HPE DSI 311
Introduction to Machine Learning

Summer 2021

Instructor: Ioannis Konstantinidis

UNIVERSITY of HOUSTON
DIVISION OF RESEARCH
HEWLETT PACKARD ENTERPRISE DATA SCIENCE INSTITUTE

Overview

- Metrics and Scoring:
  - Confusion Matrix
  - Error Functions
  - Regularization
- Hands-on examples
  - Classification
  - Regression

What is a model?

Linear Multivariate Regression

Statistics:

$$y = a + b_0 X_0 + b_1 X_1 + b_2 X_2$$ (equation notation)

Linear Multivariate Regression

Statistics:

$$y = a + b_0 X_0 + b_1 X_1 + b_2 X_2$$ (equation notation)

Math:

$$\mathbf{y} = \mathbf{X}\,\beta + \alpha$$ (matrix notation)

Linear Multivariate Regression

Statistics:

$$y = a + b_0 X_0 + b_1 X_1 + b_2 X_2$$ (equation notation)

Math:

$$\mathbf{y} = \mathbf{X}\,\beta + \alpha$$ (matrix notation)

Computer Science:

```
y = Model(X, b, a)
```
(functional notation)

Linear Multivariate Regression

Statistics:

$$y = a + b_0 X_0 + b_1 X_1 + b_2 X_2$$ (equation notation)

Math:

$$\mathbf{y} = \mathbf{X}\,\beta + \alpha$$ (matrix notation)

Computer Science:

```
Model.fit(X,y)
y = Model.predict(X)
```

(object oriented notation)

Linear Multivariate Regression

Statistics:

$$y = a + b_0 X_0 + b_1 X_1 + b_2 X_2$$

(equation notation)

Math:

$$\mathbf{y} = \mathbf{X}\,\beta + \alpha$$

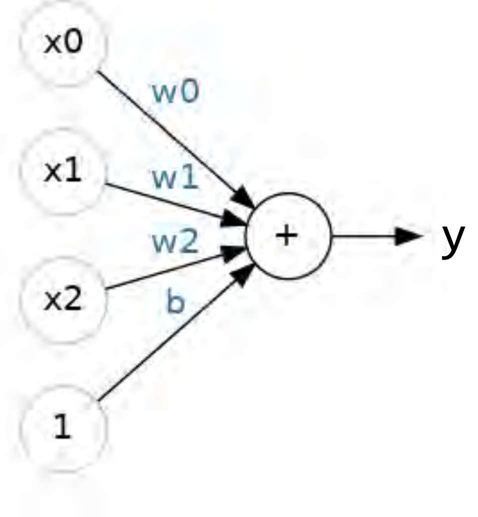
(matrix notation)

Computer Science:

```
Model.fit(X,y)
y = Model.predict(X)
```

(object oriented notation)

Deep Learning/ ML:



(network notation)

You say to-may-to, I say to-mah-to

<u>Math:</u>

y, X **->** variables

α, β **->** parameters

<u>Statistics:</u>

y, X_i **->** variables

a, b_i **->** parameters

<u>Computer Science:</u>

X , y **->** parameters when fitting

b , a **->** parameters when predicting

called weights w for networks

How do you "fit" a model?

Assessing model fitness – supervised ML

Calculate the **parameter values** that make model predictions fit the training data **most closely**

Assessing model fitness – supervised ML

Calculate the **parameter values** that make model predictions match the training data **most closely**

Naïve solution: Exhaustive Search

```
Input: X_train, y_train, Model

For [b, a] in someParameterSpace
   y_predict                <- Model( X_train, b, a )
   penaltyList.append       <- Penalty( y_predict, y_train )

Output: [b_fit, a_fit] = argmin(penaltyList)
```

Assessing model fitness – supervised ML

Calculate the **parameter values** that make the model predictions match the training data **most closely**

Naïve solution: Exhaustive Search

```
Input: X_train, y_train, Model

For [b, a] in someParameterSpace
   y_predict               <- Model( X_train, b, a )
   penaltyList.append      <- Penalty( y_predict, y_train )

Output: [b_fit, a_fit] = argmin(penaltyList)
```

Need an appropriate Penalty() for y

Assessing model fitness – unsupervised ML

Calculate the training **data points** that **are closest to** the new data point

Assessing model fitness – unsupervised ML

Calculate the training **data point** that **is closest to** the new data point

Need an appropriate Penalty() for X

Assessing model fitness

Penalty() for y : compare N pairs (y_predict, y_train)

- Metric / Objective / Cost / Loss function

Penalty() for X : compare two N-dimensional points

- Similarity / Affinity

Testing model performance

Penalty() for y : compare N pairs (y_predict, y_train)

- Scoring / Error function

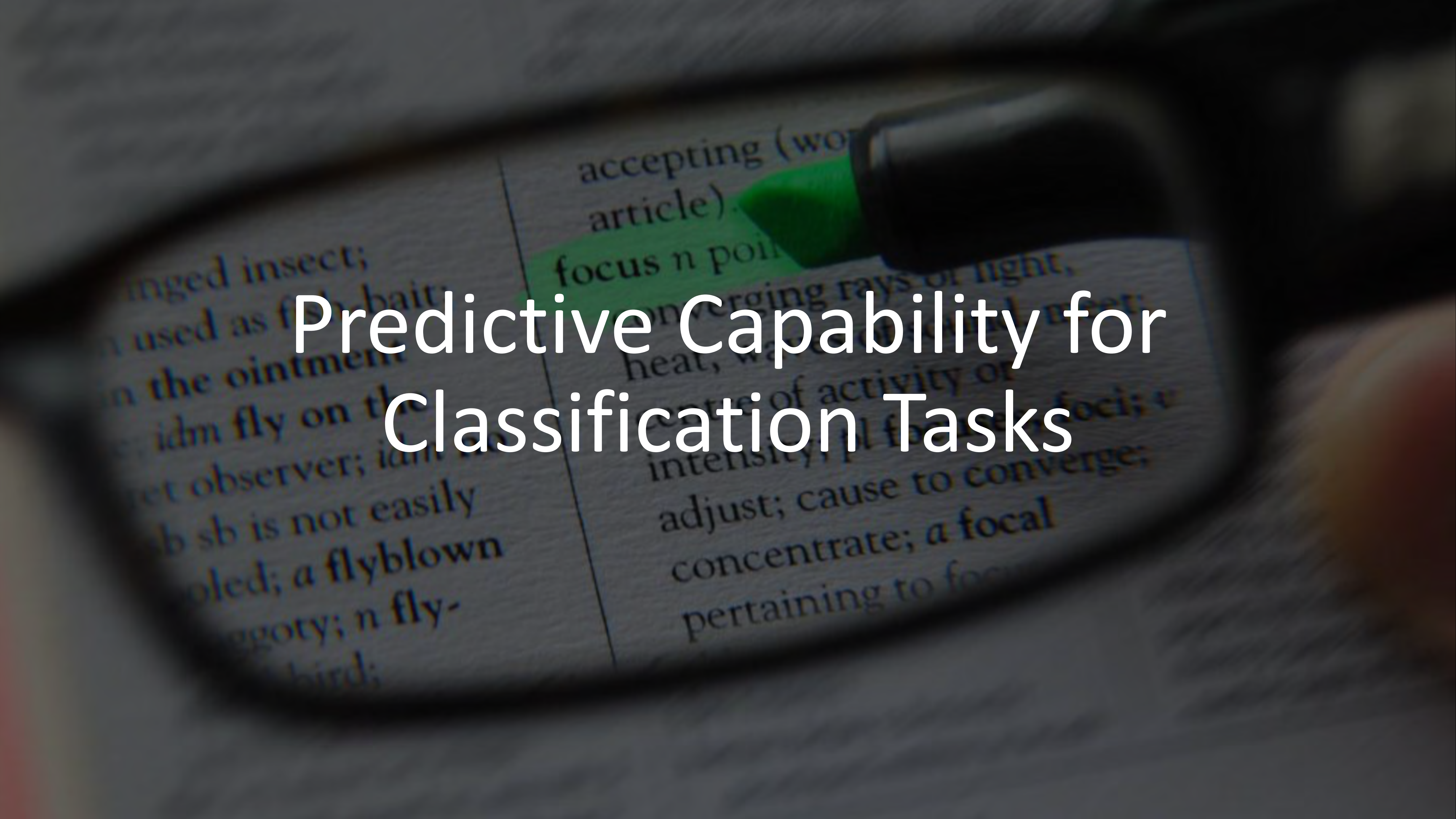
Penalty() for X : compare two N-dimensional points

- Distance

Penalty()

Metric function for assessing fitness during training
Scoring function for testing performance during evaluation

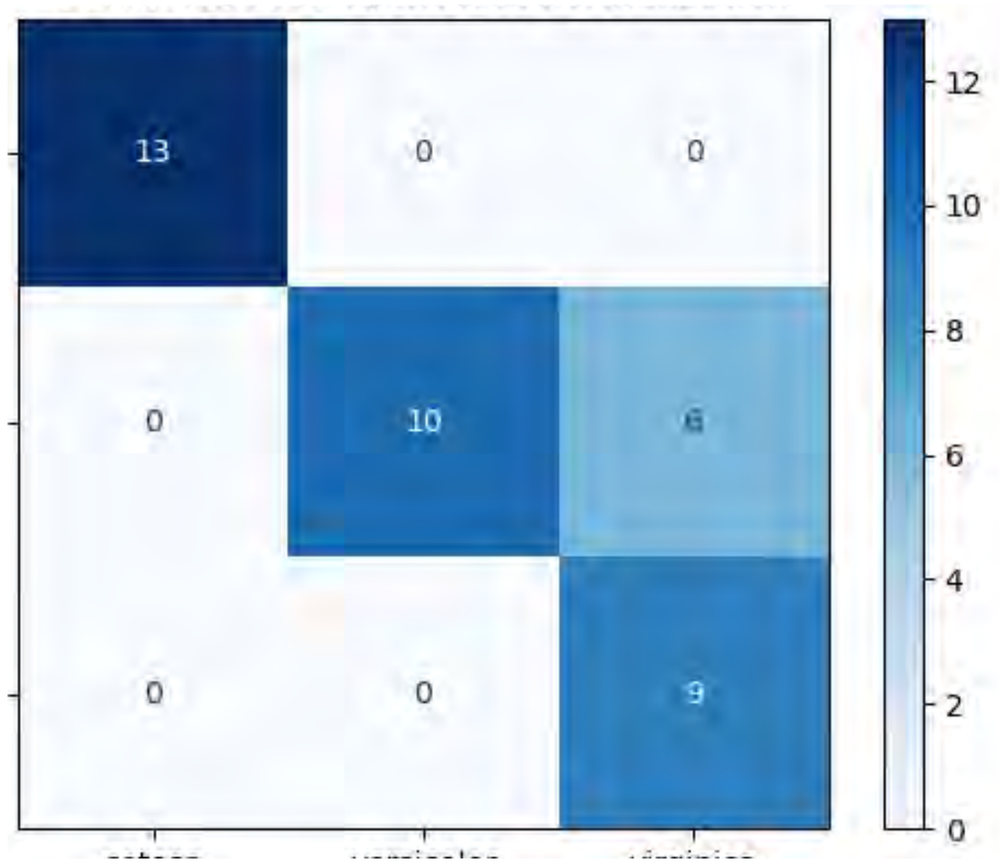
How to define penalty functions



Predictive Capability for Classification Tasks

Confusion Matrix

Confusion Matrix



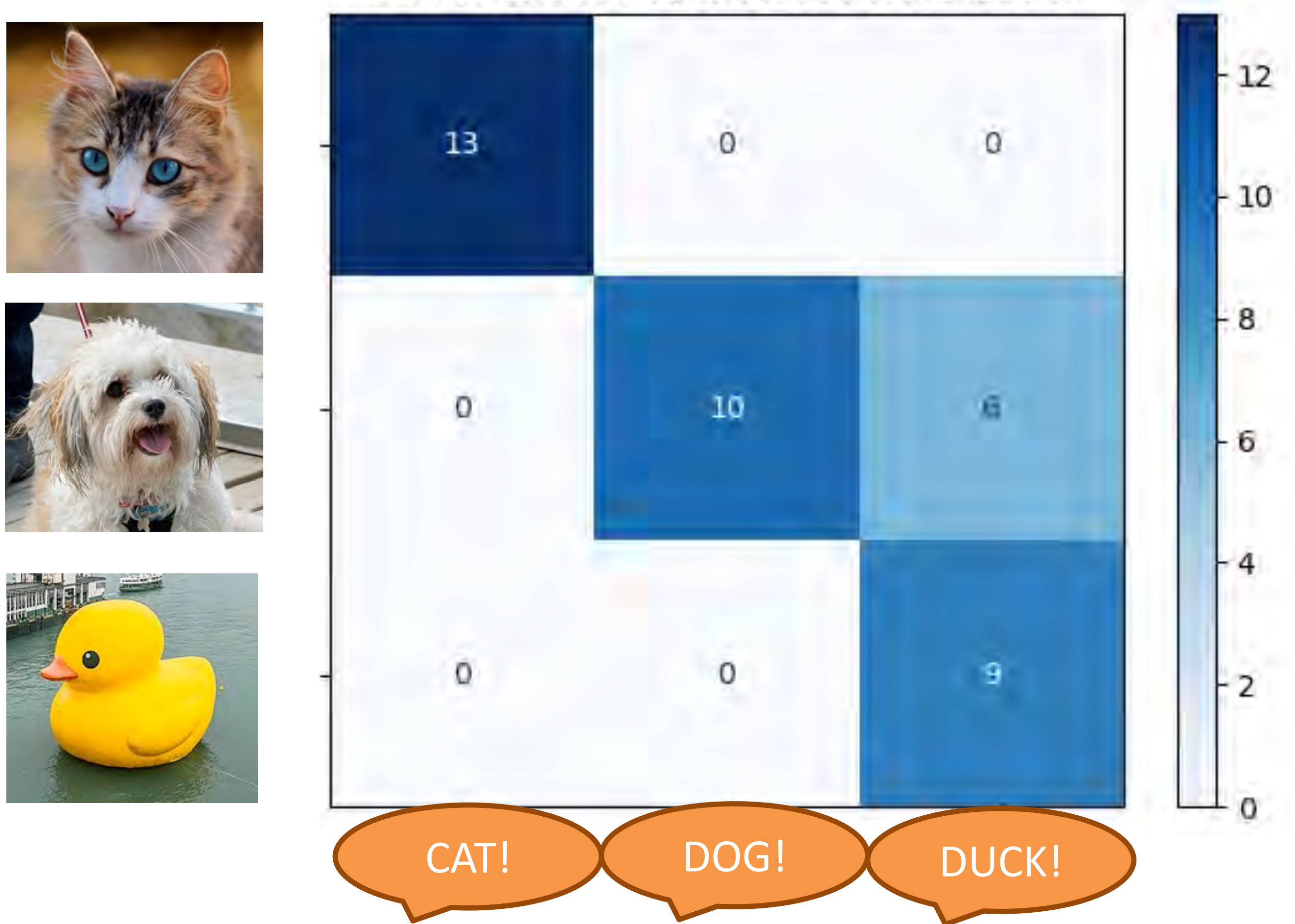
Confusion Matrix

Columns: predictions made by the classifier (labels y)

Rows: actual observations (points X)

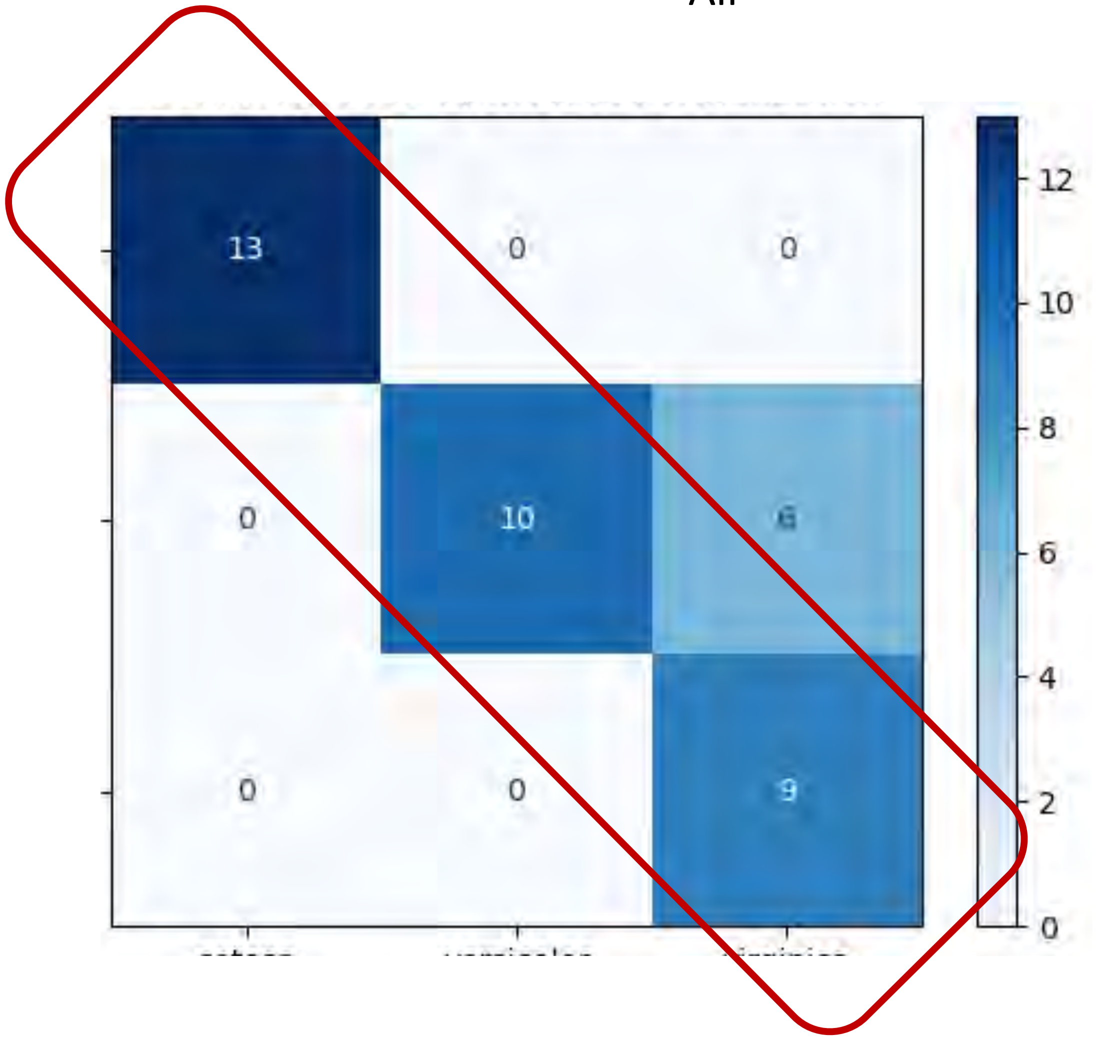
Confusion Matrix

- Diagonal: # of points for which predicted label = true label
- Off-diagonal: # of points that are mislabeled by the classifier
- The higher the diagonal values of the confusion matrix, the better

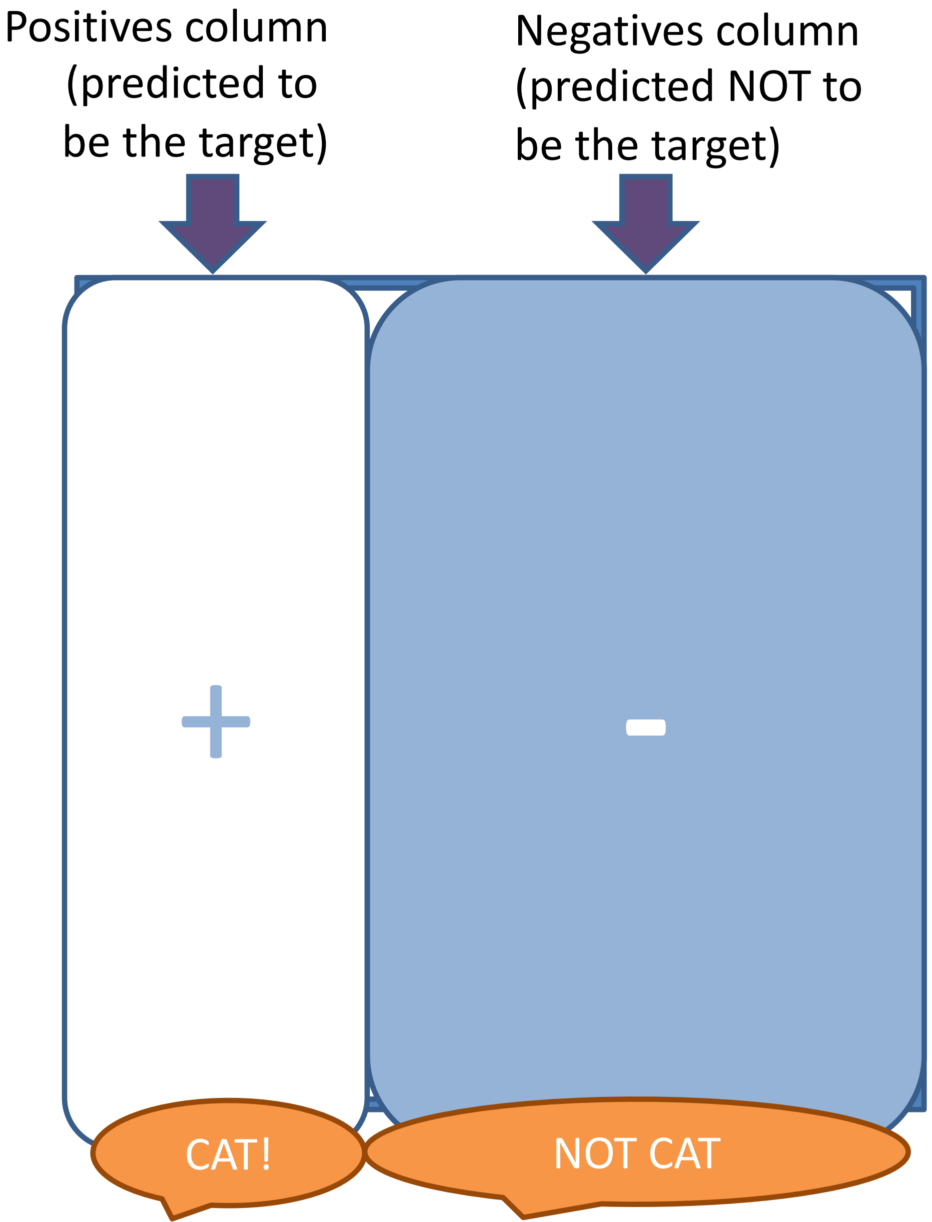


Confusion Matrix

$$\textbf{Accuracy} = \frac{\text{Diagonal}}{\text{All}}$$

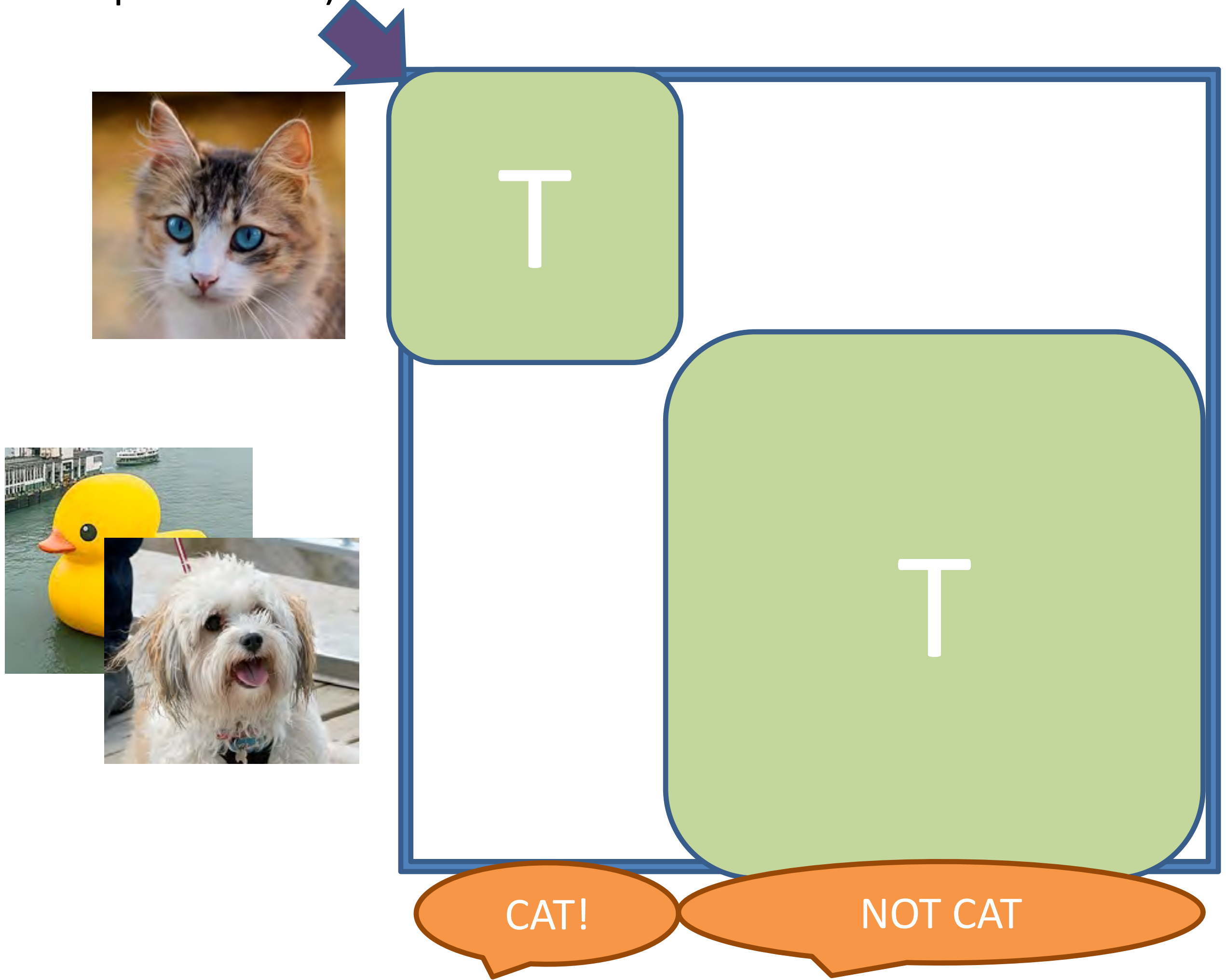


Focus on a single label: is it a cat?

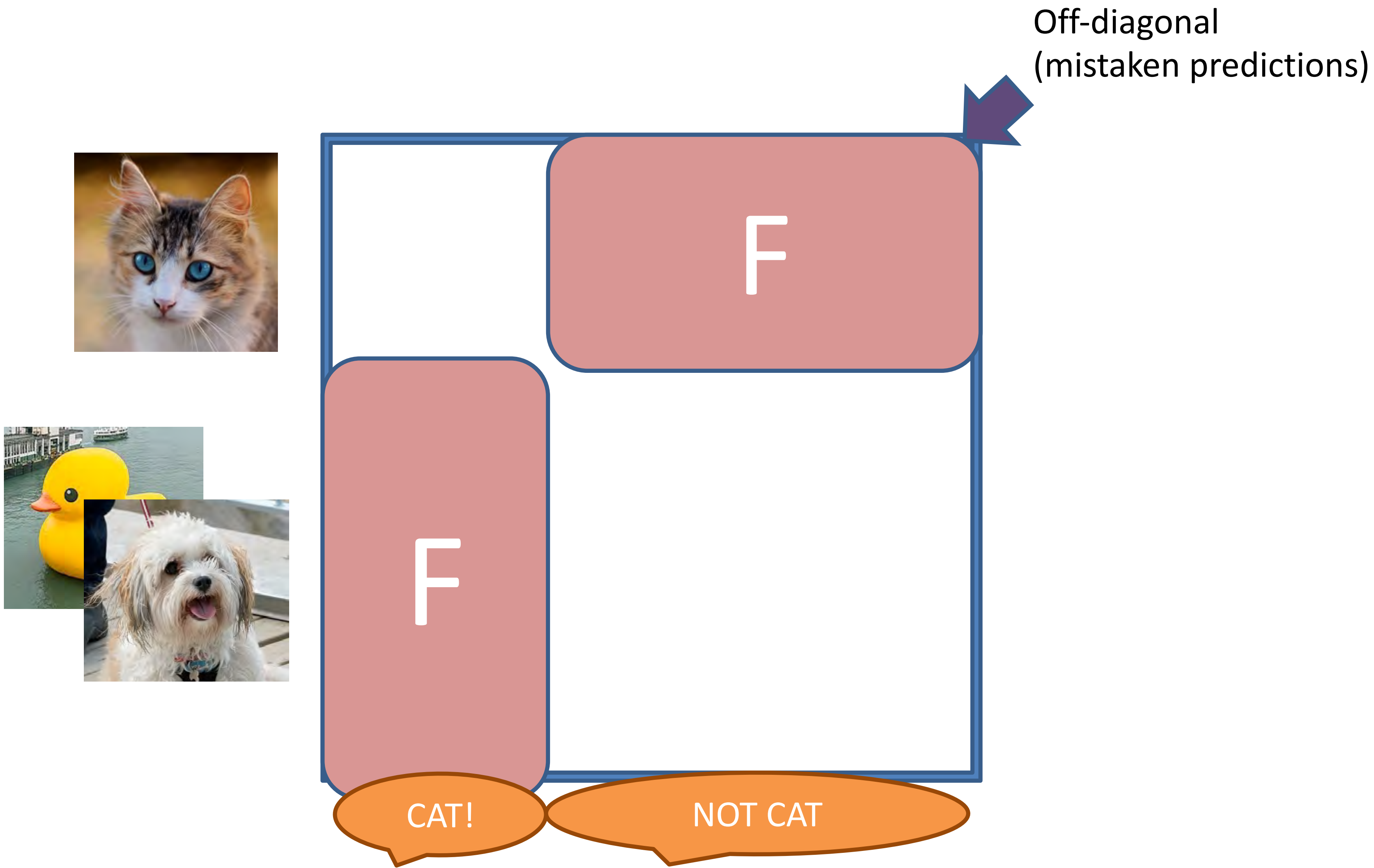


Focus on a single label: is it a cat?

Diagonal
(accurate predictions)



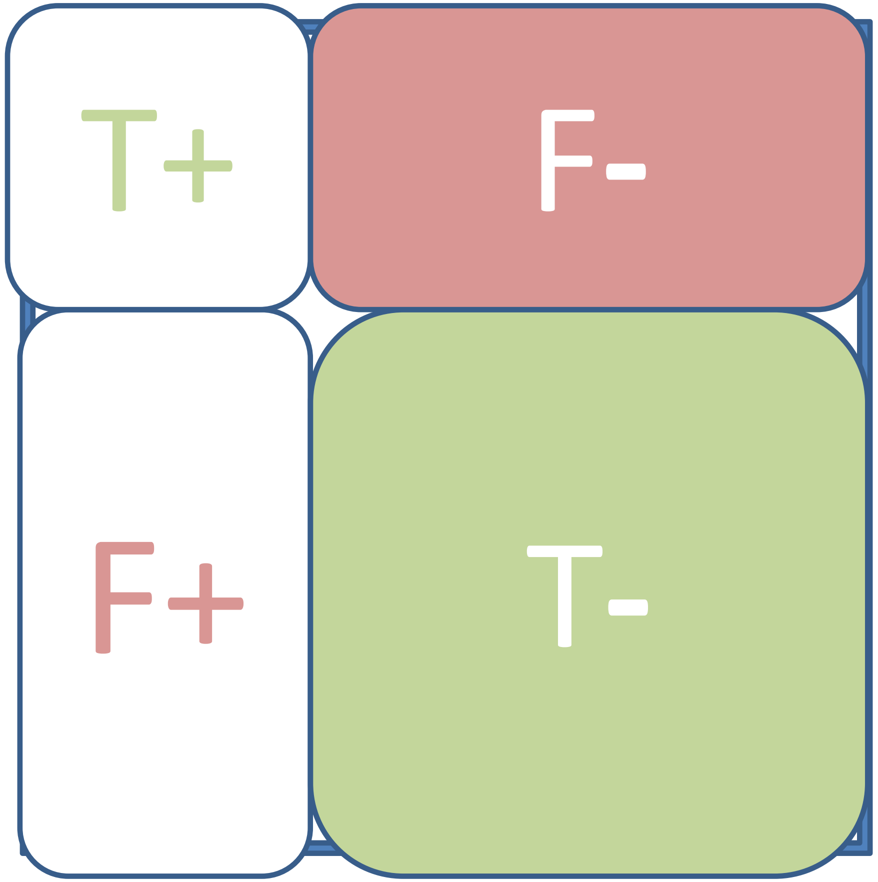
Focus on a single label: is it a cat?



Focus on a single label: is it a cat?

Columns: predictions made by the classifier (labels y)

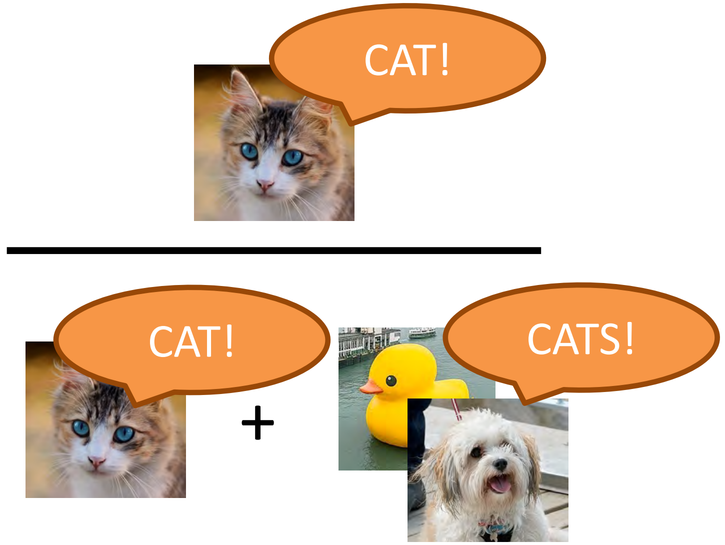
Rows: actual observations (points X)



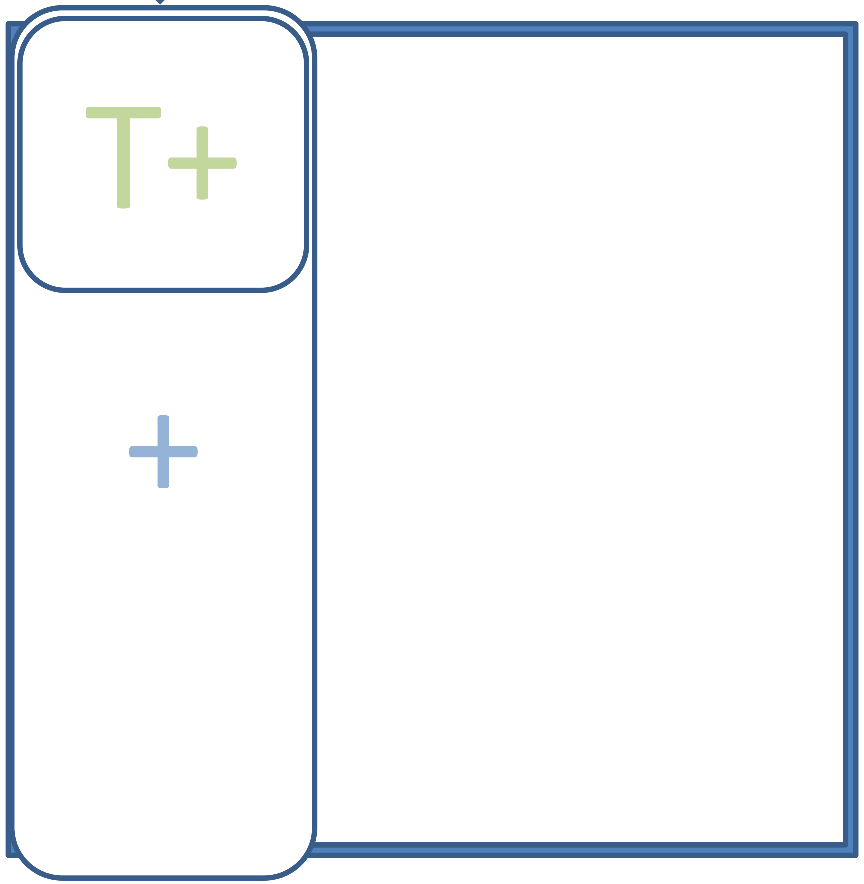
Focus on a single label: is it a cat?

Focus on a single label: Precision

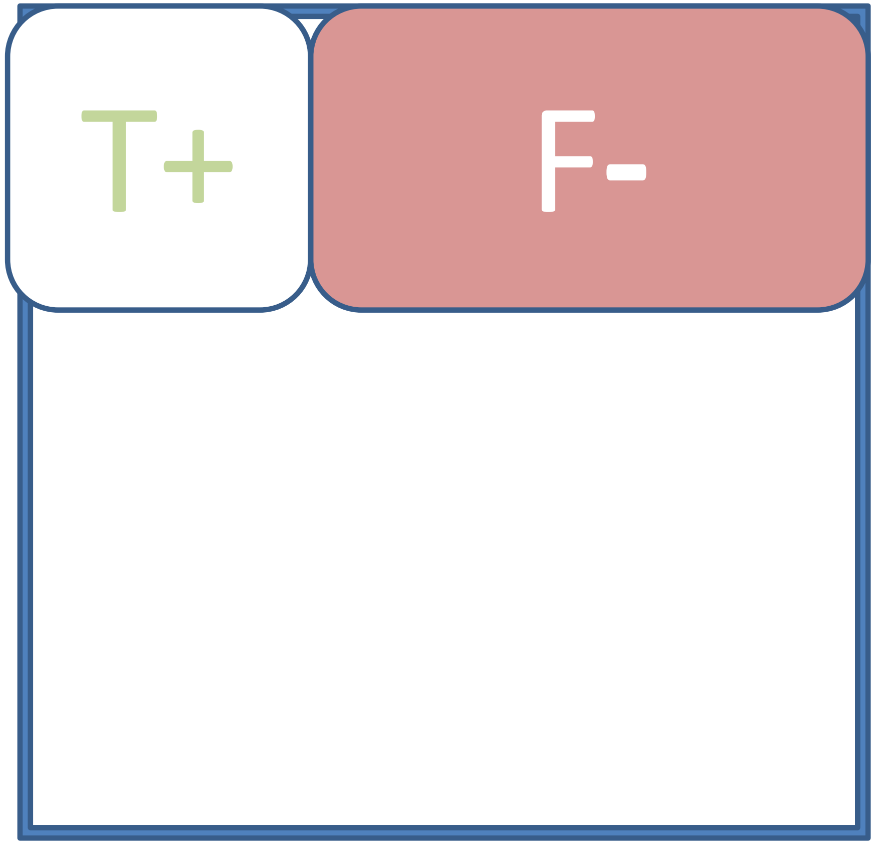
$$\text{Precision}=\frac{\boxed{T+}}{\boxed{+}}$$



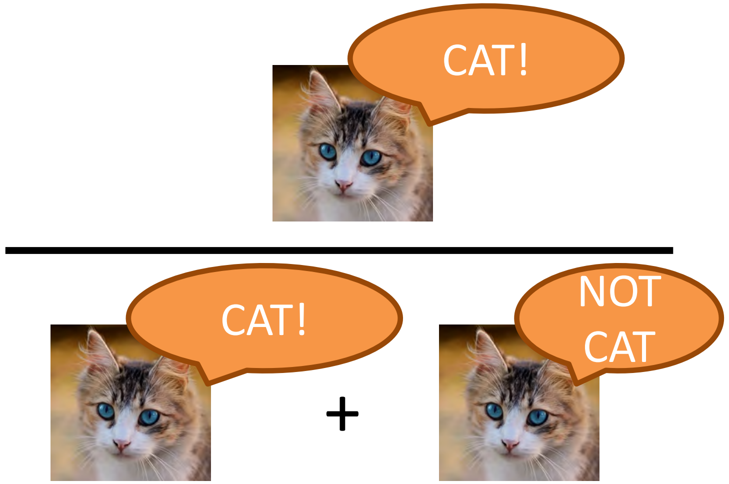
Positives column (predicted to be the target)



Focus on a single label: Recall

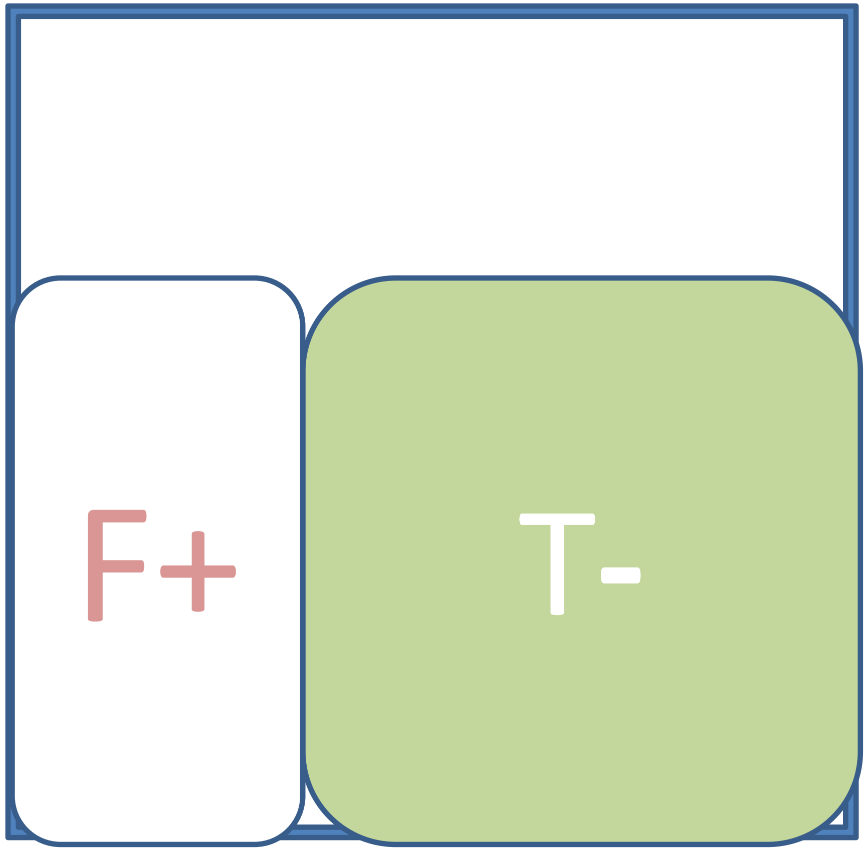


$$\text{Recall} = \frac{T+}{T+ \;+\; F-}$$

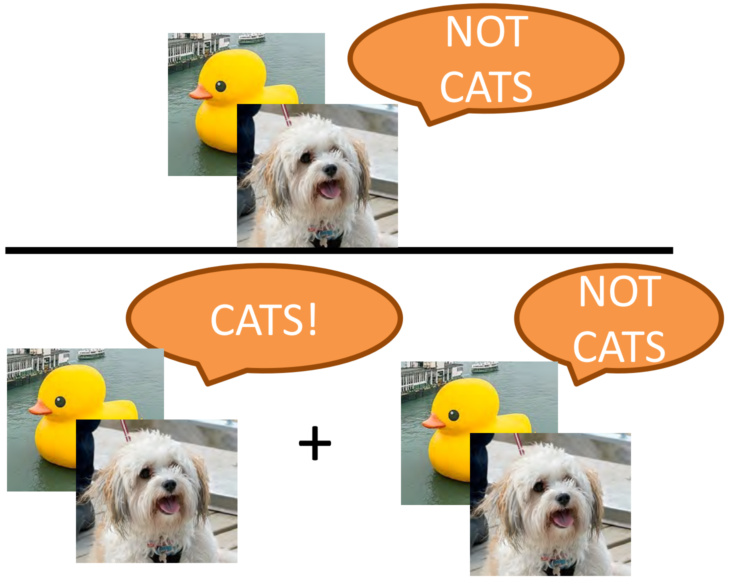


AKA sensitivity, hit rate

Focus on a single label: specificity

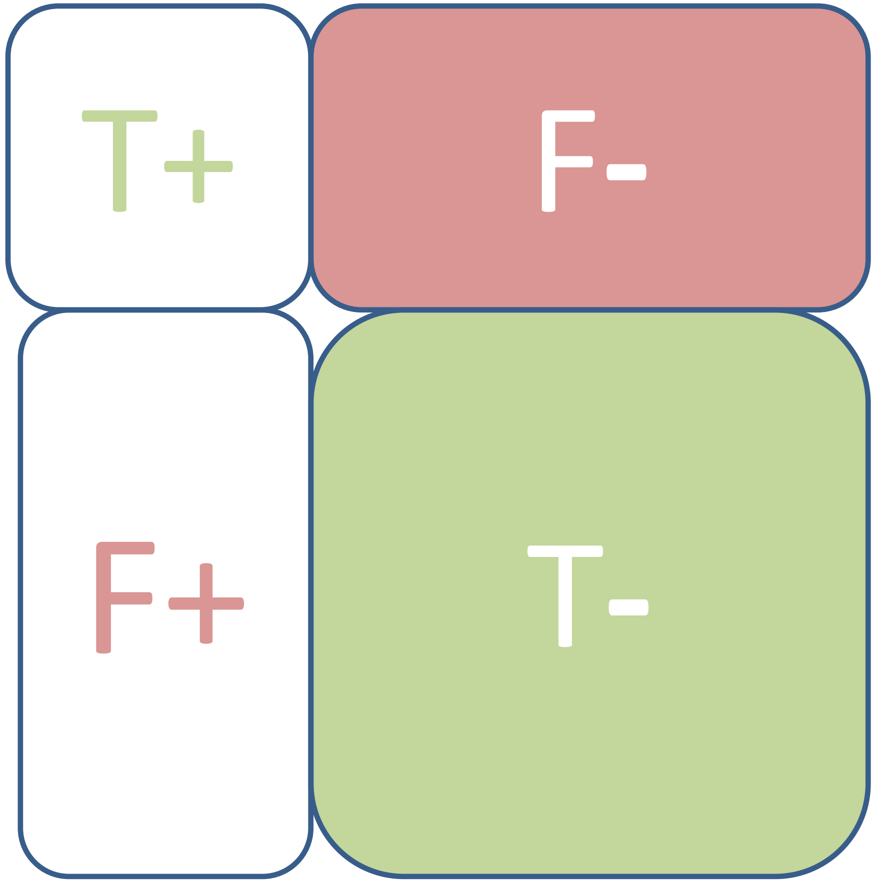


$$\text{Specificity} = \frac{\text{T-}}{\text{F+} + \text{T-}}$$



AKA selectivity

Focus on a single label: combinations



$$\text{Recall} = \frac{\boxed{T+}}{\boxed{T+} + \boxed{F-}}$$

$$\text{Precision} = \frac{\boxed{T+}}{\boxed{T+} + \boxed{F+}}$$

Focus on a single label: combinations

T+	F-
F+	T-

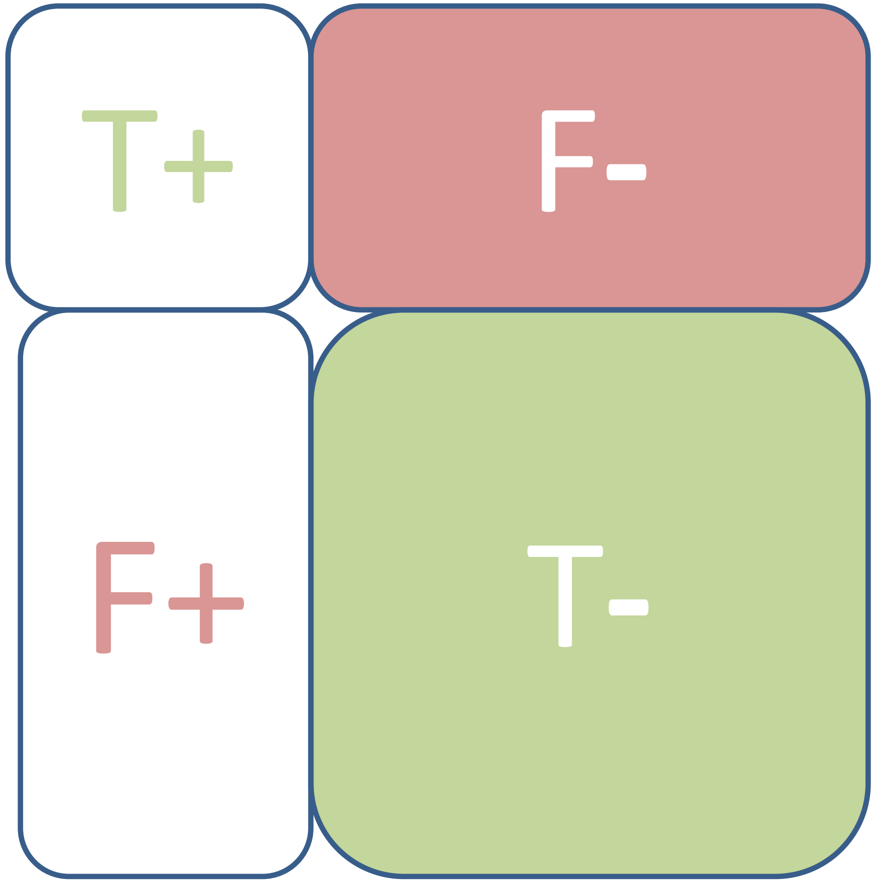
$$\text{Recall} = \frac{T+}{T+ \; + \; F-}$$

F- = (# of misses) =

$$\text{Precision} = \frac{T+}{T+ \; + \; F+}$$

F+ = (# of mistakes) =

Focus on a single label: combinations

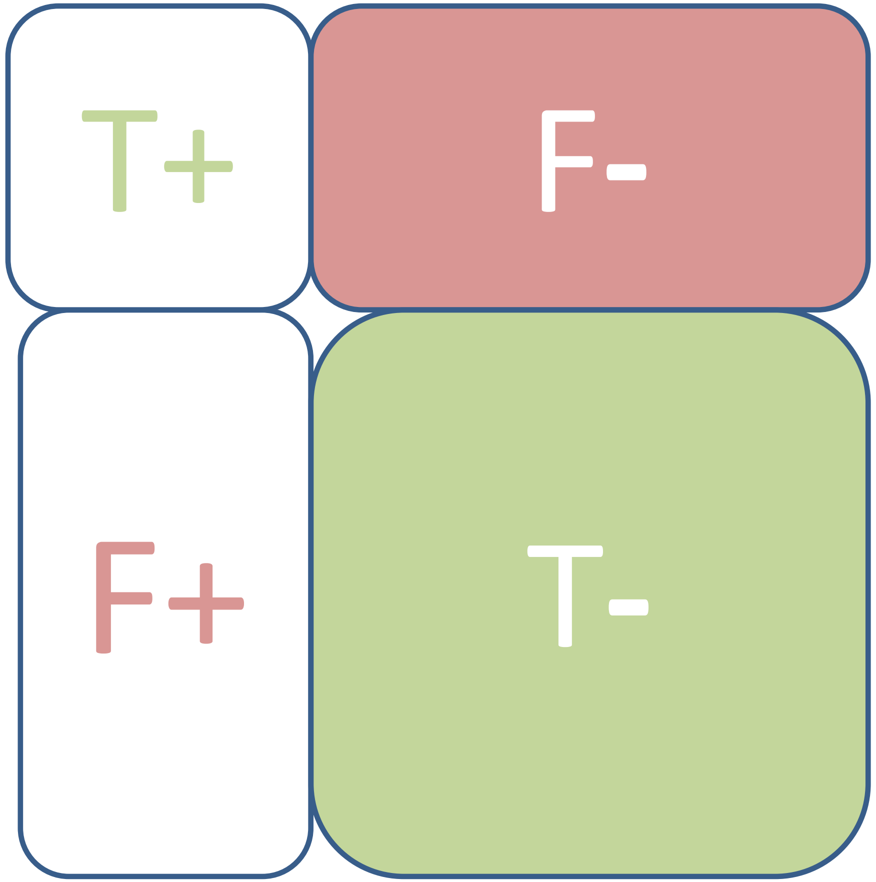


$$\text{Recall} = \frac{\boxed{T+}}{\boxed{T+} + \boxed{\text{\# of misses}}}$$

$$F_1 = \frac{\boxed{T+}}{\boxed{T+} + \dfrac{\boxed{\text{\# of mistakes}} + \boxed{\text{\# of misses}}}{2}}$$

$$\text{Precision} = \frac{\boxed{T+}}{\boxed{T+} + \boxed{\text{\# of mistakes}}}$$

Focus on a single label: combinations

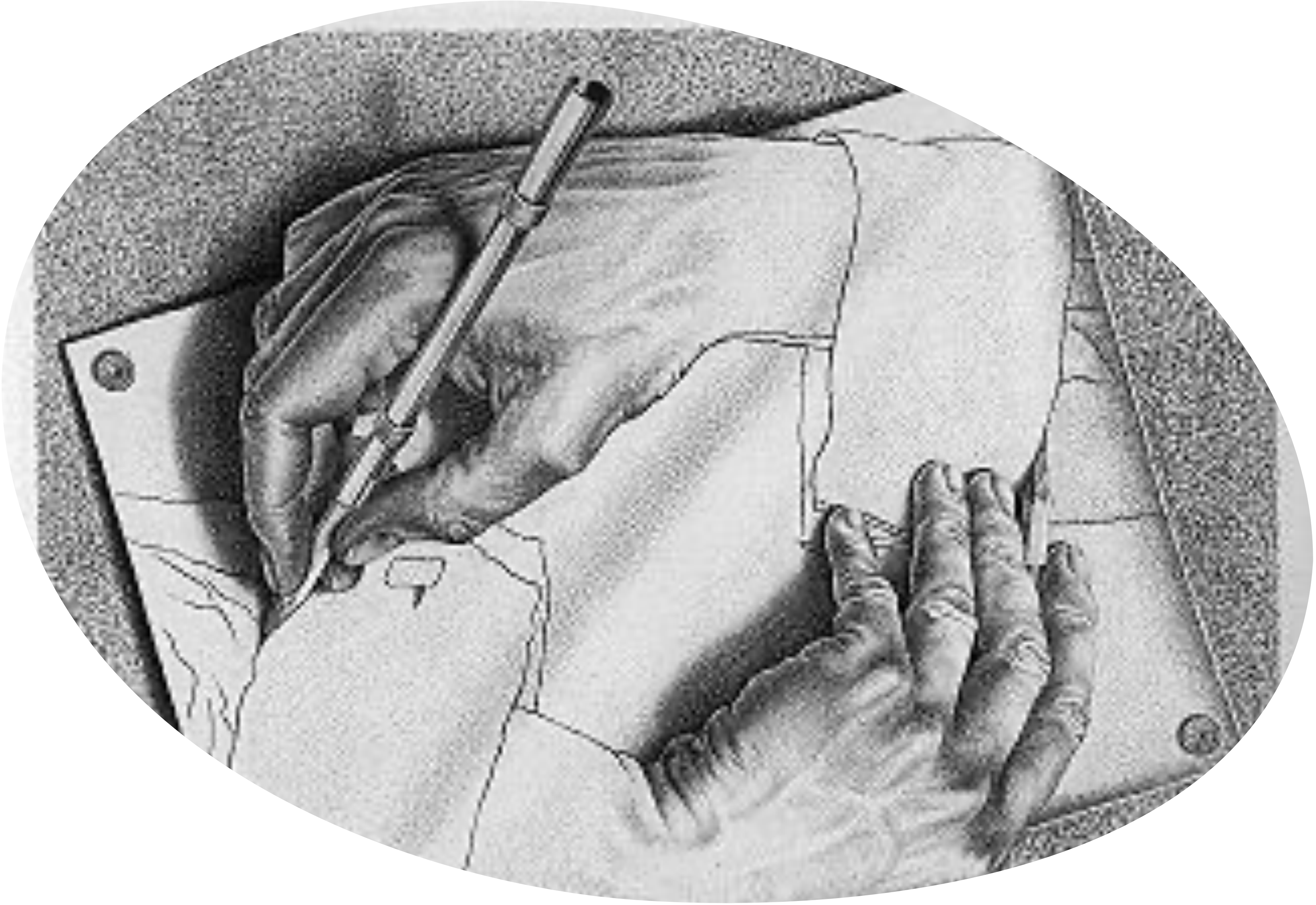


$$\text{Recall} = \frac{T+}{T+ \; + \; F-}$$

$$\text{Precision} = \frac{T+}{T+ \; + \; F+}$$

$$F_1 = \frac{1}{\frac{1}{2}\left(\frac{1}{\text{recall}} + \frac{1}{\text{precision}}\right)}$$

$$= 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$



Hands-on Example:

Classification using k-NN + Logistic Regression

Confusion matrix

Plot_confusion_matrix(*estimator*, *X*, *y_true*,
labels=None,
sample_weight=None,
normalize=None,
display_labels=None,
include_values=True,
xticks_rotation='horizontal',
values_format=None,
cmap='viridis',
ax=None)

https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix

Confusion matrix

Labels: List of labels to index the matrix. This may be used to reorder or select a subset of labels. If None is given, those that appear at least once in y_true or y_pred are used in sorted order.

Normalize: Normalizes confusion matrix over the true (rows), predicted (columns) conditions or all the population. If None, confusion matrix will not be normalized.

include_values: Includes values in confusion matrix.

Classification Report

classification_report(*y_true*, *y_pred*,
labels=None,
target_names=None,
sample_weight=None,
digits=2,
output_dict=False,
zero_division='warn')

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

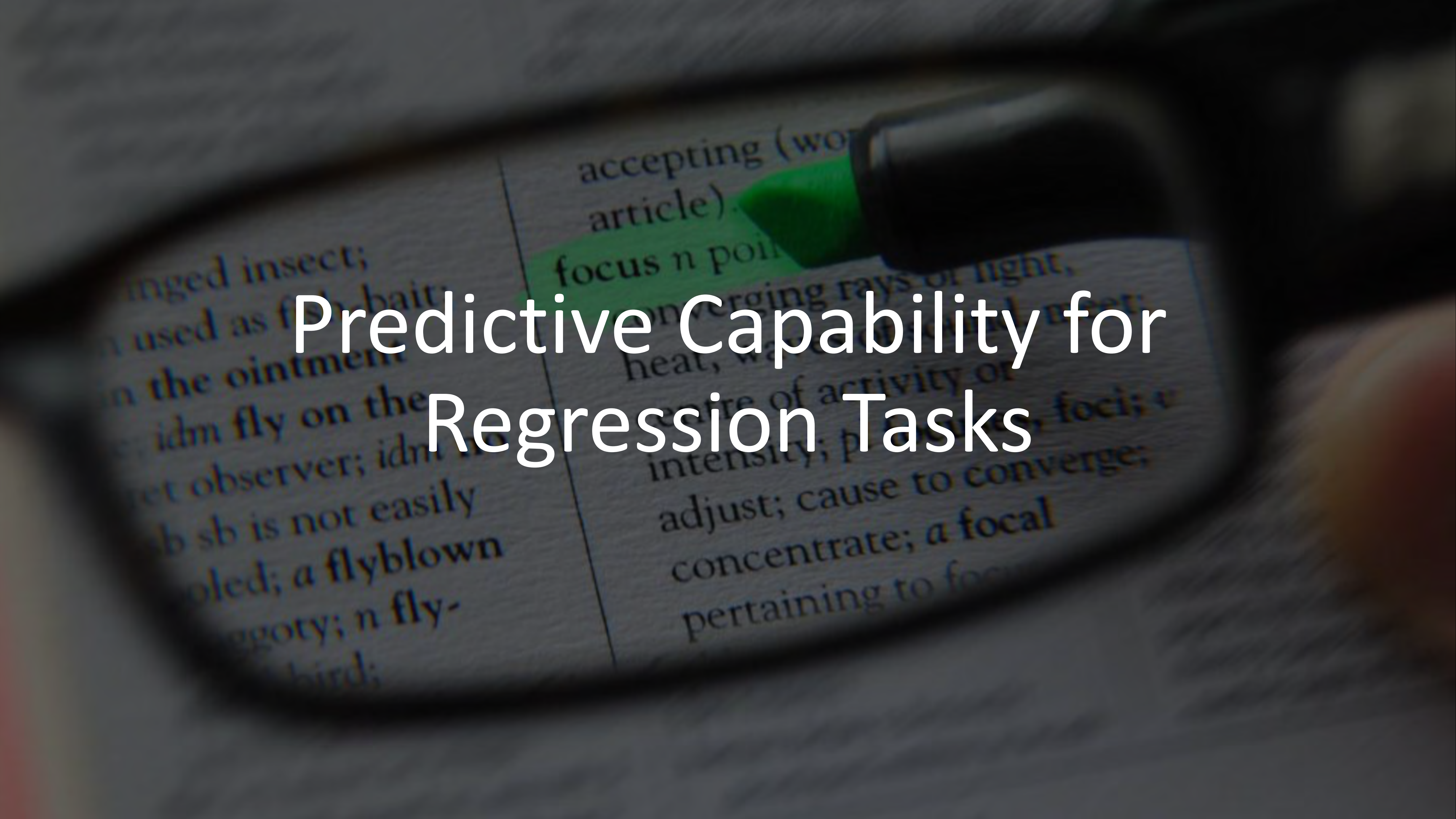
Classification Report

'macro’: Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

'weighted’: Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters ‘macro’ to account for label imbalance; it can result in an F-score that is not between precision and recall.

Note that if all labels are included, “micro”-averaging in a multiclass setting will produce precision and recall scores that are all identical to accuracy.

Predictive Capability for Regression Tasks



Mean Squared Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

where N is the number of data points, f_i the value returned by the model and y_i the actual value for data point i.

Mean Squared Error (MSE)

$$MSE = \frac{1}{N}\boxed{\sum_{i=1}^{N}(f_i - y_i)^2}$$

where N is the number of data points, f_i the value returned by the model and y_i the actual value for data point i.

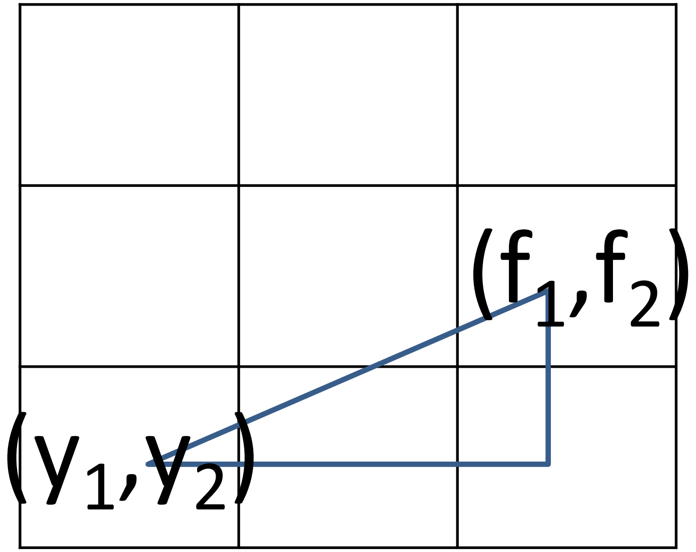
Euclidean distance squared, divided by number of points

Mean Squared Error (MSE)

$$MSE = \frac{1}{N}\boxed{\sum_{i=1}^{N}(f_i - y_i)^2}$$

where N is the number of data points, f_i the value returned by the model and y_i the actual value for data point i.

N=2



Mean Squared Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

where N is the number of data points, f_i the value returned by the model and y_i the actual value for data point i.

N=2

2	**2.236**	**2.828**
1	**1.414**	**2.236**
(y_1, y_2)	**1**	**2**

Mean Absolute Deviation (MAD)

$$\frac{1}{N}\sum_{i=1}^{N}\left|(f_i - y_i)\right|$$

Mean Absolute Deviation (MAD)

Manhattan distance
divided by number of points

$$\frac{1}{N}\sum_{i=1}^{N}|f_i - y_i|$$

N=2

2	3	4
1	2	3
(y_1,y_2)	1	2

Maximum error

N=2

2	2	2
1	1	2
(y_1,y_2)	1	2

Recall the L^p norms

$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}$$

$$\|x\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$$

Unit circle for different values of p

Regularization: mix and match

Mix and match

Multivariate Regression: F = X β + constant

$$F = A + B_1 X_1 + B_2 X_2 + \ \dots \ + \ B_K X_K$$

Mix and match

Multivariate Regression: F = X β + constant

$$\sum_{i=1}^{N}(f_i - y_i)^2$$

$$= (y - X\beta)^T(y - X\beta)$$

Mix and match

Multivariate Regression: F = X β + constant

$$\text{Ridge Cost} = (y - X\beta)^T(y - X\beta) + \alpha||\beta||_2^2$$
$$\text{Lasso Cost} = (y - X\beta)^T(y - X\beta) + \alpha||\beta||_1$$

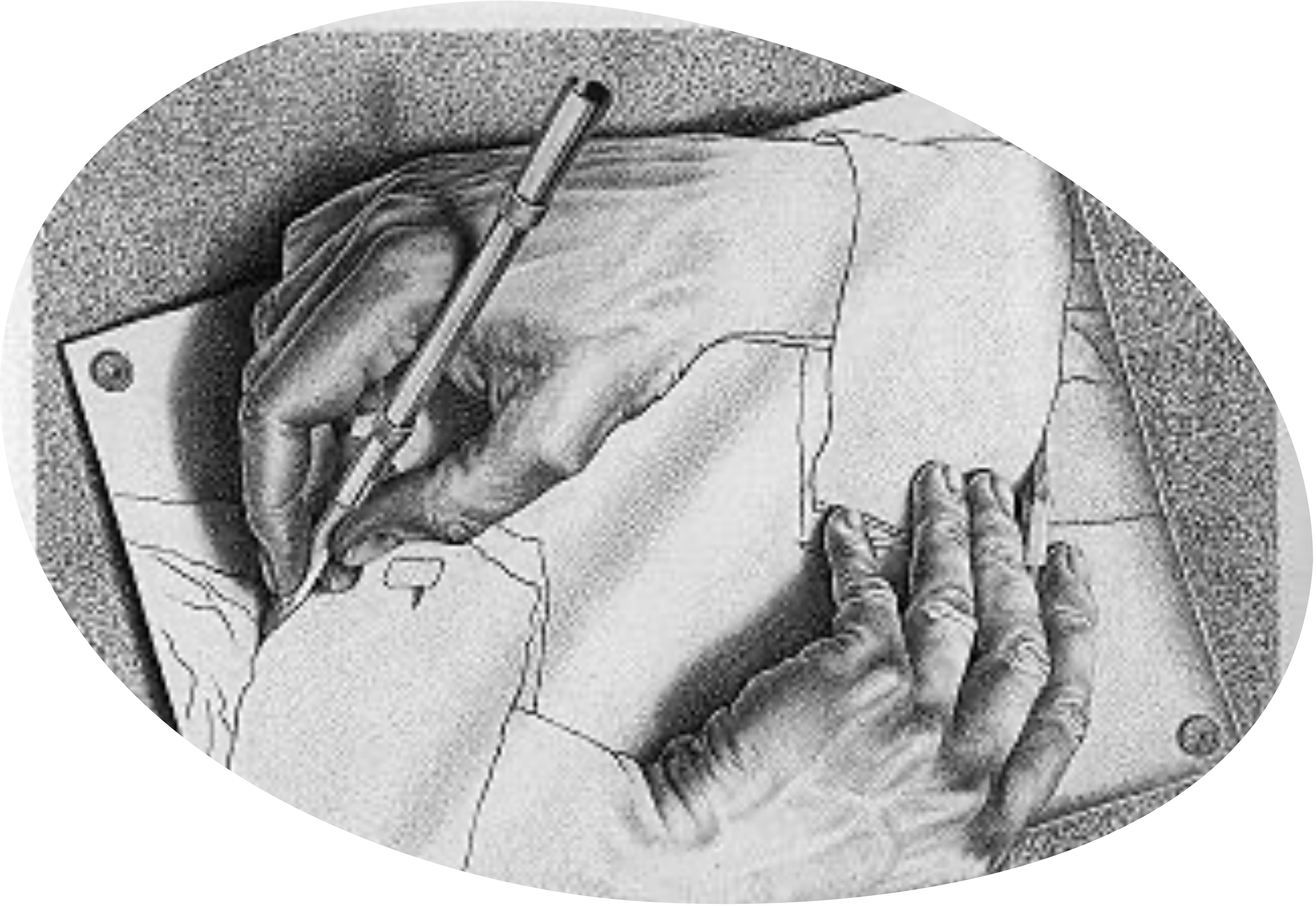
α is the regularization (hyper)parameter

Mix and match

Multivariate Regression: F = X β + constant

$$\text{Ridge Cost} = \underbrace{(y - X\beta)^T(y - X\beta)}_{L^2} + \alpha \underbrace{||\beta||_2^2}_{L^2}$$

$$\text{Lasso Cost} = \underbrace{(y - X\beta)^T(y - X\beta)}_{L^2} + \alpha \underbrace{||\beta||_1}_{L^1}$$



Hands-on Example:

Linear Regression