RESUME CLASSIFICATION PROJECT

- Simon Ravula

Business Objective

The selection of a suitable job applicant from the pool of thousands applications is often daunting job for an employer. The categorization of job applications submitted in form of Resumes against available vacancy(s) takes significant time and efforts of an employer. Thus, Resume Classification System (RCS) using the Natural Language Processing (NLP) and Machine Learning (ML) techniques could automate this tedious process. Moreover, the automation of this process can significantly expedite and transparent the applicants' screening process with mere human involvement.

Steps Involved

- Data Extraction
- Text Cleaning
- EDA
- Model Building
- Model Evaluation
- Deployment

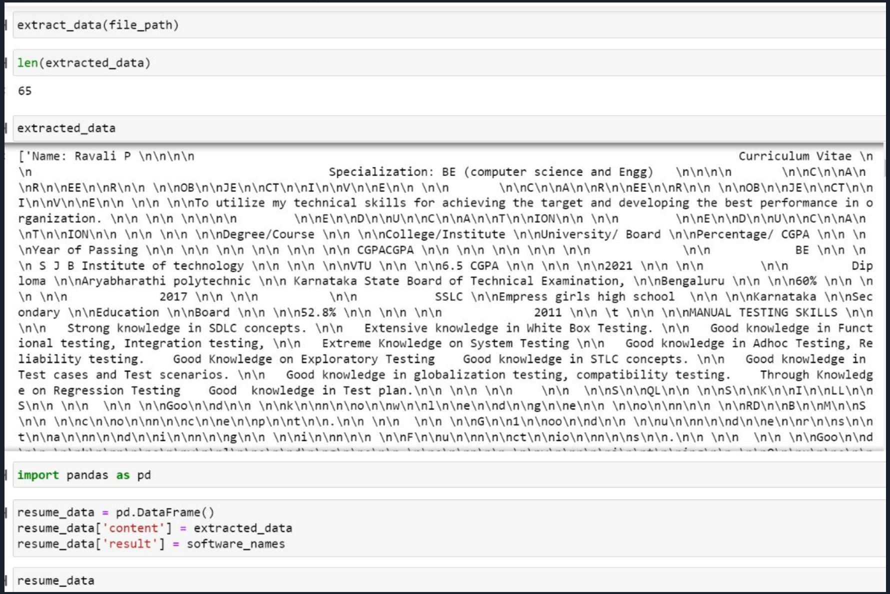
Data Extraction

We can extract the data using different methods like the conventional file handling techniques in python or upload directly from google drive. In both cases we neutralize the different extensions which the files have, like pdf, docx, etc., so that we will be able to work on all the files in a generalized manner.

```
import os
import docx2txt

file_path = r"C:\Users\simon\EXCELR\Project222\Resumes"
extracted_data = []
software_names = []
def extract_data(file_path):
    for file in os.listdir(file_path):
        if file == 'developer':
            final = os.path.join(file_path, file)
            for data in os.listdir(final):
                if data.endswith('.docx') :
                    final_path = os.path.join(final, data)
                    extracted_data.append(docx2txt.process(final_path))
                    software_names.append(file)
        elif file == 'Peoplesoft resumes':
            final = os.path.join(file_path, file)
            for data in os.listdir(final) :
                if data.endswith('.docx') :
                    final_path = os.path.join(final, data)
                    extracted_data.append(docx2txt.process(final_path))
                    software_names.append(file)
        elif file == 'SQL Developer Lightning insight' :
            final = os.path.join(file_path, file)
            for data in os.listdir(final) :
                if data.endswith('.docx'):
                    final_path = os.path.join(final, data)
                extracted_data.append(docx2txt.process(final_path))
                software_names.append(file)
        elif file == 'workday resumes':
            final = os.path.join(file_path, file)
            for data in os.listdir(final) :
                if data.endswith('.docx'):
                    final_path = os.path.join(final, data)
                extracted_data.append(docx2txt.process(final_path))
                software_names.append(file)
```

```
file_path = []
category = []
directory = '/content/drive/MyDrive/Resumes'
for i in os.listdir(directory):
    if i.endswith('.docx') or i.endswith('.doc') or i.endswith('.pdf'):
      os.path.join(directory, i)
      file_path.append((textract.process(os.path.join(directory, i))).decode('utf-8'))
      category.append('React JS Developer')
file_path, category

file_path_1 = []
category_1 = []
directory_1 = '/content/drive/MyDrive/Resumes/Peoplesoft resumes'
for i in os.listdir(directory_1):
    if i.endswith('.docx') or i.endswith('.doc') or i.endswith('.pdf'):
      os.path.join(directory_1, i)
      file_path_1.append((textract.process(os.path.join(directory_1, i))).decode('utf-8'))
      category_1.append('Peoplesoft resumes')
file_path_1, category_1

file_path_2 = []
category_2= []
directory_2 = '/content/drive/MyDrive/Resumes/SQL Developer Lightning insight'
for i in os.listdir(directory_2):
    if i.endswith('.docx') or i.endswith('.doc') or i.endswith('.pdf'):
      os.path.join(directory_2, i)
      file_path_2.append((textract.process(os.path.join(directory_2, i))).decode('utf-8'))
      category_2.append('SQL Developer Lightning insight')
file_path_2, category_2

file_path_3 = []
category_3 = []
directory_3 = '/content/drive/MyDrive/Resumes/workday resumes'
for i in os.listdir(directory_3):
    if i.endswith('.docx') or i.endswith('.doc') or i.endswith('.pdf'):
      os.path.join(directory_3, i)
      file_path_3.append((textract.process(os.path.join(directory_3, i))).decode('utf-8'))
```

We the convert the entire extracted data into a CSV format and put it all into a dataframe using pandas package. We put this data into a file, resume.csv. The libraries used are displayed in the screenshot below too.



```
extract_data(file_path)
```

```
len(extracted_data)
```

65

```
extracted_data
```

```
['Name: Ravali P \n\n\n\n                                  Curriculum Vitae \n
\n                          Specialization: BE (computer science and Engg)   \n\n\n\n      \n\nC\n\nA\n
\nR\n\nEE\n\nR\n\n \n\nOB\n\nJE\n\nCT\n\nI\n\nV\n\nE\n\n \n\n       \n\nC\n\nA\n\nR\n\nEE\n\nR\n\n \n\nOB\n\nJE\n\nCT\n\n
I\n\nV\n\nE\n\n \n\n \n\nTo utilize my technical skills for achieving the target and developing the best performance in o
rganization. \n\n \n\n \n\n\n\n       \n\nE\n\nD\n\nU\n\nC\n\nA\n\nT\n\nION\n\n \n\n       \n\nE\n\nD\n\nU\n\nC\n\nA\n
\nT\n\nION\n\n \n\n \n\n \n\nDegree/Course \n\n \n\nCollege/Institute \n\nUniversity/ Board \n\nPercentage/ CGPA \n\n \n
\nYear of Passing \n\n \n\n \n\n \n\n \n\n \n\n CGPACGPA \n\n \n\n \n\n \n\n \n\n            \n\n           BE \n\n \n
\n S J B Institute of technology \n\n \n\n \n\nVTU \n\n \n\n6.5 CGPA \n\n \n\n \n\n2021 \n\n \n\n      \n\n        Dip
loma \n\nAryabharathi polytechnic \n\n Karnataka State Board of Technical Examination, \n\nBengaluru \n\n \n\n60% \n\n \n
\n \n\n           2017 \n\n \n\n         \n\n          SSLC \n\nEmpress girls high school  \n\n \n\nKarnataka \n\nSec
ondary \n\nEducation \n\nBoard \n\n \n\n52.8% \n\n \n\n \n\n           2011 \n\n \t \n\n \n\nMANUAL TESTING SKILLS \n\n
\n\n   Strong knowledge in SDLC concepts. \n\n   Extensive knowledge in White Box Testing. \n\n   Good knowledge in Funct
ional testing, Integration testing, \n\n   Extreme Knowledge on System Testing \n\n   Good knowledge in Adhoc Testing, Re
liability testing.   Good Knowledge on Exploratory Testing   Good knowledge in STLC concepts. \n\n   Good knowledge in
Test cases and Test scenarios. \n\n   Good knowledge in globalization testing, compatibility testing.   Through Knowledg
e on Regression Testing    Good  knowledge in Test plan.\n\n \n\n \n\n    \n\n  \n\nS\n\nQL\n\n \n\nS\n\nK\n\nI\n\nLL\n\n
S\n\n \n\n  \n\n \n\nGoo\n\nd\n\n  \n\nk\n\nn\n\nno\n\nw\n\nl\n\ne\n\nd\n\ng\n\ne\n\n \n\no\n\nn\n\n \n\nRD\n\nB\n\nM\n\nS
\n\n \n\nc\n\no\n\nn\n\nc\n\ne\n\np\n\nt\n\n.\n\n \n\n  \n\n \n\nG\n\n1\n\noo\n\nd\n\n \n\nu\n\nn\n\nd\n\ne\n\nr\n\ns\n\n
t\n\na\n\nn\n\nd\n\ni\n\nn\n\ng\n\n \n\ni\n\nn\n\n \n\nF\n\nu\n\nn\n\nct\n\nio\n\nn\n\ns\n\n.\n\n \n\n  \n\n \n\nGoo\n\nd
```

```
import pandas as pd
```

```
resume_data = pd.DataFrame()
resume_data['content'] = extracted_data
resume_data['result'] = software_names
```

```
resume_data
```

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from catboost import CatBoostClassifier
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

```
resume_content = pd.read_csv('resume.csv',encoding = 'utf-8')
resume_content
```

	Unnamed: 0	content	result
0	0	Name: Ravali P \n\n\n\n ...	developer
1	1	SUSOVAN BAG \n\nSeeking a challenging po...	developer
2	2	Kanumuru Deepak Reddy\n\n\n\n\t\n\n\n\nCAREER ...	developer
3	3	HARIPRIYA BATTINA \n\nExperience as UI Develop...	developer
4	4	KAMALAKAR REDDY. A \n\nLinked In: https://www....	developer
...	...	...	...
60	60	Ramesh A\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t...	workday resumes
61	61	Srikanth (WORKDAY hCM Consultant)\t ...	workday resumes
62	62	WORKDAY | HCM | FCM\n\nName \t\t: Kumar S.S\n\...	workday resumes
63	63	WORKDAY | HCM | FCM\n\nName \t\t: Kumar S.S\n\...	workday resumes
64	64	Vinay kumar .v\nWorkday Functional Consultant\...	workday resumes

65 rows × 3 columns

Text Cleaning

Using re, string, nltk and different processing techniques, we clean the data which involves steps like lemmatization, tokenization, neutralizing lower and upper case letters, removing numeric and special characters, stopwords and other redundant characters.

Text cleaning

```
import re
import string
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
```

```
def preprocess(sentence):
    sentence=str(sentence)
    sentence = sentence.lower()
    sentence=sentence.replace('{html}',"")
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, '', sentence)
    rem_url=re.sub(r'http\S+', '',cleantext)
    rem_num = re.sub('[0-9]+', '', rem_url)
    tokenizer = RegexpTokenizer(r'\w+')
    tokens = tokenizer.tokenize(rem_num)
    filtered_words = [w for w in tokens if len(w) > 2 if not w in stopwords.words('english')]
    lemmatizer= WordNetLemmatizer()
    lemma_words=[lemmatizer.lemmatize(w) for w in filtered_words]
    return " ".join(lemma_words)
```

```
def lemmatise(sentence):
    tokenizer = RegexpTokenizer(r'\w+')
    tokens = tokenizer.tokenize(sentence)
    lemmatizer = WordNetLemmatizer()
    lemma_words=[lemmatizer.lemmatize(w) for w in tokens]
    return " ".join(lemma_words)
```

```
tokenizer = RegexpTokenizer(r'\w+')
resume_content['lemmatized_content']=resume_content['content'].map(lambda x : tokenizer.tokenize(lemmatise(x)))
resume_content['lemmatized_content']
```

Lemmatization and Vectorization

Lemmatization is a text pre-processing technique used in NLP models to break a word down to its root meaning to identify similarities. Word vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics.

```
words=resume_content['lemmatized_content']
allwords=[]
for wordlist in words:
    allwords+=wordlist
print(allwords)
print('length of all words : ',len(allwords))
```

```
['Name', 'Ravali', 'P', 'Curriculum', 'Vitae', 'Specialization', 'BE', 'computer', 'science', 'and'
'R', 'EE', 'R', 'OB', 'JE', 'CT', 'I', 'V', 'E', 'C', 'A', 'R', 'EE', 'R', 'OB', 'JE', 'CT', 'I', '
e', 'my', 'technical', 'skill', 'for', 'achieving', 'the', 'target', 'and', 'developing', 'the', 'b
'in', 'organization', 'E', 'D', 'U', 'C', 'A', 'T', 'ION', 'E', 'D', 'U', 'C', 'A', 'T', 'ION', 'De
ege', 'Institute', 'University', 'Board', 'Percentage', 'CGPA', 'Year', 'of', 'Passing', 'CGPACGPA'
'Institute', 'of', 'technology', 'VTU', '6', '5', 'CGPA', '2021', 'Diploma', 'Aryabharathi', 'polyt
'State', 'Board', 'of', 'Technical', 'Examination', 'Bengaluru', '60', '2017', 'SSLC', 'Empress', '
l', 'Karnataka', 'Secondary', 'Education', 'Board', '52', '8', '2011', 'MANUAL', 'TESTING', 'SKILLS
e', 'in', 'SDLC', 'concept', 'Extensive', 'knowledge', 'in', 'White', 'Box', 'Testing', 'Good', 'kn
ional', 'testing', 'Integration', 'testing', 'Extreme', 'Knowledge', 'on', 'System', 'Testing', 'Go
n', 'Adhoc', 'Testing', 'Reliability', 'testing', 'Good', 'Knowledge', 'on', 'Exploratory', 'Testin
e', 'in', 'STLC', 'concept', 'Good', 'knowledge', 'in', 'Test', 'case', 'and', 'Test', 'scenario',
'in', 'globalization', 'testing', 'compatibility', 'testing', 'Through', 'Knowledge', 'on', 'Regres
d', 'knowledge', 'in', 'Test', 'plan', 'S', 'QL', 'S', 'K', 'I', 'LL', 'S', 'Goo', 'd', 'k', 'n', '
'd', 'g', 'e', 'o', 'n', 'RD', 'B', 'M', 'S', 'c', 'o', 'n', 'c', 'e', 'p', 't', 'G', '1', 'oo', 'd
'r', 's', 't', 'a', 'n', 'd', 'i', 'n', 'g', 'i', 'n', 'F', 'u', 'n', 'ct', 'io', 'n', 's', 'Goo',
'w', 'l', 'e', 'd', 'g', 'e', 'o', 'n', 'w', 'r', 'i', 't', 'ing', 'Q', 'u', 'e', 'r', 'i', 'e', 's
'n', 'ow', 'l', 'e', 'dg', 'e', 'i', 'n', 'J', 'o', 'i', 'n', 's', 'Goo', 'd', 'k', 'n', 'o', 'w',
'e', 'o', 'n', 'S', 'u', 'bq', 'u', 'e', 'r', 'i', 'e', 's', 'E', 'xt', 'r', 'e', 'me', 'K', 'n', '
```

Vecotorizering

```
# transform the text to numeric data
tfidf_vect= TfidfVectorizer()
X_train_tfidf = tfidf_vect.fit(resume_content['content'])
```

```
X_train_tfidf_transform = X_train_tfidf.transform(resume_content['content'])
```

```
#SPLITTING THE TRAINING DATASET INTO TRAIN AND TEST
X_train, X_test, y_train, y_test = train_test_split(X_train_tfidf_transform,resume_content['result'],test_
```

```
print(X_train.shape,y_train.shape)
X_test.shape,y_test.shape
```

```
(45, 3602) (45,)
```

44]: ((20, 3602), (20,))

```
def accuracy(y_train,y_train_pred,y_test,y_test_pred):
    print('Train Accuracy\n')
    print(classification_report(y_train,y_train_pred))
    print('\n',confusion_matrix(y_train,y_train_pred))
    print('\n',accuracy_score(y_train,y_train_pred))
    print('*'*100)
    print('Test Accuracy\n')
    print(classification_report(y_test,y_test_pred))
    print('\n',confusion_matrix(y_test,y_test_pred))
    print('\n',accuracy_score(y_test,y_test_pred))
```

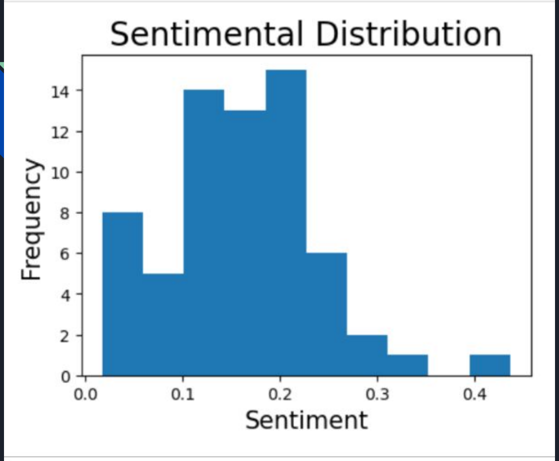
Exploratory Data Analysis

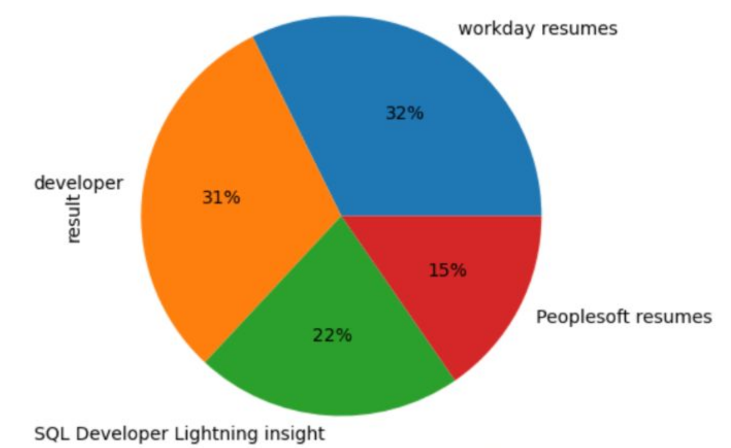
Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

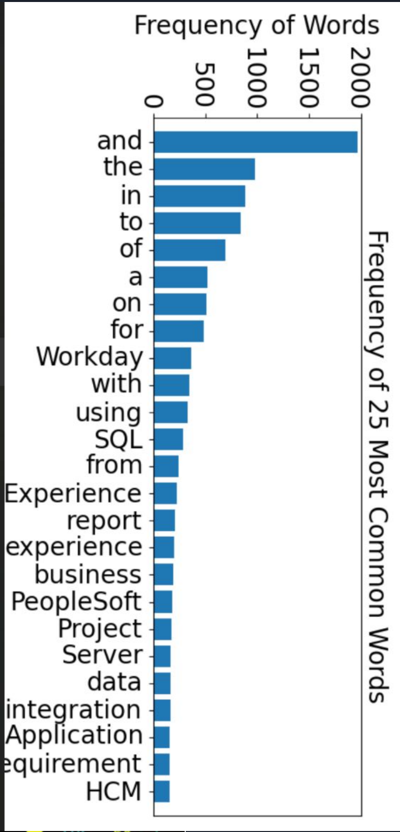
```
from textblob import TextBlob
import matplotlib.pyplot  as plt
import seaborn as sns
```

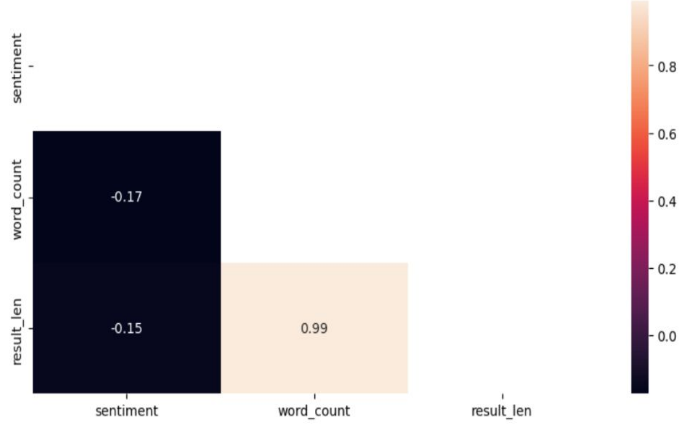
```
resume_content['sentiment']=resume_content['content'].apply(lambda x : TextBlob(x).sentiment.polarity)
resume_content
```

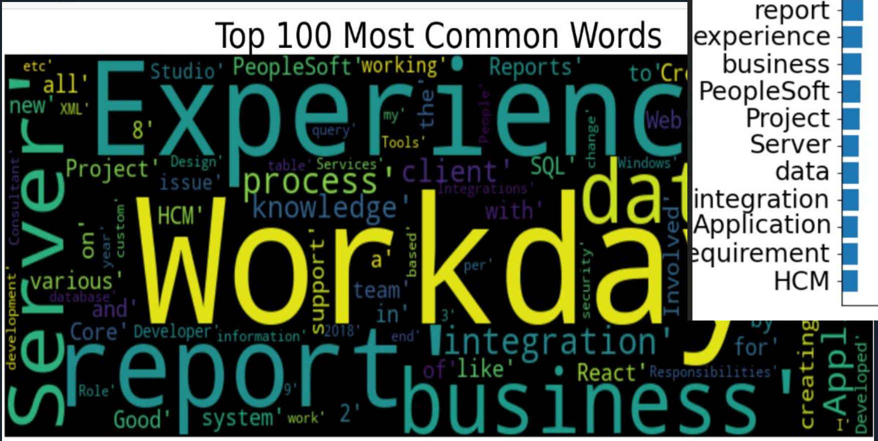
	content	result	lemmatized_content	sentiment
0	Name: Ravali P \n\n\n\n ...	developer	[Name, Ravali, P, Curriculum, Vitae, Specializ...	0.436385
1	SUSOVAN BAG \n\nSeeking a challenging po...	developer	[SUSOVAN, BAG, Seeking, a, challenging, positi...	0.346032
2	Kanumuru Deepak Reddy\n\n\n\n\t\n\n\n\nCAREER ...	developer	[Kanumuru, Deepak, Reddy, CAREER, OBJECTIVE, T...	0.223770
3	HARIPRIYA BATTINA \n\nExperience as UI Develop...	developer	[HARIPRIYA, BATTINA, Experience, a, UI, Develo...	0.202606
4	KAMALAKAR REDDY. A \n\nLinked In: https://www....	developer	[KAMALAKAR, REDDY, A, Linked, In, http, www, l...	0.121131
...	...	...	...	...
60	Ramesh A\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t...	workday resumes	[Ramesh, A, Workday, HCM, Consultant, PROFESSI...	0.176319
61	Srikanth (WORKDAY hCM Consultant)\t ...	workday resumes	[Srikanth, WORKDAY, hCM, Consultant, Seeking, ...	0.190324
62	WORKDAY | HCM | FCM\n\nName \t\t: Kumar S.S\n\...	workday resumes	[WORKDAY, HCM, FCM, Name, Kumar, S, S, Role, W...	0.162635
63	WORKDAY | HCM | FCM\n\nName \t\t: Kumar S.S\n\...	workday resumes	[WORKDAY, HCM, FCM, Name, Kumar, S, S, Role, W...	0.162635
64	Vinay kumar .v\nWorkday Functional Consultant\...	workday resumes	[Vinay, kumar, v, Workday, Functional, Consult...	0.195803


Sentimental Distribution
Frequency
Sentiment
0 2 4 6 8 10 12 14
0.0 0.1 0.2 0.3 0.4


workday resumes
32%
developer
result
31%
15%
22%
Peoplesoft resumes
SQL Developer Lightning insight


Frequency of Words
0 500 1000 1500 2000
and
the
in
to
of
a
on
for
Workday
with
using
SQL
from
Experience
report
experience
business
PeopleSoft
Project
Server
data
integration
Application
equirement
HCM
Frequency of 25 Most Common Words


sentiment
word_count
result_len
-0.17
-0.15
0.99
sentiment
word_count
result_len
0.8
0.6
0.4
0.2
0.0


Top 100 Most Common Words
Experience
Workday
report
business
Server
integration
process
knowledge
client
PeopleSoft
Reports
Studio
working
Project
HCM
various
support
team
Core
Developer
system
Good
SQL
React
Responsibilities
creating
Involved

Model Building

Building a model in machine learning is creating a mathematical representation by generalizing and learning from training data. We use different models here to determine the accuracy of the predicted data and select the model with the best accuracy.

1.LogisticRegression

```
lg = LogisticRegression()
lg.fit(X_train,y_train)
y_train_pred = lg.predict(X_train)
y_test_pred = lg.predict(X_test)
```

```
accuracy(y_train,y_train_pred,y_test,y_test_pred)
```

```
Train Accuracy

                               precision    recall  f1-score   support

             Peoplesoft resumes       0.00      0.00      0.00         4
SQL Developer Lightning insight       1.00      1.00      1.00         8
                      developer       0.81      1.00      0.89        17
                workday resumes       1.00      1.00      1.00        16

                       accuracy                           0.91        45
                      macro avg       0.70      0.75      0.72        45
                   weighted avg       0.84      0.91      0.87        45


 [[ 0  0  4  0]
 [ 0  8  0  0]
 [ 0  0 17  0]
 [ 0  0  0 16]]

 0.9111111111111111
**********
Test Accuracy

                               precision    recall  f1-score   support

             Peoplesoft resumes       0.00      0.00      0.00         6
SQL Developer Lightning insight       1.00      0.83      0.91         6
```

2.MultinomialNB

```
nb= MultinomialNB()
nb.fit(X_train,y_train)
y_train_pred = nb.predict(X_train)
y_test_pred = nb.predict(X_test)
```

```
accuracy(y_train,y_train_pred,y_test,y_test_pred)
```

```
Train Accuracy

                               precision    recall  f1-score   support

             Peoplesoft resumes       0.00      0.00      0.00         4
SQL Developer Lightning insight       1.00      0.62      0.77         8
                      developer       1.00      1.00      1.00        17
                workday resumes       0.70      1.00      0.82        16

                       accuracy                           0.84        45
                      macro avg       0.67      0.66      0.65        45
                   weighted avg       0.80      0.84      0.81        45


 [[ 0  0  0  4]
 [ 0  5  0  3]
 [ 0  0 17  0]
 [ 0  0  0 16]]

 0.8444444444444444
**********
Test Accuracy

                               precision    recall  f1-score   support

             Peoplesoft resumes       0.00      0.00      0.00         6
SQL Developer Lightning insight       1.00      0.33      0.50         6
```

3.RandomForestClassifier

```
rf = RandomForestClassifier(max_features=400)
rf.fit(X_train,y_train)
y_train_pred = rf.predict(X_train)
y_test_pred = rf.predict(X_test)
```

```
accuracy(y_train,y_train_pred,y_test,y_test_pred)
```

```
Train Accuracy

                                precision    recall  f1-score   support

                  Peoplesoft resumes       1.00      1.00      1.00         4
SQL Developer Lightning insight            1.00      1.00      1.00         8
                       developer           1.00      1.00      1.00        17
                    workday resumes        1.00      1.00      1.00        16

                           accuracy                             1.00        45
                          macro avg        1.00      1.00      1.00        45
                       weighted avg        1.00      1.00      1.00        45


[[ 4  0  0  0]
 [ 0  8  0  0]
 [ 0  0 17  0]
 [ 0  0  0 16]]

 1.0
**********************************************************************************
Test Accuracy

                                precision    recall  f1-score   support

                  Peoplesoft resumes       1.00      0.33      0.50         6
SQL Developer Lightning insight            1.00      1.00      1.00         6
```

Light Gradient Boosting Classifier

```
lgb_clf = LGBMClassifier()
lgb_clf.fit(X_train, y_train)
prediction_8 = lgb_clf.predict(X_test)
print('Accuracy of LightGradientBoosting Classifier on training set: {:.4f}'.format(lgb_clf.score(X_train, y_train)))
print('Accuracy of LightGradientBoosting Classifier on test set     : {:.4f}'.format(lgb_clf.score(X_test, y_test)))
```

```
Accuracy of LightGradientBoosting Classifier on training set: 1.0000
Accuracy of LightGradientBoosting Classifier on test set     : 0.8333
```

```
print("\n Classification report for Light Gradient Boosting Classifier %s:\n%s\n" % (lgb_clf, metrics.classification_report(y
```

```
 Classification report for Light Gradient Boosting Classifier LGBMClassifier():
              precision    recall  f1-score   support

           0       1.00      0.83      0.91         6
           1       0.70      1.00      0.82         7
           2       0.50      0.25      0.33         4
           3       1.00      1.00      1.00         7

    accuracy                           0.83        24
   macro avg       0.80      0.77      0.77        24
weighted avg       0.83      0.83      0.81        24
```

```
accuracy_8 = round(accuracy_score(y_test,prediction_8),4)
precision_8 = round(precision_score(y_test,prediction_8,average = 'macro'),4)
recall_8 = round(recall_score(y_test,prediction_8, average = 'macro'),4)
f1_score_8 = round(f1_score(y_test,prediction_8, average = 'macro'),4)
```

```
print('Accuracy        : ', accuracy_8)
print('Precision       : ', precision_8)
print('Recall Score    : ', recall_8)
print('f1-Score        : ', f1_score_8)
print('Confusion Matrix:\n',confusion_matrix(y_test,prediction_8))
```

```
Accuracy        :  0.8333
Precision       :  0.8
Recall Score    :  0.7708
f1-Score        :  0.7665
```

Xtreme Gradient Boosting Classifier

```
xgb_clf = XGBClassifier()
xgb_clf.fit(X_train, y_train)
prediction_7 = xgb_clf.predict(X_test)
print('Accuracy of XtremeGradientBoosting Classifier on training set: {:.4f}'.format(xgb_clf.score(X_train, y_train)))
print('Accuracy of XtremeGradientBoosting Classifier on test set     : {:.4f}'.format(xgb_clf.score(X_test, y_test)))
```

```
Accuracy of XtremeGradientBoosting Classifier on training set: 1.0000
Accuracy of XtremeGradientBoosting Classifier on test set     : 0.7500
```

```
print("\n Classification report for Xtreme GradientBoosting Classifier %s:\n%s\n" % (xgb_clf, metrics.classification_report(y
```

```
 Classification report for Xtreme GradientBoosting Classifier XGBClassifier(objective='multi:softprob'):
              precision    recall  f1-score   support

           0       0.71      0.83      0.77         6
           1       0.88      1.00      0.93         7
           2       1.00      0.25      0.40         4
           3       0.62      0.71      0.67         7

    accuracy                           0.75        24
   macro avg       0.80      0.70      0.69        24
weighted avg       0.78      0.75      0.73        24
```

```
accuracy_7 = round(accuracy_score(y_test,prediction_7),4)
precision_7 = round(precision_score(y_test,prediction_7,average = 'macro'),4)
recall_7 = round(recall_score(y_test,prediction_7, average = 'macro'),4)
f1_score_7 = round(f1_score(y_test,prediction_7, average = 'macro'),4)
```

```
print('Accuracy        : ', accuracy_7)
print('Precision       : ', precision_7)
print('Recall          : ', recall_7)
print('f1-Score        : ', f1_score_7)
print('Confusion Matrix:\n',confusion_matrix(y_test,prediction_7))
```

```
Accuracy        :  0.75
Precision       :  0.8036
Recall          :  0.6994
f1-Score        :  0.6923
```

Support Vector Classifier

```
sv_clf = SVC()
sv_clf.fit(X_train, y_train)
prediction_2 = sv_clf.predict(X_test)
print('Accuracy of Support Vector Classifier on training set: {:.2f}'.format(sv_clf.score(X_train, y_train)))
print('Accuracy of Support Vector Classifier on test set     : {:.2f}'.format(sv_clf.score(X_test, y_test)))
```

```
Accuracy of Support Vector Classifier on training set: 1.00
Accuracy of Support Vector Classifier on test set     : 0.92
```

```
print("\n Classification report for Support Vector Classifier %s:\n%s\n" % (sv_clf, metrics.classification_report(y_test, pre
```

```
 Classification report for Support Vector Classifier SVC():
              precision    recall  f1-score   support

           0       1.00      0.83      0.91         6
           1       0.78      1.00      0.88         7
           2       1.00      0.75      0.86         4
           3       1.00      1.00      1.00         7

    accuracy                           0.92        24
   macro avg       0.94      0.90      0.91        24
weighted avg       0.94      0.92      0.92        24
```

```
accuracy_2 = round(accuracy_score(y_test,prediction_2),4)
precision_2 = round(precision_score(y_test,prediction_2,average = 'macro'),4)
recall_2 = round(recall_score(y_test,prediction_2, average = 'macro'),4)
f1_score_2 = round(f1_score(y_test,prediction_2, average = 'macro'),4)
```

```
print('Accuracy        : ', accuracy_2)
print('Precision       : ', precision_2)
print('Recall          : ', recall_2)
print('f1-Score        : ', f1_score_2)
print('Confusion Matrix :\n',confusion_matrix(y_test,prediction_2))
```

```
Accuracy        :  0.9167
Precision       :  0.9444
Recall          :  0.8958
f1-Score        :  0.9103
```

4.GradientBoostingClassifier

```
gb = GradientBoostingClassifier()
gb.fit(X_train,y_train)
y_train_pred = gb.predict(X_train)
y_test_pred = gb.predict(X_test)
```

```
accuracy(y_train,y_train_pred,y_test,y_test_pred)
```

```
Train Accuracy

                                precision    recall  f1-score   support

                  Peoplesoft resumes       1.00      1.00      1.00         4
SQL Developer Lightning insight            1.00      1.00      1.00         8
                       developer           1.00      1.00      1.00        17
                    workday resumes        1.00      1.00      1.00        16

                           accuracy                             1.00        45
                          macro avg        1.00      1.00      1.00        45
                       weighted avg        1.00      1.00      1.00        45


[[ 4  0  0  0]
 [ 0  8  0  0]
 [ 0  0 17  0]
 [ 0  0  0 16]]

 1.0
**********************************************************************************
Test Accuracy

                                precision    recall  f1-score   support

                  Peoplesoft resumes       1.00      1.00      1.00         6
SQL Developer Lightning insight            1.00      1.00      1.00         6
                       developer           1.00      1.00      1.00         3
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier
ab_clf = AdaBoostClassifier()
ab_clf.fit(X_train, y_train)
prediction_5 = ab_clf.predict(X_test)
print('Accuracy of AdaBoost Classifier on training set: {:.4f}'.format(ab_clf.score(X_train, y_train)))
print('Accuracy of AdaBoost Classifier on test set     : {:.4f}'.format(ab_clf.score(X_test, y_test)))
```

```
Accuracy of AdaBoost Classifier on training set: 0.7358
Accuracy of AdaBoost Classifier on test set     : 0.6250
```

```
print("\n Classification report for AdaBoost Classifier %s:\n%s\n" % (ab_clf, metrics.classification_report(y_test, predictio
```

```
 Classification report for AdaBoost Classifier AdaBoostClassifier():
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         6
           1       0.50      1.00      0.67         7
           2       1.00      0.75      0.86         4
           3       1.00      0.71      0.83         7

    accuracy                           0.62        24
   macro avg       0.62      0.62      0.59        24
weighted avg       0.60      0.62      0.58        24
```

```
accuracy_5 = round(accuracy_score(y_test,prediction_5),4)
precision_5 = round(precision_score(y_test,prediction_5,average = 'macro'),4)
recall_5 = round(recall_score(y_test,prediction_5, average = 'macro'),4)
f1_score_5 = round(f1_score(y_test,prediction_5, average = 'macro'),4)
```

```
print('Accuracy        : ', accuracy_5)
print('Precision       : ', precision_5)
print('Recall          : ', recall_5)
print('f1-Score        : ', f1_score_5)
print('Confusion Matrix:\n',confusion_matrix(y_test,prediction_5))
```

```
Accuracy        :  0.625
Precision       :  0.625
Recall          :  0.6161
f1-Score        :  0.5893
```

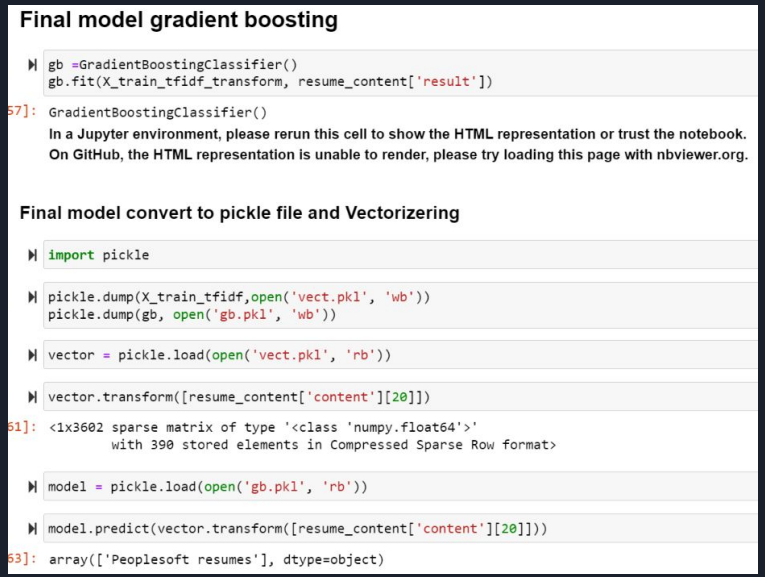
Model Evaluation

Model evaluation is the process that uses some metrics which help us to analyze the performance of the model. As we all know that model development is a multi-step process and a check should be kept on how well the model generalizes future predictions. Therefore evaluating a model plays a vital role so that we can judge the performance of our model. For our project we are going to use Gradient Boosting.

	Classifier	Accuracy	Precision	Recall	F1-Score
0	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000
1	SVM Classifier	0.9167	0.9444	0.8958	0.9103
2	Multinomial NB Classifier	0.9167	0.9375	0.8750	0.8833
3	Logistic Regression	0.9167	0.9444	0.8958	0.9103
4	AdaBoost Classifier	0.6250	0.6250	0.6161	0.5893
5	Gradient Boosting Classifier	0.9583	0.9643	0.9643	0.9615
6	Xtreme Gradient Boosting Classifier	0.7500	0.8036	0.6994	0.6923
7	Light Gradient Boosting Classifier	0.8333	0.8000	0.7708	0.7665

Deployment

Deploying a machine learning model, known as model deployment, simply means to integrate a machine learning model and integrate it into an existing production environment (1) where it can take in an input and return an output. First we convert our final model i.e., Gradient Boosting model and our vectorization codes into pickle files so that it is easier to access them during deployment.



Final model gradient boosting

```
gb =GradientBoostingClassifier()
gb.fit(X_train_tfidf_transform, resume_content['result'])
```

```
57]: GradientBoostingClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Final model convert to pickle file and Vectorizering

```
import pickle
```

```
pickle.dump(X_train_tfidf,open('vect.pkl', 'wb'))
pickle.dump(gb, open('gb.pkl', 'wb'))
```

```
vector = pickle.load(open('vect.pkl', 'rb'))
```

```
vector.transform([resume_content['content'][20]])
```

```
61]: <1x3602 sparse matrix of type '<class 'numpy.float64'>'
        with 390 stored elements in Compressed Sparse Row format>
```

```
model = pickle.load(open('gb.pkl', 'rb'))
```

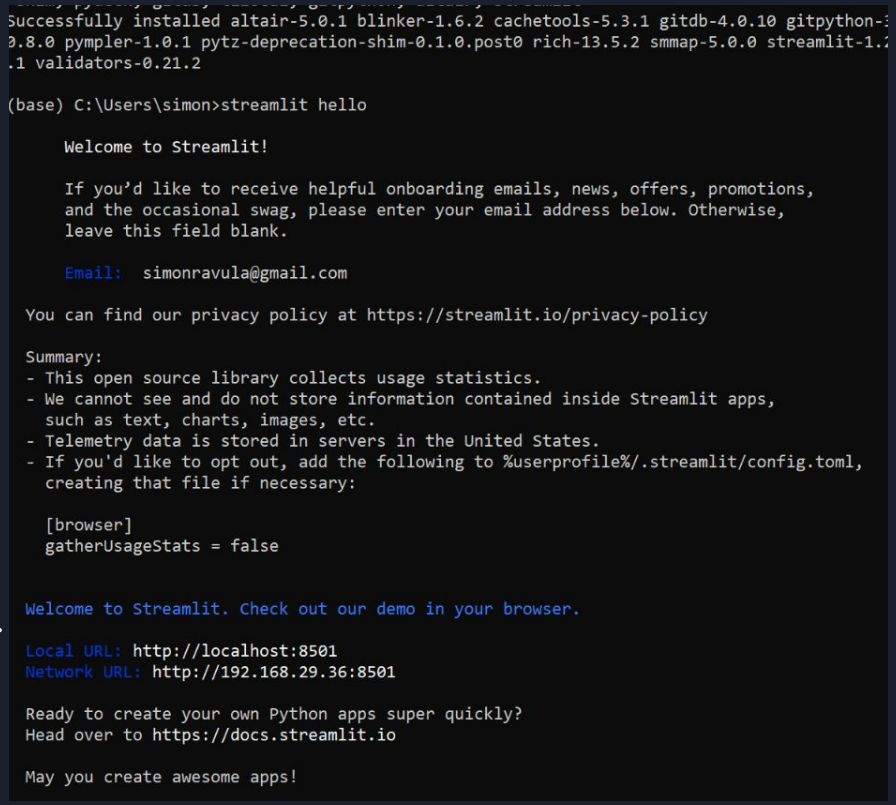
```
model.predict(vector.transform([resume_content['content'][20]]))
```

```
63]: array(['Peoplesoft resumes'], dtype=object)
```

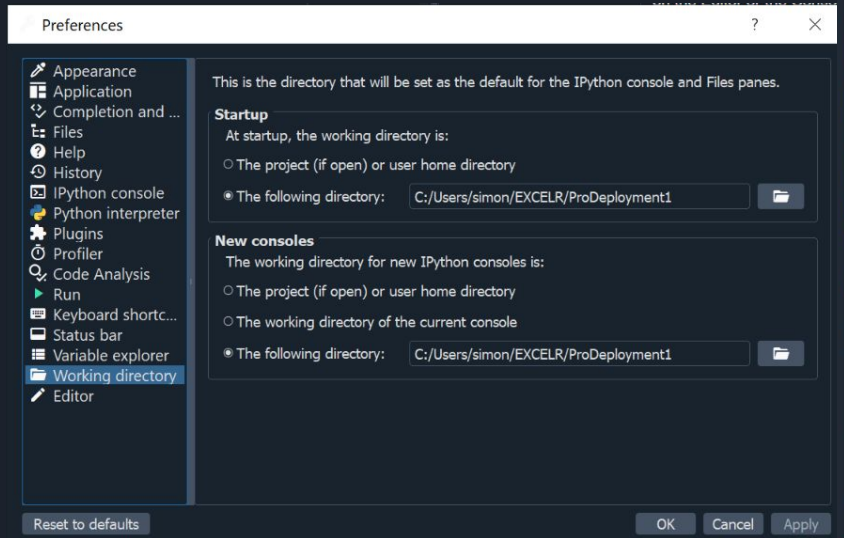
Working with Streamlit

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.
It is a Python-based library specifically designed for machine learning engineers.

- Install streamlit in the terminal prompt
- Verify the successful installation Streamlit and provide our email
- Write our deployment code in a separate file calling our pickled files
- Write our functions for different processes like data cleaning, preprocessing, most common words, etc.,.



We then provide to the directory our path where our code, pickle files are saved, import necessary libraries. We then call our pickle files into the program.




```
import pandas as pd
import streamlit as st
import docx2txt
import pdfplumber
import re
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.probability import FreqDist
from wordcloud import WordCloud, ImageColorGenerator
import matplotlib.pyplot  as plt
import plotly.express as px
stop=set(stopwords.words('english'))
import pickle
vectors = pickle.load(open('C://Users//simon//EXCELR//ProDeployment1//vect.pkl','rb'))
model = pickle.load(open('C://Users//simon//EXCELR//ProDeployment1//gb.pkl','rb'))
rf_model = pickle.load(open('C://Users//simon//EXCELR//ProDeployment1//gb.pkl','rb'))


nltk.download('wordnet')
nltk.download('stopwords')


resume = []

def display(doc_file):
    if doc_file.type == "application/vnd.openxmlformats-officedocument.wordprocessingml.
        resume.append(docx2txt.process(doc_file))
    else :
        with pdfplumber.open(doc_file) as pdf:
            pages=pdf.pages[0]
            resume.append(pages.extract_text())
    return resume
```

We write our functions for different preprocessing techniques and then the main function.

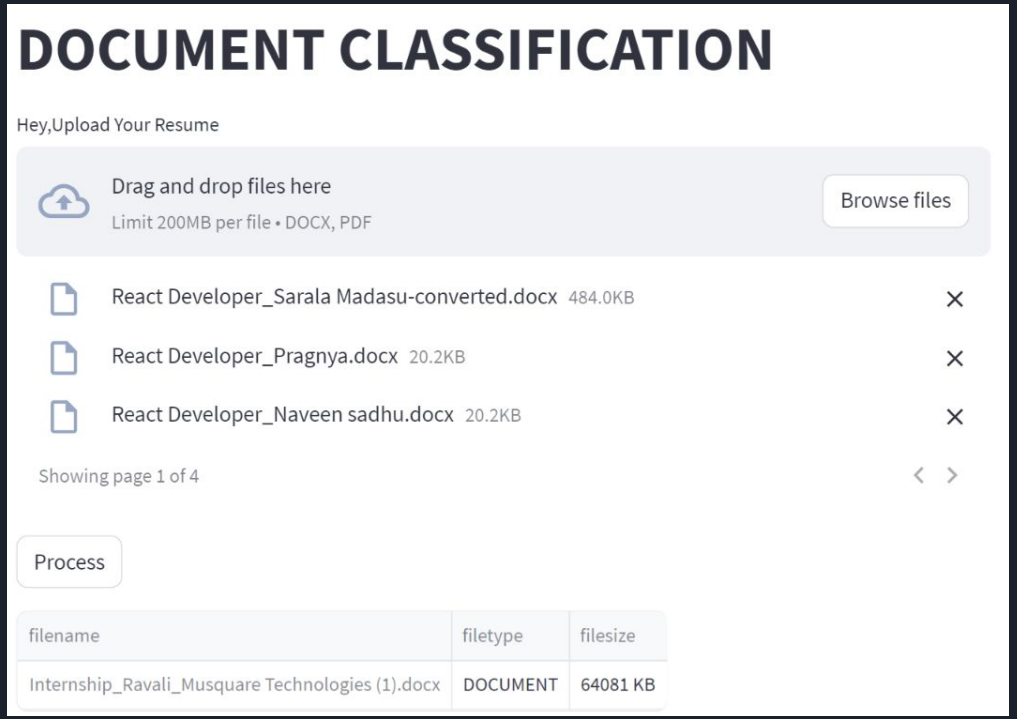
```python
def preprocess(sentence):
    sentence=str(sentence)
    sentence = sentence.lower()
    sentence=sentence.replace('{html}',"")
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, '', sentence)
    rem_url=re.sub(r'http\S+', '',cleantext)
    rem_num = re.sub('[0-9]+', '', rem_url)
    tokenizer = RegexpTokenizer(r'\w+')
    tokens = tokenizer.tokenize(rem_num)
    filtered_words = [w for w in tokens if len(w) > 2 if not w in stopwords.words('
    lemmatizer = WordNetLemmatizer()
    lemma_words=[lemmatizer.lemmatize(w) for w in filtered_words]
    return " ".join(lemma_words)
```

```python
def mostcommon_words(cleaned,i):
    tokenizer = RegexpTokenizer(r'\w+')
    words=tokenizer.tokenize(cleaned)
    mostcommon=FreqDist(cleaned.split()).most_common(i)
    return mostcommon

def display_wordcloud(mostcommon):
    wordcloud=WordCloud(width=1000, height=600, background_color='black').generate(
    a=px.imshow(wordcloud)
    st.plotly_chart(a)

def display_words(mostcommon_small):
    x,y=zip(*mostcommon_small)
    chart=pd.DataFrame({'keys': x,'values': y})
    fig=px.bar(chart,x=chart['keys'],y=chart['values'],height=700,width=700)
    st.plotly_chart(fig)
```

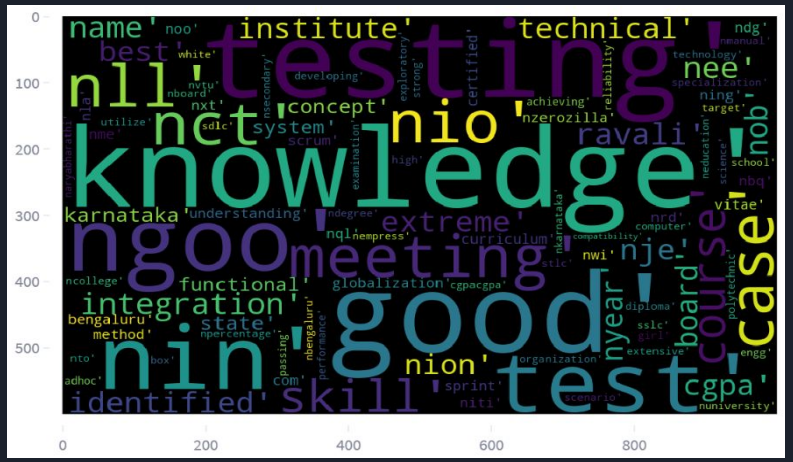
```python
def main():
    st.title('DOCUMENT CLASSIFICATION')
    upload_file = st.file_uploader('Hey,Upload Your Resume ',
                                   type= ['docx','pdf'],accept_multiple_files=True)
    if st.button("Process"):
        for doc_file in upload_file:
            if doc_file is not None:
                file_details = {'filename':[doc_file.name],
                                'filetype':doc_file.type.split('.')[-1].upper(),
                                'filesize':str(doc_file.size)+' KB'}
                file_type=pd.DataFrame(file_details)
                st.write(file_type.set_index('filename'))
                displayed=display(doc_file)

                cleaned=preprocess(display(doc_file))
                predicted= model.predict(vectors.transform([cleaned]))

                string='The Uploaded Resume is belongs to '+predicted[0]
                st.header(string)

                st.subheader('WORDCLOUD')
                display_wordcloud(mostcommon_words(cleaned,100))

                st.header('Frequency of 20 Most Common Words')
                display_words(mostcommon_words(cleaned,20))

if __name__ == '__main__':
    main()
```

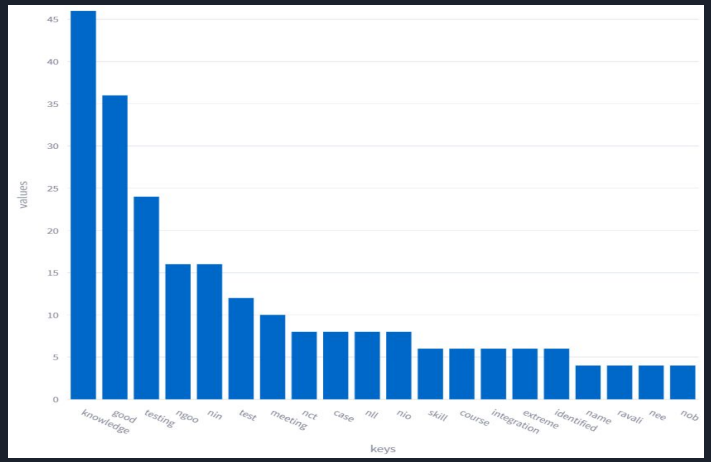
Output

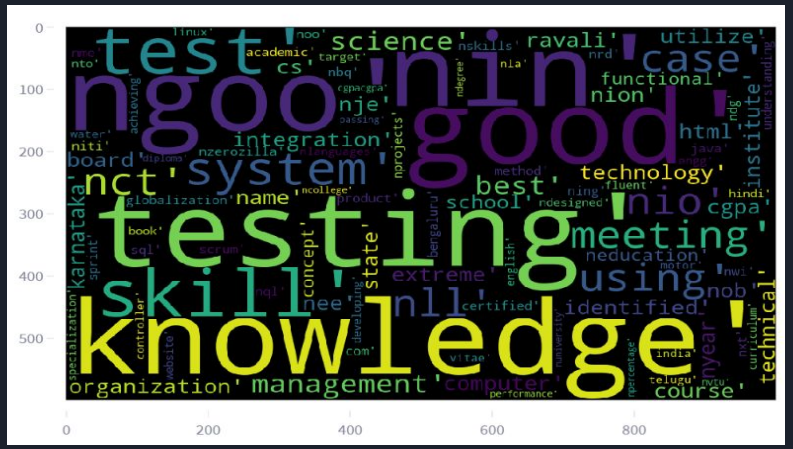
We enter the command streamlit run deployment222.py in the prompt window and it directs us to the webpage where we select files as input and then the output is displayed there.

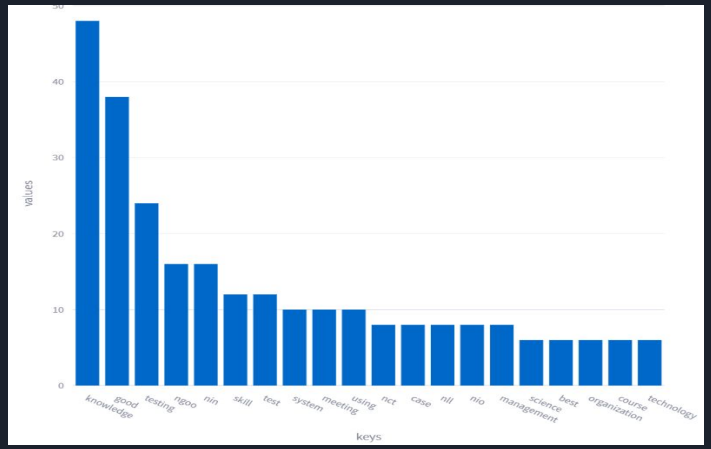


Wordcloud and frequency of Developer and Peoplesoft Resumes

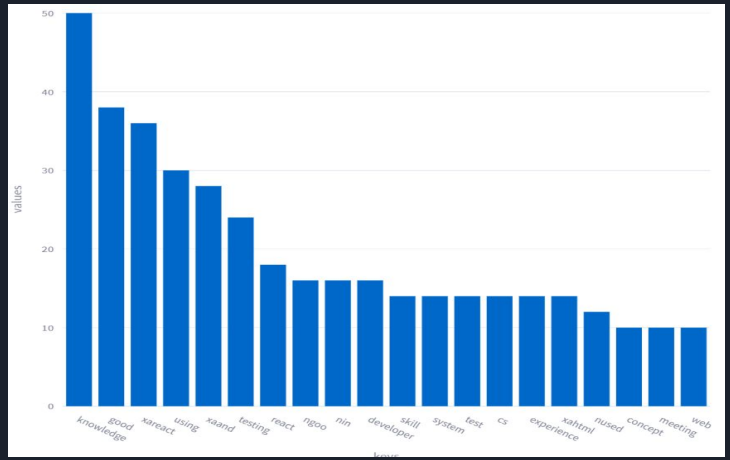


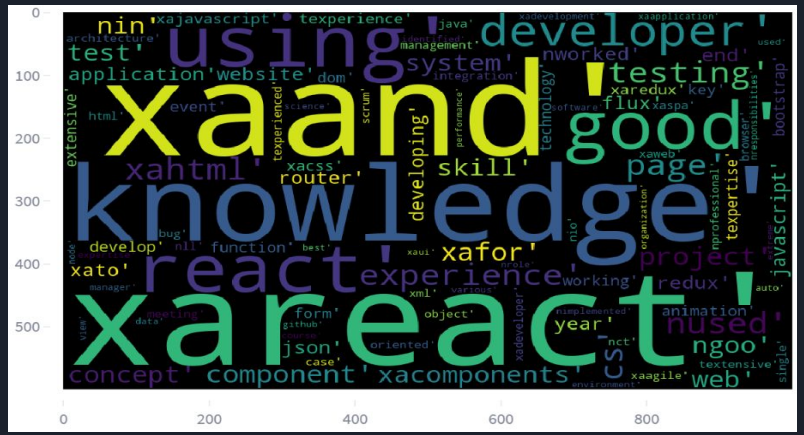


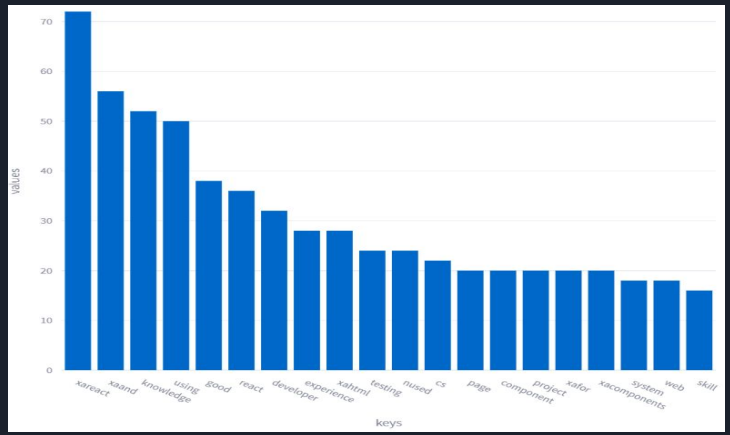




Wordcloud and Frequency of SQL Dev Lightning Insights and Workaday Resumes







Challenges Faced

- Problems with paths because all the files were in different folders. We have solved this issue by uploading all files and folders to Google drive and were able to access all of the data using Google colab.
- We couldn't use en_core_web_sm library spyder notebook directly in order to use features like POS tagging, Named Entity Recognition, etc.,. We had to install spaCy library which solved this error.
- Time taken to run text processing programs is high for just 70 files in our case, implying that it could take much longer if there are more files in a realistic world.
- Misspellings and same words with different meanings are hard to identify according to context.