|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Kingdom of Saudi Arabia**  **Ministry of Education**  **University of Jeddah**  **College of Computer Science and Engineering**  **Department of Computer Science and Artificial Intelligence** | Logo, company name  Description automatically generated | **المملكة العربية السعودية**  **وزارة التعليم**  **جامعة جدّة**  **كلية علوم وهندسة الحاسب**  **قسم علوم الحاسب والذكاء الاصطناعي** | |  |  |

|  |
| --- |
| **Lab 4** |
| **CCAI 312 Pattern Recognition** |
| **Third Trimester 2023**   |  |  | | --- | --- | | **Lab Date/Time:**  **Lab assignment submission Date/Time:** |  | | **Student Name: \_\_\_\_\_\_\_\_\_bushra dajam\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Student ID: \_\_\_\_\_\_\_\_\_\_\_2110054\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | |

|  |  |
| --- | --- |
| **Instructor Name** | **Section** |
|  |  |

**Instructions**:

The lab assignments must be submitted before the allocated Date/Time.

The lab assignments must by uploaded on LMS / sent by email to teacher@uj.edu.sa.

Plagiarism will be punished according to university rules.

|  |  |
| --- | --- |
| **PLO/CLO** | **SO** |
| **PLO S2 (CLO 2):** **Implement** a suitable pattern recognition technique for a given problem using Python | **SO 2:** Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline |
| **PLO C4 (CLO 3):** Discover and apply new knowledge as needed, using appropriate learning strategies. | **SO 7:** An ability to acquire and apply new knowledge as needed, using appropriate learning strategies. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Max Score** | **Student Score** |
| **PLO S2 / CLO 2 / SO 2** | **Task 1** | **2** |  |
| **PLO C4 / CLO 3 / SO 7** | **Task 2** | **2** |  |
| **Total** | |  |  |

# Lab Description

In this lab, we will learn about decision tree classifiers and walk through an example of how to create decision trees using Scikit-Learn. We'll see how the algorithm works with both numeric and non-numeric data. We will also learn how to evaluate the algorithm's accuracy and improve it using grid search and cross-validation.

# Objectives

* Have a working knowledge of decision trees models.
* Utilizing decision trees to develop and assess Apple/Pear detection and Titanic Survival prediction model.
* Learn how to perform hyper parameter tuning using Gridsearch.
* Learn how to convert categorical features to numeric features using onehotencoding.

# Lab Tool(s)

<https://www.kaggle.com/>

or

<https://colab.research.google.com/>

# Lab Deliverables

Submit A notebook to Blackboard containing your solution to the lab assessment at the end of this document.

# References:

<https://datagy.io/sklearn-decision-tree-classifier/>

<https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset>

Other Resources:

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

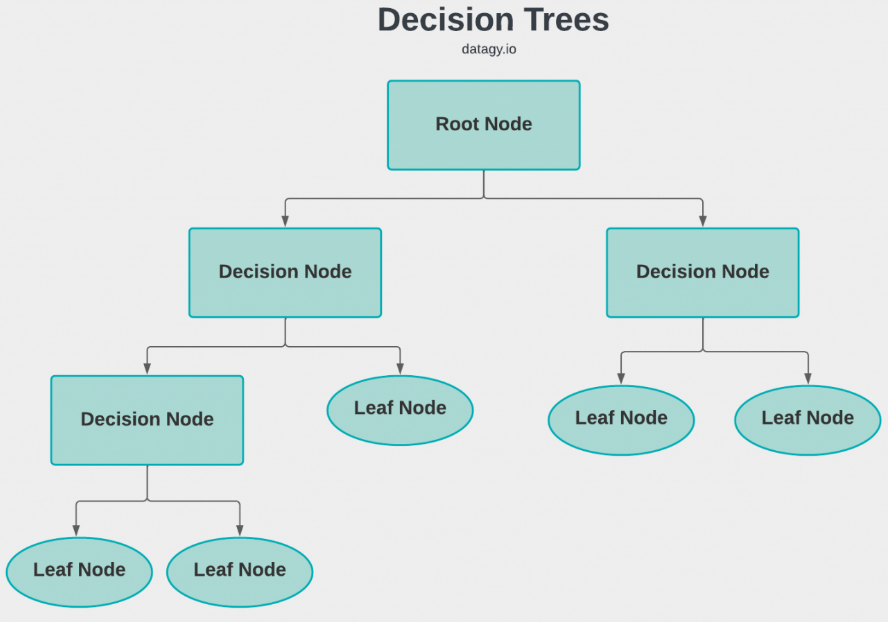
# Decision Tree

**1.1 Introducing Decision tree classifiers**

Decision tree classifiers are **supervised machine learning models**. This means that they use prelabelled data in order to train an algorithm that can be used to make a prediction. Decision trees can also be used for regression problems. Much of the information that you’ll learn in this lab can also be applied to regression problems.

Decision tree classifiers work like flowcharts. Each node of a decision tree represents a decision point that splits into two leaf nodes. Each of these nodes represents the outcome of the decision and each of the decisions can also turn into decision nodes. Eventually, the different decisions will lead to a final classification.

The diagram below demonstrates how decision trees work to make decisions. The top node is called the **root node**. Each of the decision points are called **decision nodes**. The final decision point is referred to as a **leaf node**.



It’s easy to see how this decision-making mirrors how we, as people, make decisions!

## **How Does a Decision Tree in Machine Learning Work?**

The process of training and predicting the target features using a decision tree in Machine Learning is given below:

* Feed a dataset, containing a number of training instances, with a set of features and a target
* Train the decision tree classification or regression models with the help of DecisionTreeClassifier () or DecisionTreeRegressor () methods, and add the required criterion while building the decision tree model
* Use Graphviz to visualize the decision tree model

The DecisionTreeClassifier() function looks like this:

**DecisionTreeClassifier (criterion = ‘gini’, random\_state = None, max\_depth = None, min\_samples\_leaf =1)**

Here are a few important parameters:

* **criterion:** It is used to measure the quality of a split in the decision tree classification. By default, it is ‘gini’; it also supports ‘entropy’.
* **max\_depth:**This is used to add maximum depth to the decision tree after the tree is expanded.
* **min\_samples\_leaf:** This parameter is used to add the minimum number of samples required to be present at a leaf node.

Part1

**Example 1:** Apple Pear Prediction

**Step 1**: Import Libraries

import pandas as pd  
from graphviz import Source  
from sklearn.tree import DecisionTreeClassifier, export\_graphviz

**Step 2**: Read and explore the ApplePear dataset

apears = pd.read\_csv('/content/ApplesPears.csv')  
apears.head()

**Step 3**: Split data to X(features) and y (target\_label)

y = apears.pop('Class').values  
apears.pop('Taste')    # Scikit-learn can't deal with category features  
ap\_features = apears.columns  
X = apears.values  
X[0]

**Step 4**: Check y and X values

y

X

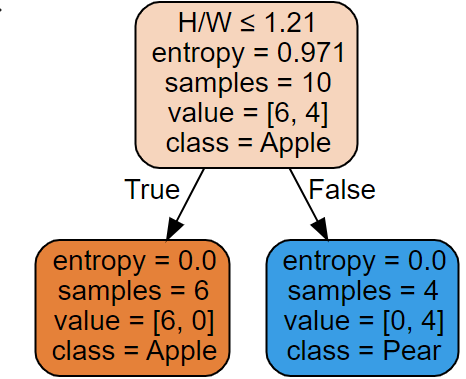
**Step 5**: Train DecisionTreeClassifier

tree = DecisionTreeClassifier(criterion='entropy')  
ap\_tree = tree.fit(X, y)

print(type(ap\_tree))  
print(type(tree))  
print(ap\_tree == tree)

**Step 6**: Draw the modeltree using graphviz

# Use BOTH 'conda install graphviz' and 'pip install graphviz'  
tree\_ap = export\_graphviz(ap\_tree, out\_file=None,   
                      feature\_names=ap\_features,  
                      class\_names=['Apple','Pear'],    
                      filled=True, rounded=True,    
                      special\_characters=True)    
graph = Source(tree\_ap)    
graph



**Step 7**: Delete H/W feature to make it harder and train a new model

apears.pop('H/W')    # Delete this feature to make it harder  
X = apears.values  
ap\_features = apears.columns  
ap2\_tree = tree.fit(X, y)

**Step 7**: Draw the new model

tree\_ap = export\_graphviz(ap2\_tree, out\_file=None,   
                      feature\_names=ap\_features,  
                      class\_names=['Apple','Pear'],    
                      filled=True, rounded=True,    
                      special\_characters=True)    
graph = Source(tree\_ap)    
graph

A picture containing text, sign

Description automatically generated

**Example 2 :** [Titanic Survival Prediction](https://www.analyticsvidhya.com/blog/2021/07/titanic-survival-prediction-using-machine-learning/)

**Dataset**

In order to build our decision tree classifier, we’ll be using the Titanic dataset.

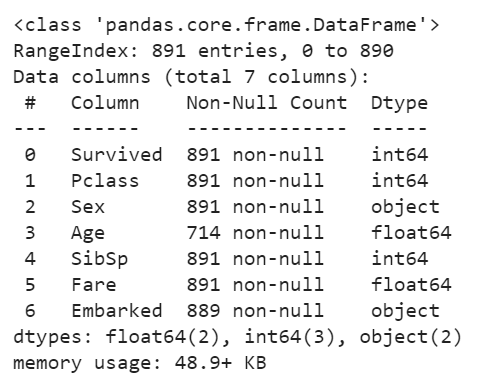
**Step 1**: Downloading an exploring the Titanic dataset

import pandas as pd data = pd.read\_csv( 'https://github.com/datagy/data/raw/main/titanic.csv', usecols=['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Fare', 'Embarked'])

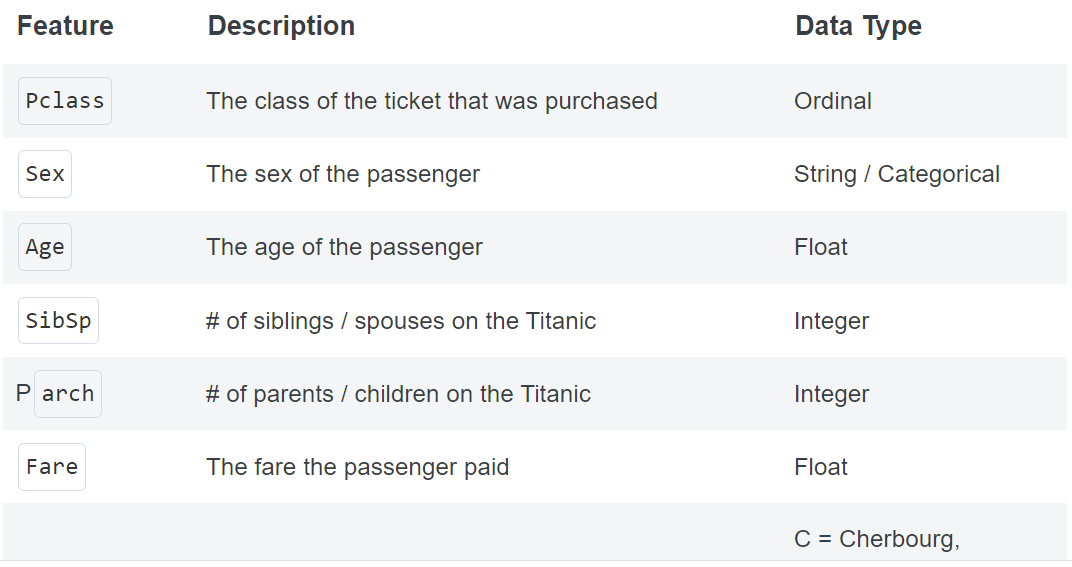
data.shape



data.info()



We have a number of features available to us, some of which are numeric and some of which are categorical. Let’s take a closer look at these features:

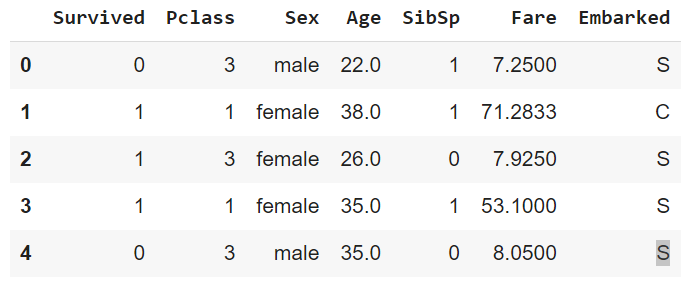


data.describe()

Table

Description automatically generated

data.head()



data['Survived'].value\_counts()

A picture containing logo

Description automatically generated

**Step 2**: drop any missing records

data = data.dropna()

data.shape



**Step 3**: Let’s better understand the distribution of the data by plotting a pairplot using Seaborn.

import seaborn as sns   
import matplotlib.pyplot as plt

sns.pairplot(data=data, hue='Survived')  
plt.show()

**Diagram

Description automatically generated**

Based on the image above, we can see that there are a number of clear separations in the data. This can be quite helpful in splitting our data into pure splits using a decision tree classifier.

**Step 4**: split the data into two variables

* X :our features matrix (because it’s a matrix, it’s denoted with a capital letter)
* y: our target variable

X = X.copy()

y = X.pop('Survived')

### **Build DecisionTree Classifier using only numeric features.**

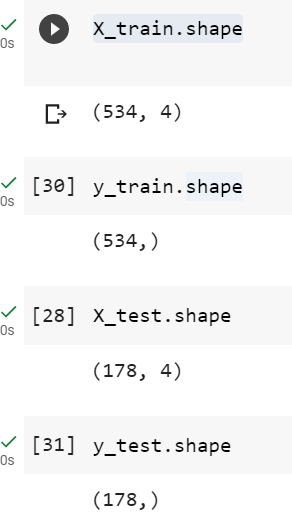
**Step 5**: drop all the non-numeric variables for now. Machine learnings tend to require numerical columns to work. We’ll focus on these later, but for now we’ll keep things simple.

X\_numeric=X.select\_dtypes(['number'])   
  
  
X\_numeric.shape



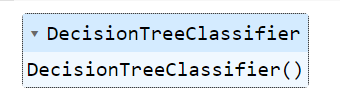
**Step 6:** Split data into training and testing data and check its shapes.

from sklearn.model\_selection import train\_test\_split  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state = 100)



**Step 7**: Create Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier  
clf = DecisionTreeClassifier()  
clf.fit(X\_train, y\_train)



**Step 8**: Use the model to get test set predictions

predictions = clf.predict(X\_test)  
print(predictions[:5])



**Step 9**:Measure the accuracy of the model

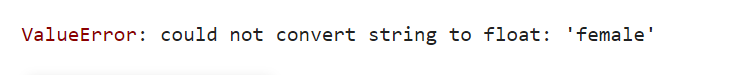
from sklearn.metrics import accuracy\_score  
print(accuracy\_score(y\_test, predictions))



### **Build DecisionTree Classifier using all features (numeric & categorical)**

### **Step 10**: Try to build DecisionTree Classifier using all features (numeric & categorical)

clf = DecisionTreeClassifier()  
clf.fit(X\_train, y\_train)



The DecisionTree Classifier does not support categorical features. We must use onehotencoding to convert categorical features to numbers.

## **Using OneHotEncoding**

OneHotEncoder class has two key methods:

1. fit to 'learn' the transform from the data,
2. transform to apply the OneHot transform to the data, the transform can be applied to other (e.g. test) datasets.

**Step 11**: Convert categorical features to numeric using OneHotEncoder

# Using make\_column\_transformer to One-Hot Encode  
from sklearn.preprocessing import OneHotEncoder  
from sklearn.compose import make\_column\_transformer  
  
column\_transformer = make\_column\_transformer((OneHotEncoder(), ['Sex', 'Embarked']),remainder='passthrough')  
  
# First, we must transform the entire dataset  
X\_transformed = column\_transformer.fit\_transform(X)  
X\_transformed = pd.DataFrame(data=X\_transformed)  
  
#second,we divide the transformed dataset into train/test dataset  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_transformed, y, random\_state = 100)

**Step 12:** Train and evaluate model with One-Hot Encoded Values

# Making Predictions with One-Hot Encoded Values  
clf = DecisionTreeClassifier()  
clf.fit(X\_train, y\_train)  
  
predictions = clf.predict(X\_test)  
print(accuracy\_score(y\_test, predictions))



By adding categorical features, we were able to increase our accuracy to 77%!

## **Hyperparameter Tuning for Decision Tree Classifiers in Sklearn**

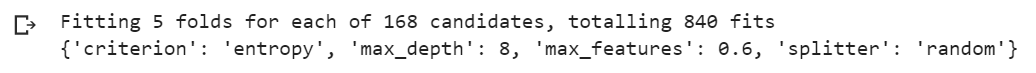
[Hyper-parameter tuning](https://datagy.io/sklearn-gridsearchcv/)**,** then, refers to the process of tuning these values to ensure a higher accuracy score**.**One way to do this is, simply, to plug in different values and see which hyper-parameters return the highest score.

In order to do this, we first need to decide which hyperparameters to test. Let’s see which ones we will be using:

* criterion – the function that’s used to determine the quality of a split
* max\_depth – the maximum depth of the tree
* max\_features – the max number of features to consider when making a split
* splitter – the strategy used to choose the split at each node

**Step 13**:

# Creating a dictionary of parameters to use in GridSearchCV  
from sklearn.model\_selection import GridSearchCV  
  
params = {  
    'criterion':  ['gini', 'entropy'],  
    'max\_depth':  [None, 2, 4, 6, 8, 10],  
    'max\_features': [None, 'sqrt', 'log2', 0.2, 0.4, 0.6, 0.8],  
    'splitter': ['best', 'random']  
}  
  
clf = GridSearchCV(  
    estimator=DecisionTreeClassifier(),  
    param\_grid=params,  
    cv=5,  
    n\_jobs=5,  
    verbose=1,  
)  
clf.fit(X\_train, y\_train)  
print(clf.best\_params\_)



**Step 14**: Train and evaluate model with best hyperparameters

clf = DecisionTreeClassifier(max\_depth=4, criterion='entropy', max\_features=0.6, splitter='best')  
clf.fit(X\_train, y\_train)  
predictions = clf.predict(X\_test)  
  
print(accuracy\_score(y\_test, predictions))

****

By using best hyperparameters values , we were able to increase our accuracy to 80%!

**Step 15**: Draw the tree using graphviz

from sklearn.tree import DecisionTreeClassifier, export\_graphviz  
from graphviz import Source  
  
tree\_im = export\_graphviz(clf, out\_file=None,    
        feature\_names=column\_transformer.get\_feature\_names\_out(),  
        class\_names=y.name,    
        filled=True, rounded=True,    
        special\_characters=True)

graph = Source(tree\_im)    
graph

Part2

**Hotel Reservations cancellation prediction**

## **Dataset**

The online hotel reservation channels have dramatically changed booking possibilities and customers’ behavior. A significant number of hotel reservations are called-off due to cancellations or no-shows. The typical reasons for cancellations include change of plans, scheduling conflicts, etc. This is often made easier by the option to do so free of charge or preferably at a low cost which is beneficial to hotel guests but it is a less desirable and possibly revenue-diminishing factor for hotels to deal with.

**Can you predict if the customer is going to honor the reservation or cancel it?**

Task **Task 1: [PLO S2 / CLO 2 / SO 2] [2 marks]**

1. First import required libraries, load “**Hotel Reservations**” dataset into a data frame and use Booking\_ID as index\_col.
2. Explore dataset and answer the following questions:

What is the dataset's shape? How many samples are there?How many features are there?

How many categorical features are there in the dataset? What is the distribution of the target label "booking\_status"?

1. Split dataframe to two variables X (all features except booking\_status) and y (booking\_status).
2. Split the dataset to train/test split.
3. Fit and evaluate DecisionTree Classifier using train/test split (use random\_state=100). What is the accuracy of the model?
4. Create a new variable X\_numric that contain only numeric features.
5. Repeat steps 5 & 6 using X\_numeric  instead of X.

How accurate is the model? Is the accuracy good or bad?

1. Use GridSearchCV to find the best values for hyperparameter (see step 13)
2. What is the best values for 'criterion' , 'max\_depth' , 'max\_features', and 'splitter'.
3. Fit and evaluate new DecisionTree Classifier using best hyperparameter values.
4. How accurate is the new model? Does the model improve?

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Task? 2: [PLO C4 / CLO 3 / SO 7] [2 mark]**

1. Use OneHotEncoder to convert categorical features to numbers (see step 11)
2. Fit and evaluate DecisionTree Classifier using all features.

How accurate is the model? Does the model improve by using categorical features?

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer

Description automatically generated