- Our first classification example will involve predicting whether or not a patient has diabetes, given other information about their health.

In [2]:
```
diabetes = pd.read_csv('data/diabetes.csv')
display_df(diabetes, cols=9)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.63	50	1
1	1	85	66	29	0	26.6	0.35	31	0
2	8	183	64	0	0	23.3	0.67	32	1
...	...	...	...	...	...	...	...	...	...
765	5	121	72	23	112	26.2	0.24	30	0
766	1	126	60	0	0	30.1	0.35	47	1
767	1	93	70	31	0	30.4	0.32	23	0

768 rows × 9 columns

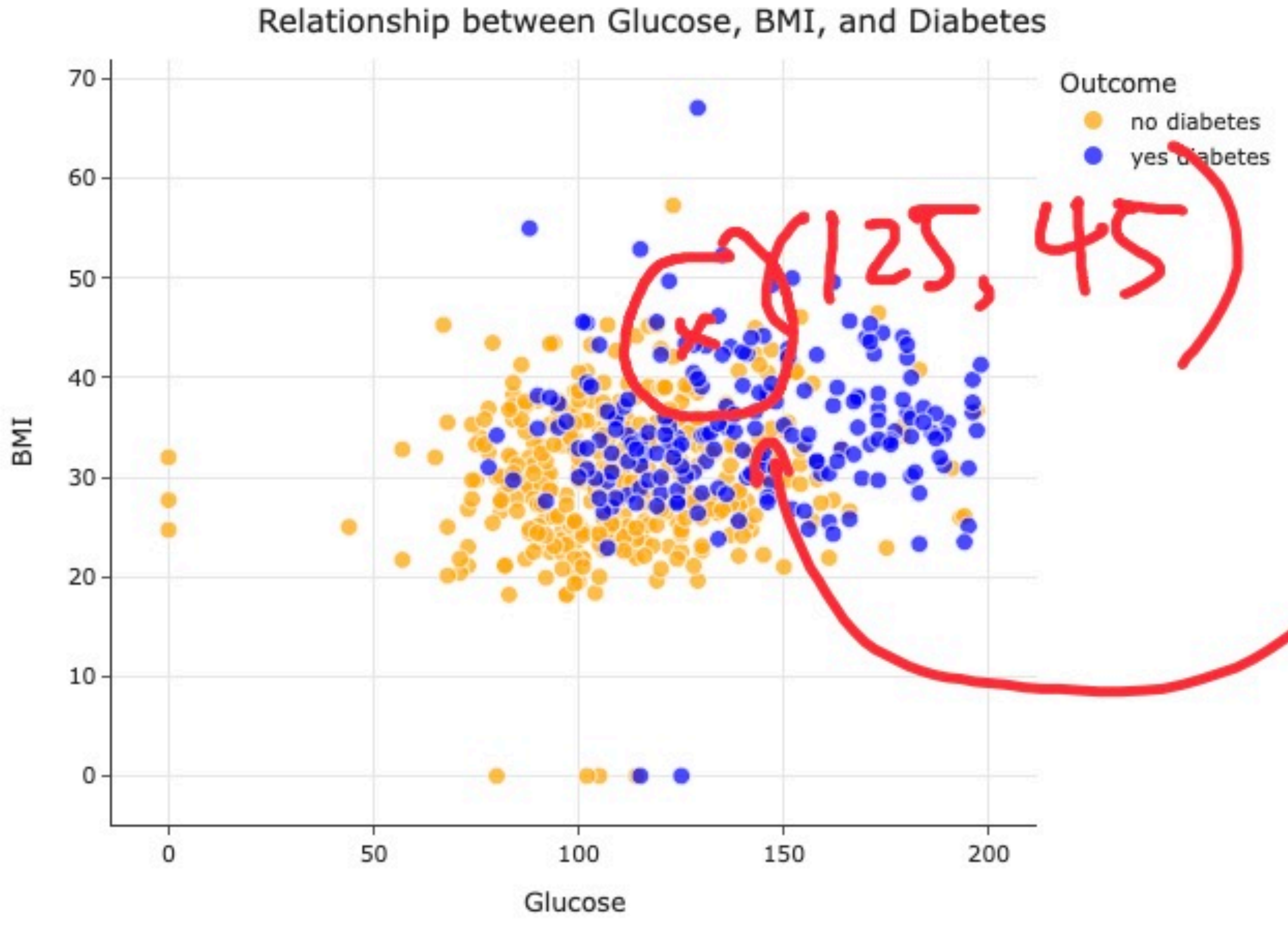
$\vec{y}$

In [3]:
```
# 0 means no diabetes, 1 means yes diabetes.
diabetes['Outcome'].value_counts()
```

Out[3]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

- Since there are only two possible `'Outcome'`s, we can draw a 2D scatter plot of `'BMI'` vs. `'Glucose'` and color each point by `'Outcome'`. Below, **class 0 (orange) is "no diabetes"** and **class 1 (blue) is "diabetes"**.

In [6]:
```
fig = util.create_base_scatter(X_train, y_train)
fig
```



look at people most similar to new patient.

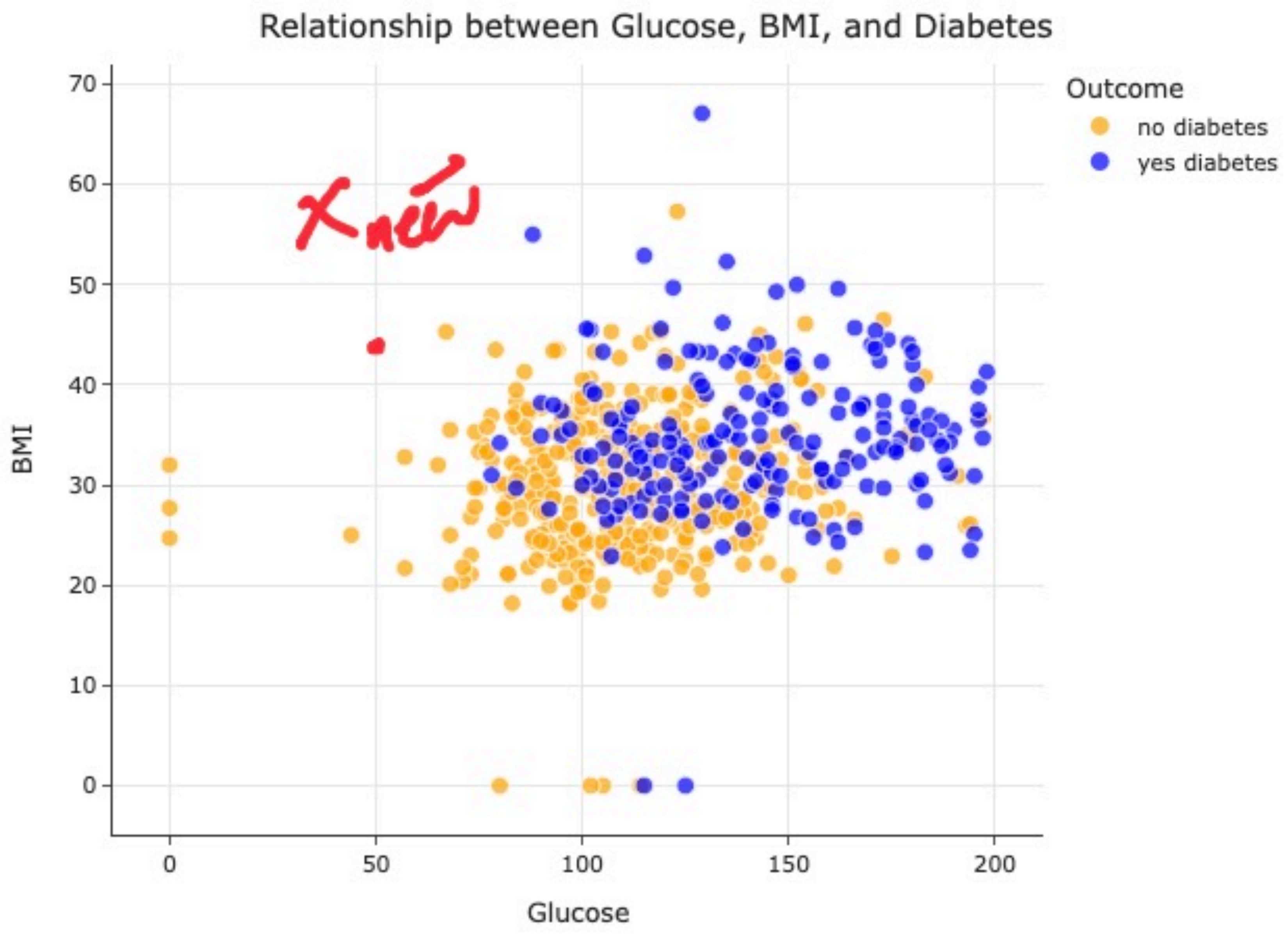
- Using this dataset, how can we classify whether someone new (not already in the dataset) has diabetes, given their `'Glucose'` and `'BMI'`?

- **Intuition**: If a new person's feature vector is **close to the blue points**, we'll predict **blue (diabetes)**; if they're **close to the orange points**, we'll predict **orange (no diabetes)**.

1. Finding the k closest po

2. Predicting that $\vec{x}_{\text{new}}$ belongs to the **most common class** among those k closest points.

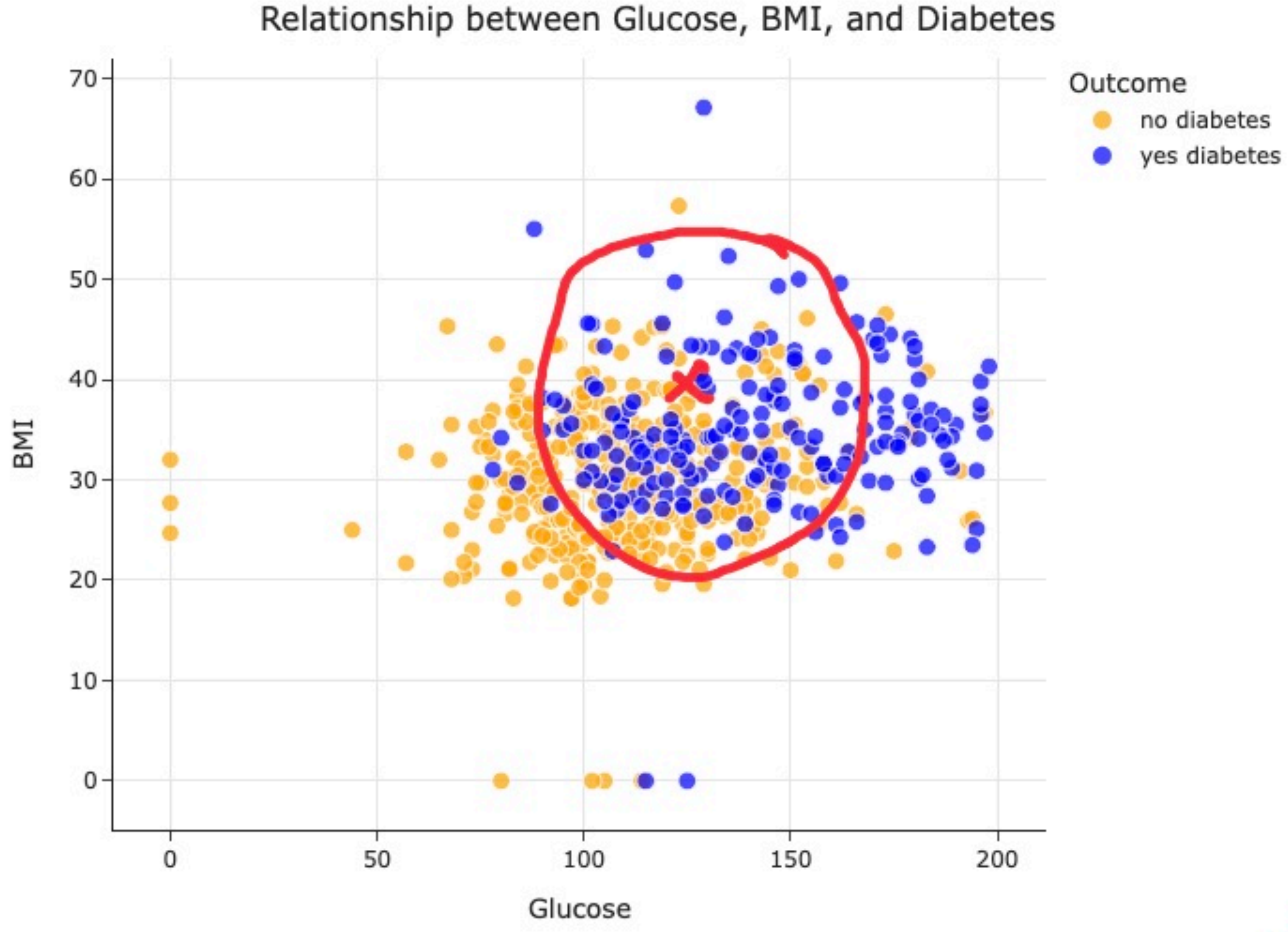
```
In [7]: fig
```



- Example: Suppose $k = 6$. If, among the 6 closest points to $\vec{x}_{\text{new}}$, there are **4 blue** and **2 orange** points, we'd predict **blue (diabetes)**.

What if there are ties? Read here.

In [11]:
```
# To know what reasonable values for 'Glucose' and 'BMI' might be, let's look at the plot again.
fig
```



In [12]:
```
model_knn.predict(pd.DataFrame([{
    'Glucose': 125,
    'BMI': 40
}]))
```

Out[12]: `array([0])`

Among the 28 closest points to (125, 40), the majority did not have diabetes.

- What does the resulting model **look like** 👀? Can we visualize it?

- The most common evaluation metric in classification is **accuracy**:

$$\text{accuracy} = \frac{\text{\# points classified correctly}}{\text{\# points}}$$

if model is a regression model, model.score returns R^2.

Accuracy ranges from 0 to 1, i.e. 0% to 100%. **Higher** values indicate **better** model performance.

In [28]:
```
# Equivalent to 75%.
(model_knn.predict(X_test) == y_test).mean()
```

Out[28]: 0.75

- This is the default metric that the `score` method of a classifier computes, too.

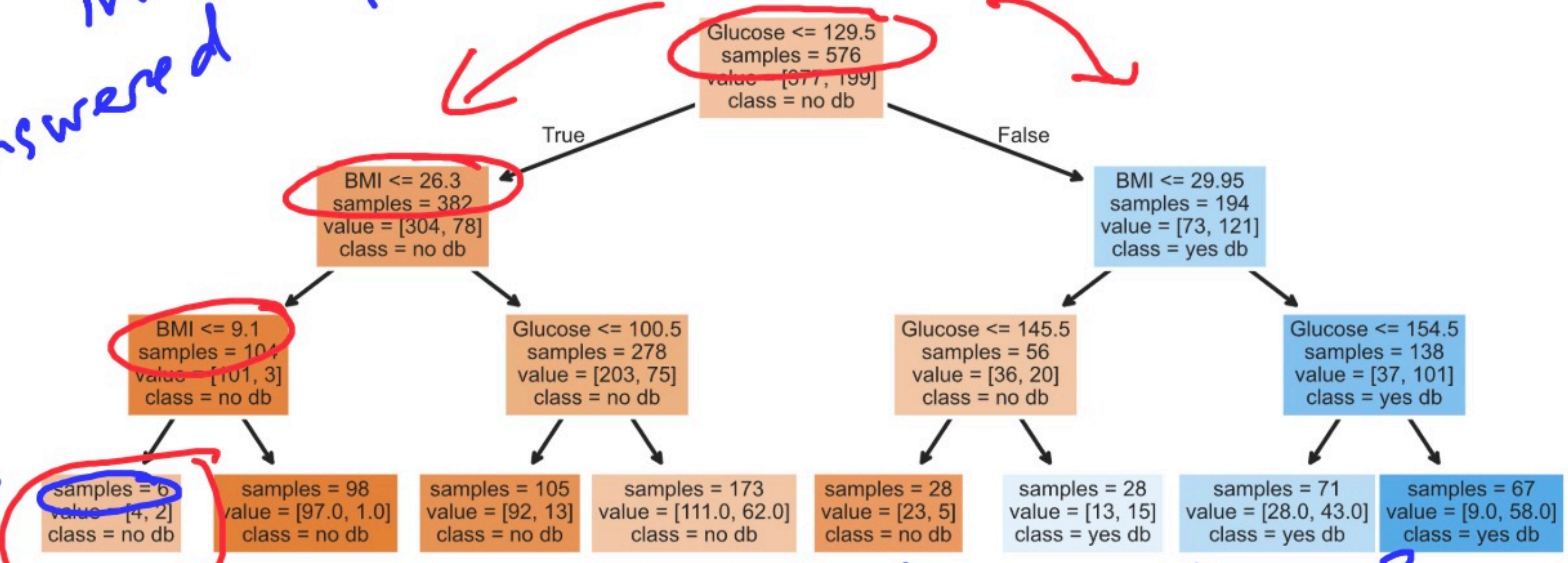
In [29]:
```
model_knn.score(X_test, y_test)
```

Out[29]: 0.75

In [ ]:
```
# For future reference.
test_scores = pd.Series()
...
test_scores
```

k like:

samples = 6 means 6 people in the training set answered "yes", then "yes", then "yes"

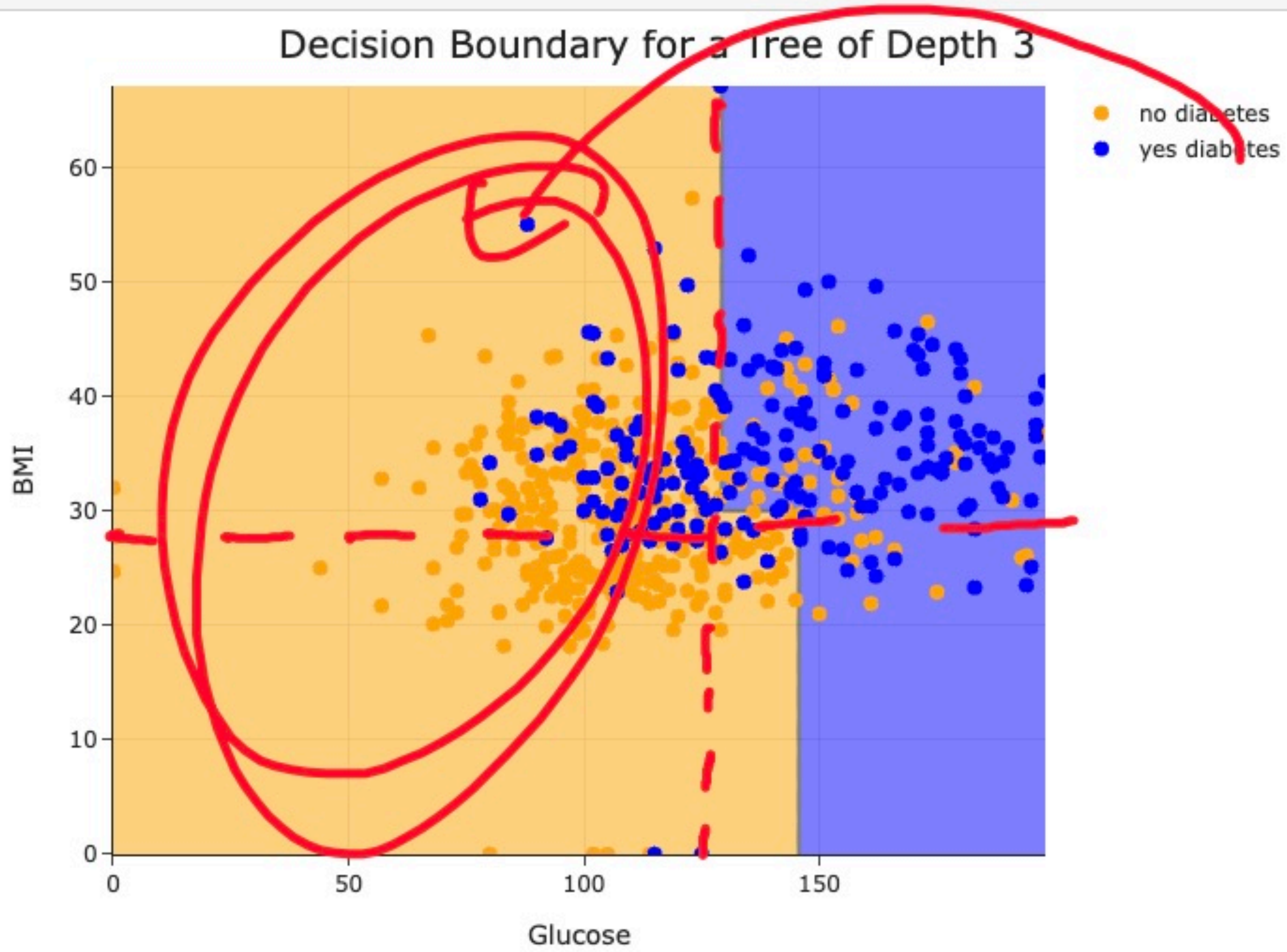


how to pick which questions?

Gini impurity, entropy → probability.

Decision boundaries for a decision tree classifier

In [51]:
```
1 util.show_decision_boundary(model_tree, X_train, y_train, title='Decision Boundary for a Tree of Depth 3')
```



answered "Yes" to

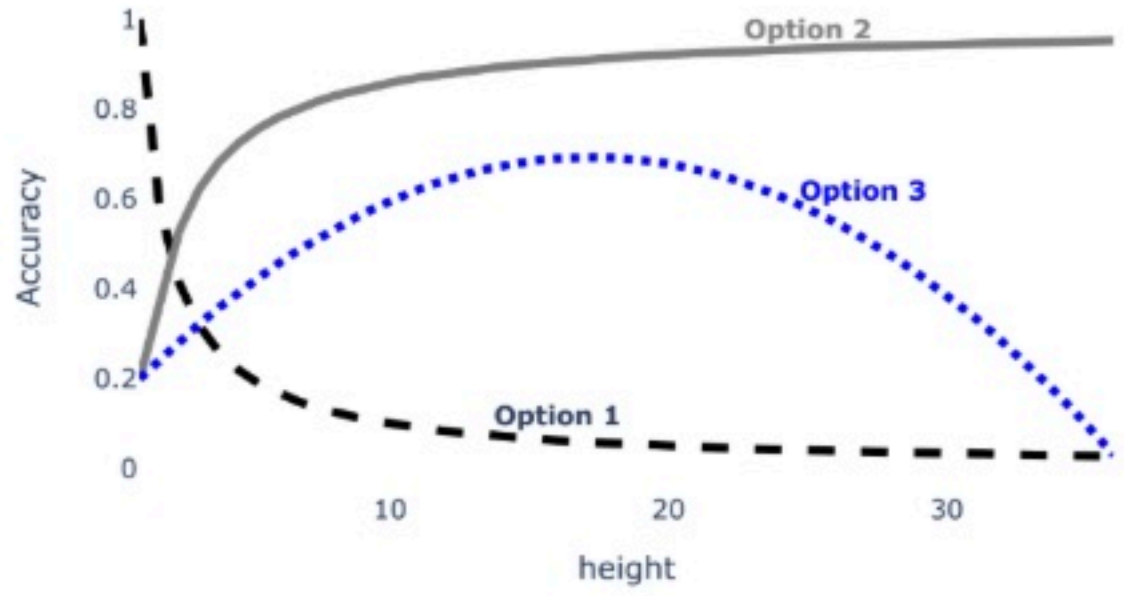
Glucose <= 129.5

- Observe that the decision boundaries – at least when we set `max_depth` to 3 – look less "jagged" than with the k-NN classifier.

Decision trees partition the feature space into rectangles.

`ChickenClassifier`s have many hyperparameters, one of which is `height`. As we increase the value of `height`, the model variance of the resulting `ChickenClassifier` also increases.

First, we consider the training and testing accuracy of a `ChickenClassifier` trained using various values of `height`. Consider the plot below.



accuracy high = good model!

Which of the following depicts **training accuracy vs. `height`**?

- Option 1
- Option 2
- Option 3

as height ↑, complexity ↑, tendency to overfit ↑

Which of the following depicts **testing accuracy vs. `height`**?

- ~~Option 1~~
- ~~Option 2~~
- Option 3

- When performing **binary** classification, there are four possible outcomes.

Note: A "positive prediction" is a prediction of 1, and a "negative prediction" is a prediction of 0.

Outcome of Prediction	Definition	True Class
True positive (TP) ✅	The predictor **correctly** predicts the positive class.	P
False negative (FN) ❌	The predictor incorrectly predicts the negative class.	P
True negative (TN) ✅	The predictor **correctly** predicts the negative class.	N
False positive (FP) ❌	The predictor incorrectly predicts the positive class.	N

FP

first letter tells you if prediction is correct

- We typically organize the four quantities above into a **confusion matrix**.

	Predicted Negative	Predicted Positive
Actually Negative	TN ✅	FP ❌
Actually Positive	FN ❌	TP ✅

- Note that in the four acronyms – TP, FN, TN, FP – the **first letter** is whether the prediction is correct, and the **second letter** is what the prediction is.

Example: Accuracy of COVID tests

- The results of 100 Michigan Medicine COVID tests are given below.

	Predicted Negative	Predicted Positive
Actually Negative	TN = 90 ✅	FP = 1 ❌
Actually Positive	FN = 8 ❌	TP = 1 ✅

Michigan Medicine test results

- 🤔 **Question:** What is the accuracy of the test?

$$\text{accuracy} = \frac{\text{\# points classified correctly}}{\text{\# points}}$$

$$\frac{91}{91+9} = \frac{91}{100} = 91\%.$$

30 . 1

Example: Accuracy of COVID tests

- The results of 100 Michigan Medicine COVID tests are given below.

	Predicted Negative	Predicted Positive
Actually Negative	TN = 90 ✅	FP = 1 ❌
Actually Positive	FN = 8 ❌	TP = 1 ✅

Michigan Medicine test results

91 people don't have COVID

only 9 people actually do

- 🤔 **Question:** What is the accuracy of the test?

$$\text{accuracy} = \frac{\text{\# points classified correctly}}{\text{\# points}}$$

- 🙋 **Answer:**

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN} = \frac{1 + 90}{100} = 0.91$$

- **Followup:** At first, the test seems good. But, suppose we build a classifier that predicts that **nobody has COVID**. What would its accuracy be?

- **Answer to followup:** Also 0.91! There is severe **class imbalance** in the dataset, meaning that most of the data points are in the

Precision

	Predicted Negative	Predicted Positive
Actually Negative	TN = 0 ✅	FP = 91 ❌
Actually Positive	FN = 0 ❌	TP = 9 ✅

everyone-has-COVID classifier

- The **precision** of a binary classifier is the proportion of **predicted positive instances** that are correctly classified. We'd like this number to be as close to 1 (100%) as possible.

$$\text{precision} = \frac{TP}{\#\text{ predicted positive}} = \frac{TP}{TP + FP}$$

- To compute precision, look at the **right (positive) column** of the above confusion matrix.

Tip: A good way to remember the difference between precision and recall is that in the denominator for 🅿recision, both terms have 🅿 in them (TP and FP).