**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

****

**PHASE-II REPORT**

On

**TOPIC BASED NEWS CLASSIFICATION**

*Submitted in partial fulfilment of the requirement for the award of Degree of*

*Bachelor of Engineering*

*In*

*Computer Science and Engineering*

*Submitted by:*

|  |  |
| --- | --- |
| HARIKA H P  CHAKARALA NAVYASREE | 1NT18CS051  1NT18CS030 |
|  |  |

Under the Guidance of

Dr.Vani Vsaudevan

Professor, Dept. of CS&E, NMIT



Department of Computer Science and Engineering

2021-22

**CHAPTER 1: ABSTRACT**

News is available in various places like website, television, papers etc. in online and offline modes, which includes politics, business, entertainment, sports and other general categories. Automatic classification of the news articles is very important for fast and effective communication. Text processing and text mining are basic techniques involved in news classification. News classification helps in various emerging applications. This project is focusing on the topic-wise news classification using Artificial Intelligence techniques. Content Mining is the application of mechanized approaches for understanding the information available in the content archives. Text Mining is delineated in a way to assist the business find out essential knowledge from text-based content. With the help of lexical approach and machine learning techniques we can perform news classification to demarcate them into distinct topics for easy deliberation.

A few of the papers from the literature in the context are studied and analysed for understanding the problem domain and the solution approaches. In our project AI based methods are implemented for doing the classification of news with topic discovery approach.

This report presents results of experimenting with a few algorithms for developing models for news classification, viz., simple baseline model, decision tree, Random forest, multi-nominal Naïve Bayesian, Multi-layered perceptron, and Support vector. Their performances are compared based on precision, recall, F1 score measure. Out of them Bayesian method and MLP are found to be the best ones. Further experiments can be done with deep neural networks.**CHAPTER 2: CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** |  |
|  | **Certificate** | ii |
|  | **Declaration** | 1 |
| 1 | **Abstract** | 3 |
| **3** | **Introduction**   * 1. Research Motivation & Problem statement      1. Research Motivation      2. Statement of the problem   2. Research Contributions and Objectives      1. Primary Objectives      2. Main Contributions | 6 |
| **4** | **Data Source and Data Quality** | 6 |
| **5** | **Data Pre processing** | 6 |
| **6** | **Machine Learning Methods** | 8 |
| **7** | **Results and discussion** | 22 |
| **8** | **Conclusion** | 28 |
| **9** | **Lessons Learnt** |  |
| **10** | **References** | 29 |
| **11** | **Appendix** |  |

**CHAPTER 3: INTRODUCTION**

* 1. **Research Motivation**

Text classification is the process of assigning text documents to one or more predefined

categories. This allows users to find desired information faster by searching only the relevant categories and not the entire information space. The importance of text classification is even more apparent when the information space is huge such as the World Wide Web. Examples of web classification systems include Yahoo! directory and Google web directory. However, such classification services are carried out by human experts, and they do not scale up well with the growth rate of web pages on the Internet. To automate the classification process, machine learning methods have been introduced. In a text classification method based on machine learning, classifiers are built (trained) with a set of training documents. The trained classifiers can therefore assign documents to their suitable categories.

Online news articles represent a type of web information that is frequently referenced. Currently, online news is provided by many dedicated newswires such as Reuters and PR Newswires. It will be useful to gather news from these sources and classify them accordingly for ease of reference. A working news classification system, named Categoriser is presented by a researcher that performs automated online news classification. Categoriser adopts SVM classification method to classify news articles into categories. These categories can be either a set of predefined categories, i.e., general categories, or special categories defined by users themselves. The latter are also known as the personalized categories. With personalized categories, Categoriser allows users to quickly locate the desired news articles with minimum effort. All these form the motivation for news classification work proposed here.

* 1. **Statement of the Problem**

Our goal in this project was to build a classifier that can determine what genre of the news given article relates to. Ideally, such a classifier would both be able to identify most of the categories widely used in the world of news, although which set of the categories to use is hard to choose, since many different news sources have slightly different methods of categorizing new; may be one source uses “science” while another has “technology”, and they have subtly different sets of articles that they contain that would be hard to distinguish. We use a few news topics based on the literature, although many alternate categories could be used; it would be interesting to see whether our method in fact maintained all of its effectiveness on those categorizations. This is a useful problem to solve, since it has many applications; for example a, search engine may want to aggregate news from many sources on a specific topic, like business news, and so it may want to be able to scan a wide variety of sources for their content and not just rely on the classification the sources themselves use, since as addressed above, different sources may classify news in many subtly distinct ways. It could also be used to analyze trends in reporting, for example, do articles about politics tend to use more words related to emotion than article about science?

* 1. **Aim and Objectieves**
     1. **Primary objectives**

1. To study various AI techniques for news classification from literature and analyse the open issues and challenges and also find out the best algorithm.

2. To design & develop a prototype model for implementing AI algorithms for the news classification

3. To implement and evaluate the AI algorithm for topic based multiclass classification of news items

4. To design an alternate method/algorithm that can perform better for the news classification; implement and evaluate this algorithm to confirm its performance advantages.

* + 1. **Main contributions**

The main contribution of this project is support for the idea that machine learning could be useful in a novel way for the task of classifying news.

**CHAPTER 4: Data Source and Quality**

**Data Source:** Taken from the **dataworld.com**

**CHAPTER 5: Data Pre-Processing**

**5.1 Vectorization of Data:**

* CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis).
* CountVectorizer creates a matrix in which unique word is represented as column and each test sample as row and value of each cell is count of the word n particular text sample.
* Lets look at what exactly is this vectorizer doing. We will first create reverse dictionary from the vectorizer. Iterating over the vectorized sentence \_Nasa scientists are good\_. We get the vector to be representative of three words "good", "nasa" and "scientists". The order has been changed because bag of words does not preserve order.
  1. **Feature Reduction:**
* It is one of the dimensionality reduction technique which allow us to minimize the number of features in a dataset by only keeping features that are important. this is One of the techniques to improve the performance of a machine learning model is to correctly select the features.
* As we deal with large dataset, we come across lot of features in which not everything is important. Adding these un-necessary feature leads to the reduce in over all accuracy and increase in complexity of model.
* So, we have used a Variance Threshold function, in this approach all features whose variance is les than the specific threshold value are removed. While using this method we assume that the features with higher variance are likely to contain more information.
  1. **Using SMOTE:**
* Almost every dataset has unequal representation of classes, this isn’t a problem until the difference is small, but if the difference is large, model do not work well in identifying the minority classes.
* So we use SMOTE (Synthetic minority oversampling technique). This aims to balance class distribution randomly increasing in minority class examples by replicating them.

**CHAPTER 6: Machine Learning Methods:**

**6.1 Algorithms Used:**

We took our data set from the website dataworld.com. We have tried to implement some algorithms on our data set to classify the news based on the category it belongs to.

The algorithms we have implemented are:

* **Baseline model:** A baseline is a simple model shown in fig. 5.1 that provides reasonable results on a task and does not require much expertise and time to build. This model uses heuristics or simple statistics or randomness to predictions in data set. The scores from these algorithms provide point of comparison when evaluating other machine learning algorithms.The main need to use this model is to understand the data. As in ML each problem is unique. So, we not know what algorithms we use and what attributes will be useful.

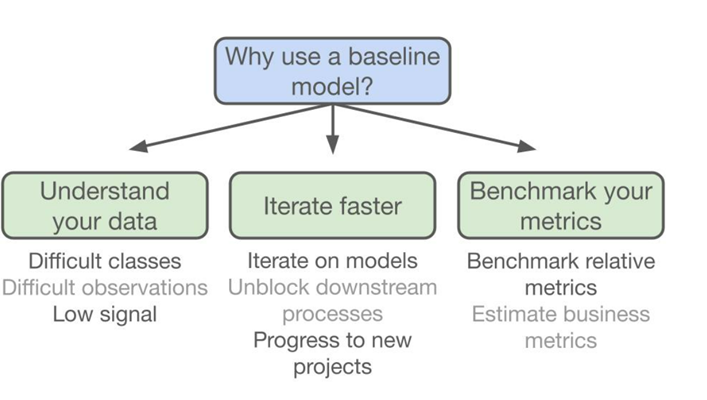


Figure 5.1 Base line model

* **Multinomial Naive Bayesian**

Naïve Bayes, This is one of the very efficient algorithms which are frequently used in text classification and this algorithm is easy to implement. This algorithm is based on Bayes theorem. Bayes theorem states that the features in the data set are independent; occurrence of one feature does not affect the probability of occurrence of other features. This algorithm calculates the probability of each tag for a given sample and then gives the tag with the highest probability.

* **Support vector Machine**

It is a supervised machine learning algorithm which can be used for classification and regression. SVM uses a technique called kernel trick to transform our data, based on these transformations it finds the optimal boundary between possible outputs. It has many unique features which are not preferred by other algorithms, one of this is features id this includes both negative and positive training sets. In this we plot each data item as a point in n- dimensional space (n is umber of features) than perform classification by finding Hyper plane of n-1 dimension. Main idea is to find a plane that separates positive and negative points and the distance between the positive and negative plane is maximum.

* **Multilayer perceptron**

A perceptron is a simple feed forward neural network. It is a classifier which maps input to output. In multilayer perceptron we will have one or more layers other than input and output layer and these layers are called hidden layers.in this all the layers are fully connected. First all the weights for a perceptron are given with small random numbers, then we adjust these weights so that the error becomes smaller and smaller.

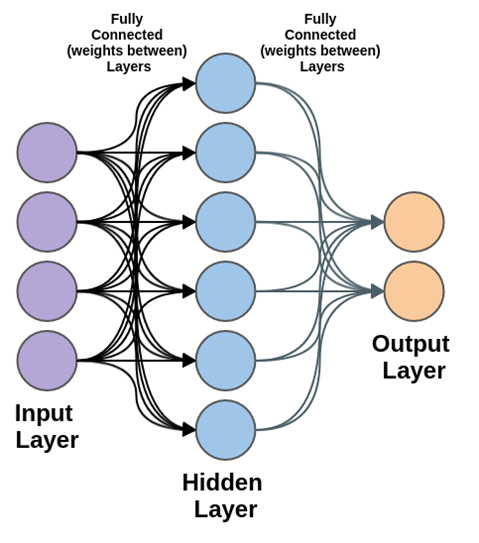


Figure 5.2 Multilayered Perceptron

* **Decision Tree**

A Decision Tree is a simple representation for classifying examples. In general, Decision tree analysis is a predictive modeling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions.

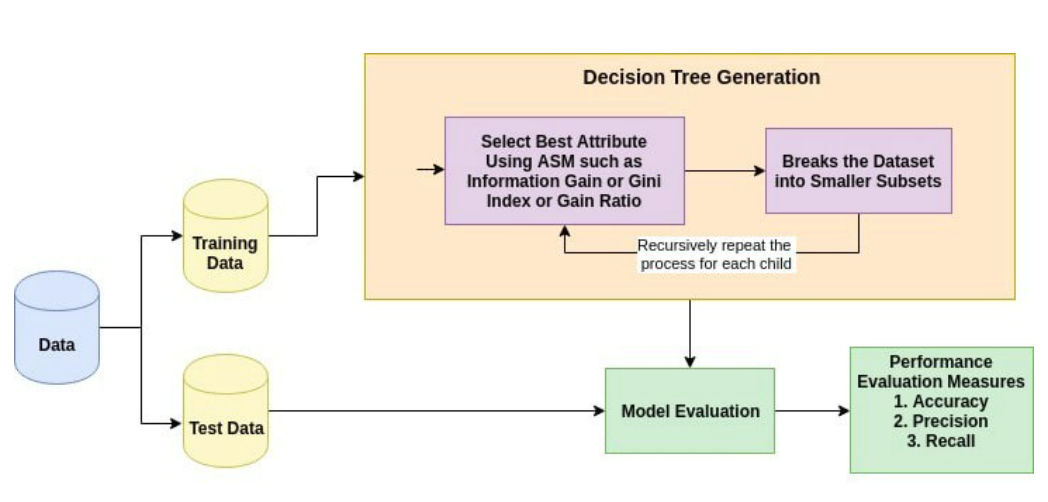


Figure 5.3 Decision Tree

* **Random forest**

Random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.

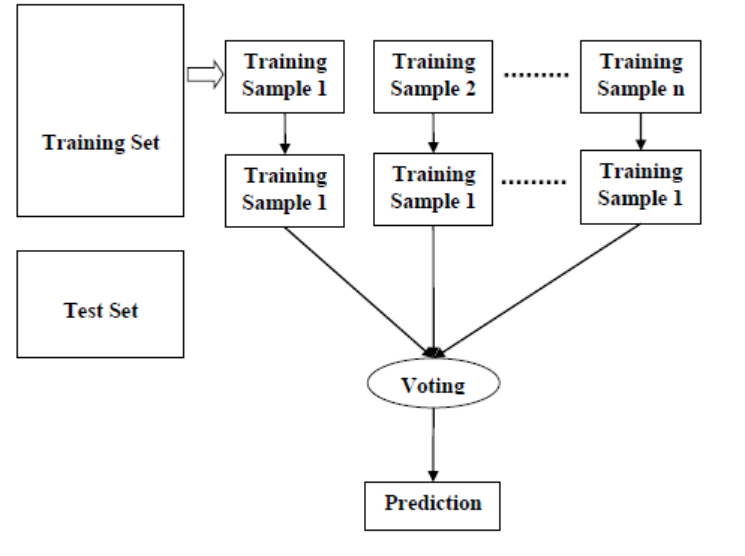


Figure 5.4 Random Forest

**CHAPTER 7: Results and Discussion**

**7.1 Calculating scores:**

###### 7.1.1 Accuracy

Accuracy is often the most used metric representing the percentage of correctly predicted observations, either true or false. To calculate the accuracy of a model performance, the following equation can be used:

In most cases, high accuracy value represents a good model, but considering the fact that we are training a classification model in our case, an article that was predicted as true while it was actually false (false positive) can have negative consequences; similarly, if an article was predicted as false while it contained factual data, this can create trust issues. Therefore, we have used three other metrics that take into account the incorrectly classified observation, i.e., precision, recall, and F1-score.

Accuracy = TP+TN/(TP+TN+FP+FN)

**7.1.2 Recall**

Recall represents the total number of positive classifications out of true class. In our case, it represents the number of articles predicted as true out of the total number of true articles.

Recall = TP/(TP+FN)

**7.1.3 Precision**

Conversely, precision score represents the ratio of true positives to all events predicted as true. In our case, precision shows the number of articles that are marked as true out of all the positively predicted (true) articles:

Precision = TP/(TP + FP)

**7.1.4. F1-Score**

F1-score represents the trade-off between precision and recall. It calculates the harmonic mean between each of the two. Thus, it takes both the false positive and the false negative observations into account. F1-score can be calculated using the following formula:

F1 = 2 x (precision x recall) / (precision + recall)

**7.2 Results:**

The F1 score values for the data world data set for the different models implemented are shown in the table 5.1 below. It is also provides means for comparative analysis.

Table 5.1. F1 score of different news topics as per different ML models

|  |  |  |
| --- | --- | --- |
| **MODELS** | **FEATURES** | **F1 SCORE** |
| **Base Line Model** | Business & Finance | 0.17 |
| Criminal Justice | 0.21 |
| Health Care | 0.16 |
| Politics & Policy | 0.26 |
| Science & Health | 0.21 |
| **Decision Tree** | Business & Finance | 0.35 |
| Criminal Justice | 0.43 |
| Health Care | 0.64 |
| Politics & Policy | 0.58 |
| Science & Health | 0.51 |
| **Random Forest** | Business & Finance | 0.35 |
| Criminal Justice | 0.43 |
| Health Care | 0.64 |
| Politics & Policy | 0.58 |
| Science & Health | 0.51 |
| **Multi-layered perceptron (MLP)** | Business & Finance | 0.55 |
| Criminal Justice | 0.61 |
| Health Care | 0.54 |
| Politics & Policy | 0.68 |
| Science & Health | 0.63 |
| **Support Vector Machine** | Business & Finance | 0.28 |
| Criminal Justice | 0.34 |
| Health Care | 0.52 |
| Politics & Policy | 0.67 |
| Science & Health | 0.52 |
| **Multinomial Naive Bayes Model** | Business & Finance | 0.55 |
| Criminal Justice | 0.61 |
| Health Care | 0.54 |
| Politics & Policy | 0.68 |
| Science & Health | 0.63 |

As can be observed from the table 5.1 , MLP, Naive Bayes and SVM are showing better performance compared to the other methods.

Naïve Bayes method uses these steps: Load data > Split train/dev/test > Remove stop words from title > vectorize title using bag of words and convert category to numbers > feature reduction using variance > oversampling of data to make distribution uniform > train various classifier on training data > use dev data to check f1-score > choose naive Bayesian model to predict the test data.

**CHAPTER 8: CONCLUSION & FURTHER WORK**

This report provides details about news classification considering the data world data set. We have tried various classifiers- Decision Tree, Support Vector Classifier, Multinomial Naive Bayesian Classifier, Multi-layered Perceptron, and Random Forest. Multinomial Naive Bayesian Classifier worked the best. It is logical for Multinomial Naive Bayesian to work the best as even we as humans classify based on keywords. We are likely to predict “Politics” is we see keywords like Obama, elections or republic and we are likely to predict “Criminal” if we see keywords like drugs, jail and so on. Naive Bayesian scans whole dataset and finds the probabilities of each word in headline being associated with a class and then finds the probability for whole headline hence it works well.

Further work involves exploring deep learning neural network models for processing the news text for classification. Also alternate data sets will be considered for evaluation. • Moreover, improvement can be made in terms of the time required to predict the category of an article by considering only the news headline. Consideration of only the news headline would eliminate tough statistical calculations and thus would decrease the overall time.

**CHAPTER 9: LESSONS LEARNT:**

* Learnt how useful the classification of the news actually is.
* And also learnt that how this news classification can be done using different machine learning algorithms.
* How different machine algorithms can give different accuracy and find what is the best fit model algorithm for our given classification of the news.

**Chapter 10:REFERENCES**

[1] Zhenzhong Li, Wenqian Shang, Menghan Yan, “News text classification model based on topic model”, IEEE International Conference on Data Mining Workshops (ICDMW), Volume: 1, Pages: 1375-1380, 2018

[2] Vrusha U.Suryawanshi, Pallavi Bogawar, Pallavi Patil, Priya Meshram, Komal Yadav, Nikhil S. Sakhare, “Automatic Text Classification System”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 4 Issue 2, February 2015.

[3] Ari Aulia Hakim, Alva Erwin, Kho I Eng, Maulahikmah Galinium, Wahyu Muliady, “Automated Document Classification for News Article in Bahasa Indonesia based on Term Frequency Inverse Document Frequency (TF-IDF) Approach”, 6th International Conference on Information Technology and Electrical Engineering (ICITEE), 2014.

[4] S. M. H. Dadgar, M. S. Araghi and M. M. Farahani, "A novel text mining approach based on TF-IDF and Support Vector Machine for news classification", IEEE International Conference on Engineering and Technology (ICETECH)*, C*oimbatore, India, 2016.

[5] Akshita Bhandari, Ashutosh Gupta, Debasis Das, “Improvised apriori algorithm using frequent pattern tree for real time applications in data mining”, International Conference on Information and Communication Technologies (ICICT), 2014.

[6] Nabamita Deb, Vishesh Jha, Alok K Panjiyar, Roshan Kr Gupta, “A Comparative Analysis Of News Categorization Using Machine Learning Approaches”, International Journal Of Scientific & Technology Research, Volume 9, Issue 01, January 2020.

[7] H. Duong and V. Truong Hoang, "A Survey on the Multiple Classifier for New Benchmark Dataset of Vietnamese News Classification," International Conference on Knowledge and Smart Technology, 2019.

[8] W. Wang, X. Cui and A. Wang, "News Analysis Based on Meta-synthesis Approach*",* IEEE International Computer Software and Applications Conference*,* 2008*.*

[9] U. Suleymanov, S. Rustamov, M. Zulfugarov, O. Orujov, N. Musayev and A. Alizade, "Empirical Study of Online News Classification Using Machine Learning Approaches”,IEEE International Conference on Application of Information and Communication Technologies (AICT), 2018.

[10] Gurmeet Kaur, and Karan Bajaj, “News Classification and Its Techniques: A Review”, IOSR Journal of Computer Engineering (IOSR-JCE), Vol.18, issue 1, 2016.

**CHAPTER 11: APPENDIX**

**Link to Dataset: https://data.world/elenadata/vox-articles**

**11.1.1 Python code:**

## Import Libraries

Loading all libraries to be used

**import** **copy**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **re**

**import** **nltk**

nltk.download('stopwords')

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn.feature\_extraction.text** **import** CountVectorizer

**from** **sklearn.preprocessing** **import** LabelEncoder

**from** **sklearn.feature\_selection** **import** VarianceThreshold

**from** **imblearn.over\_sampling** **import** SMOTE

**from** **sklearn.dummy** **import** DummyClassifier

**from** **sklearn.naive\_bayes** **import** MultinomialNB

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn.neural\_network** **import** MLPClassifier

**from** **sklearn.ensemble** **import** RandomForestClassifier

*#from sklearn.metrics import accuracy\_score*

*#from sklearn.model\_selection import cross\_val\_score, KFold*

**from** **sklearn.metrics** **import** confusion\_matrix

**from** **sklearn.metrics** **import** classification\_report

**import** **seaborn** **as** **sns**

[nltk\_data] Error loading stopwords: HTTP Error 503: No healthy IP

[nltk\_data] available for the backend

# Data preparation

## Load data

Lets load the data from dsjVoxArticles.tsv file. We will clean the title to remove special characters and punctuations. We will store title in titles and Category in categories

titles = []

categories = []

**with** open('dsjVoxArticles.tsv','r',encoding='utf-8') **as** tsv:

count = 0;

**for** line **in** tsv:

a = line.strip().split('**\t**')[:3]

**if** a[2] **in** ['Business & Finance', 'Health Care', 'Science & Health', 'Politics & Policy', 'Criminal Justice']:

title = a[0].lower()

title = re.sub('\s\W',' ',title)

title = re.sub('\W\s',' ',title)

titles.append(title)

categories.append(a[2])

We can print and check the data loaded in titles and categories

print("Titles-**\n**", "**\n**".join(titles[:5]))

print("**\n**Categories-**\n**", "**\n**".join(categories[:5]))

Titles-

bitcoin is down 60 percent this year here's why i'm still optimistic.

9 charts that explain the history of global wealth

remember when legal marijuana was going to send crime skyrocketing?

obamacare succeeded for one simple reason it's horrible to be uninsured

the best obamacare data comes from a home office in michigan

Categories-

Business & Finance

Business & Finance

Criminal Justice

Health Care

Health Care

## Split data

Split data into 3 parts - training, development and test. We will use training data to train out model and use development data to check and tune hyper parameters. And finally use test data to see how our model performs

title\_tr, title\_te, category\_tr, category\_te = train\_test\_split(titles,categories)

title\_tr, title\_de, category\_tr, category\_de = train\_test\_split(title\_tr,category\_tr)

print("Training: ",len(title\_tr))

print("Developement: ",len(title\_de),)

print("Testing: ",len(title\_te))

Training: 1779

Developement: 594

Testing: 792

Using wordCloud we can visualize our data.

**from** **wordcloud** **import** WordCloud

text = " ".join(title\_tr)

wordcloud = WordCloud().generate(text)

plt.figure()

plt.subplots(figsize=(20,12))

wordcloud = WordCloud(

background\_color="white",

max\_words=len(text),

max\_font\_size=40,

relative\_scaling=.5).generate(text)

plt.imshow(wordcloud)

plt.axis("off")

plt.show()

<Figure size 432x288 with 0 Axes>

# Data Preprocessing

## Vectorization of data

Vectorize the data using Bag of words (BOW)

tokenizer = nltk.tokenize.RegexpTokenizer(r"\w+")

stop\_words = nltk.corpus.stopwords.words("english")

vectorizer = CountVectorizer(tokenizer=tokenizer.tokenize, stop\_words=stop\_words)

*# splitting the senctences to words and removing stop words such as is,the and words which do not*

*#add meaning to the sentence.*

*# this type of transformation is known as bag of words .*

vectorizer.fit(iter(title\_tr))

Xtr = vectorizer.transform(iter(title\_tr))

Xde = vectorizer.transform(iter(title\_de))

Xte = vectorizer.transform(iter(title\_te))

*# x contains the words of an article.*

*#xtr is the training x set which contains the frequency of the words in the form of sparse matrix.*

encoder = LabelEncoder()

encoder.fit(list(set(category\_tr)))

*# label encoder converts the category in the form of a matrix.*

*# for example bussiness is encoded as [1,0,0,0,0] if 5 categories are present.*

Ytr = encoder.transform(category\_tr)

Yde = encoder.transform(category\_de)

Yte = encoder.transform(category\_te)

*# y are the target variables.*

C:\Users\abc\anaconda3\lib\site-packages\sklearn\feature\_extraction\text.py:516: UserWarning: The parameter 'token\_pattern' will not be used since 'tokenizer' is not None'

warnings.warn(

Lets look at what exactly is this vectorizer doing. We will first create reverse dictionary from the vectorizer. Iterating over the vectorized sentence Nasa scientists are good. We get the vector to be representative of three words "good", "nasa" and "scientists". The order has been changed because bag of words does not preserve order.

reverse\_vocabulary = {}

vocabulary = vectorizer.vocabulary\_

**for** word **in** vocabulary:

index = vocabulary[word]

reverse\_vocabulary[index] = word

vector = vectorizer.transform(iter(['Nasa scientists are good']))

indexes = vector.indices

**for** i **in** indexes:

print (reverse\_vocabulary[i])

good

nasa

scientists

## Feature Reduction

We can check the variance of the feature and drop them based on a threshold

print("Number of features before reduction : ", Xtr.shape[1])

selection = VarianceThreshold(threshold=0.001)

Xtr\_whole = copy.deepcopy(Xtr)

Ytr\_whole = copy.deepcopy(Ytr)

selection.fit(Xtr)

Xtr = selection.transform(Xtr)

Xde = selection.transform(Xde)

Xte = selection.transform(Xte)

print("Number of features after reduction : ", Xtr.shape[1])

*# this is to remove unwanted features which do not help in the improvemnt of accuracy.*

Number of features before reduction : 4337

Number of features after reduction : 1833

## Sampling data

We will count the number of diffrent labels in dataset and plot a pie chart distribution.

labels = list(set(Ytr))

counts = []

**for** label **in** labels:

counts.append(np.count\_nonzero(Ytr == label))

plt.pie(counts, labels=labels, autopct='**%1.1f%%**')

plt.show()

*# it shows how various categories of data are there.*

As we can see the class labels are not uniformy distributed so we will use SMOT and oversample the classes which are less in number so that classes are equally distributed

sm = SMOTE(random\_state=42)

Xtr, Ytr = sm.fit\_resample(Xtr, Ytr)

*# this method will uniformly distribute various categories.*

# Train Models

### Baseline Model

“stratified”: generates predictions by respecting the training set’s class distribution.

dc = DummyClassifier(strategy="stratified")

dc.fit(Xtr, Ytr)

pred = dc.predict(Xde)

print(classification\_report(Yde, pred, target\_names=encoder.classes\_))

*#Recall is the Ratio of the correct predictions and the total number of correct items in the set.*

*#Precision is measured over the total predictions of the model.*

*# F1 Score is the 2\*((precision\*recall)/(precision+recall)).*

*#It is also called the F Score or the F Measure.*

*#F1 score conveys the balance between the precision and the recall.*

precision recall f1-score support

Business & Finance 0.13 0.22 0.17 74

Criminal Justice 0.17 0.27 0.21 78

Health Care 0.12 0.23 0.16 60

Politics & Policy 0.45 0.20 0.27 259

Science & Health 0.21 0.21 0.21 123

accuracy 0.22 594

macro avg 0.22 0.23 0.20 594

weighted avg 0.29 0.22 0.23 594

### Decision Tree

dt = DecisionTreeClassifier()

dt.fit(Xtr, Ytr)

pred = dt.predict(Xde)

print(classification\_report(Yde, pred, target\_names=encoder.classes\_))

precision recall f1-score support

Business & Finance 0.28 0.46 0.35 74

Criminal Justice 0.42 0.45 0.43 78

Health Care 0.59 0.70 0.64 60

Politics & Policy 0.66 0.53 0.58 259

Science & Health 0.54 0.49 0.51 123

accuracy 0.52 594

macro avg 0.50 0.52 0.50 594

weighted avg 0.55 0.52 0.53 594

### Random Forest

rf = RandomForestClassifier(n\_estimators=40)

rf.fit(Xtr, Ytr)

pred = rf.predict(Xde)

print(classification\_report(Yde, pred, target\_names=encoder.classes\_))

precision recall f1-score support

Business & Finance 0.33 0.42 0.37 74

Criminal Justice 0.48 0.54 0.51 78

Health Care 0.58 0.73 0.65 60

Politics & Policy 0.73 0.59 0.65 259

Science & Health 0.54 0.55 0.55 123

accuracy 0.57 594

macro avg 0.53 0.57 0.54 594

weighted avg 0.59 0.57 0.58 594

### Multinomial Naive Bayesian

nb = MultinomialNB()

nb.fit(Xtr, Ytr)

pred = nb.predict(Xde)

print(classification\_report(Yde, pred, target\_names=encoder.classes\_))

precision recall f1-score support

Business & Finance 0.41 0.42 0.42 74

Criminal Justice 0.59 0.60 0.59 78

Health Care 0.55 0.78 0.65 60

Politics & Policy 0.73 0.64 0.68 259

Science & Health 0.69 0.70 0.70 123

accuracy 0.64 594

macro avg 0.59 0.63 0.61 594

weighted avg 0.64 0.64 0.64 594

### Support Vector Classification

**from** **sklearn.svm** **import** SVC

svc = SVC()

svc.fit(Xtr, Ytr)

pred = svc.predict(Xde)

print(classification\_report(Yde, pred, target\_names=encoder.classes\_))

precision recall f1-score support

Business & Finance 0.32 0.24 0.28 74

Criminal Justice 0.77 0.22 0.34 78

Health Care 0.79 0.38 0.52 60

Politics & Policy 0.54 0.88 0.67 259

Science & Health 0.73 0.40 0.52 123

accuracy 0.56 594

macro avg 0.63 0.42 0.46 594

weighted avg 0.61 0.56 0.53 594

### Multilayered Perceptron

mlp = MLPClassifier(solver='adam', alpha=1e-5, hidden\_layer\_sizes=(100, 20), random\_state=1, max\_iter=400)

mlp.fit(Xtr, Ytr)

pred = mlp.predict(Xde)

print(classification\_report(Yde, pred, target\_names=encoder.classes\_))

precision recall f1-score support

Business & Finance 0.34 0.38 0.36 74

Criminal Justice 0.46 0.51 0.48 78

Health Care 0.56 0.68 0.62 60

Politics & Policy 0.68 0.62 0.65 259

Science & Health 0.67 0.63 0.65 123

accuracy 0.58 594

macro avg 0.54 0.56 0.55 594

weighted avg 0.59 0.58 0.59 594

# Final Model: Multinomial Naive Bayesian

**Multinomial Naive Bayesian** works the best. Lets run NB on our test data and get the confusion matrix

## Predict test data

pred = nb.predict(Xte)

print(pred)

print(classification\_report(Yte, pred, target\_names=encoder.classes\_))

*#sns.heatmap(confusion\_matrix(Yte, pred))*

[2 0 4 2 3 3 2 2 4 2 0 4 2 3 4 4 2 3 3 2 1 3 3 1 1 2 3 3 3 4 3 2 4 2 2 2 2

0 0 3 3 3 2 4 0 3 0 3 2 0 3 4 2 2 1 1 4 0 0 0 3 3 3 3 4 3 3 3 3 4 4 2 3 2

3 1 3 3 4 0 0 4 0 3 2 2 4 0 3 1 4 3 4 3 1 2 3 4 3 0 2 3 4 4 0 0 0 0 0 2 2

0 2 3 3 0 4 0 3 2 1 1 0 3 4 3 4 4 1 1 0 4 4 1 3 2 1 2 4 3 4 1 3 4 2 3 2 2

2 3 1 3 1 2 1 4 0 2 4 1 0 4 4 3 0 3 0 4 0 4 2 0 3 0 1 1 0 1 3 0 3 2 3 0 3

3 2 3 0 3 4 1 0 3 4 3 3 4 4 3 0 1 3 1 3 0 4 1 3 3 1 3 0 3 3 3 3 4 2 3 3 1

2 3 1 4 0 0 0 3 4 0 4 3 2 3 3 3 4 3 4 3 3 3 4 0 3 4 4 3 3 3 0 1 3 4 2 3 3

1 4 3 4 3 3 3 4 1 4 2 4 4 3 0 3 0 0 1 4 3 4 3 4 3 1 3 3 3 1 1 2 0 2 3 0 3

4 4 4 3 4 3 0 1 2 1 2 4 1 4 3 2 0 3 1 4 3 1 3 3 4 4 2 0 2 4 3 1 3 3 1 1 3

3 2 0 3 3 0 0 0 4 2 3 4 4 3 3 2 0 3 2 3 0 2 4 4 3 0 3 2 0 4 2 1 3 3 4 0 2

3 4 0 1 0 2 0 1 1 3 1 3 0 3 0 4 4 3 3 1 2 4 2 3 3 3 3 4 1 3 3 0 2 4 2 3 2

1 3 0 3 3 3 3 2 3 2 2 0 1 3 4 3 0 3 2 2 4 3 2 2 1 3 2 1 3 2 2 3 2 3 4 1 1

0 3 1 2 3 3 3 2 3 3 2 1 3 4 3 3 3 3 0 0 3 0 0 4 3 2 4 4 4 3 0 3 4 1 0 0 3

3 3 2 3 0 3 2 3 1 0 3 3 4 3 3 1 3 2 0 2 3 0 3 3 2 3 2 3 3 4 2 0 3 3 4 4 0

1 3 0 3 4 3 3 3 4 4 2 3 3 3 3 3 1 1 3 4 4 2 0 4 4 3 4 0 0 3 3 3 1 0 0 1 3

0 1 3 1 3 4 1 3 4 1 4 4 3 2 4 3 3 2 4 3 3 2 1 0 3 3 3 3 2 1 3 4 0 3 3 1 3

3 4 3 2 4 3 4 3 3 4 1 1 3 1 3 0 2 4 1 3 2 1 1 3 4 1 3 3 3 1 3 1 2 0 3 4 0

3 1 3 1 3 2 1 1 0 3 2 4 0 3 3 3 3 0 0 3 3 0 1 3 3 4 3 3 4 2 4 0 3 3 0 3 3

3 1 3 3 2 3 1 1 4 3 4 1 0 0 3 0 1 3 2 4 3 4 2 1 3 2 2 1 3 3 2 3 3 3 0 4 3

3 3 4 3 3 4 3 1 3 3 3 3 1 4 1 4 3 0 3 1 2 3 2 4 1 4 1 0 1 4 4 4 1 3 1 0 3

3 4 1 4 3 3 3 3 3 4 1 1 1 0 4 3 4 1 4 0 0 3 4 3 2 3 3 0 3 3 3 2 3 1 3 1 1

1 3 1 2 3 3 0 1 3 2 2 3 2 4 2]

precision recall f1-score support

Business & Finance 0.55 0.55 0.55 116

Criminal Justice 0.58 0.64 0.61 104

Health Care 0.45 0.68 0.54 77

Politics & Policy 0.73 0.64 0.68 342

Science & Health 0.65 0.61 0.63 153

accuracy 0.63 792

macro avg 0.59 0.62 0.60 792

weighted avg 0.64 0.63 0.63 792

**11.1.2 Setup to execute the code:**

The development work involves Python programming language which we have done in jupyter note book with some libraries like numpy, scikit-learn, imblearn, nltk and the Windows OS 10 with related software.