Project Report on “Help boost our ONLINE REACH”

~By imt2019071(Rohan Thakkar) AND

imt2019053(Mayank Jain)

Project Definition

Problem Statement:

Given a dataset of raw html, meta statistics and a binary label for each web page . Use them to build a machine learning model that can classify a website as either “relevant” or “irrelevant” depending on whether the web pages could generate prolonged online traffic so that the ads can have a long lasting reach.

Aim:

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.

PRE-PROCESSING

Filling Missing Values

In the given Dataset we were provided with most of the data, the datas that we were missing were either given as question marks(‘?’) or as ‘NaN’.

We first converted all the entries that were given as question marks(‘?’) to ‘Nan’ .

After performing the above , we replaced the ‘NaN’ entries with their mode value of their respective column.

How did we convert ‘?’ to nan and dealing with missing data code

```
Changing '?' to nan
df_train = pd.read_csv(r"C:\Users\Rohan\OneDrive\Documents\ML\train_data.csv",
na_values = ['?'])
df_test = pd.read_csv(r"C:\Users\Rohan\OneDrive\Documents\ML\test_data.csv",
na_values = ['?'])
```

Checking for null values:

```
final_df.head()
final_df.isnull().sum()
```

Dealing with missing data code

```
final_df['alchemy_category_score'] =
final_df['alchemy_category_score'].fillna(final_df['alchemy_category_score'].mode()[0])
final_df_test['alchemy_category_score'] =
final_df_test['alchemy_category_score'].fillna(final_df_test['alchemy_category_score'].mode
()[0])
final_df['isNews'] = final_df['isNews'].fillna(final_df['isNews'].mode()[0])
final_df['isFrontPageNews'] =
final_df['isFrontPageNews'].fillna(final_df['isFrontPageNews'].mode()[0])
final_df_test['isNews'] = final_df_test['isNews'].fillna(final_df['isNews'].mode()[0])
final_df_test['isFrontPageNews'] =
final_df_test['isFrontPageNews'].fillna(final_df_test['isFrontPageNews'].mode()[0])
```

One-Hot Encoding:

In the column named “alchemy_category”, we applied ‘One-hot encoding’ method. By this the categorical data got converted into numerical data which can be used for further methods such as predicting the test data.

Code for One-Hot Encoding:

```
new_data=df_train
data=pd.get_dummies(new_data["alchemy_category"])
new_data1=df_test
data1=pd.get_dummies(new_data1["alchemy_category"])


frames=[df_train,data]
final_df = pd.concat([df_train, data], axis=1, join='inner')
frames=[df_test,data1]
final_df_test = pd.concat([df_test, data1], axis=1, join='inner')
```

Removing outliers

In supervised modelling, outliers are those datas that deceive the training process which results in a less precise model . We used interquartile (IQR) method to remove outliers in our given dataset.

Code for Removing outliers:

```
class OutlierRemoval:
    def __init__(self, lower_quartile, upper_quartile):
        self.lower_whisker = lower_quartile - 1.5*(upper_quartile - lower_quartile)
        self.upper_whisker = upper_quartile + 1.5*(upper_quartile - lower_quartile)
    def removeOutlier(self, x):
        return (x if x <= self.upper_whisker and x >= self.lower_whisker else (self.lower_whisker if x <
self.lower_whisker else (self.upper_whisker)))
```

Preprocessing and cleaning the test

Function(Clean Text):

Convert data to lower case

Remove Special Characters

Remove extra spaces

Remove Stopwords

Code for clean text

```
def clean_text(text):
    # convert to lowercase
    text = text.lower()
    # remove special characters
    text = re.sub(r'[^0-9a-zA-Z]', ' ', text)
    # remove extra spaces
    text = re.sub(r'\s+', ' ', text)
    # remove stopwords
    text = " ".join(word for word in text.split() if word not in STOPWORDS)
    return text
```

Lemmatization:

It aims to remove inflectional endings only and to return the base form of a word by reading and understanding the context.

Code for Lemmatization:

```
import nltk
nltk.download('wordnet')
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()
```

```
final_df['webpageDescription']=final_df['webpageDescription'].apply(lambda
x:' '.join(lemmatizer.lemmatize(w) for w in x.split()))
```

1) We checked for outliers in every columns and if outliers were found we removed it.

What are outliers...?

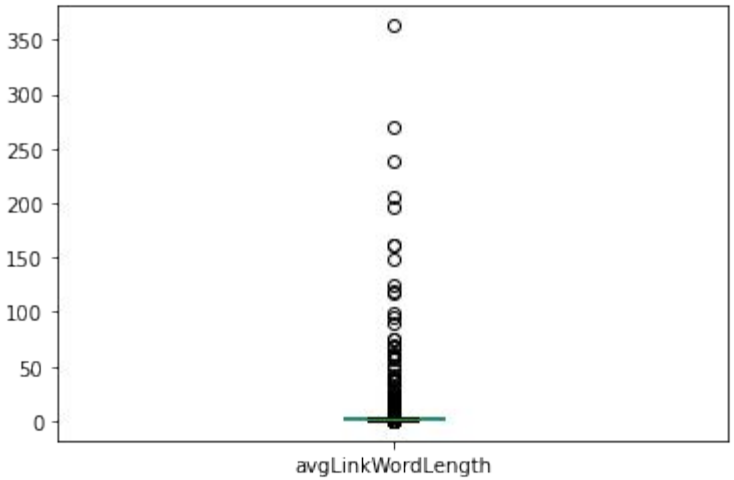
An *outlier* is an observation that lies an abnormal distance from other values in a random sample from a population. In a sense, this definition leaves it up to the analyst (or a consensus process) to decide what will be considered abnormal. Before abnormal observations can be singled out, it is necessary to characterize normal observations.

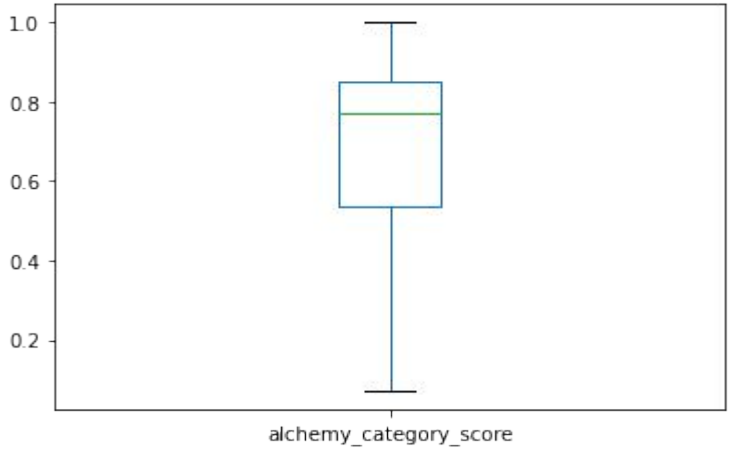
2)We removed the missing data by mode of the column,there were ? in the columns we changed it first to Nan then filled the missing data in the column

3)We did one hot encoding on our categorical column named **`alchemy_category`**

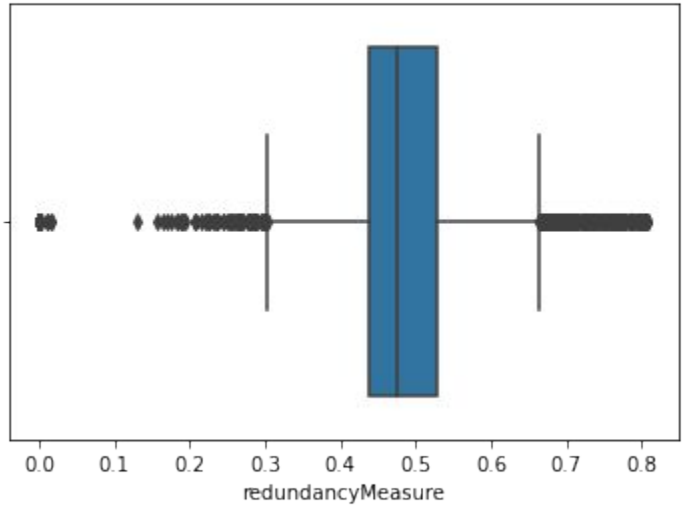
4)Applied Standard Scalar on numerical data

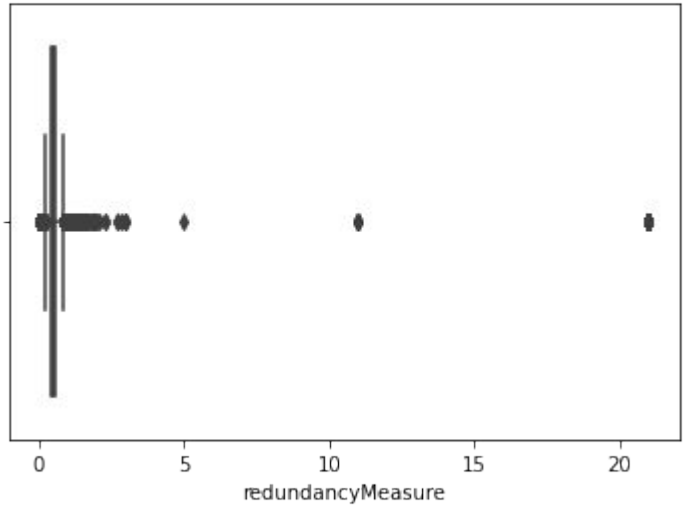
Some images of outliers present in data

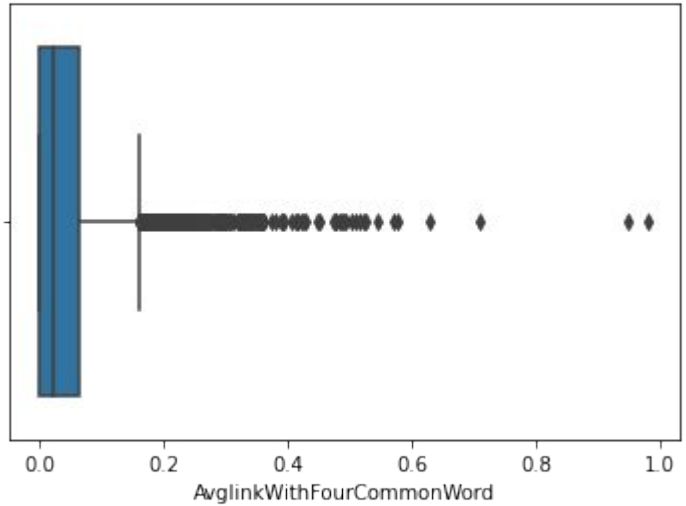


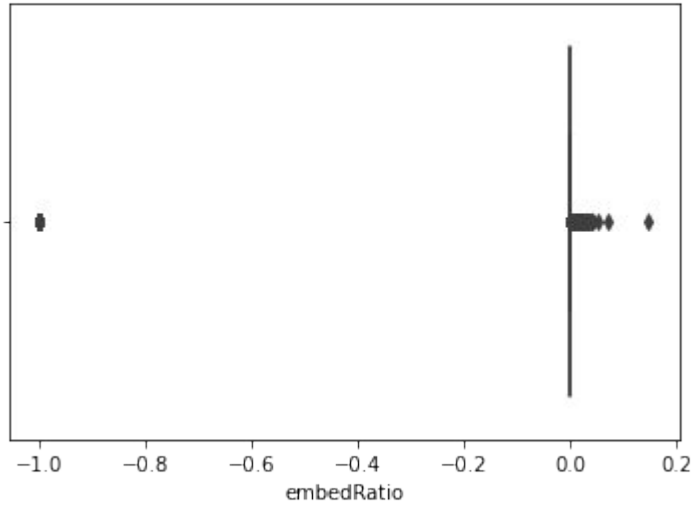


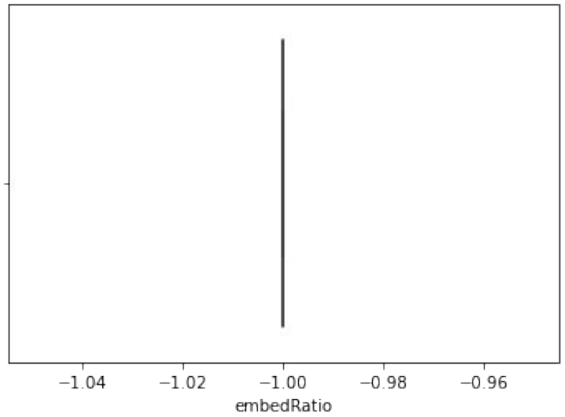
Some more plots

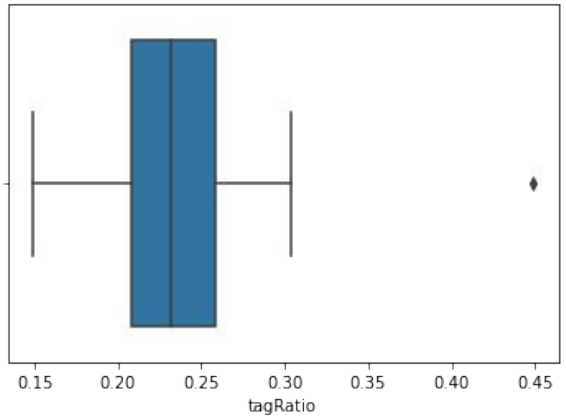


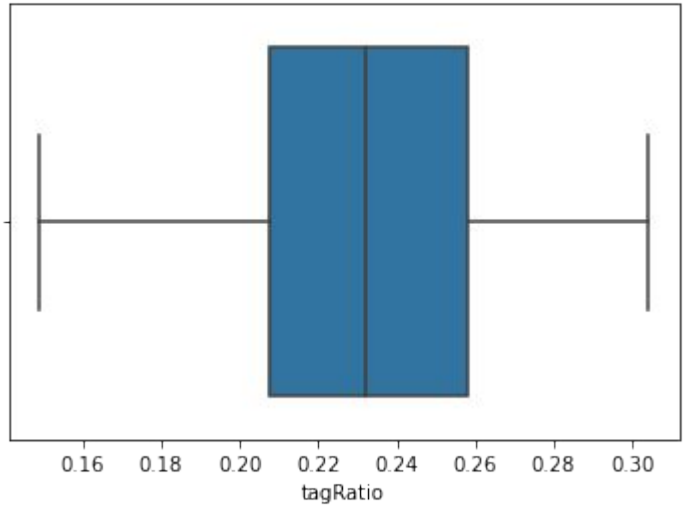


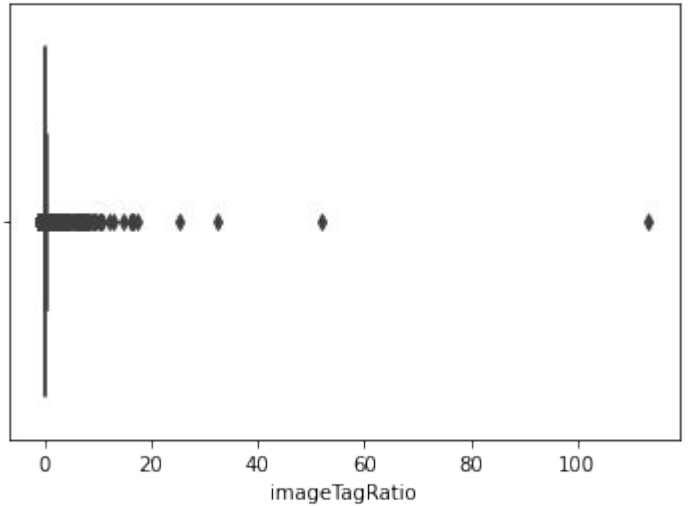

0.0
0.2
0.4
0.6
0.8
1.0
AvglinkWithFourCommonWord

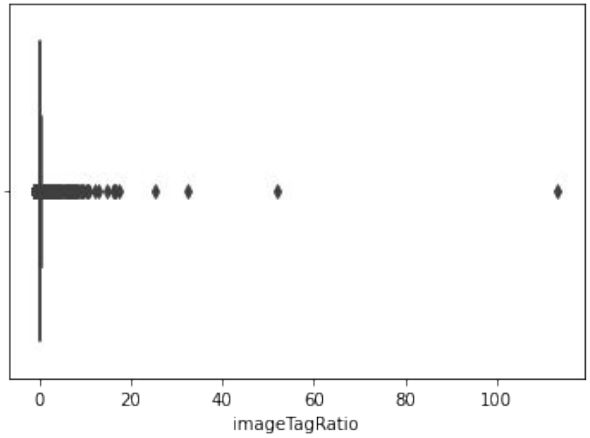

−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
embedRatio

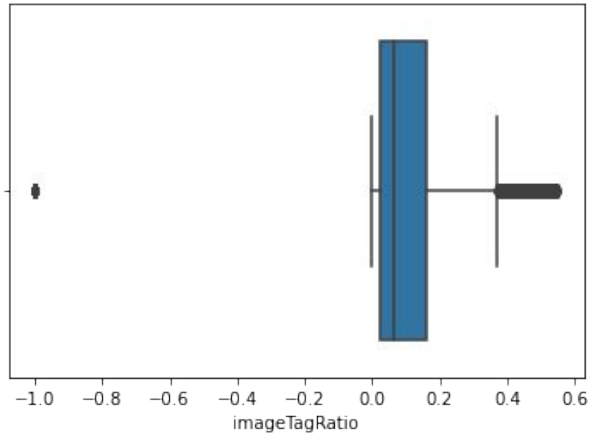

−1.04
−1.02
−1.00
−0.98
−0.96
embedRatio

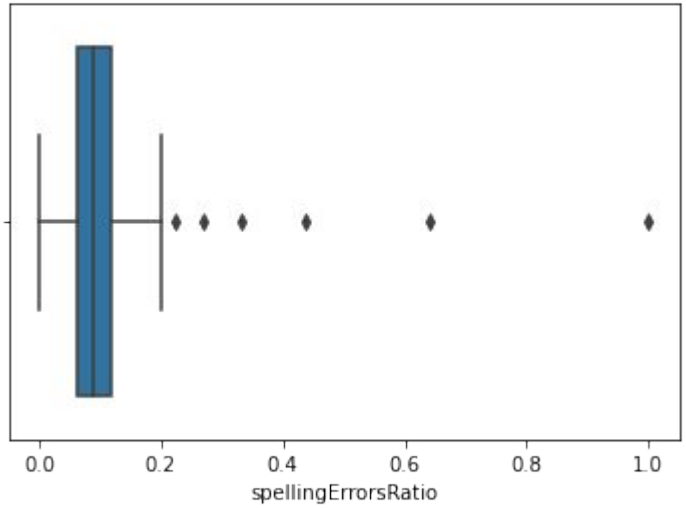

0.15
0.20
0.25
0.30
0.35
0.40
0.45
tagRatio

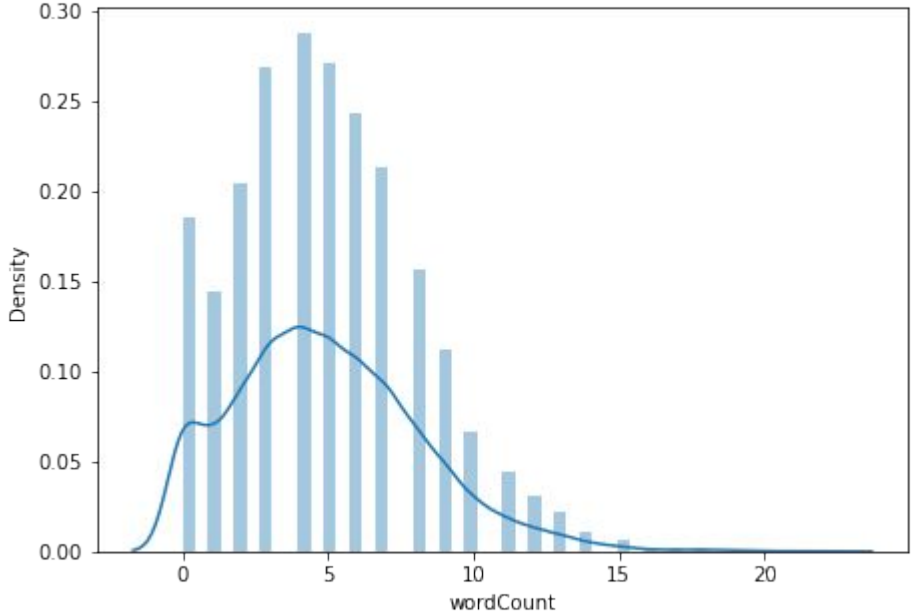

0.16
0.18
0.20
0.22
0.24
0.26
0.28
0.30
tagRatio


0
20
40
60
80
100
imageTagRatio


0
20
40
60
80
100
imageTagRatio


−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
imageTagRatio


0.0
0.2
0.4
0.6
0.8
1.0
spellingErrorsRatio


0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
0
5
10
15
20
wordCount

Function for removing outliers

```
for columnName in all_cols2:
    feature = final_df[columnName]
    feature_outlier_remover = OutlierRemoval(feature.quantile(0.25),
feature.quantile(0.75))
    outlier_removed_feature =
feature.apply(feature_outlier_remover.removeOutlier)
    final_df.assign(columnName=outlier_removed_feature)
```

Feature selection and Engineering

Vectorizer:

We used Tf-idf method for vectorization . We used it to transform text into a meaningful representation of numbers which we further used to fit machine algorithm for predictions.

Code:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()

X = vectorizer.fit_transform(final_df['webpageDescription']).toarray()
```

PCA

What is PCA..?

Principal component analysis (**PCA**) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

Code for PCA

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components = 1000)
```

```
X1 = pca.fit_transform(X1)
```

Standard Scalar

We used StandardScalar for standardizing a feature, that is , we subtract the feature by the mean and then divide all the values with standard deviation.

Code for standard scalar

```
std_scaler = StandardScaler()
```

```
df_scaled = std_scaler.fit_transform(X)
```

Experiments conducted and Challenges faced

We used all kinds of models such as logistic regression,Xgboost,SVM. We also did different kind of preprocessing steps so that our accuracy can be maximized ,We also applied PCA on our data and we also applied Standard scalar due to which our accuracy got boosted,We also tried applying Neural Networks but it was tough to be implemented and also we tried applying correlation matrix.Overall it was tough and challenging project but we got to learn alot about machine learning programming and especially NLP which will be useful for our studies in future.

Models Used

- Logistic Regression
- Random Forest
- Xgboost
  (code)

Code for different models

1)Logistic Regression:

```
my_model = LogisticRegression()
my_model.fit(X1, y)
```

2)Random Forest Regression

```
my_model = RandomForestRegression()
my_model.fit(X1, y)
```

3)Xgboost

```
my_model = Xgboost()
my_model.fit(X1, y)
```

Code for predicting accuracy

```
proba = my_model.predict_proba(X1).T[1]
```

```
roc_auc_score(y,proba)
```

Tables of Models

Model	Score(Accuracy)
Xgboost	0.88515
Random Forest	0.87993
Logistic Regression	0.88232

Individual Contribution

We worked as a team and collected every information available and expremented everything as a team there was no individual contribution but we worked as a team

Conclusion:

After implementing everything our accuracy score comes out to be 0.88515 on test data and also we learned about NLP in details and got to know how NLP code works and how to work with text data and how to handle the test data and convert it into something that can be used in predicting the model.

References

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

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