معالجة اللغات الطبيعية/2018/مشاريع/7

من FCIH Wiki

محتويات

Project data

Project ID: 7

Project topic: Text Classification

Project Github repo: https://github.com/ansamhazem/NLP-project

Team members

First member: 20140107 [انسام حازم محمد كمال نوار]

On Github: ansamhazem

Second member: 20140221 [20140221]

On Github: tareq95

Third member: 20140010 [احمد ابو العلا حسين صديق اسماعيل]

On Github: aboalella

Abstract

Our project based on text classification, Text classification is a core problem to many applications, like spam detection, sentiment analysis or smart replies. The goal of text classification is to assign documents (such as emails, posts, text messages, product reviews, etc...) to one or multiple categories. Such categories can be review scores, spam v.s. non-spam, or the language in which the document was typed. In our project we classify articles as we classify any input come to one category of (Business, science, health, entertainment) . our classification based on headline of article as we tokenize it from article and take the most effective term which can differentiate between article and another.

Data set

News Aggregator dataset contains headlines, URLs, and categories for 422,937 news stories collected by a web aggregator between March 10th, 2014 and August 10th, 2014.

News categories included in this dataset include business; science and technology; entertainment; and health. Different news articles that refer to the same news item (e.g., several articles about recently released employment statistics) are also categorized together.

Content The columns included in this dataset are:

ID : the numeric ID of the article TITLE : the headline of the article, URL : the URL of the article, PUBLISHER : the publisher of the article, CATEGORY : the category of the news item; one of: -- b : business -- t : science and technology -- e : entertainment -- m : health, STORY : alphanumeric ID of the news story that the article discusses, HOSTNAME : hostname where the article was posted, TIMESTAMP : approximate timestamp of the article's publication, given in Unix time (seconds since midnight on Jan 1, 1970),

Data Set link [١] (http://archive.ics.uci.edu/ml/datasets/News+Aggregator)

Methodology

Document representation Techniques Used:

a. **bag-of-words (BOW):** is a simplifying representation used in natural language processing and information retrieval (IR). Also known as the vector space model. In this model, a text (such as a sentence or a document) is represented as the bag multiset of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model has also been used for computer vision. The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier. An early reference to "bag of words" in a linguistic context can be found in Zellig Harris's 1954 article on Distributional Structure.

b. **Term frequency–inverse document frequency (TFIDF):** Is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Tf-idf is one of the most popular term-weighting schemes today; 83% of text-based recommended systems in digital libraries use tf-idf.

c. **Word embedding:** A word embedding is a class of approaches for representing words and documents using a dense vector representation. It is an improvement over more the traditional bag-of-word model encoding schemes where large sparse vectors were used to represent each word or to score each word within a vector to represent an entire vocabulary. These representations were sparse because the vocabularies were vast and a given word or document would be represented by a large vector comprised mostly of zero values. Instead, in an embedding, words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space. The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. The position of a word in the learned vector space is referred to as its embedding.

Classification Algorithms used :

Naive Bayes algorithm (NB): It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Support Vector Machine (SVM): "Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well .

Recurrent neural network (RNN): The primary role of the neural models is to represent the variable-length text as a fixed-length vector. These models generally consist of a projection layer that maps words, subword units or n-grams to vector representations (often trained beforehand with unsupervised methods), and then combine them with the different architectures of neural networks. There are several kinds of models to model text, such as, recurrent neural network (RNN) and convolutional neural network (CNN). Traditionally, a simple strategy for modeling sequence is to map the input sequence to a fixed-sized vector using one RNN, and then to feed the vector to a softmax layer for classification or other tasks.

Convolution neural network : Convolutional neural networks (CNNs) have dramatically improved the approaches to many active research problems. One of the key differentiators between CNNs and traditional machine learning approaches is the ability for CNNs to learn complex feature representations. We apply a CNN-based approach to categorization articles to (Business, science, health, entertainment). we building 3 layers all of them are dense layers, the first one take input with activation function "relu" with 512 neurons, the second one take 256 neurons with "sigmoid" activation function and finally the latest one is the output layer which take number of score to sum up the score of each class and take the highest one and used softmax activation function in last layer to make probability distribution and can easily know the win class

K-nearest neighbour : this approach doesn't depend on training like previous algorithms just load all data in memory and start compare the testing data with all data in memory all of this depends and the number of k we have assigned

Logistic Regression (LG) :more general Framework , Logistic is a special mathematical function it uses and regression is Combines a weight vector with observations to create an answer More general cookbook for building conditional probability distributions . Naïve Bayes (later today) is a special case of logistic regression.

Experiment #1(CNN):

First split my data to train 80% and test 20%, this test determine even the training in right path or not it's called validation data. then this data have been tokenized and then converted to indices vector to can fit to train then we build 3 layers input hidden and output layer , each one have number of neurons , the first layer has 512 number of neurons , second layer contain 256 neuron and the last one have neuron to easily give score to each class , all these neuron are connected fully connected we use activation function relu in the first two layers and softmax in the last one because softmax is a probability distribution make the sum of all scores 1 so we easily determine which one right

Experiment #2(CNN):

all like first experiment but we will change some hyper parameter, we will use sgd optimizer instead of Adam

Experiment #3(CNN):

all like first experiment but we will change some hyper parameter, we will use sgd optimizer instead of Adam and covert activation function to relu like the first two functions

Experiment #4 (CNN):

use the same hyper parameter of first experiment and decrease the number of neurons in each one, the first one become 256, second one 64 and the last 4

Experiment #5(KNN):

in this algorithm we have 26000 record of data 80%train loaded to memory and 20% test , test here is to compare each record with all data has been loaded to memory , our hyper parameter is number of k = 11 , weight ='uniform' , weight='distance'

Experiment #6(KNN):

in this algorithm we have 26000 record of data 80%train loaded to memory and 20% test , test we will change number of k from 11 to 5 and parameter of weight still unchanged

Experiment #7(KNN):

in this algorithm we have 26000 record of data 80%train loaded to memory and 20% test , test we will remove parameter weight ='distance ' with the stable of remaining parameters k=11

Experiment #8(KNN):

in this algorithm we have 26000 record of data 80%train loaded to memory and 20% test , test we will remove parameter weight =uniform ' with the stable of remaining parameters k=11

Experiment #9:

In this experiment we use a TFIDF representation of each document. And also a linear Support Vector Machine (SVM) classifier. We split the data, so that 20% of them remain for testing. And also we used Max feature as 1700 with removing stop words . We used linear SVM with c=1 which is , Penalty parameter C of the error term. It also controls the tradeoff between smooth decision boundary and classifying the training points correctly, and gamma=1 which is, Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. Higher the value of gamma, will try to exact fit the as per training data set i.e. generalization error and cause overfitting problem.

Experiment #10:

As the experiment above but we changed the Max feature parameter to 10 with removing stop words.

Experiment #11:

As the experiment above but we use a Bag Of Words (BOW) representation of each document. We split the data, so that 20% of them remain for testing. And also we used Max feature as 1700 with removing stop words .

Experiment #12:

In this experiment we use a Term Frequency (TF) representation of each document. And also a linear Support Vector Machine (SVM) classifier. We split the data, so that 20% of them remain for testing. As we changed the parameter use_idf in TfidfVectorizer to false.

Experiment #13:

The same as the experiment #1 we changed the gamma parameter to 10 to fine tune the parameters to get the best results.

Experiment #14:

In this experiment we use word embedding as word and document representation technique with Recurrent neural network as Classification Model . First we splitting the data train and test (20% test) and preprocessing the data according to MAX_DOCUMENT_LENGTH we specified which is 10 and convert string label to and integer then Convert indexes of words into embeddings. This creates embeddings matrix of [n_words, EMBEDDING_SIZE =50] and then maps word indexes of the sequence into [batch_size, sequence_length, EMBEDDING_SIZE], after that Split into list of embedding per word, while removing doc length dim. Then Create an unrolled Recurrent Neural Networks to length of 10 as MAX_DOCUMENT_LENGTH and passes word_list as inputs for each unit. Given encoding of RNN, take encoding of last step and pass it as features for softmax classification over output classes.

Experiment #15:

In this experiment we also use Recurrent neural network as Classification Model but with Bag of words (BOW) document representation . First we splitting the data train and test (20% test) and preprocessing the data according to MAX_DOCUMENT_LENGTH we specified which is 10 and convert string label to and integer then get feature column that represents sequences of integers based on n_words then Pass this to embedding_column to convert sequence categorical data into dense representation for input to sequence RNN ,then Builds input layer for given feature_columns , build last dense layer as classification layer then pass it as features for softmax classification over output classes .

Experiment #16:

In this experiment we use a TFIDF representation of each document. And also a Naïve Bayes (NB) classifier. We split the data, so that 20% of them remain for testing. And also we used Max feature as 170 with removing stop words .

Experiment #17:

As the experiment above but we changed the Max feature parameter to default with removing stop words.

Experiment #18:

As the experiment 16 but we changed the Max feature parameter to 11500 with removing stop words.

Experiment #19:

As the experiment 16 but we changed the Max feature parameter to 9500 with removing stop words.

Experiment #20:

In this experiment we use a BOW representation of each document. And also a Naïve Bayes (NB) classifier. We split the data, so that 20% of them remain for testing. And also we used Max feature as 1700 with removing stop words.

Experiment #21:

As the experiment above but we changed the Max feature parameter to 9500 with removing stop words.

Experiment #22:

As the experiment 20 but we changed the Max feature parameter to 10500 with removing stop words.

Experiment #23:

As the experiment 20 but we changed the Max feature parameter to 11500 with removing stop words.

Experiment #24:

In this experiment we use a TFIDF representation of each document. And also a Naïve Bayes (LG) classifier. We split the data, so that 20% of them remain for testing. And also we used Max feature as 1700 with removing stop words.

Experiment #25:

As the experiment above but we changed the Max feature parameter to default with removing stop words.

Experiment #26:

As the experiment 16 but we changed the Max feature parameter to 11500 with removing stop words.

Experiment #27:

As the experiment 16 but we changed the Max feature parameter to 9500 with removing stop words.

Experiment #28:

In this experiment we use a BOW representation of each document. And also a Naïve Bayes (LG) classifier. We split the data, so that 20% of them remain for testing. And also we used Max feature as 1700 with removing stop words.

Experiment #29:

As the experiment above but we changed the Max feature parameter to 9500 with removing stop words.

Experiment #30:

As the experiment 20 but we changed the Max feature parameter to 10500 with removing stop words.

Experiment #31:

As the experiment 20 but we changed the Max feature parameter to 11500 with removing stop words

Results

Experiment#1:

result is :

```
Loss       Accuracy    Validation Loss   Validation Accuracy
  0.0149     0.9941          0.1527                     0.9955
```

Experiment#2:

result is :

```
   Loss       Accuracy    Validation Loss   Validation Accuracy
  1.286      0.348           1.274                    0.35
```

Experiment#3:

result is :

```
  Loss       Accuracy    Validation Loss   Validation Accuracy
  1.3863     0.3470          1.3863                     0.3480
```

Experiment#4:

result is :

```
  Loss       Accuracy    Validation Loss   Validation Accuracy
  0.0091    0.9959          0.1648                     0.9520
```

Experiment# 5 :

result is :

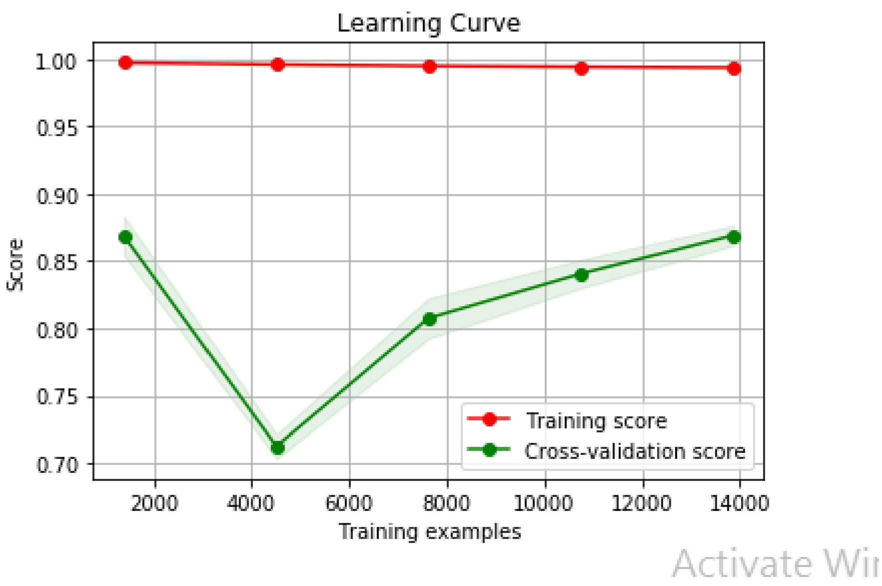
```
Classification Report:
              precision    recall  f1-score   support

          b       0.89      0.88      0.88      1583
          e       0.90      0.95      0.93      1789
          m       0.95      0.83      0.89       551
          t       0.88      0.86      0.87      1277

avg / total       0.90      0.89      0.89      5200
```

```
accuracy: 0.94961358
```

Graph of learning curve:



Experiment# 6 :

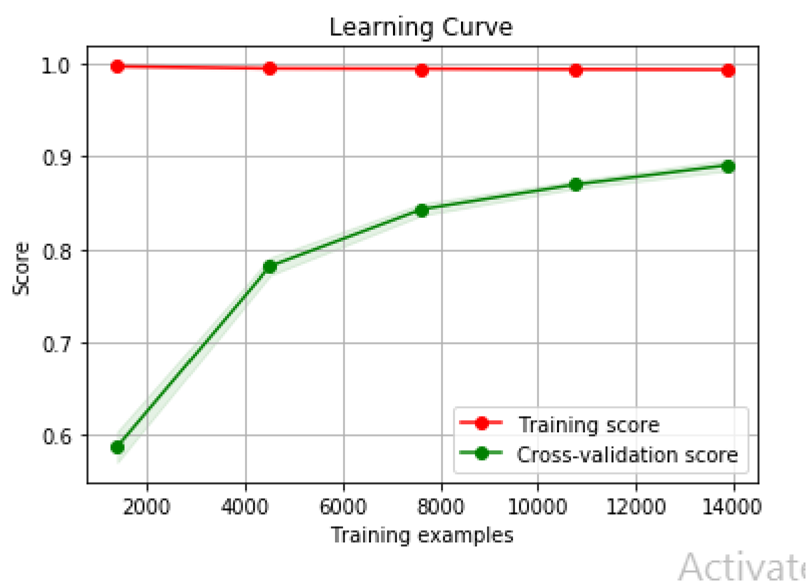
result is :

```
Classification Report:
              precision    recall  f1-score   support

          b       0.90      0.91      0.90      1583
          e       0.93      0.96      0.94      1789
          m       0.91      0.88      0.89       551
          t       0.91      0.87      0.89      1277

avg / total       0.91      0.91      0.91      5200
accuracy :0.941730769
```

Graph of learning curve:



Experiment# 7 :

result is :

```
Classification Report:
              precision    recall  f1-score   support

          b       0.86      0.87      0.87      1583
          e       0.88      0.95      0.91      1789
          m       0.95      0.80      0.87       551
          t       0.88      0.84      0.86      1277

avg / total       0.88      0.88      0.88      5200

0.8813461538461539 :  accuracy
```

Graph of learning curve:

Experiment# 8 :

result is :

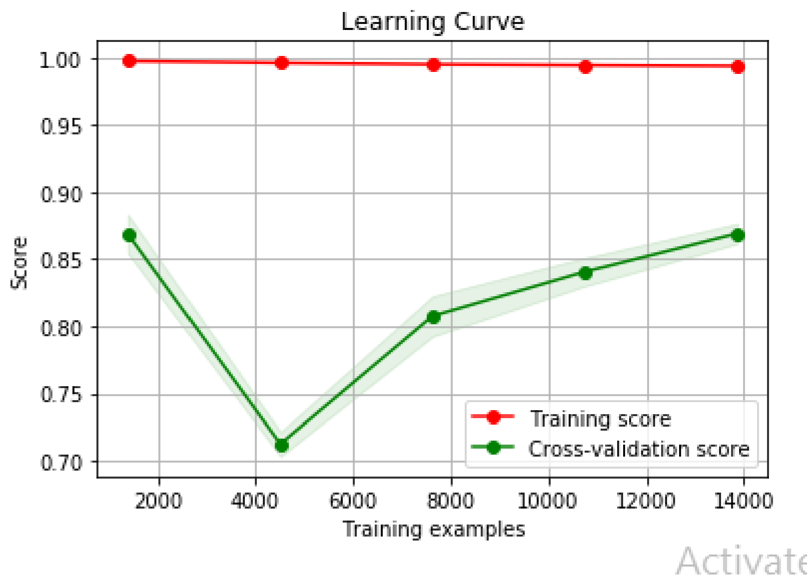
```
Classification Report:
              precision    recall  f1-score   support

          b       0.89      0.88      0.88      1583
          e       0.90      0.95      0.93      1789
          m       0.95      0.83      0.89       551
          t       0.88      0.86      0.87      1277

avg / total       0.90      0.89      0.89      5200

 0.8948076923 :  accuracy
```

Graph of learning curve:

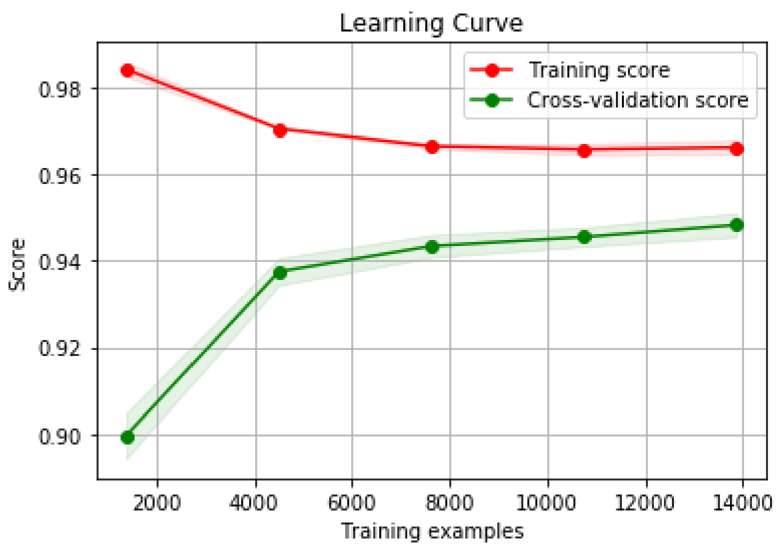


Experiment #9:

Results:

```
        Classification Report:
              precision    recall  f1-score   support
          b       0.90      0.97      0.93      1583
          e       0.99      0.97      0.98      1789
          m       0.98      0.90      0.94       551
          t       0.95      0.92      0.93      1277
avg / total       0.95      0.95      0.95      5200
Accuracy:  0.950576923077
```

The below figure is the learning graph of training and validation data .



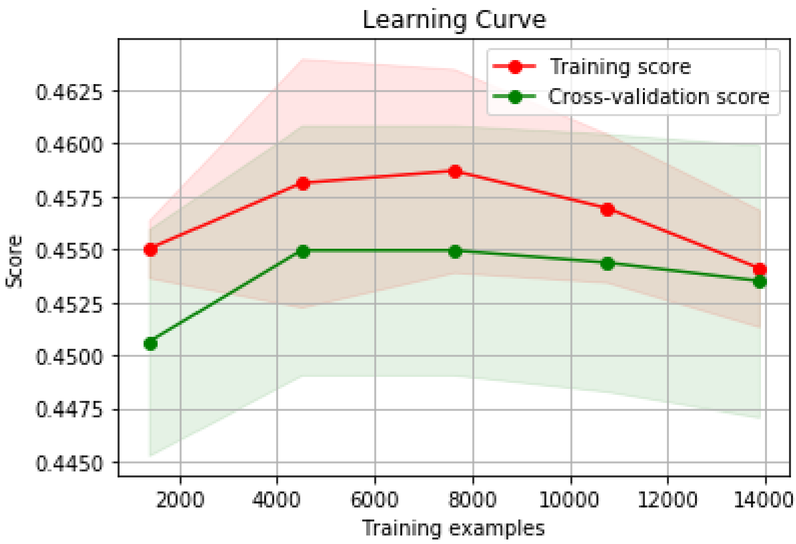
Experiment #10:

Results:

```
Classification Report:
              precision    recall  f1-score   support
          b       0.63      0.19      0.29      1583
          e       0.39      0.98      0.56      1789
          m       0.00      0.00      0.00       551
          t       0.90      0.20      0.32      1277
```

```
avg / total       0.55      0.44      0.36      5200
Accuracy: 0.4428865178
```

The below figure is the learning graph of training and validation data .

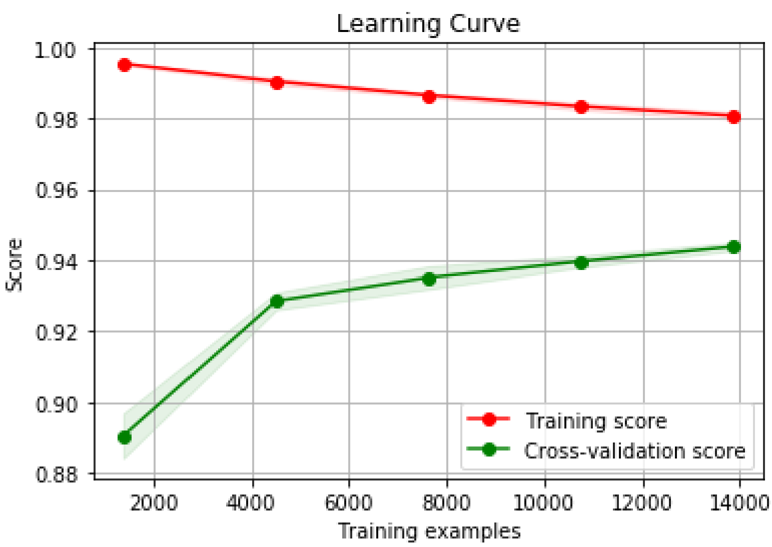


Experiment #11:

Results:

```
Classification Report:
              precision    recall  f1-score   support
          b       0.91      0.96      0.93      1583
          e       0.99      0.97      0.98      1789
          m       0.96      0.91      0.93       551
          t       0.94      0.93      0.93      1277
avg / total       0.95      0.95      0.95      5200
Accuracy: 0.948269230769
```

The below figure is the learning graph of training and validation data .

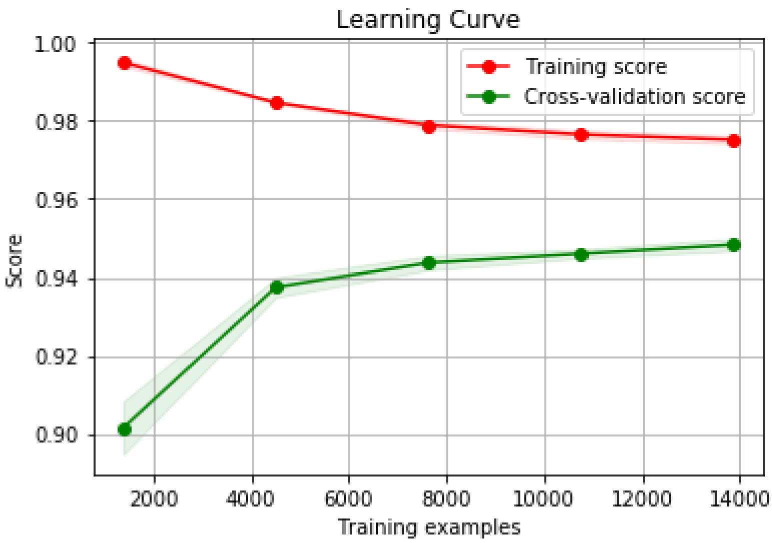


Experiment #12:

Results:

```
Classification Report:
             precision    recall  f1-score   support
          b       0.92      0.96      0.94      1583
          e       0.98      0.97      0.98      1789
          m       0.96      0.91      0.94       551
          t       0.94      0.93      0.94      1277
avg / total       0.95      0.95      0.95      5200
Accuracy: 0.951153846154
```

The below figure is the learning graph of training and validation data .

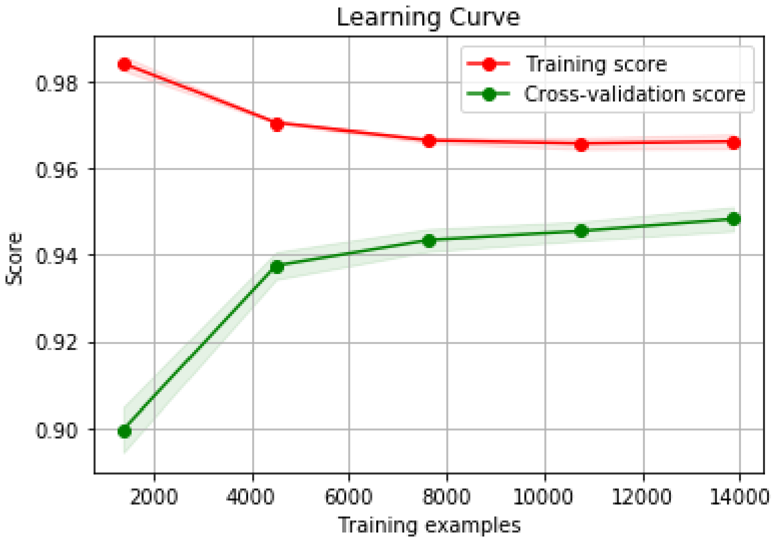


Experiment #13:

Results:

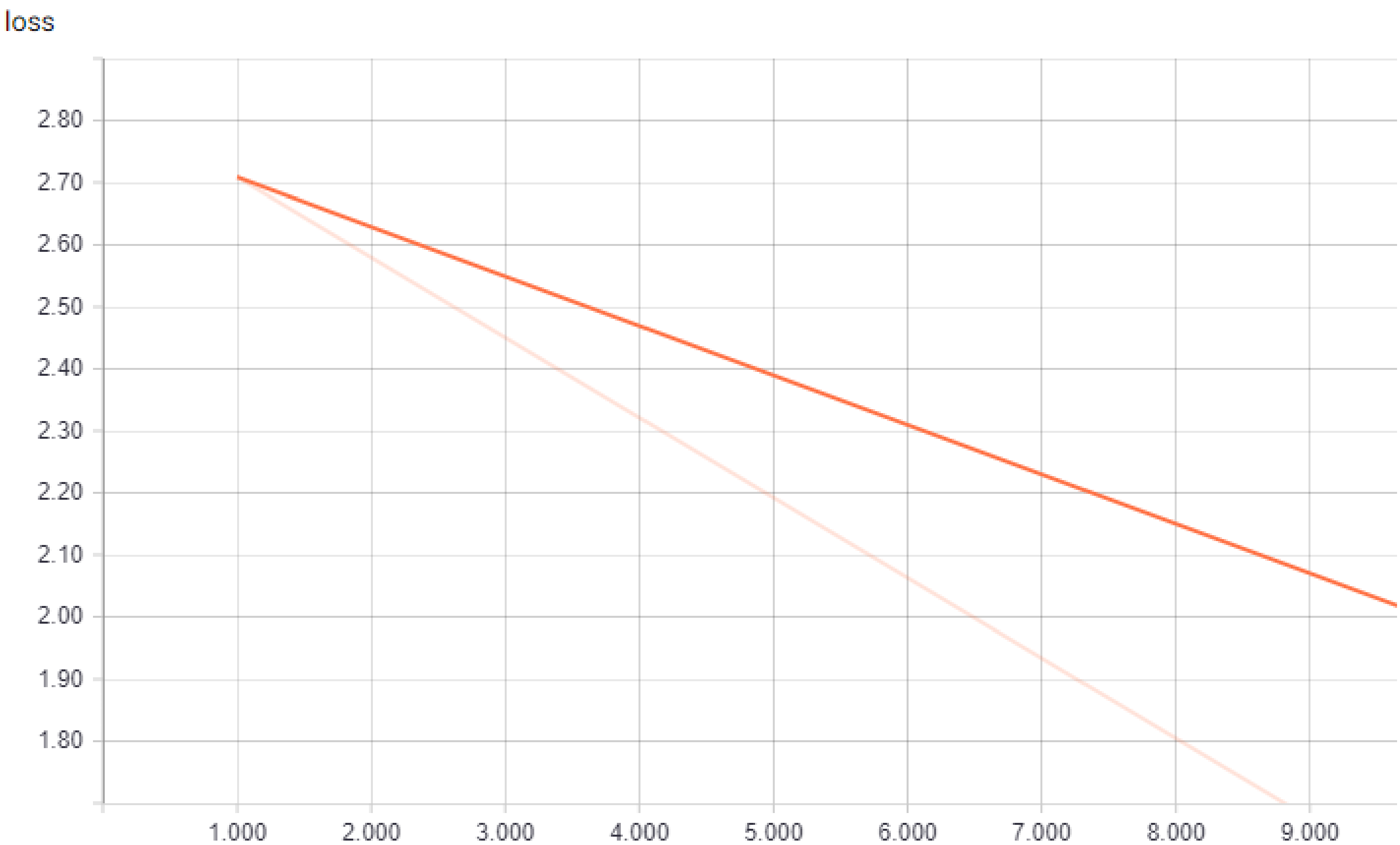
```
Classification Report:
             precision    recall  f1-score   support
          b       0.90      0.97      0.93      1583
          e       0.99      0.97      0.98      1789
          m       0.98      0.90      0.94       551
          t       0.95      0.92      0.93      1277
avg / total       0.95      0.95      0.95      5200
Accuracy: 0.950576923077
```

The below figure is the learning graph of training and validation data .



Experiment #14:

Results: Loss graph:




```
Accuracy (tensorflow): 0.935000
```

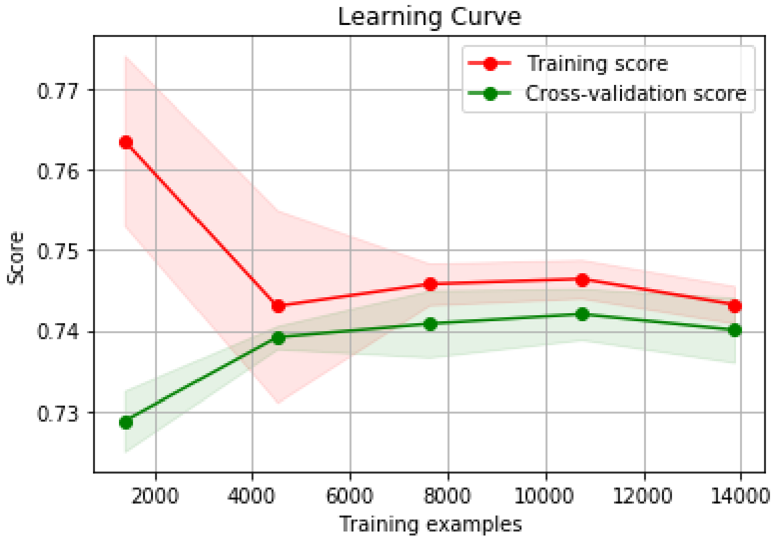
Experiment #15:

Results:

```
Accuracy (tensorflow): 0.943269
```

Experiment #16:

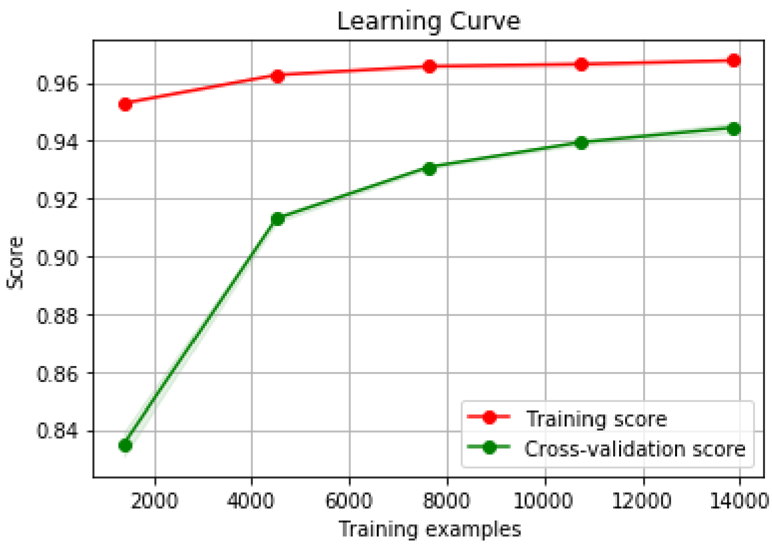
Results: Graph:




```
Accuracy : 0.732692307692
```

Experiment #17:

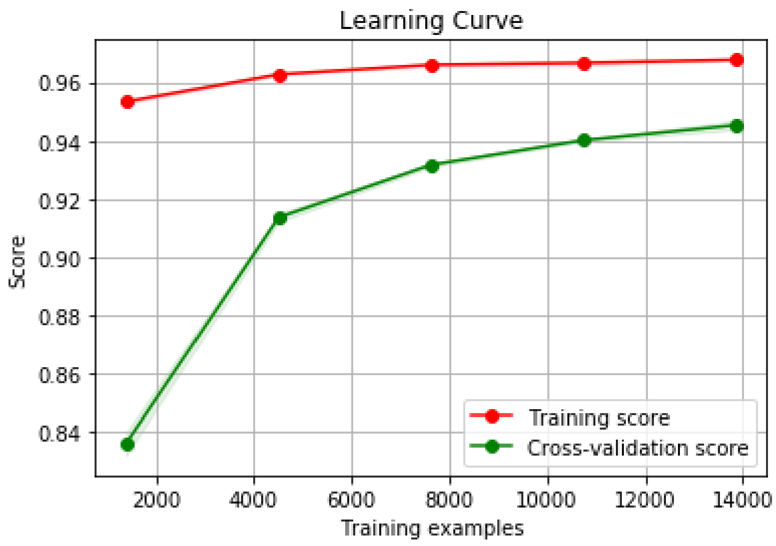
Results: Graph:




```
Accuracy : 0.955961538462
```

Experiment #18:

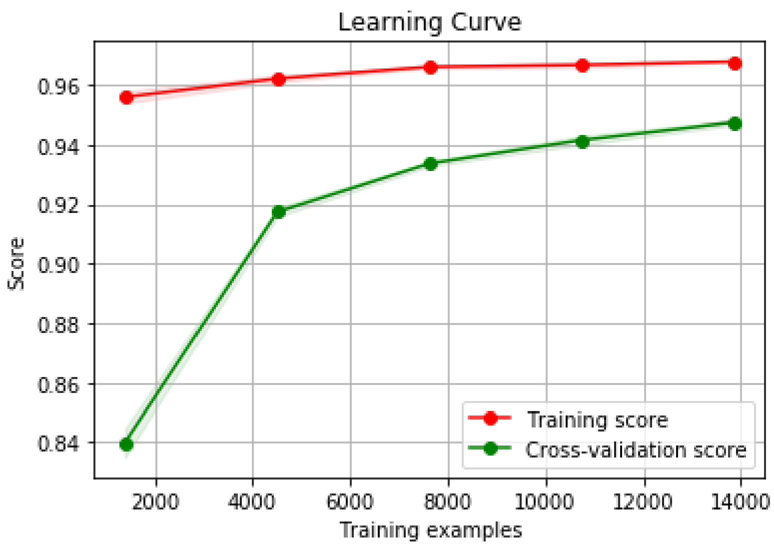
Results: Graph:




```
Accuracy : 0.955
```

Experiment #19:

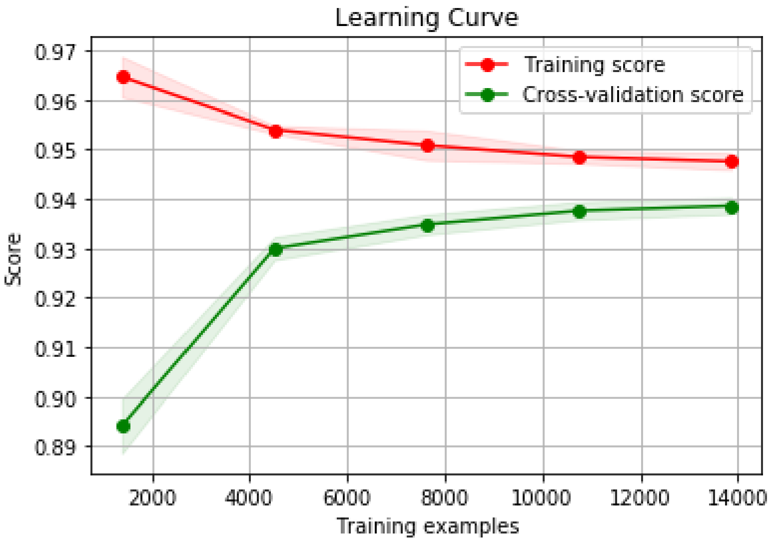
Results: Graph:




```
Accuracy :  0.956153846154
```

Experiment #20:

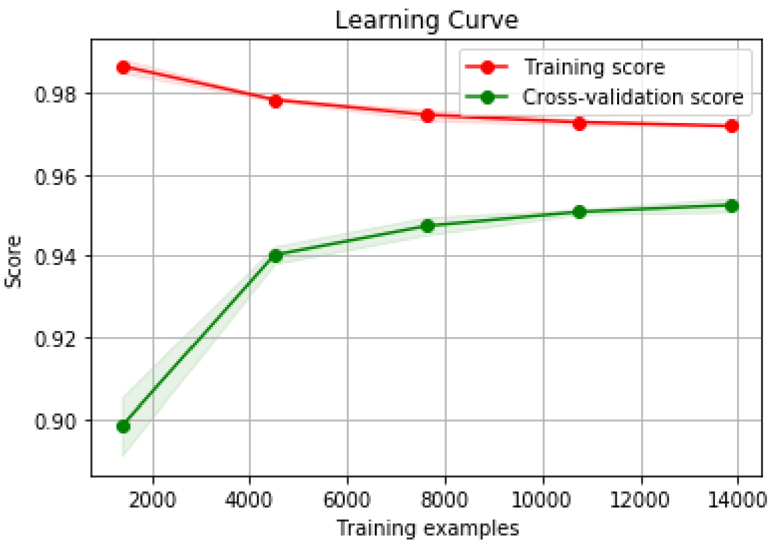
Results: Graph:




```
Accuracy :  0.939423076923
```

Experiment #21:

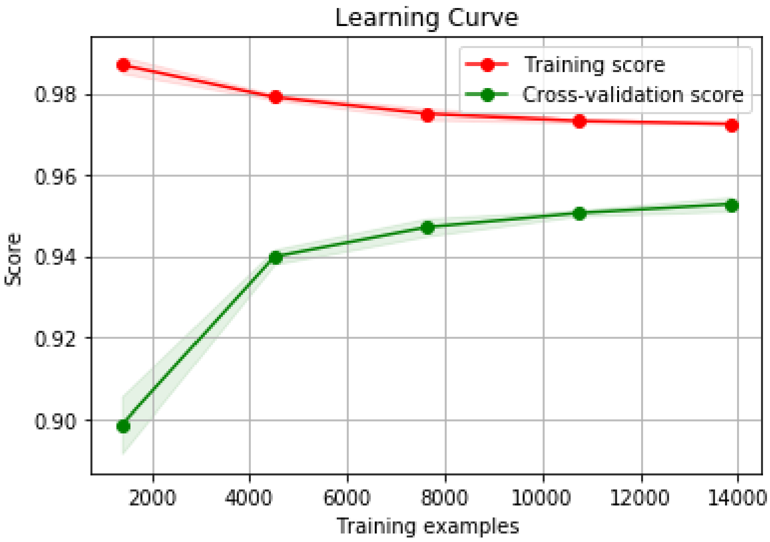
Results: Graph:




```
Accuracy : :0.958846153846
```

Experiment #22:

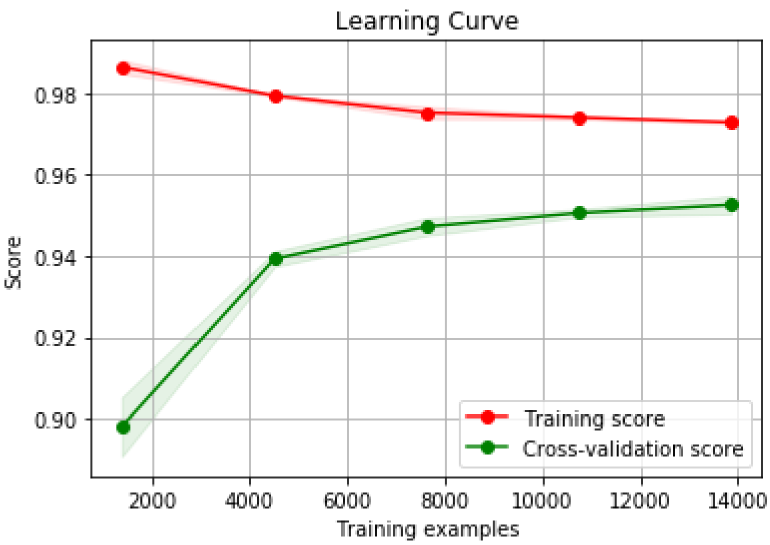
Results: Graph:




```
Accuracy : 0.959038461538
```

Experiment #23:

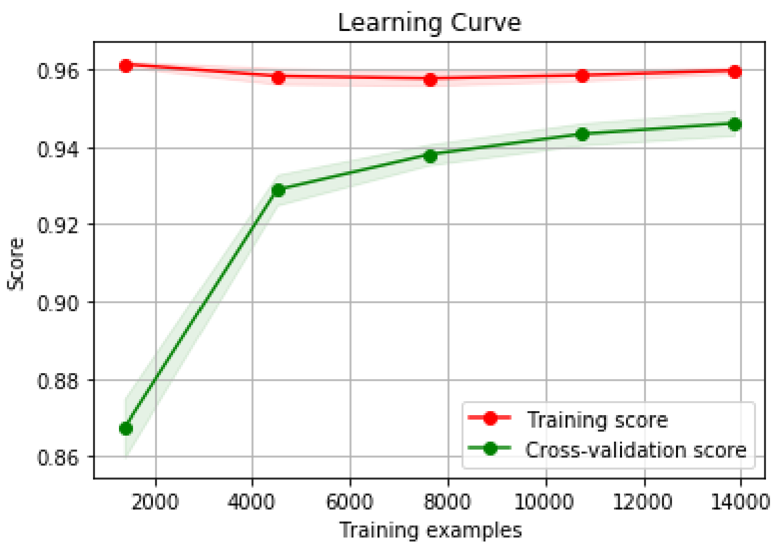
Results: Graph:




```
Accuracy : 0.959230769231
```

Experiment #24:

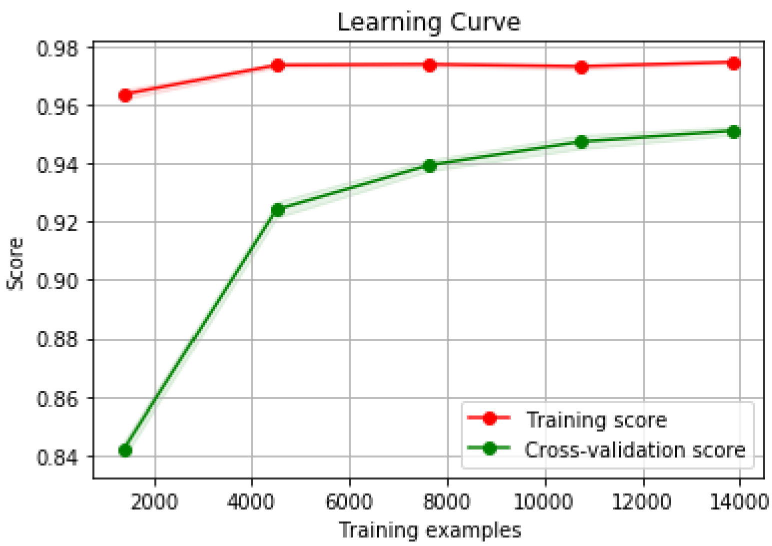
Results: Graph:




```
Accuracy : 0.949038461538
```

Experiment #25:

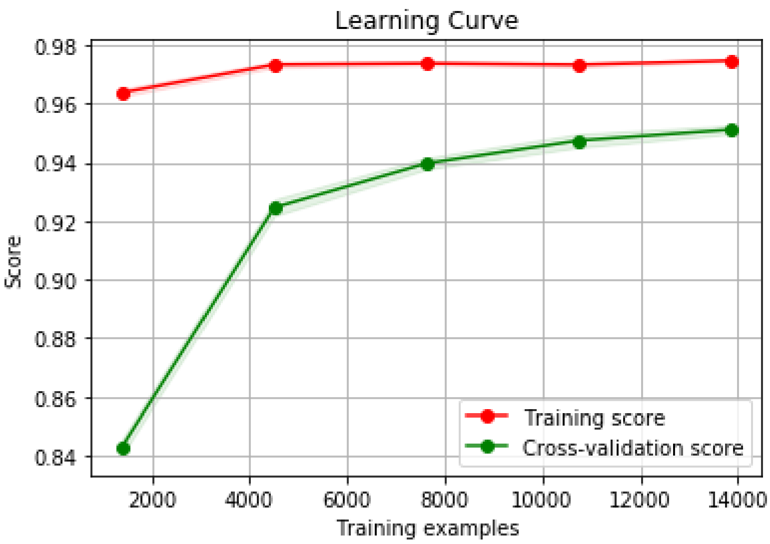
Results: Graph:




```
Accuracy :  0.959230769231
```

Experiment #26:

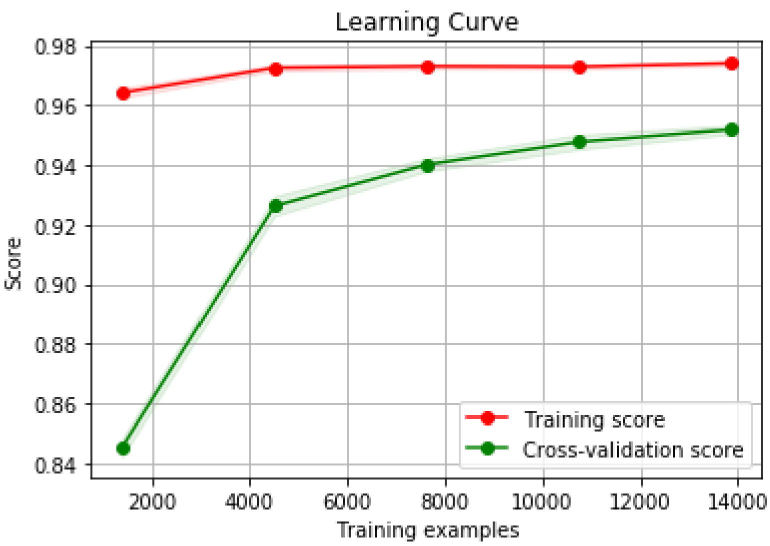
Results: Graph:




```
Accuracy : 0.959038461538
```

Experiment #27:

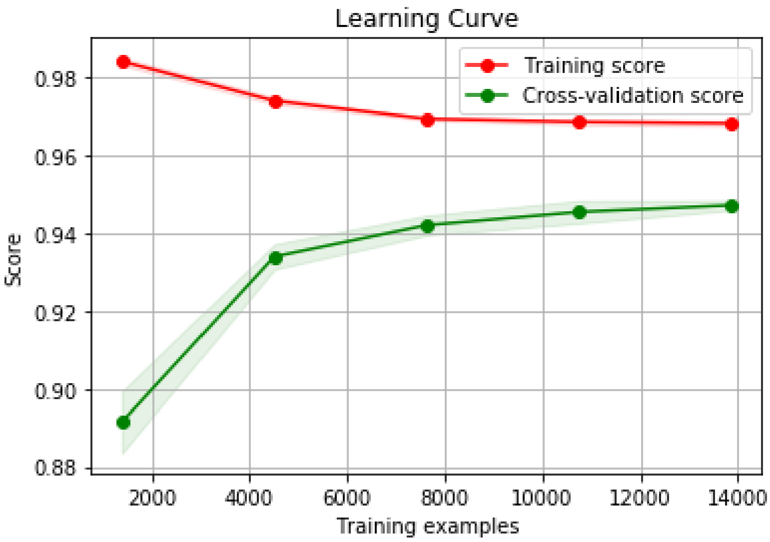
Results: Graph:




```
Accuracy : 0.958269230769
```

Experiment #28:

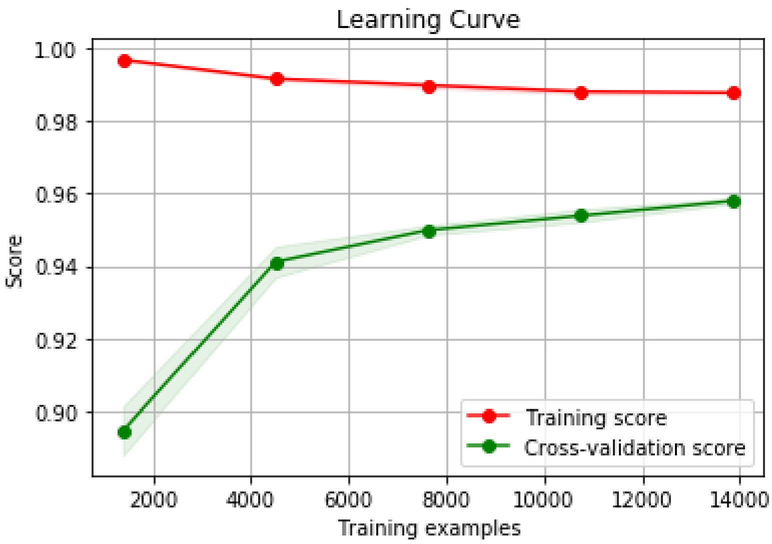
Results: Graph:




```
Accuracy : 0.952307692308
```

Experiment #29:

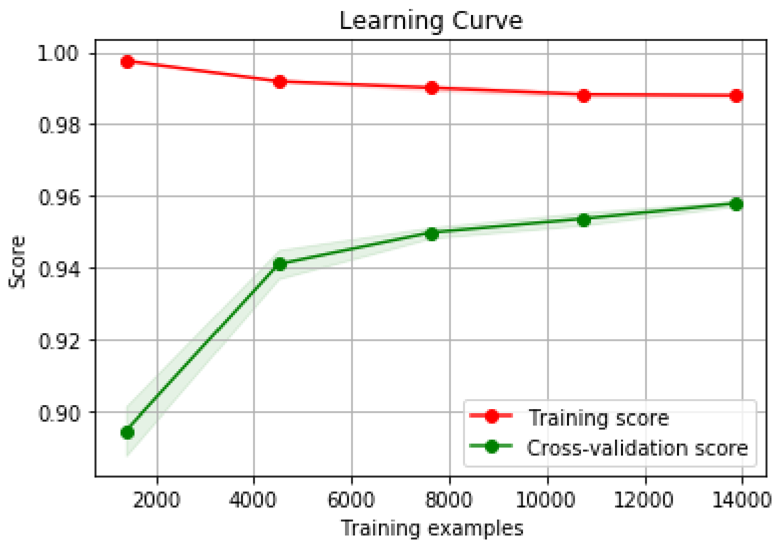
Results: Graph:




```
Accuracy : 0.964230769231
```

Experiment #30:

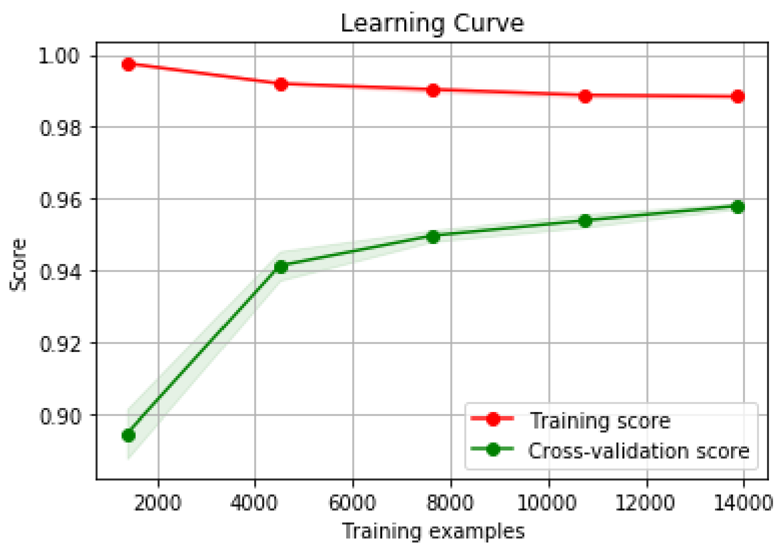
Results: Graph:




```
Accuracy : 0.964230769231
```

Experiment #31:

Results: Graph:




```
Accuracy : 0.964230769231
```

Discussion

In CNN approach we used loss function (sparse_categorical_crossentropy) because our target is integer’s numbers not on binary values.

In Experiment#1, we used also adam optimizer and set out number of neurons to 512, 256, 64, 4 All of these give us a large percentage of accuracy in accuracy of training and validation in experiment#2: we used sgd optimizer instead of adam optimizer but it gives very low accuracy because SGDs are smoother after 100 epochs than adaptive optimizers because the later have larger update value when they reach steady state (where 2nd-order momentum is small , then sgd to give high accuracy we must change on momentum parameter make large number of epoch to learn from this momentum than the other approaches which don't need high epoch like sgd In experiment#3 : we replace the activation function in last layer from softmax to relu , softmax preferable to use in last layer because it consider as probability distributed function the sum of all scores equal 1 , so determining which class is the right become easier , but in relu it make every negative value equal zero (0,max) , so the accuracy with softmax higher than relu

In experiment#4: when we decrease number of neurons the learning from data decrease but not with large scale

In KNN: In experiment#5 : we set 11 neighbor and begin showing how far it similar to them , in this experiment we use parameter weight which is parameter in prediction , we make weight='uniform' which give equal weight to all neighbor , and weight='distance ' it means that the closer neighbor influence more than other neighbor In experiment#6: we set 5 neighbor and begin showing how far it similar to them, here we change the hyper parameter and according to our data the accuracy decrease according to k=11 and also use weight='uniform' which give equal weight to all neighbor, and weight='distance' it means that the closer neighbor influence more than other neighbor In experiment#7: we set 11 neighbor and use only weight ='uniform' , here all neighbor affect equally on testing data , and the accuracy become less than when we make the closer neighbor effect on testing data In experiment#8: we set 11 neighbor and use only weight ='distance' , all neighbor doesn't have equal weight and according to our data it affects the accuracy and decrease

finally , Choosing the optimal K is almost impossible for a variety of problems , as the performance of a KNN classifier varies significantly when K is changed as well as the change of distance metric used and how they are uniformly distributed

In SVM algorithm we experimented it with TFIDF and BOW document representation as we see the result above when we changed the max features in TFIDF from experiment 9 to 10 the accuracy drops down very fast which mean that the more feature we specify the more accuracy we get, and also when we change the document representation to BOW the accuracy drops to 94 which make sense , since the bag of word technique doesn't care of the words order but TFIDF is the term weight that creates feature vector for the document, where each feature is a word (term) and the feature's value is a term weight. In experiment 12 we change the use_idf parameter to false which means we used TF only but what is the difference between TF and TFIDF ? The difference of TF and TF/IDF is on whether the corpus-frequencies of words are used or not. The TF/IDF is by far a better choice, independent of classifier. Using only TF we don't really care if a word is common or not. Thus, common words like articles receive a large weight even if they contribute no real information. In TF/IDF the more frequent a word is in the corpus, the smaller weight it receives. Thus, common words like articles receive small weights but rare words, that it is assumed to carry more information, receive larger weights. Due to the results it seems that we don't really care if a word is common or not since the common words are removed (stop words). By changing the gamma to 100 strangely nothing changed the same accuracy and same everything.

In RNN model we experimented it with Word embedding and BOW document representation as we see the result above when experimented the RNN with bag of words and word embedding it seems that word embedding result best due to word embedding is an improvement over more the traditional bag-of-word model encoding schemes where large sparse vectors were used to represent each word or to score each word within a vector to represent an entire vocabulary is replaced with dense vectors where a vector represents the projection of the word into a continuous vector space.

In NB algorithm we use TF-IDF or BOW and calculate the accuracy according to validation data. from experiment 16 to 19 we use NB with TF-IDF and exchange the max_Feature This parameter is absolutely optional and should be calibrated according to the rational thinking and the data structure. Sometimes it is not effective to transform the whole vocabulary, as the data may have some exceptionally rare words, which, if passed to tf_transformer.fit_transform() will add unwanted dimensions to inputs in the future. One of the appropriate techniques, in this case, the best result of this experiment when we make max_feature = 9500. From experiment 20 to 23 we use NB with BOW and exchange the max_Feature as a four experiment above the best accuracy appear when make the Max_Feature = 11500.

In LG algorithm we use TF-IDF or BOW and calculate the accuracy according to validation data. Form experiment 24 to 27 we use LG with TF-IDF and exchange the max_Feature as a four experiment above the best accuracy appear when make the Max_Feature is default value.form experiment 28 to 31 we use LG with BOW and exchange the max_Feature as a four experiment above the best accuracy appear when make the Max_Feature =11500.

Conclusion

In this project we worked on News Classification problem we experimented many algorithms K-nearest neighbor , Logistic regression , Naïve bayis , Support vector machine and Neural network models like CNN and RNN, we used different document representation like Bag of words , TF/IDF and Word embedding . All the training done on the News-aggregator data set which is labeled data set, After trying different models it turns out the accuracy ranges from 34% to 99% depends on hyper parameters in each model. we find CNN model in experiment 1 is better than other due to its accuracy that we recommended.