MBTI Personality Classifier

This programme will classify people into mbti personality types based on their past 50 posts on social media using the basic naivebayesclassifier

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
import string
from nltk.classify import NaiveBayesClassifier
```

Importing the dataset

In [2]:
```
data_set = pd.read_csv("mbti_1.csv")
data_set.tail()
```

Out[2]:

	type	posts
8670	ISFP	'https://www.youtube.com/watch?v=t8edHB_h908||...
8671	ENFP	'So...if this thread already exists someplace ...
8672	INTP	'So many questions when i do these things. I ...
8673	INFP	'I am very conflicted right now when it comes ...
8674	INFP	'It has been too long since I have been on per...

Checking the dataset for missing values

In [3]:
```
data_set.isnull().any()
```

Out[3]:
```
type     False
posts    False
dtype: bool
```

Exploring the dataset

The size of the dataset

In [4]:
```
data_set.shape
```

Out[4]: (8675, 2)

Explroing the posts in posts field

In [5]:
```
data_set.iloc[0,1].split('|||')
```

```
Out[5]: ["'http://www.youtube.com/watch?v=qsXHcwe3krw",
         'http://41.media.tumblr.com/tumblr_lfouy03PMA1qa1rooo1_500.jpg',
         'enfp and intj moments  https://www.youtube.com/watch?v=iz7lE1g4XM4  sportsc
        enter not top ten plays  https://www.youtube.com/watch?v=uCdfze1etec  prank
        s',
         'What has been the most life-changing experience in your life?',
         'http://www.youtube.com/watch?v=vXZeYwwRDw8   http://www.youtube.com/watch?v
        =u8ejam5DP3E  On repeat for most of today.',
         'May the PerC Experience immerse you.',
         'The last thing my INFJ friend posted on his facebook before committing suic
        ide the next day. Rest in peace~   http://vimeo.com/22842206',
         "Hello ENFJ7. Sorry to hear of your distress. It's only natural for a relati
        onship to not be perfection all the time in every moment of existence. Try to
        figure the hard times as times of growth, as...",
         '84389  84390  http://wallpaperpassion.com/upload/23700/friendship-boy-and-g
        irl-wallpaper.jpg  http://assets.dornob.com/wp-content/uploads/2010/04/round-
        home-design.jpg ...',
         'Welcome and stuff.',
         'http://playeressence.com/wp-content/uploads/2013/08/RED-red-the-pokemon-mas
        ter-32560474-450-338.jpg  Game. Set. Match.',
         "Prozac, wellbrutin, at least thirty minutes of moving your legs (and I do
        n't mean moving them while sitting in your same desk chair), weed in moderati
        on (maybe try edibles as a healthier alternative...",
         "Basically come up with three items you've determined that each type (or whi
        chever types you want to do) would more than likely use, given each types' co
        gnitive functions and whatnot, when left by...",
         'All things in moderation.  Sims is indeed a video game, and a good one at t
        hat. Note: a good one at that is somewhat subjective in that I am not complet
        ely promoting the death of any given Sim...',
         'Dear ENFP:  What were your favorite video games growing up and what are you
        r now, current favorite video games? :cool:',
         'https://www.youtube.com/watch?v=QyPqT8umzmY',
         'It appears to be too late. :sad:',
         "There's someone out there for everyone.",
         'Wait... I thought confidence was a good thing.',
         "I just cherish the time of solitude b/c i revel within my inner world more
        whereas most other time i'd be workin... just enjoy the me time while you ca
        n. Don't worry, people will always be around to...",
         "Yo entp ladies... if you're into a complimentary personality,well, hey.",
         '... when your main social outlet is xbox live conversations and even then y
        ou verbally fatigue quickly.',
         'http://www.youtube.com/watch?v=gDhy7rdfm14  I really dig the part from 1:46
        to 2:50',
         'http://www.youtube.com/watch?v=msqXffgh7b8',
         'Banned because this thread requires it of me.',
         'Get high in backyard, roast and eat marshmellows in backyard while conversi
        ng over something intellectual, followed by massages and kisses.',
         'http://www.youtube.com/watch?v=Mw7eoU3BMbE',
         'http://www.youtube.com/watch?v=4V2uYORhQOk',
         'http://www.youtube.com/watch?v=SlVmgFQQ0TI',
         "Banned for too many b's in that sentence. How could you! Think of the B!",
         'Banned for watching movies in the corner with the dunces.',
         'Banned because Health class clearly taught you nothing about peer pressur
        e.',
         'Banned for a whole host of reasons!',
         'http://www.youtube.com/watch?v=IRcrv41hgz4',
         "1) Two baby deer on left and right munching on a beetle in the middle.  2)
```

```
Using their own blood, two cavemen diary today's latest happenings on their d
esignated cave diary wall.  3) I see it as...",
 'a pokemon world  an infj society  everyone becomes an optimist',
 '49142',
 'http://www.youtube.com/watch?v=ZRCEq_JFeFM',
 'http://discovermagazine.com/2012/jul-aug/20-things-you-didnt-know-about-des
erts/desert.jpg',
 'http://oyster.ignimgs.com/mediawiki/apis.ign.com/pokemon-silver-version/d/d
d/Ditto.gif',
 'http://www.serebii.net/potw-dp/Scizor.jpg',
 "Not all artists are artists because they draw. It's the idea that counts in
forming something of your own... like a signature.",
 "Welcome to the robot ranks, person who downed my self-esteem cuz I'm not an
avid signature artist like herself. :proud:",
 'Banned for taking all the room under my bed. Ya gotta learn to share with t
he roaches.',
 'http://www.youtube.com/watch?v=w8IgImn57aQ',
 'Banned for being too much of a thundering, grumbling kind of storm... ye
p.',
 "Ahh... old high school music I haven't heard in ages.   http://www.youtube.
com/watch?v=dcCRUPCdB1w",
 "I failed a public speaking class a few years ago and I've sort of learned w
hat I could do better were I to be in that position again. A big part of my f
ailure was just overloading myself with too...",
 "I like this person's mentality. He's a confirmed INTJ by the way. http://ww
w.youtube.com/watch?v=hGKLI-GEc6M",
 "Move to the Denver area and start a new life for myself.'"]
```

Finding the number of posts

In [6]:
```
len(data_set.iloc[1,1].split('|||'))
```

Out[6]: 50

Finding the unique vales from type of personality column

In [7]:
```
types = np.unique(np.array(data_set['type']))
types
```

Out[7]:
```
array(['ENFJ', 'ENFP', 'ENTJ', 'ENTP', 'ESFJ', 'ESFP', 'ESTJ', 'ESTP',
       'INFJ', 'INFP', 'INTJ', 'INTP', 'ISFJ', 'ISFP', 'ISTJ', 'ISTP'],
      dtype=object)
```

The total number of posts for each type

In [8]:
```
total = data_set.groupby(['type']).count()*50
total
```

Out[8]:

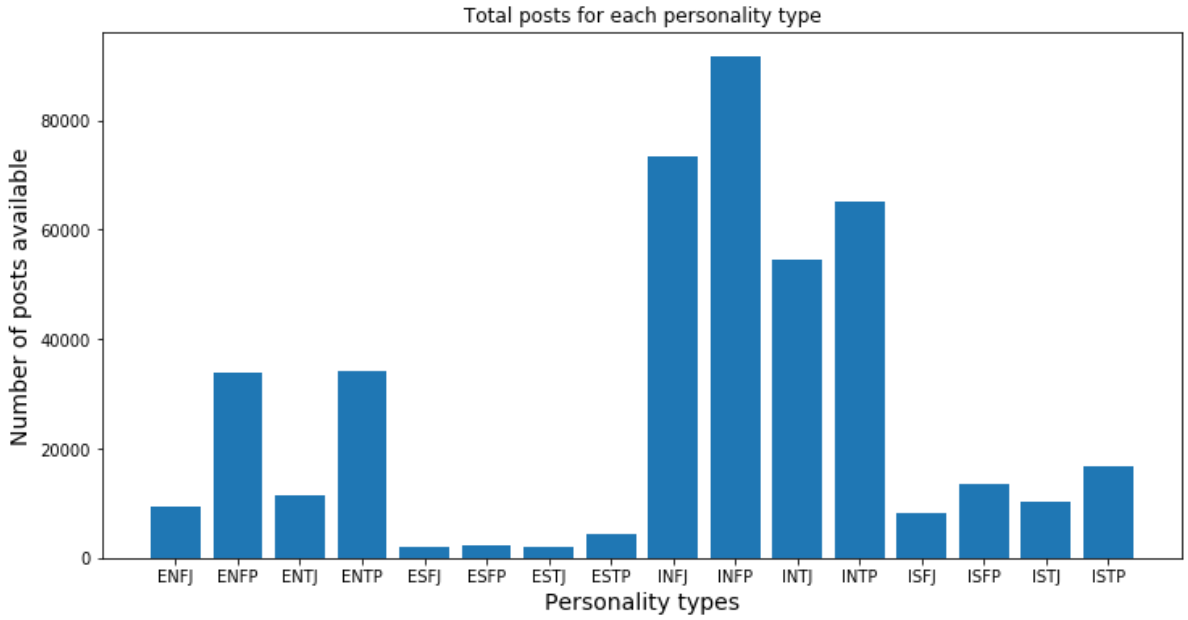
	posts
type	
ENFJ	9500
ENFP	33750
ENTJ	11550
ENTP	34250
ESFJ	2100
ESFP	2400
ESTJ	1950
ESTP	4450
INFJ	73500
INFP	91600
INTJ	54550
INTP	65200
ISFJ	8300
ISFP	13550
ISTJ	10250
ISTP	16850

Graphing it for better visualization

In [9]:
```
plt.figure(figsize = (12,6))

plt.bar(np.array(total.index), height = total['posts'],)
plt.xlabel('Personality types', size = 14)
plt.ylabel('Number of posts available', size = 14)
plt.title('Total posts for each personality type')
```

Out[9]: Text(0.5,1,'Total posts for each personality type')



Organising the data to create a bag words model

Segrating all the posts by their personality types and ***creating a new dataframe to store all this in***

In [10]:
```
all_posts= pd.DataFrame()
for j in types:
    temp1 = data_set[data_set['type']==j]['posts']
    temp2 = []
    for i in temp1:
        temp2+=i.split('|||')
    temp3 = pd.Series(temp2)
    all_posts[j] = temp3
```

In [11]:
```
all_posts.tail()
```

Out[11]:

	ENFJ	ENFP	ENTJ	ENTP	ES
9283	I am VERY particular. And I have not dated cer...	'Very true. The thing is to recognize and ove...	Um, totally off ROFL. I am female, 30, no opin...	Actually I was none of these labels of alpha o...	Na
9284	Could you not tell the guy was INTP? Is that ...	i'm not real sure it's a great idea to date un...	Hahah that's hilarious	Double post, dammit. >_<	Na
9285	I cannot speak for all ENFJ's, but I am hard-w...	Watch some comedies, kids are joyful and make ...	https://www.onlineassessmenttool.com/instinctu...	Radicalism 83 Socialism 100 Tenderness 71.875 ...	Na
9286	Of course you do not see the point of the vide...	Meditation is great! Read up on TFT and EFT ta...	ENTJ - idunno...	Quoted for astonishing truth. One thing thoug...	Na
9287	The types fit because the way they use their c...	2 Physical Touch 7 Quality Time 7 Words of...	I miscarried recently and was quite distraught...	Sure. I was taken aback by what I saw (and pre...	Na

Creating a function to tokenize the words

In [12]:
```
useless_words = nltk.corpus.stopwords.words("english") + list(string.punctuation)
def build_bag_of_words_features_filtered(words):
    words = nltk.word_tokenize(words)
    return {
        word:1 for word in words \
        if not word in useless_words}
```

A random check of the function

In [13]:
```
build_bag_of_words_features_filtered(all_posts['INTJ'].iloc[1])
```

Out[13]:
```
{'Dear': 1,
 'ENTJ': 1,
 'sub': 1,
 'Long': 1,
 'time': 1,
 'see': 1,
 'Sincerely': 1,
 'Alpha': 1}
```

Creating an array of features

In [14]:
```
features=[]
for j in types:
    temp1 = all_posts[j]
    temp1 = temp1.dropna() #not all the personality types have same number of
 files
    features += [[(build_bag_of_words_features_filtered(i), j) \
    for i in temp1]]
```

Because each number of personality types have different number of posts they must be splitted accordingle. Taking 80% for training and 20% for testing

In [15]:
```
split=[]
for i in range(16):
    split += [len(features[i]) * 0.8]
split = np.array(split,dtype = int)
```

In [16]:
```
split
```

Out[16]:
```
array([7430, 7430, 7430, 7430, 1614, 1772, 1536, 3469, 7430, 7430, 7430,
       7430, 6496, 7430, 7430, 7430])
```

Data for training

In [17]:
```
train=[]
for i in range(16):
    train += features[i][:split[i]]
```

Training the model

In [18]:
```
sentiment_classifier = NaiveBayesClassifier.train(train)
```

Testing the model on the dataset it was trained for accuracy

In [19]:
```
nltk.classify.util.accuracy(sentiment_classifier, train)*100
```

Out[19]: 43.904281855160065

Creating the test data

In [20]:
```
test=[]
for i in range(16):
    test += features[i][split[i]:]
```

Testing the model on the test dataset which it has never seen before

In [21]:
```
nltk.classify.util.accuracy(sentiment_classifier, test)*100
```

Out[21]: 10.238794851632662

The model performs at efficieny of only 10% which is pretty bad.

Hence, instead of selecting all 16 types of personalitys as a unique feature I explored the dataset further and decided to simplify it.

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

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

We will use this and create 4 classifyers to classify the person

Creating a classifyer for Introversion (I) and Extroversion (E)

Note: The details for the steps over here are same as the ones while creating the model above, hence I will only explain the changes

In [22]:
```
# Features for the bag of words model
features=[]
for j in types:
    temp1 = all_posts[j]
    temp1 = temp1.dropna() #not all the personality types have same number of
 files
    if('I' in j):
        features += [[(build_bag_of_words_features_filtered(i), 'introvert') \
        for i in temp1]]
    if('E' in j):
        features += [[(build_bag_of_words_features_filtered(i), 'extrovert') \
        for i in temp1]]
```

Data for training

In [23]:
```
train=[]
for i in range(16):
    train += features[i][:split[i]]
```

Training the model

In [24]:
```
IntroExtro = NaiveBayesClassifier.train(train)
```

Testing the model on the dataset it was trained for accuracy

In [25]:
```
nltk.classify.util.accuracy(IntroExtro, train)*100
```

Out[25]: 81.12443979837917

Creating the test data

In [26]:
```
test=[]
for i in range(16):
    test += features[i][split[i]:]
```

Testing the model on the test dataset which it has never seen before

In [27]:
```
nltk.classify.util.accuracy(IntroExtro, test)*100
```

Out[27]: 58.20469312585358

Seeing that this model has good somewhat good results, I shall repeat the same with the rest of the traits

Creating a classifyer for Intuition (N) and Sensing (S)

In [28]:
```
# Features for the bag of words model
features=[]
for j in types:
    temp1 = all_posts[j]
    temp1 = temp1.dropna() #not all the personality types have same number of
 files
    if('N' in j):
        features += [[(build_bag_of_words_features_filtered(i), 'Intuition') \
        for i in temp1]]
    if('E' in j):
        features += [[(build_bag_of_words_features_filtered(i), 'Sensing') \
        for i in temp1]]
```

Data for training

In [29]:
```
train=[]
for i in range(16):
    train += features[i][:split[i]]
```

Training the model

In [30]:
```
IntuitionSensing = NaiveBayesClassifier.train(train)
```

Testing the model on the dataset it was trained for accuracy

In [31]:
```
nltk.classify.util.accuracy(IntuitionSensing, train)*100
```

Out[31]: 70.14524215640667

Creating the test data

In [32]:
```
test=[]
for i in range(16):
    test += features[i][split[i]:]
```

Testing the model on the test dataset which it has never seen before

In [33]:
```
nltk.classify.util.accuracy(IntuitionSensing, test)*100
```

Out[33]: 54.46262259027357

Creating a classifyer for Thinking (T) and Feeling (F)

In [34]:
```
# Features for the bag of words model
features=[]
for j in types:
    temp1 = all_posts[j]
    temp1 = temp1.dropna() #not all the personality types have same number of
 files
    if('T' in j):
        features += [[(build_bag_of_words_features_filtered(i), 'Thinking') \
        for i in temp1]]
    if('F' in j):
        features += [[(build_bag_of_words_features_filtered(i), 'Feeling') \
        for i in temp1]]
```

Data for training

In [35]:
```
train=[]
for i in range(16):
    train += features[i][:split[i]]
```

Training the model

In [36]:
```
ThinkingFeeling = NaiveBayesClassifier.train(train)
```

Testing the model on the dataset it was trained for accuracy

In [37]:
```
nltk.classify.util.accuracy(ThinkingFeeling, train)*100
```

Out[37]: 80.03456948570128

Creating the test data

In [38]:
```
test=[]
for i in range(16):
    test += features[i][split[i]:]
```

Testing the model on the test dataset which it has never seen before

In [39]:
```
nltk.classify.util.accuracy(ThinkingFeeling, test)*100
```

Out[39]: 59.41315234035509

Creating a classifyer for Judging (J) and Percieving (P)

In [40]:
```
# Features for the bag of words model
features=[]
for j in types:
    temp1 = all_posts[j]
    temp1 = temp1.dropna() #not all the personality types have same number of
 files
    if('J' in j):
        features += [[(build_bag_of_words_features_filtered(i), 'Judging') \
        for i in temp1]]
    if('P' in j):
        features += [[(build_bag_of_words_features_filtered(i), 'Percieving')
 \
        for i in temp1]]
```

Data for training

In [41]:
```
train=[]
for i in range(16):
    train += features[i][:split[i]]
```

Training the model

In [42]:
```
JudgingPercieiving = NaiveBayesClassifier.train(train)
```

Testing the model on the dataset it was trained for accuracy

In [43]:
```
nltk.classify.util.accuracy(JudgingPercieiving, train)*100
```

Out[43]: 79.79341109742592

Creating the test data

In [44]:
```
test=[]
for i in range(16):
    test += features[i][split[i]:]
```

Testing the model on the test dataset which it has never seen before

In [45]:
```
nltk.classify.util.accuracy(JudgingPercieiving, test)*100
```

Out[45]: 54.40549600629061

Summarizing the results of the models

In [46]:
```
temp = {'train' : [81.12443979837917,70.14524215640667,80.03456948570128,79.79
341109742592], 'test' : [58.20469312585358,54.46262259027357,59.41315234035509
,54.40549600629061]}
results = pd.DataFrame.from_dict(temp, orient='index', columns=['Introvert - E
xtrovert', 'Intuition - Sensing', 'Thinking - Feeling', 'Judging - Percieivin
g'])
results
```

Out[46]:

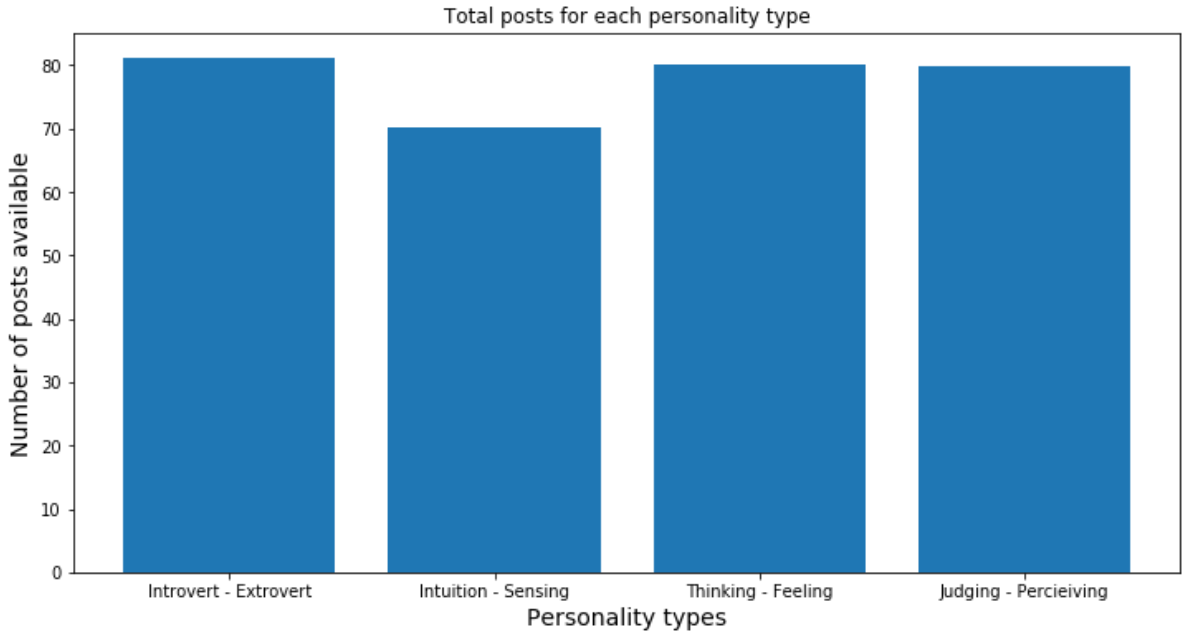
	Introvert - Extrovert	Intuition - Sensing	Thinking - Feeling	Judging - Percieiving
train	81.124440	70.145242	80.034569	79.793411
test	58.204693	54.462623	59.413152	54.405496

Plotting the results for better appeal

In [47]:
```
plt.figure(figsize = (12,6))

plt.bar(np.array(results.columns), height = results.loc['train'],)
plt.xlabel('Personality types', size = 14)
plt.ylabel('Number of posts available', size = 14)
plt.title('Total posts for each personality type')
```

Out[47]: Text(0.5,1,'Total posts for each personality type')



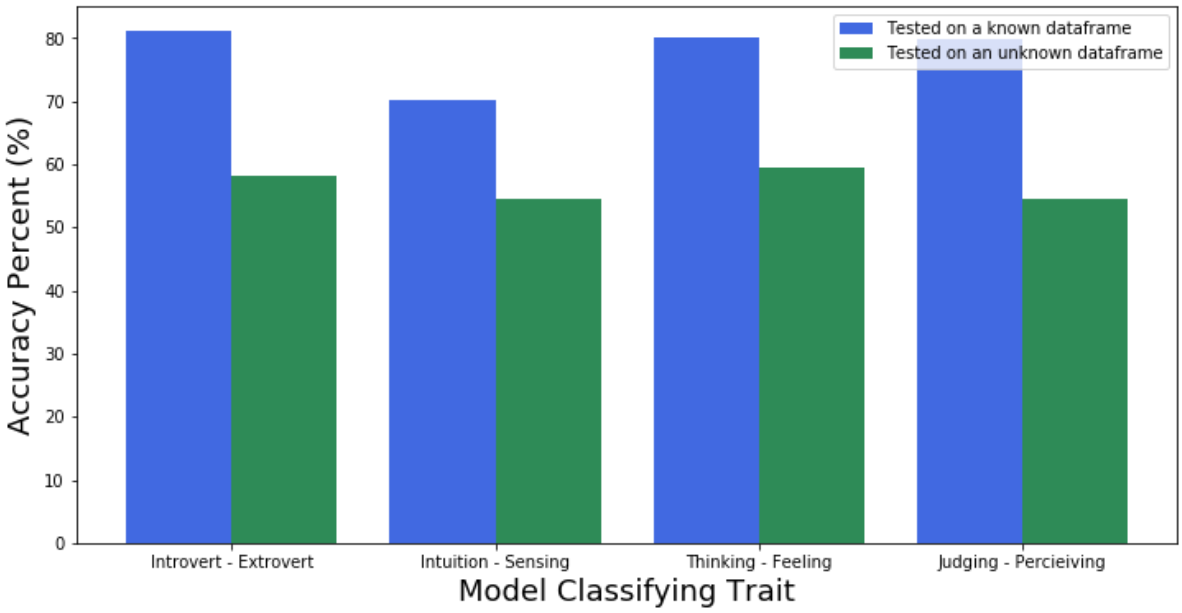
In [48]:
```
labels = np.array(results.columns)

training = results.loc['train']
ind = np.arange(4)
width = 0.4
fig = plt.figure()
ax = fig.add_subplot(111)
rects1 = ax.bar(ind, training, width, color='royalblue')

testing = results.loc['test']
rects2 = ax.bar(ind+width, testing, width, color='seagreen')

fig.set_size_inches(12, 6)
fig.savefig('Results.png', dpi=200)

ax.set_xlabel('Model Classifying Trait', size = 18)
ax.set_ylabel('Accuracy Percent (%)', size = 18)
ax.set_xticks(ind + width / 2)
ax.set_xticklabels(labels)
ax.legend((rects1[0], rects2[0]), ('Tested on a known dataframe', 'Tested on a
n unknown dataframe'))
plt.show()
```



Testing the models to predict my trait my feeding few of my quora writings

link to my quora answers feed: https://www.quora.com/profile/Divya-Bramhecha (https://www.quora.com/profile/Divya-Bramhecha)

Defining a functions that inputs the writings, tokenizes them and then predicts the output based on our earlier classifiers

In [49]:
```
def MBTI(input):
    tokenize = build_bag_of_words_features_filtered(input)
    ie = IntroExtro.classify(tokenize)
    Is = IntuitionSensing.classify(tokenize)
    tf = ThinkingFeeling.classify(tokenize)
    jp = JudgingPercieiving.classify(tokenize)

    mbt = ''

    if(ie == 'introvert'):
        mbt+='I'
    if(ie == 'extrovert'):
        mbt+='E'
    if(Is == 'Intuition'):
        mbt+='N'
    if(Is == 'Sensing'):
        mbt+='S'
    if(tf == 'Thinking'):
        mbt+='T'
    if(tf == 'Feeling'):
        mbt+='F'
    if(jp == 'Judging'):
        mbt+='J'
    if(jp == 'Percieving'):
        mbt+='P'
    return(mbt)
```

Building another functions that takes all of my posts as input and outputs the graph showing percentage of each trait seen in each posts and sums up displaying your personality as the graph title

Note: The input should be an array of your posts

In [50]:
```
def tellmemyMBTI(input):
    a = []
    trait1 = pd.DataFrame([0,0,0,0],['I','N','T','J'],['count'])
    trait2 = pd.DataFrame([0,0,0,0],['E','S','F','P'],['count'])
    for i in input:
        a += [MBTI(i)]
    for i in a:
        for j in ['I','N','T','J']:
            if(j in i):
                trait1.loc[j]+=1
        for j in ['E','S','F','P']:
            if(j in i):
                trait2.loc[j]+=1
    trait1 = trait1.T
    trait1 = trait1*100/len(input)
    trait2 = trait2.T
    trait2 = trait2*100/len(input)


    #Finding the personality
    YourTrait = ''
    for i,j in zip(trait1,trait2):
        temp = max(trait1[i][0],trait2[j][0])
        if(trait1[i][0]==temp):
            YourTrait += i
        if(trait2[j][0]==temp):
            YourTrait += j


    #Plotting

    labels = np.array(results.columns)

    intj = trait1.loc['count']
    ind = np.arange(4)
    width = 0.4
    fig = plt.figure()
    ax = fig.add_subplot(111)
    rects1 = ax.bar(ind, intj, width, color='royalblue')

    esfp = trait2.loc['count']
    rects2 = ax.bar(ind+width, esfp, width, color='seagreen')

    fig.set_size_inches(10, 7)
    fig.savefig('Results.png', dpi=200)

    ax.set_xlabel('Finding the MBTI Trait', size = 18)
    ax.set_ylabel('Trait Percent (%)', size = 18)
    ax.set_xticks(ind + width / 2)
    ax.set_xticklabels(labels)
    ax.set_yticks(np.arange(0,105, step= 10))
    ax.set_title('Your Personality is '+YourTrait,size = 20)
    plt.grid(True)
```

```
    plt.show()
```

Importing my quora answers from a text file

I copied all my answer from the link i provided before (i broke down the paragraphs as separte posts)

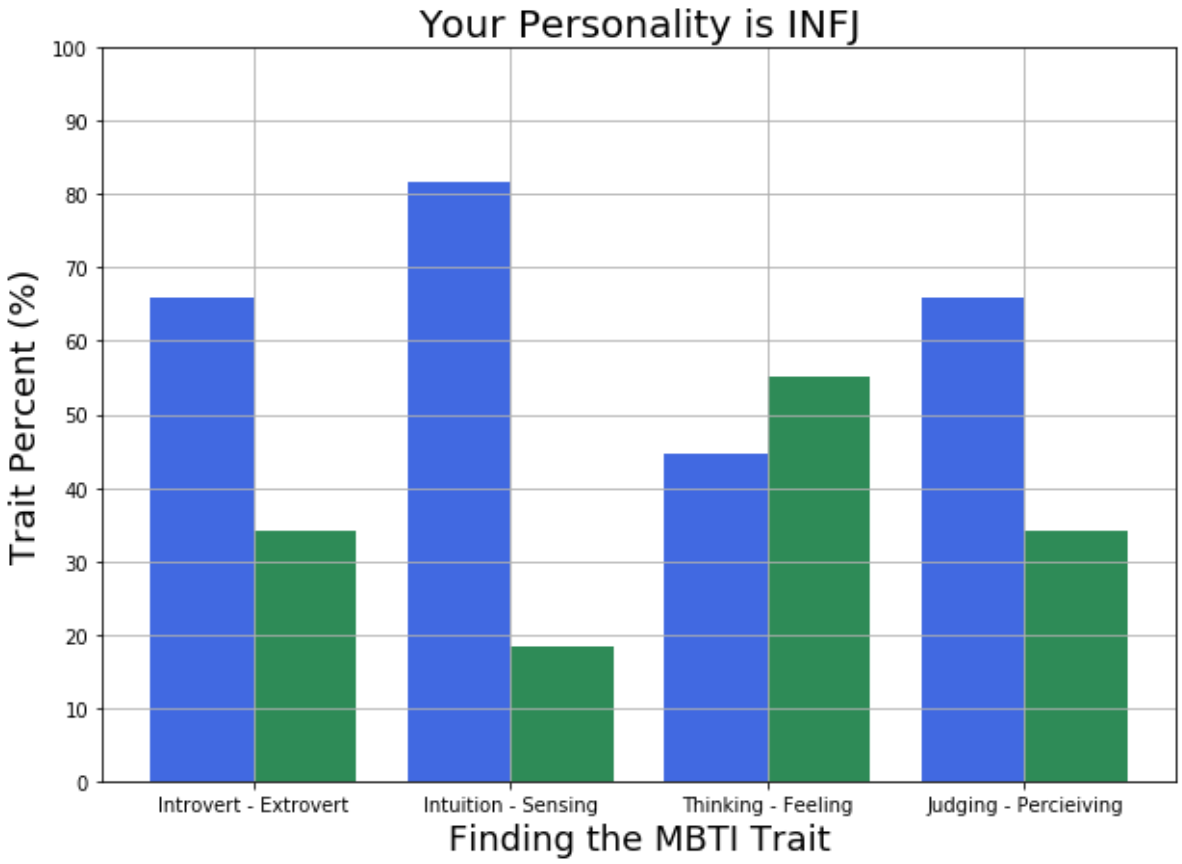
In [51]:
```
My_writings = open("Myquora.txt")
my_writing = My_writings.readlines()
#my_writing
```

In [52]:
```
my_posts = my_writing[0].split('|||')
len(my_posts)
#my_posts
```

Out[52]: 38

Using the classifier to predict my personality type

In [53]:
```
tellmemyMBTI(my_posts)
```



Concluding note

My profile according to https://www.16personalities.com/ (https://www.16personalities.com/) is INTJ.

I am pretty happy that using such a basic model it was pretty close to my real profile, only 1 different. And even that difference was very close, between 10% inaccuary which pretty good.

Although, I am not sure how the classifier will perform on all test cases in general. Specially, the data for some profiles was very less.