Interactive Technologies Institute
LARSyS

Google Developer Groups
Madeira

UNIVERSIDADE da MADEIRA

MADEIRA INTERNATIONAL WORKSHOP IN MACHINE LEARNING

IKnowD

ORDEM DOS ENGENHEIROS TÉCNICOS
PORTUGAL

ORDEM DOS ENGENHEIROS TÉCNICOS

Platinum sponsor:

malogica solutions

Silver sponsor:

asseco
PST

Institutional support

arditi

startup madeira

Outline

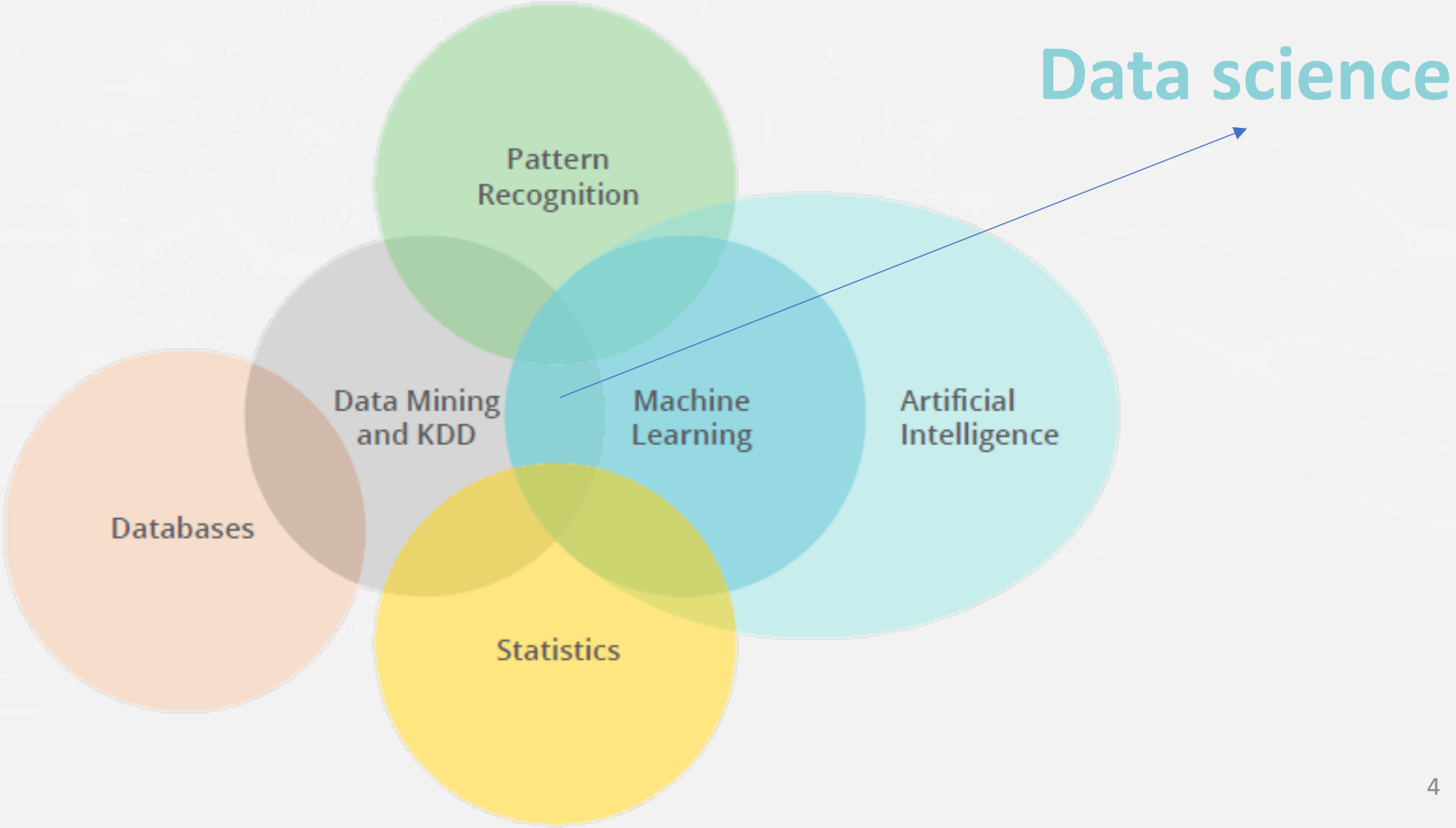
1- What is machine learning?

2- Examples of machine learning applications

3- Key remarks

1- What is machine learning?

1- What is machine learning?



1- What is machine learning?

"Field of study that **gives computers** the **ability to learn without** being **explicitly programmed**"

Arthur Samuel, 1959

"A computer program is said to **learn from experience** E with respect to some task T and some performance measure P if its performance on T, as measured by P, **improves with experience** E"

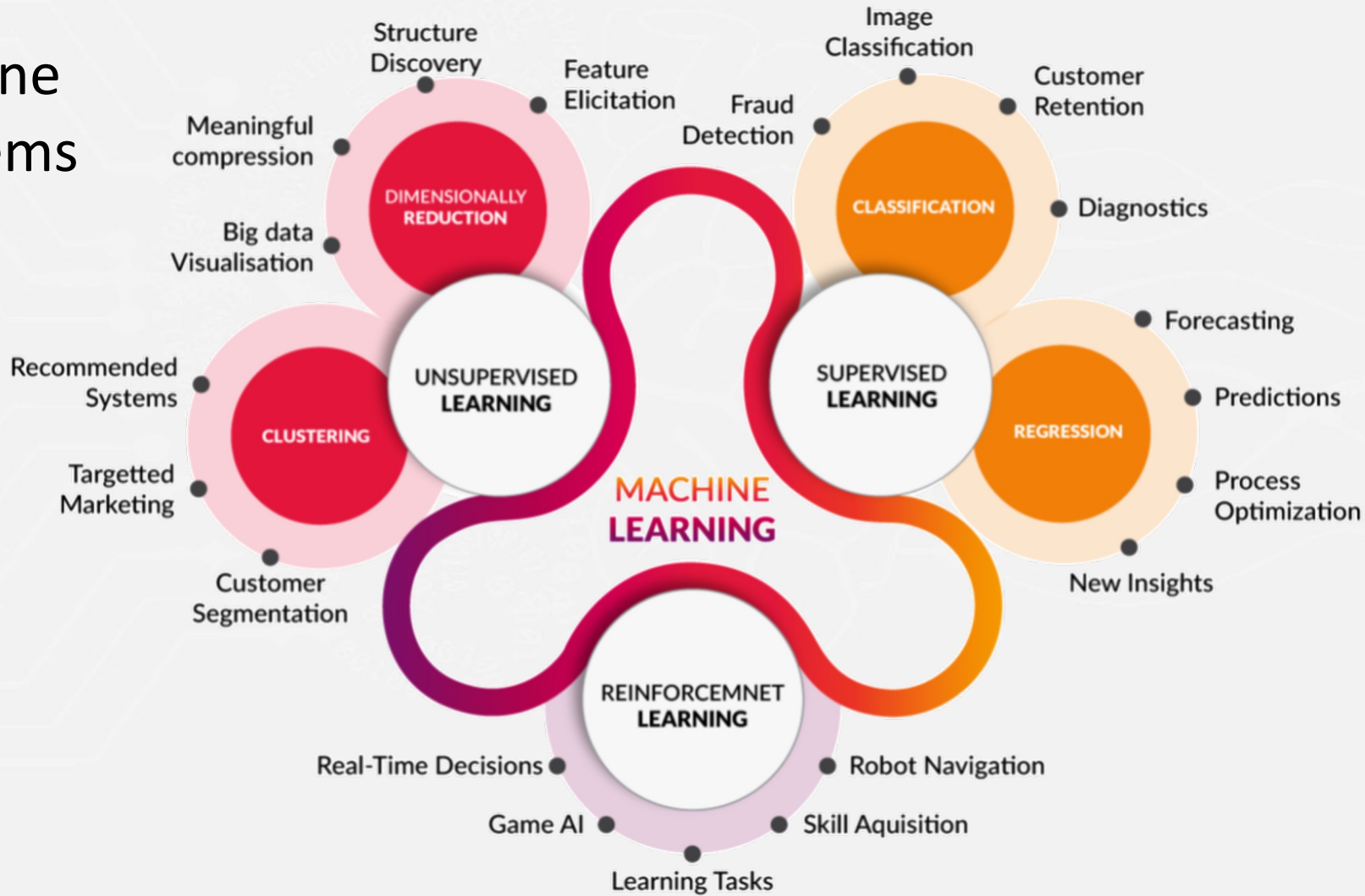
Tom Mitchell, 1997

Using data for **answering questions**

Training **Predicting**

1- What is machine learning?

Types of machine learning problems



1- What is machine learning?

Supervised

Unsupervised

Reinforcement

Learn through **examples** of which we know the desired output (what we want to predict).

-Is this an image of a cat, a dog or a horse?

-Are these emails spam or not?

-What will be the energy production of this windfarm?

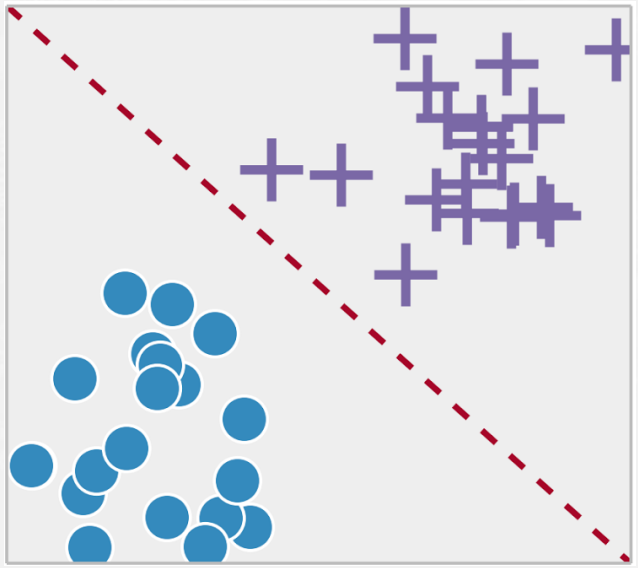
1- What is machine learning?

Supervised

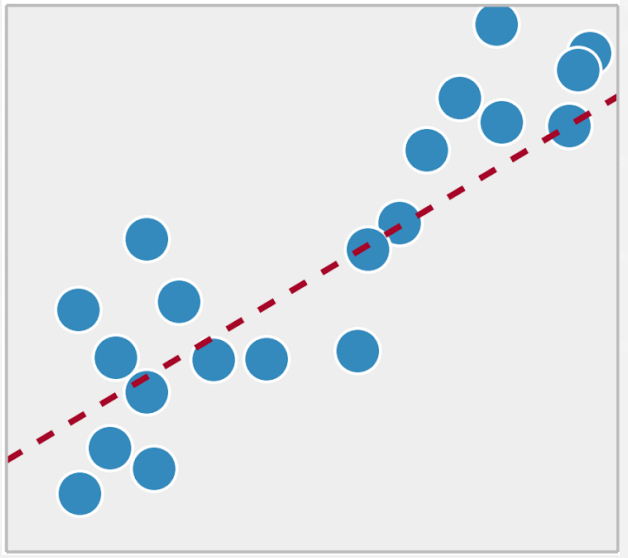
Unsupervised

Reinforcement

Classification



Regression



1- What is machine learning?

Supervised

Unsupervised

Reinforcement

There is **no desired output** thus it learns something about the data.

-I have set of sports balls and want to divide them into five groups according to how similar they are.

-Learn a representation from a group of photos from a cat.

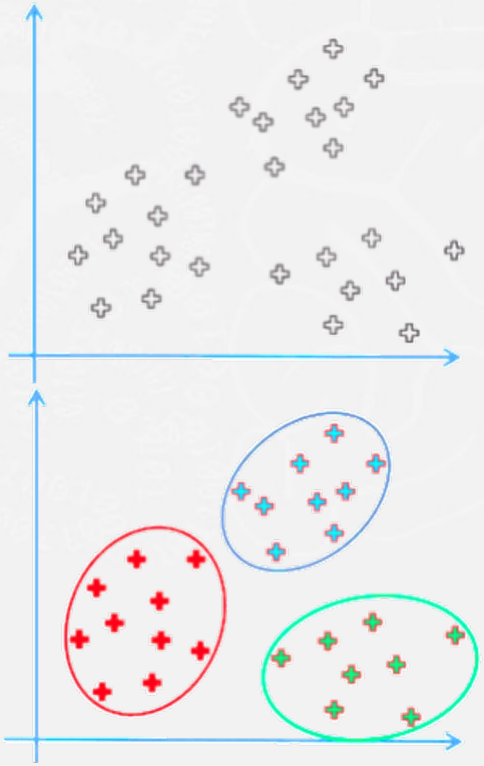
1- What is machine learning?

Supervised

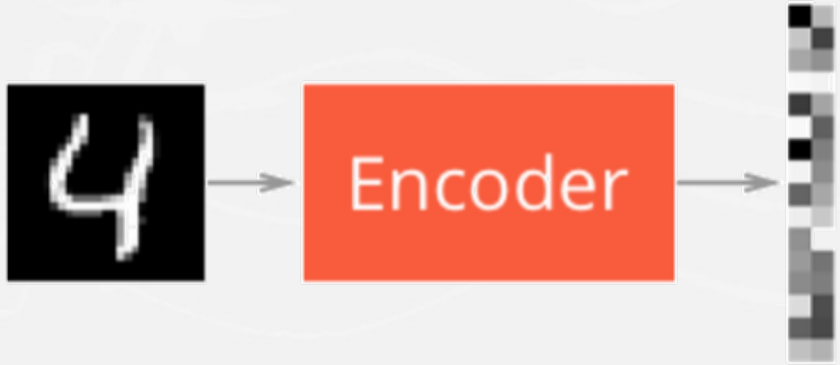
Unsupervised

Reinforcement

Clustering



Dimensionality reduction



1- What is machine learning?

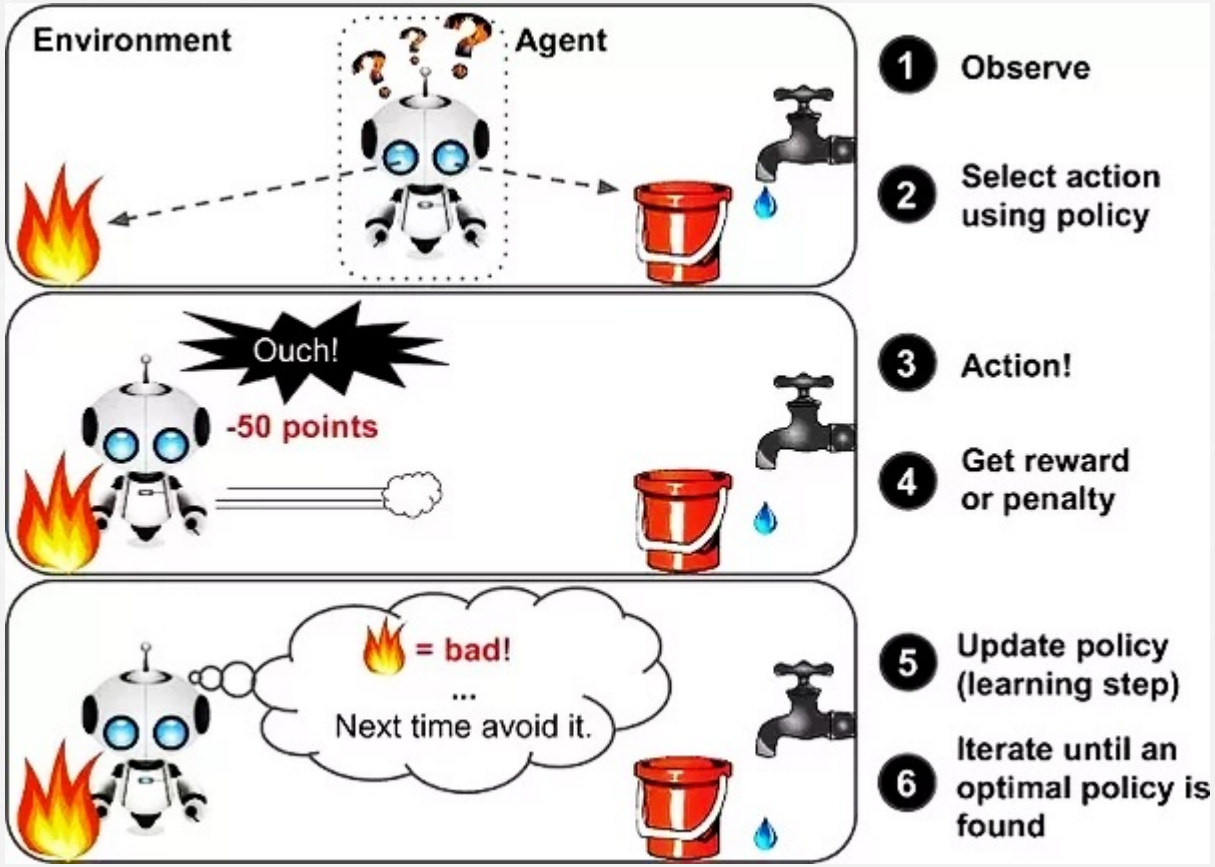
- Supervised
- Unsupervised
- Reinforcement

An agent **interacts** with an **environment** and watches the results of the interaction. The **environment** gives **feedback** via positive or negative **reword signal**.

-Decide the next action a character should do in a game.

1- What is machine learning?

- Supervised
- Unsupervised
- Reinforcement

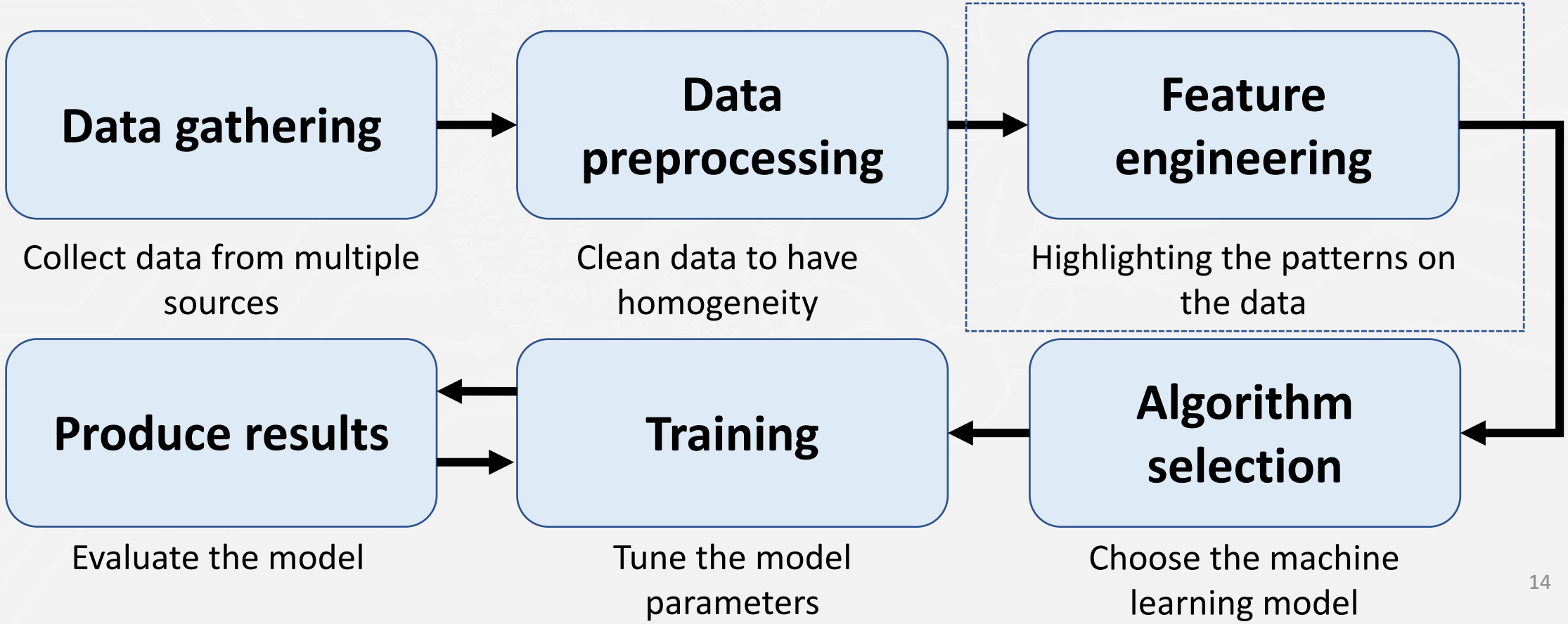


2- Examples of machine learning applications

2- Examples of machine learning applications

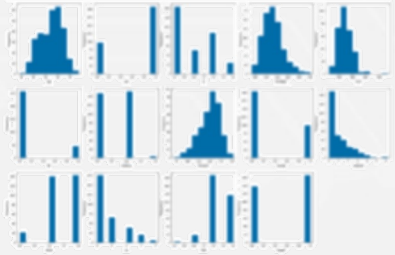
Steps to solve a machine learning problem

Step not mandatory when using deep learning



Exploratory Data Analysis (EDA)

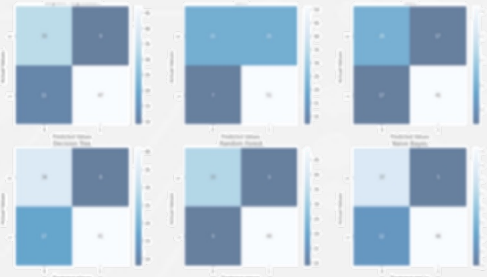
1) **Histogram:** df.plot(kind = 'hist')
2) **Box Plot:** sns.boxplot()
3) **Grouped Bar Chart:** sns.countplot()



Model Evaluation

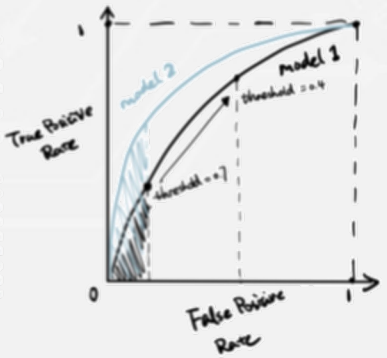
Confusion Matrix

confusion_matrix(y_test, y_pred)

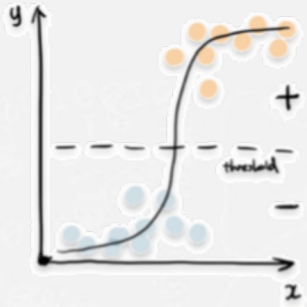


ROC & AUC

metrics.auc(fpr, tpr)

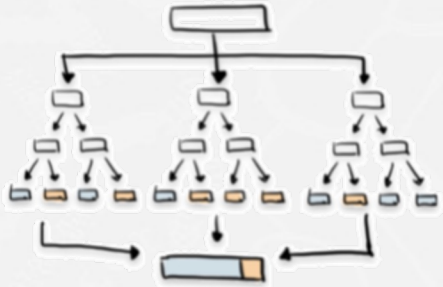


Logistic Regression

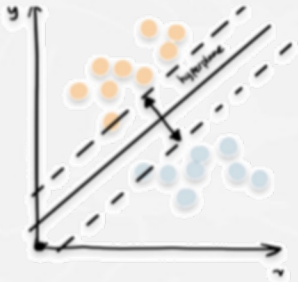


Decision Tree

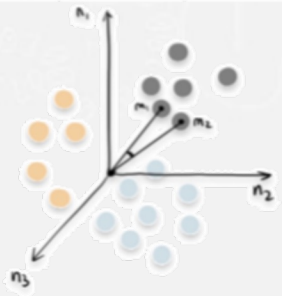
Random Forest



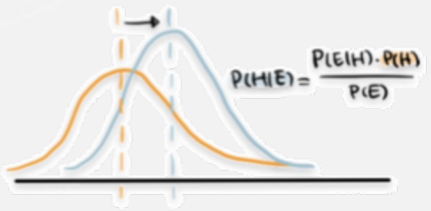
Support Vector Machine



K Nearest Neighbour



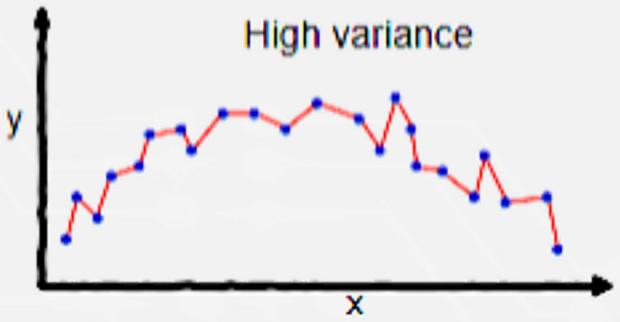
Naive Bayes



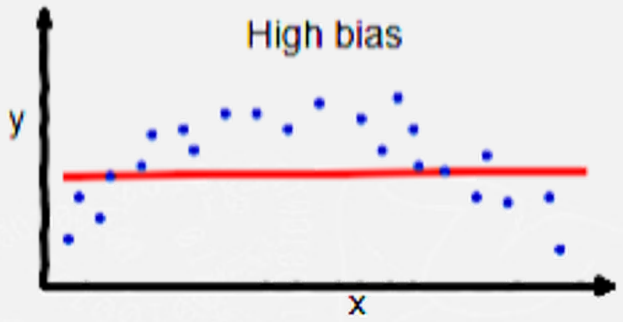
Dataset

Training | **Testing**

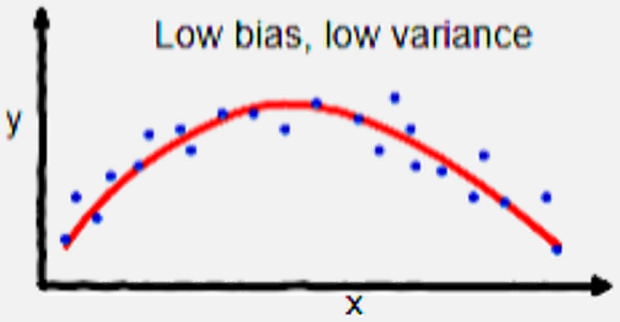
Training | **Validation** | **Testing**



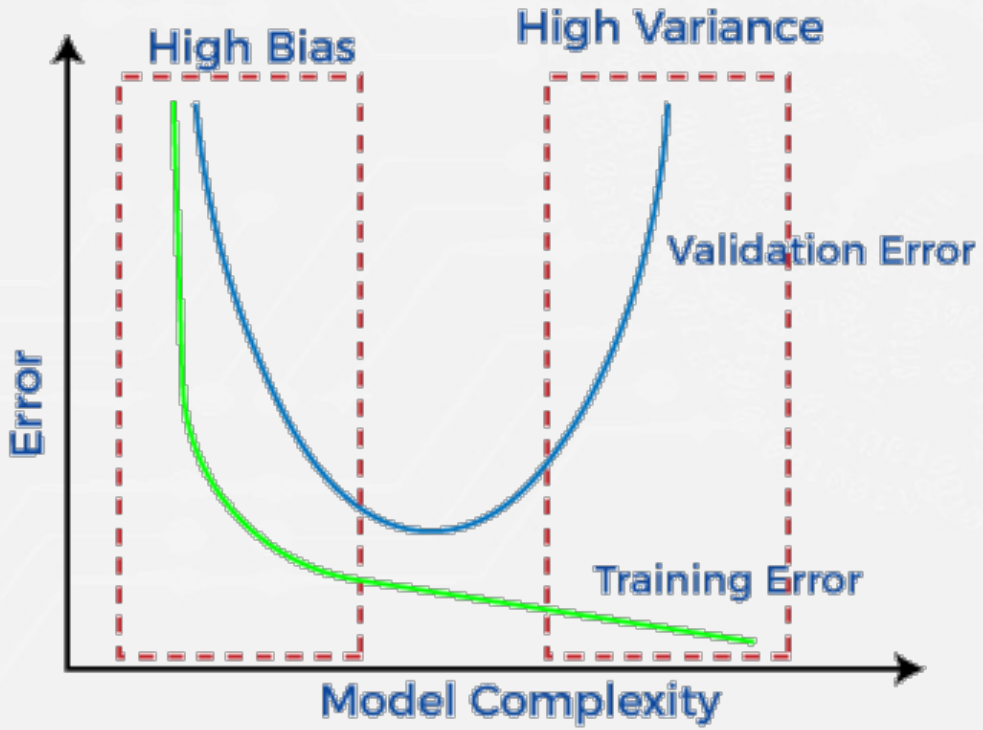
overfitting

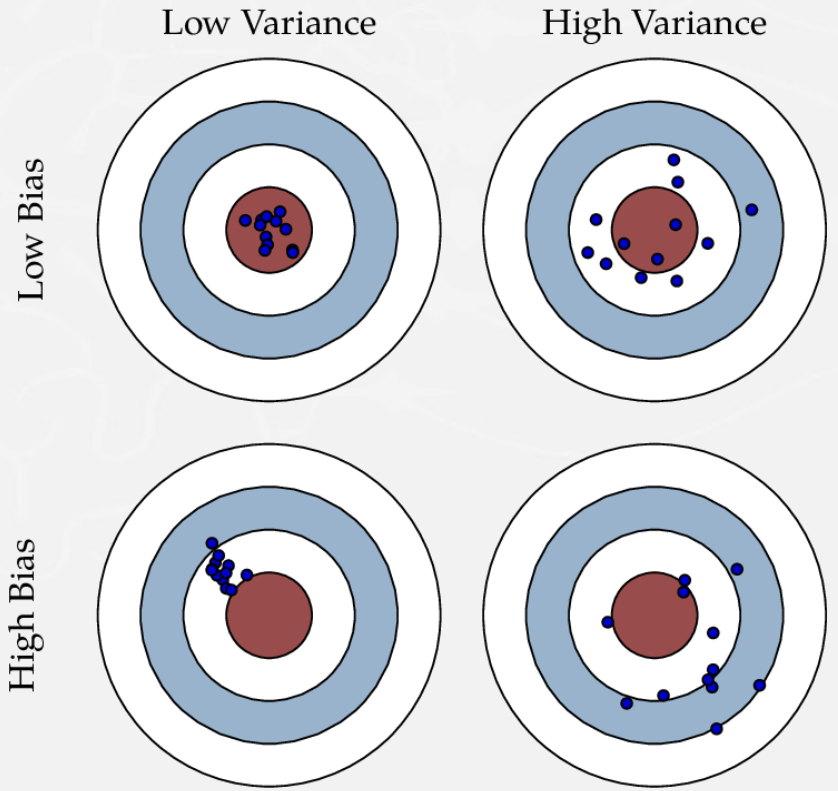


underfitting



Good balance



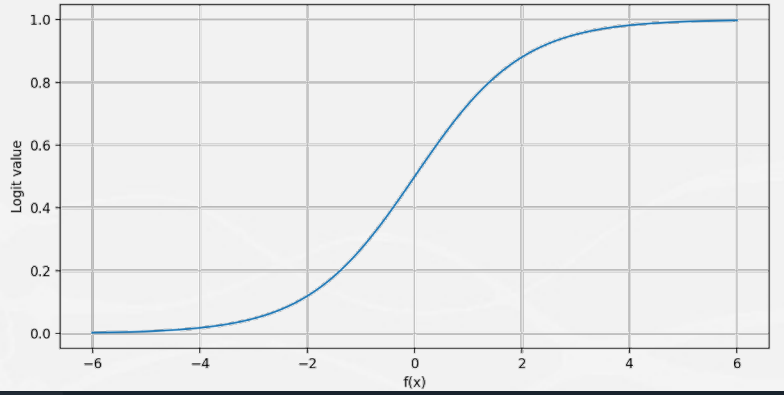


2- Examples of machine learning applications

Classification

2- Examples of machine learning applications

Classification - **Logistic regression**



Logistic function:

$$y = \frac{e^{f(x_1,x_1,\ldots,x_n)}}{1 + e^{f(x_1,x_1,\ldots,x_n)}} = \frac{1}{1 + e^{-f(x_1,x_1,\ldots,x_n)}}$$

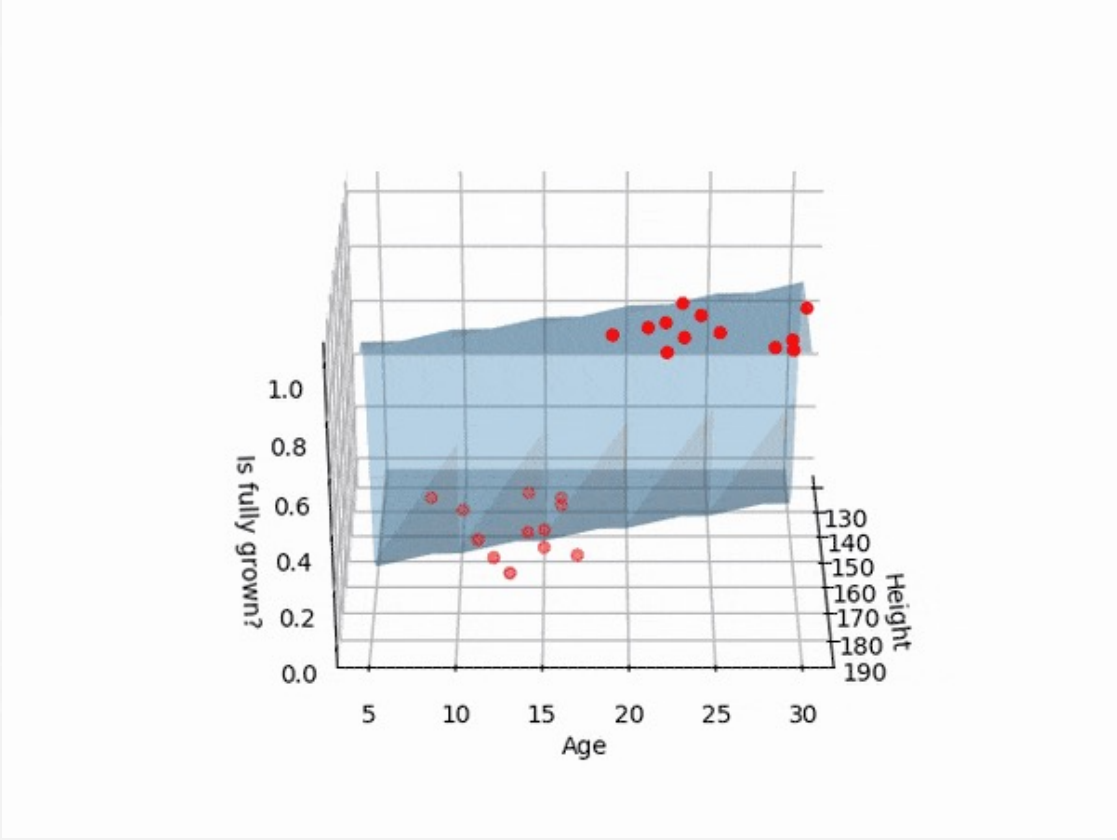
Where

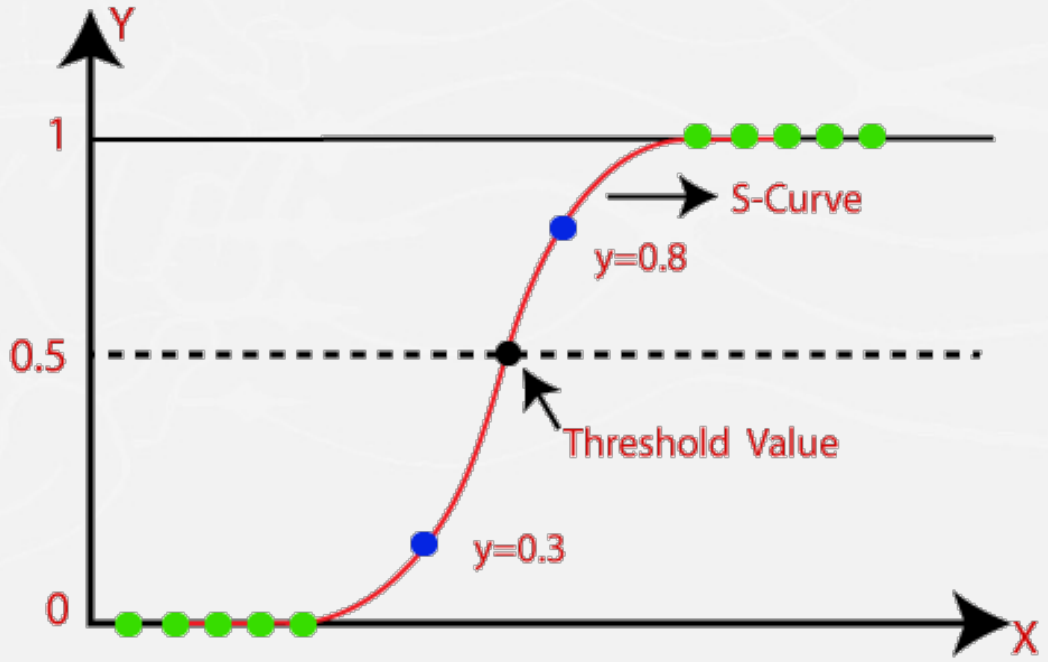
$$f(x_1, x_1, \ldots, x_n) = a_0 + a_1 x_1 + \cdots + a_n x_n$$

```
1   """
2   Plot logit function
3   """
4   #Sigmoid Function
5   import matplotlib.pyplot as plt
6   import numpy as np
7   import math
8   def logit(fx):
9       return math.exp(fx)/(1+math.exp(fx))
10  fx = np.linspace(-6,6,100)
11  y = np.vectorize(logit)
12  plt.plot(fx,y(fx))
13  plt.xlabel('f(x)')
14  plt.ylabel('Logit value')
15  plt.grid()
```

2- Examples of machine learning applications

Classification - **Logistic regression**





2- Examples of machine learning applications

Diabetes prediction

Pima Indian Diabetes dataset

ATTRIBUTE	DESCRIPTION	Mean	Standard deviation	Min/max
Preg	Number of pregnancies	3.8	3.4	1/17
Plas	Plasma glucose concentration in an oral glucose tolerance test	120.9	32	56/197
Pres	Diastolic blood pressure	69.1	19.4	24/110
Skin	Triceps skin fold thickness	20.5	16	7/52
Insu	2-Hour serum insulin	79.8	115.2	15/846
Mass	Body mass index	32	7.9	18.2/57.3
Pedi	Diabetes pedigree function	0.5	0.3	0.0850/2.3290
Age	Age of an individual	33.2	11.8	21/81
class	Tested positive / negative			

2- Examples of machine learning applications

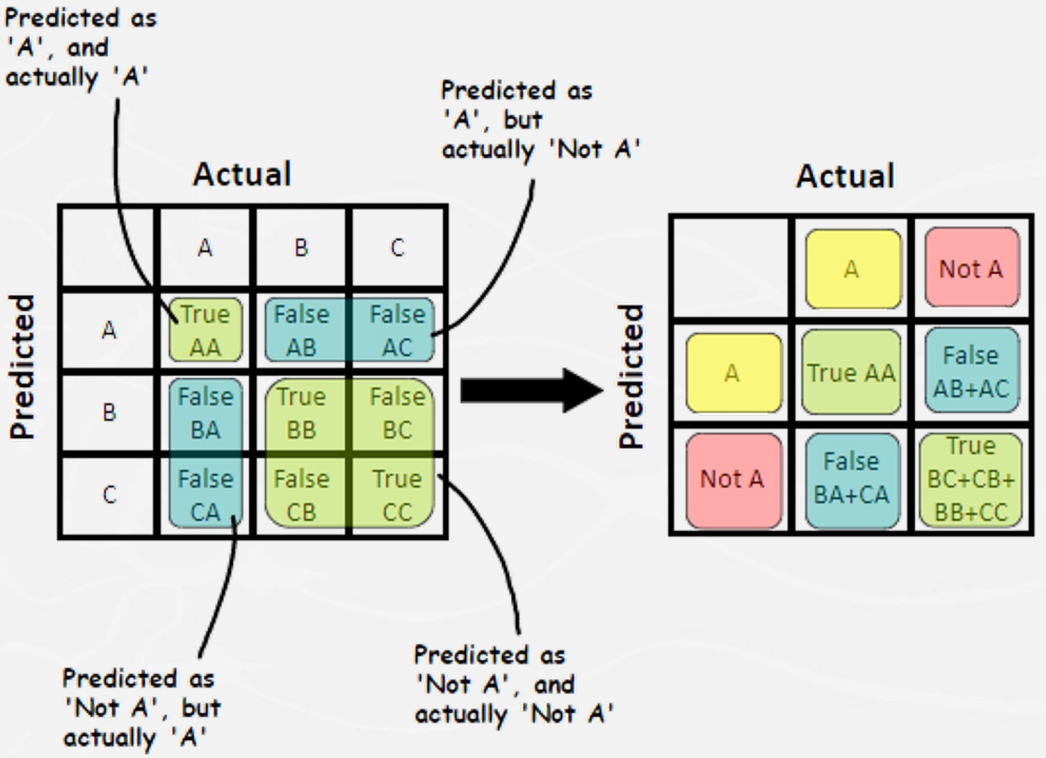
Classification - **Logistic regression**

```
"""
Loading Data
"""
#import pandas
import pandas as pd
col_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
             'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
# load dataset
pima = pd.read_csv("diabetes.csv", header=None, names=col_names)
pima = pima.iloc[1:]
"""
Selecting Feature
"""
#split dataset in desired features to be examined and target variable
feature_cols = ['Pregnancies', 'Insulin', 'BMI', 'Age','Glucose',
                'BloodPressure','DiabetesPedigreeFunction']
X = pima[feature_cols] # Features
y = pima.Outcome # Target variable
"""
Splitting Data
"""
# split X and y into training and testing sets
from sklearn.model_selection  import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,
                                               random_state=0)
```

```
"""
Model Development and Prediction
"""
# import the class
from sklearn.linear_model import LogisticRegression
# instantiate the model (using default parameters with maximum of 1000 runs)
logreg = LogisticRegression(max_iter=1000)
# fit the model with data
logreg.fit(X_train,y_train)
# make predictions on the testing set
y_pred=logreg.predict(X_test)
"""
Model Evaluation
"""
from sklearn import metrics
y_test = [ int(s) for s in y_test]
y_pred = [ int(s) for s in y_pred]
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
import matplotlib.pyplot as plt
y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

2- Examples of machine learning applications

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$



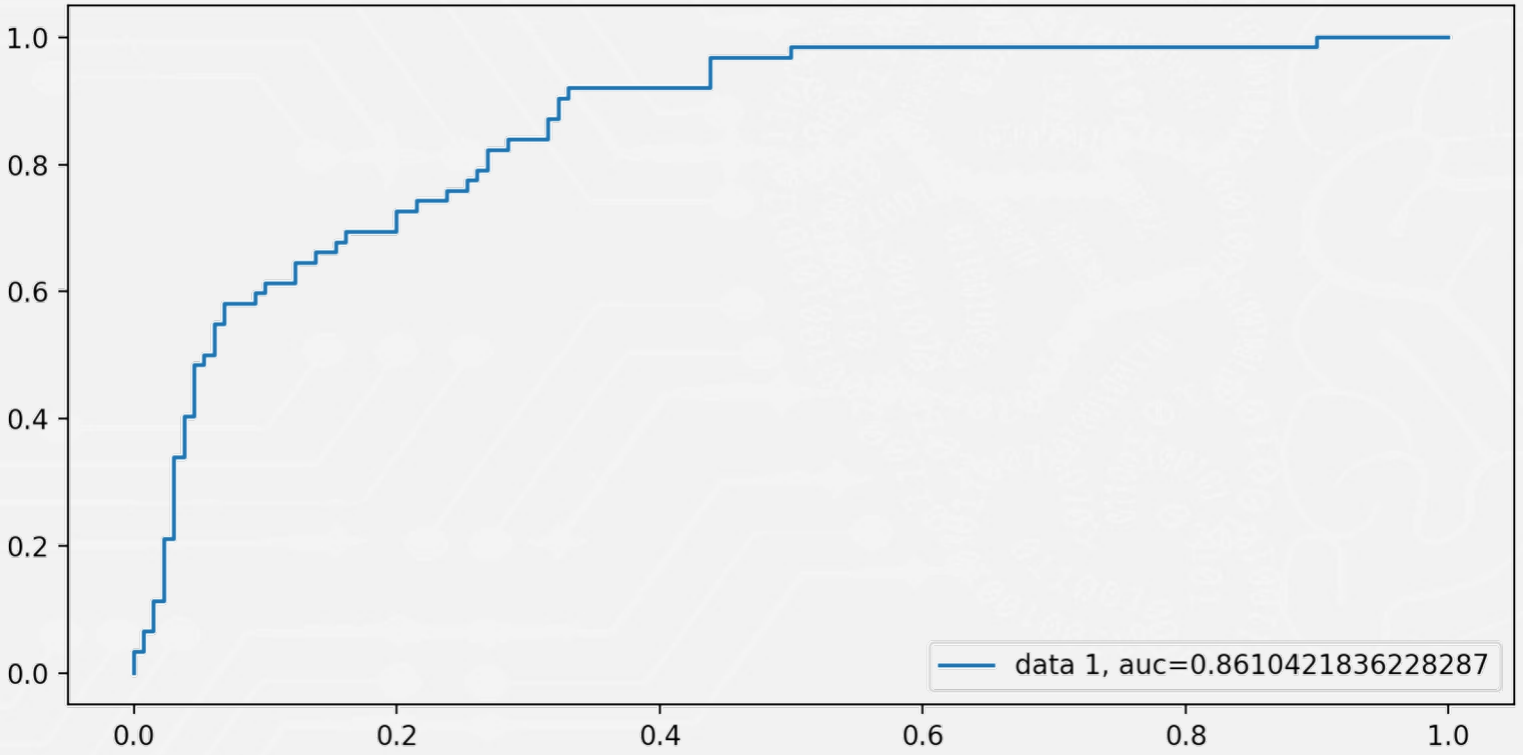
id	color
1	red
2	blue
3	green
4	blue

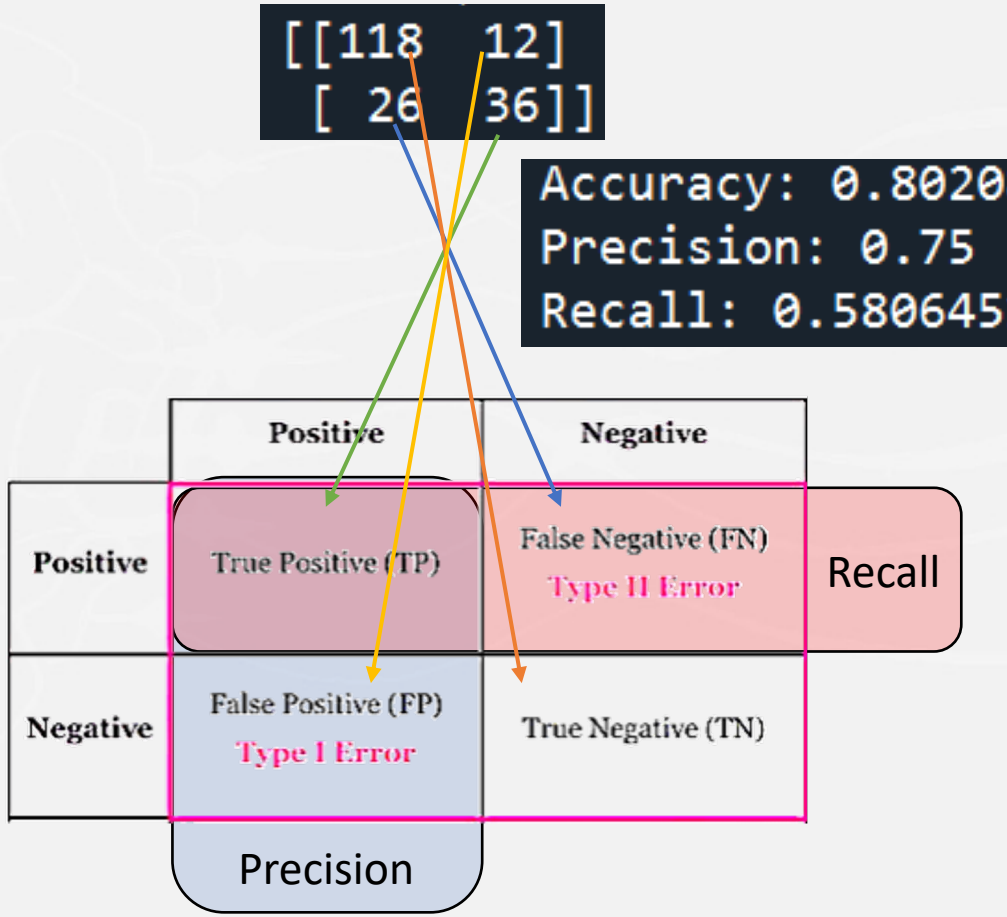
One Hot Encoding

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

2- Examples of machine learning applications

Classification - **Logistic regression**



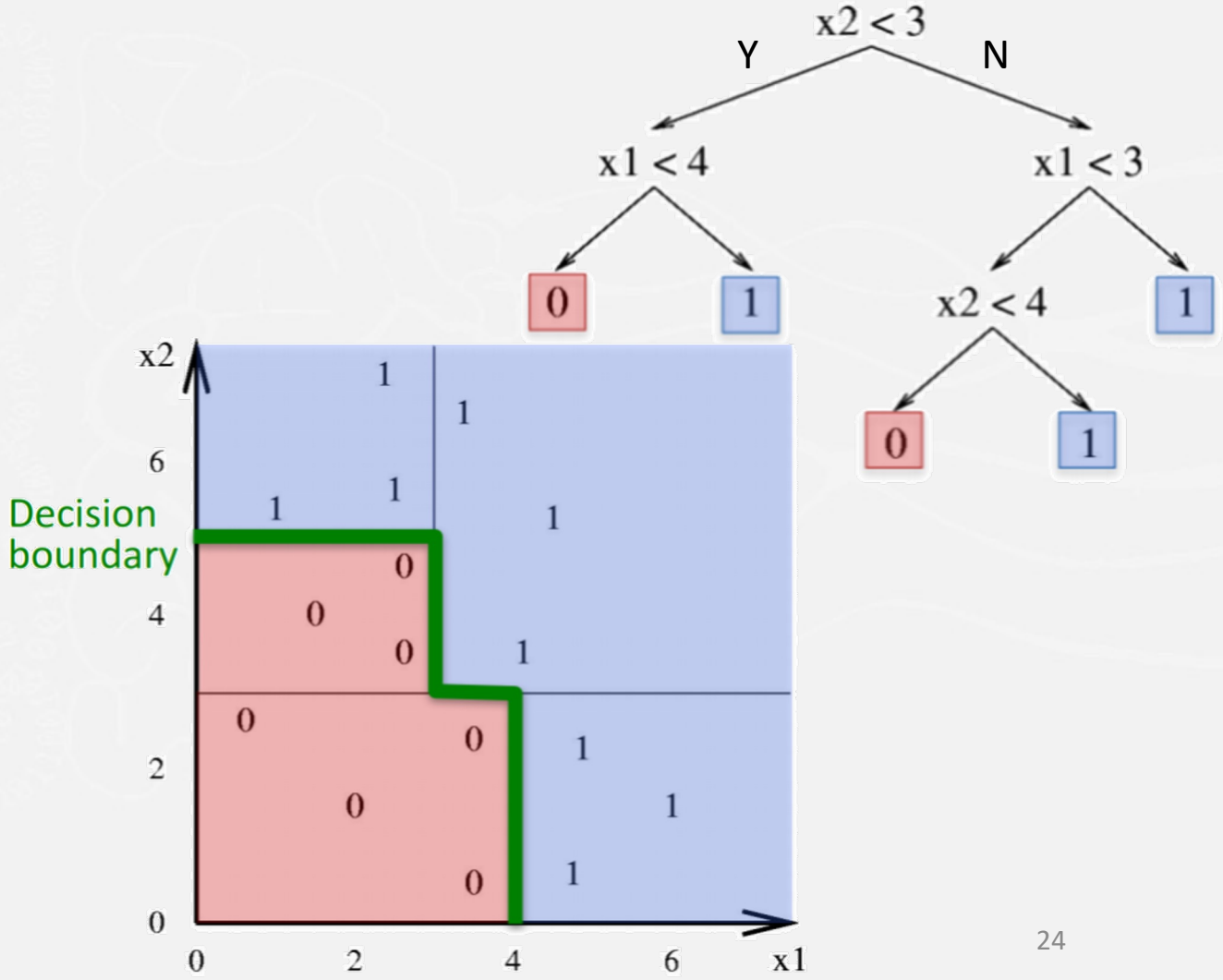


2- Examples of machine learning applications

Classification - **Decision tree**

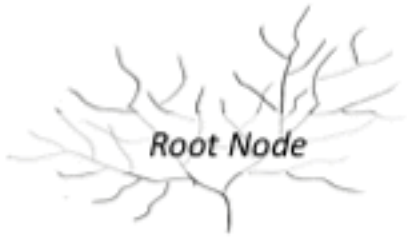
Divide the feature space into axis-parallel **(hyper)rectangles**

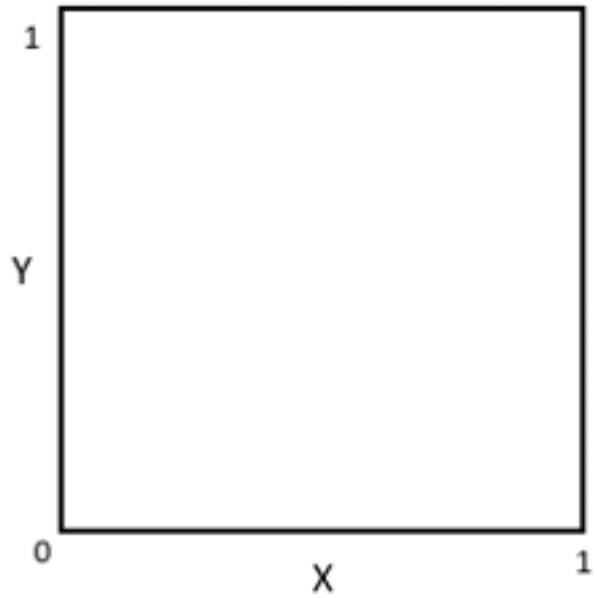
Each rectangular region is labeled with one label or a probability distribution over labels



2- Examples of machine learning applications

Classification - **Decision tree**



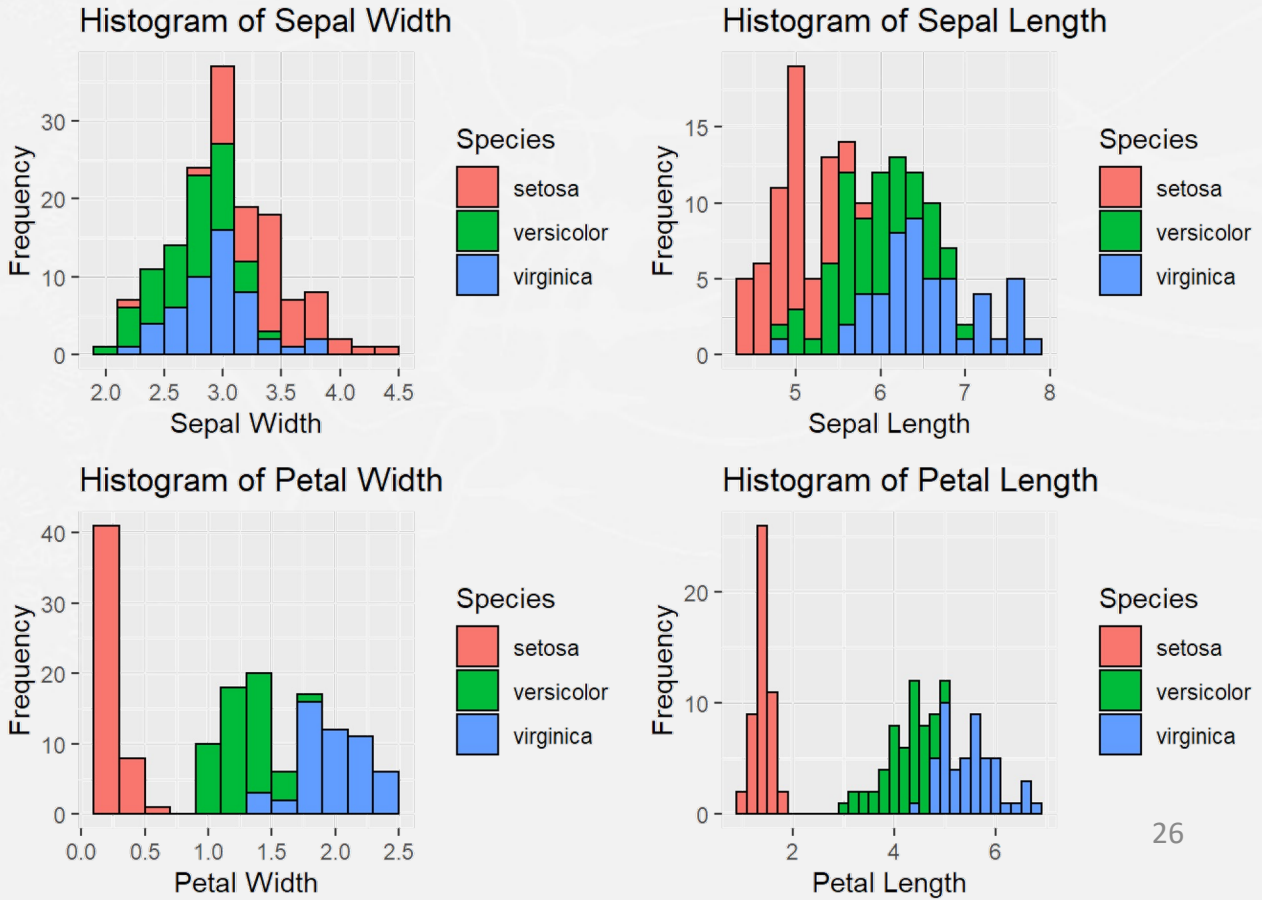


2- Examples of machine learning applications

Iris flower species prediction

Fisher's Iris Dataset

Iris Versicolor **Iris Setosa** **Iris Virginica**

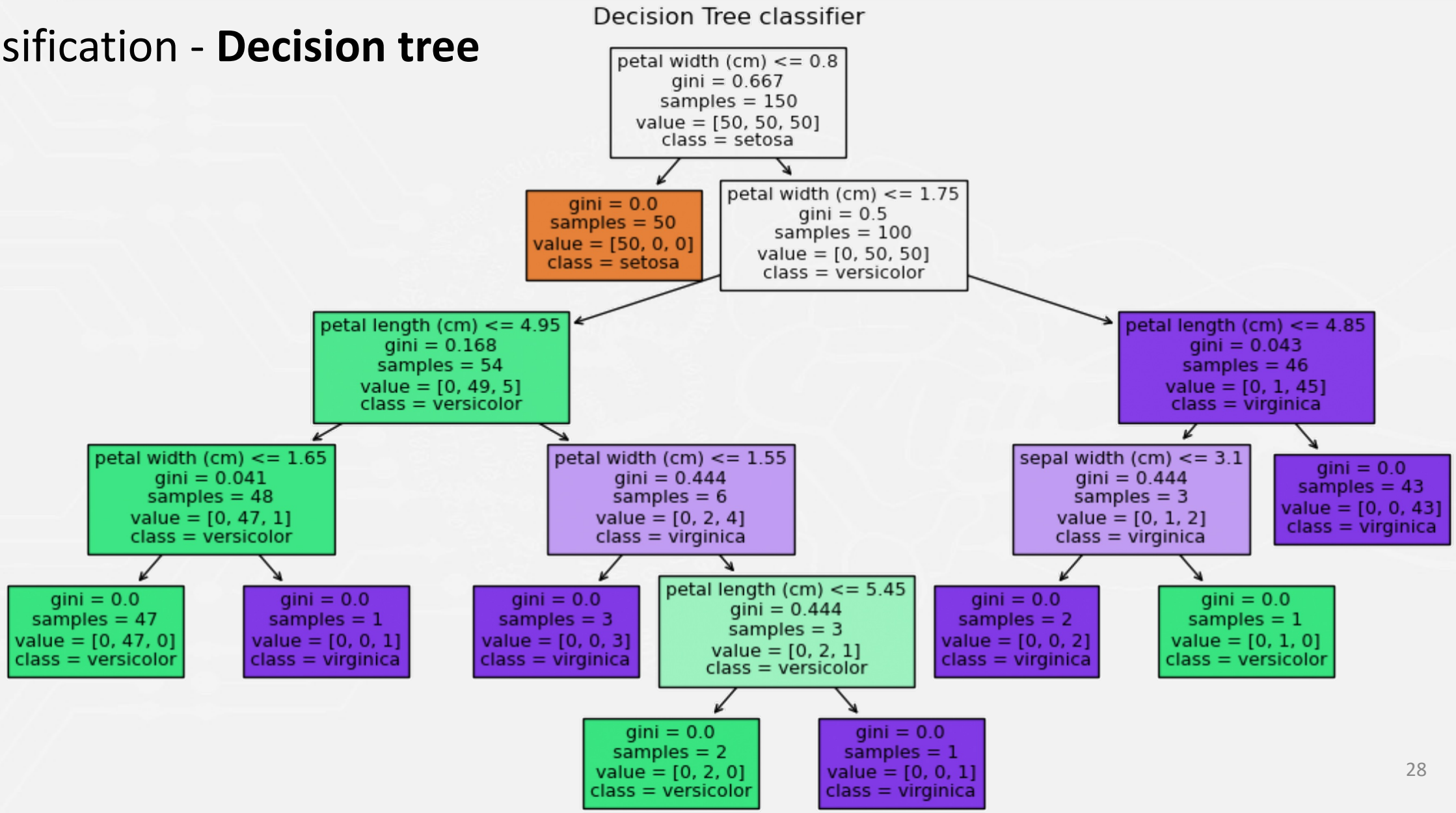


Classification - **Decision tree**

```
"""
Import libraries
"""
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import plot_tree
"""
Import the dataset
"""
iris = load_iris()
X = iris.data
y = iris.target
"""
Define the model
"""
clf = DecisionTreeClassifier()
"""
Train and validate the model using cross-validation
"""
cv_scores = cross_val_score(clf, X, y, cv=5)
print("Cross-validation scores:", cv_scores)
print("Mean CV accuracy:", np.mean(cv_scores))
"""
Train the model on the entire training set
"""
clf.fit(X, y)
"""
Plot the decision tree
"""
plt.figure(figsize=(15, 10))
plot_tree(clf, filled=True, feature_names=iris.feature_names,
          class_names=list(iris.target_names))
plt.title("Decision Tree classifier")
plt.show()
```

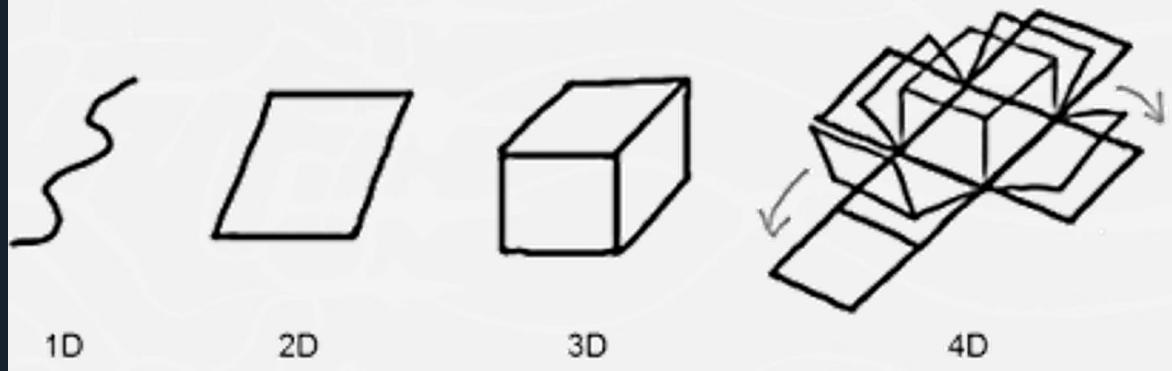

```
Cross-validation scores: [0.96666667 0.96666667 0.9        0.96666667 1.        ]
Mean CV accuracy: 0.9600000000000002
```

Classification - **Decision tree**

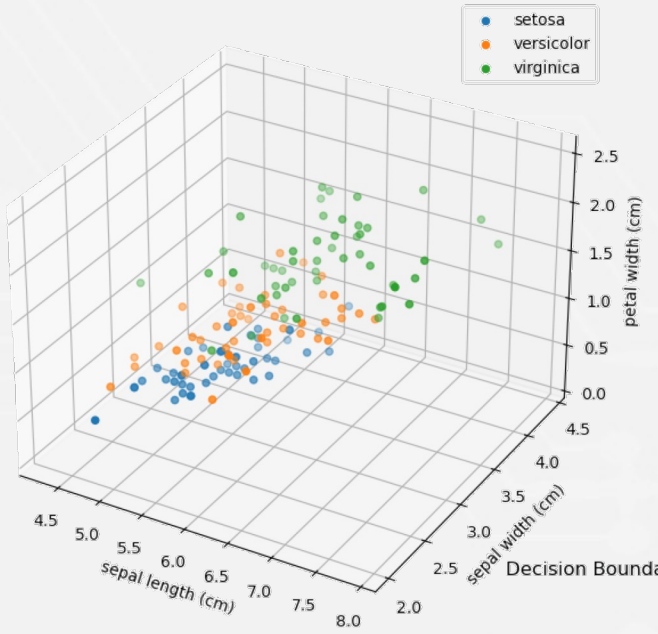


Classification - **Decision tree**

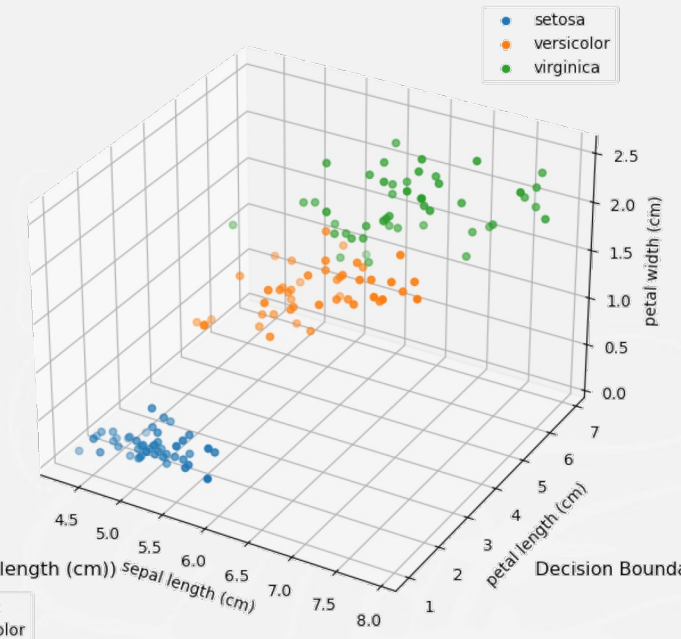
```
    """
    Sow the decision boundries by first defining the feature triplets for plotting
    """
    feature_triplets = [(0, 1, 2), (0, 1, 3), (0, 2, 3), (1, 2, 3)]
    triplet_names = [
        (iris.feature_names[i], iris.feature_names[j], iris.feature_names[k])
        for i, j, k in feature_triplets
    ]
    """
    Create separate 3D plots for each feature triplet
    """
    for i, (x_feature, y_feature, z_feature) in enumerate(triplet_names):
        fig = plt.figure(figsize=(10, 8))
        ax = fig.add_subplot(111, projection='3d')
        x_index, y_index, z_index = feature_triplets[i]
        X_triplet = X[:, [x_index, y_index, z_index]]
        #plot the data points
        for j, target_name in enumerate(iris.target_names):
            ax.scatter(
                X_triplet[y == j, 0],
                X_triplet[y == j, 1],
                X_triplet[y == j, 2],
                label=target_name
            )
        #set the axis labels
        ax.set_xlabel(x_feature)
        ax.set_ylabel(y_feature)
        ax.set_zlabel(z_feature)
        ax.set_title(f"Decision Boundaries in 3D ({x_feature}, {y_feature}, {z_feature})")
        plt.legend()
        #show the plot
        plt.show()
```



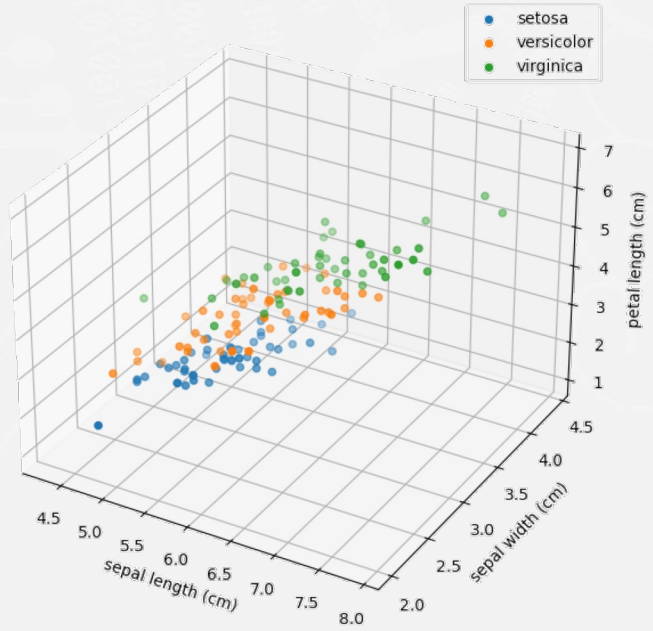
Decision Boundaries in 3D (sepal length (cm), sepal width (cm), petal width (cm))



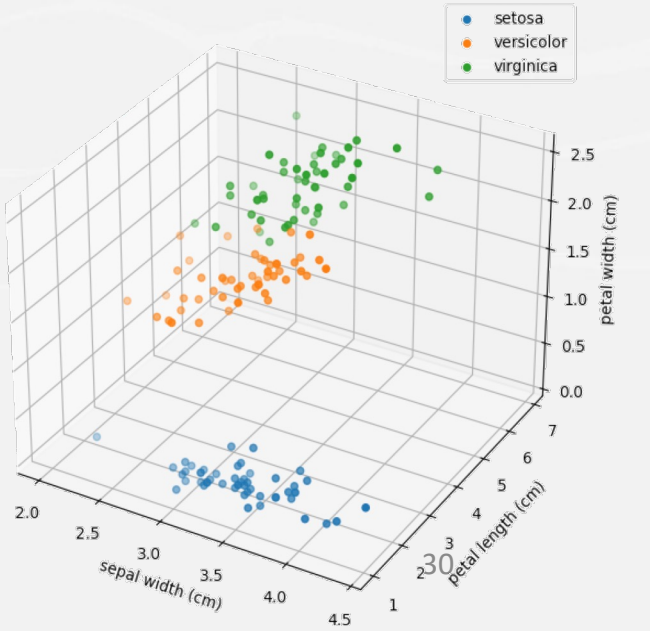
Decision Boundaries in 3D (sepal length (cm), petal length (cm), petal width (cm))



Decision Boundaries in 3D (sepal length (cm), sepal width (cm), petal length (cm))



Decision Boundaries in 3D (sepal width (cm), petal length (cm), petal width (cm))



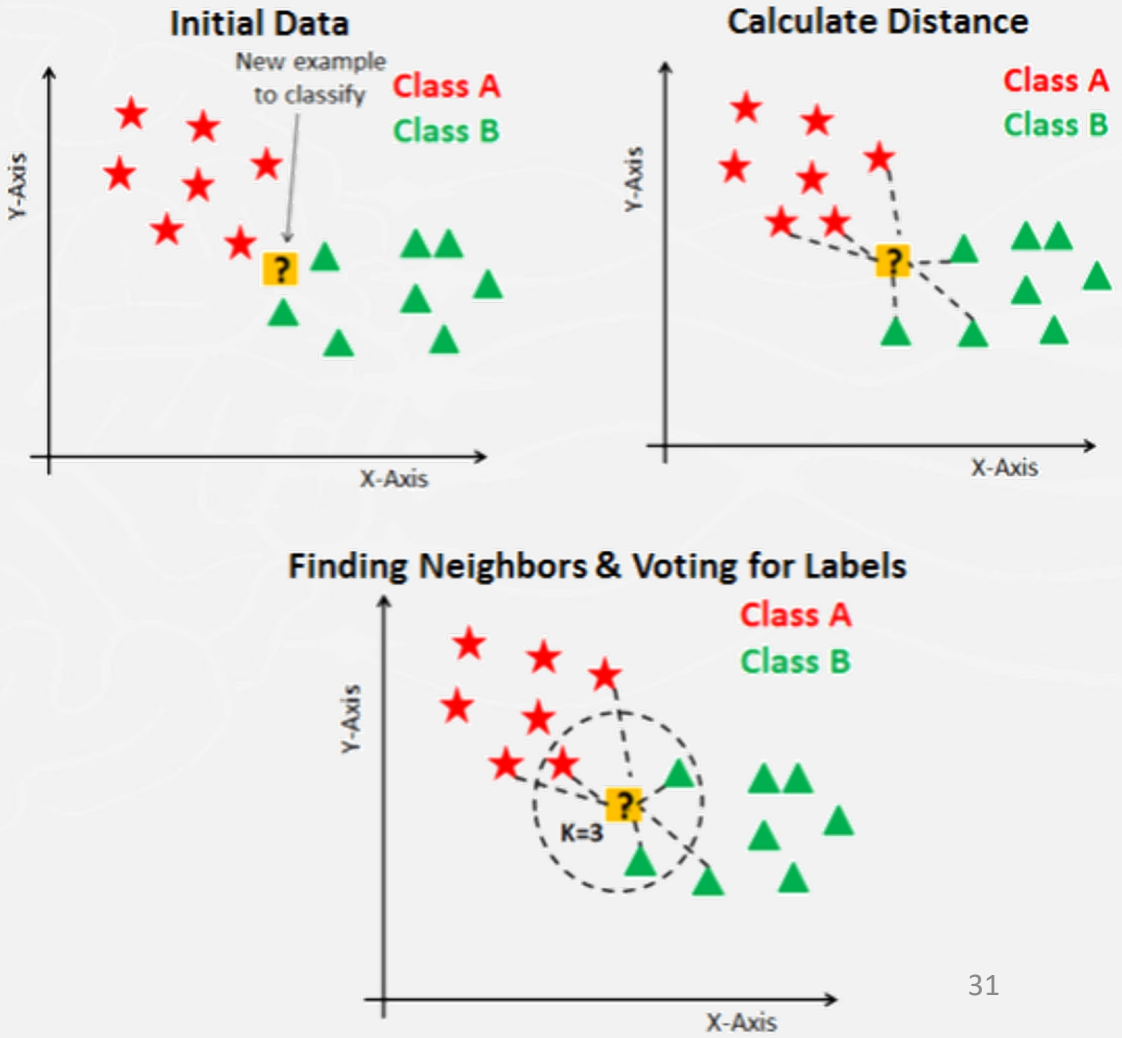
2- Examples of machine learning applications

Classification - **K Nearest Neighbor (kNN)**

Calculates the distance of a new data point to all other training data points.

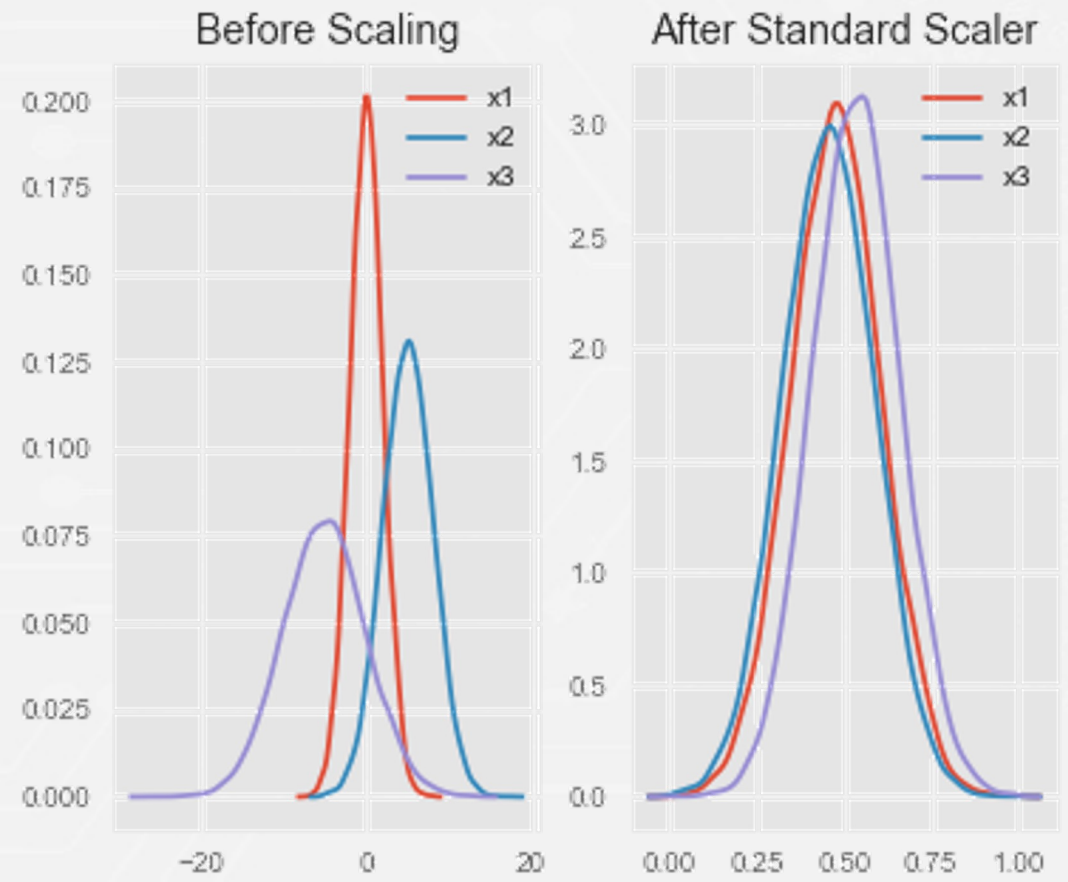
Selects the K-nearest data points.

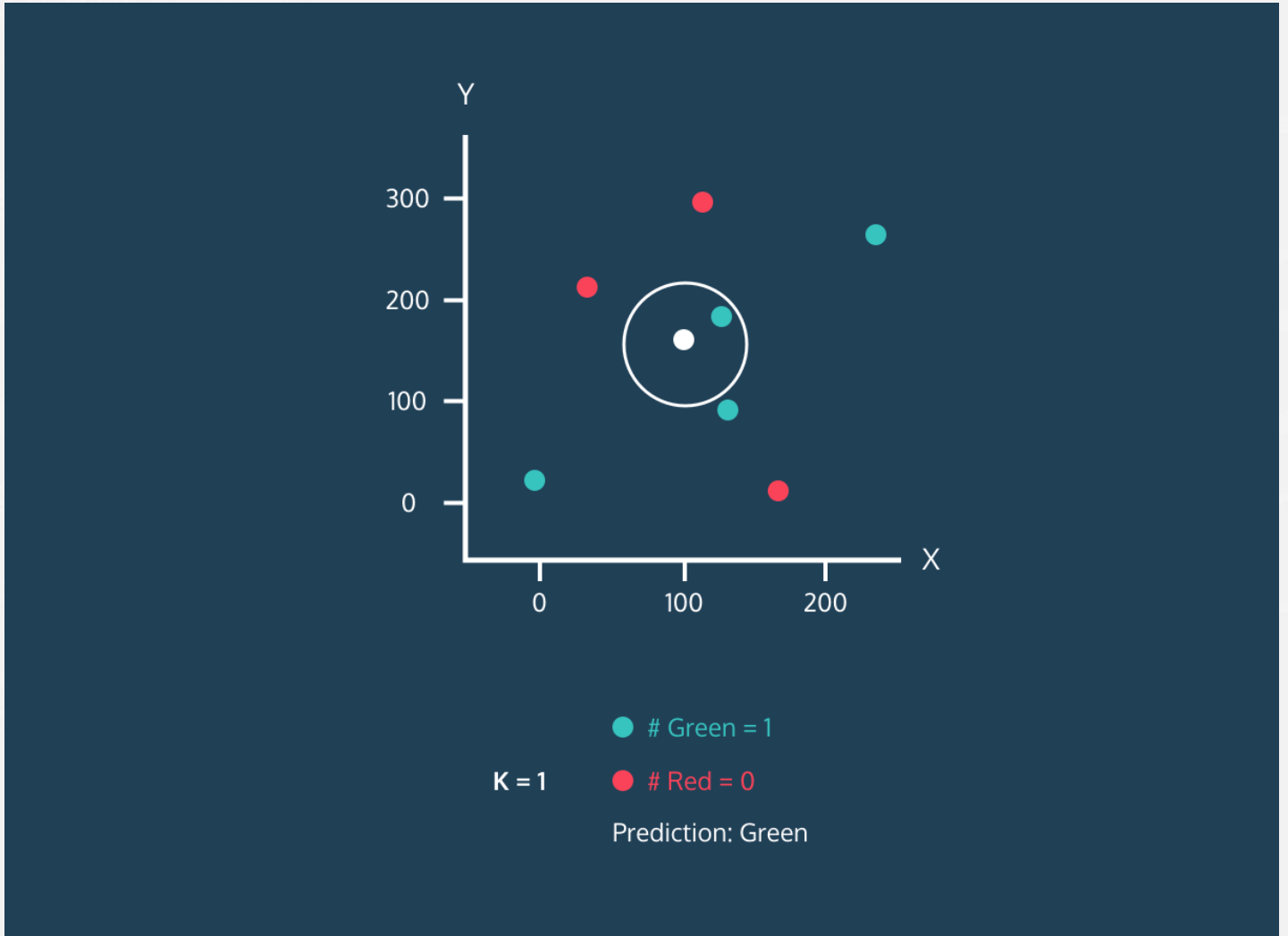
Assigns the data point to the class to which the majority of the K data points belong.



2- Examples of machine learning applications

Classification - **kNN**





2- Examples of machine learning applications

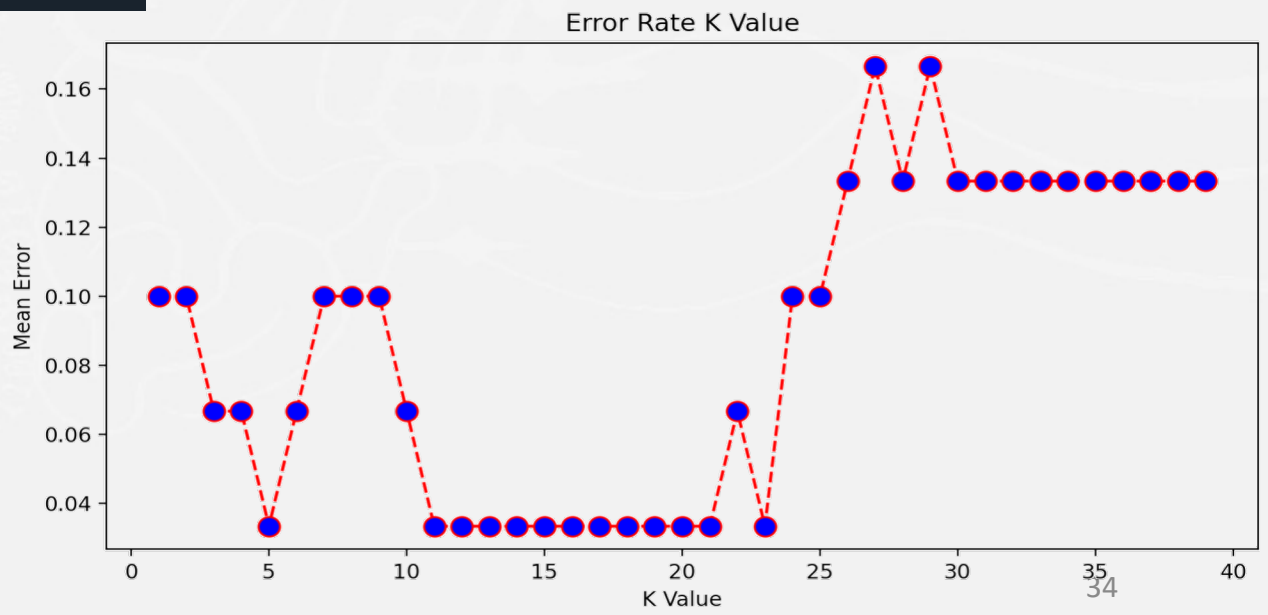
Classification - **kNN**

```
"""
Importing libraries
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
"""
Importing the dataset
"""
url="https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
# Assign colum names to the dataset
names=['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
# Read dataset to pandas dataframe
dataset = pd.read_csv(url, names=names)
# Check the data
dataset.head()
"""
Preprocessing
"""
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
"""
Create training and test splits,
"""
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

```
"""
Feature scaling
"""
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
"""
Training and predictions
"""
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
"""
Evaluating the algorithm
"""
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
48  """
49  Comparing error rate with the K value
50  """
51  error = []
52  # Calculating error for K values between 1 and 40
53  for i in range(1, 40):
54      knn = KNeighborsClassifier(n_neighbors=i)
55      knn.fit(X_train, y_train)
56      pred_i = knn.predict(X_test)
57      error.append(np.mean(pred_i != y_test))
58  plt.figure(figsize=(12, 6))
59  plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
60           markerfacecolor='blue', markersize=10)
61  plt.title('Error Rate K Value')
62  plt.xlabel('K Value')
63  plt.ylabel('Mean Error')
```

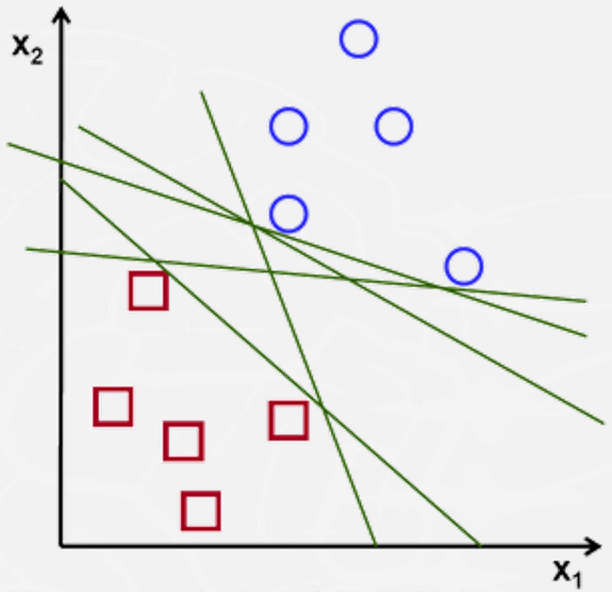
```
[[ 8  0  0]
 [ 0 10  0]
 [ 0  1 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       0.91      1.00      0.95        10
 Iris-virginica       1.00      0.92      0.96        12

       accuracy                           0.97        30
      macro avg       0.97      0.97      0.97        30
   weighted avg       0.97      0.97      0.97        30
```

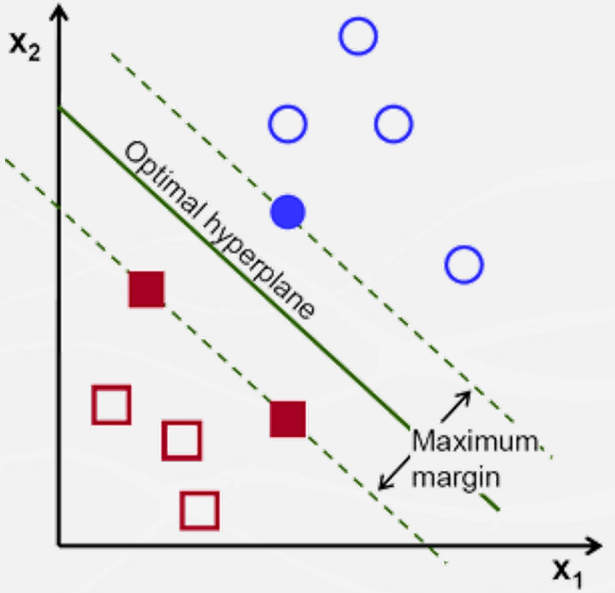


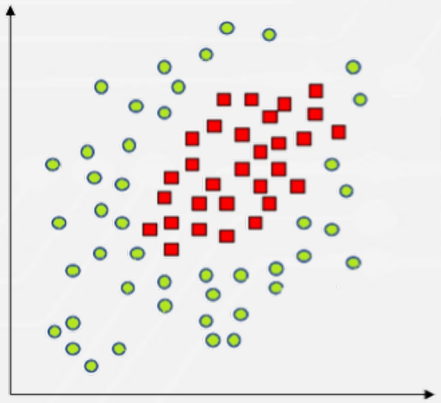
2- Examples of machine learning applications

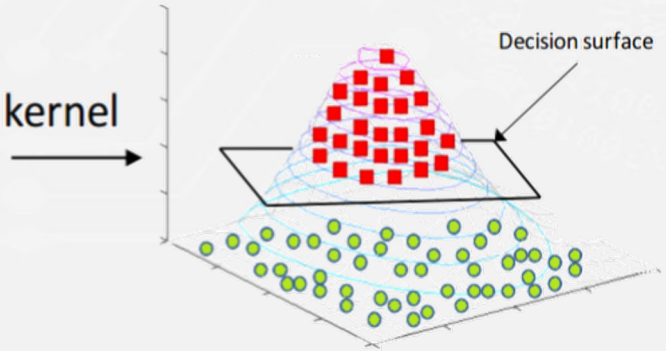
Classification - **Support Vector Machine (SVM)**

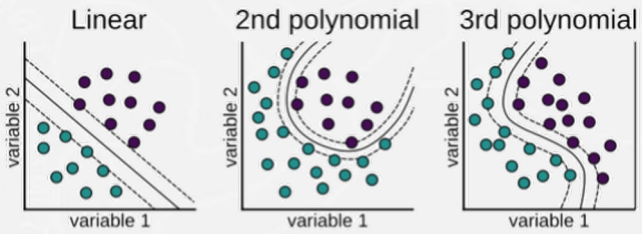
Find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

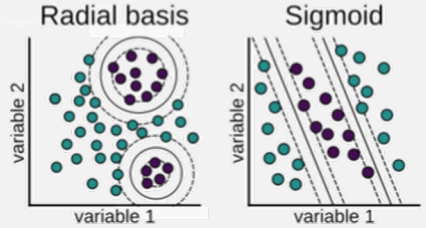






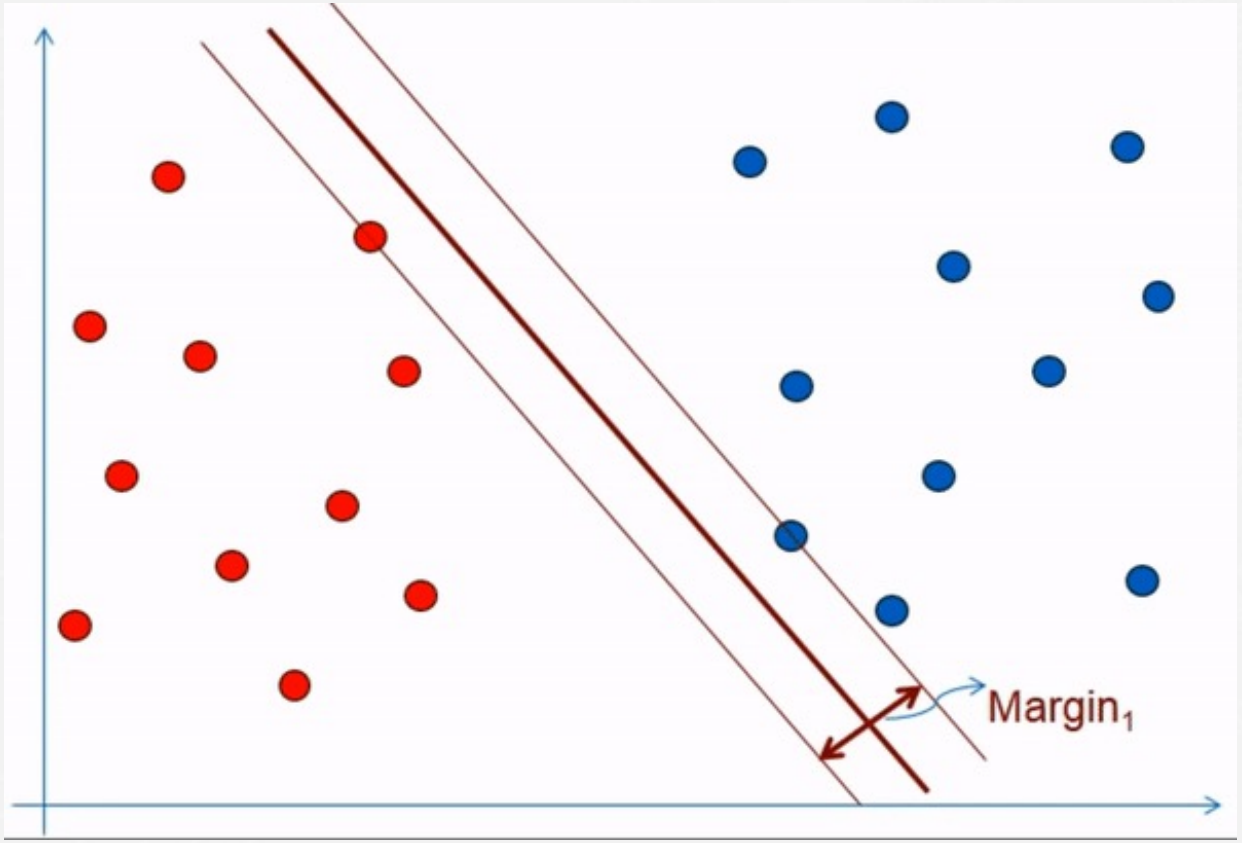


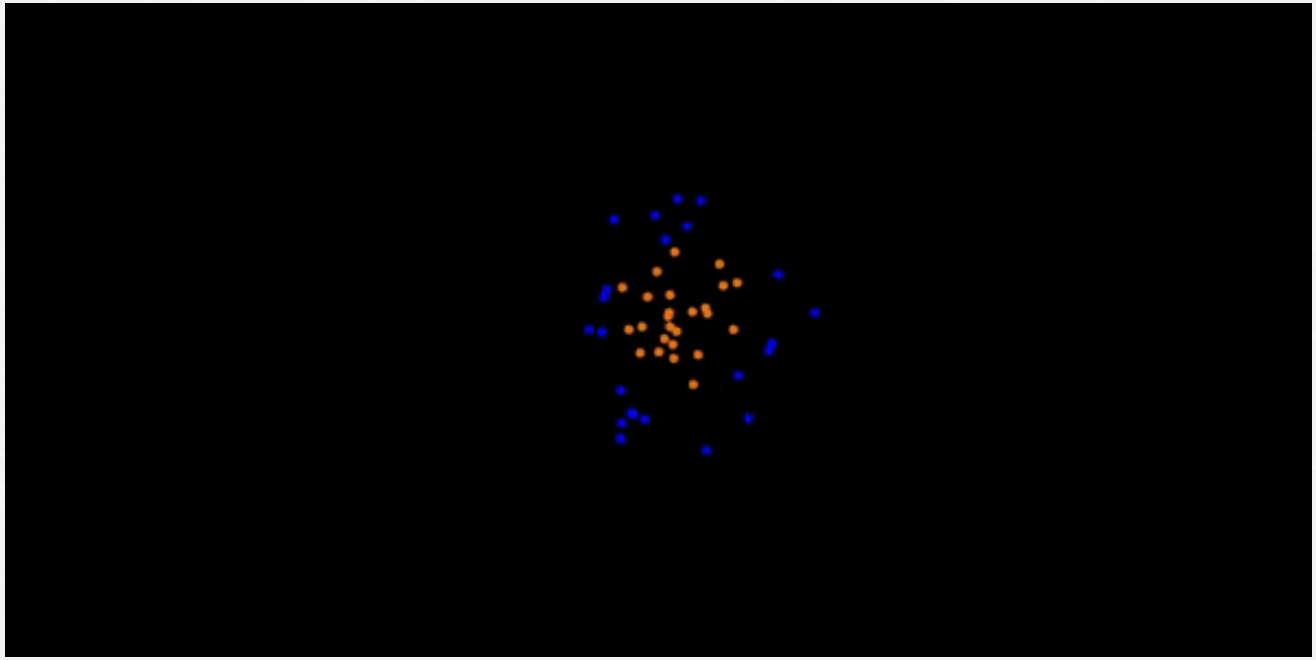




2- Examples of machine learning applications

Classification - **SVM**





Classification - **SVM**

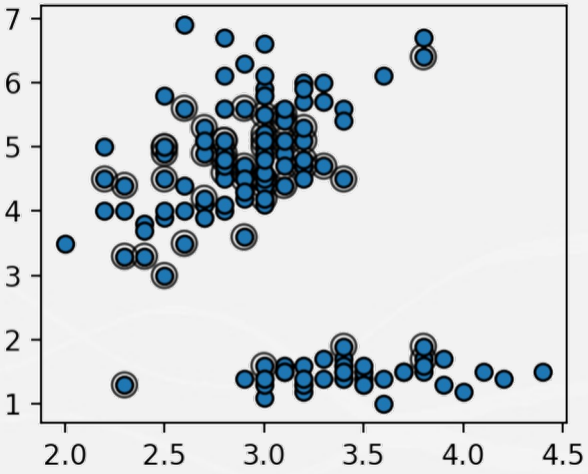
```
"""
Importing libraries
"""
import pandas as pd
"""
Importing the dataset
"""
url="https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
# Assign colum names to the dataset
colnames=['sepal-length','sepal-width','petal-length','petal-width','Class']
# Read dataset to pandas dataframe
irisdata = pd.read_csv(url, names=colnames)
"""
Preprocessing
"""
X = irisdata.drop('Class', axis=1)
y = irisdata['Class']
"""
Train Test Split
"""
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

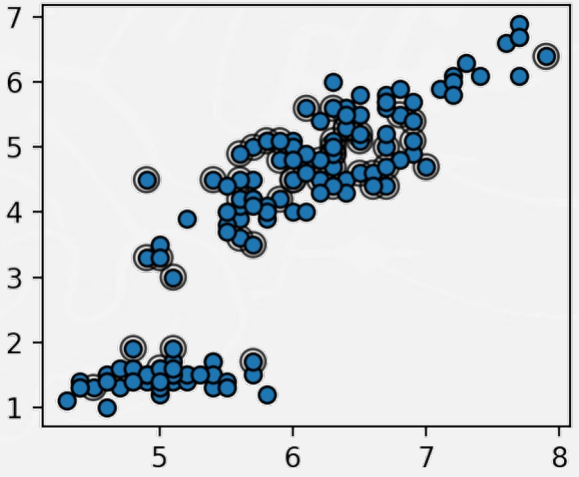
```
"""
Training the algorithm with Gaussian Kernel
"""
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
"""
Prediction and evaluation
"""
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

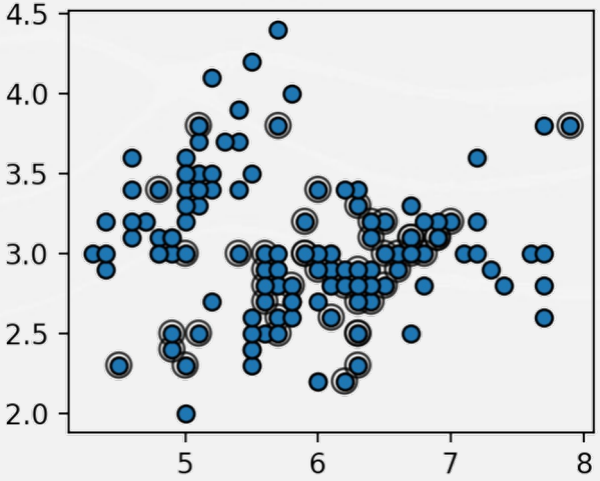
```
[[18  0  0]
 [ 0  6  0]
 [ 0  0 21]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        18
Iris-versicolor       1.00      1.00      1.00         6
 Iris-virginica       1.00      1.00      1.00        21

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45
```

Examples of support vectors for three of the four features

```
36      """
37      Plot the suport vectors
38      """
39      import matplotlib.pyplot as plt
40      plt.figure(0)
41      plt.scatter(svclassifier.support_vectors_[:, 0],
42                  svclassifier.support_vectors_[:, 1], s=80,
43                  facecolors='none', edgecolors='k',cmap=plt.cm.coolwarm, alpha=0.8)
44      plt.scatter(X.to_numpy()[:, 0], X.to_numpy()[:, 1],  cmap=plt.cm.coolwarm,
45                  edgecolors='k')
46      plt.figure(1)
47      plt.scatter(svclassifier.support_vectors_[:, 0],
48                  svclassifier.support_vectors_[:, 2], s=80,
49                  facecolors='none', edgecolors='k',cmap=plt.cm.coolwarm, alpha=0.8)
50      plt.scatter(X.to_numpy()[:, 0], X.to_numpy()[:, 2],  cmap=plt.cm.coolwarm,
51                  edgecolors='k')
52      plt.figure(2)
53      plt.scatter(svclassifier.support_vectors_[:, 1],
54                  svclassifier.support_vectors_[:, 2], s=80,
55                  facecolors='none', edgecolors='k',cmap=plt.cm.coolwarm, alpha=0.8)
56      plt.scatter(X.to_numpy()[:, 1], X.to_numpy()[:, 2],  cmap=plt.cm.coolwarm,
57                  edgecolors='k')
```

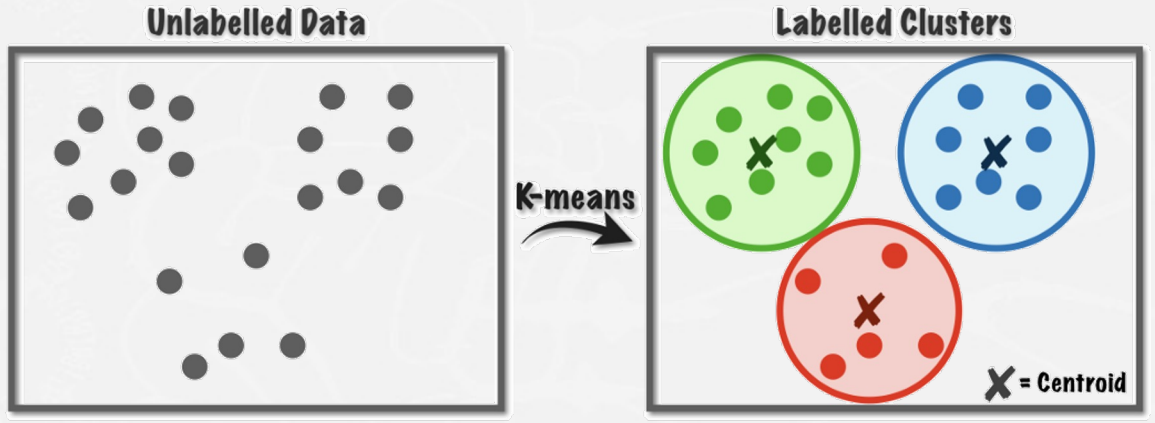




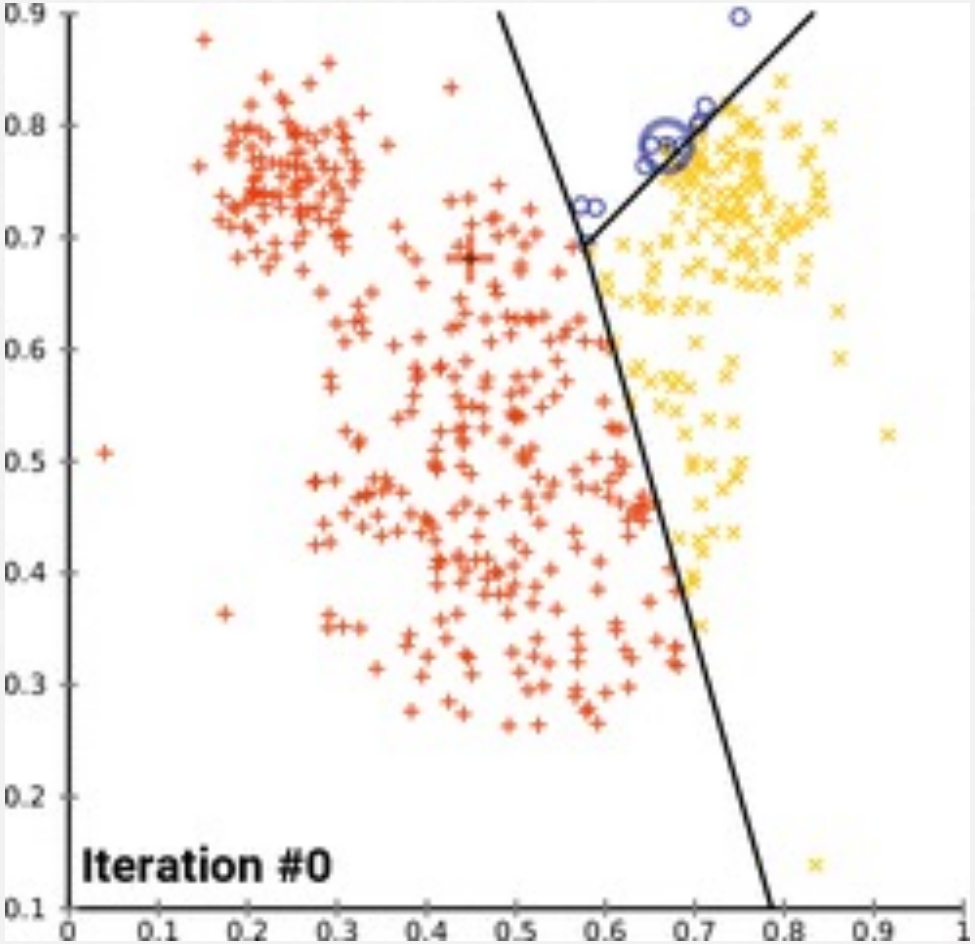


Clustering - **K-means**

Identifies K number of **centroids**, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.



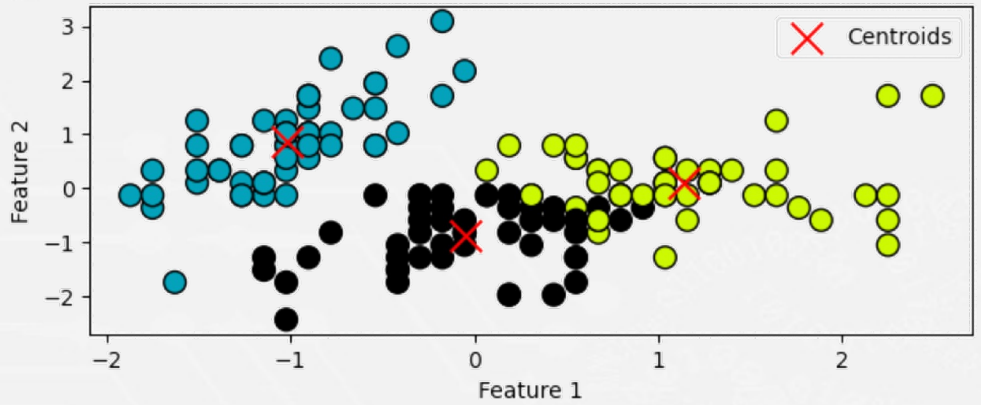
Clustering - **K-means**

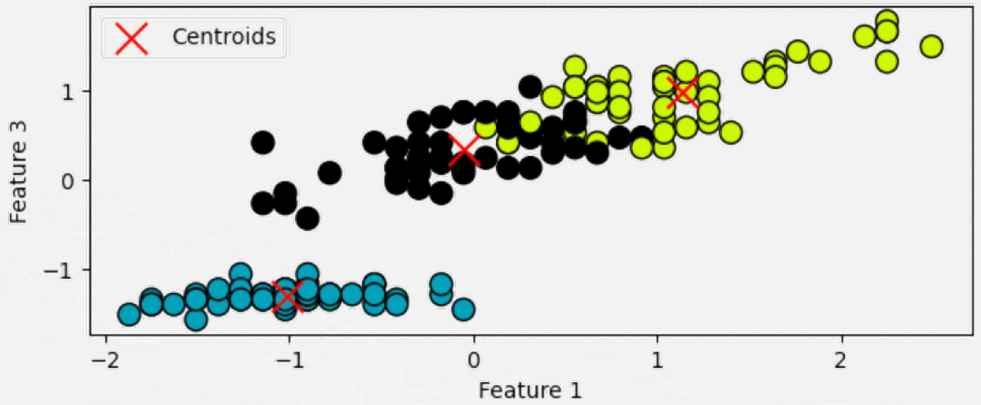


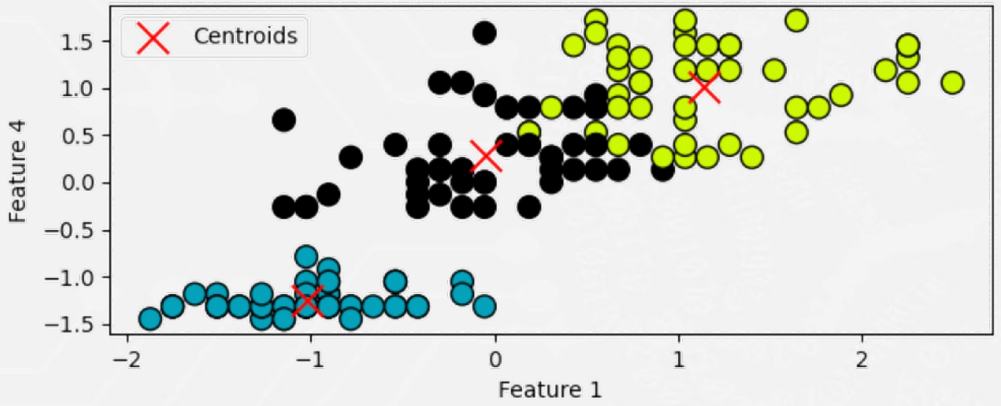
Clustering - **K-means**

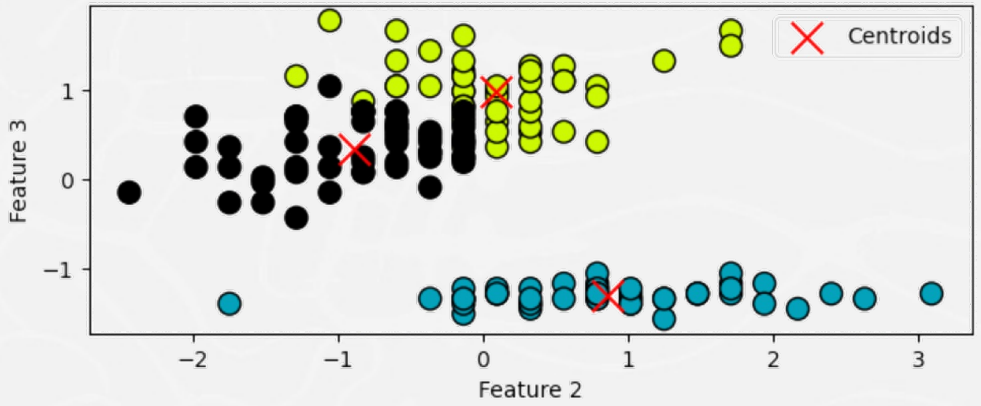
```python
"""
Import libraries
"""
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.preprocessing import StandardScaler
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm
from itertools import combinations
"""
Import the dataset
"""
iris = load_iris()
X = iris.data
"""
Standardize the features
"""
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
"""
Set the number of clusters
"""
n_clusters = 3
"""
Perform k-means clustering
"""
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
cluster_labels = kmeans.fit_predict(X_scaled)
```

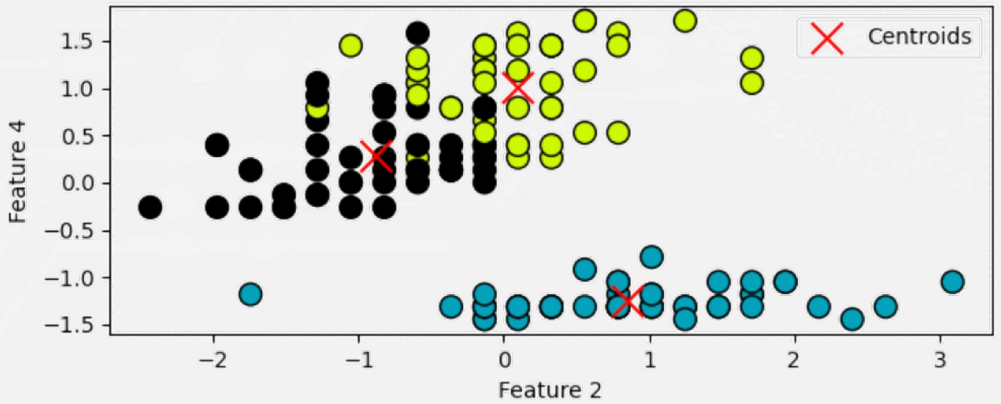
```python
"""
Get silhouette scores for each sample
"""
silhouette_avg = silhouette_score(X_scaled, cluster_labels)
"""
Plot the clustering regions and the centroids for all 2D combinations
"""
#get all combinations of 2 features from the dataset
feature_combinations = list(combinations(range(X_scaled.shape[1]), 2))
#create subplots for all feature combinations
fig, axes = plt.subplots(len(feature_combinations) // 2, 2, figsize=(8, 8))
axes = axes.ravel()
for i, (feature1, feature2) in enumerate(feature_combinations):
    ax = axes[i]
    #plot the clustering regions in 2D for the current feature pair
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax.scatter(
        X_scaled[:, feature1], X_scaled[:, feature2],
        marker='o', s=100, c=colors, edgecolor='k'
    )
    #plot the centroids of each cluster
    cluster_centers = kmeans.cluster_centers_
    ax.scatter(
        cluster_centers[:, feature1], cluster_centers[:, feature2],
        marker='x', s=200, c='red', label='Centroids'
    )
    ax.set_xlabel(f'Feature {feature1 + 1}')
    ax.set_ylabel(f'Feature {feature2 + 1}')
    ax.legend()
#adjust and show the subplots
plt.tight_layout()
plt.show()
```

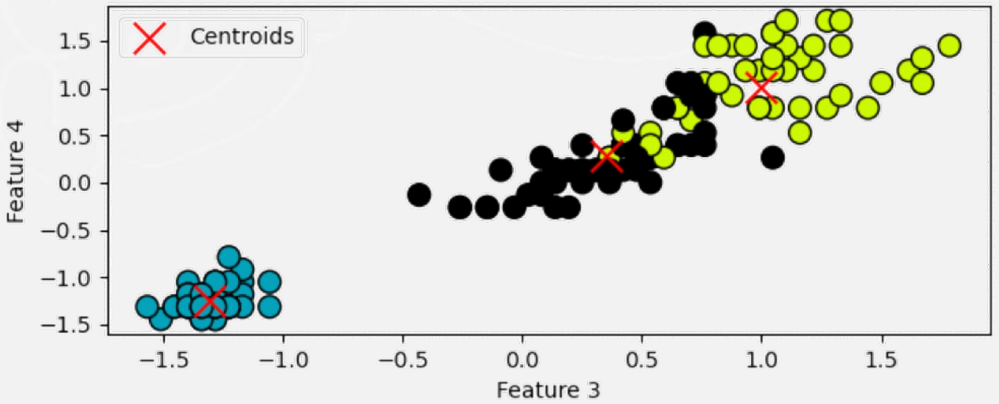

Centroids
Feature 2
Feature 1


Centroids
Feature 3
Feature 1

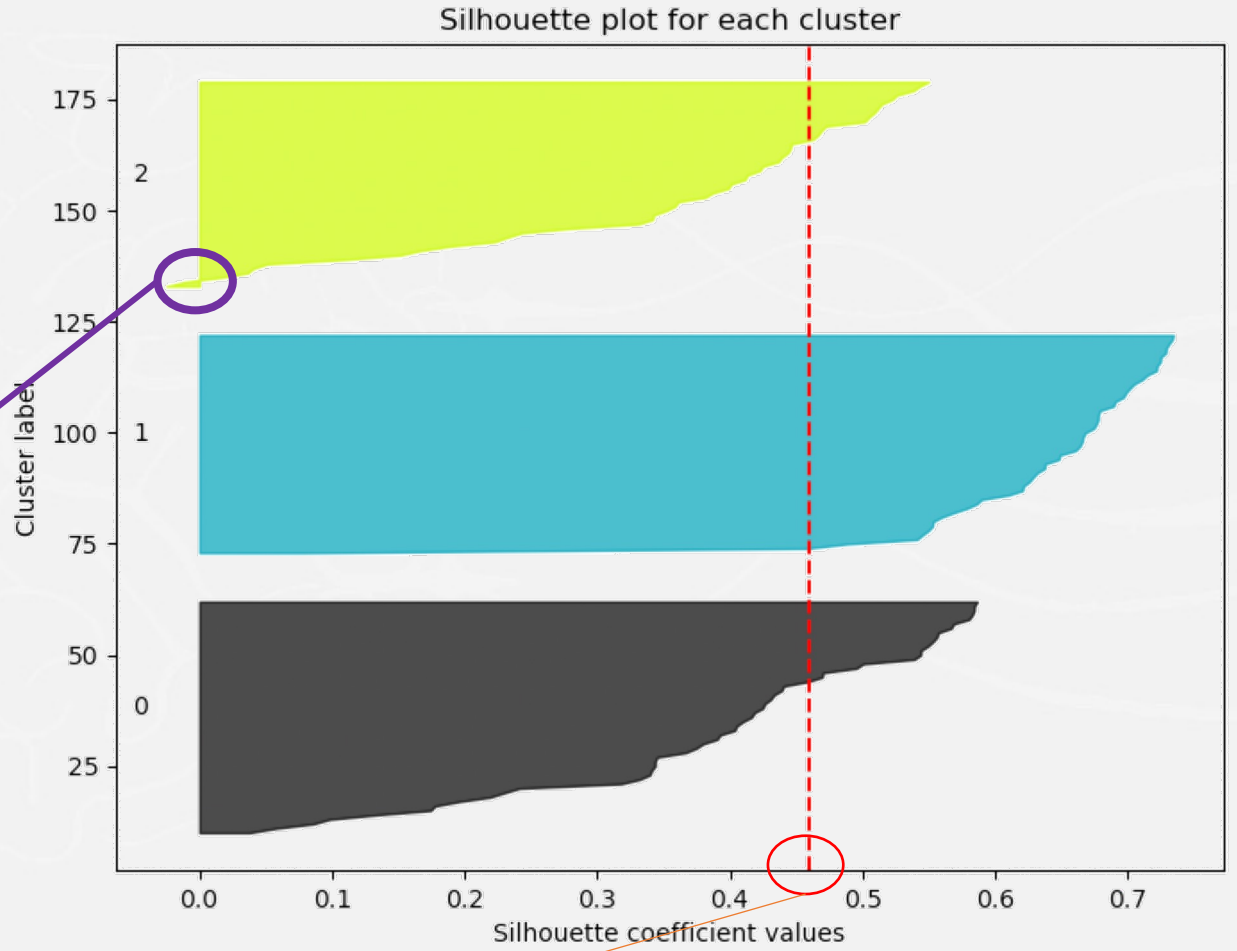

Centroids
Feature 4
Feature 1


Centroids
Feature 3
Feature 2


Centroids
Feature 4
Feature 2

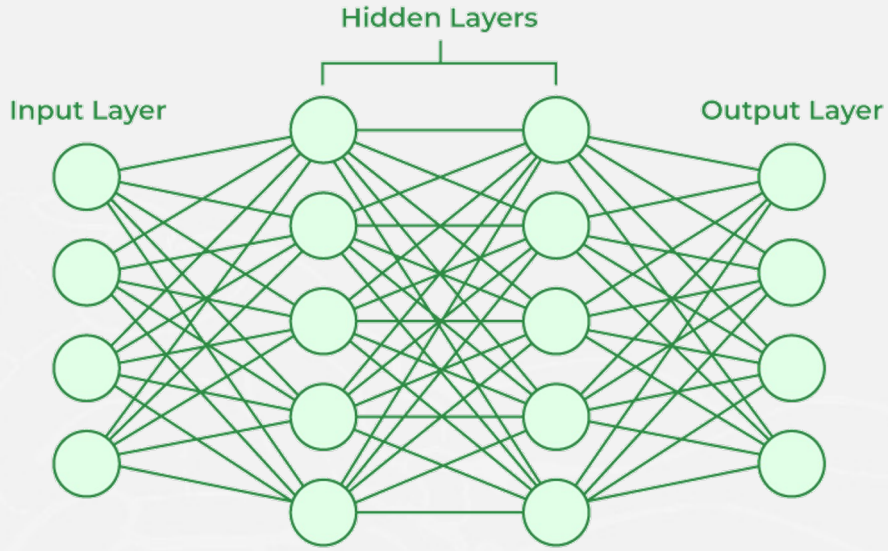

Centroids
Feature 4
Feature 3

```
    """
    Plot the silhouette scores for each sample
    """
    plt.figure(figsize=(8, 6))
    sample_silhouette_values = silhouette_samples(X_scaled, cluster_labels)
    y_lower = 10
    for i in range(n_clusters):
        ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]
        ith_cluster_silhouette_values.sort()
        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i
        color = cm.nipy_spectral(float(i) / n_clusters)
        plt.fill_betweenx(np.arange(y_lower, y_upper), 0,
                          ith_cluster_silhouette_values, facecolor=color,
                          edgecolor=color, alpha=0.7)
        plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
        y_lower = y_upper + 10
    plt.axvline(x=silhouette_avg, color="red", linestyle="--")
    plt.title("Silhouette plot for each cluster")
    plt.xlabel("Silhouette coefficient values")
    plt.ylabel("Cluster label")
    plt.show()
```



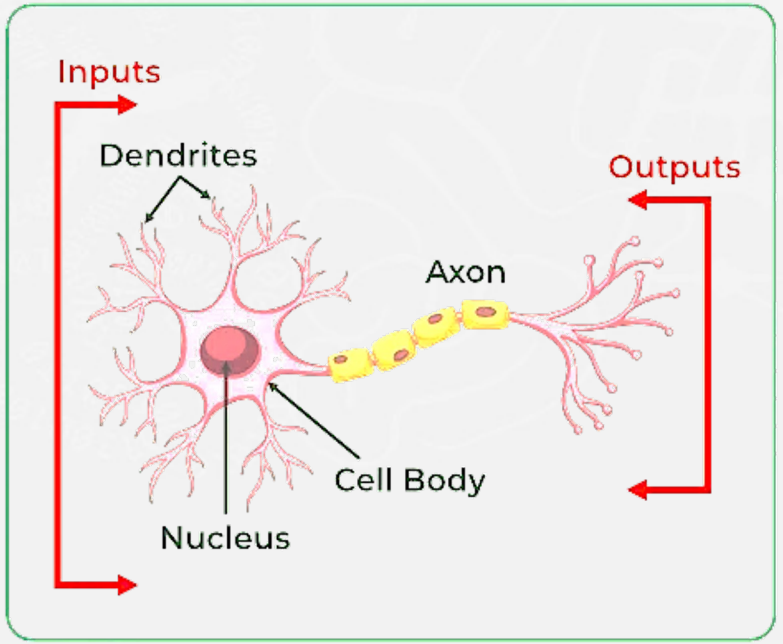

```
silhouette_avg
0.45994823920518635
```

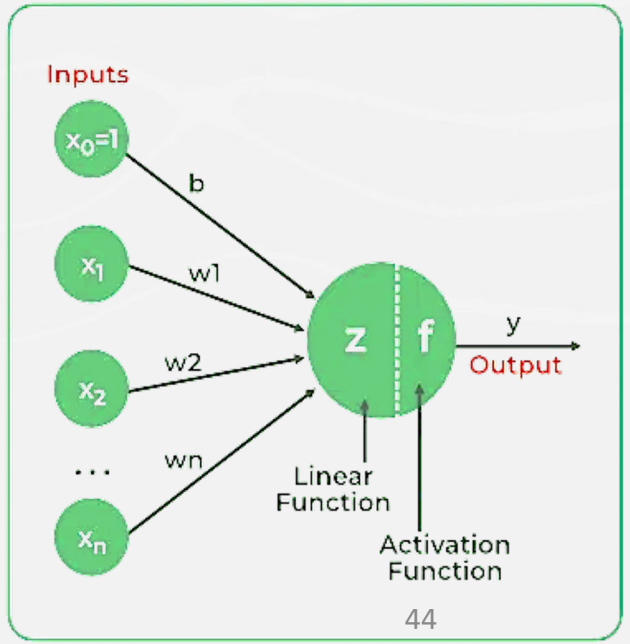
Regression - **Artificial neural network (ANN)**



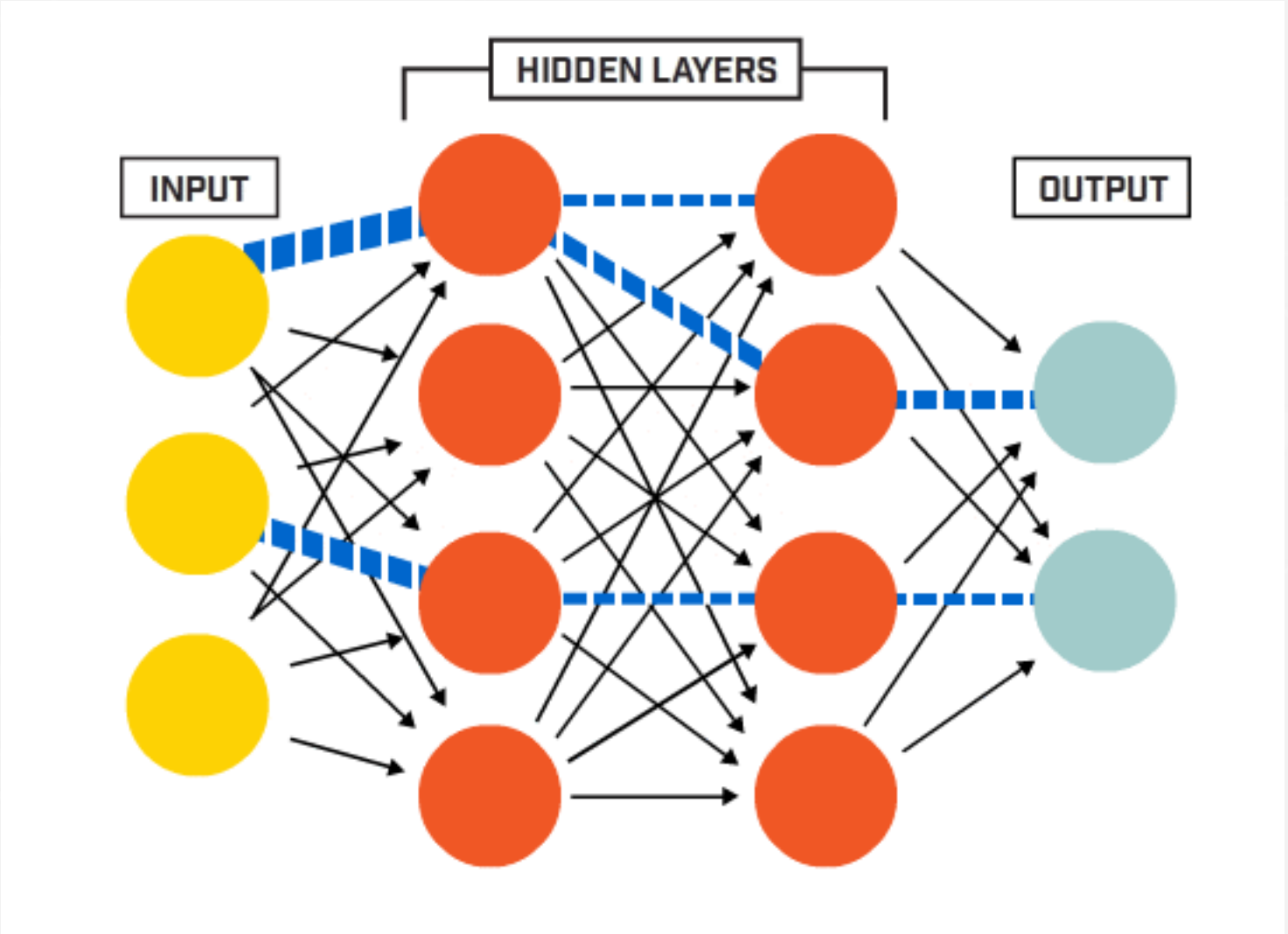
Comprised of a node layers, where each node connects to another and has an associated weight and threshold.

Optimize the weights by using corrective feedback.





Regressions - **ANN**



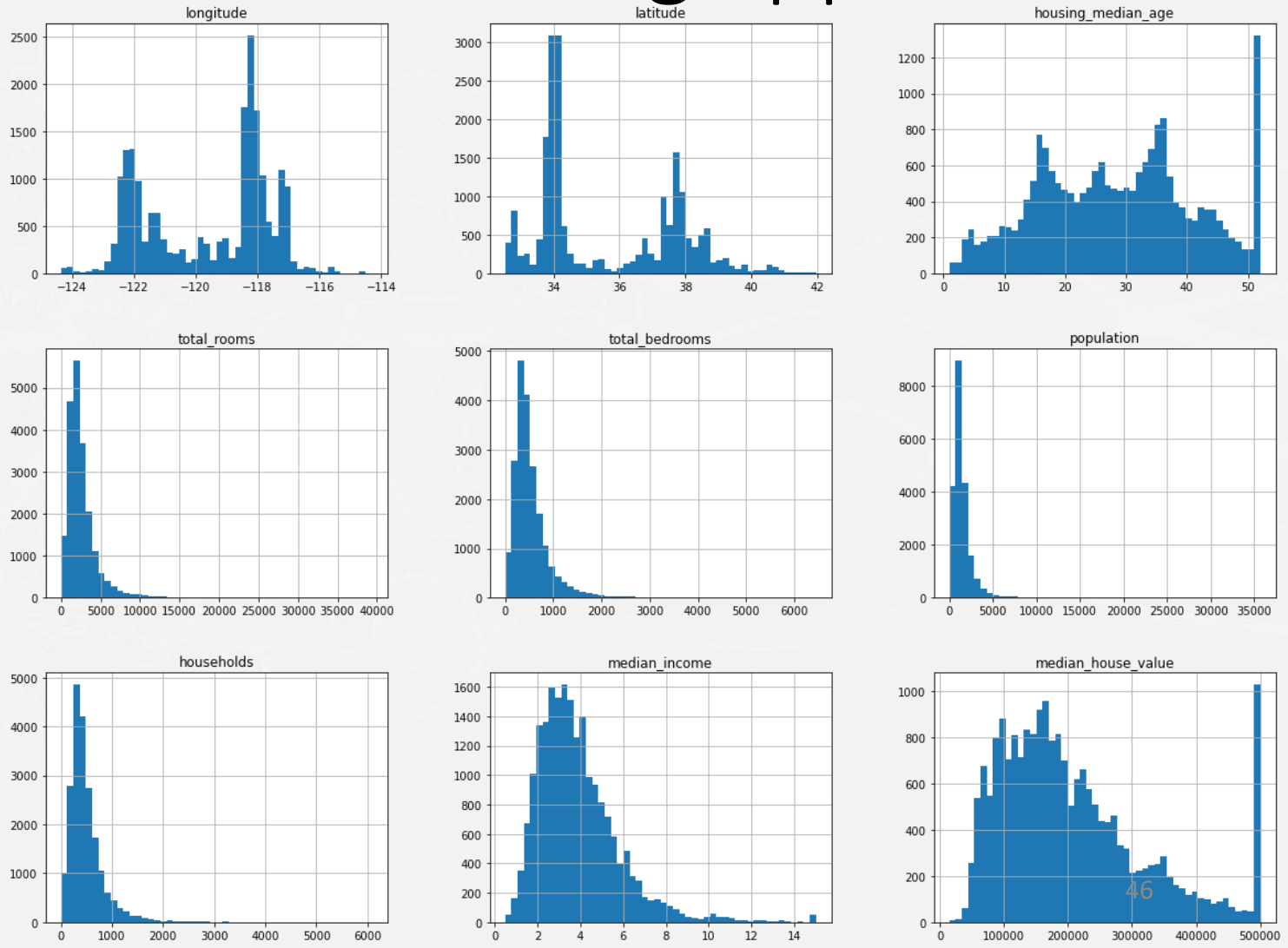
2- Examples of machine learning applications

House price prediction

California housing Dataset

Target variable – median housing value

Features:
- Median income in block
- Median house age in block
- Average number of rooms per house
- Average number of bedrooms per house
- Population of block
- Average number of household members per house
- Latitude of block
- Longitude of block



2- Examples of machine learning applications

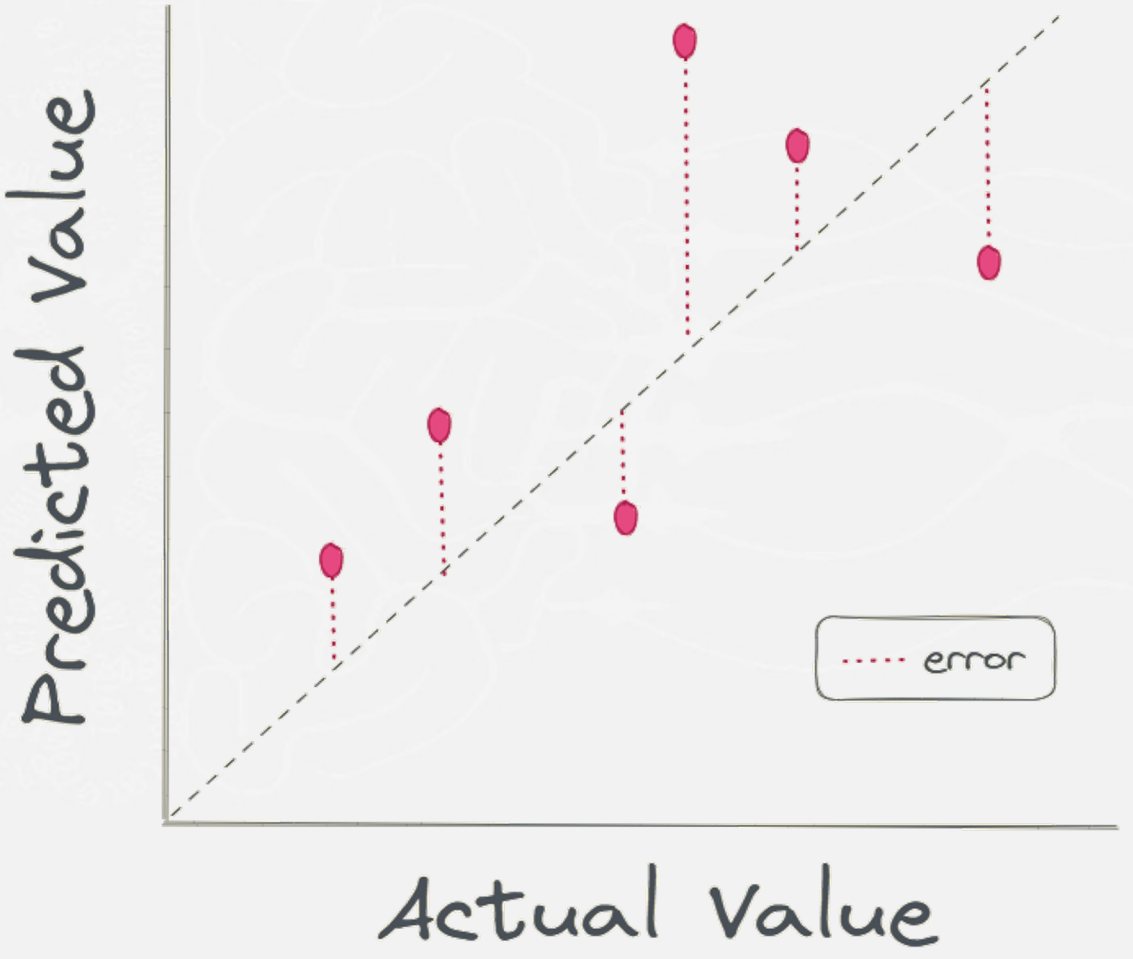
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$



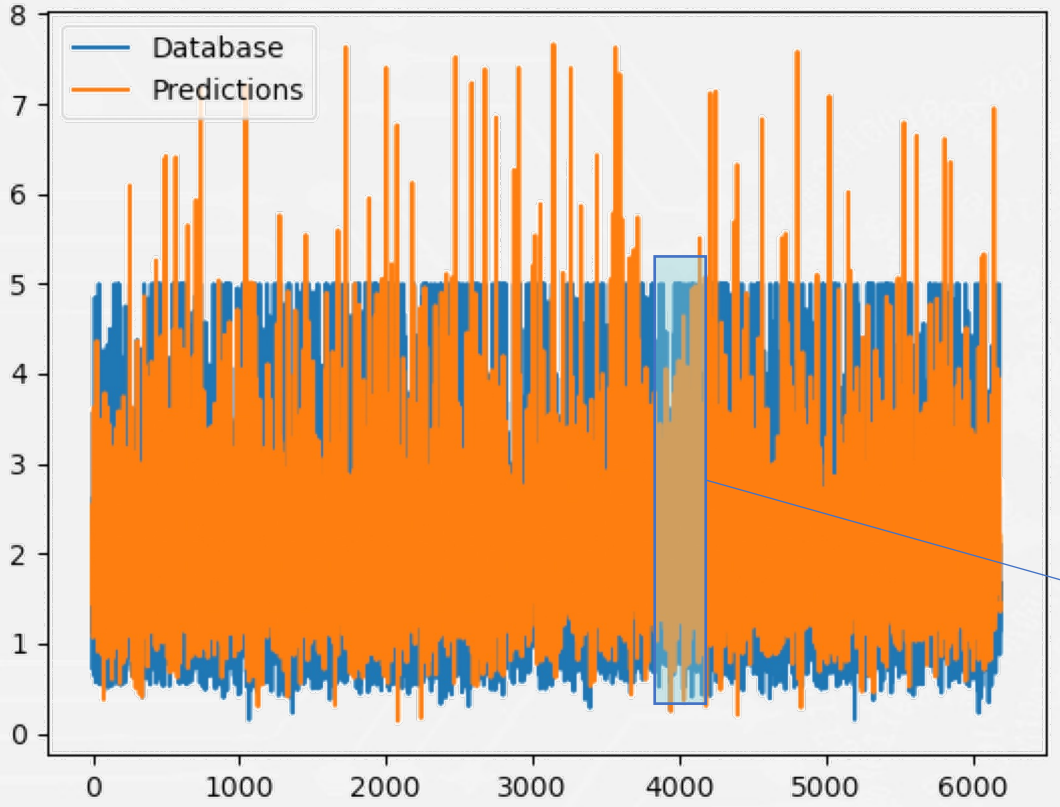
Regression - **ANN**

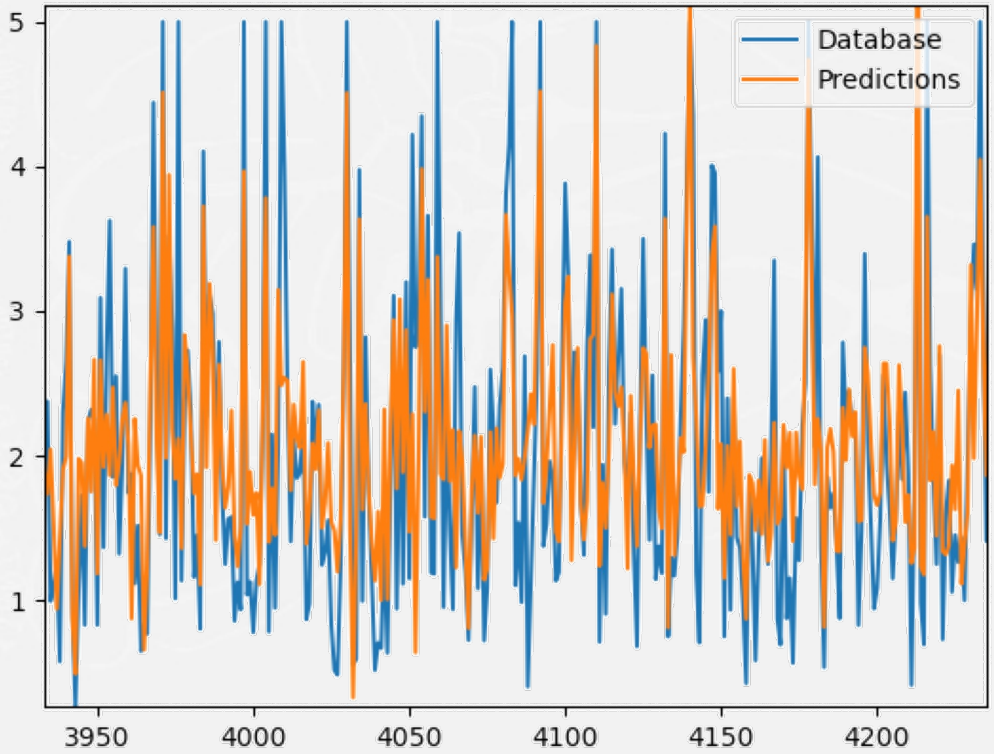
```python
"""
Import libraries
"""
from sklearn.neural_network import MLPRegressor
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_california_housing
import numpy as np
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
"""
Import the dataset
"""
housing = fetch_california_housing()
X = housing.data
y = housing.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
"""
Define the model
"""
nn = MLPRegressor(
    activation='relu',
    hidden_layer_sizes=(1000),
    alpha=0.01,
    early_stopping=True
)
"""
Train the model
"""
nn.fit(X_train, y_train)
```

```python
"""
Make predictions and estimate the error
"""
pred = nn.predict(X_test)
test_set_rsquared = nn.score(X_test, y_test)
test_set_rmse = np.sqrt(mean_squared_error(y_test, pred))
print('Rsquared value: ', test_set_rsquared)
print('RMSE: ', test_set_rmse)
"""
Plot the predictions
"""
plt.plot(y_test, label = "Database")
plt.plot(pred, label = "Predictions")
plt.legend()
plt.show()
```

```
Rsquared value:  0.5144754879313564
RMSE:  0.7989435898734182
```

Regression - **ANN**



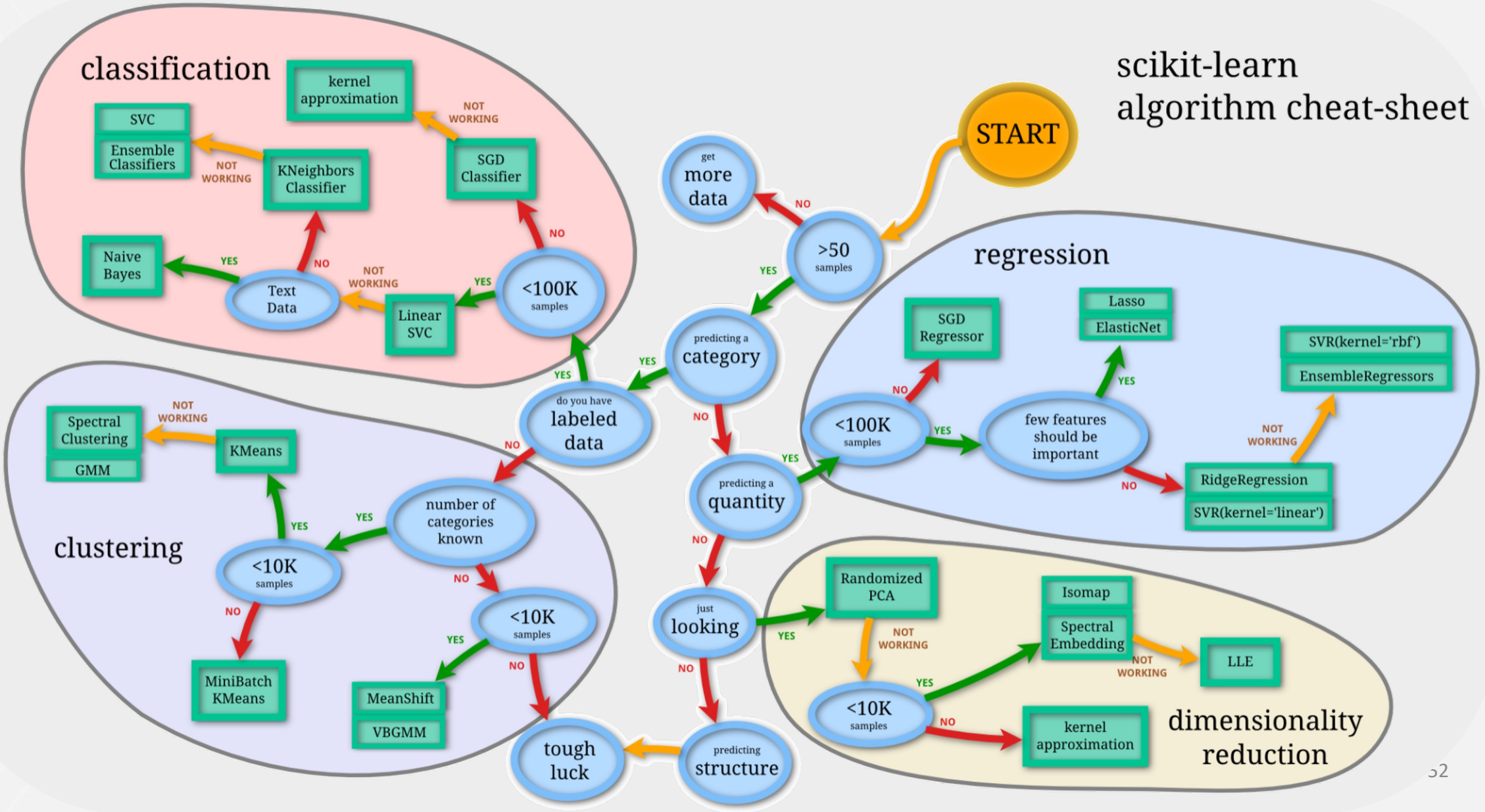


3- Key remarks

Machine learning algorithms **learn from data** to find hidden relations and make predictions.

A machine learning algorithm is as good as its input data.

The algorithms require proper optimization and validation methods to ensure the generalization capability of the model.


scikit-learn
algorithm cheat-sheet
START
get more data
NO
>50 samples
YES
predicting a category
YES
do you have labeled data
YES
<100K samples
NO
SGD Classifier
NOT WORKING
kernel approximation
NOT WORKING
KNeighbors Classifier
NOT WORKING
SVC
Ensemble Classifiers
NO
Text Data
YES
Naive Bayes
NO
Linear SVC
YES
classification
NO
predicting a quantity
YES
<100K samples
NO
SGD Regressor
YES
few features should be important
YES
Lasso
ElasticNet
NO
RidgeRegression
SVR(kernel='linear')
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors
regression
NO
number of categories known
YES
<10K samples
YES
KMeans
NOT WORKING
Spectral Clustering
GMM
NO
MiniBatch KMeans
NO
<10K samples
YES
MeanShift
VBGMM
NO
tough luck
clustering
NO
just looking
YES
Randomized PCA
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
YES
<10K samples
NO
kernel approximation
NO
predicting structure
dimensionality reduction