Lecture 3:
Supervised Learning

https://github.com/kaopanboonyuen/SC310005_ArtificialIntelligence_2025s1

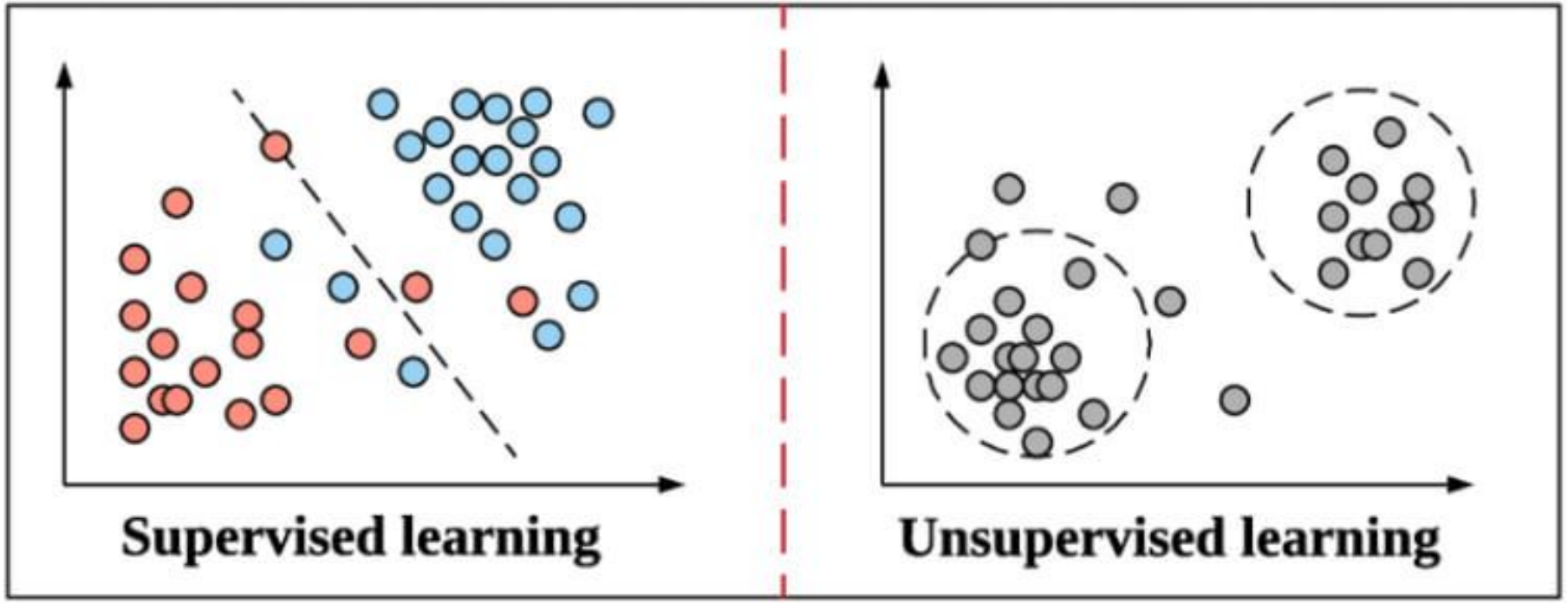
Teerapong Panboonyuen
https://kaopanboonyuen.github.io

Introduction to Supervised Learning

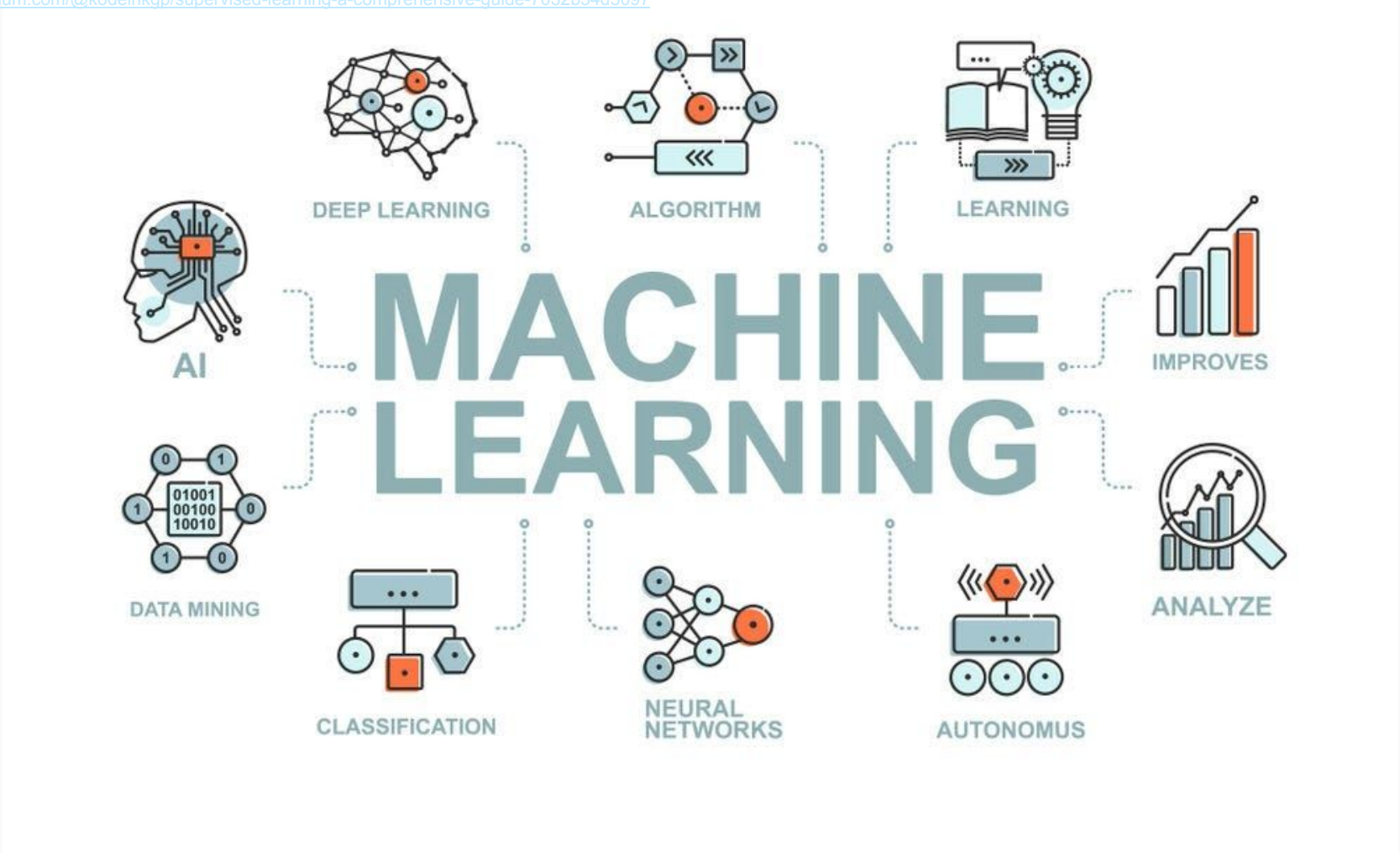
What is Supervised Learning?

- **Definition**: Supervised learning is a machine learning technique where the model learns from labeled data to make predictions.
- **Goal**: The model tries to learn the relationship between input features and the output label (target variable).

https://www.superannotate.com/blog/supervised-learning-and-other-machine-learning-tasks

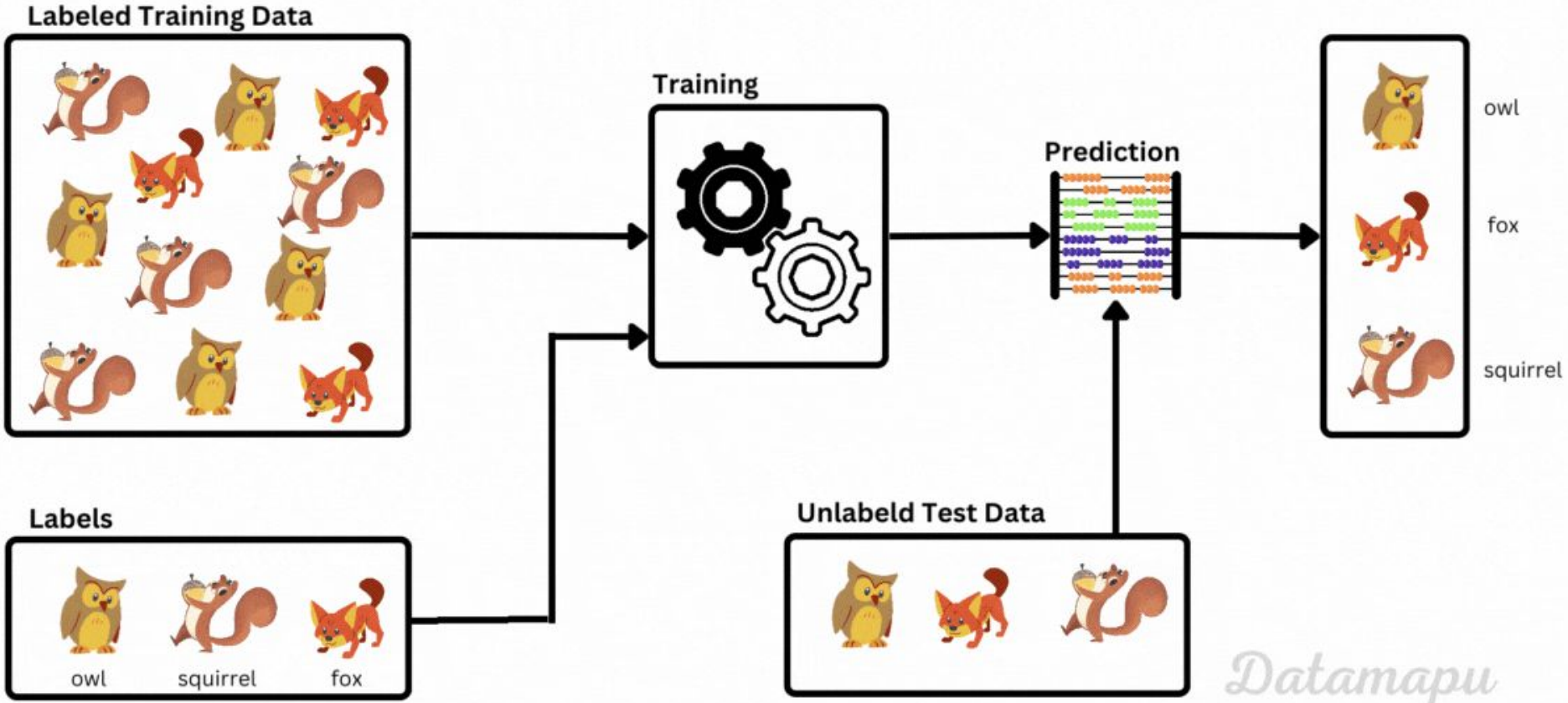


https://medium.com/@kodeinkgp/supervised-learning-a-comprehensive-guide-7032b34d5097



https://datamapu.com/posts/ml_concepts/supervised_unsupervised

Supervised Learning



Introduction to Supervised Learning

Example in Practice:

- **Predicting Cancer Diagnosis**: Given a set of features such as age, gender, and medical history, we predict whether a patient has cancer (binary classification: "Cancer" or "No Cancer").

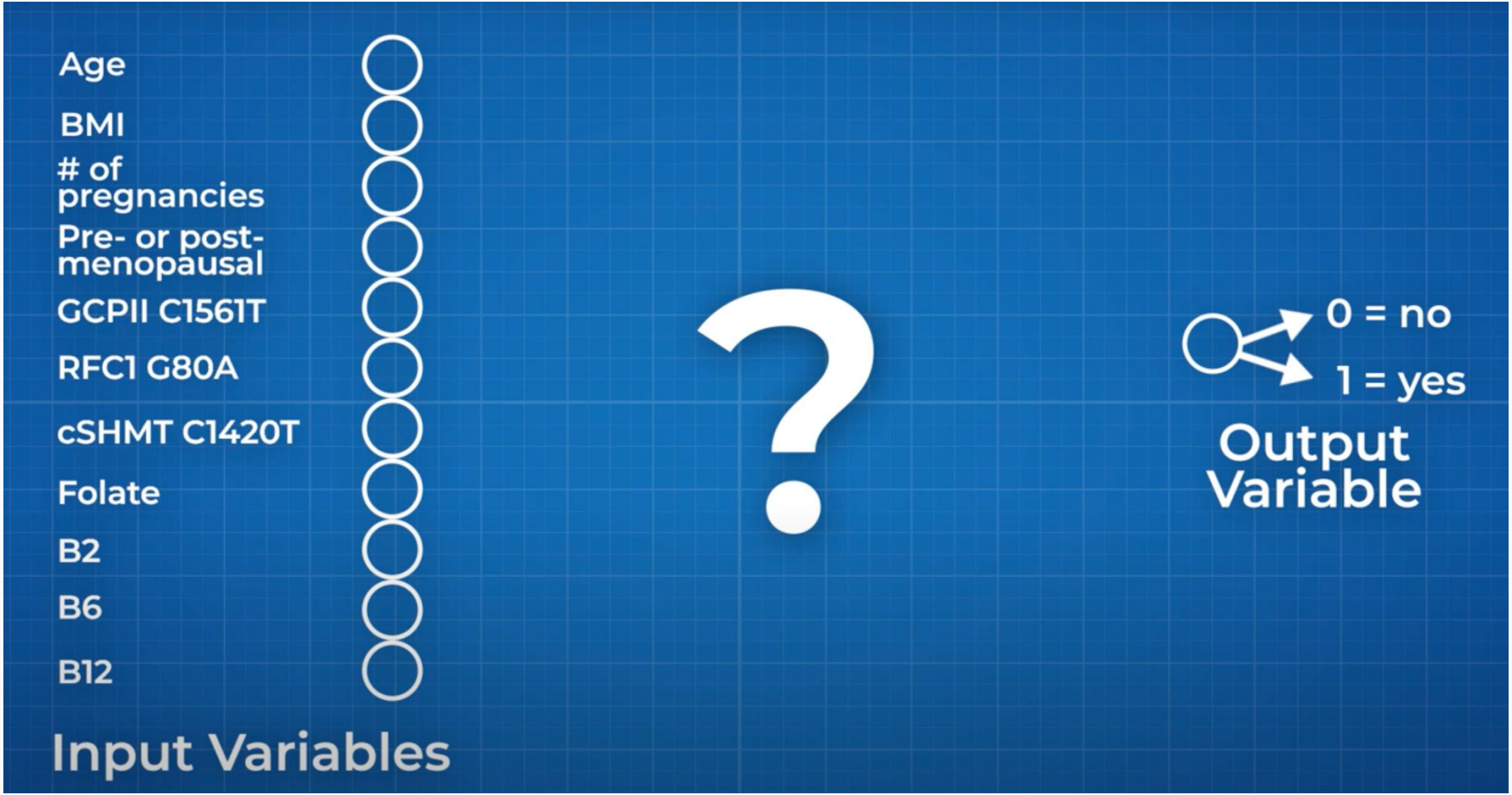
Key Terms:

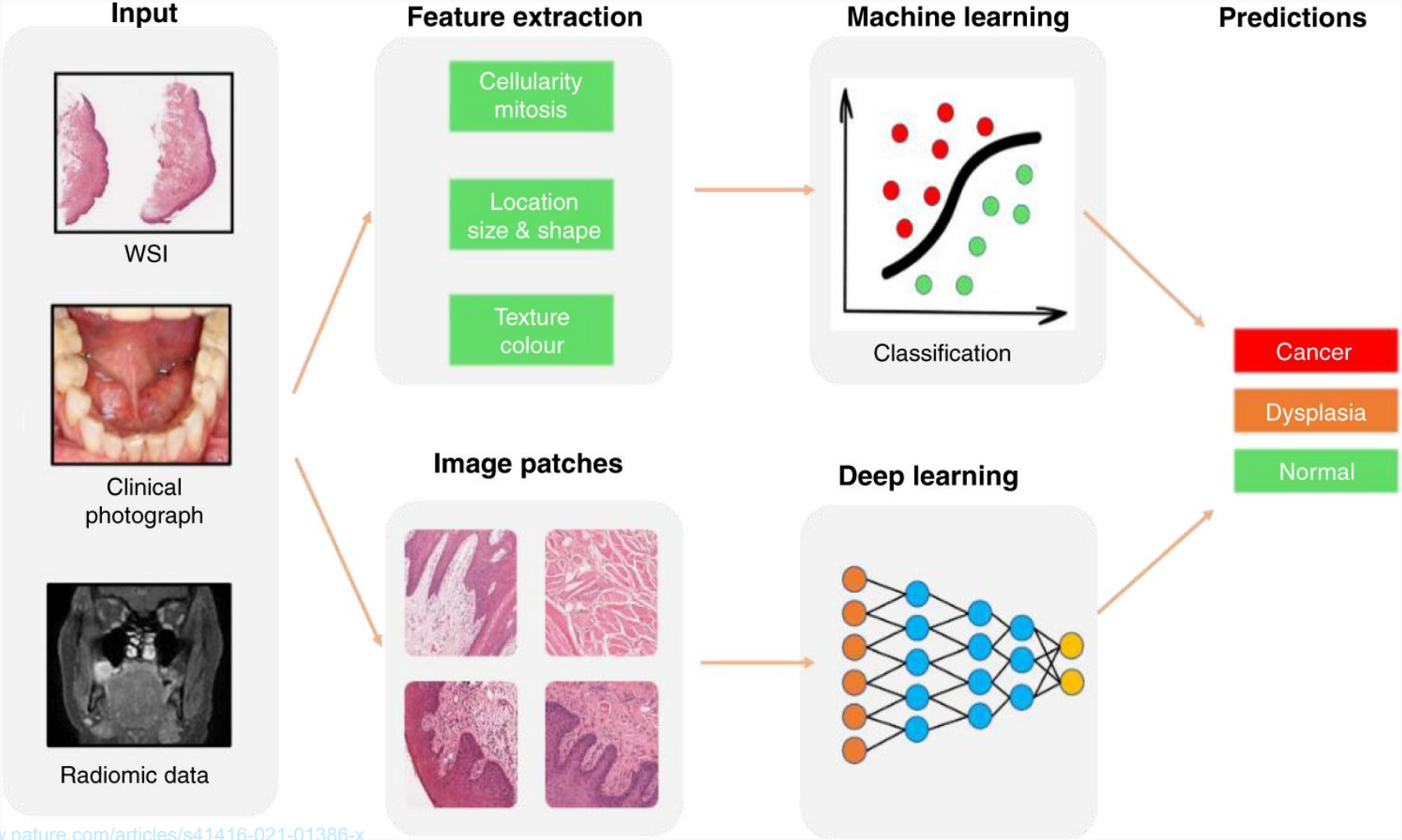
- **Features (X)**: The input variables (e.g., age, gender, test results).
- **Target (y)**: The label we are trying to predict (e.g., cancer diagnosis).

https://www.youtube.com/watch?v=ALQ_RNSRE40

https://www.youtube.com/watch?v=ALQ_RNSRE40

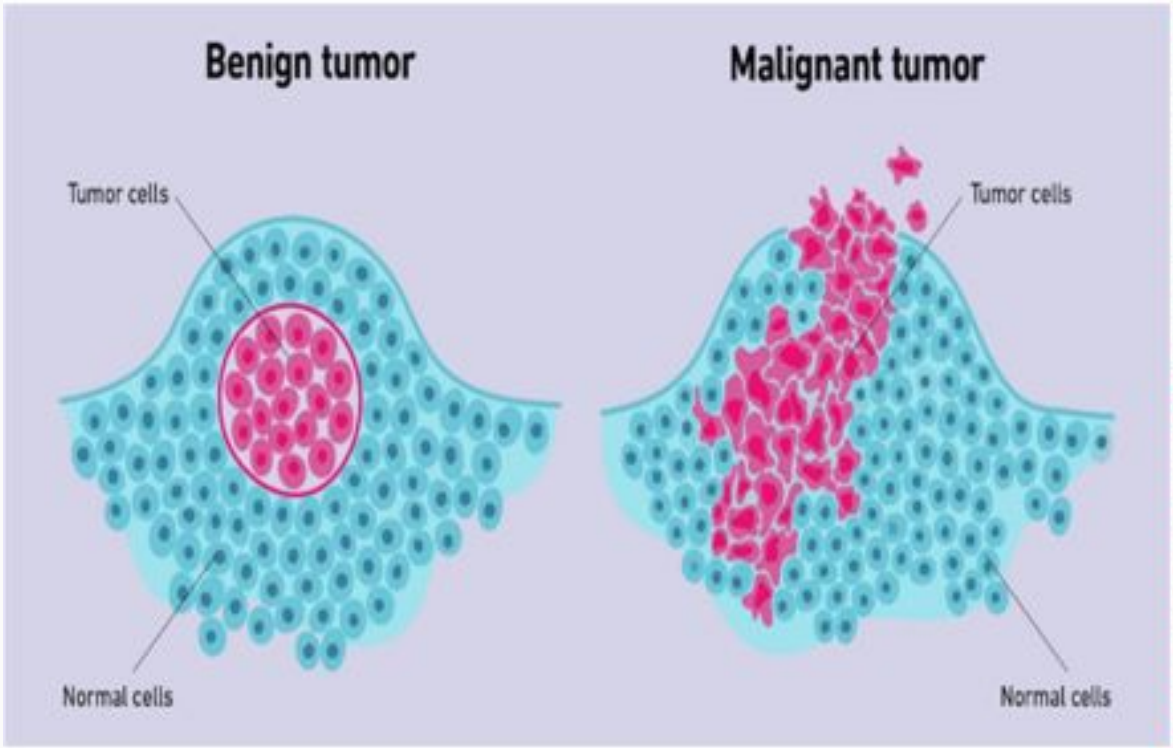
https://www.youtube.com/watch?v=ALQ_RNSRE40

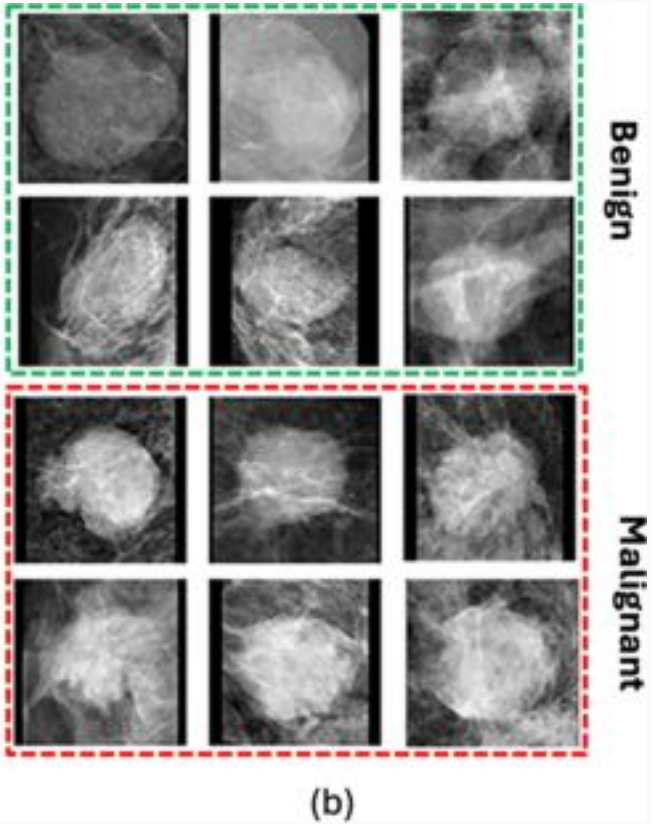




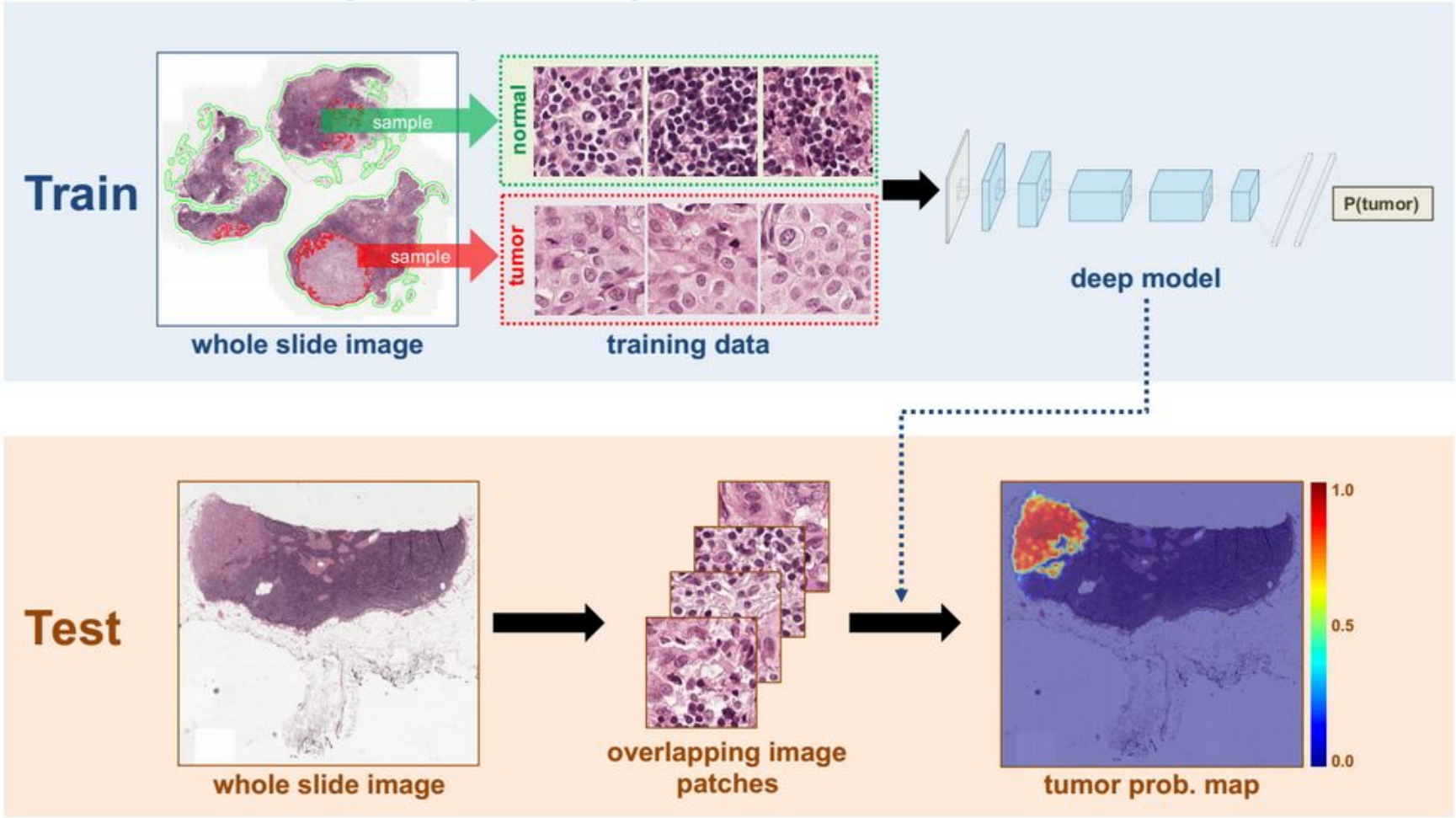
https://www.nature.com/articles/s41416-021-01386-x

https://www.nature.com/articles/s41598-024-57740-5





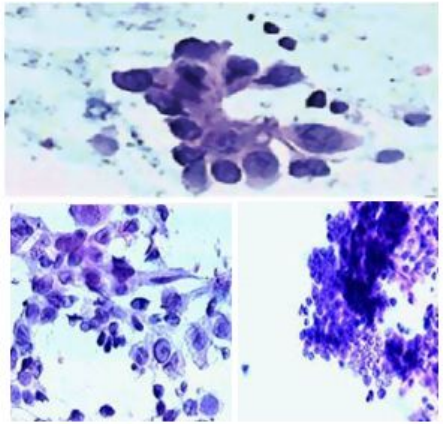
https://medium.com/data-science/understanding-cancer-using-machine-learning-84087258ee18



https://www.linkedin.com/pulse/how-i-used-machine-learning-classify-two-types-cancer-bhandari-gcjtc

Breast Cancer Detection using Machine Learning

Breast Tumor Images



Extracted Features

Radius
Texture
Area
Perimeter
Smoothness
Compactness
Concavity
Symmetry
Fractal Dimension

Data pre-processing

Machine Learning

Classification

Malignant

Benign

Overview of Scikit-learn for Supervised Learning

Why Scikit-learn?

- **Simple**: Scikit-learn provides an easy-to-use API for machine learning.
- **Wide Range of Algorithms**: Supports many algorithms for classification, regression, clustering, and more.
- **Efficient**: Offers optimized algorithms for quick model building.

Machine Learning Workflow in Scikit-learn

1. **Load and Prepare Data**
2. **Preprocess Data** (e.g., encoding, scaling, etc.)
3. **Split Data** into training and test sets
4. **Choose a Model** (e.g., Decision Tree, Logistic Regression)
5. **Train the Model** using the training set
6. **Evaluate the Model** using metrics (e.g., accuracy, precision, recall)
7. **Fine-tune and Optimize** the model (e.g., hyperparameter tuning)
8. **Deploy** for predictions on new data

Exploratory Data Analysis (EDA)

Before training a model, understanding the dataset is essential.
EDA helps uncover patterns, detect outliers, and understand the relationships between variables.

Basic Steps in EDA:

1. **Check Dataset Shape**: Understand the size and structure of your data.
2. **Summary Statistics**: Get a statistical summary for numerical features.
3. **Count Categories**: Understand the distribution of categorical variables.
4. **Visualize**: Use plots to identify patterns in the data.

Code Example:

python Copy Edit

```python
# Display basic information about the dataset
print(df.info())
print(df.describe())
```

Data Preprocessing: Handling Categorical Variables

Why is Preprocessing Important?

- **Machine Learning models require numerical data**.
- We need to convert categorical variables into numerical formats (e.g., Male/Female → 0/1).

One-hot Encoding:

- **Definition**: Convert each category in a feature into a separate binary column.
- **Example**: "Gender" (Male, Female) → Create columns for "Male" and "Female".

Code Example:

python Copy Edit

```python
# One-hot encode categorical variables
df = pd.get_dummies(df, columns=['Gender', 'Symptom'], dummy_na=True)
```

Splitting Data into Training and Testing Sets

To evaluate the model's performance, we split the data into two sets:

- **Training Set**: Used to train the model.
- **Testing Set**: Used to evaluate how well the model generalizes to unseen data.

Why Split the Data?

- The model is trained on the training data, and its accuracy is tested on the test data to simulate real-world scenarios.

Code Example:

python

Copy Edit

```python
from sklearn.model_selection import train_test_split

X = df.drop(columns=['Diagnosis'])  # Features
y = df['Diagnosis']  # Target label

# Split data with a random seed for reproducibility
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Choosing and Training Supervised Models

Different Supervised Learning Models:

1. **Decision Trees**: A flowchart-like model used for classification and regression.
2. **Random Forests**: An ensemble of decision trees to improve accuracy.
3. **Logistic Regression**: A statistical model used for binary classification.
4. **Neural Networks**: A complex model inspired by the human brain, used for more intricate data patterns.

Training the Models:

- Fit each model using the training data.
- Evaluate performance on the test set.

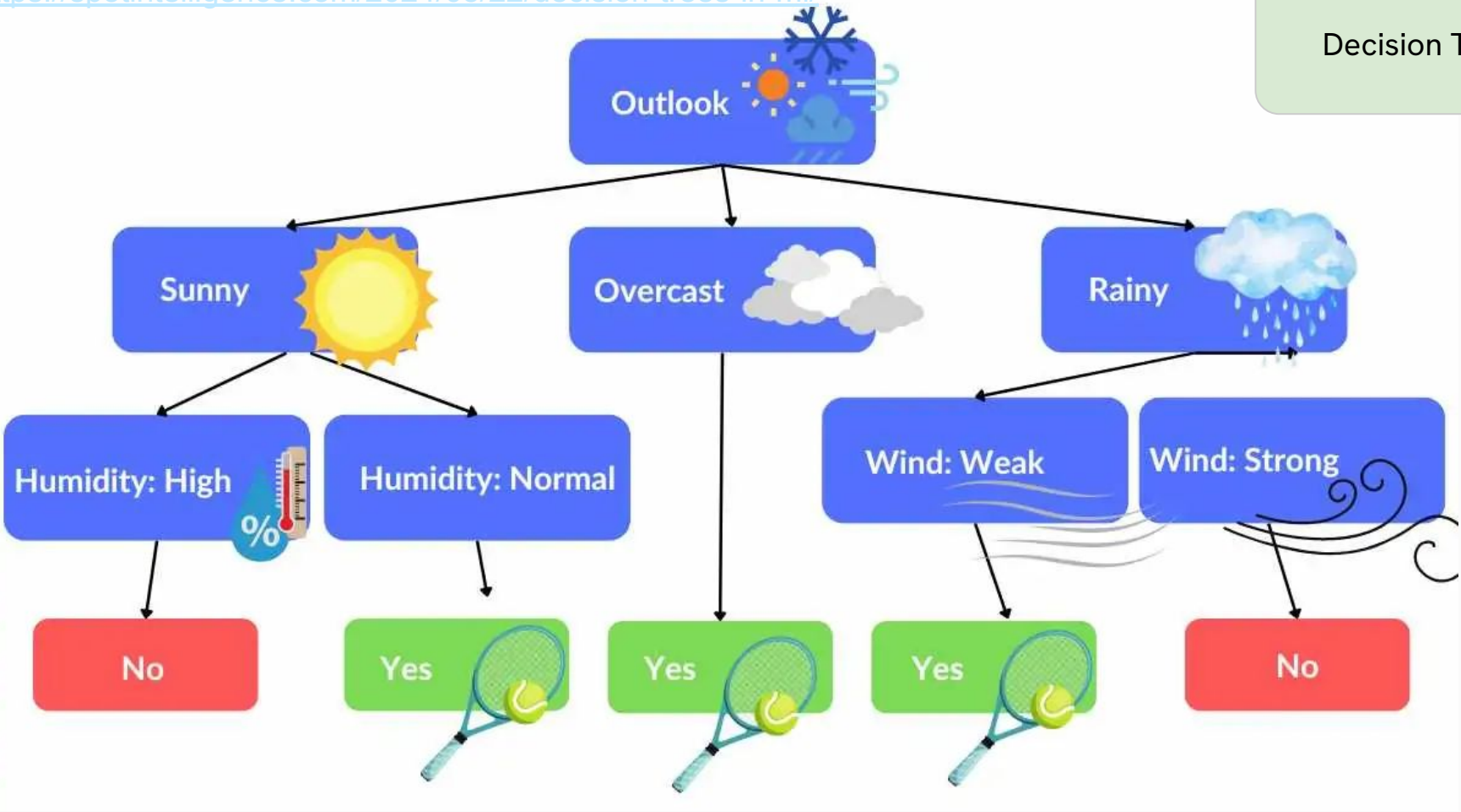
Code Example:

python Copy Edit

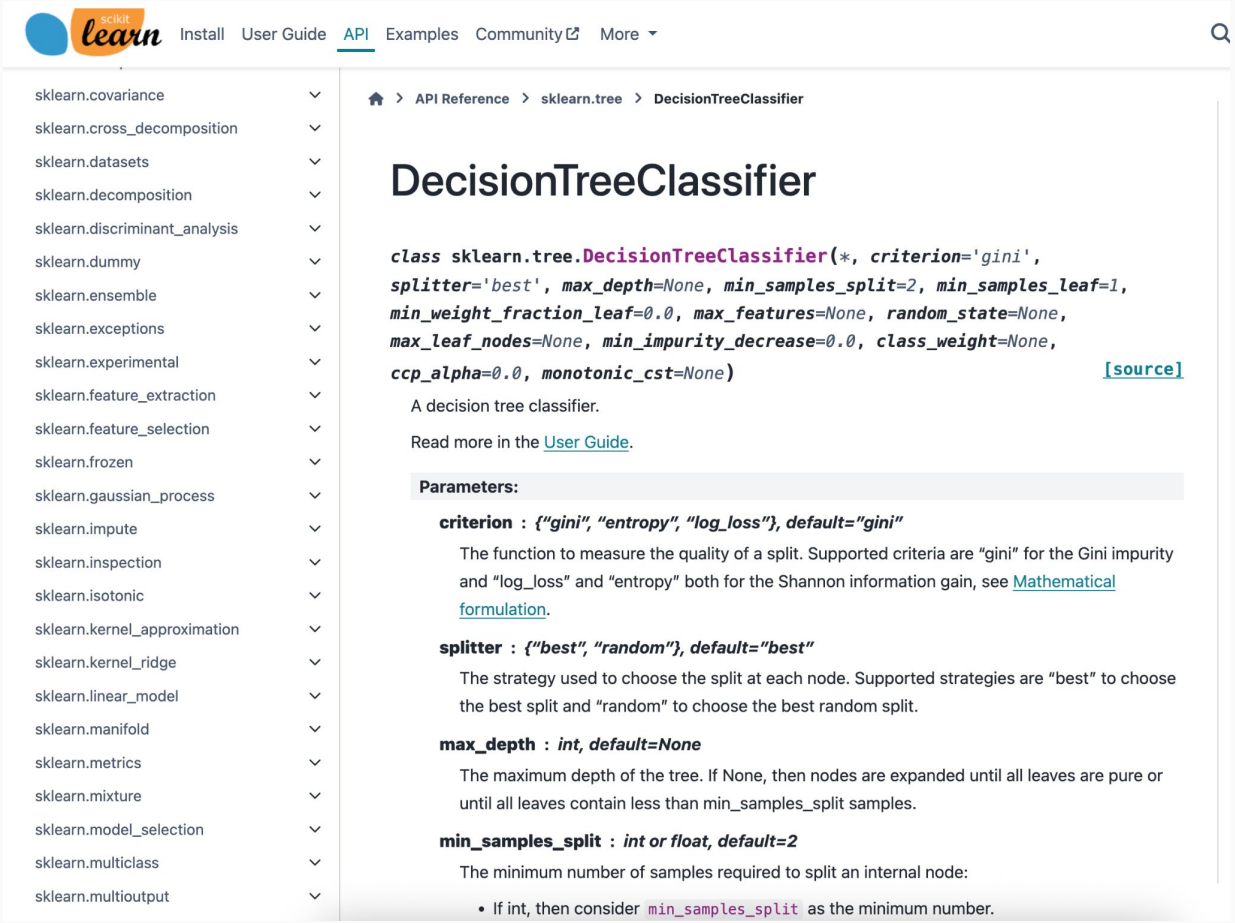
```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier

models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42),
    "Neural Network": MLPClassifier(hidden_layer_sizes=(32, 16), max_iter=1000, random_st
}

# Train and evaluate models
results = []
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    results.append([name, acc])
```

https://spotintelligence.com/2024/05/22/decision-trees-in-ml/

Decision Tree

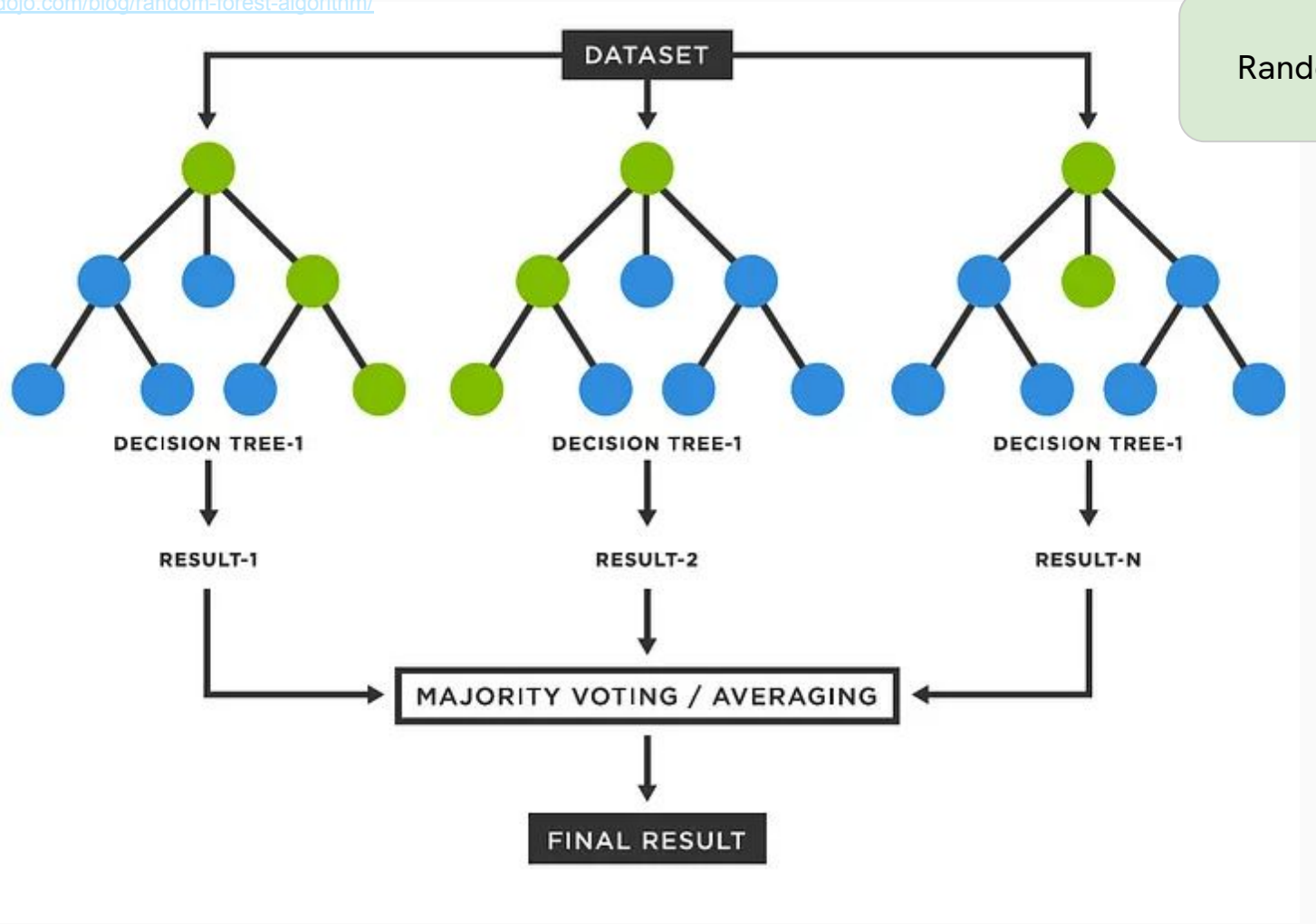


https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

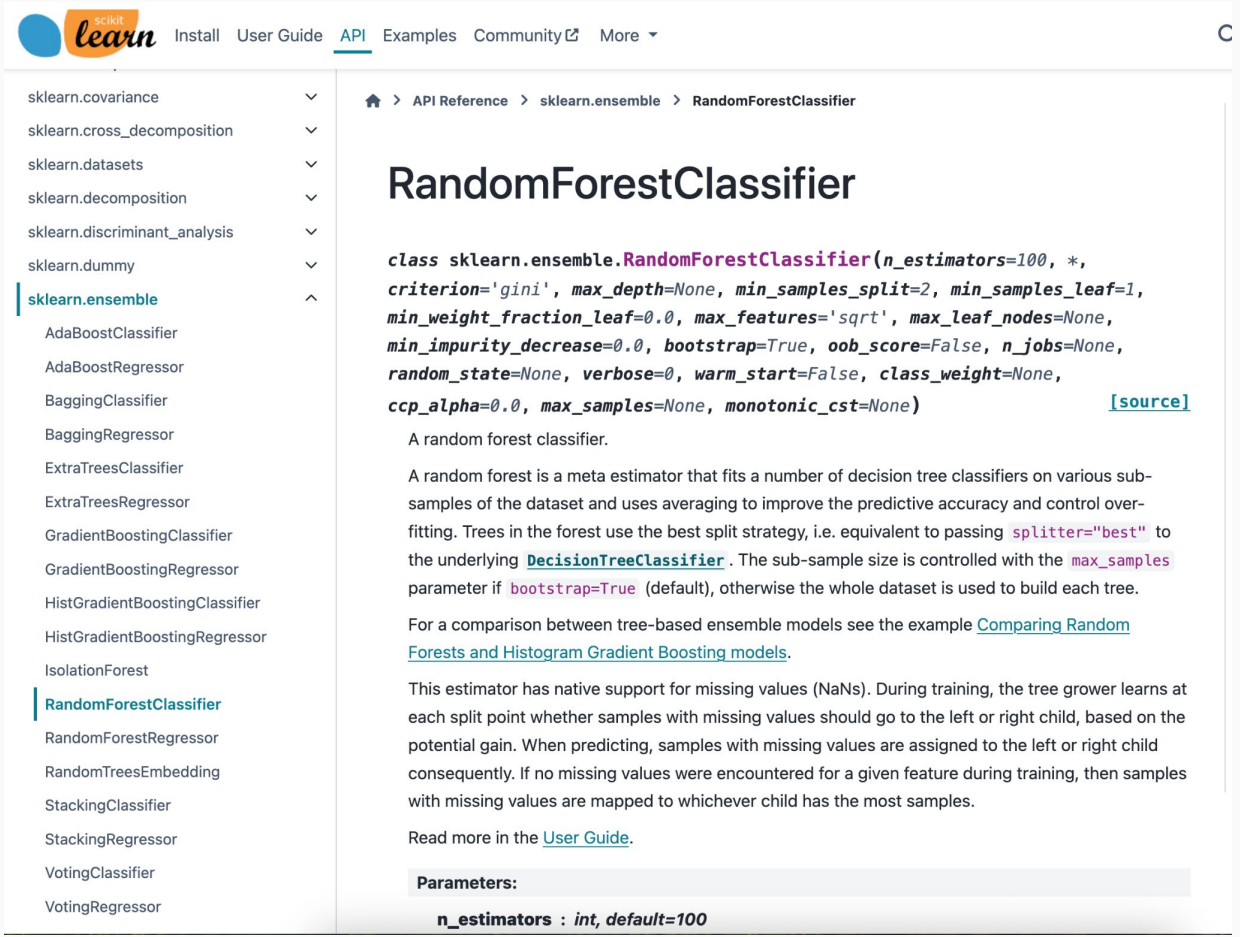


https://datasciencedojo.com/blog/random-forest-algorithm/

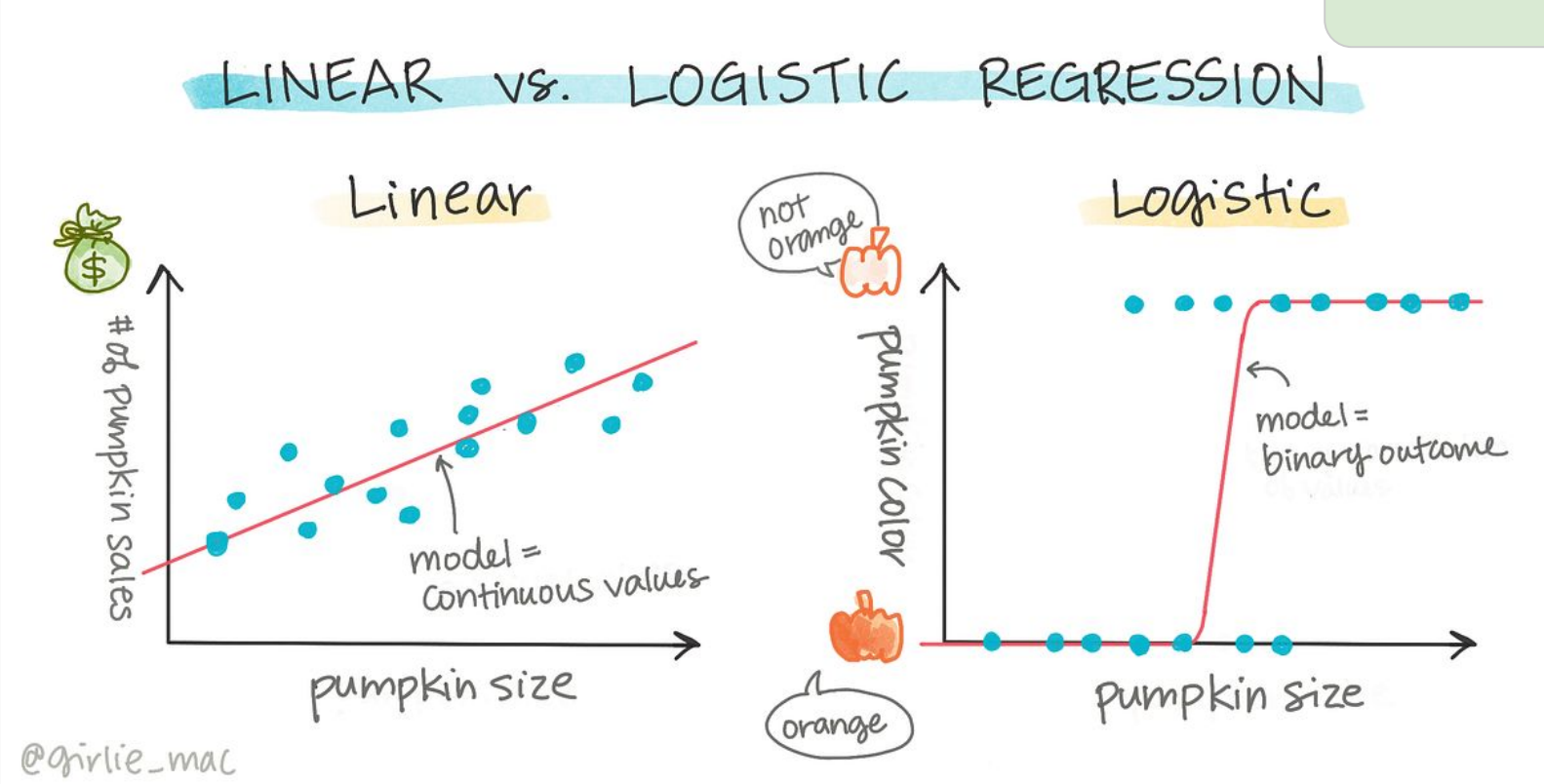
Random Forest



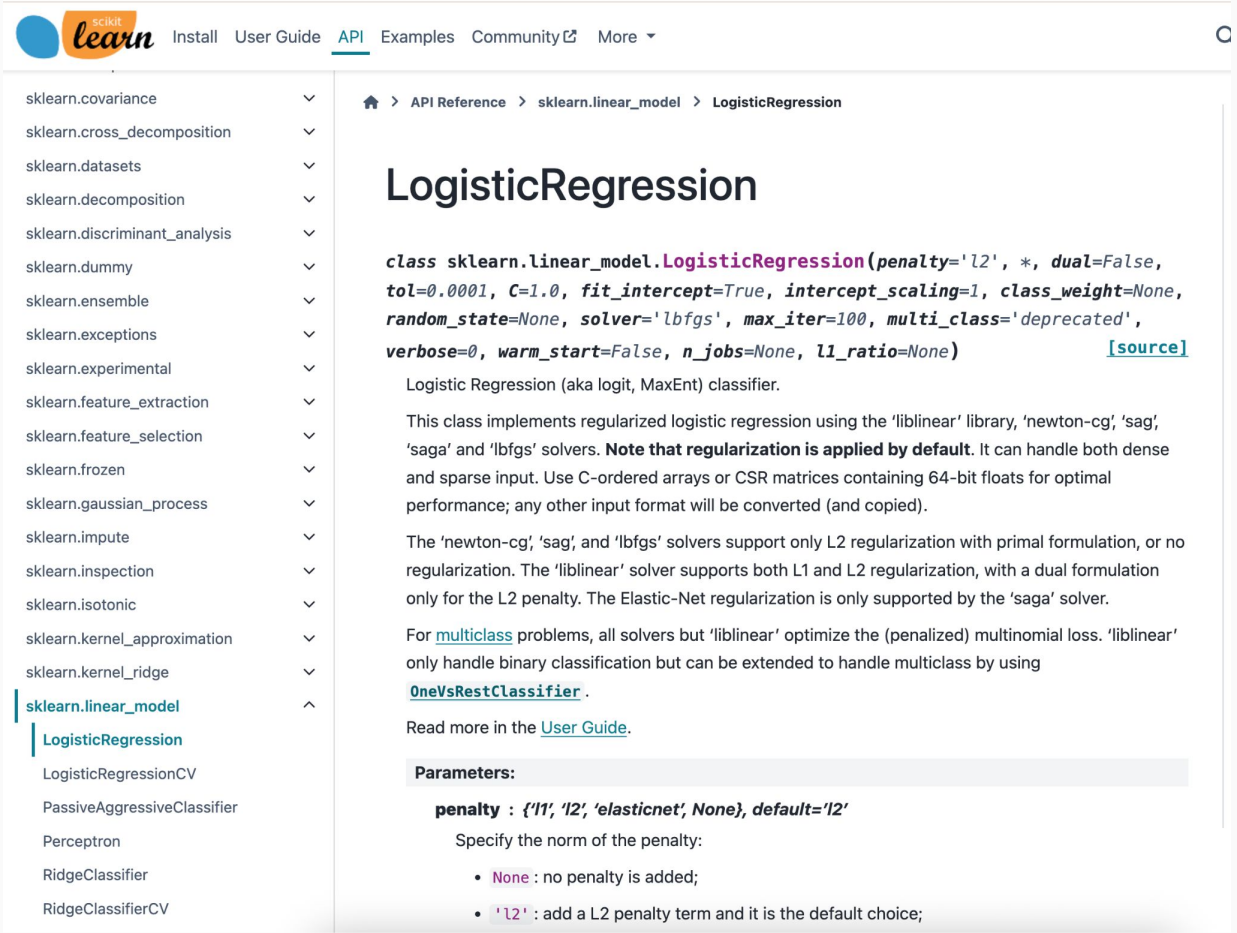
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html



https://blog.gopenai.com/linear-and-logistic-regression-same-regression-but-different-purpose-f6ff5f93b7ef?gi=820735240867

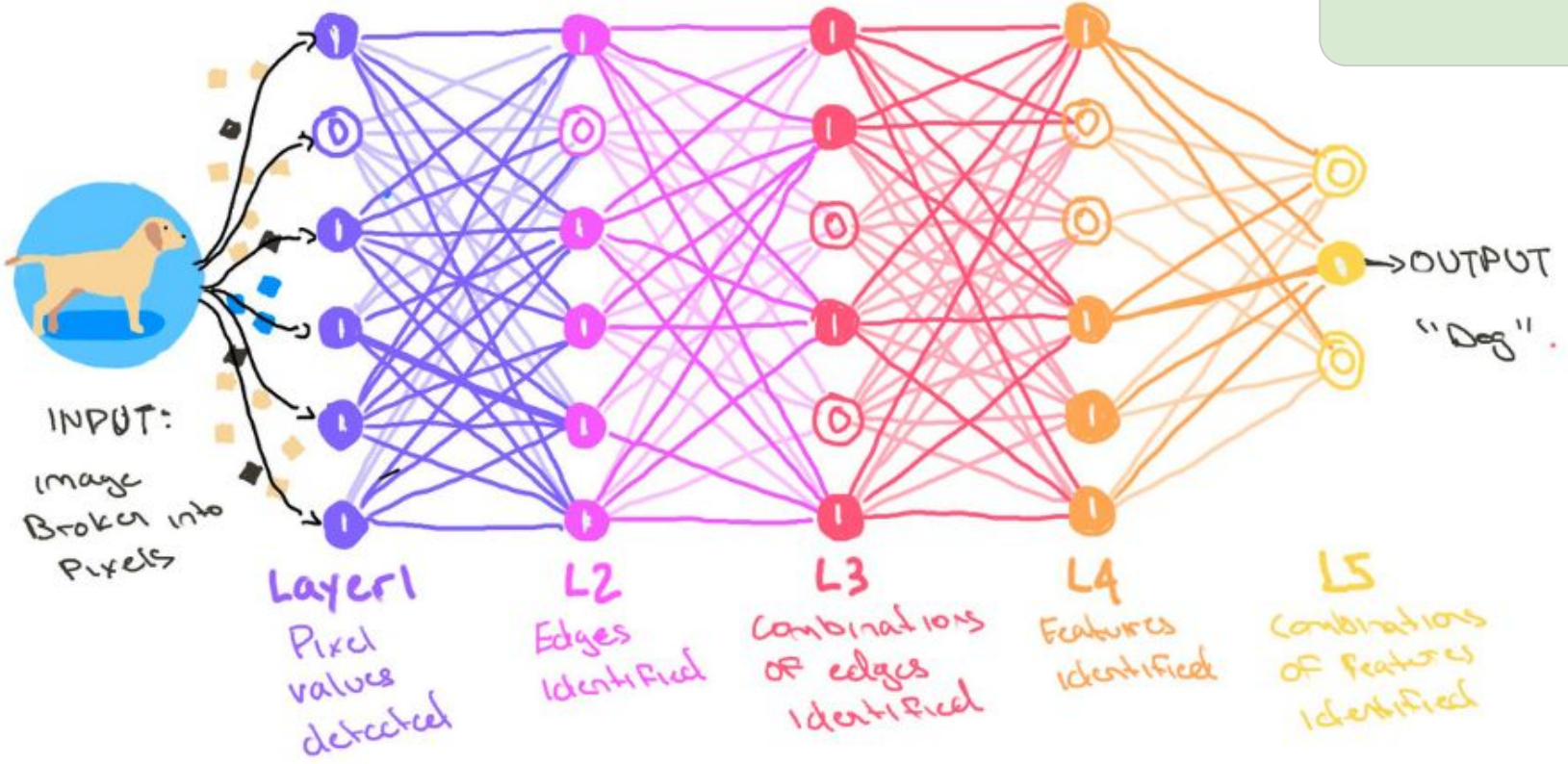


https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

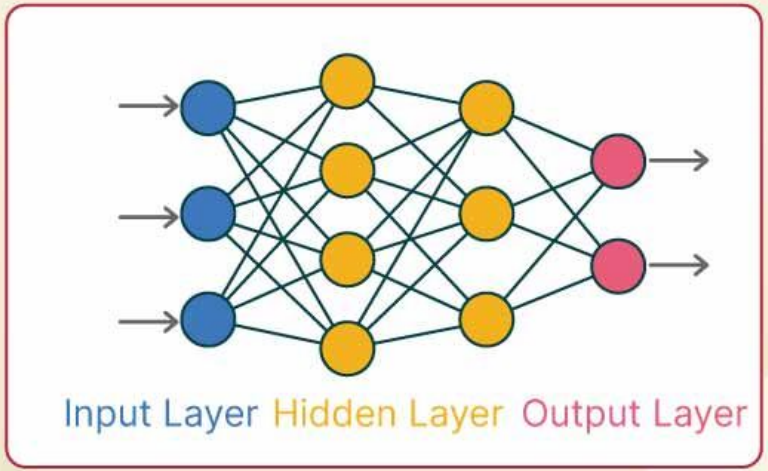


https://www.notablecap.com/blog/the-anatomy-of-a-neural-network

Neural Network



Multilayer Perceptron (MLP) Neural Networks



https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

scikit learn Install User Guide API Examples Community More

- sklearn.covariance
- sklearn.cross_decomposition
- sklearn.datasets
- sklearn.decomposition
- sklearn.discriminant_analysis
- sklearn.dummy
- sklearn.metrics
- sklearn.mixture
- sklearn.model_selection
- sklearn.multiclass
- sklearn.multioutput

API Reference > sklearn.neural_network > MLPClassifier

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]

Multi-layer Perceptron classifier.

This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

Added in version 0.18.

Parameters:

hidden_layer_sizes : *array-like of shape(n_layers - 2,), default=(100,)*

The ith element represents the number of neurons in the ith hidden layer.

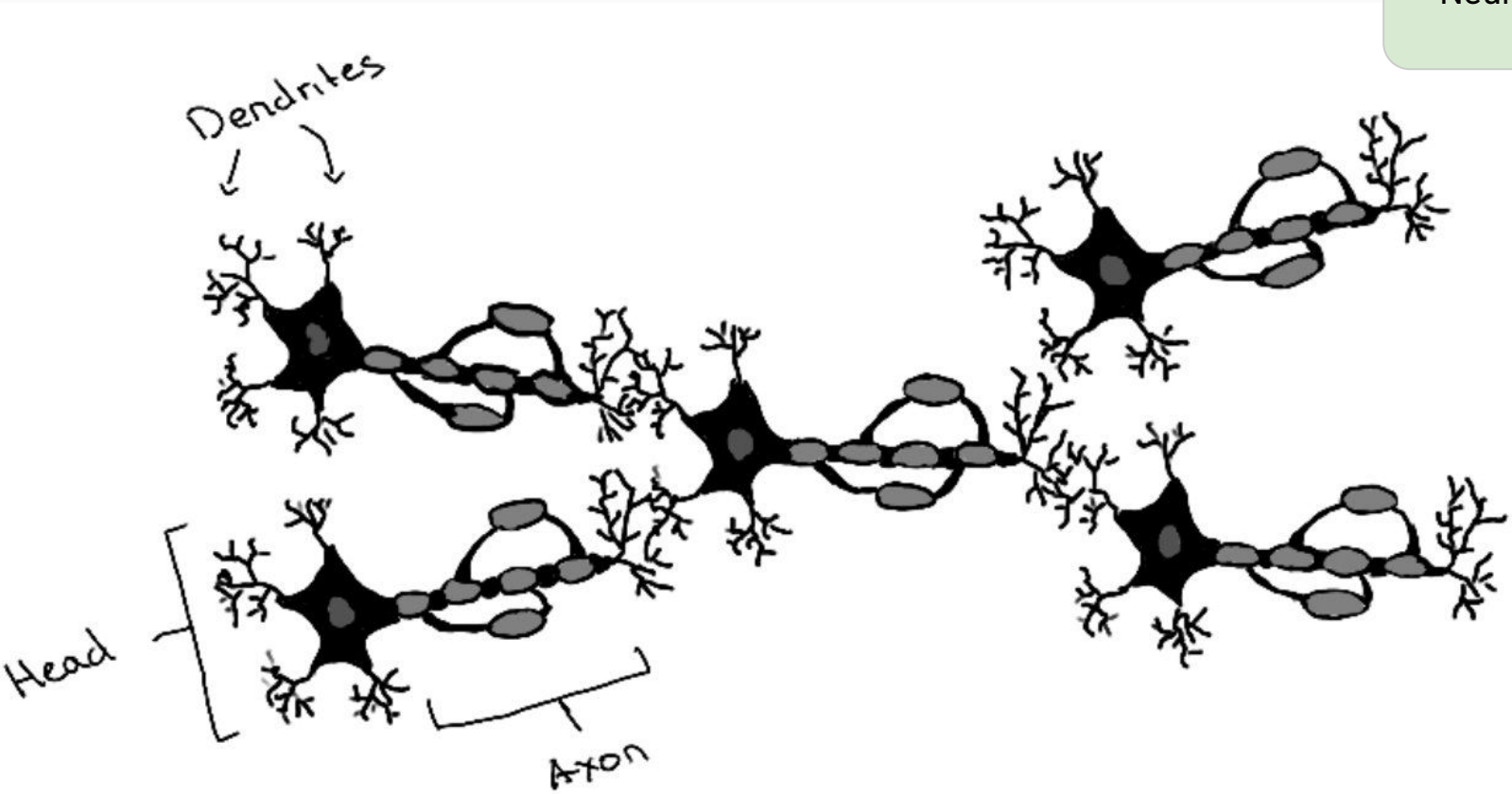
activation : *{'identity', 'logistic', 'tanh', 'relu'}, default='relu'*

Activation function for the hidden layer.

- 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
- 'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
- 'tanh', the hyperbolic tan function, returns f(x) = tanh(x).
- 'relu', the rectified linear unit function, returns f(x) = max(0, x)

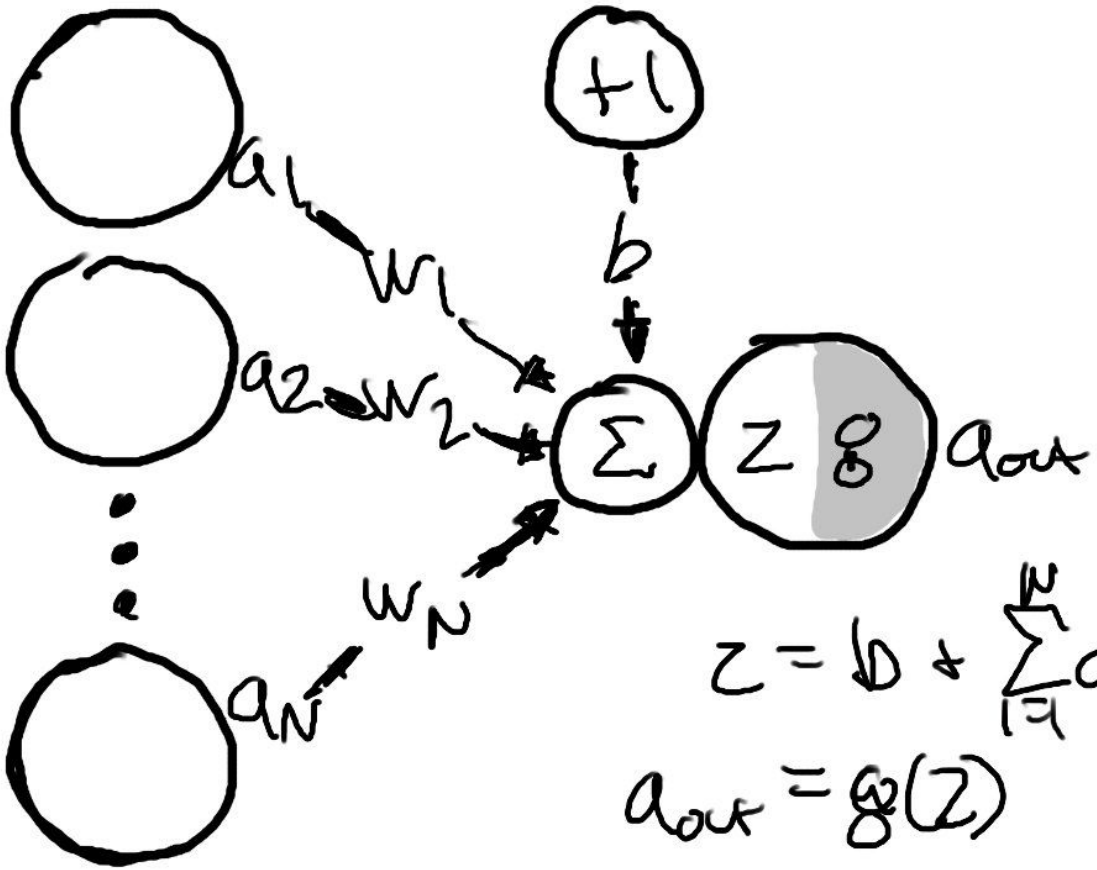
https://www.notablecap.com/blog/the-anatomy-of-a-neural-network

Neural Network



https://www.notablecap.com/blog/the-anatomy-of-a-neural-network

Neural Network



Model Evaluation: Accuracy, Precision, Recall, F1-Score

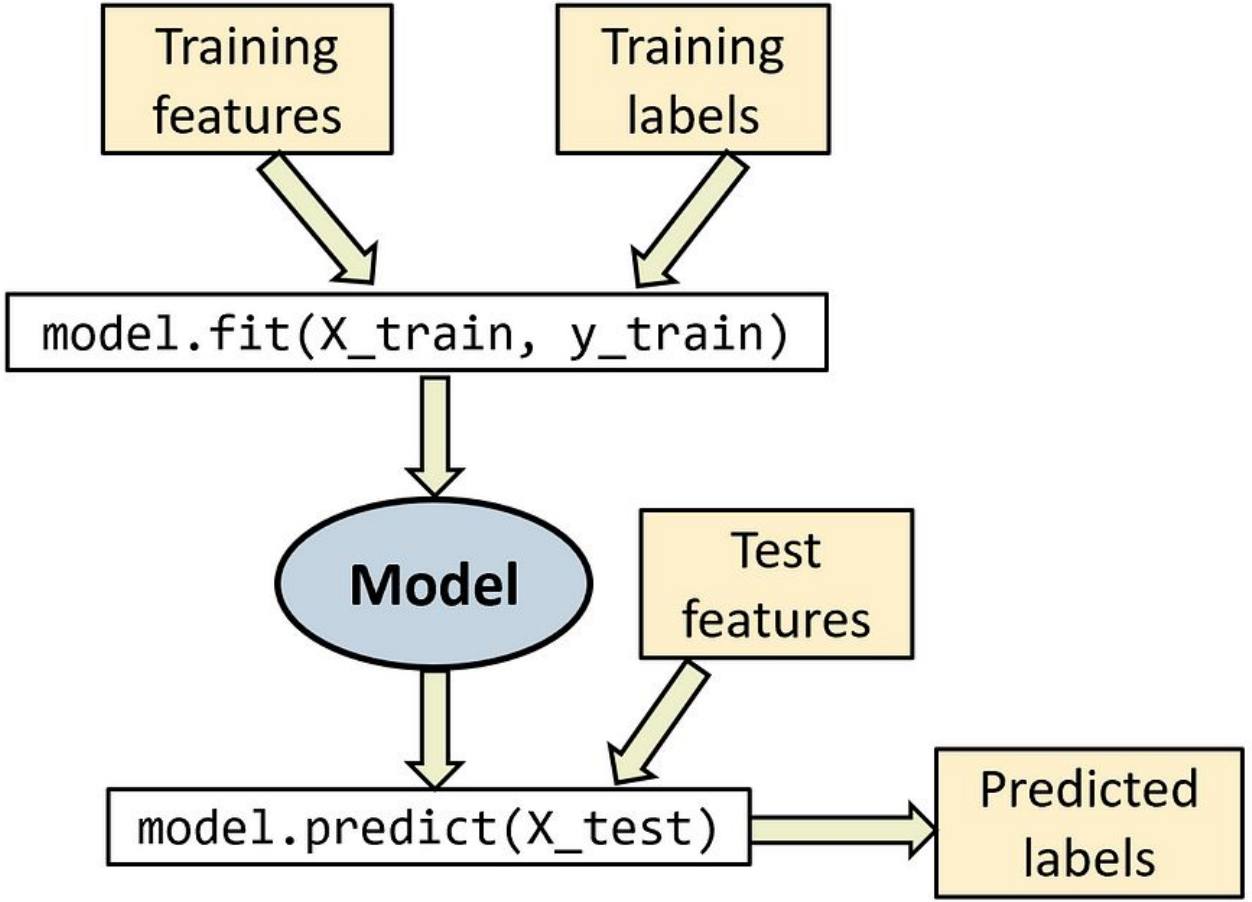
Model Evaluation Metrics:

1. **Accuracy**: Percentage of correctly predicted labels.
2. **Precision**: How many of the predicted positives were actually positive?
3. **Recall**: How many actual positives were predicted correctly?
4. **F1-Score**: A balance between precision and recall (important for imbalanced datasets).

Why These Metrics Matter?

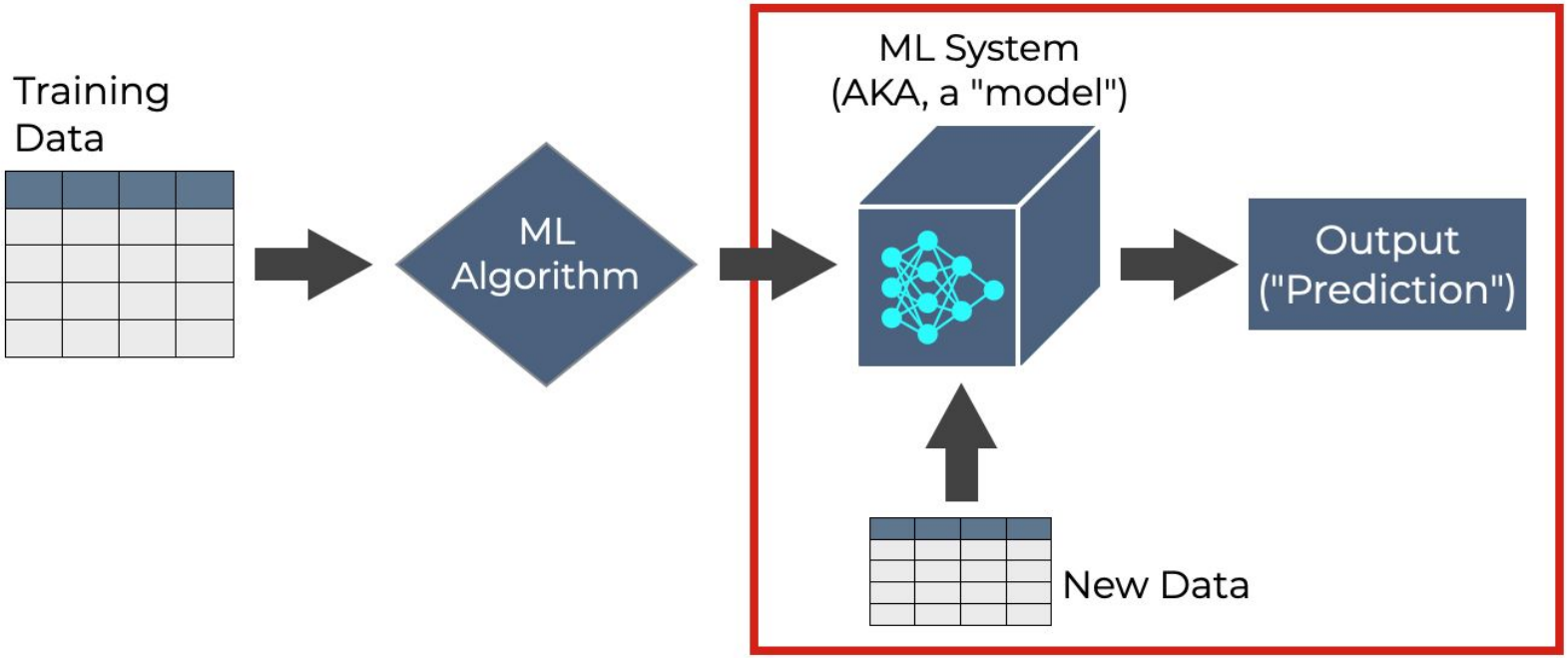
- **Accuracy** is not always reliable (e.g., in imbalanced datasets).
- **Precision** and **Recall** are useful when we care about the consequences of false positives or false negatives (e.g., diagnosing cancer).

https://levelup.gitconnected.com/introduction-to-scikit-learn-1792dc6937e0?gi=bae6004db2d1



https://r-craft.org/how-to-use-the-sklearn-predict-method/

WITH SOME MODELS, WE CAN MAKE A "PREDICTION" ON THE BASIS OF NEW INPUT VALUES



Code Example:

python Copy Edit

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    pr = precision_score(y_test, y_pred)
    rc = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    print(f"{name} - Accuracy: {acc}, Precision: {pr}, Recall: {rc}, F1 Score: {f1}")
```

Visualizing Model Performance: Confusion Matrix

A **Confusion Matrix** is a summary of how well the model is performing. It shows the count of:

- **True Positives (TP)**: Correctly predicted positive cases.
- **False Positives (FP)**: Incorrectly predicted positive cases.
- **True Negatives (TN)**: Correctly predicted negative cases.
- **False Negatives (FN)**: Incorrectly predicted negative cases.

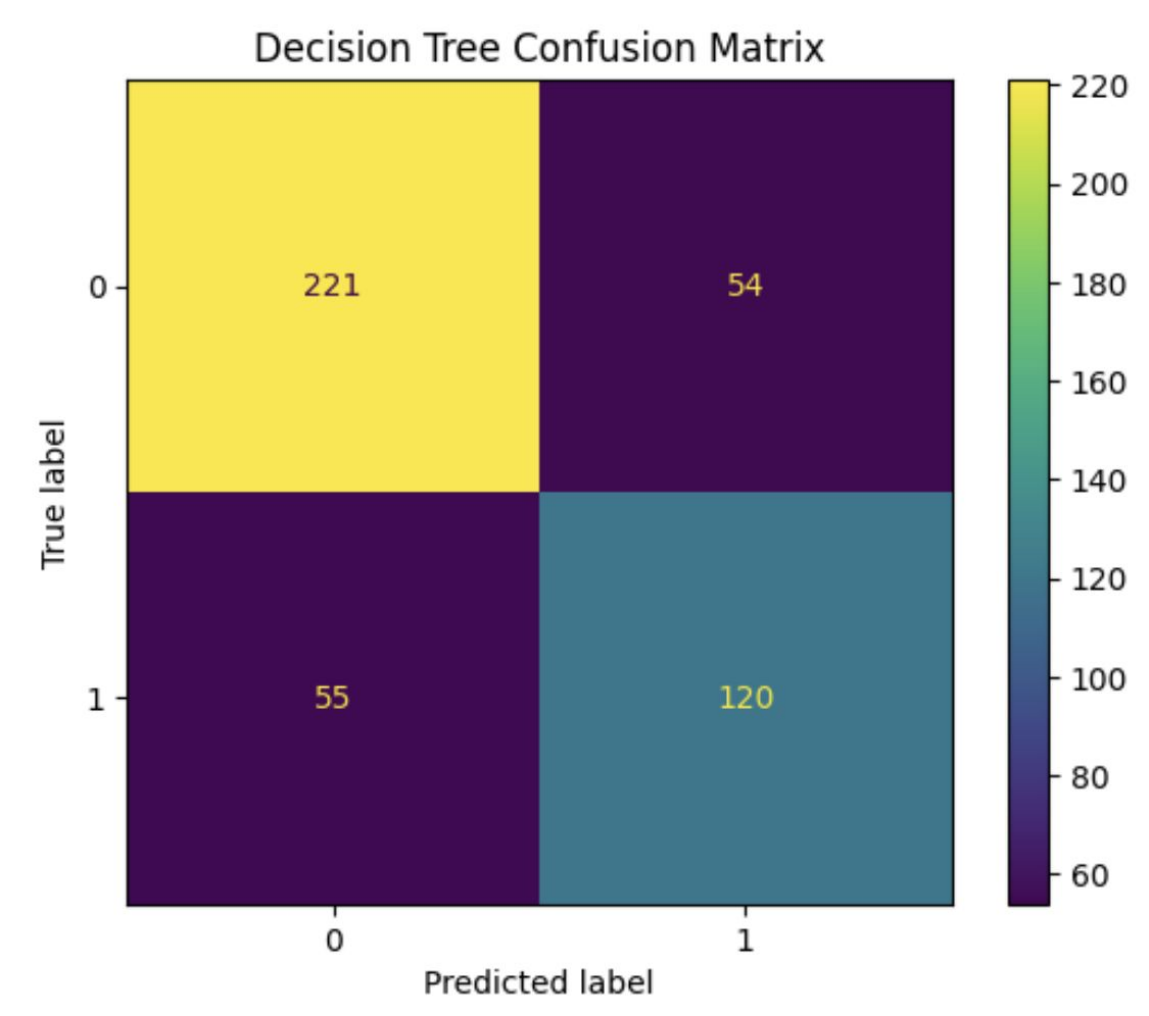
Visualizing the confusion matrix allows us to intuitively understand model performance.

Code Example:

python

Copy Edit

```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

for name, model in models.items():
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)

    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot()
    plt.title(f"{name} Confusion Matrix")
    plt.show()
```

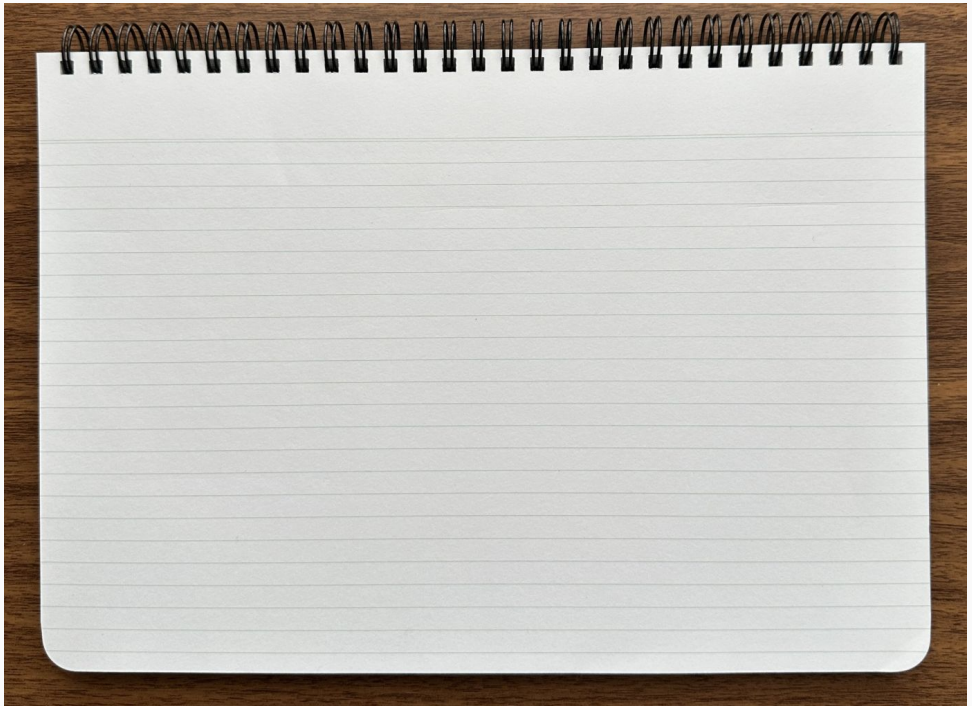

Decision Tree Confusion Matrix
221
54
55
120
0
1
True label
0
1
Predicted label
220
200
180
160
140
120
100
80
60

ACTUAL VALUES	POSITIVE	NEGATIVE
POSITIVE	TP	FN
NEGATIVE	FP	TN

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

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

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



Comparing Model Performance: Bar Plot

Once we evaluate the models using various metrics, we can **compare their performance** visually.

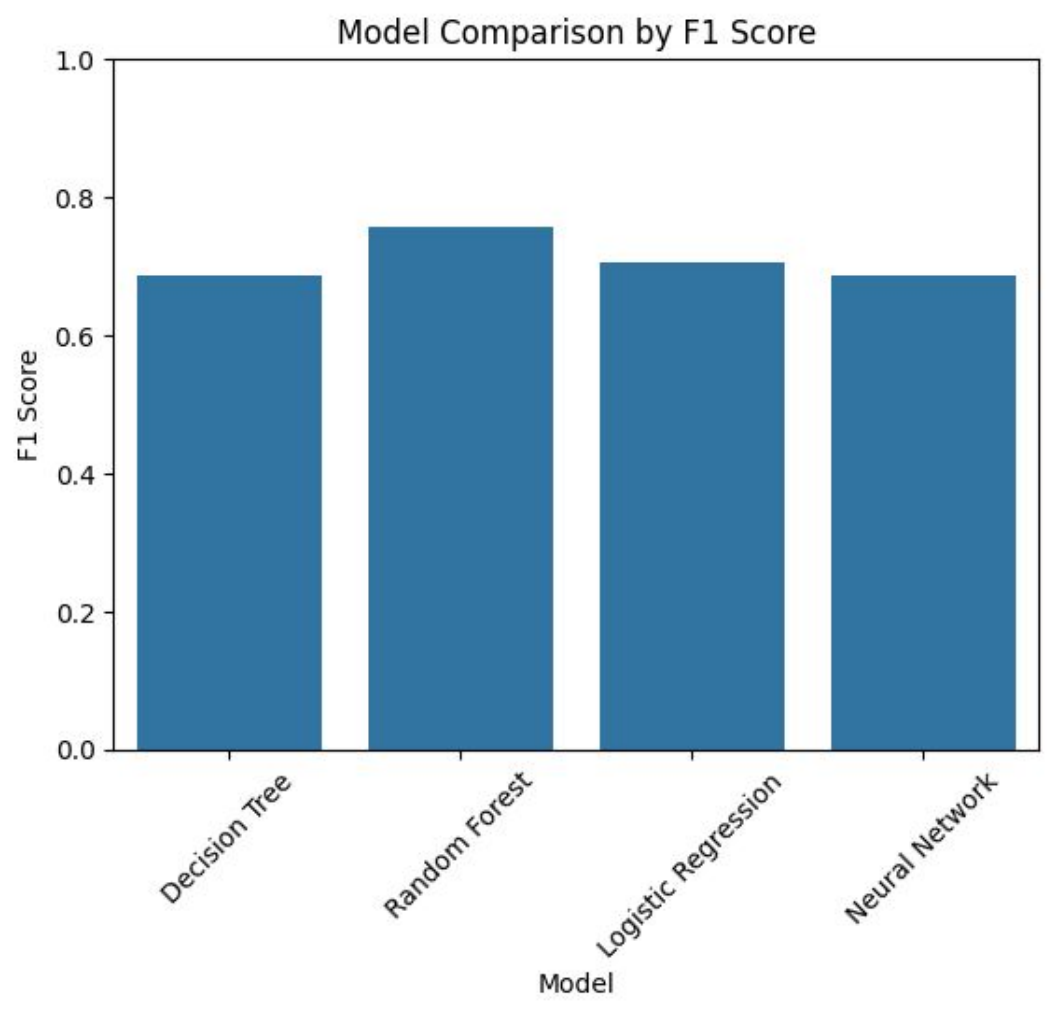
Using F1-Score for Comparison:

- **F1-Score** provides a balanced measure for imbalanced datasets.

Code Example:

python Copy Edit

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Create a DataFrame of results
results_df = pd.DataFrame(results, columns=["Model", "F1 Score"])

# Visualize the comparison
sns.barplot(data=results_df, x='Model', y='F1 Score')
plt.title("Model Comparison by F1 Score")
plt.ylim(0, 1)
plt.xticks(rotation=45)
plt.show()
```


Model Comparison by F1 Score
1.0
0.8
0.6
0.4
0.2
0.0
F1 Score
Decision Tree
Random Forest
Logistic Regression
Neural Network
Model

ROC Curve and AUC (Area Under the Curve)

The **ROC curve** helps visualize the trade-off between the True Positive Rate (Recall) and the False Positive Rate. The **AUC** score measures the overall ability of the model to distinguish between classes.

AUC Interpretation:

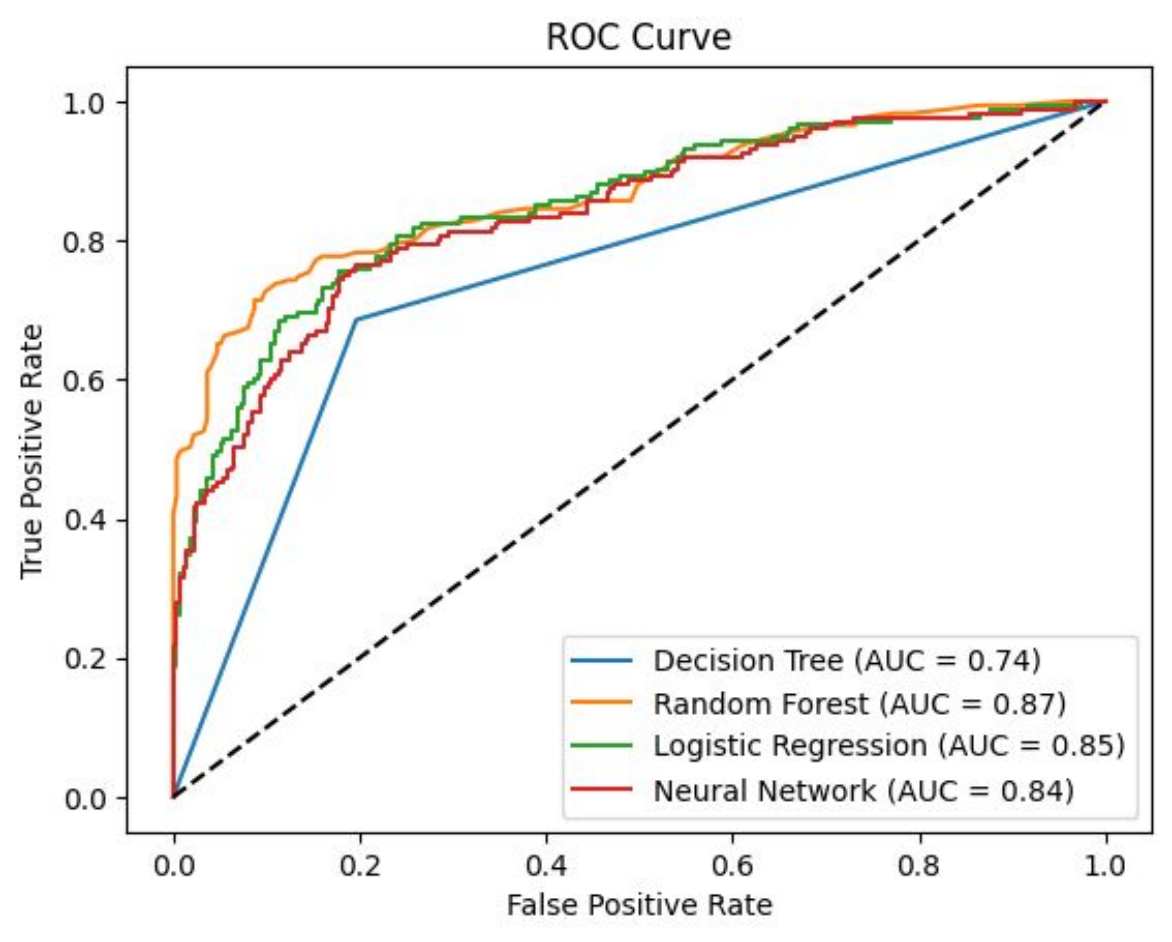
- **0.5**: No discrimination (random guessing).
- **1**: Perfect discrimination (ideal model).

Code Example:

python

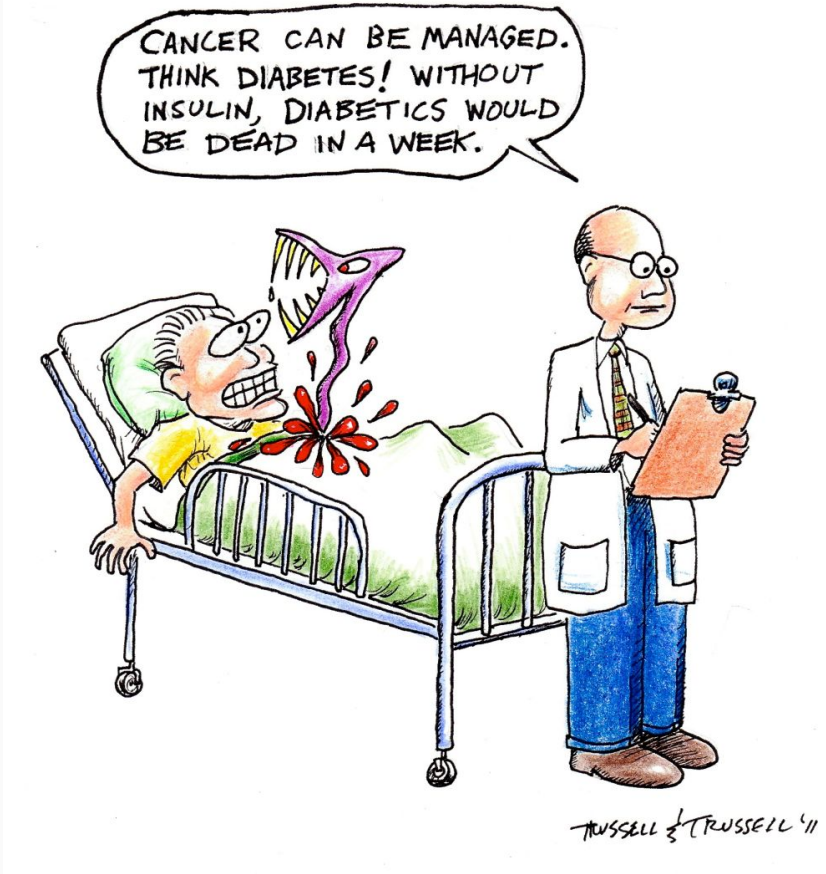
Copy Edit

```python
from sklearn.metrics import roc_curve, roc_auc_score

for name, model in models.items():
    y_prob = model.predict_proba(X_test)[:, 1]  # For probabilistic models
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    auc = roc_auc_score(y_test, y_prob)
    plt.plot(fpr, tpr, label=f"{name} (AUC = {auc:.2f})")

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Decision Tree (AUC = 0.74)
Random Forest (AUC = 0.87)
Logistic Regression (AUC = 0.85)
Neural Network (AUC = 0.84)

Conclusion: Model Evaluation and Next Steps

- **Evaluate multiple models** to choose the best one for your task.
- **Consider hyperparameter tuning** (e.g., using GridSearchCV) to optimize model performance.
- **Continue learning** about more advanced techniques (e.g., ensemble methods, deep learning).



https://donnatrussell.wordpress.com/cancer-cartoons/

Q&A Session:

- Any questions regarding the concepts covered today?
- Do you feel comfortable working with the models and evaluation metrics?