1. PROBLEM DEFINITION:-

The main problem a company faces while hiring new candidates is the selection of suitable candidates based on their resumes and the job description for the first round of the interview process. It's like finding a needle in a haystack, especially for the big companies like MAANG or even other popular MNCs.

Many job search companies like naukri.com, google jobs, LinkedIn have made finding and applying for a job very easy, which has its own problem. Nowadays a company receives thousands or even tens of thousands of resumes against a single post. Traditionally all resumes require manual review by some HR managers for further process.

This means the big companies need to have a sufficient number of HR managers for this task. Also, there are different types of job posts in a company so these managers should have proper knowledge of the job descriptions. So that they can select the suitable resumes. It may also introduce human bias because some people are very strict and some are lenient.

The selection of the suitable candidates based on the job description and the resumes become a very time-consuming process when we scale it for large companies. Luckily with advancement of NLP (natural language processing), we can solve this problem in a very less amount of time. Which will save the workforce and hence also some money for the company.

Let's say an HR manager can check a resume in 5 minutes and he/she works 8 hours per day. Then he/she will be able to check only 96 resumes in a whole day. Which is a work of a few seconds for a machine learning model.

According to inc.com, Google receives about 3 million resumes every year. Which according to the above estimate will take about 250 thousand hours of workforce to select candidates for the first round of the interview process. And a machine learning / NLP model will take a few minutes or hours to process based on the availability of compute process. Along with time, the NLP model will save a ton of money for the company.

There are some ATS (application tracking system) available in the industry which are few years old. But with the new NLP technologies such as transformer and attention based models we can develop far more powerful tools to solve this problem of candidate selection with much more accuracy.

In this project, I will be calculating the match percentage of the resumes for a job post. Based on the match percentage a company can ask top n candidates for the

first round of the interview. Or the company can choose candidates with match percentage greater than a certain threshold.

The match percentage will indicate how fit a candidate is for the job role. A match percentage of 100 means that the candidate is perfectly fit for the job. Whereas a match percentage of 0 means that the candidate is not at all fit for the job. The higher the match percentage, the better it is for the candidate.

2. DATASET DESCRIPTION:-

To tackle this problem I will be using a dataset titled “A Perfect Fit” [1] which is available on Kaggle. Which was published by Mukund in 2021 under CC0: Public Domain license. This dataset originally belonged to HackerEarth's monthly machine learning challenge.

This dataset has only 1 job description/job role and 90 resumes which has output values i.e the match percentage. The job description and the resumes are of pdf file type. The resumes have different formats/templates, which is a nice thing because in the real world we see resumes of different styles.

There is a csv file too which has two columns “CandidateID” and “Match Percentage”. The “CandidateID” is same as the file name of the resume. And the “Match Percentage” is numeric value between 0 and 100. It represents how much percent a candidate is fit for the given role.

There are 60 more resumes in test folder. But we don’t have access to the match percentage of those resumes. So we can’t use them.

The dataset size is very small in our case. So obviously we will not face issues with processing the data. But it will create some overfit issues and the model might not generalize better. To get around this problem I can scrap LinkedIn for job description and public resumes but we won’t have match percentage values. It will require manual labeling. I will do these steps to get more data if required in the future.

To process the csv data we will have to use pandas. Pandas is a very popular library to process tabular data. It is fast and reliable. Tabular data in pandas are called dataframe.

To read the pdf we can use one of the various libraries such as PyPDF or PDFMiner. These libraries can convert the well-formatted PDFs into text.

3. KEY PERFORMANCE INDICATOR (KPI) METRIC:-

There are mainly two types of metrics for machine learning use cases. One is the business metric which is mainly used by the business folks to determine how much profit will the company make. Or in our case how much money will the company save with reduction in the workload. We have discussed about the business metric in brief in the Problem Definition section.

The other type of metrics is the performance metric, which is used by the model while training. And also by the data scientists to understand and compare results of different models. When we say metric, we often mean the performance metric. The Key Performance Indicator (KPI) is the performance metric that is used for training the model.

Different types of tasks such as classification and regression have different types of performance metrics. Since we are predicting numeric values, this problem comes under the regression task. Regression models have various interesting performance metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), coefficient of determination (R^2), etc. Along with the traditional performance metrics, we can also view the distribution/PDF of the errors. Each of the mentioned performance metrics have their own pros and cons.

The MAE is linear whereas MSE is quadratic in nature. The MSE penalizes more if the errors are large. The MAPE does not work well when we have very small output values. MAPE is used when we could expect large difference between predicted and actual output for higher values and small difference for lower values.

I have decided to use mean squared error (MSE) as the KPI metric for this task. We will also keep track of R-squared because it is easy to understand. And we will also view the PDF of the errors wherever it is necessary to check for any anomalies.

MSE is a very simple and most commonly used performance metric for regression tasks. MSE is average of square of difference between predicted and actual outputs. MSE is always greater than equal to zero. MSE of zero means the actual outputs and predicted outputs are the same. However, we don't see MSE of zero in practice. The lower the MSE value the better it is. There is no upper bound for MSE. This make MSE a little bit hard to understand. However, we can get around this problem by comparing the MSE with a mean predictor as in R-squared. MSE is used by default for most of the regression problems. Unless a specific performance metric is required, we use MSE.

The formula for MSE is,

$$MSE = 1/n \sum_{i=1}^{n} \left(y_i - \widehat{y_i} \right)^2$$

The formula for R-squared is,

$$R^2 = 1 - \frac{RSS\ (Sum\ of\ squares\ residuals)}{TSS\ (Total\ sum\ of\ squares)}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \widehat{y_i} \right)^2}{\sum_{i=1}^{n} \left(y_i - \bar{y} \right)^2}$$

where,

y_i = actual output

$\widehat{y_i}$ = predicted output

$\bar{y}$ = mean of actual output

Instead of MSE we could have used RMSE or MAE. RMSE is square root of MSE, so there is not much difference, we could use either one of them. I used MSE instead of MAE because I wanted to penalize more if the error is large. That means we are okay with multiple small errors but we would like to avoid large errors.

The advantage of MSE is that it avoids large errors. That means we can be sure that we are not making any huge mistakes in prediction. This advantage becomes a disadvantage if we have a large number of outliers. Because the outliers will affect the MSE value a lot because of the effect of squaring.

4. REAL-WORLD CHALLENGES AND CONSTRAINTS:-

In the real world, we don't have any strict latency requirements. We can calculate the percentage of match between the resumes and the job description every night, every hour, etc. Or we can calculate the percentage as soon as the candidate uploads the resume. There is no strict latency requirement because we are not showing the match percentage to the candidate and it will be used by the employer.

One challenge is that we can manually filter out few of the candidate for whom the model output a larger match percentage. But there should not be large number of such cases because it will also require some manual labour.

The most concerning challenge we could face with the model is that we could miss out on some good candidates, if the model outputs a very small value for a fit candidate. This does not only affect the candidate but also the employer. Because finding candidates is easy but finding good candidates based on the job role is not so easy.

So basically we need to avoid large errors. Hence I have used MSE as the key performance indicator.

Scalability is not a big concern here because there is no such strict latency requirement. So if the number of resumes submitted increases, the employer has to wait a little longer for the match percentage.

Interpretability of the model is not a must but a good to have feature. For interpretability, we could extract the keywords from the resume. From these keywords, one can easily understand the justification behind the predicted output value.

There are multiple challenges with generalization. There are multiple types of job role in a industry. We would want our model to work for all types of job roles. Another concern is that there are multiple templates/formats of resumes. This is a very big concern because people write resumes very differently from one another. The same resume will look very different when it follows different templates. Also choosing of words differ from person to person. Some people write explanations and some just write in very brief. We will have to tackle these generalization issues while building the model.

Till now we have discussed the constraints and challenges in productionization of the model. But there are some challenges in the training the model too. Currently, we have a very small dataset for training. We could scrap the internet for job descriptions and resumes. But we have to manually set the output values. Those output values will certainly have some human biases. These biases will make the dataset a bit noisy. Also, the current dataset contains only one job description, so we can’t create a fully generalized model.

5. LITERATURE REVIEW:-

Before discussing how others have tackled this problem let me share what initial solutions we discussed in our group discussion. The first cut and the simplest solution that we could think of was using one-hot encoding and then some similarity based metric like cosine similarity. We could use this as a baseline model and build on top of it.

We could also try some simple such as Linear regression with BoW or TF-IDF or Word2Vec representations. Or advanced machine learning techniques such as XGBoost regression, Random Forest regression, etc.

Then we could also try deep learning based advanced techniques such as BERT to get embeddings of the resumes and job descriptions. And train a few simple MLPs with the embeddings.

Most of the blog posts and research papers [2,3] have used similar approach as our first cut solution. Where they have done preprocessing steps like tokenization, stop word removal, lemmatization, etc. and then used text featurization techniques like BoW and TF-IDF. Then they have used cosine similarity to sort the candidates based on relavance.

One more interesting approach Pradeep Kumar Roy et al. [3] has discussed is to first categorised the resumes in different categories. And then apply similarity technique on most relevant category of resumes. For this we would also require to categorise the job based on the description.

In another blog [4] , Dennis de Voogt has used spaCy tool to calculate the match percentage with a interesting and human like approach. After preprocessing, the author has used NER (named entity recognition) to extract the technical skills of the candidate from the resume text. The NER is used to extract important relevant information from text. These information could be phone number, country, etc. And in this case it is the skills of the candidate and skills mentioned in job description. After getting the entities we can use similarity based techniques.

6. REFERENCES:-

- Mukund. A Perfect Fit. https://www.kaggle.com/mukund23/a-perfect-fit (2021)
- Baban Deep Singh. Resume — Summarizing and Matching. (2020)
- Pradeep Kumar Roy, Sarabjeet Singh Chowdhary, Rocky Bhatia. A Machine Learning approach for automation of Resume Recommendation system. (2020)
- Dennis de Voogt, Always be learning. Matching resumes with job offers using spaCy. (2020)