Transfer Learning MNIST

- Train a simple convnet on the MNIST dataset the first 5 digits [0-4].
- Freeze convolutional layers and fine-tune dense layers for the classification of digits [5-9].

MNIST Dataset

The MNIST database contains 60,000 training images and 10,000 testing images taken from America
American high school students. The MNIST dataset is one of the most common datasets used for im
many different sources. In fact, even Tensorflow and Keras allow us to import and download the MNI

Let's import keras and load MNIST dataset

```
# Initialize the random number generator
import random
random.seed(0)

import warnings
warnings.filterwarnings("ignore")


from keras.backend import backend
from keras.datasets import mnist

# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Using TensorFlow backend.
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tens
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
```

X_train and X_test contain greyscale RGB codes (from 0 to 255) while y_train and y_test contains labe
number they actually are.

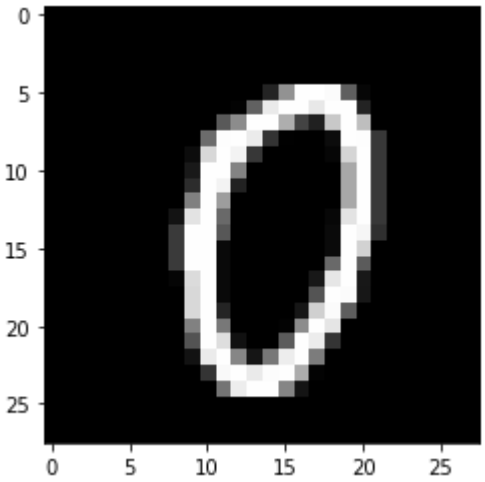
Let's visualize some numbers using matplotlib

```
X_train.shape
```

```
(60000, 28, 28)
```

```
import matplotlib.pyplot as plt
%matplotlib inline
print("Label: {}".format(y_train[1000]))
plt.imshow(X_train[1000], cmap='gray')
```

```
Label: 0
<matplotlib.image.AxesImage at 0x7fcab4ff69e8>
```



Question 1

Create two datasets

- First having digits from 0 to 4
- Second having digits from 5 to 9

Hint: use labels to separate data

```
X_train_0_4 = []
y_train_0_4 = []
X_train_5_9 = []
y_train_5_9 = []
for i in range(0,y_train.shape[0]):
  if y_train[i] in [0,1,2,3,4]:
    X_train_0_4.append(X_train[i])
    y_train_0_4.append(y_train[i])
  else:
      X_train_5_9.append(X_train[i])
      y_train_5_9.append(y_train[i])


X_test_0_4 = []
y_test_0_4 = []
X_test_5_9 = []
y_test_5_9 = []
for i in range(0,y_test.shape[0]):
  if y_test[i] in [0,1,2,3,4]:
```

```
      X_test_0_4.append(X_test[i])
      y_test_0_4.append(y_test[i])
    else:
        X_test_5_9.append(X_test[i])
        y_test_5_9.append(y_test[i])



import numpy as np
X_train_0_4 = np.array(X_train_0_4)
y_train_0_4 = np.array(y_train_0_4)
X_train_5_9 = np.array(X_train_5_9)
y_train_5_9 = np.array(y_train_5_9)

X_test_0_4 = np.array(X_test_0_4)
y_test_0_4 = np.array(y_test_0_4)
X_test_5_9 = np.array(X_test_5_9)
y_test_5_9 = np.array(y_test_5_9)
```

Question 2

Print shape of the data

- print shape of all variables of both the datasets you created

```
print("Shape of X_train_0_4 is - ",X_train_0_4.shape)
print("Shape of y_train_0_4 is - ",y_train_0_4.shape)
print("Shape of X_test_0_4 is - ",X_test_0_4.shape)
print("Shape of y_test_0_4 is - ",y_test_0_4.shape)
print("===="*10)
print("Shape of X_train_5_9 is - ",X_train_5_9.shape)
print("Shape of y_train_5_9 is - ",y_train_5_9.shape)
print("Shape of X_test_5_9 is - ",X_test_5_9.shape)
print("Shape of X_test_5_9 is - ",y_test_5_9.shape)
```

```
Shape of X_train_0_4 is -  (30596, 28, 28)
Shape of y_train_0_4 is -  (30596,)
Shape of X_test_0_4 is -  (5139, 28, 28)
Shape of y_test_0_4 is -  (5139,)
========================================
Shape of X_train_5_9 is -  (29404, 28, 28)
Shape of y_train_5_9 is -  (29404,)
Shape of X_test_5_9 is -  (4861, 28, 28)
Shape of X_test_5_9 is -  (4861,)
```

Question 3

Reshape data

- reshape first dataset
- To be able to use the dataset in Keras, we need 4-dims numpy arrays.
- reshape features to pass it to a Conv2D layer
- channel = 1
- reshape features of first dataset only
- do not reshape labels

```
trainX_0_4 = X_train_0_4.reshape(X_train_0_4.shape[0], 28, 28,1 ).astype('float32')
testX_0_4 = X_test_0_4.reshape(X_test_0_4.shape[0], 28, 28,1 ).astype('float32')
```

Question 4

Normalize data

- normalize first dataset
- we must normalize our data as it is always required in neural network models
- we can achieve this by dividing the RGB codes to 255 (which is the maximum RGB code minus t
- normalize X_train and X_test
- make sure that the values are float so that we can get decimal points after division

```
trainX_0_4 /= 255
testX_0_4 /= 255
```

Print shape of data and number of images

- for first dataset
- print shape of X_train
- print number of images in X_train
- print number of images in X_test

```
trainX_0_4.shape
```

```
(30596, 28, 28, 1)
```

```
trainX_0_4.shape[0]
```

```
30596
```

```
testX_0_4.shape[0]
```

```
testX_0_4.shape[0]
```

```
5139
```

Question 5

One-hot encode the class vector

- encode labels of first dataset
- convert class vectors (integers) to binary class matrix
- convert y_train and y_test
- number of classes: 5
- we are doing this to use categorical_crossentropy as loss

Hint: you can use keras.utils.to_categorical

```
from keras.utils import np_utils
trainY_0_4  = np_utils.to_categorical(y_train_0_4, 5)
testY_0_4  = np_utils.to_categorical(y_test_0_4 , 5)
```

```
from __future__ import absolute_import, division, print_function
import numpy as np
import keras
from keras.datasets import cifar10, mnist
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Reshape
from keras.layers import Convolution2D, MaxPooling2D
from keras.utils import np_utils
import pickle
from matplotlib import pyplot as plt
import seaborn as sns
plt.rcParams['figure.figsize'] = (15, 8)
```

Question 6

We will build our model by using high level Keras.

Initialize a sequential model

- define a sequential model
- add 2 convolutional layers

    - no of filters: 32
    - kernel size: 3x3
    - activation: "relu"
    - input shape: (28, 28, 1) for first layer
- add a max pooling layer of size 2x2
- add a dropout layer
    - dropout layers fight with the overfitting by disregarding some of the neurons while training
    - use dropout rate 0.2

Question 7

Add classification layers

- do this after doing question 6
- flatten the data
    - add Flatten later
    - flatten layers flatten 2D arrays to 1D array before building the fully connected layers
- add 2 dense layers
    - number of neurons in first layer: 128
    - number of neurons in last layer: number of classes
    - activation function in first layer: relu
    - activation function in last layer: softmax
    - we may experiment with any number of neurons for the first Dense layer; however, the fina the number of output classes
- you can add a dropout layer in between, if necessary

```
# Define model
model1 = Sequential()

# 1st Conv Layer
model1.add(Convolution2D(32, 3, 3, input_shape=(28, 28, 1)))
model1.add(Activation('relu'))

# 2nd Conv Layer
model1.add(Convolution2D(32, 3, 3))
model1.add(Activation('relu'))

# Max Pooling
model1.add(MaxPooling2D(pool_size=(2,2)))

# Dropout
```

```
# Dropout
model1.add(Dropout(0.2))

# Fully Connected Layer
model1.add(Flatten())
model1.add(Dense(128))
model1.add(Activation('relu'))

# Prediction Layer
model1.add(Dense(5))
model1.add(Activation('softmax'))
```

Question 8

Compile and fit the model

- compile your model
  - loss: "categorical_crossentropy"
  - metrics: "accuracy"
  - optimizer: "sgd"
- fit your model
  - give train data - features and labels
  - batch size: 128
  - epochs: 10
  - give validation data - features and labels

```
# Loss and Optimizer
model1.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

# Store Training Results
early_stopping = keras.callbacks.EarlyStopping(monitor='val_acc', patience=10, verbose=1, mod
callback_list = [early_stopping]

# Train the model2
model1.fit(trainX_0_4, trainY_0_4, batch_size=128, nb_epoch=10,validation_split=0.1, callback
```

```
Train on 27536 samples, validate on 3060 samples
Epoch 1/10
27536/27536 [==============================] - 2s 62us/step - loss: 0.0696 - acc: 0.9780
Epoch 2/10
27536/27536 [==============================] - 1s 51us/step - loss: 0.0666 - acc: 0.9784
Epoch 3/10
27536/27536 [==============================] - 1s 50us/step - loss: 0.0629 - acc: 0.9797
Epoch 4/10
27536/27536 [==============================] - 1s 51us/step - loss: 0.0591 - acc: 0.9816
Epoch 5/10
27536/27536 [==============================] - 1s 51us/step - loss: 0.0564 - acc: 0.9831
Epoch 6/10
27536/27536 [==============================] - 1s 52us/step - loss: 0.0520 - acc: 0.9831
Epoch 7/10
27536/27536 [==============================] - 1s 51us/step - loss: 0.0520 - acc: 0.9838
Epoch 8/10
27536/27536 [==============================] - 1s 51us/step - loss: 0.0486 - acc: 0.9850
Epoch 9/10
27536/27536 [==============================] - 1s 50us/step - loss: 0.0469 - acc: 0.9854
Epoch 10/10
27536/27536 [==============================] - 1s 52us/step - loss: 0.0442 - acc: 0.9865
<keras.callbacks.History at 0x7fcaa59c93c8>
```

Question 9

Evaluate model

- evaluate your model and get accuracy
- use test features and labels

```
scores = model1.evaluate(testX_0_4,testY_0_4, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 99.12%
```

Question 10

Transfer learning

Now we will apply this model on second dataset (5-9 digits)

- fix the first convolution layers so that the weights in the convolution layers dont get updated in t
- get the second dataset
- train the last 2 dense layers

- predict the accuracy and loss

Make only dense layers trainable

- set trainalble = False for all layers other than Dense layers

```
# Define model
model1 = Sequential()

# 1st Conv Layer
model1.add(Convolution2D(32, 3, 3, input_shape=(28, 28, 1)))
model1.trainable = False
model1.add(Activation('relu'))
model1.trainable = False
# 2nd Conv Layer
model1.add(Convolution2D(32, 3, 3))
model1.trainable = False
model1.add(Activation('relu'))
model1.trainable = False
# Max Pooling
model1.add(MaxPooling2D(pool_size=(2,2)))
model1.trainable = False
# Dropout
model1.add(Dropout(0.2))
model1.trainable = False
# Fully Connected Layer
model1.add(Flatten())
model1.add(Dense(128))
model1.add(Activation('relu'))

# Prediction Layer
model1.add(Dense(5))
model1.add(Activation('softmax'))
```

Modify data

- in your second data, class labels will start from 5 to 9 but for keras.utils.to_categorical the label
- so you need to subtract 5 from train and test labels

```
print("Shape of X_train_5_9 is - ",X_train_5_9.shape)
print("Shape of y_train_5_9 is - ",y_train_5_9.shape)
print("Shape of X_test_5_9 is - ",X_test_5_9.shape)
print("Shape of X_test_5_9 is - ",y_test_5_9.shape)
```

```
Shape of X_train_5_9 is -  (29404, 28, 28)
Shape of y_train_5_9 is -  (29404,)
Shape of X_test_5_9 is -  (4861, 28, 28)
Shape of X_test_5_9 is -  (4861,)
```

```
y_train_5_9_new = []
for i in range(0,y_train_5_9.shape[0]):
      y_train_5_9_new.append((y_train_5_9[i]-5))

y_train_5_9_new = np.array(y_train_5_9_new)


y_test_5_9_new = []
for i in range(0,y_test_5_9.shape[0]):
      y_test_5_9_new.append((y_test_5_9[i]-5))

y_test_5_9_new = np.array(y_test_5_9_new)


y_test_5_9[0]
```

```
7
```

```
y_test_5_9_new[0]
```

```
2
```

Reshape data

- reshape second dataset
- To be able to use the dataset in Keras, we need 4-dims numpy arrays.
- reshape features to pass it to a Conv2D layer
- channel = 1
- reshape features of first dataset only
- do not reshape labels

```
trainX_5_9 = X_train_5_9.reshape(X_train_5_9.shape[0], 28, 28,1 ).astype('float32')
testX_5_9 = X_test_5_9.reshape(X_test_5_9.shape[0], 28, 28,1 ).astype('float32')
```

Normalize data

- normalize second data
- we must normalize our data as it is always required in neural network models
- we can achieve this by dividing the RGB codes to 255 (which is the maximum RGB code minus t
- normalize X_train and X_test

- make sure that the values are float so that we can get decimal points after division

```
trainX_5_9 /= 255
testX_5_9 /= 255
```

Print shape of data and number of images

- print shape of X_train
- print number of images in X_train
- print number of images in X_test

```
trainX_5_9.shape
```

```
(29404, 28, 28, 1)
```

```
trainX_5_9.shape[0]
```

```
29404
```

```
testX_5_9.shape[0]
```

```
4861
```

One-hot encode the class vector

- convert class vectors (integers) to binary class matrix
- convert y_train and y_test
- number of classes: 5
- we are doing this to use categorical_crossentropy as loss

Hint: you can use keras.utils.to_categorical

```
from keras.utils import np_utils
trainY_5_9  = np_utils.to_categorical(y_train_5_9_new, 5)
testY_5_9  = np_utils.to_categorical(y_test_5_9_new , 5)
```

Fit the model

- give train data - features and labels
- batch size: 128
- epochs: 10
- give validation data - features and labels

```
# Loss and Optimizer
model1.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

```
model1.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

# Store Training Results
early_stopping = keras.callbacks.EarlyStopping(monitor='val_acc', patience=10, verbose=1, mod
callback_list = [early_stopping]

# Train the model2
model1.fit(trainX_5_9, trainY_5_9, batch_size=128, nb_epoch=10,validation_split=0.1, callback
```

```
Train on 26463 samples, validate on 2941 samples
Epoch 1/10
26463/26463 [==============================] - 2s 69us/step - loss: 0.8636 - acc: 0.7670
Epoch 2/10
26463/26463 [==============================] - 1s 51us/step - loss: 0.2511 - acc: 0.9180
Epoch 3/10
26463/26463 [==============================] - 1s 51us/step - loss: 0.1949 - acc: 0.9373
Epoch 4/10
26463/26463 [==============================] - 1s 51us/step - loss: 0.1707 - acc: 0.9464
Epoch 5/10
26463/26463 [==============================] - 1s 52us/step - loss: 0.1510 - acc: 0.9527
Epoch 6/10
26463/26463 [==============================] - 1s 50us/step - loss: 0.1369 - acc: 0.9558
Epoch 7/10
26463/26463 [==============================] - 1s 51us/step - loss: 0.1235 - acc: 0.9599
Epoch 8/10
26463/26463 [==============================] - 1s 53us/step - loss: 0.1156 - acc: 0.9630
Epoch 9/10
26463/26463 [==============================] - 1s 51us/step - loss: 0.1075 - acc: 0.9652
Epoch 10/10
26463/26463 [==============================] - 1s 52us/step - loss: 0.0975 - acc: 0.9684
<keras.callbacks.History at 0x7fca3407fb38>
```

Evaluate model

- evaluate your model and get accuracy
- use test features and labels

```
scores = model1.evaluate(testX_5_9,testY_5_9, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 97.22%
```

Sentiment analysis

The objective of the second problem is to perform Sentiment analysis from the tweets collected from devices. Based on the tweet posted by a user (text), we will classify if the sentiment of the user target or not.

Question 1

Read the data

- read tweets.csv
- use latin encoding if it gives encoding error while loading

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473189

Enter your authorization code:
··········
Mounted at /content/drive
```

```
import pandas as pd
filepath = "/content/drive/My Drive/AIML/NLP AND RECOMMENDED SYSTEMS/NLP PROJECT 1/tweets.csv
```

```
df = pd.read_csv(filepath,encoding = 'ISO-8859-1')
```

```
df.shape
```

```
(9093, 3)
```

```
df.isnull().sum()
```

```
tweet_text                                            1
emotion_in_tweet_is_directed_at                    5802
is_there_an_emotion_directed_at_a_brand_or_product    0
dtype: int64
```

```
df.head(6)
```

```
df.is_there_an_emotion_directed_at_a_brand_or_product.unique()
```

```
array(['Negative emotion', 'Positive emotion',
       'No emotion toward brand or product', "I can't tell"], dtype=object)
```

2 @swonderlin Can not wait for #iPad 2 also. The iPad

Drop null values

- drop all the rows with null values

5 @teachntech00 New iPad Apps For #SpeechTherapy NaN

```
df.dropna(inplace=True)
```

Print the dataframe

- print initial 5 rows of the data
- use df.head()

```
df.head(5)
```

	tweet_text	emotion_in_tweet_is_directed_at	is_th
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	

Question 2

Preprocess data

- convert all text to lowercase - use .lower()
- select only numbers, alphabets, and #+_ from text - use re.sub()
- strip all the text - use .strip()
    - this is for removing extra spaces

```
import string , re
import nltk,plotly
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;][‘’“”…]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
PUNCT_SYMBOLS_RE = re.compile('[%s]' % re.escape(string.punctuation))
NUM_RE = re.compile('\w*\d\w*') ## \w = [a-zA-Z0-9_]  \W = [^a-zA-Z0-9_]   -- Remove words co
STOPWORDS = set(stopwords.words('english'))
```

```
def clean_text(text):
    """
        text: a string

        return: modified initial string
    """
    text = text.lower() # lowercase text
    text = text.strip() # for removing extra spaces


    text = REPLACE_BY_SPACE_RE.sub(' ', text) # replace REPLACE_BY_SPACE_RE symbols by space
    text = BAD_SYMBOLS_RE.sub('', text) # remove symbols which are in BAD_SYMBOLS_RE from tex
    text = text.replace('x', '')
    text = PUNCT_SYMBOLS_RE.sub('', text)
    text = NUM_RE.sub('', text)


    text = ' '.join(word for word in text.split() if word not in STOPWORDS) # remove stopwors
    #text = re.sub(r'\W+', '', text)
    return text
```

```
df.head(6)
```

	tweet_text	emotion_in_tweet_is_directed_at	is_th
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	
7	#SXSW is just starting, #CTIA is around the co...	Android	

```
df['tweet_text'] = df['tweet_text'].apply(clean_text)
df['emotion_in_tweet_is_directed_at'] = df['emotion_in_tweet_is_directed_at'].apply(clean_tex
df['is_there_an_emotion_directed_at_a_brand_or_product'] = df['is_there_an_emotion_directed_a
```

```
df.head(6)
```

	tweet_text	emotion_in_tweet_is_directed_at	is_ther
0	iphone hrs tweeting riseaustin dead need upgra...	iphone	
1	jessedee know fludapp awesome ipadiphone app y...	ipad iphone app	
2	swonderlin wait ipad also sale ssw	ipad	
3	ssw hope years festival isnt crashy years ipho...	ipad iphone app	
4	ststate great stuff fri ssw marissa mayer goog...	google	
7	ssw starting ctia around corner googleio hop s...	android	

print dataframe

```
df.shape
```

```
(3291, 3)
```

Question 3

Preprocess data

- in column "is_there_an_emotion_directed_at_a_brand_or_product"
  - select only those rows where value equal to "positive emotion" or "negative emotion"
- find the value counts of "positive emotion" and "negative emotion"

```
df_new = df[((df['is_there_an_emotion_directed_at_a_brand_or_product'] == 'negative emotion')
```

```
print(df_new.is_there_an_emotion_directed_at_a_brand_or_product.unique())
print(df_new.shape)
```

```
['negative emotion' 'positive emotion']
(3191, 3)
```

Question 4

Encode labels

- in column "is_there_an_emotion_directed_at_a_brand_or_product"

- change "positive emotion" to 1
- change "negative emotion" to 0

- use map function to replace values

```
df_new.head(5)
```

	tweet_text	emotion_in_tweet_is_directed_at	is_ther
0	iphone hrs tweeting riseaustin dead need upgra...	iphone	
1	jessedee know fludapp awesome ipadiphone app y...	ipad iphone app	
2	swonderlin wait ipad also sale ssw	ipad	
3	ssw hope years festival isnt crashy years ipho...	ipad iphone app	
4	ststate great stuff fri ssw marissa mayer goog...	google	

```
df_new['is_there_an_emotion_directed_at_a_brand_or_product'] = df_new['is_there_an_emotion_di
```

```
df_new.head(5)
```

	tweet_text	emotion_in_tweet_is_directed_at	is_ther
0	iphone hrs tweeting riseaustin dead need upgra...	iphone	
1	jessedee know fludapp awesome ipadiphone app y...	ipad iphone app	
2	swonderlin wait ipad also sale ssw	ipad	
3	ssw hope years festival isnt crashy years ipho...	ipad iphone app	
4	ststate great stuff fri ssw marissa mayer goog...	google	

Question 5

Get feature and label

- get column "tweet_text" as feature
- get column "is_there_an_emotion_directed_at_a_brand_or_product" as label

```
X = df_new['tweet_text']
y = df_new['is_there_an_emotion_directed_at_a_brand_or_product']
```

Create train and test data

- use train_test_split to get train and test set

- set a random_state
- test_size: 0.25

```
from sklearn.model_selection import train_test_split


X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state =2)
```

Question 6

Vectorize data

- create document-term matrix
- use CountVectorizer()
  - ngram_range: (1, 2)
  - stop_words: 'english'
  - min_df: 2
- do fit_transform on X_train
- do transform on X_test

```
from sklearn.feature_extraction.text import CountVectorizer

# Initialize the "CountVectorizer" object, which is scikit-learn's
# bag of words tool.
vectorizer = CountVectorizer(analyzer = "word", min_df=2,  \
                             tokenizer = None,    \
                             preprocessor = None, \
                             stop_words = 'english',   \
                             max_features = 5000)

# fit_transform() does two functions: First, it fits the model
# and learns the vocabulary; second, it transforms our training data
# into feature vectors. The input to fit_transform should be a list of
# strings.
train_data_features = vectorizer.fit_transform(X_train)
test_data_features = vectorizer.transform(X_test)



# Numpy arrays are easy to work with, so convert the result to an
# array  ## ,columns=train_data_features.get_feature_names()
train_data_features_df = pd.DataFrame(train_data_features.toarray(),columns=vectorizer.get_fe
train_data_features_df
```

	aapl	able	absolutely	abt	access	aclu	aclugoogle	acquired	action	actsofsha
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
2388	0	0	0	0	0	0	0	0	0	
2389	0	0	0	0	0	0	0	0	0	
2390	0	0	0	0	0	0	0	0	0	
2391	0	0	0	0	0	0	0	0	0	
2392	0	0	0	0	0	0	0	0	0	

2393 rows × 2151 columns

Question 7

Select classifier logistic regression

- use logistic regression for predicting sentiment of the given tweet
- initialize classifier

```
from sklearn import  linear_model

lgr = linear_model.LogisticRegressionCV()
```

Fit the classifer

- fit logistic regression classifier

```
lgr.fit(train_data_features, y_train)
```

```
LogisticRegressionCV(Cs=10, class_weight=None, cv=None, dual=False,
                     fit_intercept=True, intercept_scaling=1.0, l1_ratios=None,
                     max_iter=100, multi_class='auto', n_jobs=None,
                     penalty='l2', random_state=None, refit=True, scoring=None,
                     solver='lbfgs', tol=0.0001, verbose=0)
```

Question 8

Select classifier naive bayes

- use naive bayes for predicting sentiment of the given tweet
- initialize classifier
- use MultinomialNB

```
from sklearn import  naive_bayes

nb = naive_bayes.BernoulliNB()
```

Fit the classifer

- fit naive bayes classifier

```
nb.fit(train_data_features, y_train)
```

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Question 9

Make predictions on logistic regression

- use your trained logistic regression model to make predictions on X_test

```
predict_lgr = lgr.predict(test_data_features)
```

Make predictions on naive bayes

- use your trained naive bayes model to make predictions on X_test
- use a different variable name to store predictions so that they are kept separately

```
predict_nb= nb.predict(test_data_features)
```

Question 10

Calculate accuracy of logistic regression

- check accuracy of logistic regression classifer
- use sklearn.metrics.accuracy_score

```
from sklearn.metrics import accuracy_score

accuracy_score(predict_lgr,y_test)
```

```
0.868421052631579
```

Calculate accuracy of naive bayes

- check accuracy of naive bayes classifer
- use sklearn.metrics.accuracy_score

```
accuracy_score(predict_nb,y_test)
```

```
0.8734335839598998
```