Week 6 Report

A Mushroom Classification Model

37237256 薛丁銘

1. Background

This week's report requires me to provide an example of a classification problem that wasn't mentioned in the lecture. However, not many topics came to mind initially. Consequently, I decided to search the Kaggle page for inspiration using various datasets. I came across a dataset related to mushroom classification. The idea struck me—why not train a model by myself as taught in the class and derive some interesting results? Therefore, I developed a logistic regression model for the mushroom classification.

2. Dataset

In this case, the dataset comprises 22 features such as cap-shape, cap-surface, cap-color, etc. The categories of these features are represented in small capital letters, the exact meanings of which are not known to me. The label is denoted as 'class,' indicating two categories of mushrooms, namely 'p' and 'e.' The dataset consists of 8124 examples. The dataset screenshot is depicted in Figure 2.

class	cap-shape	cap-surfac	cap-color	bruises	odor	gill-attachi	gill-spacin	gill-size	gill-color
p	x	s	n	t	p	f	c	n	k
e	x	s	y	t	a	f	c	b	k
e	b	s	w	t	l	f	c	b	n
p	x	y	w	t	p	f	c	n	n
e	x	s	g	f	n	f	w	b	k
e	x	y	y	t	a	f	c	b	n
e	b	s	w	t	a	f	c	b	g
e	b	y	w	t	l	f	c	b	n
p	x	y	w	t	p	f	c	n	p
e	b	s	y	t	a	f	c	b	g
e	x	y	y	t	l	f	c	b	g
e	x	y	y	t	a	f	c	b	n
e	b	s	y	t	a	f	c	b	w
p	x	y	w	t	p	f	c	n	k
e	x	f	n	f	n	f	w	b	n
e	s	f	g	f	n	f	c	n	k
e	f	f	w	f	n	f	w	b	k
p	x	s	n	t	p	f	c	n	n
p	x	y	w	t	p	f	c	n	n

Figure 2 Screenshot of the dataset

3. Logistic Regression Model

I opted for the logistic regression model in this scenario because I believe it is well-suited for handling classification tasks involving categorical features. Additionally, I utilized the 'liblinear' option in the model, as per the documentation, highlighting its efficacy in dealing with binary classification problems. To address convergence issues, I increased the iteration limit to 5000,

considering that the default iteration limit of 100 was insufficient to produce meaningful results. For the training and testing split, 75% of the dataset was allocated for training, while the remaining 25% was reserved for testing. The results and the corresponding code are presented below:

```
Shape of Dataset:  (8124, 23)
datatype of Dataset:  object
Accuracy: 0.9556868537666174
Confusion Matrix:
 [[985  55]
 [ 35 956]]
Coefficients: [[-5.20236669e-03  1.05136311e-01 -1.16184007e-02 -2.49853622e-01
  -3.97561375e-01 -2.24248573e+00 -4.80096433e-01  8.46859706e-01
  -1.08891542e-01  3.01343289e-01  6.42480169e-02 -6.09975237e-01
  -9.35121773e-02 -4.41736471e-02 -3.04631833e-02 -5.95863033e+00
   4.23746606e+00  1.66161894e+00  3.26914645e-01 -2.94360188e-02
  -1.08433183e-01 -3.00522716e-02]]
PS D:\code\school>
```

Figure.3 (a) Prediction Results

From Figure.3 (a) we can see that the accuracy is considerably high as 95.57%. The coefficient indicates that the 8th and 13th features have the most influence on the classification, which are ‘gill-size’ and ‘stalk-surface-below-ring’.

Introduction to Machine Learning > Week06_exercise > mushroom_classification.py > ...

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

import numpy as np
import pandas as pd

# Read the csv file from local
data = np.array(pd.read_csv('d:\code\school\Introduction to Machine Learning\Week06_exercise\mushrooms.csv'))

print('Shape of Dataset: ', data.shape)
print('datatype of Dataset: ', data.dtype)

# Data preprocessing
for row in range(len(data)):
    for column in range(len(data[row])):
        element = data[row, column]
        data[row, column] = ord(element) - ord('a') + 1

label = data[:,0].astype(int)

for i in range(len(label)):
    if label[i] > 5:
        label[i] = 1
    else:
        label[i] = 0
features = data[:,1:].astype(int)

# Training dataset and Test dataset preparation
X_train, X_test, y_train, y_test = train_test_split(features, label, test_size=0.25, random_state=42)

# Logistic Regression
# mushroomsRegression = LogisticRegression(max_iter = 5000)
mushroomsRegression = LogisticRegression(solver = 'liblinear', max_iter = 5000)
mushroomsRegression.fit(X_train, y_train)

mushroomsPredict = mushroomsRegression.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, mushroomsPredict)
conf_marix = confusion_matrix(y_test, mushroomsPredict)
coefficients = mushroomsRegression.coef_

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_marix)
print("Coefficients:", coefficients)
```

Figure. 3(b) Coding