Natural Language Processing (COMM061)

Part 1

GROUP 21

TEAM MEMBERS: KRISHNAKUMAR KANNAN, ASHITHA REJITH, ARAVIND RAJU

Model Comparision

```
In this python notebook we are combining all the models that was done individually, namely:

🔷  Logistic Regression

🔷  Decision Tree

🔷  Random Forest

🔷  Support Vector Classification

🔷  Complement Naive Bayes

🔷  Multinomial Naive Bayes .

Initial text cleaning is done as necessary, stopwords, special characters were removed, text was tokenized, lemmatised and then combined together. Data was split using hold out method-test-train split using 80-20 ratio. Later they were vectorized using TF-IDF vectorizer, and then fed into each of the models. The final model with highest accuracy is indicated as the conclusion.
```

In [1]:

```
#importing libraries

import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv('Research_Article_train.csv') #reading the csv
```

In [3]:
```
df.head() #displaying the first 5 elements of the dataframe
```

Out[3]:

	ID	TITLE	ABSTRACT	Computer Science	Physics	Mathematics	Statistics	Quantitative Biology
0	1	Reconstructing Subject-Specific Effect Maps	Predictive models allow subject-specific inf...	1	0	0	0	0
1	2	Rotation Invariance Neural Network	Rotation invariance and translation invarian...	1	0	0	0	0
2	3	Spherical polyharmonics and Poisson kernels fo...	We introduce and develop the notion of spher...	0	0	1	0	0
3	4	A finite element approximation for the stochas...	The stochastic Landau--Lifshitz--Gilbert (LL...	0	0	1	0	0
4	5	Comparative study of Discrete Wavelet Transfor...	Fourier-transform infra-red (FTIR) spectra o...	1	0	0	1	0

In [4]:
```
df['text'] = df['TITLE']+" " + df['ABSTRACT']
df.drop(['ID','TITLE','ABSTRACT'], inplace = True, axis = 1)
df.head()
```

Out[4]:

	Computer Science	Physics	Mathematics	Statistics	Quantitative Biology	Quantitative Finance	text
0	1	0	0	0	0	0	Reconstructing Subject-Specific Effect Maps ...
1	1	0	0	0	0	0	Rotation Invariance Neural Network Rotation ...
2	0	0	1	0	0	0	Spherical polyharmonics and Poisson kernels fo...
3	0	0	1	0	0	0	A finite element approximation for the stochas...
4	1	0	0	1	0	0	Comparative study of Discrete Wavelet Transfor...

Pre-processing the data

In [5]:
```
from nltk.tokenize import TreebankWordTokenizer    #tokenizing
tree_tokeniser=TreebankWordTokenizer()
```

In [6]:
```
from nltk.stem import WordNetLemmatizer     #Word net Lemmatize
def lema_text(text):
    lematized_text=[WordNetLemmatizer().lemmatize(i) for i in t
    return lematized_text
```

In [7]:
```
from nltk.corpus import stopwords              # importing stopwo
stop_words=set(stopwords.words('english'))


def cleaning_stopwords(text):
    no_stopword_text = [w for w in text.split() if not w in sto
    return ' '.join(no_stopword_text)

```

In [8]:
```
def pre_processing(df,text, cleaned_text):
    df[cleaned_text]=df[text].str.lower()  #making each word to
    df[cleaned_text]=df[cleaned_text].apply(lambda strip: re.su
    df[cleaned_text]=df[cleaned_text].apply(lambda strip: re.su
    df[cleaned_text]=df[cleaned_text].apply(lambda s:' '.join([
    df[cleaned_text]=df[cleaned_text].apply(lambda s: tree_toke
    df[cleaned_text]=df[cleaned_text].apply(lambda s: lema_text
    df[cleaned_text]=df[cleaned_text].apply(lambda s: ' '.join(
    return df
```

In [9]:
```
df_new = pre_processing(df,'text','cleaned_text')
df_new.head()
```

Out[9]:

	Computer Science	Physics	Mathematics	Statistics	Quantitative Biology	Quantitative Finance	text	
0	1	0	0	0	0	0	Reconstructing Subject-Specific Effect Maps ...	r si
1	1	0	0	0	0	0	Rotation Invariance Neural Network Rotation ...	n
2	0	0	1	0	0	0	Spherical polyharmonics and Poisson kernels fo...	p p
3	0	0	1	0	0	0	A finite element approximation for the stochas...	a
4	1	0	0	1	0	0	Comparative study of Discrete Wavelet Transfor...	s

In [10]:
```
x=df_new.iloc[:,7] #selecting the input labels - x
x
```

Out[10]:
```
0        reconstructing subjectspecific effect map pred...
1        rotation invariance neural network rotation in...
2        spherical polyharmonics poisson kernel polyhar...
3        finite element approximation stochastic maxwel...
4        comparative study discrete wavelet transforms ...
                               ...
20967    contemporary machine learning guide practition...
20968    uniform diamond coating wcco hard alloy cuttin...
20969    analysing soccer game clustering conceptors pr...
20970    efficient simulation lefttail sum correlated l...
20971    optional stopping problem bayesians recently o...
Name: cleaned_text, Length: 20972, dtype: object
```

In [11]:
```
y=df_new.iloc[:,0:6] #selecting the output features - y
y
```

Out[11]:

	Computer Science	Physics	Mathematics	Statistics	Quantitative Biology	Quantitative Finance
0	1	0	0	0	0	0
1	1	0	0	0	0	0
2	0	0	1	0	0	0
3	0	0	1	0	0	0
4	1	0	0	1	0	0
...	...	...	...	...	...	...
20967	1	1	0	0	0	0
20968	0	1	0	0	0	0
20969	1	0	0	0	0	0
20970	0	0	1	1	0	0
20971	0	0	1	1	0	0

20972 rows × 6 columns

In [21]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y, test_size
```

In [22]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectoriser=TfidfVectorizer(min_df=5, max_df=0.9, token_pa
                                 max_features=10000,ngram_range
x_train_tvect=tfidf_vectoriser.fit_transform(x_train)
x_test_tvect=tfidf_vectoriser.transform(x_test)
```

1. Logistic regression for multi-label classification using a one-vs-rest

In [23]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier

# define model
model = LogisticRegression()

# define the ovr strategy
ovr = OneVsRestClassifier(model)

# fit model
ovr.fit(x_train_tvect, y_train)

# make predictions
yhat = ovr.predict(x_test_tvect)

from sklearn.metrics import accuracy_score

# View accuracy score
accuracy_m1 = accuracy_score(y_test, yhat)
print('The accuracy of Logitic Regression model with tfidf vect
```

```
The accuracy of Logitic Regression model with tfidf vectorizing is
0.6522050059594756
```

2. Decision tree classifier

In [15]:

```
 from sklearn.tree import DecisionTreeClassifier

# calling the decision tree into a parameter named classifier
classifier = DecisionTreeClassifier()

#training the model by fitting the TFIDFvectorized data onto th
classifier.fit(x_train_tvect,y_train)


#passing on the TFIDF vectorized test data (x_test_tvect) to ev
yhat = classifier.predict(x_test_tvect)


from sklearn.metrics import accuracy_score

# View accuracy score
accuracy_m2 = accuracy_score(y_test, yhat)
print('The accuracy of Decision Tree model with tfidf vectorizi
```

```
The accuracy of Decision Tree model with tfidf vectorizing is  0.5
022646007151371
```

3. Random Forest Classifier

In [16]:

```
from sklearn.ensemble import RandomForestClassifier

forest= RandomForestClassifier(n_estimators = 200)
model = forest.fit(x_train_tvect, y_train)

yhat= model.predict(x_test_tvect)
yhat

from sklearn.metrics import accuracy_score
# View accuracy score
accuracy_m3 = accuracy_score(y_test, yhat)
print('The accuracy of Random Forest model with tfidf vectorizi
```

```
The accuracy of Random Forest model with tfidf vectorizing is  0.5
973778307508939
```

4. Multinomial Naive Bayes

In [17]:
```
from sklearn.naive_bayes import MultinomialNB

model_nb=MultinomialNB()
model_nb=OneVsRestClassifier(model_nb)
model_nb.fit(x_train_tvect,y_train)
yhat = model_nb.predict(x_test_tvect)

accuracy_m4 = accuracy_score(y_test, yhat)
print('The accuracy of Multinomial Naive Bayes model with tfidf
```

```
The accuracy of Multinomial Naive Bayes model with tfidf vectorizi
ng is  0.6421930870083432
```

5. Complement Naive Bayes

In [18]:
```
from sklearn.naive_bayes import ComplementNB

model=ComplementNB()
model=OneVsRestClassifier(model_nb)
model.fit(x_train_tvect,y_train)
yhat = model_nb.predict(x_test_tvect)

accuracy_m5 = accuracy_score(y_test, yhat)
print('The accuracy of Complement Naive Bayes model with tfidf
```

```
The accuracy of Complement Naive Bayes model with tfidf vectorizin
g is  0.6421930870083432
```

6. Linear Support vector classification

In [19]:
```
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsRestClassifier


model= LinearSVC()
ovr=OneVsRestClassifier(model)
ovr.fit(x_train_tvect,y_train)
yhat = ovr.predict(x_test_tvect)

accuracy_m6 = accuracy_score(y_test, yhat)
print('The accuracy of SVC model with tfidf vectorizing is ',ac
```

```
The accuracy of SVC model with tfidf vectorizing is  0.64243146603
09892
```

Comparing the best among the 6 models

In [24]:
```
classification = {0:'Logistic Regression',1:'Decision Tree', 2:
                      4: 'Gaussian Naive Bayes', 5:'Support Vector
array1= [accuracy_m1,accuracy_m2,accuracy_m3, accuracy_m4, accu
acc1 = np.argmax([accuracy_m1,accuracy_m2,accuracy_m3,accuracy_
print('Best classifier according to TF IDF Vectorization is {a}
```

```
Best classifier according to TF IDF Vectorization is Logistic Regr
ession with accuracy 0.6522
```

Conclusion

🔷 From the results it can be concluded that Logistic Regression model with TF-IDF vectorization technique yields the best accuracy of approximately 65.22%. We further conduct the experiments based on this model and accuracy.

🔷 In our comparison to other techniques for supervised classification such as SVMs or ensemble methods, logistic regression is rather fast and witha a better accuracy.

🔷 By applying a logarithmic transformation to the outcome variable, we can model a nonlinear association linearly

In [ ]:
```

```