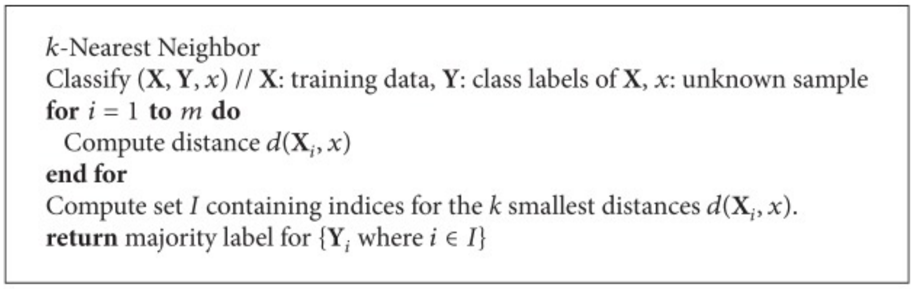
Building a K-Nearest Neighbors Classifier

StartOnAI Machine Learning Tutorial 6:

This is a tutorial which navigates the process of implementing a K-Nearest Neighbors Classifier

1. What is the K-Nearest Neighbors Algorithm?

The K-Nearest Neighbors (KNN) algorithm is a simple example of supervised learning (input being mapped to output). KNN is a classification algorithm and classifies an input as a discrete or categorical output. Other examples of supervised learning classifiers include decision trees, naïve Bayes, and random forest models. Since KNN is an instance-based lazy learning algorithm, it is usually simple and intuitive to train. The fundamental assumption for the KNN algorithm is that datapoints with similar behavior exist in close proximity to each other. If this assumption does not remain fulfilled for a certain sample problem or dataset, the KNN model will not provide statistically significant results. In other words, KNN captures the concept of “closeness” or proximity. Thus, the KNN algorithm is successful in sample datasets where similar outputs cluster together in close proximity. The algorithm works by computing the distances from a specified example to other local examples. The algorithm is known as “K”-Nearest Neighbors since K nearby points are examined. K is a parameter which can be chosen and tuned via iterations of training a KNN algorithm. In most cases, K is an initial arbitrary point. Pseudocode for KNN is given below.

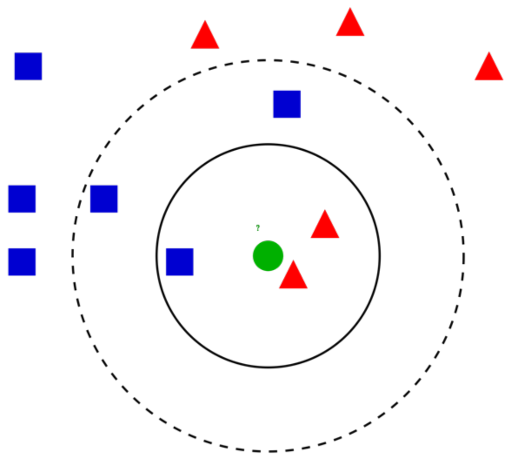


```
k-Nearest Neighbor
Classify ($\mathbf{X}, \mathbf{Y}, x$) // $\mathbf{X}$: training data, $\mathbf{Y}$: class labels of $\mathbf{X}$, $x$: unknown sample
for $i = 1$ to $m$ do
  Compute distance $d(\mathbf{X}_i, x)$
end for
Compute set $I$ containing indices for the $k$ smallest distances $d(\mathbf{X}_i, x)$.
return majority label for $\{\mathbf{Y}_i$ where $i \in I\}$
```

2. How does it physically classify?

Consider a problem where we must classify a specified new example based on the feature map of several labeled examples. A KNN algorithm will begin by calculating the distance to local data points (a pre-selected K nearest neighbors). For the case of two sample neighbors, the KNN must evaluate the distance between our initial unspecified example to the two sample feature vectors ($\vec{v}$,$\vec{w}$). The goal is to identify the class of a new feature vector ($\vec{u}$) based on its distance to the two other feature vectors. The vector, $\vec{u}$, is assigned the class of the feature vector that it was closer to. The two parameters which affect the performance of KNN the most are the value of K and the number of dimensions

(n). A smaller value of K could allow noise to have a major influence on the prediction (majority vote), whereas a large value of K makes the problem more computationally expensive. The common consensus is to give K the value of $\sqrt{N}$ where N is the number of training data points. Thus, the value of K heavily affects performance of the classifier. A typical set-up for a KNN is given below.



3. Apply KNN to Breast Cancer Classification?

We will now apply a KNN classifier for the problem of breast cancer classification. breast cancer tumors can be of two distinguishable types: malignant phase or benign phase. A KNN can be applied to breast cancer since the breast cancer problem fulfills the necessary criterion of proximity. In other words, breast cancer datasets typically see clusters of points with similar behavior. We will use the `Scikit-Learn` Library to fit a KNN to a specific Breast Cancer dataset.

4. Code Walkthrough

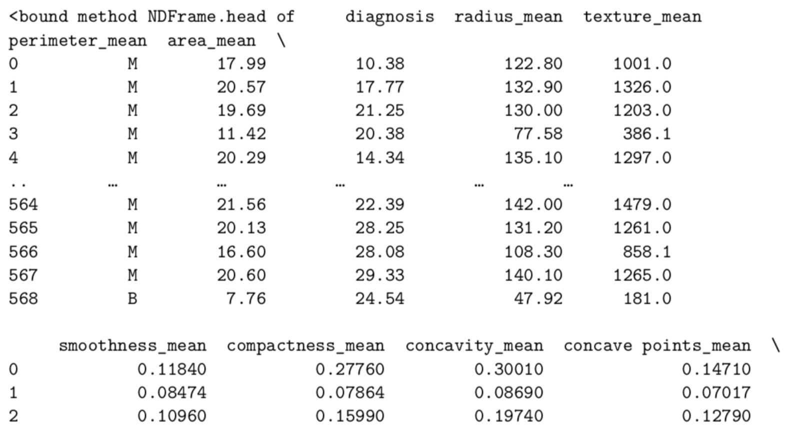
1. We begin by importing the main libraries for this program: `numpy, pandas, matplotlib, and seaborn`. `Numpy` gives us the ability to manipulate arrays, `Pandas` allows us to read in data, and `Matplotlib/Seaborn` are helpful for data visualization

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. We now can import our data in as a `.csv` file. For this program, we used the Wisconsin Breast Cancer dataset, an open source archived dataset with approximately 560 data points with around 330 benign cases and 230 malignant cases. After reading in the dataset, we use specific methods in Python to open it and label categories as diagnosis. At the end of this small script, we import the `train-test-split` module from `Sklearn`.

```
df = pd.read_csv("data 2.csv")
df.replace('?',-99999,inplace=True)
df.drop(['id'],1,inplace=True)

print (df.head)

df.info()

X=np.array(df.drop(['diagnosis'],1))
y=np.array(df['diagnosis'])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.35,␣
 ↪random_state = 42)
```

Output:



```
<bound method NDFrame.head of     diagnosis  radius_mean  texture_mean
perimeter_mean  area_mean  \
0           M        17.99         10.38          122.80     1001.0
1           M        20.57         17.77          132.90     1326.0
2           M        19.69         21.25          130.00     1203.0
3           M        11.42         20.38           77.58      386.1
4           M        20.29         14.34          135.10     1297.0
..        …          …          …             …        …
564         M        21.56         22.39          142.00     1479.0
565         M        20.13         28.25          131.20     1261.0
566         M        16.60         28.08          108.30      858.1
567         M        20.60         29.33          140.10     1265.0
568         B         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0            0.11840           0.27760         0.30010              0.14710
1            0.08474           0.07864         0.08690              0.07017
2            0.10960           0.15990         0.19740              0.12790
```

3. We are now ready to define our dataset and begin the building of our KNN algorithm following pre-processing. We start by defining the value of diagnosis between Malignant and Benign tumors. In this case, 1 is 'M' and 0 is 'Benign'. We apply these values to all cases in our diagnosis category. We then plot our data using `Seaborn` to better understand it. We will graph `radius_mean` versus `texture_mean` since they are key features in our model.

```python
# define Malignant and Benign as 1 and 0 respectively
def diagnosis_value(diagnosis):
    if diagnosis == 'M':
        return 1
    else:
        return 0

df['diagnosis'] = df['diagnosis'].apply(diagnosis_value)
```

```python
# visualization
sns.lmplot(x = 'radius_mean', y = 'texture_mean', hue = 'diagnosis', data = df)
```

```
<seaborn.axisgrid.FacetGrid at 0x1a23941790>
```

4. We now can fit a KNN module to our dataset. We use the `KNeighborsClassifier` from Scikit-Learn and adopt an arbitrary $2/3$ to $1/3$ train-test split. We use the `.fit()` and specify $K = 13$ neighbors. 13 neighbors was chosen after testing for the loss minimum K.

```python
# do train-test split (2/3 to 1/3)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.33, random_state = 42)
```

```python
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 13)
knn.fit(X_train, y_train)
```