Alexa Bosworth
Project 2
COMP135

Bag of Words
Using nltk and sklearn dependencies, I implemented bag of words by first loading the training file. I continued with preprocessing, like lowering all uppercased letters, and removing punctuation and digits. I transformed the data into word frequency vectors using CountVectorizer and continued with feature extraction via the TfidfTransformer. I then fit and transformed the data and saved the vectorizer object for testing.

Word Embeddings

Using Gensim, nltk and sklearn dependencies, I implemented word embeddings by loading the glove document. I used a python script by Gensim that converts a glove document into a KeyedVector object. I preprocessed the test file as before, by lowering all uppercased letters, removing punctuation and digits, as well as removing stop words. I created a new feature matrix and label vector. For every word in every document, I summed the word embedding values, then stored it into the new feature matrix as a new instance in the matrix.

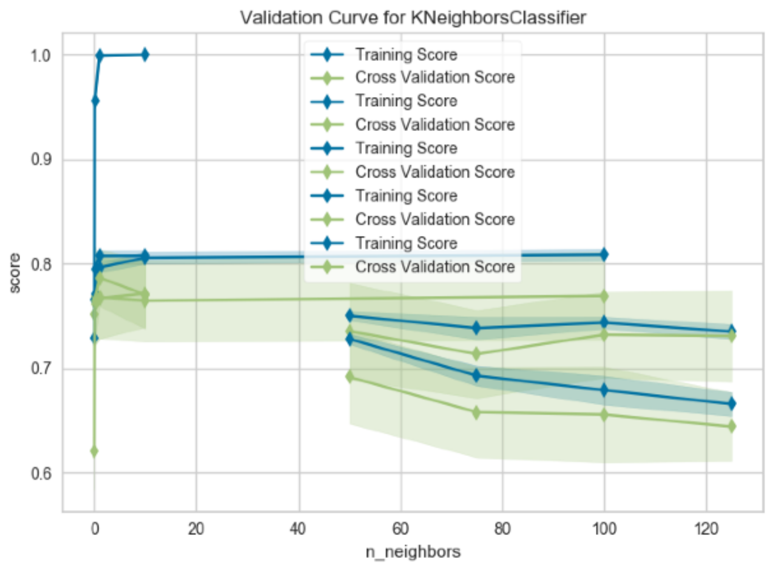
Learning Algorithms
KNN
n_neighbors: the number of neighbors which vote for the label of the test example. Generally, the higher the number indicates greater accurate and more computationally expense.
Candidate params: [50,75,100, 125]. My intuition that a slightly higher number of neighbors would be beneficial proved correct, as our sample size is rather large and there exists some ambiguity in the data (particularly in Bag of Words).

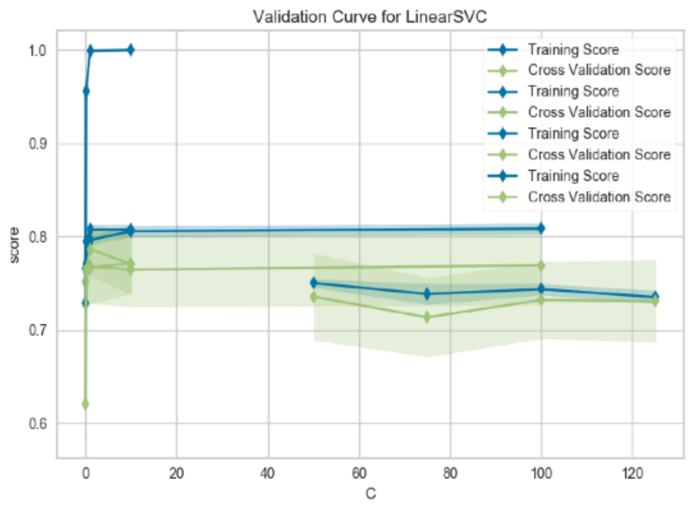
Logistic Regression
C: the inverse of the lambda regularizer value (1/lambda), s.t. smaller values promote stronger regularization effect (reducing overfitting).
Candidate params: [100, 10, 1, .1], non-zero values are not allowed and since the conventional choices for lambda are these values, it is intuitive to choose similar values.

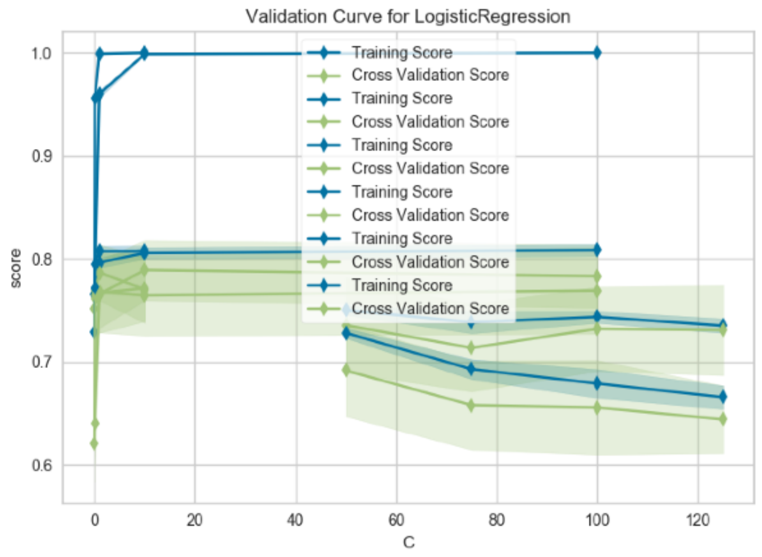
SVM
C: a penalty parameter of the error term which sets the amount of regularization. Choosing a value for this parameter suggests the need to take sample size into account, since the loss function depends on sample size.
Candidate params: [.01, .1, 1, 10], I've seen these values chosen in the sklearn manpages.

Model Selection:
Using sklearn dependency, I implemented a for loop which iterated on every learning model for word embeddings, then subsequently, bag of words. GridSearchCV implements a cross-validated grid-search over the list of parameters provided by the user. I gathered statistics on the classifiers and took note of how the model did for each hyperparameter term. I chose the model which appeared to be able to generalize over the data best with various values of hyperparameters, ie there was little change to how the model performed when supplied with a different hyper parameter value.

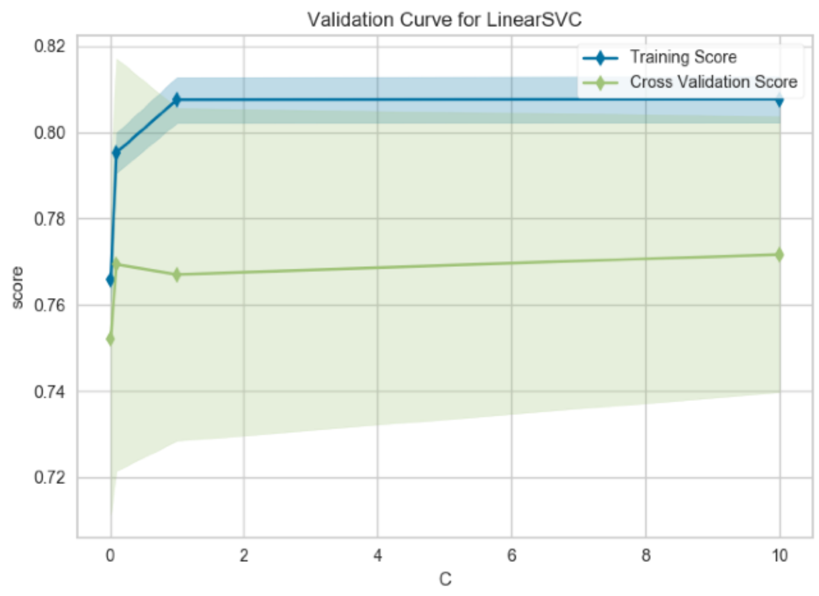
Bag of Words

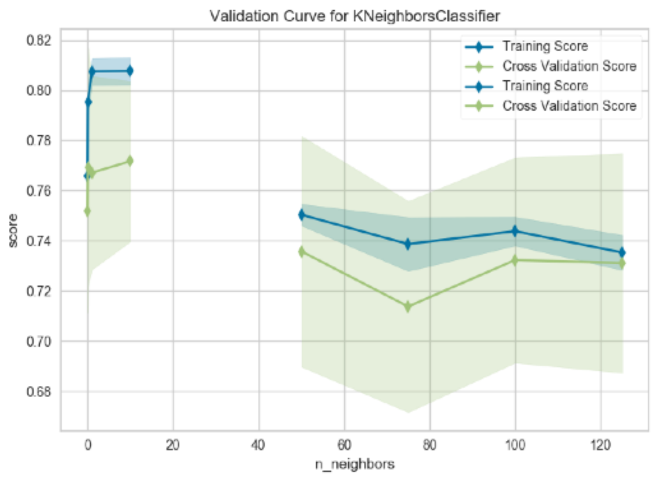

Validation Curve for KNeighborsClassifier
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
score
n_neighbors
1.0
0.9
0.8
0.7
0.6
0
20
40
60
80
100
120

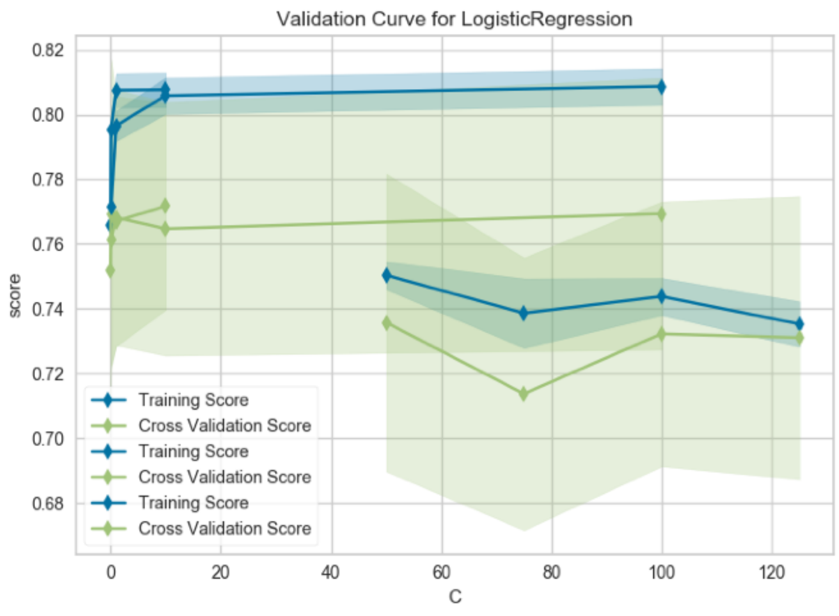

Validation Curve for LinearSVC
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
score
C
1.0
0.9
0.8
0.7
0.6
0
20
40
60
80
100
120


Validation Curve for LogisticRegression
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
Training Score
Cross Validation Score
score
C
1.0
0.9
0.8
0.7
0.6
0
20
40
60
80
100
120

Word Embeddings







Final Classifier

Bag of Words Logistic Regression with C = 10 wins:

```
(LogisticRegression(C=10, 
class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=40000, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='lbfgs',
          tol=0.0001, verbose=0, warm_start=False), {'C': 10}, 0.7891077636152954)
```

My confidence interval for the accuracy of this classifier is: [0.67532, 0.90268]