Name: Debangshu Bhattacharya

Roll: MDS201910

DMML Assignment 2

<u>EDA , Feature Selection and Preprocessing:</u>

The dataset is highly imbalanced with about 88.7% of the data belonging to 'no' class. From the histogram of the numeric attributes it can be seen that some attributes can be encoded as an ordinal feature. For example, "pdays" is best represented if we encode it as a categorical feature. The value **'999'** means that the client was not previously contacted and hence is encoded 0. The other values are also encoded according to groups with the ordered value being higher if the client was recently contacted. Out of the categorical columns, "**poutcome"** is obviously an ordinal feature as if the outcome of the previous marketing campaign was a *success* for the client then client is more likely to subscribe again than if it was a *failure*. So, we encode the *success* with a high value, *failure* with a low value and *unknown* with a value in between. Similarly, we encode **default** as a person with default credit is less likely to subscribe than a person with no default. So we encode *no* value of **default** with a high value, *yes* value with a low value and *unknown* with a value in between. I did feature selection by taking the top 10 best features using Sklearn's SelectKBest function and chi score as my evaluation function. The rest of the attributes were dropped. The resulting features are:

Numeric columns: ['duration', 'previous', 'emp.var.rate', 'euribor3m', 'nr.employed']
Ordinal columns: ['pdays_cat', 'poutcome', 'default', 'contact']
Nominal columns: ['education']

For Numeric features, I did a Standard Normalization where each feature value is subtracted from its mean and divided by its standard deviation. For ordinal columns, I did a Min Max Normalization where each feature value is subtracted from the minimum feature value and divided by the range of the feature attribute values(max – min). For, nominal features I did a One Hot Encoding of the feature values.

<u>Stratified 10 fold split</u>: Since the data is highly imbalanced, I did a stratified 10 fold split which maintains the data distribution of each split according to the whole dataset class label distribution.

<u>Hyperparameter Tuning</u>: I used the GridSearchCV and searched for the best hyper-parameters for the three classifiers based on "accuracy" scoring. The hyper-parameters found are as follows:

Decision Tree Classifier: criterion='entropy', max_depth = 20, max_leaf_nodes = 30, min_samples_split = 5

Gaussian Naïve Bayes Classifier: var_smoothing = 1 (For further strategies to improve recall, I used the value 0.05 for var_smoothing instead as it gave better results.)

SVM: kernel = 'rbf', C=5, gamma=0.1

Using these hyper-parameters I went on with evaluating cross validation scores. By just fitting the data with these classifiers gave good accuracy but the recall of class "yes" was low. For such an imbalanced data, accuracy is not a good evaluation metric. So we should consider the recall and F1-score of class "yes". The main reason for low recall score is that when the data is as imbalanced as this, the model overfits to the more frequent class. So, I tried a few methods to deal with this.

<u>Using Class Weights</u>: The model penalizes the misclassification of the classes based on the class weights. So by giving a higher weight to the lower frequent class the model would force the model to recognize the class "yes" instances better and hence increase the recall. I used "balanced" class weights which uses the formula:

class_weight[i] = total_numer_of_training_samples/(number_of_classes * number_of_ instances_of_class_i)

Unfortunately sklearn's Naïve Bayes Classifier does not implement class weights. So, I used class weights for Decision Tree Classifier and Support Vector Machine.

<u>Under Sampling from Majority Class</u>: Another technique, I tried is that for each cross validation split I randomly under sampled from majority class in the training data to have a balanced 50-50 split and used that resampled data to fit our model. Each resampled trainin g data consisted of 4176 instances of 'yes' and 4176 instances of 'no'. For Random under sampling from majority class, I used the RandomUnderSampler function from "imblearn" package. This gave way good recall scores for each of the 3 classifiers compared to if we just fitted the data using these models.

<u>Over Sample from the Minority Class and Under Sample from the Majority Class with balanced Class weights of resulting resampled training data</u>:
This strategy gave best recall for "yes" class. First I oversampled the minority class using the Adaptive Synthetic (ADASYN)Sampling approach. This produces synthetically generated data for the minority class based on the nearest neighbours of the minority class examples .Then, I performed a Random Under Sampling from the Majority class. For Decision Trees and SVM , I under sampled with a sampling strategy of 0.7 (Number of samples in minority class= 0.7 times the number of samples in the majority class) whereas for Naïve Bayes Classifier, I undersampled with a sampling strategy of 1 (both classes have equal number of samples). The resultant sampled data distribution is there were 6842 'no' and 4790 'yes' instances for each split for evaluating Decision Tree and SVM. For Naïve Bayes, for each split there were 4790 '

yes' and 4790 'no' samples. The overall mean accuracies, mean recall of 'yes' class and avg F1-score of 'yes' class for the different strategies are shown below for the 3 classifiers:

Decision Tree Classifier

Strategy	Overall Mean Accuracy	Mean Recall of class 'yes'	Mean F1-Score of class 'yes'
Basic	91.52%	59.89%	61.39%
Using Class Weights	81.99%	95.17%	54.42%
Under Sample from majority class	82.50%	95.32%	55.19%
OverSample Minority class + UnderSample majority class + class weight	81.62%	**96.01%**	54.09%

Naïve Bayes Classifier

Strategy	Overall Mean Accuracy	Mean Recall of class 'yes'	Mean F1-Score of class 'yes'
Basic	89.66%	30.38%	39.81%
Using Class Weights	N/A	N/A	N/A
Under Sample from majority class	78.03%	78.04%	44.49%
OverSample Minority class + UnderSample majority class	76.7%	**79.29%**	43.41%

SVM

Strategy	Overall Mean Accuracy	Mean Recall of class 'yes'	Mean F1-Score of class 'yes'
Basic	91.24%	42.17%	52.03%
Using Class Weights	83.67%	92.63%	56.11%
Under Sample from majority class	83.48%	92.46%	55.79%
OverSample Minority class + UnderSample majority class + class weight	81.56%	**94.63%**	53.64%

Conclusion and Remarks:

Since, the dataset is heavily skewed and for this problem, correctly identifying potential subscribers is much more important than just an overall accuracy which is bound to overfit due to the data imbalance. The models with the strategies (oversample from minority class , under sample from majority class and class weights) not only have great recalls for **'yes'** class but also the precision for **'no'** class is great. This means that not only is our model very good at identifying the clients who would actually subscribe to a new term but also if our model predicts that the client won't subscribe to a new term, it is almost definitely true and bank will not need to waste further resources behind the client in convincing him. Our model has more false positives than the ones without the modifications, but this tradeoff is acceptable for this problem given my assumption that a bank would try to make the client subscribe to a new term anyways and the bank would profit much more if they can correctly identify potential subscribers and hence, false negative would hurt more than false positives and hence a higher recall is preferred. Given these assumptions that the model with the best recall is preffered, the best recall scores of class “yes” obtained are **96.01%** for **Decision Tree, 79.29%** for **Gaussian Naïve Bayes and 94.63%** for **SVM classifiers.**