# MODEL EVALUATION

# How Do We Evaluate?

- what is overall "accuracy"
- are we better predictiong (green or red) labels?
- how much better are we compared with random ("coin" flipping)

# Many Metrics

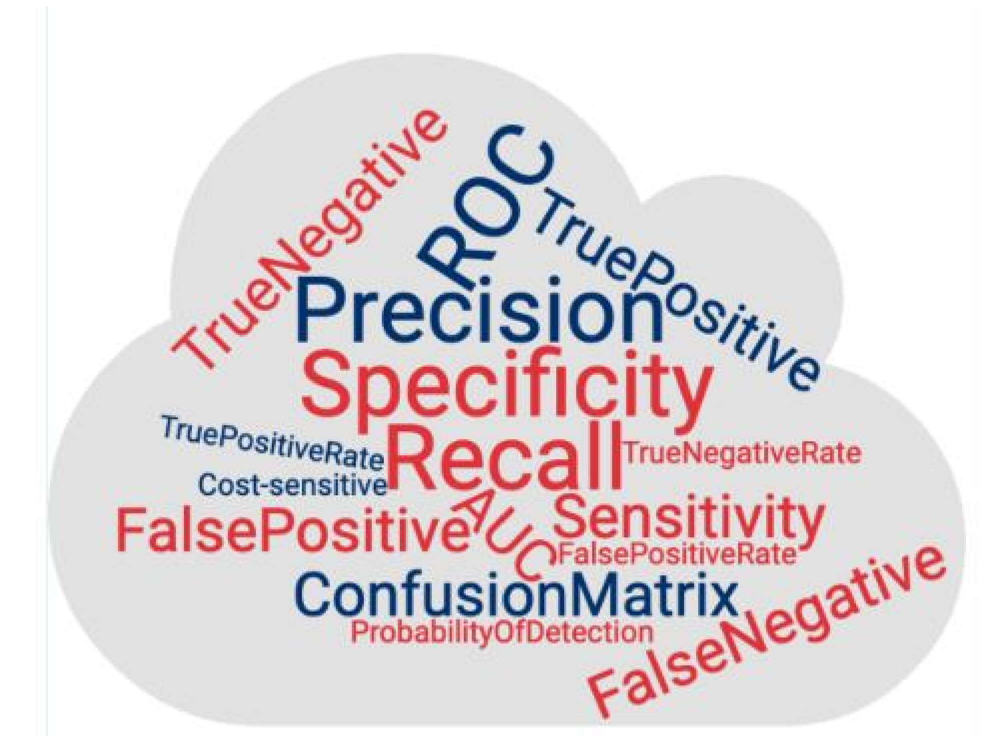



figure reprinted from www.kdnuggets.com with explicit permission of the editor

# A Numerical Dataset

| object $x_i$ | Height (H) | Weight (W) | Foot (F) | Label (L) |
|---|---|---|---|---|
| $x_1$ | 5.00 | 100 | 6 | green |
| $x_2$ | 5.50 | 150 | 8 | green |
| $x_3$ | 5.33 | 130 | 7 | green |
| $x_4$ | 5.75 | 150 | 9 | green |
| $x_5$ | 6.00 | 180 | 13 | red |
| $x_6$ | 5.92 | 190 | 11 | red |
| $x_7$ | 5.58 | 170 | 12 | red |
| $x_8$ | 5.92 | 165 | 10 | red |

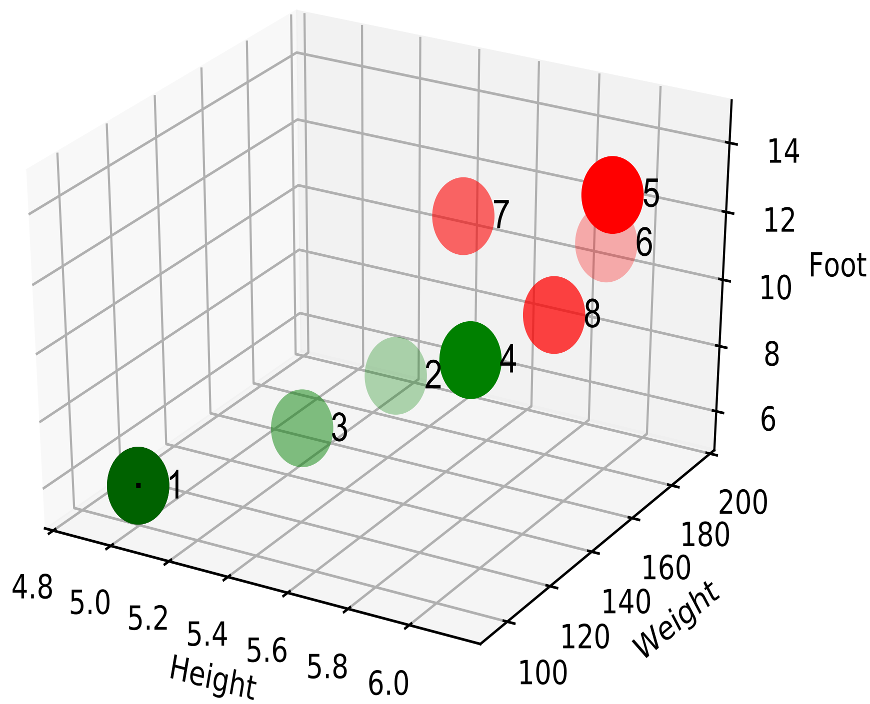
# Code for the Dataset

```python
import pandas as pd

data = pd.DataFrame(
 {"id":[ 1,2,3,4,5,6,7,8],
  "Label":["green","green","green","green",
                   "red","red","red","red"],
  "Height":[5,5.5,5.33,5.75,6.00,5.92,5.58,5.92],
  "Weight":[100,150,130,150,180,190,170,165],
  "Foot":[6, 8, 7, 9, 13, 11, 12, 10]},
  columns=["id","Height","Weight",
                        "Foot","Label"])
```

```
>> data
  id Height Weight Foot Label
0  1   5.00    100    6 green
1  2   5.50    150    8 green
2  3   5.33    130    7 green
3  4   5.75    150    9 green
4  5   6.00    180   13   red
5  6   5.92    190   11   red
6  7   5.58    170   12   red
7  8   5.92    165   10   red
```

# A Dataset Illustration



# Three Models

- objects:

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$$

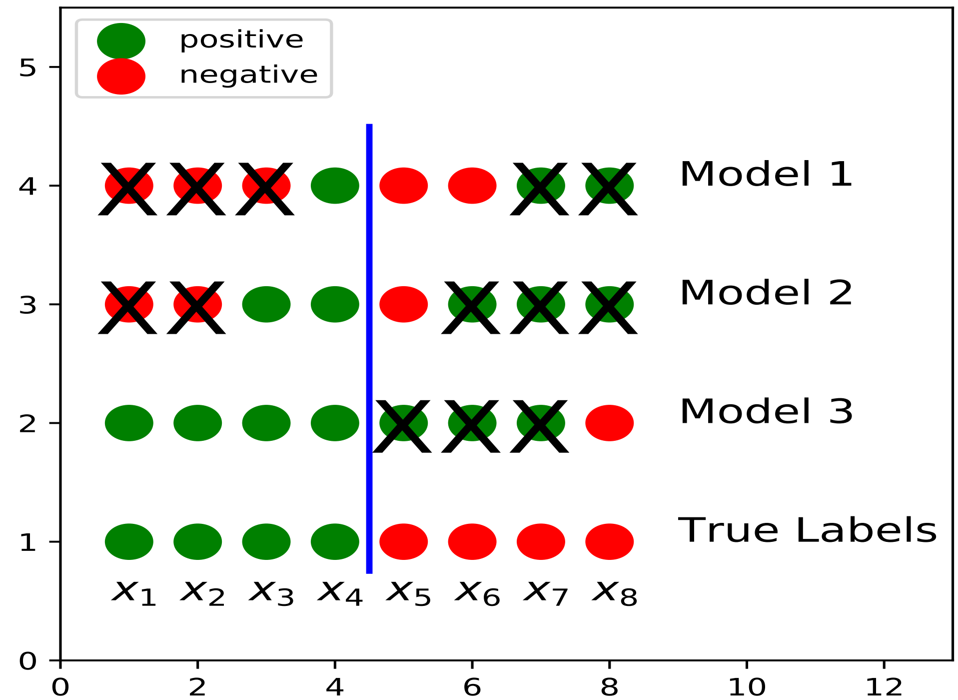
- "green" is 1, 'red" is 0
- true labels: [1,1,1,1,0,0,0,0]
- want to compare 3 models:
  1. Model 1: [0,0,0,1,0,0,1,1]
  2. Model 2: [0,0,1,1,0,1,1,1]
  3. Model 3: [1,1,1,1,1,1,1,0]

# Python Code

```
import pandas as pd
data = pd.DataFrame(
{"id":[ 1,2,3,4,5,6,7,8],
  "Label":["green","green","green","green",
                  "red","red","red","red"],
  "Height":[5,5.5,5.33,5.75,6.00,5.92,5.58,5.92],
  "Weight":[100,150,130,150,180,190,170,165],
  "Foot":[6, 8, 7, 9, 13, 11, 12, 10]},
  columns=["id","Height","Weight",
                        "Foot","Label"])
data["Class"]=data["Label"].apply(lambda x: \
                  1 if x=="green" else 0)
y_true = data["Class"].values
# assume that we got predictions from 3 models:
y_pred_1 = [0,0,0,1,0,0,1,1]
y_pred_2 = [0,0,1,1,0,1,1,1]
y_pred_3 = [1,1,1,1,1,1,1,0]
```

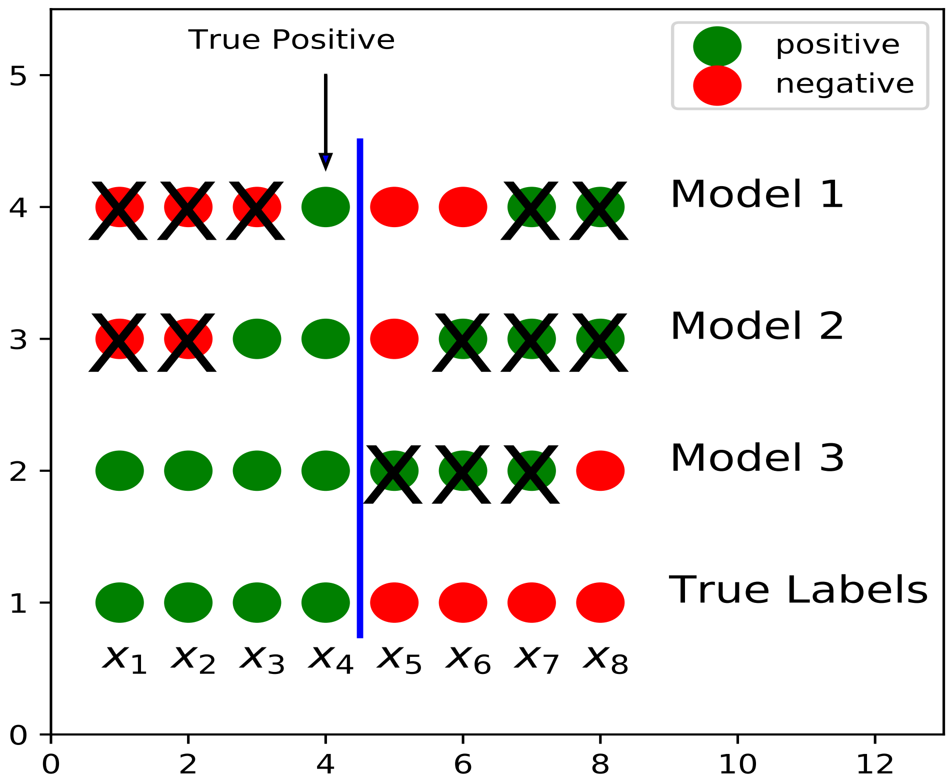
```
>> y_true
[1, 1, 1, 1, 0, 0, 0, 0]
>> y_pred_1
[0, 0, 0, 1, 0, 0, 1, 1]
```

# Comparison of Models



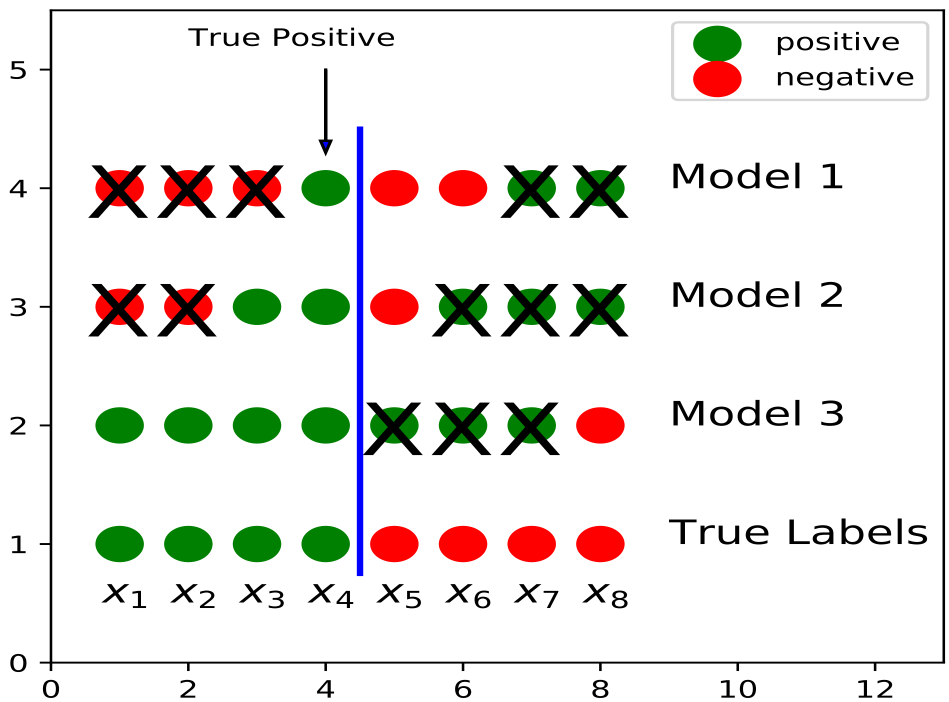

- "same" accuracy for 1 and 2
- sometimes 3 is the "best"
- how do we compare?

# True Positives (TP)



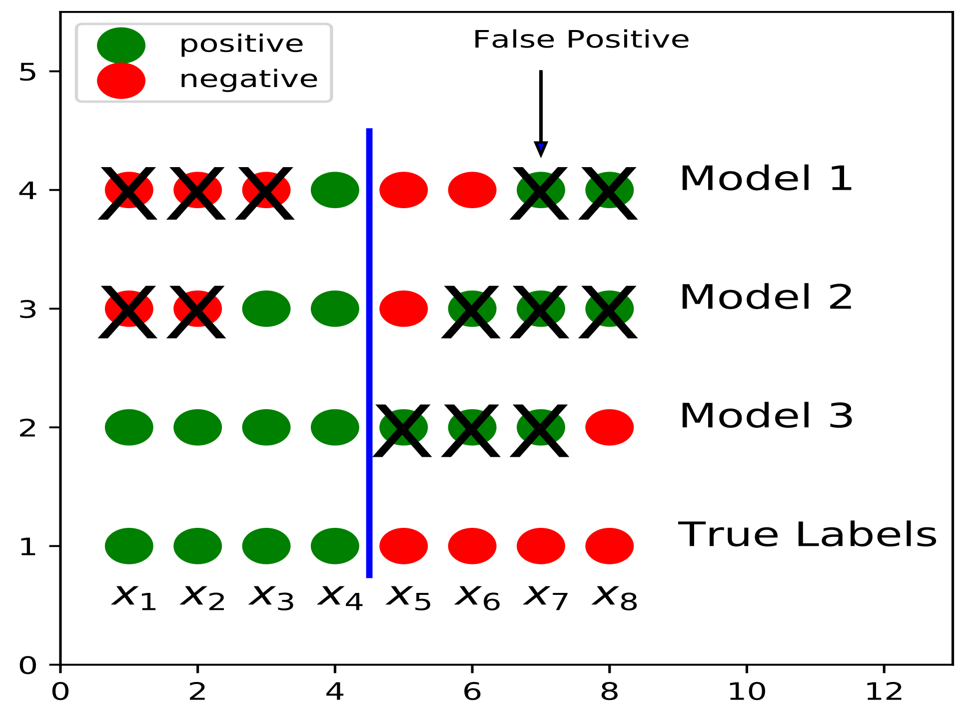

- predicted positive and true labels are positive

# True Positives (TP)



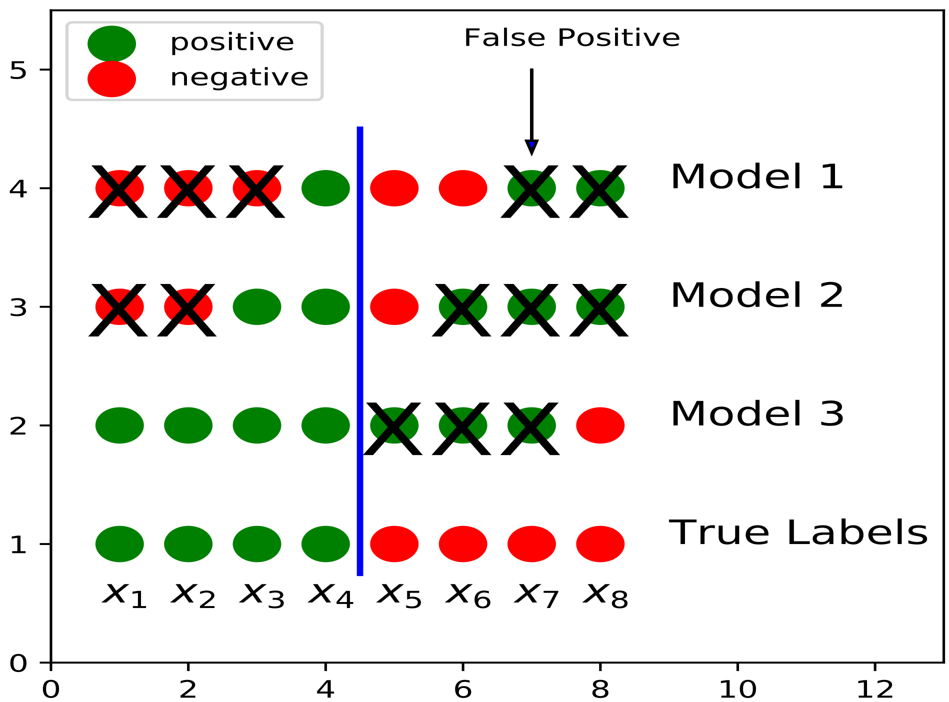

| Model | True Positive | Actual Positive |
|---|---|---|
| 1 | $x_4$ | $x_1, x_2, x_3, x_4$ |
| 2 | $x_3, x_4$ | $x_1, x_2, x_3, x_4$ |
| 3 | $x_1, x_2, x_3, x_4$ | $x_1, x_2, x_3, x_4$ |

# False Positives (FP)



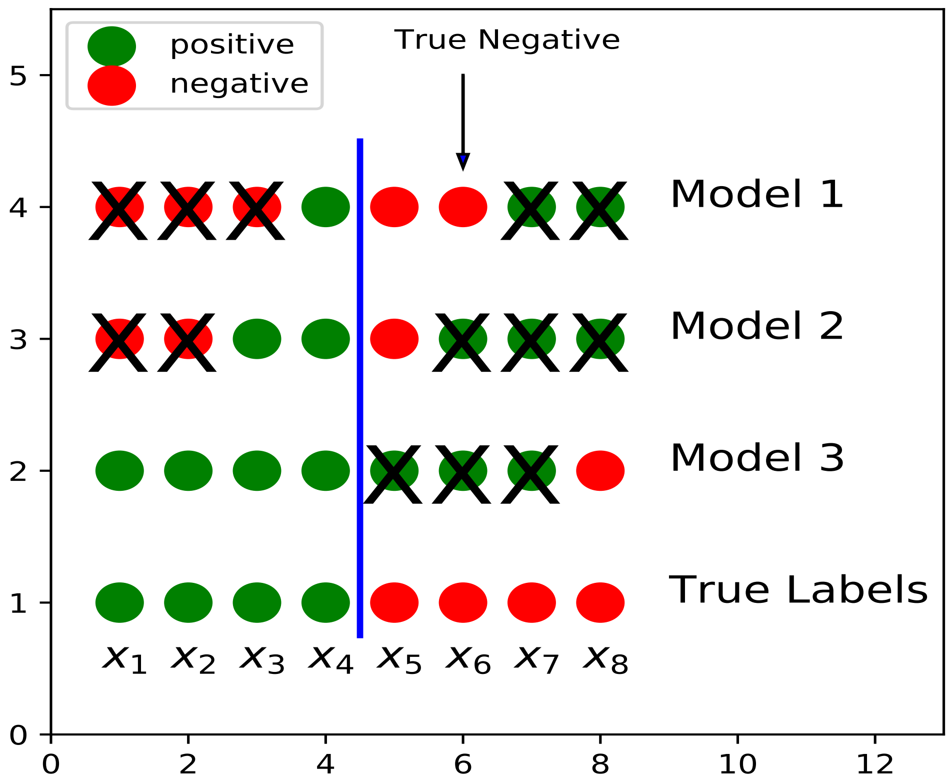

- predicted positive but true labels are negative
- "Type I" error ("false alarm")

# False Positives (FP)



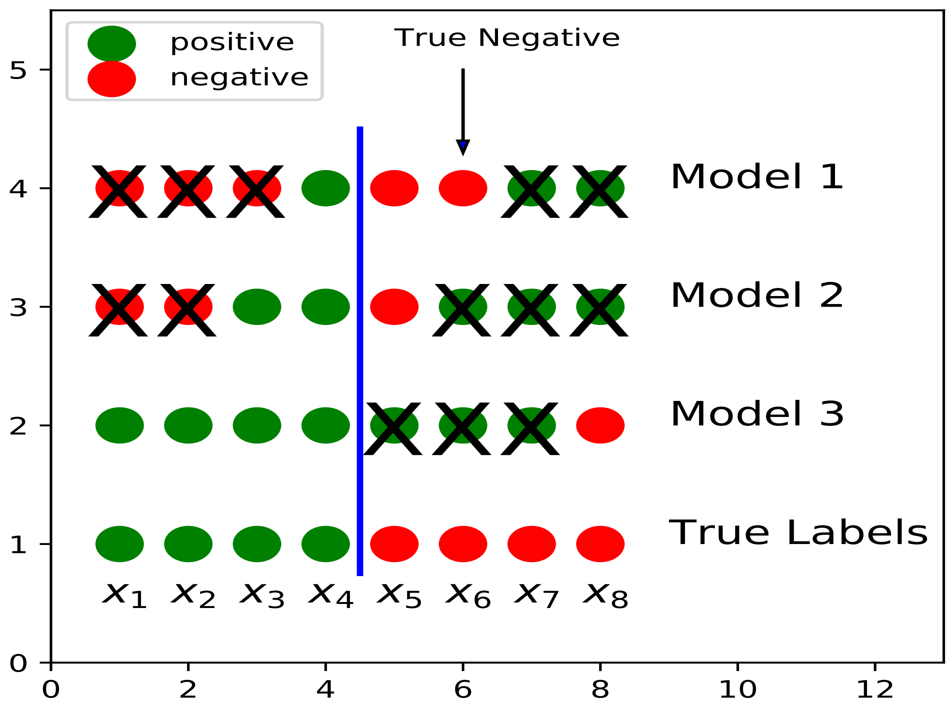

| Model | False Positive | Actual Negative |
|---|---|---|
| 1 | $x_7, x_8$ | $x_5, x_6, x_7, x_8$ |
| 2 | $x_6, x_7, x_8$ | $x_5, x_6, x_7, x_8$ |
| 3 | $x_5, x_6, x_7$ | $x_5, x_6, x_7, x_8$ |

# True Negative (TN)



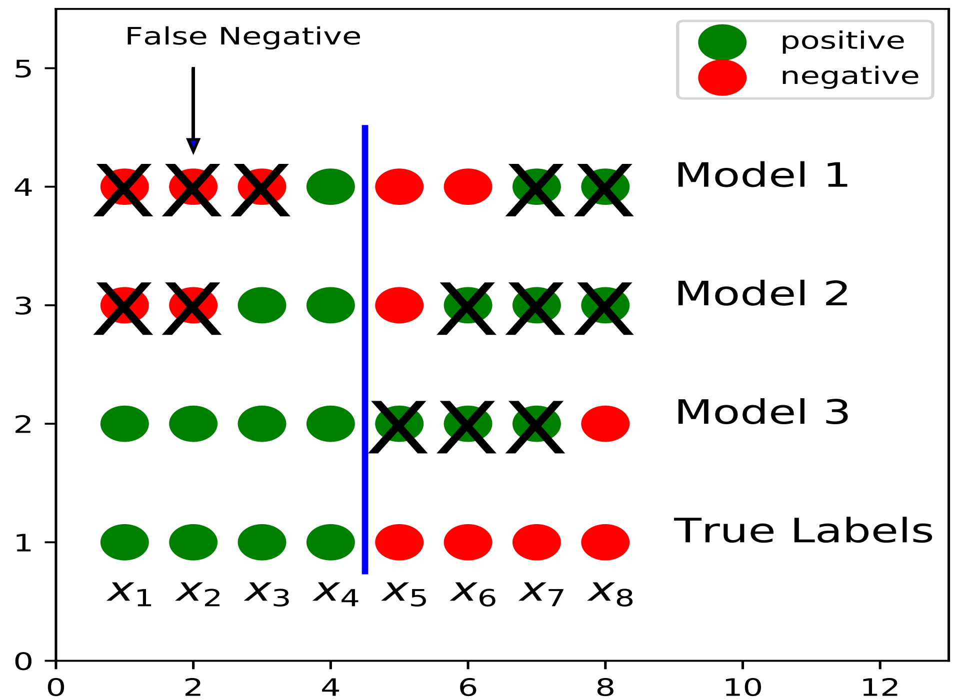

- predicted negative and true labels are negative

# True Negatives (TN)



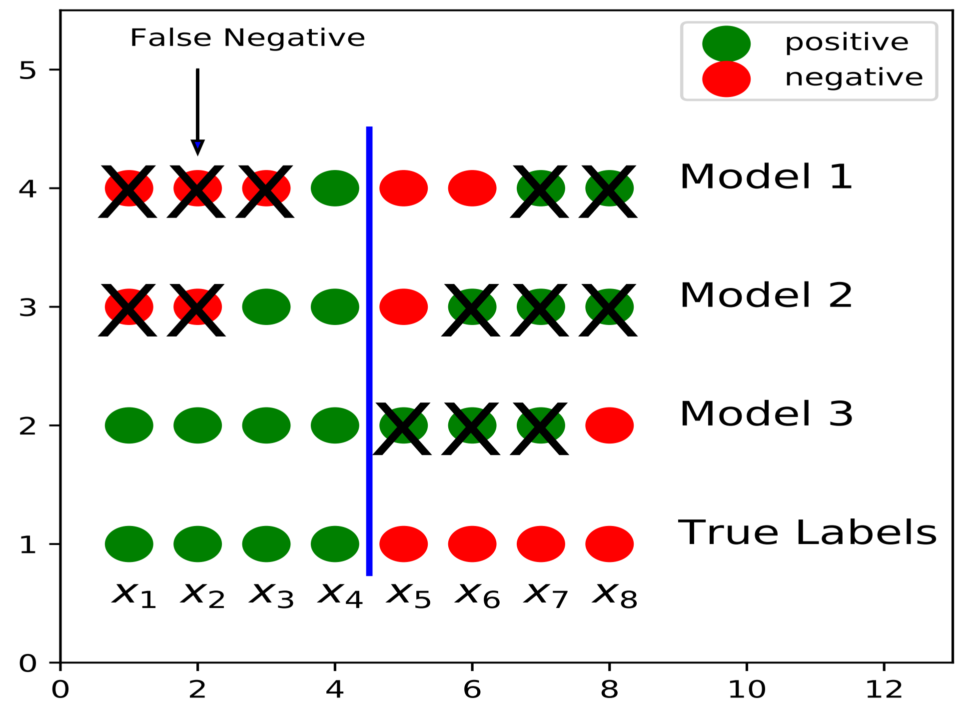

| Model | True Negative | Actual Negative |
|---|---|---|
| 1 | $x_5, x_6$ | $x_5, x_6, x_7, x_8$ |
| 2 | $x_5$ | $x_5, x_6, x_7, x_8$ |
| 3 | $x_8$ | $x_5, x_6, x_7, x_8$ |

# False Negative (FN)



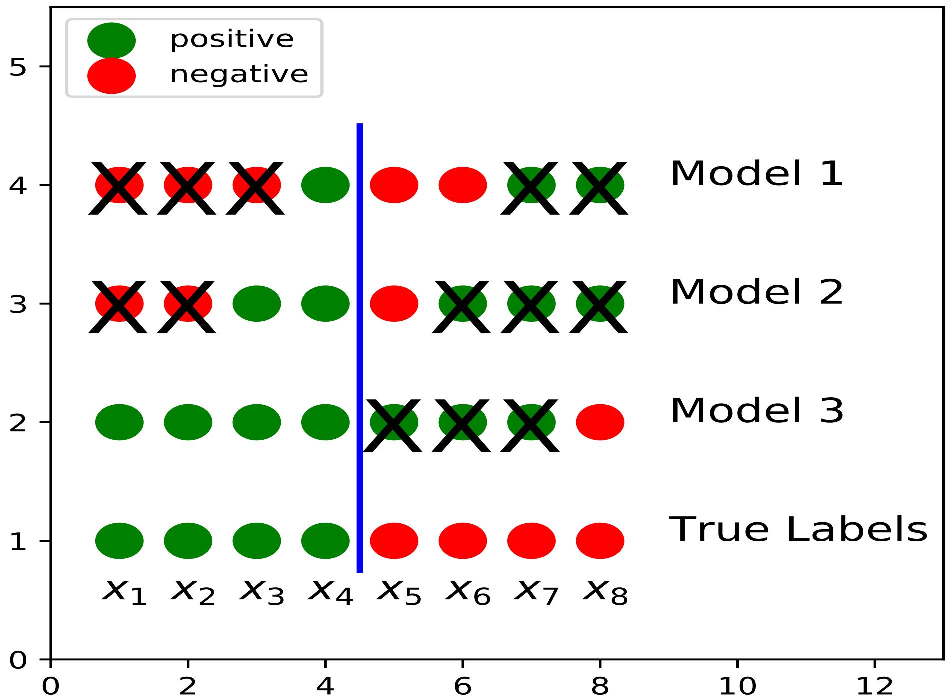

- predicted negative, but true labels are positive
- "Type II" error ("miss")

# False Negative (FN)



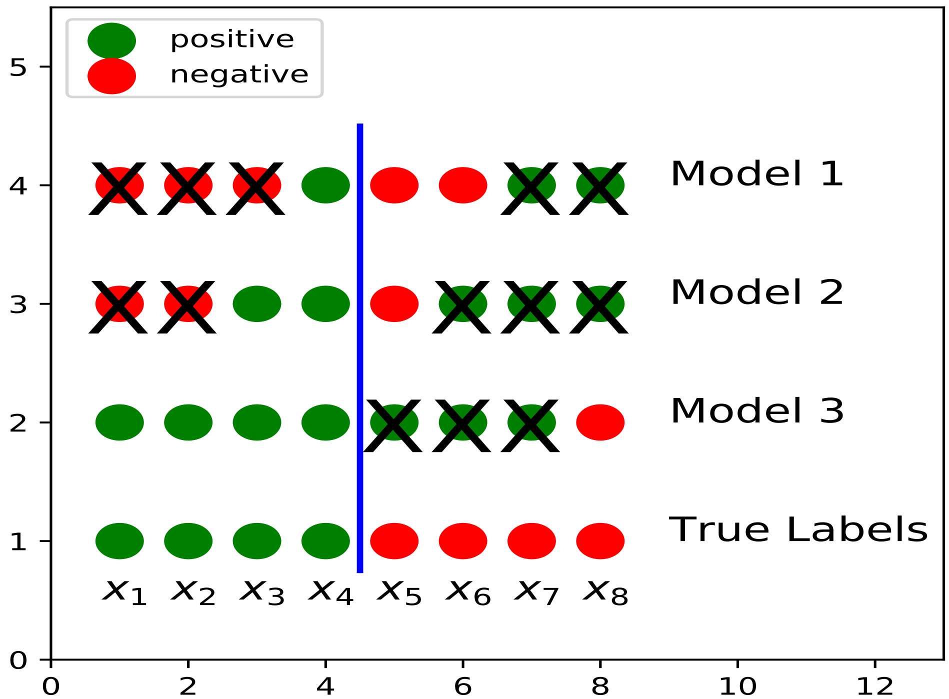

| Model | False Negative | Actual Positive |
|---|---|---|
| 1 | $x_1, x_2, x_3$ | $x_1, x_2, x_3, x_4$ |
| 2 | $x_1, x_2$ | $x_1, x_2, x_3, x_4$ |
| 3 | none | $x_1, x_2, x_3, x_4$ |

# Pos. & Neg. Summary




| Model | TP | FP | TN | FN |
|---|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | $x_5, x_6$ | $x_1, x_2, x_3$ |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | $x_5$ | $x_1, x_2$ |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | $x_8$ | none |

# Pos. & Neg. Summary




| Model | TP | FP | TN | FN |
|---|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | $x_5, x_6$ | $x_1, x_2, x_3$ |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | $x_5$ | $x_1, x_2$ |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | $x_8$ | none |

# Confusion Matrix

- each row represents predictions
- each column represents actual class
- $C = [C_{ij}]$
- $C_{ij}$ - observations in group $j$ predicted for group $i$

# Visual Representation

- rows represent predictions
- columns represent actual class

$$C = \begin{bmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{bmatrix}$$

TN/()

- total positive $\text{P} = \text{TP} + \text{FN}$
- total negative $\text{N} = \text{TN} + \text{FP}$

# Confusion Matrices

| Model | TP | FP | TN | FN |
|---|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | $x_5, x_6$ | $x_1, x_2, x_3$ |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | $x_5$ | $x_1, x_2$ |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | $x_8$ | none |

$$C = \begin{bmatrix} \mathrm{TN} & \mathrm{FP} \\ \mathrm{FN} & \mathrm{TP} \end{bmatrix}$$

$$C_1 = \begin{bmatrix} 2 & 2 \\ 3 & 1 \end{bmatrix},\ C_2 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix},\ C_3 = \begin{bmatrix} 1 & 3 \\ 0 & 4 \end{bmatrix}$$

# Python Code

```
import pandas as pd
from sklearn.metrics import confusion_matrix

data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                   'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                      'Foot', 'Label'] )

data['Class']   = data['Label'].apply(lambda x: 1
                              if x=='green' else 0)
y_true = data['Class'].values
# assume that we got predictions from 3 models:
y_pred_1 = [0,0,0,1,0,0,1,1]
y_pred_2 = [0,0,1,1,0,1,1,1]
y_pred_3 = [1,1,1,1,1,1,1,0]
cf_1 = confusion_matrix(y_true,y_pred_3)
cf_2 = confusion_matrix(y_true,y_pred_3)
cf_3 = confusion_matrix(y_true,y_pred_3)
```

```
>> cf_3
[[1 3]
 [0 4]]
```

# True Positive Rate (TPR)

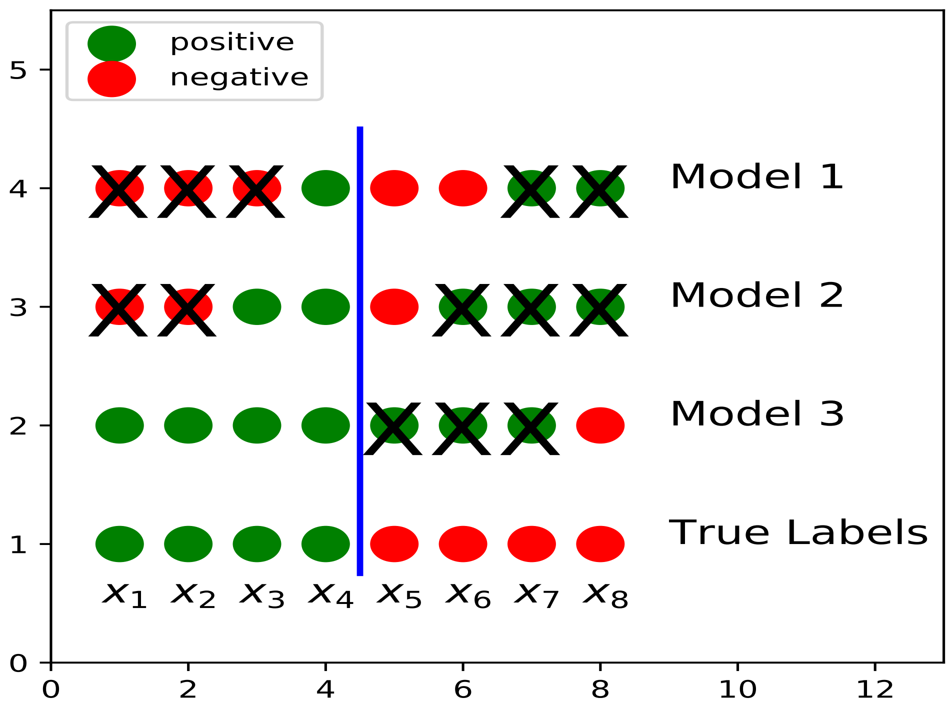
- sensitivity, recall, or hit rate

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

| Model | TP | FN | TPR |
|---|---|---|---|
| 1 | $x_4$ | $x_1, x_2, x_3$ | 0.25 |
| 2 | $x_3, x_4$ | $x_1, x_2$ | 0.5 |
| 3 | $x_1, x_2, x_3, x_4$ | none | 1 |

- fraction of positive labels predicted correctly

# $\mathbf{TPR = TP/(TP + FN)}$




| **Model** | **TP** | **FN** | **TPR** |
|---|---|---|---|
| 1 | $x_4$ | $x_1, x_2, x_3$ | 0.25 |
| 2 | $x_3, x_4$ | $x_1, x_2$ | 0.5 |
| 3 | $x_1, x_2, x_3, x_4$ | none | 1 |

# Python Code

```
import pandas as pd
from sklearn.metrics import recall_score
```

Ture Positive rate call recall

```
data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                  'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                    'Foot', 'Label'] )

data['Class']   = data['Label'].apply(lambda x: 1
                              if x=='green' else 0)
y_true = data['Class'].values
# assume that we got predictions from 3 models:
y_pred_1 = [0,0,0,1,0,0,1,1]
y_pred_2 = [0,0,1,1,0,1,1,1]
y_pred_3 = [1,1,1,1,1,1,1,0]

tpr_1 = recall_score(y_true, y_pred_1)
tpr_2 = recall_score(y_true, y_pred_2)
tpr_3 = recall_score(y_true, y_pred_3)
```

```
>> print(tpr_1, tpr_2, tpr_3)

0.25 0.5 1.0
```

# True Negative Rate (TNR)

- "specificity" or "selectivity"

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

| **Model** | **FP** | **TN** | **TNR** |
|---|---|---|---|
| 1 | $x_7, x_8$ | $x_5, x_6$ | 0.50 |
| 2 | $x_6, x_7, x_8$ | $x_5$ | 0.25 |
| 3 | $x_5, x_6, x_7$ | $x_8$ | 0.25 |

- fraction of negative labels predicted correctly

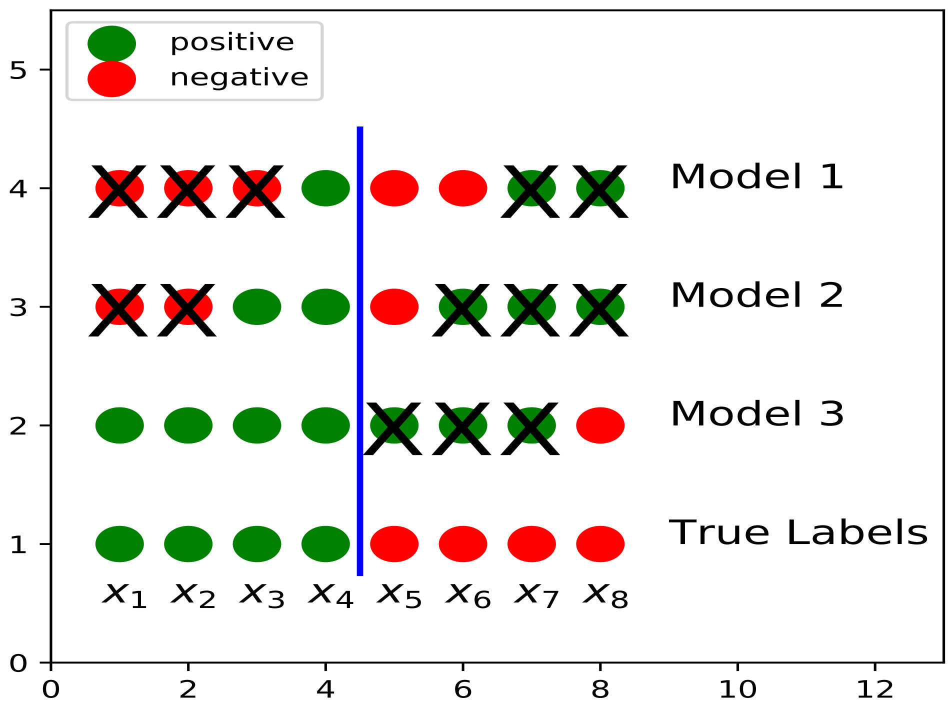
# Positive Predicted Value (PPV)

- "precision"

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

| **Model** | **TP** | **FP** | **PPV** |
|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | 0.33 |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | 0.40 |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | 0.57 |

# PPV = TP/(TP+FP)




| Model | TP | FP | PPV |
|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | 0.33 |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | 0.40 |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | 0.57 |

# Python Code

```
import pandas as pd
from sklearn.metrics import precision_score

data = pd.DataFrame(                              fa
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                  'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                  6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                  180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                     'Foot', 'Label'] )

data['Class']   = data['Label'].apply(lambda x: 1
                                 if x=='green' else 0)
y_true = data['Class'].values
# assume that we got predictions from 3 models:
y_pred_1 = [0,0,0,1,0,0,1,1]
y_pred_2 = [0,0,1,1,0,1,1,1]
y_pred_3 = [1,1,1,1,1,1,1,0]

ppv_1 = precision_score(y_true, y_pred_1)
ppv_2 = precision_score(y_true, y_pred_2)
ppv_3 = precision_score(y_true, y_pred_3)
```

```
>> print(ppv_1, ppv_2, ppv_3)

0.333333333333 0.4 0.571428571429
```

# Negative Predicted Value (NPV)

$$\text{NPV} = \frac{\text{TN}}{\text{TN} + \text{FN}}$$

| **Model** | **TN** | **FN** | **NPV** |
|---|---|---|---|
| 1 | $x_5, x_6$ | $x_1, x_2, x_3$ | 0.40 |
| 2 | $x_5$ | $x_1, x_2$ | 0.33 |
| 3 | $x_8$ | none | 1.0 |

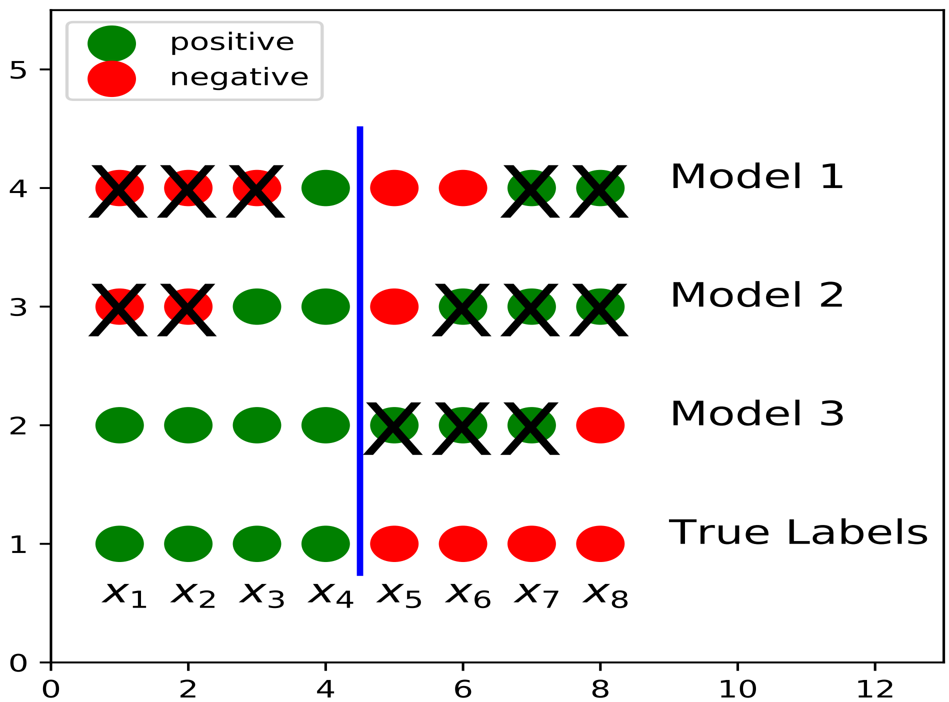
# Accuracy

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

| Model | TP | FP | TN | FN | ACC |
|---|---|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | $x_5, x_6$ | $x_1, x_2, x_3$ | 0.375 |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | $x_5$ | $x_1, x_2$ | 0.375 |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | $x_8$ | none | 0.625 |

- fraction of all labels predicted correctly
- models 1 and 2 - same accuracy, different precision

# ACC = (TP+TN)/ALL




| Model | TP | FP | TN | FN | ACC |
|---|---|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | $x_5, x_6$ | $x_1, x_2, x_3$ | 0.375 |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | $x_5$ | $x_1, x_2$ | 0.375 |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | $x_8$ | none | 0.625 |

# Python Code

```
import pandas as pd
from sklearn.metrics import accuracy_score

data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                   'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                      'Foot', 'Label'] )

data['Class']   = data['Label'].apply(lambda x: 1
                              if x=='green' else 0)
y_true = data['Class'].values
# assume that we got predictions from 3 models:
y_pred_1 = [0,0,0,1,0,0,1,1]
y_pred_2 = [0,0,1,1,0,1,1,1]
y_pred_3 = [1,1,1,1,1,1,1,0]

acc_1 = accuracy_score(y_true, y_pred_1)
acc_2 = accuracy_score(y_true, y_pred_2)
acc_3 = accuracy_score(y_true, y_pred_3)
```

```
>> print(acc_1, acc_2, acc_3)

0.375 0.375 0.625
```

# $F_1$ Score

- harmonic mean of precision and sensitivity

$$F_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}}$$

$$= \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

| **Model** | **TP** | **FP** | **FN** | $F_1$ |
|---|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | $x_1, x_2, x_3$ | 0.29 |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | $x_1, x_2$ | 0.44 |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | none | 0.73 |

# Python Code

```
import pandas as pd
from sklearn.metrics import f1_score

data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                   'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                      'Foot', 'Label'] )

data['Class']   = data['Label'].apply(lambda x: 1
                           if x=='green' else 0)
y_true = data['Class'].values
# assume that we got predictions from 3 models:
y_pred_1 = [0,0,0,1,0,0,1,1]
y_pred_2 = [0,0,1,1,0,1,1,1]
y_pred_3 = [1,1,1,1,1,1,1,0]

f1_1 = f1_score(y_true,y_pred_3)
f1_2 = f1_score(y_true,y_pred_3)
f1_3 = f1_score(y_true,y_pred_3)
```

```
>> print(f1_1, f1_2, f1_3)

0.2857142857 0.4444444444 0.7272727272
```

# Comparing Models

| Metric | Model 1 | Model 2 | Model 3 |
|---|---|---|---|
| recall (TPR) | 0.25 | 0.5 | 1 |
| specificity (TPR) | 0.5 | 0.25 | 0.25 |
| precision (PPV) | 0.33 | 0.4 | 0.57 |
| accuracy | 0.375 | 0.375 | 0.625 |
| $F_1$ | 0.29 | 0.44 | 0.73 |

- choice of model depends on the metric

# Additional Measures

- False Negative Rate:

$$\mathrm{FNR} = 1 - \mathrm{TPR}$$

- False Positive Rate:

$$\mathrm{FPR} = 1 - \mathrm{TNR}$$

- False Discovery Rate:

$$\mathrm{FDR} = 1 - \mathrm{PPV}$$

- False Omission Rate:

$$\mathrm{FOR} = 1 - \mathrm{NPV}$$

# False Negative Rate (FNR)

- ”miss” rate

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}}$$

| **Model** | **TP** | **FN** | **FNR** |
|---|---|---|---|
| 1 | $x_4$ | $x_1, x_2, x_3$ | 0.75 |
| 2 | $x_3, x_4$ | $x_1, x_2$ | 0.5 |
| 3 | $x_1, x_2, x_3, x_4$ | none | 0 |

# False Positive Rate (FPR)

- "fall-out"

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

| **Model** | **FP** | **TN** | **FPR** |
|---|---|---|---|
| 1 | $x_7, x_8$ | $x_5, x_6$ | 0.5 |
| 2 | $x_6, x_7, x_8$ | $x_5$ | 0.75 |
| 3 | $x_5, x_6, x_7$ | $x_8$ | 0.75 |

# False Discovery Rate (FDR)

$$\text{FDR} = \frac{\text{FP}}{\text{FP} + \text{TP}}$$

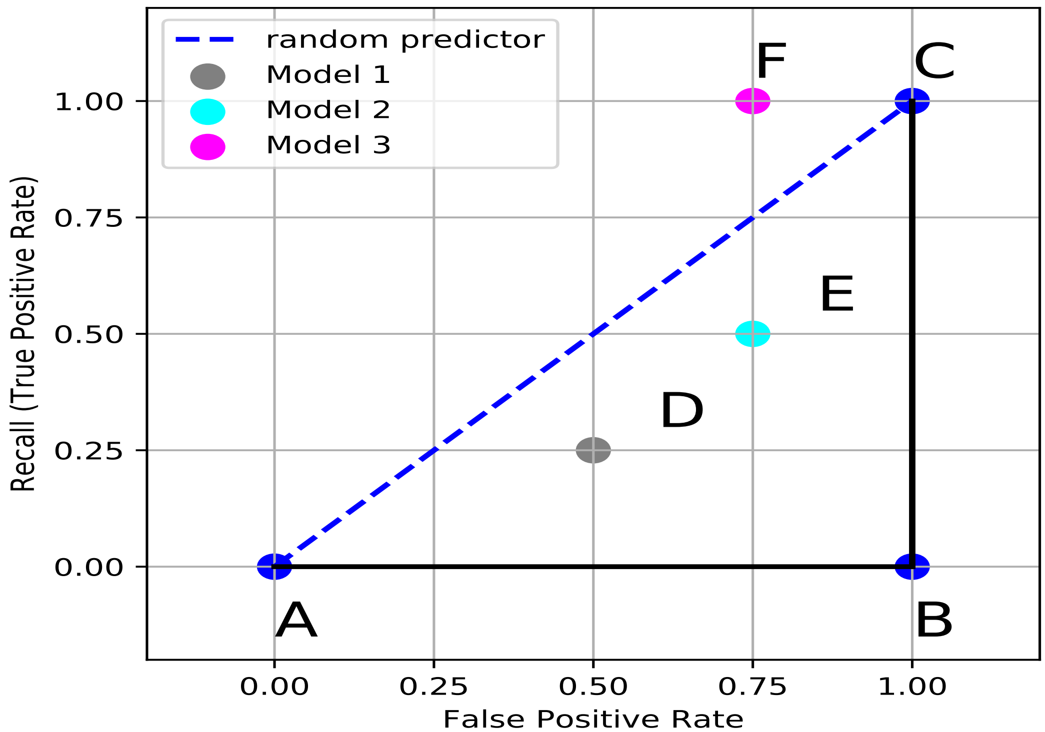
| **Model** | **TP** | **FP** | **FDR** |
|---|---|---|---|
| 1 | $x_4$ | $x_7, x_8$ | 0.67 |
| 2 | $x_3, x_4$ | $x_6, x_7, x_8$ | 0.60 |
| 3 | $x_1, x_2, x_3, x_4$ | $x_5, x_6, x_7$ | 0.43 |

# False Omission Rate (FOR)

$$\text{FOR} = \frac{\text{FN}}{\text{FN} + \text{TN}}$$

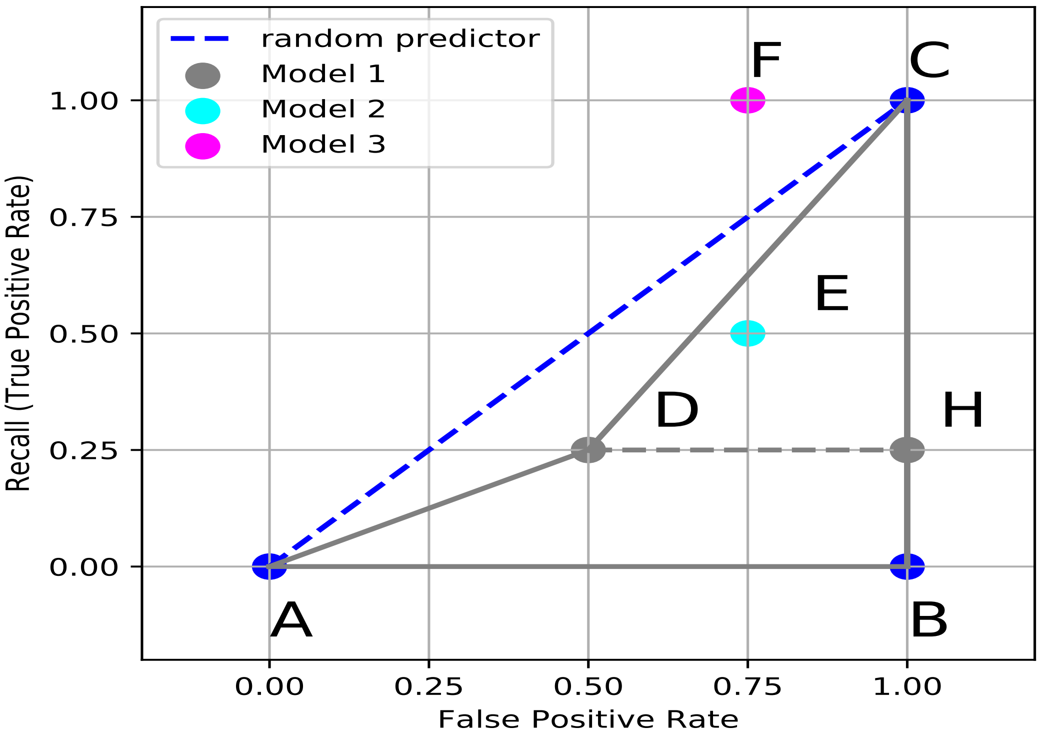
| **Model** | **TN** | **FN** | **FOR** |
|---|---|---|---|
| 1 | $x_5, x_6$ | $x_1, x_2, x_3$ | 0.60 |
| 2 | $x_5$ | $x_1, x_2$ | 0.66 |
| 3 | $x_8$ | none | 0 |

# ROC/AUC Curve



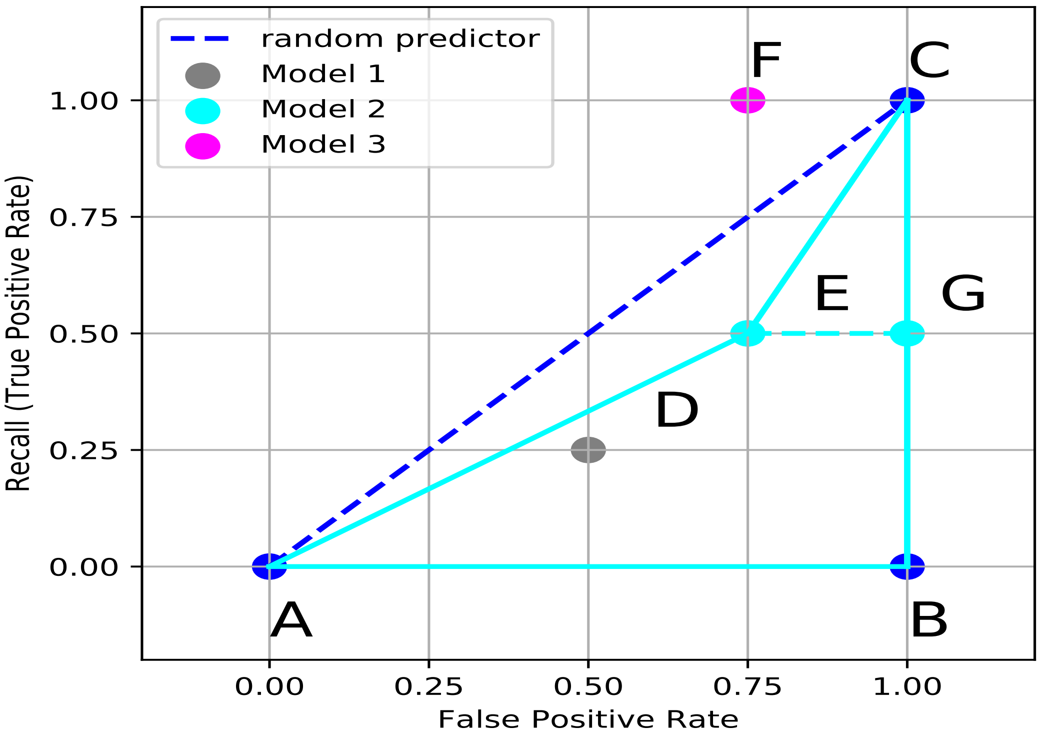

- receiver operating characteristic (ROC) - describe binary classifiers
- area under curve (AUC) - compare classifiers vs. random

# AUC For Model 1



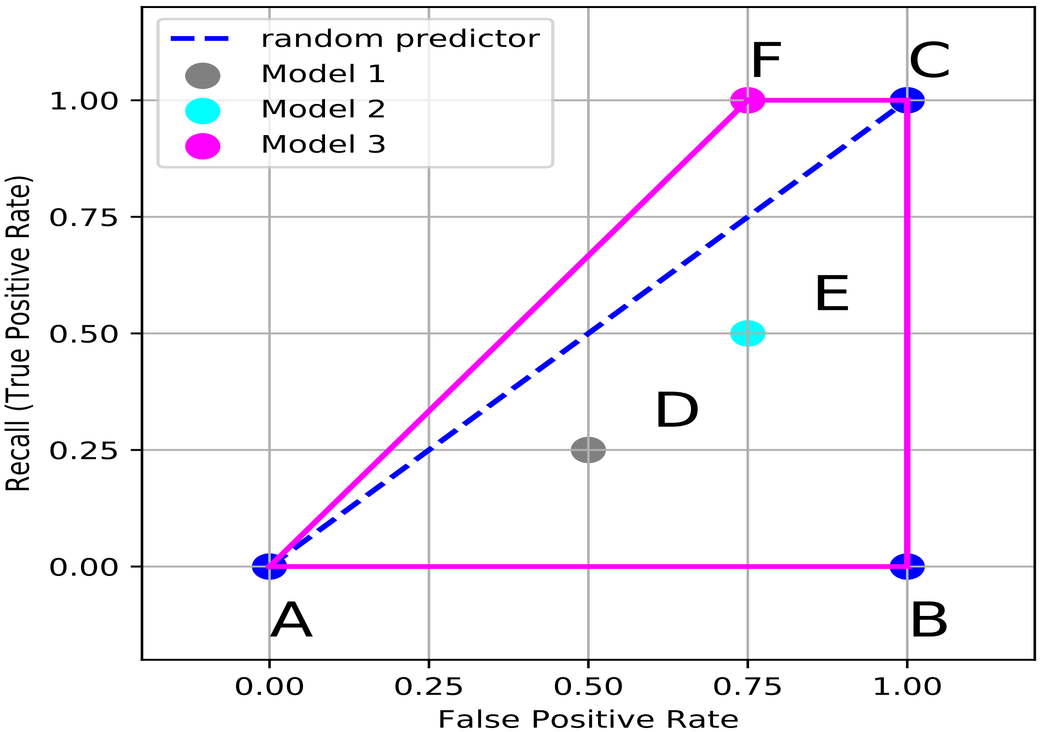

$$\begin{aligned}\text{area}(ABCD) &= \text{area}(ABDH) + \text{area}(DHC) \\ &= \frac{(1+0.5)}{2}\cdot 0.25 + 0.5\cdot 0.5\cdot 0.75 \\ &= 0.1875 + 0.1875 = 0.375\end{aligned}$$

# AUC For Model 2




$$
\begin{aligned}
\text{area}(ABCE) &= \text{area}(ABGE) + \text{area}(GCE) \\
&= \frac{(1+0.25)}{2} \cdot 0.5 + 0.25 \cdot 0.5 \cdot 0.5 \\
&= 0.3125 + 0.0625 = 0.375
\end{aligned}
$$

# AUC For Model 3




$$\text{area}(ABCF) = \frac{(1 + 0.25)}{2} = 0.625$$

# Python Code

```
import pandas as pd
from sklearn.metrics import roc_auc_score

data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                   'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                      'Foot', 'Label'] )

data['Class']   = data['Label'].apply(lambda x: 1
                               if x=='green' else 0)
y_true = data['Class'].values
# assume that we got predictions from 3 models:
y_pred_1 = [0,0,0,1,0,0,1,1]
y_pred_2 = [0,0,1,1,0,1,1,1]
y_pred_3 = [1,1,1,1,1,1,1,0]

auc_1 = roc_auc_score(y_true, y_pred_1)
auc_2 = roc_auc_score(y_true, y_pred_2)
auc_3 = roc_auc_score(y_true, y_pred_3)
```

```
>> print(auc_1, auc_2, auc_3)

0.375  0.375  0.625
```

# Concepts Check:

(a) true and false positive

(b) true and false negatives

(c) sensitivity (or recall)

(d) specificity, precision

(e) type I and II error

(f) confusion matrix

(g) $F_1$ score

(h) receiver operating characteristic (ROC)

(i) area uner curve (AUC)