Machine learning project

Bank Marketing Data Set

Introduction To Machine Learning

Mateusz Dobroń
Suresh Kumar Swetha
Sebastian Alejandro Espin Novillo

Machine learning project

Bank Marketing Data Set

by

Mateusz Dobroń
Suresh Kumar Swetha
Sebastian Alejandro Espin Novillo

Student Name	Student Number
Mateusz Dobroń	313000
Sebastian Alejandro Espin Novillo	309363
Suresh Kumar Swetha	324353

Cover: Canadarm 2 Robotic Arm Grapples SpaceX Dragon by NASA under CC BY-NC 2.0 (Modified)
Style: TU Delft Report Style, with modifications by Daan Zwaneveld
Dataset : [Moro et al., 2014] archive.ics.uci.edu/ml/datasets/Bank%2BMarketing

Contents

List of Figures

1

Introduction

I saw a bank that said '24 Hour Banking', but I don't have that much time[4]

We have decided to use the Bank Marketing Data Set as a personal choice of ours considering that many members like the finance world.

The data is related to direct marketing campaigns via phone calls of a banking institution an approach that is frequently used to connect with certain client segments in order to accomplish a specific goal.

In order to create an effective Machine Learning model we will follow common practices across related industries and explain the steps, tools and algorithms Used.

We present how Logistics regression, decision tree, and K-nearest neighbors are trained in different levels of data training

2

Methodology

Following the SEMMA steps Sample, Explore, Modify, Model, Assess, and considering that this is more of an academic project we will describe the functions and the mathematical principles underlying each of the SEMMA steps in relation to the development of our project.

However, the assessment process will be presented in the next chapter.

2.1. Sample

This step involves selecting a subset of the correct volume from a large dataset; however, because the computer can handle the entire data set with its 45211 entries, we will use the entire data set.

2.2. Explore

Activities are carried out at this phase to comprehend data gaps and connections between variables. In univariate analysis, each variable is examined individually to understand its distribution, whereas in multivariate analysis the relationship between each variable is explored.[3]

- Many of the features that describe a person are of the type categorical such as : job, marital status, and level of education the only numerical value that describes a person is age.
- Features describing financial services that a client use is categorical housing and personal loan
- social and economic context attributes are all numerics: employment variation rate, consumer price index, the consumer confidence index

We will use data visualization to make the data easier to interpret.
Using matplotlib.bar we will display the information, some of the graphs resemble probability distributions.

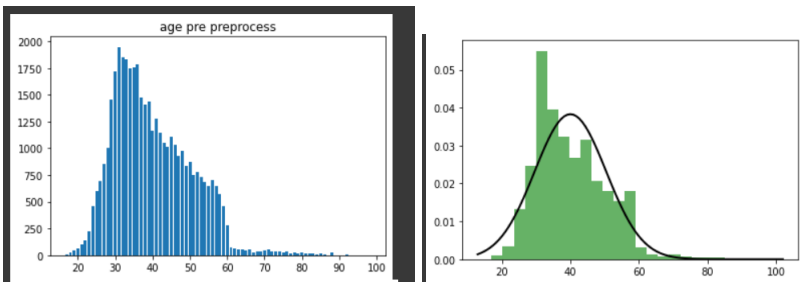

Figure 2.1: age in the data set vs fitted into normal distributions

2.3. Modify

Where necessary, variables are cleaned at this phase. Applying business logic to existing features in accordance with the need results in the creation of newly derived features. If required, variables are converted. A clean dataset that can be used by the ML algorithm to create the model is the phase's output. looks at each variable in isolation to determine its distribution.

2.3.1. Categorical Data

As discussed before many of our attributes are categorical which can not be used in the model if we do not change to numerical before.

We will list the methods that can be possibly used and those who were applied

- **Use a dummy variable**: A Boolean variable that has the value 1 for the existence of a category and 0 to indicate otherwise.[3]
- **Convert to number**: A number is used to represent the text. There is a chance of losing important information with this strategy. Combining groups based on comparable frequency is another option (a new category can be high, medium, low) [3]

We converted to numbers the following default, housing, and loan to the Boolean equivalent 0 or 1 not nullable since these values represent whether or not people had a loan or were in default. Output Y was converted as well, let's recall 'y' means the client subscribed for a term deposit.

We use One Hot Encoder, a function in Scikit-learn, to create the dummy variable for marital, job, education, and poutcome (outcome of the previous marketing campaign).

2.3.2. Quantile preprocessing

A quantile transformation offers a means to change the data distribution of a numerical input variable, which may then be utilized as input to a predictive model, taking into account that numerical values often do not have a standard distribution like normal, gamma, etc. and tend to be extremely skewed.

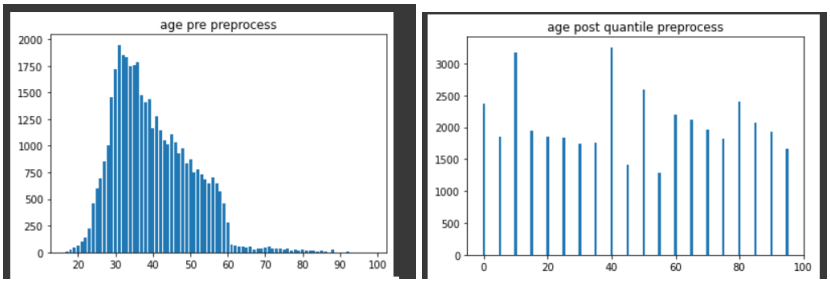

Figure 2.2: Age before and after Quantile preprocessing

2.3.3. Normalize

The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values [1]

We use the method fit_transform(X[, y]) in the StandardScaler class from scikit-learn
The documentation on the matter is:

Standardize features by removing the mean and scaling to unit variance.
The standard score of a sample x is calculated as:

$$z = (x - u)/s$$

where u is the mean of the training samples or zero if with_mean=False, and s is the standard deviation of the training samples or one if with_std=False. [2]

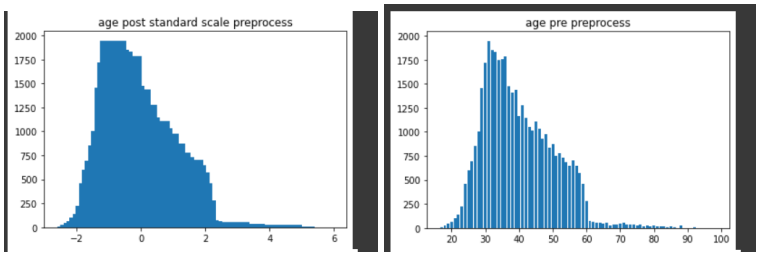

Figure 2.3: Age before and after preprocess

We use standardized data to generate a covariance matrix

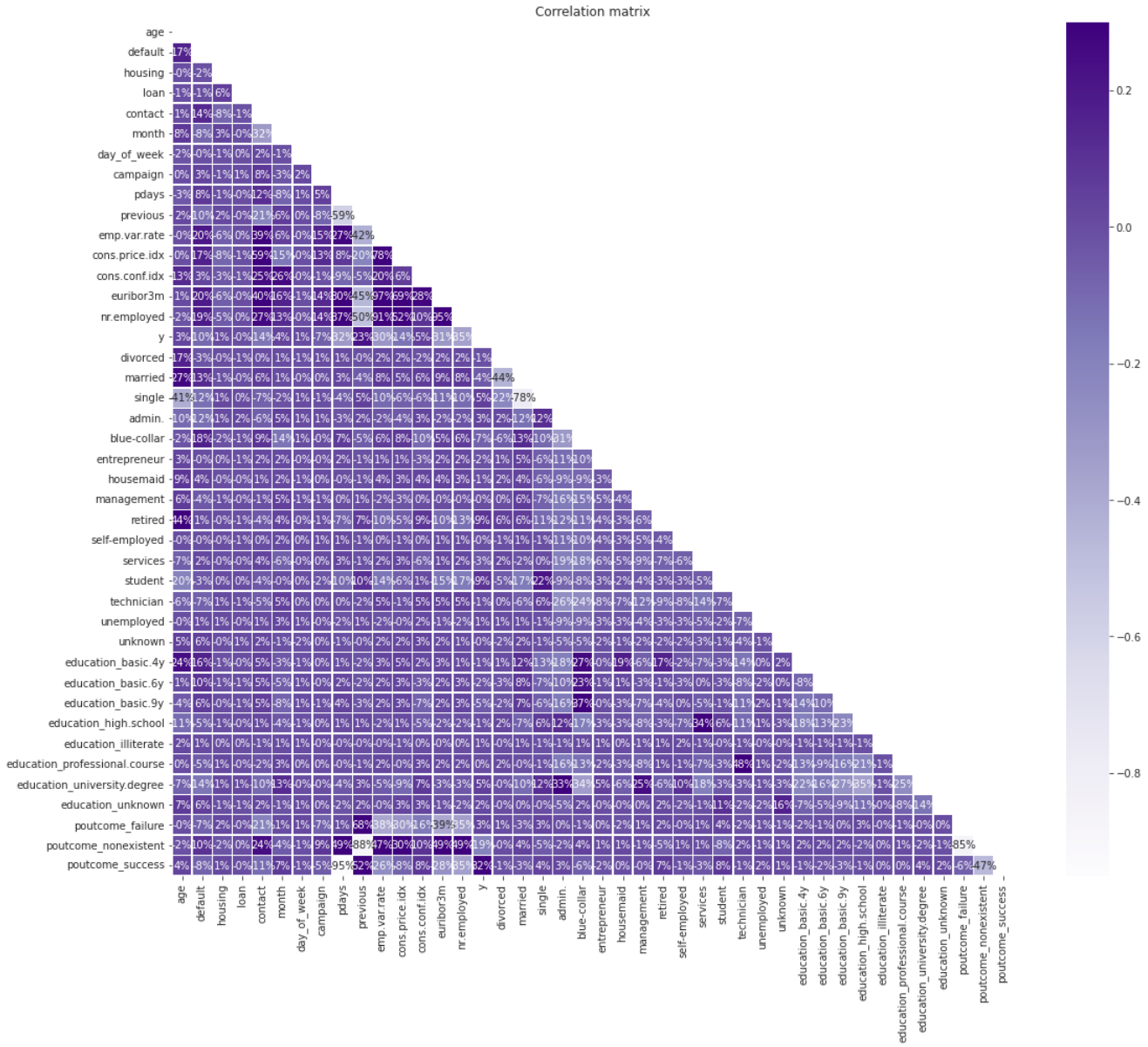

Figure 2.4: Covariance Matrix

2.3.4. Missing Data

This step was omitted. There were no missing values in our dataset.

2.4. Model

In this phase, various modeling or data mining techniques are applied to the preprocessed data, to benchmark their performance against the desired outcome. We will check the performance of different machine learning functions provided by scikit-learn library against our dataframe with numeric values preprocessed in different ways. Namely, we compare dataframe with no changes to numeric variables, with numeric variables being preprocessed using quantization and numeric variables after StandardScaler call. However, all categorical columns are preprocessed using dummy variables in all cases. Such a setting will allow us to determine, how important it is to preprocess the numerical data well and what kind of improvement may be obtained if this step is performed carefully.
Now let us shortly describe the functions/algorithms we use.

2.4.1. Logistic Regression

```
LogisticRegression(C, max_iter)
```

We test this function for different C parameter values to check its relevance to the final result. This parameter denotes “inverse of regularization strength… smaller values specify stronger regularization”. Since we are not able to obtain the result for the default max_iter iterations equal to 100 we set it to 10 000 and thanks to this the solvers finally may converge.

2.4.2. Decision Tree

```
DecisionTreeClassifier(random_state, criterion, min_samples_split)
```

As the documentation says “to obtain a deterministic behavior during fitting, random_state has to be fixed to an integer” [2] and since we want to obtain a deterministic result because our problem is a clustering one we set this parameter to 0. Then we execute this function for different values of criterion and min_samples_split to check if it is possible to improve the algorithm reliability by changing these parameters to non-default ones. criterion is “function to measure the quality of a split. Supported criteria are “Gini” for the Gini impurity and “log_loss” and “entropy” so we execute a decision tree for all of these possibilities. min_samples_split is “the minimum number of samples required to split an internal node” [2] so we also try with different settings of this parameter.

2.4.3. K Neighbors

```
KNeighborsClassifier(n_neighbors, p, metric='minkowski', algorithm=alg)
```

Similarly here we will test a few different parameter settings to check if some improvement is

reachable. n_neighbors is the “number of neighbors to use by default”[2] , we try setting this value to 2, 5 and 10, where 5 is the default one. p is the parameter for the Minkowski metric. Variable metric denotes the metric to use for distance computation. The function is also executed for different algorithms, which in our case are all possible for the function provided by scikit-learn library, namely: ‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’

2.4.4. Multi-layer Perceptron classifier

```
MLPClassifier(random_state, maX_iter, solver)
```

We set in each case the random_state to 1, because as the documentation says “pass an int for reproducible results across multiple functions calls” [2] and we are willing to make comparisons between differently normalize data frames and different solvers. max_iter variable denotes the maximal number of iterations, the solver iterates until convergence or this number of iterations. We want it to converge, but in some cases, it turned out to achieve such a result. This value is set to 1000 as it is greater than the default one, so we give a function higher chances to produce some output. However, making it even larger may result in an unacceptably long compilation time. We execute this function for different solvers, we test all available ones, namely: ‘lbfgs’, ‘sgd’, ‘adam’.

2.5. Assess

2.5.1. Logistics Regression

The logistics Regression model trained on:

- **numerical columns not preprocessed** - The results from this model achieved an accuracy of around 90% on both the training and test sets, with a confusion matrix that shows that most of the predictions were correct, but there were a fair number of false positives and false negatives. The model’s performance did not change significantly when the regularization parameter C was changed.
- **numerical columns preprocessed using quantalization** - The results from this model achieved an accuracy of around 90% on both the training and test sets, with a confusion matrix that shows that most of the predictions were correct, but there were a fair number of false positives and false negatives. The model’s performance improved slightly when the regularization parameter C was increased from 0.01 to 1 or 1000.
- **numerical columns preprocessed using standardization with the function provided by sklearn** - The results from this model achieved an accuracy of around 90% on both the training and test sets, with a confusion matrix that shows that most of the predictions were correct, but there were a fair number of false positives and false negatives. The model’s performance improved slightly when the regularization parameter C was increased from 0.01 to 1 or 1000.

In overall, the results show that preprocessing the numerical columns of the data using quantalization or standardization with the function provided by sklearn improved the model’s performance slightly compared to not preprocessing

2.5.2. Decision Tree

The Decision Tree model trained on:

- **numerical columns not preprocessed** - The accuracy of the model on the training data was high, around 0.995, but the accuracy on the test data was lower, around 0.839. This suggests that the model may have overfitted to the training data, and was not able to generalize well to new, unseen data. The confusion matrix for the test data also shows that the model had a higher number of false positives (1039) compared to true positives (439).
- **numerical columns preprocessed using quantalization** - The accuracy of the model on the training data was slightly lower than in the first set of results, around 0.991, but the accuracy on the test data was also slightly lower, around 0.841. Comparing this to the first set of results, it is clear that preprocessing the numerical columns helped to reduce overfitting, as the accuracy on the test data is closer to the accuracy on the training data.

- **numerical columns preprocessed using standardization with the function provided by sklearn** - The accuracy of the model on the training data and test data was similar to the first set of results, around 0.995 and 0.839 respectively. Comparing this to the second set of results, it is clear that standardizing the numerical columns had a similar effect as quantization, helping to reduce overfitting.

In terms of the effect of the *min_samples_split* and *criterion parameters*, it can be observed that as the *min_samples_split* increases, the accuracy on the training data decreases while the accuracy on the test data increases. This suggests that increasing *min_samples_split* helps to reduce overfitting. In terms of criterion, it appears that both Gini and entropy produce similar results, with Gini.

2.5.3. k-Nearest Neighbors

The k-Nearest Neighbors model trained on:

- **numerical columns not preprocessed** - The accuracy on the training set is 0.913, and the accuracy on the test set is 0.891. The confusion matrix on the training set shows that 25052 true positive and 1281 true negative predictions were made, while 527 false positives and 1971 false negative predictions were made. The confusion matrix on the test set shows that 10627 true positive and 395 true negative predictions were made, while 342 false positives and 993 false negative predictions were made.
- **numerical columns preprocessed using quantalization** - The accuracy on the training set is 0.911, and the accuracy on the test set is 0.887. The confusion matrix on the training set shows that 25059 true positive and 1233 true negative predictions were made, while 520 false positives and 2019 false negative predictions were made. The confusion matrix on the test set shows that 10617 true positive and 351 true negative predictions were made, while 352 false positives and 1037 false negative predictions were made.
- **numerical columns preprocessed using standardization with the function provided by sklearn** - The accuracy on the training set is 0.912, and the accuracy on the test set is 0.893. The confusion matrix on the training set shows that 25064 true positive and 1258 true negative predictions were made, while 515 false positives and 1994 false negative predictions were made. The confusion matrix on the test set shows that 10629 true positive and 409 true negative predictions were made, while 340 false positives and 979 false negative predictions were made.

In summary, K-Nearest Neighbours model which is trained on numerical columns not preprocessed has the highest accuracy on the test set, and the model trained on numerical columns preprocessed using standardization with the function provided by sklearn has the highest accuracy on the training set

2.5.4. Multi-layer Perceptron classifier

The MLPClassifier model trained on:

- **numerical columns not preprocessed** - The accuracy on the training set is 88.35%, and the accuracy on the test set is 88.62%. The confusion matrix shows that 24322 samples were correctly classified as Class 0 and 1153 samples were correctly classified as Class 1, while 1257 samples were misclassified as Class 0 and 2099 samples were misclassified as Class 1.
- **numerical columns preprocessed using quantalization** - The training accuracy is 90.48% and the testing accuracy is 89.6%. The confusion matrix shows that 25270 samples were correctly classified as Class 0 and 819 samples were correctly classified as Class 1, while 309 samples were misclassified as Class 0 and 2433 samples were misclassified as Class 1.
- **numerical columns preprocessed using standardization with the function provided by sklearn** - The training accuracy is 91.98% and the testing accuracy is 88.93%. The confusion matrix shows that 24947 samples were correctly classified as Class 0 and 1573 samples were correctly classified as Class 1, while 632 samples were misclassified as Class 0 and 1679 samples were misclassified as Class 1.

Overall, it can be seen that preprocessing the numerical columns improves the performance of the MLP classifier. The standardization method performed slightly better than quantalization on the test set, but the training accuracy is slightly less. This means that the model may be overfitting to the training data. One way to avoid overfitting is to use a technique like regularization.

3

Result and Discussions

Based on the results from the Logistic Regression, Decision Tree, k-Nearest Neighbors, and MLPClassifier models, it can be seen that all four models performed similarly, with accuracy scores around 90% on both the training and test sets. However, there were some differences in the performance of the models when the numerical columns were preprocessed using quantization or standardization.

For Logistic Regression, preprocessing the numerical columns using quantalization or standardization improved the model's performance slightly. For the Decision Tree, preprocessing the numerical columns using quantization helped to reduce overfitting, as the accuracy on the test data was closer to the accuracy on the training data. However, standardizing the numerical columns had a similar effect as quantization, and did not improve the model's performance.

For k-Nearest Neighbors, the model which was trained on numerical columns not preprocessed had the highest accuracy on the test set, while the model trained on numerical columns preprocessed using standardization with the function provided by sklearn had the highest accuracy on the training set. For MLPClassifier, the model performed similarly regardless of whether the numerical columns were preprocessed or not, with accuracy scores around 88% on both the training and test sets. In terms of overall performance, it appears that the k-Nearest Neighbors model performed the best, with the highest accuracy on the test set. The Decision Tree model had high accuracy on the training set but lower accuracy on the test set, indicating overfitting. The Logistic Regression and MLPClassifier models had similar accuracy scores on both the training and test sets.

4

Conclusion

We must develop knowledge optimization initiatives to leverage our key learnings. $ScottAdams$

In this report, We analyzed data from a Portuguese bank. The goal was to predict if a person will subscribe to a long-term deposit in this institution using attributes like personal information, services the person uses, and financial and economical status that were known before the telemarketing call was executed. We can conclude that our models will help the institution by reducing the numbers of call and obtain a similar success rate by just calling the most likely candidates.

References

[1] Urvashi Jaitley. *Medium*. 2018. URL: medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029 (visited on 01/26/2023).

[2] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[3] Manohar Swamynathan. *Mastering Machine Learning with Python in Six Steps*. 2nd ed. Bangalore, Karnataka, India: APRESS, 2019.

[4] Steven Wright.