Erblin Marku
Queen Mary University of London
MSc Computer Science
March 2022

Term Deposit prediction and algorithm bias assessment in the Bank Marketing Dataset (S. Moro, 2014)

Abstract

I will perform a Term Deposit prediction to see if a client will subscribe. During this analysis I will try to determine how good is the algorithm for this prediction by manually tuning my model and assess bias presence in the data and algorithm. I will show some future possible work on the algorithm for more accurate result and possible dataset improvements that can help our model predict more correctly. Bias is an ambiguous term and we may have heard a lot the saying – all data is biased. This is of course practically true because the data is collected by humans and also is collected for a certain purpose, which means certain data is gathered (which a human may have seen relevant) but also a large amount of data (which a human may have seen as not relevant) is left behind. We will take in consideration all kind of biases, but we will be more determined on bias which effects our prediction more.

1 Introduction

The public and private sectors are turning to Artificial Intelligence systems and Machine learning more and more to automate their workflow and decision-making techniques. Digitalization of data is disrupting most economic sectors, and this has called for regularization and call for attention when using machine learning algorithms which are not safe against bias. The availability of massive datasets has made it easy for humans to use Ai in their decision-making.

Despite the strong start of these algorithms in the latest years research has shown some troubling examples in which these decisions replicated or even amplified human bias. (CHODOSH, 2018) These examples and others showed that bias is present and cannot be easily reduced, although the best propositions come for a trade-off.

This trade-off has to be made between 'fairness' and 'accuracy' (Nicol Turner Lee, 2019)which practically means for businesses or corporation to take more risk in order to be fair and reduce bias. This cost is not always taken into consideration, but overall is a matter of trust. Other recommendations include the involvement of humans in maintaining the algorithms, government policies, employee algorithm literacy etc. All these come with cost and risk which is part of the trade-off.

2 Dataset Exploration

Our dataset is large and has 4 sets of different types of data collected in it for this purpose. The first 7 are information about the clients. These are *age, job, marital, education, default, housing and lo*an. Age is the only numerical value here, all the others being categorical. Age

is one of the most biased data in AI datasets so we will have a close look at this. The next category is the data from the last marketing campaign which includes: *contact, month, day_of_week and duration.* We can see a lot of missing data here and in the overall data which is labelled as unknown, we will need to check the counts of these fields or see the graphs to determine whether we should treat *unknown* types as a category or should we discard or replace them in some way. The third group is some data about previous campaigns and previous contacts with clients. The labels are: campaign, pdays, previous and poutcome. There is anote for the campaign which says that the if the client has not been contacted, the pday value is set to 999. We will have to look at this since it is a relatively large number according to the other values in this label. The last group is data regarding the economic and social context: emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m and nr.employed. We will see the analysis for more details on this.

Our target is the y label which is a binary value of 'yes' or 'no' indicating the subscription of the client in the term deposit.

```
[7]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

Figure 1 Dataset exploration, after importing the dataset as pandas dataframe we look at an overall information of the dataframe.

We can see we have 20 columns and 41188 entries in our dataset with no null values. Now let's look at our data in more detail, first let's check if our dataset is balanced by calculating the number of yes and no values in our target label.

Lets check if the dataset is balanced

```
[13]: (df['y'].value_counts() / len(df))

[13]: no     0.887346
      yes    0.112654
      Name: y, dtype: float64
```

Figure 2 Result shows that more we have around 88% -'no' and 11%-'yes' values in the target label.

This shows that our dataset is imbalanced and indicates that we should be very careful with the learning techniques. Most of the algorithms assume balanced datasets, so need to choose our model carefully to minimize the effect on the testing set. (Zhaohui Zheng, 2004). There are different ways to handle imbalanced datasets. Random Oversampling and Undersampling are an option, others being threshold method, one-class learning and cost-effective learning. The best option seems to be a hybrid method which combines classifiers. As resulted in (Sotiris Kotsiantis, 2006) the best result performers are hybrid methods which do better than cost-sensitive and random resampling.

In this research I will use CatBoost which is an open source ml algorithm from Yandex. The main feature of CatBoost is that it works well with multiple categories of data. The other feature is that uses gradient boosting algorithm which is widely used in business. This model benefits us because it has robust results, which mean we don't need hyper-parameter tuning and the chances for overfitting are very low. This means we get a more generalized model.

3 Statistical overview

Cleaning is an important part of this process, so lets se more detail from our dataframe.

Lets see the description of the df data

```
[14]: df.describe().T
```

[14]:

	count	mean	std	min	25%	50%	75%	max
age	41188.0	40.024060	10.421250	17.000	32.000	38.000	47.000	98.000
duration	41188.0	258.285010	259.279249	0.000	102.000	180.000	319.000	4918.000
campaign	41188.0	2.567593	2.770014	1.000	1.000	2.000	3.000	56.000
pdays	41188.0	962.475454	186.910907	0.000	999.000	999.000	999.000	999.000
previous	41188.0	0.172963	0.494901	0.000	0.000	0.000	0.000	7.000
emp_var_rate	41188.0	0.081886	1.570960	-3.400	-1.800	1.100	1.400	1.400
cons_price_idx	41188.0	93.575664	0.578840	92.201	93.075	93.749	93.994	94.767
cons_conf_idx	41188.0	-40.502600	4.628198	-50.800	-42.700	-41.800	-36.400	-26.900
euribor3m	41188.0	3.621291	1.734447	0.634	1.344	4.857	4.961	5.045
nr_employed	41188.0	5167.035911	72.251528	4963.600	5099.100	5191.000	5228.100	5228.100

Figure 3 Describe statistics from pandas.

Looking at the statistics we see some problematic values at the std(standard deviation) of the duration and pdays values. This means their data is dispersed far from the mean value indication we may have outliners which we have to remove. Before starting preprocessing lets look at the correlations between labels.

```
[71]: #correlation map to see if we have correlating features
      f,ax = plt.subplots(figsize=(15, 10))
      sns.heatmap(df.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

```
[71]: <AxesSubplot:>
```

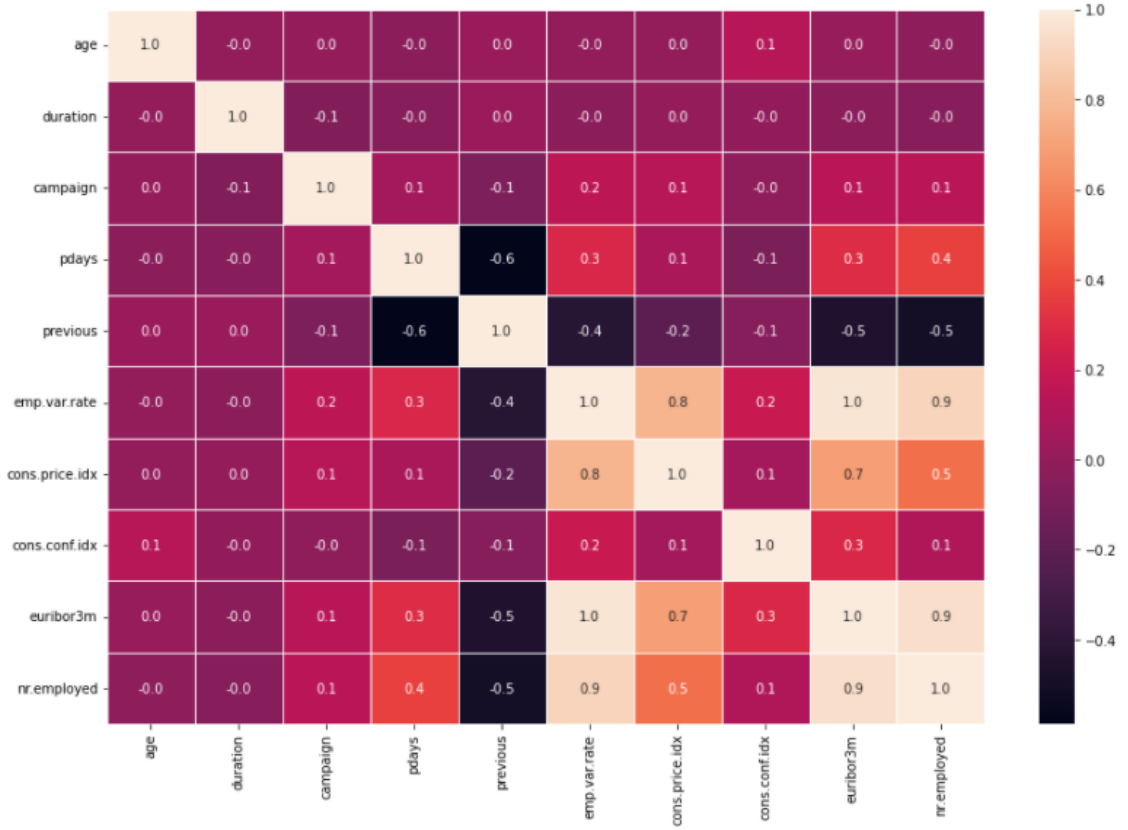

Figure 4 Heatmap of correlations, the lighter the color is and the closer to 1, correlation is greater.

It looks like we have correlation between *euribor3m-emp_var_rate, euribor3m-nr_employed, nr_employed-emp_var_rate and cons_price_rate-emp_var_rate*. We will need to decide which one to drop, for now it looks like *emp_var_state* needs to be dropped and maybe the *euribor3m* but since we have imbalanced data and to avoid cutting useful information we will see their distribution in each of the target class and if they are problematic there we will drop them.

4 Visualization

Now we will build graphs to see what other pre-processing and cleaning we need to do to our dataset before going to training and testing. First, we will plot box graphs to spot outliners and remove them

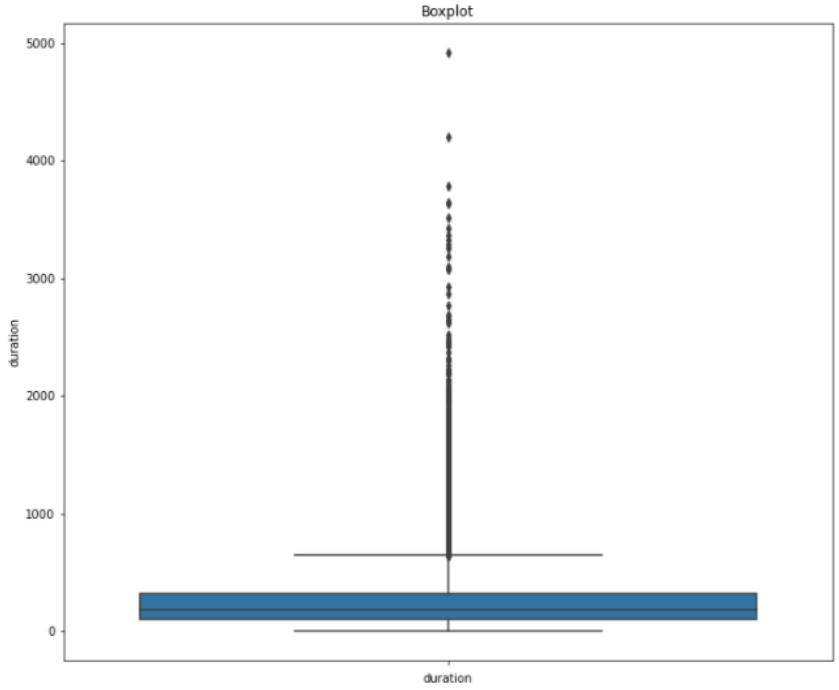

Figure 5 Box Graph for duration label indicates outliners over 3800 value

Other outliners appear in age greater than 90 and campaign over 35 so we will remove those values form our dataset.

We will plot 2 graphs to see the client's education and job according to their age to see if we have a normal distribution of these values through the age range.

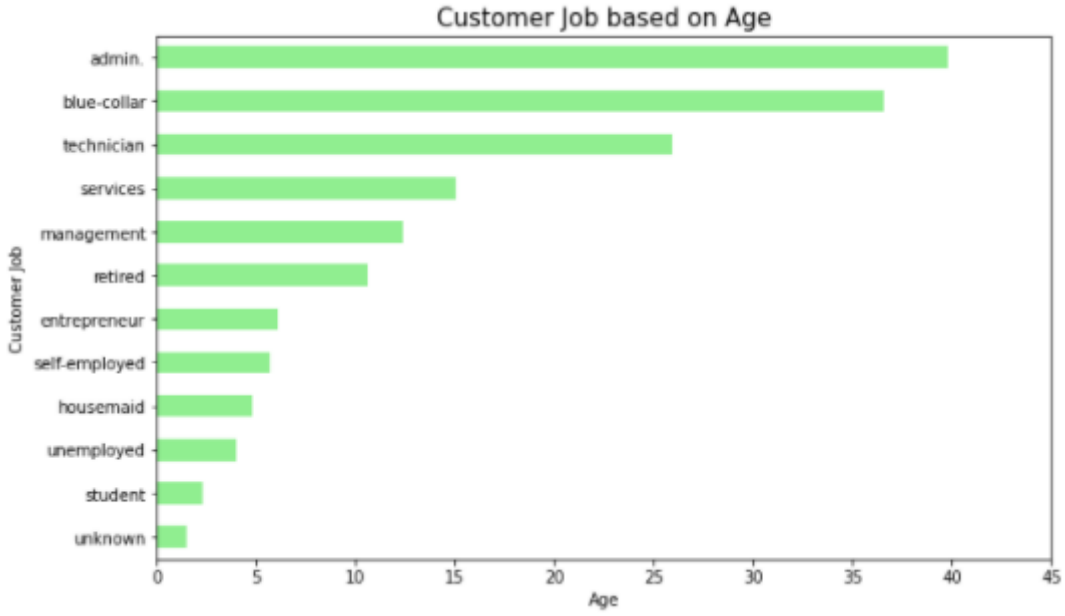

Figure 6 Clients job based on age distribution.

These don't seem to have problems and also the graphs for the other categories of data seem good. The only one problematic seems

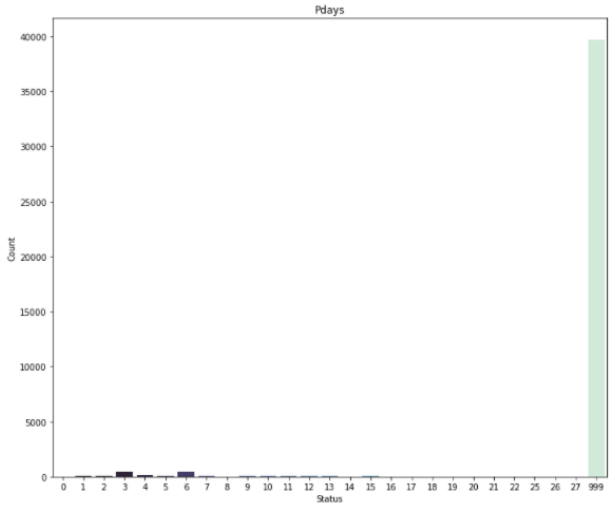

Figure 7 Shows pdays values which have a very large number of 999 values.

We will deal with this by changing the 99 values to -1 to get them closer to the other positive values and look at the graph again.

```
[34]: df['pdays'].value_counts()
```

```
[34]: -1     39664
       3       437
       6       411
       4       118
       9        64
       7        60
       2        60
       12       58
       10       52
       5        46
       13       36
       11       28
       1        26
       15       24
       14       20
       8        18
       0        15
       16       11
       17        8
       18        7
       22        3
       19        3
       21        2
       25        1
       26        1
       27        1
       20        1
      Name: pdays, dtype: int64
```

```
*[35]: #we still have alot of -1 values so it is better to make this a categorical value
       #by looking at the values I will use 3 categories and will define the edges as bins
       bins = [0, 9, 21, 30]
       labels = ['10d', '3w', '>3w']
       df['pdays_grp'] = pd.cut(df['pdays'], bins=bins, labels=labels, include_lowest=False)
```

Figure 8 Count values for pday and change them to categories.

We see that we have an imbalanced dispersion, so what I will do is at least form 3 groups so I can challenge the large values of not called costumers. I form 3 categories: 10days, 3 weeks and greater than 3 weeks.

5 Bias and imbalance check

In order to check imbalance and biased information I will plot all the variables in composite graphs to see their counts and percentages in each of the 'yes' and 'no' categories. Because sometimes the problem is not the imbalanced dataset but rather the imbalanced classes. For this I wrote a function to plot histograms for numerical labels and bar plots for categorical values.

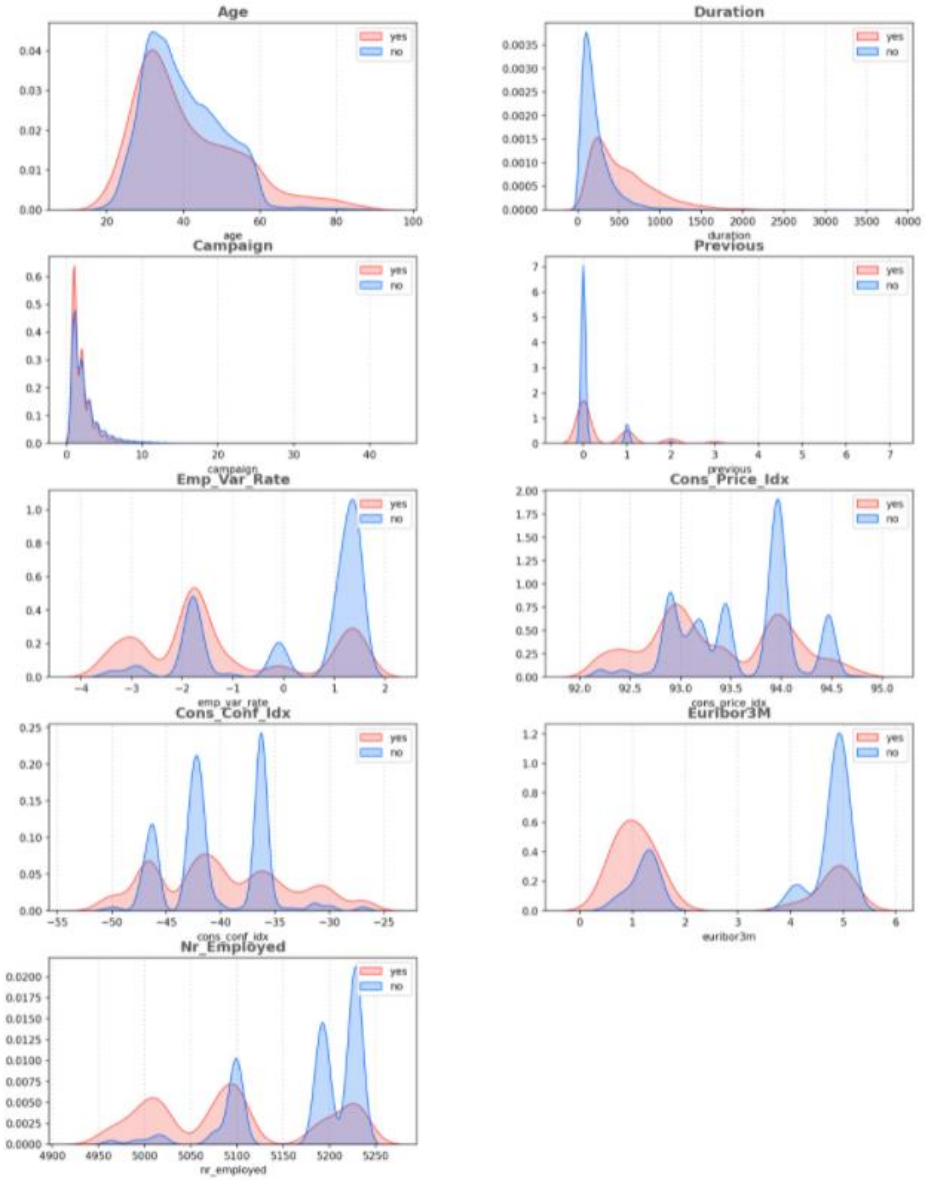

Figure 9 Numerical vaule labels in each of the target categories. Yes is pink, No is in blue.

On the duration graph we see a peak at no, which as indicated before it is a label to be dropped, this graph confirms it and we will delete this column. The other numerical labels have all more peaks and values in the no category except for campaign which seems to perform same for both categories.

Let's see the categorical values in the bar plots.

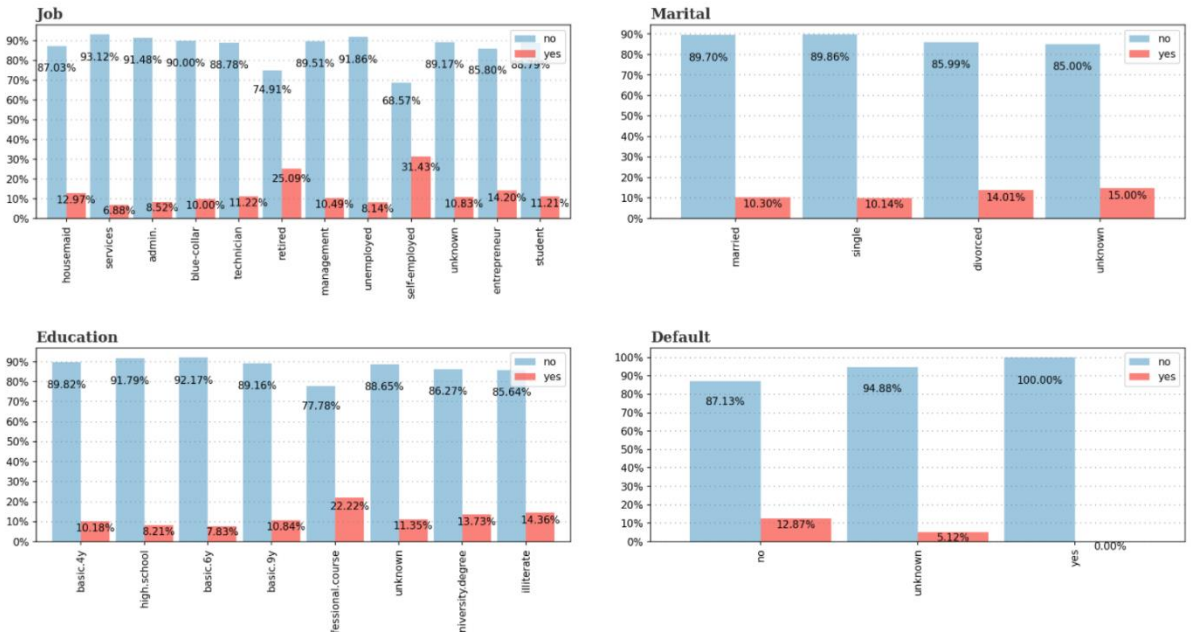

Figure 10 Categorical values plots according to yes-no targets

All the categories seem to have similar distributions in both targets of course being in majority in the 'no' target. One thing we can spot is that on the default label there are no 'yes' target categories.

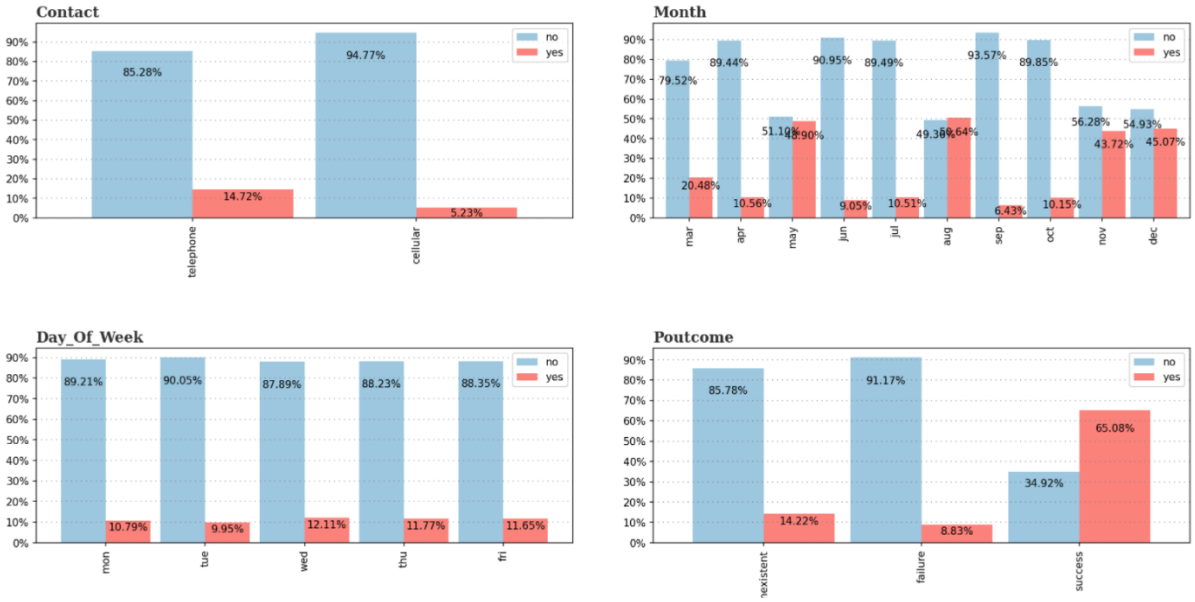

Figure 11 Other categorical plots according to yes-no targets.

Here we can see some values like success in the Poutcome to be more 'yes' targets than 'no', or in some months the 'yes' values overpasses the 'no' one.

By the graphs we can see that most of the labels are found in the 'no' target. This is proof that the algorithm will be biased and has greater probability of giving 'no' as an answer in a test set.

6 **CatBoost Model** (Liudmila Prokhorenkova, 2019)

CatBoost model has a lot of parameters but they are easily manually manipulated. We will start with some and change them to see if we can get better results. The model works on gradient boosted decision trees, so during training the model builds a set of decision trees consecutively. When moving to the next successive tree, the new one is built with reduced loss compared to the previous one. The model has a overfitting detector to stop a tree from being built if it is triggered. CatBoost uses Bootstrap options for regularization and speeding up the building tree process.

Since this model uses categorical values to split and build trees, we will need to convert our data in categories. For this I will use *StandartScaler* from *sklearn* to get all values to categories. Then I build the CatBoost model with initial parameters as in the Figure 12.

[66]:
```
# Initialising Base CatBoost Model
import catboost
# 1. Base Model
cb = catboost.CatBoostClassifier(loss_function='Logloss',
                         eval_metric='Logloss',
                         boosting_type='Ordered', # use permutations
                         random_seed=2405,
                         use_best_model=True,
                         one_hot_max_size = 6,
                         silent=True)



# 2. Fitting the Model
cb.fit(X_train,y_train,cat_features=categorical_features_indices, eval_set=(X_test, y_test))

# 3. Initial Prediction of Results
y_pred = cb.predict(X_test)

# 4. Predicting Probabilites
y_pred_proba = cb.predict_proba(X_test)

# 5. Printing Classification Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.99      0.95     10918
           1       0.70      0.25      0.37      1435

    accuracy                           0.90     12353
   macro avg       0.80      0.62      0.66     12353
weighted avg       0.88      0.90      0.88     12353
```

Figure 12 CatBoost model first try with Logloss function.

As we can see the model performs well with a *f1-score* on *accuracy* 0.90 and *macro avg f1-score* of 0.66. As we can see the classification report proves one more time the existence of algorithm bias. The model performs very well on the '0' label which is the 'no' label but performs poorly with the 'yes' target. Recall value is 0.25 meaning our model fails falsely predicts the yes label as 'no' too many times. Since it performs well on the other values this is a problem that comes from the imbalanced dataset and therefore our algorithm is biased.

To see the model is overfitting or underfitting I will plot the ROC-AUC score and see it is a general model.

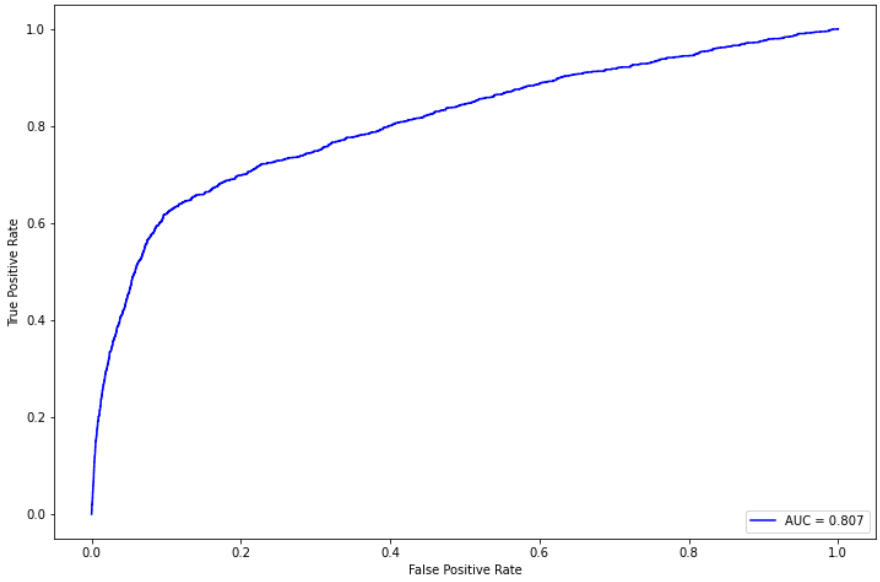

Figure 13 ROC-AUC score

The curve shows that the model has a good performance since the curve is closer to the top-left corner. I try different loss functions for the gradient and different tree and leaf sizes, but the results are not getting better.

7 Further work

For further work and result there are some possibilities. First regarding the model, we can train the model with more different parameters to increase the evaluation metrics, but this won't save us from biased results. More elaborate work can be done with features, there is a way in CatBoost with *gridsearch* to evaluate the features and get the best ones to then use in the model. Due to the time needed for the assignment this was not possible.

Regarding the dataset there are things that can be adjusted, like getting more information for the 'yes' target to even out the imbalance. Other improvement would be replacing client information for loans, housing or default (the ones that are unknown) with real information. This would be more relevant for our target since alongside client information it is directly connected to the client finances and so can have more weight in our learning process.

Other propositions include hybrid methods like training each target class separately and then get samples of each trained set to get the overall model together.

8 Conclusion

We can conclude by the evaluation metrics of our model that there has trace of algorithm bias since our model does well on the class which has more data but poorly performs on the other class. This is mainly due to the imbalance of the dataset because by the graph visualization we see very little class imbalance. The performance of the model is still acceptable, but we must keep in mind that the algorithm tends to give no as an answer.

References

CHODOSH, S. (2018). Courts use algorithms to help determine sentencing, but random people get the same results. *POPULAR SCIENCE*.

Liudmila Prokhorenkova, G. G. (2019). *CatBoost: unbiased boosting with categorical features.* Yandex, Moscow, Russia: Moscow Institute of Physics and Technology, Dolgoprudny, Russia.

Nicol Turner Lee, P. R. (2019). *Algorithmic bias detection and mitigation: Best practices and policies to reduce consumer harms.* Michigan, USA: BROOKINGS.

S. Moro, P. C. (2014). A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems. *Elsevier*, 62:22-31.

Sotiris Kotsiantis, D. k. (2006). Handling imbalanced dataset: A review. *GESTS International Transactions on Computer Science and Engineering* (p. Vol.30). ESDL, University of Patrs, Greece.

Zhaohui Zheng, X. W. (2004). Feature selection for text categorization on imbalanced data. *ACM SIGKDD Explorations Newsletter*, volume 6.

Figures