Development of an Intelligent Job Recommender System for Freelancers using Client's Feedback Classification and Association Rule Mining Techniques

Md. Sabir Hossain, Mohammad Shamsul Arefin*
Dept. of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong-4349, Bangladesh.

* Corresponding author. Tel.: +8801716890204; email: sarefin@cuet.ac.bd
Manuscript submitted March 14, 2019; accepted May 22, 2019.
doi: 10.17706/jsw.14.7.312-331

Abstract: Most of the freelancer's time is killed in finding suitable jobs due to the huge number of freelance marketplaces. Freelancing sites send email notifications or show in newsfeed about posted jobs but most of them are irrelevant. Recommending relevant jobs to freelancers to minimize job finding time has drawn the attraction of researchers. Here, in this paper, we propose a recommender system to find out appropriate jobs for freelancers using client's feedback classification and Association rule mining techniques. After collecting the previous work history of freelancers, we analyze the sentiment of client's feedback using Logistic Regression and Linear Support Vector Machine model to classify the completed jobs into two categories: positive and negative. We apply the Association rule mining technique to find out freelancer's frequent skillsets used in both categories of completed jobs. Then, we find out the jobs matched with the positive frequent skillsets using set operations. We also discard jobs that contain negative frequent skillsets. Finally, a collaborative filtering algorithm is applied considering the client's overall rating, the minimum budget/ hourly rate, deadline, re-hire, etc. to generate a more accurate recommendation. After extensive experiments on the real dataset collected from different online marketplaces, we are able to prove that our proposed method correctly recommends the appropriate jobs with 83.40% (Logistic Regression) and 84.03% (Linear SVM) accuracy.

Key words: Association rule mining, collaborative filtering, freelancing, frequent skillsets, logistic regression, possible jobs, sentiment analysis, support vector classifier.

1. Introduction

Freelancing is becoming popular day by day and drawing the attention of workers, the fact demonstrated by Online Labour Index (OLI) [1]. People love to work independently (perform fascinated jobs at a suitable time) as well as to earn extra money. Also, clients want to hire people rather than recruiting permanent staff for his/her jobs. Within a decade, freelancers will be the majority working force of U.S.A. and already 50% of millennial employees are involved [2]. The study also showed that about 57.3 million people are freelancing and their annual contribution to the American economy is about $1.4 trillion. A survey has conducted by Payoneer [3] where over 21,000 freelancers from 170 countries have participated. According to the survey, online marketplaces (e.g. Odesk, Elance, Freelancer.com, etc) are becoming popular and 70% freelancers use them to find out their desirable jobs. There are more than 85 online market places exist [www.jobmob.co.il]. So, it is becoming a nightmare to find out appropriate jobs from a huge number of online marketplaces [4],

[5]. Most of the freelancers simply see the first page of the posted job. Most of the freelancer's time is killed in job searching. To find jobs, freelancers spend a different amount of time. According to [2], 33% freelancers from the legal field, 17% freelancers from IT and Programming, spend over 7 hours a week to find out the new jobs. The scenario turns into so challenging that many freelancers desire virtual assistants to find their suitable jobs.

Most of the freelancing marketplaces work in an identical way. Freelancers and clients both have to register on the site. Clients publish the job mentioning the job category, description, required skills, the budget or minimum hourly rate, deadline, etc. The system sends jobs as email notification or the web site job feed by matching the job category and competencies freelancers point out while registering. Also, the freelancers can see the job posts immediately on the website. Freelancers write a proposal for the job he/she feels interested. On the other hand, some websites suggest potential candidates for clients matching the criteria as a premium feature which is no longer usually possible. After taking the interview of the applicants or the potential candidates, clients figure out who he/she needs to hire. Finally, clients assign jobs to the appropriate candidates. While suggesting jobs to freelancers, these sites don't consider the previous work experiences of the workers nor the frequent skill sets used to perform tasks. Several frameworks have proposed by different researchers for assessing the freelancer's profile. Some researchers additionally tried to develop the platform to recommend the appropriate job for the freelancers for a stand-alone crowdsourcing website by performing different scoring-based approaches.

We propose a recommender system to discover suitable occupations for freelancers utilizing client's feedback classification and Association rule mining techniques. We have collected data from different online marketplaces and we are the first to host dataset in Mendely [6] on freelancing work history. Using Logistic Regression and Linear Support Vector Machine model, we analyze the sentiment of client's feedback after collecting the previous work history of freelancers to classify completed jobs into two categories: positive and negative. We apply the technique of Association rule mining to find out frequent skillsets used by freelancers in both categories of completed jobs. Then, using set operations, we find the jobs that match the positive frequent skillsets. We also ignore jobs which require frequent negative skillsets. Finally, a collaborative filtering algorithm is used to generate a more accurate recommendation, taking into account the overall rating of the client, the minimum budget/hourly rate, deadline, re-hire, etc. The key contributions of this paper can be summarized as follow:

- Build a dataset by retrieving freelancer's job-related data from different online marketplaces.
- Classify freelancer's completed jobs into positive and negative based on client's feedback analysis using Logistic Regression and Linear Support Vector Machine Models.
- Find out the frequent skillsets of freelancers from previous work history using Association Rule mining algorithm.
- Propose a filtering algorithm to generate a possible job list based on the freelancer's frequent skillsets, job category, client's rating, the minimum budget/ hourly rate, deadline, re-hire, etc.

The rest of the paper is organized as follows. Section II provides the discussion of the related work with associated limitations. Section III describes the proposed methodology in details including sentiment analysis using Logistic Regression and Association rule mining technique. Section IV discusses the experimental results of the proposed method. Finally, Section V concludes this paper with future directions to this research.

2. Related Work

Recommending appropriate jobs for freelancers is an interesting field for the researchers. Yuen *et al.* [7] provided a survey in which crowdsourcing literature was classified into four categories, namely the voting system, the information sharing system, the game, and the creative system. Since there is so much diversity

in the characteristics of freelancers (i.e. education, skills, experience, etc.), it is really a complicated task to hire suitable candidates. Kokkodis *et al.* [5] studied the behavior of the employee and proposed a model for rationally selecting the best possible one at hand. They proposed a sequence of probabilistic models to estimate the hiring probability of each candidate. They showed that after training and testing the models on huge application data from ODesk, their proposed system outperforms the fundamental algorithm. They identify some important issues related to freelancer selection are: freelancers and clients previously worked, freelancer profile accessible, freelancers skill sets and early application for the job.

Md Sabir Hossain and Mohammad Shamsul Arefin [8] proposed recent work to generate possible job list for freelancers. They offer an intelligent system to help freelancers find relevant jobs from diverse freelance websites by examining their past work histories and analyzing the facts found on jobs via multiple keyword search algorithms. They produce a list of frequent skills used in previous tasks by using the association mining algorithm. Their suggested scheme generates a job list that is viable taking into account freelancer skills, client rating, the budget / hourly rate minimum, deadline, etc. The main limitation of this work is that they only consider synthetic data to proof efficiency of their proposed system.

Yuen *et.al* has proposed a novel task [9] that matches the crowdsourcing systems, taking into account the worker's choice and previous work records. The proposed system provided the best matching job list while selecting the task from a crowdsourcing system. To do so, the predefined category of the crowdsourcing task must be established and the worker's preferences must be stored. TaskRank motivated employees of diverse backgrounds to work on long-term crowdsourcing tasks to improve work quality. Yuen et al. [10] used the matrix factorization approach to develop a system to recommend Amazon Mechanical Turk (Mturk) tasks. By analyzing the performance history of the worker and the task search history, their proposed model extracts the preferred tasks of workers. They showed that more than 86 percent of the workers are motivated by money and the type of tasks to prefer tasks. In another research paper, Yuen et al. [11] proposed a dynamic Task Recommendation (TaskRec) framework using a unified probabilistic matrix factorization in which the ratings are inferred from interactive behavior. Chilton et al. [12] showed that the favorable task search decreases the completion time by 30% and the unfavorable task search decreases the amount of money.

An essential framework using the classification method was proposed by Ambati *et al*. [13]. This method creates a workers model by inspecting the worker's successful work records in the crowdsourcing system. Then a classification algorithm is used to classify the available tasks into fascinating or non-fascination to the workers based on the workers model. Also, other researchers proposed the task recommendation in crowdsourcing platform different scoring based techniques [14]- [16].

Safran Mejdl and Dunren Che [17] proposed a real-time recommendation algorithm for crowdsourcing systems. Real-time recommendations are crying need not for employees but also for the job requesters. Employees wish to get the top-k most suited tasks matched with their skills and preferences. Requesters also want a reliable recommendation of the top-k best employees for their tasks in terms of worker's qualifications and accountability. Two real-time recommendation algorithms for crowdsourcing system have proposed by the authors named TOP-K-T and TOP-K-W that computes the top-k most suitable tasks for a given employee and the top-k best employees for a requester with relation to a given task respectively.

Zahid *et. al.* [18] introduce a framework to relate personality type and task characters. They use the Myers-Briggs type indicator to measure the participant's personality type. The same author proposes a task assignment model (TAM) [19]which would give CSD workers a primary structure to find suited tasks and a platform for direct assignment of tasks. The key element throughout this proposed system is to match the personality and task. To get away from the risk of providing jobs to a fallacious person, Capretz and Ahmed [20] have proposed an appropriate model where they recommended that tasks assigned to developers need to be based totally on their appropriate personality types. As the example, they showed that the personality

of a programmer must be Introvert (I), the personality of the system analyst need to be Extrovert (E), the tester has sensing (S) and Thinking (T) personality. The software designer ought to be with intuitive (N) and think (T) personality. The application of this model cannot be carried out due to the fact of its non-empirical nature.

In [21], Kazai *et al.* use behavioral observations to outline five employee types: Spammer, Sloppy, Incompetent, Competent, and Diligent. They relate the employee types to label accuracy and personality trait facts alongside the 'Big Five' personality dimensions. Demirörs et al. [22][23] also showed that an individual's personality has a direct correlation with the project of crowdsourcing software. Capretz et al. [23] mentioned that successful completion of any project relies closely on assigning a project to a specific personality type. Abhinav et. al [24] proposed a multidimensional evaluation framework named CrowdAdvisor which considers the current information about the freelancer as well as the previous accomplished jobs. They developed algorithms to generate a rating based totally on different dimensions. The proposed framework overall performance is higher than the baseline algorithm.

Difallah *et al*. [25] proposed a Crowdsourcing platform for personalized human intelligence task assignment based on social networks. Their proposed system selection of workers to perform a particular task is accomplished with the aid of extracting records from the worker's profile on social media. Based on the records extracted from the task description on crowdsourcing platforms and the worker's preferences from social networks are used to match the task and worker automatically. They showed the push approach works better than the pull method after the extensive experiment.

In the case of LinkedIn, the chance of getting a job for the job seekers will be reduced if so much job seekers compete for the same job. Therefore, it is very important to balance between job seekers and job posters. A system named LiJAR [26] proposed by Borisyuk et al. where they present a dynamic forecasting model in order to measure the number of candidates for a job on the cut-off date and algorithms, that on the basis of forecasting models, decides whether to promote or penalize jobs. The satisfaction of both parties has expanded through enforcing their system without affecting the number of candidates for a particular job on the expiration date.

To make the behavior of workers truthful to each other team formation in social networks (SNs) is very important as most of the workers are prejudiced and don't fancy to disclose individual information. Wang *et al.* [27] tried to optimize the benefit of workers by forming truthful team in small and large scale. For a complex task system consist of several independent micro-tasks, Tran-Thanh et al. proposed the BudgetFix cost-effective method [28] by assuming the hardness of each phase using prior knowledge and setting arbitrary arguments for each phase. They demanded quality guarantees of each phase of the workflow.

Matching buyers with freelancers for IT services in two-sided platforms is becoming challenging increasingly because of the difficulty in predicting the price the buyers should pay to freelancers. Zheng et al. [29] proposed a method to match buyers and freelancers where they used bid price dispersion to remove buyer indecision and freelancer regret. They empirically tested their model against a famous online market, "Freelancer.com" and showed that the satisfaction of both parties has increased. They also discussed the practical implications associated with two-sided platforms.

3. Proposed Methodology

The system architecture of the Intelligent Job Recommender Framework comprises four basic modules: storage module, client's review classification module, freelancer's frequent skillsets generation module, job recommendation module. The function of the storage module is to store the data retrieved from different marketplaces. Client's review classification module predicts the category of the feedback provided by clients for a specific completed job by freelancers. We use two algorithms (Logistic Regression and Linear Support

Vector Machine) to train and test our proposed model. Freelancer's frequent skillsets generation module finds out the frequent skillsets used in completed jobs with both positive and negative feedbacks. This module generates positive and negative frequent skillsets using Association Rule Mining (Apriori algorithm). Job recommendation module generates a possible job list by matching the jobs with freelancer's frequent skillsets using set operations. It also applies a filtering algorithm to generate a more realistic recommendation. We have discarded the jobs with less than minimum budget or an hourly rate of freelancers. We also discard the jobs where the client's rating is low and payment is not verified. Jobs with less than 5 days application deadlines are also removed from the list. The architecture of the framework is shown briefly in Fig. 1. The working procedure is explained step by step in the next consecutive subsections.

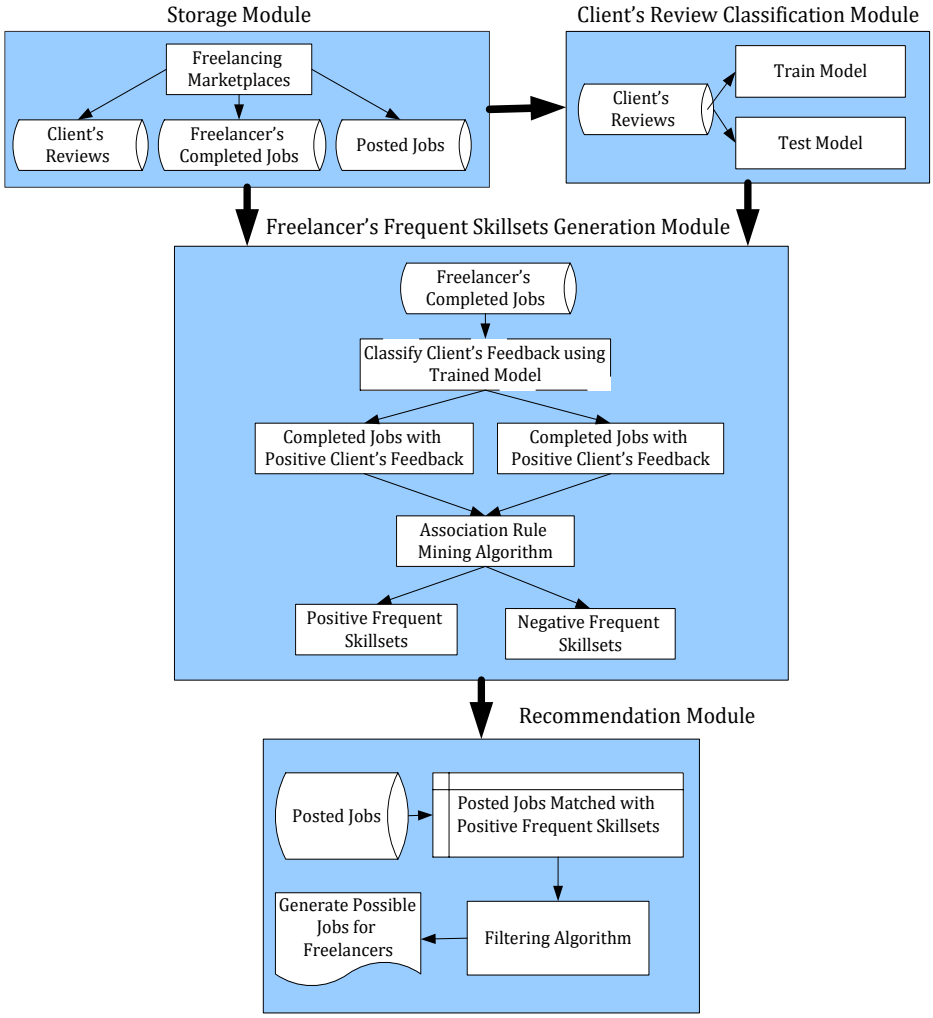


Fig. 1. Proposed methodology to recommend possible jobs to freelancers.

3.1. Storage Module

Best of our knowledge, there is no freelancing datasets publicly available. As freelancers generally don't want to share their work experiences publicly, it's become so hard to analysis their past work history. We have collected data from several freelancing marketplaces i.e. freelancer.com, upwork.com, peopleperhour.com and guru.com to prepare our experimental dataset which is now available at [6]. Client's feedback table contains the job rating, client's feedback/review, review type (positive/negative) and web source. In freelancer's work history table, we store job title, job description, skills required to complete the

job, job type denoting fixed or hourly. It also contains completed job rating and feedback provided by clients. Posted job table contains job title, job description, skills requirement, job category, rate, client's rating, payment (verified or unverified), and proposal deadline.

3.2. Client's Feedback Classification using Sentiment Analysis

Oxford Dictionary [https://en.oxforddictionaries.com/] defined sentiment analysis as "The process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative or neutral". Sentiment analysis is becoming popular because it helps the industrialist to understand the consumer's mind and make business policy accordingly, create a campaign, etc. Among rule-based, machine Learning based and hybrid systems, machine-based sentiment analysis categorizes the input text based on past observations. Pre-labeled texts (examples) are supplied into a classifier model to learn the category of the input in training stage and unknown text inputs are provided to get expected label from the trained model in testing stage (Fig. 2).

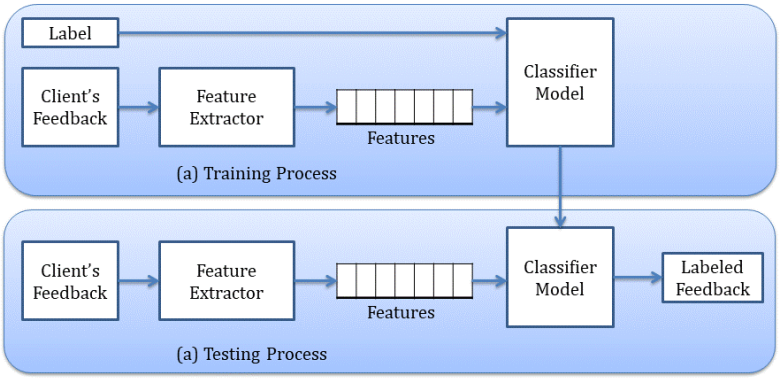


Fig. 2. Client feedback classification using sentiment analysis.

3.2.1. Training and testing process

Client's feedbacks with associated labels (e.g. positive or negative) are supplied to the classifier model. Features extracted from supplied feedbacks are transferred to feature vector. The classifier model learns in the training period from pairs of feature vectors and labels shown in Fig. 2(a).

In the testing process, feature extractor transfers the unknown client's feedbacks into feature vectors [Fig. 2(b)]. The classifier model predicts the tags of the unknown feedbacks from these feature vectors.

3.2.2. Feature extraction from text

Before supplying the texts into the classifier model, they need to be transformed into a numerical representation, generally called vector. Each component of the vectors denotes word frequency of the input texts. The process is known as text vectorizations alternatively feature extraction. Bag-of-words with their frequencies is the most common approach of feature extraction.

3.2.3. Bag of words (bow)

Bag-of-words (BOW) is a simple and flexible model to extract features from text documents in order to use in the machine learning classifier models. A bag-of-words indicates the frequency of words within a document. It comprises of two components. One is a vocabulary of known words. Another is a measure of the

occurrence of known words. As words order and structure information in the documents are ignored, the process is known as a "bag" of words. The BOW model keeps tracks of the occurrences of the words in the documents not the position of the word occurrences in the document. Generally, similar documents contain similar contents. Moreover, we can easily perceive the meaning of the documents from the contents contained.

3.2.4. Term frequency-inverse document frequency (TFIDF)

Term Frequency-Inverse Document Frequency (TFIDF) is a popular method used in the Vector Space Model [30]. It is also used in information retrieval including text mining. It statistically measures the significance of a word in the document against the whole corpus. The term frequency is simply calculated in proportion to the number of occurrences a word appears in the document and usually normalized in the positive quadrant between 0 and 1 to eliminate bias towards lengthy documents [31]. To construct the index of terms in TFIDF, punctuation is removed, and all texts are lowercase during tokenization. In addition, there is no stemming on the text and no stopwords are removed.

The first two letter TF or term frequency refers to how important if it occurs more frequently in a document. Therefore, the higher TF reflects the more estimated that the term is significant in respective documents. Additionally, IDF or Inverse Document Frequency calculated on how infrequent a word or term is in the documents. The weighted value is estimated using the whole training dataset. The idea of IDF is that a word is not considered to be a good candidate to represent the document if it is occurring frequently in the whole dataset as it might be the stopwords or common words that are generic. Hence only infrequent words in contrast to the entire dataset are relevant for that document. TF-IDF is calculated using (1).

$$tf\text{–}idf(t,d) = tf(t,d) * idf(t,d) \tag{1}$$

where, $tf(t, d)$ is the raw term frequency, $idf(t, d)$ is the inverse document frequency is calculated as (2).

$$idf(t, d) = \log(n_d/(1+df(d, t)) \tag{2}$$

where, n is the total number of documents and $df(t,d)$ is the number of documents where the term t appears. The 1 addition in the denominator is just to avoid zero terms for terms that appear in all documents and the log ensures that low-frequency term doesn't get too much weight.

3.2.5. Grid search

A grid search is a process of performing hyperparameter tuning in order to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyperparameter values specified [32]. If we work with machine learning, we know what a nightmare it is to stipulate values for hyperparameters. A list of values to choose from should be given to each hyperparameter of the model. We can change these values and experiment more to see which value ranges give better performance. A cross-validation process is performed in order to determine the hyperparameter value set which provides the best accuracy levels.

3.3. Logistic Regression Classifier

Logistic regression is a popular algorithm is similar to linear regression in many cases. It varies from each other based on the application. Basically, linear regression is used for forecasting or predicting purposes whereas logistic regression is greatly used for classification purpose. Instead of a linear activation function, the logistic function is used to handle the case of sample outlier. The schematic diagram for logistic regression is shown in Fig. 3.

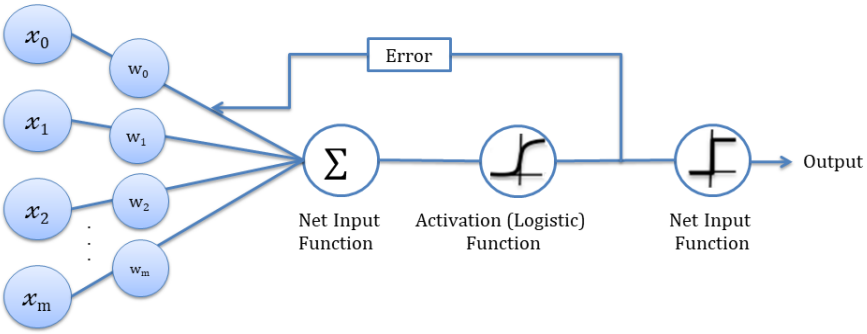


Fig. 3. Schematic diagram for logistic regression classification.

3.3.1. Function logistic (sigmoid) function

To deal with the outlier, logistic function (3) is used to predict the value between two classes, i.e. positive and negative. A linear equation (4) to predict the value of the dependent variable with independent predictors is used in logistic regression algorithm. The predicted value from (4) may vary from negative infinity to positive infinity. But we the algorithm requires the value to be a class i.e. 0 or 1. To achieve the class value, the value from (4) is supplied to (3) and it becomes (5). Equation (5) provides the class value into a range of [0, 1] is shown in Fig. 4.

$$g(x) = 1/(1 + e^{-x}) \tag{3}$$

$$z = \theta_0 + \theta_1 . x_1 + \theta_2 . x_2 + \ldots \tag{4}$$

$$h = g(z) = 1/(1 + e^{-z}) \tag{5}$$

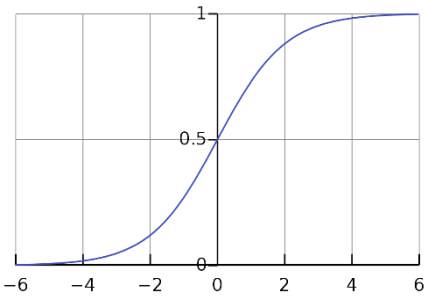


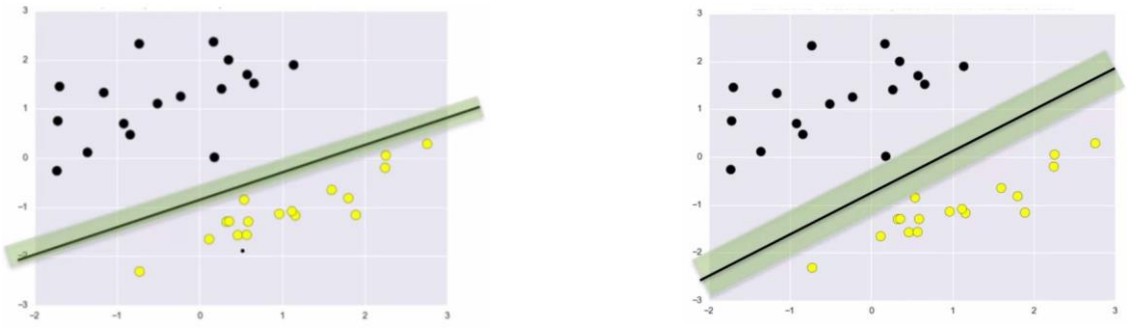
Fig. 4. Sigmoid function.

3.4. Linear Support Vector Classifier

For a binary classification problem, linear support vector classifier performs very well. Linear SVC uses the same linear functional form as logistic regression [33]-[35]. But instead of taking a continuous target value, it takes the output of a linear function and applies the sine function (6) to produce a binary output with two possible values, corresponding to the two possible class labels (positive and negative), where x is the feature vector, w is the weight vector and b is the bias. If the target value is greater than zero the function returns plus one and if it's less than zero, it returns minus one. By applying the simple linear formula, we can produce a class value for any point in this two-dimensional features space.

$$f(x, w, b) = sign\ (x.\ w + b) = sign\ (\sum w[i] \times [i] + b) \tag{6}$$

One way to define a good classifier is to reward classifiers for the amount of separation that can provide between the two classes. The classifier margin is the width that the decision boundary can be increased before hitting a data point. In Fig. 5(a), the classifier margin a very small whereas Fig. 5(b) has a larger classifier margin. We can actually do a search for the classifier that has the maximum margin. This maximum margin classifier is called the Linear Support Vector Machine, also known as an LSVM or a support vector machine with a linear kernel.



(a) (b)

Fig. 5 (a) Classifier margin, (b) Maximum margin classifier.

3.5. Association Rule Mining

To find out significant correlations among datasets, R. Agrawal *et al*. [36] first proposed the concept of Association Rule Mining. Association rule can be expressed as a $X \rightarrow Y$, where $X \in I$ is an item set, $Y \in I$, $Y \notin X$ is a single item and $X, Y \in I$, where I is an item set, $I = i1, i2, i3 \ldots im$. Let, D is the transaction database and for each transaction, $T \in D$, $X \rightarrow Y$ means that if transaction T contains X, there is a possibility that it also includes item Y.

Support, confidence, and lift are used for mining association rule [37]. Support of any item is the frequency that it comes out in the data. Support is calculated using (7).

$$Support\,(X) = P(X) = n(X)/n(D) \tag{7}$$

where $P(X)$ is a probability of event X, $n(X)$ is a number of event X in the transaction, and $n(D)$ is the number of transactions on database D. Confidence is the probability that a consequence will occur if the precedent occurs. Confidence is calculated using (8).

$$Confidence\,(X \rightarrow Y) = P(Y/X) = P(X \cup Y) \,/\, P(X) \tag{8}$$

where $P(Y/X)$ is the conditional probability of happening of Y when X events happened, $P(X \cup Y)$ is a probability of happening of X and Y simultaneously, and $P(X)$ is a probability of happening X. We can determine the robustness of an association rule by inspecting the value of lift ratio which alternatively used of the two parameters. Lift ratio can be expressed as the ratio of the probability of happening of X and Y simultaneously and the probability of happening X and Y expressed in (9).

$$Lift\,(X \rightarrow Y) = P(X \cup Y) \,/\, P(X)\,P(Y) \tag{9}$$

Steps to generate association rule using Apriori algorithm is shown in Fig. 6. While applying the Apriori algorithm the principle must be followed is “the frequent itemsets always contains frequent sub-itemsets [36]. At first, support of 1-itemset is calculated by scanning the transaction databases. We get the frequent 1-itemset by pruning the itemsets whose supports are less than minimum support. Then join operation is performed to generate candidate 2-item sets. The infrequent itemsets from this candidate sets are removed using Apriori property. Again, support of candidate 2-itemsets is calculated by scanning the transaction database. We achieve frequent 2-itemsets by pruning the itemsets whose supports are less than minimum support. This process is repeated until there are no candidate itemsets. Then, generate all possible non-empty subsets of each nonempty frequent itemsets, L. The nonempty subsets are expressed as rule "s -> (1-s)” with confidence value greater than or equal to minimum confidence.

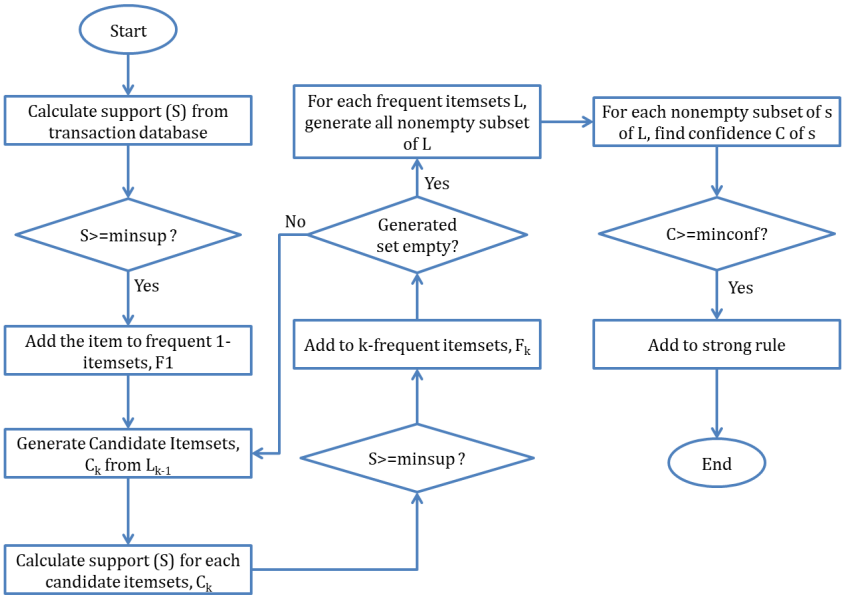


Fig. 6. Steps of association rule mining technique (apriori algorithm).

4. Experimental Result and Discussion

4.1. Experimental Setup and Environment

All experiments have performed on Hewlett-Packard HP Probook 450 G0 with Intel core i-3, 2.40 GHz processor, 4.0 GB DD3 RAM, the 32-bit operating system, Windows 10.1 version in a regular load. Experiments are performed several times and the average value of each experiment are considered for demonstration. We use “Anaconda” [https://www.anaconda.com/distribution/], an open-source Python/R distribution is a flexible way to perform machine learning and data science researches on different operating systems. We implement our idea with a Python-based open-source software, SciPy[https://www.scipy.org/], extensively used for mathematics, science, and engineering [50]. We use some core machine learning packages listed in Table 1. with their functionalities in brief.

4.2. Performance Evaluation Metrics

There are several metrics to measure the performances of a recommender system [38] [39] [40]. T. Zuva et al. provide an extensive survey of techniques, challenges and evaluation metrics related to recommender system [39]. M. Chen and P. Liu categorize performance evaluation systems in three ways i.e. offline analytics,

user study, and online experiment four perspectives (machine learning, information retrieval, human-computer interaction and software engineering) [38]. We have used offline analytics as a performance metric as it requires no human interaction and relatively easy among these three. To perform offline analysis, a dataset describing the real phenomenon of user interaction is needed which should be as far as unbiased.

Table 1. Python Core Packages used in this Research

Packages	Functionality
NumPy	Numpy is used for numerical computation. It defines and performs different operations on the numerical array and matrix types.
Matplotlib	Matplotlib is a commonly used plotting package. It offers a good resolution for 2D plotting and basic 3D plotting as well.
Pandas	Pandas is used to handle different types of data structures.
scikit-learn	Collection of algorithms and tools for machine learning.
nltk	NLTK is a preeminent natural language processing package to work with human natural language. It is enriched with more than 50 corpora. It also serves the lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries [https://www.nltk.org/].
Apyori	Apyori package provides an API for Apriori algorithm with Python 2.7 and 3.3 - 3.5 with a command-line interface [https://pypi.org/project/apyori/].

4.2.1. K-fold Cross-validation

Cross-validation is a computational technique to measure the accuracy of a predictive model while the dataset size is smaller. Cross-validation method has fewer biases than other approaches. In this method, a portion of data is used in the training process and the rest of the data is used for validation purpose. In the K-fold cross-validation method, the data is partitioned into k subsample, approximately equal size. One portion from k folds is used as test/validation set and rest k – 1 folds are fitted into the model. This process continues for k times and the mean squared error, MSE (10) is calculated for each of the iterations. So, the test error, MSE_1, MSE_2, . . . ,MSE_k are calculated using (11) [41]. Finally, the k-fold CV is calculated by taking an average of these values.

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i \tag{10}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{11}$$

where $\hat{f}$ provides the predicted value for i-th iteration.

4.2.2. Accuracy, precision, recall and f-1 score

Accuracy, Precision, Recall and F-1 score are very popular metrics to measure the performance of a predictive model [42], [40]. These values can be calculated from a matrix is called the confusion matrix shown in Table 2. To present the performance of the classification model on a set of test data with known true or false values. The correctly predicted classes in this table are the true positive and true negative shown in green. The false positives and false negatives marked with red color are inaccurately predicted classes.

Table 2. Confusion Matrix.

	Predicted Class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

True Positives (TP) - True positive is appropriately predicted positive value means that both actual class and predicted class values are "yes".

True Negatives (TN) - True negative is appropriately predicted negative value means that both actual class and predicted class values are "no".

False Positives (FP) - False positive is mistakenly predicted positive value indicates that the actual class value is negative (no) but the model predicted as positive (yes).

False Negatives (FN) - False negative is mistakenly predicted negative value indicates that the actual class value is positive (yes) but the model predicted as negative (no).

Accuracy, Precision, Recall, and F1 score are calculated using these four parameters (*TP*, *TN*, *FP*, and *FN*).

Accuracy - The most instinctively used measurement accuracy can be defined as the ratio (12) of correctly predicted class values (*TP* and *TN*) and total predictions. The higher the accuracy value indicates the soundness of the classification model. But it works well only for the case of symmetric datasets and requires equal values of false positive and false negatives.

$$\text{Accuracy} = TP+TN/TP+FP+FN+TN \tag{12}$$

Precision - Precision is the proportion (13) of accurately observed positive assumptions for the positive overall predicted measurements. The relationship between false positive and precision is inversely proportional.

$$\text{Precision} = TP/TP+FP \tag{13}$$

Recall (**Sensitivity**) - Recall can be defined as the ratio (14) of appropriately anticipated positive observations towards all actual class observations - yes. The relationship between false negative and precision is also inverse proportional.

$$\text{Recall} = TP/TP+FN \tag{14}$$

F1 score - F1 Score is the compromisation between Precision and Recall and calculated by taking the weighted average of these two measures (15). When the datasets are symmetric, Accuracy is the best metric to measure the performance. But F1 score is more valuable where the dataset's classes are uneven.

$$\text{F1 Score} = 2*(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision}) \tag{15}$$

4.2.3. Receiver operating curve (ROC) and area under curve (AUC)

Receiver Operating Characteristic (ROC), a signal detection analysis tool. is extensively used to judge the performance of machine learning techniques including information retrieval [43]. The ROC curve is demonstrated as a Cartesian graph with a false positive rate (FPR) in X-axis and the true positive rate (TPR) in Y-axis. ROC curve is the visualization of the relationship between TPR and FPR (Fig. 7). The parameters of the recommender system can be tuned through the ROC curve which is a compromisation between the TPR and FPR [38]. Alternatively, the Area Under the ROC curve (AUC) is also used to evaluate the performance of different recommender systems.

4.3. Results Analysis

In this section, we have analyzed the experimental results to support our claims. We have justified our classification model using K-fold cross-validation. We have calculated the model's prediction accuracy using

Precision, Recall, and F-Score for different folds of datasets and variations in dataset size. We have also calculated ROC and AUC for classification models. We show the recommendation accuracy without sentiment analysis on the client's feedback, with logistic regression and linear support vector classifier.

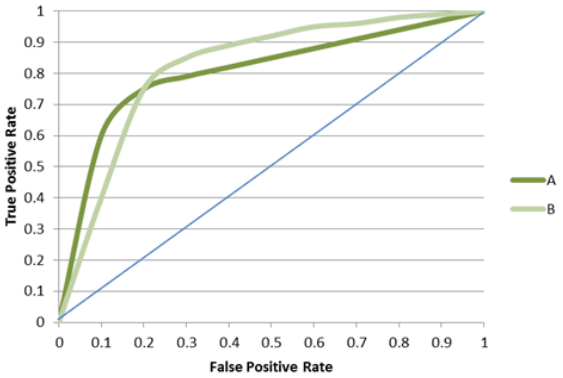


Fig. 7. ROC Curve [38].

Table 3. Cross-Validation Accuracy

K	Logistic Regression Classifier		Linear Support Vector Classifier	
	Training	Testing	Training	Testing
5	0.875	0.923	0.875	0.923
6	.0875	0.923	0.875	0.923
7	.0875	0.923	0.875	0.923
8	.0875	0.846	0.875	0.923
9	.0875	0.846	0.875	0.923
10	.0875	0.846	0.875	0.923
Avg.	**0.875**	**0.885**	**0.875**	**0.923**

4.4. Results Analysis

In this section, we have analyzed the experimental results to support our claims. We have justified our classification model using K-fold cross-validation. We have calculated the model's prediction accuracy using Precision, Recall, and F-Score for different folds of datasets and variations in dataset size. We have also calculated ROC and AUC for classification models. We show the recommendation accuracy without sentiment analysis on the client's feedback, with logistic regression and linear support vector classifier.

4.4.1. K-Fold cross validation accuracy

We evaluate the quality of our classifier model using K-fold cross-validation. We have calculated training and testing/validation accuracy for logistic regression and linear support vector classifier by the varying value of k are listed in Table 3. Average training accuracy (88%) for both classifiers is same but testing accuracy for linear support vector classifier (93%) is greater than logistic regression classifier. From Table 3, we can see that for small k values (5, 6, 7), accuracy for both models is good. So, we use k=5 for our proposed model. K-fold cross-validation is very useful to handle overfitting and underfitting of the classifier model.

4.4.2. Model training and testing time

Time requirements for both classifiers are shown in Fig. 8 and Fig. 9. Fig. 8 demonstrates the model training time and Fig. 9 shows the testing time for logistic regression and support vector classifier respectively. Model training time for linear support vector classifier is greater than logistic regression classifier because SVM is more complex than logistic regression. Small k values, testing time for both models are similar. The

time required for SVM is sharply increased for SVM but on the other hand, it drastically decreased for logistic regression for k values larger than 7.

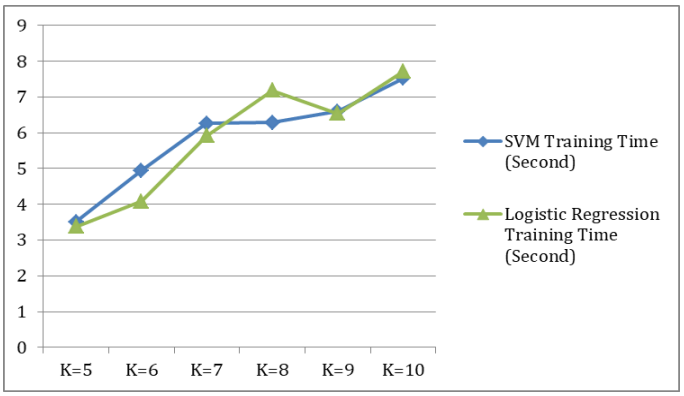


Fig. 8. Training time requirements.

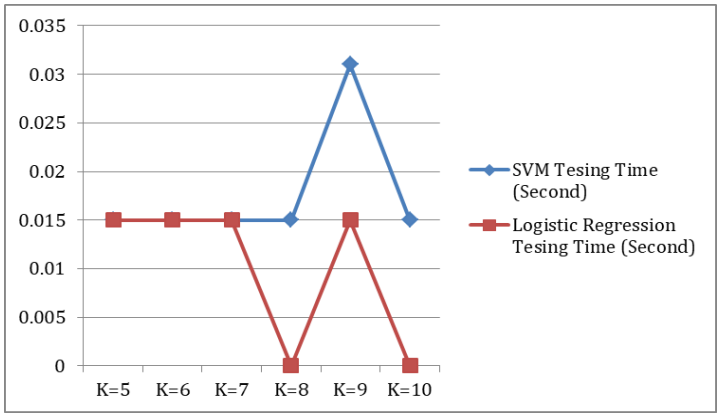


Fig. 9. Testing time requirements.

4.4.3. Prediction accuracy for the different splitting of dataset

We have calculated the Precision, Recall, and F-Score values listed in Table 4 by splitting dataset in different training and testing proportions considering K=5. From Fig. 10, we can see that Precision, Recall, and F-Score values for both models are higher. Although the average precision and recall value of SVM is larger than logistic regression, the average F-score value is opposite.

Table 4. Precision, Recall, and F-Score for Different Dataset Splitting

Training	Logistic Regression Classifier			Support Vector Classifier		
	Precision	Recall	F-Score	Precision	Recall	F-Score
30	0.78	0.79	0.79	0.78	0.79	0.79
40	0.78	0.79	0.79	0.77	0.81	0.77
50	0.78	0.79	0.79	0.87	0.84	0.79
60	0.64	0.80	0.71	0.63	0.76	0.69
70	0.71	0.84	0.77	0.70	0.79	0.74
80	0.93	0.92	0.91	0.93	0.92	0.91
Avg.	**0.77**	**0.82**	**.079**	**0.78**	**.082**	**.078**

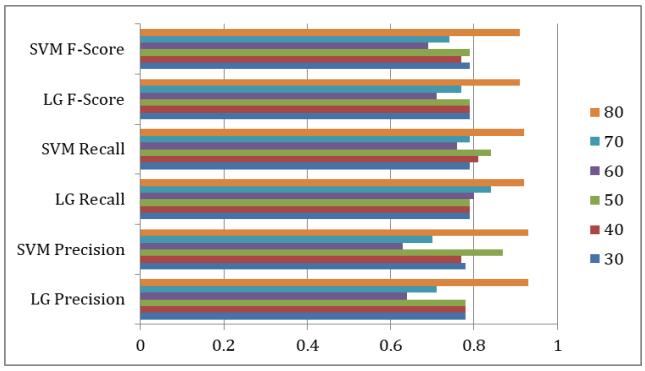


Fig. 10. Prediction accuracy for a different splitting of dataset.

4.4.4. Prediction accuracy for client's feedback analysis for different K values

The Precision, Recall, and F-Score for different K values are listed in Table 5 considering 80% data for training and 20% data for testing. The Precision, Recall, and F-Score values for support vector machine are the same for different K values [Fig. 11]. In the case of logistic regression, these values are decreased with the increase in the K values.

Table 5. Precision, Recall, and F-Score for different K Values

K	Logistic Regression Classifier			Support Vector Classifier		
	Precision	Recall	F-Score	Precision	Recall	F-Score
5	0.93	0.92	0.91	0.93	0.92	0.91
6	0.93	0.92	0.91	0.93	0.92	0.91
7	0.93	0.92	0.91	0.93	0.92	0.91
8	0.72	0.85	0.78	0.93	0.92	0.91
9	0.72	0.85	0.78	0.93	0.92	0.91
10	0.72	0.85	0.78	0.93	0.92	0.91

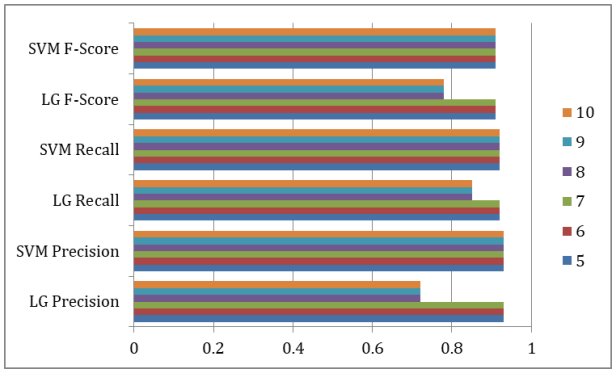


Fig. 11. Prediction accuracy for different K values.

4.4.5. ROC and AUC for logistic regression and SVM

ROC curves for logistic regression and linear support vector classifier for different K values are shown in Fig. 12 and 13. We also calculate AUC and plot in the same figures. From Fig. 12 and, it is clear that the performance of support vector classifier is better than the logistic regression model.

4.4.6. Job recommendation accuracy

The accuracy of job recommendation to freelancers is listed in Table 6. We have calculated accuracy three cases: without applying sentiment analysis to the client's feedback on completed jobs, sentiment analysis

with logistic regression and sentiment analysis with support vector classifier. For each case, we have calculated recommending accuracy for five users. Finally, we got an average of five users to compare the results with other cases. From Table 6, we can see that recommendation accuracy for linear support vector classifier (84.03%) is higher than the other two cases.

Table 6. Recommendation Accuracy Comparison

User No.	Without Sentiment Analysis	Sentiment Analysis with Logistic Regression	Sentiment Analysis with Support Vector
1	77.32%	83.50%	83.75%
2	76.50%	85.24%	84.60%
3	78.60%	82.33%	83.80%
4	80.20%	84.50%	85.36%
5	76.45%	81.45%	82.65%
Avg.	**77.4%**	**83.40%**	**84.03%**

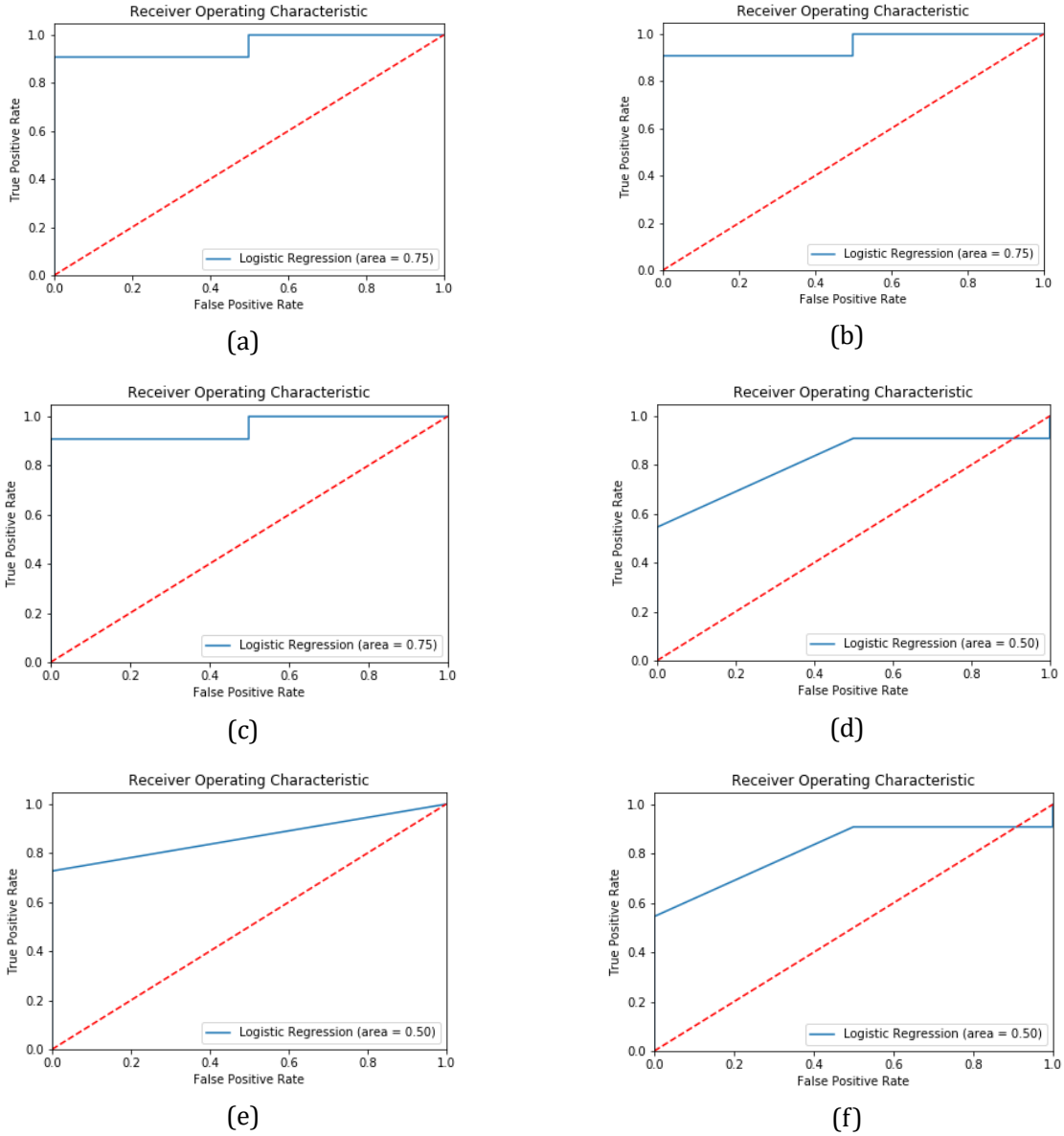


Fig. 12. ROC Curves for different k for Logistic Regression, (a) CV=5, (b) CV=6, (c) CV=7, (d) CV=8, (e) CV=9, (f) CV=10.

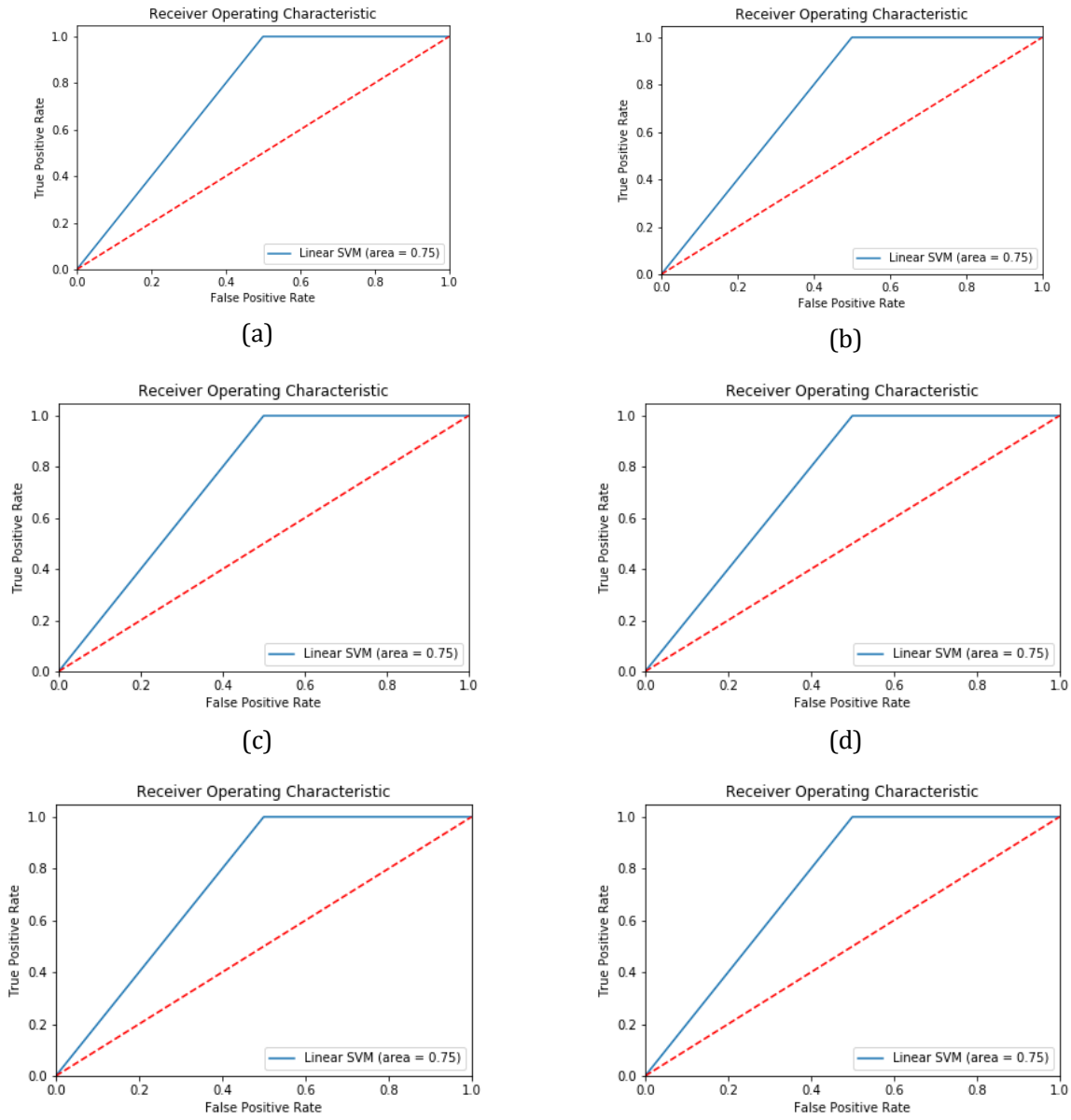


Fig. 13. ROC Curves for different k for Support Vector Machine, (a) CV=5, (b) CV=6, (c) CV=7, (d) CV=8, (e) CV=9, (f) CV=10.

5. Conclusion and Future Research Directions

In this paper, we have proposed an intelligent system to recommend appropriate jobs to freelancers from different online marketplaces. We have collected the client's feedback and train our classification model with logistic regression and linear support vector machine model to classify the posted jobs to positive and negative. Linear SMV outperforms than logistic regression in the metric cross-validation. Then, frequent skillsets used in completed jobs of freelancers are found out using Apriori association rule mining algorithm. A possible job list is created to the freelancers by matching these frequent skillsets with the skills required in the posted jobs. Finally, appropriate jobs are recommended a filtering algorithm. We got recommendation accuracy **77.4%** without sentiment analysis on client's feedback, **83.4%** with logistic regression and **84.03%** with linear support vector classifier. Our proposed smart system will not only help the freelancers find their suitable jobs easily but will also assist the requesters in assigning jobs to suitable candidates.

Personality types of the workers have not yet considered for assigning the job this research. This work can be extended by applying classification algorithms to the posted jobs. The efficiency of the proposed system can also be enhanced by considering the skills in recent jobs. Recommending jobs by taking re-hire in consideration might also improve the recommendation as well.

References

[1] Kassi, O., & Lehdonvirta, V. (2016). Online labour index: Measuring the online gig economy for policy and research.

[2] Upwork, F. (2017). *Union, Freelancing in America*, 1–8.

[3] Benchmark, G. (2018). The payoneer freelancer income survey global benchmark report for hourly rates. 1–18.

[4] Parvathi, R., & Kommineni, M. (2013). Risk analysis for exploring the opportunities in cloud outsourcing.

[5] Kokkodis, M., Papadimitriou, P., & Ipeirotis, P. G. (2015). Hiring behavior models for online labor markets. *Proceedings of the Eighth ACM Int. Conf. Web Search Data Min*.

[6] Hossain, M. S., & Arefin, M. S. (2019). Freelancer's work history and client's review data set.

[7] Yuen, M., King, I., & Leung, K. (2011). A survey of crowdsourcing systems. *Proceedings of the IEEE Int. Conf. Privacy, Secur. Risk, Trust. IEEE Int. Conf. Soc. Comput*.

[8] Hossain, M. S., & Arefin, M. S. (2019). An intelligent system to generate possible job list for freelancers. *Proceedings of the International Conference on Advancements in Computing & Management*.

[9] Yuen, M. C., King, I., & Leung, K. S. (2011). Task matching in crowdsourcing, *Proceedings of the 2011 IEEE Int. Conf. Internet Things Cyber, Phys. Soc. Comput*.

[10] Yuen, M. C., & Irwin, K. (2012). Task recommendation in crowdsourcing systems, *Proceedings of the First Int. Work. Crowdsourcing Data Min*.

[11] Yuen, M. C., King, I., & Leung, K.-S. (2015). TaskRec: A task recommendation framework in crowdsourcing systems. *Neural Process. Lett.*, *41(2)*, 223–238.

[12] Chilton, L. B., Horton, J. J., Miller, R. C., & Azenkot, S. (2010). Task search in a human computation market categories and subject descriptors. *Kdd-Hcomp*, 1–9.

[13] Ambati, V., Vogel, S., & Carbonell, J. (2011). Towards task recommendation in micro-task Markets. *Hum. Comput*.

[14] Schnitzer, S., Neitzel, S., Schmidt, S., & Rensing, C. (2016). Perceived task similarities for task recommendation in crowdsourcing systems. *Proceedings of the 25th Int. Conf. Companion World Wide Web - WWW '16 Companion*

[15] Schnitzer, S., Rensing, C., & Schmidt, S. (2015). Demands on task recommendation in crowdsourcing platforms - the worker's perspective. *Proceedings of the RecSys CrowdRec 2015 Crowdsourcing Hum. Comput. Recomm. Syst*.

[16] Yang, Y., Karim, M. R., Saremi, R., & Ruhe, G. (2016). Who should take this task?: Dynamic decision support for crowd workers. *Proceedings of the 10th ACM/IEEE Int. Symp. Empir. Softw. Eng. Meas*.

[17] Safran, M., & Che, D. (2017). Real-time recommendation algorithms for crowdsourcing systems. *Appl. Comput. Informatics*, *13(1)*, 47–56.

[18] Tunio, M. Z. *et al.*, (2017). Impact of personality on task selection in crowdsourcing software development: A sorting approach, *IEEE Access*.

[19] Tunio, M. Z., Luo, H., Wang, C., & Zhao, F. (2018). Task assignment MODEL for crowdsourcing software development : TAM. *J. Inf. Process. Syst.*, *14(3)*, 621–630.

[20] apretz, L. F., & Ahmed, F. (2010). Making sense of software development and personality types. *IT Prof.*, *12(1)*, 6–13.

[21] Kazai, G., Kamps, J., & Milic-Frayling, N. (2011). Worker types and personality traits in crowdsourcing relevance labels. *Proceedings of the 20th ACM Int. Conf. Inf. Knowl. Manag*.

[22] Demirörs, O. (2013). A case study on the need to consider personality types.

[23] Capretz, L. F., Varona, D., & Raza, A. (2015). Influence of personality types in software tasks choices. *Comput. Human Behav*.

[24] Abhinav, K., Dubey, A., Jain, S., Virdi, G., Kass, A., & Mehta, M. (2017). Crowd advisor: A framework for freelancer assessment in online marketplace. *Proceedings of the 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. Softw. Eng. Pract. Track*.

[25] Difallah, D. E., Demartini, G., & Cudré-mauroux, P. (2013). Pick-A-crowd: Tell me what you like, and I'll tell you what to do.

[26] Borisyuk, F., Zhang, L., & Kenthapadi, K. (2017). LiJAR: A system for job application redistribution towards efficient career marketplace. *Proceedings of the 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min*.

[27] Wang, W., He, Z., Shi, P., Wu, W., & Jiang, Y. (2016). Truthful team formation for crowdsourcing in social networks.

[28] Tran-Thanh, L., Huynh, T. D., Rosenfeld, A., Ramchurn, S., & Jennings, N. (2014). BudgetFix : Budget limited crowdsourcing for interdependent task allocation with quality guarantees. *Proceedings of the IFAAMAS/ACM Int. Conf. Auton. Agents Multi-agent Syst*.

[29] Zheng, A., Hong, Y., & Pavlou, P. A. (2016). Matching in two-sided platforms for IT services: Evidence from online labor markets. *Ssrn*, 1–38.

[30] Nazirul, W., Wan, H., Nur, N., & Sjarif, A. (2018). SMS spam classification using vector space model and artificial neural network

[31] He, B., & Ounis, I. Term frequency normalisation tuning for BM25 and DFR models.

[32] Brito, J. A. A., McNeill, F. E., Webber, C. E., & Chettle, D. R. (2005). Grid search: An innovative method for the estimation of the rates of lead exchange between body compartments. *J. Environ. Monit.*, *7(3)*, 241–247.

[33] Ben, H. A., & Weston, J. (2010). A user ' s guide to support vector machines preliminaries: Linear classifiers. *Humana Press. a part Springer Sci. Media*.

[34] Schölkopf, B. (2003). An introduction to support vector machines. *Recent Advances and Trends in Nonparametric Statistics*.

[35] Weston, J. (2005). Support vector machine tutorial. *Bioinformatics*.

[36] Agrawal, R., Swami, A., & Imielinski, T. (1993). Database mining: A performance perspective. *IEEE Trans. Knowl. Data Eng.*,

[37] Han, J. P. J. (2011). Micheline kamber, *Data Mining – Concepts & Techniques*.

[38] Chen, M., & Liu, P. (2017). Performance evaluation of recommender systems. *Int. J. Performability Eng.*, *13(8)*, 1246–1256.

[39] Zuva, T., Ojo, S. O., Ngwira, S. M., & Zuva, K. (2012). A survey of recommender systems techniques, challenges and evaluation metrics. *Int. J. Emerg. Technol. Adv. Eng.*

[40] Koehrsen, W. (2018). Beyond accuracy: Precision and recall. Retrieved from: https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

[41] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning with *Applications* in *R*,.

[42] Martin, D., & Ward, P. (2011). Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation.

[43] Fawcett, T. (2006). An introduction to ROC analysis, *Pattern Recognit. Lett.*, *27(8)*, 861–874.

Sabir Hossain received his bachelor degree in computer science and engineering from Chittagong University of Engineering & Technology (2015) with an outstanding result. He is now perusing the M.Sc. degree in computer science and engineering from the same university. His research interests are data mining, machine learning, big data, and SOFTWARE engineering. He is currently working as a faculty member in the same department of his university.

Mohammad Shamsul Arefin received his B.Sc. engineering in computer science and engineering from Khulna University, Khulna, Bangladesh in 2002, and completed his M.Sc. engineering in computer science and engineering in 2008 from Bangladesh University of Engineering and Technology (BUET), Bangladesh. He has completed his Ph.D. from Hiroshima University, Japan. He is currently working a professor and the head in the Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh. His research interest includes data privacy and data mining, Cloud computing, big data, IT in agriculture, and OO system development.