1. Result Analysis Outcome of Hashing Trick

In ML models, the performance of the models was very poor using TF-IDF but using hashing it has been improved as sparsity has reduced. Because we often deal with huge vocabularies, it is hard to have enough data to really see samples of all the things that people can say. Data sparsity is a bigger concern in NLP than it is in other machine learning domains. There will be a lot of real-life sentences that we never encounter in the training data. It can be solved in various data mining techniques [18], but we have solved it by using hashing. The main advantages of this method that we found is that:

- It is suitable for large amounts of data because it stores the features corresponding hash values as a result training is faster [19].
- There may be the same hash value of the two features and one of the duplicated values parallel processing and then takes the other vacant position in a column which reduces the sparsity and it's a quick approach to make a vector space model [20] and also reduces the dimension.

In DL models, If the original column has a significant number of unique values, one hot encoding is prone to cause a major issue (too many predictors). Another drawback of one-hot encoding is that it causes multicollinearity among the variables, which reduces the model's accuracy and also it concerns the memory computability that increases the models complexity. In binary text classification problem hashing trick or feature hashing overcomes these concerns [21]. It also reduces the dimensions and model complexity. The feature vectors are not sparse and the feature values are stored in a vocabulary in a certain range. My thesis is also a binary classification problem and so the models performed well using hashing.

2. Result Analysis Outcome of Hashing-Autoencoder

I used this method in the ML models for regenerating the feature vectors. After hashing, the feature vectors still remain sparse, but better than the TF-IDF or Bag of Words. After sending the feature vectors to the autoencoders, The vectors regenerate the space which totally removes the sparsity. Thus the performance has improved. In DL models, the feature values are stored in a vocabulary in a certain range and the feature vectors are not sparse. Autoencoder on the DL models slow the training and also down the performance.