Aid Escalating Internet Coverage

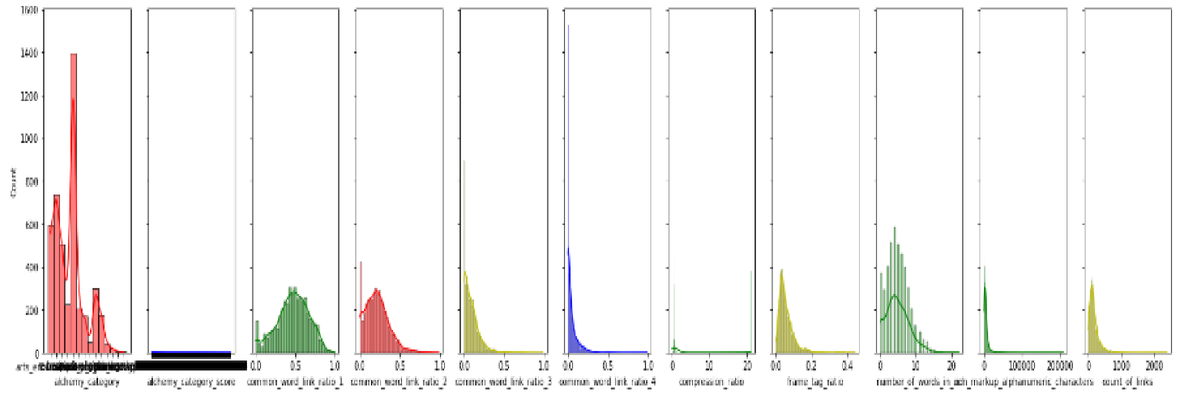
Group Name : Ragnarok

Anshul Sharma
MT2022143

Chinmay Kavimandan
MT2022031

Pioneering Excellence in Education, Research & Innovation

DATA ANALYSIS :



- We have given Train and Test data separate.
- In Train data we have 4437 rows and 24 columns.
- In Test data we have 2958 rows and 23 columns.
- We didn't have any Null value in data but many of rows has "?".
- Some of Column's maximum entries are Zero.
- In "page description" column we have text data.
- In "alchemy category" column we have categorical data.
- Some columns has very low correlation with Output "label".

SINCE 1998 SILVER JUBILEE YEAR
Pioneering Excellence in Education, Research & Innovation

Preprocessing :

- Drop some columns based on low correlation with output these columns are: “embed_ratio,frame_based,has_domain_link”
- Some columns have very large value so we have Normalize them with formula : Min-Max Scaling : (X – Min)/(Max-Min)
- Replace the “?” from alchemy_category column with “others”.
- Apply Label Encoding using category codes approach that assign each category a unique value.
- Replace “?” from alchemy_category_score with mean of that column.

Exploratory Data Analysis :

- Now we have to preprocess text based column that is : page_description.
- Using gensim.utils.simple_preprocess we remove the special character from the text and it converts the character into lower case, this function return sentence in form of list of words then we have to join them together in a sentence.
- Now convert the text data into numeric data with TFIDF (Term frequency Inverse Document Frequency).
- This return data into a Matrix form for some models we have to convert this into array form using toarray() function.

Intermediate Model for alchemy category

- Divide the dataframe in two parts based on value “?” of alchemy category
- Use the data which does not contain “?” for training of the model
- These columns are used for training of the model :

'common_word_link_ratio_1','common_word_link_ratio_2','common_word_link_ratio_3','common_word_link_ratio_4','link_word_score'

SINCE 1998 SILVER JUBILEE YEAR iiit-b ज्ञानमुत्तमम्
Pioneering Excellence in Education, Research & Innovation

Intermdiate Model for alchemy category score

- We used alchemy category values for predicting alchemy category score
- After predicting values for “?” we used one hot encoding to convert each individual alchemy category into column and used these columns for training of the model
- These columns are used for training of the model :

'arts_entertainment','recreation','sports','computer_internet','health','gaming','unknown'

Pioneering Excellence in Education, Research & Innovation

Features use for training the model

- We combine the dataset splitted earlier for predicting the values of alchemy category and score
- We trained the model by applying TF-IDF on page description column and then convert the sparse matrix into a data frame.
- These columns are inserted in the dataframe formed by TF IDF :
- 'common_word_link_ratio_1','common_word_link_ratio_2','common_word_link_ratio_3','common_word_link_ratio_4','link_word_score',’alchemy_category_score’

Pioneering Excellence in Education, Research & Innovation

Text-Preprocessiong

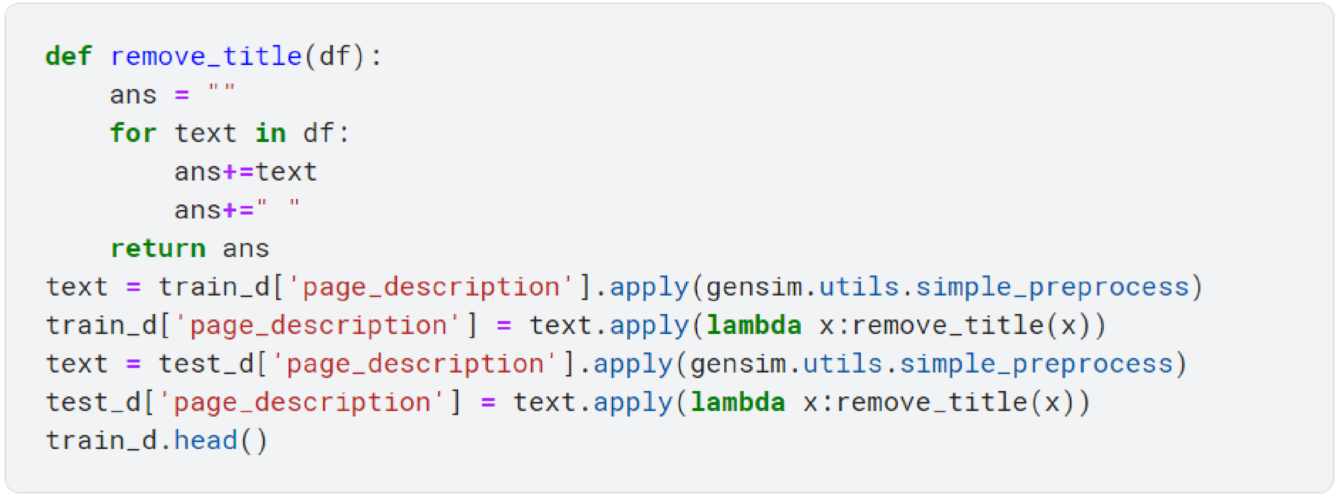
- Removing punctuations like . , ! $( ) * % @
- Removing Stop words
- Tokenization
- Lower casing
- Stemming
- Lemmatization

We use genism to remove special characters, lowering the words and tokenization

Use STOPWORDS for remove the common words.

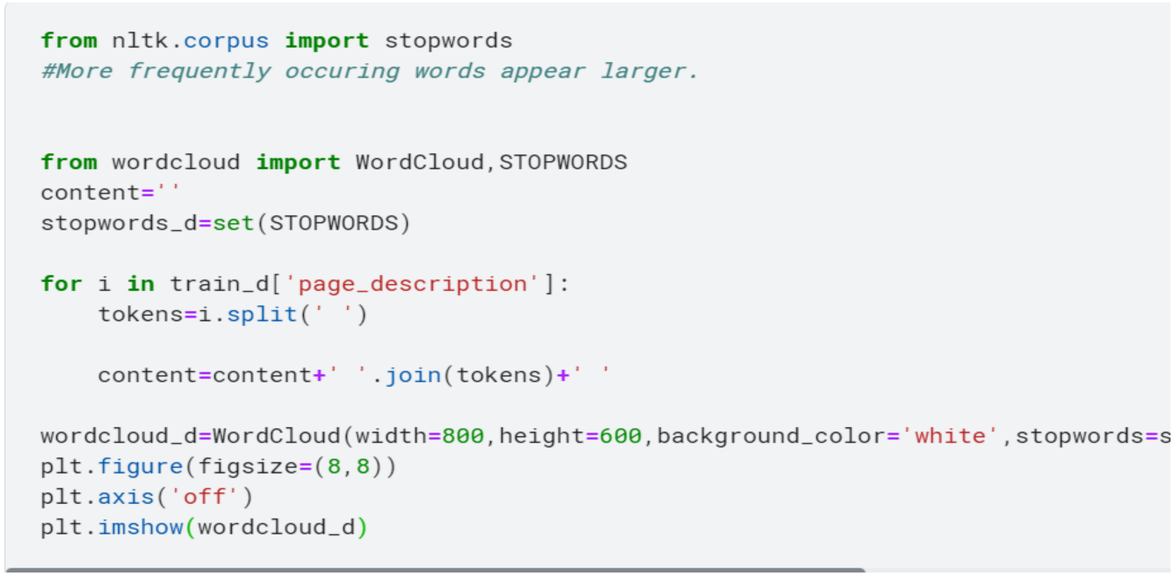
Functions use in code :

Function use to remove the special word, make token of sentences and lowering the words :

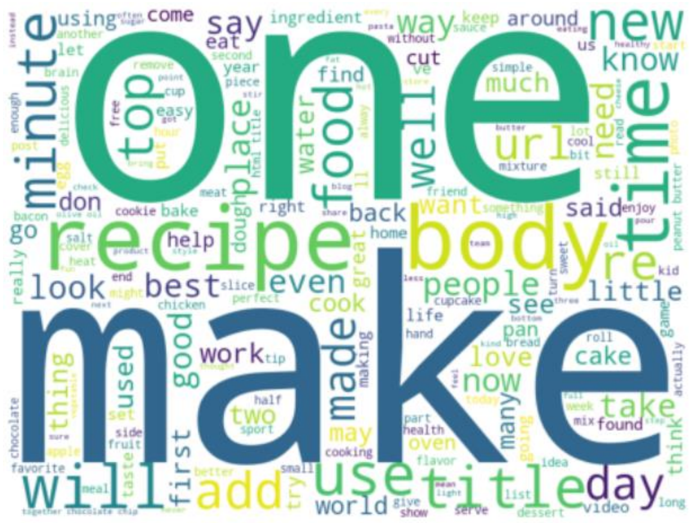


```python
def remove_title(df):
    ans = ""
    for text in df:
        ans+=text
        ans+=" "
    return ans
text = train_d['page_description'].apply(gensim.utils.simple_preprocess)
train_d['page_description'] = text.apply(lambda x:remove_title(x))
text = test_d['page_description'].apply(gensim.utils.simple_preprocess)
test_d['page_description'] = text.apply(lambda x:remove_title(x))
train_d.head()
```

Function use for print most frequent words:



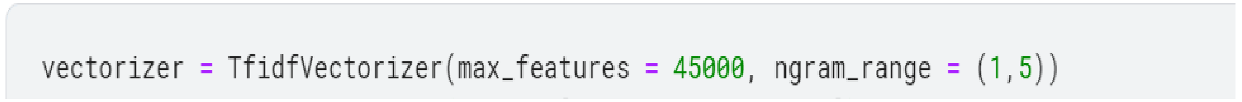
```python
from nltk.corpus import stopwords
#More frequently occuring words appear larger.


from wordcloud import WordCloud,STOPWORDS
content=''
stopwords_d=set(STOPWORDS)

for i in train_d['page_description']:
    tokens=i.split(' ')

    content=content+' '.join(tokens)+' '

wordcloud_d=WordCloud(width=800,height=600,background_color='white',stopwords=s
plt.figure(figsize=(8,8))
plt.axis('off')
plt.imshow(wordcloud_d)
```



Pioneering Excellence in Education, Research & Innovation

TF-IDF & Word2vec

- Parameters use in TF-IDF are max_features = 45000 & ngram = 1 to 5.



```
vectorizer = TfidfVectorizer(max_features = 45000, ngram_range = (1,5))
```

- We have used word2Vec feature vectors for training the RBF SVM, also used gridSearchCV to find out the optimal parameters for training SVM
- We use Google news data set to train Word2vec

```
w2v_model=KeyedVectors.load_word2vec_format
('GoogleNews-vectors-negative300.bin',
binary=True)
```

Model Used

- We use Logistic Regression, Svm and random forest
- When Logistic is used with TF-IDF the maximum accuracy we get is 0.8773.
- In svm we use kernel = 'rbf', and svm with TF-IDF give 0.8763
- In Random forest we use n_estimators = 100.
- In Decision tree we use random_state=42 and max_depth = 2

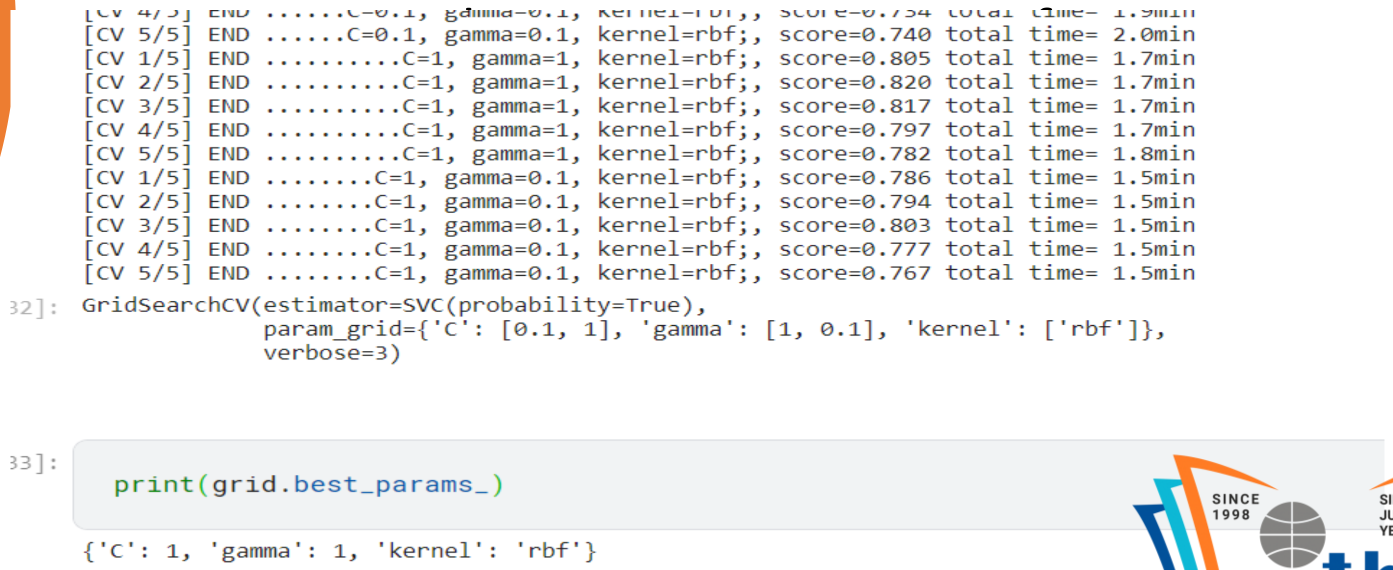
Pioneering Excellence in Education, Research & Innovation

Tables of Models and score :

Preprocessing	Model	Hyperparameters	Local score	Kaggle score
Database columns + TF-IDF features	Random Forest	n_estimator = 100	0.8890	0.8603
Database columns + TF-IDF features	Logistic Regression		0.8710	0.8777
TF-IDF features	Logistic Regression		0.8663	.8773
TF-IDF features	Random Forest Classifier	n_estimator = 100	0.8553	.8754
TF-IDF features	SVM	Kernel = 'rbf' and	0.8683	0.8773
TF-IDF features	Kneighbors Classifier	n_neighbors = 60,leaf_size = 5	0.85	0.81

Hyperparameter Tuning

- Use GridserchCV for Hyperparameter Tuning.
- Apply GridserchCV on SVM with C = “0.1”, “1”, ”10” and kernel =‘rbf’ , gamma = ‘1’ , ‘0.1’, 0.01’



```
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.754 total time= 1.9min
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.740 total time= 2.0min
[CV 1/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.805 total time= 1.7min
[CV 2/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.820 total time= 1.7min
[CV 3/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.817 total time= 1.7min
[CV 4/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.797 total time= 1.7min
[CV 5/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.782 total time= 1.8min
[CV 1/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.786 total time= 1.5min
[CV 2/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.794 total time= 1.5min
[CV 3/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.803 total time= 1.5min
[CV 4/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.777 total time= 1.5min
[CV 5/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.767 total time= 1.5min
```

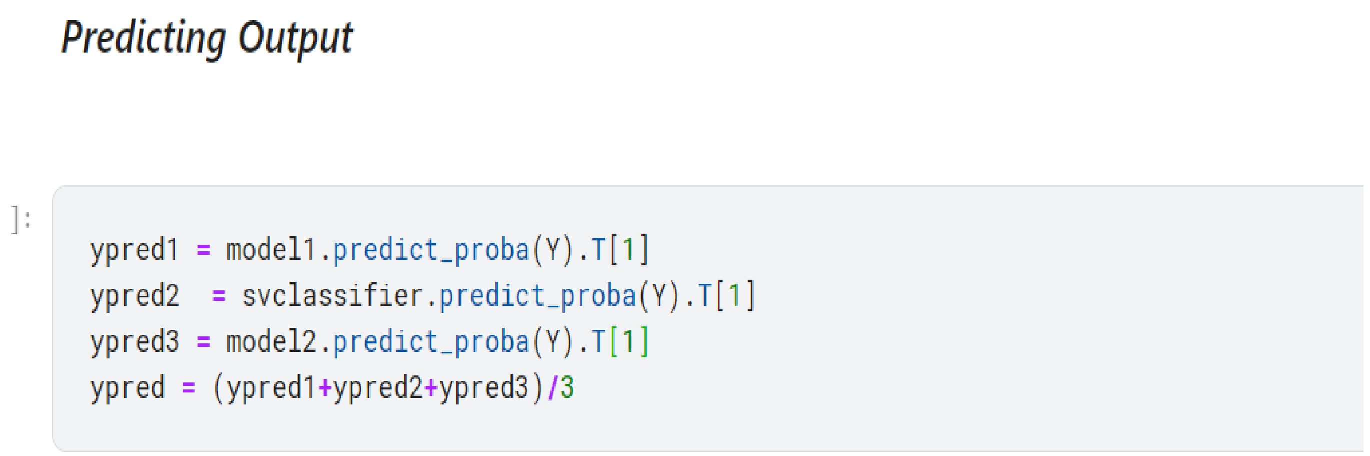
```
32]:  GridSearchCV(estimator=SVC(probability=True),
                   param_grid={'C': [0.1, 1], 'gamma': [1, 0.1], 'kernel': ['rbf']},
                   verbose=3)
```

```
33]:  print(grid.best_params_)

      {'C': 1, 'gamma': 1, 'kernel': 'rbf'}
```

SINCE 1998 SILVER JUBILEE YEAR iiit-b ज्ञानमुत्तमम्
Pioneering Excellence in Education, Research & Innovation

Slacking

- Slacking is done for optimize prediction in which Logistic Regression, Random forest and SVM were taken and the resultant submission score was 0.8783 for TF-IDF.



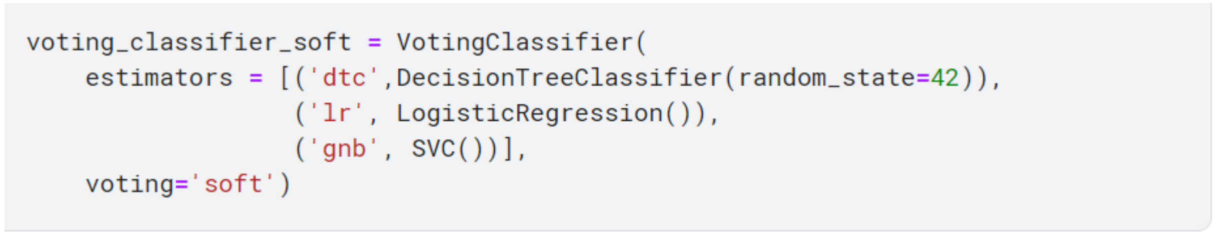
Predicting Output

```
]:
ypred1 = model1.predict_proba(Y).T[1]
ypred2  = svclassifier.predict_proba(Y).T[1]
ypred3 = model2.predict_proba(Y).T[1]
ypred = (ypred1+ypred2+ypred3)/3
```

Pioneering Excellence in Education, Research & Innovation

Voting

- Voting is one of the simplest ways of combining the predictions from multiple machine learning algorithms
- If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.
- We use Logistic Regression, Decision Tree and SVM in Voting



```
voting_classifier_soft = VotingClassifier(
    estimators = [('dtc',DecisionTreeClassifier(random_state=42)),
                  ('lr', LogisticRegression()),
                  ('gnb', SVC())],
    voting='soft')
```

Work done by both team mates :

Work done by Anshul Sharma:

- Text Processing using genism and stopwords.
- Apply various models to test various accuracy.
- Apply slacking with various models.
- Apply TF-IDF with different parameters.
- Apply voting to get best accuracy.

- Work done by Chinmay:
- Normalising columns link_word_score,non_markup_alphanumeric_characters,count_of_links and number_of_words_in_url
- Prediction of '?' in Alchemy Category and Alchemy Category score using intermediate model
- Implementing word2Vec to find feature vector for column page description
- Using GridSearchCV for finding optimal parameters for SVM model