SC310005 Artificial Intelligence

Lecture 6: Supervised Learning (Part II)

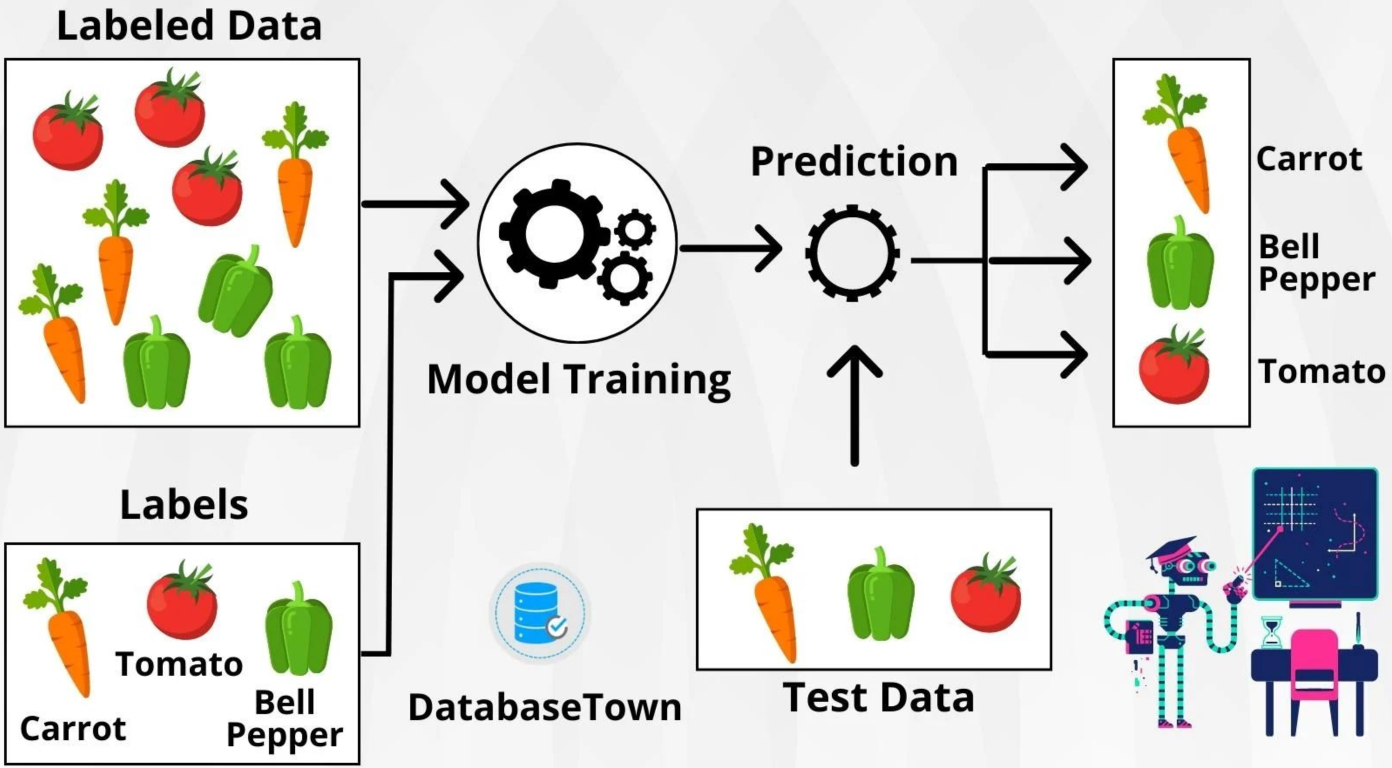
teerapong.pa@chula.ac.th

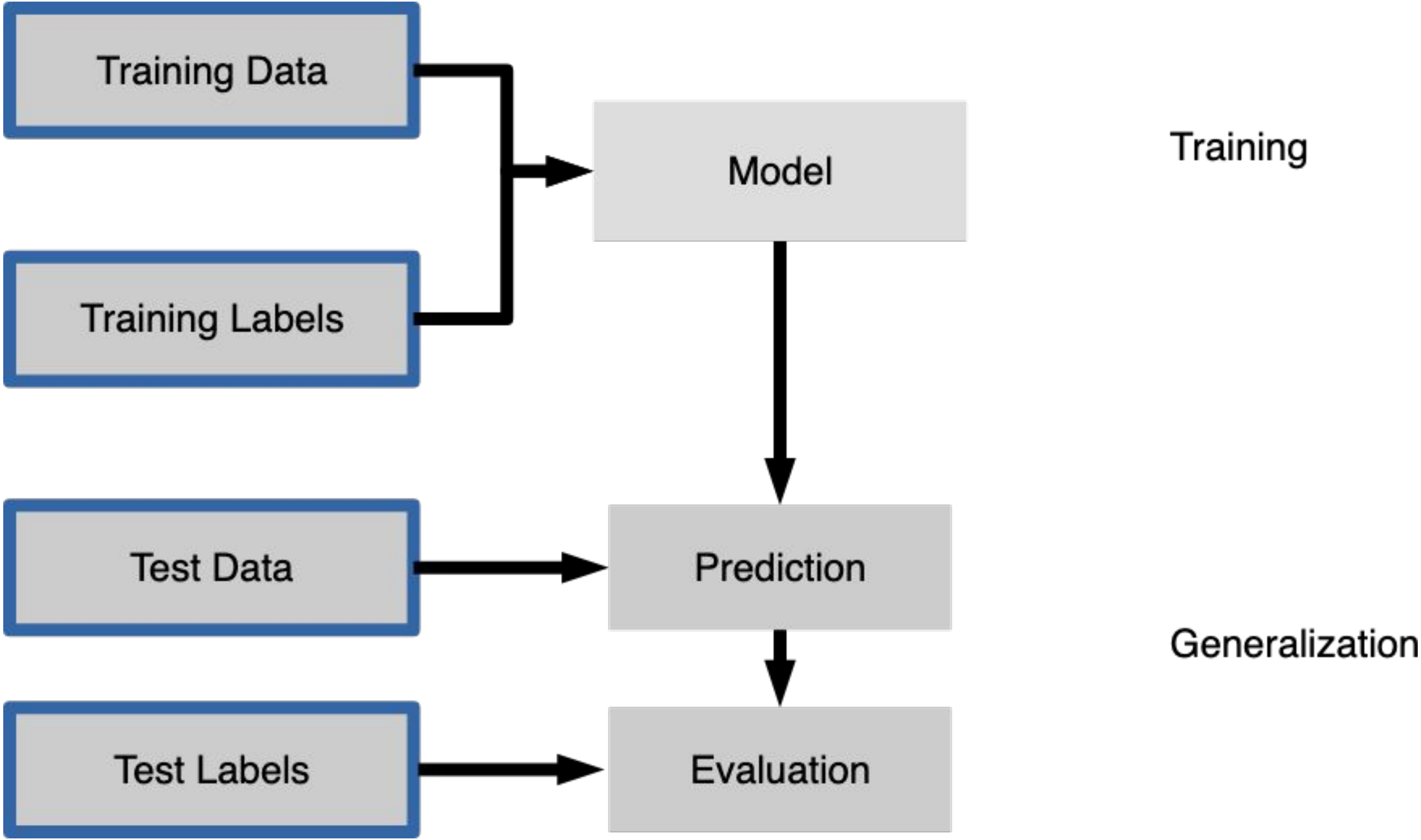
Reference

- https://anasbrital98.github.io/blog/2021/Random-Forest/
- https://medium.com/@favourphilic/decision-tree-5c1c7b6db59
- https://mljar.com/machine-learning/extra-trees-vs-random-forest/
- https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
- https://gaussian37.github.io/ml-concept-RandomForest/

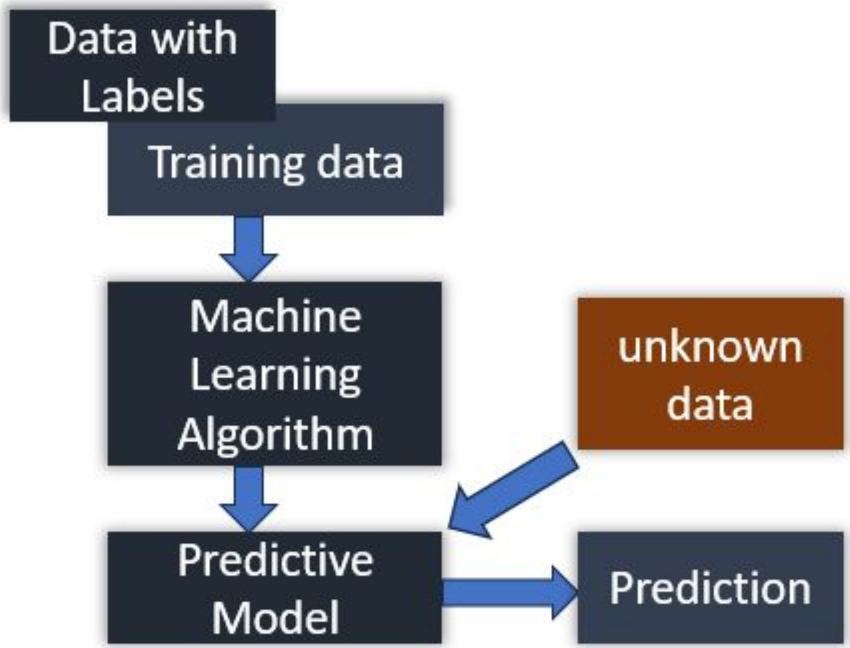
SUPERVISED LEARNING

Supervised machine learning is a branch of artificial intelligence that focuses on training models to make predictions or decisions based on labeled training data.

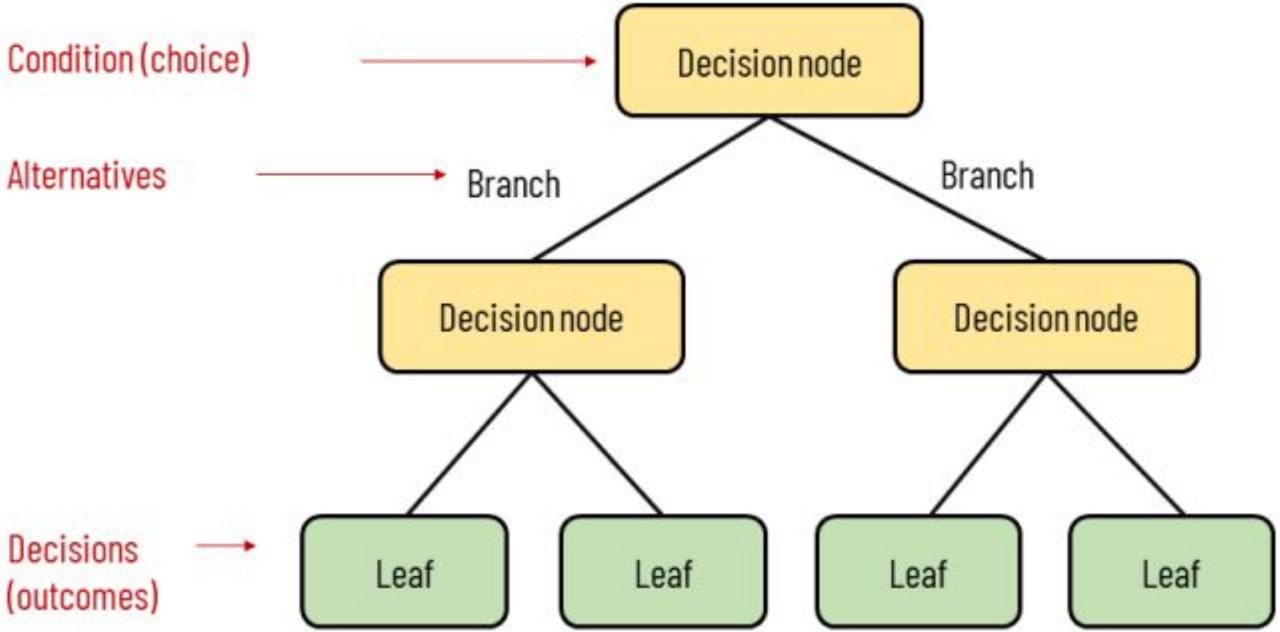



Training Data
Training Labels
Model
Training
Test Data
Prediction
Test Labels
Evaluation
Generalization

Supervised Learning



Elements of a decision tree



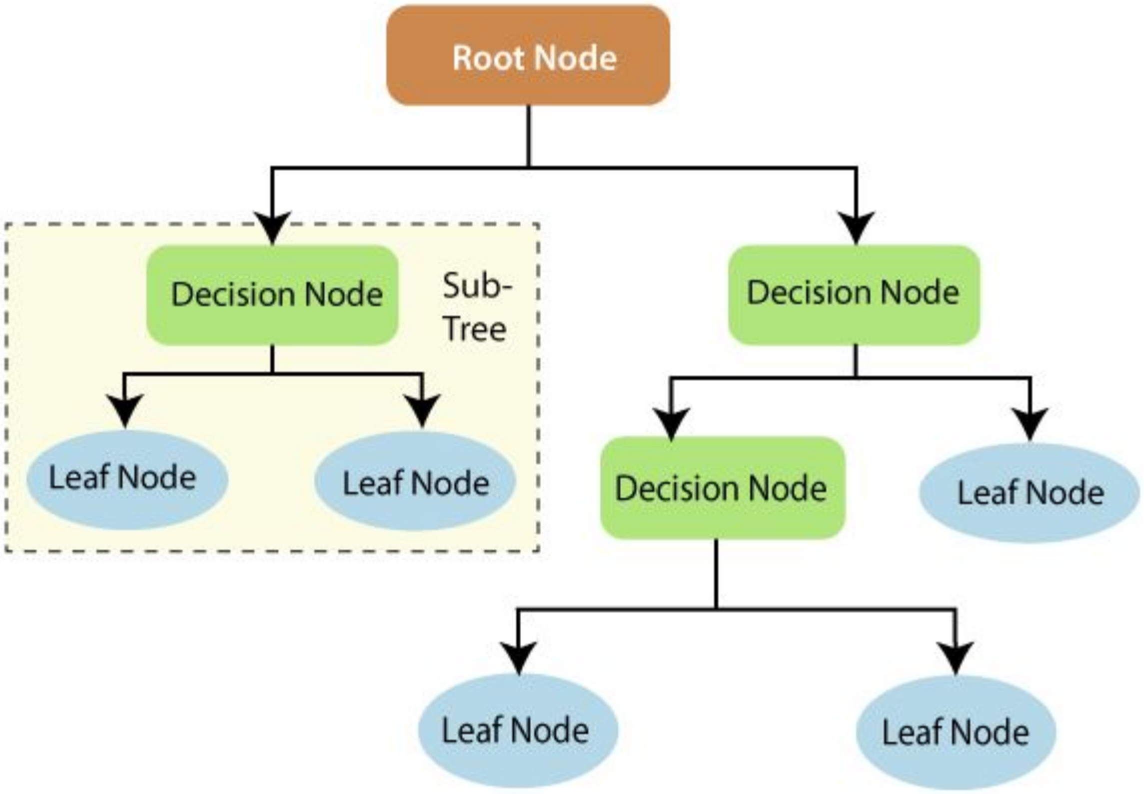
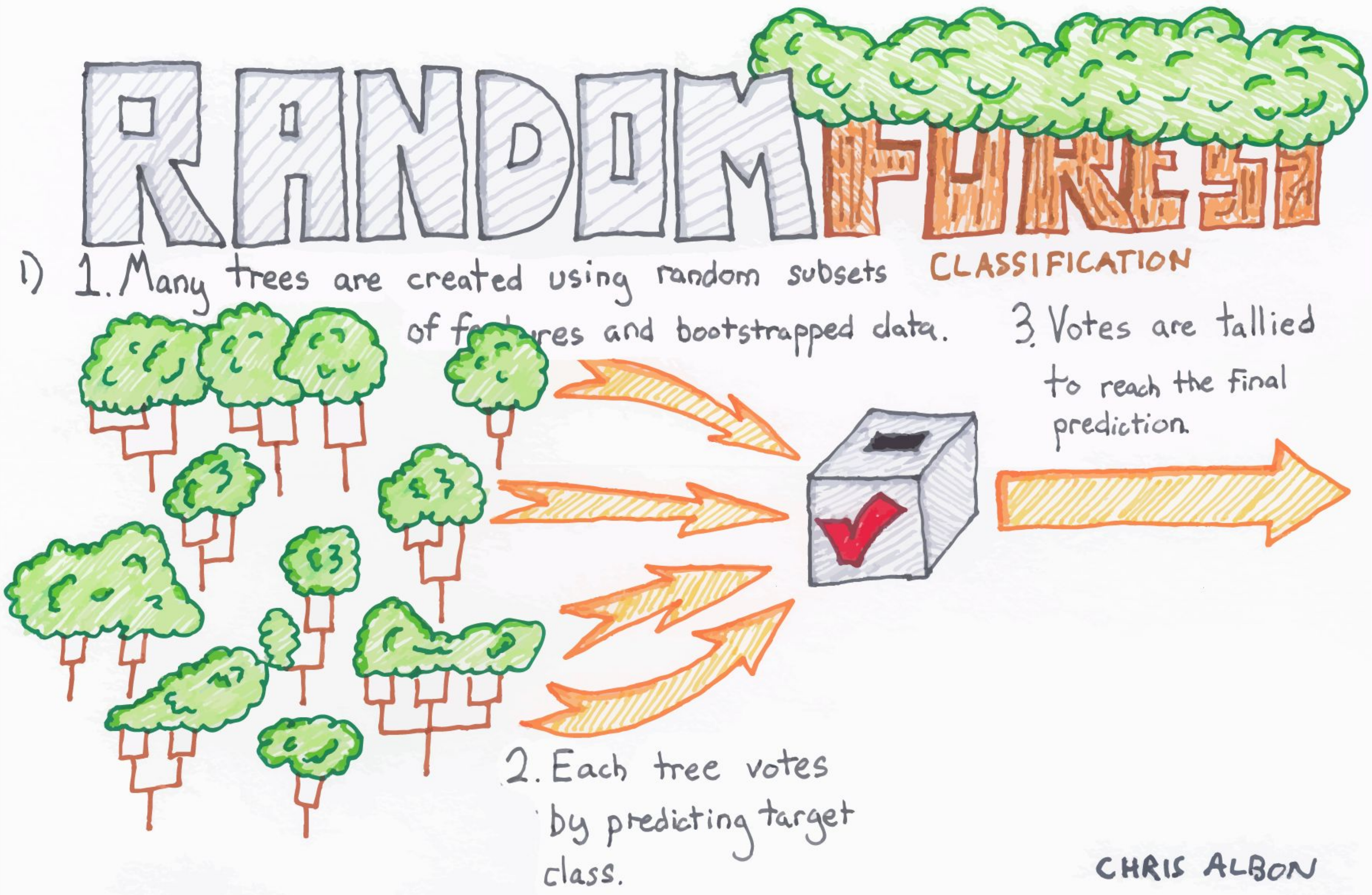


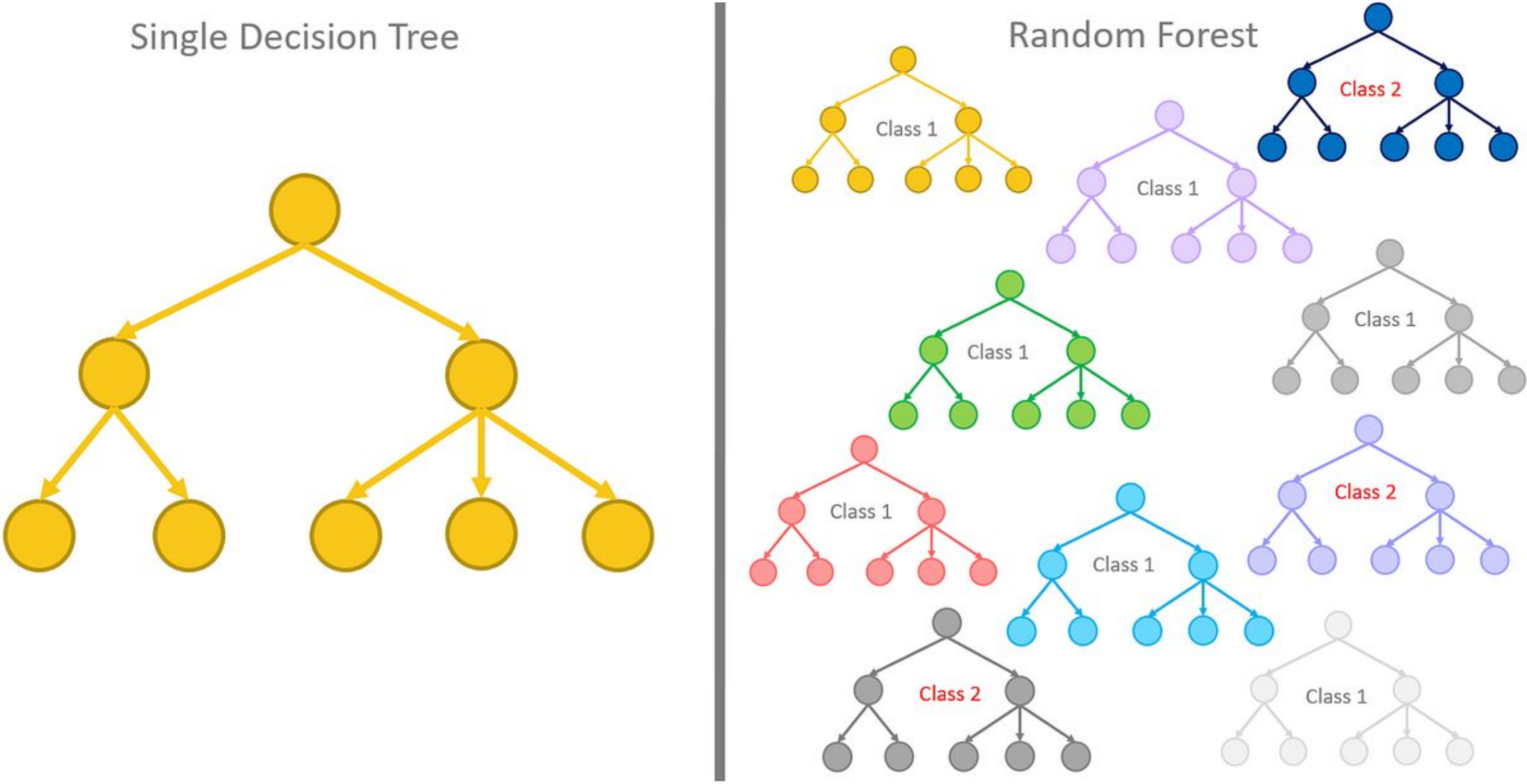
Fig-1: This is how decision tree looks.

Random Forest

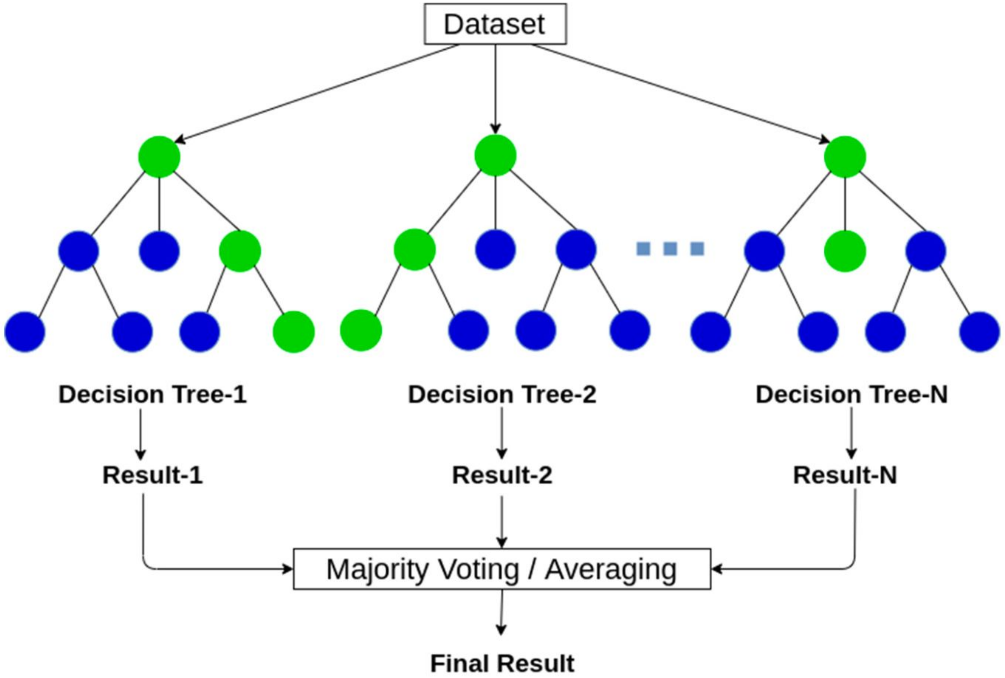
Random Forest is an ensemble learning algorithms that constructs many decision trees during the training. It predicts the mode of the classes for classification tasks and mean prediction of trees for regression tasks.

It is using random subspace method and bagging during tree construction. It has built-in feature importance.


RANDOM FOREST
CLASSIFICATION
1) 1. Many trees are created using random subsets of features and bootstrapped data.
2. Each tree votes by predicting target class.
3. Votes are tallied to reach the final prediction.
CHRIS ALBON


Single Decision Tree
Random Forest
Class 1
Class 2
Class 1
Class 1
Class 1
Class 1
Class 1
Class 2
Class 2
Class 1

Random Forest



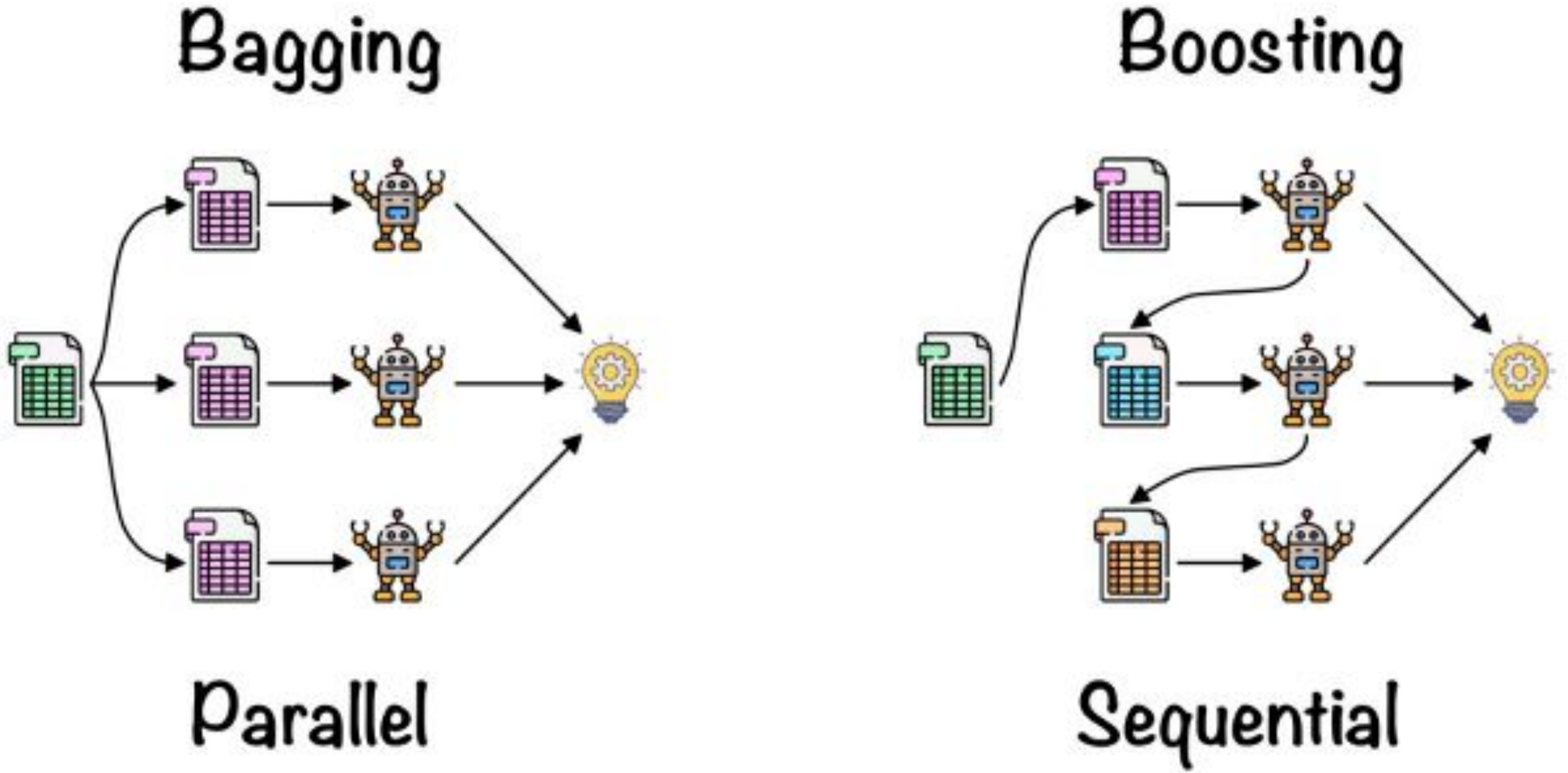
Extra Trees

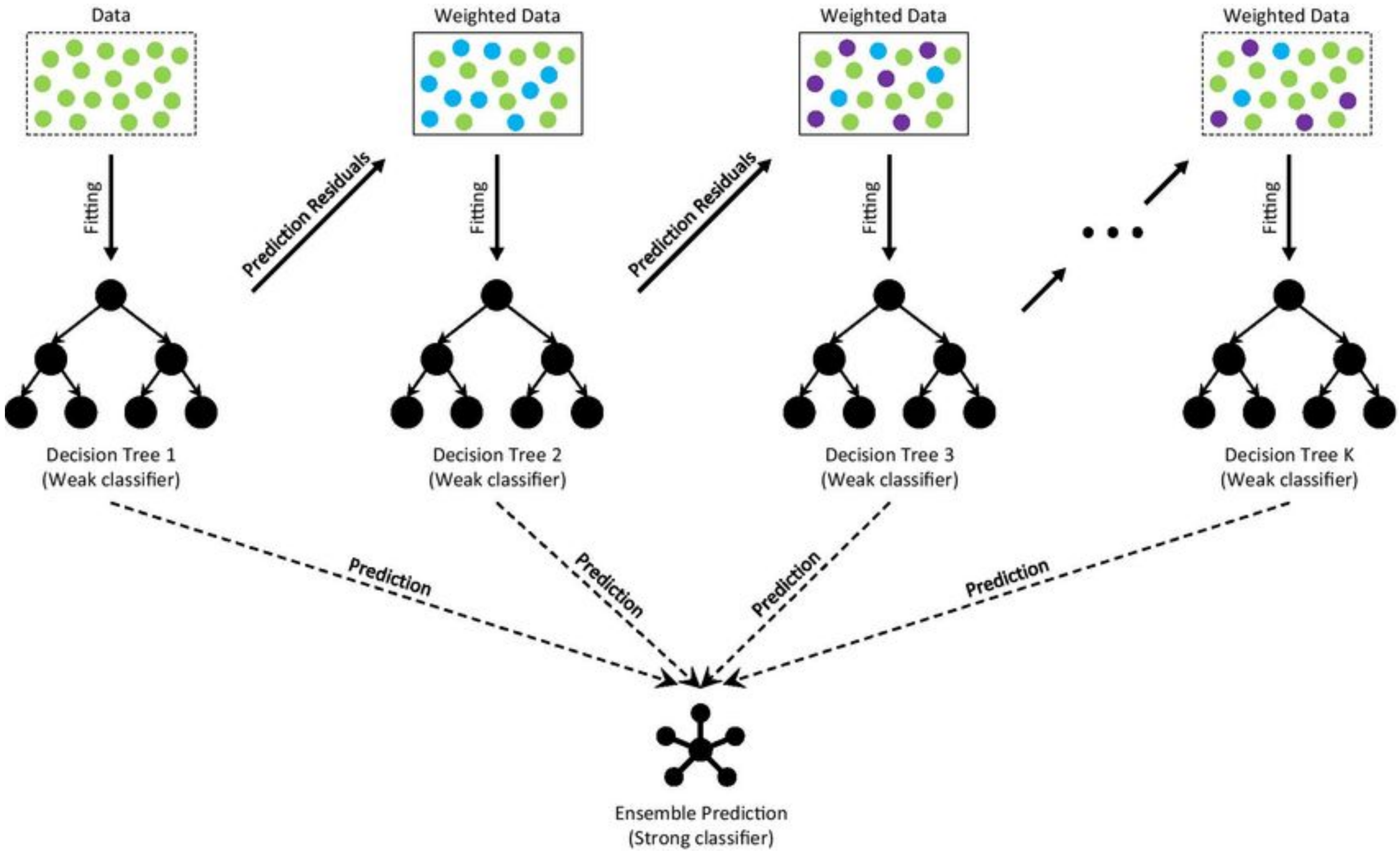
Extra Trees (Extremely Randomized Trees) the ensemble learning algorithms. It constructs the set of decision trees. During tree construction the decision rule is randomly selected. This algorithm is very similar to Random Forest except random selection of split values.

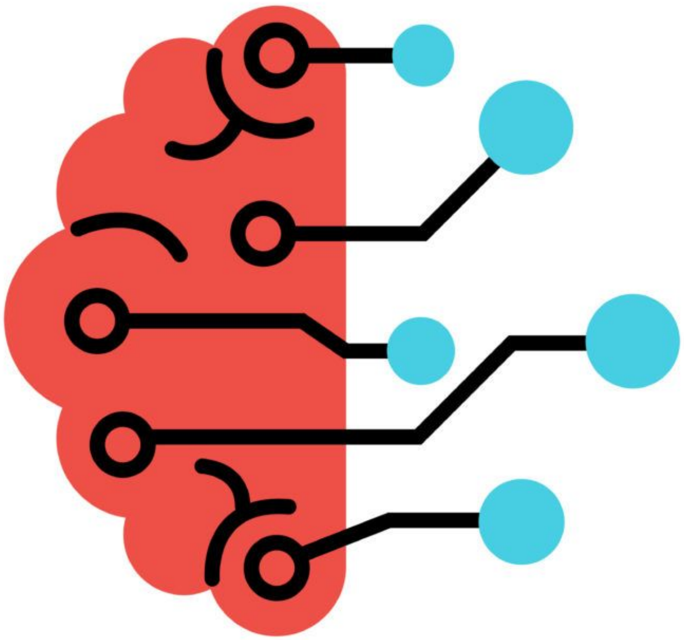
Ensemble Model

Ensemble uses two types of methods:

- **Bagging**
  - It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.
- **Boosting**
  - It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, Gradient Boosting, ADA BOOST, XG BOOST.


Bagging
Boosting
Parallel
Sequential

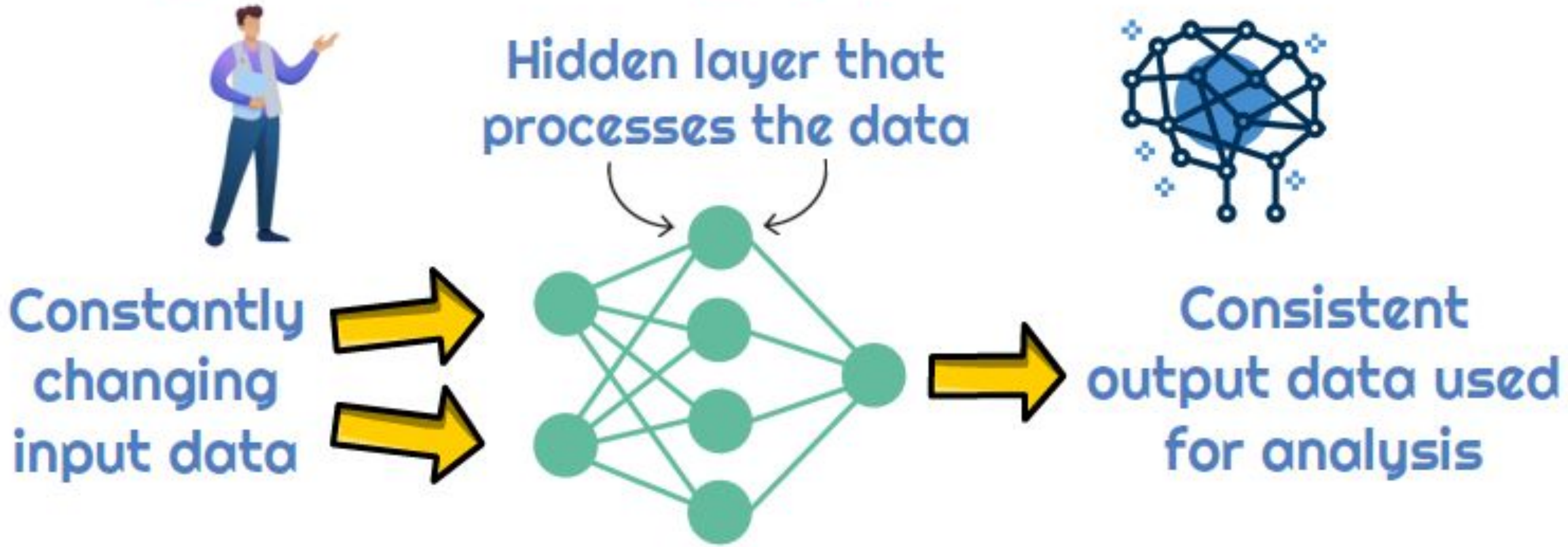

Data
Weighted Data
Weighted Data
Weighted Data
Fitting
Prediction Residuals
Fitting
Prediction Residuals
Fitting
Fitting
Decision Tree 1
(Weak classifier)
Decision Tree 2
(Weak classifier)
Decision Tree 3
(Weak classifier)
Decision Tree K
(Weak classifier)
Prediction
Prediction
Prediction
Prediction
Ensemble Prediction
(Strong classifier)



NEURAL NETWORK

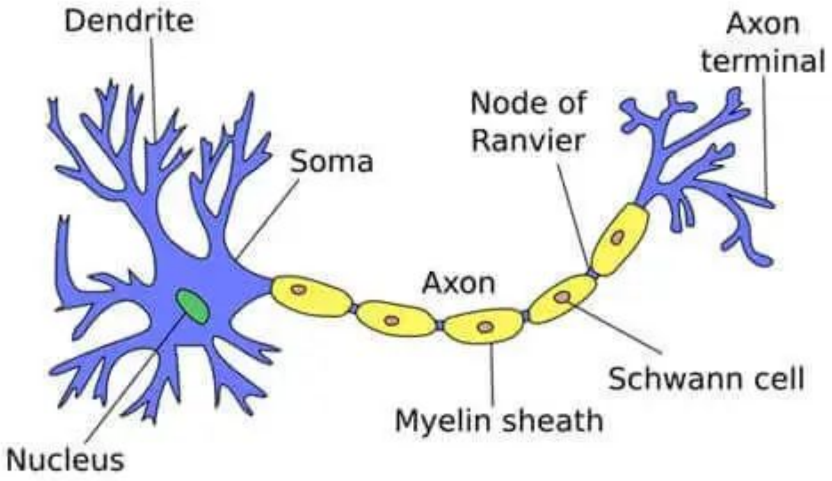
Neural Networks

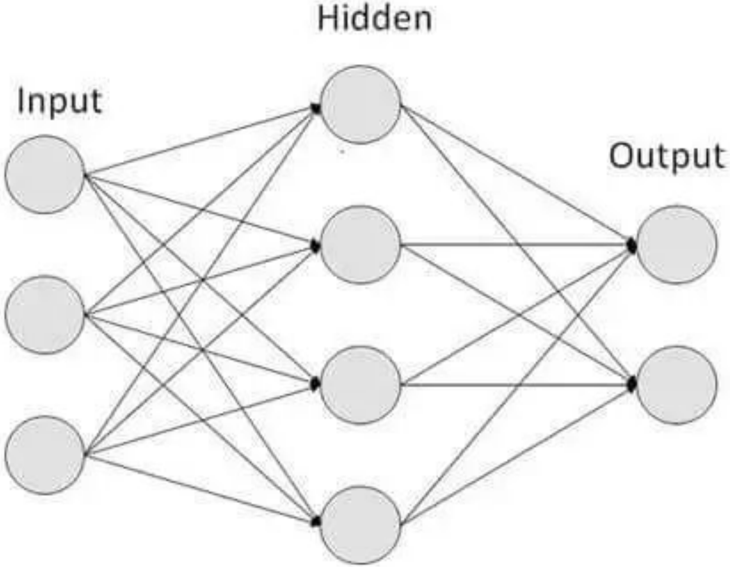
Similar to humans in that a nueral network constantly adjusts based on changing inputs or situations.

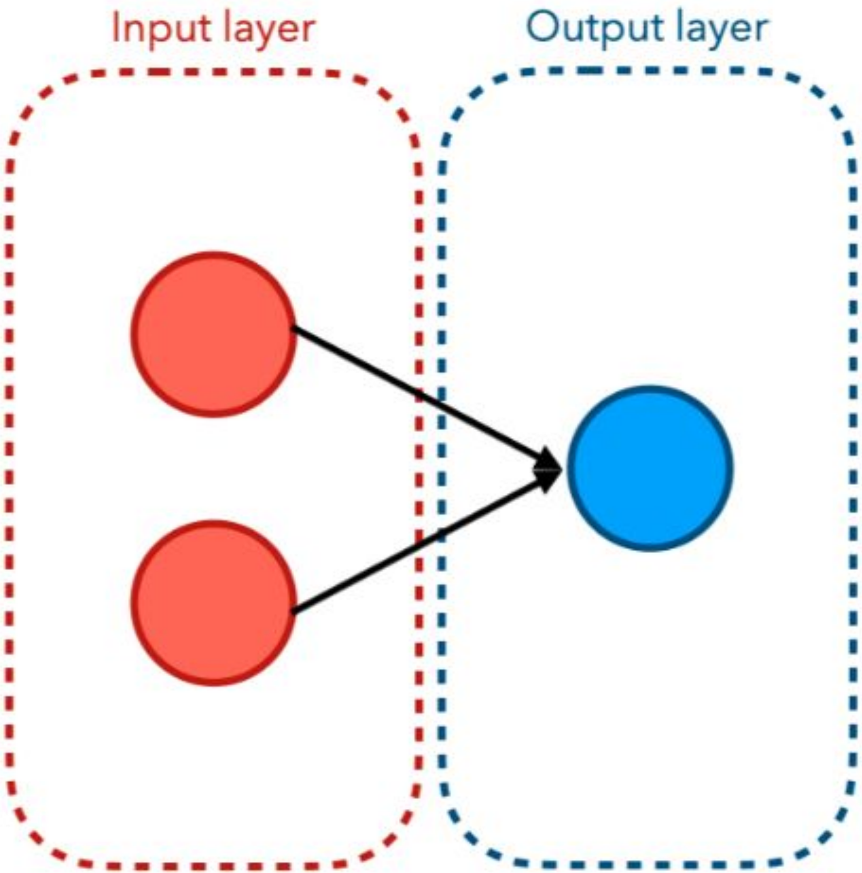


Neural Network Vs Human Brain:

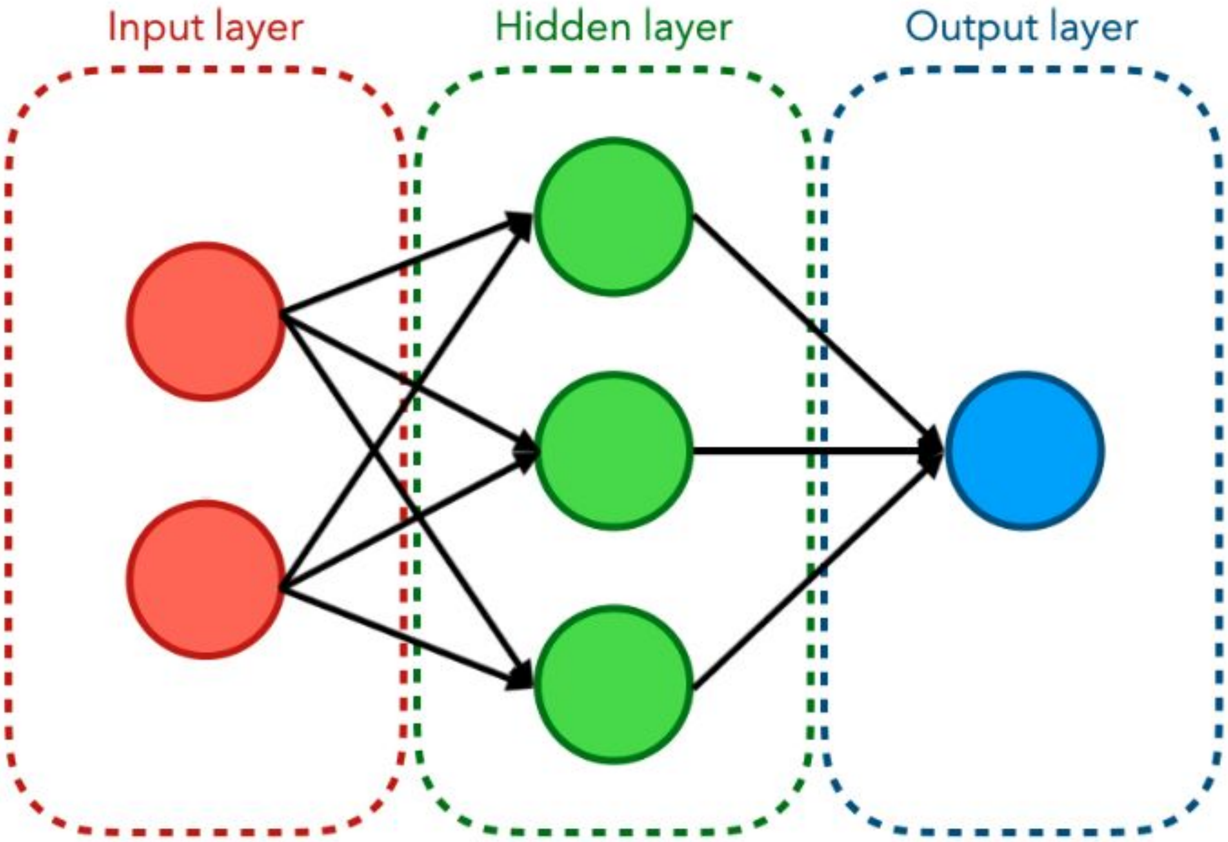
What is the Difference?

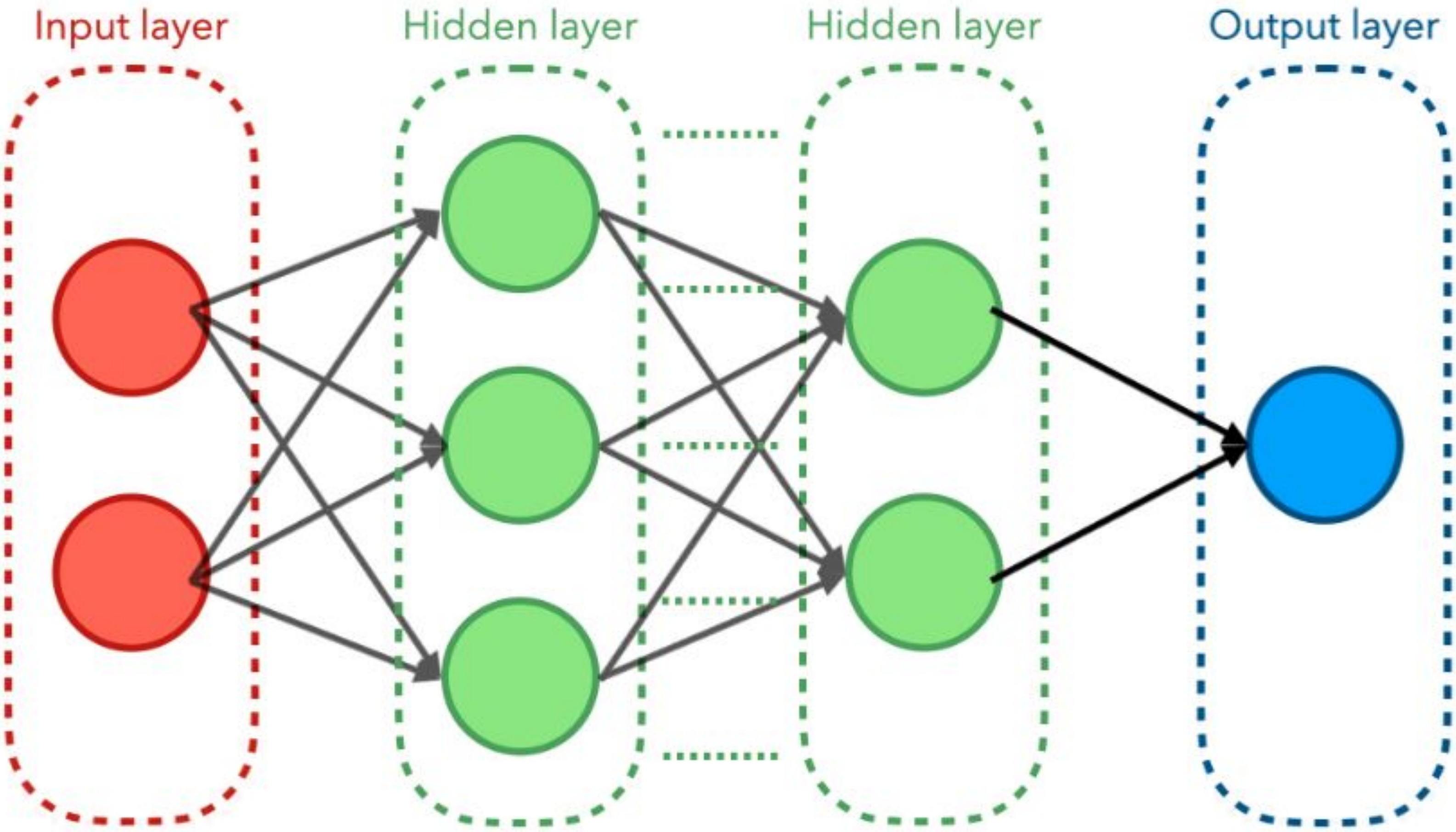

Dendrite
Axon
terminal
Node of
Ranvier
Soma
Axon
Schwann cell
Myelin sheath
Nucleus

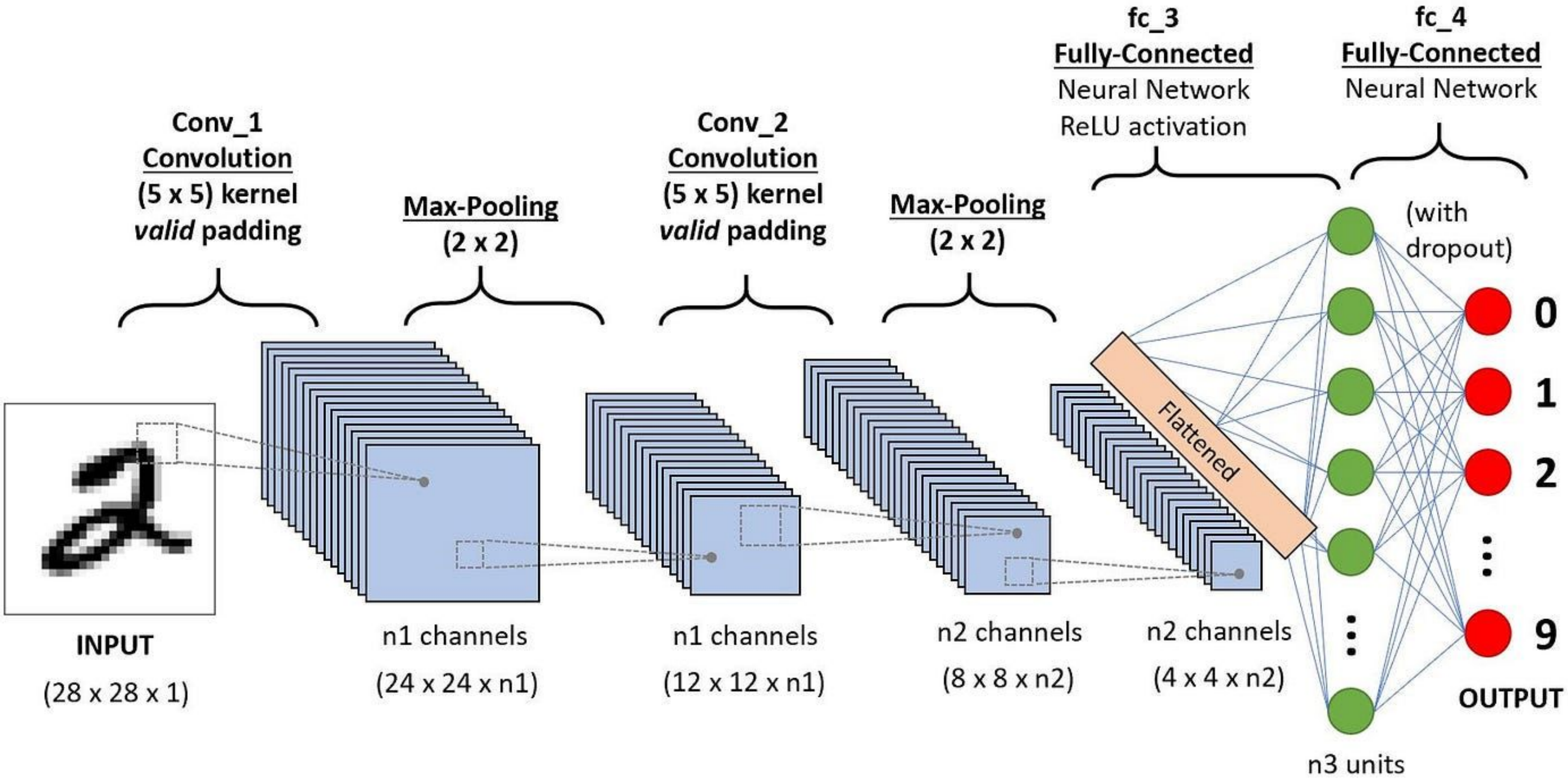

Hidden
Input
Output



Input layer
Output layer


Input layer
Hidden layer
Output layer


Input layer
Hidden layer
Hidden layer
Output layer



Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
OUTPUT

Confusion Matrix

		Predicted	
		Negative **(N)** **-**	Positive **(P)** **+**
Actual	Negative **-**	True Negative **(TN)**	**False Positive (FP)** **Type I Error**
	Positive **+**	**False Negative (FN)** **Type II Error**	True Positive **(TP)**

Accuracy

ACTUAL VALUES		POSITIVE	NEGATIVE
	POSITIVE	TP	FN
	NEGATIVE	FP	TN

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

Accuracy

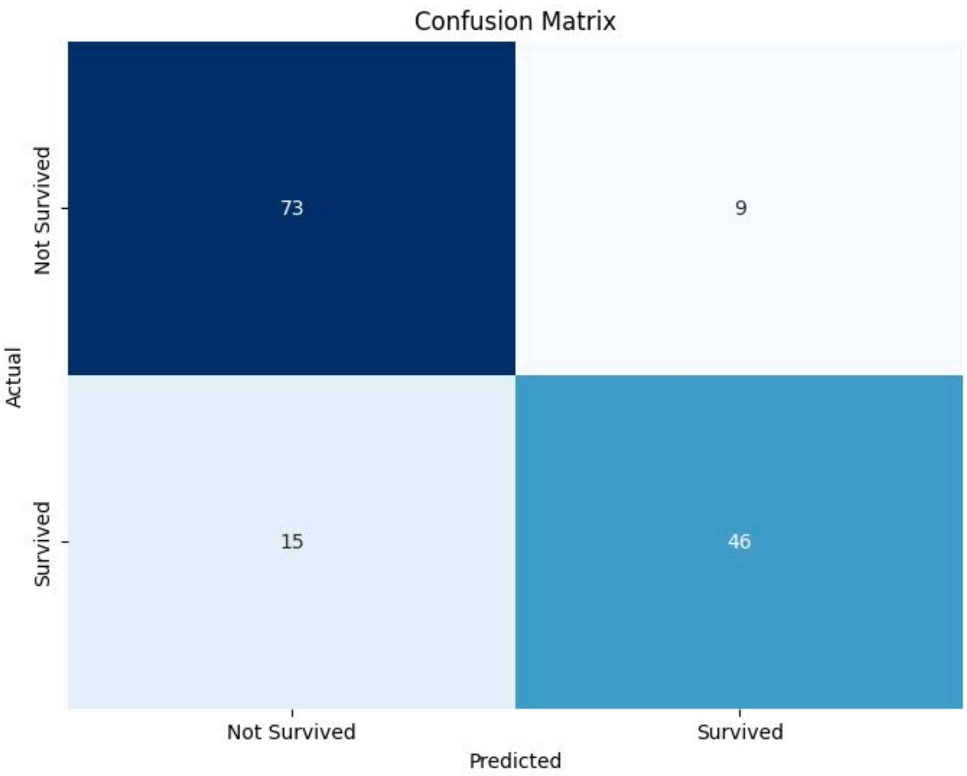
ACTUAL VALUES		POSITIVE	NEGATIVE
	POSITIVE	TP	FN
	NEGATIVE	FP	TN

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

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

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

Confusion Matrix (Sample)



[46]
```
print("\nClassification Report:")
print(classification_report(y_test, predictions,digits=4))
```

```
Classification Report:
              precision    recall  f1-score   support

           0     0.8295    0.8902    0.8588        82
           1     0.8364    0.7541    0.7931        61

    accuracy                         0.8322       143
   macro avg     0.8330    0.8222    0.8260       143
weighted avg     0.8325    0.8322    0.8308       143
```

Import Library

```
[34] import pandas as pd

     import matplotlib.pyplot as plt
     import seaborn as sns

     from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
     from sklearn.neural_network import MLPClassifier
     from sklearn.metrics import confusion_matrix, classification_report
```

```
[35] # Load the Titanic dataset
     data = pd.read_csv('titanic_dataset.csv')
```

```
[36] # Data preprocessing
     selected_features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked', 'Survived']

     data = data[selected_features].dropna()
     data = pd.get_dummies(data, columns=['Sex', 'Embarked'])

     X = data.drop('Survived', axis=1)
     y = data['Survived']
```

```
✓ 0s
[37] # Split the data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2023)
```

0s [38]

```python
# Initialize models
models = {
    "DecisionTree": DecisionTreeClassifier(random_state=42, max_depth=5),
    "RandomForest": RandomForestClassifier(random_state=42, max_depth=5),
    "GradientBoost": GradientBoostingClassifier(random_state=42, max_depth=5),
    "NeuralNetwork": MLPClassifier(random_state=42, max_iter=1000, solver='adam')
}
```

✓ 1s [39]

```python
# Train and evaluate models
results = {'Model': [], 'Accuracy': []}

for name, model in models.items():
    model.fit(X_train, y_train)
    accuracy = model.score(X_test, y_test)
    results['Model'].append(name)
    results['Accuracy'].append(accuracy)
```

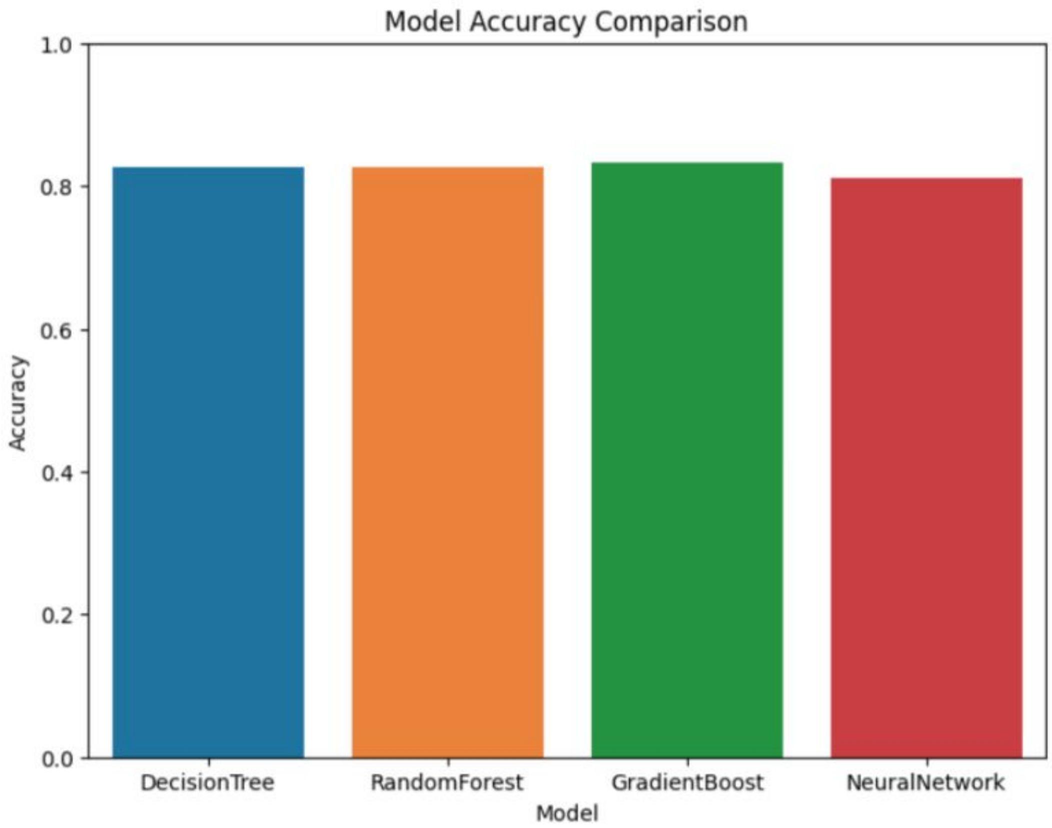
[40]

```python
# Create DataFrame for results
results_df = pd.DataFrame(results)

# Display the accuracy comparison
results_df
```

	Model	Accuracy
0	DecisionTree	0.825175
1	RandomForest	0.825175
2	GradientBoost	0.832168
3	NeuralNetwork	0.811189

```python
# Plotting the accuracy comparison
plt.figure(figsize=(8, 6))
sns.barplot(x='Model', y='Accuracy', data=results_df)
plt.title('Model Accuracy Comparison')
plt.ylim(0, 1)
plt.show()
```



Week 6: Assignment (Part II)

Predicting Heart Attack Risk Using **4 Machine Learning Models**

You are given a dataset containing various attributes of patients. Your task is to create a machine learning model using Machine learning Classifiers to predict the likelihood of a heart attack for a patient based on their medical attributes.