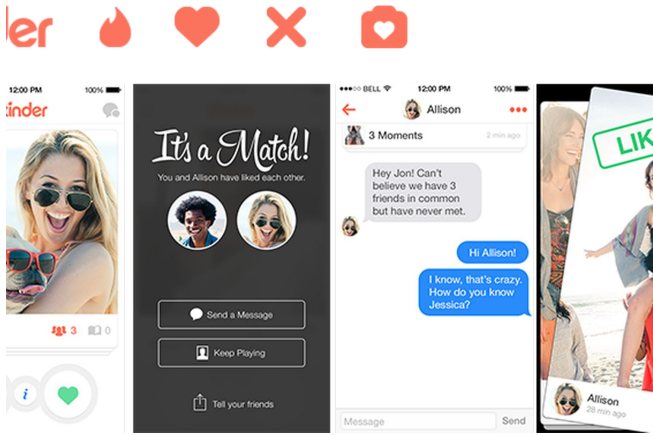


Project 3

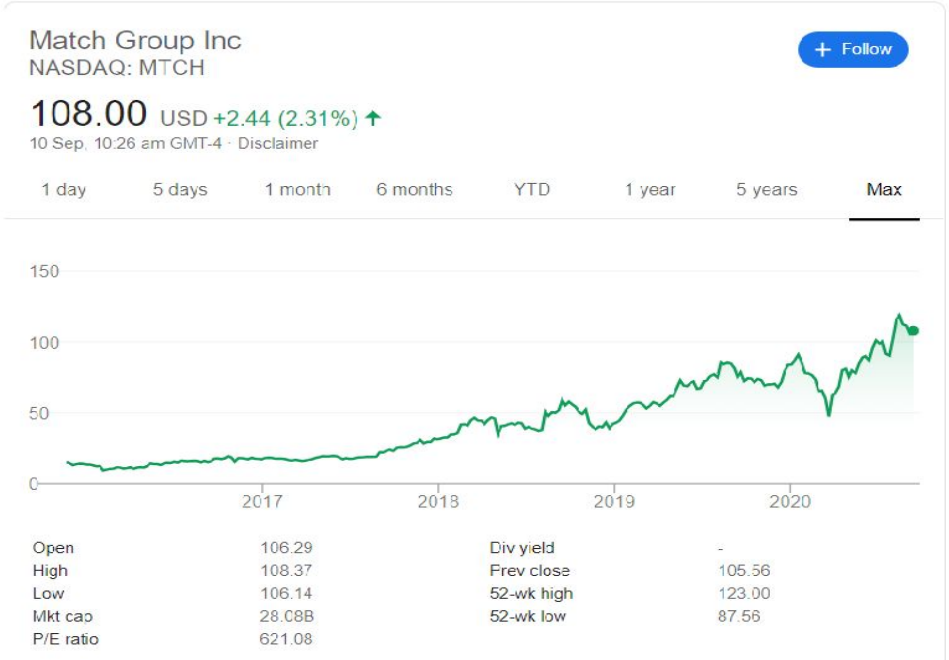
Build a classifier model to classify words that belong to either long term dating subreddits or short term subreddits. OKC vs tinder.

What are the differences in OKcupid and tinder subreddits?

Okcupid was launched 19th jan 2004 a dating site and used more for serious dating.

Tinder launched and went live Oct 2016 as a dating app but was used mainly for short term dating and hookups.

Match.com is a 25 billion dollar market cap company with 2.5billion that owns okcupid , tinder , plenty of fish,hinge , match.com and many other apps.

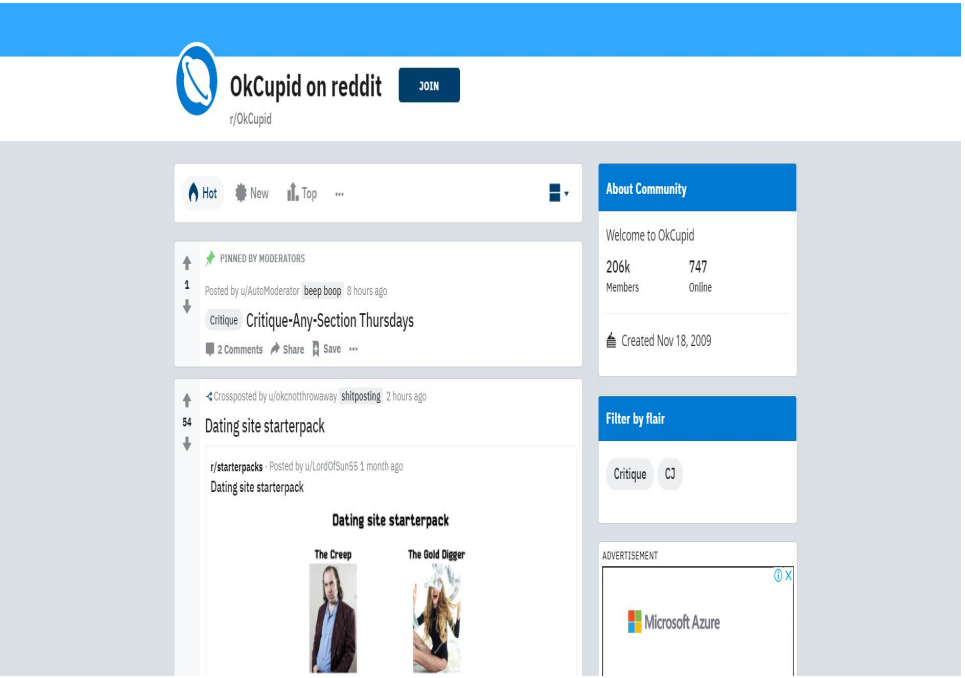


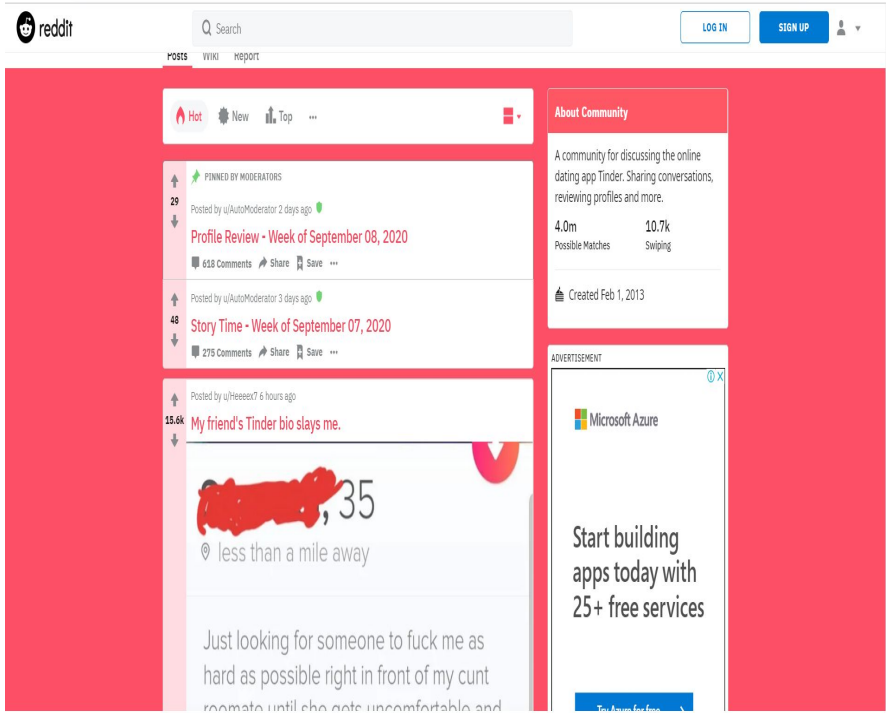
What would we like to achieve?

The stake holder is a new age dating app startup looking to build analytics into their users as they would like to **match serious daters** with people who are **looking for long term commitment** and match singles who are looking for fun with each other from the words they use. This is a billion dollar startup if it succeeds in solving the problem for matching people for love.

Coming in as a consultant , i advise them based on the models used to classify words in a post or they use in text based on NLP and logistic regression models that classify them to accurately belong to more of a serious long term dating kind of person like the profiles found on OKCUPID, or a casual dating /hookup kind of person TINDER based on the subreddits they come from.

The two subreddits Okcupid with 250k members and Tinder with 4million





The 5 main processes

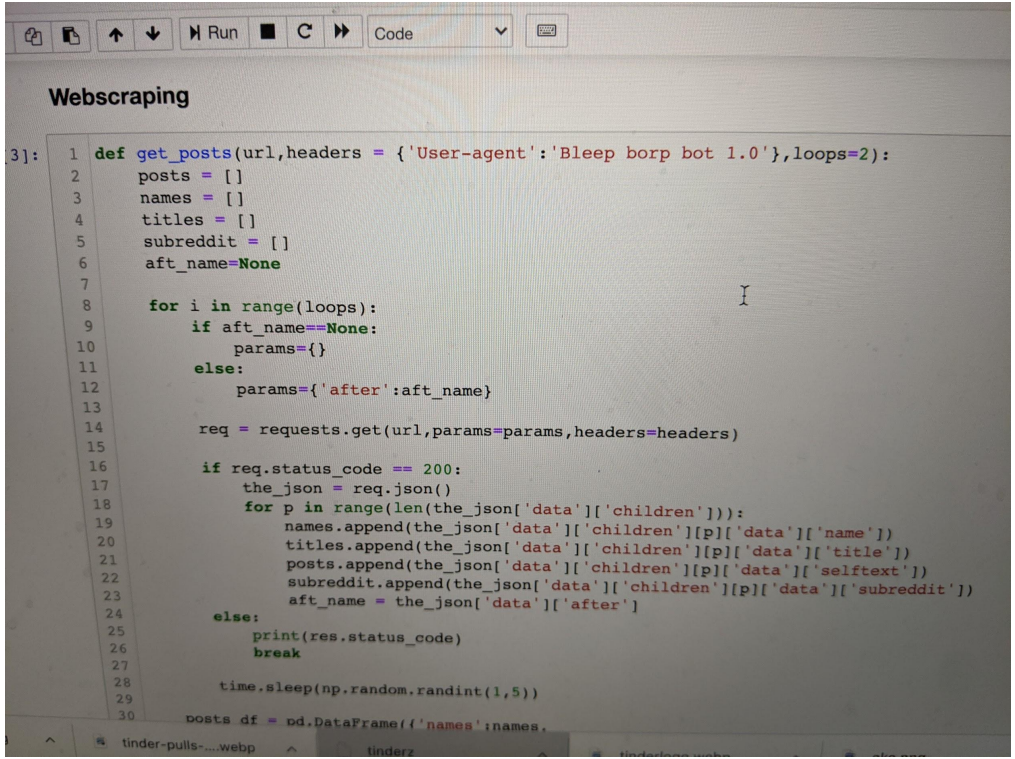
1) Webscraping
2) EDA (WordCloud and top 40 words) ,reading posts
3) Cleaning Data / lemmitization/stemming/removing stopwords/ common words with high frequency in both subreddits
4) Modelling and Classification

    a) Logistic Regression with count Vectorizer

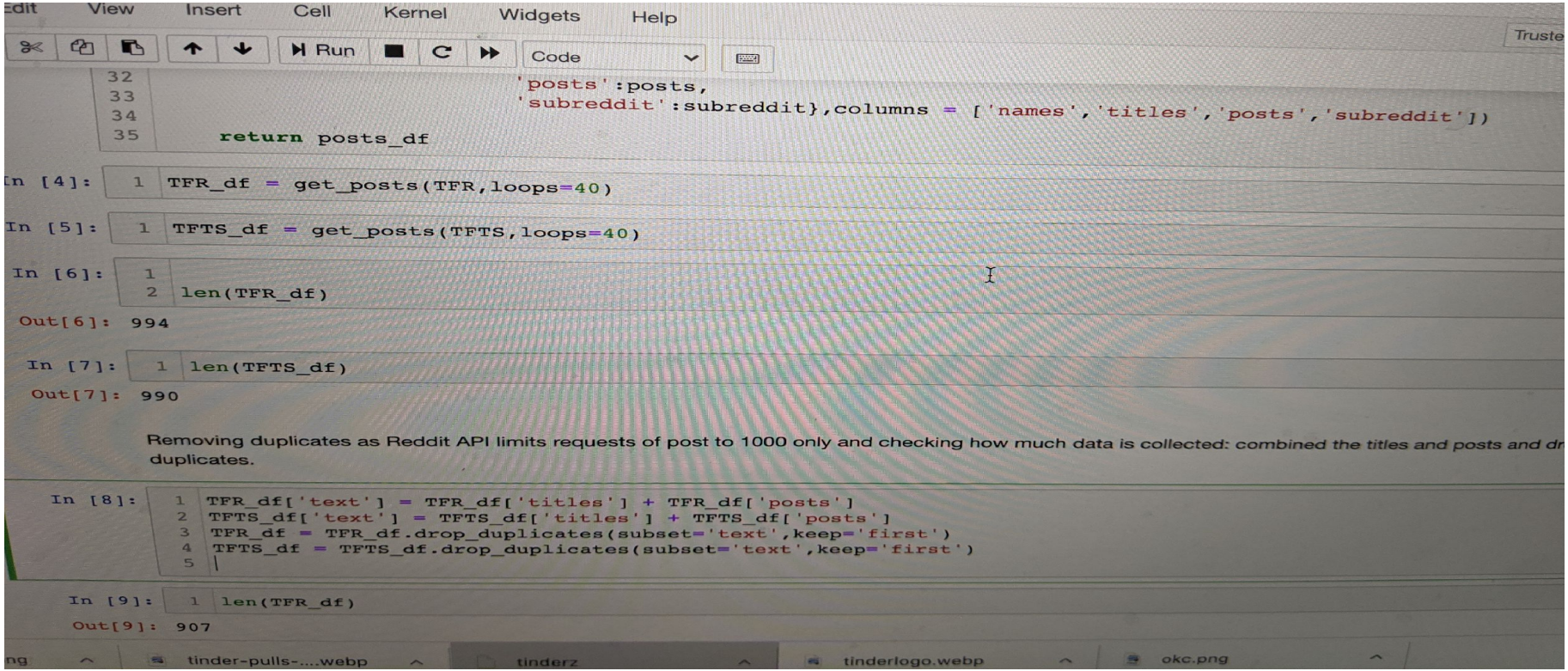
    b)Multinomial Naive Bayes with Count Vectorizer

5) Confusion matrix , model selection , accuracy and AUC/ROC
6) Conclusions and recommendations

1)Webscraping



1)WebScraping

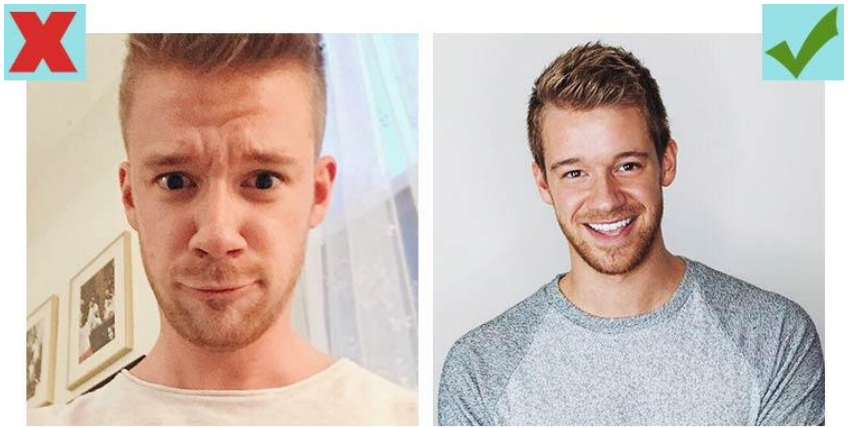


1)After Webscraping

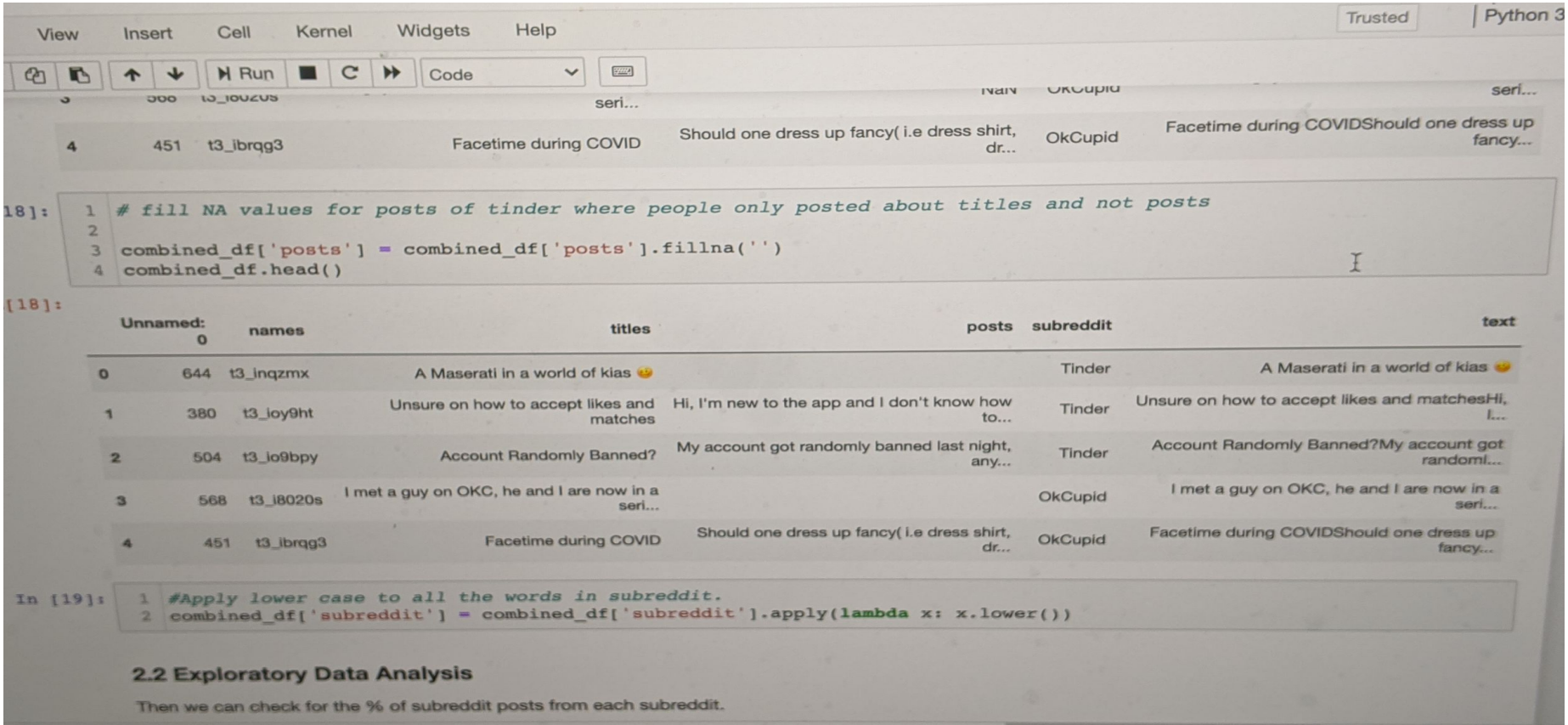
I had 994 posts for okcupid , and 990 posts for tinder.

After removing duplicates , had 907 Okc posts , 733 tinder posts and combined it in a dataframe.

55% of my total posts are from okcupid and 44% are from tinder.



What the posts look like



View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

3	568	t3_i8020s	I met a guy on OKC, he and I are now in a seri...		NaN	seri...
4	451	t3_ibrqg3	Facetime during COVID	Should one dress up fancy(i.e dress shirt, dr...	OkCupid	Facetime during COVIDShould one dress up fancy...

In [18]:

```
# fill NA values for posts of tinder where people only posted about titles and not posts

combined_df['posts'] = combined_df['posts'].fillna('')
combined_df.head()
```

Out[18]:

	Unnamed: 0	names	titles	posts	subreddit	text
0	644	t3_inqzmx	A Maserati in a world of kias 😀		Tinder	A Maserati in a world of kias 😀
1	380	t3_ioy9ht	Unsure on how to accept likes and matches	Hi, I'm new to the app and I don't know how to...	Tinder	Unsure on how to accept likes and matchesHi, I...
2	504	t3_io9bpy	Account Randomly Banned?	My account got randomly banned last night, any...	Tinder	Account Randomly Banned?My account got randoml...
3	568	t3_i8020s	I met a guy on OKC, he and I are now in a seri...		OkCupid	I met a guy on OKC, he and I are now in a seri...
4	451	t3_ibrqg3	Facetime during COVID	Should one dress up fancy(i.e dress shirt, dr...	OkCupid	Facetime during COVIDShould one dress up fancy...

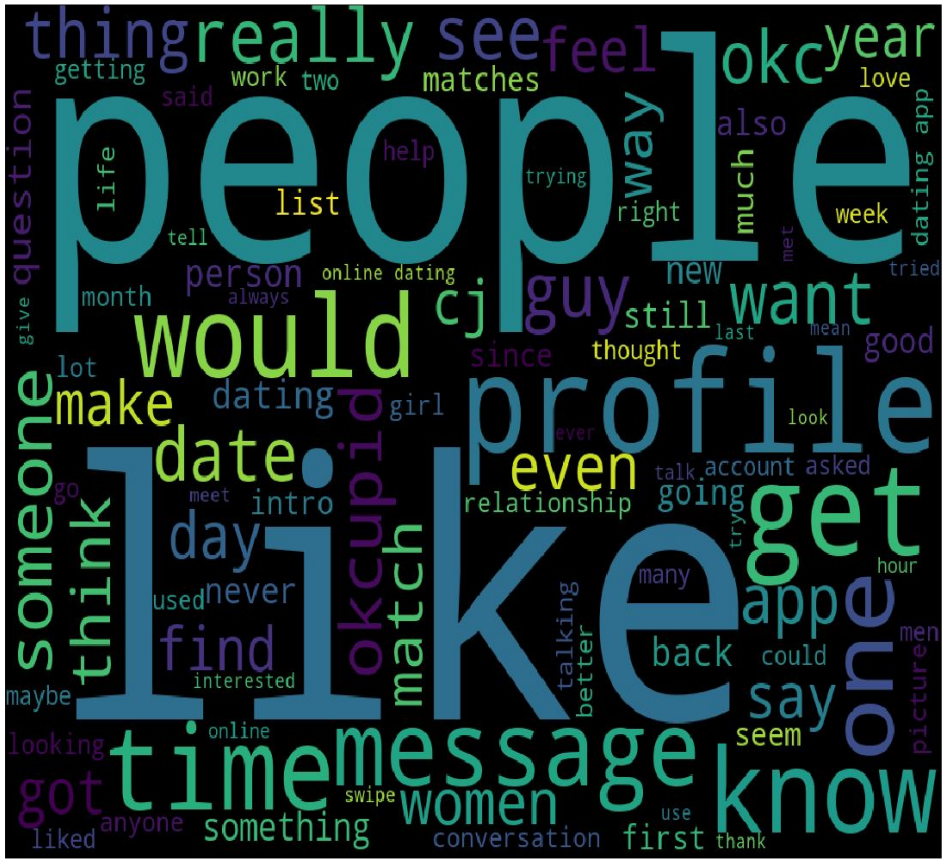
In [19]:

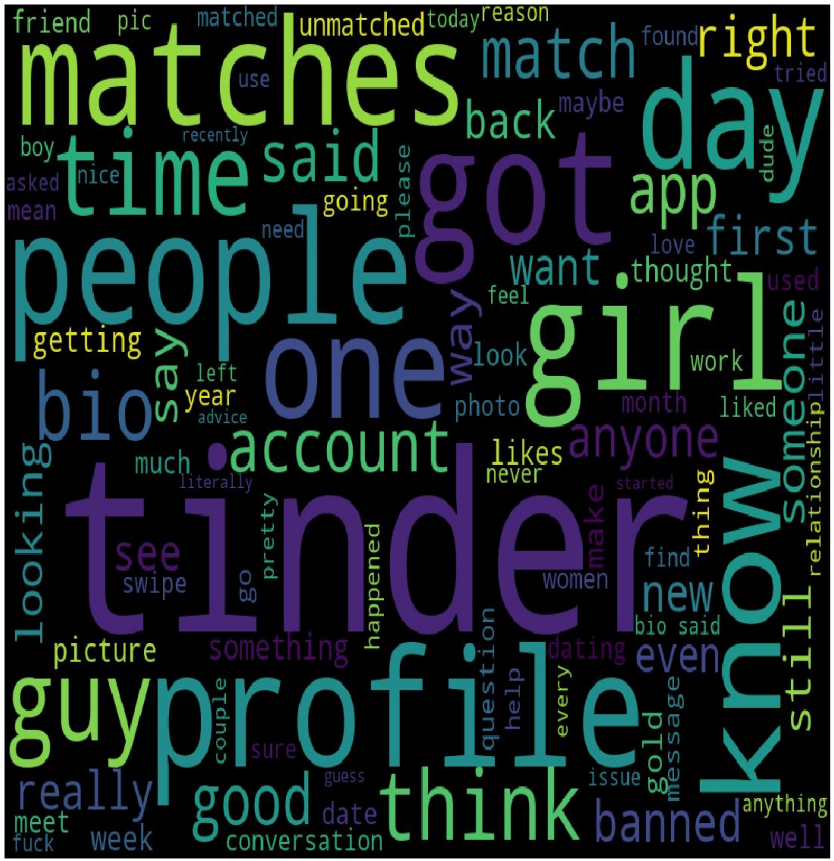
```
#Apply lower case to all the words in subreddit.
combined_df['subreddit'] = combined_df['subreddit'].apply(lambda x: x.lower())
```

2.2 Exploratory Data Analysis

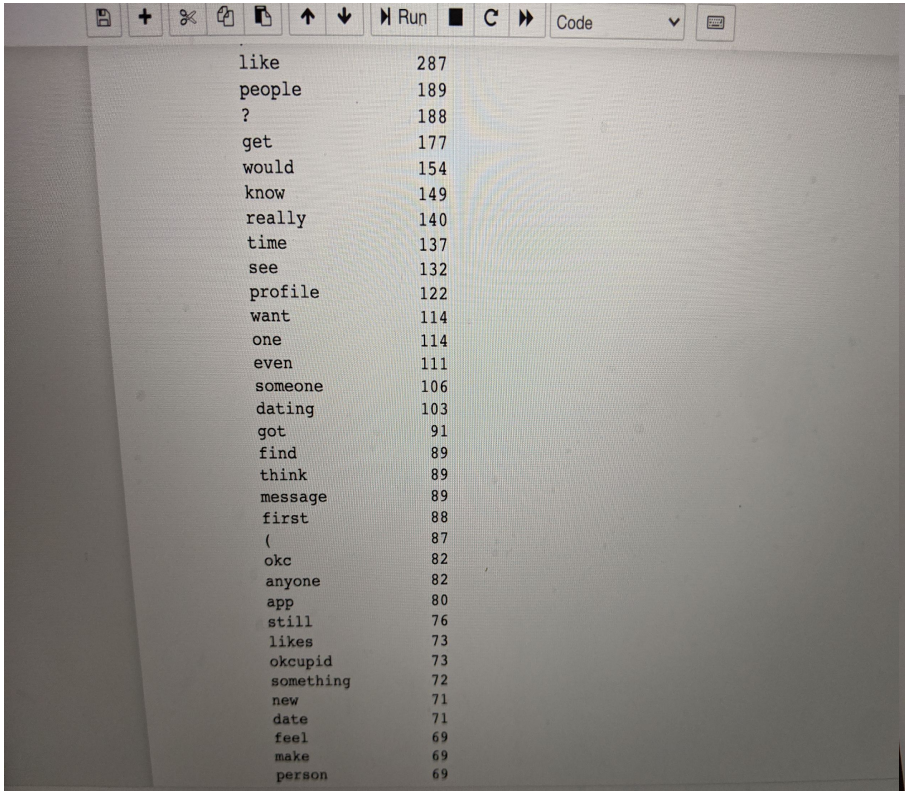
Then we can check for the % of subreddit posts from each subreddit.

2) EDA (Wordcloud OKC and tinder on right

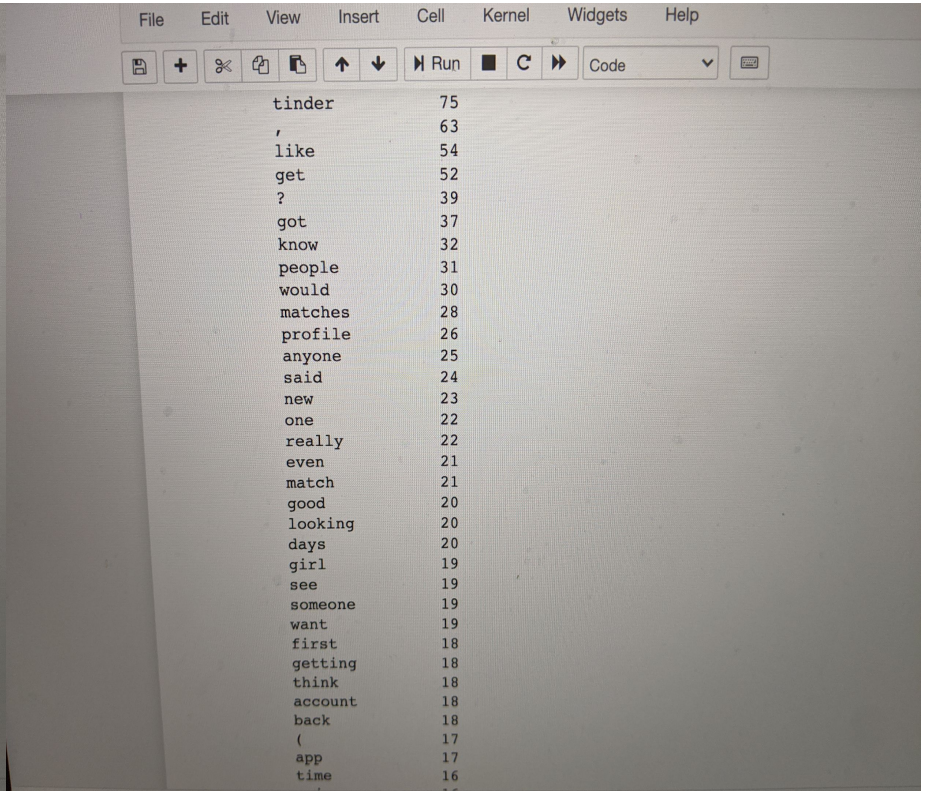




Top 40 words okcupid and tinder



like	287
people	189
?	188
get	177
would	154
know	149
really	140
time	137
see	132
profile	122
want	114
one	114
even	111
someone	106
dating	103
got	91
find	89
think	89
message	89
first	88
(	87
okc	82
anyone	82
app	80
still	76
likes	73
okcupid	73
something	72
new	71
date	71
feel	69
make	69
person	69



tinder	75
,	63
like	54
get	52
?	39
got	37
know	32
people	31
would	30
matches	28
profile	26
anyone	25
said	24
new	23
one	22
really	22
even	21
match	21
good	20
looking	20
days	20
girl	19
see	19
someone	19
want	19
first	18
getting	18
think	18
account	18
back	18
(	17
app	17
time	16

Top words that appeared in both Okcupid subreddit and tinder

1)Like

2)people

3)get

4)anyone

5)said

6)profile

7)someone

8)one

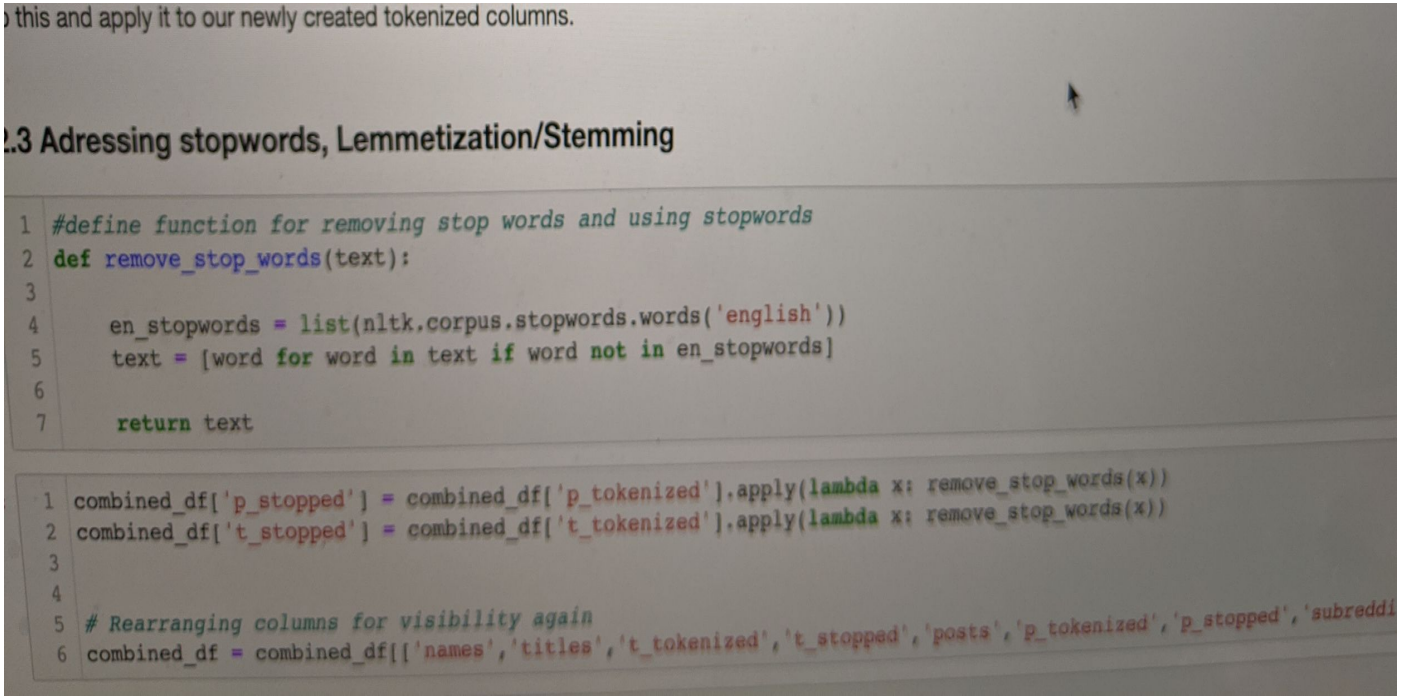
9)really

I then added these to a list of stopwords to remove the words from the combined dataframe as these are high frequency words that appear in both and would not affect our predictions.

Step3) cleaning

1) Lemmatized words
2) Tokenized words,
3) Remove stop words
4) Remove common words in both subreddits with high frequency top 40 words
5) Lower case words
6) Replaced Null values with “”
7) Assigned 1 to Okcupid and 0 to Tinder.
8) Combined titles and posts to a combined column

Cleaning stopwords



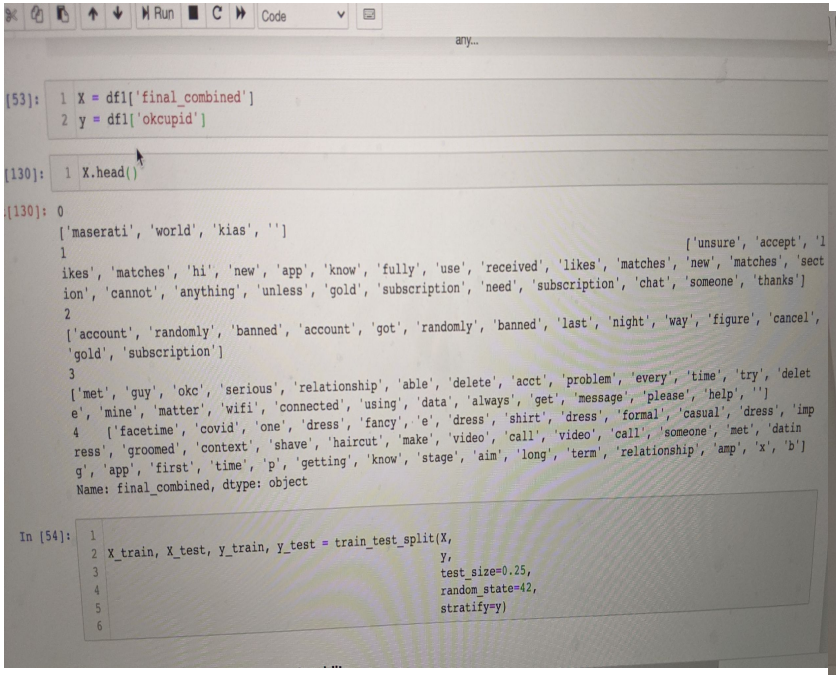
Cleaning (using regex to remove html tags etc)

In [23]:

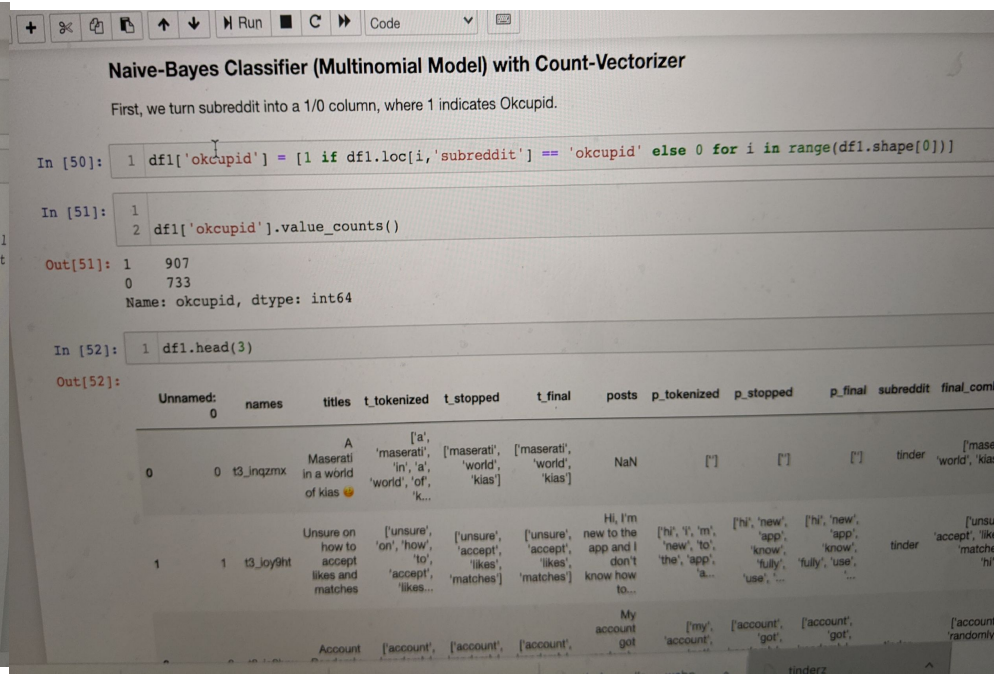
```
#define function to clean text with regex and split words

def string_clean(text):

    # Remove all HTML tags
    text = re.sub(r'<.*?>', '', text)

    # Remove non-letters.
    text = re.sub("[^a-zA-Z]", " ", text)
    # Remove all URL tags
    text = re.sub(r'^https?:\/\/.*[\r\n]*','',text)

    # Keep text without punctuation
    text = re.sub(r'[^\w\s]','',text)

    # convert text to lowercase
    text = text.strip().lower()

    # split text into a list of words
    token_text = re.split('\W+',text)

    return token_text
```

Then, we can go ahead and apply this function to both our posts and titles to return us the tokens of each corresponding column.

Countvectorize and train test split



```
[53]: 1 X = df1['final_combined']
      2 y = df1['okcupid']

[130]: 1 X.head()

[130]: 0
['maserati', 'world', 'kias', '']
1
['unsure', 'accept', 'l
ikes', 'matches', 'hi', 'new', 'app', 'know', 'fully', 'use', 'received', 'likes', 'matches', 'new', 'matches', 'sect
ion', 'cannot', 'anything', 'unless', 'gold', 'subscription', 'need', 'subscription', 'chat', 'someone', 'thanks']
2
['account', 'randomly', 'banned', 'account', 'got', 'randomly', 'banned', 'last', 'night', 'way', 'figure', 'cancel',
'gold', 'subscription']
3
['met', 'guy', 'okc', 'serious', 'relationship', 'able', 'delete', 'acct', 'problem', 'every', 'time', 'try', 'delet
e', 'mine', 'matter', 'wifi', 'connected', 'using', 'data', 'always', 'get', 'message', 'please', 'help', '']
4    ['facetime', 'covid', 'one', 'dress', 'fancy', 'e', 'dress', 'shirt', 'dress', 'formal', 'casual', 'dress', 'imp
ress', 'groomed', 'context', 'shave', 'haircut', 'make', 'video', 'call', 'video', 'call', 'someone', 'met', 'datin
g', 'app', 'first', 'time', 'p', 'getting', 'know', 'stage', 'aim', 'long', 'term', 'relationship', 'amp', 'x', 'b']
Name: final_combined, dtype: object

In [54]:
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=42,
                                                    stratify=y)
```



Naive-Bayes Classifier (Multinomial Model) with Count-Vectorizer

First, we turn subreddit into a 1/0 column, where 1 indicates Okcupid.

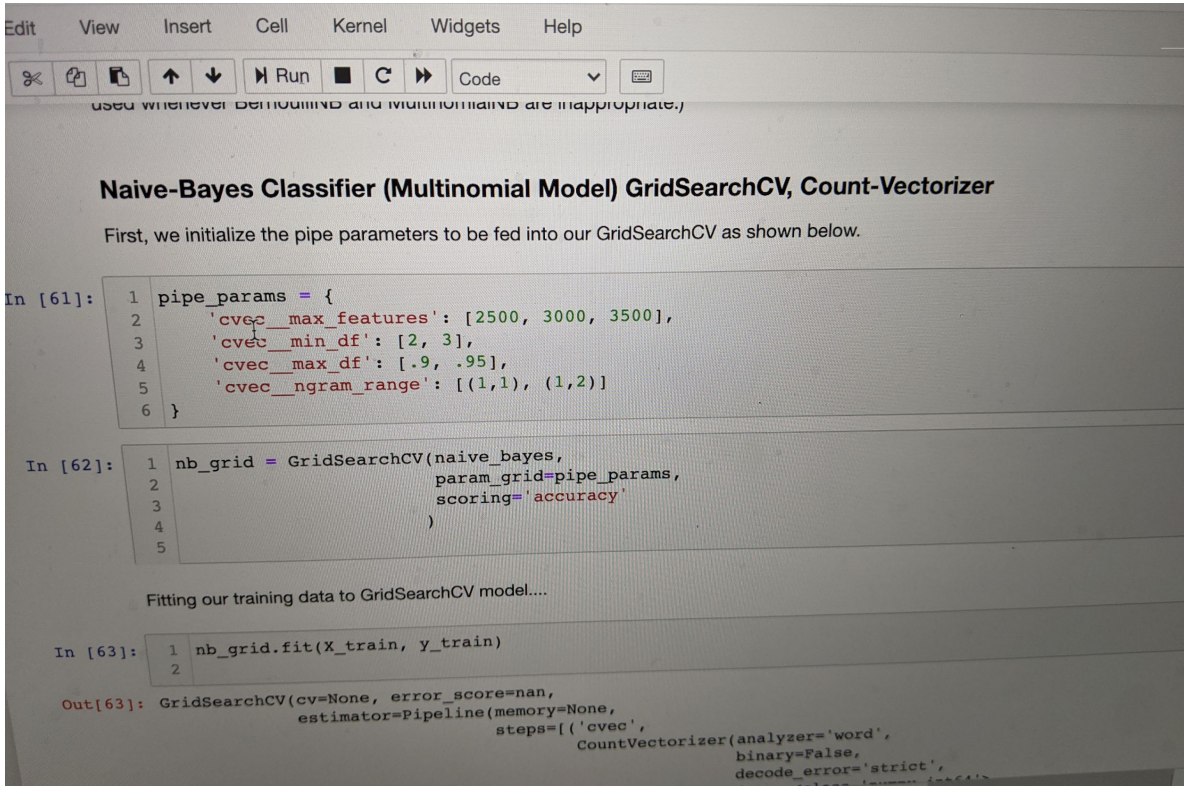
```
In [50]: 1 df1['okcupid'] = [1 if df1.loc[i,'subreddit'] == 'okcupid' else 0 for i in range(df1.shape[0])]

In [51]: 1
         2 df1['okcupid'].value_counts()

Out[51]: 1    907
         0    733
         Name: okcupid, dtype: int64

In [52]: 1 df1.head(3)

Out[52]:
```

	Unnamed: 0	names	titles	t_tokenized	t_stopped	t_final	posts	p_tokenized	p_stopped	p_final	subreddit	final_comb
0	0	t3_inqzmx	A Maserati in a world of kias	['a', 'maserati', 'in', 'a', 'world', 'of', 'k...	['maserati', 'world', 'kias']	['maserati', 'world', 'kias']	NaN	['']	['']	['']	tinder	['maser 'world', 'kia
1	1	t3_joy9ht	Unsure on how to accept likes and matches	['unsure', 'on', 'how', 'to', 'accept', 'likes...	['unsure', 'accept', 'likes', 'matches']	['unsure', 'accept', 'likes', 'matches']	Hi, I'm new to the app and I don't know how to...	['hi', 'i', 'm', 'new', 'to', 'the', 'app', 'a...	['hi', 'new', 'app', 'know', 'fully', 'use', '...	['hi', 'new', 'app', 'know', 'fully', 'use', '...	tinder	['unsu 'accept', 'like 'matche 'hi
			Account	['account',	['account',	['account',	My account got	['my', 'account',	['account', 'got',	['account', 'got',	tinderz	['accoun 'randomly

Modeling and Classification

1) Fit Countvectorized dataframe(sparse matrix) into naive bayes model pipeline
2) GridsearchCV for best parameters
3) Instantiate a new NaiveBayes model with best parameters
4) Did logistic regression with the count vectorized models.
5) Did logistic regression with TFIDvectorizer model.(Count vectorize followed by TFID transform)
6) Did Multinomial naive bayes model with CountVectorizer Model
7) Did Multinomial Naive bayes model with TFIDvectorizer model.

Grid Search for best Parameters



Logistic Regression with CountVectorizer model

interpret the results correspondingly below.

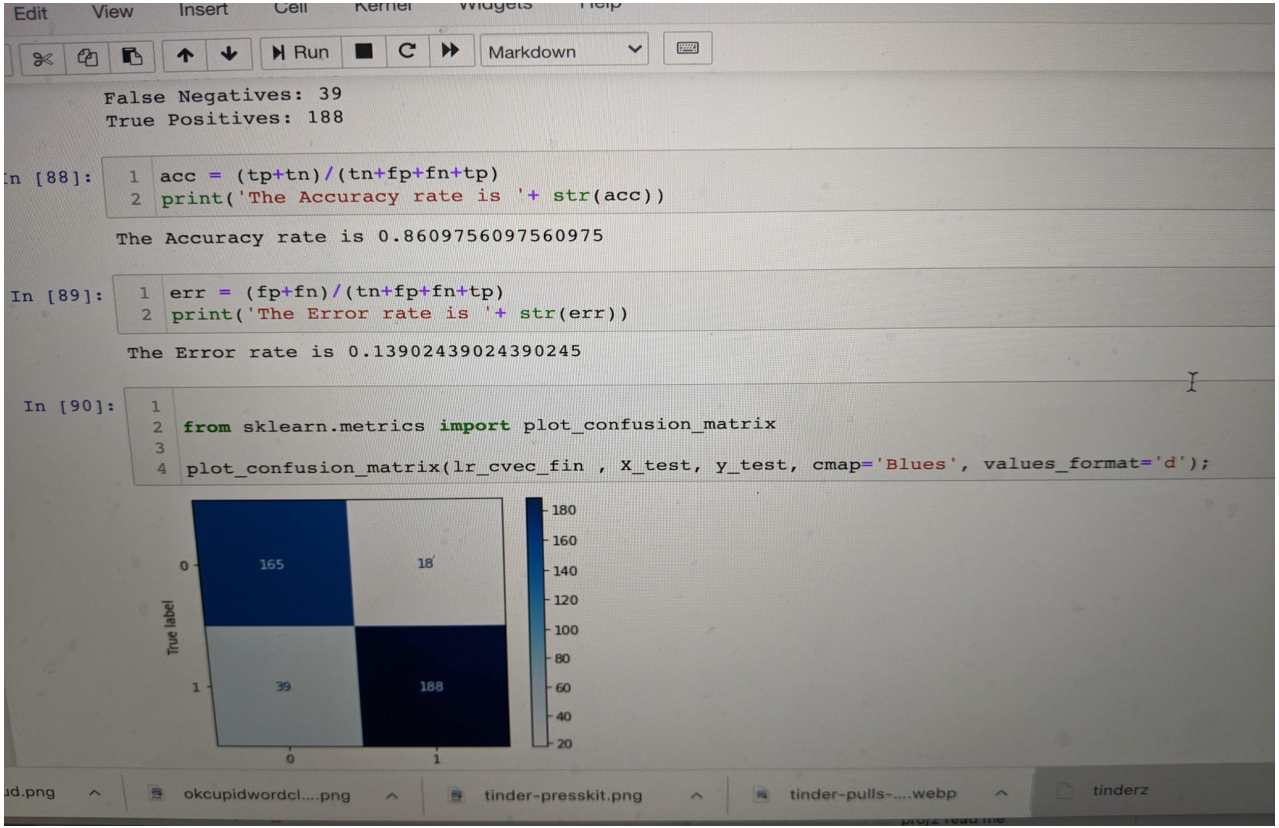
Final Logistic Regression with CountVectorizer model:

```
]:
1 lr_cvec_fin = Pipeline([('cvec',CountVectorizer(max_df = 0.9,
2                                                 max_features = 2500,
3                                                 min_df = 3,
4                                                 ngram_range = (1, 1))),
5                         ('lr', LogisticRegression())])
```

```
[86]:
1 lr_cvec_fin.fit(X_train,y_train)
2
3 lr_cvec_ypred_fin = lr_cvec_fin.predict(X_test)
4 lr_cvec_fin_acc = accuracy_score(y_test,lr_cvec_ypred_fin)
5
6 print(f'Accuracy: {lr_cvec_fin_acc}')
7 print(classification_report(y_test,lr_cvec_ypred_fin,target_names=['okcupid','tinder']))
```

```
Accuracy: 0.8609756097560975
              precision    recall  f1-score   support

     okcupid       0.81      0.90      0.85       183
      tinder       0.91      0.83      0.87       227

    accuracy                           0.86       410
   macro avg       0.86      0.86      0.86       410
weighted avg       0.87      0.86      0.86       410
```

Logistic Regression with CountVectorizer model



Logistic Regression with CountVectorizer model

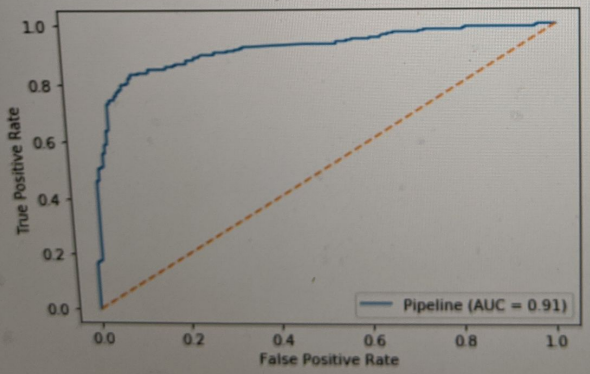
Edit View Insert Cell Kernel Widgets Help

Run Markdown

In [91]:

```
from sklearn.metrics import plot_roc_curve

plot_roc_curve(lr_cvec_fin, X_test, y_test);

plt.plot(np.linspace(0, 1, 200),
         np.linspace(0, 1, 200),
         label='baseline',
         linestyle='--')
```

Out[91]: [<matplotlib.lines.Line2D at 0x7fcdc2f8a6d0>]



Logistic Regression with CountVectorizer model

1)Accuracy of predicting posts from Okcupid to be 86%

2) AUC ROC high AUC of 0.91 , being able to separate Okcupid posts and tinder posts .

Logistic Regression with TFIDFVectorizer model

Final Logistic Regression with TfidfVectorizer model:

```
lr_tvec_fin = Pipeline([('tvec',TfidfVectorizer(max_df = 0.9,
                                                max_features = 2500,
                                                min_df = 3,
                                                ngram_range = (1, 1))),
                        ('lr', LogisticRegression())])
```

```
lr_cvec_ypred_fin = lr_cvec_fin.predict(X_train)
```

```
lr_tvec_fin.fit(X_train,y_train)

lr_tvec_ypred_fin = lr_tvec_fin.predict(X_test)
lr_tvec_fin_acc = accuracy_score(y_test,lr_tvec_ypred_fin)

print(f'Accuracy: {lr_tvec_fin_acc}')
print(classification_report(y_test,lr_tvec_ypred_fin,target_names=['okcupid','tinder'
```

```
Accuracy: 0.8365853658536585
              precision    recall  f1-score   support

     okcupid       0.83      0.80      0.81       183
```

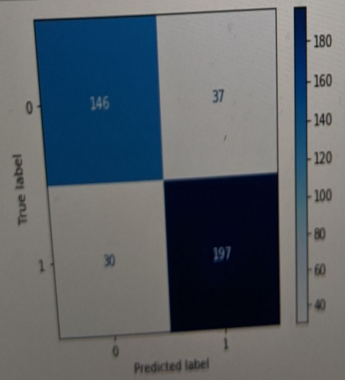
```
acc = (tp+tn)/(tn+fp+fn+tp)
print('The Accuracy rate is '+ str(acc))
```

```
The Accuracy rate is 0.8365853658536585
```

```
err = (fp+fn)/(tn+fp+fn+tp)
print('The Error rate is '+ str(err))
```

```
The Error rate is 0.16341463414634147
```

```

from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(lr_tvec_fin , X_test, y_test, cmap='Blues', values_format='d');
```

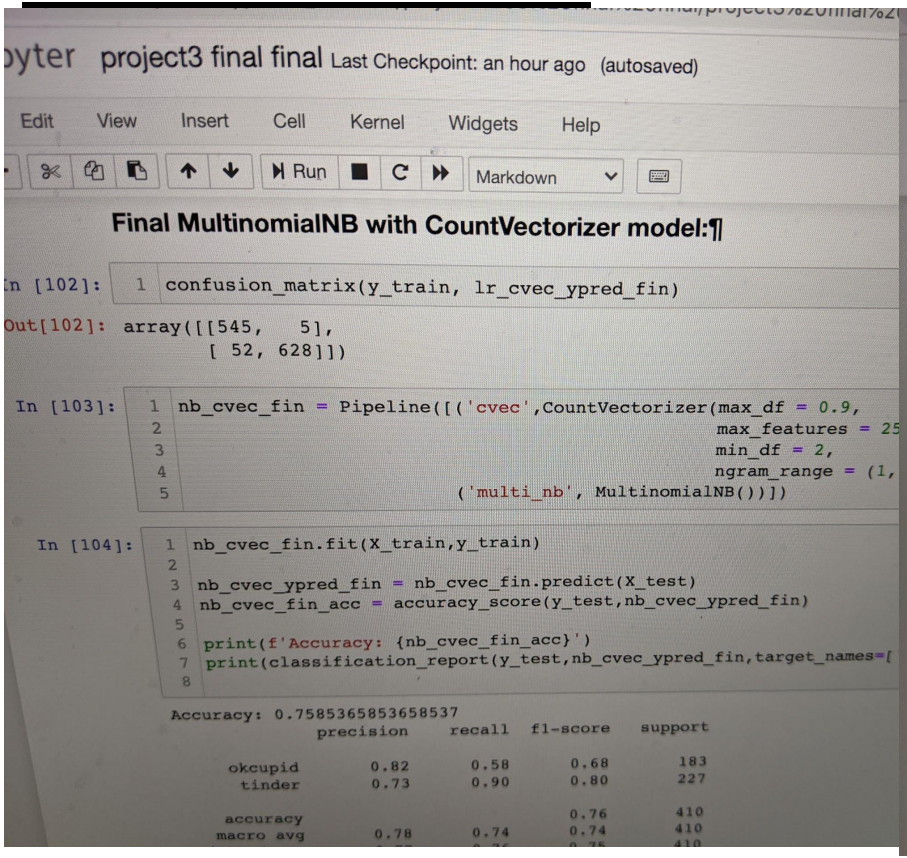


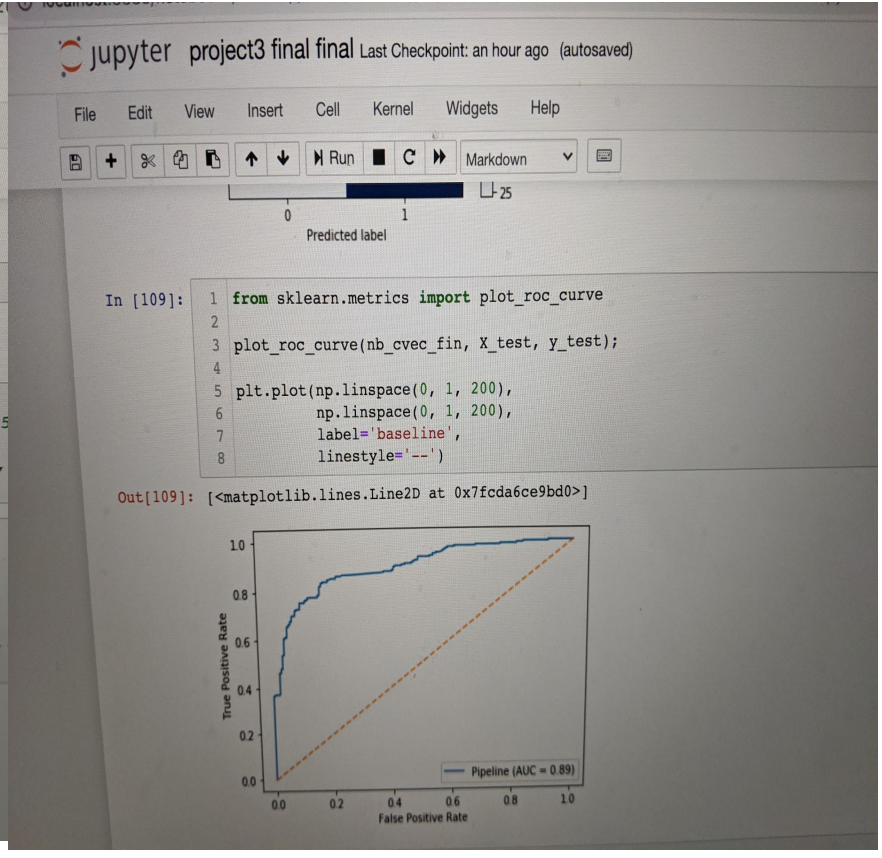
Logistic Regression with CountVectorizer model

1)Accuracy of predicting posts from Okcupid to be 83%

2) High AUC ROC score of 0.91,being able to separate Okcupid posts and tinder posts .

Final MultiNomialNaiveBayes with Countvectorizer





Final MultiNomialNaiveBayes with Countvectorizer

1)Accuracy of predicting posts from Okcupid to be 75%, lower than logistic regression.

2) High AUC ROC score of 0.89,being able to separate Okcupid posts and tinder posts .

Final MultiNomialNaiveBayes with TFIDFCountvectorizer

Final MultinomialNB with TfidfVectorizer model:

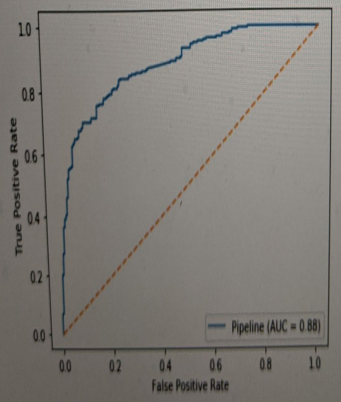
```
In [113]:
nb_tvec_fin = Pipeline([('tvec',TfidfVectorizer(max_df = 0.9,
                                                max_features = 2500,
                                                min_df = 3,
                                                ngram_range = (1, 2))),
                        ('multi_nb', MultinomialNB())])
```

```
In [114]:
nb_tvec_fin.fit(X_train,y_train)

nb_tvec_ypred_fin = nb_tvec_fin.predict(X_test)
nb_tvec_fin_acc = accuracy_score(y_test,nb_tvec_ypred_fin)

print(f'Accuracy: {nb_tvec_fin_acc}')
print(classification_report(y_test,nb_tvec_ypred_fin,target_names=['Okcupid','Tinder']))
```

```
Accuracy: 0.732360097323601
              precision    recall  f1-score   support

     Okcupid       0.86      0.49      0.62       185
      Tinder       0.69      0.93      0.79       226

    accuracy                           0.73       411
   macro avg       0.77      0.71      0.71       411
weighted avg       0.76      0.73      0.72       411
```

```
In [119]:
from sklearn.metrics import plot_roc_curve

plot_roc_curve(nb_tvec_fin, X_test, y_test);

plt.plot(np.linspace(0, 1, 200),
         np.linspace(0, 1, 200),
         label='baseline',
         linestyle='--')
```

```
Out[119]: [<matplotlib.lines.Line2D at 0x7ffd5f2c6a90>]
```



Final MultiNomialNaiveBayes with TFIDCountvectorizer

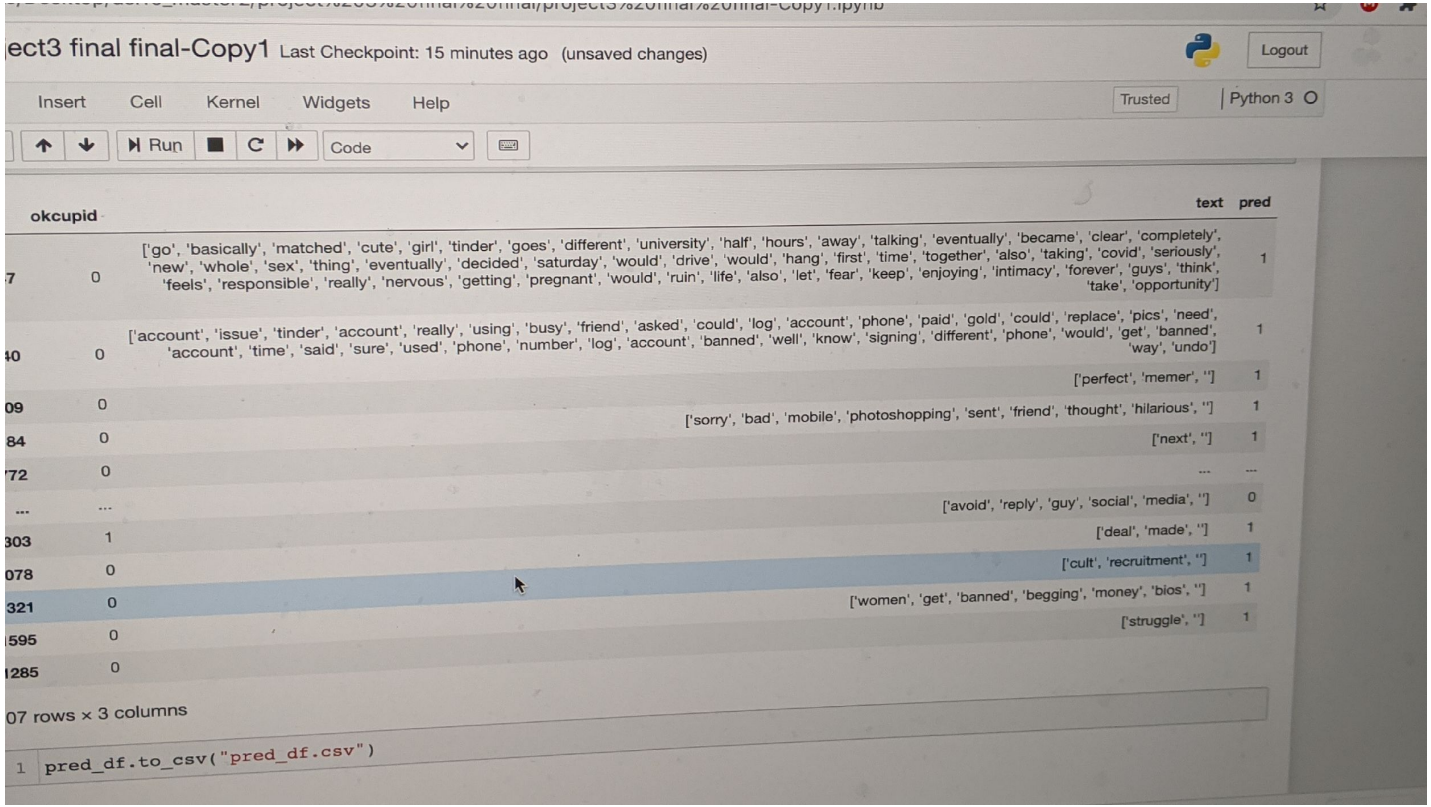
1)Accuracy of predicting posts from Okcupid to be 73%, lower than logistic regression.

2) High AUC ROC score of 0.88,being able to separate Okcupid posts and tinder posts .

5)Model Selection

Best model with accuracy of 86% and high AUC ROC score of 0.91 is **logistic regression with count vectorizer.**

MissClassification



Conclusions and Recommendations for the dating app startup to become a multibillion dollar company

1) We would use the logistic regression with count vectorizer model to predict by the text that a person send whether hes more into long term dating okcupid reddit or short term dating as of tinder subreddits.
2) We could do more models on different reddits like dating and forums to better predict the text from daters.
3) We could also do sentiment analysis for the startup to see the suitability of whether the person being positive or negative about dating .

Thank you and happy dating if you haven’t found love =) remember to use naive bayes model to predict if the partner you are dating is actually in for long term dating or short term dating.