Logistic Regression Basics

Justin Post

Logistic Regression Model

Used when you have a **binary** response variable

- Consider just a binary response
  - What is the mean of the response?

Logistic Regression Model

Suppose you have a predictor variable as well, call it x

- Given two values of x we could model separate proportions

$$E(Y|x = x_1) = P(Y = 1|x = x_1)$$

$$E(Y|x = x_2) = P(Y = 1|x = x_2)$$

Logistic Regression Model

Suppose you have a predictor variable as well, call it x

- Given two values of x we could model separate proportions

$$E(Y|x=x_1) = P(Y=1|x=x_1)$$

$$E(Y|x=x_2) = P(Y=1|x=x_2)$$

- For a continuous x, we could consider a SLR model

$$E(Y|x) = P(Y=1|x) = \beta_0 + \beta_1 x$$

Linear Regression Isn't Appropriate

- Consider data about water potability

```python
import pandas as pd
water = pd.read_csv("data/water_potability.csv")
water.head()
```

```
##          ph    Hardness        Solids  ...  Trihalomethanes  Turbidity  Potability
## 0       NaN  204.890455  20791.318981  ...        86.990970   2.963135           0
## 1  3.716080  129.422921  18630.057858  ...        56.329076   4.500656           0
## 2  8.099124  224.236259  19909.541732  ...        66.420093   3.055934           0
## 3  8.316766  214.373394  22018.417441  ...       100.341674   4.628771           0
## 4  9.092223  181.101509  17978.986339  ...        31.997993   4.075075           0
##
## [5 rows x 10 columns]
```

Potability Summary

- Summarize water potability

```
water.Potability.value_counts()
```

```
## 0    1998
## 1    1278
## Name: Potability, dtype: int64
```

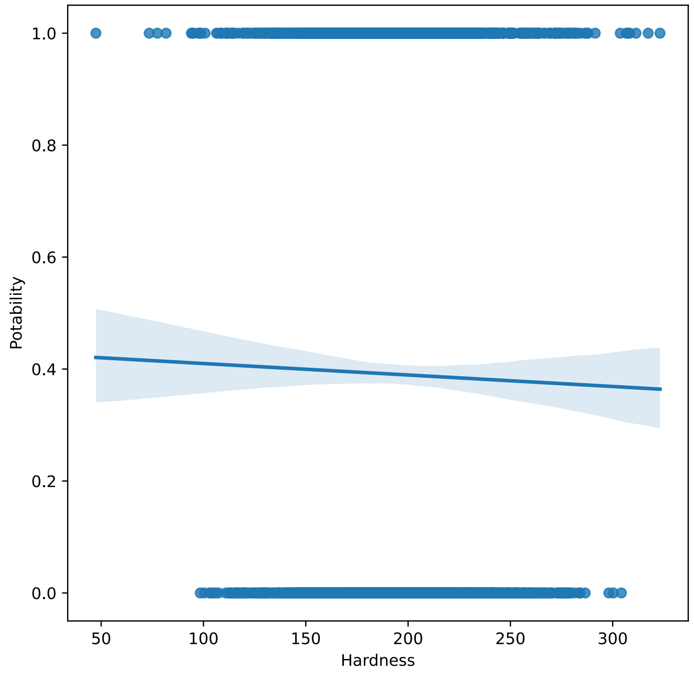
```
water.groupby("Potability")[["Hardness", "Chloramines"]].describe()
```

```
##            Hardness                          ... Chloramines
##               count        mean        std   ...         50%       75%        max
## Potability                                   ...
## 0            1998.0  196.733292  31.057540   ...    7.090334  8.066462  12.653362
## 1            1278.0  195.800744  35.547041   ...    7.215163  8.199261  13.127000
##
## [2 rows x 16 columns]
```

Linear Regression Isn't Appropriate

- Plot SLR model fit

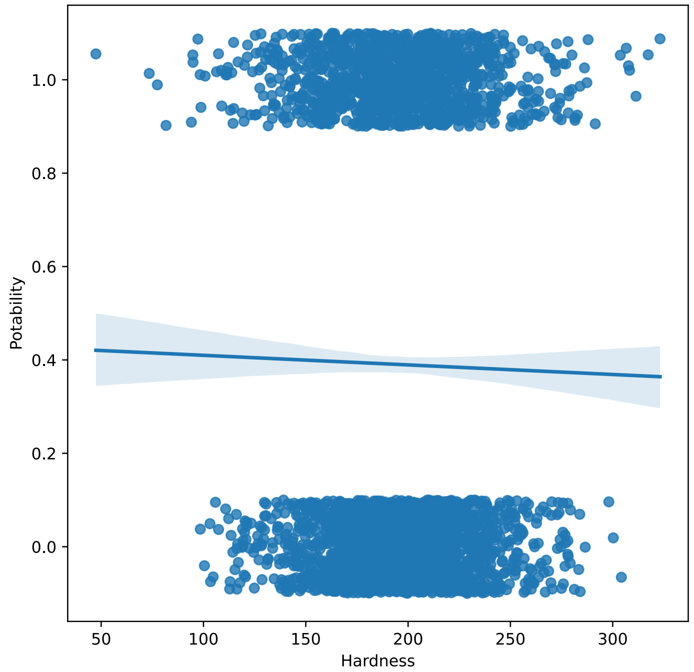
```
import seaborn as sns
sns.regplot(x = water["Hardness"], y = water["Potability"])
```



Linear Regression Isn't Appropriate

- Plot SLR model fit with jittered points

```
import seaborn as sns
sns.regplot(x = water["Hardness"], y = water["Potability"], y_jitter = 0.1)
```



NC STATE UNIVERSITY

Linear Regression Isn't Appropriate

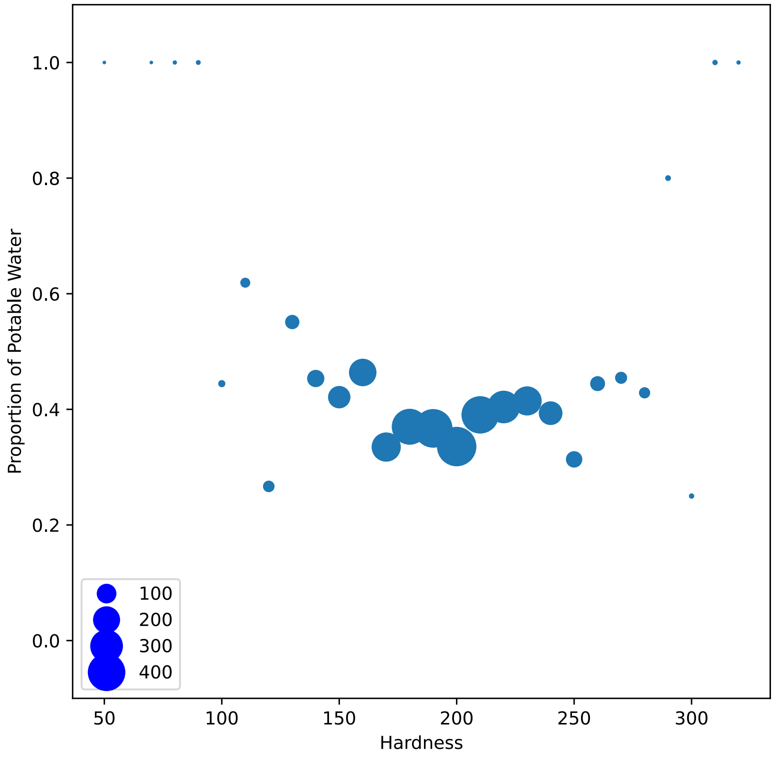
```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

water["Hardnessgroups"] = pd.cut(water['Hardness'], range(45, 335, 10))
props = water[["Hardnessgroups", "Potability"]] \
    .groupby("Hardnessgroups") \
    .agg(prop = ('Potability', 'mean'), counts = ('Potability', 'count'))

sc = plt.scatter(pd.Series(range(50,330,10)), props.prop, s = props.counts)
plt.xlabel("Hardness")
plt.ylabel("Proportion of Potable Water")
plt.ylim([-0.1, 1.1])
plt.legend(*sc.legend_elements("sizes", num=5, color = "blue"))
plt.show()
```

Linear Regression Isn't Appropriate

```
## Text(0.5, 0, 'Hardness')
## Text(0, 0.5, 'Proportion of Potable Water')
## (-0.1, 1.1)
```



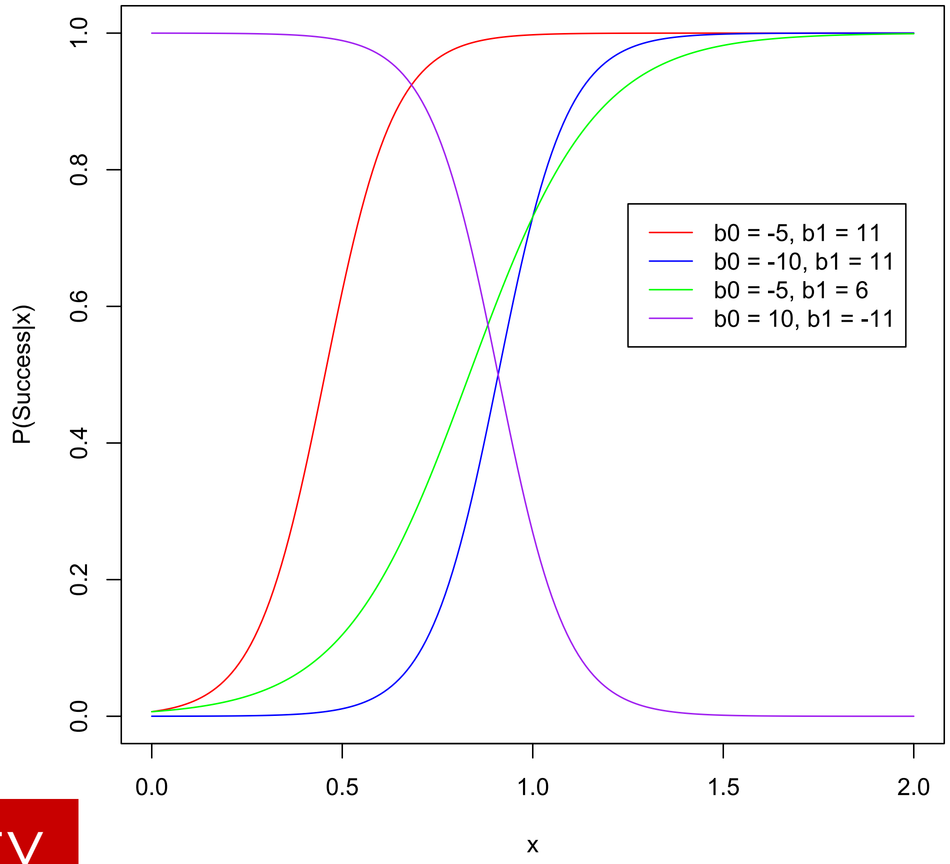
NC STATE UNIVERSITY

Logistic Regression

- Response = success/failure, then modeling average number of successes for a given x is a probability!
    - predictions should never go below 0
    - predictions should never go above 1
- Basic Logistic Regression models success probability using the *logistic function*

$$P(Y = 1|x) = P(success|x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

Logistic Regression



Logistic Regression

$$P(Y=1|x) = P(success|x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$$

- The logistic regression model doesn't have a closed form solution (maximum likelihood often used to fit parameters)

Logistic Regression

$$P(Y = 1|x) = P(success|x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

- The logistic regression model doesn't have a closed form solution (maximum likelihood often used to fit parameters)
- Back-solving shows the *logit* or *log-odds* of success is linear in the parameters

$$log\left(\frac{P(success|x)}{1 - P(success|x)}\right) = \beta_0 + \beta_1 x$$

NC STATE UNIVERSITY

Logistic Regression

$$P(Y=1|x) = P(success|x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$$

- The logistic regression model doesn't have a closed form solution (maximum likelihood often used to fit parameters)
- Back-solving shows the *logit* or *log-odds* of success is linear in the parameters

$$log\left(\frac{P(success|x)}{1-P(success|x)}\right) = \beta_0 + \beta_1 x$$

- Coefficient interpretation changes greatly from linear regression model!
- β_1 represents a change in the log-odds of success

Hypotheses of Interest

For inference, what do you think would indicate that x is related to the probability of success here?

Fitting a Logistic Regression Model in Python

- Use `sklearn` to fit model

```
from sklearn.linear_model import LogisticRegression
```

- Similar to fitting an MLR model, we create an instance and then use the `.fit()` method

NC STATE UNIVERSITY

Fitting a Logistic Regression Model in Python

- Use `sklearn` to fit model

```
from sklearn.linear_model import LogisticRegression
```

- Similar to fitting an MLR model, we create an instance and then use the `.fit()` method

```
log_reg = LogisticRegression(penalty = 'none')
log_reg.fit(X = water["Hardness"].values.reshape(-1,1), y = water["Potability"].values)
```

```
print(log_reg.intercept_, log_reg.coef_)
```

```
## [-0.27748213] [[-0.00086296]]
```

Prediction with a Logistic Regression Model

- Still use the `.predict()` method to predict success or failure

```
import numpy as np
log_reg.predict(np.array([[50], [150], [200], [250], [300]]))
```

```
## array([0, 0, 0, 0, 0], dtype=int64)
```

Prediction with a Logistic Regression Model

- Still use the `.predict()` method to predict success or failure

```
import numpy as np
log_reg.predict(np.array([[50], [150], [200], [250], [300]]))
```

```
## array([0, 0, 0, 0, 0], dtype=int64)
```

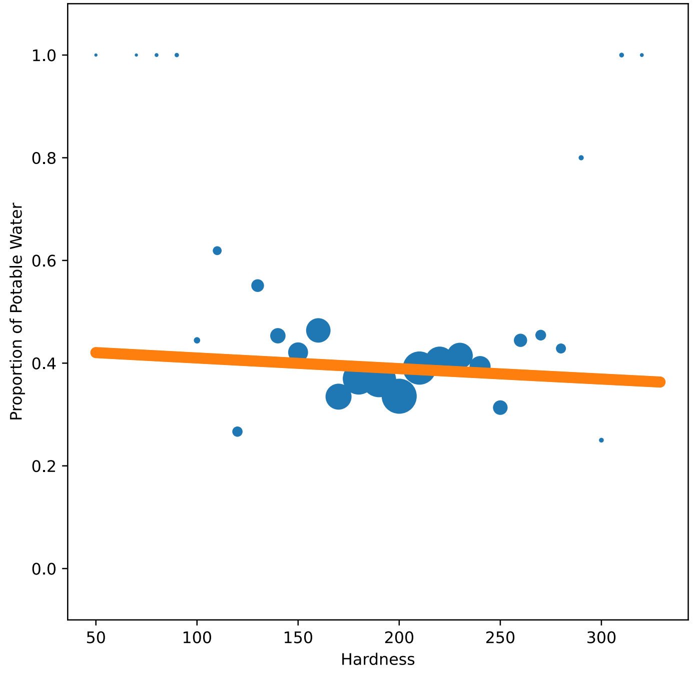
- Also have `.predict_log_proba()` and `.predict_proba()` to obtain log probabilities and probabilities, respectively

```
log_reg.predict_proba(np.array([[50], [150], [200], [250], [300]]))
#returns P(Y=0), P(Y=1) estimates for each value
```

```
## array([[0.57947776, 0.42052224],
##        [0.60035045, 0.39964955],
##        [0.61065667, 0.38934333],
##        [0.62086496, 0.37913504],
##        [0.63096734, 0.36903266]])
```

Plotting the Fit

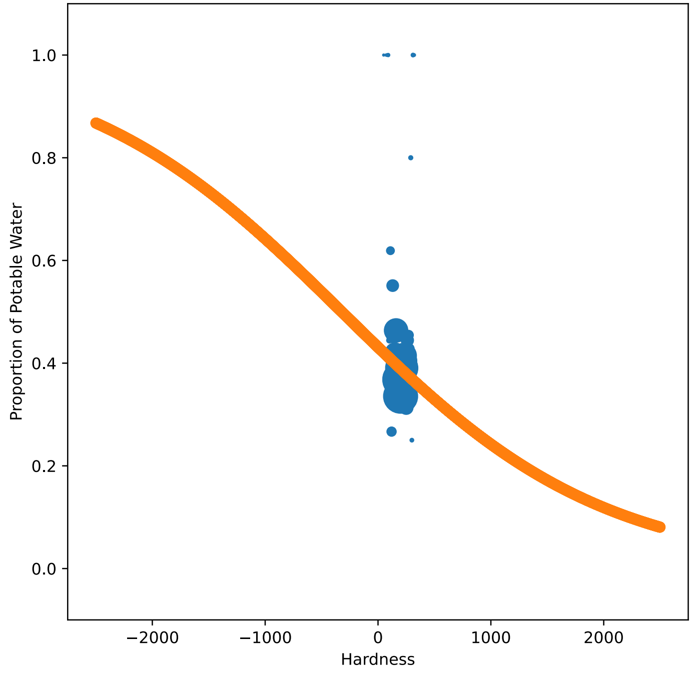
```
sc = plt.scatter(pd.Series(range(50,330,10)), props.prop, s = props.counts)
preds = log_reg.predict_proba(np.array(range(50,330)).reshape(-1,1))
plt.scatter(x = np.array(range(50,330)), y = preds[:,1])
plt.ylim([-0.1,1.1]); plt.xlabel("Hardness"); plt.ylabel("Proportion of Potable Water"); plt.show()
```



NC STATE UNIVERSITY

Truly is a sigmoid type function!

```
preds = log_reg.predict_proba(np.array(range(-2500,2500)).reshape(-1,1))
plt.scatter(pd.Series(range(50,330,10)), props.prop, s = props.counts)
plt.scatter(x = np.array(range(-2500,2500)), y = preds[:,1])
plt.ylim([-0.1,1.1]); plt.xlabel("Hardness"); plt.ylabel("Proportion of Potable Water"); plt.show()
```



Inference with a Logistic Regression Model

- Not implemented in `sklearn`... can use `statsmodels` package!

```
import statsmodels.api as sm
log_reg = sm.GLM(water["Potability"], water["Hardness"], family=sm.families.Binomial())
res = log_reg.fit()
print(res.summary())
```

```
##                  Generalized Linear Model Regression Results
## ==============================================================================
## Dep. Variable:             Potability   No. Observations:                 3276
## Model:                            GLM   Df Residuals:                     3275
## Model Family:                Binomial   Df Model:                            0
## Link Function:                  Logit   Scale:                          1.0000
## Method:                          IRLS   Log-Likelihood:                -2191.5
## Date:                Fri, 14 Mar 2025   Deviance:                       4383.0
## Time:                        17:56:34   Pearson chi2:                 3.28e+03
## No. Iterations:                     4   Pseudo R-squ. (CS):         -0.0003092
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          z      P>|z|      [0.025      0.975]
## ------------------------------------------------------------------------------
## Hardness      -0.0022      0.000    -12.421      0.000      -0.003      -0.002
## ==============================================================================
```

Recap

- Logistic regression often a reasonable model for a binary response
- Uses a sigmoid function to ensure valid predictions
- Can predict success or failure using estimated probabilities
    - Usually predict success if probability > 0.5