International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol.4 Issue 9, September-2020, Pages: 6-14

Machine Learning and Job Posting Classification: A Comparative Study

Ibrahim M. Nasser[1] and Amjad H. Alzaanin[2]

Gaza - Palestine
Ibrahimnasser.research@gmail.com
Data Science Department. The Islamic University, Gaza – Palestine
Amjad@live.it

Abstract—*In this paper, we investigated multiple machine learning classifiers which are, Multinomial Naive Bayes, Support Vector Machine, Decision Tree, K Nearest Neighbors, and Random Forest in a text classification problem. The data we used contains real and fake job post. We cleaned and pre-processed our data, then we applied TF-IDF for feature extraction. After we implemented the classifiers, we trained and evaluated them. Evaluation metrics used are precision, recall, f-measure, and accuracy. For each classifier, results were summarized and compared with others.*

Keywords—Machine Learning; Text Classification; Natural Language Processing

1. INTRODUCTION

Machine Learning (ML) filed handles mathematical programs which capable of imitation the learning process in humans [1]. It has been used for predictive analytics in various fields, e.g. diagnosis models in medicine, analyzing call patterns in telecommunication, and analysis of large amount of data for different purposes in theoretical science. Because of ML is considered as artificial intelligence, its models should have the ability to learn, and adapt to the change in its environment, so it can provide solutions for all possible circumstances. ML models learns by optimizing their performance based on past experience (training data) [2].

ML classification programs are almost learn through supervision, supervised learning involves giving the model already classified data, so learning makes progress by comparing the actual versus predicted class and adjust their mathematical equations until it got acceptable margin between the actual class and the predicted one [3]. In the other hand, there is unsupervised learning, where is no labels (classes) provided, the model has to figure out patterns in the data provided, generate reasonable labels, and then trying to map training data with these labels [4]. In this paper, we will show various supervised ML methods applied on a text classification problem, which is became an important research topic due to the numerous amounts of text data and documents that we deal with online [5].

The purpose of our research is to measure the performance of the most common ML techniques on a text classification problem, providing a comparison between them. The ML methods used in our research are Multinomial Naïve Bayes, Support Vector Machine, Decision Tree, K Nearest Neighbors, and Random Forest. The coming sub-sections briefly explain the ML methods mentioned, and in section two, we will provide a brief of the previous related works. In section three we will explain our research methodology, and the remaining sections to explain the ML models' evaluation metrics used and results, and finally our conclusion.

1.1 Multinomial Naive Bayes

Bayes' theorem is a probability and statistics theory which helps in determine the probability of an event based on a previous knowledge of circumstances that related to that event [6]. The theorem formula is illustrated in Eq. (1). Letting that:

A, B: events,

P: Probability,

P(A|B): Giving that B has occurred, what is the probability for event A,

P(B|A): Giving that A has occurred, what is the probability for event B,

and P(A), P(B): The independent probabilities of A and B.

$$p(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

Naïve Bayes classifiers are Bayes-based ML algorithms which proved their efficiency in text classification problems [7]. The naïve Bayes model is a heavy simplification of the Bayes' theorem. It operates on a solid independence assumptions [8]. Which means, the probability of one attribute does not affect the probability of another; Giving a number (n) of attributes, the naïve Bayes model

makes (2n!) independence assumptions. However, the classifier algorithm often gives correct results [8]. This independence assumption makes learning easier, because the parameter of each attribute can be learned separately, especially if the we have a large a number of attributes, e.g. text classification problems, that's why it has been successfully applied on this research area [9].
There are two different common event models that based on naïve Bayes assumptions, the multi-variate Bernoulli, and the multinomial naïve Bayes (MNB). The MNB event model captures word frequency information in documents, which can help in classification. In MNB, the text document is considered as an ordered sequence of word events, so each document is drawn from a multinomial distribution of words [9].

1.2 Support Vector Machine

Support Vector Machine (SVM) is a statistical learning theory published by Vapnik [10]. It has the ability to learn functions from a set of labelled training vectors. Computer scientist built ML classifiers based on Vapnik work, and they were very promising especially in text classification problems [11], [12]. The SVM purpose is to find the optimal hyperplane (in case of 2-D problem, a linear equation that try the best to separate the classes), the optimal hyperplane can found by maximizing the margin between the nearest two different classes data points and the hyperplane, so it can be able to generalize the training pattern [13]. HC Kim, S Pang, HM Je, D Kim, and SY Bang [14], briefly introduced the theoretical background of the SVM. Theoretically, SVM classification is based on the classical structural risk minimization (SRM) model, which determines the classification decision function by minimizing the practical risk, as in Eq. (2):

$$R = \frac{1}{l}\sum_{i=1}^{l}|f(x_i) - y_i| \qquad (2)$$

Where l represents the length of the sample data, and f represents the classification decision function. When it comes to SVM, the primary purpose is finding the optimal separating hyperplane that results in a low generalization error. When the problem is linear, the classification function represented by Eq. (3).

$$f_{w,b} = sign(w.x + b) \qquad (3)$$

In SVM, the algorithm finds the optimal hyperplane by letting the largest margin between different classes, so the optimal hyperplane is required to meet the constrained minimization, as shown in Eq. (4):

$$\begin{aligned} &Min : \frac{1}{2}w^T w, \\ &y_i(w \,.\, x_i + b) \geq 1 \end{aligned} \qquad (4)$$

In the case of non-linear problems, the minimizations problem can be modified to allow misclassified data points. SVM originally is for binary classification problems, although it can be applied to multi-classification problems by combining SVMs.

1.3 Decision Tree

Decision Tree Classifier (DT) is a common ML method used for classification and prediction problems, the robustness of DTs lies in the fact that it represents rules. DT classifier has a form of tree structure, each node in the tree is either a decision node or a leaf node, the decision node specifies a rule or test that carried out on a single attribute value, while the leaf node indicates the class of a data sample, and the decision node may directly followed by a class, or it will have a sub-tree for each possible outcome of test carried out in the node. A DT can classify a data sample starting at the root of the tree and moving down (apply tests to attributes values) until it reaches a leaf node which is the class of that data sample [15]. Figure (1) illustrates a DT model predicts what type of car should a person buy. The data attributes are age, and marital status, and the class is either a sports car or a mini-van.

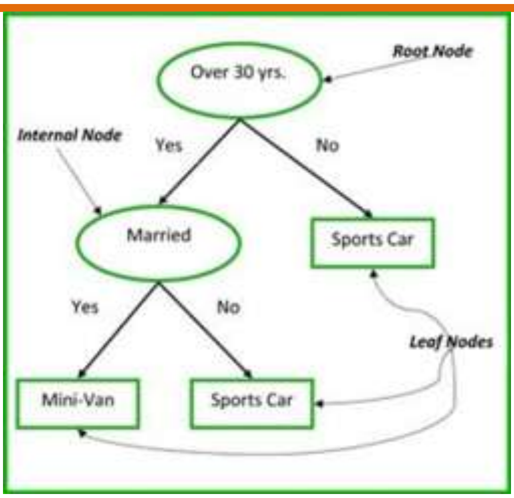

Figure 1: Decision Tree Example. Copyrights to Medium [16]

1.4 K-Nearest Neighbors

K- Nearest Neighbors (KNN) is a common ML classification technique due to its simplicity and efficiency [17]–[19]. It is a statistical method that utilizes the standard Euclidean distance and assesses the individual features. It makes no expectations with respect to the statistical structure of the data [20], [21]. KNN predicts the class of a new datapoint depending on a number (k) represents the nearest training examples in the feature space. The algorithm chooses the k nearest samples from the classified training vectors and determines the class of the new sample according to the most (alike) and the nearest samples. Euclidean distance is the metric that used in the algorithm to select the neighborhoods, letting that x_i and y_i are two points in Euclidean n-space, its formula is shown in Eq. (5) [22].

$$Euclidean\ distance = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{5}$$

1.5 Random Forest

Random Forest (RF) is a group (forest/ensemble) of Decision Trees (DTs), in a way that each tree provides a class prediction, then the class with the most votes becomes the forest prediction [23]. Creating a random forest starts with creating a bootstrapped dataset that is the same size as the original, and to create it, we randomly select instances from the original dataset with the ability of duplication. Each bootstrapped dataset will miss instances from the original dataset, these instances are called out-of-bag dataset (OOB), so each bootstrapped dataset has its own OOB. Then we start creating a DT using the bootstrapped dataset, but we only use a random subset of features at each step to create a node instead of choose from the whole set of features as in DT. Using bootstrapped datasets and considering only a subset of the features at each step results in a wide variety of trees, this variety is what makes the RF more effective than individual DT. In evaluation, we test each OOB dataset through all of the trees that were built without it, and for each, if the most voted class was correct, so our RF correctly classified that OOB. Finally, we can measure how accurate our RF is by the proportion of OOB samples that were correctly classified by the RF. Moreover, we calculate the proportion of OOB samples that were incorrectly classified, and that gives us the OOB Error. Again, we choose a larger random subset of features each step to build another RF, and then we can compare the OOB error for the previous RF to the new OOB error with the larger subset of feature. And So on, we test a bunch of different settings until we got the most accurate RF.

2. Related Works

Several research studies have been published in machine learning (ML) and text classification (TC). In TC, ML researchers have found a challenging application, because, datasets consist of numerous documents which characterized by a large number of terms. In addition, in TC problems, the challenge lying in trying to capture real semantic patterns, while other problems are just capturing content, e.g., image classification, ML models try to capture color distribution, shapes, and texture [24].

Khan, Aurangzeb, et al. provided a review of ML algorithms for text documents classification, they concluded that SVM, NB, and KNN classifiers were the most appropriate in TC [25]. The NB was the best performing in spam filtering and email categorization. In the other hand, SVM classifier has been recognized as one of the most effective TC method comparing to the supervised ML methods [26]. And when it comes to KNN, it will achieve a very good results if a suitable text pre-processing is performed [27], [28]. As for NB, it also needed suitable pre-processing.

In [29]: Artificial Neural Network (ANN), SVM, NB, and J48 classifiers were applied on email data classification, which is a binary classification problem (1 for spam, and 0 for other). The research was performed based on different data size and different feature size. That research paper showed that simple J48 classifier which makes a binary tree, could be efficient for the dataset which could be classified as binary tree.

Regardless what is classification method is being used, Wu, Weijun, Qigang Gao, and Muhong Wang found that the performance of a classification algorithm in data mining is greatly affected by the quality of data being classified. Issues in the data, e.g., irrelevant or redundant features, increase the cost of the mining process and reduce the quality of the results [30].

Our research, however, aims to evaluate the most common ML methods (MNB, SVM, DT, KNN, and RF), testing their efficiency on job posts classification problems, the classification problem we have is binary problem; in which to check the reality of the job posts, is it real, or fake.

3. Research Methodology

Using Google Collaboratory environment [31], we uploaded the data and pre-processed it, then we applied features extraction. We split the resulted pre-processed data vectors into training (70% of total), testing (30% of total), then we fed the data to the ML methods we programmed using python's libraries [32], [33] and evaluated them. Figure (2) illustrates the general model of our methodology.

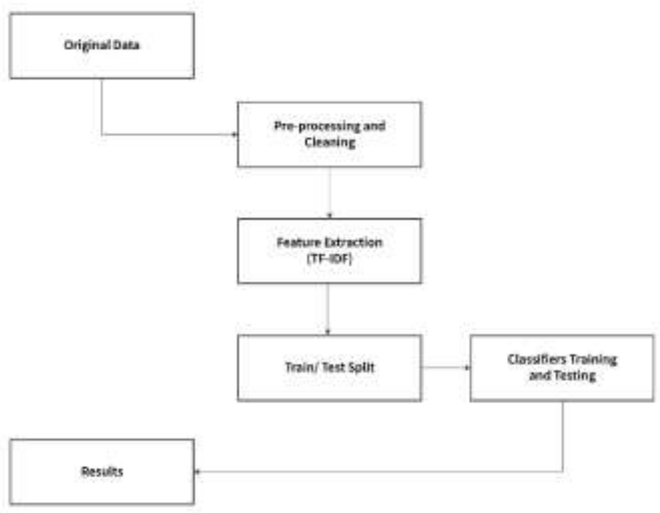
Original Data
Pre-processing and Cleaning
Feature Extraction (TF-IDF)
Train/ Test Split
Classifiers Training and Testing
Results

Figure 2: Research Methodology Diagram

3.1 Original Data and Pre-processing

The dataset used in this research is obtained from Kaggle, titled by "[Real or Fake] Fake Job Posting Prediction" [34]. The dataset contains about 18K job description out of which 800 are fake. The dataset has 17 attributes and one class label with binary values, dataset described in table (1).

Table 1: Original Data Description

Attribute	Data Type
Job ID	Integer
Title	Text
Location	Text
Department	Text

Salary Range	Integer
Company Profile	Text
Description	Text
Requirements	Text
Benefits	Text
Telecommuting?	Binary
Has Logo?	Binary
Has Questions?	Binary
Employment Type	Text (Categorical)
Required Experience	Text (Categorical)
Required Education	Text (Categorical)
Industry	Text
Function	Text
Fraudulent?	Binary

Because we are interested in a text classification problem, which is the job post itself, we just took the feature (Description) along with the class values, so the data just have the job description as the feature and which is real or not as the class. Then we applied a bunch of text cleaning techniques to prepare the text for the next step, which is features extraction. Text cleaning techniques used are:

3.1.1 Lowercasing

After we uploaded our data and save it into pandas' data frame, we lowercased all the job posts using the function *str.lower()*.

3.1.2 Drop null

Removing the empty text samples.

3.1.3 Tokenization

Using the natural language toolkit (NLTK) [35], which is one the best-known and most-used natural language processing (NLP) libraries, we tokenized the job posts by the function *word_tokenize()*, which returns a tokenized copy of the text.

3.1.4 Remove punctuation

Every token with non-alphabetical characters was removed.

3.1.5 Remove stopping words

A stop word is a commonly used word, such as the, a, an, in, etc. which any search engine ignores when indexing entries or retrieving them. In text mining, those words are not wanted, because, they reserve space in the dataset, and take valuable processing time. So, we removed them from our text data.

3.1.6 Stemming

Stemming helps in reducing words like 'plays', 'playing', 'played' to their common base form, like 'play'. We applied stemming to our data using the *PorterStemmer* class from the *nltk.stem* library.

3.2 Feature Extraction Using TF-IDF

In text classification problems, the challenge lies in the pre-processing and feature extraction, because text data has to be transformed into a representation, which is suitable for the learning algorithm [36]. We need to convert the text data into term vector. The term vector gives us numeric values corresponding to each term appearing in a text [37]. TF-IDF is the most useful and popular way to convert terms into vector [36]. The method extracted the most frequent words in the document and uses them as the features vector, in a way that each text data element (sample) is represented by the TF-IDF formula. E.g., let's say our text data is array of texts as following:

['This is the first document.', 'This document is the second document.', 'And this is the third one.', 'Is this the first document?']

And the features extracted are:

['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']

Then the data will be represented in a term vector model as shown in table (2):

Table 2: Term Vector Model Example

	and	*document*	*first*	*is*	*one*	*second*	*the*	*third*	*this*
First sample	0	TF-IDF	TF-IDF	TF-IDF	0	0	TF-IDF	0	TF-IDF
Second sample	0	TF-IDF	0	TF-IDF	0	TF-IDF	TF-IDF	0	TF-IDF
…	…	…	..	….	…	…..	….	…	…

TF-IDF value is computed using the formula stated in Eq. (6).

$$TF - IDF = \frac{No.of\ times\ the\ featured\ word\ appears\ in\ the\ sample}{Total\ no.of\ words\ in\ the\ sample} * \log \frac{Total\ numbre\ of\ samples}{No.of\ samples\ that\ contain\ the\ featured\ word} \qquad (6)$$

TF-IDF gives the higher weight to the important term and lesser weight to the unimportant one [37]. Now our data are converted to term vector model (TVM), applying TF-IDF resulted in a new shape of our data, which is (17879, 46319), which means the method extracted 46320 featured words. After preparing our TVM, now we can apply ML methods to our data and evaluate them.

4. Evaluation Merics

We used the confusion matrix to show the results of the ML models implemented, a confusion matrix is a specific table layout to visualize the performance of a supervised learning algorithm., its layout is shown in table (3).

Table 3: Confusion Matrix Layout

	Predicted Class		
Actual Class		Positive	Negative
	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Where (in case of our problem): **TP** defined as the real job posts that correctly classified as real, **FP** defined as the real job posts that incorrectly classified as fake, **FN** defined as the fake job posts that incorrectly classified as real, and **TN** defined as the fake job posts that correctly classified as fake. Moreover, the metrics used to evaluate the performance of the ML methods are: accuracy (Acc), recall (Sensitivity, or True Positive Rate (TPR)), precision (Positive Predicted Value (PPV)), and F-measure (F1), their equations are given in Eq. (7-10)

$$\boldsymbol{Accuracy} = \frac{Number\ of\ correctly\ classified\ samples}{Number\ of\ total\ samples} = \frac{TP + TN}{TP + FP + FN + TN} \qquad (7)$$

$$\boldsymbol{TPR} = \qquad (8)$$

$$\frac{Number\ of\ samples\ that\ correctly\ classified\ as\ real\ posts}{Number\ of\ samples\ that\ classified\ as\ real\ posts} = \frac{TP}{TP+FN}$$

$$\boldsymbol{PPV} = \frac{Number\ of\ real\ posts\ that\ correctly\ classified\ as\ real}{Number\ of\ samples\ that\ actually\ real\ posts} = \frac{TP}{TP+FP} \qquad (9)$$

$$\boldsymbol{F1} = 2 * \left(\frac{Precision * Recall}{Precision + Recall}\right) \qquad (10)$$

5. Results

Table (4) illustrates the confusion matrices we got after we trained and tested each ML model, in addition, table (5) shows the evaluation metrics results.

Table 4: Resulted Confusion Matrices

MNB Confusion Matrix		
	Real (predicted)	Fake (predicted)
Real (Actual)	5127	6
Fake (Actual)	231	0
SVM Confusion Matrix		
	Real (predicted)	Fake (predicted)
Real (Actual)	5131	2
Fake (Actual)	119	112
DT Confusion Matrix		
	Real (predicted)	Fake (predicted)
Real (Actual)	5063	70
Fake (Actual)	69	162
KNN Confusion Matrix		
	Real (predicted)	Fake (predicted)
Real (Actual)	5105	28
Fake (Actual)	91	140
RF Confusion Matrix		
	Real (predicted)	Fake (predicted)
Real (Actual)	5132	1
Fake (Actual)	96	135

Table 5: Evaluation Metrics Results

	Accuracy	Recall	Precision	F-measure
MNB	95.6	95.7	99.9	97.8
SVM	97.7	97.7	99.9	98.8
DT	97.4	98.7	98.6	98.6
KNN	97.8	98.2	99.5	98.8
RF	98.2	98.2	99.9	99.0

6. Conclusion And Discussion

Comparing the results, it turned out that DT, followed by RF and KNN achieved the highest recall, while the highest precision is achieved equally by MNB, SVM, and RF. Moreover, we have noticed that RF achieved the best f-measure, and accuracy. So, we are concluding that the most effective ML method dealing with our text classification problem is Random Forest, noticing that, it also achieved the least incorrectly predicted samples (FP+FN = 97), followed by the KNN (FP+FN = 119).

Although, Multinomial Naïve Bayes got the lowest accuracy. We think the reason behind the low performance in MNB lies in the nature of features as stated in [38]. Our feature selection method ended up with a vocabulary size of 46319 (about 46K) which is too large, and as stated in [39], MNB achieves better performance when smaller vocabulary size is used. We already know that using the full vocabulary limits the model applicability on memory constrained scenarios [40], [41], and its unnecessary in a way that many words may contribute little to the TC task and could have been removed safely from the vocabulary [42]. Nevertheless, we have tried to lower the size of vocabulary size, but it resulted in low performance in other ML methods. Moreover, we think the large dataset size made the RF the most efficient method, which is similar to a previous research results that showed RF achieves high accuracy in the case of large number of instances [43].

So, finally, according to our TC problem, and taking into consideration the text pre-processing we did, and feature extraction we have applied, the Random Forest is the best Machine Learning algorithm to solve the problem at hand.

References

[1] B. K. Natarajan, *Machine learning: a theoretical approach*. Elsevier, 2014.

[2] E. Alpaydin, “Introduction to Machine Learning. [Sl].” The MIT Press, 2010.

[3] B. C. Love, “Comparing supervised and unsupervised category learning,” *Psychon. Bull. Rev.*, vol. 9, no. 4, pp. 829–835, 2002.

[4] N. Japkowicz, “Supervised learning with unsupervised output separation,” in *International conference on artificial intelligence and soft computing*, 2002, vol. 3, pp. 321–325.

[5] Y. Yang, “An evaluation of statistical approaches to text categorization,” *Inf. Retr. Boston.*, vol. 1, no. 1–2, pp. 69–90, 1999.

[6] J. Joyce, “Bayes’ theorem,” 2003.

[7] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, “Multinomial naive bayes for text categorization revisited,” in *Australasian Joint Conference on Artificial Intelligence*, 2004, pp. 488–499.

[8] S. J. RUSSEL and P. Norvig, “Artificial Intelligence–A Modern Approach. Person Edu ca tion,” *Inc., New Jersey*, pp. 736–741, 2003.

[9] A. McCallum and K. Nigam, “A comparison of event models for naive bayes text classification,” in *AAAI-98 workshop on learning for text categorization*, 1998, vol. 752, no. 1, pp. 41–48.

[10] V. N. Vapnik, “The nature of statistical learning,” *Theory*, 1995.

[11] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *European conference on machine learning*, 1998, pp. 137–142.

[12] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, “Inductive learning algorithms and representations for text categorization,” in *Proceedings of the seventh international conference on Information and knowledge management*, 1998, pp. 148–155.

[13] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[14] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S. Y. Bang, “Pattern classification using support vector machine ensemble,” in *Object recognition supported by user interaction for service robots*, 2002, vol. 2, pp. 160–163.

[15] N. M. Tahir, A. Hussain, S. A. Samad, K. A. Ishak, and R. A. Halim, “Feature selection for classification using decision tree,” in *2006 4th Student Conference on Research and Development*, 2006, pp. 99–102.

[16] "Decision Trees — A simple way to visualize a decision." https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb (accessed Jun. 19, 2020).

[17] C. Eyupoglu, "Implementation of color face recognition using PCA and k-NN classifier," in *2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*, 2016, pp. 199–202.

[18] J. Wang, P. Neskovic, and L. N. Cooper, "Improving nearest neighbor rule with a simple adaptive distance measure," *Pattern Recognit. Lett.*, vol. 28, no. 2, pp. 207–213, 2007.

[19] N. Zhang, J. Yang, and J.-J. Qian, "Component-based global k-NN classifier for small sample size problems," *Pattern Recognit. Lett.*, vol. 33, no. 13, pp. 1689–1694, 2012.

[20] S. I. Niwas, P. Palanisamy, K. Sujathan, and E. Bengtsson, "Analysis of nuclei textures of fine needle aspirated cytology images for breast cancer diagnosis using Complex Daubechies wavelets," *Signal Processing*, vol. 93, no. 10, pp. 2828–2837, 2013.

[21] G. Shakhnarovish, T. Darrell, and P. Indyk, "Nearest-neighbor methods in learning and vision," in *Cambridge, MA*, MIT press, 2005.

[22] L. Shen, D. Cao, Q. Xu, X. Huang, N. Xiao, and Y. Liang, "A novel local manifold-ranking based K-NN for modeling the regression between bioactivity and molecular descriptors," *Chemom. Intell. Lab. Syst.*, vol. 151, pp. 71–77, 2016.

[23] L. Breiman, "Random forests, machine learning 45," *J. Clin. Microbiol*, vol. 2, no. 30, pp. 199–228, 2001.

[24] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.

[25] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "A review of machine learning algorithms for text-documents classification," *J. Adv. Inf. Technol.*, vol. 1, no. 1, pp. 4–20, 2010.

[26] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 42–49.

[27] P. Yuan, Y. Chen, H. Jin, and L. Huang, "MSVM-kNN: Combining SVM and k-NN for Multi-class Text Classification," in *IEEE international workshop on Semantic Computing and Systems*, 2008, pp. 133–140.

[28] F. Colas and P. Brazdil, "Comparison of SVM and some older classification algorithms in text classification tasks," in *IFIP International Conference on Artificial Intelligence in Theory and Practice*, 2006, pp. 169–178.

[29] S. Youn and D. McLeod, "A comparative study for email classification," in *Advances and innovations in systems, computing sciences and software engineering*, Springer, 2007, pp. 387–391.

[30] W. Wu, Q. Gao, and M. Wang, "An efficient feature selection method for classification data mining.," *WSEAS Trans. Inf. Sci. Appl.*, vol. 3, no. 10, pp. 2034–2040, 2006.

[31] E. Bisong, "Google Colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Springer, 2019, pp. 59–64.

[32] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[33] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011.

[34] S. Bansal, "[Real or Fake] Fake JobPosting Prediction | Kaggle." https://www.kaggle.com/shivamb/real-or-fake-fake-jobposting-prediction (accessed Jun. 03, 2020).

[35] S. Bird, E. Loper, and E. Klein, "Natural language processing with python O'reilly media Inc," 2009.

[36] L.-P. Jing, H.-K. Huang, and H.-B. Shi, "Improved feature selection approach TFIDF in text mining," in *Proceedings. International Conference on Machine Learning and Cybernetics*, 2002, vol. 2, pp. 944–946.

[37] V. Kalra and R. Aggarwal, "Importance of Text Data Preprocessing & Implementation in RapidMiner," in *Proceedings of the First International Conference on Information Technology and Knowledge Management–New Dehli, India*, 2017, vol. 14, pp. 71–75.

[38] S. Raschka, "Naive bayes and text classification i-introduction and theory," *arXiv Prepr. arXiv1410.5329*, 2014.

[39] L. M. Rudner and T. Liang, "Automated essay scoring using Bayes' theorem," *J. Technol. Learn. Assess.*, vol. 1, no. 2, 2002.

[40] D. Yogatama, M. Faruqui, C. Dyer, and N. Smith, "Learning word representations with hierarchical sparse coding," in *International Conference on Machine Learning*, 2015, pp. 87–96.

[41] M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. Smith, "Sparse overcomplete word vector representations," *arXiv Prepr. arXiv1506.02004*, 2015.

[42] W. Chen, Y. Su, Y. Shen, Z. Chen, X. Yan, and W. Wang, "How large a vocabulary does text classification need? a variational approach to vocabulary selection," *arXiv Prepr. arXiv1902.10339*, 2019.

[43] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *Int. J. Comput. Sci. Issues*, vol. 9, no. 5, p. 272, 2012.