|  |  |
| --- | --- |
| Algorithm: K-Nearest-Neighbors | |
| USN : 1MS17CS143 | NAME : Sathvik K P |
| USN : 1MS17CS148 | NAME : Sathvik B |

**Description of the Algorithm:**

A k-nearest-neighbor is a data classification algorithm that attempts to determine what group a data point is in by looking at the data points around it.

An algorithm, looking at one point on a grid, trying to determine if a point is in group A or B, looks at the states of the points that are near it. The range is arbitrarily determined, but the point is to take a sample of the data. If the majority of the points are in group A, then it is likely that the data point in question will be A rather than B, and vice versa.

The k-nearest-neighbor is an example of a "lazy learner" algorithm because it does not generate a model of the data set beforehand. The only calculations it makes are when it is asked to poll the data point's neighbors. This makes k-nn very easy to implement for data mining.

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.



Notice in the image above that most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics i.e, calculating the distance between points on a graph.

**Algorithm Pseudocode:**

1.Load the data

2.Initialize K to your chosen number of neighbors

3.For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.

3.2 Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances

5. Pick the first K entries from the sorted collection

6. Get the labels of the selected K entries

7. If regression, return the mean of the K labels

8. If classification, return the mode of the K labels

**Data set Used: (Attach Screen shot of the few rows)**

****

**Challenges faced during the implementation of the program:**

1. It is a lazy learner
2. Model is dependent on input for which is tested
3. Not always accurate, other algorithms work better in most cases.

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import confusion\_matrix, accuracy\_score

df = pd.read\_csv("Iris.csv")

df.isna().sum()

df.drop(['Id'],axis=1, inplace=True)

sns.set(style='darkgrid')

setosa = df.query("Species=='Iris-setosa'")

versicolor = df.query("Species=='Iris-versicolor'")

virginica = df.query("Species=='Iris-virginica'")

f, ax = plt.subplots(figsize=(8, 8))

# ax.set\_aspect('equal')

ax = sns.kdeplot(setosa.SepalWidthCm, setosa.SepalLengthCm, cmap="Greens",

          shade=True, shade\_lowest=False)

ax = sns.kdeplot(versicolor.SepalWidthCm, versicolor.SepalLengthCm, cmap="Reds",

          shade=True, shade\_lowest=False)

ax = sns.kdeplot(virginica.SepalWidthCm, virginica.SepalLengthCm, cmap="Blues",

          shade=True, shade\_lowest=False)

red = sns.color\_palette("Reds")[-2]

blue = sns.color\_palette("Blues")[-2]

green = sns.color\_palette("Greens")[-2]

ax.text(2.5, 8.2, "virginica", size=16, color=blue)

ax.text(1.8, 6.5, "versicolor", size=16, color=red)

ax.text(3.8, 4.5, "setosa", size=16, color=green)

df['Species'] = df['Species'].apply(lambda x: 0 if x=='Iris-setosa' else (1 if x=='Iris-versicolor' else 2))

X=df.drop(['Species'],axis=1)

y=df['Species']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

from sklearn.neighbors import KNeighborsClassifier

neigh = KNeighborsClassifier(n\_neighbors=3,metric='minkowski')

neigh.fit(X\_train, y\_train)

y\_pred = neigh.predict(X\_test)

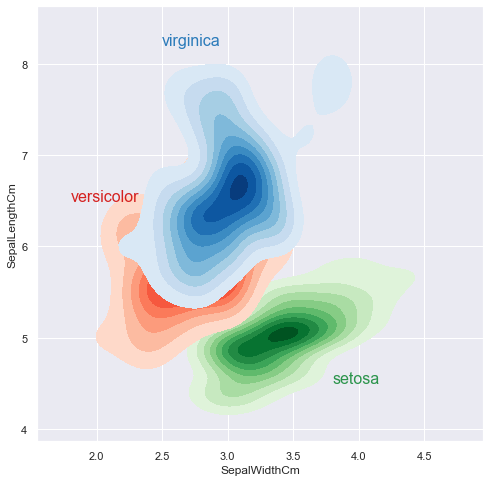
cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm,annot=True)

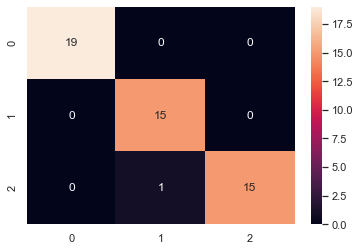
print(accuracy\_score(y\_test,y\_pred))

**Output: (Screen shots)**

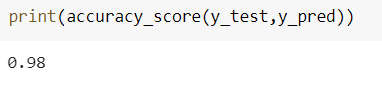
Exploratory Data Analysis : Heatmap Graph of sepal width vs sepal length (in cm)



Confusion matrix



Results



**References:**

1. <https://www.techopedia.com/definition/32066/k-nearest-neighbor-k-nn>
2. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
3. <https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>
4. <https://www.geeksforgeeks.org/k-nearest-neighbours/>