Project Machine Learning For Big Data

Pranay Gheewala - B00826923

Karan Thakkar - B00823819

In [75]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import zipfile
import numpy as np
import os
import pandas as pd
import string
import xml.etree.ElementTree as et
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.metrics.cluster import homogeneity_score
from sklearn.metrics.cluster import completeness_score
from sklearn.metrics.cluster import v_measure_score
from sklearn.metrics.cluster import adjusted_rand_score
import matplotlib.cm as cm
from keras.layers import Input, Dense
from keras.models import Model
from keras.models import Sequential
from keras.callbacks import EarlyStopping
import nltk
from nltk.stem import PorterStemmer
import string
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
from sklearn.metrics import davies_bouldin_score
from keras import utils
```

Extracting Data from the folder. We have used all the data provided for the project.

In [76]:
```
#Data extraction path
directory = 'C:\\Users\\Pranay\\Desktop\\dalhousie\\Machine Learning for Big Data\\Project\\Data\\Data'

df_cols = ["itemid","headline","text","date","XMLfilename","bip"]
rows = []

# function for extracting data from xml file
def createDataFrame (xml_file):

    xtree = et.parse(directory+'\\'+filename+'\\'+xml_file)
    xroot = xtree.getroot()
    itemid= xroot.attrib.get("itemid")

    for headline in xroot.iter('headline'):
        news_headline = headline.text

    text=xroot.find("text")
    textData = ''
    for pTag in text.findall('./p'):
        textData = textData + pTag.text

    dateData=''
    date=xroot.find("metadata")
    for dateTag in date.findall('./dc'):
        if dateTag.attrib.get("element")=='dc.date.published':
            dateData = dateTag.attrib.get("value")

    bipData=''
    bip=xroot.find("metadata")
    for bipTag in bip.findall('./codes'):
        if bipTag.attrib.get("class")=='bip:topics:1.0':
            for bipcode in bipTag.findall('./code'):
                bipData = bipData+ bipcode.attrib.get("code")
                break

    rows.append({"itemid": itemid,"headline": news_headline,"text" : textData, "date" : dateData, "XMLfilename"

    return rows

for filename in os.listdir(directory):

    for name in os.listdir(directory+'\\'+filename):
```

```
        if name.endswith('.xml'):

                row_data=createDataFrame(name)

doc_df = pd.DataFrame(row_data,columns = df_cols) #creating dataframe
doc_df.head()
```

Out[76]:

	itemid	headline	text	date	XMLfilename	bip
0	432107	Detroit gunman killed after he kills 3, wounds 2.	Reciting the Lord's Prayer while he fired a sh...	1997-03-11	432107newsML.xml	GCAT
1	432108	COMMISSION ADOPTS MEMORANDUM ON WORKERS RIGHTS...	The Commission, in paragraph 11.3.2 of the Med...	1997-03-11	432108newsML.xml	E41
2	432109	REUTER EC REPORT LONG-TERM DIARY (PAGE 6/10 - ...	EC Report Long-Term Diary (page 7/10 - June 11...	1997-03-11	432109newsML.xml	G15
3	432110	REUTER EC REPORT LONG-TERM DIARY (PAGE 2/10 - ...	EC Report Long-Term Diary (page 4/10 - Apr 24-...	1997-03-11	432110newsML.xml	G15
4	432111	REUTER EC REPORT LONG-TERM DIARY FOR MARCH 17-...	****HIGHLIGHTS****AMSTERDAM - The Netherlands ...	1997-03-11	432111newsML.xml	G15

Preprocessing Text

1.) Cleaning Data. Removing punctuation, digits and special characters.

In [77]:
```
# function to clean data using regular expression
def cleanData (dataframe):
    # removing punctuation
    dataframe['text']=dataframe['text'].str.replace('[{}]'.format(string.punctuation), '')

    #converting all text data to lower case
    dataframe["text"] = dataframe["text"].str.lower()

    #removing digits from text data
    dataframe["text"] = dataframe["text"].str.replace(r"[^a-z]+"," ")

    return dataframe

cleanData(doc_df)
```

Out[77]:

	itemid	headline	text	date	XMLfilename	bip
0	432107	Detroit gunman killed after he kills 3, wounds 2.	reciting the lords prayer while he fired a sho...	1997-03-11	432107newsML.xml	GCAT
1	432108	COMMISSION ADOPTS MEMORANDUM ON WORKERS RIGHTS...	the commission in paragraph of the mediumterm ...	1997-03-11	432108newsML.xml	E41
2	432109	REUTER EC REPORT LONG-TERM DIARY (PAGE 6/10 - ...	ec report longterm diary page june wednesday m...	1997-03-11	432109newsML.xml	G15
3	432110	REUTER EC REPORT LONG-TERM DIARY (PAGE 2/10 - ...	ec report longterm diary page apr may thursday...	1997-03-11	432110newsML.xml	G15
4	432111	REUTER EC REPORT LONG-TERM DIARY FOR MARCH 17-...	highlightsamsterdam the netherlands hosts summ...	1997-03-11	432111newsML.xml	G15
5	432112	OFFICIAL JOURNAL CONTENTS - OJ C 75 OF MARCH 1...	note contents are displayed in reverse order t...	1997-03-11	432112newsML.xml	G15
6	432113	OFFICIAL JOURNAL CONTENTS - OJ C 76 OF MARCH 1...	note contents are displayed in reverse order t...	1997-03-11	432113newsML.xml	G15

2.) Removing Stop Words

In [78]:
```
# function to remove stopwords
def removeStopWords(dataframe):

    dataframe['text']=dataframe['text'].str.split()
    stop_words = stopwords.words('english')
    dataframe['text']=dataframe['text'].apply(lambda x: [item for item in x if item not in stop_words])
    return dataframe

removeStopWords(doc_df)
```

45668	477872	Uzbek President Karimov to visit Greece.	[uzbek, president, islam, karimov, leaves, mon...	1997-03-31	477872newsML.xml	GDIP
45669	477873	St Jude to reduce Ventritex price - Journal.	[st, jude, medical, inc, expected, announce, a...	1997-03-31	477873newsML.xml	C31
45670	477874	Journal.	[bermudabased, partnerre, ltd, set, announce, ...	1997-03-31	477874newsML.xml	C33
45671	477875	India retains import curbs on consumer electro...	[indias, commerce, ministry, said, monday, exp...	1997-03-31	477875newsML.xml	E512
45672	477876	India rupee recovers, shrugs off political wor...	[indian, rupee, held, steady, dollar, monday, ...	1997-03-31	477876newsML.xml	M13
45673	477877	Bulgaria seeks advisers for sale of state firms.	[bulgarian, government, seeking, financial, ad...	1997-03-31	477877newsML.xml	C183
45674	477878	INTERVIEW-Lion Teck Chiang sees net up 70 pct ...	[property, steel, group, lion, teck, chiang, l...	1997-03-31	477878newsML.xml	C152
45675	477879	World Bank says Bangladesh must open up economy.	[world, bank, asked, bangladesh, reduce, impor...	1997-03-31	477879newsML.xml	E11
45676	477880	Disney Online to charge for kid website - Times.	[walt, disney, co, plans, charge, per, month, ...	1997-03-31	477880newsML.xml	

3.) Applying Stemmer to reduce words to their root word or to chop off the derivational affixes.

In [79]:
```
#funtion to apply stemming to text data
def applyStemming(dataframe):
    ps = PorterStemmer()
    dataframe["text"] = dataframe["text"].apply(lambda x: [ps.stem(y) for y in x])
    return dataframe

applyStemming(doc_df)
```

Out[79]:

	itemid	headline	text	date	XMLfilename	bip
0	432107	Detroit gunman killed after he kills 3, wounds 2.	[recit, lord, prayer, fire, shotgun, man, dres...	1997-03-11	432107newsML.xml	GCAT
1	432108	COMMISSION ADOPTS MEMORANDUM ON WORKERS RIGHTS...	[commiss, paragraph, mediumterm, social, actio...	1997-03-11	432108newsML.xml	E41
2	432109	REUTER EC REPORT LONG-TERM DIARY (PAGE 6/10 - ...	[ec, report, longterm, diari, page, june, wedn...	1997-03-11	432109newsML.xml	G15
3	432110	REUTER EC REPORT LONG-TERM DIARY (PAGE 2/10 - ...	[ec, report, longterm, diari, page, apr, may, ...	1997-03-11	432110newsML.xml	G15
4	432111	REUTER EC REPORT LONG-TERM DIARY FOR MARCH 17-...	[highlightsamsterdam, netherland, host, summit...	1997-03-11	432111newsML.xml	G15
5	432112	OFFICIAL JOURNAL CONTENTS - OJ C 75 OF MARCH 1...	[note, content, display, revers, order, print,...	1997-03-11	432112newsML.xml	G15
6	432113	OFFICIAL JOURNAL CONTENTS - OJ C 76 OF MARCH 1...	[note, content, display, revers, order, print,...	1997-03-11	432113newsML.xml	G15

4.) Applying Lemmatizer to reduce words to their base word, which is linguistically correct lemmas.

In [80]:
```
#Lemmatizing the dataset to reduce words to their base word, which is linguistically correct lemmas


#function to apply  lemmatizing to the text data
def applyLemmatizing(dataframe):
    lemmatize_data = WordNetLemmatizer()
    dataframe["text"] = dataframe["text"].apply(lambda a: [lemmatize_data.lemmatize(word) for word in a])
    return dataframe

applyLemmatizing(doc_df)
```

Out[80]:

	itemid	headline	text	date	XMLfilename	bip
0	432107	Detroit gunman killed after he kills 3, wounds 2.	[recit, lord, prayer, fire, shotgun, man, dres...	1997-03-11	432107newsML.xml	GCAT
1	432108	COMMISSION ADOPTS MEMORANDUM ON WORKERS RIGHTS...	[commiss, paragraph, mediumterm, social, actio...	1997-03-11	432108newsML.xml	E41
2	432109	REUTER EC REPORT LONG-TERM DIARY (PAGE 6/10 - ...	[ec, report, longterm, diari, page, june, wedn...	1997-03-11	432109newsML.xml	G15
3	432110	REUTER EC REPORT LONG-TERM DIARY (PAGE 2/10 - ...	[ec, report, longterm, diari, page, apr, may, ...	1997-03-11	432110newsML.xml	G15
4	432111	REUTER EC REPORT LONG-TERM DIARY FOR MARCH 17-...	[highlightsamsterdam, netherland, host, summit...	1997-03-11	432111newsML.xml	G15
5	432112	OFFICIAL JOURNAL CONTENTS - OJ C 75 OF MARCH 1...	[note, content, display, revers, order, print,...	1997-03-11	432112newsML.xml	G15
6	432113	OFFICIAL JOURNAL CONTENTS - OJ C 76 OF MARCH 1...	[note, content, display, revers, order, print,...	1997-03-11	432113newsML.xml	G15
7	432114	Canada T-bill auction averages, historical rates.	[mo, mo, yr, brate, c, yr, dlr, tse, note, feb...	1997-03-11	432114newsML.xml	M12
8	432115	Canadian key economic indicators.	[indic, period, latest, prev, prev, nextunempl...	1997-03-11	432115newsML.xml	E71
9	432116	RESEARCH ALERT - Goldman starts Canadian oils.	[goldman, sach, said, initi, coverag, suncor, ...	1997-03-11	432116newsML.xml	C15
10	432117	Petro-Canada lowers heavy oil price.	[compani, eff, date, bow, river, hardisti, med...	1997-03-11	432117newsML.xml	M14
11	432118	Petro-Canada lowers light oil prices.	[compani, eff, date, lt, sweet, lt, sour, cdlr...	1997-03-11	432118newsML.xml	M14

	itemid	headline	text	date	XMLfilename	bip
12	432119	Suncor cuts Canada posted oil prices.	[suncor, inc, said, lower, price, would, pay, ...	1997-03-11	432119newsML.xml	M14
13	432120	Canada agency sets framework for TV competition.	[canada, telecommun, regul, said, tuesday, put...	1997-03-11	432120newsML.xml	C13
14	432121	Dorel Industries Inc Q4 profit rises.	[shr, c, c, net, rev, month, shr, c, c, net, r...	1997-03-11	432121newsML.xml	C15
15	432122	Wascana to consider Talisman meeting.	[wascana, energi, inc, earlier, reject, talism...	1997-03-11	432122newsML.xml	C18
16	432123	CBRS reaffirms Gulf after Clyde deal.	[canadian, bond, rate, servic, said, tuesday, ...	1997-03-11	432123newsML.xml	C17
17	432124	Bank of Canada 91-day T-bill yields avg 2.993 ...	[bank, canada, said, averag, yield, week, auct...	1997-03-11	432124newsML.xml	M13
18	432125	Canada Jan new motor vehicle sales fall 6.3 pct.	[new, motor, vehicl, sale, fell, percent, seas...	1997-03-11	432125newsML.xml	C31
19	432126	Wascana Energy sets rights separation date.	[wascana, energi, inc, target, c, billion, tak...	1997-03-11	432126newsML.xml	C17
20	432127	U.S. Cattle/Beef Daily Summary.	[midwest, high, plain, direct, slaughter, catt...	1997-03-11	432127newsML.xml	M14
21	432128	KinderCare to relocate to Portland, Oregon.	[kindercar, learn, center, inc, said, tuesday,...	1997-03-11	432128newsML.xml	C11
22	432129	CONSENSUS: U.S. corporate earnings surprise data.	[earn, surpris, march, provid, zack, invest, r...	1997-03-11	432129newsML.xml	C15
23	432130	Seaboard Q4 shr profit vs loss.	[shr, profit, loss, net, profit, loss, sale, m...	1997-03-11	432130newsML.xml	C15
24	432131	FULL TEXT - Checkpoint, Ultrak to merge.	[checkpoint, system, inc, ultrak, inc, tuesday...	1997-03-11	432131newsML.xml	C18
25	432132	RESEARCH ALERT - IPALCO Enterprises estimates ...	[everen, secur, inc, said, tuesday, rais, earn...	1997-03-11	432132newsML.xml	C15
26	432133	RESEARCH ALERT - CPFL downgraded to hold.	[credit, lyonnai, secur, said, report, downgra...	1997-03-11	432133newsML.xml	C15
27	432134	RESEARCH ALERT - Emmis Broadcast started.	[ub, secur, said, initi, coverag, emmi, broadc...	1997-03-11	432134newsML.xml	C15
28	432135	RESEARCH ALERT - St. Paul Bancorp downgraded.	[everen, secur, inc, said, tuesday, downgrad, ...	1997-03-11	432135newsML.xml	C15

	itemid	headline	text	date	XMLfilename	bip
29	432136	RESEARCH ALERT - Nike strong buy repeated.	[salomon, brother, said, analyst, brett, barak...	1997-03-11	432136newsML.xml	C15
...	...	...	...	...	...	...
45652	477856	PRESS DIGEST - Cyprus - March 31.	[lead, stori, greek, cypriot, press, monday, r...	1997-03-31	477856newsML.xml	GCAT
45653	477857	PRESS DIGEST - Israel - March 31.	[lead, stori, isra, newspap, monday, reuter, v...	1997-03-31	477857newsML.xml	GCAT
45654	477858	Iran's Rafsanjani not decided on Saudi haj visit.	[iranian, presid, akbar, hashemi, rafsanjani, ...	1997-03-31	477858newsML.xml	GCAT
45655	477859	FEATURE-Egyptian women suffer violence in sile...	[marlen, tadro, veteran, egyptian, human, righ...	1997-03-31	477859newsML.xml	GCAT
45656	477860	Bank of Portugal injects 4.0 bln esc, 6.5 pct.	[bank, portug, said, inject, billion, escudo, ...	1997-03-31	477860newsML.xml	M13
45657	477861	Greek 1M ATHIBOR rose to 10.90 from 10.72pct.	[athen, interbank, offer, rate, athibor, one, ...	1997-03-31	477861newsML.xml	M12
45658	477862	Greek 1M ATHIBOR edges up to 10.72 from 10.72pct.	[athen, interbank, offer, rate, athibor, one, ...	1997-03-31	477862newsML.xml	M12
45659	477863	PRESS DIGEST - Portugal -- March 31.	[follow, lead, domest, stori, portugues, newsp...	1997-03-31	477863newsML.xml	GCAT
45660	477864	PRESS DIGEST - Malta - March 31.	[follow, lead, stori, maltes, press, monday, r...	1997-03-31	477864newsML.xml	GCAT
45661	477865	Bank/Portugal sets 10 bln esc variable rate repo.	[bank, portug, offer, inject, billion, escudo,...	1997-03-31	477865newsML.xml	M13
45662	477866	Yield rises on Portuguese 182-day bills.	[averag, yield, rose, percent, percent, previo...	1997-03-31	477866newsML.xml	M13
45663	477867	Table of Greek mutual fund flows, assets.	[ionian, bank, report, follow, daili, yeartod,...	1997-03-31	477867newsML.xml	C15
45664	477868	FEATURE - Valencia sets itself ablaze in fiery...	[deafen, blast, thunder, street, burn, cinder,...	1997-03-31	477868newsML.xml	GCAT
45665	477869	FEATURE - Art show recalls happier Franco-Belg...	[shoot, frenchman, goe, old, belgian, joke, ai...	1997-03-31	477869newsML.xml	GCAT
45666	477870	NEW YORK, March 31 (Reuter) - Wang Laboratorie.	[wang, laboratori, gave, chairman, chief, exec...	1997-03-31	477870newsML.xml	C171
45667	477871	Hearst's Ganzi seen in line for CEO post - WSJ.	[hearst, corp, name, victor, ganzi, execut, vi...	1997-03-31	477871newsML.xml	

	itemid	headline	text	date	XMLfilename	bip
45668	477872	Uzbek President Karimov to visit Greece.	[uzbek, presid, islam, karimov, leav, monday, ...	1997-03-31	477872newsML.xml	GDIP
45669	477873	St Jude to reduce Ventritex price - Journal.	[st, jude, medic, inc, expect, announc, agreem...	1997-03-31	477873newsML.xml	C31
45670	477874	Journal.	[bermudabas, partnerr, ltd, set, announc, mond...	1997-03-31	477874newsML.xml	C33
45671	477875	India retains import curbs on consumer electro...	[india, commerc, ministri, said, monday, expor...	1997-03-31	477875newsML.xml	E512
45672	477876	India rupee recovers, shrugs off political wor...	[indian, rupe, held, steadi, dollar, monday, b...	1997-03-31	477876newsML.xml	M13
45673	477877	Bulgaria seeks advisers for sale of state firms.	[bulgarian, govern, seek, financi, advis, plan...	1997-03-31	477877newsML.xml	C183
45674	477878	INTERVIEW-Lion Teck Chiang sees net up 70 pct ...	[properti, steel, group, lion, teck, chiang, l...	1997-03-31	477878newsML.xml	C152
45675	477879	World Bank says Bangladesh must open up economy.	[world, bank, ask, bangladesh, reduc, import, ...	1997-03-31	477879newsML.xml	E11
45676	477880	Disney Online to charge for kid website - Times.	[walt, disney, co, plan, charg, per, month, ac...	1997-03-31	477880newsML.xml	
45677	477881	U.S. to back fewer supercomputer centers - Times.	[nation, scienc, foundat, plan, reduc, number,...	1997-03-31	477881newsML.xml	
45678	477882	Indian shares plunge 8.6 pct on political crisis.	[indian, share, plung, eight, percent, panicki...	1997-03-31	477882newsML.xml	M11
45679	477883	Singapore shares open weak, funds stay sidelined.	[singapor, share, open, weaker, monday, fund, ...	1997-03-31	477883newsML.xml	M11
45680	477884	Selecta declares two centavo cash dividend.	[selecta, dairi, product, inc, declar, monday,...	1997-03-31	477884newsML.xml	C151
45681	477885	Tokyo stocks down in thin morning trade.	[tokyo, stock, fell, monday, morn, last, day, ...	1997-03-31	477885newsML.xml	M11

45682 rows × 6 columns

Extracting Features Using TfidfVectorizer

We have used TfidfVectorizer as we researched about it and found out that it helps to process text data and set up the vector space model in order to provide a convenient data structure for text categorization [1]. Also according to this paper using TFIDF-based feature selection approach to improve its accuracy [1].

In [81]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import preprocessing


textlist=[]

#function to extract features and labels
def featureslabelsExtraction_1(dataframe):

    new_df = dataframe
    new_df=new_df.drop(["itemid","headline","date","XMLfilename"], axis=1) #generating new dataframe consisting


    vectorizer = TfidfVectorizer(
    min_df = 0.01,
    max_df = 0.8,
    stop_words = 'english')

    for text in new_df['text']:
        text = ' '.join(text)
        textlist.append(text)

    X_train_tfidf = vectorizer.fit_transform(textlist) #extracting features and return term-document matrix.

    return X_train_tfidf,vectorizer

X_train_tfidf,vectorizer=featureslabelsExtraction_1(doc_df)
```

Finding Optimal Value For k(number of clusters to be formed)

1.) Elbow Graph

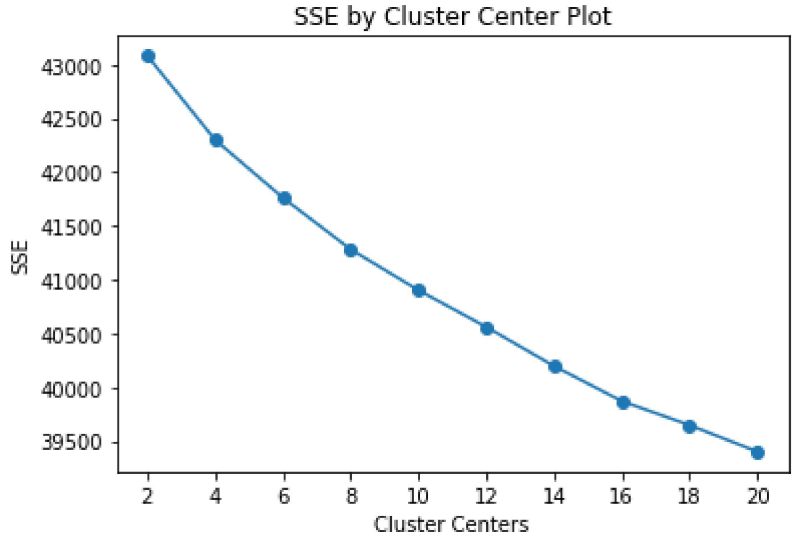
We first tried plotting elbow to find optimal value for k but as it can be seen from the graph, it is very ambigous and it is difficult to select optimal value for k from this graph as there is no clear and sharp elbow in the graph that can be considered.

In [62]:
```
def calculateSSE():
    sse = []
    max_k = 20
    k_rng = range(2, max_k+1, 2)
    for k in k_rng:
        km = KMeans(n_clusters=k,init='k-means++')
        km.fit(X_train_tfidf)
        sse.append(km.inertia_)

    return sse
```

In [63]:
```
sse = calculateSSE()
```

In [66]:
```
def plotElbowGraph(sse):
    max_k = 20
    k_rng = range(2, max_k+1, 2)
    f, ax = plt.subplots(1, 1)
    ax.plot(k_rng, sse, marker='o')
    ax.set_xlabel('Cluster Centers')
    ax.set_xticks(k_rng)
    ax.set_xticklabels(k_rng)
    ax.set_ylabel('SSE')
    ax.set_title('SSE by Cluster Center Plot')
```

In [67]:
```
plotElbowGraph(sse)
```



2.) Silhouette Graph

We researched to find the alternative for the elbow graph in order to find visual cluster valuation. We found out that Silhouette Graph allows an appreciation of the relative quality of the clusters and an overview of the data configuration [2]. The average silhouette width provides an evaluation of clustering validity, and might be used to select an 'appropriate' number of clusters [2].

Silhouette analysis can be used to study the separation distance between the resulting clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like number of clusters visually. Silhouette coefficients near +1 indicate that the sample is far away from the neighboring clusters. A

value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster [3].

In [106]:
```
range_n_clusters = [2,4,6,8,10,12,14,16,18,20]
```

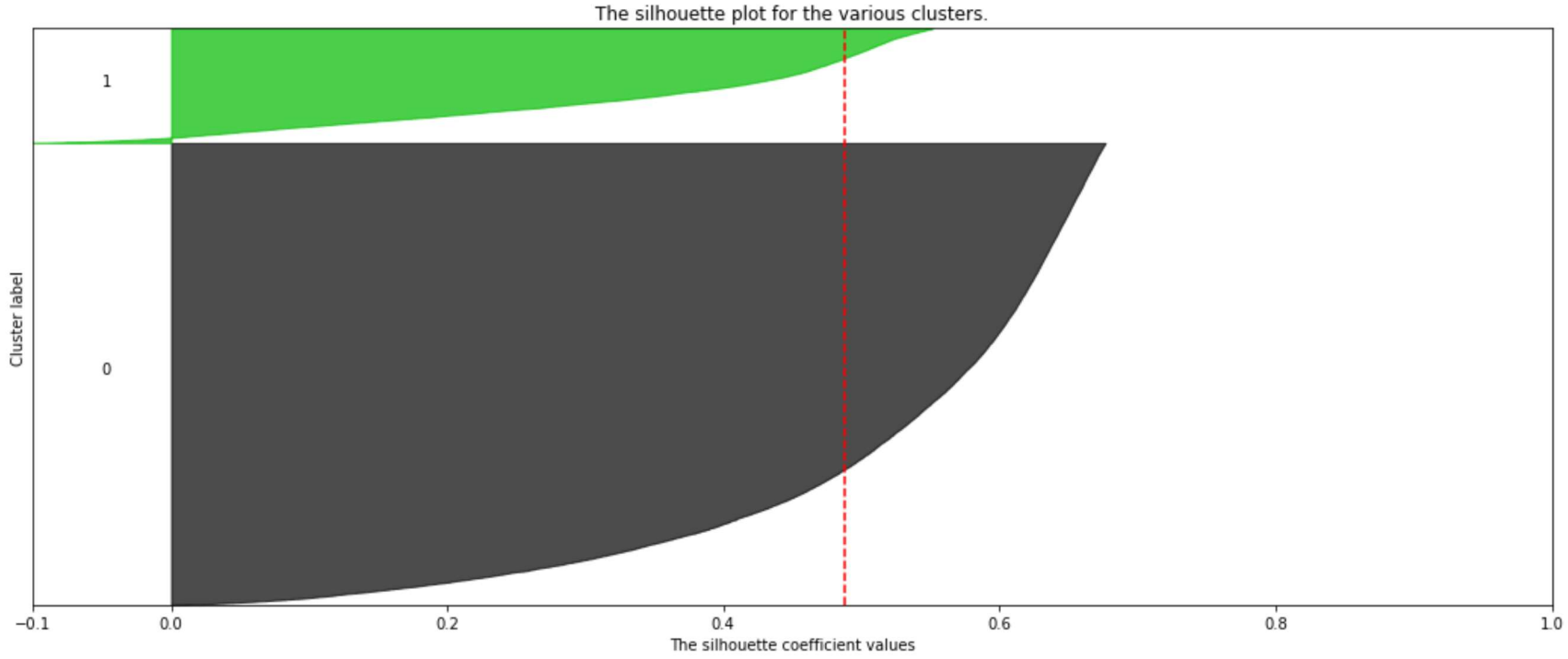
In [73]:
```
# function to plot silhouette graph to find optimal number for cluster formation
def plotSilhouetteGraph():
    for n_clusters in range_n_clusters:
        fig, (ax1) = plt.subplots(1, 1)
        fig.set_size_inches(18, 7)
        ax1.set_xlim([-0.1, 1])
        ax1.set_ylim([0, len(X_train_tfidf) + (n_clusters + 1) * 10])
        clusterer = KMeans(n_clusters=n_clusters, random_state=10,init='k-means++')
        cluster_labels = clusterer.fit_predict(X_train_tfidf)
        silhouette_avg = silhouette_score(X_train_tfidf, cluster_labels)
        print("For n_clusters =", n_clusters,
              "The average silhouette_score is :", silhouette_avg)
        sample_silhouette_values = silhouette_samples(X_train_tfidf, cluster_labels)
        y_lower = 10
        for i in range(n_clusters):
            ith_cluster_silhouette_values = \
                sample_silhouette_values[cluster_labels == i]

            ith_cluster_silhouette_values.sort()

            size_cluster_i = ith_cluster_silhouette_values.shape[0]
            y_upper = y_lower + size_cluster_i

            color = cm.nipy_spectral(float(i) / n_clusters)
            ax1.fill_betweenx(np.arange(y_lower, y_upper),
                              0, ith_cluster_silhouette_values,
                              facecolor=color, edgecolor=color, alpha=0.7)
            ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
            y_lower = y_upper + 10
        ax1.set_title("The silhouette plot for the various clusters.")
        ax1.set_xlabel("The silhouette coefficient values")
        ax1.set_ylabel("Cluster label")
        ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

        ax1.set_yticks([])  # Clear the yaxis labels / ticks
        ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    plt.show()
```

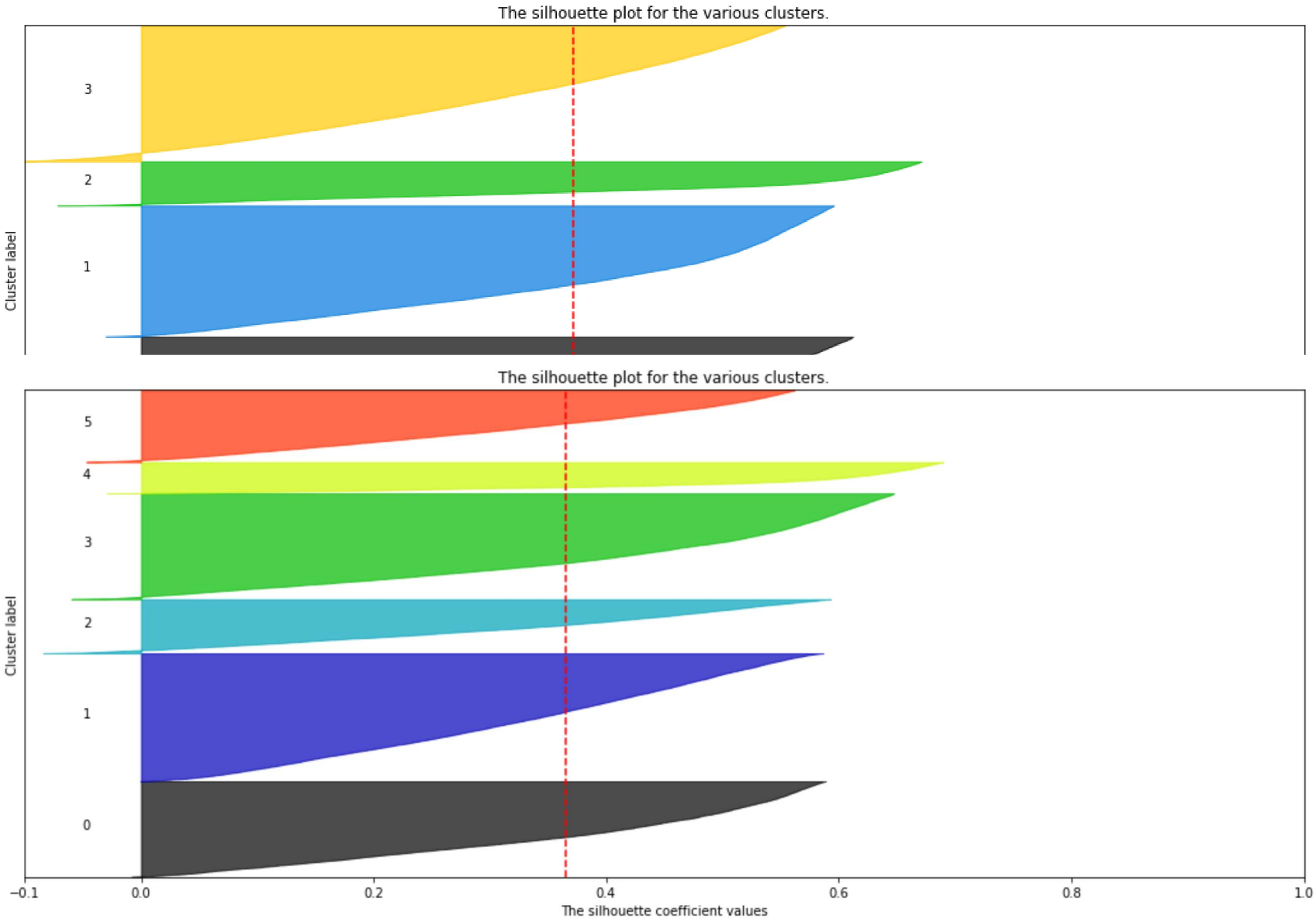
As it can be seen from the silhouette score by forming 10 clusters of data gives the highest silhouetter score which indicates the clusters are well formed than other number of clusters and the samples in each cluster are far away from other clusters. Moreover it can be visualize from the graph plot that the thickness of each cluster in n_cluster = 10 is almost uniformly distributed. Also every cluster formed has silhouette score greater than the mean silhouette score which can be observed from the red dotted line.

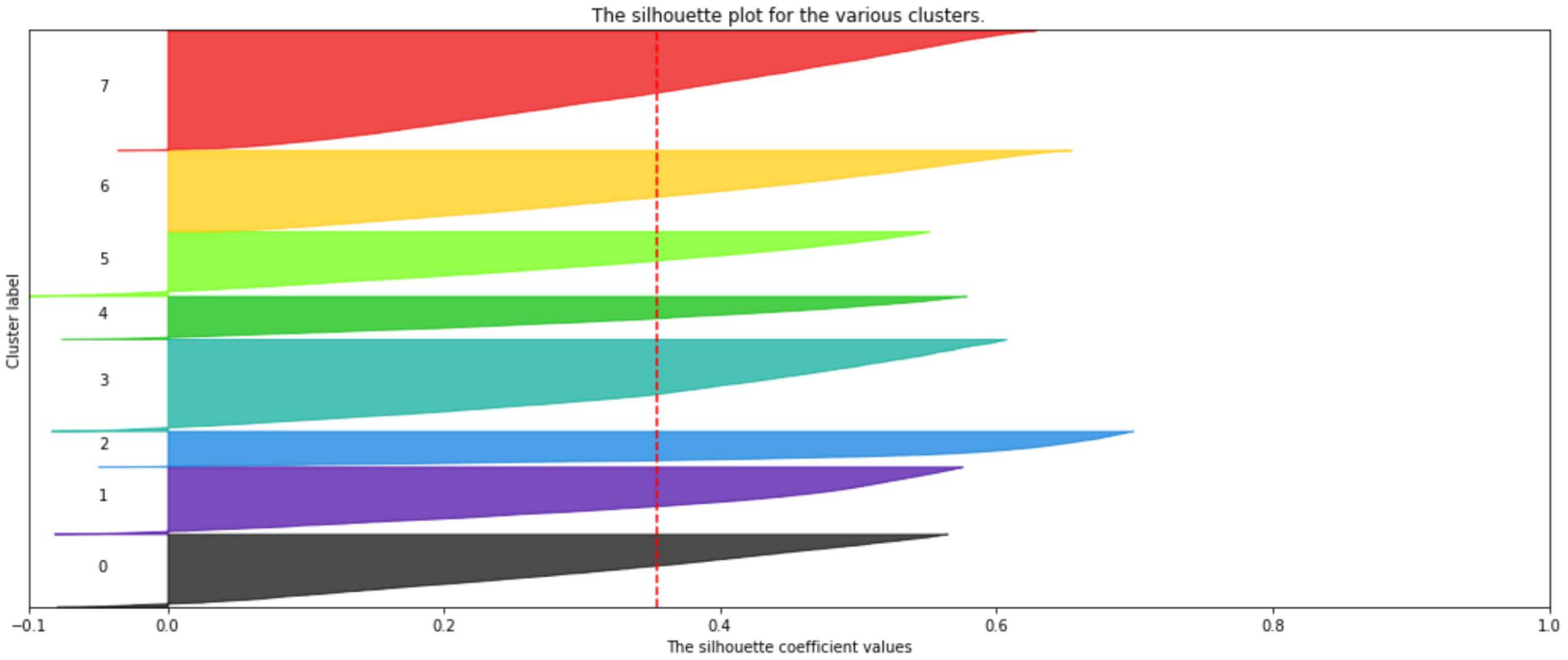
Thus we decided to have 10 number of clusters that should be formed for our data.

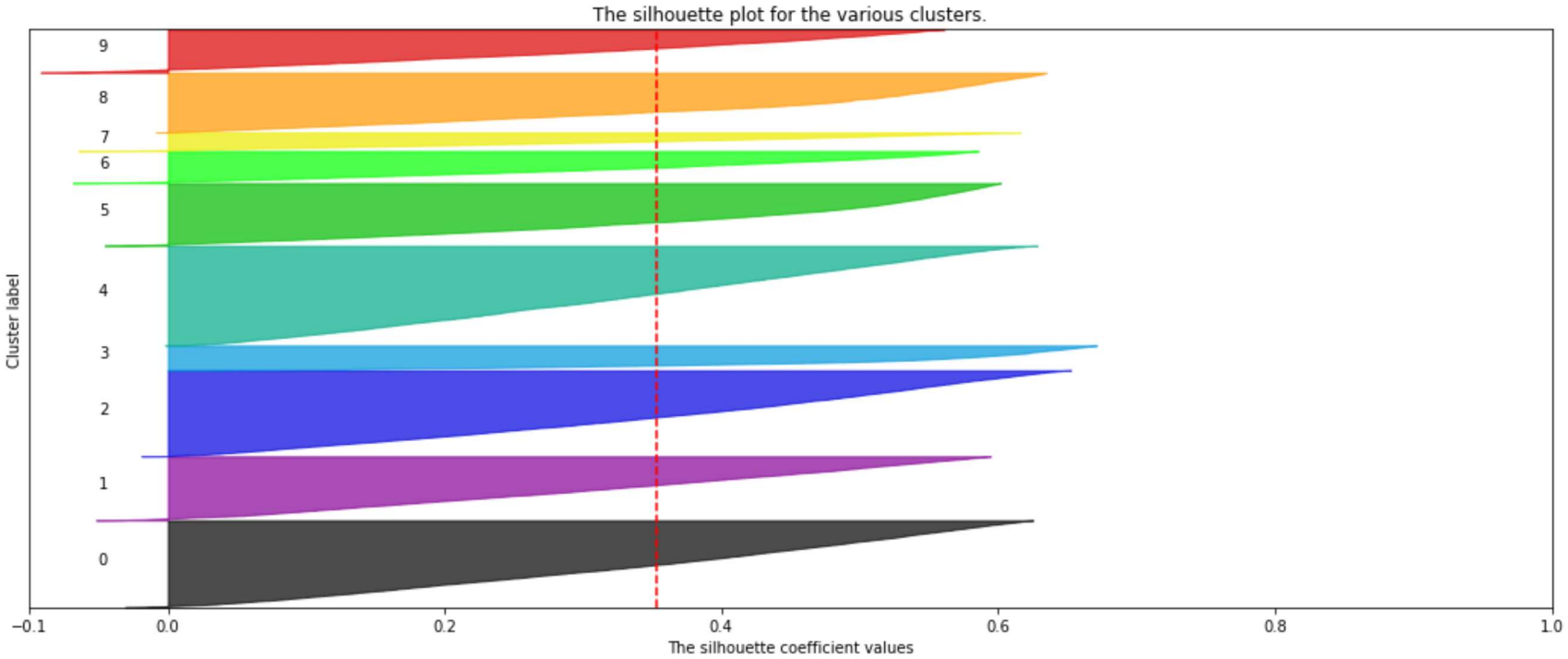
In [74]:
```
plotSilhouetteGraph()
```

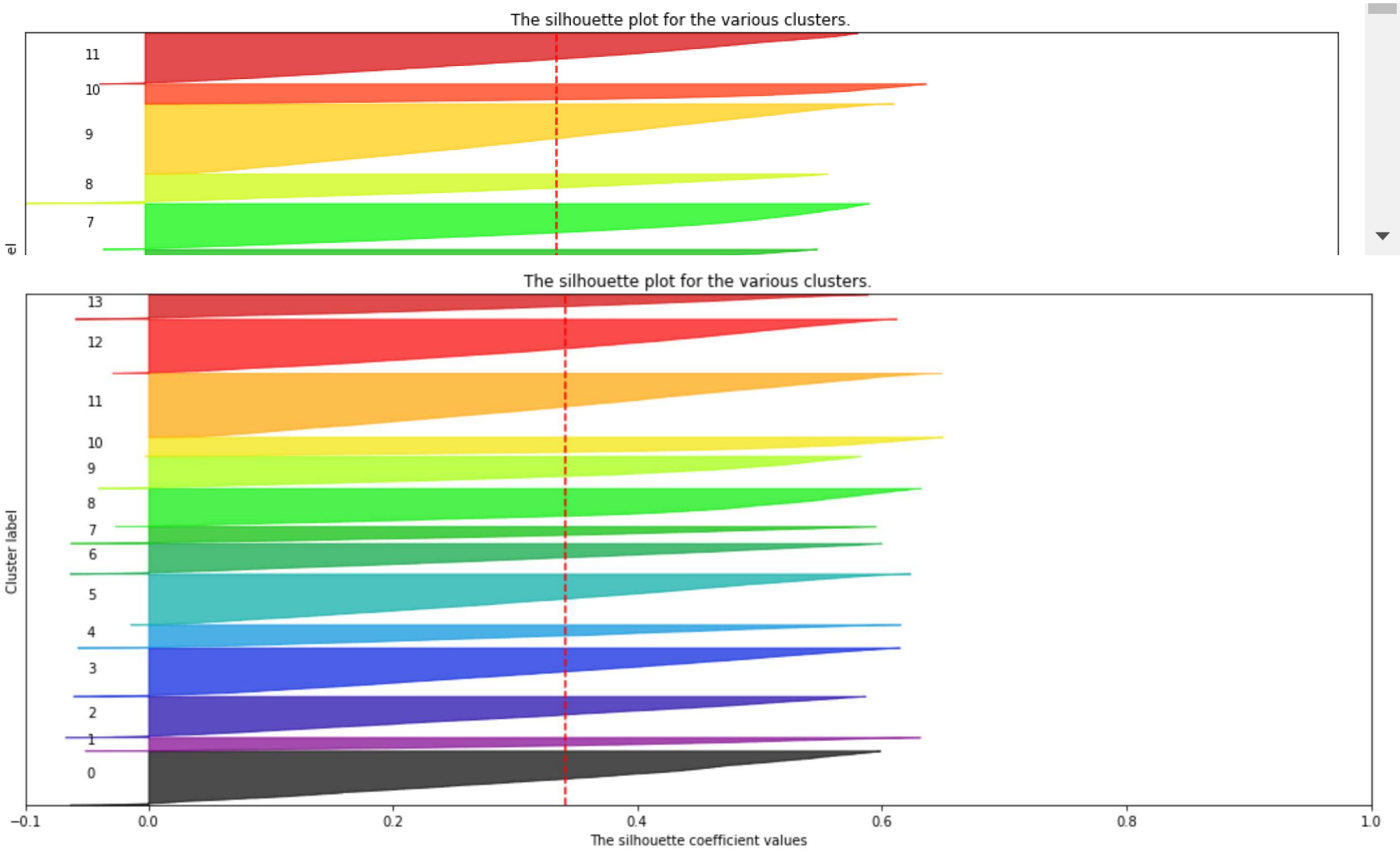
```
For n_clusters = 2 The average silhouette_score is : 0.35362674987598636
For n_clusters = 4 The average silhouette_score is : 0.3653895859118908
For n_clusters = 6 The average silhouette_score is : 0.37189850744137426
For n_clusters = 8 The average silhouette_score is : 0.4213590155488445
For n_clusters = 10 The average silhouette_score is : 0.4882782239819362
For n_clusters = 12 The average silhouette_score is : 0.3751566140887957
For n_clusters = 14 The average silhouette_score is : 0.35103845310238234
For n_clusters = 16 The average silhouette_score is : 0.3404261731014825
For n_clusters = 18 The average silhouette_score is : 0.3379405464954568
For n_clusters = 20 The average silhouette_score is : 0.3376210271746718
```

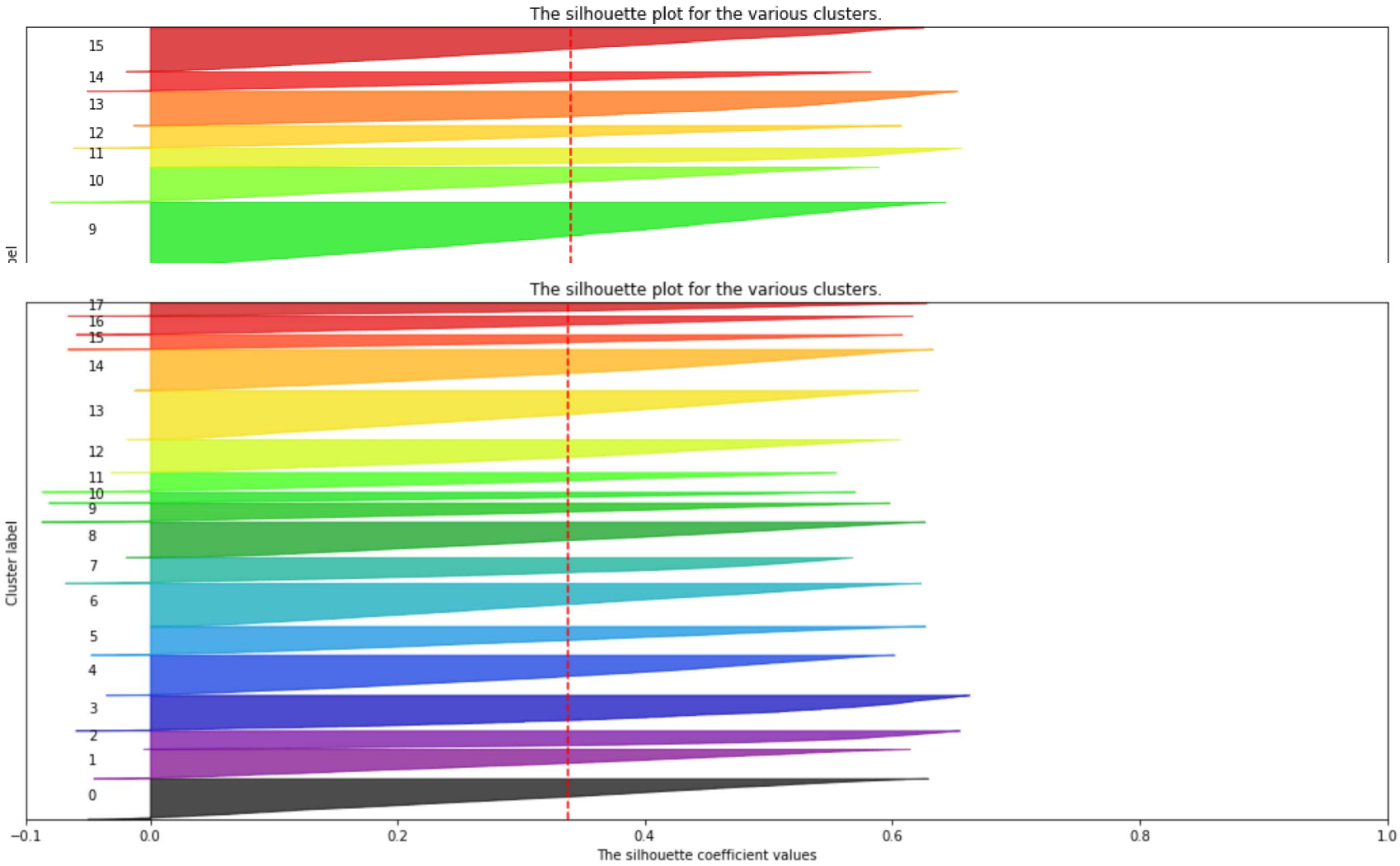


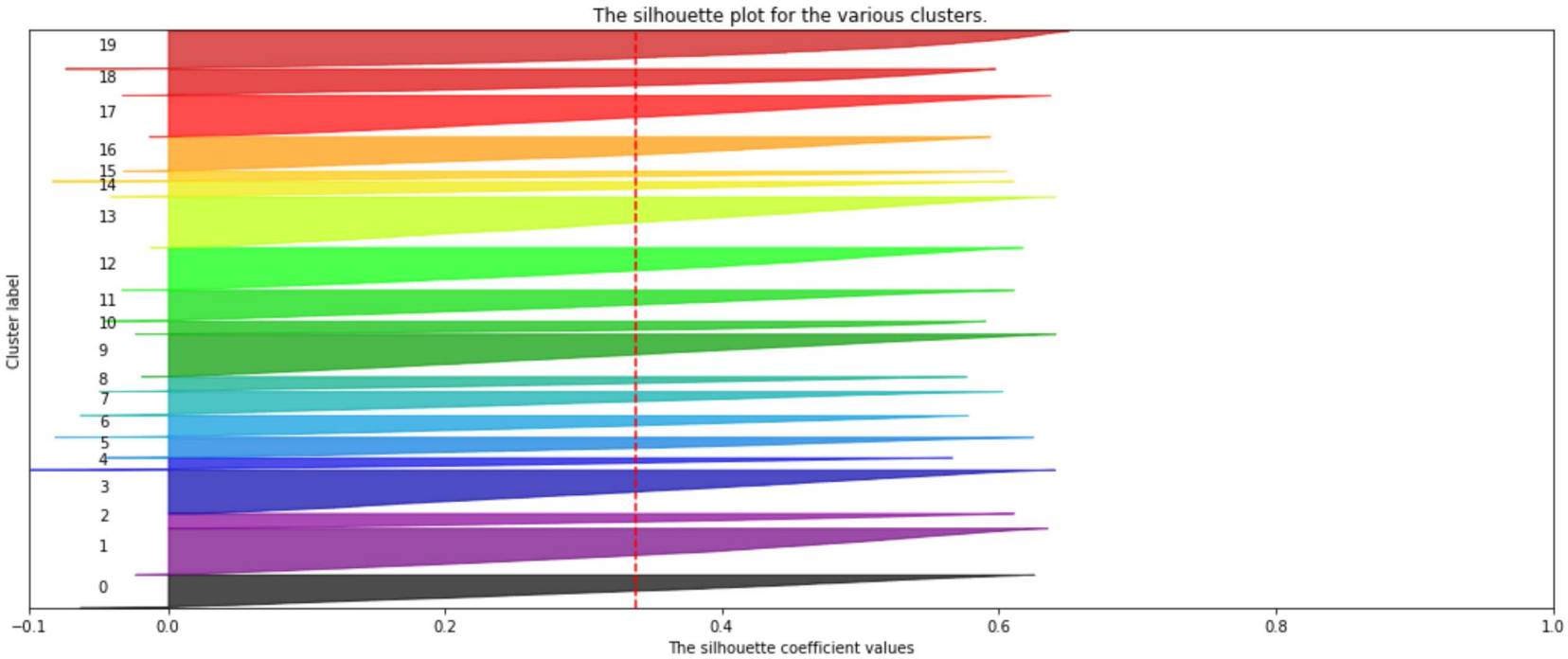

The silhouette plot for the various clusters.
Cluster label
3
2
1
The silhouette plot for the various clusters.
Cluster label
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The silhouette plot for the various clusters.
Cluster label
7
6
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The silhouette plot for the various clusters.
Cluster label
9
8
7
6
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The silhouette plot for the various clusters.
11
10
9
8
7
el
The silhouette plot for the various clusters.
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Cluster label
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The silhouette plot for the various clusters.
15
14
13
12
11
10
9
The silhouette plot for the various clusters.
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Cluster label
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The silhouette plot for the various clusters.
Cluster label
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values

K Means Algorithm to cluster the data with K-means++ to initialize the centroid

We have used K- means algorithm as it is fast and efficient and as the data is huge K-means will perform efficiently in compare to other algorithm. Also it tries to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares [4]. However K-means is highly dependent on the initialization of the centroids. As a result, the computation is often done several times, with different initializations of the centroids. So we needed a more effcient way to initialize the centroids rather than random initialization.

We researched and found that K-means++ solves this issue initializes the centroids to be (generally) distant from each other, leading to provably better than random initialization [5].

In [94]:
```
# function for creating model for K-means clustering using K-means++ centroid initialization
def clusterData(kcluster):
    number_of_cluster = kcluster
    model = KMeans(n_clusters=number_of_cluster, init='k-means++', max_iter=100, n_init=1)
    return model
```

In [95]:
```
clusteringModel=clusterData(10)
```

In [96]:
```
# Function to create clusters for documents
def predictClusterID(model,X_train_tfidf):
    y_predicted = model.fit_predict(X_train_tfidf)
    return y_predicted
```

In [97]:
```
# Assigning Cluster Id for each document to know which document belongs to which cluster
ClusterID = predictClusterID(clusteringModel,X_train_tfidf)
doc_df['cluster']=ClusterID
```

In [98]:
```
doc_df
```

Out[98]:

	itemid	headline	text	date	XMLfilename	bip	cluster
0	432107	Detroit gunman killed after he kills 3, wounds 2.	[recit, lord, prayer, fire, shotgun, man, dres...	1997-03-11	432107newsML.xml	GCAT	9
1	432108	COMMISSION ADOPTS MEMORANDUM ON WORKERS RIGHTS...	[commiss, paragraph, mediumterm, social, actio...	1997-03-11	432108newsML.xml	E41	7
2	432109	REUTER EC REPORT LONG-TERM DIARY (PAGE 6/10 - ...	[ec, report, longterm, diari, page, june, wedn...	1997-03-11	432109newsML.xml	G15	8
3	432110	REUTER EC REPORT LONG-TERM DIARY (PAGE 2/10 - ...	[ec, report, longterm, diari, page, apr, may, ...	1997-03-11	432110newsML.xml	G15	8
4	432111	REUTER EC REPORT LONG-TERM DIARY FOR MARCH 17-...	[highlightsamsterdam, netherland, host, summit...	1997-03-11	432111newsML.xml	G15	8
5	432112	OFFICIAL JOURNAL CONTENTS - OJ C 75 OF MARCH 1...	[note, content, display, revers, order, print,...	1997-03-11	432112newsML.xml	G15	8
6	432113	OFFICIAL JOURNAL CONTENTS - OJ C 76 OF MARCH 1...	[note, content, display, revers, order, print,...	1997-03-11	432113newsML.xml	G15	7

In [88]:
```
#Finding the top 10 terms of each cluster
def termsPerCluster(model,vectorizer):
    order_centroids = model.cluster_centers_.argsort()[:, ::-1]
    terms = vectorizer.get_feature_names()
    for i in range(10):
     print('\033[1m' + "Cluster %d: " % i + '\033[0m'),
     for ind in order_centroids[i, :10]:
         print('%s' % terms[ind])
```

As it can seen from the terms of each cluster that the clusters have been well formed. The words which relate to each other are grouped in one cluster as it is evident from the terms displayed.

In [89]:
```
termsPerCluster(clusteringModel,vectorizer)
```

```
Cluster 0:
match
play
cup
second
game
team
beat
win
leagu
score
Cluster 1:
bond
municip
coupon
date
moodi
sale
desk
issuer
price
sp
Cluster 2:
bank
rate
percent
said
market
dollar
central
billion
bond
dealer
Cluster 3:
share
stock
percent
said
index
market
```

```
trade
close
point
million
Cluster 4:
said
minist
govern
presid
state
offici
parti
peopl
countri
say
Cluster 5:
net
profit
incom
loss
shr
share
dividend
unless
oper
million
Cluster 6:
yen
latest
forecast
specifi
million
quarterli
prior
yearago
unless
actual
Cluster 7:
pct
price
lb
march
mar
```

```
feb
na
jan
week
percent
Cluster 8:
tonn
cent
trader
oil
price
said
wheat
export
market
crude
Cluster 9:
said
compani
million
percent
year
billion
group
new
product
sale
```

In [109]:
```
# Function to create dataframe for each individual cluster.
cluster_df_list = []
cluster_number=[0,1,2,3,4,5,6,7,8,9]
def createIndividualClusterDF(dataframe):
    for cluster_df in (cluster_number):
        cluster_df_list.append(dataframe[dataframe.cluster==cluster_df])

    len(cluster_df_list)
```

In [110]:
```
createIndividualClusterDF(doc_df)
```

In [111]:
```
doc_df
```

Out[111]:

	itemid	headline	text	date	XMLfilename	bip	cluster
0	432107	Detroit gunman killed after he kills 3, wounds 2.	[recit, lord, prayer, fire, shotgun, man, dres...	1997-03-11	432107newsML.xml	GCAT	9
1	432108	COMMISSION ADOPTS MEMORANDUM ON WORKERS RIGHTS...	[commiss, paragraph, mediumterm, social, actio...	1997-03-11	432108newsML.xml	E41	7
2	432109	REUTER EC REPORT LONG-TERM DIARY (PAGE 6/10 - ...	[ec, report, longterm, diari, page, june, wedn...	1997-03-11	432109newsML.xml	G15	8
3	432110	REUTER EC REPORT LONG-TERM DIARY (PAGE 2/10 - ...	[ec, report, longterm, diari, page, apr, may, ...	1997-03-11	432110newsML.xml	G15	8
4	432111	REUTER EC REPORT LONG-TERM DIARY FOR MARCH 17-...	[highlightsamsterdam, netherland, host, summit...	1997-03-11	432111newsML.xml	G15	8
5	432112	OFFICIAL JOURNAL CONTENTS - OJ C 75 OF MARCH 1...	[note, content, display, revers, order, print,...	1997-03-11	432112newsML.xml	G15	8
6	432113	OFFICIAL JOURNAL CONTENTS - OJ C 76 OF MARCH 1...	[note, content, display, revers, order, print,...	1997-03-11	432113newsML.xml	G15	7

In [101]:
```
# defined function to extract features from individual clusters
def featureslabelsExtraction(dataframe):
    textList = []
    new_df = dataframe

    new_df = new_df.drop(['itemid', 'headline', 'date','XMLfilename'], axis=1)
    for text in new_df['text']:
        text = ' '.join(text)
        textList.append(text)

    #if-idf Vectorizer
    from sklearn.feature_extraction.text import TfidfVectorizer
    vec = TfidfVectorizer(min_df = 0.01,
    max_df = 0.8,
    stop_words = 'english')

    #feature data
    feature_data=vec.fit_transform(textList)

    #encoding labels
    labelencoder = preprocessing.LabelEncoder()
    #label data
    labels = labelencoder.fit_transform(dataframe['bip'])

    return feature_data,labels
```

We have used train test split method because the amount of time required for training will be lower than Cross Validation. Also, this is helpful as we have computational constraints and the dataset is large and using cross validation for every cluster would require high computational power. We have used random state parameter so that data will be choosen randomly so that model won't just learn certain cases.

In [102]:
```
#splitting train set and test set
def trainModel(feature_data, labels):
    train_features, test_features, train_labels, test_labels = train_test_split(feature_data, labels , test_size

    return train_features, test_features, train_labels, test_labels
```

In [103]:
```
#Function that takes features and labels and return a trained classifier
def getClassifier(model,train_features,train_labels):
    model = model.fit(train_features,train_labels)
    return model
```

We have used confusion matrix and accuracy score to evaluate quality of my classifier as confusion matrix gives complete picture of how the classifer is performing. It gives deep insight not only into errors but also the type of errors that is being made by the classifier. Using confusion matrix we have evaluate accuracy score, sensitivity and specificity of the model to find how accurate the model is, how sensitive is the classifier in detecting positive instances and how specific is the model in predicitng positive instances.

In [104]:
```
# Function to evaluate classifier to see how it performed on the data
def evaluateClassifier(model,test_features,test_labels):

    y_pred= model.predict(test_features)
    accuracy=accuracy_score(test_labels, y_pred)
    confusion=confusion_matrix(test_labels, y_pred) #finding confusion matrix


    TP = confusion[1, 1]
    TN = confusion[0, 0]
    FP = confusion[0, 1]
    FN = confusion[1, 0]

    sensitivity = TP / float(FN + TP)
    specificity = TN / (TN + FP)

    return accuracy,sensitivity,specificity
```

We have used MultiLayer Perceptron (Neural Network) classifier to classify data as have the ability to learn and model non-linear and complex relationships, which is really important as we had non-linear relationship with the text data and the labels. Given a set of features a target, it can learn a non-linear function approximator for either classification or regression. Also, it performs well for any given any amount of data [6]. Also, it would help to capture important patterns and distill non important feature to predict better. Moreover, we wanted to see a fair comparison before and after extracting features using autoencoder and a deep learning model. Thus using neural network before would give a good idea on how autoencoder performs to extract features and improving accuracy as we have to use a deep learning model after extracting features from autoencoder.

In [105]:
```
# Function to train using MLP classifier and find out how it performed
def neuralNetwork(feature_data,labels):
    train_features, test_features, train_labels, test_labels = trainModel(feature_data, labels)
    clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(15,15), random_state=1,max_iter=100)
    trained_model = getClassifier(clf,train_features,train_labels)

    model_accuracy_score,model_sensitivity,model_specificity=evaluateClassifier(trained_model,test_features,test
    print('\033[1m'+"Accuracy"+'\033[0m',model_accuracy_score)
    print('\033[1m'+"Sensitivity"+'\033[0m',model_sensitivity)
    print('\033[1m'+"Specificity"+'\033[0m',model_specificity)
    print()
    print()
    print()
```

In [106]:
```
#Function to extract fetaures and run a classifer on it
def genrateCLassifier(dataframe):
    feature_data,labels = featureslabelsExtraction(dataframe)
    neuralNetwork(feature_data,labels)
```

We have made this program dynamic which extracts features and uses MLP classifier on each individual cluster and prints each individual cluster's model accuracy, sensitivity and specificity. As it can be seen below for each cluster

number of documents per cluster is printed and MPL classfier is used on it to find out its aacuracy. As we have selected 10 clusters there are 10 outputs which shows each individual cluster's model performance.

In [113]:
```
cluster_number=[0,1,2,3,4,5,6,7,8,9]
cluster_accuracy = []
for y in cluster_number:
    df =cluster_df_list[y]
    range =(df.shape[0])
    print ('\033[1m'+'Cluster number -  '+'\033[0m' + str(y) +'\033[1m'+ '   number of documents per cluster - '
    genrateCLassifier(df)
    cluster_accuracy.append((acc))
```

Cluster number - 0   number of documents per cluster - 2647

D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: RuntimeWarning: invalid value encountered in true_div
ide

D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: RuntimeWarning: invalid value encountered in longlong
_scalars
  from ipykernel import kernelapp as app

Accuracy 0.850453172205438
Sensitivity nan
Specificity nan

Cluster number - 1   number of documents per cluster - 7180
Accuracy 0.5983286908077995
Sensitivity 1.0
Specificity 0.0

Cluster number - 2   number of documents per cluster - 4434

D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: RuntimeWarning: invalid value encountered in true_div
ide

Accuracy 0.7754733994589721
Sensitivity nan
Specificity 1.0

Cluster number - 3 **number of documents per cluster -** 2857

D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: RuntimeWarning: invalid value encountered in longlong_scalars
  from ipykernel import kernelapp as app

Accuracy 0.9818181818181818
Sensitivity 0.9985754985754985
Specificity nan

Cluster number - 4 **number of documents per cluster -** 3249
Accuracy 0.5682656826568265
Sensitivity 0.3333333333333333
Specificity 1.0

Cluster number - 5 **number of documents per cluster -** 2746

D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: RuntimeWarning: invalid value encountered in true_divide

Accuracy 0.8879184861717613
Sensitivity nan
Specificity 0.0

Cluster number - 6 **number of documents per cluster -** 1262
Accuracy 0.680379746835443
Sensitivity 1.0
Specificity 0.5

Cluster number - 7 **number of documents per cluster -** 12159
Accuracy 0.5634868421052631
Sensitivity 1.0
Specificity 0.0

Cluster number - 8 **number of documents per cluster -** 6118

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: RuntimeWarning: invalid value encountered in true_div
ide

D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: RuntimeWarning: invalid value encountered in longlong
_scalars
  from ipykernel import kernelapp as app
```

Accuracy 0.665359477124183
Sensitivity nan
Specificity nan

Cluster number - 9 **number of documents per cluster -** 3030
Accuracy 0.9366754617414248
Sensitivity nan
Specificity nan

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: RuntimeWarning: invalid value encountered in true_div
ide

D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: RuntimeWarning: invalid value encountered in longlong
_scalars
  from ipykernel import kernelapp as app
```

As it can be seen from the above result that the classifier didn't had good results as there were many fetaures as it was a text data and the model generated became very complex which lead to bad performance of the model. Some of the classifier seems to underfit and some of them seems to overfit.

In [99]:
```
# function to measure quality of cluster
def clusterQuality(X,labels, kmlabels):
    print("Davies Bouldin Score: %0.3f" %davies_bouldin_score(X, kmlabels))
    print("Silhouette Coefficient: %0.3f" %silhouette_score(X, kmlabels))
```

We have used Davies Bouldin Score and Silhouette Coefficient to measure the quality of cluster as one will tell how good the cluster are formed and other will tell well assigned the points are in the cluster

1. Davies Bouldin Score : We have used this metric as it captures the intuition that clusters that are (1) well-spaced from each other and (2) themselves very dense are likely a ‘good’ clustering. Thus, clusters which are farther apart and less dispersed will result in a better score. The minimum score is zero, with lower values indicating better clustering. As it can be seen from the result of the score which is 0.236 which is closer to 0 that the clusters formed are well spaced from each other and less dispersed. Thus, it can be said that different documents are nicely clustered and less dispersed [7].

2. Silhouette Coefficient : We have used this metric as tells us how well-assigned each individual point. The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. As it can be observed from our score it is close to 1 which indicates the samples are well assigned to individual clusters and are not falling between two clusters [8].

In [100]:
```
clusterlabels = model.labels_
clusterQuality(X_train_tfidf,doc_df['bip'],clusterlabels)
```

```
Davies Bouldin Score: 0.236
Silhouette Coefficient: 0.6882782239819362
```

As it can be seen the classifier have not performed well on the perviously

extracted features from the data using TF-IDF vectorizer. So we decided to implement a better way to extract features from the data. We have used research paper number 3 from the given papers which explain the use of autoencoder to extract features from multidimensional raw data [9]. Autoencoder helps to address the problem of normal machine learning models which do not perform well with high dimensional data like text data. Moreover, normal machine learning model may be difficult to scale and prone to information loss, affecting the effectiveness and maintainability. As we had text data which has large set of multidimensional features so we have used this method to solve the problem of normal classifiers. Autoencoders helps to reduce the dimensionaity of the features and as we have huge text data it would be useful to extract only features which are valaubale for the classification process. Autoencoder encodes the features and converts them to numeric values by using a parametrized function.

In [115]:
```
# function to use autoencoder to extract features
def auto_encoder(feature_data, labels, column_value):
    train_features, test_features, train_labels, test_labels = trainModel(feature_data, labels)
    encoding_text = int(column_value/2)
    # this is our input placeholder
    input_text = Input(shape=(column_value,))
    # "encoded" is the encoded representation of the input
    encoded = Dense(encoding_text, activation='relu')(input_text)
    # "decoded" is the lossy reconstruction of the input
    decoded = Dense(column_value, activation='sigmoid')(encoded)
    # this model maps an input to its reconstruction
    autoencoder = Model(input_text, decoded)
    encoder = Model(input_text, encoded)
    # create a placeholder for an encoded (32-dimensional) input
    encoded_input = Input(shape=(encoding_text,))
    # retrieve the last layer of the autoencoder model
    decoder_layer = autoencoder.layers[-1]
    # create the decoder model
    decoder = Model(encoded_input, decoder_layer(encoded_input))

    autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

    autoencoder.fit(train_features, train_features,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(test_features, test_features))

    return encoder,decoder
```

We have used a Deep Learning model called Sequential Model as they give the best result with unstructured data and as we had text data we wanted to see good it really is. It also does not requires data to be labeled and reduces unncessary cost. Also it reduces the need for feature engineering, one of the most time-consuming parts of machine learning practice. Thus we wanted to see how good it performs with the autoencoder and we expected to see good result by combining both these techniques.

In [120]:
```
# Function to use sequantial deep learning model to classify the documents
def Sequential_model(encoded_imgs, labels, column_value):
    train_features, test_features, train_labels, test_labels = trainModel(encoded_imgs, labels )
    train_labels = utils.to_categorical(train_labels, 103)
    test_labels = utils.to_categorical(test_labels, 103)
    model_mc = Sequential()
    #add model layers
    i_shape = int(column_value/2)
    model_mc.add(Dense(256, activation='relu', input_shape=(i_shape,)))
    model_mc.add(Dense(226, activation='relu'))
    model_mc.add(Dense(256, activation='relu'))
    model_mc.add(Dense(103))

    #compile model using mse as a measure of model performance
    model_mc.compile(optimizer='adam', loss='mean_squared_error',metrics=['accuracy'])
    early_stopping_monitor = EarlyStopping(patience=3)
    #train model
    history = model_mc.fit(train_features, train_labels, validation_split=0.2, epochs=30, callbacks=[early_stopp
    score=model_mc.evaluate(test_features,test_labels)
    print(score)
    print()
    print()
    print()
    print(history.history.keys())
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    print()
    print()
    print()
    print()
```

In [121]:
```
# Function to train deep learning model and evaluate its performance
def generateDeepNeuralNetworkModel(df):
    #cluster_df =cluster_df_list[]
    cluster_feature,cluster_labels = featureslabelsExtraction(df)
    column = cluster_feature.shape[1]
    encoder,decoder = auto_encoder(cluster_feature,cluster_labels,column)
    encoded_imgs = encoder.predict(cluster_feature)
    Sequential_model(encoded_imgs, cluster_labels,column)
```

In [122]:
```
cluster_number=[0,1,2,3,4,5,6,7,8,9]
new_cluster_accuracy = []
```

In [124]:
```
for x in cluster_number:
    cluster_df = cluster_df_list[x]
    generateDeepNeuralNetworkModel(cluster_df)
    new_cluster_accuracy.append(score[1])
```

```
Train on 1985 samples, validate on 662 samples
Epoch 1/50
1985/1985 [==============================] - 1s 349us/step - loss: 0.6762 - val_loss: 0.6343
Epoch 2/50
1985/1985 [==============================] - 1s 317us/step - loss: 0.5617 - val_loss: 0.4319
Epoch 3/50
1985/1985 [==============================] - 1s 326us/step - loss: 0.3192 - val_loss: 0.1783
Epoch 4/50
1985/1985 [==============================] - 1s 313us/step - loss: 0.1200 - val_loss: 0.0642
Epoch 5/50
1985/1985 [==============================] - 1s 303us/step - loss: 0.0513 - val_loss: 0.0395
Epoch 6/50
1985/1985 [==============================] - 1s 303us/step - loss: 0.0377 - val_loss: 0.0353
Epoch 7/50
1985/1985 [==============================] - 1s 311us/step - loss: 0.0353 - val_loss: 0.0346
Epoch 8/50
1985/1985 [==============================] - 1s 333us/step - loss: 0.0346 - val_loss: 0.0341
Epoch 9/50
1985/1985 [==============================] - 1s 327us/step - loss: 0.0341 - val_loss: 0.0337
Epoch 10/50
1985/1985 [==============================] - 1s 317us/step - loss: 0.0337 - val_loss: 0.0333
Epoch 11/50
1985/1985 [==============================] - 1s 327us/step - loss: 0.0333 - val_loss: 0.0329
Epoch 12/50
1985/1985 [==============================] - 1s 307us/step - loss: 0.0329 - val_loss: 0.0327
Epoch 13/50
1985/1985 [==============================] - 1s 312us/step - loss: 0.0326 - val_loss: 0.0324
Epoch 14/50
1985/1985 [==============================] - 1s 492us/step - loss: 0.0324 - val_loss: 0.0322
Epoch 15/50
1985/1985 [==============================] - 1s 543us/step - loss: 0.0322 - val_loss: 0.0321
Epoch 16/50
1985/1985 [==============================] - 1s 404us/step - loss: 0.0320 - val_loss: 0.0319
Epoch 17/50
1985/1985 [==============================] - 1s 331us/step - loss: 0.0318 - val_loss: 0.0318
Epoch 18/50
```

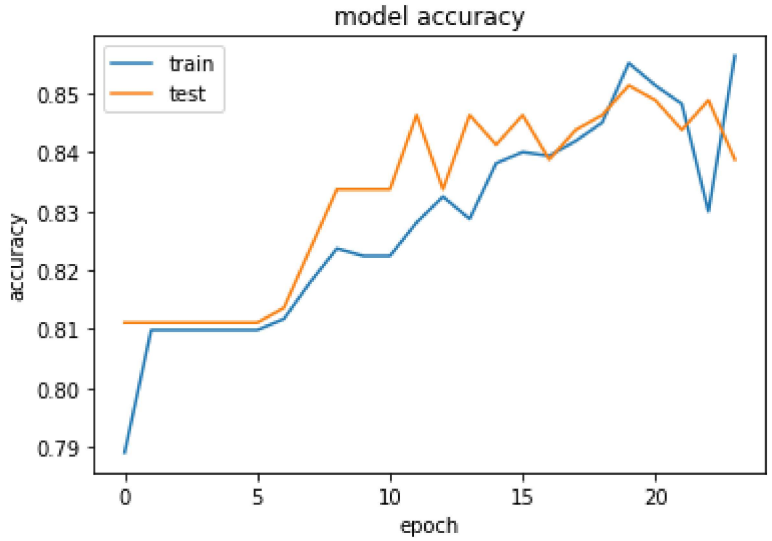
```
1985/1985 [==============================] - 1s 302us/step - loss: 0.0316 - val_loss: 0.0317
Epoch 19/50
1985/1985 [==============================] - 1s 299us/step - loss: 0.0315 - val_loss: 0.0315
Epoch 20/50
1985/1985 [==============================] - 1s 306us/step - loss: 0.0313 - val_loss: 0.0314
Epoch 21/50
1985/1985 [==============================] - 1s 302us/step - loss: 0.0312 - val_loss: 0.0313
Epoch 22/50
1985/1985 [==============================] - 1s 342us/step - loss: 0.0310 - val_loss: 0.0312
Epoch 23/50
1985/1985 [==============================] - 1s 307us/step - loss: 0.0309 - val_loss: 0.0311
Epoch 24/50
1985/1985 [==============================] - 1s 332us/step - loss: 0.0308 - val_loss: 0.0310
Epoch 25/50
1985/1985 [==============================] - 1s 313us/step - loss: 0.0307 - val_loss: 0.0309
Epoch 26/50
1985/1985 [==============================] - 1s 299us/step - loss: 0.0305 - val_loss: 0.0308
Epoch 27/50
1985/1985 [==============================] - 1s 299us/step - loss: 0.0304 - val_loss: 0.0307
Epoch 28/50
1985/1985 [==============================] - 1s 306us/step - loss: 0.0303 - val_loss: 0.0306
Epoch 29/50
1985/1985 [==============================] - 1s 306us/step - loss: 0.0302 - val_loss: 0.0305
Epoch 30/50
1985/1985 [==============================] - 1s 354us/step - loss: 0.0301 - val_loss: 0.0304
Epoch 31/50
1985/1985 [==============================] - 1s 334us/step - loss: 0.0300 - val_loss: 0.0303
Epoch 32/50
1985/1985 [==============================] - 1s 352us/step - loss: 0.0299 - val_loss: 0.0302
Epoch 33/50
1985/1985 [==============================] - 1s 403us/step - loss: 0.0298 - val_loss: 0.0302
Epoch 34/50
1985/1985 [==============================] - 1s 466us/step - loss: 0.0297 - val_loss: 0.0301
Epoch 35/50
1985/1985 [==============================] - 1s 436us/step - loss: 0.0296 - val_loss: 0.0300
Epoch 36/50
1985/1985 [==============================] - 1s 430us/step - loss: 0.0295 - val_loss: 0.0299
Epoch 37/50
1985/1985 [==============================] - 1s 353us/step - loss: 0.0294 - val_loss: 0.0298
Epoch 38/50
1985/1985 [==============================] - 1s 367us/step - loss: 0.0293 - val_loss: 0.0298
Epoch 39/50
1985/1985 [==============================] - 1s 457us/step - loss: 0.0292 - val_loss: 0.0297
```

```
Epoch 40/50
1985/1985 [==============================] - 1s 390us/step - loss: 0.0292 - val_loss: 0.0296
Epoch 41/50
1985/1985 [==============================] - 1s 425us/step - loss: 0.0291 - val_loss: 0.0296
Epoch 42/50
1985/1985 [==============================] - 1s 495us/step - loss: 0.0290 - val_loss: 0.0295
Epoch 43/50
1985/1985 [==============================] - 1s 445us/step - loss: 0.0289 - val_loss: 0.0294
Epoch 44/50
1985/1985 [==============================] - 1s 431us/step - loss: 0.0288 - val_loss: 0.0293
Epoch 45/50
1985/1985 [==============================] - 1s 451us/step - loss: 0.0288 - val_loss: 0.0293
Epoch 46/50
1985/1985 [==============================] - 1s 365us/step - loss: 0.0287 - val_loss: 0.0292
Epoch 47/50
1985/1985 [==============================] - 1s 360us/step - loss: 0.0286 - val_loss: 0.0292
Epoch 48/50
1985/1985 [==============================] - 1s 408us/step - loss: 0.0286 - val_loss: 0.0291
Epoch 49/50
1985/1985 [==============================] - 1s 478us/step - loss: 0.0285 - val_loss: 0.0290
Epoch 50/50
1985/1985 [==============================] - 1s 511us/step - loss: 0.0284 - val_loss: 0.0290
Train on 1588 samples, validate on 397 samples
Epoch 1/30
1588/1588 [==============================] - 1s 333us/step - loss: 0.0043 - accuracy: 0.7890 - val_loss: 0.003
3 - val_accuracy: 0.8111
Epoch 2/30
1588/1588 [==============================] - 0s 300us/step - loss: 0.0032 - accuracy: 0.8098 - val_loss: 0.003
3 - val_accuracy: 0.8111
Epoch 3/30
1588/1588 [==============================] - 0s 278us/step - loss: 0.0032 - accuracy: 0.8098 - val_loss: 0.003
0 - val_accuracy: 0.8111
Epoch 4/30
1588/1588 [==============================] - 1s 327us/step - loss: 0.0030 - accuracy: 0.8098 - val_loss: 0.003
1 - val_accuracy: 0.8111
Epoch 5/30
1588/1588 [==============================] - 0s 237us/step - loss: 0.0029 - accuracy: 0.8098 - val_loss: 0.003
0 - val_accuracy: 0.8111
Epoch 6/30
1588/1588 [==============================] - 0s 259us/step - loss: 0.0028 - accuracy: 0.8098 - val_loss: 0.002
8 - val_accuracy: 0.8111
Epoch 7/30
1588/1588 [==============================] - 0s 268us/step - loss: 0.0027 - accuracy: 0.8117 - val_loss: 0.002
```

```
6 - val_accuracy: 0.8136
Epoch 8/30
1588/1588 [==============================] - 0s 251us/step - loss: 0.0026 - accuracy: 0.8180 - val_loss: 0.002
5 - val_accuracy: 0.8237
Epoch 9/30
1588/1588 [==============================] - 0s 243us/step - loss: 0.0025 - accuracy: 0.8237 - val_loss: 0.002
8 - val_accuracy: 0.8338
Epoch 10/30
1588/1588 [==============================] - 0s 311us/step - loss: 0.0025 - accuracy: 0.8224 - val_loss: 0.002
5 - val_accuracy: 0.8338
Epoch 11/30
1588/1588 [==============================] - 0s 261us/step - loss: 0.0026 - accuracy: 0.8224 - val_loss: 0.002
4 - val_accuracy: 0.8338
Epoch 12/30
1588/1588 [==============================] - 0s 274us/step - loss: 0.0024 - accuracy: 0.8281 - val_loss: 0.002
4 - val_accuracy: 0.8463
Epoch 13/30
1588/1588 [==============================] - 0s 257us/step - loss: 0.0024 - accuracy: 0.8325 - val_loss: 0.002
4 - val_accuracy: 0.8338
Epoch 14/30
1588/1588 [==============================] - 0s 254us/step - loss: 0.0023 - accuracy: 0.8287 - val_loss: 0.002
3 - val_accuracy: 0.8463
Epoch 15/30
1588/1588 [==============================] - 0s 246us/step - loss: 0.0023 - accuracy: 0.8382 - val_loss: 0.002
4 - val_accuracy: 0.8413
Epoch 16/30
1588/1588 [==============================] - 0s 249us/step - loss: 0.0022 - accuracy: 0.8401 - val_loss: 0.002
2 - val_accuracy: 0.8463
Epoch 17/30
1588/1588 [==============================] - 0s 284us/step - loss: 0.0023 - accuracy: 0.8394 - val_loss: 0.002
4 - val_accuracy: 0.8388
Epoch 18/30
1588/1588 [==============================] - 0s 241us/step - loss: 0.0021 - accuracy: 0.8419 - val_loss: 0.002
2 - val_accuracy: 0.8438
Epoch 19/30
1588/1588 [==============================] - 0s 253us/step - loss: 0.0021 - accuracy: 0.8451 - val_loss: 0.002
2 - val_accuracy: 0.8463
Epoch 20/30

1588/1588 [==============================] - 0s 246us/step - loss: 0.0020 - accuracy: 0.8552 - val_loss: 0.0
022 - val_accuracy: 0.8514
Epoch 21/30
1588/1588 [==============================] - 0s 242us/step - loss: 0.0021 - accuracy: 0.8514 - val_loss: 0.0
```

```
022 - val_accuracy: 0.8489
Epoch 22/30
1588/1588 [==============================] - 0s 242us/step - loss: 0.0021 - accuracy: 0.8482 - val_loss: 0.0
025 - val_accuracy: 0.8438
Epoch 23/30
1588/1588 [==============================] - 0s 239us/step - loss: 0.0024 - accuracy: 0.8300 - val_loss: 0.0
023 - val_accuracy: 0.8489
Epoch 24/30
1588/1588 [==============================] - 0s 242us/step - loss: 0.0020 - accuracy: 0.8564 - val_loss: 0.0
022 - val_accuracy: 0.8388
662/662 [==============================] - 0s 78us/step
[0.002080407102657004, 0.8504531979560852]
```

```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

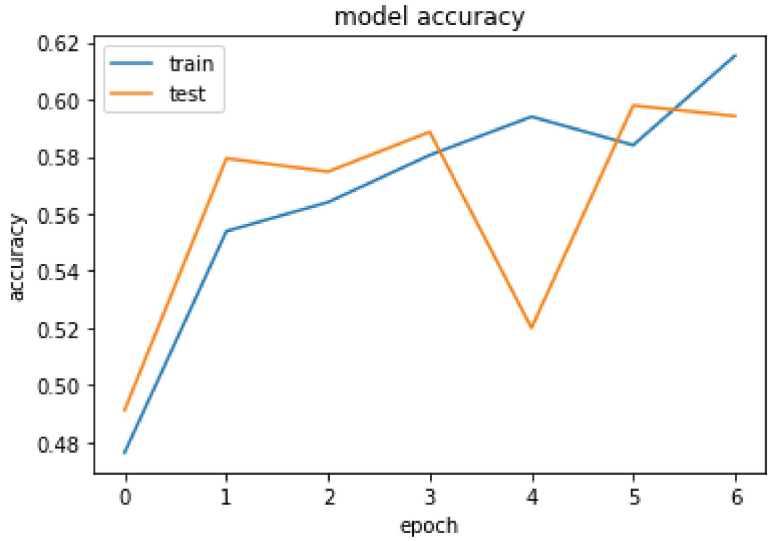



```
Train on 5385 samples, validate on 1795 samples
Epoch 1/50
5385/5385 [==============================] - 2s 345us/step - loss: 0.5566 - val_loss: 0.2330
Epoch 2/50
5385/5385 [==============================] - 2s 319us/step - loss: 0.0910 - val_loss: 0.0367
```

```
Epoch 3/50
5385/5385 [==============================] - 2s 323us/step - loss: 0.0352 - val_loss: 0.0347
Epoch 4/50
5385/5385 [==============================] - 2s 391us/step - loss: 0.0342 - val_loss: 0.0339
Epoch 5/50
5385/5385 [==============================] - 2s 415us/step - loss: 0.0335 - val_loss: 0.0334
Epoch 6/50
5385/5385 [==============================] - 2s 371us/step - loss: 0.0332 - val_loss: 0.0332
Epoch 7/50
5385/5385 [==============================] - 2s 420us/step - loss: 0.0329 - val_loss: 0.0329
Epoch 8/50
5385/5385 [==============================] - 2s 388us/step - loss: 0.0327 - val_loss: 0.0327
Epoch 9/50
5385/5385 [==============================] - 2s 337us/step - loss: 0.0325 - val_loss: 0.0325
Epoch 10/50
5385/5385 [==============================] - 2s 364us/step - loss: 0.0323 - val_loss: 0.0323
Epoch 11/50
5385/5385 [==============================] - 2s 353us/step - loss: 0.0321 - val_loss: 0.0322
Epoch 12/50
5385/5385 [==============================] - 2s 369us/step - loss: 0.0319 - val_loss: 0.0320
Epoch 13/50
5385/5385 [==============================] - 2s 397us/step - loss: 0.0317 - val_loss: 0.0318
Epoch 14/50
5385/5385 [==============================] - 2s 345us/step - loss: 0.0315 - val_loss: 0.0317
Epoch 15/50
5385/5385 [==============================] - 2s 356us/step - loss: 0.0314 - val_loss: 0.0316
Epoch 16/50
5385/5385 [==============================] - 3s 477us/step - loss: 0.0312 - val_loss: 0.0314
Epoch 17/50
5385/5385 [==============================] - 2s 364us/step - loss: 0.0311 - val_loss: 0.0313
Epoch 18/50
5385/5385 [==============================] - 2s 335us/step - loss: 0.0310 - val_loss: 0.0312
Epoch 19/50
5385/5385 [==============================] - 2s 340us/step - loss: 0.0308 - val_loss: 0.0311
Epoch 20/50
5385/5385 [==============================] - 2s 360us/step - loss: 0.0307 - val_loss: 0.0310
Epoch 21/50
5385/5385 [==============================] - 2s 387us/step - loss: 0.0306 - val_loss: 0.0309
Epoch 22/50
5385/5385 [==============================] - 2s 335us/step - loss: 0.0305 - val_loss: 0.0309
Epoch 23/50
5385/5385 [==============================] - 2s 356us/step - loss: 0.0304 - val_loss: 0.0308
Epoch 24/50
```

```
5385/5385 [==============================] - 2s 337us/step - loss: 0.0303 - val_loss: 0.0307
Epoch 25/50
5385/5385 [==============================] - 2s 372us/step - loss: 0.0302 - val_loss: 0.0306
Epoch 26/50
5385/5385 [==============================] - 2s 416us/step - loss: 0.0302 - val_loss: 0.0306
Epoch 27/50
5385/5385 [==============================] - 2s 429us/step - loss: 0.0301 - val_loss: 0.0305
Epoch 28/50
5385/5385 [==============================] - 2s 456us/step - loss: 0.0300 - val_loss: 0.0304
Epoch 29/50
5385/5385 [==============================] - 3s 505us/step - loss: 0.0299 - val_loss: 0.0304
Epoch 30/50
5385/5385 [==============================] - 2s 384us/step - loss: 0.0299 - val_loss: 0.0303
Epoch 31/50
5385/5385 [==============================] - 2s 335us/step - loss: 0.0298 - val_loss: 0.0302
Epoch 32/50
5385/5385 [==============================] - 2s 324us/step - loss: 0.0297 - val_loss: 0.0302
Epoch 33/50
5385/5385 [==============================] - 2s 312us/step - loss: 0.0296 - val_loss: 0.0301
Epoch 34/50
5385/5385 [==============================] - 2s 312us/step - loss: 0.0295 - val_loss: 0.0300
Epoch 35/50
5385/5385 [==============================] - 2s 332us/step - loss: 0.0295 - val_loss: 0.0300
Epoch 36/50
5385/5385 [==============================] - 2s 373us/step - loss: 0.0294 - val_loss: 0.0299
Epoch 37/50
5385/5385 [==============================] - 2s 345us/step - loss: 0.0293 - val_loss: 0.0298
Epoch 38/50
5385/5385 [==============================] - 2s 401us/step - loss: 0.0292 - val_loss: 0.0297
Epoch 39/50
5385/5385 [==============================] - 2s 439us/step - loss: 0.0291 - val_loss: 0.0296
Epoch 40/50
5385/5385 [==============================] - 2s 420us/step - loss: 0.0290 - val_loss: 0.0296
Epoch 41/50
5385/5385 [==============================] - 2s 398us/step - loss: 0.0289 - val_loss: 0.0295
Epoch 42/50
5385/5385 [==============================] - 2s 378us/step - loss: 0.0288 - val_loss: 0.0294
Epoch 43/50
5385/5385 [==============================] - 2s 375us/step - loss: 0.0287 - val_loss: 0.0293
Epoch 44/50
5385/5385 [==============================] - 2s 371us/step - loss: 0.0286 - val_loss: 0.0292
Epoch 45/50
5385/5385 [==============================] - 2s 376us/step - loss: 0.0285 - val_loss: 0.0291
```

```
Epoch 46/50
5385/5385 [==============================] - 2s 382us/step - loss: 0.0284 - val_loss: 0.0290
Epoch 47/50
5385/5385 [==============================] - 2s 349us/step - loss: 0.0283 - val_loss: 0.0289
Epoch 48/50
5385/5385 [==============================] - 2s 372us/step - loss: 0.0282 - val_loss: 0.02880s - l
Epoch 49/50
5385/5385 [==============================] - 2s 384us/step - loss: 0.0281 - val_loss: 0.0287
Epoch 50/50
5385/5385 [==============================] - 2s 392us/step - loss: 0.0280 - val_loss: 0.0286
Train on 4308 samples, validate on 1077 samples
Epoch 1/30
4308/4308 [==============================] - 1s 272us/step - loss: 0.0069 - accuracy: 0.4763 - val_loss: 0.0
065 - val_accuracy: 0.4912
Epoch 2/30
4308/4308 [==============================] - 1s 306us/step - loss: 0.0059 - accuracy: 0.5539 - val_loss: 0.0
056 - val_accuracy: 0.5794
Epoch 3/30
4308/4308 [==============================] - 1s 275us/step - loss: 0.0057 - accuracy: 0.5641 - val_loss: 0.0
055 - val_accuracy: 0.5747
Epoch 4/30
4308/4308 [==============================] - 1s 278us/step - loss: 0.0055 - accuracy: 0.5805 - val_loss: 0.0
054 - val_accuracy: 0.5887
Epoch 5/30
4308/4308 [==============================] - 1s 257us/step - loss: 0.0053 - accuracy: 0.5940 - val_loss: 0.0
060 - val_accuracy: 0.52000s - loss: 0.0
Epoch 6/30
4308/4308 [==============================] - 1s 214us/step - loss: 0.0054 - accuracy: 0.5840 - val_loss: 0.0
056 - val_accuracy: 0.5980
Epoch 7/30
4308/4308 [==============================] - 1s 192us/step - loss: 0.0051 - accuracy: 0.6154 - val_loss: 0.0
055 - val_accuracy: 0.5942
1795/1795 [==============================] - 0s 55us/step
[0.00532433472857175, 0.6044568419456482]



dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```



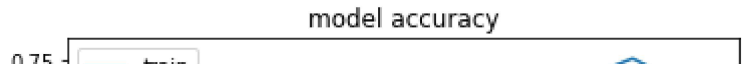

```
Train on 3325 samples, validate on 1109 samples
Epoch 1/50
3325/3325 [==============================] - 1s 375us/step - loss: 0.6445 - val_loss: 0.5154
Epoch 2/50
3325/3325 [==============================] - 1s 417us/step - loss: 0.3075 - val_loss: 0.1111
Epoch 3/50
3325/3325 [==============================] - 1s 423us/step - loss: 0.0621 - val_loss: 0.0391
Epoch 4/50
3325/3325 [==============================] - 1s 431us/step - loss: 0.0375 - val_loss: 0.0363
Epoch 5/50
3325/3325 [==============================] - 1s 384us/step - loss: 0.0365 - val_loss: 0.0356
Epoch 6/50
3325/3325 [==============================] - 1s 372us/step - loss: 0.0357 - val_loss: 0.0348
Epoch 7/50
3325/3325 [==============================] - 1s 364us/step - loss: 0.0350 - val_loss: 0.0343
Epoch 8/50
3325/3325 [==============================] - 1s 371us/step - loss: 0.0345 - val_loss: 0.0339
Epoch 9/50
3325/3325 [==============================] - 1s 402us/step - loss: 0.0342 - val_loss: 0.0336
Epoch 10/50
3325/3325 [==============================] - 1s 384us/step - loss: 0.0339 - val_loss: 0.0334
```

```
Epoch 11/50
3325/3325 [==============================] - 1s 393us/step - loss: 0.0336 - val_loss: 0.0331
Epoch 12/50
3325/3325 [==============================] - 1s 421us/step - loss: 0.0334 - val_loss: 0.0329
Epoch 13/50
3325/3325 [==============================] - 1s 381us/step - loss: 0.0332 - val_loss: 0.0327
Epoch 14/50
3325/3325 [==============================] - 1s 394us/step - loss: 0.0330 - val_loss: 0.0325
Epoch 15/50
3325/3325 [==============================] - 1s 364us/step - loss: 0.0328 - val_loss: 0.0324
Epoch 16/50
3325/3325 [==============================] - 1s 352us/step - loss: 0.0326 - val_loss: 0.0322
Epoch 17/50
3325/3325 [==============================] - 1s 361us/step - loss: 0.0324 - val_loss: 0.0320
Epoch 18/50
3325/3325 [==============================] - 1s 362us/step - loss: 0.0322 - val_loss: 0.0319
Epoch 19/50
3325/3325 [==============================] - 1s 373us/step - loss: 0.0320 - val_loss: 0.0317
Epoch 20/50
3325/3325 [==============================] - 1s 363us/step - loss: 0.0319 - val_loss: 0.0316
Epoch 21/50
3325/3325 [==============================] - 1s 371us/step - loss: 0.0317 - val_loss: 0.0314
Epoch 22/50
3325/3325 [==============================] - 1s 372us/step - loss: 0.0316 - val_loss: 0.0313
Epoch 23/50
3325/3325 [==============================] - 1s 390us/step - loss: 0.0314 - val_loss: 0.0312
Epoch 24/50
3325/3325 [==============================] - 1s 387us/step - loss: 0.0313 - val_loss: 0.0311
Epoch 25/50
3325/3325 [==============================] - 1s 371us/step - loss: 0.0312 - val_loss: 0.0310
Epoch 26/50
3325/3325 [==============================] - 1s 377us/step - loss: 0.0311 - val_loss: 0.0309
Epoch 27/50
3325/3325 [==============================] - 1s 390us/step - loss: 0.0310 - val_loss: 0.0308
Epoch 28/50
3325/3325 [==============================] - 1s 362us/step - loss: 0.0308 - val_loss: 0.0307
Epoch 29/50
3325/3325 [==============================] - 1s 358us/step - loss: 0.0307 - val_loss: 0.0306
Epoch 30/50
3325/3325 [==============================] - 1s 359us/step - loss: 0.0306 - val_loss: 0.0305
Epoch 31/50
3325/3325 [==============================] - 1s 404us/step - loss: 0.0305 - val_loss: 0.0305
Epoch 32/50
```

```
3325/3325 [==============================] - 1s 447us/step - loss: 0.0304 - val_loss: 0.0304
Epoch 33/50
3325/3325 [==============================] - 2s 460us/step - loss: 0.0304 - val_loss: 0.0303
Epoch 34/50
3325/3325 [==============================] - 1s 364us/step - loss: 0.0303 - val_loss: 0.0302
Epoch 35/50
3325/3325 [==============================] - 1s 360us/step - loss: 0.0302 - val_loss: 0.0301
Epoch 36/50
3325/3325 [==============================] - 1s 374us/step - loss: 0.0301 - val_loss: 0.0301
Epoch 37/50
3325/3325 [==============================] - 1s 374us/step - loss: 0.0300 - val_loss: 0.0300
Epoch 38/50
3325/3325 [==============================] - 1s 403us/step - loss: 0.0299 - val_loss: 0.0299
Epoch 39/50
3325/3325 [==============================] - 1s 360us/step - loss: 0.0299 - val_loss: 0.0299
Epoch 40/50
3325/3325 [==============================] - 1s 360us/step - loss: 0.0298 - val_loss: 0.0298
Epoch 41/50
3325/3325 [==============================] - 1s 361us/step - loss: 0.0297 - val_loss: 0.0297
Epoch 42/50
3325/3325 [==============================] - 1s 355us/step - loss: 0.0296 - val_loss: 0.0296
Epoch 43/50
3325/3325 [==============================] - 1s 404us/step - loss: 0.0295 - val_loss: 0.0296
Epoch 44/50
3325/3325 [==============================] - 2s 463us/step - loss: 0.0295 - val_loss: 0.0295
Epoch 45/50
3325/3325 [==============================] - 1s 417us/step - loss: 0.0294 - val_loss: 0.0294
Epoch 46/50
3325/3325 [==============================] - 1s 382us/step - loss: 0.0293 - val_loss: 0.0294
Epoch 47/50
3325/3325 [==============================] - 1s 389us/step - loss: 0.0292 - val_loss: 0.0293
Epoch 48/50
3325/3325 [==============================] - 1s 357us/step - loss: 0.0292 - val_loss: 0.0292
Epoch 49/50
3325/3325 [==============================] - 1s 369us/step - loss: 0.0291 - val_loss: 0.0292
Epoch 50/50
3325/3325 [==============================] - 1s 358us/step - loss: 0.0290 - val_loss: 0.0291
Train on 2660 samples, validate on 665 samples
Epoch 1/30
2660/2660 [==============================] - 1s 290us/step - loss: 0.0067 - accuracy: 0.4880 - val_loss: 0.006
0 - val_accuracy: 0.5338
Epoch 2/30
2660/2660 [==============================] - 1s 198us/step - loss: 0.0050 - accuracy: 0.6444 - val_loss: 0.005
```

```
6 - val_accuracy: 0.5654
Epoch 3/30
2660/2660 [==============================] - 1s 201us/step - loss: 0.0044 - accuracy: 0.6880 - val_loss: 0.005
1 - val_accuracy: 0.6256
Epoch 4/30
2660/2660 [==============================] - 1s 199us/step - loss: 0.0043 - accuracy: 0.6981 - val_loss: 0.005
1 - val_accuracy: 0.6195
Epoch 5/30
2660/2660 [==============================] - 1s 199us/step - loss: 0.0040 - accuracy: 0.7150 - val_loss: 0.004
7 - val_accuracy: 0.6526
Epoch 6/30
2660/2660 [==============================] - 1s 198us/step - loss: 0.0038 - accuracy: 0.7301 - val_loss: 0.004
3 - val_accuracy: 0.6977
Epoch 7/30
2660/2660 [==============================] - 1s 196us/step - loss: 0.0037 - accuracy: 0.7320 - val_loss: 0.004
4 - val_accuracy: 0.6767
Epoch 8/30
2660/2660 [==============================] - 1s 210us/step - loss: 0.0035 - accuracy: 0.7515 - val_loss: 0.004
3 - val_accuracy: 0.6902
Epoch 9/30
2660/2660 [==============================] - 1s 200us/step - loss: 0.0039 - accuracy: 0.7267 - val_loss: 0.004
3 - val_accuracy: 0.6992
1109/1109 [==============================] - 0s 49us/step
[0.004006644065103034, 0.7357980012893677]
```

```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```




```
Train on 2142 samples, validate on 715 samples
Epoch 1/50
2142/2142 [==============================] - 0s 87us/step - loss: 0.6808 - val_loss: 0.6588
Epoch 2/50
2142/2142 [==============================] - 0s 45us/step - loss: 0.6334 - val_loss: 0.5863
Epoch 3/50
2142/2142 [==============================] - 0s 46us/step - loss: 0.5390 - val_loss: 0.4600
Epoch 4/50
2142/2142 [==============================] - 0s 45us/step - loss: 0.3969 - val_loss: 0.3047
Epoch 5/50
2142/2142 [==============================] - 0s 46us/step - loss: 0.2485 - val_loss: 0.1779
Epoch 6/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.1454 - val_loss: 0.1092
Epoch 7/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0953 - val_loss: 0.0813
Epoch 8/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0759 - val_loss: 0.0709
Epoch 9/50
2142/2142 [==============================] - 0s 46us/step - loss: 0.0686 - val_loss: 0.0668
Epoch 10/50
2142/2142 [==============================] - 0s 46us/step - loss: 0.0655 - val_loss: 0.0648
Epoch 11/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0639 - val_loss: 0.0637
Epoch 12/50
2142/2142 [==============================] - 0s 46us/step - loss: 0.0630 - val_loss: 0.0630
Epoch 13/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0624 - val_loss: 0.0625
Epoch 14/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0619 - val_loss: 0.0621
Epoch 15/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0615 - val_loss: 0.0618
Epoch 16/50
2142/2142 [==============================] - 0s 49us/step - loss: 0.0613 - val_loss: 0.0616
Epoch 17/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0610 - val_loss: 0.0614
Epoch 18/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0608 - val_loss: 0.0612
```

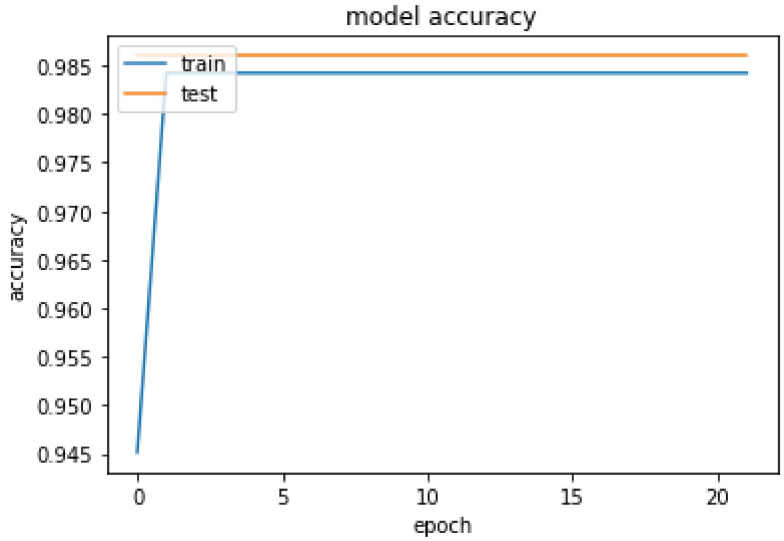
```
Epoch 19/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0606 - val_loss: 0.0610
Epoch 20/50
2142/2142 [==============================] - 0s 46us/step - loss: 0.0605 - val_loss: 0.0609
Epoch 21/50
2142/2142 [==============================] - 0s 50us/step - loss: 0.0603 - val_loss: 0.0607
Epoch 22/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0602 - val_loss: 0.0606
Epoch 23/50
2142/2142 [==============================] - 0s 46us/step - loss: 0.0600 - val_loss: 0.0605
Epoch 24/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0599 - val_loss: 0.0603
Epoch 25/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0598 - val_loss: 0.0602
Epoch 26/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0596 - val_loss: 0.0600
Epoch 27/50
2142/2142 [==============================] - 0s 54us/step - loss: 0.0595 - val_loss: 0.0599
Epoch 28/50
2142/2142 [==============================] - 0s 55us/step - loss: 0.0593 - val_loss: 0.0597
Epoch 29/50
2142/2142 [==============================] - 0s 50us/step - loss: 0.0592 - val_loss: 0.0596
Epoch 30/50
2142/2142 [==============================] - 0s 51us/step - loss: 0.0590 - val_loss: 0.0594
Epoch 31/50
2142/2142 [==============================] - 0s 46us/step - loss: 0.0588 - val_loss: 0.0592
Epoch 32/50
2142/2142 [==============================] - 0s 50us/step - loss: 0.0586 - val_loss: 0.0590
Epoch 33/50
2142/2142 [==============================] - 0s 54us/step - loss: 0.0585 - val_loss: 0.0588
Epoch 34/50
2142/2142 [==============================] - 0s 52us/step - loss: 0.0582 - val_loss: 0.0586
Epoch 35/50
2142/2142 [==============================] - 0s 51us/step - loss: 0.0580 - val_loss: 0.0584
Epoch 36/50
2142/2142 [==============================] - 0s 51us/step - loss: 0.0578 - val_loss: 0.0582
Epoch 37/50
2142/2142 [==============================] - 0s 52us/step - loss: 0.0576 - val_loss: 0.0579
Epoch 38/50
2142/2142 [==============================] - 0s 50us/step - loss: 0.0573 - val_loss: 0.0577
Epoch 39/50
2142/2142 [==============================] - 0s 44us/step - loss: 0.0571 - val_loss: 0.0574
Epoch 40/50
```

```
2142/2142 [==============================] - 0s 47us/step - loss: 0.0568 - val_loss: 0.0572
Epoch 41/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0566 - val_loss: 0.0570
Epoch 42/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0564 - val_loss: 0.0567
Epoch 43/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0561 - val_loss: 0.0565
Epoch 44/50
2142/2142 [==============================] - 0s 50us/step - loss: 0.0559 - val_loss: 0.0563
Epoch 45/50
2142/2142 [==============================] - 0s 50us/step - loss: 0.0557 - val_loss: 0.0560
Epoch 46/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0554 - val_loss: 0.0558
Epoch 47/50
2142/2142 [==============================] - 0s 44us/step - loss: 0.0552 - val_loss: 0.0556
Epoch 48/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0550 - val_loss: 0.0554
Epoch 49/50
2142/2142 [==============================] - 0s 48us/step - loss: 0.0548 - val_loss: 0.0552
Epoch 50/50
2142/2142 [==============================] - 0s 47us/step - loss: 0.0545 - val_loss: 0.0550
Train on 1713 samples, validate on 429 samples
Epoch 1/30
1713/1713 [==============================] - 0s 207us/step - loss: 0.0021 - accuracy: 0.9451 - val_loss: 3.742
8e-04 - val_accuracy: 0.9860
Epoch 2/30
1713/1713 [==============================] - 0s 138us/step - loss: 3.9970e-04 - accuracy: 0.9842 - val_loss:
3.5554e-04 - val_accuracy: 0.9860
Epoch 3/30
1713/1713 [==============================] - 0s 146us/step - loss: 3.7296e-04 - accuracy: 0.9842 - val_loss:
3.2812e-04 - val_accuracy: 0.9860
Epoch 4/30
1713/1713 [==============================] - 0s 140us/step - loss: 3.5747e-04 - accuracy: 0.9842 - val_loss:
3.2258e-04 - val_accuracy: 0.9860
Epoch 5/30
1713/1713 [==============================] - 0s 144us/step - loss: 3.6473e-04 - accuracy: 0.9842 - val_loss:
3.1688e-04 - val_accuracy: 0.9860
Epoch 6/30
1713/1713 [==============================] - 0s 143us/step - loss: 3.4496e-04 - accuracy: 0.9842 - val_loss:
3.0778e-04 - val_accuracy: 0.9860
Epoch 7/30
1713/1713 [==============================] - 0s 145us/step - loss: 3.4368e-04 - accuracy: 0.9842 - val_loss:
2.9969e-04 - val_accuracy: 0.9860
```

```
Epoch 8/30
1713/1713 [==============================] - 0s 151us/step - loss: 3.4469e-04 - accuracy: 0.9842 - val_loss:
2.9465e-04 - val_accuracy: 0.9860
Epoch 9/30
1713/1713 [==============================] - 0s 134us/step - loss: 3.4369e-04 - accuracy: 0.9842 - val_loss:
3.1098e-04 - val_accuracy: 0.9860
Epoch 10/30
1713/1713 [==============================] - 0s 162us/step - loss: 3.2332e-04 - accuracy: 0.9842 - val_loss:
2.8203e-04 - val_accuracy: 0.9860
Epoch 11/30
1713/1713 [==============================] - 0s 185us/step - loss: 3.3769e-04 - accuracy: 0.9842 - val_loss:
3.1165e-04 - val_accuracy: 0.9860
Epoch 12/30
1713/1713 [==============================] - 0s 157us/step - loss: 3.1536e-04 - accuracy: 0.9842 - val_loss:
2.7531e-04 - val_accuracy: 0.9860
Epoch 13/30
1713/1713 [==============================] - 0s 158us/step - loss: 3.1659e-04 - accuracy: 0.9842 - val_loss:
2.7882e-04 - val_accuracy: 0.9860
Epoch 14/30
1713/1713 [==============================] - 0s 147us/step - loss: 3.1610e-04 - accuracy: 0.9842 - val_loss:
2.7394e-04 - val_accuracy: 0.9860
Epoch 15/30
1713/1713 [==============================] - 0s 153us/step - loss: 3.1438e-04 - accuracy: 0.9842 - val_loss:
2.7206e-04 - val_accuracy: 0.9860
Epoch 16/30
1713/1713 [==============================] - 0s 149us/step - loss: 3.0725e-04 - accuracy: 0.9842 - val_loss:
2.7443e-04 - val_accuracy: 0.9860
Epoch 17/30
1713/1713 [==============================] - 0s 152us/step - loss: 3.1238e-04 - accuracy: 0.9842 - val_loss:
2.6631e-04 - val_accuracy: 0.9860
Epoch 18/30
1713/1713 [==============================] - 0s 149us/step - loss: 2.9955e-04 - accuracy: 0.9842 - val_loss:
2.6334e-04 - val_accuracy: 0.9860
Epoch 19/30
1713/1713 [==============================] - 0s 144us/step - loss: 3.0458e-04 - accuracy: 0.9842 - val_loss:
2.6236e-04 - val_accuracy: 0.9860

Epoch 20/30
1713/1713 [==============================] - 0s 189us/step - loss: 2.9634e-04 - accuracy: 0.9842 - val_loss:
2.6656e-04 - val_accuracy: 0.9860
Epoch 21/30
1713/1713 [==============================] - 0s 167us/step - loss: 2.9884e-04 - accuracy: 0.9842 - val_loss:
2.6636e-04 - val_accuracy: 0.9860
```

```
Epoch 22/30
1713/1713 [==============================] - 0s 162us/step - loss: 2.9253e-04 - accuracy: 0.9842 - val_loss:
2.7406e-04 - val_accuracy: 0.9860
715/715 [==============================] - 0s 79us/step
[0.00013087000880004525, 0.9944055676460266]
```

```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```




```
Train on 2436 samples, validate on 813 samples
Epoch 1/50
2436/2436 [==============================] - 1s 371us/step - loss: 0.6663 - val_loss: 0.5966
Epoch 2/50
2436/2436 [==============================] - 1s 334us/step - loss: 0.4797 - val_loss: 0.2973
Epoch 3/50
2436/2436 [==============================] - 1s 336us/step - loss: 0.1850 - val_loss: 0.0818
Epoch 4/50
2436/2436 [==============================] - 1s 329us/step - loss: 0.0567 - val_loss: 0.0386
Epoch 5/50
2436/2436 [==============================] - 1s 337us/step - loss: 0.0357 - val_loss: 0.0335
Epoch 6/50
2436/2436 [==============================] - 1s 340us/step - loss: 0.0332 - val_loss: 0.0328
```

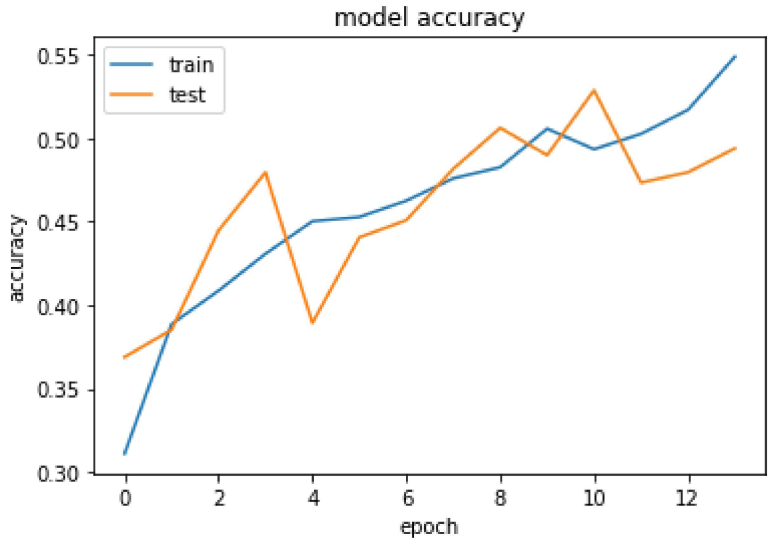
```
Epoch 7/50
2436/2436 [==============================] - 1s 343us/step - loss: 0.0327 - val_loss: 0.0325
Epoch 8/50
2436/2436 [==============================] - 1s 333us/step - loss: 0.0322 - val_loss: 0.0321
Epoch 9/50
2436/2436 [==============================] - 1s 332us/step - loss: 0.0318 - val_loss: 0.0318
Epoch 10/50
2436/2436 [==============================] - 1s 330us/step - loss: 0.0315 - val_loss: 0.0315
Epoch 11/50
2436/2436 [==============================] - 1s 335us/step - loss: 0.0313 - val_loss: 0.0313
Epoch 12/50
2436/2436 [==============================] - 1s 339us/step - loss: 0.0310 - val_loss: 0.0311
Epoch 13/50
2436/2436 [==============================] - 1s 332us/step - loss: 0.0309 - val_loss: 0.0310
Epoch 14/50
2436/2436 [==============================] - 1s 330us/step - loss: 0.0307 - val_loss: 0.0309
Epoch 15/50
2436/2436 [==============================] - 1s 333us/step - loss: 0.0305 - val_loss: 0.0307
Epoch 16/50
2436/2436 [==============================] - 1s 333us/step - loss: 0.0304 - val_loss: 0.0306
Epoch 17/50
2436/2436 [==============================] - 1s 336us/step - loss: 0.0303 - val_loss: 0.0305
Epoch 18/50
2436/2436 [==============================] - 1s 335us/step - loss: 0.0301 - val_loss: 0.0304
Epoch 19/50
2436/2436 [==============================] - 1s 355us/step - loss: 0.0300 - val_loss: 0.0303
Epoch 20/50
2436/2436 [==============================] - 1s 332us/step - loss: 0.0299 - val_loss: 0.0302
Epoch 21/50
2436/2436 [==============================] - 1s 334us/step - loss: 0.0298 - val_loss: 0.0301
Epoch 22/50
2436/2436 [==============================] - 1s 330us/step - loss: 0.0296 - val_loss: 0.0300
Epoch 23/50
2436/2436 [==============================] - 1s 337us/step - loss: 0.0295 - val_loss: 0.0299
Epoch 24/50
2436/2436 [==============================] - 1s 398us/step - loss: 0.0294 - val_loss: 0.0298
Epoch 25/50
2436/2436 [==============================] - 1s 343us/step - loss: 0.0293 - val_loss: 0.0297
Epoch 26/50
2436/2436 [==============================] - 1s 348us/step - loss: 0.0292 - val_loss: 0.0296
Epoch 27/50
2436/2436 [==============================] - 1s 340us/step - loss: 0.0291 - val_loss: 0.0295
Epoch 28/50
```

```
2436/2436 [==============================] - 1s 334us/step - loss: 0.0290 - val_loss: 0.0295
Epoch 29/50
2436/2436 [==============================] - 1s 333us/step - loss: 0.0290 - val_loss: 0.0294
Epoch 30/50
2436/2436 [==============================] - 1s 371us/step - loss: 0.0289 - val_loss: 0.0293
Epoch 31/50
2436/2436 [==============================] - 1s 338us/step - loss: 0.0288 - val_loss: 0.0293
Epoch 32/50
2436/2436 [==============================] - 1s 349us/step - loss: 0.0287 - val_loss: 0.0292
Epoch 33/50
2436/2436 [==============================] - 1s 334us/step - loss: 0.0286 - val_loss: 0.0291
Epoch 34/50
2436/2436 [==============================] - 1s 377us/step - loss: 0.0286 - val_loss: 0.0291
Epoch 35/50
2436/2436 [==============================] - 1s 397us/step - loss: 0.0285 - val_loss: 0.0290
Epoch 36/50
2436/2436 [==============================] - 1s 357us/step - loss: 0.0284 - val_loss: 0.0290
Epoch 37/50
2436/2436 [==============================] - 1s 370us/step - loss: 0.0284 - val_loss: 0.0289
Epoch 38/50
2436/2436 [==============================] - 1s 359us/step - loss: 0.0283 - val_loss: 0.0289
Epoch 39/50
2436/2436 [==============================] - 1s 346us/step - loss: 0.0282 - val_loss: 0.0288
Epoch 40/50
2436/2436 [==============================] - 1s 346us/step - loss: 0.0282 - val_loss: 0.0288
Epoch 41/50
2436/2436 [==============================] - 1s 392us/step - loss: 0.0281 - val_loss: 0.0287
Epoch 42/50
2436/2436 [==============================] - 1s 341us/step - loss: 0.0281 - val_loss: 0.0287
Epoch 43/50
2436/2436 [==============================] - 1s 411us/step - loss: 0.0280 - val_loss: 0.0286
Epoch 44/50
2436/2436 [==============================] - 1s 356us/step - loss: 0.0279 - val_loss: 0.0286
Epoch 45/50
2436/2436 [==============================] - 1s 363us/step - loss: 0.0279 - val_loss: 0.0285
Epoch 46/50
2436/2436 [==============================] - 1s 355us/step - loss: 0.0278 - val_loss: 0.0285
Epoch 47/50
2436/2436 [==============================] - 1s 348us/step - loss: 0.0278 - val_loss: 0.0284
Epoch 48/50
2436/2436 [==============================] - 1s 350us/step - loss: 0.0277 - val_loss: 0.0284
Epoch 49/50
2436/2436 [==============================] - 1s 355us/step - loss: 0.0276 - val_loss: 0.0284
```

```
Epoch 50/50
2436/2436 [==============================] - 1s 345us/step - loss: 0.0276 - val_loss: 0.0283
Train on 1948 samples, validate on 488 samples
Epoch 1/30
1948/1948 [==============================] - 1s 285us/step - loss: 0.0086 - accuracy: 0.3111 - val_loss: 0.007
3 - val_accuracy: 0.3689
Epoch 2/30
1948/1948 [==============================] - 0s 209us/step - loss: 0.0073 - accuracy: 0.3886 - val_loss: 0.007
0 - val_accuracy: 0.3852
Epoch 3/30
1948/1948 [==============================] - 0s 204us/step - loss: 0.0070 - accuracy: 0.4086 - val_loss: 0.006
8 - val_accuracy: 0.4447
Epoch 4/30
1948/1948 [==============================] - 0s 212us/step - loss: 0.0069 - accuracy: 0.4307 - val_loss: 0.006
6 - val_accuracy: 0.4795
Epoch 5/30
1948/1948 [==============================] - 0s 202us/step - loss: 0.0066 - accuracy: 0.4502 - val_loss: 0.007
0 - val_accuracy: 0.3893
Epoch 6/30
1948/1948 [==============================] - 0s 205us/step - loss: 0.0067 - accuracy: 0.4528 - val_loss: 0.006
8 - val_accuracy: 0.4406
Epoch 7/30
1948/1948 [==============================] - 0s 207us/step - loss: 0.0065 - accuracy: 0.4625 - val_loss: 0.006
4 - val_accuracy: 0.4508
Epoch 8/30
1948/1948 [==============================] - 0s 219us/step - loss: 0.0065 - accuracy: 0.4759 - val_loss: 0.006
4 - val_accuracy: 0.4816
Epoch 9/30
1948/1948 [==============================] - 0s 201us/step - loss: 0.0063 - accuracy: 0.4825 - val_loss: 0.006
3 - val_accuracy: 0.5061
Epoch 10/30
1948/1948 [==============================] - 0s 199us/step - loss: 0.0062 - accuracy: 0.5056 - val_loss: 0.006
5 - val_accuracy: 0.4898
Epoch 11/30
1948/1948 [==============================] - 0s 205us/step - loss: 0.0063 - accuracy: 0.4933 - val_loss: 0.006
1 - val_accuracy: 0.5287
Epoch 12/30
1948/1948 [==============================] - 0s 206us/step - loss: 0.0062 - accuracy: 0.5026 - val_loss: 0.006
3 - val_accuracy: 0.4734
Epoch 13/30
1948/1948 [==============================] - 0s 220us/step - loss: 0.0061 - accuracy: 0.5169 - val_loss: 0.006
3 - val_accuracy: 0.4795
Epoch 14/30
```

```
1948/1948 [==============================] - 0s 200us/step - loss: 0.0059 - accuracy: 0.5488 - val_loss: 0.006
2 - val_accuracy: 0.4939
813/813 [==============================] - 0s 53us/step
[0.006123718961404331, 0.5178351998329163]
```

```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```




```
Train on 2059 samples, validate on 687 samples
Epoch 1/50
2059/2059 [==============================] - 0s 215us/step - loss: 0.6778 - val_loss: 0.6400
Epoch 2/50
2059/2059 [==============================] - 0s 168us/step - loss: 0.5917 - val_loss: 0.4921
Epoch 3/50
2059/2059 [==============================] - 0s 167us/step - loss: 0.4058 - val_loss: 0.2718
Epoch 4/50
2059/2059 [==============================] - 0s 174us/step - loss: 0.1999 - val_loss: 0.1164
Epoch 5/50
2059/2059 [==============================] - 0s 176us/step - loss: 0.0876 - val_loss: 0.0597
Epoch 6/50
2059/2059 [==============================] - 0s 177us/step - loss: 0.0520 - val_loss: 0.0445
```

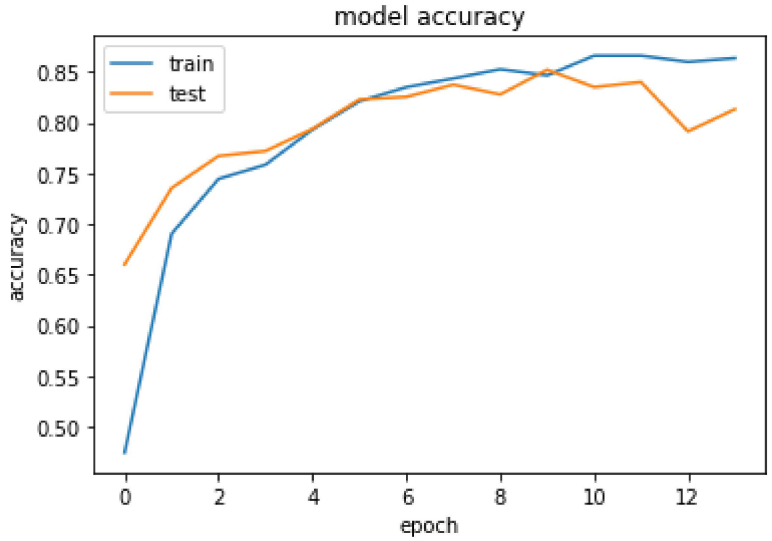
```
Epoch 7/50
2059/2059 [==============================] - 0s 171us/step - loss: 0.0425 - val_loss: 0.0403
Epoch 8/50
2059/2059 [==============================] - 0s 173us/step - loss: 0.0397 - val_loss: 0.0388
Epoch 9/50
2059/2059 [==============================] - 0s 173us/step - loss: 0.0385 - val_loss: 0.0379
Epoch 10/50
2059/2059 [==============================] - 0s 173us/step - loss: 0.0377 - val_loss: 0.0373
Epoch 11/50
2059/2059 [==============================] - 0s 172us/step - loss: 0.0372 - val_loss: 0.0369
Epoch 12/50
2059/2059 [==============================] - 0s 188us/step - loss: 0.0367 - val_loss: 0.0365
Epoch 13/50
2059/2059 [==============================] - 0s 169us/step - loss: 0.0364 - val_loss: 0.0363
Epoch 14/50
2059/2059 [==============================] - 0s 169us/step - loss: 0.0361 - val_loss: 0.0360
Epoch 15/50
2059/2059 [==============================] - 0s 177us/step - loss: 0.0359 - val_loss: 0.0358
Epoch 16/50
2059/2059 [==============================] - 0s 165us/step - loss: 0.0357 - val_loss: 0.0356
Epoch 17/50
2059/2059 [==============================] - 0s 173us/step - loss: 0.0355 - val_loss: 0.0354
Epoch 18/50
2059/2059 [==============================] - 0s 167us/step - loss: 0.0353 - val_loss: 0.0353
Epoch 19/50
2059/2059 [==============================] - 0s 176us/step - loss: 0.0351 - val_loss: 0.0351
Epoch 20/50
2059/2059 [==============================] - 0s 165us/step - loss: 0.0349 - val_loss: 0.0350
Epoch 21/50
2059/2059 [==============================] - 0s 177us/step - loss: 0.0348 - val_loss: 0.0348
Epoch 22/50
2059/2059 [==============================] - 0s 180us/step - loss: 0.0346 - val_loss: 0.0347
Epoch 23/50
2059/2059 [==============================] - 0s 171us/step - loss: 0.0345 - val_loss: 0.0346
Epoch 24/50
2059/2059 [==============================] - 0s 169us/step - loss: 0.0344 - val_loss: 0.0345
Epoch 25/50
2059/2059 [==============================] - 0s 175us/step - loss: 0.0342 - val_loss: 0.0343
Epoch 26/50
2059/2059 [==============================] - 0s 173us/step - loss: 0.0341 - val_loss: 0.0342
Epoch 27/50
2059/2059 [==============================] - 0s 169us/step - loss: 0.0340 - val_loss: 0.0341
Epoch 28/50
```

```
2059/2059 [==============================] - 0s 165us/step - loss: 0.0339 - val_loss: 0.0340
Epoch 29/50
2059/2059 [==============================] - 0s 164us/step - loss: 0.0337 - val_loss: 0.0339
Epoch 30/50
2059/2059 [==============================] - 0s 179us/step - loss: 0.0336 - val_loss: 0.0338
Epoch 31/50
2059/2059 [==============================] - 0s 166us/step - loss: 0.0335 - val_loss: 0.0337
Epoch 32/50
2059/2059 [==============================] - 0s 178us/step - loss: 0.0334 - val_loss: 0.0336
Epoch 33/50
2059/2059 [==============================] - 0s 165us/step - loss: 0.0333 - val_loss: 0.0335
Epoch 34/50
2059/2059 [==============================] - 0s 171us/step - loss: 0.0332 - val_loss: 0.0334
Epoch 35/50
2059/2059 [==============================] - 0s 184us/step - loss: 0.0331 - val_loss: 0.0333
Epoch 36/50
2059/2059 [==============================] - 0s 175us/step - loss: 0.0330 - val_loss: 0.0331
Epoch 37/50
2059/2059 [==============================] - 0s 162us/step - loss: 0.0328 - val_loss: 0.0330
Epoch 38/50
2059/2059 [==============================] - 0s 181us/step - loss: 0.0327 - val_loss: 0.0329
Epoch 39/50
2059/2059 [==============================] - 0s 164us/step - loss: 0.0326 - val_loss: 0.0328
Epoch 40/50
2059/2059 [==============================] - 0s 169us/step - loss: 0.0325 - val_loss: 0.0327
Epoch 41/50
2059/2059 [==============================] - 0s 180us/step - loss: 0.0323 - val_loss: 0.0326
Epoch 42/50
2059/2059 [==============================] - 0s 163us/step - loss: 0.0322 - val_loss: 0.0325
Epoch 43/50
2059/2059 [==============================] - 0s 177us/step - loss: 0.0321 - val_loss: 0.0323
Epoch 44/50
2059/2059 [==============================] - 0s 226us/step - loss: 0.0320 - val_loss: 0.0322
Epoch 45/50
2059/2059 [==============================] - 0s 166us/step - loss: 0.0318 - val_loss: 0.0321
Epoch 46/50
2059/2059 [==============================] - 0s 162us/step - loss: 0.0317 - val_loss: 0.0319
Epoch 47/50
2059/2059 [==============================] - 0s 161us/step - loss: 0.0315 - val_loss: 0.0318
Epoch 48/50
2059/2059 [==============================] - 0s 159us/step - loss: 0.0314 - val_loss: 0.0317
Epoch 49/50
2059/2059 [==============================] - 0s 162us/step - loss: 0.0312 - val_loss: 0.0315
```

```
Epoch 50/50
2059/2059 [==============================] - 0s 162us/step - loss: 0.0311 - val_loss: 0.0314
Train on 1647 samples, validate on 412 samples
Epoch 1/30
1647/1647 [==============================] - 1s 388us/step - loss: 0.0072 - accuracy: 0.4748 - val_loss: 0.0
050 - val_accuracy: 0.6602
Epoch 2/30
1647/1647 [==============================] - 0s 182us/step - loss: 0.0043 - accuracy: 0.6903 - val_loss: 0.0
036 - val_accuracy: 0.7354
Epoch 3/30
1647/1647 [==============================] - 0s 168us/step - loss: 0.0035 - accuracy: 0.7444 - val_loss: 0.0
032 - val_accuracy: 0.7670
Epoch 4/30
1647/1647 [==============================] - 0s 180us/step - loss: 0.0033 - accuracy: 0.7583 - val_loss: 0.0
030 - val_accuracy: 0.7718
Epoch 5/30
1647/1647 [==============================] - 0s 197us/step - loss: 0.0031 - accuracy: 0.7930 - val_loss: 0.0
029 - val_accuracy: 0.7937
Epoch 6/30
1647/1647 [==============================] - 0s 188us/step - loss: 0.0027 - accuracy: 0.8209 - val_loss: 0.0
029 - val_accuracy: 0.8228
Epoch 7/30
1647/1647 [==============================] - 0s 190us/step - loss: 0.0026 - accuracy: 0.8349 - val_loss: 0.0
029 - val_accuracy: 0.8252
Epoch 8/30
1647/1647 [==============================] - 0s 182us/step - loss: 0.0024 - accuracy: 0.8434 - val_loss: 0.0
023 - val_accuracy: 0.8374
Epoch 9/30
1647/1647 [==============================] - 0s 183us/step - loss: 0.0023 - accuracy: 0.8525 - val_loss: 0.0
026 - val_accuracy: 0.8277
Epoch 10/30
1647/1647 [==============================] - 0s 199us/step - loss: 0.0023 - accuracy: 0.8464 - val_loss: 0.0
023 - val_accuracy: 0.8519
Epoch 11/30
1647/1647 [==============================] - 0s 172us/step - loss: 0.0022 - accuracy: 0.8658 - val_loss: 0.0
022 - val_accuracy: 0.8350
Epoch 12/30
1647/1647 [==============================] - 0s 182us/step - loss: 0.0022 - accuracy: 0.8658 - val_loss: 0.0
025 - val_accuracy: 0.8398
Epoch 13/30
1647/1647 [==============================] - 0s 197us/step - loss: 0.0023 - accuracy: 0.8597 - val_loss: 0.0
029 - val_accuracy: 0.7913
Epoch 14/30
```

```
1647/1647 [==============================] - 0s 181us/step - loss: 0.0021 - accuracy: 0.8634 - val_loss: 0.0
026 - val_accuracy: 0.8131
687/687 [==============================] - 0s 67us/step
[0.0025373398159593994, 0.8253275156021118]
```

```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```




```
Train on 946 samples, validate on 316 samples
Epoch 1/50
946/946 [==============================] - 0s 434us/step - loss: 0.6892 - val_loss: 0.6785
Epoch 2/50
946/946 [==============================] - 0s 346us/step - loss: 0.6685 - val_loss: 0.6430
Epoch 3/50
946/946 [==============================] - 0s 345us/step - loss: 0.6210 - val_loss: 0.5708
Epoch 4/50
946/946 [==============================] - 0s 376us/step - loss: 0.5340 - val_loss: 0.4574
Epoch 5/50
946/946 [==============================] - 0s 363us/step - loss: 0.4103 - val_loss: 0.3210
Epoch 6/50
946/946 [==============================] - 0s 364us/step - loss: 0.2760 - val_loss: 0.1975
```

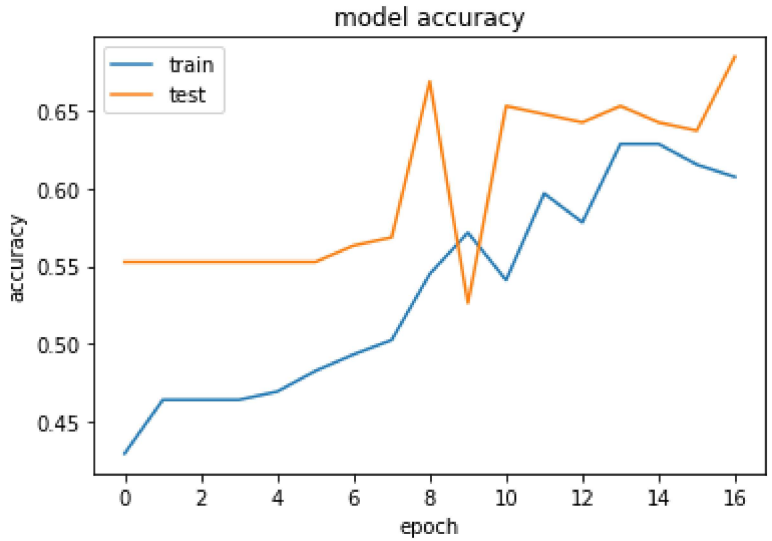
```
Epoch 7/50
946/946 [==============================] - 0s 350us/step - loss: 0.1653 - val_loss: 0.1134
Epoch 8/50
946/946 [==============================] - 0s 357us/step - loss: 0.0952 - val_loss: 0.0685
Epoch 9/50
946/946 [==============================] - 0s 368us/step - loss: 0.0599 - val_loss: 0.0480
Epoch 10/50
946/946 [==============================] - 0s 363us/step - loss: 0.0444 - val_loss: 0.0393
Epoch 11/50
946/946 [==============================] - 0s 375us/step - loss: 0.0377 - val_loss: 0.0358
Epoch 12/50
946/946 [==============================] - 0s 360us/step - loss: 0.0351 - val_loss: 0.0344
Epoch 13/50
946/946 [==============================] - 0s 354us/step - loss: 0.0340 - val_loss: 0.0339
Epoch 14/50
946/946 [==============================] - 0s 358us/step - loss: 0.0336 - val_loss: 0.0336
Epoch 15/50
946/946 [==============================] - 0s 388us/step - loss: 0.0333 - val_loss: 0.0334
Epoch 16/50
946/946 [==============================] - 0s 375us/step - loss: 0.0330 - val_loss: 0.0332
Epoch 17/50
946/946 [==============================] - 0s 351us/step - loss: 0.0328 - val_loss: 0.0330
Epoch 18/50
946/946 [==============================] - 0s 355us/step - loss: 0.0326 - val_loss: 0.0328
Epoch 19/50
946/946 [==============================] - 0s 389us/step - loss: 0.0324 - val_loss: 0.0326
Epoch 20/50
946/946 [==============================] - 0s 352us/step - loss: 0.0322 - val_loss: 0.0324
Epoch 21/50
946/946 [==============================] - 0s 361us/step - loss: 0.0320 - val_loss: 0.0323
Epoch 22/50
946/946 [==============================] - 0s 373us/step - loss: 0.0319 - val_loss: 0.0322
Epoch 23/50
946/946 [==============================] - 0s 348us/step - loss: 0.0317 - val_loss: 0.0320
Epoch 24/50
946/946 [==============================] - 0s 350us/step - loss: 0.0316 - val_loss: 0.0319
Epoch 25/50
946/946 [==============================] - 0s 351us/step - loss: 0.0315 - val_loss: 0.0318
Epoch 26/50
946/946 [==============================] - 0s 363us/step - loss: 0.0313 - val_loss: 0.0317
Epoch 27/50
946/946 [==============================] - 0s 338us/step - loss: 0.0312 - val_loss: 0.0316
Epoch 28/50
```

```
946/946 [==============================] - 0s 352us/step - loss: 0.0311 - val_loss: 0.0316
Epoch 29/50
946/946 [==============================] - 0s 382us/step - loss: 0.0311 - val_loss: 0.0315
Epoch 30/50
946/946 [==============================] - 0s 352us/step - loss: 0.0310 - val_loss: 0.0314
Epoch 31/50
946/946 [==============================] - 0s 351us/step - loss: 0.0309 - val_loss: 0.0314
Epoch 32/50
946/946 [==============================] - 0s 342us/step - loss: 0.0308 - val_loss: 0.0313
Epoch 33/50
946/946 [==============================] - 0s 378us/step - loss: 0.0308 - val_loss: 0.0312
Epoch 34/50
946/946 [==============================] - 0s 427us/step - loss: 0.0307 - val_loss: 0.0312
Epoch 35/50
946/946 [==============================] - 0s 369us/step - loss: 0.0306 - val_loss: 0.0311
Epoch 36/50
946/946 [==============================] - 0s 365us/step - loss: 0.0306 - val_loss: 0.0311
Epoch 37/50
946/946 [==============================] - 0s 341us/step - loss: 0.0305 - val_loss: 0.0310
Epoch 38/50
946/946 [==============================] - 0s 363us/step - loss: 0.0304 - val_loss: 0.0310
Epoch 39/50
946/946 [==============================] - 0s 335us/step - loss: 0.0304 - val_loss: 0.0309
Epoch 40/50
946/946 [==============================] - 0s 347us/step - loss: 0.0303 - val_loss: 0.0309
Epoch 41/50
946/946 [==============================] - 0s 347us/step - loss: 0.0303 - val_loss: 0.0308
Epoch 42/50
946/946 [==============================] - 0s 368us/step - loss: 0.0302 - val_loss: 0.0308
Epoch 43/50
946/946 [==============================] - 0s 361us/step - loss: 0.0301 - val_loss: 0.0307
Epoch 44/50
946/946 [==============================] - 0s 371us/step - loss: 0.0301 - val_loss: 0.0307
Epoch 45/50
946/946 [==============================] - 0s 346us/step - loss: 0.0300 - val_loss: 0.0306
Epoch 46/50
946/946 [==============================] - 0s 347us/step - loss: 0.0300 - val_loss: 0.0306
Epoch 47/50
946/946 [==============================] - 0s 353us/step - loss: 0.0299 - val_loss: 0.0306
Epoch 48/50
946/946 [==============================] - 0s 344us/step - loss: 0.0299 - val_loss: 0.0305
Epoch 49/50
946/946 [==============================] - 0s 351us/step - loss: 0.0298 - val_loss: 0.0305
```

```
Epoch 50/50
946/946 [==============================] - 0s 329us/step - loss: 0.0298 - val_loss: 0.0304
Train on 756 samples, validate on 190 samples
Epoch 1/30
756/756 [==============================] - 0s 364us/step - loss: 0.0086 - accuracy: 0.4299 - val_loss: 0.0066
- val_accuracy: 0.5526
Epoch 2/30
756/756 [==============================] - 0s 203us/step - loss: 0.0071 - accuracy: 0.4643 - val_loss: 0.0063
- val_accuracy: 0.5526
Epoch 3/30
756/756 [==============================] - 0s 223us/step - loss: 0.0070 - accuracy: 0.4643 - val_loss: 0.0062
- val_accuracy: 0.5526
Epoch 4/30
756/756 [==============================] - 0s 222us/step - loss: 0.0069 - accuracy: 0.4643 - val_loss: 0.0060
- val_accuracy: 0.5526
Epoch 5/30
756/756 [==============================] - 0s 208us/step - loss: 0.0068 - accuracy: 0.4696 - val_loss: 0.0059
- val_accuracy: 0.5526
Epoch 6/30
756/756 [==============================] - 0s 223us/step - loss: 0.0065 - accuracy: 0.4828 - val_loss: 0.0056
- val_accuracy: 0.5526
Epoch 7/30
756/756 [==============================] - 0s 210us/step - loss: 0.0068 - accuracy: 0.4934 - val_loss: 0.0059
- val_accuracy: 0.5632
Epoch 8/30
756/756 [==============================] - 0s 245us/step - loss: 0.0062 - accuracy: 0.5026 - val_loss: 0.0052
- val_accuracy: 0.5684
Epoch 9/30
756/756 [==============================] - 0s 228us/step - loss: 0.0059 - accuracy: 0.5450 - val_loss: 0.0052
- val_accuracy: 0.6684
Epoch 10/30
756/756 [==============================] - 0s 206us/step - loss: 0.0055 - accuracy: 0.5714 - val_loss: 0.0068
- val_accuracy: 0.5263
Epoch 11/30
756/756 [==============================] - 0s 228us/step - loss: 0.0058 - accuracy: 0.5410 - val_loss: 0.0046
- val_accuracy: 0.6526
Epoch 12/30
756/756 [==============================] - 0s 239us/step - loss: 0.0052 - accuracy: 0.5966 - val_loss: 0.0044
- val_accuracy: 0.6474
Epoch 13/30
756/756 [==============================] - 0s 220us/step - loss: 0.0054 - accuracy: 0.5780 - val_loss: 0.0046
- val_accuracy: 0.6421
Epoch 14/30
```

```
756/756 [==============================] - 0s 210us/step - loss: 0.0051 - accuracy: 0.6283 - val_loss: 0.0044
- val_accuracy: 0.6526
Epoch 15/30
756/756 [==============================] - 0s 210us/step - loss: 0.0050 - accuracy: 0.6283 - val_loss: 0.0045
- val_accuracy: 0.6421
Epoch 16/30
756/756 [==============================] - 0s 203us/step - loss: 0.0051 - accuracy: 0.6151 - val_loss: 0.0046
- val_accuracy: 0.6368
Epoch 17/30
756/756 [==============================] - 0s 224us/step - loss: 0.0054 - accuracy: 0.6071 - val_loss: 0.0050
- val_accuracy: 0.6842
316/316 [==============================] - 0s 54us/step
[0.005435371482626924, 0.6265822649002075]
```

```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

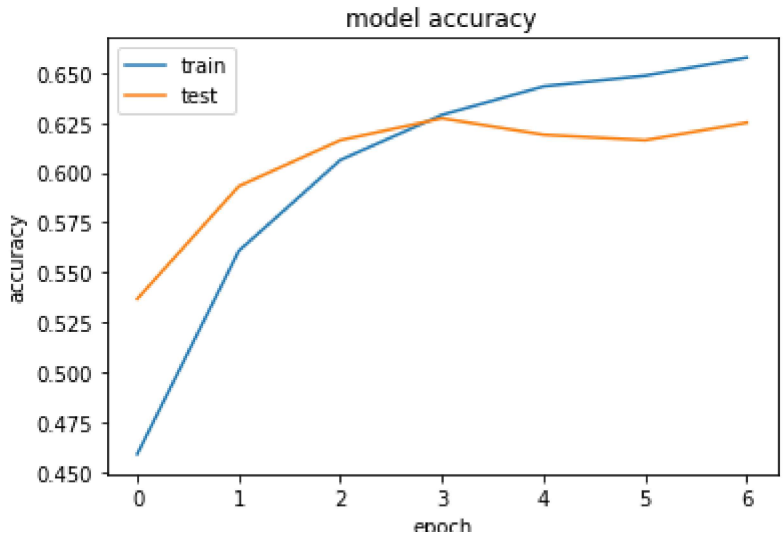



```
Train on 9119 samples, validate on 3040 samples
Epoch 1/50
9119/9119 [==============================] - 4s 452us/step - loss: 0.3739 - val_loss: 0.0409
Epoch 2/50
9119/9119 [==============================] - 4s 443us/step - loss: 0.0306 - val_loss: 0.0288
```

```
Epoch 3/50
9119/9119 [==============================] - 4s 427us/step - loss: 0.0282 - val_loss: 0.0279
Epoch 4/50
9119/9119 [==============================] - 4s 440us/step - loss: 0.0274 - val_loss: 0.0273
Epoch 5/50
9119/9119 [==============================] - 4s 441us/step - loss: 0.0269 - val_loss: 0.0269
Epoch 6/50
9119/9119 [==============================] - 4s 477us/step - loss: 0.0265 - val_loss: 0.0265
Epoch 7/50
9119/9119 [==============================] - 4s 464us/step - loss: 0.0262 - val_loss: 0.0262
Epoch 8/50
9119/9119 [==============================] - 4s 437us/step - loss: 0.0259 - val_loss: 0.0259
Epoch 9/50
9119/9119 [==============================] - 4s 432us/step - loss: 0.0256 - val_loss: 0.0257
Epoch 10/50
9119/9119 [==============================] - 4s 436us/step - loss: 0.0254 - val_loss: 0.0255
Epoch 11/50
9119/9119 [==============================] - 4s 442us/step - loss: 0.0252 - val_loss: 0.0253
Epoch 12/50
9119/9119 [==============================] - 4s 447us/step - loss: 0.0250 - val_loss: 0.0251
Epoch 13/50
9119/9119 [==============================] - 4s 448us/step - loss: 0.0249 - val_loss: 0.0250
Epoch 14/50
9119/9119 [==============================] - 4s 445us/step - loss: 0.0247 - val_loss: 0.0249
Epoch 15/50
9119/9119 [==============================] - 4s 423us/step - loss: 0.0246 - val_loss: 0.0248
Epoch 16/50
9119/9119 [==============================] - 4s 417us/step - loss: 0.0245 - val_loss: 0.0247
Epoch 17/50
9119/9119 [==============================] - 4s 420us/step - loss: 0.0244 - val_loss: 0.0246
Epoch 18/50
9119/9119 [==============================] - 4s 464us/step - loss: 0.0243 - val_loss: 0.0245
Epoch 19/50
9119/9119 [==============================] - 4s 454us/step - loss: 0.0242 - val_loss: 0.0244
Epoch 20/50
9119/9119 [==============================] - 5s 518us/step - loss: 0.0241 - val_loss: 0.0243
Epoch 21/50
9119/9119 [==============================] - 4s 462us/step - loss: 0.0239 - val_loss: 0.0242
Epoch 22/50
9119/9119 [==============================] - 4s 460us/step - loss: 0.0238 - val_loss: 0.0240
Epoch 23/50
9119/9119 [==============================] - 4s 441us/step - loss: 0.0237 - val_loss: 0.0239
Epoch 24/50
```

```
9119/9119 [==============================] - 4s 444us/step - loss: 0.0235 - val_loss: 0.0238
Epoch 25/50
9119/9119 [==============================] - 4s 463us/step - loss: 0.0234 - val_loss: 0.0236
Epoch 26/50
9119/9119 [==============================] - 7s 815us/step - loss: 0.0233 - val_loss: 0.0235
Epoch 27/50
9119/9119 [==============================] - 4s 481us/step - loss: 0.0231 - val_loss: 0.0233
Epoch 28/50
9119/9119 [==============================] - 6s 635us/step - loss: 0.0230 - val_loss: 0.0232
Epoch 29/50
9119/9119 [==============================] - 6s 624us/step - loss: 0.0228 - val_loss: 0.0231
Epoch 30/50
9119/9119 [==============================] - 7s 744us/step - loss: 0.0227 - val_loss: 0.0229
Epoch 31/50
9119/9119 [==============================] - 6s 613us/step - loss: 0.0226 - val_loss: 0.0228
Epoch 32/50
9119/9119 [==============================] - 5s 594us/step - loss: 0.0224 - val_loss: 0.0227
Epoch 33/50
9119/9119 [==============================] - 7s 775us/step - loss: 0.0223 - val_loss: 0.0225
Epoch 34/50
9119/9119 [==============================] - 7s 785us/step - loss: 0.0222 - val_loss: 0.0224
Epoch 35/50
9119/9119 [==============================] - 6s 648us/step - loss: 0.0220 - val_loss: 0.0223
Epoch 36/50
9119/9119 [==============================] - 4s 462us/step - loss: 0.0219 - val_loss: 0.0221
Epoch 37/50
9119/9119 [==============================] - 4s 434us/step - loss: 0.0217 - val_loss: 0.0220
Epoch 38/50
9119/9119 [==============================] - 5s 525us/step - loss: 0.0216 - val_loss: 0.0218
Epoch 39/50
9119/9119 [==============================] - 4s 441us/step - loss: 0.0214 - val_loss: 0.0217
Epoch 40/50
9119/9119 [==============================] - 4s 435us/step - loss: 0.0213 - val_loss: 0.0215
Epoch 41/50
9119/9119 [==============================] - 4s 442us/step - loss: 0.0211 - val_loss: 0.0214
Epoch 42/50
9119/9119 [==============================] - 4s 427us/step - loss: 0.0210 - val_loss: 0.0212
Epoch 43/50
9119/9119 [==============================] - 4s 432us/step - loss: 0.0208 - val_loss: 0.0211
Epoch 44/50
9119/9119 [==============================] - 4s 437us/step - loss: 0.0207 - val_loss: 0.0210
Epoch 45/50
9119/9119 [==============================] - 4s 451us/step - loss: 0.0205 - val_loss: 0.0208
```

```
Epoch 46/50
9119/9119 [==============================] - 7s 757us/step - loss: 0.0204 - val_loss: 0.0207
Epoch 47/50
9119/9119 [==============================] - 5s 530us/step - loss: 0.0202 - val_loss: 0.0205
Epoch 48/50
9119/9119 [==============================] - 7s 795us/step - loss: 0.0201 - val_loss: 0.0204
Epoch 49/50
9119/9119 [==============================] - 5s 541us/step - loss: 0.0200 - val_loss: 0.0203
Epoch 50/50
9119/9119 [==============================] - 5s 516us/step - loss: 0.0198 - val_loss: 0.0201
Train on 7295 samples, validate on 1824 samples
Epoch 1/30
7295/7295 [==============================] - 2s 237us/step - loss: 0.0067 - accuracy: 0.4589 - val_loss: 0.0
058 - val_accuracy: 0.5367
Epoch 2/30
7295/7295 [==============================] - 2s 221us/step - loss: 0.0056 - accuracy: 0.5608 - val_loss: 0.0
054 - val_accuracy: 0.5932
Epoch 3/30
7295/7295 [==============================] - 2s 227us/step - loss: 0.0052 - accuracy: 0.6063 - val_loss: 0.0
051 - val_accuracy: 0.6162
Epoch 4/30
7295/7295 [==============================] - 2s 211us/step - loss: 0.0050 - accuracy: 0.6289 - val_loss: 0.0
050 - val_accuracy: 0.6272
Epoch 5/30
7295/7295 [==============================] - 2s 242us/step - loss: 0.0048 - accuracy: 0.6432 - val_loss: 0.0
051 - val_accuracy: 0.6190
Epoch 6/30
7295/7295 [==============================] - 2s 234us/step - loss: 0.0047 - accuracy: 0.6485 - val_loss: 0.0
050 - val_accuracy: 0.6162
Epoch 7/30
7295/7295 [==============================] - 2s 247us/step - loss: 0.0046 - accuracy: 0.6576 - val_loss: 0.0
050 - val_accuracy: 0.6250
3040/3040 [==============================] - 0s 57us/step
[0.005010414922511891, 0.616447389125824]



dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```



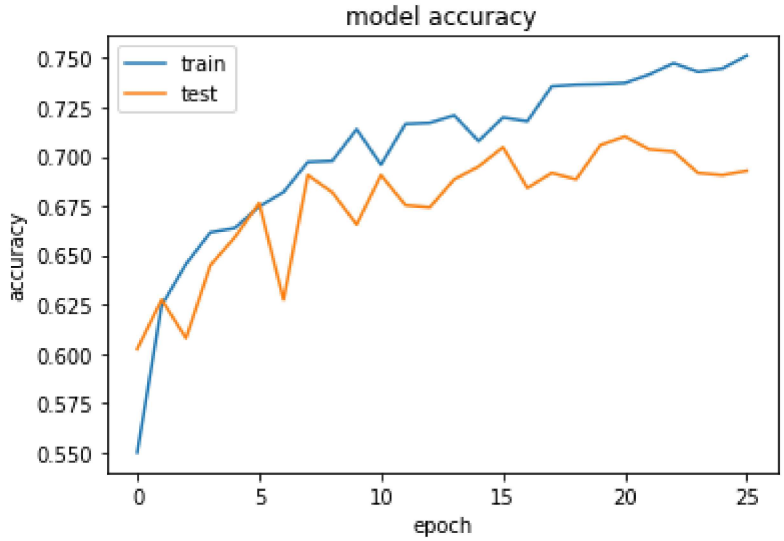

```
Train on 4588 samples, validate on 1530 samples
Epoch 1/50
4588/4588 [==============================] - 4s 786us/step - loss: 0.5585 - val_loss: 0.2294
Epoch 2/50
4588/4588 [==============================] - 3s 730us/step - loss: 0.0754 - val_loss: 0.0278
Epoch 3/50
4588/4588 [==============================] - 3s 732us/step - loss: 0.0285 - val_loss: 0.0287
Epoch 4/50
4588/4588 [==============================] - 3s 744us/step - loss: 0.0284 - val_loss: 0.0275
Epoch 5/50
4588/4588 [==============================] - 4s 768us/step - loss: 0.0272 - val_loss: 0.0267
Epoch 6/50
4588/4588 [==============================] - 3s 752us/step - loss: 0.0266 - val_loss: 0.0264
Epoch 7/50
4588/4588 [==============================] - 4s 773us/step - loss: 0.0264 - val_loss: 0.0262
Epoch 8/50
4588/4588 [==============================] - 3s 756us/step - loss: 0.0262 - val_loss: 0.0262
Epoch 9/50
4588/4588 [==============================] - 3s 718us/step - loss: 0.0261 - val_loss: 0.0261
Epoch 10/50
4588/4588 [==============================] - 3s 740us/step - loss: 0.0261 - val_loss: 0.0260
Epoch 11/50
4588/4588 [==============================] - 3s 740us/step - loss: 0.0260 - val_loss: 0.0260
```

```
Epoch 12/50
4588/4588 [==============================] - 3s 751us/step - loss: 0.0259 - val_loss: 0.0259
Epoch 13/50
4588/4588 [==============================] - 4s 771us/step - loss: 0.0258 - val_loss: 0.0258
Epoch 14/50
4588/4588 [==============================] - 4s 782us/step - loss: 0.0257 - val_loss: 0.0258
Epoch 15/50
4588/4588 [==============================] - 3s 723us/step - loss: 0.0257 - val_loss: 0.0257
Epoch 16/50
4588/4588 [==============================] - 3s 726us/step - loss: 0.0256 - val_loss: 0.0256
Epoch 17/50
4588/4588 [==============================] - 3s 731us/step - loss: 0.0255 - val_loss: 0.0256
Epoch 18/50
4588/4588 [==============================] - 4s 768us/step - loss: 0.0254 - val_loss: 0.0255
Epoch 19/50
4588/4588 [==============================] - 4s 786us/step - loss: 0.0254 - val_loss: 0.0255
Epoch 20/50
4588/4588 [==============================] - 3s 739us/step - loss: 0.0253 - val_loss: 0.0254
Epoch 21/50
4588/4588 [==============================] - 3s 760us/step - loss: 0.0252 - val_loss: 0.0253
Epoch 22/50
4588/4588 [==============================] - 4s 764us/step - loss: 0.0251 - val_loss: 0.0253
Epoch 23/50
4588/4588 [==============================] - 3s 725us/step - loss: 0.0251 - val_loss: 0.0252
Epoch 24/50
4588/4588 [==============================] - 3s 707us/step - loss: 0.0250 - val_loss: 0.0251
Epoch 25/50
4588/4588 [==============================] - 3s 731us/step - loss: 0.0249 - val_loss: 0.0251
Epoch 26/50
4588/4588 [==============================] - 3s 760us/step - loss: 0.0248 - val_loss: 0.0250
Epoch 27/50
4588/4588 [==============================] - 3s 722us/step - loss: 0.0247 - val_loss: 0.0249
Epoch 28/50
4588/4588 [==============================] - 3s 745us/step - loss: 0.0246 - val_loss: 0.0248
Epoch 29/50
4588/4588 [==============================] - 3s 749us/step - loss: 0.0246 - val_loss: 0.0248
Epoch 30/50
4588/4588 [==============================] - 3s 758us/step - loss: 0.0245 - val_loss: 0.0247
Epoch 31/50
4588/4588 [==============================] - 4s 788us/step - loss: 0.0244 - val_loss: 0.0246
Epoch 32/50
4588/4588 [==============================] - 3s 751us/step - loss: 0.0243 - val_loss: 0.0245
Epoch 33/50
```

```
4588/4588 [==============================] - 3s 740us/step - loss: 0.0242 - val_loss: 0.0244
Epoch 34/50
4588/4588 [==============================] - 3s 713us/step - loss: 0.0241 - val_loss: 0.0243
Epoch 35/50
4588/4588 [==============================] - 3s 699us/step - loss: 0.0240 - val_loss: 0.0243
Epoch 36/50
4588/4588 [==============================] - 3s 690us/step - loss: 0.0239 - val_loss: 0.0242
Epoch 37/50
4588/4588 [==============================] - 3s 750us/step - loss: 0.0238 - val_loss: 0.0241
Epoch 38/50
4588/4588 [==============================] - 3s 692us/step - loss: 0.0237 - val_loss: 0.0240
Epoch 39/50
4588/4588 [==============================] - 3s 690us/step - loss: 0.0236 - val_loss: 0.0239
Epoch 40/50
4588/4588 [==============================] - 3s 696us/step - loss: 0.0235 - val_loss: 0.0239
Epoch 41/50
4588/4588 [==============================] - 3s 693us/step - loss: 0.0235 - val_loss: 0.0238
Epoch 42/50
4588/4588 [==============================] - 3s 693us/step - loss: 0.0234 - val_loss: 0.0237
Epoch 43/50
4588/4588 [==============================] - 3s 690us/step - loss: 0.0233 - val_loss: 0.0236
Epoch 44/50
4588/4588 [==============================] - 3s 701us/step - loss: 0.0232 - val_loss: 0.0235
Epoch 45/50
4588/4588 [==============================] - 3s 690us/step - loss: 0.0231 - val_loss: 0.0235
Epoch 46/50
4588/4588 [==============================] - 3s 692us/step - loss: 0.0230 - val_loss: 0.0234
Epoch 47/50
4588/4588 [==============================] - 3s 693us/step - loss: 0.0230 - val_loss: 0.0233
Epoch 48/50
4588/4588 [==============================] - 3s 689us/step - loss: 0.0229 - val_loss: 0.0233
Epoch 49/50
4588/4588 [==============================] - 3s 742us/step - loss: 0.0228 - val_loss: 0.0232
Epoch 50/50
4588/4588 [==============================] - 3s 760us/step - loss: 0.0227 - val_loss: 0.0231
Train on 3670 samples, validate on 918 samples
Epoch 1/30
3670/3670 [==============================] - 2s 567us/step - loss: 0.0061 - accuracy: 0.5499 - val_loss: 0.0
051 - val_accuracy: 0.6024
Epoch 2/30
3670/3670 [==============================] - 2s 503us/step - loss: 0.0051 - accuracy: 0.6245 - val_loss: 0.0
049 - val_accuracy: 0.6275
Epoch 3/30
```

```
3670/3670 [==============================] - 2s 507us/step - loss: 0.0048 - accuracy: 0.6455 - val_loss: 0.0
051 - val_accuracy: 0.6078
Epoch 4/30
3670/3670 [==============================] - 2s 480us/step - loss: 0.0046 - accuracy: 0.6616 - val_loss: 0.0
046 - val_accuracy: 0.6449
Epoch 5/30
3670/3670 [==============================] - 2s 418us/step - loss: 0.0045 - accuracy: 0.6638 - val_loss: 0.0
045 - val_accuracy: 0.6590
Epoch 6/30
3670/3670 [==============================] - 1s 320us/step - loss: 0.0044 - accuracy: 0.6747 - val_loss: 0.0
045 - val_accuracy: 0.6765
Epoch 7/30
3670/3670 [==============================] - 1s 281us/step - loss: 0.0044 - accuracy: 0.6820 - val_loss: 0.0
049 - val_accuracy: 0.6275
Epoch 8/30
3670/3670 [==============================] - 1s 281us/step - loss: 0.0043 - accuracy: 0.6973 - val_loss: 0.0
043 - val_accuracy: 0.6906
Epoch 9/30
3670/3670 [==============================] - 1s 327us/step - loss: 0.0042 - accuracy: 0.6978 - val_loss: 0.0
045 - val_accuracy: 0.6819
Epoch 10/30
3670/3670 [==============================] - 1s 273us/step - loss: 0.0040 - accuracy: 0.7139 - val_loss: 0.0
044 - val_accuracy: 0.6656
Epoch 11/30
3670/3670 [==============================] - 1s 306us/step - loss: 0.0042 - accuracy: 0.6959 - val_loss: 0.0
042 - val_accuracy: 0.6906
Epoch 12/30
3670/3670 [==============================] - 1s 292us/step - loss: 0.0040 - accuracy: 0.7166 - val_loss: 0.0
042 - val_accuracy: 0.6754
Epoch 13/30
3670/3670 [==============================] - 1s 292us/step - loss: 0.0040 - accuracy: 0.7172 - val_loss: 0.0
043 - val_accuracy: 0.6743
Epoch 14/30
3670/3670 [==============================] - 1s 350us/step - loss: 0.0039 - accuracy: 0.7210 - val_loss: 0.0
043 - val_accuracy: 0.6885
Epoch 15/30
3670/3670 [==============================] - 1s 293us/step - loss: 0.0040 - accuracy: 0.7079 - val_loss: 0.0
041 - val_accuracy: 0.6950
Epoch 16/30
3670/3670 [==============================] - 1s 331us/step - loss: 0.0039 - accuracy: 0.7199 - val_loss: 0.0
040 - val_accuracy: 0.7048
Epoch 17/30
3670/3670 [==============================] - 1s 306us/step - loss: 0.0039 - accuracy: 0.7180 - val_loss: 0.0
```

```
043 - val_accuracy: 0.6841
Epoch 18/30
3670/3670 [==============================] - 1s 293us/step - loss: 0.0037 - accuracy: 0.7357 - val_loss: 0.0
040 - val_accuracy: 0.6917
Epoch 19/30
3670/3670 [==============================] - 1s 295us/step - loss: 0.0037 - accuracy: 0.7365 - val_loss: 0.0
042 - val_accuracy: 0.6885
Epoch 20/30

3670/3670 [==============================] - 1s 253us/step - loss: 0.0037 - accuracy: 0.7368 - val_loss: 0.004
0 - val_accuracy: 0.7059
Epoch 21/30
3670/3670 [==============================] - 1s 288us/step - loss: 0.0037 - accuracy: 0.7373 - val_loss: 0.004
0 - val_accuracy: 0.7102
Epoch 22/30
3670/3670 [==============================] - 1s 277us/step - loss: 0.0036 - accuracy: 0.7417 - val_loss: 0.004
1 - val_accuracy: 0.7037
Epoch 23/30
3670/3670 [==============================] - 1s 356us/step - loss: 0.0036 - accuracy: 0.7474 - val_loss: 0.003
9 - val_accuracy: 0.7026
Epoch 24/30
3670/3670 [==============================] - 1s 396us/step - loss: 0.0036 - accuracy: 0.7431 - val_loss: 0.004
1 - val_accuracy: 0.6917
Epoch 25/30
3670/3670 [==============================] - 1s 391us/step - loss: 0.0036 - accuracy: 0.7447 - val_loss: 0.004
1 - val_accuracy: 0.6906
Epoch 26/30
3670/3670 [==============================] - 1s 356us/step - loss: 0.0035 - accuracy: 0.7512 - val_loss: 0.004
1 - val_accuracy: 0.6928
1530/1530 [==============================] - 0s 211us/step
[0.004245807698264328, 0.6836601495742798]


dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```




```
Train on 2272 samples, validate on 758 samples
Epoch 1/50
2272/2272 [==============================] - 3s 1ms/step - loss: 0.6637 - val_loss: 0.5778
Epoch 2/50
2272/2272 [==============================] - 2s 945us/step - loss: 0.4080 - val_loss: 0.1876
Epoch 3/50
2272/2272 [==============================] - 2s 964us/step - loss: 0.0914 - val_loss: 0.0362
Epoch 4/50
2272/2272 [==============================] - 2s 904us/step - loss: 0.0294 - val_loss: 0.0274
Epoch 5/50
2272/2272 [==============================] - 2s 910us/step - loss: 0.0280 - val_loss: 0.0282
Epoch 6/50
2272/2272 [==============================] - 2s 996us/step - loss: 0.0287 - val_loss: 0.0281
Epoch 7/50
2272/2272 [==============================] - 2s 879us/step - loss: 0.0283 - val_loss: 0.0275
Epoch 8/50
2272/2272 [==============================] - 2s 1ms/step - loss: 0.0275 - val_loss: 0.0268
Epoch 9/50
```

```
2272/2272 [==============================] - 2s 845us/step - loss: 0.0268 - val_loss: 0.0263
Epoch 10/50
2272/2272 [==============================] - 2s 866us/step - loss: 0.0263 - val_loss: 0.0260
Epoch 11/50
2272/2272 [==============================] - 2s 974us/step - loss: 0.0260 - val_loss: 0.0258
Epoch 12/50
2272/2272 [==============================] - 2s 862us/step - loss: 0.0258 - val_loss: 0.0257
Epoch 13/50
2272/2272 [==============================] - 2s 955us/step - loss: 0.0256 - val_loss: 0.0257
Epoch 14/50
2272/2272 [==============================] - 2s 1ms/step - loss: 0.0255 - val_loss: 0.0256
Epoch 15/50
2272/2272 [==============================] - 2s 892us/step - loss: 0.0255 - val_loss: 0.0256
Epoch 16/50
2272/2272 [==============================] - 2s 982us/step - loss: 0.0254 - val_loss: 0.0255
Epoch 17/50
2272/2272 [==============================] - 2s 879us/step - loss: 0.0253 - val_loss: 0.0255
Epoch 18/50
2272/2272 [==============================] - 2s 891us/step - loss: 0.0253 - val_loss: 0.0254
Epoch 19/50
2272/2272 [==============================] - 2s 864us/step - loss: 0.0252 - val_loss: 0.0254
Epoch 20/50
2272/2272 [==============================] - 2s 935us/step - loss: 0.0252 - val_loss: 0.0253
Epoch 21/50
2272/2272 [==============================] - 3s 1ms/step - loss: 0.0251 - val_loss: 0.0253
Epoch 22/50
2272/2272 [==============================] - 2s 1ms/step - loss: 0.0251 - val_loss: 0.0252
Epoch 23/50
2272/2272 [==============================] - 2s 908us/step - loss: 0.0250 - val_loss: 0.0252
Epoch 24/50
2272/2272 [==============================] - 2s 903us/step - loss: 0.0250 - val_loss: 0.0252
Epoch 25/50
2272/2272 [==============================] - 2s 858us/step - loss: 0.0249 - val_loss: 0.0251
Epoch 26/50
2272/2272 [==============================] - 2s 832us/step - loss: 0.0249 - val_loss: 0.0251
Epoch 27/50
2272/2272 [==============================] - 2s 820us/step - loss: 0.0248 - val_loss: 0.0250
Epoch 28/50
2272/2272 [==============================] - 2s 911us/step - loss: 0.0248 - val_loss: 0.0250
Epoch 29/50
2272/2272 [==============================] - 2s 1ms/step - loss: 0.0247 - val_loss: 0.0249
Epoch 30/50
2272/2272 [==============================] - 3s 1ms/step - loss: 0.0247 - val_loss: 0.0249
```

```
Epoch 31/50
2272/2272 [==============================] - 2s 932us/step - loss: 0.0246 - val_loss: 0.0248
Epoch 32/50
2272/2272 [==============================] - 2s 954us/step - loss: 0.0246 - val_loss: 0.0248
Epoch 33/50
2272/2272 [==============================] - 2s 988us/step - loss: 0.0245 - val_loss: 0.0248
Epoch 34/50
2272/2272 [==============================] - 2s 903us/step - loss: 0.0244 - val_loss: 0.0247
Epoch 35/50
2272/2272 [==============================] - 2s 966us/step - loss: 0.0244 - val_loss: 0.0247
Epoch 36/50
2272/2272 [==============================] - 2s 1ms/step - loss: 0.0243 - val_loss: 0.0246
Epoch 37/50
2272/2272 [==============================] - 2s 864us/step - loss: 0.0243 - val_loss: 0.0246
Epoch 38/50
2272/2272 [==============================] - 2s 885us/step - loss: 0.0242 - val_loss: 0.0245
Epoch 39/50
2272/2272 [==============================] - 2s 966us/step - loss: 0.0241 - val_loss: 0.0244
Epoch 40/50
2272/2272 [==============================] - 2s 977us/step - loss: 0.0241 - val_loss: 0.0244
Epoch 41/50
2272/2272 [==============================] - 2s 849us/step - loss: 0.0240 - val_loss: 0.0243
Epoch 42/50
2272/2272 [==============================] - 2s 884us/step - loss: 0.0239 - val_loss: 0.0243
Epoch 43/50
2272/2272 [==============================] - 3s 1ms/step - loss: 0.0239 - val_loss: 0.0242
Epoch 44/50
2272/2272 [==============================] - 3s 1ms/step - loss: 0.0238 - val_loss: 0.0241
Epoch 45/50
2272/2272 [==============================] - 2s 870us/step - loss: 0.0237 - val_loss: 0.0241
Epoch 46/50
2272/2272 [==============================] - 2s 881us/step - loss: 0.0236 - val_loss: 0.0240
Epoch 47/50
2272/2272 [==============================] - 2s 906us/step - loss: 0.0236 - val_loss: 0.0239
Epoch 48/50
2272/2272 [==============================] - 2s 1ms/step - loss: 0.0235 - val_loss: 0.0239
Epoch 49/50
2272/2272 [==============================] - 2s 890us/step - loss: 0.0234 - val_loss: 0.0238
Epoch 50/50
2272/2272 [==============================] - 2s 919us/step - loss: 0.0234 - val_loss: 0.0237
Train on 1817 samples, validate on 455 samples
Epoch 1/30
1817/1817 [==============================] - 1s 403us/step - loss: 0.0026 - accuracy: 0.9053 - val_loss: 0.001
```

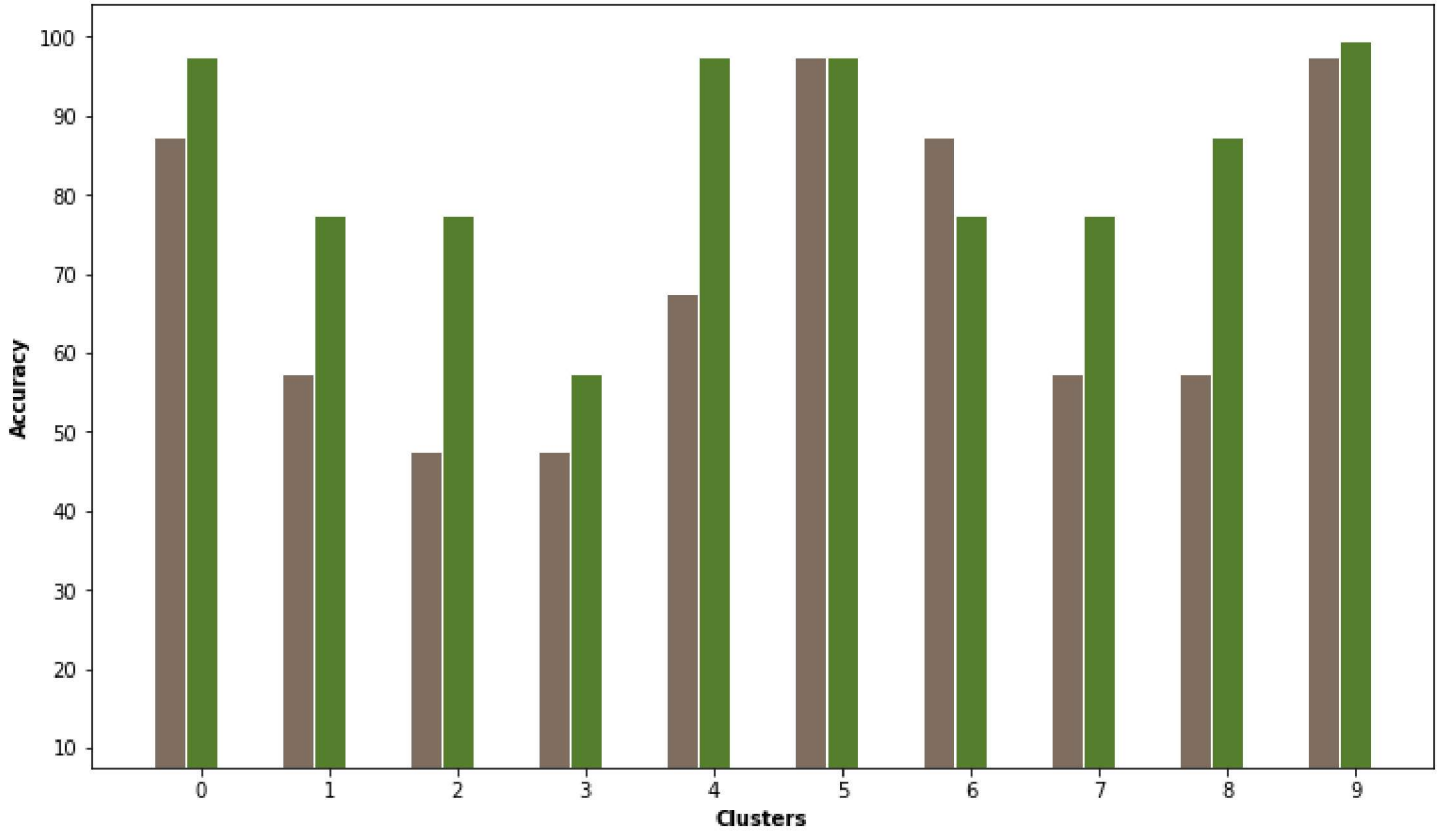
```
6 - val_accuracy: 0.9165
Epoch 2/30
1817/1817 [==============================] - 1s 375us/step - loss: 0.0014 - accuracy: 0.9296 - val_loss: 0.001
6 - val_accuracy: 0.9165
Epoch 3/30
1817/1817 [==============================] - 1s 324us/step - loss: 0.0014 - accuracy: 0.9296 - val_loss: 0.001
5 - val_accuracy: 0.9165
Epoch 4/30
1817/1817 [==============================] - 1s 322us/step - loss: 0.0014 - accuracy: 0.9296 - val_loss: 0.001
5 - val_accuracy: 0.9165
Epoch 5/30
1817/1817 [==============================] - 1s 302us/step - loss: 0.0014 - accuracy: 0.9296 - val_loss: 0.001
5 - val_accuracy: 0.9165
Epoch 6/30
1817/1817 [==============================] - 1s 304us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
5 - val_accuracy: 0.9165
Epoch 7/30
1817/1817 [==============================] - 1s 300us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
5 - val_accuracy: 0.9165
Epoch 8/30
1817/1817 [==============================] - 1s 300us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
8 - val_accuracy: 0.9165
Epoch 9/30
1817/1817 [==============================] - 1s 324us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
4 - val_accuracy: 0.9165
Epoch 10/30
1817/1817 [==============================] - 1s 310us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
4 - val_accuracy: 0.9165
Epoch 11/30
1817/1817 [==============================] - 1s 295us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
6 - val_accuracy: 0.9165
Epoch 12/30
1817/1817 [==============================] - 1s 292us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
4 - val_accuracy: 0.9165
Epoch 13/30
1817/1817 [==============================] - 1s 296us/step - loss: 0.0012 - accuracy: 0.9296 - val_loss: 0.001
7 - val_accuracy: 0.9165
Epoch 14/30
1817/1817 [==============================] - 1s 299us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
4 - val_accuracy: 0.9165
Epoch 15/30
1817/1817 [==============================] - 1s 328us/step - loss: 0.0012 - accuracy: 0.9296 - val_loss: 0.001
4 - val_accuracy: 0.9165
```

```
Epoch 16/30
1817/1817 [==============================] - 1s 294us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
5 - val_accuracy: 0.9165
Epoch 17/30
1817/1817 [==============================] - 1s 358us/step - loss: 0.0012 - accuracy: 0.9296 - val_loss: 0.001
4 - val_accuracy: 0.9165
Epoch 18/30
1817/1817 [==============================] - 1s 330us/step - loss: 0.0012 - accuracy: 0.9301 - val_loss: 0.001
5 - val_accuracy: 0.9165
Epoch 19/30
1817/1817 [==============================] - 1s 319us/step - loss: 0.0012 - accuracy: 0.9307 - val_loss: 0.001
4 - val_accuracy: 0.9165
Epoch 20/30
1817/1817 [==============================] - 1s 336us/step - loss: 0.0012 - accuracy: 0.9285 - val_loss: 0.001
6 - val_accuracy: 0.9165
Epoch 21/30
1817/1817 [==============================] - 1s 328us/step - loss: 0.0013 - accuracy: 0.9296 - val_loss: 0.001
5 - val_accuracy: 0.9165
Epoch 22/30
1817/1817 [==============================] - 1s 311us/step - loss: 0.0012 - accuracy: 0.9296 - val_loss: 0.001
4 - val_accuracy: 0.9165
758/758 [==============================] - 0s 45us/step
[0.0010163145183430774, 0.9419525265693665]
```

```
dict_keys(['val_loss', 'val_accuracy', 'loss', 'accuracy'])
```

We have plot a comparison graph to see how well the autoencoder and deep learning model worked against the normal machine learning model. As it can be visualize from the graph that for every cluster the new autoencoder technique and deep learning model have outperformed the previous machine learning model. The accuracy has been improved after using autoencoder.

In [17]:
```
# Plotting Comparison graph of accuracy

cluster_id = [0,1,2,3,4,5,6,7,8,9]
accuracy=[10,20,30,40,50,60,70,80,90,100]


plt.figure(figsize=(12,7))
barWidth = 0.25

r1 = np.arange(len(values1))
r2 = [x + barWidth for x in r1]

plt.bar(r1,cluster_accuracy, color='#7f6d5f', width=barWidth, edgecolor='white', label='var1')
plt.bar(r2, new_cluster_accuracy, color='#557f2d', width=barWidth, edgecolor='white', label='var2')

plt.xlabel('Clusters', fontweight='bold')
plt.ylabel('Accuracy', fontweight='bold')
plt.xticks([r + barWidth for r in range(len(cluster_accuracy))], names)
plt.yticks([r + barWidth for r in range(len(cluster_accuracy))], accuracy)



plt.show()
```



In [ ]:

References

[1] Li-Ping Jing, Hou-Kuan Huang, and Hong-Bo Shi. Improved feature selection approach TFIDF in text mining. In
[2] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analy
[3] sklearn.metrics.silhouette_score – scikit-learn 0.22 documentation. (2019). Retrieved 12 December 2019, fro
[4] 2.3. Clustering – scikit-learn 0.22 documentation. (2019). Retrieved 12 December 2019, from https://scikit-l
[5] k-means++: The advantages of careful seeding" Arthur, David, and Sergei Vassilvitskii, Proceedings of the ei
[6] 1.17. Neural network models (supervised) – scikit-learn 0.22 documentation. (2019). Retrieved 12 December 20
[7] Davies, David L.; Bouldin, Donald W. (1979). "A Cluster Separation Measure". IEEE Transactions on Pattern An
[8] Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Ana
[9] Maggipinto, M., Masiero, C., Beghi, A., & Susto, G. A. (2018). A Convolutional Autoencoder Approach for Feat