How did you process the data, including any splitting, scaling, and selection. Justify any such decisions.

Firstly, I fill in the training data which is null. Then because in the testing data there is no target label. So, what I did is I split the training data into a new training data set and a new testing data set.

I scale the data which data type is fractional and integer. Also, I use one hot encoder method to convert column text data into one or more columns of data with only zeros and ones and extend features

I used the over-sampling method in this task to balance the dataset because the oversampling method can help me to create a more similar dataset which the number of this data is a small portion of the dataset.

Which machine learning algorithm did you use and why is it suitable for the task at hand?

I used the random forest algorithm as a classification. Because random forest algorithms can deal with multiclassification and the training speed of random forest algorithms is relatively fast, and it is easy to make a parallel method. Also, if there is a significant portion of the feature is missing, the accuracy of using random forest can still be maintained. Finally, for unbalanced data sets, the random forest algorithm can balance out errors.

What are the hyperparameters of your model, what do they represent, and how did you set them?

In a random forest algorithm, the hyperparameters of random forest are “n_estimators”, “oob_score” and “criterion”.

“n_estimators” represents the maximum number of decision trees. Generally speaking, N_ESTIMATORS are too small and tend to be under-fitted. If N_ESTIMATORS is too large, the calculation amount will be too large. Moreover, after N_ESTIMATORS reach a certain number, the model improvement obtained by increasing N_ESTIMATORS will be very small, so a moderate value is generally chosen. So, I set this as 7000.

“oob_score” represents whether the out-of-pocket sample is used to evaluate the model and I set this as “true‘.’

“criterion” represents evaluation criteria for characteristics in CART tree division and I set this using default setting.

“max_features”: represents the maximum number of features randomly selected by each node of the base decision tree and I set it as 10.

What type of regularisation is employed by your model?

the default parameters of random forest are usually fairly good to avoid over-fitting.

But I limit the maximum allowable tree depth. Because if I let random decision trees in the forest get too deep, it may cause over-fitting.

Which optimization strategy did you use and how does it work? What are its strength and weaknesses and why did you choose it over some other approach?

The traditional decision tree model considers all possible features when selecting features, which reduces the diversity of a single tree.

So, the optimization strategy I used is using the advantages of random forest based on ensemble learning, reducing MAX_FEATURES will not only improve the algorithm speed but also improve the algorithm speed. It is also possible to reduce the test error, which is also an improvement of the RF model based on the Bagging ensemble learning method. The choice of max_features is a case-by-case trial until you find a better value.

Which evaluation metrics did you use and why are they suitable for the task at hand? How well did your algorithm perform? How can you be sure that the performance you report is a repeatable result and not a statistical fluke?

I use "metrics.classification_report" to evaluate my task because it is multiclassification, so Confusion Matrix maybe can not give us a straight answer. My algorithm performs around 50% percentage correctness but in Leaderboard the output is lower. And the whole training is processed on a pipeline, the answer is repeatable.

Are some classes easier than others and why do you think that is?

The training dataset is unbalanced. So, some data relative to their target class will be more than other data. So, at the training process, the model will be trained more in this class.

Compare your solution to a simple logistic regression classifier from sklearn and explain any differences.

Simple logistic regression can be called binary classification. The logistic regression model is the probability (P (Y | X)), which is a simple discriminant model. The logical regression model mainly uses maximum likelihood estimation (MLE) to learn the parameters. The logical regression model adopts the Sigmoid activation function to map the domain R to between (0,1). That's mapping a linear combination to a probability.

Logical regression is superior to decision tree in the analysis of the global structure of data, while decision tree is superior to logistic regression in the analysis of local structure.

Logical regression is good at analyzing linear relationships, but it is not as good as Logical Regression in dealing with the online relationship of the decision tree.

Logical regression is sensitive to extreme values and susceptible to the influence of maximum and minimum values, while a decision tree is better than logistic regression in this respect.

The main difference between the two is the algorithmic logic at the bottom. Because of this segmentation approach, the decision tree can drill down into the details of the data, but at the same time lose a grasp of the big picture. This can easily lead to local optimality. Moreover, once a layer of the decision tree is formed, its relationship with nodes in other layers will be cut off, and subsequent

mining can only be carried out locally. At the same time, since the training mode is segmented, the number of training samples keeps decreasing, and multivariate simultaneous detection cannot be supported. Logical regression focuses on the fitting of the overall data, which can better grasp the overall situation. But logistic regression cannot consider local data.