SC310005 Artificial Intelligence

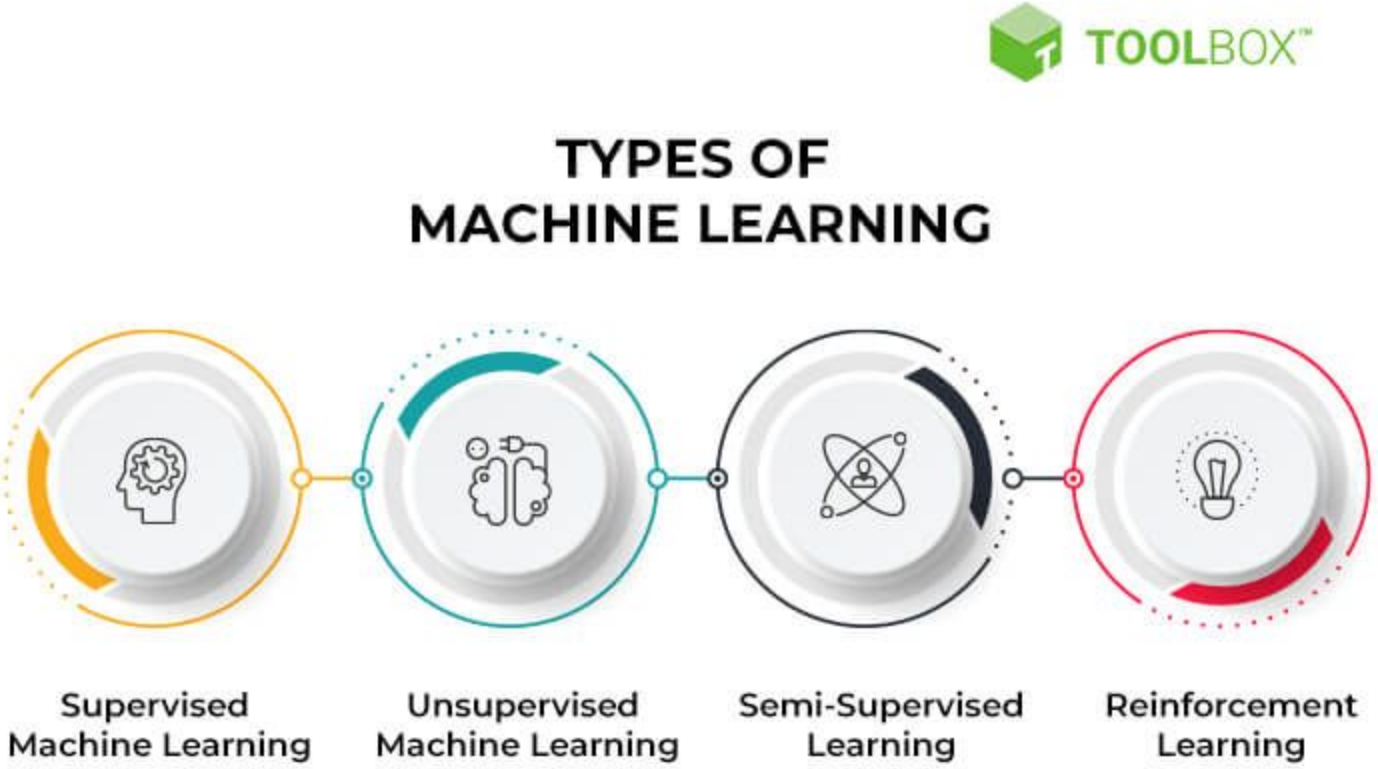
Lecture 5: Supervised Learning (Part I)

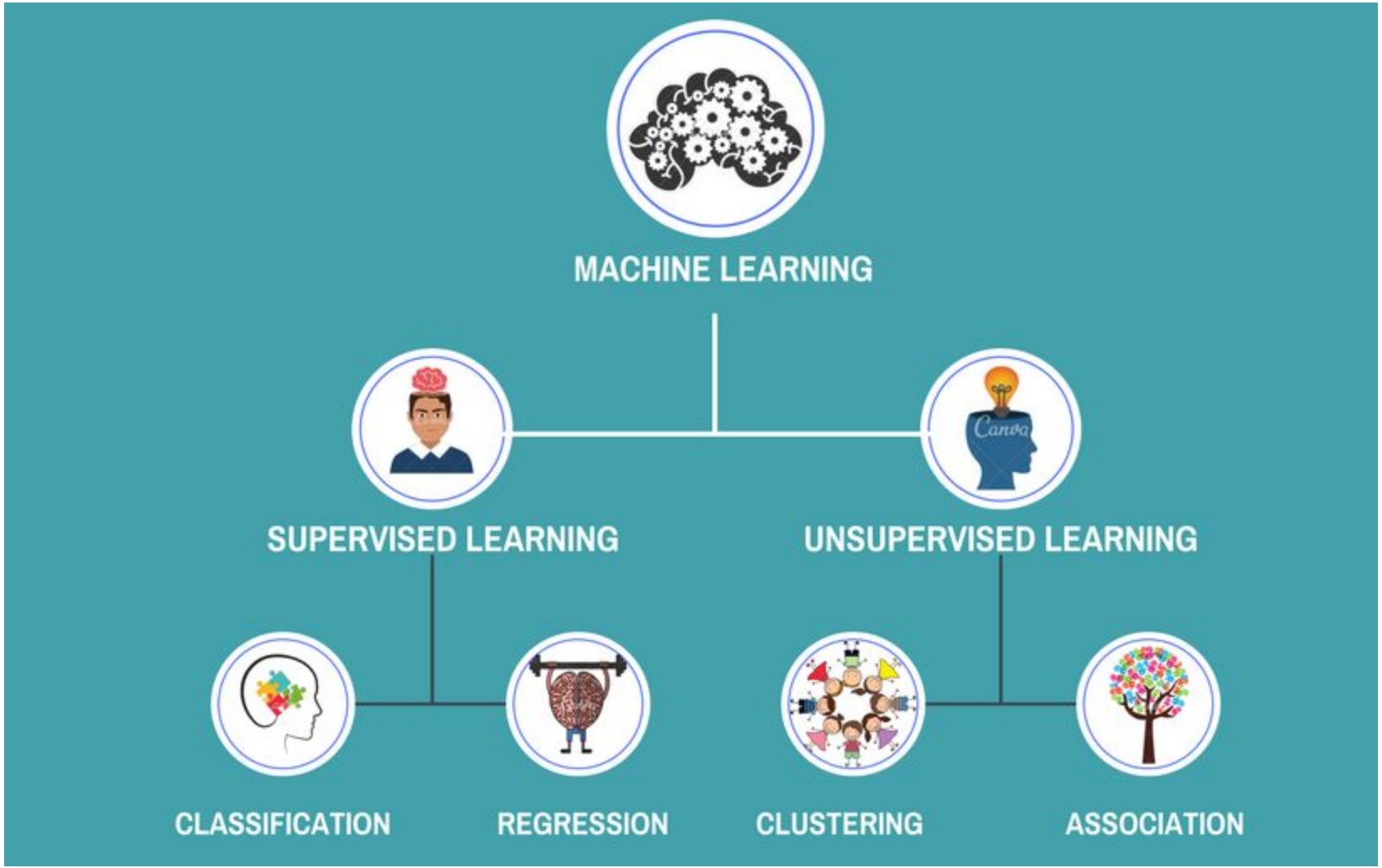
teerapong.pa@chula.ac.th

Reference

- https://www.labellerr.com/blog/supervised-vs-unsupervised-learning-whats-the-difference/
- https://www.turing.com/kb/importance-of-decision-trees-in-machine-learning
- https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052
- https://www.section.io/engineering-education/hyperparmeter-tuning/
- https://medium.com/titansoft-engineering/decision-tree-easily-explained-c775c80abecb

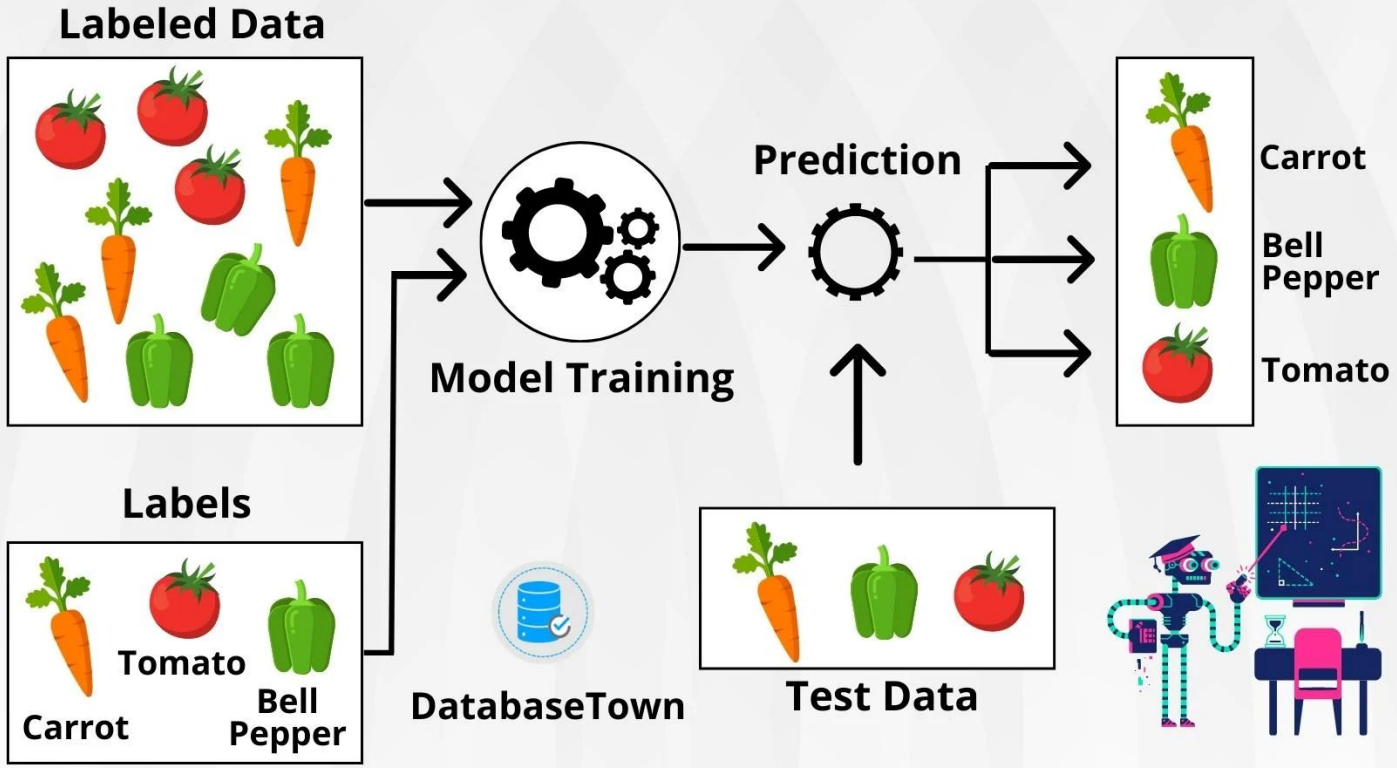
https://www.superannotate.com/blog/supervised-learning-and-other-machine-learning-tasks

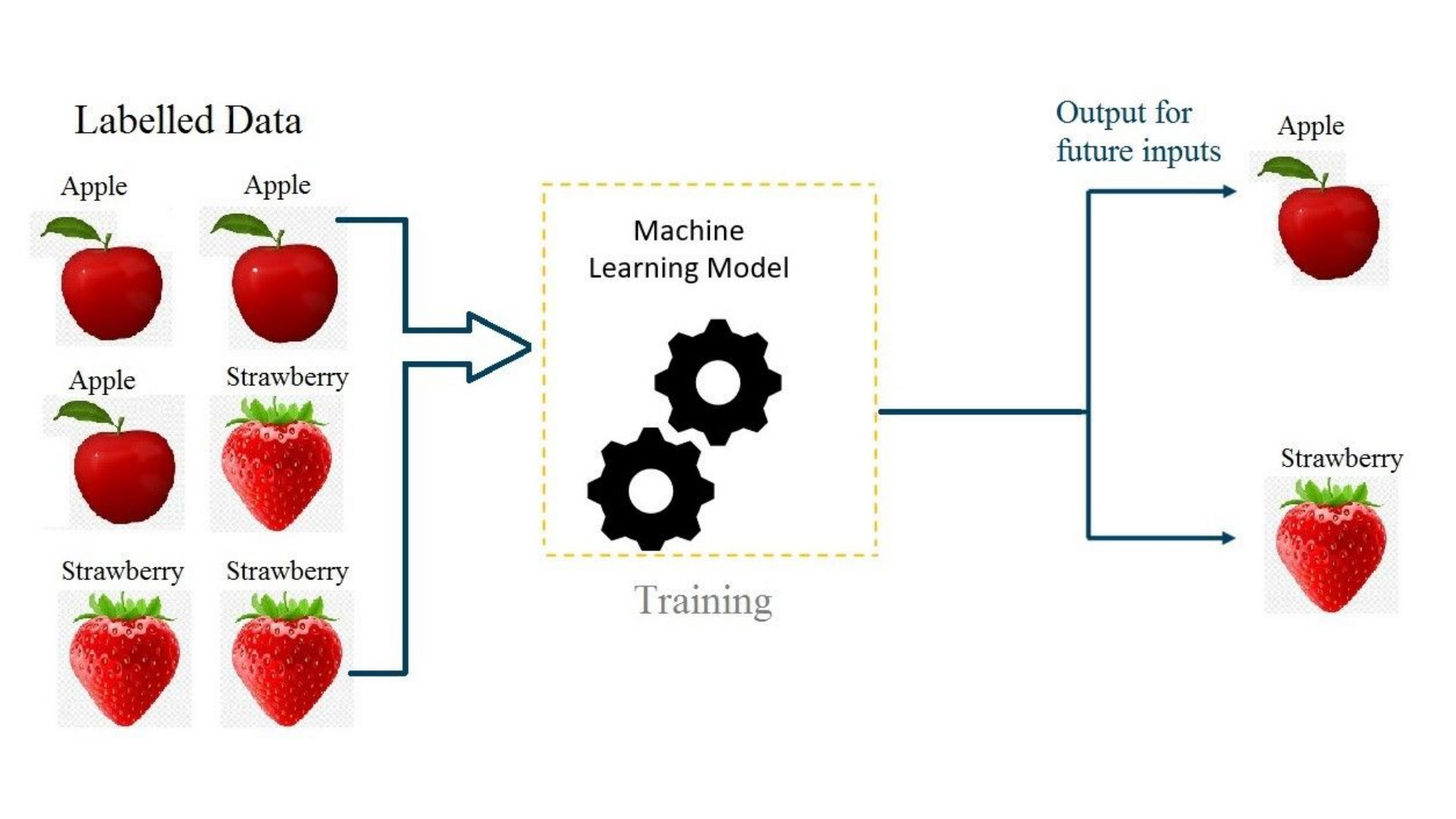



MACHINE LEARNING
SUPERVISED LEARNING
UNSUPERVISED LEARNING
CLASSIFICATION
REGRESSION
CLUSTERING
ASSOCIATION

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.

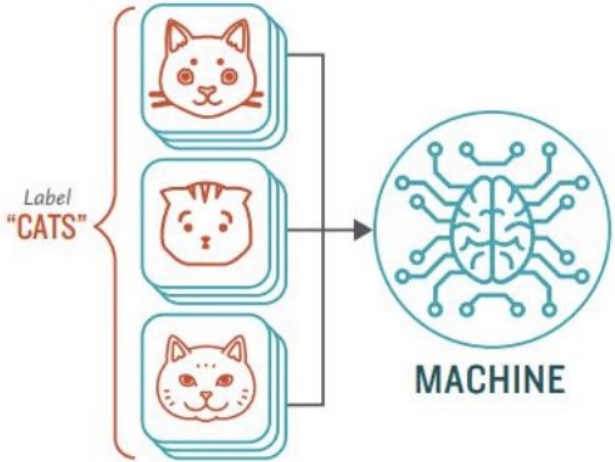



Labelled Data
Apple
Apple
Apple
Strawberry
Strawberry
Strawberry
Machine
Learning Model
Training
Output for
future inputs
Apple
Strawberry

How **Supervised** Machine Learning Works

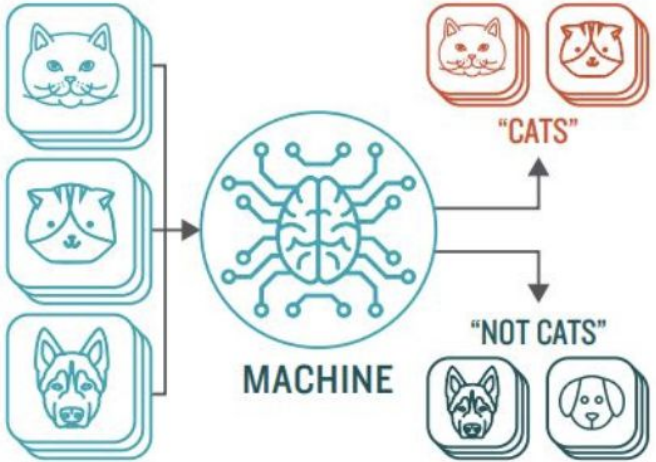
STEP 1

Provide the machine learning algorithm categorized or "labeled" input and output data from to learn

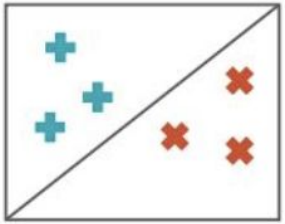


STEP 2

Feed the machine new, unlabeled information to see if it tags new data appropriately. If not, continue refining the algorithm

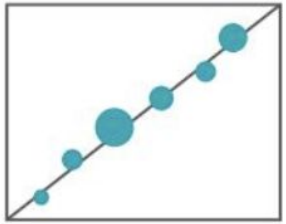


TYPES OF PROBLEMS TO WHICH IT'S SUITED



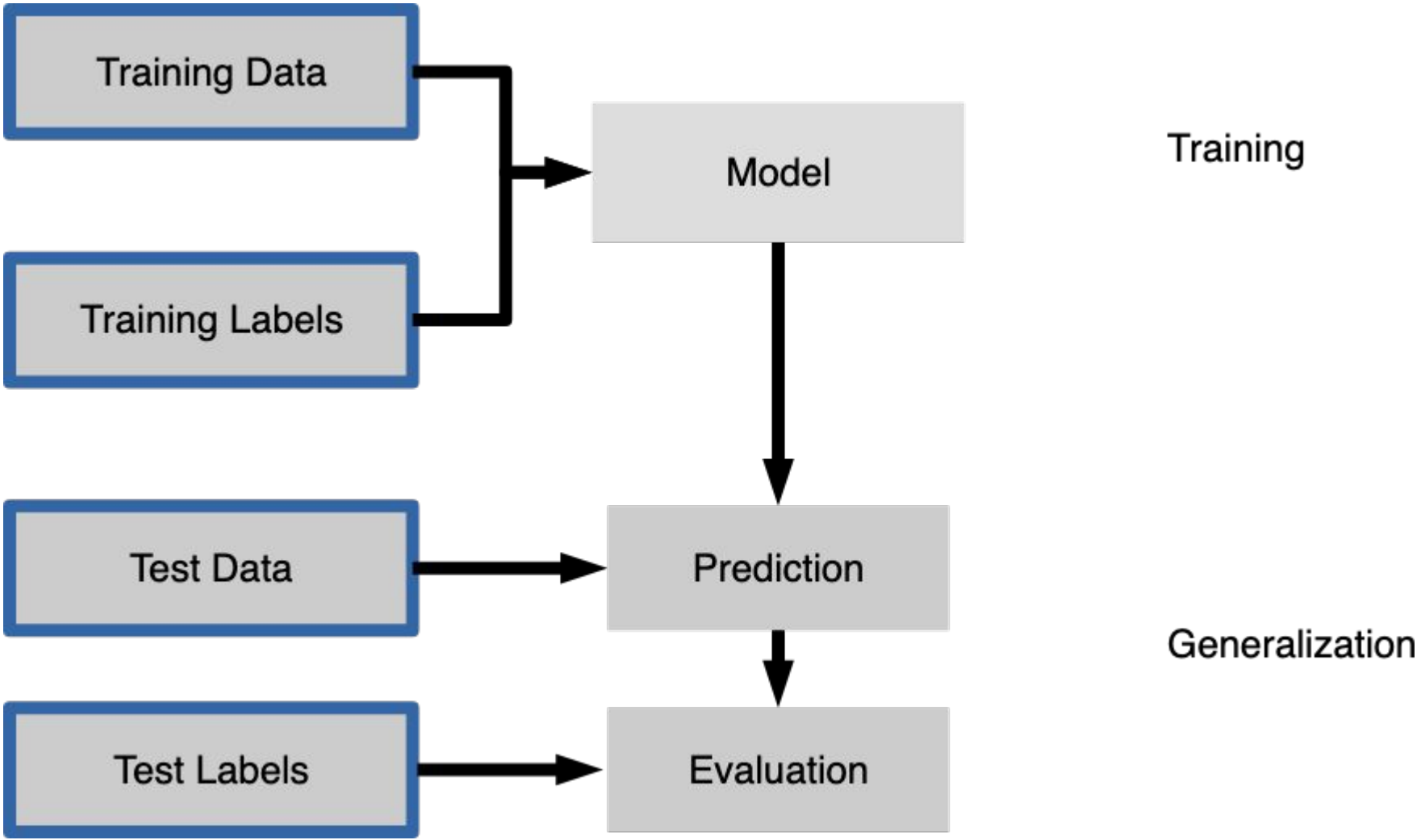
CLASSIFICATION

Sorting items into categories

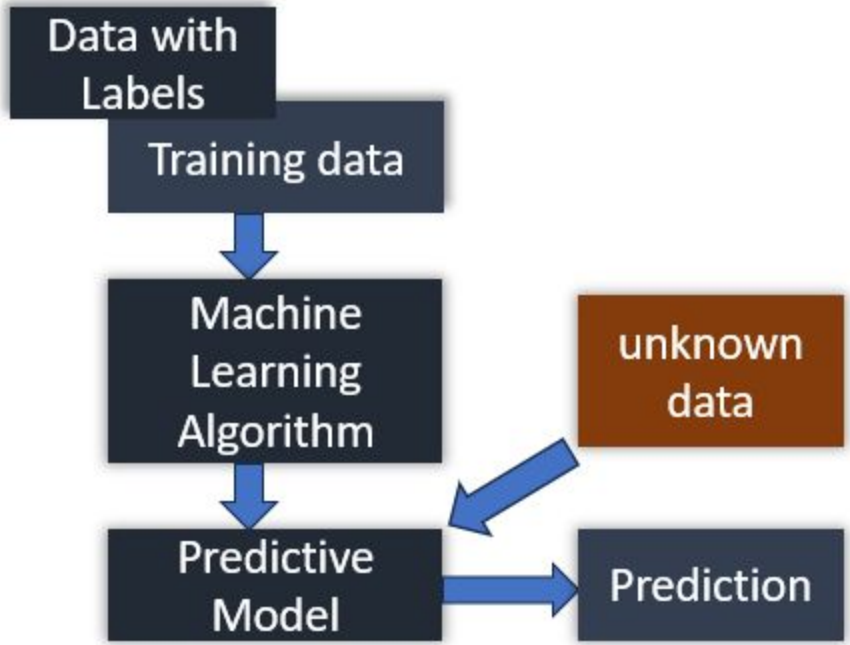


REGRESSION

Identifying real values (dollars, weight, etc.)


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

Supervised Learning



machine learning in Python

KDnuggets | Visit KDnuggets.com for more cheatsheets and additional learning resources.

Scikit-learn CheatSheet

Scikit-learn is an open-source Python library for all kinds of predictive data analysis. You can perform classification, regression, clustering, dimensionality reduction, model tuning, and data preprocessing tasks.

Loading the Data

Classification

```
from sklearn import datasets
X, y = datasets.load_wine(return_X_y=True)
```

Regression

```
diabetes = datasets.load_diabetes()
X, y = diabetes.data, diabetes.target
```

Training And Test Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0
)
```

Preprocessing the Data

Standardization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

Normalization

```
from sklearn.preprocessing import Normalizer
norm = Normalizer()
norm_X_train = norm.fit_transform(X_train)
norm_X_test = norm.transform(X_test)
```

Binarization

```
from sklearn.preprocessing import Binarizer
binary = Binarizer(threshold=0.0)
binary_X = binary.fit_transform(X)
```

Encoding Categorical Features

```
from sklearn.preprocessing import LabelEncoder
lab_enc = LabelEncoder()
y = lab_enc.fit_transform(y)
```

Imputer

```
from sklearn.impute import SimpleImputer
imp_mean = SimpleImputer(missing_values=0,
strategy='mean')
imp_mean.fit_transform(X_train)
```

Supervised Learning Model

Linear Regression

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

Support Vector Machines

```
from sklearn.svm import SVC
svm_svc = SVC(kernel='linear')
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
```

Unsupervised Learning Model

Principal Component Analysis

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
```

K Means

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, random_state=0)
```

Model Fitting

Supervised Learning

```
lr.fit(X_train, y_train)
svm_svc.fit(X_train, y_train)
```

Unsupervised Learning

```
model = pca.fit_transform(X_train)
kmeans.fit(X_train)
```

Prediction

Supervised Learning

```
y_pred = lr.predict_proba(X_test)
y_pred = svm_svc.predict(X_test)
```

Unsupervised Learning

```
y_pred = kmeans.predict(X_test)
```

Evaluation

Accuracy Score

```
lr.score(X_test, y_test)

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Mean Squared Error

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
```

R2 Score

```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Adjusted Rand Index

```
from sklearn.metrics import adjusted_rand_score
adjusted_rand_score(y_test, y_pred)
```

Cross-Validation

```
from sklearn.model_selection import cross_val_score
cross_val_score( lr, X, y, cv=5, scoring='f1_macro')
```

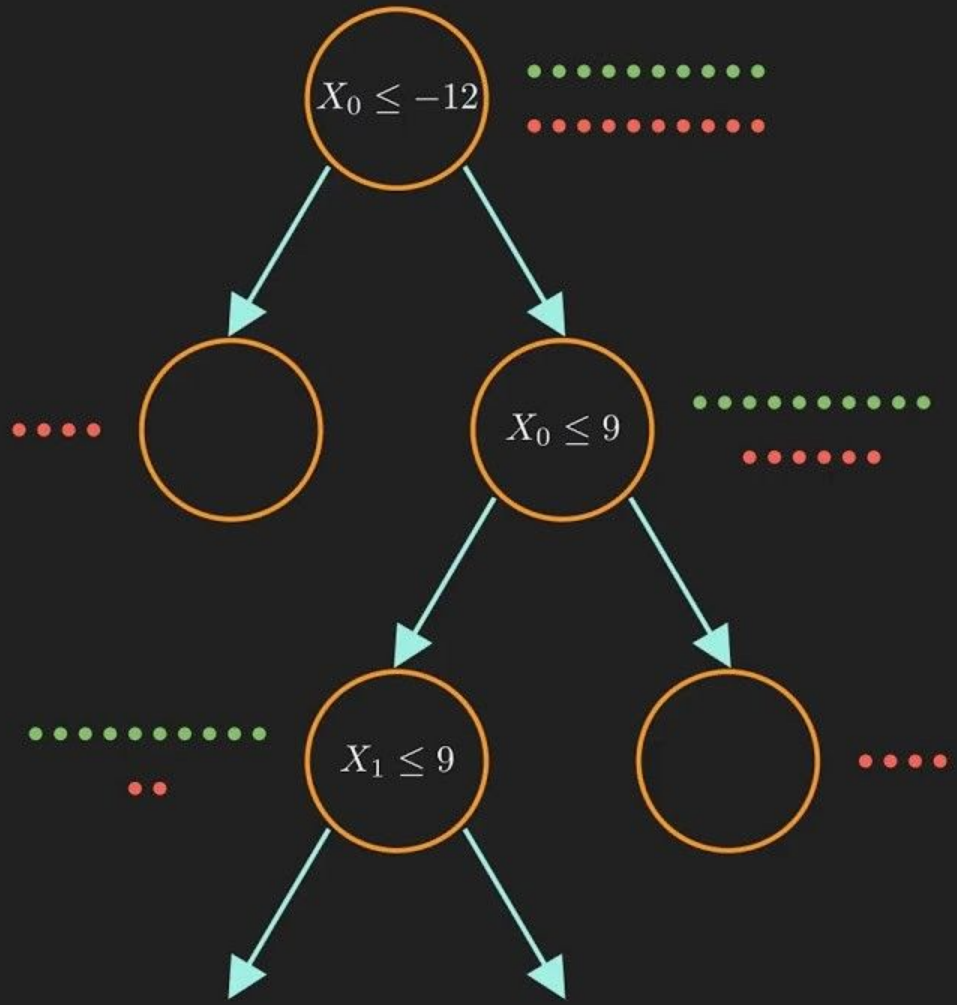
Model Tuning

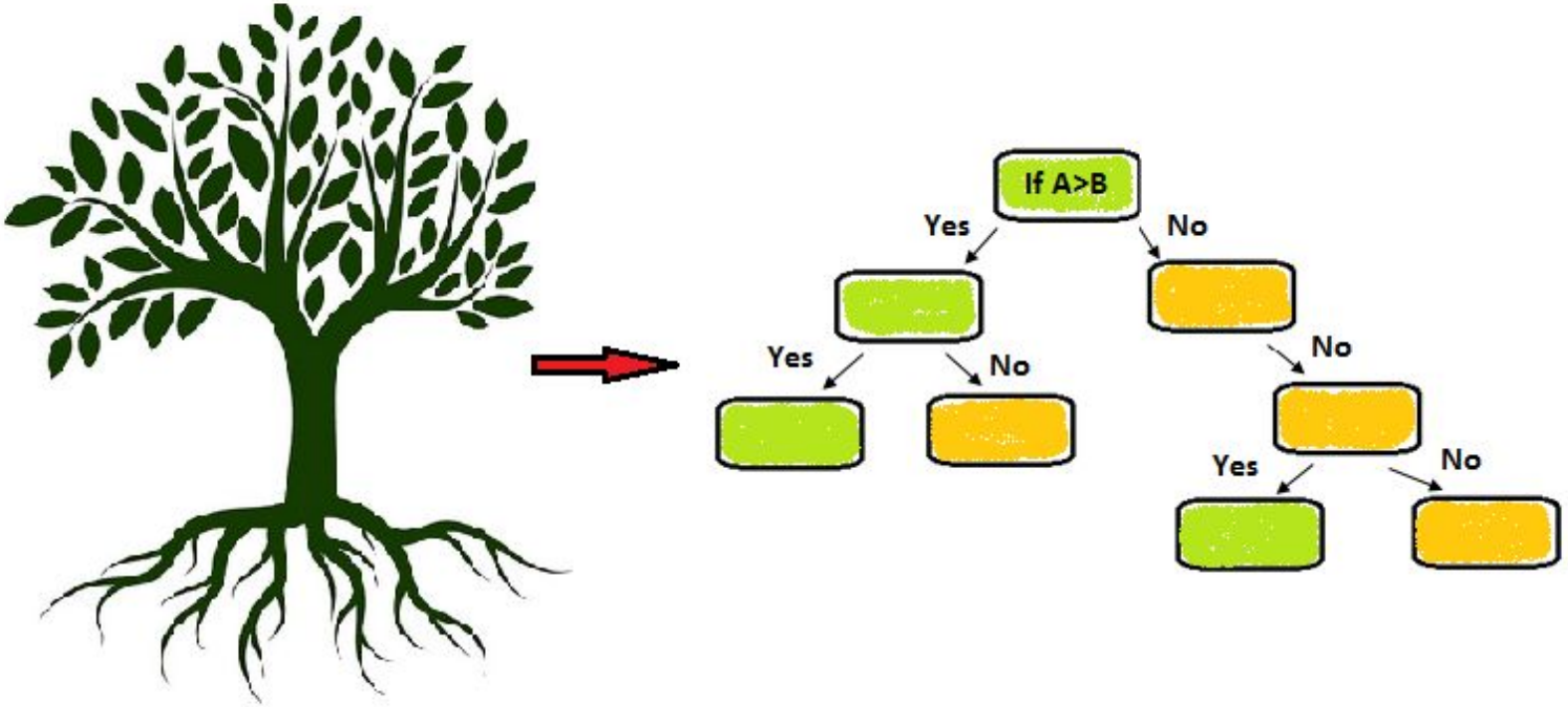
```
from sklearn.model_selection import GridSearchCV
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
model = GridSearchCV(svm_svc, parameters)
model.fit(X_train, y_train)
print(model.best_score_)
print(model.best_estimator_)
```

Subscribe to KDnuggets News

Abid Ali Awan | 2022

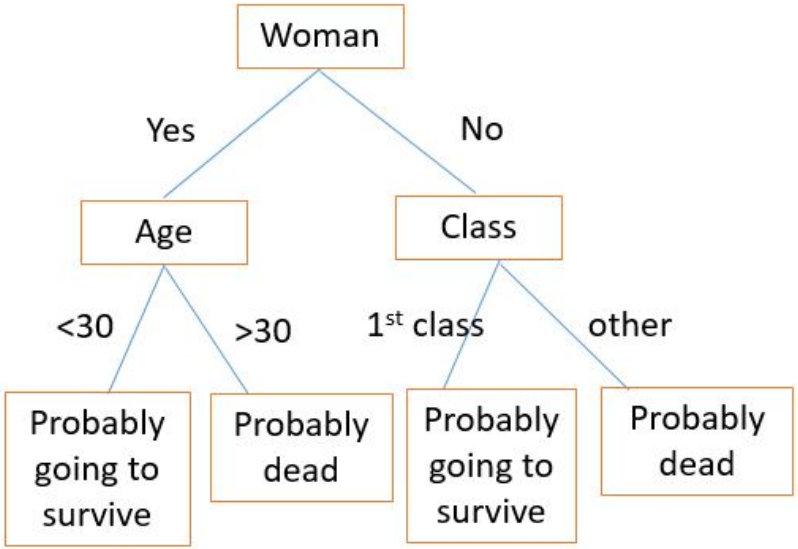
Decision Tree Classifier

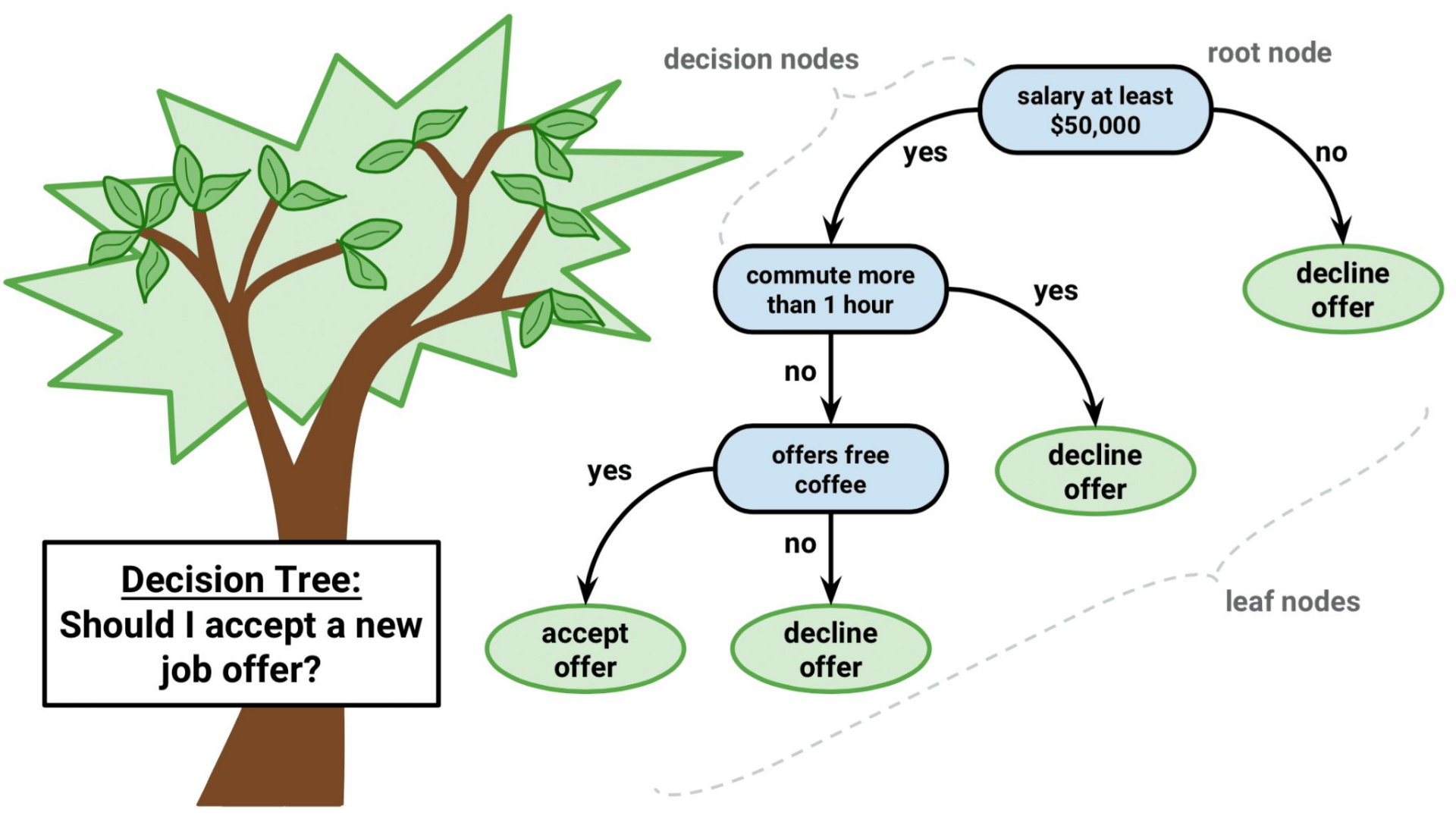



If A>B
Yes
No
Yes
No
No
Yes
No

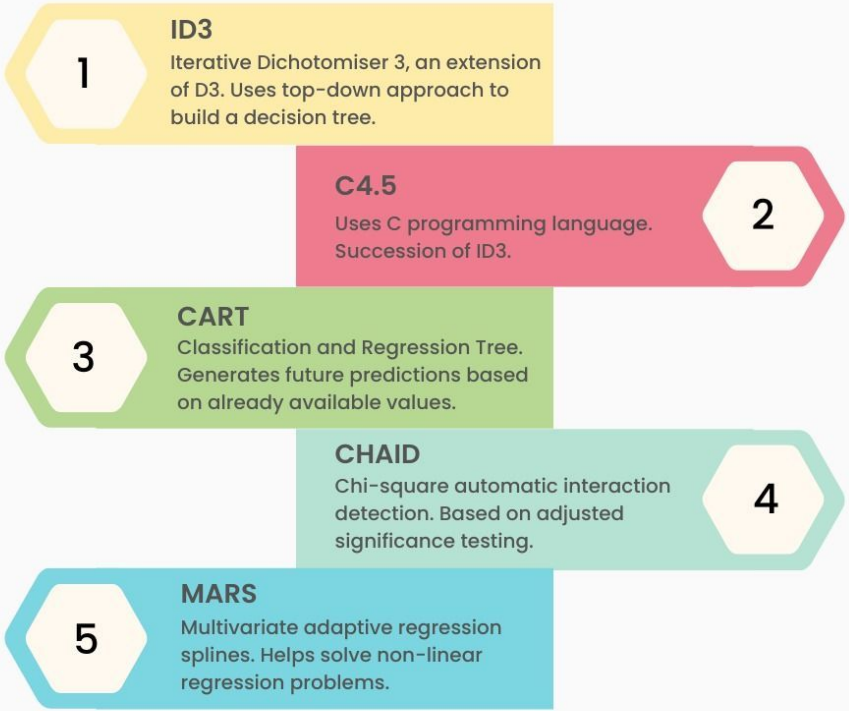
A normal tree

A decision tree

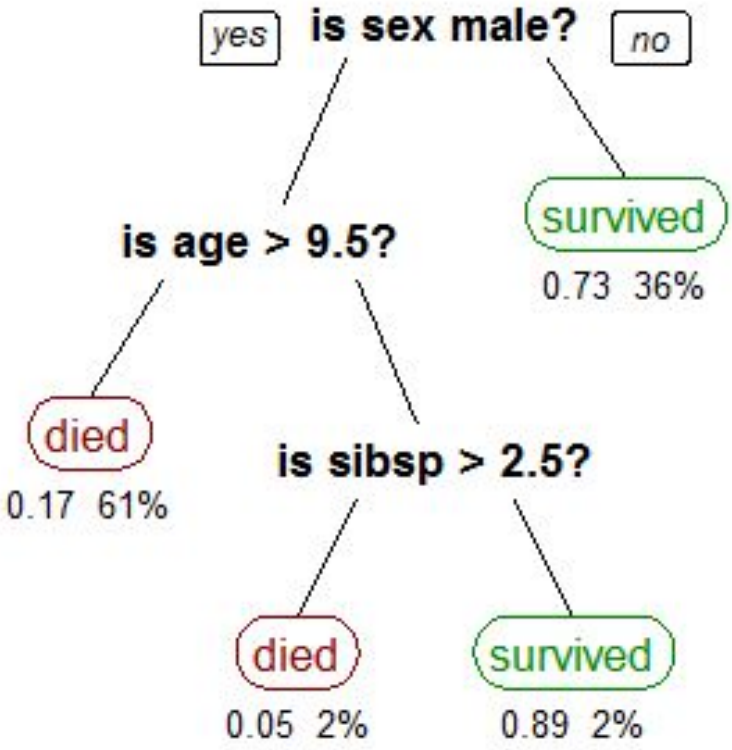



Decision Tree:
Should I accept a new job offer?
decision nodes
root node
salary at least $50,000
yes
no
commute more than 1 hour
yes
decline offer
no
offers free coffee
yes
decline offer
no
accept offer
decline offer
leaf nodes

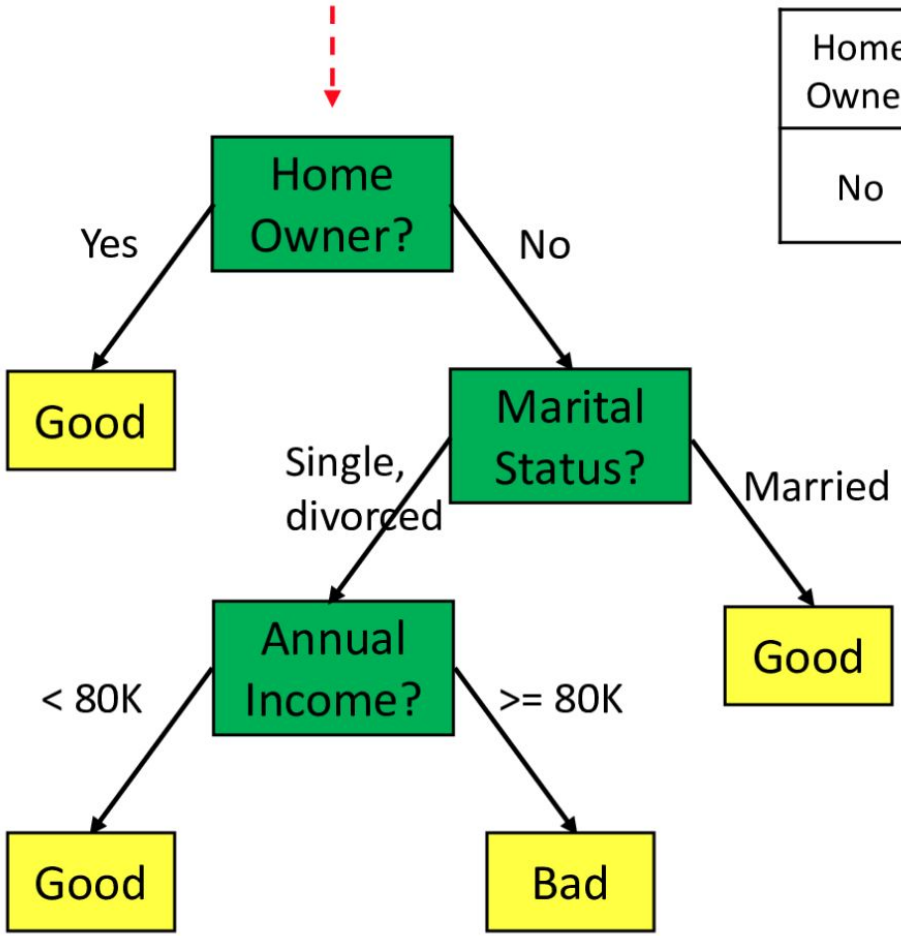
Algorithms used in Decision Trees



Turing

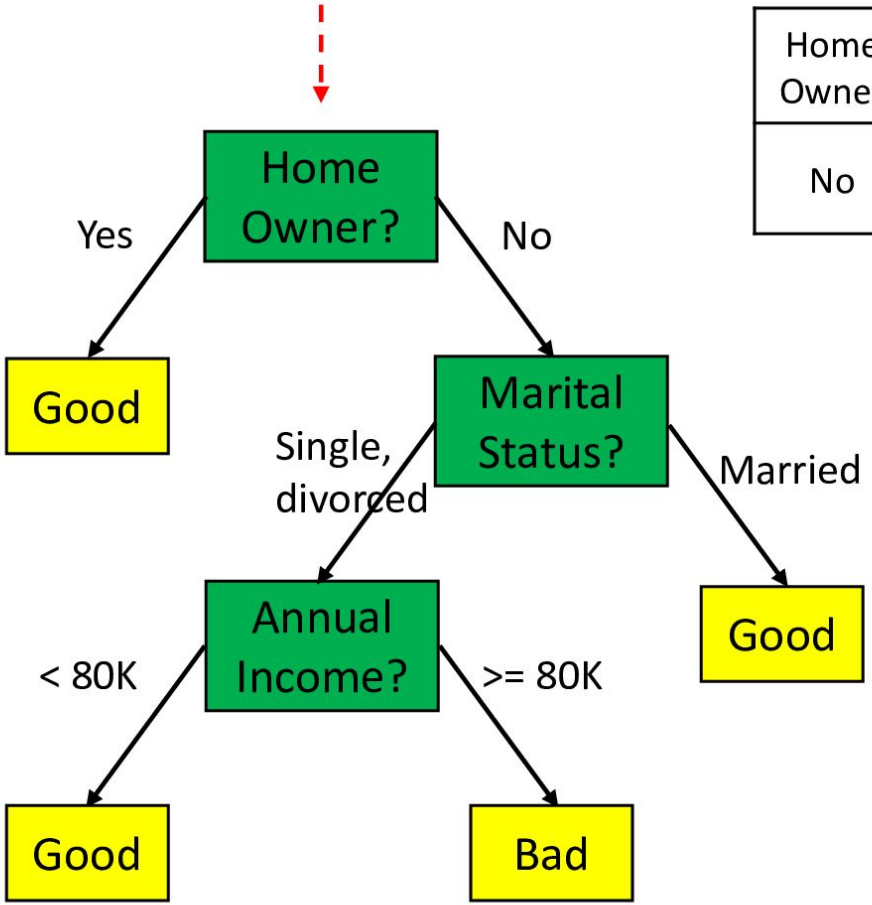

yes
is sex male?
no
is age > 9.5?
survived
0.73 36%
died
0.17 61%
is sibsp > 2.5?
died
0.05 2%
survived
0.89 2%

Start from the root node.



Home Owner	Marital Status	Annual Income	Credit Rating
No	Married	80K	?

Start from the root node.



Home Owner	Marital Status	Annual Income	Credit Rating
No	Married	80K	?

Install the required libraries

✓ 10s

```
[1]  # Remember to install the required libraries (pandas, sklearn, matplotlib) if you haven't already done so:

     !pip install pandas scikit-learn matplotlib --quiet
```

Libraries

✓ 2s

```
[2]  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.metrics import confusion_matrix, classification_report
     from sklearn import tree
```

```
[3]  # Load the Titanic dataset
     # Replace 'path_to_dataset' with the actual path of your dataset

     data = pd.read_csv('titanic_dataset.csv')
```

Load the Titanic dataset

```
[3]  # Load the Titanic dataset
     # Replace 'path_to_dataset' with the actual path of your dataset

     data = pd.read_csv('titanic_dataset.csv')
```

Display the first few rows of the dataset

```
[4] # Display the first few rows of the dataset to understand its structure
    data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Data preprocessing

[5]
0s

```
# Data preprocessing - handle missing values, feature selection, encoding categorical variables, etc.
# For simplicity, you might want to drop some columns or perform imputation for missing values
# Replace 'selected_features' with the features you want to use for prediction
selected_features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked', 'Survived']
data = data[selected_features].dropna()
```

Data preprocessing (cont.)

0s

```
data.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Survived
0	3	male	22.0	1	0	7.2500	S	0
1	1	female	38.0	1	0	71.2833	C	1
2	3	female	26.0	0	0	7.9250	S	1
3	1	female	35.0	1	0	53.1000	S	1
4	3	male	35.0	0	0	8.0500	S	0

Convert categorical variables to numerical using one-hot encoding

```
[7]  # Convert categorical variables to numerical using one-hot encoding
     data = pd.get_dummies(data, columns=['Sex', 'Embarked'])
```

0s

```
[8]  data.head()
```

0s

	Pclass	Age	SibSp	Parch	Fare	Survived	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	3	22.0	1	0	7.2500	0	0	1	0	0	1
1	1	38.0	1	0	71.2833	1	1	0	1	0	0
2	3	26.0	0	0	7.9250	1	1	0	0	0	1
3	1	35.0	1	0	53.1000	1	1	0	0	0	1
4	3	35.0	0	0	8.0500	0	0	1	0	0	1

Split the data into features and target variable

```
[9]  # Split the data into features and target variable
     X = data.drop('Survived', axis=1)
     y = data['Survived']
```

Split the data into training and testing sets

```
[10] # 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)
```

Create and train the decision tree classifier

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

```
[11] # Create and train the decision tree classifier
     clf = DecisionTreeClassifier(random_state=42, max_depth=5)
     clf.fit(X_train, y_train)
```

```
▼ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, random_state=42)
```

Make predictions

✓ 0s

```
[12] # Make predictions
     predictions = clf.predict(X_test)
```

Evaluate the model

✓ 0s

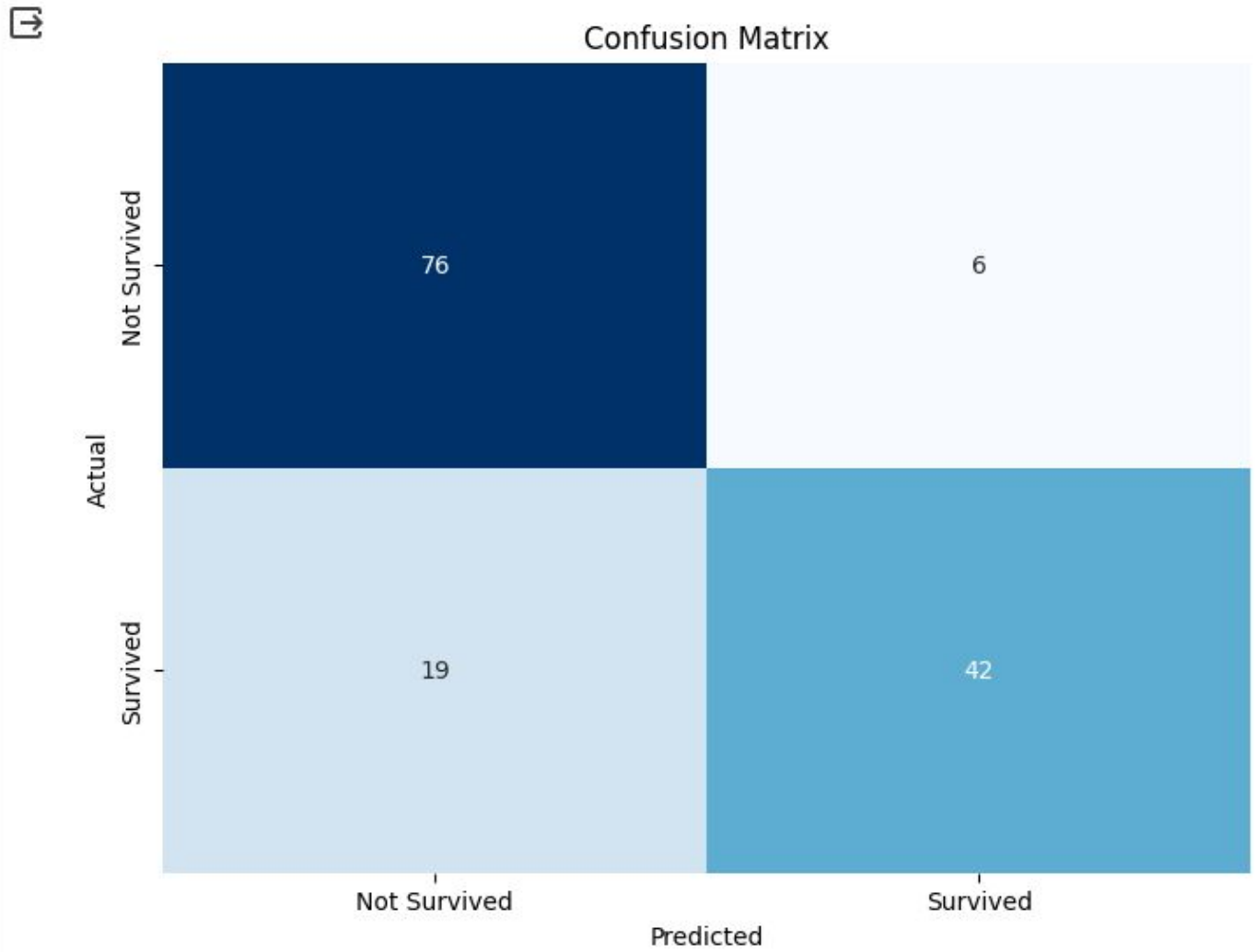
```
[13] # Evaluate the model
     conf_matrix = confusion_matrix(y_test, predictions)

     print("Confusion Matrix:")
     print(conf_matrix)
```

```
Confusion Matrix:
[[76  6]
 [19 42]]
```

Plot confusion matrix as a heatmap

0s

```
# Plot confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Not Survived', 'Survived'],
            yticklabels=['Not Survived', 'Survived'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
Not Survived
Survived
Actual
76
6
19
42
Not Survived
Survived
Predicted

Classification Report

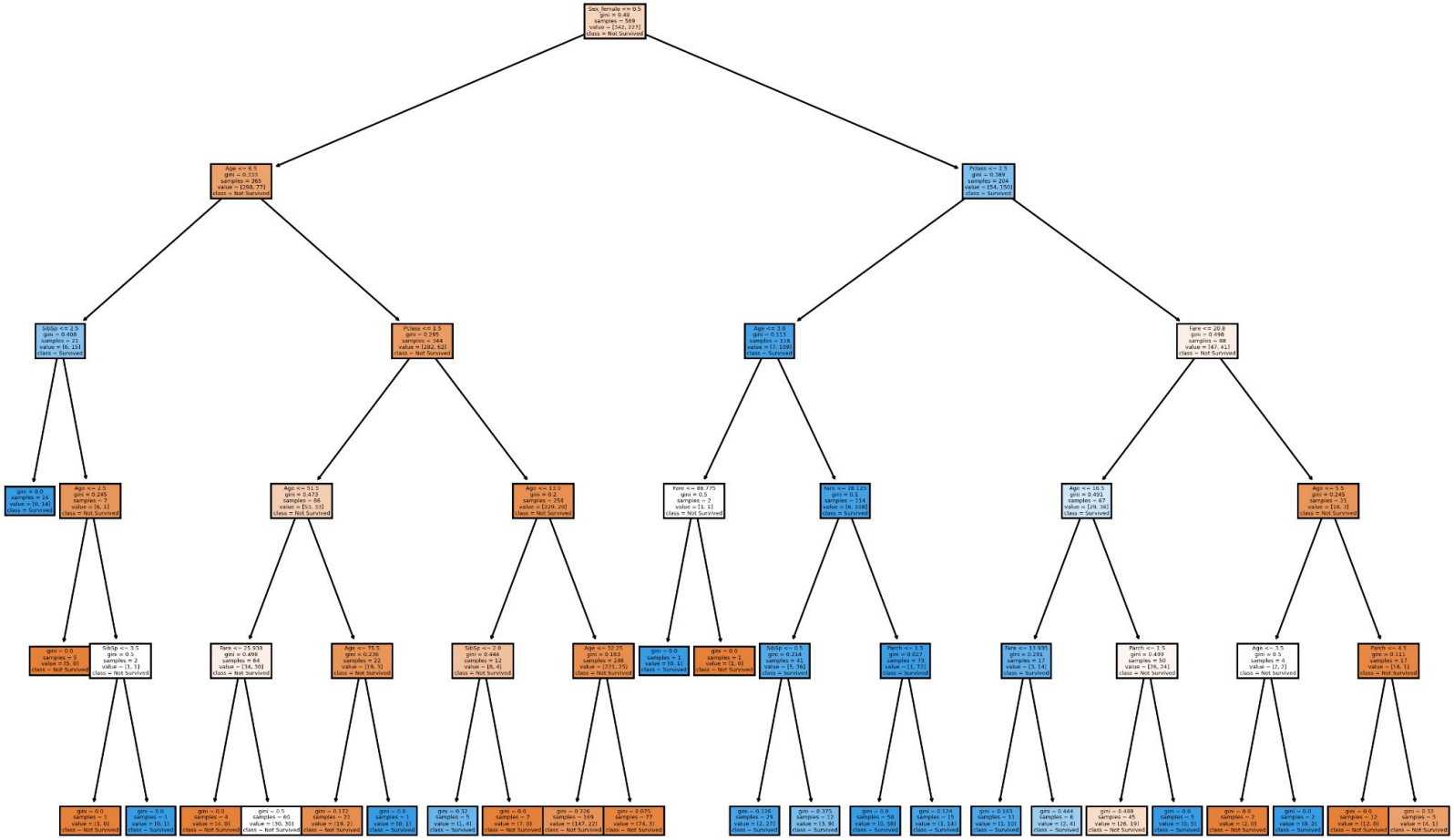
```
[15] print("\nClassification Report:")
     print(classification_report(y_test, predictions,digits=4))
```

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

           0     0.8000    0.9268    0.8588        82
           1     0.8750    0.6885    0.7706        61

    accuracy                         0.8252       143
   macro avg     0.8375    0.8077    0.8147       143
weighted avg     0.8320    0.8252    0.8212       143
```

Plot the Decision Tree

6s

```
# Set high resolution for tree plot
plt.figure(figsize=(15, 10), dpi=300)  # Adjust figsize and dpi for the desired resolution

# Plot the decision tree (optional)
tree.plot_tree(clf, feature_names=X.columns, class_names=['Not Survived', 'Survived'], filled=True)
plt.savefig('my_titanic_decision_tree_model.png', format='png', bbox_inches='tight')
plt.show()
```


Sex_female <= 0.5
gini = 0.469
samples = 569
value = [342, 227]
class = Not Survived
Age <= 6.5
gini = 0.333
samples = 265
value = [200, 77]
class = Not Survived
Pclass <= 2.5
gini = 0.389
samples = 204
value = [54, 150]
class = Survived
SibSp <= 2.5
gini = 0.408
samples = 21
value = [6, 15]
class = Survived
PClass <= 1.5
gini = 0.295
samples = 244
value = [202, 62]
class = Not Survived
Age <= 3.0
gini = 0.153
samples = 118
value = [7, 109]
class = Survived
Fare <= 20.8
gini = 0.498
samples = 88
value = [47, 41]
class = Not Survived
gini = 0.0
samples = 14
value = [0, 14]
class = Survived
Age <= 2.5
gini = 0.245
samples = 7
value = [6, 1]
class = Not Survived
Age <= 51.5
gini = 0.473
samples = 86
value = [53, 33]
class = Not Survived
Age <= 11.5
gini = 0.2
samples = 258
value = [229, 29]
class = Not Survived
Fare <= 88.775
gini = 0.5
samples = 2
value = [1, 1]
class = Not Survived
SibSp <= 26.125
gini = 0.1
samples = 114
value = [6, 108]
class = Survived
Age <= 16.5
gini = 0.491
samples = 67
value = [29, 38]
class = Survived
Age <= 5.5
gini = 0.245
samples = 21
value = [18, 3]
class = Not Survived
gini = 0.0
samples = 5
value = [5, 0]
class = Not Survived
SibSp <= 3.5
gini = 0.5
samples = 2
value = [1, 1]
class = Not Survived
Fare <= 25.938
gini = 0.496
samples = 64
value = [36, 50]
class = Not Survived
Age <= 75.5
gini = 0.236
samples = 22
value = [19, 3]
class = Not Survived
SibSp <= 2.0
gini = 0.444
samples = 12
value = [8, 4]
class = Not Survived
Age <= 32.25
gini = 0.163
samples = 246
value = [221, 25]
class = Not Survived
gini = 0.0
samples = 1
value = [0, 1]
class = Survived
gini = 0.0
samples = 1
value = [1, 0]
class = Not Survived
SibSp <= 0.5
gini = 0.214
samples = 41
value = [5, 36]
class = Survived
Parch <= 1.5
gini = 0.027
samples = 73
value = [1, 72]
class = Survived
Fare <= 13.935
gini = 0.291
samples = 17
value = [3, 14]
class = Survived
Parch <= 1.5
gini = 0.499
samples = 50
value = [26, 24]
class = Not Survived
Age <= 3.5
gini = 0.5
samples = 4
value = [2, 2]
class = Not Survived
Parch <= 4.5
gini = 0.111
samples = 17
value = [16, 1]
class = Not Survived
gini = 0.0
samples = 1
value = [1, 0]
class = Not Survived
gini = 0.0
samples = 1
value = [0, 1]
class = Survived
gini = 0.0
samples = 4
value = [4, 0]
class = Not Survived
gini = 0.32
samples = 5
value = [1, 4]
class = Survived
gini = 0.0
samples = 7
value = [7, 0]
class = Not Survived
gini = 0.226
samples = 169
value = [147, 22]
class = Not Survived
gini = 0.075
samples = 77
value = [74, 3]
class = Not Survived
gini = 0.126
samples = 29
value = [2, 27]
class = Survived
gini = 0.375
samples = 12
value = [3, 9]
class = Survived
gini = 0.0
samples = 58
value = [0, 58]
class = Survived
gini = 0.134
samples = 15
value = [1, 14]
class = Survived
gini = 0.165
samples = 11
value = [1, 10]
class = Survived
gini = 0.444
samples = 6
value = [2, 4]
class = Survived
gini = 0.488
samples = 45
value = [26, 19]
class = Not Survived
gini = 0.0
samples = 5
value = [0, 5]
class = Survived
gini = 0.0
samples = 7
value = [7, 0]
class = Not Survived
gini = 0.0
samples = 2
value = [0, 2]
class = Survived
gini = 0.0
samples = 12
value = [12, 0]
class = Not Survived
gini = 0.32
samples = 5
value = [4, 1]
class = Not Survived

Week 5: Assignment

Predicting Heart Attack Risk Using Decision Tree

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