ML Project Report

Help Boost Our Online Reach!

Problem Statement:

Consider yourself to be a consultant for an online advertising agency. The agency spends a considerable amount of time and money to find the best web pages to publish their ads on. They select web pages that will generate prolonged online traffic so that their ads can have a long-lasting reach.

Now, wouldn't it be great if you could somehow automate this process and save company resources? To facilitate this, the agency has created a dataset of raw html, meta statistics and a binary label for each webpage. The binary label represents whether the webpage was selected for ad placement or not.

The aim of this task is to identify the relevant, high-quality web pages from a pool of user-curated web pages, for the identification of "ad-worthy" web pages. The challenge requires you to build large-scale, end-to-end machine learning models that can classify a website as either "relevant" or "irrelevant", based on attributes such as alchemy category and its score, meta-information of the web pages and a one-line description of the content of each webpage. This task aims to gently introduce you to the domain of NLP, as you would be required to convert the string attributes of the dataset to some form of numerical data, and then construct your ML models on this numerical data.

Can this fast-paced ad agency bank on you to deliver on this project?

EDA and Preprocessing:

- Checked for NULL and '?' values
- Removed framebased and domainLink columns as they contain 0's mostly.
- Checked the correlation between the columns.
- Performed outlier removal for numeric data.
- Performed one-hot encoding for "alchemy category" column.
- Replaced "webpageDescription" with 2-columns "body", "title".
- Modified "url" column by deleting symbols and strings which are not useful in predictions.
- Skew removal for numeric columns.
- HeatMap and

Feature Selection and Engineering:

- Natural Language Processing
    - Removing punctuations
    - Tokenization
    - Remove Stop words
    - Lemmatizing and Stemming
    - Vectorizing data using count vectorization and TFIDF
- Used PCA for dimensionality reduction.

Experiments conducted and challenges faced:

- We tried both stemming and lemmatization to bring words to their root forms and the results were better when we used lemmatization.
- We tried both 'bag of words', 'ngram' and 'tfidf' for vectorizing data and we got better results when we use 'tfidf'.
- We experimented with changing hyper parameters in models and tfidf (max_features). The combinations of hyperparameters used is shown below in the table.

Models used:

- Logistic regression
- Naïve bayes
- SVM
- ADABOOST
- Bagging
- XGBoost
- Decision Tree
- Neural networks

Tables of models and their scores:

1. Logistic Regression and Naïve Bayes

PreProcessing	Model	Hyper Parameters	Local score	Kaggle Score
Basic pre-processing mentioned before. and standard scalar	Logistic Regression	max_iter = 10000 max_features = 500	0.79845	0.80850
Same as above	Same as above	max_iter = 10000 max_features = 100	0.83163	0.81298
Same as above	Same as above	max_iter = 10000 max_features = 300	0.81461	0.79438
Removed Outliers	Logistic Regression	max_iter = 10000 max_features = 300	-	0.79556
Used tfidf instead of count vectorizer	Same as above	max_iter = 10000 max_features = 300	-	0.85477
Didn't use outlier removal	Same as above	max_iter = 10000 max_features = 300	-	0.85311
Same as above	Same as above	max_iter = 10000 max_features = 200	0.86190	0.84819
Used stemming instead of lemmatisation	Same as above	max_iter = 10000 Max_features = 100	0.8619	0.84894
Same as above	Naïve Bayes	max_iter = 10000 max_features = 500	0.74559	0.79592
Same as above	Same as above	max_iter = 10000 max_features = 300	0.71673	0.49887

2. SVM

PreProcessing	Model	Hyper Parameters	Local score	Kaggle Score
Pre-Processing	SVM	kernel = rbf class_weights = balanced Vectorizer features = 300	0.86837	0.86418
Same as above	Same as above	Same as above and C = 0.5	0.86411	0.86324
Same as above	Same as above	Same as above but C = 0.3	0.85489	0.86278
Same as above	Same as above	Same as above but C = 0.5 Vectorizing features = 500	-	0.86528
Same as above	Same as above	Same as above but Vectorizing features = 300	0.85948	0.86483
Pre-Processing	SVM + PCA	kernel = rbf class_weights = balanced Vectorizer features = 500 C = 0.5 n-components = 500	0.86417	0.86327
Same as above	Same as above	Same as above but Vectorizing features = 1000	0.87669	0.87331
Pre-Processing	SVM	kernel = rbf class_weights = balanced Vectorizer features = 300	0.86837	0.86418
Same as above	Same as above	Same as above and C = 0.5	0.86411	0.86324
Same as above	Same as above	Same as above but C = 0.3	0.85489	0.86278
Same as above	Same as above	Same as above but C = 0.5 Vectorizing features = 500	-	0.86528
Same as above	Same as above	Same as above but Vectorizing features = 300	0.85948	0.86483
Same as above	Same as above	Same as above but Vectorizing features = 3000	0.88020	0.92245
Same as above	Same as above	Same as above but Vectorizing using all features		0.86323

3. Neural Networks

PreProcessing	Model	Hyper Parameters	Local score	Kaggle Score
Pre-Processing	Multi Layer Perceptron	hidden_layer_sizes=(8,8,8,8,8,8,8), activation='identity', solver='adam', max_iter=5000, alpha=1e-5 Vectorizing features = 3000 Pca features = 300	0.86255	0.87544
Same as above	Same as above	Same as above but hidden_layer_sizes = (30,40,20,30)	-	0.86878
Same as above	Same as above	Same as above but Vectorizing All features PCA features = 300		0.86828

4. Decision Trees and Random Forests

PreProcessing	Model	Hyper Parameters	Local score	Kaggle Score
Pre-processing	AdaBoost	learning_rate=0.1, n_estimators=300, random_state=0 vectorizing features = 300	0.8600	0.88957
Same as above	Same as above	Same as above n_estimators=1000 vectorizing features = 300 pca features = 300	0.86290	0.91649
Same as above	Bagging	base_estimator=SVC, max_samples=30, n_estimators=1000, random_state=0 PCA and tfidf same as above	0.86280	0.87976
Same as above	Decision Tree	class_weight='balanced', random_state=0 PCA and tfidf same as above	0.7355	0.86475

Individual contributions:

Group Name – Phantom Troupe

- IMT2019039 – Kanigiri Naveen
    - NLP, Logistic Regression, SVM, Decision Trees, Bagging, Mixture of Models, Report Generation

- IMT2019041 – Kasturi Siva Hitesh
    - Basic Preprocessing and EDA, Neural Networks, XGBoost, Mixture of Models, Report Generation

- IMT2019045 – Kopparapu Sai Krishna
    - NLP, Naïve Bayes, Outliers, ADAboost, Mixture of Models, HTML Preprocessing, Report Generation

Conclusions:

1. We got a good understanding on –
    a. How to use NLP and different preprocessing techniques under NLP.
    b. How to play around with Hyper parameters to result in good model accuracy.
    c. Models and their working.

2. Top Best Models –

    a. SVM -
        i. Hyper Parameters – TFIDF features = 3000, PCA features = 300, C = 0.5
        ii. Pre-Processing – Null values, Outliers, NLP
        iii. Score = 0.92245

    b. ADABoost –
        i. Hyper Parameters - learning_rate=0.1, n_estimators=1000, random_state=0, TFIDF features = 3000, PCA features = 300
        ii. Pre-Processing – NULL values, Outliers, NLP
        iii. Score = 0.91649

References:

https://towardsdatascience.com/text-classification-in-python-dd95d264c802

https://towardsdatascience.com/natural-language-processing-nlp-for-machine-learning-d44498845d5b

https://towardsdatascience.com/website-data-cleaning-in-python-for-nlp-dda282a7a871

https://scikit-learn.org/stable/user_guide.html

Files Submitted:

a. PhantomTroupe_BestSubmission.ipynb – Best Submission
b. PhantomTroupe_Classification.ipynb – Pre-Processing and Models used
c. PhantomTroupe_HtmlProcessing.ipynb – Tried HTML Pre-Processing