Functions to familiarize:

1.Linear SVC The objective of a Linear SVC (Support Vector Classifier) is **to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes**, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.

2.CountVectorizer:

Creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample.

Given a data set of support tickets. Each ticket also has an associated "urgency score" of between 0 and 3, and where 0 is "very urgent" and 3 is "not urgent". It would be useful if we could have a machine guess how urgent a ticket is, based on the description, so the urgent tickets can be resolved first

1. For the given data set, perform text classification using SVM and find out the accuracy of the model.

Ref: https://www.codementor.io/blog/text-classification-6mmol0q8oj

CODE

```
from sklearn.feature_extraction.text import  CountVectorizer

from sklearn.metrics import classification_report

from sklearn.svm import LinearSVC

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix


with open("tickets.txt") as f:

    tickets = f.read().strip().split("\n")
```

```
with open("labels_4.txt") as f:
    labels = f.read().strip().split("\n")


X_train, X_test, y_train, y_test = train_test_split(tickets, labels, test_size=0.1,
random_state=1337)


vectorizer = CountVectorizer()
svm = LinearSVC()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
_ = svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print(classification_report(y_test, y_pred))
```

output

```
 precision    recall  f1-score   support

           0       0.75      0.79      0.77       159
           1       0.53      0.52      0.53       147
           2       0.56      0.54      0.55       154
           3       0.96      0.95      0.95       140

    accuracy                           0.70       600
   macro avg       0.70      0.70      0.70       600
weighted avg       0.70      0.70      0.70       600
```

```
print(confusion_matrix(y_test, y_pred))
```

output

```
[[126  18  14   1]
 [ 20  77  48   2]
 [ 19  49  83   3]
 [  3   1   3 133]]
```