# Logistic Regression

**Professor Song Yao**
Olin Business School

**Customer Analytics**



# Why do we have for predictive models so far?

# RFM has some major shortcomings

## RFM DISADVANTAGES

- Not very 'sophisticated,' i.e. built as rule of thumb
- Does not "scale" well to include other variables
- Predicts average "response rate" for a customer on the basis of **membership in a specific RFM cell**
- Does not predict individual "response probability" based on **individual customer characteristics**

**Need a more flexible, powerful model to predict response / purchase probability**



# Many firms monitor the activities of potential customers to determine whether they might be good leads

## EXAMPLE: SMARTSTORAGE

- A top cloud storage provider (huge capacity, speed less important)
- Serve many clients with large cloud storage needs (e.g. major photo sharing websites)
- Have limited number of potential clients
- Can identify most potential decision makers on their site without requiring them to log in
- Identified 14,000 visitors to their website who are potential buyers of their service

## Smartstorage keeps track of behaviors and demographics of potential decision makers

### BEHAVIORS

**Website**

- Visited webpage/blog
- Viewed introductory content
- Viewed mid-stage content
- Viewed late-stage content
- Visited pricing page
- Visited career page
- Watched demos

**Other**

- Visited at trade-show
- Contacted company
- Provided e-mail
- ...

### DEMOGRAPHICS

- High-relevance employer
- High-relevance job title
- Relevant past experience
- Small potential client
- Large potential client
- ...



## How would we implement predictive analytics for lead scoring at Smartstorage? (simplified example)

### BEHAVIORS

**Website**

- Visited webpage/blog
- Viewed introductory content
- Viewed mid-stage content
- Viewed late-stage content
- Visited pricing page
- Visited career page
- Watched demos

**Other**

- Visited at trade-show
- Contacted company
- Provided e-mail
- ...

### DEMOGRAPHICS

- High-relevance firm
- High-relevance job title
- Relevant past experience
- Small potential client
- Large potential client
- ...

**Sale (Purchase)?**

## We use information on 180 leads, including whether they converted to a sale within 150 days of first ID

| id | webpageviews | viewedpricing | highrelevancefirm | sale |
|---|---|---|---|---|
| 639 | 15 | 0 | 1 | 0 |
| 272 | 35 | 0 | 1 | 1 |
| 491 | 7 | 0 | 1 | 0 |
| 226 | 18 | 1 | 1 | 0 |
| 7195 | 13 | 0 | 0 | 0 |
| 9080 | 23 | 0 | 0 | 0 |
| 548 | 14 | 0 | 1 | 0 |
| 9605 | 36 | 0 | 0 | 0 |
| 5352 | 28 | 0 | 0 | 0 |
| 4343 | 35 | 0 | 0 | 0 |
| 14971 | 3 | 0 | 0 | 0 |
| 11298 | 34 | 0 | 0 | 1 |
| 317 | 10 | 0 | 1 | 0 |
| … | … | … | … | … |



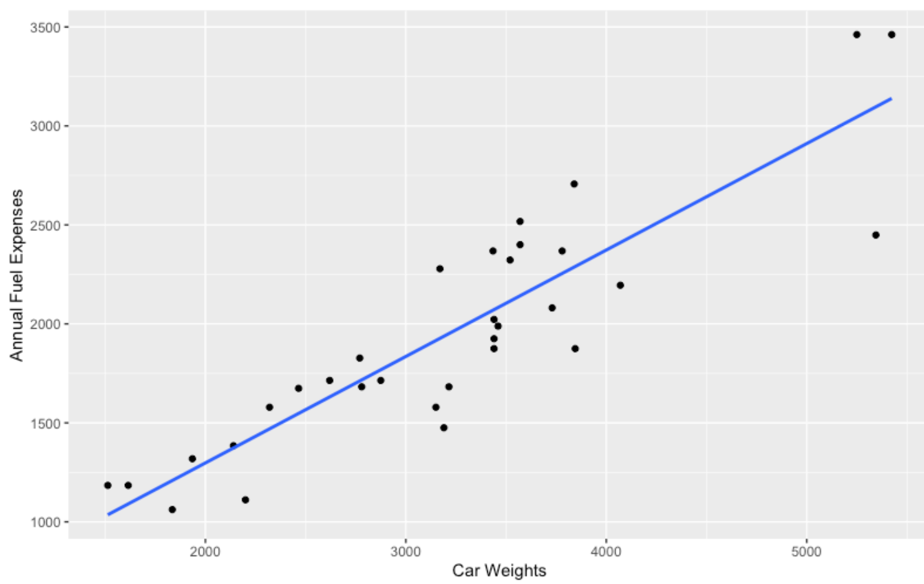
## How would we determine the relationship between demographics and behaviors and sales success?

| id | webpageviews | viewedpricing | highrelevancefirm | sale |
|---|---|---|---|---|
| 639 | 15 | 0 | 1 | 0 |
| 272 | 35 | 0 | 1 | 1 |
| 491 | 7 | 0 | 1 | 0 |
| 226 | 18 | 1 | 1 | 0 |
| 7195 | 13 | 0 | 0 | 0 |
| 9080 | 23 | 0 | 0 | 0 |
| 548 | 14 | 0 | 1 | 0 |
| 9605 | 36 | 0 | 0 | 0 |
| 5352 | 28 | 0 | 0 | 0 |
| 4343 | 35 | 0 | 0 | 0 |
| 14971 | 3 | 0 | 0 | 0 |
| 11298 | 34 | 0 | 0 | 1 |
| 317 | 10 | 0 | 1 | 0 |
| … | … | … | … | … |

| id | webpageviews | viewedpricing | highrelevancefirm | sale |
|---|---|---|---|---|
| 7236 | 27 | 0 | 0 | |
| 687 | 25 | 1 | 1 | ? |
| 453 | 16 | 0 | 0 | |
| 563 | 6 | 0 | 1 | |

## How do data scientists create a prediction other than RFM?

### EXAMPLE: YEARLY FUEL COST AND VEHICLE WEIGHT



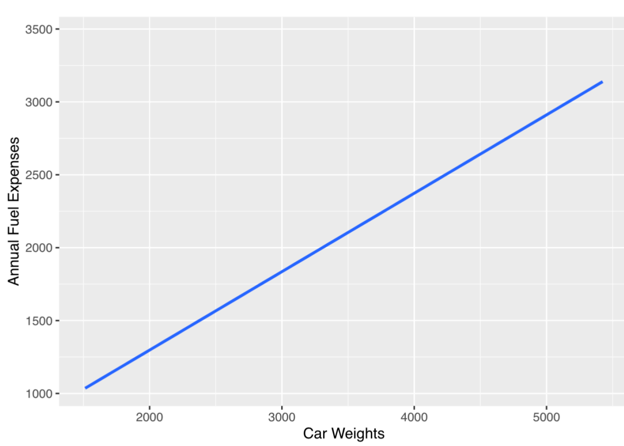

**Inputs to calculate yearly fuel cost (YFC):**

- Weight of vehicle
- 12,000 miles per year
- $ 3 / gallon



## How do data scientists create a prediction using a regression?

### EXAMPLE: YEARLY FUEL COST AND VEHICLE WEIGHT

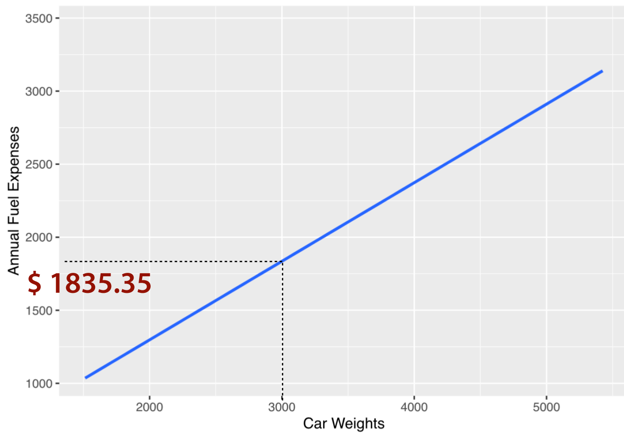



| weight | total_gas_expense | predicted_value |
|---|---|---|
| 3000 | NA | ? |
| 4500 | NA | ? |
| 2350 | NA | ? |

**YFC = 220.08 + 0.54*Weight**

## How do data scientists create a prediction using a regression?

### EXAMPLE: YEARLY FUEL COST AND VEHICLE WEIGHT



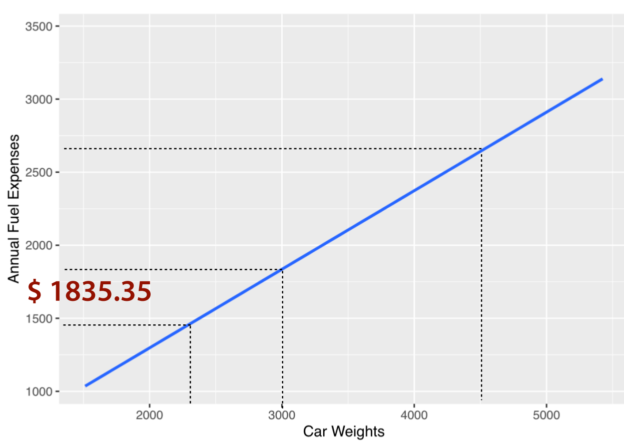

| weight | total_gas_expense | predicted_value |
|---|---|---|
| 3000 | NA | 1835.353 |
| 4500 | NA | ? |
| 2350 | NA | ? |

**YFC = 220.08 + 0.54*Weight**



## How do data scientists create a prediction using a regression?

### EXAMPLE: YEARLY FUEL COST AND VEHICLE WEIGHT




| weight | total_gas_expense | predicted_value |
|---|---|---|
| 3000 | NA | 1835.353 |
| 4500 | NA | 2641.988 |
| 2350 | NA | 1485.811 |

**YFC = 220.08 + 0.54*Weight**

## Why do we need an alternative predictive model?



## Can we use this approach for predicting qualified leads?

### PREDICTION APPROACHES

**Yearly fuel costs and weight**

- After running regression we found that this formula describes the data

**YFC = 220.08 + 0.54*Weight**

- Can now predict YFC for any weight

**Sales and lead characteristics**

**Sale = A + B * (# webpages/blogs visited)**

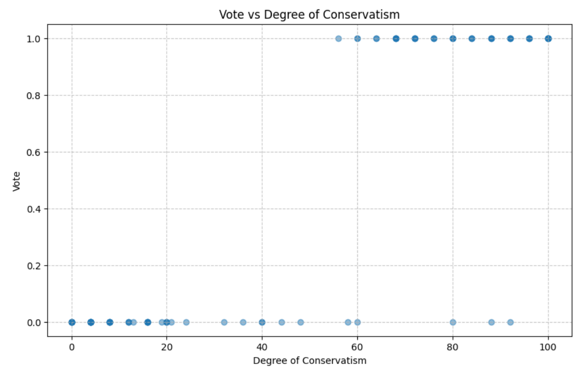
- What kind of variable is “sale”?
- How do we interpret "predicted sale"?

## Let's consider the 1999 Clinton Senate impeachment vote as an example

### PREDICTING THE SENATE IMPEACHMENT VOTE

- **Dependent** variable: **vote1** -- *"guilty"* (1) or *"not guilty"* (0)
- **Predictor** variable: degree of ideological conservatism ("conservatism")
  - 0-100 scale, 100 is most conservative
  - Issued by the "American Conservative Union" (http://conservative.org/)
  - Based on Senator voting records

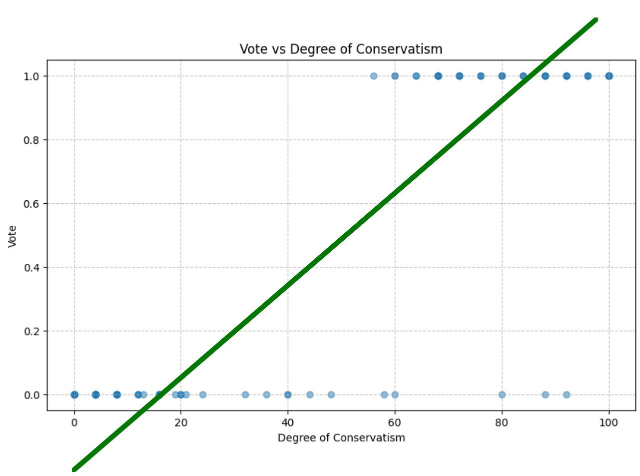
| vote1 | Frequency | Average conservatism |
|---|---|---|
| not guilty | 55 | 18 |
| guilty | 45 | 83 |






## How would we use a regression to predict?

### REGRESSION FOR IMPEACHMENT VOTE



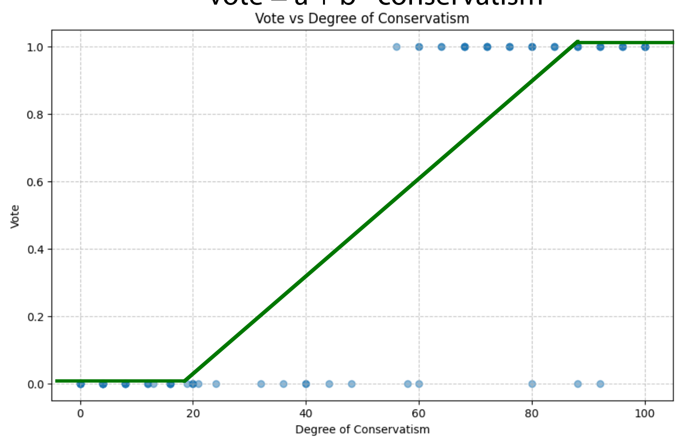

vote = A + B • conservatism

Interpretation?

## The regression approach has several problems

### REGRESSION FOR IMPEACHMENT VOTE

vote = a + b • conservatism




**Problems:**

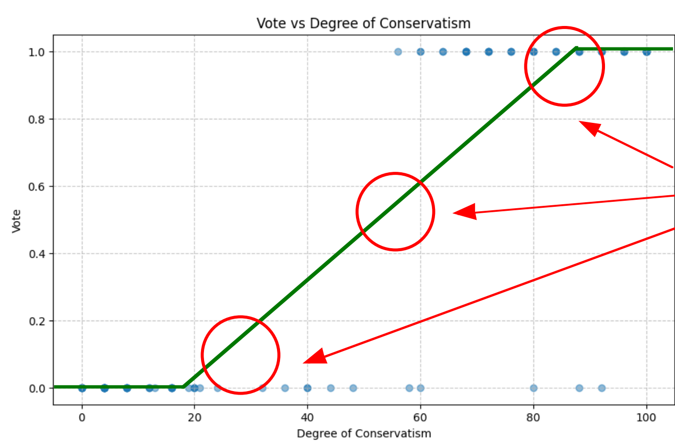
1. “out-of-range” predictions
2. What else?



## The regression approach has several problems

### REGRESSION FOR IMPEACHMENT VOTE

vote = a + b • conservatism

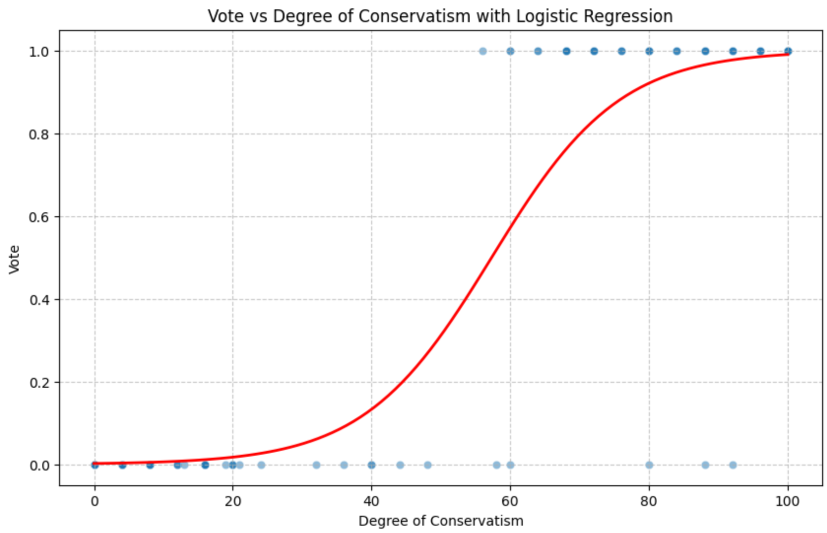



**Problems:**

1. “out-of-range” predictions
2. Linear effect on probabilities

# We would like a method that corrects the shortcomings of regression

## "IDEAL" PROBABILITY PREDICTION






# Logistic Model

## Logit is a flexible way to predict binary choices

### PROPERTIES OF LOGISTIC REGRESSION

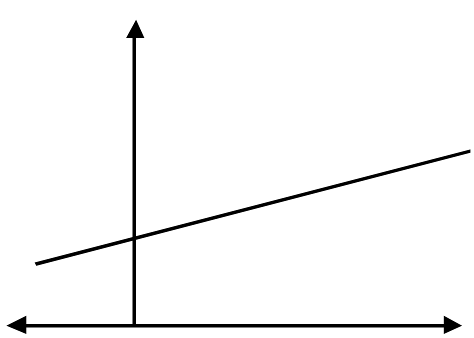
- Also known as **"Logistic regression"**
  (Daniel McFadden, Berkeley Econ Nobel Laureate 2000)
- Used when the dependent variable is binary
  - Buy / do not buy (purchase choice models)
  - Left / stayed (attrition, churn models)
  - Failed / did not fail (predictive maintenance)
- From data scientists view, works similar to regular regression
  - Can include many different variables
  - Fast
  - One of the most popular approaches used in data science



## The logit model allows us to easily estimate probabilities

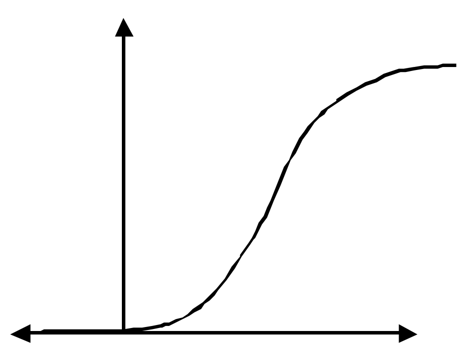
### COMPARISON OF REGRESSION APPROACHES

**OLS Regression**



$$Y = a + b\,X$$

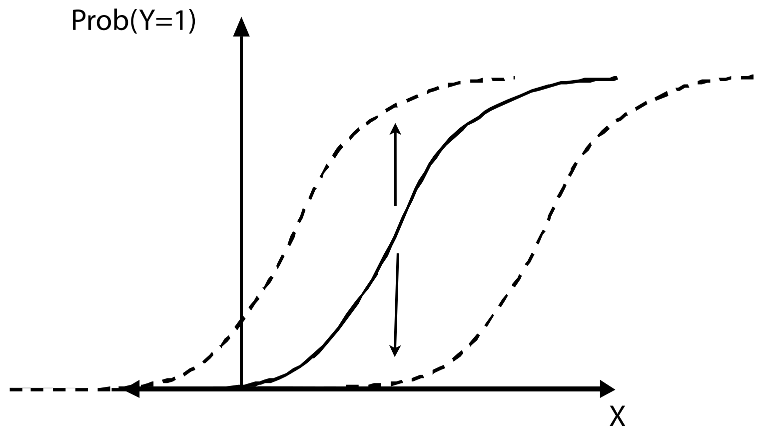
**Logistic Regression**



$$\text{Prob}(Y=1) = \frac{e^{a+bX}}{1+e^{a+bX}}$$

## The “a” coefficient shifts the probability curve

**EFFECT OF “a” COEFFICIENT**

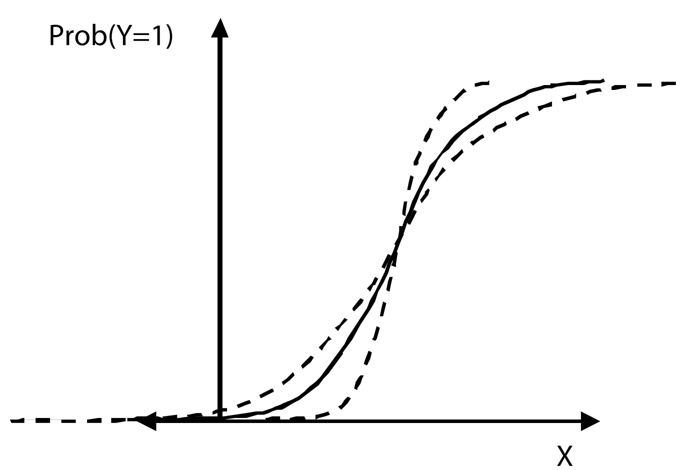



Interpretation?



## The “b” coefficient controls the steepness of the probability curve

**EFFECT OF “b” COEFFICIENT**




Interpretation?

# A comparison of OLS and logistic regression shows that each has advantages and disadvantages

### OLS REGRESSION MODEL

$$Y = a + b\,X$$

- Depending on the values of the independent variables, the predicted values for Y may fall outside of [0,1] (–)
- Changes in X have a linear effect on estimated "probabilities" (–)
- Coefficients are easy to interpret: Measure the amount the dependent variable will increase when the independent variable is increased by one (+)

### LOGIT MODEL

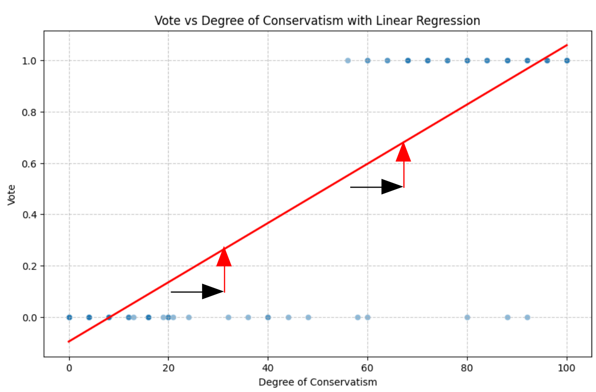
$$\text{Prob}(Y = 1) = \frac{e^{a+bX}}{1 + e^{a+bX}}$$

- The predicted values always fall in [0,1] (+)
- Changes in X can have different effect on probabilities for different levels of X (+)
- How do we interpret the coefficients? (–)

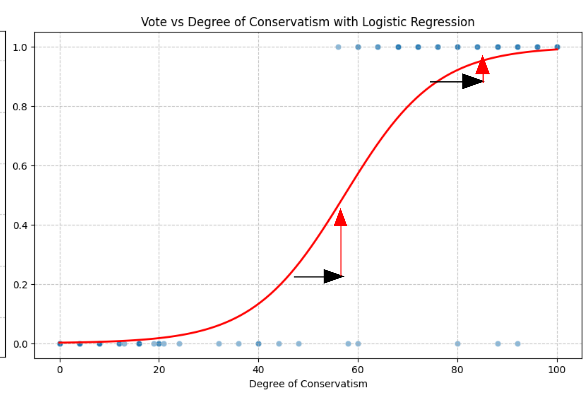


# Logit coefficients are harder to interpret than “normal” regression coefficients

### OLS REGRESSION




### LOGISTIC REGRESSION




- An increase of 1 in the degree of conservatism does not have a constant effect on the dependent variable (predicted probability of voting guilty)
- How do we measure the degree of association between the independent and dependent variable?

# In R we use the "glm" command to run a logistic regression

## PREDICTING THE SENATE IMPEACHMENT VOTE

- Dependent variable: "vote" (guilty (1) or not guilty (0))
- Independent variable: "conservatism" (0-100 scale, 100 is most conservative)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Load the data from url
data = pd.read_csv("https://songyao21.github.io/course_data/impeach.csv")
logit_model = smf.logit(formula='vote1 ~ conserv', data=data).fit()
```

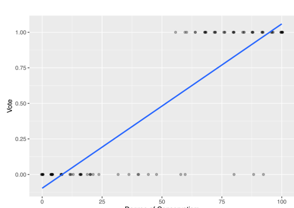
$$\text{Prob}(Y=1) = \frac{e^{a+bX}}{1+e^{a+bX}}$$

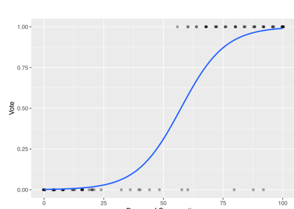
```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -6.2067      1.567     -3.962      0.000      -9.277      -3.136
conserv        0.1083      0.024      4.543      0.000       0.062       0.155
==============================================================================
```



# Interpreting the impact of "conservatism" is tricky

**Impact of "conservatism" on voting probability is *not stable***




| Conservatism | Pr(vote guilty) | Change in Prob |
|---|---|---|
| 1 | 0.002 | |
| 2 | 0.002 | 0.0003 |
| 3 | 0.003 | 0.0003 |
| 4 | 0.003 | 0.0003 |
| 5 | 0.003 | 0.0004 |
| | | |
| 42 | 0.160 | 0.0140 |
| 43 | 0.175 | 0.0151 |
| 44 | 0.191 | 0.0162 |
| 45 | 0.208 | 0.0173 |
| 46 | 0.227 | 0.0184 |
| | | |
| 54 | 0.411 | 0.0259 |
| 55 | 0.437 | 0.0264 |
| 56 | 0.464 | 0.0268 |
| 57 | 0.491 | 0.0270 |
| 58 | 0.518 | 0.0271 |
| | | |
| 96 | 0.985 | 0.0017 |
| 97 | 0.987 | 0.0015 |
| 98 | 0.988 | 0.0014 |
| 99 | 0.989 | 0.0012 |
| 100 | 0.990 | 0.0011 |

- **The size of the impact depends on the specific value of conservatism**

## We can make a more general statement about the "*Average Marginal Effect*" of the predictors

```
# Compute the average marginal effect
marginal_effects = logit_model.get_margeff(at='overall', method='dydx')

# Print the summary of marginal effects
print(marginal_effects.summary())
```

```
        Logit Marginal Effects
=====================================
Dep. Variable:                  vote1
Method:                          dydx
At:                           overall
==============================================================================
                dy/dx    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
conserv        0.0059      0.001      8.828      0.000       0.005       0.007
==============================================================================
```



# Applying Logistic Model

# We can look at the predicted probabilities at different levels of the predictor in the data, the conservatism score

```python
# Generate 5 random conservatism scores
np.random.seed(42)  # for reproducibility
random_conserv = np.round(np.random.uniform(low=0, high=100, size=5))

# Create a DataFrame with these scores
random_df = pd.DataFrame({'conserv': random_conserv})

# Add a constant term to the random conservatism scores
random_X = sm.add_constant(random_df['conserv'])

# Predict probabilities for these random scores
predicted_probs = logit_model.predict(random_X)

# Combine the results into a DataFrame
random_results = pd.DataFrame({
    'Hypothetical Conservatism Score': random_conserv,
    'Predicted Probability': predicted_probs
})

# Display the results
print("Predictions for 5 random conservatism scores:")
print(random_results.to_string(index=False, float_format='{:.4f}'.format))
```

```
Predictions for 5 random conservatism scores:
 Hypothetical Conservatism Score  Predicted Probability
                         37.0000                 0.0997
                         95.0000                 0.9833
                         73.0000                 0.8451
                         60.0000                 0.5717
                         16.0000                 0.0113
```



# The prediction can easily be made in Excel

## HOW TO PREDICT WITH A LOGIT

$$\text{Prob(vote)} = \frac{e^{-6.2+0.11\cdot\text{conservatism}}}{1+e^{-6.2+0.11\cdot\text{conservatism}}}$$

From a statistical program:

R, Python, SAS, Stata, ...

| | A | B | C |
|---|---|---|---|
| 1 | Coefficients | | |
| 2 | A | -6.206741 | |
| 3 | B | 0.1082609 | |
| 4 | | | |
| 5 | Conservatism | 45 | |
| 6 | | | |
| 7 | Logit Formula | =EXP(B2+B3*B5)/(1+EXP(B2+B3*B5)) | |
| 8 | (evaluated) | **21%** | |

Prediction

# Back to predictive lead scoring …

| id | webpageviews | viewedpricing | highrelevancefirm | sale |
|---|---|---|---|---|
| 639 | 15 | 0 | 1 | 0 |
| 272 | 35 | 0 | 1 | 1 |
| 491 | 7 | 0 | 1 | 0 |
| 226 | 18 | 1 | 1 | 0 |
| 7195 | 13 | 0 | 0 | 0 |
| 9080 | 23 | 0 | 0 | 0 |
| 548 | 14 | 0 | 1 | 0 |
| 9605 | 36 | 0 | 0 | 0 |
| 5352 | 28 | 0 | 0 | 0 |
| 4343 | 35 | 0 | 0 | 0 |
| 14971 | 3 | 0 | 0 | 0 |
| 11298 | 34 | 0 | 0 | 1 |
| 317 | 10 | 0 | 1 | 0 |
| … | … | … | … | … |

| id | webpageviews | viewedpricing | highrelevancefirm | sale |
|---|---|---|---|---|
| 7236 | 27 | 0 | 0 | ? |
| 687 | 25 | 1 | 1 | |
| 453 | 16 | 0 | 0 | |
| 563 | 6 | 0 | 1 | |



# Estimation

## LEAD SCORING EXAMPLE

```
# Read the new data from the provided URL
lead_url = "https://songyao21.github.io/course_data/Lead_SmartStorage.csv"
lead = pd.read_csv(lead_url)
```

```
# Fit the logistic regression model
logit_model_lead = smf.logit(formula='sale ~ webpageviews + viewedpricing + highrelevancefirm',
                             data=lead).fit()

# Print the summary of the model
print(logit_model_lead.summary())

# Calculate the average marginal effect for 'webpageviews'
marginal_effects = logit_model_lead.get_margeff(at='overall', method='dydx')
print(marginal_effects.summary())
```

```
================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const               -10.8631      2.268     -4.790      0.000     -15.308      -6.418
webpageviews          0.2423      0.064      3.801      0.000       0.117       0.367
viewedpricing         2.3606      0.941      2.510      0.012       0.517       4.204
highrelevancefirm     2.8682      0.873      3.285      0.001       1.157       4.579
================================================================================
                       dy/dx    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
webpageviews          0.0121      0.003      4.574      0.000       0.007       0.017
viewedpricing         0.1179      0.043      2.736      0.006       0.033       0.202
highrelevancefirm     0.1433      0.039      3.705      0.000       0.067       0.219
================================================================================
```

# Prediction

## We can then predict purchase prob (often called "score" in practice).

```
# Predict sale probabilities for all observations
lead['predicted_sale_prob'] = logit_model_lead.predict(lead)

# Create a sample of 10 observations with observed sale and 10 with missing sale
sample_observed = lead[lead['sale'].notnull()].sample(5, random_state=42)
sample_missing = lead[lead['sale'].isnull()].sample(5, random_state=42)
```

| webpageviews | viewedpricing | highrelevancefirm | sale | predicted_sale_prob |
|---|---|---|---|---|
| 22 | 0 | 0 | 0.0 | 0.003941 |
| 31 | 0 | 0 | 0.0 | 0.033847 |
| 2 | 0 | 1 | 0.0 | 0.000547 |
| 18 | 0 | 1 | 0.0 | 0.025748 |
| 23 | 1 | 0 | 0.0 | 0.050720 |
| 27 | 0 | 0 | NaN | 0.013116 |
| 26 | 0 | 1 | NaN | 0.155138 |
| 33 | 0 | 0 | NaN | 0.053817 |
| 25 | 0 | 1 | NaN | 0.125959 |
| 6 | 0 | 1 | NaN | 0.001441 |



# Interpreting the impacts of variables: Scaling

## LEAD SCORING EXAMPLE

| | dy/dx | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| webpageviews | 0.0121 | 0.003 | 4.574 | 0.000 | 0.007 | 0.017 |
| viewedpricing | 0.1179 | 0.043 | 2.736 | 0.006 | 0.033 | 0.202 |
| highrelevancefirm | 0.1433 | 0.039 | 3.705 | 0.000 | 0.067 | 0.219 |

```
# Calculate the standard deviation of webpageviews
print("Standard deviation of webpageviews:")
print(lead.webpageviews.std())
```

```
Standard deviation of webpageviews:
9.001820028144058
```

=0.01211 * 9.00182
=0.109

# Interpreting the impacts of variables: Scaling

## We can also scale the data in advance before the regression

```
# Prepare the data for logistic regression
lead['webpageviews_scaled'] = lead['webpageviews'] / lead.webpageviews.std()

# Fit the logistic regression model
logit_model_lead_prescaled = \
    smf.logit(formula='sale ~ webpageviews_scaled + viewedpricing + highrelevancefirm',
              data=lead).fit()

# Calculate the average marginal effect for 'webpageviews'
marginal_effects_prescaled = logit_model_lead_prescaled.get_margeff(at='overall', method='dydx')
```

| | dy/dx | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| webpageviews | 0.1090 | 0.024 | 4.574 | 0.000 | 0.062 | 0.156 |
| viewedpricing | 0.1179 | 0.043 | 2.736 | 0.006 | 0.033 | 0.202 |
| highrelevancefirm | 0.1433 | 0.039 | 3.705 | 0.000 | 0.067 | 0.219 |