# Course Project

**Title :** ResumeAi – A Resume Parser and Classifier

Paladugu Reshvanth - RA1911030010099

# INDEX

**No Particulars PgNo.**

1. Abstract 3
2. Issues in existing system 3
3. Problem Statement 4
4. Objective 4
5. Proposed methodology- Diagramatic representation 5
6. Codes 6
7. Demo and screenshot 9
8. Proof of GitHub upload 10
9. Purpose of problem statement (societal benefit) 10
10. References 11

**Date :** 18-04-2022

**Title :** ResumAi – A Resume Parser and Classifier

## PROJECT ABSTRACT:

**ResuméAI** is a categorization and summarization tool for resumes. Given a resume in PDF format, our model extracts any form of text from the pdf using Optical Character Recognition (OCR).

This text was cleaned by transformation to a better encoding scheme, removal of punctuation marks, stop words, and everything was reduced to lower case to avoid redundancy. Following the rigorous cleaning, the text was passed to our model for classification, which accurately predicts the category of resume.

The overall output of our model would be the class of the input resume along with a summarized version of the resume which pulls out only the significant features of the resume.

For the process of classification, a range of models were tried out ranging from Multinomial Naive Bayes, Support Vector Machines, Logistic Regression. After repeated cleaning and training, our model was seen to perform fairly well with an overall accuracy of 70% using Support Vector Classifier.

## ISSUES IN EXISTING SYSTEM:

The current system created has a few issues that plaques it, they are:

1. The system has no conceivable Frontend, i.e there exists no interface for which a user can easily interact with.
2. There are multiple pipelines laid out throughout the project for which data has to be passed through in order to be fully processed.
3. The whole system is a bit slow because there are multiple inlets and outlets for which the data has to be passed through and be processed.

## PROBLEM STATEMENT:

Every day over a million job openings are posted, and on average around 250 candidates send their resumes/CV to apply for each job opening available. From this data that means every day on average 250 million resumes are sent.

Usually companies have multiple job opening spread out across multiple departments, therefore it can be difficult for them to manually categorize the resumes sent to them to their appropriate department.

Bigger companies usually have an ATS(Automatic Tracking System) to scan and parse resumes to help shortlist the candidates who applied.

But most other companies don’t have such services and even the ones that do have an ATS have noticed that they aren’t very accurate at times.

This can be a problem as it could mean that worthwhile candidates could not get shortlisted.

## OBJECTIVE:

* ResuméAI aims to help combat this predicament by offering a cheaper and far simpler yet accurate system.
* Its very small in size, i.e it doesn’t take up much storage space and can immediately be run with very minimal setup.
* It even goes as far as to predict the personality of the candidate as well thus making it easier for the HRs to assess the candidate as to whether they are a good fit for the company or not.

## PROPOSED METHODOLOGY – DIAGRAMMATIC REPRESENTATION:

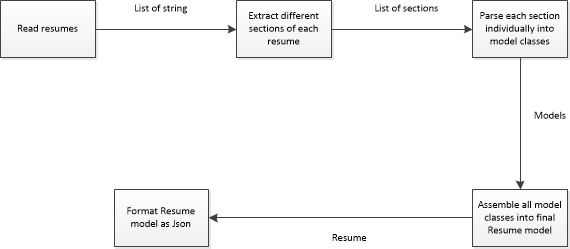
* Proposed Pipeline



* Steps of execution of the combined tool



* Flow of execution of the Parser



## CODES:

**Modelling.ipynb**

from sklearn.linear\_model import LogisticRegression from sklearn.ensemble import RandomForestClassifier from sklearn.svm import LinearSVC

from sklearn.model\_selection import cross\_val\_score models = [

RandomForestClassifier(n\_estimators=200, max\_depth=3, random\_state=0), LinearSVC(),

MultinomialNB(), LogisticRegression(random\_state=0),

]

CV = 5

cv\_df = pd.DataFrame(index=range(CV \* len(models))) entries = []

for model in models:

model\_name = model. class\_\_. name

accuracies = cross\_val\_score(model, features, labels, scoring='accuracy', cv=CV) for fold\_idx, accuracy in enumerate(accuracies):

entries.append((model\_name, fold\_idx, accuracy))

cv\_df = pd.DataFrame(entries, columns=['model\_name', 'fold\_idx', 'accuracy'])

import seaborn as sns

import matplotlib.pyplot as plt

sns.boxplot(x='model\_name', y='accuracy', data=cv\_df) sns.stripplot(x='model\_name', y='accuracy', data=cv\_df,

size=8, jitter=True, edgecolor="gray", linewidth=2) plt.show()

cv\_df.groupby('model\_name').accuracy.mean()

model = LinearSVC()

X\_train, X\_test, y\_train, y\_test, indices\_train, indices\_test = train\_test\_split(features, labels, df.index, test\_size=0.30, random\_state=0)

model.fit(X\_train, y\_train) y\_pred = model.predict(X\_test)

from sklearn.metrics import confusion\_matrix conf\_mat = confusion\_matrix(y\_test, y\_pred) fig, ax = plt.subplots(figsize=(10,10)) sns.heatmap(conf\_mat, annot=True, fmt='d',

xticklabels=category\_id\_df.Category.values, yticklabels=category\_id\_df.Category.values) plt.ylabel('Actual')

plt.xlabel('Predicted') plt.show()

## svm.ipynb

vectorizer= CountVectorizer()

res\_counts= vectorizer.fit\_transform(res\_train)

tfidf= TfidfTransformer()

res\_tfidf= tfidf.fit\_transform(res\_counts)

from sklearn.svm import SVC classifier= SVC(kernel='linear') classifier.fit(res\_counts, cat\_train)

predicted = [] for i in res\_test:

predicted.append((classifier.predict(vectorizer.transform([i])))[0])

a= pd.DataFrame(cat\_test) a['predicted'] = predicted a.tail()

## pdftotext.ipynb:

def convertPDFtoText(path): rsrcmgr = PDFResourceManager() retstr = StringIO()

codec = 'utf-8'

laparams = LAParams()

device = TextConverter(rsrcmgr, retstr, codec=codec, laparams=laparams) fp = open(path, 'rb')

interpreter = PDFPageInterpreter(rsrcmgr, device) password = ""

maxpages = 0 caching = True pagenos=set()

for page in PDFPage.get\_pages(fp, pagenos, maxpages=maxpages, password=password,caching=caching, check\_extractable=True):

interpreter.process\_page(page) fp.close()

device.close()

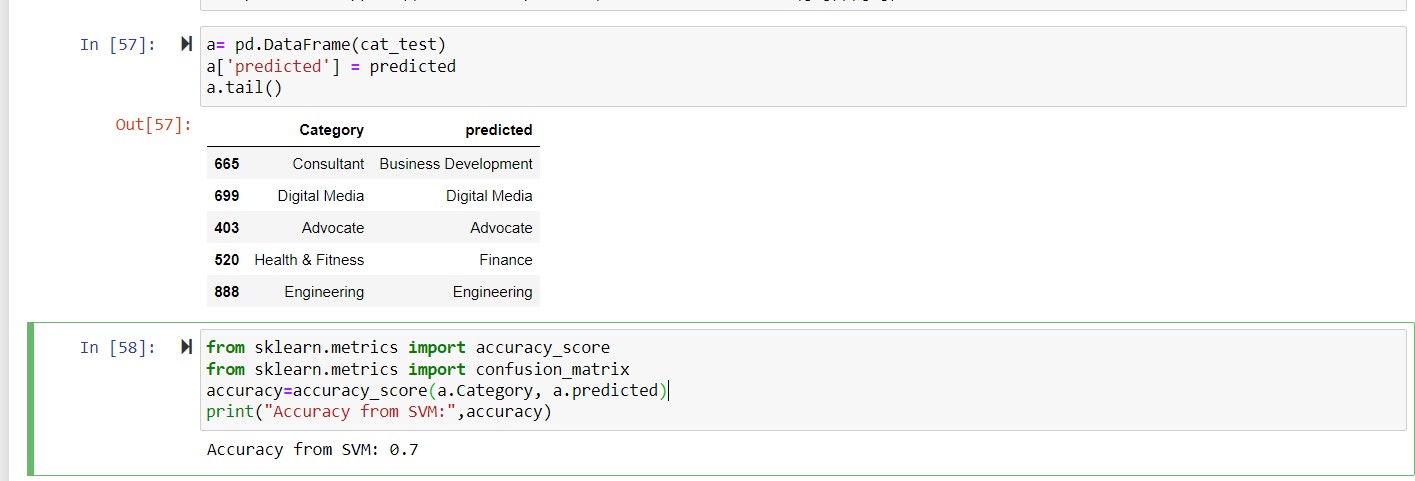
string = retstr.getvalue() retstr.close()

return string s=convertPDFtoText("/home/ganesh/ACADEMICS/PROJECTS/ResumeRise/sam ple\_input.pdf")

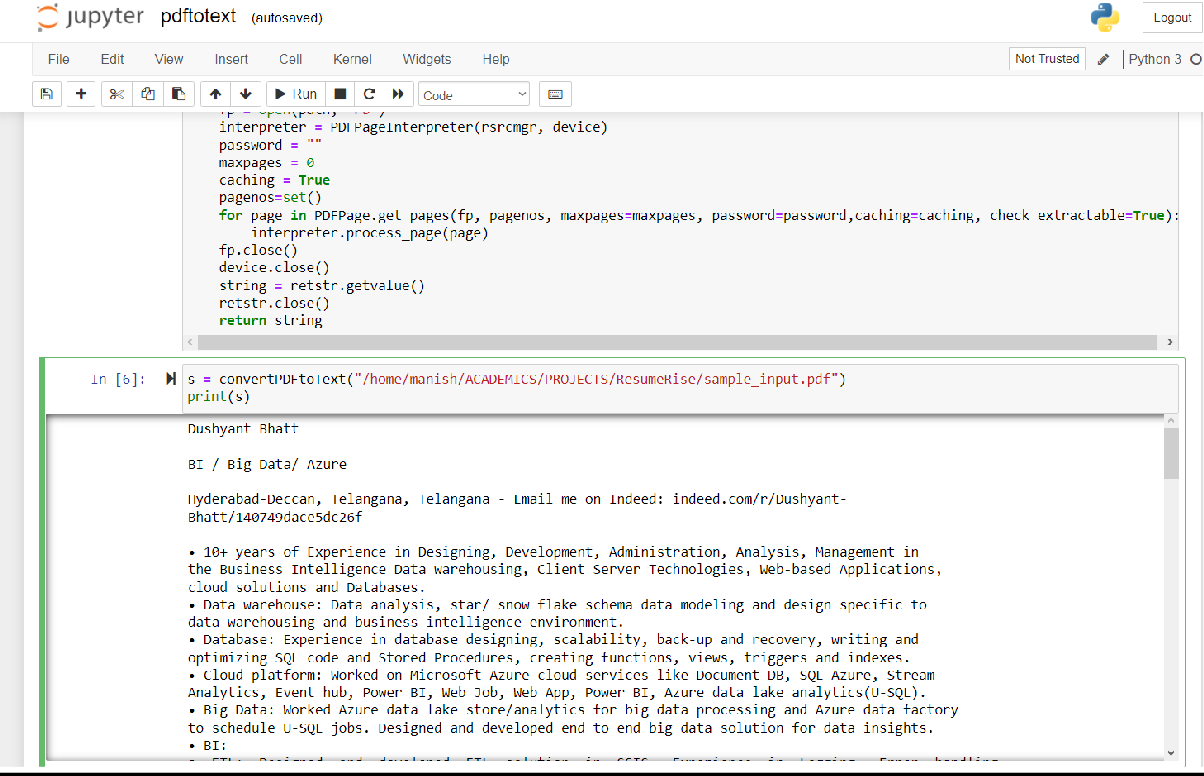
print(s)

## DEMO & SCREENSHOT:

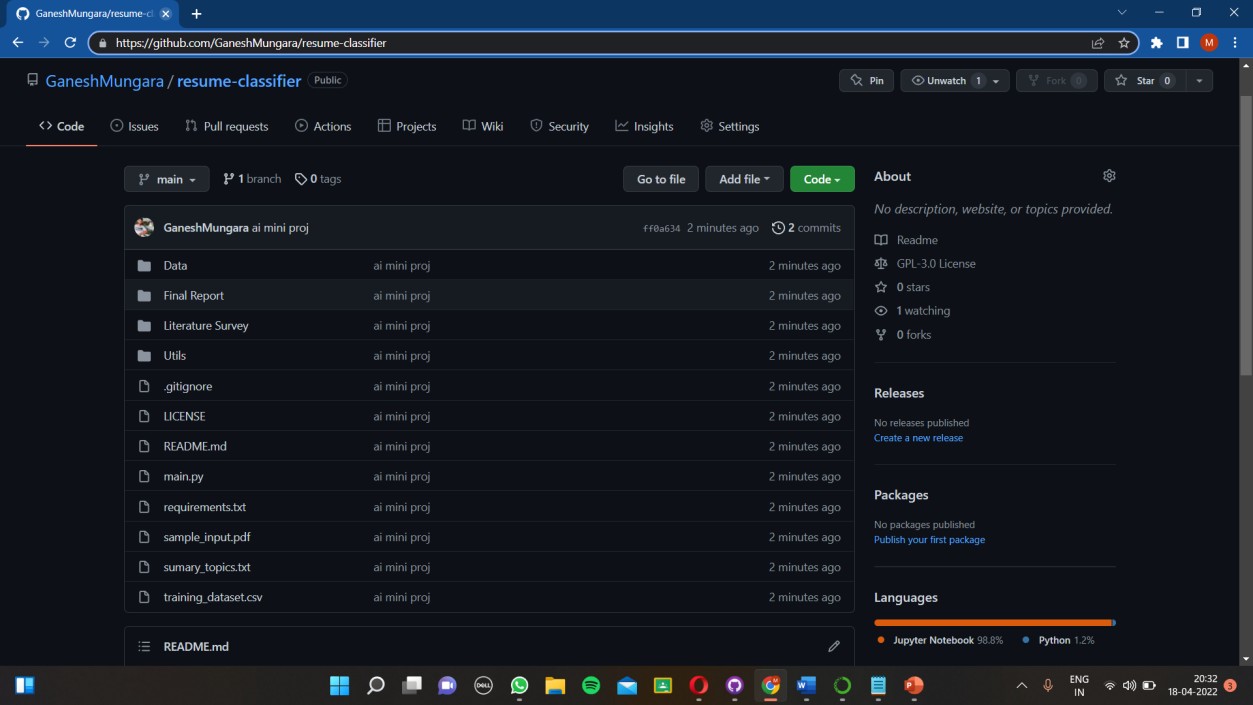
* ResuméAI has categorised the uploaded resume in their following categories with an accuracy of 70%.
* Multiple models have been implemented, and out of them SVM(Support Vector Machines) proved to be the most accurate.



* The entire Resume (or) CV has been summarised in a short paragraph making it easy for the recruiters to quickly go through the important information of the candidates.



## PROOF OF GITHUB UPLOAD:



Repository Link: <https://github.com/GaneshMungara/resume-classifier>

**SOCIETAL BENEFIT:**

ResumAI has a lot of implications on societal benefits:

* Its low memory space, affordability and high overall accuracy makes it an appealing choice for companies that receives a lot of resumes sent to them especially ones that do not have the capability to expend extra resources.
* It in turn helps a lot of candidates as it helps shortlist and select those that are deserving to be selected without leaving much room for error.
* It also helps the recruiters make their jobs easier as they do not need to extensively interview and grind the candidates which in turn helps save time, resources and money for the companies as well.

**REFERENCES:**

Dataset used:

The [dataset](https:/[/www.kaggle.com/iammhaseeb/resumes](http://www.kaggle.com/iammhaseeb/resumes-dataset-with-labels))-[dataset-with-labels)](http://www.kaggle.com/iammhaseeb/resumes-dataset-with-labels)) consists of 1000 labelled resumes (labelled according to the primary category/class that a particular resume belongs to) in a csv format. We will be using this csv formatted resume dataset to train our model for classification. Our model should then be able to work on any unseen resume.

Files for reference:

-Utils/Analysis.ipynb\*\* : Data cleaning + Pre-processing + Visualizations + Insights

-Utils/Summarize.ipynb\*\* : Resume Summarization algorithm

-Utils/pdftotext.ipynb\*\* : odf to text conversion using pdfminer

-Utils/Modelling.ipynb\*\* :main file + representational changes + training + comparison of models + testing

-Utils/naive\_bayes.ipynb\*\* : multinomial naive bayes implementation

-Utils/svm.ipynb\*\* : svm implementation

-Utils/clean\_data1.csv\*\*: cleaned resume dataset