Problem Understanding:

As Observed the Data set, I came to know this is a text Classification Data set. The goal of text classification is to automatically classify the text documents into one or more defined categories.

Train & test accuracy score

```
# Naive Bayes on Count Vectors

accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_count, train_y, xvalid_count)

print ("NB, Count Vectors: ", accuracy)
```
NB, Count Vectors: 0.6619097956307258

```
# Naive Bayes on Word Level TF IDF Vectors

accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf, train_y, xvalid_tfidf)

print ("NB, WordLevel TF-IDF: ", accuracy)
```
NB, WordLevel TF-IDF: 0.5722339675828048

```
# Naive Bayes on Ngram Level TF IDF Vectors

accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf_ngram, train_y, xvalid_tfidf_ngram)

print ("NB, N-Gram Vectors: ", accuracy)
```
NB, N-Gram Vectors: 0.5156800563777308

```
# Naive Bayes on Character Level TF IDF Vectors

accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf_ngram_chars, train_y, xvalid_tfidf_ngram_chars)

print ("NB, CharLevel Vectors: ", accuracy)
```
NB, CharLevel Vectors: 0.514446793516561

```
# Linear Classifier on Count Vectors
```

```
accuracy = train_model(linear_model.LogisticRegression(), xtrain_count, train_y, xvalid_count)

print ("LR, Count Vectors: ", accuracy)
LR, Count Vectors:  0.6629668780831571


# Linear Classifier on Word Level TF IDF Vectors

accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf, train_y, xvalid_tfidf)

print ("LR, WordLevel TF-IDF: ", accuracy)
LR, WordLevel TF-IDF:  0.6416490486257929


# Linear Classifier on Ngram Level TF IDF Vectors

accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf_ngram, train_y, xvalid_tfidf_ngram)

print ("LR, N-Gram Vectors: ", accuracy)
LR, N-Gram Vectors:  0.5251937984496124


# Linear Classifier on Character Level TF IDF Vectors

accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf_ngram_chars, train_y, xvalid_tfidf_ngram_chars)

print ("LR, CharLevel Vectors: ", accuracy)
LR, CharLevel Vectors:  0.6272022551092319



# SVM on Ngram Level TF IDF Vectors

accuracy = train_model(svm.SVC(), xtrain_tfidf_ngram, train_y, xvalid_tfidf_ngram)

print ("SVM, N-Gram Vectors: ", accuracy)
SVM, N-Gram Vectors:  0.5162085976039464

# RF on Count Vectors

accuracy = train_model(ensemble.RandomForestClassifier(), xtrain_count, train_y, xvalid_count)

print ("Xgb, Count Vectors: ", accuracy)
Xgb, Count Vectors:  0.629492600422833
```

Extereme Gradient Boosting on Word Level TF IDF Vectors

```
accuracy = train_model(xgboost.XGBClassifier(), xtrain_tfidf.tocsc(), train_y, xvalid_tfidf.tocsc())

print ("Xgb, WordLevel TF-IDF: ", accuracy)
```

Xgb, WordLevel TF-IDF: 0.607646229739253

```
def create_model_architecture(input_size):
    # create input layer
    input_layer = layers.Input((input_size, ), sparse=True)

    # create hidden layer
    hidden_layer = layers.Dense(100, activation="relu")(input_layer)

    # create output layer
    output_layer = layers.Dense(1, activation="sigmoid")(hidden_layer)

    classifier = models.Model(inputs = input_layer, outputs = output_layer)
    classifier.compile(optimizer=optimizers.Adam(), loss='binary_crossentropy',metrics=["accuracy"])
    return classifier

classifier = create_model_architecture(xtrain_tfidf_ngram.shape[1])

accuracy = train_model(classifier, xtrain_tfidf_ngram, train_y, xvalid_tfidf_ngram, is_neural_net=True)
```

C:\Users\Dell PC\anaconda3\lib\site-packages\tensorflow\python\framework\indexed_slices.py:449: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/model_1/dense_2/embedding_lookup_sparse/Reshape_1:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/model_1/dense_2/embedding_lookup_sparse/Reshape:0", shape=(None, 100), dtype=float32), dense_shape=Tensor("gradient_tape/model_1/dense_2/embedding_lookup_sparse/Cast:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)

```
533/533 [==============================] - 7s 11ms/step - loss: -16.3551 - accuracy: 0.0632

print ("NN, Ngram Level TF IDF Vectors",  accuracy)
```

NN, Ngram Level TF IDF Vectors 0.09936575052854123

#While the above framework can be applied to a number of text classification problems, but to achieve a good accuracy some improvements can be done in the overall framework. For example, following are some tips to improve the performance of text classification models and this framework.

1. Text Cleaning

2. Hstacking Text / NLP features with text feature vectors
3. Hyperparamter Tuning in modelling
4. Ensemble Models