(110) Classification Measures.

* finding how good or bad our classification algorithm is

↓

in case of linear regression we used coefficient of determination

↓

i) accuracy (how many, we got right, how many we got wrong)

← 50 0's, 50 1's

↓

95 predicted correctly

↓

95 % accuracy

↳ problem: if the dataset is very skew i.e.

95 0's, 5 1's

↓

our classifier starts predicted all 0's

↓

still the accuracy will be 95%.

even though the classifier is pretty bad.

↓

Hence, accuracy might not be a good metric to judge classification algos.

ii) confusion matrix

	pred 0	pred 1
true 0	40	10
true 1	5	45

class 0 → 50 → 40 → 0, 10 → 1

class 1 → 50 → 45 → 1, 5 → 0

true 0	95	0
true 1	5	0

↳ indicative that something is wrong

class 0 → 95 → 95 → 0, 0 → 1

class 1 → 5 → 5 → 1, 0 → 0

* confusion matrix gives an insight (better) on how the classification algorithm is performing.
↓
degree of confusion between classes

(*) terms : i) true +ve : predicted +ve and they were true

ii) true -ve : predicted -ve and they were true.

iii) false +ve : predicted +ve and they were false (they were actually -ve)
↓
prediction

iv) false -ve : similar to above
↓
I predicted -ve, but I was wrong.

usable for binary majorly.
↓
but confusion matrix will be used all the time

* Confusion matrix for multi-class problems.
↓
predicting b/w 0 to 9.

	0	1	2	3	4	5	6	7	8	9
0	90	0	1	0	1	0	0	0	5	1

→ confusion matrix for 0
↓
tells us the classes b/w which our algo. is getting confused.

(112) * confusion matrix on iris data (for logistic regression) using sklearn

↓

i) load iris from sklearn

ii) split it

iii) load Logistic Regression

iv) fit on logistic Regression

v) predict using logistic regression

v) import confusion matrix from sklearn.matrics.

↓

```
from sklearn.metrics import confusion_matrix
```

↓

```
confusion_matrix(y_train, y_train_pred)
```

(y_train → true, y_train_pred → predicted)

class 1 had 31 dp and all were correctly classified

31	0	0
0	41	4
0	11	35

class 2 had 45 total datapoints and 41 were correctly classified, whereas 4 were classified as class 3

↓

insight into what the classifier is doing.

↓

classifier accuracy → all datapoints predicted right / all datapoints

↓

for the above → 107/112

↓

0.95535

* extending concept of confusion matrix to come up with metrics to decide how well certain algorithm is performing.

↓

without them, it is majorly intuition based

(column) i) precision → how many of the total predicted values of a certain class are right

(row) ii) recall → how many true values of a class were predicted correctly.

how many values among actual values were recalled

↓

these values are always for a certain class.

	pred a	pred b
true class a	40	10
true class b	8	42

→ pre(a) = 40/48

pre(b) = 42/52

recall(a) = 40/50

recall(b) = 42/50

	pred a	pred b
a	95	0
b	5	0

↓

we would want precision and recall to be high for any class

↓

precision(a) = 95/100 = 0.95

Precision(b) = 0/0 (ND)

recall(a) = 95/95 = 1

← recall(b) = 0/5 = 0

classifier is skewed or biased

↑

very bad on class b

(14) * to use precision and recall from sklearn

↓

we need to import them from

sklearn

↓

```
from sklearn.metrics import classification_report
```

↓

```
print(classification_report(y_test, y_test_pred))
```

↓

f1 score → harmonic mean of precision and recall, of a class

↓

$$\frac{2}{\frac{1}{x} + \frac{1}{y}}$$

↓

to reach to a single no. rather than depending on two

support → actual true values of a class that we had in the dataset