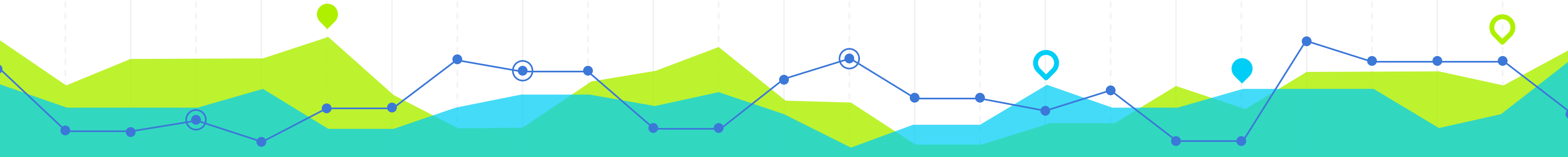
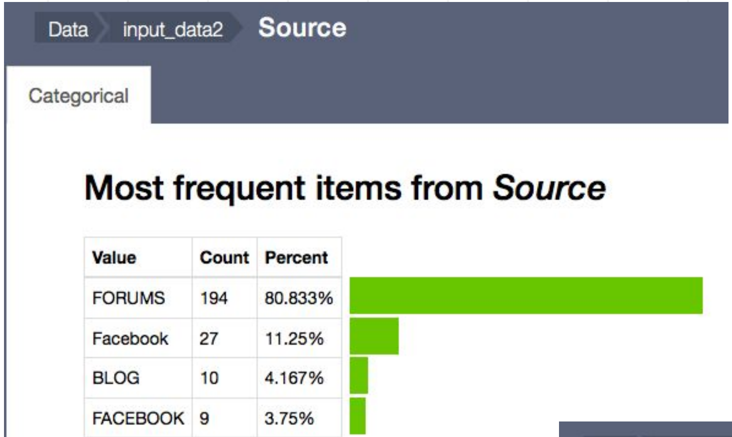


AKHIL PUNIA

VIT, VELLORE

Data Exploration

- Separating Patients and Non-Patients for observing General trends in various columns.
- Take for ex. Source, Host etc.
- Look for Number of Unique Values, NANs etc.



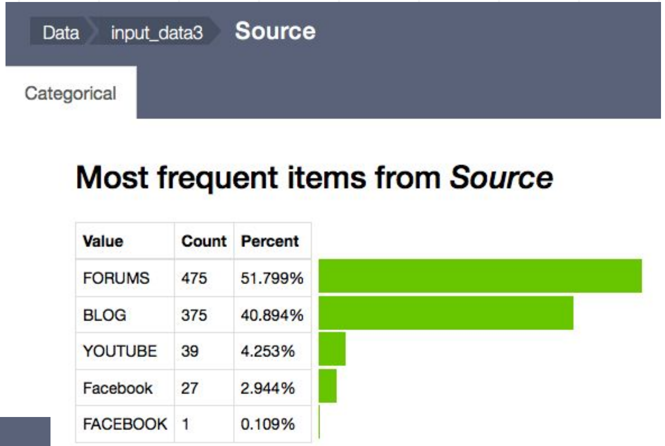
Data input_data2 **Source**

Categorical

Most frequent items from *Source*

Value	Count	Percent
FORUMS	194	80.833%
Facebook	27	11.25%
BLOG	10	4.167%
FACEBOOK	9	3.75%

Fig-1: Patients



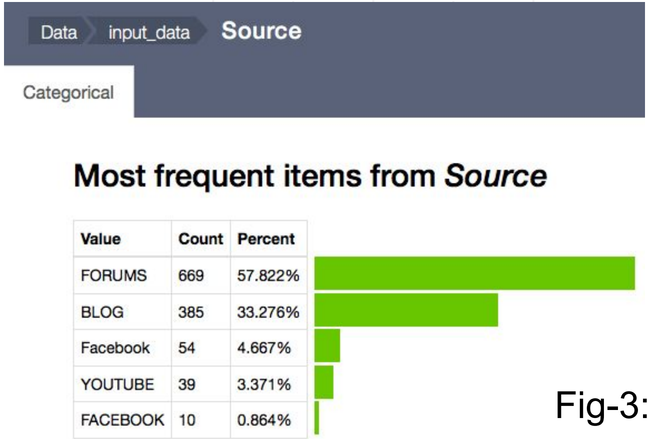
Data input_data3 **Source**

Categorical

Most frequent items from *Source*

Value	Count	Percent
FORUMS	475	51.799%
BLOG	375	40.894%
YOUTUBE	39	4.253%
Facebook	27	2.944%
FACEBOOK	1	0.109%

Fig-2: Non-Patients



Data input_data **Source**

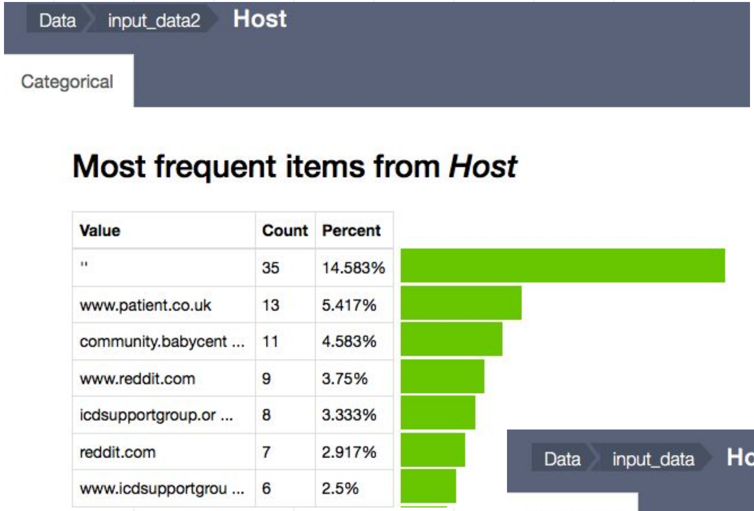
Categorical

Most frequent items from *Source*

Value	Count	Percent
FORUMS	669	57.822%
BLOG	385	33.276%
Facebook	54	4.667%
YOUTUBE	39	3.371%
FACEBOOK	10	0.864%

Fig-3: All Samples

- Values like FORUM have a huge significance in Positive Cases.



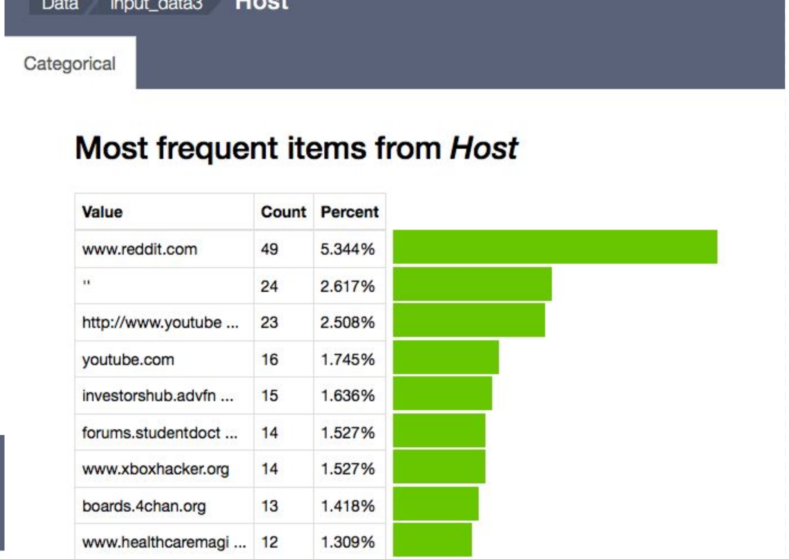
Data input_data2 **Host**

Categorical

Most frequent items from *Host*

Value	Count	Percent
''	35	14.583%
www.patient.co.uk	13	5.417%
community.babycent ...	11	4.583%
www.reddit.com	9	3.75%
icdsupportgroup.or ...	8	3.333%
reddit.com	7	2.917%
www.icdsupportgrou ...	6	2.5%

Fig-1: Patients



Data input_data3 **Host**

Categorical

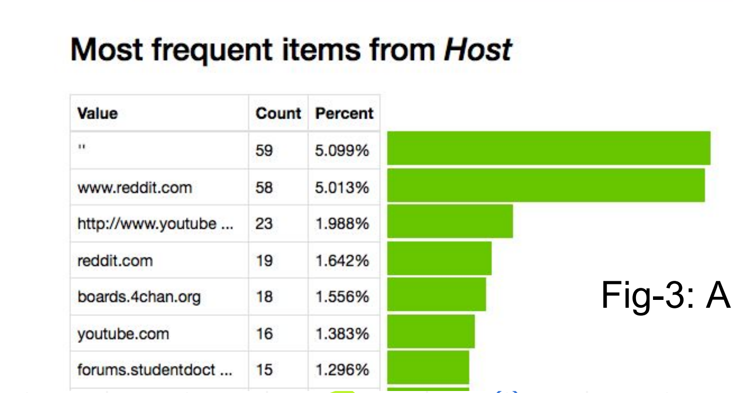
Most frequent items from *Host*

Value	Count	Percent
www.reddit.com	49	5.344%
''	24	2.617%
http://www.youtube ...	23	2.508%
youtube.com	16	1.745%
investorshub.advfn ...	15	1.636%
forums.studentdoct ...	14	1.527%
www.xboxhacker.org	14	1.527%
boards.4chan.org	13	1.418%
www.healthcaremagi ...	12	1.309%

Fig-2: Non-Patients

Data input_data **Host**

Categorical

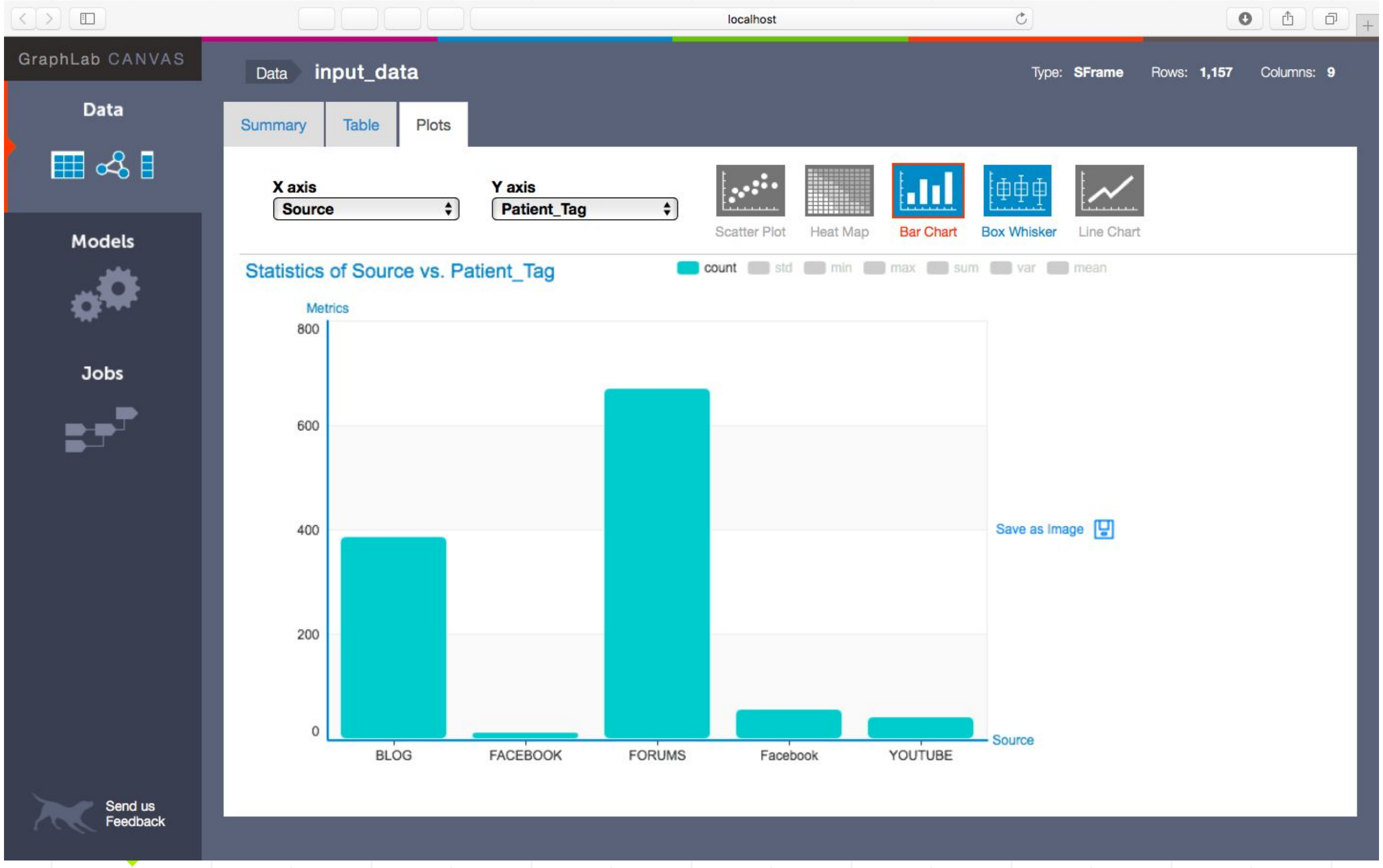


Most frequent items from *Host*

Value	Count	Percent
''	59	5.099%
www.reddit.com	58	5.013%
http://www.youtube ...	23	1.988%
reddit.com	19	1.642%
boards.4chan.org	18	1.556%
youtube.com	16	1.383%
forums.studentdoct ...	15	1.296%

Fig-3: All Samples

- Not Clean
- Requires preprocessing
- Overall impact- doubtful



Data input_data

Type: SFrame Rows: 1,157 Columns: 9

X axis: Source

Y axis: Patient_Tag

Insight into Data - I

- Variables like Host & Link are useful but, quite noisy.

  For ex: in the Link section we have ‘www.reddit.com’ and just ‘reddit.com’ , which has to be taken into account.
- Variables carrying date & time are not unique to help a classifier.
- Lots of Missing Values in ‘Title’ variable.

Insight into Data - II

- Variables like Source can be quite useful and can be used for classification as we see in case of ‘FORUMS’ .

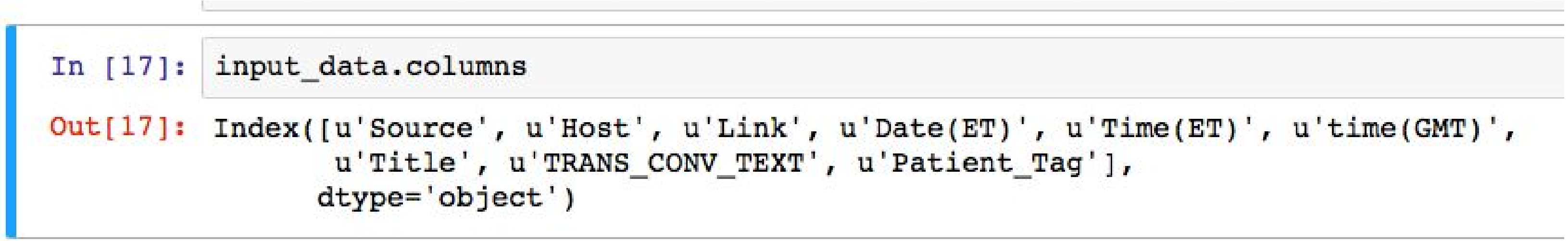
  ~ 80% of Patients have posted on Forums-> (194/240) .

- User Messages is a pretty unique variable with a lot of information. We can extract information out of it using Count Vectoriser and Tfid.

Analysis & Approach

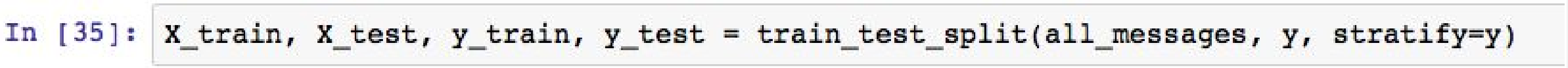
- I have selected the ‘TRANS_CONV_TEXT’ feature to make this problem similar to a Document Classification problem.

  where, our target variable is ‘Patient_Tag’.



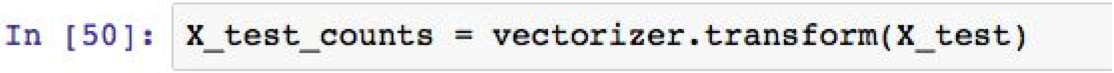
```
In [17]: input_data.columns

Out[17]: Index([u'Source', u'Host', u'Link', u'Date(ET)', u'Time(ET)', u'time(GMT)',
               u'Title', u'TRANS_CONV_TEXT', u'Patient_Tag'],
              dtype='object')
```

- I have splitted the data into Training & Testing.



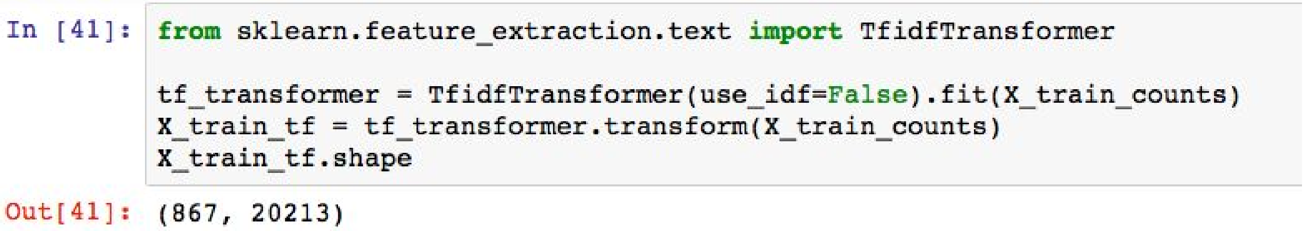
```
In [35]: X_train, X_test, y_train, y_test = train_test_split(all_messages, y, stratify=y)
```

- I have used count vectoriser to map the highest occurring words in the training set.
- Then, used this vectoriser to map the tests set.



```
In [50]: X_test_counts = vectorizer.transform(X_test)
```

- Then, I used tfid to extract the features which are important to the particular dataset.



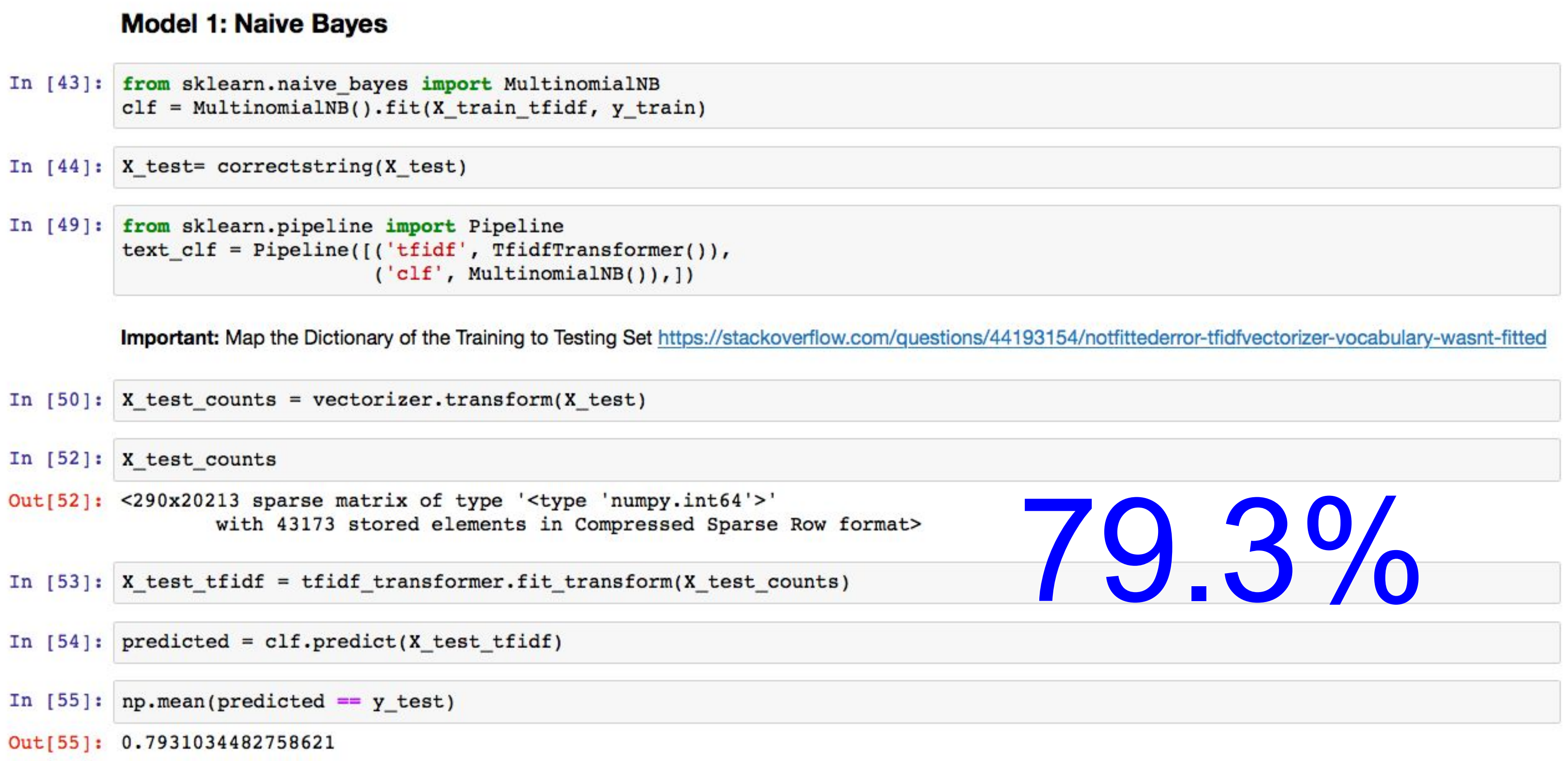
```
In [41]: from sklearn.feature_extraction.text import TfidfTransformer

         tf_transformer = TfidfTransformer(use_idf=False).fit(X_train_counts)
         X_train_tf = tf_transformer.transform(X_train_counts)
         X_train_tf.shape

Out[41]: (867, 20213)
```

Validation Results & Inferences

- I have tried 3 Models on the engineered features.

1. Naive Bayes
2. SGD Classifier
3. Random Forest

- While creating Train & Test sets, I have made sure the data split takes into account the unbalanced nature of dataset.
- ie. 240 +ve , 918 -ve cases.



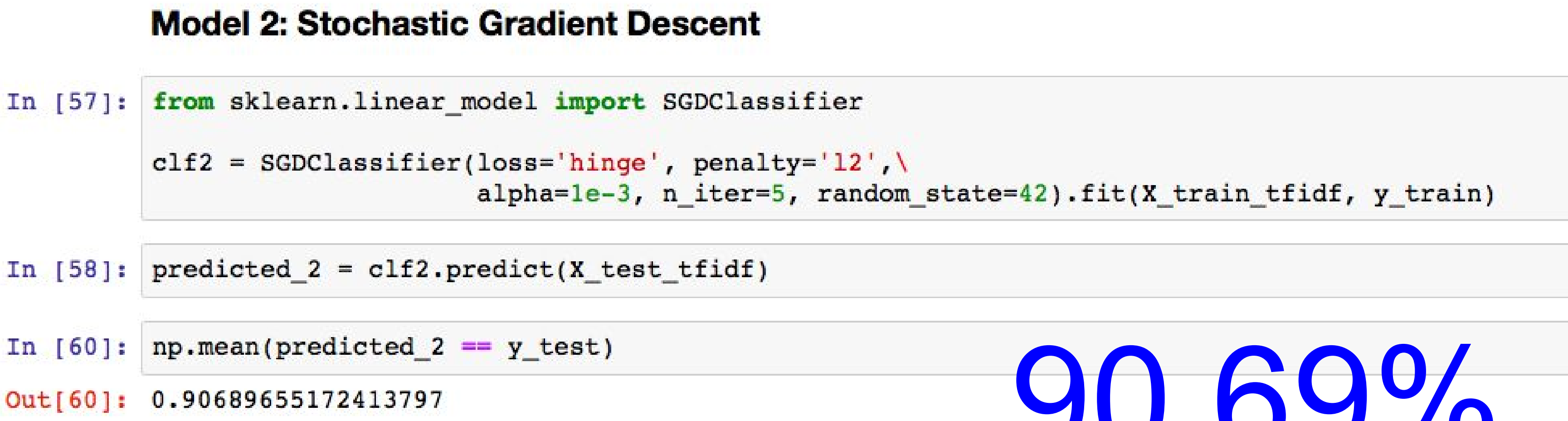
Model 1: Naive Bayes

```
In [43]: from sklearn.naive_bayes import MultinomialNB
         clf = MultinomialNB().fit(X_train_tfidf, y_train)

In [44]: X_test= correctstring(X_test)

In [49]: from sklearn.pipeline import Pipeline
         text_clf = Pipeline([('tfidf', TfidfTransformer()),
                              ('clf', MultinomialNB()),])
```

Important: Map the Dictionary of the Training to Testing Set https://stackoverflow.com/questions/44193154/notfittederror-tfidfvectorizer-vocabulary-wasnt-fitted

```
In [50]: X_test_counts = vectorizer.transform(X_test)

In [52]: X_test_counts

Out[52]: <290x20213 sparse matrix of type '<type 'numpy.int64'>'
                 with 43173 stored elements in Compressed Sparse Row format>

In [53]: X_test_tfidf = tfidf_transformer.fit_transform(X_test_counts)

In [54]: predicted = clf.predict(X_test_tfidf)

In [55]: np.mean(predicted == y_test)

Out[55]: 0.7931034482758621
```

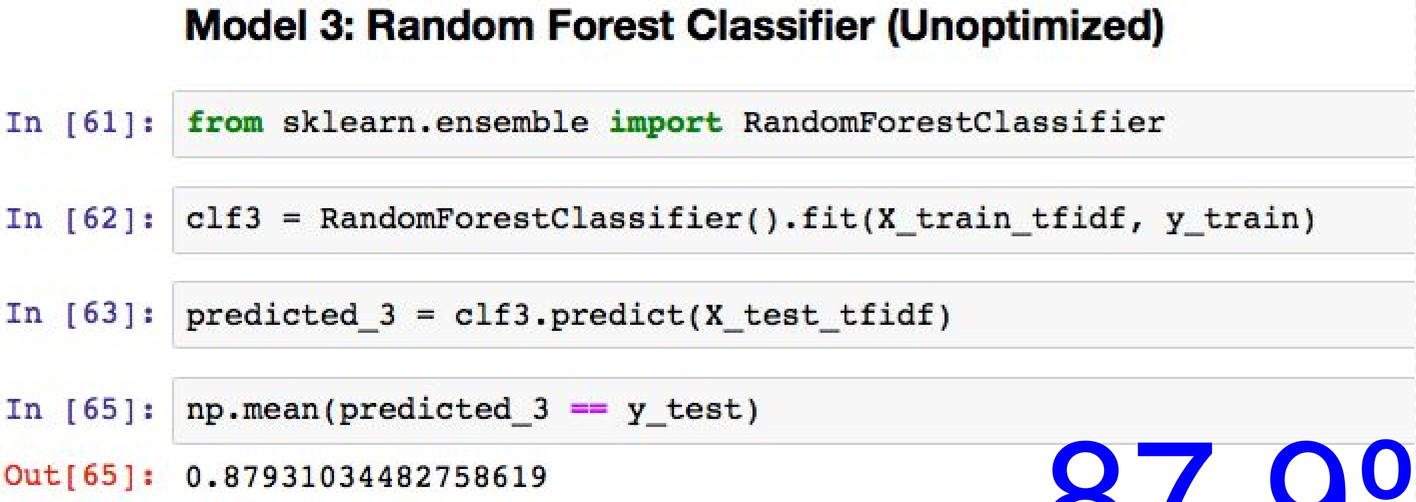
79.3%



Model 2: Stochastic Gradient Descent

```
In [57]: from sklearn.linear_model import SGDClassifier

         clf2 = SGDClassifier(loss='hinge', penalty='l2',\
                                  alpha=1e-3, n_iter=5, random_state=42).fit(X_train_tfidf, y_train)

In [58]: predicted_2 = clf2.predict(X_test_tfidf)

In [60]: np.mean(predicted_2 == y_test)

Out[60]: 0.90689655172413797
```

90.69%



Model 3: Random Forest Classifier (Unoptimized)

```
In [61]: from sklearn.ensemble import RandomForestClassifier

In [62]: clf3 = RandomForestClassifier().fit(X_train_tfidf, y_train)

In [63]: predicted_3 = clf3.predict(X_test_tfidf)

In [65]: np.mean(predicted_3 == y_test)

Out[65]: 0.8793103448275862
```

87.9%

Possible Next Steps

- Implement GridSearchCV to find the best hyperparameter for the RF Classifier.
- Use state-of-the-art XGBoost algorithm to improve the Results.
- Instead, of focusing on just the Accuracy, we can use f-score as the metric to give more weightage to predicting the Patient Correctly.