# CS6005 Deep Learning Techniques

## Natural Language Processing Project

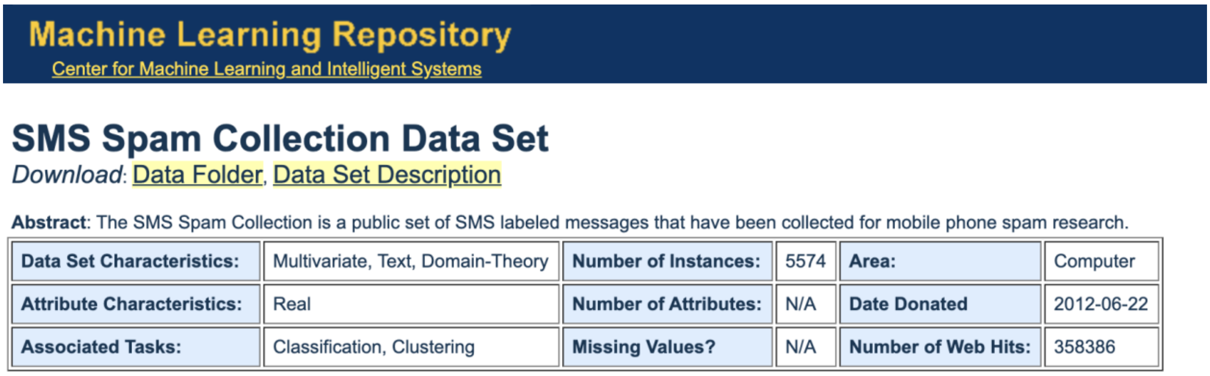
### Spam V/S Ham SMS Classification

|  |  |
|---|---|
| **Name** | Gokul. S |
| **Roll Number** | 2018103026 |
| **Batch** | P |
| **Semester** | 6 (RUSA) |
| **Date of Submission** | 21. 05. 2021 |
| **Github Profile** | www.github.com/gokul-sunilkumar |

**Problem Statement:** To classify an SMS message as either a spam/ham using text analytics, natural language processing and machine learning

**Dataset:** SMS Spam Collection Data Set

**Description**: The dataset consists of more than 5000 SMS phone messages belonging to two classes namely spam, ham.



Machine Learning Repository

Center for Machine Learning and Intelligent Systems

SMS Spam Collection Data Set

*Download*: Data Folder, Data Set Description

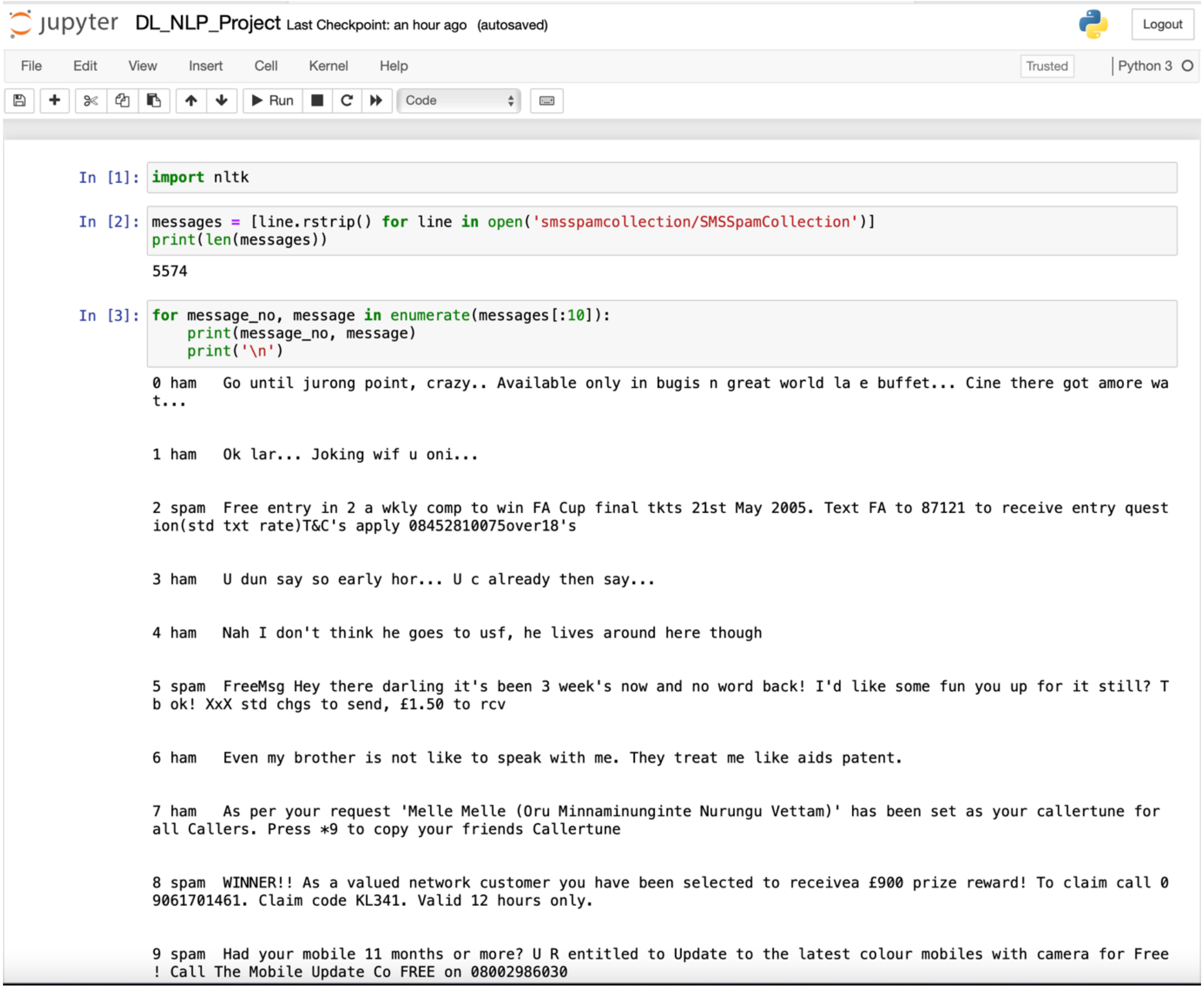
**Abstract**: The SMS Spam Collection is a public set of SMS labeled messages that have been collected for mobile phone spam research.

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate, Text, Domain-Theory | **Number of Instances:** | 5574 | **Area:** | Computer |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | N/A | **Date Donated** | 2012-06-22 |
| **Associated Tasks:** | Classification, Clustering | **Missing Values?** | N/A | **Number of Web Hits:** | 358386 |

**URL**: (https://archive.ics.uci.edu/ml/datasets/sms+spam+collection) → UCI Machine Learning Repository

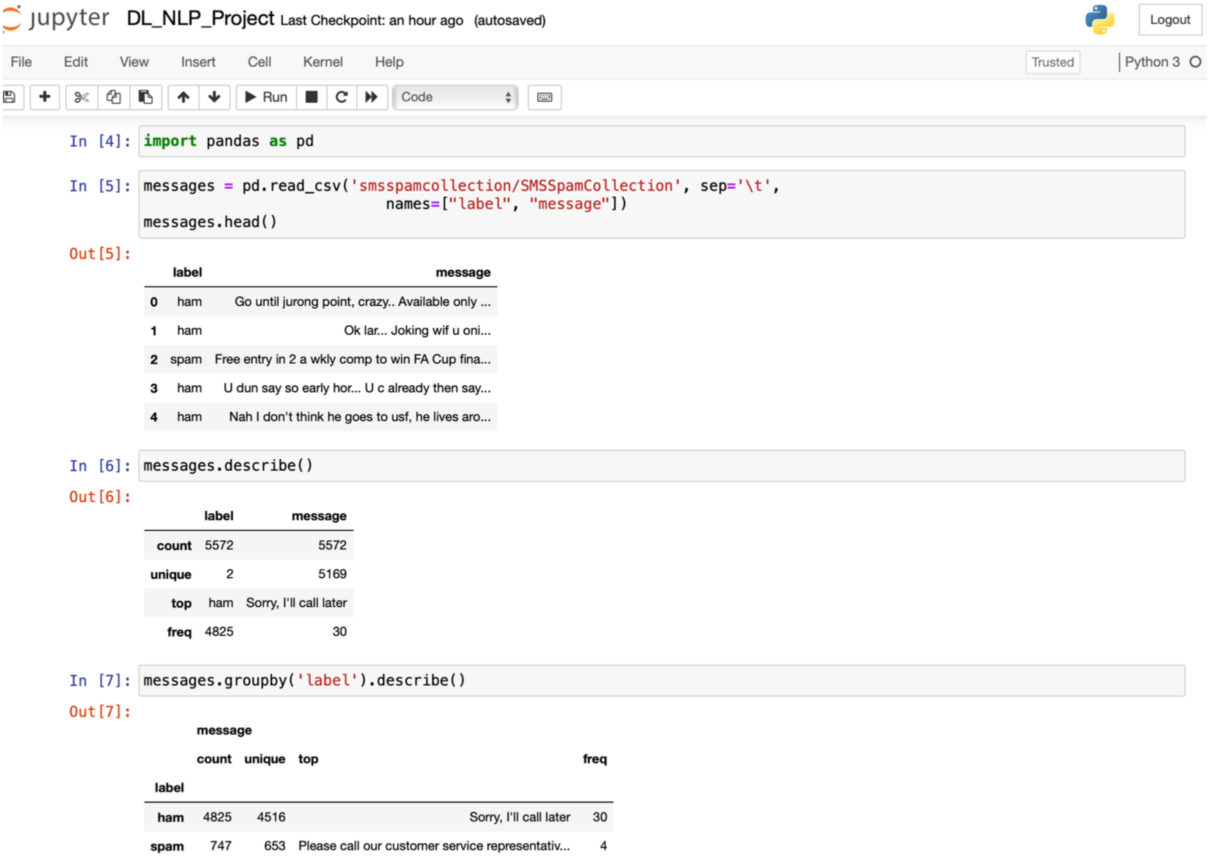
## Code/ Execution Snapshots:

# Importing packages and checking the structure, shape and contents of the dataset.



```
In [1]: import nltk

In [2]: messages = [line.rstrip() for line in open('smsspamcollection/SMSSpamCollection')]
        print(len(messages))

        5574

In [3]: for message_no, message in enumerate(messages[:10]):
            print(message_no, message)
            print('\n')

        0 ham   Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wa
        t...


        1 ham   Ok lar... Joking wif u oni...


        2 spam  Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry quest
        ion(std txt rate)T&C's apply 08452810075over18's


        3 ham   U dun say so early hor... U c already then say...


        4 ham   Nah I don't think he goes to usf, he lives around here though


        5 spam  FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? T
        b ok! XxX std chgs to send, £1.50 to rcv


        6 ham   Even my brother is not like to speak with me. They treat me like aids patent.


        7 ham   As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for
        all Callers. Press *9 to copy your friends Callertune


        8 spam  WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 0
        9061701461. Claim code KL341. Valid 12 hours only.


        9 spam  Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free
        ! Call The Mobile Update Co FREE on 08002986030
```

# Read the csv data file and create a Pandas Dataframe. Check how balanced the dataset is.



In [4]:
```
import pandas as pd
```

In [5]:
```
messages = pd.read_csv('smsspamcollection/SMSSpamCollection', sep='\t',
                           names=["label", "message"])
messages.head()
```

Out[5]:

|  | label | message |
|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... |
| 1 | ham | Ok lar... Joking wif u oni... |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... |
| 3 | ham | U dun say so early hor... U c already then say... |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... |

In [6]:
```
messages.describe()
```

Out[6]:

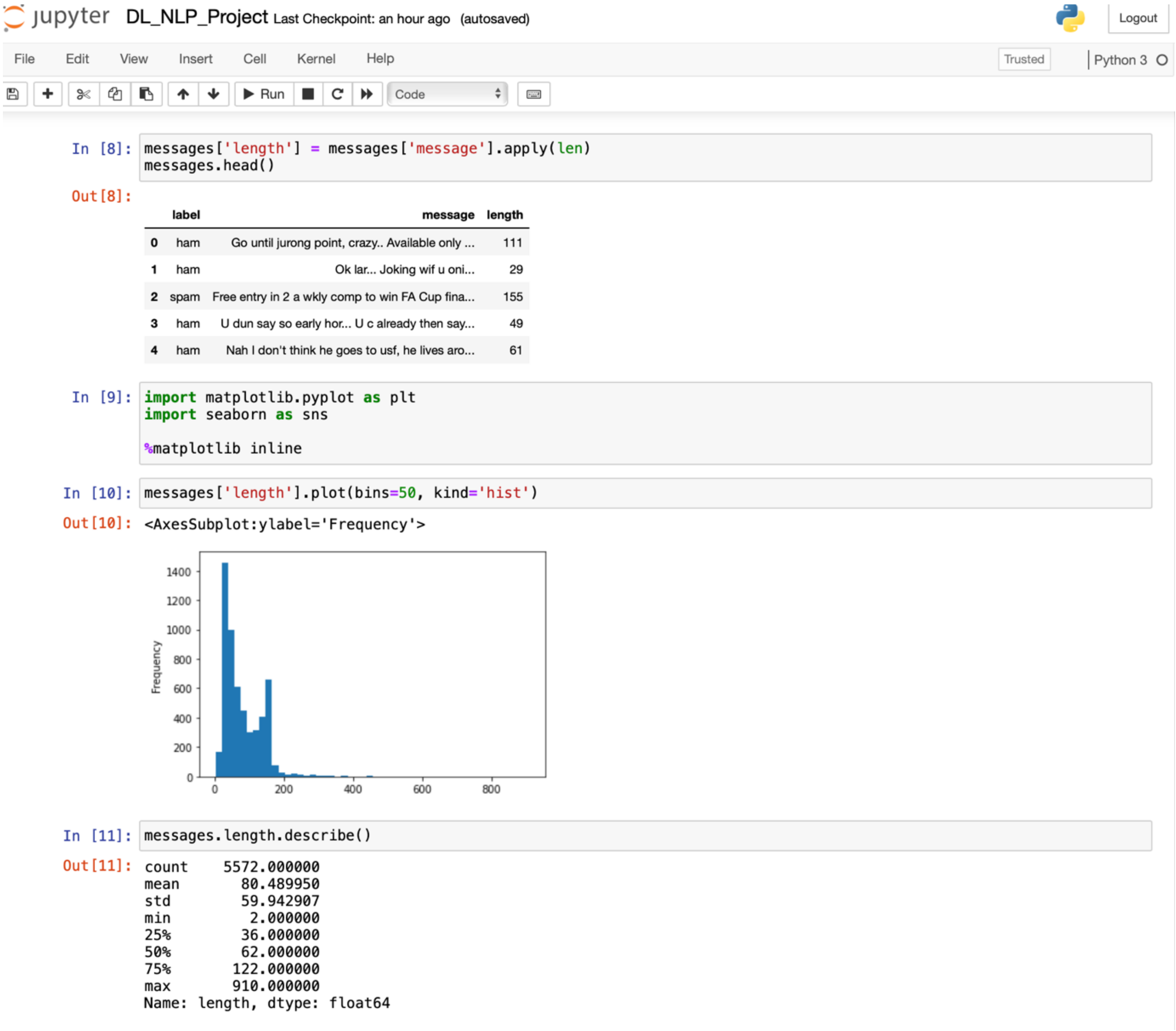
|  | label | message |
|---|---|---|
| count | 5572 | 5572 |
| unique | 2 | 5169 |
| top | ham | Sorry, I'll call later |
| freq | 4825 | 30 |

In [7]:
```
messages.groupby('label').describe()
```

Out[7]:

|  | message |  |  |  |
|---|---|---|---|---|
|  | count | unique | top | freq |
| label |  |  |  |  |
| ham | 4825 | 4516 | Sorry, I'll call later | 30 |
| spam | 747 | 653 | Please call our customer service representativ... | 4 |

# Create an additional meaningful column called length and perform Exploratory Data Analysis (EDA)



In [8]:

```
messages['length'] = messages['message'].apply(len)
messages.head()
```

Out[8]:

| | label | message | length |
|---|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... | 111 |
| 1 | ham | Ok lar... Joking wif u oni... | 29 |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | 155 |
| 3 | ham | U dun say so early hor... U c already then say... | 49 |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... | 61 |

In [9]:

```
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [10]:

```
messages['length'].plot(bins=50, kind='hist')
```

Out[10]:

```
<AxesSubplot:ylabel='Frequency'>
```

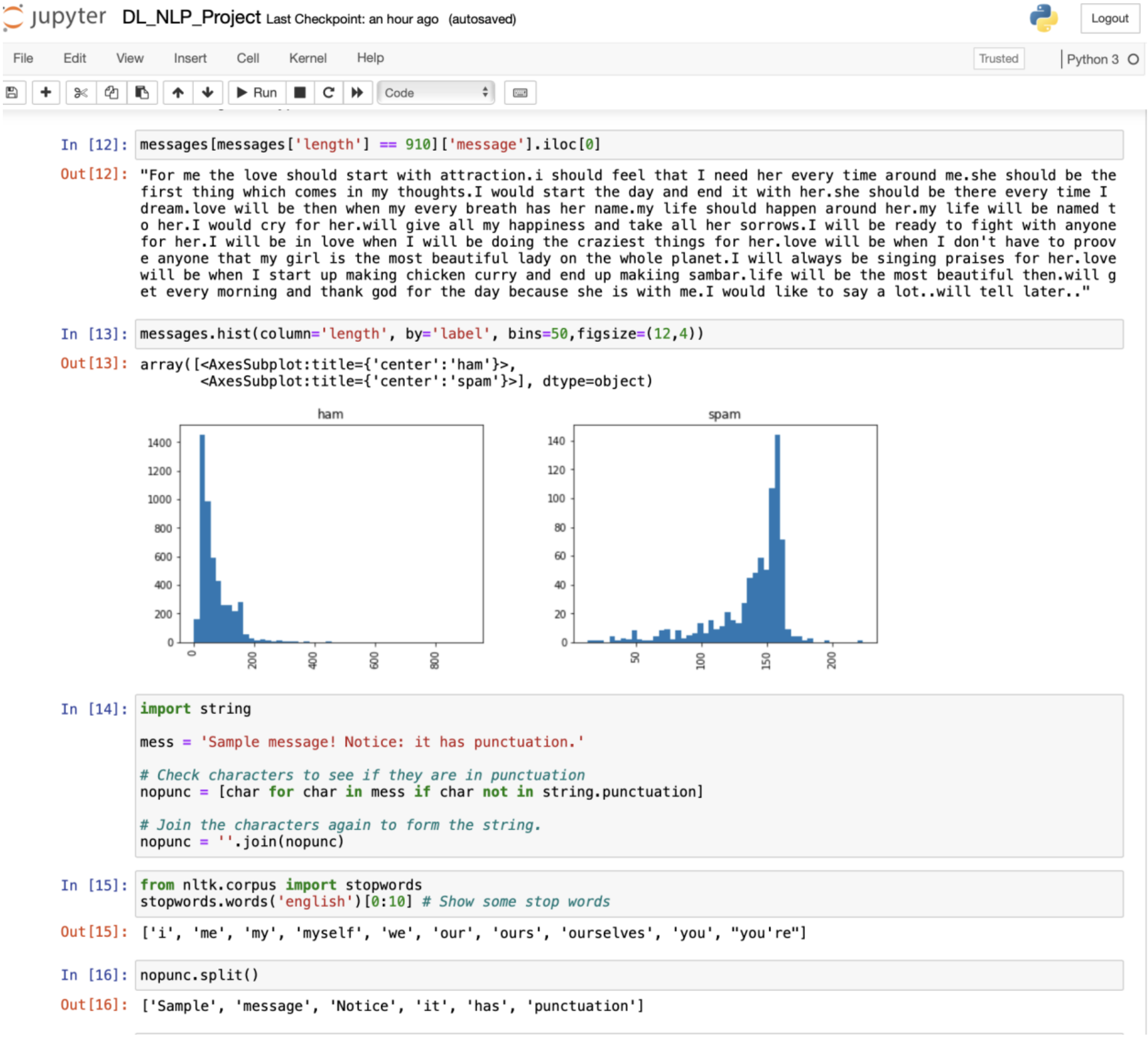
In [11]:

```
messages.length.describe()
```

Out[11]:

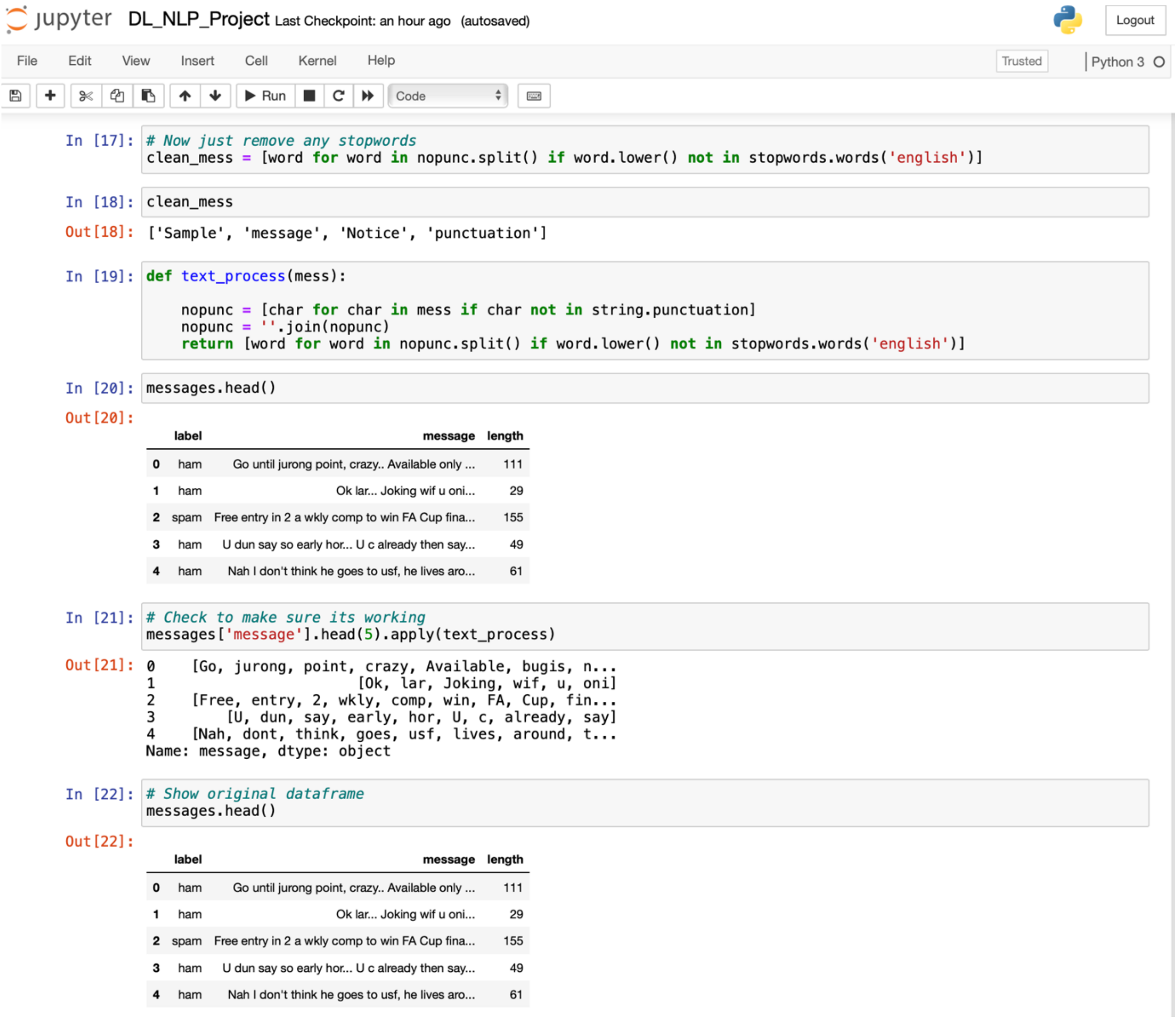
```
count    5572.000000
mean       80.489950
std        59.942907
min         2.000000
25%        36.000000
50%        62.000000
75%       122.000000
max       910.000000
Name: length, dtype: float64
```

# Remove punctuations and stopwords from the input messages



```
In [12]: messages[messages['length'] == 910]['message'].iloc[0]

Out[12]: "For me the love should start with attraction.i should feel that I need her every time around me.she should be the
         first thing which comes in my thoughts.I would start the day and end it with her.she should be there every time I
         dream.love will be then when my every breath has her name.my life should happen around her.my life will be named t
         o her.I would cry for her.will give all my happiness and take all her sorrows.I will be ready to fight with anyone
         for her.I will be in love when I will be doing the craziest things for her.love will be when I don't have to proov
         e anyone that my girl is the most beautiful lady on the whole planet.I will always be singing praises for her.love
         will be when I start up making chicken curry and end up makiing sambar.life will be the most beautiful then.will g
         et every morning and thank god for the day because she is with me.I would like to say a lot..will tell later.."

In [13]: messages.hist(column='length', by='label', bins=50,figsize=(12,4))

Out[13]: array([<AxesSubplot:title={'center':'ham'}>,
                <AxesSubplot:title={'center':'spam'}>], dtype=object)

In [14]: import string

         mess = 'Sample message! Notice: it has punctuation.'

         # Check characters to see if they are in punctuation
         nopunc = [char for char in mess if char not in string.punctuation]

         # Join the characters again to form the string.
         nopunc = ''.join(nopunc)

In [15]: from nltk.corpus import stopwords
         stopwords.words('english')[0:10] # Show some stop words

Out[15]: ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]

In [16]: nopunc.split()

Out[16]: ['Sample', 'message', 'Notice', 'it', 'has', 'punctuation']
```

# Pre-process the dataframe using the above strategies and update it in place.



In [17]:
```
# Now just remove any stopwords
clean_mess = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]
```

In [18]:
```
clean_mess
```

Out[18]:
```
['Sample', 'message', 'Notice', 'punctuation']
```

In [19]:
```
def text_process(mess):

    nopunc = [char for char in mess if char not in string.punctuation]
    nopunc = ''.join(nopunc)
    return [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]
```

In [20]:
```
messages.head()
```

Out[20]:

| | label | message | length |
|---|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... | 111 |
| 1 | ham | Ok lar... Joking wif u oni... | 29 |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | 155 |
| 3 | ham | U dun say so early hor... U c already then say... | 49 |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... | 61 |

In [21]:
```
# Check to make sure its working
messages['message'].head(5).apply(text_process)
```

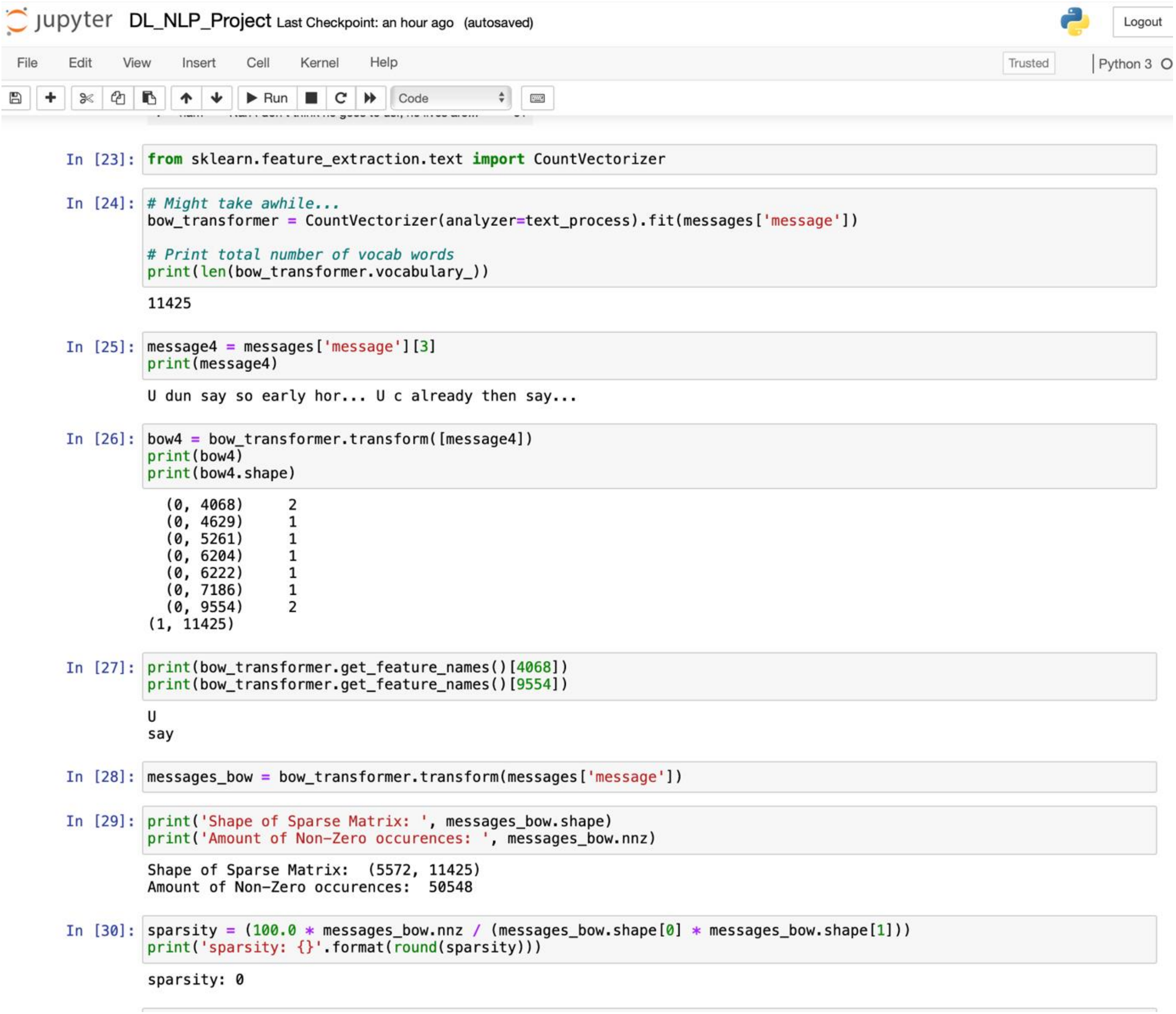
Out[21]:
```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                       [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: message, dtype: object
```

In [22]:
```
# Show original dataframe
messages.head()
```

Out[22]:

| | label | message | length |
|---|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... | 111 |
| 1 | ham | Ok lar... Joking wif u oni... | 29 |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | 155 |
| 3 | ham | U dun say so early hor... U c already then say... | 49 |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... | 61 |

# Convert cleaned messages to their bag of words form using Sklearn’s CountVectorizer



```
In [23]: from sklearn.feature_extraction.text import CountVectorizer
```

```
In [24]: # Might take awhile...
         bow_transformer = CountVectorizer(analyzer=text_process).fit(messages['message'])

         # Print total number of vocab words
         print(len(bow_transformer.vocabulary_))

         11425
```

```
In [25]: message4 = messages['message'][3]
         print(message4)

         U dun say so early hor... U c already then say...
```

```
In [26]: bow4 = bow_transformer.transform([message4])
         print(bow4)
         print(bow4.shape)

           (0, 4068)     2
           (0, 4629)     1
           (0, 5261)     1
           (0, 6204)     1
           (0, 6222)     1
           (0, 7186)     1
           (0, 9554)     2
         (1, 11425)
```

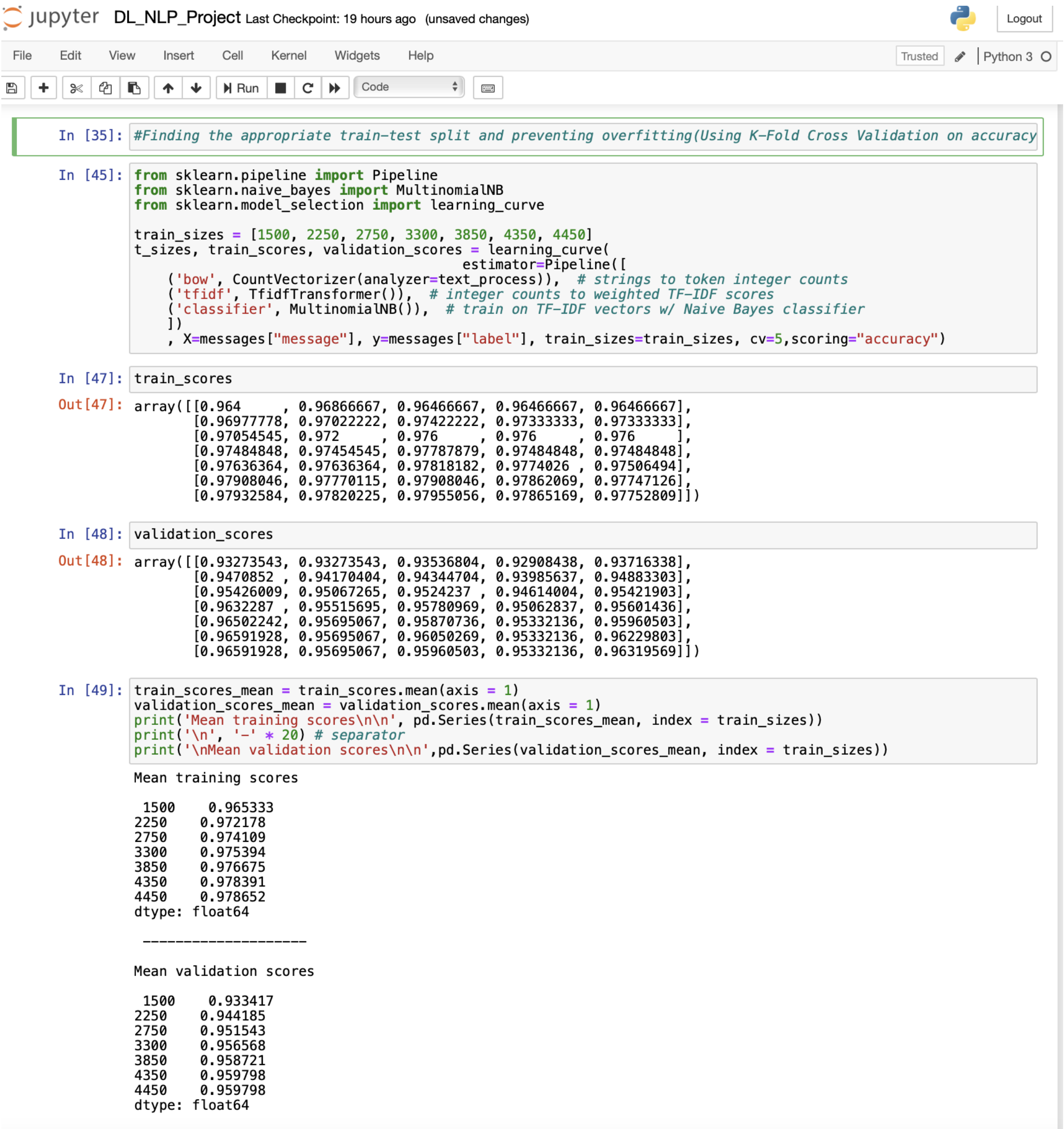
```
In [27]: print(bow_transformer.get_feature_names()[4068])
         print(bow_transformer.get_feature_names()[9554])

         U
         say
```

```
In [28]: messages_bow = bow_transformer.transform(messages['message'])
```

```
In [29]: print('Shape of Sparse Matrix: ', messages_bow.shape)
         print('Amount of Non-Zero occurences: ', messages_bow.nnz)

         Shape of Sparse Matrix:  (5572, 11425)
         Amount of Non-Zero occurences:  50548
```

```
In [30]: sparsity = (100.0 * messages_bow.nnz / (messages_bow.shape[0] * messages_bow.shape[1]))
         print('sparsity: {}'.format(round(sparsity)))

         sparsity: 0
```

# Use Sklearn’s learning_curve function to plot the accuracy curve for different train-test split sizes (considering 50:50, 60:40, 70:30, 80:20) with a cross validation K-Fold value of 5 and the average of the five subordinate accuracies is the overall accuracy for that train size.


jupyter DL_NLP_Project Last Checkpoint: 19 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 ○
Run
Code

```
In [35]: #Finding the appropriate train-test split and preventing overfitting(Using K-Fold Cross Validation on accuracy
```

```
In [45]: from sklearn.pipeline import Pipeline
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.model_selection import learning_curve

         train_sizes = [1500, 2250, 2750, 3300, 3850, 4350, 4450]
         t_sizes, train_scores, validation_scores = learning_curve(
                                                 estimator=Pipeline([
             ('bow', CountVectorizer(analyzer=text_process)),  # strings to token integer counts
             ('tfidf', TfidfTransformer()),  # integer counts to weighted TF-IDF scores
             ('classifier', MultinomialNB()),  # train on TF-IDF vectors w/ Naive Bayes classifier
             ])
             , X=messages["message"], y=messages["label"], train_sizes=train_sizes, cv=5,scoring="accuracy")
```

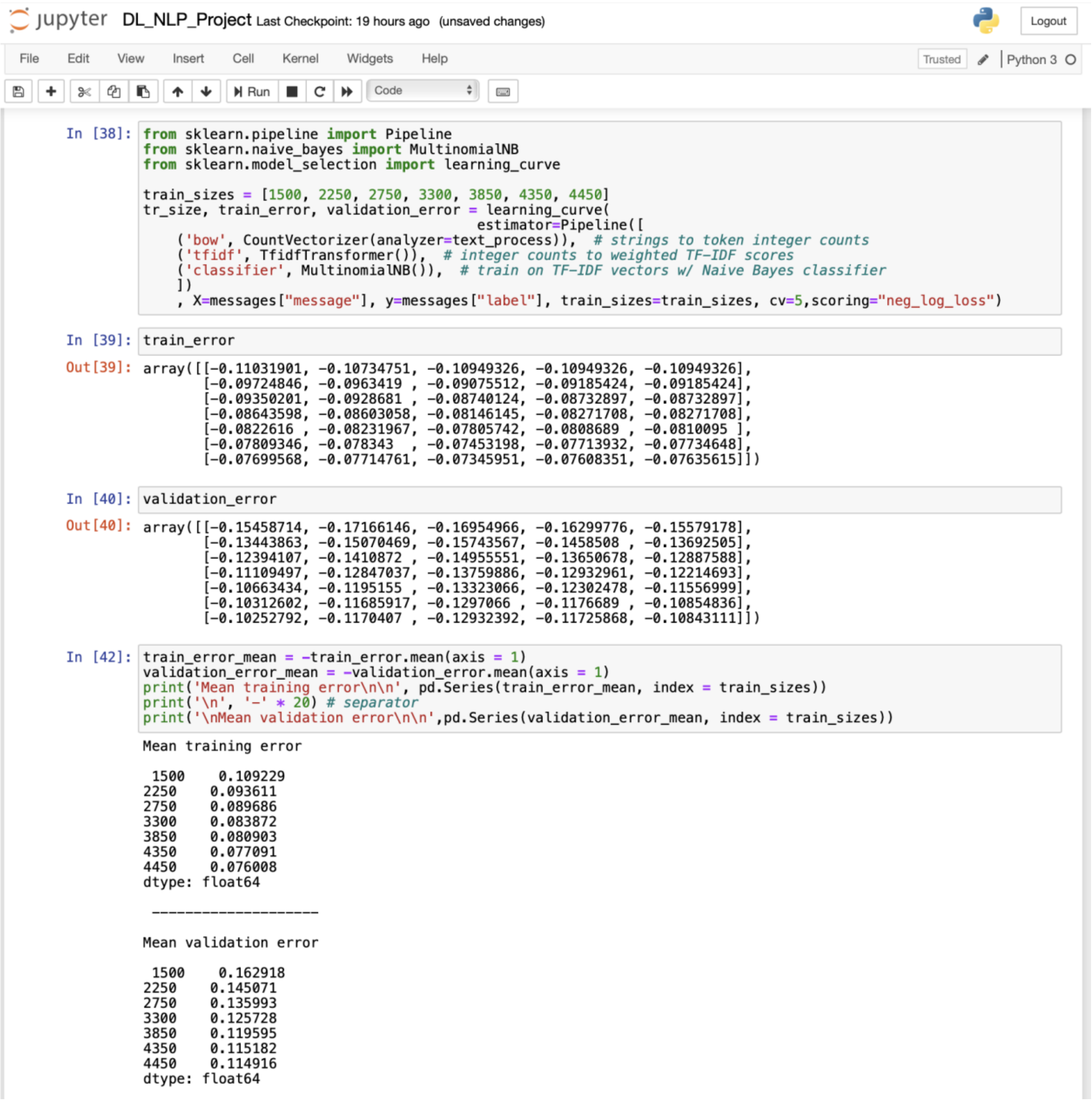
```
In [47]: train_scores

Out[47]: array([[0.964     , 0.96866667, 0.96466667, 0.96466667, 0.96466667],
                [0.96977778, 0.97022222, 0.97422222, 0.97333333, 0.97333333],
                [0.97054545, 0.972     , 0.976     , 0.976     , 0.976     ],
                [0.97484848, 0.97454545, 0.97787879, 0.97484848, 0.97484848],
                [0.97636364, 0.97636364, 0.97818182, 0.9774026 , 0.97506494],
                [0.97908046, 0.97770115, 0.97908046, 0.97862069, 0.97747126],
                [0.97932584, 0.97820225, 0.97955056, 0.97865169, 0.97752809]])
```

```
In [48]: validation_scores

Out[48]: array([[0.93273543, 0.93273543, 0.93536804, 0.92908438, 0.93716338],
                [0.9470852 , 0.94170404, 0.94344704, 0.93985637, 0.94883303],
                [0.95426009, 0.95067265, 0.9524237 , 0.94614004, 0.95421903],
                [0.9632287 , 0.95515695, 0.95780969, 0.95062837, 0.95601436],
                [0.96502242, 0.95695067, 0.95870736, 0.95332136, 0.95960503],
                [0.96591928, 0.95695067, 0.96050269, 0.95332136, 0.96229803],
                [0.96591928, 0.95695067, 0.95960503, 0.95332136, 0.96319569]])
```

```
In [49]: train_scores_mean = train_scores.mean(axis = 1)
         validation_scores_mean = validation_scores.mean(axis = 1)
         print('Mean training scores\n\n', pd.Series(train_scores_mean, index = train_sizes))
         print('\n', '-' * 20) # separator
         print('\nMean validation scores\n\n',pd.Series(validation_scores_mean, index = train_sizes))

         Mean training scores

          1500    0.965333
         2250    0.972178
         2750    0.974109
         3300    0.975394
         3850    0.976675
         4350    0.978391
         4450    0.978652
         dtype: float64

          --------------------

         Mean validation scores

          1500    0.933417
         2250    0.944185
         2750    0.951543
         3300    0.956568
         3850    0.958721
         4350    0.959798
         4450    0.959798
         dtype: float64
```

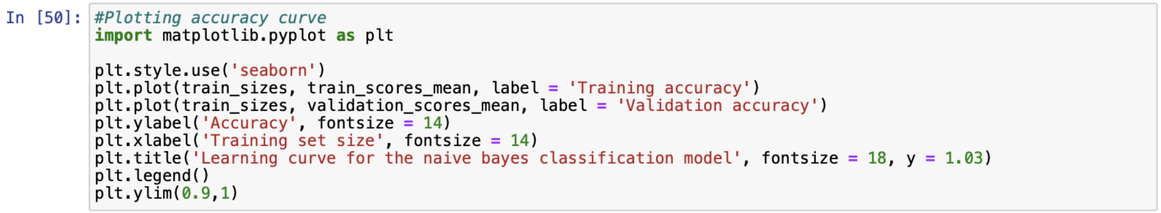
# Use Sklearn’s learning_curve function to plot the log_loss_curve for different train-test split sizes (considering 50:50, 60:40, 70:30, 80:20) with a cross validation K-Fold value of 5 and the average of the five subordinate accuracies is the overall log loss for that train size.

# Scoring parameter here is given as “neg_log_loss” as  learning_curve function tries to maximise the scoring parameter.


jupyter DL_NLP_Project Last Checkpoint: 19 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

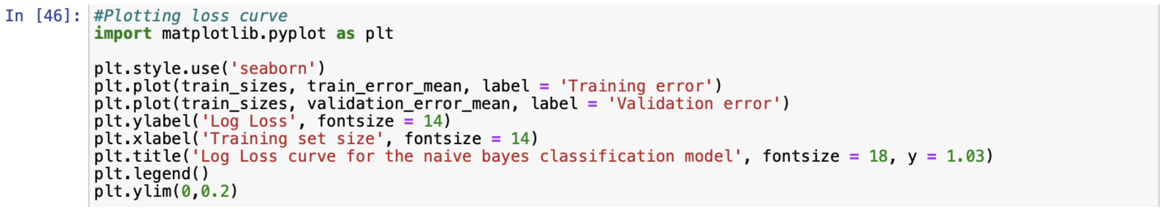
```
In [38]: from sklearn.pipeline import Pipeline
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.model_selection import learning_curve

         train_sizes = [1500, 2250, 2750, 3300, 3850, 4350, 4450]
         tr_size, train_error, validation_error = learning_curve(
                                                 estimator=Pipeline([
             ('bow', CountVectorizer(analyzer=text_process)),  # strings to token integer counts
             ('tfidf', TfidfTransformer()),  # integer counts to weighted TF-IDF scores
             ('classifier', MultinomialNB()),  # train on TF-IDF vectors w/ Naive Bayes classifier
             ])
             , X=messages["message"], y=messages["label"], train_sizes=train_sizes, cv=5,scoring="neg_log_loss")

In [39]: train_error

Out[39]: array([[-0.11031901, -0.10734751, -0.10949326, -0.10949326, -0.10949326],
                [-0.09724846, -0.0963419 , -0.09075512, -0.09185424, -0.09185424],
                [-0.09350201, -0.0928681 , -0.08740124, -0.08732897, -0.08732897],
                [-0.08643598, -0.08603058, -0.08146145, -0.08271708, -0.08271708],
                [-0.0822616 , -0.08231967, -0.07805742, -0.0808689 , -0.0810095 ],
                [-0.07809346, -0.078343  , -0.07453198, -0.07713932, -0.07734648],
                [-0.07699568, -0.07714761, -0.07345951, -0.07608351, -0.07635615]])

In [40]: validation_error

Out[40]: array([[-0.15458714, -0.17166146, -0.16954966, -0.16299776, -0.15579178],
                [-0.13443863, -0.15070469, -0.15743567, -0.1458508 , -0.13692505],
                [-0.12394107, -0.1410872 , -0.14955551, -0.13650678, -0.12887588],
                [-0.11109497, -0.12847037, -0.13759886, -0.12932961, -0.12214693],
                [-0.10663434, -0.1195155 , -0.13323066, -0.12302478, -0.11556999],
                [-0.10312602, -0.11685917, -0.1297066 , -0.1176689 , -0.10854836],
                [-0.10252792, -0.1170407 , -0.12932392, -0.11725868, -0.10843111]])

In [42]: train_error_mean = -train_error.mean(axis = 1)
         validation_error_mean = -validation_error.mean(axis = 1)
         print('Mean training error\n\n', pd.Series(train_error_mean, index = train_sizes))
         print('\n', '-' * 20) # separator
         print('\nMean validation error\n\n',pd.Series(validation_error_mean, index = train_sizes))

         Mean training error

          1500    0.109229
         2250    0.093611
         2750    0.089686
         3300    0.083872
         3850    0.080903
         4350    0.077091
         4450    0.076008
         dtype: float64

          --------------------

         Mean validation error

          1500    0.162918
         2250    0.145071
         2750    0.135993
         3300    0.125728
         3850    0.119595
         4350    0.115182
         4450    0.114916
         dtype: float64
```

Plotting Accuracy Curve:



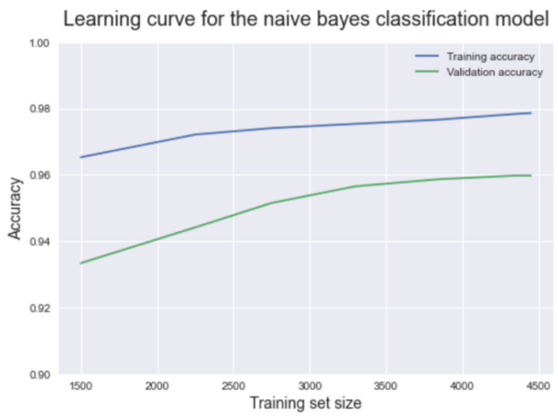
```
In [50]: #Plotting accuracy curve
         import matplotlib.pyplot as plt

         plt.style.use('seaborn')
         plt.plot(train_sizes, train_scores_mean, label = 'Training accuracy')
         plt.plot(train_sizes, validation_scores_mean, label = 'Validation accuracy')
         plt.ylabel('Accuracy', fontsize = 14)
         plt.xlabel('Training set size', fontsize = 14)
         plt.title('Learning curve for the naive bayes classification model', fontsize = 18, y = 1.03)
         plt.legend()
         plt.ylim(0.9,1)
```

Plotting Log Loss Curve:

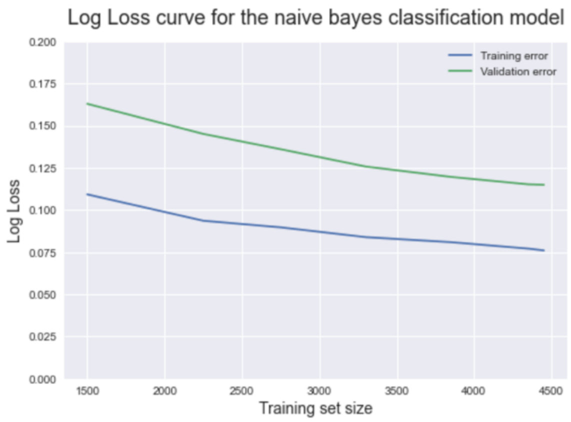


```
In [46]: #Plotting loss curve
         import matplotlib.pyplot as plt

         plt.style.use('seaborn')
         plt.plot(train_sizes, train_error_mean, label = 'Training error')
         plt.plot(train_sizes, validation_error_mean, label = 'Validation error')
         plt.ylabel('Log Loss', fontsize = 14)
         plt.xlabel('Training set size', fontsize = 14)
         plt.title('Log Loss curve for the naive bayes classification model', fontsize = 18, y = 1.03)
         plt.legend()
         plt.ylim(0,0.2)
```

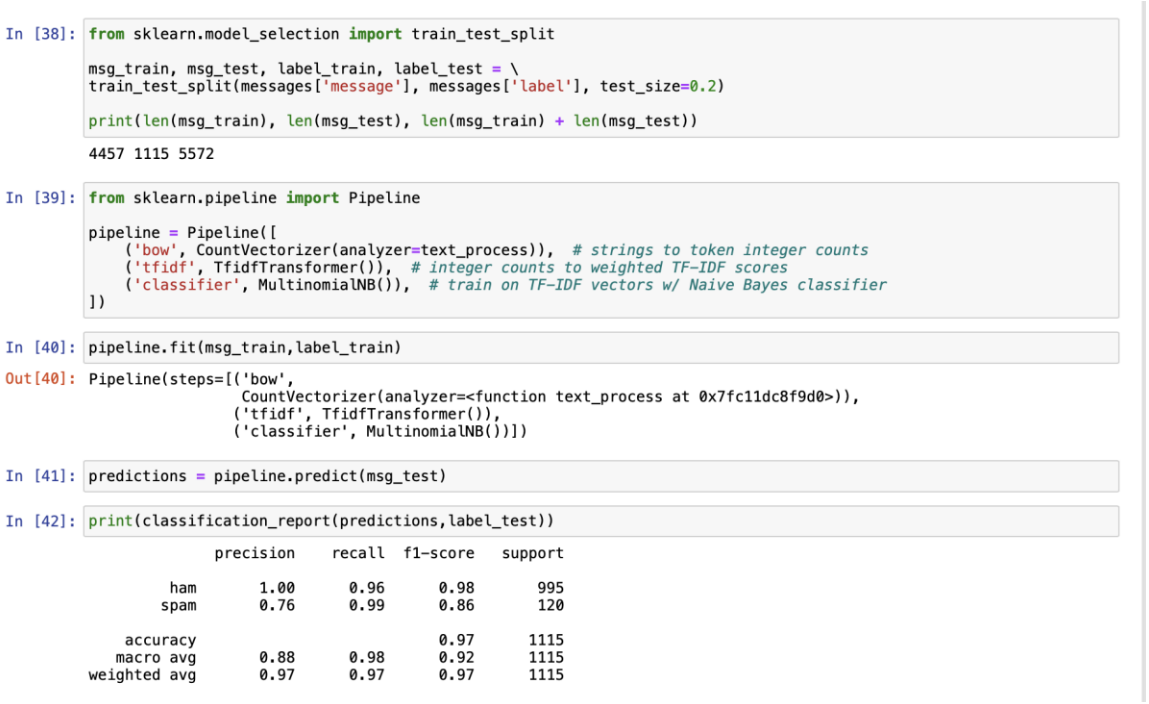
Accuracy Graph:



Loss Graph:



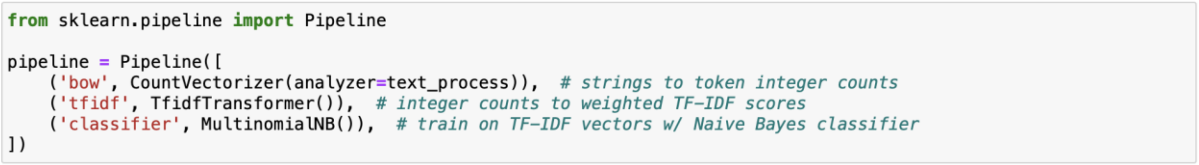
# Training and fitting the final model (Pipeline) on the cleaned dataset (With the best hyperparameters and the most optimal train-test split found above).



```
In [38]: from sklearn.model_selection import train_test_split

         msg_train, msg_test, label_train, label_test = \
         train_test_split(messages['message'], messages['label'], test_size=0.2)

         print(len(msg_train), len(msg_test), len(msg_train) + len(msg_test))

         4457 1115 5572

In [39]: from sklearn.pipeline import Pipeline

         pipeline = Pipeline([
             ('bow', CountVectorizer(analyzer=text_process)),  # strings to token integer counts
             ('tfidf', TfidfTransformer()),  # integer counts to weighted TF-IDF scores
             ('classifier', MultinomialNB()),  # train on TF-IDF vectors w/ Naive Bayes classifier
         ])

In [40]: pipeline.fit(msg_train,label_train)

Out[40]: Pipeline(steps=[('bow',
                          CountVectorizer(analyzer=<function text_process at 0x7fc11dc8f9d0>)),
                         ('tfidf', TfidfTransformer()),
                         ('classifier', MultinomialNB())])

In [41]: predictions = pipeline.predict(msg_test)

In [42]: print(classification_report(predictions,label_test))

                       precision    recall  f1-score   support

                  ham       1.00      0.96      0.98       995
                 spam       0.76      0.99      0.86       120

             accuracy                           0.97      1115
            macro avg       0.88      0.98      0.92      1115
         weighted avg       0.97      0.97      0.97      1115
```

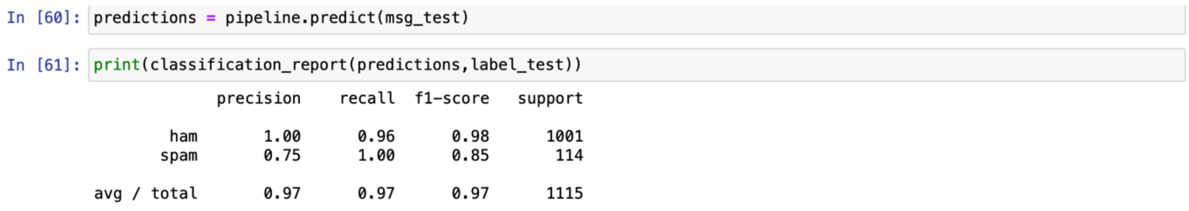
**Methodology:**

- Import dataset using pandas
- Exploratory data analysis to check how balanced the dataset actually is (Group by label).
- Add additional length column as a new text attribute for the final MultinomialNB ML model.
- Text pre-processing by removing punctuations, stopwords, etc.
- Tokenization, Normalization followed by Vectorization using CountVectorizer to create a Bag of Words representation of the data.
- Pass through TF-IDF transformer to get the new updated representation.
- Train test split the dataset using Sklearn Model Selection package.
- Create an Sklearn data pipeline and perform the above mentioned steps.



```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process)),  # strings to token integer counts
    ('tfidf', TfidfTransformer()),  # integer counts to weighted TF-IDF scores
    ('classifier', MultinomialNB()),  # train on TF-IDF vectors w/ Naive Bayes classifier
])
```

- Fit the pipeline on the cleaned train dataset and transform the cleaned output dataset using the same trained pipeline.
- Check the overall final classification report.



```
In [60]: predictions = pipeline.predict(msg_test)

In [61]: print(classification_report(predictions,label_test))
                      precision    recall  f1-score   support

                 ham       1.00      0.96      0.98      1001
                spam       0.75      1.00      0.85       114

         avg / total       0.97      0.97      0.97      1115
```

## Result Metrics:

**Final Accuracy:** 97%
**Final Recall:** 97%
**Final F1-Score:** 97%
**Final Support:** 1115

## Conclusion:

The final model achieves an overall test accuracy of 97 % and a recall of 97% which is really good considering the naturality of the real world dataset that has been considered.

## References:

[1] https://www.nltk.org/api/nltk.html

[2] https://en.wikipedia.org/wiki/Natural_language_processing

[3] https://www.ibm.com/cloud/learn/natural-language-processing

[4] Deep Learning With Python. Manning Publications Co., 3 Lewis Street Greenwich, CT, United States

[5] https://www.sas.com/en_in/insights/analytics/what-is-natural-language-processing-nlp.html