Assignment 1

Question 1:

a) Raw data consists of a dictionary like structure with keys as names and values as arrays/descriptions/location. Names of keys are: data,target, target_names, DESCR, feature_names etc.

```
Out[7]: {'data': array([[5.1, 3.5, 1.4, 0.2],
                        [4.9, 3. , 1.4, 0.2],
                        [4.7, 3.2, 1.3, 0.2],
                        [4.6, 3.1, 1.5, 0.2],
                        [5. , 3.6, 1.4, 0.2],
                        [5.4, 3.9, 1.7, 0.4],
```

```
       [5.9, 3. , 5.1, 1.8]]),
'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
'DESCR': '    iris dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Ch
```

b) Instances are also known as records or observations or samples. Suppose we have a matrix with some columns (in our case 5 columns) and rows with some values ([5.1, 3.5, 1.4, 0.3,0]). These values in rows are instances. In our dataset we have in total 150 instances.

```
In [10]: iris_df.head()
```

Out[10]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0

c) A feature or an attribute could be described as a property/characteristic. For example: Sepal length, sepal width, petal length, petal width are describing iris data.

```
In [11]: iris_df.columns

Out[11]: Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
                'petal width (cm)', 'Target'],
               dtype='object')
```

Feature vector represents/ describes instance. I can say feature vector could be subset of instance. For example, I just want to have sepal length and sepal width for one instance. This (sepal length, sepal width) here will become a feature vector.

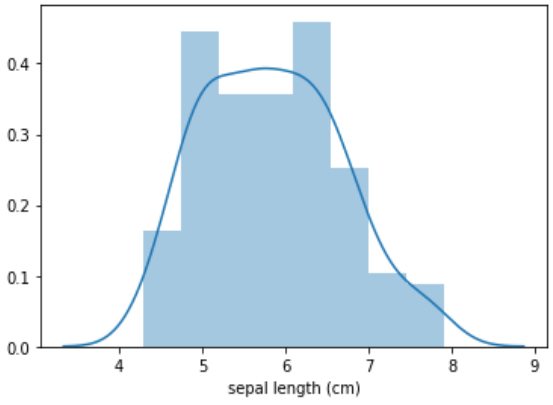
d) Categories, State of nature, Label, Class-Label, Target or Target-Variable are synonymous. This is nothing but a value assignment to an object/instance. This value is assigned according to the description of the features. In our case name of the class label is 'Target'. Also, [5.1, 3.5, 1.4, 0.3] the first instance's target value is '0' which is setosa.

```
In [9]: iris_df["Target"]

Out[9]: 0      0
        1      0
        2      0
        3      0
        4      0
              ..
        145    2
        146    2
        147    2
        148    2
        149    2
        Name: Target, Length: 150, dtype: int32
```

e) Explanatory variable is a type of Independent variable which explains changes in the response variable and whose value will not dependent on any other variable. But response/dependent variable's value depends on other variables. Explanatory variable is plotted on x-axis and response variable is plotted on y-axis.

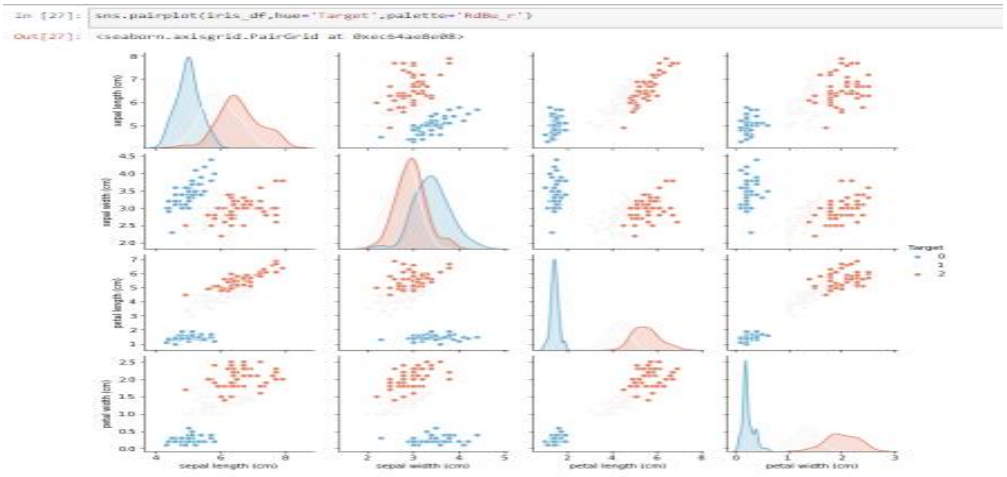
f) Distribution of a feature means how its values are spread across the range. For eg: for sepal length we can see the distribution below, example: Median value (50%) of sepal length is 5.8.

```
In [16]: sns.distplot(iris_df['sepal length (cm)'])

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0xec5b8d4a48>
```

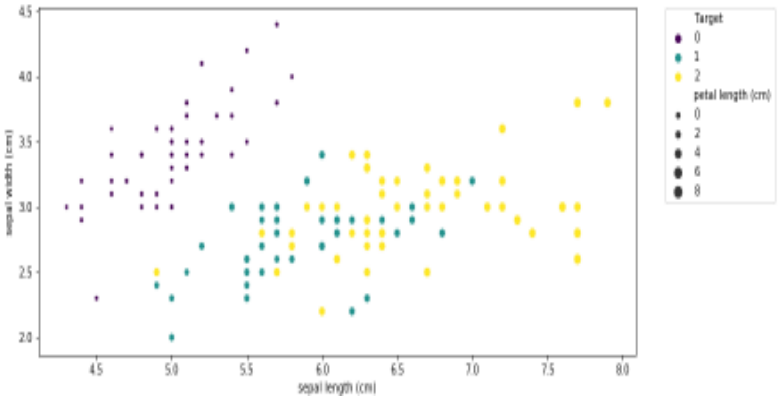


```
In [19]: iris_df['sepal length (cm)'].describe()

Out[19]: count    150.000000
         mean       5.843333
         std        0.828066
         min        4.300000
         25%        5.100000
         50%        5.800000
         75%        6.400000
         max        7.900000
         Name: sepal length (cm), dtype: float64
```

g) To visualize more than three dimensions we can use pairplot or we can use scatter plot with dimensions x and y, hue, depth, size etc.



```
In [26]: plt.figure(figsize= (12,5))
sns.scatterplot(x='sepal length (cm)',y='sepal width (cm)',hue='Target',size='petal length (cm)',palette='viridis',data=iris_df)
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

Out[26]: <matplotlib.legend.Legend at 0xec644c8b08>
```



PCA visualization:

Here we can see that ‘Setosa’ is well separated from ‘Versicolor’ and ‘Virginica’.

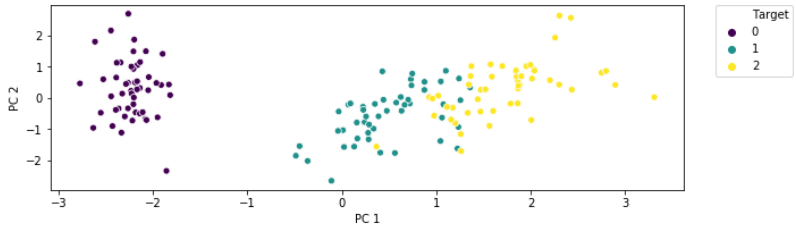
In [33]:
```
pc_df.head()
```

Out[33]:

	PC 1	PC 2	Target
0	-2.264703	0.480027	0
1	-2.080961	-0.674134	0
2	-2.364229	-0.341908	0
3	-2.299384	-0.597395	0
4	-2.389842	0.646835	0

In [50]:
```
plt.figure(figsize=(10,3))
sns.scatterplot(x='PC 1',y='PC 2',hue='Target',palette='viridis',data=pc_df)
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[50]:
```
<matplotlib.legend.Legend at 0xc892b65548>
```



Question 2:

a) Raw data looks like below :

In [65]:
```
news_group.data
```

Out[65]:
```
["From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?\nNntp-Posting-Host: rac3.wam.umd.edu\nOrganizat
n: University of Maryland, College Park\nLines: 15\n\n I was wondering if anyone out there could enlighten me on this car
aw\nthe other day. It was a 2-door sports car, looked to be from the late 60s/\nearly 70s. It was called a Bricklin. The d
s were really small. In addition,\nthe front bumper was separate from the rest of the body. This is \nall I know. If anyon
an tellme a model name, engine specs, years\nof production, where this car is made, history, or whatever info you\nhave on
is funky looking car, please e-mail.\n\nThanks,\n- IL\n   ---- brought to you by your neighborhood Lerxst ----\n\n\n\n\n",
 "From: guykuo@carson.u.washington.edu (Guy Kuo)\nSubject: SI Clock Poll - Final Call\nSummary: Final call for SI clock re
ts\nKeywords: SI,acceleration,clock,upgrade\nArticle-I.D.: shelley.1qvfo9INNc3s\nOrganization: University of Washington\nL
s: 11\nNNTP-Posting-Host: carson.u.washington.edu\n\nA fair number of brave souls who upgraded their SI clock oscillator h
\nshared their experiences for this poll. Please send a brief message detailing\nyour experiences with the procedure. Top
ed attained, CPU rated speed,\nadd on cards and adapters, heat sinks, hour of usage per day, floppy disk\nfunctionality wi
800 and 1.4 m floppies are especially requested.\n\nI will be summarizing in the next two days, so please add to the netwo
\nknowledge base if you have done the clock upgrade and haven't answered this\npoll. Thanks.\n\nGuy Kuo <guykuo@u.washingt
edu>\n",
```

In [67]:
```
print(news_group.target , news_group.target_names)
```

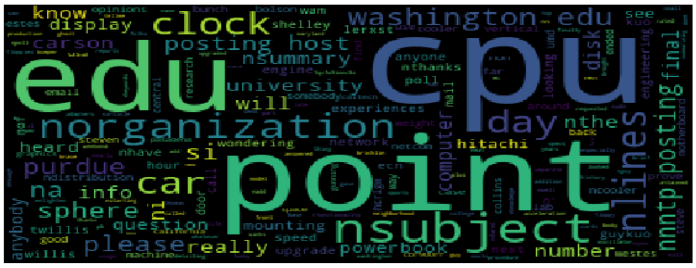
```
[7 4 4 ... 3 1 8] ['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardwar
e', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt', 's
ci.electronics', 'sci.med', 'sci.space', 'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast', 'talk.politic
s.misc', 'talk.religion.misc']
```

b)

```
In [68]: from sklearn.feature_extraction.text import TfidfVectorizer

In [70]: vectorizer = TfidfVectorizer()

In [73]: vectors = vectorizer.fit_transform(news_group_df['Text'])

In [74]: vectors.shape
Out[74]: (11314, 130107)
```

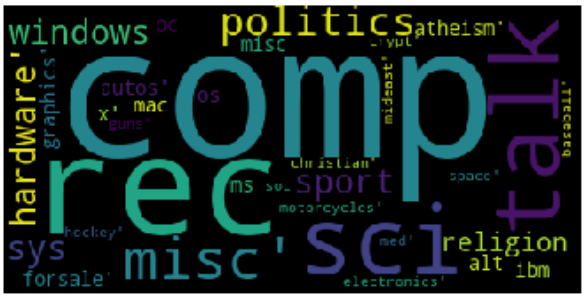
c) Tag cloud for words there in feature ‘Text’ is :

```
In [20]: from wordcloud import WordCloud, STOPWORDS

In [32]: text1 = news_group_df['Text'].values
         wordcloud = WordCloud().generate(str(text1))

In [34]: plt.figure(figsize=(10,6))
         plt.imshow(wordcloud)
         plt.axis("off")
         plt.show()
```



Tag cloud for Target names or Target values is:

```
In [36]: text2 = news_group.target_names
         wordcloud1 = WordCloud().generate(str(text2))
         plt.figure(figsize=(8,3))
         plt.imshow(wordcloud1)
         plt.axis("off")
         plt.show()
```



d) Features like n gram and rare words by IDF values:

```
In [118]: from sklearn.feature_extraction.text import TfidfVectorizer

In [142]: vectorizer = TfidfVectorizer(ngrams)

In [143]: vectors = vectorizer.fit_transform(news_group_df['Text'])

In [144]: vectors.shape
Out[144]: (2034, 34118)
```

Question 3:

a) Text classification is a type of supervised machine learning. Here we classify a group of words into a particular category. Classification of text has broader applications, for example: in our case (news_group)it can be used to classify news text into technical, computers, science, politics, religion etc. It is also used for filtering of spam emails, in sentiment analysis etc.

```
In [50]: news_group_df.head()
```

Out[50]:

	Text	Target
0	From: lerxst@wam.umd.edu (where's my thing)\nS...	alt.atheism
1	From: guykuo@carson.u.washington.edu (Guy Kuo)...	comp.graphics
2	From: twillis@ec.ecn.purdue.edu (Thomas E Will...	comp.os.ms-windows.misc
3	From: jgreen@amber (Joe Green)\nSubject: Re: W...	comp.sys.ibm.pc.hardware
4	From: jcm@head-cfa.harvard.edu (Jonathan McDow...	comp.sys.mac.hardware

b) Pre-processing in text classification means transforming raw data in some form and as a result we get index terms. Various steps in pre-processing are: tokenisation, stop word removal, stemming and lemmatization.

Removing Punctuation and numbers and then tokenisation:

```
In [51]: import nltk

In [52]: from nltk.tokenize import word_tokenize

In [78]: result = str(news_group_df['Text'][0]).translate(str.maketrans('','',string.punctuation))
         result = result.translate(str.maketrans('','','1234567890'))

In [79]: tokens = word_tokenize(result)

In [80]: print(tokens)
['From', 'lerxstwamumdedu', 'wheres', 'my', 'thing', 'Subject', 'WHAT', 'car', 'is', 'this', 'NntpPostingHost', 'racwamumdedu',
'Organization', 'University', 'of', 'Maryland', 'College', 'Park', 'Lines', 'I', 'was', 'wondering', 'if', 'anyone', 'out', 'th
ere', 'could', 'enlighten', 'me', 'on', 'this', 'car', 'I', 'saw', 'the', 'other', 'day', 'It', 'was', 'a', 'door', 'sports',
'car', 'looked', 'to', 'be', 'from', 'the', 'late', 's', 'early', 's', 'It', 'was', 'called', 'a', 'Bricklin', 'The', 'doors',
'were', 'really', 'small', 'In', 'addition', 'the', 'front', 'bumper', 'was', 'separate', 'from', 'the', 'rest', 'of', 'the',
'body', 'This', 'is', 'all', 'I', 'know', 'If', 'anyone', 'can', 'tellme', 'a', 'model', 'name', 'engine', 'specs', 'years', 'o
f', 'production', 'where', 'this', 'car', 'is', 'made', 'history', 'or', 'whatever', 'info', 'you', 'have', 'on', 'this', 'funk
y', 'looking', 'car', 'please', 'email', 'Thanks', 'IL', 'brought', 'to', 'you', 'by', 'your', 'neighborhood', 'Lerxst']
```

Stop word Removal:

```
In [82]: from nltk.corpus import stopwords

In [83]: stopWords = set(stopwords.words('english'))

In [84]: tokens = [word for word in tokens if not word in stopWords]

In [85]: print(tokens)

['From', 'lerxstwamumdedu', 'wheres', 'thing', 'Subject', 'WHAT', 'car', 'NntpPostingHost', 'racwamumdedu', 'Organization', 'Un
iversity', 'Maryland', 'College', 'Park', 'Lines', 'I', 'wondering', 'anyone', 'could', 'enlighten', 'car', 'I', 'saw', 'day',
'It', 'door', 'sports', 'car', 'looked', 'late', 'early', 'It', 'called', 'Bricklin', 'The', 'doors', 'really', 'small', 'In',
'addition', 'front', 'bumper', 'separate', 'rest', 'body', 'This', 'I', 'know', 'If', 'anyone', 'tellme', 'model', 'name', 'eng
ine', 'specs', 'years', 'production', 'car', 'made', 'history', 'whatever', 'info', 'funky', 'looking', 'car', 'please', 'emai
l', 'Thanks', 'IL', 'brought', 'neighborhood', 'Lerxst']
```

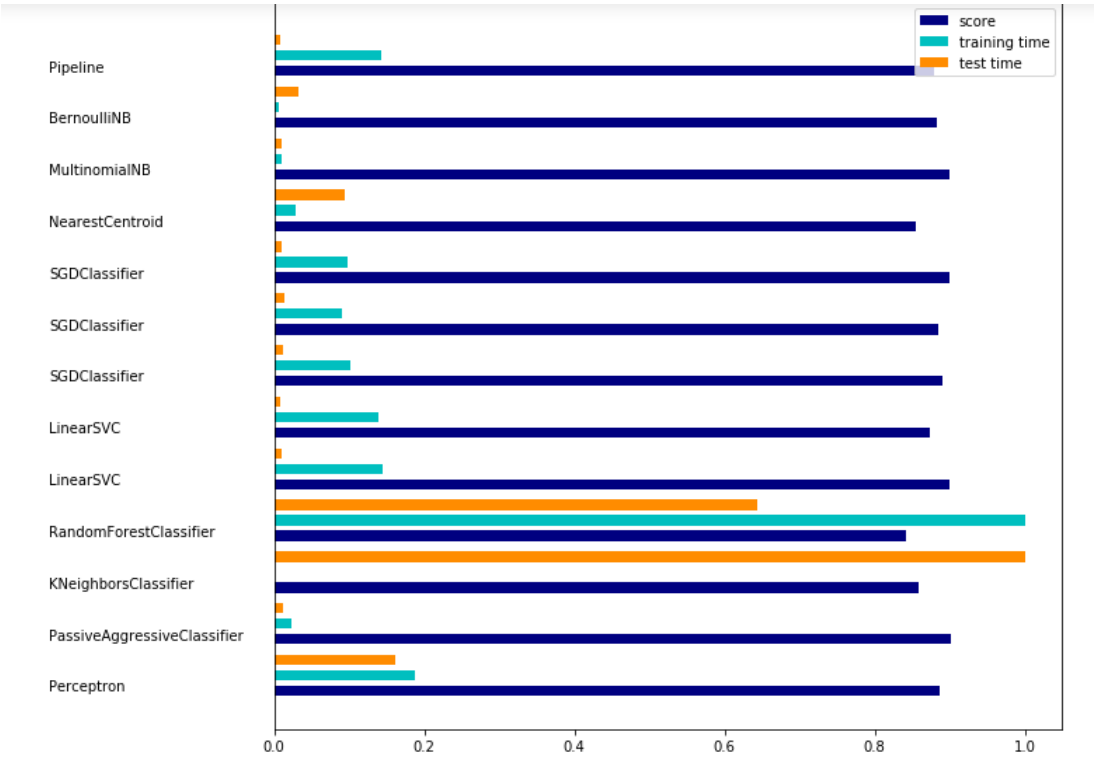
Stemming:

```
In [86]: from nltk.stem import PorterStemmer

In [87]: stemm = PorterStemmer()

In [93]: for word in tokens:
             print(stemm.stem(word),end=' ')

from lerxstwamumdedu where thing subject what car nntppostinghost racwamumdedu organ univers maryland colleg park line I wonder
anyon could enlighten car I saw day It door sport car look late earli It call bricklin the door realli small In addit front bum
per separ rest bodi thi I know If anyon tellm model name engin spec year product car made histori whatev info funki look car pl
eas email thank IL brought neighborhood lerxst
```

Benefits of Pre-processing:

- Cleaning of data, like punctuations, numbers these can be removed which are not contributing to the classification process. Same goes for stop word removal.
- Pre-processing can enhance the accuracy of the classification process.

c) The provided example[2] given for this question has shown many models which can be implemented. For example: Perceptron, Random Forest, KNN, Bernoulli and Multinomial NB etc.



d) A model is a representation of training data in the form of a pattern which is later used to perform on unknown data. For training data we have instances corresponding to some labels from where we get a pattern. Then this model is used on unknown data to predict its label.

Definition from Tom Mitchell:

A computer program is said to learn from **experience** *E* with respect to some class of **tasks** *T* and **performance** measure *P* if its performance at tasks in T, as measured by P, improves with experience E.

So, Model could be experience E which is formed after performing tasks T.

e) By taking **example code from sklearn website**[1], I implemented Multinominal Naïve Bayes which is a supervised machine learning method.

```
In [3]: from sklearn.feature_extraction.text import TfidfVectorizer
        categories = ['alt.atheism', 'talk.religion.misc',
                      'comp.graphics', 'sci.space']
        newsgroups_train = fetch_20newsgroups(subset='train',
                                              categories=categories)
        vectorizer = TfidfVectorizer()
        vectors = vectorizer.fit_transform(newsgroups_train.data)
        vectors.shape
Out[3]: (2034, 34118)
```

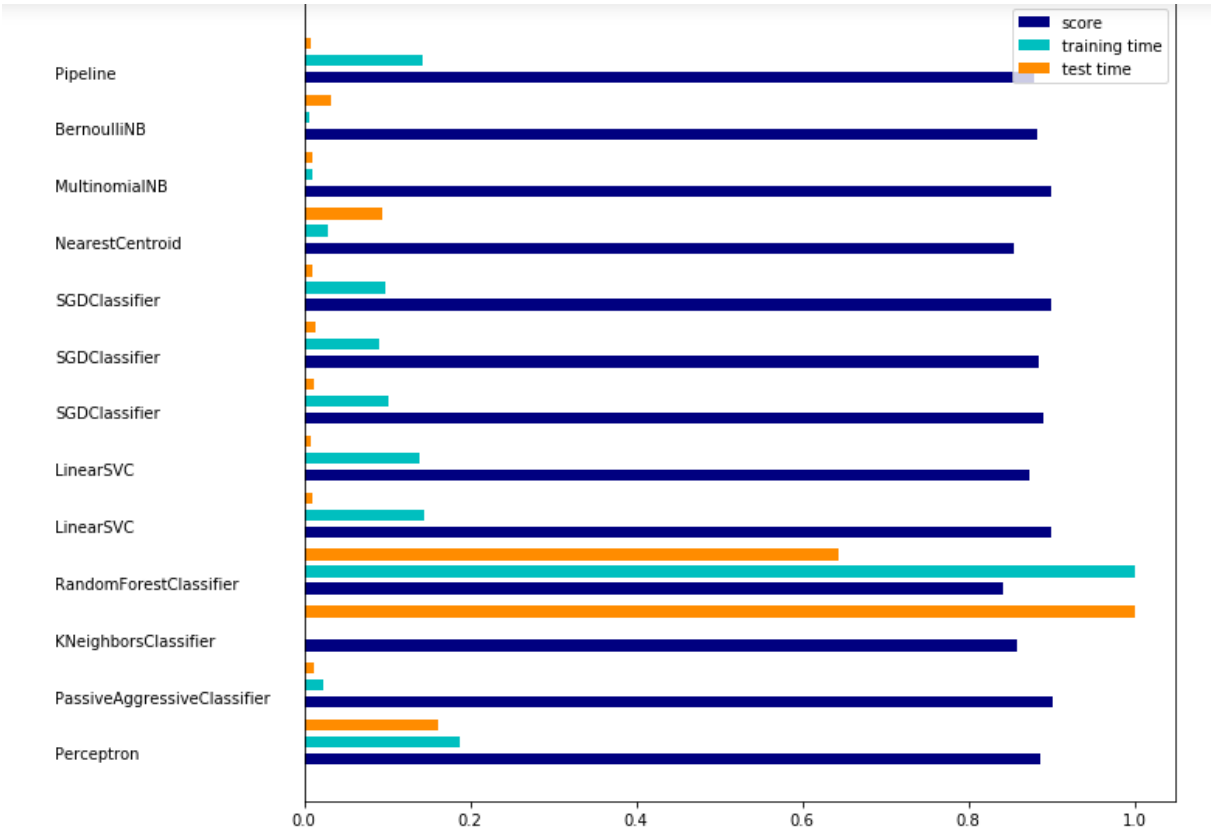
Modelling:

```
In [6]: from sklearn.naive_bayes import MultinomialNB
        from sklearn import metrics

In [7]: newsgroups_test = fetch_20newsgroups(subset='test',
                                             categories=categories)
        vectors_test = vectorizer.transform(newsgroups_test.data)

In [8]: clf = MultinomialNB(alpha=.01)

In [9]: clf.fit(vectors, newsgroups_train.target)
        pred = clf.predict(vectors_test)
```

Evaluation:

```
In [10]: metrics.f1_score(newsgroups_test.target, pred, average='macro')

Out[10]: 0.8821359240272957
```

f) We can measure whether model was able to classify test data properly or not by using different metrics. I am taking example of visualization from sklearn website[2].



In this example score is accuracy. Accuracy for multinomial naïve bayes came out to be around 89%.

```
In [13]: metrics.accuracy_score(newsgroups_test.target, pred)
Out[13]: 0.893569844789357
```

[1] https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

[2] https://scikit-learn.org/0.19/auto_examples/text/document_classification_20newsgroups.html#