Using Word Frequencies for Predicting Integrative Complexity Scores

CMPE 251 Data Analytics

Ryan Kinsella – 10194574

To Dr. Skilicorn

Hard deadline November 22nd, 2019

Contents

List of Figures

Introduction

American presidential speeches have been ranked in terms of integrative complexity (IC), their score for having more than one viewpoint. IC scores will range from 0-4, 4 being aware of many viewpoints. 148 records have been supplied with the most occurring word rates, with corresponding IC scores to match each speech. Each IC score will be attempted to be predicted using word rates from the 148 speeches, using both classification and regression models.

The goal is not to produce the best accuracies on this given data set, but given *any* new data set to be able to confidently predict the IC score. Overfitting to specificities of the training data set is tempting, but will not yield accurate future test results. Useful words most correlated to the IC score will be determined and used as attributes, using univariate attribute selection. Certain "structure words" that could objectively be placed in either a minimum or maximum IC score speech should be removed as attributes to be used as predictive to generalize future predictions.

The all 3000 words data set will create overfitting models. All measures and insights will be taken to avoid complex overfitting models, while still producing reasonably improved accuracy results from the original 3000-word data set. Results will be compared using the 3000-word data set and the useful word data set, to see if the attribution selection process will aid in prediction accuracy.

1. Attribute/word selection process

Ideal attributes will be selected to build a generalized, non-overfitting model. Using word frequencies alone can cause predictive models to be easily susceptible to overfitting based off of the specificities of the training data set. The issue with overfitting is that when given a new set of test data, the model will perform poorly with low accuracies. To avoid overfitting, only words that are objectively correlatable to the IC value for a new random set of test data should be included in the model generation. For example, within the training set there could be several records with very low IC scores that have high word rates of the word "it". As a human one can understand that "it" should not be used in predicting IC scores since there should not be a correlation, as well as other words such as simple nouns like "Utah". The process of selecting which words to include into the KNIME models will be further explained below.

1.1. Univariate attribute selection

Determining which words are most correlated to IC scores was accomplished using a jupyter notebook with the imported libraries pandas, numpy, and sklearn for the selection of useful words for prediction, and libraries seaborn and matplotlib to visualize the data. All code used to generate csv files and figures can be found in the Appendix. When using sklearns "SelectPercentile" [1], CSV file rownormalizeddocword was used in place of ICperSpeech_WordMAtrix to normalize word frequencies such that word rates are used, to prevent longer speeches indirectly having an impact on the weights for IC predictions.

Univariate attribute selection works by selecting the best attributes (in this case, words) based on univariate statistical tests. It can be seen as a preprocessing step to an estimator, which will be used as input attributes into the KNIME predictors. Using sklearns univariate attribute selection method SelectPercentile removes all but a user-specified highest scoring percentage of attributes, which was determined to be 10% to reduce enough of the noise of the structure words (see 1.2. Structure words), but keep enough attributes to achieve good prediction accuracy. This will filter and remove *most* structure words because structure words should not (in theory, however because of the specificities of the training data still will) have any correlation between the IC scores.

This SelectPercentile method was used as an alternative to a clustering method of sorting words into useful words vs structure words because of issues of determining the number of clusters, choosing clusters based of sparsity, high dimensionality for the given dataset, amongst other challenges that may not directly consider correlation. Function SelectPercentile also would have mathematically superior attribute selection than a clustering method since the imported library is designed to sort and assign in order attributes with the highest correlation to the target(s). For predicting target attributes correlation is most important, therefore it was determined that using this library was superior to a clustering technique for prediction accuracy. In the following sub-sections methods of not overgeneralizing training data will be discussed and implemented after using SelectPercentile to create a subset of data to use for the models.

1.2. Structure words

There are some words (such as the first four words in this sentence, or are simple nouns such as Utah) that are objectively not relevant to the IC rating in a speech and are used primarily and only for sentence structure, which will be referred to henceforth as structure words. If these structure words are used in building the model, the training set will be overfitting to the specificities of the training data and will not perform well on new test data. Therefore, only the top 10% of words that are most correlated to the IC scores were chosen to be used as the model attributes (words/attributes) to use. 10% was chosen to receive enough strongly correlated attributes that the models can have enough data to predict, while attempting to filter out structure words. This should greatly limit, but not completely negate the amount of structure words input into the model.

1.3. Useful word attribute selection: filtering out structure words

As previously mentioned, only the top 10% of words that have the strongest correlation between IC scores will be used. Of this top 10%, the non-structured words in this group will henceforth be referred to as useful words, or only_useful_words. In code plot_stats.ipynb, rownormalizeddoc.csv is approximately sorted by high word frequencies to low word frequencies, as shown in the Figure 1 below with the sum of the word rates as a function of each word.

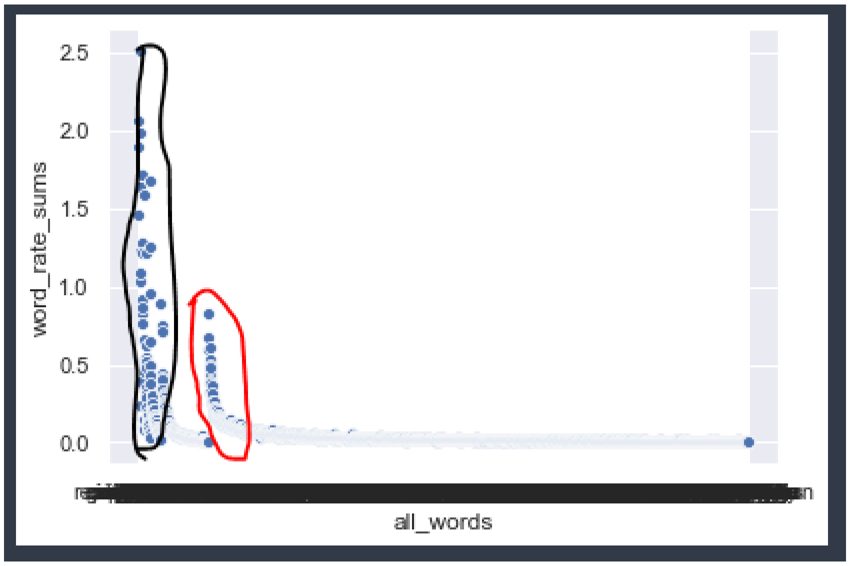
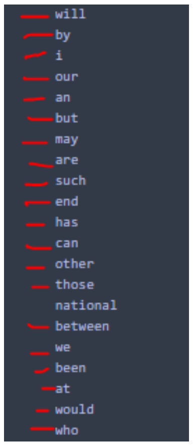


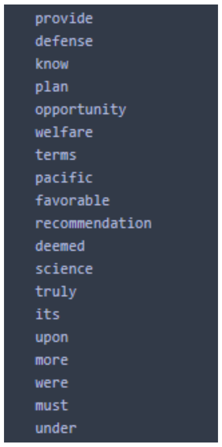
Figure 1: Word rate sums distribution in rownormalizeddoc.csv.

The majority of the first several words with high frequencies will be words that are used for sentence structure, such as the word "by", circled in black in Figure 1 above. The second peak in Figure 1 above circled in red after viewing the raw data, were not determined to be structure words so will be excluded from any filtering. On the first run through with the top 10% correlated words, there remain several words with high frequencies still that stick out as structure words. Regardless of the top percentage of useful words chosen, there will inevitably be specific structure words that correlate to the IC score as an output from the training data. Note that keeping structure words as training data would improve test result accuracy for this dataset, but this would be due to overfitting and any new datasets test cases would not perform as accurately. To avoid overfitting, structure words should not be used as attributes for the model. Once the useful words were determined, still the first 30 (30 was chosen after looking at the data set) words needed to be filtered out because all of these words are high frequency and the majority of these words are structure words, shown in code select_best_attributes.ipynb and below in Figure 2. Shown below in Figure 2 is the output of the top 10% correlated attributes before filtering with a red dashed line beside to indicate the structure words, and below in Figure 23 is after filtering out the first 30 words which the majority were structure words.



```
will
by
i
our
an
but
may
are
such
end
has
can
other
those
national
between
we
been
at
would
who
```

Figure 2: Structure words to be filtered out.



```
provide
defense
know
plan
opportunity
welfare
terms
pacific
favorable
recommendation
deemed
science
truly
its
upon
more
were
must
under
```

Figure 3: Output of only_useful_words.csv.

There is no way to automate the process of removing simple nouns that should have no impact on the IC score (such as Utah). Leaving these in would have overfitting potential, so these were manually removed from the final useful words' csv file, named "only_useful_words.csv". Structure words were only removed if they could be used objectively equally for both a speech with an IC score of 0 or 4. This manual removal process was only searching from a total of 270 words removing to 248 words, which was much quicker than doing so for all 3000 words. Removing these words does not aid in prediction, but does allow the data to become more generalized to prevent overfitting to the training data. The file only_useful_words.csv contains no structure words, and will be input into KNIME models in the results section.

A heatmap has been constructed for the contents of only_useful_words.csv in script select_best_attributes.ipynb, shown below in Figure 4. From left to right the heatmap takes every 10th word from the pandas DataFrame. This figure describes that words with very low word rates, such as the ones on the right side of the figure, can still have a greater than 10% correlation to the IC value.

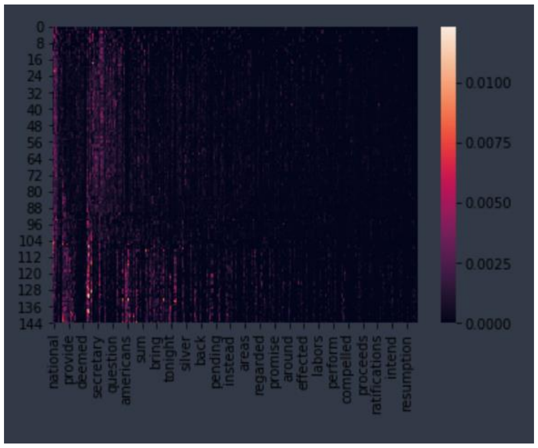


Figure 4: Heatmap of only_useful_words.csv.

1.4. Correlation

KNIME's Rank Correlation node plots the Pearson's Correlation Coefficient. A value of 1 is a strong positive correlation, a value of -1 is a strong negative correlation, and a value of 0 is no correlation. Thus in the plot, blue and red are both strong correlations, while white indicates no correlation. The KNIME workflow for using Rank Correlation is shown below in Figure 5.

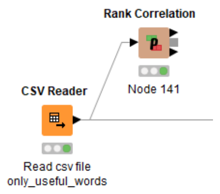


Figure 5: KNIME workflow rank correlation.

The outputs of rank correlation are described in the figures below. Figure 6 below shows that using only_useful_words.csv keeps words that have a strong correlation to the IC value, seen as outlined in black to the far right. Both strong positive (blue) and negative (red) values can both be useful for predicting IC, and as shown below there is a pattern of strong correlations in only_useful_words.csv. Figure 67 below to the right is the correlation read using all 3000 words. Words such as "deficit" or "young" or "occurred" are whiteish with nearly 0 correlations to IC, which will be filtered out from SelectPercentile and not included in only_useful_words.csv.

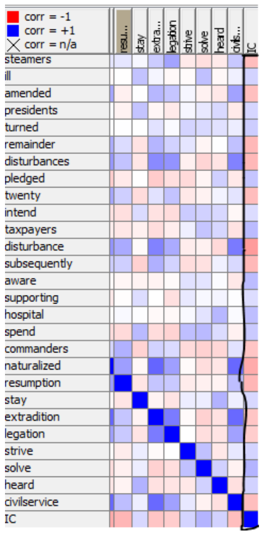


Figure 6: IC correlation for only_useful_words.csv.

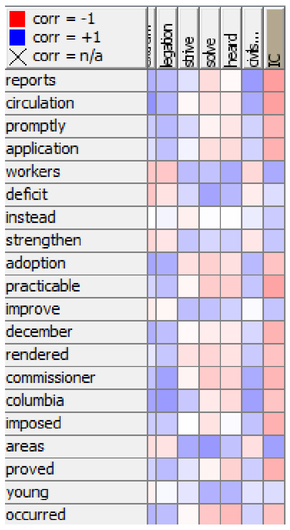


Figure 7: IC correlation for all 3000 words.

2. Prediction design choices

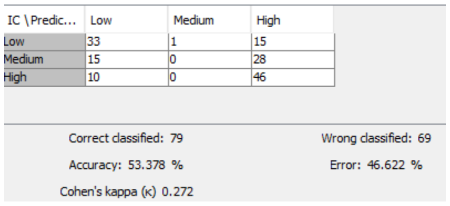
All predictions were executed using analytics platform KNIME and nodes where concepts were learned through the course content, for both classification and regression. Normalizing the values consistently decreased the prediction accuracy, therefore was omitted from the KNIME non-neural network model setup. Predictions will be from the original entire rownormalizeddocword.csv with 3000 words, henceforth referred to as all 3000 words, and compared to the useful words data set only_useful_words.csv. For each classification/regression type, two generated models will be compared: using all 3000 words in rownormalizeddocword.csv, and using the only_useful_words.csv Each model generated will be created with the same parameters for both data sets such that prediction accuracy can be compared.

2.1. Partitioning method

Cross validation is a very useful method of partitioning the dataset. Instead of creating a single model through partitioning the data into 80% training 20% test, it creates a model from the averages of a set of n-fold models. This reduces the likelihood that the model performance results were falsely high or low due to the data that was selected to train on.

The dataset received is short and wide, with few rows and twenty times the number of columns, which is unfortunate since cross validation is most effective for long datasets that can be partitioned into many folds. Using several folds may decrease model performance since models will be created only using limited records as data. Testing was completed using two, three, and four folds. When implemented in only_useful_words, using any more than two folds would cause the model to almost never predict the medium class (see 2.2. Equally sized bins for IC scores), shown below in the columns of Figure 8 and

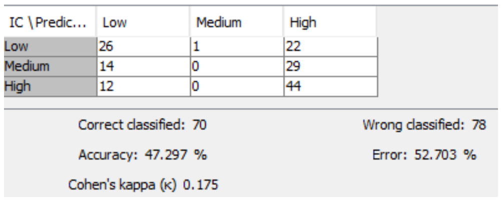
Figure 9 for a total of *one* medium prediction each. Figure 10 below shows that medium is predicted a total of 26 times when using two folds.



IC \ Predic...	Low	Medium	High
Low	33	1	15
Medium	15	0	28
High	10	0	46

Correct classified: 79 Wrong classified: 69
Accuracy: 53.378 % Error: 46.622 %
Cohen's kappa (κ) 0.272

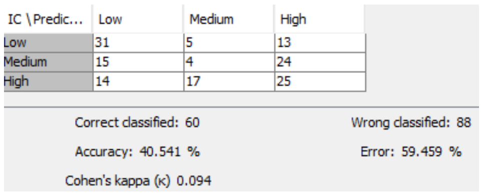
Figure 8: Three folds, only_useful_words.csv.



IC \ Predic...	Low	Medium	High
Low	26	1	22
Medium	14	0	29
High	12	0	44

Correct classified: 70 Wrong classified: 78
Accuracy: 47.297 % Error: 52.703 %
Cohen's kappa (κ) 0.175

Figure 9: Four folds, only_useful_words.csv.

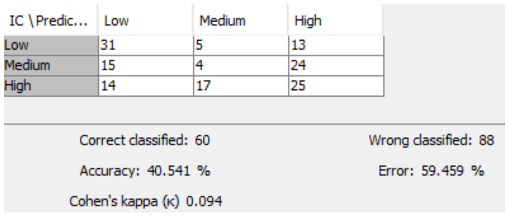


IC \ Predic...	Low	Medium	High
Low	31	5	13
Medium	15	4	24
High	14	17	25

Correct classified: 60 Wrong classified: 88
Accuracy: 40.541 % Error: 59.459 %
Cohen's kappa (κ) 0.094

Figure 10: Two folds predicts medium, only_useful_words.csv.

This was not a trend in the all 3000 words model, where at 4 folds there were 43 predictions total for medium, shown in the Figure 11 below.



IC \ Predic...	Low	Medium	High
Low	31	5	13
Medium	15	4	24
High	14	17	25

Correct classified: 60 Wrong classified: 88
Accuracy: 40.541 % Error: 59.459 %
Cohen's kappa (κ) 0.094

Figure 11: Four folds, all 3000 words.

When using more than two folds, there may not be enough records and attributes for each fold causing them to create underfitting predictions without ever predicting the medium class. The attributes included in the only_useful_words are the strongest correlated to the IC scores, so there would need to be many records of training to learn when different combinations of these attributes will result in the medium class, instead of just the extreme strong correlation cases determining the high and low classes. The all 3000 words data contains enough variation in correlation to consistently predict the medium class, seen above in Figure 11: Four folds, all 3000 words.Figure 11. This demonstrates the importance of including enough records for a training set. The results discussed in 3.1.1. SVM using all 3000 words and 3.1.2. SVM using only_useful_words.csv contains evidence that averaging the models will be a fair assessment of model performance. After testing, cross validation will be utilized on this short dataset with two folds for both model-sets all 3000 words and only_useful_words to keep accuracy results consistent.

2.2. Equally sized bins for IC scores

For classification IC was binned as evenly as possible, with low < 1.6, high >= 1.9, and medium covering the rest. Since the IC data was given as estimations usually to one decimal place, there was no clean cut-off for perfectly equal sized bins. The best even bins included items of medium at 55, low at 49, and high at 44, shown in the Figure 12 below.

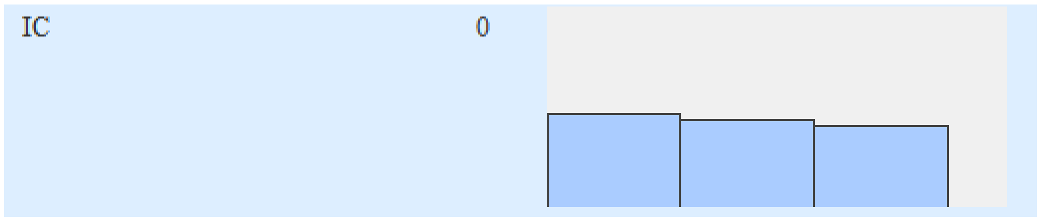


Figure 12: IC scores into class bins medium (55), low (49), and high (44) respectively.

The classes were approximately a third of the size of the records each because it avoids having the issue with a majority classifier, so predictions could be separable and distinct. A poor choice for the class boundaries would be to make the medium class values cover 120 of the 148 records, since a simple majority predictor without much insight could often predict the medium class correctly with 81% accuracy.

3. Classification results

All classifications were achieved using analytics platform KNIME. Three IC score approximately equal classes will be predicted: low, medium, and high.

3.1. Classification: SVM

A support vector machine (SVM) is a top-tier predictor in predicting 2-3 classes. An SVM is a kernelized maximum-margin hyperplane classifier, that performs well in both classification and runtime. It uses the kernel trick to reuse dot product calculations to determine the distance between points, as well as increasing dimensionality to improve prediction accuracy.

3.1.1. SVM using all 3000 words

A KNIME workflow for an SVM predictor has been constructed below in Figure 13. The prediction accuracy is also shown below in Figure 14.

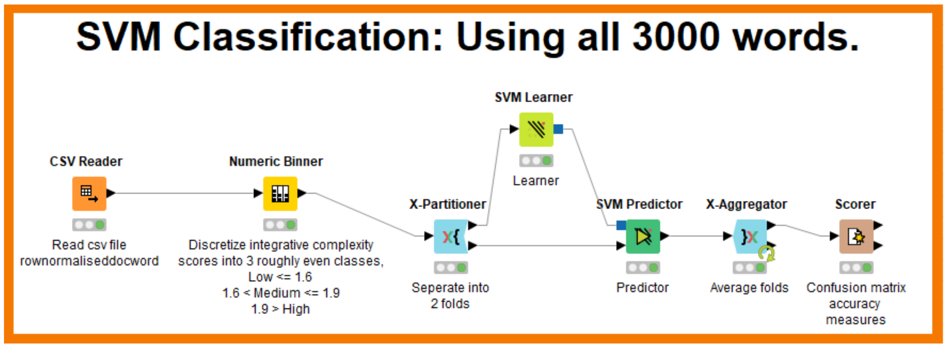
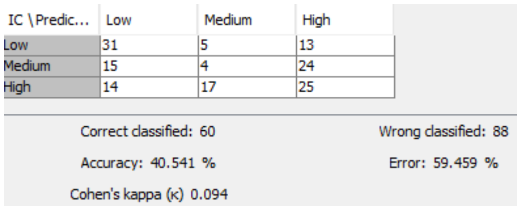

SVM Classification: Using all 3000 words.
SVM Learner
Learner
CSV Reader
Read csv file rownormaliseddocword
Numeric Binner
Discretize integrative complexity scores into 3 roughly even classes, Low <= 1.6 1.6 < Medium <= 1.9 1.9 > High
X-Partitioner
Seperate into 2 folds
SVM Predictor
Predictor
X-Aggregator
Average folds
Scorer
Confusion matrix accuracy measures

Figure 13: SVM classification: using all 3000 words.



IC \ Predic...	Low	Medium	High
Low	31	5	13
Medium	15	4	24
High	14	17	25

Correct classified: 60 Wrong classified: 88

Accuracy: 40.541 % Error: 59.459 %

Cohen's kappa (κ) 0.094

Figure 14: SVM all words prediction results.

These results will be compared below in 3.1.2. SVM using only_useful_words.csv. Note before using cross validation for using partitioning of regular 80% training 20% test data, the accuracy result was 33.33%. Cross-validation was able to average this falsely low result.

3.1.2. SVM using only_useful_words.csv

A KNIME workflow for an SVM predictor has been constructed below in Figure 15. The prediction accuracy is also shown below in Figure 16.

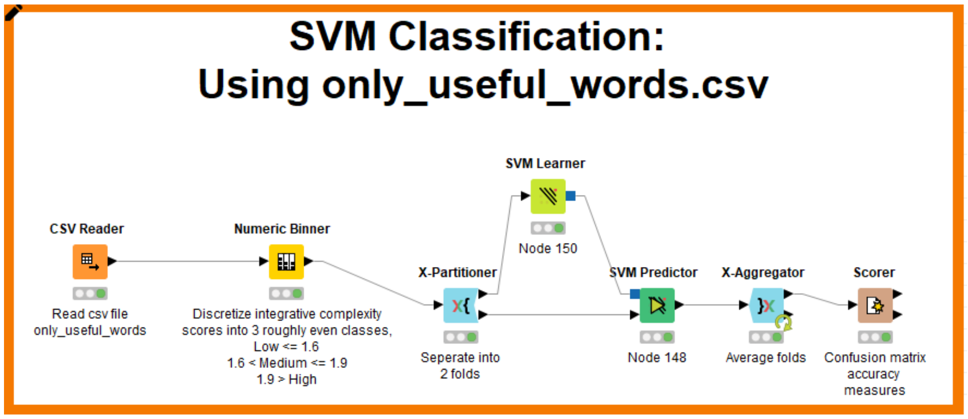
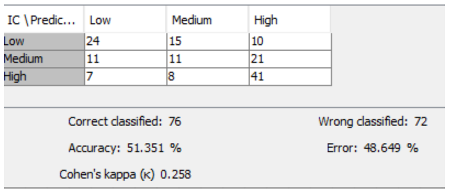

SVM Classification: Using only_useful_words.csv
SVM Learner
Node 150
CSV Reader
Read csv file only_useful_words
Numeric Binner
Discretize integrative complexity scores into 3 roughly even classes, Low <= 1.6 1.6 < Medium <= 1.9 1.9 > High
X-Partitioner
Seperate into 2 folds
SVM Predictor
Node 148
X-Aggregator
Average folds
Scorer
Confusion matrix accuracy measures

Figure 15: SVM classification: using only_useful_words.csv.



IC \ Predic...	Low	Medium	High
Low	24	15	10
Medium	11	11	21
High	7	8	41

Correct classified: 76 Wrong classified: 72

Accuracy: 51.351 % Error: 48.649 %

Cohen's kappa (κ) 0.258

Figure 16: SVM only_useful_words.csv prediction results.

There is an increase in prediction accuracy of 11% from the all 3000 words model to the only_useful_words model. Narrowing the scope of the words to only include useful words increased the test accuracy of the classification model for SVM. Note before using cross validation for using partitioning of regular 80% training 20% test data, the accuracy result was 53.33%. Cross-validation was able to average this falsely high result.

3.2. Classification: Random Forest

A random forest is an ideal predictor for determining a subset of attributes to include, and which attributes are irrelevant for predictive power. As described in 1. , this was a pre-processing step for only_useful_words. In classification this is achieved using a large number of decision trees (100 for both models), each using only some of the records/attributes of the dataset to participate in a voting process. For classification problems, there is enough attribute data in only_useful_words to generate a powerful random forest predictor. Advantages of using a random forest include runtime, low variance and bias, high accuracy, effective for large numbers of attributes, and generally avoids overfitting.

3.2.1. Random forest using all 3000 words

A KNIME workflow for a random forest predictor has been constructed below in Figure 17. The prediction accuracy is also shown below in Figure 18.

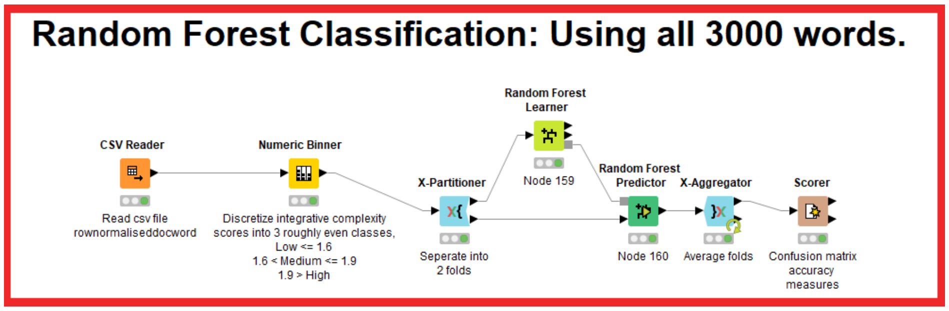
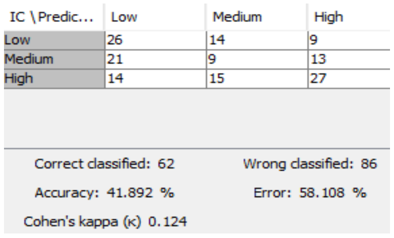

Random Forest Classification: Using all 3000 words.
CSV Reader
Read csv file rownormaliseddocword
Numeric Binner
Discretize integrative complexity scores into 3 roughly even classes, Low <= 1.6 1.6 < Medium <= 1.9 1.9 > High
X-Partitioner
Seperate into 2 folds
Random Forest Learner
Node 159
Random Forest Predictor
Node 160
X-Aggregator
Average folds
Scorer
Confusion matrix accuracy measures

Figure 17: Random forest classification: using all 3000 words.



IC \ Predic...	Low	Medium	High
Low	26	14	9
Medium	21	9	13
High	14	15	27

Correct classified: 62 Wrong classified: 86

Accuracy: 41.892 % Error: 58.108 %

Cohen's kappa (κ) 0.124

Figure 18: Random forest all words prediction results.

These results will be further compared below in 3.2.2. Random forest using only_useful_words.csv.

3.2.2. Random forest using only_useful_words.csv

A KNIME workflow for a random forest predictor has been constructed below in Figure 19. The prediction accuracy is also shown below in Figure 20.

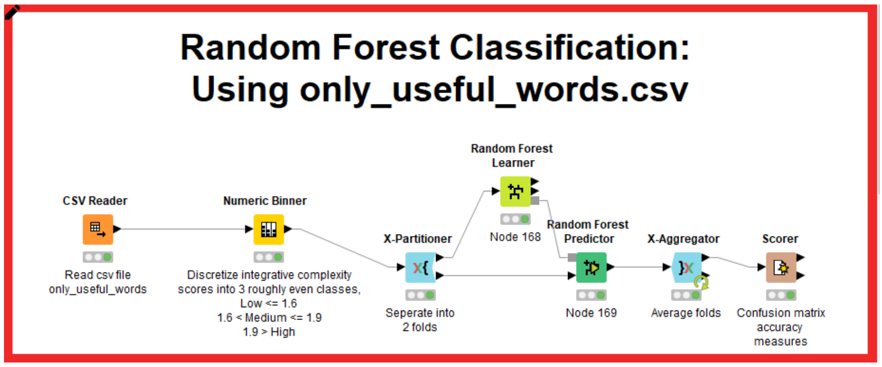
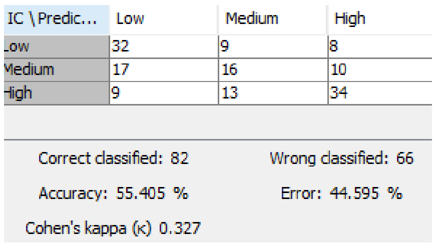

Random Forest Classification:
Using only_useful_words.csv
CSV Reader
Read csv file only_useful_words
Numeric Binner
Discretize integrative complexity scores into 3 roughly even classes, Low <= 1.6 1.6 < Medium <= 1.9 1.9 > High
X-Partitioner
Seperate into 2 folds
Random Forest Learner
Node 168
Random Forest Predictor
Node 169
X-Aggregator
Average folds
Scorer
Confusion matrix accuracy measures

Figure 19: Random forest classification: using only_useful_words.csv.



IC \ Predic...	Low	Medium	High
Low	32	9	8
Medium	17	16	10
High	9	13	34

Correct classified: 82 Wrong classified: 66

Accuracy: 55.405 % Error: 44.595 %

Cohen's kappa (κ) 0.327

Figure 20: Random forest only_useful_words.csv prediction results.

There is an increase in prediction accuracy of 13% from the all 3000 words model to the only_useful_words model. The results of the random forest performed slightly better than the SVM in terms of overall accuracy for both models generated, a 2% increase for the all 3000 words model and a 4% increase for the only_useful_words model. Having increased similar increased performances using the words from only_useful_words across two models, the only_useful_words are well suited to classify the IC score.

4. Regression results

The discretization/binning of the IC scores from the classification is removed, to enable a regression model of predicting the IC score. Regression aims to construct a line with the least error between points. The accuracy of regression in this report will be measured using two of the most standard error measures: root mean squared error and mean absolute error. Root mean squared error will be large if there are many outlier points very far from the regression line, where mean absolute error will be the average the total distance of all points to the line. Similarly to the classification models, all regressions were achieved using analytics platform KNIME.

4.1. Regression: neural network

A neural network is a great predictor for a regression problem. It uses several internal nodes, which become activated after some threshold is reached using weighted inputs. It uses hidden layers to increase the complexity of the prediction. Neural networks can become exponentially computationally expensive as input attributes increase. Taking the top 10% correlated attributes in only_useful_words will make this runtime a fraction of the all 3000 words input.

The same number of hidden layers/neurons was chosen for all 3000 words and only_useful_words for consistency purposes. As a rule of thumb described in class, the number of neurons per hidden layer should be the square root of the attributes. This was implemented for only_useful_words with 248 attributes, resulting in 15 neurons per layer. After some testing with the only_useful words model, 15 neurons per layer was the number that gave the local loss minimum. Using too many hidden layers and neurons can create complex representations of the dataset that would be subject to overfitting [2]. Avoiding extremely complex overrepresenting deep models, 3 hidden layers will be used. To ensure that the results from all 3000 words and only_useful_words are consistent with each other, the same initial random seed will be used.

4.1.1. neural network using all 3000 words

A KNIME workflow for a neural network predictor has been constructed below in Figure 21. The prediction accuracy is also shown below in Figure 22.

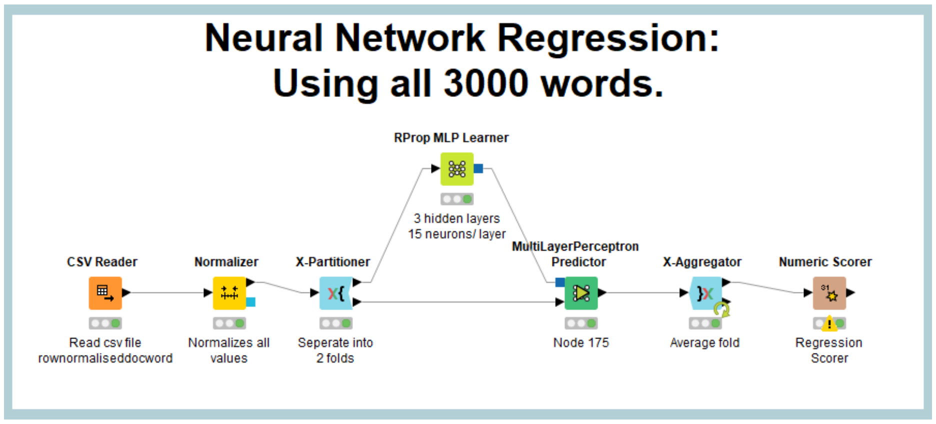
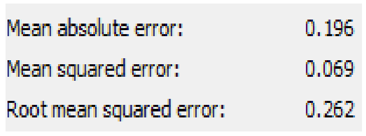


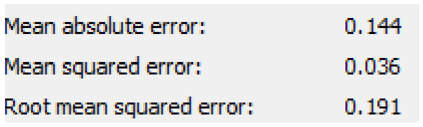
Figure 21: Neural network regression: using all 3000 words.



Mean absolute error:	0.196
Mean squared error:	0.069
Root mean squared error:	0.262

Figure 22: Neural network all 3000 words 3 hidden layers, 15 neurons per layer prediction results.

Additionally, using in class rule of thumb: number of neurons per hidden layer should be the square root of the attributes was tested, having 55 neurons per layer for all 3000 words. Figure 23 below shows the output of the same setup, this time using 55 neurons per layer.



Mean absolute error:	0.144
Mean squared error:	0.036
Root mean squared error:	0.191

Figure 23: Neural network all 3000 words 3 hidden layers, 55 neurons per layer prediction results.

These results will be compared below in 4.1.2. neural network using only_useful_words.csv.

4.1.2. neural network using only_useful_words.csv

A KNIME workflow for a neural network predictor has been constructed below in Figure 24. The prediction accuracy is also shown below in Figure 25.

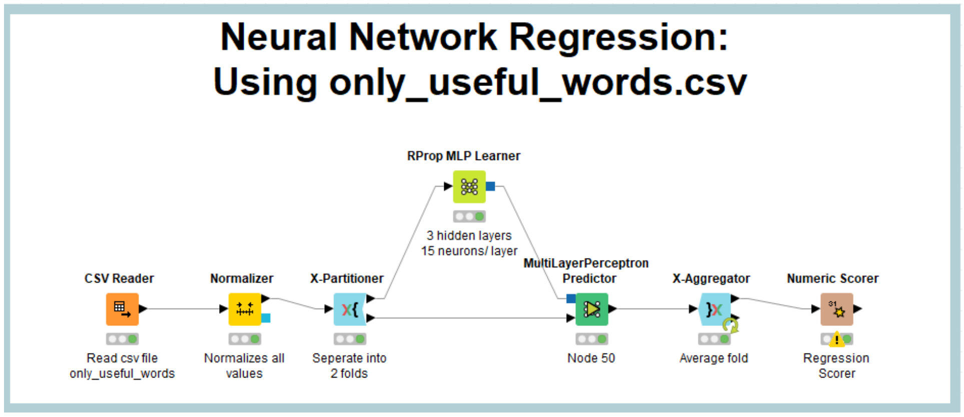
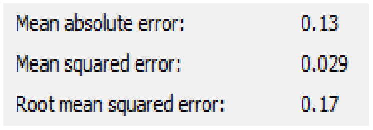


Figure 24: Neural network regression: using only_useful_words.csv.



Mean absolute error:	0.13
Mean squared error:	0.029
Root mean squared error:	0.17

Figure 25: Neural network only_useful_words.csv 3 hidden layers, 15 neurons per layer prediction results.

For using the same hidden layers and neurons for both models, there was an overall loss reduction in the only_useful_words model. The root mean squared error in the all 3000 words model was 0.262, and was decreased in the only_useful_words model to 0.170 for a performance increase of 35%. The mean absolute error in all 3000 words was 0.196, and was decreased in the only_useful_words model to 0.130 for a performance increase of 34%.

Additionally, the only_useful_words model slightly outperformed the optimized 55 neurons per layer all 3000 words model. The mean absolute error of the only_useful_words model was 0.014 less, and the root mean squared error was 0.021 less. Despite the long computational runtime for an optimized all 3000 words model, the only_useful_words still achieved the better error performance.

4.2. Regression: simple regression tree

A second regression problem is chosen to compare the results of the neural network. A simple regression tree using XGBoost was not the first choice for a regression prediction problem. As discussed in class when using trees on their own as predictors errors are often higher than using other “superior” methods, such as neural networks or random forests. The other methods, however, could not be chosen. A random forest regressor could not perform with the only_useful_words because there were too few attributes to create meaningful predictions, and were giving mean absolute errors of 2.6. Gradient boosted trees were not used since they were not covered in class. Linear/polynomial regression was considered, but not utilized since the patterns of the regression line are unknown and would vary between the all 3000 words model and only_useful_words model. If using polynomial regression, the line of best fit has potential for overfitting if the degree of the polynomial is higher than what the data is representing. In summary, when choosing a regression model, a simple regression tree is not the first choice, but was best suited for this task. A simple regression tree, however, did perform meaningful results in comparing all 3000 words to only_useful_words, described below.

4.2.1. Simple regression tree using all 3000 words

A KNIME workflow for a neural network predictor has been constructed below in Figure 26. The prediction accuracy is also shown below in Figure 27.

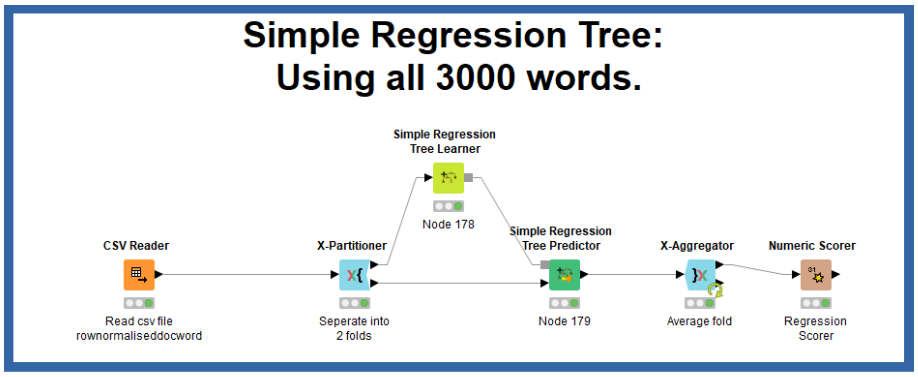
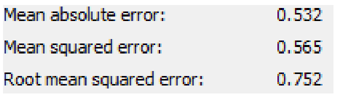


Figure 26: Simple regression tree: using all 3000 words.



Mean absolute error:	0.532
Mean squared error:	0.565
Root mean squared error:	0.752

Figure 27: Simple regression tree all 3000 words prediction results.

These results will be compared below in 4.2.2. Simple regression tree using only_useful_words.csv.

4.2.2. Simple regression tree using only_useful_words.csv

A KNIME workflow for a neural network predictor has been constructed below in Figure 28. The prediction accuracy is also shown below in Figure 29.

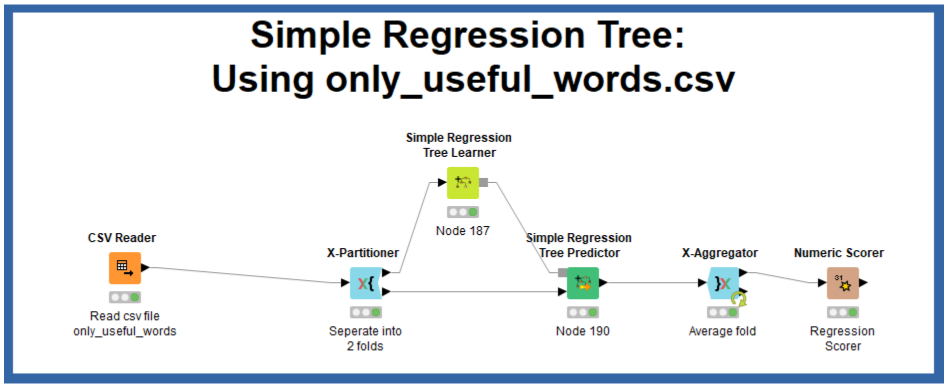
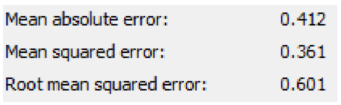


Figure 28: Simple regression tree: using only_useful_words.csv.



Mean absolute error:	0.412
Mean squared error:	0.361
Root mean squared error:	0.601

Figure 29: Simple regression tree only_useful_words.csv prediction results.

Similar to the neural network regression, the simple regression tree only_useful_words outperformed the all 3000 words model. The root mean squared error in the all 3000 words model was 0.752, and was decreased in the only_useful_words model to 0.601 for a performance increase of 20%. The mean absolute error in all 3000 words was 0.532, and was decreased in the only_useful_words model to 0.412 for a performance increase of 23%.

The error performance results from the simple regression tree models were worse than the neural network models as expected. Similarly to classification, using an optimal subset of strongly correlated attributes will help optimize predictions and regression loss.

Conclusion

IC scores were able to be predicted based off of simply word frequencies alone. For an approximately three equal binned class, the best classification model was a random forest predictor with 55% accuracy. The best regression model was a neural network, with a root mean squared error of 0.170. It is concluded that word frequencies as an attribute is difficult to build generalized models with high predictive power.

Choosing the top 10% of correlated attributes to the target was able to consistently outperform a model using all attributes, for both classification and regression. The classification models improved accuracies by at least 11% each, using two-fold cross validation to ensure that the model results were averaged. The random forest model proved to be the best at prediction accuracy for the three classes. The two regression model types for the all 3000 words were outperformed by the only_useful_words models by at least 20%. The neural network model performed the best, with the only_useful_words model reducing errors from the all 3000 words model by at least 34%.

Not only did using only the useful words have an impact on prediction accuracies, but could be assumed since structure words and simple nouns were not included, were able to generalize the models well without overfitting. If given new test data the model *should* (not tested, but in theory should) be able to produce similar results since the structure words were removed.

Appendix

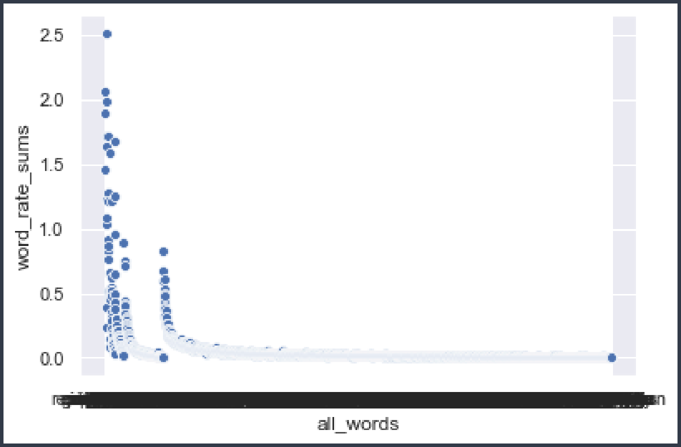
Note that for all code, rownormalizeddocword.csv has the column names included.

plot_stats.ipynb

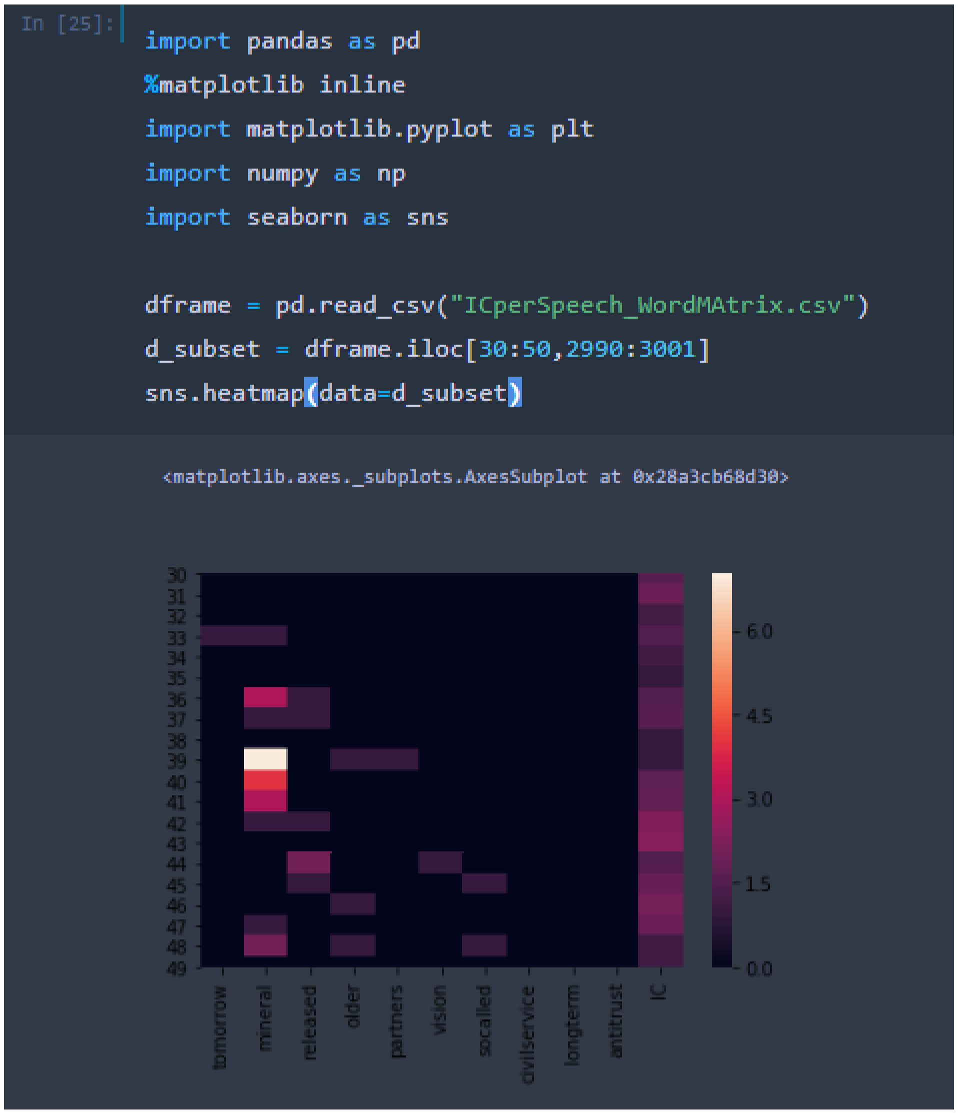
```
import pandas as pd
# %matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
sns.set(style="darkgrid")

df = pd.read_csv("rownormaliseddocword.csv")
df = df.drop(['IC'], axis=1)
# plot word frequencies on the y, words on the x
sums = df.sum(axis=0)
all_sums=[]
for col in df:
    all_sums.append(sums[col])

df_plot = pd.DataFrame()
df_plot['word_rate_sums'] = all_sums # len 3001
df_plot['all_words'] = df.columns # len 3001
```

```
sns.scatterplot(x="all_words", y="word_rate_sums", data=df_plot)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c73b410278>
```



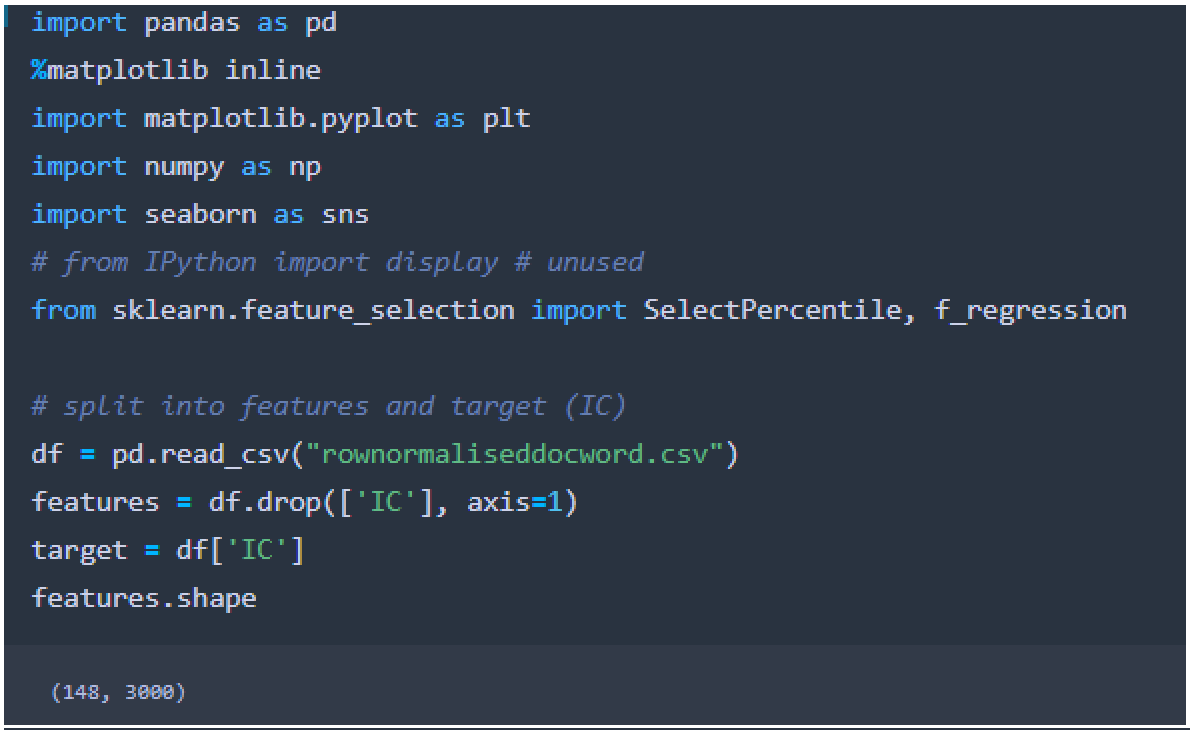
heatmap.ipynb



```
In [25]: import pandas as pd
         %matplotlib inline
         import matplotlib.pyplot as plt
         import numpy as np
         import seaborn as sns

         dframe = pd.read_csv("ICperSpeech_WordMAtrix.csv")
         d_subset = dframe.iloc[30:50,2990:3001]
         sns.heatmap(data=d_subset)
```

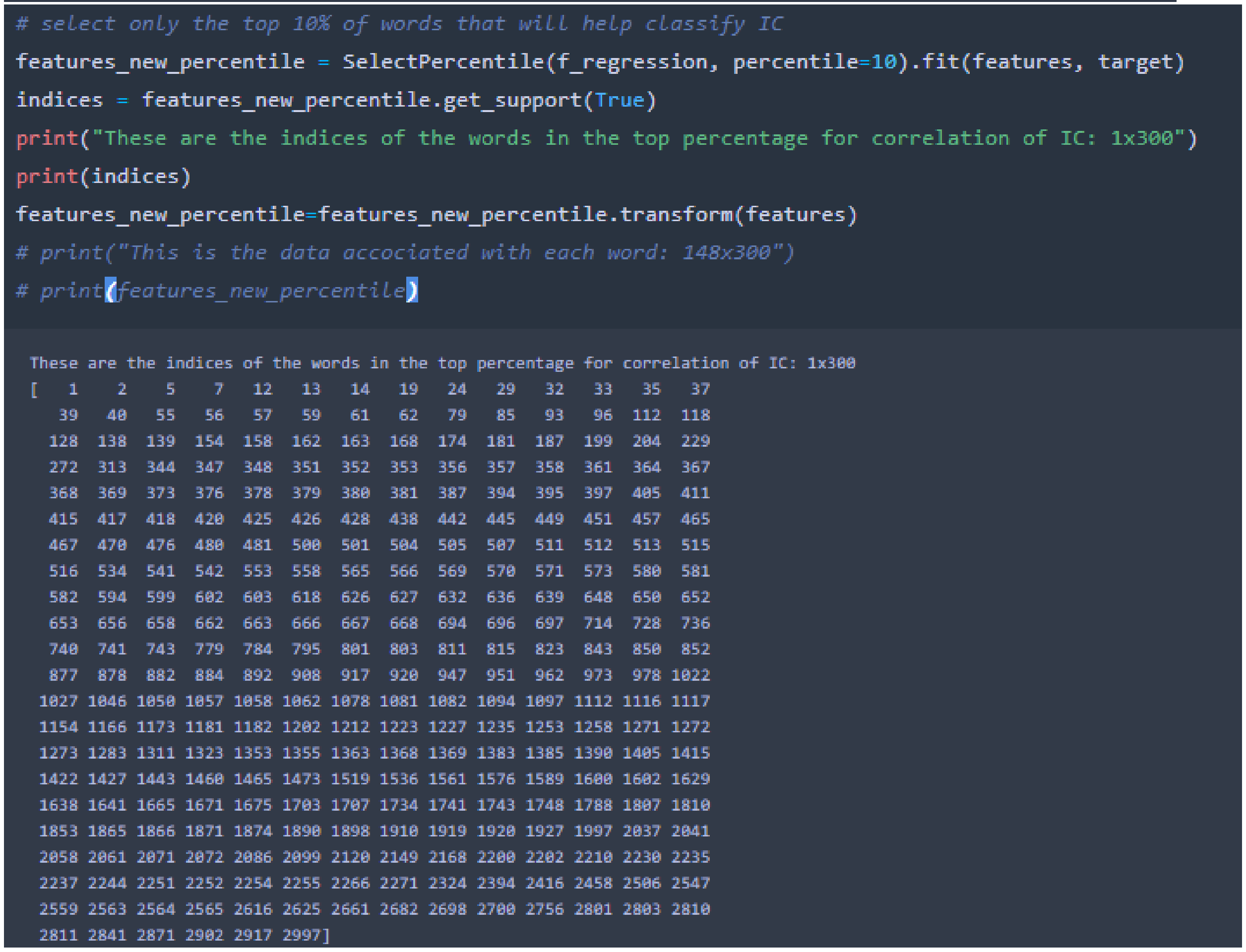
```
<matplotlib.axes._subplots.AxesSubplot at 0x28a3cb68d30>
```

select_best_attributes.ipynb



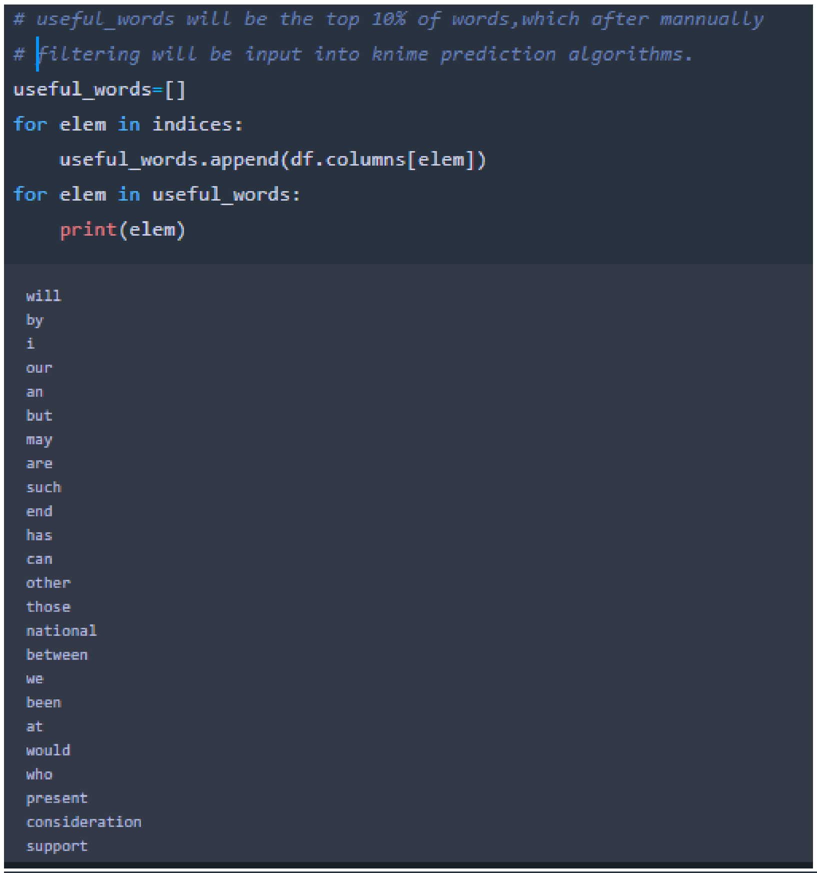
```
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
# from IPython import display # unused
from sklearn.feature_selection import SelectPercentile, f_regression

# split into features and target (IC)
df = pd.read_csv("rownormaliseddocword.csv")
features = df.drop(['IC'], axis=1)
target = df['IC']
features.shape
```

```
(148, 3000)
```



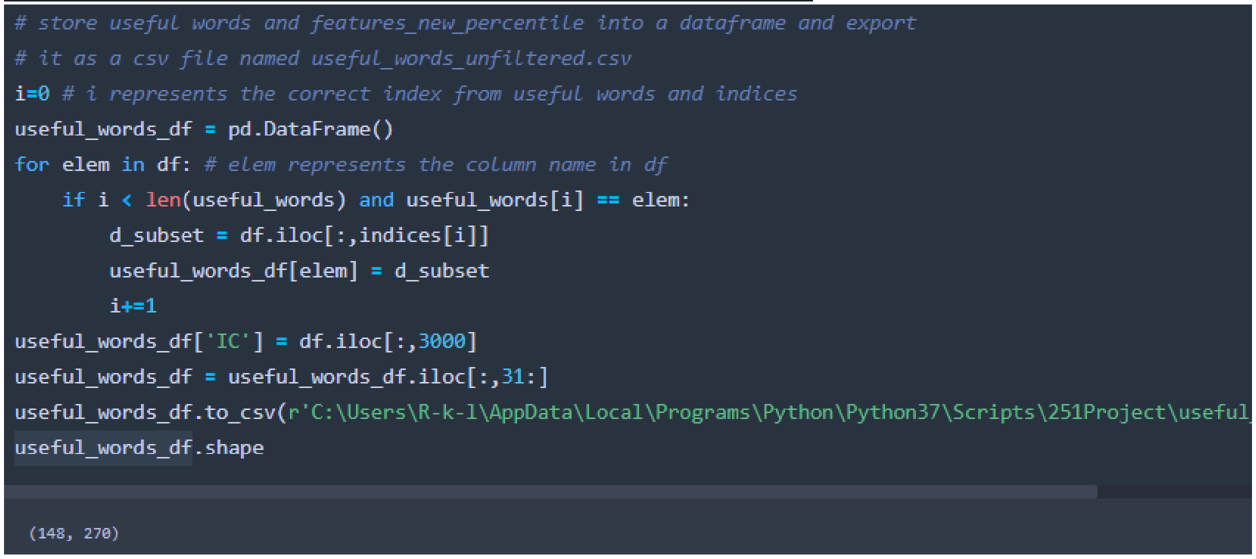
```
# select only the top 10% of words that will help classify IC
features_new_percentile = SelectPercentile(f_regression, percentile=10).fit(features, target)
indices = features_new_percentile.get_support(True)
print("These are the indices of the words in the top percentage for correlation of IC: 1x300")
print(indices)
features_new_percentile=features_new_percentile.transform(features)
# print("This is the data accociated with each word: 148x300")
# print(features_new_percentile)
```

```
These are the indices of the words in the top percentage for correlation of IC: 1x300
[   1    2    5    7   12   13   14   19   24   29   32   33   35   37
   39   40   55   56   57   59   61   62   79   85   93   96  112  118
  128  138  139  154  158  162  163  168  174  181  187  199  204  229
  272  313  344  347  348  351  352  353  356  357  358  361  364  367
  368  369  373  376  378  379  380  381  387  394  395  397  405  411
  415  417  418  420  425  426  428  438  442  445  449  451  457  465
  467  470  476  480  481  500  501  504  505  507  511  512  513  515
  516  534  541  542  553  558  565  566  569  570  571  573  580  581
  582  594  599  602  603  618  626  627  632  636  639  648  650  652
  653  656  658  662  663  666  667  668  694  696  697  714  728  736
  740  741  743  779  784  795  801  803  811  815  823  843  850  852
  877  878  882  884  892  908  917  920  947  951  962  973  978 1022
 1027 1046 1050 1057 1058 1062 1078 1081 1082 1094 1097 1112 1116 1117
 1154 1166 1173 1181 1182 1202 1212 1223 1227 1235 1253 1258 1271 1272
 1273 1283 1311 1323 1353 1355 1363 1368 1369 1383 1385 1390 1405 1415
 1422 1427 1443 1460 1465 1473 1519 1536 1561 1576 1589 1600 1602 1629
 1638 1641 1665 1671 1675 1703 1707 1734 1741 1743 1748 1788 1807 1810
 1853 1865 1866 1871 1874 1890 1898 1910 1919 1920 1927 1997 2037 2041
 2058 2061 2071 2072 2086 2099 2120 2149 2168 2200 2202 2210 2230 2235
 2237 2244 2251 2252 2254 2255 2266 2271 2324 2394 2416 2458 2506 2547
 2559 2563 2564 2565 2616 2625 2661 2682 2698 2700 2756 2801 2803 2810
 2811 2841 2871 2902 2917 2997]
```



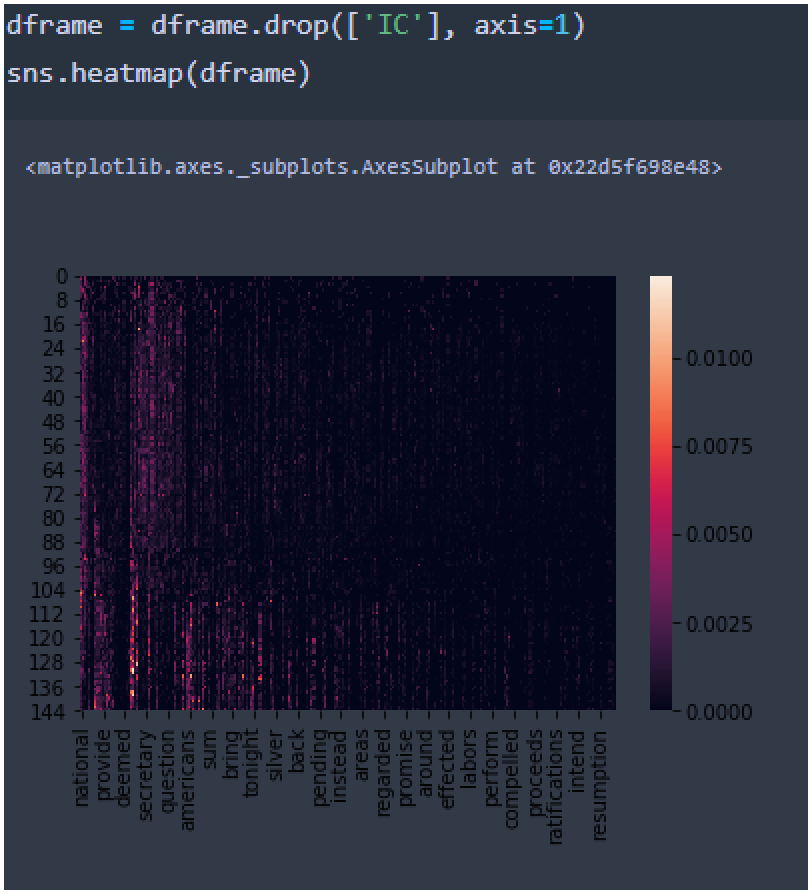
```
# useful_words will be the top 10% of words,which after mannually
# filtering will be input into knime prediction algorithms.
useful_words=[]
for elem in indices:
    useful_words.append(df.columns[elem])
for elem in useful_words:
    print(elem)
```

```
will
by
i
our
an
but
may
are
such
end
has
can
other
those
national
between
we
been
at
would
who
present
consideration
support
```



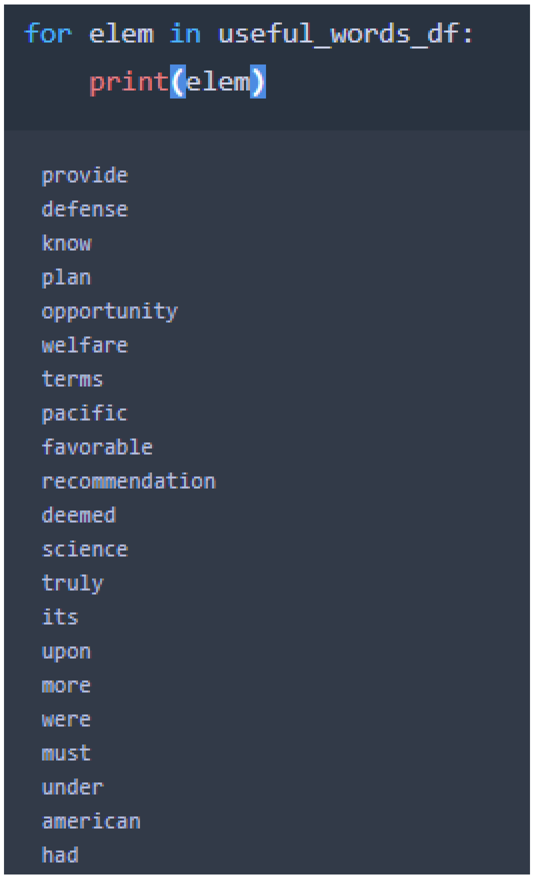
```
# store useful words and features_new_percentile into a dataframe and export
# it as a csv file named useful_words_unfiltered.csv
i=0 # i represents the correct index from useful words and indices
useful_words_df = pd.DataFrame()
for elem in df: # elem represents the column name in df
    if i < len(useful_words) and useful_words[i] == elem:
        d_subset = df.iloc[:,indices[i]]
        useful_words_df[elem] = d_subset
        i+=1
useful_words_df['IC'] = df.iloc[:,3000]
useful_words_df = useful_words_df.iloc[:,31:]
useful_words_df.to_csv(r'C:\Users\R-k-l\AppData\Local\Programs\Python\Python37\Scripts\251Project\useful_
useful_words_df.shape
```

```
(148, 270)
```



```
dframe = dframe.drop(['IC'], axis=1)
sns.heatmap(dframe)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22d5f698e48>
```



```
for elem in useful_words_df:
    print(elem)
```

```
provide
defense
know
plan
opportunity
welfare
terms
pacific
favorable
recommendation
deemed
science
truly
its
upon
more
were
must
under
american
had
```

References

[1] sklearn, "1.1.3 Feature Selection," [Online]. Available: https://scikit-learn.org/stable/modules/feature_selection.html?fbclid=IwAR0vi3H5KS6MrVclF3EIWFULhlgcgAfcISt5v1p37YL_q2XFvYqKiaQkObM.

[2] J. Heaton, "Heaton Research," Ph.D. is a computer scientist, data scientist, and indie publisher., [Online]. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html#:~:targetText=Underfitting%20occurs%20when%20there%20are,layers%20may%20result%20in%20overfitting..