Hewlett Packard Enterprise
Data Science Institute
UNIVERSITY OF HOUSTON

HPE DSI 311
Introduction to Machine Learning

Spring 2023

Instructor: Ioannis Konstantinidis

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:

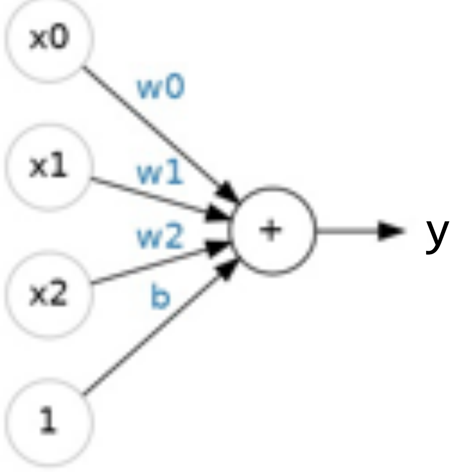
$$\mathbf{y} = \mathbf{X}\,\beta + \alpha$$ (matrix notation)

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

Math:

y, X -> variables

α, β -> parameters

Statistics:

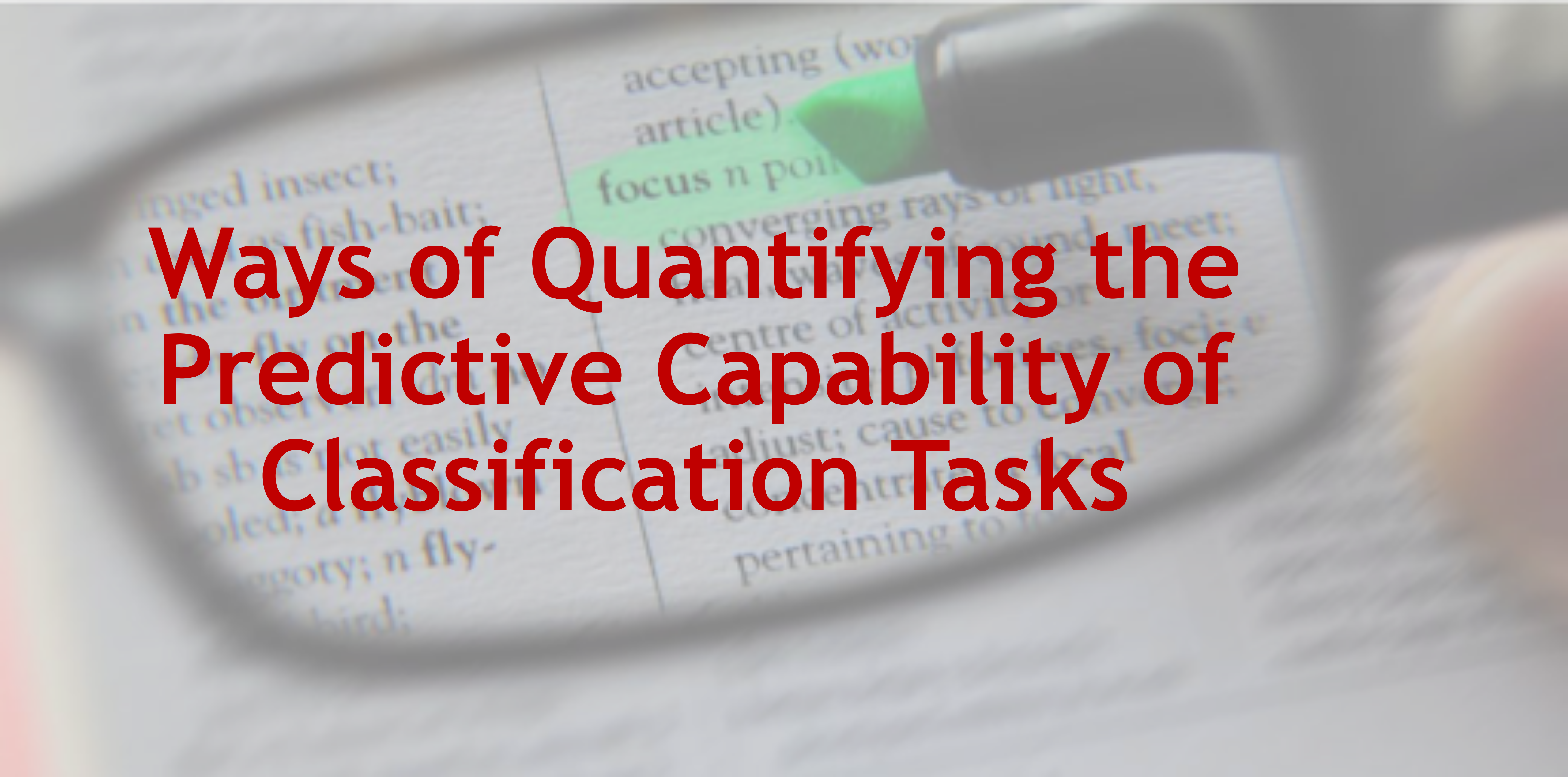
y, X_i -> variables

a, b_i -> parameters

Computer Science:

X, y -> parameters when fitting

b, a -> parameters when predicting
called weights w for networks



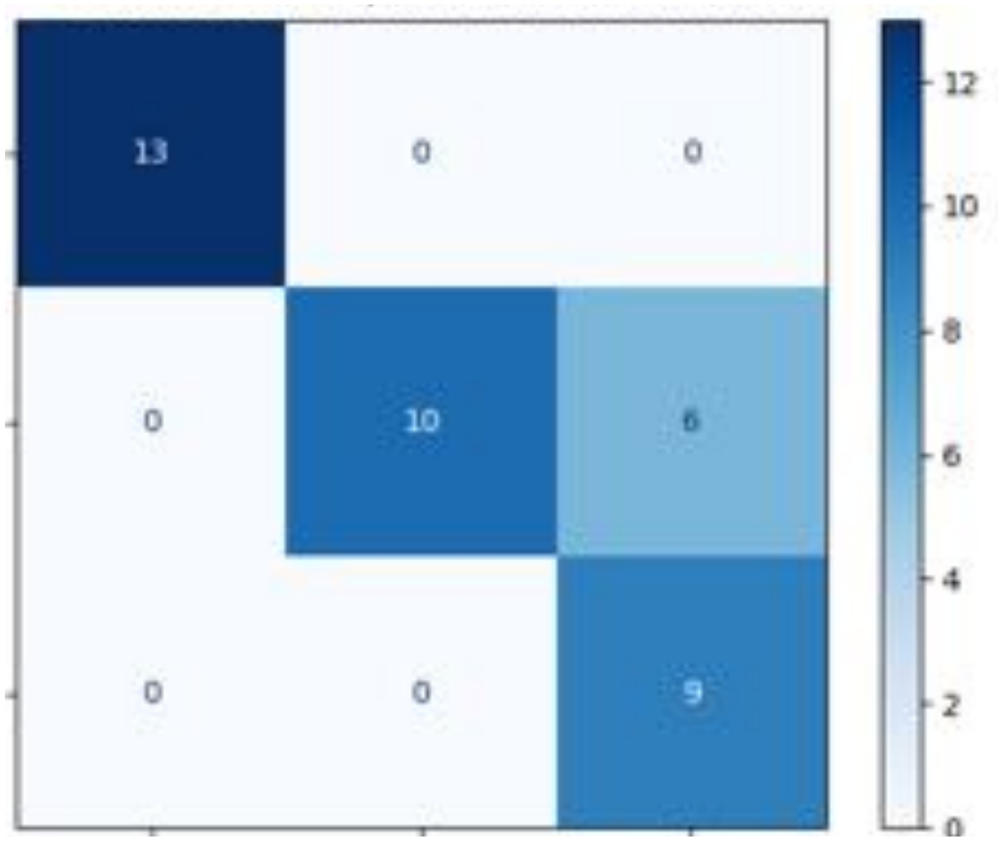
Ways of Quantifying the Predictive Capability of Classification Tasks

Confusion Matrix

Confusion Matrix

Columns: predictions made by the classifier (labels y)

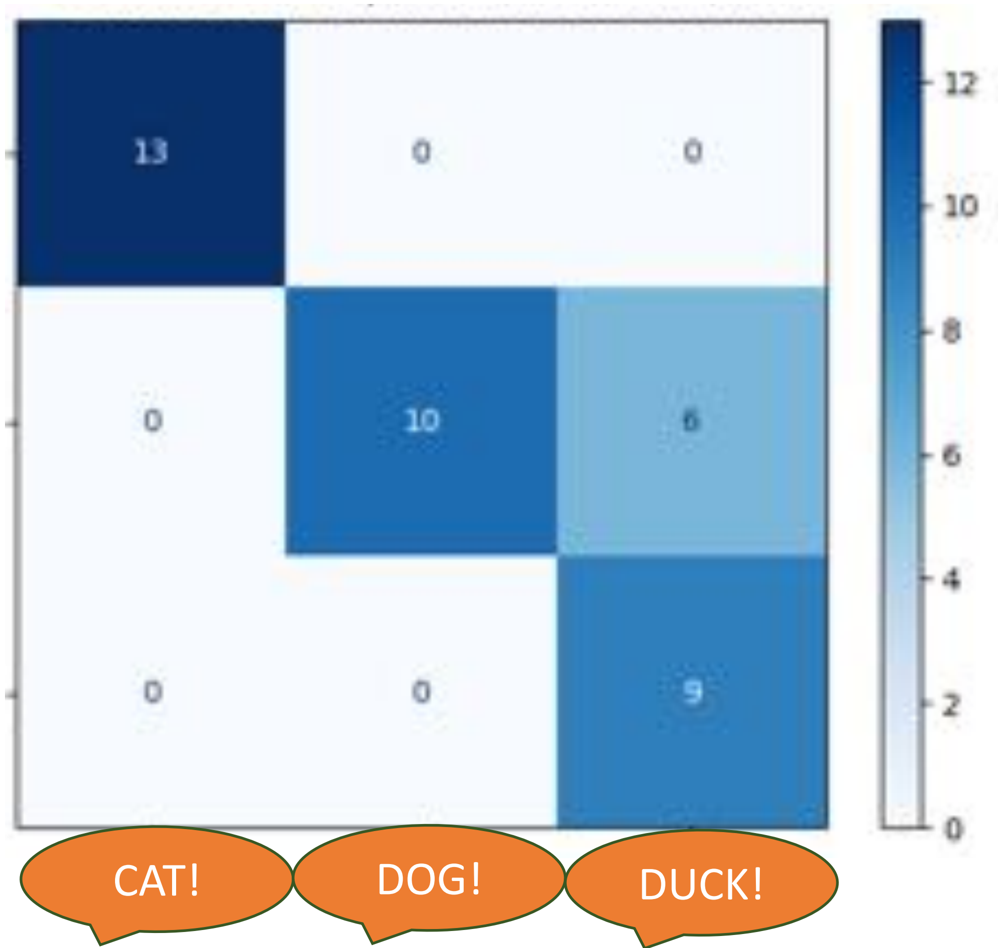
Rows: actual observations (points X)



Confusion Matrix

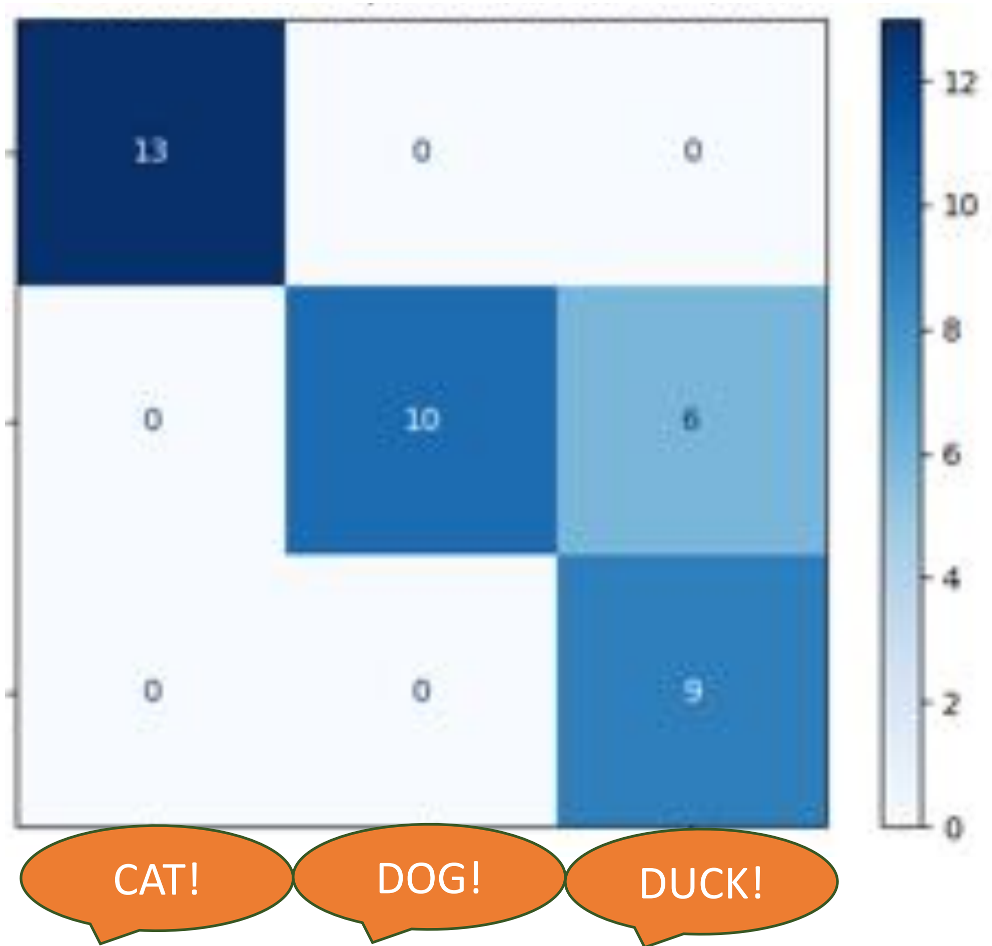
Columns: predictions made by the classifier (labels y)

Rows: actual observations (points X)



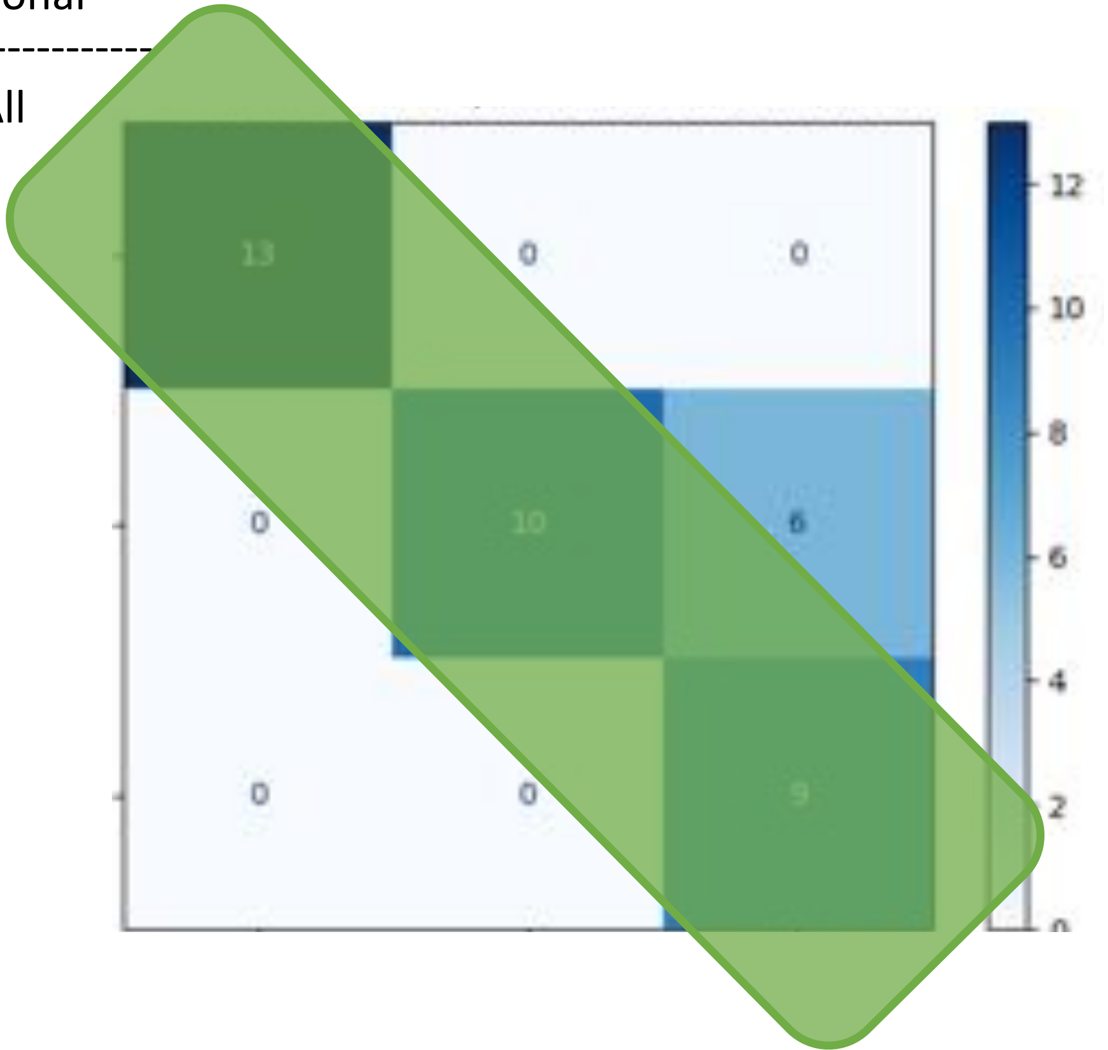
Confusion Matrix

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



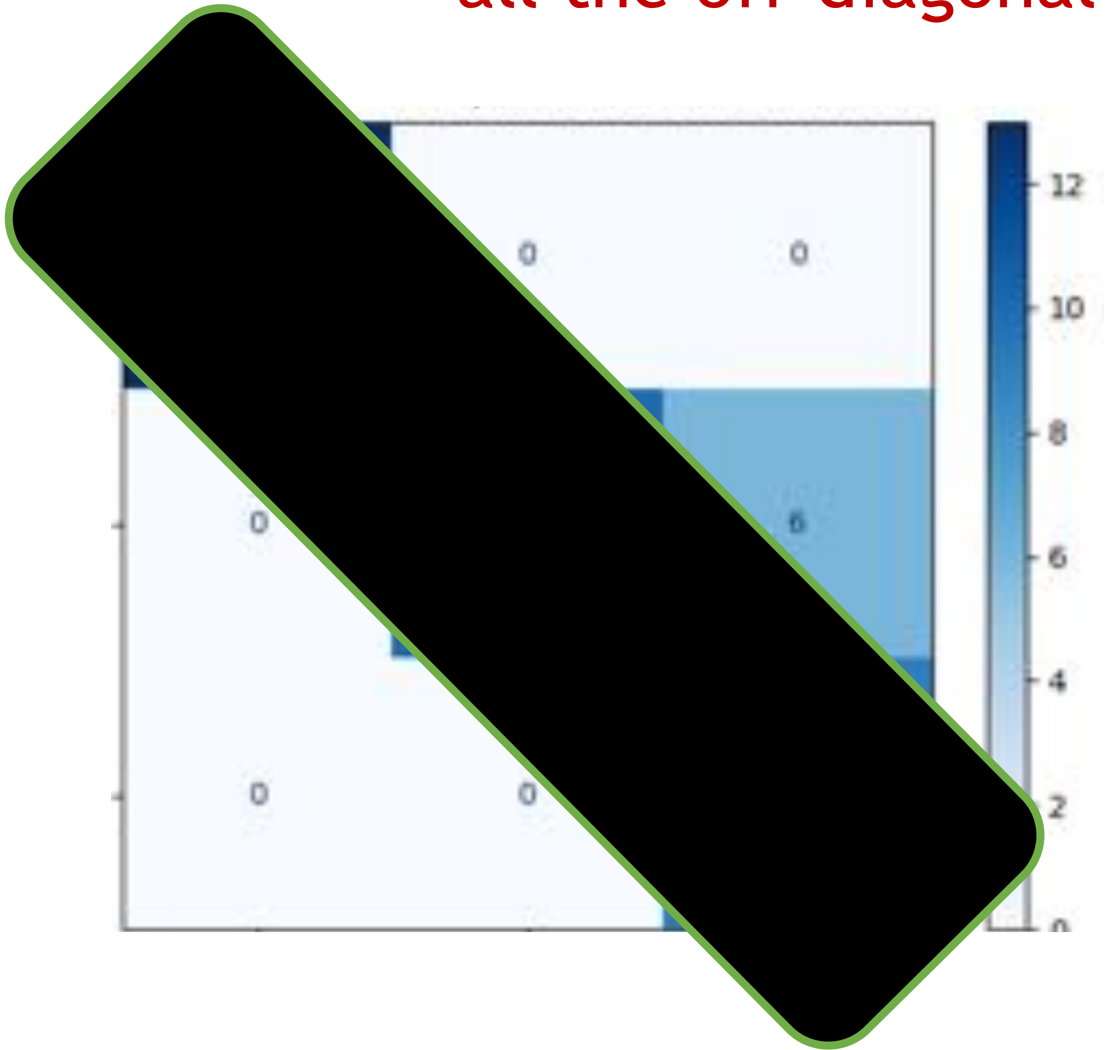
Confusion Matrix

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

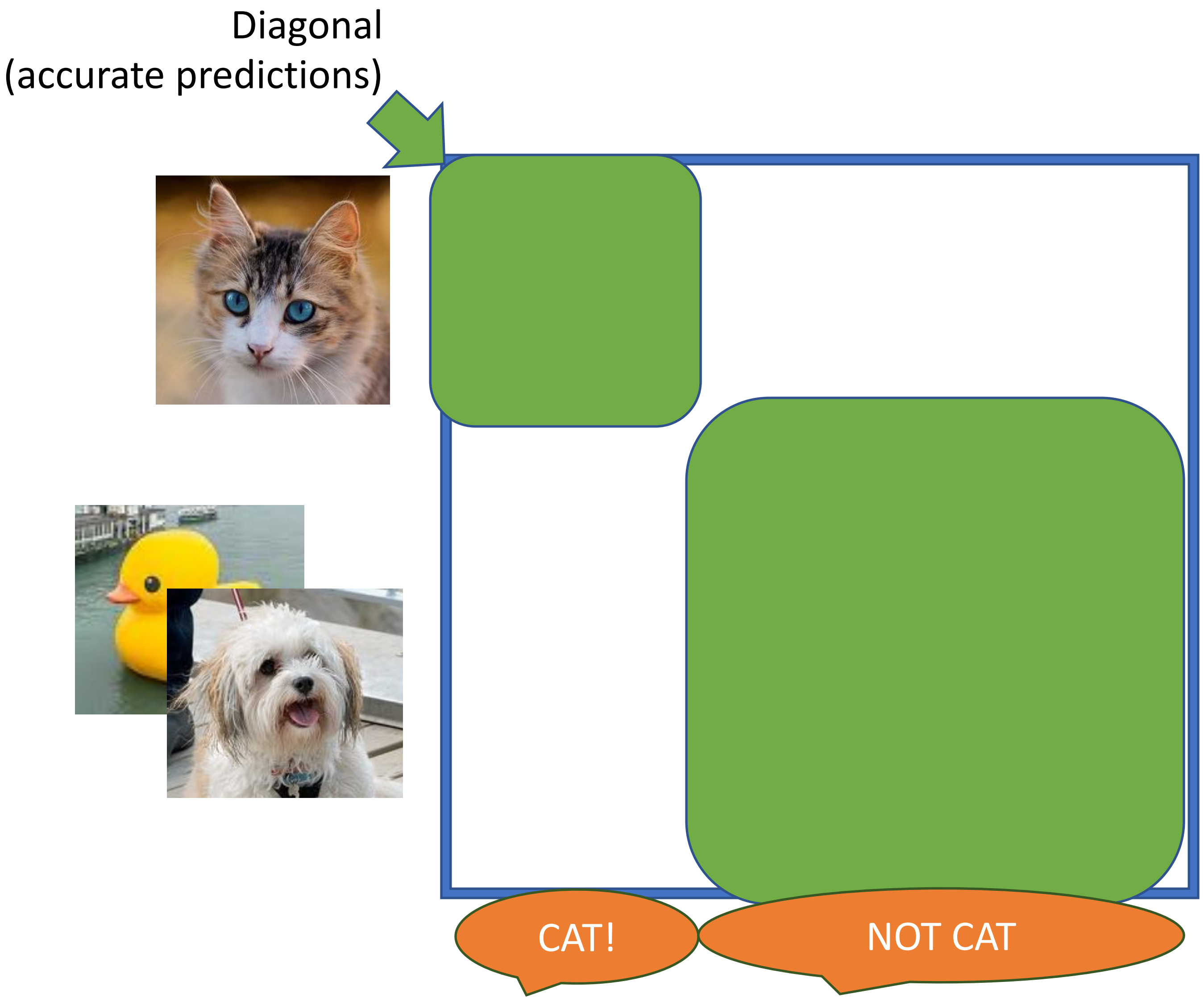


Confusion Matrix

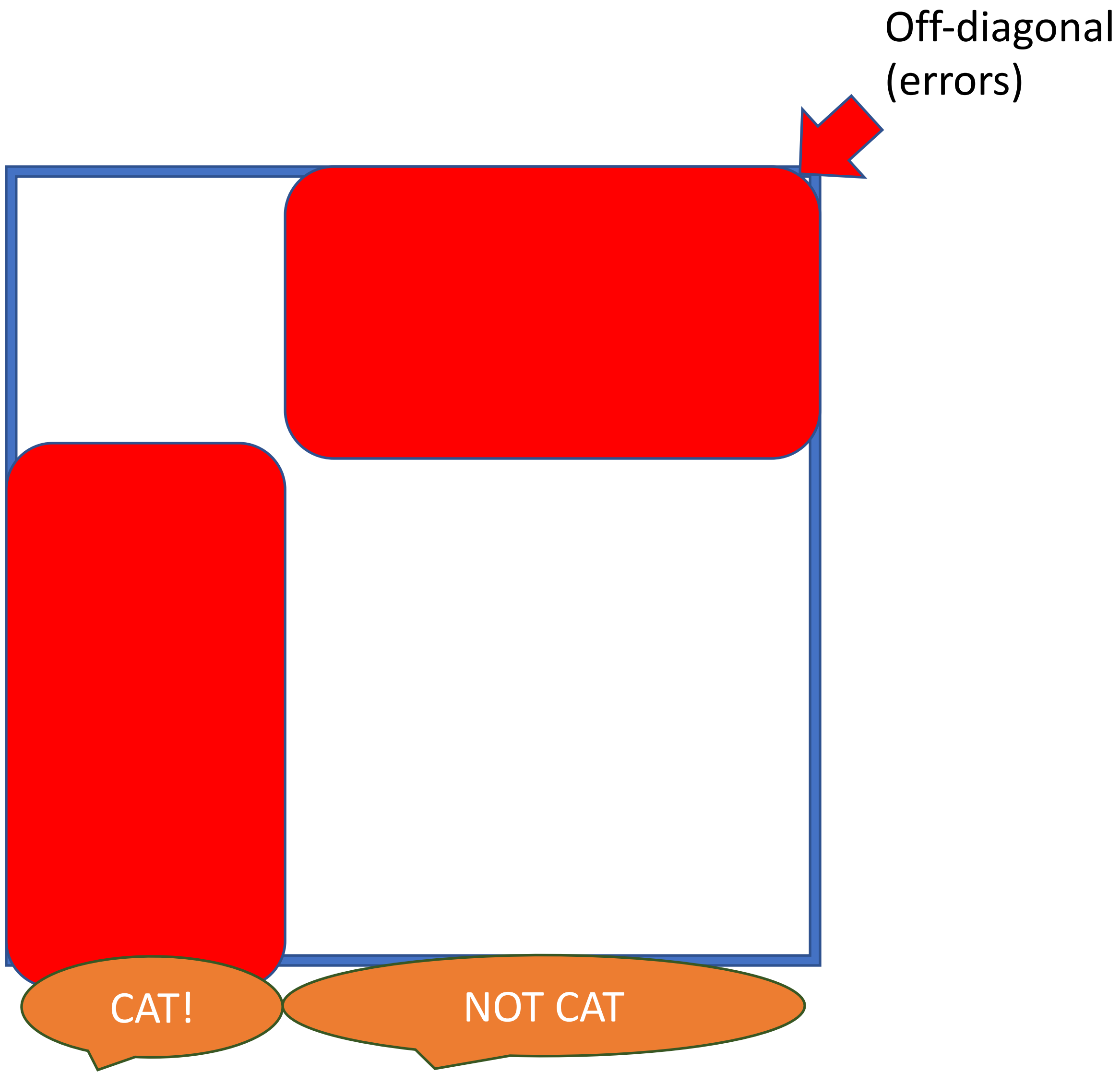
One type of error:
all the off-diagonal entries



Focus on a single label

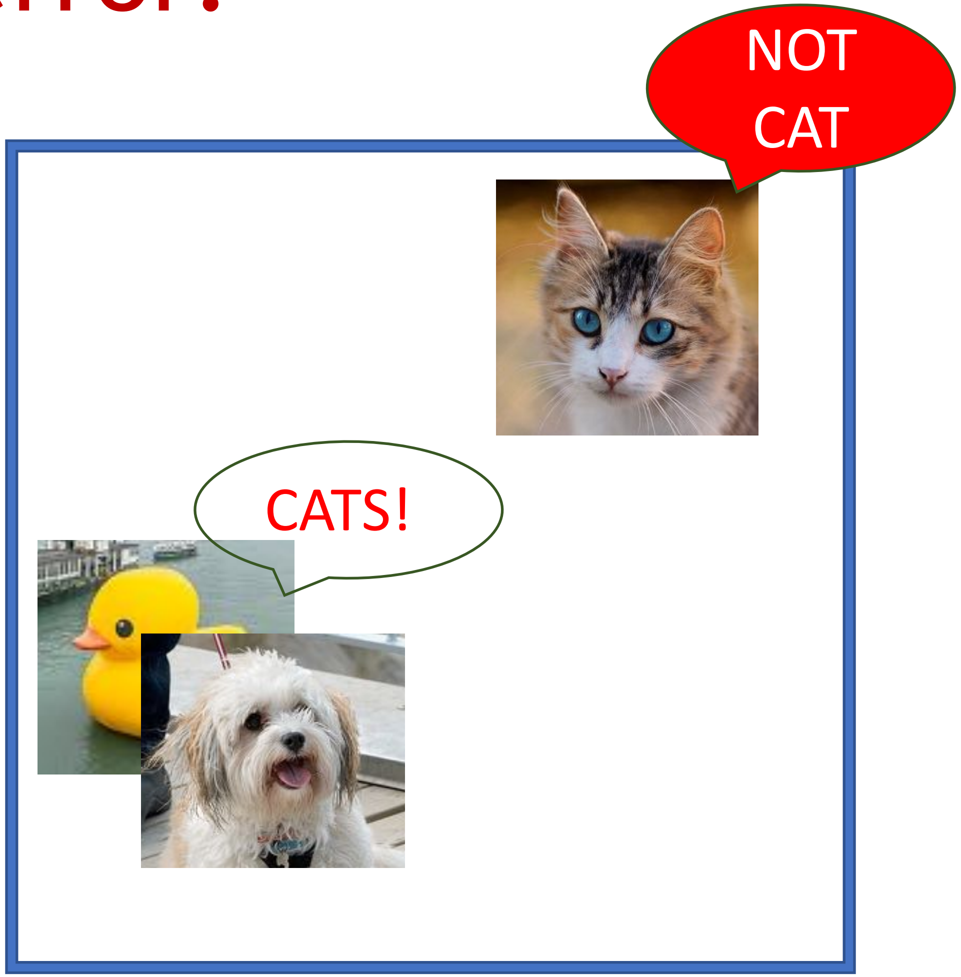


Focus on a single label

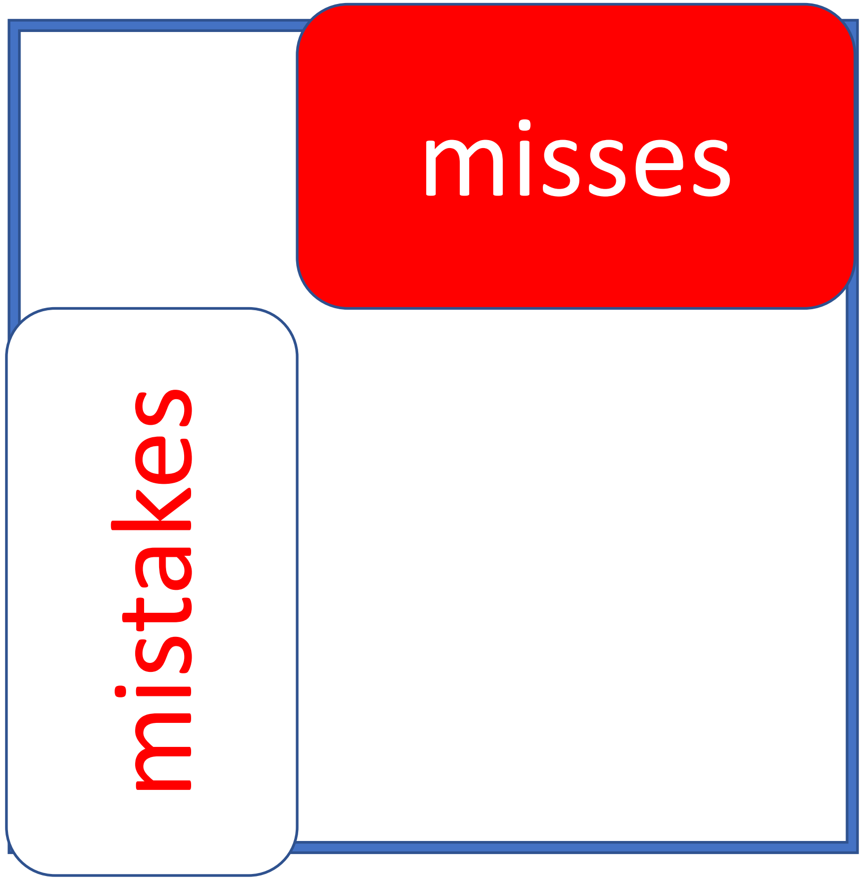


Focus on a single label

TWO types of error:

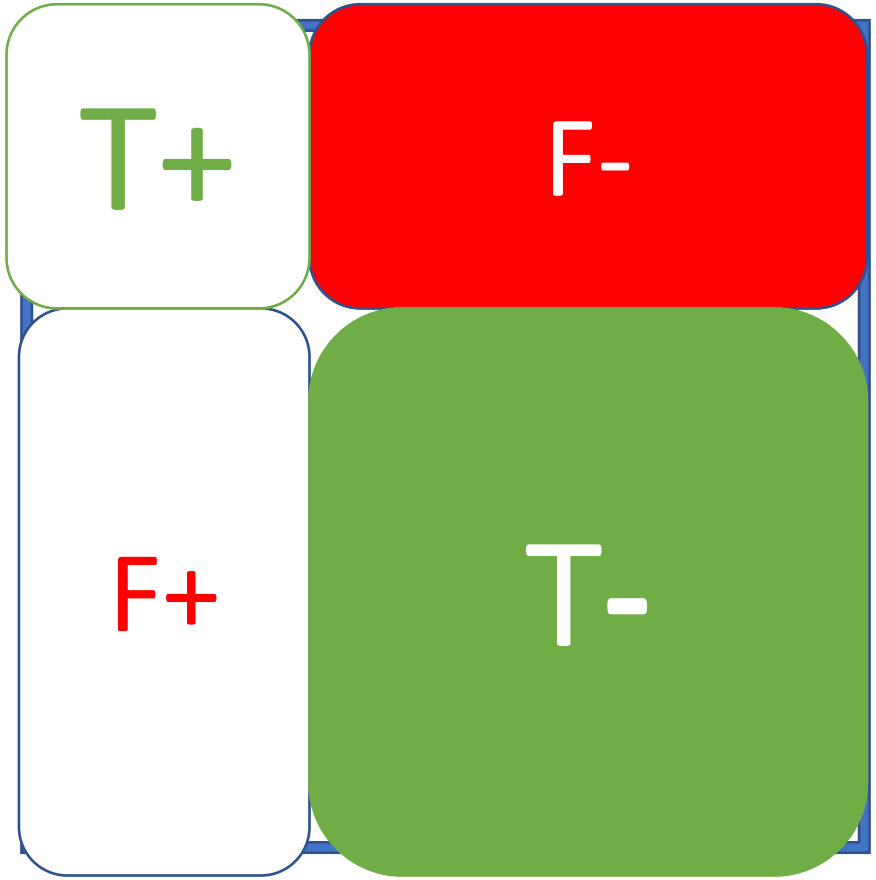


TWO types of error:



Focus on a single label

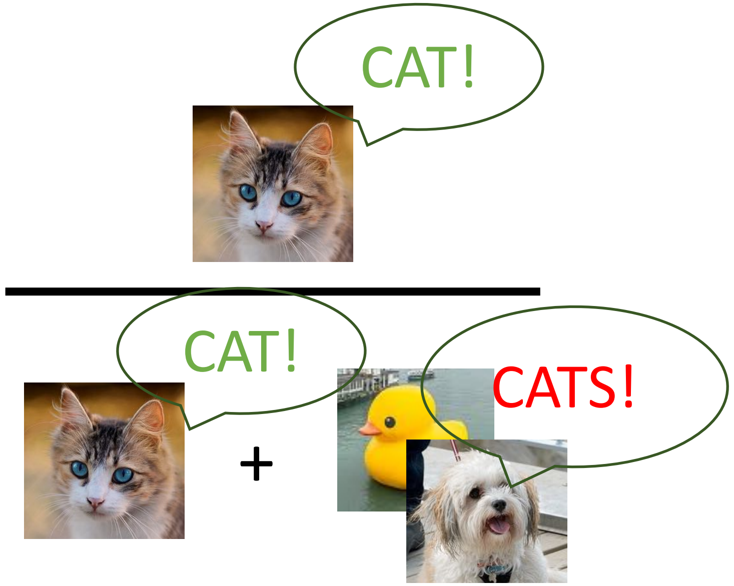
TWO types of error and two correct types



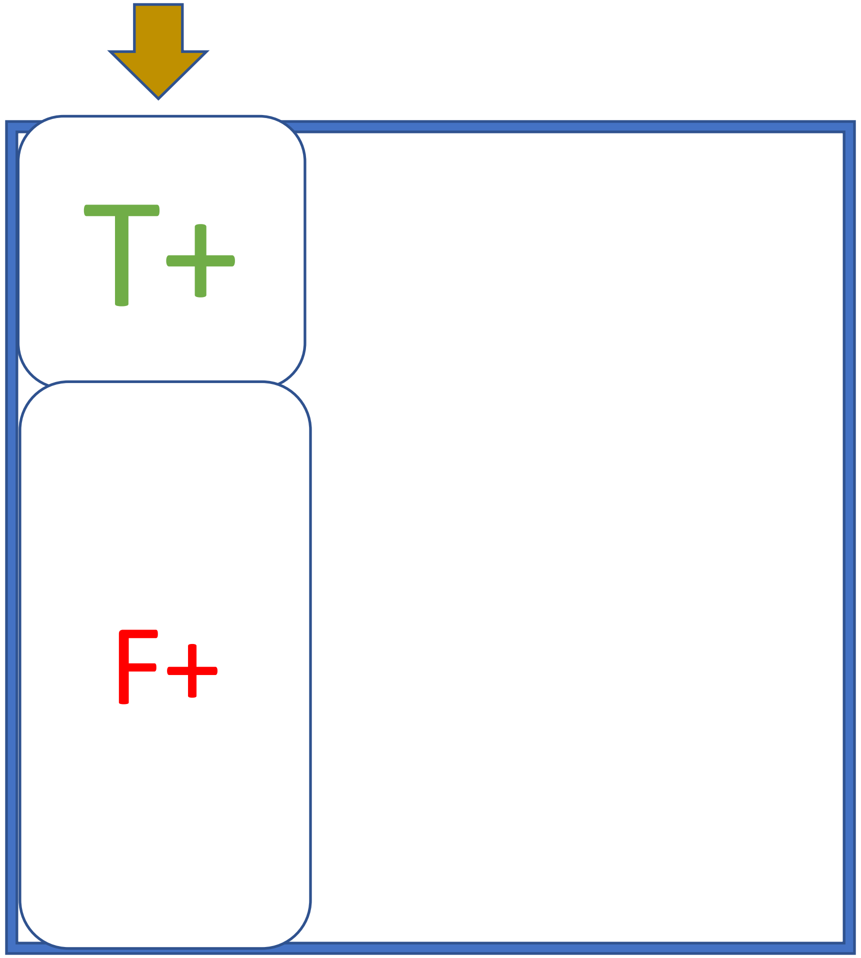
Focus on a single label

Hewlett Packard Enterprise Data Science Institute
UNIVERSITY OF HOUSTON

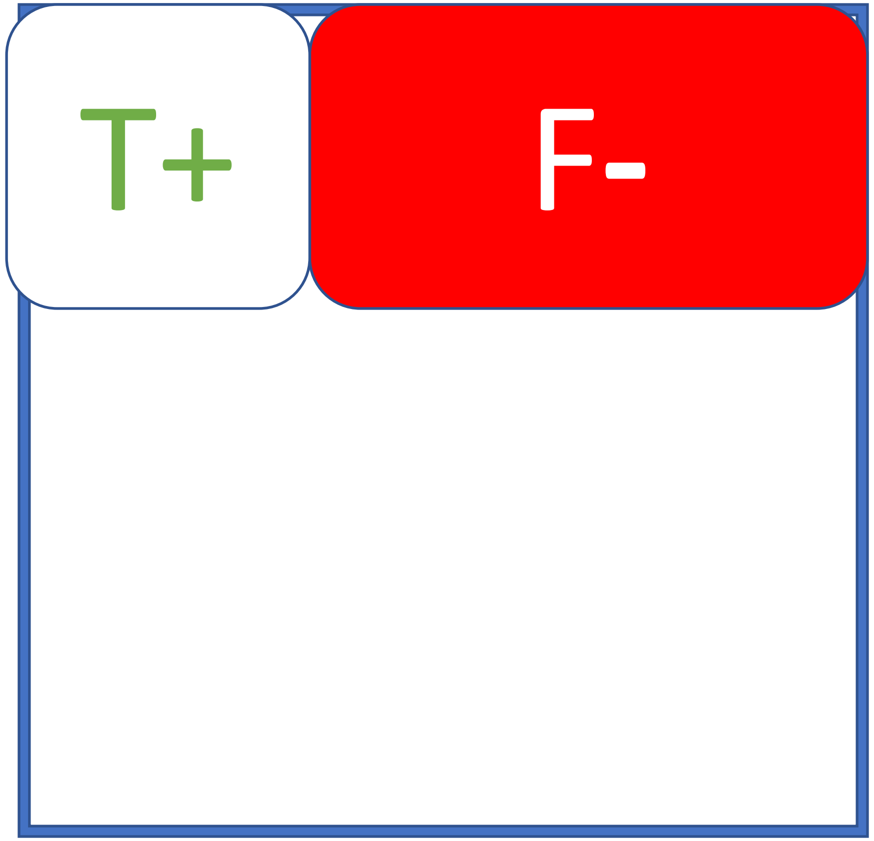
$$\textbf{Precision} = \frac{\boxed{T+}}{\boxed{T+} + \boxed{F+}}$$



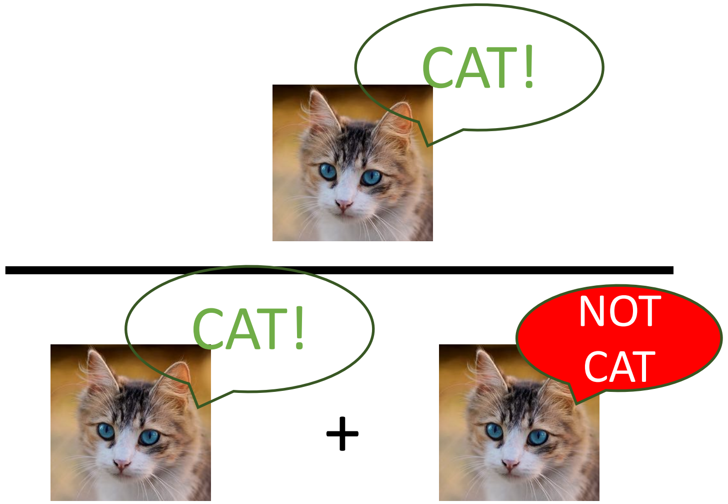
Positives column
(predicted to be the target)



Focus on a single label

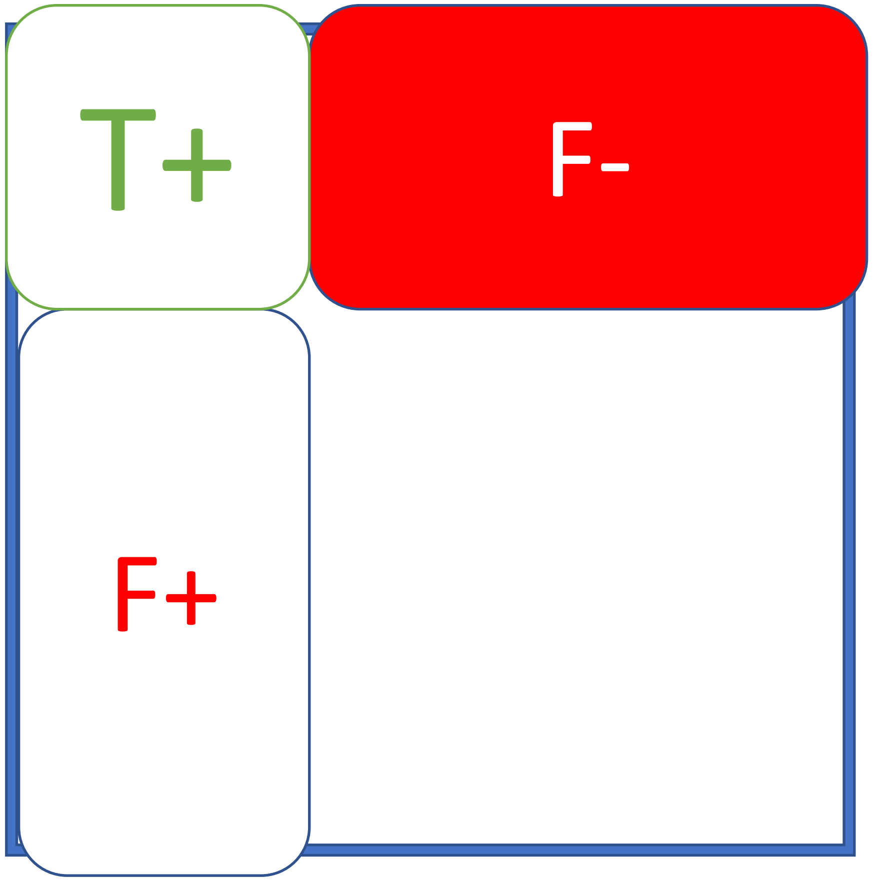


$$\textbf{Recall} = \frac{T+}{T+ \; + \; F-}$$



AKA sensitivity, hit rate

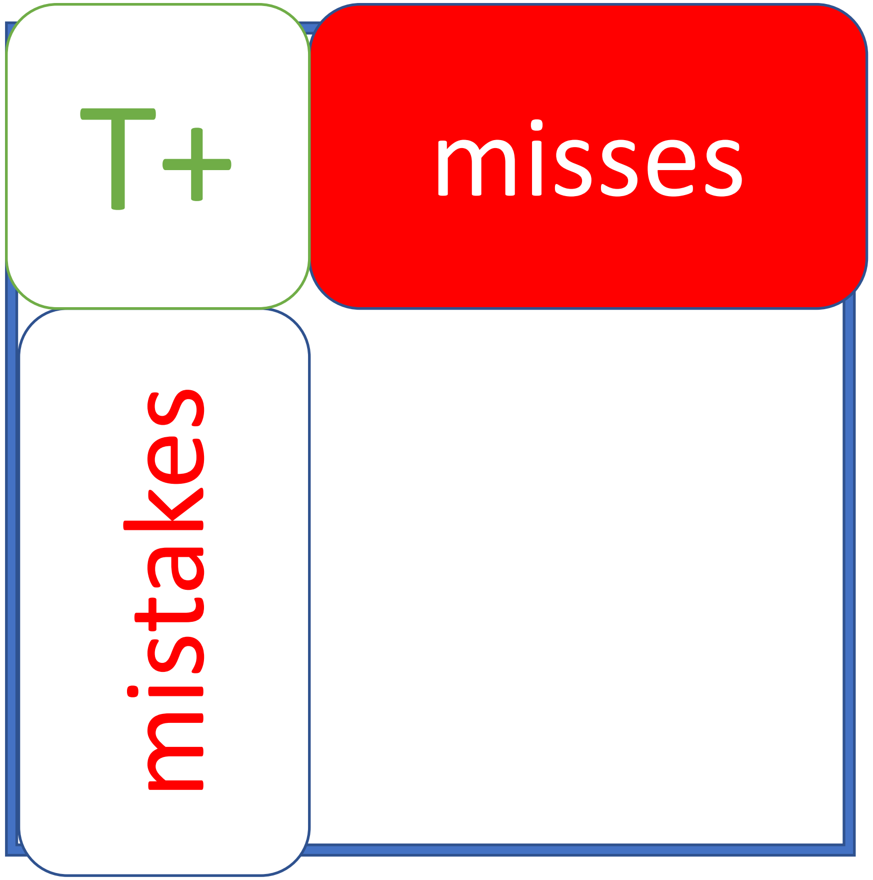
Focus on a single label



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

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

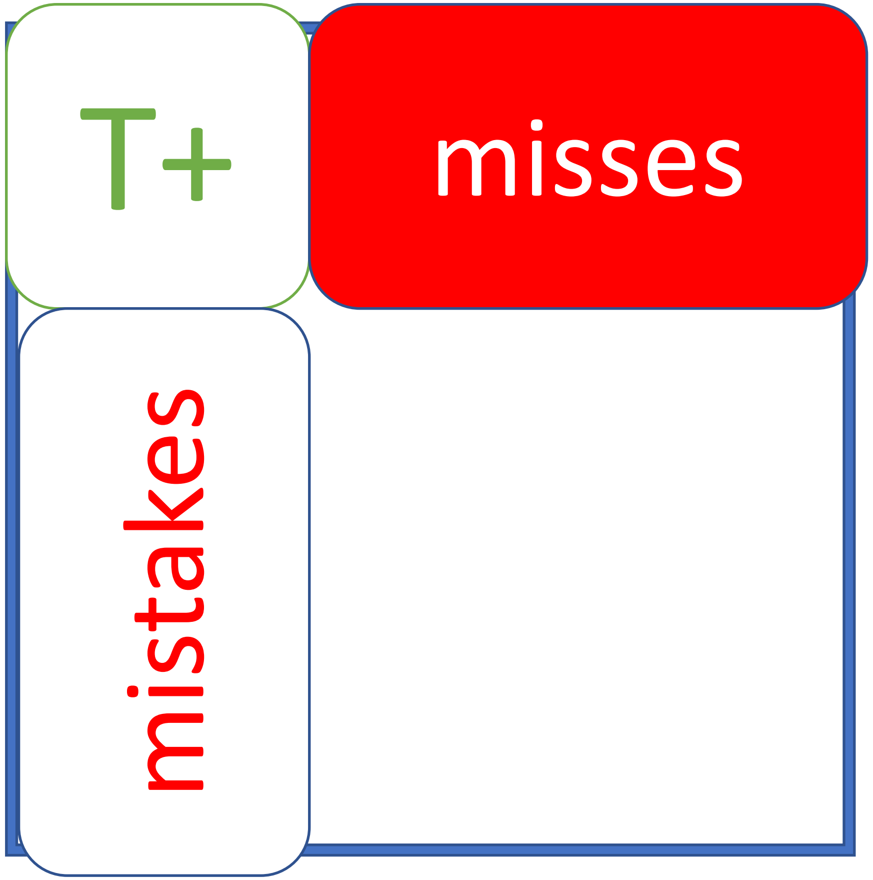
Focus on a single label



$$\textbf{Recall} = \frac{T+}{T+ \; + \; \text{misses}}$$

$$\textbf{Precision} = \frac{T+}{T+ \; + \; \text{mistakes}}$$

Focus on a single label



$$\textbf{Recall} = \frac{\text{T+}}{\text{T+} + \text{misses}}$$

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

$$\textbf{Precision} = \frac{\text{T+}}{\text{T+} + \text{mistakes}}$$

Focus on a single label

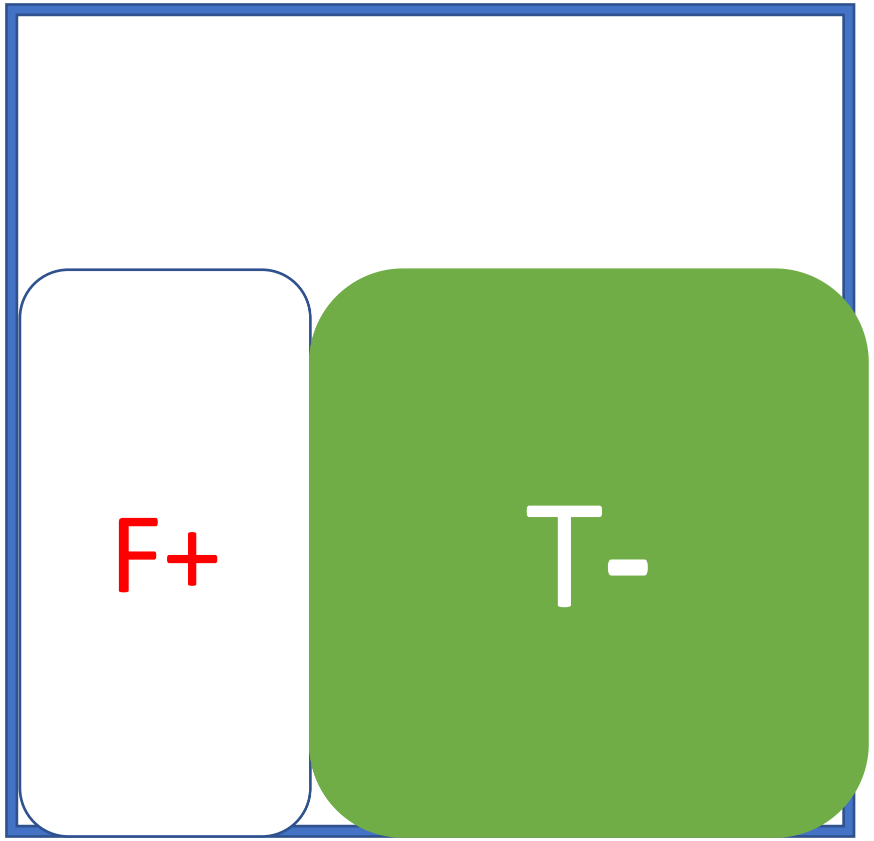
Hewlett Packard Enterprise Data Science Institute
UNIVERSITY OF HOUSTON

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

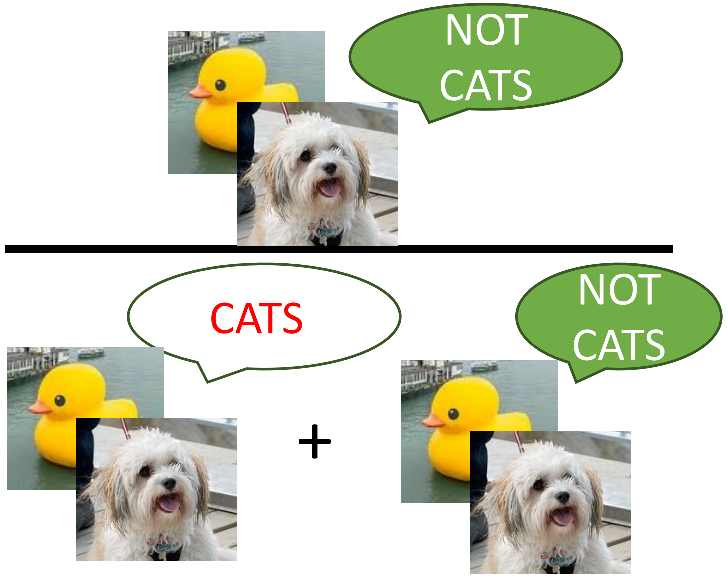
$$= \frac{1}{\frac{1}{2}\left(\frac{1}{\text{recall}} + \frac{1}{\text{precision}}\right)}$$

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

Focus on a single label



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



AKA selectivity

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.

How do you “fit” a model?

Assessing model fitness - supervised ML

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

Assessing model fitness - supervised ML

Calculate the **parameter values** that make model predictions which 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 model predictions which 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 points** that **are closest to** the new data point

Need an appropriate Penalty() for X

Testing model performance

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 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.

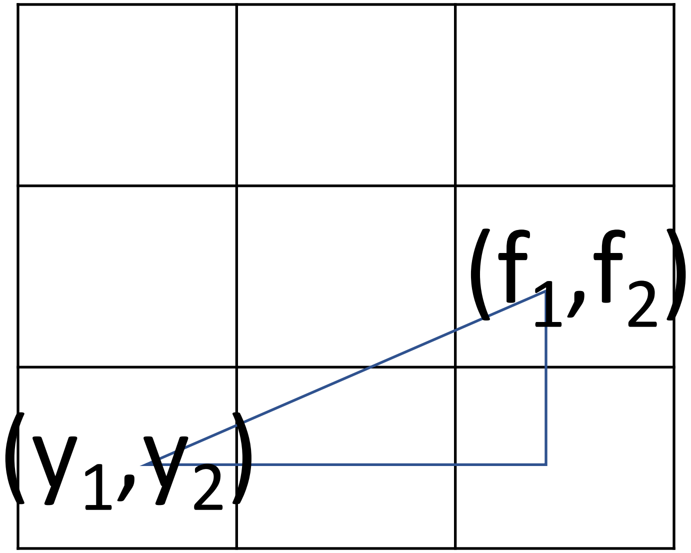
Euclidean distance squared, divided by number of points

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

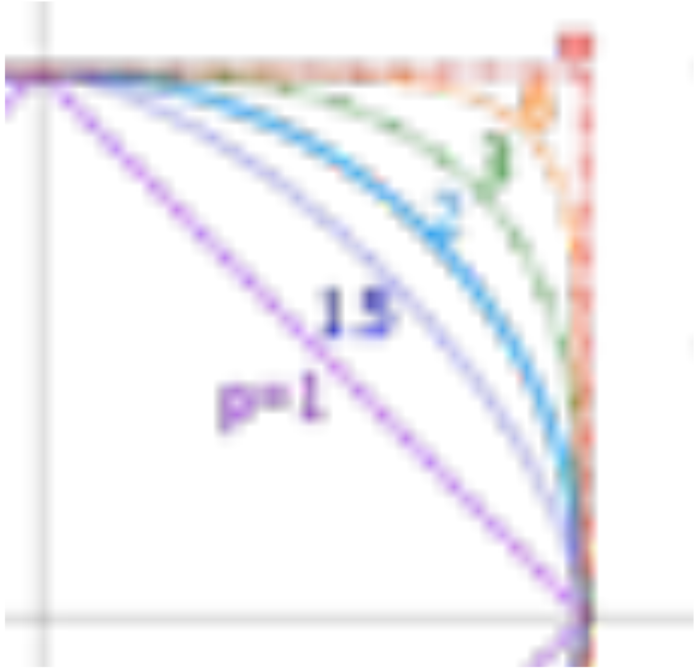
N=2

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

The L^p norms

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

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



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

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

Mix and match

Multivariate Regression: F = X β + constant

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

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

Mix and match

Multivariate Regression: F = X β + constant

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

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

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

Hands-on Example:

Linear Regression