DDA4210 Mini-Project Report

Xiao, Zhijia
120090046

1. Introduction

This mini project aims to classify if a data sample is from the pre-COVID-19 period (label = 1) or the post-COVID-19 period (label = 1) and each data sample x is a vector with 4 features. Firstly, I will display some feature analyses and then focus on data pre-processing. Last part concerns modelling and prediction using an ensemble learning model, the Balanced Bagging with SVM.

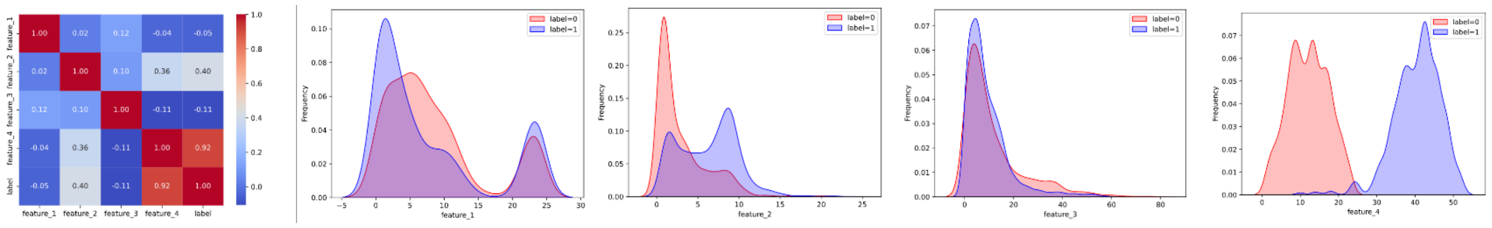
2. Feature Analysis and Pre-processing

2.1 Resampling

According to the training dataset, it has 4000 data with label 1 and 2000 with label 0. Unbalanced data distribution may lead to biased machine learning models that perform poorly on the minority class. Therefore, I apply SVMSMOTE from imblearn to balance the dataset.

2.2 Standardization

From the distribution and correlation matrix of feature 1~4 below, we can see that they are not normal distribution and may have outliers such as feature 4. Besides, more emphasis needs to put on feature 2 and 4 because they have larger correlation with labels.



We need StandardScaler() to make standardization to retain the distribution characteristics of original data, and for secondary processing.

Pseudocode:

```
sm = SVMSMOTE(random_state=0)
over_feature, over_label = sm.fit_resample(total_train_feature, total_train_label)
scaler = preprocessing.StandardScaler().fit(over_feature)
train_transformed = scaler.transform(over_feature)
test_transformed = scaler.transform(final_test)
X_train, X_test, y_train, y_test = train_test_split(train_transformed, over_label, test_size=0.25, random_state=0)
```

3. Modelling

3.1 Simple Modelling

After several tests, I find that it's very easy to overfit on training set. So, I choose SVM as the basic model to lower the complexity and pretend overfitting.

As for hyperparameter tunning, I performed a grid search optimization for SVC classifiers but it turns out to become overfitting because of the gradient descent so I delete it. Instead, I manually tun the hyperparameter and get over 98% accuracy on test set.

Pseudocode:

```
SVC(C=0.01,kernel='rbf',gamma=0.1)
```

3.2 Ensemble Modelling

To improve the soundness and the generalizability, I apply Balanced Bagging Classifier. Firstly, bootstrap aggregating (bagging) can reduce the variance of the model to resist overfitting.

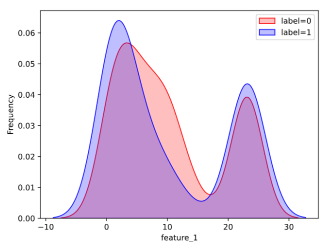
Besides, although I used oversampling in pre-process part, perhaps there is still some differences between labels. So, the balanced version is more proper for this unbalanced dataset.

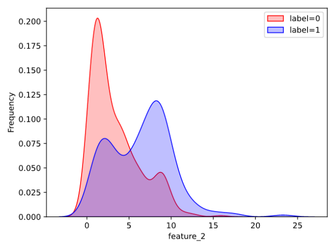
Pseudocode:

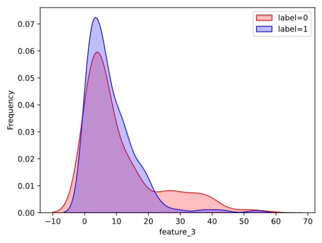
```
clf_bala_bagging = BalancedBaggingClassifier(estimator=SVC(C=0.01,kernel='rbf',gamma=0.1),
n_estimators=10,random_state=0)
clf_bala_bagging.fit(X_train,y_train)
bagging_pred = clf_bala_bagging.predict(X_test)
```

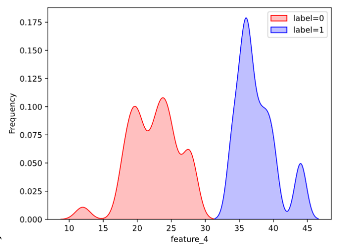
4. Prediction Analysis

We first get the distribution of predicted labels based on feature 1~4 and compare them with the graphs of training set.









Obviously, as for feature 4, label 0 contains all the data whose feature 4 is smaller than 30. However, on training set, feature 4 has some outliers that are smaller than 25. This may cause an error in classification. So, I select all the data whose feature 4 is smaller than 25 and after manually testing, I found that all the error comes from feature 4 equals to 12, but all the other data with feature 4 smaller or equals to 25 is correctly classified.

It might be caused by other features' high weight when classifying because in bagging models, the algorithm will automatically change the weights not only based on the correlations.

After fixing the errors manually, the accuracy rises to 100%, which verifies my hypothesis.