News Classification Project

Import Libraries

Loading all libraries to be used from sklearn

```
import copy
import numpy as np
import matplotlib.pyplot as plt
import re
import nltk
nltk.download('stopwords')
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import VarianceThreshold
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import datasets, linear_model, metrics
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
```

Data preparation

Load data

Mounting google drive

```
from google.colab import drive
drive.mount('/content/drive')
```

Lets load the data from dsjVoxArticles.tsv file. We will clean the title to remove special characters and punctuations. We will store title in titles and Category in categories

```
titles = []
categories = []
with open('/content/drive/MyDrive/Data/dsjVoxArticles.tsv','r') as tsv:
    count = 0;
    for line in tsv:
        a = line.strip().split('\t')[:3]
        if a[2] in ['Business & Finance', 'Health Care', 'Science & Health', 'Politics & Policy', 'Criminal
            title = a[0].lower()
```

```
            title = re.sub('\s\W',' ',title)
            title = re.sub('\W\s',' ',title)
            titles.append(title)
            categories.append(a[2])
```

We can print and check the data loaded in titles and categories

```
print("Titles-\n", "\n".join(titles[:5]))
print("\nCategories-\n", "\n".join(categories[:5]))
```

Titles-
 bitcoin is down 60 percent this year here's why i'm still optimistic.
9 charts that explain the history of global wealth
remember when legal marijuana was going to send crime skyrocketing?
obamacare succeeded for one simple reason it's horrible to be uninsured
the best obamacare data comes from a home office in michigan

Categories-
 Business & Finance
Business & Finance
Criminal Justice
Health Care
Health Care

Split data

Split data into 3 parts - training, development and test. We will use training data to train out model and use development data to check and tune hyper parameters. And finally use test data to see how our model performs

```
title_tr, title_te, category_tr, category_te = train_test_split(titles,categories)
title_tr, title_de, category_tr, category_de = train_test_split(title_tr,category_tr)
print("Training: ",len(title_tr))
print("Development: ",len(title_de),)
print("Testing: ",len(title_te))
```

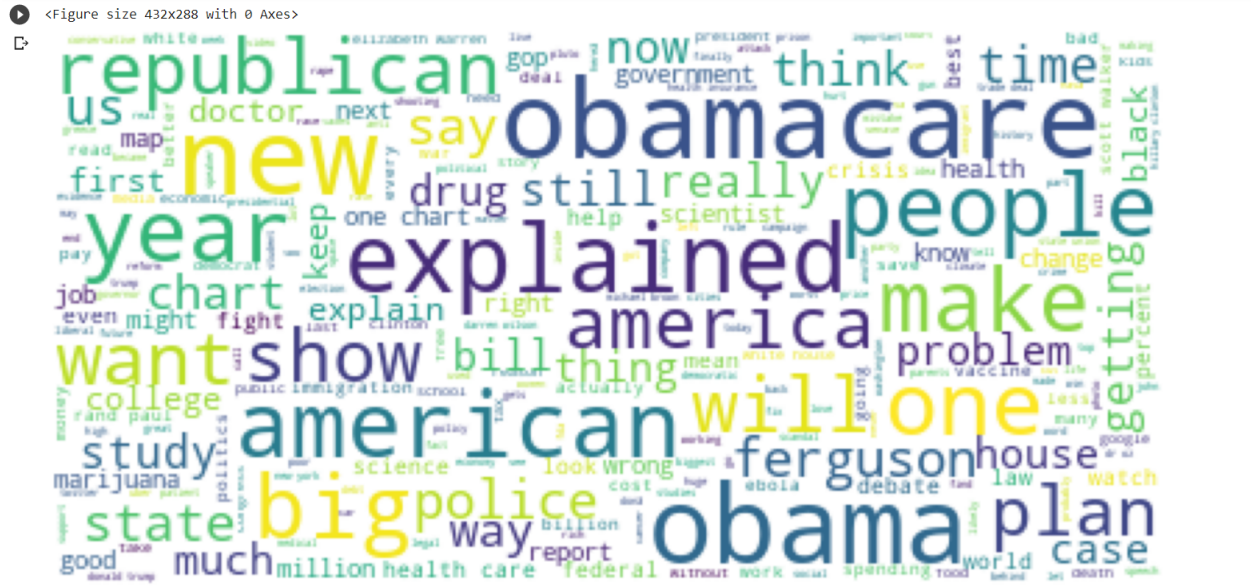
Training: 1779
Development: 594
Testing: 792

Visualising data

Using wordCload we can visualize our data.

```
from wordcloud import WordCloud
text = " ".join(title_tr)
wordcloud = WordCloud().generate(text)
plt.figure()
plt.subplots(figsize=(20,12))
wordcloud = WordCloud(
    background_color="white",
    max_words=len(text),
    max_font_size=40,
    relative_scaling=.5).generate(text)
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Data Preprocessing

Vectorization of data

Vectorize the data using Bag of words (BOW)

One of the biggest problems with text is that it is messy and unstructured, and machine learning algorithms prefer structured, well defined fixed-length inputs and by using the Bag-of-Words technique we can convert variable-length texts into a fixed-length vector

```
tokenizer = nltk.tokenize.RegexpTokenizer(r"\w+")
stop_words = nltk.corpus.stopwords.words("english")
```

```
vectorizer = CountVectorizer(tokenizer=tokenizer.tokenize, stop_words=stop_words)

vectorizer.fit(iter(title_tr))
Xtr = vectorizer.transform(iter(title_tr))
Xde = vectorizer.transform(iter(title_de))
Xte = vectorizer.transform(iter(title_te))

encoder = LabelEncoder()
encoder.fit(category_tr)
Ytr = encoder.transform(category_tr)
Yde = encoder.transform(category_de)
Yte = encoder.transform(category_te)
```

Lets look at what exactly is this vectorizer doing. We will first create reverse dictionary from the vectorizer. Iterating over the vectorized sentence Nasa scientists are good. We get the vector to be representative of three words "good", "nasa" and "scientists". The order has been changed because bag of words does not preserve order.

```
reverse_vocabulary = {}
vocabulary = vectorizer.vocabulary_
for word in vocabulary:
    index = vocabulary[word]
    reverse_vocabulary[index] = word

vector = vectorizer.transform(iter(['Nasa scientists are good']))
indexes = vector.indices
for i in indexes:
    print (reverse_vocabulary[i])
```

good
nasa
scientists

Feature Reduction

We can check the variance of the feature and drop them based on a threshold.

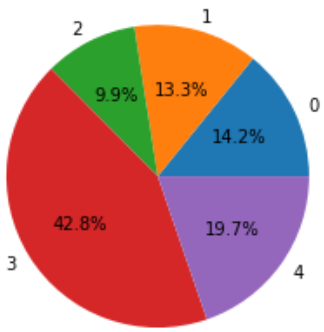
```
print("Number of features before reduction : ", Xtr.shape[1])
selection = VarianceThreshold(threshold=0.001)
Xtr_whole = copy.deepcopy(Xtr)
Ytr_whole = copy.deepcopy(Ytr)
selection.fit(Xtr)
Xtr = selection.transform(Xtr)
Xde = selection.transform(Xde)
Xte = selection.transform(Xte)
print("Number of features after reduction : ", Xtr.shape[1])
```

Number of features before reduction : 4300
Number of features after reduction : 1780

Sampling data

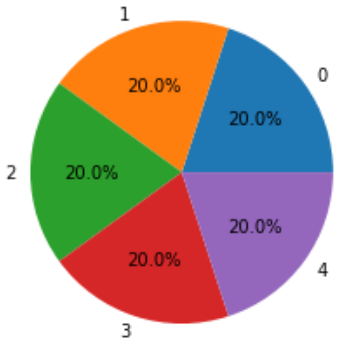
We will count the number of diffrent labels in dataset and plot a pie chart distribution.

```
labels = list(set(Ytr))
counts = []
for label in labels:
    counts.append(np.count_nonzero(Ytr == label))
plt.pie(counts, labels=labels, autopct='%1.1f%%')
plt.show()
```



As we can see the class labels are not uniformy distributed so we will use SMOT and oversample the classes which are less in number so that classes are equally distributed

```
sm = SMOTE(random_state=42)
Xtr, Ytr = sm.fit_sample(Xtr, Ytr)
labels = list(set(Ytr))
counts = []
for label in labels:
    counts.append(np.count_nonzero(Ytr == label))
plt.pie(counts, labels=labels, autopct='%1.1f%%')
plt.show()
```



Train Models

Linear Regression

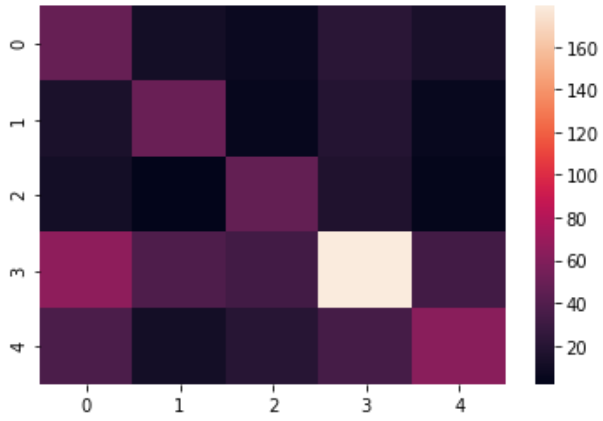
Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It's used to predict values within a continuous range, rather than trying to classify them into categories.

```
from sklearn import datasets, linear_model, metrics
reg = linear_model.LinearRegression()
reg.fit(Xtr,Ytr)
pred = reg.predict(Xde)
pred = (pred > 0.5)
pred
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

	precision	recall	f1-score	support
Business & Finance	0.30	0.49	0.37	82
Criminal Justice	0.44	0.48	0.46	73
Health Care	0.48	0.66	0.56	62
Politics & Policy	0.67	0.52	0.59	261
Science & Health	0.61	0.47	0.53	116
accuracy			0.52	594
macro avg	0.50	0.52	0.50	594
weighted avg	0.56	0.52	0.53	594

Heat map

```
pred = reg.predict(Xte)
sns.heatmap(confusion_matrix(Yte, pred))
```



Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

```
from sklearn.linear_model import LogisticRegression
Lr = LogisticRegression()
Lr.fit(Xtr,Ytr)
pred = Lr.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

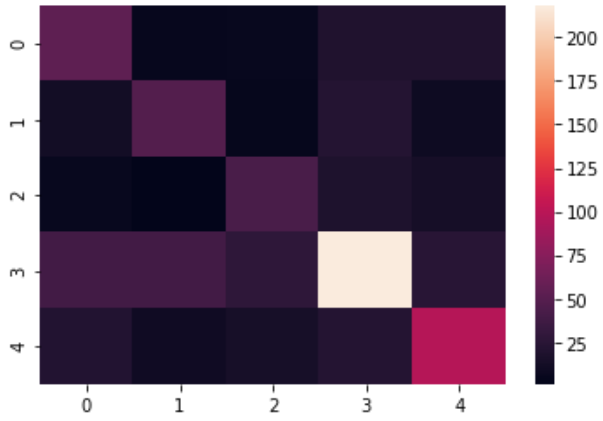
```
→                     precision    recall  f1-score   support

   Business & Finance       0.45      0.52      0.49        82
     Criminal Justice       0.50      0.51      0.50        73
          Health Care       0.54      0.63      0.58        62
     Politics & Policy       0.73      0.63      0.68       261
     Science & Health        0.56      0.61      0.58       116

             accuracy                           0.60       594
            macro avg       0.56      0.58      0.57       594
         weighted avg       0.61      0.60      0.60       594
```

Heat map

```
pred = Lr.predict(Xte)
sns.heatmap(confusion_matrix(Yte, pred))
```



SVM

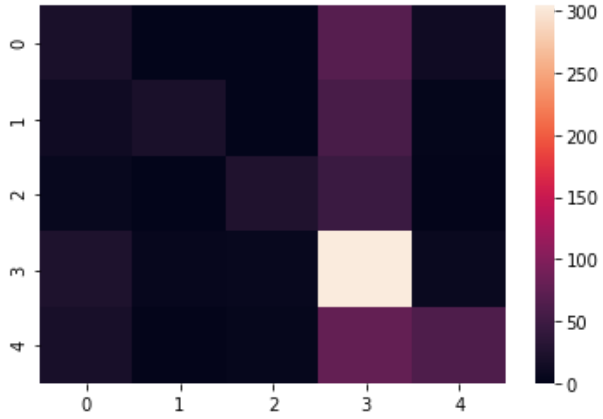
In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(Xtr, Ytr)
pred = svc.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

	precision	recall	f1-score	support
Business & Finance	0.33	0.26	0.29	82
Criminal Justice	0.57	0.16	0.26	73
Health Care	0.55	0.27	0.37	62
Politics & Policy	0.55	0.88	0.68	261
Science & Health	0.69	0.36	0.47	116
accuracy			0.54	594
macro avg	0.54	0.39	0.41	594
weighted avg	0.55	0.54	0.50	594

Heat map

```
pred = svc.predict(Xte)
sns.heatmap(confusion_matrix(Yte, pred))
```



Random Forest

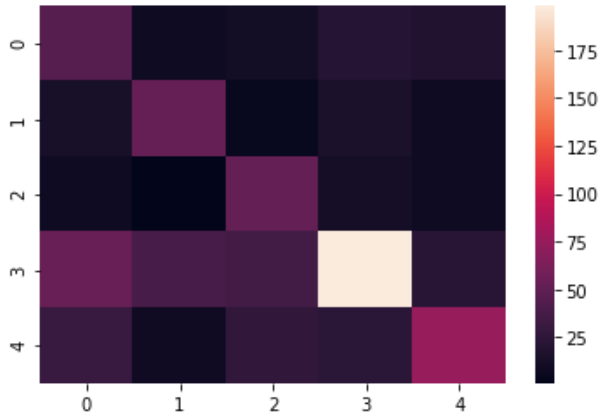
Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

```
rf = RandomForestClassifier(n_estimators=40)
rf.fit(Xtr, Ytr)
pred = rf.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

	precision	recall	f1-score	support
Business & Finance	0.37	0.41	0.39	82
Criminal Justice	0.46	0.52	0.49	73
Health Care	0.46	0.73	0.57	62
Politics & Policy	0.71	0.59	0.65	261
Science & Health	0.61	0.55	0.58	116
accuracy			0.57	594
macro avg	0.52	0.56	0.53	594
weighted avg	0.59	0.57	0.57	594

Heat map

```
pred = rf.predict(Xte)
sns.heatmap(confusion_matrix(Yte, pred))
```



Neural Network

MLP classifier

Class MLP classifier implements a multi-layer perceptron (MLP) algorithm that trains using back proportion.

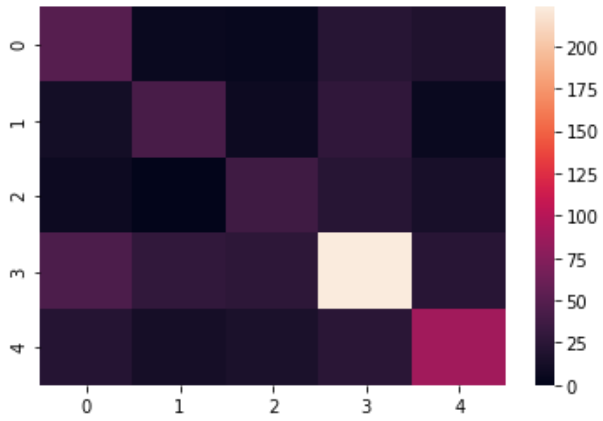
MLP trains on two arrays: array X of size (n_samples, n_features), which holds the training samples represented as floating-point feature vectors; and array y of size (n_samples,), which holds the target values (class labels) for the training samples:

```
mlp = MLPClassifier(solver='adam', alpha=1e-5, hidden_layer_sizes=(100, 20), random_state=1, max_iter=400)
mlp.fit(Xtr, Ytr)
pred = mlp.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

	precision	recall	f1-score	support
Business & Finance	0.60	0.56	0.58	93
Criminal Justice	0.58	0.59	0.59	78
Health Care	0.48	0.58	0.53	53
Politics & Policy	0.67	0.65	0.66	241
Science & Health	0.67	0.67	0.67	129
accuracy			0.63	594
macro avg	0.60	0.61	0.61	594
weighted avg	0.63	0.63	0.63	594

Heat map

```
pred = mlp.predict(Xte)
sns.heatmap(confusion_matrix(Yte, pred))
```



Unsupervised Learning

k-means clustering

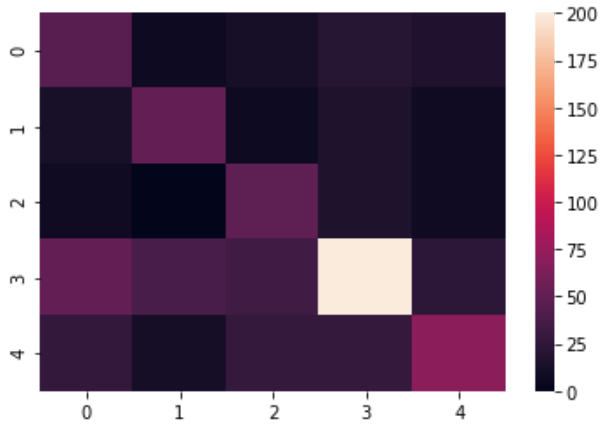
To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3, random_state=1)
kmeans.fit(Xtr, Ytr)
pred = kmeans.predict(Xde)
print(classification_report(Yde, pred, target_names=encoder.classes_))
```

	precision	recall	f1-score	support
Business & Finance	0.39	0.46	0.42	82
Criminal Justice	0.41	0.51	0.45	73
Health Care	0.44	0.69	0.54	62
Politics & Policy	0.70	0.58	0.63	261
Science & Health	0.64	0.50	0.56	116
accuracy			0.55	594
macro avg	0.52	0.55	0.52	594
weighted avg	0.58	0.55	0.56	594

Heat map

```
pred = kmeans.predict(Xte)
sns.heatmap(confusion_matrix(Yte, pred))
```



Comparative Analysis

Kfold

That k-fold cross validation is a procedure used to estimate the skill of the model on new data.

```
seed = 50
kfold = model_selection.KFold(n_splits=10, random_state=seed)
```

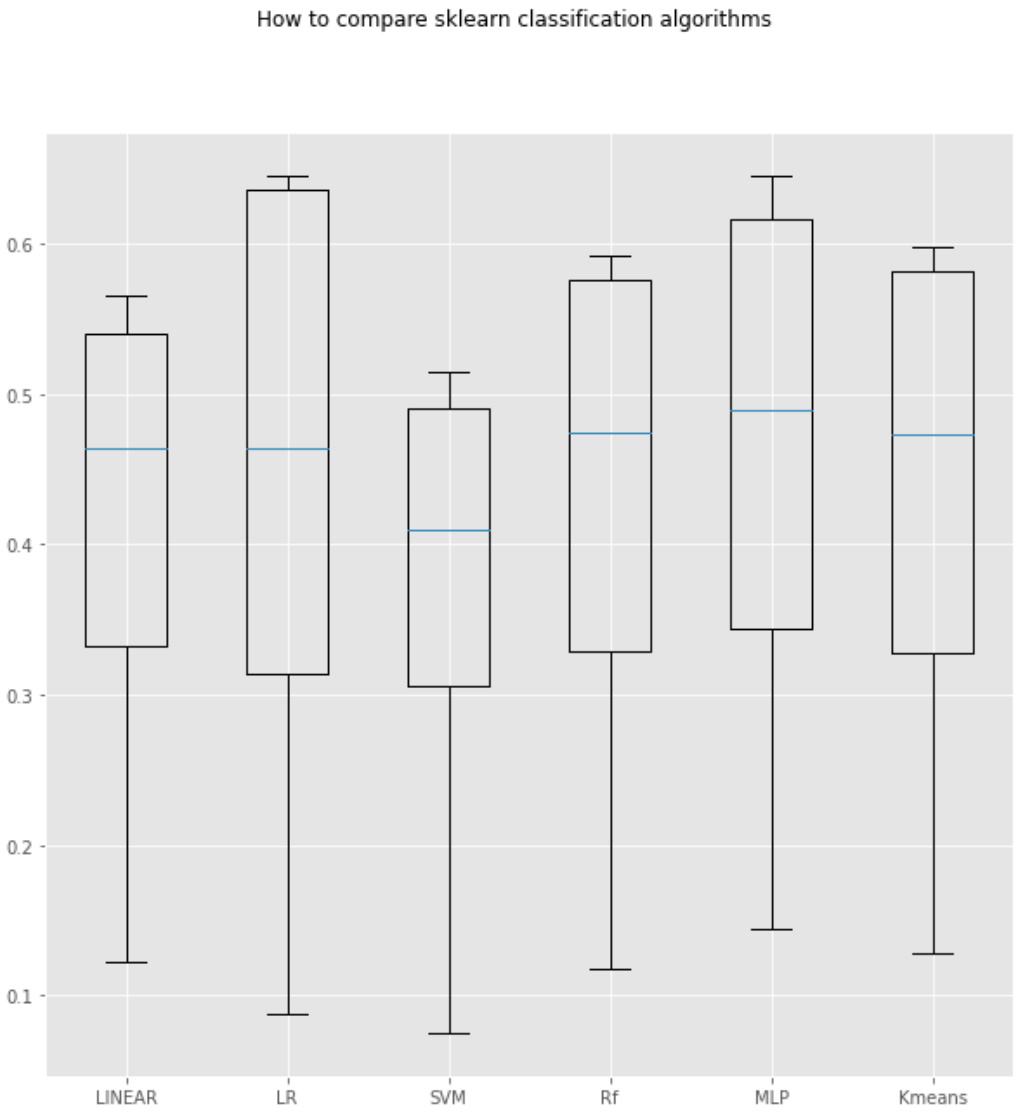
```
models = []
    models.append(('LINEAR', LinearDiscriminantAnalysis()))
    models.append(('LR', LogisticRegression()))
```

```
    models.append(('SVM', SVC()))
    models.append(('Rf', RandomForestClassifier()))
    models.append(('MLP', MLPClassifier(solver='adam', alpha=1e-5, hidden_layer_sizes=(100, 20), random_sta
    models.append(('Kmeans', KMeans()))
```

```
results = []
    names = []
    scoring = 'accuracy'
    for name, model in models:
         kfold = model_selection.KFold(n_splits=10, random_state=seed)
         cv_results = model_selection.cross_val_score(model, Xtr, Ytr, cv=kfold, scoring=scoring)
         results.append(cv_results)
         names.append(name)
         msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
         print(msg)
```

```
fig = plt.figure(figsize=(10,10))
    fig.suptitle('How to compare sklearn classification algorithms')
    ax = fig.add_subplot(111)
    plt.boxplot(results)
    ax.set_xticklabels(names)
    plt.show()
```

Results:



Final Model: Logistic Regression

We will now try to understand why Logistic Regression is getting good results. We will get all the coefficients of the features and then print the top 20 words based on their weight. As we can see all the words are closely related to the category, hence multinomial naive bayesian predicts correct label with a good F1 score.

```
Lr1 = LogisticRegression()
Lr1.fit(Xtr_whole,Ytr_whole)

coefs = Lr1.coef_
target_names = encoder.classes_

for i in range(len(target_names)):
    words = []
    for j in coefs[i].argsort()[-20:]:
```

```
        words.append(reverse_vocabulary[j])
    print (target_names[i], '-', words, "\n")
```

Business & Finance - ['buying', 'sony', 'economics', 'money', 'neutrality', 'recovery', 'google', 'cellphone', 'self', 'economy', 'chipotle', 'microsoft', 'bitcoin', 'stock', 'greek', 'deal', 'facebook', 'amazon', 'greece', 'apple']

Criminal Justice - ['yes', 'racist', 'sentences', 'black', 'fergusonâ', 'violent', 'moments', 'prisons', 'drug', 'wilson', 'darren', 'shooting', 'cosby', 'prison', 'rape', 'eric', 'marijuana', 'crime', 'police', 'ferguson']

Health Care - ['medical', 'happy', 'teens', 'soda', 'salt', 'lives', 'hospitals', 'healthcare', 'outbreaks', 'increasing', 'doctor', 'birth', 'care', 'medicare', 'cigarettes', 'hospital', 'health', 'medicaid', 'va', 'obamacare']

Politics & Policy - ['politics', 'president', 'jeb', 'presidential', 'democratic', 'paul', 'political', 'student', 'scott', 'walker', 'elizabeth', 'republicans', '1', 'law', '2016', 'republican', 'trump', 'clinton', 'gop', 'obama']

Science & Health - ['fall', 'earth', 'brain', 'oz', 'energy', 'food', 'dr', 'medicine', 'deadly', 'studies', 'exercise', 'cancer', 'fda', 'solar', 'space', 'ebola', 'pluto', 'nasa', 'science', 'scientists']