COMP809 Data Mining and Machine Learning

Patricio Maturana-Russel

p.maturana.russel@aut.ac.nz

Department of Mathematical Sciences and Computer Science and Software Engineering Departments, Auckland University of Technology, Auckland, New Zealand

Semester 1, 2024

Contents

Linear model

In the linear regression model, we have that

$$Y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i,$$

where
- Y_i: response for the ith observation, with $i = 1, \ldots, n$
- x_{ij}: jth predictor for the ith observation, for $j = 0, \ldots, p$
- β_j: parameter
- ϵ_i: error term

And we usually assume that

$$\epsilon_i \overset{\text{iid}}{\sim} \mathsf{N}(0, \sigma^2) \implies Y_i \overset{\text{iid}}{\sim} \mathsf{N}(\mu_i, \sigma^2)$$

where $\mu_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$, and iid stands for independent and identically distributed.

With this analysis we are modeling linearly (in terms of the βs) the mean of the response $\mu_i = \mathsf{E}(Y_i | \boldsymbol{x}_i)$.

Generalised linear model

Question: What if the response Y does not follow a normal distribution?

Answer: Generalised linear model (GLM).

Generalised linear model

GLM assumes that

$$Y \sim \text{Exponential famility distribution},$$

relating its mean to the predictors through a link function g as follows:

$$g(\mu_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip},$$

where g, the link function, maps a non-linear relationship to a linear one, so a linear model can be fit to the data, and $\mu_i = \mathrm{E}(Y_i|\boldsymbol{x}_i)$.

Exponential family:

- Normal
- Bernoulli
- Binomial
- Poisson
- Multinomial
- Exponential
- Beta
- Gamma
- Geometric

Generalised linear model

Distribution	Variable	Parameters	Notation
Bernoulli	indicate a success (1) or a failure (0) of an experiment	p: success probability	$X \sim \text{Bernoulli}(p)$
Binomial	number of successes out of n experiments	n: number of experiments p : success probability	$X \sim \text{Binomial}(n, p)$
Multinomial	number of successes for exactly one of k categories out of n experiments	$p_1, \ldots, p_k$: event probabilities	$\boldsymbol{X} \sim \text{Multinomial}(p_1, \ldots, p_k)$ $\boldsymbol{X} = (X_1, \ldots, X_k)$
Poisson	number of events in a period of time or space	λ: expected number of events within a given time/space interval	$X \sim \text{Poisson}(\lambda)$

Information about the implementation of GLM on Python can be found on www.statsmodels.org.

GLM: logistic regression model

In this model, $Y_i \sim \text{Binomial}(n_i, p_i)$ with

$$\log\left(\frac{p_i}{1-p_i}\right) = \texttt{logit}(p_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip},$$

for $i = 1, \ldots, n$. The link function is the `logit` function.

It is also called logit model.

GLM: probit regression model

In this model, $Y_i \sim \text{Binomial}(n_i, p_i)$ with

$$\Phi^{-1}(p_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip},$$

for $i = 1, \ldots, n$ and where Φ is the cumulative distribution function of a standard normal distribution.

Actually, the inverse of any continuous cumulative distribution function (CDF) can be used as a link function.

The logistic and probit regression models can be used for classification.

GLM: Poisson regression

In this model $Y_i \sim \text{Poisson}(\mu_i)$ with

$$\log(\mu_i) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip},$$

i.e., $\mathsf{E}(Y_i|\boldsymbol{x}_i) = \mu_i = e^{\beta_0+\beta_1 x_{i1}+\cdots+\beta_p x_{ip}}$.

The model for the *expected rate* of the occurrence of event is:

$$\log\left(\frac{\mu_i}{n_i}\right) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$$
$$\log(\mu_i) = \log(n_i) + \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$$

where n_i is an index of the time or space for the i^{th} observation.

Over-dispersion

In the logistic/probit regression model, we have that

$$
\begin{aligned}
Y_i|\boldsymbol{x}_i &\sim \text{Binomial}(n_i, p_i(\boldsymbol{x}_i)) \\
\text{E}(Y_i|\boldsymbol{x}_i) &= n_i \times p_i(\boldsymbol{x}_i) \\
\text{Var}(Y_i|\boldsymbol{x}_i) &= n_i \times p_i(\boldsymbol{x}_i) \times (1 - p_i(\boldsymbol{x}_i))
\end{aligned}
$$

Over (under)-dispersion means the variance of the response does not equal that of a binomial probability model.

Analogously, we have that in the Poisson model

$$
\begin{aligned}
Y_i|\boldsymbol{x}_i &\sim \text{Poisson}(\mu(\boldsymbol{x}_i)) \\
\text{E}(Y_i|\boldsymbol{x}_i) &= \mu_i(\boldsymbol{x}_i) \\
\text{Var}(Y_i|\boldsymbol{x}_i) &= \mu_i(\boldsymbol{x}_i)
\end{aligned}
$$

Over-dispersion: responses are more variable than expected.
Under-dispersion: responses are less variable than expected.

Over-dispersion

Consequences:

- Standard errors are underestimated.
- Consequently, non-reliable p- values.
- Predictions are not necessarily affected.

Causes:

- Important explanatory variables have not been included.
- Outliers.
- When some n_i are small.

Checking over-dispersion:

- Calculate `OD=Pearson chi2`[1]`/Df Residuals`,
  - if `OD` $>>1 \implies$ over-dispersion.
  - if `OD` $<<1 \implies$ under-dispersion.
- Run `model.fit(scale="X2")` or `model.fit(scale="dev")`
  - If `Scale` $>>1 \implies$ over-dispersion.
  - If `Scale` $<<1 \implies$ under-dispersion.

  The scale parameter multiplies the original variance. This is known as **quasi-binomial model**.

[1] it compares the observed and expected probabilities of success and failure at each group of observations.

Over-dispersion

We can also assess the goodness of fit testing the following hypotheses

$$\mathrm{H}_0 : \text{logistic model}$$
$$\mathrm{H}_1 : \text{saturated model}$$

calculating the p-value as follows

$$\mathrm{P}(\chi^2_{df} > \text{Residual Deviance / Degrees of freedom}).$$

When the logistic regression model is an adequate fit to the data and the sample size is large, the deviance has a chi-square distribution with $n - (p + 1)$ degrees of freedom.

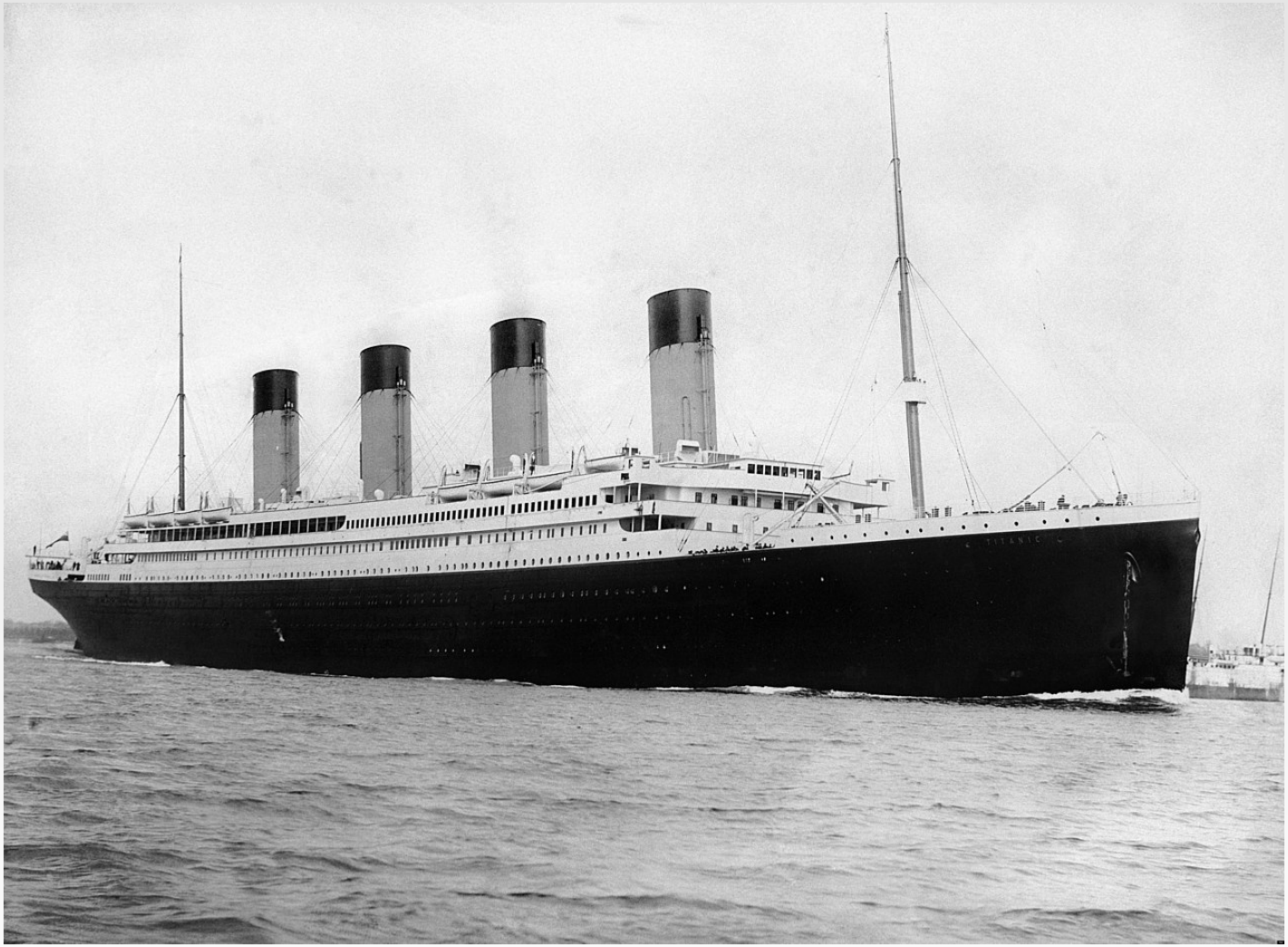
We can get all this information from the Python outputs.

In the case over-dispersion is detected in the Poisson regression, the negative binomial regression can be an alternative.

Titanic

RMS Titanic was a British passenger liner that sank in the North Atlantic Ocean on 15 April 1912, after striking an iceberg during her maiden voyage from Southampton to New York City. Of the estimated 2,224 passengers and crew aboard, more than 1,500 died, making the sinking at the time one of the deadliest of a single ship and the deadliest peacetime sinking of a superliner or cruise ship to date. The ocean liner carried some of the wealthiest people in the world, as well as hundreds of emigrants from Great Britain and Ireland, Scandinavia and elsewhere throughout Europe, who were seeking a new life in the United States. The carried lifeboats were enough for 1,178 people–about half the number on board, and one third of her total capacity–due to the maritime safety regulations of those days. At the time of the sinking, the lowered lifeboats were only about half-filled (Wikipedia).

Titanic



RMS Titanic departing Southampton on April 10, 1912 (Wikipedia).

Titanic

The data set contains several variables, but we are going to consider the following:

1. `survival`: Survival (true or false)
2. `class`: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
3. `sex`: Sex (female or male)
4. `age`: Age

The dataset contains several missing observations which will be excluded from the analysis.

Titanic

We will study if the *sex*, *age* and *class* are related to the probability of surviving the disaster. For this, we will consider the following model:

$$\begin{aligned}\text{survival}_i|\boldsymbol{x}_i &\sim \text{Binomial}(n_i, p_i)\\ \texttt{logit}(p_i) &= \beta_0 + \beta_1 \times \text{sex}_i + \beta_2 \times \text{class2}_i + \beta_3 \times \text{class3}_i + \beta_4 \times \text{age}_i,\end{aligned}$$

where class2 and class3 are dummy variables. For instance, class2 is 1 if the passenger travelled in 2nd class, otherwise is 0. 1st class is considered the baseline.

Titanic

```
>>> import pandas as pd;
>>> titanic = pd.read_csv("titanic.csv");

>>> data    = titanic[["Survived", "Age", "Pclass", "Sex"]];
>>> data    = data.dropna(); # Drop all rows with NaN values

>>> import statsmodels.api as sm;
>>> model  = sm.GLM.from_formula("Survived ~ Age + C(Sex) + C(Pclass)",
                                 family=sm.families.Binomial(), data=data);

>>> result = model.fit();
>>> print(result.summary());
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                Survived   No. Observations:                  714
Model:                            GLM   Df Residuals:                      709
Model Family:                Binomial   Df Model:                            4
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -323.64
Date:                Tue, 31 Jan 2023   Deviance:                       647.28
Time:                        16:36:41   Pearson chi2:                     767.
No. Iterations:                     5   Pseudo R-squ. (CS):             0.3587
```

Titanic

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept          3.7770      0.401      9.416      0.000       2.991       4.563
C(Sex)[T.male]    -2.5228      0.207    -12.164      0.000      -2.929      -2.116
C(Pclass)[T.2]    -1.3098      0.278     -4.710      0.000      -1.855      -0.765
C(Pclass)[T.3]    -2.5806      0.281     -9.169      0.000      -3.132      -2.029
Age               -0.0370      0.008     -4.831      0.000      -0.052      -0.022
==============================================================================
```

Remarks:

- All the predictors are significant.
- Over-dispersion must be checked.

```
>>> dev    = result.deviance; # Residual Deviance
>>> df     = result.df_resid; # Degree of freedoms of Residuals
>>> import scipy;
>>> pvalue = 1 - scipy.stats.chi2.cdf(dev, df); # p-value
>>> print(pvalue);
    0.9526628776499968
```

The Goodness of fit p-value for this model is 0.95, which suggests that there is not enough evidence to say the model is not adequate.

Titanic

We also run a quasi-binomial model, even though it is not necessary, as

```
>>> result2 = model.fit(scale="X2");
>>> print(result2.summary());
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                Survived   No. Observations:                  714
Model:                            GLM   Df Residuals:                      709
Model Family:                Binomial   Df Model:                            4
Link Function:                  Logit   Scale:                          1.0822
Method:                          IRLS   Log-Likelihood:                -323.64
Date:                Wed, 01 Feb 2023   Deviance:                       647.28
Time:                        13:32:27   Pearson chi2:                     767.
No. Iterations:                     7   Pseudo R-squ. (CS):             0.3587
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept          3.7770      0.417      9.051      0.000       2.959       4.595
C(Sex)[T.male]    -2.5228      0.216    -11.693      0.000      -2.946      -2.100
C(Pclass)[T.2]    -1.3098      0.289     -4.528      0.000      -1.877      -0.743
C(Pclass)[T.3]    -2.5806      0.293     -8.814      0.000      -3.154      -2.007
Age               -0.0370      0.008     -4.644      0.000      -0.053      -0.021
==============================================================================
```

Titanic

Note that

- Scale = 1.08 is approximately 1, which indicates that the model fits well the data.
- Deviance / Df Residuals = 647.28/709 = 0.91 is close to 1.

The model is adequate.

Note that in general the quasi binomial has higher standard errors.

Overdispersion cannot occur when ungrouped binary data are being modelled. For a discussion about it, see www.highstat.com.

Titanic

Interpretation:

β_1: Holding *age* and *class* at a fixed value, the expected odds of surviving for males (sex=1) over the odds of surviving for females (sex=0) is exp(-2.5228) = 0.08. The odds are reduced by a factor of 0.08. In terms of percentage change, the odds of surviving for males are reduced by 92%.

β_4: Holding *sex* and *class* at a fixed value, we expect that the odds of surviving decrease in exp(-0.037) = 0.96 for a year increase. In terms of percentage, the odds are reduced by 4% for a year increase.

Titanic

We are going to predict the survival probability for someone 19 years old, female and male, traveling in first, second and third class.

```
>>> predXf = {"Age":[19,19,19], "Sex":["female","female","female"], "Pclass":[1,2,3]};
>>> predXm = {"Age":[19,19,19], "Sex":["male","male","male"], "Pclass":[1,2,3]};
>>> predXf = pd.DataFrame(data=predXf);
>>> predXm = pd.DataFrame(data=predXm);

>>> pred_pbb_female = result.predict(predXf);
>>> print(pred_pbb_female);
    [0.955820, 0.853772, 0.620970]
>>> pred_pbb_male = result.predict(predXm);
>>> print(pred_pbb_male);
    [0.634486, 0.319018, 0.116180]
```

	Class		
	1	2	3
Female	0.96	0.85	0.62
Male	0.63	0.32	0.12

To estimate if the person survive, we need to specify a threshold, for instance:

```
>>> predictions_female = ["0" if x < 0.5 else "1" for x in pred_pbb_female];
>>> predictions_male   = ["0" if x < 0.5 else "1" for x in pred_pbb_male];
>>> print(predictions_female);
    ['1', '1', '1']
>>> print(predictions_male);
    ['1', '0', '0']
```

Model evaluation

Confusion matrix

	Model prediction	
Observation	No survived (0)	Survived (1)
No survived (0)	TN	FP
Survived (1)	FN	TP

TN = True Negative
TP = True Positive
FN = False Negative
FP = False Positive

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

$$\text{Sensitivity} = \frac{\text{TP}}{\text{FN}+\text{TP}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}}$$

Model evaluation

```
# Checking proportion of 0s and 1s
>>> response_count = data.groupby("Survived")["Survived"].count();
>>> print(response_count);
    Survived
    0    424
    1    290
>>> print("Percentage of 0s:", 100*response_count[0]/np.sum(response_count));
    Percentage of 0s: 59.38375350140056
>>> print("Percentage of 1s:", 100*response_count[1]/np.sum(response_count));
    Percentage of 1s: 40.61624649859944

# The data is not dramatically unbalanced.
>>> X = data.iloc[:, 1:]; # predictors
>>> y = data['Survived']; # response

# We define training and testing sets.
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                        random_state=0,shuffle=True);
>>> df_train = pd.concat([X_train, y_train], axis = 1);

>>> model  = sm.GLM.from_formula("Survived ~ Age + C(Sex) + C(Pclass)",
                                 family=sm.families.Binomial(), data=df_train);
>>> result = model.fit();

# Predictions
>>> predictions = result.predict(X_test);
```

Model evaluation

```
>>> predictions_nominal = [ 0 if x < 0.5 else 1 for x in predictions];

>>> from sklearn.metrics import confusion_matrix, classification_report
>>> cm = confusion_matrix(y_test, predictions_nominal)
>>> print("Confusion matrix: ", cm);
    [[109  16]
     [ 20  70]]

>>> print("Acuraccy: ", round(np.sum(np.diagonal(cm))/np.sum(cm),3));
    Acuraccy:  0.833
>>> print("Sensitivity: ", round(cm[1,1]/np.sum(cm[1,:]),3));
    Sensitivity:  0.778
>>> print("Specificity: ", round(cm[0,0]/np.sum(cm[0,:]),3));
    Specificity:  0.872
```

Model evaluation

The logistic regression model correctly predicted

- the survived variable 83.3% of the times (accuracy),
- 77.8% of the times those who survived (sensitivity), and
- 87.2% of the times those who did not survive (specificity).

Alternatively,

```
# We can also get those values as follows
>>> print(classification_report(y_test, predictions_nominal, digits = 3));
                   precision     recall  f1-score  support
                0      0.845      0.872     0.858      125
                1      0.814      0.778     0.795       90

         accuracy                           0.833      215
        macro avg      0.829      0.825     0.827      215
     weighted avg      0.832      0.833     0.832      215
```

Model evaluation

About the classification report:

Metrics	Definition
Precision	It is defined as the ratio of true positives to the sum of true and false positives.
Recall	It is defined as the ratio of true positives to the sum of true positives and false negatives.
F1 Score	It is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is.
Support	It is the number of actual occurrences of the class in the dataset. It does not vary between models, it just diagnoses the performance evaluation process.

End