**Problem statement**

The loan default dataset has 8 variables and 850 records, each record being loan default status for each customer. Each Applicant was rated as “Defaulted” or “Not-Defaulted”. New applicants for loan application can also be evaluated on these 8 predictor variables and classified as a default or non-default based on predictor variables.

**Data set.**

1. **Bank loan.**

The data set was provided by the Edwisor team.

Here’s glimpse of data

age ed employ address income debtinc creddebt othdebt default

0 41 3 17 12 176 9.3 11.359392 5.008608 1.0

1 27 1 10 6 31 17.3 1.362202 4.000798 0.0

2 40 1 15 14 55 5.5 0.856075 2.168925 0.0

3 41 1 15 14 120 2.9 2.658720 0.821280 0.0

4 24 2 2 0 28 17.3 1.787436 3.056564 1.0

**Variables and their description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Var. #** | **Variable** | **Description** | **Variable** |
|  | **Name** |  | **Type** |
| 1. | Age | Age of each customer | Numerical |
| 2. | Education | Education categories | Categorical |
| 3 | Employment | Employment status - | Numerical |
| 4 | Address | Corresponds to job status and being converted to numeric format  Geographic area - | Numerical |
| 5 | Income | Converted to numeric values  Gross Income of each | Numerical |
| 6 | debtinc | customer Individual’s debt | Numerical |
| 7 | creddebt | payment to his or her gross income  debt-to-credit ratio is a | Numerical |
| 8 | othdebt | measurement of how much you owe your creditors as a percentage of your available credit (credit limits)  Any other debts | Numerical |

**Models and algorithms used to create the prediction model**

1. Logistic Regression
2. Decision Tree Classifier
3. Random forest classifier
4. SVM
5. Ada Boost Classifier
6. Gradient boost Classifier

At the end I used Pycaret library to determine the best possible model and to create a deployable model.

We used confusion matrix, classification report F1-score and precision recall curve to determine the accuracy and the recall and precision of the model.

1. **Confusion Matrix**
   * True positives (TP): These are cases in which we predicted yes and the true value was yes
   * True negatives (TN): We predicted no, and the true value was no.
   * False positives (FP): We predicted yes, but the actual value was No
   * False negatives (FN): We predicted no, but the actual value was yes.
2. **Accuracy:**

Accuracy=( TP+TN)/total

1. **Precision:**

Precision=TP/predicted yes

1. **RC Curve:**

ROC curve: This is a commonly used graph that summarizes the performance of a classifier over all possible thresholds. It is generated by plotting the True Positive Rate (y-axis) against the False Positive Rate (x-axis) as you vary the threshold for assigning observations to a given class

1. **F1\_score:**

F1\_score=2\*[ ( precision\* recall)/precision+recall].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1\_score |
| Logistic regression | 0.84 | 0.97 | 0.85 | 0.91 |
| Decision tree classifier | 0.73 | 0.82 | 0.84 | 0.83 |
| Random forest Classifier | 0.79 | 0.82 | 0.94 | 0.88 |
| SVM | 0.82 | 0.83 | 0.96 | 0.89 |
| Ada boost classifier | 0.84 | 0.87 | 0.92 | 0.90 |
| Gradient boost classifier | 0.80 | 0.84 | 0.90 | 0.87 |

**We used Pycaret library to test the best model. The results are shown below.**

|  | **Model** | **Accuracy** | **AUC** | **Recall** | **Prec.** | **F1** | **Kappa** | **MCC** | **TT (Sec)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **rf** | Random Forest Classifier | 0.8065 | 0.8001 | 0.3318 | 0.6459 | 0.4194 | 0.3214 | 0.3546 | 0.1550 |
| **xgboost** | Extreme Gradient Boosting | 0.8044 | 0.7803 | 0.4191 | 0.5941 | 0.4782 | 0.3653 | 0.3796 | 0.2380 |
| **lightgbm** | Light Gradient Boosting Machine | 0.8000 | 0.7890 | 0.4273 | 0.5692 | 0.4729 | 0.3572 | 0.3685 | 0.0810 |
| **lda** | Linear Discriminant Analysis | 0.7982 | 0.7962 | 0.3245 | 0.6285 | 0.4155 | 0.3070 | 0.3376 | 0.0340 |
| **gbc** | Gradient Boosting Classifier | 0.7960 | 0.7775 | 0.3518 | 0.5780 | 0.4218 | 0.3106 | 0.3309 | 0.0700 |
| **lr** | Logistic Regression | 0.7939 | 0.8033 | 0.3145 | 0.6144 | 0.4004 | 0.2907 | 0.3221 | 0.5950 |
| **ridge** | Ridge Classifier | 0.7919 | 0.0000 | 0.1918 | 0.6195 | 0.2805 | 0.2016 | 0.2560 | 0.1550 |
| **ada** | Ada Boost Classifier | 0.7876 | 0.7512 | 0.3436 | 0.5488 | 0.4099 | 0.2912 | 0.3091 | 0.0650 |
| **et** | Extra Trees Classifier | 0.7875 | 0.7759 | 0.2445 | 0.5567 | 0.3282 | 0.2288 | 0.2598 | 0.1330 |
| **catboost** | CatBoost Classifier | 0.7875 | 0.7979 | 0.2936 | 0.5808 | 0.3770 | 0.2653 | 0.2942 | 1.7340 |
| **svm** | SVM - Linear Kernel | 0.7747 | 0.0000 | 0.2555 | 0.5726 | 0.2710 | 0.1936 | 0.2517 | 0.0200 |
| **knn** | K Neighbors Classifier | 0.7579 | 0.6851 | 0.2173 | 0.4212 | 0.2803 | 0.1538 | 0.1687 | 0.0380 |
| **dt** | Decision Tree Classifier | 0.7516 | 0.6362 | 0.4291 | 0.4312 | 0.4141 | 0.2634 | 0.2701 | 0.0130 |
| **nb** | Naive Bayes | 0.7372 | 0.7429 | 0.3064 | 0.4146 | 0.2978 | 0.1689 | 0.1913 | 0.0120 |
| **qda** | Quadratic Discriminant Analysis | 0.7166 | 0.6729 | 0.3409 | 0.3724 | 0.3146 | 0.1809 | 0.1859 | 0.1070 |

**According to the results obtain we could see that the Random forest has performed the best and we used random forest to create our model. We selected this model as it had good accuracy amongst all the model and the precision , recall values where also better from all other models.**

**We also have the precision recall curve for various models.**

**Precision recall curve**

As the name suggests, this curve is a direct representation of the precision(y-axis) and the recall(x-axis). If you observe our definitions and formulae for the Precision and Recall above, you will notice that at no point are we using the True Negatives.

This is particularly useful for the situations where we have an imbalanced dataset and the number of negatives is much larger than the positives. In such cases, our higher concern would be detecting the customers correctly as possible and would not need the defaulter.

**PRC Interpretation:**

At the lowest point, i.e. at (0, 0)- the threshold is set at 1.0. This means our model makes no distinctions between the patients who have heart disease and the patients who don’t.

At the highest point i.e. at (1, 1), the threshold is set at 0.0. This means that both our precision and recall are high and the model makes distinctions perfectly.

The rest of the curve is the values of Precision and Recall for the threshold values between 0 and 1. Our aim is to make the curve as close to (1, 1) as possible- meaning a good precision and recall.

Similar to ROC, the area with the curve and the axes as the boundaries is the Area Under Curve(AUC). Consider this area as a metric of a good model. The AUC ranges from 0 to 1. Therefore, we should aim for a high value of AUC.

**Instruction for loading the model.**

* Install all the required libraries
* Load the model and the application in the command prompt by giving all the required path
* Load streamlit and run the application using streamlit using code streamlit run /path/ bank\_default\_app3.py