**PROJECT REPORT** Domain Classification

Team Members:

Burhanuddin Plumber

Dhruv Mistry

Hetul Mehta

Saniya Jain

Siddhi Mule

Urvik Gala

1

**TABLE OF CONTENT**

**1. INTRODUCTION 2** 1.1 ROLES AND RESPONSIBILITIES 2 1.2 TIMELINE 3 1.3 CHALLENGES FACED 3

**2. LITERATURE REVIEW 4** 2.1 MACHINE LEARNING ALGORITHM USED 4 2.2 TECHNOLOGY USED 7

**3. METHODOLOGY 8** 3.1 DATASET CREATION 8 3.2 DATA PREPROCESSING AND EXPLORATION 9 3.3 MODEL SELECTION 10 3.4 TRAINING THE MODEL 12

**4. RESULTS 12** 4.1 CLASSIFICATION REPORT 12 4.1 CONFUSION MATRIX 13 4.2 DEPLOYMENT 14

**5. CONCLUSION 18**

2

**1. INTRODUCTION**

In this project, we were interested to see what domain a website belongs to.

To address this question we look at a dataset of different websites that includes information about the website such as the URL, the name of the website, parsed website text and the top five categories along with the probabilities to which the website belongs to.

Our analysis depicts how these variables interact with the top category, and essentially predict the category for new websites. Using models on the training data we are going to predict the dependent variable on the test data.

**1.1 ROLES AND RESPONSIBILITIES**

The project was divided into two parts. The first part was Dataset creation and the second part was Model Training.

● DATASET CREATION:

To find related keywords, all the team members took 2-3 categories from the list of categories decided. Each team member scraped around 100 websites of each category.

● MODEL TRAINING:

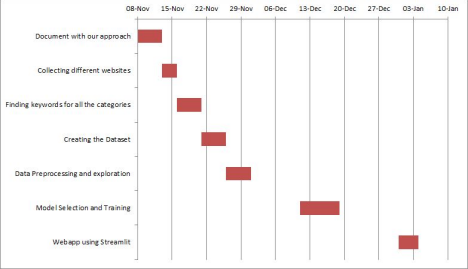
For this part of the project, we decided to divide ourselves into two groups, each consisting of three team members.

One group did the Data pre-processing, exploration and cross validation of different models.

The other group did the training of the model and created the webapp .

3

**1.2 TIMELINE**

**Fig.1. Timeline for the project.**

**1.3 CHALLENGES FACED**

Finding a dataset on the internet that we needed for this project was the biggest challenge that we came across. To overcome this, we created a dataset with the needed information.

Selection of the best fitting machine learning algorithm was another obstacle we faced. Doing extensive research helped us in selecting the most suitable machine learning model.

4

**2. LITERATURE REVIEW**

Classification of the Web pages have been considered largely since the Internet has become a huge source of information. Classification being considered a supervised learning problem in which a set of labelled data is used to train a classifier which can be applied to label upcoming instances. Before starting to work on the project a thorough research was done by us. We first researched on how to go about the project, referred some research papers to understand what domain classification actually is. We researched about the popular categories in which websites are classified. We then searched about different models such as data scraping, classification based on keywords for creating a dataset and applying neural networks on the model. We also read about some common techniques for text classification which are naive bayes classifier, linear classifier , support vector machine , bagging models , boosting models and deep neural networks and watched tutorials about these. For selecting the model, we had thought of a few approaches like using Latent Dirichlet allocation (LDA) and classifying the cleaned website text. Another approach was that we could use TFIDF for cleaned website text and tokenize the matched categories and then apply classification models such as Naives Bayes or GaussianNB.

**2.1 MACHINE LEARNING ALGORITHM USED**

The following machine learning algorithms were used in the project and the model with the highest cross validation score was selected to fit the dataset.

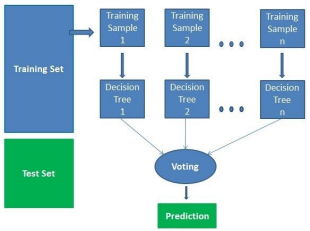
● RANDOM FOREST CLASSIFIER

Random forest is a supervised learning algorithm. The "forest", it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

The basic flow of the Random Forest Classifier and how the decision making is done .

5

**Fig.2. Basic flow of Random Forest Classifier .**

Random Forest is intrinsically suited for multiclass problems and works well with categorical features and hence was used in this project.

● MULTINOMIAL NAIVE BAYES

Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features.

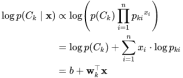
The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

This is the basic idea about how the Multinational NB works



6

The multinomial naïve Bayes classifier becomes a linear classifier when expressed in log-space



● GAUSSIAN NAIVE BAYES

A Gaussian Naive Bayes algorithm is a special type of NB algorithm. It’s specifically used when the features have continuous values. It’s also assumed that all the features are following a gaussian distribution i.e, normal distribution.

We have used the TF-IDF transform to encode the text into continuous-valued features and hence the Gaussian Naive Bayes algorithm works well with our dataset.

This is the basic math behind the Gaussian NB algorithm



● LINEAR SUPPORT VECTOR CLASSIFICATION

The most applicable machine learning algorithm for our project is Linear SVC. It is similar to SVC with parameter kernel=’linear’, but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

● CALIBRATED CLASSIFIER CV

A classifier can be calibrated in scikit-learn using the CalibratedClassifierCV class. There are two ways to use this class: prefit and cross-validation.

7

You can fit a model on a training dataset and calibrate this prefit model using a hold out validation dataset.

Alternatively, the CalibratedClassifierCV can fit multiple copies of the model using k-fold cross-validation and calibrate the probabilities predicted by these models using the hold out set. Predictions are made using each of the trained models.

Hence we used the CalibratedClassifierCV to calibrate the fitted and linearSVC model in the project to give the probability predictions for all classes.

**2.2 TECHNOLOGY USED**

Programming language : PYTHON

Webapp framework: streamlit

Python Libraries and packages used:

● Requests

● BeautifulSoup

● Flashtext

● Urlparse

● Spacy

● Numpy

● Pandas

● Seaborn

● Matplotlib

● Scikit Learn

8

**3. METHODOLOGY**

**3.1 DATASET CREATION**

We collected a list of different websites for creating the dataset. A scraper tool was used to parse and scrape each webpage using BeautifulSoup package and the text contents of each website such as the text content of <title> tag from a webpage, the text content of <meta> tags related to keywords and description from a webpage, the text content of heading tags as they might contain relatively important text and text content of the whole page with exception to some tags were collected.

The collected texts from the webpages were cleaned using the spacy library where the pronouns were removed and stopwords were filtered out. The words were lemmatized and lowercased.

For any machine learning algorithm, we need some training set and test set for training the model and testing the accuracy of that model. Hence to create the set of data for the model, we already had the text from different websites, which we just classified according to the keywords, and then applied the results to train the model and test it.

We classified the websites into 16 categories namely:

1. Travel

2. Streaming Services

3. Sports

4. Social Networking and Messaging

5. Photography

6. News

7. Law and Government

8. Health and Fitness

9. Games

10. Forums

11. Food

12. Education

9

13. E-Commerce

14. Computers and Technology

15. Business/Corporate

16. Adult

The approach here is that we will have certain keywords belonging to the particular category, and we will match those keywords with the text and find the class with the maximum *Matching\_value.*

Matching\_value *= (Number of keywords matched with one category)/(Total number of keywords matched)*

We used KeywordProcessor from flashtext package on pypi to find keywords inside the cleaned text received from the URLs.

We then loaded KeywordProcessor objects with the keywords which we further used for finding the matching keywords. We defined a function percentage to find the percentage value of Matching\_value.

We used the extract\_keywords(string) method to find keywords present in the text. And then found the length of that list to find the number of matching keywords in the text and the category with maximum percentage is selected.

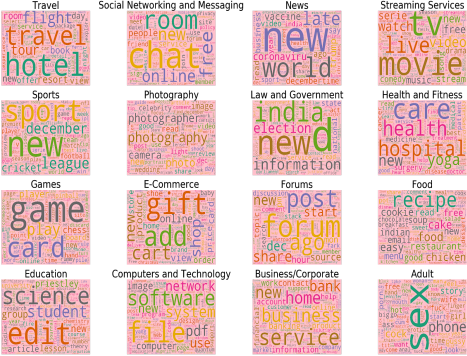
We then found the category of all websites based on the keywords and then saved the classified data into a file *FINAL\_DATASET.csv*. The dataset has 1408 rows and 16 columns.

**3.2 DATA PREPROCESSING AND EXPLORATION**

The dataset contains features that are not necessary to solve our multiclass-classification problem. For this text classification problem, we used the features ‘website\_url’, 'cleaned\_website\_text' and 'Category’. Then we represented each category as a number and stored it in a new column ‘category\_id’, so our predictive model can better understand the different categories.

To better understand the feature ‘cleaned\_website\_text’, wordclouds for each category were created as shown:

10

**Fig.3. WordCloud for each Category.**

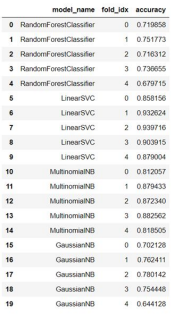
We then used TfidfVectorizer to transform each cleaned\_text into a vector. Each of the 1408 text is represented by 18865 features (TF-IDF score of unigrams and bigrams)

We then split the data into train and test sets, the original data was divided into features (X) which is the TF-IDF vector of cleaned\_website\_text and target (y) which is the category\_id, which were then split into the train (75%) and test (25%) sets. Thus, the algorithms would be trained on one set of data and tested out on a completely different set of data (not seen before by the algorithm)

**3.3 MODEL SELECTION**

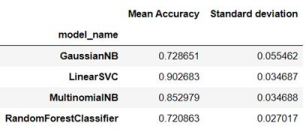
We used the cross validation method to check the accuracies of different models like RandomForestClassifier, LinearSVC, MultinomialNB and GaussianNB

11



**Table.1. 5-fold cross validation score.**

The mean accuracy and standard deviation of each model is :



**Table.2. Mean cross validation score for all the models.**

12

**3.4 TRAINING THE MODEL**

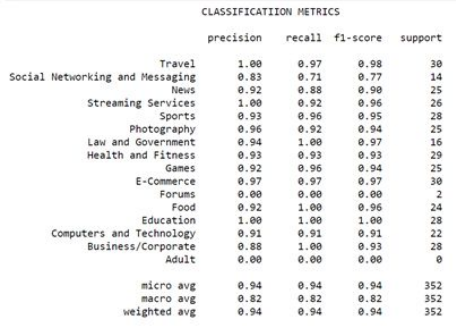
As we can see LinearSVC has the highest cross validation score so we use it to fit and test our dataset. We used CalibratedClassifierCV for prediction along with the probability of each category.

**4. RESULTS**

The accuracy of our model, LinearSVC was found to be 0.9403409090909091. **4.1 CLASSIFICATION REPORT**

The classification report displays the precision, recall, F1, and support scores for the model where precision can be seen as a measure of a classifier’s exactness, recall is a measure of the classifier’s completeness, the F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0 and support is the number of actual occurrences of the class in the dataset as shown

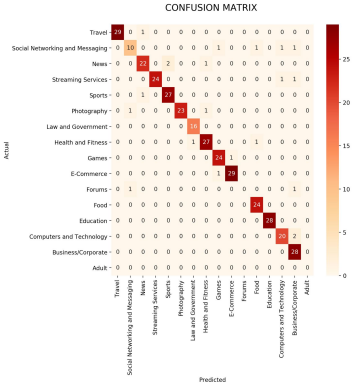
13

**Table.3. Classification Report.**

**4.1 CONFUSION MATRIX**

The confusion matrix is a summarized table of the number of correct and incorrect predictions (or actual and predicted values) yielded by the classifier as shown:

14

**Fig.4. Confusion Matrix.**

We can see that a few categories are misclassified such as:

'Photography' predicted as 'Social Networking and Messaging' : 1 examples. 'Forums' predicted as 'Social Networking and Messaging': 1 examples.

'Travel' predicted as 'News' : 1 examples.

'Sports' predicted as 'News' : 1 examples.

'News' predicted as 'Sports' : 2 examples.

'Health and Fitness' predicted as 'Law and Government' : 1 examples.

'News' predicted as 'Health and Fitness' : 1 examples.

'Photography' predicted as 'Health and Fitness' : 1 examples.

'Social Networking and Messaging' predicted as 'Games' : 1 examples.

'E-Commerce' predicted as 'Games' : 1 examples.

'Games' predicted as 'E-Commerce' : 1 examples.

'Social Networking and Messaging' predicted as 'Food' : 1 examples.

15

'Health and Fitness' predicted as 'Food' : 1 examples.

'Social Networking and Messaging' predicted as 'Computers and Technology' : 1 examples. 'Streaming Services' predicted as 'Computers and Technology' : 1 examples. 'Social Networking and Messaging' predicted as 'Business/Corporate' : 1 examples. 'Streaming Services' predicted as 'Business/Corporate' : 1 examples.

'Forums' predicted as 'Business/Corporate' : 1 examples.

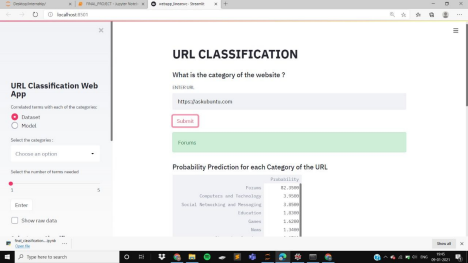
'Computers and Technology' predicted as 'Business/Corporate' : 2 examples.

**4.2 DEPLOYMENT**

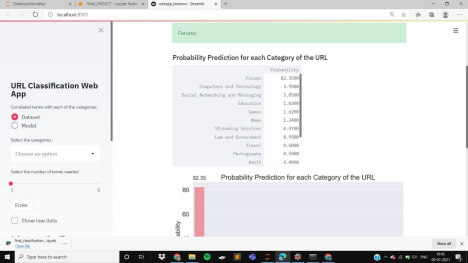
We created a web app using streamlit where the classification of websites was done. We can also train different models on the dataset and test it to show us the accuracy and different metrics such as classification report and confusion matrix. The webapp also has various other features like displaying the raw dataset and the correlated terms for each of the categories.

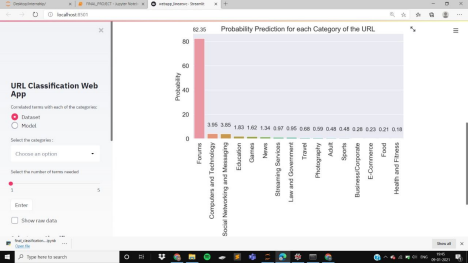
**Here are a few snippets of the webapp:**

The webapp classifies different URLs and shows the probability prediction of each category of the given URL.

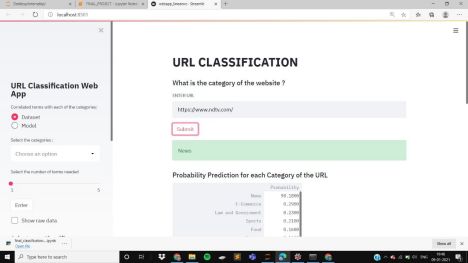
**Fig.5. Webapp predicting the category of URL https://askubuntu.com/.**

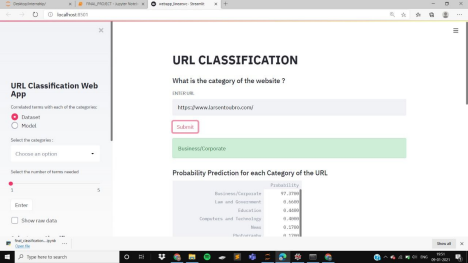
16

**Fig.6. Webapp displaying the probability prediction for the given URL https://askubuntu.com/.**

**Fig.7. Webapp displaying the graph of probability prediction for the given URL https://askubuntu.com/.**

17

**Fig.8. Webapp predicting the category of URL https://www.ndtv.com/.**

**Fig.9. Webapp predicting the category of URL https://askubuntu.com/.**

18

The most correlated terms with the selected categories in the dataset

**Fig.10. Webapp displaying the most correlated terms with the selected categories in the dataset**

We can train different models on the dataset

**Fig.11. Webapp displaying the accuracy and classification report for MultinomialNB.**

19

**5. CONCLUSION**

In this project we created a dataset of URLs, their scraped text and the categories they belonged to. We used machine learning models that best work with multiclass classification. Out of the four classification models cross validated, Linear Support Vector Classifier seemed to have a better accuracy and was therefore used to predict the category of the given URL. we created a webapp for better user interface, given a URL it will predict its category.