Logistic Regressions

Modeling **probabilities**

Charles Parker
VP ML Algorithms, BigML, Inc

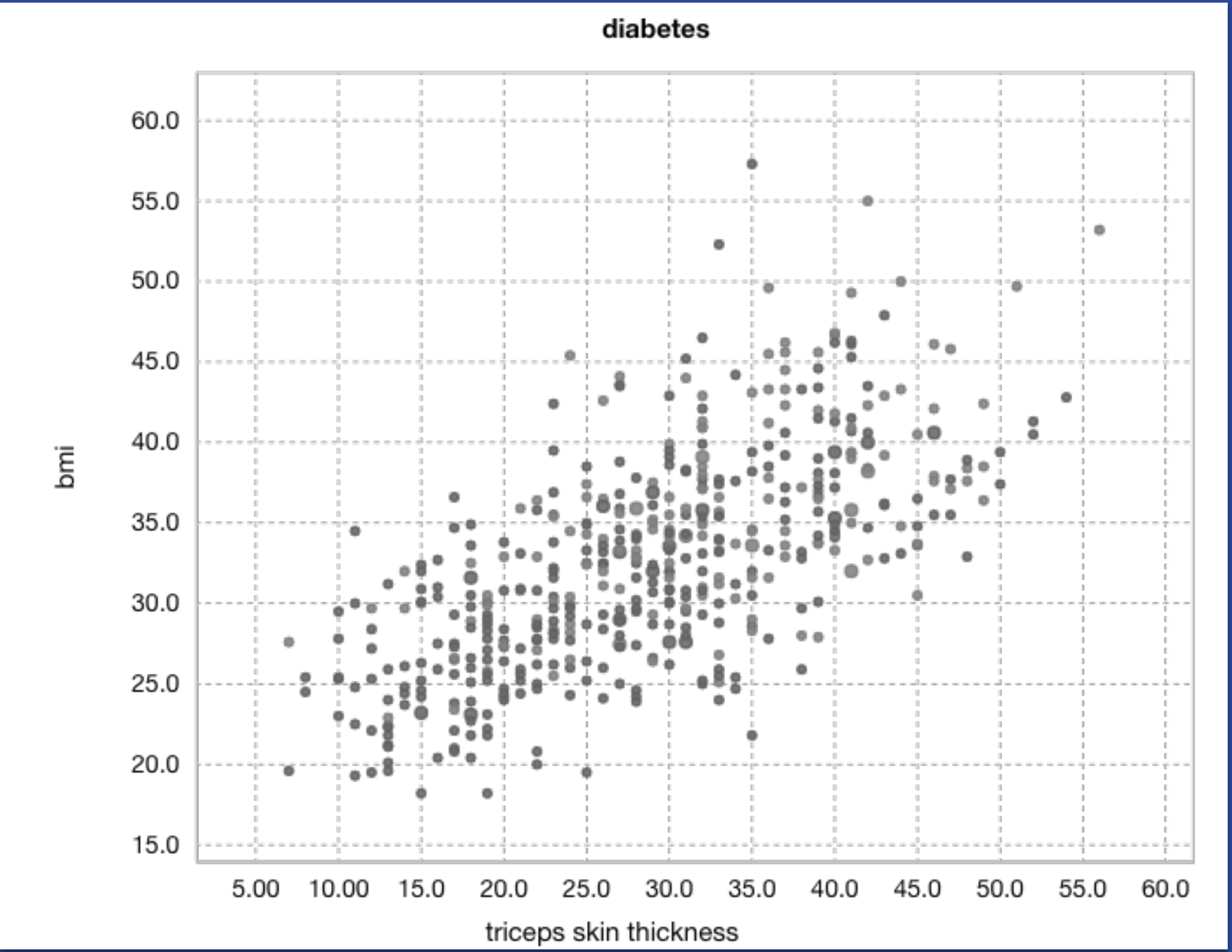
Logistic Regression

Potential Confusion:

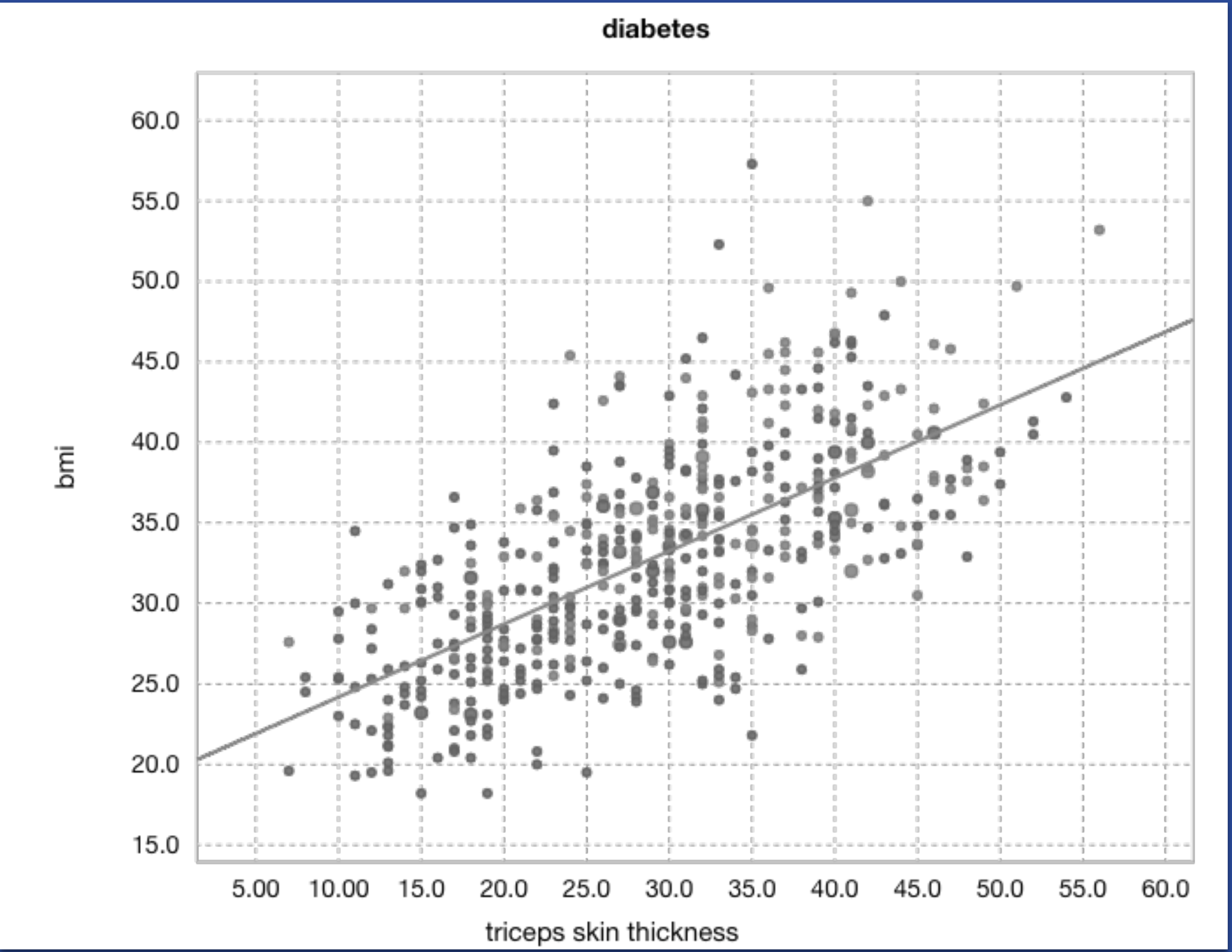
Logistic Regression is a classification algorithm

- Classification implies a discrete objective. How can this be a regression?
- Why do we need another classification algorithm?
- more questions….

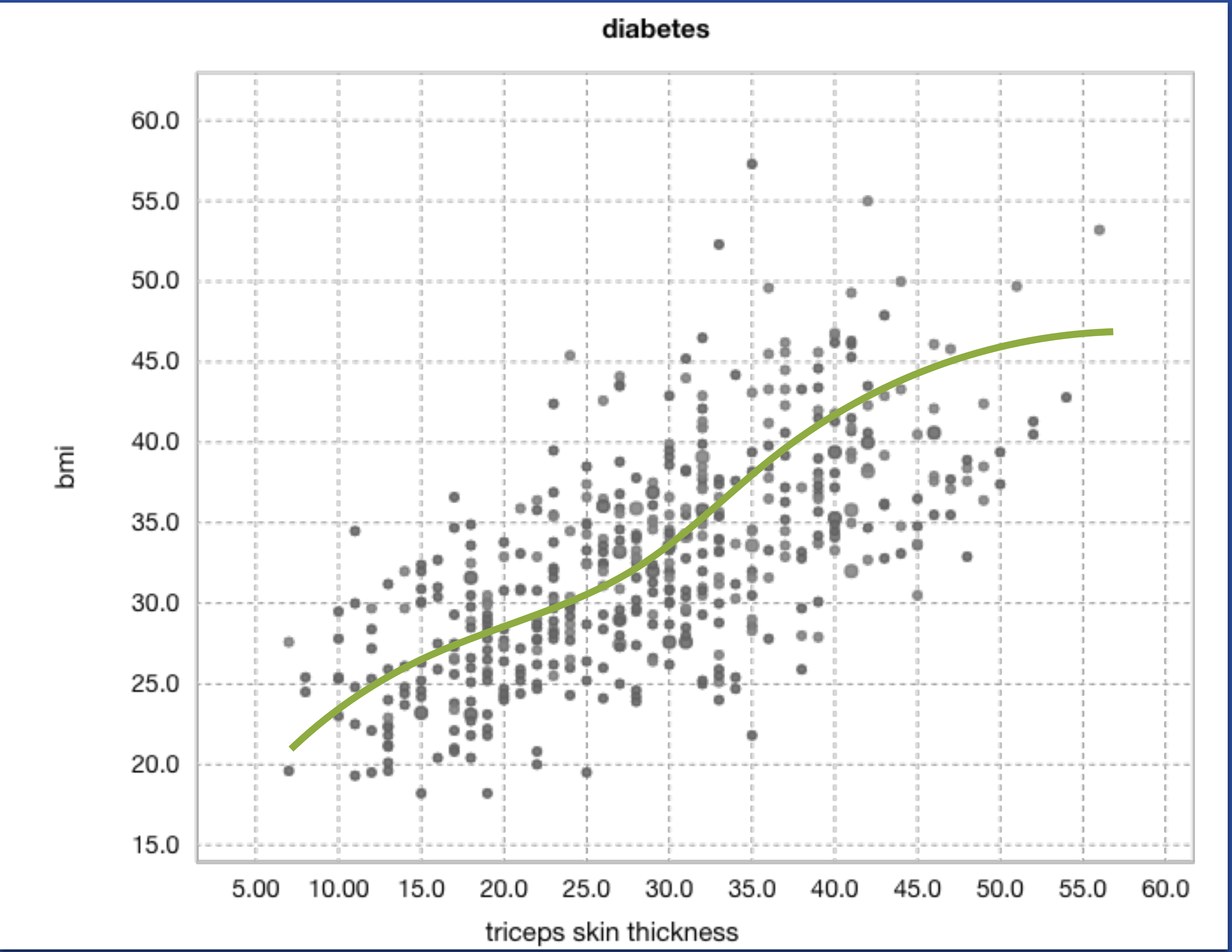
Linear Regression



Linear Regression



Polynomial Regression



Regression

Key Take-Away:

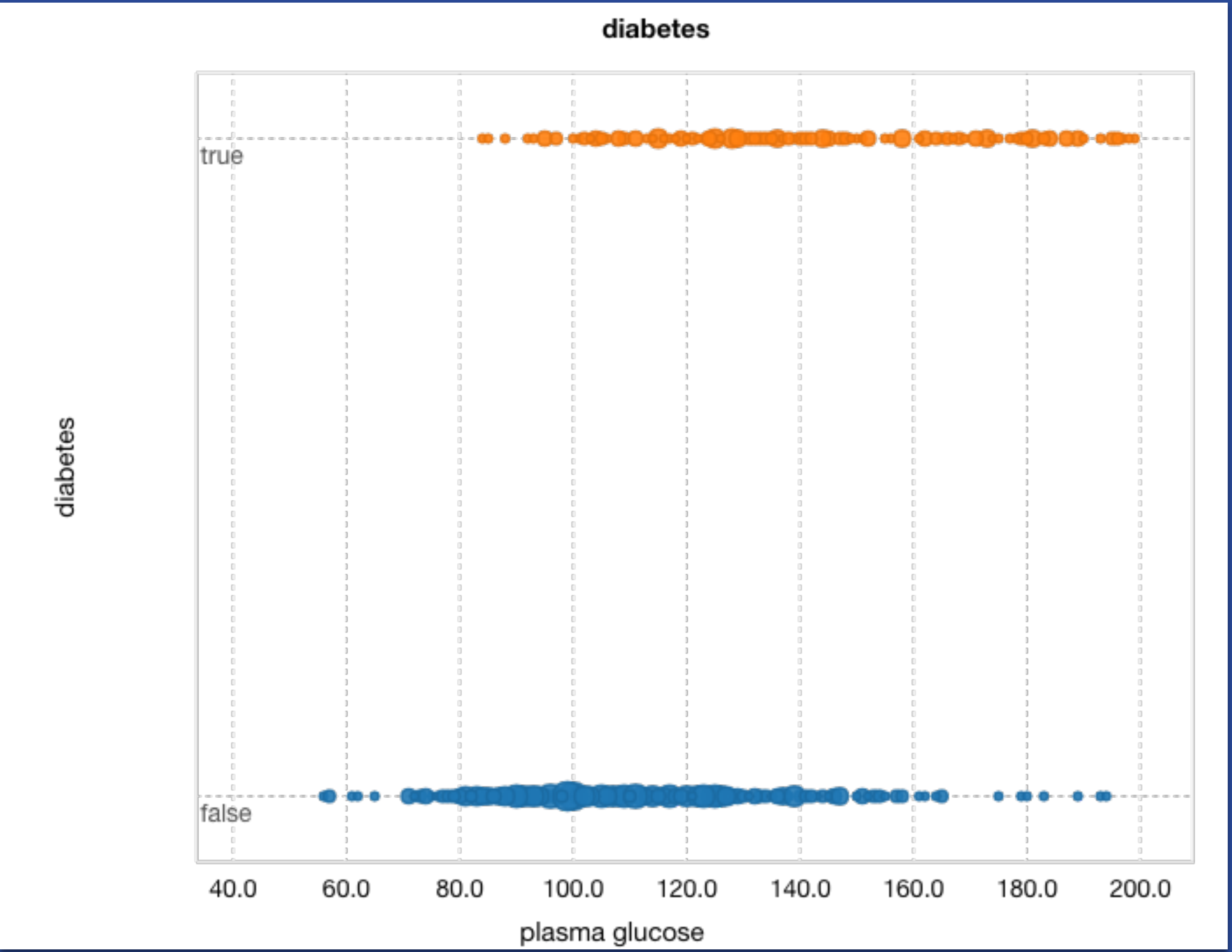
Regression is the process of "fitting" a function to the data

- Linear Regression: $\beta_0 + \beta_1 \cdot (\text{INPUT}) \approx \text{OBJECTIVE}$
- Quadratic Regression: $\beta_0 + \beta_1 \cdot (\text{INPUT}) + \beta_2 \cdot (\text{INPUT})^2 \approx \text{OBJECTIVE}$
- Decision Tree Regression: $\text{DT}(\text{INPUT}) \approx \text{OBJECTIVE}$

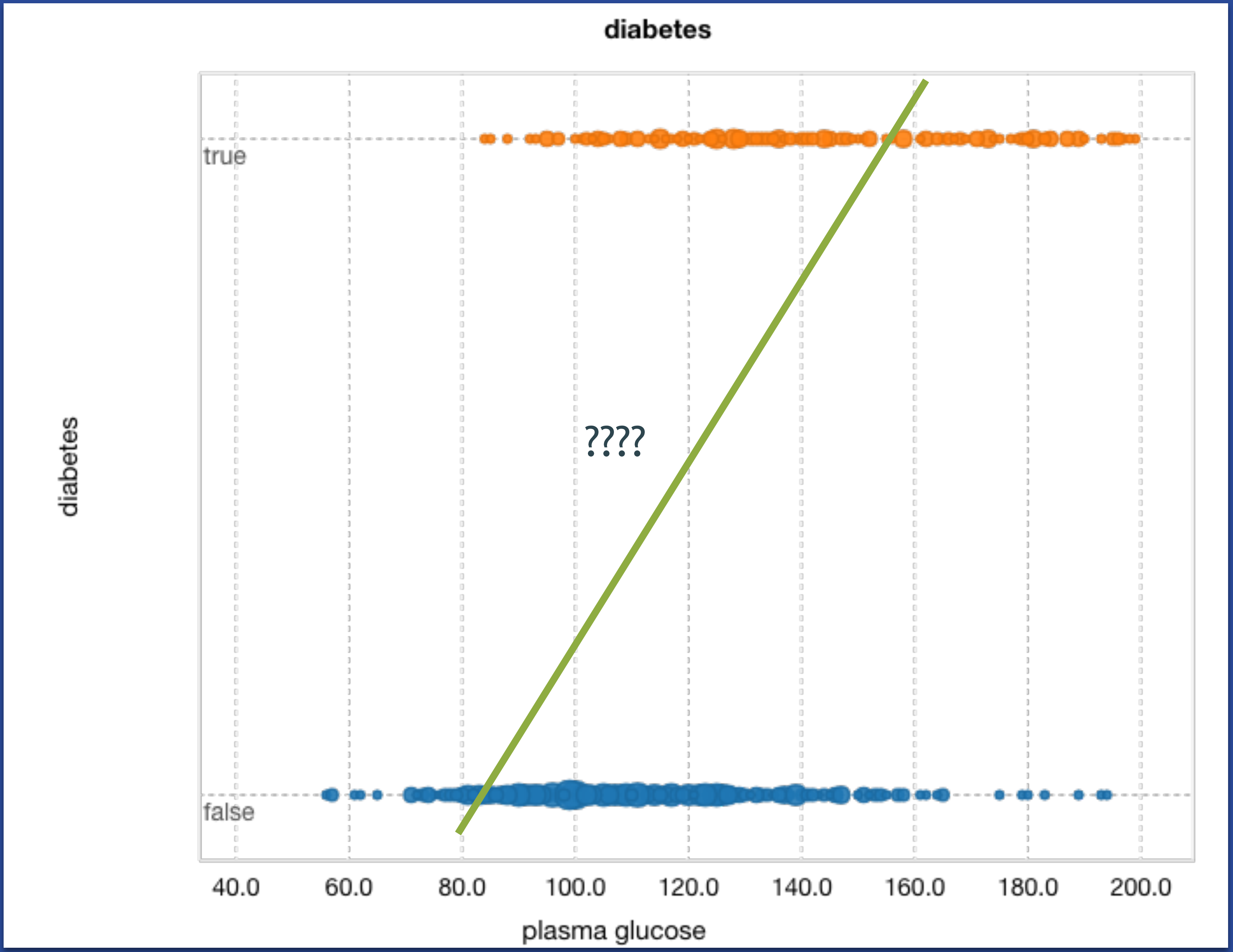
New Problem:

- What if we want to do a classification problem: **T/F** or **1/0**
- What function can we fit to **discrete data**?

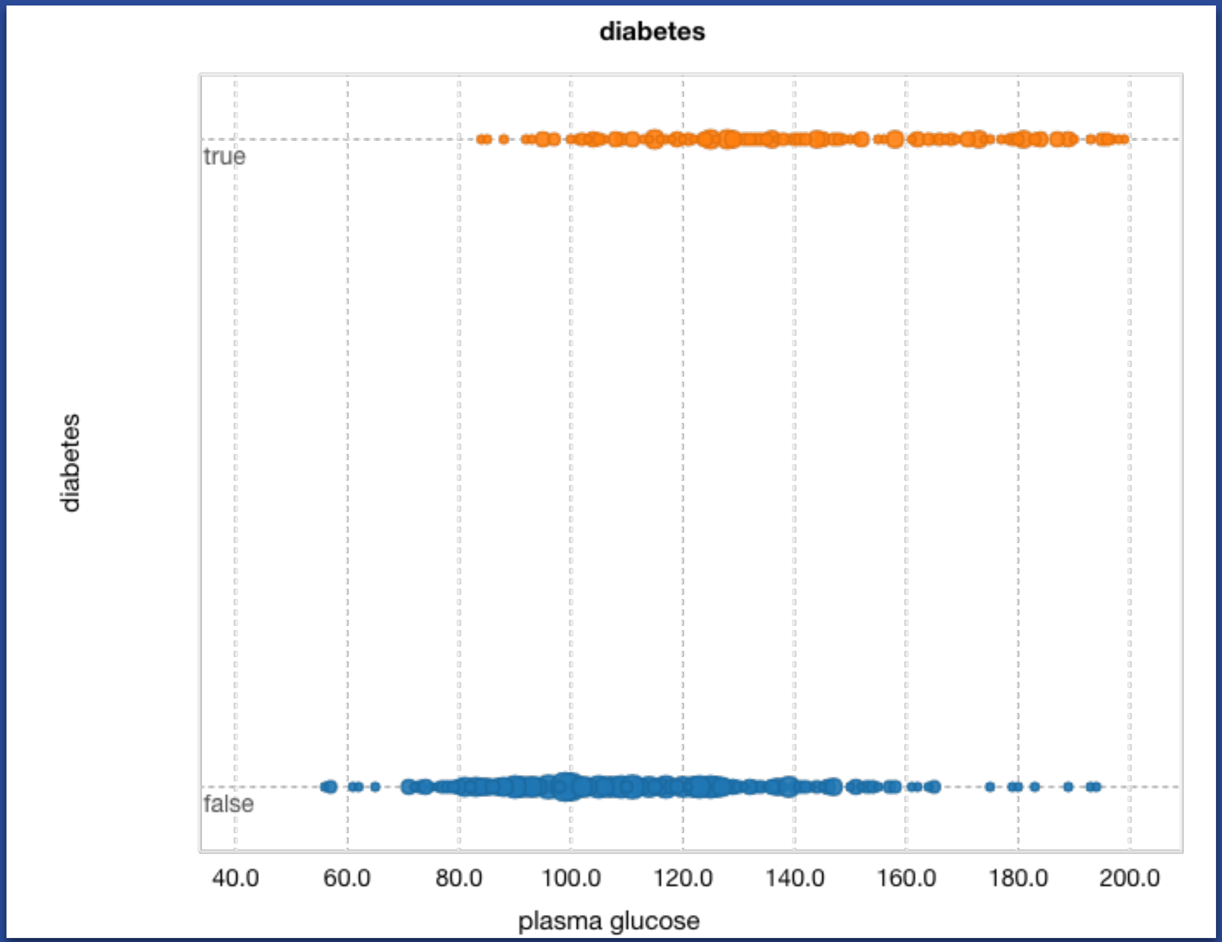
Discrete Data Function?

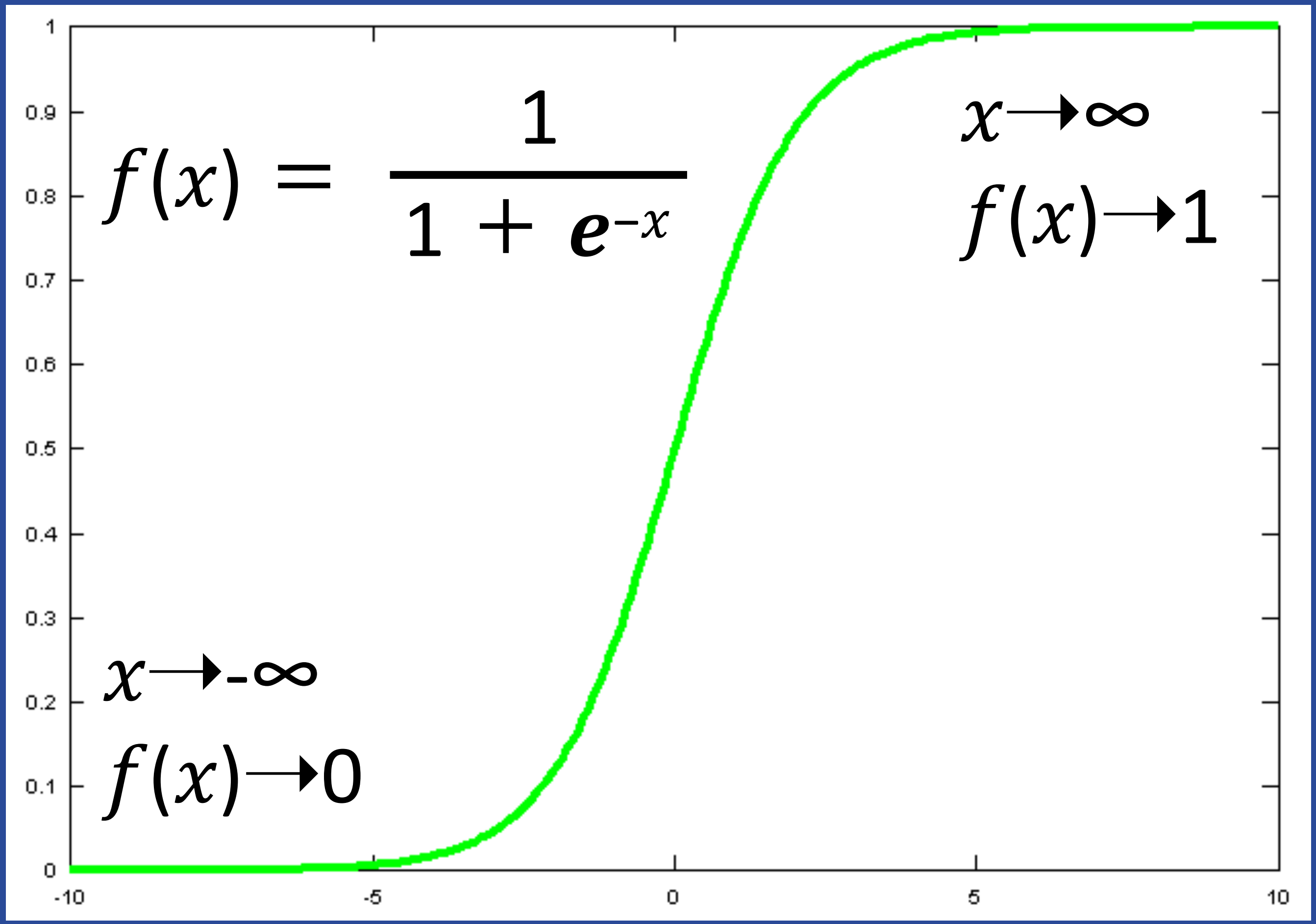


Discrete Data Function?



Logistic Function



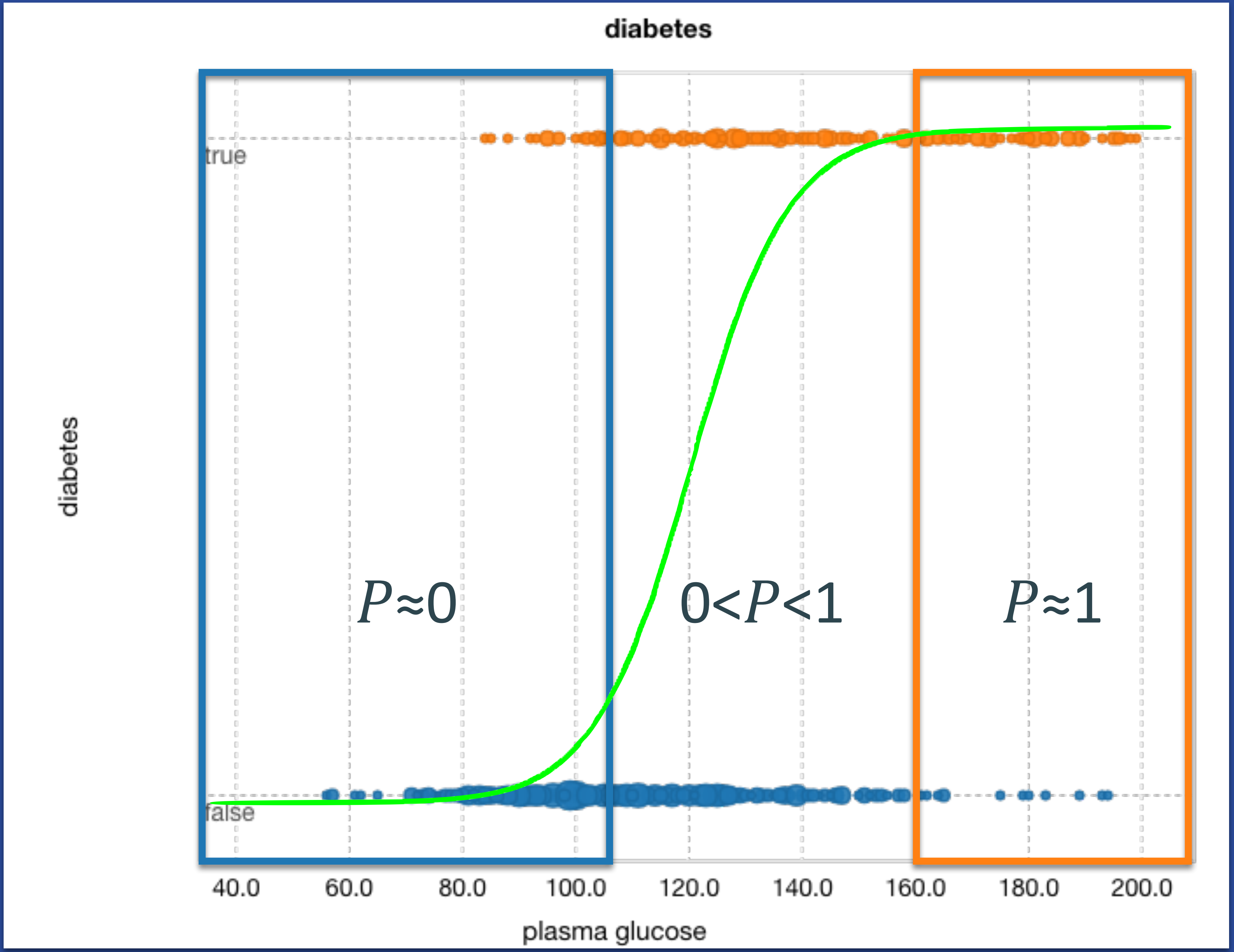


Goal

Logistic Function

- Looks promising, but still not "discrete"
- What about the "green" in the middle?
- Let's change the problem…

Modeling Probabilities



Logistic Regression

Clarification:

*LR is a **classification** algorithm … that uses a **regression** … to **model the probability** of the discrete objective*

Caveats:

- Assumes that output is linearly related to "predictors"
  - What? (hang in there…)
  - Sometimes we can "fix" this with feature engineering
- Question: how do we "fit" the logistic function to real data?

Logistic Regression

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

β_0 is the "intercept"

β_1 is the "coefficient"

- Given training data consisting of inputs x, and probabilities P
- Solve for β_0 and β_1 to fit the logistic function
- How? The inverse of the logistic function is called the "logit":

$$ln\left(\frac{P(x)}{P(x')}\right) = ln\left(\frac{P(x)}{1 - P(x')}\right) = \beta_0 + \beta_1 x$$

- In which case solving is now a linear regression
- But this is only **one dimension**, that is one feature x…

Logistic Regression

For "i" dimensions, $\boldsymbol{X} = [\, x_1, x_2, \cdots, x_i \,]$, we solve

$$P(\boldsymbol{X}) = \frac{1}{1 + e^{-f(\boldsymbol{X})}}$$

where:

$$f(\boldsymbol{X}) = \beta_0 + \boldsymbol{\beta} \cdot \boldsymbol{X} = \beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i$$

Interpreting Coefficients

- LR computes β_0 and coefficients β_j for each feature x_j
  - negative $\beta_j \rightarrow$ negatively correlated: $x_j \uparrow$ then $P(\boldsymbol{X}) \downarrow$
  - positive $\beta_j \rightarrow$ positively correlated: $x_j \uparrow$ then $P(\boldsymbol{X}) \uparrow$
  - "larger" $\beta_j \rightarrow$ more impact: $x_j >$ then $P(\boldsymbol{X}) \gg$
  - "smaller" $\rightarrow$ less impact: $x_j \gg$ then $P(\boldsymbol{X}) >$
- β_j "size" should not be confused with field importance
- Can include a coefficient for "missing" (if enabled)
  - $P(\boldsymbol{X}) = \beta_0 + \cdots + \beta_j x_j + \cdots + \beta_{j+1}[\ x_j \equiv \text{Missing}\ ]$
- Binary Classification (true/false) coefficients are complementary
  - $P(\text{True}) \equiv 1 - P(\text{False})$

LR Demo #1

LR Parameters

Lots of things to get wrong…

1. **Default Numeric**: Replaces missing numeric values
2. **Missing Numeric**: Adds a field for missing numerics
3. **Stats**: Extended statistics, ex: p-value (runs slower)
4. **Bias**: Enables/Disables the intercept term - β_0
    - Don't disable this…
5. **Regularization**: Reduces over-fitting by minimizing β_j
    - **L1**: prefers reducing individual coefficients
    - **L2** (default): prefers reducing all coefficients
6. **Strength "C"**: Higher values reduce regularization
7. **EPS**: The minimum error between steps to stop
    - Larger values stop earlier but quality may be less
8. **Auto-scaling**: Ensures that all features contribute equally
    - Don't change this unless you have a specific reason
9. Field Encodings: Changes the way categorical values are handled.

Curvilinear LR

- Logistic Regression is expecting a linear relationship between the features and the objective
    - Remember - it's a linear regression under the hood
    - This is actually pretty common in natural datasets
    - But non-linear relationships will impact model quality
- This can be addressed by adding non-linear transformations to the **features**
- Knowing which transformations requires
    - domain knowledge
    - experimentation
    - both

Curvilinear LR

Instead of

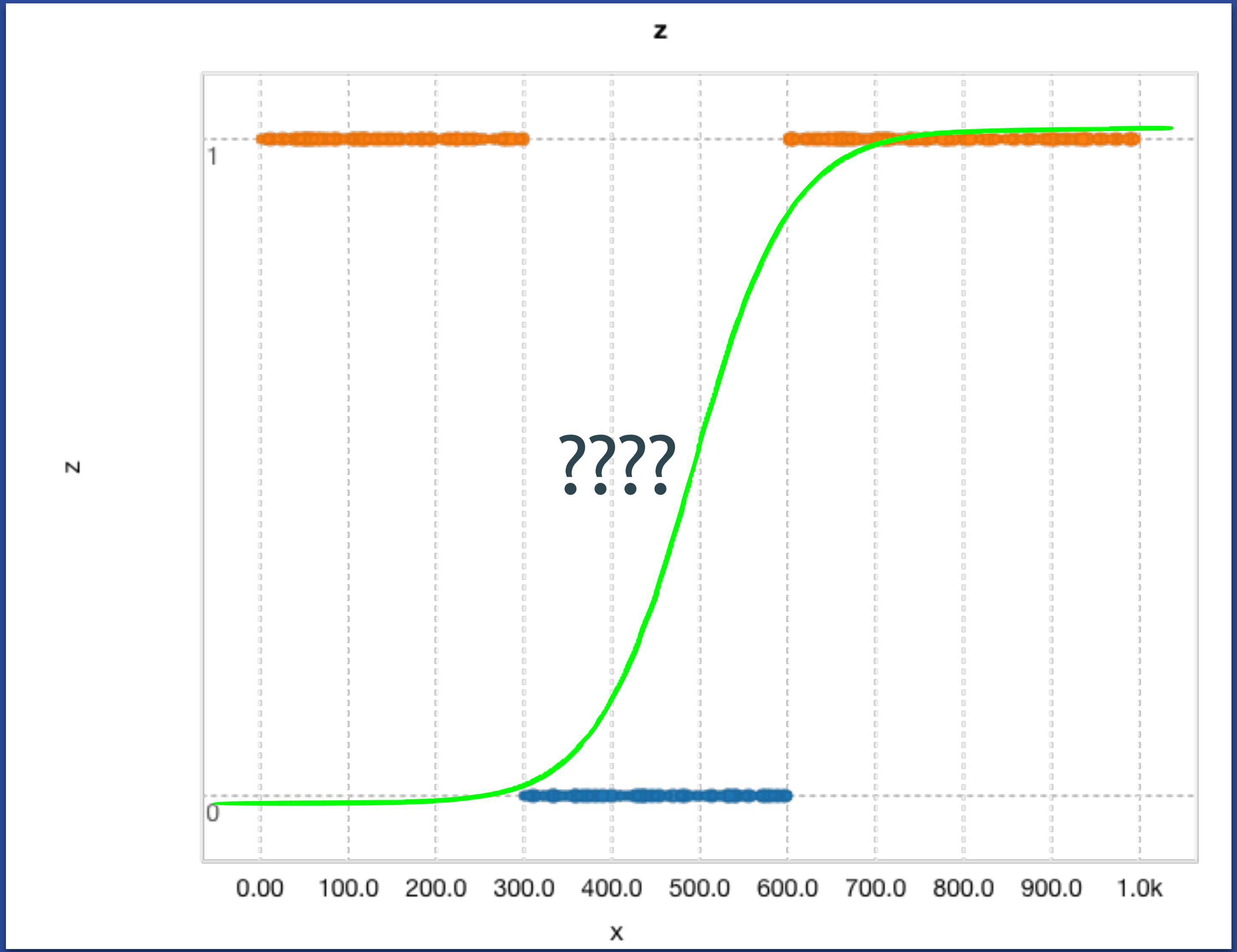
$$\beta_0 + \beta_1 x_1$$

We could add a feature

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Where

$$x_1 \equiv {x_2}^2$$



Possible to add any higher order terms or other functions to match shape of data

LR vs DT

Logistic Regression	Decision Tree
• Expects a "smooth" linear relationship with predictors.	• Adapts well to ragged non-linear relationships
• LR is concerned with probability of a discrete outcome.	• No concern: classification, regression, multi-class all fine.
• Lots of parameters to get wrong: regularization, scaling, codings	• Virtually parameter free
• Slightly less prone to over-fitting	• Slightly more prone to over-fitting
• Because fits a shape, might work better when less data available.	• Prefers surfaces parallel to parameter axes, but given enough data will discover any shape.

Your Turn!

- Build a Logistic Regression with the Diabetes 80% Training set.
- What feature is the best single predictor of Diabetes?
- Evaluate with the 20% Test set.
- Compare to the Model and Ensemble evaluations you did earlier
- Which performs *better*?