4. Experimental setup:

4.1 Random Forest

Random Forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

For the case of the project, firstly, the preprocessed data are read and '.head()' and '.describe()' functions are applied to show the information about the data. By doing this, we notice that there is a column named 'Unnamed: 0', which is the index in the csv file. Thus, these columns are dropped by using '.drop()' function. Then, the last column is defined as 'labels' that indicates weather the disaster is real or not.

Secondly, two model is fitted with different data. One of them is all train data are used to train the random forest model. Then, we get the prediction result by using the given test data. Because there is no target data in the given test data, the result is saved into a csv file and submitted to the Kaggle Submission to see the score.

Another model is only train data is used. The train data is separated into two parts, training data and testing data. 20 percent train data is selected as testing data by setting 'test_size=0.2'. 'test_size=0.2' and 'random_state=100' are setted in each model in this project so that the results of all models are comparable.

When fitting the random forest model, there are relatively few important frame parameters, such as n_estimators, bootstrap, oob score, criterion and so on. The main concern is n_estimators, which is the maximum number of decision tree in random forest. N_estimator is the maximum number of weak learners, whose default is 10. Generally speaking, the model would be underfit if n_estimators is too small, while the model would be overfit if n_estimators is too large. Thus, it's better to choose a moderate value. For random forest, increasing the 'number of sub-models' (n_estimators) can significantly reduce the variance of the overall model without affecting the deviation and variance of the sub-models. The accuracy of the model will increase with the increase of the 'number of sub-models'(n_estimators). Therefore, we study which n_esimators is the one with highest model accuracy. Two ranges are checked. One is from 10 to 3000 with step 100 (the left image of Figure 1), indicating the accuracy reach highest when n_estimator is equal to 1700. Thus, another range is from 1600 to 1800 with step 10 (the right image of Figure 1), which shows that accuracy is highest when n_estimator is equal to 1660. Thus, finally, we choose n_estimators as 1660. Other parameters, like bootstrap, are set as default based on previous study and setting.

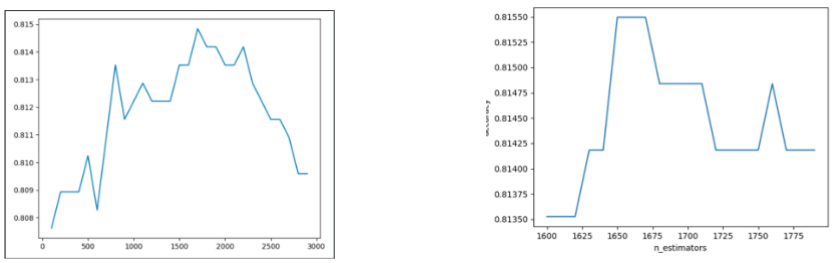

Figure 1. Results of the Accuracy with Different N_estimator

4.2 Naïve Bayes

Naïve Bayes is a classifier based on conditional probability, which is often used as a baseline. It is extremely fast and accurate given its 'naïve assumption'.

Bayes theorem is shown as Equation 1:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \qquad \text{Equation 1}$$

We can find the probability of y happening, given that X has occurred, where X is the evidence and y is the hypothesis. The assumption made here is that the predictors are independent. That is presence of one particular feature does not affect the other. Another assumption is that all the predictors have an equal effect on the outcome.

To the special case here, similar to the treatment of random forest model above, after reading data, data preprocessing and defining predictor and response variables, two models are fitted.

Firstly, like in random forest, we use all train data to train the naïve bayes model. Then taking the given test data as input, we get prediction result. The result is saved as another csv file and put to the Kaggle Submission to get the accuracy score.

Secondly, we use only train data to fit the model. Train data is separated into two parts, training data and testing data. 'test_size' is also set as 0.2 and 'random_state' is also set as 100 in order to make results from different model comparable.

5. Results

5.1 Random Forest

5.1.1 Using all train data to train the model

By submitting the csv file to Kaggle, it returns the score and the screen shoot of it is shown as Figure 2:

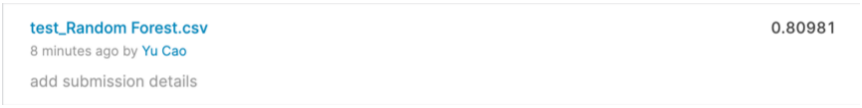

Figure 2. Kaggle Accuracy Score of Random Forest Model

The accuracy of the model is 0.80981, which is very high. It is the 874th in 2602 results in Kaggle Submission. Therefore, by only using train data we can see that random forest model performs very well, it can predict the correct situation, which includes positive observation with positive prediction and negative observation with negative prediction, with 80.98% accuracy.

5.1.2 Separating train data into training and testing to train the model

The accuracy of the model is 0.815, which can also be calculated by confusion matrix shown in Table 1 by using Equation 2:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{Equation 2}$$

Table 1

Confusion Matrix of Random Forest Model

N = 1523	Predicted: Yes	Predicted: No
Actual: Yes	794 (TP)	80 (FN)
Actual: No	202 (FP)	447 (TN)

Note: True Positive (TP): Observation is positive, and is predicted to be positive.
False Negative (FN): Observation is positive, but is predicted negative.
True Negative (TN): Observation is negative, and is predicted to be negative.
False Positive (FP): Observation is negative, but is predicted positive.

Table 2

Classification Matrix of Random Forest Model

	precision	recall	f1-score	support
0	0.80	0.91	0.85	874
1	0.85	0.69	0.76	649
accuracy			0.81	1523
macro avg	0.82	0.80	0.80	1523
weighted avg	0.82	0.81	0.81	1523

In Table 2, classification matrix, some useful outputs are given, like recall and precision.

Recall is defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. It can be calculated by applying Equation 3. High recall indicates the class is correctly recognized. Here, recall is 0.81, indicating that 81% class are recognized correctly.

Precision can be calculated by dividing the total number of correctly classified positive examples by the total number of predicted positive examples, as shown in Equation 4. High precision signifying the ratio of an example labelled as positive is indeed positive. Here, precision is 0.82, showing a high accuracy of labelling correctly.

$$\textit{Recall} = \frac{TP}{TP + FN} \quad \text{Equation 3}$$

$$\textit{Precision} = \frac{TP}{TP + FP} \quad \text{Equation 4}$$

Based on the above results, we can draw a preliminary conclusion that random forest model fitted well. However, accuracy assumes equal costs for both kinds of errors. Thus, a high accuracy can't make sure an excellent result. As a result, we take ROC curve and P-R curve for further analysis.

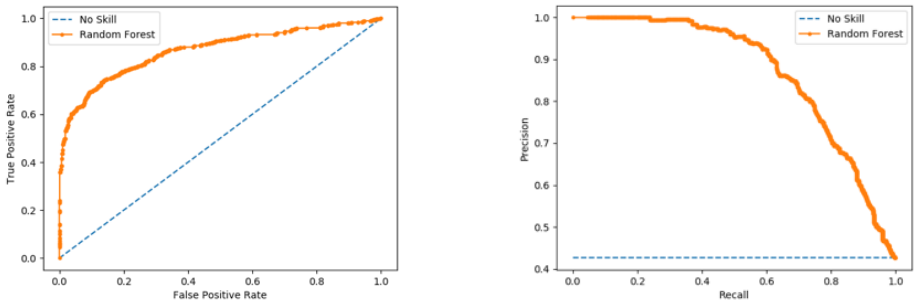

Figure 3. ROC Curve and P-R Curve of Random Forest Model

ROC Curve (the left figure in Figure 3) and P-R Curve (the right figure in Figure 3) are both diagnostic tools that help in the interpretation of probabilistic forecast for binary classification predictive modeling problems.

ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds, while Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.

A no-skill classifier is one that cannot discriminate between the classes and would predict a random class or a constant class in all cases. A model with no skill is represented at the point (0.5, 0.5). A model with no skill at each threshold is represented by a diagonal line from the bottom left of the plot to the top right and has an AUC of 0.5. Shown as the blue line in the left figure of Figure 3.

AUC is the area under the curve. Here, AUC of the ROC curve is 0.866. And values on the y-axis of the plot is large, indicating a high true positives and low false negatives. And values on the x-axis of the plot is small, thus it has a low false positives and high true negatives. This is same as confusion matrix showing. The f1 in P-R curve is 0.760, the AUC is 0.870.

Finally, we visualize a single decision tree in Figure 4, which is very large. To make things easier, we limit the depth of tree in the forest to produce an understandable image by setting 'max_depth' as 3. Then, a smaller tree is shown in Figure 5. Based on this tree, we can make a prediction for any new data point.

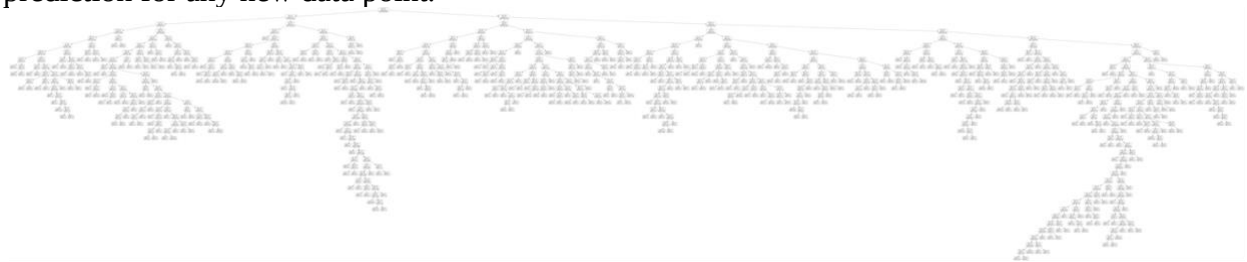

Figure 4. A Single Decision Tree of Random Forest

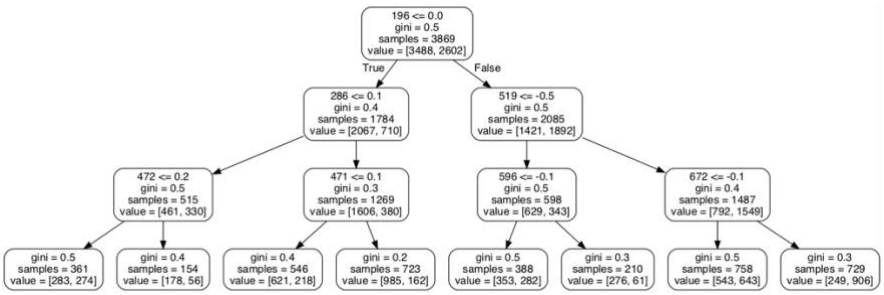

Figure 5. A Smaller Tree

5.2 Naïve Bayes

5.2.1 Using all train data to train the model

By submitting the Naïve Bayes test csv file to Kaggle, it returns the score and the screen shoot of it is shown as Figure 6:

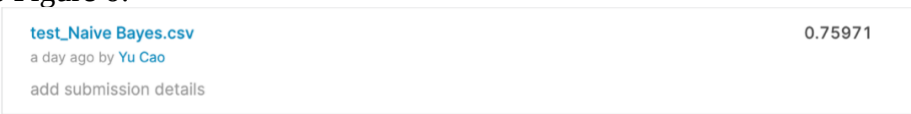

Figure 6. Kaggle Accuracy Score of Naïve Bayes Model

The accuracy of Naïve Bayes model is 0.75971, which is much lower than the score of random forest. By only using train data we can see that naïve bayes model can predict the correct situation with 75.97% accuracy, which is good, but worse than random forest model. As a baseline, that the accuracy of random forest model is higher suggests that random forest model can improve the accuracy of model.

5.1.2 Separating train data into training and testing to train the model

The accuracy of model is 0.759, same as the score it in Kaggle. The confusion matrix and classification matrix of naïve bayes model are shown in Table 3 and Table 4 as following:

Table 3

Confusion Matrix of Naïve Bayes Model

N = 1523	Predicted: Yes	Predicted: No
Actual: Yes	683 (TP)	191 (FN)
Actual: No	176 (FP)	473 (TN)

Note: True Positive (TP): Observation is positive, and is predicted to be positive.
False Negative (FN): Observation is positive, but is predicted negative.
True Negative (TN): Observation is negative, and is predicted to be negative.
False Positive (FP): Observation is negative, but is predicted positive.

Table 4

Classification Matrix of Naïve Bayes Model

	precision	recall	f1-score	support
0	0.80	0.78	0.79	874
1	0.71	0.73	0.72	649

accuracy			0.76	1523
macro avg	0.75	0.76	0.75	1523
weighted avg	0.76	0.76	0.76	1523

In Table 4, classification matrix, recall is 0.76, indicating that 76% class are recognized correctly. And precision is 0.76, showing 76% examples labelled correctly. Both recall and precision here are lower than those of random forest model. Therefore, we can get a possible conclusion that naïve bayes model fitted good, but not as good as random forest model. Then, ROC curve and P-R curve are also shown for further analysis.

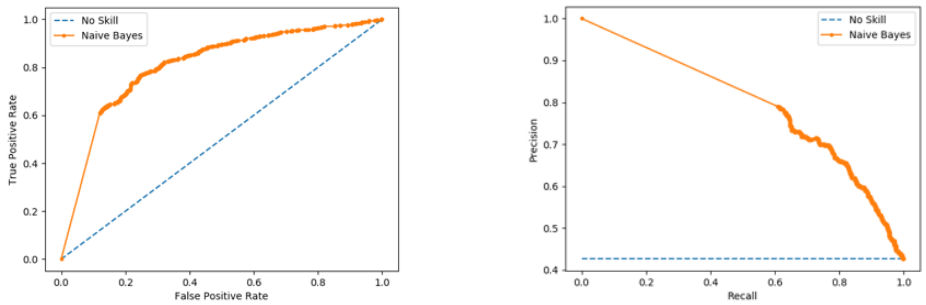

Figure 7. ROC Curve and P-R Curve of Naïve Bayes Model

Here AUC of ROC curve (the left image of Figure 7) is 0.810. The f1 in P-R curve (the right image of Figure 7) is 0.720, the AUC is 0.794. All of these are smaller than those of random forest model. We know that The AUC value is equivalent to the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative example. The larger the AUC value, the more likely the classification algorithm is to rank positive samples in front of negative samples, which means a better classification. Thence, we can conclude that random forest model is a better classification compared with naïve bayes model.

6. Summary and conclusion

In summary, we can conclude that random forest is a better classifier than Naive Bayes. Therefore, it can predict whether a given tweet is describing a real disaster more accurately. Relevant departments can choose this classifier as the standard of judgment, so that when a disaster occurs, they can make more accurate predictions and judgments, and guide the public to respond.

However, in the process of writing this project, there is still room for improvement. For example, for the selection of parameters, more methods can be chosen and applied to see whether the model can be optimized.

7. Reference

1. Brownlee, J. (2019, December 18). How to Use ROC Curves and Precision-Recall Curves for Classification in Python. Retrieved from https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

2. Brownlee, J. (2020, January 14). ROC Curves and Precision-Recall Curves for Imbalanced Classification. Retrieved from https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-imbalanced-classification/
3. Fawcett, Tom. “An Introduction to ROC Analysis.” *Pattern Recognition Letters*, vol. 27, no. 8, 2006, pp. 861–874., doi:10.1016/j.patrec.2005.10.010.
4. Gandhi, Rohith. “Naive Bayes Classifier.” *Medium*, Towards Data Science, 17 May 2018, towardsdatascience.com/naive-bayes-classifier-81d512f50a7c.
5. Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *Plos One*, *10*(3). doi: 10.1371/journal.pone.0118432
6. Yiu, Tony. “Understanding Random Forest.” *Medium*, Towards Data Science, 14 Aug. 2019, towardsdatascience.com/understanding-random-forest-58381e0602d2.
7. ZHOU, Z. H. I.-H. U. A. (2020). *Machine Learning*. S.l.: SPRINGER VERLAG, SINGAPOR.