6103 Project：Real or Not, NLP with Disaster Tweets

Jinyi Shang　Yu Cao

Weifei Wang　Yuan Gao

The George Washington University

Content

Twitter has played an important role in social media life. Nowadays, people like to post Tweets to share things and many people get information from Tweets, like news, weather. For example, some people may post a Tweet when they see disasters to alarm other people. However, not every Tweet is about real disasters, sometimes people use words like “ablaze” metaphorically to describe the beautiful view.

To help people avoid some unnecessary harm from this ‘disaster’ information, we found a dataset from Kaggle which is related to this topic. What’s more, we attempt to use this dataset to build some machine learning models that predict which Tweets are about real disasters and which one’s aren’t.

In this report, we will cover following 6 parts to state this challenge: (1) introduction of the data;(2) data preprocessing; (3) 6 models used in the report; (4) evaluation of these models ;(5) conclusion and (6) reference.

1, Introduction of Data

The data is from Kaggle, and here is the link:

https://www.kaggle.com/c/nlp-getting-started/overview

There are three csv file: train, test and sample submission. For the train set, there are 7613 rows by 5 columns. each column represents id - a unique identifier for each tweet, keyword- a particular keyword from the tweet (may be blank), location - the location the tweet was sent from (may be blank), text- the text of the tweet and target-this denotes whether a tweet is about a real disaster (1) or not (0). For the test set, there are 3264 rows by 4 columns, which contains id, keyword location and text, with no target.

Since the target are only 1 and 0, it’s a binary classification problem. Our goal is using the train set to train some suitable models and predicting the test texts by our models.

2, Data Preprocessing

What we need to clean are some text contents. We checked the texts in our train set and found there are many noises in the text. Some of texts contains abbreviation and slang; Some of texts use special symbols like #, $, @ etc; what’s more, almost each of text has website information and uses punctuation. These are not conducive to our text analysis.

In the light of of the problems above, we provide 9 nine methods of data Preprocessing aiming to obtain the clean data: Mislabel, CONTRACTION_MAP, @/#/http, Repetitive Letter, Abbreviation, Lowercase, punctuation, Stopwords and Keywords Variable.

·Mislabel

At the beginning, we grouped the train data by text and found there is a dozen of sentence have a different target, but the content of text are the same. Therefore, we a new column: target_relabeled and revise the mislabel manually.

·**Contraction map**

Here, we use a contraction dictionary from the Internet to achieve some phrase like: ain't, don't, hadn't. we can use this dictionary to convert these phrase to their original form:is not, do not, had not. You can see the dictionary in the our github.

·**@/#/http**

In the train data, there are lots of website information, @ and # ,this part used for cleaning them.

·**Repetitive Letter**

Here, we write some loops to convert some words like 'loooook' , ‘goooood’ to 'look' ‘good’.

·**Abbreviation**

This part is different from contraction part. In English, there are some expression like 'lmao' means happy, 'icq' means 'I seek you'. To convert these slang to their normal form, we still downloaded a dictionary and a slang text. Then, we combine these two part to clean clean our text.

·**Lowercase**

This part we are going to uniform the case of all of our text content. We convert all of uppercase letter to lowercase.

·**Punctuation**

This part used for deleting all of punctuation of the text. We think that punctuation is sometimes meaningless when we do data analysis.

·**Stopwords**

In our text analysis, there are many words which didn't have actual meaning, like: 'to','at' etc. Here, we use the package **nltk.corpus**, which contains many stopwords to clean the text.

·**Keywords Variable**

Finally, in the training data, there is a column called keywords, it contains the disaster words of the text or the disaster this text described (didn't contain in the text). Since these words are important, we add them to the text.

3, Bert as service: converting text to vector

After prepocessing the text, we will encode the text content to a vector with specific length to run our traditional machine leaning model.

Our methods is bert as service. Now, what is bert as service? Simply speaking, bert as service is the encoder of bert. As we know, bert is a powerful deep learning

model, which can solve nearly all of NLP problem. Therefore, we use bert as service to achieve our text to vector process.

Here is a link of bert as service tutorial bert as service:

https://github.com/hanxiao/bert-as-service

4, Models used in this project

4.1 K-Nearest Neighbors (KNN)

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression[1]. In both cases, the input consists of the k closest training examples in the feature space. In this project, k-NN was used for classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). Thus, the k value is the most important parameter in this model.

To begin with, we read the preprocessed data using pandas. By ".head()" function, we found that the column named "Unnamed: 0" repeat the index of the .csv files which need to be dropped in both train and test data. Then, the last column is defined as 'labels' that indicates weather the disaster is real or not, using 1 and 0 separately. The rest of the columns became the features which is assigned to be x of train data.

After feature engineering, the KNeighborsClassifier from sklearn package was applied for KNN modeling. The prediction was created with the x of test data. We saved the result in a csv file and made it match the format for submission in Kaggle website. Besides, upload the result of KNN with default and get the score of accuracy as our base line.

The next step is to find the best k parameter for KNN. In this part, we used train data in the last step for both training and testing without submitting to Kaggle. The data was split into 5 folds using KFold package from sklearn. Each fold in 5 folds worked as a test data and the rest of folds became the train data every loop. We set the k range from 5 to 20 and all the 5 folds would have 15 iterations for different k value. Then we saved and compared the score of accuracy in each fold. Thus, we got 5 best k for 5 different folds and we picked the best k by majority rule.

Moreover, we run the model again and replace the default k with the best k we found. Then, the ROC curve and PR curve were used to evaluate our result.

4.2 Logistic Regression

In statistics, the logistic model is used to model the probability of a certain class

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

Figure 1. The equation of logistic curve.

or events would be assigned a probability between 0 and 1 and the sum adding to one. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression.[2] Here is the logistic curve relates the independent variable, X, to the rolling mean of the DV, P (Y).(Fig.1)

The pre-processed and cleaned train data, just like data in KNN, was used to run the logistic regression model from sklearn. The prediction was submitted to Kaggle and we got the accuracy score with default as a baseline.

Furthermore, we tried to improve our accuracy score by using three solvers in L2 penalty, including liblinear, lbfgs and newton-cg. Besides, gradient iteration number was another parameter we take into account. The candidate iteration number contains 100, 150, 200, 250, 300. Thus, we used two loops to find the proper solver as well as the best iteration number. The original train data was split into 5 folders having both train and test groups. The result of parameter was saved as a csv file.

Additionally, we also calculated the class weight, then we run the logistic regression model again and replace the default setting with the best parameter we found. Then, the ROC curve and PR curve were used to evaluate our result.

4.3 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

4.4 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.

4.5 Decision Tree

Decision tree is a prediction method, covering both classification and regression. It uses a tree-like model to make a prediction like its name. Each branch represents the result of the test, and each leaf node shows the class label. Therefore, a competed decision tree can explain how the prediction come.

The first step is to read data. The preprocessed data are read and by using the function of pandas. By checking the data, we notice that there is a column named 'Unnamed: 0', which is the index in the csv file, so we use '.drop()' function to delete this column. After that, we defined the last column as 'labels' to show the result about whether the disaster is real or not and '1' represents 'real disaster' while '0' represents not. And the rest columns became the features. The second step is to import DecisionTressClassifier from sklearn.tree to model the data. After modeling, we got a result csv file. We submitted this result to Kaggle and got score of 0.69325(Fig.1).

Figure 1. Result before Modifying Parameters of Decision Tree.

The next step is to modify the parameters of decision tree to improve the result score. We made two lists of max depth and max leaf nodes to find the numbers with better accuracy by using function of KFold. Max depth is the maximum depth of the tree, which means that no matter how many features can be branched when the depth of the tree reaches max_depth, the decision tree will stop computing. Max leaf nodes represent the maximum number of leaf nodes. Unlimited when it is None. KFold is a cross validation method which can evaluate machine learning models and has a single parameter called k that can divide given data into k groups. As the figure2 shows, we chose max depth = 250 and max leaf nodes = 100 as the final parameters. After modified the parameters, we submitted the result to Kaggle again.

```
max depth=100, average accuracy=0.6950845396143638
max depth=150, average accuracy=0.6948873298861176
max depth=200, average accuracy=0.6951504151988102
max depth=250, average accuracy=0.696835613597779
max depth=300, average accuracy=0.6945150415198811
max depth=None, average accuracy=nan
max leaf nodes=100, average accuracy=0.7126872981922107
max leaf nodes=150, average accuracy=0.7108689681274909
max leaf nodes=200, average accuracy=0.6978214033958497
max leaf nodes=250, average accuracy=0.6914715636341463
max leaf nodes=300, average accuracy=0.6854736211496718
max leaf nodes=None, average accuracy=nan
```

Figure 2. Results of the Modified Parameters

4.6 Support Vector Machine

A support vector machine is a machine learning model which uses classification algorithms to solve the problems of two-group classification.

$$\max \frac{|w^T x + b|}{||w||}$$

Figure 3. The equation of SVM.

We import SVC to model the data and we got scores of 0.62167.

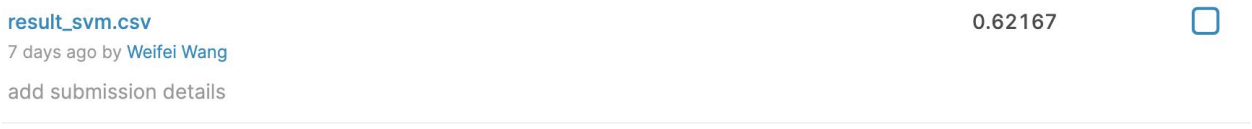

Figure 3. Result before Modifying Parameters of SVM.

And then we modified parameter of ‘C’ and ‘gamma’.

```
best parameter for SVC {'C': 1.15, 'gamma': 'scale'}
best score for SVC 0.8024459815893488
```

Figure 4. Results of the Modified Parameters

5, Result and Evaluation

5.1 K-Nearest Neighbors (KNN)

5.1.1 The parameter in KNN

We compared the score of accuracy in each fold. Then, we got 5 best k for 5 different folds and we picked the best k by majority rule. Thus, the best k is 12 that displayed in Figure 3.

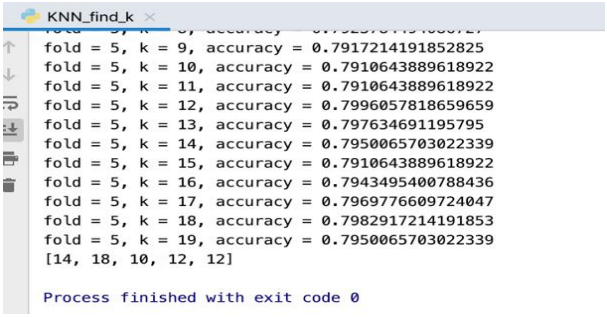

Figure 3. The result of best k value for KNN.

5.1.2 Using all train data to train the model

The score 0.77300 is the baseline of KNN's performance with default (Figure 2).

Figure 2. Kaggle Accuracy Score of KNN (with default).

By submitting the KNN result csv file to Kaggle, it returns the score and the screen shoot of it is shown as Figure 3. Compare the two scores, we find that the accuracy has been improved by 3%.

Figure 3. Kaggle Accuracy Score of KNN (k=12).

5.1.3 ROC curve and PR curve

For KNN model, AUC of ROC curve is 0.85. The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. The AUC describes how good the model is at predicting the positive class when the actual outcome is positive.

The AP for PR curve is 0.82. Compute average precision (AP) from prediction scores

This score corresponds to the area under the precision-recall curve.

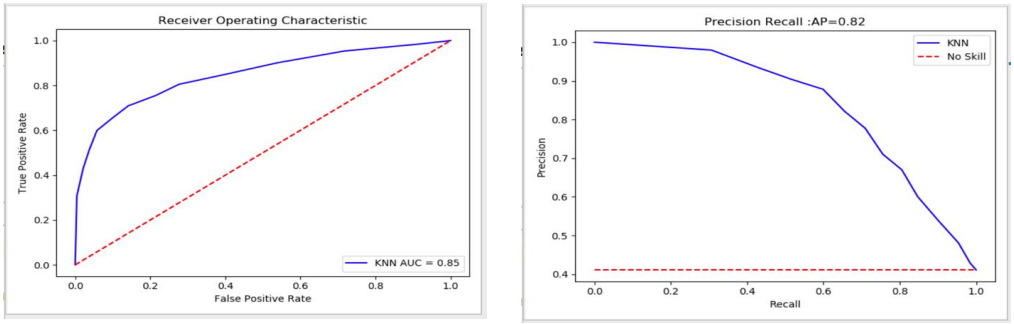

Figure 6. ROC Curve and P-R Curve of KNN Model

5.2 Logistic Regression

5.2.1 The parameter in Logistic Regression

Among "liblinear", "lbfgs" and "newton-cg", "lbfgs" showed the higher average accuracy within five folders. Besides, iteration number equals to 150 performed best among gradient iteration numbers.

```
fold = 5, iteration num = 200, solver = lbfgs, accuracy = 0.7910643889618922
fold = 5, iteration num = 200, solver = newton-cg, accuracy = 0.7917214191852825
fold = 5, iteration num = 250, solver = liblinear, accuracy = 0.7897503285151117
fold = 5, iteration num = 250, solver = lbfgs, accuracy = 0.7897503285151117
fold = 5, iteration num = 250, solver = newton-cg, accuracy = 0.7917214191852825
fold = 5, iteration num = 300, solver = liblinear, accuracy = 0.7897503285151117
fold = 5, iteration num = 300, solver = lbfgs, accuracy = 0.7890932982917214
fold = 5, iteration num = 300, solver = newton-cg, accuracy = 0.7917214191852825
iteration=100, average accuracy=0.7823003334187516
iteration=150, average accuracy=0.7826946090734882
iteration=200, average accuracy=0.7826068900022979
iteration=250, average accuracy=0.7824317682237808
iteration=300, average accuracy=0.7823441641939954
solver=liblinear, average accuracy=0.7824754551972686
solver=lbfgs, average accuracy=0.782606895754368
solver=newton-cg, average accuracy=0.7823443079957514

Process finished with exit code 0
```

Figure 4. The parameter for LR Model. "Iteration=150" and "solver= ldfgs" performed best among these parameters.

5.2.2 Using all train data to train the model

The score 0.80163 is the baseline of LR's performance with default (Figure 5):

Figure 5. Kaggle Accuracy Score of Logistic Regression (with default).

By submitting the Logistic Regression test csv file to Kaggle, it returns the score and the screen shoot of it is shown as Figure 6. The score =0.80061 shows that the parameter changing didn't work well.

Figure 6. Kaggle Accuracy Score of Logistic Regression (iteration=150, solver= "ldfgs")

5.2.3 ROC curve and PR curve

For Logistic Regression, AUC of ROC curve is 0.86. The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. The AUC describes how good the model is at predicting the positive class when the actual outcome is positive.

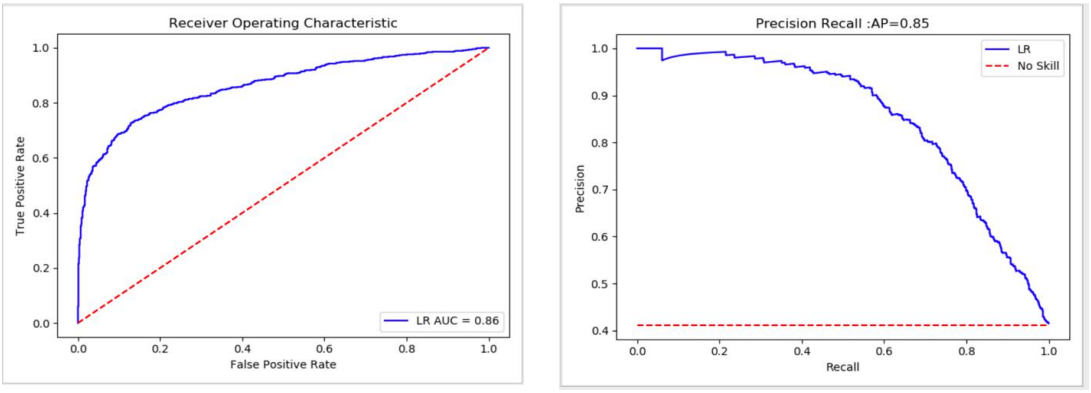

Figure 6. ROC Curve and P-R Curve of Logistic

AP, here is 0.85, is a single number used to summarize the PR curve. Compute average precision (AP) from prediction scores This score corresponds to the area under the precision-recall curve.

5.3 Random Forest

5.3.1 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:

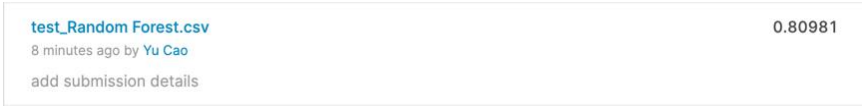

Figure 2. Kaggle Accuracy Score of Random Forest Model

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.

5.3.2 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:

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

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.

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

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

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.

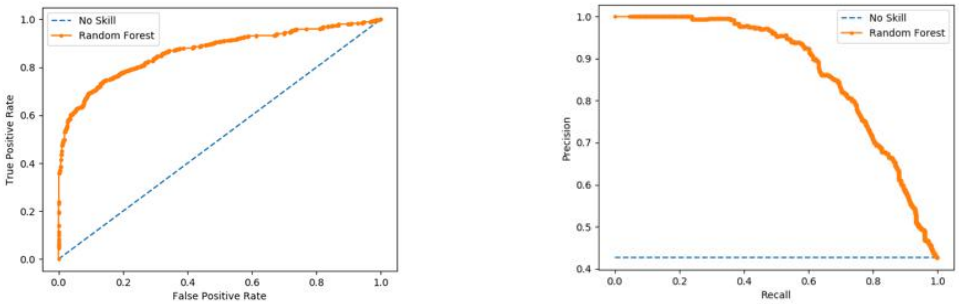

Figure 3. ROC Curve and P-R Curve of Random Forest Model

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.

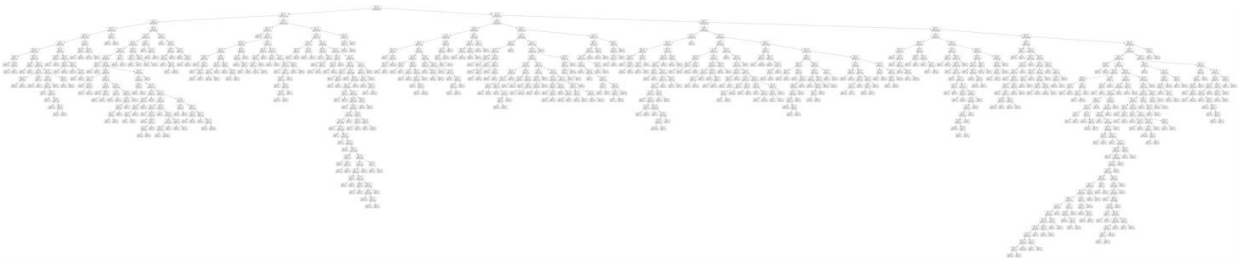

Figure 4. A Single Decision Tree of Random Forest

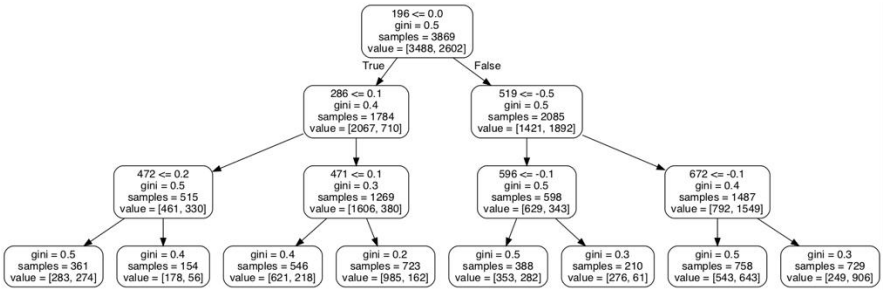

Figure 5. A Smaller Tree

5.4 Naïve Bayes

5.4.1 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:

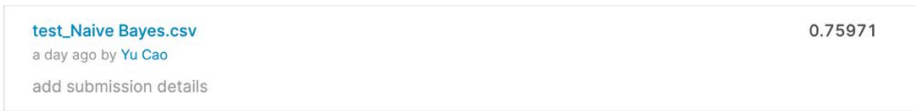

Figure 6. Kaggle Accuracy Score of Naïve Bayes Model

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.

5.4.2 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.

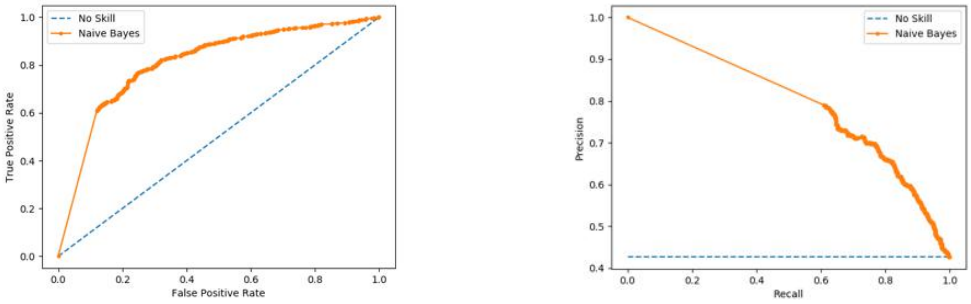

Figure 7. ROC Curve and P-R Curve of Naïve Bayes Model

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.

5.5 Decision Tree

5.5.1 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 5:

Figure 5. Kaggle Accuracy Score of Decision Tree Model

The accuracy of the model is 0.74948, which is relatively low. Therefore, decision tree is not suitable for this data.

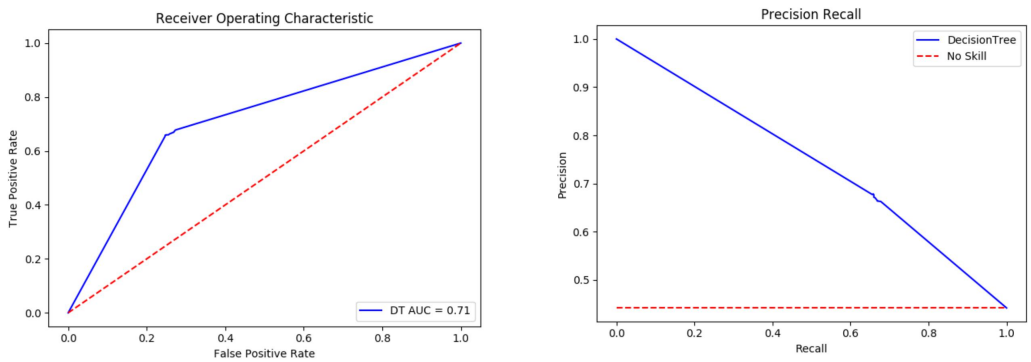

Figure 6. ROC Curve and P-R Curve of Decision Tree Model

ROC Curve (the left figure in Figure 6) and P-R Curve (the right figure in Figure 6) 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.

AUC is the area under the curve. Here, AUC of the ROC curve is 0.71 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.

5.6 Support Vector Machine

5.6.1 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 7:

Figure 7. Kaggle Accuracy Score of Support Vector Machine Model

The accuracy of Naïve Bayes model is 0.81595 which means SVM performs well.

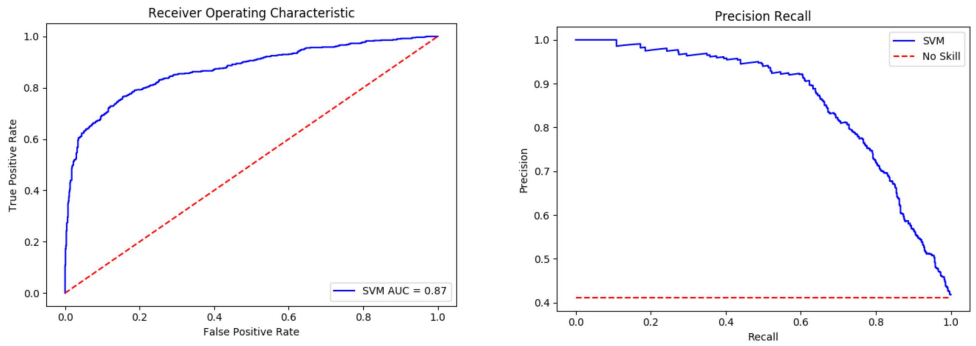

Figure 8. ROC Curve and P-R Curve of SVM Model

Here AUC of ROC curve (the left image of Figure 8) is 0.87. 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. Therefore, we can conclude that support vector machine is a better classification compared with decision tree in this case.

6, Conclusion

In summary, the accuracy score of these six models are:

Model	Accuracy score
KNN	0.80470
Logistic regression	0.80163
Random forest	0.80981
Bayes	0.75971
Decision tree	0.74948
Support vector machine	0.81595

Support vector machine here obtains the best result. What's more, these models all have a relatively good result. Therefore, we can use these 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.

In addition to this, we can try other models, not only traditional machine learning model in the future to obtain a better classification result.

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.
8. Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression" (PDF). The American Statistician. 46 (3): 175–185.
9. Tolles, Juliana; Meurer, William J (2016). "Logistic Regression Relating Patient Characteristics to Outcomes". JAMA. 316 (5): 533–4
10. Brownlee, J. (2016, November 9). How To Implement The Decision Tree Algorithm From Scratch In Python. Retrieved from https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/
11. Brownlee, J. (2018, May 23). A Gentle Introduction to k-fold Cross-Validation. Retrieved from https://machinelearningmastery.com/k-fold-cross-validation/
12. 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/
13. Brownlee, J. (2016, April 20). Support Vector Machines for Machine Learning. Retrieved from https://machinelearningmastery.com/support-vector-machines-for-machine-learning/
14. 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.
15. Gandhi, Rohith. "Naive Bayes Classifier." *Medium*, Towards Data Science, 17

May 2018, towardsdatascience.com/naive-bayes-classifier-81d512f50a7c.

16. 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
17. Yiu, Tony. "Understanding Random Forest." *Medium*, Towards Data Science, 14 Aug. 2019, towardsdatascience.com/understanding-random-forest-58381e0602d2.
18. ZHOU, Z. H. I.-H. U. A. (2020). *Machine Learning*. S.l.: SPRINGER VERLAG, SINGAPOR.