Exploring the impact of machine learning algorithms with unlabelled data

Anonymous

1 Introduction

Predicting salary based on job descriptions is a challenging task in the field of natural language processing and machine learning. In the current digital age, many recruiters seek to find suitable candidates through multiple channels — e.g., online job portals, professional networks — as well as traditional avenues, such as word of mouth and mass media (Shenoy and Aithal, 2018).

The dataset is derived from the large dataset called *mycareersfuture* (Bhola et al., 2020). The dataset has a total of 17377 data, consisting of 13902 train data, 1738 validation data, and 1737 test data. The dataset is shown in the table 1:

Table 1: Dataset Information

Data Type	Labeled	Unlabeled	Total
Train	8000	5902	13902
Validation	1738	-	1738
Test	-	1737	1737
Total	9738	7639	17377

The distribution of salary bin is shown in the figure 1. We observe that the salary bin distribution exhibits an uneven and imbalanced pattern, which may potentially affect the performance of the machine learning algorithms.

To answer the question "Does Unlabelled data improve Job salary prediction?", We will analyse and compare the performance of different machine learning algorithms for this dataset (labelled and unlabelled data) and finally explore whether unlabelled data can be effectively combined to increase the performance of the model.

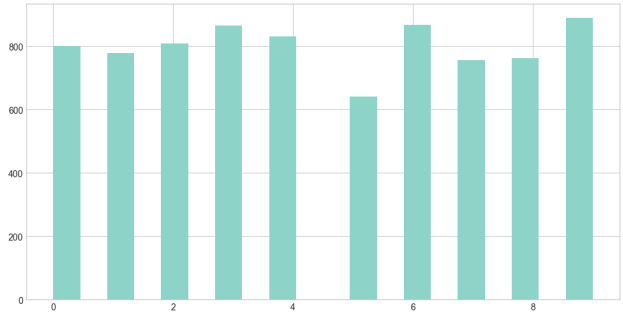


Figure 1. Salary Bin Distribution

2 Literature review

In predicting salary based on job descriptions, there are many studies that have been conducted. mycareersfuture dataset with job descriptions and their corresponding skills labels is presented by (Bhola et al., 2020).

Many methodology are proposed to predict salary based on job descriptions, such as BERT-XMLC (Bhola et al., 2020), framework for comprehensively evaluating the performance of debiasing methods (Han et al., 2022).

Many traditional machine learning algorithms are also mentioned in this paper, such as kNN (Zhang and Cheng, 1908), Decision Tree (Dutta et al., 2018), Naive Bayes (Mani et al., 1997) and so on.

3 Methods

In this study, we adopt two feature representations from the raw job descriptions.

- **TF-IDF**: We compute the TF-IDF vectors for job descriptions using the method proposed by (Manning et al., 2008). This method captures the importance of terms within a document and across the entire corpus.

- **Embedding**: We adopt the pretrained Sentence Transformer model (Reimers and Gurevych, 2019) to obtain word embeddings for the job descriptions. These embeddings provide semantic representation for the text data.

Through these two features, we explore three different machine learning algorithms paradigms: Supervised learning, Unsupervised learning and Semi-supervised learning.

To find the parameters which can lead the highest performance of the machine learning algorithms, we consider to use some search strategies. Due to a high dimentions of the features, we adopt Grid search here because Grid search can suffer from high dimensional spaces (Liashchynskyi and Liashchynskyi, 2019).

We train the models between TFIDF and Embedding features, but we will only choose TFIDF features to analyse the results and evaluate the models. To evaluate classifiers, we use the accuracy and F_1 score (3) that combines recall (2) and precision (1) in the following way: (Tan, 2006)

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3}$$

3.1 Supervised learning

In the supervised learning part, we adopt 9 different machine learning algorithms to predict the salary bin.

3.1.1 KNN Classifier

We decide to use k-Nearest-Neighbours (kNN) as our baseline model, because the k-Nearest-Neighbours is a simple but effective method for classification (Guo et al., 2003).

To ensure what weights and p value in KNN classifier lead to a better performance, we set the parameters of Grid search as follows:

- **k**: 1 - 11
- **p**: 1, 2
- **weights**: uniform, distance

Using Grid search, we find that in TF-IDF, using kNN algorithm's best accuracy is 18.77%. In Embedding, using kNN algorithm's best accuracy is 23.95%. The best parameters are shown in the table 2.

Features	k	p	weights	Accuracy
TF-IDF	3	2	distance	18.77
Embedding	3	2	distance	23.95

Table 2: Best accuracies and parameters for kNN algorithm using TF-IDF and Embedding

In the experiment, we found that K is the most important factor in kNN algorithm. So we keep increasing k's value to 200. And set p equal to 2 and weight is "distance". The output is shown in figure 2.

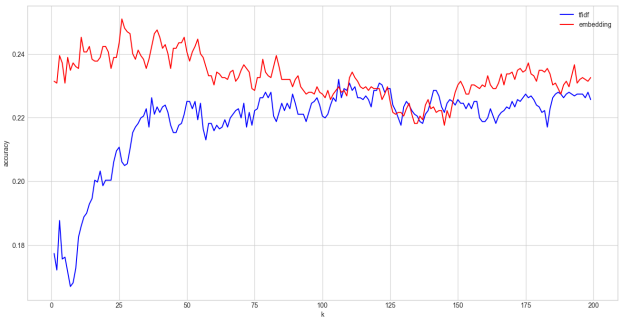


Figure 2. KNN Classifier

From the results, we can when the k value is 106, the accuracy for tfidf is 23.20% which is the best accuracy for tfidf. For embedding data, the best accuracy is 25.1% when k is 26.

3.1.2 Decision Tree Classifier

Decision tree classifier is an effient supervised learning algorithm. It focuses on generating classification rules displayed as decision trees that is deduced or concluded from a group of disorder and irregular instances. (Dutta et al., 2018) During the experiment, we adopted the decision tree classifier on the given dataset to see the performance.

We set 5 parameters during the experiment as follows:

- **criterion**: gini, entropy
- **max_depth**: 5, 10, 15
- **min_samples_split**: 2, 5, 10

- **min_samples_leaf**: 1, 2, 5
- **splitter**: best, random

After using Grid search, we find that in TF-IDF, using decision tree classifier's best accuracy is 20.38%. In embedding, using decision tree classifier's best accuracy is 18.77%. The best parameters are shown in the table 3.

Method	Parameters	Accuracy
TF-IDF	criterion: gini max_depth: 15 min_samples_split: 10 min_samples_leaf: 2 splitter: random	20.38
Embedding	criterion: entropy max_depth: 5 min_samples_split: 2 min_samples_leaf: 1 splitter: random	18.77

Table 3: Best accuracies and parameters for decision tree classifier using TF-IDF and Embedding

3.1.3 Naive Bayes Classifier

Naive Bayes (Mani et al., 1997) is a type of classifier based on probability. We adopted two types of Naive Bayes classifier: BernoulliNB (4) and GaussianNB (5).

$$P(x_i|y) = P(x_i = 1|y)x_i + (1 - P(x_i = 1|y))(1 - x_i) \quad (4)$$

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}) \quad (5)$$

For naive Bayes classifier, we do not need to set parameters (as var_smoothing does not influence a lot). The accuracy is shown in the table 4.

Table 4: Accuracy of Naive Bayes Models

Model	TF-IDF	EMD
GaussianNB	21.99	22.91
BernoulliNB	21.47	21.3

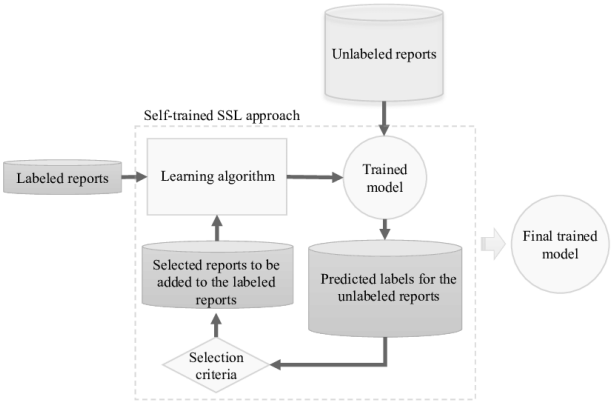


Figure 3. Self-trained semi-supervised learning architecture (Hassanzadeh et al., 2018)

From the table 4, we can see that the accuracy of GaussianNB is higher than BernoulliNB.

3.1.4 Other Classifier

In the experiment, we also tried other supervised classifiers such as SVM (Rejani and Selvi, 2009), Adaboost and some ensemble methods. The results are shown in the table 7. Of these, the SVM model got the best accuracy of 26.89% in embedding features.

3.2 Unsupervised learning

Due to there has many unlabeled data in the dataset, we also tried to use unsupervised learning to train the model. However, the result is unsatisfactory.

In the experiment, we used K-means clusting algorithm and Gaussian mixture model (GMM) algorithm to train the model. The accuracy is shown in the table 5.

Table 5: Accuracy of Unsupervised Models

Model	TF-IDF	Embedding
K-means	15.66	16.41
GMM	11.23	11.51

3.3 Semi-supervised learning

For semi-supervised learning, we adopted self-training method (Hassanzadeh et al., 2018). The architecture is shown in figure 3.

The method of self-training is to use the labeled data to train the model, and then use the trained model to predict the unlabeled data's label. If the probability of the predicted label

Table 6: F_1 scores of all models

Method	**Supervised**		**Semi-supervised**	
	TF-IDF	**Emb**	**TF-IDF**	**Emb**
KNN	0.2118	0.2108	0.1042	0.2194
DT	0.1602	0.1678	0.1377	0.1597
GNB	0.1994	0.2082	0.1884	0.2114
BNB	0.1778	0.1908	0.1761	0.2038
Adaboost	0.1765	0.1651	0.1880	0.1779
DT+Adaboost	0.2116	0.1934	0.1849	0.1905
GNB+Adaboost	0.1911	0.1605	0.1154	0.1263
SVM	**0.2466***	**0.2616***	**0.2333***	**0.2389** *

Table 7: Accuracy of all models

Model	TFIDF		Embedding	
	Labaled data	Unlabeled data	Labaled data	Unlabaled data
KNN	18.77	20.21	23.95	25.1
GNB	21.99	21.12	22.91	21.99
BNB	21.47	20.43	21.3	20.84
Adaboost	22.39	20.43	21.7	19.8
Decision Tree (DT)	20.38	20.03	16.29	17.9
Adaboost + DT	22.22	19.86	21.7	23.2
Adaboost + GNB	21.99	10.99	17.67	13.82
Adaboost + BNB	22.51	10.99	21.53	14.91
SVM	**25.62***	**24.64***	**26.89***	**24.87***
K-means	-	**15.66***	-	**16.41***
GMM	-	11.23	-	11.51

Note: The ublabeled data of supervised learning means using the self-training method.

is greater than the threshold, then we add the unlabeled data with its predicted labels to the training set and retrain the model.

On the basis of this, we using the unlabeled data by iteratively training the model and selecting the high-confidence (greater than the threshold). We set two conditions for stopping the iteration:

- There is no high-confidence samples in the unlabeled dataset. (high-confidence means the probability of the predicted label is greater than the threshold)
- There is no improvement after iterations. The moethod keep tracking the best accuracy that is achieved by the model on the validation set. If the accuracy does not improve of 3 iterations, the method will stop. This helps avoid the overfitting and saves compute resources and time.

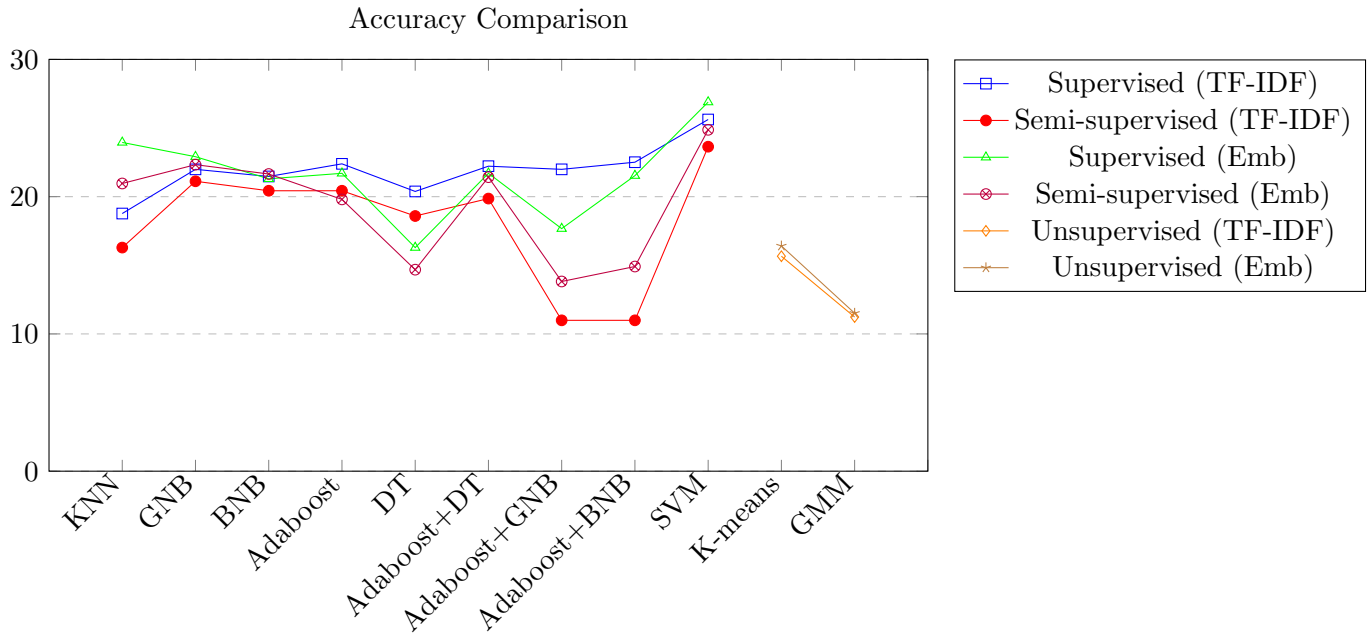
The goal is to improve the performance of the model by using the unlabeled data.

In the experiment, we define two self-trained models, one with the threshold as 0.8, and one without threshold. We dicover that self-trained model can not improve the performance of the model. The accuracy is shown in the table 7.

We discover that the self-trained method can not improve much performance of the model, some accuracy have even decreased.

4 Results

In this section, we present the results of our experiments. We evaluate the performance of our models by using the F_1 score and accuracy.



The results which shown in the table 7 and table 6 are the results of models with best parameters after adopting grid search.

Table 6 shows the F_1 scores of all models, and table 7 shows the accuracy of all models.

In the experiment, we adopt the wandb to record the our runs results. The detailed results of the experiment will be published in the wandb reporepository [1].

5 Discussion

Through the experiment, we discover that unlabelled data does not shows an improvement after adopting self-trained method.

We test supervised learning, unsupervised learning and semi-supervised learning. However, the accuracy is about 20%. Of these, the SVM model got the best accuracy of 26.89% and the best F_1 score 0.2616. which increase 1.27% compared with the baseline model (kNN).

After I plot the probability estimates by using the predict_proba method in sklearn, I discover that the probability of the predicted labels do not have a high probability (most of them are below 0.4).

So, when I apply all the unlabled data with its predicted labels to the training set, the data who has a low probability sometimes get an incorrect predicted label. This is a reason that make the accuracy and F_1 score decrease.

After that, we search if there has a method to avoid this probelm. We find that we can set a threshold to filter the data who has a low probability. (Hassanzadeh et al., 2018) We set the threshold as 0.8, however, it still not improve the performance.

6 Conclusions

In this study, we explore the performance of various machine learning models on the task of predicting salary based on job descriptions using supervised, unsupervised and semi-supervised learning algorithms.

Our result indicate that the overall performance of the models is not satisfactory, with approximately 20% for most models. Among the models, we find that the SVM model got the best accuracy of 26.89% and the best F_1 score 0.2616.

In semi-supervised learning, we observe that the incorporation of unlabeled data using trained model does not consistently improve the performance. After checking the predicted probabilities generated by the predict_proba method in scikit-learn, we find that the probability of the predicted labels do not have a high

[1] Due to it is anonymous, it can not be shown now.

probability (most of them are below 0.4). As a result, when we add all the unlabled data with its predicted labels to the training set, the data who has a low probability sometimes get an incorrect predicted label. This is a reason that make the accuracy and F_1 score decrease.

To address this probelm, We explore the possibility of setting a threshold to filter out instances with low probability estimates. following the approach proposed by (Hassanzadeh et al., 2018). By iteratively training the model and selecting the high-confidence (greater than the threshold) unlabeled data, which is effiently

References

Bhola, A., Halder, K., Prasad, A., and Kan, M.-Y. (2020). Retrieving skills from job descriptions: A language model based extreme multi-label classification framework. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5832–5842, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Dutta, S., Halder, A., and Dasgupta, K. (2018). Design of a novel prediction engine for predicting suitable salary for a job. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 275–279.

Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. (2003). Knn model-based approach in classification. In Meersman, R., Tari, Z., and Schmidt, D. C., editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 986–996, Berlin, Heidelberg. Springer Berlin Heidelberg.

Han, X., Shen, A., Cohn, T., Baldwin, T., and Frermann, L. (2022). Systematic evaluation of predictive fairness. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 68–81, Online only. Association for Computational Linguistics.

Hassanzadeh, H., Kholghi, M., Nguyen, A., and Chu, K. (2018). Clinical document classification using labeled and unlabeled data across hospitals.

Liashchynskyi, P. and Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: A big comparison for nas.

Mani, S., Pazzani, M. J., and West, J. (1997). Knowledge discovery from a breast cancer database. In Keravnou, E., Garbay, C., Baud, R., and Wyatt, J., editors, *Artificial Intelligence in Medicine*, pages 130–133, Berlin, Heidelberg. Springer Berlin Heidelberg.

Manning, C., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. An Introduction to Information Retrieval. Cambridge University Press.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Rejani, Y. I. A. and Selvi, S. T. (2009). Early detection of breast cancer using svm classifier technique.

Shenoy, V. and Aithal, S. (2018). Literature review on primary organizational recruitment sources. *International Journal of Management, Technology, and Social Sciences*, pages 37–58.

Tan, S. (2006). An effective refinement strategy for knn text classifier. *Expert Systems with Applications*, 30(2):290–298.

Zhang, J. and Cheng, J. (2019/08). Study of employment salary forecast using knn algorithm. In *Proceedings of the 2019 International Conference on Modeling, Simulation and Big Data Analysis (MSBDA 2019)*, pages 166–170. Atlantis Press.