AWS FALSE KEY DETECTION

Approach - 1

Approach 1 is done through two matrices namely matrix_lower.npy and matrix_upper.npy as a numpy arrays to seperate the text into two categories namely uppercase letters and lowercase letters respectively. But in this case there is an error encountered namely mismatch error. Since the data has a possibility of having both uppercase and lowecase in some instances, it is not suited for this problem. Not only that since the threshold value is too high (the constant value) and there is no possibility to change the value, it is also not convinient for the model.

Approach - 2

Approcah 2 is fully based on Machine learning model. Since this problem want to classify whether it is a true AWS key or not, it is being treated as a binary classifier. So as an attempt, we used Logistic Regression since in the best case gives much accuracy for the model.But the errors we encountered are listed below:

- Since the no of data trained is not equally correct since there is repetition of data in the trained data. Hence one of the important error we encountered while we are discussing is Overfitting. Overfitting is nothing but when we give the actual data which is trained as a input to the model, only for that input it will give you the correct output, Not for all other data which is being treated as negative from your side.

- Secondly the features of the data. Since the data is the mixture of uppercase and lowercase letters the training of data is not so easy. And the valid AWS key is combination of letters, symbols and numbers, the data is not good to train. So to enhcane that we have used Root mean square loss(RMSE), as a loss function. Though it is being used to regret these errors what we have encountered, but it is not the case. It is still showing Overfiitng condition.

- Since in this model pipeline is used to convert the text data into numerical value, it also fails to give you the same output.

Approach - 3

Approach 3 is based on Multi Layer Perceptron. Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function by training on a dataset,

s a supervised lear

$$f(\cdot) : R^m \rightarrow R^o$$

Where m is the number of dimensions for input and n is the number of dimensions for output. Given a set of features and a target , it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. The failures what we encountered are listed below:

- Since the last model what we trained is Logistic Regression Using Sigmoid function the Logit function of the Linear discriminant function, we encountered the redundant data problem and we end up with Overfitting. But in this Model the error is complement to the error what we got in the Logistic Model. In the logistic Model we got the error for Valid Data, but for this model we got it for Invalid Data.
- In this model The data what we got to train the model is too small(concern to this model). Hence it cannot extract the features of the data.
- Since using CountVectorizer or tfidfVectorizer to convert the text data into numerical value, for some valid data for instance A@34DEFRRyeukr#ubeub//24de11, for this data we can't able to find the numerical value since for each and every testing the value is being changed.

Approach - 4

In Approach 4 we planned to take this problem to the unsupervised learning model where we planned to cluster all the valid data as one cluster and invalid data as an another cluster. So for this approach, K - Means Clustering with ensemble methods to enhance the accuracy of the model(Ada Boost Algorithm for ensembling with bootstrap). The errors what we encountered are listed below:

- Here in this model There is no redundant errors since we refined the data. But the new problem is finding the absolute nearest cluster. For a particular data, it is being close to the both the centroids of the cluster. So the model fails to understand what the data is and what cluster it belongs to.

- Using Ada Boost we almost trained all the data. But without using using it it almost fails to give you the correct cluster. But the error while we using Ada Boost Algorithm with Bootstrap is redundant of data multiplication. This leads to the Overfitting case.

Approach 5 (Final Approach)

Finally we ended up with the Natural Language Processing. With the help of NLTK(Natural Language toolkit) package which is available in python, We have done the following:

- Firstly we split the data based in camel case splitting(Feature extraction purposes)
- Nextly checking the split data with the words what we generated with the help of nltk package.
- If the split data is present inside the words list we are going to label them as Invalid
- else the model will detect the valid data
- To search in more enhanced version, we removed alphabets, stop words from the words list

Using this model maximum of 70 to 80 % data can be detected correctly without any issues encountered in the above approaches.