Comparision of Classification Models for Bank Marketing

Abhishek Nagrecha
Department of Computer Science
Lakehead University
Thunder Bay, Canada
nagrechaa@lakeheadu.ca

Sahand Asri
Department of Computer Science
Lakehead University
Thunder Bay, Canada
sasri@lakeheadu.ca

Prerak Patel
Department of Computer Science
Lakehead University
Thunder Bay, Canada
ppatel74@lakeheadu.ca

I. Introduction

In the field of telemarketing, there are mainly two strategies for promoting the sale of a particular product or service that is mass campaigns and direct marketing [1]. In mass campaigns, the company usually promotes its product or service through medias such as Television ads, Newspaper ads, and Billboards to get the attention of their audience. However, in the past few years, this approach has not been successful. Since they were not able to correctly identify their target audience. On the other hand, direct marketing has shown better results than mass campaigns in terms of success rate.
Based on [2], we know that long term deposit was introduced as a result of the financial crisis and massive competition among the European banks. To sustain in such a competitive environment, the bank needs more customers to subscribe to its long term deposit.
In this project, we aim to discover whether a given client would subscribe to the bank deposit or not, using different classification models. For that, we will make use of popular Data-Mining techniques to prepare our desired data for classifying the users and making predictions according to the result of a classifier, when applied on a dataset containing client details. Our objective is to analyze the performance change for multiple classifiers when applied to the original and pre-processed dataset. The results of these classifiers will help the bank in the future making any decision about their target audience which in a way increase the overall campaign efficiency by using direct marketing.

II. Literature Review

The application of classification models is popular in many areas. Using classification models, we can use prior collected data to predict future goals or decide the direction in which we need to develop. Yet, there is a problem of selecting an appropriate classifier according to the dataset. Many previous works have been done to deal with this problem. We refer to some research papers to get in-depth knowledge about this classification problem. Such as in [3], they have compared the performance of various classification algorithms for predicting the success rate of bank marketing campaigns. Here, they use the rminer package and R tool to compare the performance of three classifiers namely, Decision Tree (DT), Naïve Bayes and Support Vector Machine (SVM). Receiver Operating Characteristic (ROC) and Lift curve analysis were used as performance metrics. A similar kind of comparison was done between Decision Tree, Artificial Neural Network (ANN), Logistic Regression (LR) and Support Vector Machine [2].

In [4], they have selected two data mining models, a Multi-Layer Perceptron (MLP) Neural Network and Ross Quinlan Decision Tree model (C5.0). These classifiers were supposed to identify whether the customer receiving a phone call about a long-term deposit will successfully subscribe or not. Both these data mining models are popularly known for their benchmark accuracy in the field of classification. But, according to their results the C5.0 decision tree model showed slightly more performance than MLP.

We referred a paper where they performed the classification model on the dataset and they reached a conclusion which helped the bank to make some changes in their strategies of marketing a product. In [5], they applied classification and clustering algorithms on the dataset which was prepared on the basis of data collected through telephonic chat between the bank and customer. After evaluating the results of these algorithms, the bank realized the importance of certain factors such as the call duration had a significant impact on the success rate of telemarketing. In addition to that, the number of contacts performed during the marketing campaign should also be controlled to prevent aversion. Moreover, some suggestions were offered to the bank's marketing staff to remain patient while implementing such marketing.

Selecting important features for classification plays a crucial role in improving the overall performance and accuracy of any classifier. Feature selection methods can

be used for this purpose. Information Gain (IG) and Chi-square methods which were used in [6] on a Naïve Bayes classifier had a great impact on the performance of the classifier. Sometime the data in the dataset could be imbalanced that means there exist any two classes in the dataset with a very smaller number of samples in comparison to others. This makes our classification model biased toward the majority class. This kind of problem was encountered in the dataset used in [7] and they used the WEKA, a Machine Learning tool written in Java, to solve this problem. There are many facilities available in WEKA to artificially balance the data. Then they applied multiple classifiers which were Decision Tree (C 4.5), Naïve Bayes, Multilayer Neural Network, Support Vector Machine, Logistic Regression, and Random Forest on both, the balance and imbalanced data, and evaluated their performances.

III. Methodology

A. Dataset

The dataset we are using is called "Bank Marketing Dataset", which is a part of the UCI machine learning repository. The data was extracted in a CSV format. In this dataset, there are 20 attributes, and 41189 instances without any missing values. Originally, the dataset was collected on the basis of 17 marketing campaigns that was conducted in the span of two years [1]. Now based on this information, the Undersampling: Near aim is to predict whether the client will subscribe for bank term deposit or not. Data classification is making use of machine learning techniques to organize the dataset into related subpopulation. This can help us to discover hidden characteristics within the data and identify the hidden catagories that the new data belongs to.

Attribute Details

On the paper which we referred [2], there were initially 150 attributes. Some of these attributes were related to contact information such as duration of the call, type of phone used, and information about the agent who contacted the client. These features were found to be ineffective in this problem. Therefore, a new set of 20 features was collected from the bank database which was more relevant to this problem [2]. The dataset contains general information about the clients of a Portuguese banking institution which are relevant to our goal, such as their age, job type, marital status, education, and previous bank history [2].

Attribute Name	Attribute Description
age	Age in years
job	Type of job
marital	Marital status
education	Education level
default	Has credit in default?
housing	Has housing loan?
loan	Has personal loan?
contact	Contact communication type
month	Last contact month of year
day of week	Last contact day of the month
duration	Last contact duration, in seconds
campaign	number of contacts performed during this campaign and for this client
pdays	Days passed after the client was last contacted from a previous campaign
previous	Number of contacts performed before this campaign and for this client
poutcome	Outcome of the previous marketing campaign
emp.var.rate	Employment variation rate
cons.price.idx	Consumer price index
cons.conf.idx	Consumer confidence index
euribor3m	Euribor 3 month rate
nr.employed	Number of employees
y	Has the client subscribed a term deposit?

B. Data Pre-Processing Methods

1) Missing Values:
In our bank marketing dataset, which we obtained from the UCI machine learning repository, it was stated that the data had no missing values but after data exploration, we realized that there were 7 attributes (6 categorical and 1 numerical) which had missing values. for categorical features, they were represented by 'unknown' and 'nonexistent' while for numeric attribute we only have one such attribute that is 'pdays' where 999 means that the clients were not previously contacted. To deal with those Missing Values we primarily used three Imputation Techniques such as Mean/Median Imputation, Common/Mode Imputation, and Regression Imputation.

Mean/Median Imputation:
In this approach, the missing values (in our case: "unknown" and "nonexistence") are replaced by computing the mean/median of the training data of that certain feature (column). This imputation technique is one of the most basic ones. However, it works fine for numerical features.

Common/Mode Imputation:
In this method, the missing values are substitute with the most-used value of that specific feature. Common Imputation is also one of the fundamental imputations. Yet, in some cases, it gives promising results.

Regression Imputation:
This strategy is the most powerful way to deal with missing values. In this approach, we are trying to guess the value we missed, using classifier and regression models.

For this, we use other features as input variables and give them to our prediction models. Each row without missing value (for that feature) are the training data of our models and the ones with missing value are our test data. Using this strategy, we rely on the predicted values of our models for missing values.
It is noteworthy that, we may have other missing values other than the one we are dealing with. For those missing values we use basic imputation techniques such as Mean Imputation and Common Imputation. Hence, we can use all of the features to train our models.
For instance, feature 'job' has 330 missing valuse. After applying regression imputation, we replaced them with the output of the model. Below you can see the imputed values for the first 15 missed values.

```
['technician' 'blue-collar' 'admin.' 'admin.' 'services' 'admin.' 'admin.'
 'admin.' 'admin.' 'admin.' 'admin.' 'admin.' 'admin.' 'admin.' 'admin.'
 'blue-collar' 'blue-collar' 'services' 'admin.' 'services' 'blue-collar'
 'blue-collar' 'blue-collar' 'blue-collar' 'blue-collar' 'blue-collar'
 'blue-collar' 'technician' 'blue-collar' 'blue-collar' 'technician'
```

Fig. 1. Imputed Values for feature 'job'

2) Feature Selection:
For our dataset, we initially had 41188 rows and 20 distinct attributes. upon data exploration, we realized that a certain attribute named 'duration' which represents the last contact call duration in seconds was not a much of use for our problem as we intend to have a realistic predictive model.however,this attribute highly affects the output target (if duration=0 then y='no').thus we dropped this feature from our dataset, hence, resulting in 19 distinct features. after that, we performed certain pre-processing steps on the dataset to deal with certain problems such as imbalanced dataset problem, missing values, etc.in order to balance the dataset we had to apply the above-mentioned methods which increased the overall number of attributes as we now had 52 attributes. Thus, for selecting the best attributes for our dataset we employed two methods which are univariate selection and correlation filter method.

Univariate Selection:
For Univariate Selection sklearn library has a class named SelectKBest for choosing the top attributes for the given dataset. The sklearn.feature_selection module is used to select the best features [8] and sklearn.feature_selection.chi2 module is used to compute chi-squared stats between every non-negative attribute and class. We have used this score for selecting the best features from the above mentioned 52 features having the most eminent values for the search. after implementing the above-mentioned method, we removed 27 features hence possessing 25 distinct features.

Correlation Filter Methods:
Correlation Asserts how the attributes must be similar to the output and even with one another. Correlation ranges between zero and one. The stronger the relationship, the bigger the number is. We use the heatmap to visualize and identify the features which are correlated to one another. For this purpose, we use the seaborn library. after we applied univariate selection, we had around 25 distinct features but from the heat map, we were able to generalize that certain attributes were highly correlated with one another and therefore one of them could be removed. we, therefore, removed four such correlated attributes which are 'poutcome_nonexistent', 'emp.var.rate', 'euribor3m', and 'poutcome_failure'. after eliminating the above-mentioned attributes we now had 21 distinct features.

3) Normalization:
Normalization is a scaling method used in pre-processing stage to get a better prediction. There are various types of normalizations such as Z-Score, Min-Max, and Decimal Scaling [9]. This dataset has both categorical and numerical attributes. Besides, the numerical attributes vary to a great extent and need to be normalized to get better results. Hence, for the numeric attributes, we are planning to normalize them using the minmaxscalar function from the sklearn library.

MinMaxScalar Function: The minmaxscalar function is responsible to transform the attributes by scaling them to a particular range. This estimator scales and translates each feature individually such that it is in the given range on the training set [10]. For instance, transform our data to the range of zero and one.

Standard Scalar Function: Another alternative normalization method is the standard scalar function which is also implemented in the sklearn library. The standard scalar function Standardizes the features by getting rid of the mean and scaling to unit variance [10]. Since we are planning to implement support vector machine and neural network classifiers, we need to regulate our features as the algorithm makes an assumption that the features are centered around zero and have the same variance as before. We have made an observation through our experiment that after scaling our attributes we obtained better results for almost all the classification models

4) Balancing the Dataset:
After performing some pre-processing steps when we tried to implement some of the classifiers, we got very good accuracy. But after keen observation, we realized that we had a very poor recall score which meant that the model couldn't classify clients who had a subscription for term deposits. that's when we realized that it was clearly an imbalanced dataset problem. Our dataset has 41189

records out of which almost 89% records are of clients without term deposit subscription and we have only 11% records of clients with term deposit subscription. There are mainly two ways to deal with such an imbalanced dataset: Over-Sampling and Under-Sampling.

Synthetic Minority Oversampling Technique:
Synthetic Minority Oversampling Technique (SMOTE) creates synthetic samples of the minority class which is 'yes' in our case, thus making both the classes balanced. We used imblearn.over_sampling module to import the SMOTE method and we get around 36548 instances for both the class types ('yes' and 'no').

NearMiss Undersampling:
NearMiss works with the opposite idea of the above-discussed oversampling method called SMOTE as it lessens the number of instances with the minority label, without any particular order. .We applied NearMiss by using imblearn.under_sampling module to import the NearMiss method and we get around 4640 instances for both the class types('yes' and 'no').

5) Hyperparameter Tuning:
As a part of preprocessing, we have performed hyperparameter tuning on classifiers like Logistic Regression Classifier, Support Vector Machines, Decision Tree, KNN, Random Forest, and Neural Networks.
In opposition to model parameters which are determined during the training phase, hyperparameters are the one which is initiated by the user ahead of training the model to have a command on the implementation aspects of the trained model. The main goal behind such tuning is to achieve the best recall, precision and f1 score for a particular model.
Hyper-parameter tuning is normally done by two conventional methods which are Grid Search and Random Search. The primary method of tuning the hyper-parameters is called Grid Search where we can give a slate of distinctive parameters to the model, out of which it selects the most suitable ones. However, this process is very cumbersome and not very fruitful. On the other hand in Random search, the hyper-parameters are chosen randomly by the classifier itself and hence making the whole process less time consuming and highly beneficial. This method provides us the opportunity to have control over the multiple hyper-parameter functionalities. For our project, we have performed a random search method.
Moreover, there has been a long discussion between grid-search and random search methods as some are of opinion that though grid-search is computationally costly but provides us with best parameters others, however, are of the opinion that random search gives us better results when compared to the previous method.
The best way to determine the hyperparameters is to try as many different combinations as possible for evaluating the performance of the model. For this purpose, we have implemented a 5-fold cross-validation search over the parameter settings. For instance, in the case of the Random Forest, there are hyper-parameters such as the number of decision trees in the forest(n_estimators), criterion(entropy, Gini etc), the maximum depth of the tree(max_depth) , The minimum number of samples required to be at a leaf node(min_samples_leaf), the minimum number of samples required to split an internal node(min_samples_split) [11].

Model	Parameters which are Tunned
LR	Regularization(C), Size of penalty, Max iter and Class weight
DT	Criterion (Gini, entropy), splitter, max_depth, min_samples_split, max_leaf_nodes and min_samples_leaf
RF	Criterion, number of estimators (trees), max_depth, min_samples_split, min_samples_split, bootstrap and maximum number of features used in the model
KNN	n_neighbors, leaf_size, weights, metric
SVM	Kernel, Error Rate (penalty parameter C), degree
MLP	Hidden layer sizes (2 or more), activation, solver, Alpha, learning rate, max_iterations, activation and solver

The hyperparameters are shown on the above table.

6) Categorical to Numerical Feature Conversion:
In our bank-marketing dataset, we have an equal number of both categorical and numerical attributes (10 each). Therefore, for classification purposes we had to convert all the categorical features to numeric ones. for this, we have utilized two methods: Label Encoding, One-Hot Encoding.

Label Encoding:
In Label Encoding, a unique number will be assigned to each unique value in the feature column but a significant problem with this method would be the assumption that the label sizes represent ordinality (i.e. a label of 3 is greater than a label of 2). thus, this method is only suitable for binary attributes. for our dataset, we have four such attributes which are 'housing', 'default', 'contact', and 'loan'. Therefore, we have label-encoded these four attributes.

One-Hot Encoding:
For one hot encoding, a new feature column will be created for each unique value in the feature column. That is value would be 1 if the value was present for that observation and 0 otherwise. this approach is suitable for attributes having multiple values. apart from the above mentioned four attributes, we had six other attributes that we encoded using this method. the features which were encoded were 'job', 'marital', 'education', 'poutcome', 'month' and 'day_of_week'.

C. Classifiers

In this project, we are using several classification models, such as Logistic Regression, Support Vector Machines, Random Forest, and Neural Networks [12].

1) Logistic Regression Classifier:
The Logistic Regression classifier (LR) is a machine learning classification algorithm that predicts the probability of a categorical dependent variable [13]. For logistic regression classifier, we use sklearn.linear_model modul from sklearn library.

2) Decision Tree:
A decision tree is one of the most effective and successful methods for classification and prediction problems. A decision tree is being constructed starting from the root node which represents the label of the class and it has various internal nodes representing the actual trial on the features. [13].
The creation of decision tree classifiers does not demand any domain expertise, and the accuracy of the model is directly proportional to the number of splits. moreover, a decision tree is built on an entire dataset using all the attributes and parameters since our dataset is high-dimensional, decision trees can prove to be a good candidate for its overall classification. Compared to numerous other classification algorithms decision trees is relatively simplistic and effective at the same time for such classification tasks such as ours
A decision tree can prove to be a good candidate for this problem as we are not sure whether the data is linearly separable or not. To add to that for a mixed data (both numeric and categorical) and DT works well. We worked on sklearn DecisionTreeClassifier. In addition, to get a better performance we have performed hyperparameter tuning. For this, we have made use of scikit-learn's RandomizedSearchCV method, in which we defined a grid of hyperparameter range. Then, we randomly sampled from the grid while performing 5-fold cross-validation.

3) Random Forest:
Random Forest (RF) is a learning algorithm for regression and classification problems. This algorithm constructs several decision trees at training time and outputs the class.
When the random forests are matched with decision trees, there is huge variation observed in the way attributes are being split, as the random forest splits multiple attributes instead of single attribute variables at various split points. Thus reduces the overfitting problem. Also, there is a drastic reduction in variance as we are making use of multiple decision trees [14].
We use sklearn.ensemble module from the sklearn library, which includes many ensemble-based methods for classification and RandomForestClassifier is one of them. To get the best accuracy we will perform hyperparameter tuning, for this we intend to use scikit-learn's RandomizedSearchCV method, in which we will define a grid of hyperparameter range. Then, we randomly sample from the grid while performing k-fold cross-validation.

4) K-Nearest Neighbors: K-Nearest Neighbors (KNN) is considered as one of the most fundamental yet essential classification algorithms in Machine Learning. It uses the entire data for classification, which is why its computationally costly and works slow for datasets with many datapoints [15].
KNN is very useful for non-linear datasets, since it does not make any underlying hypotheses about the data. In KNN, predictions are made for a new instance by exploring the entire training set to find the K most similar.
There are several ways to calculate the similarity between two features in KNN. Euclidean Distance and Manhattan Distance are the most popular and common functions to calculate the similarities. It must be noted that, similarity and distance are negatively correlated.
We have used sklearn.neighbors module to import the KNeighborsClassifier. To get the best performance we applied hyperparameter tuning. For this, we have made use of scikit-learn's RandomizedSearchCV method, in which we defined a grid of hyperparameter range. for instance, there is a 'metric' parameter that represents the distance metric that we need to use for building the tree.in our case, we have given two parameters in the grid which are 'euclidean' and 'city block'. Then, we randomly sampled from the grid while performing 5-fold cross-validation.

5) Support Vector Machine:
The Support vector machine (SVM) is used in many real-world applications. For instance, they can be effectively used for text and image classification.

The primary objective of support vector machine (SVM) is to separate the two classes which it gets by a decision boundary. its main aim is to get a decision boundary separating the classes to be as wide as possible. It is believed that this method is one of the best for classification tasks.

SVM has numerous advantages. First of all, it provides a regularization parameter that can be used to prevent overfitting [16]. Secondly, in our dataset, as we are not able to assume that whether the data is linearly separable or not. So, it will be rewarding for us to apply SVM as it can transform the data by using the kernel technique and maximize the overall margin between classes. To elaborate on that, in case of a non-linearly separable data the kernel method in SVM will transform the data into a high dimensional space where it will treat the non-separable data as a separable one [16].
As our dataset is having more than 40k instances, we can't apply SVM on the whole dataset as we know SVM does not work well with such large datasets. Besides, SVM is computationally slow and it is not feasible to run the algorithm on the whole dataset. To solve this problem, we are taking a small portion of our dataset randomly and we are planning to test the model on that. However, there are few limitations while using SVM, like its difficult to determine the relevant hyperparameters which will get us the best accuracy. For this purpose, we will use parameter tuning. To do so, we use the scikit-learn library, to import GridSearchCV from sklearn.grid_search. After this, with the specific classification model, we give a slate of different hyperparameters to be tuned to the above-mentioned method.

6) Neural Network: Artificial Neural Networks (ANN) are used mainly for pattern classification, prediction and optimization purposes. For our dataset, we will implement Multi-Layer Perceptron Neural Network (MLPNN) with backpropagation which is known to be one of the most powerfull neural network architecture [17], [18]. We intend to use the sckit learn library to import the MLPClassifier. It's being argued that compared to the other models like SVM, DT, and LR, MLPClassifier turns out to be the most suitable model for data classification.
An MLP has at least three layers where we might have one or more hidden layers. Apart from the input node, every node is a neuron using a non-linear activation function. as far as our dataset is concerned this method will prove to be beneficial as it can distinguish the data which is not linearly separable through the usage of multiple layers and non-linear activation function.

IV. Justification

In this project, we performed certain preprocessing steps, like missing value imputation, feature selection, feature elimination algorithms and hyper-parameter tuning to find the best features for our dataset.
We applied six classifiers namely, LR, DT, RF, KNN, SVM and MLP in order to classify our problem. One of the reasons for choosing LR is the simplicity of the model. In addition, LR has numerous advantages of fitting models which can be easily understood. Also, it gives us a great prediction in binary classification tasks.
We intend to use RF as one of our classifiers. Since we believe that the random forest could be very helpful as we are not able to assume that whether our classes are linearly separable or not. Additionally, for our dataset half of the features are not even numeric thus it is wise to apply random forest classifier as it works well with such datasets. Moreover, the dataset is also unbalanced and this classifier is known to work well with such a dataset.
SVM is used because this model can give us better accuracy compared to other classifiers and it can handle non-linearly separable data, which can be a case in our dataset. Moreover, we can also overcome the overfitting problem.
Compared to other classifiers, MLP classifiers are one of the most popular and powerful models and using them, we can find non-linear and complex relationships between the features and output along with excellent performance and accuracy [19].
We expect better performance for our classifiers by performing the mentioned pre-processing methods for this dataset.

V. Results

In this project, we use sklearn.feature_selection.chi2 module to find the chi-square scores. Then, we use these chi-square scores to select the top 25 attributes. The table below shows the top five features, we extract using this method.

Top Attribues	Attribute Score
nr.employed	5234.58
poutcome_success	3982.54
euribor3m	3240.97
previous	3090.17
emp.var.rate	2598.25

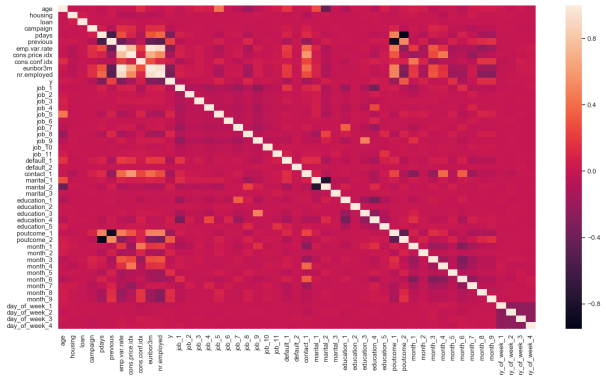

Fig. 2. Heatmap of Correlation Matrix

As shown in Fig. 2, we found out that in our dataset, we have several features that are highly correlated. Hence, we decide to eliminate some of them, since there is no need for all of them. For instance, we detect that nr.employed and emp.var.rate are positively correlated, while previous and pdays are negatively correlated. We need to remove one of the correlated features. In previous papers, they randomly eliminate one of the features. However, in this project, we considered the chi-square score of the attributes before elimination. In other words, the features with lower chi-square scores were discarded. Before feature elimination, we had 25 features. We removed 4 features out of them which were "poutcome_failure", "poutcome_nonexistent", "euribor3m" and "emp.var.rate".

Classifier	Accuracy	Recall	Precision	F1 Score
LR	0.8225	0.82	0.83	0.82
DT	0.7090	0.71	0.71	0.71
RF	0.7819	0.78	0.78	0.78
KNN	0.7808	0.78	0.80	0.78
SVM	0.8268	0.83	0.84	0.83
MLP	0.7349	0.73	0.81	0.82

The above table depicts the performance of the different classifiers without applying any preprocessing steps excluding SMOTE and NearMiss. As you can see in the bellow table, the performance metrics such as accuracy, recall, precision, F1-score greatly improved after applying preprocessing steps including feature selection, missing value imputation, normalization and hyperparameter tuning.

Classifier	Accuracy	Recall	Precision	F1 Score
LR	0.8455	0.85	0.85	0.84
DT	0.8236	0.82	0.83	0.82
RF	0.8920	0.89	0.89	0.89
KNN	0.8617	0.86	0.88	0.86
SVM	0.8606	0.86	0.87	0.86
MLP	0.8793	0.88	0.89	0.88

The bellow table compares the F1-score of all of the classifiers before and after performing preprocessing steps on the balanced dataset. As demonstrated, SVM is our best classifier with an F1-score of 0.83 without implementing preprocessing steps. After applying the preprocessing and tunning the hyperparameters, we observe that RandomForest gives us the best F1-score of 0.89 when the dataset was balanced using SMOTE and normalized by MinMaxScaler.

Classifier	F1-Score (Without Pre)	F1-Score (With Pre)
LR	0.82	0.84
DT	0.71	0.82
RF	0.78	0.89
KNN	0.78	0.86
SVM	0.83	0.86
MLP	0.82	0.88

In this project, to get better performance for our models, we create a set of different values for each hyperparameter of every model. To extract the best value for the hyperparameters, we implement RandomizeSearchCV with 5 Fold Cross-Validation. In the bellow table, you can observe the top values for the hyperparameters of the classifiers.

Classifier	Best Hyperparameters
LR	C: 1000, max_iter: 116, penalty: 12
DT	criterion: 'gini', max_depth: 399, max_leaf_nodes: 16 , min_samples_leaf: 2, min_samples_split: 9, splitter: 'best'
RF	criterion: 'entropy', max_depth: 55, max_depth: 55, min_samples_leaf: 2, min_samples_split: 5, n_estimators: 495
KNN	leaf_size: 2, matric: 'cityblock', n_neighbors: 6, weights: 'distance'
SVM	C: 5, degree: 7, kernel: 'rbf'
MLP	activation: 'relu', alpha: '0.001', hidden_layers_sizes: (40, 40) learning_rate: 'constant', solver: 'lbfgs'

VI. Discussion

we were successful in accomplishing the overall objective of our experiment and after dataset exploration and classifier evaluation we could infer the following:

- Our dataset was profoundly imbalanced. Thus, we had to use SMOTE and NearMiss to balance it out so the models could correctly classify the clients who are going to subscribe for term deposits.
- We had a mixed bag of attributes both in categorical and numerical form. Hence, it needed to be converted to one type. We preferred to convert categorical to numerical using label encoding and one-hot encoding.
- After data exploration, we realized that there were some attributes with missing values, denoted by 'unknown' and 'nonexistent'. Therefore, we decide to impute these attributes for better results.
- We found out that we cannot use an attribute named 'duration' in our prediction models as this attribute highly affects the output target.

- After feature selection, we found out that the number of employees (nr.employed), the outcome of the previous campaign (poutcome), Euribor 3 month rate(euribor3m) and employment variation rate(emp.bar.rate) are our top features that highly affected the classifier's performance.
- There is another attribute named 'pdays' which represents the number of days passed since the client was last contacted from the previous campaign. It appears at the top of the list when we applied univariate selection. However, we realized that it had around 92 percent of missing values. That is why after applying missing value imputation, the attribute was discarded and was no longer an important feature.
- We discovered random forest as our most suitable classifier when we balanced the dataset using SMOTE and normalized it by MinMax Scalar, giving us an accuracy of around 90 percent with F1-score of 0.89.

VII. Conclusion

After performing the above-mentioned experiments we can conclude that our objective to increase the performance of our models was successful. Since, different performance metrics including precision, recall, and F1-scores improved to a great extent by our preprocessing steps. The preprocessing steps include missing value imputation through three distinct ways, converting categorical features to numerical, normalization, feature selection and correlation filter method. In addition to that, hyperparameter tuning with 5-Fold Cross-Validation also helped us immensely to raise the performance of the classification models. The results of these classifiers will help the banks in the future while determining their target audience which in a way will strengthen their direct marketing campaign effectively.

VIII. Future Works

For future work we can use different models and algorithms like Isolation Forest, and One-Class SVM. Moreover, since this is a large dataset it wasn't feasible for us to train certain models like SVM and MLP with all the hyperparameters as it demanded a lot of time. Thus, we can enhance our hyperparameter tuning in future. Besides, we used RandomizedSearchCV for tuning the parameters so GridsearchCV can be applied in the future.
Also, different feature selection methods such as Recursive Feature Selection (RFE), Sequential Forward and Backward Selection can be used.
We may also perform other algorithms such as AdaBoost, Gradient Boosting and Deep Neural Network. Additionally, by applying different clustering methods such as Model-based clustering, Density-based clustering, Fuzzy clustering, we can observe the similarity between clients. Accordingly, we can study the clients' behaviours and responses.

References

[1] S. Moro, R. Laureano, and P. Cortez, "Using data mining for bank direct marketing: An application of the crisp-dm methodology," in Proceedings of European Simulation and Modelling Conference-ESM'2011. EUROSIS-ETI, 2011, pp. 117–121.

[2] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," Decision Support Systems, vol. 62, pp. 22–31, 2014.

[3] S. Moro, P. Cortez, and R. Laureano, "A data mining approach for bank telemarketing using the rminer package and r tool," 2013.

[4] H. A. Elsalamony and A. M. Elsayad, "Bank direct marketing based on neural network and c5. 0 models," International Journal of Engineering and Advanced Technology (IJEAT), vol. 2, no. 6, 2013.

[5] Q. Zhuang, Y. Yao, and O. Liu, "Application of data mining in term deposit marketing," in Proceedings of the International Multi-Conference of Engineers and Computer Scientists, IMECS 2018, March 14-16, 2018, vol. 2, 2018.

[6] T. Parlar and S. K. ACARAVCI, "Using data mining techniques for detecting the important features of the bank direct marketing data," International Journal of Economics and Financial Issues, vol. 7, no. 2, pp. 692–696, 2017.

[7] A. Verma, "Evaluation of classification algorithms with solutions to class imbalance problem on bank marketing dataset using weka," International Research Journal of Engineering and Technology, pp. 54–60, 2019.

[8] A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, A. Mueller, J. Kossaifi, A. Gramfort, B. Thirion, and G. Varoquaux, "Machine learning for neuroimaging with scikit-learn," Frontiers in neuroinformatics, vol. 8, p. 14, 2014.

[9] S. Patro and K. K. Sahu, "Normalization: A preprocessing stage," arXiv preprint arXiv:1503.06462, 2015.

[10] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn," in ICML workshop on AutoML, vol. 9. Citeseer, 2014.

[11] P. Probst, M. N. Wright, and A.-L. Boulesteix, "Hyperparameters and tuning strategies for random forest," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 9, no. 3, p. e1301, 2019.

[12] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," The Mathematical Intelligencer, vol. 27, no. 2, pp. 83–85, 2005.

[13] G. Bonaccorso, Machine learning algorithms. Packt Publishing Ltd, 2017.

[14] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[15] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, 2003, pp. 986–996.

[16] A. Verma, "Study and evaluation of classification algorithms in data mining," International Research Journal of Engineering and Technology (IRJET) Volume, vol. 5, 2018.

[17] T. Munakata, Fundamentals of the new artificial intelligence: neural, evolutionary, fuzzy and more. Springer Science & Business Media, 2008.

[18] B. Chaudhuri and U. Bhattacharya, "Efficient training and improved performance of multilayer perceptron in pattern classification," Neurocomputing, vol. 34, no. 1-4, pp. 11–27, 2000.

[19] S. Haykin, Neural Networks and Learning Machines, 3/E. Pearson Education India, 2010.