(low FN) high recall important: medical tests

(low FP) high precision important: accusing someone.

Activity

Consider the confusion matrix shown below.

	Predicted Negative	Predicted Positive
Actually Negative	TN = 22 ✅	FP = 2 ❌
Actually Positive	FN = 23 ❌	TP = 18 ✅

What is the accuracy of the above classifier? The precision? The recall?

$\frac{18+22}{18+22+23+2} = \frac{40}{65}$

After calculating all three on your own, click below to see the answers.

👉 **Accuracy**

👉 **Precision** $\frac{TP}{TP+FP} = \frac{18}{20}$

👉 **Recall** $\frac{TP}{TP+FN} = \frac{18}{41}$

```
309       Gentoo
Name: species, Length: 249, dtype: object
```

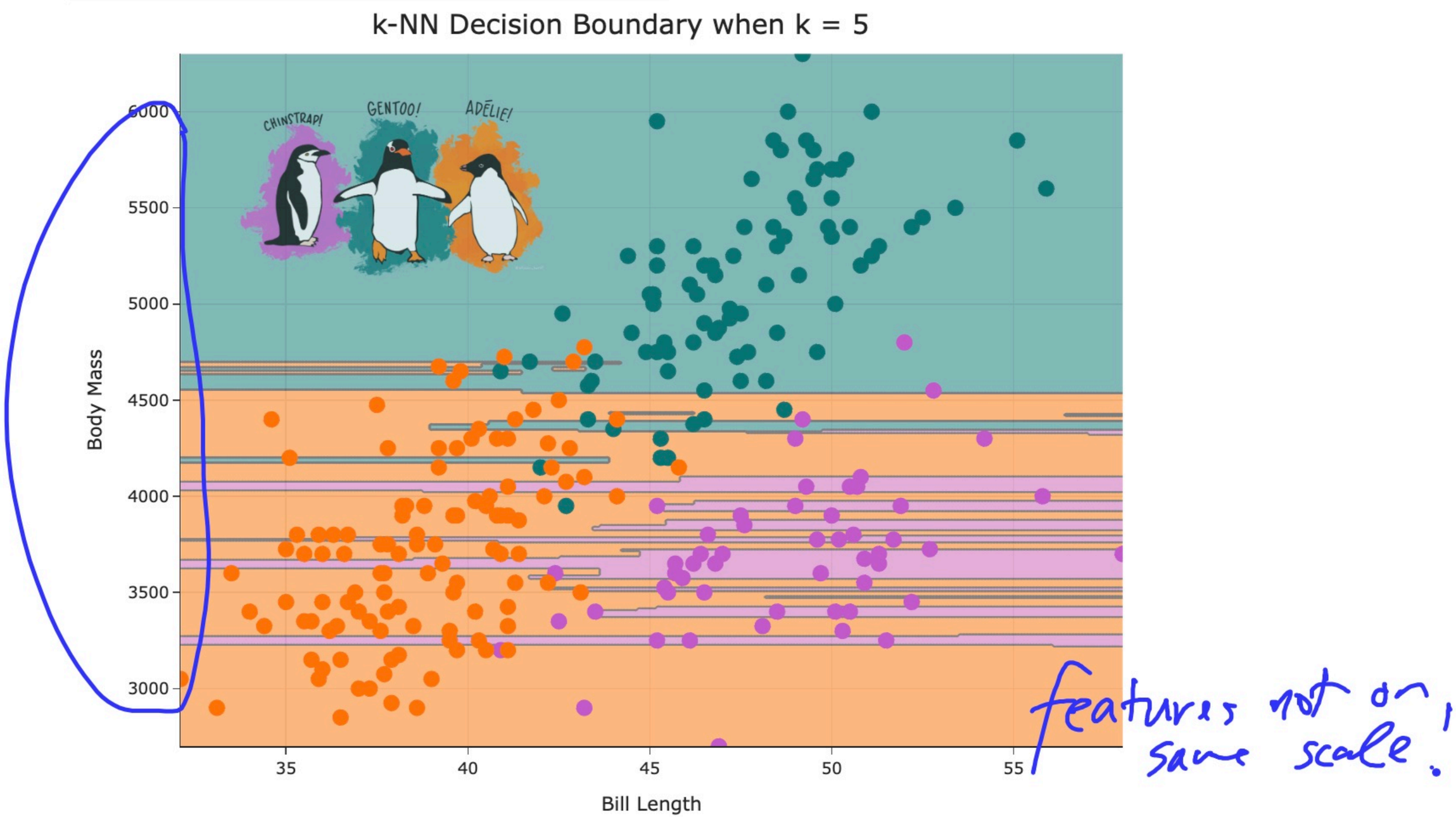
- Here, each row corresponds to a single penguin.

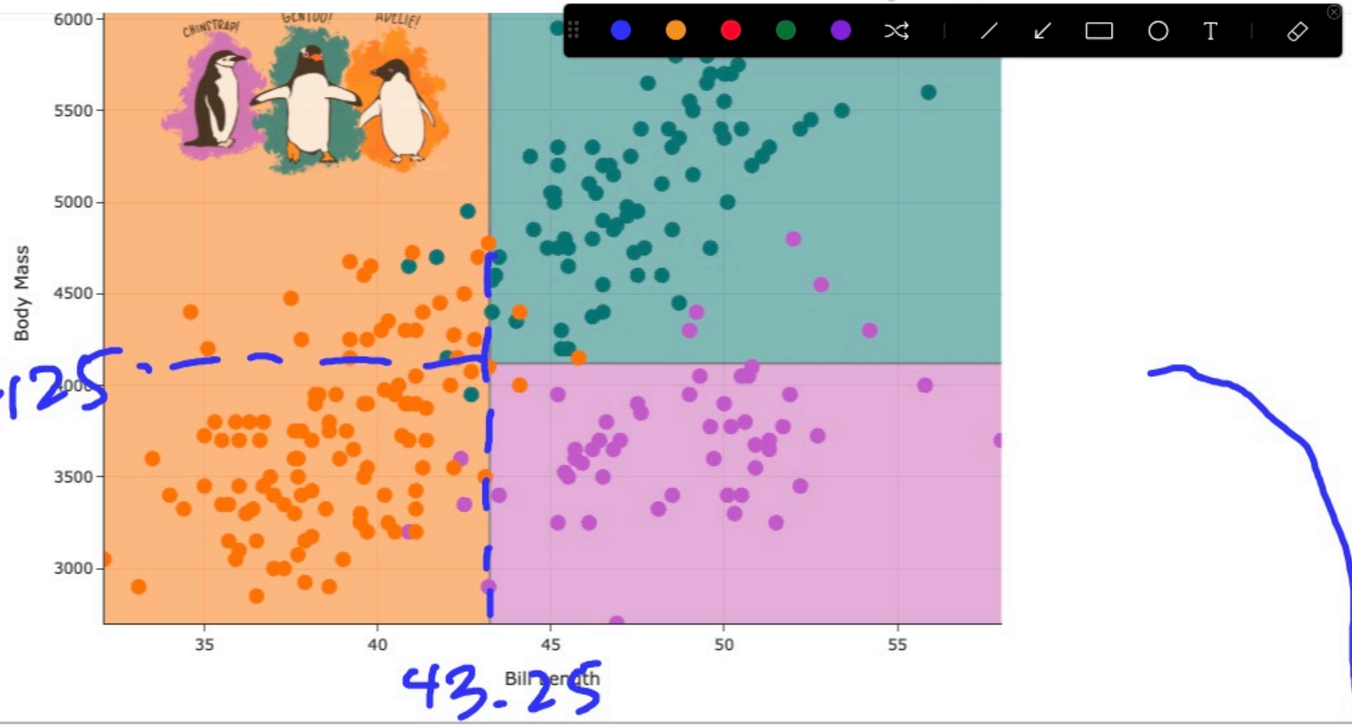
- There are three `'species'` of penguin: Adelie, Chinstrap, an

```
[5]: 1 y_train.value_counts(normalize=True)
```

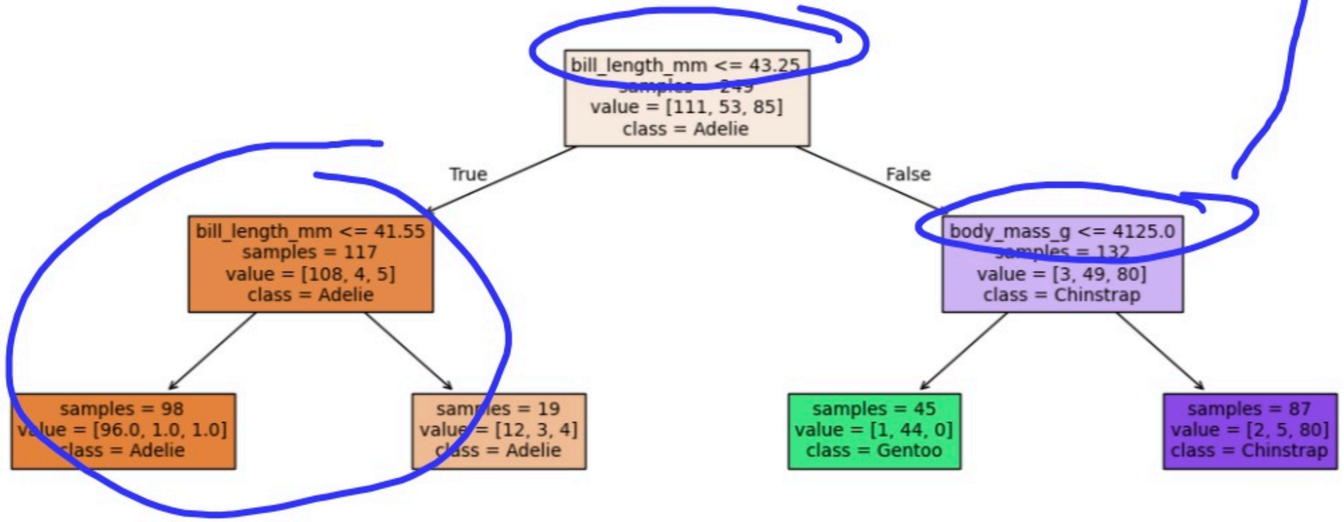
```
t[5]: species
      Adelie       0.45
      Gentoo       0.34
      Chinstrap    0.21
      Name: proportion, dtype: float64
```

constant predictor of "Adelie" gets 45% training accuracy.


k-NN Decision Boundary when k = 5
CHINSTRAP!
GENTOO!
ADÉLIE!
Body Mass
6000
5500
5000
4500
4000
3500
3000
35
40
45
50
55
Bill Length
features not on same scale!



In [16]:
```
1 util.show_penguin_decision_tree(model_tree, X_train);
```

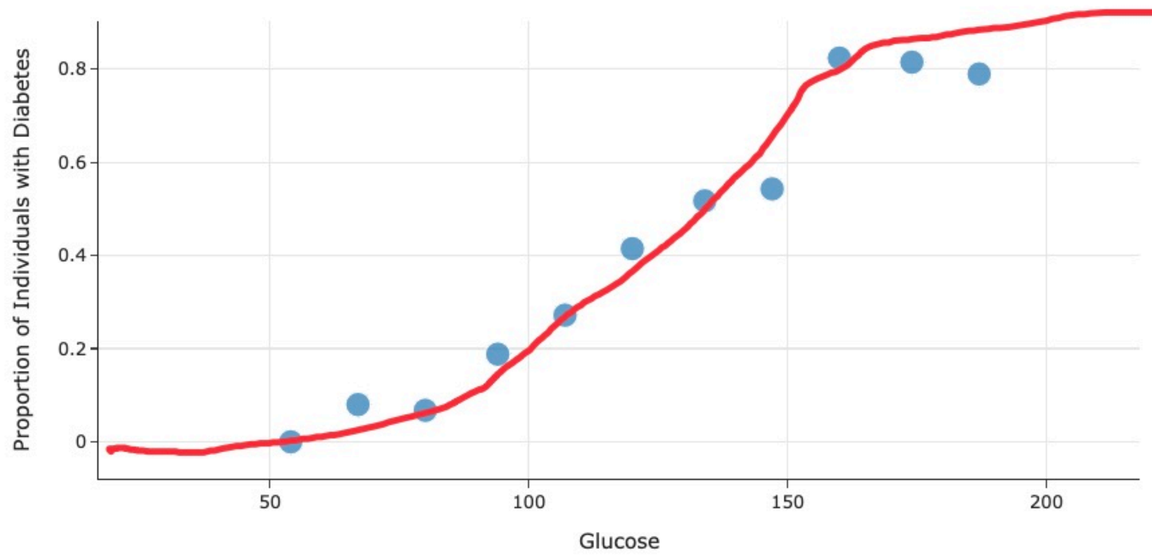


In [3]:
```
1 util.show_one_feature_plot(X_train, y_train)
```

- It seems that as a patient's `'Glucose'` value increases, the **chances they have diabetes** also increases.

- Another approach we could try is to:
  - Place `'Glucose'` values into **bins**, e.g. 50 to 55, 55 to 60, 60 to 65, etc.
  - Within each bin, compute the proportion of patients in the training set who had diabetes.

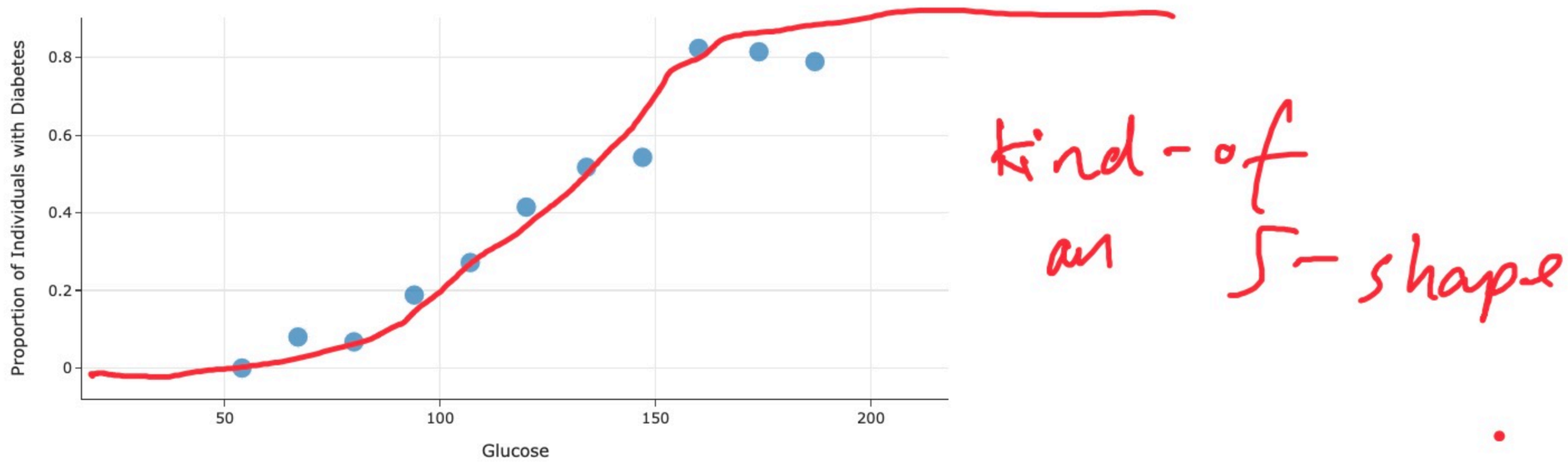
In [6]:
```
# Take a look at the source code in lec22_util.py to see how we did this!
# We've hidden a lot of the plotting code in the notebook to make it cleaner.
util.make_prop_plot(X_train, y_train)
```



kind-of an S-shape.

- Another approach we could try is to:
  - Place `'Glucose'` values into **bins**, e.g. 50 to 55, 55 to 60, 60 to 65, etc.
  - Within each bin, compute the proportion of patients in the training set who had diabetes.

In [6]:
```
# Take a look at the source code in lec22_util.py to see how we did this!
# We've hidden a lot of the plotting code in the notebook to make it cleaner.
util.make_prop_plot(X_train, y_train)
```



The logistic function

- The **logistic function** resembles an S-shape.

$$\sigma(t) = \frac{1}{1 + e^{-t}} = \frac{1}{1 + \exp(-t)}$$

"sigma"

The logistic function is an example of a **sigmoid function**, which is the general term for an S-shaped function. Sometimes, we use the terms "logistic function" and "sigmoid function" interchangeably.

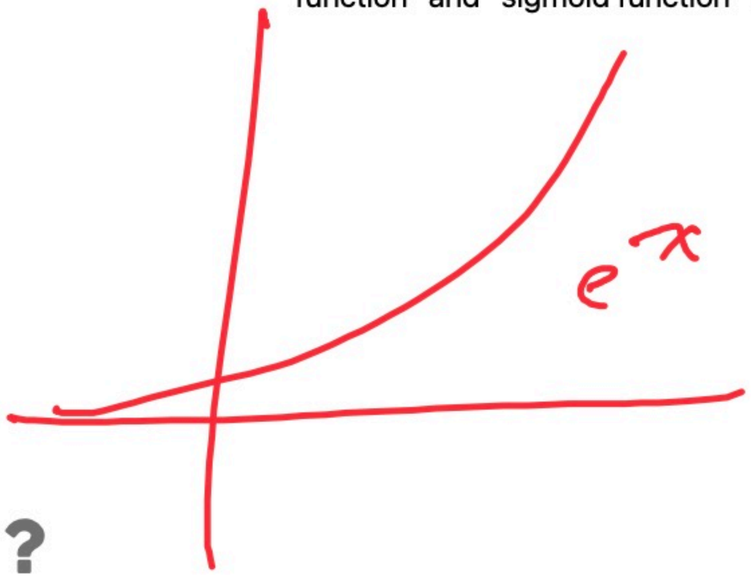
The logistic function

- The **logistic function** resembles an S-shape.

$$\sigma(t) = \frac{1}{1 + e^{-t}} = \frac{1}{1 + \exp(-t)}$$

"sigma"

The logistic function is an example of a **sigmoid function**, which is the general term for an S-shaped function. Sometimes, we use the terms "logistic function" and "sigmoid function" interchangeably.



$t \to -\infty$

$\sigma(t) \to \frac{1}{1+e^{\infty}} = 0$

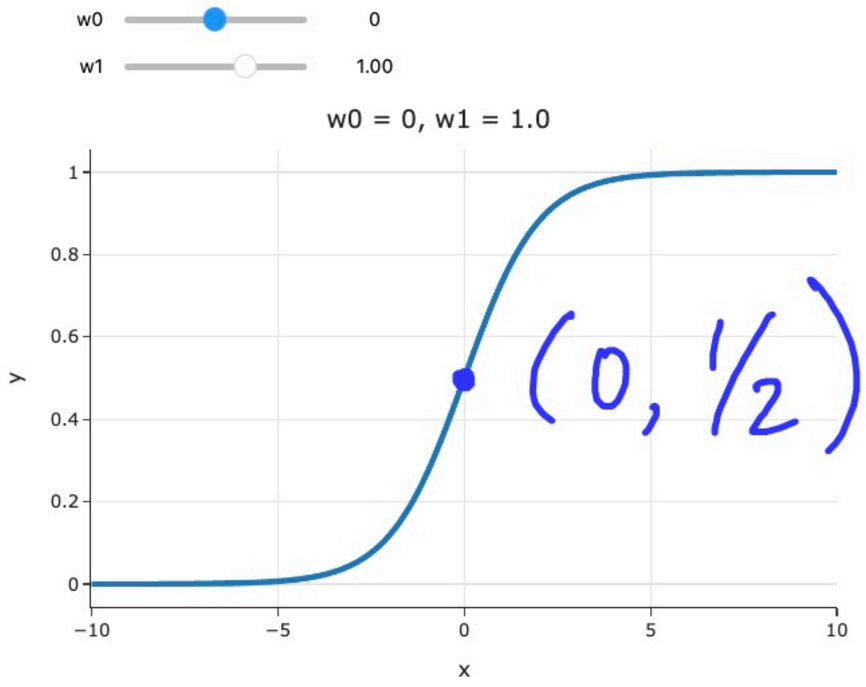
$t \to \infty \quad \sigma(t) = \frac{1}{1+e^{-\infty}} = \frac{1}{1+0} = 1$

$\sigma(t) = \frac{1}{1+e^{-t}}$

Below, interact with the sliders to change the values of w_0 and w_1.

$$y = \sigma(w_0 + w_1 x)$$

In [8]:
```
1 interact(util.plot_sigmoid, w0=(-15, 15), w1=(-3, 3, 0.1));
```



$y = \sigma(x)$

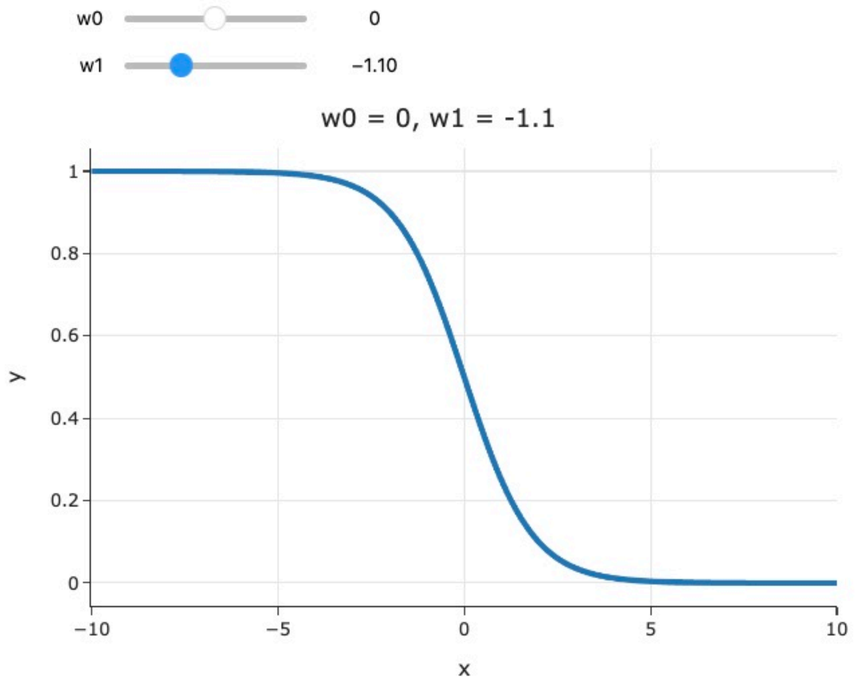
if $D > 0$, $\sigma(D) > 1/2$

if $D < 0$ $\sigma(D) < 1/2$.

Below, interact with the sliders to change the values of w_0 and w_1.

$$y = \sigma(w_0 + w_1 x)$$

In [8]:
```
1 interact(util.plot_sigmoid, w0=(-15, 15), w1=(-3, 3, 0.1));
```



w_1 controls steepness and orientation

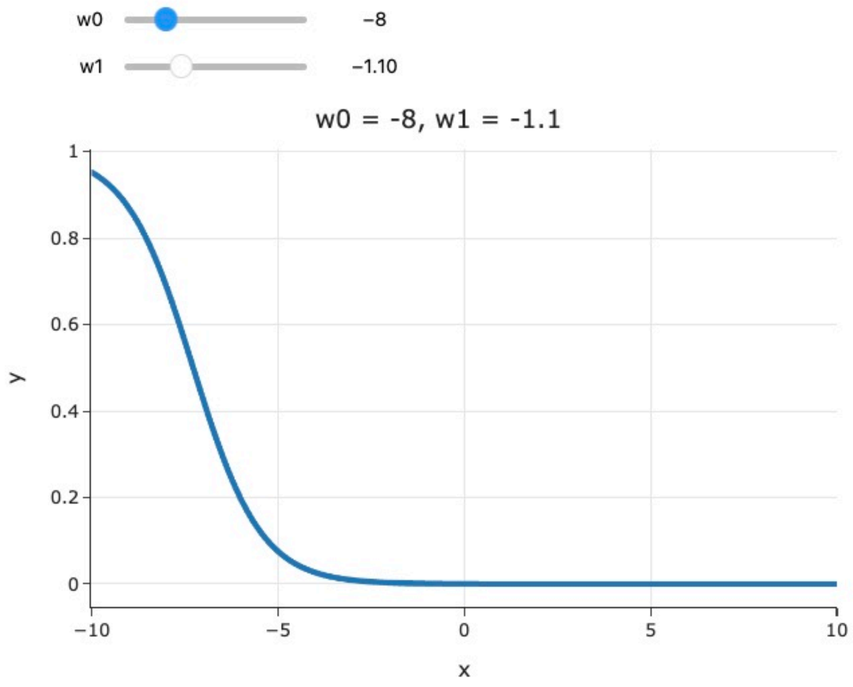
$w_1 > 0$,

$w_1 < 0$,

Below, interact with the sliders to change the values of w_0 and w_1.

$$y = \sigma(w_0 + w_1 x)$$

In [8]:
```
1 interact(util.plot_sigmoid, w0=(-15, 15), w1=(-3, 3, 0.1));
```



w_1 controls horizontal pos.:

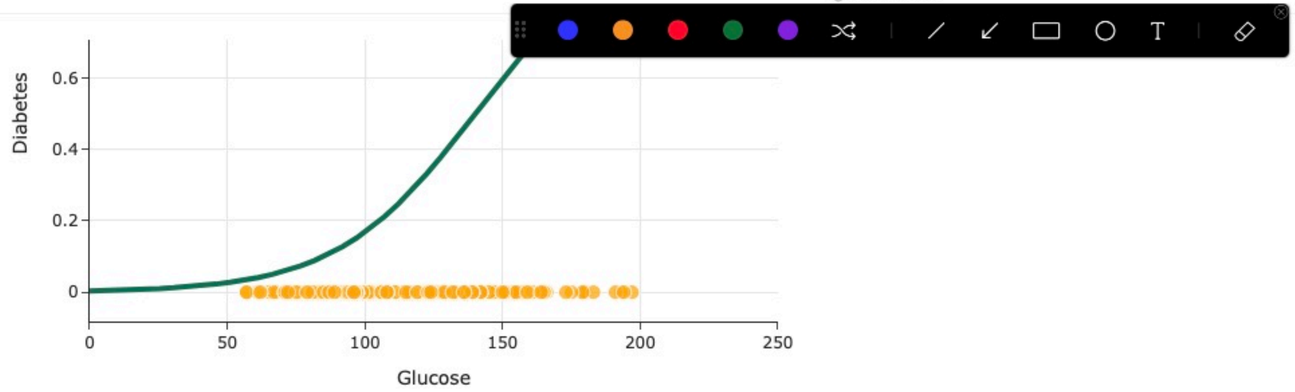
Logistic regression

- Logistic **regression** is a linear **classification** technique that builds upon linear regression.
  It is **not** called logistic**al** regression!

- It models **the probability of belonging to class 1, given a feature vector**:

$$P(y_i = 1|\vec{x}_i) = \sigma(\underbrace{w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \ldots + w_d x_i^{(d)}}_{\text{linear regression model}}) = \sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)$$

forces output to be between 0, 1 so we can interpret as probabilities.



- So, if a patient has a `'Glucose'` level of 150, the model's predicted probability that they have diabetes is:

$$\sigma(-5.59 + 0.04 \cdot 150) \approx \sigma(0.41) \approx 0.601$$

In [15]:
```
1 model_logistic.predict([[150]])
```

Out[15]:
```
array([1])
```

In [18]:
```
1 model_logistic.predict_proba(X_train[['Glucose']])
```

Out[18]:
```
array([[0.75, 0.25],
       [0.83, 0.17],
       [0.44, 0.56],
       ...,
       [0.96, 0.04],
       [0.51, 0.49],
       [0.88, 0.12]])
```

P_0 P_1

In [14]:
```
1 model_logistic.predict_proba([[150]])
```

Attempting to use squared loss

- Our default loss function has always been squared loss, so we could try and use it here.

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)\right)^2$$

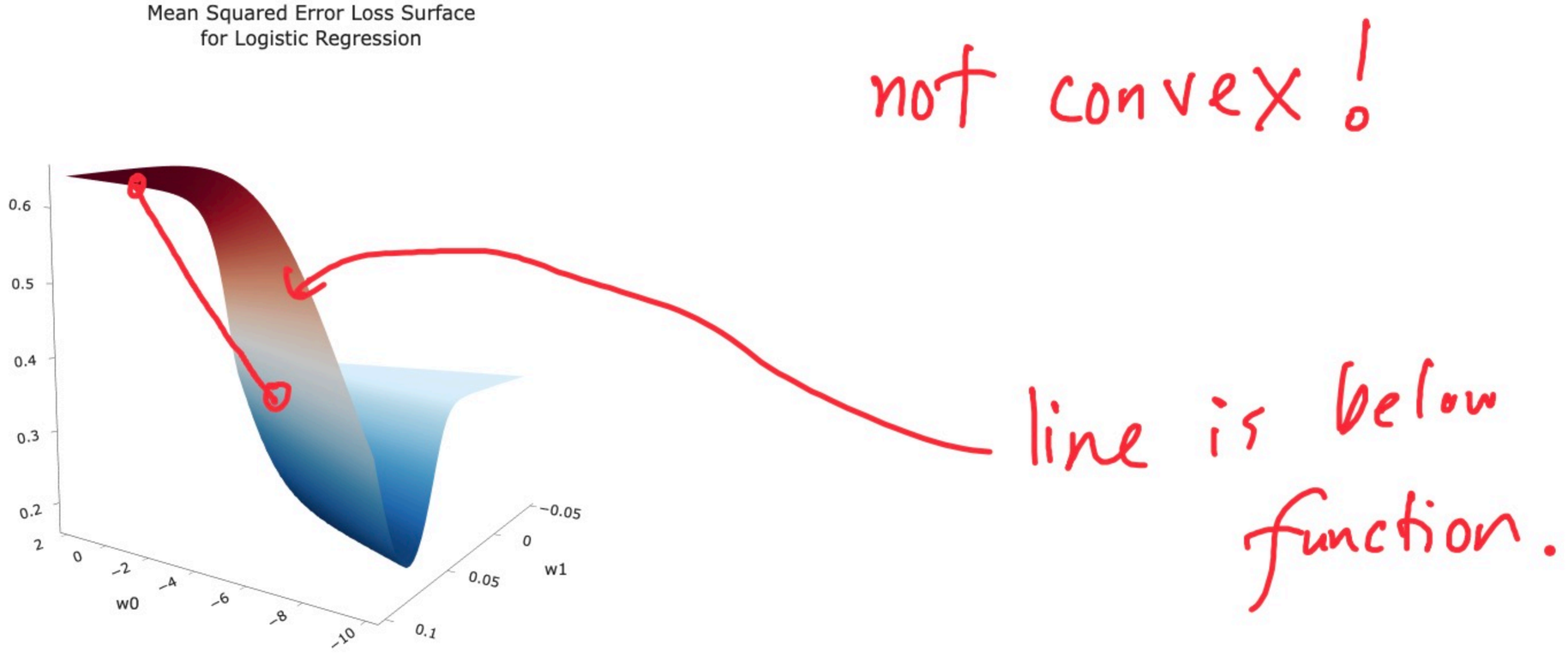
logistic

MSE

$$L_2(y_i, H(x_i)) = (y_i - H(x_i))^2$$

$$R_{\text{sq}}(w_0, w_1) = \frac{1}{n} \sum_{i=1} \left(y_i - \sigma(w_0 + w_1 \underbrace{x_i}_{\text{Glucose}_i}) \right)$$

In [19]:
```
1 util.show_logistic_mse_surface(X_train, y_train)
```

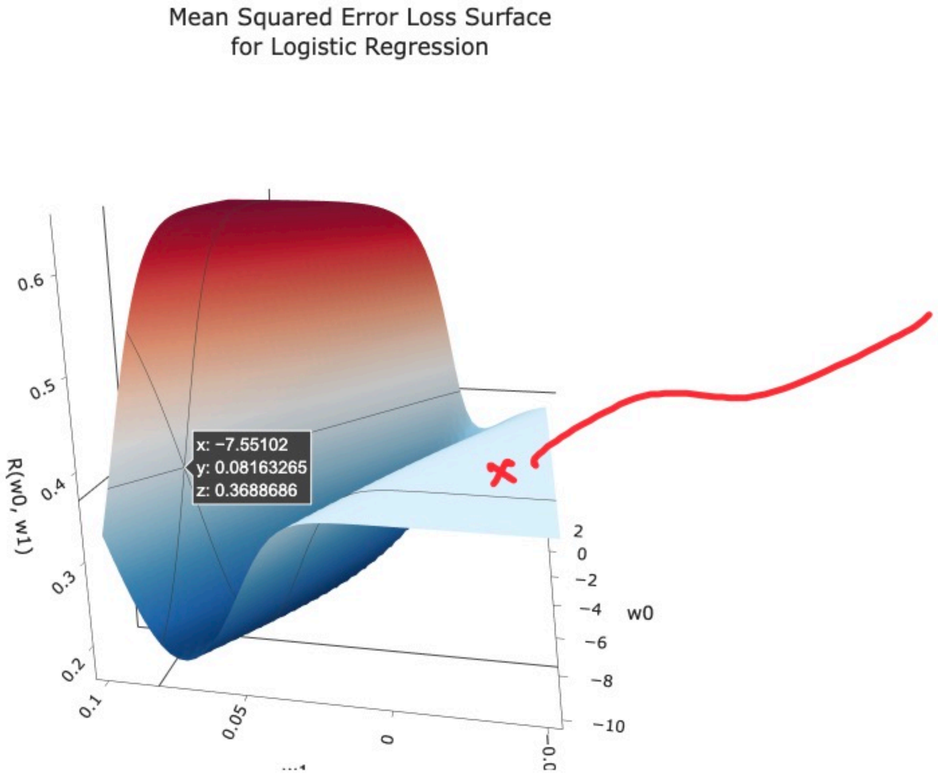


not convex!

line is below function.

$$R_{\text{sq}}(w_0, w_1) = \frac{1}{n} \sum_{i=1} \left(y_i - \sigma(w_0 + w_1 \underbrace{x_i}_{\text{Glucose}_i}) \right)$$

In [19]:
```
1 util.show_logistic_mse_surface(X_train, y_train)
```



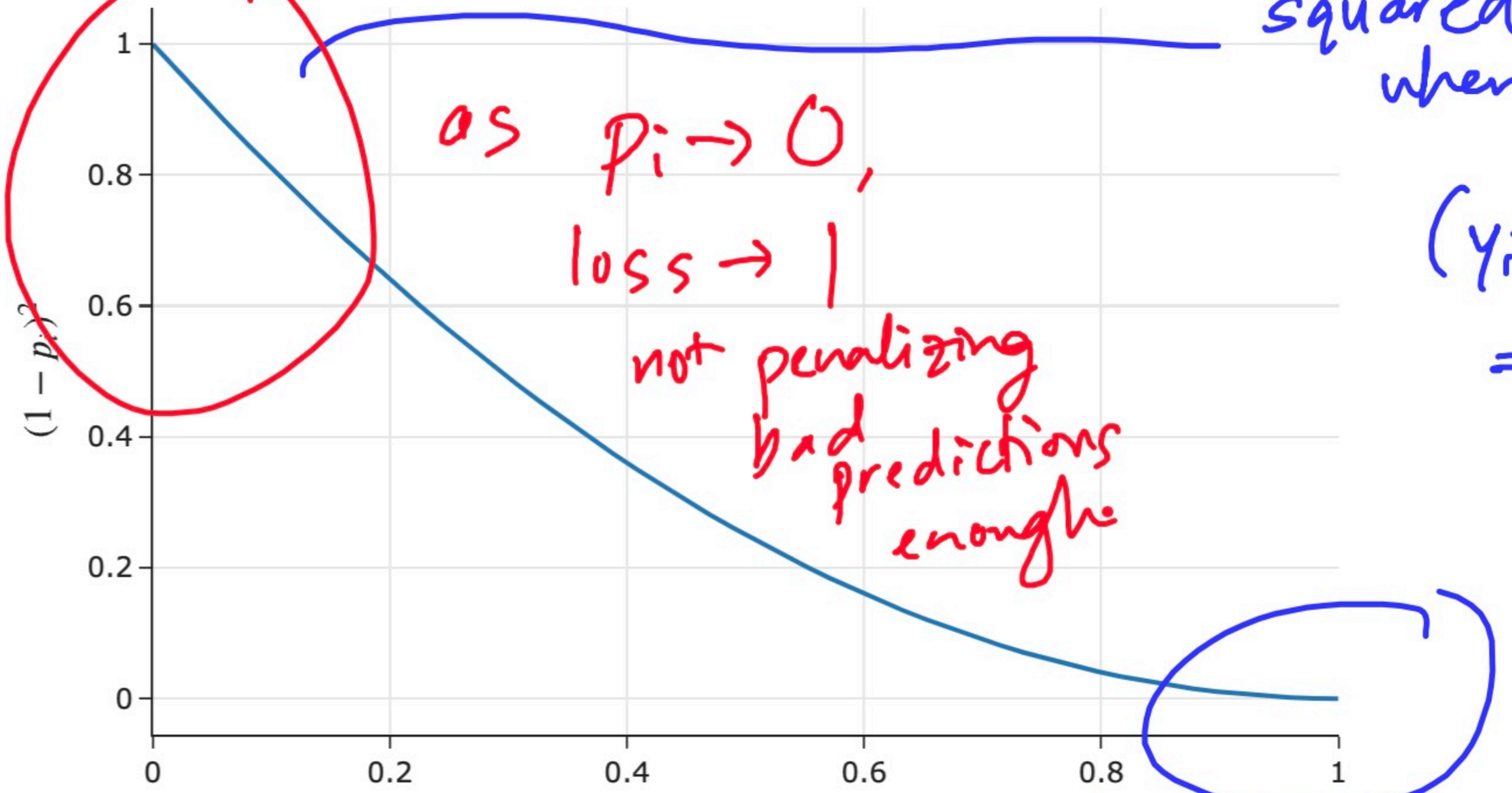
flat region

$\nabla R(\vec{w}) \approx \begin{bmatrix} 0 \\ 0 \end{bmatrix}$,

so gradient descent can get trapped ☹

[20]: 1 util.show_squared

Squared loss is bounded to (0, 1) when predicting probabilities!



squared loss when $y_i = 1$

$(y_i - H(x_i))^2$

$= (1 - p_i)^2$

as $p_i \to 0$, loss $\to 1$

not penalizing bad predictions enough

as $p_i \to 1$, loss $\to 0$

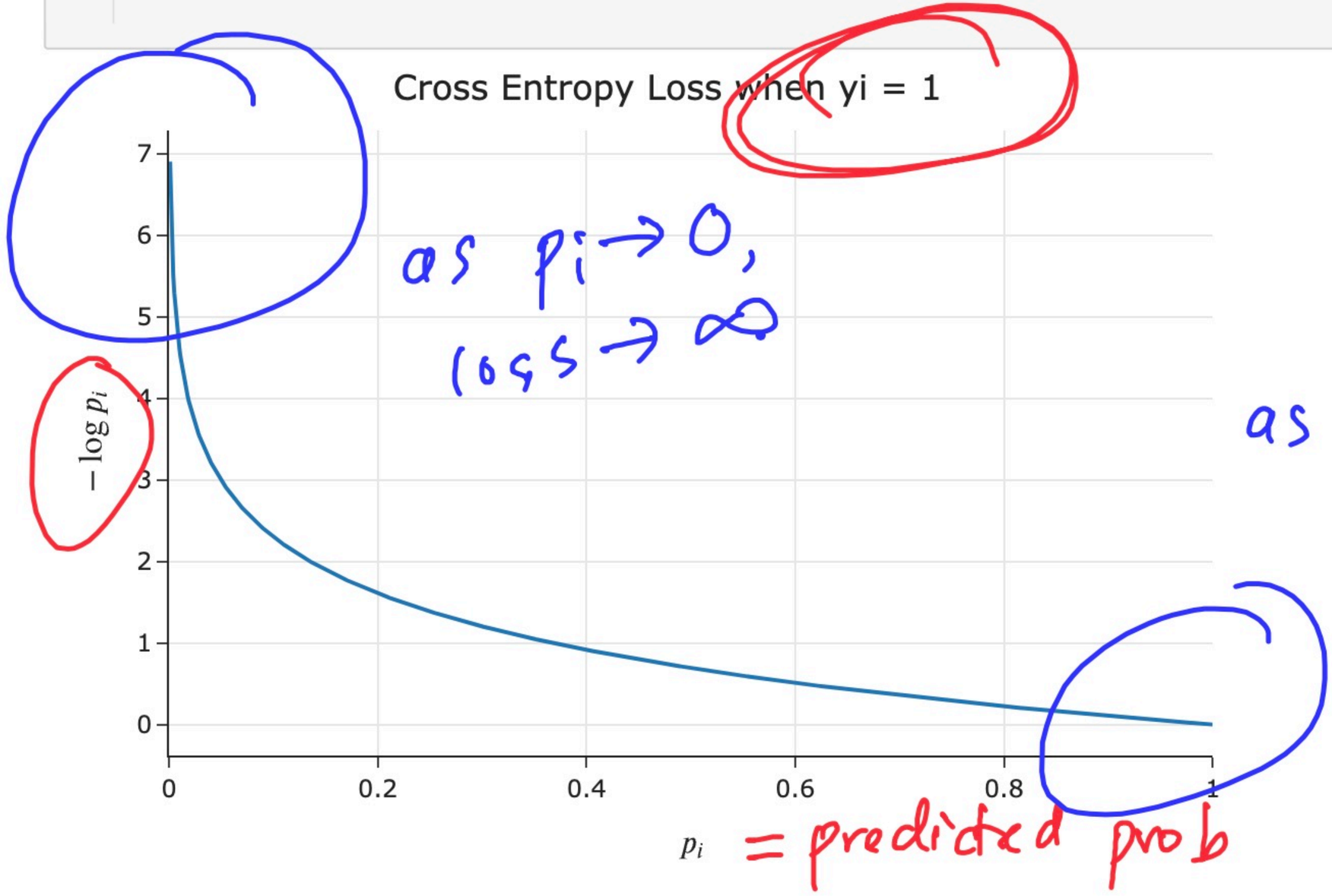
p_i = predicted prob.

ss-entropy loss function piecewise. If y_i is an observed value

$$L_{\text{ce}}(y_i, p_i) = \begin{cases} -\log(p_i) & \text{if } y_i = 1 \\ -\log(1 - p_i) & \text{if } y_i = 0 \end{cases}$$

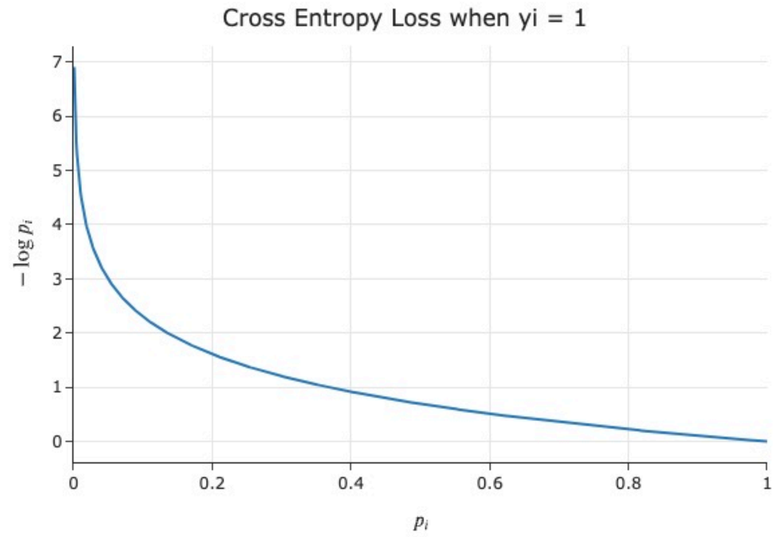
predicted probability

In [21]:
```
1 util.show_ce_loss_individual_1()
```



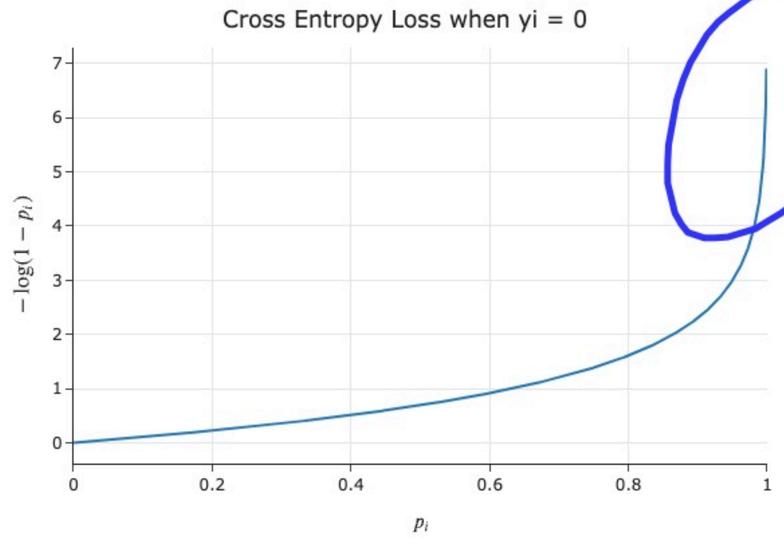
In [ ]:
```
1 util.show_ce_loss_individual_0()
```

In [21]:
```
1 util.show_ce_loss_individual_1()
```



mirror image

In [22]:
```
1 util.show_ce_loss_individual_0()
```



↑ ∞

A non-piecewise definition of cross-entropy loss

- We can define the cross-entropy loss function piecewise. If y_i is an observed value and p_i is a predicted **probability**, then:

$$L_{\text{ce}}(y_i, p_i) = \begin{cases} -\log(p_i) & \text{if } y_i = 1 \\ -\log(1 - p_i) & \text{if } y_i = 0 \end{cases}$$

- An equivalent formulation of L_{ce} that isn't piecewise is:

$$L_{\text{ce}}(y_i, p_i) = -\left(y_i \log p_i + (1 - y_i) \log(1 - p_i)\right)$$

the same

Average cross-entropy loss

- **Cross-entropy loss** for an observed value y_i and predicted **probability** $p_i = P(y_i = 1|\vec{x}_i) = \sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)$ is:

$$L_{\text{ce}}(y_i, p_i) = -\left(y_i \log p_i + (1 - y_i) \log(1 - p_i)\right)$$

- To find $\vec{w}^*$, then, we minimize **average cross-entropy loss**:

$$R_{\text{ce}}(\vec{w}) = -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \log p_i + (1 - y_i) \log(1 - p_i)\right)$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log\left(\sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)\right) + (1 - y_i) \log\left(1 - \sigma\left(\vec{w} \cdot \text{Aug}(\vec{x}_i)\right)\right)\right]$$

minimize this to find $\vec{w}^*$

`LogisticRegression` in `sklearn`, revisited

- The `LogisticRegression` class in `sklearn` has a lot of hidden, default hyperparameters.

In [26]:
```
1 LogisticRegression?
```

- It performs L_2 regularization ("ridge logistic regression") **by default**. The hyperparameter for regularization strength, C, is the **inverse** of λ; by default, it sets $C = 1$.

$$C = \frac{1}{\lambda}$$

- So, for a given value of C, it minimizes:

$$R_{\text{ce-reg}}(\vec{w}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log\left(\sigma\left(\vec{w}\cdot \text{Aug}(\vec{x}_i)\right)\right) + (1-y_i)\log\left(1-\sigma\left(\vec{w}\cdot \text{Aug}(\vec{x}_i)\right)\right)\right] + \frac{1}{C}\sum_{j=1}^{d} w_j^2$$

average CE loss

L_2 reg.

- By default, the `predict` method of a fit `LogisticRegressio`

In [30]:
```
model_logistic.predict(pd.DataFrame([{
    'Glucose': 125,
}]))
```

predicts class with larger prob

Out[30]:
```
array([0])
```

- But, logistic regression is designed to predict **probabilities**. We c probabilities using the `predict_proba` method, as we saw ear

In [29]:
```
model_logistic.predict_proba(pd.DataFrame([{
    'Glucose': 125,
}]))
```

p_0, p_1

Out[29]:
```
array([[0.65, 0.35]])
```

Thresholding probabilities

- As we did with other classifiers, we can visualize the **decision boundary** of a fit logistic regression model.

- If we pick a threshold of T, any patient with a `'Glucose'` value such that:

$$\sigma(w_0^* + w_1^* \cdot \text{Glucose}_i) \geq T$$

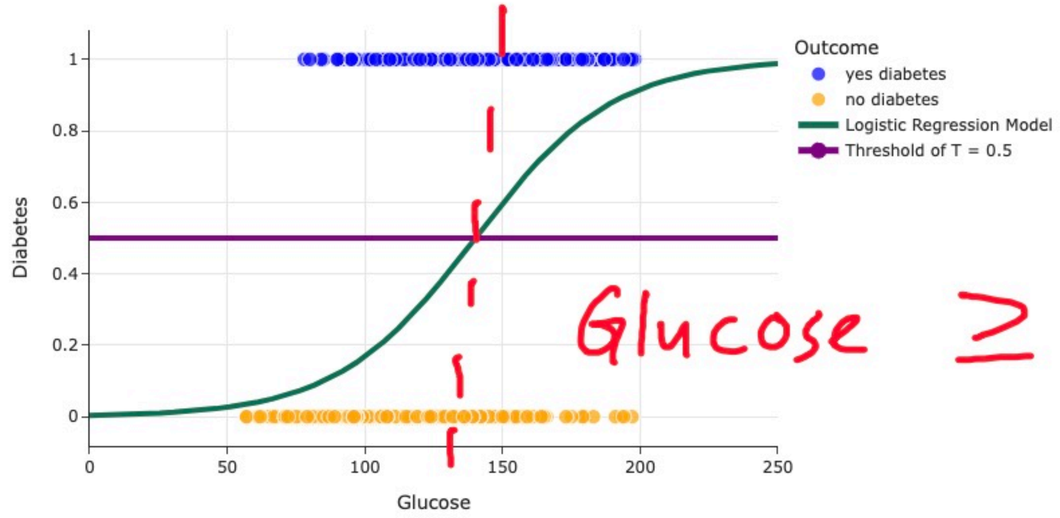
is classified as having diabetes.

by default, $T = 0.5$

is classified as having diabetes.

- For example, if $T = 0.5$:

In [31]:
```
1 util.show_one_feature_plot_with_logistic_and_y_threshold(X_train, y_train, 0.5)
```



Decision boundaries for logistic regression

- In our single feature model that predicts `'Outcome'` given just `'Glucose'`, our predicted probabilities are of the form:

$$P(y_i = 1 | \text{Glucose}_i) = \sigma\left(w_0^* + w_1^* \cdot \text{Glucose}_i\right)$$

- Suppose we fix a threshold, T. Then, our **decision boundary** is of the form:

$$\sigma\left(w_0^* + w_1^* \cdot \text{Glucose}_T\right) = T$$

σ^{-1} σ^{-1}

want to solve for the Glucose_T that's on the boundary. How?

- Suppose an event occurs with probability p.

- The **odds** of that event are:

$$\text{odds}(p) = \frac{p}{1-p}$$

- For instance, if there's a $p = \frac{3}{4}$ chance that Michigan wins this week, then the **odds** that Michigan wins this week are:

$$\text{odds}\left(\frac{3}{4}\right) = \frac{\frac{3}{4}}{\frac{1}{4}} = 3$$

- Interpretation: it's 3 times more likely that Michigan wins than loses.

- **We can interpret $\sigma^{-1}(p) = \log\left(\frac{p}{1-p}\right)$ as the "log odds" of p!**

See the reference slides for more details.

look for Homework 11.

Solving for the decision boundary

$\sigma(t) = \frac{1}{1+e^{-t}}$.

- Previously, we said that if we pick a threshold T, then:

$$\sigma\left(w_0^* + w_1^* \cdot \text{Glucose}_T\right) = T$$

- We re-arranged this for the `'Glucose'` value on the threshold, Glucose_T:

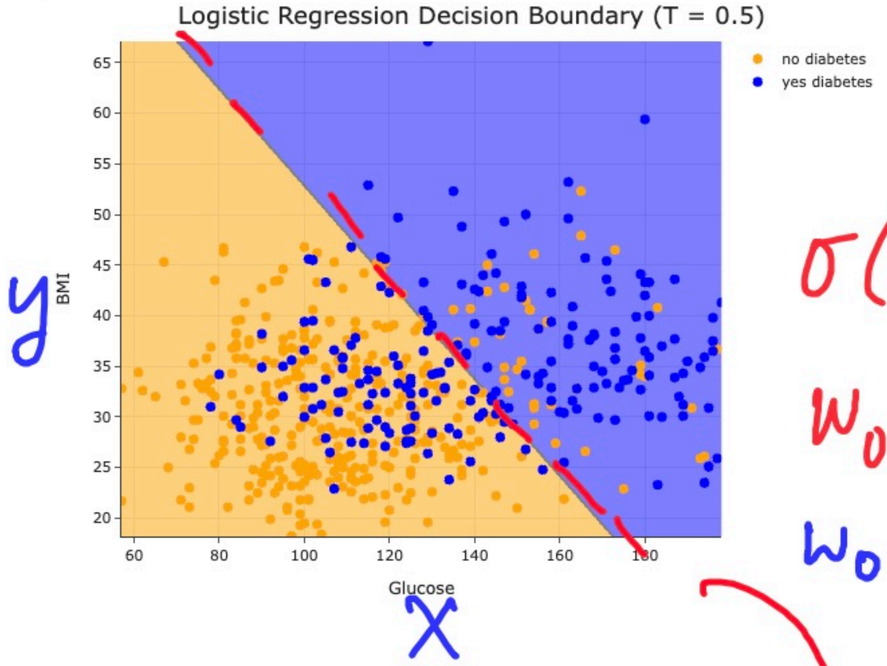
$$\text{Glucose}_\text{T} = \frac{\sigma^{-1}(T) - w_0^*}{w_1^*}$$

$\log\left(\frac{T}{1-T}\right)$

The decision boundary in the feature space

- What does the resulting decision boundary look like, in a $d = 2$ dimensional plot?

```
In [46]: 1 util.show_decision_boundary(model_logistic_multiple, X_train, y_train, title='Logistic Regression Decision Boundary (T = 0.5)')
```



$\sigma(w_0 + w_1 \text{Glucose} + w_2 \text{BMI}) = 0.5$

$w_0 + w_1 \text{Glucose} + w_2 \text{BMI} = \sigma^{-1}(0.5)$

$w_0 + w_1 x + w_2 y = \sigma^{-1}(0.5)$

$y = \square + \square x$

$\text{BMI} = \square + \square \text{Glucose}$

- Note that unlike the decision boundaries for k-nearest neighbors and decision trees, this decision boundary is **linear**.

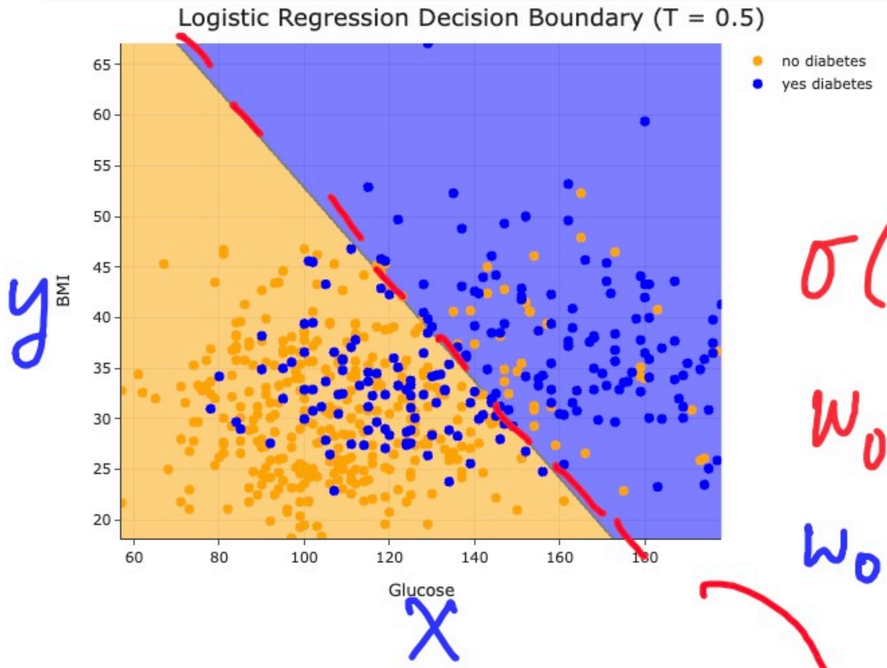
- Specifically, the decision boundary in the feature space is of the form:

$$a \cdot \text{Glucose}_i + b \cdot \text{BMI}_i + c = 0$$

The decision boundary in the feature space

- What does the resulting decision boundary look like, in a $d = 2$ dimensional plot?

```
In [46]: 1 util.show_decision_boundary(model_logistic_multiple, X_train, y_train, title='Logistic Regression Decision Boundary (T = 0.5)')
```



linear comb. of features!

$\sigma(w_0 + w_1 \text{Glucose} + w_2 \text{BMI}) = 0.5$

$w_0 + w_1 \text{Glucose} + w_2 \text{BMI} = \sigma^{-1}(0.5)$

$w_0 + w_1 x + w_2 y = \sigma^{-1}(0.5)$

$y = \square + \square x$

$\text{BMI} = \square + \square \text{Glucose}$

- Note that unlike the decision boundaries for k-nearest neighbors and decision trees, this decision boundary is **linear**.

- Specifically, the decision boundary in the feature space is of the form:

$$a \cdot \text{Glucose}_i + b \cdot \text{BMI}_i + c = 0$$

$ax + by + c = 0$