MACHINE LEARNING IN PYTHON

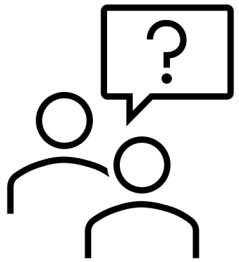
COS 2021 SESSION 4 | HOLLY WIBERG

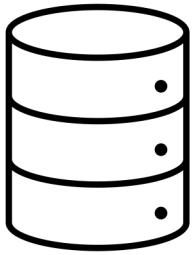
MACHINE LEARNING WORKFLOW FOR SUPERVISED LEARNING

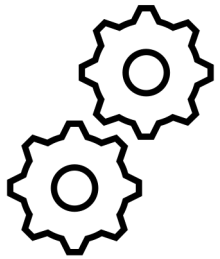
Data Processing

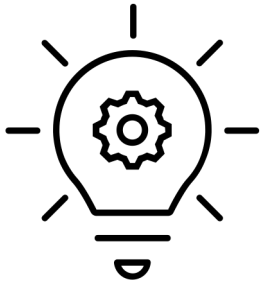
Model Selection

Problem definition → Feature space construction → Model training → Model evaluation + interpretation









SCIKIT-LEARN: THE PYTHON ML TOOLKIT

- Standardization
- Imputation
- Classification
- Regression
- Clustering
- Parameter Tuning
- Model selection
- Model evaluation

I/II: DATA PROCESSING

1. PROBLEM DEFINITION

What is our outcome of interest (y)?

Regression

- Continuous outcome: $y \in \mathbb{R}$
- Model outputs an estimated value
  $\hat{y} \in \mathbb{R}$
- **Example: price of an Airbnb listing**

Classification

- Discrete outcome: binary ($y \in \{0,1\}$) or multi-class
- Model outputs an estimated probability
  $\hat{y} = P(y = 1) \in [0,1]$
- **Example: whether a listing has an elevator**

II. FEATURE SPACE CONSTRUCTION

What are the relevant variables available in the data that we can use for our predictive task?

Data cleaning: many algorithms require *complete*, *numeric* data to run models.

- Categorical features → one-hot encode
  - Add indicator variables for all possible levels of a categorical feature
- Missing data → remove/impute
  - Remove columns/rows that have "too much" missingness
  - Impute remaining missing values: median/mode imputation, K-nearest neighbors imputation, etc.

II. FEATURE SPACE CONSTRUCTION

Listing ID	Max. Guests	City	…	Elevator?	Price
1	NA	NA		yes	105
2	8	Boston		no	150
3	2	Cambridge		yes	90
4	4	Boston		no	130
~~5~~	~~3~~	~~NA~~		~~yes~~	~~NA~~
6	NA	Brookline		no	110
7	6	Boston		no	98
8	2	Cambridge		no	70

Remove missing outcomes

II. FEATURE SPACE CONSTRUCTION

Listing ID	Max. Guests	City	…	Elevator?	Price
1	5	Boston		1	105
2	8	Boston		0	150
3	2	Cambridge		0	90
4	4	Boston		0	130
6	2	Brookline		0	110
7	6	Boston		0	98
8	2	Cambridge		0	70

Impute missing values

Tools: `sklearn.impute.SimpleImputer()`, `sklearn.impute.KNNImputer()`

II. FEATURE SPACE CONSTRUCTION

Listing ID	Max. Guests	City_Boston	City_Brookline	City_Cambridge	…	Elevator?	Price
1	5	1	0	0		1	105
2	8	1	0	0		0	150
3	2	0	0	1		0	90
4	4	1	0	0		0	130
6	2	0	0	1		0	110
7	6	1	1	0		0	98
8	2	0	0	1		0	70

One-hot encode categorical values

Tools: `pd.get_dummies()`

III/IV. MODEL SELECTION

MODEL SELECTION

Model training

- **Train models** using various ML methods.
- **Tune parameters** within each method using the training and validation sets.
  - Parameter tuning is used to prevent our model from overfitting to the training data, making the models more generalizable to unseen data.

Tools: `sklearn.model_selection.GridSearchCV()`

Model evaluation + interpretation

- Compare performance across methods
  - **Regression:** R^2, root mean-squared error (RMSE)
  - **Classification:** Area under the ROC curve (AUC), threshold-based metrics (accuracy, sensitivity, specificity)
- Compare model outputs: how interpretable are the final outputs?

Tools: `sklearn.metrics`

III. MODEL TRAINING

Now that our feature space is clean, we can split the data into train/validation/test sets to prepare for model training:

1. **Training Set:** The sample of the data used to create the predictive model. It is important this subset is large enough to yield statistically significant results. Generally, at least 60% of the data is allocated to the training set.
2. **Validation Set:** The sample of data used to provide an unbiased evaluation of a model on the training dataset while tuning model hyperparameters.
   - *When using cross-validation, we will not explicitly separate the training/validation data. Multiple splits will be used to tune the hyperparameters.*
3. **Testing Set:** The subset of data used to provide an unbiased evaluation of the final model. Evaluation on the testing set is also referred to as **out-of-sample** evaluation.

Tools: `sklearn.model_seletion.train_test_split()`

COMMON ML METHODS

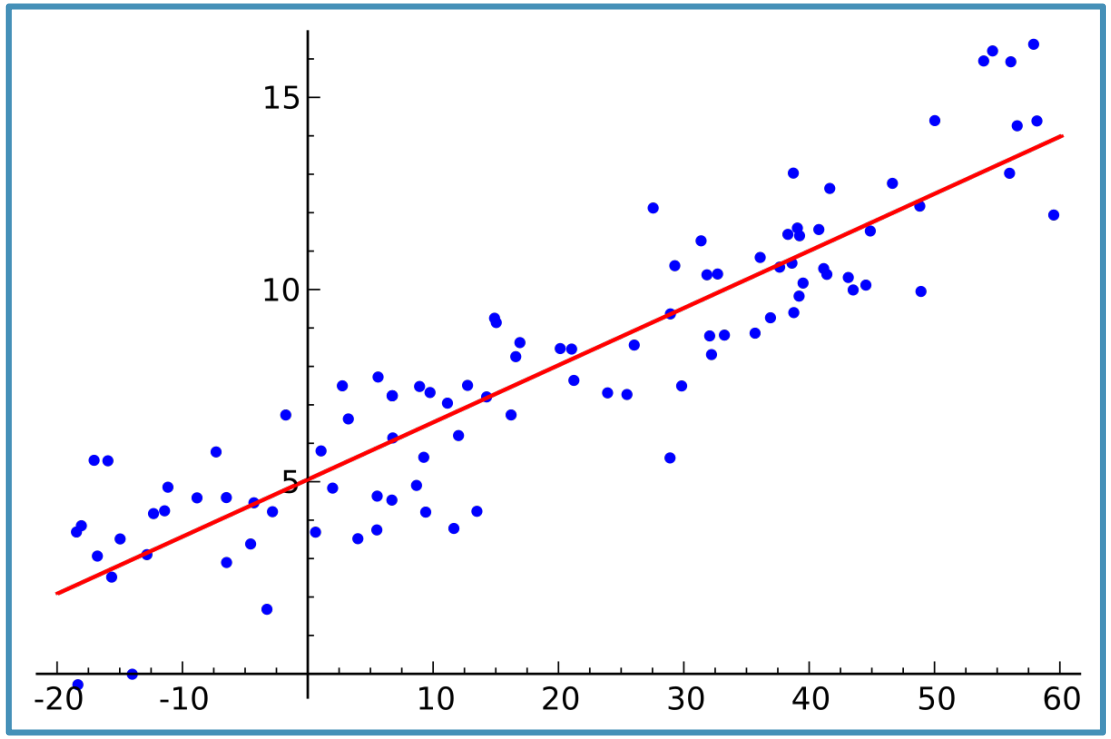
LINEAR MODELS: OLS REGRESSION

Ordinary Least Squares (OLS)

- Predict $y \in \mathbb{R}$ as a linear function of the input variables $(\mathrm{x_i} \in \mathbb{R}^p)$: $\hat{y} = \beta^T x$
- Find the coefficients $\beta \in \mathbb{R}^p$ by minimizing the least squares loss function

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2$$

- We won't talk about the details behind solving this problem, but the optimal β^* can be written in closed form.



LINEAR MODELS: *REGULARIZED* REGRESSION

Regularization is a tool which helps us to avoid overfitting by penalizing model complexity. Mathematically, we add a term to the loss function in the optimization problem to be solved. Our new loss function is:

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2 \; + \boldsymbol{\lambda \Omega(\beta)}$$

LASSO: $\Omega(\beta) = \|\beta\|_1$
Ridge Regression: $\Omega(\beta) = \|\beta\|_2^2$

LASSO and Ridge regression both shrink the elements of the optimal β^* vector towards 0, but in different ways.

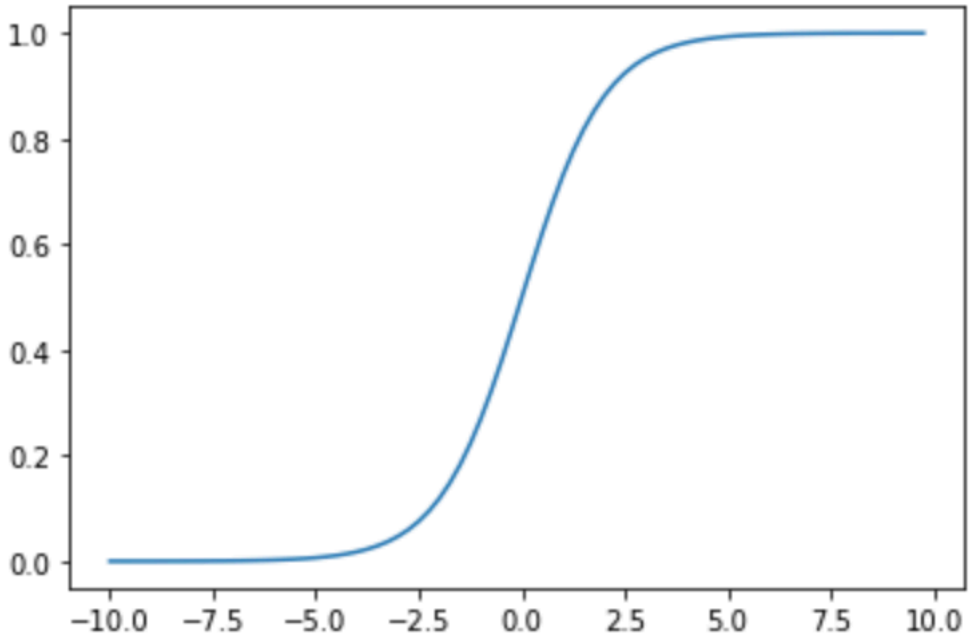
We will focus on LASSO -- which tends to shrink the coefficients so that some are equal to 0. This is nice because it helps us interpret the model by making it ***sparser***.

LINEAR MODELS: LOGISTIC REGRESSION

Logistic Regression is a generalized linear model (GLM) for binary classification. It maps a linear fit through a nonlinear function $f()$ and returns a probability that the outcome class is 1.

$$P(y_i = 1) = f(\beta^T x_i) = \frac{1}{1 + e^{-(\beta^T x_i)}}$$

Since the function stays between zero and one, it can be interpreted as a mapping from predictor values to a probability of being in one of two classes.

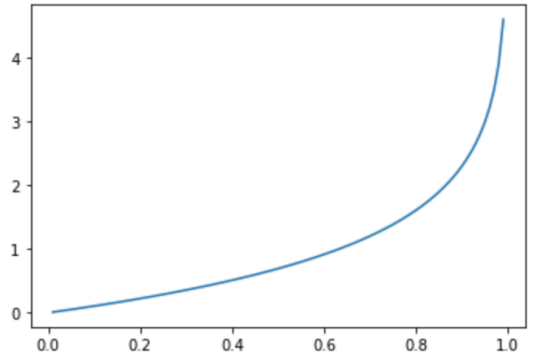


LINEAR MODELS: LOGISTIC REGRESSION

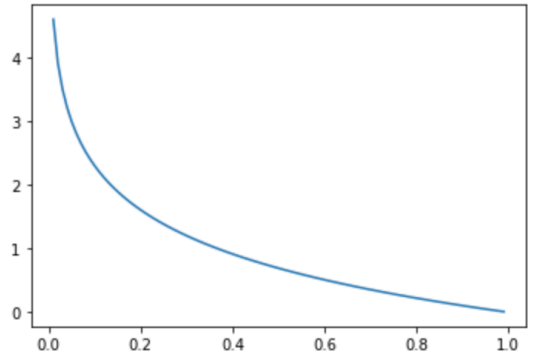
Logistic Regression is a generalized linear model (GLM) for binary classification. It maps a linear fit through a nonlinear function $f()$ and returns a probability that the outcome class is 1.

We find the optimal β by minimizing the logistic loss:

$$\min_{\beta}\left(\frac{1}{n}\sum_{i=1}^{n}-y_i\log f(\beta^T x_i)-(1-y_i)\log\big(1-f(\beta^T x_i)\big)\right)$$

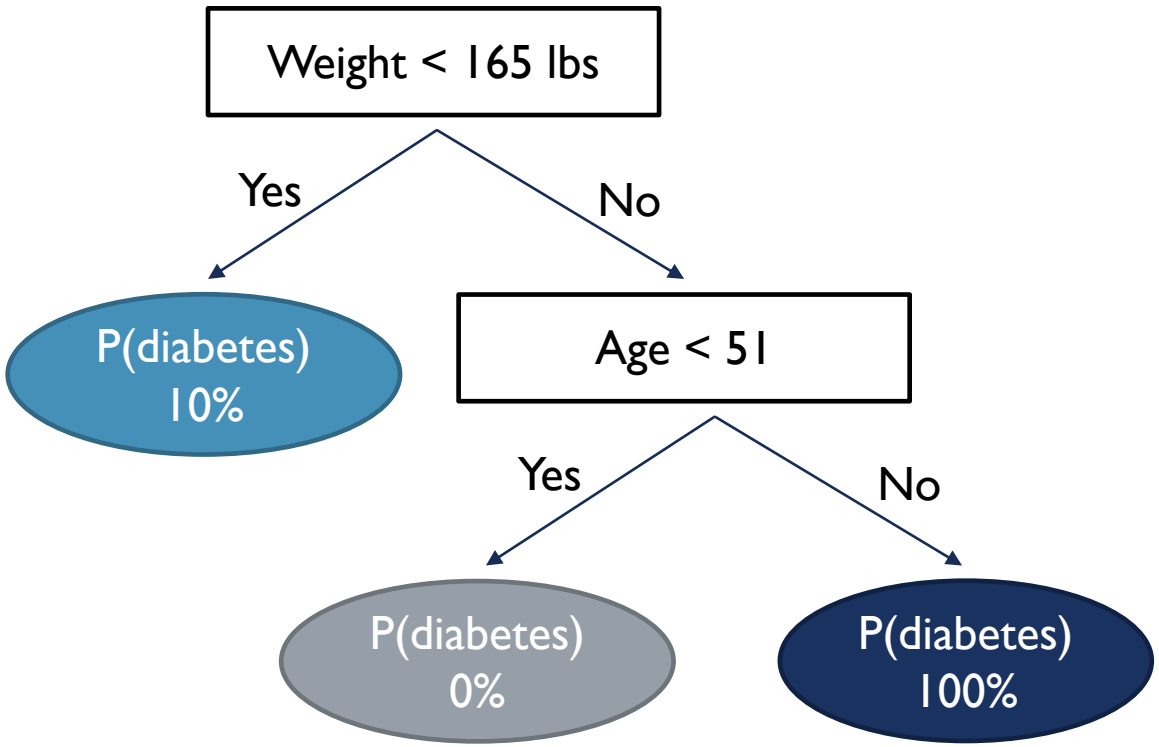


Loss if $y_i = 0$
→ high penalty for predictions near 1



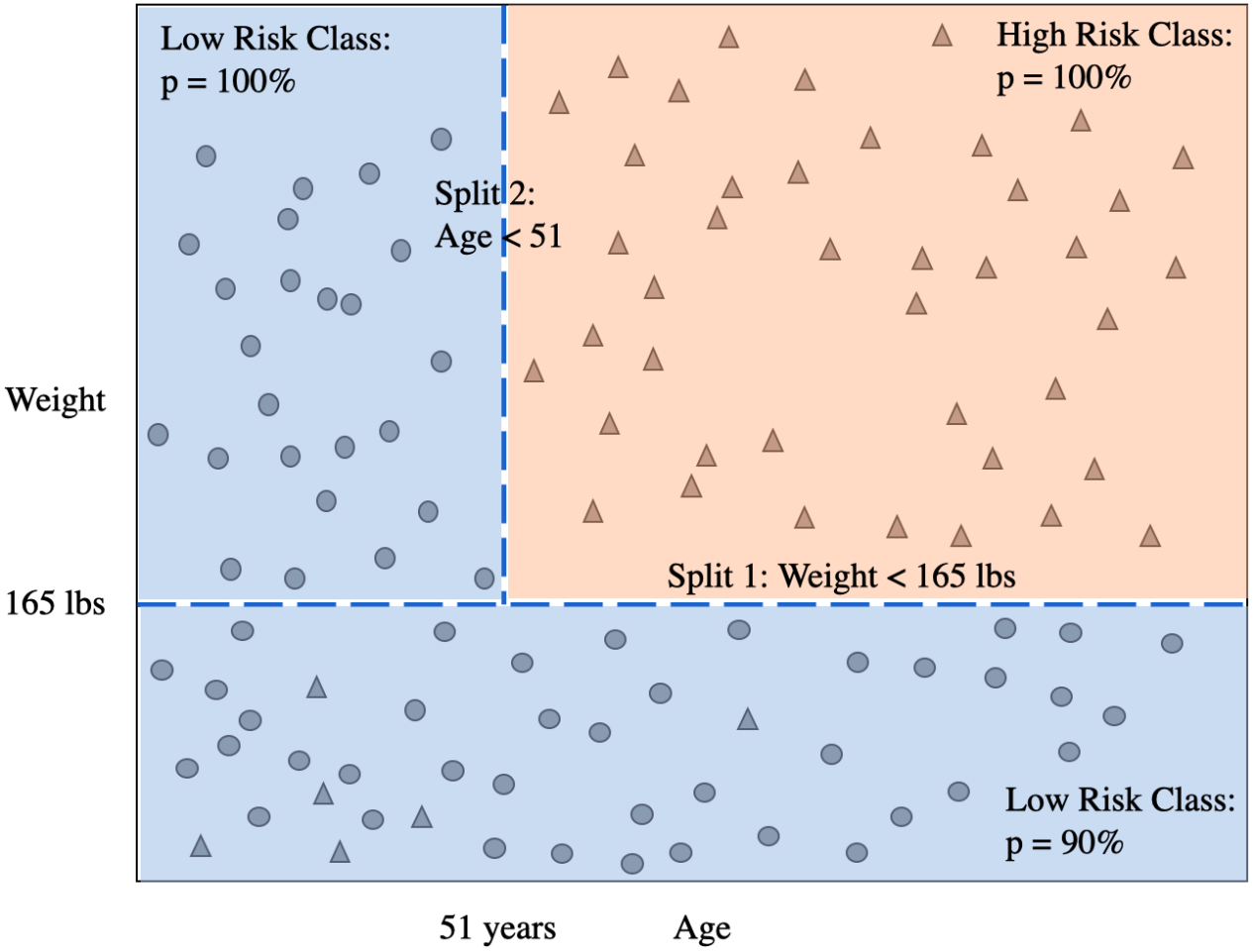
Loss if $y_i = 1$
→ high penalty for predictions near 0

DECISION TREES



- Decision trees partition the feature space through a series of **splits,** such that each observation ends up in a **leaf** node.
  - It is built to **minimize** a **loss function** which quantifies the error of the model. Splits are chosen based on how much they improve the tree's loss.
- Each leaf has a **predicted** y based on the average outcome of the leaf members.
- Various implementations: CART (Breiman 1984), Optimal Classification/Regression Trees (Bertsimas and Dunn 2017).

DECISION TREES



- Decision trees partition the feature space through a series of **splits,** such that each observation ends up in a **leaf** node.
  - It is built to **minimize** a **loss function** which quantifies the error of the model. Splits are chosen based on how much they improve the tree's loss.
- Each leaf has a **predicted *y*** based on the average outcome of the leaf members.
- Various implementations: CART (Breiman 1984), Optimal Classification/Regression Trees (Bertsimas and Dunn 2017).

ENSEMBLE MODELS

- A Random Forest model consist of a **collection of classification trees**, where each one uses a random sample of the training data and predictive variables.
- **Each tree "votes"** on the predicted outcome of every observation, and the final prediction is taken to be the majority vote.

- **Ensemble Method:** Uses a large group of trees to generate predictions. This allows it to capture more complex decision boundaries. XGBoost is another popular method of this type.

Higher Accuracy – Lower Interpretability

III. MODEL TRAINING

Method	Type	Python Implementation	Pros	Cons
Linear/Logistic Regression	Linear	`sklearn.linear_model` - `Lasso()` - `LogisticRegression()`	Highly interpretable.	Does not capture interactions between variables.
Decision Tree	Tree-Based	`sklearn.tree` - `DecisionTreeClassifier()` - `DecisionTreeRegressor()`	Highly interpretable, models nonlinear relationships.	Does not capture linear dependence on variables.
Random Forests	Ensemble	`sklearn.ensemble` - `RandomForestClassifier()` - `RandomForestRegressor()`	Highly performant, models nonlinear relationships.	No single final model, less interpretable.
XGBoost	Ensemble	`xgb` - `XGBClassifier()` - `XGBRegressor()`	State-of-the-art performance, models nonlinear relationships.	No single final model, less interpretable, more parameter tuning.

… and many others!

IV. EVALUATION + INTERPRETATION

IV. MODEL EVALUATION: CONTINUOUS OUTCOME

(Root) Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

R-Squared (R^2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- Proportion of variance in the data which is explained by the model: how much better does the model predict y than a baseline of just predicting the average?
- More interpretable measure of accuracy (maximum value = 1)

IV. MODEL EVALUATION: CLASSIFICATION

Threshold-based metrics: compare actual outcomes to predicted outcomes using a *confusion matrix (classification matrix).* We turn probabilities into 0/1 predictions by prediction 1 if $\hat{y} > \tau$ and 0 otherwise, for a threshold τ

	Predicted Class = 0	Predicted Class = 1
Actual Class = 0	True Negatives (TN) ✓	False Positives (FP) ✗
Actual Class = 1	False Negatives (FN) ✗	True Positives (TP) ✓

- **Accuracy = (TN+TP)/ # of observations**
- A different threshold value (τ) changes the types of errors
  - **Sensitivity = TP/(TP+FN)**
  - **Specificity = TN/(TN + FP)**

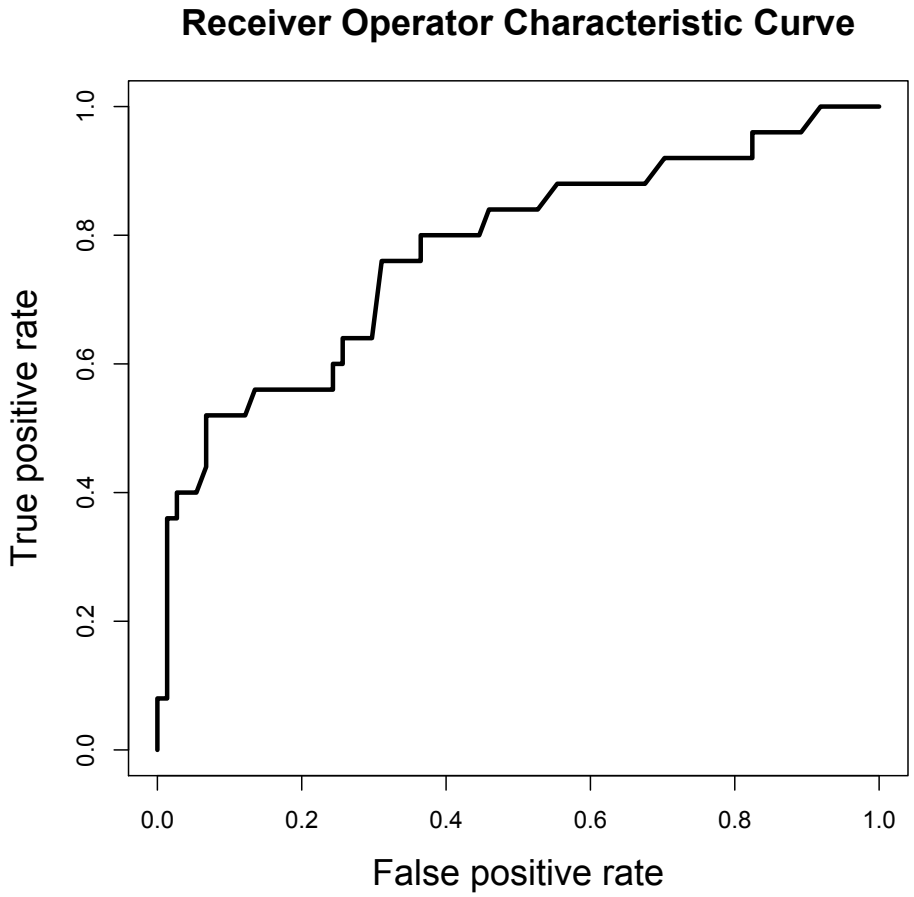
Question: What is the "right" threshold for a prediction task?

IV. MODEL EVALUATION: CLASSIFICATION

ROC Curve: plot true positives vs. false positives across all thresholds $\tau \in [0,1]$

Area under the ROC Curve (AUC) measures how well the model discriminates between positive and negative cases, *independent* of threshold.

- An AUC of 0.5 means the model is random.
- An AUC of 1 means that the model perfectly identifies true positives and negatives.



IV. MODEL INTERPRETATION

Method	How to Interpret?
Linear/Logistic Regression	**Coefficients**: **direction** (positive -> increase prediction) and **magnitude** (but be careful about variable scales!)
Decision Tree	**Tree decision paths**: what characterizes leaves with high (and low) predictions? What dependencies do you find between variables?
Random Forests	**Feature importance plots**: estimate how much each variable contributes to predictions. **SHAP values**: advanced importance estimation.
XGBoost	

DEMO: PREDICTING AIRBNB PRICE AND AMENITIES

Let's move over to Python!