Data Mining Project 1 Report

309552007 袁鈺勛

一、 資料前處理

```python
# Fill NaN with 0
new_info['Comorbidities'] = new_info['Comorbidities'].fillna(0)
new_info['Antibiotics'] = new_info['Antibiotics'].fillna(0)
new_info['Bacteria'] = new_info['Bacteria'].fillna(0)

# Assign unique values to distinct values in the column
new_info['Comorbidities'] = new_info.groupby('Comorbidities').ngroup()

# Assign 0 if it's 0, 1 if it's a string
new_info['Antibiotics'] = new_info.groupby('Antibiotics').ngroup().astype(bool).astype(int)
new_info['Bacteria'] = new_info.groupby('Bacteria').ngroup().astype(bool).astype(int)
```

在 Info sheet，我把共病症、抗體、細菌中的 missing value 都填 0，並把每種共病症組合都編號，抗體和細菌只要欄位是 0 就填 0，非 0 就填 1。

```python
# Group all training data by patient number
sectors = training_tpr.groupby('No')

# Get means of each patient
training_patients = sectors.mean()

# Standardize all columns
ss = StandardScaler()
training_patients[['T', 'P', 'R', 'NBPS', 'NBPD']] = ss.fit_transform(
    training_patients[['T', 'P', 'R', 'NBPS', 'NBPD']])

# Get mean and variance of training data
m = ss.mean_
variance = ss.var_

# Group all testing data by patient number
sectors = testing_tpr.groupby('No')

# Get means of each patient
testing_patients = sectors.mean()

# Scale all columns
testing_patients[['T', 'P', 'R', 'NBPS', 'NBPD']] = (testing_patients[
                                                    ['T', 'P', 'R', 'NBPS', 'NBPD']] - m) / np.sqrt(variance)
```

在 TPR sheet，我把 missing value 填上平均值，並且做 z-score normalization，將 training data 中拿到的平均值和變異數用來標準化 testing data。

二、 Feature 選擇

```python
# Use mutual information to select top k features
list_of_col = SelectKBest(mutual_info_classif, k=k).fit(training_data, training_target).get_support(indices=True)
features = list(map(list(training_data).__getitem__, list_of_col))
```

透過 mutual information 選出和 target 最有關係的前 k 個 features。

三、 模型建置

在預測前，我選了 3 個 model 並透過選出的 features 訓練並測試。

```
# Train and predict
nb = GaussianNB().fit(data_train[features], target_train)
```

第一個是 Naïve Bayes。

```
# Train and predict
svm = SVC(kernel='linear').fit(data_train[features], target_train)
```

第二個是 Linear Support Vector Machine。

```
# Train and predict
dt = DecisionTreeClassifier(max_depth=2).fit(data_train[features], target_train)
```

第三個是 Dicision Tree，最大深度是 2。

四、 驗證方法

```
# Setup K fold
skf = RepeatedStratifiedKFold(n_repeats=10, random_state=0)
```

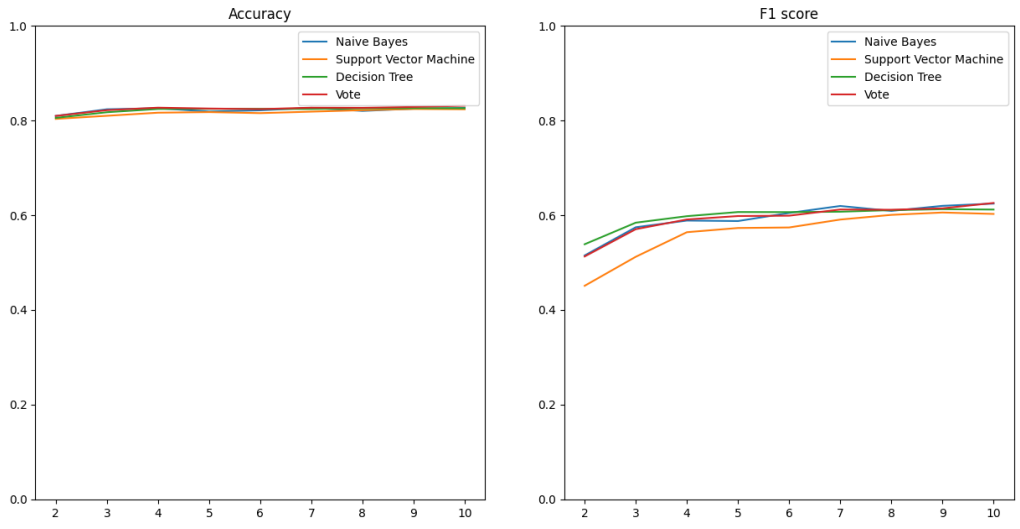
用 stratified Kfold 來測試 3 個 model，切 5 個 fold，測試 10 次。

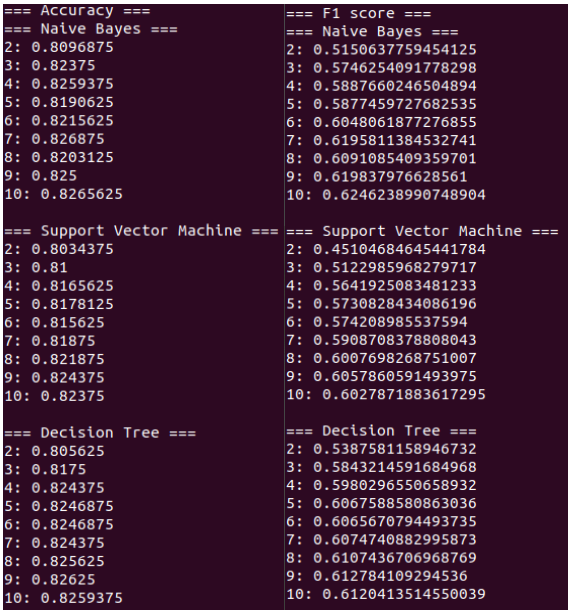
```
# Use training set to select features
for k in range(2, total_features):
    features = feature_selection(data_train, target_train, k)
```

用每次 Kfold 得到的 training set 取出 features，並且測試 2 到全部的 feature 數量。

```
# Return accuracy, f1 score and prediction
return accuracy_score(prediction, target_test), f1_score(prediction, target_test), list(
    prediction), confusion_matrix(target_test, prediction)
```

利用每個 model 得到的結果計算 accuracy、f1 score 來比較。





```
=== Accuracy ===
=== Naive Bayes ===
2: 0.8096875
3: 0.82375
4: 0.8259375
5: 0.8190625
6: 0.8215625
7: 0.826875
8: 0.8203125
9: 0.825
10: 0.8265625

=== Support Vector Machine ===
2: 0.8034375
3: 0.81
4: 0.8165625
5: 0.8178125
6: 0.815625
7: 0.81875
8: 0.821875
9: 0.824375
10: 0.82375

=== Decision Tree ===
2: 0.805625
3: 0.8175
4: 0.824375
5: 0.8246875
6: 0.8246875
7: 0.824375
8: 0.825625
9: 0.82625
10: 0.8259375
```

```
=== F1 score ===
=== Naive Bayes ===
2: 0.5150637759454125
3: 0.5746254091778298
4: 0.5887660246504894
5: 0.5877459727682535
6: 0.6048061877276855
7: 0.6195811384532741
8: 0.6091085409359701
9: 0.619837976628561
10: 0.6246238990748904

=== Support Vector Machine ===
2: 0.45104684645441784
3: 0.5122985968279717
4: 0.5641925083481233
5: 0.5730828434086196
6: 0.574208985537594
7: 0.5908708378808043
8: 0.6007698268751007
9: 0.6057860591493975
10: 0.6027871883617295

=== Decision Tree ===
2: 0.5387581158946732
3: 0.5843214591684968
4: 0.5980296550658932
5: 0.6067588580863036
6: 0.6065670794493735
7: 0.6074740882995873
8: 0.6107436706968769
9: 0.612784109294536
10: 0.6120413514550039
```

根據上面 3 張圖，我選擇 Naïve Bayes 來當作預測時要用的 model。

五、 預測

```python
# Get features
features = feature_selection(training_data, training_target, 7)

# Train the model
nb = GaussianNB().fit(training_data[features], training_target)

# Get prediction
prediction = nb.predict(testing_data[features])
```

根據前面的測試，選擇取出前 7 高的 features，並利用 Naïve Bayes 來訓練以及預測。