SCIENTIFIC DOCUMENT CLASSIFICATION

Vaibhavi Bhardwaj - 30154987
Hassaan Ali Khan - 30703700
Gayatri Aniruddha - 30945305

November 6, 2020

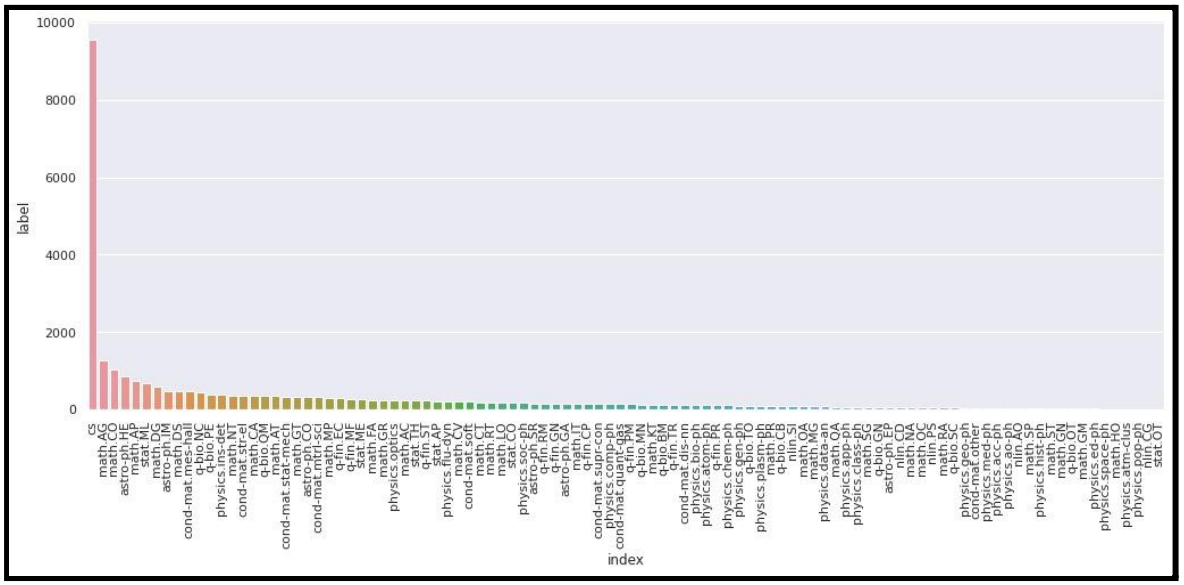
Abstract
This project is about scientific document classification on the dataset which was provided as part of the Applied Data Analysis course at Monash University. This project revolves around data pre-processing, feature extraction, and developing classifiers, and fitting these classifiers to the testing data. The three main models that have been discussed and implemented are Logistic Regression, SVM, and RNN. The main features which have been extracted are the TF-IDF vectors. The highest accuracy on the testing dataset was around 53.8% on 30% of the test data which was achieved with the Logistic Regression model.

Table of Contents

1 Introduction

In our assignment, we have been provided with the following two datasets. The training dataset - **train_data_labels.csv** contains around 29,638 articles, and 100 classes. It contains training ids, abstracts, and labels. The testing dataset - **test_data.csv** consists of around 7410 articles. Here, only testing ids and abstracts are available.

Also, from the below diagram we can see that 32% of the dataset contains the “cs label”.The data exploration was done using seaborn in Python.



This assignment was implemented using Python 3 and revolves around basically three major steps i.e generation of features, development of proper classifiers, and application of the classifier to the testing data.

label	most correlated unigrams	most correlated bigrams
astro-ph.HE	gamma and ray	high energy and gamma ray
cond-mat.mtrl-sci	alloy and perovskite	principle calculation and first principle
cs	network and learning	state art and neural network
math.AG	variety and sheaf	smooth projective and modulus space
math.CO	vertex and graph	chromatic number and graph vertex
math.DG	manifold and curvature	scalar curvature and riemannian manifold
math.IT	code and coding	crosstalk avoidance and network coding
physics.ins-det	calorimeter and detector	double beta and beta decay
stat.ML	gradient and learning	machine learning and gradient descent

We analyzed what kind of bigram and unigram we are encountering to validate the usage of ngram and we can see unique bigram words that can relate to a specific label.

This classifier should be able to assign a set of scientific abstracts to their corresponding labels. Hence, we divided the work into three separate sections: Data wrangling and data pre-processing, Model development, and Report consolidation.

2 Pre-processing and feature generation

Preprocessing is a very systematic approach where we convert our text into a form through which we can analyze the data. Data preprocessing is important because we cannot provide the text as input to classification algorithms. With appropriate data preprocessing, we can drastically reduce the size of the vocabulary, reduce the dimensionality, and finally improve the performance and accuracy! The training and testing data which has been provided to us by the University has not been processed yet.

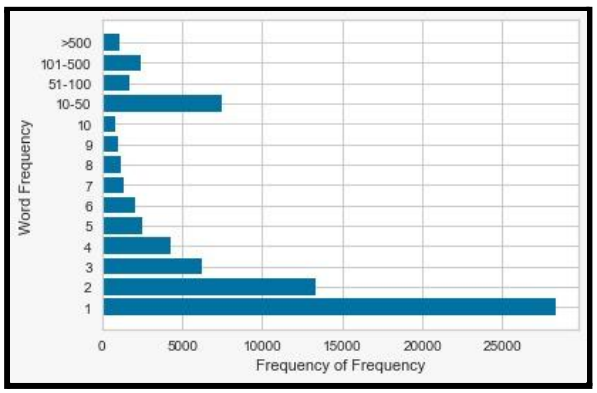
2.1 Pre-processing

We have followed the following steps to pre-process our data:
We have then used a TF-IDF transformer and this is done only for supervised models. We would be using this transformer to generate the features. This transformer performs the following processing on the data:

- **Case Normalisation**: Here, we have converted the entire text to lower-case.
- **Tokenization:** The remaining text has been converted into tokens.
- **Removal of least/most frequent words:** We have removed words occurring in less than
- **Removal of short-length words:** Words with less than length 3 have been removed.
- **Stopwords:** Stopwords are the most commonly occurring words in the English language.
    - Stopwords have been removed.

2.2 Feature generation

We have extracted the following features after the text-processing performed **(for Supervised Learning)**:

- **N-grams generation:** Unigrams, bigrams, and trigrams were generated.
- **Feature extraction:** Also, here we have extracted only around 50,000 features.
    - The following graph gives us a clear picture of the word frequency. With this, we will be able to set the features parameter.
    - 

    - We can see that there are words with frequency 1 that occur in 25,000 documents. Hence, we need to consider the words which occur more frequently. Thus, we consider the addition of the remaining words.
    - **TF-IDF** i.e Term Frequency and Inverse Document Frequency have been generated. They are a measure of how important a given word is to a document of a collection.
    - **Count-Vectorizer:** CountVectorizer gives us a simple, easier way to tokenize a collection of text. We can then even build a vocabulary of known words using those tokens and encode new documents using that vocabulary. We did implement it in the beginning. However, we did not get a decent value of accuracy. Hence, we did not further implement it.

Feature generation for RNN **(for Unsupervised Learning)**:

- Here, the feature generation for RNN is by the usage of GloVe (https://nlp.stanford.edu/projects/glove/) unsupervised algorithms to convert words to their vector representations is the only feature generation. The resultant representation provides linear structures of vector space. We have transformed the data from each abstract to a vector representation which is taken further for the other layers of the RNN.
- As in unsupervised learning features, extraction and classification models are combined into the layers of the neural network. Hence, we can say that the Glove is embedded as one of the layers of the neural net.

3. Models

We implemented many models: Logistic Regression, SVM, NaiveBayes, CNN, RNN, SGD and Random Forest. Then, based on the accuracy values of our predictions, we have decided to go ahead with the following three models. We have developed the following three classifiers to predict the values of labels corresponding to every abstract:

- SVM: Support Vector Machine Model
- Logistic Regression Model
- RNN: Recurrent Neural Network Model

We started with the SVM classifier. To improve accuracy, we then added some more data and developed the Linear Regression Model. The preprocessing and feature extraction remains the same for the models developed. The only thing different is the combination of the dataset used to fit and develop the model.

3.1 Model 1: SVM Model

Basic Idea of the model:

- The basic of SVM is that the algorithm creates a line or a hyperplane which separates our input data into classes.
- The Support Vectors Classifier finds the best hyperplane which is then used to separate the different classes present. This is done by maximizing the distance calculated between the different sample points and the hyperplane.
- The linear SVM uses a linear kernel and this is just a linear interpolation.

Construction and model development:

- Here, we have only used the dataset provided to us: train_data_labels and test_data.
- The training dataset and the testing dataset have been transformed using the TF-IDF transformer and we get the processed datasets that would be used to fit and transform the model.

Assumptions:

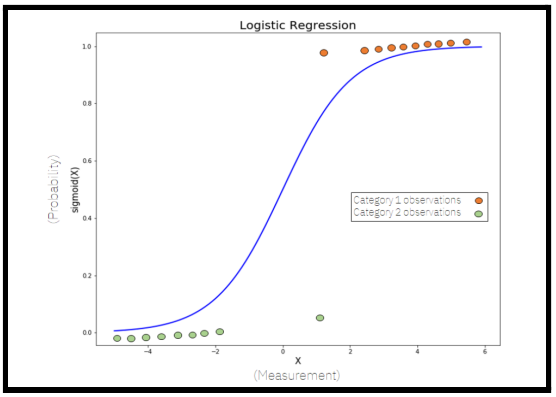
- The dataset which has been used to fit the data is identically distributed.

Accuracy:

We have achieved an accuracy of 52.7% on our training and testing dataset respectively, using the SVM'linear' model.

3.2 Model 2: Logistic Regression Model

Basic Idea of the model:



- The logistic regression model uses a logistic function through which we model a binary dependent variable.
- This model uses a supervised classification algorithm.
- Logistic Regression is used to define a relationship between one dependent binary variable and one or more independent variables.

Construction and model development:

- Here, we have only used the dataset provided to us: train_data_labels and test_data.
- The training dataset and the testing dataset have been transformed using the TF-IDF transformer and we get the processed datasets that would be used to fit and transform the model.

Assumptions:

- Here, the various observations have to be independent of each other.
- The level of collinearity between the different independent variables has to be less.
- This model works pretty well when we have a huge dataset.
- This model also assumes a dependent structure of the variables.
- Also, this model assumes the linearity of log odds and independent variables.

Accuracy:

- We have achieved an accuracy of 51.6% and 51.9% on our training and testing dataset respectively.

3.3 Model 3: RNN

Basic Idea of the model:

- It's clear that the data we have to train the model is highly skewed. Moreover, 100 classes need to be classified. In this case, CNN isn't recommended as it is a neural net for image classification and text classification where the number of classes is less than 100.
- RNN is similar to Feed-Forward Neural Network but has a looping factor at each neuron. The looping is responsible for handling the series factor involved in the data. This was the only reason to choose RNN over CNN.

Construction and model development:

```
Bidirectional LSTM
Model: "functional_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 10000)]           0
_________________________________________________________________
embedding (Embedding)        (None, 10000, 100)        6321700
_________________________________________________________________
bidirectional_4 (Bidirection (None, 200)               160800
_________________________________________________________________
dense_5 (Dense)              (None, 100)               20100
=================================================================
Total params: 6,502,600
Trainable params: 6,502,600
Non-trainable params: 0
_________________________________________________________________
```

- InputLayer + Embedding: The model takes into consideration 10000 words from each abstract and converts it into a vector representation of dimension 100 by calling the GloVe (6B tokens, 400K vocab, uncased, 100d) unsupervised learning algorithm.
- Bidirectional LSTM: It takes the input from the embedding and layer and is an extension of LSTM. The bidirectional of LSTM takes care of the recurrent nature of the RNN. This layer is given 200 neurons.
- The Dense layer at the end has 100 neurons as we have 100 classes.

Assumptions:

- The model training is done on the university data training data provided. The neural network is fitted by taking the number of epochs as 10 and the batch size as 20. The model is slow as it is sequential and bidirectional.
- A 25% validation split is applied to the training data for checking if the model overfits the training data. The model stops learning the weights if the validation accuracy drops down consecutively for 3 epochs and assigns the model the best weights after breaking the epoch loop.

3.4 Discussion of model difference(s)

Model 1: SVM Model

Advantages:

- SVM is a linear model that is useful for linear, non-linear practical problems.
- SVM works well when there is a clear line of separation between the various classes.
- This model efficiently uses the various boundary cases to build the separating curve between the various classes.

Disadvantages:

- An SVM model is not suitable for large datasets and does not perform well when the target classes overlap each other. Furthermore, SVM fails to perform accurately when we have a large number of classes to classify.
- This model is really expensive to train the models.
- Since the SVM Modelworks by classifying the data points above and below the separating line or hyperplane, the probabilistic explanation for the required classification is not provided.

Model 2: Logistic Regression Model

Advantages:

- Since we have a large, the logistic regression model works well with our dataset.

- This model is very simple, straightforward, easier to implement, and is very efficient to train our dataset.
- Furthermore, it can easily be interpreted.

Disadvantages:

- The major limitation of this model is that it assumes a linear relationship between the dependent and independent variables.
- This works well only when the variable to be predicted is binary.
- It cannot function well when there are too many features or variables to deal with.
- Using this model, it is really hard to obtain complex relationships. Thus, to further improve the accuracy of the model, we need to use algorithms such as Neural Networks.

Model 3: RNN Model

Advantages:

- RNN can be used to process inputs of any length.
- The model size does not increase with increasing input.
- The RNN model computation takes into account historical information as it's good for data that are serial in nature.

Disadvantages:

- The computations are slow and they take time.
- Takes more time to train the model compared to CNN.
- It becomes difficult to retrieve information from a long time ago.
- This model cannot consider future input for a current state.

4 Experiment setups

We implemented many models other than the models mentioned above. We implemented four models of SVM, one KNN model, and four models of Logistic Regression.

Handling Imbalanced Data:

We observed that there were some classes that were occurring in less than 1% of the documents. These classes were added to a low-list. This caused an imbalance in our dataset. In order to overcome the imbalance, we did remove it. However, this removal further dropped down accuracy. Hence, we did not implement this.

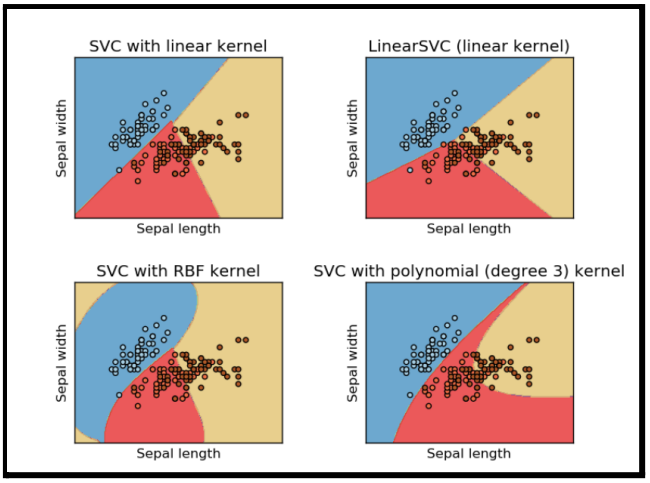
Four Models of Logistic Regression along with their different parameters:

Here, while implementing the different models of Logistic Regression, we change the solver parameters. We have considered the following solvers: sag, saga, lbfgs and newton-cg.

Parameter: solver

The sag solver is faster than when compared to the other solvers for large datasets when we have both a large number of samples and a large number of features. Similarly, the saga solver is a different variation of the sag solver which handles large datasets and is mainly used to deal with multinomial logistic regression. The blog's solver is handy for smaller datasets. However, this solver's performance drops for larger datasets.

Four Models of SVM along with their different parameters:



Here, the different models of SVM were implemented by changing the kernel values.
Parameter Kernel: We have assigned a linear kernel, polynomial kernel, and a rbf kernel. Here, the kernel tells us the type of hyperplane to be used to separate the data. By using a linear kernel, we are using a linear hyperplane, which in the case of 2D data is just a line. Similarly, by using a "rbf" or a "poly" kernel, we are using a non-linear hyper-plane.
Parameter Gamma: Here, gamma value is a parameter which is used for non-linear hyperplanes. The higher the value of gamma, the higher the model tries to exactly fit the training data set. This increasing gamma value can also lead to over-fitting.
Parameter Degree: Degree is used when the kernel is assigned the value poly. It refers to the degree of the polynomial which is used to find the hyperplane to split the data.
Parameter C: C is the penalty parameter of the error term. This term controls the trade-off between the boundaries and correct classification of the training points. Increasing the value of C may also lead to overfitting.

Cross validation:
We have used cross-validation to split the training dataset into further two subsets of training data and testing data, using a split percentage of 30%. We have done this to cross verify the accuracy of our subset test dataset with the dataset present on Kaggle. We have applied 'stratified' on the label column to have a split that will have all labels in both subsets for more accurate and valid outcomes.

Cross-validation is also used to calculate the score of a model using `cross_val_score`. We have further combined it with `RepeatedStratifiedKFold` to achieve healthier outcomes. For KFold we have assigned 5 numbers of folds and 3 numbers of n_repeats (cross-validation repetition) and a random state. In cross_val_score we have predicted the score (accuracy) for training dataset K-FOLD times calculated the mean and standard deviation of it as the final score.

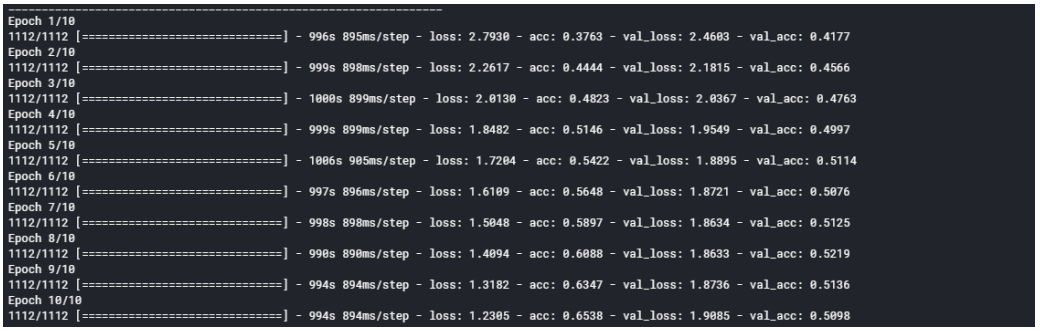
RNN along with their different parameters:
For the KNN model, we have 10000 tokens as the input layer. We have selected GloVe for making vectors for our tokens. The total number of distinct tokens formed is 63k thousand in number for our model. The below code shows the compiling parameters used by the neural network.

```
model.compile(loss='categorical_crossentropy',
optimizer='rmsprop', metrics=['acc'])
```

- **Loss:** 'categorical_crossentropy' is for a multi-class dataset. Meaning that each data point is of a single class. 'binary_crossentropy' cannot be used for our dataset as it is not multi-label.

- **Optimizer:** It is an optimizer that uses the RMSprop algorithm. 'rmsprop' is similar to the gradient descent algorithm with momentum. With the help of this optimization algorithm, the model can learn faster as the model takes larger steps in the horizontal direction. Furthermore, the 'rmsprop' avoids oscillation vertically and Optimizer selection is relative to the model requirement.
- **Metric:** The metric to measure the model performance is taken as the accuracy of prediction i.e how often correct predictions are made. The same is stated in the assessment description that the model performance is calculated by taking the percentage of correct predictions.

Validation steps for the given **RNN** :



- The model uses a training-validation approach to evaluate the performance of the model. The validation dataset is 25 percent of the training data provided.
- It can be seen in the above snip how the validation accuracy is increasing as the number of epochs is increasing. Furthermore, we have added an early stopping criterion for the epochs run when it encounters overfitting of training data or when the validation accuracy is lower than it's previous 3 consecutive runs. The best weights are stored in the model and later on used for prediction.

5 Experimental results

We tried several models in order to achieve maximum accuracy. The following table gives us the accuracy achieved for training and testing data.

Note: We have only considered the best accuracy based on different parameters applied, also for whom we have uploaded our result we have changed our best based on the provided test regardless if it is best in training or not.

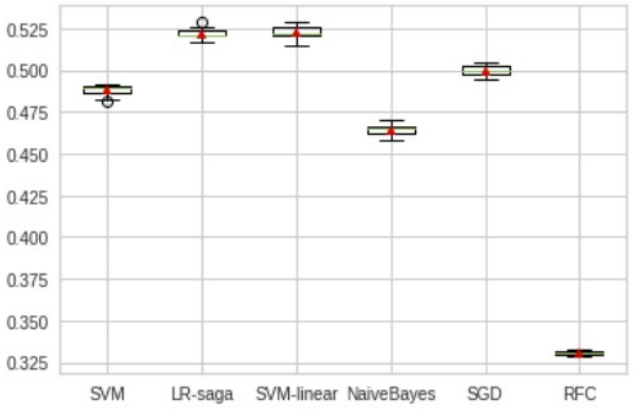
Model Name	Types	Training Data Accuracy	Validation Data Accuracy	Testing Data Accuracy
Logistic Regression	lbfgs	0.516	0.519	0.44669
	saga[1]	0.516	0.519	0.53801
	newton-cg	0.516	0.519	-
	sag	0.516	0.519	-
SVM	LinearSVC	0.515	0.506	-
	linear[2]	0.527	0.527	0.53396
	poly	0.452	0.434	-
	rbf	0.452	0.434	-
NaiveBayes	-	0.457	0.470	-
CNN	-	0.52	0.48	0.48088

[1] This was considered for submission because of quicker transformation and better large data handling

[2] Linear had by far the best svm outcomes on gamma

RNN Model [3]	Bidirectional LSTM	0.65	0.510	0.53036
SGD	log	0.498	0.496	-
Random Forest	-	0.330	0.331	-

We can see from the above values that the best three models which gave us the highest values of accuracy were: Logistic Regression, SVM Model, and RNN Model, and hence they have been implemented.



The above plot gives us a clear view of the training accuracy for some models. The lowest accuracy is of Random Forest Classifier and the highest is of LR-Saga. SVM linear and LR-Saga are competitive in nature.

6. Conclusion

Thus, we were successfully able to preprocess our datasets, develop classifier models and test the model on our testing dataset. There was a clash between the Logistic Regression Model and RNN Model.

Thus, since RNN is slower, based on our analysis and experimentation, we discovered that the optimal classifier was the Logistic Regression Model. Also, the best set of features which were used by the classifier are : TF-IDF.

We learnt a lot of lessons while working on this project. We learnt the importance of data preprocessing which helps in decreasing the vocabulary size and improving the performance and accuracy.

We realised that the training data which was provided by the University was highly imbalanced and this led to the addition of external data. There were certain labels which were assigned to only 3 rows. Similarly, there were more than 500 rows assigned to certain labels. Thus, this caused an imbalance in the dataset. Furthermore, the test data was skewed.

Another problem that faced was that of over-fitting of the model. In the earlier stages of the assignment, we considered the entire training data to train the model. In order to overcome this problem, we split the training dataset into training and test subsets to check the accuracy values.

[3] train-validation : validation size equal to 25% of training data

After experimenting with more than 10 models, we were able to study the advantages and disadvantages of the models. It was also a good idea to provide us with 30% of the testing dataset. Also, we were successfully able to learn the various advantages and disadvantages of the models.

References

Jiahao Wang (2019, August 30). NLP Text Processing: A Practical Guide and Template
Retrieved from: https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79

Rafael Zubairov (2017, October 20). Importance of data preprocessing in NLP/NLU
Retrieved from:
https://medium.com/@zubra.bubra/importance-of-data-preprocessing-in-nlp-nlu-d8098f0b6620

Mohtabi Ben Fraj (2018, January 5). In Depth: Parameter tuning for SVC
Retrieved from:
https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

Jaime Zornoza (2020, February 9). Logistic Regression Explained
Retrieved from:
https://towardsdatascience.com/logistic-regression-explained-9ee73cede081

sklearn.linear_model.LogisticRegression -- scikit-learn 0.23.2 documentation. (2020)
Retrieved from:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Beams Adept(2020, July 24) Logistic Regression python solvers' definitions
Retrieved from:
https://stackoverflow.com/questions/38640109/logistic-regression-python-solvers-defintions/52388406#52388406

Rens(2017, November 21). Sklearn: adding lemmatizer to CountVectorizer
Retrieved from:
https://stackoverflow.com/questions/47423854/sklearn-adding-lemmatizer-to-countvectorizer

Rohith Gandhi (2018, June 8). Support Vector Machine - Introduction to Machine Learning Algorithms
Retrieved from:
https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Aravind Pal(2020, February 17) CNN vs RNN - Analyzing 3 Types of Neural Networks in Deep Learning.
https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/

missinglink.ai (2020, November 8) 7 Types of Activation Functions in Neural Networks: How to Choose?
Retrieved from:
https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/