	0.98	0.98
Recall	0.93	0.84	0.96	1.00	0.84	0.79
Precision	0.99	0.97	0.92	0.09	0.94	0.93
F1 Score	0.96	0.90	0.94	0.16	0.89	0.86

We observe that through these methods, a large change in metrics is observed when compared to the models applied in Part 1 of this document.

For our problem, Random Forest and Boosting (Extreme Gradient Boost) seem to work out the best.

Bagging seems to have issues in the test data set. This can be fixed by modifying the hyperparamters of the model. However, for this exercise, we do not tune this further in the interest of time.





From the ROC curves, we observe that we have good performance for RF and XGB, but poor test performance for Bagging. Therefore, we discard bagging for now (it is possible to make this better though, and leave this for a later time).

Below is an explanation for each of the ensemble methods :

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest outputs a class prediction and the class with the most votes/weightage becomes the model's prediction. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

It is important that the models have a low correlation between them. Uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. This is because the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.

Bootstrap Aggregating (Bagging) is also an ensemble method and can be thought of as a precursor to random forest. Random samples of the training data set are created with replacment (sub sets of training data set). Then, a model is built (classifier or Decision tree) for each sample. Finally, results of these multiple models are combined using average or majority voting. As each model is exposed to a different subset of data and we use their collective output at the end, so we are making sure that problem of overfitting is taken care of by not clinging too closely to our training data set. Thus, Bagging helps us to reduce the variance error. Combinations of multiple models decreases variance, especially in the case of unstable models, and may produce a more reliable prediction than a single model.

Boosting is an iterative technique which adjusts the weight of an observation based on the last classification. If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa. Boosting in general decreases the bias error and builds strong predictive models. Boosting has shown better predictive accuracy than bagging, but it also tends to over-fit the training data as well. Thus, parameter tuning becomes a crucial part of

boosting algorithms to make them avoid overfitting. Boosting is a sequential technique in which, the first algorithm is trained on the entire data set and the subsequent algorithms are built by fitting the residuals of the first algorithm, thus giving higher weight to those observations that were poorly predicted by the previous model.

b. Any other model tuning measures(if applicable)

GRID SEARCH - Grid search is a tuning technique that attempts to compute the optimum values of hyper-parameters. It is an exhaustive search that is performed on specific parameter values of a model. The model is also known as an estimator.Grid search exercise can save us time, effort and resources.

The RANDOM FOREST model, CART and other models were tuned using **GRID SEARCH** to arrive at the optimum hyperparameters. Please refer to the code to understand how this can be implemented.

CROSS VALIDATION : Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.

The three steps involved in cross-validation are as follows :

1. Reserve some portion of sample data-set.
2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.

To do this numerically, we iteratively can try various CV values and keep increasing the count till it makes no significant difference to the CV score. At this point, we can freeze the CV number

and use it for further processing as shown in the following figure. In this example, CV values of 3,5 and 10 are taken and the train and test cross validation scores are observed. We see that by doubling the number from 5 to 10, not much of a difference is seen. So we freeze the CV value at 5.

```
3
Train CV Score : 0.9734524132911998
Test CV Score : 0.9670234171303861
5
Train CV Score : 0.9747641713307502
Test CV Score : 0.968019975031211
10
Train CV Score : 0.975513585259213
Test CV Score : 0.9680224438902743
```

c. Interpretation of the most optimum model and its implication on the business

Amongst all the methods tried out for this dataset, the best methods that we arrive at are **CART, Random Forest and XGBoost**. We can discard other algorthims for this purpose even though the enjoy a a good accuracy score, as their 'other' metrics such as precision & recall are not agreeable.

BUSINESS IMPLICATION : As explained earlier, i**n the case of predicting if a loan would default/foreclose — It would be better to have a high Recall** as the banks don't want to lose money and it would be a good idea to for the bank to be conservative and not release the loan if there is a slight doubt about a default.

Low precision, in this case, might be okay.

BEST MODEL : Therefore, we can choose the model with a good recall value. In our case, both RANDOM FOREST and XGBOOST models exhibit good RECALL values, as well as healthy in other metrics. Therefore, we can go for either one of the models. Given a choice, one can go for XGBOOST as there is reduced risk of overfitting