```
import re
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

def read_data(file):
    data = []
    with open(file, 'r')as f:
        for line in f:
            line = line.strip()
            label = ' '.join(line[1:line.find("]")].strip().split())
            text = line[line.find("]")+1:].strip()
            data.append([label, text])
    return data

file = 'text2.txt'
data = read_data(file)
print("Number of instances: {}".format(len(data)))


def ngram(token, n):
    output = []
    for i in range(n-1, len(token)):
        ngram = ' '.join(token[i-n+1:i+1])
        output.append(ngram)
    return output

def create_feature(text, nrange=(1, 1)):
    text_features = []
    text = text.lower()
    text_alphanum = re.sub('[^a-z0-9#]', ' ', text)
    for n in range(nrange[0], nrange[1]+1):
        text_features += ngram(text_alphanum.split(), n)
    text_punc = re.sub('[a-z0-9]', ' ', text)
    text_features += ngram(text_punc.split(), 1)
    return Counter(text_features)


def convert_label(item, name):
    items = list(map(float, item.split()))
    items
    label = ""
    for idx in range(len(items)):
        if items[idx] == 1:
            label += name[idx] + " "

    return label.strip()

emotions = ["joy", 'fear', "anger", "sadness", "disgust", "shame", "guilt"]
```

```
X_all = []
y_all = []
for label, text in data:
    y_all.append(convert_label(label, emotions))
    X_all.append(create_feature(text, nrange=(1, 4)))


X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, test_size = 0.2,

def train_test(clf, X_train, X_test, y_train, y_test):
    clf.fit(X_train, y_train)
    train_acc = accuracy_score(y_train, clf.predict(X_train))
    test_acc = accuracy_score(y_test, clf.predict(X_test))
    return train_acc, test_acc

from sklearn.feature_extraction import DictVectorizer
vectorizer = DictVectorizer(sparse = True)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)


svc = SVC()
lsvc = LinearSVC(random_state=123)
rforest = RandomForestClassifier(random_state=123)
dtree = DecisionTreeClassifier()

clifs = [svc, lsvc, rforest, dtree]

# train and test them
print("| {:25} | {} | {} |".format("Classifier", "Training Accuracy", "Test Accurac
print("| {} | {} | {} |".format("-"*25, "-"*17, "-"*13))
for clf in clifs:
    clf_name = clf.__class__.__name__
    train_acc, test_acc = train_test(clf, X_train, X_test, y_train, y_test)
    print("| {:25} | {:17.7f} | {:13.7f} |".format(clf_name, train_acc, test_acc))


l = ["joy", 'fear', "anger", "sadness", "disgust", "shame", "guilt", "love"]
l.sort()
label_freq = {}
for label, _ in data:
    label_freq[label] = label_freq.get(label, 0) + 1

# print the labels and their counts in sorted order
for l in sorted(label_freq, key=label_freq.get, reverse=True):
    print("{:10}({})  {}".format(convert_label(l, emotions), l, label_freq[l]))


emoji_dict = {"joy":"😂 ", "fear":"😱 ", "anger":"😠 ", "sadness":"😢 ", "disgust":"😒
t1 = "This looks so joy"
t2 = "I have a fear of dogs and cats"
t3 = "I have pass the exams successfully"
t4 = "I saw a cripple in rags with small children in Italy.  He was probably an imp

texts = [t1, t2, t3, t4]
for text in texts:
```

```
features = create_feature(text, nrange=(1, 4))
features = vectorizer.transform(features)
prediction = clf.predict(features)[0]
print( text,emoji_dict[prediction])
```