TERM DEPOSIT SUBSCRIPTION PREDICTION

ISHAL ABHISHEK MUMMIDIVARAPU . RAMYA KONDRAKUNTA . SAI VENKATA MANOJ VUNGARALA

ABSTRACT

What are term deposits? Term deposits are fixed-term investments which involve deposit of money to an account in a financial institution such a bank for instance. Who would subscribe to a term deposit? The bank must reach out to multiple candidates who might subscribe to their term deposit. This can involve either reaching out to their existing customers or generating new leads by taking the time to talk with them about how a term deposit is beneficial and a good short term investment. Searching prospective candidates for the term deposit subscription is a time consuming, expensive job and so these financial institutions usually hire marketing companies to do it for them. Marketing companies would pursue a wide range of leads which includes both potential subscribers and those who wouldn't subscribe. Trying to pursue the latter would be a complete waste of time.

With the help of this term deposit predictor, the bank can effectively identify which of these leads are potential subscribers and which are not. This way they can focus on pursuing only potential candidates and not waste time on others. They can also predict whether a candidate will subscribe or not based on their profile and thus can easily generate their own leads rather than depend on a marketing company. Using the data collected by a telemarketing company hired by a Portuguese banking institution, we intend to build a predictive model which we believe would save time, money and increase search efficiency if implemented by the bank.

STATEMENT OF CONTRIBUTION

Ishal Abhishek Mummidivarapu:

Data preprocessing
Undersampling the data and hyperparameter tuning for it

Ramya Kondrakunta:

Exploratory data analysis
Performing SMOTE on data and hyperparameter tuning for it

Sai Venkata Manoj Vungarala:

HTML Deployment
Oversampling data and hyperparameter tuning for it

Note: All team members have contributed in documentation such as project proposal, project presentation and project report.

INTRODUCTION

This project aims to build a term deposit predictive model. The *data* we have used is taken from the UCI Machine Learning Repository. This data was collected from a telemarketing campaign by the Portuguese Banking Institution. We have used the “bank-full.csv” from the data source. It has 16 input variables and 1 output variable. The data is described below.

Input Variables

1. *age*: Age of the candidate approached. It is a numeric variable.
2. *job*: Candidate’s job. It is a categorical variable. Job categories are 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed'
3. *marital*: Candidate’s marital status. It is a categorical variable. Marital status categories are 'divorced', 'married', 'single’. Note that 'divorced' means divorced or widowed
4. *education*: Candidates’ education level. It is a categorical variable. Education level categories include 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree'
5. *default*: Indicates if the candidate has credit in default or not. It is a categorical variable with categories ‘yes’ or ‘no’.
6. *balance*: It is the account balance of the candidate. It is a numeric variable.
7. housing: Indicates if the candidate has a housing loan or not. It is a categorical variable with categories ‘yes’ or ‘no’.
8. *loan*: Indicates if the candidate has a personal loan or not. It is a categorical variable with categories ‘yes’ or ‘no’.
9. *contact*: The contact communication type It is a categorical variable with categories 'cellular' or 'telephone'
10. *day*: Last day the candidate was contacted. It is a numeric variable.
11. *month*: Month that the candidate was last contacted. It is a categorical variable with categories ‘jan’, ‘feb’, ‘mar’ and so on.
12. *duration*: Duration of call when the candidate was last contacted. This attribute affects the output significantly. It is a numeric variable.
13. *campaign*: Number of contacts performed this campaign for this candidate. It is a numeric variable.
14. *pdays*: Number of days that passed by after the client was last contacted from a previous campaign. It is a numeric variable.
15. *previous*: Number of contacts performed in the previous campaign for this candidate. It is a numeric variable.
16. *poutcome*: Outcome of the previous marketing campaign. It is a categorical variable with categories 'failure', 'nonexistent', 'success'.

Output Variables

1. *y*: The response variable represents the subscription status of a candidate. It is a categorical variable with categories ‘yes’ or ‘no’.

FEATURE EXTRACTION AND PROCESSING

Data Analysis before preprocessing the data

The data set has 16 predictors, one response variable and has 45211 observations. Exploratory data analysis is important to understand what data has to offer us before we work on it. We have found a few interesting relations between the predictors and response variables.

Fig.1 shows that longer the duration of call, more candidates have subscribed to the term deposit. Fig.2 is a plot showing only candidates who have subscribed. We can observe that candidates between 25-50 years speak for a longer time. As the age increases the duration of call seems to decrease. This probably happens as younger candidates are more interested in the details of the investment than the older candidates.

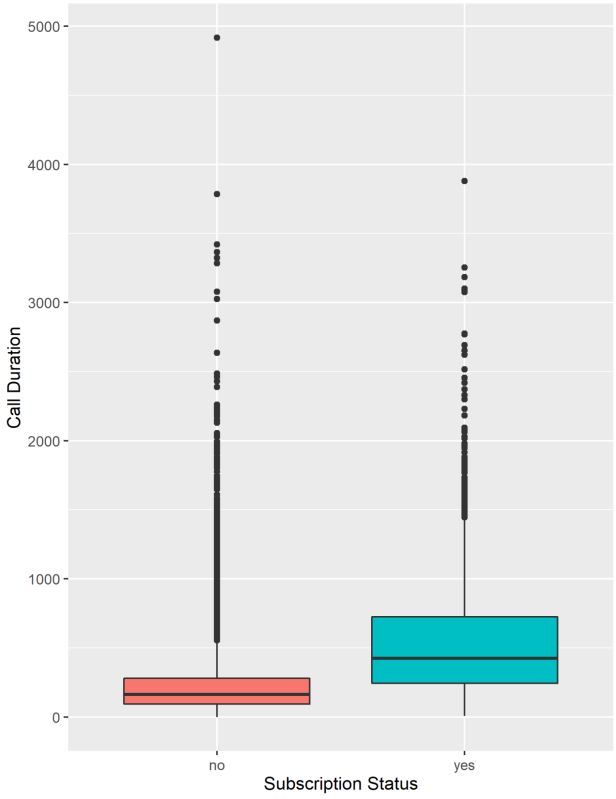


Figure 1

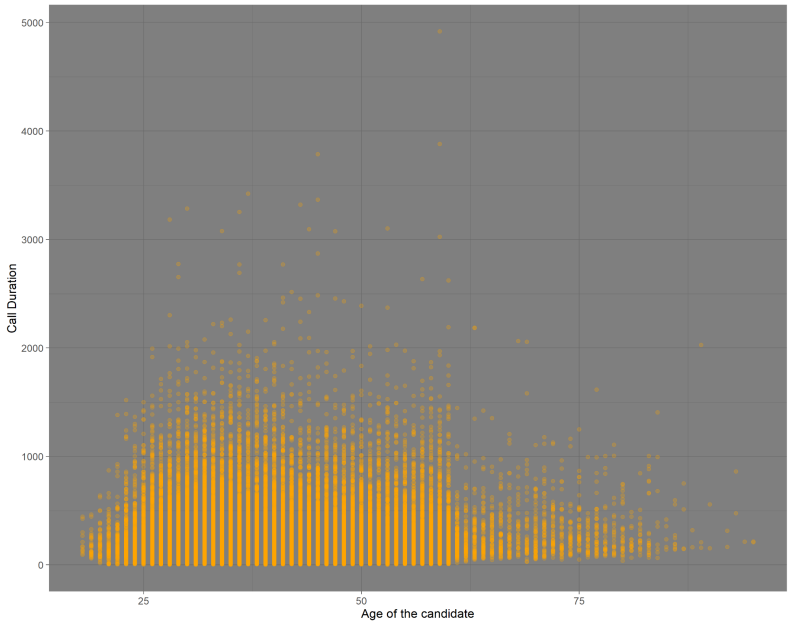


Figure 2

Preprocessing and Feature Extraction

Handling Categorical Attributes

As seen in the data description, we had multiple categorical attributes to deal with. One can encode these categories into numeric values before plotting a heatmap and proceeding towards modeling.
We have used ordinal encoding to do this. It is also called integer encoding where the categories are encoded beginning from 0. This encoding is easily reversible. Ordinal encoding as the name suggests induces a sense of order or hierarchy. Integers naturally have an order; this helps while training a model. The categorical variables have therefore been encoded with integers in an order of low to high based on the count for categories that resulted in a *no* to the subscription and which ones resulted in a *yes*.

Fig.3 shows plots that indicate the order of importance (high to low) for the categories of various attributes for the subscription status to be a *yes*.

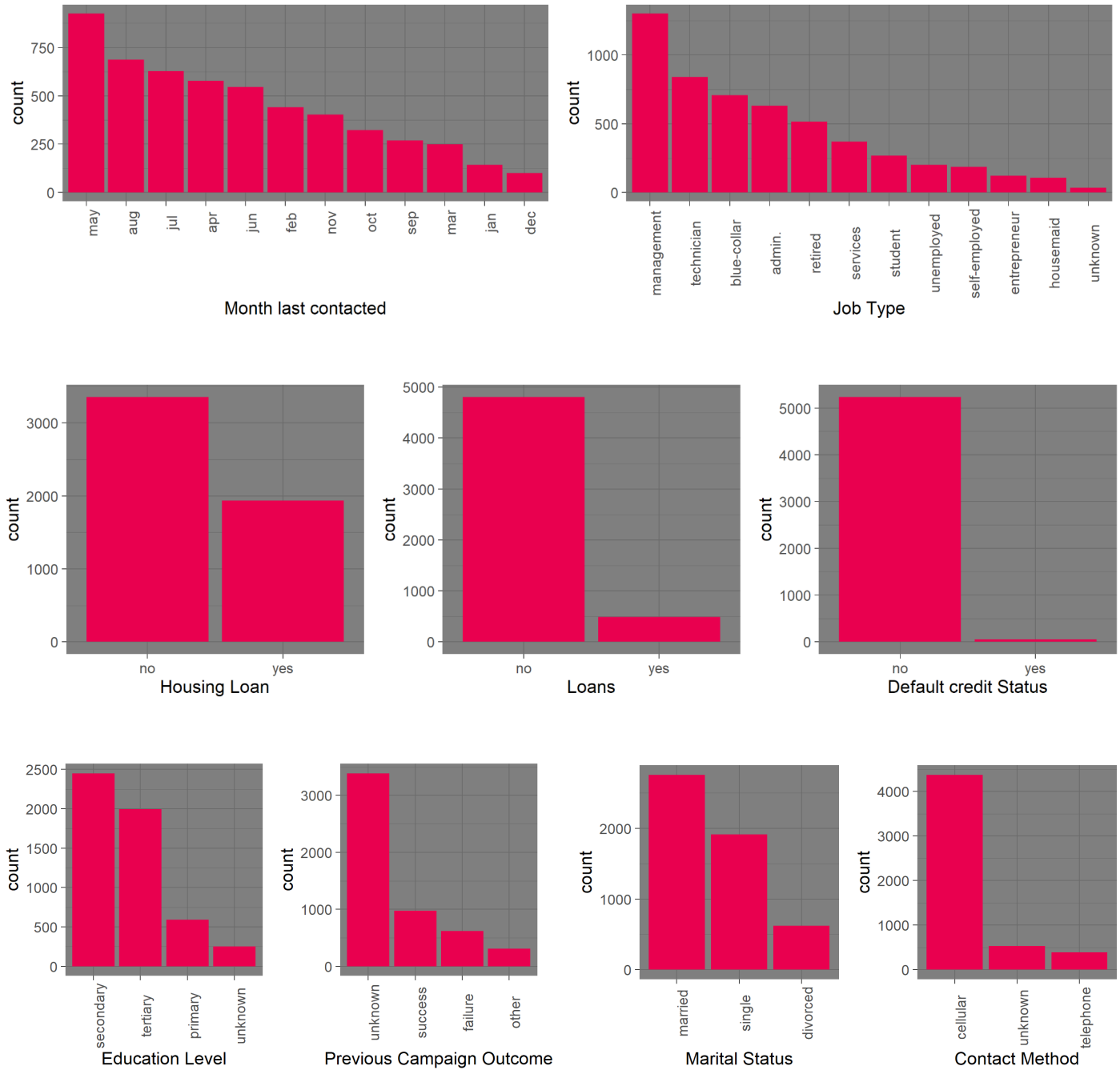


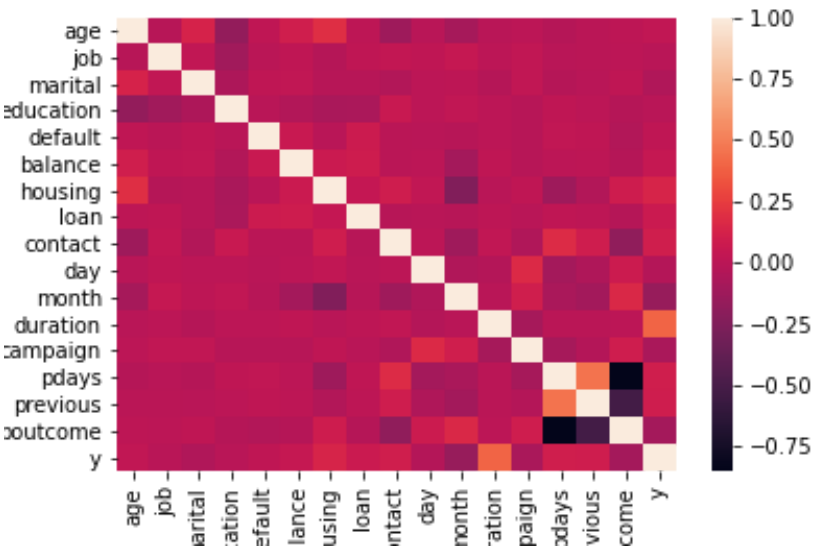
Figure 3

For better understanding, let's take an example. Consider the last plot i.e. 'Contact Method' plot. Cellular has the highest count for a candidate to subscribe to the term deposit followed by unknown and then telephone. Hence, the encoding for categories of contact attribute would be: telephone=0; unknown=1; cellular=2.
Note that the 'unknown' categories are not considered as NAs since in real-time these values might be unknown for various reasons. To make our model more robust we have considered it as a category.

The processed data's heatmap is as shown in Fig.4. The table shows the correlation of all the attributes with the response attribute i.e. y.

y	1.000000
duration	0.394521
housing	0.139173
pdays	0.103621
contact	0.100822
previous	0.093236
loan	0.068185
balance	0.052838
age	0.025155
default	0.022419
job	-0.005085
education	-0.009795
day	-0.028348
marital	-0.043846
campaign	-0.073172
poutcome	-0.096257
month	-0.150419

Figure 4



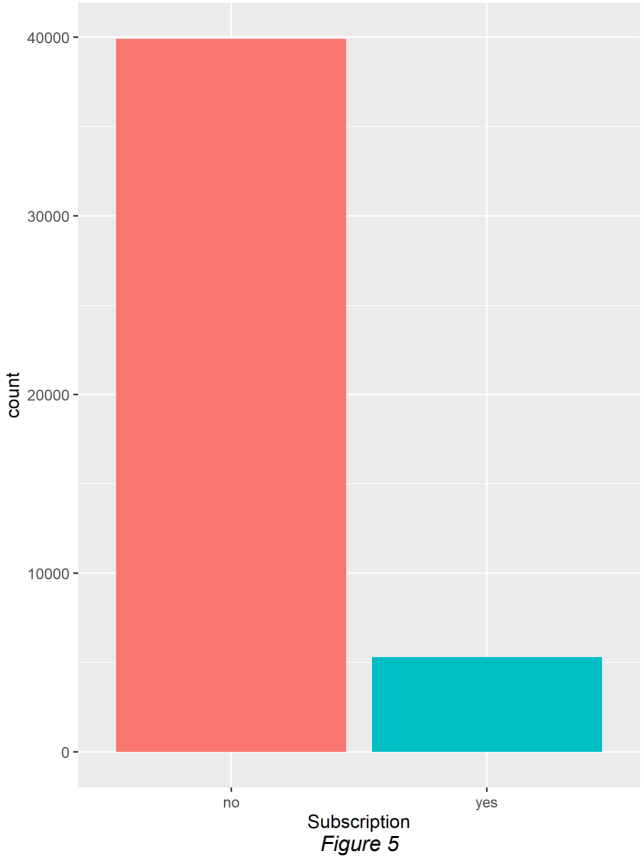


Figure 5

From the above observations we can say that ***duration, housing, pdays, contact, month, poutcome*** and ***previous*** are best correlated with the response variable. Of all, duration is most strongly correlated.

In Fig.5 we can clearly see a class imbalance problem. Class imbalance can be dealt most commonly by data augmentation of two types: undersampling and oversampling.

Handling class imbalance

1. Oversampling

It involves randomly selecting examples from minority classes, with replacement, and adding them to the training data set.

2. Synthetic Minority Oversampling Technique(SMOTE)

It is a statistical technique for increasing the number of cases in the dataset in a balanced way.

3. Undersampling

It keeps all of the data in the minority class and decreases the size of the majority class.

MODELS

Decision trees

- The data is continuously split according to a certain parameter
- Tree consists of
    - nodes : test for the value of a certain attribute
    - edges/ branch : correspond to the outcome of a test and connect to the next node or leaf
    - leaf nodes : terminal nodes that predict the outcome (represent class labels or class distribution)

Random Forest

- Is an ensemble learning method that operates by constructing multitude of decision trees during training time
- In classification problems, output is the class selected my most trees
- Usually outperform decision trees but have lower accuracy that gradient boosted trees

Logistic Regression

- At the simplest of its form, we perform binary classification using logistic regression
- Its called logistic regression as we use logistic function that is nothing but the sigmoid function
- We can perform classification even for multiple classes

XGBoost

- XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework

Naive Bayes

- A Naive Bayes classifier is a probabilistic machine learning model
- The crux of the classifier is based on the Bayes theorem

Evolution of Decision Trees

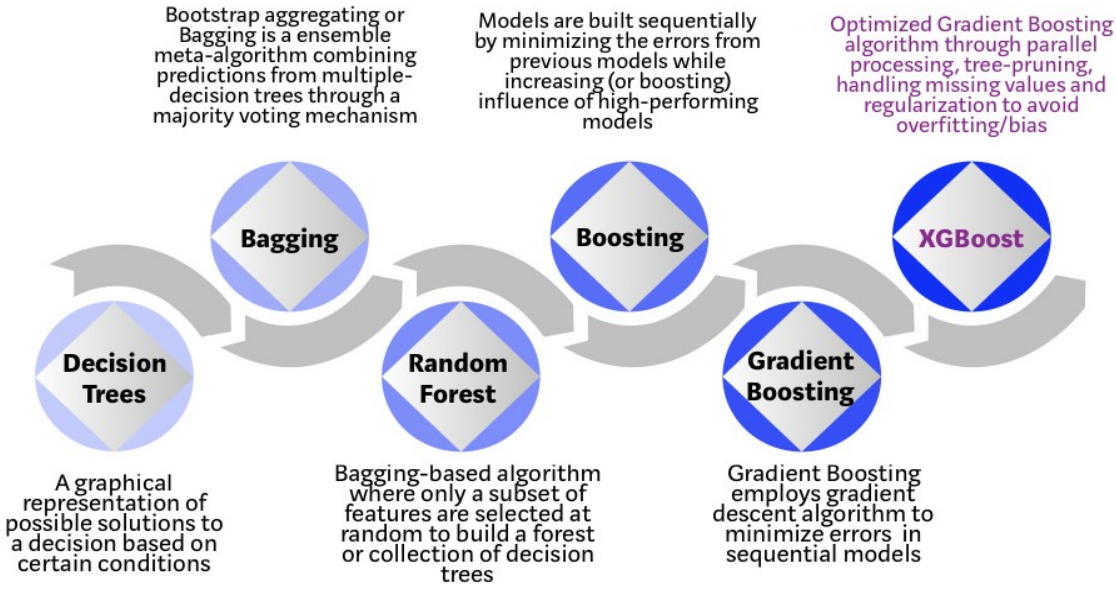


Figure 6

Initial Analysis of all models on the data

The processed dataset was split as 80 percent train data and 20 percent test dataset. All the models were trained with default parameters. Class imbalance was not handled while performing the initial analysis.

Fig. 7.a and Fig. 7.b show the Accuracy and Log Loss on train and test datasets respectively for all the models. We can clearly say that the Random Forest classifier, XGBoost (Gradient Boosting) classifier and Decision Tree classifiers are performing well on this data set.

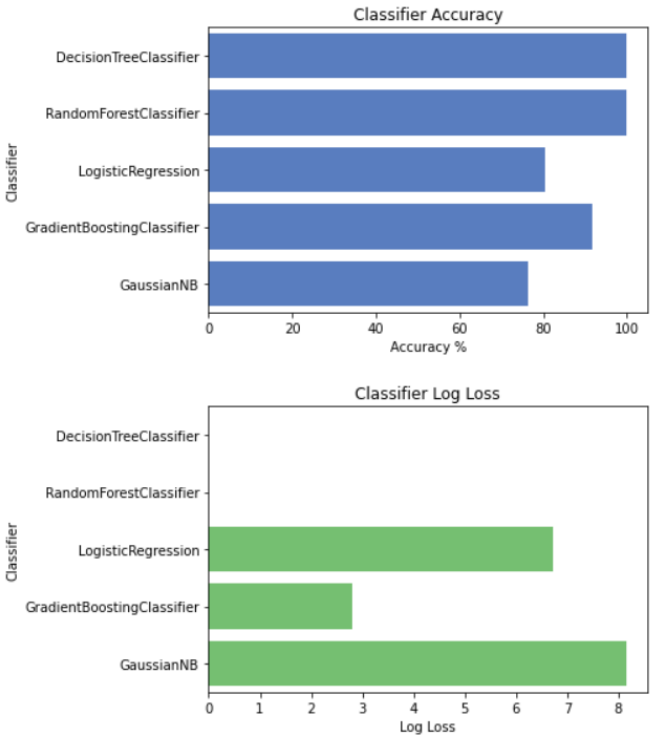


Figure 7.a

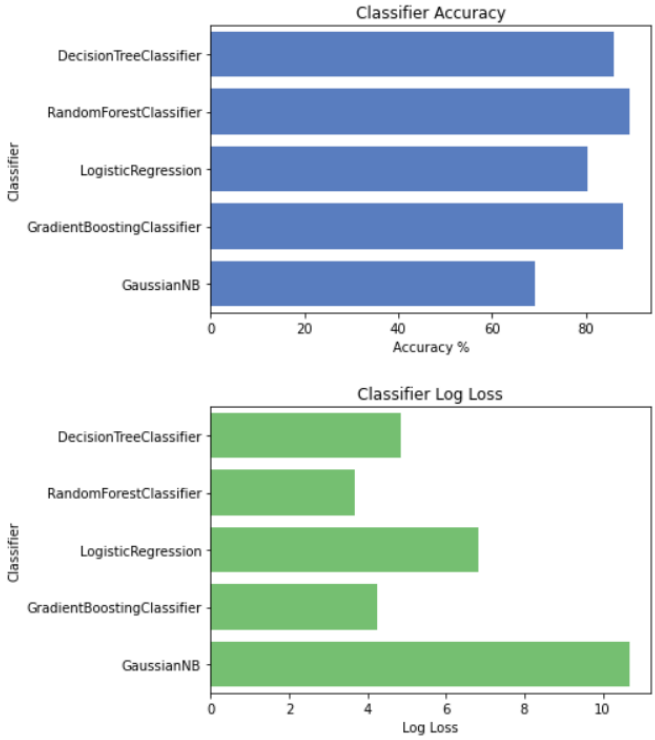


Figure 7.b

EXPERIMENTS AND EVALUATIONS

After the initial analysis, we first oversampled and undersampled the data before performing *grid search.* During grid search, for all the classifiers we considered cross validation to be 5 and the scoring metric to be ROC AUC score. After hyperparameter tuning, below are the final parameters for oversampled data, oversampled data using SMOTE and undersampled data.

Oversampling Hyperparameters

- DecisionTreeClassifier(criterion= 'gini', max_depth=9, min_samples_leaf= 1, min_samples_split= 6)
- RandomForestClassifier(bootstrap = True, max_depth = 20, max_features = 4, min_samples_leaf = 3, min_samples_split = 3, n_estimators = 1500)
- LogisticRegression(C=0.1, class_weight=”balanced”, penalty='l1', solver='liblinear')
- XGBClassifier(learning_rate=0.1, max_depth=9, n_estimators=180, scale_pos_weight=2.7473, scoring=”roc_auc”, class_weight=”balanced”)
- GaussianNB(var_smoothing=1.519911082952933e-08)

SMOTE Hyperparameters

- DecisionTreeClassifier(criterion= 'entropy', max_depth= 9, min_samples_leaf= 4, min_samples_split= 3, class_weight = "balanced")
- RandomForestClassifier(bootstrap = True, max_depth = 20, max_features = 4, min_samples_leaf = 3, min_samples_split = 4, n_estimators = 1450, class_weight = "balanced")
- LogisticRegression(C=0.01, class_weight="balanced", penalty='l1', solver='liblinear')
- XGBClassifier(cv=5,learning_rate=0.05, max_depth=9, n_estimators=140, scoring='roc_auc', seed=42, class_weight="balanced", scale_pos_weight= 2.7473)
- GaussianNB(var_smoothing=1.0)

Undersampling Hyperparameters

- DecisionTreeClassifier(criterion= 'gini', max_depth=9, min_samples_leaf= 1, min_samples_split= 2)
- RandomForestClassifier(bootstrap = True, max_depth = 15, max_features = 3, min_samples_leaf = 3, min_samples_split = 3, n_estimators = 1350)
- LogisticRegression(C=0.01,class_weight={1: 0.6, 0: 0.4},penalty='l1',solver='liblinear')
- XGBClassifier(max_depth=4, n_estimators=180)
- GaussianNB(var_smoothing=0.002848035868435802)

The table shows analysis of all models on the oversampled, SMOTE and undersampled data. XGBoost classifier and Random Forest classifier are performing well on the data. We can see that XGBoost has outperformed Random Forest. For this dataset, handling class imbalance with oversampled data is better than handling it using SMOTE. Undersampling data did not work well.

Classifier	Sampling Type	Dataset	Accuracy	ROC AUC Score	F1 Score	Log Loss
Decision Tree	Oversampling	Train	0.8753	0.8753	0.8768	4.3057
		Test	0.8409	0.8241	0.5489	5.4923
	SMOTE	Train	0.8784	0.8784	0.8787	4.1977
		Test	0.8517	0.8101	0.5513	5.1219
	Undersampling	Train	0.92	0.92	0.92	2.5
		Test	0.46	0.64	0.28	18.3
Random Forest	Oversampling	Train	0.9832	0.9832	0.9834	0.5806
		Test	0.8927	**0.8278**	0.6254	3.7048
	SMOTE	Train	0.9824	0.9824	0.9826	0.6055
		Test	0.8918	0.8092	0.6097	3.7354
	Undersampling	Train	0.95	0.95	0.95	1.6
		Test	0.51	0.68	0.31	16.6
Logistic Regression	Oversampling	Train	0.8057	0.8057	0.8022	6.709
		Test	0.8166	0.8063	0.5106	6.3326
	SMOTE	Train	0.8051	0.8051	0.8047	6.729
		Test	0.8019	0.794	0.4884	6.8406
	Undersampling	Train	0.87	0.87	0.84	4.2
		Test	0.52	0.68	0.31	16.2
XGBoost	Oversampling	Train	0.9617	0.9617	0.9631	1.3202
		Test	0.8671	**0.8501**	0.6005	4.5871
	SMOTE	Train	0.9544	0.9544	0.9561	1.5741
		Test	0.8778	0.8443	0.6124	4.2205
	Undersampling	Train	0.93	0.91	0.92	2.3
		Test	0.48	0.66	0.29	17.9
Naïve Bayes	Oversampling	Train	0.7565	0.7565	0.7526	8.4089
		Test	0.7635	0.7512	0.4286	8.166
	SMOTE	Train	0.774	0.774	0.7777	7.8029
		Test	0.7569	0.7688	0.4378	8.3952
	Undersampling	Train	0.8	0.82	0.75	6.8
		Test	0.61	0.63	0.29	13.3

RESULTS

XGBoost classifier is the best model for this data with an ROC AUC score of **0.8501** on the oversampled data.

Final Model: **XGBClassifier(**
learning_rate=0.1,
max_depth=9,
n_estimators=180,
scale_pos_weight=2.7473,
scoring="roc_auc",
class_weight="balanced")

Fig. 8.a and Fig. 8.b show the ROC AUC score for oversampled train and test data respectively.

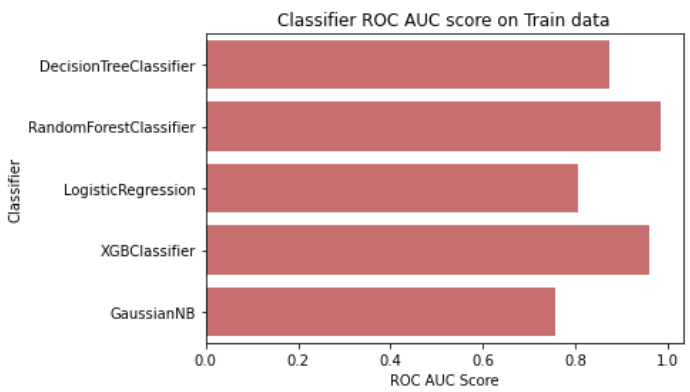


Figure 8.a

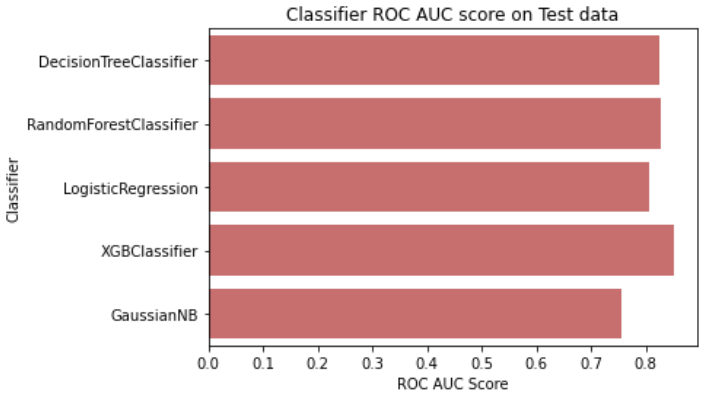


Figure 8.b

Fig. 9.a and Fig. 9.b show the ROC AUC score for SMOTE train and test data respectively.

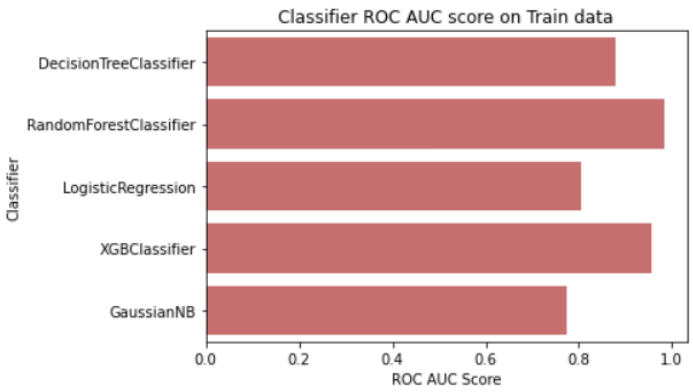


Figure 9.a

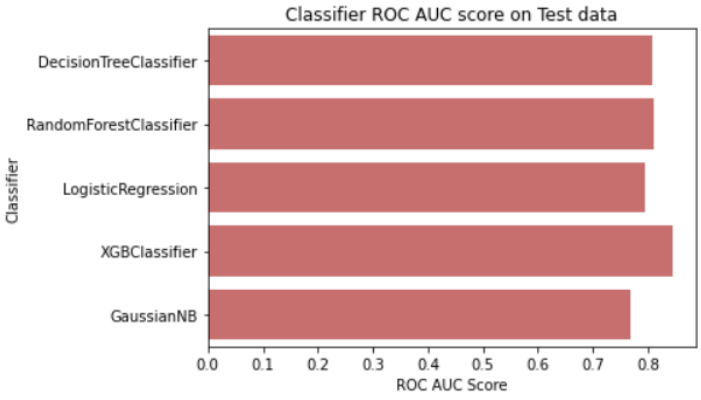


Figure 9.b

Fig. 10.a and Fig. 10.b show the ROC AUC score for undersampled train and test data respectively.

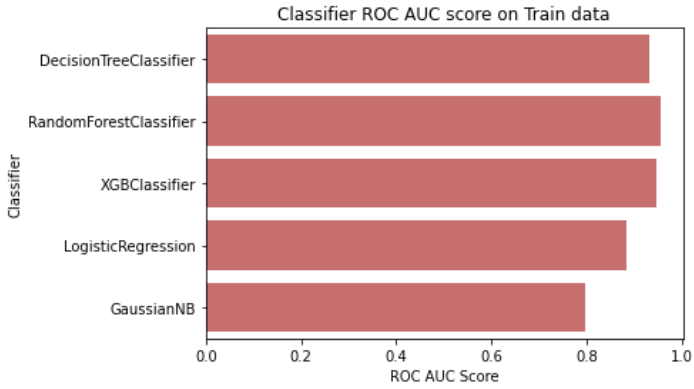


Figure 10.a

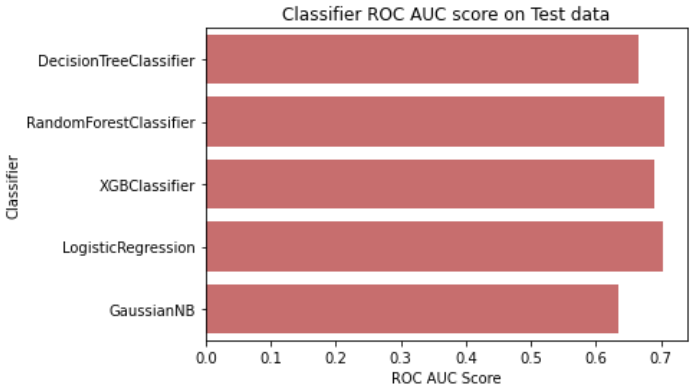
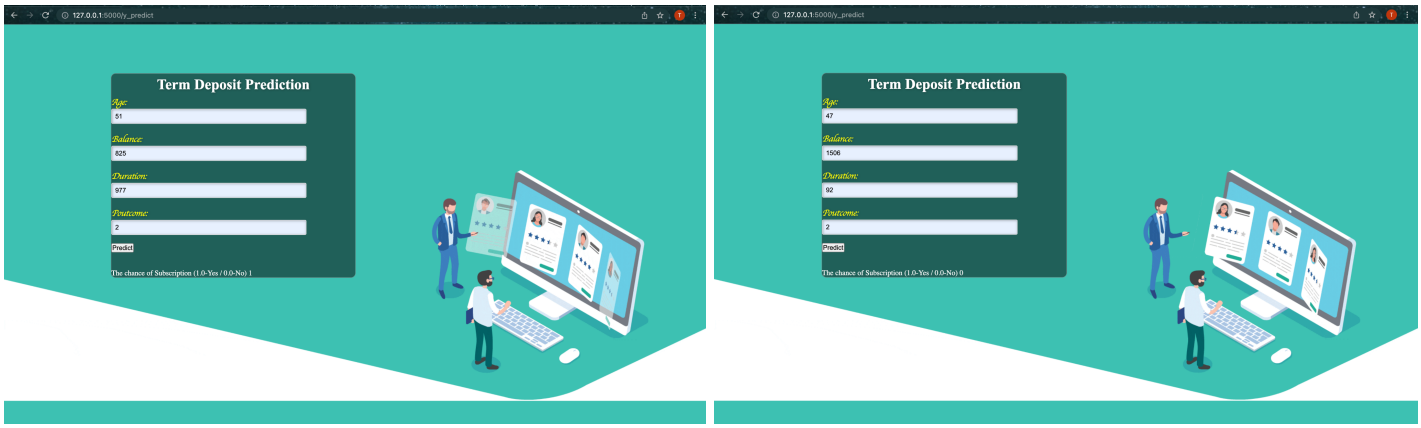


Figure 10.b

Deploying the model to HTML webpage

The best model has been deployed into an html page. The output looks as shown in Fig. 11.



Github Repo Link: *https://github.com/ishal471/Term_Deposit_Subscription_Prediction*

CONCLUSION

This project dealt with highly imbalanced data that had a lot of categorical variables.

Ordinal encoding of the features was essential to work with this data. This was performed after analyzing the data visualizations.

Different approaches for handling imbalanced classes were implemented such as oversampling, SMOTE and undersampling. In all three approaches, 5 classification models were implemented namely, Decision Tree, Random Forest, XGBoost, Logistic Regression and Naive Bayes.

Overall, Random Forest and XGBoost classifiers were working well in all the three approaches. An interesting thing to note is that oversampled data worked better with the models than SMOTE data did.

The best performing model is the XGBoost classifier on the oversampled data.

REFERENCES

1. X. Guo, Y. Yin, C. Dong, G. Yang and G. Zhou, "On the Class Imbalance Problem," 2008 Fourth International Conference on Natural Computation, 2008, pp. 192-201, doi: 10.1109/ICNC.2008.871.
2. A. Gosain and S. Sardana, "Handling class imbalance problem using oversampling techniques: A review," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 79-85, doi: 10.1109/ICACCI.2017.8125820.
3. D. Dhanalakshmi and A. S. Vijendran, "A novel approach in oversampling algorithm for imbalanced data sets in the context of ordinal classification," 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2016, pp. 1-5, doi: 10.1109/ICCIC.2016.7919694.
4. Bradley, A. P. (1997). The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. Pattern Recognition, 30(6), 1145–1159.
5. Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, P. (2000). SMOTE: Synthetic Minority Over-sampling TEchnique. In International Conference of Knowledge Based Computer Systems, pp. 46–57. National Center for Software Technology, Mumbai, India, Allied Press.
6. Cost, S., & Salzberg, S. (1993). A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. Machine Learning, 10(1), 57–78.
7. Małgorzata Bach, Aleksandra Werner, Mateusz Palt,The Proposal of Undersampling Method for Learning from Imbalanced Datasets,Procedia Computer Science,Volume 159,2019,Pages 125-134, ISSN 1877-0509,https://doi.org/10.1016/j.procs.2019.09.167.
8. https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/
9. https://machinelearningmastery.com/xgboost-for-imbalanced-classification/
10. https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/