Binary Classification of Disaster Tweets

GUNA SEKHAR SAI HARSHA LAGUDU
ASU ID: 1223118858

1 Introduction:

In times of disaster, Twitter has become a crucial communication route. Because cellphones are so widely available, users may report an emergency in real time. As a result, more agencies are interested in monitoring Twitter programmatically (i.e. disaster relief organizations and news agencies). However, it is not always evident if a person's statements are announcing a tragedy. In this project, I have used a dataset of 10,000 hand-classified tweets from kaggle [1]. In this project, I am attempting to create a best machine learning model that predicts which tweets are about actual disasters and which aren't.

2 Dataset:

The data for this project is sourced from 'kaggle' named "Natural Language Processing with Disaster Tweets". It has 10,000 hand-classified tweets and it provides 4 parameters and the target variable i.e, 1 (for real disaster) or 0 (for fake) as shown in Table 1. The given dataset has 2 files in which 1 file contains a training file and another file contains a test file. The training file contains 7613 tweets while the test file contains 3263 tweets.

Table 1: Description of Dataset

Parameter	Description
ID	a unique identifier for each tweet
Text	the text of the tweet
Location	this denotes from where the tweet was sent from (may be blank)
Keyword	a particular keyword from the tweet (may be blank)
Target	this denotes whether a tweet is about a real disaster (1) or not (0)

In the above mentioned parameters, ID has no impact on whether the tweet is about a real disaster or not. In "Location" and "Keyword" parameters have an impact on the output label but in the given dataset more than half of the overall tweets have these parameters left empty. So in

this project, parameters like ID, Location and Keyword are not taken into consideration during the classification of the tweet.

Data Preprocessing: After dropping the ID, Location and Keyword, the parameter “Text” is preprocessed. Initially, the tweets are cleaned by removing “@” mentions, “#” tag symbols, extra white spaces, new lines, hyperlinks, emojis e.t.c. Then all the text is converted into lower case.

3 Experiment:

In this project, I utilized 6 classifiers with different hyperparameters as shown in Table 2 & 7 setups to train/test data. Then picked the best classifier based on their well-known good performance for huge and tough datasets like the one we're using. The classifiers employed here are SVM, Naive Bayes, Logistic Regression, K Nearest Neighbours, AdaBoost and Random Forest. I have used GridSearchCV to find the best parameters for a given classifier for a given setup. GridSearchCV tries all the combinations of the hyper parameter values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method.

Table 2: Details of Machine Learning models and it’s hyper parameters

Model	Hyper Parameter	Parameter Description	Parameter values used
SVM (Support Vector Machine)	C	Regularization parameter	[0.1, 1, 10]
	gamma	Kernel coefficient	[1, 0.1, 0.01]
	kernel	Specifies the kernel type to be used in the algorithm.	['rbf','linear','poly', 'sigmoid']
MultinomialNB	alpha	Additive smoothing parameter	np.linspace(0.5, 1.5, 6),
	fit_prior	Whether to learn class prior probabilities or not.	[True, False]
Logistic Regression	C	Inverse of regularization strength	[0.1, 1, 20, 40, 60, 80, 100]
	solver	Algorithm to use in the optimization problem.	['lbfgs', 'liblinear']

KNN	n_neighbors	Number of neighbors to use	[5, 10, 20, 50, 80, 100, 200]
	weights	Weight function used in prediction.	['uniform', 'distance']
Random Forest	n_estimators	The number of trees in the forest	[50,100,150]
	max_depth	The maximum depth of the tree.	[2,3,None]
	criterion	The function to measure the quality of a split.	['gini','entropy']
AdaBoost	n_estimators	The maximum number of estimators at which boosting is terminated.	[50,100,150]
	learning_rate	Weight applied to each classifier at each boosting iteration.	[0.5,1,1.5]

After training, the best model with the best setup is used to predict the test file.
Note: The test file contains only 4 variables: ID, text, location and keyword. This file is used as a submission file for the kaggle competition.

3.1 Setup 1: Removing Punctuation:

In this setup, all the models are trained and tested after removing punctuations from the corpus. The training file is divided into 80% for the training set and 20% for the test set. The results are shown in Table 3.

Table 3: Accuracy score and best hyper parameters for each model using Setup 1

S.NO	Model	Best_score	Best_parameters
1	SVC	0.804797	'C': 1, 'gamma': 1, 'kernel': 'linear'
2	MultinomialNB	0.798801	'alpha': 0.5, 'fit_prior': True
3	logistics_regression	0.790806	'C': 20, 'solver': 'liblinear'

4	K_Nearest_Neighbors	0.784810	'n_neighbors': 50, 'weights':'distance'
5	random_forest	0.780147	'criterion': 'gini', 'max_depth':None, 'n_estimators': 100
6	AdaBoost	0.742172	'learning_rate':0.5, 'n_estimators': 150

3.2 Setup 2: Removing Stop-words:

All the models are trained and tested after removing stop-words from the corpus. The training file is divided into 80% for the training set and 20% for the test set. The results are shown in Table 4.

Table 4: Accuracy score and best hyper parameters for each model using Setup 2

S.NO	Model	Best_score	Best_parameters
1	SVC	0.802132	'C': 1, 'gamma': 1, 'kernel': 'linear'
2	MultinomialNB	0.802798	'alpha': 0.5, 'fit_prior': True
3	logistics_regression	0.784810	'C': 40, 'solver': 'lbfgs'
4	K_Nearest_Neighbors	0.782145	'n_neighbors': 100, 'weights':'distance'
5	random_forest	0.762159	'criterion': 'gini', 'max_depth':None, 'n_estimators': 50
6	AdaBoost	0.756163	'learning_rate':0.5, 'n_estimators': 150

3.3 Setup 3: Removing Numbers:

All the models are trained and tested after removing numbers from the corpus. The training file is divided into 80% for the training set and 20% for the test set. The results are shown in Table 5

Table 5: Accuracy score and best hyper parameters for each model using Setup 3

S.NO	Model	Best_score	Best_parameters
1	SVC	0.801466	'C': 1, 'gamma': 1, 'kernel': 'linear'
2	MultinomialNB	0.795470	'alpha': 0.5, 'fit_prior': True
3	logistics_regression	0.787475	'C': 40, 'solver': 'lbfgs'
4	K_Nearest_Neighbors	0.783478	'n_neighbors': 80, 'weights':'distance'
5	random_forest	0.775483	'criterion': 'gini', 'max_depth':None, 'n_estimators': 50
6	AdaBoost	0.742172	'learning_rate':0.5, 'n_estimators': 150

3.4 Setup 4: Removing Repeating Characters:

All the models are trained and tested after removing repeating characters. The training file is divided into 80% for the training set and 20% for the test set. The results are shown in Table 6.

Table 6: Accuracy score and best hyper parameters for each model using Setup 4

S.NO	Model	Best_score	Best_parameters
1	SVC	0.801466	'C': 1, 'gamma': 1, 'kernel': 'linear'
2	MultinomialNB	0.802132	'alpha': 0.5, 'fit_prior': True
3	logistics_regression	0.791472	'C': 80, 'solver': 'lbfgs'
4	K_Nearest_Neighbors	0.784144	'n_neighbors': 80, 'weights':'distance'
5	random_forest	0.768821	'criterion':'entropy, 'max_depth':None, 'n_estimators': 150
6	AdaBoost	0.744837	'learning_rate':0.5,

			'n_estimators':150

3.5 Setup 5: Stemming and Lemmatization:

All the models are trained and tested after applying stemming and lemmatization. The training file is divided into 80% for the training set and 20% for the test set. The results are shown in Table 7.

Table 7: Accuracy score and best hyper parameters for each model using Setup 5

S.NO	Model	Best_score	Best_parameters
1	SVC	0.788141	'C': 10, 'gamma': 0.1, 'kernel': 'sigmoid'
2	MultinomialNB	0.799467	'alpha': 0.5, 'fit_prior': True
3	logistics_regression	0.782811	'C': 20, 'solver': 'lbfgs'
4	K_Nearest_Neighbors	0.790140	'n_neighbors': 80, 'weights':'distance'
5	random_forest	0.774817	'criterion': 'gini', 'max_depth':None, 'n_estimators': 150
6	AdaBoost	0.754830	'learning_rate':0.5, 'n_estimators': 150

3.6 Setup 6: Applying Setup 1–5:

All the models are trained and tested after removing punctuation, stop-words, numbers, repeating words, stemming and lemmatization. The training file is divided into 80% for the training set and 20% for the test set. The results are shown in Table 8.

Table 8: Accuracy score and best hyper parameters for each model using Setup 6

S.NO	Model	Best_score	Best_parameters
1	SVC	0.794137	'C': 1, 'gamma': 1, 'kernel': 'linear'
2	MultinomialNB	0.796802	'alpha': 0.5, 'fit_prior': True
3	logistics_regression	0.785476	'C': 20,

			'solver': 'liblinear'
4	K_Nearest_Neighbors	0.783478	'n_neighbors': 50, 'weights':'distance'
5	random_forest	0.778814	'criterion':'entropy, 'max_depth':None, 'n_estimators': 150
6	AdaBoost	0.756163	'learning_rate':0.5, 'n_estimators': 150

3.7 Setup 7: Keeping all above features:

All the models are trained and tested without eliminating any of the above special features. The training file is divided into 80% for the training set and 20% for the test set. The results are shown in Table 9.

Table 9: Accuracy score and best hyper parameters for each model using Setup 7

S.NO	Model	Best_score	Best_parameters
1	SVC	0.804131	'C': 1, 'gamma': 1, 'kernel': 'linear'
2	MultinomialNB	0.800799	'alpha': 0.5, 'fit_prior': True
3	logistics_regression	0.788141	'C': 40, 'solver': 'lbfgs'
4	K_Nearest_Neighbors	0.786809	'n_neighbors': 80, 'weights':'distance'
5	random_forest	0.762825	'criterion':'entropy, 'max_depth':None, 'n_estimators': 100
6	AdaBoost	0.743504	'learning_rate':0.5, 'n_estimators': 150

3.8 Summary:

From Table 3-9, it is observed that there is no drastic change in best accuracy score for each model if the setup is changed. Hence, we can consider setup 6 as the best configuration because the number of feature words are less and there is not much change in best accuracy scores for each model compared to other setup's. It is also observed that from the setup 1-7, SVM classifier has the best accuracy score compared to other models.

Hence, I have taken the SVM classifier and setup-6 as the best configuration to predict the test file. For this combination the best hyper parameters are C=1, gamma = 1, kernel = linear and the accuracy score is 79.41%. The confusion matrix is shown in Figure 1.

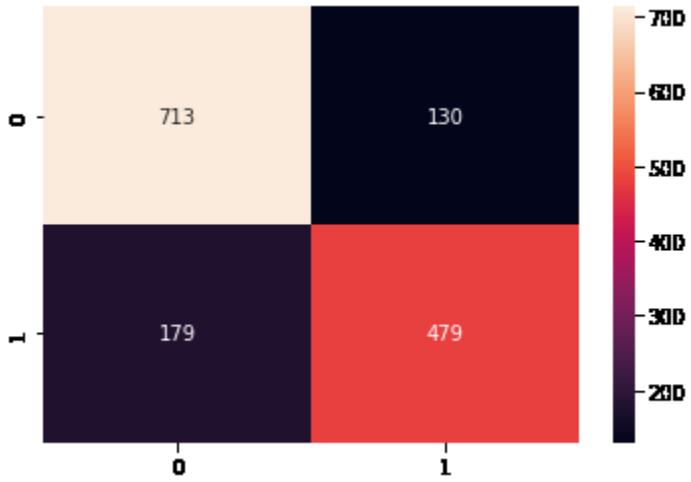

Figure 1: Confusion Matrix for SVC with C=1, gamma=1, kernel='linear' using Setup-6

4 Conclusion:

The purpose of this study was to identify a machine learning model with best hyper parameters which can predict tweets about actual disasters and which aren't in the "Natural Language Processing with Disaster Tweets" dataset. Employing several well-known machine learning methods, namely SVM, Naive Bayes, Logistic Regression, K Nearest Neighbours, AdaBoost and Random Forest, we have been able to classify tweets about actual disasters and which aren't. The **GridSearchCV** method has become very helpful to identify best hyperparameters for a given model. Also, it has been observed that model performance after removing punctuation, stop-words, numbers, repeating words, stemming and lemmatization is similar to the model with all these features. Hence, we can say that punctuations, stop-words, numbers, repeating words, stemming and lemmatization has no much importance on the model performance.

5 References:

[1] https://www.kaggle.com/competitions/nlp-getting-started/data