Python初級數據分析員證書

（六）數據分析 及可視化專案

# 13. 數據分析專案
# Machine Learning – Part 1

# Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling
- Machine Learning

## Chapter Summary

- Introduction
- Application
- Types of Machine Learning
- Example: Iris Species Classification
- Generalization
- Underfitting & Overfitting
- KNN

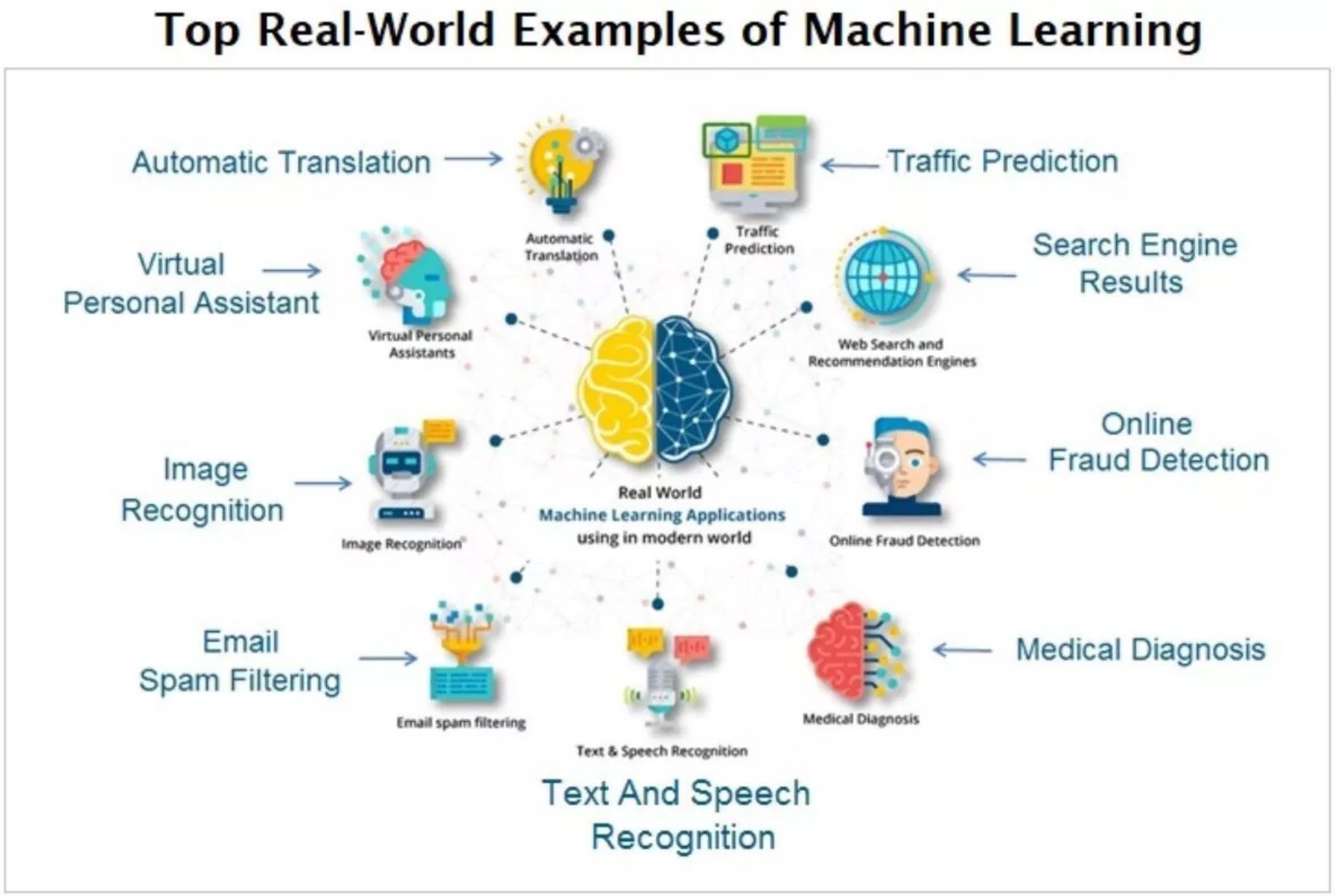
# Introduction

Machine learning is about extracting knowledge from data. It is a research field at the intersection of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning.

The application of machine learning methods has in recent years come into our life everyday everywhere.

The goal of machine learning is to develop algorithms that can automatically learn patterns and relationships in data, and use that knowledge to **make predictions or decisions**.

# Application



# Types of machine learning

- **Supervised learning**

  Train labelled dataset, where the correct answer is provided for each example. learns to make predictions on new, unseen data based on what it learned from the labelled examples. In our course, we mainly focus on supervised learning from scratch.

- **Unsupervised learning**

  Train unlabelled dataset, where the correct answer is not provided. The model then learns to identify patterns and relationships in the data on its own.

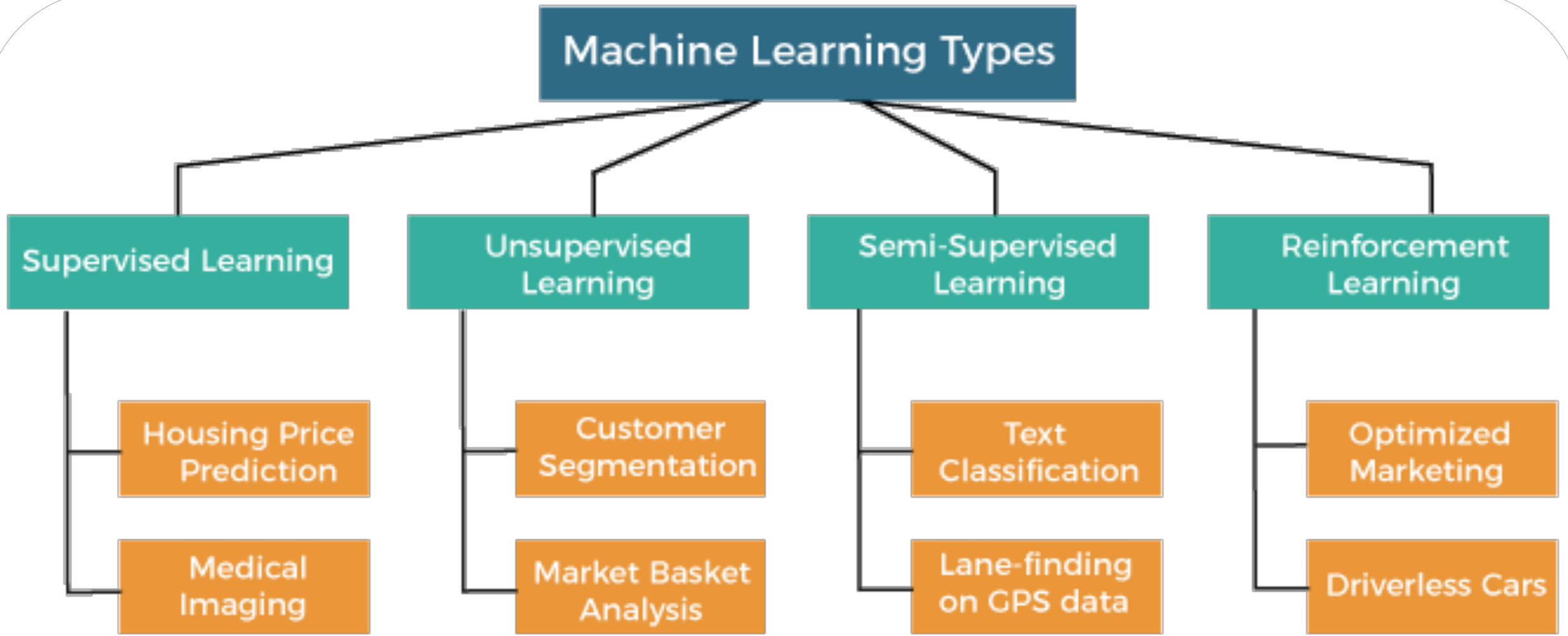
- **Semi-supervised learning**

  Train both labelled and unlabelled examples. The model can use the labelled examples to learn how to make predictions on new data, while also using the unlabelled examples to identify patterns and relationships in the data.

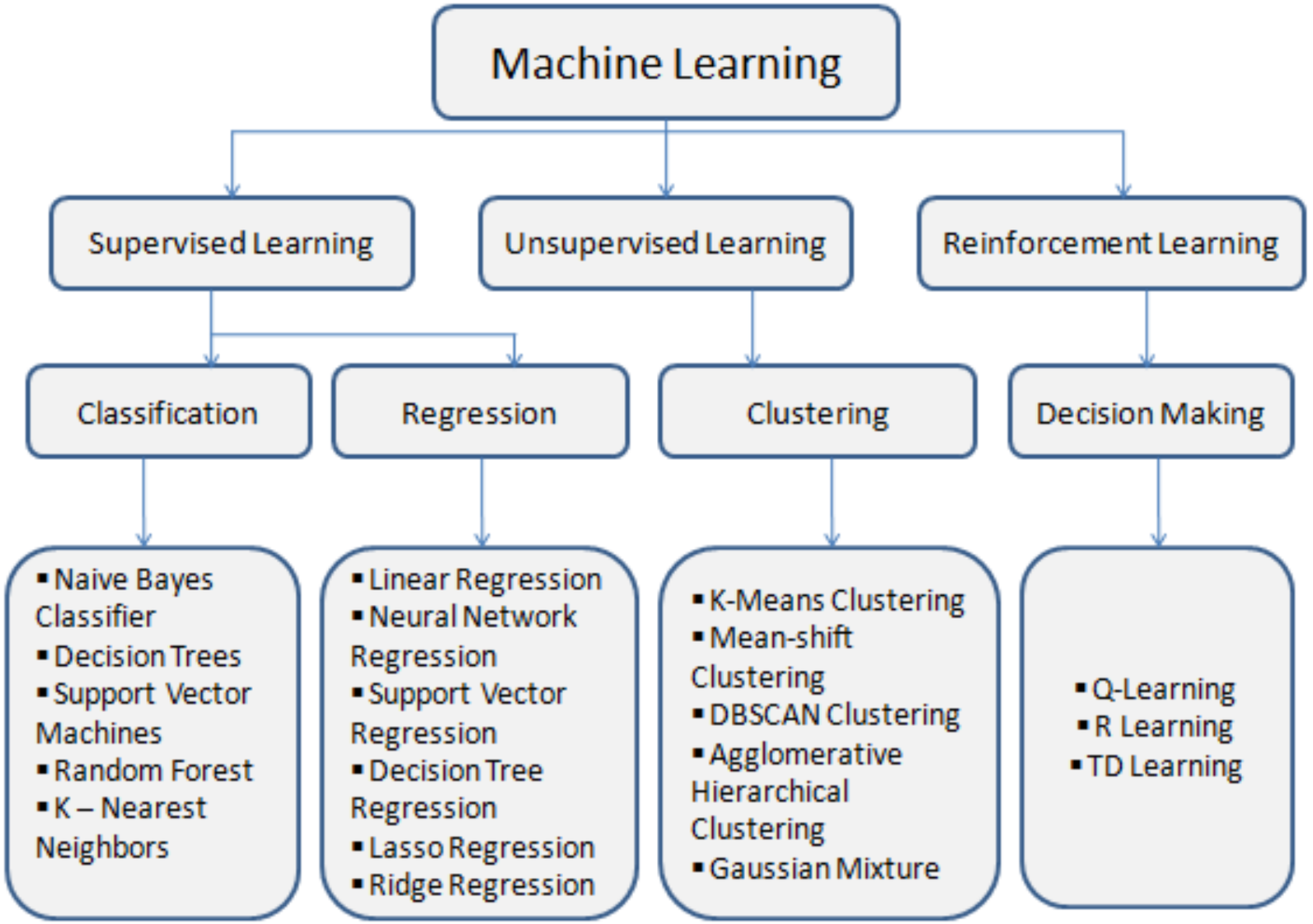
- **Reinforcement learning**

  Train making decisions based on rewards or punishments it receives for its actions. The model learns to take actions that lead to the greatest reward over time.

# Types of machine learning



# Typical Machine Learning Methods



# Why Python

Python combines the power of general programming language with the ease of specific scripting language like MATLAB or R. Python has libraries for data loading, visualization, statistics, image processing, natural language processing and much more.

By using Jupyter Note Book or terminal, Python is interactive with code. It is essential for these processes to have tools that allow quick iteration and easy interaction.

# Scikit-learn

Scikit-learn is an open source machine learning project that consistently being developed and improved. It contains a number of advanced machine learning algorithms.

Scikit-learn works well with many other major python tools, such as NumPy, Pandas, SciPy and more.

Scikit-learn dependency packages: NumPy and SciPy.

# Scikit-learn Installation

Install scikitk-learn in iPython.

```
1 pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /Users/h
1.1)
Requirement already satisfied: numpy>=1.17.3 in /Users/
rom scikit-learn) (1.23.1)
Requirement already satisfied: scipy>=1.3.2 in /Users/h
om scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.0.0 in /Users/
rom scikit-learn) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
ages (from scikit-learn) (3.1.0)
```

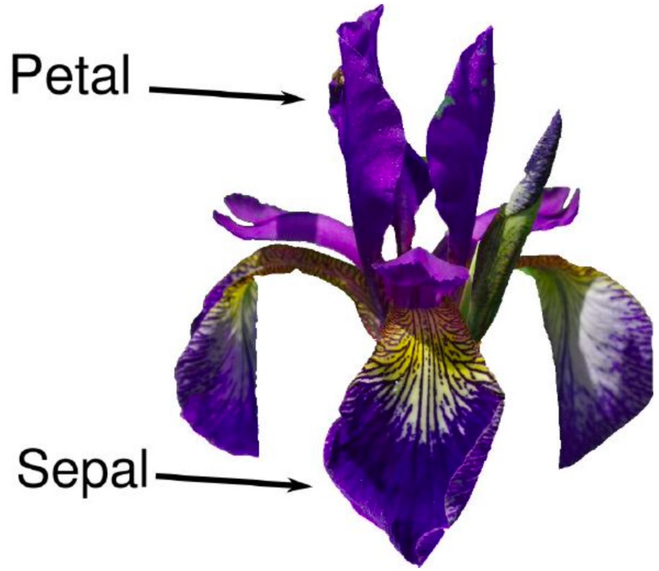
# Classify Iris Species

## Build your first Machine Learning application

```
1  import pandas as pd
2  import numpy as np
3  import plotly.express as px
4  import plotly.graph_objects as go
5  from sklearn.neighbors import KNeighborsClassifier
6  from sklearn.model_selection import train_test_split
7  import warnings
8  warnings.simplefilter("ignore")
9
10 iris_df = pd.read_csv("iris.csv")
11 iris_df.sample(3)
```

| | sepal.length | sepal.width | petal.length | petal.width | variety |
|---|---|---|---|---|---|
| **17** | 5.1 | 3.5 | 1.4 | 0.3 | Setosa |
| **97** | 6.2 | 2.9 | 4.3 | 1.3 | Versicolor |
| **117** | 7.7 | 3.8 | 6.7 | 2.2 | Virginica |

Our target is classifying the Iris species.



# Check Target and Define Features

```
1 iris_df['variety'].unique()
```

```
array(['Setosa', 'Versicolor', 'Virginica'], dtype=object)
```

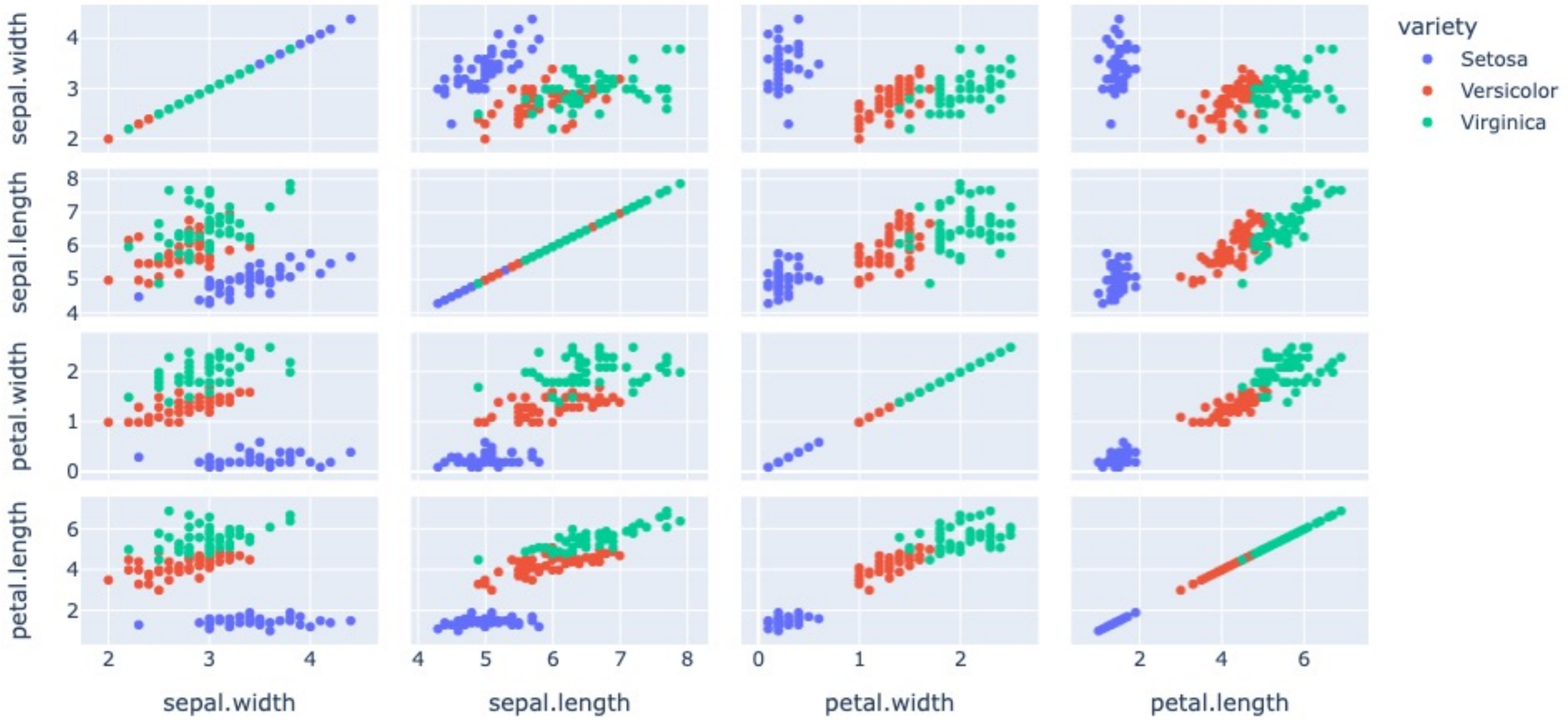
```
1 iris_df['variety'].value_counts()
```

```
Setosa          50
Versicolor      50
Virginica       50
Name: variety, dtype: int64
```

```
1 features = ['sepal.length','sepal.width','petal.length','petal.width']
```

# Visualize Features and Target

```
1  fig = px.scatter_matrix(iris_df, dimensions=["sepal.width", "sepal.length",
2                                               "petal.width", "petal.length"], color="variety")
3  fig.show()
```

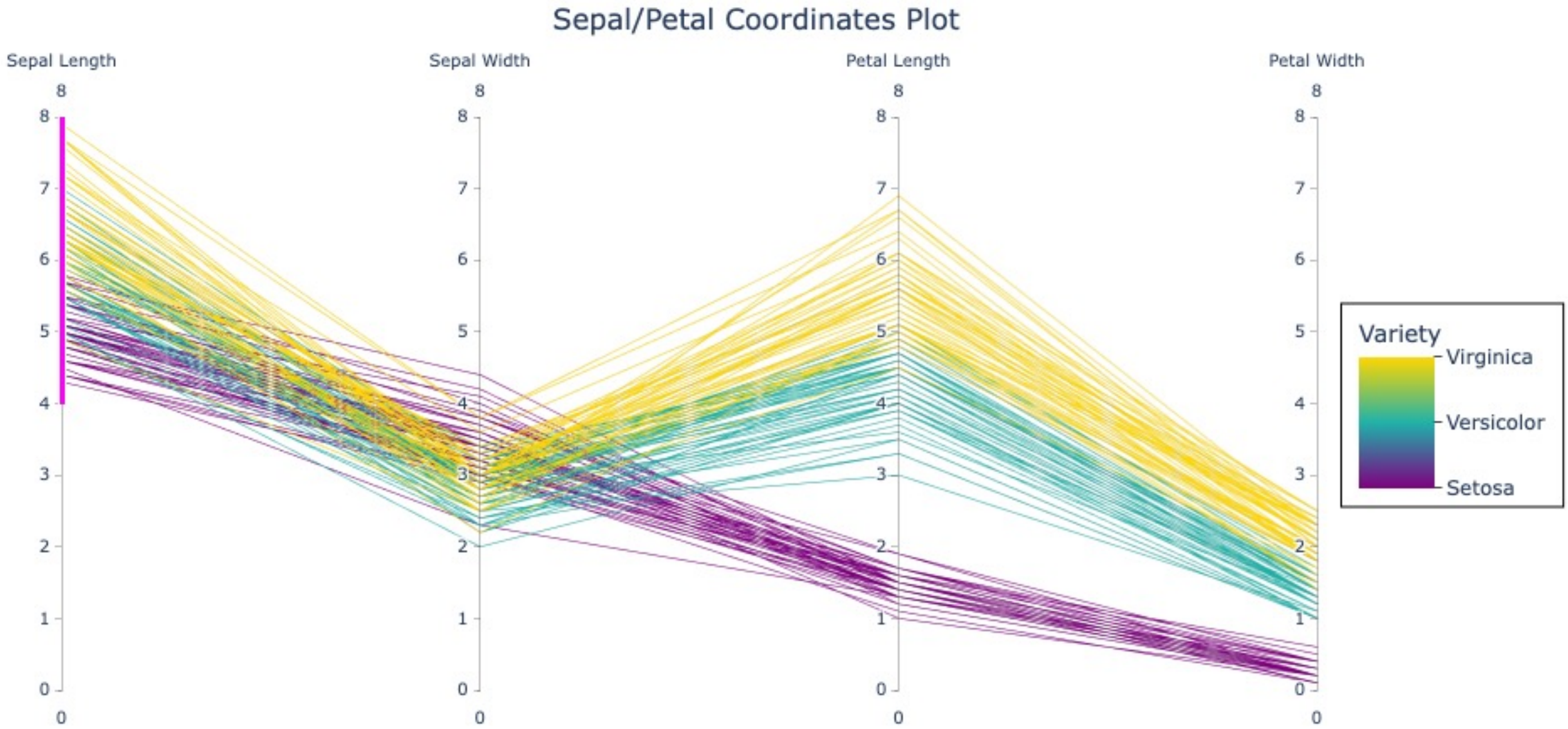


# Visualize with Coordinate Plot

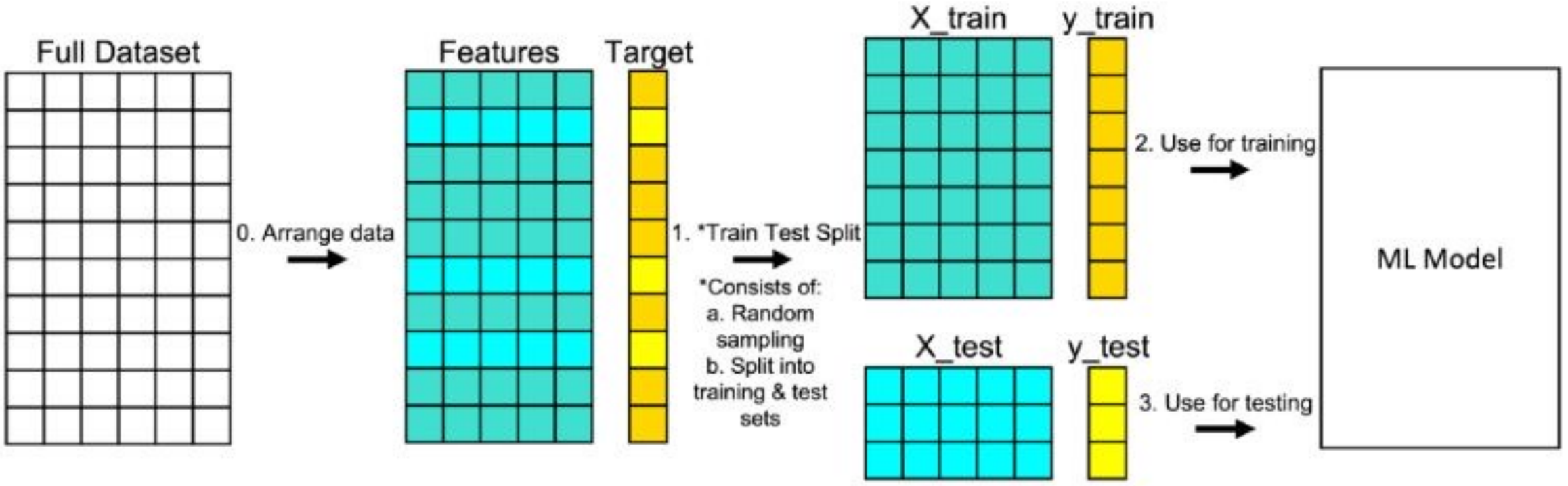
```
fig = go.Figure(data = go.Parcoords(
    line = dict(color=iris_df.variety.astype('category').cat.codes,
                colorscale = [[0,'purple'],[0.5,'lightseagreen'],[1,'gold']],
                colorbar=dict(title='Variety', thickness=45,
                              tickvals=[0,1,2], ticktext=['Setosa', 'Versicolor', 'Virginica'],
                              lenmode='pixels', len=120, bordercolor='#000', borderwidth=1,
                              ticks='outside'), ),
    dimensions = list([
        dict(range = [0,8],
            constraintrange = [4,8],
            label = 'Sepal Length', values = iris_df['sepal.length']),
        dict(range = [0,8],
            label = 'Sepal Width', values = iris_df['sepal.width']),
        dict(range = [0,8],
            label = 'Petal Length', values = iris_df['petal.length']),
        dict(range = [0,8],
            label = 'Petal Width', values = iris_df['petal.width'])
                ]),
            )
            )
fig.update_layout(plot_bgcolor = 'white', paper_bgcolor = 'white',
                  title_text='Sepal/Petal Coordinates Plot', title_x=0.5
)

fig.show()
```

# Visualize with Coordinate Plot



# Split DF into Train and Test



# Split DF into Train and Test

```
1 X_train, X_test, y_train, y_test = train_test_split(
2         iris_df[features], iris_df['variety'], random_state=0)
```

The train_test_split function is X_train, X_test, y_train, and y_test, which are all NumPy arrays. X_train contains 75% of the rows of the dataset, and X_test contains the remaining 25%

```
1 print("X_train shape: {}".format(X_train.shape))
2 print("y_train shape: {}".format(y_train.shape))
```

```
X_train shape: (112, 4)
y_train shape: (112,)
```

# Train the model with 75% train data, with train Target

```
# Train the model
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

```
# Accuracy Score (Max: 1)
knn.score(X_test, y_test)
```

```
0.9736842105263158
```

# Make a prediction of unknown species

```
1 X_new = pd.DataFrame(np.array([[5, 2.9, 1, 0.2]]), columns=features )
2 X_new
```

| | sepal.length | sepal.width | petal.length | petal.width |
|---|---|---|---|---|
| **0** | 5.0 | 2.9 | 1.0 | 0.2 |

```
1 print("This is a",knn.predict(X_new)[0])
```

```
This is a Setosa
```

# Classification and Regression

There are two major types of supervised machine learning problems, called *classification* and *regression*.

In classification, the goal is to predict a class label, which is a choice from a predefined list of possibilities. We just used the example of classifying irises into one of three possible species.

For regression tasks, the goal is to predict a continuous number, or a floating-point number in programming terms (or real number in mathematical terms).

# Classification and Regression

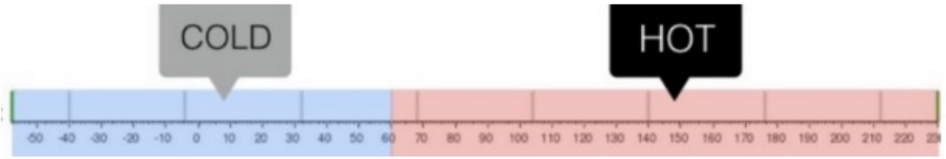
## Regression

What will be the temperature tomorrow?

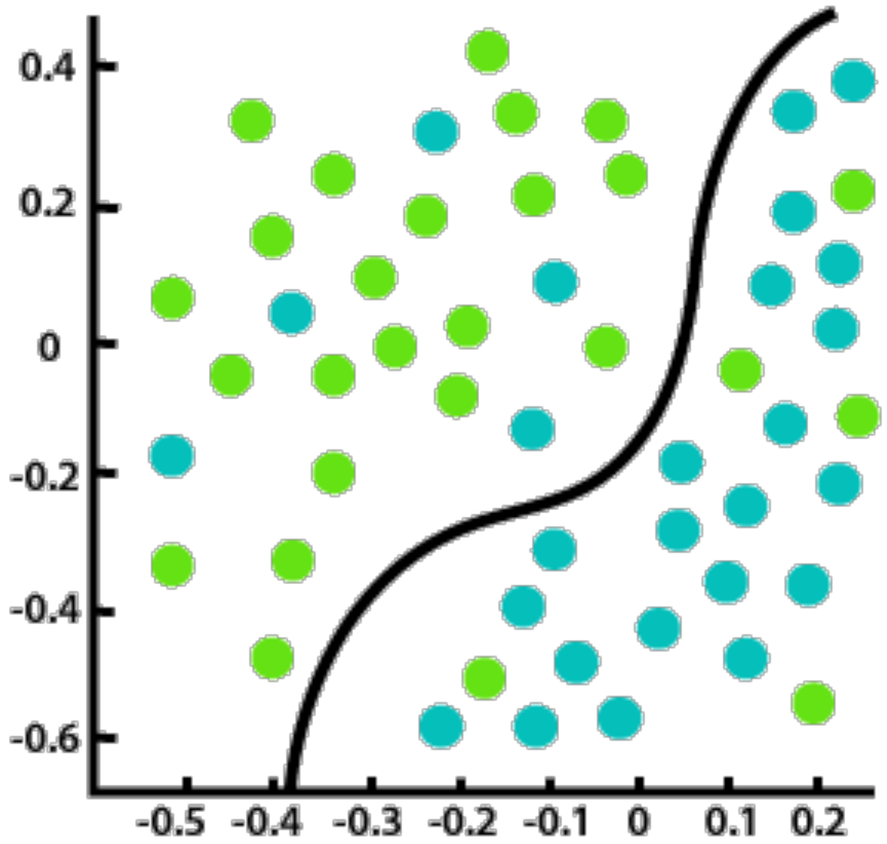

Fahrenheit

## Classification

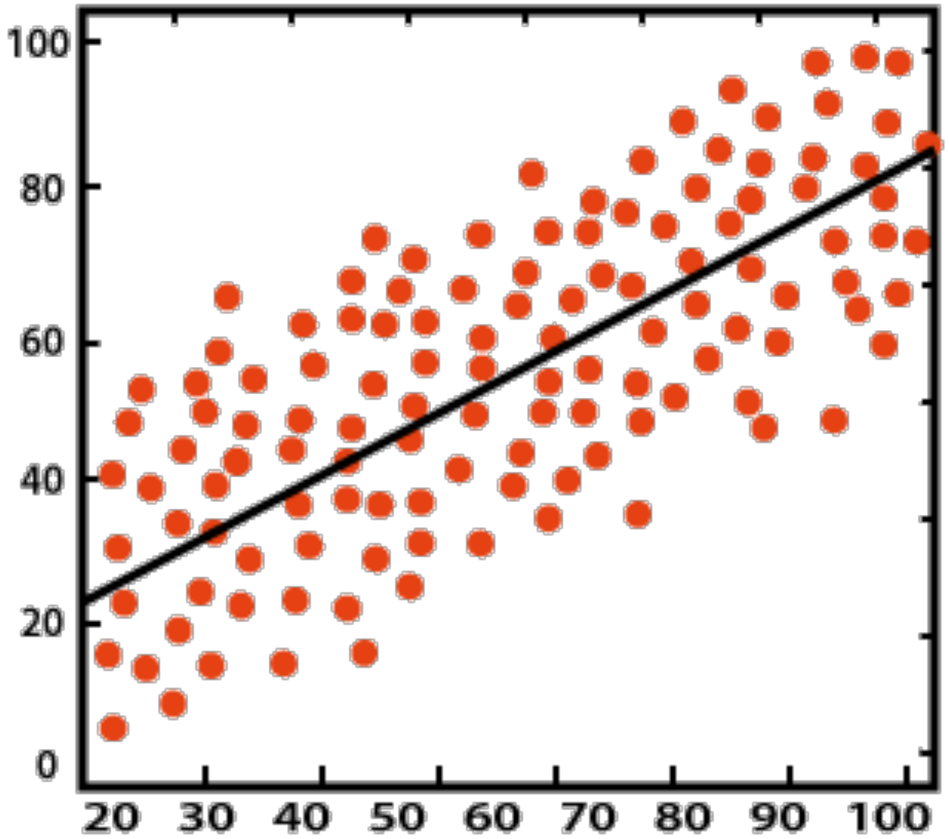
Will it be hot or cold tomorrow?




Fahrenheit



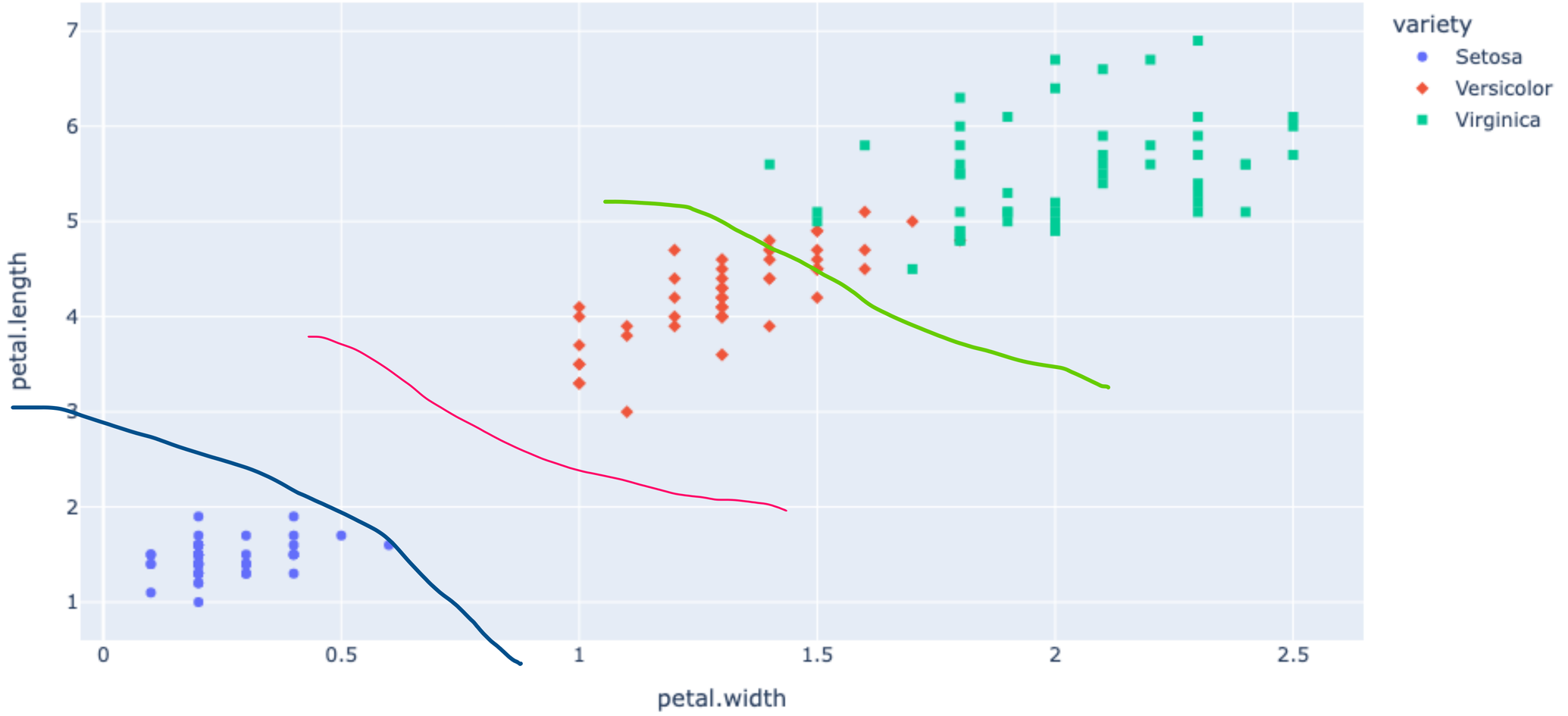

Classification




Regression

# Idea of Iris Classification

```
1 fig = px.scatter(iris_df, x="petal.width", y="petal.length",
2                   color="variety", symbol="variety")
3 fig.show()
```



# Generalization

In supervised learning, we want to build a model on the training data and then be able to make accurate predictions on new, unseen data that has the same characteristics as the training set that we used.

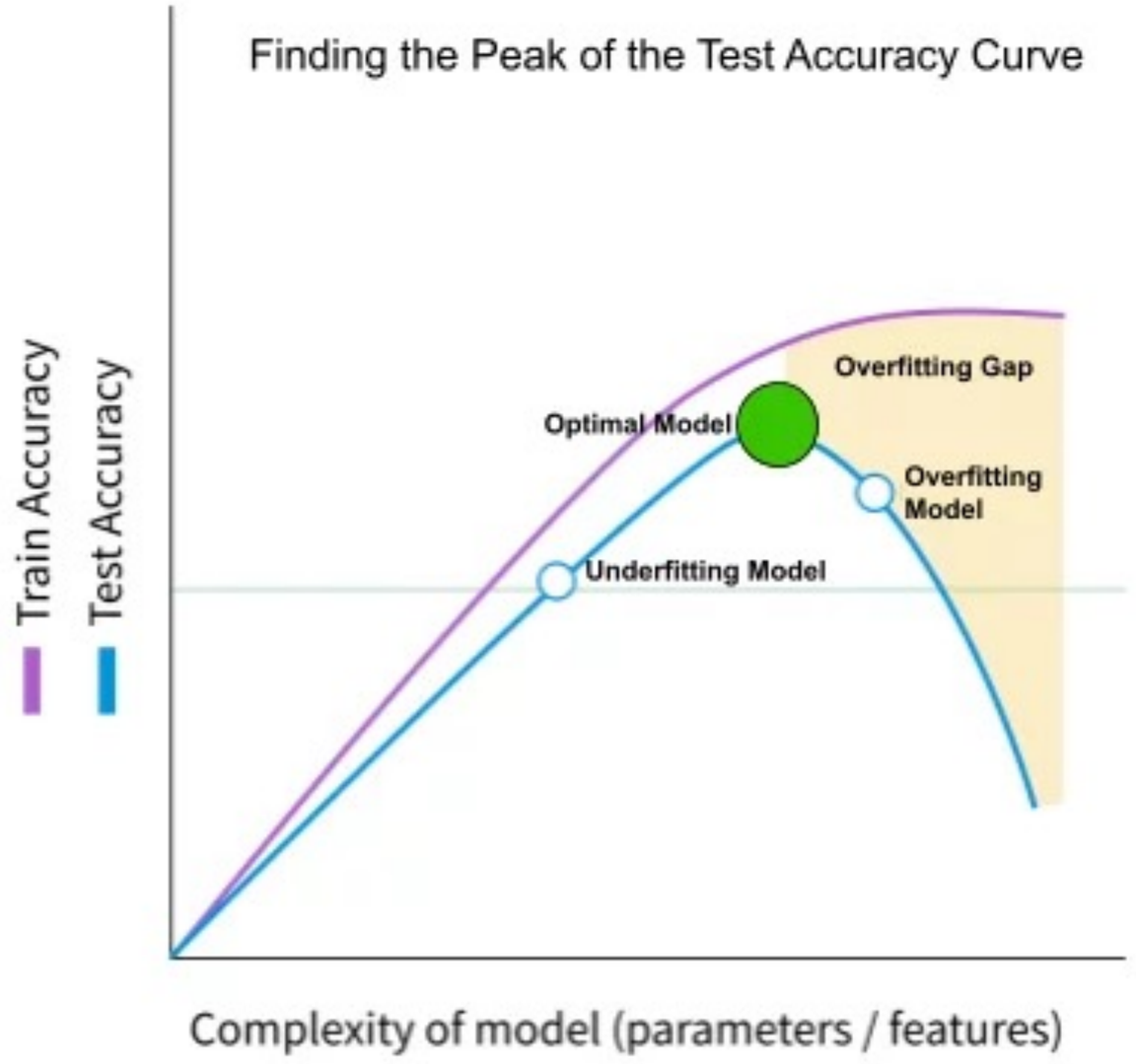
If a model is able to make accurate predictions on unseen data, we say it is able to **generalize** from the training set to the test set.

# Overfitting & Underfitting

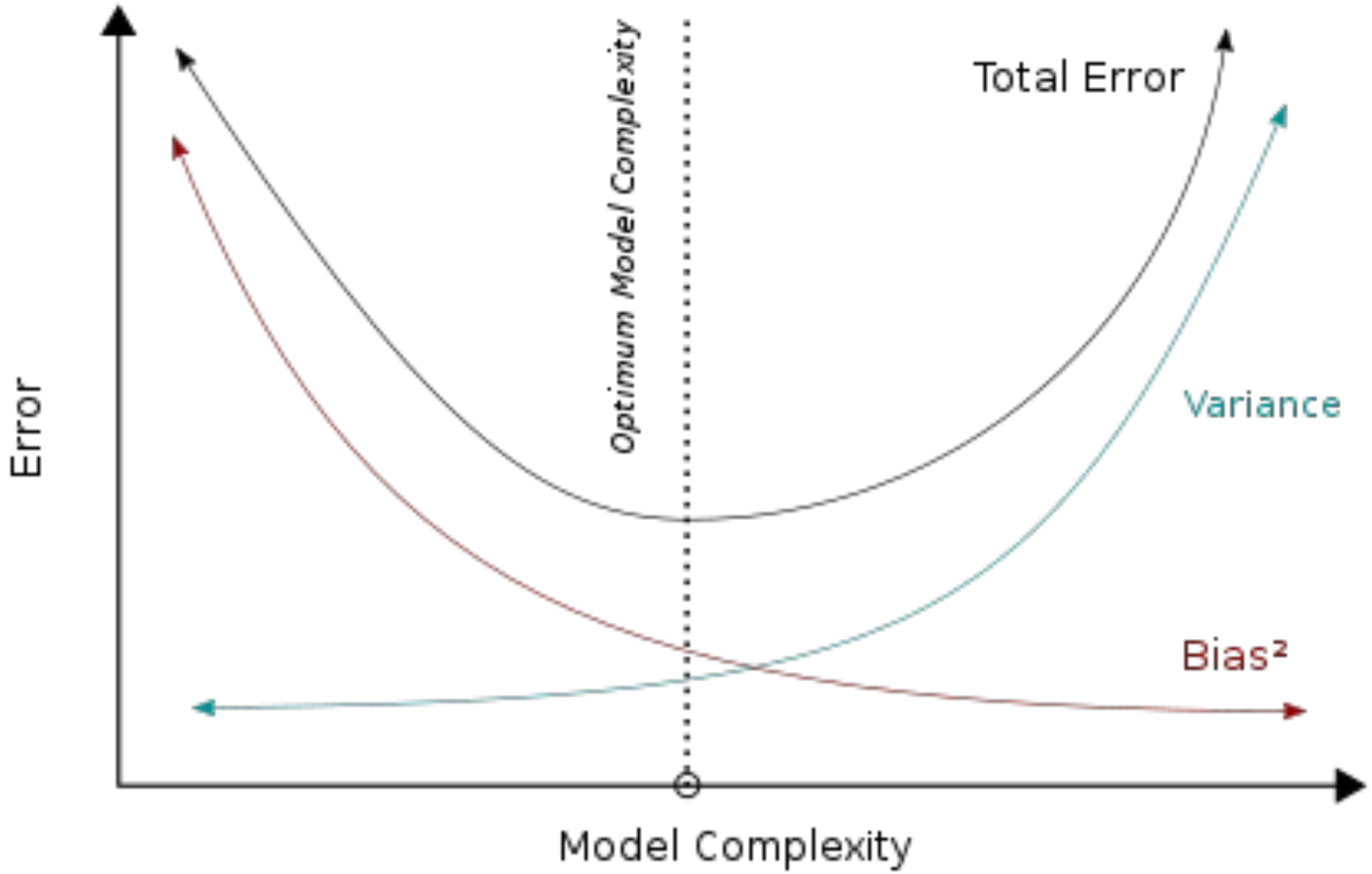
Underfitting occurs when a model is too simple to capture the underlying patterns in the data. This means that the model's predictions are not accurate, and it is said to have high bias. In other words, the model is not able to fit the training data well enough to generalize predictions on new, unseen data.

Overfitting occurs when a model is too complex and captures noise or random fluctuations in the training data as well as the underlying patterns. This means that the model's predictions are not accurate on new, unseen data, and it is said to have high variance.

# Overfitting & Underfitting



# Overfitting & Underfitting

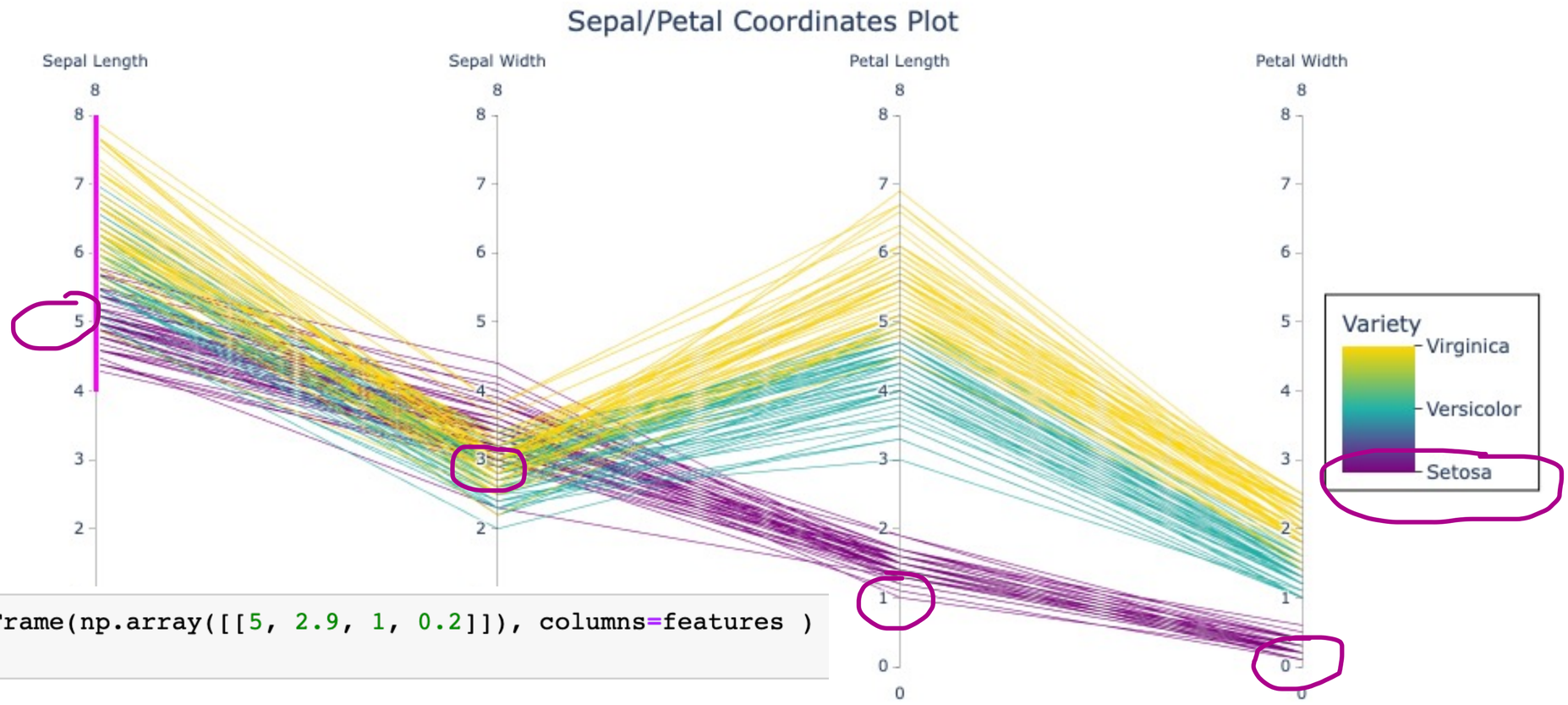


# KNN

The KNN algorithm is arguably the simplest machine learning algorithm. Building the model consists only of storing the training dataset. To make a prediction for a new data point, the algorithm finds the closest data points in the training dataset—its “nearest neighbours.”

The basic idea behind KNN is to find the K nearest neighbours to a given data point and use their labels or values to predict the label or value of the given data point. The "K" in KNN refers to the number of nearest neighbours to consider.

# KNN's idea




```
X_new = pd.DataFrame(np.array([[5, 2.9, 1, 0.2]]), columns=features )
X_new
```

| | sepal.length | sepal.width | petal.length | petal.width |
|---|---|---|---|---|
| **0** | 5.0 | 2.9 | 1.0 | 0.2 |

```
print("This is a",knn.predict(X_new)[0])
```

```
This is a Setosa
```

# KNN algorithm

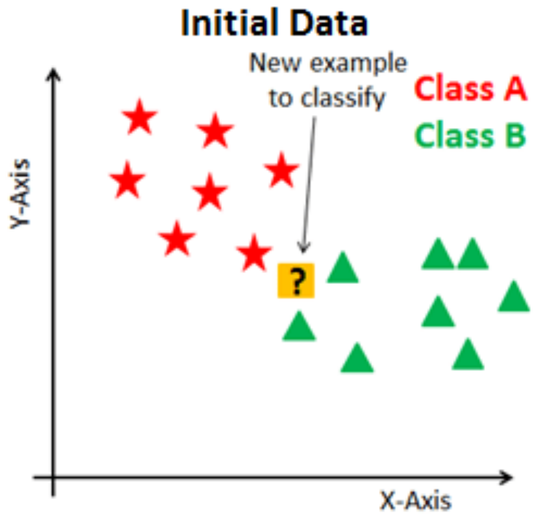
## The KNN algorithm works as follows:

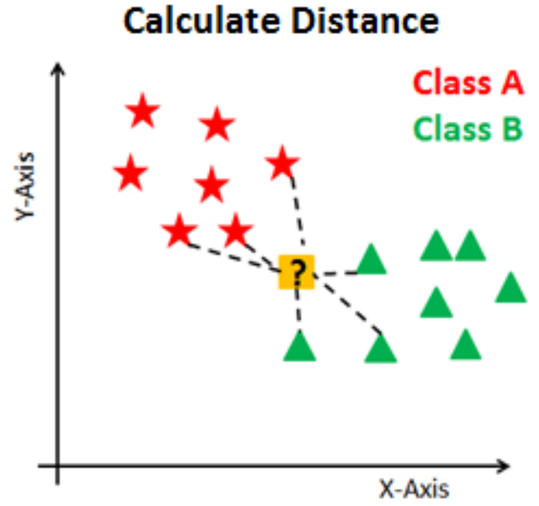
1. Given a new data point, find the K nearest neighbours to that point based on some distance metric (e.g., Euclidean distance).
2. Determine the class label or value of the new data point based on the most common class label or average value of its K nearest neighbours.
3. Repeat steps 1 and 2 for all new data points.

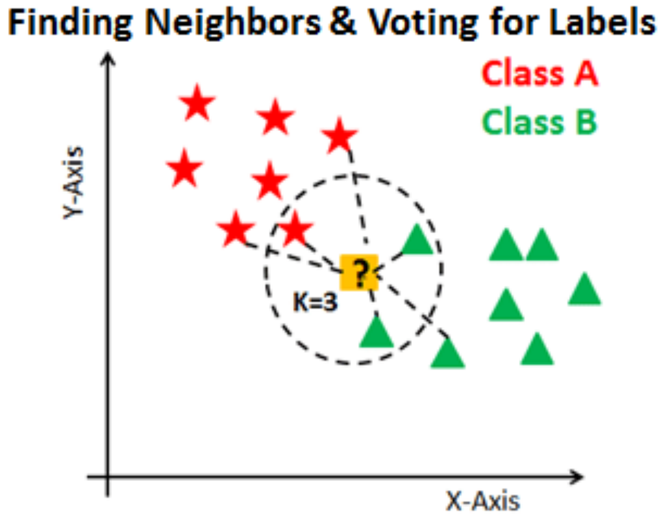
# Why we usually take more than 5 neighbour?

In real case of sample training data, the classes data may overlap. So it would be more accuracy to take average of a numbers of nearest neighbour for label the target.

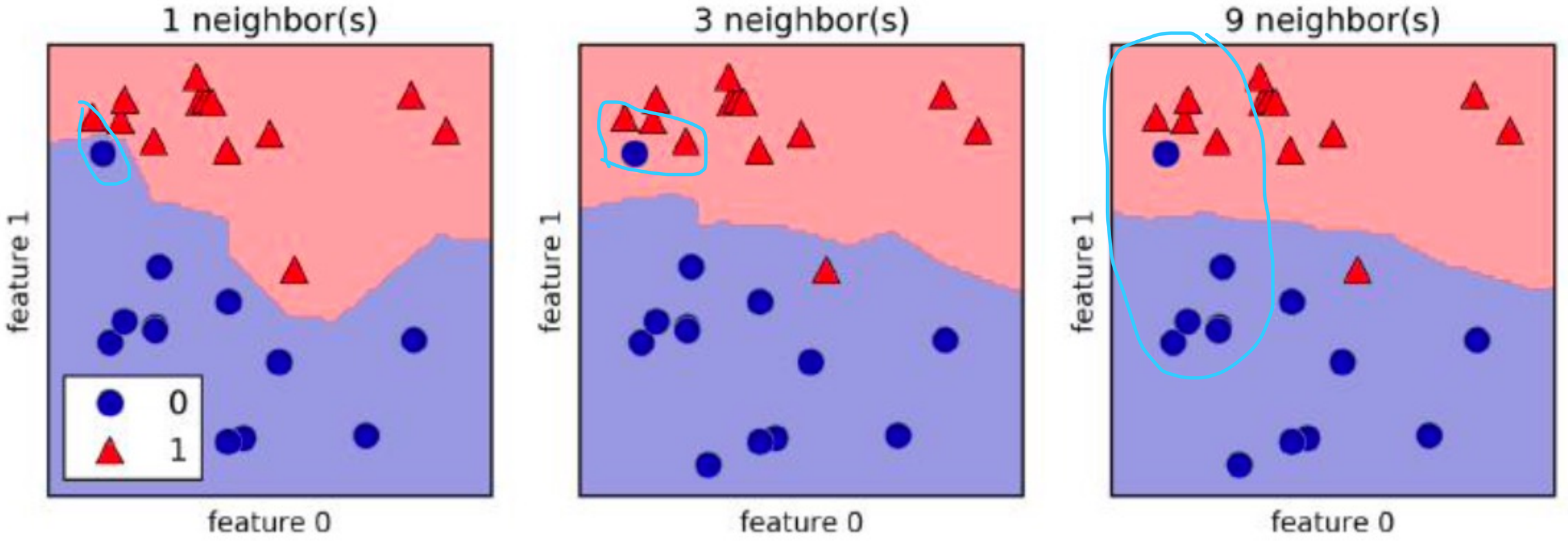
The KNN default is take 5 nearest neighbours.



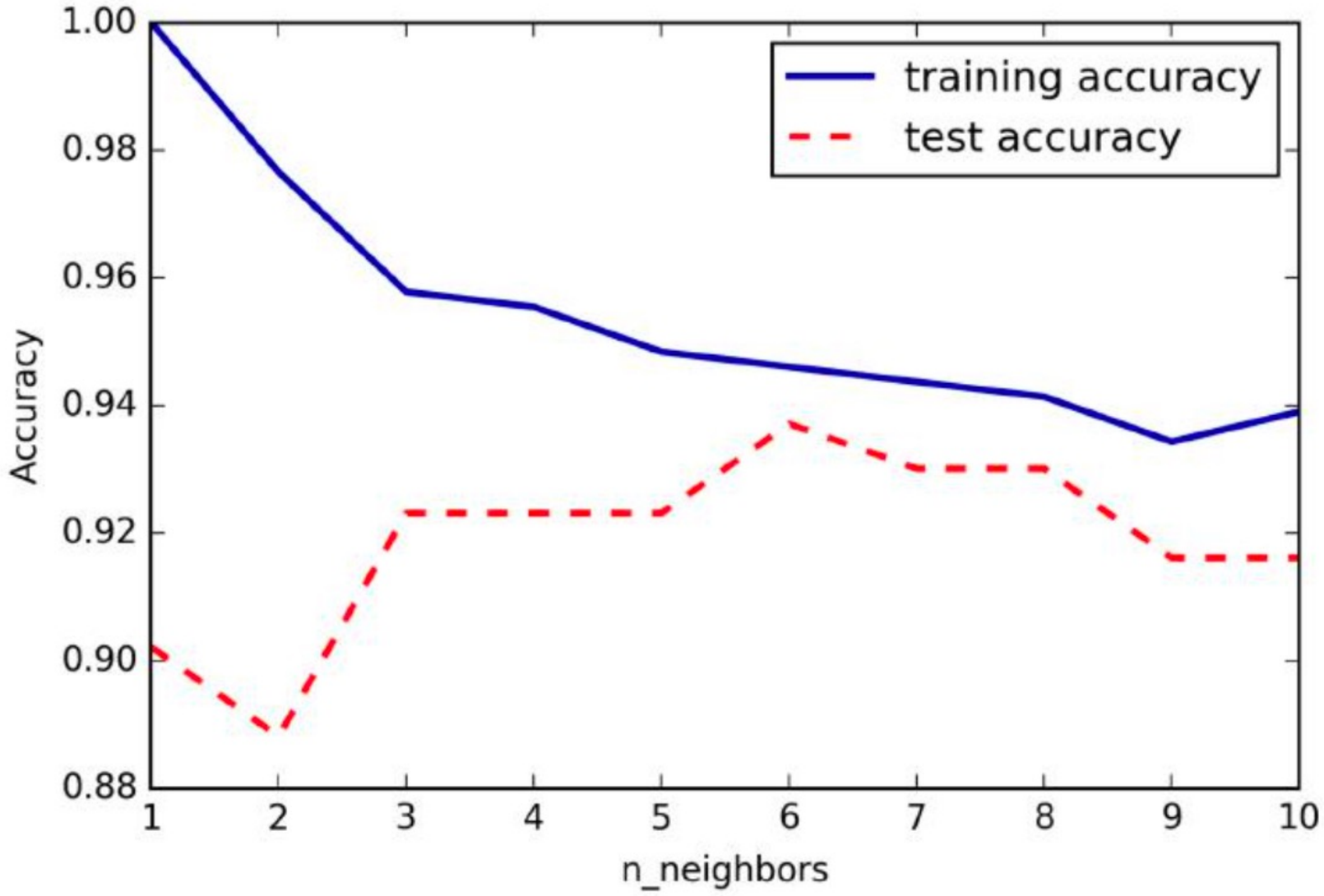



# 1 vs 3 vs 9 neighbors



# Should I use many neighbour as possible?




Remember the complexity model. Less bias in extreme may lead to bigger variance.

# Chapter Wrap Up

| Classification Algorithm | Regression Algorithm |
|---|---|
| In Classification, the output variable must be a discrete value. | In Regression, the output variable must be of continuous nature or real value. |
| The task of the classification algorithm is to map the input value(x) with the discrete output variable(y). | The task of the regression algorithm is to map the input value (x) with the continuous output variable(y). |
| Classification Algorithms are used with discrete or continuous data. | Regression Algorithms are used with continuous data. |
| In Classification, we try to find the decision boundary, which can divide the dataset into different classes. | In Regression, we try to find the best fit line, which can predict the output more accurately. |
| Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc. | Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc. |
| The Classification algorithms can be divided into Binary Classifier and Multi-class Classifier. | The regression Algorithm can be further divided into Linear and Non-linear Regression. |

# Reference & Resources

Scikit Learn:

https://scikit-learn.org/

IPython Cookbook, 2nd edition

https://ipython-books.github.io/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

Seaborn:

https://seaborn.pydata.org/examples/index.html

Matplotlib:

https://matplotlib.org/