510 DATA SCIENCE

Lecture 10

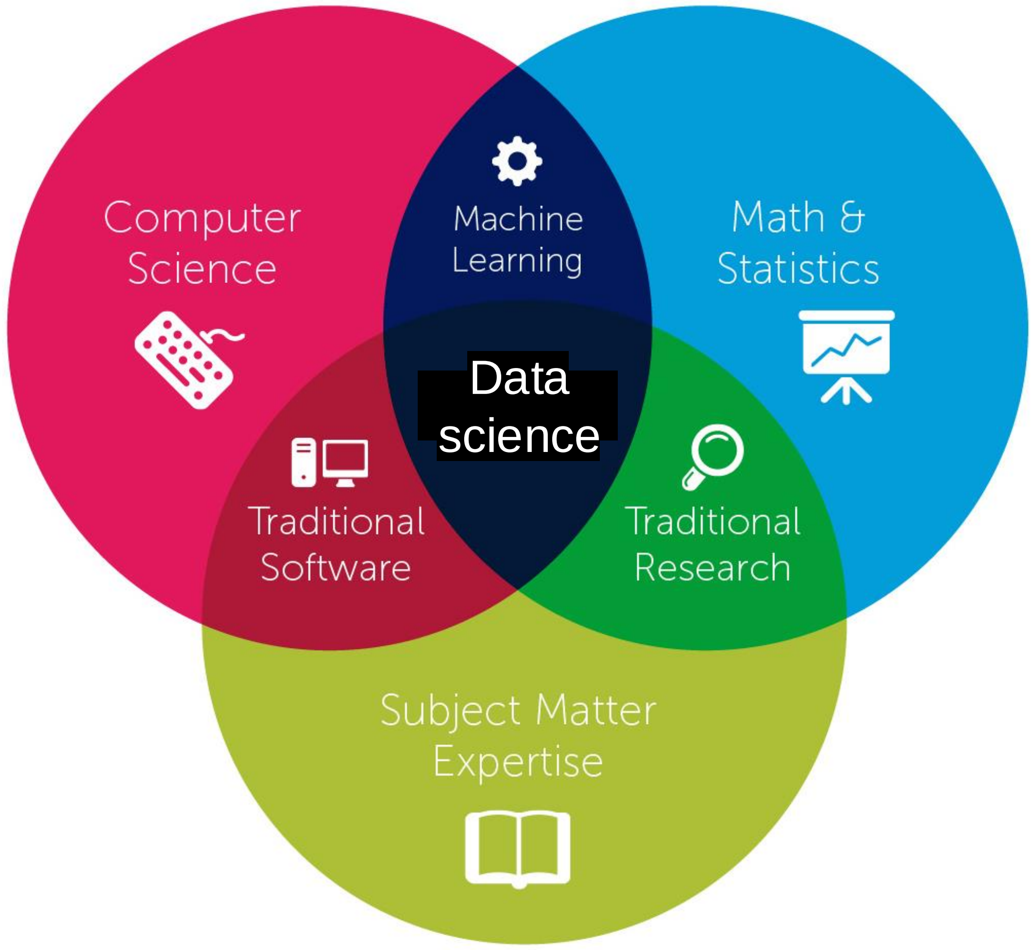
Fall 2024

Instructor: Assoc. Prof. Şener Özönder

Email: sener.ozonder@bogazici.edu.tr

Institute for Data Science & Artificial Intelligence

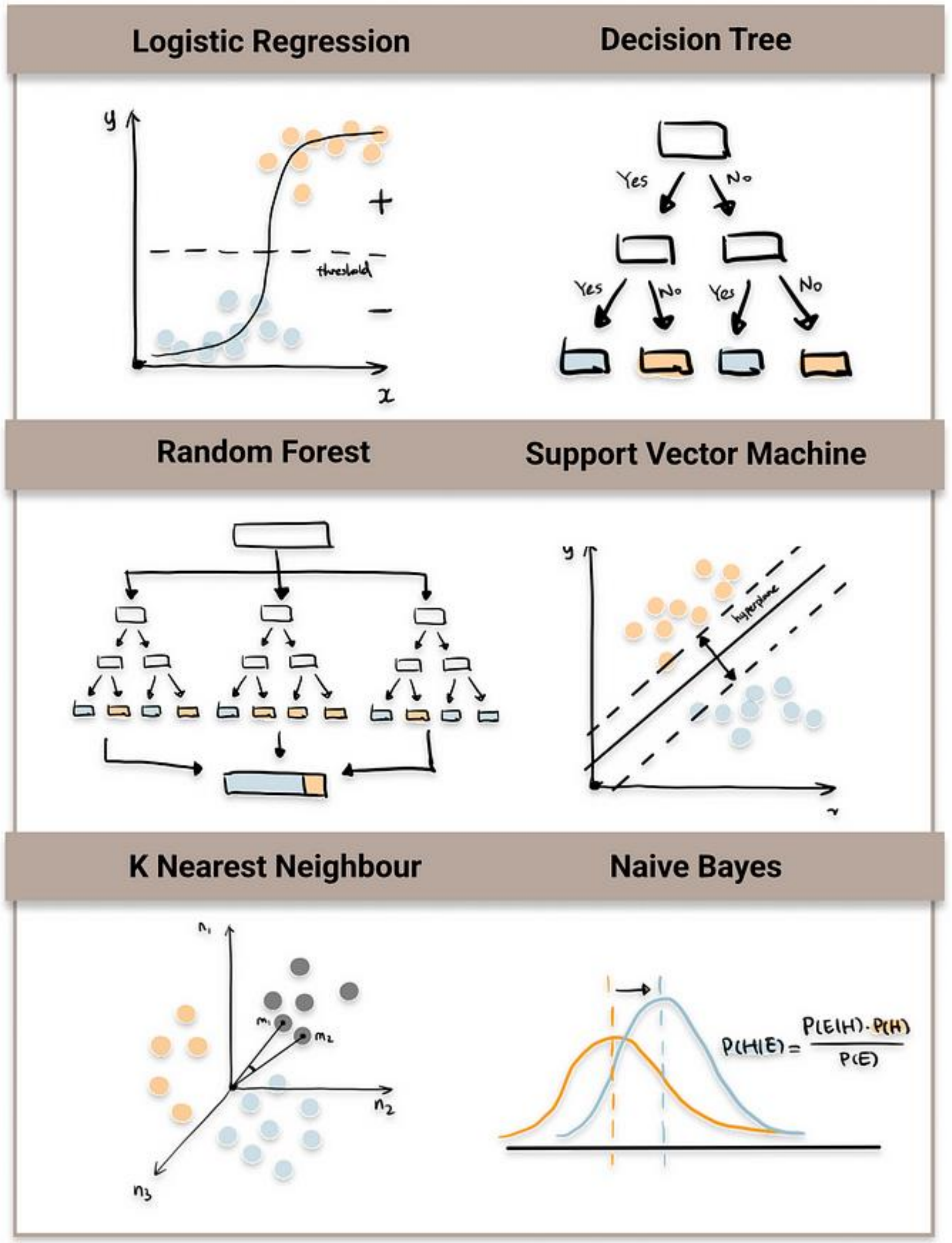
Boğaziçi University



Modeling: Classification

- Classification is a supervised learning approach where algoritms take data with several predictors and tries to assign a class to them.

- Sometimes the problem is binary classification (tumor is malignant or benign) and sometimes multiclass classification (cat, dog, monkey).

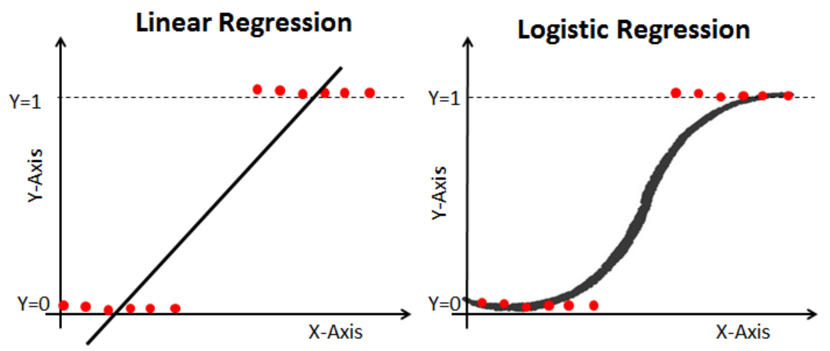


Logistic Regression

sigmoid/logistic function

$$f(x) = \frac{1}{1+e^{-(x)}}$$

- Linear regression is not a good choice if we want the output to be only 0 or 1, i.e., when the response variable is not continous but binary as in determining if a tumor is benign or malignant.



- Linear regression prediction can be any number in (-∞,+∞). For binary classification, we need to transform linear regression with a function so that the prediction is only between 0 and 1. One function that can do it is the sigmoid function.
- The logistic regression is a probabilistic model. Usually the threshold value of 0.5 is chosen. This equation gives the probability of a record label being 1. If $p(y=1) \geq 0.5$, then that record will be classified as 1, if $p(y=1) < 0.5$, it will be classified as 0.

$$p = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots)}}$$

Logistic Regression

- **Training:** Use an appropriate loss function and gradient descent algorithm to find optimal βs.

$$p(y=1)=\frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots)}}$$

y	x_1	x_2	x_3	x_4	x_5
diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
M 1	17.99	10.38	122.8	1001	0.1184
M 1	20.57	17.77	132.9	1326	0.08474
B 0	19.69	21.25	130	1203	0.1096
B 0	11.42	20.38	77.58	386.1	0.1425
M 1	20.29	14.34	135.1	1297	0.1003

- **Test:** Once you have βs, you have the model; now you can use it for prediction on the data that's not used during training. When you plug in the predictors (x_1, x_2 etc) values into the model, *p* will be some number between 0 and 1. If you get $p \geq 0.5$, we count it as 1, so this would mean that particular record belongs to a malignant tumor (assuming we chose the encoding M⟶1 and B⟶0).

Logistic Regression

- Why is logistic regression is not a «linear» model?

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

- Taylor expand around β_0 and β_1 around zero up to second power:

$$\text{In[32]:=}\ \mathbf{Normal@Series}\left[\frac{1}{1+\mathrm{E}^{-\beta 0+\beta 1\, x}},\ \{\beta 0,\ 0,\ 2\},\ \{\beta 1,\ 0,\ 2\}\right]\ \mathbf{//\ Expand}$$

$$\text{Out[32]=}\ \frac{1}{2}+\frac{\beta 0}{4}-\frac{x\,\beta 1}{4}+\frac{1}{16}\,x\,\boxed{\beta 0^2\,\beta 1}-\frac{1}{16}\,x^2\,\boxed{\beta 0\,\beta 1^2}$$

Nonlinear terms
(nonlinear in model parameters)

Logistic Regression

$$p(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots)}}$$

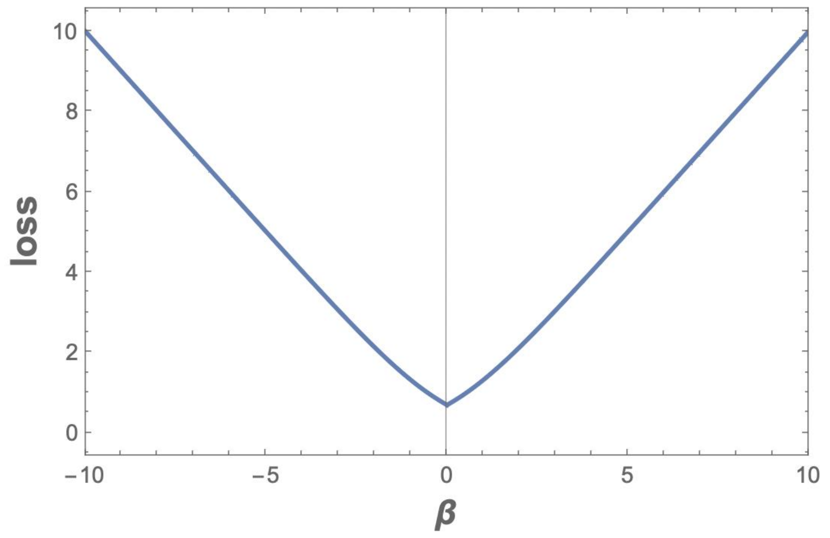
- For training to find βs, we need a loss (cost) function

loss fnc: $L(\beta) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$

- Since this loss function for logistic regression is convex (no local minima) wrt βs, we can use gradient descent algorithm. Iterate this equation until the loss function is minimum.

iterate until $L(\beta)$ is minimum

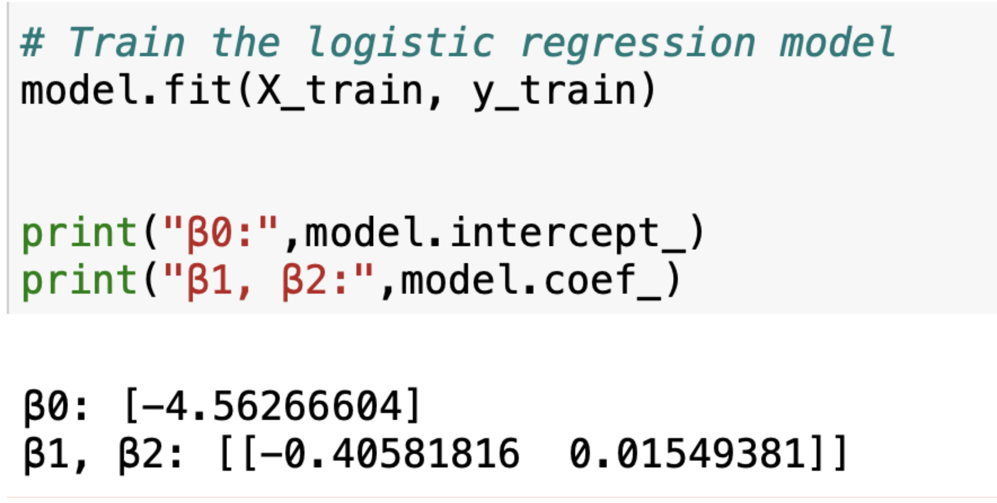
$$\beta_{j,new} = \beta_{j,old} - \alpha \frac{\partial L(\beta)}{\partial \beta_j}\bigg|_{\beta = \beta_{old}}$$



Logistic Regression

- Example: Consider the breast cancer data. Can we decide if a patient's tumor is benign or malignant based on only two predictors: **radius_mean** and **area_mean**? The predictors are numeric and we want the output to be binary (0/1). This problem can be approached with logistic regression modelling.
- Model fit result:

$$p(y=1) = \frac{1}{1+e^{-[(-4.56)+(-0.41)\ \textbf{radius\_mean}+0.02\ \textbf{area\_mean}]}}$$



```
# Train the logistic regression model
model.fit(X_train, y_train)


print("β0:",model.intercept_)
print("β1, β2:",model.coef_)
```

```
β0: [-4.56266604]
β1, β2: [[-0.40581816  0.01549381]]
```

- Now we can use this model on the records of new patients. Let's say you used a patient with **radius_mean**=11 and **area_mean**=350. You get $p(y=1)=0.11$. This is less than our threshold 0.5, so the prediction is 0, which means the tumor is benign.

Logistic Regression

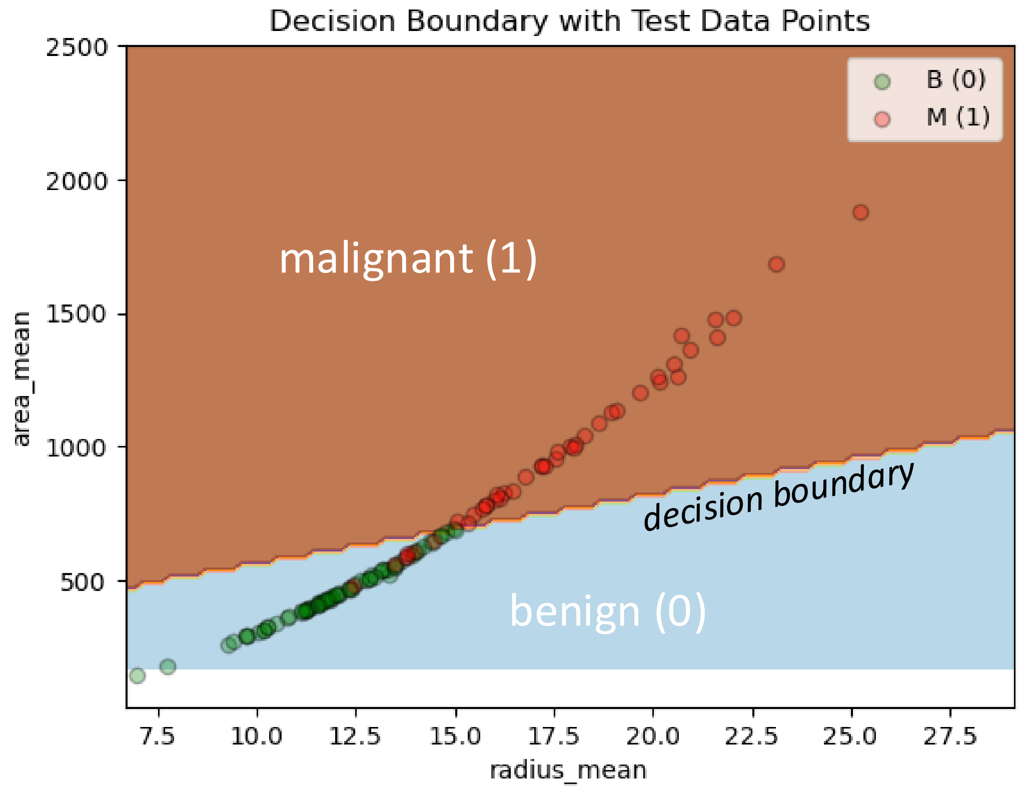
- Ok, we got the model. But how good is it? Is there something like R^2? Well, kinda. There are some definitions of pseudo-R^2 that you can use.
- But instead of them, we can randomly split the breast records as %80 (training data) and %20 (test data), train the model on the training data, then use the trained model *p*(*y*=1) on the test data to find the predictions $\hat{y}$. Then compare the predictions with the labels y (M/B) of the test data to decide on the performance (correctness) of the model.
- Model:

$$p(y=1) = \frac{1}{1 + e^{-[\,(-4.56) + (-0.41)\ \mathbf{radius\_mean} + 0.02\ \mathbf{area\_mean}]}}$$

- Before the performance metrics in classification, let's discuss decision boundary first. The boundary is given by the line equation

(−4.56) + (−0.41) **radius_mean** + 0.02 **area_mean = 0**

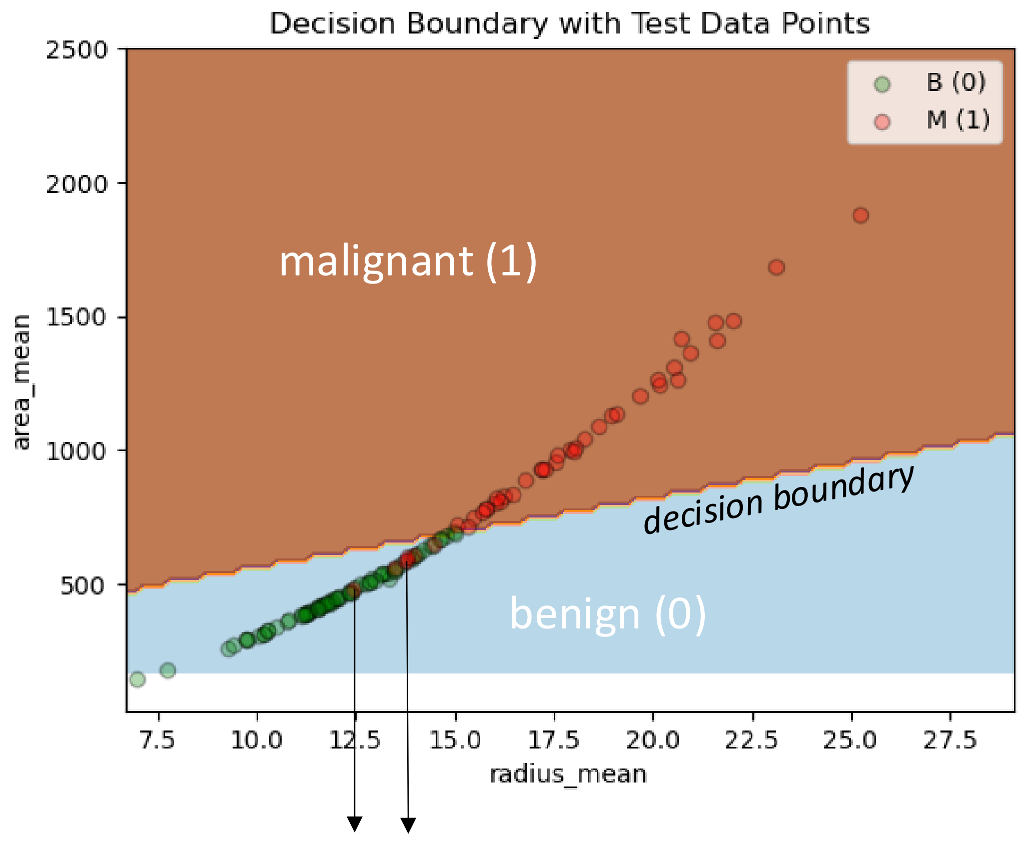
→ **area_mean**=(4.56 + 0.41 **radius_mean**)/0.02



Logistic Regression

- Red color of the dots means true value of diagnosis is malignant, green color means benign. The model is pretty good in separating two classes, but there are few red dots (patients with malignant tumor) fell in the blue (benign) area. These are false negatives (negative=0=benign, positive=1=malignant).

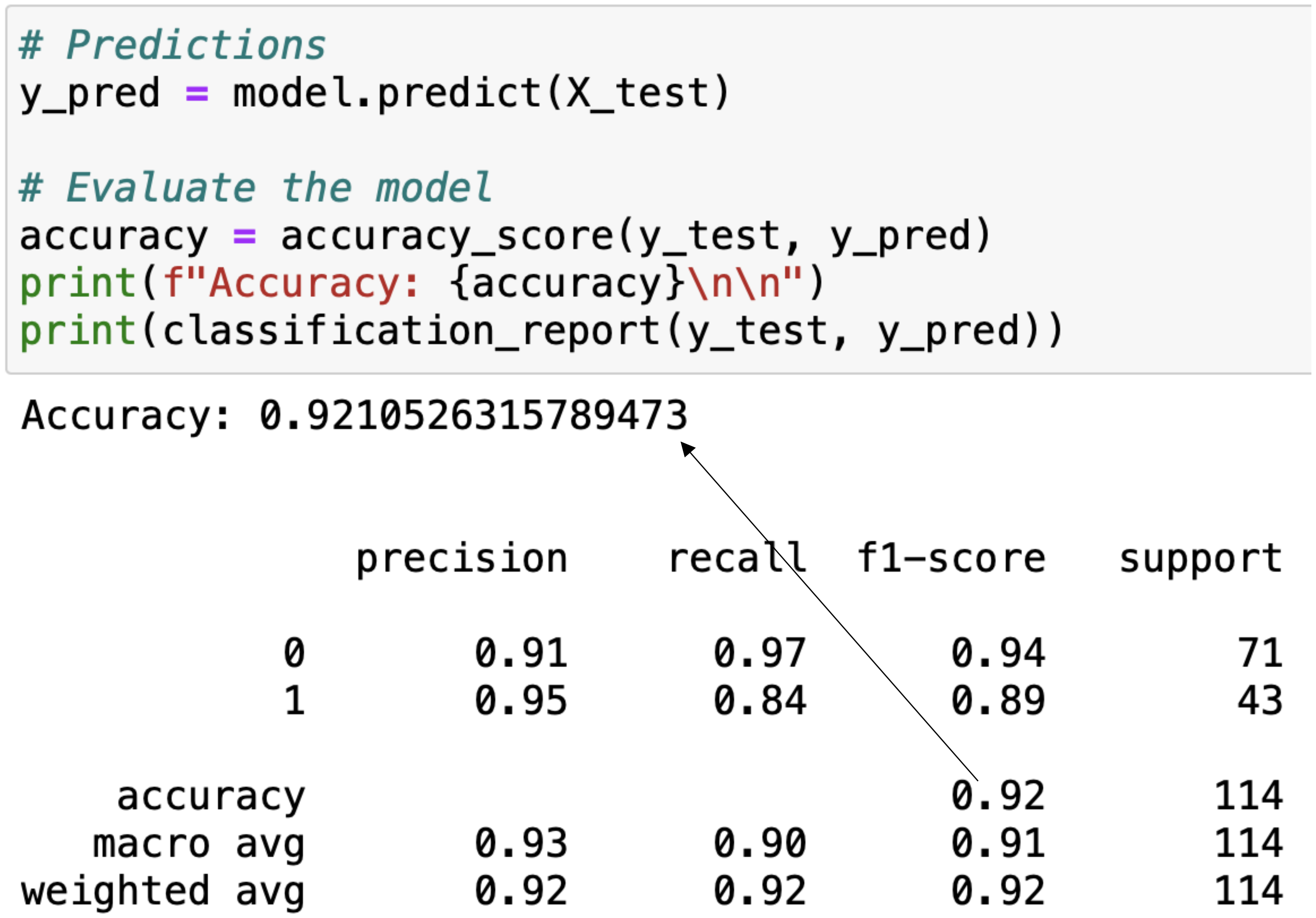
- What to do to improve the model? Maybe add polynomial terms, interaction terms etc. Or maybe just don't use logistic regression and switch to a more complex model like decision tree.



Red dots in the blue region:
A few malignant tumors classified as benign.

Logistic Regression

- Now let’s assess the model.
- Which metrics should we look at?



```python
# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}\n\n")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9210526315789473


              precision    recall  f1-score   support

           0       0.91      0.97      0.94        71
           1       0.95      0.84      0.89        43

    accuracy                           0.92       114
   macro avg       0.93      0.90      0.91       114
weighted avg       0.92      0.92      0.92       114
```

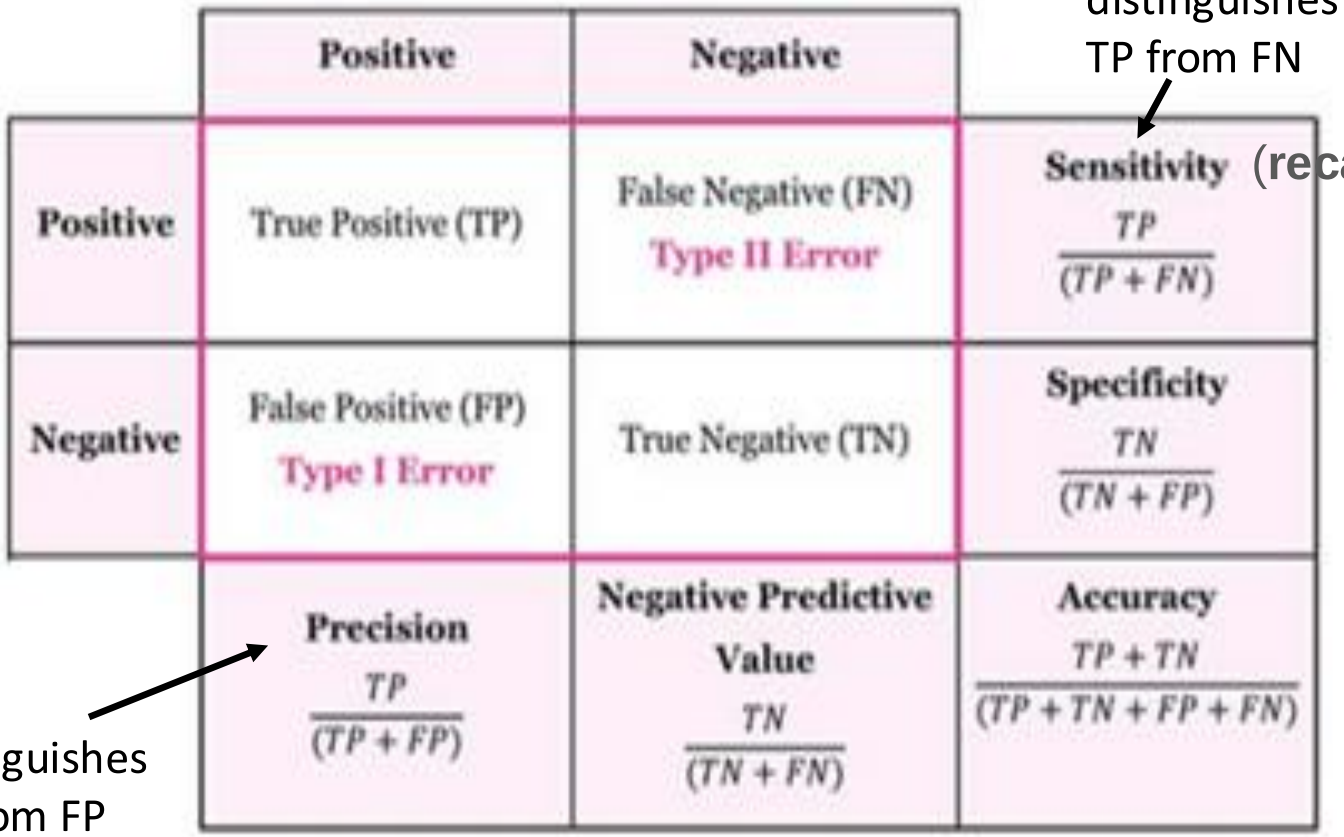
Logistic Regression

True positive: Tumor is malignant and model predicted **positive** ✓
False positive: Tumor is benign but model predicted **positive** ✗
True negative: Tumor is benign and model predicted **negative** ✓
False negative: Tumor is malignant but model predicted **negative** ✗

(H_0: not pregnant. H_a: pregnant)

Model/test prediction

Actual (data)



	Positive	Negative	
Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP+FN)}$
Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

distinguishes TP from FN

(recall)

distinguishes TP from FP

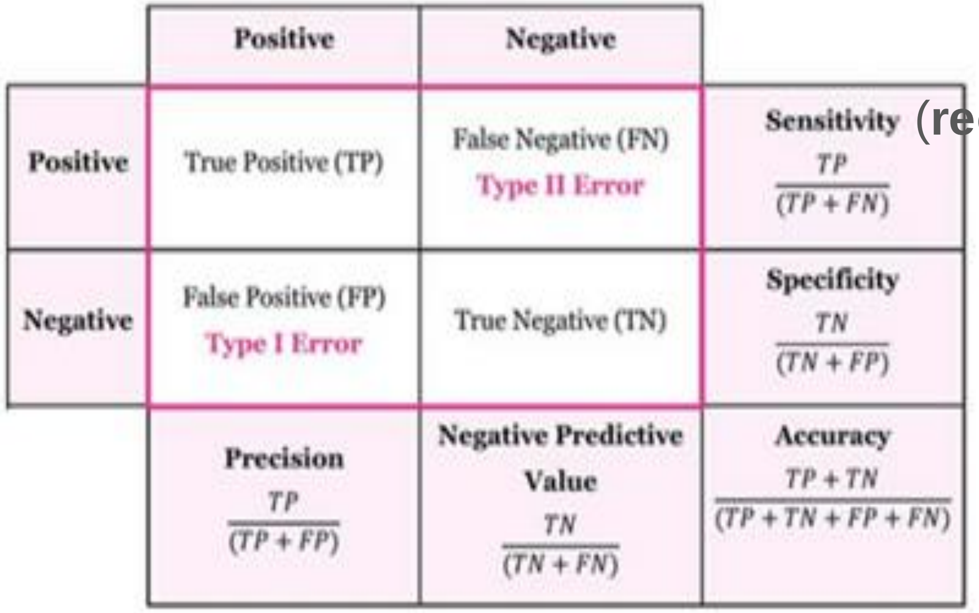
- You won't have experimental tests (like covid test) or statistical models (like logistic regression) where all metrics are maximized. There's a trade-off between the metrics.
- For example, you may want to increase precision by sacrificing some specificity, or vice verse.
- We'll discuss which metric becomes relatively more important in which situations in the following.

Logistic Regression

- **Precision** is the accuracy of catching true positives against false positives. TP: spam email labeled spam. FP: normal email labeled spam.
- **Sensitivity (recall)** is the metric of catching TP against FN. It’s central when the cost of a FN is high. In medical diagnostics for a severe disease (covid), you'd want to catch as many cases as possible (high recall), so few actual cases go undiagnosed (person is covid but test gives negative).

Model/test prediction

Actual (data)



	Positive	Negative	
Positive	True Positive (TP)	False Negative (FN) Type II Error	**Sensitivity** (**recall**) $\frac{TP}{(TP+FN)}$
Negative	False Positive (FP) Type I Error	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

- **F1-score** ($\frac{2}{\frac{1}{precision}+\frac{1}{recall}}$) is crucial when you seek a balance between precision and recall especially when FP and FN have a similar cost. Tries to catch TP against FP and FN in a balanced way. Good for spam filters.

- **Specificity** is important when it's crucial to identify TN against FP, as in PSA tests for prostate cancer. FP can lead to invasive biopsies, psychological stress and unnecessary medical interventions, all of which carry potential health risks and economic costs.

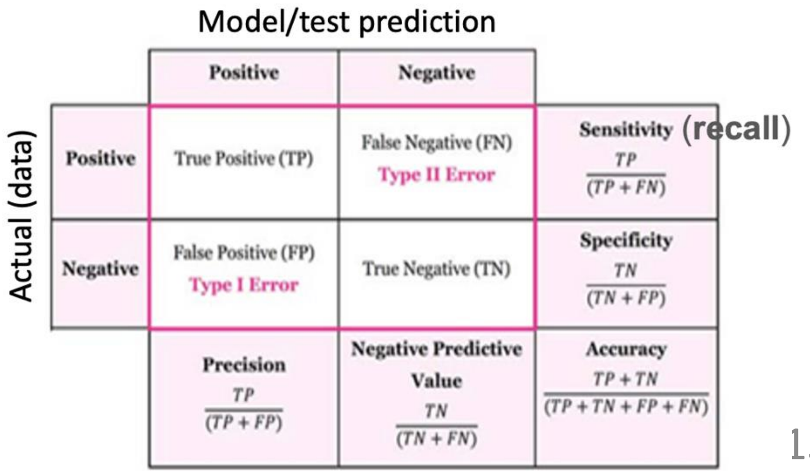
Logistic Regression

COVID-19 Test

- **Sensitivity (Recall) for Positive Case**: This is critical for infectious diseases like COVID-19, where failing to identify an infected person (false negative) can lead to the disease spreading to others. Hence, for a COVID-19 test, a high sensitivity is crucial to ensure that most people who are infected are correctly identified and quarantined.

- **Specificity**: While specificity is also important, in the context of covid tests it may be considered slightly less critical than sensitivity because a false positive (where the test indicates a person has the disease when they do not) would lead to self-isolation but not to further spread of the disease.

- So okay we have to use sensitivity metric for Covid19, but how? Well, during model selection, you will look at the sensitivity metric, not others, and take the model where sensitivity is the highest (for Covid19).



Model/test prediction

Actual (data)

	Positive	Negative	
Positive	True Positive (TP)	False Negative (FN) Type II Error	**Sensitivity** (**recall**) $\frac{TP}{(TP+FN)}$
Negative	False Positive (FP) Type I Error	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

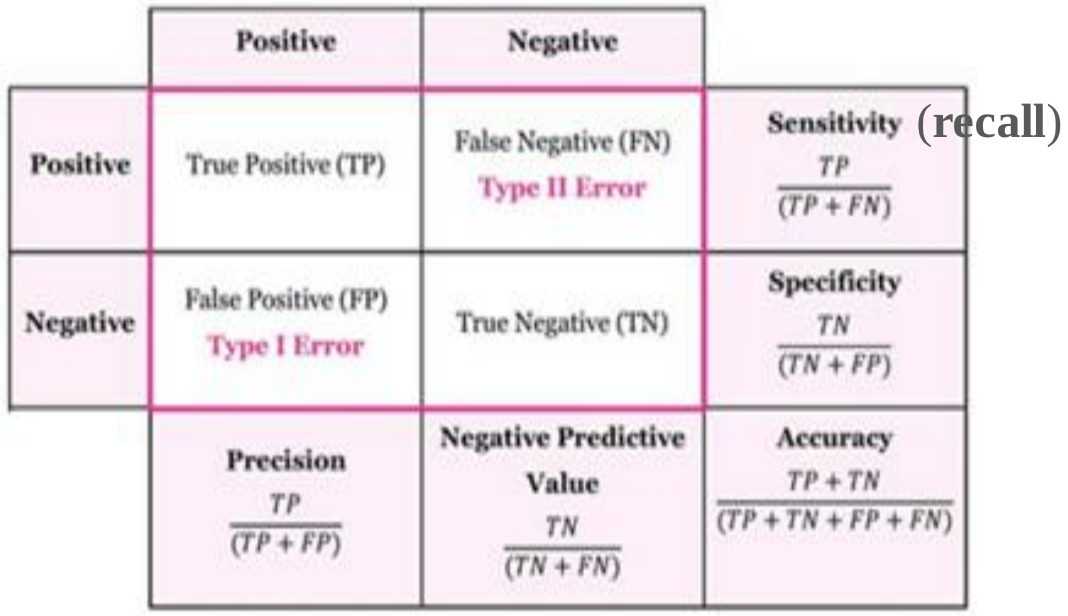
Logistic Regression

Situations When It's Okay to Have Lower Metrics:

- **When false positives are less critical**: For example, in preliminary diagnostic tests where a positive result leads to more specific testing, a lower specificity can be acceptable to ensure that all potential cases are forwarded for further testing. (mass screening tests)
- **When diseases have a low prevalence**: A slightly lower sensitivity may be acceptable if the disease is not immediately life-threatening, or if there are follow-up tests or procedures that can catch cases missed in the initial screening.

Model/test prediction

Actual (data)



	Positive	Negative	
Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity (recall) $\frac{TP}{(TP+FN)}$
Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

When It's Costly to Have Low Metrics:

- **High-Stakes Diagnosis**: For diseases with significant mortality or morbidity risk if untreated (e.g., cancer, heart disease), both high sensitivity and high specificity are crucial to correctly diagnose those with the disease and to avoid false positives that could lead to invasive and risky procedures.
- **Highly Infectious Diseases**: In the case of highly infectious diseases, low sensitivity could be very costly as it may result in missed cases and wider spread of the disease.

Logistic Regression

- Specificity is missing in the output of **classification_report()**. We can print the confusion matrix and manually calculate the specificity.

```python
# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}\n\n")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9210526315789473
```

=(bening_accuracy + malignant_accuracy)/2

(number of instances)

```
                 precision    recall  f1-score   support

benign      0         0.91      0.97      0.94        71
malignant   1         0.95      0.84      0.89        43

       accuracy                           0.92       114
      macro avg       0.93      0.90      0.91       114
   weighted avg       0.92      0.92      0.92       114
```

```python
cm = confusion_matrix(y_test, y_pred, labels=[1,0])
cm
```

```
array([[36,  7],
       [ 2, 69]])
```

```python
tp, fn, fp, tn = cm.ravel()
print("tp, fn, fp, tn: ", tp, "," ,fn,",", fp,"," ,tn)
```

```
tp, fn, fp, tn:  36 , 7 , 2 , 69
```

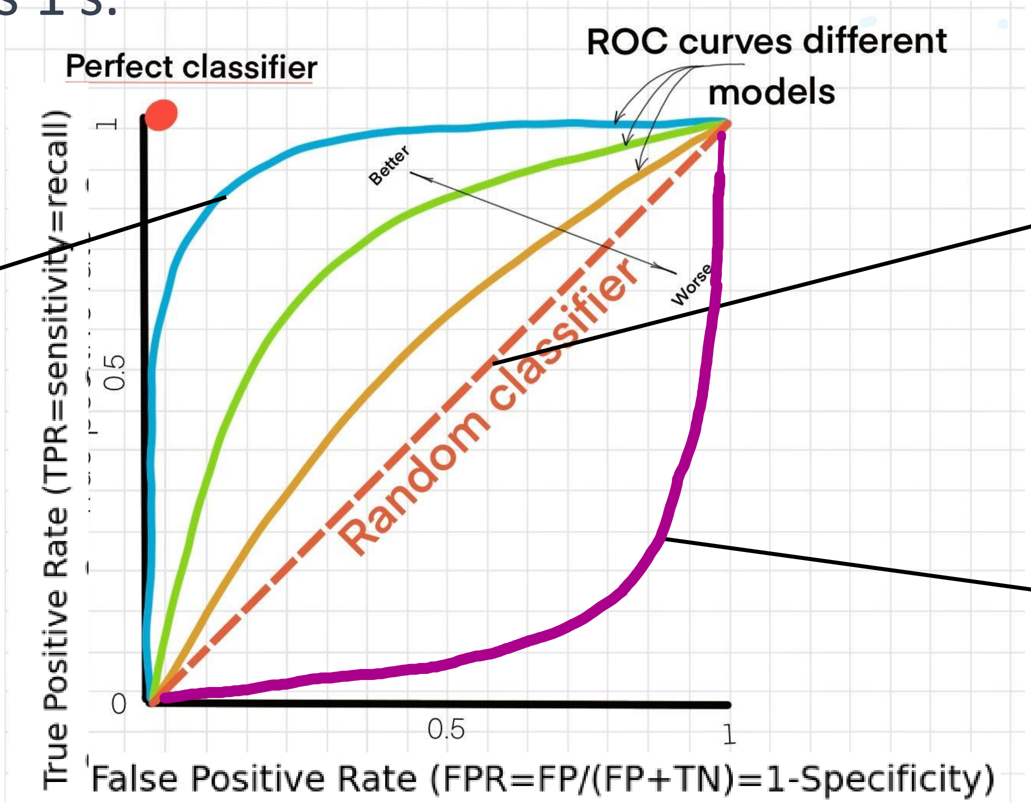
```python
specificity = tn / (tn + fp)
print(f"Specificity: {specificity}")
precision = tp / (tp + fp)
print(f"Precision: {precision}")
sensitivity = tp / (tp + fn)
print(f"Sensitivity (recall): {sensitivity}")
```

```
Specificity: 0.971830985915493
Precision: 0.9473684210526315
Sensitivity (recall): 0.8372093023255814
```

The formulas give you the metrics listed where "1" is.

Logistic Regression: ROC curve

- The ROC curve illustrates the balance between TPR (True Positive Rate) and FPR (False Positive Rate, which relates to false positives).
- If you have multiple classifiers (classification models), choose the one whose ROC curve gets closest to the top left, which is marked as perfect classifier).
- High ROC curve → high area under curve (AUC) → Model is mostly predicting 0's as 0's and 1's as 1's.



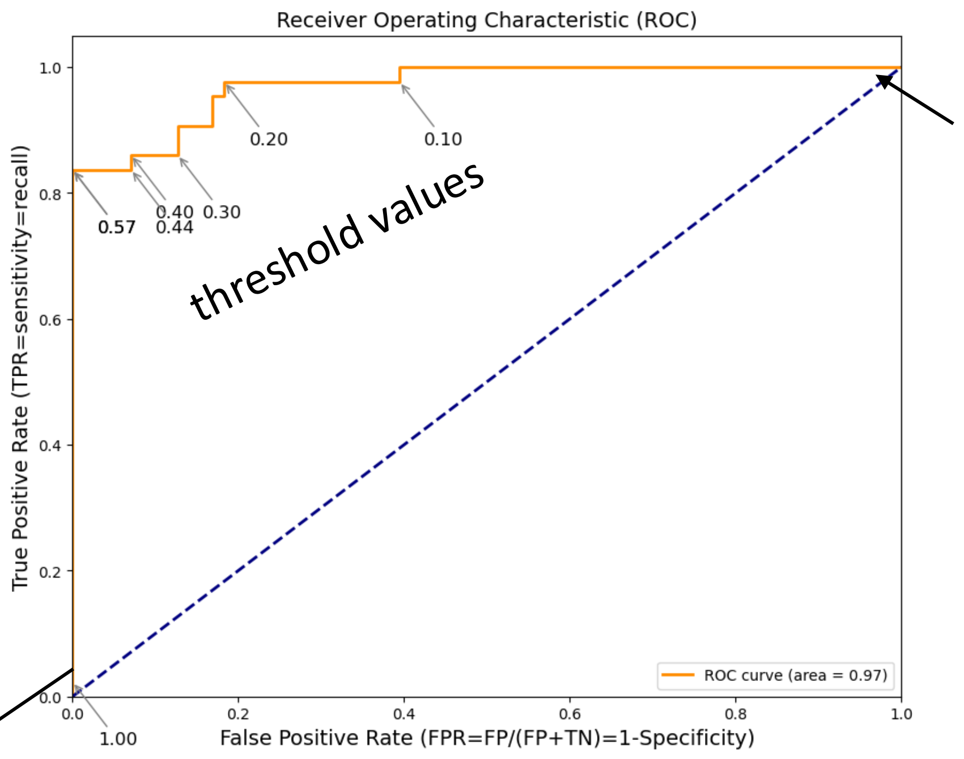
Best model's ROC curve is this one.

If the ROC curve is exactly on this red dashed line, then the model has no discrimination ability (i.e. it makes random guesses).

This model predicsts malignants as benign and benigns as malignant. It's worse than random classifier. If you get a curve below random classifier, this indicates the labels in the test data somehow got inverted!

Logistic Regression

- Below is the ROC curve for the logistic regression model for two predictors, **radius mean** and **area mean** for different treshold values (previously we had chosen it to be 0.5).
- This orange ROC curve is created by graphing the TPR against the FPR across various thresholds. Reducing the threshold for classifying observations as positive leads to a rise in True Positives, but also inadvertently increases False Positives. Trade-off...



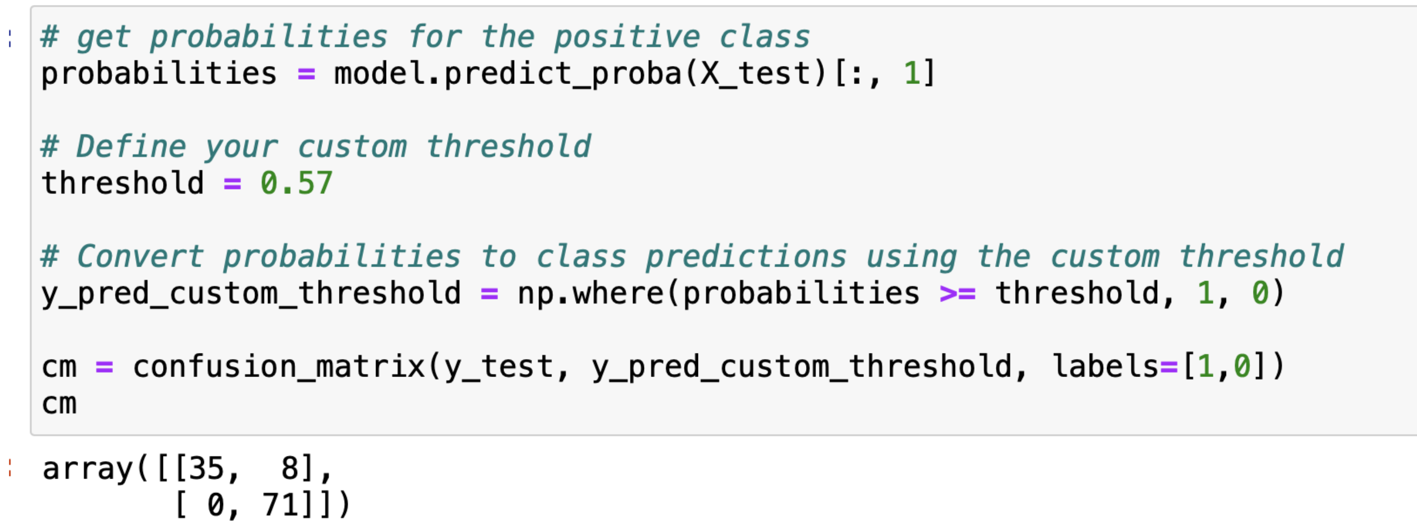
threshold=0. The model will classify every record as positive (=1=maglinant). So sensitivity is 1, specificity is zero (=FPR is one). not good.

threshold = 1. The model will classify every record as negative (=0=benign) not good.

- This curve is created by graphing the TPR against the FPR across various thresholds. Reducing the threshold for classifying observations as positive leads to a rise in True Positives, but also inadvertently increases False Positives.
- ROC curve for the logistic regression model for two predictors, **radius mean** and **area mean**.

Logistic Regression

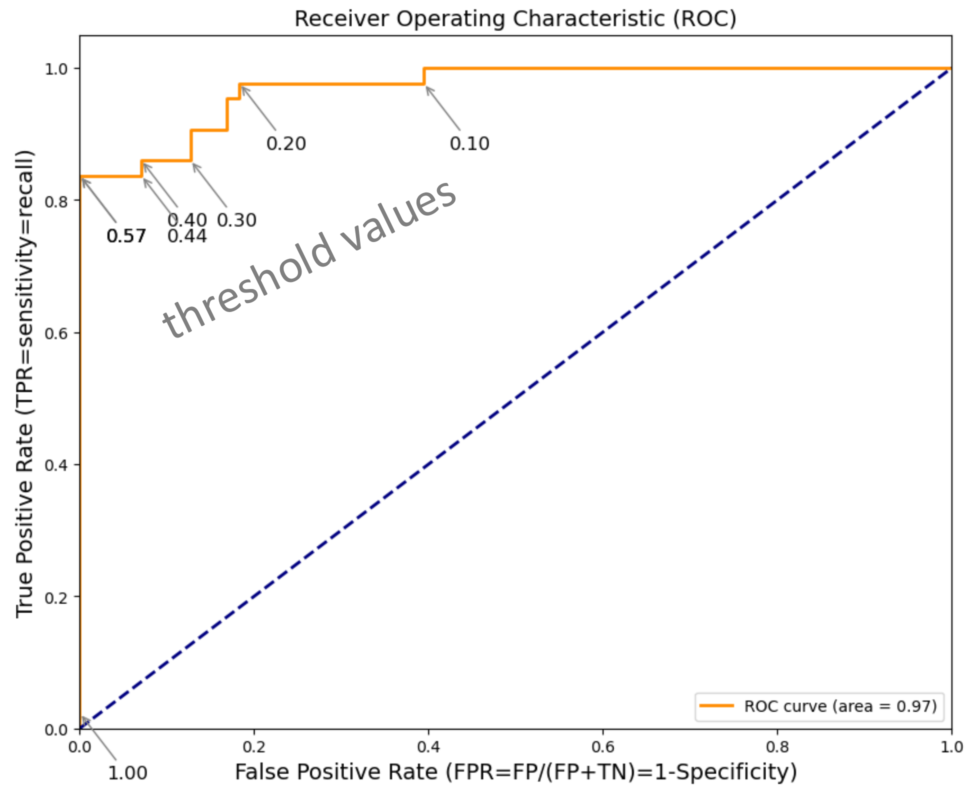
- 0.57 yields zero false positive (bottom left corner of the confusion matrix; see the code below).



```
# get probabilities for the positive class
probabilities = model.predict_proba(X_test)[:, 1]

# Define your custom threshold
threshold = 0.57

# Convert probabilities to class predictions using the custom threshold
y_pred_custom_threshold = np.where(probabilities >= threshold, 1, 0)

cm = confusion_matrix(y_test, y_pred_custom_threshold, labels=[1,0])
cm
```

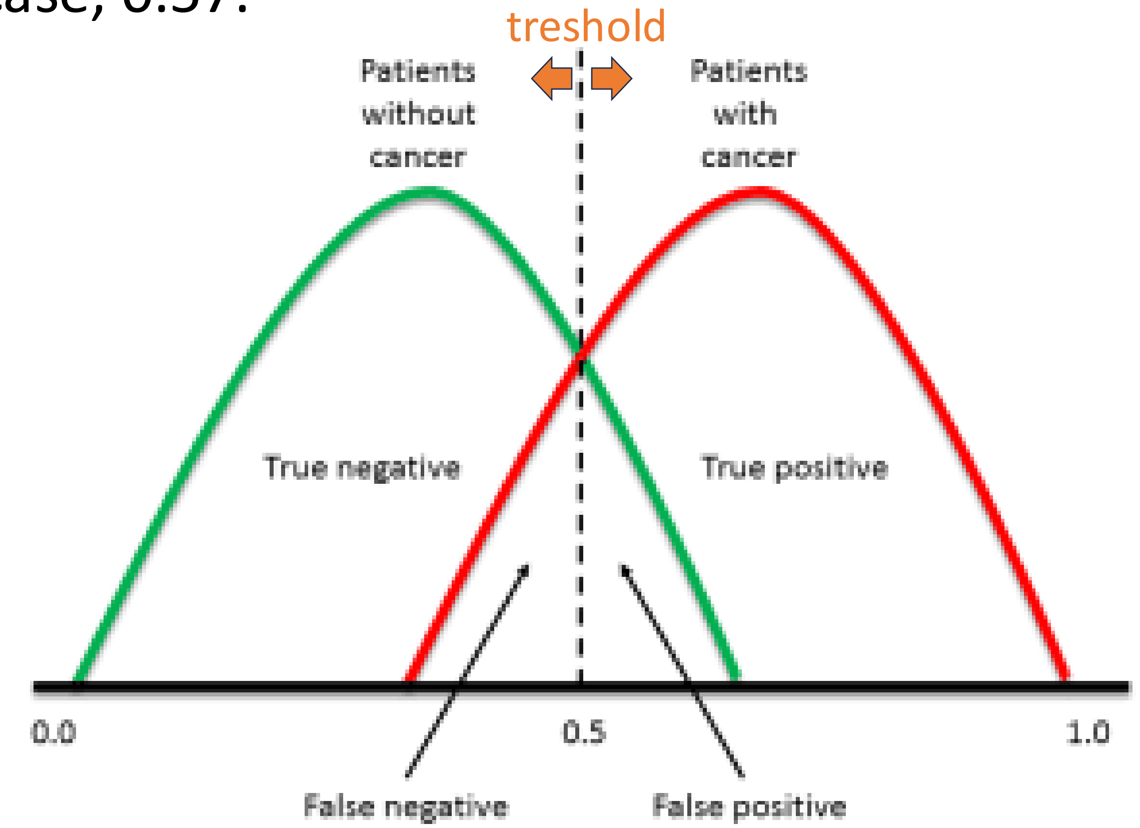
```
array([[35,  8],
       [ 0, 71]])
```

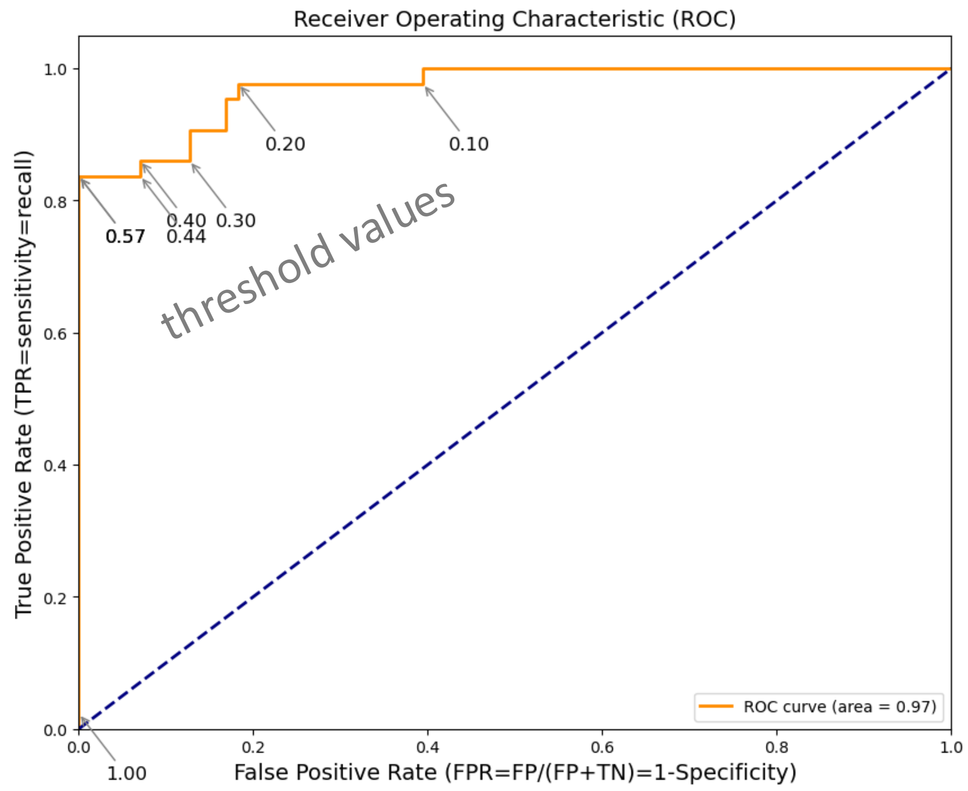
- You may want more true positives at the expense of a few false positives. In that case, you may lower the threshold. Looking at the ROC curve, lowering it down to 0.2 is possible. How lower can you go in the threshold? You decide it; as go lower in that you'll get more true positives at the expense of more false positives. So you decide how much false positives you can tolarate depending on the problem at hand.



Logistic Regression

- Take COVID-19 testing as an example: prioritizing a higher detection rate for true positives is crucial, even if it leads to a rise in false positives. The impact of false positives is relatively minor, generally resulting in self-isolation, but missing true positives could have significant health consequences. So, for covid, move threshold to lower values, in this case, 0.57.


treshold
Patients without cancer
Patients with cancer
True negative
True positive
0.0
0.5
1.0
False negative
False positive


Receiver Operating Characteristic (ROC)
True Positive Rate (TPR=sensitivity=recall)
False Positive Rate (FPR=FP/(FP+TN)=1-Specificity)
threshold values
ROC curve (area = 0.97)