* I know that I didn’t split dataset into training and testing datasets and I should, and that this is a very basic thing to do. I was just focused on other things and I reminded myself about it at the end and I didn’t have time to implement this.
* our dataset is imbalanced: in the y variable we have a lot of zeros and few ones and because of that model sometimes predicts all the time zero. In order to deal with that we can:
* make a new loss function / model which calculates bigger loss for false negatives then for false positives (penalization)
* Another solution is to remove some rows with zeros for training or duplicate rows with ones.
* Use a model for anomaly detection (when campaign is successfull this is an anomaly)
* Divide the more abundant class into L distinct clusters. Then train L predictors, where each predictor is trained on only one of the distinct clusters, but on all of the data from the rare class. To be clear, the data from the rare class is used in the training of all L predictors. Finally, use model averaging for the L learned predictors as your final predictor
* Let N be number of samples in the rare class. Cluster the abundant  
  class into N clusters (agglomerative clustering may be best here), and use the resulting cluster mediods/means as the training data for the abundant class. To be clear, you *throw out* the original training data from the abundant class, and use the mediods instead. Voila, now your classes are balanced! But your dataset is much smaller, so that might be an issue
* we have a mix of categorical and continuous variables:
* we can change continuous variables into categorical by assigning a range to every value, for example to the value 3 we assign range ‘0-5’ and to the value 6 we assign range ‘5-10’
* We can use one hot encoding for categorical variables and concatenate them with continuous variables
* For categorical variables use 1 neural network and concatenate output of that network with continuous variables:

<https://datascience.stackexchange.com/questions/29634/how-to-combine-categorical-and-continuous-input-features-for-neural-network-trai>

* I think that for columns with binary values like 'default', 'housing', 'loan', 'contactedBefore' I don't need to use one hot encoding but for now I will use it because that is simpler
* we could also build a model which predicts if someone will subscribe to the term deposit withoud being part of a campaign. If we know that someone will subscribe anyway then we don’t need to call him.
* we can use tokenizing instead of one hot encoding. With this approach there might a problem such that in different column we can have the same word. For example in columns 'default', 'housing' and 'loan' there might be word 'yes' or 'no'. We would need to differentiate them some how. For example instead of writing ‘yes’ in the column default we can write ‘default\_yes’ or something like this.