K-Nearest Neighbors Algorithm

- Nayan Man Singh Pradhan

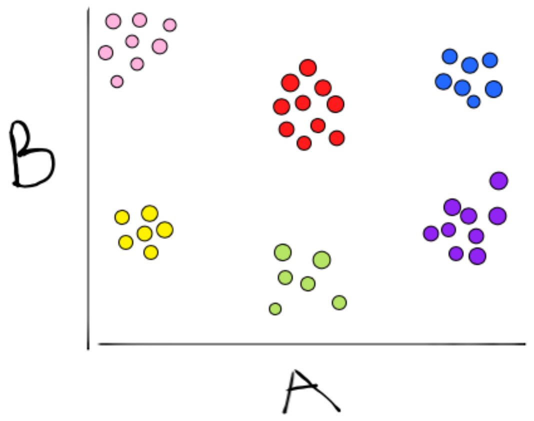
The K-Nearest Neighbors (KNN) Algorithm is a popular **supervised** machine learning algorithm that can be used for both **classification** and **regression** problems (although it is used more for classification problems).

This blog aims to:

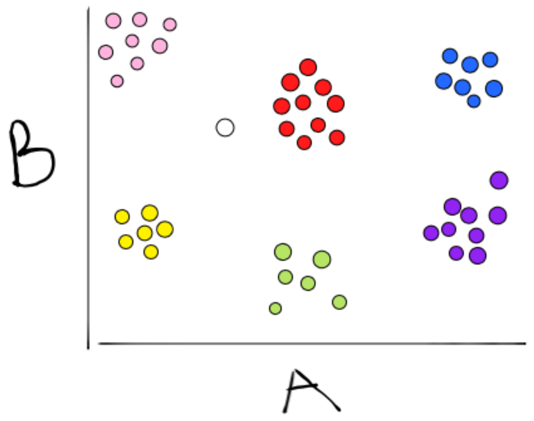
- Introduce the KNN algorithm
- Demonstrate how the KNN algorithm works through visual pictures
- Answer what the best value for K is
- Discuss the advantages and disadvantages of the KNN algorithm
- Show an example of the algorithm working

How the KNN algorithm works:

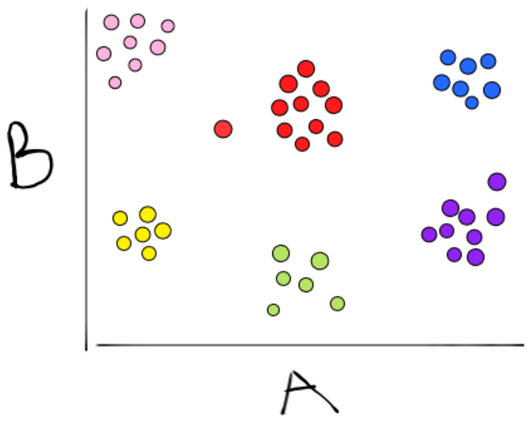
- Cluster the given dataset into the known categories.



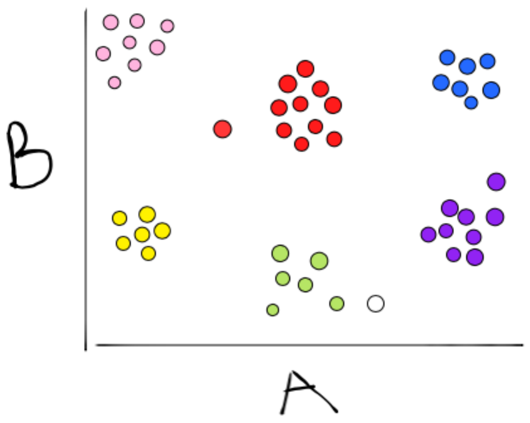
- Add the unknown cell that we want to categorize.



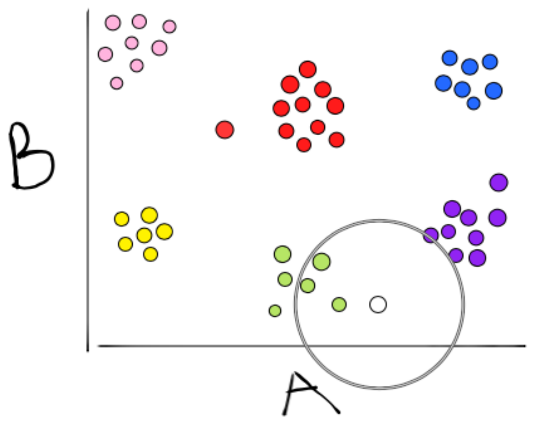
- Classify the new data point by looking at the nearest neighbors. The K in KNN signifies how many neighbors the algorithm considers while classifying the new data point.

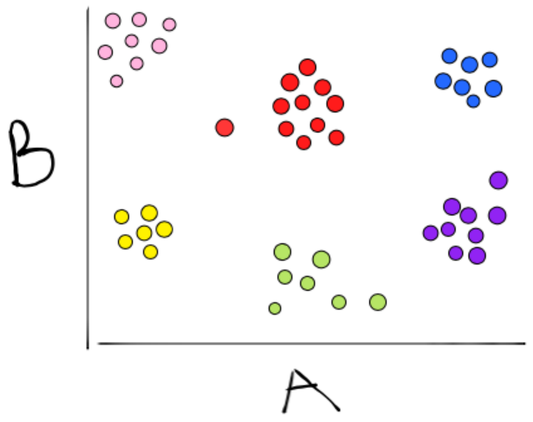


- The K plays a significant role in classification. Let us assume a new data point in the figure.

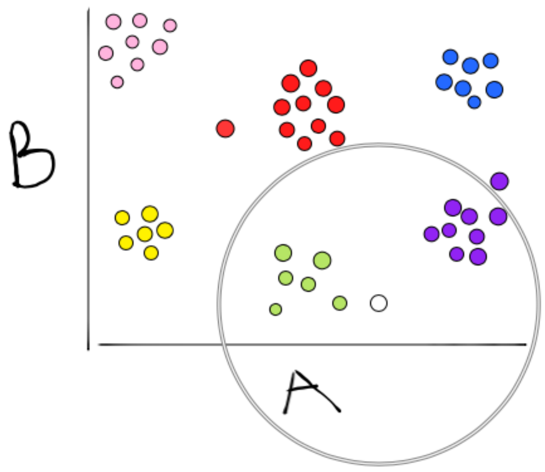


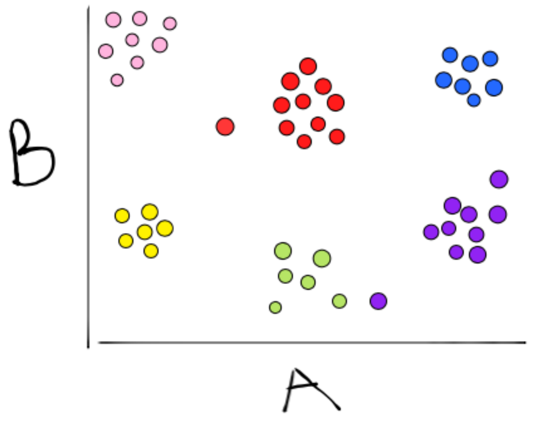
- If K = 3, the data point is classified as green. This is because, the 3 closest given data points are classified as green.





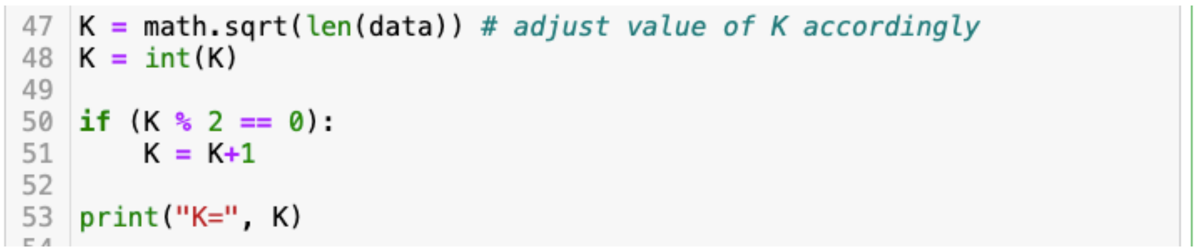
- But, if K = 14, the data point is classified as purple. This is because, there are 6 green nearest data points and 8 purple nearest data points. The KNN algorithm classifies the point to the data set with the largest number of nearest neighbors. Hence, one must be very careful while choosing K.





How to pick the optimal K value:

- There is no formula the gives the correct/optimal value for K.
- The value of K differs according the the number of data sets.
- Having K as an odd number is usually recommended as the extra data point acts as a tie breaker (if the data sets are evenly spread out between two classifications).
- Personally, I like to choose K as the integer equal to the square root of the length of the data set. If K is an even integer, I add 1 to K. This is illustrated by the code snippet below:



```python
K = math.sqrt(len(data)) # adjust value of K accordingly
K = int(K)

if (K % 2 == 0):
    K = K+1

print("K=", K)
```

Advantages of the KNN algorithm:

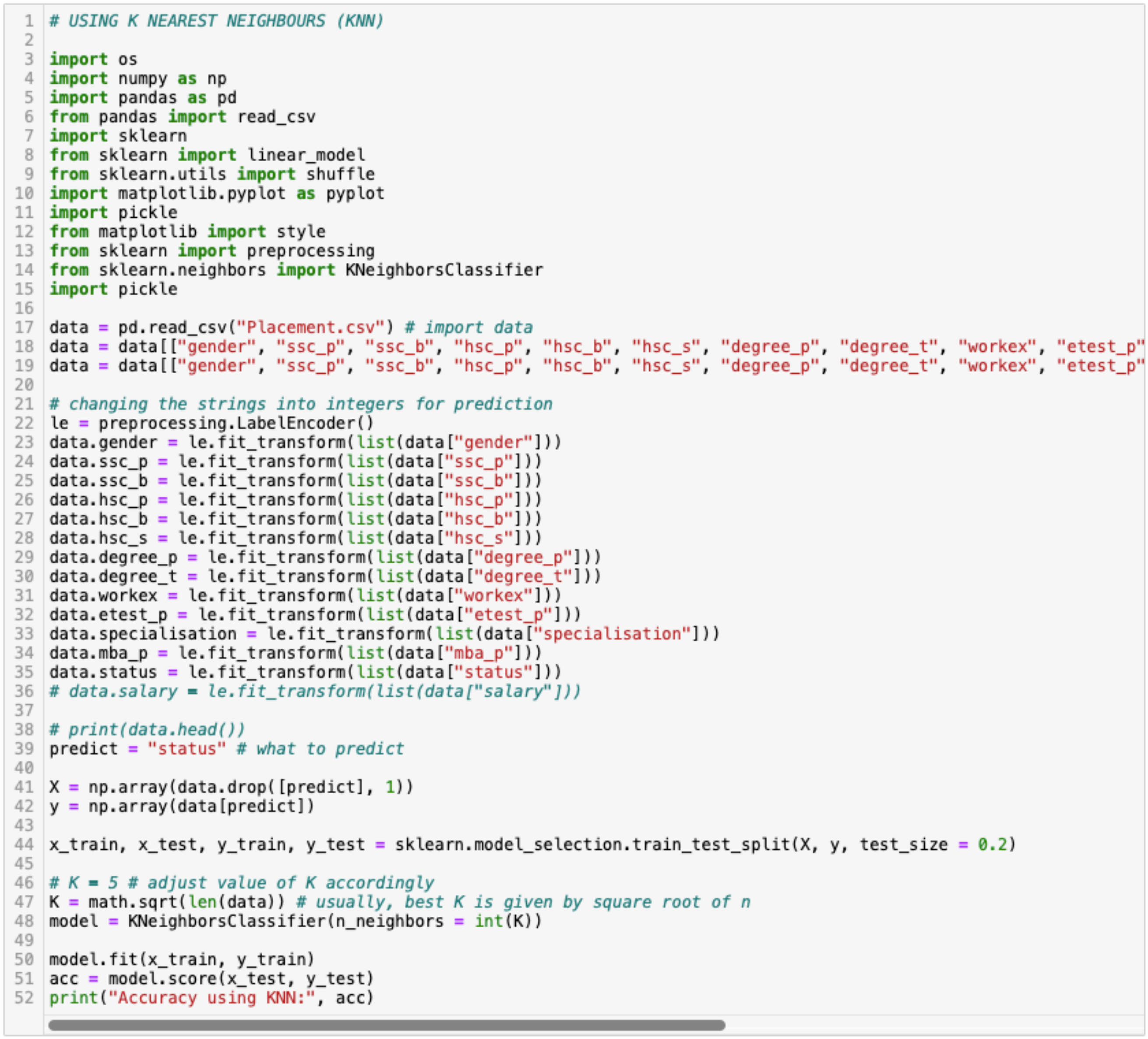
- Understanding and implementing the algorithm is simple.
- The algorithm can be used for classification and regression problems, hence, it is versatile.
- There is no need for building models and adjusting various parameters.

Disadvantages of the KNN algorithm:

- The algorithm gets slow as the number of independent variables increases.

Example of the KNN algorithm:

The algorithm below is predicting whether an employer gets accepted or rejected while applying for a job.



```python
# USING K NEAREST NEIGHBOURS (KNN)

import os
import numpy as np
import pandas as pd
from pandas import read_csv
import sklearn
from sklearn import linear_model
from sklearn.utils import shuffle
import matplotlib.pyplot as pyplot
import pickle
from matplotlib import style
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
import pickle

data = pd.read_csv("Placement.csv") # import data
data = data[["gender", "ssc_p", "ssc_b", "hsc_p", "hsc_b", "hsc_s", "degree_p", "degree_t", "workex", "etest_p"
data = data[["gender", "ssc_p", "ssc_b", "hsc_p", "hsc_b", "hsc_s", "degree_p", "degree_t", "workex", "etest_p"

# changing the strings into integers for prediction
le = preprocessing.LabelEncoder()
data.gender = le.fit_transform(list(data["gender"]))
data.ssc_p = le.fit_transform(list(data["ssc_p"]))
data.ssc_b = le.fit_transform(list(data["ssc_b"]))
data.hsc_p = le.fit_transform(list(data["hsc_p"]))
data.hsc_b = le.fit_transform(list(data["hsc_b"]))
data.hsc_s = le.fit_transform(list(data["hsc_s"]))
data.degree_p = le.fit_transform(list(data["degree_p"]))
data.degree_t = le.fit_transform(list(data["degree_t"]))
data.workex = le.fit_transform(list(data["workex"]))
data.etest_p = le.fit_transform(list(data["etest_p"]))
data.specialisation = le.fit_transform(list(data["specialisation"]))
data.mba_p = le.fit_transform(list(data["mba_p"]))
data.status = le.fit_transform(list(data["status"]))
# data.salary = le.fit_transform(list(data["salary"]))

# print(data.head())
predict = "status" # what to predict

X = np.array(data.drop([predict], 1))
y = np.array(data[predict])

x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y, test_size = 0.2)

# K = 5 # adjust value of K accordingly
K = math.sqrt(len(data)) # usually, best K is given by square root of n
model = KNeighborsClassifier(n_neighbors = int(K))

model.fit(x_train, y_train)
acc = model.score(x_test, y_test)
print("Accuracy using KNN:", acc)
```

```
Accuracy using KNN: 0.7906976744186046
```

***The full code can be found in my GitHub:**

https://github.com/nayan-pradhan/MachineLearning_Summer2020/blob/master/jobAcceptance/predictionUsingKNN.ipynb