```
from google.colab import drive
from bs4 import BeautifulSoup
import pickle
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string
nltk.download('punkt')
nltk.download('stopwords')
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize.treebank import TreebankWordDetokenizer
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import PorterStemmer
import numpy as np
import pandas as pd
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
!pip install langdetect
!pip install translate
from translate import Translator
from langdetect import detect
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting langdetect
  Downloading langdetect-1.0.9.tar.gz (981 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 981.5/981.5 kB 13.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from langdetect) (1.16.0)
Building wheels for collected packages: langdetect
  Building wheel for langdetect (setup.py) ... done
  Created wheel for langdetect: filename=langdetect-1.0.9-py3-none-any.whl size=993243 sha256=1294af2f90462eb40fd46a7bd39ad28c144520353
  Stored in directory: /root/.cache/pip/wheels/d1/c1/d9/7e068de779d863bc8f8fc9467d85e25cfe47fa5051fff1a1bb
Successfully built langdetect
Installing collected packages: langdetect
Successfully installed langdetect-1.0.9
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting translate
  Downloading translate-3.6.1-py2.py3-none-any.whl (12 kB)
Collecting libretranslatepy==2.1.1
  Downloading libretranslatepy-2.1.1-py3-none-any.whl (3.2 kB)
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from translate) (4.9.2)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from translate) (8.1.3)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from translate) (2.27.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->translate) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->translate) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->translate) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->translate) (3.4)
Installing collected packages: libretranslatepy, translate
Successfully installed libretranslatepy-2.1.1 translate-3.6.1
```

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
train_data = pd.read_csv("/content/drive/My Drive/IR/Project/Job_Posting_dataset.csv")
test_data = pd.read_csv("/content/drive/My Drive/IR/Project/testing_dataset.csv")
test_data
```

	LinkedIn Resume	Job Profiles
0	Shyvee Shi\n(She/Her)\n3rd degree connection\n...	['Product Manager','Business Analyst','UX Desi...
1	Elisa Bellagamba\n\nAbout\nHigh-impact product...	['Product Manager','Business Analyst','Marketi...
2	Joni (Rafalski) Hoadley (She/Her)\n\nAbout\nI'...	['Product Manager','UX Designer','Business Ana...
3	Shane Connelly\n\nAbout\nI lead the product ma...	['Product Manager','Software Developer,'Machin...
4	Dana Tom\n(She/Her)\n\nAbout\nI'm a product ma...	['Product Manager','Marketing Analyst','Machin...

Removing all the rows that contain description in other languages than English

```
#@title Removing all the rows that contain description in other languages than English
for index, row in train_data.iterrows():
    try:
        lang = detect(row['Job_Description'])
        if lang != 'en':
            train_data.drop(index, inplace=True)
    except:
        # if an error occurs, assume the language is not English and drop the row
        train_data.drop(index, inplace=True)
train_data.reset_index(inplace=True)
train_data
```

	index	Job ID	Job Title	Company	Job_Description	Job Profile
0	1	3489403427	Software Engineer	LinkedIn	The ideal candidate will help build, maintain,...	Software Developer
1	2	3490979195	Software Engineer	PayPal	At PayPal (NASDAQ: PYPL), we believe that ever...	Software Developer
2	3	3507663809	Junior Software Developer (Web/Front-End)	Samsung Brasil	Position Summary\n\n\n\n\nDevelop a differenti...	Software Developer
3	4	3497871312	Software Engineer	Oracle	Want to come join the Oracle Health Data & Ana...	Software Developer
4	6	3497654432	Software Engineer	Illuma	We are looking for a highly motivated Software...	Software Developer
...	...	...	...	...	...	...
909	985	3496041048	Marketing Analyst	TI Fluid Systems	Description\n\nDescription:\n\nPosition Summar...	Marketing Analyst
910	986	3496037782	Marketing Analyst	TI Fluid Systems	Description\n\nDescription:\n\nPosition Summar...	Marketing Analyst

Removing all the rows that contain description in other languages than English

```
#@title Removing all the rows that contain description in other languages than English
for index, row in test_data.iterrows():
    try:
        lang = detect(row['LinkedIn Resume'])
        if lang != 'en':
            test_data.drop(index, inplace=True)
    except:
        # if an error occurs, assume the language is not English and drop the row
        test_data.drop(index, inplace=True)
test_data.reset_index(inplace=True)
test_data
```

	index	LinkedIn Resume	Job Profiles
0	0	Shyvee Shi\n(She/Her)\n3rd degree connection\n...	['Product Manager','Business Analyst','UX Desi...
1	1	Elisa Bellagamba\n\nAbout\nHigh-impact product...	['Product Manager','Business Analyst','Marketi...
2	2	Joni (Rafalski) Hoadley (She/Her)\n\nAbout\nI'...	['Product Manager','UX Designer','Business Ana...
3	3	Shane Connelly\n\nAbout\nI lead the product ma...	['Product Manager','Software Developer,'Machin...

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

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

# Define stop words
stop_words = set(stopwords.words('english'))

# Initialize lemmatizer
lemmatizer = WordNetLemmatizer()
def preprocess_text(text):
    text = text.lower()
    words = word_tokenize(text)
    words = [word for word in words if word not in stop_words]
    words = [lemmatizer.lemmatize(word) for word in words]
    text = ' '.join(words)
    return text

# Apply preprocess_text() function to job description column
train_data['Job_Description'] = train_data['Job_Description'].apply(preprocess_text)
test_data['LinkedIn Resume'] = test_data['LinkedIn Resume'].apply(preprocess_text)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
train_data.head()
```

	index	Job ID	Job Title	Company	Job_Description	Job Profile
0	1	3489403427	Software Engineer	LinkedIn	ideal candidate help build , maintain , troubl...	Software Developer
1	2	3490979195	Software Engineer	PayPal	paypal (nasdaq : pypl) , believe every perso...	Software Developer
2	3	3507663809	Junior Software Developer (Web/Front-End)	Samsung Brasil	position summary develop differentiated system...	Software Developer
					want come join oracle health data &	Software

```
def getmappings(le):
  d=dict(zip(le.classes_, le.transform(le.classes_)))
  revd=dict(zip(le.classes_, le.transform(le.classes_)))
  return d, revd
def encodetest(df,le):
  d, revd=getmappings(le)
  print(d)
  lst=df.at[0, 'Job Profiles']
  for i in range(len(df['Job Profiles'])):
    st=df.at[i, 'Job Profiles']
    st = st.replace("'","")
    st = st.replace("[","")
    st = st.replace("]","")
    lst = st.split(',')
    # lst=ast.literal_eval(lst)
    for j in range(len(lst)):
      # print(lst[2])
      lst[j]=int(d[lst[j]])
    df.at[i, 'Job Profiles']=lst
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train_data['Job Profile'] = le.fit_transform(train_data['Job Profile'])
encodetest(test_data,le)
```

```
{'Account Manager': 0, 'Business Analyst': 1, 'Machine Learning Engineer': 2, 'Marketing Analyst': 3, 'Product Manager': 4, 'Sales Analy
```

COUNT VECTORIZER (BASELINE)

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Instantiate a CountVectorizer object
vectorizer = CountVectorizer()

# Fit the vectorizer on the training data
vectorizer.fit(train_data['Job_Description'])

# Transform the job descriptions into numerical features using TF-IDF algorithm
X_train = vectorizer.transform(train_data['Job_Description'])
X_test = vectorizer.transform(test_data['LinkedIn Resume'])
y_train = train_data['Job Profile']


# Extract the class labels from the "job_profile" column
y_test = test_data["Job Profiles"].values
```

```
pd.DataFrame(X_train)
```

```
                                             0
  0    (0, 461)\t1\n (0, 830)\t1\n (0, 1021)\t1\n...
  1     (0, 140)\t2\n (0, 167)\t1\n (0, 168)\t1\n ...
  2     (0, 520)\t1\n (0, 539)\t1\n (0, 908)\t1\n ...
  3        (0, 1)\t2\n (0, 22)\t1\n (0, 23)\t1\n (0,...
  4     (0, 232)\t1\n (0, 462)\t1\n (0, 500)\t1\n ...
 ...                                             ...
 909    (0, 461)\t2\n (0, 535)\t3\n (0, 719)\t1\n ...
 910    (0, 461)\t2\n (0, 535)\t3\n (0, 719)\t1\n ...
 911    (0, 461)\t3\n (0, 523)\t1\n (0, 544)\t1\n ...
 912      (0, 1)\t1\n (0, 281)\t1\n (0, 461)\t2\n (...
 913    (0, 201)\t1\n (0, 461)\t2\n (0, 561)\t2\n ...
914 rows × 1 columns
```

TF-IDF VECTORIZER MODEL (INCLUDED IN MID-SEM PROJECT REVIEW)

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
# Instantiate a TfidfVectorizer object
vectorizer = TfidfVectorizer()

# Fit the vectorizer on the training data
vectorizer.fit(train_data['Job_Description'])

# Transform the job descriptions into numerical features using TF-IDF algorithm
X_train = vectorizer.transform(train_data['Job_Description'])
X_test = vectorizer.transform(test_data['LinkedIn Resume'])
y_train = train_data['Job Profile']


# Extract the class labels from the "job_profile" column
y_test = test_data["Job Profiles"].values
```

```
X_train
```

```
<914x12083 sparse matrix of type '<class 'numpy.float64'>'
        with 198655 stored elements in Compressed Sparse Row format>
```

```
'''import spacy
# Load the pre-trained GloVe model from spaCy
nlp = spacy.load('en_vectors_web_lg')

# Define a function to extract features from the job descriptions
def get_features(text):
    doc = nlp(text)
    features = []
    for token in doc:
        if token.has_vector:
            features.append(token.vector)
    if len(features) == 0:
        return [0] * 300  # 300 is the size of the GloVe vector
    else:
        return np.mean(features, axis=0)

# Extract the features from the job descriptions in the training data
X_train = [get_features(text) for text in train_data['Job_Description']]
X_train = pd.DataFrame(X_train)

# Extract the features from the job descriptions in the testing data
X_test = [get_features(text) for text in test_data['LinkedIn Resume']]
X_test = pd.DataFrame(X_test)

# Extract the labels from the dataset
y_train = train_data['Job Profile']
y_test = test_data['Job ProfileS']'''
```

```
X_train
```

PRE-TRAINED BERT MODEL (FINAL PROJECT REVIEW)

```
! pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: transformers in /usr/local/lib/python3.9/dist-packages (4.27.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-packages (from transformers) (1.22.4)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.9/dist-packages (from transformers) (0.13.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from transformers) (23.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.9/dist-packages (from transformers) (4.65.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.9/dist-packages (from transformers) (6.0)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from transformers) (2.27.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in /usr/local/lib/python3.9/dist-packages (from transformers) (0.13.3)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.9/dist-packages (from transformers) (2022.10.31)
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-packages (from transformers) (3.10.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.9/dist-packages (from huggingface-hub<1.0,>=0.11.0->
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (1.26.15)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (2.0.12
```

```
import torch
from transformers import DistilBertTokenizer, DistilBertModel
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Load the pre-trained DistilBERT model and tokenizer
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
model = DistilBertModel.from_pretrained('distilbert-base-uncased')
cnt = 0
# Define a function to extract features from the job descriptions
def get_features(text):
    # Truncate the input text if it's too long
    max_input_length = 512
    if len(text) > max_input_length:
        text = text[:max_input_length]
    input_ids = torch.tensor(tokenizer.encode(text, add_special_tokens=True)).unsqueeze(0)
    outputs = model(input_ids)
    last_hidden_states = outputs[0].squeeze(0)
    global cnt
    print(cnt)
    cnt = cnt+1
    return last_hidden_states.mean(dim=0).tolist()
# Extract the features from the job descriptions in the training data
X_train = [get_features(text) for text in train_data['Job_Description']]
X_train = pd.DataFrame(X_train)

# Extract the features from the job descriptions in the testing data
X_test = [get_features(text) for text in test_data['LinkedIn Resume']]
X_test = pd.DataFrame(X_test)

# Extract the labels from the dataset
y_train = train_data['Job Profile']
y_test = test_data['Job Profiles']
```

PRE-Trained GLOVE Model (FINAL PROJECT REVIEW)

```
!pip install gensim
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove.6B.zip -d glove
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gensim in /usr/local/lib/python3.9/dist-packages (4.3.1)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.9/dist-packages (from gensim) (6.3.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.9/dist-packages (from gensim) (1.22.4)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.9/dist-packages (from gensim) (1.10.1)
--2023-04-18 16:35:50--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2023-04-18 16:35:50--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2023-04-18 16:35:50--  https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  5.02MB/s    in 2m 39s

2023-04-18 16:38:30 (5.18 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]

Archive:  glove.6B.zip
  inflating: glove/glove.6B.50d.txt
  inflating: glove/glove.6B.100d.txt
  inflating: glove/glove.6B.200d.txt
  inflating: glove/glove.6B.300d.txt
```

```
import gensim.downloader as api

# Load the pre-trained GloVe embeddings
glove_model_100 = api.load("glove-wiki-gigaword-100")
glove_model = api.load("glove-wiki-gigaword-300")
```

```
[==================================================] 100.0% 376.1/376.1MB downloaded
```

```
D = 100
X_glove = np.zeros((len(train_data), D))
n = 0
for job_desc in train_data['Job_Description']:
  tokens = job_desc.split()
  vectors = []
  for word in tokens:
    try:
      vec = glove_model.get_vector(word)
      vectors.append(vec)
    except KeyError:
      #Keyerror occurs when word is not in the glove model
      pass
  if len(vectors) > 0:
    vectors = np.array(vectors)
    #taking mean vector as the embeddings for the one job description
    X_glove[n] = vectors.mean(axis=0)
  n += 1


D = 100
X_glove_test = np.zeros((len(test_data), D))
n = 0
for linkedin_resume in test_data['LinkedIn Resume']:
  tokens = linkedin_resume.split()
  vectors = []
  for word in tokens:
    try:
      vec = glove_model.get_vector(word)
      vectors.append(vec)
    except KeyError:
      #Keyerror occurs when word is not in the glove model
      pass
  if len(vectors) > 0:
    vectors = np.array(vectors)
    #taking mean vector as the embeddings for the one job description
    X_glove_test[n] = vectors.mean(axis=0)
  n += 1
```

```
X_glove
```

```
array([[-0.13502905,  0.10844683,  0.13811409, ..., -0.47229081,
         0.49913955,  0.11759271],
       [-0.05198411,  0.22824885,  0.24000978, ..., -0.39871737,
         0.50780141,  0.12462815],
       [-0.16504216,  0.15381232,  0.2522625 , ..., -0.48584154,
         0.56266677,  0.12977031],
       ...,
       [-0.03069113,  0.06611344,  0.10509066, ..., -0.40805864,
         0.59447598,  0.3063038 ],
       [-0.04434165,  0.1398156 ,  0.17068748, ..., -0.39596388,
         0.54596347,  0.1681754 ],
       [-0.12111861,  0.06542835,  0.11423052, ..., -0.41674417,
         0.57977182,  0.3347691 ]])
```

```
X_glove_embeddings = np.array(X_glove)
print(X_glove_embeddings.shape)

X_glove_test = np.array(X_glove_test)
print(X_glove_test.shape)
```

```
(914, 100)
(372, 100)
```

```
X_train = pd.DataFrame(X_glove_embeddings)
X_test = pd.DataFrame(X_glove_test)
y_train = train_data['Job Profile']
y_test = test_data['Job Profiles']
y_test
```

```
0      [4, 1, 7]
1      [4, 1, 3]
2      [4, 7, 1]
3      [4, 6, 2]
4      [4, 3, 2]
         ...
367    [5, 3, 1]
368    [5, 3, 1]
369    [5, 3, 1]
370    [5, 3, 1]
371    [5, 3, 1]
Name: Job Profiles, Length: 372, dtype: object
```

```
X_test
```

	0	1	2	3	4	5	6	7	8	9	...	
0	-0.076246	0.098259	0.310040	-0.329935	0.058622	-0.127922	0.053327	0.096167	-0.139239	-0.032951	...	0.02
1	-0.113037	0.084546	0.408734	-0.384656	0.027958	-0.150104	0.079483	0.074335	-0.135612	-0.033781	...	0.02
2	-0.077130	0.099903	0.260176	-0.261314	0.076752	-0.128020	0.080558	0.062915	-0.081423	-0.002541	...	0.01
3	-0.118886	0.116174	0.367731	-0.331074	0.057518	-0.118181	0.040713	0.100542	-0.107146	-0.015307	...	0.03
4	-0.104953	0.095309	0.373889	-0.420044	0.012531	-0.111475	0.055291	0.087240	-0.140327	-0.080362	...	0.02
...	...	...	...	...	...	...	...	...	...	...	...	
367	-0.063829	0.114406	0.377738	-0.308444	0.043279	-0.137845	0.015534	0.044441	-0.125098	-0.058371	...	0.06
368	-0.119136	0.080892	0.318145	-0.276158	0.054552	-0.208238	0.015754	0.113905	-0.125031	-0.005817	...	0.04
369	-0.037674	0.213186	0.418919	-0.445933	-0.025197	0.010179	0.103158	0.137608	-0.057110	-0.115010	...	-0.04
370	-0.091015	0.088470	0.385511	-0.440085	-0.000155	-0.132269	0.064407	0.108871	-0.114457	-0.107573	...	0.02
371	-0.005571	0.169480	0.101734	-0.094436	-0.032939	-0.225354	0.029388	-0.083177	-0.113624	0.090461		0.16

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import StackingClassifier
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
estimators = [('rf', RandomForestClassifier(random_state=42)), ('xgb', XGBClassifier(random_state=42))]
#clf = RandomForestClassifier(random_state=42)
clf = StackingClassifier(
     estimators=estimators, final_estimator=LogisticRegression()
)

# clf.fit(X_train, X_test).score(X_test, y_test)
clf.fit(X_train, y_train)

y_pred = clf.predict_proba(X_test)
def intersection(lst1, lst2):
    lst3 = [value for value in lst1 if value in lst2]
    return lst3

sorted_probs = (-y_pred).argsort()
k = 3
#getting top k classes with the highest probabilities
top_k = [sorted_probs[i, :k] for i in range(len(sorted_probs))]
print(y_pred[0])
print(sorted_probs[0])
print(top_k[0])
#Applying the formula for Precision@k
correct = [len(intersection(y_test[i][:k], top_k[i])) for i in range(len(y_test))]
precision = (np.sum(correct))/(len(y_test)*k)
print("Precision@k: " +str(precision))
```

```
[0.21240943 0.13281577 0.0522725  0.19215975 0.16643762 0.13831235
 0.07335203 0.03224056]
[0 3 4 5 1 6 2 7]
[0 3 4]
Precision@k: 0.6093189964157706
```

```
#K can be any value between 1 and 3 including both.
sorted_probs = (-y_pred).argsort()
k = 2
top_k = [sorted_probs[i, :k] for i in range(len(sorted_probs))]
# print(y_pred[0])
# print(sorted_probs[0])
# print(top_k[0])

#Applying the formula for Recall@k
correct = [len(intersection(y_test[i], top_k[i])) for i in range(len(y_test))]
precision = np.sum(correct) / (len(y_test) * k)
print("Recall@k: " +str(precision))
```

```
Recall@k: 0.6706989247311828
```

```
def instancePrecision(k,i):
  correct=0
  if(k<=3):
    correct=len(intersection(y_test[i][:k], sorted_probs[i][:k]))
  else:
    correct=len(intersection(y_test[i], sorted_probs[i][:k]))
  prec=correct/k
  return prec

#Finding Precision at all K<=3 at a particular instance.
precision3=[instancePrecision(3,i) for i in range(len(y_test))]
precision2=[instancePrecision(2,i) for i in range(len(y_test))]
precision1=[instancePrecision(1,i) for i in range(len(y_test))]
precisionAll=[[instancePrecision(1,i),instancePrecision(2,i),instancePrecision(3,i)] for i in range(len(y_test))]

#taking sum and mean.
meansum=0
for i in range(len(precision3)):
  sum=0
  for j in range(len(precisionAll[i])):
    sum+=precisionAll[i][j]
  mean=sum/len(precisionAll[i])
  meansum=meansum+mean

meanAvgprec=meansum/len(y_test)
print("Mean Average Precision: " +str(meanAvgprec))
```

```
    Mean Average Precision: 0.4611708482676221
```