

Predicting Fake Job Posts

***Abstract:*** People are facing with the problem of fake job posts for long period of time. To get close to solving such problem, we tried with machine learning algorithms and data set to teach machine to predict whether job post is fraudulent (fake) or not. We used data set from Kaggle `Fake JobPosting Prediction` that consists out of 18K job posts (out of which 800 are fake) and 17 attributes (id, description, benefits). Through data exploration we learned that most important attribute in determining whether post is fake it is its description. To be more precise, number of certain frequent words that were connected to job description. Through different data preprocessing we prepared our data to be consist only of frequent words from description and fraudulent column (target attribute). To predict whether job post is fake we decided to try on two algorithms most suitable for this kind of classification. We decided to use kNN and Svm algorithm. With tuning of those two algorithms and training them on Train Set (80% of data), with cross validation of 10 we obtained on Test set (20% of data), 97.2% accuracy for kNN and 96.3% for Svm, which led us to conclusion, because of imbalanced data and almost 23% larger specificity to using Svm with specific tuning settings as main algorithm for predicting fake job post.

**Kenan Jamakovic**   
Faculty of Engineering and Natural Sciences  
International University of Sarajevo  
kenanjmk@gmail.com

**Asaf Kurbegovic**   
Faculty of Engineering and Natural Sciences  
International University of Sarajevo  
asafkurbegovic@gmail.com

**Amel Becic**   
Faculty of Engineering and Natural Sciences  
International University of Sarajevo  
becic.amel98@gmail.com

# Introduction

Job posting is a primary way that companies use to recruit new workers for available position. Job postings were present ever since the first job opportunities opened. Whether it was a black and white paper or big billboard on the side of the road it was always there [1]. Nowadays job postings are mainly present through online sources. It made finding jobs much easier for job seekers but it also made it easier for companies to find right workers for the job. But those benefits also came with the cost. With online job postings it is much easier to post fake job postings than ever before. Nowadays people can see many job postings on popular job seeking websites, many of them look very attractive, big salary, good working hours etc. [1]. Not all of them are real job postings and it is very important for job seeker to know which ones are real and which ones are or could be fake job postings. For this project we decided to create machine learning algorithm to predict whether a job posting is real or fake. For that task we will be using the [Real or Fake] Job Posting dataset found on Kaggle. Algorithm will be developed using knn and svM. After finishing the project, we plan to have an algorithm which will with high accuracy be able to tell whether a certain job post from the dataset is real or fake.

# materials and methods

## DATA SET USED AND DATA EXPLORATION

The dataset used for this project is [Real or Fake] Fake JobPosting prediction from kaggle (https://www.kaggle.com/shivamb/real-or-fake-fake-jobposting-prediction). This dataset contains 17,880 rows. There are 18 features in this dataset where five features are put as a text (title, company profile, description, requirements and benefits) and all other are categorical data or numeric fields. Dataset has a column named fraudulent where there is value 0 for fake job and value 1 for real job posting. There are also many missing values in dataset which are used as valid observations. Those missing values could make it easier to find fake job postings as they often have some missing fields.

## DATA PREPROCESSING PERFORMED

After finishing data exploration, we started preprocessing data in order to make best model possible. First thing we had to make on our dataset was to eliminate id column because that type of column cannot give us any valuable information. When we dropped id column from dataset, we moved to next step which is creating text corpus from description column in our dataset. To be able to crate corpus object we had to use “tm” library. Now when we have our corpus, we have to remove all punctations and stop words so that they cannot make impact on final results because they are appearing commonly in our dataset, but they are not important. After that cleaning, we performed stemming of corpus and then created matrix of all words so it can find which word are occurring most frequently. For that matrix we used a function which will remove term sparsity is less than 0.995. When preprocessing of terms occurring in job description, we are left to create data frame from it, assign it column names, add dependent variable which is fraudulent column and remove duplicate column names. When all preprocessing is done, we can move to training our model on preprocessed data and check the results.

## METHODS

### kNN Algorithm

K – nearest neighbor algorithm is developed by information theorist Thomas Covers who used it for classification and regression in statistics. The input for this algorithm consists of k (number used for determination for number of neighbors) closest training example in feature space (individual measurable property of a phenomenon being observed).

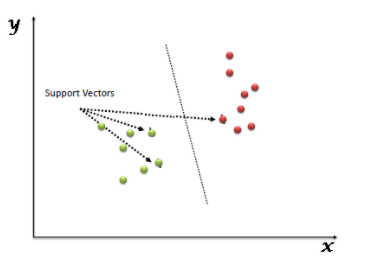
Output for Knn classification is a class membership. That is an object which is classified by plurality of its neighbors. If k is declared as 1 then object will be assigned simply to its one nearest neighbor.

Training examples for Knn algorithm are vectors with a class label. In the arrangement stage client will characterize k and unlabeled vector will be ordered by assigning the name which is generally regular among the k preparing tests closest to the query point.

Euclidean distance is most commonly used distance metric for continuous variables and for discrete variables (such as text) overlap metric (Hamming distance). Classification precision can be improved if the distance metric is found out with particular calculation as Large Margin Nearest Neighbor.

### SvM Algorithm

SVM (Support Vector Machine) is machine learning algorithm which is used for classification and regression challenges. The way it works is it takes data and transforms it and then based on that transformation it finds the optima boundary between outputs. This technique is called kernel trick [3]. In SVM each data value is plotted in n-dimensional space. Value of every feature is put as a coordinate. After we plot each coordinates algorithm performs classification by finding hyper-plane which differentiates two tested classes.



Support vectors are basically all co-ordinates of every observation. SVM classifier is the best segregation of both classes [4].

## FEATURES AND SAMPLES

After all data is preprocessed and is ready to train, we have to divide it on training set and test set. We decided to make training set 90% of overall data and rest 10% test data. For creating training and test samples we used caret library which helped us divide data and set number of folds for k - NN and SVM algorithms. Beside that we had to set fraudulent column as factor.

### k - NN Algorithm

1. # Training the KNN algorithm and tuning for optimized k
2. knnAlgorithm <- train(fraudulent ~ .,
3. data = trainData, method = "knn", preProcess = c("center","scale"),
4. trControl = trainControl , tuneGrid = expand.grid(k = c(5)))

To train k- NN algorithm we used train method from caret library. For training our model we used all columns from data of train set and based on that model was able to decide is given job post fake or not. The algorithm method was “knn” and set to look for 5 nearest neighbors and have 5 folds.

### SVM Algorithm

1. svmAlgorithm <- train(fraudulent ~ .,data = trainData,
2. method = "svmLinear", preProcess = c("center","scale"),
3. tuneGrid = grid\_svm, trControl = trainControl)

Support vector algorithm is trained using train function. Fraudulent is put as a main data frame. TrainData is a data frame from which variables specified in formula will be taken. A method used in this train functionis svmLinear. tuneGrid is a data frame with possible tuning values, for tuneGrid we put grid\_svm. Lastly for the list of control parameters trControl we put train Control list. The full line goes as:

# RESULTS AND DISCUSSION

For the kNN and Svm algorithm we used same hyperparameters and we used same data split way, on which we trained and tested. We divided our data into test set and train set of 90% for train which is around (16.200 Posts) and 20% for test data which is around 1788 posts. We used seed to set it on sample kind of rounding and to be assure that it will in randomize order distribute data to train and test. Besides that, we used cross validation of 5 with verboseIter set to true (as suggested from Stack Overflow). Both train and test data as earlier said were crated out of sparse data frame from data set which was composed out of frequent words as attributes that we extracted from description and value that indicates how many times that word is repeating inside of that particular description of the post. Except 1911 Attributes (Words) there is one more column which is `Fraudulent` which indicates whether that particular job post with those frequent words was fake or not. Also, for both algorithms our preprocess was set to center and scale.

## KNN ALGORITHM RESULTS OVERVIEW

With train test and test set mentioned from the above that are used for both algorithms, and cross validation of 5 there are some additional hyperparameters that we tuned only for knn. In the first run we set our k to be tested up to 5 and got the results that are in the tables below. With the settings mentioned above and the k set to up 5 we achieved accuracy of 97.2% and other statistics mentioned below in the table.

Table 1. STATISTICS FOR KNN ALGORITHM

|  |  |
| --- | --- |
| Accuracy | 97.2% |
| Sensitivity | 99.71% |
| Specificity | 42.31% |
| AUC | 97.92% |

Table 2. CONFUSION MATRIX FOR KNN

|  |  |  |
| --- | --- | --- |
|  | Reference | |
| Prediction | Not Fake (0) | Fake (1) |
| Not Fake (0) | 1705 TP | 45 FP |
| Fake (1) | 5 FN | 33 TN |

Before we start analysis of performance of our knn algorithm, we have to mention that our positive class is 0 – Not Fake, so it represents that the post is not fraudulent or in other words the post is real job call. So, for the test set of size 1788 we got accuracy of 97.2%, which means that for 1738 out of those 1788 posts it got right prediction. It is high accuracy for algorithm that regarding it is white box but simplicity of logic on which this algorithm works, gave us impressive results, but underneath surface is that like that? So only 50 people, or 2.8% of test data was misclassified. It is high accuracy which makes this set gladly usable for this type of prediction in learning purposes and other if needed. Our Sensitivity is at 99.71% which means 1705 posts will get Not Fake and they are truly Not Fake (True Positive) and 0.29% of posts (5 posts) will get false negative that they got `Fake` but they are Not Fake (Not Fraudulent). Our Specificity is at 42.31% which is pretty low compared to our parameters. Reason for that except imbalance in data is as we can see that our algorithm misclassified 45 posts in case of false positive which is more than it got true for 33 true negatives. For True Negative (33 posts) they will get result of Fake and they are truly Fake and it will return 57.69% of false positive, which means that for (45 posts) they will get that they are not fake but they are actually fake. So, to conclude specificity and its results with sensitivity and accuracy in general we can say that knn when the post is really not fake, our algorithm will without any problem in 99.71% cases detect it, but because of imbalanced data and really small number of fake posts compared to overall data set when the post is actually fake there are 57.69% chances that our algorithm will not detect it.

We tried tuning it with k up to 10 and 15 but we got only lower Accuracy. So, after deep analysis of knn up to k=5 we decided to tune it up to 10 or 15 to see our results. But Accuracy was not that big of a deal, it lowered only 1 or 2% but Specificity went to lower than 20% which meant that in almost 80% cases when post is truly fake it won’t detect it. No matter that it was even better for detecting not fake posts when they are not fake it just made bigger percentage miss than k up to 5, which is mainly the problem of imbalanced data.

## SVM ALGORITHM RESULTS OVERVIEW

For the Svm algorithm just like for the knn, we used same hyperparameters and we used same data split way mentioned on the beginning of results and discussion section. Cross validation was also set to 5 and there are some specific hyperparameter tunes that we made only for svm. We set up our grid up to 0.01 and for the method linear svm was used.

Table 3. STATISTICS FOR SVM ALGORITHM

|  |  |
| --- | --- |
| Accuracy | 96.31% |
| Sensitivity | 97.89% |
| Specificity | 61.54% |
| AUC | 97.13% |

Table 2. CONFUSION MATRIX FOR SVM

|  |  |  |
| --- | --- | --- |
|  | Reference | |
| Prediction | Not Fake (0) | Fake (1) |
| Not Fake (0) | 1674 TP | 30 FP |
| Fake (1) | 36 FN | 48 TN |

Before we start analysis of performance of our knn algorithm, we have to mention that our positive class is 0 – Not Fake, so it represents that the post is not fraudulent or in other words the post is real job call. So, for the test set of size 1788 we got accuracy of 96.31%, which means that for 1722 out of those 1788 posts it got right prediction. Even though the accuracy is less than knn this set by other parameters gives better results overall which we will see. This set misclassified only 66 posts from the test set. Our Sensitivity is at 97.89% which means 1674 posts will get Not Fake and they are truly Not Fake (True Positive) and 2.11% of posts (36 posts) will get false negative that they got `Fake` but they are Not Fake (Not Fraudulent). Our Specificity is at 61.54% which is pretty low compared to our parameters but still bigger than in last algorithm. Reason for that except imbalance in data is as we can see that our algorithm misclassified 30 posts since it classified more successfully, it got true for 48 true negatives. For True Negative (48 posts) they will get result of Fake and they are truly Fake and it will return 38.46% of false positive, which means that for (45 posts) they will get that they are not fake but they are actually fake. So, to conclude in all three aspects, what is most important since our data is imbalanced and positive class is not fake is specificity and we managed to increase it to 61.54% even though generally it is really low with some tuning we managed to increase it to nice 62% that for imbalanced data it is good.

## KNN AND SVM RESULTS COMPARISION AND DISCUSION

|  |  |  |  |
| --- | --- | --- | --- |
| R | Accuracy | Sensitivity | Specificity |
| 1. | KNN (97.2%) | KNN (99.7%) | SVM (61.54%) |
| 2. | SVM (96.3%) | SVM (97.8%) | KNN (42.31%) |

When it comes to accuracy both algorithms have high accuracy but KNN is better for 0.9% with its 97.2% accuracy compared to svm accuracy of 96.3%. But we are working with imbalanced data and this is the real example how imbalanced data makes it difficult for us. Because of imbalanced data we have to watch other aspects and analyze them deeply. As in case of Accuracy, same for Sensitivity, knn algorithm has larger sensitivity than svm. Since our positive class is not fake and we have 17000+ of not fake posts in our data set and 800 fake, there is no problem with sensitivity in any algorithm. For knn it misclassified only 5 posts to fake and they were not. Svm misclassified only 36 posts to fake and they were not. But as we said if the post real status is not fake, our both algorithms have no problem detecting that and classifying it correctly. But because of imbalanced data when real status is fake that is where problems start. As we saw for knn it has 42.31% sensitivity and in 57.69% cases it will misclassify fake post as a not fake one. But with svm we increased that specificity to 61.54% for what we can say not great not terrible. But it would lead us to conclusion in term of results that svm is better choice and better algorithm for this type of classification with this set, also it is really secure for detecting not fake post when the post really is not but in 40% cases it can classify fake post as not that we saw from specificity as we said many times because of imbalance.

# CONCLUSION

To finish this project, we can conclude that accuracy and sensitivity is higher for k – NN algorithm compared to our results for SVM algorithm, but specificity is higher for SVM. That means that k – NN algorithm can recognize positive class better than SVM, and on the other side SVM is better at recognizing negative class correctly. But when we take overall result and get deeper in results, we can notice that k – NN recognized only 33 out of 78 fake job posts and SVM 48. This happened because dataset that we used was imbalanced. Our dataset had 1710 real job posts and only 78 fake what makes it really difficult for algorithms to be reliable for use in real life. At the first opinion the results of our algorithms were great because they had accuracy over 95% but after more research, we concluded that due to imbalanced data we cannot have reliable algorithms for detecting job posts and to train good algorithm we need more balanced database

# REFERENCES

[1] Irene A. Blake.. “What is job posting” Chron.. January 17.2019.

[2] Anne Fisher. “Too good to be true? Beware of fake online jobs” Fortune,, March 2.2020.

[3] Zoya Gavrilov. “What is support vector machine, and why would I use it” KDnuggets.. February 2017

[4] Sunil Ray. “Understanding SVM algorithm from examples” Analytics Vidhya October 6.2015.

[5] LANTZ, Brett. *Machine learning with R*. Packet publishing ltd, 2013.

[6] TAN, Pang-Ning; STEINBACH, Michael; KUMAR, Vipin. *Introduction to data mining*. Pearson Education India, 2016.

**Project Contribution**

**Project name: Predicting Fake Job Posts**

|  |  |
| --- | --- |
| Team Member Name | Specific Task Completed |
| Kenan Jamakovic | Coding (Exploration, Preprocessing, Training Algorithms, Improving), For report: Abstract, knn Result overview, svm Result overview, knn and svm result comparison and discussion, merging coordinating the team and ppt making |
| Asaf Kurbegovic | Coding (Exploration, Preprocessing, Training Algorithms), For report:  Research about k – NN algorithm, Preprocessing data (Features and samples), k – NN training model |
| Amel Becic | Coding (Exploration, Preprocessing, Training Algorithms), For report:  Research about SVM algorithm, Materials and methods, SVM training model |