The IMDB dataset actually comes packaged with keras and its allready tokenized, menaing the text is allready
of unique word indices. The IMDB dataset contains 50,000 movie reviews (25,000 for training and 25,000 for
contains of 50% positive and 50% negative reviews (12,500 x 2).

```
import numpy as np
from keras.datasets import imdb
import matplotlib.pyplot as plt
```

```
vocabulary=7500 # we will only use the 7500 most frequently used words
```

Next code block is to fix a bug in keras. If bug would not exist this block would be just 1 line of code: (train_da
(test_data, test_labels) = imdb.load_data(num_words=vocabulary)

I suggest to just copy the fix and it does not really matter if you understand it or not

```
# save np.load
np_load_old = np.load

# modify the default parameters of np.load
np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)

# call load_data with allow_pickle implicitly set to true
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=vocabu

# restore np.load for future normal usage
np.load = np_load_old
```

In the next line of code we will print the lists that contain sequences of words represented by a word index. If
converted to a sequence of indices we would need to add one pre-processing step using Tokenizer

```
print(train_data[1]) # train_data is a list of word sequences
```

Now we will vectorize the training and test data. Basically we will create a matrix where the rows are the revie
columns represent the vocabulary (7500 columns). We will set a 1 in the correct column if the word of the rev
the vocabulary. This means that matrix will be rather sparse.

```
def vectorize_sequences(sequences, dimension=vocabulary):
    results=np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence]=1
    return results
```

Now we apply the function to our training and test data as well as the labels. For the labels we use a different
the asarray function to convert the list to an array and we assign the items in the array to float32

```
x_train=vectorize_sequences(train_data)
x_test=vectorize_sequences(test_data)

y_train=np.asarray(train_labels).astype('float32')
y_test=np.asarray(test_labels).astype('float32')
```

Now we are ready to apply Random Forests

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=100, random_state=42) #n_estimators provides
model.fit(x_train,y_train)
y_pred=model.predict(x_test)

from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
from sklearn.metrics import confusion_matrix
y_pred=model.predict(x_test)
confusion_matrix(y_test,y_pred)
```

For 100 trees we have 2030 false negatives and 1825 false positives. Recall in our case is TP/(TP+FN) = 83%
TP/(TP+FP)=85%