Machine Learning and Deep Learning
Lecture-02

By: Somnath Mazumdar
Assistant Professor
sma.digi@cbs.dk

Spring 2026

Outline

- Data pre-processing
- EDA
- Bias-variance
- Loss functions
- Machine Learning
- CV schemes

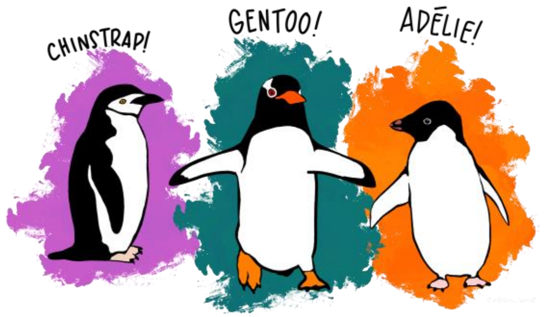
Data Pre-processing

Data Features

Labels

Features (also variables, attributes)

	species_short	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.100	18.700	181.000	3750.000	MALE
1	Adelie	Torgersen	39.500	17.400	186.000	3800.000	FEMALE
2	Adelie	Torgersen	40.300	18.000	195.000	3250.000	FEMALE
4	Adelie	Torgersen	36.700	19.300	193.000	3450.000	FEMALE
5	Adelie	Torgersen	39.300	20.600	190.000	3650.000	MALE

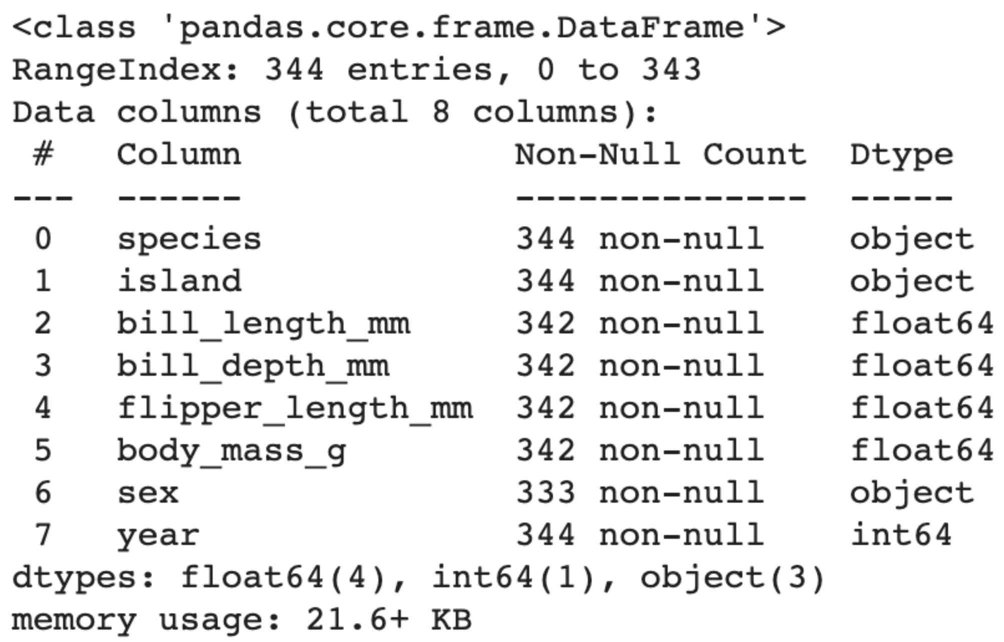


n: number of rows
m: number of features

https://github.com/allisonhorst/palmerpenguins

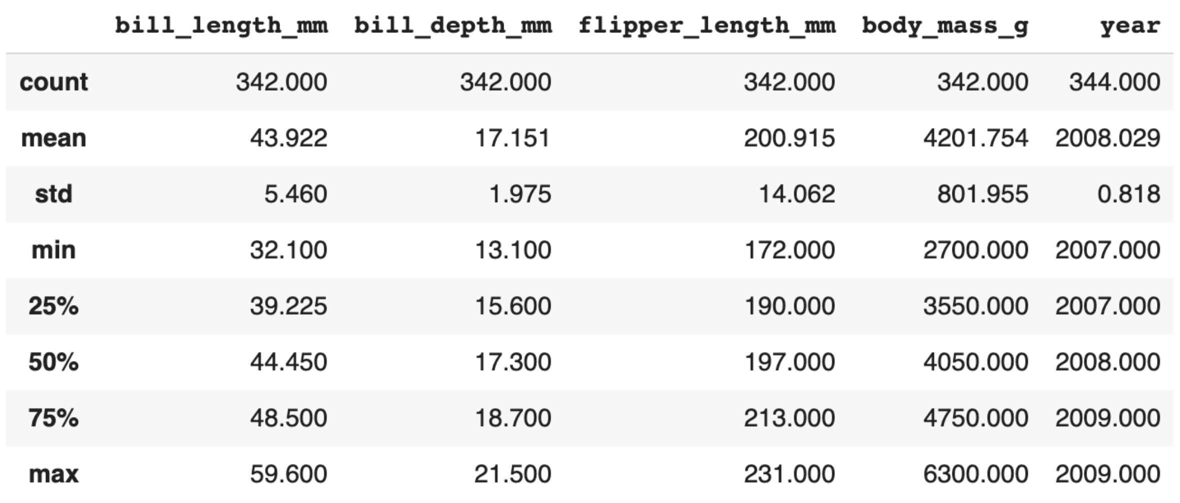
Data Types and Summary

df.info()



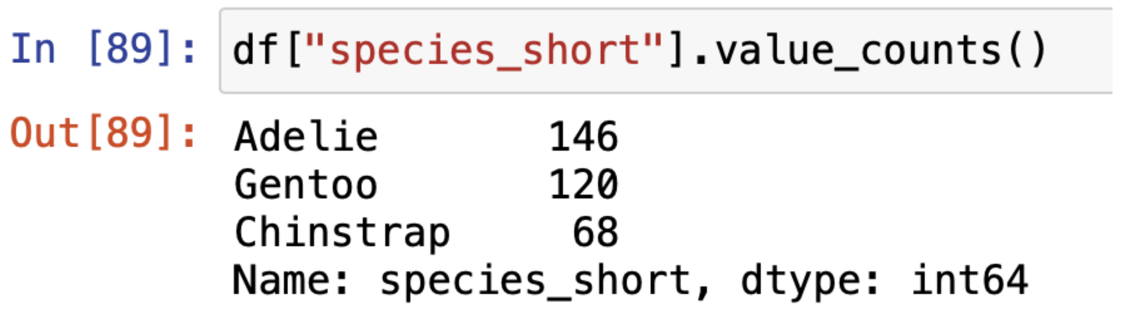
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object 
 7   year               344 non-null    int64  
dtypes: float64(4), int64(1), object(3)
memory usage: 21.6+ KB
```

df.describe()



	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	year
count	342.000	342.000	342.000	342.000	344.000
mean	43.922	17.151	200.915	4201.754	2008.029
std	5.460	1.975	14.062	801.955	0.818
min	32.100	13.100	172.000	2700.000	2007.000
25%	39.225	15.600	190.000	3550.000	2007.000
50%	44.450	17.300	197.000	4050.000	2008.000
75%	48.500	18.700	213.000	4750.000	2009.000
max	59.600	21.500	231.000	6300.000	2009.000

df[feature].value_counts()



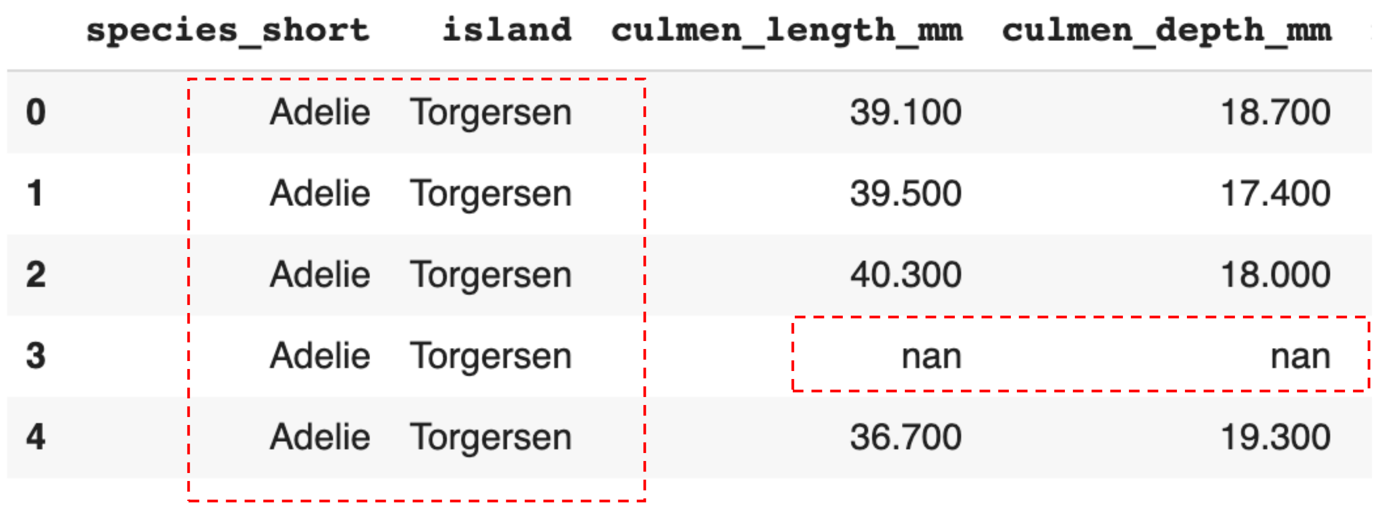
```
In [89]: df["species_short"].value_counts()

Out[89]: Adelie       146
         Gentoo       120
         Chinstrap     68
         Name: species_short, dtype: int64
```

Cleaning and Preprocessing Data

- Data requires some preprocessing and cleaning.

- Issues can be known in advance or noticed during exploration

- Common issues:
    - Missing values
    - Faulty data
    - Wrong data type
    - Duplicates



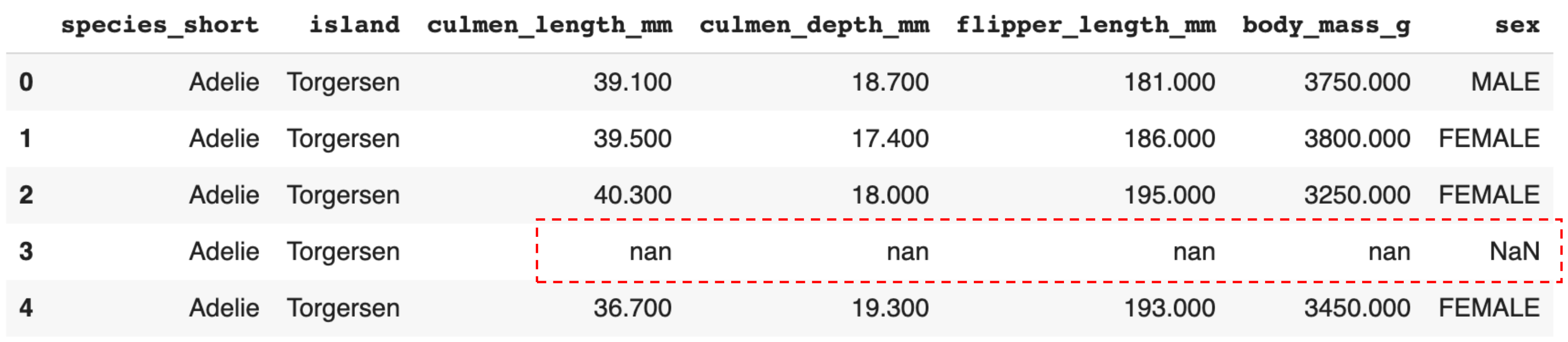
	species_short	island	culmen_length_mm	culmen_depth_mm
0	Adelie	Torgersen	39.100	18.700
1	Adelie	Torgersen	39.500	17.400
2	Adelie	Torgersen	40.300	18.000
3	Adelie	Torgersen	nan	nan
4	Adelie	Torgersen	36.700	19.300

Cleaning and Preprocessing Data

Some examples of ways to clean and preprocess data:

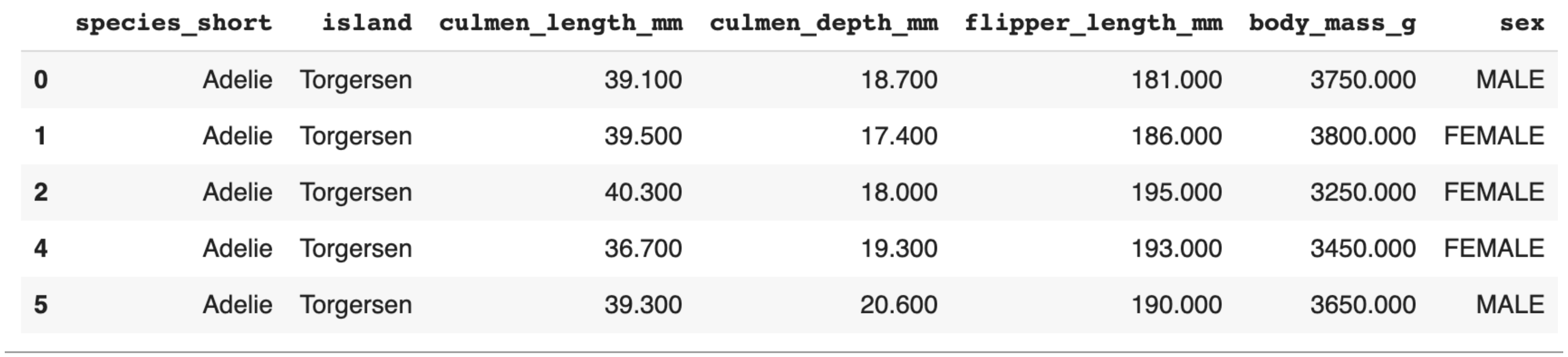
- Convert fields with text into numeric form
- Remove rows with missing / faulty data
- Fix rows with faulty datapoints
- Drop unnecessary variables (columns)
- Scaling or normalizing data
- Creating new variables
- Renaming variables (remove capitalization, spaces)

Removing Missing Values



	species_short	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.100	18.700	181.000	3750.000	MALE
1	Adelie	Torgersen	39.500	17.400	186.000	3800.000	FEMALE
2	Adelie	Torgersen	40.300	18.000	195.000	3250.000	FEMALE
3	Adelie	Torgersen	nan	nan	nan	nan	NaN
4	Adelie	Torgersen	36.700	19.300	193.000	3450.000	FEMALE

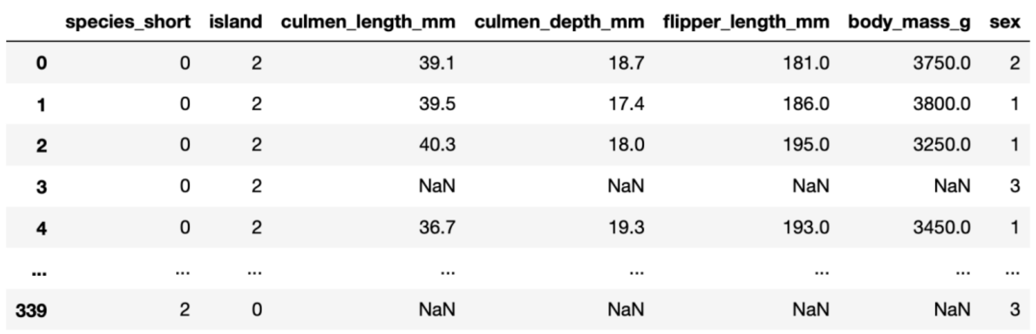
In this example we drop all rows with missing data df.dropna()



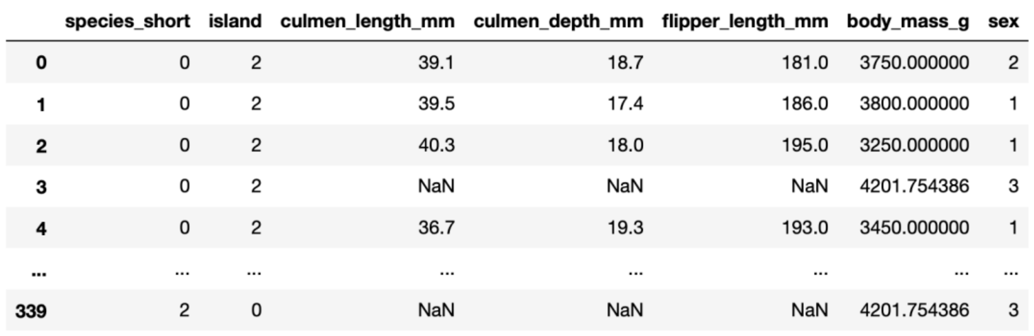
	species_short	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.100	18.700	181.000	3750.000	MALE
1	Adelie	Torgersen	39.500	17.400	186.000	3800.000	FEMALE
2	Adelie	Torgersen	40.300	18.000	195.000	3250.000	FEMALE
4	Adelie	Torgersen	36.700	19.300	193.000	3450.000	FEMALE
5	Adelie	Torgersen	39.300	20.600	190.000	3650.000	MALE

Imputing missing values

- Replace missing values by mean or median of the features.



	species_short	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	0	2	39.1	18.7	181.0	3750.0	2
1	0	2	39.5	17.4	186.0	3800.0	1
2	0	2	40.3	18.0	195.0	3250.0	1
3	0	2	NaN	NaN	NaN	NaN	3
4	0	2	36.7	19.3	193.0	3450.0	1
...	...	...	...	...	...	...	...
339	2	0	NaN	NaN	NaN	NaN	3



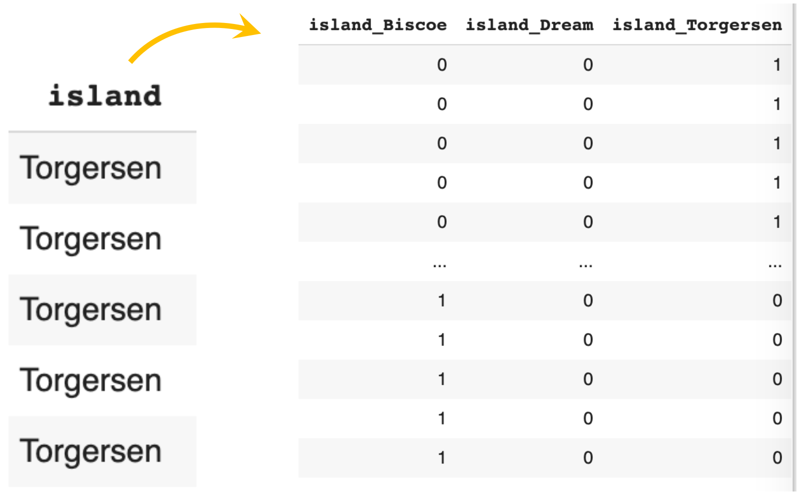
	species_short	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	0	2	39.1	18.7	181.0	3750.000000	2
1	0	2	39.5	17.4	186.0	3800.000000	1
2	0	2	40.3	18.0	195.0	3250.000000	1
3	0	2	NaN	NaN	NaN	4201.754386	3
4	0	2	36.7	19.3	193.0	3450.000000	1
...	...	...	...	...	...	...	...
339	2	0	NaN	NaN	NaN	4201.754386	3

```
df[["body_mass_g"]] = imp.fit_transform(df[["body_mass_g"]])
```

Changing categorical to numerical values

- Datasets can contain categorical and ordinal features with text.
- These need to be converted to numeric form

- Several ways are:
    - **One-hot encoding (use)**
    - Label encoding
    - Ordinal encoding (avoid)



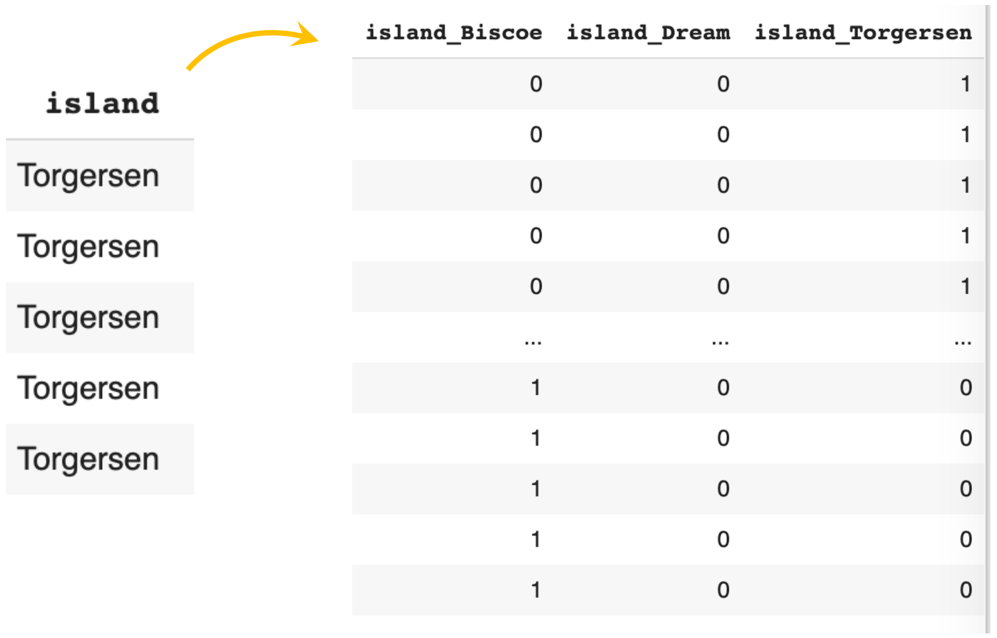
island
Torgersen
Torgersen
Torgersen
Torgersen
Torgersen

island_Biscoe	island_Dream	island_Torgersen
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1
...	...	...
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0

Categorical Variables: Variables that take on names or labels.
Ex: Marital status (“married”, “single”, “divorced”)

Dummy variables and one-hot encoding

- A dummy variable is a binary variable (0/1)
- One-hot encoding transforms each possible value from a categorical variable into a new binary column.



island
Torgersen
Torgersen
Torgersen
Torgersen
Torgersen

island_Biscoe	island_Dream	island_Torgersen
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1
...	...	...
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0

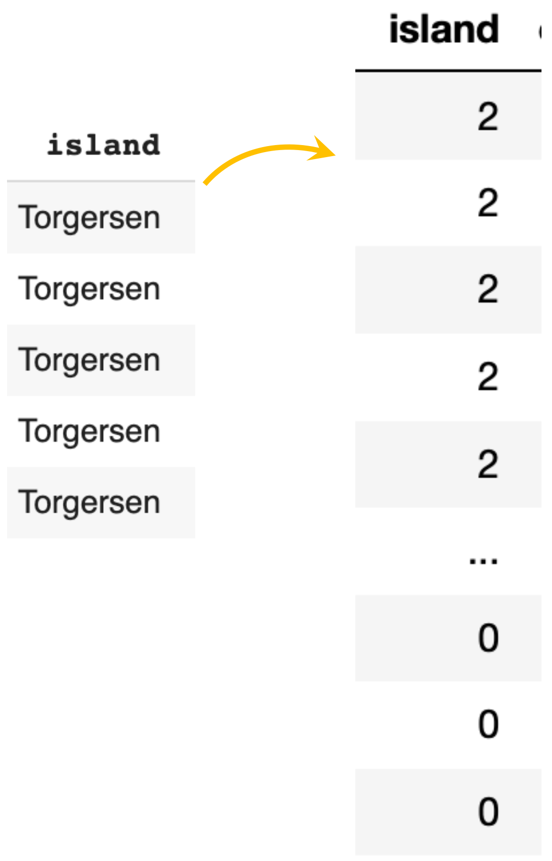
pd.get_dummies()

sklearn.preprocessing.OneHotEncoder()

Use one-hot encoding for nominal features in most ML models.

Label encoding

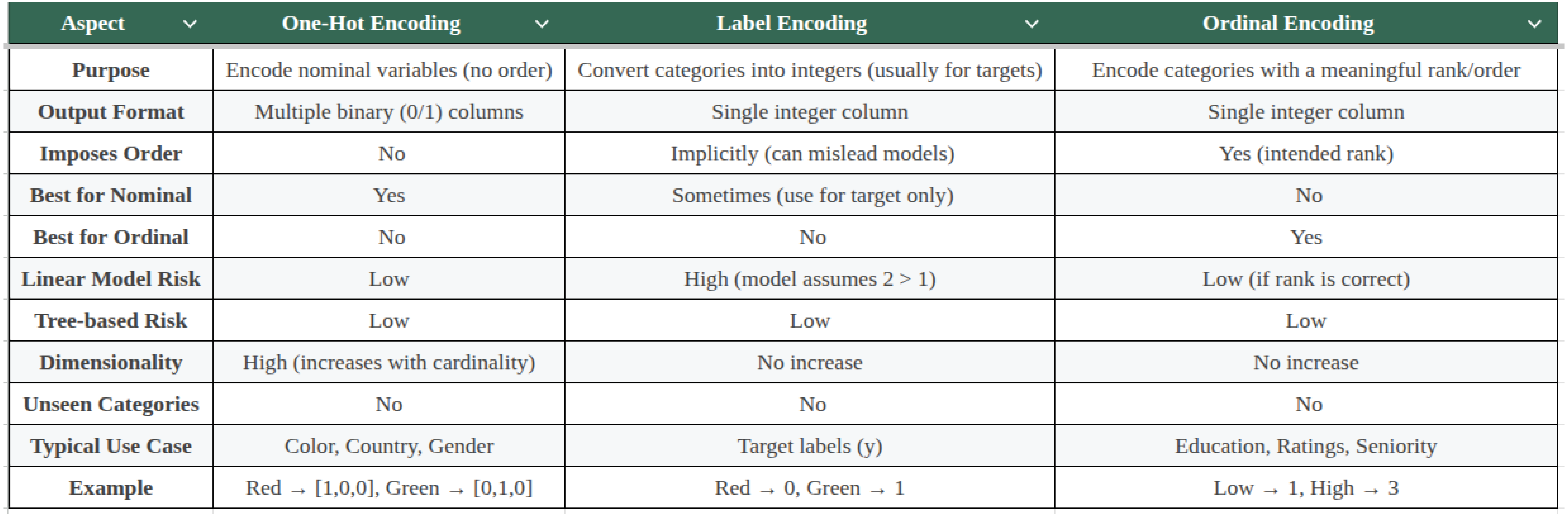
- Instead of creating columns with binary encoding, each possible value is given a unique integer value
- Penguins case: Torgensen = 2, Biscoe = 1, Dream = 0
- Issue: Algorithms might misjudge there to be an order or relationship in the data that does not really exist!!



island
Torgersen
Torgersen
Torgersen
Torgersen
Torgersen

island
2
2
2
2
2
...
0
0
0

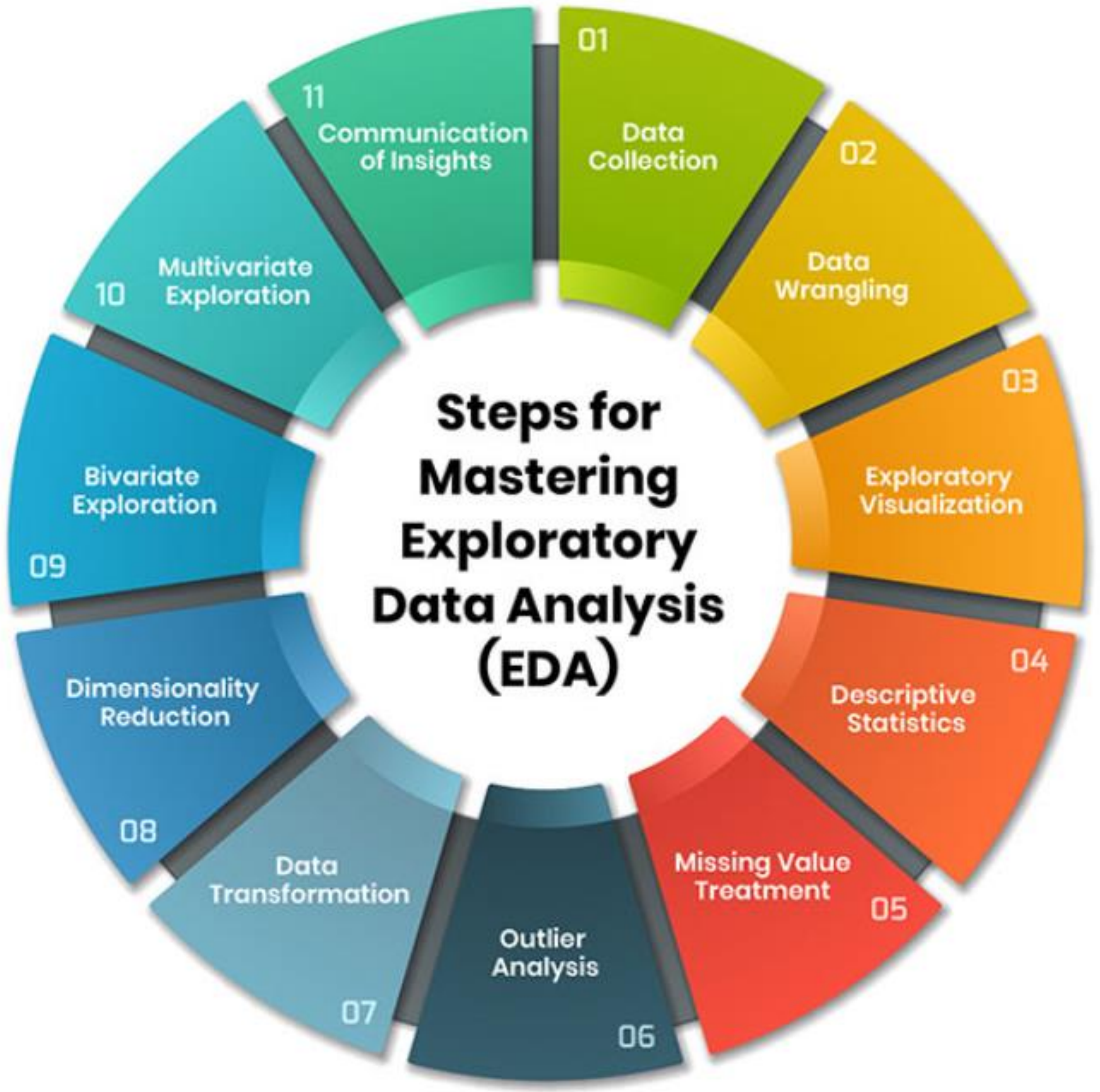
sklearn.preprocessing.LabelEncoder()



Aspect	One-Hot Encoding	Label Encoding	Ordinal Encoding
Purpose	Encode nominal variables (no order)	Convert categories into integers (usually for targets)	Encode categories with a meaningful rank/order
Output Format	Multiple binary (0/1) columns	Single integer column	Single integer column
Imposes Order	No	Implicitly (can mislead models)	Yes (intended rank)
Best for Nominal	Yes	Sometimes (use for target only)	No
Best for Ordinal	No	No	Yes
Linear Model Risk	Low	High (model assumes 2 > 1)	Low (if rank is correct)
Tree-based Risk	Low	Low	Low
Dimensionality	High (increases with cardinality)	No increase	No increase
Unseen Categories	No	No	No
Typical Use Case	Color, Country, Gender	Target labels (y)	Education, Ratings, Seniority
Example	Red → [1,0,0], Green → [0,1,0]	Red → 0, Green → 1	Low → 1, High → 3

Exploratory Data Analysis (EDA)

- It involves investigating the key characteristics, relationships and patterns in a dataset to gain useful insights.
- EDA can help uncover hidden trends, identify anomalies, assess data quality issues and generate hypotheses for further analysis.


Steps for Mastering Exploratory Data Analysis (EDA)
01 Data Collection
02 Data Wrangling
03 Exploratory Visualization
04 Descriptive Statistics
05 Missing Value Treatment
06 Outlier Analysis
07 Data Transformation
08 Dimensionality Reduction
09 Bivariate Exploration
10 Multivariate Exploration
11 Communication of Insights

Exploratory Data Analysis (EDA)

- An approach for data analysis that employs a variety of techniques (mostly graphical) to
  - uncover underlying structure;
  - extract important variables;
  - detect outliers and anomalies;
  - test underlying assumptions;
  - develop parsimonious models; and
  - determine optimal factor settings.

https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm

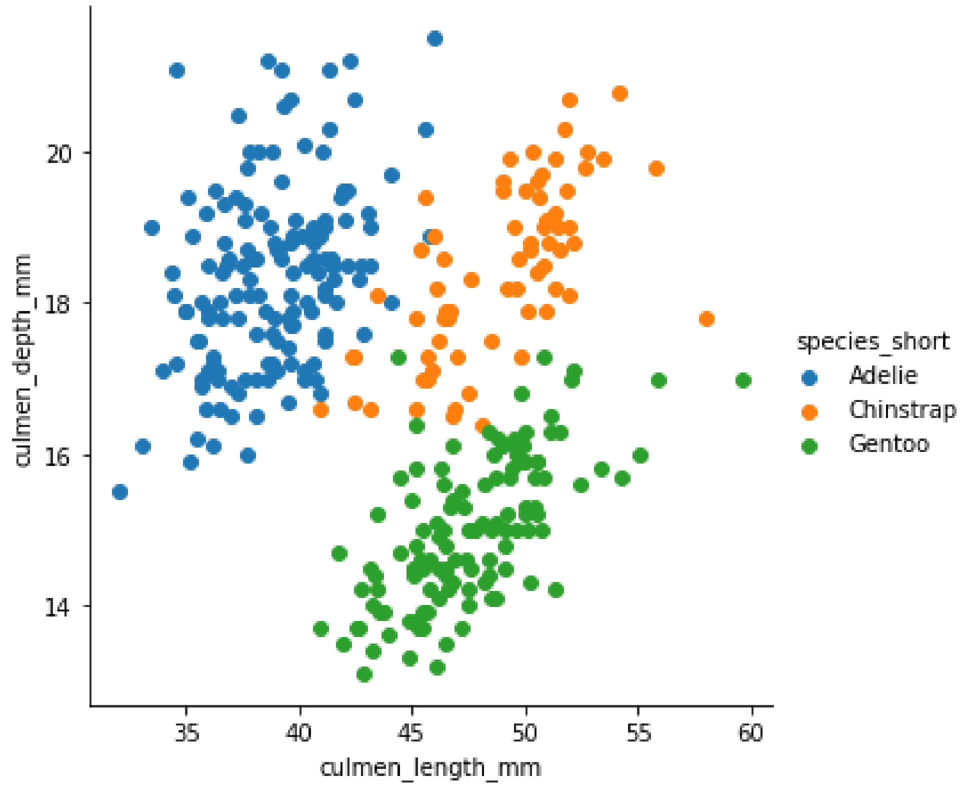
Analysis Steps

- Univariate Distributions (ONE variable)
  - **Histograms:** Purpose: see shape, center, spread; spot extreme values.
  - **Skewness check** *Use this together with histograms* to decide if transformations (log, etc.) might be needed.
  - **Boxplots** Purpose: visually confirm outliers and variability.
- What do my variables look like individually? Any weird values?
- **Scatter plots:** For continuous–continuous relationships that have clear clinical meaning.
  - Which variables look associated with heart disease?

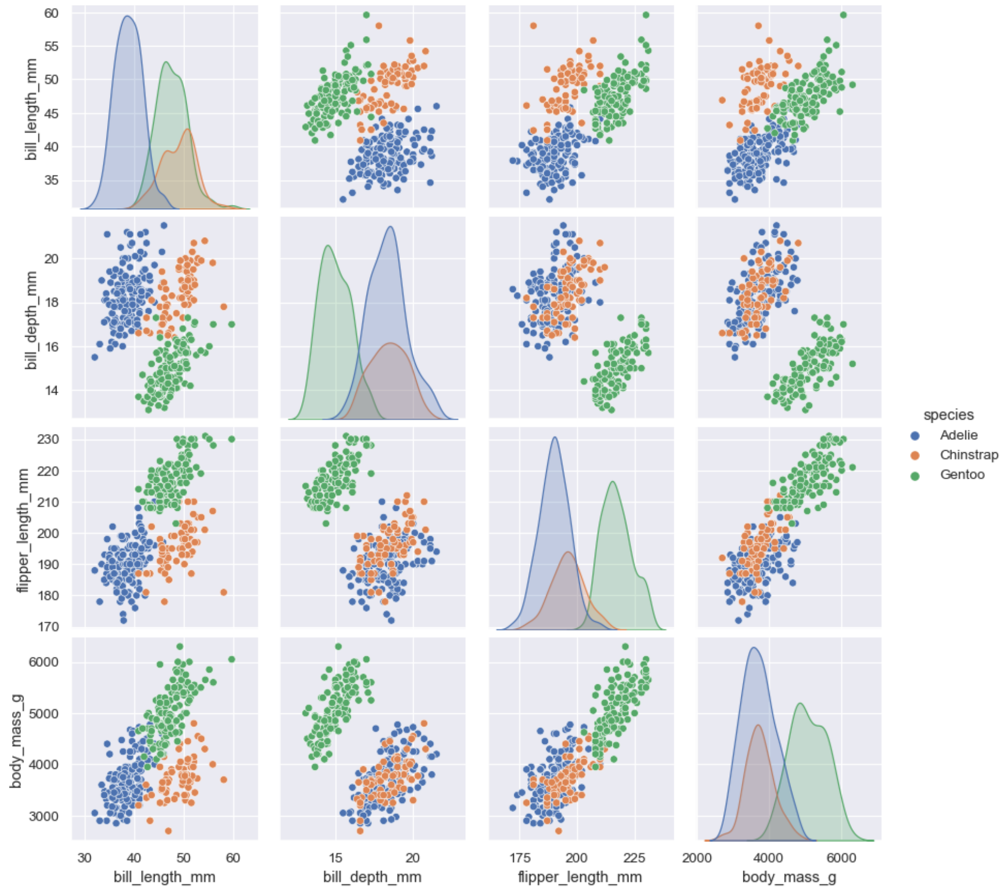
Analysis Steps

- Multivariate Relationships (Features vs Features)
- Pair plot (selected subset)
  - Use 4–6 important variables + target.
  - Good for an overview of multiple bivariate relationships at once.
- Correlation heatmap
  - On numeric features.
  - Detect highly correlated predictors
- How do predictors relate to each other?
- Any redundancy or strong patterns?

EDA Examples



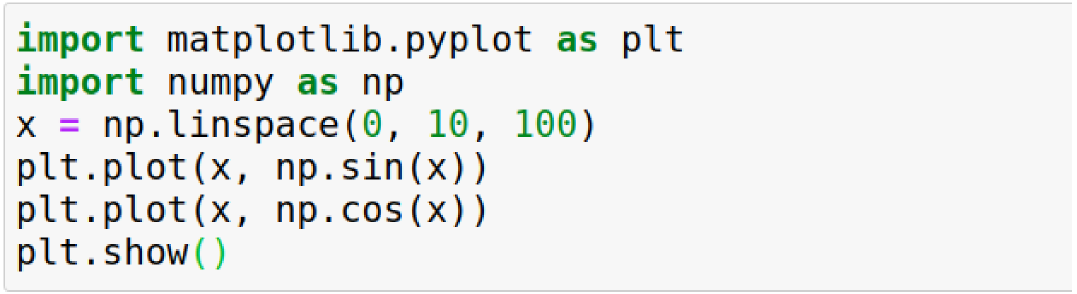
sns.FacetGrid(df, hue="species_short", height=5).map(plt.scatter, "culmen_length_mm", "culmen_depth_mm").add_legend()



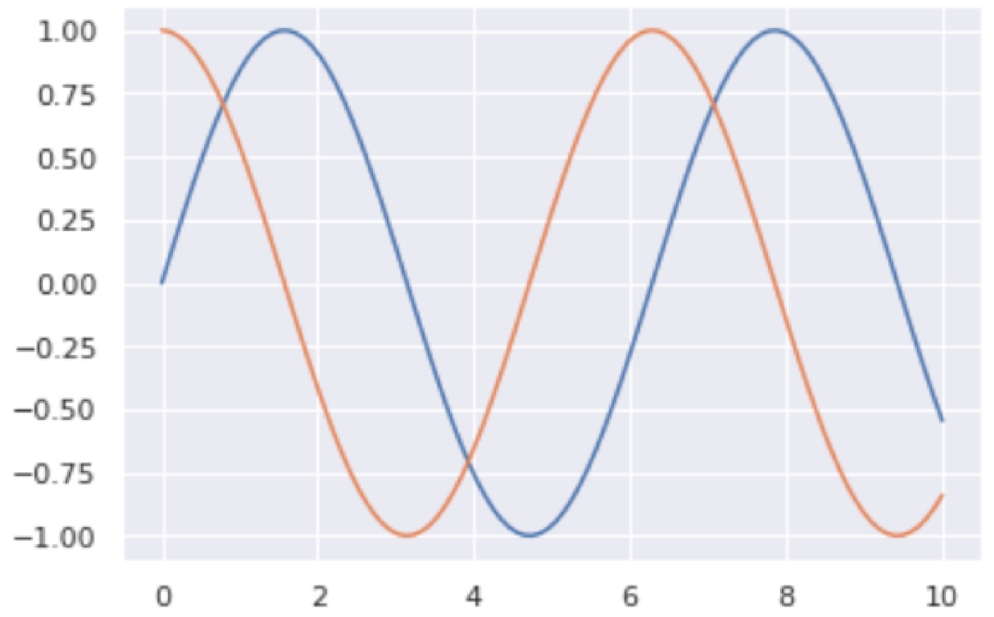
sns.pairplot(df, hue="species_short")

Visualization with Matplotlib

- import matplotlib as mpl
- import matplotlib.pyplot as plt
- plt interface is most often used
- Set "classic style": plt.style.use('classic')
- plt.show() command should be used only once per Python session.
- Can save a figure using the savefig(): fig.savefig('my_figure.png').
- %matplotlib notebook: for interactive plots
- %matplotlib inline: for static images

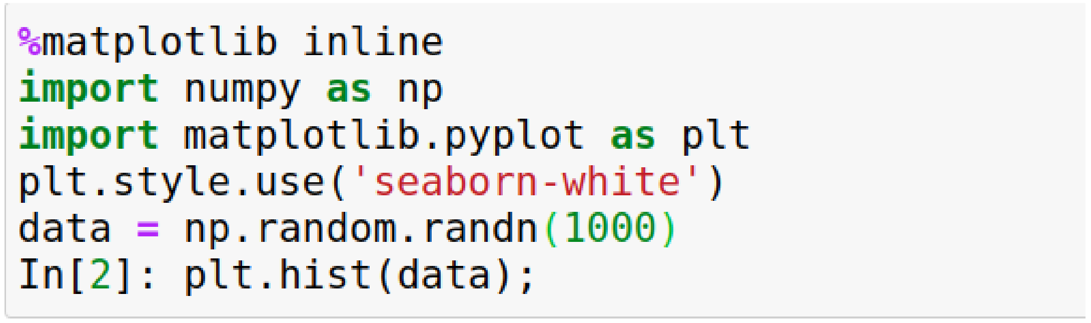


```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.show()
```

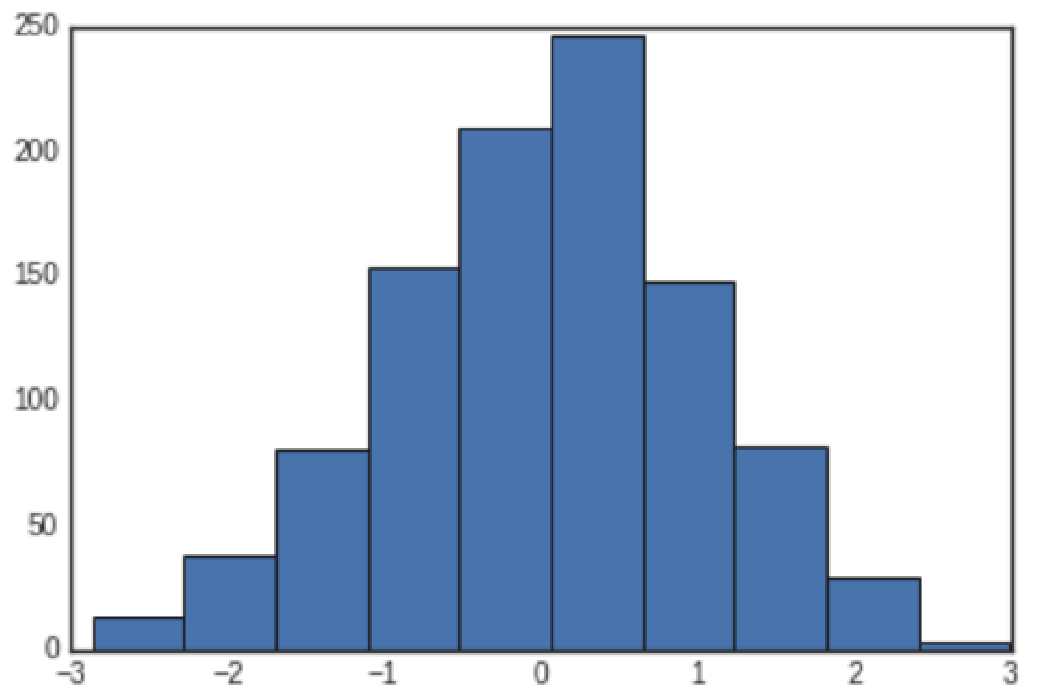


Visualization and Histogram

- plt.plot is more efficient than plt.scatter.
- plt.scatter can be used to create plots where the properties of each individual point (size, face color, edge color, etc.) can be individually controlled or mapped to data.
- Histogram can be a great first step in understanding a dataset

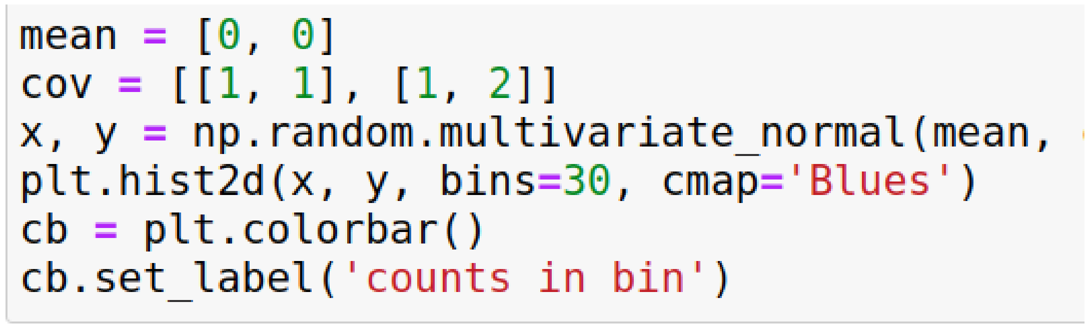


```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
data = np.random.randn(1000)
In[2]: plt.hist(data);
```

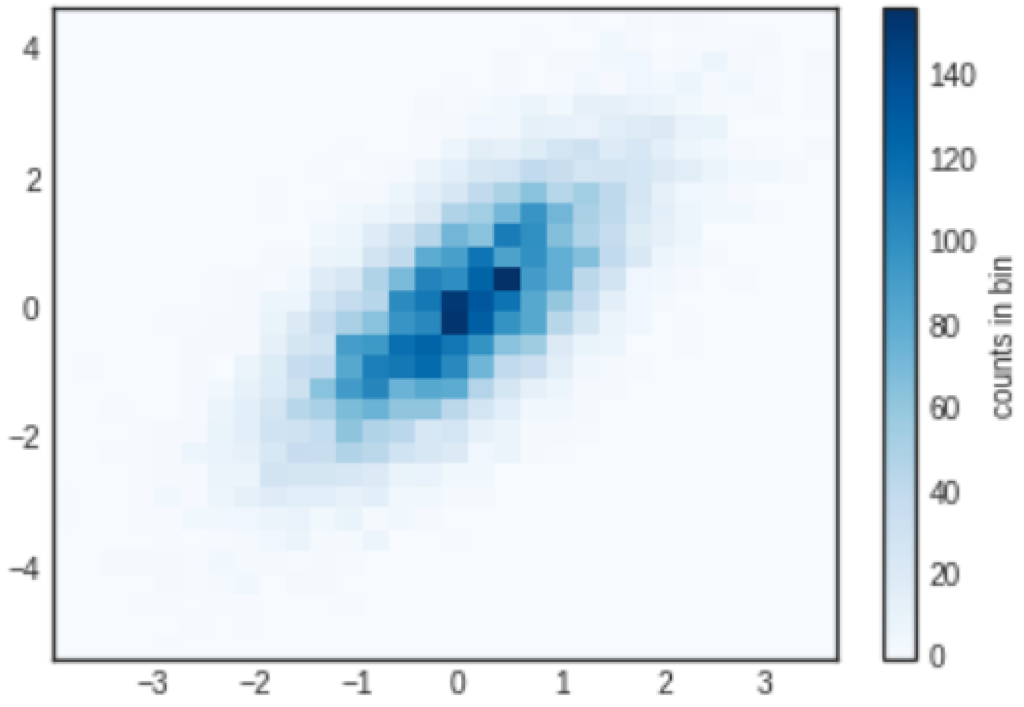


2D Histogram

- We created 1D-histogram by dividing the number line into bins.
- Can create 2D by dividing points among 2D bins.
    - Use Matplotlib's plt.hist2d function.



```
mean = [0, 0]
cov = [[1, 1], [1, 2]]
x, y = np.random.multivariate_normal(mean,
plt.hist2d(x, y, bins=30, cmap='Blues')
cb = plt.colorbar()
cb.set_label('counts in bin')
```



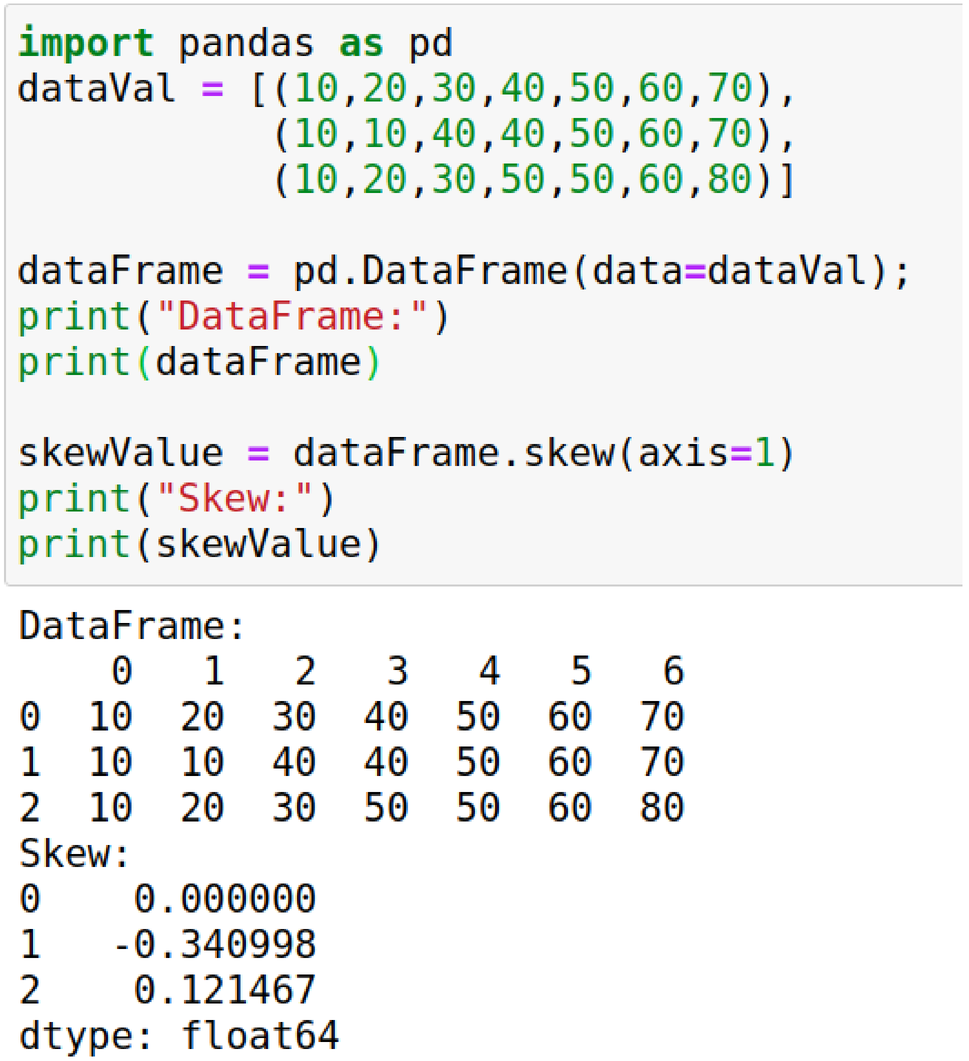
Skewness

- Measure of asymmetry of probability distribution of a real-valued random variable about its mean. [Wikipedia*].
  - Output 0 denotes a symmetrical distribution.
- **Example:**
  - Many clinical variables are naturally right-skewed, such as cholesterol levels, triglycerides, CRP (inflammation marker, blood test).
  - Skewed distributions may require log transformation before modeling to improve performance.
- **Why It Matters:**
  - Skewness affects model accuracy because extreme clinical values can disproportionately influence predictions.
  - Understanding skewness helps identify when a variable needs transformation or normalization.

https://en.wikipedia.org/wiki/Skewness

Skewness Example

- dataframe.skew() returns unbiased skew over requested axis (of DataFrame object) Normalized by N-1.
- Output 0 denotes a symmetrical distribution.

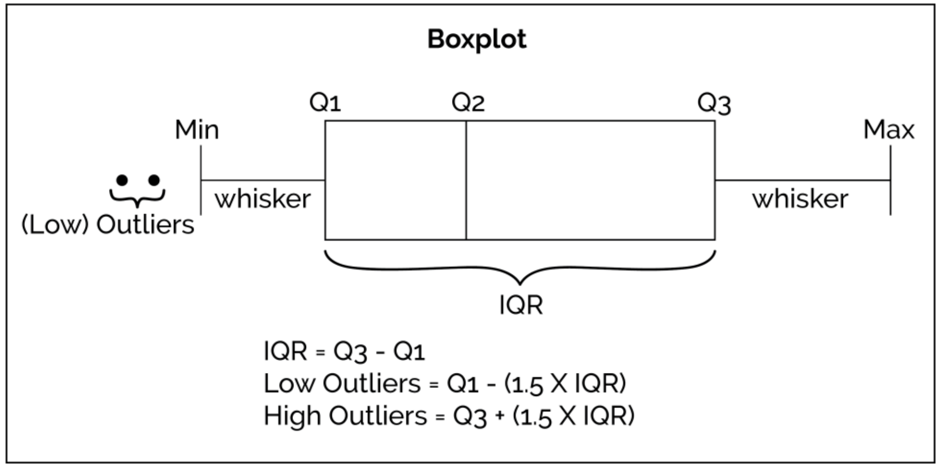


```python
import pandas as pd
dataVal = [(10,20,30,40,50,60,70),
           (10,10,40,40,50,60,70),
           (10,20,30,50,50,60,80)]

dataFrame = pd.DataFrame(data=dataVal);
print("DataFrame:")
print(dataFrame)

skewValue = dataFrame.skew(axis=1)
print("Skew:")
print(skewValue)
```

```
DataFrame:
    0   1   2   3   4   5   6
0  10  20  30  40  50  60  70
1  10  10  40  40  50  60  70
2  10  20  30  50  50  60  80
Skew:
0    0.000000
1   -0.340998
2    0.121467
dtype: float64
```

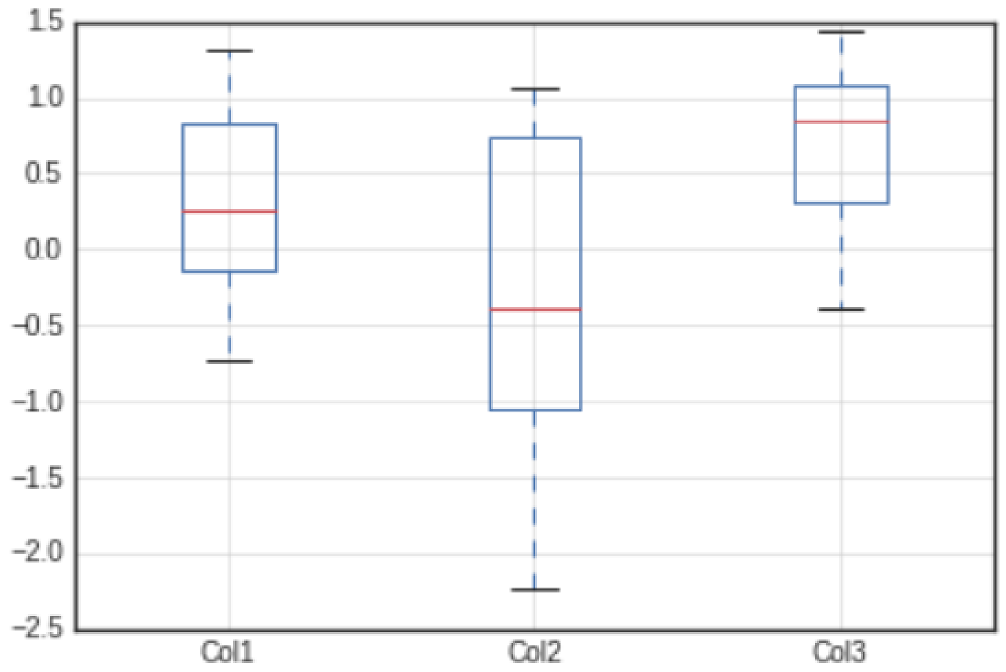
Box Plot

- A box plot is a method for graphically depicting groups of numerical data through their quartiles.



IQR = Q3 - Q1
Low Outliers = Q1 - (1.5 X IQR)
High Outliers = Q3 + (1.5 X IQR)

```
np.random.seed(1234)
df = pd.DataFrame(np.random.randn(10, 4),
                  columns=['Col1', 'Col2', 'Col3', 'Col4'])
boxplot = df.boxplot(column=['Col1', 'Col2', 'Col3'])
```

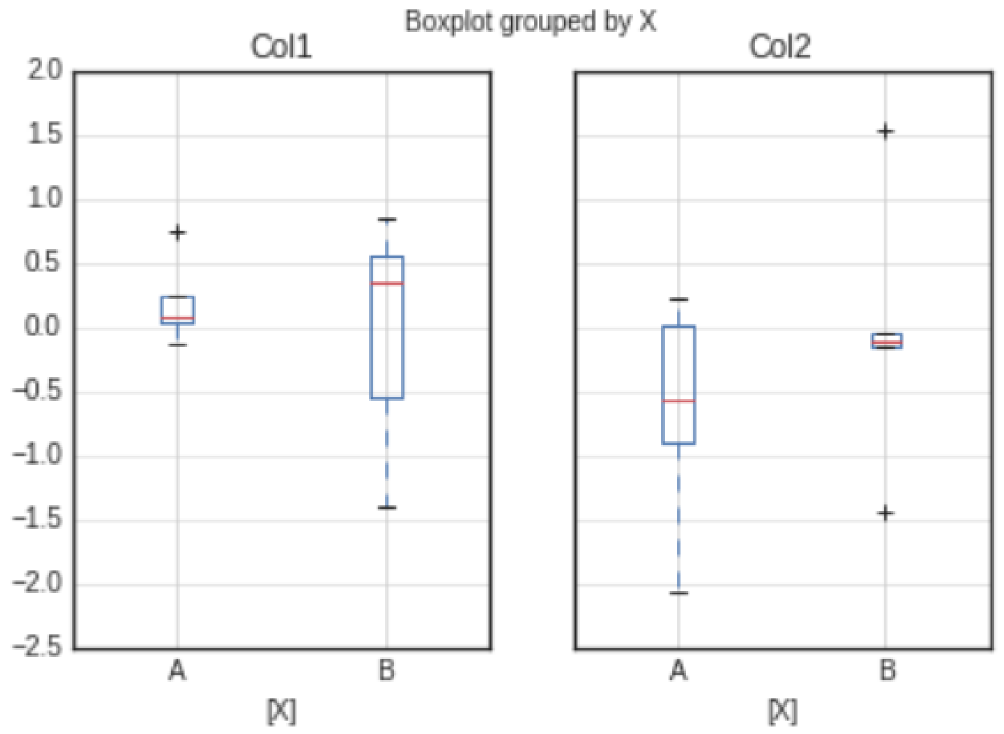


https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.boxplot.html

Box Plot

- Boxplots of variables distributions grouped by the values of a third variable can be created using the option **by**.

```
df = pd.DataFrame(np.random.randn(10, 2),
                  columns=['Col1', 'Col2'])
df['X'] = pd.Series(['A', 'A', 'A', 'A', 'A',
                     'B', 'B', 'B', 'B', 'B'])
boxplot = df.boxplot(by='X')
```



Bias-variance & Loss functions

Classification and Regression Task

- In regression, output is quantitative: e.g., revenue, temperature or forecasted values of a ticker.
- In classification, output is qualitative, where classes can be categorical (unordered, yes or no), nominal (unordered, like dog, cat or fish) or ordinal (ordered, like low, medium or high risk).

Normalization and Standardization

- Why Scaling Matters: The distribution of the data is unknown or does not follow a Gaussian distribution.
  - Normalization is needed when using ML algorithms that rely on distances between data points (such as k-Nearest Neighbors)
  - Scales features to a specific range .

- Standardization (called z-score scaling) transforms data to have a mean of 0 and a standard deviation of 1.
- Adjusts feature values by subtracting mean and dividing by SD.
- Appropriate in the following cases:
  - Gradient-based Algorithms: Support Vector Machine (SVM) requires standardized data for optimal performance.
  - Dimensionality Reduction: Standardization is in dimensionality reduction techniques.

Bias-Variance

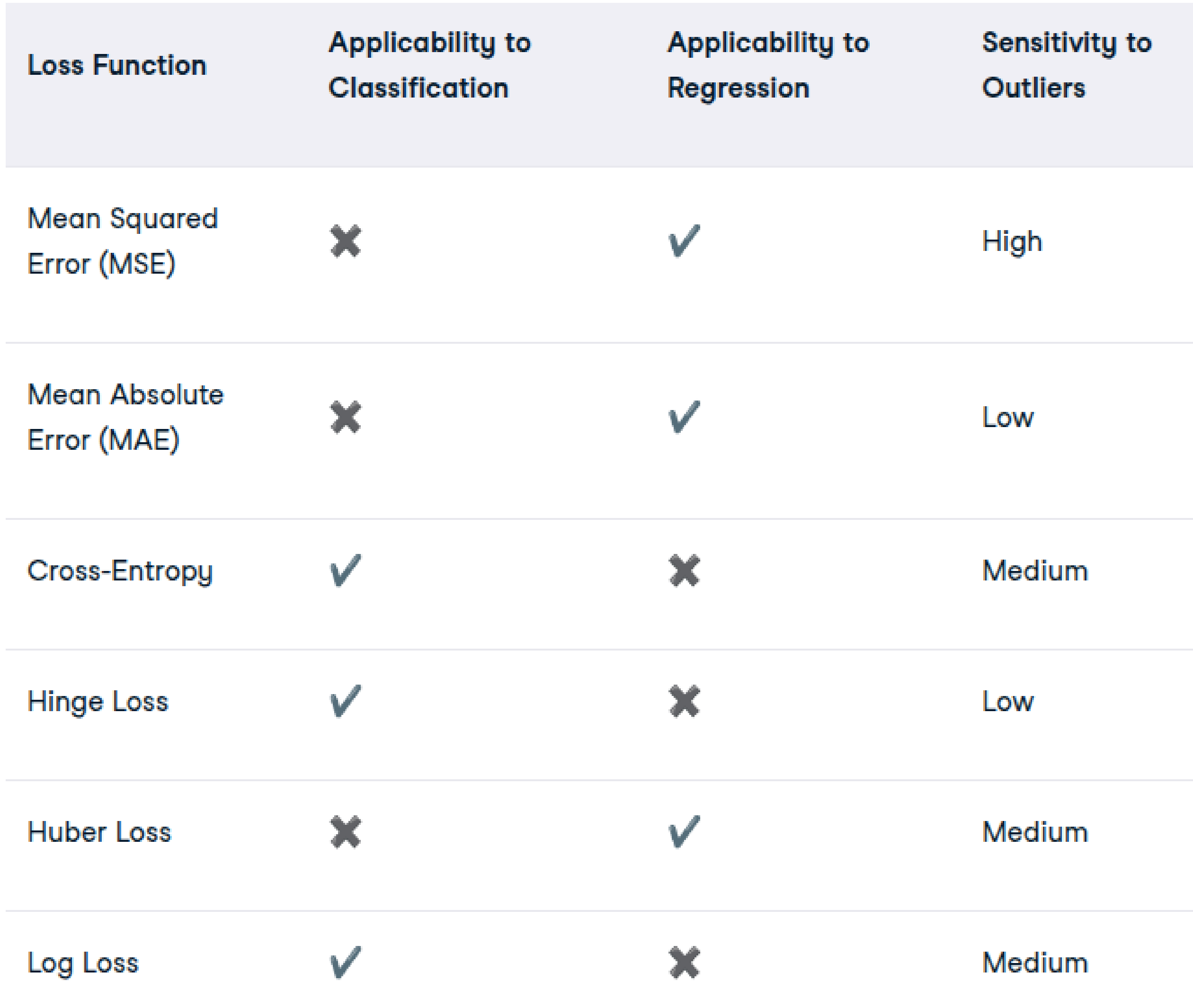
- Influence predictive model performance.
- Goal: balance both to generalize well to unseen data
- Bias:
  - Error due to overly simplistic model assumptions.
  - Leads to underfitting (poor capture of underlying patterns).
  - Bias decreases –> model better fits training data.
- Variance:
  - Error from a model’s sensitivity to fluctuations in training data.
  - Leads to overfitting (fits noise as if signal).
  - Variance increases –> model more sensitive to sample noise

Bias-Variance

- Bias-variance tradeoff explains why models that fit training data *perfectly* often fail in production.
    - Excess complexity → overfitting
    - Too simple → underfitting
- Cross-validation to estimate generalization error.
- Control Techniques:
    - Regularization (penalize complexity).
    - Ensemble methods (reduce variance).
    - Hyperparameter tuning (complexity control).

Loss Functions

- Loss function also referred as error function
    - Used to measure model performance by calculating the deviation of a model's predictions from the correct, "ground truth" predictions.
- MSE is recommended for ML scenarios where it is conducive to the learning process to penalize significantly the presence of outliers.
- MAE is inherently less sensitive to outliers because it assigns an equal weight to all errors, regardless of their magnitude.



Loss Function	Applicability to Classification	Applicability to Regression	Sensitivity to Outliers
Mean Squared Error (MSE)	✖	✓	High
Mean Absolute Error (MAE)	✖	✓	Low
Cross-Entropy	✓	✖	Medium
Hinge Loss	✓	✖	Low
Huber Loss	✖	✓	Medium
Log Loss	✓	✖	Medium

Src: DataCamp

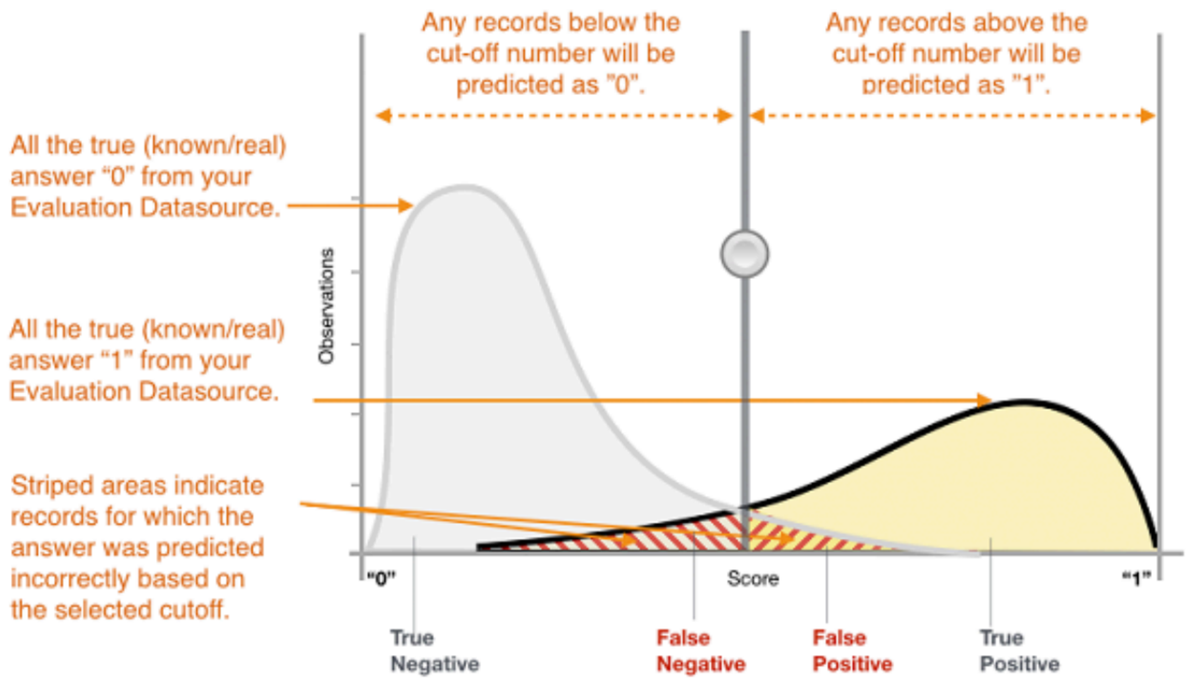
Loss Functions

- Binary Cross-Entropy Loss (BCE) is a performance measure for classification models that outputs a prediction with a probability value (between 0 and 1).
    - It corresponds to likelihood of a data sample belonging to class/category.
        - Entropy: calculate the degree of randomness or disorder within a system
        - Cross-entropy: measures the differences between two probability distributions
- Huber loss operates in two modes that are switched based on the size of the calculated difference between the actual target value and the prediction of the ML algorithm.
- Choosing the right loss function is crucial for effective ML model training.

Src: DataCamp

Binary Classification

- Actual output of many binary classification algorithms is a prediction score which indicates system's certainty that given observation belongs to positive class.
- Binary classification accuracy metrics quantify the two types of correct predictions and two types of errors.
- Typical metrics are accuracy (ACC), precision, recall, false positive rate, F1-measure.


Any records below the cut-off number will be predicted as "0".
Any records above the cut-off number will be predicted as "1".
All the true (known/real) answer "0" from your Evaluation Datasource.
All the true (known/real) answer "1" from your Evaluation Datasource.
Striped areas indicate records for which the answer was predicted incorrectly based on the selected cutoff.
Observations
"0"
Score
"1"
True Negative
False Negative
False Positive
True Positive

Classification Metrics

- Classification models assign each observation to a **category** (e.g., disease vs. no disease).
- Metrics evaluate prediction quality for each class.
- Classification models have discrete output.
  - Accuracy: Overall proportion of correct classifications
  - Recall: How many actual positives did we detect?
  - and Precision: How many predicted positives were actually correct?”
  - F1 Score: Balanced measure useful when classes are imbalanced.
  - Confusion Matrix (not a metric)
  - AU-ROC

Accuracy, Recall and Precision

- Accuracy is the proportion of all classifications that were correct, whether positive or negative. Use as a rough indicator of model training progress/convergence for balanced datasets.

$$\text{Accuracy} = \frac{\text{correct classifications}}{\text{total classifications}} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Recall or true positive rate (TPR) is the proportion of all actual positives that were classified correctly as positives.

$$\text{Recall (or TPR)} = \frac{\text{correctly classified actual positives}}{\text{all actual positives}} = \frac{TP}{TP + FN}$$

- Precision is the proportion of all the model's positive classifications that are actually positive.

$$\text{Precision} = \frac{\text{correctly classified actual positives}}{\text{everything classified as positive}} = \frac{TP}{TP + FP}$$

TP is the number of true positives
FN is the number of false negatives
FP is the number of false positives

In highly imbalanced datasets with very few positive instances, Precision and Recall may exhibit high variance and should be complemented with additional metrics.

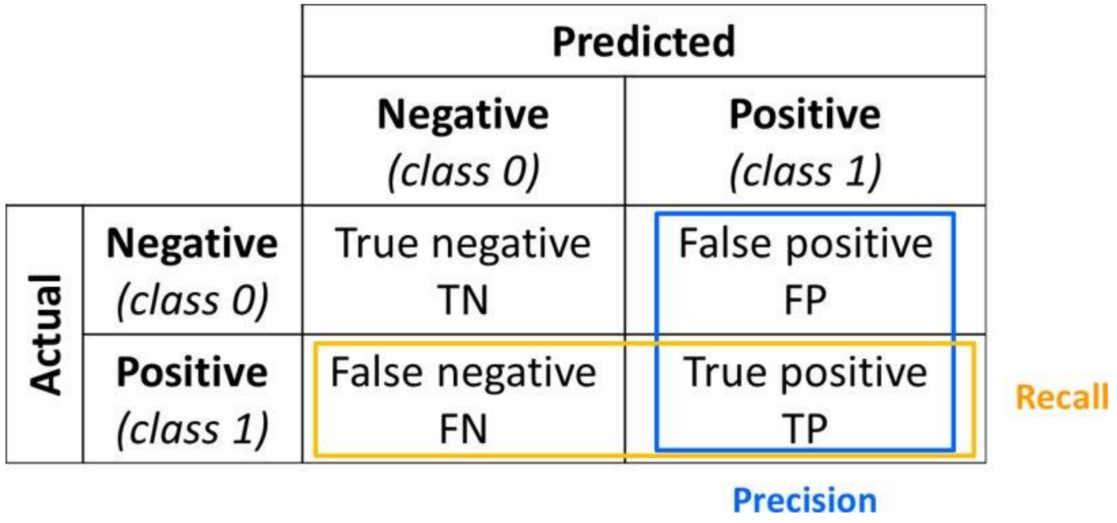
F1 Score

- F1 score/ balanced F-score / F-measure: is a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.
- Relative contribution of precision and recall to F1 score are equal.
- F1 is by default calculated as 0. [0 means no true positives, false negatives, or false positives].
- It balances the importance of precision and recall, and is preferable to accuracy for class-imbalanced datasets.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Confusion Matrix

- Confusion matrix evaluate the accuracy of a classification.
  - Rows represent the actual classes the outcomes should have been.
  - Columns represent the predictions we have made.
    - True Positives (TP): Number of instances correctly predicted as positive.
    - True Negatives (TN): Number of instances correctly predicted as negative.
    - False Positives (FP): Number of instances incorrectly predicted as positive.
    - False Negatives (FN): Number of instances incorrectly predicted as negative.



		Predicted	
		Negative *(class 0)*	**Positive** *(class 1)*
Actual	**Negative** *(class 0)*	True negative TN	False positive FP
	Positive *(class 1)*	False negative FN	True positive TP

Recall
Precision

Example

- **Disease Diagnosis (binary classification)**
  - TP → correctly diagnosing pneumonia
  - FN → failing to detect pneumonia
  - FP → unnecessary antibiotic prescription
  - TN → normal X-ray correctly classified
  - Here, recall is especially important (avoid missing patients).
- **Predicting ICU Admission (risk scoring)**
  - High recall → fewer dangerous misses
  - High precision → fewer unnecessary ICU transfers
- **Cancer Screening**
  - FN is extremely costly → prioritize sensitivity (recall)
  - ROC-AUC helps compare models independent of thresholds

AUC - ROC Curve

- **AUC**, area under the receiver operating characteristic (**ROC**) curve.
- The Reciever operating characteristic curve plots the true positive (**TP**) rate versus the false positive (**FP**) rate at different classification thresholds.
- The thresholds are different probability cutoffs that separate the two classes in binary classification.
- It uses probability to tell us how well a model separates the classes.

- Measures the ability of a binary classification model to assign higher scores to positive examples than to negative examples.

Regression Metrics

- Regression models have continuous output.
- Calculate some sort of distance between predicted and ground truth.
- Regression models:
  - Mean Absolute Error (MAE),
  - Mean Squared Error (MSE),
  - Root Mean Squared Error (RMSE),
  - R^2 (R-Squared).

Regression Metrics

- **Mean Absolute Error (MAE)**: is the average of the difference between the ground truth and the predicted values.
  - Robust towards outliers
  - Doesn’t give an idea of the error direction (under/over prediction)
  - MAE is non-differentiable

$$MAE = \frac{1}{N}\sum_{j=1}^{N}|y_j - \check{y}_j|$$

Where:
y_j: ground-truth value
y_hat: predicted value
N: number of datums

- **Mean Squared Error (MSE)**: finds average of squared difference between target value and value predicted by regression model.
- Can be optimized better but more prone to outliers
- Penalizes small errors by squaring them, leads to an overestimation.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(y_j - \check{y}_j)^2$$

Regression Metrics

- Root Mean Squared Error (RMSE): sqrt(MSE): is square root of average of squared difference between target value and value predicted by regression model.
  - Penalizes small errors done by MSE by square rooting it.
  - Less prone outliers.

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_j - \check{y}_j)^2}$$

R^2

- R^2 (R-Squared): Range $(-\infty,1)$
- R^2 Coefficient of determination:
  - How much the total variation in target is explained by the variation in regression line?
  - This is calculated using the sum of squared errors.
  - If the sum of Squared Error is small: Regression has captured 100% of the variance in the target variable.
  - If high then wasn’t able to capture any variance.
    - If model overfits then variance explained will be 100% [Issue]
    - To solve it, we use Adjusted R^2; is lower than R^2
      - Only shows the real improvement

Example

- **Predicting Length of Hotel Stay**
  - RMSE is important since large errors (predicting 2 days vs 10 days) matter operationally.
- **Predicting Blood Glucose Levels**
  - MAE may be preferred for interpretable error magnitudes.
- **Predicting Cost of Medical Treatment**
  - Outliers are common → MAE is more robust than MSE.

Machine Learning

Categories of ML

- Subcategories:
  - **Supervised** ML models are trained with labeled data sets.
  - **Unsupervised** ML models looks for patterns in unlabeled data.
  - **Reinforcement** ML trains machines through trial and error to take the best action by establishing a reward system.
    - Difficult to precisely specify the task.
    - Learning process of task can be dangerous.

Src: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

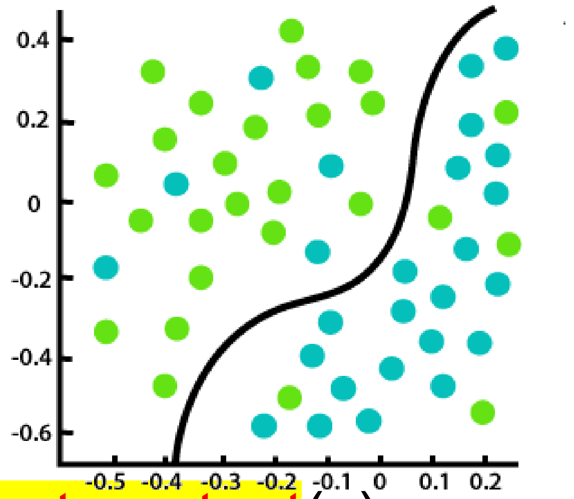
Supervised Learning

- Models are trained using "labeled" data, and predict output.
- Goal: Find a mapping function to map input var(x) with output var(y).
- E.g., training a model to recognize dog breeds from photos.
- Application: Image classification, Fraud Detection, spam filtering

Stanford Dogs dataset
(120 breeds)

Classification and Regression

- Process of finding a function which divides dataset into classes based on different parameters.
- Algorithms:
    - K-Nearest Neighbours
    - Support Vector Machines
    - Decision Tree Classification
    - Random Forest Classification



- Algorithm finds mapping function to map input(x) to discrete output(y).

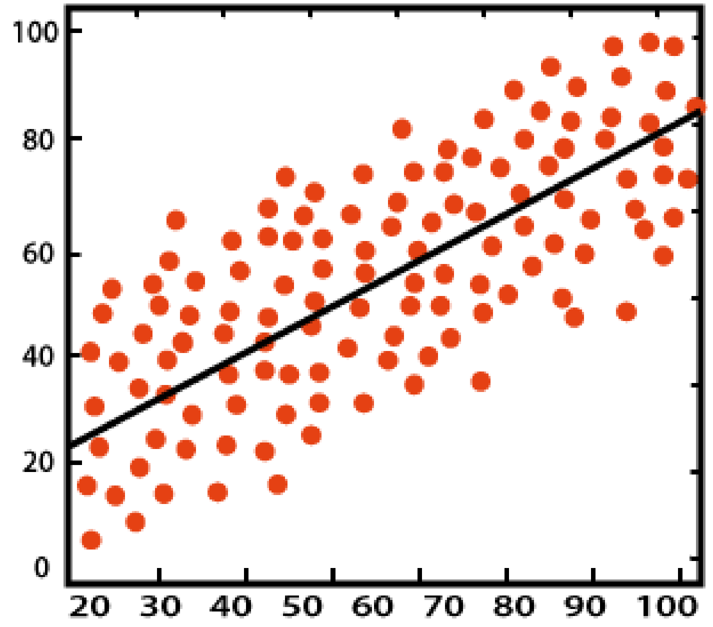
- Application:
    - Image labeling, Email Spam Detection.

https://www.javatpoint.com/regression-vs-classification-in-machine-learning

Classification and Regression

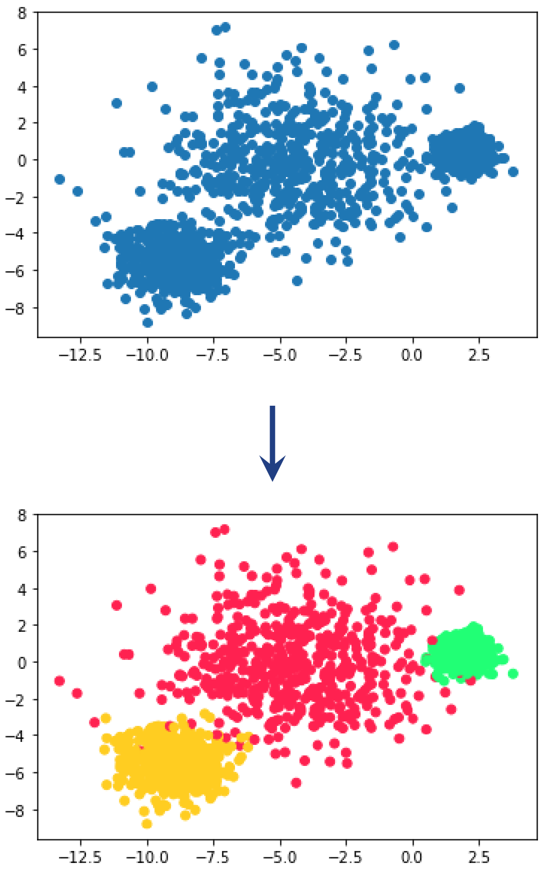
- Process of finding correlations between dependent and independent variables.
- Predicts continuous variables.
- Algorithm**:**
  - Simple Linear Regression
  - Multiple Linear Regression
  - Support Vector Regression
- Finds mapping function to map input var(x) to continuous output var(y).

- Application:
  - Stock prices, temperature.



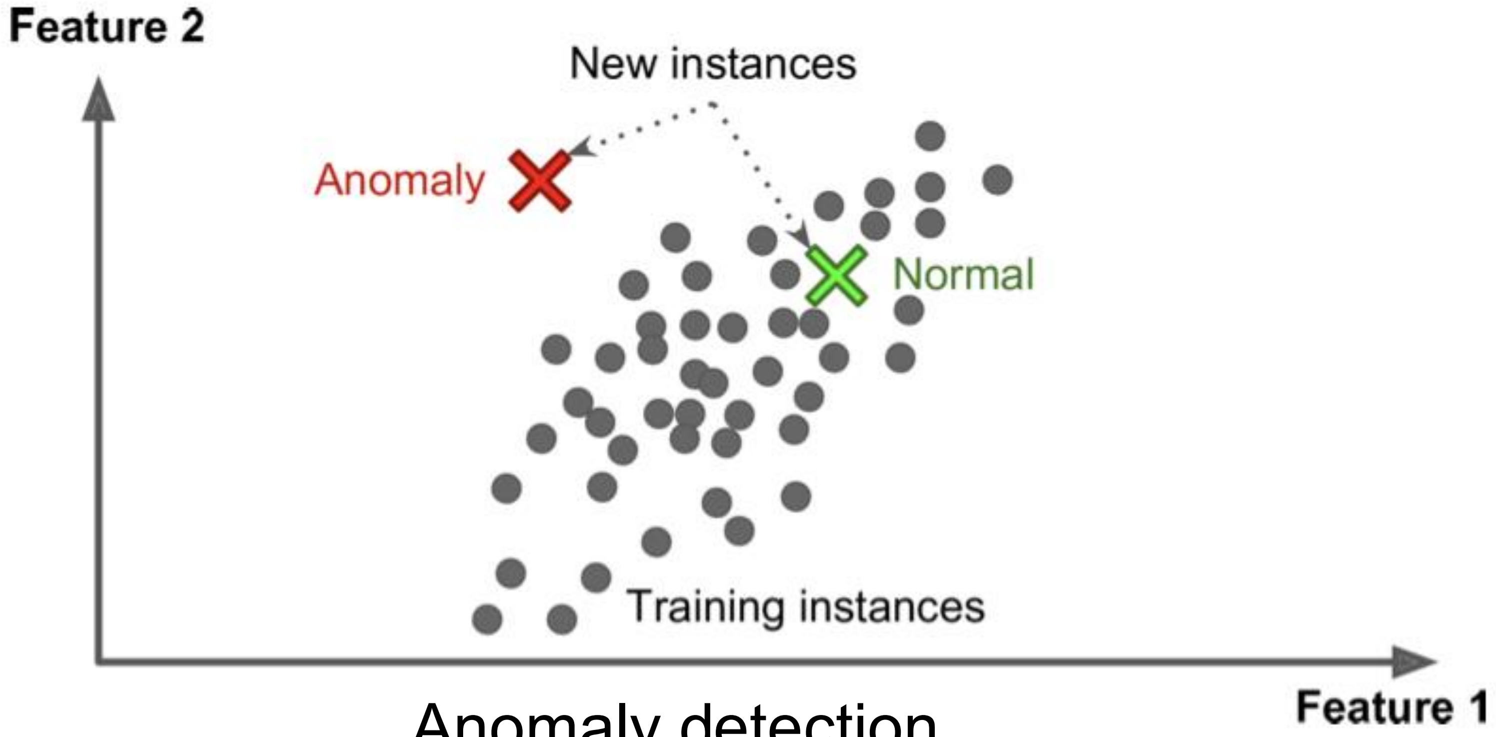
Unsupervised Learning

- Labels are not provided, algorithm learns from data
- E.g., Finding an outlier in transactions.
- Algorithms:
    - Clustering: K-means, Hierarchical CA, DBSCAN
    - Dimensionality reduction: Principal components analysis (PCA)



Unsupervised Learning

- Labels are not provided
- Algorithm learns from data.



Anomaly detection

Src: Image from Géron, A. (2019)

Overview of ML

- **Components**: Data, Model, Objective Function, Optimization Algorithm.
- **Data:** Input variables (features) and target outputs.
  - *Example:* In a diabetes dataset, each row represents a patient; columns include age, BMI, glucose levels, blood pressure, etc.
  - This forms the **design matrix** (X), while the outcome (e.g., diabetes yes/no) is the **label** (y).
- **Model:** A function that maps inputs to outputs.
- **Parametric models** (fixed number of parameters):
  Example: Logistic regression for predicting heart disease risk.

Algorithm and Machine Learning

- Choosing the Right Algorithm
  - Different ML algorithms have strengths and limitations.
  - Choice depends on data size, need for interpretability, and clinical risk.

Algorithm	Strengths	Limitations
Logistic Regression	Highly interpretable; fast	Limited to linear relationships
Decision Trees	Easy to explain; handles mixed data	Prone to overfitting
Random Forests	High accuracy; robust	Less interpretable
Support Vector Machines (SVM)	Effective for high-dimensional data	Hard to interpret; slow on large datasets
Neural Networks / Deep Learning	Excellent for images & signals	Low interpretability; high compute cost

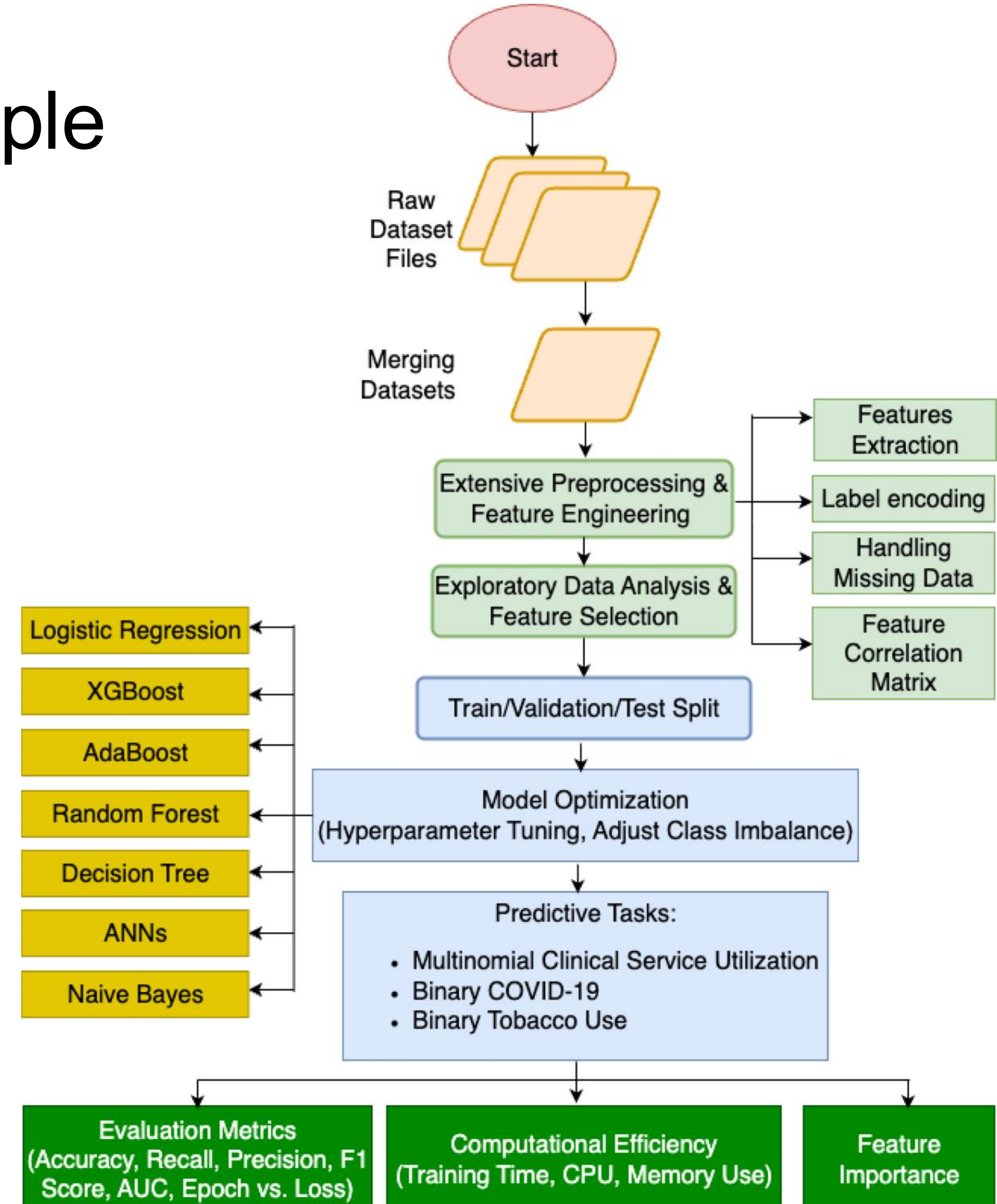
Algorithms & Machine Learning

- Interpretability vs. Performance
  - High interpretability (Logistic Regression, Decision Trees) → preferred for decision support where explanations are required.
  - High performance (Random Forest, SVM, Neural Networks) → used when predictive accuracy is critical.

- Computational Cost
  - Low cost: Logistic Regression, Decision Trees
  - Medium cost: Random Forest, SVM
  - High cost: Neural Networks (especially CNNs, RNNs)

Example

- This study analyzes over 6.3 million records from the Louisiana Department of Health.
- **AIM**: to identify the most effective models for predicting clinical service utilization, COVID-19 infections, and tobacco use.



Start
Raw Dataset Files
Merging Datasets
Extensive Preprocessing & Feature Engineering
Features Extraction
Label encoding
Handling Missing Data
Feature Correlation Matrix
Exploratory Data Analysis & Feature Selection
Train/Validation/Test Split
Model Optimization (Hyperparameter Tuning, Adjust Class Imbalance)
Logistic Regression
XGBoost
AdaBoost
Random Forest
Decision Tree
ANNs
Naive Bayes
Predictive Tasks:
Multinomial Clinical Service Utilization
Binary COVID-19
Binary Tobacco Use
Evaluation Metrics (Accuracy, Recall, Precision, F1 Score, AUC, Epoch vs. Loss)
Computational Efficiency (Training Time, CPU, Memory Use)
Feature Importance

Pipelines

- Data transformation steps needed to be executed multiple times in right order.
    - Pipeline class from Scikit-Learn.
- Pipeline constructor takes a list of name/estimator pairs defining a sequence of steps
- All but the last estimator must be transformers (i.e., they must have a fit_transform() method)

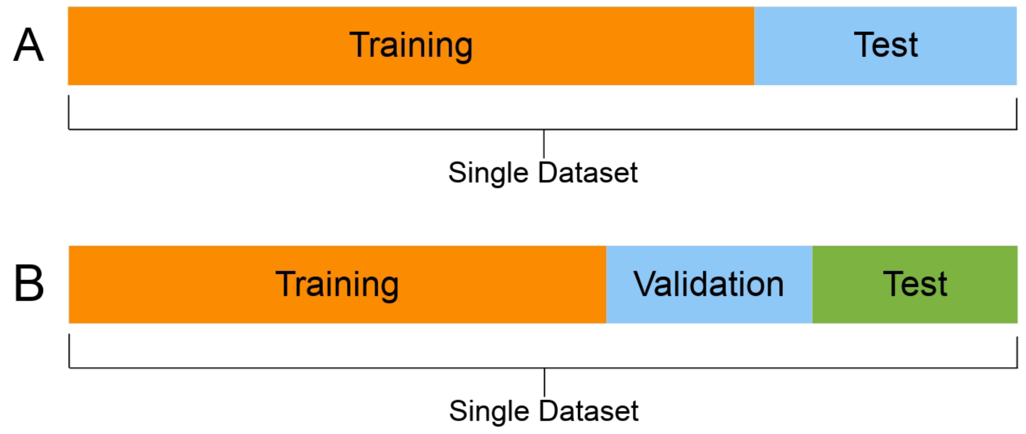
```
Pipeline([('imputer', SimpleImputer()),('scaler', StandardScaler())])
```

Train, validation and test sets

Datasets can be divided into 2 or 3 subsets:
- Training set, to train the model
- Test set, to confirm that the model works

Or
- Training set, to train the model
- Validation set, to tune the hyperparameters
- Test set, to confirm that the model works

X_train, X_test, y_train, y_test = train_test_split(…)



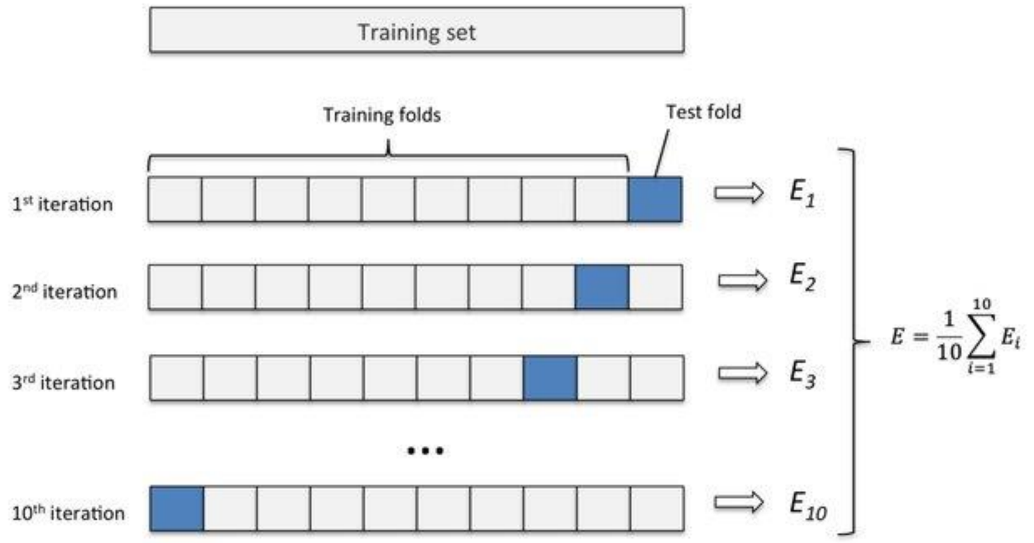
Why use a validation set?

- Fine-tuning an ML model is an iterative process of adjusting thealgorithm to improve performance on training and unseen data.
- Use a validation set (not the test set) for hyperparameter tuning toensure the model generalizes well.
- Reserve the test (holdout) set exclusively for final evaluation to obtainan unbiased measure of performance and confirm the model hasn’toverfit.

Cross-validation

- A less biased or less optimistic estimate of the model than train/test split.

- Typically implemented as K-folds cross validation, where k is the number of splits.



KFold(n_splits=10, random_state=42, shuffle=False)

Image from Introduction to Support Vector Machines and Kernel Methods, Ashfaque and Iqbal, 2019

Conclusion

- Healthcare analytics follows a clear pipeline: ***Data collection*** → ***Preprocessing & cleaning*** → ***EDA*** → ***Model building*** → ***Evaluation*** → ***Deployment.***
- High-quality healthcare data is essential for generating reliable insights. **[Quality data** → **Quality models]**.
- Understanding data structure and clinical context (features, labels, distributions, scaling, skewness) is critical before applying any ML method.
- Explainability and responsible use remain central challenges, especially due to risk, bias, privacy regulations, and model transparency.

Conclusion

- Plot Order Sequence
  - Histograms
  - – Skewness values
  - – Overall boxplots
  - – Boxplots by target
  - – Key scatter plot
  - – Pair plot (subset)
  - – Correlation heatmap