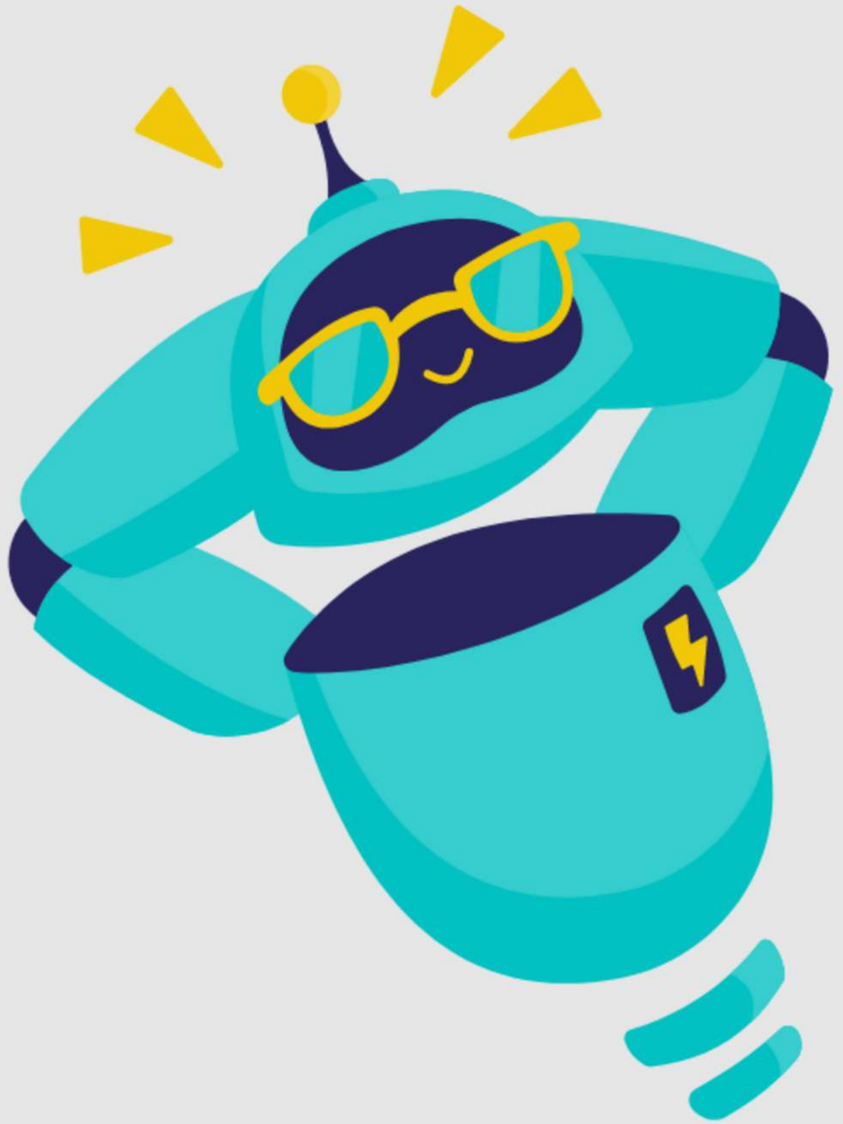
FUNDAMENTAL DATA SCIENCE TRAINING



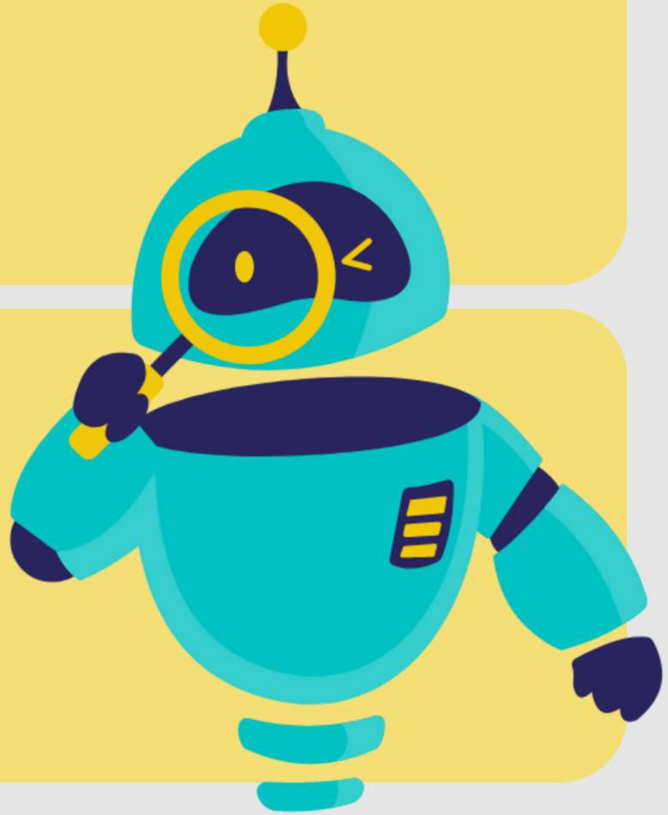
OUTLINE

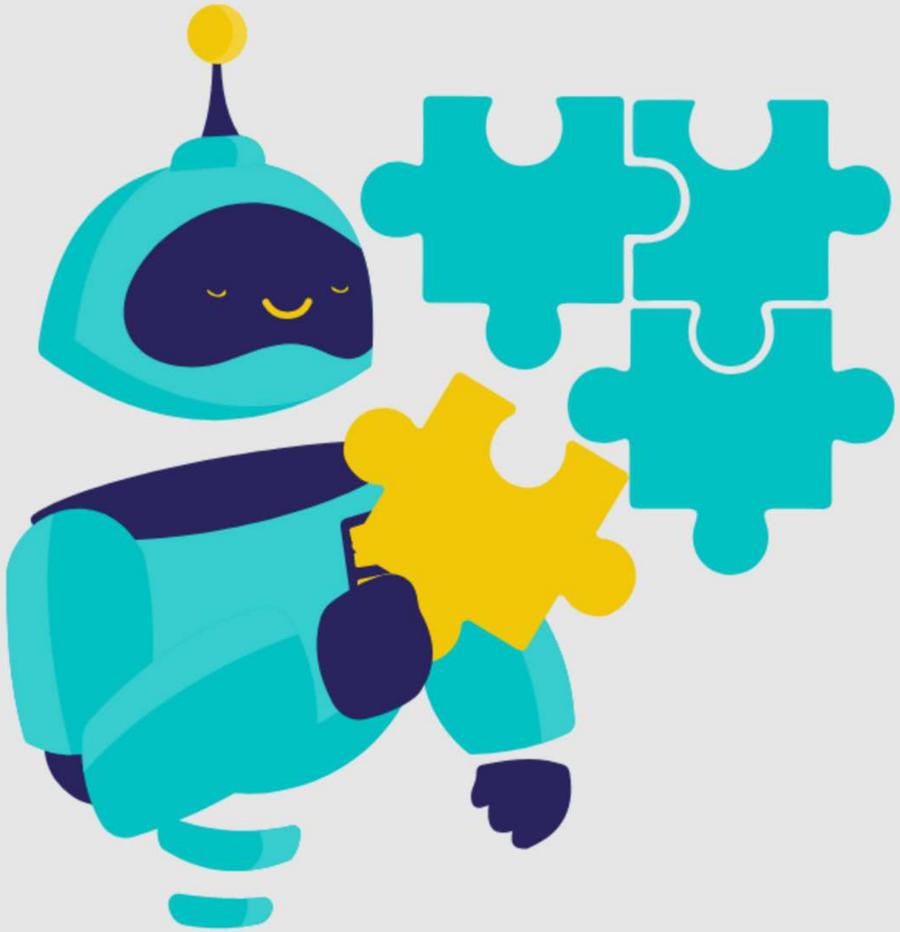
Introduction to AI, ML, and Generative AI

- What is AI
- Machine Learning
- Generative AI
- Use Cases

Machine learning pipeline

- Data Import
- EDA
- Model Building
- Model Evaluation





INTRODUCTION TO AI, ML, AND GENERATIVE AI

WHAT IS ARTIFICIAL INTELLIGENCE

- AI enables machines to mimic human intelligence.
- It learns from data to make decisions and solve problems.

EXAMPLES:

- Self-Driving Cars: AI drives safely for you
- Predictive Maintenance: predict parts needing service before they fail.
- Data Analytics: Analyze customer data to gain insights

MACHINE LEARNING (ML)

SUPERVISED LEARNING

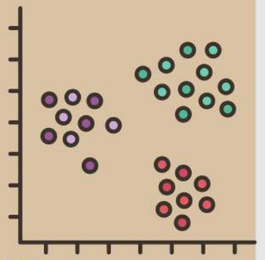
Learns from **labeled data**
e.g., speed → fuel use

Why It's Useful: Accurate predictions.

Car Industry: Predictive maintenance, quality control, lead scoring

UNSUPERVISED LEARNING

Finds patterns **without labels**



Why It's Useful: Discovers trends, automates grouping.

Car Industry: Customer segmentation, supply chain insights.

REINFORCEMENT LEARNING

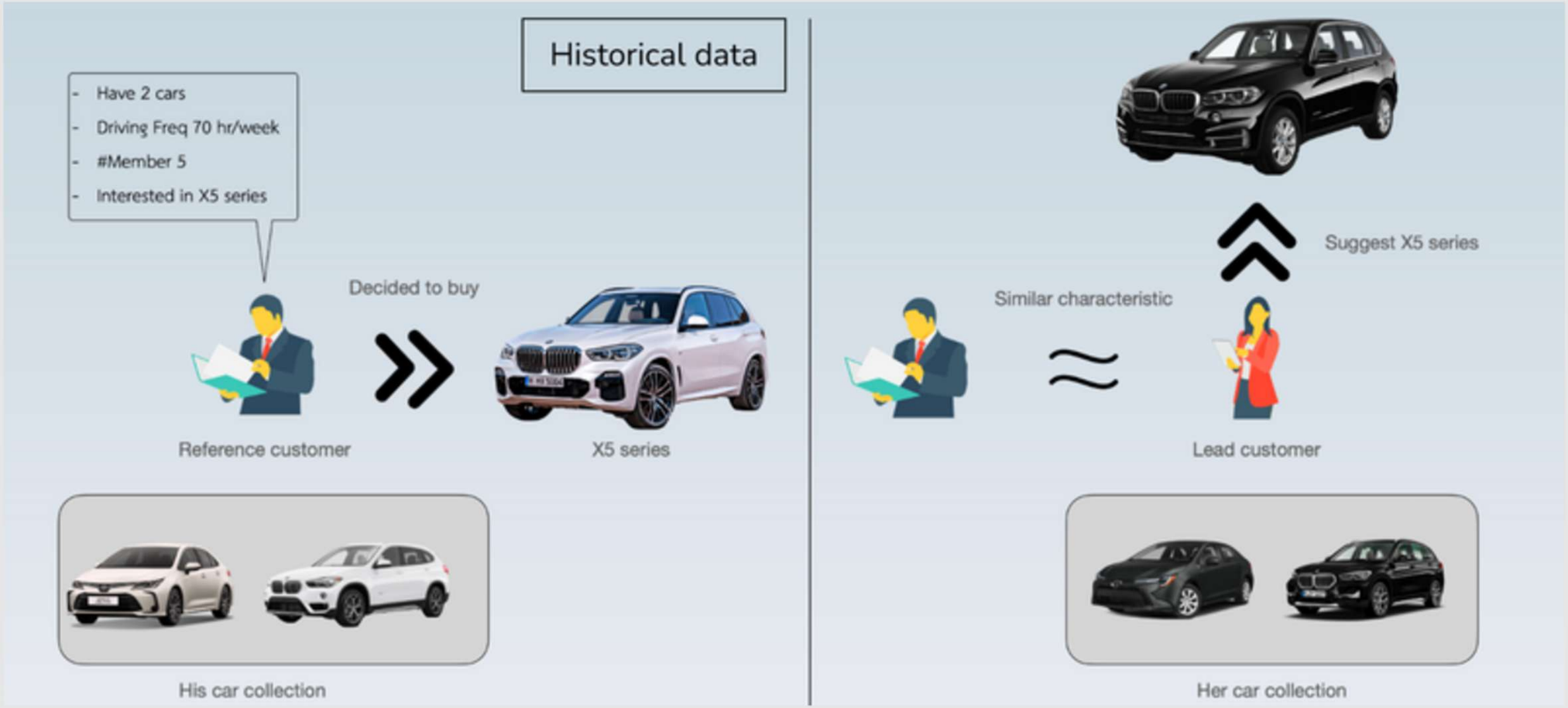
Learns by **trial and error** with **rewards.**

Why It's Useful: Adapts to changing conditions.

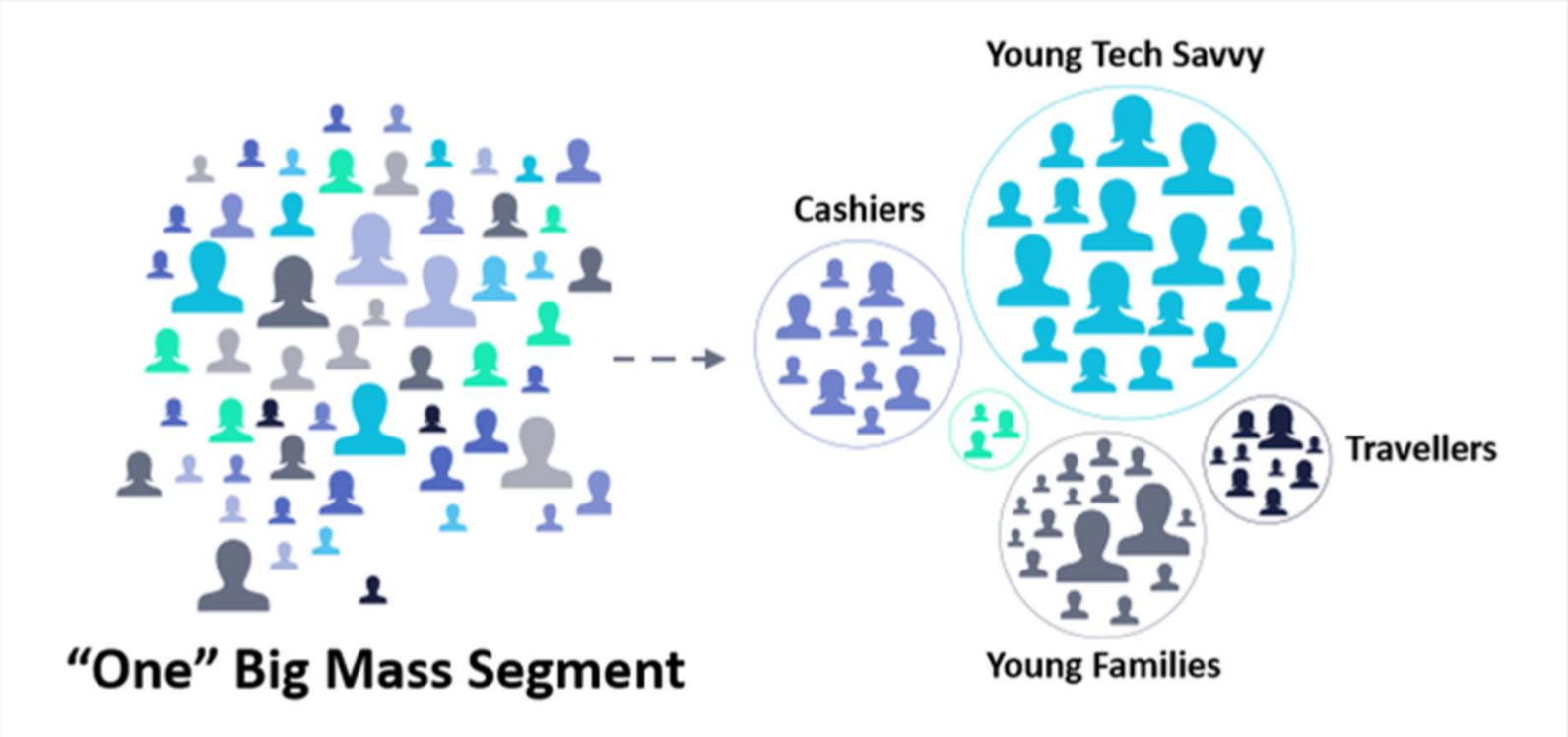
Car Industry: Self-driving cars, production optimization.

Supervised Learning Example: Lead Score

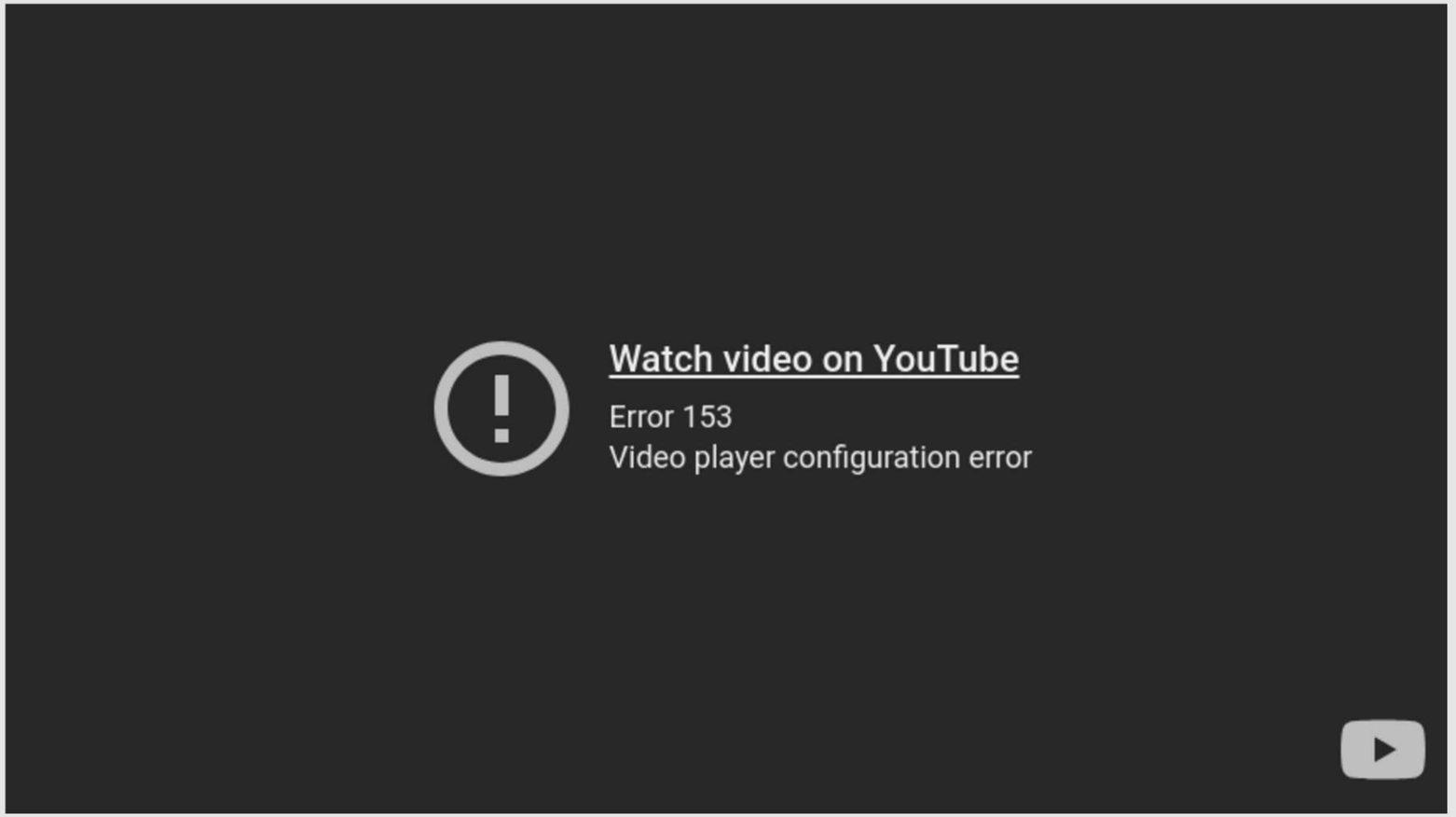
Supervised Learning Example: Next Best Offers



Unsupervised Learning Example: Customer Segmentation



Reinforcement Learning Example: Self-Driving Cars



GENERATIVE AI

USE CASES

Text generation

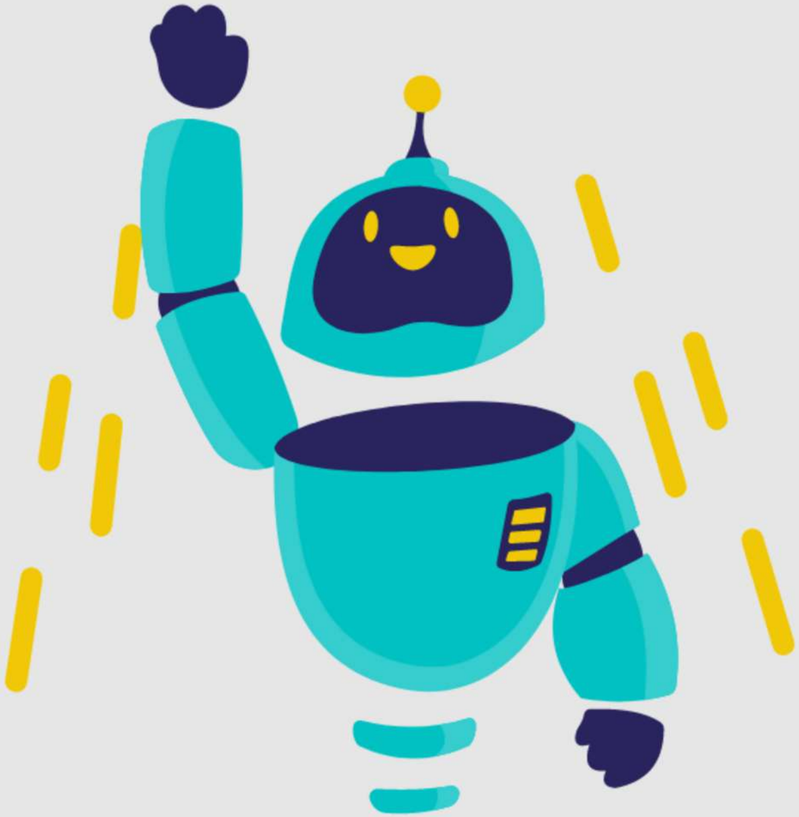
- Write personalized email, manuals, marketing content **instantly**

Image generation

- Produce marketing visuals in hours instead of weeks.

Code generation

- Write code like professionals!

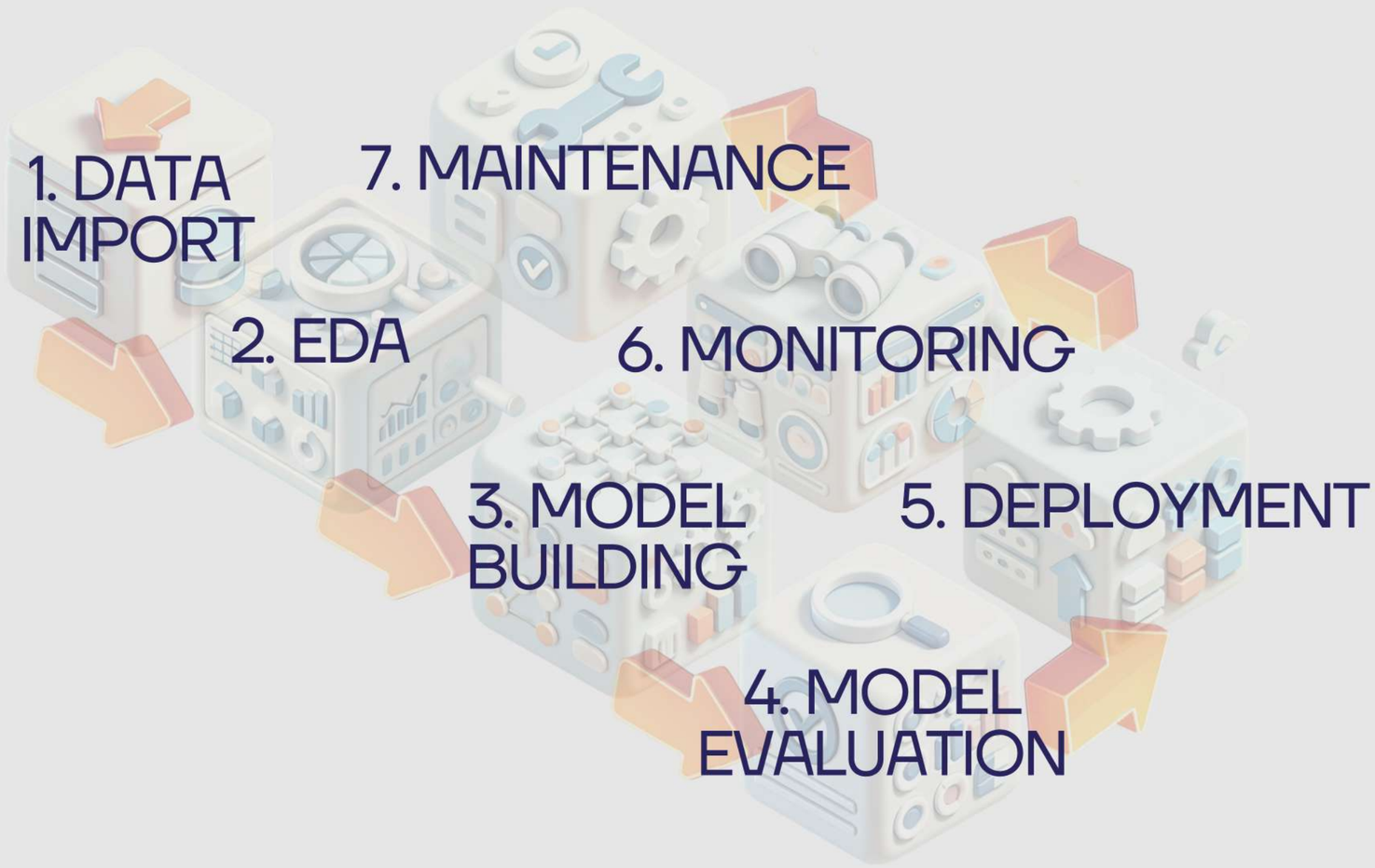


MACHINE LEARNING PIPELINE

FROM DATA TO DEPLOYMENT

4Plus Consulting

ML PIPELINE


1. DATA IMPORT
2. EDA
3. MODEL BUILDING
4. MODEL EVALUATION
5. DEPLOYMENT
6. MONITORING
7. MAINTENANCE

DATA IMPORT AND PREPROCESSING

Data table example: Purchased Model

|<----------inputs----------->| |<--target-->|

ID	Age	Gender	Marital Status	Income	Purchased Model
414428	26	Male	Single	20,000	Hilux Revo
414429	35	Female	Single	40,000	Camry
414430	30	Male	Married	50,000	Corolla Cross
414431	51	Male	Single	30,000	Yaris
414432	42	Female	Married	60,000	Yaris

rows { *records, samples,*

columns *features, attributes*

Data types

Categorical: qualitative
(you can't add/subtract/divide)

ID	Age	Gender	Marital Status	Income	Purchased Model
414428	26	Male	Single	20,000	Hilux Revo
414429	35	Female	Single	40,000	Camry
414430	30	Male	Married	50,000	Corolla Cross
414431	51	Male	Single	30,000	Yaris
414432	42	Female	Married	60,000	Yaris

Numerical: we can measure and compare

Common data types in Python

String (str): a sequence of characters (letters/numbers/punctuations), don't made for math

Integer (int): a whole number ..., -3, -2, -1, 0, 1, 2, 3, ...

Float (float): a real number (can have fraction)

Challenges

Big dataset: millions of records

Time gaps: missing data for some period of time

Transaction data: hard to integrate

Non-numeric data: text, image

Changing patterns over time

Missing and extreme values

DATA IMPORT AND PREPROCESSING

Principle of data preparation

1. Examine the dataset

- Check the size: How many rows and columns?
- Identify data types (numbers, text, dates, etc.).

2. Remove irrelevant columns

- Drop unnecessary IDs or irrelevant features.
- Remove flat variables for input (e.g. 99% male, 1% female).
- Manually remove columns unrelated to the task.

3. Decide your target *(for supervised learning)*

- Identify the column you want to predict.

4. Handle missing values

- Impute (fill) missing data or
- Remove rows/columns with too many gaps

5. Handle categorical features

- Drop columns with too many unique values (e.g., IDs)
- Convert categories to numbers (ML models need numeric inputs)

Colab example

List of prompts: docs.google.com

EXPLANATORY DATA ANALYSIS

4Plus Consulting

What is EDA?

A process to understand, clean, and summarize data before building models

EDA Tools

matplotlib

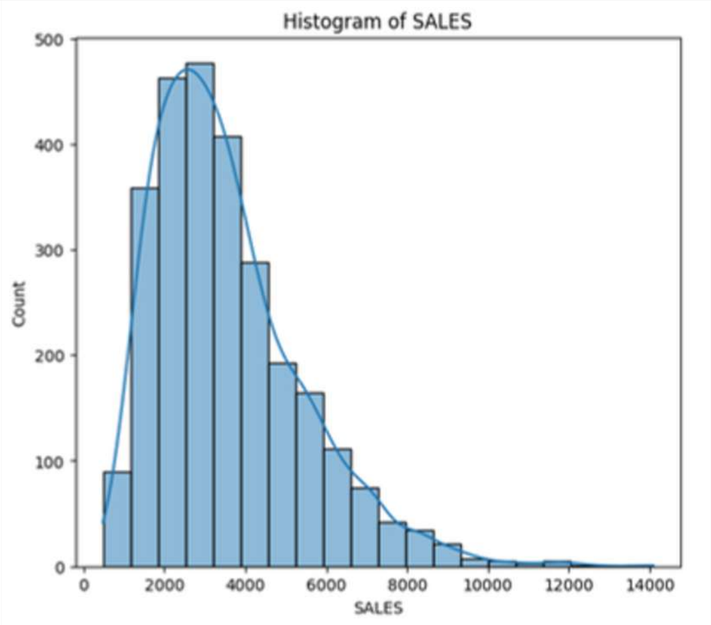
plotly

seaborn

Why EDA?

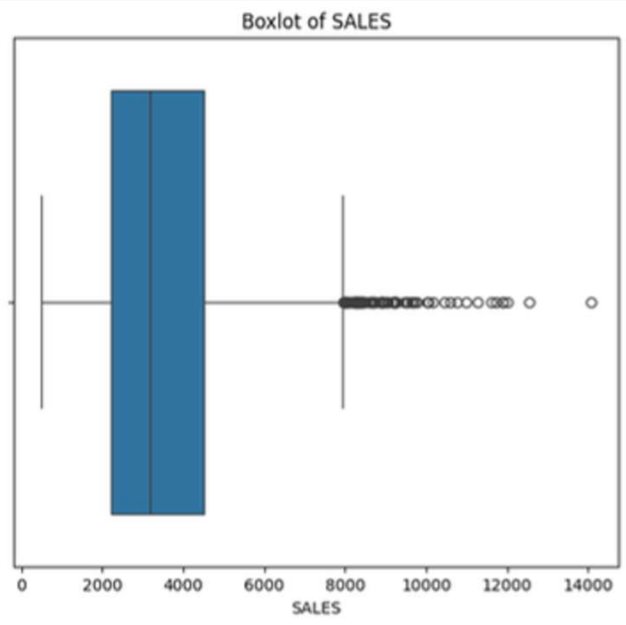
- **Understand the data**
  - What's in the dataset?
  - Are there missing or extreme values?
- **Spot patterns**
  - Identify relationships between variables.
- **Find issues**
  - Detect missing data, duplicates, or outliers.
- **Prepare for modeling**
  - Decide what features to keep, remove, or transform.

Common Types of plot in EDA



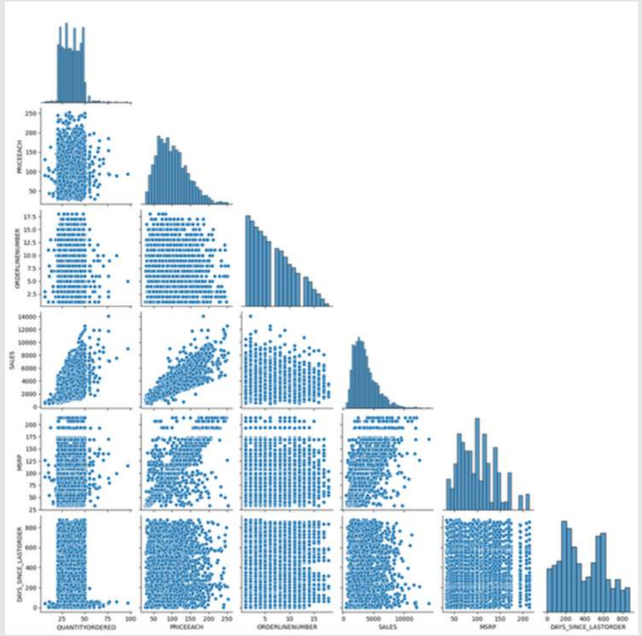
Histogram

Understand data distribution



Box Plot

Spot outliers and visualize: Q1-Q3, Median, Range, and Skewness.



Pair Plot

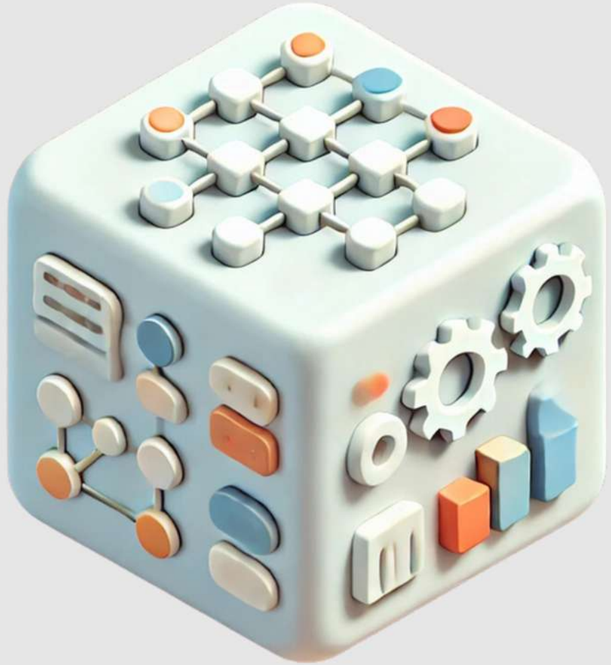
Identify pairwise correlation and trends

source: kaggle.com

Colab example

List of prompts: docs.google.com

MODEL BUILDING



What is a model

A model is a **representation of the data** it has learned during training.

Purpose: We use trained model to **predict unseen data** accurately and efficiently.

A model is assigned **two main types of tasks**:

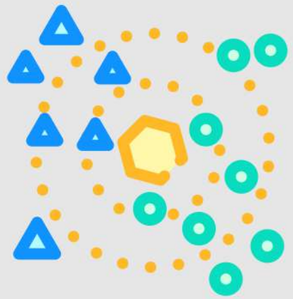
Regression: Predict numerical outputs (e.g., car prices).

Classification: Predict categories (e.g., spam or not spam)

How many models are there?

Regression models

- linear regression
- logistic regression



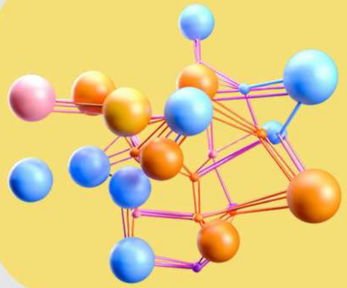
k-Nearest Neighbors

Tree-based models

- decision tree
- random forest
- XGBoost

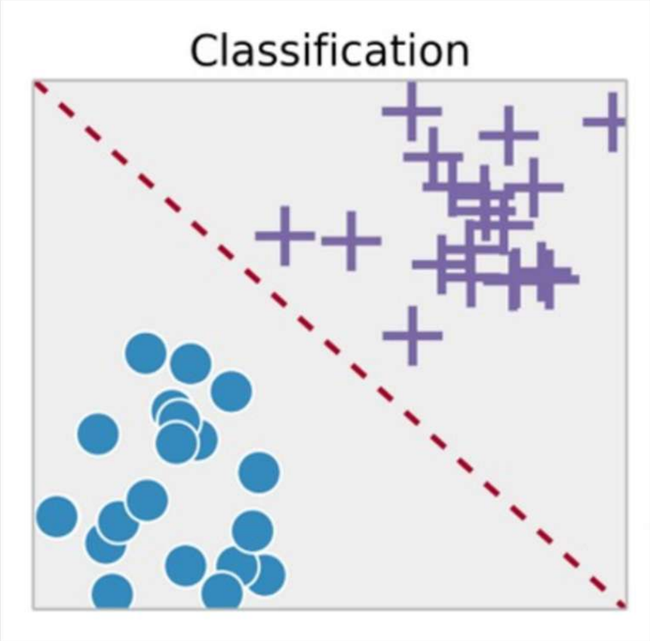
Bayesian Learning

Support Vector Machine (SVM)

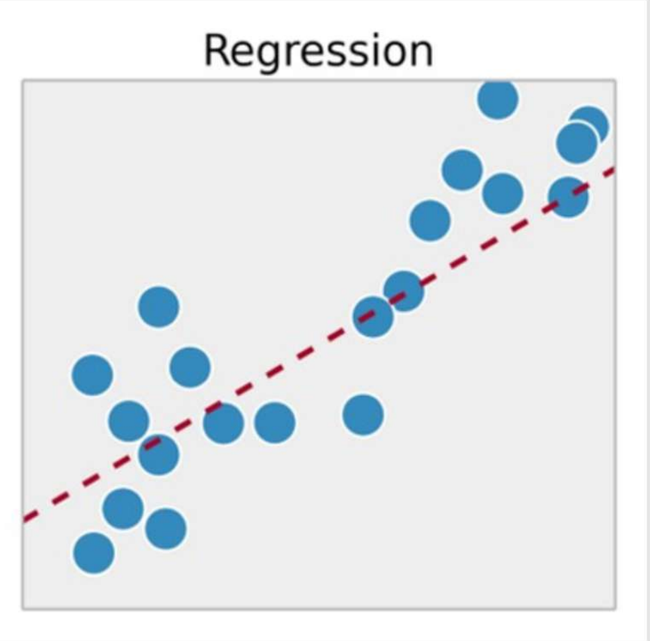


Artificial Neural Netwroks (ANN)

Classification vs. Regression



- Classification = grouping
- Predict which group the data belongs to
- E.g. cats/dogs, buy/don't buy,



- Regression predicts number
- E.g., stock price, store sales, customer spendings

image source

Overfitting

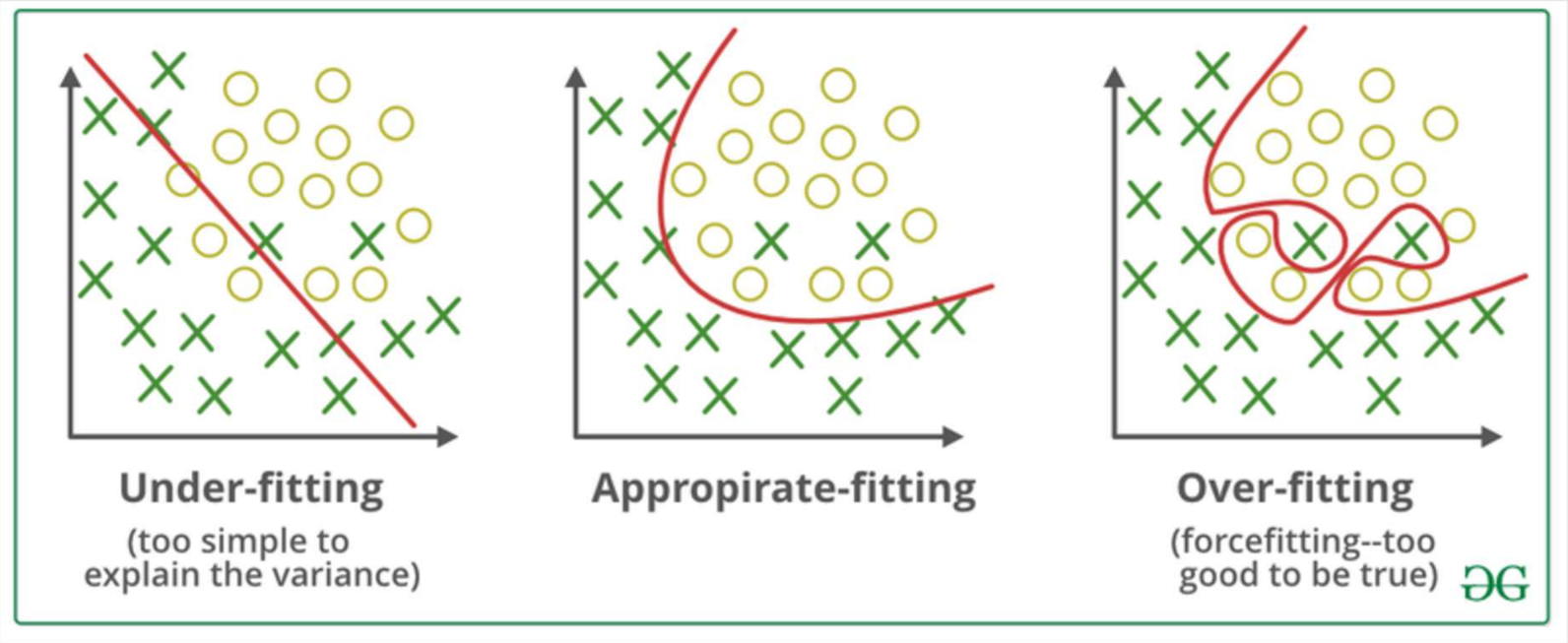
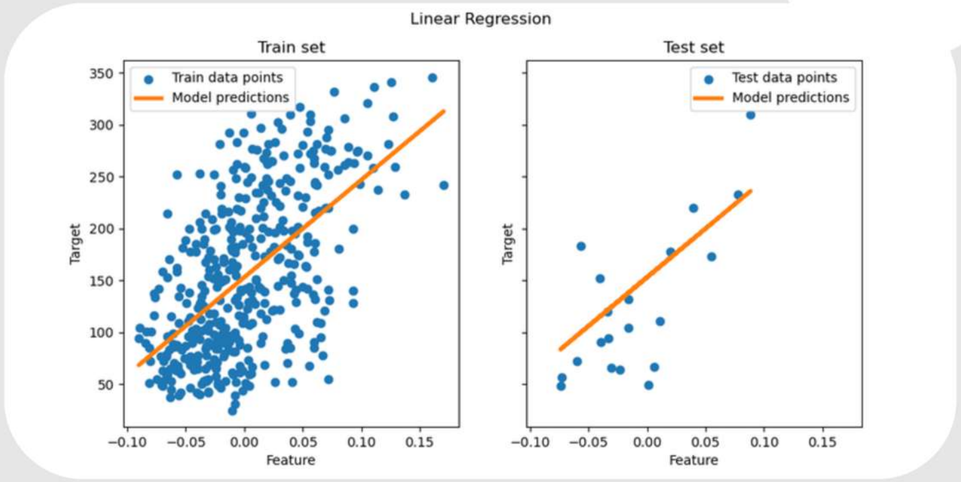


image source

Regression models



What for?

Predicting numerical outputs (e.g., predicting car mileage).

How they work?

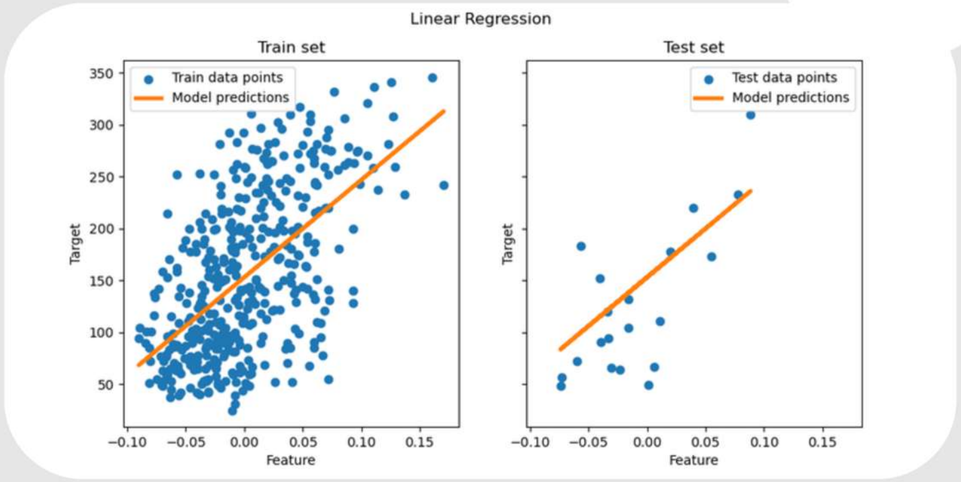
Fit a line or curve to data based on relationships between input (X) and output (Y).

Limitations

Sensitive to outliers (unusual data points that stand out from the rest)

read more: scikit-learn.org

Regression models



What for?

Predicting numerical outputs (e.g., predicting car mileage).

How they work?

Fit a line or curve to data based on relationships between input (X) and output (Y).

Limitations

Sensitive to outliers (unusual data points that stand out from the rest)

read more: scikit-learn.org

MODEL BUILDING

LinearRegression

```
class sklearn.linear_model.LinearRegression(*, fit_intercept=True,
copy_X=True, n_jobs=None, positive=False)
```
[source]

Ordinary least squares Linear Regression.

LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Parameters:

fit_intercept : *bool, default=True*

Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

copy_X : *bool, default=True*

If True, X will be copied; else, it may be overwritten.

Code example

```
import numpy as np
from sklearn.linear_model import LinearRegression
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
# y = 1 * x_0 + 2 * x_1 + 3
y = np.dot(X, np.array([1, 2])) + 3
reg = LinearRegression().fit(X, y)
reg.score(X, y)
reg.coef_
reg.intercept_
reg.predict(np.array([[3, 5]]))
```

read more: scikit-learn.org

MODEL BUILDING

LogisticRegression

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *,
dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1,
class_weight=None, random_state=None, solver='lbfgs', max_iter=100,
multi_class='deprecated', verbose=0, warm_start=False, n_jobs=None,
l1_ratio=None)
```
[source]

Logistic Regression (aka logit, MaxEnt) classifier.

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices

Parameters:

penalty : *{'l1', 'l2', 'elasticnet', None}, default='l2'*

Specify the norm of the penalty:

- `None`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

Code example

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

X, y = load_iris(return_X_y=True)
clf = LogisticRegression(random_state=0).fit(X, y)
clf.predict(X[:2, :])
clf.predict_proba(X[:2, :])
clf.score(X, y)
```

read more: scikit-learn.org

Decision tree

What for?

Both classification and regression.

Intuitive for complex, non-linear relationships.

How they work?

Break data into smaller, decision-based groups.

Limitations

Can overfit on small datasets.

We often use 'ensemble models' instead

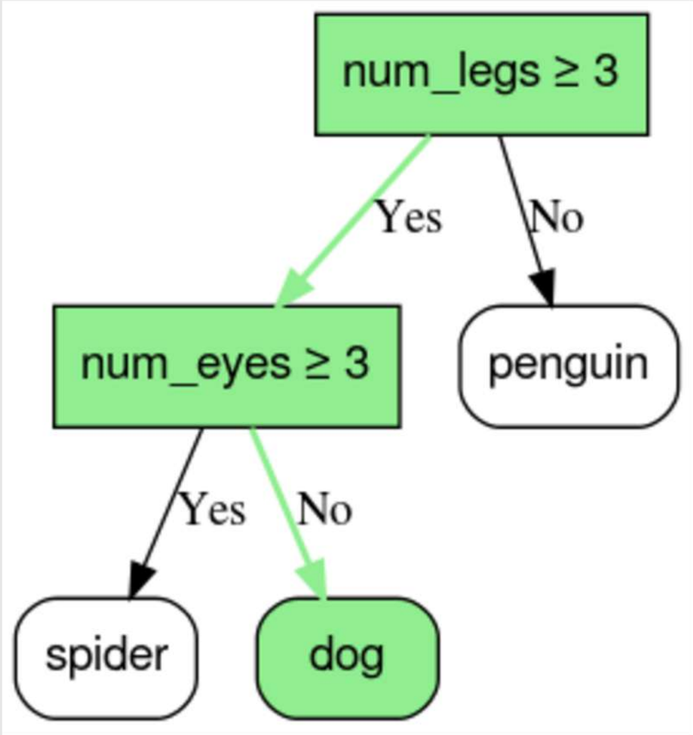


image source: developers.google.com

Important Parameters in Decision Tree

- Splitting measure (criterion) : gini / entropy
- Maximum depth : ~5-10 (depend on number of feature)
- Maximum leaf nodes : depend on number of class (target) and feature
- Minimum sample split : 5 – 20% (depend on number of data)
- Minimum impurity decrease : (default 0)
- Pruning adjustment (ccp_alpha) : 0.001 - 0.01 (depend on size of tree)

Ensemble models: Random Forest

What for?

- Handle complex data for classification and regression tasks.
- Reduces overfitting seen in single Decision Trees.

How it works?

- Builds multiple Decision Trees using random subsets of data and features (Bagging),
- predict by using average (for regression) or majority votes (for classification).

Limitations

- Can be slow for very large datasets.
- Less interpretable than decision tree

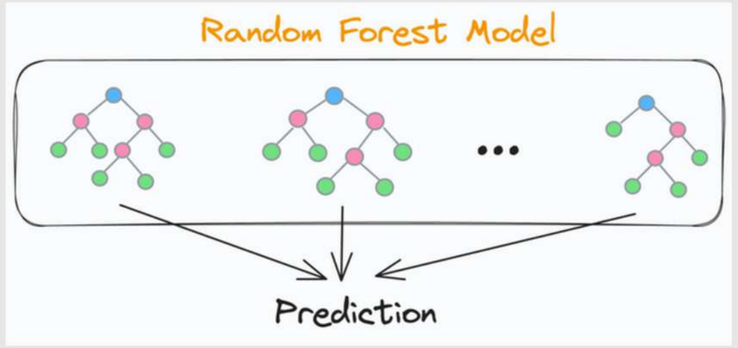


image source: blog.dailydoseofds.com

MODEL BUILDING

RandomForestClassifier

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100,
*, criterion='gini', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0,
max_features='sqrt', max_leaf_nodes=None,
min_impurity_decrease=0.0, bootstrap=True, oob_score=False,
n_jobs=None, random_state=None, verbose=0, warm_start=False,
class_weight=None, ccp_alpha=0.0, max_samples=None,
monotonic_cst=None)
```

[source]

read more: scikit-learn.org

RandomForestRegressor

```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *,
criterion='squared_error', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=1.0,
max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True,
oob_score=False, n_jobs=None, random_state=None, verbose=0,
warm_start=False, ccp_alpha=0.0, max_samples=None,
monotonic_cst=None)
```

[source]

read more: scikit-learn.org

Code example

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4,
                           n_informative=2, n_redundant=0,
                           random_state=0, shuffle=False)
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X, y)
print(clf.predict([[0, 0, 0, 0]]))
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
X, y = make_regression(n_features=4, n_informative=2,
                       random_state=0, shuffle=False)
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(X, y)
print(regr.predict([[0, 0, 0, 0]]))
```

MODEL BUILDING

Ensemble models: Gradient Boosting

What for?

- High-accuracy predictions for structured/tabular data.
- Optimized for performance in competitions and real-world use cases.

How it works?

- Builds trees sequentially, correcting errors from previous trees (Boosting).
- Assigns higher weights to misclassified data to improve learning.

Limitations

- Can overfit on small datasets without careful tuning.
- Requires hyperparameter tuning for best results.

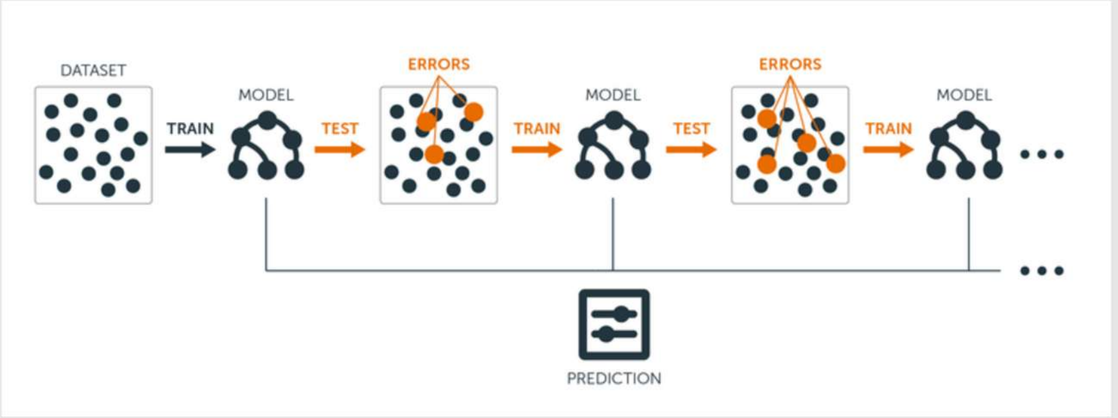


image source: iq.opengenus.org

MODEL BUILDING

GradientBoostingClassifier

```
class sklearn.ensemble.GradientBoostingClassifier(*,
loss='log_loss', learning_rate=0.1, n_estimators=100,
subsample=1.0, criterion='friedman_mse', min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3,
min_impurity_decrease=0.0, init=None, random_state=None,
max_features=None, verbose=0, max_leaf_nodes=None,
warm_start=False, validation_fraction=0.1, n_iter_no_change=None,
tol=0.0001, ccp_alpha=0.0) #                                [source]
```

read more: scikit-learn.org

GradientBoostingRegressor

```
class sklearn.ensemble.GradientBoostingRegressor(*,
loss='squared_error', learning_rate=0.1, n_estimators=100,
subsample=1.0, criterion='friedman_mse', min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3,
min_impurity_decrease=0.0, init=None, random_state=None,
max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None,
warm_start=False, validation_fraction=0.1, n_iter_no_change=None,
tol=0.0001, ccp_alpha=0.0)                                  [source]
```

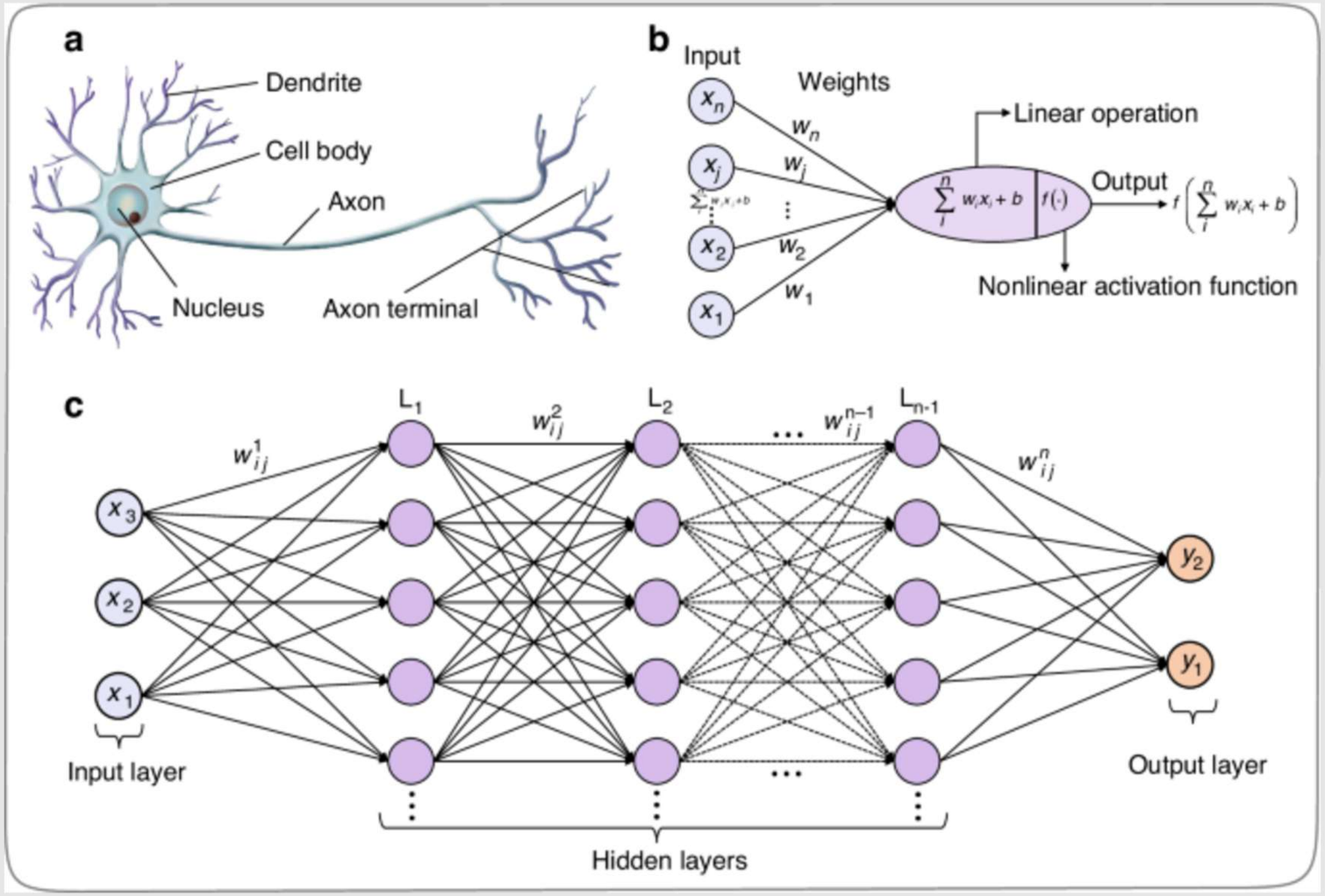
read more: scikit-learn.org

Code example

```
from sklearn.datasets import make_hastie_10_2
from sklearn.ensemble import GradientBoostingClassifier
X, y = make_hastie_10_2(random_state=0)
X_train, X_test = X[:2000], X[2000:]
y_train, y_test = y[:2000], y[2000:]
clf = GradientBoostingClassifier(n_estimators=100,
learning_rate=1.0,
    max_depth=1, random_state=0).fit(X_train, y_train)
clf.score(X_test, y_test)
```

```
from sklearn.datasets import make_regression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
X, y = make_regression(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0)
reg = GradientBoostingRegressor(random_state=0)
reg.fit(X_train, y_train)
reg.predict(X_test[1:2])
reg.score(X_test, y_test)
```

Deep Learning



Deep Learning Usecase:
Image Classification (CNN Explainer)

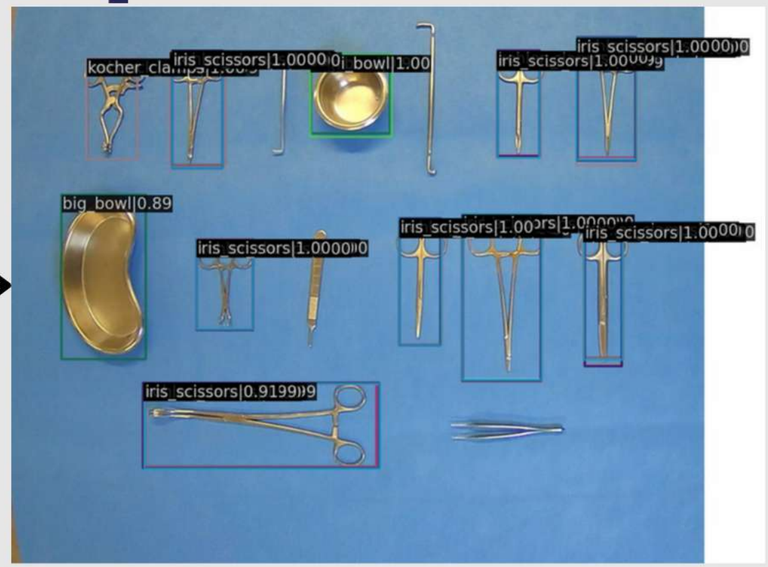
CNN EXPLAINER

Deep Learning Usecase:

Object detection and segmentation

Input image

object detection



segmentation

Deep Learning Usecase:
Violent Parking

MOD

Ar

Wh

•

Ho

•

Lin

•

•

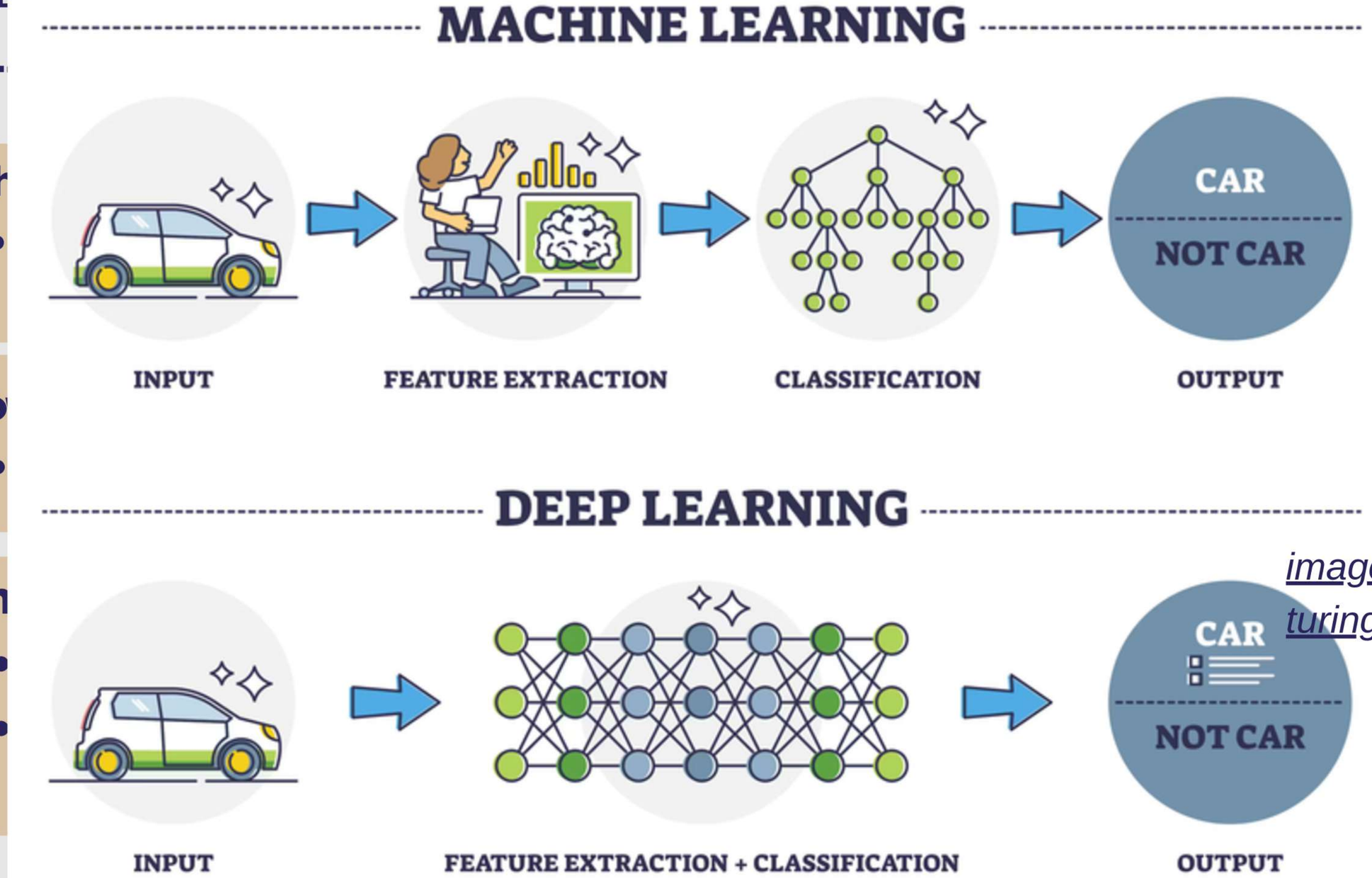


image source: turing.com

MODEL BUILDING

MLPClassifier

```
class sklearn.neural_network.MLPClassifier(hidden_layer_sizes=
(100,), activation='relu', *, solver='adam', alpha=0.0001,
batch_size='auto', learning_rate='constant',
learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True,
random_state=None, tol=0.0001, verbose=False, warm_start=False,
momentum=0.9, nesterovs_momentum=True, early_stopping=False,
validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08,
n_iter_no_change=10, max_fun=15000)
```

[source]

read more: scikit-learn.org

MLPRegressor

```
class sklearn.neural_network.MLPRegressor(hidden_layer_sizes=
(100,), activation='relu', *, solver='adam', alpha=0.0001,
batch_size='auto', learning_rate='constant',
learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True,
random_state=None, tol=0.0001, verbose=False, warm_start=False,
momentum=0.9, nesterovs_momentum=True, early_stopping=False,
validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08,
n_iter_no_change=10, max_fun=15000)
```

[source]

read more: scikit-learn.org

Code example

```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
X, y = make_classification(n_samples=100, random_state=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
                                                    random_state=1)
clf = MLPClassifier(random_state=1, max_iter=300).fit(X_train, y_train)
clf.predict_proba(X_test[:1])
clf.predict(X_test[:5, :])
clf.score(X_test, y_test)
```

```
from sklearn.neural_network import MLPRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
X, y = make_regression(n_samples=200, random_state=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,

random_state=1)
regr = MLPRegressor(random_state=1,
max_iter=500).fit(X_train, y_train)
regr.predict(X_test[:2])
regr.score(X_test, y_test)
```

Model customization: hyperparameter tuning

What Are Hyperparameters?

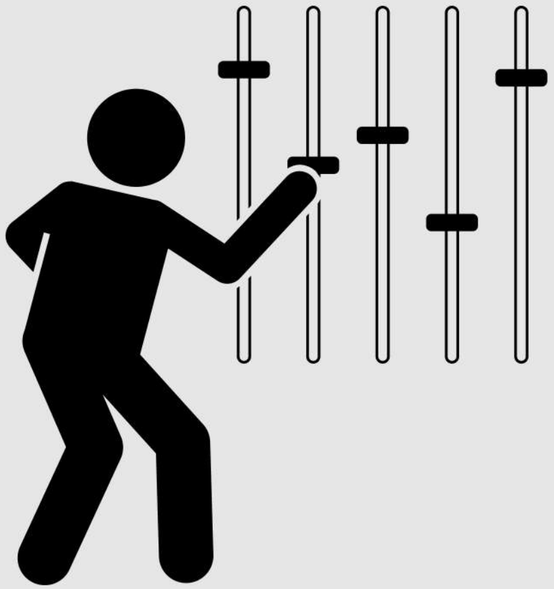
- Adjustable settings that control how models learn.
- Examples: Learning rate, tree depth, number of neurons.

Why Tuning Matters:

- Adjusting hyperparameters improves model performance.

How to Tune Hyperparameters?

- Grid Search: Test all possible combinations.
- Random Search: Test random combinations to save time.



MODEL EVALUATION

Model Evaluation: Why, How, and What's Good Enough?

Why evaluate models?

- Select the Best Model: Choose the one with the highest performance.
- Test for Production Readiness: Ensure the model is reliable and won't make bad decisions.

How to evaluate models?

1. Split the data
   - Training Set (80%): Train the model.
   - Validation Set (10%): Tune hyperparameters and choose the best model.
   - Test Set (10%): Final performance evaluation.

2. Use performance metrics
   - Classification:
     - Accuracy
     - Precision
     - Recall
     - F1-Score
   - Regression
     - L1 (Mean Absolute Error)
     - L2 (Mean Squared Error)

Confusion Matrix

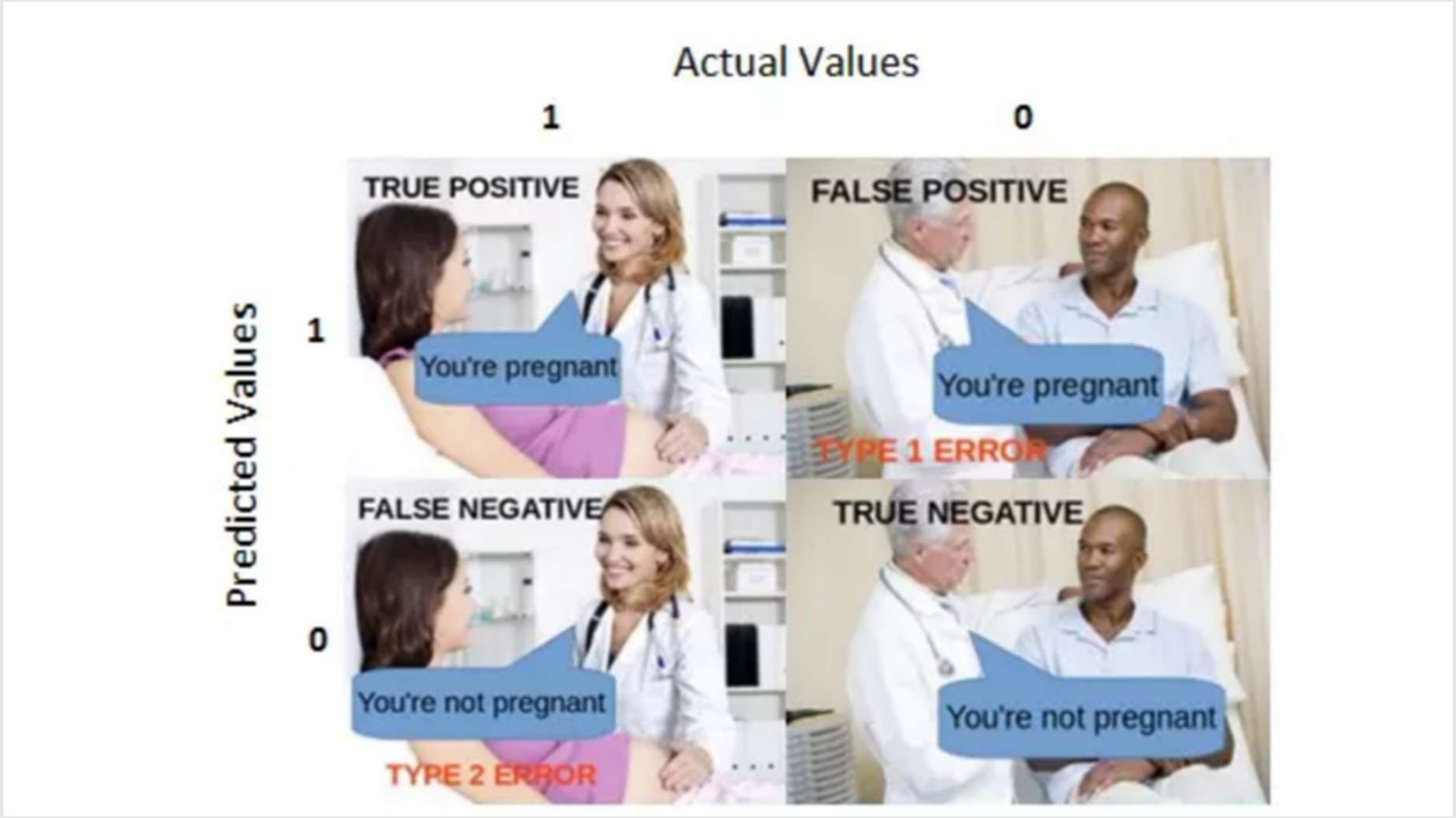


image source

Confusion Matrix

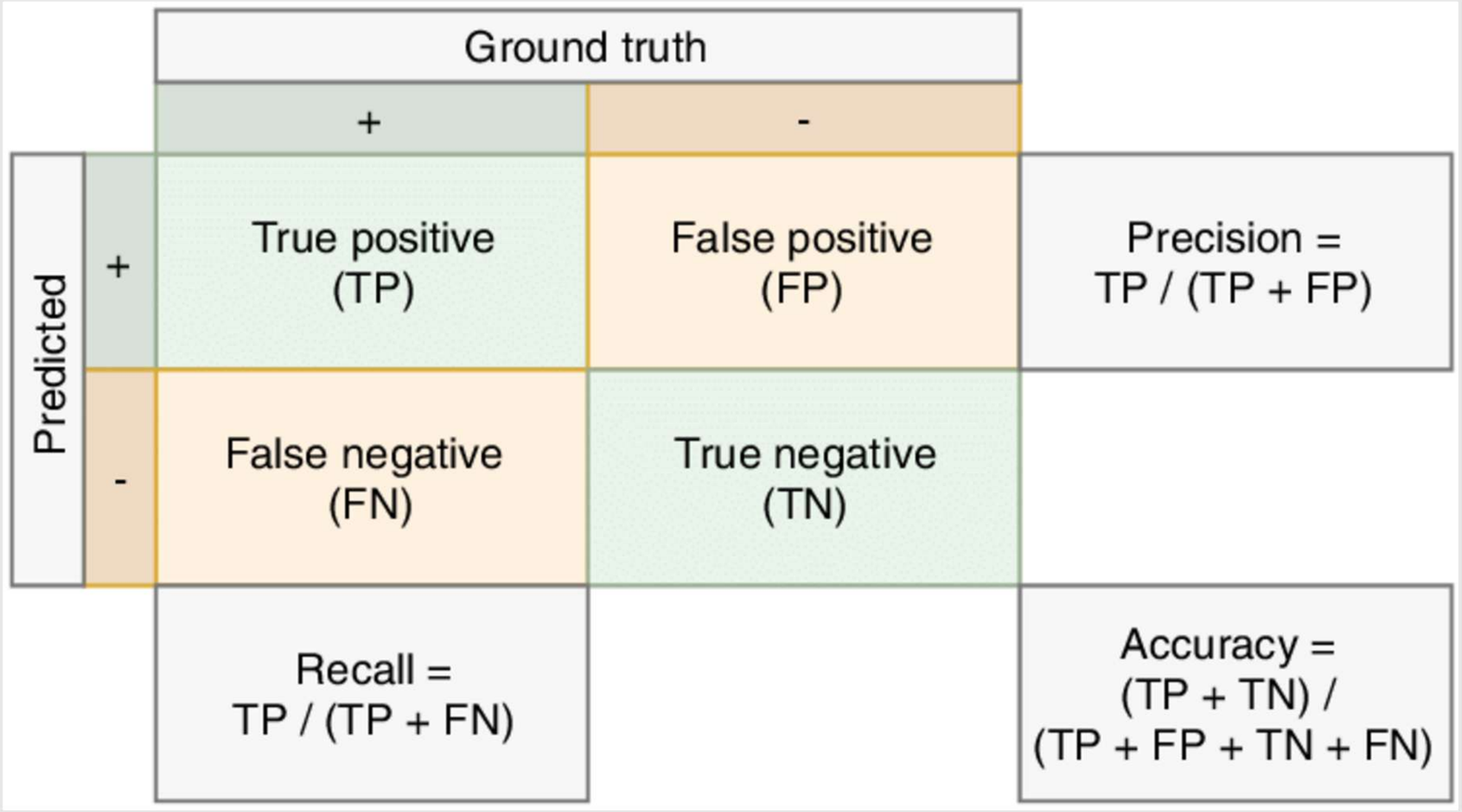
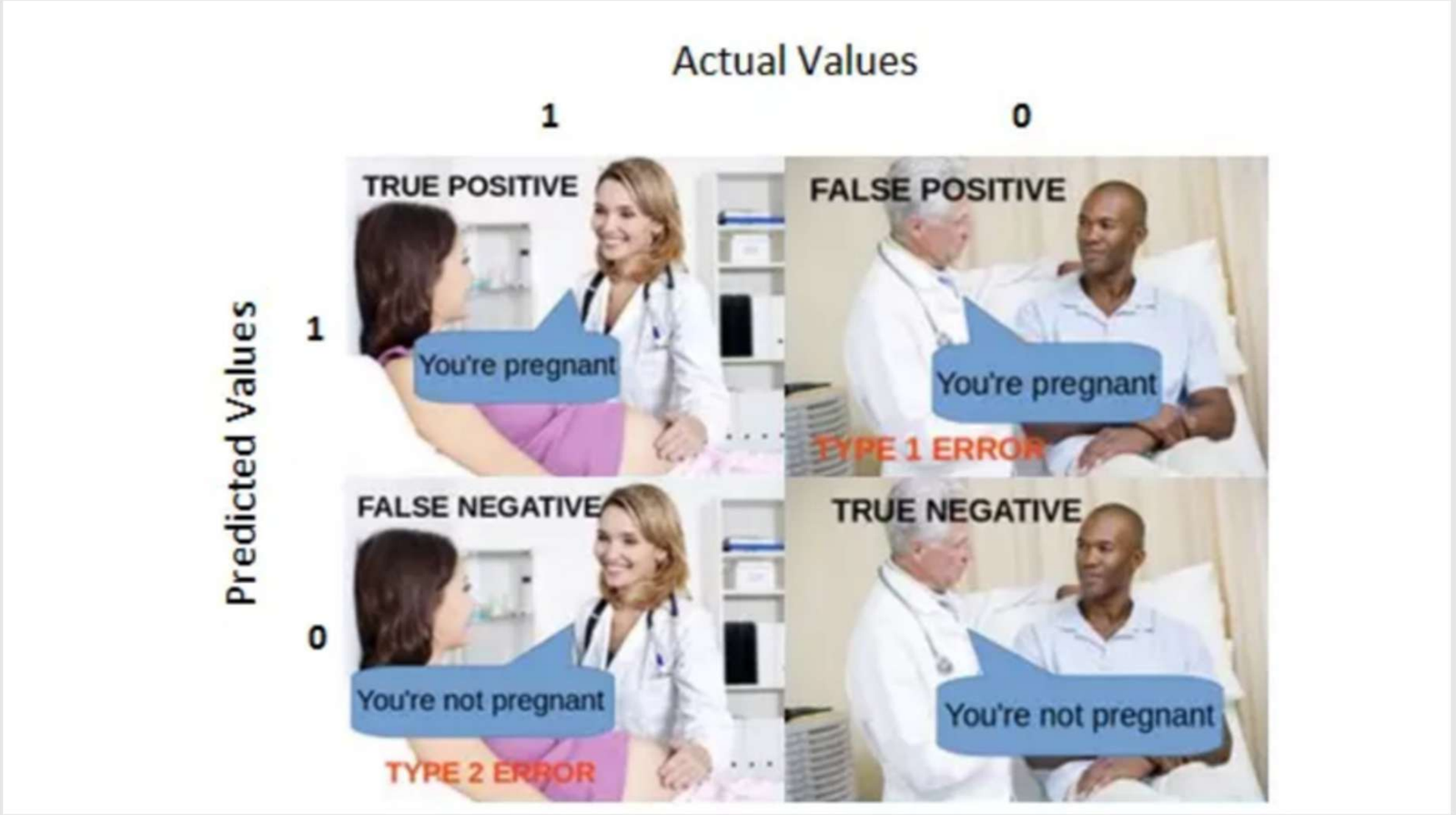
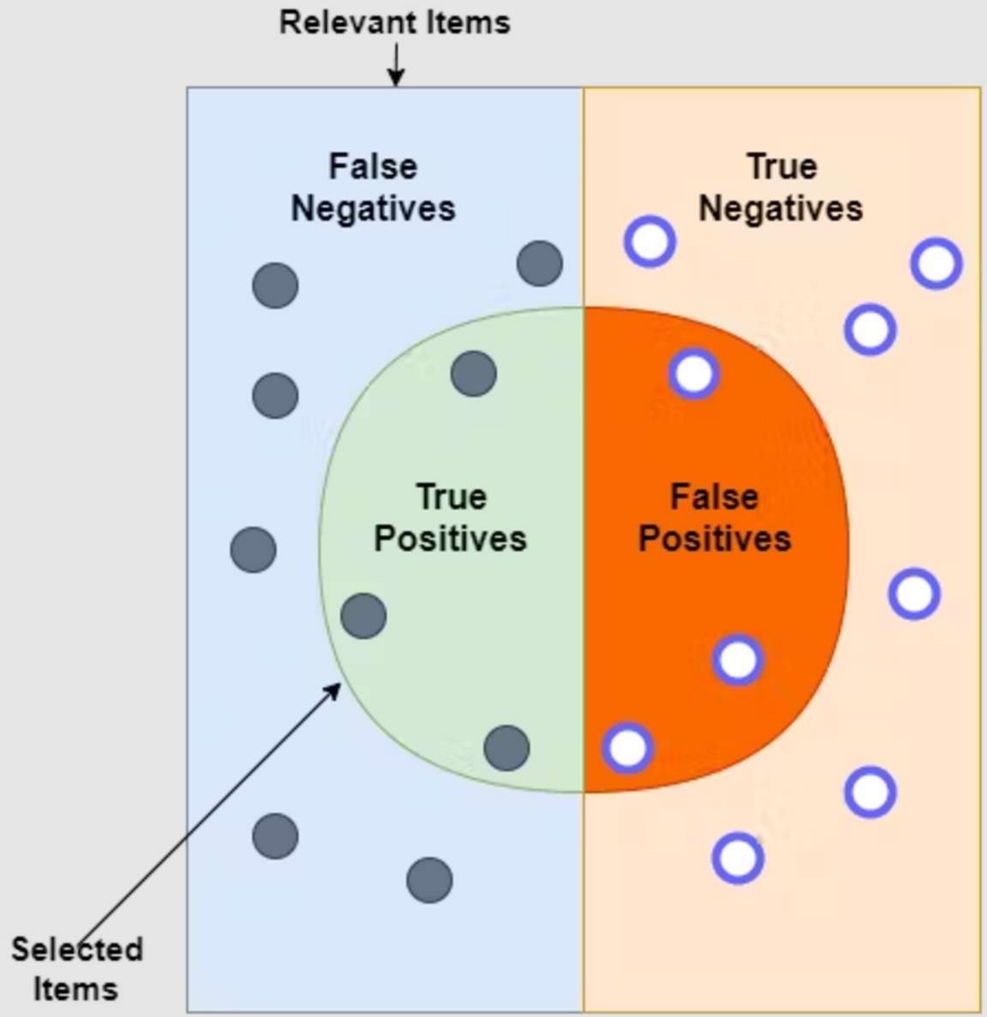


image source

Confusion Matrix



Precision and Recall



Precision = How many selected items are relevant?

Recall = How many relevant items are selected?

F1-score

$$\text{F1 Score} = \frac{2}{\left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}\right)}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Model Evaluation: Why, How, and What's Good Enough?

How much is good enough?

It Depends: How much error can you tolerate?

- In critical tasks (e.g., medical diagnosis): Near-perfect accuracy is required.
- In less sensitive tasks (e.g., ad recommendations): Some error is acceptable.

THANK YOU

WWW.REALLYGREATSITE.COM