PERSONALITY PREDICTION SYSTEM

NAME OF GROUP MEMBERS

ROSHIN JOHN (1800911)

SANDEEP KUMAR (1800913)

TARANDEEP SINGH (1800926)

TARANPREET KAUR (1800927)

NAME OF PROJECT GUIDE

ER. KHYATI MARWAH

CONTENT

01. INTRODUCTION

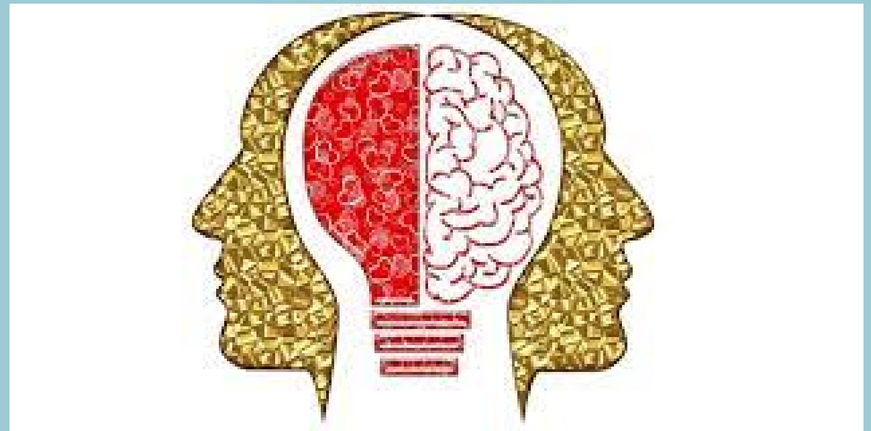
The project is based on identifying the personality of an individual using machine learning algorithms and big 5 models. The personality of a human plays a major role in his personal and professional life.

Nowadays, many organizations have also started shortlisting the candidates based on their personality as this increases the efficiency of the work because the person is working on what he is good at than what he is forced to do.

The basic aim of our project is to build a more reliable and smarter prediction system.

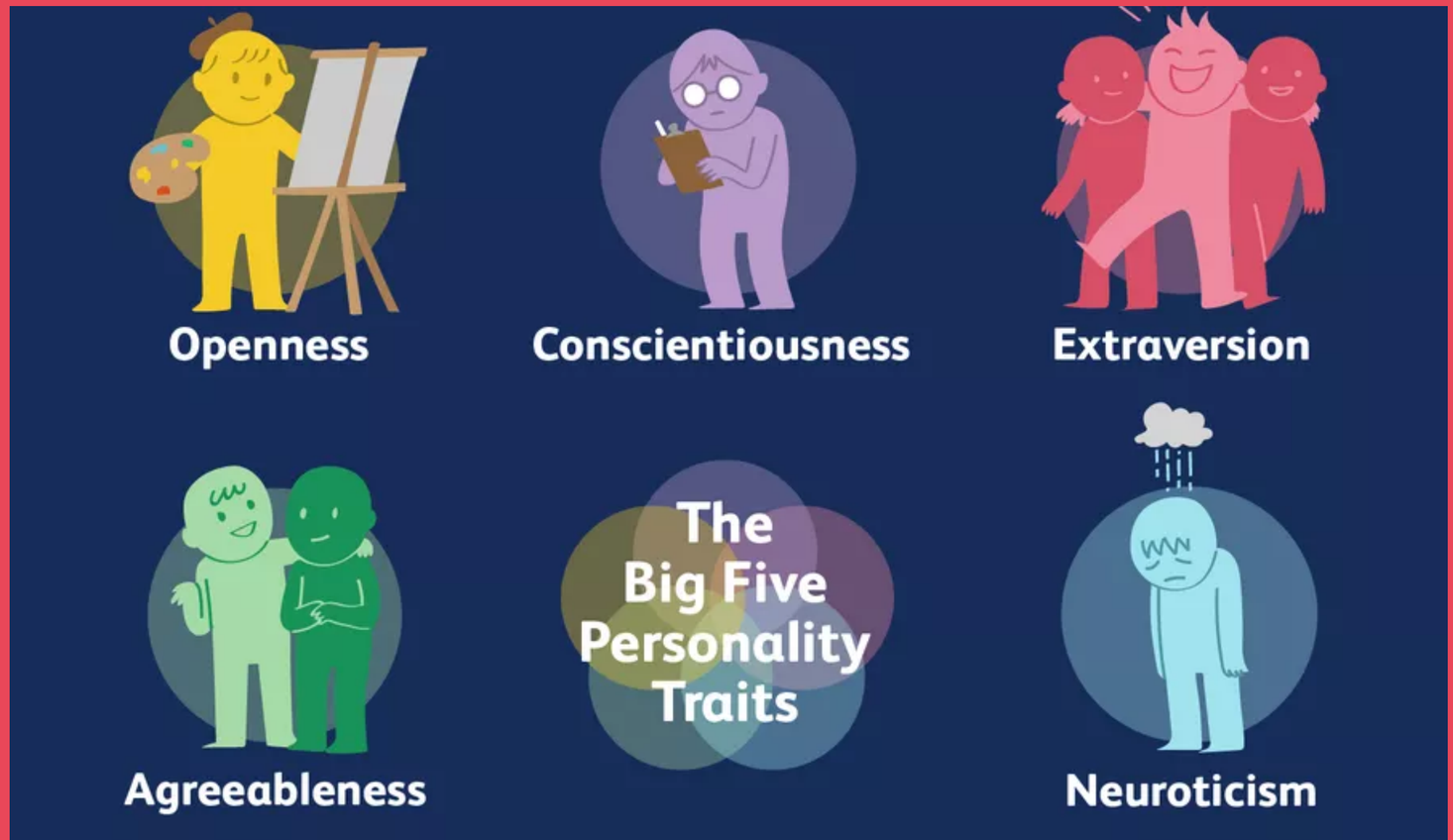
The idea is to develop a system that fetches data from the social media account and predicts the user's personality.

All these processes are carried out with the help of various machine learning algorithms.



THE BIG FIVE
PERSONALITY TRAITS

The Big Five model resulted from the contributions of many independent researchers. Gordon Allport and Henry Odbert first formed a list of 4,500 terms relating to personality traits in 1936 (Vinney, 2018).



PERSONALITY TRAITS

OPEN TO EXPERIENCE:

It involves various dimensions, like imagination, sensitivity, attentiveness, preference to variety, and curiosity.

AGREEABLENESS:

It is a quality that analyses the individual behavior based on the generosity, sympathy, cooperativeness and ability to adjust with people

CONSCIENTIOUSNESS:

This trait is used to describe the carefulness and diligence of the person. It is the quality that describes how organized and efficient a person is.

NEUROTICISM

This trait usually describes a person to have mood swings and has extreme expressive power.

EXTRAVERSION:

It is the trait that describes how the best candidates can interact with people that is how good are his/her social skills.

WHAT IS PREDICTION & PREDICTIVE ANALYSIS?

“Prediction” refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome,

Predictive analytics describes the use of statistics and modeling to determine future performance based on current and historical data. Predictive analytics looks at patterns in data to determine if those patterns are likely to emerge again, which allows businesses and investors to adjust where they use their resources to take advantage of possible future events.

02. LITERATURE SURVEY

Allport’s Trait model:

01

Gordon Allport set out one of the earliest personality models, which groups personality traits into three categories, cardinal traits, which shape the person, his/her attitudes, and his/her behaviors; central traits, which are the factors that determine most of individual behavior; and secondary traits, which may only be revealed in certain situations

Cattell’s 16 personality factor model:

02

Cattell’s [13] model includes 16 essential personality factors, which are listed under five major categories. This model had a profound influence on the development of the later Big Five model.

Eysenck’s Giant Three model:

03

This model was originally known as the PEN model before psychoticism was added to his original two traits of extraversion and neuroticism to form the Giant Three

The Myers-Briggs Type Indicator (MBTI) model:

04

This model [4] covers four personality dimensions:

(1) Extroversion (E) vs Introversion (I);

(2) Sensing (S) vs Intuition (N);

(3) Thinking (T) vs. Feeling (F); and

(4) Judging (J) vs. Perceiving (P)

03. PROBLEM STATEMENT

PROBLEM STATEMENT

- Often co-workers face problems due to personality differences.
- Until now, many jobs needed psychometric tests.
- Moreover, personality is relevant to several interactions.
- Personality assessment helps us to get a broader view of predicting job satisfaction and lifestyle.

04. PROPOSED SOLUTION

PROPOSED SOLUTION

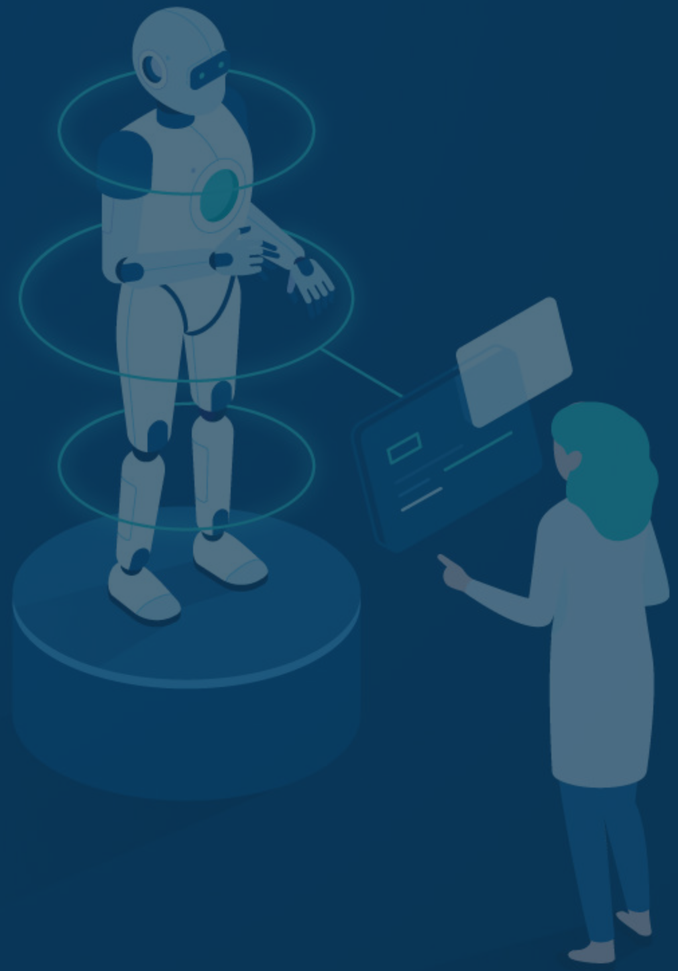
- We predict the personality on the basis of the data collected from various social media platforms of the user which may be very beneficial in many aspects.
- Fetch data from the various social media accounts.
- Each post is treated as raw data and applies a learning model to predict the personality.

04. TECHNOLOGY USED

- PROGRAMMING LANGUAGE
  - PYTHON 3
  - JUPYTER NOTEBOOK
- TECHNOLOGY
  - MACHINE LEARNING
  - SUPERVISED LEARNING

MACHINE LEARNING

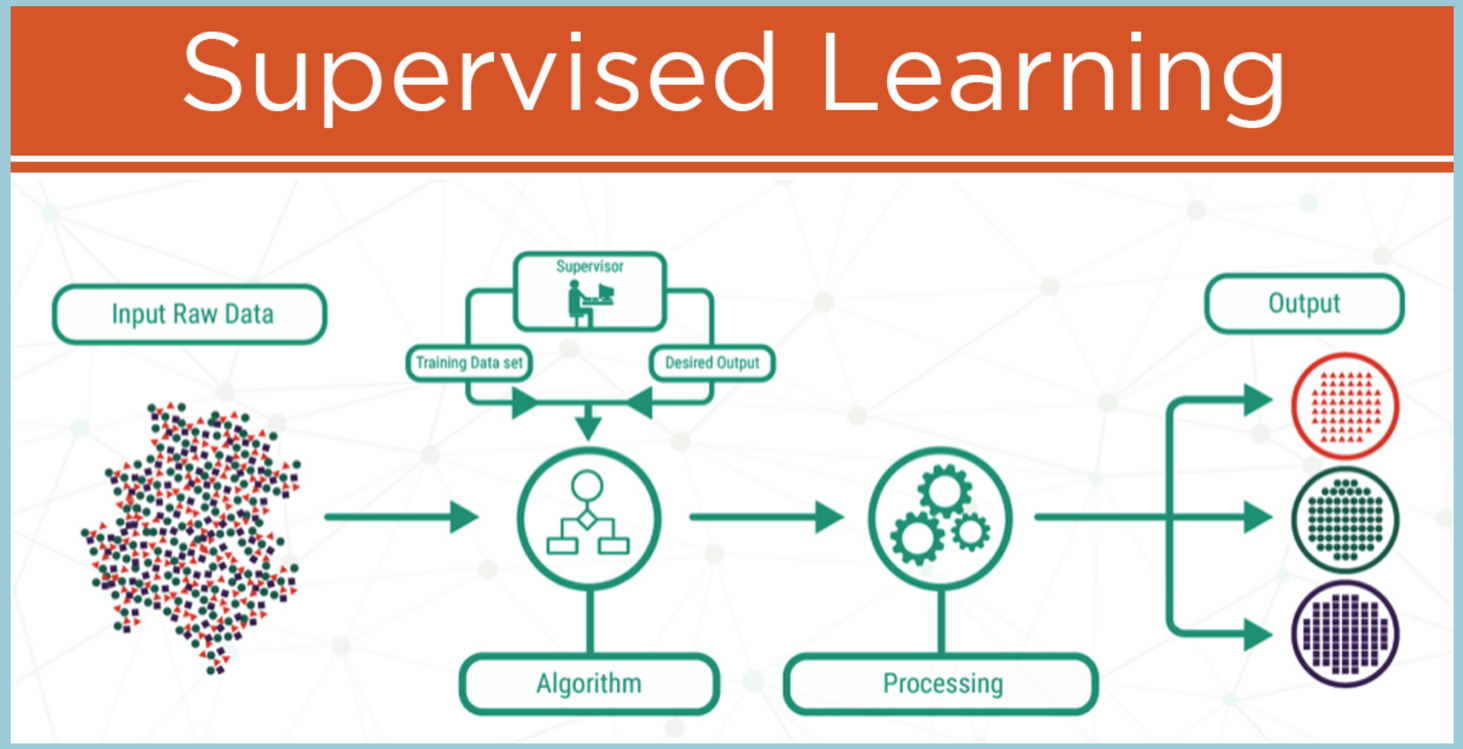
Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.



SUPERVISED LEARNING

Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with the correct answer.

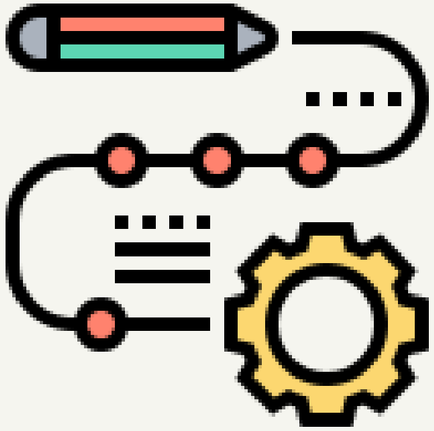
After that, the machine is provided with a new set of examples(data) so that supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data.



06. DESIGN METHODOLOGY

METHODOLOGY

DATA COLLECTION



PREDICTIVE MODEL

DATA PREPROCEESING

DATA TRAINING

DATA ANALYSIS

STEPS INVOLVED IN WORKING OF THE MODEL

- DATA GATHERING
- IMPORT LIBRARIES
- IMPORT DATASET
- EXPLORATORY DATA ANALYSIS
- PRE-PROCESSING OF DATASET
- FEATURE ENGINEERING
- TRAINING & EVALUATING 60-40 SPLIT
- ACCURACY & COMPARISON OF ALGORITHMS
- TRAINING & EVALUATING 70- 30 SPLIT
- FOUR CLASSIFIERS ACROSS MBTI AXIS

CONTINUED....

STEPS INVOLVED IN WORKING OF THE MODEL

- FEATURES CORRELATION ANALYSIS
- LEMMATIZATION IN PRE-PROCESSING
- FEATURE ENGINEERING INCLUDING Tf-IDF
- MODEL TESTING

07. DEMONSTRATION WITH THE SCREENSHOTS

jupyter PROJECT- PERSONAILTY PREDICTION SYSTEM Last Checkpoint: 14 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

PROJECT NAME- "PERSONALITY PREDCITION SYSTEM" ¶

DEPARTMENT- B.Tech

COURSE- COMPUTER SCIENCE

SEMESTER- 6TH

PROBLEM STATEMENT-

- Personality is a bleed of indiviual's actions, feelings, inspirations and thoughts.
- Personality has great impact on person's life as it also affects ones' choices in life.
- This project proposes an automatic personality prediction appraoch of user using socila media data using Machine Learning
- So, this model will extract peronality traits of persons's based on Big Five Factor Theory and MBTI traits.

MOTIVATION-

- Previously person has to go explicit for the pyhscometric tests and questionnaire.
- So there is a need to automate this process which will consider multiple parameters to predict personality of a person that can be used in
  and academic.

INTODUCTION-

The attributes which charcaterize a peron such as emotions, beahviour, mind and temperament defin

jupyter PROJECT- PERSONAILTY PREDICTION SYSTEM Last Checkpoint: 14 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Markdown

BIG FIVE FRAMEWORK

Trait	Description
Openness	Curious, original, intellectual, creative, and open to new ideas.
Conscientiousness	Organized, systematic, punctual, achievement oriented, and dependable.
Extraversion	Outgoing, talkative, sociable, and enjoys being in social situations.
Agreeableness	Affable, tolerant, sensitive, trusting, kind, and warm.
Neuroticism	Anxious, irritable, temperamental, and moody.

01 GATHERING OF DATASET

- COLLECT THE DATA FORM VARIOUS SOCIAL MEDIA PLATFORMS

- COMBINE ALL THE RAW DATA INTO AN EXCEL FORMAT WITH AN EXTENSION OF. CSV FILE

- BASICALLY, OUR DATASET HAS BEEN TAKEN FROM - https://www.kaggle.com/datasnaek/mbti-type

02 IMPORT LIBRARIES

Step 1- IMPORT LIBRARIES

In [6]:

```
# Data Analysis
import pandas as pd
import numpy as np
from numpy import asarray
from numpy import savetxt
from numpy import loadtxt
import pickle as pkl
from scipy import sparse

# Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt
import wordcloud
from wordcloud import WordCloud, STOPWORDS

# Text Processing
import re
import itertools
import string
import collections
from collections import Counter
from sklearn.preprocessing import LabelEncoder
import nltk
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import stopwords
from nltk.corpus import stopwords
from nltk import word_tokenize
from nltk import word_tokenize
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer

# Machine Learning packages
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import sklearn.cluster as cluster
```

```
import nltk
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import stopwords
from nltk.corpus import stopwords
from nltk import word_tokenize
from nltk import word_tokenize
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer

# Machine Learning packages
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import sklearn.cluster as cluster
from sklearn.manifold import TSNE

# Model training and evaluation
from sklearn.model_selection import train_test_split

#Models
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
from xgboost import plot_importance

#Metrics
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, multilabel_confusion_matrix, confusion_matrix
from sklearn.metrics import classification_report

# Ignore noise warning
import warnings
warnings.filterwarnings("ignore")
```

03 IMPORT DATASET

Step 2- IMPORT THE DATASET

- This dataset from Kaggle comes with two columns: the Myers-Briggs type of a user and 50 user posts stored as strings.

In [8]:
```
#Loading dataset
data_set = pd.read_csv("mbti_1.csv")

print(data_set.head(10))
print("*"*40)
print(data_set.info())
```

```
   type                                              posts
0  INFJ  'http://www.youtube.com/watch?v=qsXHcwe3krw|||...
1  ENTP  'I'm finding the lack of me in these posts ver...
2  INTP  'Good one  _____   https://www.youtube.com/wat...
3  INTJ  'Dear INTP,   I enjoyed our conversation the o...
4  ENTJ  'You're fired.|||That's another silly misconce...
5  INTJ  '18/37 @.@|||Science  is not perfect. No scien...
6  INFJ  'No, I can't draw on my own nails (haha). Thos...
7  INTJ  'I tend to build up a collection of things on ...
8  INFJ  I'm not sure, that's a good question. The dist...
9  INTP  'https://www.youtube.com/watch?v=w8-egj0y8Qs||...
****************************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8675 entries, 0 to 8674
Data columns (total 2 columns):
type     8675 non-null object
posts    8675 non-null object
dtypes: object(2)
memory usage: 135.7+ KB
None
```

04 EXPLORATORY DATA ANALYSIS

4.1 DATA VISUALISATION

(i)Data visualization for no. of posts for each personality type

(ii)SWARM PLOT

(iii)DISTANCE PLOT:

4.2 WORDCLOUD

(i)Data visualization for no. of posts for each personality type

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

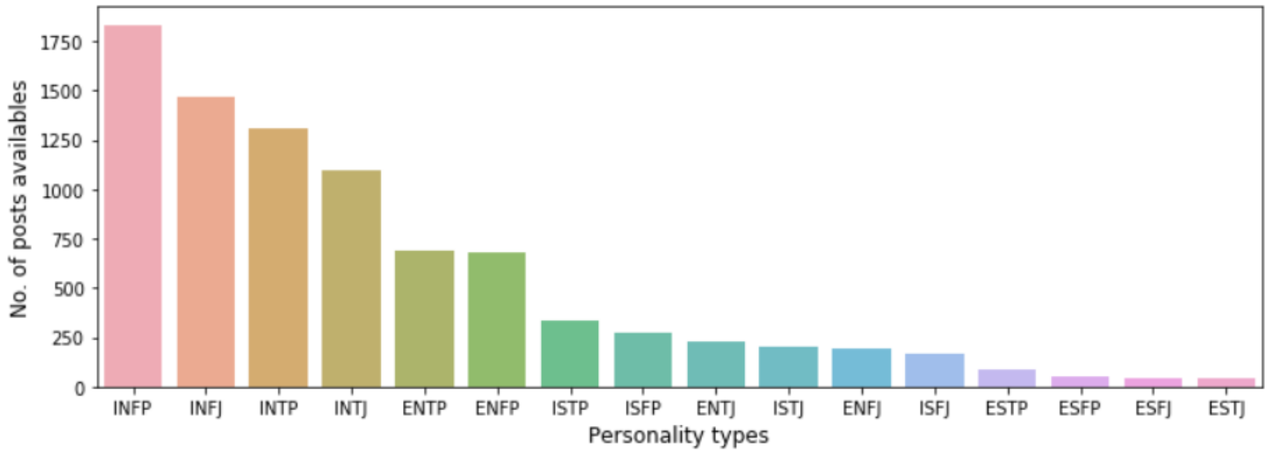
Run Markdown

(i)Data visualization for no. of posts for each personality type

The dataset is clearly unbalanced throughout the different classes. We observe that some of the personality types has a lot more data than others, the most common Kaggle users personality is INFP (Introvert Intuition Feeling Perceiving).

However, we reach this conclusion based on user comments: we can consider for now that users who comment on social media more frequently are more intoverted, perceptive, and emotional.

In [11]:
```
#Plotting this in descending order for better understanding of this visualization
cnt_srs = data_set['type'].value_counts()
plt.figure(figsize=(12,4))
sns.barplot(cnt_srs.index, cnt_srs.values, alpha=0.8)
plt.xlabel('Personality types', fontsize=12)
plt.ylabel('No. of posts availables', fontsize=12)
plt.show()
```



- Since the original dataset only came with 2 features, the Type and 50 posts for each person, we decided to create additional features for exploring & analysing our dataset.

(ii)SWARM PLOT

jupyter PROJECT- PERSONAILTY PREDICTION SYSTEM Last Checkpoint: 14 hours ago (autosaved)

Logout

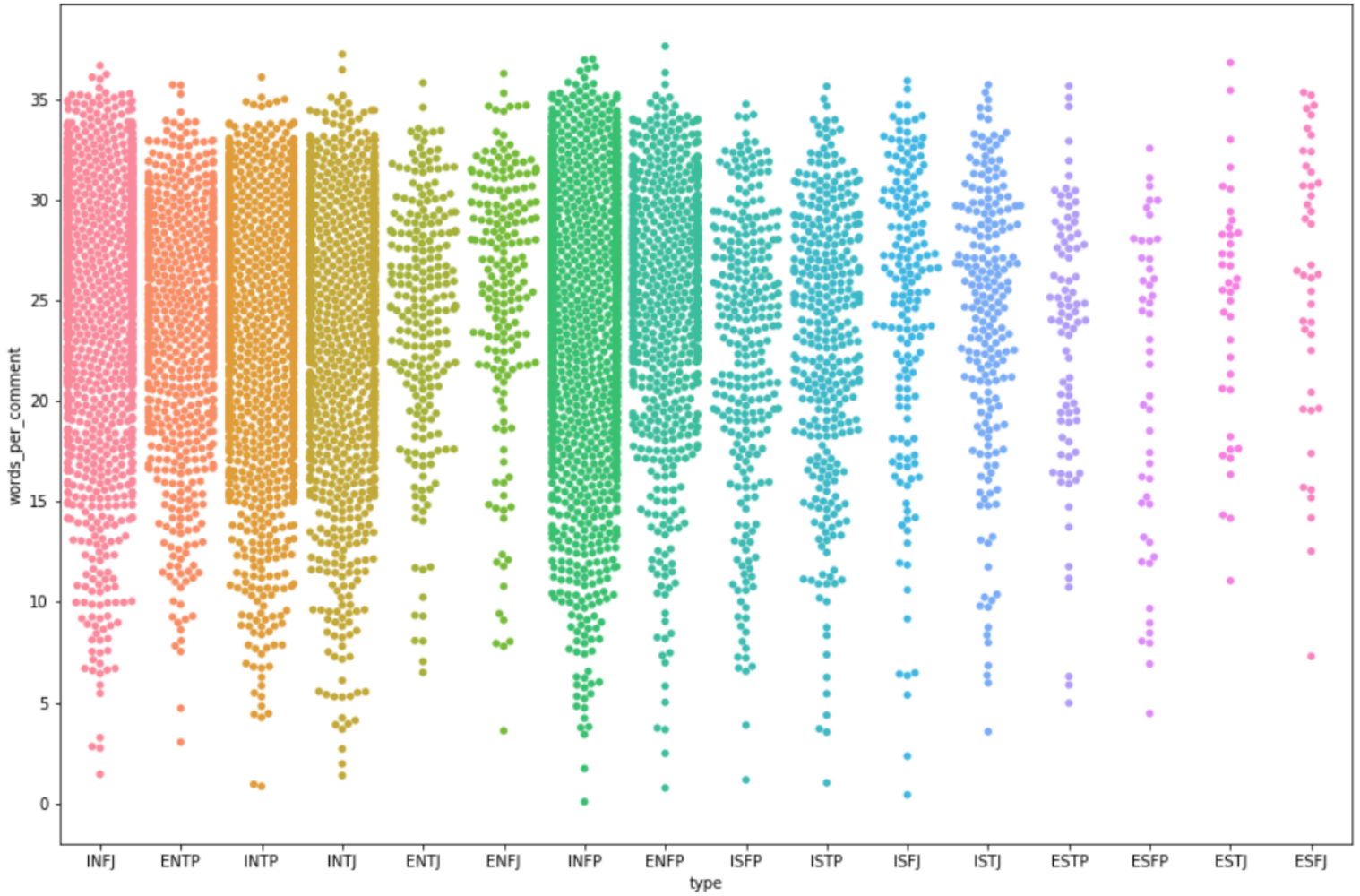
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

```
sns.swarmplot("type", "words_per_comment", data=df)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x21564415448>



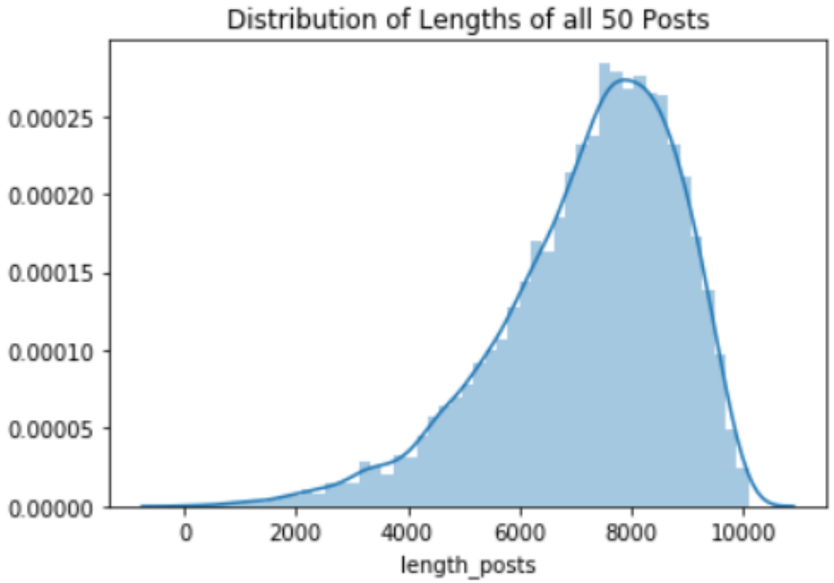
(iii)DISTANCE PLOT:

(iii)DISTANCE PLOT:

This seaborn visualization method shows the histogram distribution of data for a single column.

In [13]:
```
df["length_posts"] = df["posts"].apply(len)
sns.distplot(df["length_posts"]).set_title("Distribution of Lengths of all 50 Posts")
```

Out[13]:
```
Text(0.5, 1.0, 'Distribution of Lengths of all 50 Posts')
```



- The posts majorly contain general words like : I, I'm, so, me, or, if, and, can etc. It is safe to assume that these words won't really provide any useful information to train the ML model as most of them are stop-words, stem-words, or other useless words.
- Hence quite a lot pre-processing is required for individual user posts for each peronality type in the given MBTI dataset

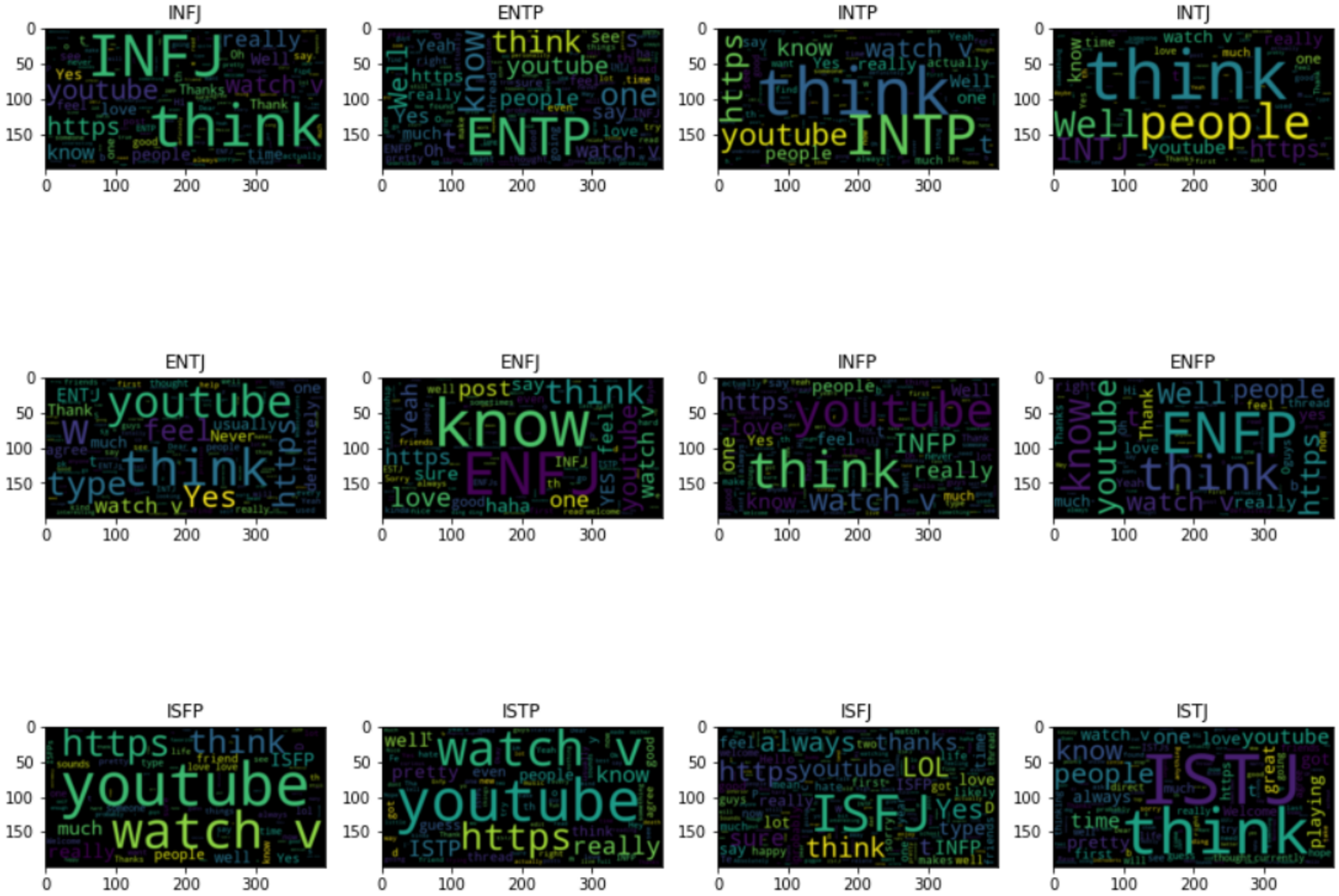
WORDCLOUD

```
# collocations to False  is set to ensure that the word cloud doesn't appear as if it contains any duplicate words
plt.figure(figsize=(25,10))
# generate word cloud, interpolation
plt.imshow(wc, interpolation='bilinear')
_ = plt.axis("off")
```

- Now we see the Wordclouds for each Personality Type. We produced 16 Word Clouds for 16 groups of personality. These word clouds are generated such that the size of each word is proportional to its appearance frequency in the top posts. We consider these word clouds to be illustrative of some of the unique ways that different MBTIs use language.

WORDCLOUD

```
ax[k].axis("off")
k+=1
```



05 PRE-PROCESSING STAGE

1. Remove links
2. Keep the End Of Sentence characters
3. Strip Punctation
4. Remove multiple full stops
5. Remove Non-words
6. Convert posts to lowercase
7. Remove multiple letters repeating words
8. Remove very short or long words
9. Remove MBTI Personality Words

```
#Remove multiple letter repeating words
    df["posts"] = df["posts"].apply(lambda x: re.sub(r'([a-z])\1{2,}[\s|\w]*','',x))

#Remove very short or long words
    df["posts"] = df["posts"].apply(lambda x: re.sub(r'(\b\w{0,3})?\b','',x))
    df["posts"] = df["posts"].apply(lambda x: re.sub(r'(\b\w{30,1000})?\b','',x))

#Remove MBTI Personality Words - crutial in order to get valid model accuracy estimation for
    if remove_special:
        pers_types = ['INFP' ,'INFJ', 'INTP', 'INTJ', 'ENTP', 'ENFP', 'ISTP' ,'ISFP' ,'ENTJ'
        pers_types = [p.lower() for p in pers_types]
        p = re.compile("(" + "|".join(pers_types) + ")")

    return df
#Preprocessing of entered Text
new_df = preprocess_text(df)
```

In [17]:
```
# Remove posts with less than X words
min_words = 15
print("Before : Number of posts", len(new_df))
new_df["no. of. words"] = new_df["posts"].apply(lambda x: len(re.findall(r'\w+', x)))
new_df = new_df[new_df["no. of. words"] >= min_words]

print("After : Number of posts", len(new_df))
```

```
Before : Number of posts 8675
After : Number of posts 8466
```

06 FEATURE ENGINEERING

6.1 Splitting into X and Y feature

(i)LabelEncoder

(ii) CountVectorizer

(i)LabelEncoder : ¶

Provided by Sklearn library that converts the the levels of categorical features (labels) into numeric form so as to convert it into the machine-readable form. It encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to a assigned earlier.

In [39]:
```
# Converting MBTI personality (or target or Y feature) into numerical form using Label Encoding
# encoding personality type
enc = LabelEncoder()
new_df['type of encoding'] = enc.fit_transform(new_df['type'])

target = new_df['type of encoding']
```

In [40]:
```
new_df.head(15)
```

Out[40]:

	type	posts	I-E	N-S	T-F	J-P	http_per_comment	music_per_comment	question_per_comment	img_per_comment	excl_per_comment	ellipsis_per_c
0	INFJ	enfp intj moments sportscenter plays...	0	0	1	0	0.48	0.02	0.36	0.12	0.06	
1	ENTP	finding lack these posts very alarming eo...	1	0	0	1	0.20	0.00	0.10	0.02	0.00	
2	INTP	good course which know thats bles...	0	0	0	1	0.10	0.00	0.24	0.00	0.08	
3	INTJ	dear intp enjoyed conversation other eos...	0	0	0	0	0.04	0.02	0.22			
4	ENTJ	youre fired eostokendot thats another silly...	1	0	0	0	0.12	0.02	0.20			

- We observe that almost all of these were were the most occuring words in our wordcloud above

(ii) CountVectorizer

- It is used to convert a collection of text documents to a vector of term/token counts and build a vocabulary of known words, b documents using that vocabulary. It also enables the pre-processing of text data prior to generating the vector representatio
- Here, we use stop_words='english' with CountVectorizer since this just counts the occurrences of each word in its vocabular words like 'the', 'and', etc. will become very important features while they add little meaning to the text. This is an important s our model can often be improved if you don't take those words into account.

42]:
```
# Vectorizing the posts for the model and filtering Stop-words
vect = CountVectorizer(stop_words='english')

# Converting posts (or training or X feature) into numerical form by count vectorization
train =  vect.fit_transform(new_df["posts"])
```

43]:
```
train.shape
```

43]:
```
(8466, 98555)
```

07 Training & Evaluating : 60-40 split

Algorithms being used in training the model:

(i) Random Forest

(ii) XG boost

(iii) Gradient Descent

(iv) Logistic Regression

(v) KNN Classifier

COMPARING ALGORITHMS

Step 6(b)-Comparing Algorithms

In [46]:
```
pd.DataFrame.from_dict(accuracies, orient='index', columns=['Accuracies(%)'])
```

Out[46]:

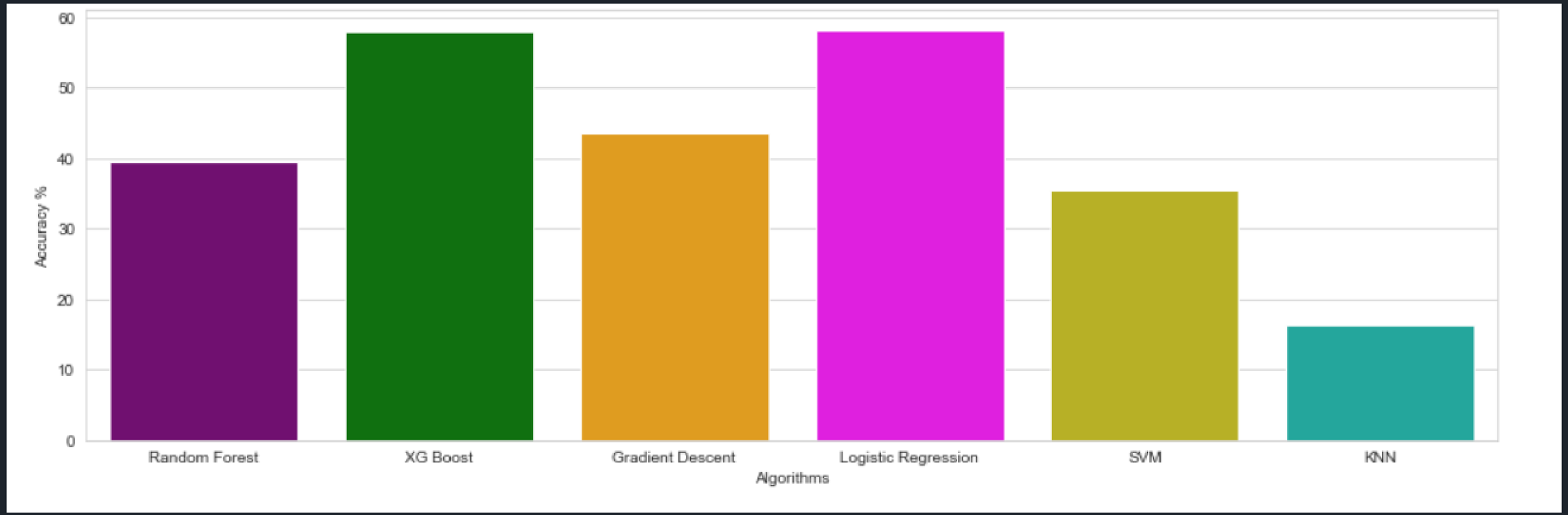
	Accuracies(%)
Random Forest	39.533510
XG Boost	57.868320
Gradient Descent	43.666962
Logistic Regression	58.193091
SVM	35.518158
KNN	16.445232

- We can clearly see that this model underfits out dataset when we apply split ratio of 60:40 on our dataset i.e. the model has not learned enough from the training data, resulting in low generalization and unreliable predictions. (almost all the results are near 50%, which is not good)

In [45]:
```
colors = ["purple", "green", "orange", "magenta","#CFC60E","#0FBBAE"]

sns.set_style("whitegrid")
plt.figure(figsize=(16,5))
plt.yticks(np.arange(0,100,10))
plt.ylabel("Accuracy %")
plt.xlabel("Algorithms")
sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()), palette=colors)
plt.show()
```

REPRESENTATION THROUGH GRAPHS



08 TRAINING & EVALUATING : 70-30 split

COMPARING ALGORITHMS

In [60]:
```
pd.DataFrame.from_dict(accuracies, orient='index', columns=['Accuracies(%)'])
```

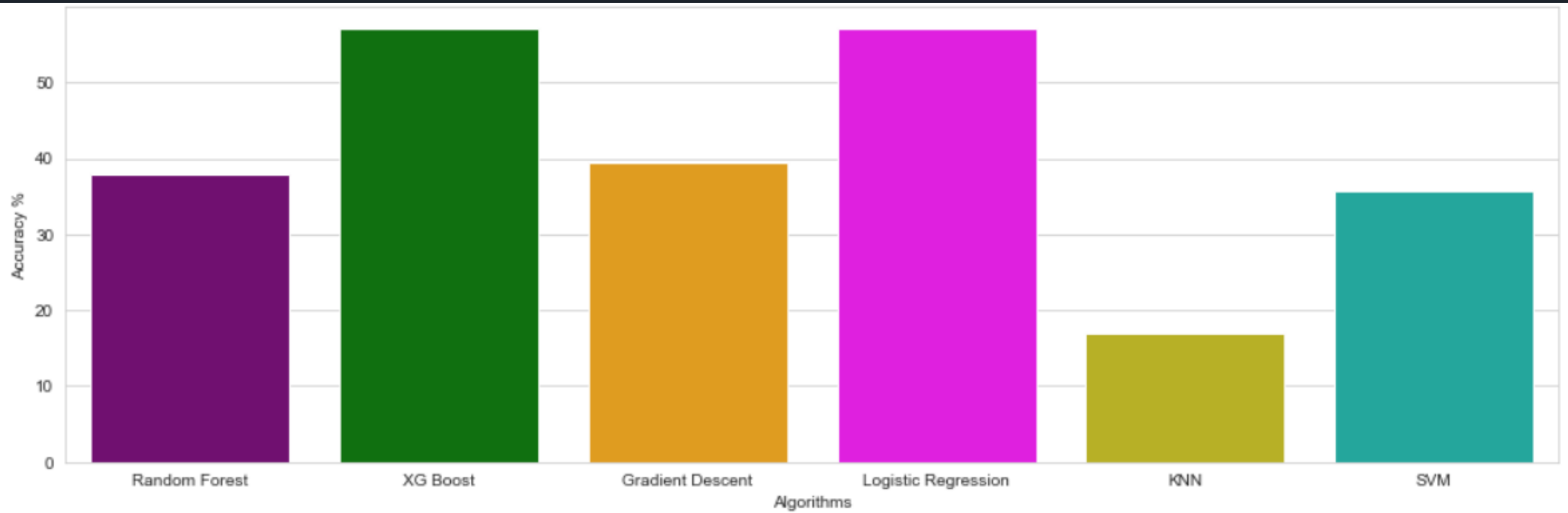
Out[60]:

	Accuracies(%)
Random Forest	37.866858
XG Boost	57.122405
Gradient Descent	39.513243
Logistic Regression	57.122405
KNN	16.964925
SVM	35.755190

In [62]:
```
colors = ["purple", "green", "orange", "magenta","#CFC60E","#0FBBAE"]

sns.set_style("whitegrid")
plt.figure(figsize=(16,5))
plt.yticks(np.arange(0,100,10))
plt.ylabel("Accuracy %")
plt.xlabel("Algorithms")
sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()), palette=colors)
plt.show()
```

REPRESENTATION THROUGH GRAPHS



- Inference : test_size=0.3 gives marginally better results for all algorithms
- As we can the above ML classifiers performs at efficieny of nearly 50% only - which is pretty bad. So, instead of selecting all 16 types of personalities as a unique feature, we hence train 4 classifiers individually to classify their personalities based on MBTI type.

09

FOUR CLASSIFIERS ACROSS MBTI AXIS

The Myers Briggs Type Indicator (or MBTI for short) is a personality type system that divides everyone into 16 distinct personality types across 4 axes:

Introversion (I) – Extroversion (E)
Intuition (N) – Sensing (S)
Thinking (T) – Feeling (F)
Judging (J) – Perceiving (P)¶

```
data.head(5)
```

Out[24]:

	type	posts	IE	NS	TF	JP
0	INFJ	'http://www.youtube.com/watch?v=qsXHcwe3krw|||...	1	1	0	1
1	ENTP	'I'm finding the lack of me in these posts ver...	0	1	1	0
2	INTP	'Good one _____ https://www.youtube.com/wat...	1	1	1	0
3	INTJ	'Dear INTP, I enjoyed our conversation the o...	1	1	1	1
4	ENTJ	'You're fired.|||That's another silly misconce...	0	1	1	1

- Using the above code, if a person has I, N, T and J, the value across the 4 axis of MBTI i.e. IE, NS, TF and JP respectively, will be 1. Else 0.
- This will help us calculate for e.g. how many Introvert posts are present v/s how many Extrovert posts are presnt, out of all the given entries in our labelled Kaggle dataset. This is done in order to extplore the dataset for all the individual Personality Indices of MBTI

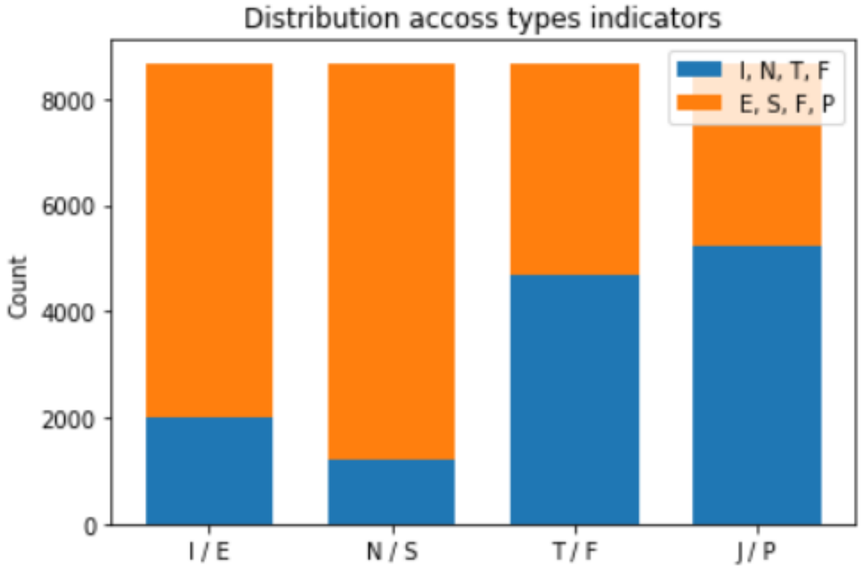
Counting No. of posts in one class / Total no. of posts in the other class

In [25]:
```
print ("Introversion (I) /  Extroversion (E):\t", data['IE'].value_counts()[0], " / ", data['IE'].value_counts()[1])
print ("Intuition (N) / Sensing (S):\t\t", data['NS'].value_counts()[0], " / ", data['NS'].value_counts()[1])
print ("Thinking (T) / Feeling (F):\t\t", data['TF'].value_counts()[0], " / ", data['TF'].value_counts()[1])
print ("Judging (J) / Perceiving (P):\t\t", data['JP'].value_counts()[0], " / ", data['JP'].value_counts()[1])
```

```
Introversion (I) /  Extroversion (E):    1999  /  6676
Intuition (N) / Sensing (S):             1197  /  7478
Thinking (T) / Feeling (F):              4694  /  3981
Judging (J) / Perceiving (P):            5241  /  3434
```

- We infer that there is unequal distribution even among each of the 4 axis in the entries of out dataset. i.e. out of IE:E is the majority, in NS:S is the majority. While TF and JP have realtively less differnce between them.

Plotting the distribution of each personality type indicator

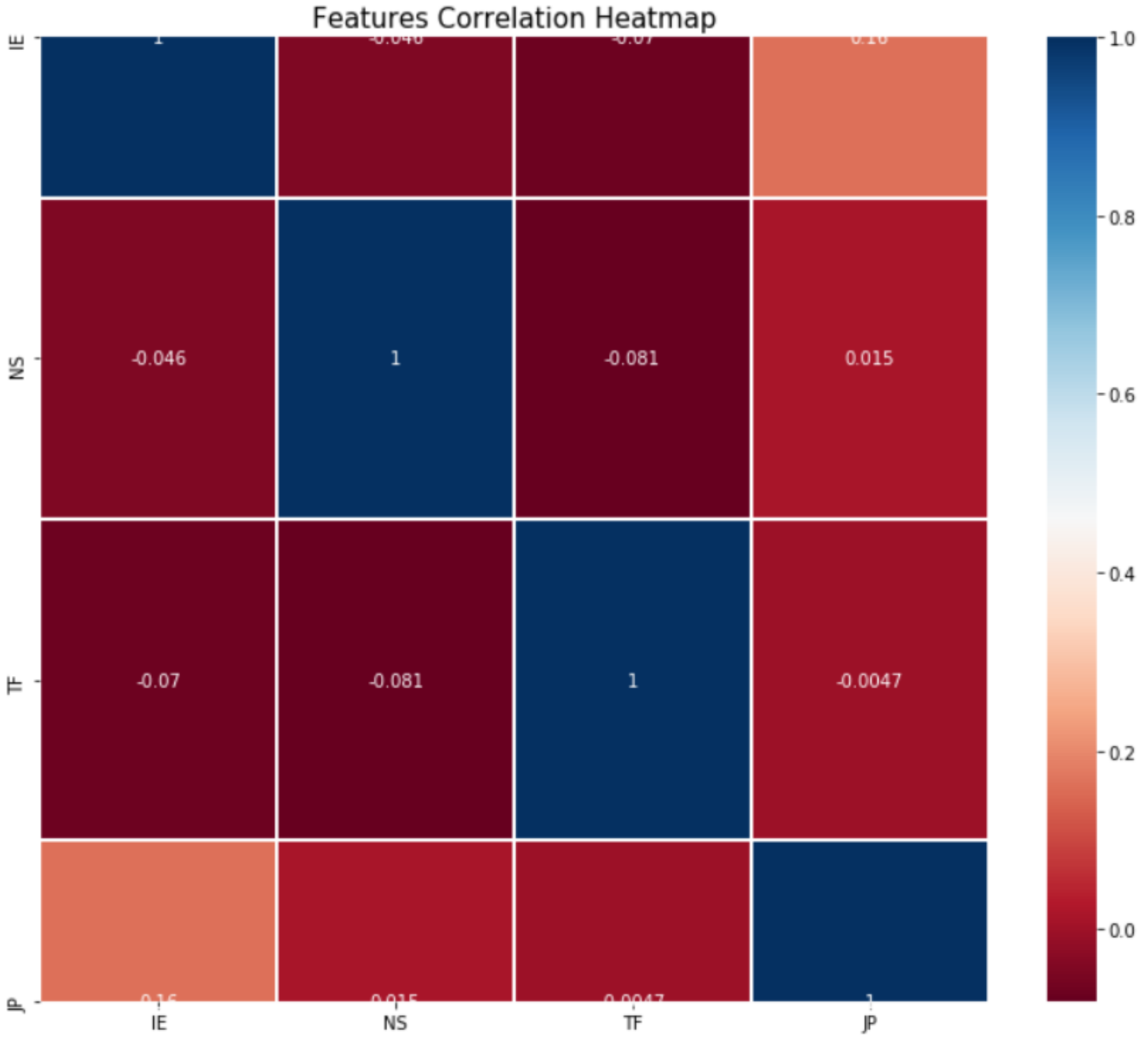


- Fun Fact : The above results match with real life findings by researchers across various personality and psycological studies like
- We can compare this with the fact that Introverts are a minority, making up roughly 16 percent of people [1]. Eventhough among introverts, there are varying degrees, and Carl Jung said, “There is no such thing as a pure Extrovert or a pure introvert" Hence it is tricky to classify a person with 1 type.
- While the population is split roughly 50/50 on the other dimensions, a full 70% of people show a preference for Sensing over Intuition when taking a personality test. Because Intuitives are the minority, the onus is on them to adjust to the Sensor way of thinking.
- The differences between Judging and Perceiving are probably the most marked differences of all the four preferences. People with strong Judging preferences might have a hard time accepting people with strong Perceiving preferences, and vice-versa. On the other hand, a "mixed" couple (one Perceiving and one Judging) can complement each other very well, if they have developed themselves enough to be able to accept each other's differences.

10

FEATURES CORRELATION ANALYSIS

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x2156a2a4808>



11

LEMMATIZATION IN PRE-PROCESSING

- We preprocess the posts by using the Lemmatization technique.

- Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words, hence we use this instead in our model. So it links words with similar meanings to one word.

12 FEATURE ENGINEERING

Tf–idf for feature engineering evaluates how relevant/important a word is to a document in a collection of documents or corpus. As we train individual classifiers here, it is very useful for scoring words in machine learning algorithms for Natural Language Processing.

For our model we vectorize using count vectorizer and tf-idf vectorizer keeping the words appearing btw 10% to 70% of the posts.

Splitting into X and Y variable

Hence we split the features as :

X: User Posts in TF-IDF representation
Y: Personality type in Binarized MBTI form

Let's see how the posts look in TF-IDF representation:
(we have taken 1st post for demonstration)

In [43]:
```
personality_type = [ "IE: Introversion (I) / Extroversion (E)", "NS: Intuition (N) / Sensing (S)",
                     "FT: Feeling (F) / Thinking (T)", "JP: Judging (J) / Perceiving (P)"  ]

for l in range(len(personality_type)):
    print(personality_type[l])
```

```
IE: Introversion (I) / Extroversion (E)
NS: Intuition (N) / Sensing (S)
FT: Feeling (F) / Thinking (T)
JP: Judging (J) / Perceiving (P)
```

Let's see how the posts look in TF-IDF representation: (we have taken 1st post for demonstration)

In [44]:
```
print("X: 1st posts in tf-idf representation\n%s" % X_tfidf[0])
```

```
X: 1st posts in tf-idf representation
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.08105478 0.07066064
 0.         0.         0.         0.         0.         0.
 0.         0.04516864 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.05321691 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.0871647  0.         0.         0.
 0.         0.         0.         0.05506308 0.0708757  0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.16585935 0.         0.         0.09676192 0.
 0.         0.04970682 0.         0.         0.         0.
 0.07397056 0.         0.         0.         0.         0.
 0.         0.0748045  0.07639898 0.09185775 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.05133662 0.         0.09442732
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
```

```
0.         0.08968056 0.         0.         0.0860263  0.
0.         0.         0.         0.06318282 0.         0.
0.         0.04256832 0.         0.         0.         0.
0.06642087 0.         0.         0.         0.09201473 0.
0.         0.0831116  0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.06971149
0.09554125 0.04625983 0.08531558 0.         0.         0.
0.06799661 0.07466644 0.         0.         0.         0.09843694
0.         0.         0.         0.06502346 0.         0.
0.         0.         0.         0.         0.         0.
0.         0.06869092 0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.08067849 0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.0466968  0.0539756  0.08760887 0.
0.1533845  0.         0.         0.         0.         0.09298479
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.10580777 0.
0.         0.11100899 0.13361762 0.         0.         0.
0.06046932 0.         0.08258902 0.         0.         0.2392193
0.         0.09217882 0.         0.04223906 0.         0.
0.08665848 0.04111178 0.         0.         0.16434695 0.04117693
0.         0.         0.         0.07231244 0.         0.
0.         0.         0.         0.         0.         0.
0.11467609 0.09387096 0.         0.         0.         0.
0.         0.         0.04833236 0.         0.         0.
0.       ]
```

Let's see how the posts look in Binarized MBTI personality indicator representation: (we have taken 1st post for demonstration)

In [48]:
```
print("For MBTI personality type : %s" % translate_back(list_personality[0,:]))
print("Y : Binarized MBTI 1st row: %s" % list_personality[0,:])
```

```
For MBTI personality type : INFJ
Y : Binarized MBTI 1st row: [0 0 0 0]
```

Therefore we have successfully converted the textual data into numerical form

13 MODEL TESTING

Out of all the models, seen above we see that on an average XG Boost gives relatively good performance, hence we choose it to build our Perso
prediction model. This will be beneficial as XGBoost model [2] can even be used to evaluate and report on the performance on a test set for the n
during training.

In [ ]:
```
# setup parameters for xgboost
param = {}

param['n_estimators'] = 200 #100
param['max_depth'] = 2 #3
param['nthread'] = 8 #1
param['learning_rate'] = 0.2 #0.1

# Individually training each mbti personlity type
for l in range(len(personality_type)):
    Y = list_personality[:,l]

    # split data into train and test sets
    seed = 7
    test_size = 0.33
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size, random_state=seed)

    # fit model on training data
    model = XGBClassifier(**param)
    model.fit(X_train, y_train)
    # make predictions for test data
    y_pred = model.predict(X_test)
    predictions = [round(value) for value in y_pred]
    # evaluate predictions
    accuracy = accuracy_score(y_test, predictions)
    print("%s Accuracy: %.2f%%" % (personality_type[l], accuracy * 100.0))
```

We find that these accuracies are improved than before. Hence we fine tune the hyperparameters XG boost and then train the Personalty detecti
model.

PERSONALITY PREDICTION #1 - COVER LETTER

Step 11(a)- Personality Prediction 1 - cover letter

In [56]:
```
my_posts  = """ Hi I am 21 years, currently, I am pursuing my graduate degree in computer science and management (Mba Te

# The type is just a dummy so that the data prep function can be reused
mydata = pd.DataFrame(data={'type': ['INFJ'], 'posts': [my_posts]})

my_posts, dummy  = pre_process_text(mydata, remove_stop_words=True, remove_mbti_profiles=True)

my_X_cnt = cntizer.transform(my_posts)
my_X_tfidf =  tfizer.transform(my_X_cnt).toarray()
```

In [58]:
```
print("The result is: ", translate_back(result))
```

```
The result is:  INFJ
```

PERSONALITY PREDICTION #2- POEM

Step 11(b)- Personality Prediction 2 - a poem

In [59]:
```
my_posts = """ They act like they care They tell me to share But when I carve the stories on my arm The doctor just calls i
mydata = pd.DataFrame(data={'type': ['INFP'], 'posts': [my_posts]})
my_posts, dummy  = pre_process_text(mydata, remove_stop_words=True, remove_mbti_profiles=True)
my_X_cnt = cntizer.transform(my_posts)
my_X_tfidf =  tfizer.transform(my_X_cnt).toarray()
```

In [61]:
```
print("The result is: ", translate_back(result))
```

```
The result is:  INTJ
```

PERSONALITY PREDICTION #3 - SHORT ESSAY

Step 11(c)- Pesonality Prediction 3 - short essay

In [62]:
```
my_posts  = """ I dont think anyone would be able to live 300 years i am not talking about the physical ability to do so but the mental f
mydata = pd.DataFrame(data={'type': ['ENTP'], 'posts': [my_posts]})
my_posts, dummy  = pre_process_text(mydata, remove_stop_words=True, remove_mbti_profiles=True)
my_X_cnt = cntizer.transform(my_posts)
my_X_tfidf =  tfizer.transform(my_X_cnt).toarray()
```

In [64]:
```
print("The result is: ", translate_back(result))
```

```
The result is:  INFJ
```

08. CONCLUSION

User behavior and personality detection is a challenging and highly focused area in cognitive-based sentiment analysis.

Predicting personality from online text is a growing trend for researchers. Several studies have already been conducted on predicting personality from the input text

We provided an insight to the
following issues of
personality recognition:

(i) Personality models;
(ii) machine learning approaches for personality recognition;

09. FUTURE SCOPE

- Users can easily identify his personality and his technical skill from this model or system.
- Can be used to predict a person's personality with an accuracy of 85.81%
- Used to identify the right candidate to the right candidate based on his personality and skill
- As a business, if you can understand your customers' behaviors, mirror their linguistic preferences and adapt your message to match their traits, you will be much more successful in attracting, engaging and converting them.

THANK YOU