INFO 2950: Intro to Data Science

Lecture 13
2023-10-11

Agenda

1. Overfitting
2. Train / Test Split
3. Evaluation Metrics

“Training a model”: single variable

- Given a df with two columns, x and y
- You run regression y~x in Python
- Python returns α-hat and β-hat
- Are there problems with this?

Overfitting: single variable

- It depends! If you just want to describe the data that you have, this is fine.

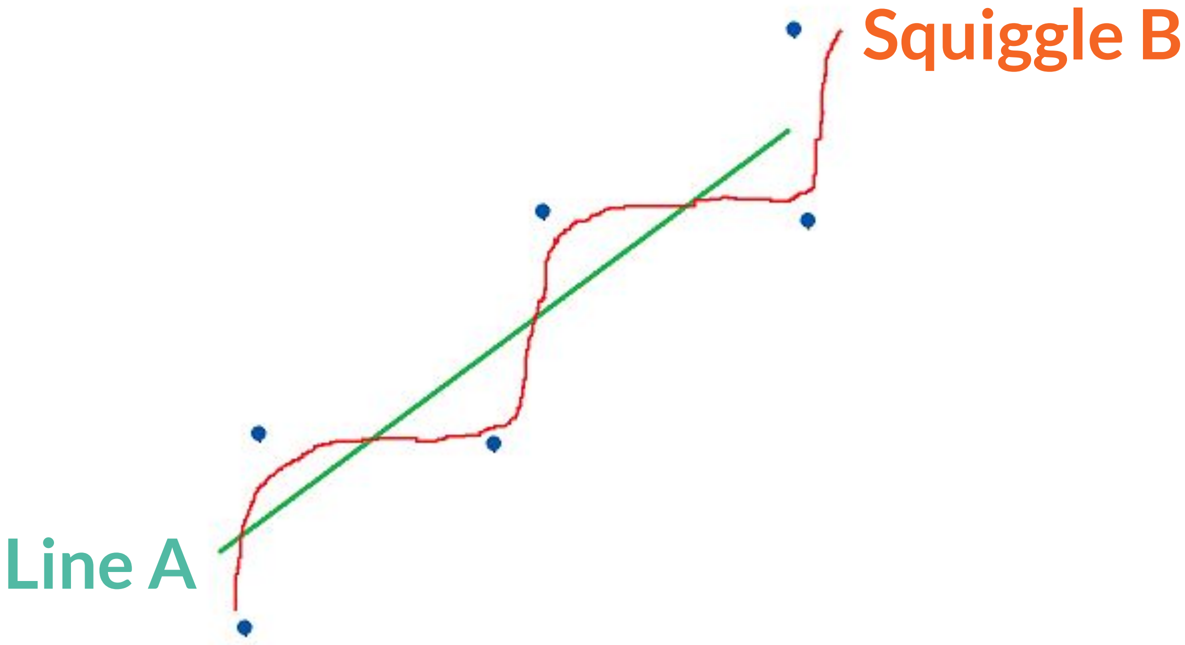
Overfitting: single variable

- It depends! If you just want to describe the data that you have, this is fine.
- If you're trying to **generalize your findings** to "new data", what happens if the (x, y) values in your df aren't representative of the "new data"?

Overfitting: single variable

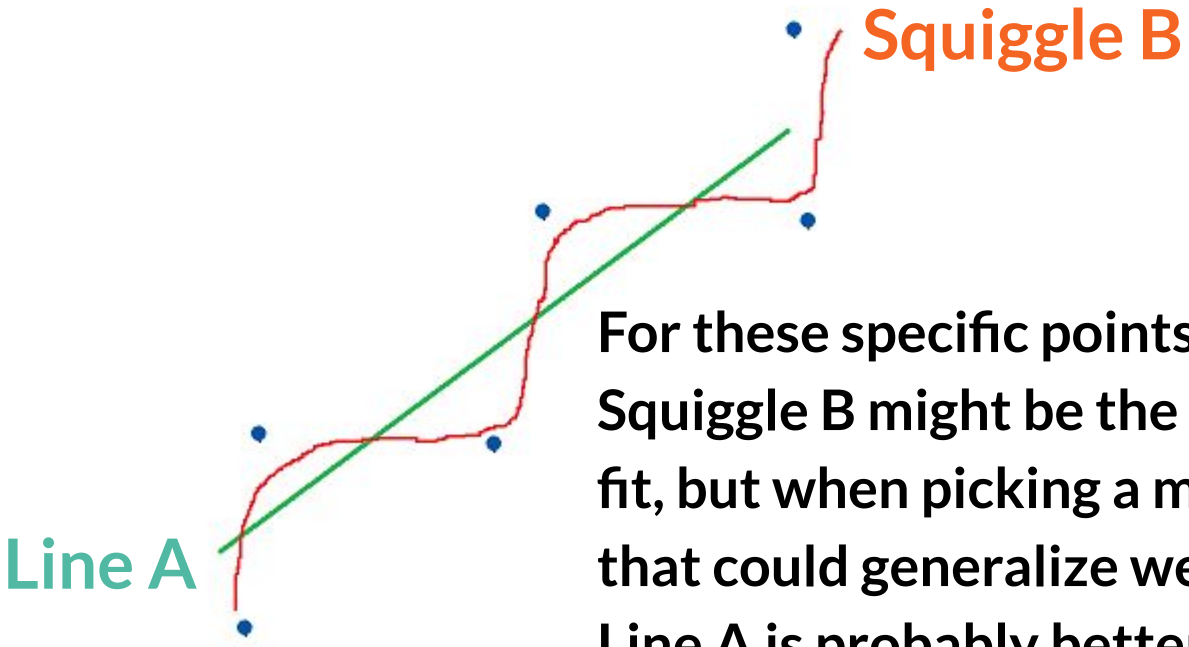
- It depends! If you just want to describe the data that you have, this is fine.
- If you're trying to generalize your findings to "new data", what happens if the (x, y) values in your df aren't representative of the "new data"?
    - You make **bad generalizations**
    - This is called **"overfitting"** to your existing data

Overfitting: which line is "better"?



https://statisticsbyjim.com/regression/overfitting-regression-models/

Overfitting: which line is “better”?



For these specific points, Squiggle B might be the best fit, but when picking a model that could generalize well, Line A is probably better.

https://statisticsbyjim.com/regression/overfitting-regression-models/

Training data: multiple variables

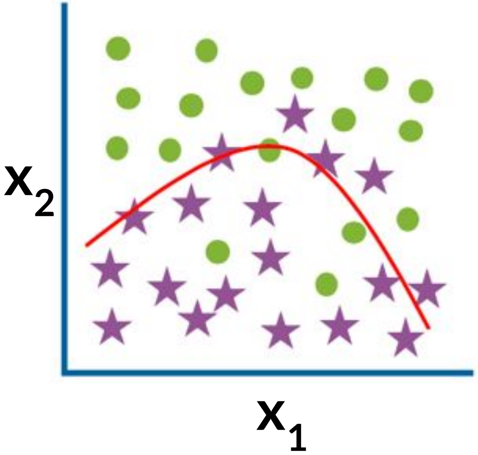
- Given multiple x’s and one y, now we run a multivariable regression
- Is overfitting a problem in this case, too?

Training data: multiple variables

- **Yes**: overfitting in high dimensions is ***much more likely***
- Many different input x's → the model can pick up on more complexity, but then you get **models adhering too much to the *data* instead of the underlying *idea***

Overfitting: multiple variables

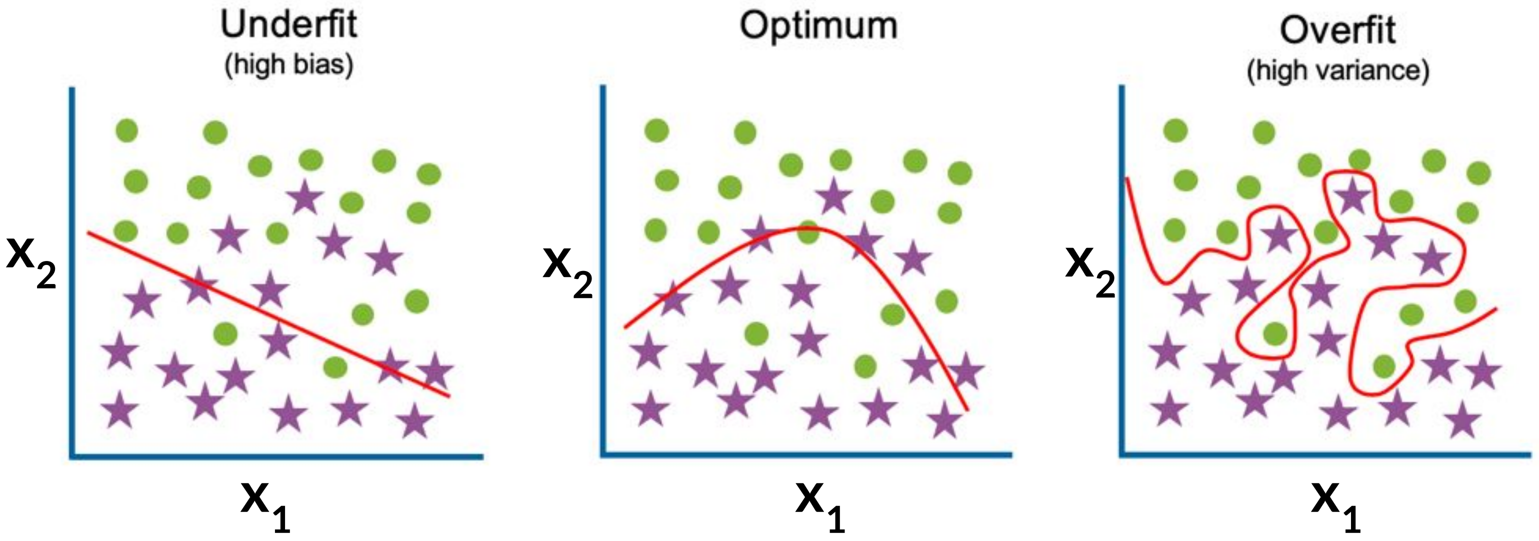
Output y (binary) represented by shape



https://www.ibm.com/cloud/learn/overfitting

Overfitting: multiple variables

Output y (binary) represented by shape



https://www.ibm.com/cloud/learn/overfitting

How to overcome overfitting?

- "**Feature selection**": identify the most important covariates (inputs) for your model and only include those. This allows you to reduce the number of dimensions of your data
- Use a **train / test split**
- **Regularization**
  - Much later in the class

Feature selection review

- Choose covariates that…
  - **make sense given domain expertise**
  - **aren't redundant** (i.e., *aren't collinear* and *don't overfit* the data)
  - allow you (with transformation) to get **random-looking residual plots**

How to overcome overfitting?

- "**Feature selection**": identify the most important covariates (inputs) for your model and only include those. This allows you to reduce the number of dimensions of your data
- Use a **train / test split**
- **Regularization**
    - Much later in the class

Train / test split

- Train/test split: **before you do anything,** randomly split your df's data rows into two subsets
    - Train set: 70% of your data
    - Test set: 30% of your data
- The 70/30 ratio not set in stone, just a rule of thumb
- *Often you will see a third dataset called the "validation set" that could yield a ~70/15/15 train/val/test split*

Train / test split for overfitting?

- Look at only your training set (a random 70% of your data) when building your model
- Then confirm it generalizes well to the other 30% of your data (the test set)!
    - *This step helps you confirm that you're not overfitting*

Train / test split for overfitting?

- Look at only your training set (a random 70% of your data) when building your model

- Then confirm it generalizes well to the other 30% of your data (the test set)!
    - *This step helps you confirm that you're not overfitting*
    - *Validation sets are nice because you can check for overfitting on just 15% of the data, and if you're overfitting, then you can keep fixing your model until you're ready to check the final 15% of your data (the test set)*

Think, pair, share: explain the meme

https://pbs.twimg.com/media/DdkUUTMV4AAsohU.jpg

Think, pair, share: explain the meme

This "model" was only *trained* on side sleepers! It works great for them, but is a terrible model if *tested* on back sleepers or people who roll around

https://pbs.twimg.com/media/DdkUUTMV4AAsohU.jpg

Think, pair, share: explain the meme

This is why it's important to represent a diversity of sleeping positions in both the *training data* and also the *test set*

https://pbs.twimg.com/media/DdkUUTMV4AAsohU.jpg

https://www.youtube.com/watch?v=DQWI1kvmwRg

How to generate a train / test split?

Step 0. Figure out what data you need to run your model

$$y \sim x_1 + x_2 + x_3 + x_4 + x_5$$

https://builtin.com/data-science/train-test-split

How to generate a train / test split?

Step 0. Figure out what data you need to run your model

$$y \sim x_1 + x_2 + x_3 + x_4 + x_5$$

How many columns *at minimum* should there be in your dataframe, in order to fit this regression?

https://builtin.com/data-science/train-test-split

How to generate a train / test split?

Step 0. Figure out what data you need to run your model

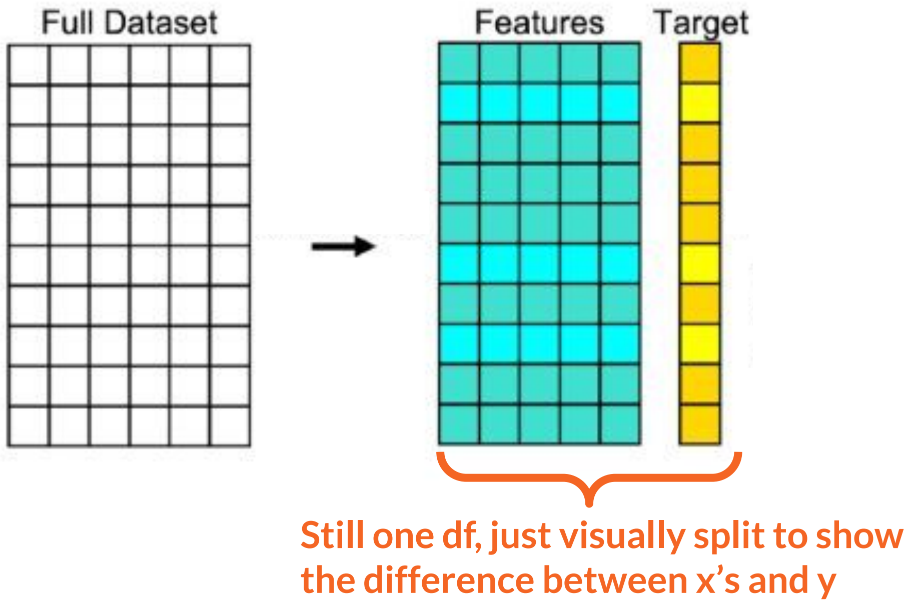
$$y \sim x_1 + x_2 + x_3 + x_4 + x_5$$

Answer: at least 5 columns for each of the x's plus 1 column for the y, so there should be at least 6 total columns present in your df to run this linear regression (without needing to reshape first)

https://builtin.com/data-science/train-test-split

How to generate a train / test split?

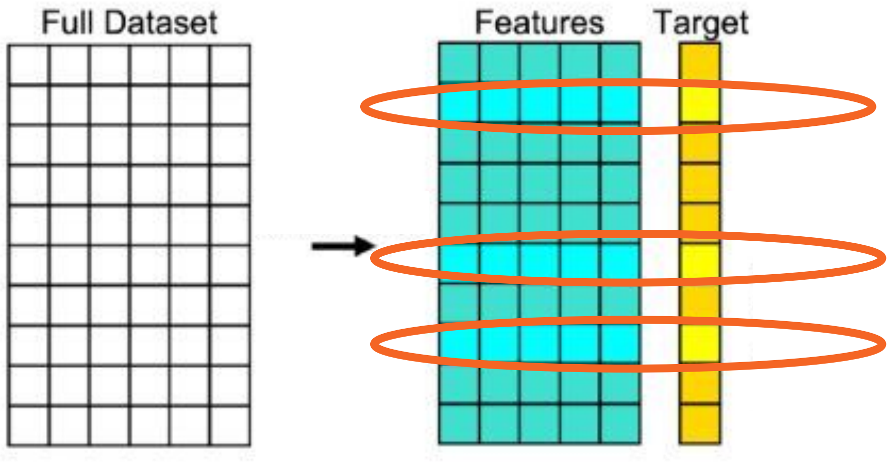
Step 1. Given your df, use a randomizer to assign 70% of your rows to the “train set” and 30% of your rows to the “test set”



https://builtin.com/data-science/train-test-split

How to generate a train / test split?

Step 1. Given your df, use a randomizer to assign 70% of your rows to the “train set” and 30% of your rows to the “test set”

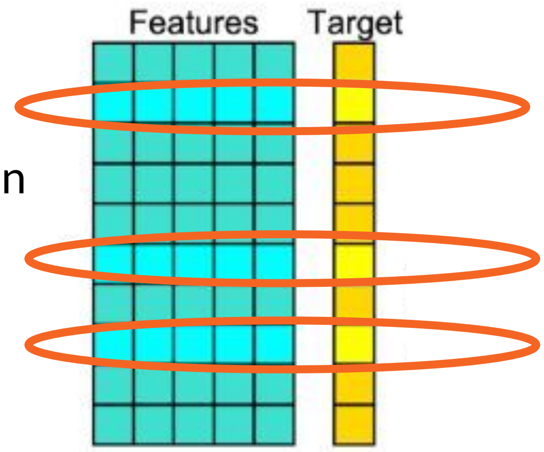


We’ve identified that these three rows will go into our test set

https://builtin.com/data-science/train-test-split

How to generate a train / test split?

Step 2. Split your df into two separate df's based on randomized values

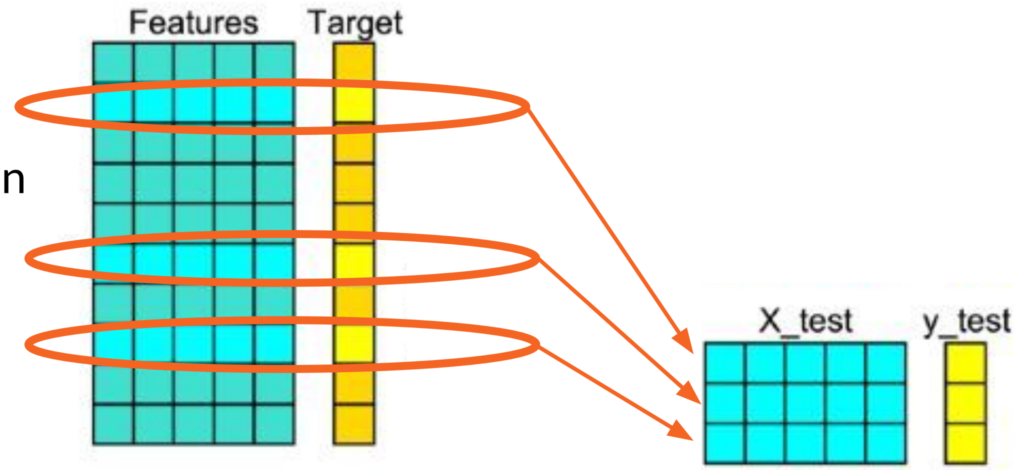


We've identified that these three rows will go into our test set

https://builtin.com/data-science/train-test-split

How to generate a train / test split?

Step 2. Split your df into two separate df's based on randomized values



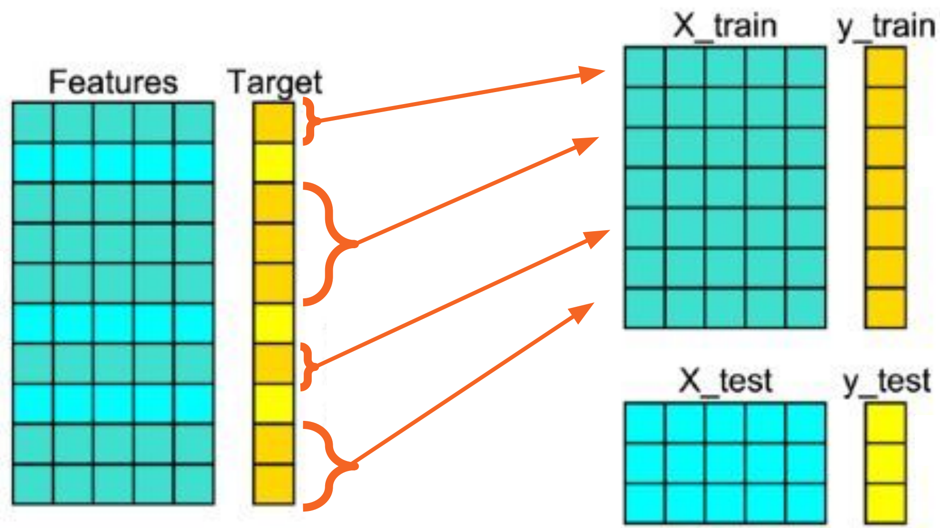
We've identified that these three rows will go into our test set

Test set (1 dataframe with 3 rows, 6 columns)

https://builtin.com/data-science/train-test-split

How to generate a train / test split?

Step 2. Split your df into two separate df's based on randomized values



Train set (1 df, 7 rows, 6 cols)

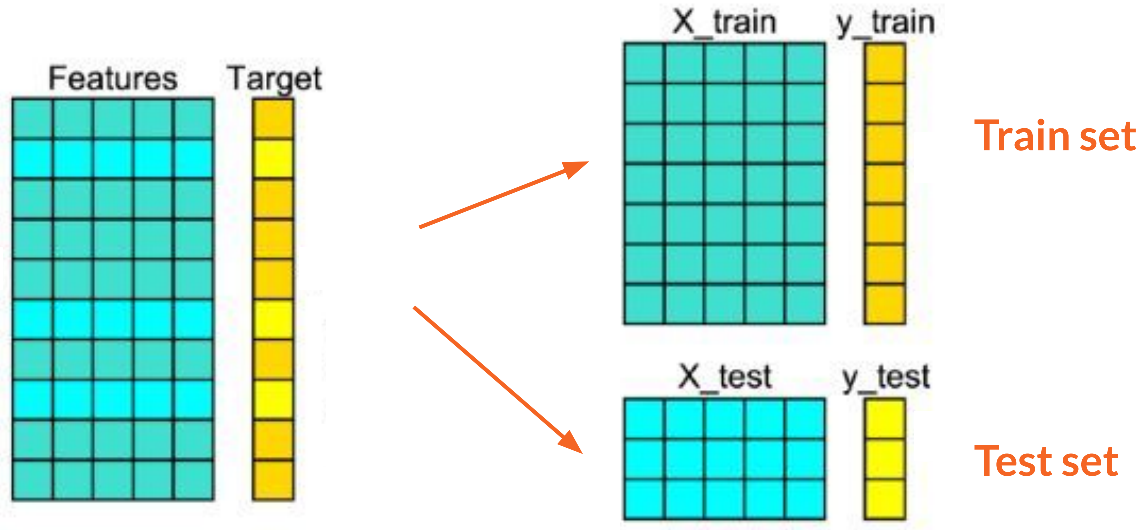
Test set

The other seven rows go to the "train set"

https://builtin.com/data-science/train-test-split

How to generate a train / test split?

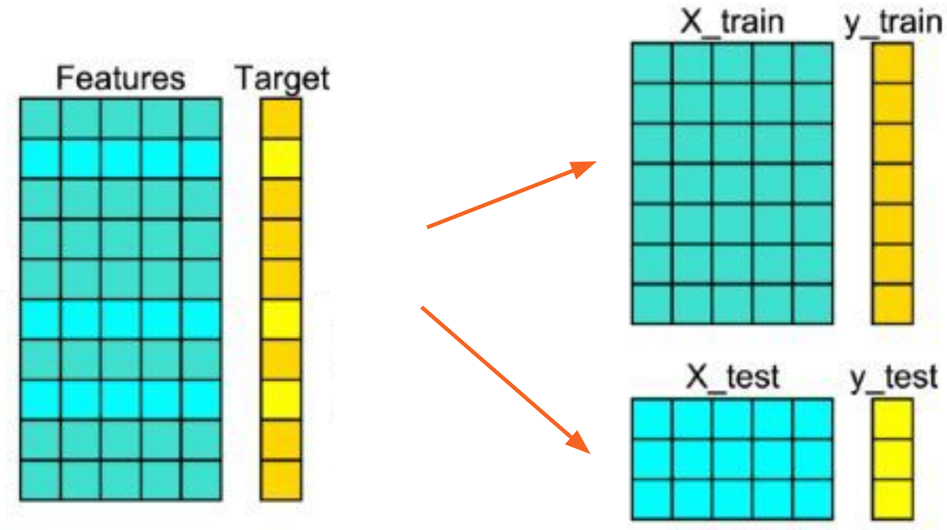
Step 2. Split your df into two separate df's based on randomized values



https://builtin.com/data-science/train-test-split

Four components generated

Step 2. Split your df into two separate df's based on randomized values



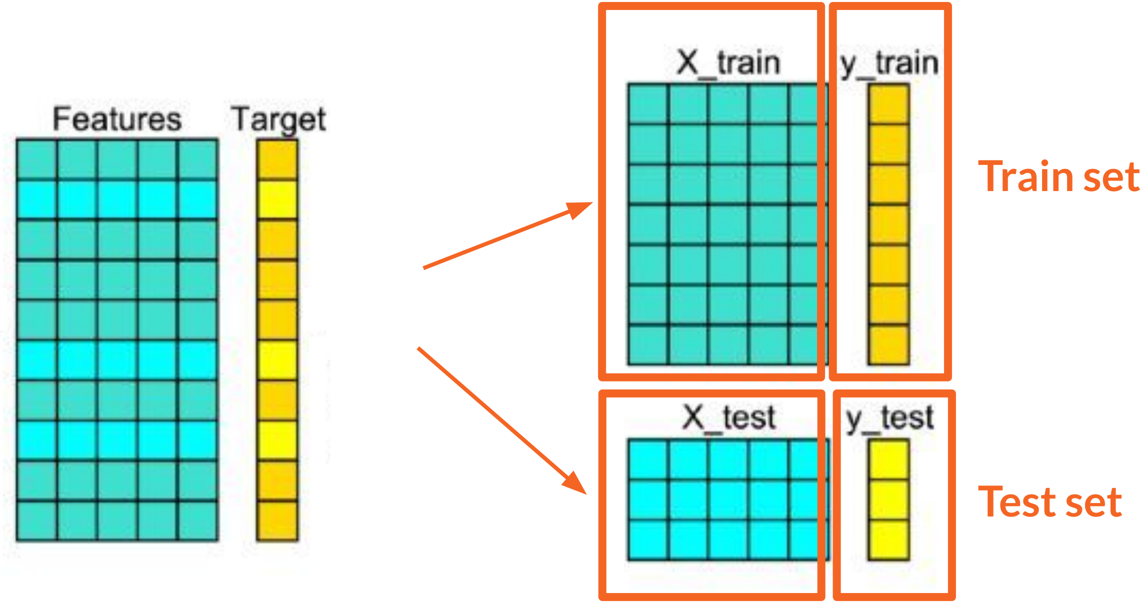
Train set

Test set

https://builtin.com/data-science/train-test-split

Four components generated

Step 2. Split your df into two separate df's based on randomized values



https://builtin.com/data-science/train-test-split

Generate train/test in Python

- `from sklearn.model_selection import train_test_split`
- `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)`

Generate train/test in Python

- `from sklearn.model_selection import train_test_split`
- `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)`

Generate train/test in Python

- `from sklearn.model_selection import train_test_split`
- `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)`

Inputs to train_test_split are X (can be multiple columns of x's) and y (one column)

Generate train/test in Python

- `from sklearn.model_selection import train_test_split`
- `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)`

70% train set, 30% test set split

Generate train/test in Python

- `from sklearn.model_selection import train_test_split`
- `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)`

set seed to a specific integer so you can reproduce your “random” results

Generate train/test in Python

- `from sklearn.model_selection import train_test_split`
- `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)`

output: 4 different datasets

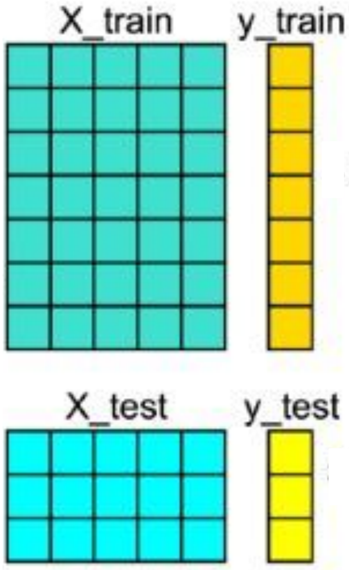
Pros/cons of train/test split?

- Pros of splitting your data:
  - Better out-of-sample generalizability
  - Confirmation that you're not overfitting across variables
  - Usually results in a more meaningful interpretation
- Cons of splitting your data:
  - Less training data means your initial model might not be as accurate; this isn't great if you're *definitely really truly* not trying to make broader generalizations (**rare**)

What do you do with train / test sets?

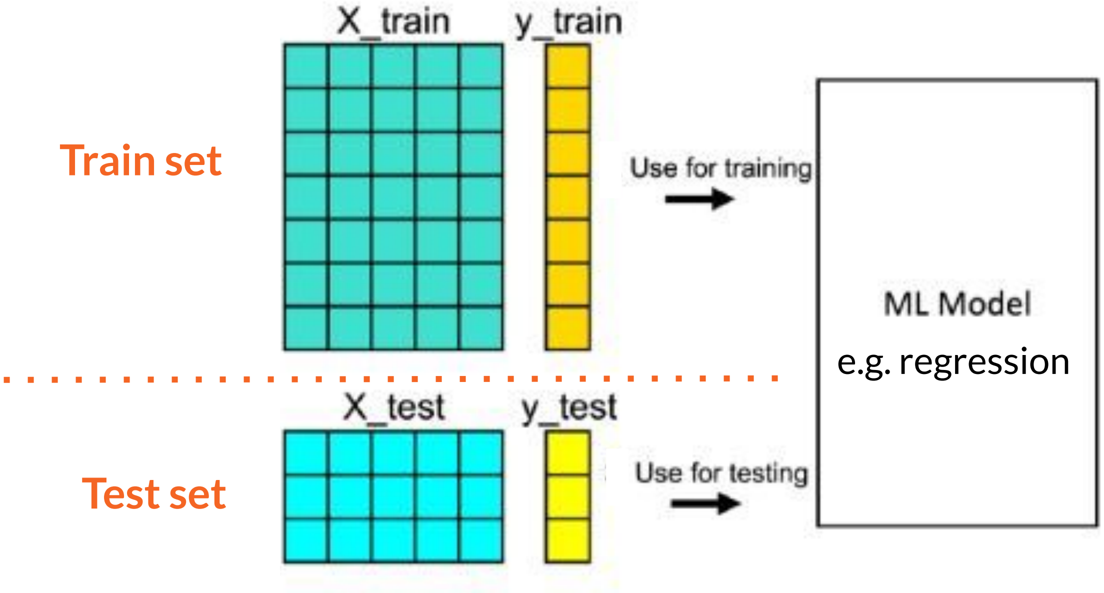
Train set

Test set



https://builtin.com/data-science/train-test-split

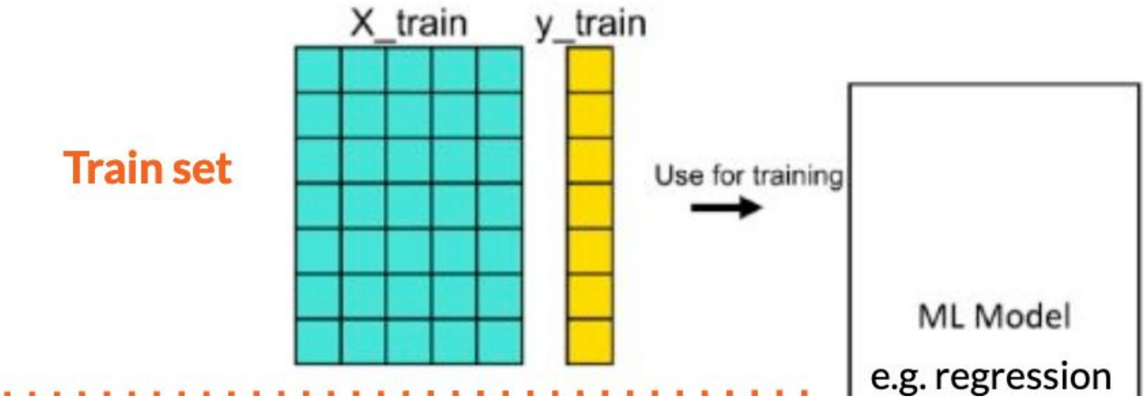
What do you do with train / test sets?



https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

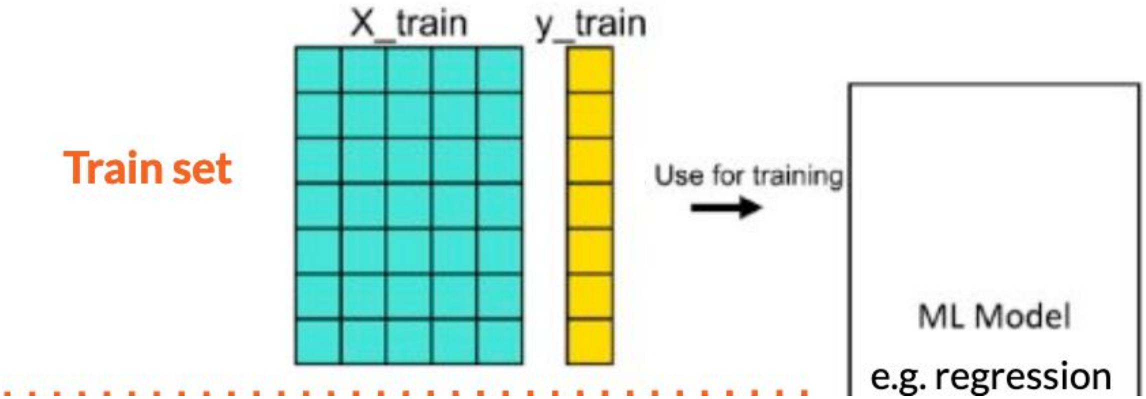
- **Step 1**: experiment with your regression on your training set. **Make any adjustments you need to here** (e.g. try different models, transformations, etc.)



https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

- **Step 1**: experiment with your regression on your training set. Make any adjustments you need to here (e.g. try different models, transformations, etc.)




```
model = LinearRegression().fit(X_train,y_train)
```

https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

- **Step 2**: make predictions using the train set only

```
model = LinearRegression().fit(X_train,y_train)

y_hat_train = model.predict(X_train)
```

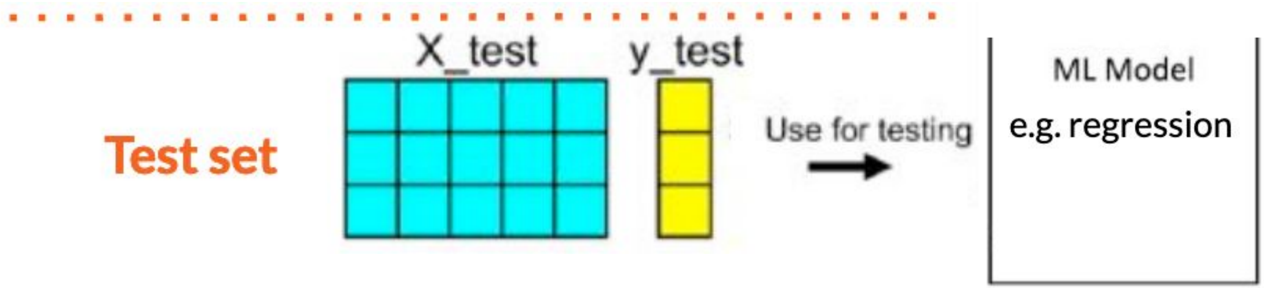
Why do we do this? **Because we want to compare our true y values (y_train) to the values predicted by our model (y_hat_train).**

Details: stay tuned for Step 4!

https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

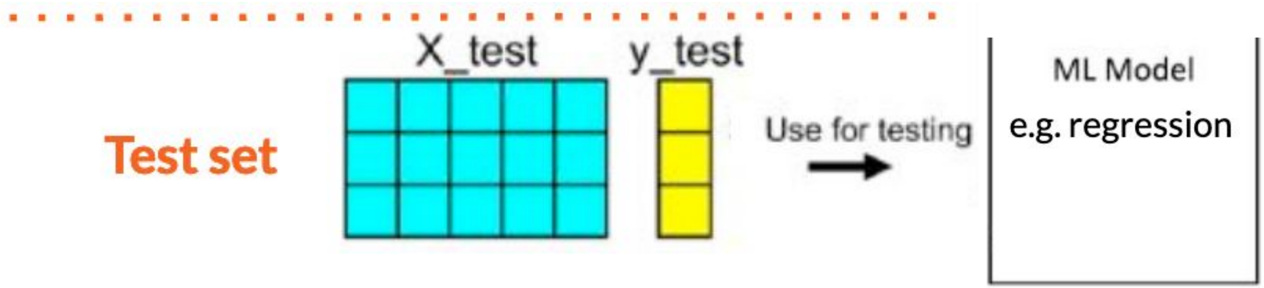
- **Step 1**: fit model on train set
- **Step 2**: predict y-hats from using train set model on X_train
- **Step 3**: predict y-hats from using train set model on **test set**



https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

- **Step 1**: fit model on train set
- **Step 2**: predict y-hats from using train set model on X_train
- **Step 3**: predict y-hats from using train set model on **test set**



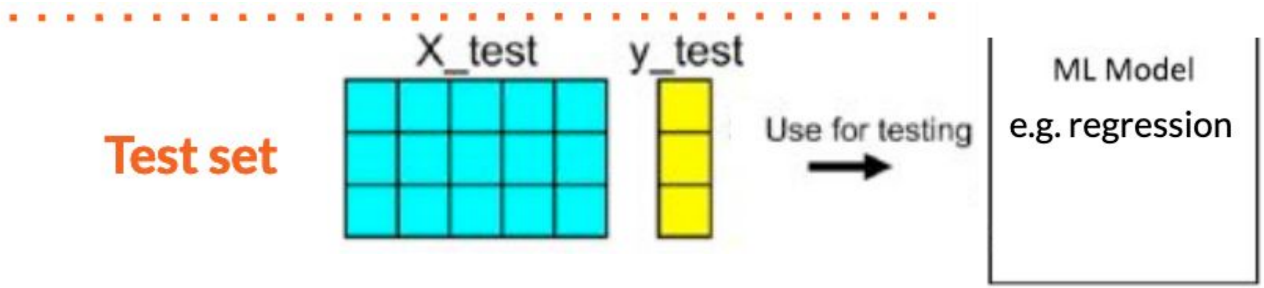

```
y_hat_test = model.predict(______)
```

https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

- **Step 1**: fit model on train set
- **Step 2**: predict y-hats from using train set model on X_train

- **Step 3**: predict y-hats from using train set model on **test set**



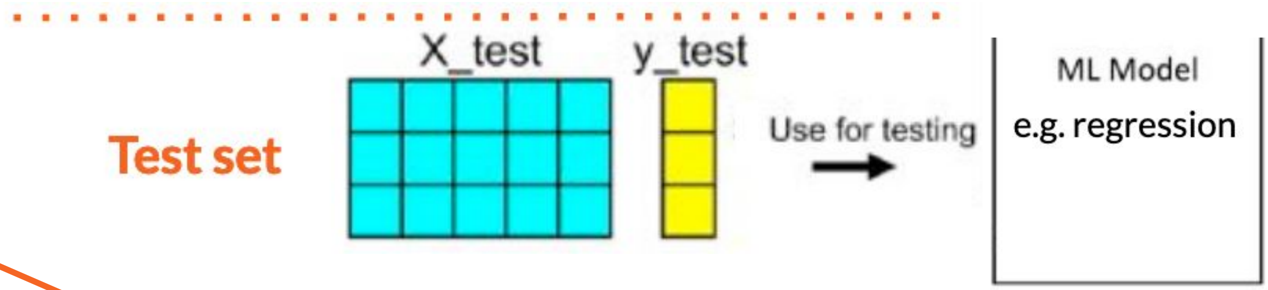

```
y_hat_test = model.predict(X_test)
```

https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

- **Step 1**: fit model on train set
- **Step 2**: predict y-hats from using train set model on X_train

- **Step 3**: predict y-hats from using train set model on **test set**



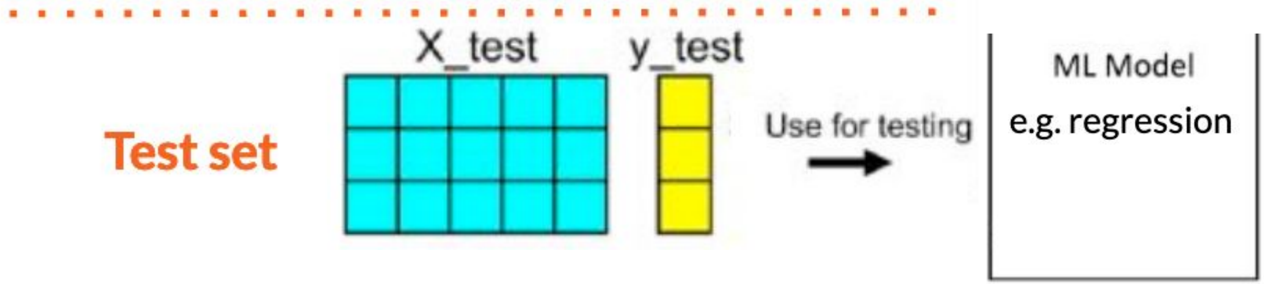
What is the "true" value of y that we want to compare y_hat_test to?

```
y_hat_test = model.predict(X_test)
```

https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

- **Step 1**: fit model on train set
- **Step 2**: predict y-hats from using train set model on X_train
- **Step 3**: predict y-hats from using train set model on **test set**




```
y_hat_test = model.predict(X_test)
```

y_test is the true output when the input is X_test (they are both in the test set)

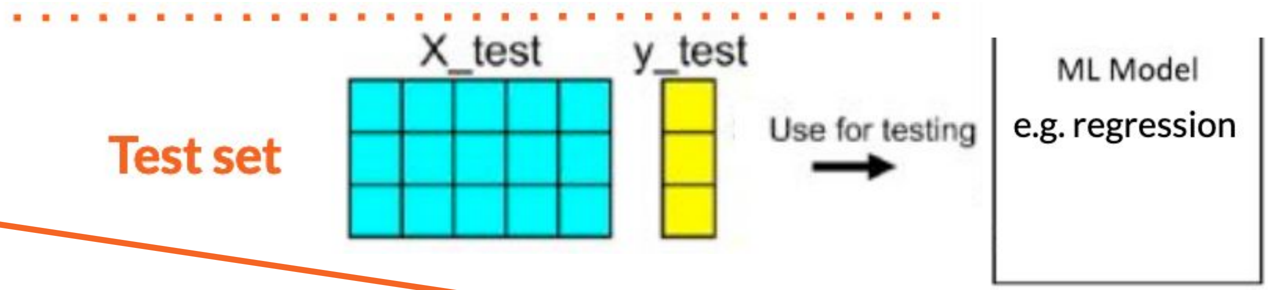
https://builtin.com/data-science/train-test-split

What do you do with train / test sets?

- **Step 1**: fit model on train set
- **Step 2**: predict y-hats from using train set model on X_train

- **Step 3**: predict y-hats from using train set model on **test set**

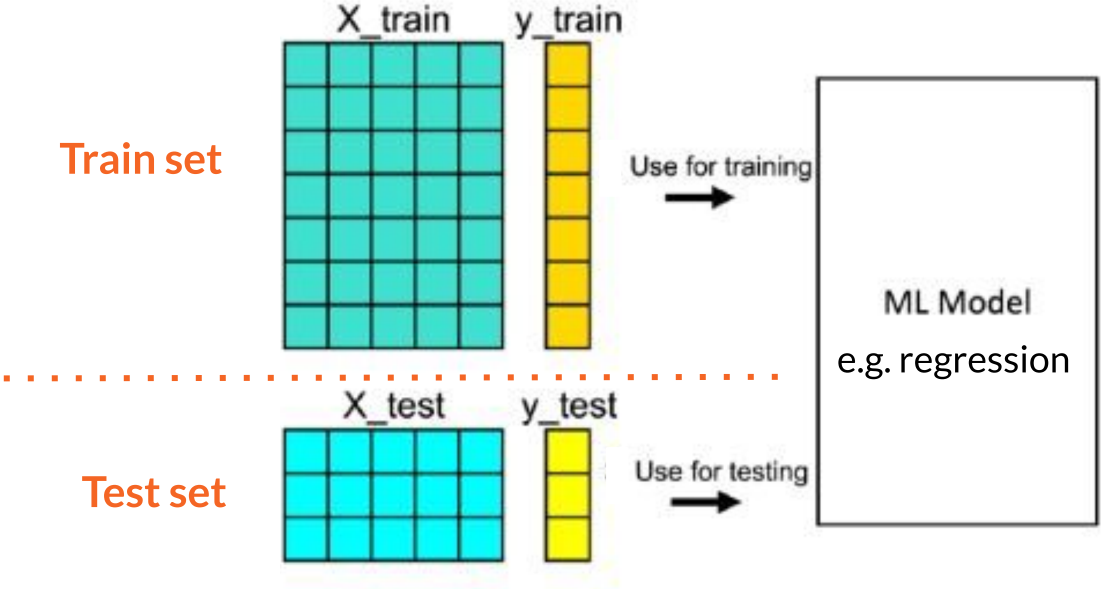
Notice we use the train set model when predicting the test set ŷ




```
y_hat_test = model.predict(X_test)
```

https://builtin.com/data-science/train-test-split

Why do you predict both train and test set y-hats with the same ML model?



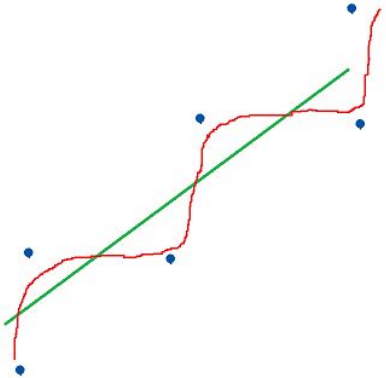
https://builtin.com/data-science/train-test-split

Think, pair, share: why don't we want to predict y_hat_test by fitting a new model on the test set?

https://twitter.com/deeplearningai_/status/1242160243204689920

Why don't we want to predict y_hat_test by fitting a new model on the test set?

We'd be 1. training a model on an even smaller dataset (only 30% of the data – so probably less generalizable) and 2. artificially reporting "good" results because y_hat_test should be pretty accurate if you trained on the test set!



https://twitter.com/deeplearningai_/status/1242160243204689920

Comparing evaluation metrics

- **Step 1**: fit model on train set
- **Step 2**: predict y-hats from using train set model on X_train
- **Step 3**: predict y-hats from using train set model on test set

- **Step 4:** calculate "**evaluation metrics**"
  - For now, think of this as "accuracy" of the model
  - Evaluate accuracy metrics on the **train** set
  - Evaluate accuracy metrics on the **test** set

Comparing evaluation metrics

- **Step 4:** calculate "**evaluation metrics**" on...
    - **Train set**:
        - Compare **y_hat_train** to **y_train**
        - Gives you a sense of whether your model is good on your train set only
    - **Test set:**
        - Compare **y_hat_test** to **y_test**
        - Gives you a good sense of how your model would generalize to "other" data *(are you overfitting?)*

Danger zone: your model might look “good”, but actually be overfitting

https://9gag.com/gag/aOYo6Ly

Match answers to the grid

Keep experimenting with your regression model – the good test set metrics are a fluke!	A
LGTM	B
Keep experimenting with your regression model – it doesn't seem to do well on any data!	C
Your model isn't generalizing well to out-of-sample data. Figure out how to fix this in your model!	D

- **What does it mean if…**

	Train set metrics "bad"	Train set metrics "good"
Test metrics "good"	?	?
Test metrics "bad"	?	?

Match answers to the grid

Keep experimenting with your regression model – the good test set metrics are a fluke!	A
LGTM	B
Keep experimenting with your regression model – it doesn't seem to do well on any data!	C
Your model isn't generalizing well to out-of-sample data. Figure out how to fix this in your model!	D

- **What does it mean if…**

	Train set metrics "bad"	Train set metrics "good"
Test metrics "good"	A	B
Test metrics "bad"	C	D

Match answers to the grid

- **What does it mean if...**

	Train set metrics "bad"	Train set metrics "good"
Test metrics "good"	Keep experimenting with your regression model – the good test set metrics are a fluke!	LGTM
Test metrics "bad"	Keep experimenting with your regression model – it doesn't seem to do well on any data!	Your model isn't generalizing well to out-of-sample data. Figure out how to fix this in your model!

Straightforward cases where either both results are good, or both results are bad

Match answers to the grid

- **What does it mean if…**

	Train set metrics "bad"	**Train set metrics "good"**
Test metrics "good"	Keep experimenting with your regression model – the good test set metrics are a fluke! *(or, you accidentally trained on the test set)*	LGTM
Test metrics "bad"	Keep experimenting with your regression model – it doesn't seem to do well on any data!	Your model isn't generalizing well to out-of-sample data. Figure out how to fix this in your model!

Match answers to the grid

- **What does it mean if…**

	Train set metrics "bad"	Train set metrics "good"
Test metrics "good"	Keep experimenting with your regression model – the good test set metrics are a fluke! *(or, you accidentally trained on the test set)*	LGTM
Test metrics "bad"	Keep experimenting with your regression model – it doesn't seem to do well on any data!	Your model isn't generalizing well to out-of-sample data. Figure out how to fix this in your model!

The "overfitting" case!

You should never ignore the test set in an attempt to get good accuracy!

https://twitter.com/deeplearningai_/status/1375100156186611712

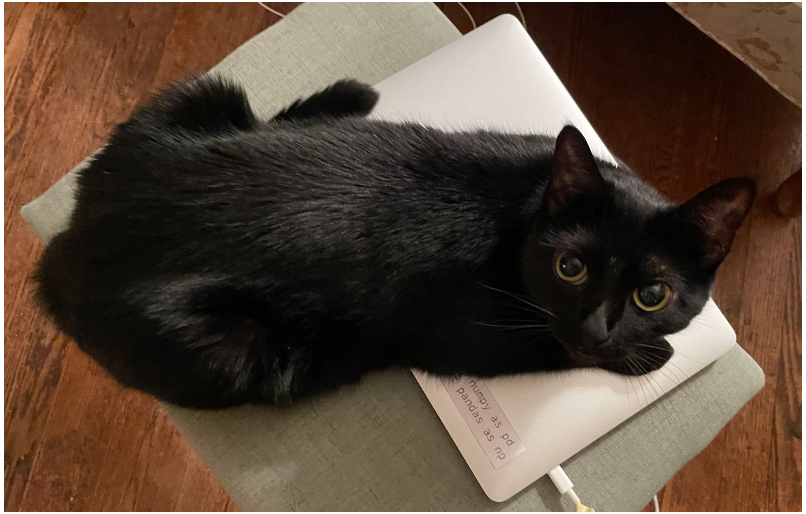
Takeaways: reg evaluation

- Use a train/test split if you want your model to be generalizable (~70/30%)
- Don't peek at the test set until you're ready to do a final confirmation that your model is generalizable
  - Train set evaluation: compare y_hat_train to y_train
  - Test set evaluation: compare y_hat_test to y_test

Caution: train/test

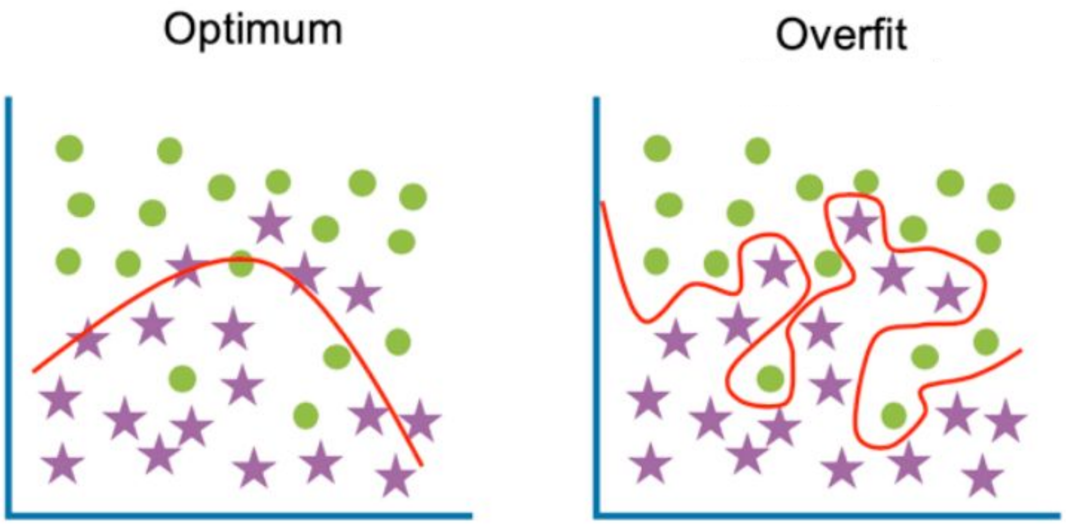
- Need to make sure your train/test sets have similar distributions
- Need to be careful if you have time series data (can't just randomly pick different times!)
- What if you just get lucky with your train set choice?
- We'll address these + discuss cross validation in a future lecture

1 minute break & attendance

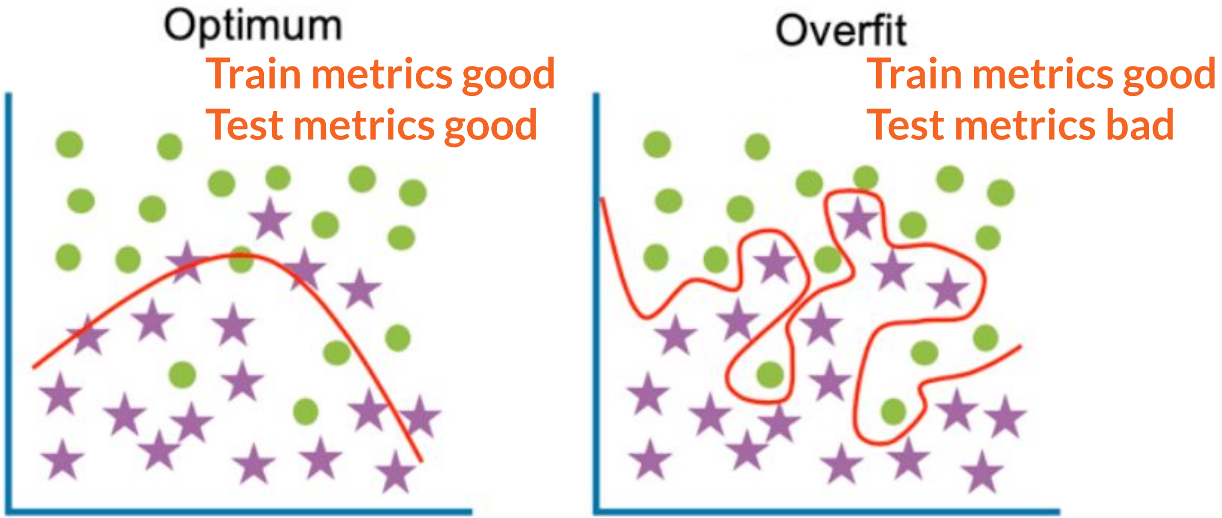


tinyurl.com/z5wm2vcw

Motivation: I want to be able to numerically find that the left/right model is good/bad.



https://www.ibm.com/cloud/learn/overfitting



https://www.ibm.com/cloud/learn/overfitting

Train metrics good
Test metrics bad

Evaluating Regressions

- How do we know if our regressions are any good?
    - Checking residual plots, correlation matrices for inputs, interaction plots, etc.

https://www.facebook.com/RegressiveRegressionMemesForHeteroskedasticTeens/

Evaluating Regressions

- How do we know if our regressions are any good?
  - Checking residual plots, correlation matrices for inputs, interaction plots, etc.
  - **Evaluation metrics**

$$R^2 = \frac{\sum(\hat{y} - \bar{\hat{y}})^2}{\sum(y - \bar{y})^2}$$

What is R^2?

- R^2 = Explained Variation / Total Variation
- Summarizes the % variation in the output that is explainable by the regression model
- R^2 is between 0 to 1, we generally want our models to have higher R^2

$$R^2 = \frac{\sum(\hat{y} - \bar{\hat{y}})^2}{\sum(y - \bar{y})^2}$$

Reasons to not use R^2

- R^2 can be low even when the model is correct
  - E.g., when variance increases, the R^2 value goes to 0
- R^2 can be high even when the model is wrong
  - E.g., non-linear data

https://library.virginia.edu/data/articles/is-r-squared-useless

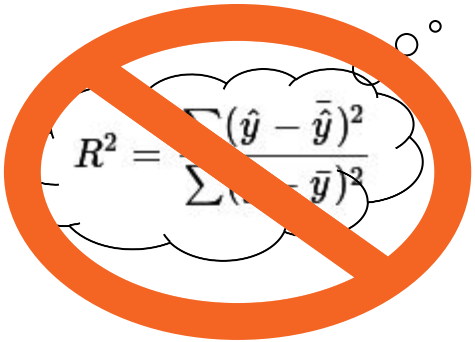
Reasons to not use R^2

$$R^2 = \frac{\sum(\hat{y} - \bar{\hat{y}})^2}{\sum(y - \bar{y})^2}$$

- R^2 can be low even when the model is correct
  - E.g., when variance increases, the R^2 value goes to 0
- R^2 can be high even when the model is wrong
  - E.g., non-linear data
- R^2 can get worse if you keep your model the same, but change the range of x, or use transformations of y
- R^2 is symmetric between x and y

https://library.virginia.edu/data/articles/is-r-squared-useless

Use R^2 with *extreme caution*



- It's relatively uncommon for data science practitioners to use R^2 in many real-world applications

$$R^2_{adjusted} = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

where

R^2 = sample R-square

p = Number of predictors

N = Total sample size.

- If you ***really must*** report R^2, use the ***adjusted*** R^2, which at least accounts for having multiple inputs (regular R^2 increases in # inputs x)

https://www.listendata.com/2014/08/adjusted-r-squared.html https://library.virginia.edu/data/articles/is-r-squared-useless

What to use if not R^2?

- There are lots of other **evaluation metrics** to use
    - More common in practice
    - Better for doing model selection of linear (and nonlinear) regressions, e.g. telling you if you're overfitting

Recall that...

- **What does it mean if...**

	Train set metrics "bad"	Train set metrics "good"
Test metrics "good"	Keep experimenting with your regression model – the good test set metrics are a fluke!	LGTM
Test metrics "bad"	Keep experimenting with your regression model – it doesn't seem to do well on any data!	Your model isn't generalizing well to out-of-sample data. Figure out how to fix this in your model!

What does “good” and “bad” mean??

- **How do you quantify your evaluation metrics?**
    - Depends on whether your y is:
        - **Numeric variable (non-binary)**
        - **Binary variable**
    - Intuition: metric should be related to residuals, so big error → “bad” metric

Nomenclature for these slides

General case	**"y_true", "y_i"**	**"y_hat", "$\hat{y}_i$"**
Train set	y_train	y_hat_train
Test set	y_test	y_hat_test

Nomenclature for these slides

We want to make this comparison

General case	“y_true”, “y_i”	“y_hat”, “$\hat{y}_i$”
Train set	y_train	y_hat_train
Test set	y_test	y_hat_test

Nomenclature for these slides

General case	**"y_true", "y_i"**	**"y_hat", "$\hat{y}_i$"**
Train set	y_train	y_hat_train
Test set	y_test	y_hat_test

We also want to make this comparison

Nomenclature for these slides

When providing formulas in these slides, we'll generalize to these

General case	**"y_true", "y_i"**	**"y_hat", "$\hat{y}_i$"**
Train set	y_train	y_hat_train
Test set	y_test	y_hat_test

If our y's are numerical non-binary

- **Mean Squared Error (MSE)** $\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2$
- **Root Mean Squared Error (RMSE)** $\sqrt{\text{MSE}}$
- **Mean Absolute Error (MAE)** $\frac{1}{n}\sum_{1}^{n}|y_i - \hat{y}_i|$
- **Mean Absolute Percent Error (MAPE)** $\frac{100}{n}\sum_{i}^{n}\frac{y_i - \hat{y}_i}{y_i}$

If our y's are numerical non-binary

All 4 of these metrics are often used in the real world!

- **Mean Squared Error (MSE)** $\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2$
- **Root Mean Squared Error (RMSE)** $\sqrt{\text{MSE}}$
- **Mean Absolute Error (MAE)** $\frac{1}{n}\sum_{1}^{n}|y_i - \hat{y}_i|$
- **Mean Absolute Percent Error (MAPE)** $\frac{100}{n}\sum_{i}^{n}\frac{y_i - \hat{y}_i}{y_i}$

If our y's are numerical non-binary

- **Mean Squared Error (MSE)** $\frac{1}{n}\sum_{i}^{n}\boxed{(y_i - \hat{y}_i)}^2$
- **Root Mean Squared Error (RMSE)** $\sqrt{\text{MSE}}$
- **Mean Absolute Error (MAE)** $\frac{1}{n}\sum_{1}^{n}\boxed{\left|y_i - \hat{y}_i\right|}$
- **Mean Absolute Percent Error (MAPE)** $\frac{100}{n}\sum_{i}^{n}\left|\frac{\boxed{y_i - \hat{y}_i}}{y_i}\right|$

All based on residuals!

Numerical prediction metrics in Python

- `from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error`

- `mse = mean_squared_error(y_true,y_hat)`
- `rmse = np.sqrt(mse)`
- `mae = mean_absolute_error(y_true,y_hat)`
- `mape = mean_absolute_percentage_error(y_true,y_hat)`

Comparing evaluation metrics

- **Step 1**: fit model on train set
- **Step 2**: predict y-hats from using train set model on X_train
- **Step 3**: predict y-hats from using train set model on **test set**

- **Step 4:** calculate "**evaluation metrics**"
  - For now, think of this as "accuracy" of the model
  - **Evaluate "accuracy" metrics on the train set**
  - **Evaluate "accuracy" metrics on the test set**

Make sure your inputs are what you really want (overall, train set only, test set only)

- `from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error`

- `mse = mean_squared_error(y_true,y_hat)`
- `rmse = np.sqrt(mse)`
- `mae = mean_absolute_error(y_true,y_hat)`
- `mape = mean_absolute_percentage_error(y_true,y_hat)`

If our y's are binary

- There are just a few very common metrics (stay tuned in three slides!)
- But to understand them, we have to think more carefully about 0's and 1's

Binary classification outcomes

“Given this item’s customer review and price, do I predict that it’s a nose pack?”

	Model predicts 1	Model predicts 0
True value is 1		
True value is 0		

Binary classification outcomes

"Given this item's customer review and price, do I predict that it's a nose pack?"

	Model predicts 1	Model predicts 0
True value is 1	**True positive** Correct prediction	**False negative**
True value is 0	**False positive**	**True negative** Correct prediction

(Lots of options for classification)

		Predicted condition		Sources: [24][25][26][27][28][29][30][31][32] view · talk · edit	
	Total population $= P + N$	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	Positive (P)	**True positive (TP),** hit	**False negative (FN),** type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** type I error, false alarm, overestimation	**True negative (TN),** correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

False positives vs false negatives

- Which metric do we care more about?
  - Depends on the application!
- Sometimes you want to prioritize minimizing fp over fn, and sometimes vice versa
  - It can be very difficult to find a method that minimizes both – sometimes you must make a trade-off!

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	tp	fn
$y=0$	fp	tn

False positives or false negatives?

- It’s Halloween and some annoying kids want to egg your house without causing damage to your house. They use an “egg classifier” for whether an egg-shaped item is an egg (egg=1, low damage to house) or a rock (egg=0, high damage to house).
    - *Do the kids care more about fp or fn?*

https://www.reddit.com/r/mildlyinteresting/comments/5uvn8r/i_collect_rocks_that_look_like_eggs/

https://towardsdatascience.com/false-positives-vs-false-negatives-4184c2ff941a

False positives or false negatives?

- It’s Halloween and some annoying kids want to egg your house without causing damage to your house. They use an “egg classifier” for whether an egg-shaped item is an egg (egg=1, low damage to house) or a rock (egg=0, high damage to house).
  - **If the kids think something is an egg but it’s actually a rock, that’s bad! If they think something is a rock and don’t throw it (but it’s really an egg), it doesn’t matter. Consequences of a Type I error are costlier, so they’ll want to minimize fp.**

https://www.reddit.com/r/mildlyinteresting/comments/5uvn8r/i_collect_rocks_that_look_like_eggs/

https://towardsdatascience.com/false-positives-vs-false-negatives-4184c2ff941a

False positives or false negatives?

True Value	Model predicts... Egg	Model predicts... Rock
Egg	**True positive** Correct prediction	**False negative** (kids' model predicts rock so it doesn't get thrown, but it's actually just an egg)
Rock	**False positive** (kids' model predicts egg, so it'll get thrown, but it's actually a rock!)	**True negative** Correct prediction

https://www.reddit.com/r/mildlyinteresting/comments/5uvn8r/i_collect_rocks_that_look_like_eggs/

False positives or false negatives?

True Value	Model predicts... Egg	Model predicts... Rock
Egg	**True positive** Correct prediction	**False negative** (kids' model predicts rock so it doesn't get thrown, but it's actually just an egg)
Rock	**False positive** (kids' model predicts egg, so it'll get thrown, but it's actually a rock!)	**True negative** Correct prediction

This is the most dangerous case! → False positive

https://www.reddit.com/r/mildlyinteresting/comments/5uvn8r/i_collect_rocks_that_look_like_eggs/

	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

False positives or false negatives?

- The kids discover that the neighborhood egg stash contains some golden eggs. They use the same “egg classifier” for whether an egg-shaped item is an egg (egg=1, potentially high $ value) or a rock (egg=0, definitely $0).
    - *Do the kids care more about fp or fn?*

https://www.increasemyefficiency.com/how-to-build-a-golden-egg/

https://towardsdatascience.com/false-positives-vs-false-negatives-4184c2ff941a

False positives or false negatives?

- The kids discover that the neighborhood egg stash contains some golden eggs. They use the same “egg classifier” for whether an egg-shaped item is an egg (egg=1, potentially high $ value) or a rock (egg=0, definitely $0).
    - **If the kids think something is an egg but it’s actually a rock, that’s okay – they just get $0 out of it. If they think something is a rock but it’s really a $$$ golden egg, they’re missing out on a ton of money. Consequences of a Type II error are costlier, so they’ll want to minimize fn.**

https://www.increasemyefficiency.com/how-to-build-a-golden-egg/

https://towardsdatascience.com/false-positives-vs-false-negatives-4184c2ff941a

False positives or false negatives?

https://www.increasemyefficiency.com/how-to-build-a-golden-egg/

Model predicts...

True Value	Egg	Rock
Egg	**True positive** Correct prediction	**False negative** (kids' model predicts rock so they don't try to cash it in, but it's actually worth $$$!)
Rock	**False positive** (kids' model predicts egg, so they try to cash it in, but it's only worth $0)	**True negative** Correct prediction

False positives or false negatives?

This is the worst case!

		Model predicts...	
		Egg	Rock
True Value	Egg	**True positive** Correct prediction	**False negative** (kids' model predicts rock so they don't try to cash it in, but it's actually worth $$$!)
	Rock	**False positive** (kids' model predicts egg, so they try to cash it in, but it's only worth $0)	**True negative** Correct prediction

https://www.increasemyefficiency.com/how-to-build-a-golden-egg/

(Lots of options for classification)

		Predicted condition		Sources: [24][25][26][27][28][29][30][31][32] view · talk · edit	
	Total population $= P + N$	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	Positive (P)	**True positive (TP),** hit	**False negative (FN),** type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** type I error, false alarm, overestimation	**True negative (TN),** correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Evaluating with accuracy

- **Accuracy = (tp + tn) / (p + n)**
  = % things you predicted correctly
- Seems intuitive…
- But rarely used in ML. Why not?

Evaluating with accuracy

- $y = 1$ if someone has an extremely rare disease (with 1% incidence in your dataset)
- You make a naive model that simply predicts $y = 0$ for every single input x
- **What accuracy would you get with this naive model?** *Accuracy = (tp + tn) / (p + n)*

Evaluating with accuracy

- y = 1 if someone has an extremely rare disease (with 1% incidence in your dataset)
- You make a naive model that simply predicts y = 0 for every single input x
- **Your naive model would get 99% accuracy. This isn't useful for what matters: making good predictions even if there are only a few true y = 1 values.**

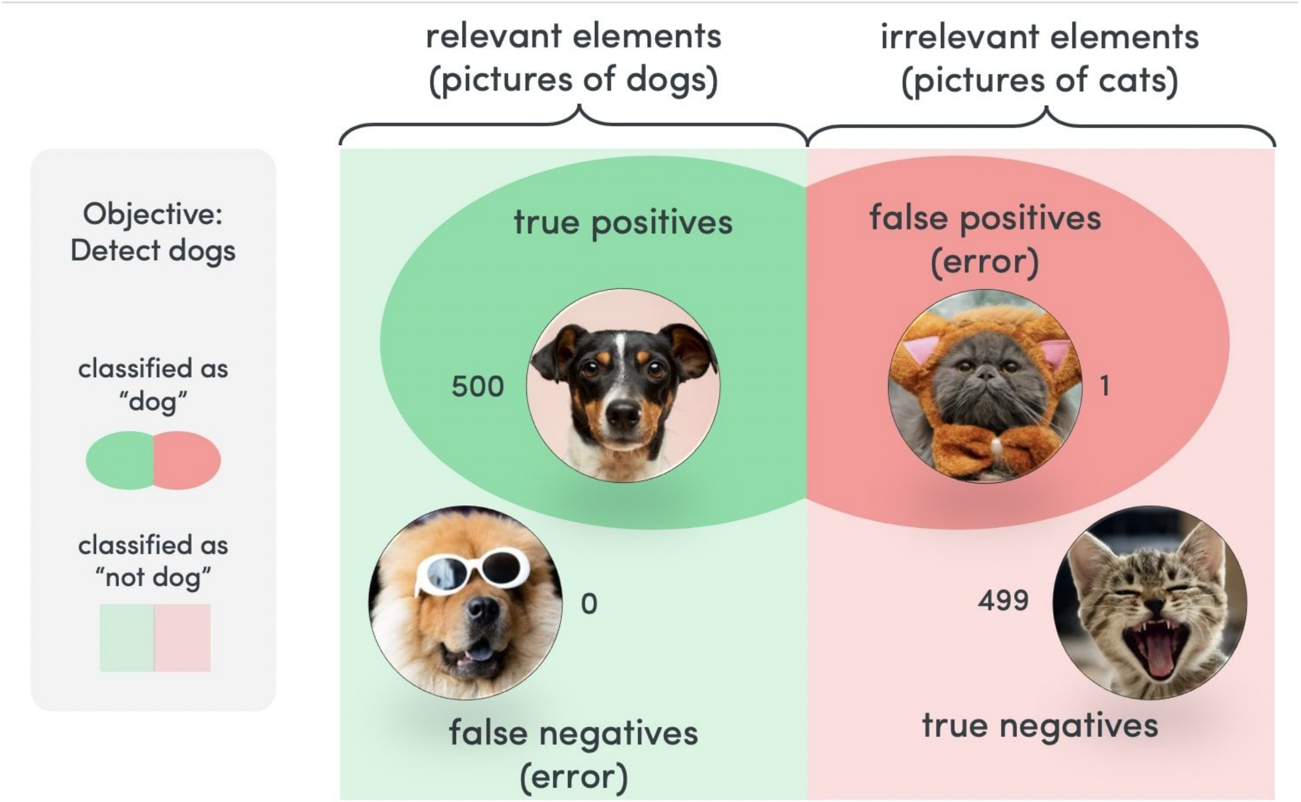
https://www.facebook.com/AnalyticsVidhya/photos/a.457177331040819/2826021784156350/?type=3

Binary classification outcomes

	Model predicts 1	Model predicts 0
True value is 1	**True positive** Correct prediction	**False negative**
True value is 0	**False positive**	**True negative** Correct prediction

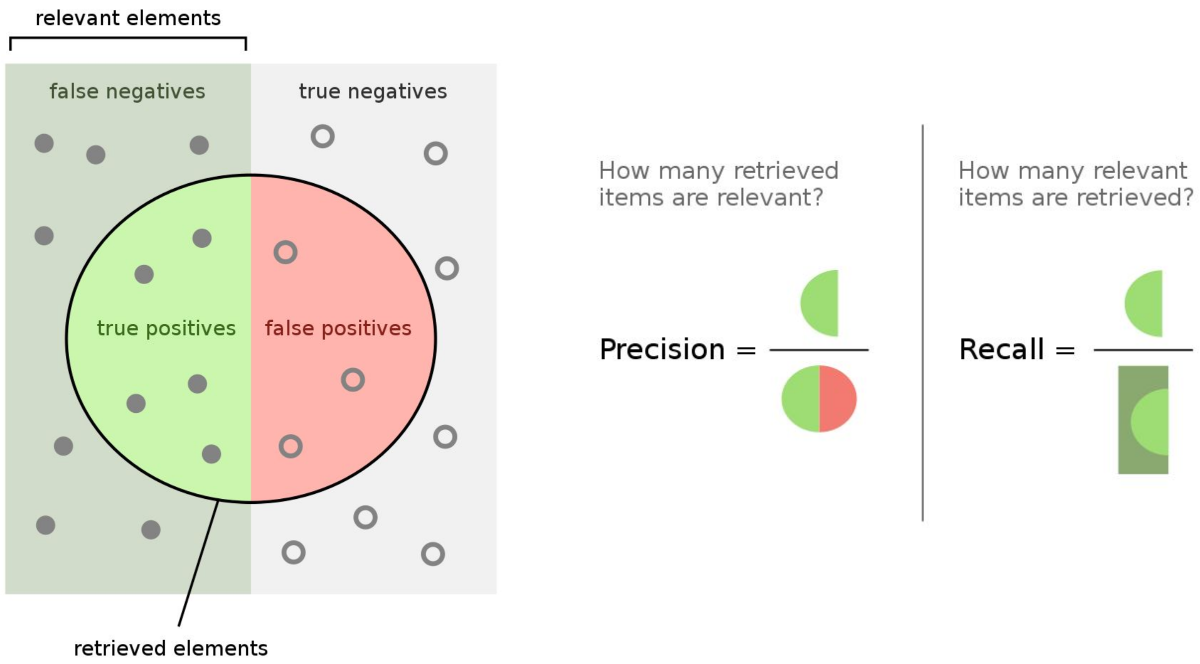
Binary evaluation metrics, part 1

- **Precision** = tp / (tp + fp)
- **Recall** (a.k.a. sensitivity) = tp / (tp + fn)
- Higher is better!



https://levity.ai/blog/precision-vs-recall

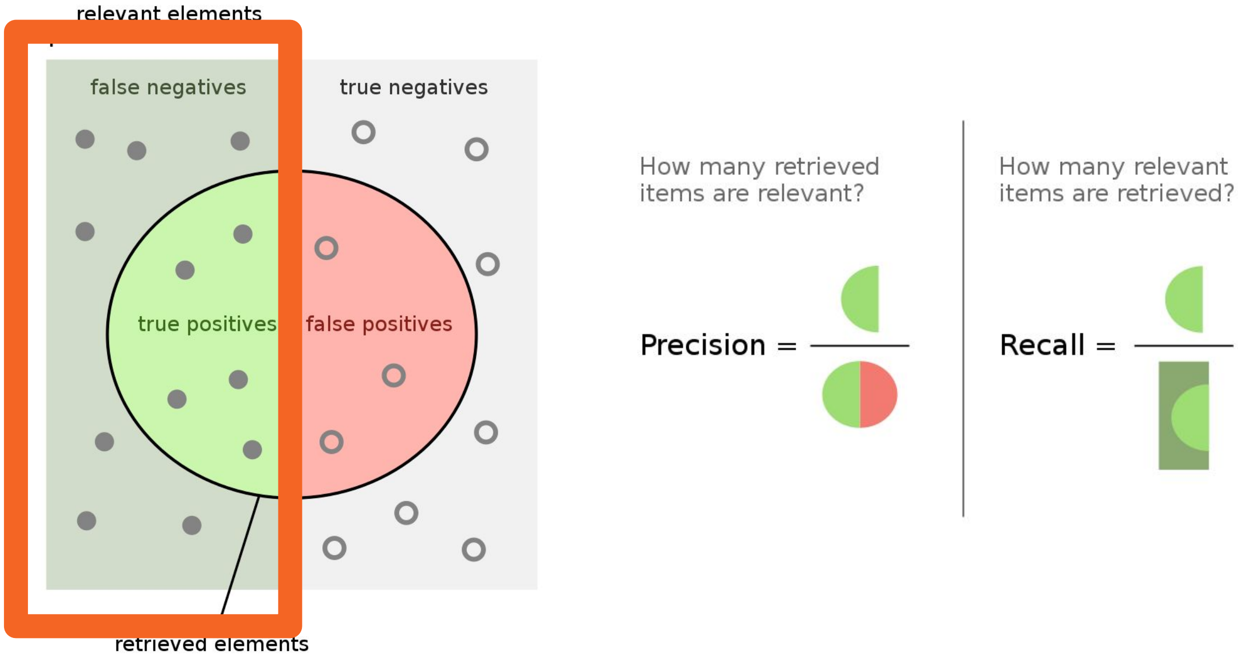
Precision, Recall



https://en.wikipedia.org/wiki/Precision_and_recall

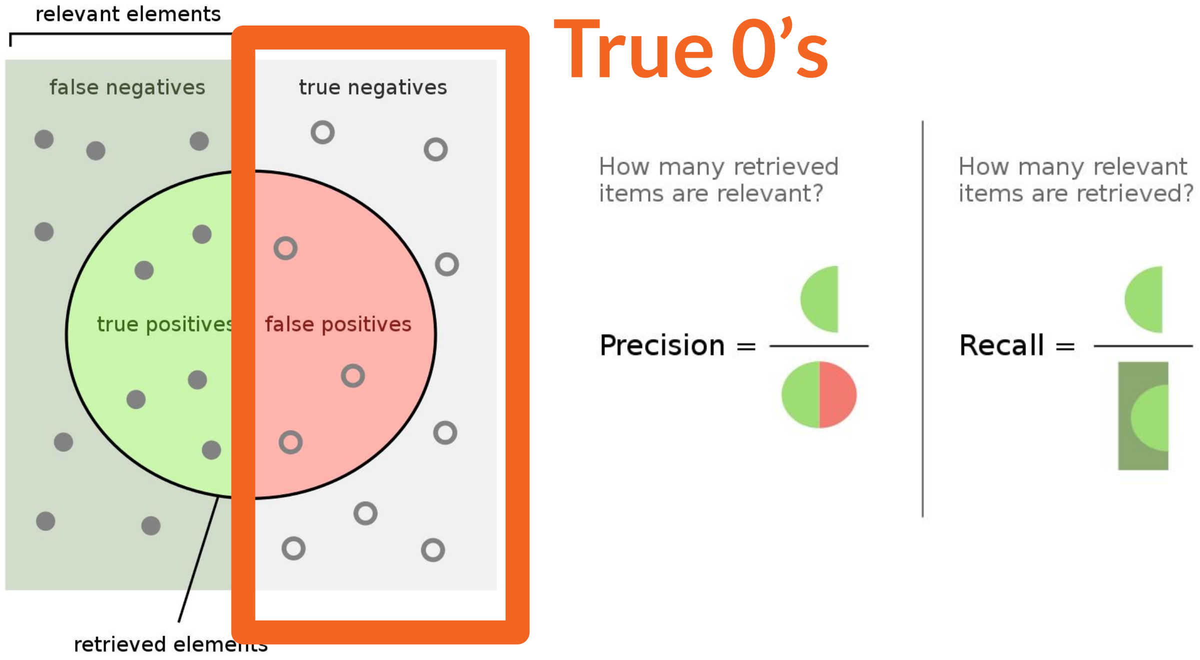
Precision, Recall

True 1's



https://en.wikipedia.org/wiki/Precision_and_recall

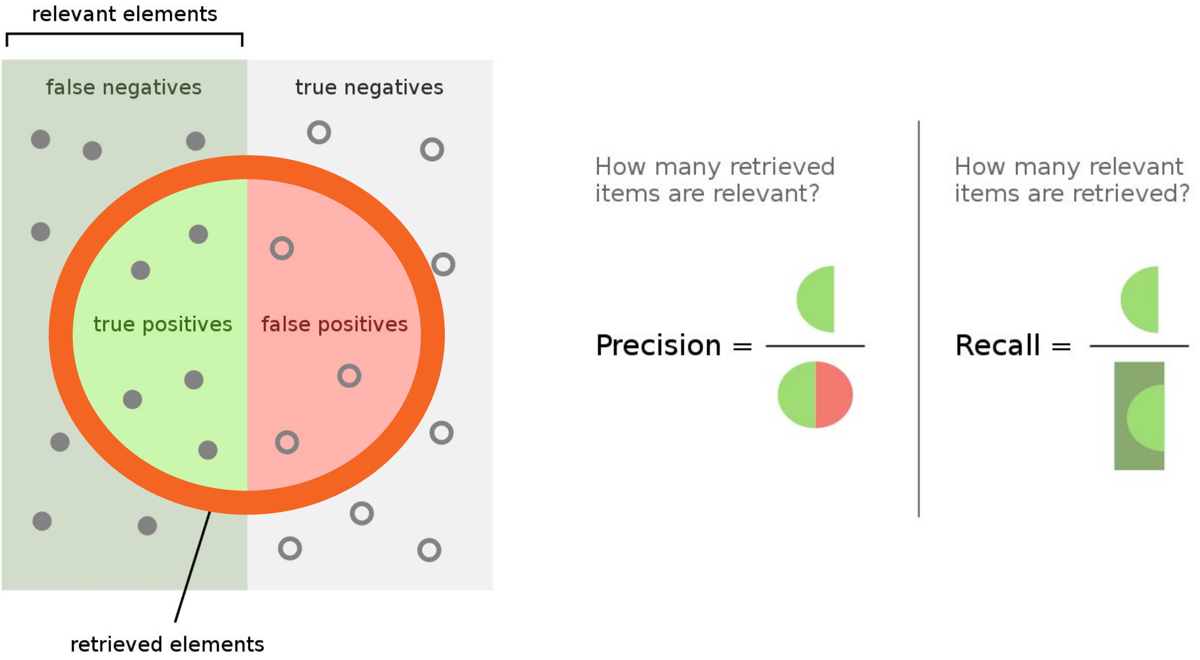
Precision, Recall



https://en.wikipedia.org/wiki/Precision_and_recall

Precision, Recall

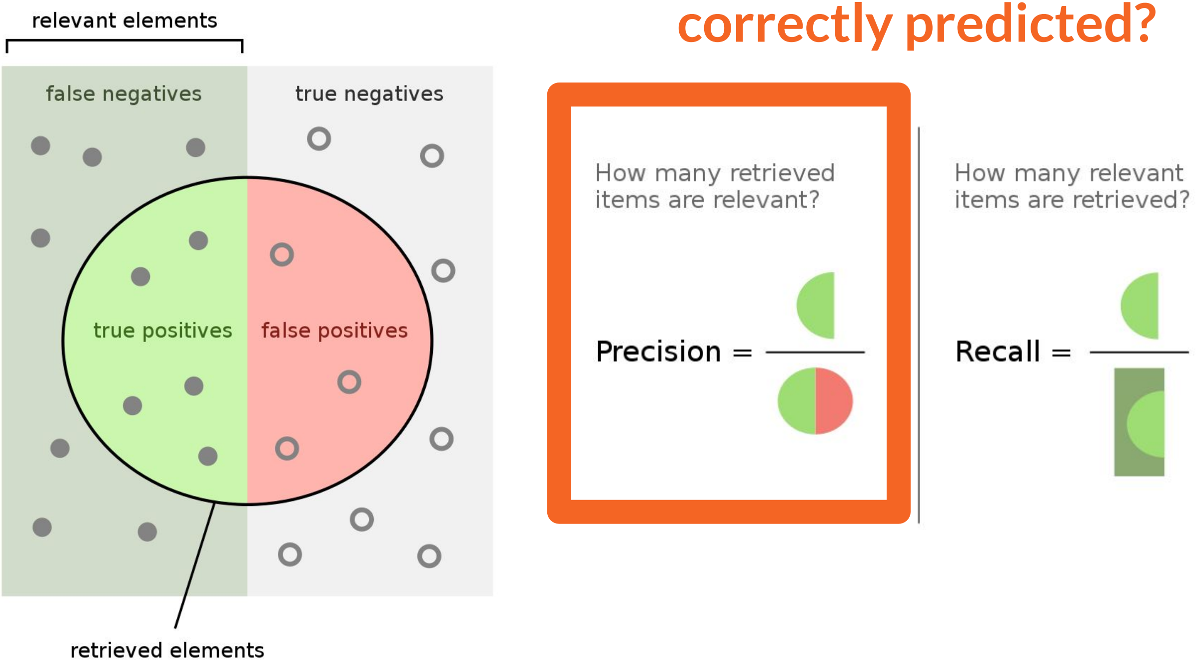
Predicted 1's



https://en.wikipedia.org/wiki/Precision_and_recall

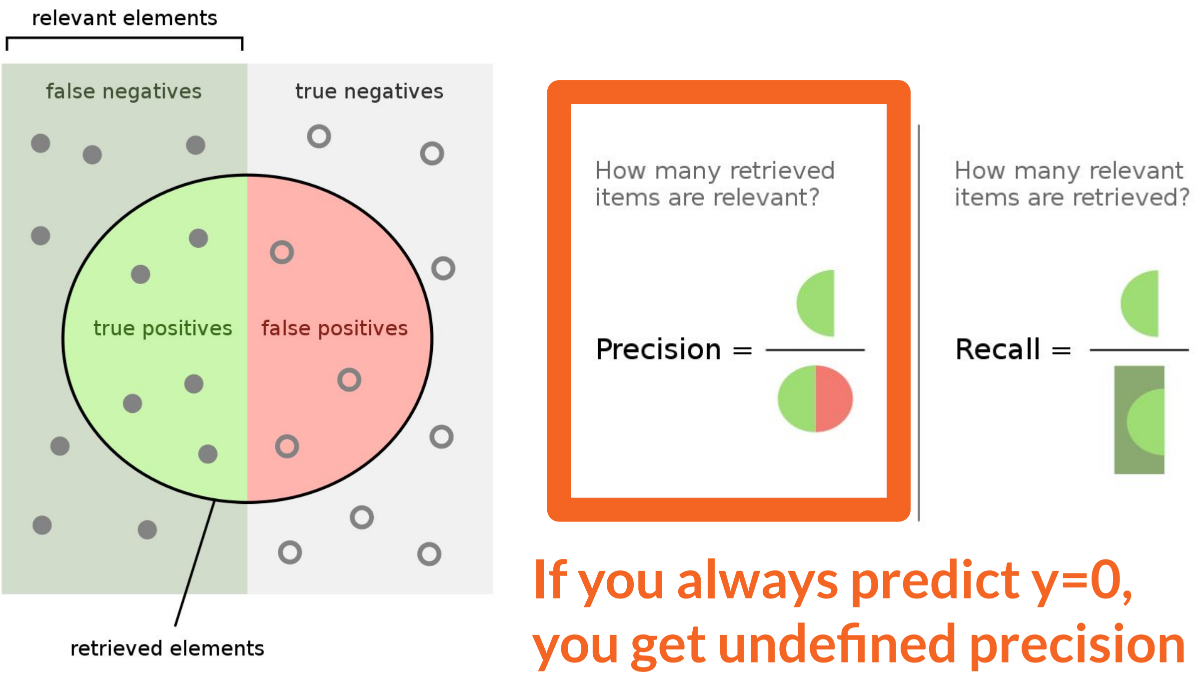
Precision, Recall

How many of your predicted y=1's were correctly predicted?



https://en.wikipedia.org/wiki/Precision_and_recall

Precision, Recall

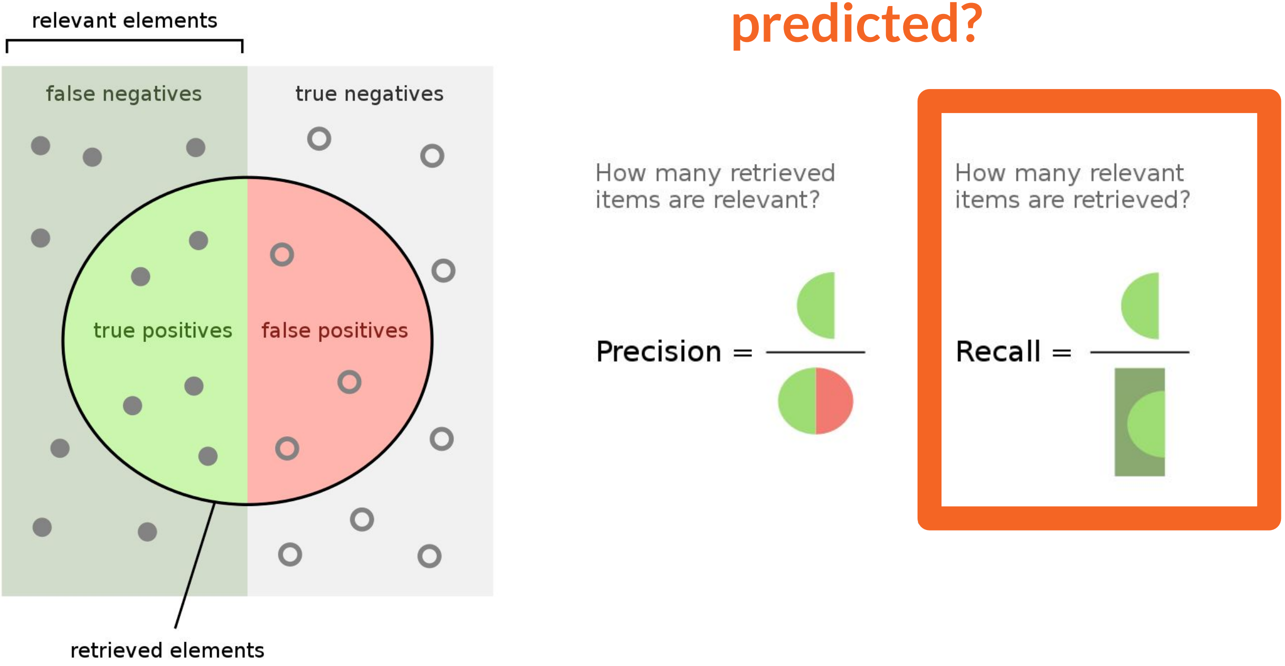


If you always predict y=0, you get undefined precision

https://en.wikipedia.org/wiki/Precision_and_recall

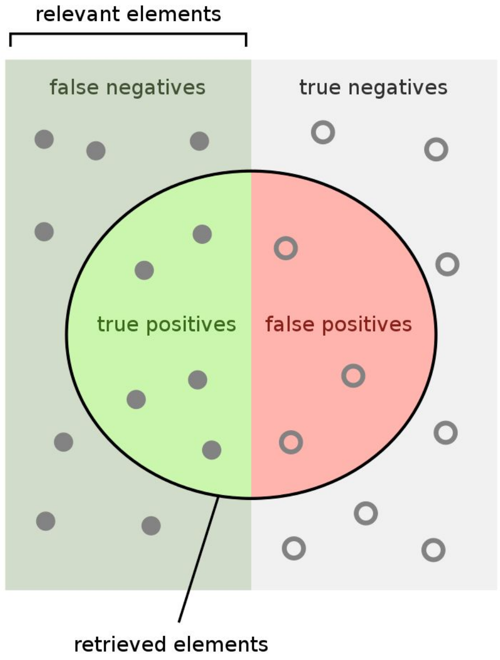
Precision, Recall

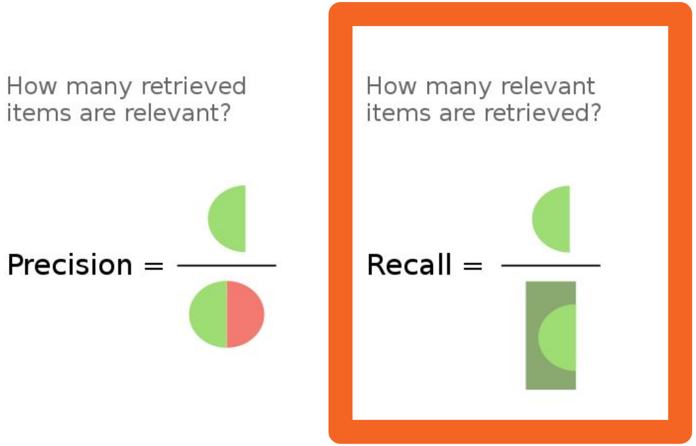
How many of the true y=1's were correctly predicted?



https://en.wikipedia.org/wiki/Precision_and_recall

Precision, Recall

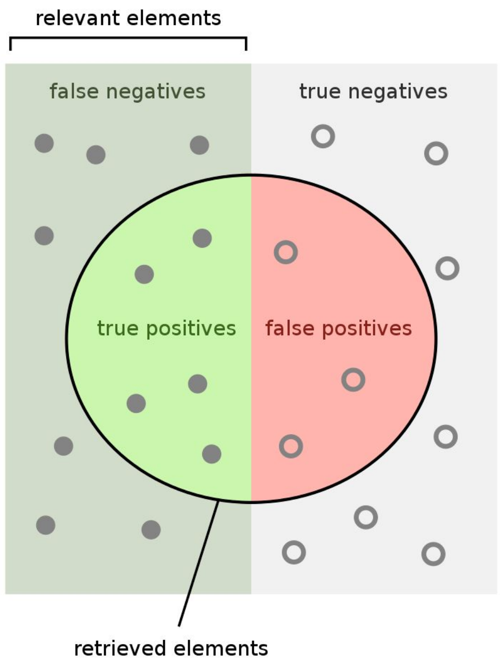


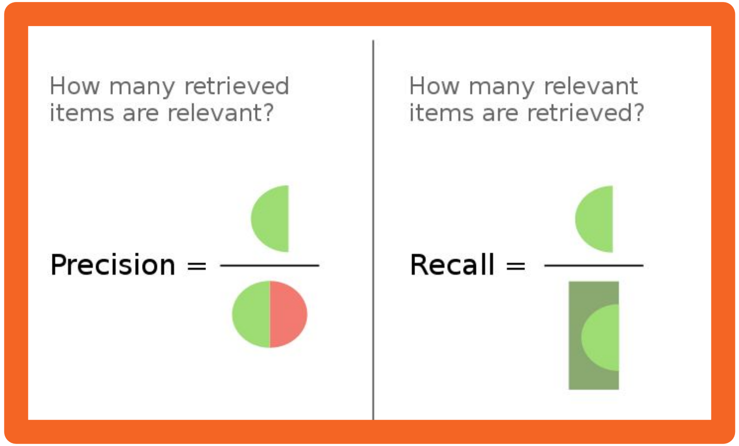


If you always predict y=0, you get 0 recall

https://en.wikipedia.org/wiki/Precision_and_recall

Precision, Recall



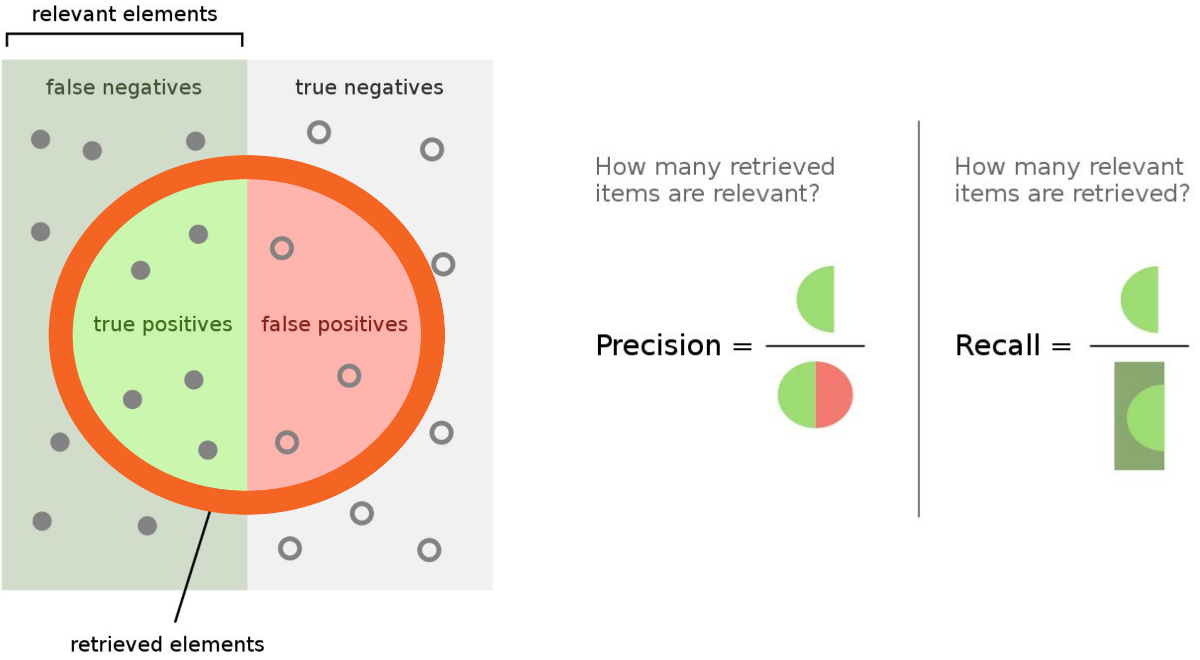


Higher is better for both precision and recall

https://en.wikipedia.org/wiki/Precision_and_recall

Can you get high precision AND recall?

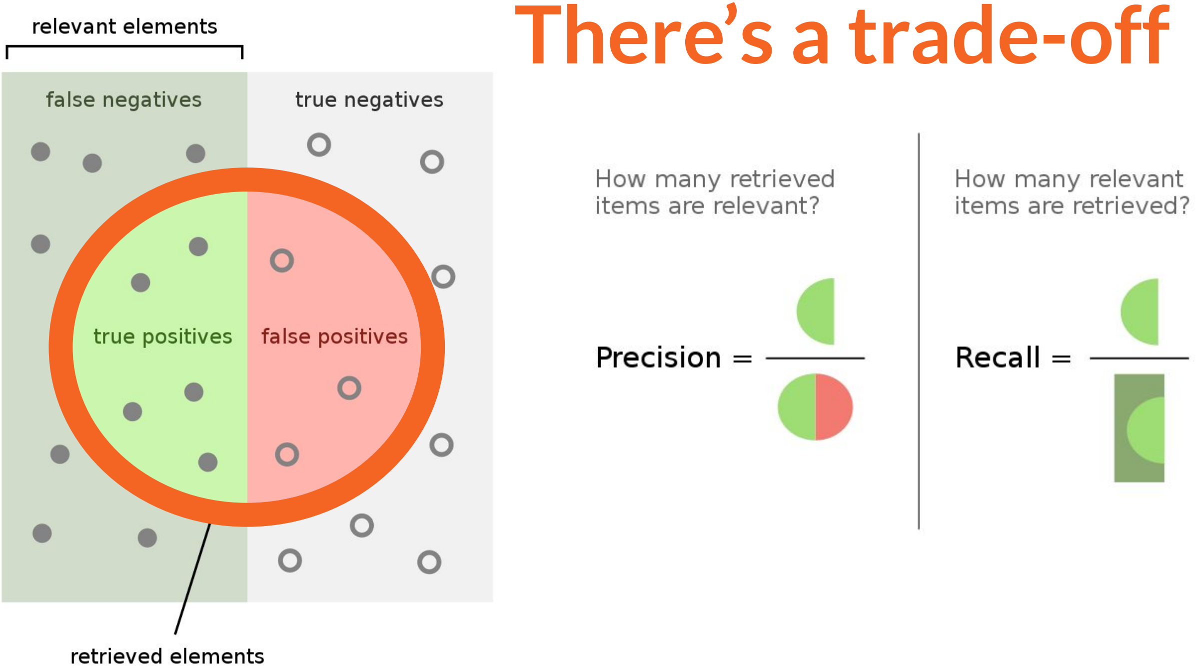
Predicted 1's



https://en.wikipedia.org/wiki/Precision_and_recall

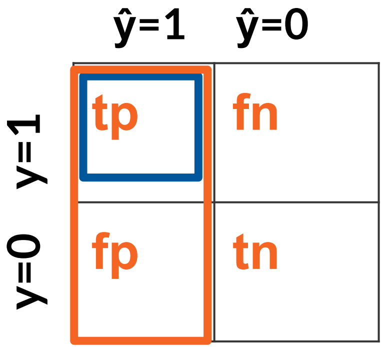
Can you get high precision AND recall?

There's a trade-off

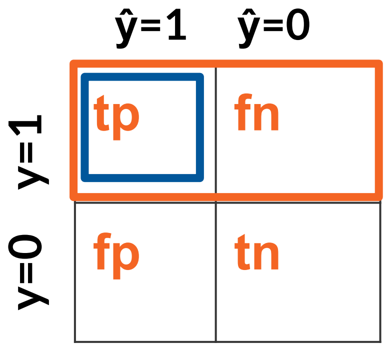


https://en.wikipedia.org/wiki/Precision_and_recall

Precision and Recall



- Precision = tp / (tp + fp)



- Recall = tp / (tp + fn)

Calculate precision, recall, accuracy

	ŷ=1	ŷ=0
y=1	tp 40	fn 10
y=0	fp 10	tn 40

- **Precision** = tp / (tp + fp)
- **Recall** = tp / (tp + fn)
- **Accuracy** = (tp + tn) / (tp + fp + fn + tn)

Calculate precision, recall, accuracy

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	tp 40	fn 10
$y=0$	fp 10	tn 40

- Precision = tp / (tp + fp)
- Recall = tp / (tp + fn)
- Accuracy
  = (tp + tn) / (tp + fp + fn + tn)

Precision = 4/5
Recall = 4/5
Accuracy = 4/5

Calculate precision, recall, accuracy

	ŷ=1	ŷ=0
y=1	tp 4	fn 1
y=0	fp 5	tn 90

- **Precision** = tp / (tp + fp)
- **Recall** = tp / (tp + fn)
- **Accuracy** = (tp + tn) / (tp + fp + fn + tn)

Calculate precision, recall, accuracy

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	tp 4	fn 1
$y=0$	fp 5	tn 90

- Precision = tp / (tp + fp)
- Recall = tp / (tp + fn)
- Accuracy
  = (tp + tn) / (tp + fp + fn + tn)

Precision = 4/9
Recall = 4/5
Accuracy = 94/100

Calculate precision, recall, accuracy

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	tp 0	fn 5
$y=0$	fp 0	tn 95

- **Precision** = tp / (tp + fp)
- **Recall** = tp / (tp + fn)
- **Accuracy**
  = (tp + tn) / (tp + fp + fn + tn)

Calculate precision, recall, accuracy

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	tp 0	fn 5
$y=0$	fp 0	tn 95

- Precision = tp / (tp + fp)
- Recall = tp / (tp + fn)
- Accuracy
  = (tp + tn) / (tp + fp + fn + tn)

Precision = NaN
Recall = 0
Accuracy = 95/100

Calculate precision, recall, accuracy

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	tp 4	fn 1
$y=0$	fp 5	tn 9000

- **Precision** = tp / (tp + fp)
- **Recall** = tp / (tp + fn)
- **Accuracy**
  = (tp + tn) / (tp + fp + fn + tn)

Calculate precision, recall, accuracy

	ŷ=1	ŷ=0
y=1	tp 4	fn 1
y=0	fp 5	tn 9000

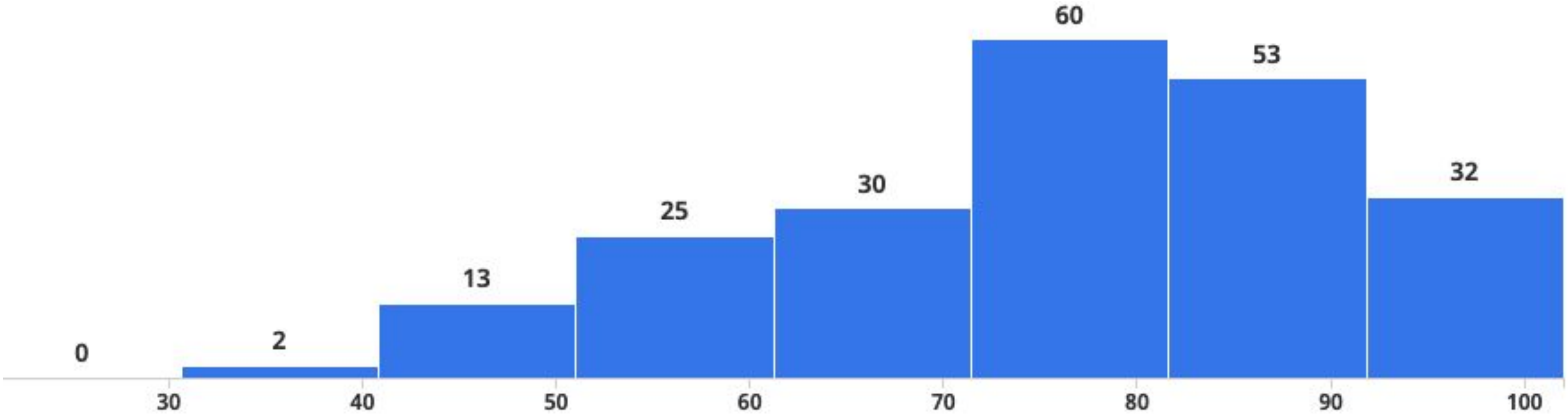
- Precision = tp / (tp + fp)
- Recall = tp / (tp + fn)
- Accuracy = (tp + tn) / (tp + fp + fn + tn)

Precision = 4/9
Recall = 4/5
Accuracy = 9004/9010

Reminder: Extra Credit! 🌟

- Two surveys, +10 points towards HW2 grade for filling out each:
  - Mid-Semester Course Feedback
  - Midterm TA Evaluations
- Surveys due on Oct 13

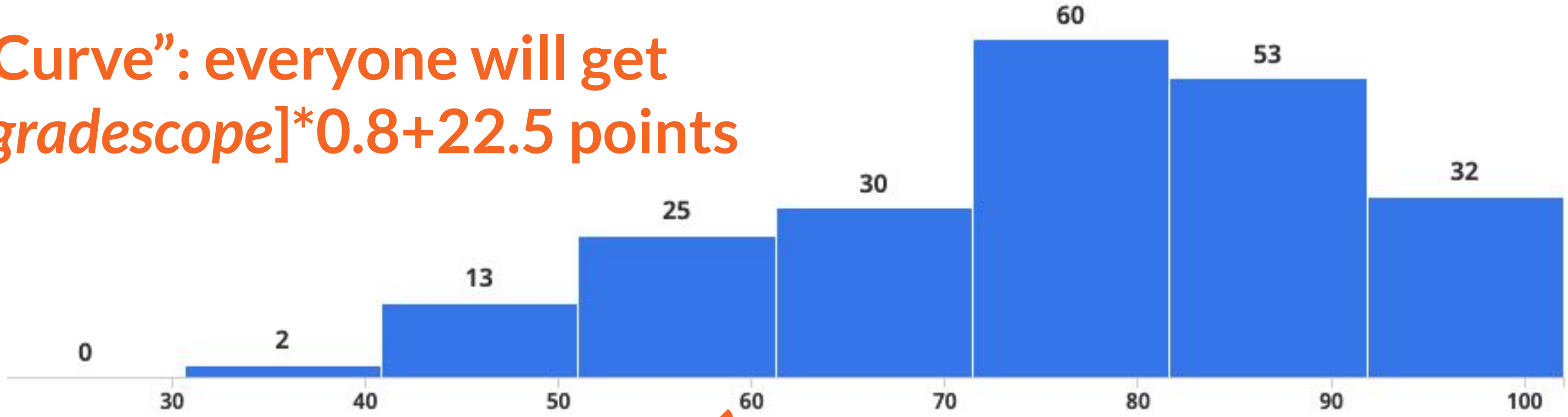
Prelim Grades will be released on Gradescope



Median	Maximum	Mean	Std Dev
78.0	101.5	76.03	14.95

Prelim Grades will be released on Gradescope

"Curve": everyone will get [*gradescope*]*0.8+22.5 points



Median	Maximum	Mean	Std Dev
78.0	101.5	76.03	14.95

84.9

103.7