**Machine Learning with scikit – Learn**

**What is Machine Learning (ML)?**

👉 Machine Learning is a branch of **Artificial Intelligence (AI)** where machines (computers) are given the ability to learn to make decisions from data **without being explicitly programmed**.

* Instead of writing fixed rules, we feed data, and the algorithm learns patterns to make predictions or decisions.

**Example of ML in daily life:**

* **Spam Email Detection**: Gmail automatically classifies emails as *spam* or *not spam* by learning from patterns in past data.
* **Netflix/YouTube Recommendation**: Suggests videos or movies based on your watch history.
* **Self-driving Cars**: Learn to detect objects, pedestrians, and traffic signals from image/video data.

**Types of Machine Learning**

**1. Supervised Learning**

* In supervised learning, the data given to the algorithm is **labeled** (i.e., we already know the correct answers/output).
* The model learns the relationship between **input (X)** and **output (Y)**.
* Later, when new (unseen) input comes, it can predict the output.

**Examples:**

* **Predicting House Price**: Input = size, location, rooms; Output = house price.
* **Email Classification**: Input = email text; Output = spam or not spam.
* **Medical Diagnosis**: Input = patient symptoms/test results; Output = disease type.

**2. Unsupervised Learning**

* In unsupervised learning, the data is **unlabeled** (no predefined output).
* The model tries to find **hidden patterns, structures, or groupings** in the data.
* **Dimensionality reduction :** Suppose you have a very large dataset with hundreds offeatures. Unsupervised learning can reduce it into fewer, more meaningful features without losing much information.

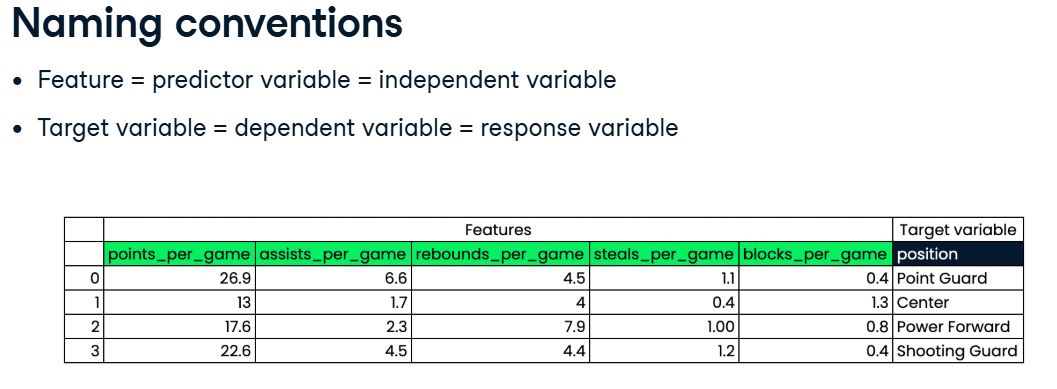
**Examples:**

* **Customer Segmentation**: An e-commerce site groups customers into clusters (e.g., budget buyers vs. premium buyers) without knowing labels beforehand.
* **Market Basket Analysis**: Discovering which products are often bought together (e.g., chips & cold drink).
* **Image Compression**: Grouping similar pixels to reduce image size.

✅ **Summary Table:**

| **Feature** |  | **Supervised Learning** | **Unsupervised Learning** |
| --- | --- | --- | --- |
| Data type |  | Labeled (with outputs) | Unlabeled (no outputs) |
| Goal |  | Predict outcomes | Find hidden patterns |
| Examples |  | House price prediction, Spam detection | Customer segmentation, Market basket analysis |

Here , we are just gonna learn about Supervised ML.



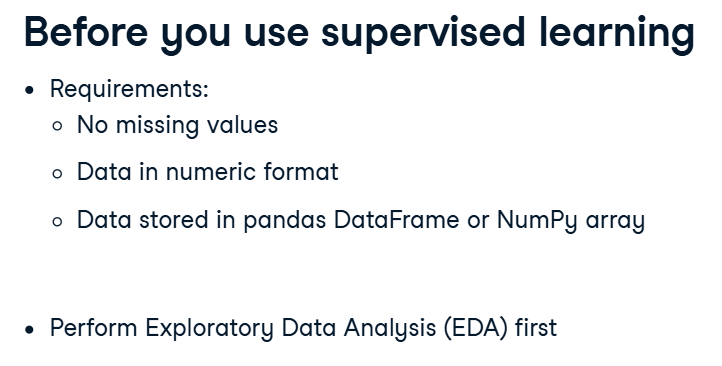
**Types of Supervised learning:**

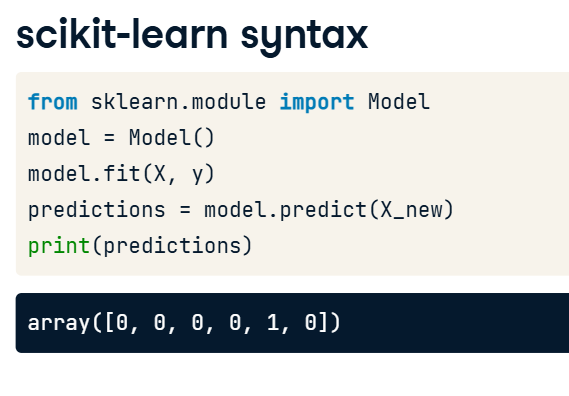
**1. Classification**

* **Definition**: Predicts a **category/class label** from input data.
* **Output**: Discrete values (Yes/No, Spam/Not Spam, Disease A/Disease B).
* **Examples**:
  + Email → Spam or Not Spam
  + Exam score → Pass or Fail
  + Patient → Has cancer or Not

**🔹 2. Regression**

* **Definition**: Predicts a **continuous numeric value** from input data.
* **Output**: Real numbers (price, temperature, weight).
* **Examples**:
  + Predicting **house price** from size & location
  + Forecasting **temperature** for tomorrow
  + Estimating a **person’s height** from age



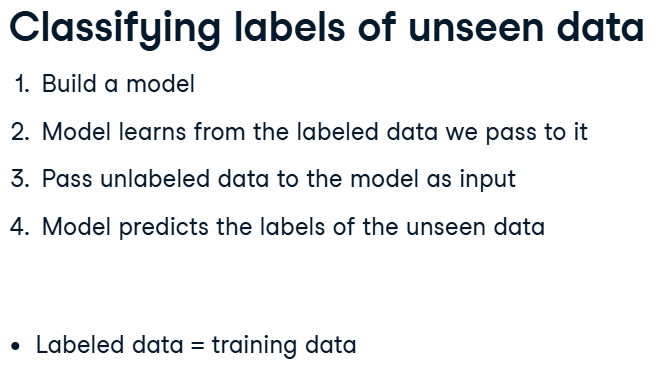


* ‘Model’ is just a placeholder name here — in real code, it would be something like:

LogisticRegression() , DecisionTreeClassifier() , KMeans()

* model.fit(X, y) → trains the model using training data (X(input) = email features, y(output) = labels: spam=1, not spam=0 ; this is binary classification as it has two labels).
* model.predict(X\_new) → predicts whether new emails are spam or not. ‘X\_new’ is the list of new inputs for which the output is to be predicted.
* **Output**: array([0, 0, 0, 0, 1, 0])
  + Means: the 5th email is **spam (1)**, and all others are **not spam (0)**.

👉 In short: the ML model is classifying emails into **spam (1)** or **not spam (0)**.



* Each column is a **feature** and each row is an **observation**
* Features (X) = Inputs
* Label/Target (y) = Outputs

k-Nearest Neighbours(KNN)

Here we predict the label of a data point by looking at the ‘k’(can be 3,4,5,6…..) closest datapoints. We take the majority vote.

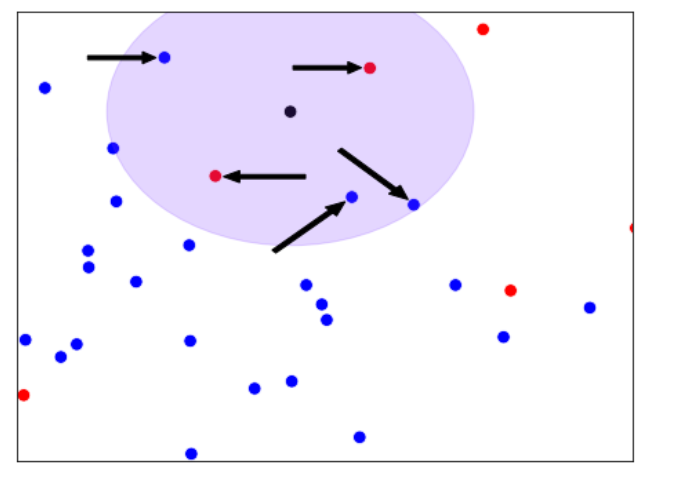
Do a **majority vote**:

* + If most neighbors are "Spam," classify as Spam.
  + If most are "Not Spam," classify as Not Spam.

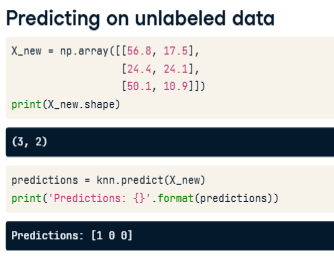
**✅ Example**

Suppose K = 3 and you want to classify an email.

* Among the 3 nearest neighbors: 2 are spam(red) (1), 1 is not spam(blue) (0).
* Result → Spam (1).



Black is labelled as red. Black is labelled as blue. For k=5.

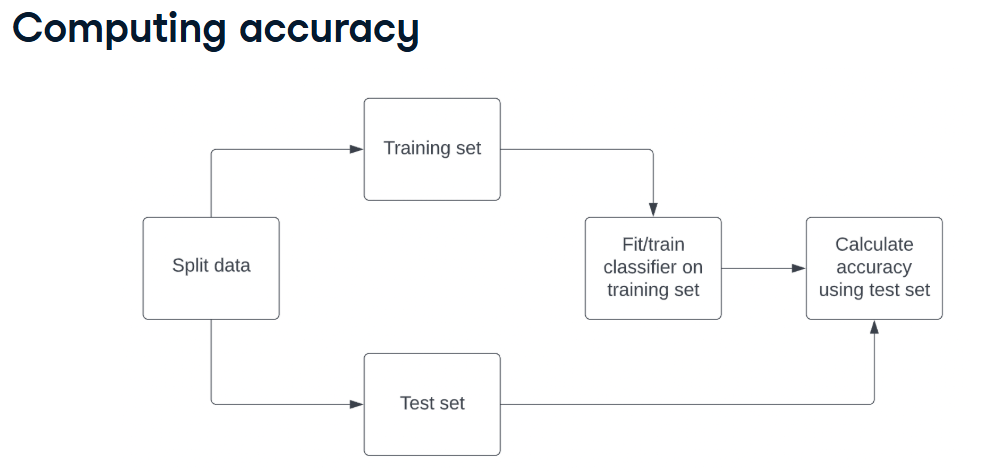


Here, ‘.values’ converts it into numpy array.

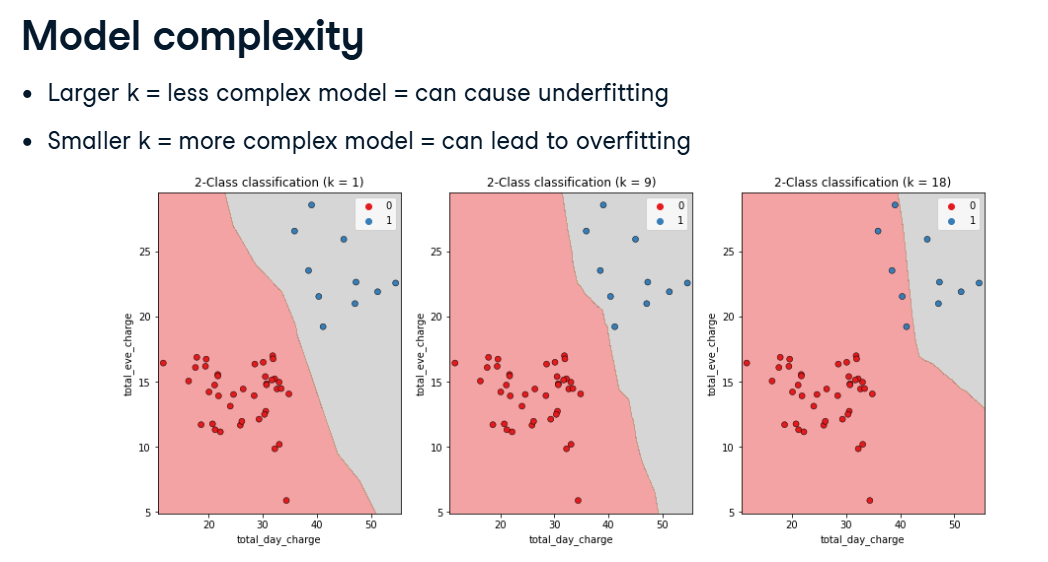
**Measuring Model performance**

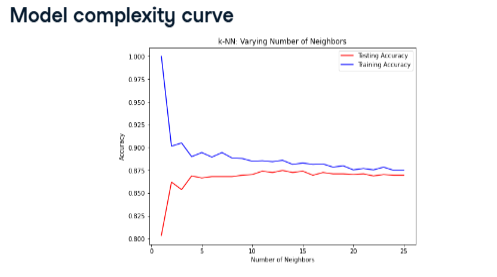
In classification, accuracy is a commonly used metric.

**Accuracy** = (correct predictions) / (total observation)









**🔹 Why train accuracy = 100% when K=1**

* In **KNN with k=1**, each training point’s nearest neighbor is **itself**.
* So during training, every point is classified correctly → **100% accuracy on training data**.

**🔹 Why test accuracy < 100% (≈80%) when K=1**

* On **test data**, the model looks for the nearest training point.
* But test points are **not exactly the same** as training points (since you did an 80-20 split).
* If a test point falls near a training point of the wrong class (due to overlap/noise in data), it gets misclassified.
* This is why test accuracy drops (around 80%).

**✅ Key Concept: Overfitting when k=1**

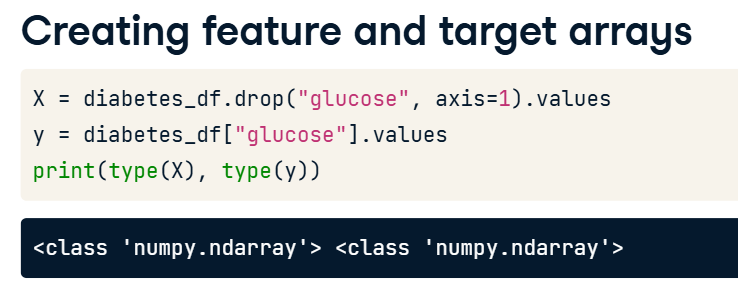
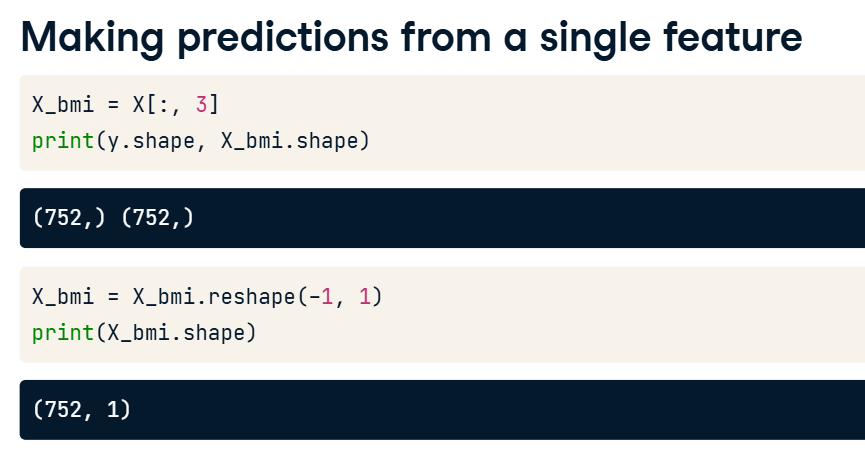
* K=1 → The model “memorizes” training data (perfect accuracy).
* But memorization does not generalize well → poor test performance.
* As you increase K (e.g., 3, 5, 7), the model generalizes better, and test accuracy usually improves/stabilizes.

**Regression**

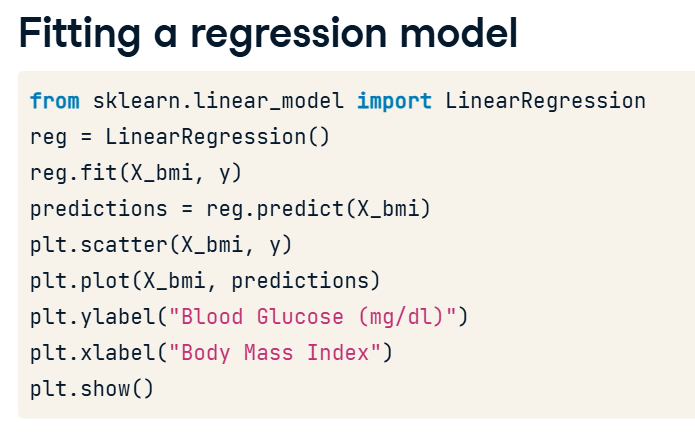
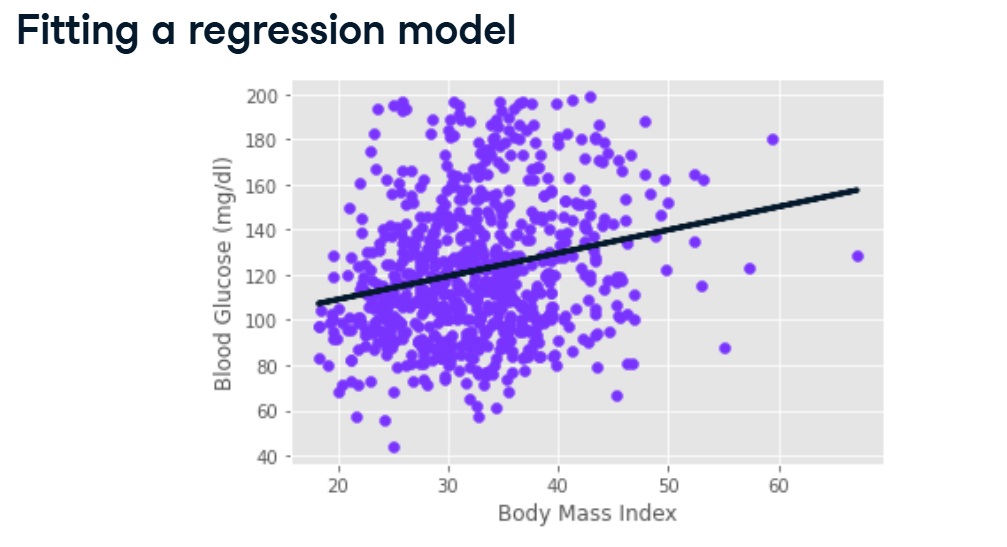
Target variable has continuous values.

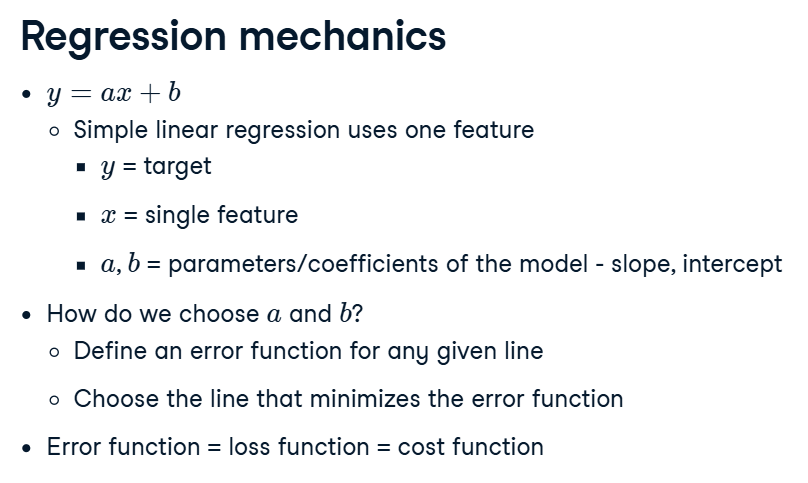
**Prediction using linear regression**

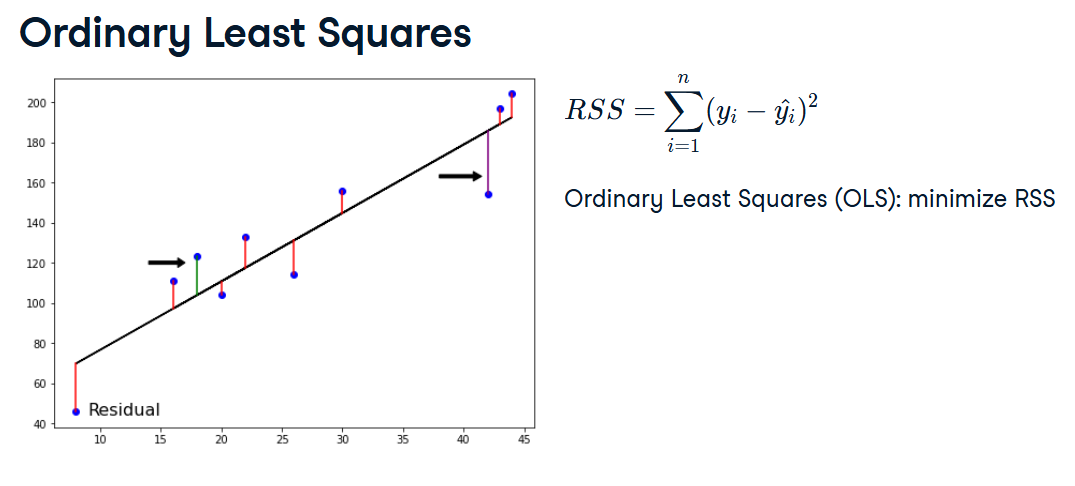
We use a linear line.

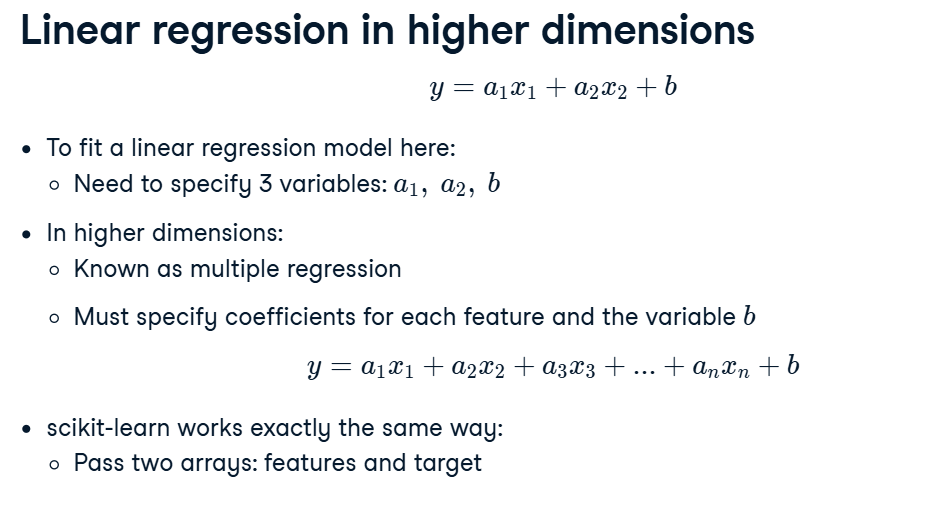


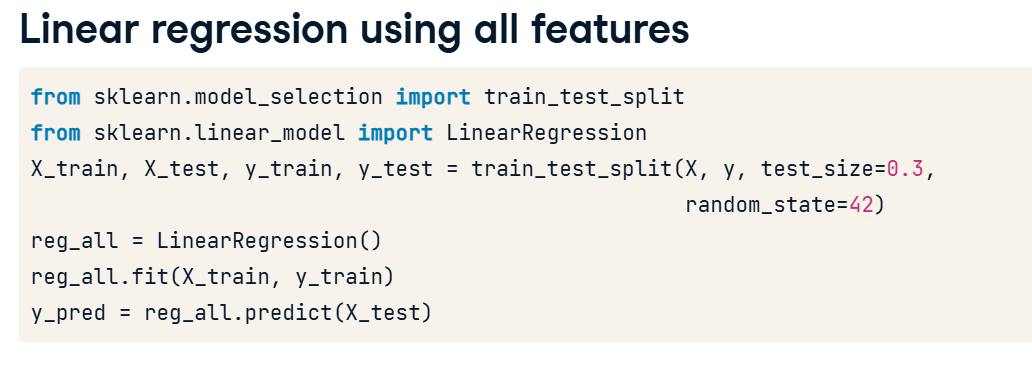
We have to make features a 2d array to be accepted by scikit learn.

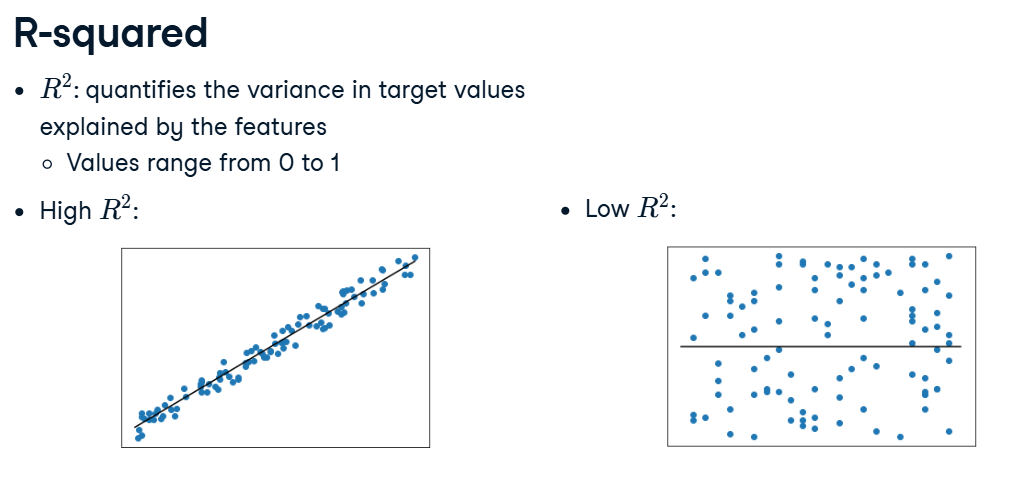
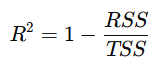






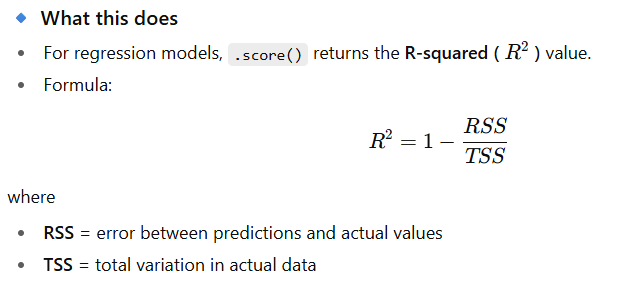


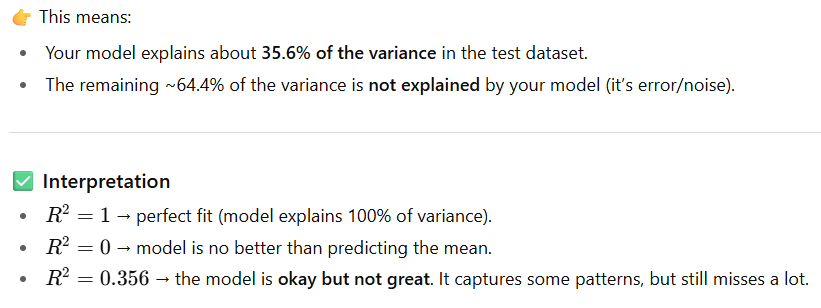


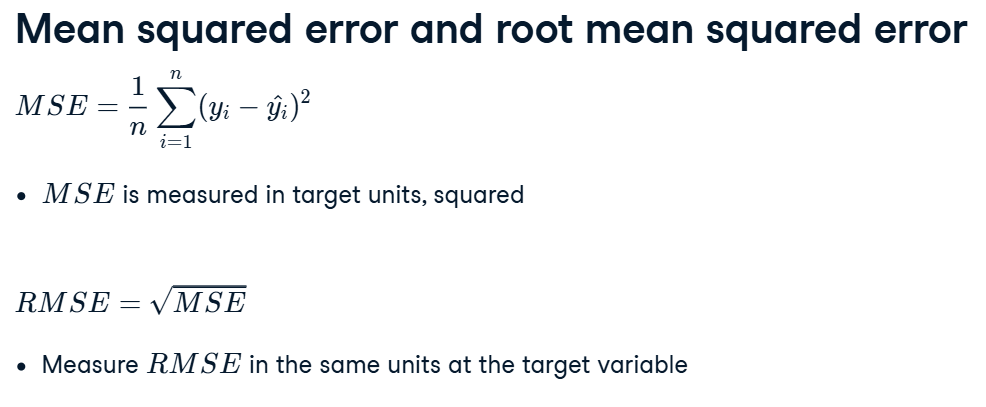


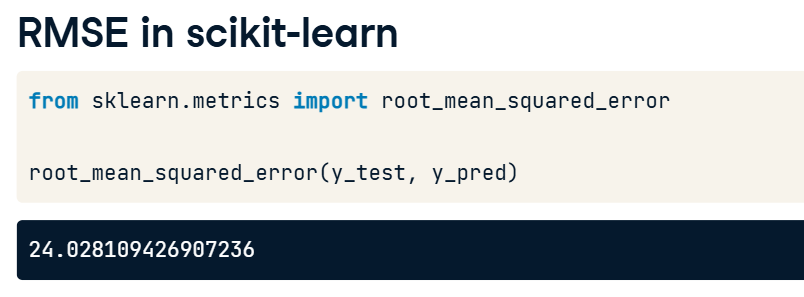
* 🡪 Perfect Prediction
* 🡪 Bad Prediction

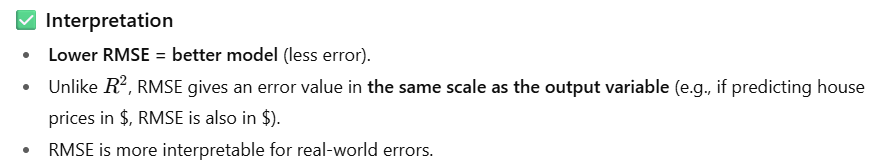






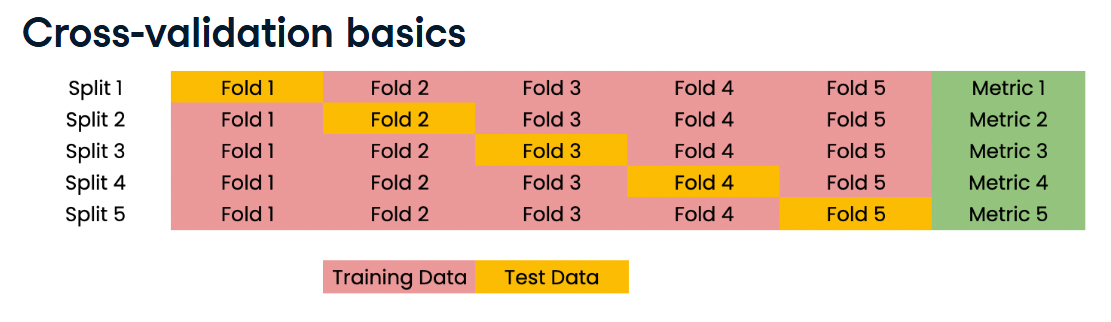


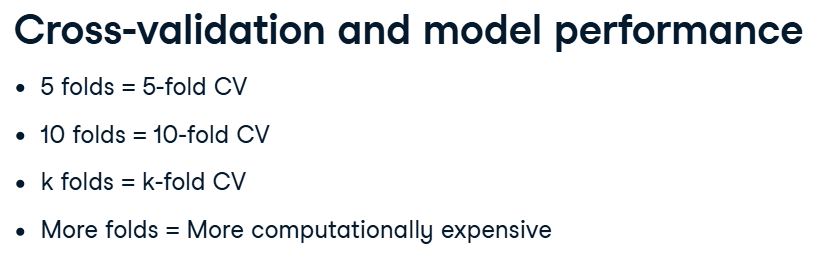


 This means, on average, your model’s predictions are **about 24 units away** from the actual values.

**CROSS-VALIDATION**

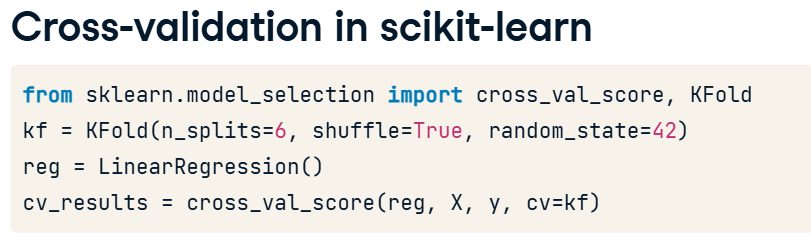
* When we train a machine learning model, we usually **split data into training and testing sets** (e.g., 80–20 split).
* Model may perform well on one split but poorly on another.
* A single train-test split gives only **one accuracy score**, which may not represent real performance.
* Cross-validation solves these issues by **splitting data into multiple folds (parts)** and training/testing multiple times.
* Each fold takes turns being the test set, while the rest are training sets.
* Finally, the average score or median or any statistics across all folds is taken.

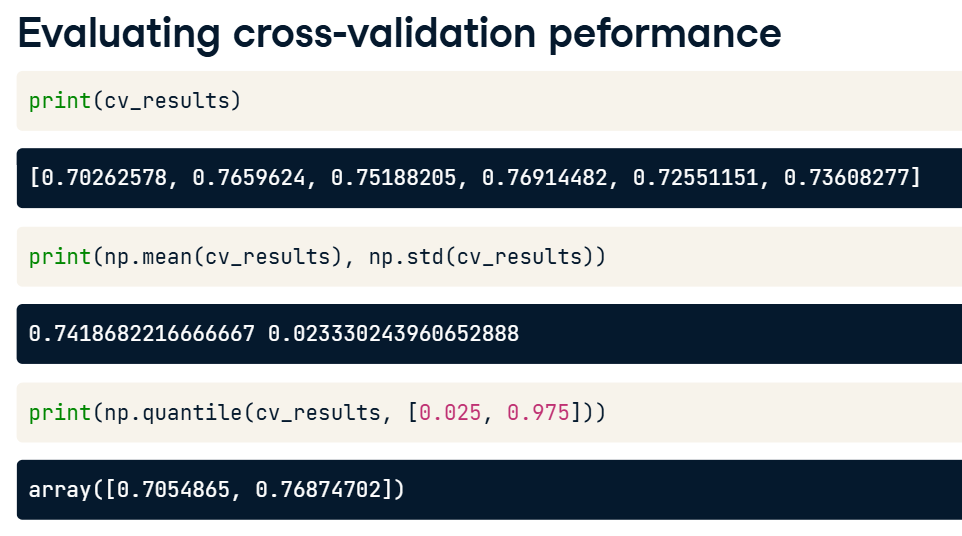




**✅ Benefits**

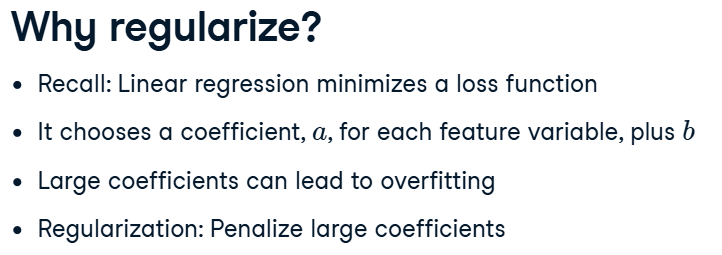
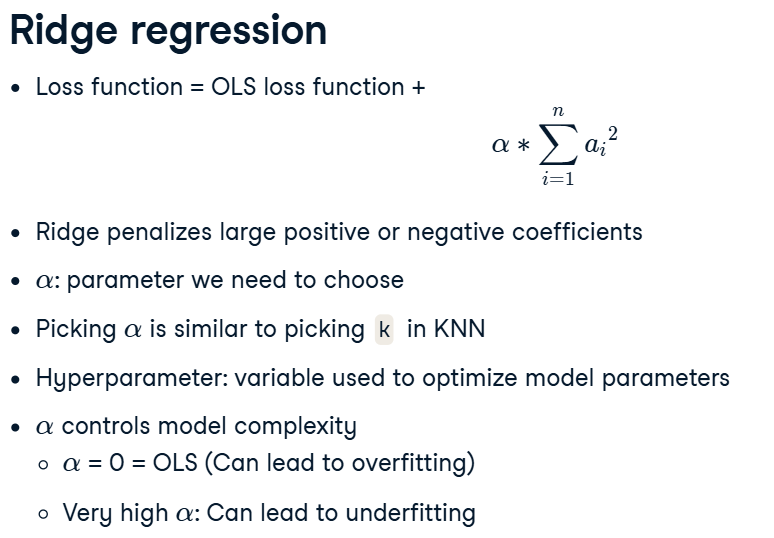
* Uses the **entire dataset** for both training and testing.
* Gives a **more reliable estimate** of model performance.
* Helps in **model selection and hyperparameter tuning** (choosing the best algorithm or K in KNN, etc.).

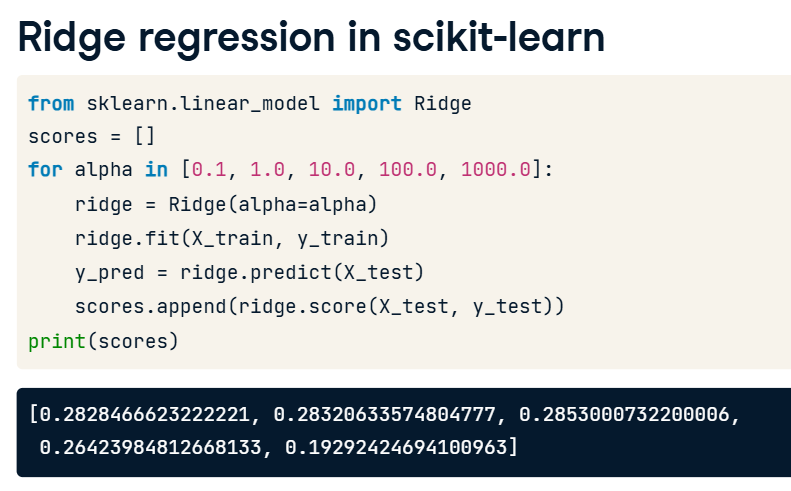


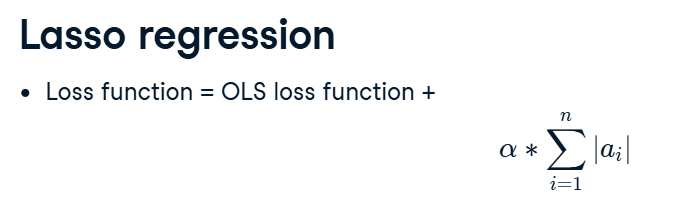
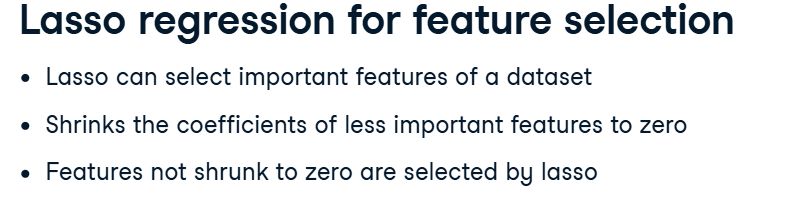


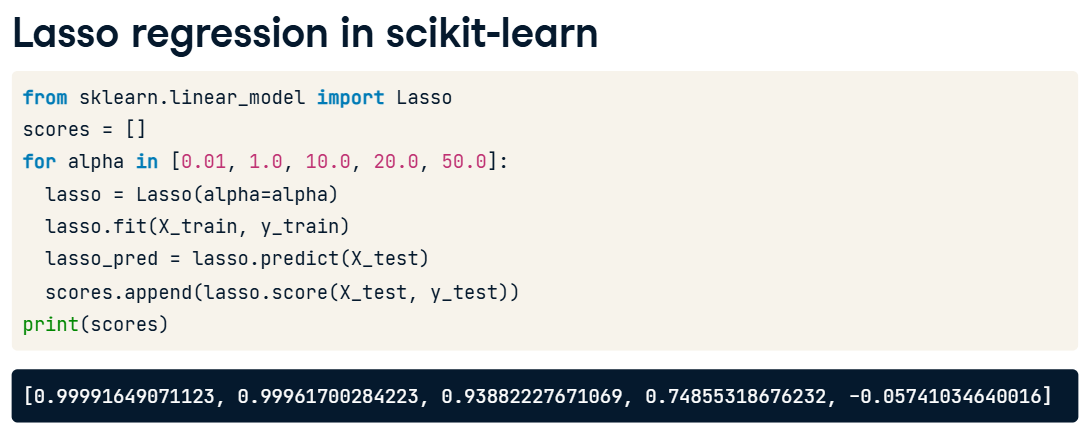
In the last one, we took 95% CI.

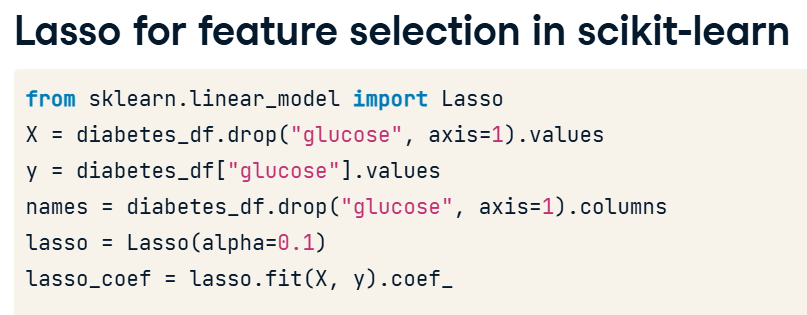
**Regularized Regression**

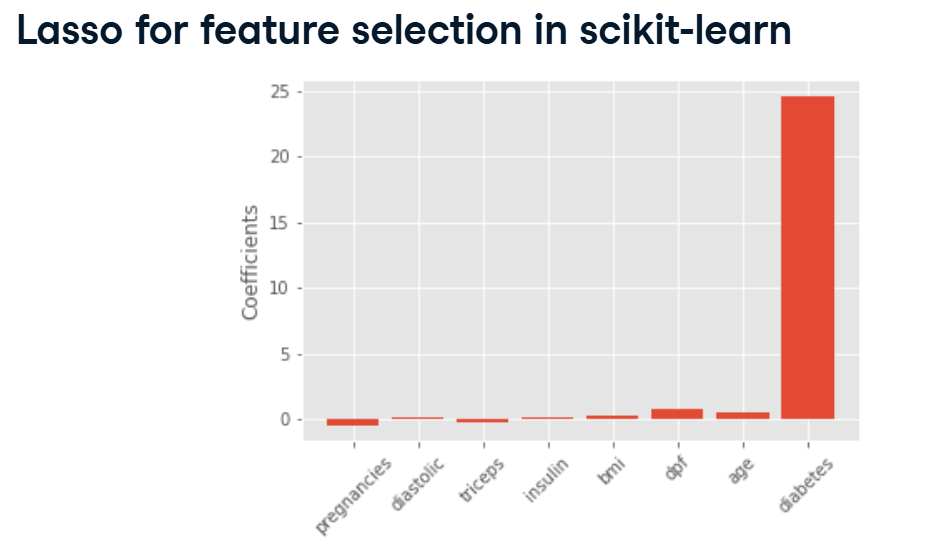
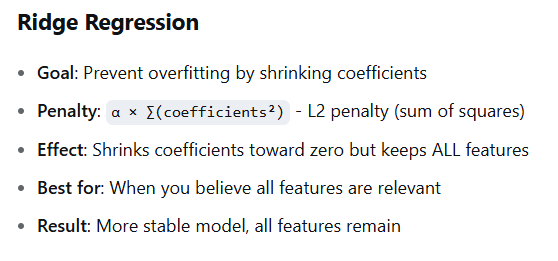
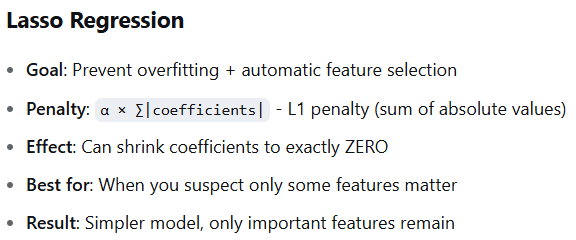


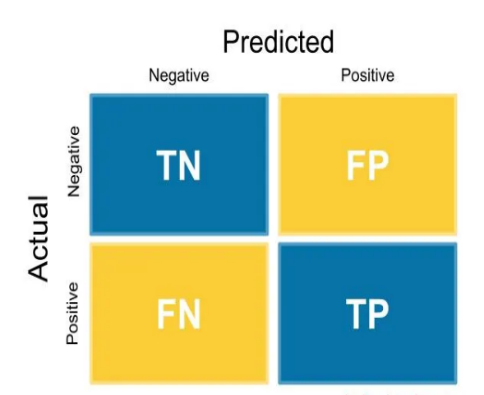


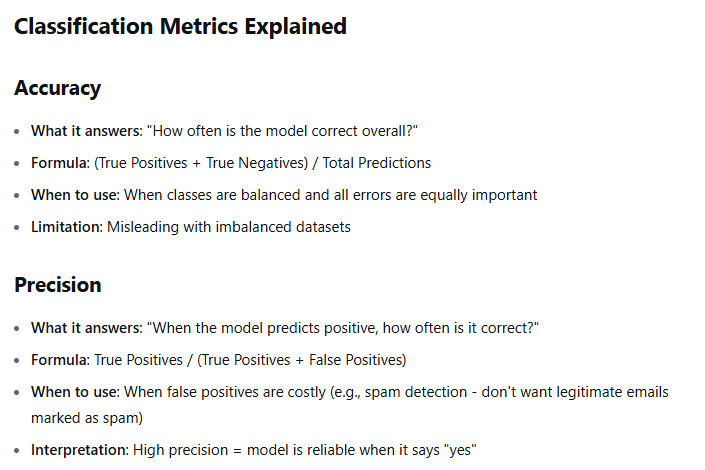


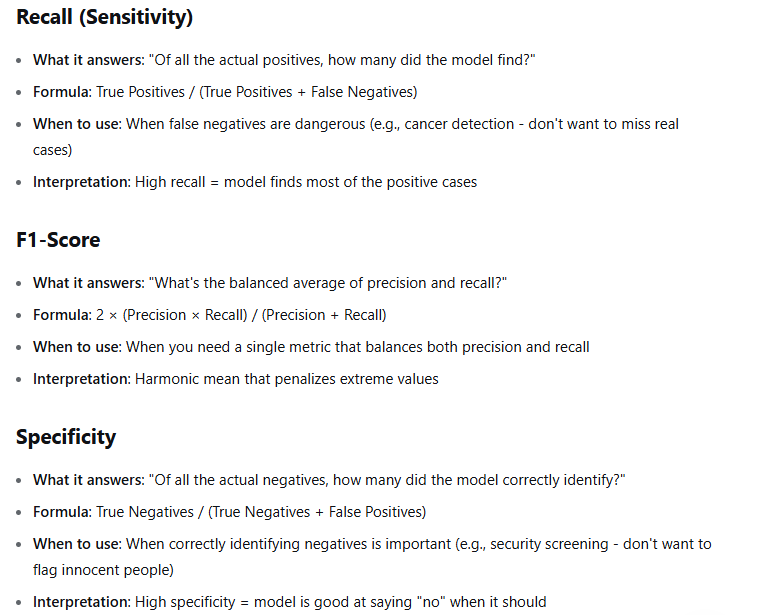


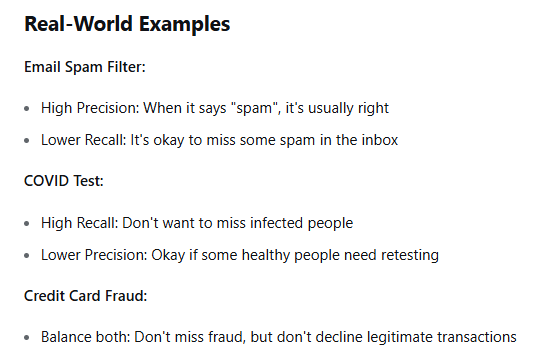
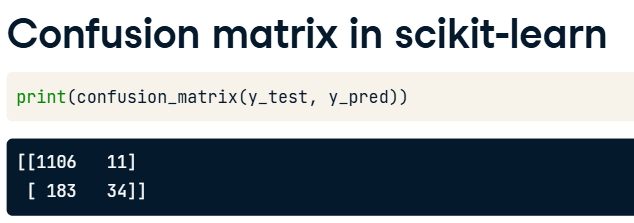


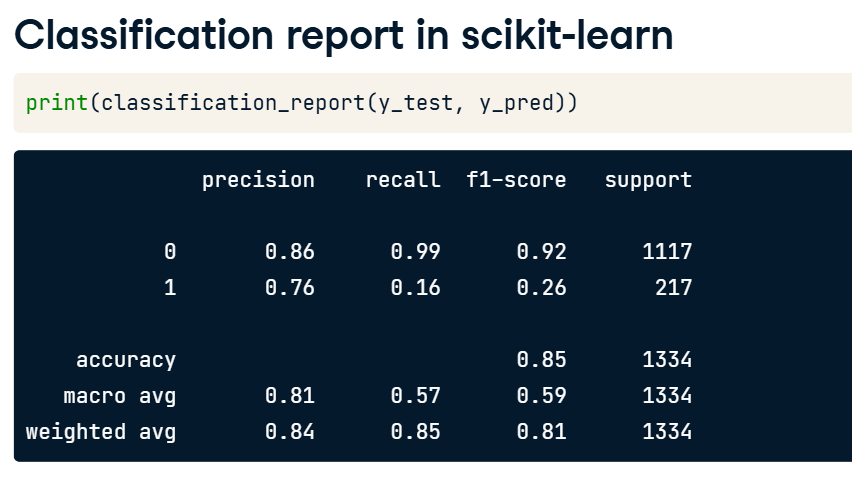


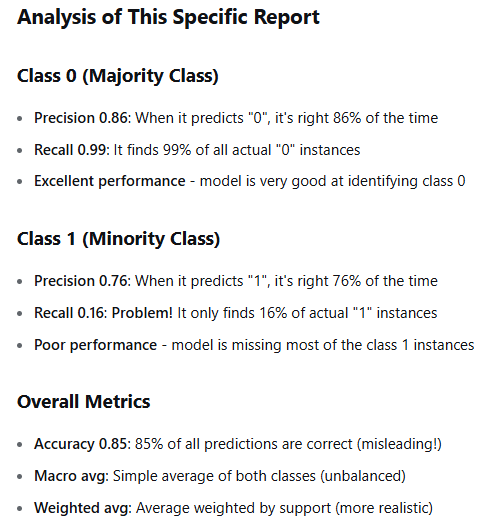
**Confusion Matrix**

****

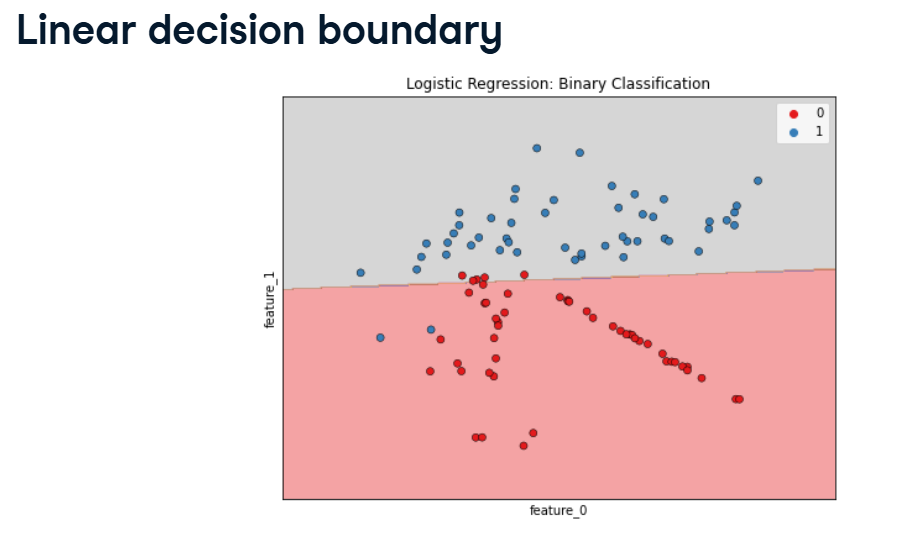
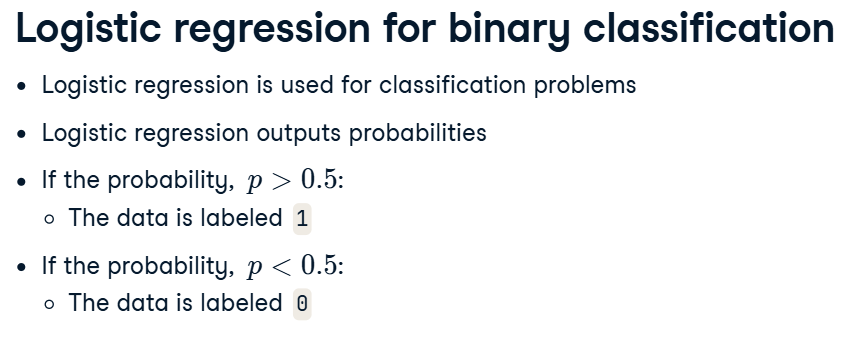
****

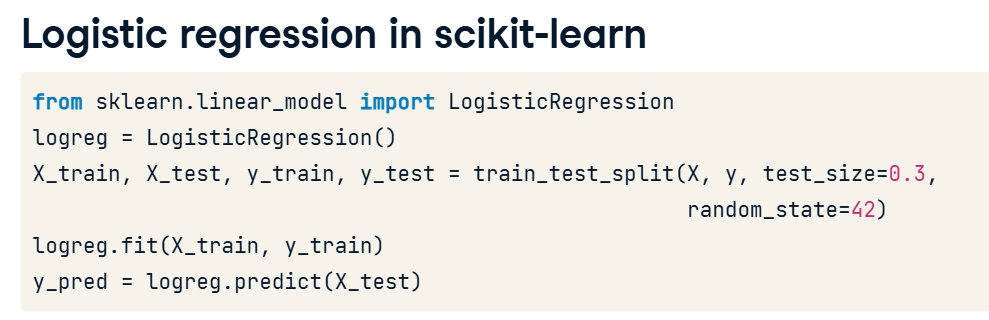
****

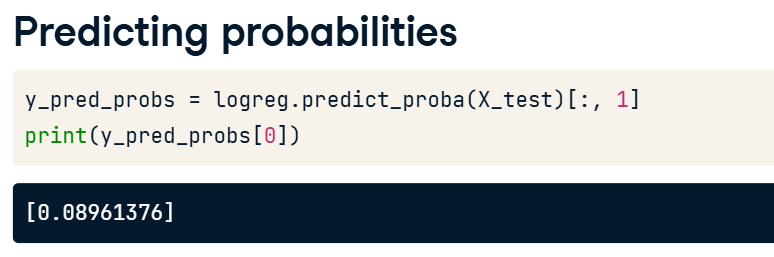
****

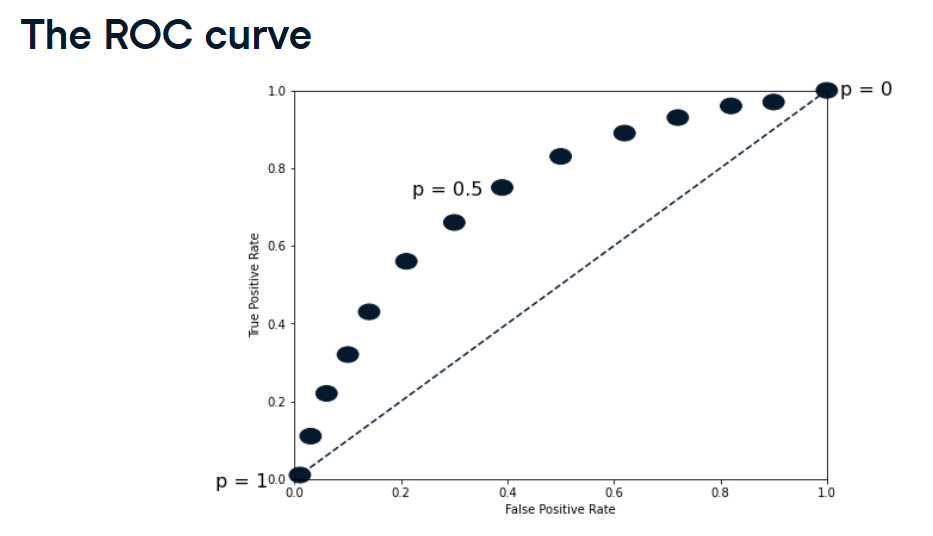
****

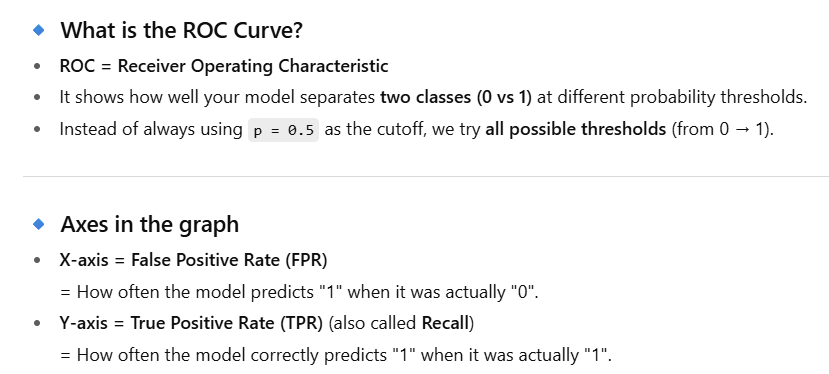
****

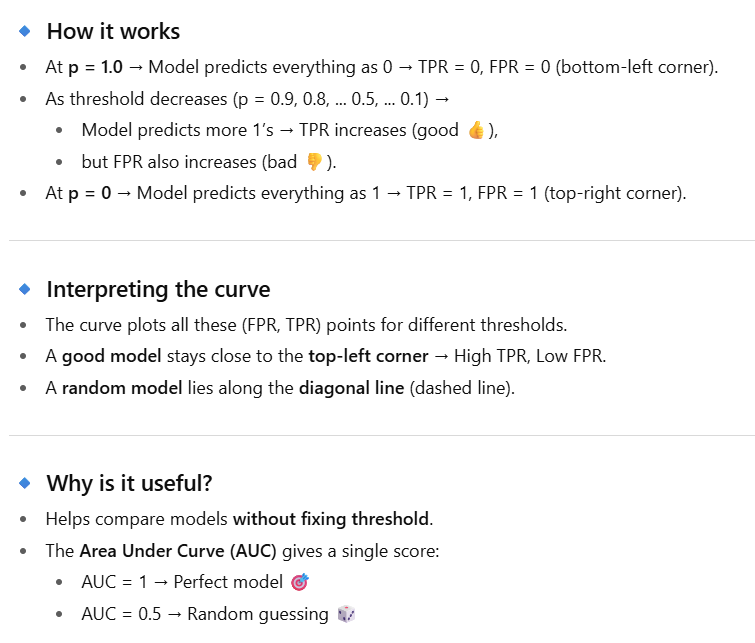
****

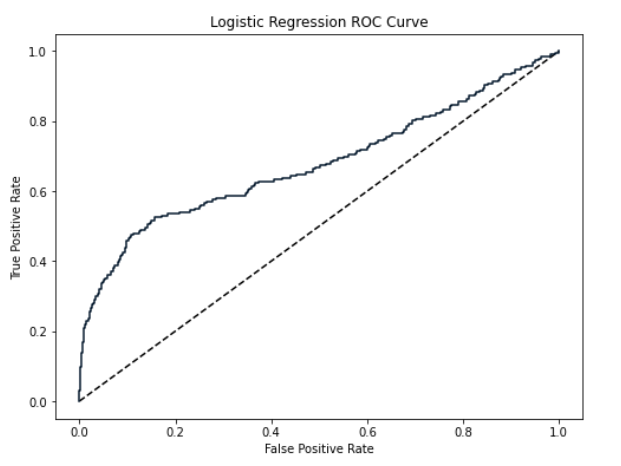
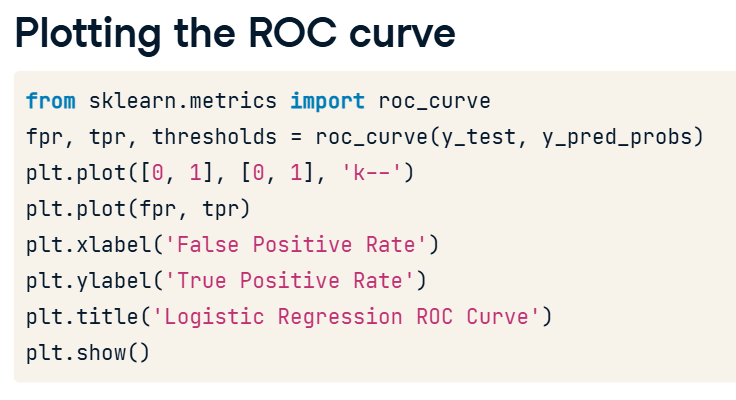
****

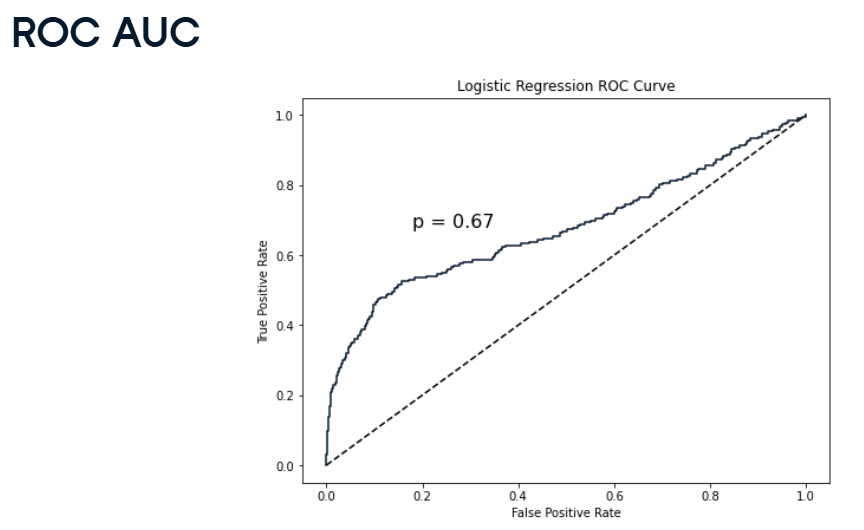
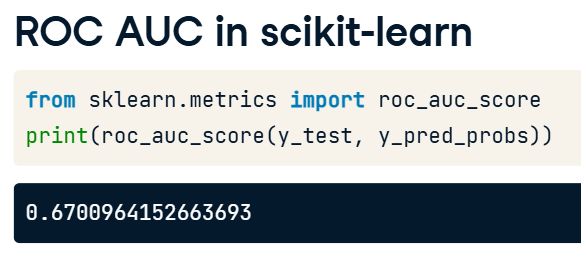
****

****

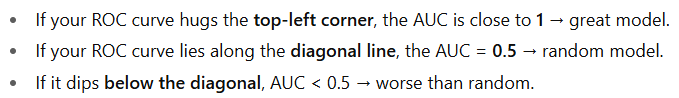
****

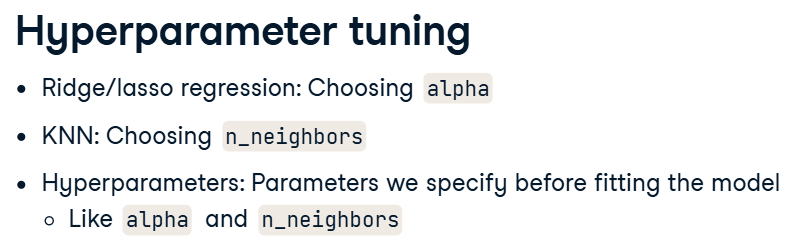
****

****

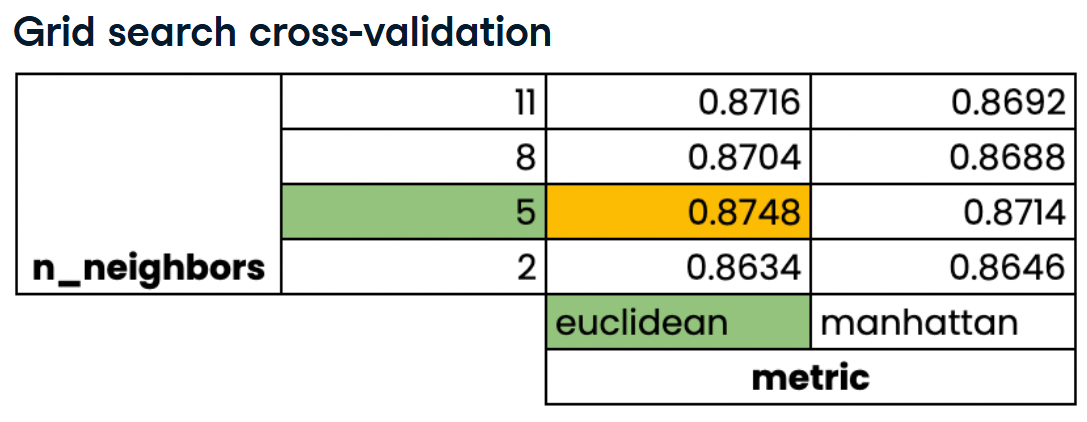
****

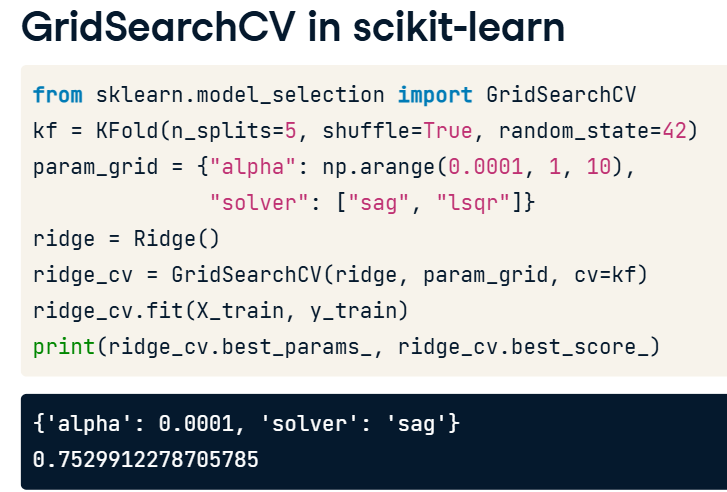
The AUC is the area under the ROC curve (above the diagonal baseline).

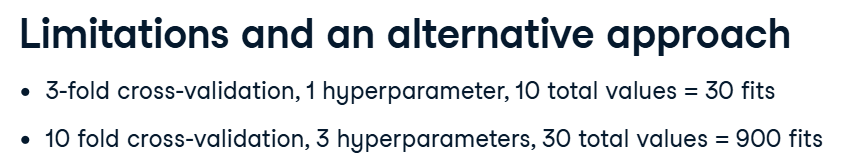


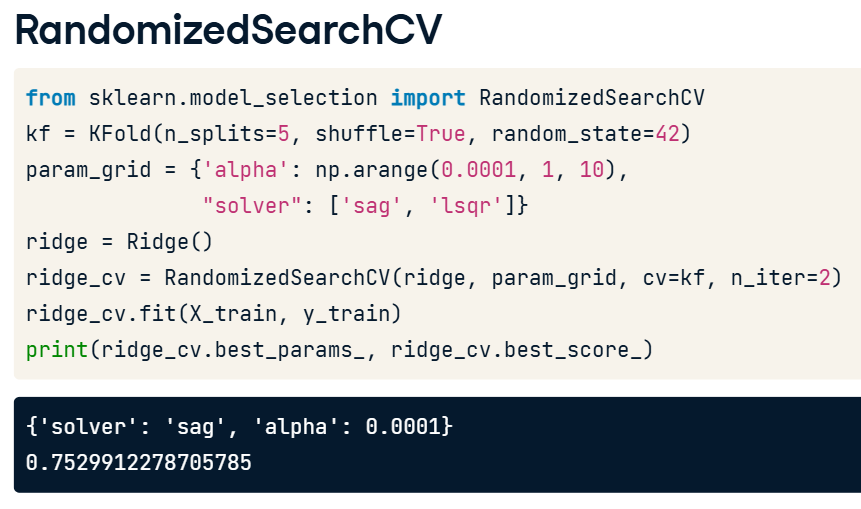


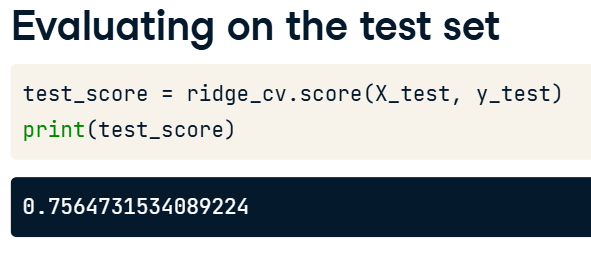




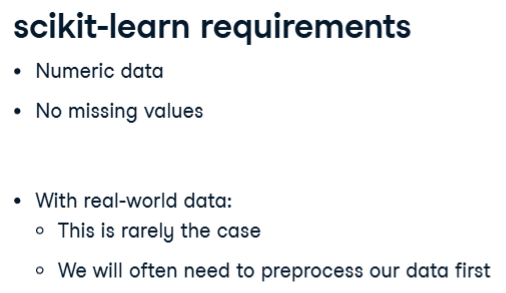




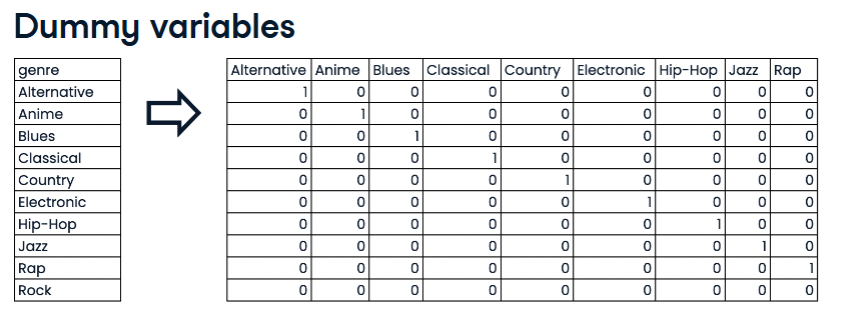




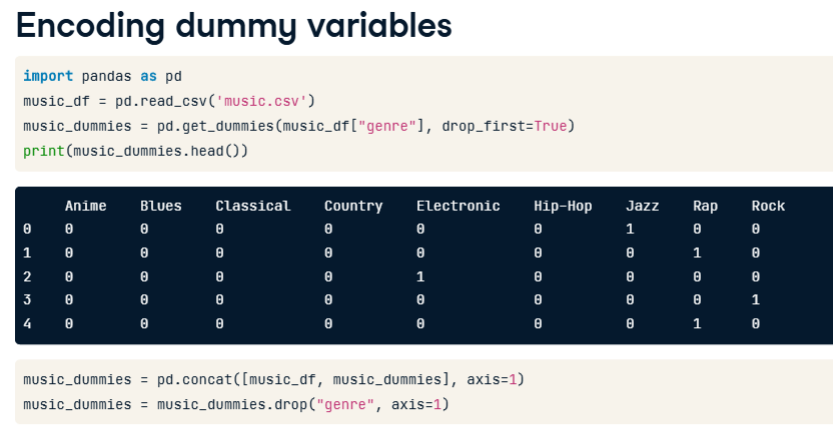
**Preprocessing Data**



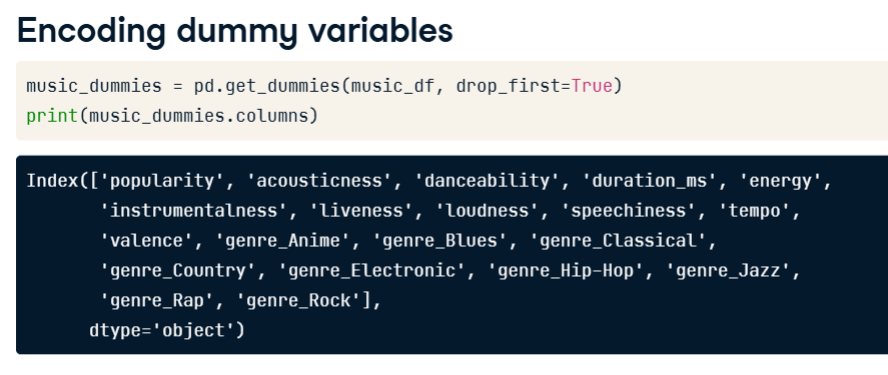


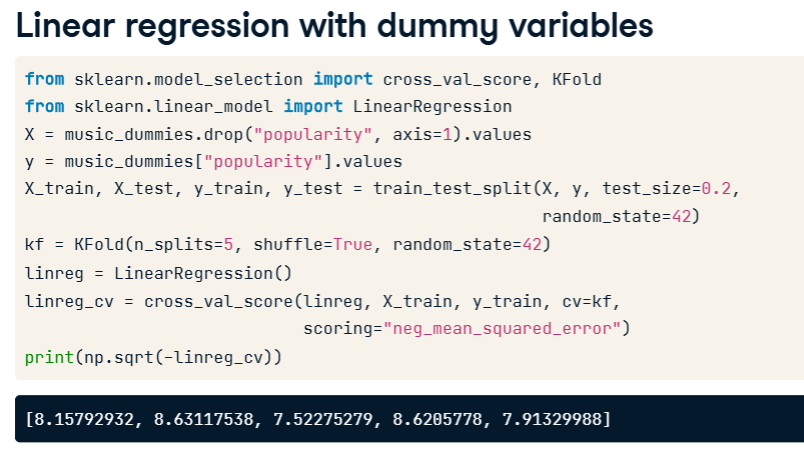


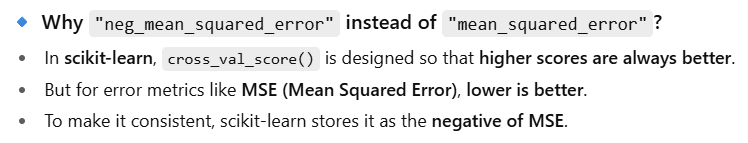
If none of these columns have ‘1’ then Rock is automatically selected.



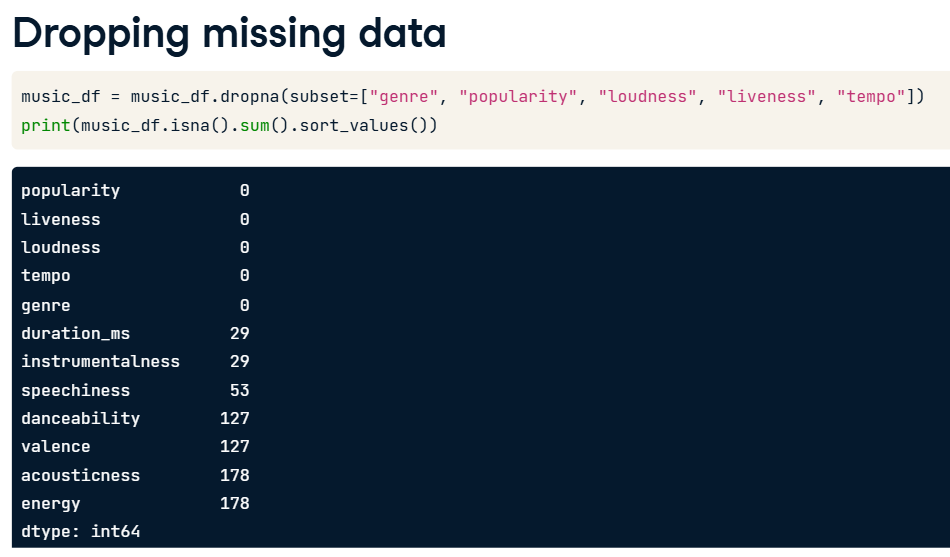
If the dataframe has only one categorical feature then we don’t need to specify the column name.

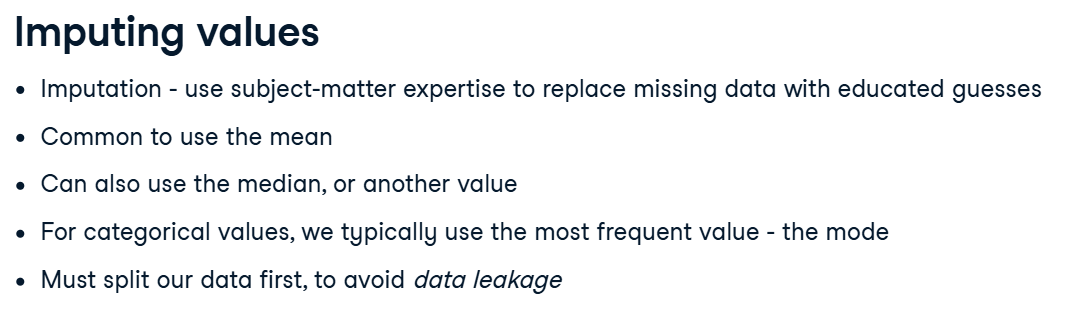


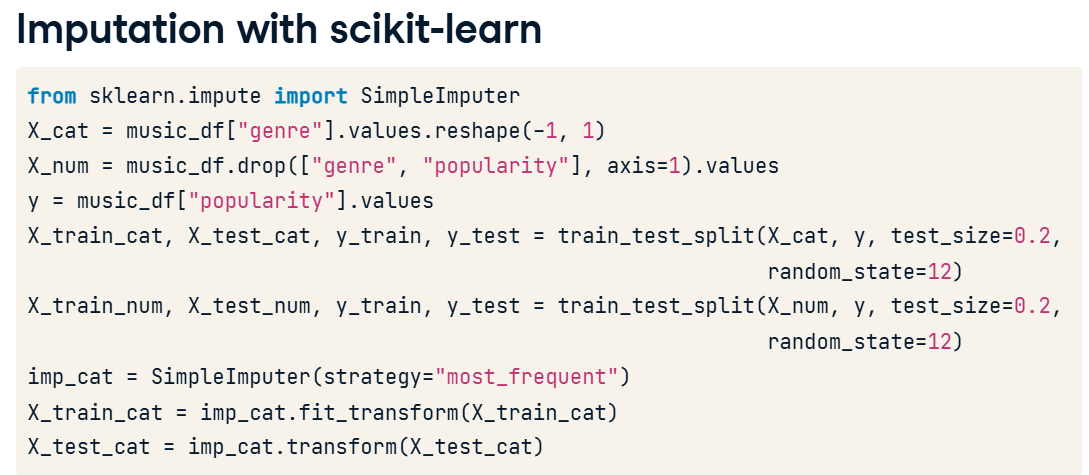


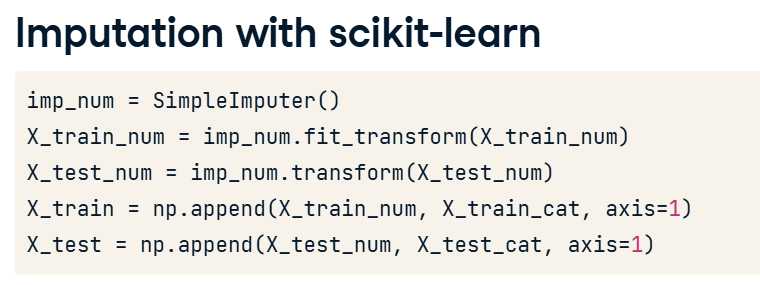


**Handling Missing Data**





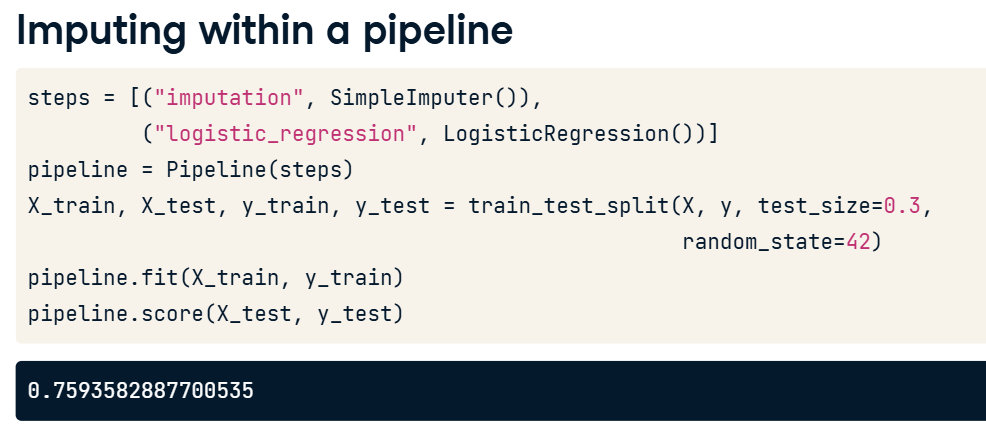
 .reshape(-1,1) changes the shape from 1D to 2D as scikit learn expects input features to be 2D arrays.



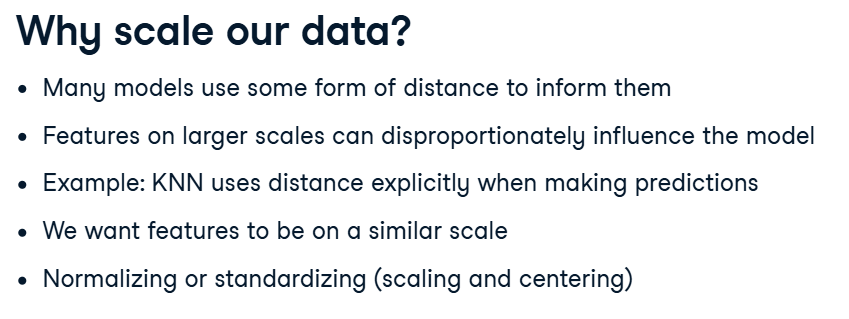
Due to their ability to transform our data, imputers are known as transformers.

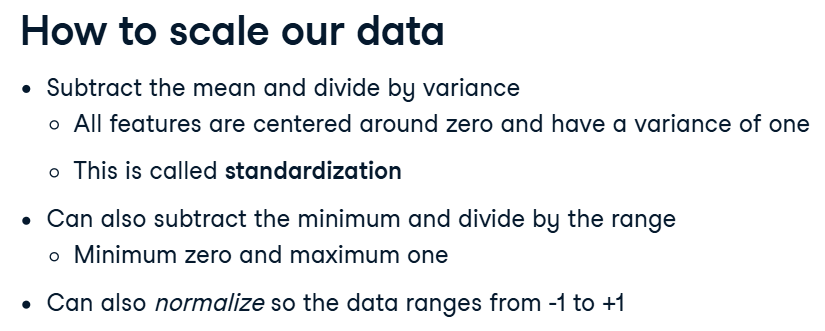
For missing categorical features , we use mode by default and for missing numeric features, we use mean by default in the above cases.

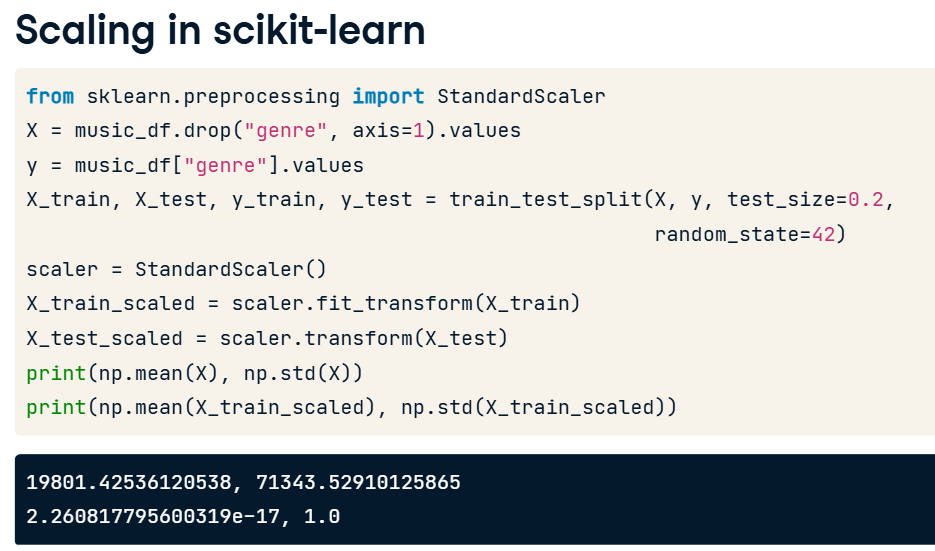


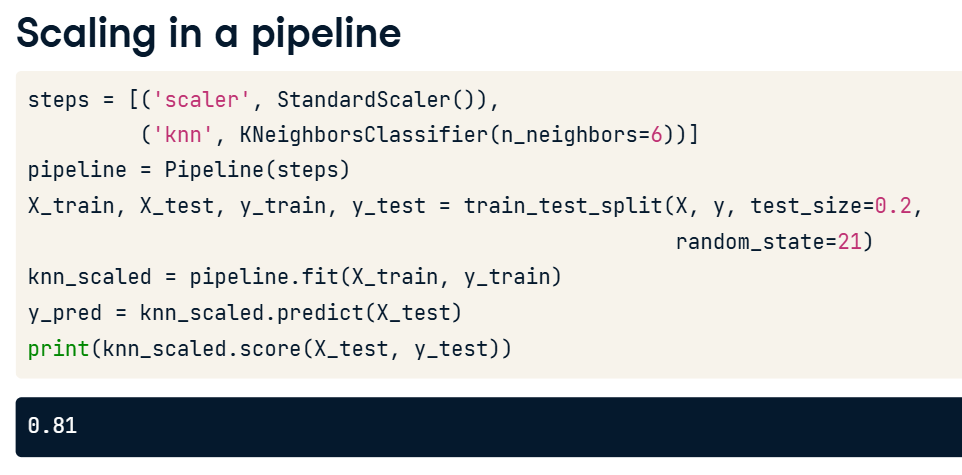


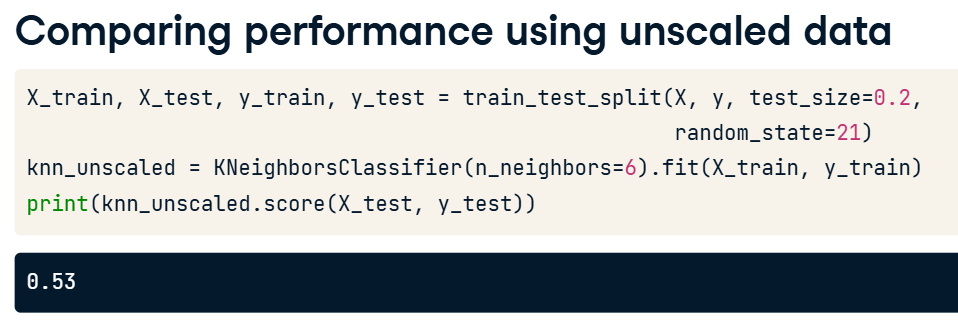
**Centering and Scaling**

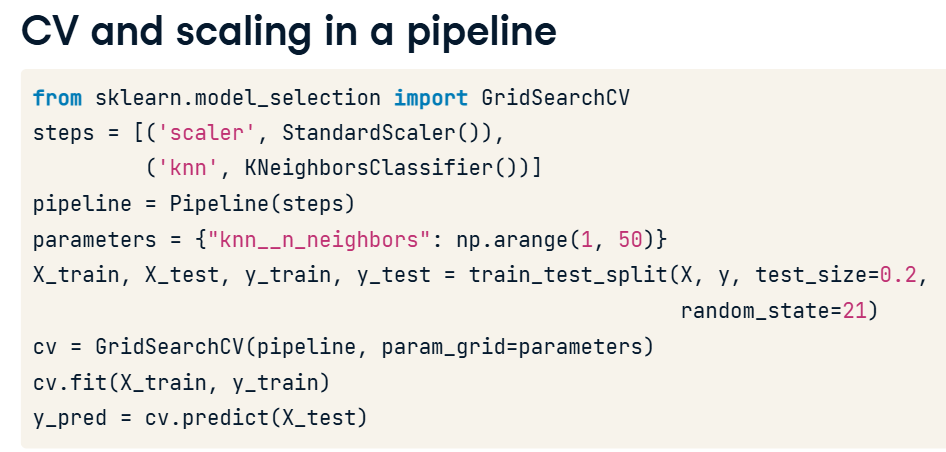


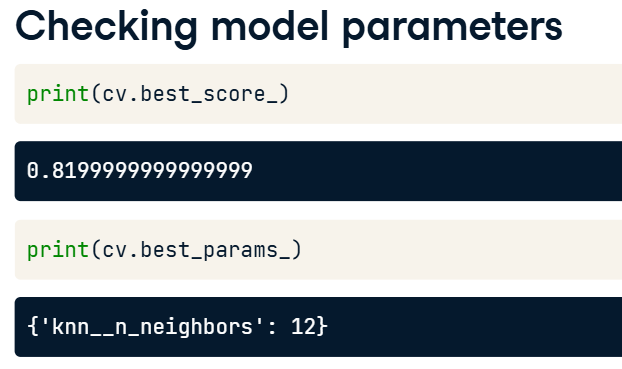




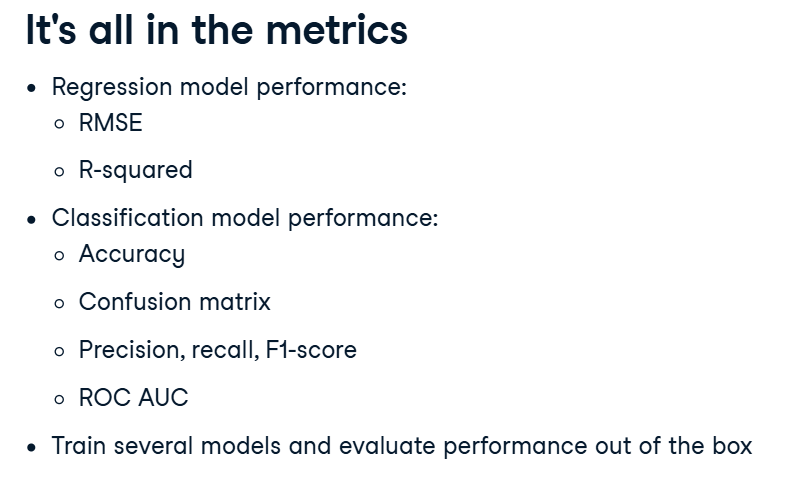


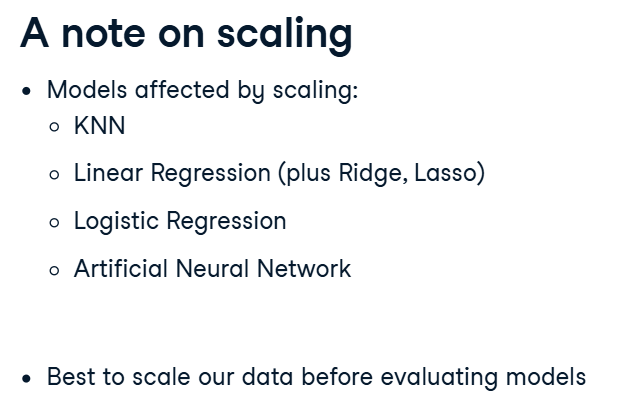


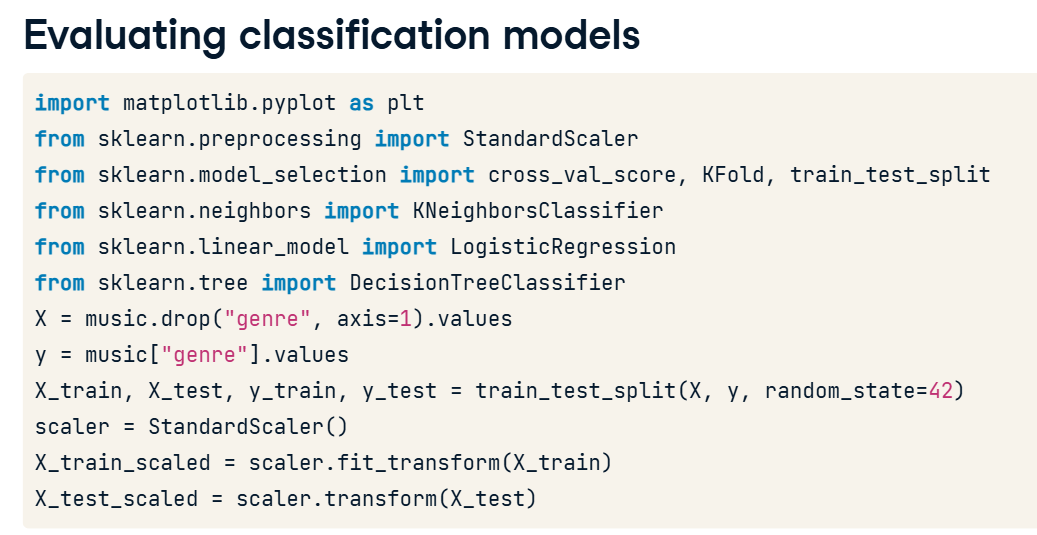


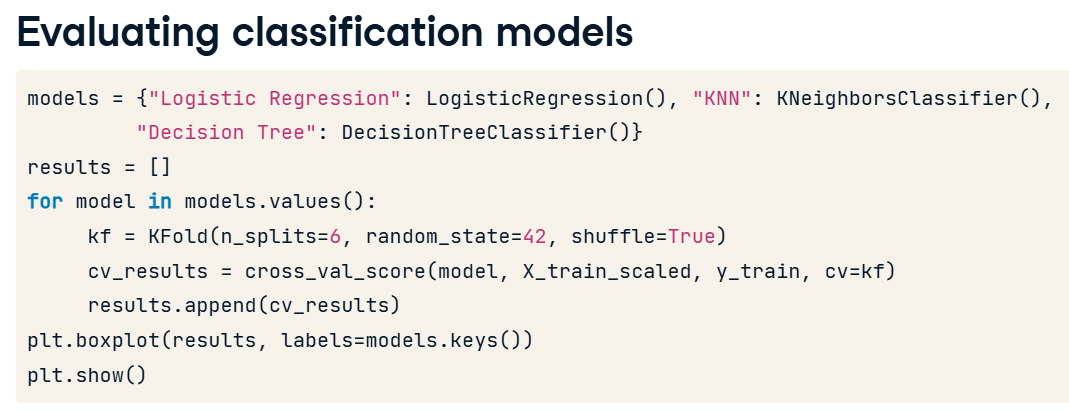


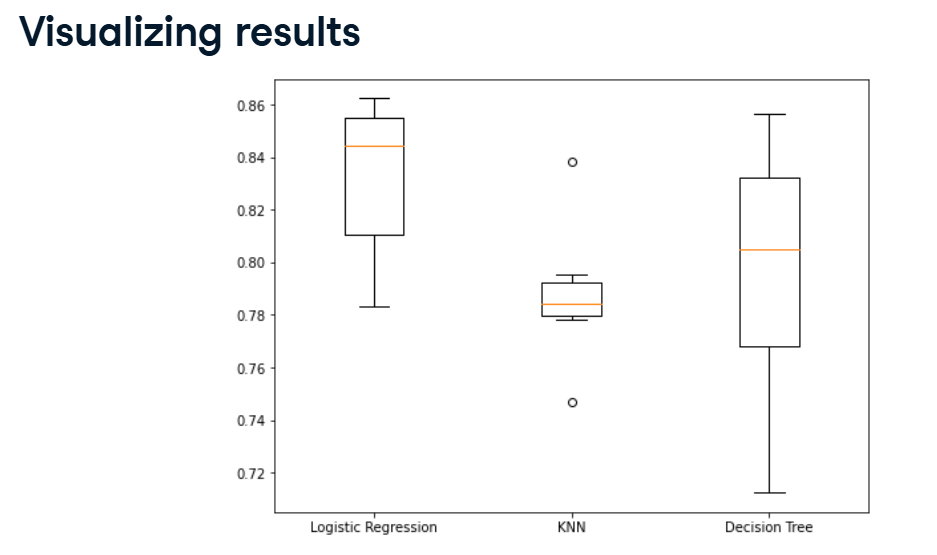


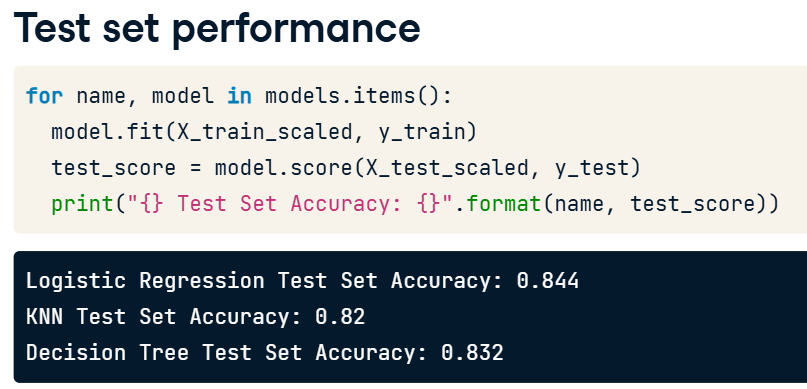




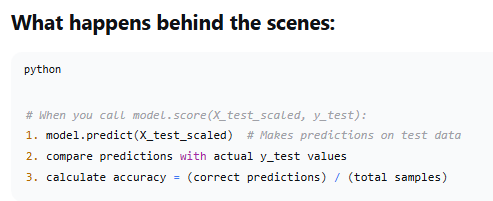


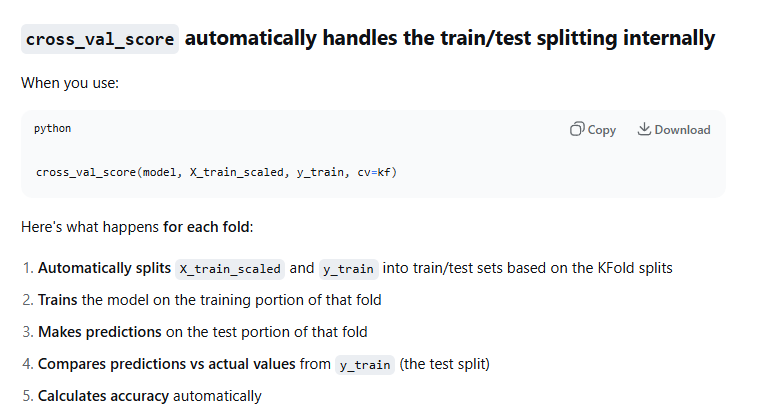




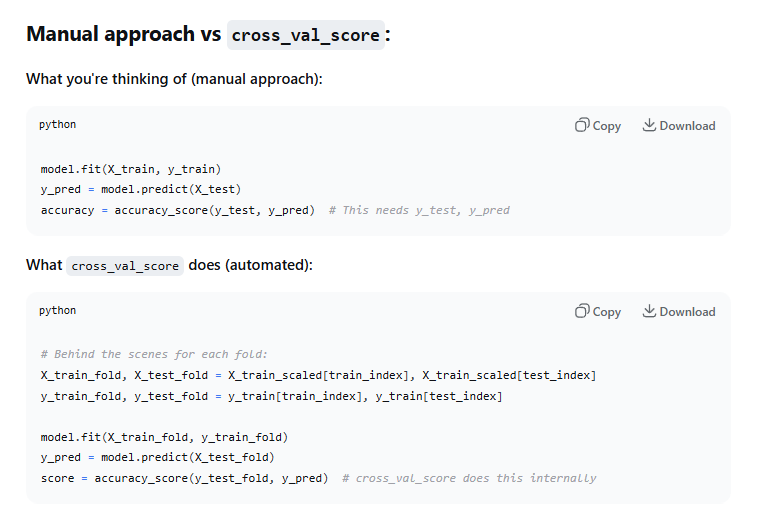


Logistic Regression performs best for this problem if we are using accuracy as a metrics.



COMMON CONFUSION SOLVED:  


X\_train\_scaled and y\_train are splitted again internally for evaluation of cross\_val\_score.



Accuracy for classification problems and for regression problems.