ML1010

Independant Coding Project 2

Problem:
Examining numeric columns was becoming a frequent task and beginning to consume a great deal of time. In the Group Project it became frequent to determine the distribution of items in our dataset. For example, while examining the length of the review, or the number of tokens, the numbers were extremely skewed with very short reviews dominating the dataset, while long reviews were much less common. It was difficult to understand and visualize this information to make an informed decision about which data to include and exclude from different experiements

Solution:
Create a utility function to allow for seeing the data distribution in various capacities by zooming in on subranges of the data as well as changing the reporting detail (data grouping by numeric binning)

Configuration

```
#Parameters
PROJECT_NAME = 'ML1010_Weekly'
ENABLE_COLAB = True

#Root Machine Learning Directory. Projects appear underneath
GOOGLE_DRIVE_MOUNT = '/content/gdrive'
COLAB_ROOT_DIR = GOOGLE_DRIVE_MOUNT + '/MyDrive/Colab Notebooks'
COLAB_INIT_DIR = COLAB_ROOT_DIR + '/utility_files'

LOCAL_ROOT_DIR = '/home/magni/Documents/ML_Projects'
LOCAL_INIT_DIR = LOCAL_ROOT_DIR + '/utility_files'
```

Bootstrap Environment

```
#add in support for utility file directory and importing
import sys
import os

if ENABLE_COLAB:
```

```
    #Need access to drive
    from google.colab import drive
    drive.mount(GOOGLE_DRIVE_MOUNT, force_remount=True)

    #add in utility directory to syspath to import
    INIT_DIR = COLAB_INIT_DIR
    sys.path.append(os.path.abspath(INIT_DIR))

    #Config environment variables
    ROOT_DIR = COLAB_ROOT_DIR

  else:
    #add in utility directory to syspath to import
    INIT_DIR = LOCAL_INIT_DIR
    sys.path.append(os.path.abspath(INIT_DIR))

    #Config environment variables
    ROOT_DIR = LOCAL_ROOT_DIR

  #Import Utility Support
  from jarvis import Jarvis
  jarvis = Jarvis(ROOT_DIR, PROJECT_NAME)

  import mv_python_utils as mvutils
```

```
Mounted at /content/gdrive
Wha...where am I?
I am awake now.

I have set your current working directory to /content/gdrive/MyDrive/Colab Notebooks/ML
The current time is 11:18
Hello sir. Extra caffeine may help.
```

Setup Runtime Environment

```
if ENABLE_COLAB:
  #!pip install scipy -q
  #!pip install scikit-learn -q
  #!pip install pycaret -q
  #!pip install matplotlib -q
  #!pip install joblib -q
  #!pip install pandasql -q

  display('Google Colab enabled')
else:
  display('Google Colab not enabled')
```

```
#Common imports
import json
import gzip
import pandas as pd
import numpy as np
import matplotlib
import re
import nltk
import matplotlib.pyplot as plt

pd.set_option('mode.chained_assignment', None)
nltk.download('stopwords')
%matplotlib inline
```

```
'Google Colab enabled'
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Load Data

```
jarvis.showAllDataFiles()
```

```
[D] /content/gdrive/MyDrive/Colab Notebooks/data/Jarvis/04_test
---[   gz][  csv]--> pima-indians-diabetes.csv.gz (8.53 KB)
---[   gz][  csv]--> wk3_task_data.csv.gz (33.47 KB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project [Empty director

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/01_original
---[   gz][ json]--> Cell_Phones_and_Accessories_5.json.gz (161.24 MB)
---[   gz][ json]--> meta_Cell_Phones_and_Accessories.json.gz (343.33 MB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/02_working
[*][  pkl]---------> 01_Cellphone_small.pkl (45.46 MB)
---[   gz][  pkl]--> 01_NLP_ReviewText_Narrow_1.pkl.gz (6.88 MB)
---[   gz][  pkl]--> 01_NLP_ReviewText_Narrow_2.pkl.gz (170.55 MB)
---[   gz][  pkl]--> 01_NLP_ReviewText_Narrow_3.pkl.gz (295.59 MB)
[*][  pkl]---------> 01_NLP_ReviewText_small.pkl (28.94 MB)
[*][  pkl]---------> 01_NLP_Summary_small.pkl (3.82 MB)
[*][  pkl]---------> 01_NLP_Title_small.pkl (2.73 MB)
---[   gz][  pkl]--> 01_NL_ReviewText_All(new).pkl.gz (593.23 MB)
---[   gz][  pkl]--> 01_NL_ReviewText_All.pkl.gz (592.92 MB)
---[   gz][  pkl]--> 01_NL_ReviewText_textSplit.pkl.gz (15.78 MB)
[*][  pkl]---------> 02_Cellphone.pkl (46.32 MB)
[*][  pkl]---------> 02_NLP_ReviewTextData.pkl (87.00 MB)
[*][  pkl]---------> 02_NLP_SummaryData.pkl (8.32 MB)
[*][  pkl]---------> 02_NLP_TitleData.pkl (16.71 MB)
[*][  pkl]---------> 03_Cellphone.pkl (46.31 MB)
[*][  pkl]---------> 03_NLP_ReviewTextData.pkl (28.94 MB)
[*][  pkl]---------> 03_NLP_ReviewText_Narrow.pkl (17.13 MB)
[*][  pkl]---------> 03_NLP_SummaryData.pkl (3.82 MB)
[*][  pkl]---------> 03_NLP_TitleData.pkl (2.73 MB)
```

```
[*][  pkl]---------> 03_NLP_TitleData.pkl (2.73 MB)
[*][  pkl]---------> 04_NLP_ReviewText_Narrow.pkl (16.95 MB)
[*][  pkl]---------> 05_NLP_ReviewText_Narrow.pkl (66.15 MB)
[*][  pkl]---------> 05_NLP_ReviewText_Narrow_full.pkl (207.91 MB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/03_train [Empty

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/04_test [Empty

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010_Weekly
---[   gz][  csv]--> complaints.csv.gz (370.67 MB)
[*][  csv]---------> movie_reviews_cleaned.csv (38.37 MB)
[*][  csv]---------> pima-indians-diabetes.csv (22.73 KB)
---[   gz][  tsv]--> rspct.tsv.gz (347.13 MB)
---[   gz][  csv]--> subreddit_info.csv.gz (37.80 KB)
[*][  csv]---------> wk3_task_data.csv (81.31 KB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010_Weekly/01_original [Empty dir

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010_Weekly/02_working [Empty dire

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010_Weekly/03_train [Empty direct

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010_Weekly/04_test [Empty directo

[D] /content/gdrive/MyDrive/Colab Notebooks/data/test_compress
[*][  pkl]---------> 02_NLP_SummaryData.pkl (8.32 MB)
[*][  pkl]---------> 02_NLP_TitleData.pkl (16.71 MB)
```

```
df = pd.read_pickle('/content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/02_wor
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63413 entries, 0 to 63412
Data columns (total 49 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   uuid                                  63413 non-null  object
 1   reviewText                            63413 non-null  object
 2   overall                               63413 non-null  float64
 3   reviewText_lemma                      63413 non-null  object
 4   reviewText_nouns                      63413 non-null  object
 5   reviewText_adjectives                 63413 non-null  object
 6   reviewText_verbs                      63413 non-null  object
 7   reviewText_nav                        63413 non-null  object
 8   reviewText_lemma_tb_pol               63310 non-null  float64
 9   reviewText_lemma_tb_subj              63310 non-null  float64
 10  reviewText_lemma_tb_tokens            63310 non-null  float64
 11  reviewText_lemma_tb_length            63310 non-null  float64
 12  reviewText_lemma_bert                 63413 non-null  object
 13  reviewText_lemma_flairSent            63310 non-null  float64
 14  reviewText_adjectives_tb_pol          50732 non-null  float64
 15  reviewText_adjectives_tb_subj         50732 non-null  float64
```

```
 16  reviewText_adjectives_tb_tokens         50732 non-null  float64
 17  reviewText_adjectives_tb_length         50732 non-null  float64
 18  reviewText_adjectives_bert              63413 non-null  object
 19  reviewText_adjectives_flairSent         50732 non-null  float64
 20  reviewText_verbs_tb_pol                 43234 non-null  float64
 21  reviewText_verbs_tb_subj                43234 non-null  float64
 22  reviewText_verbs_tb_tokens              43234 non-null  float64
 23  reviewText_verbs_tb_length              43234 non-null  float64
 24  reviewText_verbs_bert                   63413 non-null  object
 25  reviewText_verbs_flairSent              43234 non-null  float64
 26  reviewText_nav_tb_pol                   62332 non-null  float64
 27  reviewText_nav_tb_subj                  62332 non-null  float64
 28  reviewText_nav_tb_tokens                62332 non-null  float64
 29  reviewText_nav_tb_length                62332 non-null  float64
 30  reviewText_nav_bert                     63413 non-null  object
 31  reviewText_nav_flairSent                62332 non-null  float64
 32  overall_posneg                          63413 non-null  int64
 33  reviewText_lemma_flairSent_norm         63310 non-null  float64
 34  reviewText_lemma_flairSent_posneg       63310 non-null  float64
 35  reviewText_adjectives_flairSent_norm    50732 non-null  float64
 36  reviewText_adjectives_flairSent_posneg  50732 non-null  float64
 37  reviewText_verbs_flairSent_norm         43234 non-null  float64
 38  reviewText_verbs_flairSent_posneg       43234 non-null  float64
 39  reviewText_nav_flairSent_norm           62332 non-null  float64
 40  reviewText_nav_flairSent_posneg         62332 non-null  float64
 41  reviewText_lemma_tb_pol_norm            63310 non-null  float64
 42  reviewText_lemma_tb_pol_posneg          63310 non-null  float64
 43  reviewText_adjectives_tb_pol_norm       50732 non-null  float64
 44  reviewText_adjectives_tb_pol_posneg     50732 non-null  float64
 45  reviewText_verbs_tb_pol_norm            43234 non-null  float64
 46  reviewText_verbs_tb_pol_posneg          43234 non-null  float64
 47  reviewText_nav_tb_pol_norm              62332 non-null  float64
 48  reviewText_nav_tb_pol_posneg            62332 non-null  float64
dtypes: float64(37), int64(1), object(11)
memory usage: 23.7+ MB
```

Independant Code Exploration

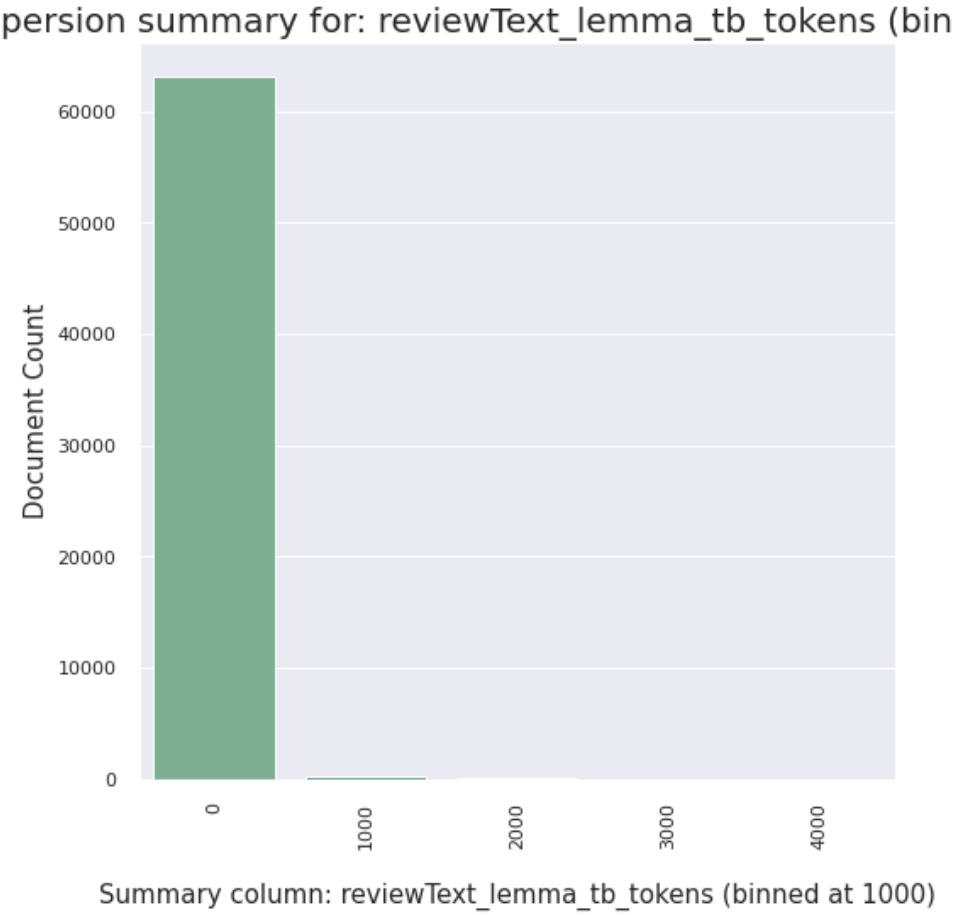
```
mvutils.showColumnSummary(df, 'reviewText_lemma_tb_tokens')
```

```
Dataframe shape (63413, 49)
Analysis column: reviewText_lemma_tb_tokens
Distinct values (incl. null): 1014
Number of na   values: 103
Number of null values: 103
Total documents in corpus: 63413
```

```
#Examine numeric column for distribution
mvutils.examineColumnNumeric(df,
```

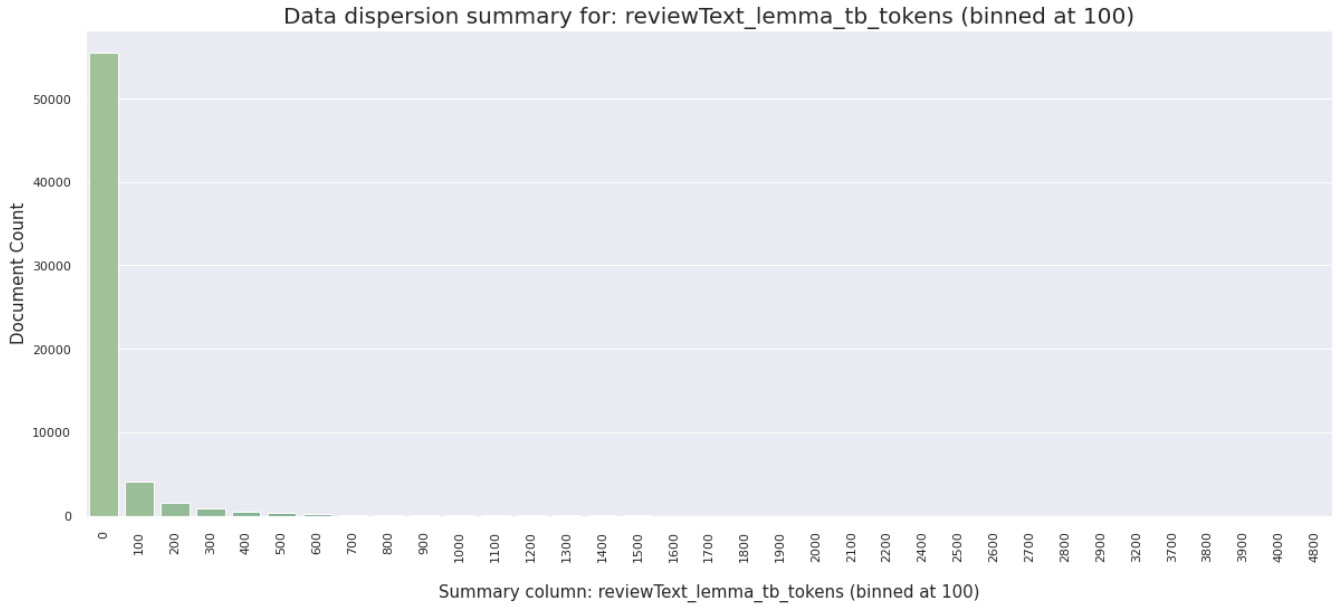
```
                        'reviewText_lemma_tb_tokens'
                        )
```

```
Warning: 103 null values detected in column. Removing for analysis
```



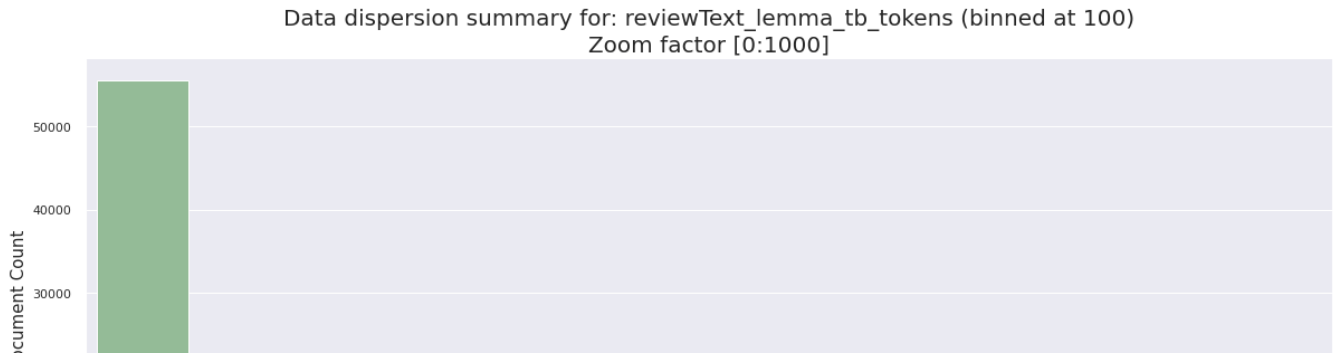

```
#Increase plotsize for better viewing,
#change binning size to 100 from default 1000
mvutils.examineColumnNumeric(df,
                        'reviewText_lemma_tb_tokens',
                        binsize=100,
                        plotsize=5
                        )
```

```
Warning: 103 null values detected in column. Removing for analysis
```




```
#Enable zoom to examine range 0-1000 binned at 100 for better viewing
mvutils.examineColumnNumeric(df,
                             'reviewText_lemma_tb_tokens',
                             binsize=100,
                             zoom=True,
                             minZoomLevel=0,
                             maxZoomLevel=1000,
                             plotsize=5)
```

```
Warning: 103 null values detected in column. Removing for analysis
```




```
#Data still dominated by 0-100
#Zoom to 0-200 with bin size of 10
mvutils.examineColumnNumeric(df,
                             'reviewText_lemma_tb_tokens',
                             binsize=10,
                             zoom=True,
                             minZoomLevel=0,
                             maxZoomLevel=200,
                             plotsize=5)
```

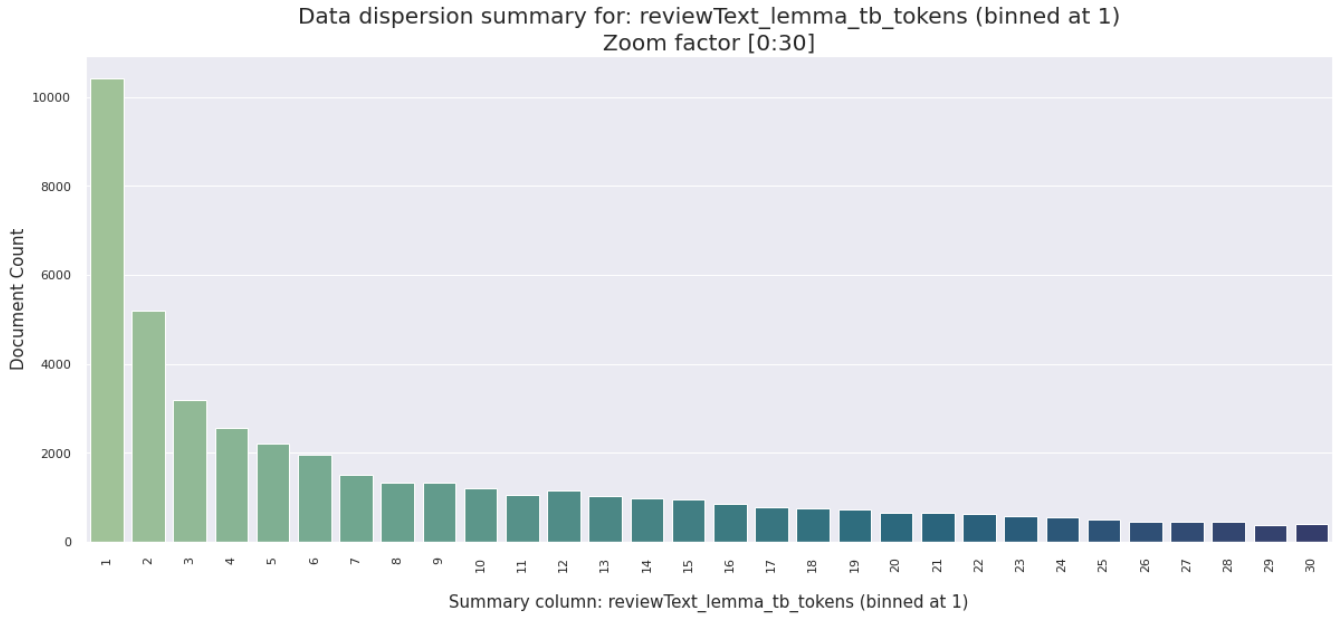
```
Warning: 103 null values detected in column. Removing for analysis
```

```
import importlib
importlib.reload(mvutils)
```

```
<module 'mv_python_utils' from '/content/gdrive/MyDrive/Colab Notebooks/utility_files/m
```


```
#Still not quite enough detail to determine where to cutoff
#Zoom with binsize 1, range 0-30
mvutils.examineColumnNumeric(df,
                             'reviewText_lemma_tb_tokens',
                             binsize=1,
                             zoom=True,
                             minZoomLevel=0,
                             maxZoomLevel=30,
                             plotsize=5,
                             verbose=True,
                             numRecords=5)
```

```
Warning: 103 null values detected in column. Removing for analysis
```

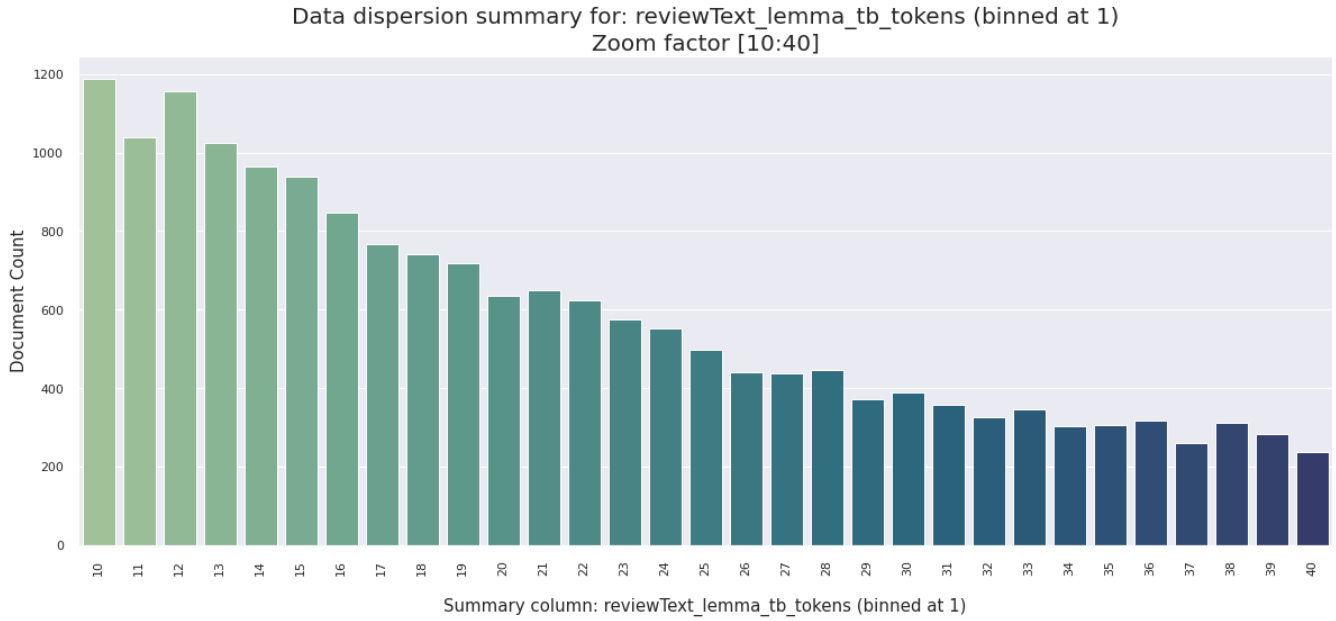



```
dataframe shape: (30, 2)

dataframe info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   bin_at_1     30 non-null     int64
 1   binnedCount  30 non-null     int64
dtypes: int64(2)
memory usage: 608.0 bytes
None

Top 5 in dataframe
```

	bin_at_1	binnedCount
0	30	388

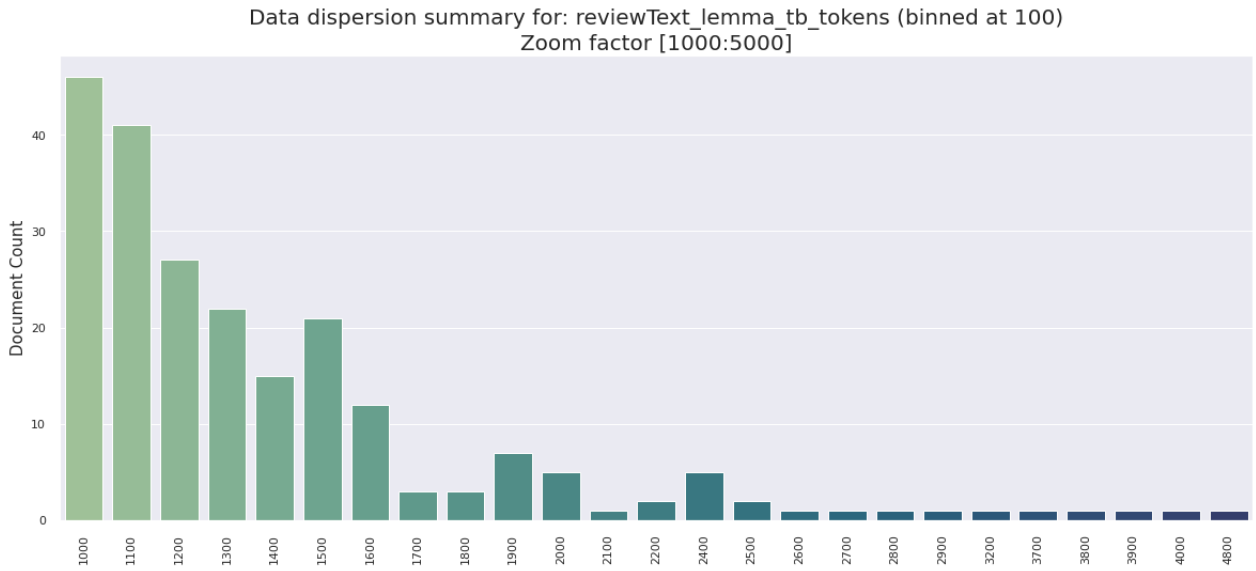
```
#Need a better view and scale for mid range numbers
#Zoom 10:40 with binsize=1
mvutils.examineColumnNumeric(df,
                             'reviewText_lemma_tb_tokens',
                             binsize=1,
                             zoom=True,
                             minZoomLevel=10,
                             maxZoomLevel=40,
                             plotsize=5)
```

```
Warning: 103 null values detected in column. Removing for analysis
```



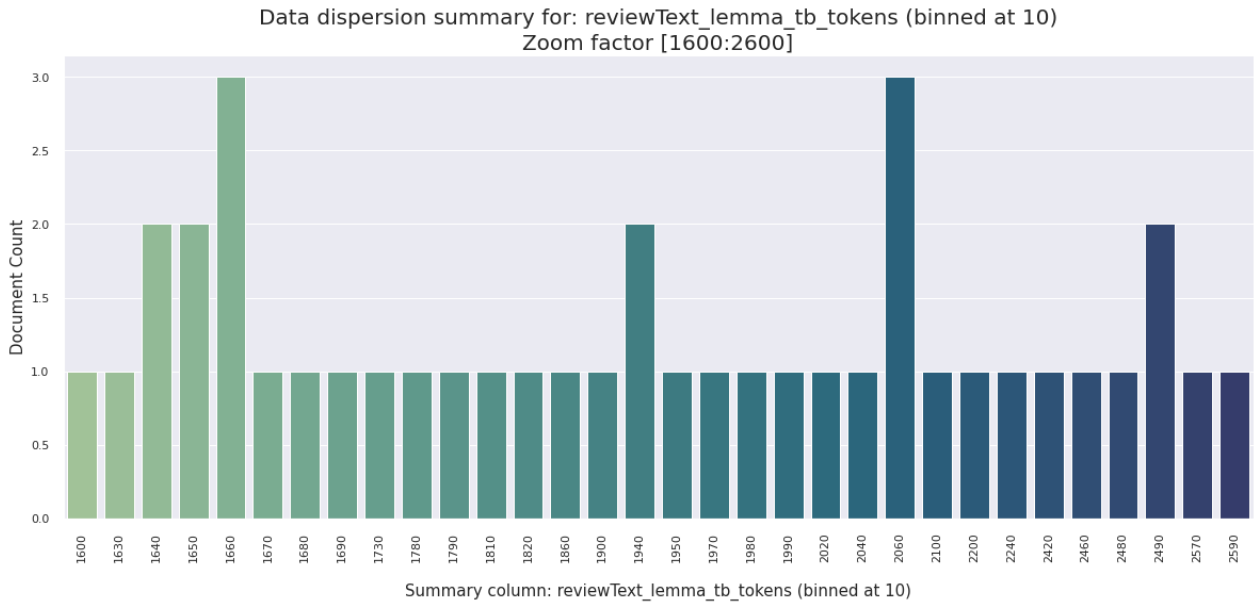

```
#Examine tail end of data (larger # of tokens)
#Zoom 1000:5000 binsize 100
mvutils.examineColumnNumeric(df,
                             'reviewText_lemma_tb_tokens',
                             binsize=100,
                             zoom=True,
                             minZoomLevel=1000,
                             maxZoomLevel=5000,
                             plotsize=5)
```

```
Warning: 103 null values detected in column. Removing for analysis
```




```
#Examine dropoff near the 1700 range
#Zoom 1600:2600, binsize 10
mvutils.examineColumnNumeric(df,
                             'reviewText_lemma_tb_tokens',
                             binsize=10,
                             zoom=True,
                             minZoomLevel=1600,
                             maxZoomLevel=2600,
                             plotsize=5)
```

```
Warning: 103 null values detected in column. Removing for analysis
```



✓ 1s completed at 11:48 AM

```
 1 # -*- coding: utf-8 -*-
 2
 3 import os
 4 import pandas as pd
 5 import seaborn as sns
 6 import matplotlib.pyplot as plt
 7 import matplotlib.gridspec as gridspec
 8 import matplotlib.ticker as mtick
 9 import numpy as np
10 import math
11 from yellowbrick.target import ClassBalance
12
13
14 # Start dataframe functions
15 def exploreDataframe(data, numRecords=1):
16     print("dataframe shape: " + str(data.shape))
17     print("\ndataframe info: ")
18     print(data.info())
19     print("\nTop " + str(numRecords) + " in dataframe")
20     display(data.head(numRecords))
21     print("\nBottom " + str(numRecords) + " in
   dataframe")
22     display(data.tail(numRecords))
23
24 def showUniqueColVals(series):
25     i = 0
26     for l in np.unique(series):
27         i += 1
28         print(l)
29     print("\n--->Unique values: " + str(i))
30
31
32 # End dataframe functions
33
34
35 # Start generic ML functions
36
37 # Create a summary frame for us to append results
```

```
38 def initSummaryFrame():
39     column_names = ["analysisStage",
40                     "columnName",
41                     "totalDocuments",
42                     "totalNulls",
43                     "totalCategories",
44                     "upperPerc",
45                     "lowerPerc",
46                     "percTotalDocuments",
47                     "percTotalCategories"
48                     ]
49     fnDf = pd.DataFrame(columns=column_names)
50     return fnDf
51
52
53 # Add columns for upper and lower boundaries
54 def addUpperAndLowerPercColumns(data,
55                                 upperPerc,
56                                 lowerPerc,
57                                 upperPercColName='
   upperPerc',
58                                 lowerPercColName='
   lowerPerc'):
59     fnDf = data
60     fnDf[upperPercColName] = upperPerc
61     fnDf[lowerPercColName] = lowerPerc
62     return fnDf
63
64
65 # Add results data to summary frame
66 def appendResultsData(resultsFrame,
67                       analysisStage,
68                       analysisColName,
69                       totalDocuments,
70                       totalNulls,
71                       totalCategories,
72                       upperPerc,
73                       lowerPerc,
```

```
 74                          upperPercColName='upperPerc',
 75                          lowerPercColName='lowerPerc',
 76                          displayAppendedFrame=False):
 77     if len(resultsFrame) == 0:
 78         # this is the first row (should be original)
 79         # add the perc total documents/columns
 80         percTotalDocuments = 1.0
 81         percTotalCategories = 1.0
 82     else:
 83         allCategories = resultsFrame['totalCategories'
    ].values[0]
 84         allDocuments = resultsFrame['totalDocuments'].
    values[0]
 85 
 86         percTotalDocuments = round(totalDocuments /
    allDocuments, 2)
 87         percTotalCategories = round(totalCategories /
    allCategories, 2)
 88 
 89     new_row = {'analysisStage': analysisStage,
 90                'columnName': analysisColName,
 91                'totalDocuments': totalDocuments,
 92                'totalNulls': totalNulls,
 93                'totalCategories': totalCategories,
 94                upperPercColName: upperPerc,
 95                lowerPercColName: lowerPerc,
 96                'percTotalDocuments':
    percTotalDocuments,
 97                'percTotalCategories':
    percTotalCategories
 98                }
 99     # append row to the dataframe
100     resultsFrame = resultsFrame.append(new_row,
    ignore_index=True)
101     if displayAppendedFrame:
102         print('Results appended. New frame:')
103         display(resultsFrame)
104 
```

```
105     return resultsFrame
106 
107 
108 def buildReportingFrame(df,
109                         analysisColName,
110                         summaryColName,
111                         cumulativeColName,
112                         colOrderName,
113                         percTotalColName,
114                         percCumulativeColName,
115                         ):
116     tDf = df.copy()
117 
118     tDf[analysisColName] = tDf[analysisColName].astype
    ('string')
119 
120     tDf = df.groupby([analysisColName]).size().
    to_frame(summaryColName)
121     tDf = tDf.sort_values(by=summaryColName, ascending
    =False)
122     tDf[cumulativeColName] = tDf[summaryColName].
    cumsum()
123 
124     totalRows = tDf[summaryColName].sum()
125 
126     tDf[percTotalColName] = round(tDf[summaryColName
    ] / totalRows, 2)
127     tDf[percCumulativeColName] = round(tDf[
    cumulativeColName] / totalRows, 2)
128     tDf[colOrderName] = np.arange(len(tDf)) + 1
129 
130     tDf.reset_index(inplace=True)
131     tDf.index = np.arange(1, len(tDf) + 1)
132 
133     tDf = tDf[
134         [colOrderName, analysisColName, summaryColName
    , cumulativeColName, percTotalColName,
    percCumulativeColName]]
```

```
135
136     return tDf
137
138
139 def showColumnSummary(df,
140                       analysisColName
141                       ):
142     tDf = df.copy()
143     totalDocuments = len(tDf)
144     totalNulls = tDf[analysisColName].isnull().sum()
145     totalNAs = tDf[analysisColName].isna().sum()
146     totalCategories = tDf[analysisColName].nunique()
147
148     print(f'Dataframe shape {str(df.shape)}')
149     print(f'Analysis column: {analysisColName}')
150     print(f'Distinct values (incl. null): {str(
    totalCategories)}')
151     print(f'Number of na   values: {str(totalNAs)}')
152     print(f'Number of null values: {str(totalNulls)}')
153     print(f'Total documents in corpus: {str(
    totalDocuments)}', end='\n\n')
154
155
156 def analyzeReportingFrame(df,
157                           summaryColName,
158                           analysisColName,
159                           resultsFrame,
160                           analysisName="original",
161                           upperPerc=1.0,  # use
    default for initial run
162                           lowerPerc=0.0,  # use
    default for initial run
163                           showRows=2):
164     tDf = df.copy()
165
166     totalDocuments = tDf[summaryColName].sum()
167     totalCategories = tDf[analysisColName].nunique()
168     totalNulls = tDf[analysisColName].isnull().sum()
```

```
169
170     print(f'Top {str(showRows)} for summary of column
    {analysisColName}')
171     display(tDf[:showRows])
172     print("", end='\n\n')
173
174     print(f'Last {str(showRows)} for summary of column
     {analysisColName}')
175     display(tDf.tail(showRows))
176     print("", end='\n\n')
177
178     resultsFrame = appendResultsData(resultsFrame,
179                                      analysisName,
180                                      analysisColName,
181                                      totalDocuments,
182                                      totalNulls,
183                                      totalCategories,
184                                      upperPerc=
    upperPerc,
185                                      lowerPerc=
    lowerPerc
186                                      )
187     return resultsFrame
188
189
190 def getFocusedDf(data,
191                  upperPerc,
192                  lowerPerc):
193     fnDf = addUpperAndLowerPercColumns(data=data,
194                                        upperPerc=
    upperPerc,
195                                        lowerPerc=
    lowerPerc
196                                        )
197
198     fnDf['upperInclude'] = np.where(fnDf['
    percCumulative'] <= fnDf['upperPerc'], 1, 0)
199     fnDf['lowerInclude'] = np.where(fnDf['percTotal'
```

```
199 ] >= fnDf['lowerPerc'], 1, 0)
200     fnDf = fnDf[(fnDf['upperInclude'] == 1) & (fnDf['
    lowerInclude'] == 1)]
201 
202     return fnDf
203 
204 
205 def plotColumnAnalysis(df, xColName,
206                        percTotalColName,
207                        percCumulativeColName,
208                        analysisColName,
209                        summaryColName,
210                        upperPerc,
211                        lowerPerc,
212                        upperPercColName='upperPerc',
213                        lowerPercColName='lowerPerc'):
214     sns.set(rc={'figure.figsize': (20, 8)})
215     fnDf = addUpperAndLowerPercColumns(data=df,
216                                        upperPerc=
    upperPerc,
217                                        lowerPerc=
    lowerPerc
218                                        )
219 
220     fnDf2 = fnDf[[xColName, percCumulativeColName,
    percTotalColName, upperPercColName, lowerPercColName]]
221 
222     print(f'Lineplot showing {analysisColName}
    distribution')
223     fig = sns.lineplot(x=xColName, y='value', hue='
    variable', data=pd.melt(fnDf2, [xColName]))
224     fig.set(xlabel='Item rank', ylabel='Percent of all
     documents in corpus')
225     fig.set_title(f'Lineplot summary for: {
    analysisColName}')
226     fig.yaxis.set_major_formatter(mtick.
    PercentFormatter(1.0))
227     plt.show()
```

```
228     print("\n\n")
229 
230     print(f'Barplot showing {analysisColName} distribution')
231     if len(df) < 100:
232         plt.xticks(rotation='vertical')
233         fig = sns.barplot(x=analysisColName, y=summaryColName, data=fnDf)
234         fig.set(xlabel=f'Items in summary column: {analysisColName}', ylabel='Document Count')
235         fig.set_title(f'Barplot summary for: {analysisColName}')
236         plt.show()
237         print("\n\n")
238     else:
239         print(f'--->Dataset too large for barchart visibility: {str(len(df))}')
240         print('\n\n')
241 
242 
243 def plotSummaryFrame(dataFrame):
244     plotOne = ['percTotalDocuments', 'percTotalCategories']
245     plotTwo = ['totalDocuments', 'totalNulls', 'totalCategories']
246 
247     tDf = pd.melt(dataFrame, id_vars=['analysisStage'], value_vars=plotOne)
248     sns.barplot(x='variable', y='value', hue='analysisStage', data=tDf)
249     plt.show()
250 
251     print('\n\n')
252 
253     tDf = pd.melt(dataFrame, id_vars=['analysisStage'], value_vars=plotTwo)
254     sns.barplot(x='variable', y='value', hue='analysisStage', data=tDf)
```

```
255     plt.show()
256 
257 
258 def columnExplore(dataFrame,
259                   analysisColName,
260                   upperPerc=0.9,
261                   lowerPerc=0.01,
262                   summaryColName='docCount',
263                   cumulativeColName='cumulativeCount',
264                   colOrderName='order',
265                   percTotalColName='percTotal',
266                   percCumulativeColName='
    percCumulative'):
267     resultsFrame = initSummaryFrame()
268 
269     print("Beginning analysis on 'Main' frame...")
270 
271     showColumnSummary(df=dataFrame,
272                       analysisColName=analysisColName)
273 
274     fnDf = buildReportingFrame(df=dataFrame,
275                                analysisColName=
    analysisColName,
276                                summaryColName=
    summaryColName,
277                                cumulativeColName=
    cumulativeColName,
278                                colOrderName=
    colOrderName,
279                                percTotalColName=
    percTotalColName,
280                                percCumulativeColName=
    percCumulativeColName)
281 
282     resultsFrame = analyzeReportingFrame(df=fnDf,
283                                          resultsFrame=
    resultsFrame,
284 
```

```
284 summaryColName=summaryColName,
285 
    analysisColName=analysisColName)
286 
287     plotColumnAnalysis(df=fnDf,
288                        xColName=colOrderName,
289                        analysisColName=analysisColName
    ,
290                        summaryColName=summaryColName,
291                        upperPerc=upperPerc,
292                        lowerPerc=lowerPerc,
293                        percTotalColName=
    percTotalColName,
294                        percCumulativeColName=
    percCumulativeColName)
295 
296     print("Beginning analysis on 'Focused' frame
    ......")
297     focusDf = getFocusedDf(data=fnDf,
298                            upperPerc=upperPerc,
299                            lowerPerc=lowerPerc)
300 
301     resultsFrame = analyzeReportingFrame(df=focusDf,
302                                          resultsFrame=
    resultsFrame,
303                                          analysisName=
    "Trimmed",
304                                          upperPerc=0.9
    ,
305                                          lowerPerc=0.
    01,
306 
    summaryColName=summaryColName,
307 
    analysisColName=analysisColName)
308 
309     plotColumnAnalysis(df=focusDf,
310                        xColName=colOrderName,
```

```
311                          analysisColName=analysisColName
    ,
312                          summaryColName=summaryColName,
313                          upperPerc=upperPerc,
314                          lowerPerc=lowerPerc,
315                          percTotalColName=
    percTotalColName,
316                          percCumulativeColName=
    percCumulativeColName)
317 
318     display(resultsFrame)
319     plotSummaryFrame(resultsFrame)
320 
321     return resultsFrame
322 
323 
324 def setPlotSize(plotsize):
325     if plotsize == 5:
326         sns.set(rc={'figure.figsize': (20, 8)})
327     elif plotsize == 4:
328         sns.set(rc={'figure.figsize': (15, 8)})
329     elif plotsize == 3:
330         sns.set(rc={'figure.figsize': (10, 8)})
331     elif plotsize == 2:
332         sns.set(rc={'figure.figsize': (8, 8)})
333     elif plotsize == 1:
334         sns.set(rc={'figure.figsize': (4, 8)})
335     else:  # Should be size 1
336         # should only be one but catch it and default
    to size 1
337         sns.set(rc={'figure.figsize': (4, 4)})
338 
339 
340 def examineColumnNumeric(df, colName,
341                          binsize=1000,
342                          verbose=False,
343                          zoom=False,
344                          minZoomLevel=0,
```

```
                            maxZoomLevel=0,
                            plotsize=1,
                            numRecords=10
                            ):
    binColName = f'bin_at_{str(binsize)}'
    binnedCountName = 'binnedCount'

    #Make a copy of the incoming frame as we will be manipulating it
    tDf = df.copy()

    # Parameter checking
    if (binsize <= 0):
        print(f'binsize of {str(binsize)} given. Must be > 0 and evenly divisible by 10 or = 1')
        return

    if (binsize % 10 != 0) and (binsize != 1):
        print(f'binsize of {str(binsize)} given. Must be evenly divisible by 10 or = 1')
        return

    if zoom:
        if maxZoomLevel < minZoomLevel:
            print(f'maxZoomLevel given as {str(maxZoomLevel)} which must ' +
                  f'be >= minZoomLevel given as {str(minZoomLevel)}')
            return

        if (maxZoomLevel % 10 != 0) or (minZoomLevel % 10 != 0):
            print(f'both maxZoomLevel given as {str(maxZoomLevel)} ' +
                  f'and minZoomLevel given as {str(minZoomLevel)} ' +
                  f'must be evenly divisible by 10')
            return
```

```
375
376     #Set size of displayed plot
377     if plotsize < 1 or plotsize > 5:
378         print(f'plotsize given as {str(plotsize)} must
     be between 1 and 5')
379         return
380     else:
381         setPlotSize(plotsize)
382
383     #Null values make it flunk out.
384     numNullValues = tDf[colName].isnull().sum()
385     if numNullValues > 0:
386         print(f'Warning: {numNullValues} null values
    detected in column. Removing for analysis')
387         tDf = tDf.dropna(subset=[colName], axis=0)
388
389     #Groupby and summarize dataframe
390     tDf = round(tDf[[colName]], 0).astype(int)
391     tDf[binColName] = [int(math.trunc(val / binsize
    ) * binsize) for val in tDf[colName]]
392     tDf = tDf.groupby(binColName).size().to_frame(
    binnedCountName).sort_values([binColName], ascending=
    False)
393     tDf.reset_index(inplace=True)
394
395     #Zoom to applicable level
396     if zoom:
397         tDf = tDf.loc[(tDf[binColName] >= minZoomLevel
    ) & (tDf[binColName] <= maxZoomLevel)]
398         tDf.reset_index(drop=True, inplace=True)
399
400     #Show me how it went
401     plt.xticks(rotation='vertical')
402     fig = sns.barplot(x=binColName, y=binnedCountName
    , data=tDf, palette="crest")
403     fig.set_xlabel('Summary column: {0} (binned at {1
    })'.format(colName, str(binsize)), fontsize=15)
404     fig.xaxis.labelpad=20
```

```
405     fig.set_ylabel('Document Count', fontsize=15)
406 
407     if zoom:
408         titleTail = '\nZoom factor [{0}:{1}]'.format(
    minZoomLevel, maxZoomLevel)
409     else:
410         titleTail = ''
411     fig.set_title(f'Data dispersion summary for: {
    colName} (binned at {str(binsize)}){titleTail}',
    fontsize=20)
412 
413     plt.show()
414 
415     if verbose:
416         exploreDataframe(tDf,numRecords=numRecords)
417 
418 
419 def classBalanceUndersample(df, columnName):
420     ttlColName = 'ttlCol'
421     tDf = df.copy()
422 
423     visualizer = ClassBalance()
424     visualizer.fit(tDf[columnName])
425     visualizer.show()
426 
427     tDfSize = tDf.groupby([columnName]).size().
    to_frame(ttlColName).sort_values(by=ttlColName)
428     tDfSize.reset_index(inplace=True)
429     sample_size = pd.to_numeric(tDfSize[ttlColName][0
    ])
430     # display(tDfSize.head(5))
431     sample_class = tDfSize[columnName][0]
432     print(f'Undersampling data to match min class: {
    str(sample_class)} of size: {sample_size}')
433     tDf = tDf.groupby(columnName, group_keys=False).
    apply(lambda x: x.sample(sample_size))
434     tDf.reset_index(drop=True, inplace=True)
435 
```

```
436     visualizer2 = ClassBalance()
437     visualizer2.fit(tDf[columnName])
438     visualizer2.show()
439 
440     return tDf
441 
442 
443 def displayClassBalance(df, columnName, showRecords=5
    ):
444     ttlColName = 'ttlCol'
445     tDf = df.copy()
446 
447     visualizer = ClassBalance()
448     visualizer.fit(df[columnName])  # Fit the data to
    the visualizer
449     visualizer.show()  # Finalize and render the
    figure
450 
451     tDfSize = tDf.groupby([columnName]).size().
    to_frame(ttlColName).sort_values(by=ttlColName)
452     tDfSize.reset_index(inplace=True)
453     tDfSize.head(5)
454 
```