1) QUERY :  
we used fit,transform and fit\_transform in out previous datsets MNIST and Amazon. We consider “fit” method as training method to find f in our case whatever SCALER we are using it finds the parameters for it and “transform” method is actually for giving transforming data.  
EXAMPLE count\_vect = CountVectorizer() #in scikit-learn  
count\_vect.fit(preprocessed\_reviews)  
final\_counts = count\_vect.transform(preprocessed\_reviews)

SO IS IT RIGHT TO SAY BOTH THESE FUNCTION ARE STILL PART OF TRAINING PROCESS ONLY AS THERE IS NO TEST DATA YET ?

Answer: Yes fit() method try to learn the data and patterns in the data and after that we transform() the data into the vector forms (here it is sparse matrix)

2) my question is TF-IDF is a technique to convert text to vector or to find the importance of a word in corpus? i am confused?

Ans: Tf-IDF is used to convert text to vector and also find the importance of a word in the corpus using the IDF value  
(Replace every word with tf-idf value means converting text to vector)

3) Can you comment on ” Classification algorithm learns a function “.

Ans: Machine learning algorithms are described as learning a target function (f) that best maps input variables (X) to an output variable (Y): Y = f(X)  
This is a general learning task where we would like to make predictions in the future (Y) given new examples of input variables (X). We don’t know what the function (f) looks like or its form (it is different for different algorithm we use). If we did, we would use it directly and we would not need to learn it from data using machine learning algorithms.

4) if the machine predicts the nature of the new data basing on the class we already know, it is referred to as the Classification.  
Regression on the other hand predicts a real value on the basis of the information the machine receives from the user input. This value is obtained by training with the training dataset.

5) pca useful for dimensionality reduction not visualization.  
tsne is for visualization.

6) Choosing the ‘correct’ notion of distance requires understanding the context i.e., The difference depends on your data. Cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter. This happens for example when working with text data represented by word counts. Another reason is that when modeling texts as vectors you will have many dimensions, thousands, the euclidean distance is not very good for very high dimensional data.

7)KNN requires full dataset DTrain and Dtest.So,more space complexity.

8) Let assume i have 1000 points as a part of D\_n (complete dataset)  
out of these 1000 points i have 800 “-ve points” & 200 “+ve points”.  
is there any chance these 200 +ve points will go to D\_test data set?  
if YES , that how our f\_n will behave ?

Ans:

To avoid this case, we do stratified sampling so that both Dtrain and Dtest have same proportion for positive and negative point. You can take a look at the parameter called stratify in the sklearn train\_test\_split method: [https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_sp](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

9) Once a model is trained and you get new data which can be used for training, you can load the previous model and train onto it. For example, you can save your model as a .pickle file and load it and train further onto it when new data is available. Do note that for the model to predict correctly, the new training data should have a similar distribution as the past data.

Predictions tend to degrade based on the dataset you are using. For example, if you are trying to train using twitter data and you have collected data regarding a product which is widely tweeted that day. But if you use use tweets after some days when that product is not even discussed, it might be biased. The frequency will be dependent on dataset and there is no specific time to state as such. If you observe that your new incoming data is deviating vastly, then it is a good practise to retrain the model.

Optimizing parameters on the aggregated data is not overfitting. Large data doesn't imply overfitting. Use cross validation to check for over-fitting.

10) The gist of the lectures we have gone through so far in this chapter:  
If our dataset pertains to Classification model:  
1. We need to find best K  
For this we’ll split our entire data set into train\_ds, CV\_ds and test\_ds in 6:2:2 ratio using SK learn Stratify technique.  
Then we’ll find best K from CV\_ds. Value of K will be chosen for which the accuracy is max and error is min.  
2. We’ll test if this K would really make our classification algo well fit.  
3. To test K, we’ll go for (train\_error, CV\_error Vs K values) plot. For the best value of K, the distance bw Train\_error and C\_error will be minimum.

11) But a small thing to remember. It is recommended not to use ‘Accuracy’ as the CV metric when you are working with imbalanced datasets. Instead of Accuracy, you can use ‘f1’, ‘auc’,etc as CV metric in case of imbalanced datasets.

12) Train the model using D\_train.  
For different values of the hyper-parameter,  
Make predictions again using the same model on D\_train and D\_cv using some metric like ‘f1\_micro’ or ‘auc’. That will give Train scores and CV scores. Subtract each of these values from 1 (ie., 1-train scores(i) and 1-cv scores(i)). That will give the train and CV errors.  
Which ever value of the hyper-parameter gives least CV error, is considered as an optimal value.

High CV error, Low Training Error –> Overfitting  
High CV error, High Training Error –> Underfitting

13) GridSearchCV, where you could get both train and cv scores directly.

14) Why cross-validation error is high in overfitting?

Ans: when train error is low and cross validation error and test error is high we can say we are doing overfitting because your model becomes too specialized to its training data that when unseen data comes along it will instead perform worse.

15) KNN Regression : <https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/>

16)