Classification

CAC 350

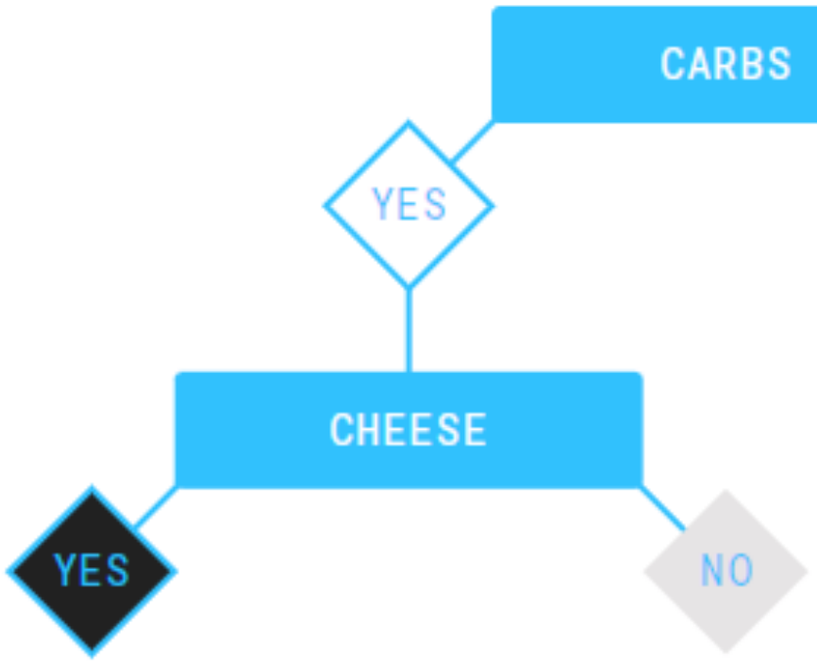
Classification

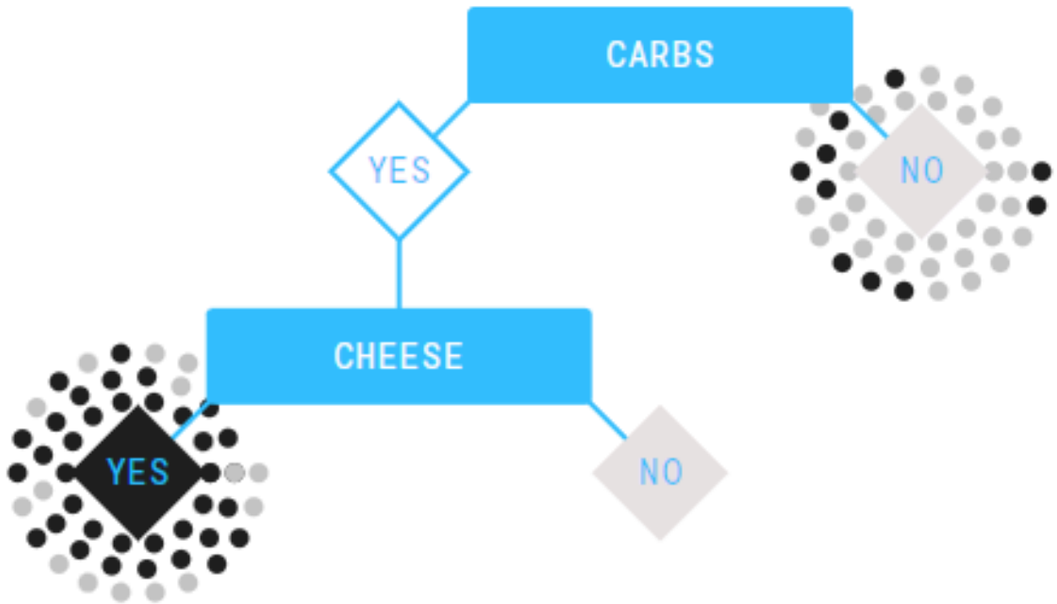
* One of the most common tasks in machine learning
* Classifier: a mapping $X \longrightarrow C$ where $C=\{C_1, C_2, ..., C_k\}$ is a finite and usually small set of class labels
* Binary classification occurs when we only have two classes, usually referred to as positive and negative
* Can you think of an example?

Classification

* Text categorization (spam filtering)
* Fraud detection
* Optical Character Recognition
* Machine vision (face detection)
* NLP (spoken language understanding)
* Market Segmentation (customer responds to promotion)
* Bioinformatics (classify protein according to their function)

Decision Tree





Let's Build our own...

	sex	mask	cape	tie	ears	smokes	class
			training data				
batman	male	yes	yes	no	yes	no	Good
robin	male	yes	yes	no	no	no	Good
alfred	male	no	no	yes	no	no	Good
penguin	male	no	no	yes	no	yes	Bad
catwoman	female	yes	no	no	yes	no	Bad
joker	male	no	no	no	no	no	Bad
			test data				
batgirl	female	yes	yes	no	yes	no	??
riddler	male	yes	no	no	no	no	??

Rob Schapire, Princeton
https://www.cs.princeton.edu/~schapire/talks/picasso-minicourse.pdf

Primary Tasks

* Figure out the rule(s) we want to split on
* Split the data based on the rule
* Repeat until leaves are pure
* Problem: Choosing the best rule
* Suggestion: Choose rule leading to greatest increase in "purity"

What would you like to do?

Male

Batman
Robin
Alfred
Joker
Penguin

Sex

Female

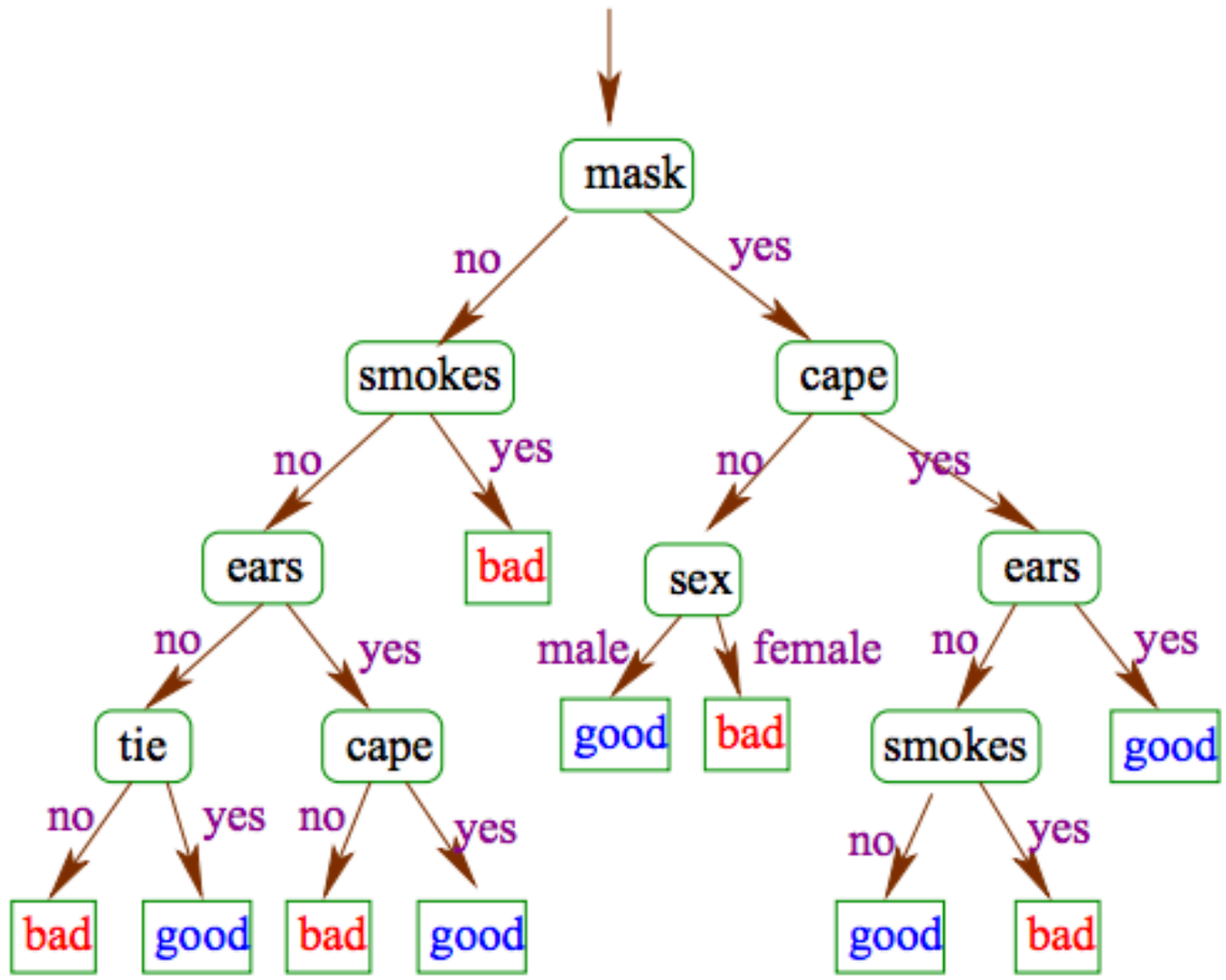
Bad:
Catwoman

Smokes

Bad:
Penguin

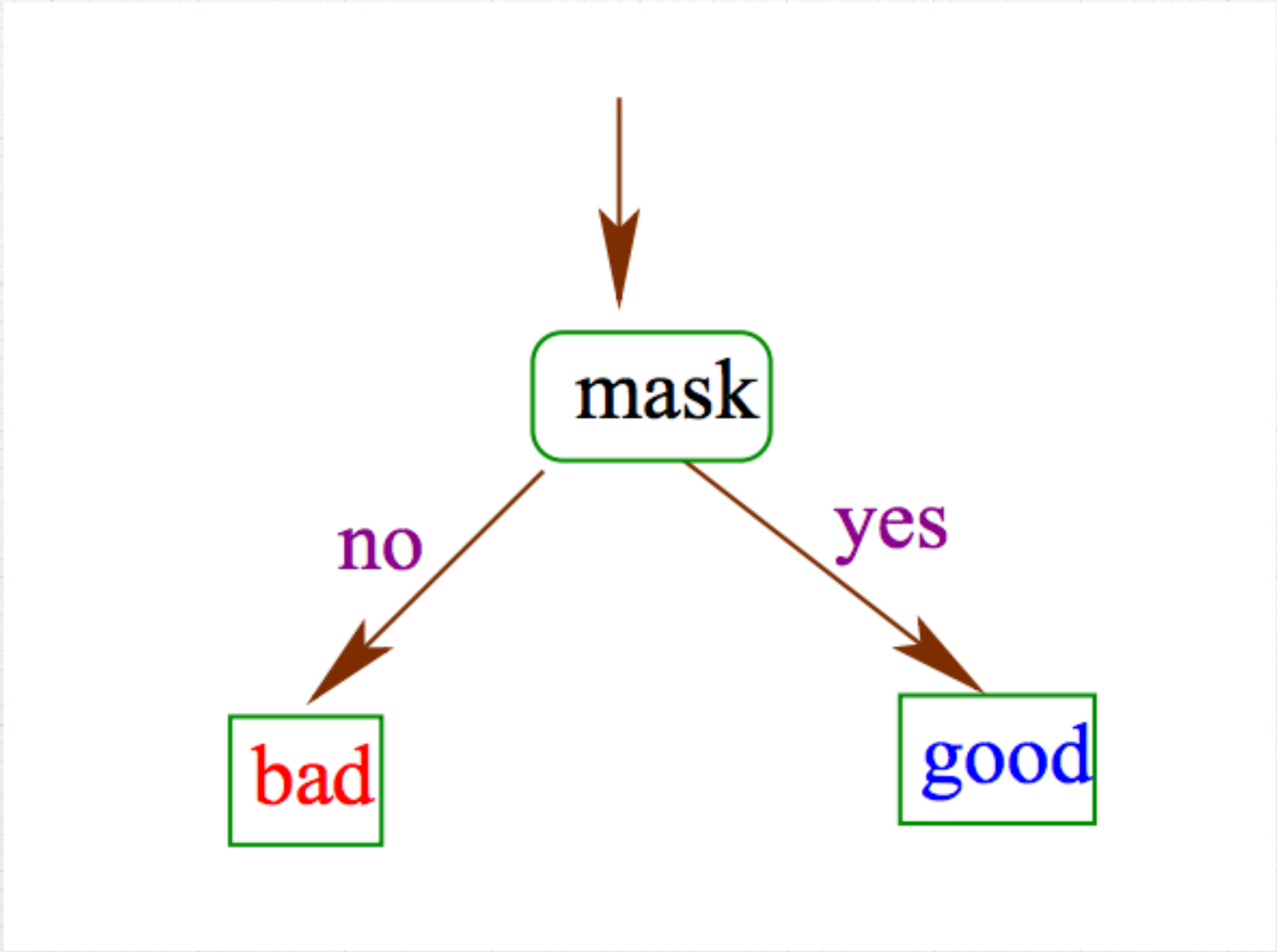
Batman
Robin
Alfred
Joker

Perfect Classifier



Overfit

Simplest Classifier



Underfit

Binary Classification

Training a Binary Classifier

* Book is using the Modified National Institution for Standards and Technology (MNIST) dataset
* Load the dataset
* Classify whether or not a number is a 5

jupyter

ML Algorithms used for Classification

* Logistic Regression
* Naïve Bayes
* Stochastic Gradient Descent
* K-Nearest Neighbors
* Decision Tree
* Random Forest
* Support Vector Machine

Stochastic Gradient Descent

* Let's look at Gradient Descent first...
* Gradient = slope of a function
* Goal is to find the values of x where y is the lowest
* Iterative process, so if you have one million data samples, you have to use all one million of them every iteration - what do we think about that?

Stochastic Gradient Descent

* Stochastic means random probability
* It's a more efficient means of using Gradient Descent because the algorithm chooses a random value from the dataset for each iteration
* Much less computationally expensive
* Has more noise, but ultimately gets the minimum value
* Well suited for online learning (on the fly)

Gradient Descent

* Other options include
  * Batch Gradient Descent
  * Mini-batch Gradient Descent

jupyter

Measuring Accuracy

Cross-Validation

* Splits the data into k distinct subsets (folds)
* Trains and evaluates the model k times, picking a different fold for evaluation every time and training on the other k-1 evaluation scores
* Less biased or less optimistic estimate than some other methods (i.e., simple train/test split)

Steps

* Shuffle the dataset randomly
* Split the dataset into k groups
* For each unique group
    * Take the group as a hold out or test data set
    * Take the remaining groups as a training data set
    * Fit a model on the training set and evaluate it on the test set
    * Retain the evaluation score and discard the model
* Summarize the skill of the model using the sample of model evaluation scores

Folds

* Folds are distinct subsets of a dataset
* Typically used in cross-validation

Choosing a K

* Three approaches:
    * Representative: value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset
    * k=10: value found through experimentation to generally result in a model skill estimate with low bias and modest variance
    * k=n: n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset (one train/test split)

Implementing Cross-Validation

```
from sklearn.model_selection import StratifiedKFold
from sklearn.base import clone

skfolds = StratifiedKFold(n_splits=3, random_state=42)

for train_index, test_index in skfolds.split(X_train, y_train_5):
    clone_clf = clone(sgd_clf)
    X_train_folds = X_train[train_index]
    y_train_folds = y_train_5[train_index]
    X_test_fold = X_train[test_index]
    y_test_fold = y_train_5[test_index]

    clone_clf.fit(X_train_folds, y_train_folds)
    y_pred = clone_clf.predict(X_test_fold)
    n_correct = sum(y_pred == y_test_fold)
    print(n_correct / len(y_pred))  # prints 0.9502, 0.96565 and 0.96495
```

jupyter

Confusion Matrix

* Count the number of times instances of class A are classified as class B

Pre-dicted class \ Actual class	Cat	Dog
Cat	**5**	2
Dog	3	**3**