Hiring A Perfect Machine Learning Engineer

Tan Kok Ann Jeff
Computer Science / Year 3
katan@stud.ntnu.no

Report in TDT4310, NTNU, 16th April 2024

Abstract

Screening of resume during the hiring processes has always been a headache for the human resource department, whether they do it manually or with an existing tool that filter out the candidates. The question now arises: how can machine learning tools be effectively incorporated to identify the ideal employee for any job, based on job descriptions and provided resumes? This paper is based on hacker earth challenge which focused on hiring a perfect machine learning engineer. Various machine learning models such as XGB, SVM and random forest are employed and compared to determine the most effective learner in this context. Furthermore, deep learning models like BERT is also used. The result will show that which model would be the best to use in this context.

1 Introduction

Usage of artificial intelligent tools have became trending in the recent years. One example is , Chatgpt which is powered by AI that works as your personal chatbot for various task. The human resource industry also incorporate the use of AI tools as seen in their hiring and firing process.

A forbes article titled "How Companies Are Hiring And Reportedly Firing With AI" talked about how companies are incorporating AI in helping them to automate the process of hiring or firing. AI are frequently used in resume screening to help the company's human resource department to offload mundane work when there are too many applicants to the company especially for the bigger companies.

Such actions taken by companies tell us that AI have been involved largely in the decision making of whether someone gets hired or not, which led to the problem of "Hiring a Perfect Machine Learning Engineer" that i am trying to tackle here. This program originated from a hacker earth challenge problem called "A perfect fit" which encourages the coders across the globe to come up with interesting way to approach the problem.

Different sets of approach are applied to the problem and the results are being graded by a score of 100. The formula to calculate the score is given below:

Score = 100*max(0, 1 - MSLE)

$$\text{MSLE} = \frac{1}{n}\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2 \tag{1}$$

Overall, this formula penalizes models more heavily for large errors (especially under predictions) and rewards models for smaller errors, encouraging better predictions while ensuring that the score remains within a reasonable range.

As the problem has no official answer published online, i will take the answer of the first in the leader-board's as the benchmark for my answer. To address the problem, I will employ the TF-IDF, SVM, Random Forest, and XGBRegression models. By utilizing different models, we can analyze their impact on the results using various performance metrics for comparison.

The overall structure of the project will be as follows: Section 2 explains in detail about the terminology and machine learning methods used in the project. Section 3 discusses about the work related to using Artificial Intelligent methods in resume screening. Section 4 will be going through the Architecture of the project and the reasoning behind the methodologies. Section 5 will be going through the experimental setup and results. Section 6 will be a discussion on our current results and a conclusion will be made in section 7.

2 Background

This section will allow you to understand the terminology or machine learning methods/techniques that are used throughout the project. It will be important to read them before going further in depth with the project.

2.1 PyMuPDF

PyMuPDF is a high-performance Python library for data extraction, analysis, conversion and manipulation of PDF (and other) documents. PyMuPDF is hosted on GitHub and registered on PyPI. This is used in the project to extract the text out of the .pdf file. However, it does not include Optical Character Recognition (OCR) capabilities. This may limit the format of the pdf , which will be discussed further in section 7.

2.2 TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) is a widely used statistical method in natural language processing and information retrieval for representing text data. This is actually a bag-of-words method that assign a vector and weight to each word. TF-IDF helps in capturing the importance of words in a document relative to the entire corpus, thus providing a more meaningful representation of text data.

2.3 Linear Regression

Linear Regression is one of the simpler machine learning learners implemented in the project. It is usually used to predict numeric variable which is helpful in this case as we are predicting the match percentage.

2.4 K-Nearest Neighbour

K-Nearest Neighbour is one of the simpler machine learning learners implemented in the project. It utilise on a single hyperparameter K which is the number of the nearest neighbour that should be considered for each point , K's most preferred value is usually 5.

2.5 Decision Tree Regressor

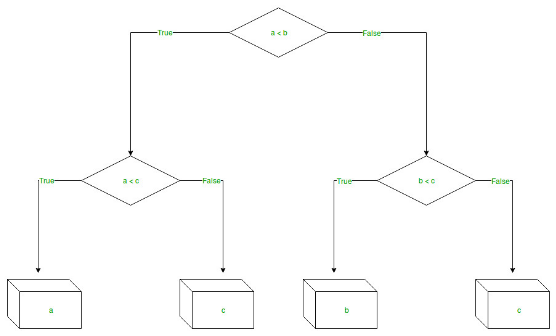

Figure 1: Decision Tree Algorithm (figure source:geeksforgeeks)

A decision tree is a supervised learning method that is used for the regression problem in this case as the output variable would be continuous. First you would have an input node and edges from the node that state which direction it goes through the network.It will repeat the process till it reaches the terminal node. Overall, it will observe a features of the input then train the model in the structure of a tree predict the data as continuous output. It is used for my project as due to their interpretability, ability to handle nonlinear relationships, and effectiveness with sparse, high-dimensional data.

2.6 SVM

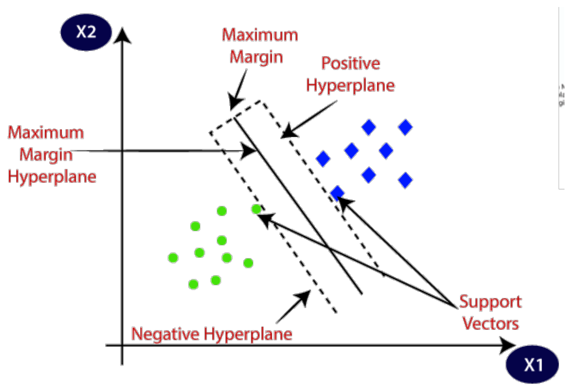

Figure 2: Support Vector Machine Algorithm (figure source:javatpoint)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. In this case we used it for the regression problem. How it works is that the SVM will create the best decision boundary that can segregate n-dimension space into classes so new data point can be put easily into the right category in the future.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

2.7 Random Forest

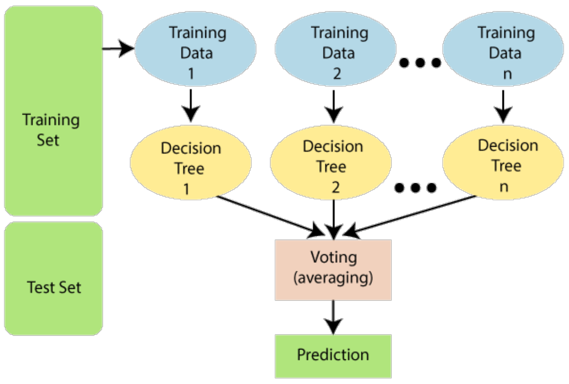

Figure 3: Random Forest Algorithm (figure source:javatpoint)

Random Forest is one of the complex machine learning learners implemented in the project. It can be used for both Classification and Regression problems. It works by constructing n number of decision trees that is built from using a subset of training data and a random subset of features.It predicts by combining individual predictions of each tree through averaging and gives the final continuous output.

2.8 xgboost

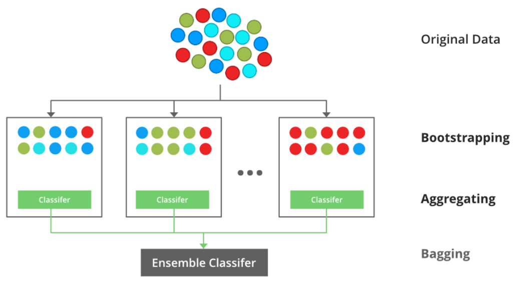

Figure 4: Extreme Gradient Boosting Algorithm (figure source:javatpoint)

XGBoost - Extreme Gradient Boosting is one of the complex machine learning learners implemented in the project. It is actually an ensemble learning methods similar to random forest that combines the prediction of the multiple weak models to produce a better prediction. It works by sequentially adding decision trees to an ensemble, where each subsequent tree will correct the errors made by the previous ones. It optimizes a loss function by iteratively fitting the new trees to the residuals of the predictions from the existing ensemble.

2.9 Stacking Regressor

Stacking Regressor is the ensemble method used in this project that is called from the scikit-learn library. It helps to combine multiple models into a single predictive model. Stacking starts with training of its base regression models that are mentioned above. Then a meta-model is used to trained on the predictions made by these base models using the training data. Finally during testing phase, when making predictions on test data, the base models will makes its predictions, and these predictions are combined with the meta-model to produce the final prediction.

2.10 Feed Forward Neuron Network

A feed forward neuron network (FFNN) is a type deep learning model that is designed to process data in a single forward direction, from input to output layers.

2.11 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a state-of-the-art technique for NLP tasks. It is based on transformer that uses encoder only structure and it is bidirectional. It can take a maximum of 512 worded input and outputs 768 dimensional representation for each token.

3 Related Work

Navale Sakshi, Doke Samiksha, Mule Divya, and Prof. Said S. K. (2022) conducted a study on the application of LSTM for resume screening. Overall, this report provides insights into using deep learning techniques like LSTM for automating the resume screening process, addressing the challenges faced by recruiters in handling large volumes of resumes efficiently. It also suggests avenues for further research and development in this area. However, there is no results of their methods as it is only a proposed approach but this give me an idea on how can i approach my problem which is much similar to the one in the study. Therefore, after reading the study , I have start of with a simpler approach of using basic machine learning models such as Linear Regression, Decision Tree, KNN , SVM , XGBoost, random forest and use ensemble stacking. However, though it was mentioned that LSTM was their approach but I will work on a feed forward neuron network to potentially get better results.

4 Architecture

This section will talk about the architecture used for the project and show what is done in the preprocessing of the data to fit it into the learners I am using.

4.1 Dataset

The dataset folder from kaggle contains the following files:

- **train.csv**: 90 Rows x 2 Column
- **trainResumes**: 90 resumes in pdf format
- **test.csv**: 60 Rows x 2 Column
- **testResumes**: 60 resumes in pdf format
- **Job description.pdf**: PDF file that represents the job description of a Machine Learning engineer

The dataset(train.csv) contains the following columns:

Column name	**Column description**
CandidateID	Represents the unique identification number of a candidate
Match Percentage	Represents the percentage that a candidate fits based on the job description

4.2 Text Extraction from PDF

I have decide to use PyMuPDF to extract the text from the pdf dataset I have. I noticed that there might be difficulty in parsing different PDF templates. For instance, if the pdf is a image of the resume that was scanned, the PyMuPDF text extraction would not work properly. Therefore, after much research , I realised that using a Optical Character Recognition (OCR) software like OCRmyPDF to insert a hidden text layer underneath the visible page would make it work. However, for the nature of the dataset I have for this project, there is no need to do so as normal PyMuPDF text extraction works on all the pdfs. This could be a potential improvement for the work.

4.3 Data Preprocessing

After extraction of the text from the PDF dataset, the cleaning process consist of lower casing all the letters, removal of punctuation, stop words and single words. Lastly, lemmatizing is done with WordNetLemmatizer. There was no need to tokenize the string of text as it will be done in TFIDF vectorizer as we called the analyzer to be word so that it analyse the string as individual word and not the whole chunk of string. The dataset provided is already split in a 40-60 ratio of test and training data. In the end both dataset are vectorized using the TF-IDF and it is ready to be insert into the machine learning learners.

4.4 Machine Learning Models

As it have been explained in the introduction section what is each machine learning model, but I will go through the machine learning model used in the project again briefly. I will be going to try most of the simple to complex learners namely, XGBOOST, SVM, Random Forest Score, Linear Regression, KNN Regression and Decision Tree Regression.

4.5 Stacking Ensemble

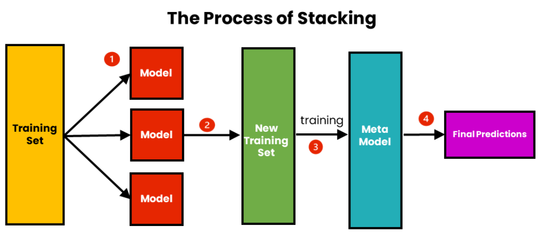

Figure 5: Stacking Ensemble

Next, with the best models out of the ones I tried, i will do a Stacked Generalization to get better results by combining the best of both features to create a better model. For this I will need choose the best model based on the score that they are giving me , meaning the higher the score , the better the model. Afterwards, I will be using a meta model which I will be trying the simpler model to avoid complexity.

4.6 Stacking Ensemble Architecture Overview

Here will be the overall view of the architecture explained in the previous sections.

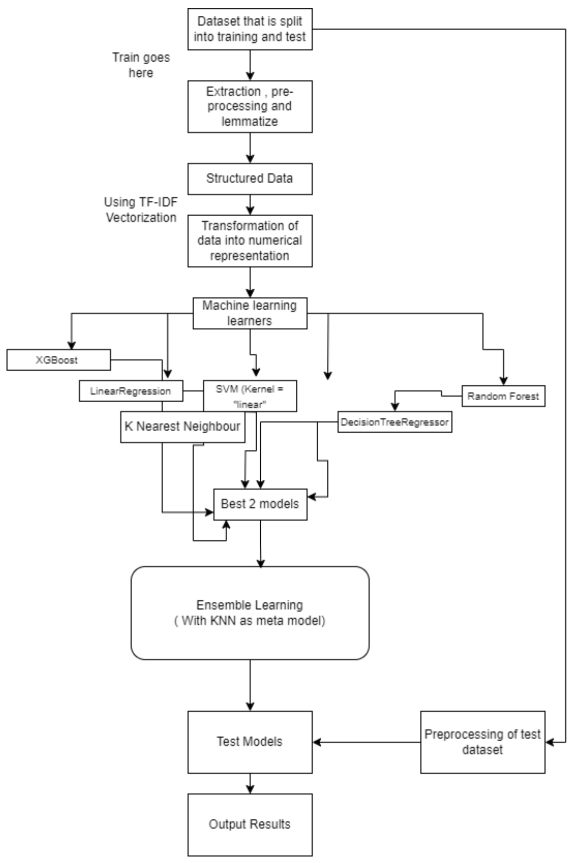
Dataset that is split into training and test
Train goes here
Extraction , pre-processing and lemmatize
Structured Data
Using TF-IDF Vectorization
Transformation of data into numerical representation
Machine learning learners
XGBoost
LinearRegression
SVM (Kernel = "linear"
Random Forest
DecisionTreeRegressor
K Nearest Neighbour
Best 2 models
Ensemble Learning (With KNN as meta model)
Test Models
Preprocessing of test dataset
Output Results

Figure 6: Architecture Overview Drawn with draw.io

4.7 Feed Forward Neural Network with TF-IDF Features

With the same preprocessed data through the TF-IDF vectorization, I have pass it into a feed forward neural network with the following architecture

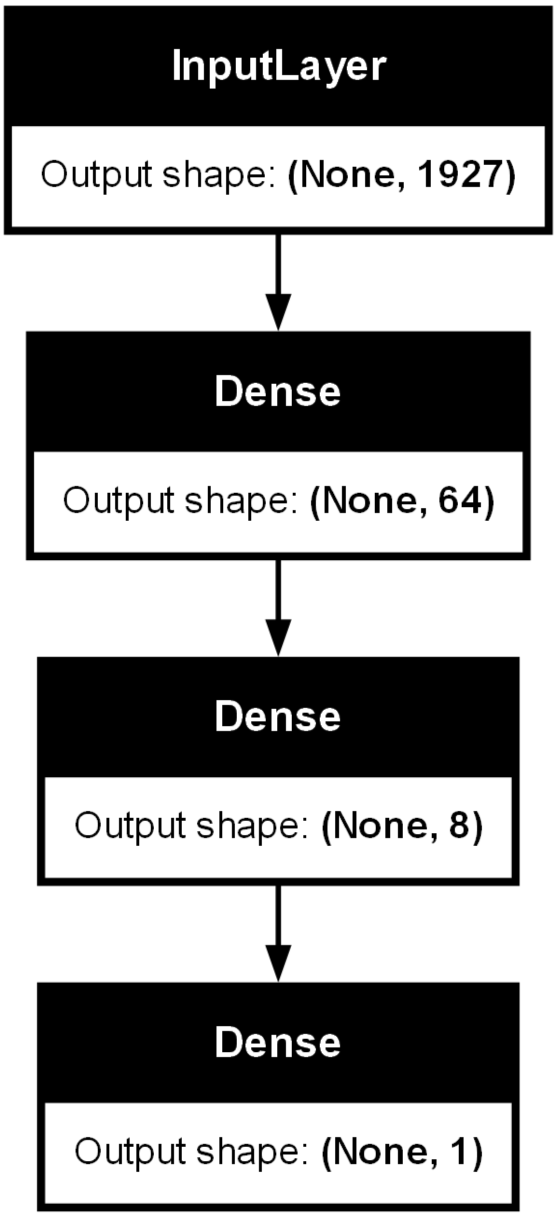

Figure 7: FFNN Structure

4.8 DistilBERT with feed forward neural network

Next, i have tried using pre-trained deep learning model to potentially get better result.The PDFs are uploaded and performed encoding using a pretrained DistilBERT-based tokenizer, setting the maximum length to 300 and applying zero padding as needed for batch training purposes. It uses the pre-trained DistilBERT model to extract features from both the job description and each resume. Specifically, it gets the output corresponding to the [CLS] token (the first token in the input) from the DistilBERT model's last hidden state. The dataset have been split the same way as mentioned in section 4.1. Then, a feedforward neural network is defined which takes in the concatenated features of the job description and resume as input and predicts a match percentage as output.

This is an overall view of the feedforward neural network architecture.

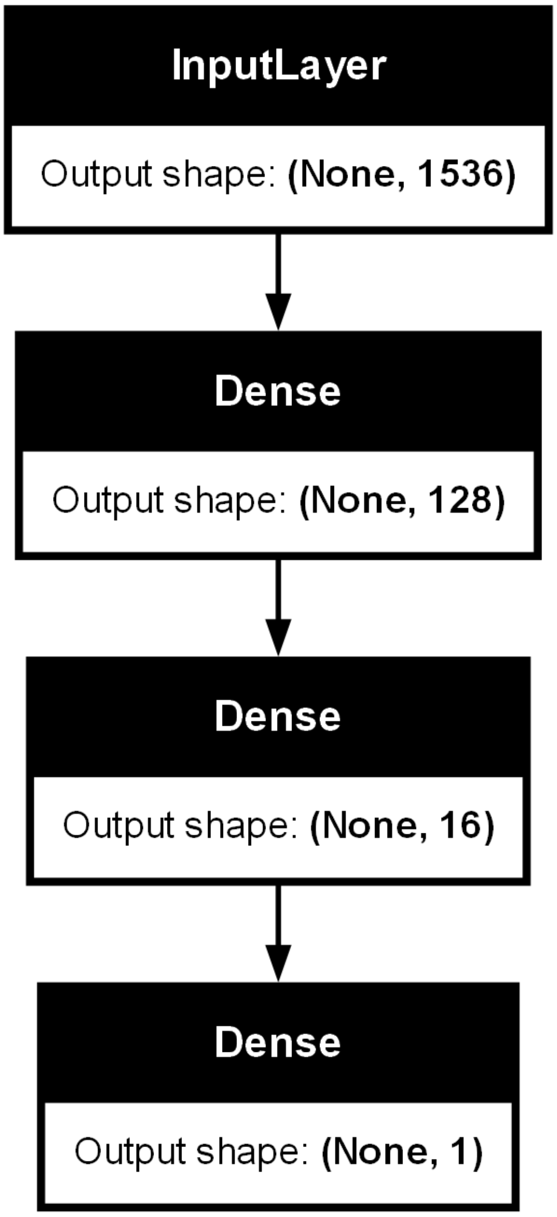

Figure 8: FFNN Structure

5 Experiments and Results

This section will guide you through the process of setting up the parameters for the experiment, as well as the results obtained.

5.1 Experimental Setup

As mentioned in the architecture, I will be testing multiple machine learning leaners first then use an ensemble learner to potentially get better results.

5.1.1 Models

We experimented with several machine learning models to predict match percentage. Each model was trained and evaluated using appropriate metrics to determine its performance.

1. **XGBoost**:
    - Learning Rate: 0.005
    - Number of Estimators: 700
    - Objective: Squared Error
    - Max Depth: 8
    - Regularization Lambda: 1.3
    - Gamma: 1
    - Minimum Child Weight: 1.5
    - Max Delta Step: 100
    - Random State: 31
2. **Support Vector Machine (SVM)**:
    - Kernel: Linear
3. **Random Forest**:
    - Number of Estimators: 100
    - Random State: 42
4. **K-Nearest Neighbors (KNN)**:
    - Number of Neighbors: 5
5. **Linear Regression**
6. **Decision Tree Regressor**:
    - Max Depth: 50

5.1.2 Stacking Ensemble

After testing the 6 different models , I have realised that the top 2 model that work the best are the Linear Regression and Random Forest. In this case for the setup of the stacking ensemble's base model will the two that worked the best during the training phase. For the meta-model , I have tested multiple model such as XGBoost

5.2 FFNN Model

- optimizer: AdamOptimizer
- learning rate: 1e-7
- epochs: 10,000

The structure of the model is seen in figure 7 and figure 8 where we define a feedforward neural network architecture to train the dataset we have preprocessed with the first method of TF-IDF vectorization and the BERT encoding.

5.3 Experimental Results

Table 1: Model Evaluation Metrics

Model	**Score**	**Mean Absolute Error**	**Mean Square Error**	**R2 Score**
XGBoost	65.51	11.96	248.22	-0.16
SVM	63.71	11.45	273.68	-0.28
Random Forest	78.72	9.44	141.18	0.34
KNN	72.09	11.62	225.62	-0.06
Linear Regression	81.15	8.34	122.00	0.43
Decision Tree Regressor	56.11	13.37	317.74	-0.49

The table above shows the relevant evaluation metrics for the machine learning models that I have tested. We can see that out of the 6 models tested, Linear Regression and Random Forest are the ones that have the best score and also the highest R2 score.

Table 2: Model Evaluation Metrics

Model	**Score**	**Mean Absolute Error**	**Mean Square Error**	**R2 Score**
StackingRegressor	83.75	8.65	128.60	0.40

It shows that after using the stacking emsemble method , I am able to achieve better results compared to the other models.

Next I will going through the results from the Feed Forward Neural Network with the TF-IDF features.

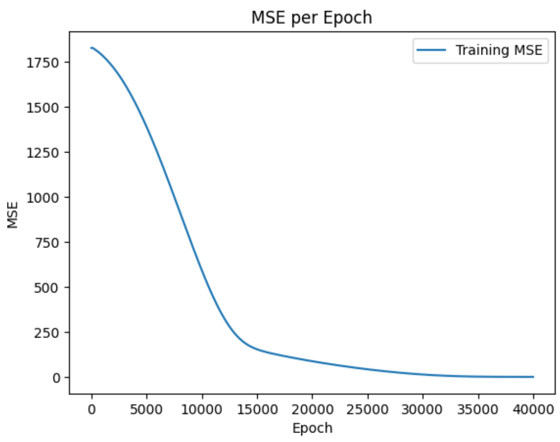

Figure 9: Mean Square Error Graph Over Epochs

We can see that after training for 40,000 epochs , the model seems to be converging to 0 MSE and this shows signs of overfitting with the train dataset. Therefore, the training was stopped and the trained model is used to predict with the test dataset as seen below.

Model	**Score**	**Mean Absolute Error**	**Mean Square Error**	**Epochs**	**Batch Size**
FFNN with TF-IDF features	79.85	9.21	138.54	40,000	4

Table 3: FFNN with TF-IDF feature Model Evaluation Metrics

Last, I be going through the results of our Feed Forward Neuron Network with BERT.

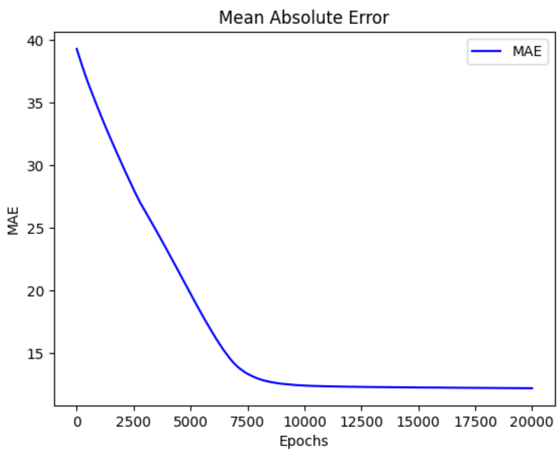

Figure 10: Mean Absolute Error Graph Over Epochs

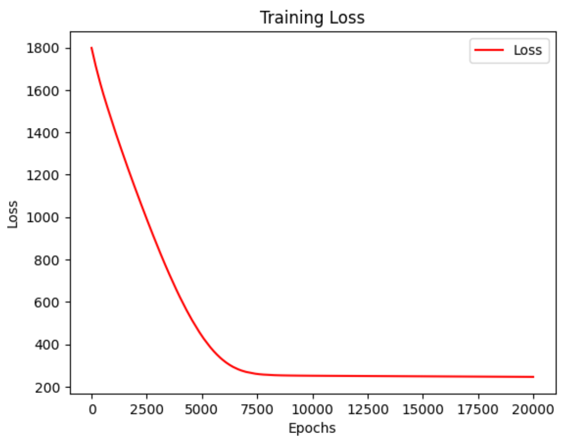

Figure 11: Training Loss Graph Over Epochs

We can see that after training for 20,000 epochs , the model seems to converges after around 10,000 epochs. Meaning that the model is trained to its best fit. The results is as shown on the table below.

Model	Score	Mean Absolute Error	Mean Square Error	Epochs	Batch Size
DistilBERT	69.49	11.03	216.51	20,000	4

Table 4: FFNN with BERT Model Evaluation Metrics

6 Evaluation and Discussion

It is apparent from the results in Section 6 that my approaches used for predicting the match percentage scores for each resume has flaws. This section is to discuss how does the flaws and results from the experiment can be possibly mitigated or lessen.

6.1 PDF Format

It is noted that there is difficulty parsing unknown PDF templates which is a non-trivial problem that can result in the failure of the whole code. However, with the current dataset, the code written was catered to normal pdf templates using PYMuPDF which have a good performance rate compared to the other libraries available mentioned by Martin Thomas (2023). One issue with this is that unknown PDF templates such as having the PDF to be an image of the resume will cause some issue. Reason being is that PYMuPDF has no Optical Character Recognition capabilities. However, with the fact that there are no different pdf resume dataset to test with, this problem will be brought to section 8 for further discussion as an enhancement to the current project.

6.2 Match Percentage

The match percentage is the metric we aim to predict for each resume. This raises a significant question: how was the match percentage predetermined in the first place? Was it determined through a tool that assesses how well the resume matches the job description, or was it based on input data related to hiring? This metric may limit the project's scope since it can only predict the match percentage based on a predetermined value assigned to the resume initially.

Therefore, a potentially better approach could involve utilizing data from companies hiring for similar roles to determine the percentage of similar resumes that successfully pass the initial screening. This data could then be used to grade the resumes initially.

6.3 Machine learning models vs Deep learning models

In the project, it is evident that I start off with using traditional machine learning models then moved on to use a deep learning model like feed forward neuron network then used a pre-trained BERT transformer with a feed forward neural network model. The usage of two different type of models yield different results. It was evident that the ensemble stacking model created based on the traditional models yield better results than the complex feed-forward neural network. This can be explained with the fact that the dataset size is too small for the deep learning model to generalize well even after 20,000/40,0000 epochs whereas on the other hand, a traditional model might perform better due to its ability to generalize with less data.

7 Conclusion and Future Work

In this work, I targeted the task of hiring a perfect candidate for a given job which in this case was a machine learning engineer. For this, I presented 9 different machine learning models, having 6 traditional methods, 1 self-customised model with stacking ensemble and lastly 2 deep learning method using BERT with Feed forward neuron network. I have trained and evaluated with the same dataset for all the methods, with only differences in the pre-processing for the traditional methods and the deep learning method involving BERT. The result was clear that the traditional methods worked the best compared to the deep learning method that was used.

The reason could be possibly that the size of the dataset was too small for the deep learning to learn properly and also there is no sequence structure for the resume as it is extracted to be just a chunk of text. However, the initial goal was to achieve a high score to prove that the methods used in this project is effective in predicting the match percentage of the given resumes on the specific job description.

However, there are still many flaws in the project and this project can be improved further with more precise performance metrics for measurements and also the fact that not all types of pdf templates can be extracted properly.

With that in mind, the first improvement to the project would be the introduced of an Optical Character Recognition libraries that can scan PDFs and extract the text from the images. One such library that can be considered to be used is pytesseract - the library wraps Tesseract OCR Engine, providing a simple interface to perform OCR in Python. This will make the pdf parsing of the project a more concise and effective approach.

Lastly, there should be clearer definitions on the match percentage that was given in the dataset as it should be properly calculated before the model made in the project can predict the rest of the resumes properly.

8 References

Navale Sakshi, Doke Samiksha, Mule Divya and Prof. Said S. K. Resume Screening Using LSTM. Apr 2022. URL $https : //ijrpr.com/uploads/V3ISSUE4/IJRPR3705.pdf$

Martin Thoma. The Python PDF Ecosystem in 2023. Mar 2023. URL $https : //martinthoma.medium.com/the-python-pdf-ecosystem-in-2023-819141977442$

Brijesh Soni. Stacking to Improve Model Performance: A Comprehensive Guide on Ensemble Learning in Python. May 2023. URL $https : //medium.com/@brijesh_soni/stacking-to-improve-model-performance-a-comprehensive-guide-on-ensemble-learning-in-python-9ed53c93ce28$

Rani Horev. BERT Explained: State of the art language model for NLP. NOV 2018. URL $https : //towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270$