Exploratory data analysis of students application dataset with benchmark of two selected machine learning algorithms

Karol Jankowski
Faculty of Electrical Engineering
Warsaw University of Technology
Warsaw, Poland
01036596@pw.edu.pl

Kamil Rajewski
Faculty of Electrical Engineering
Warsaw University of Technology
Warsaw, Poland
01114730@pw.edu.pl

This article presents students application dataset EDA and machine learning algorithms evaluation used for Dean's decision prediction. Dataset features and pretraining data transformations are shortly described.

Keywords: machine learning, classification, EDA, SVM, Random Forest, Python, pandas, scikit-learn

I. Introduction

Data science uses data analysis, machine learning algorithms and business knowledge for automatization of recurrent tasks and successfully reduces human work time and duties. One of many applications of data science can be Dean's office work improvement by automatic student's application decision prediction.

The aim of this paper is to prepare exploratory data analysis of students applications which helps with features selection and gives better data understanding. EDA is a first step before machine learning algorithms usage for Dean's decision prediction. After all, various ML algorithms are evaluated in order to finding the best for this situation. Algorithms learning time comparison is presented.

II. Dataset Overview

Provided data is an exported real data from ISOD system. Data contains information about students applications, with plenty of student individual features, Dean's office work data and Dean's decision to application. Row dataset contains 43 columns and 5883 applications samples.

III. Features Selection

Becoming familiar with the received data is a first step of EDA. After seeing various columns values distributions feature selection was done. Some of the features in the dataset at first glance could not be used for two main reasons: too large data gaps and lack of connection to a given problem.

Values gaps in some columns were too big to replace missing values with mean or median. The features dropped at this stage with to many NaN values: "Stopień zaawansowania pracy", "Semestr rejestracji", "Informacje o pracach dyplomowych", "Brakujące przedmioty".

Some features presented coded personal data or data not connected with problem and because of that some columns were dropped. Dropped columns: "Student", "Osoba przypisana", "Osoba przyjmująca", "Numer dokumentu", "Data złożenia", "Data zamknięcia", "Liczba prac dyplomowych", "Język wykładowy", "Kierunek", „Semestr aktualny".

The next features, which were considered to drop, were columns with single words like: regulamin, formalny, postęp, rodzin, famil , zdrowie , bradzo proszę, losow, praca, obiecuję, zobowiązuje, promise. Algorithms were trained with and without that columns and results were slightly better with that columns, so we decided to keep it.

Various correlation coefficients where checked to find correlated columns, but even if correlation occurred there was not strict division by "Status" values. In most cases data correlation was a effect of logical after-effects of studies length. Fig. 2. presents relationship between two columns with high correlation coefficient (> 0.8) but does not give addional information about "Status" value.

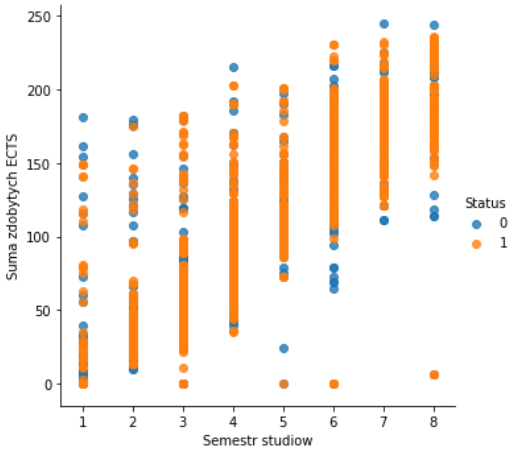

Fig. 2. Relationship between 'Semestr studiow' and 'Suma zdobytych punktow' (in hue of 'Status')

IV. Data Cleaning

In next step data cleaning and categorical values with numerical replacing was made. Columns with just two values were transformed to 0 or 1 values by hand – e.g. 'Tryb studiow', 'Typ studiow'. If column had more than two categorical values, column was transformed by one-hot encoding – e.g. 'Rodzaj wniosku' with 7 categorical values.

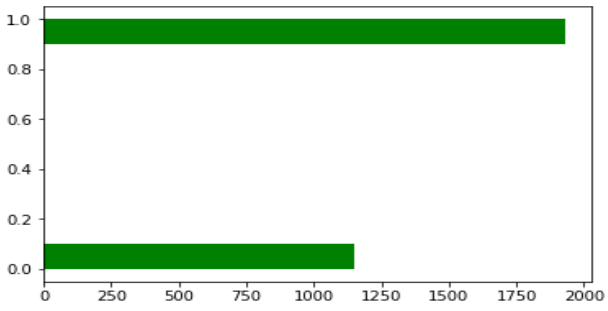

Fig. 1. Dean's decision histogram (1 – Positive , 0 - Negative)

Prediction columns- "Status"- had 9 unique possible values. Samples with 'Status' value suggesting not proceeded process were dropped – "Status" values: "Waiting

for the Dean's decision", "Composite – waiting for a reception at the Dean's Office", "Closed – other", "Canceled – deleted". Afterwards, status categories were split into positive (value 1) and negative (value 0). Fig. 1. presents histogram of Dean decisions after transformations.

Finally, dataset with selected features contains complete 3077 samples with no missing values, 34 features columns and 1 columns of explained veriables.

V. ALGORTIHMS OVERVIEW

As the task was to predict Dean's decision, the problem have to be solved as a classification task with two classes - applications approved and applications rejected. As in all classification tasks many machine learning algorithms can be used, such as:

- Naive Bayes Classifier – simple probabilistic classifier, uses Bayes theorem, naive independence assumption between the features,
- KNN Classifier - uses sample features neighborhood to find the most similar samples,
- Logistic Regression – uses Sigmoid function to estimate features importance,
- SVM Classifier – finds a hyperplane in N dimensional space, that can separate samples into proper classes,
- Random Forest Classifier – decision trees and ensemble learning combining plenty of decision trees to get better results than DT

VI. DESCRIPTION OF ALGORITHMS FOR THE EVALUATION

A. *SVM (Support Vector Machine)*

This is a linear model for classification and regression problems. In this case classification was needed. It can solve linear and non-linear problems. The aim of the algorithm, is to create a line or a hyperplane which separates the data into classes.

At first approximation what SVMs do is to find a separating line(or hyperplane) between data of two classes. SVM is an algorithm that takes the data as an input and outputs a line that separates those classes if possible. According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane. Thus SVM tries to make a decision boundary in such a way that the separation between the two classes is as wide as possible.

To fully understand the operation of SVM, it is necessary to understand what is hyperplane. A hyperplane in an n-dimensional Euclidean space is a flat, n-1 dimensional subset of that space that divides the space into two disconnected parts. For example let's assume a line to be our one dimensional Euclidean space (i.e. let's say our datasets lie on a line). Now pick a point on the line, this point divides the line into two parts. The line has 1 dimension, while the point has 0 dimensions. So a point is a hyperplane of the line. For two dimensions we saw that the separating line was the hyperplane.

B. *Random Forest*

Random Forest is a supervised learning algorithm. The Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Scikit-Learn provides a great tool for this, that measures a features importance by looking at how much the tree nodes, which use that feature, reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results, so that the sum of all importance is equal to 1. Through looking at the feature importance, you can decide which features you may want to drop, because they don't contribute enough or nothing to the prediction process.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems, but in this case classification is only needed.

VII. DESCRIPTION OF BENCHMARKING EXPERIMENT

Transformed dataset was split into two parts: train and test. To do that train_test_split from sklearn was used with test dataset size 0.2 of original one. Result recurrence was provided by defining random state for splitting. Algorithms were used in pipelines with two steps: scaling and algorithm implementation. For each algorithm grid search was done to find the best hyperparameters with 3 fold cross validation. Evaluation was prepared basing on accuracy, confusion matrix and classification report for both train and test dataset.

For SVM classificator various regularization parameter, kernel, polynomial degree and gamma parameter were searched.

For Random Forest different max depth, max used features, min samples in leaf, min samples in split and n estimators were searched.

VIII. ALGORITHMS BENCHMARKING

The best accuracy result was received in Random Forest algorithm with score 0.77. SVM classificatory was slightly worse with score 0.74. Both test results are worse than on train dataset, but Random Forest seems to be overfitted – the difference between train and test accuracy is 0.14. All accuracy results are presented in Table 1.

TABLE I. ACCURACY RESULTS

	SVM	**Random Forest**
Train accuracy	0.80	0.91
Test accuracy	0.74	0.77

As the dataset classes were not balanced – there were two times more positive decisions than negative – accuracy is not the best algorithm evaluator. In unbalanced dataset it is necessary to check more sophisticated metrics as precision, recall and F1- score.

Results in confusion matrix assure that algorithms distinguish classes and do not predict just one class for all samples. For all indicators in classification report Random Forest gives better results. Although for training data results are much more better – over 10 percentage points, for test data the difference is just 4 percentage points. Depending on situation such results can be used in real time situation. Confusion matrix for train dataset is presented in Table II and Table III contains confusion matrix for test dataset.

TABLE II. CLASSIFICATION RAPORT FOR TRAIN DATASET

	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
0	0.75	0.72	0.73	0.90	0.87	0.88
1	0.84	0.86	0.85	0.92	0.94	0.93
AVG	0.80	0.80	0.80	0.91	0.91	0.91

TABLE III. CLASSIFICATION RAPORT FOR TEST DATASET

	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
0	0.66	0.65	0.65	0.69	0.76	0.72
1	0.79	0.79	0.79	0.84	0.79	0.82
AVG	0.74	0.74	0.74	0.78	0.78	0.78

Very interesting part of evaluation is comparing results basing on student's application kind. Some kinds of applications have almost 1.0 accuracy what means that for this applications machine learning algorithm can fully cover human work - application kinds like "Wniosek o potwierdzenie opłaty", "Wniosek o zmniejszenie opłaty" and "Wniosek o przedłużenie terminu pracy dyplomowej". Table IV contains classification reports depending on application kind. Again Random Forest mostly gives better results than SVM, though for "Wniosek o urlop" SVM gives better results. It is worth mentioning that samples quantity of different kind of applications were not balanced.

TABLE IV. SUMMARY CLASSIFICATION RAPORT FOR DIFFERENT KINDS OF APPLICATION

Wniosek o rejestracje						
	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
AVG	0.68	0.68	0.68	0.81	0.81	0.81
Wniosek ogólny						
	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
AVG	0.70	0.70	0.70	0.72	0.72	0.72
Wniosek Potwierdzenie opłaty						
	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
AVG	1	1	1	1	1	1
Wniosek o przedłużenie ternimu pracy dyplomowej						
	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
AVG	0.95	0.94	0.94	0.95	0.94	0.94
Wniosek o rozłożenie opłaty na raty						
	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
AVG	0.69	0.69	0.69	0.89	0.86	0.85
Wniosek o urlop						
	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
AVG	0.64	0.8	0.71	0.53	0.4	0.46
Wniosek o zmniejszenie opłaty						
	SVM			Random Forest		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
AVG	0.86	0.83	0.76	0.77	0.81	0.77

IX. TIME EFFICIENCY

One on many factors considered during machine learning project implementation and distribution is algorithm learning time. When algorithm need to be refitted to new data, learning time can have crucial impact on received results. For Dean's decision classification various algorithms were tested and the best results was given by SVM and Random Forest – two the most computationally complex and time consuming. Table V contains algorithms learning time.

TABLE V. ALGORITHM LEARNING TIME

Algorithm	**LR**	**SVM**	**NB**	**KNN**	**RF**
Time mean (ms)	64,10	502,00	1,94	2,92	752,00
Time std (ms)	3,24	2,69	0,52	0,22	5,25

X. SUMMARY

Machine learning algorithms are increasingly used in real life situations and are helping with more and more human duties and responsibilities automatization.

For Dean's decisions two machine learning algorithms – SVM and Random Forest - were evaluated. Though Random Forest gives better results than SVM, the result can be used in real life only for specific kind of applications. Both algorithms are computationally complex what results in long training time. Algorithms can give better results with more data as some kinds of applications had few samples. More sophisticated algorithms like complex neural networks or xgboost can provide better results too.

XI. REFERENCES

[1] C. Cortes and V. Vapnik. Support-vector networks. Machine Learning, 20:273–297, November 1995.

[2] Decoste, Dennis (2002). "Training Invariant Support Vector Machines" *Machine Learning **46**: 160-190*

[3] Tin Kam Ho (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

[4] Breiman L (2001). "Random Forests". *Machine Learning.* **45** (1) 5–32.

[5] "scikit-learn 0.20.2 documentation"