AI200: APPLIED MACHINE LEARNING

MODEL EVALUATION FOR CLASSIFICATION MODELS

MODEL EVALUATION METRICS: CONFUSION MATRIX

- For all classification models, we would use the **Confusion Matrix** (a collation of model evaluation metrics) to aid in our evaluation of model performance

- For example, after we use a classification model to predict whether 10 people would have diabetes or not, we would collate the results a **Confusion Matrix** as shown below

Predicted by ML Models

S/N	Glucose	BMI	Actual Values	Predicted Values
1	148	33.6	1	1
2	85	26.6	0	0
3	183	23.3	1	1
4	89	28.1	0	0
5	137	43.1	1	1
6	116	25.6	0	0
7	78	31.0	1	1
8	115	35.3	0	1
9	197	30.5	1	1
10	125	0.0	1	0

Predicted Values	Actual Values: Positive (1)	Actual Values: Negative (0)
Positive (1)	**True Positive** 5	**False Positive** 1
Negative (0)	**False Negative** 1	**True Negative** 3

- Using the confusion matrix, we can compute metrics (the most used is <u>accuracy</u>, but we will cover several others) to evaluate the model.

MODEL EVALUATION METRICS: CONFUSION MATRIX (ANALOGY)

- Let's use a memorable analogy to help us remember the intuition of the confusion matrix
- Imagine we created a classification model to predict whether 10 people are pregnant or not, and after the model churned out its predictions, we want to see how well the model performed

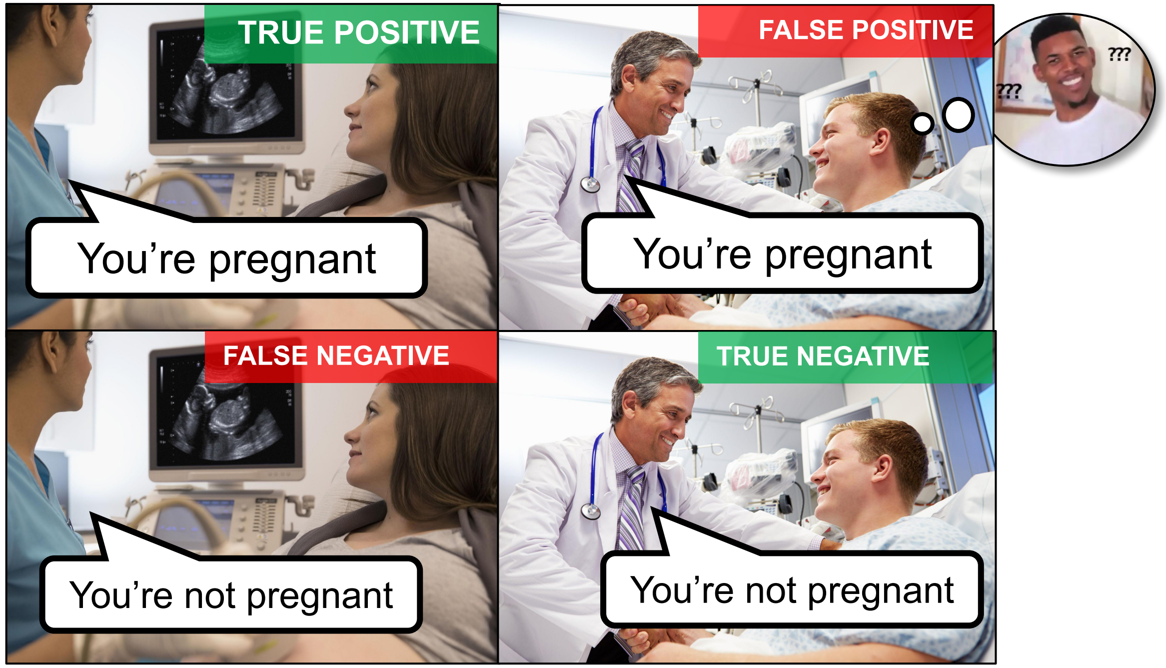
Actual Values

1 **0**

Predicted Values

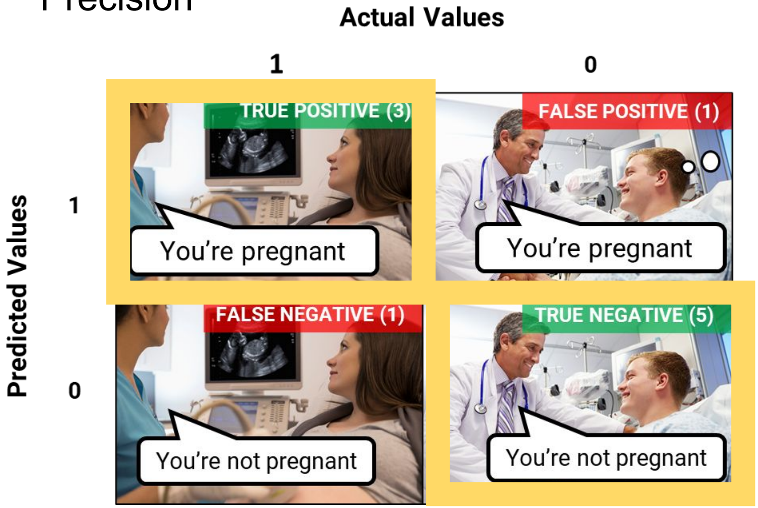
1

0



MODEL EVALUATION METRICS: ACCURACY

- Using the confusion matrix, we can compute metrics to evaluate the performance of classification models:
  - Accuracy
  - Error Rate
  - Sensitivity / Recall
  - Specificity
  - Precision



True Negative (TN):
- The model predicted that the person is not pregnant
- The person indeed is not pregnant
- Therefore, the model's **prediction of a negative outcome was true**

True Positive (TP):
- The model predicted that the person is pregnant
- The person is indeed pregnant
- Therefore, the model's **prediction of a positive outcome was true**

Accuracy of model: Overall, how often is the classifier correct?

$$\frac{TN+TP}{Number\ of\ Predictions\ Generated} = \frac{TN+TP}{TN+TP+FN+FP}$$

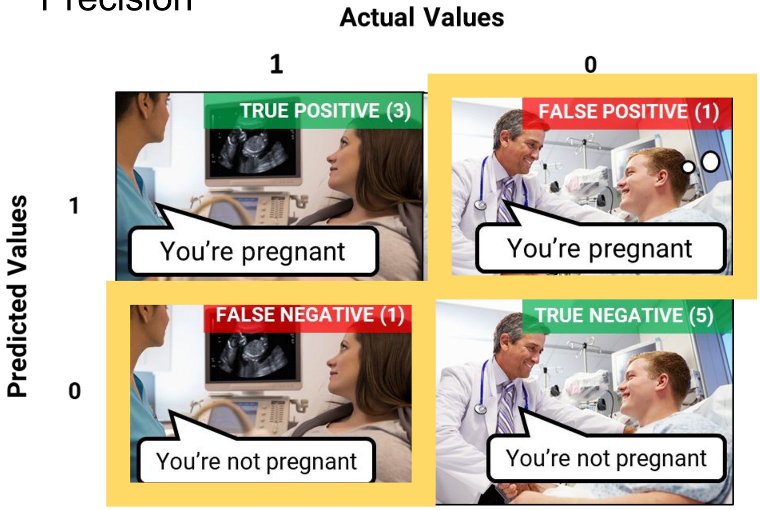
$$= \frac{3+5}{3+5+1+1}$$

$$= 0.8$$

This is actually very intuitive. If you think about it, we are just dividing the total number of instances our model was correct, by the total number of predictions. In this example, the accuracy rate of our diabetes prediction model is 80%.

MODEL EVALUATION METRICS: ERROR RATE

- Using the confusion matrix, we can compute metrics to evaluate the performance of classification models:
  - Accuracy
  - Error Rate
  - Sensitivity / Recall
  - Specificity
  - Precision



False Negative (FN), aka Type II Error:
- The model predicted that the person is not pregnant
- But the person is actually pregnant
- Therefore, the model's **prediction of a negative outcome was false**

False Positive (FP), aka Type I Error:
- The model predicted that the person is pregnant
- But the person is not pregnant (c'mon, he is a guy) ☹
- Therefore, the model's **prediction of a positive outcome was false**

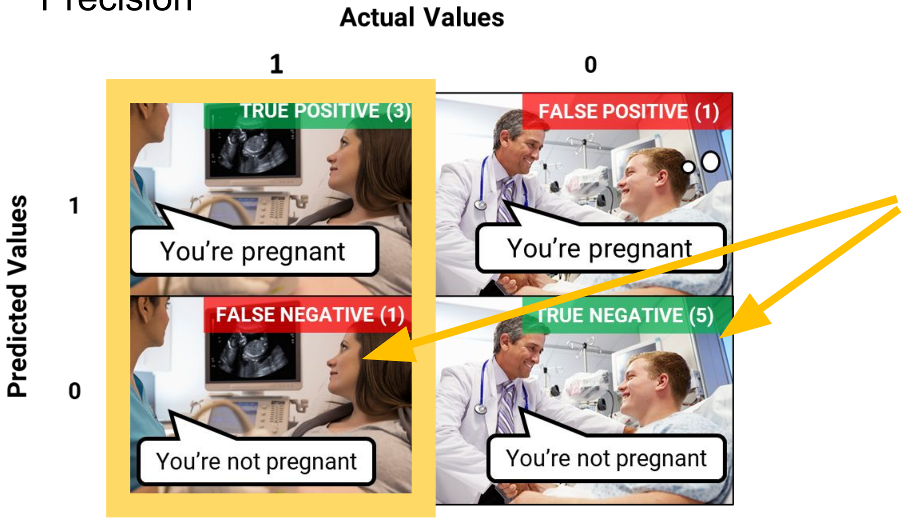
Error Rate of model: Overall, how often is the classifier wrong?

$$\frac{FN+FP}{Number\ of\ Predictions\ Generated} = \frac{FN+FP}{TN+TP+FN+FP} = \frac{1+1}{3+5+1+1} = 0.2$$

This is actually very intuitive. If you think about it, we are just dividing the total number of instances our model was wrong, by the total number of predictions. In this example, the classification error rate of our diabetes prediction model is 20%.

MODEL EVALUATION METRICS: SENSITIVITY

- Using the confusion matrix, we can <u>compute metrics</u> to evaluate the performance of classification models:
  - Accuracy
  - Error Rate
  - Sensitivity / Recall
  - Specificity
  - Precision



Given that we have a model that only predicts zeros, all data points would either be **False Negatives** or True Negatives.

Thus, Sensitivity: TP / (TP + FN)
= 0 / (0 + FN) = 0 if there are false negatives.

This metric provides additional information beyond "accuracy", which does not distinguish between Type I and Type II errors.

Sensitivity / Recall of model: When the actual outcome is positive, how often is the prediction correct?

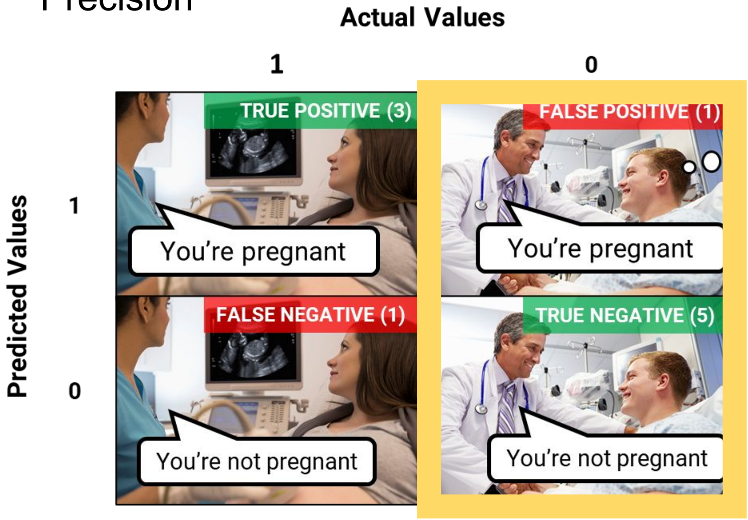
$$\frac{TP}{Number\ of\ Actual\ Positive\ Outcomes} = \frac{TP}{TP+FN} = \frac{5}{5+1} = 0.83333$$

Sensitivity measures how effective is the classifier in detecting positive instances, also known as **"True Positive Rate" or "Recall".**

For **use cases like cancer diagnosis**, this metric is very important, because any case which you falsely diagnose as benign is another potential life lost.

MODEL EVALUATION METRICS: SPECIFICITY

- Using the confusion matrix, we can compute metrics to evaluate the performance of classification models:
  - Accuracy
  - Error Rate
  - Sensitivity / Recall
  - Specificity
  - Precision



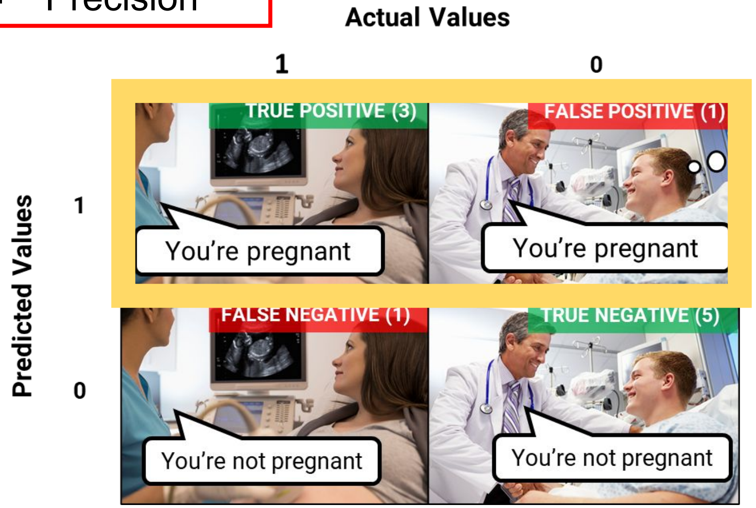
Specificity of model: When the actual outcome is negative, how often is the prediction correct?

$$\frac{TN}{Number\ of\ Actual\ Negative\ Outcomes} = \frac{TN}{TN+FP} = \frac{3}{3+1} = 0.75$$

Specificity measures how effective the classifier is in detecting negative cases.

MODEL EVALUATION METRICS: PRECISION

- Using the confusion matrix, we can compute metrics to evaluate the performance of classification models:
  - Accuracy
  - Error Rate
  - Sensitivity / Recall
  - Specificity
  - Precision



Precision of model: When the predicted outcome is positive, how often is the prediction correct?

$$\frac{TP}{Number\ of\ Actual\ Negative\ Outcomes} = \frac{TP}{TP+FP}$$

$$= \frac{5}{5+1}$$

$$=0.83333$$

This metric measures how relevant the positive instances flagged by the classifier are to the user -- are they ***True Positives***, or ***False Positives***?

MODEL EVALUATION METRICS: HOW TO COLLECT THEM?

- To collect the metrics to measure the performance of the model, we can use the two procedures which was covered during our previous lesson on Regression problems:

  - Train-test split
  - Cross Validation

- The only difference is that we calculate accuracy (or other metrics) on validation datasets, instead of the RMSE.

LOGISTIC REGRESSION

MODEL EVALUATION PROCEDURE (SELF-READING)

** This section was already covered in Lesson 3, but is added here explicitly so that you understand that the same techniques apply to classification problems as well.*

MODEL EVALUATION / ASSESSING MODEL ACCURACY: TRAIN-TEST SPLIT

- Train-test split is a technique that estimates a model's test error by leaving out a subset of provided data during training / fitting process and using this subset as a test dataset. The specific steps are:

  - **Step 1**: Randomly divide available data into two parts (usual norm for splitting is 70-30):
    - **Training** set
    - **Validation** set
  - **Step 2**: Use the **training set** to fit the model
  - **Step 3**: Generate predictions for the **validation set** with the model
  - **Step 4**: Calculate the accuracy for the model based on the **validation set**

Age	Marital Status	Monthly Income	...	Job Satisfaction	...	Years at Company	Attrition
33	Single	4400	...	4	...	5	0
37	Married	3300	...	4	...	2	1
...							
27	Married	3200	...	3	...	1	0
...							
25	Single	3000	...	3	...	1	0

Instances/ Observations

Features / Attributes / Input Variables

Class label / Target Variable

MODEL EVALUATION / ASSESSING MODEL ACCURACY: TRAIN-TEST SPLIT

- Train-test split is a technique that estimates a model's test error by leaving out a subset of provided data during training / fitting process and using this subset as a test dataset. The specific steps are:

  - **Step 1**: Randomly divide available data into two parts (usual norm for splitting is 70-30):
    - **Training** set
    - **Validation** set
  - **Step 2**: Use the **training set** to fit the model
  - **Step 3**: Generate predictions for the **validation set** with the model
  - **Step 4**: Calculate the accuracy for the model based on the **validation set**

Age	Marital Status	Monthly Income	...	Job Satisfaction	...	Years at Company	Attrition
33	Single	4400	...	4	...	5	0
37	Married	3300	...	4	...	2	1
...							
27	Married	3200	...	3	...	1	0
...							
25	Single	3000		3	...	1	0

X_train

y_train

X_test

y_test

Instances/ Observations

Features / Attributes / Input Variables

Class label / Target Variable

MODEL EVALUATION / ASSESSING MODEL ACCURACY: CROSS VALIDATION

- Cross Validation is a is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold **cross-validation.** The specific steps are:
  - **Step 1**: Randomly split the dataset into K equal partitions or 'folds'
  - **Step 2**: Use **Fold 1 as validation set**, and the rest of the **other folds as training data**
  - **Step 3**: Fit the model on the **training set** and estimate the model's accuracy using the **validation set**
  - **Step 4**: Repeat step 2-3 **using a different fold as validation set** at each iteration
  - **Step 5**: Take the average error as the estimate of test error

Instances/ Observations

Age	Marital Status	Monthly Income	...	Job Satisfaction	...	Years at Company	Attrition
33	Single	4400	...	4	...	5	0
37	Married	3300	...	4	...	2	1
...							
27	Married	3200	...	3	...	1	0
...							
25	Single	3000	...	3	...	1	0

Features / Attributes / Input Variables

Class label / Target Variable

MODEL EVALUATION / ASSESSING MODEL ACCURACY: CROSS VALIDATION

- Cross Validation is a is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold **cross-validation.** The specific steps are:

  - **Step 1**: Randomly split the dataset into K equal partitions or 'folds'
  - **Step 2**: Use **Fold 1 as validation set**, and the rest of the **other folds as training data**
  - **Step 3**: Fit the model on the **training set** and estimate the model's accuracy using the **validation set**
  - **Step 4**: Repeat step 2-3 **using a different fold as validation set** at each iteration
  - **Step 5**: Take the average error as the estimate of test error

Age	Marital Status	Monthly Income	...	Job Satisfaction	...	Years at Company	Attrition
33	Single	4400	Fold-1	4	...	5	Fold-1
37	Married	3300	Fold-2	4	...	2	Fold-2
...							
27	Married	3200	…	3	...	1	0 …
...							
25	Single	3000	Fold-k	3	...	1	Fold-k

If k = 10, that means each fold contains 10% of the data, where the entire dataset is split into 10 equal parts

MODEL EVALUATION / ASSESSING MODEL ACCURACY: CROSS VALIDATION

- <u>Cross Validation</u> is a is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold **cross-validation.** The specific steps are:

  - **Step 1**: Randomly split the dataset into K equal partitions or ‘folds’
  - **Step 2**: Use **Fold 1 as validation set**, and the rest of the **other folds as training data**
  - **Step 3**: Fit the model on the **training set** and estimate the model’s accuracy using the **validation set**
  - **Step 4**: Repeat step 2-3 **using a different fold as validation set** at each iteration
  - **Step 5**: Take the average error as the estimate of test error

Age	Marital Status	Monthly Income	...	Job Satisfaction	...	Years at Company	Attrition
X_test							y_test
37	Married	3300	...	4	...	2	1
...							
27	Married	3200	X_train		...	1	y_train
...							
25	Single	3000	...	3	...	1	0

We would use fold-1 as the test data set, and calculate the RSME based on it

MODEL EVALUATION / ASSESSING MODEL ACCURACY: CROSS VALIDATION

- Cross Validation is a is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold **cross-validation.** The specific steps are:
  - **Step 1**: Randomly split the dataset into K equal partitions or ‘folds’
  - **Step 2**: Use **Fold 1 as validation set**, and the rest of the **other folds as training data**
  - **Step 3**: Fit the model on the **training set** and estimate the model’s accuracy using the **validation set**
  - **Step 4**: Repeat step 2-3 **using a different fold as validation set** at each iteration
  - **Step 5**: Take the average error as the estimate of test error

Age	Marital Status	Monthly Income	...	Job Satisfaction	...	Years at Company	Attrition
33	Single	4400	X_train		...	5	y_train
X_test							y_test
...							
27	Married	3200	...	3	...	1	0
			X_train				y_train
...							
25	Single	3000	...	3	...	1	0

We repeat this with the subsequent fold.

MODEL EVALUATION / ASSESSING MODEL ACCURACY: CROSS VALIDATION

- Cross Validation is a is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold **cross-validation.** The specific steps are:
  - **Step 1**: Randomly split the dataset into K equal partitions or 'folds'
  - **Step 2**: Use **Fold 1 as validation set**, and the rest of the **other folds as training data**
  - **Step 3**: Fit the model on the **training set** and estimate the model's accuracy using the **validation set**
  - **Step 4**: Repeat step 2-3 **using a different fold as validation set** at each iteration
  - **Step 5**: Take the average error as the estimate of test error

Age	Marital Status	Monthly Income	...	Job Satisfaction	...	Years at Company	Attrition
33	Single	4400	...	4	...	5	0
37	Married	3300	...	4	...	2	1
...		X_train					y_train
27	Married	3200	...	3	...	1	0
...							
		X_test					y_test

And we keep doing this until each fold has served as the validation dataset

MODEL EVALUATION / ASSESSING MODEL ACCURACY: CROSS VALIDATION

- Pros: Cross Validation / K-Fold Cross Validation provides a **more accurate estimate of test error**. It **uses the dataset more efficiently** than just a single train-test split

- Cons: However, the **tradeoff is the computational cost**. A k-fold cross validation takes **k times longer** than the train-split test approach.