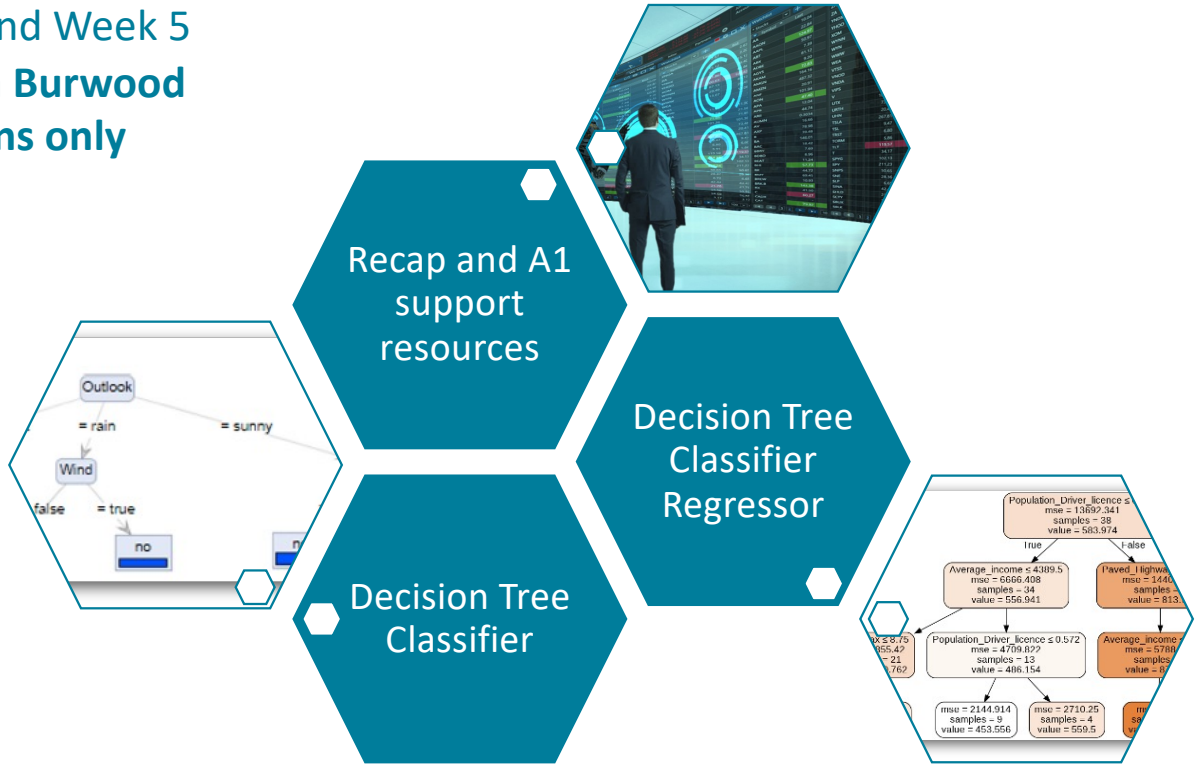
MIS710 Machine Learning in Business

Topic 5- Decision Trees

Associate Professor Lemai Nguyen

Predictive Machine Learning with Decision Trees

- A1 new due date: 12th August 8.00pm AEST, 3:30pm IST
- Consultation sessions in Week 4 and Week 5
  - **Two f2f and zoom sessions in Burwood**
  - **Two after-hours zoom sessions only**



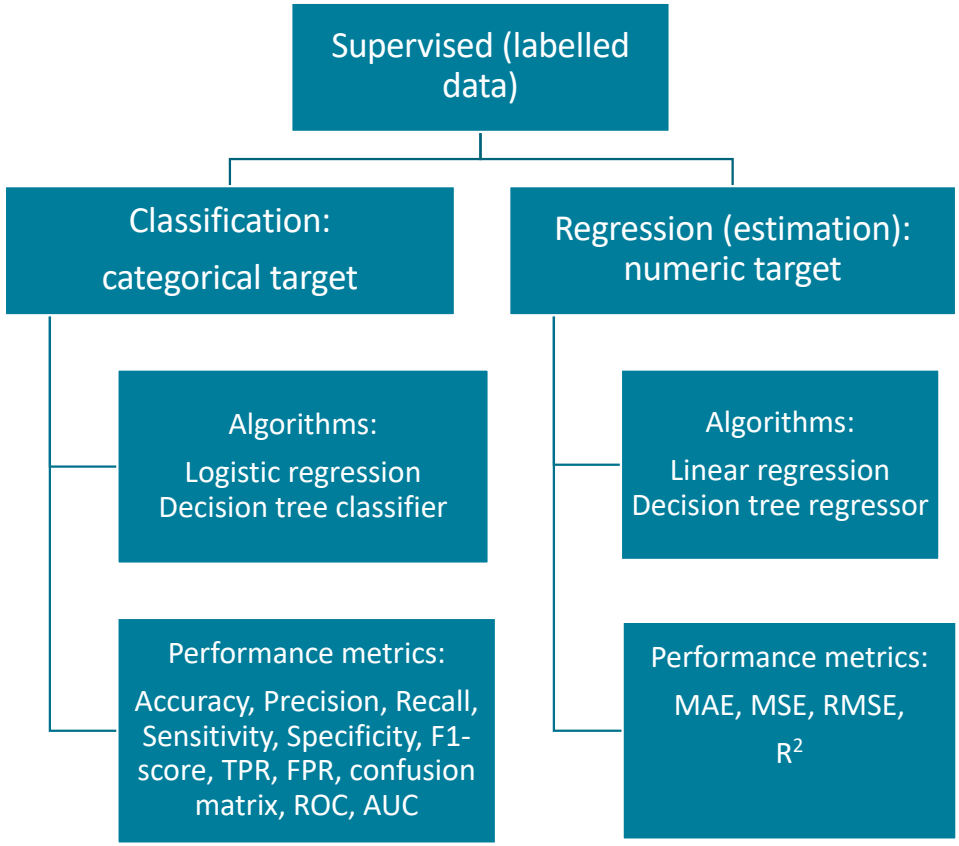
Week 5: updated consultation sessions

- **Monday 5th August 2.00-3.00 pm AEST**, room **HE1.009, ZOOM** Durgesh Samariya
- **Tuesday 6th August 5.00-6.00 pm AEST, 12.30-1.30 pm IST**, **ZOOM**: Abhishek Jha
- **Wednesday 7th August 7.30-8.30 pm AEST, 3.00-4.00 pm IST, ZOOM** : Emran Ali.
- **Friday 9th August 3.00-4.00 pm AEST, 10.30-11.30 am IST**, **ZOOM**, room TBA: Dat Le
- MLCafe forum, AICafe forum, and A1 forum: Lemai Nguyen

Assignment 1 Part 3
Competition - Build your portfolio!

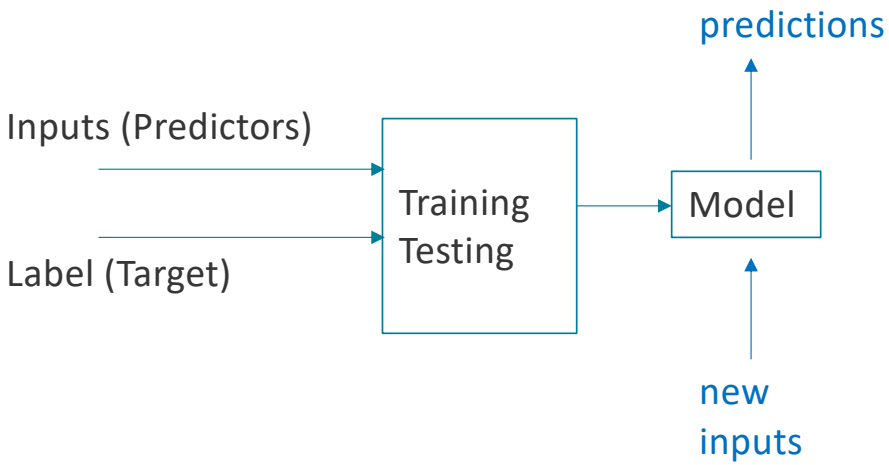
Optional Part 3: If you participate in the A1 Challenge then submit a csv deployment file with your predicted labels.

Recap: Supervised machine learning

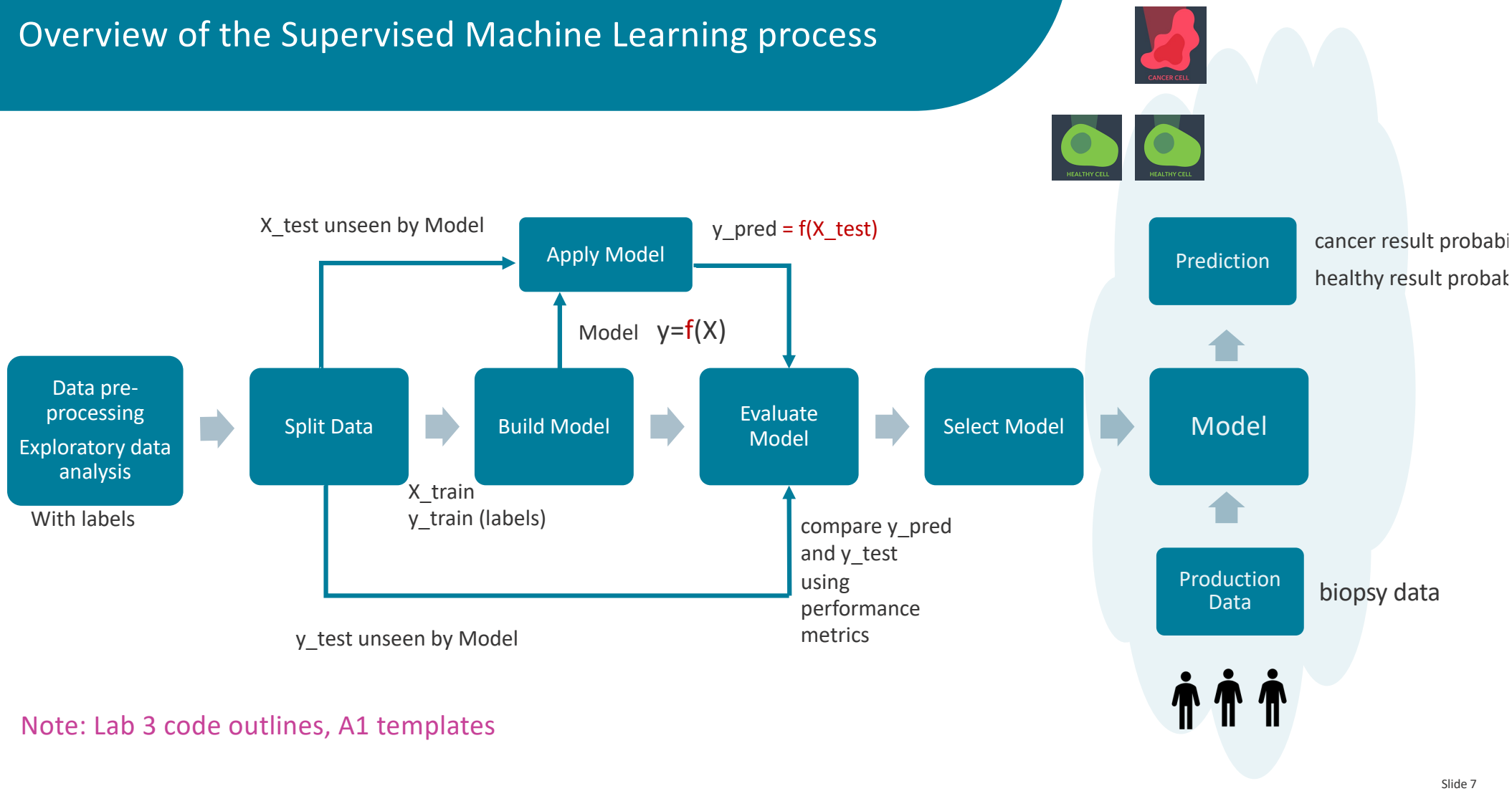


Recap: Supervised machine learning

- infer a representation model i.e., relationship between inputs and an output, based on labelled training data
- use this model to map new unlabelled data to predict target variables



Overview of the Supervised Machine Learning process



Note: Lab 3 code outlines, A1 templates

Example:
Application of BACCM

Check the rubric for different standards

1. Business Understanding

Business Problem Identification

VayuAir aims to enhance customer satisfaction by identifying key satisfaction drivers in a competitive airline industry. Yet, without clear insights into what affects satisfaction, VayuAir risks losing customers and damaging its brand. Addressing this challenge is critical; hence, VayuAir is adopting a data-driven strategy to not just meet, but foresee and fulfill customer needs, fostering loyalty and advocacy.

The BACCM for VayuAir's Enhancement of Customer Satisfaction:

Need: VayuAir, a rapidly growing airline, faces the critical challenge of deciphering and **enhancing** the variables influencing **customer satisfaction**.

Solution: The solution to this business need is the deployment of a bespoke machine learning

integrating feedback loops into service training programs and digital platforms.

Value: The analysis promises significant value by pinpointing **and profitability** through refined service delivery.

Stakeholders: Real-life stakeholders in this study include VayuAir's **managerial staff**, who

decisions; **frontline employees** whose day-to-

and, fundamentally, the **passengers**, whose

Change: We recommend a series of calculated changes based on the data insights, ranging from immediate **enhancements in in-flight service options** to **long-term**

continuous feedback and iteration.

Context: The strategy takes place against a dynamic backdrop of an evolving industry landscape, with

A1 resources:

- A1 resources
  - Report structure
  - Code template
  - Feedbackfruits

- Lectures 1-5 and recordings
  - Meeting with Preetham D Bangera in Week 4
- Labs 1-5 and online seminar recordings
- Labs 1-5 solutions

- Python resources
- 4 Consultations in Weeks 1, 2, and 3
- 7 A1 consultation sessions (online and Burwood) in Weeks 4 and 5
- AICafe, MLCafe, and A1 forum

Topic 4 (cont.): Classification Model evaluation

FP – false alarms

FN – missed detection

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} = 1 - Error$$

$$Error = \frac{FP+FN}{TP+FP+TN+FN} = 1 - Accuracy$$

True labels

	Negative	Positive
False labels	**TN**	FP (type 1)
True labels	FN (type 2)	**TP**

Predictions

Table 8.2 Evaluation Measures

Term	Definition	Calculation
Sensitivity	Ability to select what needs to be selected	TP/(TP+FN)
Specificity	Ability to reject what needs to be rejected	TN/(TN+FP)
Precision	Proportion of cases found that were relevant	TP/(TP+FP)
Recall	Proportion of all relevant cases that were found	TP/(TP+FN)
Accuracy	Aggregate measure of classifier performance	(TP+TN)/(TP+TN+FP+FN)

Kotu and Deshpande, 2019, chapter 4

Model evaluation

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

True labels

	Negative	Positive
False labels	**TN**	FP
True labels	FN	**TP**

Predictions

$$Precision(PRE) = \frac{TP}{TP + FP}$$

Useful in spam detection

True labels

	Negative	Positive
False labels	**TN**	FP
True labels	FN	**TP**

Predictions

$$Specificity = \frac{TN}{TN+FP}$$

True labels

	Negative	Positive
False labels	**TN**	FP
True labels	FN	**TP**

Predictions

Model evaluation

Sensitivity – Recall (REC) – True Positive Rate (TPR)

$$TPR = REC = \frac{TP}{TP+FN} = \frac{TP}{P}$$

FN=0 then TPR=1

True labels

	Negative	Positive
False labels	**TN**	FP
True labels	FN	**TP**

Predictions

False Positive Rate (FPR)

$$FPR = \frac{FP}{FP+TN} = \frac{FP}{N} = 1 - Specifity$$

TN=0 then FPR=1

True labels

	Negative	Positive
False labels	**TN**	FP
True labels	FN	**TP**

Predictions

Useful in imbalanced class problems such as cancer diagnosis – recall
Consider the domain problems

Accuracy Paradox and F1 score

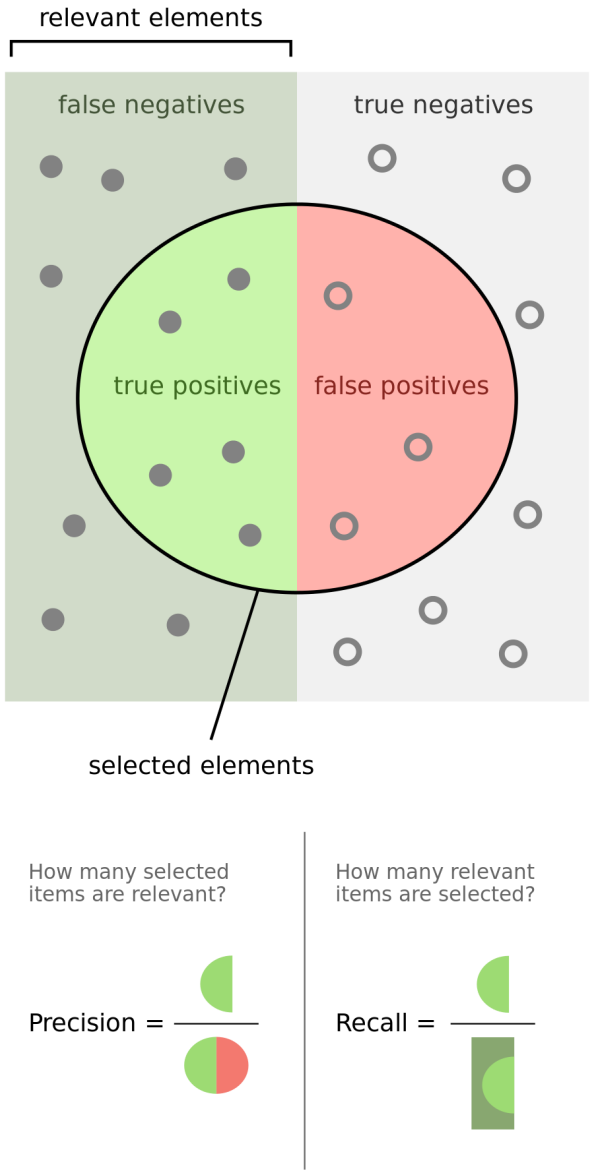
Problem: class imbalance but equal importance to precision and recall.

F1 score: harmonic mean of precision and recall:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Aim to get high F1, especially when FN and FP are important.

A good way to summarise the classification performance.

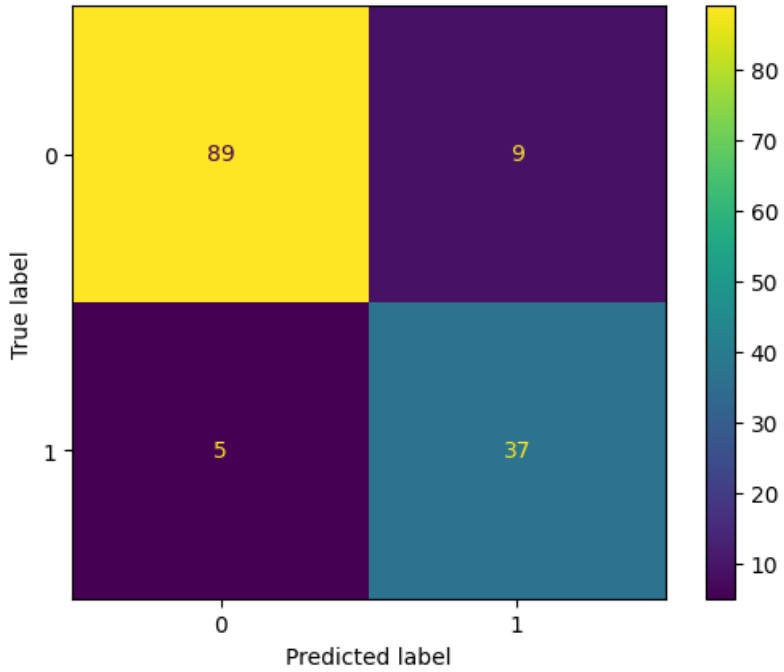


Model performance evaluation in scikit-learn

True labels	Negative	Positive
False labels	TN	FP
True labels	FN	TP

```
#print confusion matrix and evaluation report
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[89  9]
 [ 5 37]]
              precision    recall  f1-score   support

           0       0.95      0.91      0.93        98
           1       0.80      0.88      0.84        42

    accuracy                           0.90       140
   macro avg       0.88      0.89      0.88       140
weighted avg       0.90      0.90      0.90       140
```



What is the best threshold?

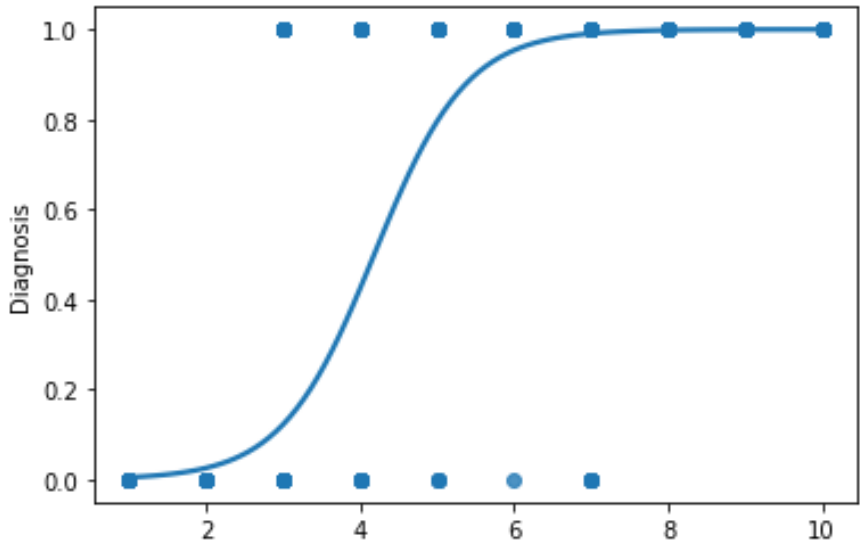
Recall that we convert

$$b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$$

into $p \in \{0,1\}$

But we need to 'classify' p into classes, and assume:

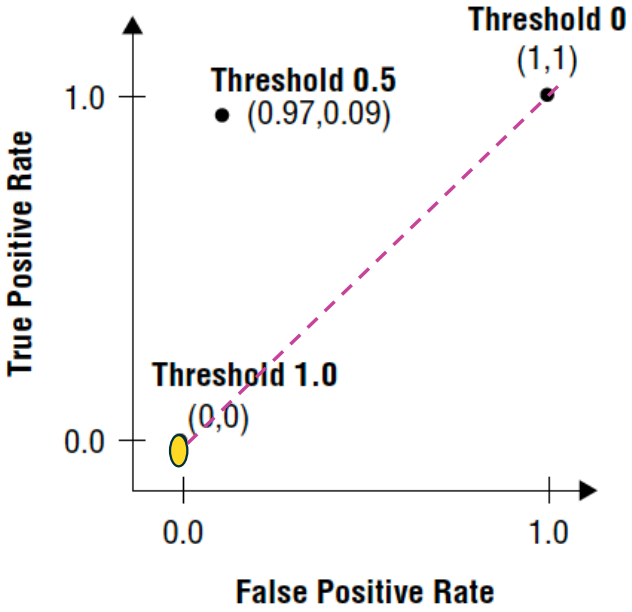
$\hat{y}$ { **1 if p> 0.5** (**is this the best threshold?**)
0 otherwise

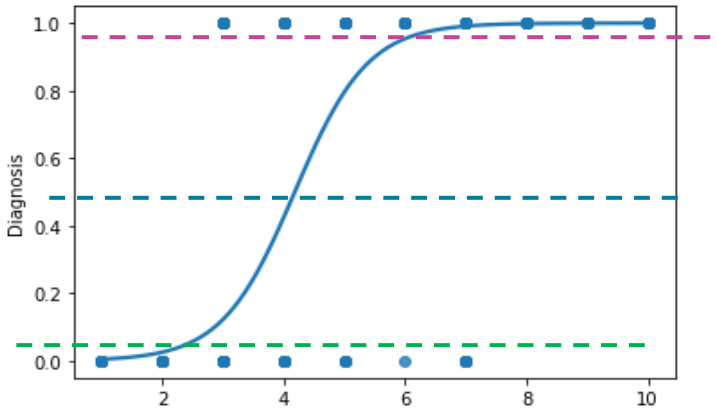


Receiver Operating Characteristic (ROC) curve

Plotting the TPR against the FPR at various threshold settings

0 = healthy
1 = cancerous





Adapted from Lee, 2019

Threshold = 1, all predictions: 0

True labels	Negative	Positive
False labels	**TN**	FP=0
True labels	FN	**TP=0**

Predictions

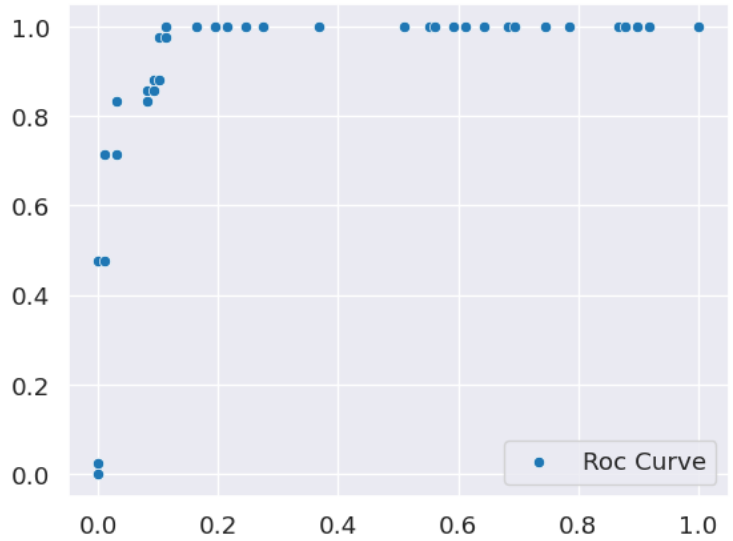
Threshold = 0, all predictions: 1

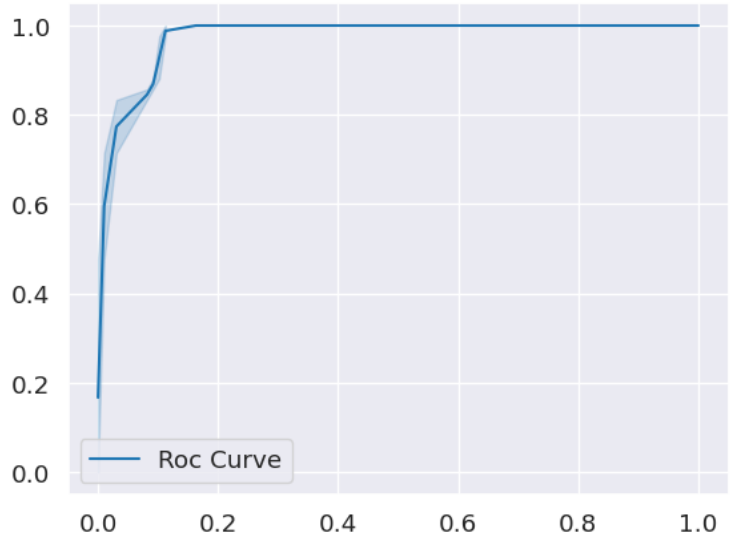
True labels	Negative	Positive
False labels	**TN = 0**	FP
True labels	FN = 0	**TP**

Predictions

Plot ROC curve

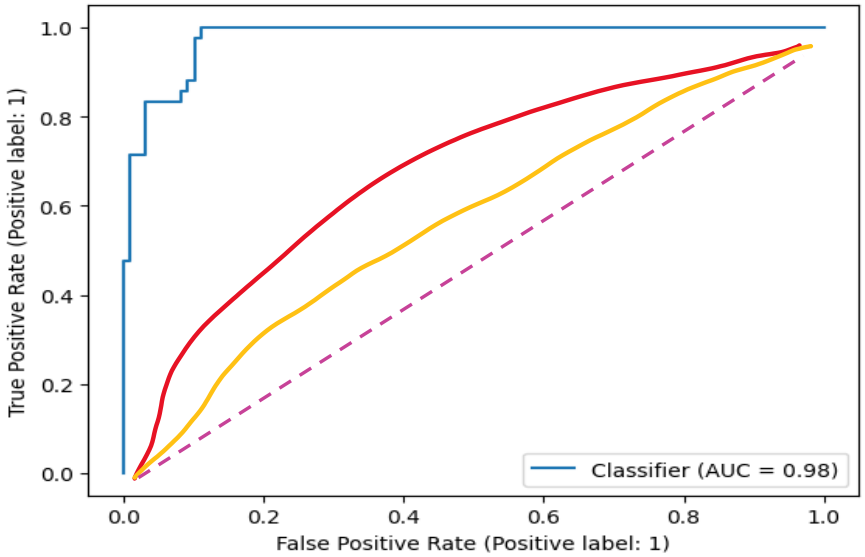
```
#get predicted probabilities for the primary label
y_pred_probs = logreg.predict_proba(X_test)
y_pred_probs = y_pred_probs [:, 1]


#get fpr and tpr and plot the ROC curve
from sklearn.metrics import roc_curve, roc_auc_score

lr_fpr, lr_tpr, thresholds = roc_curve(y_test, y_pred_probs)

sns.scatterplot(x=lr_fpr, y=lr_tpr, label='Roc Curve')


sns.lineplot(x=lr_fpr, y=lr_tpr, label='Roc Curve')
```





Receiver Operating Characteristic (ROC) curve

Plotting the TPR against the FPR at various threshold settings.

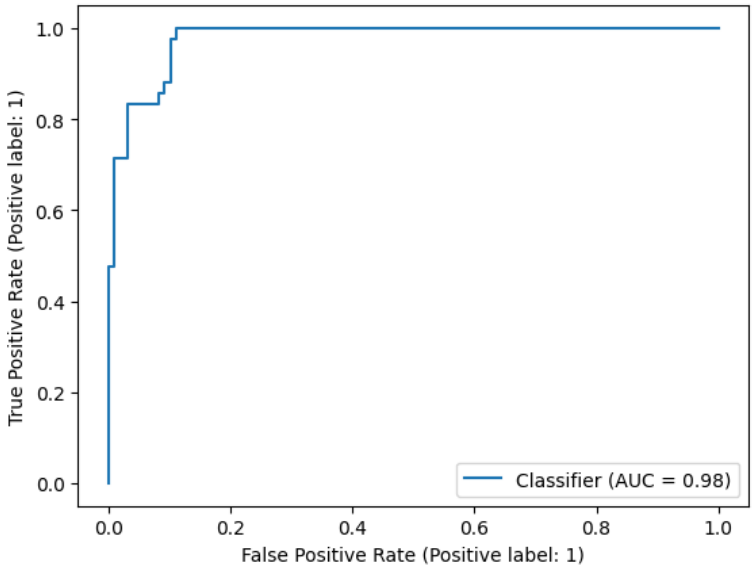


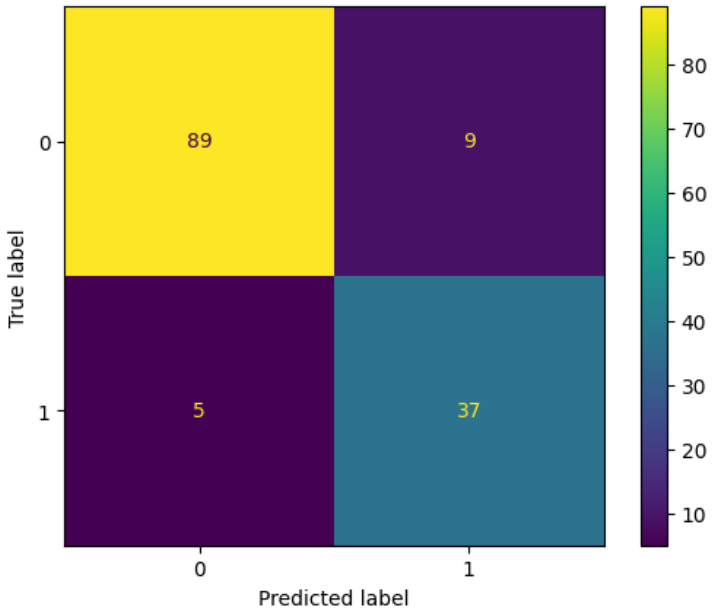
The area under the curve (AUC) is often considered as a summary of the model performance.

Plot ROC curve and visualise Confusion matrix using built in functions

NOTE: Multiple-class: One vs Rest (OvR) or One vs All (OvA)

```
#import classes to display RocCurve and Confusion Matrix
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import ConfusionMatrixDisplay

RocCurveDisplay.from_predictions(y_test, y_pred_probs)
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.show()
```





What is the best the threshold?
based on accuracy

```
# Calculate ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)

# initialize variables to store the best threshold and the highest accuracy
score
best_threshold = None
highest_ac_score = 0

# iterate over the thresholds and compute the accuracy score for each
for threshold in thresholds:
    y_pred_test = (y_pred_probs >= threshold).astype(int)
    ac_score = metrics.accuracy_score(y_test, y_pred_test)
if ac_score > highest_ac_score:
    highest_ac_score = ac_score
    best_threshold = threshold

# Assign predictions based on the new threshold
y_pred = (y_pred_probs >= best_threshold).astype(int)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

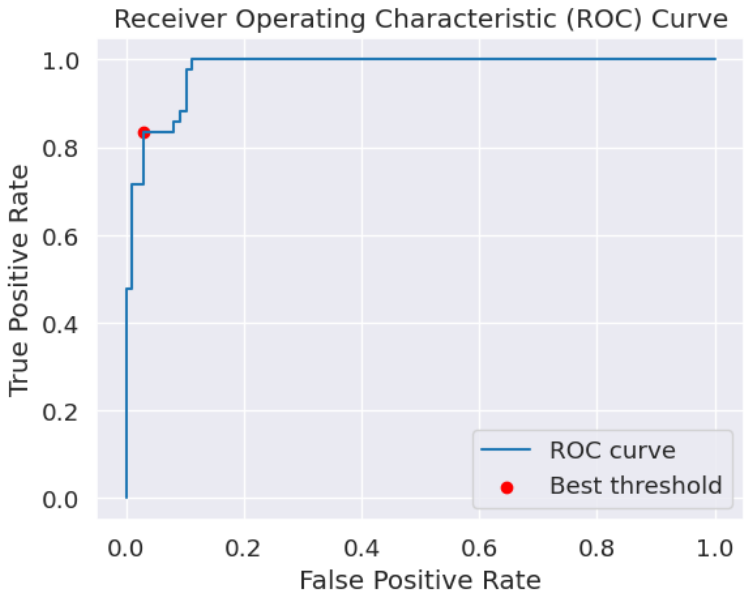
```
[[95  3]
 [ 7 35]]
              precision    recall  f1-score   support

           0       0.93      0.97      0.95        98
           1       0.92      0.83      0.88        42

    accuracy                           0.93       140
   macro avg       0.93      0.90      0.91       140
weighted avg       0.93      0.93      0.93       140
```

```
# print the best threshold and the highest AUC score on the test data
print('Best threshold:', best_threshold)
print('Highest accuracy score:', highest_ac_score)
```

```
Best threshold:
0.9143633687597816
Highest accuracy score:
0.9285714285714286
```

```
# Find the index corresponding to the specific threshold
best_index = (np.abs(thresholds - best_threshold)).argmin()
```

```
# plot the ROC curve and the best point
plt.plot(fpr, tpr, label='ROC curve')
plt.scatter(fpr[best_index], tpr[best_index], marker='o',
color='red', label='Best threshold')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```



Logistic regression

Pros

- Logistic regression works best with binary (dichotomous) targets.
- Explainable: Easy to interpret.
- Less effort for data preparation, but normalization can be beneficial.
- Works for both numerical and categorical predictors.
- Visual representation possible for one predictor.

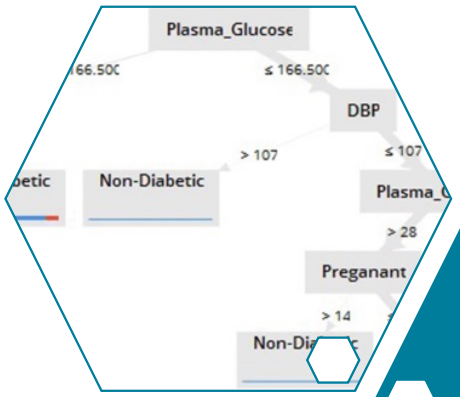
Cons

- More complex with multi-class targets, use one-vs-rest (OvR)
- Assumes a linear relationship between predictors and the logit.
- Requires sufficient cases per predictor to avoid overfitting.
- Visual representation is challenging with multiple predictors but possible using specific techniques.

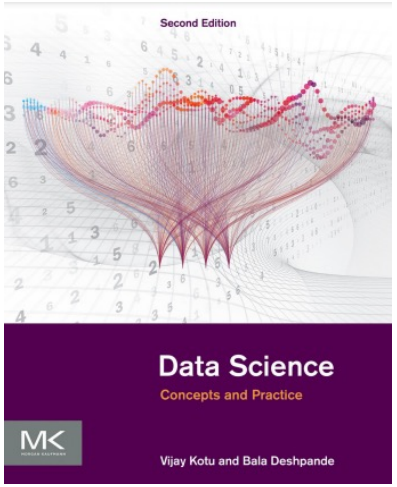
Assumptions

- The target should be categorical.
- The datapoints are independent.
- No extreme outliers (though logistic regression is somewhat robust, it's best to address them).
- No severe collinearity among the predictors.
- There exists a linear relationship between each predictor and the logit of the target.
- Sample size is large enough, with sufficient data points for each predictor case.

https://www.statology.org/assumptions-of-logistic-regression/



Decision Tree Classifier



Chapter 4

Intuition

Table 4.1 The Classic Golf Data Set

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	78	FALSE	yes
rain	70	96	FALSE	yes
rain	68	80	FALSE	yes
rain	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rain	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rain	71	80	TRUE	no

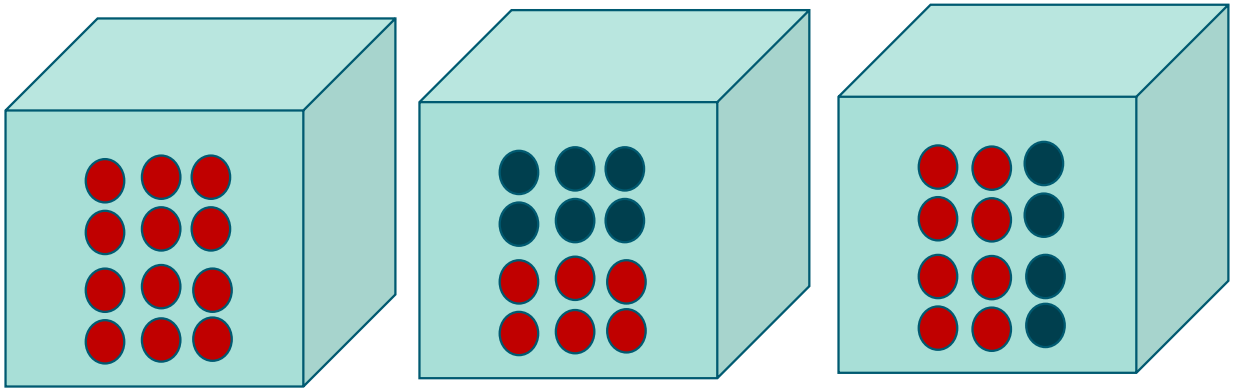
Source of Figures: Kotu and Deshpande, 2019, chapter 4

Let’s make some observations

Entropy

From intuition to conceptualisation

How pure are these boxes?

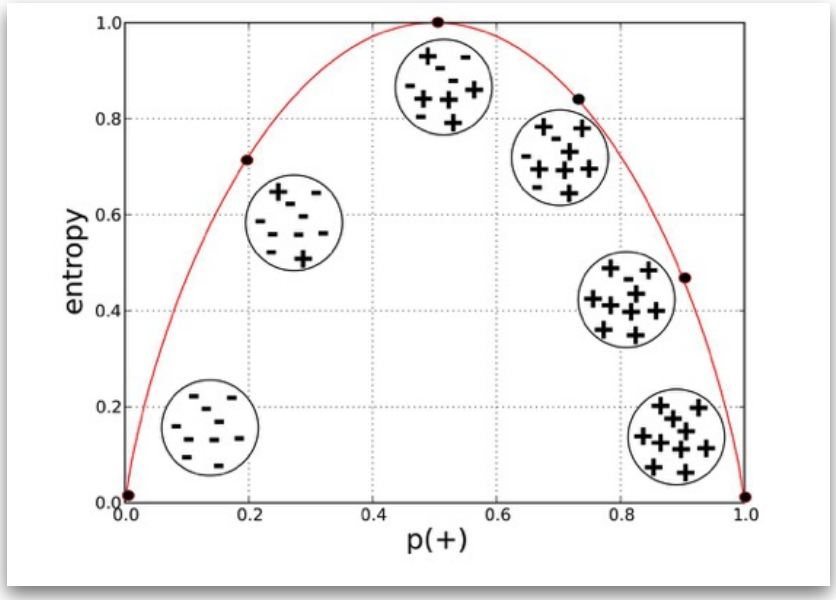


100% purity

50% purity

66.7% purity

Provost, Foster; Fawcett, Tom. Data Science for Business, Ch.. 3



Entropy

a measure of the uncertainty (how similar/different data are)

$$H = -\sum p_k \log_2 (p_k)$$

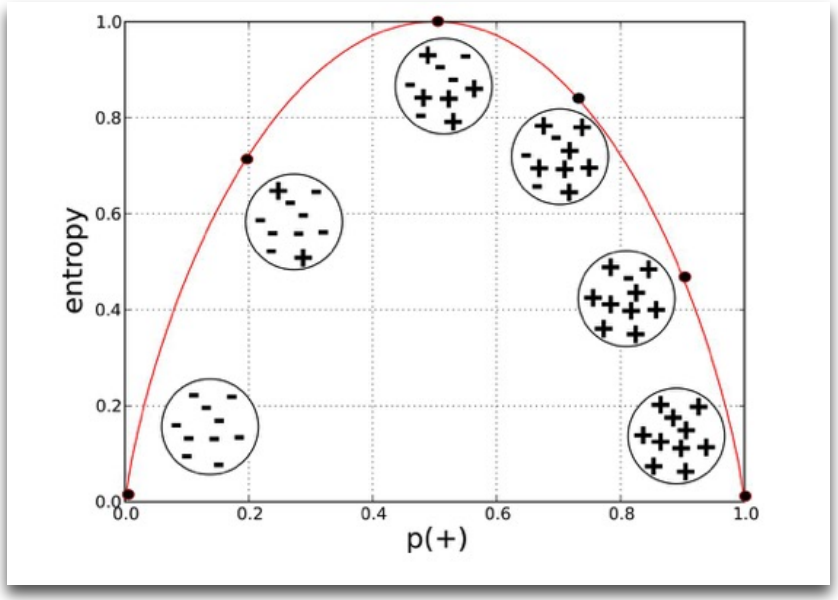
Shannon, 1948

p – probability that an event occurs
k: number of classes
H – (Greek capital eta)

Data equality divided: p = 50%
$H = -[(0.5 \log_2 0.5) + (0.5 \log_2 0.5)] = -\log_2 0.5 = -(-1) = 1$

Completely homogeneous data: p=100%
$H = -1 \log_2 1 = 0$

Provost, Foster; Fawcett, Tom. Data Science for Business, Ch.. 3



\+ –

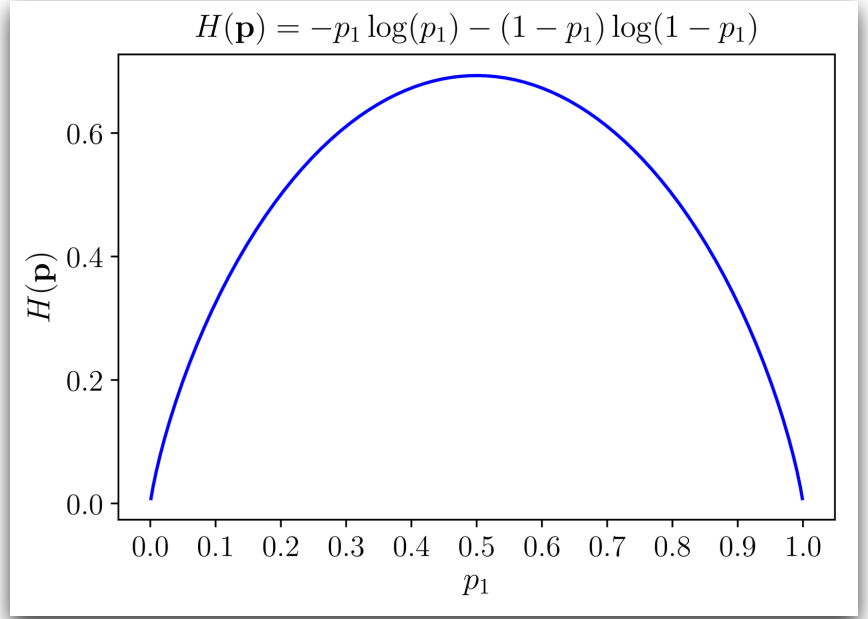
Entropy

From intuition to conceptualisation

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log(p_i)$$

p – probability that an event i occurs

n – number of partitions



https://machinelearningcoban.com/2018/01/14/id3/

Algorithm: How to split the data?

Table 4.1 The Classic Golf Data Set

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	78	FALSE	yes
rain	70	96	FALSE	yes
rain	68	80	FALSE	yes
rain	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rain	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rain	71	80	TRUE	no

Source of Figures: Kotu and Deshpande, 2019, chapter 4

Iterative Dichotomizer 3, or ID3

The original decision tree algorithms (Quinlan, 1986)

(not implemented in scikit-learn)

How to split the data?

Table 4.1 The Classic Golf Data Set

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	78	FALSE	yes
rain	70	96	FALSE	yes
rain	68	80	FALSE	yes
rain	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rain	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rain	71	80	TRUE	no

Source of Figures: Kotu and Deshpande, 2019, chapter 4

$$H = -\sum p_k \log_2 (p_k)$$

Total information without partition

$$I_{\text{no partition}} = -(5/14)\log_2(5/14) - (9/14)\log_2(9/14) = 0.940$$

Row No.	Play	Outlook
3	yes	overcast
7	yes	overcast
12	yes	overcast
13	yes	overcast
4	yes	rain
5	yes	rain
6	no	rain
10	yes	rain
14	no	rain
1	no	sunny
2	no	sunny
8	no	sunny
9	yes	sunny
11	yes	sunny

Source of Figures: Kotu and Deshpande, 2019, chapter 4

Information gain

a measure of how much entropy has been decreased after partitioning with a predictor's values

$$H = -\sum p_k \log_2 (p_k)$$

$$H_{outlook:overcast} = -(0/4)\log_2(0/4) - (4/4)\log_2(4/4) = 0.0$$

$$H_{outlook:sunny} = -(2/5)\log_2(2/5) - (3/5)\log_2(3/5) = 0.971$$

$$H_{outlook:rain} = -(3/5)\log_2(3/5) - (2/5)\log_2(2/5) = 0.971$$

Total information with partitions

$$I_{outlook} = p_{outlook:overcast} * H_{outlook:overcast} + p_{outlook:sunny} * H_{outlook:sunny} + p_{outlook:rain} * H_{outlook:rain}$$

$$I_{outlook} = (4/14) * 0 + (5/14) * 0.971 + (5/14) * 0.971 = 0.693$$

Information gain (reduction of entropy) with partitions

$$I_{outlook,\ no\ partition} - I_{outlook} = 0.940 - 0.693 = 0.247$$

Row No.	Play	Outlook
3	yes	overcast
7	yes	overcast
12	yes	overcast
13	yes	overcast
4	yes	rain
5	yes	rain
6	no	rain
10	yes	rain
14	no	rain
1	no	sunny
2	no	sunny
8	no	sunny
9	yes	sunny
11	yes	sunny

Source of Figures: Kotu and Deshpande, 2019, chapter 4

How best to split the dataset?

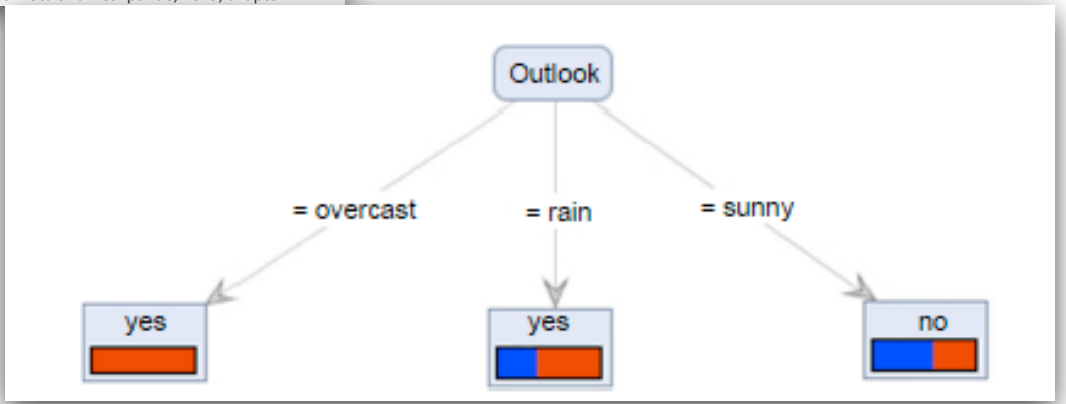
Table 4.2 Computing the Information Gain for All Attributes

Attribute	Information Gain
Temperature	0.029
Humidity	0.102
Wind	0.048
Outlook	0.247

Source of Figures: Kotu and Deshpande, 2019, chapter 4

Selected the predictor with the largest information gain (with partitioning)

- Means of available values can be used to split numerical data
- Numerical data can be classified into groups



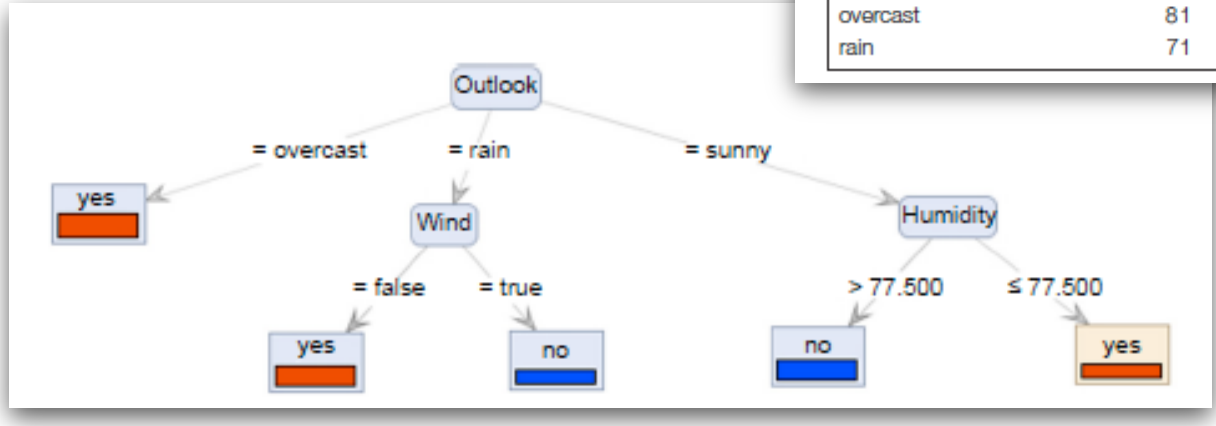
Source of Figures: Kotu and Deshpande, 2019, chapter 4

A full decision tree

Table 4.1 The Classic Golf Data Set

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	78	FALSE	yes
rain	70	96	FALSE	yes
rain	68	80	FALSE	yes
rain	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rain	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rain	71	80	TRUE	no

Source of Figures: Kotu and Deshpande, 2019, chapter 4



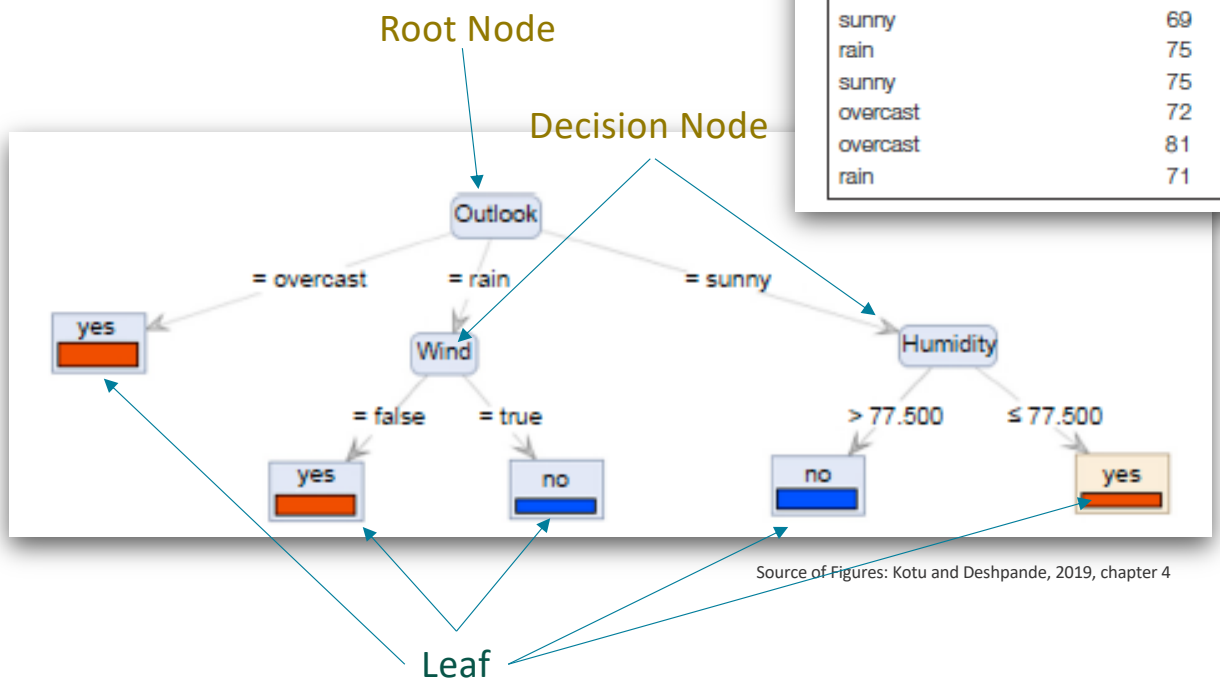
Source of Figures: Kotu and Deshpande, 2019, chapter 4

A full decision tree

Table 4.1 The Classic Golf Data Set

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	78	FALSE	yes
rain	70	96	FALSE	yes
rain	68	80	FALSE	yes
rain	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rain	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rain	71	80	TRUE	no

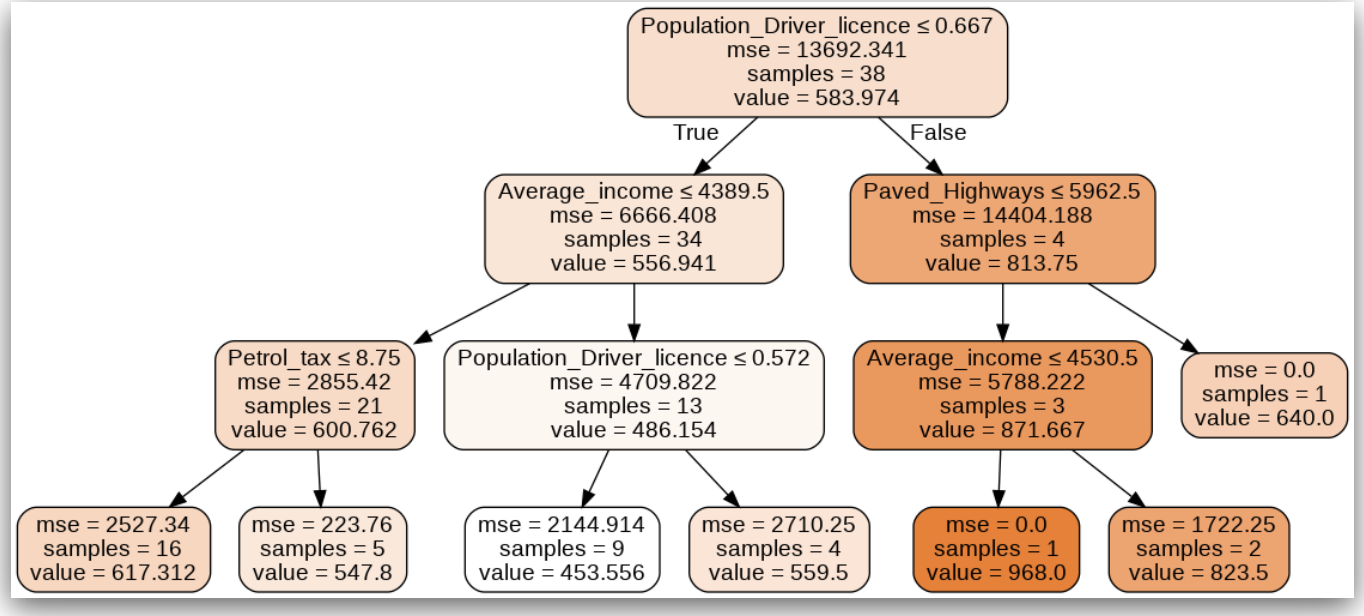
Source of Figures: Kotu and Deshpande, 2019, chapter 4



Source of Figures: Kotu and Deshpande, 2019, chapter 4

Classification And Regression Tree (CART)

Scikit-learn implements an optimized version of CART and constructs binary trees using the feature and threshold that yield the largest information gain at each node.

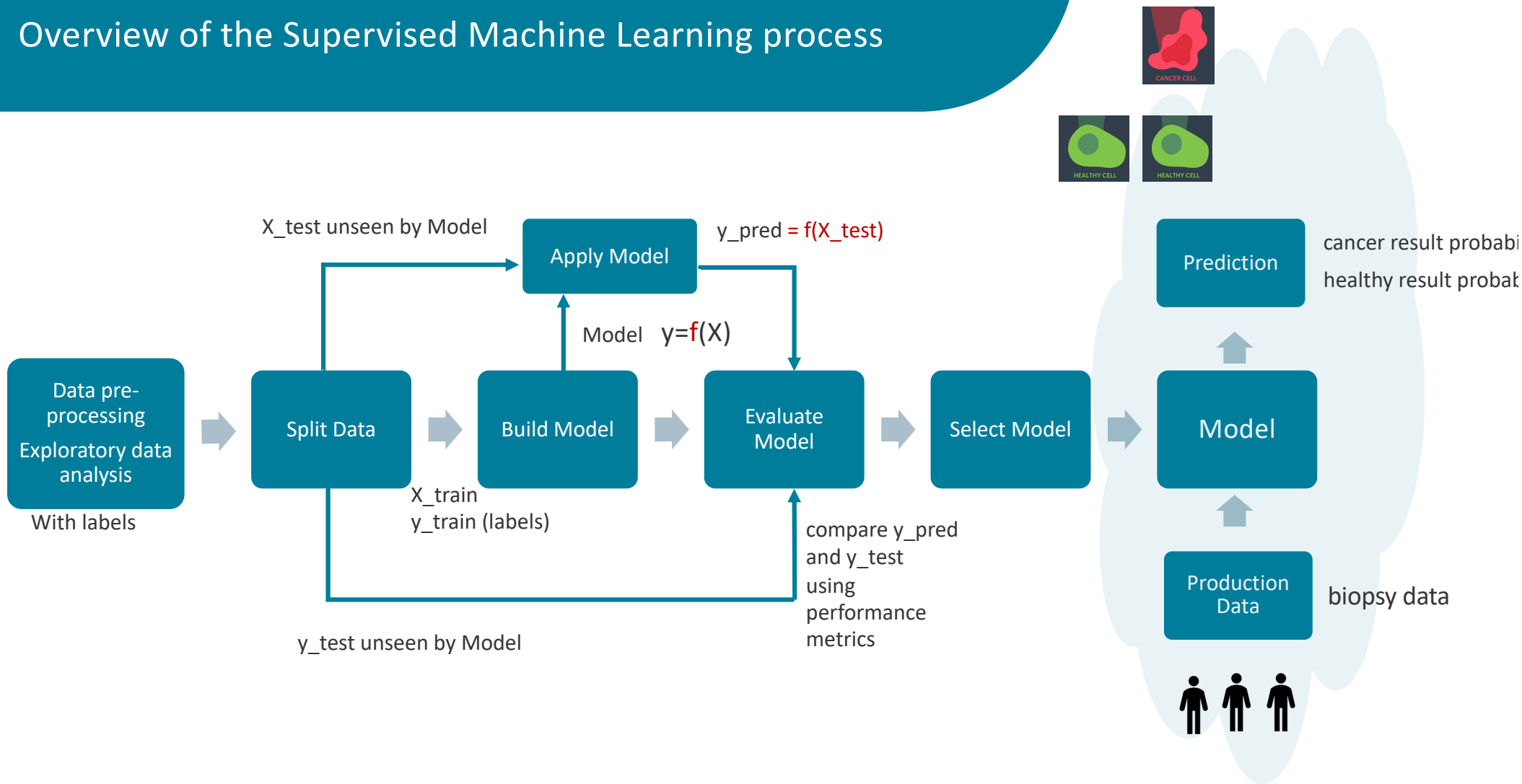


Decision tree regression looks like a decision tree with leaf nodes holding target numeric values.

Implementation decision trees in Python

- No missing data
- Need to transform categorical data into numeric
- Let’s build a model to fit the whole dataset 14 datapoints
- Instantiate a decision tree model **clf**
- Fit the model **clf** with the dataset X and y
- Visualise the model

Overview of the Supervised Machine Learning process



Transform categorical data into numeric

```
#Convert categorical variables to numerical using get dummies
records=pd.get_dummies(records, columns=['Outlook'])

#Convert categorical variables to numerical using LabelEncoder
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
records['Play'] = encoder.fit_transform(records['Play'])
```

	Temperature	Humidity	Play	Outlook_overcast	Outlook_rain	Outlook_sunny	\
0	85	85	0	0	0	1	
1	80	90	0	0	0	1	
2	83	78	1	1	0	0	
3	70	96	1	0	1	0	
4	68	80	1	0	1	0	

	Wind_False	Wind_True
0	1	0
1	0	1
2	1	0
3	1	0
4	1	0

Build a model to fit the whole dataset for the illustration purpose

```
#Selecting predictors
features=['Outlook_overcast', 'Outlook_rain', 'Temperature','Humidity', 'Wind_True']
X=records[features] #Input data
y=records['Play'] # Target variable


from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for model evaluation


# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy") #default criterion is gini, max_depth=25

# Train Decision Tree Classifer with the traning dataset
clf = clf.fit(X, y)
```

Evaluate and visualise the DT model

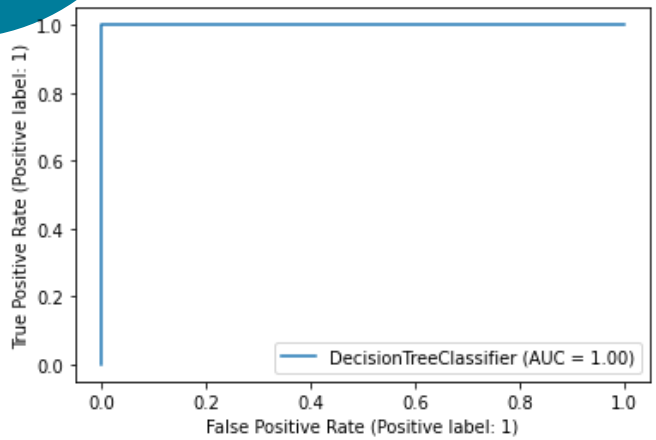
```
#Make predictions for the test dataset
y_pred = clf.predict(X)
```

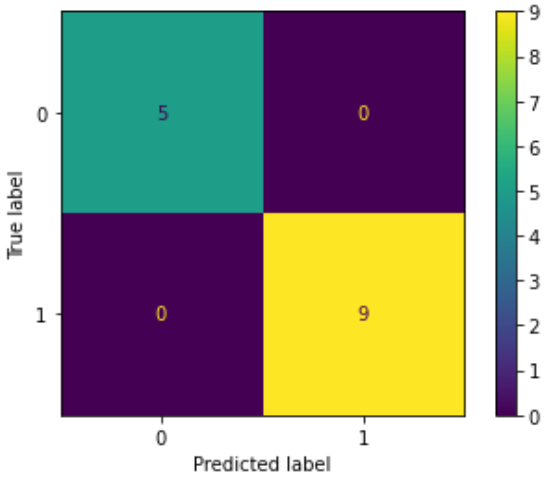
```
#import evaluation functions
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import plot_precision_recall_curve
from sklearn.metrics import plot_confusion_matrix
```

```
#Model Evaluation, calculate metrics: Accuracy, Precision, Recall, F1,
print("Accuracy: ", metrics.accuracy_score(y,y_pred))
print("Precision: ", metrics.precision_score(y,y_pred))
print("Recall: ", metrics.recall_score(y,y_pred))
print("F1: ", metrics.f1_score(y,y_pred))
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1: 1.0
```

WHY SO GOOD?





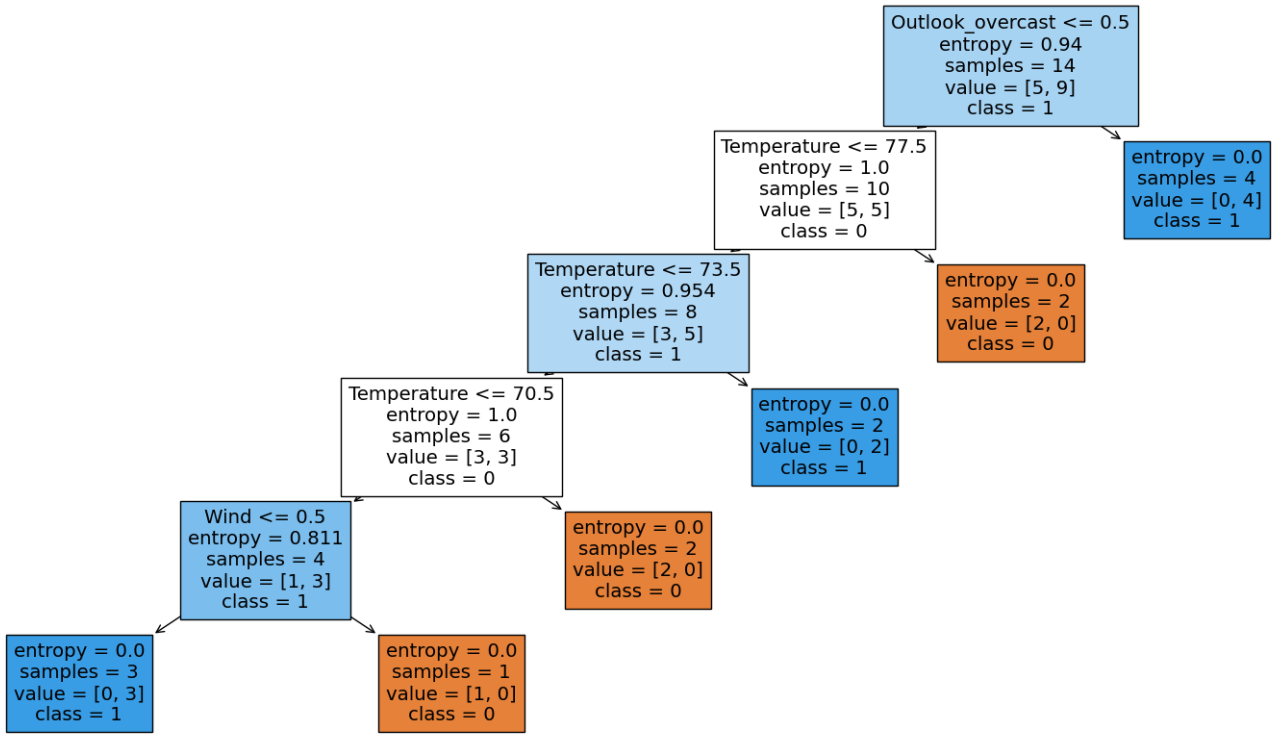
Extract Rules and Visualising Decision Trees in Python

```
#Import plot_tree to extract rules and visualise the decition tree model
from sklearn.tree import plot_tree, export_text
```

```
# Print the decision rules
rules = export_text(clf, feature_names=features)
print(rules)
```

```
|--- Outlook_overcast <= 0.50
|   |--- Temperature <= 77.50
|   |   |--- Temperature <= 73.50
|   |   |   |--- Temperature <= 70.50
|   |   |   |   |--- Wind <= 0.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- Wind >  0.50
|   |   |   |   |   |--- class: 0
|   |   |   |--- Temperature >  70.50
|   |   |   |   |--- class: 0
|   |   |--- Temperature >  73.50
|   |   |   |--- class: 1
|   |--- Temperature >  77.50
|   |   |--- class: 0
|--- Outlook_overcast >  0.50
|   |--- class: 1
```

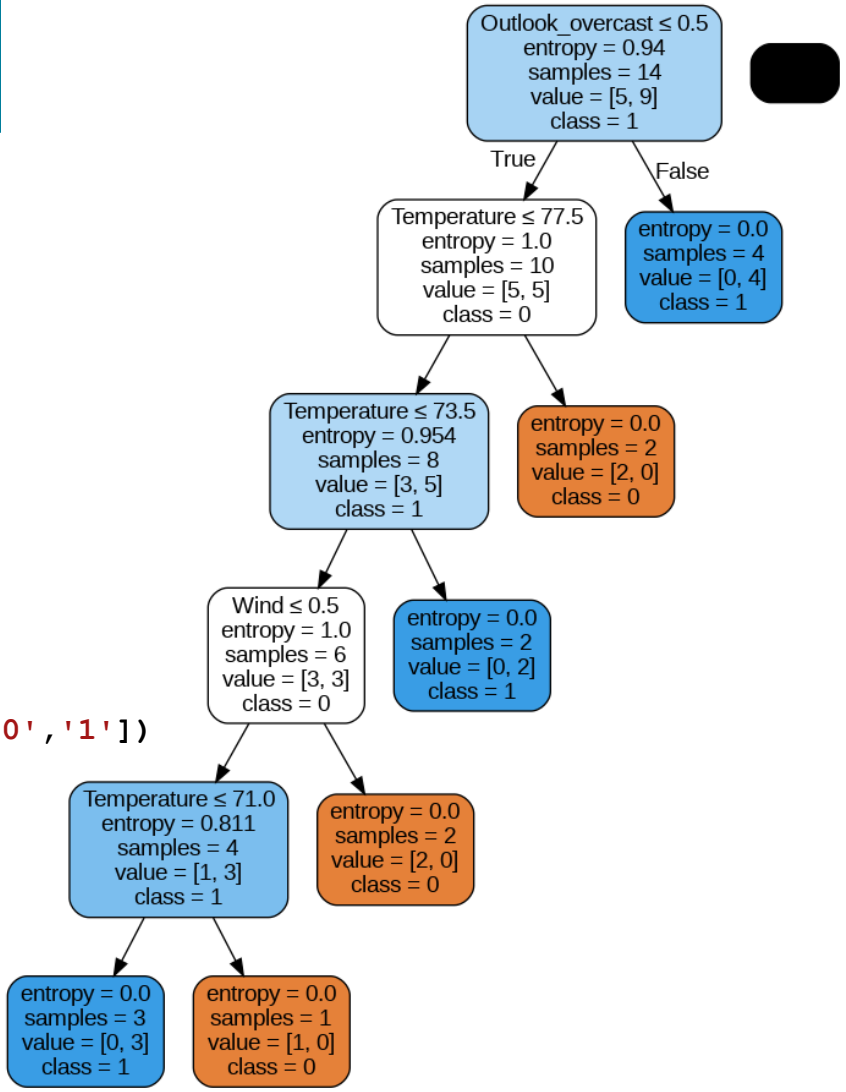
Visualising decision trees in Python




```
# visualize the decision tree
plt.figure(figsize=(20,10))
plot_tree(clf, filled=True, feature_names=features, class_names=['0','1'])
plt.show()
```

Visualising decision trees in Python

```
#Import libraries and classes
import six
import sys
sys.modules['sklearn.externals.six'] = six

from six import StringIO
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
filled=True, rounded=True,
special_characters=True,feature_names = features,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('Golf.png')
Image(graph.create_png())
```



Biopsy example

- Long delay in returning pathology results
- Novice pathologists need training

Business needs

- Sources: Past biopsy data and results
  - Internal and external
- **Data content: Biomarkers and diagnoses**

Data

- **Supervised ML, classification**
- Build and test a logistic regression model that best fit the historical biomarkers and diagnoses

Analytics (ML)

- Decision augmentation: in interpreting biopsy data -> reduced time and improved accuracy
- Training novice pathologists

Deployment

Biopsy example

```
# load libraries
import pandas as pd #for data manipulation and analysis
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from pandas.io.parsers.readers import annotations

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for model evaluation

#print confusion matrix and evaluation report
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import ConfusionMatrixDisplay
```

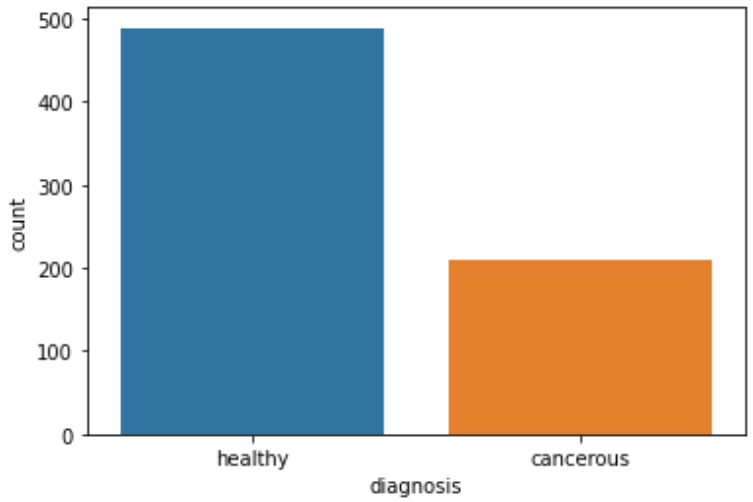
Biopsy example

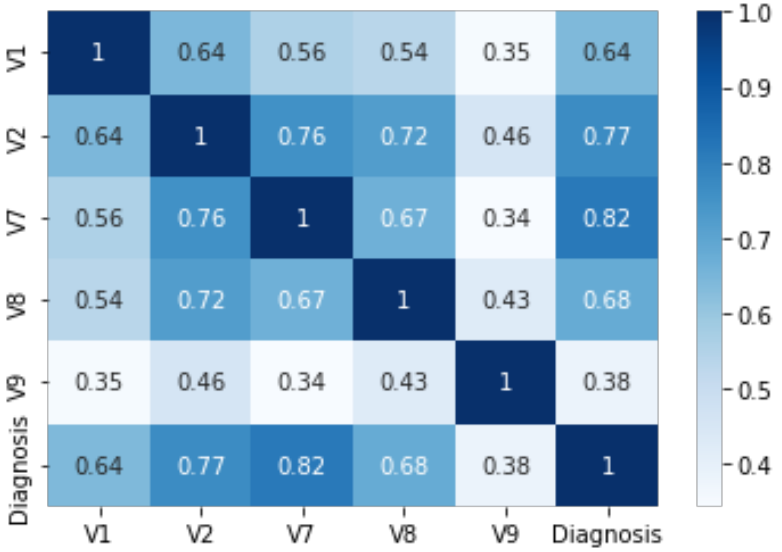
```
# load the dataset
records = pd.read_csv('https://raw.githubusercontent.com/VanLan0/MIS710-ML/main/Datasets/biopsy_ln.csv')

#Inspect missing data
records.isnull().sum()

#drop irrelevant variables
records=records.drop(['ID'], axis=1)

#convert categorical data to numerical
def coding_diagnosis(x):
 if x=='cancerous': return 1
 if x=='healthy': return 0
records['Diagnosis'] = records['diagnosis'].apply(coding_diagnosis)

records.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   V1         699 non-null    int64
 1   V2         699 non-null    int64
 2   V7         699 non-null    int64
 3   V8         699 non-null    int64
 4   V9         699 non-null    int64
 5   diagnosis  699 non-null    object
 6   Diagnosis  699 non-null    int64
dtypes: int64(6), object(1)
memory usage: 38.4+ KB
```

Biopsy example

```
#create barchats
sns.countplot(data=records, x='diagnosis')
```





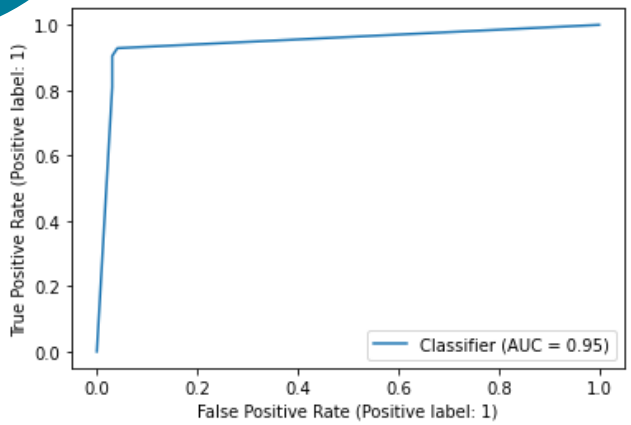

```
sns.heatmap(data=records.corr(), cmap="Blues", annot=True)
```

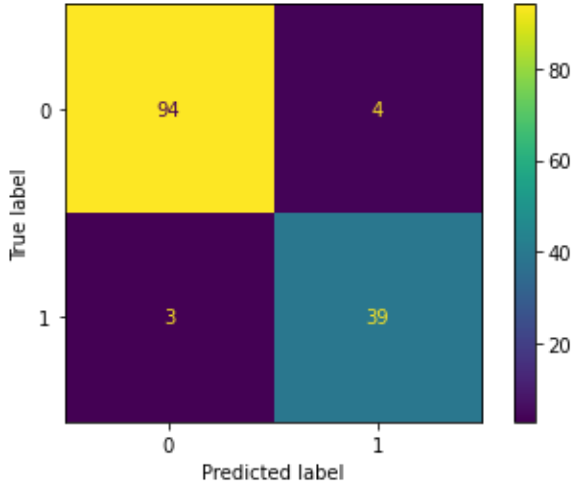
Biopsy example

```
#Selecting predictors and label
features = records.columns[0:5]
X=records[features] #Input data
y=records['Diagnosis'] # Target variable


# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=2024, stratify=y) # 80% training and 20% testing

#inspect the split datasets
print(X_train.head())
print(y_train.head())

print('Training dataset size:',X_train.shape[0])
print('Test dataset size:',X_test.shape[0])
```

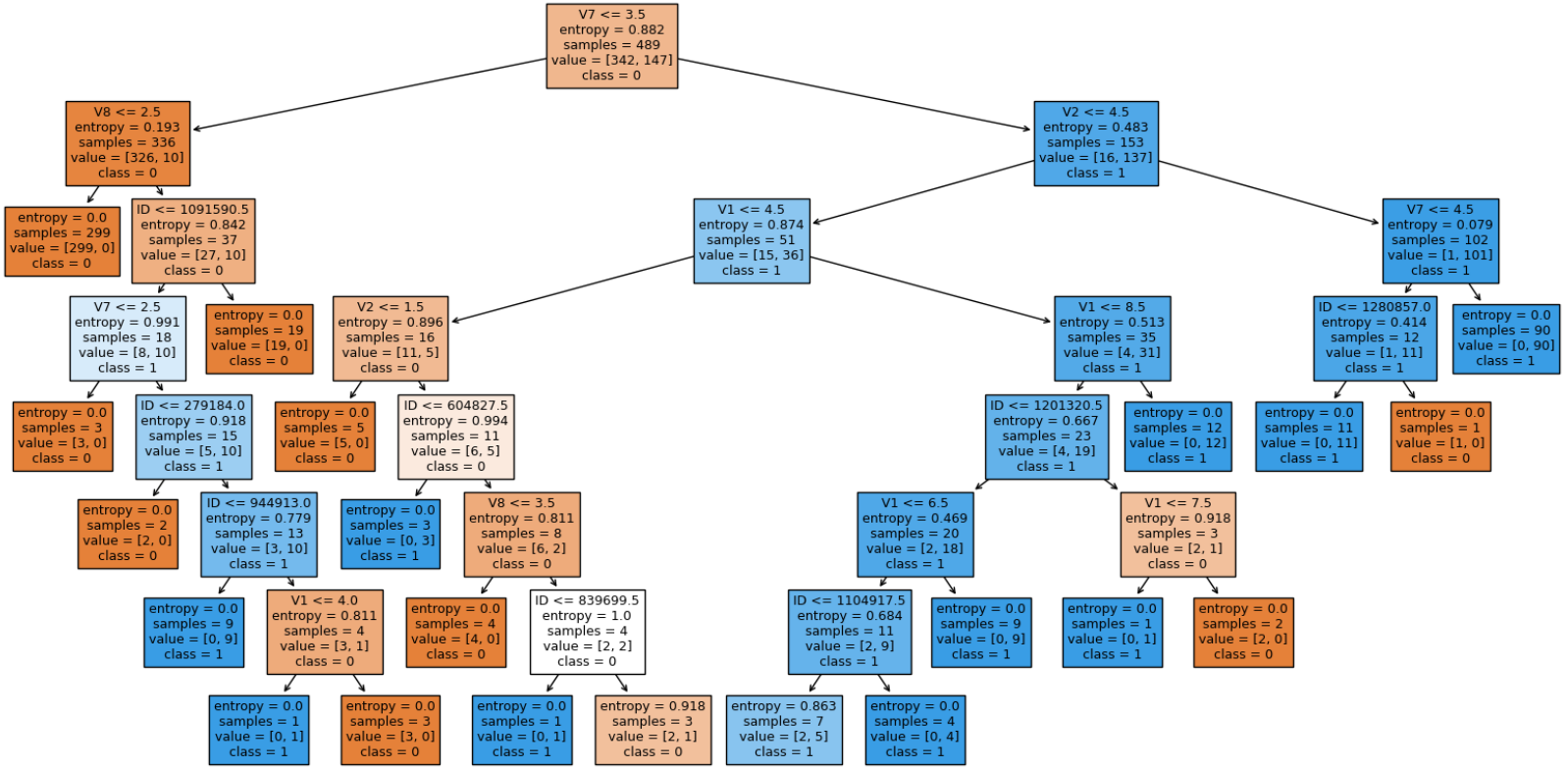
Biopsy example

```
# Create Decision Tree classifier object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=7)
#default criterion is gini, max_depth=25

# Train Decision Tree Classifer with the traning dataset
clf = clf.fit(X_train, y_train)

#Make predictions for the test dataset
y_pred = clf.predict(X_test)

#get predicted probabilities for the main class
y_pred_probs = clf.predict_proba(X_test)
y_pred_probs = y_pred_probs[:, 1]


#print confusion matrix and evaluation report
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```





Biopsy example

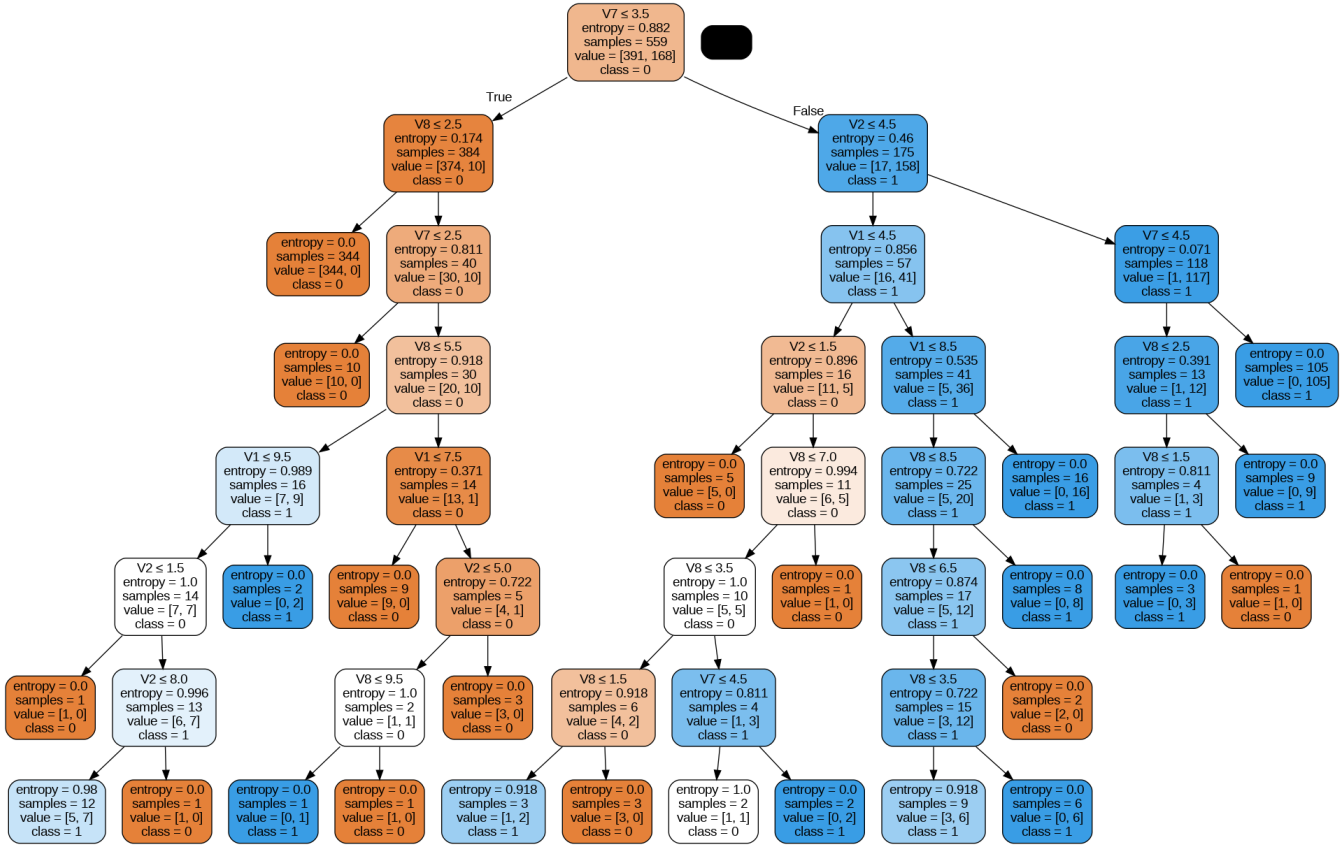



```
# visualize the decision tree
plt.figure(figsize=(20,10))
plot_tree(clf, filled=True, feature_names=features, class_names=['0','1'])
plt.show()
```

Biopsy example

```
#Import libraries and classes
import six
import sys
sys.modules['sklearn.externals.six'] = six

from six import StringIO
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus


dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
filled=True, rounded=True,
special_characters=True,feature_names = features,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('Biopsy.png')
Image(graph.create_png())
```

Biopsy example

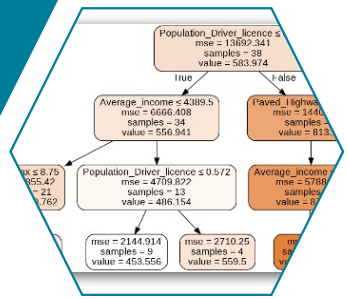


Quiz!

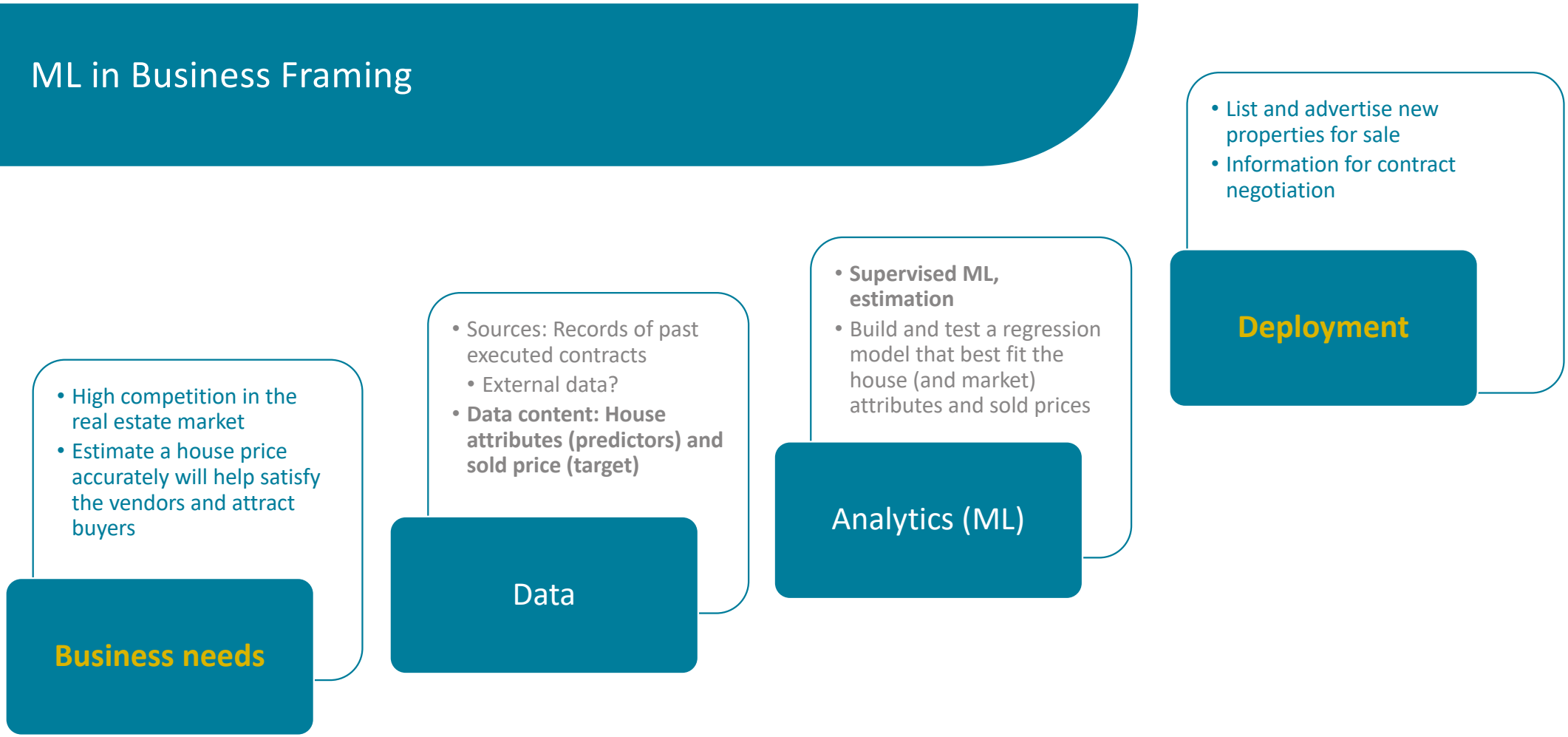
Source: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Predictive Machine Learning with Decision Trees

Decision Tree
Classifier
Regressor



ML in Business Framing



Decision tree regression example

```
print(records.sample(5))
```

```
        area  bedrooms  bathrooms  stories  mainroad  guestroom  basement  \
33  5960.000         3          3        2         1          1         2
151 4400.000         4          1        2         1          0         1
535 3360.000         2          1        1         1          0         1
483 5144.561         3          1        2         1          0         1
384 4500.000         2          1        1         0          0         1

     hotwaterheating  airconditioning  parking  prefarea  furnishingstatus  \
33                 0                1        1         1                 3
151                0                2        2         2                 2
535                0                1        1         1                 3
483                0                1        0         1                 2
384                0                1        0         1                 1

       price
33   8190000
151  5565000
535  2100000
483  2940000
384  3570000
```

```
#Feature selection
features=['area','bedrooms', 'parking','mainroad','guestroom','basement', 'hotwaterheating',
'airconditioning', 'furnishingstatus']
X = records[features]
y = records['price']
```

Decision tree regression example

```
#Import train_test_split function
from sklearn.model_selection import train_test_split #


#Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=2024

#inspect the split datasets
print('Training dataset size:',X_train.shape)
print('Test dataset size:',X_test.shape)
```

```
Training dataset size: (408, 12)
Test dataset size: (137, 12)
```

Decision tree regression

```
#Import DecisionTreeRegressor
from sklearn.tree import DecisionTreeRegressor

#instantiate a decision tree regressor and fit it with the training data
regressor = DecisionTreeRegressor(criterion='absolute_error', max_depth=20, max_leaf_nodes=15,
random_state=2024)
regressor.fit(X_train, y_train)
```

```
criterion can be 'squared_error',
'poisson', 'friedman_mse', 'absolute_error'
```

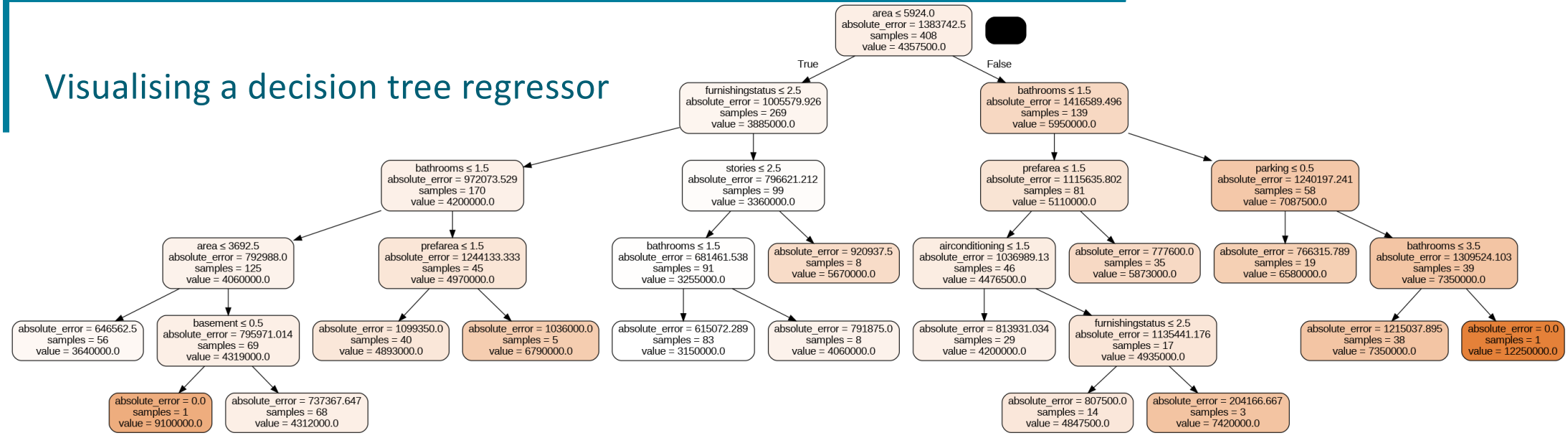
```
#predict prices
y_pred = regressor.predict(X_test)
```

Decision tree regression

```
#Evaluate the model
from sklearn import metrics
print('Mean Absolute Error:', '%.0f' % metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', '%.0f' % metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', '%.0f' %np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 864670
Mean Squared Error: 1431930928386
Root Mean Squared Error: 1196633
```

Visualising a decision tree regressor




```
dot_data = StringIO()
export_graphviz(regressor, out_file=dot_data,
filled=True, rounded=True,
special_characters=True, feature_names = features,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('HousePricePrediction.png')
Image(graph.create_png())
```

Pros and cons and beyond

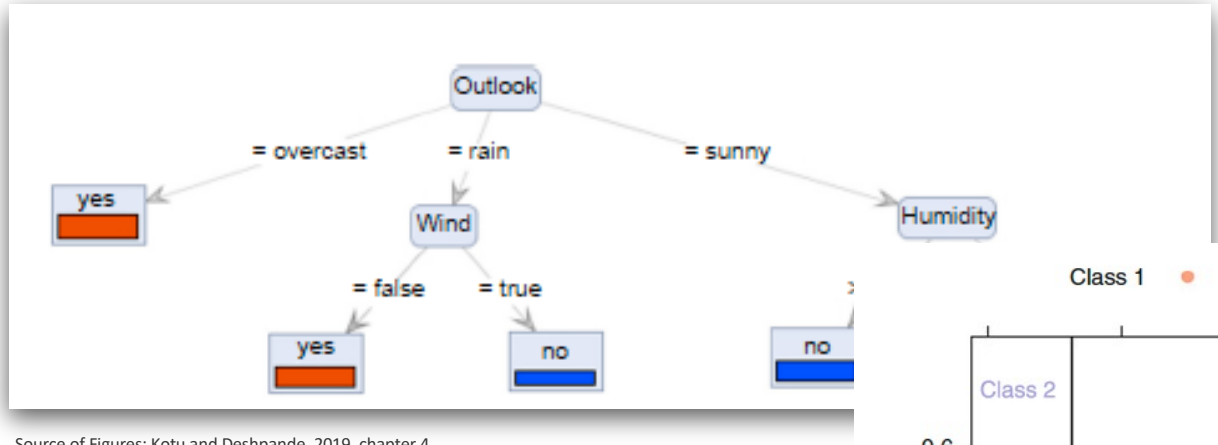
Pros

- Can handle both numerical and categorical targets (with classification and regression trees).
- Non-parametric: No assumptions on data distribution.
- Less effort for data preparation; no need for normalisation.
- Can handle non-linear relationships.
- Easy to understand and interpret.
- Possible to visualise.

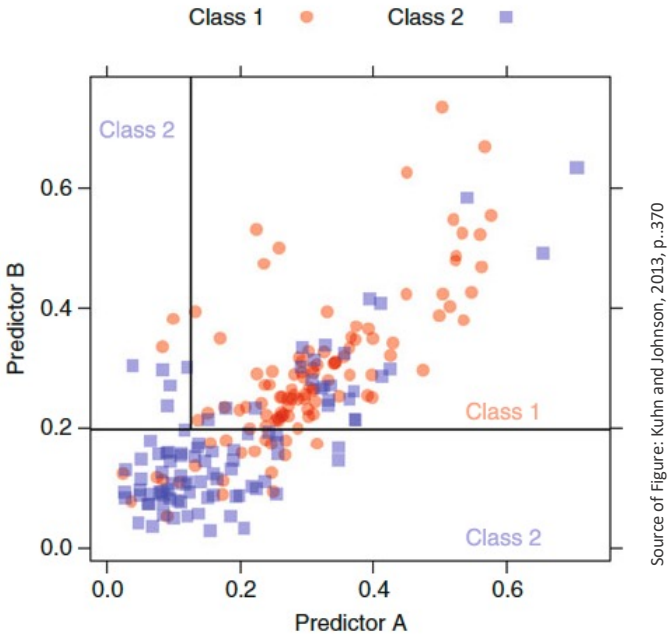
Cons

- Depending on the DT implementation, data preparation is required e.g., numerical or categorical.
- High probability of overfitting: can be addressed through pruning;
- Sensitive to training datasets, leading to overfitting; Instability: small changes to data lead to different trees
- Loss of information when predicting continuous targets: due to the discretisation of continuous values.
- Complex when having multi-class targets
- May not scale well with very large datasets or datasets with a large number of features

What are these decision trees?

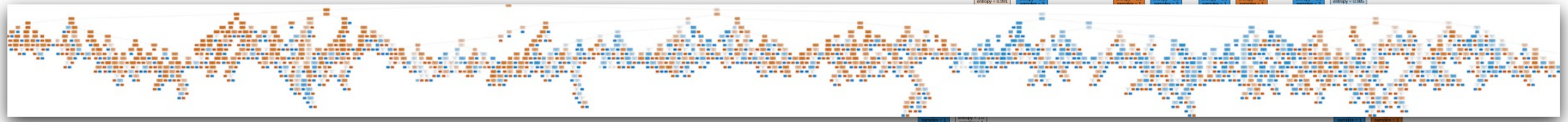


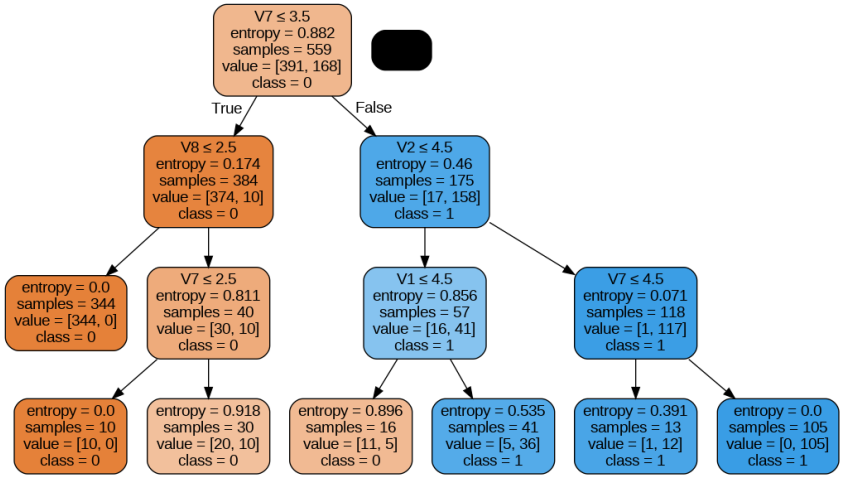
Source of Figures: Kotu and Deshpande, 2019, chapter 4



Source of Figure: Kuhn and Johnson, 2013, p.:370

Accuracy and When to stop partitioning?





Let's talk again in a subsequent Topic!

Hyper parameters - CART implementation in scikit-learn

max_depth *int, default=None* The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split *default=2* The minimum number of samples required to split an internal node

min_samples_leaf *default=1* The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches.

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

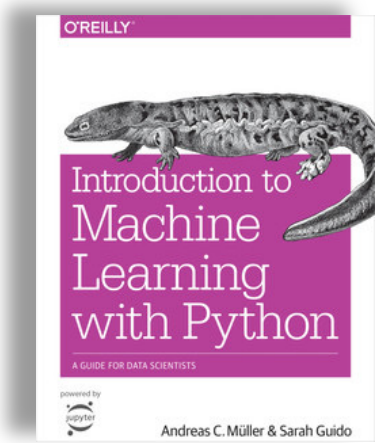
criterion*{"gini", "entropy", "log_loss"}, default="gini"*

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

criterion*{"squared_error", "friedman_mse", "absolute_error", "poisson"}, default="squared_error"*

- Good luck with A1
- Enjoy the break!

Useful readings



Chapter 2

https://learning.oreilly.com/library/view/introduction-to-machine/9781449369880/?sso_link=yes&sso_link_from=Deakin

AND See useful sites in Lecture 5 and Lab 5.

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

- Navlani, A., Decision Tree Classification in Python, https://www.datacamp.com/community/tutorials/decision-tree-classification-python
- Robinson , S., Decision Trees in Python with Scikit-Learn, https://stackabuse.com/decision-trees-in-python-with-scikit-learn/#disqus_thread
- Vũ Hữu Tiệp, Machine Learning cơ bản, https://machinelearningcoban.com/2018/01/14/id3/
- Decision Tree Algorithm Explained, Chauhan N. S., https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html
- Kuhn and Johnson, Applied Predictive Modelling, 2013, chapters 4, 8