DEBUGGING:

<https://www.codementor.io/allisonf/how-to-debug-python-code-beginners-print-line-du107ltvx>

<https://medium.freecodecamp.org/how-to-improve-your-debugging-skills-abb5b363bdb8>

DATA ANALYSIS:

1. Before you go and solve any problem please ask questions at data, and try to answer them using visual plots, by printing stats.
2. These plots and stats will help you a lot to how to proceed in solving that problem and creating better features and better feature encoding.

FIT, TRANSFORM, FIT\_TRANSFORM:

When you do

```*vec =CountVectorizer()*```

then it will initiate the `*CountVectorizer*` with default parameters.

*```vec.fit(train\_text)```*

simply means that internally it is only learning the vocabulary of `Text` means it is preparing your data to be ready to transform by learning the internal parameters of your data (in your case, it's vocabulary)

*```bag\_of\_words = vec.transform(train\_text)```*

means it is applying that learned parameters (vocabulary) to the data and thus giving you output. Now, as you should know that the *`vocabulary(number of words)`* can be different for *`train text`* and *`test text`* thus they will give you different dim (number of words). So, what you should do is -

*```vec = CountVectorizer (), vec.fit(train\_text)```*

which learns the vocabulary of `Train Text` and then apply or transform your both *`train text`* and *`test text`* using that learned vocabulary to ensure the same dim. for both by doing -

*```bag\_of\_words\_train = vec.transform(train\_text)```*

and

*```bag\_of\_words\_test = vec.transform(test\_text)```*

Ex:

*model = CountVectorizer()*

*model.fit(train\_text)*

*train\_bow = model.transform(train\_text)*

*cv\_bow = model.transform(cv\_text)*

*test\_bow = model.transform(test\_text)*

BATCH WISE PREDICTION:

There are some algorithms ex: KNN that takes lot of processing time and consume memory which leads problems like 'memory error' and 'system getting stuck', in that cases the best way to get rid of these problems is to predict the cross validation/test data points batch wise i.e.

when you are predicting the test/cv data labels, instead of predicting all at a time, you can predict like.

*predicted\_labels = []*

*for i in range(1, n, 1000):*

*predicted\_labels.extend(model.predict(testdata[i to i+1000]))*

OVER SAMPLING AND UNDER SAMPLING:

Do go through these blogs

<http://www.alfredo.motta.name/cross-validation-done-wrong/>

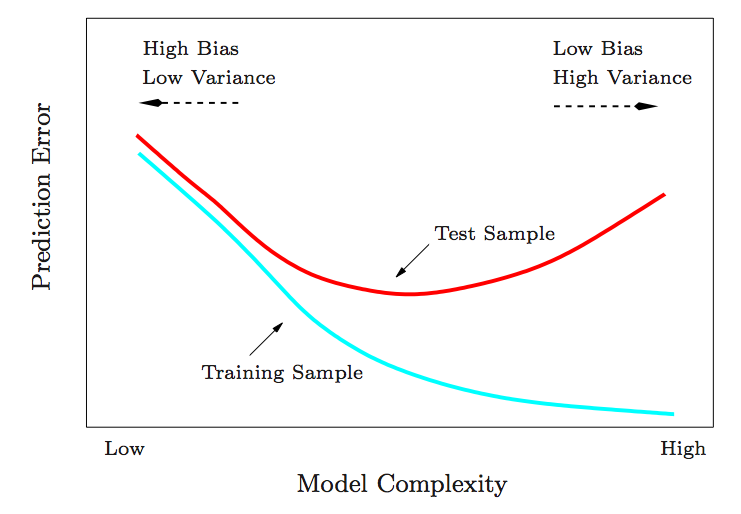
<https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation>

<https://elitedatascience.com/imbalanced-classes>

DATA LEAKAGE:

1. When you are featurizing your train data, you should not look at the test/cv data information
2. When you are featurizing/encoding any data, you should never look at the output variable.

OVER FITTING and UNDERFITTING:

1. We determine over fitting or under fitting, by looking at the error plots
2. 
3. To find out the underfitting and overfitting you must plot both train metric and test metric in the same plot as we have given in the instruction notebooks.