Comparison of Support Vector Classifiers and Decision Trees in Fradulent Employment Offers' Detection.

Human-Centered Machine Learning Course 2022 – Group 9

Mireia Mei Torrecillas Cisa, Marta Borràs, Vilja Lott & Patrik Vodila

mireiameiya@gmail.com, martaborrasd@gmail.com, viljalott@gmail.com, patrik.vodila@gmail.com

Abstract.

The present paper is the group project developed throughout the Human-Centered Machine Learning Course, which has taken place in Aalto university. This work aims to be one of the main components by which students will further increase their knowledge on machine learning (ML). It is also meant to be an opportunity to put to practice the methodologies that have been taught in the lectures and to develop the necessary skills to solve a basic real-life ML problem.

Keywords: *Scientific Paper, Machine Learning, Group Project, Model, Classification, Loss Function, Data.*

1. Introduction.

1.1. Choosing a real-life ML problem: Detection of fraudulent employment offers.

The students enrolled in the summer course were divided into twelve different groups, and each one was assigned a topic. This group in particular, group 9, was assigned the topic of "Detection of Misinformation". Therefore, the real-life problem that the team has decided to focus on is the detection of fake employment offers from online applications and web pages.

Thanks to the current digital era, online job searching has become the most common way of job seeking, since it is the fastest and easiest way to gain access to all kinds of job offers from different backgrounds. However, it is far from being a safe environment, as it is hard to detect whether an offer is, in fact, genuine.

Scam activity has been noticeably risen throughout the years, thanks largely to the anonymity that the internet offers. These spurious agents commonly post promising opportunities on job boards, sometimes in the name of reputable recruiters, companies or firms, which target vulnerable or desperate job seekers. These cyber criminals deceive victims into believing they have a job or a potential offer, and leverage their position as "employers" to persuade victims to provide them with their personal identifiable information, which can be used for many nefarious purposes, and thus becoming subjects of identity and money theft. This matter is also concerning for companies whose identity has been stolen for illegal purposes [1] [2] [3].

Cyber criminals often request the same information as legitimate employers, making it difficult to discern the legitimacy of the offer until it is already too late [3]. To address this problem, the present project aims to develop a machine learning based solution which enables users to automatically detect whether an employment posting on the internet is fraudulent or not. The system will be trained on previous legitimate and non-legitimate job offers, and will use classification techniques to recognize offer genuinity.

1.2. Organization of the contents.

The present paper is structured in the following sections:

Problem formulation: Definition of the ML problem, selection of the dataset, and determination of the baseline.

Methods: Explanation of the choice of data and features, classification methods and hyperparameters, and training and validation procedures.

Results: Presentation of results obtained for the models, and comparison between them, and the resulting decision for the final model.

Conclusion: Recompilation of results obtained and critical discussion of overall performance of the models, and proposition of possible improvements.

2. Problem Formulation.

We have derived our underlying dataset from kaggle.com where it has been uploaded by Shivam Bansal under the name “Real/Fake Job Posting Prediction” [4]. The dataset consists of 17,880 job descriptions (our data points), of which 17,014 are real and 866 are fake. The data contains textual information and metadata about the offers. The job offer information are used to build binary classification models that can detect whether a job offer is real or fake.

Each of the data points in our dataset is a job offer consisting of different features and the label “fraudulent” with binary values “yes” and “no”, depending on whether the job offer is fake. Besides the label, the dataset has another 16 features. We have five string features: “title” of the job offer, “company profile” describing the company, “job description”, “job requirements” such as language or software skills, and “benefits” offered by the employer. The remaining eleven features are categorical. Three of them are binary categorical features: whether the “company logo” is present, whether the job includes “telecommuting”, and whether there are “selection questions”. In addition, other categorical features with more than two values are provided. The “employment type” of the job with five values, including “full-time” and “part-time” employment. The “required experience” feature with seven values is about work experience and the “required education” feature with 13 values is about the expected level of education. Then there is the job “function” with 37 values and the “industry” the company is part of with a total of 131 values. Three features have a lot more values: the “salary range” with 874 unique values, the “department” of the job within the company with 1,337 values, and the “location” of the job consisting of city, state, and country with a total of 3,105 unique values in the dataset.

As a baseline we have chosen to use a majority class classifier. A majority classifier predicts the majority class for each data point. So, in our case, the classifier labels each job offer as real. By comparing this classifier with our final machine learning model, we can find out if and how much more skilful our trained model is in detecting fake job offers.

3. Methods.

3.1. Data Gathering and Pre-Processing.

Before deciding how and what data to use from the dataset, we analysed each of the features.

In total we have 17,880 data points, but not every data point has information about every feature. Nevertheless, we considered all data for training and testing of our models. We have decided not to include three features in our models, these are “location”, “department” and “salary range”. One reason for this is that these features all have a large number of unique values (mentioned in Section 2). In addition, for “department” and “salary range”, more than half of the data points lack information about these features. Since we still have 13 features for our machine learning models without those three features, we decided not to include them. In order for the chosen machine learning models to handle the features, we needed to pre-process them. We converted the existing string features into numerical features. Therefore, we counted the number of characters in the strings which resulted in the used features for the models. Since there were now numeric and categorical features, we did further pre-processing to ensure we could use them all. For the five numerical features, we used a standard scaler that normalizes each feature by removing the mean for each feature and setting the variance to one [5]. For the eight categorical features, we used the one-hot encoder method, which converts categorical data into numerical data [6]. Consequently, we ended up with 13 numerical features in 17,880 data points to train our models with.

3.2. Chosen Models.

Since we have a binary classification task where we want to decide whether a given job offer is fake or real, we considered classification methods. We decided to use three different models: support vector classifier (SVC), decision tree classifier (DTC), and random forest classifier (RFC). When visualizing the individual feature values, we saw strong similarities between the fake job offers and the real job offers, so we were not sure whether our data could be linearly separated. For this reason, we decided to use classification methods that can handle non-linear data. An additional motivation for using the DTC and RFC is that these classifiers are easy to interpret and one can track their decisions.

For SVC, we use hinge loss as the loss function, while for DTC and RFC we use gini impurity as the loss function, respectively. We chose gini impurity instead of information gain because there is only a small difference in the results, yet Gini is easier to calculate and therefore faster as Gini is not using logarithm in its formula [7]. We decided to use these loss functions because they allowed us to use the ready-made Python library scikit-learn.

However, these were only the loss functions to learn the hypotheses. To compare the performance of the different models we used the metrics accuracy and F1-score. In the end, for the calculation of the performance of the final model we used the metrics again.

3.3. Hyperparameters.

For the SVC, we tried different values for the three hyperparameters “kernel”, “C”, and “gamma”. The hyperparameter “kernel” defines the kernel type to be used in the algorithm; we used a “linear” and the “rbf” kernel. “C” defines the misclassification-margin trade-off. Setting the “C”-value high tends to minimize the misclassification of training data, and setting it low tends to maintain a smooth classification. We have tried 12 different values for “C”: 0.1, 1, 3, 5, 8, 10, 12, 15, 16, 17, 18, and 20. The “gamma” hyperparameter defines how far the influence of a single training example reaches; we tried the values “scale” and “auto” for it [8].

To tune the DTC model we had three hyperparameters called “min_samples_leaf”, “min_samples_split”, and “max_depth”. The hyperparameter "min_samples_leaf" defines the minimum number of samples that must be in a leaf. We tried the values 2, 4, and 6 in the hyperparameter search. The “min_samples_split” parameter specifies the minimum number of samples required to split an internal node. We tried the values 5, 6, and 7. Lastly, the hyperparameter “max_depth” specifies the maximum tree depth, for this we chose the values 14, 16, 18, and 20. For the RFC the hyperparameters are the same as for the DTC. However, the hyperparameter “n_estimators” is added, which specifies the number of trees in the forest. For this we tried the values 30, 40, 50, 60, 70, 80, 90, and 100 [9][10].

Moreover, we always set the weights of the classes “true” and “fake” to be balanced for all classifiers, so that we counteract the imbalance in our dataset during training and avoid a majority class classifier. For testing we didn’t use balanced weights.

3.4. Model Validation.

To train and validate our model, we divided the whole data set into a train set and a test set. The train set consisted of 70% of the data set (12,516 data points) and the test set of the remaining 30% (5,364 data points). Since the data set is imbalanced in terms of the amount of fake and real labelled job offers, we made sure that the ratio between the two classes remained the same in all splits. We used the train set to find the best hyperparameters for the models with grid search and 5-fold cross-validation. That means our train set was split into 5 folds and always trained on 4 folds and validated on the fifth fold. After finding the best hyperparameter configurations for the models, we decided which model is best depending on the accuracy and F1-score. To do so, we trained the models on 70% of the train data and took 30% of the train data as the validation set. After that, for our final model, we used the entire train set to train the model on and then used the test set to validate the performance of the model using accuracy and F1-score. By splitting the data set and using cross-validation, we wanted to prevent our models from overfitting. We wanted to test our models on data not seen in training to see how well our models can generalize [11].

4. Results.

Firstly, we present the results of the train- and test-score for learning the hypotheses on the training set for each model followed by the accuracies of all models on the validation set and the accuracy and F1-score of the final model on the test set.

For SVC, the model with the best training and test score in our hyperparameter space has a value of “20” for parameter “C”, “scale” for gamma, and an “rbf” kernel. The values of hinge-loss are between 0 and 1, with 0 for a perfect model and 1 for a poor model. Using 5-fold cross-validation the hinge-loss on the training set is 0.0511 (var = 0.008) and on the test set 0.097 (var = 0.008). Thus, both values are relatively low which indicates a good performance. The values differ a bit, which indicates slight overfitting, but we have considered this to be an acceptable discrepancy. Using the grid search with 5-fold cross-validation, the best found hyperparameter configuration for the DTC has a train-score of 96.4% (var = 0.002) and a test-score of 94% (var = 0.002). The corresponding hyperparameter configuration consists of “min_samples_leaf” of 2, “min_samples_split” of 5, and “max_depth” of 18. The best hyperparameter configuration we could find for the RFC has the same values for “min_samples_leaf” and “max_depth” as the DTC. For “min_samples_split” the value is 6 and for the hyperparameter “n_estimators” 70. For RFC, the cross-validation gave a train-score of 95.7% (var = 0.005) and a test-score of 94.2% (var = 0.005) for accuracy. For both models, all accuracy values are very high and thus satisfactory. Furthermore, the difference between the train and test scores is not very big and therefore acceptable.

Calculating the accuracies for all three models on the validation set resulted in an accuracy of 93.82% and F1-score of 51.87% for the SVC. For the DTC an accuracy of 92.73% and an F1 of 49.54%. Lastly, for the RFC the accuracy is 93.58% while the F1-score is 53.74%. The results for the SVC and the RFC are very similar, but since the F1-score for the RFC is a bit better, we decided to use it as our final model.

We trained our final RFC model with the hyperparameters we found on the entire training set. The test set consists of 30% of the whole dataset, it is data that was not used at all in the previous evaluation. The ratio between fake and real labelled job offers in the test set is the same as in the whole dataset. When validated on the test data, we obtained an accuracy of 93.92% an F1-score of 56.53% for the SVC. The accuracy achieves a good score, while the F1-score is not as satisfactory.

5. Conclusion.

We aimed to develop a binary classifier based on nearly 18,000 job offers, of which about 5% were fake, to detect whether a job offer is real or not. We used 13 different features and tried three different machine learning methods, namely support vector classifier, decision tree classifier, and random forest classifier. Our trained RFC model proved to be the best method we could find for this task with an accuracy of 93.92% and F1-score of 56.53% on the test data. These are acceptable results. Our baseline, the majority class classifier, achieves an accuracy of 95.15% and F1-score of 0% on the test data. Thus, our trained model is a bit worse concerning accuracy but significantly better in terms of F1-score and consequently not a majority class classifier. However, a very high accuracy and a lower F1 score may indicate that our trained model has a slight tendency to be a majority class classifier. This means that it often labels the job offers as real and is usually correct due to the imbalance in our dataset; however, it has difficulties to reliably detect the fake job offers.

It might be possible to improve the RFC model even more by looking for further hyperparameter configurations. But we will probably never find an optimal tree since for DTC and RFC learning an optimal decision tree is an NP-complete problem [12]. Also, we have to be careful not to get into overfitting when further searching hyperparameter configurations, since decision trees and random forests tend to overfit [13]. Nevertheless, overall it is very convenient that with RFC the decision process is transparent.

Concerning the SVC model, it might be possible to improve it as well, by testing further hyperparameter configurations. However, it could also be possible that we are not able to get better results, as SVM/SVC algorithms are not well suited for large datasets and our dataset is relatively large with almost 18,000 data points. Furthermore, the features in our dataset have overlapping values in the two classes “fake” and “real”, which is also not favourable for a good SVM/SVC performance [14].

6. Bibliography.

[1] Joanna Zambas Content Manager and Career Expert Joanna joined the CareerAddict content team in 2017. (2022, June 27). *How to check if a job offer is fake or genuine*. CareerAddict. Retrieved August 26, 2022, from https://www.careeraddict.com/recognize-a-fake-job-offer-letter

[2] *Don't be a victim of fake job offers – run the scam check [infographic]*. Vista Projects. (2021, June 3). Retrieved August 26, 2022, from https://www.vistaprojects.com/blog/identify-fake-job-offers/

[3] FBI. (2021, April 21). *FBI warns Cyber Criminals are using fake job listings to target applicants' personally identifiable information*. FBI. Retrieved August 26, 2022, from https://www.fbi.gov/contact-us/field-offices/elpaso/news/press-releases/fbi-warns-cyber-criminals-are-using-fake-job-listings-to-target-applicants-personally-identifiable-information

[4] Bansal, S. (2020, February 29). *Real / fake job posting prediction*. Kaggle. Retrieved August 26, 2022, from https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction

[5] *Sklearn.preprocessing.StandardScaler*. scikit. (n.d.). Retrieved August 26, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[6] *Sklearn.preprocessing.onehotencoder*. scikit. (n.d.). Retrieved August 26, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

[7] Aznar, P. (2020, December 13). *Decision trees: Gini vs entropy Quantdare*. Quantdare. Retrieved August 26, 2022, from https://quantdare.com/decision-trees-gini-vs-entropy/

[8] *Sklearn.svm.SVC*. scikit. (n.d.). Retrieved August 26, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[9] *1.10. decision trees*. scikit. (n.d.). Retrieved August 26, 2022, from https://scikit-learn.org/stable/modules/tree.html

[10] *Sklearn.ensemble.randomforestclassifier*. scikit. (n.d.). Retrieved August 26, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[11] *3.1. cross-validation: Evaluating estimator performance*. scikit. (n.d.). Retrieved August 26, 2022, from https://scikit-learn.org/stable/modules/cross_validation.html

[12] *Constructing optimal binary decision trees is NP-complete*. (n.d.). Retrieved August 26, 2022, from https://people.csail.mit.edu/rivest/HyafilRivest-ConstructingOptimalBinaryDecisionTreesIsNPComplete.pdf

[13] *1.10. decision trees*. scikit. (n.d.). Retrieved August 26, 2022, from https://scikit-learn.org/stable/modules/tree.html

[14] K, D. (2020, December 26). *Top 4 advantages and disadvantages of support vector machine or SVM*. Medium. Retrieved August 26, 2022, from https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107