Chair of Decision Sciences and Systems
TUM School of Computation, Information and Technology
Technical University of Munich

Business Analytics & Machine Learning
Homework sheet 3: Logistic Regression – Solution

Prof. Dr. Martin Bichler
Julius Durmann, Markus Ewert, Yutong Chao, Dr. Mete Ahunbay

Exercise H3.1 *Master's course applications*

In this exercise, you will fit a logistic regression model for predicting admittance to Master's courses. You will assess its performance based on various metrics.
Note: Use the provided notebook as a template. It implements some portions of the code so you don't need to start from scratch. Be aware that the code is based on statsmodels.

The data (*admit-train.csv* and *admit-test.csv*) consists of values for the variables *admit*, *gre*, *gpa* and *rank*. The attribute *admit* indicates whether a student has been admitted to a Master's course. The attributes *gre* and *gpa* contain the results of certain exams. The attribute *rank* represents the reputation rank of the student's current university. The smaller the rank, the higher is the university's reputation.

Load the data and visualize it by using the following code (already in the template):

```python
import pandas as pd
# Read data into dataframe
df_train = pd.read_csv("admit-train.csv")
# Show data
print(df_train.head())
# Pairplot
pd.plotting.scatter_matrix(df_train)
```

a) Briefly describe the data set:

1) Name the dependent variable and the independent variables.

2) Which scales of measurement do the variables belong to (e.g., nominal, ordinal, interval or ratio)?

Due to the fact that the dependent attribute *admit* is binary, we use a logistic regression model.
Use the code snippet below (already in the template) to create a logit-model from the training data and to obtain the results.

```python
import statsmodels.formula.api as smf
# Mark "rank" as categorical
df_train["rank"] = df_train["rank"].astype("category")
# Define and fit a model
logreg = smf.logit("admit ~ gre + gpa + rank", data=df_train)
result = logreg.fit()
print(result.summary())
```

b) Which attributes are statistically significant regarding a significance level of 5%?

c) Interpret the coefficients.

d) The *rank* attribute is split into multiple sub-variables (`rank[T.2]`, `rank[T.3]`, `rank[T.4]`) by our model. They indicate whether the rank has a certain value:

$$rank[T.i] = \mathbb{I}[rank = i], \qquad i \in \{2, 3, 4\}$$

For $rank = 1$, we don't need another variable since this case can be inferred if all other rank-variables are zero.
To test the significance of the attribute *rank*, you thus need to test if the null-hypothesis

$$H_0 : \quad \beta_{rank=2} = 0 \quad \wedge \quad \beta_{rank=3} = 0 \quad \wedge \quad \beta_{rank=4} = 0$$

can be rejected.
You can do that by a Wald test which allows to test combined hypotheses as the one above:

```
wald_test_result = result.wald_test(
    "(rank[T.2] = 0, rank[T.3] = 0, rank[T.4] = 0)",
    scalar=True
)
```

Is the attribute *rank* statistically significant w.r.t. a level of $\alpha = 5\%$?

e) In order to gain a better understanding of the model, have a look at the predicted probabilities of some observations. Adjust only one parameter and keep the others constant. For example keep *gre* and *gpa* constant (using their mean/average) and vary *rank*. Can you draw any conclusions?

f) Find the McFadden ratio and interpret the results.

g) Load the test data from *admit-test.csv* and predict the probability of its entries.
Construct the confusion matrix for the test data.

h) Compute the accuracy

$$\text{acc} = \frac{TP + TN}{TP + TN + FP + FN}$$

of your trained model on the test set.

Solution

Please refer to the provided solution notebook.

Exercise H3.2 *Car ownership by age*

You and your friends are have always wondered your chances of getting a table at a bar on a Saturday evening in Munich, without a reservation. You either get a table or not, and can thus be modelled as a binary variable, taking on the value of one when you get a table and zero if you end up not getting in. *Moreover, you have data from Saturday evenings, from 7PM to 10PM.* Therefore, you did what every cool person would think of doing, by running a logistic regression, obtaining the following results:

Variable	Est. coefficient	Standard error
Group size	-.722	.189
Time of day	-3.61	.786
Constant	54.2	2.73

The time of the day is measure on a 24 hour clock for the purposes of the data.

a) According to the model above, what effect (qualitative and quantitative) does a change in group size (+1) have on the dependent variable? If a friend asks to join in at the last minute, would you rather let them?

b) Find the hour of the day in which a group of six friends have a 50% chance of getting a table, according to the model. You can either use an exact formula or rely on an approximate graphical solution. Does it make sense?

Solution

a) We have

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 = 54.2 - .722\,\text{group size} - .3.61\text{time of the day}$$

Hence,

$$e^{-.722} = .489 = \frac{\frac{p(\text{group size}+1)}{1-p(\text{group size}+1)}}{\frac{p(\text{group size})}{1-p(\text{group size})}} = \frac{\text{odds}(\text{group size}+1)}{\text{odds}(\text{group size})} = \text{odds ratio}$$

and

$$\frac{p(\text{group size}+1)}{1-p(\text{group size}+1)} = .489\frac{p(\text{age})}{1-p(\text{age})}.$$

The odds of getting a table decreases by 51.1% if the independent variable group size gains one unit. In other words, it becomes 51.1% more unlikely you get a table. It follows that the probability of getting a table decreases as well, but an exact number cannot be determined in general.

b) Here, we fix the group size to be 6. For the odds to be even, we require:

$$p = 1-p = 0.5 \;\Leftrightarrow\; \frac{p}{1-p} = \frac{0.5}{0.5} = 1 \;\Leftrightarrow\; \ln\left(\frac{p}{1-p}\right) = \ln(1) = 0.$$

Therefore, it holds $\beta_0 + \beta_1 \cdot 6 + \beta_2 x_2^* = 0$ and time of the day$^* = \frac{49.868}{3.61} \simeq 13.81$, or namely about 13:50.
You could also plot this in Python to get an approximate solution. Compute the values of p for different values of age and then plot the result. Recall that,

$$p = \sigma(54.2 - .722\,\text{group size} - .3.61\text{time of the day}) = \frac{e^{54.2-.722\,\text{group size}-.3.61\text{time of the day}}}{1+e^{54.2-.722\,\text{group size}-.3.61\text{time of the day}}}.$$

Here, we have $54.2 - .722 \cdot 6 = 49.868$. So in Python:

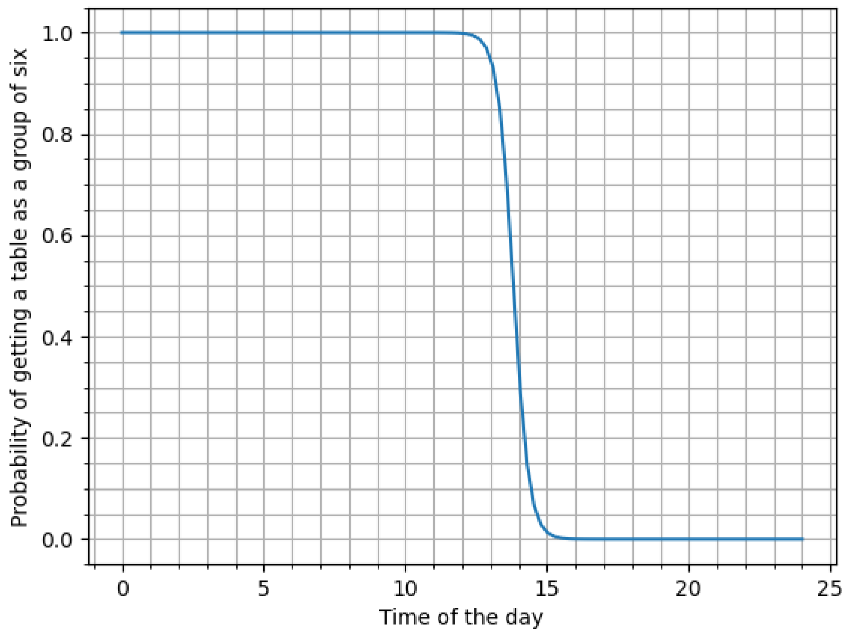
```python
import numpy as np
import matplotlib.pyplot as plt


def sigmoid(x: np.ndarray) -> np.ndarray:
    """Custom implementation of the sigmoid function"""
    return 1 / (1 + np.exp(-x))
```

```
# Model parameters and values
beta_0, beta_2 = 49.868, -3.61
time = np.linspace(0, 24, 100)
p_table = sigmoid(beta_0 + beta_1 * time)

# Plot
plt.minorticks_on()
plt.grid(True, which="both")
plt.plot(time, p_table)
plt.xlabel("Time of the day")
plt.ylabel("Probability of getting a table as a group of six")
plt.show()
```

The figure should look something like this:



This is beside the point, however. Your data is from 7PM to 10PM, and perhaps it is not the best practice to extrapolate from the model to make predictions about lunchtime odds of getting a table.