# Automated Tagging of Stack Exchange Data Science Posts Using Natural Language Processing

A Capstone Project in Fulfillment of
Springboard's Data Science Career Track Program

By

## Laura Elliott

And In Association with

## Ajith Patnaik

Springboard Mentor

# Table of Contents

# 1. ABSTRACT / PROBLEM STATEMENT

For many question answering websites (Stack Overflow, StackExchange, Quora, Answerbag, Yahoo, WikiAnswers, etc.), a plethora of questions on many different subjects are submitted each day. Each question requires tagging with words or phrases to provide context for and to categorize that question. These tags are an important ranking factor in most of these website's search algorithms, so it is important for them to be as accurate as possible in order to ensure the best customer experience. The more quickly and accurately the tags can be assigned, the better the service can be about sending the right question to the right set of people for answering and the quicker the turnaround time for an answer to that question. The tags also provide the users with the ability to sort and filter through questions and answers on their topic of concern. Therefore automated tagging has become a necessity among these websites. The objective of this capstone is to build an algorithm to predict the tags using the text in the questions themselves.

The client could be a StackExchange (or other question answering entity) site developer who is looking for a faster and more accurate way to automatically tag questions. Based on the results of this analysis, the client may decide to deploy this new algorithm for auto-tagging questions in order to improve the efficiency of his team and the customers' experience.

## Data Description

Questions and tags from the Data Science Stack Exchange database were used for this analysis. The Data Science Stack Exchange website is a question and answer site for Data Science professionals, Machine Learning specialists, and those interested in learning more about the field. It is one of 173 Stack Exchange Q&A communities (which includes the largest and most well-known site for programmers - Stack Overflow). The website link is:

https://datascience.stackexchange.com/

The dataset was extracted from the Data Science Stack Exchange database on July 22, 2020 via the Stack Exchange data dump site using this link:

https://archive.org/details/stackexchange

These data were archived on May 31, 2020, so contain information up to that point in time.

The data is formatted as a separate archive consisting of XML files zipped via 7-zip using bzip2 compression. Both the datascience.meta.stackexchange.com.7z and datascience.stackexchange.com.7z files were downloaded and extracted. The archive includes Badges, Comments, Posts, PostHistory, PostLinks, Tags, Users and Votes. The entire data schema is provided in Appendix 1.

For preliminary examination of the xml files, a free application called XML ValidatorBuddy was used to view the file contents and structure. This application was particularly helpful for

reviewing the larger xml files which could not be opened with other text readers due to size limitations.

The Tags.xml file in the meta archive was quite useful in evaluating the distribution and number of tags. The Posts.xml (79 MB) file contained within the datascience.stackexchange.com.7z contained all the necessary information for the analysis. Therefore, no joining of separate xml files was required.

The following features from the Posts.xml file were loaded for the analysis.

| Features Name | Data Type | Description |
|---|---|---|
| Id | int | unique post identifier |
| PostTypeId | tinyint | 1 = question; 2 = answer |
| CreationDate | datetime | Date of post |
| Score | int | sum of upvotes - downvotes |
| ViewCount | int | count of number of views |
| Body | nvarchar | question / answer body text |
| OwnerUserId | int | id of owner/user |
| LastActivityDate | datetime | date of last post activity |
| Title | nvarchar | brief description of question |
| Tags | nvarchar | comma-separated list of the tags associated with the question |
| AnswerCount | int | count of the number of answers |
| CommentCount | int | count of the number of comments |
| FavoriteCount | int | count of the number the post has been marked as a favorite |
| ClosedDate | datetime | date the post was closed |
| ContentLicense | nvarchar | description of content license |

Table 1: Feature Description from the Posts.xml file

Data Overview:

The title feature was not included in the analysis.

## 2. DATA WRANGLING

The following sections outline the steps involved in the data wrangling process.

### Conversion of XML file to Pandas DataFrame

The initial process of converting the Posts.xml file into a pandas dataframe was conducted in the "CP2_01_XML to DF.ipynb" Jupyter Notebook. The following pseudocode outlines the process:

1. Load and Parse the Posts.xml file using the xml.dom.minidom package.
2. Test whether the parsing worked by getting a sample of items from the file.
3. Create a header row from the items in each row of the xml; `df_cols = ['Id','PostTypeId','CreationDate','Score','ViewCount','Body','OwnerUserId','LastActivityDate','Title','Tags','AnswerCount','CommentCount','FavoriteCount','ClosedDate','ContentLicense']`
4. Create an empty list to append data to.
5. Write a for loop to extract all items from each row of the xml making sure to write an N/A for those fields that are empty.
6. Create a pandas dataframe with the extracted data and headers.
7. QC the dataframe and export as a pickle file.

Input is the original Posts.xml file from the Stack Exchange data dump. Output is a pandas dataframe called posts_df and associated pickle file called posts_df_09112020.pickle. The resulting posts_df dataframe consisted of 51395 rows and 15 columns (features listed in the table above).

For this notebook, only 3 libraries were used: xml.dom.minidom, pandas, and pickle.

### Data Conditioning, FIltering, and Cleaning

Additional conditioning and cleaning steps were performed in the "CP2_02_Load DF Clean and Analyze.ipynb" Jupyter notebook. The following pseudocode outlines the preliminary filtering and cleaning steps :

1. Load the posts_df_09112020.pickle file created in the first Jupyter Notebook.
2. Examine contents to assure everything transferred properly.
   a. Identified some cells filled with only white space. These were replaced with nans.
3. Filter down to just questions (PostTypeId = '1')
4. Convert objects to appropriate integer and date dtypes (text attributes Body, Title, and Tag were converted in later steps).
5. Check for missing values.
6. Check for and remove duplicates.
7. Preprocess tags - remove html tags and convert to string.
8. ...

Input is the pandas dataframe called posts_df and associated pickle file called posts_df_09112020.pickle from the first Jupyter Notebook.

### Check of Missing Data

The following bar chart and table describe the number of missing values in each column.  The chart is shown as a ratio of missing to total values per column.
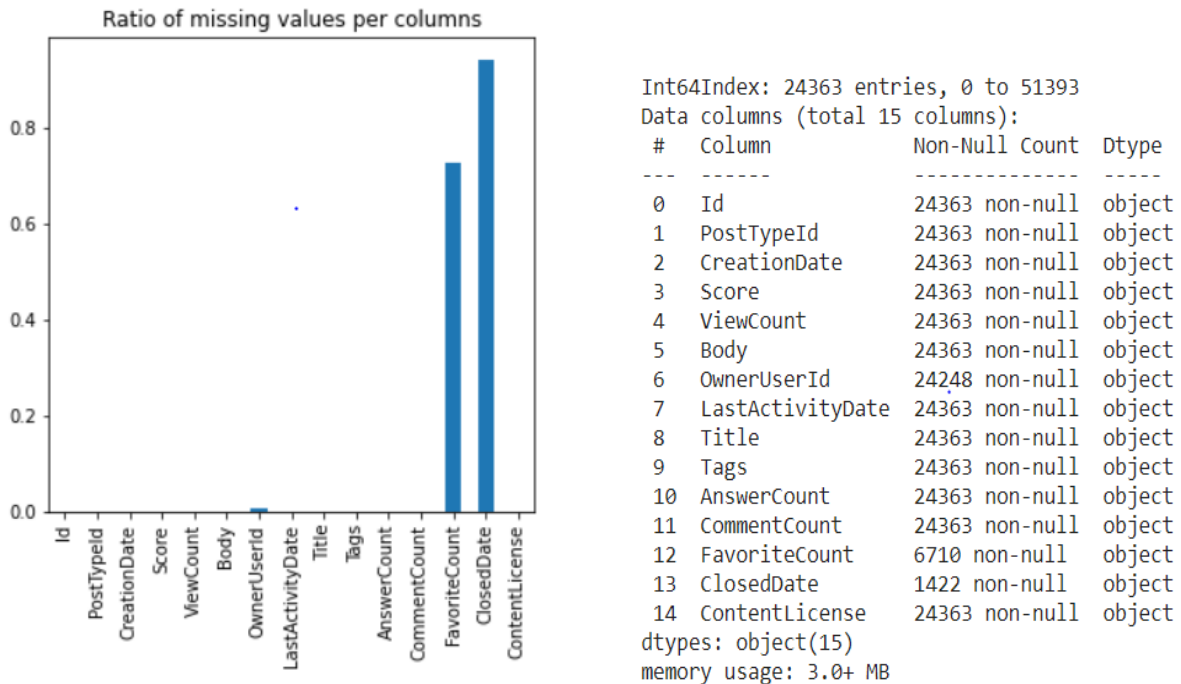


```
Int64Index: 24363 entries, 0 to 51393
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              24363 non-null  object
 1   PostTypeId      24363 non-null  object
 2   CreationDate    24363 non-null  object
 3   Score           24363 non-null  object
 4   ViewCount       24363 non-null  object
 5   Body            24363 non-null  object
 6   OwnerUserId     24248 non-null  object
 7   LastActivityDate 24363 non-null object
 8   Title           24363 non-null  object
 9   Tags            24363 non-null  object
 10  AnswerCount     24363 non-null  object
 11  CommentCount    24363 non-null  object
 12  FavoriteCount   6710 non-null   object
 13  ClosedDate      1422 non-null   object
 14  ContentLicense  24363 non-null  object
dtypes: object(15)
memory usage: 3.0+ MB
```

*Fig 1: Ratio of missing to total values in each column (plotted with matplotlib) and results of dataframe.info query providing a count of non-null values.*

The results indicate that the critical columns have no missing values.  Only the OwnerUserId, FavoriteCount and ClosedDate columns have missing values.  The missing values in the latter 2 features are expected, since not all question posts will be identified as having a Favorite answer and not all will be Closed.  The OwnerUserId has a few missing values, but was not used in the analysis.

### Check of Duplicates

The dataframe was checked for duplicates of both the question (Body) and Title and 10 duplicate entries were found.  Two of the duplications  are shown below.

| Id | PostTypeId | CreationDate | Score | ViewCount | Body | OwnerUserId | LastActivityDate | Title | Tags | AnswerCount | CommentCount | FavoriteCount | ClosedDate | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 461 | 1 | 2014-06-18 22:10:58.497 | 12 | 1336 | \<p>There's this side project I'm working on wh... | 986 | 2017-11-06 08:07:09.260 | Preference Matching Algorithm | \<bigdata>\<text-mining> \<recommender-system> | 2 | 0 | 2 | NaT | |
| 477 | 1 | 2014-06-18 22:15:43.820 | 2 | 561 | \<p>There's this side project I'm working on wh... | 986 | 2014-06-19 10:30:43.993 | Preference Matching Algorithm | \<algorithms> | 1 | 3 | NaN | 2014-06-26 05:31:41.193 | |
| 2313 | 1 | 2014-10-20 06:38:16.490 | 5 | 1156 | \<p>Where is the difference between one-class, ... | 4717 | 2015-03-03 14:21:51.033 | Machine Learning - Where is the difference bet... | \<machine-learning>\<data-mining> \<classification... | 2 | 2 | 5 | NaT | |
| 2316 | 1 | 2014-10-19 21:03:35.390 | 1 | 47 | \<p>Where is the difference between one-class, ... | 4717 | 2014-10-20 10:38:59.437 | Machine Learning - Where is the difference bet... | \<machine-learning> \<classification> \<data-mining> | 0 | 1 | NaN | 2014-10-21 14:09:06.617 | |

*Fig 2: Partial results of check for duplication in Title and Body.*

The results illustrate that these duplicates were created by the same owner but have slightly different creation dates and only one of the grouped pairs has a closed date. Additionally Score, LastActivityDate, Tags, and ViewCount, AnswerCount, CommentCount, and FavoriteCount are different. For this reason, the second entry with lower counts and closed dates were removed. Removal of these duplicates reduced the total row count of the dataframe from 24363 to 24353 entries.

### *Preprocess Tags*

As noted in Figure 2 above, each of the tags in the Tags column are separated by html <> characters. These were removed and the text was converted to a space delimited string in preparation for further analysis.

## Preprocess Body (Questions)

Many cleaning steps were required to condition the question text ('Body' attribute) for analysis. These steps were performed in a third Jupyter notebook "CP2_03_Clean_Body_Text_Analyze.ipynb".

The following pseudocode outlines the steps taken:

1. Load the pickled pandas dataframe from 02 notebook
2. Examine the feature 'Body' (which are the questions) to identify cleaning tasks
3. ...
4. ...
5. Cleaning tasks:
    a. Remove code snippets
    b. Remove html formatting
    c. Expand contractions
    d. Language detection to make sure everything is in English
    e. Remove special characters
    f. Simple lemmatization
    g. Named Entity Recognition
    h. Part Of Speech Tagging
    i. Convert to lowercase
    j. Remove task remove stop words
6. Export the dataframe with cleaned text and additional features for further analysis

Input is the pandas dataframe called questions_df and associated pickle file called questions_df_09252020.pickle from the second Jupyter Notebook.

For these steps, the following libraries were used: collections, string, textwrap, unicodedata, contractions, fasttext, spacy, nltk.

To identify cleaning tasks, a sampling of the questions were printed for examination. Figure 3 shows a few examples.

```
HighestRank ********************************************************************************************************

Question id: 410
Rank : 3
Question Tags   : machine-learning neural-network deep-learning optimization hyperparameter
Question Title  : Choosing a learning rate
Question Body   :

<p>I'm currently working on implementing Stochastic Gradient Descent, <code>SGD</code>, for neural nets using back-propagation, and while I understand its
purpose I have some questions about how to choose values for the learning rate.</p>  <ul>  <li>Is the learning rate related to the shape of the error gradient,
as it dictates the rate of descent?</li>  <li>If so, how do you use this information to inform your decision about a value?</li>  <li>If it's not what sort of
values should I choose, and how should I choose them?</li>  <li>It seems like you would want small values to avoid overshooting, but how do you choose one such
that you don't get stuck in local minima or take to long to descend?</li>  <li>Does it make sense to have a constant learning rate, or should I use some metric
to alter its value as I get nearer a minimum in the gradient?</li>  </ul>  <p>In short: How do I choose the learning rate for SGD?</p>

LowestRank ********************************************************************************************************

Question id: 73375
Rank: 24348
Question Tags   : time

Question Title  : Getting a constant accuracy for a ping-like command

Question Body   :

<p>very simple and naïve question here, I'm trying to mesure some RTTs with a reasonnable accuracy. The Ubuntu ping command provides a pretty good mesure with a
10µs accuracy. but only when the total RTT time is under 10ms. </p>  <p>Actually, it seems that the output time is constantly given with 3 digits (e.g. 6.34 ms
; 17.3 ms ; 137 ms).</p>  <p>I would like to keep the 10µs accuracy whatever the output value.</p>  <p>Does anyone knows if such an option is available with the
command ping, or if there exist another tool which will allow me to get what I want.</p>  <p>Thanks in advance, and have a great day.</p>  <p>PS : I'm not a
native english speaker so i may have made some grammatical mistakes, sincere apologies for that.</p>

HighestRank ********************************************************************************************************

Question id: 761
Rank : 7
Question Tags   : machine-learning python clustering k-means geospatial
Question Title  : Clustering geo location coordinates (lat,long pairs)
Question Body   :

<p>What is the right approach and clustering algorithm for geolocation clustering?</p>  <p>I'm using the following code to cluster geolocation coordinates:</p>
<pre><code>import numpy as np import matplotlib.pyplot as plt from scipy.cluster.vq import kmeans2, whiten  coordinates= np.array([          [lat, long],
[lat, long],          ...          [lat, long]   ]) x, y = kmeans2(whiten(coordinates), 3, iter = 20)   plt.scatter(coordinates[:,0],
coordinates[:,1], c=y); plt.show() </code></pre>  <p>Is it right to use K-means for geolocation clustering, as it uses Euclidean distance, and not <a
href="https://en.wikipedia.org/wiki/Haversine_formula" rel="noreferrer">Haversine formula</a> as a distance function?</p>
```

*Fig 3: Question examples showing the raw question "Body" text, with various text highlighted to show important items identified for cleaning. Text was formatted using 'textwrap' for increased readability.*

The first thing that stands out as an issue requiring cleaning is the html formatting, such as the <p> </p> and <li> </li> pairs that denote paragraph and line formatting. In some cases, text is bolded for emphasis. There are also hyperlinks and code snippets that tend to clutter the overall question but are provided as examples or reference. Standard special character removal and contraction expansion is also required.

Note in the examples shown that oftentimes more than one question is asked within the Body or sometimes no questions is asked at all. Question length can be quite variable also. Therefore, prior to performing the cleaning steps, certain features were extracted from the Body (such as the number of code snippets, number of question marks, body text length, and number of bolded text items).

It appears that acronyms like LDA, PCA, SGD, RTT are important to recognize, which indicates that we should attempt a Named Entity Recognition and Part of Speech tagging process as a part of the text processing.

Some misspellings were also identified, but have been kept in place. Some examples are shown above. In one case, "but" was misspelled as "bot", the latter being an important word with a very different meaning in the data science world.

Beautiful Soup was then used to identify and remove the code snippets  and then to remove the remaining html formats.  Figure 4 shows an example of a question before and after this process.

```
Question Title  : How to do numpy matmul broadcasting between two numpy tensors?

Question Body   :

<p>I have the Pauli matrices which are (2x2) and complex</p>  <pre class="lang-py prettyprint-override"><code>II = np.identity(2, dtype=complex) X =
np.array([[0, 1], [1, 0]], dtype=complex) Y = np.array([[0, -1j], [1j, 0]], dtype=complex) Z = np.array([[1, 0], [0, -1]], dtype=complex) </code></pre>  <p>and
a <code>depolarizing_error</code> function which takes in a normally distributed random number <code>param</code>, generated by
<code>np.random.normal(noise_mean, noise_sd)</code></p>  <pre class="lang-py prettyprint-override"><code>def depolarizing_error(param):    XYZ =
np.sqrt(param/3)*np.array([X, Y, Z])    return np.array([np.sqrt(1-param)*II, XYZ[0], XYZ[1], XYZ[2]]) </code></pre>  <p>Now if I feed in a single number for
<code>param</code> of let's say <code>a</code>, my function should return an output of <code>np.array([np.sqrt(1-a)*II, a*X, a*Y, a*Z])</code> where
<code>a</code> is a <code>float</code> and <code>*</code> denotes the element-wise multiplication between <code>a</code> and the entries of the (2x2) matrices
<code>II, X, Y, Z</code>. Now for vectorization purposes, I wish to feed in an array of <code>param</code> i.e.</p>  <pre class="lang-py prettyprint-
override"><code>param = np.array([a, b, c, ..., n])    Eqn(1) </code></pre>  <p>again with all <code>a, b, c, ..., n</code> generated independently by
<code>np.random.normal(noise_mean, noise_sd, n)</code> (I think it's doable with <code>np.random.normal(noise_mean, noise_sd, n)</code> or something) such that my
function now returns:</p>  <pre class="lang-py prettyprint-override"><code>np.array([[np.sqrt(1-a)*II, a*X, a*Y, a*Z],        [np.sqrt(1-b)*II, b*X, b*Y,
b*Z],        ..............................,        [np.sqrt(1-n)*II, n*X, n*Y, n*Z]]) </code></pre>  <p>I thought feeding in something like
<code>np.random.normal(noise_mean, noise_sd, n)</code> as <code>param</code>, giving output as <code>np.array([a, b, c,...,n])</code> would sort itself out and
return what I want above. but my <code>XYZ = np.sqrt(param/3)*np.array([X, Y, Z])</code> ended up doing element-wise dot product instead of element-wise
multiplication. I tried using param as <code>np.array([a, b])</code> and ended up with </p>  <pre><code>np.array([np.dot(np.sqrt(1-[a, b]), II),
np.dot(np.sqrt([a, b]/3),  X),        np.dot(np.sqrt([a, b]/3),  Y),        np.dot(np.sqrt([a, b]/3),  Z)]) </code></pre>  <p>instead. So far I've tried
something like</p>  <pre><code>def depolarizing_error(param):      XYZ = np.sqrt(param/3)@np.array([X, Y, Z])     return np.array([np.sqrt(1-param)*II, XYZ[0],
XYZ[1], XYZ[2]]) </code></pre>  <p>thinking that the matmul @ will just broadcast it conveniently for me bot then I got really bogged down by the
dimensions.</p>  <p>Now my motivation for wanting to do all this is because I have another matrix that's given by:</p>  <pre><code>def random_angles(sd,
seq_length):    return np.random.normal(0, sd, (seq_length,3))  def unitary_error(params):    e_1 =
np.exp(-1j*(params[:,0]+params[:,2])/2)*np.cos(params[:,1]/2)    e_2 = np.exp(-1j*(params[:,0]-params[:,2])/2)*np.sin(params[:,1]/2)    return np.array([[e_1,
e_2], [-e_2.conj(), e_1.conj()]],        dtype=complex).transpose(2,0,1) </code></pre>  <p>where here the size of <code>seq_length</code> is
equivalent to the number of entries in Eqn(1) <code>param</code>, denoting <code>N = seq_length = |param|</code> say. Here my <code>unitary_error</code>
function should give me an output of </p>  <pre><code>np.array([V_1, V_2, ..., V_N]) </code></pre>  <p>such that I'll be able to use <code>np.matmul</code> as
an attempt to implement vectorization like this</p>  <pre><code>np.array([V_1, V_2, ..., V_N])@np.array([[np.sqrt(1-a)*II, a*X, a*Y, a*Z],
[np.sqrt(1-b)*II, b*X, b*Y, b*Z],
[np.sqrt(1-n)*II, n*X, n*Y, n*Z]])@np.array([V_1, V_2, ..., V_N]) </code></pre>  <p>to finally give</p>  <pre><code>np.array([[V_1@np.sqrt(1-a)*II@V_1,
V_1@a*X@V_1, V_1@a*Y@V_1, V_1@a*Z@V_1],        [V_2@np.sqrt(1-b)*II@V_2, V_2@b*X@V_2, V_2@b*Y@V_2, V_2@b*Z@V_2],        ...............................,
[V_N@np.sqrt(1-n)*II@V_N, V_N@n*X@V_N, V_N@n*Y@V_N, V_N@n*Z@V_N]]) </code></pre>  <p>where here <code>@</code> denotes the element-wise dot-product</p>
************************************************************************************************************************************************

Question Body WO Code   :

I have the Pauli matrices which are (2x2) and complex  and a  function which takes in a normally distributed random number , generated by   Now if I feed in a
single number for  of let's say , my function should return an output of  where  is a  and  denotes the element-wise multiplication between  and the entries of
the (2x2) matrices . Now for vectorization purposes, I wish to feed in an array of  i.e.  again with all  generated independently by  (I think it's doable with
or something) such that my function now returns:  I thought feeding in something like  as , giving output as  would sort itself out and return what I want
above. but my  ended up doing element-wise dot product instead of element-wise multiplication. I tried using param as  and ended up with   instead. So far I've
tried something like  thinking that the matmul @ will just broadcast it conveniently for me but then I got really bogged down by the dimensions. Now my
motivation for wanting to do all this is because I have another matrix that's given by:  where here the size of  is equivalent to the number of entries in
Eqn(1) , denoting  say. Here my  function should give me an output of   such that I'll be able to use  as an attempt to implement vectorization like this  to
finally give  where here  denotes the element-wise dot-product
************************************************************************************************************************************************
```

*Fig 4: Example question before and after removal of code snippets and html formatting.*

The next step was to expand contractions, using the contractions module. Figure 5 shows a before and after example of this process.

```
Before:  As a researcher and instructor, I'm looking for open-source books (or similar materials) that provide a relatively thorough overview of data science from an
applied perspective. To be clear, I'm especially interested in a thorough overview that provides material suitable for a college-level course, not particular
pieces or papers.

After:  As a researcher and instructor, I am looking for open-source books (or similar materials) that provide a relatively thorough overview of data science from an
applied perspective. To be clear, I am especially interested in a thorough overview that provides material suitable for a college-level course, not particular
pieces or papers.
```

*Fig 5: Example question before and after contraction expansion.*

The next step was a check to see if all questions are in English, using Facebook's fasttext library and their prebuilt model (lid.176.bin).  Only 3 questions returned as a different language (fr,ja,kn) but a review of these rows indicates that the processing of the questions done in previous steps reduced them to just one word or character, and were therefore producing erroneous language results. A preliminary pass of the English language detection routine prior to cleaning the code snippets identified some string examples within the text that were in an alternate language, but these were eliminated by first removing the code snippets.  To summarize, all text from the Stack Exchange data dump is in English, although some examples

provided within the question code snippets were in a different language. These examples were all removed as a part of the code snippet removal process.

The next step was to remove all special characters in the text. In order to identify what characters should be removed, further analysis was performed using the nltk dictionary 'punct_only'. There are 329 unique special characters in the Body text. Most of the special characters are punctuation and symbols and these were removed:

!"#$%&\'()*+,./:;-<=>?@[\\]^_`{|}~

However Greek characters such as α, β, θ, σ, $\lambda$,Ψ, and μ were maintained because they have special significance in the data science field.

NLTK's WordNetLemmatizer was then used to perform a simple lemmatization of the text. Sample results of the special character removal and lemmatizing process are shown in Figure 6 below.

```
Before Special Character Removal:
I have a bunch of customer profiles stored in a elasticsearch cluster. These profiles are now used for creation of target groups for our email subscriptions.
Target groups are now formed manually using elasticsearch faceted search capabilities (like get all male customers of age 23 with one car and 3 children). How
could I search for interesting groups automatically - using data science, machine learning, clustering or something else? r programming language seems to be a
good tool for this task, but I can not form a methodology of such group search. One solution is to somehow find the largest clusters of customers and use them
as target groups, so the question is: How can I automatically choose largest clusters of similar customers (similar by parameters that I do not know at this
moment)? For example: my program will connect to elasticsearch, offload customer data to CSV and using R language script will find that large portion of
customers are male with no children and another large portion of customers have a car and their eye color is brown.

Before Lemmatization:
I have a bunch of customer profiles stored in a elasticsearch cluster These profiles are now used for creation of target groups for our email subscriptions
Target groups are now formed manually using elasticsearch faceted search capabilities like get all male customers of age 23 with one car and 3 children How
could I search for interesting groups automatically  using data science machine learning clustering or something else r programming language seems to be a good
tool for this task but I can not form a methodology of such group search One solution is to somehow find the largest clusters of customers and use them as
target groups so the question is How can I automatically choose largest clusters of similar customers similar by parameters that I do not know at this moment
For example my program will connect to elasticsearch offload customer data to CSV and using R language script will find that large portion of customers are male
with no children and another large portion of customers have a car and their eye color is brown

After:
I have a bunch of customer profile store in a elasticsearch cluster These profile be now use for creation of target group for our email subscription Target
group be now form manually use elasticsearch faceted search capability like get all male customer of age 23 with one car and 3 child How could I search for
interest group automatically use data science machine learn cluster or something else r program language seem to be a good tool for this task but I can not form
a methodology of such group search One solution be to somehow find the large cluster of customer and use them a target group so the question be How can I
automatically choose large cluster of similar customer similar by parameter that I do not know at this moment For example my program will connect to
elasticsearch offload customer data to CSV and use R language script will find that large portion of customer be male with no child and another large portion of
customer have a car and their eye color be brown
```

*Fig 6.   Sample showing questions text before and after special character removal and lemmatizing.*

An attempt was made to recognize, extract, and label named entities using Spacy's Named Entity Recognition and corresponding corpus 'en_core_web_sm'. Running this process, even with this small dataset takes quite a bit of time, and while effective at recognizing and extracting the named entities, the labelling results were poor.  Figure 7 below shows that terms like "Libsvm", "Liblinear", and "MapReduce" are identified erroneously as a Geopolitical Entity (GPE), an Event, and a Person, respectively.  This is a result of using a very general corpus for such a technical dataset.  Building and using a corpus specific to data science terms could yield better results but was deemed beyond the scope of this capstone project.

```
Before:  I use Libsvm to train data and predict classification on semantic analysis problem But it have a performance issue on largescale data because semantic analysis
concern ndimension problem Last year Liblinear be release and it can solve performance bottleneck But it cost too much memory Is MapReduce the only way to solve
semantic analysis problem on big data Or be there any other method that can improve memory bottleneck on Liblinear

After:  [['Libsvm', 'GPE'], ['Last year', 'DATE'], ['Liblinear', 'EVENT'], ['MapReduce', 'PERSON']]
```

*Fig 7.   Sample showing poor labelling results for technical data science terms during the NER process.*

Nouns were also recognized and extracted using NLTK's pos_tag module. The results will be examined and discussed further in the exploratory analysis section of this report.

```
Before:  I use Libsvm to train data and predict classification on semantic analysis problem But it has a performance issue on largescale data because semantic analysis
concerns ndimension problem Last year Liblinear was release and it can solve performance bottleneck But it cost too much memory Is MapReduce the only way to
solve semantic analysis problem on big data Or are there any other methods that can improve memory bottleneck on Liblinear

After:  ['data', 'classification', 'analysis', 'problem', 'performance', 'issue', 'data', 'analysis', 'concerns', 'ndimension', 'problem', 'year', 'Liblinear', 'release', 'performance'
```

*Fig 8. Same sample as in Figure 7 showing results of POS tagging for nouns. Some, but not all, named entities are recognized also through this process.*

The next cleaning step for the question text was to convert all to lowercase. It was determined via experimentation that performing the lowercase step prior to NER and POS tasks produced less satisfying results. The final step of text cleaning was to remove stop words. The following words were removed from the stopwords list since they are meaningful for data scientists - 're', 'r', and 'q'. A before and after sample of the lowercase conversion and stop word removal is shown below.

```
Before LC Conversion:
We create a social network application for eLearning purpose it be an experimental project that we be research on in our lab It have be use in some case study
for a while and the data in our relational DBMS SQL Server 2008 be get big it be a few gigabyte now and the table be highly connect to each other The
performance be still fine but when should we consider other option Is it the matter of performance

Before StopWord Removal:
we create a social network application for elearning purpose it be an experimental project that we be research on in our lab it have be use in some case study
for a while and the data in our relational dbms sql server 2008 be get big it be a few gigabyte now and the table be highly connect to each other the
performance be still fine but when should we consider other option is it the matter of performance

After:  create social network application elearning purpose experimental project research lab use case study data relational dbms sql server 2008 get big gigabyte table
highly connect performance still fine consider option matter performance
```

*Fig 9. Sample showing results of lowercase conversion and stop word removal.*

Final Output is a pandas dataframe called questions_df consisting of 24363 entries and 63 columns. The associated pickle file exported is named questions_df_ner_results_10282020.pickle.

# 3.  EXPLORATORY DATA ANALYSIS

## Tag Distribution Analysis

The following pseudocode outlines the process used to analyze the tag distributions.  These steps were performed in the last portion of the  "CP2_02_Load DF Clean and Analyze.ipynb".

7.  ...
8.  Get a list of all the Tags and Frequency Counts
9.  Exploratory analysis / visualization of tags related to their frequency
10. Exploratory analysis / visualization on other features such as ViewCount and Score
11. Create TopTagsList based both on frequency and ranking by ViewCount and Score
12. Write out the filtered data frame with new features

For these steps, the following libraries were used: collections, Beautiful Soup, nltk, and re. For visualizations the following libraries were used: matplotlib, seaborn, and wordcloud.
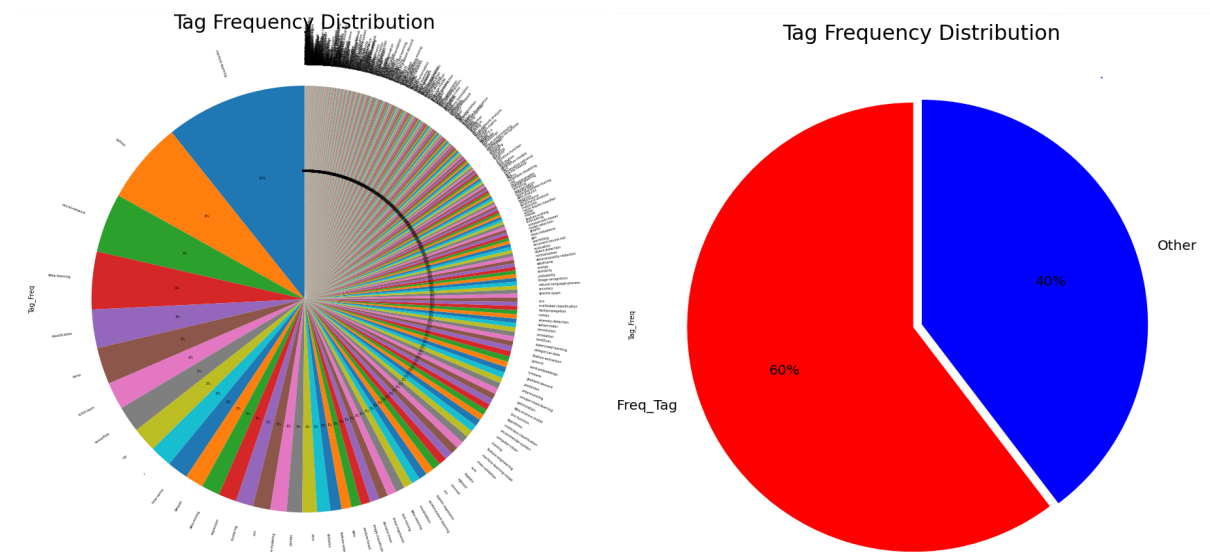


*Fig 10: Pie chart showing the frequency distribution of all of the tags.  There are a total of 590 unique tags in the dataset, but the top 32 tags represent 60% of the data.*

As shown in Figure 10, of a total of 590 unique tags, the top 32 most frequent tags encompass 60% of the total data.
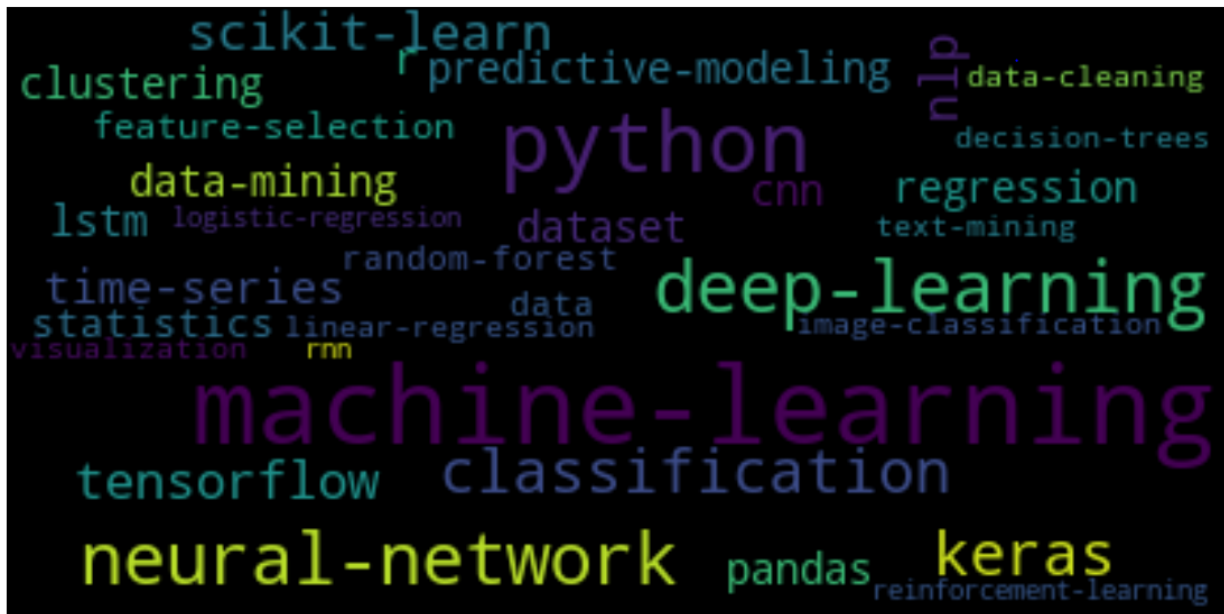
*Fig 11: Wordcloud of top 32 tags, showing the popularity of machine and deep learning topics.*

Figure 11 shows the top 32 tags as a wordcloud. You can see that machine-learning, deep-learning, and neural-networks and their associated software and libraries such as keras and tensorflow are the hottest topics in the forum.

Additional exploratory analysis was also performed on a newly created feature, the number of tags per question (TagCount), ViewCount, and Score, and the correlations between them.
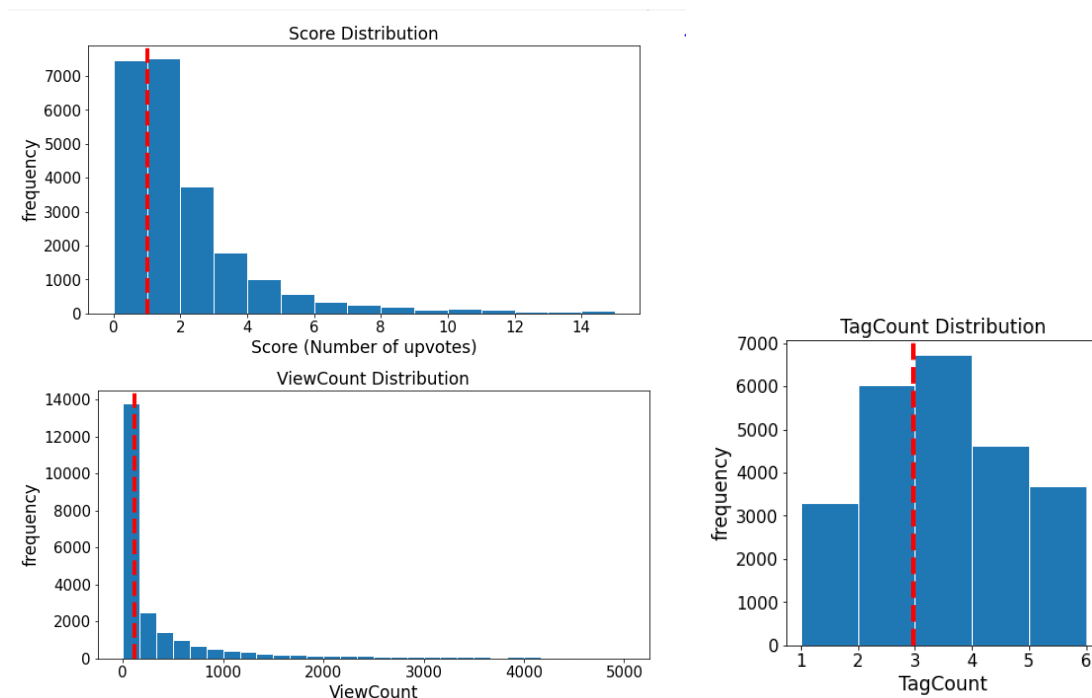


*Fig 12: Frequency Distribution of Score, ViewCounts, and TagCount.*

|       | Score        | ViewCount     | TagCount     |
|-------|--------------|---------------|--------------|
| count | 24353.000000 | 24353.000000  | 24353.000000 |
| mean  | 2.005872     | 1387.329651   | 2.974705     |
| std   | 5.433220     | 6895.850492   | 1.257866     |
| min   | -6.000000    | 2.000000      | 1.000000     |
| 25%   | 0.000000     | 36.000000     | 2.000000     |
| 50%   | 1.000000     | 114.000000    | 3.000000     |
| 75%   | 2.000000     | 578.000000    | 4.000000     |
| max   | 214.000000   | 234355.000000 | 5.000000     |

*Table 2:  Feature Statistics*

As would be expected for count features and shown in Figure 12 and Table 2, both Score and ViewCount have a skewed poisson distribution with mean values of 2 and 1387 respectively. TagCount has a normal distribution with min of 2, and maximum number of tags per question of 5.

Further exploratory analysis was performed to identify whether the questions with top tags have higher Scores, ViewCounts or TagCounts on average than the others.  The questions were split into 2 groups, those with tags on the TopTagsList and those without, and means for these attributes were calculated for each group.  The results are shown in Figure 13 below.
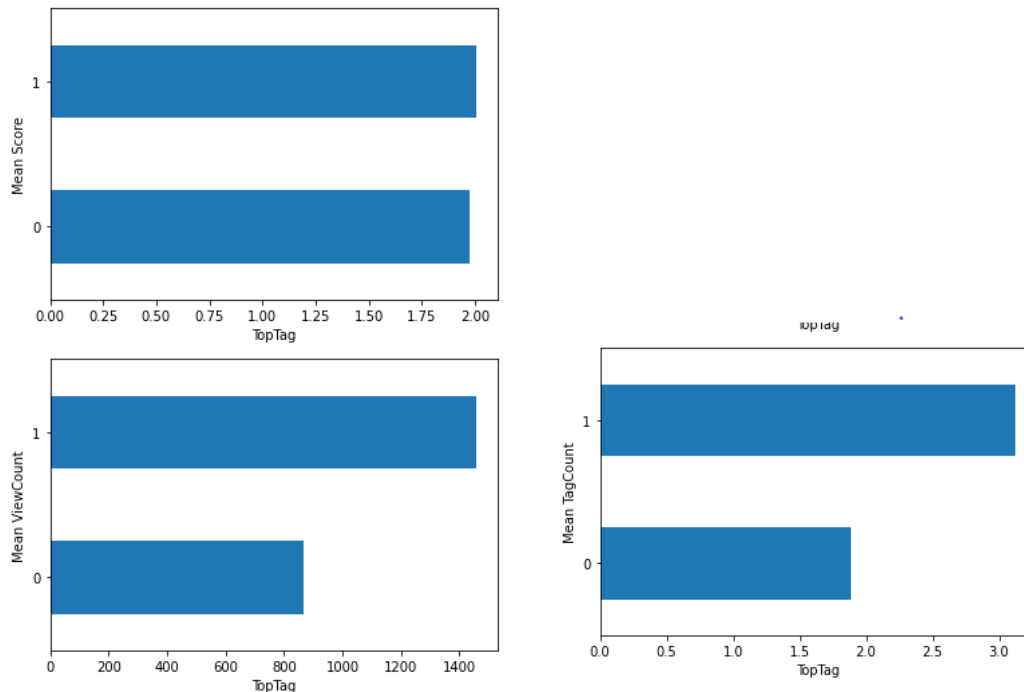


*Fig 13:  Mean Score, ViewCount, and TagCount by TopTag Grouping*

The results indicate that the mean difference of the Score between the questions with Top Tags and those without is not statistically significant; however the ViewCounts and TagCounts are statistically significant being almost twice the value where questions are tagged with one or more of the top 32 most frequent tags.

Based on the strong correlations between these latter two attributes and the top tags, the tags were further high-graded, and re-ranked using not only the tag frequency but also the ViewCount and Score attributes. Rank order was determined by tag frequency + ViewCount + TagCount.

During this analysis, it became clear that use of all tags (up to 5 per question) produced more scatter and variation in the TopTag results. Using only the first tag for each question and excluding all the remaining tags (plus rank ordering) produced a list of 33 TopTags that comprised 88% of the data (as opposed to the initial 60%). It was therefore decided to use only the first (and most frequent) tag for each question for tag prediction.
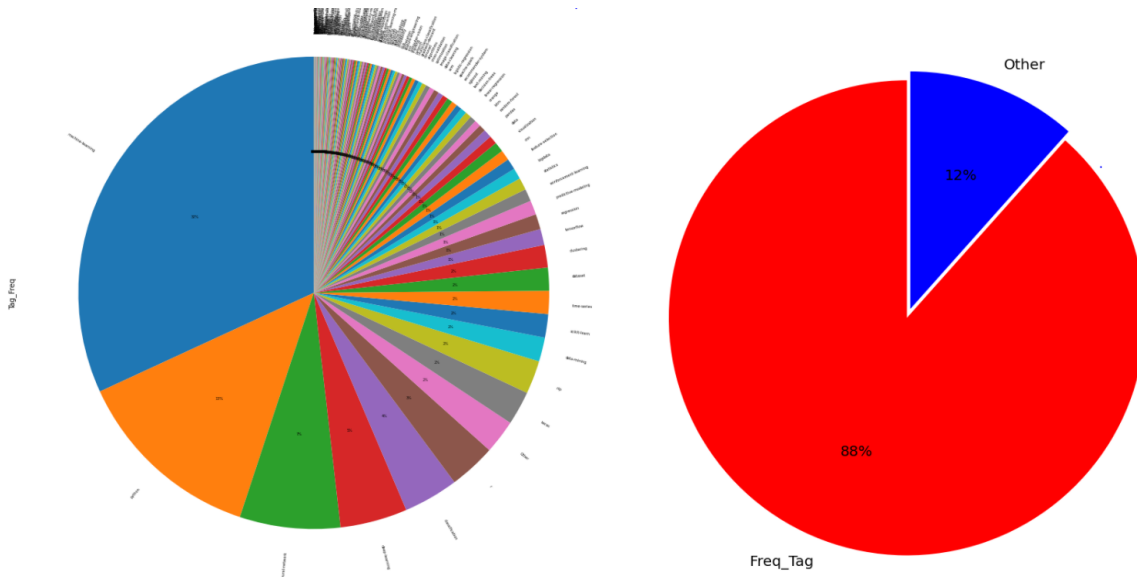


*Fig 14: Pie chart showing the frequency distribution of the ranked ordered first tags. There are a total of 349 unique first tags in the dataset, but the top 33 tags represent 88% of the data.*

As shown in Figure 14, of a total of 349 unique first tags, the top 33 most frequent tags now encompass 88% of the total data.
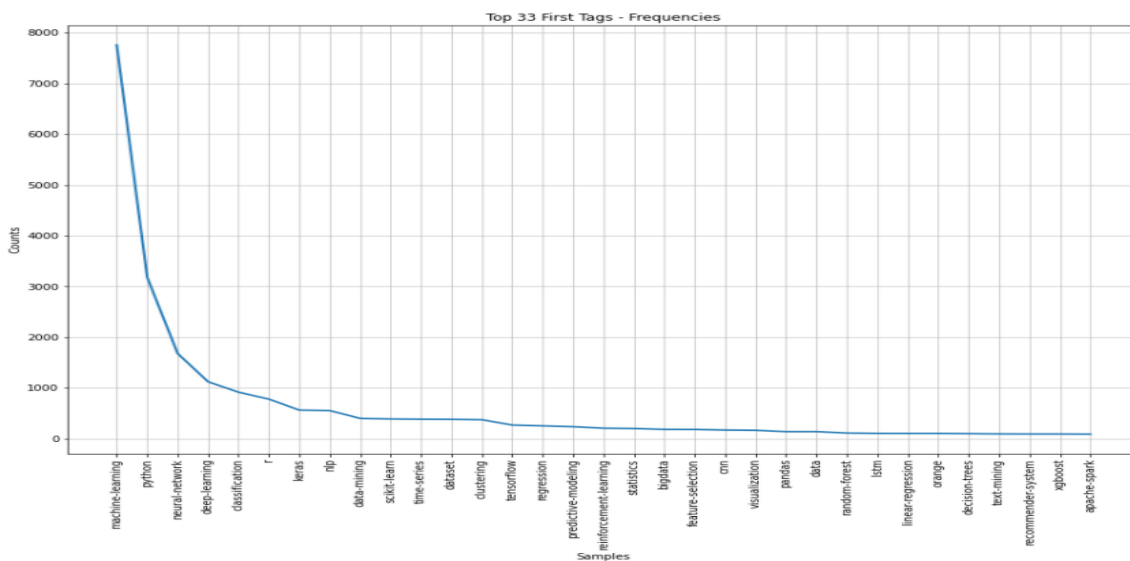


*Fig 15:  Revised Frequency distribution of the ranked ordered first 33 tags.*

The revised 33 first top tags were used for the prediction. All other tags were renamed to 'OTHER'.

Output is a pandas dataframe called questions_df consisting of 24363 entries and 40 columns. The associated pickle file exported is named questions_df_09252020.pickle.

## Question Text "Body" Analysis

The following pseudocode outlines the process used to analyze the 'Body' attribute - the question text. Some of these steps were performed in the preliminary portion of the "CP2_03_Clean_Body_Text_Analyze.ipynb"

2. …
3. Examine the feature 'Body' which are the questions to gain statistical insights
4. Get counts of certain special characters to add to features prior to cleaning
   a. Question Marks
   b. Bolded Text
   c. Number of Paragraphs
   d. Code Examples

Additional exploration and visualization steps were performed in the "CP2_04_CP2_04_Analyze_Features.ipynb" notebook.

1. Load the pickled pandas dataframe from 03 notebook
2. Examine contents to assure everything transferred properly
3. Analyze length of body (both characters and words) against other features and id any correlations
4. Analyze number of question marks against other features and id any correlations
5. Analyze text bolding against other features and id any correlations
6. Analyze number of paragraphs against other features and id any correlations
7. Analyze number of code examples and length of those examples and id any correlations
8. Analyze NER results against tags for correlation
9. Create Bag of Words and get word frequency count by tag
10. Export the dataframe with cleaned text and additional features for further analysis

Input is a pandas dataframe called questions_df consisting of 24363 entries and 63 columns. The associated pickle file is named questions_df_ner_results_10282020.pickle.

For these steps, the following libraries were used: pandas, numpy, pickle, collections.
For visualizations the following libraries were used: matplotlib and seaborn.

Freq distribution of word and character counts

No correlation between whether question tag is a top tag and question length.

Inverse relationship between both ViewCount/ Score and question length. Also inverse relationship between ViewCount/Score and a large amount of questions, bold count, and paragraph counts.

## Final DataFrame

| Date | API | API8 | sidetrack_num | well_type_cat | Oper_Cur_Num | Oper_Cur_Name | Oper_Hist_Num | Oper_Hist_Name | Well_Title | Ground_Ele |
|---|---|---|---|---|---|---|---|---|---|---|
| 2012-12-01 | 10975301 | 109753 | 01 | Horizontal | 10646 | AXIS EXPLORATION LLC | 10338 | CARRIZO OIL & GAS INC | 4-28-11-3-64 WEP | 5579.0 |
| 2013-01-01 | 10975301 | 109753 | 01 | Horizontal | 10646 | AXIS EXPLORATION LLC | 10133 | HILCORP ENERGY COMPANY | 4-28-11-3-64 WEP | 5579.0 |
| 2013-02-01 | 10975301 | 109753 | 01 | Horizontal | 10646 | AXIS EXPLORATION LLC | 10439 | CARRIZO NIOBRARA LLC | 4-28-11-3-64 WEP | 5579.0 |
| 2013-03-01 | 10975301 | 109753 | 01 | Horizontal | 10646 | AXIS EXPLORATION LLC | 10338 | CARRIZO OIL & GAS INC | 4-28-11-3-64 WEP | 5579.0 |
| 2013-05-01 | 10975301 | 109753 | 01 | Horizontal | 10646 | AXIS EXPLORATION LLC | 10338 | CARRIZO OIL & GAS INC | 4-28-11-3-64 WEP | 5579.0 |
| 2013-06-01 | 10975301 | 109753 | 01 | Horizontal | 10646 | AXIS EXPLORATION LLC | 10439 | CARRIZO NIOBRARA LLC | 4-28-11-3-64 WEP | 5579.0 |
| 2013-07-01 | 10975301 | 109753 | 01 | Horizontal | 10646 | AXIS EXPLORATION LLC | 10439 | CARRIZO NIOBRARA LLC | 4-28-11-3-64 WEP | 5579.0 |
| 2013-08-01 | 10975301 | 109753 | 01 | Horizontal | 10646 | AXIS EXPLORATION LLC | 10338 | CARRIZO OIL & GAS INC | 4-28-11-3-64 WEP | 5579.0 |

*Fig 3: Slice of final dataframe - need to update.*

Unnecessary columns were dropped; performed after complete exploratory analysis steps as these data were needed to understand any important features needed for the prediction process.

## 4. COUNT VECTORIZATION, TF-IDF and WORD EMBEDDINGS

The final step in data conditioning is to convert the text to a numerical format. Three different methods were employed and will be compared. The first process is count vectorization, the second is tf-idf. ELABORATE

And finally a more sophisticated method of vectorizing the text to create word embeddings will be employed. We will then be able to present all the words as vectors in feature space

The following pseudocode outlines the process to perform these tasks. These steps were performed in the "CP2_05_CountVectorization.ipynb" notebook.

# 5. FEATURE ENGINEERING

A number of additional features were calculated or extrapolated from the original datasets to aid in the analysis.

A total of ?? features
The following is a list of features and their description.

| Features | Description |
|----------|-------------|
|          |             |

# 6. MACHINE LEARNING

Build an algorithm that creates decision boundaries between these; this will be a multi-class classification problem. Therefore probabilities for multiple tags will be determined for each question; what packages used (sci-kit learn)

Split into train and test sets and run the model.

Evaluate the model; a jacard score will be used to evaluate since it is well suited for evaluating multi-class classifications.

Outline steps

## Finding the Optimal K value

## K-Means Clustering

Applying the model with K=6, results in 6 subgroups with distribution of grids as shown below,

## Visualization of K Clusters

### Principal Component Analysis

Displaying the clusters in subgroups in 92 dimensions is not possible. We will reduce the dimensions to 2 dimensions using Principal Component Analysis [PCA] in order to visualize the datapoints in subgroups.

## Analysis of K Clusters

Clusters are plotted against different dimensions. Subgroups from the clustering model shows defined volume ranges for each dimension. Presence of outliers indicates wrong assignment of the data points to a subgroup with respect to that dimension alone. In general, this clustering model has done well for Internet volumes. In case of SMS & Calls Cluster1, Cluster5 & Cluster3 are clearly formed with few outliers only.

T-tests and one-way ANOVA tests can be performed on the subgroups for different dimensions in order to understand the similarities and dissimilarities in their behavioral patterns.

### Model Evaluation

After identifying the parameter values using auto_arima(), we fit our model on the new dataset, and use it to evaluate its accuracy by predicting volumes for days whose volumes are known.

```
MAPE = 5.12%
Accuracy = 94.88%
The Root Mean Squared Error of our prediction is 15270.31
```

*Fig 38: Model Evaluation*

# 7. REFERENCES

- Colorado Oil and Gas Conservation Commission
  https://cogcc.state.co.us/data2.html#/downloads

- Milne, 2014

- Wikipedia https://en.wikipedia.org/wiki/Denver_Basin

# 8. APPENDIX

## 8.1 Stack Exchange Data Science Data Schema