Pre-Processing and Exploratory Data Analysis

```
from google.colab import drive
drive.mount('/content/drive')
```

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

```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm.notebook import tqdm
import pickle
import warnings
import spacy
import nltk
import regex as re
from nltk.corpus import stopwords
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from textblob import TextBlob, Word
from sklearn.feature_extraction.text import CountVectorizer
import plotly.express as px
from collections import Counter
from string import punctuation
from nltk.tokenize import word_tokenize
warnings.filterwarnings('ignore')
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from imblearn.pipeline import make_pipeline as make_imbal_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_classification
from sklearn.metrics import classification_report, confusion_matrix,accuracy_score, recall_score,plot_confusion_matrix
from sklearn.ensemble import RandomForestClassifier, HistGradientBoostingClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.calibration import CalibratedClassifierCV, CalibrationDisplay
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
from sklearn.metrics import f1_score
from keras.models import Sequential
import tensorflow as tf
from keras.layers import *

%matplotlib inline
```

```
# Read df
df = pd.read_csv('/content/drive/MyDrive/AML Project/mbti_1.csv')

# Global Vars
nltk.download('stopwords')
nltk.download('punkt')
nlp = spacy.load("en_core_web_sm")
cachedStopWords = stopwords.words("english")
types = df['type'].tolist()
set_types = set([i.lower() for i in types])
print(set_types)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
{'intp', 'enfj', 'isfj', 'isfp', 'entj', 'infj', 'istj', 'entp', 'enfp', 'estp', 'istp', 'esfp', 'estj', 'infp', 'intj', 'esfj'}
```

Text Pre-processing & Cleaning

```
#function to remove stop words like the, is, a..etc

def remove_stop(row):
  global cachedStopWords
  global set_types

  row = ' '.join([word for word in row.split() if word not in cachedStopWords])
```

```
  row = ' '.join([word for word in row.split() if word not in set_types])
  return row

#function to perform lemmatization

def lemmatize(row):
  doc = nlp(row)
  return ' '.join([token.lemma_ for token in doc])


#function to get the keywords from the posts

def get_keywords(text):
    keywords = []
    pos_tag = ['PROPN', 'ADJ', 'NOUN']
    doc = nlp(text)
    for token in doc:

        if(token.text in punctuation):
            keywords.append(token.text)

        if(token.pos_ in pos_tag):
            keywords.append(token.text)
    return ' '.join(word for word in keywords)


#remove unwanted spaces

def remove_unwanted_space(text):
    val1 = '.'
    sentences = text.split('.')
    updated_sentences = []
    for sentence in sentences:
        updated_sentences.append(sentence.strip())
    try:
        while True:
            updated_sentences.remove(val1)
    except ValueError:
        pass
    val2 = ''
    try:
        while True:
            updated_sentences.remove(val2)
    except ValueError:
        pass
    updated_text = ". ".join(updated_sentences)
    return updated_text


#main function which peforms all the text pre-processing

def process_text(df):

  df['posts'] = df['posts'].apply(lambda x: x.lower())
  df['posts'] = df['posts'].apply(lambda x: re.sub(r'http\S+', '', x))
  df['posts'] = df['posts'].apply(lambda x: x.replace("'", ""))

  df['posts'] = df['posts'].apply(lambda x: re.sub(r'[^ a-z\.]+', '', x))
  df['posts'] = df['posts'].apply(lambda x: remove_stop(x))
  df['posts'] = df['posts'].apply(lambda x: lemmatize(x))
  df['posts'] = df['posts'].apply(lambda x: get_keywords(x))
  df['posts'] = df['posts'].apply(lambda x: remove_unwanted_space(x))
  return df


df = process_text(df)
```

Word Vector Embeddings

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove*.zip
!ls
!pwd
print('Indexing word vectors.')

embeddings_index = {}
f = open('glove.6B.100d.txt', encoding='utf-8')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
```

```
f.close()

print('Found %s word vectors.' % len(embeddings_index))
```

```
--2022-12-04 20:57:20--  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]
--2022-12-04 20:57:20--  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]
--2022-12-04 20:57:21--  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  5.11MB/s    in 3m 5s

2022-12-04 21:00:26 (4.44 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]

Archive:  glove.6B.zip
  inflating: glove.6B.50d.txt
  inflating: glove.6B.100d.txt
  inflating: glove.6B.200d.txt
  inflating: glove.6B.300d.txt
drive              glove.6B.200d.txt  glove.6B.50d.txt  sample_data
glove.6B.100d.txt  glove.6B.300d.txt  glove.6B.zip
/content
Indexing word vectors.
Found 400000 word vectors.
```

```
#Functions to convert words into vectors

def do_embedding(row):
  vector_list = []
  for word in row:
    try:
      vector_list.append(embeddings_index[word])
    except:
      pass
  return vector_list

def word_embeddings(df):
  df['vectors'] = df['posts'].apply(lambda x: do_embedding(x))
  return df


# Save df
vectorized_df = word_embeddings(df)


print(len(df), len(vectorized_df))
vectorized_df
```

```
8675 8675
```

	type	posts	vectors
0	INFJ	moment sportscenter top prankswhat lifechangin...	[[0.29492, 0.56874, -0.20245, 0.50244, -0.6829...
1	ENTP	lack post alarming. sex bore position. example...	[[-0.45433, 1.0234, 0.024278, -0.086367, -0.69...
2	INTP	good course blessing positive good friend amaz...	[[-0.37628, 0.37102, 0.32594, -0.085084, -0.55...
3	INTJ	dear conversation day. esoteric gabbing nature...	[[-0.91091, 0.50459, 0.058175, -0.78618, 0.088...
4	ENTJ	silly misconception. approach key unlocking en...	[[0.13739, 0.77891, 0.80054, 0.13819, -0.49792...
...	...	...	...
8670	ISFP	cat fi dom reason. website neo nazis perc. im ...	[[-0.11752, 0.97272, -0.29021, 0.25914, -0.426...
8671	ENFP	thread someplace hereooop hard movie watch thr...	[[0.13482, 0.40224, -0.42266, -0.055631, -0.55...
8672	INTP	many question thing. purple pill. win lottery ...	[[0.29492, 0.56874, -0.20245, 0.50244, -0.6829...
8673	INFP	conflict right child. maternal instinct. none ...	[[-0.11752, 0.97272, -0.29021, 0.25914, -0.426...
8674	INFP	personalitycafe change bit good. doctor world ...	[[-0.6517, 0.80484, 0.048731, 0.37962, -1.1151...

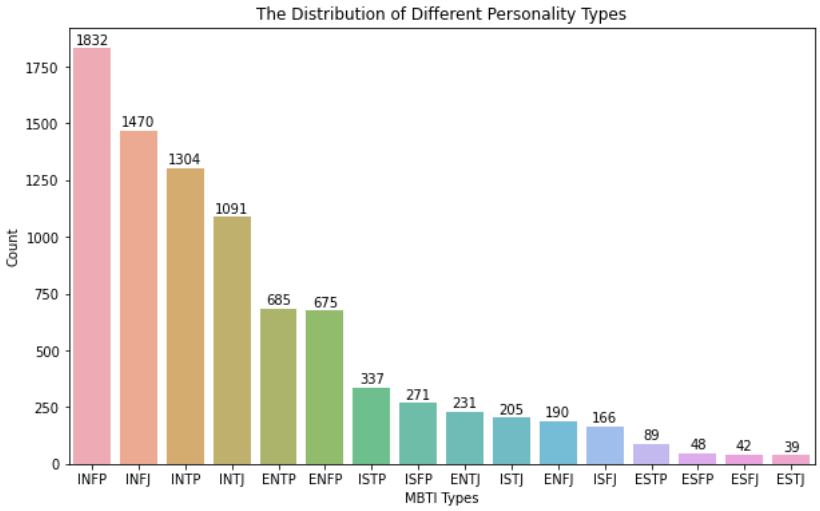
8675 rows × 3 columns

```
df = vectorized_df
```

EDA

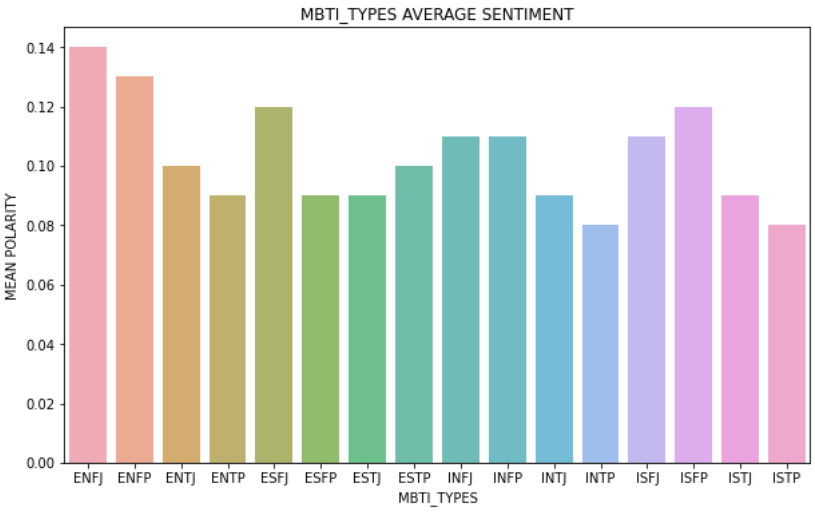
```
data = vectorized_df
```

```
x = data.type.value_counts()
plt.figure(figsize=(10,6))
ax = sns.barplot(x.index, x.values, alpha=0.8)
plt.title('The Distribution of Different Personality Types')
plt.ylabel('Count')
plt.xlabel('MBTI Types')
rects = ax.patches
labels = x.values
for rect, label in zip(rects, labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2, height + 5, label, ha='center', va='bottom')
plt.show();
```



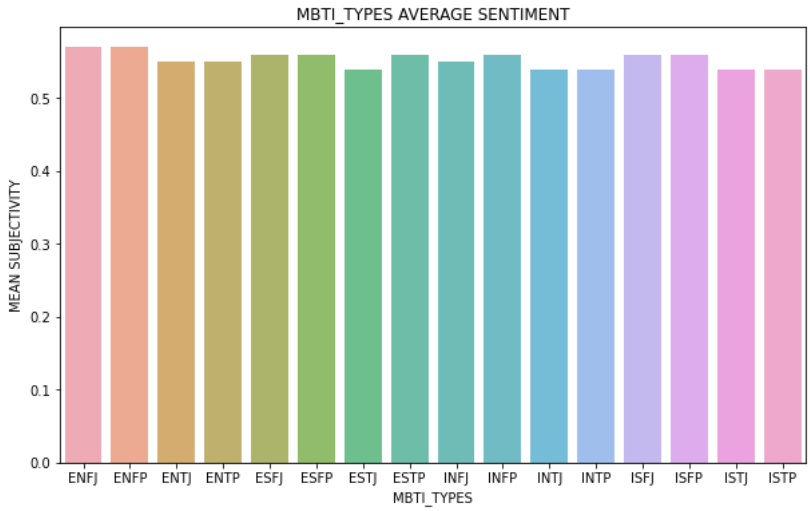

```
#calculating the polarity scores of every post to understand sentiment

data['polarity'] = data['posts'].map(lambda x:TextBlob(x).sentiment.polarity)
```

```
x = round(data.groupby('type')['polarity'].mean(), 2)
plt.figure(figsize=(10,6))
ax = sns.barplot(x.index, x.values, alpha=0.8)
plt.title("MBTI_TYPES AVERAGE SENTIMENT")
plt.ylabel('MEAN POLARITY')
plt.xlabel('MBTI_TYPES')
plt.show()
```




```
#calculating the subjectivity scores of every post to understand the difference between fact/opinion

data['subjectivity'] = data['posts'].map(lambda x:TextBlob(x).sentiment.subjectivity)
```

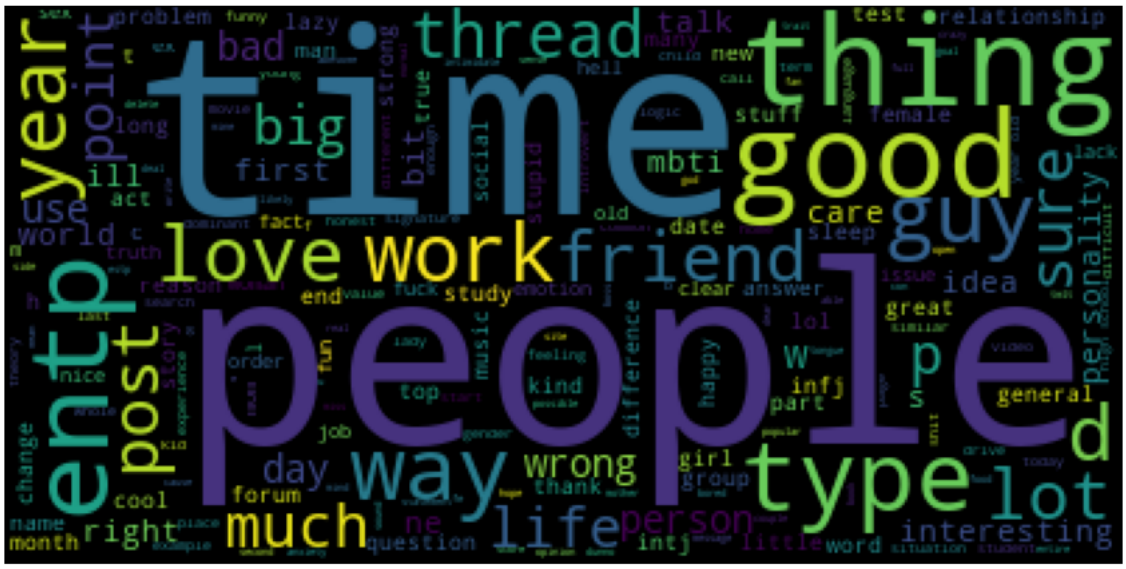
```
x = round(data.groupby('type')['subjectivity'].mean(), 2)
plt.figure(figsize=(10,6))
ax = sns.barplot(x.index, x.values, alpha=0.8)
plt.title("MBTI_TYPES AVERAGE SENTIMENT")
plt.ylabel('MEAN SUBJECTIVITY')
plt.xlabel('MBTI_TYPES')
plt.show()
```



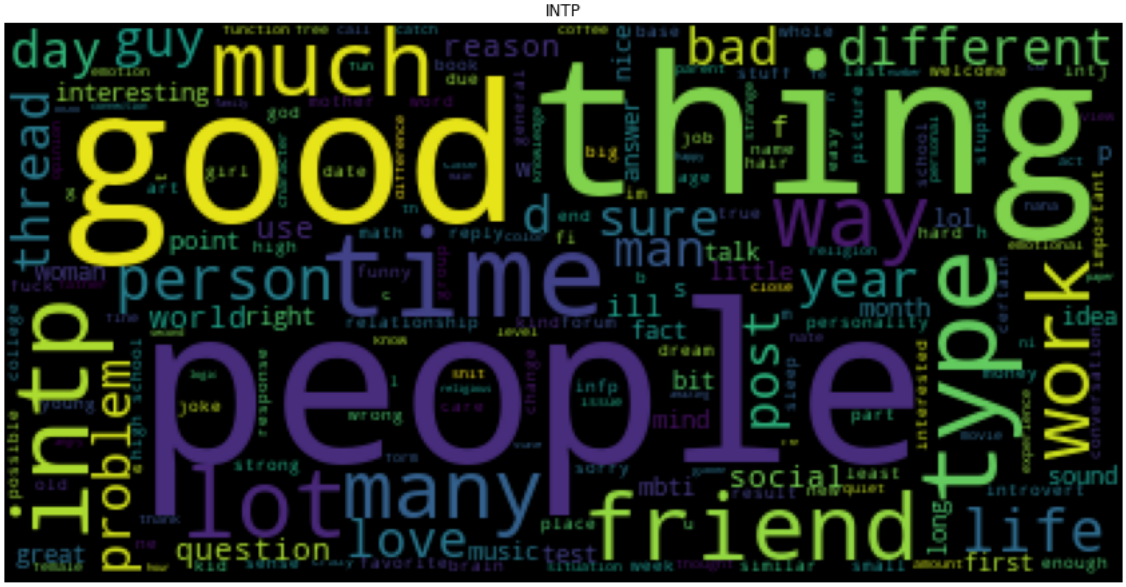

```
def generate_wordcloud(text, title):
    # Create and generate a word cloud image:
    wordcloud = WordCloud(background_color="white").generate(text)

    # Display the generated image:
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.title(title, fontsize = 40)
    plt.show()


df_by_personality = data.groupby("type")['posts'].apply(' '.join).reset_index()


#Generating word clouds and histograms to understand the most commonly used words for each personality type

fig, ax = plt.subplots(len(data['type'].unique()), sharex=True, figsize=(15,10*len(data['type'].unique())))

k = 0
for i in data['type'].unique():
    df_4 = data[data['type'] == i]
    wordcloud = WordCloud().generate(df_4['posts'].to_string())
    ax[k].imshow(wordcloud)
    ax[k].set_title(i)
    ax[k].axis("off")
    k+=1
```

INFJ

ENTP



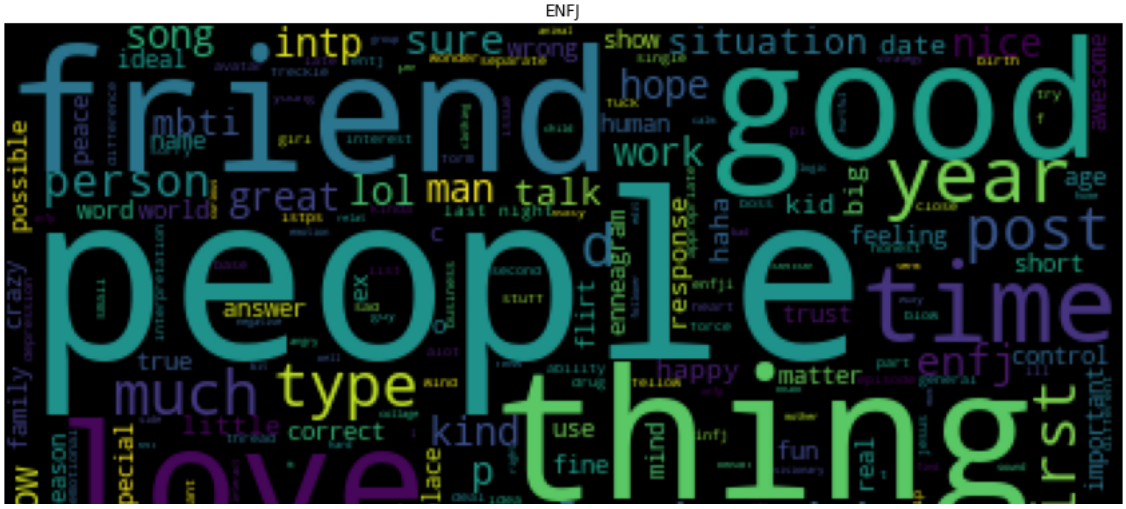
INTP



INTJ

ENTJ

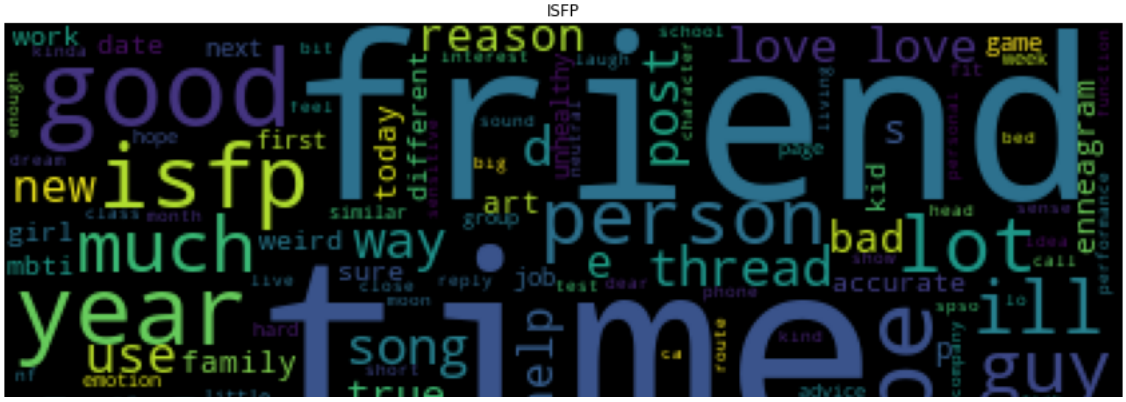
ENFJ



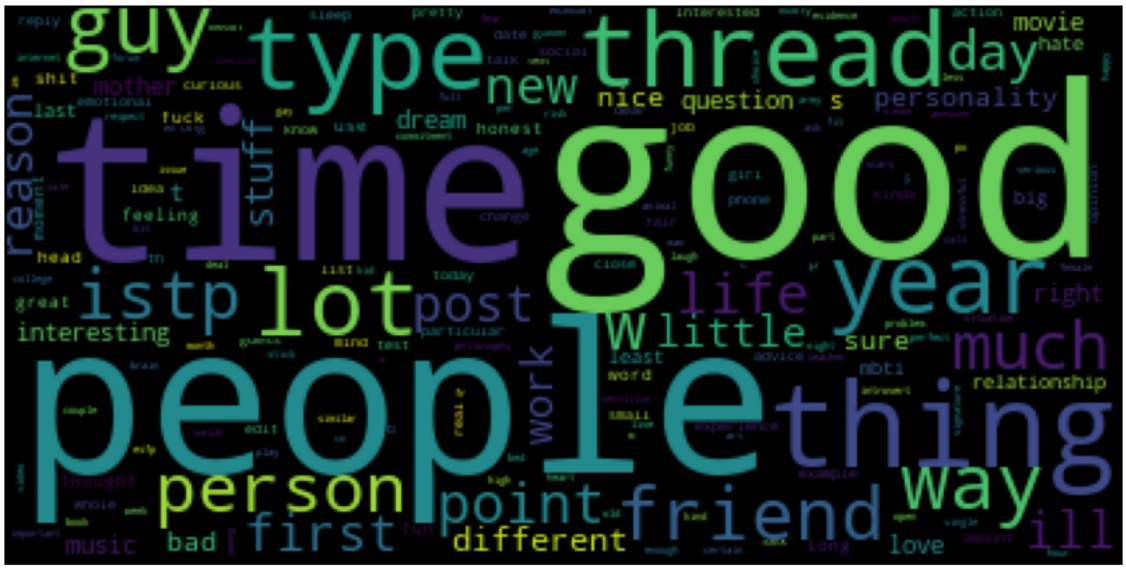
INFP

ENFP

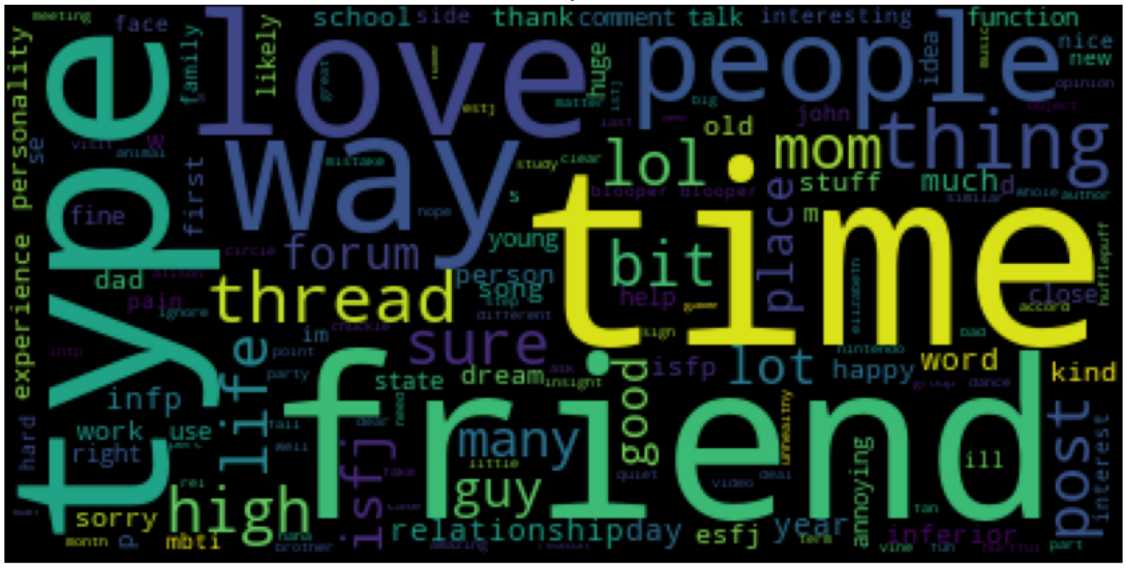
ISFP



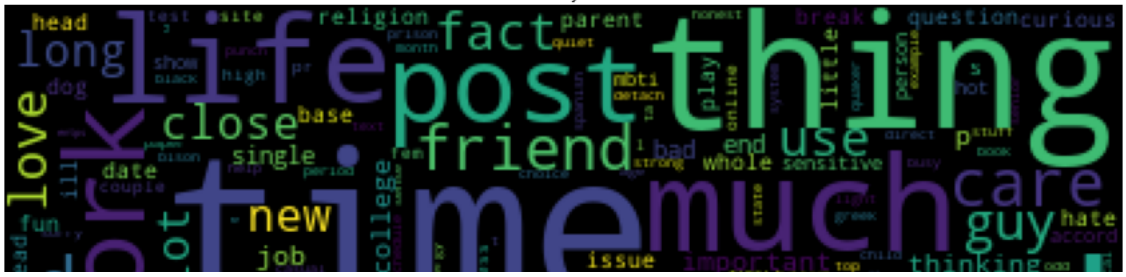
ISTP



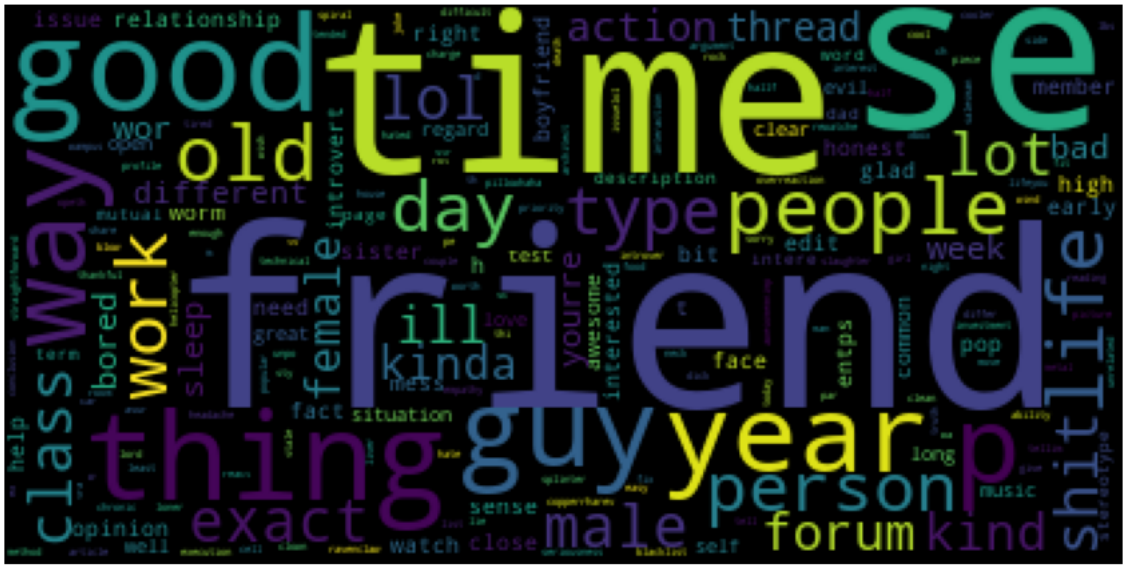
ISFJ



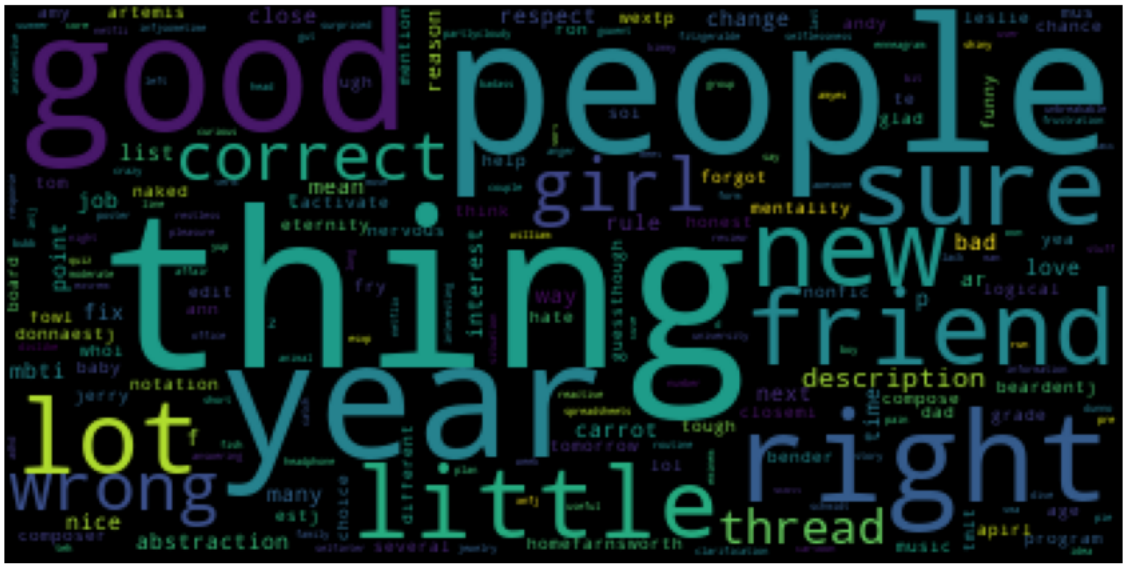
ISTJ



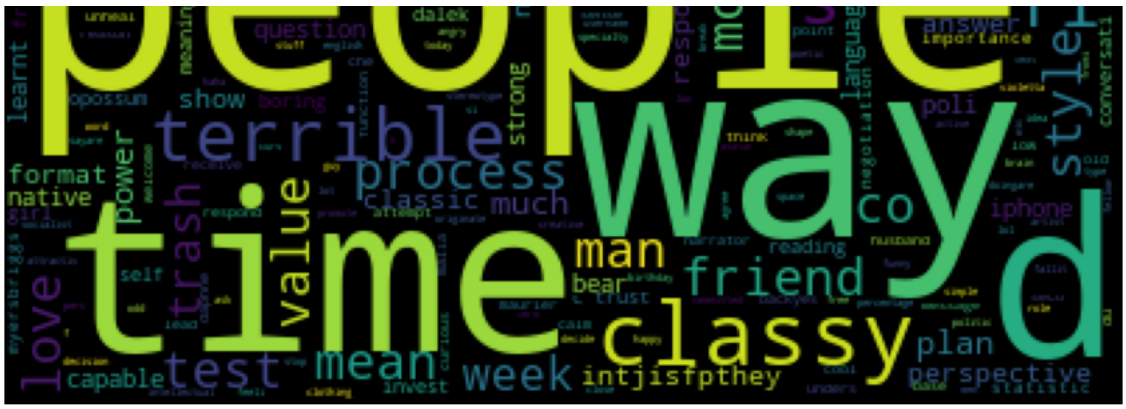
ESTP



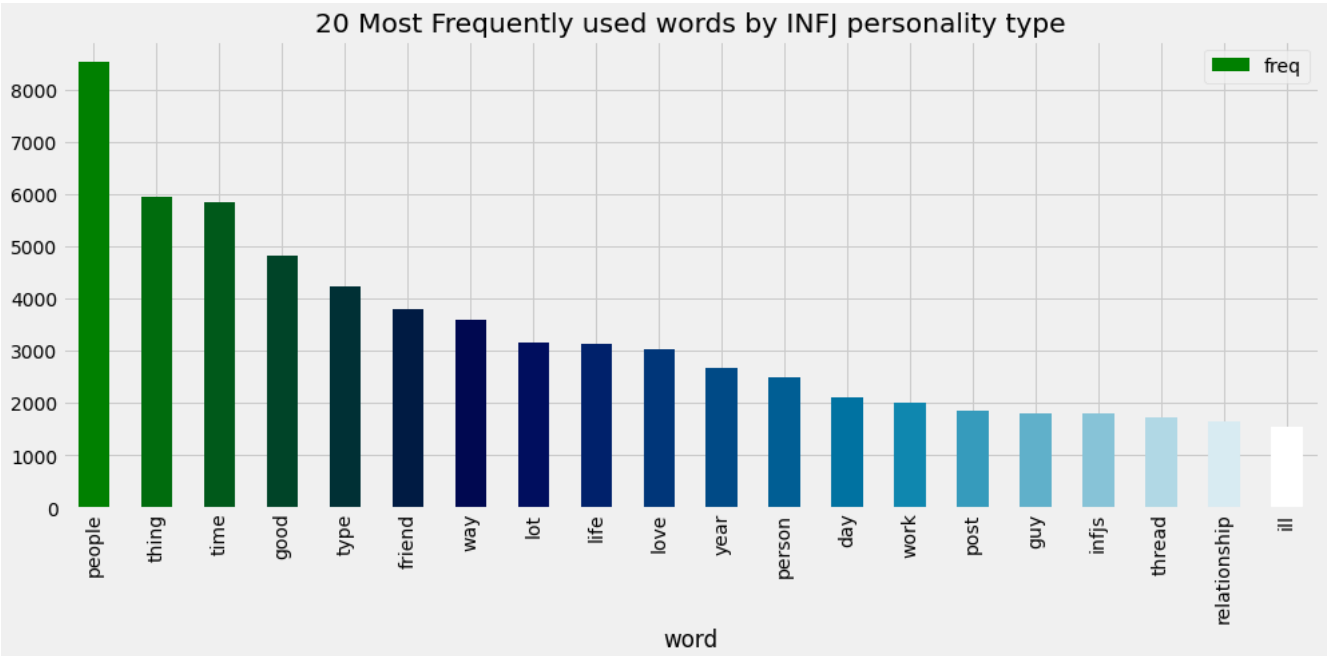
ESFP

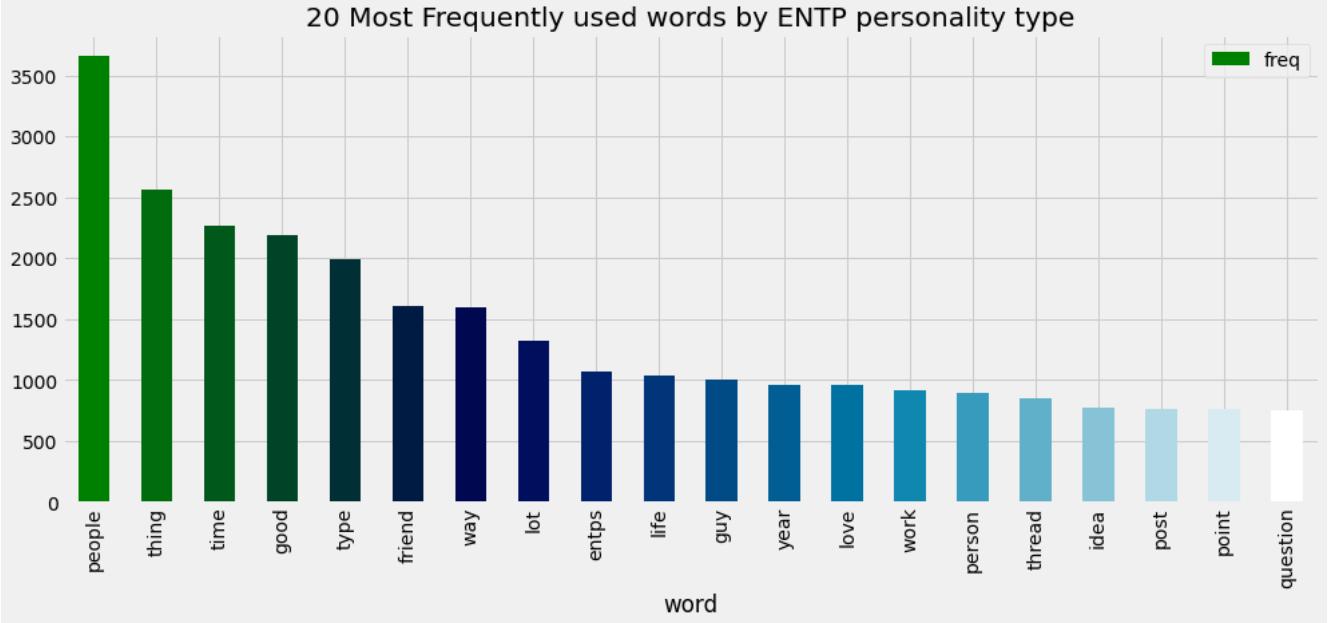


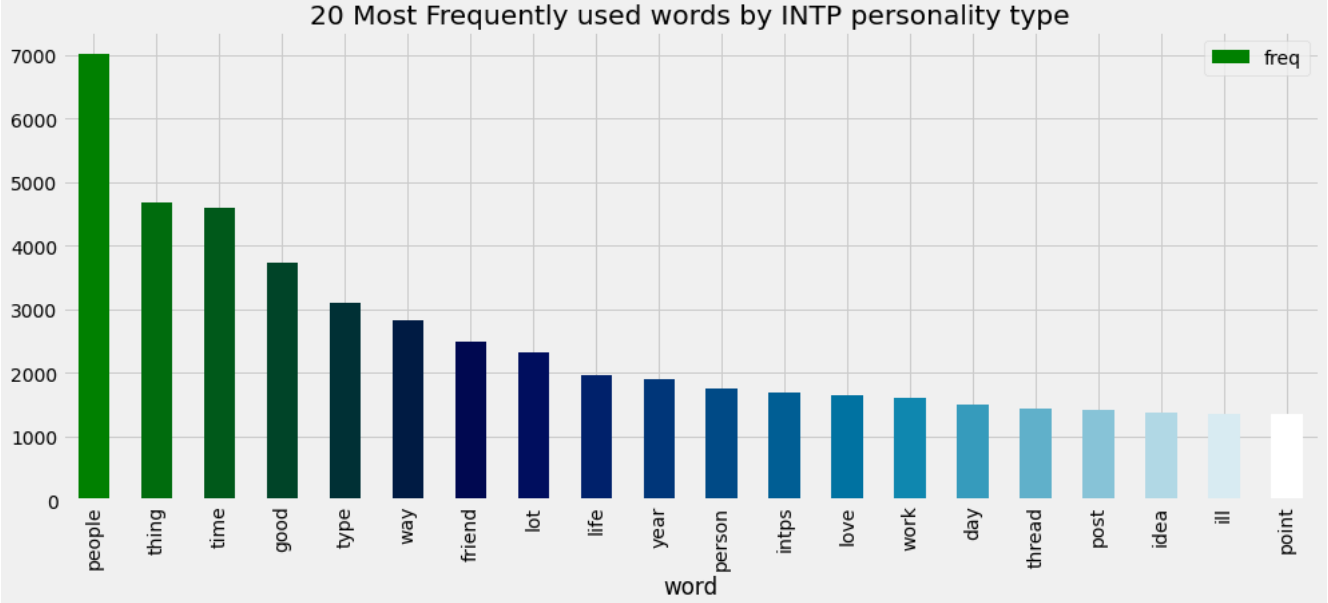
ESTJ

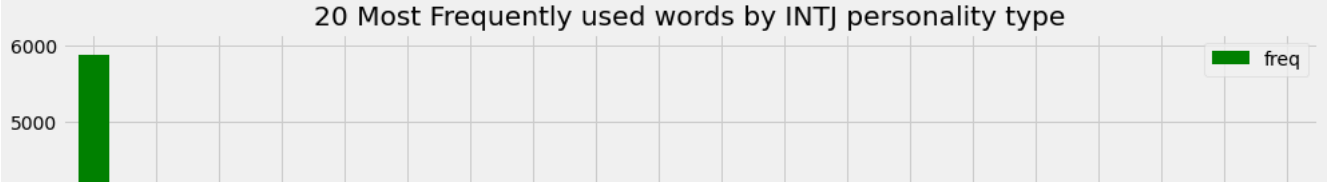


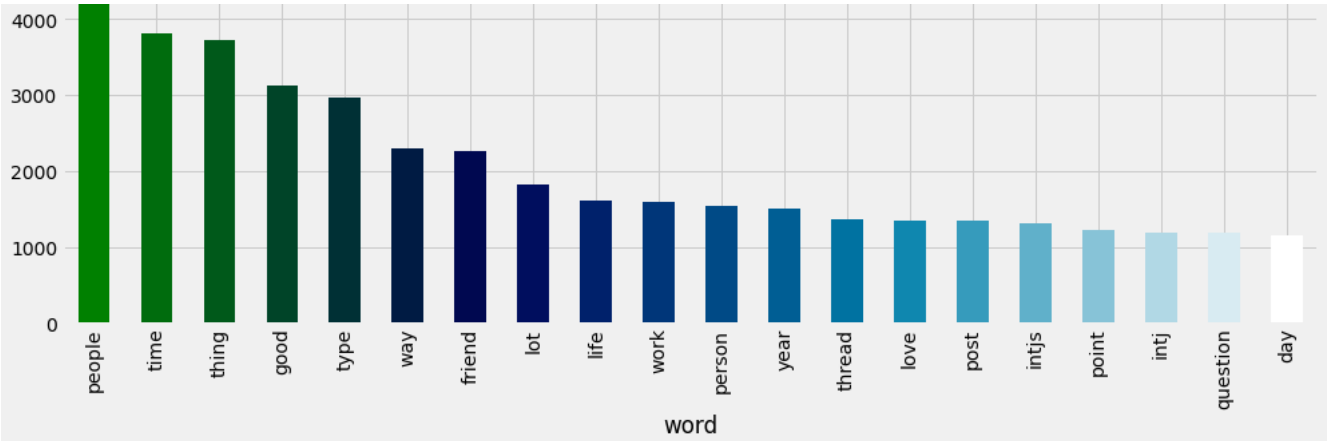

```
for i in data['type'].unique():
    cv = CountVectorizer(stop_words = 'english')
    words = cv.fit_transform(data[data['type'] == i].posts)
    sum_words = words.sum(axis=0)


    words_freq = [(word, sum_words[0, idx]) for word, idx in cv.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
    frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

    plt.style.use('fivethirtyeight')
    color = plt.cm.ocean(np.linspace(0, 1, 20))
    frequency.head(20).plot(x='word', y='freq', kind='bar', figsize=(15, 6), color = color)
    plt.title("20 Most Frequently used words by {} personality type".format(i))
    plt.show()
```


20 Most Frequently used words by INFJ personality type
freq
8000
7000
6000
5000
4000
3000
2000
1000
0
people
thing
time
good
type
friend
way
lot
life
love
year
person
day
work
post
guy
infjs
thread
relationship
ill
word

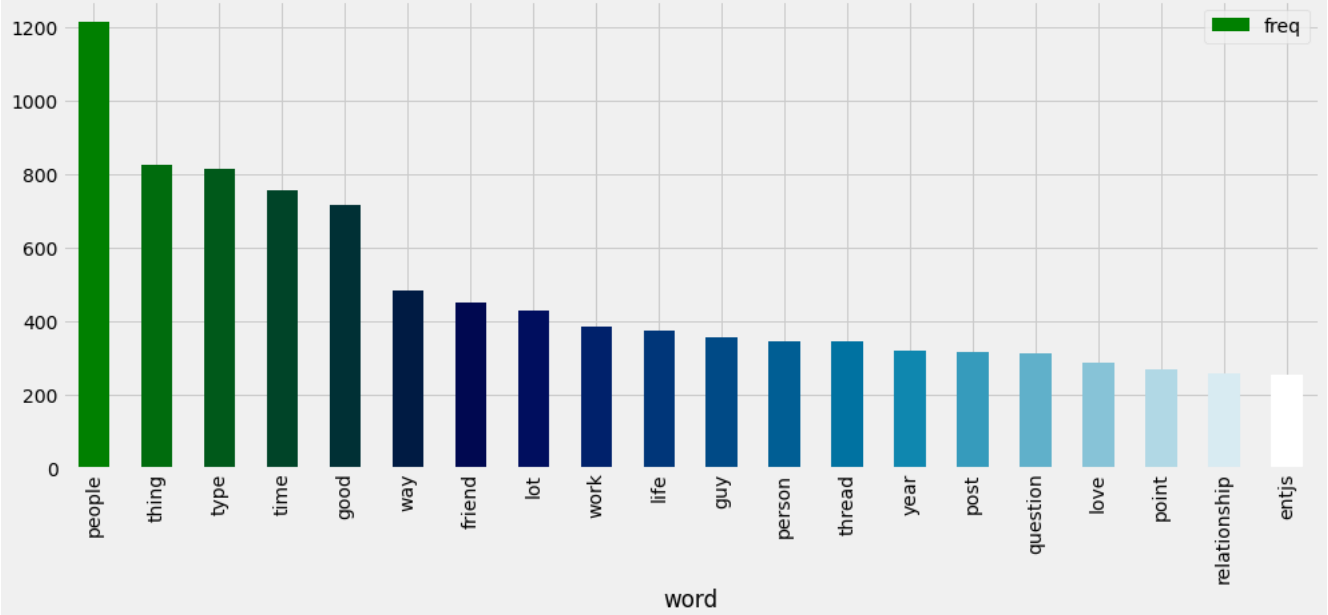

20 Most Frequently used words by ENTP personality type
freq
3500
3000
2500
2000
1500
1000
500
0
people
thing
time
good
type
friend
way
lot
entps
life
guy
year
love
work
person
thread
idea
post
point
question
word


20 Most Frequently used words by INTP personality type
freq
7000
6000
5000
4000
3000
2000
1000
0
people
thing
time
good
type
way
friend
lot
life
year
person
intps
love
work
day
thread
post
idea
ill
point
word

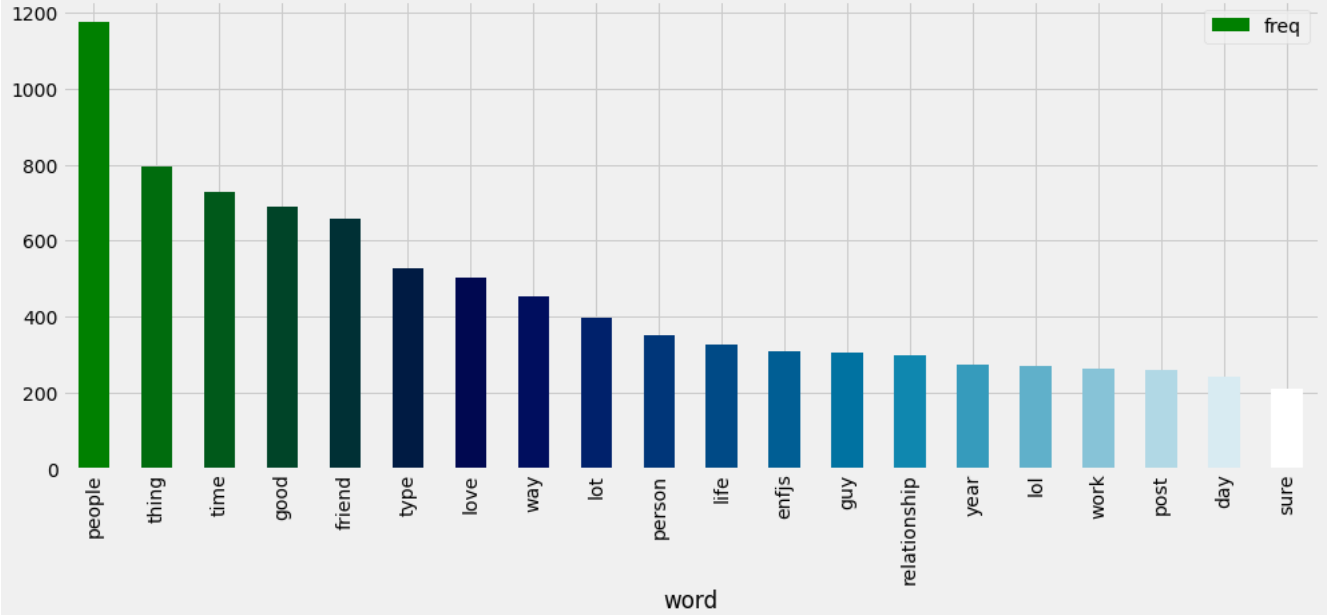

20 Most Frequently used words by INTJ personality type
freq
6000
5000



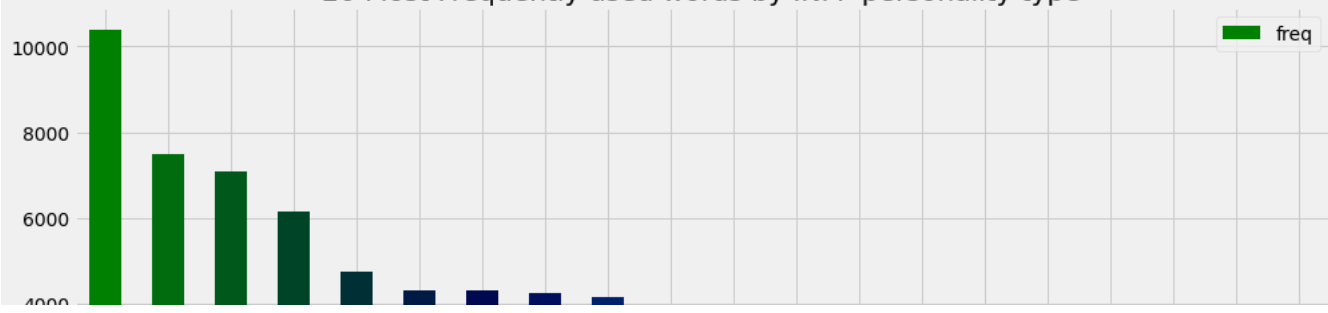
20 Most Frequently used words by ENTJ personality type

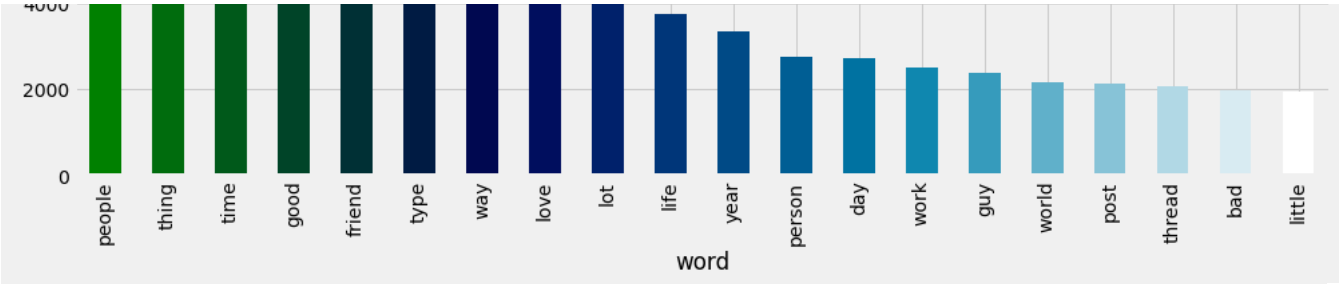


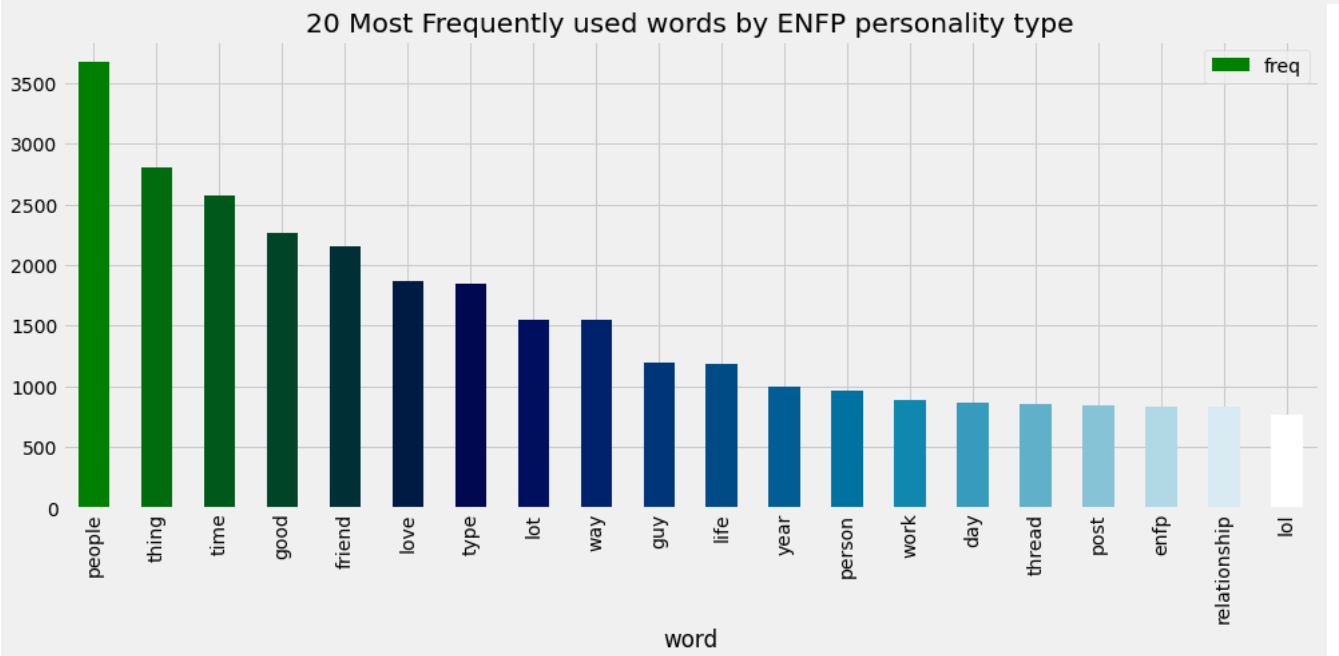
20 Most Frequently used words by ENFJ personality type

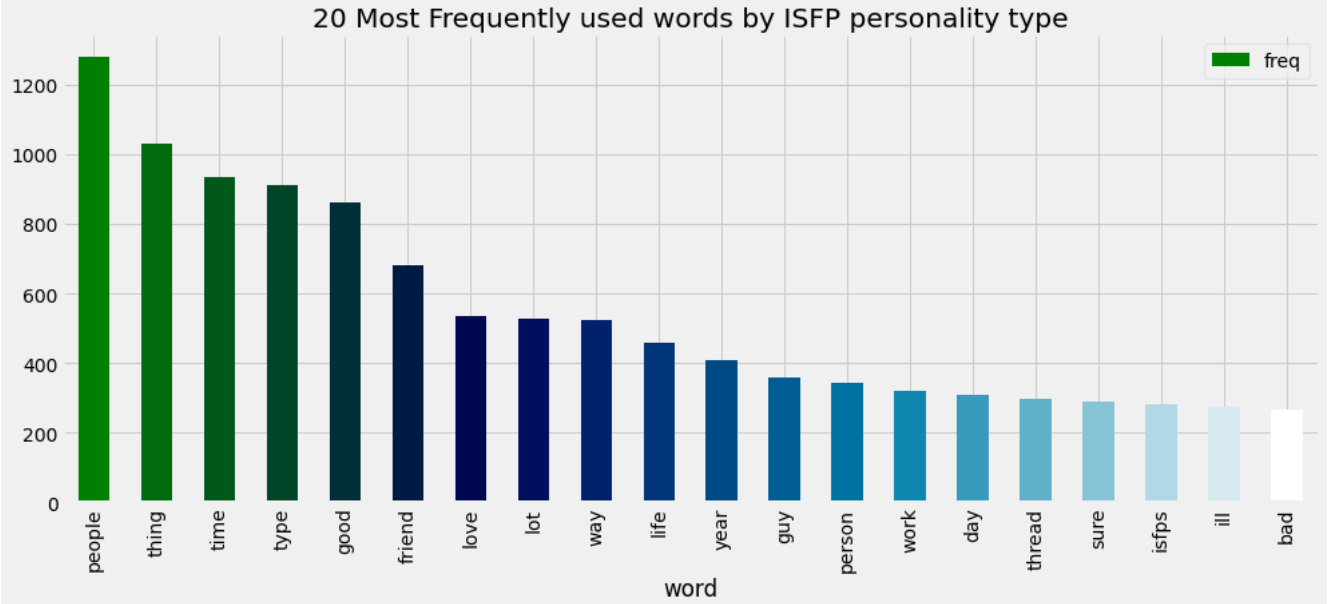


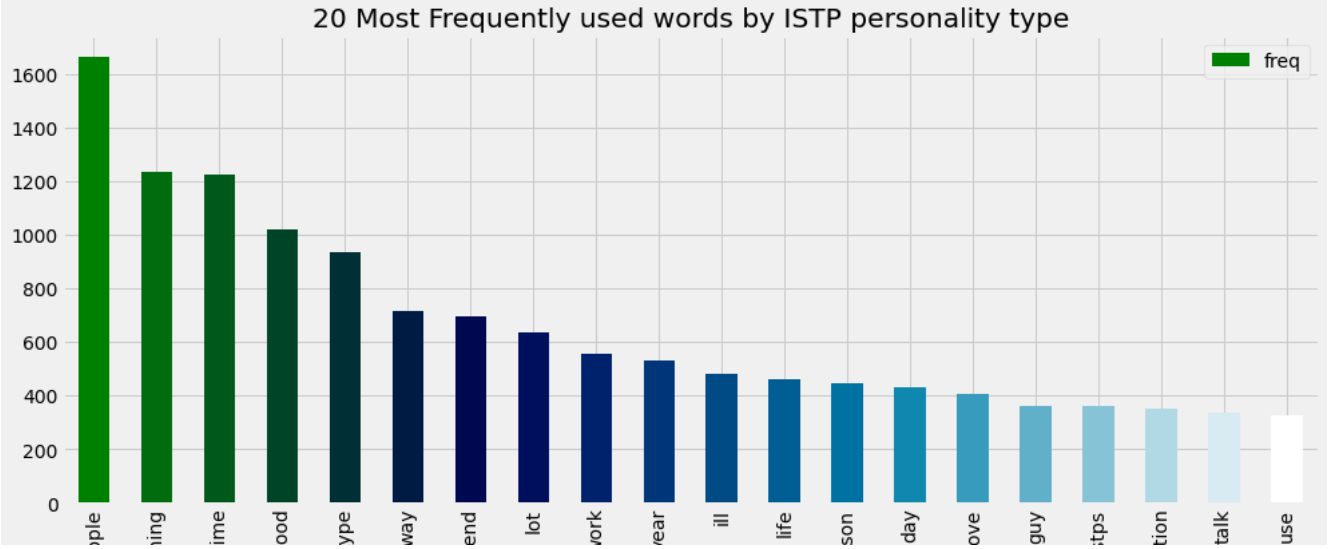
20 Most Frequently used words by INFP personality type

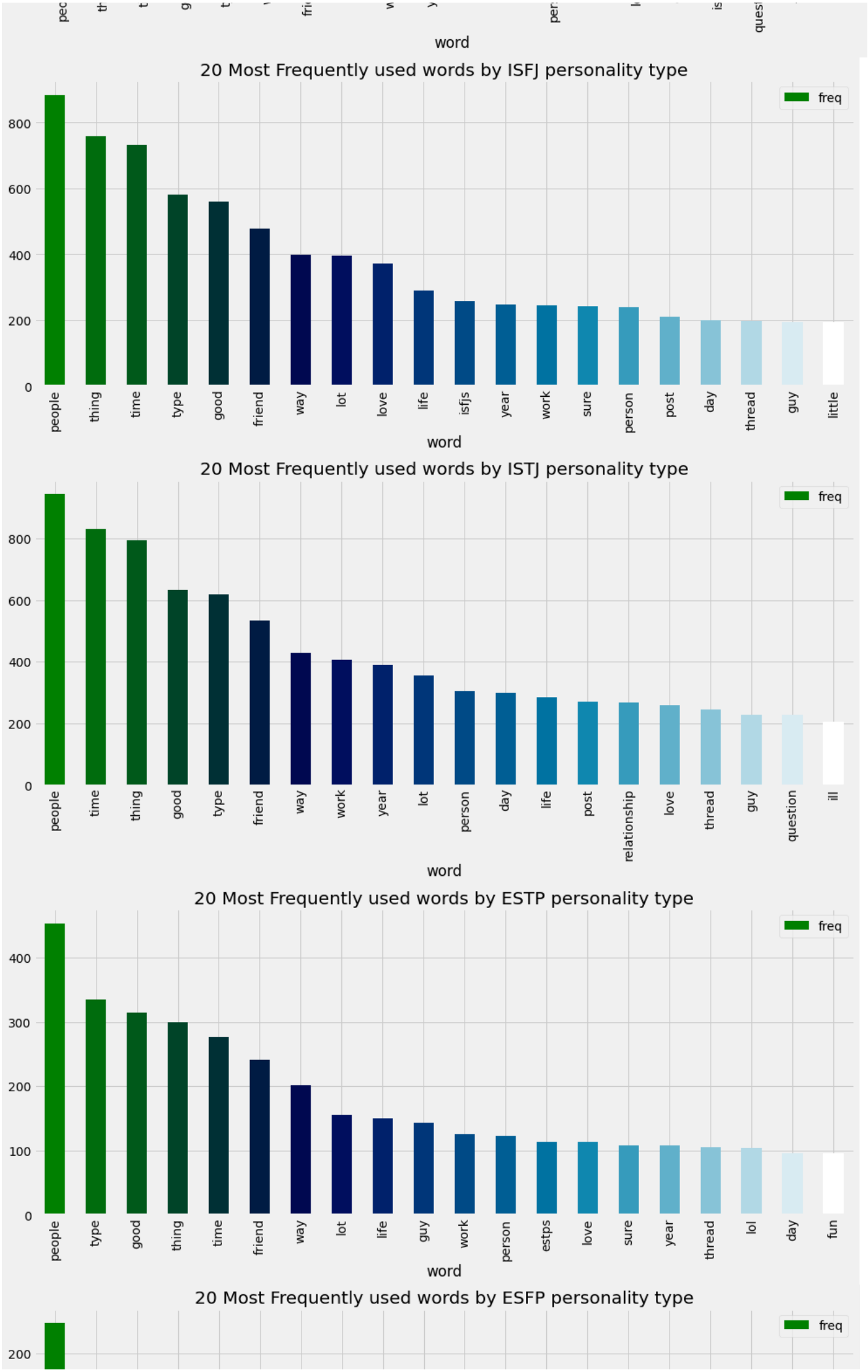



2000
0
people
thing
time
good
friend
type
way
love
lot
life
year
person
day
work
guy
world
post
thread
bad
little
word


20 Most Frequently used words by ENFP personality type
freq
3500
3000
2500
2000
1500
1000
500
0
people
thing
time
good
friend
love
type
lot
way
guy
life
year
person
work
day
thread
post
enfp
relationship
lol
word


20 Most Frequently used words by ISFP personality type
freq
1200
1000
800
600
400
200
0
people
thing
time
type
good
friend
love
lot
way
life
year
guy
person
work
day
thread
sure
isfps
ill
bad
word


20 Most Frequently used words by ISTP personality type
freq
1600
1400
1200
1000
800
600
400
200
0


word
20 Most Frequently used words by ISFJ personality type
freq
0
200
400
600
800
people
thing
time
type
good
friend
way
lot
love
life
isfjs
year
work
sure
person
post
day
thread
guy
little
word
20 Most Frequently used words by ISTJ personality type
freq
0
200
400
600
800
people
time
thing
good
type
friend
way
work
year
lot
person
day
life
post
relationship
love
thread
guy
question
ill
word
20 Most Frequently used words by ESTP personality type
freq
0
100
200
300
400
people
type
good
thing
time
friend
way
lot
life
guy
work
person
estps
love
sure
year
thread
lol
day
fun
word
20 Most Frequently used words by ESFP personality type
freq
200