Final Report: Predicting Hospital Readmittance of Diabetic Patients – Team#5

1. Introduction

The management of hyperglycemia, also known as high blood sugar, has a significant impact on the medical outcomes of diabetic patients. Healthcare costs are increased as a result of hospital readmissions, which negatively affect the reputation of hospitals, as well as deter people from life-saving treatments. Identifying patients at high risk of readmission at an early stage allows providers to give special attention to these patients, reducing overall healthcare costs and improving the quality of care (Current Health n.d.). This can reduce the mortality rate of diabetic patients and healthcare expenditures. This project intends to analyze a large clinical data set and to build a binary classification model to predict early readmission (i.e., <30 days) based on the patient's features. There are three categories of readmittance: less than 30 days, greater than 30 days, or no readmittance (Beata, 2014, p. 2). For the purposes of discussion, we will say greater than 30 days and no readmittance are better outcomes than less than 30 days to minimize the cost of hospital operations. In this report we will briefly examine each step in our project such as data exploration and preprocessing and will discuss the models we have used and its evaluations.

2. Data Exploration

Our data comes from the UCI Machine Learning Repository: Diabetes 130-US hospitals for year 1999-2008 Data Set (Beata, 2014, p. 2). There are 101,766 records and 50 attributes. Each record in the 'diabetic_data.csv' file describes a unique patient's earliest admission at a participating health institution as recorded by their electronic medical record (EMR). A logistic regression model requires statistically independent observations, which was not possible in the original dataset as some patients had multiple inpatient visits. As a result, it counts only one encounter per patient; in particular, the first visit to each patient is considered the primary admission, and we determine whether the patient is readmitted within 30 days. Beata (2014) discusses how the dataset was created and gives a good understanding of each attribute and its impact on the research. The 'IDs_mapping.csv' file contains the IDs and their descriptions for the admission type, discharge disposition, and admission source. We have broken down these mappings into their own individual files (admission_type.csv, discharge_disposition.csv, and admission_source.csv) for joining onto the diabetic dataset.

To identify missing values, we identified the number of null values for variables with missing values, non-available values and ambiguous values. As we studied the structure of the data frame, we learned about the different data types, and we separated numerical and categorical columns. The target variable is readmission. In order to predict a patient's readmission to the hospital, there are certain features in the dataset that are irrelevant for the predictive model. In the next step, Python functions were implemented using the Pandas library to deal with missing values. These columns were dropped for the following reasons:

- Examide - Due to single category type existance in the attribute
- Citoglipton - Due to single category type existance in the attribute
- Weight - Due to majority null values (97% of the values missing)
- Payer Code - Due to majority null values
- Unnamed: 0 - No value addition, duplicated created in file loading
- encounter_id - No value addition to the dataset, given each value is unique for a visit

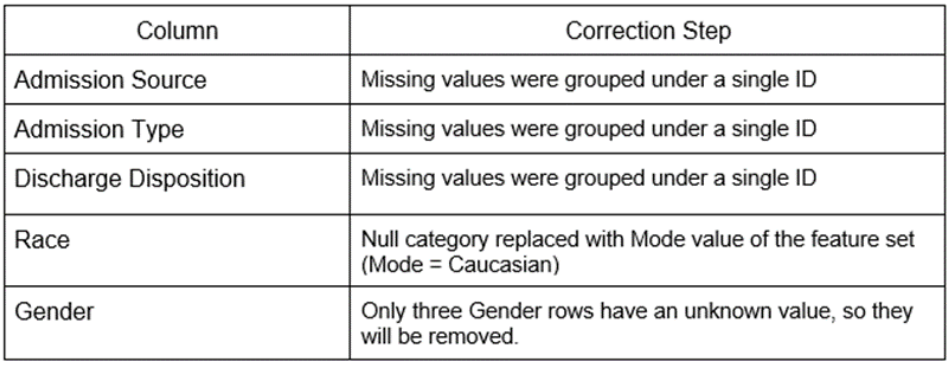
3. Methodology

The experimental process of this study included two steps: data preprocessing and model development and evaluation.

3.1. Data Preprocessing

Performing training and testing on datasets requires pre-processing before applying the algorithm. This is done by combining both train and test datasets together with a unique identifier labeled 'dataset_type'. Except for clearly stating that each training and testing set will be treated individually, all pre-processing methods are applied to the combined dataset.

3.1.1. Null Value Corrections

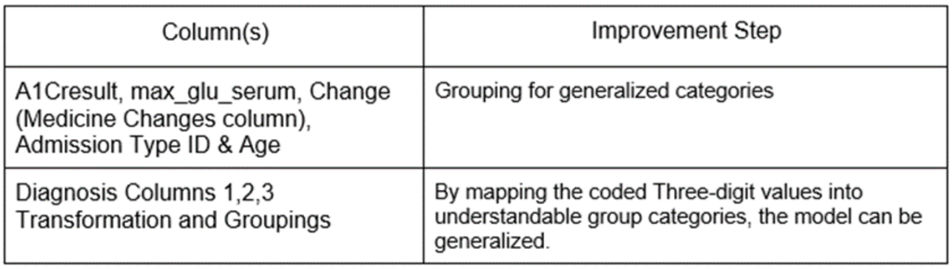


Column	Correction Step
Admission Source	Missing values were grouped under a single ID
Admission Type	Missing values were grouped under a single ID
Discharge Disposition	Missing values were grouped under a single ID
Race	Null category replaced with Mode value of the feature set (Mode = Caucasian)
Gender	Only three Gender rows have an unknown value, so they will be removed.

Table 01: Null Value Corrections

3.1.2. Modifications of Features Set

There is a need to prop up some categories to generalize a model well to a better extend. For A1Cresult, max_glu_serum, Change (Medicine Changes column), Admission Type ID & Age columns we did grouping and encoding them with some meaningful values and assigned the outliers values to the missing categories in addition to generalized categories as improvement steps. For diagnosis columns 1,2,3 we used functions in order to transform and groupings into valuable categories so that the models will be generalized well. After this we encoded all columns before given to the model using to ensures all required columns for modeling are in numerical format using Label Encoding, OneHot Encoding, and Rule based Encoding techniques.

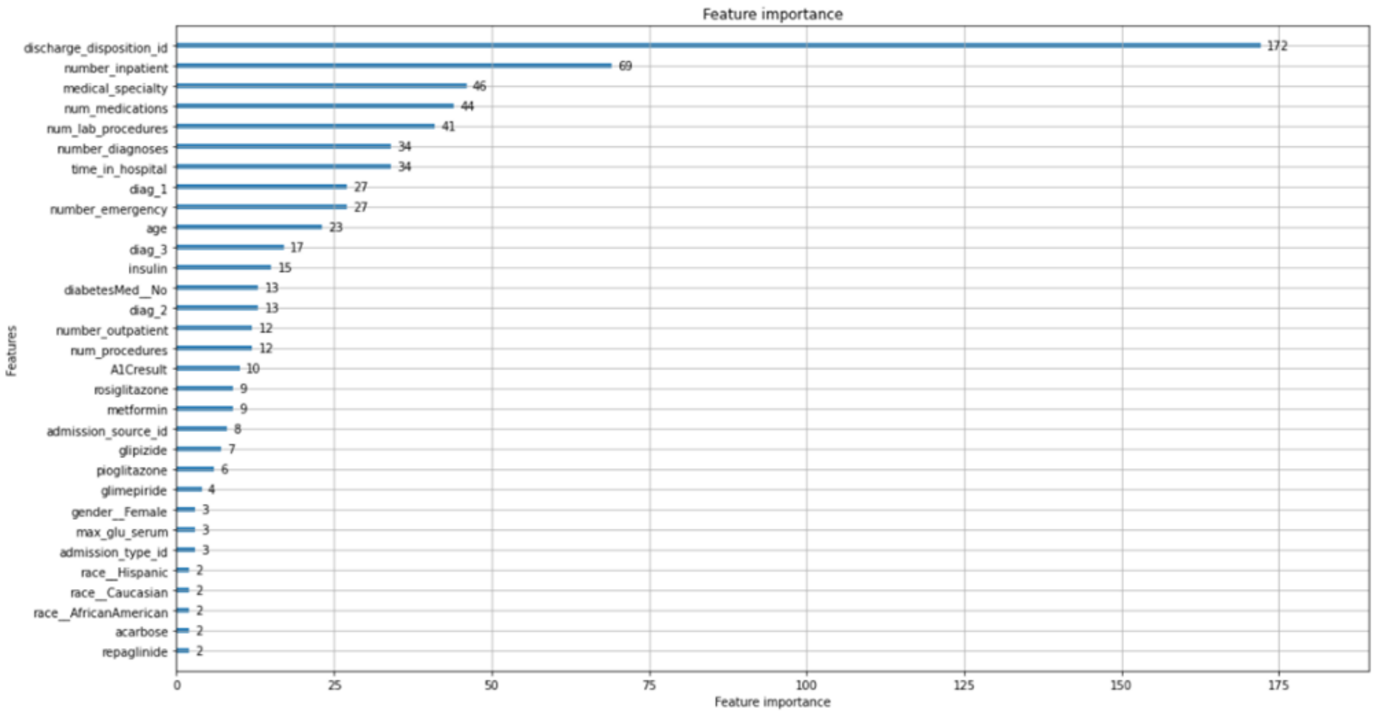


Column(s)	Improvement Step
A1Cresult, max_glu_serum, Change (Medicine Changes column), Admission Type ID & Age	Grouping for generalized categories
Diagnosis Columns 1,2,3 Transformation and Groupings	By mapping the coded Three-digit values into understandable group categories, the model can be generalized.

Table 02: Modifications of Features Set

3.2. Feature Selection

In order to reduce input variables in our model, we select the best features of the data and eliminate noise in the data. As part of this process we solved the unneeded columns that were dropped due to issues. Multiple methodologies are used for feature selection, such as Test RFECV (Recursive feature elimination with cross-validation).



After extensive testing, we implemented two methods to find the most useful set of features, using BernulliNB based RFECV (Recursive feature elimination method with cross-validation) since the majority of data is discrete and LGBM classifier to produced 40 features, where discharge_disposition_id has the highest weightage and I selected the top 15 feature using this feature selection approach.

We realized we have an imbalance dataset (Class 0: 4441, Class 1: 52707). To solve this issue, we used SMOTEEN since it has been the priority method for solving class imbalance because it utilizes both types of sampling methods (Unsampling & Oversampling to solve the class imbalance issue). Results from the SMOTE resulted in random values of the Target Class '1" not being randomized properly, causing the recall and precision of the said target class attribute to fall way below normal conditions.

3.3. Models Development

Due to the categorical nature of the dataset, we have applied decision tree, random forest, and LightGBM algorithms to determine the major importance factor between the variables since its information gain system will allow us to measure each feature's importance accordingly. In addition, since decision tree-correlation between categorical data varies a lot, we decide to use it to assess the importance of each feature based on its information gain system. The random forest algorithm is used to find the best tree and LightGBM as a tree-based learning algorithm is known for great performance and faster training speed and higher efficiency.

3.3.1. Decision Tree Model

Based on the patient's features and the categorical characteristics of the data, we use the decision tree model to construct a binary classification model that is able to predict early (<30 days) readmission. The major advantage of a decision tree is that it forces the consideration of all possible outcomes and traces each path to its conclusion.

For the decision tree classifier, we used “entropy” as the *criterion* and set the *max_depth* to 10 and the *splitter* was chosen “random”. However, we used a function to limit the *max_depth* to find the one that has the maximum validation set accuracy. When *max_depth* is 10 the model has the highest accuracy. Results for whole data: Accuracy with train set: 0.78, Accuracy with Test set: 0.60, Precision score: 0.89 Recall score: 0.62, F1 score: 0.73. A trial-and-error procedure was followed to determine which parameters worked best and to tune the parameters to improve performance.

DT model Analysis: In training and test sets using *max-depth*, *min_sample_leaf*, and *criterion* as tuning hyperparameters, using GridSearchCV method, resulted in an increase in overall accuracy, the AUC increased minutely. The recall score for target class '1' dropped by a few percent. The Decision Tree model is shown to be able to predict all the relevant '1' falls. Class ‘1’ indicates patients revisiting hospital within <30 days.

From the model we can conclude that around 25% of patients readmitted within 30 days (i.e <30days target class) are misclassified and the accuracy of the model just slightly increases after hyperparameter tuning. Nevertheless, 14% of patients remain misclassified. The accuracy of the train and test data of the model does not differ significantly after it is run. This is because of major class imbalances in the data. Images of the output are shown in Figures 01, 02, and 03 in appendix.

3.3.2. Random Forest Model

To improve bagging, we used a random forest model to check if accuracy could increase. In addition to bagging, we used the random forest model since it is suitable for either regression or classification tasks and it can handle binary, categorical, and numerical data types.

For our random forest classifier, we set *n_estimators* to 1500, criteria to "entropy", *random_state* to 42, *depth* to 15, and *features* to 22. Result for whole data: Accuracy with train set: 0.92, Accuracy with Test set: 0.72, Precision score: 0.89 Recall score: 0.77, F1 score: 0.83. We used trial-and-error to identify which parameters improved performance and tuned the parameters accordingly.

RF model Analysis: In training and test sets, *max_depth*, *n_estimators*, *max_features*, and *min_samples_split* as tuning hyperparameters, using GridSearchCV method, resulted in an increase in overall accuracy, the AUC increased minutely. A few percent dropped from the recall score for the target class '1'. It shows how a

Decision Tree model is able to predict each class '1' fall. Class ‘1’ indicates patients revisiting hospital within <30 days.

From the model we can conclude that around 12% of patients readmitted within 30 days(i.e <30days target class) are misclassified and the accuracy of the model just slightly decreases after hyperparameter tuning, Where still 15% of patients are misclassified. After tuning, accuracy decreases, which indicates data imbalances between classes. Images of the output are shown in Figures 04, 05, and 06 in appendix.

3.3.3. Light Gradient Boosting Model (LightGBM)

A Gradient Boosting Model (LightGBM) utilizes decision trees to optimize the model's performance and reduce memory consumption. It consists of two techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB), which overcome the limitations of the traditional histogram-based algorithm, which is typically used in GBDT (Gradient Boosting Decision Tree) frameworks. Since we have a large dataset including more than 100 thousand records, we used LightGBM since it is well known for its powerful framework, speed and parameter tuning. For the LGBMC classifier we set *learning_rate* to 0.05, *max_depth* to 3, *n_estimators* to 200, and subsample to 0.1. We used a trial-and-error method to identify variables that improved accuracy and tuned parameters accordingly.

LGBM model Analysis: In training and test sets, tuning of hyperparameters such as *learning_rate*, *n_estimators*, subsample and *max_depth*, using GridSearchCV method, caused a significant decrease in overall accuracy (test set - 0.77 to 0.64), the AUC decreased minutely. The recall score for the target class '1' improved fairly considerably (0.32-0.42) for boosting type = 'dart'. Precision remained nearly the same. There is a quick and significant trade-off between the recall score of Target class '1' and the accuracy of the overall prediction.

According to the model, approximately 10% of patients with a 30-day readmission are misclassified, and the accuracy of the model increases by just a bit after tuning hyperparameters, where only 9% of patients are misclassified. Accuracy Remains constant after tuning hyperparameters. Images of the output are shown in Figures 07, 08, and 09 in appendix.

3.4. Models Performances

Data preparation and training of a machine learning model are important steps in the machine learning pipeline. However, it is also essential to measure and evaluate the performance of this trained model. Most model-performance measures are determined by comparing the predictions of the model with actual values of the dependent variable in a dataset.

We split data into two parts of the train set and test set to implement all the models and used cross validation to estimate the measures. Cross validation is one of the techniques used to test the effectiveness of machine learning models.

To assess the performance of our models, we started using the confusion matrix since it provides an in-depth breakdown of the correct and incorrect classifications of each class. Other metrics used to determine the performance of our models include accuracy scores, precision scores, recall scores, and f1 scores. As precision is focusing on type -1 errors, and recall on type 2 errors, we used precision and recall in order to provide an idea of incorrect classification of target classes in the model. As part of our evaluation, we also used the AUC (Area under the ROC curve) metric. The AUC metric is useful for choosing the best model since it captures the essential balance between true and false positives. AUC represents a model's ability to distinguish between positive and negative classes in a dataset.

Below is the table 03, showing the differences between each model in terms of different metrics.

Models	Accuracy on Train set		Accuracy on Test set		Precision (Target class 1)		Precision (Target class 1)		F1-score (Target class 1)		AUC	
	Base Model	Tunned Model	Base Model	Tunned Model	Base Model	Tunned Model	Base Model	Tunned Model	Base Model	Tunned Model	Base Model	Tunned Model
Decision Tree	83.51	92.38	76.52	77.08	0.89	0.90	0.83	0.84	0.86	-	0.54	0.52
Random Forest	93.99	86.97	85.59	75.94	0.89	0.90	0.95	0.82	0.92	-	0.61	0.59
LGBM	97.42	88.86	82.65	82.85	0.90	0.90	0.90	0.91	0.90	-	0.62	0.6

Table 03: Summary of performances of different models

4. Evaluation

Due to the categorical nature of the dataset, we have applied decision tree and random forest and LightGBM algorithms to determine the major importance factor between the variables since its information gain system will allow us to measure each feature's importance accordingly. In model training, the most challenging issue is class imbalance of the data, as the data (predict <30) is limited and related to health. In order to solve this problem, we used smote, under sampling, and oversampling.

Due to a lot of discrepancies in this real-world dataset, the models were sub optimized. Even solving the class imbalance problem through sampling techniques had some effect on the end-result. Our study has implemented multiple models including decision trees, random forests, and light gradient boosting models, out of which LGBM produced the best accuracy results (0.9570 - train & 0.77500 - test) and AUC values (0.6). Best Features were extracted using LGBM Classifier and BernulliNB based RFECV. The recall and precision for "*30 admissions" are inversely related to the model's accuracy.

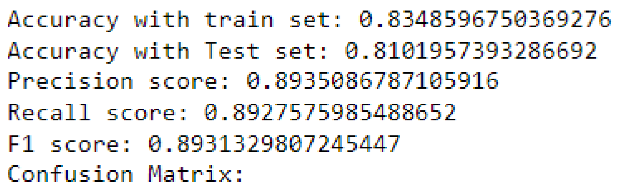
For the further steps and the dataset is hugely imbalanced, adding further samples to the dataset could improve the result for the recall and precision rates for "<30 admissions". Moreover, since SVM and naive bayes models have been implemented, tuning their hyperparameters would also be beneficial.

5. Reference

1. Current Health. (n.d.). 3 Strategies to Address the $40B Hospital Readmission Problem. https://www.currenthealth.com/resources/blog/3-strategies-to-address-the-40b-hospital-readmission-problem/
2. Beata, S. & Jonathan P,. D., Chris G., Juan L., O., Sebastian, V., Krzysztof J., C., & John N., C., (2014). “Impact of HbA1c Measurement on Hospital Readmission Rates”: Analysis of 70,000 Clinical Database Patient Records. *Hindawi Publishing Corporation, BioMed Research International, Volume 2014, Article ID 781670,* 11 pages. https://doi.org/10.1155/2014/781670
3. Jochanan, B. & Mark, T. (2000). Hospital Readmissions as a Measure of Quality of Health Care Advantages and Limitations. *Health Policy Research Program, JDC Brookdale Institute, Jerusalem, Israel*. https://jamanetwork.com/journals/jamainternalmedicine/fullarticle/415392
4. Andrea C, T., Noah M, I., Jeremy M, G., David, M., Lucy, T., James, G., Ilana, H., Brigitte, V., T, R., Braden, M., Marcello, T., & Kaveh, S. (2012). *The Lancet Journal*, Pages 2252-2261. https://www.sciencedirect.com/science/article/pii/S0140673612604802
5. Raghav, P. (2018, July 03). *Understanding Hyperparameters and its Optimisation techniques*. Towards Data Science. https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568
6. Mauro Di, P. (2020, May 11). *Machine Learning with Python: Classification*. Towards Data Science. https://towardsdatascience.com/machine-learning-with-python-classification-complete-tutorial-d2c99dc524ec
7. *sklearn.feature_selection.RFECV*.Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html

Appendix: Model Outputs

1. Decision Tree Model



```
Accuracy with train set: 0.8348596750369276
Accuracy with Test set: 0.8101957393286692
Precision score: 0.8935086787105916
Recall score: 0.8927575985488652
F1 score: 0.8931329807245447
Confusion Matrix:
```

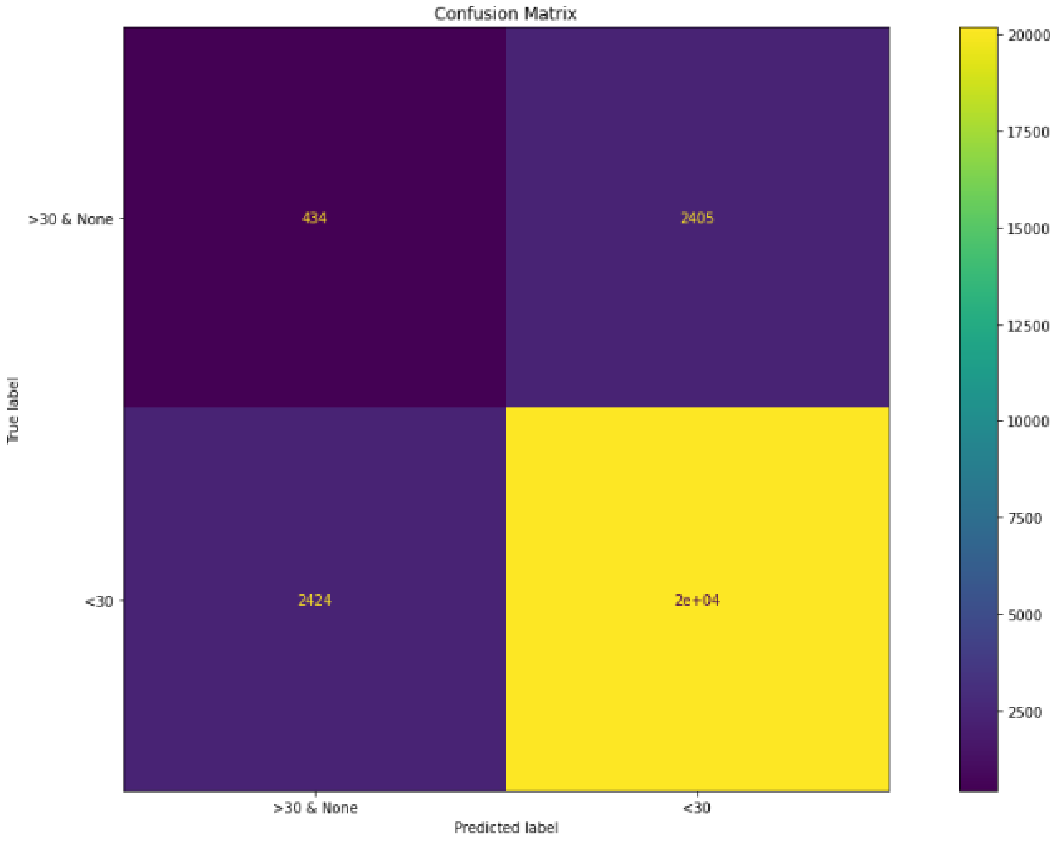
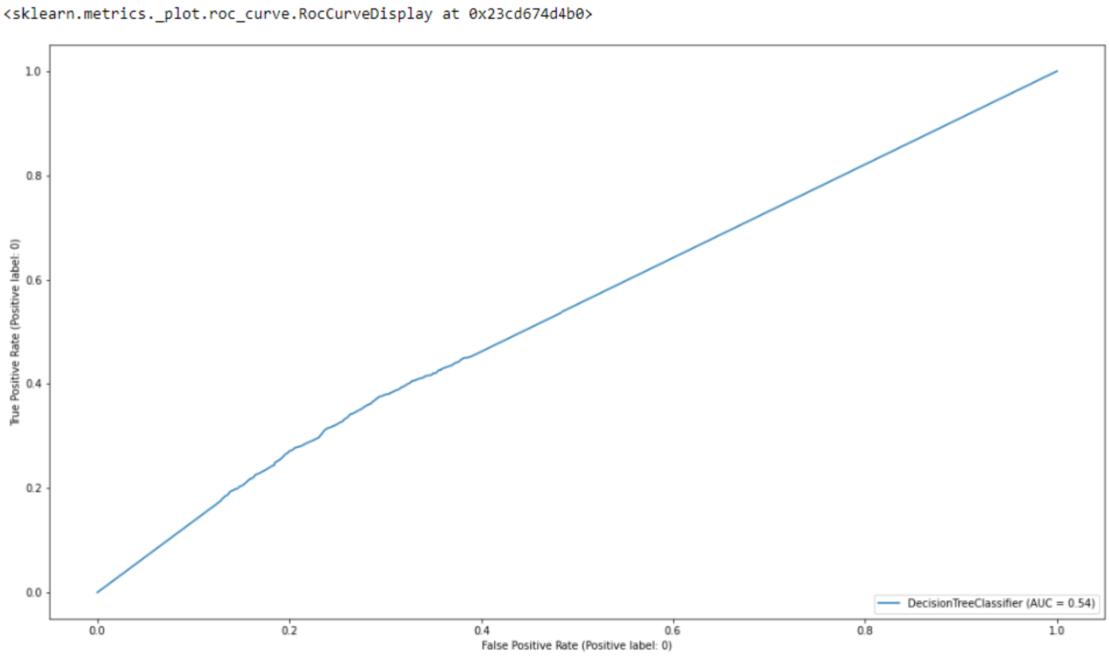
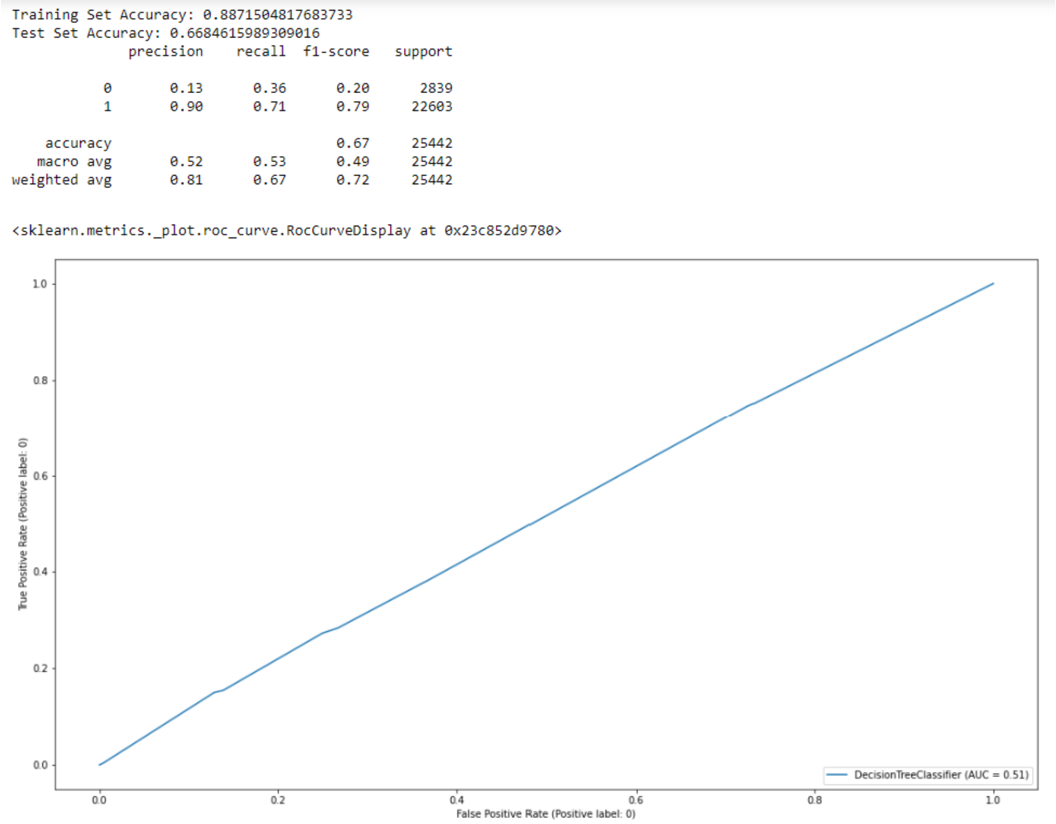


Figure 01: Decision tree model: Classifier result and confusion matrix



```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23cd674d4b0>
```

Figure 02: Decision tree model: ROC curve plot

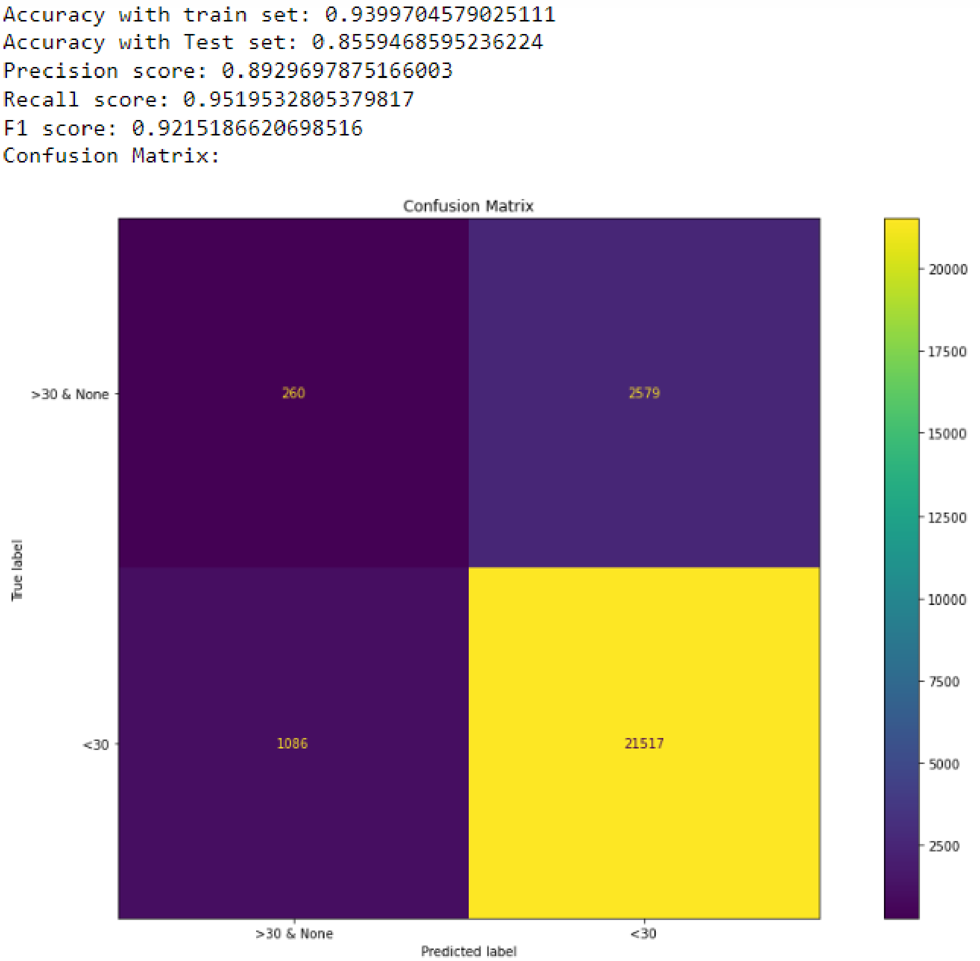


```
Training Set Accuracy: 0.8871504817683733
Test Set Accuracy: 0.6684615989309016
              precision    recall  f1-score   support

           0       0.13      0.36      0.20      2839
           1       0.90      0.71      0.79     22603

    accuracy                           0.67     25442
   macro avg       0.52      0.53      0.49     25442
weighted avg       0.81      0.67      0.72     25442

<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23c852d9780>
```

True Positive Rate (Positive label: 0)
False Positive Rate (Positive label: 0)
0.0
0.2
0.4
0.6
0.8
1.0
DecisionTreeClassifier (AUC = 0.51)

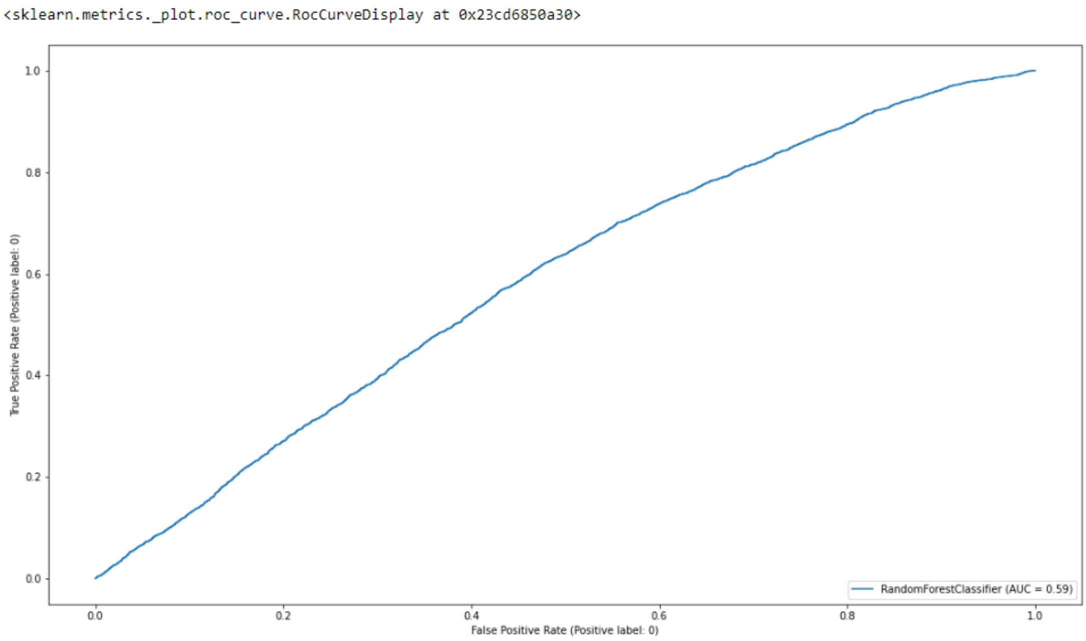
Figure 03: Decision tree model: Re-training predictive model with tuned hyperparameters

2. Random Forest Model



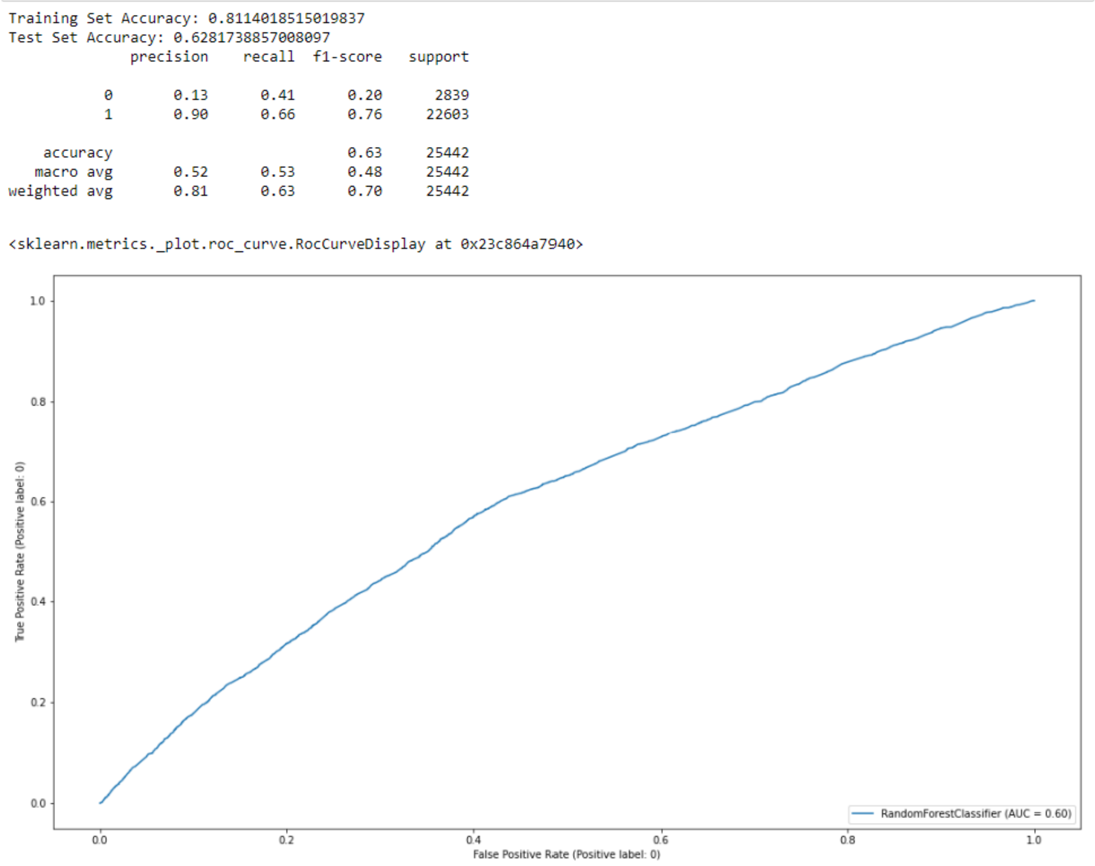
```
Accuracy with train set: 0.9399704579025111
Accuracy with Test set: 0.8559468595236224
Precision score: 0.8929697875166003
Recall score: 0.9519532805379817
F1 score: 0.9215186620698516
Confusion Matrix:
```

Figure 04: Random Forest model: Classifier result and confusion matrix



```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23cd6850a30>
```

Figure 05: Random Forest model: ROC curve plot

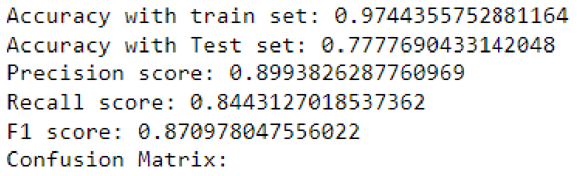


```
Training Set Accuracy: 0.8114018515019837
Test Set Accuracy: 0.6281738857008097
              precision    recall  f1-score   support

           0       0.13      0.41      0.20      2839
           1       0.90      0.66      0.76     22603

    accuracy                           0.63     25442
   macro avg       0.52      0.53      0.48     25442
weighted avg       0.81      0.63      0.70     25442

<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23c864a7940>
```

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 0)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 0)
RandomForestClassifier (AUC = 0.60)

Figure 06: Random Forest model: Re-training predictive model with tuned hyperparameters

3. LightGBM Model



```
Accuracy with train set: 0.9744355752881164
Accuracy with Test set: 0.7777690433142048
Precision score: 0.8993826287760969
Recall score: 0.8443127018537362
F1 score: 0.870978047556022
Confusion Matrix:
```

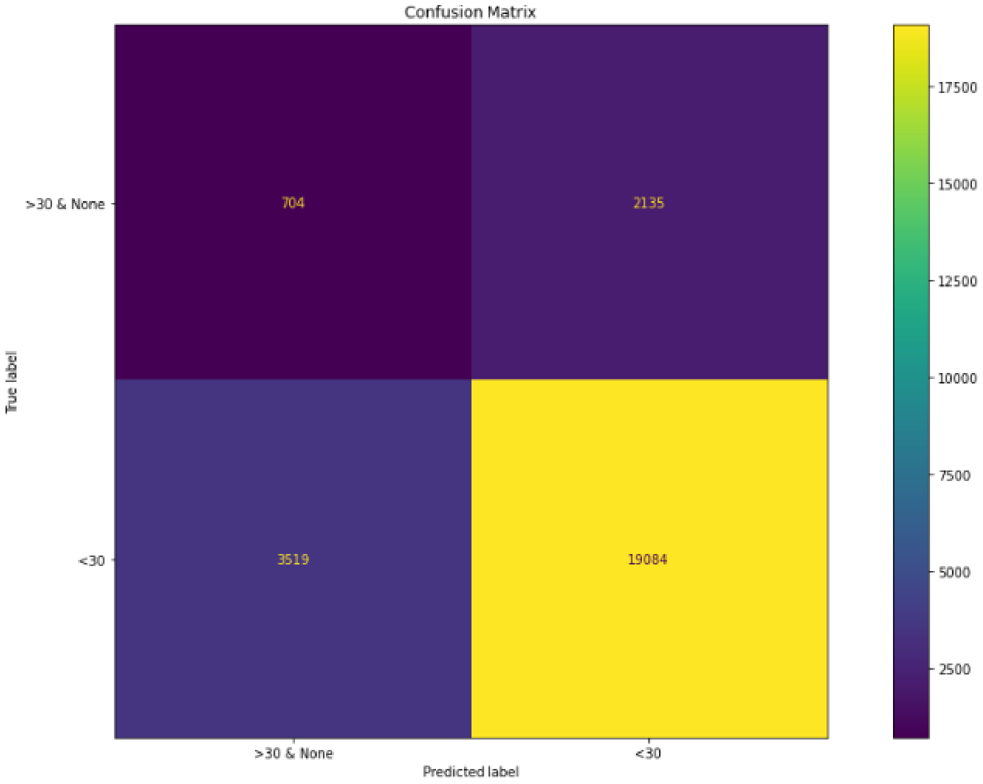


Figure 07: LightGBM model: Classifier result and confusion matrix

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23cd68e5cf0>
```

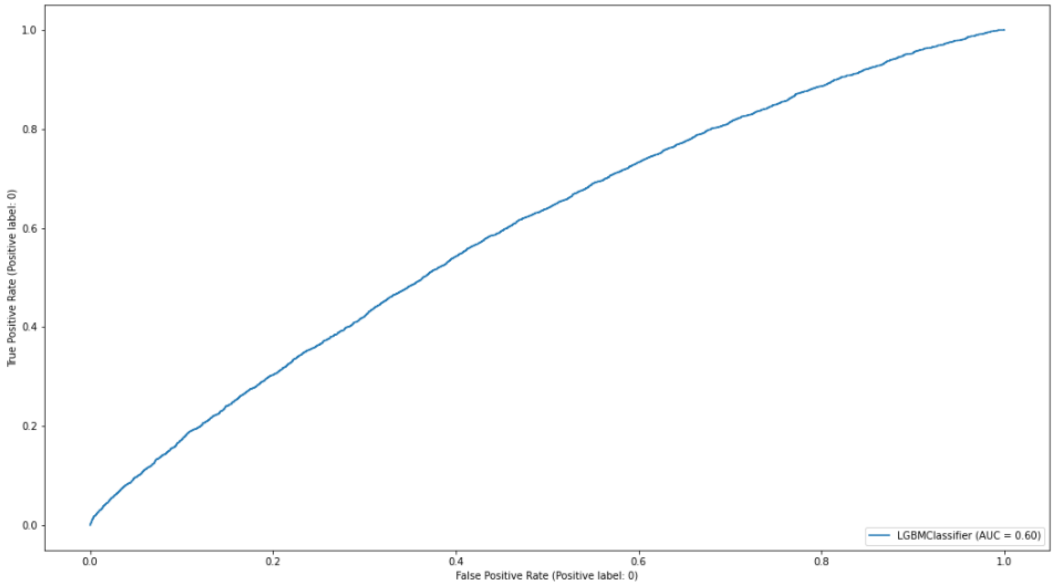
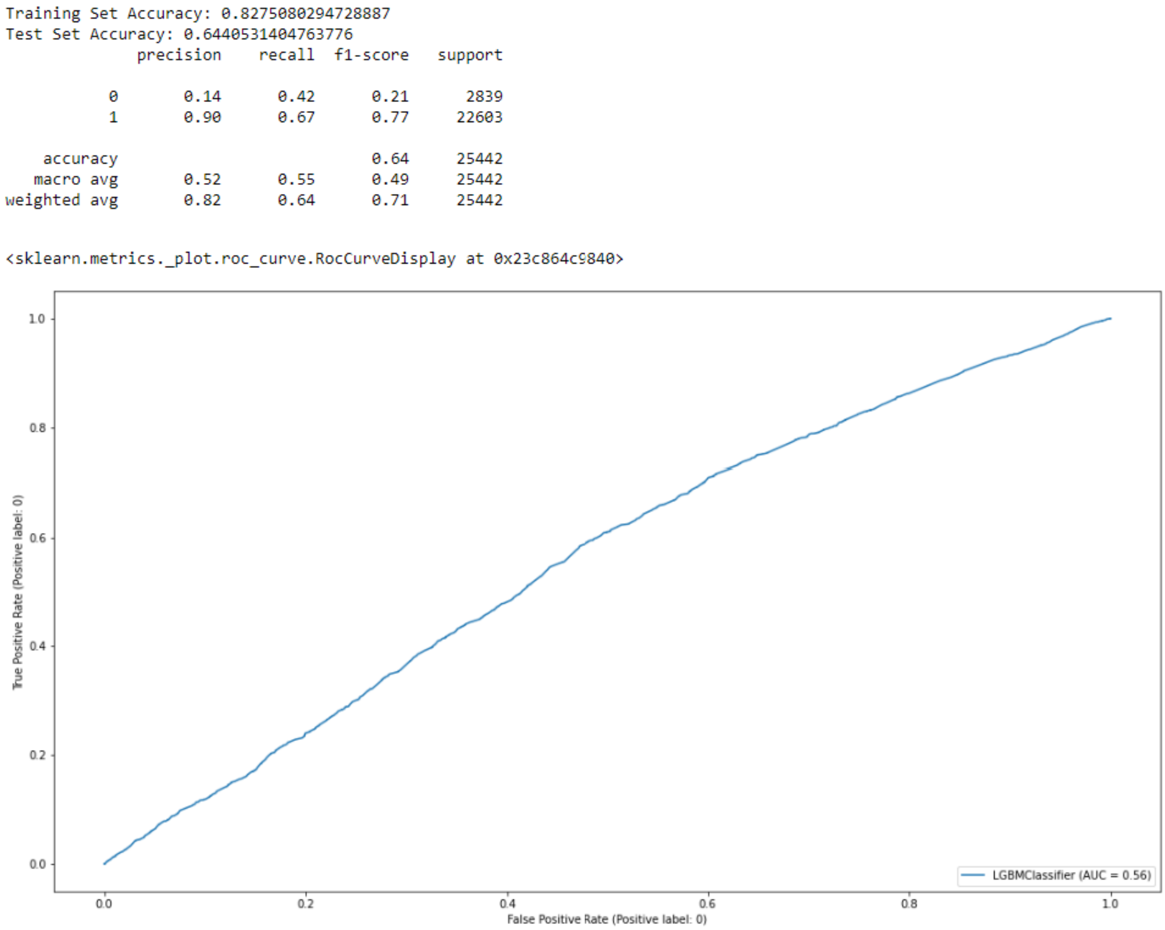


Figure 08: LightGBM model: ROC curve plot



```
Training Set Accuracy: 0.8275080294728887
Test Set Accuracy: 0.6440531404763776
              precision    recall  f1-score   support

           0       0.14      0.42      0.21      2839
           1       0.90      0.67      0.77     22603

    accuracy                           0.64     25442
   macro avg       0.52      0.55      0.49     25442
weighted avg       0.82      0.64      0.71     25442

<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x23c864c9840>
```

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 0)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 0)
LGBMClassifier (AUC = 0.56)

Figure 09: LightGBM model: Re-training predictive model with tuned hyperparameters