Machine Learning Track, Task 5,

Classification Model Implementation.

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import sklearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn import neighbors
from sklearn import preprocessing
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
```

File handling

```
#open files
Train1 = open("/positive.review", "r")
Train2 = open("/negative.review", "r")
Test = open("/unlabeled.review", "r")

#Read the files and split them by lines
Train_pos = Train1.read().splitlines()
Train_neg = Train2.read().splitlines()
Test_reviews = Test.read().splitlines()

#Close the file pointers
Train1.close()
Train2.close()
Test.close()
```

Combining the training sets into one

```
#Combine the training sets to deal with one training set (easeir)
train_set= [(x, 1) for x in Train_pos] + [(x, -1) for x in Train_neg]
print("The total number of training data is: ", len(train_set))
```

Extracting the features from the training set

```
#extract the features from the training data
features_dic = {}
id_ = 0
for review, label in train_set:
  features = review.strip().split()[:-1]
  for item in features:
    feat, val = item.strip().split(":")
    if feat not in features_dic:
      features_dic[feat]= id_
      id_+=1
```

Filling the matrix using feature ids

```
#Represent the reviews using the obtaine features
x = np.zeros((len(train_set), len(features_dic)))
y = [y for (x,y) in train_set]
#We need to fill the matrix X properly now using feature ids
for i, (review, label) in enumerate(train_set):
  features = review.strip().split()[:-1]
  for item in features:
    feat, val = item.strip().split(":")
    x[i][features_dic[feat]] = val
```

Sadly being unable to handle the kNN classifier, we had to use the example

```
X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X, y)

print(neigh.predict([[1.1]]))

print(neigh.predict_proba([[0.9]]))
```

- There are no pre-defined statistical methods to find the most favorable value of K.
- Choosing a small value of K leads to unstable decision boundaries.
- The substantial K value is better for classification as it leads to smoothening the decision boundaries.
- Derive a plot between error rate and K denoting values in a defined range. Then choose the K value as having a minimum error rate.

Storing the word embedding into a dictionary

```
embeddings_dict = {}
with open("/glove.6B.50d.txt", "r", encoding="utf8") as f:
  for line in f:
    values = line.split()
    word = values[0]
    vector = np.asarray(values[1:], "float32")
    embeddings_dict[word] = vector
```

Logistic Regression classifier but being unable to use our dataset and we used an example with iris dataset

```
X, y = load_iris(return_X_y = True)
LRG = linear_model.LogisticRegression(
   random_state = 0, solver = 'liblinear', multi_class='auto')
LRG.fit(X, y)
print(LRG.score(X, y))
```