## Model Evaluation: key classification metrics

○ Accuracy 🡪 (Cons: Unbalanced data)

○ Recall

○ Precision

○ F1-Score -->Hormonic --> 2\* (Precision \* Recall / Precision + Recall)

A picture containing table

Description automatically generated

Regression Metric Error types

### Evaluation metrics for regression🡪 (True Value, (predicted value))

from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error

○ Mean Absolute Error (MAE)

A picture containing text, gauge

Description automatically generated

MAE = mean\_absolute\_error(y\_test,test\_predictions)

Make Absolute average of True value vs Predicted value

○ Mean Squared Error (MSE) 🡪 MSE "punishes" larger errors

Text

Description automatically generated

MSE = mean\_squared\_error(y\_test,test\_predictions)

Make Squared average of True value vs Predicted value

○ Root Mean Square Error (RMSE)

Text

Description automatically generated

RMSE = np.sqrt(MSE)

Make Root Squared average of True value vs Predicted value

# Linear Regression with SciKit-Learn

#### numpy.polyfit()

p = np.polyfit(x, y, d) 🡪 ‘d’ *argument is degree of polynomial*

predictFun = np.poly1d(p) 🡪 returns full equation as per Degree arg

y\_pred = predictFun(x\_test) 🡪 return results for a given value

np.polyfit(x, y, 1) linear polynomial coefficient value

***y=mx+c(returns* m** **c*)*** two constant-coefficients

np.polyfit(x, y, 2) quadratic polynomial coefficient value

***y=ax\*\*2+bx+c (returns* a, b** and **c*)*** three constant-coefficients

np.polyfit(x, y, 3) cubic polynomial coefficient value

***y=ax\*\*3+bx\*\*2+cx+d (returns* a, b, c,** and **d*)*** four constant-coefficients