Capstone Project - Final Report

Automation of Resume short listing using Machine Learning

Group - 1

1. Amit Kapoor
2. Ankur Girdhar
3. Avinash Singh Jawa
4. Rocky
5. Sarabjeet Singh Chowdhary

Table of Contents

[1 Domain and Context 3](#_Toc5562459)

[2 Problem Statement 4](#_Toc5562460)

[3 Process Flow Diagram 5](#_Toc5562461)

[4 Dataset(s) 5](#_Toc5562462)

[5 Data Pre-processing 6](#_Toc5562463)

[6 Feature Extraction 8](#_Toc5562464)

[7 Model Building and Model Selection 9](#_Toc5562465)

[8 Model Evaluation 15](#_Toc5562466)

[9 Implication 16](#_Toc5562467)

[10 Limitations 17](#_Toc5562468)

[11 Closing Reflections 17](#_Toc5562469)

[12 References 17](#_Toc5562470)

# Domain and Context

Talent acquisition is an important, complex, and time-consuming function within Human Resources. The sheer scale of India’s market is overwhelming. Not only is there a staggering one million people coming into the job market every month, but there is also huge turnover — per LinkedIn, India has the highest percentage of the workforce that is “actively seeking a new job”. Clearly this is an extremely liquid, massive market but one that also has many frustrating inefficiencies. The most challenging part is lack of a standard structure and format for resume which makes short listing of desired profiles for roles very tedious and time consuming.

Effective screening of resumes requires domain knowledge to be able to understand the relevance and applicability of a profile for the job role. With a huge number of different job roles existing today along with the typically large number of applications received, short listing poses a challenge for the human resource department which is only further worsened by the lack of diverse skill and domain knowledge within the HR department, required for effective screening. Being able to weed out non-relevant profiles as early as possible in the pipeline results in cost savings, both in terms of time as well as money.

Today the industry face three major challenges:-

**1. Separating right candidates from the pack -**

India being a huge job market and with millions seeking jobs; it is humanly impossible to screen the CVs and find the right match. This make the whole hiring process slow and inefficient costing resources to the companies.

**2: Making sense of candidate CVs -**

Second challenges is posed by the fact that the CVs in the market are not standard, practically every resume in the market has different structure and format. HR has to manually go through the CVs to find the right match to the job description. This is resource intensive and prone to error where by a right candidate for the job might get missed in the process.

**3: Knowing that candidates can do the job before you hire them -**

The third and the major challenge is mapping the CV to the job description to understand if the candidate would be able to do the job for which s\he is being hired.

# Problem Statement

Today the major problem being faced across industry is how to acquire right talent, using minimal resources over the internet [1] and in minimal time.

As described in section 1, there are three major challenges that are required to be overcome, to bring efficiencies to the complete process.

1. Separating right candidates from the pack
2. Making sense of candidate CVs
3. Knowing that candidates can do the job before you hire them

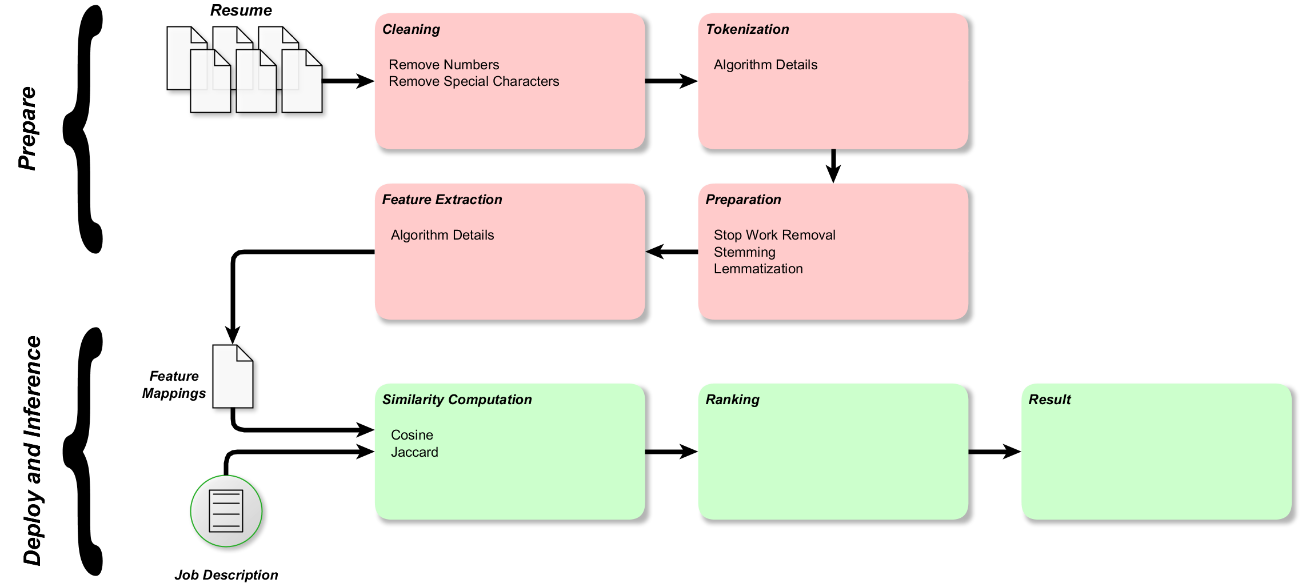
Our group intends to provide a solution to the above mentioned challenges by automating the process. The solution would help finding the right CV from the large dumps of CVs; would be agnostic to the format in which CV has been created and would give with the list of CVs which are best match to the job description provided by the recruiter.

The proposed solution involves unsupervised learning to classify the resumes into various categories corresponding to the various domains of expertise of the candidates. A multi-pronged approach to classification is proposed as follows [3]:

1. Perform NER, NLP and Text classification using n-grams.
2. Use distance-metric based classification.

The solution shall provide a feedback loop closure to adjust/improve the accuracy by incorporating the feedback corresponding to the incorrectly screened profiles.

# Process Flow Diagram



On a high level the model being designed would have two parts Prepare and Deployment & Inference

**Prepare** - In this process the CVs being provided as input would be cleansed to remove special or any junk characters that are there in the CVs. The data would be then tokenised and prepared for comparison with the job description provided by the recruiter. So, data mining [2] plays a very important role while providing input to the process

**Deployment & Inference** - In this process the tokenised CV data and the job descriptions would be compared and the model would provide CVs relevant to the job description as an output.

# Dataset(s)

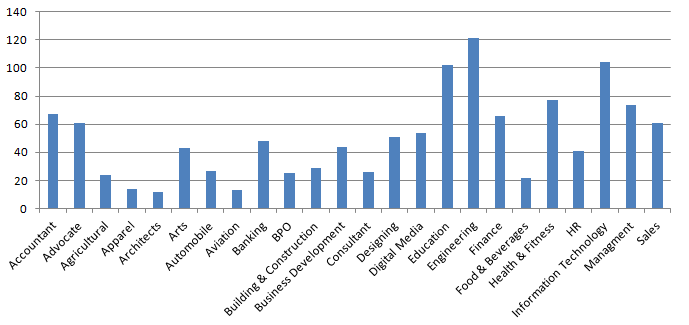
The data was downloaded from online portal(s) and from Kaggle. The data is in Excel format, with three column ID, Category, and Resume.

**ID** - Sequence number of the resume

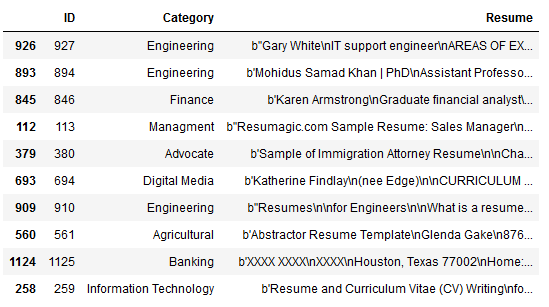
**Category** - Industry sector to which the resume belongs to

**Resume** - The complete CV of the candidate

*CV distribution across categories*



*Dataset Sample*



# Data Pre-processing

The raw CV file was imported and the data in the resume field was cleansed to remove the numbers and the extra spaces in the date.

Data masking was done as

1. Mask string fragments like \x

2. Mask string fragments for escape sequences like \a \b \t \n

3. Mask all numbers

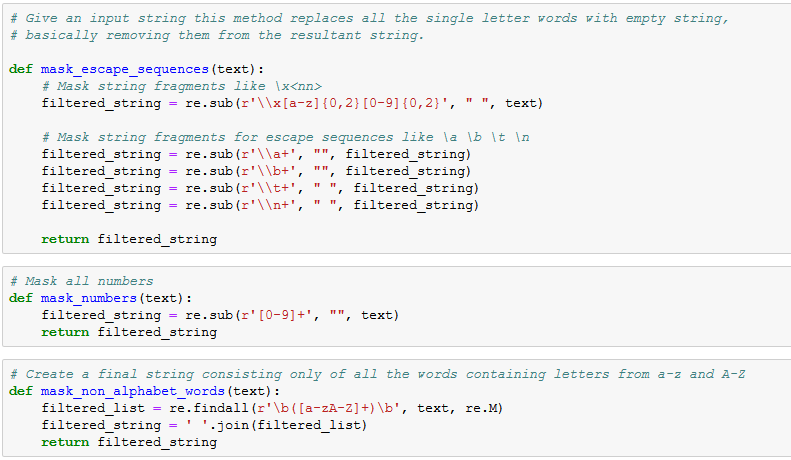
4. Replace all the single letter words with empty string

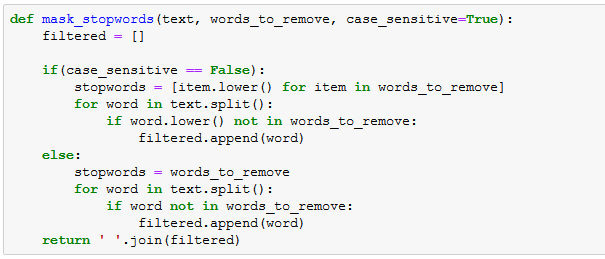
5. Mask email addresses

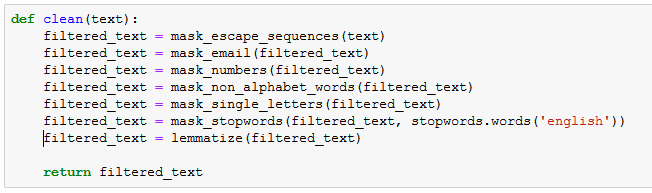
6. Stop words were masked from the dataset

7. Lemmatization

*Code Snippet*

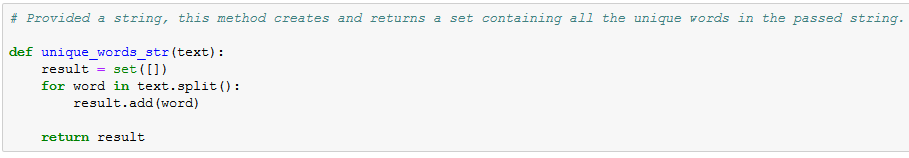


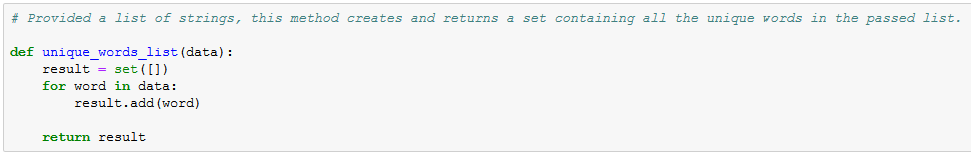




After cleansing the data all the unique words from the CV were extracted in to a list

*Code Snippet*





Post cleansing the data was saved into a new csv file named *resume\_cleaned.csv*

Cleansing of data has been done in ***raw\_resume\_processor.ipynb***

# Feature Extraction

The cleansed data was imported and feature extraction was carried out using TFIDF

The classifiers and learning algorithms cannot directly process the text documents in their original form, as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length. Therefore, during the preprocessing step, the texts are converted to a more manageable representation.

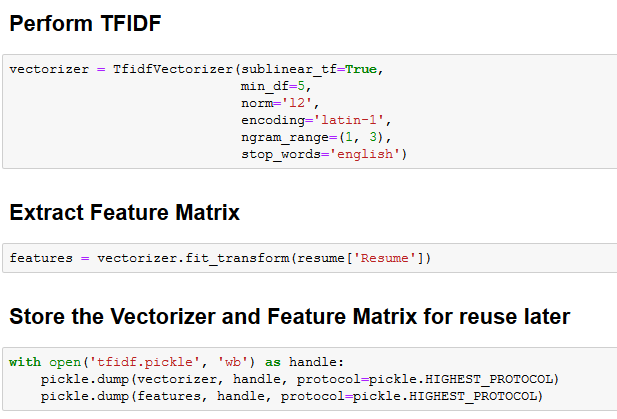
One common approach for extracting features from text is to use the bag of words model: a model where for each document, a complaint narrative in our case, the presence (and often the frequency) of words is taken into consideration, but the order in which they occur is ignored.

Specifically, for each term in our dataset, we will calculate a measure called Term Frequency, Inverse Document Frequency, abbreviated to tf-idf.

We are using sklearn.feature\_extraction.text.TfidfVectorizer to calculate a tf-idf vector

* sublinear\_df is set to True to use a logarithmic form for frequency
* min\_df is the minimum numbers of documents a word must be present in to be kept
* norm is set to l2, to ensure all our feature vectors have a euclidian norm of 1
* gram\_range is set to (1, 2) to indicate that we want to consider both unigrams and bigrams
* stop\_words is set to "english" to remove all common pronouns ("a", "the", ...) to reduce the number of noisy features.

*Code Snippet*



Feature extraction has been done in ***Featurization.ipynb***

# Model Building and Model Selection

Two models have been build on the cleansed data

1. Classification -

Based on the resume and category the model has been designed to categories the resume in right category

2. Recommendation -

The model would create summary of the resume and job description provided by the recruiter and give the list of most relevant resume based on the similarity between resume and jobs description

**Classification**

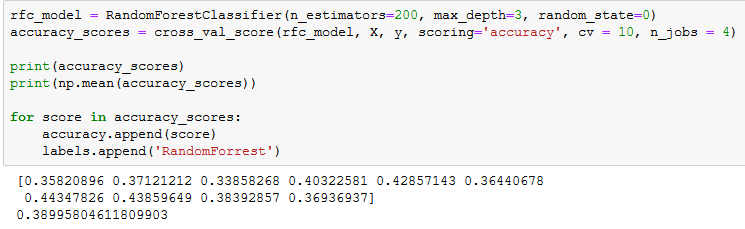
The classification was done using 4 different models and their accuracy score were recorded.

*Models*

1. Random Forest
2. Multinomial Naive Bayes
3. Logistic Regression
4. Linear Support Vector Classifier

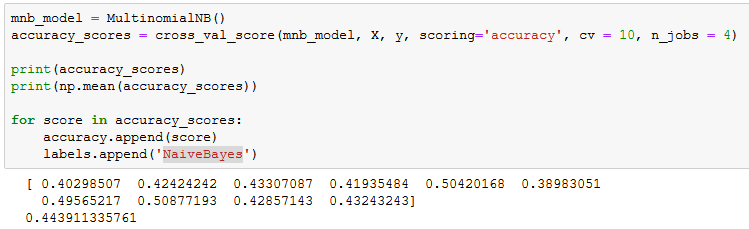
***Random Forest***

Random forest is an ensemble learning method for [classification](https://en.wikipedia.org/wiki/Statistical_classification) that operates by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) of the individual trees.

******

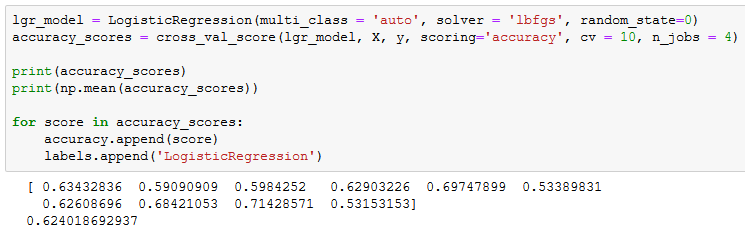
***Multinomial Naive Bayes***

Naive Bayes classifiers are a family of simple "[probabilistic classifiers](https://en.wikipedia.org/wiki/Probabilistic_classifier)" based on [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with strong [independence](https://en.wikipedia.org/wiki/Statistical_independence) assumptions between the features.

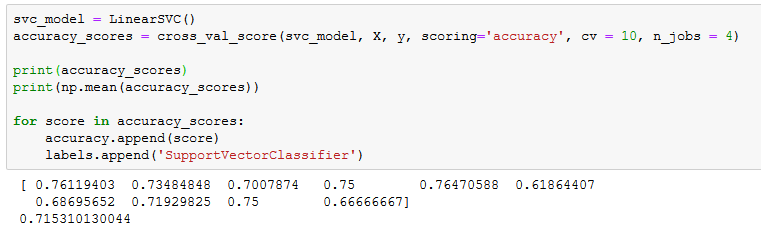


***Logistic Regression***

Logistic Regression uses a [logistic function](https://en.wikipedia.org/wiki/Logistic_function) to model a [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable)

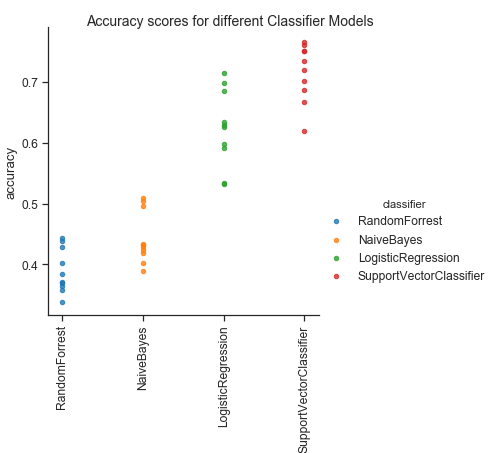


***Linear Support Vector Classifier***



Accuracy of all the models was calculated using cross validation. The average accuracy of each model is

1. Random Forest - 0.3899
2. Multinomial Naive Bayes - 0.4439
3. Logistic Regression - 0.6240
4. Linear Support Vector Classifier - 0.7153



The accuracy score of Linear Support Vector Classifier higher compared to other models have we found this model to reliable and best fit for the resume classification.

The code for the classification models is in *Classification.ipynb*

**CV Recommendation Model**

The recommendation model is designed to take job description and CVs as input and provide the list of CVs which are closest to the provided job description

This is done using two approaches

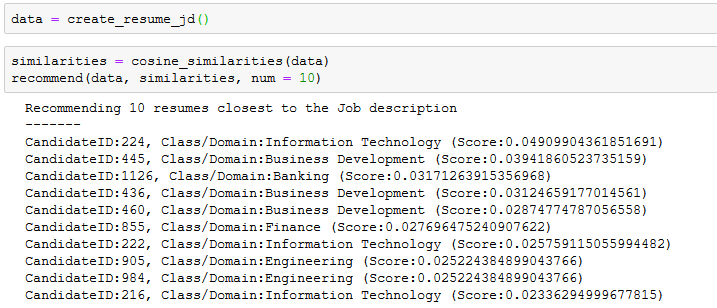
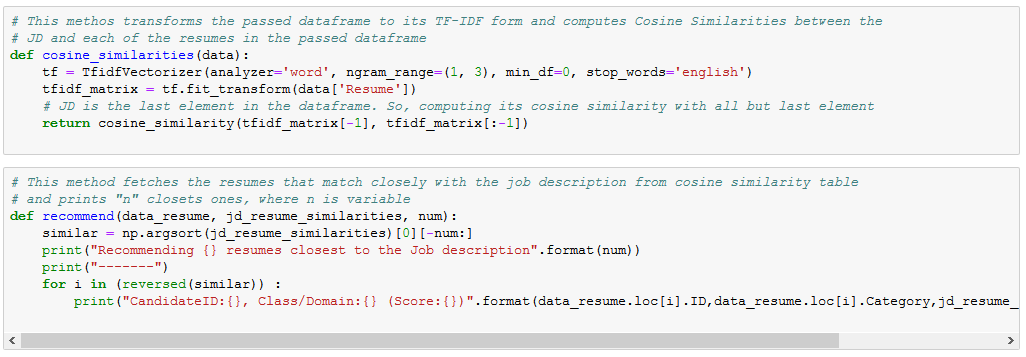
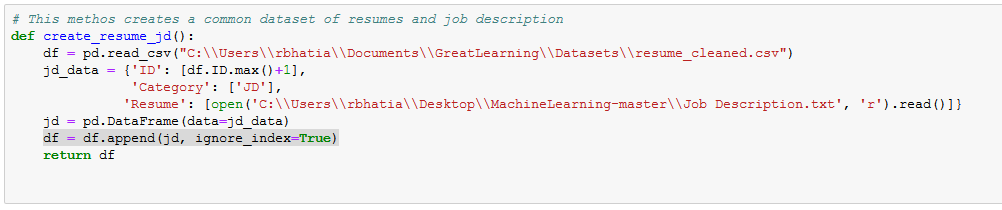
1. Content Based Recommendation (Cosine Similarity)

2. k-nearest neighbours

***Content Based Recommender***

Considering this is the case of document similarity identification, we have gone with the Content based recommender where Job Description provided by the employer is matched with the content of resumes in the space and the top n (n being configurable) matching resumes are recommended to the recruiter. The model takes the cleansed resume data and job description and combines the two into a single data set, and then computes **cosine similarity** between the job description and CVs.

10 CVs with highest score are given as recommendations by the model

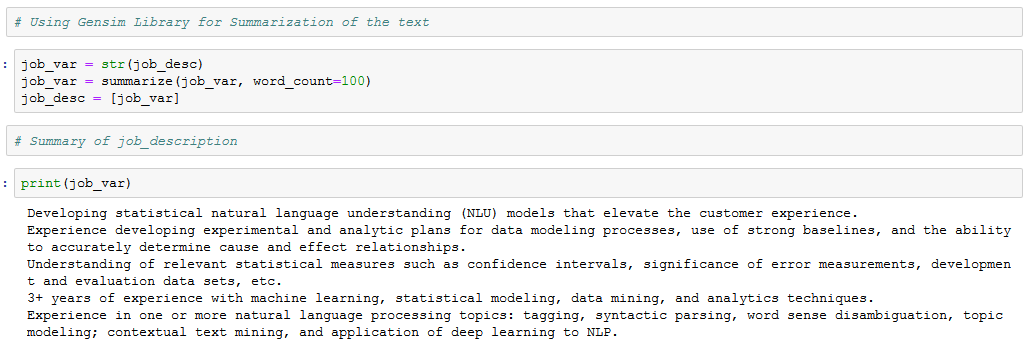


**k-nearest neighbours**

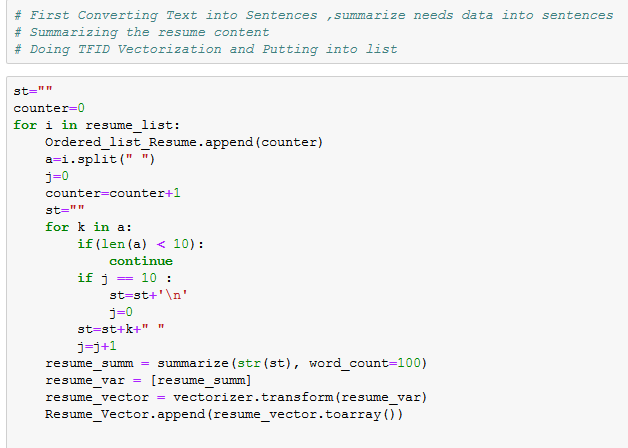
In this model k-NN is used to identify the CVs that are nearest to the provide job description, in other words the CVs that are close match to the provided job description. First to get the JD and CVs to similar scale, we have used an open source library called "gensim", this library generates the summary of the provided text in the provided word limit.

So to get the JD and CVs to similar word scale this library was used to generate summary of JD and CVs and then K-NN was applied to find the CVs which are closely matching the provided JD.

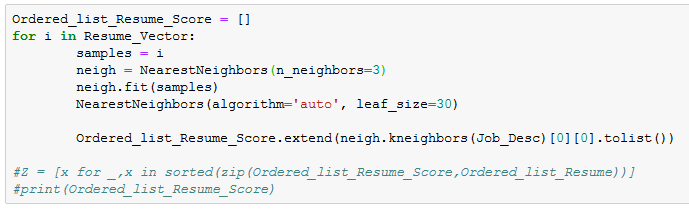
*Code Snippet for creating summary of JD*



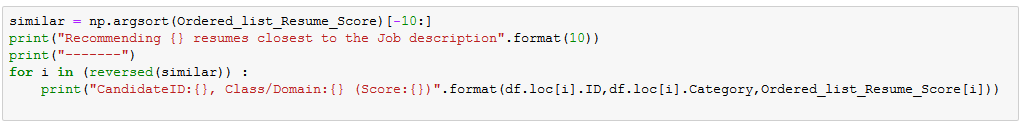
*Code Snippet for creating summary of CVs*



*Code Snippet for identifying the CVs closest to job description*



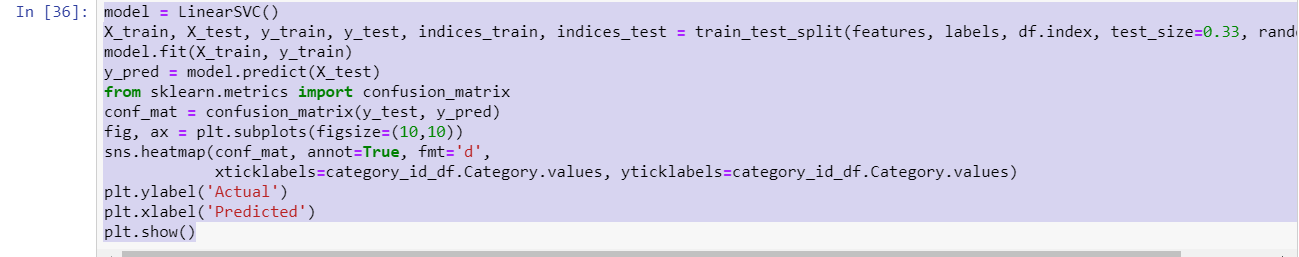
*Code Snippet for giving list of CVs closest to job description*

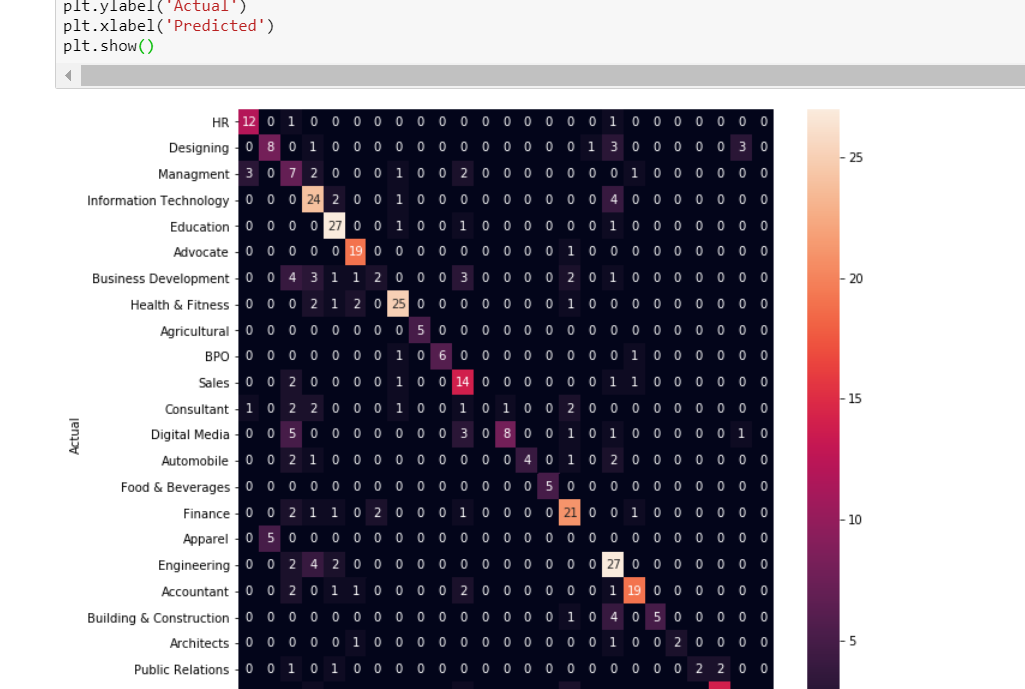


The recommendation can help recruiter to easily identify the CVs which are best match to the job description they have provided.

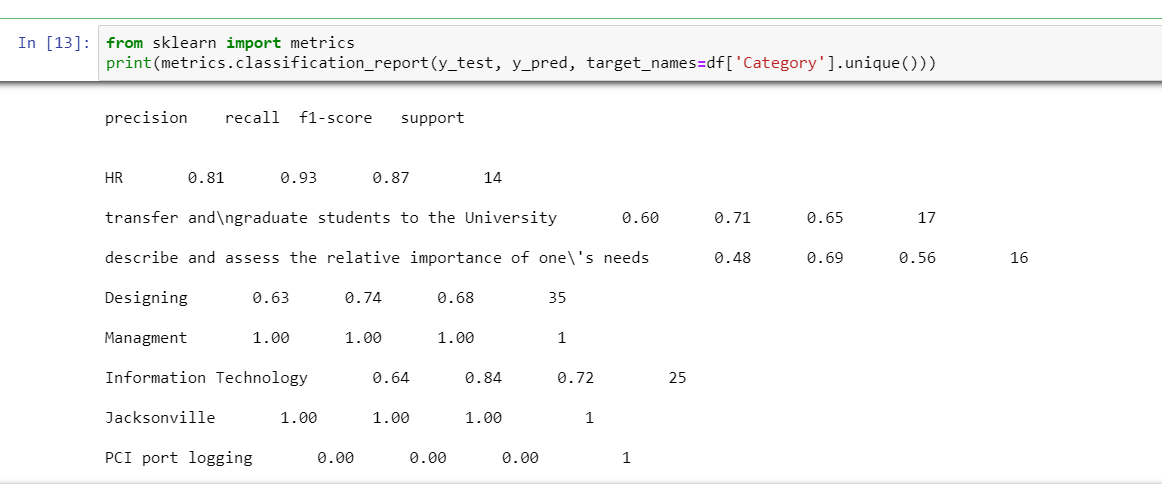
# Model Evaluation

Continue with our best model (LinearSVC), we are going to look at the confusion matrix, and show the discrepancies between predicted and actual labels. This is for the classification scope of our problem.





Finally, we print out the classification report for each class:



# Implication

The model designed is best suited for the first level of screening of the resumes by recruiter. This would help the recruiter to classify the resumes as per the requirements and easily identify the CVs that are best match to the job description.

Model would assist the recruiter in hastening the profile shortlisting, at the same time ensuring credibility of the shortlisting process, as they would be able to screen thousands of resumes very quickly, and with right fit, which would not have been possible for a human to do in near real time. This would aid in making the recruitment process efficient and very effective in identifying the right talent. Also, this would help recruiter to reduce the resources spent in identifying the right talent making the process cost effective.

On second level, the model provides the ranking to the CVs as per their fit vis-a-vis the job description, making it easier for the recruiter by giving the resume list in order of their relevant to the job.

The recommendation made by the model are currently for the varied industry but the model and be further enhanced to target specific industry which would make it more effective, and give better recommendations.

# Limitations

There are few limitation to the model design as of now, but these can be overcome by having more data to train the model.

The current limitation of the model are

1. Model takes CVs in CSV format, but in real world the CVs are either in .doc, .pdf etc format. Due to limitation of the data set, the model could not be enhanced to take .doc or .pdf as input, but using a library "textract" this can be achieved. The library can read varied file format and convert them in to single format which can be used as input to the model

2. The library "gensim" which is being used for generating summary for CVs and JDs, might cause the important feature to be lost in the conversion. To have the better result, custom library should be built, which ensure that while creating summary, important feature of data like candidate skill and experience are not lost

# Closing Reflections

In the process of designing the model, we have learnt about different ways of cleansing the data and how we can convert raw data lacking any structure and extract features from it to create a model.

Also, while working on the model we have been exposed to the possibility of using unstructured text data for creating a model.

If we get chance to further enhance the model, we would like to use deep learning techniques to see if we can achieve better results. Also, would like to get more data for single industry and try and create industry focused model, which might perform better and have better utility in the domain.

# References

[1] S. T. Al-Otaibi and M. Ykhlef, “A survey of job recommender systems,” International Journal of Physical Sciences

[2] J. K. Tarus, Z. Niu, and A. Yousif, “A hybrid knowledge based recommender system for e-learning based on ontology and sequential pattern mining,”

[3] Jie Chen, Chunxia Zhang, and Zhendong Niu “A Two-Step Resume Information Extraction Algorithm”