Here is a very basic Python example using **KNN regression** to predict the price of a house based on its **size and number of bedrooms**:

```
from sklearn.neighbors import KNeighborsRegressor

# Training data
# [size in sq ft, number of bedrooms]
X = [
    [1000, 2],
    [1200, 2],
    [1500, 3],
    [1600, 3],
    [2000, 4]
]

# House prices
y = [
    200000,
    220000,
    280000,
    300000,
    360000
]

# Create the KNN regression model
knn = KNeighborsRegressor(n_neighbors=3)

# Train the model
knn.fit(X, y)

# New house: 1550 sq ft, 3 bedrooms
new_house = [[1550, 3]]

# Predict the price
prediction = knn.predict(new_house)

print("Predicted house price: $", prediction[0])
```

Expected result

Approximately:

```
Predicted house price: $ 313333.33
```

What is happening?

The important line is:

Python Ejecutar

This tells Python:

> **Use the 3 houses most similar to the new house.**

Then:

Python Ejecutar

asks:

> **What is the predicted price of a 1,550 sq-ft house with 3 bedrooms?**

KNN finds the **3 nearest houses** and averages their prices.

$$\frac{280000 + 300000 + 360000}{3} = 313333.33$$

So:

$$\boxed{\text{Predicted price} \approx \$313,333}$$

One important teaching point: in a real KNN model, the features should usually be **standardized** because square footage and number of bedrooms are on very different scales. This simple example intentionally leaves that out so students can see the basic idea first.

provide the code for the classification example too

Absolutely. Here is the same **simple KNN classification example** using the student **Pass/Fail** problem.

```
from sklearn.neighbors import KNeighborsClassifier

# Training data
# [hours studied, assignments completed]
X = [
    [1, 2],
    [2, 1],
    [2, 3],
    [3, 2],
    [7, 8],
    [8, 7],
    [9, 8],
    [8, 9]
]

# Results
# 0 = Fail, 1 = Pass
y = [
    0,
    0,
    0,
    0,
    1,
    1,
    1,
    1
]

# Create the KNN classification model
knn = KNeighborsClassifier(n_neighbors=3)

# Train the model
knn.fit(X, y)

# New student
# 6 hours studied, 7 assignments completed
```

```
new_student = [[6, 7]]
```

Python

Ejecutar

```
prediction = knn.predict(new_student)

# Display result
if prediction[0] == 1:
    print("Prediction: Pass")
else:
    print("Prediction: Fail")
```

Expected output

```
Prediction: Pass
```

What KNN is doing

We have a new student:

$$(6, 7)$$

KNN looks for the **3 closest students**:

```
(7,8) → Pass
(8,7) → Pass
(8,9) → Pass
```

The majority is:

```
Pass = 3
Fail = 0
```

Therefore:

$$\boxed{\text{Prediction} = \text{Pass}}$$

Classification vs. Regression

The two examples are useful together:

Problem	KNN model	What do we do with neighbors?
Student → Pass/Fail	`KNeighborsClassifier`	**Majority vote**

Problem	KNN model	What do we do with neighbors?
House → Price	`KNeighborsRegressor`	**Average values**

The key conceptual difference for students is:

Classification → vote

Regression → average

define in this examples sensitivity, especificity and accuracy in the classification examples

In the **student Pass/Fail KNN classification example**, sensitivity, specificity, and accuracy are calculated from the **confusion matrix**.

First, remember:

- **1 = Pass**
- **0 = Fail**

1. Confusion matrix

Suppose we test KNN on 100 students and obtain:

	Actual Pass	Actual Fail
Predicted Pass	40	5
Predicted Fail	10	45

These four values have names:

- **True Positive (TP) = 40** → predicted Pass, actually Pass
- **False Positive (FP) = 5** → predicted Pass, actually Fail
- **False Negative (FN) = 10** → predicted Fail, actually Pass
- **True Negative (TN) = 45** → predicted Fail, actually Fail

2. Sensitivity

Sensitivity answers:

> Of all the students who actually passed, how many did KNN correctly identify as passing?

Formula:

$$Sensitivity = \frac{TP}{TP + FN}$$

Therefore:

$$Sensitivity = \frac{40}{40 + 10} = \frac{40}{50} = 0.80$$

$$\boxed{Sensitivity = 80\%}$$

So KNN correctly identifies **80% of the students who actually pass**.

---

3. Specificity

Specificity answers:

> Of all the students who actually failed, how many did KNN correctly identify as failing?

Formula:

$$Specificity = \frac{TN}{TN + FP}$$

Therefore:

$$Specificity = \frac{45}{45 + 5} = \frac{45}{50} = 0.90$$

$$\boxed{Specificity = 90\%}$$

So KNN correctly identifies **90% of the students who actually fail**.

---

4. Accuracy

Accuracy answers:

> How many predictions did KNN get correct overall?

Formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Therefore:

$$Accuracy = \frac{40 + 45}{40 + 45 + 5 + 10}$$

$$= \frac{85}{100} = 0.85$$

$$\boxed{Accuracy = 85\%}$$

So KNN makes the correct prediction for **85% of all students**.

Simple interpretation

Metric	Question	Result
Sensitivity	How well do we identify students who **pass**?	**80%**
Specificity	How well do we identify students who **fail**?	**90%**
Accuracy	How many predictions are correct overall?	**85%**

Easy way to remember

If **Pass = Positive**:

Sensitivity → "Find the Passes"

$$\boxed{TP/(TP + FN)}$$

Specificity → "Find the Fails"

$$\boxed{TN/(TN + FP)}$$

Accuracy → "How often am I right?"

$$\boxed{(TP + TN)/(TP + TN + FP + FN)}$$

One important point for your students: **accuracy alone can be misleading when the classes are unbalanced**. For example, if 95% of students pass, a model that always predicts "Pass" has 95% accuracy but has **0% specificity**.