

Project 3

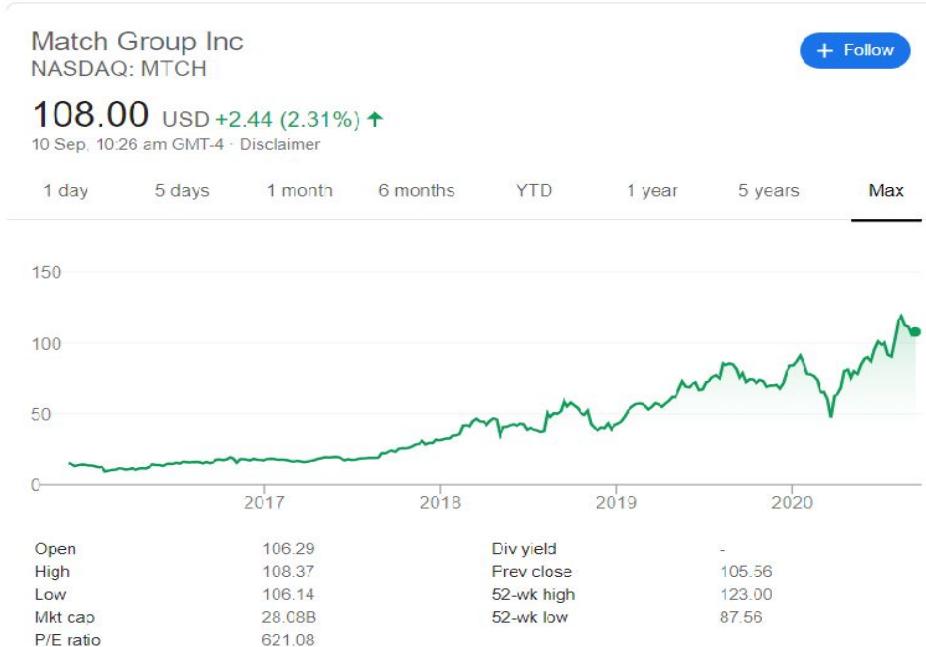
Build a classifier model to classify words that belong to either long term dating subreddits or short term subreddits. OKC vs tinder.

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.



Match Group

Online dating service company



Match Group, Inc. is an Internet company headquartered in Dallas that owns and operates several online dating services including Tinder, Match.com, Meetic, OkCupid, Hinge, PlentyOfFish, and OurTime. In 2019, the company had 9.283 million subscribers, of which 4.554 million were in North America. [Wikipedia](#)

Number of employees: 1,700 (2019)

CEO: Shar Dubey (1 Mar 2020–)

Revenue: 2.051 billion USD (2019)

Headquarters: Dallas, Texas, United States

Founded: 12 February 2009

Subsidiaries: Meetic, Plentyoffish Media, Care.com, Vimeo, LLC, MORE

Disclaimer

People also search for

View 15+ more

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

OkCupid on reddit

r/OkCupid

Hot New Top ...

PINNED BY MODERATORS

1 Posted by u/AutoModerator beep boop 8 hours ago

Critique Critique-Any-Section Thursdays

2 Comments Share Save ...

Dating site starterpack

↳ Crossposted by u/okcnottrowaway shittingposting 2 hours ago

54 r/starterpacks - Posted by u/LordOfSun55 1 month ago

Dating site starterpack

Dating site starterpack

The Creep

The Gold Digger

reddit

Search

POSTS WIKI REPORT

Hot New Top ...

PINNED BY MODERATORS

29 Posted by u/AutoModerator 2 days ago

Profile Review - Week of September 08, 2020

618 Comments Share Save ...

48 Posted by u/AutoModerator 3 days ago

Story Time - Week of September 07, 2020

275 Comments Share Save ...

15.6K Posted by u/HeeexC 6 hours ago

My friend's Tinder bio slays me.

35 less than a mile away

Just looking for someone to fuck me as hard as possible right in front of my cunt roommate until she gets uncomfortable and

About Community

A community for discussing the online dating app Tinder. Sharing conversations, reviewing profiles and more.

4.0m Possible Matches 10.7k Swiping

Created Feb 1, 2013

ADVERTISEMENT

Microsoft Azure

Start building apps today with 25+ free services

The 5 main processes

- 1) Webscraping
- 2) EDA (WordCloud and top 40 words) ,reading posts
- 3) Cleaning Data / lemmatization/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

The screenshot shows a Jupyter Notebook interface with a code cell titled "Webscraping". The code cell contains the following Python script:

```
[3]: 1 def get_posts(url,headers = {'User-agent':'Bleep borp bot 1.0'},loops=2):
2     posts = []
3     names = []
4     titles = []
5     subreddit = []
6     aft_name=None
7
8     for i in range(loops):
9         if aft_name==None:
10             params={}
11         else:
12             params={'after':aft_name}
13
14     req = requests.get(url,params=params,headers=headers)
15
16     if req.status_code == 200:
17         the_json = req.json()
18         for p in range(len(the_json['data']['children'])):
19             names.append(the_json['data']['children'][p]['data']['name'])
20             titles.append(the_json['data']['children'][p]['data']['title'])
21             posts.append(the_json['data']['children'][p]['data']['selftext'])
22             subreddit.append(the_json['data']['children'][p]['data']['subreddit'])
23             aft_name = the_json['data']['after']
24
25         print(res.status_code)
26         break
27
28     time.sleep(np.random.randint(1,5))
29
30 posts_df = pd.DataFrame({'names':names,
31                         'titles':titles,
32                         'posts':posts,
33                         'subreddit':subreddit})
```

The code defines a function `get_posts` that scrapes a specified URL for posts over two loops. It uses the `requests` library to make GET requests and the `json` library to parse the JSON response. The scraped data is stored in lists for names, titles, posts, and subreddit, which are then combined into a single DataFrame named `posts_df`.

1) WebScraping

Edit View Insert Cell Kernel Widgets Help

Truste

```
32
33
34
35     return posts_df

'posts':posts,
'subreddit':subreddit},columns = ['names','titles','posts','subreddit'])

In [4]: 1 TFR_df = get_posts(TFR,loops=40)

In [5]: 1 TFTS_df = get_posts(TFTS,loops=40)

In [6]: 1
2 len(TFR_df)

Out[6]: 994

In [7]: 1 len(TFTS_df)

Out[7]: 990

Removing duplicates as Reddit API limits requests of post to 1000 only and checking how much data is collected: combined the titles and posts and drop duplicates.

In [8]: 1 TFR_df['text'] = TFR_df['titles'] + TFR_df['posts']
2 TFTS_df['text'] = TFTS_df['titles'] + TFTS_df['posts']
3 TFR_df = TFR_df.drop_duplicates(subset='text',keep='first')
4 TFTS_df = TFTS_df.drop_duplicates(subset='text',keep='first')
5

In [9]: 1 len(TFR_df)

Out[9]: 907
```

tinder-pulls-.webp

tinderz

tinderlogo.webp

okc.png

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

The screenshot shows a Jupyter Notebook interface with the following details:

- Toolbar:** View, Insert, Cell, Kernel, Widgets, Help, Run, Code.
- Cell 18:** Contains Python code to fill NA values for posts where people only posted titles.

```
1 # fill NA values for posts of tinder where people only posted about titles and not posts
2
3 combined_df['posts'] = combined_df['posts'].fillna('')
4 combined_df.head()
```
- Data Preview:** A table showing rows 0 to 4 of the combined DataFrame. The columns are Unnamed: 0, names, titles, posts, subreddit, and text.

Unnamed: 0	names	titles	posts	subreddit	text
0	644 t3_lnqzmx	A Maserati in a world of kias 😊		Tinder	A Maserati in a world of kias 😊
1	380 t3_loy9ht	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,
2	504 t3_lo9bpy	Account Randomly Banned?	My account got randomly banned last night, any...	Tinder	Account Randomly Banned?My account got random...
3	568 t3_j8020s	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...

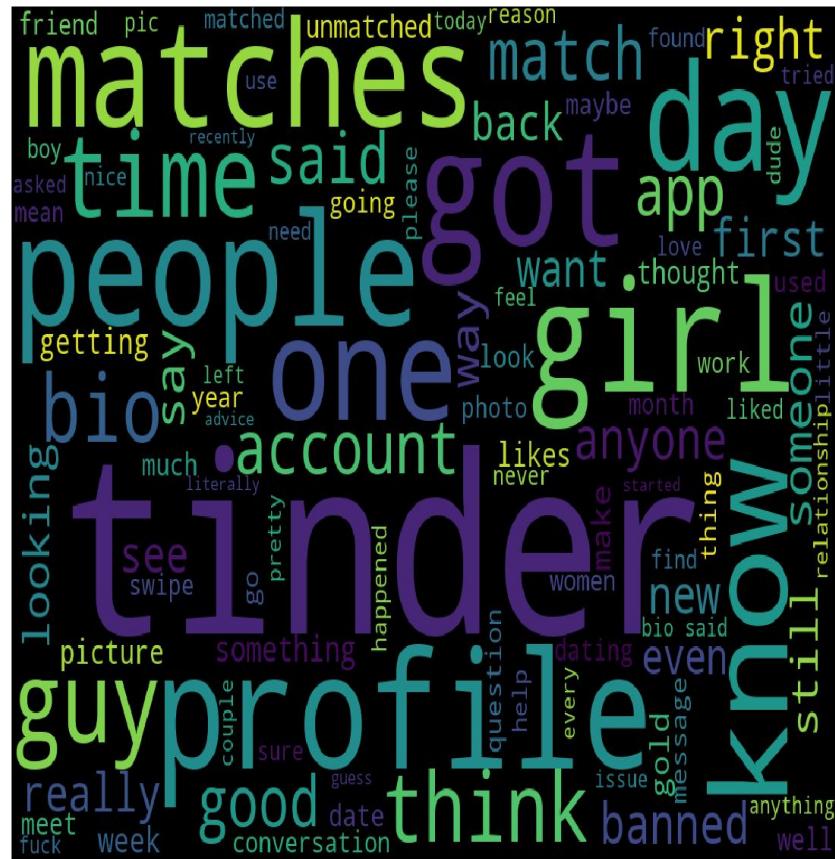
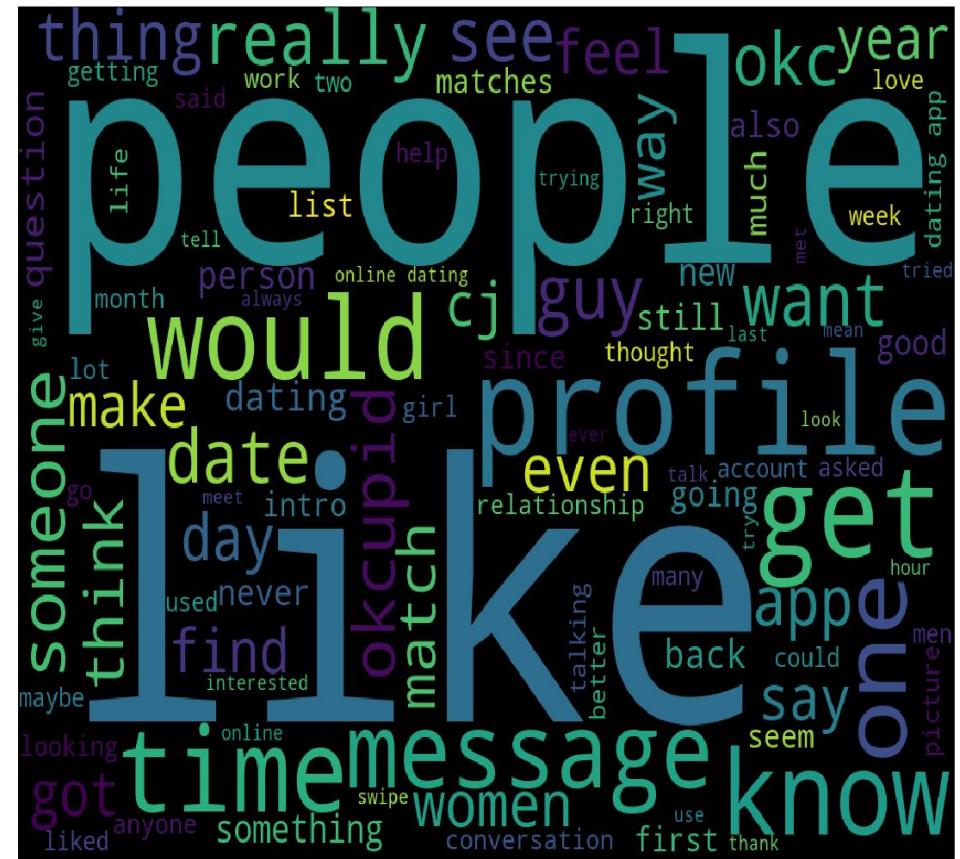
- Cell 19:** Contains Python code to apply lower case to all words in subreddit column.

```
1 #Apply lower case to all the words in subreddit.
2 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

File	Edit	View	Insert	Cell	Kernel	Widgets	Help
like	287	tinder	75				
people	189	,	63				
?	188	like	54				
get	177	get	52				
would	154	?	39				
know	149	got	37				
really	140	know	32				
time	137	people	31				
see	132	would	30				
profile	122	matches	28				
want	114	profile	26				
one	114	anyone	25				
even	111	said	24				
someone	106	new	23				
dating	103	one	22				
got	91	really	22				
find	89	even	21				
think	89	match	21				
message	89	good	20				
first	88	looking	20				
(87	days	20				
okc	82	girl	19				
anyone	82	see	19				
app	80	someone	19				
still	76	want	19				
likes	73	first	18				
okcupid	73	getting	18				
something	72	think	18				
new	71	account	18				
date	71	back	18				
feel	69	(17				
make	69	app	17				
person	69	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

to this and apply it to our newly created tokenized columns.

2.3 Addressing stopwords, Lemmatization/Stemming

```
1 #define function for removing stop words and using stopwords
2 def remove_stop_words(text):
3
4     en_stopwords = list(nltk.corpus.stopwords.words('english'))
5     text = [word for word in text if word not in en_stopwords]
6
7     return text
8
9
10 combined_df['p_stopped'] = combined_df['p_tokenized'].apply(lambda x: remove_stop_words(x))
11 combined_df['t_stopped'] = combined_df['t_tokenized'].apply(lambda x: remove_stop_words(x))
12
13
14 # Rearranging columns for visibility again
15 combined_df = combined_df[['names', 'titles', 't_tokenized', 't_stopped', 'posts', 'p_tokenized', 'p_stopped', 'subreddi
```

Cleaning (using regex to remove html tags etc)

```
In [23]: 1 #define function to clean text with regex and split words
2
3 def string_clean(text):
4
5     # Remove all HTML tags
6     text = re.sub(r'<.*?>', '', text)
7
8     # Remove non-letters.
9     text = re.sub("[^a-zA-Z]", " ", text)
10    # Remove all URL tags
11    text = re.sub(r'^https?:\/\/.*[\r\n]*', '', text)
12
13    # Keep text without punctuation
14    text = re.sub(r'{^\w\s}', '', text)
15
16    # convert text to lowercase
17    text = text.strip().lower()
18
19    # split text into a list of words
20    token_text = re.split('\W+',text)
21
22    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

The image shows two Jupyter Notebook environments side-by-side.

Left Notebook:

- In [53]:

```
1 X = df1['final_combined']
2 y = df1['okcupid']
```
- In [130]:

```
1 X.head()
```

any...

```
['maserati', 'world', 'kias', ''
1
['unsure', 'accept', '1
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]:

```
1
2 X_train, X_test, y_train, y_test = train_test_split(X,
3
4
5
6
    y,
    test_size=0.25,
    random_state=42,
    stratify=y)
```

Right Notebook:

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_com
0	0	t3_jnqzmx	A in a world of kias	['a', 'maserati', 'in', 'a', 'world', 'of', 'kias']	['maserati', 'world', 'world', 'kias']	NaN	[]	[]	[]	[]	tinder	'world', 'kia
1	1	t3_joy6ht	Unsure on how to accept likes and matches	['unsure', 'on', 'how', 'accept', 'accept', 'accept', 'likes...', 'likes...']	['unsure', 'to', 'accept', 'accept', 'accept', 'matches']	new to the app and I don't know how to...	['hi', 'new', 'app', 'know', 'fully', 'use', 'use', ...]	['hi', 'new', 'app', 'know', 'fully', 'use', 'use', ...]	My account got	tinder	'unsu 'accept', 'matche 'hi'	

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

The screenshot shows a Jupyter Notebook interface with a toolbar at the top and a code editor below. The code editor displays several cells:

- In [61]:** A Python code block defining `pipe_params` as a dictionary containing parameters for a CountVectorizer step. The parameters include `'cvec_max_features'`, `'cvec_min_df'`, `'cvec_max_df'`, and `'cvec_ngram_range'`.
- In [62]:** A Python code block creating a `GridSearchCV` object named `nb_grid` with the following parameters: `naive_bayes` as the estimator, `param_grid=pipe_params`, and `scoring='accuracy'`.
- A message "Fitting our training data to GridSearchCV model...." is displayed between the two code blocks.
- In [63]:** A Python code block fitting the `nb_grid` model to the training data `(X_train, y_train)`.
- Out[63]:** The output of the fit operation, showing the `GridSearchCV` object with its configuration: `cv=None`, `error_score=nan`, `estimator=Pipeline(memory=None,`, and `steps=[('cvec', CountVectorizer(analyzer='word', binary=False, decode_error='strict'))]`.

Logistic Regression with CountVectorizer model

interpret the results correspondingly below.

Final Logistic Regression with CountVectorizer model:

```
[5]: 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

False Negatives: 39
True Positives: 188

```
In [88]: 1 acc = (tp+tn)/(tn+fp+fn+tp)
2 print('The Accuracy rate is '+ str(acc))

The Accuracy rate is 0.8609756097560975
```

```
In [89]: 1 err = (fp+fn)/(tn+fp+fn+tp)
2 print('The Error rate is '+ str(err))

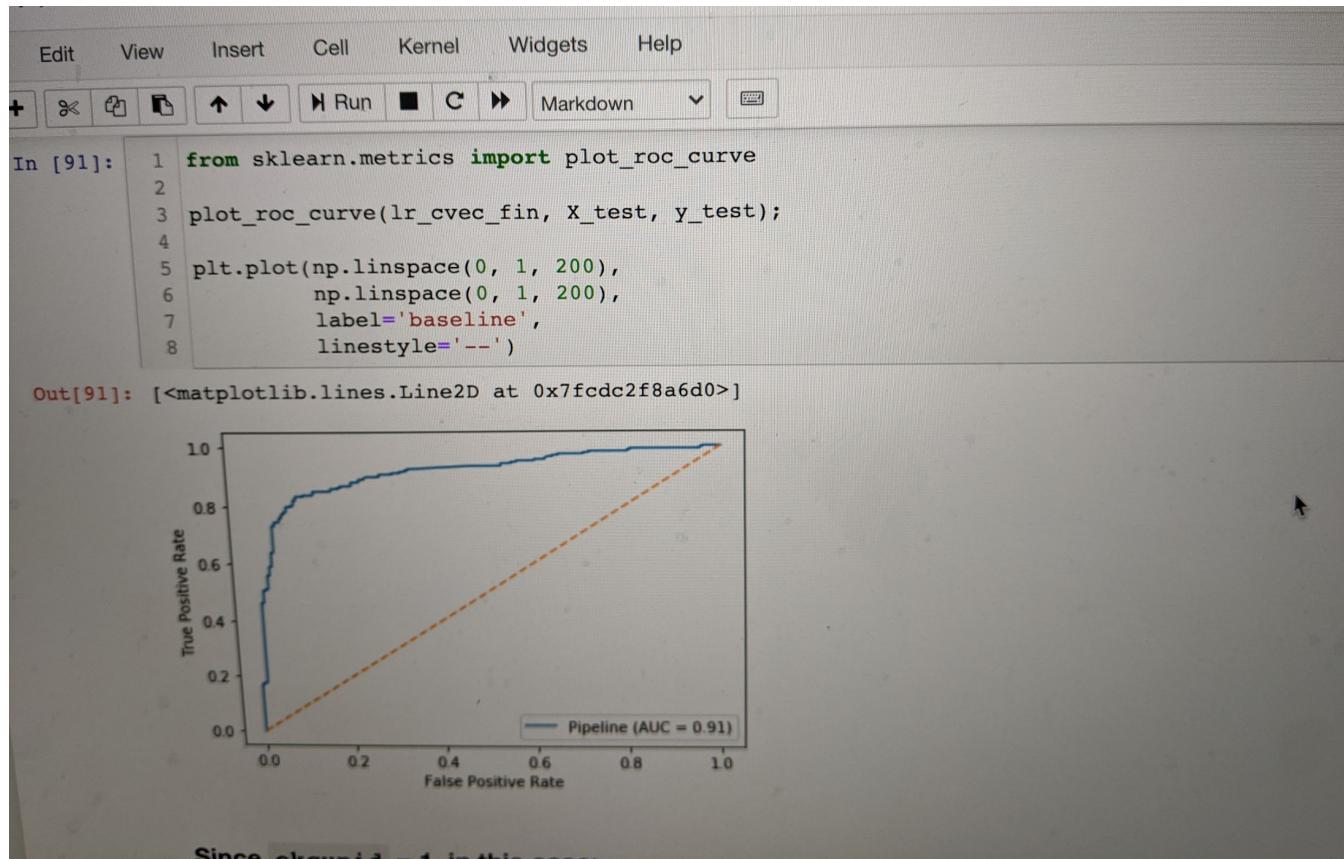
The Error rate is 0.13902439024390245
```

```
In [90]: 1
2 from sklearn.metrics import plot_confusion_matrix
3
4 plot_confusion_matrix(lr_cvec_fin , x_test, y_test, cmap='Blues', values_format='d');
```

Predicted		0	1
True label	0	165	18
1	39	188	

ad.png ^ okcupidwordcl....png ^ tinder-presskit.png ^ tinder-pulls-.....webp ^ tinderz

Logistic Regression with CountVectorizer model



Since `sklearn_id = 1` in this case,

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:

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

```
3]: 1
2 lr_cvec_ypred_fin = lr_cvec_fin.predict(X_train)
```

```
94]: 1 lr_tvec_fin.fit(X_train,y_train)
2
3 lr_tvec_ypred_fin = lr_tvec_fin.predict(X_test)
4 lr_tvec_fin_acc = accuracy_score(y_test,lr_tvec_ypred_fin)
5
6 print(f'Accuracy: {lr_tvec_fin_acc}')
7 print(classification_report(y_test,lr_tvec_ypred_fin,target_names=['okcupid','tinder']))
```

	precision	recall	f1-score	support
okcupid	0.83	0.80	0.81	183
tinder	0.87	0.85	0.86	227
accuracy	0.8365853658536585			

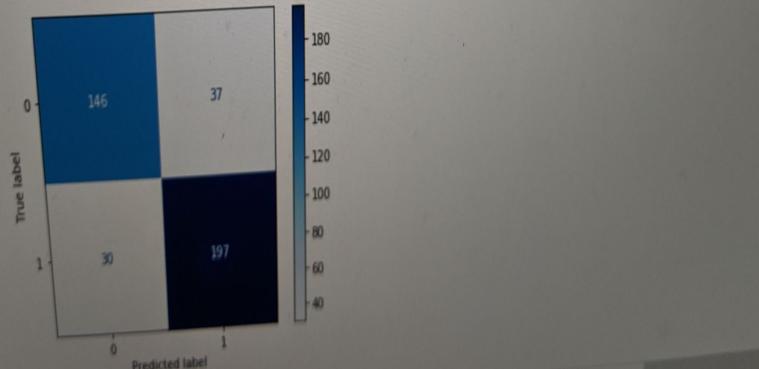
```
[96]: 1 acc = (tp+tn)/(tn+fp+fn+tp)
2 print('The Accuracy rate is '+ str(acc))
```

The Accuracy rate is 0.8365853658536585

```
In [97]: 1 err = (fp+fn)/(tn+fp+fn+tp)
2 print('The Error rate is '+ str(err))
```

The Error rate is 0.16341463414634147

```
In [98]: 1 from sklearn.metrics import plot_confusion_matrix
2
3
4 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

jupyter project3 final final Last Checkpoint: an hour ago (autosaved)

Edit View Insert Cell Kernel Widgets Help

Run C Markdown

Final MultinomialNB with CountVectorizer model:

```
In [102]: 1 confusion_matrix(y_train, lr_cvec_ypred_fin)
```

```
Out[102]: array([[545,      5,
       52, 628]])
```

```
In [103]: 1 nb_cvec_fin = Pipeline([('cvec',CountVectorizer(max_df = 0.9,
2                                         max_features = 25,
3                                         min_df = 2,
4                                         ngram_range = (1,
5                                         ('multi_nb', MultinomialNB())))] )
```

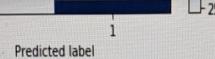
```
In [104]: 1 nb_cvec_fin.fit(X_train,y_train)
2
3 nb_cvec_ypred_fin = nb_cvec_fin.predict(X_test)
4 nb_cvec_fin_acc = accuracy_score(y_test,nb_cvec_ypred_fin)
5
6 print(f'Accuracy: {nb_cvec_fin_acc}')
7 print(classification_report(y_test,nb_cvec_ypred_fin,target_names=[
```

```
Accuracy: 0.7585365853658537
precision    recall   f1-score   support
okcupid      0.82      0.58      0.68      183
tinder        0.73      0.90      0.80      227
accuracy      0.76      0.76      0.74      410
macro avg    0.78      0.74      0.74      410
```

jupyter project3 final final Last Checkpoint: an hour ago (autosaved)

Edit View Insert Cell Kernel Widgets Help

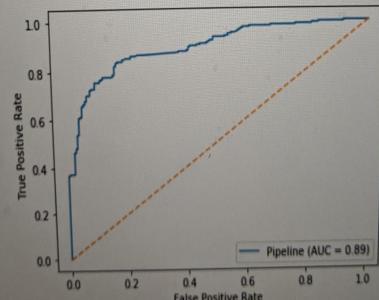
Run C Markdown



```
In [109]: 1 from sklearn.metrics import plot_roc_curve
```

```
2
3 plot_roc_curve(nb_cvec_fin, X_test, y_test);
4
5 plt.plot(np.linspace(0, 1, 200),
6          np.linspace(0, 1, 200),
7          label='baseline',
8          linestyle='--')
```

```
Out[109]: <matplotlib.lines.Line2D at 0x7fcda6ce9bd0>
```



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

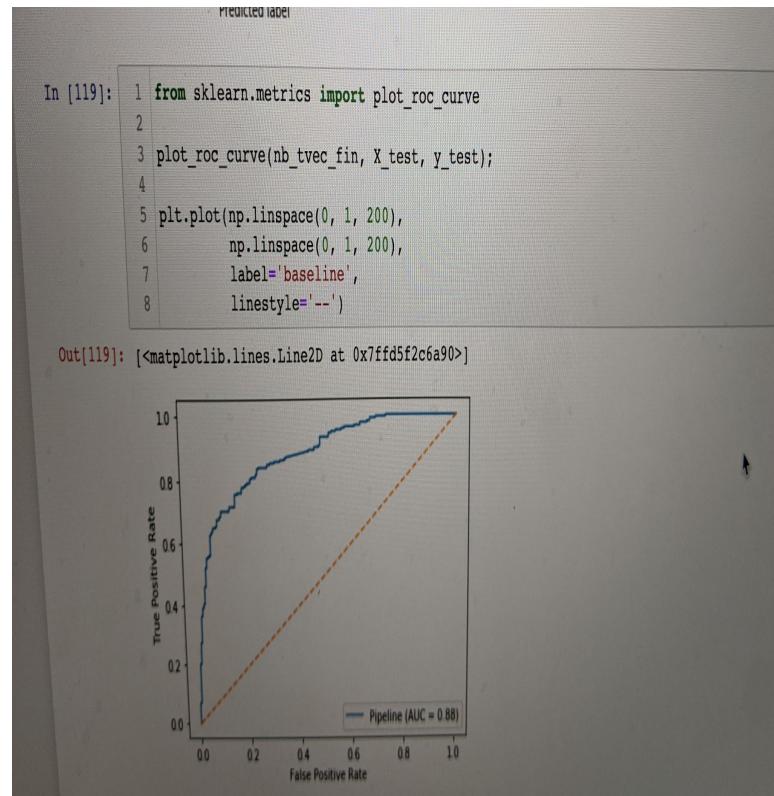
Final MultinomialNB with TfIdfVectorizer model:

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

```
1 nb_tvec_fin.fit(X_train,y_train)
2
3 nb_tvec_ypred_fin = nb_tvec_fin.predict(X_test)
4 nb_tvec_fin_acc = accuracy_score(y_test,nb_tvec_ypred_fin)
5
6 print(f'Accuracy: {nb_tvec_fin_acc}')
7 print(classification_report(y_test,nb_tvec_ypred_fin,target_names=['Okcupid','Tinder']))
8
```

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



Final MultiNomialNaiveBayes with TfidfCountvectorizer

- 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

ect3 final final-Copy1 Last Checkpoint: 15 minutes ago (unsaved changes)

Logout

Insert Cell Kernel Widgets Help Trusted Python 3

Run C Code

okcupid	text	pred
7	[go', 'basically', 'matched', 'cute', 'girl', 'tinder', 'goes', 'different', 'university', 'half', 'hours', 'away', 'talking', 'eventually', 'became', 'clear', 'completely', 'new', 'whole', 'sex', 'thing', 'eventually', 'decided', 'saturday', 'would', 'drive', 'would', 'hang', 'first', 'time', 'together', 'also', 'taking', 'covid', 'seriously', 'feels', 'responsible', 'really', 'nervous', 'getting', 'pregnant', 'would', 'ruin', 'life', 'also', 'let', 'fear', 'keep', 'enjoying', 'intimacy', 'forever', 'guys', 'think', 'take', 'opportunity']	1
40	['account', 'issue', 'tinder', 'account', 'really', 'using', 'busy', 'friend', 'asked', 'could', 'log', 'account', 'phone', 'paid', 'gold', 'could', 'replace', 'pics', 'need', 'account', 'time', 'said', 'sure', 'used', 'phone', 'number', 'log', 'account', 'banned', 'well', 'know', 'signing', 'different', 'phone', 'would', 'get', 'banned', 'way', 'undo']	1
09	['perfect', 'memer', '']	1
84	['sorry', 'bad', 'mobile', 'photoshopping', 'sent', 'friend', 'thought', 'hilarious', '']	1
72	['next', '']	1
...
303	['avoid', 'reply', 'guy', 'social', 'media', '']	0
078	['deal', 'made', '']	1
321	['cult', 'recruitment', '']	1
595	['women', 'get', 'banned', 'begging', 'money', 'bios', '']	1
1285	['struggle', '']	1

07 rows x 3 columns

```
1 pred_df.to_csv("pred_df.csv")
```

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 reddit 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.

