Rule based learning not useful for problems such as speech recognition

\Most common is supervised

Featur is measurable property, target is what you want to predict

Can solve; predict a continuous value (regression problem), classification problem like separating into flower species

Data has to be in reconised format, in this case numpy arrays

Features matrix 2d arrya, target vector is 1d

Use of .values can help convert into numpy array

Check if 2d

Can’t have null values

Target variable no second dimension when do .shape, but x needs one

Df.dropna(how=’any’)

To do x you need second set of square brackets

X=df.loc[:,[‘x’]].values#makes 2d

Y=df.loc[:,’Y’].values#makes 1d

Fit\_intercept=true#better is false intercept is 0,0

Measure regression by R\*\*2

Reg.score(x,y)#how to calc r2

Reg.coef\_ is slope

Reg.intercept\_#intecept

#model performance

Score=reg.score(x\_test,y\_test)

## Logistic regression is used for classification

Binary classification

Scaling

Using probability to see which of binary outcomes is more likely

Chart

Description automatically generated#blue in green is a false result

Confusion matrix provides more info on what is being misclassified

Multiclass logistic regression

Split task into multiple binary classification, create model on each, 1vs all or 1 vs rest strategy

# Decision trees

Decision trees don’t need standardised features

Like Kmeans there is a max accuracy after a certain depth

Decision trees tend to overfit training set

Tree is more interpretable if colourful

Bagged trees rely on that multiple decision trees prevent over fitting as rely on eachother

## Random Forests

Random forests build off bagged trees, bagged trees may end up too correlated and random forests helps split this up further

Estimaters are number od decision trees used

Feature importance metric is crucial to seeing how made I this model

## Which ML algorithm to use

Algorithm

Diagram

Description automatically generated

# Unsupervised Machine Learning

Only provides features matric, no target vector

Clustering or dimensionality reduction

Natural grouping like market segmentation

Dimensionality reduction – data compression algorithms, makes visualisation easier and fitting quicker

## Kmeans

Needs standardisation

vIsualise on scatter to look for natural clusters

number of clusters set as k

## PCA Principle component algorithm

Technique used to smartly reduce dimensionality while losing minimum data

Good for data visualisation

Explained variance is good to see where comes from

How to speed up fitting

Reduce rows/columns can help but hard but methods like PCA help

Can be used for images

## How to build pipelines

Lot of code, so how keep organised and bug free?

3 fit steps if just in an order

With a pipeline like below there is one fit step

%matplotlib inline

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

from sklearn.pipeline import Pipeline

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.linear\_model import LogisticRegression

‘’’ # Train Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df[df.columns[:-1]], df['label'], random\_state=0)

# Create a pipeline

pipe = Pipeline([('scaler', StandardScaler()),

('pca', PCA(n\_components = .90, random\_state=0)),

('logistic', LogisticRegression())])

pipe.fit(X\_train, y\_train)

# Get Model Performance

print(pipe.score(X\_test, y\_test))’’’

#pipelines much easier and quite cool so I will be using in future`