$P(y=1|\vec{x}) = \sigma(\underbrace{\vec{w}\cdot \text{Aug}(\vec{x})})$

Properties of $\sigma(t)$

$P(y=1|\vec{x}) = p = \sigma(\vec{w}\cdot \text{Aug}(\vec{x}))$

Take $\sigma^{-1}(\ )$ of both sides

$$\boxed{\sigma^{-1}(p) = \vec{w}\cdot \text{Aug}(\vec{x})}$$

$$\log\left(\frac{p}{1-p}\right) = \vec{w}\cdot \text{Aug}(\vec{x})$$

$$\boxed{\frac{p}{1-p} = e^{\underbrace{\vec{w}\cdot \text{Aug}(\vec{x})}}}$$

odds ratio!

Question 🤔 (Answer at practicaldsc.org/q)

Which expression describes the **odds ratio**,

$$\frac{P(y=1|\vec{x})}{P(y=0|\vec{x})}$$

in the logistic regression model?

important: $\sigma^{-1}(p) = \log\left(\frac{p}{1-p}\right)$

- A. $\vec{w} \cdot \text{Aug}(\vec{x})$
- B. $-\vec{w} \cdot \text{Aug}(\vec{x})$
- C. $e^{\vec{w} \cdot \text{Aug}(\vec{x})}$
- D. $\sigma(\vec{w} \cdot \text{Aug}(\vec{x}))$
- E. None of the above.

"sigmoid" = "logistic function"

$P(y=1|\vec{x}) = \sigma(\vec{w} \cdot \text{Aug}(\vec{x}))$

Fact about σ:

looking for $P(y=0|\vec{x}) = 1 - \sigma(\vec{w} \cdot \text{Aug}(\vec{x}))$

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

Question 🤔 (Answer at practicaldsc.org/q)

Which expression describes $P(y=0|\vec{x})$ in the logistic regression model?

- A. $\sigma\left(\vec{w} \cdot \text{Aug}(\vec{x})\right)$
- B. $-\sigma\left(\vec{w} \cdot \text{Aug}(\vec{x})\right)$
- C. $\sigma\left(-\vec{w} \cdot \text{Aug}(\vec{x})\right)$
- D. $1 - \log\left(1 + e^{\vec{w} \cdot \text{Aug}(\vec{x})}\right)$
- E. $1 + \log\left(1 + e^{-\vec{w} \cdot \text{Aug}(\vec{x})}\right)$

$P(y=0|\vec{x}) = \sigma(-\vec{w} \cdot \text{Aug}(\vec{x})) = 1 - \sigma(\vec{w} \cdot \text{Aug}(\vec{x}))$

$\sigma(-t) = \underline{1 - \sigma(t)} \Leftarrow \boxed{\sigma(t) = 1 - \sigma(-t)}$

Here, a false positive (FP) is when we predict that someone has diabetes when they do not.

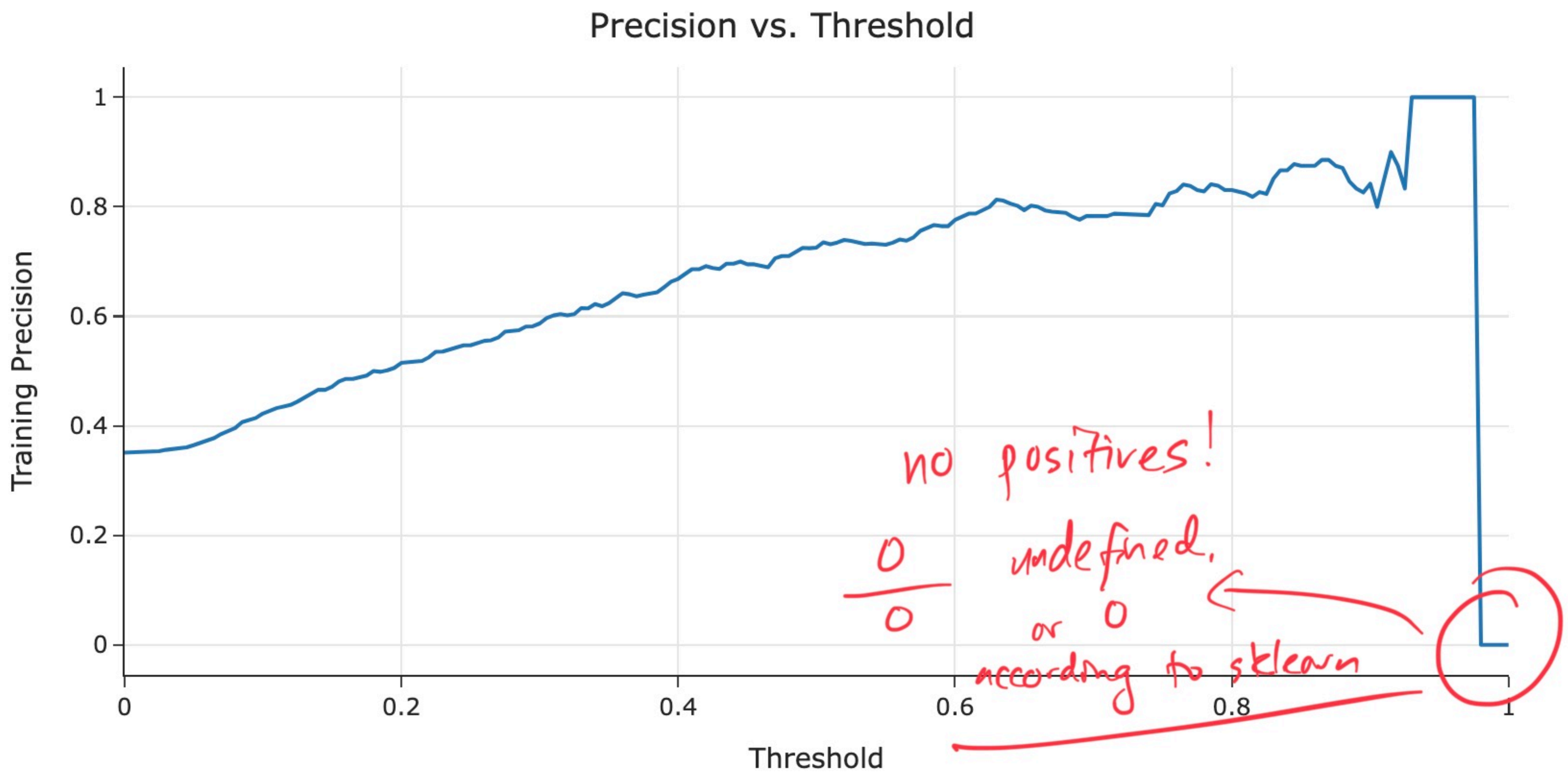
precision = $\frac{TP}{TP+FP}$

- How does the model's training **precision** change as the threshold changes?

In [23]:
```
util.plot_vs_threshold(X_train, y_train, 'Precision')
```

as threshold ↑,
positives ↓

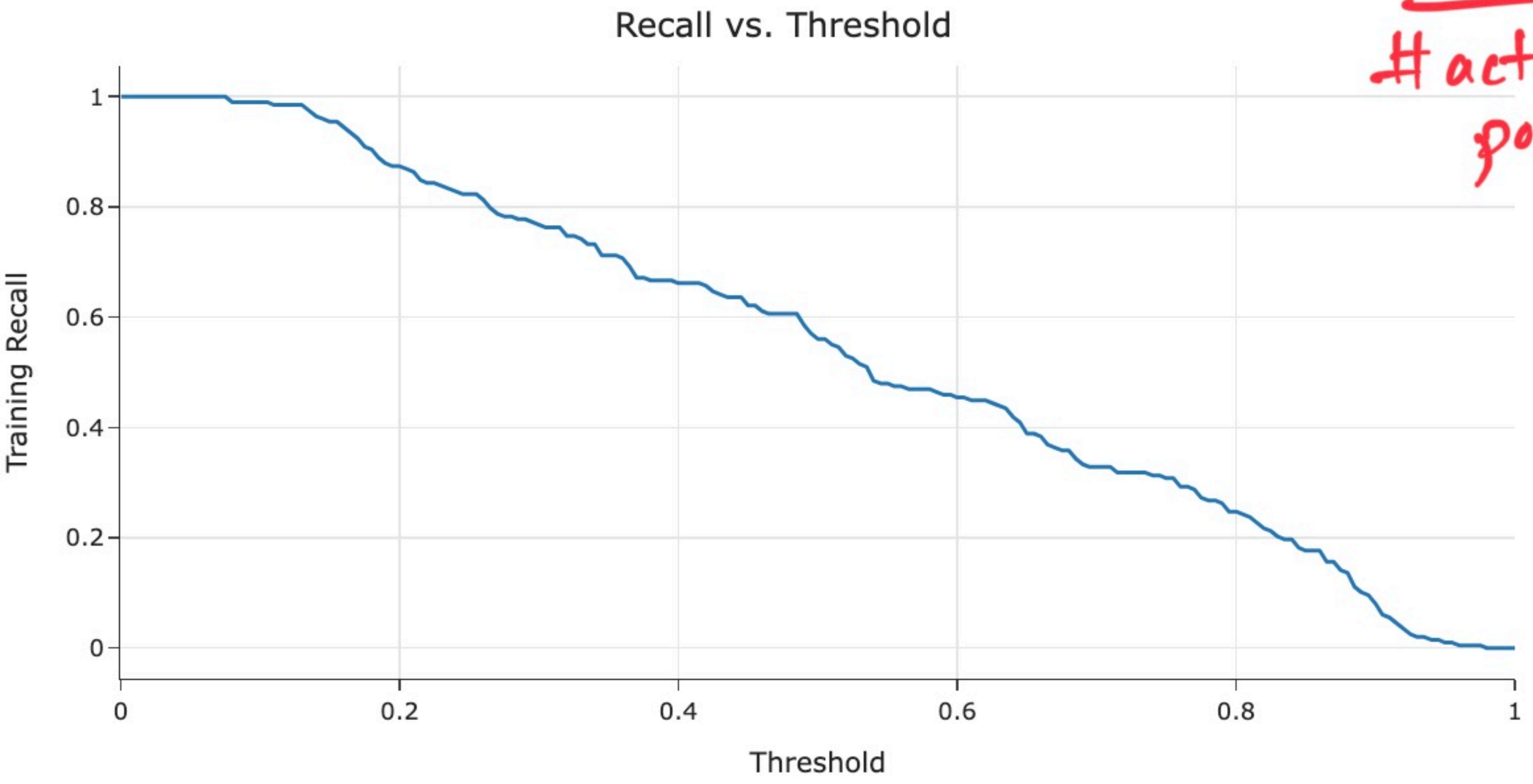
23]:
```
util.plot_vs_threshold(X_train, y_train, 'Precision')
```



changes?

$$\text{recall} = \frac{TP}{TP+FN} = \frac{TP}{\#\text{ actually po}}$$

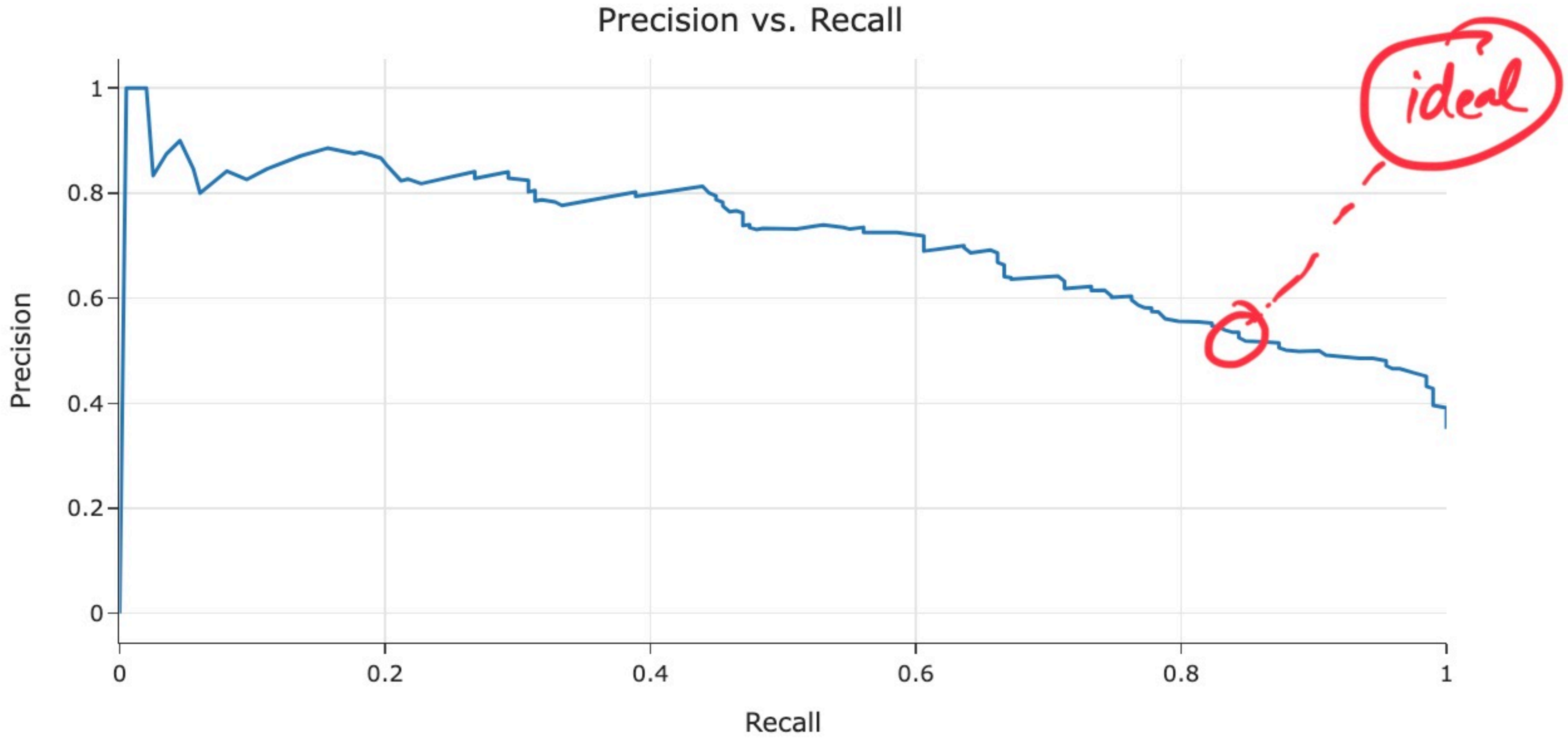
In [24]:
```
util.plot_vs_threshold(X_train, y_train, 'Recall')
```



- We can visualize how precision and recall vary **together**.

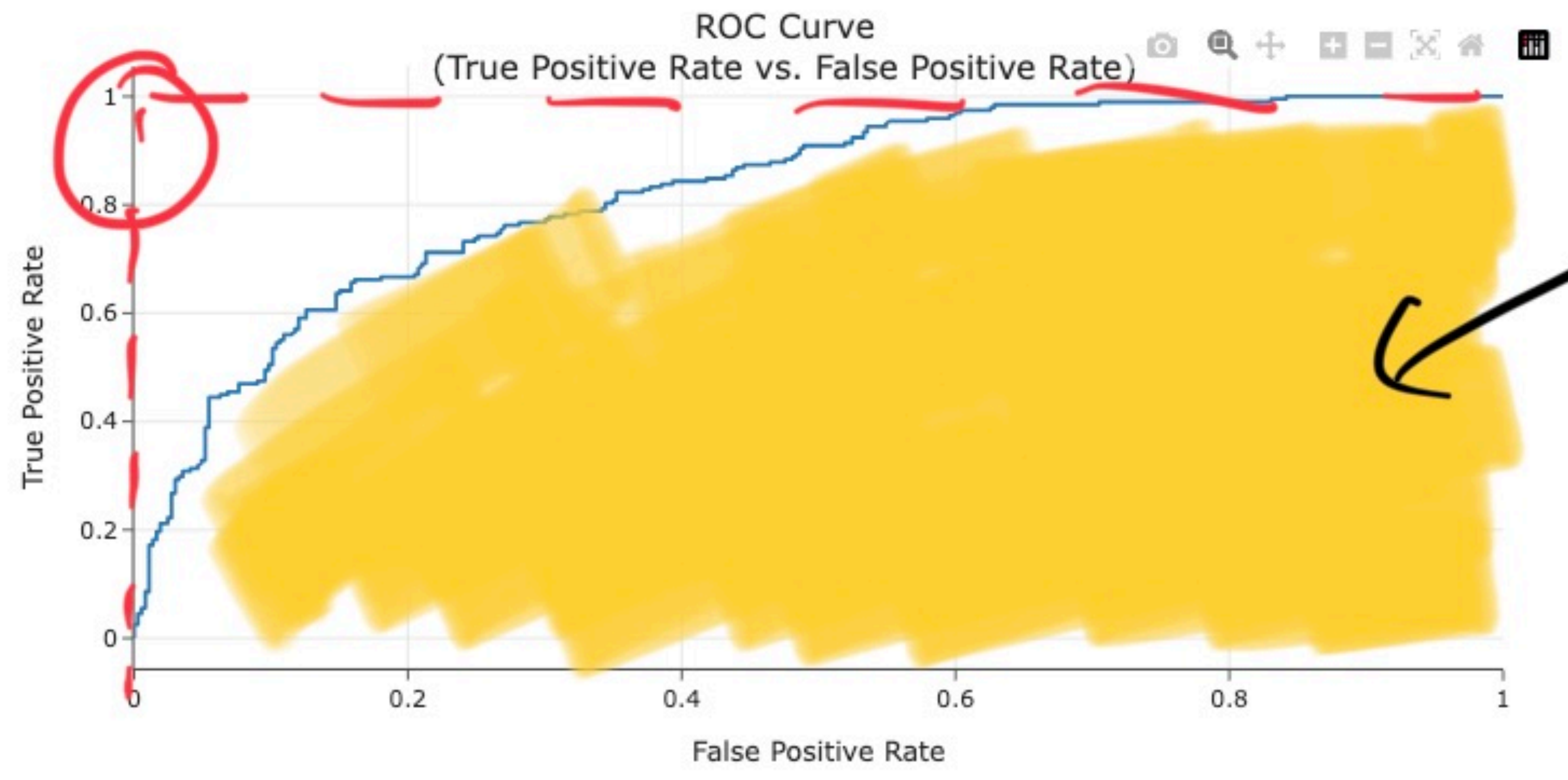
```
In [25]: util.pr_curve(X_train, y_train)
```

How to choose T?



- The ROC curve for our classifier looks like:

```
In [26]: util.draw_roc_curve(X_train, y_train)
```



Approximately 0.8

ideal classifier: 1

want AUC to be high!

- If we care about TPR and FPR equally, the best threshold is the one whose point is closest to the **top left corner** in the plot above.

Why? The top left corner is where $TPR = 1$ and $FPR = 0$, and we want TPR to be high and FPR to be low.

- A common metric for the quality of a binary classifier is the area under curve (AUC) for the ROC curve.

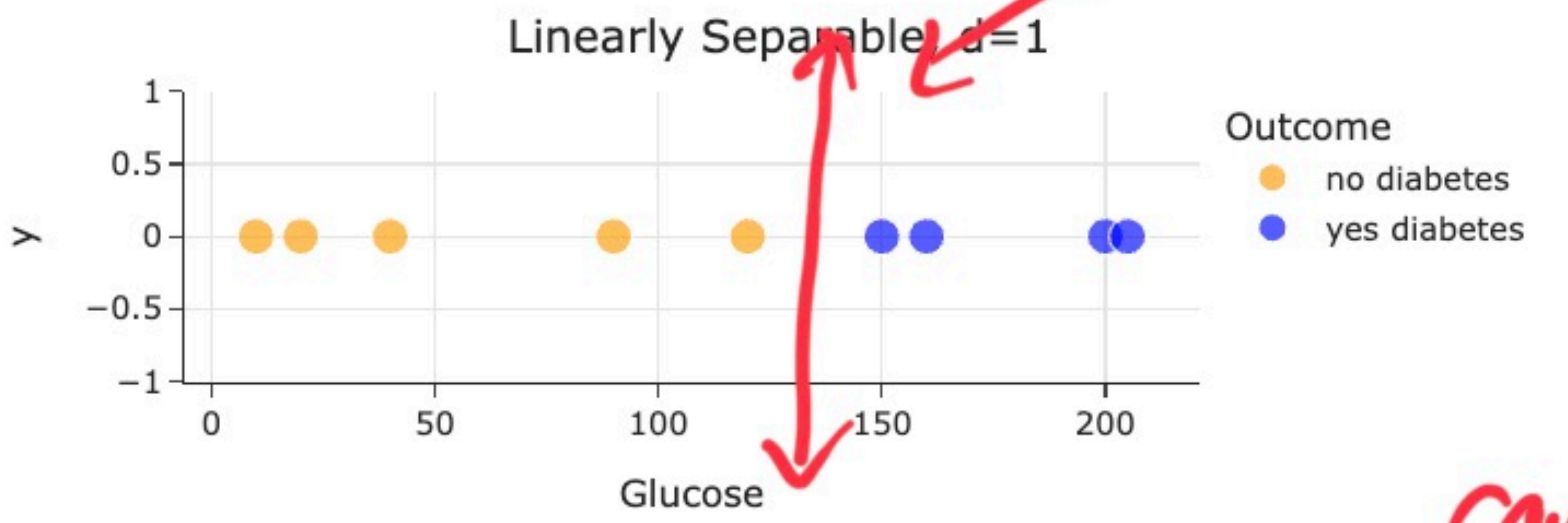
Larger values are better!

- A dataset is **linearly separable** if a line, plane, or hyperplane can be drawn in d-dimensio **perfectly separates** the two classes.

- Example: $d = 1$.

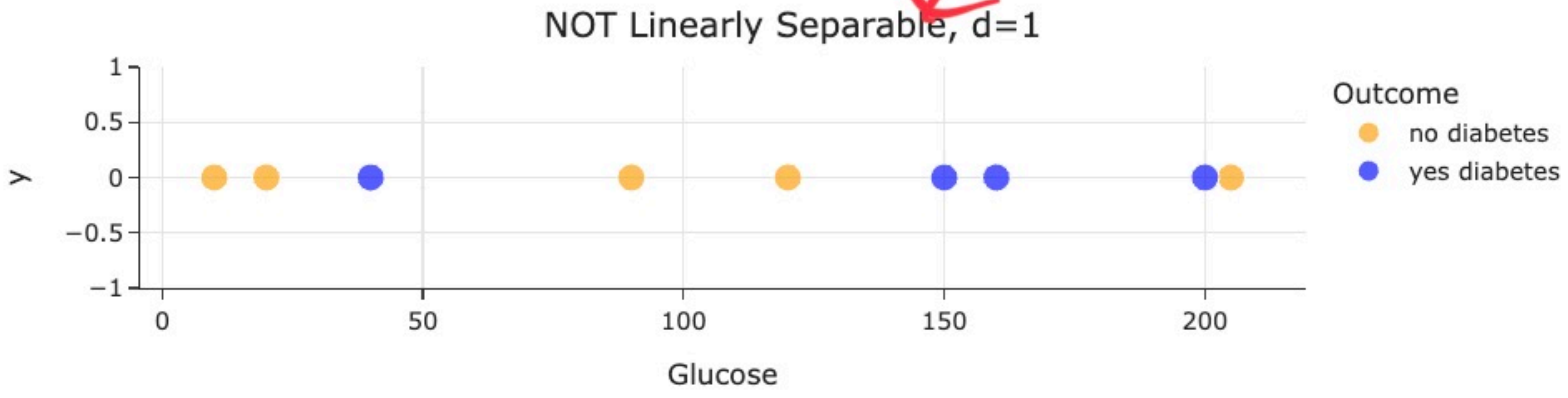
separates the classes!

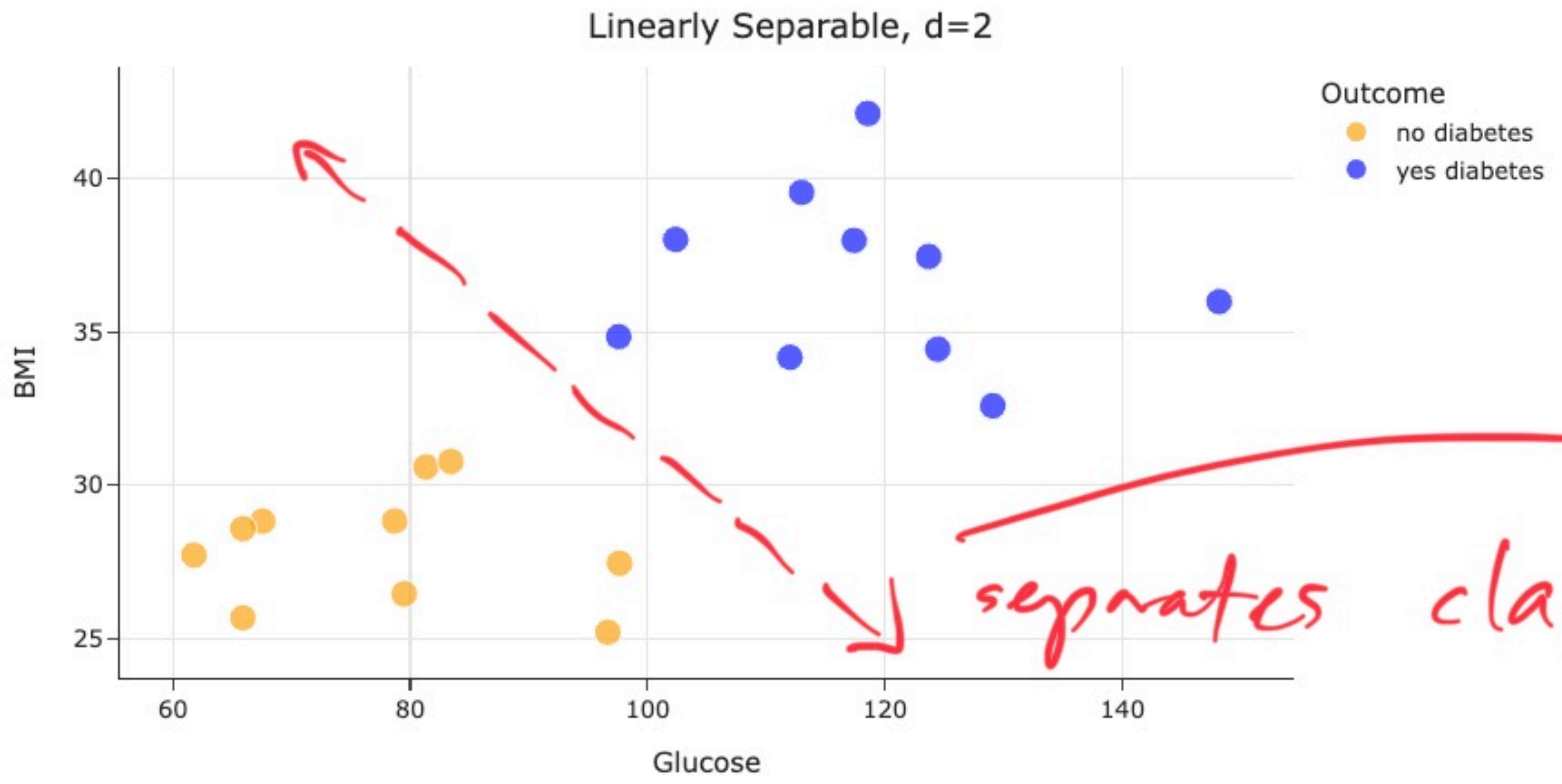
```
In [29]: util.lin_sep_1D()
```



can't separate

```
In [30]: util.non_lin_sep_1D()
```



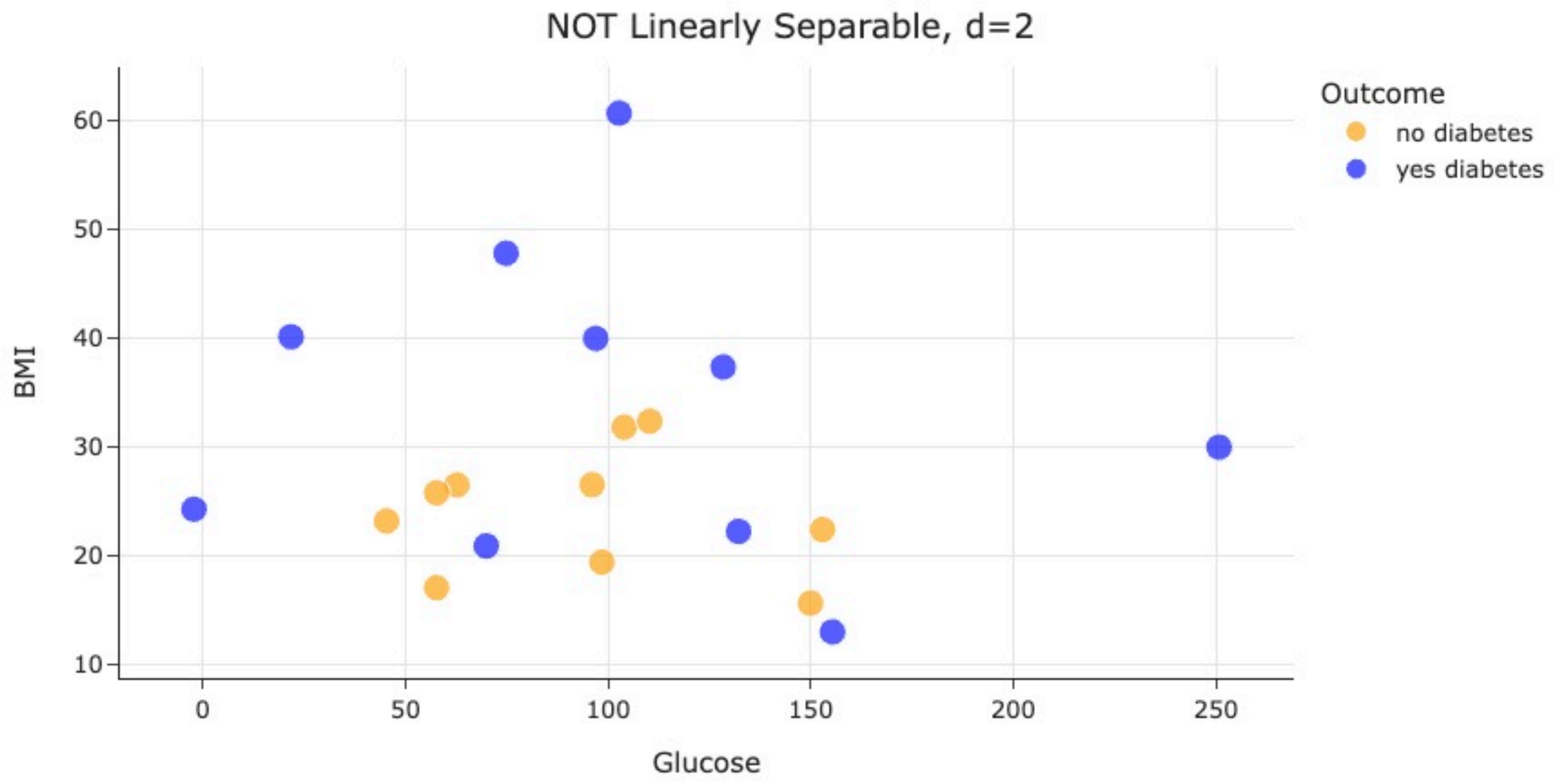


2 features

$BMI = -4 \cdot Glucose + 15$
(example)

separates classes

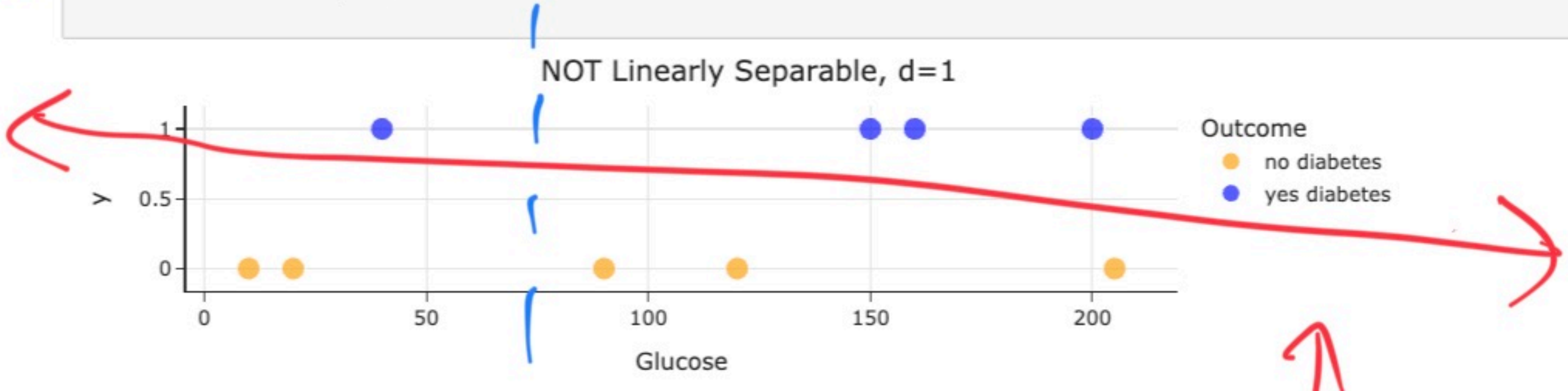
In [32]:
```
util.non_lin_sep_2D()
```



no such line!

- Why is the dataset below **not** linearly separable?

In [33]:
```
util.bad_example_1D()
```



no valid vertical line.

doesn't count! 2D line, but we only have d=1 features

Glucose

- Why would the optimal w_1^* below tend to ∞? See the annotated slides for more details.

$$\sigma(t) = \underbrace{\frac{1}{1+e^{-t}}}_{\text{never exactly 0 or exactly 1}}$$

$$P(y=1|\text{Glucose}) = \sigma(w_0 + w_1 \cdot \text{Glucose}) = \frac{1}{1+e^{-(w_0 + w_1 \cdot \text{Glucose})}}$$

```
In [36]: util.lin_sep_1D_elevated()
```

steepness of $\int$ $= w_1^*$

keep getting steeper

29 . 1

0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
...	...	...	...	...	...	...	...
330	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
331	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
332	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

333 rows × 7 columns

- Here, each row corresponds to a single penguin.

- There are three `'species'` of penguin: Adelie, Chinstrap, and Gentoo.

In [38]:
```
penguins['species'].value_counts(normalize=True)
```

Out[38]:
```
species
Adelie       0.44
Gentoo       0.36
Chinstrap    0.20
Name: proportion, dtype: float64
```

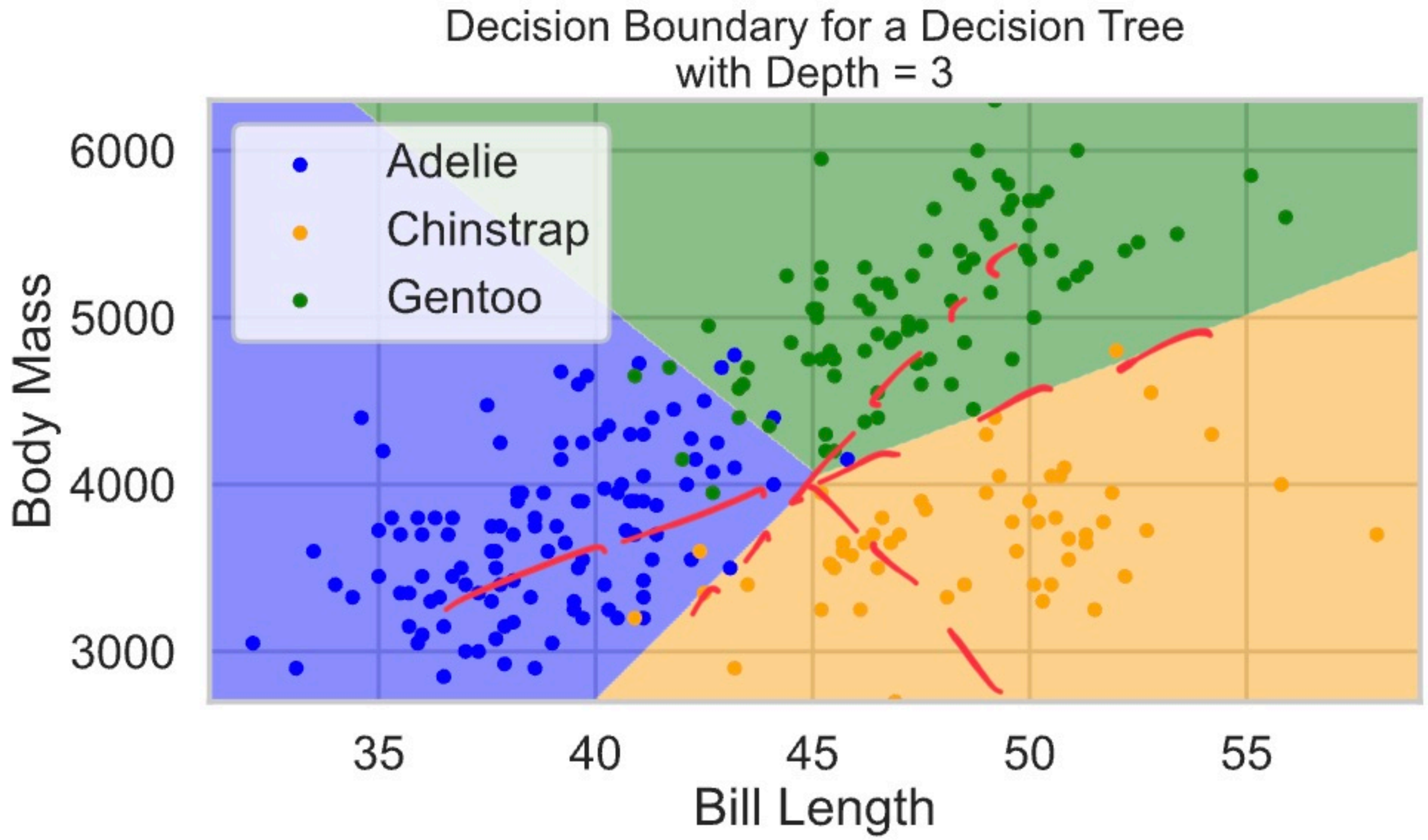
always predict Adelie, accuracy = 44%

- **Question**: What accuracy would the best "constant" classifier achieve on this data?

▸ **estimator: LogisticRegression**

▸ LogisticRegression

In [55]:
```
util.penguin_decision_boundary(model_logistic_ovr, X_train, y_train, title="Decision Boundary for a Decision Tree\
```



- Note that the resulting decision boundaries are still linear!