FIT1043 Introduction to Data Science

Assignment 2

Name:Tan Yoong Kiat
Student ID: 32142773
Date: 27/4/2021

1. Introduction

Machine Learning has been a trending topic since the 20th century and is becoming more and more demanding in the world regarding which field. In this assignment, we will be dealing with machine learning model which is classifications on predicting marks on essay on several features. For example, number of character, stemmed word and average word length

Importing necessary libraries

StandardScaler() from sklearn.preprocessing

In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn import metrics
```

Read CSV File

In [2]:

```
essay_features = pd.read_csv("FIT1043-Essay-Features.csv")
#essay_features.head()
essay_features.head()
```

Out[2]:

	essayid	chars	words	commas	apostrophes	punctuations	avg_word_length	sentences
0	1457	2153	426	14	6	0	5.053991	16
1	503	1480	292	9	7	0	5.068493	11
2	253	3964	849	19	26	1	4.669022	49
3	107	988	210	8	7	0	4.704762	12
4	1450	3139	600	13	8	0	5.231667	24

In [3]:

```
features, label = essay_features.iloc[:, :-1], essay_features.iloc[:,[-1]]
```

In [4]:

```
import numpy as np
unique_elements, counts_elements = np.unique(label, return_counts=True)
print("Frequency of unique values:")
print(np.asarray((unique_elements, counts_elements)))
```

```
Frequency of unique values:
[[  1   2   3   4   5   6]
 [ 18 110 557 583  60   4]]
```

Descriptive Statistic

In [5]:

```
essay_features.describe()
```

Out[5]:

	essayid	chars	words	commas	apostrophes	punctuations	avg_w
count	1332.00000	1332.000000	1332.000000	1332.000000	1332.000000	1332.00000	1
mean	905.27027	2101.745495	424.485736	14.667417	8.141141	0.47973	
std	526.68760	865.963750	171.873730	10.920781	6.124520	1.27168	
min	0.00000	169.000000	36.000000	0.000000	2.000000	0.00000	
25%	442.75000	1527.250000	310.000000	7.000000	4.000000	0.00000	
50%	914.50000	2029.500000	411.000000	13.000000	6.000000	0.00000	
75%	1369.75000	2613.500000	525.000000	21.000000	11.000000	0.00000	
max	1799.00000	6142.000000	1170.000000	72.000000	51.000000	26.00000	

In [6]:

```
essay_features.corr()
```

Out[6]:

	essayid	chars	words	commas	apostrophes	punctuatic
essayid	1.000000	0.020961	0.021810	0.025137	-0.020580	-0.008
chars	0.020961	1.000000	0.991998	0.647711	0.437508	0.184
words	0.021810	0.991998	1.000000	0.647159	0.456932	0.189
commas	0.025137	0.647711	0.647159	1.000000	0.397497	0.206
apostrophes	-0.020580	0.437508	0.456932	0.397497	1.000000	0.123
punctuations	-0.008286	0.184317	0.189927	0.206741	0.123361	1.000
avg_word_length	-0.025634	0.236563	0.123142	0.126595	-0.060541	-0.002
sentences	0.016737	0.366223	0.422922	0.207493	0.153522	0.073
questions	0.019837	0.328230	0.336836	0.316261	0.261627	0.186
avg_word_sentence	-0.027987	-0.040719	-0.042562	-0.025439	-0.041249	0.001
POS	0.021576	0.992162	0.999907	0.648915	0.458441	0.189
POS/total_words	-0.024392	0.317080	0.305294	0.288851	0.239325	0.054
prompt_words	0.020400	0.948406	0.960853	0.656856	0.399875	0.154
prompt_words/total_words	-0.024958	-0.034214	-0.026602	0.128626	-0.126017	-0.097
synonym_words	0.016857	0.912003	0.924474	0.535154	0.384010	0.115
synonym_words/total_words	-0.023628	-0.286498	-0.273979	-0.316782	-0.203050	-0.188
unstemmed	0.017443	0.953534	0.948491	0.620890	0.418899	0.206
stemmed	0.018011	0.955315	0.950299	0.625511	0.420205	0.210
score	0.033463	0.683983	0.662091	0.525055	0.322052	0.157

1. Total 1332 of essays in this file.
2. The maximum word wrote for the essay is 1170 words.
3. The average score obtained by the 1332 people is 3.427 marks.
4. prompt_words/total_words may not relate to the score
5. Both unstemmed and stemmed are highly related to the score

2. Supervised Learning

Supervised Machine Learning

- Supervised Learning
- Part of Machine Learning & Artificial Intelligence
- Train algorithms in classifying data/predicitng outcomes by labelled datasets
- user insert data to the model/algorithms -> model/algorithms adjust weight and come out with a output
- Regression/Classifications/Support Vector Machine(SVM)/Random Forest Model

Notion of labelled data

- Have a set of unlabelled data, augmenting it with some meaningful tag that easy to know
- Process of having a set of raw data -> add a meaningful name/label (informative) -> model learn from it
- Datasets about Fruits:
  - Columns: length , weight , shape , percentage of vitamin C
  - Rows will be each unique fruit

Training and test datasets

Training datasets

- Datasets used to train the model
- Usually taking 80% or 70% of the dataset to train the model
- Data will split to features and labels

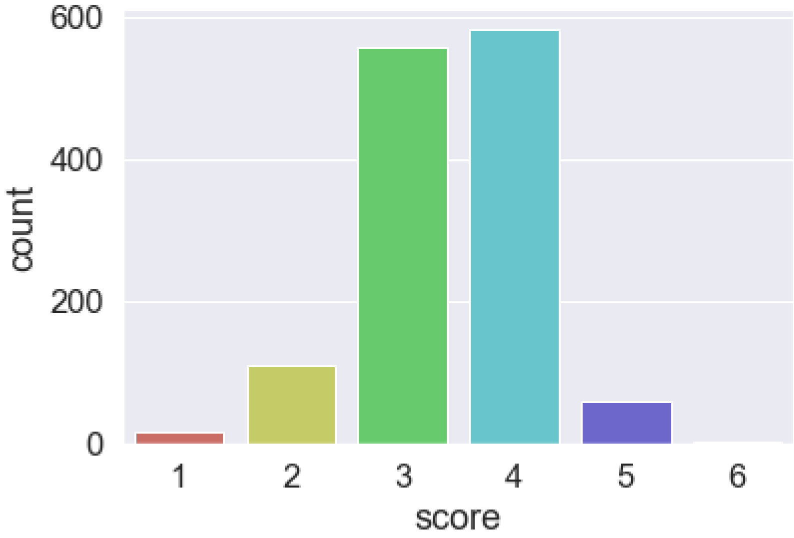
Test datasets

- Datasets to test the outcome of the prediction
- Independent with training datasets
- Same features columns but not having the label

Roughly visualize the score

In [7]:

```
sns.set(font_scale=1.5)
countplt=sns.countplot(x='score', data=essay_features, palette ='hls')
plt.show()
```



Codes of splitt features and labels

In [8]:

```
features, label = essay_features.iloc[:, :-1], essay_features.iloc[:,[-1]]

print(features)
print(label)
```

```
      essayid  chars  words  commas  apostrophes  punctuations  \
0        1457   2153    426      14            6             0
1         503   1480    292       9            7             0
2         253   3964    849      19           26             1
3         107    988    210       8            7             0
4        1450   3139    600      13            8             0
...       ...    ...    ...     ...          ...           ...
1327     1151   2404    467      16           10             0
1328     1015   1182    241       0           14             0
1329     1345   1814    363       5           11             0
1330      344   1427    287       5            8             0
1331     1077   2806    542      24            6             0

      avg_word_length  sentences  questions  avg_word_sentence         POS
\
0            5.053991         16          0          26.625000  423.995272
1            5.068493         11          0          26.545455  290.993103
2            4.669022         49          2          17.326531  843.990544
3            4.704762         12          0          17.500000  207.653784
4            5.231667         24          1          25.000000  594.652150
...               ...        ...        ...                ...         ...
1327         5.147752         22          0          21.227273  462.987069
1328         4.904564         16          0          15.062500  238.655462
1329         4.997245         13          3          27.923077  362.329640
1330         4.972125         13          1          22.076923  284.657277
1331         5.177122         22          3          24.636364  538.988889

      POS/total_words  prompt_words  prompt_words/total_words  synonym_wor
ds  \
0            0.995294           207                  0.485915            1
05
1            0.996552           148                  0.506849
77
2            0.994100           285                  0.335689            1
30
3            0.988828           112                  0.533333
62
4            0.991087           255                  0.425000            1
65
...               ...           ...                       ...
...
1327         0.991407           200                  0.428266            1
13
1328         0.990272            94                  0.390041
67
1329         0.998153           170                  0.468320            1
07
1330         0.991837           144                  0.501742
83
1331         0.994444           284                  0.523985            1
55

      synonym_words/total_words  unstemmed  stemmed
0                      0.246479        424      412
1                      0.263699        356      345
2                      0.153121        750      750
3                      0.295238        217      209
4                      0.275000        702      677
...                         ...        ...      ...
1327                   0.241970        529      519
1328                   0.278008        293      283
```

```
1329                  0.294766       427       415
1330                  0.289199       323       312
1331                  0.285978       596       575

[1332 rows x 18 columns]
      score
0         4
1         4
2         4
3         3
4         4
...     ...
1327      4
1328      3
1329      3
1330      3
1331      4

[1332 rows x 1 columns]
```

In [9]:

```
essay_features.corr()
```

Out[9]:

	essayid	chars	words	commas	apostrophes	punctuatio
essayid	1.000000	0.020961	0.021810	0.025137	-0.020580	-0.0082
chars	0.020961	1.000000	0.991998	0.647711	0.437508	0.1843
words	0.021810	0.991998	1.000000	0.647159	0.456932	0.1899
commas	0.025137	0.647711	0.647159	1.000000	0.397497	0.2067
apostrophes	-0.020580	0.437508	0.456932	0.397497	1.000000	0.1233
punctuations	-0.008286	0.184317	0.189927	0.206741	0.123361	1.0000
avg_word_length	-0.025634	0.236563	0.123142	0.126595	-0.060541	-0.0029
sentences	0.016737	0.366223	0.422922	0.207493	0.153522	0.0738
questions	0.019837	0.328230	0.336836	0.316261	0.261627	0.1869
avg_word_sentence	-0.027987	-0.040719	-0.042562	-0.025439	-0.041249	0.0012
POS	0.021576	0.992162	0.999907	0.648915	0.458441	0.1898
POS/total_words	-0.024392	0.317080	0.305294	0.288851	0.239325	0.0547
prompt_words	0.020400	0.948406	0.960853	0.656856	0.399875	0.1543
prompt_words/total_words	-0.024958	-0.034214	-0.026602	0.128626	-0.126017	-0.0977
synonym_words	0.016857	0.912003	0.924474	0.535154	0.384010	0.1155
synonym_words/total_words	-0.023628	-0.286498	-0.273979	-0.316782	-0.203050	-0.1886
unstemmed	0.017443	0.953534	0.948491	0.620890	0.418899	0.2066
stemmed	0.018011	0.955315	0.950299	0.625511	0.420205	0.2109
score	0.033463	0.683983	0.662091	0.525055	0.322052	0.1579

Codes of split of training and test datasets

In [10]:

```
features_train, features_test, label_train, label_test = train_test_split(features,labe
l,test_size=0.2,random_state=0)
print(features_train)
print(label_train)
```

```
      essayid  chars  words  commas  apostrophes  punctuations  \
568      1724   1881    378      15            2             0   
1273      895   5013    953      24           16             0   
49        657   1445    287       3            8             0   
1165     1468   1027    207       4            2             0   
1268      362   3207    663      27            7             0   
...       ...    ...    ...     ...          ...           ...   
763       110   1833    372       6            5             0   
835      1161   2473    457      23            6             0   
1216      489   2952    581      12           27             0   
559       243   3316    716      21           25             0   
684       513    591    124       3            3             0   

      avg_word_length  sentences  questions  avg_word_sentence         POS
\
568          4.976190         15          0          25.200000  374.994652   
1273         5.260231         32          6          29.781250  945.987368   
49           5.034843          8          2          35.875000  280.328554   
1165         4.961353          6          0          34.500000  203.990148   
1268         4.837104         33          3          20.090909  657.987879   
...               ...        ...        ...                ...         ...   
763          4.927419         25          0          14.880000  367.983740   
835          5.411379         21          0          21.761905  456.330403   
1216         5.080895         29          4          20.034483  578.324122   
559          4.631285         31          2          23.096774  712.991597   
684          4.766129          6          0          20.666667  120.322129   

      POS/total_words  prompt_words  prompt_words/total_words  synonym_wor
ds  \
568          0.992049           209                  0.552910            1
07   
1273         0.992642           454                  0.476390            2
15   
49           0.976755           129                  0.449477             
94   
1165         0.985460            98                  0.473430             
54   
1268         0.992440           322                  0.485671            1
89   
...               ...           ...                       ...             
...   
763          0.989204           213                  0.572581            1
16   
835          0.998535           243                  0.531729            1
07   
1216         0.995394           318                  0.547332            1
68   
559          0.995798           275                  0.384078            1
55   
684          0.970340            51                  0.411290             
35   

      synonym_words/total_words  unstemmed  stemmed  
568                    0.283069        445      429  
1273                   0.225603        750      750  
49                     0.327526        330      314  
1165                   0.260870        273      265  
1268                   0.285068        658      631  
...                         ...        ...      ...  
763                    0.311828        364      351  
835                    0.234136        534      521  
```

```
1216                     0.289157        597       579
559                      0.216480        750       750
684                      0.282258        172       170

[1065 rows x 18 columns]
      score
568       3
1273      5
49        3
1165      3
1268      4
...     ...
763       3
835       4
1216      4
559       3
684       2

[1065 rows x 1 columns]
```

In [11]:

```
un_ele, count_ele = np.unique(label_train, return_counts=True)
print("Frequency of unique values of label testing datasets: ")
print(np.asarray((un_ele, count_ele)))
```

```
Frequency of unique values of label testing datasets:
[[  1   2   3   4   5   6]
 [ 16  90 442 464  50   3]]
```

In [12]:

```
un_ele, count_ele = np.unique(label_test, return_counts=True)
print("Frequency of unique values of label testing datasets: ")
print(np.asarray((un_ele, count_ele)))
```

```
Frequency of unique values of label testing datasets:
[[  1   2   3   4   5   6]
 [  2  20 115 119  10   1]]
```

Classification

3a. Binary & Multi-Class Classification

Binary Classifications

- Binary meaning is 0 or 1 (False or True) in computer. From the word binary, the meaning of binary classfications is easy to know, which is classfying the input data into 2 groups.
- For example in the Titanic dataset, we can classfied the passengers into survived or not survived

Multi-Class Classifications

- Multi-Class classifications is a machine learning classfications that it consists of two or more output or classes
- For example given images of fruits, identify them into which type of fruit
- A machine learning model to predict the input data/classes and come out with 2 or more output

3b. Normalisation

Why Normalise Data

- Help Improve data cleaning
- Reduce Duplicating Data
- Improve speed and effiency of models
- Raw Data ranges widely hence ML algorithms can't work as it expected

In [13]:

```
scaler = StandardScaler()
```

In [14]:

```
# fit & transform training data
scaled_train = scaler.fit_transform(features_train)
# transform test data
scaled_test = scaler.transform(features_test)
```

3c. SVM

Explain SVM (must have some reference comparison to linear regression)

- supervised machine learning algorithms.
- classification or regression but mostly used in classifications
- Implemented by plotting data item as a point in the n-dimensional space,
  - n is number of features
  - with value of each feature being the value of a coordinate.
- Eg 2 features
  - A group of point gathered at a side, another group gather at a side
  - Seperated by a hyper-plane
- Perform non-linear classification by implementing different kernel in the codes
- SVM application
  - Speech Recognition
  - Cancer Diagnosis
  - Text Classification
  - Facial Expression

Kernel in SVM / SVR

- Kernel is used due to mathematicals functions used in SVM
- A kernel take input data and transform into a output by a series of formula
- Generally transform training set of data
  - Non-Linear decision to Linear decision
  - Higher number of dimensional space
- Default kernel is rbf
- Types of Kernel
  - Linear
  - Poly
  - Rbf
  - Sigmoid
  - Pre-Computed

Code to Build the Model by training dataset

In [15]:

```
sv = SVC(kernel='linear').fit(scaled_train,label_train)
```

```
C:\Users\KIAT\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using r
avel().
  y = column_or_1d(y, warn=True)
```

3d. Prediction

In [16]:

```
y_pred = sv.predict(scaled_test)
```

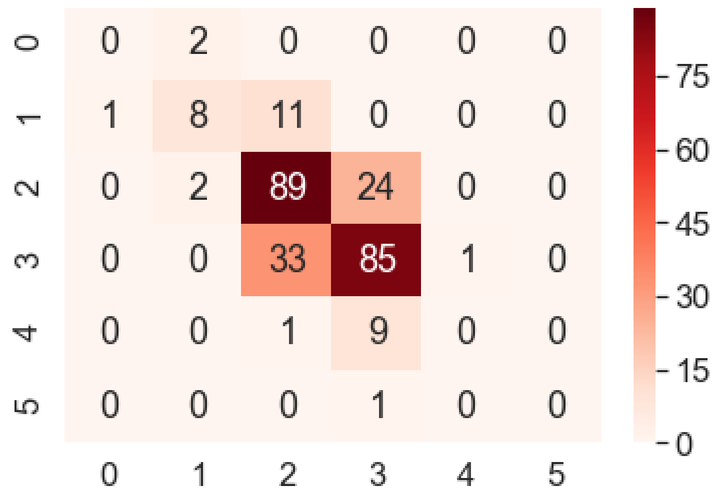
Confusion Matrix

- Also known as Error Matrix
- A table layout to visualize the performance of an algorithms
- Typically used in Supervised Learning
- A confusion matrix is a square matrix or a nxn matrix
  - Each column of the matrix shows the instances in the predicted class
  - Each row of the matrix show the instances in an actual class
- Diagonal is the correct prediction
- Terms of Confusion Matrix:
  - TP: True Positive
  - TN: True Negative
  - FP: False Positive
  - FN: False Negative
- Measuring with Confusion Matrix:
  - Precision: TP / (TP+FP)
  - Recall: TP / (TP+FN)
  - Accuracy: (TP+TN)/ (TP+TN+FP+FN)
  - Specificity: TN/ (TN+FP)

Confusion Matrix in plotted by seaborn (visualization)

In [17]:

```
cmatrix = metrics.confusion_matrix(label_test,y_pred)
# plt.figure(figsize=(8,8))
ax = sns.heatmap(cmatrix,annot=True,cmap='Reds')
ax.set_ylim(sorted(ax.get_xlim(), reverse=True))
plt.show()
```



Normal Confusion Matrix

In [18]:

```
O = metrics.confusion_matrix(label_test,y_pred)
print ('Confusion Matrix:')
print(O)
```

```
Confusion Matrix:
[[ 0  2  0  0  0  0]
 [ 1  8 11  0  0  0]
 [ 0  2 89 24  0  0]
 [ 0  0 33 85  1  0]
 [ 0  0  1  9  0  0]
 [ 0  0  0  1  0  0]]
```

Quadratic Weighted Kappa

- Also know as QWK
- Varies from -1 to 1
  - 1 is a perfect score (prediction and actual are the same)
  - -1 is worst (predictions are furthest away from actual)
  - Generally, a score above 0.6 is considered as a good score
- Calculated between known score and predicted score
- Good in imbalanced datasets & classifications problems

QWK Score

In [19]:

```
print(metrics.cohen_kappa_score(y_pred,label_test, weights ='quadratic'))
```

```
0.6215007866901376
```

4. Kaggle Submission

In [20]:

```
score_1 = essay_features[essay_features['score'] == 1]
score_2 = essay_features[essay_features['score'] == 2]
score_3 = essay_features[essay_features['score'] == 3]
score_4 = essay_features[essay_features['score'] == 4]
score_5 = essay_features[essay_features['score'] == 5]
score_6 = essay_features[essay_features['score'] == 6]
essay_features = score_1
essay_features = essay_features.append(score_2)
essay_features = essay_features.append(score_3[:150])
essay_features = essay_features.append(score_4[:150])
essay_features = essay_features.append(score_5)
essay_features = essay_features.append(score_6)
print(essay_features)
```

```
      essayid  chars  words  commas  apostrophes  punctuations  \
27        901    257     61       0            6             1
154       263    218     44       0            2             0
327       181    916    194       3            4             0
337       401   2254    457      19            9             1
341        88    389     78       4            2             0
...       ...    ...    ...     ...          ...           ...
1321      910   2409    479      25           15             0
294      1654   3850    767      31            8             1
758       450   3014    567      13            4             0
1030     1239   5881   1170      43           19             1
1119     1083   3096    580      21           10             1

      avg_word_length  sentences  questions  avg_word_sentence          PO
S  \
27           4.213115          3          4          20.333333    60.00000
0
154          4.954545          2          0          22.000000    43.00000
0
327          4.721649          8          1          24.250000   192.65277
8
337          4.932166         24          0          19.041667   454.66079
3
341          4.987179          1          0          78.000000    75.65765
8
...               ...        ...        ...                ...
...
1321         5.029228         25          2          19.160000   474.64849
8
294          5.019557         30         17          25.566667   761.98692
8
758          5.315697         36          3          15.750000   558.30965
1
1030         5.026496         74          8          15.810811  1158.98456
3
1119         5.337931         23          7          25.217391   575.98960
1

      POS/total_words  prompt_words  prompt_words/total_words  synonym_wor
ds  \
27           0.983607            22                  0.360656
20
154          0.977273            25                  0.568182
14
327          0.993056            90                  0.463918
51
337          0.994881           210                  0.459519
99
341          0.969970            40                  0.512821
22
...               ...           ...                       ...
...
1321         0.990915           224                  0.467641            1
14
294          0.993464           350                  0.456323            1
70
758          0.984673           236                  0.416226            1
34
1030         0.990585           543                  0.464103            2
66
1119         0.993086           244                  0.420690            1
```

```
30

      synonym_words/total_words  unstemmed  stemmed  score
27                     0.327869         81       81      1
154                    0.318182         64       64      1
327                    0.262887        245      240      1
337                    0.216630        490      457      1
341                    0.282051        106      106      1
...                         ...        ...      ...    ...
1321                   0.237996        527      510      5
294                    0.221643        750      750      6
758                    0.236332        696      680      6
1030                   0.227350        750      750      6
1119                   0.224138        681      666      6

[492 rows x 19 columns]
```

In [21]:

```
X, y = essay_features.iloc[:, :-1], essay_features.iloc[:,[-1]]
print(X)
print(y)
```

```
      essayid  chars  words  commas  apostrophes  punctuations  \
27        901    257     61       0            6             1
154       263    218     44       0            2             0
327       181    916    194       3            4             0
337       401   2254    457      19            9             1
341        88    389     78       4            2             0
...       ...    ...    ...     ...          ...           ...
1321      910   2409    479      25           15             0
294      1654   3850    767      31            8             1
758       450   3014    567      13            4             0
1030     1239   5881   1170      43           19             1
1119     1083   3096    580      21           10             1

      avg_word_length  sentences  questions  avg_word_sentence          PO
S  \
27           4.213115          3          4          20.333333    60.00000
0
154          4.954545          2          0          22.000000    43.00000
0
327          4.721649          8          1          24.250000   192.65277
8
337          4.932166         24          0          19.041667   454.66079
3
341          4.987179          1          0          78.000000    75.65765
8
...               ...        ...        ...                ...
...
1321         5.029228         25          2          19.160000   474.64849
8
294          5.019557         30         17          25.566667   761.98692
8
758          5.315697         36          3          15.750000   558.30965
1
1030         5.026496         74          8          15.810811  1158.98456
3
1119         5.337931         23          7          25.217391   575.98960
1

      POS/total_words  prompt_words  prompt_words/total_words  synonym_wor
ds  \
27           0.983607            22                  0.360656
20
154          0.977273            25                  0.568182
14
327          0.993056            90                  0.463918
51
337          0.994881           210                  0.459519
99
341          0.969970            40                  0.512821
22
...               ...           ...                       ...
...
1321         0.990915           224                  0.467641            1
14
294          0.993464           350                  0.456323            1
70
758          0.984673           236                  0.416226            1
34
1030         0.990585           543                  0.464103            2
66
1119         0.993086           244                  0.420690            1
```

```
30

      synonym_words/total_words  unstemmed  stemmed
27                     0.327869         81       81
154                    0.318182         64       64
327                    0.262887        245      240
337                    0.216630        490      457
341                    0.282051        106      106
...                         ...        ...      ...
1321                   0.237996        527      510
294                    0.221643        750      750
758                    0.236332        696      680
1030                   0.227350        750      750
1119                   0.224138        681      666

[492 rows x 18 columns]
      score
27        1
154       1
327       1
337       1
341       1
...     ...
1321      5
294       6
758       6
1030      6
1119      6

[492 rows x 1 columns]
```

In [22]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=6)
print(X_train)
print(y_test)
```

```
      essayid  chars  words  commas  apostrophes  punctuations  \
670      1632    794    156       4           10             0   
556       299   4173    829      41           17             2   
203       988    977    216       4            8             0   
150       941   2926    573      12           12             2   
107       981   2938    602      25            7             0   
...       ...    ...    ...     ...          ...           ...   
1110      399   1593    335      13            3             0   
809      1187   3339    638      21            3             1   
219      1120   1326    255       1            3             0   
164       904    778    172       6            4             0   
266       599   2156    425      20            9             0   

      avg_word_length  sentences  questions  avg_word_sentence         POS  
\
670          5.089744          5          0          31.200000  153.662281   
556          5.033776         42          6          19.738095  824.661010   
203          4.523148          8          0          27.000000  213.644860   
150          5.106457         32          3          17.906250  569.989492   
107          4.880399         32          0          18.812500  597.989983   
...               ...        ...        ...                ...         ...   
1110         4.755224         16          0          20.937500  331.658610   
809          5.233542         42          0          15.190476  633.321785   
219          5.200000          9          0          28.333333  253.664021   
164          4.523256          7          0          24.571429  170.325444   
266          5.072941         18          0          23.611111  422.655634   

      POS/total_words  prompt_words  prompt_words/total_words  synonym_wor
ds  \
670          0.985015            71                  0.455128             
42   
556          0.994766           398                  0.480097            1
95   
203          0.989097            75                  0.347222             
68   
150          0.994746           264                  0.460733            1
46   
107          0.993339           306                  0.508306            1
55   
...               ...           ...                       ...             
...   
1110         0.990026           187                  0.558209             
91   
809          0.992667           285                  0.446708            1
33   
219          0.994761           117                  0.458824             
60   
164          0.990264            75                  0.436047             
58   
266          0.994484           196                  0.461176            1
05   

      synonym_words/total_words  unstemmed  stemmed  
670                    0.269231        208      201  
556                    0.235223        750      750  
203                    0.314815        262      260  
150                    0.254799        639      618  
107                    0.257475        557      543  
...                         ...        ...      ...  
1110                   0.271642        396      382  
809                    0.208464        718      704  
```

```
219                     0.235294        311       302
164                     0.337209        225       217
266                     0.247059        490       475

[393 rows x 18 columns]
     score
317      5
173      4
38       4
225      4
320      3
..     ...
403      2
18       4
340      3
265      3
240      4

[99 rows x 1 columns]
```

In [23]:

```
lof = LocalOutlierFactor()
yhat = lof.fit_predict(X_train)
# select all rows that are not outliers
mask = yhat != -1
X_train, y_train = X_train.iloc[mask, :], y_train.iloc[mask]
print(X_train.shape, y_train.shape)
```

```
(353, 18) (353, 1)

C:\Users\KIAT\Anaconda3\lib\site-packages\sklearn\neighbors\lof.py:236: Fu
tureWarning: default contamination parameter 0.1 will change in version 0.
22 to "auto". This will change the predict method behavior.
  FutureWarning)
```

In [24]:

```
# define scaler used
#scaler = StandardScaler()
# fit & transform training data
scaled_Xtrain = scaler.fit_transform(X_train)
scaled_Xtest = scaler.transform(X_test)
```

In [25]:

```
sv = SVC(C=0.05,kernel='linear',gamma=100)
sv.fit(scaled_Xtrain,y_train)
y_pred = sv.predict(scaled_Xtest)
print('QWK score:',metrics.cohen_kappa_score(y_test, y_pred,weights="quadratic"))
```

```
QWK score: 0.7532219570405727

C:\Users\KIAT\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using r
avel().
  y = column_or_1d(y, warn=True)
```

In [26]:

```
cmatrix = metrics.confusion_matrix(y_test,y_pred)
# plt.figure(figsize=(8,8))
ax = sns.heatmap(cmatrix,annot=True,cmap='RdBu')
ax.set_ylim(sorted(ax.get_xlim(), reverse=True))
plt.show()
```

In [27]:

```
submission_features = pd.read_csv("FIT1043-Essay-Features-Submission.csv")
submission_features.shape
```

Out[27]:

```
(199, 18)
```

In [28]:

```
scaleX = scaler.transform(submission_features)
y_pred = sv.predict(scaleX)
```

In [29]:

```
import numpy as np
unique_elements, counts_elements = np.unique(y_pred, return_counts=True)
print("Frequency of unique values of the said array:")
print(np.asarray((unique_elements, counts_elements)))
```

```
Frequency of unique values of the said array:
[[ 2  3  4  5]
 [20 83 84 12]]
```

In [30]:

```
draft = pd.read_csv('99999999-YourName-1.csv')
draft.head()
```

Out[30]:

	essayid	score
0	1623	NaN
1	1143	NaN
2	660	NaN
3	1596	NaN
4	846	NaN

In [31]:

```
for i in range(len(draft)):
    draft.iloc[i,-1] = int(y_pred[i])
```

In [32]:

```
draft.head()
```

Out[32]:

	essayid	score
0	1623	4.0
1	1143	4.0
2	660	3.0
3	1596	4.0
4	846	4.0

In [33]:

```
draft['score']=draft.score.astype('int64')
draft.head()
draft = draft.set_index('essayid')
#draft = draft.droplevel()
```

In [34]:

```
sub = draft.to_csv('32142773-TanYoongKiat-1.csv')
```