Machine Learning Models for Salary Prediction Dataset using Python

Reham Kablaoui
Computer Engineering Department
Kuwait University
Khaldiya, Kuwait
reham.kablaoui@ku.edu.kw

Ayed Salman
Computer Engineering Department
Kuwait University
Khaldiya, Kuwait
ayed.salman@ku.edu.kw

2022 International Conference on Electrical and Computing Technologies and Applications (ICECTA) | 978-1-6654-5600-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICECTA57148.2022.9990316

Abstract— **In today's world, salary is the primary source of motivation for many regular employees, which makes salary prediction very important for both employers and employees. It helps employers and employees to make estimations of the expected salary. Fortunately, technological advancements like Data Science and Machine Learning (ML) have made salary prediction more realistic. In this paper, we exploit the benefits of data science to collect a 20,000+ dataset of salaries in the USA. We then apply three supervised ML techniques to the obtained datasets to produce salary prediction. The learning models are linear regression, random forest, and neural networks. The output of the three models is analyzed and compared to show the following; neural network outperforms the other ML models for better accuracy with accuracy level 83.2%, and linear regression has the fastest time of 0.363s for training the model.**

Keywords—Linear Regression, Machine Learning, Neural Networks, Random Forest, Salary Prediction, Supervised Learning.

I. Introduction

For many people, the most common reason for resignation is their salaries; higher salaries motivate employees to stay more in a company, and low or unraised ones encourage employees to switch their work to a different company [1]. Usually, specific human traits, educational background, and work experience highly affect one's salary. Salary prediction is needed to make one aware of their salary estimation. At the same time, it helps to allow a company recognizes what an employee is expecting from them [1]. Fortunately, the fast-emerging topics of Data Science and Machine Learning have allowed us to find enormous datasets for salaries and apply prediction techniques to them.

In many disciplines, data science has increased the discovery of probabilistic outcomes. Nowadays, data science is an emerging discipline that uses statistical methods and computer science knowledge to make meaningful predictions and insights in multiple conventional scholarly fields [2]. At the same time, another hugely emerging technology is machine learning, coming in with many of its enormous disciplines and mechanism to allow computers to learn without being explicitly programmed. Machine learning consists of two main concepts; supervised learning, and unsupervised learning. Supervised learning is when an algorithm can generate a function of input through a given output; the data has a label. In unsupervised learning, data is unlabeled; the algorithm works in such a way as to make the model learn the features on its own [3].

A simple supervised machine learning technique is logistic regression; it predicts the probability output of an event occurring. It takes a set of independent inputs and gives a categorically dependent value predicted [4]. The dependent variable is a binary variable that contains data coded as 0 or 1.

Another simple supervised machine learning algorithm is the Random Forest (RF); it builds decision trees from multiple data samples. Then, it uses its majority vote for classification and its average for regression [5]. Random forest is more accurate than traditional decision trees as it prevents overfitting. Also, it can handle missing values effectively while producing predictions without hyperparameter tuning.

An enormous technique of ML is the Neural Network (NN); it is a supervised machine learning algorithm widely used in many applications. It is a massively parallel distributed processor made up of simple processing units with a natural predisposition for learning to store and create experimental knowledge readily available for usage [6]. In brief, NN is a network of multiple processors connected to mimic human behaviors. NN works by building three types of layers; the input layer, the hidden layer, and the output layer.

Other supervised machine learning techniques are; support vector machine (SVM), K-nearest neighbors (KNN), Naïve Bayes, and nearest centroid. SVMs and KNN are ML algorithms that solve classification and regression problems. SVM is a low-dimensional input space that can be transformed into a higher-dimensional space by the kernel [7]. And then, KNN uses proximity to classify or anticipate how a set of individual data points will be arranged [8]. The Naïve Bayes is a quick and simple machine learning approach for predicting a class of datasets [9]. And lastly, the nearest centroid works by assigning a label to each training data closest to the centroid [10].

In this paper, we build different supervised machine learning techniques on an enormous dataset to make salary predictions. The paper proposes the implementation of three ML algorithms; linear regression, random forest, and neural networks. We implement our models using Python and run our algorithms on Google Colab. Then, a complete classification report and the accuracy of each of the models will be reported and compared. The main aim of this paper is to exploit the use of data science and machine learning techniques to predict salaries and then provide feedback on the most suitable ML model for such type of data.

The flow of this paper is as follows; section II includes a complete literature review, and section III has the implementation of the three models. Then, section IV shows the results, and section V concludes this paper.

II. Literature Review

Many researchers are interested in salary prediction, so different researchers applied different ML supervised learning techniques for salary prediction. In this section, we compare multiple ML models used for salary prediction by conducting a complete literature review. Thus, this will help us know the currently available research and its possible limitations. ML-supervised learning techniques work best for classified data; which is when input data maps to output data. Salary prediction datasets run among many supervised learning algorithms; this includes regression, random forest, support vector machine, K-nearest neighbors, Naïve Bayes, and nearest centroid.

Salary prediction models that use linear regression are implemented by Lothe et al. [1] on a small dataset and based only on four factors, and by Mukherjee & Satyasaivani [11] are based only on employee years of experience. The accuracy of both models is between 96% to 98%. Similarly, a regression ML model was implemented [12], and salary predictions were based only on the employee's job position [12]. At the same time, regression and SVM models were applied [13] to a huge dataset for salary prediction. The results show an accuracy of 79% for regression and 83% for SVM [13].

Furthermore, the regression, RF, SVM, KNN, and nearest centroid techniques predict salary based on different companies' traits and positions for a huge dataset in the UK. The accuracy of the models is as follows; regression model is between 73% to 74%, RF is 81%, SVM is 60%, KNN is 81%, and the nearest centroid is 65% [14]. Likewise, Martin et al. [15] apply linear regression, RF, SVM, and KNN models to a dataset including 4000 job posts in Spain, and the models have an accuracy of 58% for linear regression, 84% for RF, 66% for SVM, and 79% for KNN [15]. Another paper included KNN and Naïve Bayes to predict salary, and accuracy is 75.9% for KNN and 93.3% Naïve Bayes [16].

From this review, we can say that models with high accuracy; either consist of a small dataset or the predictions are made based on a small number of parameters. Thus, making the model's accuracy rate higher as the dataset is simple. Nevertheless, in terms of salary prediction conditions, this will be inaccurate as many factors affect one's salary. So, the dataset should include as many parameters as possible.

III. Implementation

To implement our ML models, we first find the datasets, then apply preprocessing techniques, prepare the training and testing data, and start modeling the data.

A. Dataset

Our dataset comes from two sources; online research and an online survey. Firstly, from research, we found 20,000+ datasets from Kaggle on the topic of salary prediction in the USA. Then, we conducted an online survey including questions similar to the ones in the dataset, and we distributed this survey to around 100 people in Kuwait with different backgrounds. We combined all the data into one CSV file to work on it.

The datasets focus on predicting whether one's salary is above 50,000 dollars per year by looking at specific traits. The traits are; age, work sector, education, marital status, occupation, gender, hours per week, and country. The dataset also has a column for the salary; this shows if salary is less than or equal to 50,000 dollars or greater than 50,000 dollars. The dataset decides the cut-off is 50,000 dollars as it is slightly close to the average household income per year [17].

The dataset mentioned above is tested over three ML algorithms to predict and check the accuracy of different ML models. The models are; logistic regression, random forest, and neural network models.

B. Data preprocessing

The dataset obtained is in a CSV format, which helped us to read and clean the data file easily. To begin with, we first had to drop all rows containing NaN values as such values could affect our model prediction by having missing information. Then, we started applying preprocessing techniques.

Our model's learning ability is influenced by; the quality of data and the meaningful information it can extract. For this reason, preprocessing is applied. We apply three techniques of preprocessing; normalization, one hot encoding, and binary encoding. Each one of the techniques used is needed to preprocess different types of data included in our dataset.

1) Normalization: is used for numeric data. In our dataset, this is the age and the number of working hours. In machine learning, normalization transforms the numeric data into a value within the range of 0 to 1. Normalization is needed to ensure that all numeric data is in the same format when used for training the model. The normalization formula is shown in equation (1).

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{1}$$

2) One hot encoding: is used for categorical data, which is string data that may be one of more than two options. In our dataset, this is the work class, education, marital status, occupation, and country. The one hot encoding is a mechanism used to format the data in such a way as to ease the training process. It produces a binary vector of categorical variables with a value of 1 for each row having this option and 0 otherwise. We use the Pandas library to convert our columns into one hot encoding.

3) Binary encoding: is used for Boolean data or data that can be one of two options. In our dataset, this is the gender and salary. Binary encoding converts the data in such a way that the new column is a statement; it is of a value 1 if it is true, else it is 0. We use Python functions to convert our columns into binary encoding.

C. Preparing the data

After cleaning the data, dropping unwanted rows, and applying the preprocessing algorithms, our dataset is now ready to be used for our proposed learning models. We will first use the train-test split mechanism to prepare the data for training and testing.

The train-test split is a mechanism used to evaluate the performance of learning algorithms. It splits the data into two type sets; a train set and a test set. The training set contains inputs with their output; this is needed for the model to learn and generalize the concept learned to other data. The testing set is a subset of the data that will allow the model to try and predict the output from the given input after being trained by the training set.

Scikit-Learn (Sklearn) library in Python can successfully perform the train-test split mechanism. The train-test split function from the Sklearn library is used in our code to achieve the train-test split. The test size given is 0.2; this means 20% of the dataset should be for testing, and the random state integer is also specified as SEED of 0 to initialize the random number generator to 0.

D. Data modeling

After preparing the data for learning, we have designed our proposed models; logistic regression, random forest, and neural network.

1) Logistic regression model: is implemented using the Sklearn library as it already has a built-in model to build the logistic regression. The model's implementation is simple, so we only need to call three functions; a logistic regression function, a "fit" for training (takes input and output train set as parameters), and a "predict" function for predictions (takes input test set as a parameter). The logistic regression function is given a stochastic average gradient descent (sag) as a solver; it is a variation of gradient descent and incremental aggregated gradient methods. SAG uses a random sample of previous gradient values. After model prediction, we print the model's accuracy and other characteristics provided in the next section.

2) Random forest model: is implemented using the Sklearn library as it already has a built-in model to build the random forest trees. The model's implementation is simple, so we only need three functions; a random forest classifier, a "fit" function for training (takes input and output train set as parameters), and a "predict" function for predictions (takes input test set as a parameter). We provide the number of trees in the model as 500 to the random forest classifier. After model prediction, we print the model's accuracy and other characteristics provided in the next section.

3) Neural network model: is implemented using Tensorflow and Keras; they are libraries in Python for machine learning. To develop a deep learning or a neural network model, we use the following functions from Tensorflow and Keras; sequential function, dense function, dropout function, compile function, fit function, and predict function.

The sequential function creates an empty linear stack of layers; this initiates our model. Then, we call multiple denser layers to fill our model. The dense function creates a fully connected layer of nodes. We provide it with the number of units, the activation function, and an input shape (for input layers only) based on the type of the layer. The activation function determines how input transforms into output. All layers implemented in our code are as follows:

- **Input layer** is given 20 units for higher accuracy, the activation function is the Rectifier Linear Unit (relu), and the input shape is the reshaped input of the dataset.
- **Hidden layers** are all also given 20 units, and the activation function is relu. Our model consists of three hidden layers as the data is quite large.
- **Output layer** only contains 1 unit, and the activation function is sigmoid because our data is divided into binary and multilabel classifications.

Then, the dropout function drops some neurons from the input or hidden layer. It helps to avoid overfitting. The dropout function takes the rate of 0.3; this is a fraction of the input unit to be dropped.

Then, we compile the neural network model based on a loss function, optimizer, and metrics using the compiler function. The loss function finds the error in the learning process. We set the loss function as binary cross entropy used for binary classifications. The optimizer optimizes the input weights by comparing the loss function and the prediction; we assign it to Adam with a learning rate of 0.001. Adam is a stochastic gradient descent method using adaptive estimation of first and second-order moments. Last, the metrics evaluate the overall performance of the model. We set the metrics to accuracy.

Then, a fit function trains the model for a fixed number of times based on the epochs. The fit function takes the input and output train data, epochs, batch size, and validation data. The epochs is 10; this is the number of iterations to pass over the entire train dataset. The batch size is 10; this is the number of samples per gradient update, and the validation data is the test sample data.

Last, a predict function predicts the output of the input test dataset (takes input test set as a parameter). After model prediction, we print the model's accuracy and other characteristics provided in the next section.

IV. Results

The three models are implemented in Google Colab using Python code with the needed libraries; Scikit-Learn, Keras, and TensorFlow. All models are used to train and predict the data. In all of our models, we use the predict function to predict the output of a sample testing input. The result of the predict function and the test sample output finds the accuracy, confusion matrix, and a complete classification report of a model.

The classification report function provides a report of the trained model that includes the value of precision, recall, f1-score, and the support of the predicted output. Furthermore, the report provides the average, macro avg, and weighted avg against those metrics.

The confusion matrix function provides the overall model's performance on the test data. The output of this function is a 2x2 matrix shown in Table I. It shows the actual result compared to the predicted output.

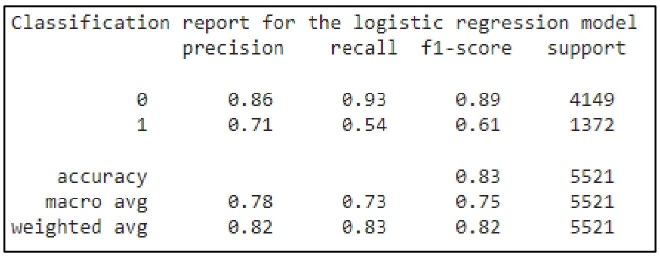
TABLE I. OUTPUT OF CONFUSION MATRIX.

N = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Negative
Predicted: Yes	False Positive	True Positive

The accuracy score function provides the accuracy value of the prediction of the trained model. Furthermore, we calculate the training time of each model by using the time library in Python.

A. Logistic regression results

In the logistic regression model, the classification report is as shown in Fig. 1. In this model, the confusion matrix is [[3838 311] [626 746]], and the accuracy score achieved is around 83.0%. And the time taken to train this model is around 0.363s.

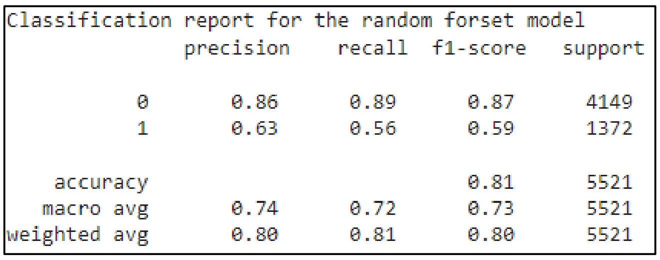


```
Classification report for the logistic regression model
              precision    recall  f1-score   support

           0       0.86      0.93      0.89      4149
           1       0.71      0.54      0.61      1372

    accuracy                           0.83      5521
   macro avg       0.78      0.73      0.75      5521
weighted avg       0.82      0.83      0.82      5521
```

Fig. 1. Logistic regression model classification report.

B. Random forest results

In the random forest model, the classification report is as shown in Fig. 2. In this model, the confusion matrix is [[3693 456] [607 765]], and the accuracy score achieved is around 80.7%. And the time taken to train this model is around 8.489s.

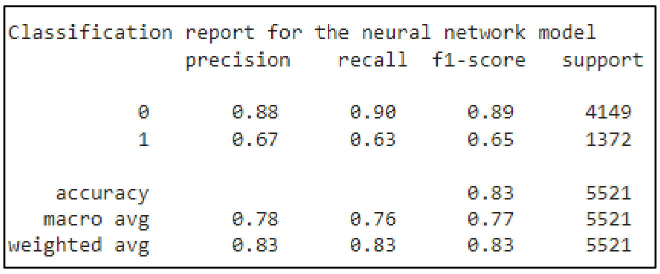


```
Classification report for the random forset model
              precision    recall  f1-score   support

           0       0.86      0.89      0.87      4149
           1       0.63      0.56      0.59      1372

    accuracy                           0.81      5521
   macro avg       0.74      0.72      0.73      5521
weighted avg       0.80      0.81      0.80      5521
```

Fig. 2. Random forest model classification report.

C. Neural network results

In the neural network model, the classification report is as shown in Fig. 3. In this model, the confusion score is [[3730 419] [508 864]], and the accuracy score achieved is around 83.2%. And the time taken to train this model is around 82.79s.



```
Classification report for the neural network model
              precision    recall  f1-score   support

           0       0.88      0.90      0.89      4149
           1       0.67      0.63      0.65      1372

    accuracy                           0.83      5521
   macro avg       0.78      0.76      0.77      5521
weighted avg       0.83      0.83      0.83      5521
```

Fig. 3. Neural network model classification report.

D. Comparing results of the models

In terms of accuracy, from the three classification reports, confusion matrix, and accuracy score, we can see that the neural network model has the best accuracy level, precision, recall, and f1-score. Then comes the logistic regression, and last is the random forest. In terms of time, the neural network is the slowest, then comes the random forest, and the fastest is the logistic regression. Table II summarizes the overall result of the accuracy level and time of the three trained models.

TABLE II. COMPARISON OF MODELS RESULTS.

Performance Metrics	**Model Name**		
	Logistic regression	***Random forest***	***Neural network***
Accurarcy	83.0%	80.7%	83.2%
Time	0.363s	8.489s	82.79s

V. CONCLUSION

In conclusion, we have tested and compared three types of supervised machine learning models; logistic regression, random forest, and neural networks. The models are tested on a salary prediction dataset to see how one's personal traits and educational background affect their salary. Our dataset's output is a binary output of 1 if the salary is above 50,000 dollars per year or 0 otherwise. From the obtained results, we can say that, on such a dataset, the neural network model is the most accurate with 83.2% accuracy, but it is the slowest as it needs 82.79s to train the model. Then, random forest is the least accurate with 80.7% accuracy, and its time taken is considered low as it takes 8.489s to train the model. Then, logistic regression has an accuracy level between the other two models of 83.0% accuracy, but it's the fastest as it only needs 0.363s to train the model. Therefore, we can conclude that a neural network is best when accuracy is the main factor, and random forest or logistic regression is better when time is the main factor. The three models are supervised machine learning techniques used to train computers to predict the output of a given set of inputs; this is makes predictions easier for any application.

The main limitation of this paper is that the obtained result is for only one type of dataset, so reduced categories of the dataset may slightly differ from the final result. However, on a large dataset (similar to the one used in this paper), the ML models' accuracy and time will always be the same. In the future, we aim to combine different ML models to see how this could affect the overall accuracy of the ML algorithms.

VI. ACKNOWLEDGMENT

We wish to acknowledge the generous financial support from the Kuwait Foundation for the Advancement of Sciences (KFAS) to present this paper at the conference under the Research Capacity Building/Scientific Missions program.

REFERENCES

[1] M. D. Lothe, P. Tiwari, N. Patil, S. Patil, and Patil, V, "Salary Prediction using Machine Learning," International Journal of Advance Scientific Research and engineering Trends, vol. 6, issue 5, 2021 pp. 199-202.

[2] L. M. Brodie, "What Is Data Science?" In book: Applied Data Science, 2019, pp. 101-130.

[3] R. Bansal, J. Singh, and R. Kaur, "Machine learning and its applications: A Review," JASC: Journal of Applied Science and Computations, 2020, pp. 1076-5131.

[4] H. A. Park, "An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain," J Korean Acad Nurs, vol. 43, issue 2, 2013, pp. 154-164.

[5] M. Azhari, A. Alaoui, Z. Acharoui, B. Ettaki, and J. Zerouaoui, "Adaptation of the Random Forest Method: Solving the problem of Pulsar Search," SCA '19: Proceedings of the 4th International Conference on Smart City Applications, 2019, pp. 1-6.

[6] A. Sharkawy, "Principle of Neural Network and Its Main Types: Review," Journal of Advances in Applied & Computational Mathematics, vol. 7, issue 1, 2020, pp. 8-19.

[7] D. Srivastava, and L. Bhambhu, "Data classification using support vector machine," Journal of Theoretical and Applied Information Technology, vol. 12, issue 1, 2010, pp. 1-7.

[8] Z. Zhang, "Introduction to machine learning: K-nearest neighbors. *Annals of Translational Medicine*," vol. 4, issue 11, 2016, pp. 218-218.

[9] T. N. Viet, and L. M. Hoang, "The Naïve Bayes algorithm for learning data analytics," Indian Journal of Computer Science and Engineering, vol. 12, issue 4, 2021, pp. 1038-1043.

[10] S. Johri, S. Debnath, A. Mocherla, A. Singh, A. Prakash, J. Kim, and I. Kerenidis, “Nearest Centroid Classification on a Trapped Ion Quantum Computer,” npj Quantum Information vol. 7, issue 1, 2021.

[11] T. Mukherjee, and B. Satyasaivani, “Employee’s Salary Prediction,” International Journal of Advance Research, Ideas and Innovations in Technology, vol. 8, issue 3, 2022, pp. 356-359.

[12] S. Das, R. Barik, and A. Mukherjee, “Salary Prediction Using Regression Techniques,” SSRN Electronic Journal, 2020.

[13] R. Voleti, and B. Jana, “Predictive Analysis of HR Salary using Machine Learning Techniques,” International Journal of Engineering Research & Technology (IJERT), vol. 10, issue 1, 2022, pp. 34-37.

[14] L. Li, X. Liu, and Y. Zhou, “Prediction of Salary in UK,” Computer Science and Engineering department of UC San Diego, 2018.

[15] I. Martin, A. Mariello, R. Battiti, and J. A. Hern´andez, “Salary Prediction in the IT Job Market with Few High-Dimensional Samples: A Spanish Case Study,” International Journal of Computational Intelligence Systems, vol. 11, 2018, pp. 1192-1209.

[16] K. Gopal, A. Singh, H. Kumar, and S. Sagar, “Salary Prediction Using Machine Learning,” International Journal of Innovative Research in Technology (IJIRT), vol. 8, issue 1, 2021, pp. 380-383.

[17] U.S. household income distribution 2021. Percentage distribution of household income in the United States in 2021 (in U.S. dollars)* [Graph]. In *Statistics*.