Which algorithm best predicts whether a client subscribes to a term deposit?

Hugo Fragata (hugofragata@ua.pt), Marcos Pires (marcosnetopires@ua.pt)

***Abstract*— This paper describes the usage and performance measurements of popular classification algorithms in a Bank Telemarketing Dataset, which aims to classify whether if a client subscribes a term deposit or not. We came to the conclusion that, for this case, a Radial Basis Function Support Vector Machine outperformed the broad set of tested models. Further, we re-tested the model, for a final set of performance metrics. We then compared our results to other papers that use this dataset.**

I. INTRODUCTION

This paper describes the steps that were done in order to test different machine learning algorithms on a dataset. This dataset is related with direct marketing campaigns, through phone calls, of a Portuguese banking company. Our objective is to ascertain which of the chosen algorithms is the most accurate in predicting if a client will subscribe or not to a term deposit.

II. PROJECT FILES

- bank-training_new.csv: training dataset
- bank-crossvalidation_new.csv: crossvalidation dataset
- bank-testing_new.csv: testing dataset
- requirements.txt: packages required to execute the project
- Executables:
  - csv_reformatting.py: for dataset reformatting
  - data_encoding.py: for data encoding
  - csv_reformatting.py: for dataset reformatting
  - data_encoding.py: for data encoding
  - data_separation.py: for separation of data in different categories (training set, testing set, etc.)
  - data_visualization.py: for visualization of data on a browser
  - models_validation.py: script that applies the chosen different algorithms to the dataset and shows the respective accuracies

III. DATASET

The chosen dataset[3] is from February the 14th of 2012 and it contains 45211 instances each with 20 inputs and an outcome, where some values are missing.

A. Attributes related with the bank client data[3]

- age: numeric value
- job: referring the type of job (categorical: ”admin.”, ”blue-collar”, ”entrepreneur”, ”housemaid”, ”management”, ”retired”, ”self-employed”, ”services”, ”student”, ”technician”, ”unemployed”, ”unknown”)
- marital : marital status (categorical: ”divorced”,”married”,”single”,”unknown”; note: ”divorced” means divorced or widowed)
- education (categorical: ”basic.4y”, ”basic.6y”, ”basic.9y”, ”high.school”, ”illiterate”, ”professional.course”, ”university.degree”, ”unknown”)
- default: has credit in default? (categorical: ”no”, ”yes”, ”unknown”)
- housing: has housing loan? (categorical: ”no”, ”yes”, ”unknown”)
- loan: has personal loan? (categorical: ”no”, ”yes”, ”unknown”)

B. Attributes related with the last contact of the current campaign[3]

- contact: contact communication type (categorical: ”cellular”, ”telephone”)
- month: last contact month of year (categorical: ”jan”, ”feb”, ”mar”, ..., ”nov”, ”dec”)
- day_of_week: last contact day of the week (categorical: ”mon”, ”tue”, ”wed”, ”thu”, ”fri”)
- duration: last contact duration, in seconds (numeric).

C. Social and economic context attributes [3]

- emp.var.rate: employment variation rate - quarterly indicator (numeric)
- cons.price.idx: consumer price index - monthly indicator (numeric)
- cons.conf.idx: consumer confidence index - monthly indicator (numeric)
- euribor3m: euribor 3 month rate - daily indicator (numeric)
- nr.employed: number of employees - quarterly indicator (numeric)

D. Other types of attributes included [3]

- campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
- previous: number of contacts performed before this campaign and for this client (numeric)
- outcome: outcome of the previous marketing campaign (categorical: ”failure”, ”nonexistent”, ”success”)

E. Output variable (desired target) [3]

- y: has the client subscribed a term deposit? (binary: ”yes”,”no”)

IV. REQUIRED PACKAGES

- Pandas [11]: for dataset reading, processing and manipulation in memory;
- SciKit-Learn [8]: for machine learning algorithms (Logistic Regression, Random Forest, Decision Trees, IPCA, Data Scaling, K-Nearest Neighbours, Support Vector Machines)
- TenserFlow [12]: for machine learning algorithms (Deep Neural Nets, DNN Linear Mixed)
- MatplotLib [13]: For confusion Matrix Visualization
- Plotly [14]: For dataset visualization

V. DATA PROCESSING

A different set of operations were executed over the raw data, making it easier to work with.

A. Data Reformatting

Because the csv file was not consistent in the formatting of its data we decided to first modify it in a way that it becomes easier to perceive and work with. The referred problem became obvious when different iterations had different attribute dividers, so we changed it so that the only attribute divider possible would be ’,’.

B. Data Encoding

For better performance a dataset should not have attributes which values are labels in String format, instead they should be converted to numeric values. For this effect, the categorical columns of the original dataset have been vectorized, namely the outcome ”y”, ”job”, ”marital”, ”education”, ”default”, ”housing”, ”loan”, ”contact”, ”day”, ”month” and ”poutcome”. [3]

C. Data Separation

A separation of the iterations was made so that we could have a training set, a testing set and a cross validation set. The distribution was roughly of 60%, 20% and 20% respectively.

D. Data Visualization

Allows the visualization of the dataset on a browser according to the duration of the call and the age of the client (X and Y coordinates respectively in the graphic), where the dots represent the outcome depending on their color: blue means ’yes’ and orange means ’no’.

VI. DATASET MODIFICATIONS

Deferent variations of the training and testing sets, obtained through the original dataset, were created for the evaluation of which of them would give us a better accuracy in predicting the outcome.

A. Unchanged dataset

Dataset obtained after running the script to encode data, where a vectorization of categorical columns is done, namely: ”job”, ”marital”, ”education”, ”default”, ”housing”, ”loan”, ”contact”, ”day”, ”month” and ”poutcome”.

B. Minimum and Maximum Scaler [15]

Transforms features by scaling each feature to a given range.

C. Standard Scaler [16]

Standardize features by removing the mean and scaling to unit variance.

D. Incremental Principal Component Analysis (IPCA) [16]

IPCA builds a low-rank approximation for the input data using an amount of memory which is independent of the number of input data samples. It keeps only the most significant singular vectors to project the data to a lower dimensional space.

E. Imputation

Replacement of missing data with substitute values and vectorization of some columns. Here columns like ”marital”, ”job”, ”contact” are dropped for having too many instances where the values are ”unknown”, the attributes ”default”,”housing”,”loan”,”day”,”month”,”poutcome” are replaced by new binary columns and the education column suffers a new vectorization. This vectorization consists in making the possible values for that attribute, ’primary’, ’secondary’ and ’tertiary’, into integers, because there is an implicit sequential increasing order in those values. When the value is ’unknown’ we impute the average of the column.

F. Used Dataset Variations

Only the following variations were used as inputs of the different algorithms chosen for this project.

- Unchanged
- Minimum and Maximum Scaler
- Standard Scaler
- Standard Scaler joined with Incremental IPCA
- Unchanged with imputation
- Minimum and Maximum Scaler with imputation
- Standard Scaler with imputation
- Standard Scaler joined with Incremental IPCA with imputation

VII. ALGORITHMS USED

A. Logistic Regression [18]

Logistic Regression is coming up with a probability function that can give us the probability of a given input being classified as one of the possible outputs.

B. K-Nearest Neighbors [19]

Learning based on the K nearest neighbors of each query point, where K is an integer value specified by the user.

C. Support Vector Machine [20]

Set of supervised learning methods used for classification, regression and outliers detection.

SVMs used:

- Linear
- Polynomial Support Vector Machine
  - 3rd degree
  - 16th degree
- Support Vector Machine with Radial Basis Function Kernel (RBF)
  - 16th degree

D. Decision Tree [21]

A non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

E. Random Forest [22]

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

F. Linear Regression [23]

It is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X.

G. Deep Neural Network [25]

A deep neural network (DNN) is a large collection of simple neural units, with multiple hidden layers of units between the input and output layers and can model complex non-linear relationships.

H. Mix between DNN and Linear [24]

Linear and DNN joined training models.

VIII. VALIDATION OF MODELS

After executing the main script we obtained the following percentages regarding the precision of result prediction.

A. Logistic Regression (of scikit-learn)

Logistic Regression	0.904966887417
MinMax-Scaled	0.904194260486
Standard-Scaled	0.906732891832
Standard-Scaled with IPCA	0.886313465784
Imputed Logistic Regression	0.902538631347
Imputed MinMax-Scaled	0.901103752759
Imputed Standard-Scaled	0.903642384106
Imputed Standard-Scaled with IPCA	0.886534216336

B. K-Nearest Neighbors (of scikit-learn)

K-Nearest Neighbors	0.884437086093
MinMax Scaled	0.897240618102
Standard Scaled	0.896026490066
Standard Scaled IPCA	0.855298013245
Imputed K Nearest Neighbors	0.884437086093
Imputed MinMax Scaled	0.896026490066
Imputed Standard Scaled	0.898454746137
Imputed Standard Scaled with IPCA	0.771854304636

C. Linear Support Vector Machine (LSVM) (of scikit-learn)

Linear Support Vector Machine	0.216445916115
MinMax Scaled	0.901434878587
Standard Scaled	0.900662251656
Standard Scaled with IPCA	0.886754966887
Imputed LSVM	0.844370860927
Imputed MinMax Scaled	0.899337748344
Imputed Standard Scaled	0.899337748344
Imputed Standard Scaled with IPCA	0.886754966887

D. Polynomial Support Vector Machine of 3rd degree (3Poly SVM) (of scikit-learn)

3Poly SVM	0.13233995585
MinMax Scaled	0.886754966887
Standard Scaled	0.905960264901
Standard Scaled with IPCA	0.886754966887
Imputed 3Poly SVM	0.871302428256
Imputed MinMax Scaled	0.886754966887
Imputed Standard Scaled	0.904856512141
Imputed Standard Scaled with IPCA	0.88642384106

E. Polynomial Support Vector Machine of 16th degree (16Poly SVM) (of scikit-learn)

16Poly SVM	0.113245033113
MinMax Scaled	0.886754966887
Standard Scaled	0.888741721854
Standard Scaled with IPCA	0.233554083885
Imputed 16Poly SVM	0.113245033113
Imputed MinMax Scaled	0.886754966887
Imputed Standard Scaled	0.88940397351
Imputed Standard Scaled with IPCA	0.887417218543

F. RBF Support Vector Machine of 16th degree (16RBF SVM) (of scikit-learn)

16RBF SVM	0.886865342163
MinMax Scaled	0.899337748344
Standard Scaled	0.906622516556
Standard Scaled with IPCA	0.888631346578
Imputed 16RBF SVM	0.886865342163
Imputed MinMax Scaled	0.899337748344
Imputed Standard Scaled	0.906843267108
Imputed Standard Scaled with IPCA	0.887306843267

G. Decision Tree (of scikit-learn)

Decision Tree	0.87770419426
MinMax Scaled	0.825717439294
Standard Scaled	0.87770419426
Standard Scaled with IPCA	0.72582781457
Imputed Decision Tree	0.875496688742
Imputed MinMax Scaled	0.806181015453
Imputed Standard Scaled	0.869426048565
Imputed Standard Scaled with IPCA	0.651545253863

H. Random Forest (of scikit-learn)

Random Forest	0.903421633554
MinMax Scaled	0.890838852097
Standard Scaled	0.902759381898
Standard Scaled with IPCA	0.854415011038
Imputed Random Forest	0.900993377483
Imputed MinMax Scaled	0.870529801325
Imputed Standard Scaled	0.902538631347
Imputed Standard Scaled with IPCA	0.827814569536

I. Linear Regression (of TensorFlow)

Linear Regression	0.845916
MinMax Scaled	0.886755
Standard Scaled	0.886313
Standard Scaled with IPCA	0.879249
Imputed Linear	0.845143
Imputed MinMax Scaled	0.886755
Imputed Standard Scaled	0.884658
Imputed Standard Scaled with IPCA	0.88543

J. Deep Neural Network (of TensorFlow)

Deep Neural	0.339404
MinMax Scaled	0.886755
Standard Scaled	0.886424
Standard Scaled with IPCA	0.886755
Imputed Deep Neural	0.840177
Imputed MinMax Scaled	0.886755
Imputed Standard Scaled	0.886755
Imputed Standard Scaled with IPCA	0.886755

K. Deep Neural Network Mixed with Linear Regression (of TensorFlow)

Deep Mixed Linear	0.845916
MinMax Scaled	0.886755
Standard Scaled	0.886313
Standard Scaled with IPCA	0.879249
Imputed Deep Mixed Linear	0.845143
Imputed MinMax Scaled	0.886755
Imputed Standard Scaled	0.884658
Imputed Standard Scaled with IPCA	0.88543

L. Best Prediction

The best accuracy obtained was of 90.7% with the Support Vector Machine algorithm of 16th degree with Radial Basis Function Kernel (RBF) and with the scaled imputed dataset.

Fig. 1. Confusion matrix, no normalization

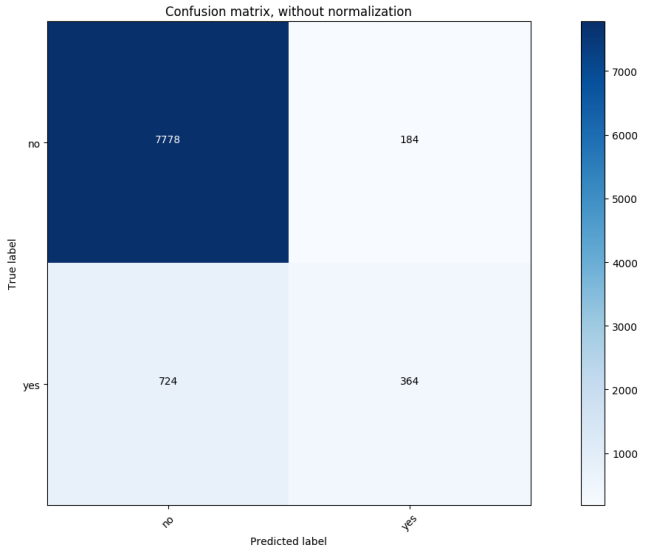
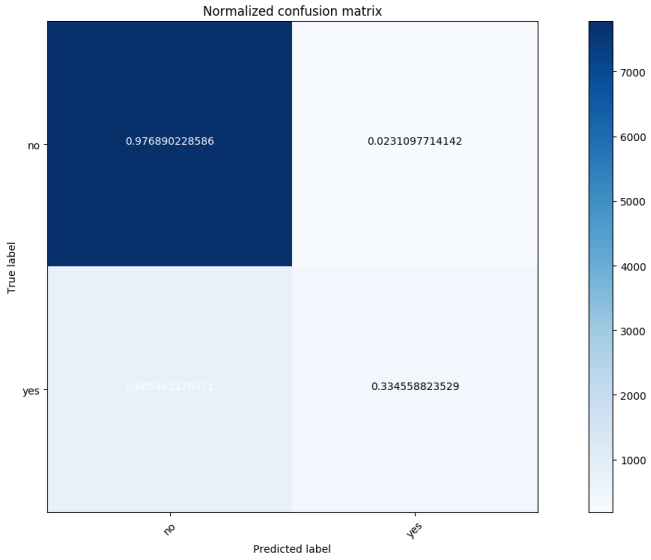


Fig. 2. Confusion matrix, with normalization



IX. FINAL PERFORMANCE MEASUREMENTS

After picking the best model from our set of cross-validated models, which was the RBF 16th Degree SVM with Imputed and Scaled Data we ran a series of performance measures on the Testing subdataset.

We test the number of True Positives (TP), True Negatives (TN) , False Positives (FP) and False Negatives (FN), we can compute different performance measurements associated to a Confusion Matrix [10], as we can see in figure 1 and the normalized figure 2.

Metric	Formula	Value
True Positive Rate	$\frac{TP}{(TP+FN)}$	0.335
True Negative Rate	$\frac{TN}{(TN+FP)}$	0.977
Positive Predictive Value	$\frac{TP}{(TP+FP)}$	0.664
Negative Predictive Value	$\frac{TN}{(TN+FN)}$	0.915
Miss Rate	$\frac{FN}{(FN+TP)}$	0.665
Accuracy	$\frac{TN+TP}{(FN+FP)}$	0.900

X. CONCLUSIONS

In the analysis of our results and the results of similar papers, we found out we outperformed the SVM Model in *A data-driven approach to predict the success of bank telemarketing by Sergio Moro et al.* [1] which achieved 89.6% Accuracy using 70% of the data for training. In contrast, we achieved a SVM Model at 90.7% accuracy in crossvalidation and 90.0% accuracy in testing using only 60% of our data for training the model.

XI. FUTURE WORK

In contrast to the other similar papers we found out we underperformed in our Neural Network. Future work should be focusing on why this happened and what should be done to improve the outcome of the performance metrics.

REFERENCES

[1] http://media.salford-systems.com/video/tutorial/2015/targeted_marketing.pdf
[2] https://repository.asu.edu/attachments/170603/content/Ejaz_asu_0010N_15774.pdf
[3] https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
[4] https://bigml.com/user/totyb/gallery/dataset/5092da63035d075cd100006c
[5] https://www2.1010data.com/documentationcenter/prod/Tutorials/MachineLearningExamples/BankMarketingDataSet.html
[6] https://arxiv.org/ftp/arxiv/papers/1503/1503.04344.pdf
[7] http://www.columbia.edu/~jc4133/ADA-Project.pdf
[8] http://scikit-learn.org/stable/index.html
[9] http://ceur-ws.org/Vol-710/paper37.pdf
[10] http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
[11] http://pandas.pydata.org/
[12] https://www.tensorflow.org/
[13] https://matplotlib.org/
[14] https://plot.ly/
[15] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
[16] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[17] http://scikit-learn.org/stable/auto_examples/decomposition/plot_incremental_pca.html
[18] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[19] http://scikit-learn.org/stable/modules/neighbors.html
[20] http://scikit-learn.org/stable/modules/svm.html
[21] http://scikit-learn.org/stable/modules/tree.html
[22] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[23] https://www.tensorflow.org/tutorials/wide
[24] https://www.tensorflow.org/tutorials/wide_and_deep
[25] https://www.tensorflow.org/versions/master/api_docs/python/tf/contrib/learn/DNNEstimator

Fig. 3. Duration of Call vs Age of Client, Subscription of term deposit

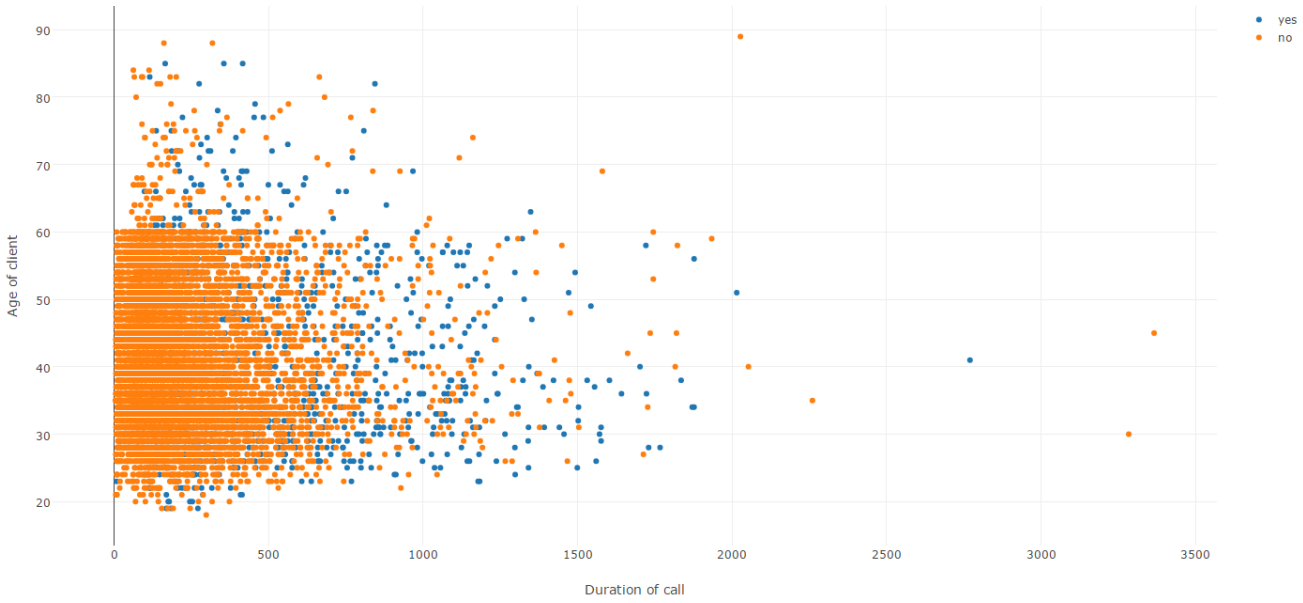


Fig. 4. Most Common Class Analysis