B.M.S College of Engineering
P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Course – Machine Learning
Course Code –20IS5PCMLG
AY 2021-2022

Job Predictor

Submitted to – Ms. Shobana TS
Assistant Professor

Submitted by -
Samartha S(1BM19IS219)

B.M.S College of Engineering
P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Table of Content

ABSTRACT

Job Predictor is a Web application that helps a user by giving them the best suggestion based on their resume by using Machine Learning Techniques. Generally, in the industry we see a mismatch between what the hirer is looking for and what the candidate has to offer. Candidates tend to apply for roles that are not best suited for their resume. By uploading their resume on our web app, they will come to know which role they are best suited for. This application will specifically help college students find what they are looking for based on the projects they've worked on and the skills they have acquired.

INTRODUCTION

A lot of the services we receive today, especially on our gadgets, are automated at some point or the other. Be it shopping apps, food delivery apps, gaming applications, etc., all of them leverage Artificial Intelligence and Machine Learning to improve user experience. In the case of our application, users can easily upload their resume and find out which job role they are most suited for. This will be of great benefit for job seekers, especially first timers, to understand which profile suits their resume the best.

We make use of 4 models :- Logistic Regression, Stochastic Gradient Descent Classifier, k Nearest Neighbours and Support Vector Classifier. We've used a Voting Classifier to determine which job role is the most optimal for the given resume.

We have made use of Flask server to deploy the working model of our predictor to a simple and intuitive front end application. The front end was designed using HTML, CSS and JavaScript.

PROBLEM STATEMENT

In the industry, in general, we see a mismatch between what job seekers want and what the companies are looking for. We also understand that for freshers it might be hard to know what job role exactly they want to pursue. This web application is built to help the aforementioned parties and at the same time improve the company's productivity and output.

LITERATURE SURVEY

Many factors affect the success of Machine Learning on a given task. The representation and quality of the instance data is first and foremost. If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. It is well known that data preparation and filtering steps take a considerable amount of processing time in ML problems. Data pre-processing includes data cleaning, normalization, transformation, feature extraction and selection, etc. The product of data pre-processing is the final training set.

Steps Involved in Data Preprocessing:

1. Data Cleaning:
The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

- **(a) Missing Data:**
  This situation arises when some data is missing in the data. It can be handled in various ways.
  Some of them are:
  1. **Ignore the tuples:**
     This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

  2. **Fill the Missing values:**
     There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

- **(b) Noisy Data:**
  Noisy data is meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :
  1. **Binning Method:**
     This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can

be used to complete the task.

2. **Regression:**
   Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

3. **Clustering:**
   This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation:

1. **Normalization:**
   It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

2. **Attribute Selection:**
   In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

3. **Discretization:**
   This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

4. **Concept Hierarchy Generation:**
   Here attributes are converted from lower level to higher level in hierarchy. For Example-The attribute “city” can be converted to “country”.

3. Data Reduction:
The various steps to data reduction are:

1. **Data Cube Aggregation:**
   Aggregation operation is applied to data for the construction of the data cube.

2. **Attribute Subset Selection:**
   The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute.The attribute having p-value greater than significance level can be discarded.

3. **Numerosity Reduction:**
   This enables us to store the model of data instead of whole data, for example: Regression Models.

4. **Dimensionality Reduction:**
   This reduces the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction, else it is called lossy reduction. The two effective methods of dimensionality reduction are:Wavelet transforms and PCA (Principal Component Analysis).

The k-Nearest-Neighbours is a non-parametric classification method, which is simple but effective in many cases . For a data record t to be classified, its k nearest neighbours are retrieved, and this forms a neighbourhood of t. Majority voting among the data records in the neighbourhood is usually used to decide the classification for t with or without consideration of distance-based weighting. However, to apply kNN we need to choose an appropriate value for k, and the success of classification is very much dependent on this value. In a sense, the kNN method is biased by k. There are many ways of choosing the k value, but a simple one is to run the algorithm many times with different k values and choose the one with the best performance.

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of K number of neighbors
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

The Support Vector Machine (SVM) was first proposed by Vapnik and has since attracted a high degree of interest in the machine learning research community. Several recent studies have reported that the SVM (support vector machines) generally are capable of delivering higher performance in terms of classification accuracy than the other data classification algorithms.

SVMs are a set of related supervised learning methods used for classification and regression They belong to a family of generalized linear classification. A special property of SVM is , SVM simultaneously minimizes the empirical classification error and maximizes the geometric margin. So SVM is called Maximum Margin Classifiers. SVM is based on Structural risk Minimization (SRM). SVM maps input vectors to a higher dimensional space where a maximal separating hyperplane is constructed. Two parallel hyperplanes are constructed on each side of the hyperplane that separate the data. The separating hyperplane is the hyperplane that maximizes the distance between the two parallel hyperplanes. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalization error of the classifier will be.

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in *text classification* that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simplest and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

The naive Bayes classifier greatly simplifies learning by assuming that features are independent given class. Although independence is generally a poor assumption, in practice naive Bayes often competes well with more sophisticated classifiers.

Types of Naïve Bayes Model:

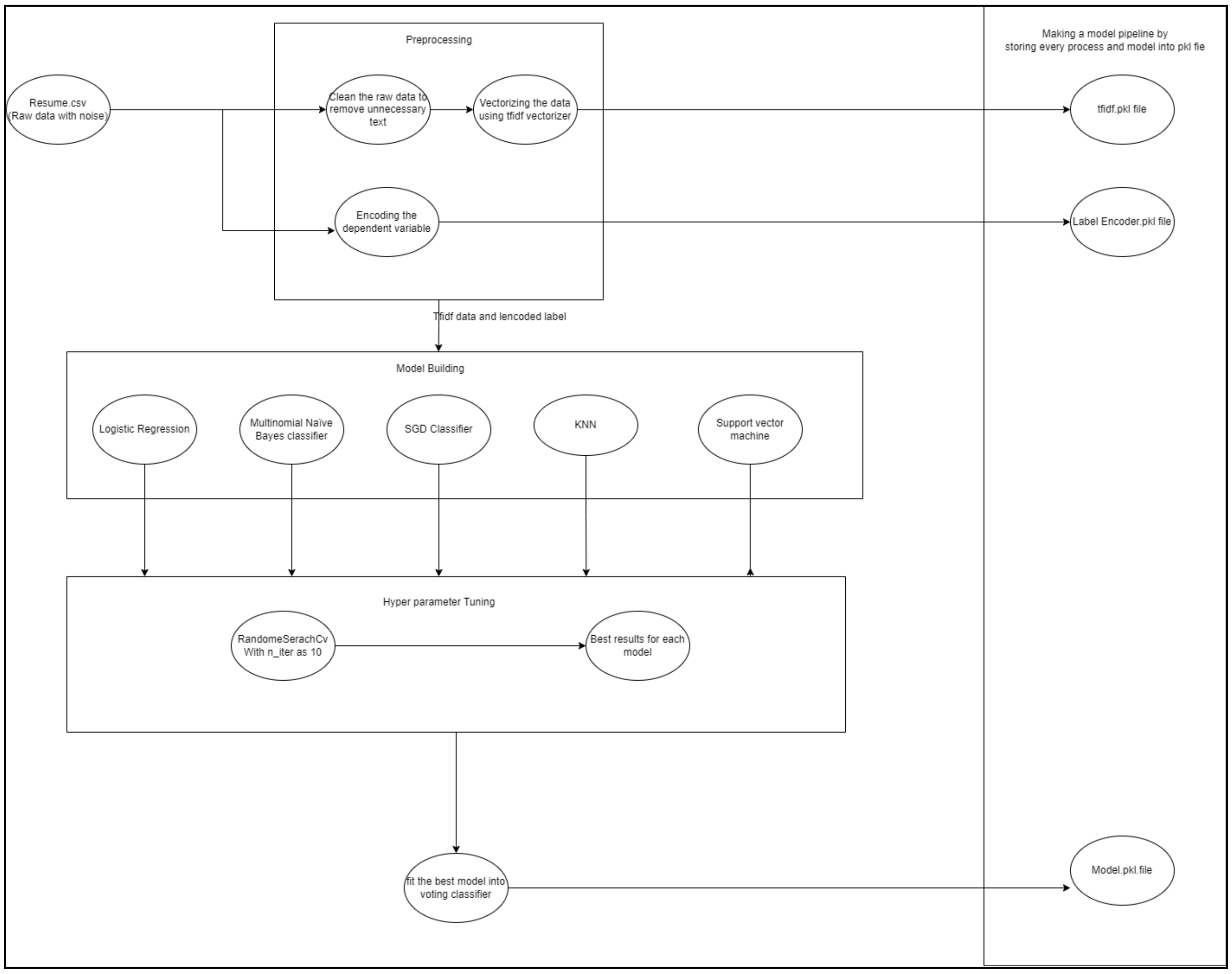
- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.
  The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is

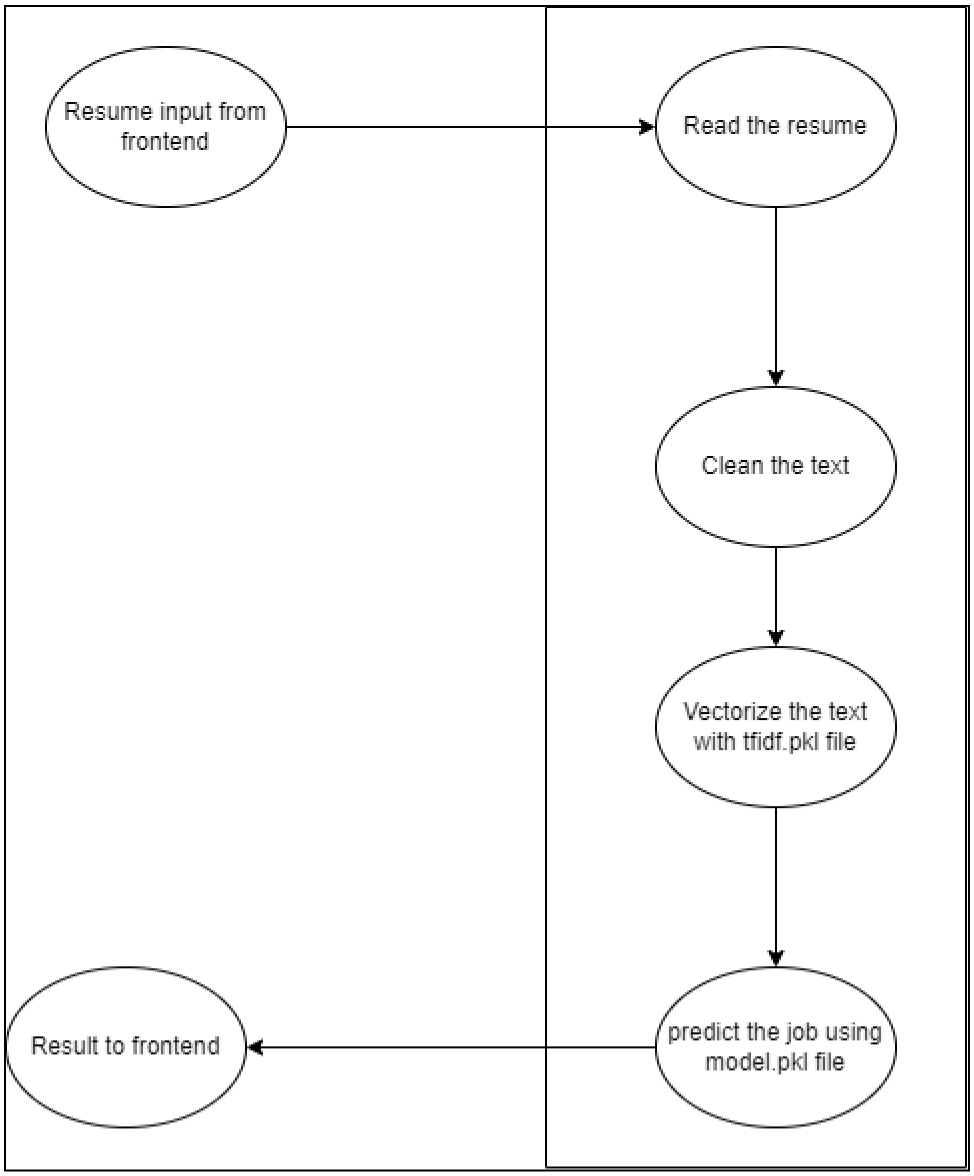
present or not in a document. This model is also famous for document classification tasks.

SYSTEM REQUIREMENT SPECIFICATIONS

1. PC/Laptop
2. Minimum 4GB RAM
3. Minimum 4 core CPU
4. Python v3.7
5. Flask v2.0
6. Any text editor
7. Any Web Browser

SYSTEM DESIGN / FLOW DIAGRAM


Preprocessing
Making a model pipeline by storing every process and model into pkl fie
Resume.csv (Raw data with noise)
Clean the raw data to remove unnecessary text
Vectorizing the data using tfidf vectorizer
tfidf.pkl file
Encoding the dependent variable
Label Encoder.pkl file
Tfidf data and lencoded label
Model Building
Logistic Regression
Multinomial Naïve Bayes classifier
SGD Classifier
KNN
Support vector machine
Hyper parameter Tuning
RandomeSerachCv With n_iter as 10
Best results for each model
fit the best model into voting classifier
Model.pkl.file


Resume input from frontend
Read the resume
Clean the text
Vectorize the text with tfidf.pkl file
predict the job using model.pkl file
Result to frontend

IMPLEMENTATION

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt


df=pd.read_csv('/content/Resume.csv')


df


df['Resume']


df.isna().sum()


k=df['Category'].value_counts()


k.index


values=k.index
```

```
counts=k.values


counts


fig, ax = plt.subplots(figsize =(16, 9))

ax.barh(values, counts)

for s in ['top', 'bottom', 'left', 'right']:

    ax.spines[s].set_visible(False)

ax.xaxis.set_ticks_position('none')

ax.yaxis.set_ticks_position('none')

ax.xaxis.set_tick_params(pad = 5)

ax.yaxis.set_tick_params(pad = 10)

ax.invert_yaxis()

for i in ax.patches:

    plt.text(i.get_width()+0.2, i.get_y()+0.5,

             str(round((i.get_width()), 2)),

             fontsize = 10, fontweight ='bold',

             color ='grey')

fig.text(0.9, 0.15, 'Distribution', fontsize = 12,

         color ='grey', ha ='right', va ='bottom',
         alpha = 0.7)

plt.show()


import nltk


from nltk.corpus import stopwords


df['cleaned_text']=''


df


import re


def clean(resume):

    resume=re.sub('httpS+s*',' ',resume)

    resume=re.sub('RT|cc',' ',resume)

    resume=re.sub('#S+',' ',resume)

    resume=re.sub('@S+','  ',resume)

    resume=re.sub('[%s]' %
re.escape("""!"#$%&'()*+,-./:;<=>?@[]^_`{|}~"""), ' ', resume)

    resume=re.sub(r'[^x00-x7f]',r' ', resume)
    resume=re.sub('s+',' ',resume)

    resume=resume.lower()

    return resume


df['cleaned_text']=df['Resume'].apply(lambda x: clean(x))


df


len(df)


df.shape


corpus=" "

for i in range(0,962):

    corpus= corpus+ df["cleaned_text"][i]


corpus[1000:2500]


from sklearn.preprocessing import LabelEncoder


le=LabelEncoder()
```

```
y=df['Category']

y=le.fit_transform(y)

y

X=df['cleaned_text']

from sklearn.feature_extraction.text import
TfidfVectorizer,CountVectorizer

#vect=CountVectorizer(encoding='utf-8',lowercase=False)

tf=TfidfVectorizer(max_features=2000,sublinear_tf=True,norm='l1',stop_word
s='english',lowercase=False)

tf.fit(X)

X_trans=tf.transform(X)

from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.naive_bayes import MultinomialNB

from sklearn.multiclass import OneVsRestClassifier



from sklearn.linear_model import LogisticRegression

lr=OneVsRestClassifier(LogisticRegression())



from sklearn.linear_model import SGDClassifier

sgd=SGDClassifier()



from sklearn.neighbors import KNeighborsClassifier



from sklearn.metrics import f1_score,accuracy_score,classification_report



m=OneVsRestClassifier(MultinomialNB())



s=StratifiedShuffleSplit(n_splits=5,test_size=0.3,random_state=42)



f1=[]

accuracy=[]

clf=OneVsRestClassifier(KNeighborsClassifier())

for t_index,te_index in s.split(X_trans,y):
    X_train,y_train=X_trans[t_index],y[t_index]

    X_test,y_test=X_trans[te_index],y[te_index]

    sgd.fit(X_train,y_train)

    f1.append(f1_score(y_test,sgd.predict(X_test),average='micro'))

    accuracy.append(accuracy_score(y_test,sgd.predict(X_test)))



f1



accuracy



from sklearn.model_selection import train_test_split



X_t,X_te,y_t,y_te=train_test_split(X_trans,y,test_size=0.2,random_state=42
)



m.fit(X_t,y_t)



y_pred=m.predict(X_te)



y_pred
```

```
print(classification_report(y_te,y_pred))


from sklearn.multiclass import OneVsRestClassifier

from sklearn.neighbors import KNeighborsClassifier


clf = OneVsRestClassifier(KNeighborsClassifier())

clf.fit(X_t, y_t)

prediction = clf.predict(X_te)


print(accuracy_score(y_te,prediction))


lr.fit(X_t,y_t)

print(classification_report(y_te,lr.predict(X_te)))


from sklearn.svm import SVC


from sklearn.model_selection import RandomizedSearchCV


di={

    'C':[1.0,0.9,0.8,1.2,1.5,2,0.5],

    'kernel':['rbf','linear']
```

```
}

search=RandomizedSearchCV(SVC(),di,n_iter=10,cv=5,random_state=2)

k=search.fit(X_train,y_train)

k.cv_results_

k.best_params_

sv=SVC(C=2,kernel='rbf')

sv.fit(X_train,y_train)

f1_score(y_test,sv.predict(X_test),average='micro')

sv.predict(X_test)

from sklearn.ensemble import VotingClassifier

vi=VotingClassifier(estimators=[
    ('LogisticRegression',lr),

    ('SGD classifier',sgd),

    ('OnevsRestKnn',clf),

    ('SVM_rbf_kernel',sv)

],voting='hard')



f1=[]

accuracy=[]

clf=OneVsRestClassifier(KNeighborsClassifier())

for t_index,te_index in s.split(X_trans,y):

    X_train,y_train=X_trans[t_index],y[t_index]

    X_test,y_test=X_trans[te_index],y[te_index]

    vi.fit(X_train,y_train)

    f1.append(f1_score(y_test,vi.predict(X_test),average='micro'))

    accuracy.append(accuracy_score(y_test,vi.predict(X_test)))



f1



accuracy



pip install PyPDF2
```

```
btext='''Skills * Programming Languages: Python (pandas, numpy, scipy,
scikit-learn, matplotlib), Sql, Java, JavaScript/JQuery. * Machine
learning: Regression, SVM, NaÃ¯ve Bayes, KNN, Random Forest, Decision
Trees, Boosting techniques, Cluster Analysis, Word Embedding, Sentiment
Analysis, Natural Language processing, Dimensionality reduction, Topic
Modelling (LDA, NMF), PCA & Neural Nets. * Database Visualizations: Mysql,
SqlServer, Cassandra, Hbase, ElasticSearch D3.js, DC.js, Plotly, kibana,
matplotlib, ggplot, Tableau. * Others: Regular Expression, HTML, CSS,
Angular 6, Logstash, Kafka, Python Flask, Git, Docker, computer vision -
Open CV and understanding of Deep learning.Education Details



Data Science Assurance Associate



Data Science Assurance Associate - Ernst & Young LLP

Skill Details

JAVASCRIPT- Exprience - 24 months

jQuery- Exprience - 24 months

Python- Exprience - 24 monthsCompany Details

company - Ernst & Young LLP

description - Fraud Investigations and Dispute Services   Assurance

TECHNOLOGY ASSISTED REVIEW

TAR (Technology Assisted Review) assists in accelerating the review
process and run analytics and generate reports.

* Core member of a team helped in developing automated review platform
tool from scratch for assisting E discovery domain, this tool implements
predictive coding and topic modelling by automating reviews, resulting in
reduced labor costs and time spent during the lawyers review.
```

* Understand the end to end flow of the solution, doing research and development for classification models, predictive analysis and mining of the information present in text data. Worked on analyzing the outputs and precision monitoring for the entire tool.

* TAR assists in predictive coding, topic modelling from the evidence by following EY standards. Developed the classifier models in order to identify "red flags" and fraud-related issues.

Tools & Technologies: Python, scikit-learn, tfidf, word2vec, doc2vec, cosine similarity, NaÃ¯ve Bayes, LDA, NMF for topic modelling, Vader and text blob for sentiment analysis. Matplot lib, Tableau dashboard for reporting.

MULTIPLE DATA SCIENCE AND ANALYTIC PROJECTS (USA CLIENTS)

TEXT ANALYTICS - MOTOR VEHICLE CUSTOMER REVIEW DATA * Received customer feedback survey data for past one year. Performed sentiment (Positive, Negative & Neutral) and time series analysis on customer comments across all 4 categories.

* Created heat map of terms by survey category based on frequency of words
* Extracted Positive and Negative words across all the Survey categories and plotted Word cloud.

* Created customized tableau dashboards for effective reporting and visualizations.

CHATBOT * Developed a user friendly chatbot for one of our Products which handle simple questions about hours of operation, reservation options and so on.

* This chat bot serves entire product related questions. Giving overview of tool via QA platform and also give recommendation responses so that user question to build chain of relevant answer.

* This too has intelligence to build the pipeline of questions as per user requirement and asks the relevant /recommended questions.

```
Tools & Technologies: Python, Natural language processing, NLTK, spacy,
topic modelling, Sentiment analysis, Word Embedding, scikit-learn,
JavaScript/JQuery, SqlServer




INFORMATION GOVERNANCE


Organizations to make informed decisions about all of the information they
store. The integrated Information Governance portfolio synthesizes
intelligence across unstructured data sources and facilitates action to
ensure organizations are best positioned to counter information risk.


* Scan data from multiple sources of formats and parse different file
formats, extract Meta data information, push results for indexing elastic
search and created customized, interactive dashboards using kibana.


* Preforming ROT Analysis on the data which give information of data which
helps identify content that is either Redundant, Outdated, or Trivial.


* Preforming full-text search analysis on elastic search with predefined
methods which can tag as (PII) personally identifiable information (social
security numbers, addresses, names, etc.) which frequently targeted during
cyber-attacks.


Tools & Technologies: Python, Flask, Elastic Search, Kibana




FRAUD ANALYTIC PLATFORM


Fraud Analytics and investigative platform to review all red flag cases.


â ¢ FAP is a Fraud Analytics and investigative platform with inbuilt case
manager and suite of Analytics for various ERP systems.


* It can be used by clients to interrogate their Accounting systems for
identifying the anomalies which can be indicators of fraud by running
advanced analytics


Tools & Technologies: HTML, JavaScript, SqlServer, JQuery, CSS, Bootstrap,
Node.js, D3.js, DC.js'''
```

```
pdfData=btext


cleaned_text=clean(pdfData)


cleaned_text=[cleaned_text]


idf=tf.transform(cleaned_text)


idf


from sklearn.metrics.pairwise import cosine_similarity


def get_index_from_title(title):
    return df[df.Category == title]["index"].values[0]


similarity=[]
for i in range(962):
    c=cosine_similarity(X_trans[i],idf)
    similarity.append((c,y[i]))


similarity
```

```
i=vi.predict(idf)


le.inverse_transform(i)


import pickle


with open('label_encoder.pkl','wb') as f:

  pickle.dump(y,f)


with open('tfid.pkl','wb') as ff:

  pickle.dump(tf,ff)


with open('votclas.pkl','wb') as fff:

  pickle.dump(vi,fff)
```

TEST RESULTS

- Four machine learning Algorithms have been used for classification task and are converted to voting classifier which is further used for deployment
- The algorithms which are used have already been mentioned in the above sections
- Hyper parameter tuning was done for each of the hyper parameters of the algorithm to obtain the best mode using RandomizedSearchCV
- It was found that Support vector classification model with regularization parameter 2 and kernel as rdf gives the best performance among the other classification algorithms(Logistic Regression,SGD classifier,KNN,Naive bayes classifier) that have been used
- The SGD classifier was found to have similar accuracy score as of SVC but when compared to f1 scores SVC was found to be the best
- The accuracy of all the models have been mentioned in the below table

Classification Algorithm	**Accuracy Score**
Logistic Regression	0.42
Naive bayes	0.45
SGD Classifier	0.986
Support Vector Machine(SVM)(C=2,kernel-’rbf’)	0.987
KNN(n_neighbours=5)	0.934

CONCLUSION

Successfully built a web application to predict the most suitable job based on a resume using Machine Learning Techniques. The simple, yet attractive website which is extremely user friendly gives the users a hassle-free experience and they can upload their resume and see which job they are most suited for within a few seconds.

We used various Machine Learning algorithms like Logistic Regression, Naive Bayes Classifier, SGD Classifier, SVM and KNN and we found out that the SVM Classifier gave the best accuracy, with an accuracy score of 0.987, and the SGD Classifier also gave almost the same performance, but SVM had the better F1 score.

REFERENCES

Research Papers referred:

1. Data Preprocessing for Supervised Learning
2. KNN Model-Based Approach in Classification
3. Data Classification Using Support Vector Machine
4. An Empirical Study of the Naïve Bayes Classifier

Websites referred:

1. https://geeksforgeeks.org
2. https://javatpoint.com
3. https://towardsdatascience.com/
4. https://medium.com/
5. https://www.analyticsvidhya.com