Practical 2 Report

Shuyue Wang, Yiwen Wang

December 4, 2021

1 Data Preparation

1.1 iris

To test NBC and logistic regression functions, here we use two data sets, including *iris* and *voting.csv*. The first data set is iris, which we directly loaded it from python.

```
from sklearn.datasets import load_iris
iris = load_iris()
print(iris.target_names)
print(iris.feature_names)
print(iris.target)
print(iris.data)
```

We printed the key information contained in the data set. In Figure 1, the results have been in it.

```
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
```

Figure 1. iris data set

In iris, there is no NAN value or other outliers. So we directly used the original data set in the following steps.

1.2 voting

In the second data set, voting, we firstly checked the whole data set to get an overview.

```
voting = pd.read_csv('./datasets/voting.csv')
voting.info()
```

The information of the data set has been displayed in Figure 2 below.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435 entries, 0 to 434
Data columns (total 17 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   label                                  435 non-null    object
 1   handicapped-infants                    423 non-null    object
 2   water-project-cost-sharing             387 non-null    object
 3   adoption-of-the-budget-resolution      424 non-null    object
 4   physician-fee-freeze                   424 non-null    object
 5   el-salvador-aid                        420 non-null    object
 6   religious-groups-in-schools            424 non-null    object
 7   anti-satellite-test-ban                421 non-null    object
 8   aid-to-nicaraguan-contras              420 non-null    object
 9   mx-missile                             413 non-null    object
 10  immigration                            428 non-null    object
 11  synfuels-corporation-cutback           414 non-null    object
 12  education-spending                     404 non-null    object
 13  superfund-right-to-sue                 410 non-null    object
 14  crime                                  418 non-null    object
 15  duty-free-exports                      407 non-null    object
 16  export-administration-act-south-africa  331 non-null    object
dtypes: object(17)
memory usage: 57.9+ KB
```

Figure 2. voting data set

In order to get more details about what kinds of data would be contained in the data set, we then stepped further on this using code:

```
voting.head()
```

Here we can have a clearer understanding of problems in the data set, such as NAN value. Figure 3 gave the result.

	label	handicapped-infants	water-project-cost-sharing	adoption-of-the-budget-resolution	physician-fee-freeze	el-salvador-aid	religious-groups-in-schools	anti-satellite-test-ban	aid-to-nicaraguan-contras	mx-missile	immigration	synfuels-corporation-cutback	education-spending	superfun right-t s
0	republican	n	y	n	y	y	y	n	n	n	y	NaN	y	
1	republican	n	y	n	y	y	y	n	n	n	n	n	y	
2	democrat	NaN	y	y	NaN	y	y	n	n	n	n	y	n	
3	democrat	n	y	y	n	NaN	y	n	n	n	n	y	n	
4	democrat	y	y	y	n	y	y	n	n	n	n	y	NaN	
5	democrat	n	y	y	n	y	y	n	n	n	n	n	n	
6	democrat	n	y	n	y	y	y	n	n	n	n	n	n	N
7	republican	n	y	n	y	y	y	n	n	n	n	n	n	
8	republican	n	y	n	y	y	y	n	n	n	n	n	y	
9	democrat	y	y	y	n	n	n	y	y	y	n	n	n	

Figure 3. voting data

The data set has 16 features and a label, overall 17 columns. We observed NAN values are included in the data set. As other values are $\boldsymbol{n}$ and $\boldsymbol{y}$ with *object* types, we had better delete the rows containing NAN values and transform the object type to data type using *OrdinalEncoder*. The reason that we did not fill them with other values, like median, is that since the value is either $\boldsymbol{n}$ or $\boldsymbol{y}$, if we randomly replace them with $\boldsymbol{n}$ or $\boldsymbol{y}$, the value may change the distribution drastically. Figure 4 showed the data set without NAN values.

```
voting.dropna().info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 232 entries, 5 to 431
Data columns (total 17 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   label                                   232 non-null    object
 1   handicapped-infants                     232 non-null    object
 2   water-project-cost-sharing              232 non-null    object
 3   adoption-of-the-budget-resolution       232 non-null    object
 4   physician-fee-freeze                    232 non-null    object
 5   el-salvador-aid                         232 non-null    object
 6   religious-groups-in-schools             232 non-null    object
 7   anti-satellite-test-ban                 232 non-null    object
 8   aid-to-nicaraguan-contras               232 non-null    object
 9   mx-missile                              232 non-null    object
 10  immigration                             232 non-null    object
 11  synfuels-corporation-cutback            232 non-null    object
 12  education-spending                      232 non-null    object
 13  superfund-right-to-sue                  232 non-null    object
 14  crime                                   232 non-null    object
 15  duty-free-exports                       232 non-null    object
 16  export-administration-act-south-africa  232 non-null    object
dtypes: object(17)
memory usage: 32.6+ KB
```

Figure 4. voting data without NAN

For the *OrdinalEncoder* coding, we used the folllowing code and results are displayed in Figure 5.

```
voting_encoded = []

for i in range(0,voting.shape[1]):
    ordinal_encoder = OrdinalEncoder()
    voting_encoded_item = ordinal_encoder.fit_transform(voting1)
    np.unique(voting_encoded, return_counts=True)
    voting_encoded.append(voting_encoded_item)

for j in range(0, len(voting_encoded)):
    if j==0:
        arr = np.array(voting_encoded[j])
    else:
        arr1 = np.array(voting_encoded[j])
        arr = np.column_stack((arr, arr1))
```

```
array([[0., 0., 1., ..., 1., 1., 1.],
       [1., 0., 1., ..., 1., 0., 1.],
       [0., 1., 1., ..., 0., 1., 1.],
       ...,
       [1., 0., 0., ..., 1., 0., 1.],
       [1., 0., 0., ..., 1., 0., 1.],
       [0., 0., 0., ..., 0., 0., 1.]])
```

Figure 5. OrdinalOrder Encoding

2 NBC vs. Logisitc

From Figure 6 and Figure 7, we can see that the logistic regression outperformed NBC in both cases. And the turning point in NBC is earlier than logistic regression.

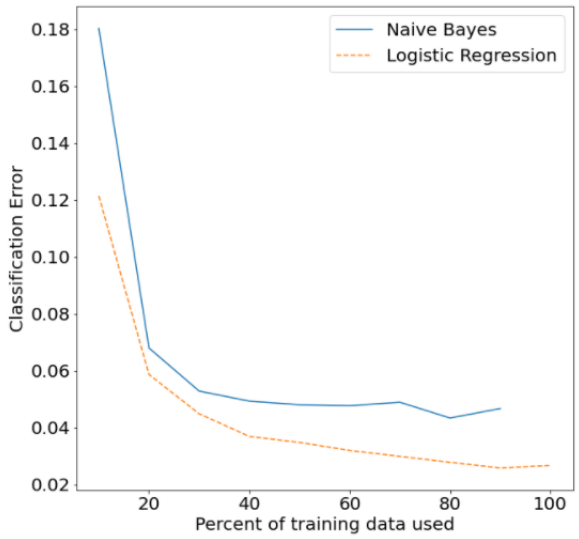

Figure 6. Testing on Iris

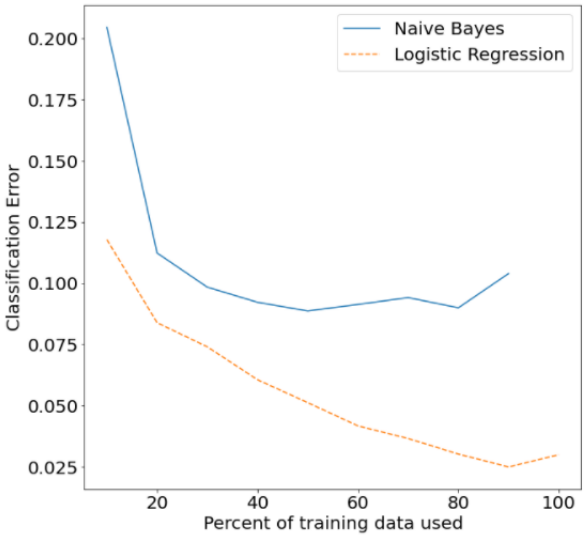

Figure 7. Testing on Voting