**This is what I did, a step by step explanation**

Problem Statement :-

Given a training set with following features: id, qid1, qid2, question 1, question 2, is\_duplicate

Clearly the set of inputs are: id, qid1, qid2, question1, quesstion2

The output is a binary classification (is\_duplicate) . 1 for duplicate question and 0 for non duplicate.

**Step 1: Feature Selection**

It was obvious just from plain observation that the features id, qid1 and qid 2 were irrelevant to the problem in hand. Hence Question 1 and Question 2 are the determining factors.

**Step 2: Initial Try with TF IDF vectorizer**

Thought of diving straight into using NLP feature extraction library.

**Library Used: Sci Kit Learn TF- IDF Vectorizer.**

What it does: Convert a collection of raw documents to a matrix of TF-IDF features.

What is TF-IDF feature? tf–idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a [document](https://en.wikipedia.org/wiki/Document) in a collection or [corpus](https://en.wikipedia.org/wiki/Text_corpus). The tf-idf value increases [proportionally](https://en.wikipedia.org/wiki/Proportionality_(mathematics)) to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

Applying TF IDF to the question pairs extracted top 256 features out of each of them. So, it vectorized a single dimension feature into a 256 dimension feature.

i.e. we took question 1 and question 2 and merged them. If training set has n rows then the new meged set has 2n rows. Now TF IDF transform resulted in a 2n\*256 sparse matrix.

Since we are looking at pairs of data, we took the difference of all question one and question two pairs with this. This resulted in a matrix that again has the same number of rows n and 256 columns that describes the relationship between the two questions.

Now it was time to visualize the data. Since we cant visualize 256 dimensions hence we did some **dimension reduction using T-SNE** ( t-SNE is the very popular algorithm to extremely reduce the dimensionality of data in order to visually present it. It is capable of mapping hundreds of dimensions to just 2 or 3 while preserving important data relationships, that is, when closer samples in the original space are closer in the reduced space.) **which uses pca** (principal component analysis) method.

We plotted the reduced dimension using scatter plots. It did not tell us much about the structure of the space that we created. There seem to be no clusters of either class present. TF IDF did not help much in this case.

**Step 3: It was time to do some Exploratory data Analysis.**

EDA basically means constructing your own features so that it may give an in sight of hidden features that relates more closely to the output.

The Features constructed were: q1characterCount, q2characterCount, q1WordCount, q2WordCount, Word Share. Word share was calculate as: (no. of words shared by the question pair)/(total no of words in q1+total no of words in q2).

Now we did the **t-SNE dimension reduction** with the manually engineered features i.e. number of words in both questions, character lengths and their word share coefficient.

t-SNE is sensitive to scaling of different dimensions and we want all of the dimensions to contribute equally to the distance measure that t-SNE is trying to preserve. **Hence, before reducing the dimension using t-SNE, we applied minmaxscalar to scale the features.**

What does MinMaxScalar Do? It Transforms features by scaling each feature to a given range.This estimator scales and translates each feature individually such that it is in the given range on the training set, i.e. between zero and one. **It is a type of Preprocessing**.

Now that we had scalarized our new engineered features it was time to reduce the dimensions using t-SNE.

The t-SNE embedding of the engineered features has much more structure than the previous one where we were only computing differences of TF-IDF encodings. In the cluster of the negatives we have few positives whereas in the cluster of positives we have a lot more negatives. That matches our observation from the boxplot of word share coefficient above, where we could see that the negative class has a lot of overlap with the positive class for high word share coefficients.

*Question character length correlations by duplication label Question character length correlations by duplication label*: The pair plot of character length of both questions by duplication label showed us that, duplicated questions seemed to have a somewhat similar amount of characters in them. Also we could see something quite intuitive, **that there is rather strong correlation in the number of words and the number of characters in a question**.

**Step 4: Train a model with the basic feature Constructed.**

For that we will use Logisitic regression, for which we will did a best parameter search with Cross Validation.

We split the train set into train and test by a factor of 0.33.

We applied GridSearch Cross validation on the train set with logistic regression as the model. We found out the best estimator from a given set of estimator.

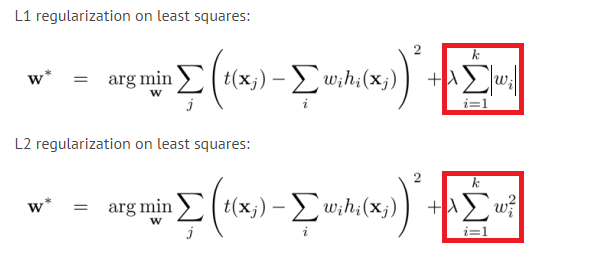
What is Grid Search? Grid search means you have a set of models (which differ from each other in their parameter values, which lie on a grid). What you do is you then train each of the models and evaluate it using cross-validation. You then select the one that performed best.

Here our set of model is a set of logistic regression models which differ from each other in regards to the parameter C. Let’s find out what is C?

**Regularization:**

Two types of penalization used: L1 and L2

What are they? It’s a regularization technique where the Weights are penalized inorder to avoid overfitting. L1 and L2 are two ways to penalize. Best described below.



C will determine how much the red portion in above equation is.

C= inverse of regularization strength. i.e. 1/regularization parameter (lambda). Lesser c implies more regularized.

**Cross-Validation:**

So we took three values of C (0.000001, 0.001, 1) and applied each f them to both L1 and L2. Now Grid Search CV will apply 3-fold cross validation to each hypothesis and find out the optimal hypothesis.

Finally got our optimal hypothesis

{'C': 1.0, 'penalty': 'l1'}

**Step 5: Performance Evaluation**

**ROC Curve:** Receiver operator characteristic, used very commonly to assess the quality of models for binary classification. We looked at three different classifiers, a strongly regularized one and two with weaker regularization. The heavily regularized model has parameters very close to zero.

**What is ROC?** The **receiver operating characteristic** (ROC) curve is a two-dimensional graph in which the **false positive rate** is plotted on the X axis and the **true positive rate** is plotted on the Y axis. The ROC curves are useful to visualize and compare the performance of classifier methods.

True positive Rate: Positives Correctly classified / total positives

False positive rate: negatives incorrectly classified / total negatives

An ROC curve demonstrates several things:

1. It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
2. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
3. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

So, as evidential from our ROC curve c=1.0 is the best fit.

**Precision Recall Curve: precision**is the fraction of retrieved instances that are relevant, while **recall** is the fraction of relevant instances that are retrieved. In the context of Information Retrieval, the precision-recall curve becomes very useful.

Precision = true positives/(true positives+false positives)

Recall = true positives/(true positives+false negetives)

For example: If I have a database with 100 documents, out of which 60 are relevant to a particular keyword. If my IR system returns a total of 50 documents, out of which 40 are relevant, the precision for this system is {40}/{50} = 0.8 and the recall is {40}/{60} = 0.66   
If instead there's another IR system that returns only 10 documents, chances are that atleast  9 of them are relevant. This would increase my precision to 0.9 but decrease it's recall to just 0.15.

It’s important to keep a balance of precision and recall. One lower and the other higher is bad. Whereas both close to each other is good.

So we performed precision recall on our model.

**Step 5: Applying the model to test dataset**

Finally applied the model to the test data provided by kaggle. Plotted the probability distribution function as it is a logistic regression.