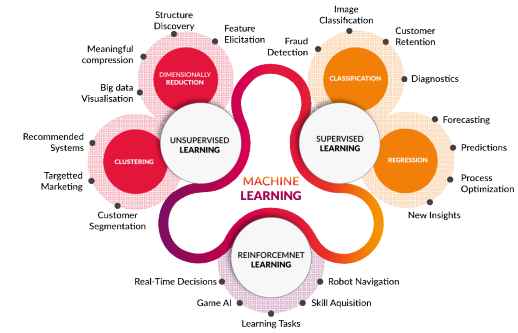
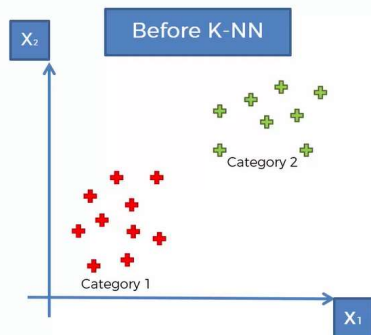


# K-NN Classifier

Machine Learning  
Dr. Adnan Abid



## What K-NN does for you



## How did it do that ?

STEP 1: Choose the number K of neighbors



STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance

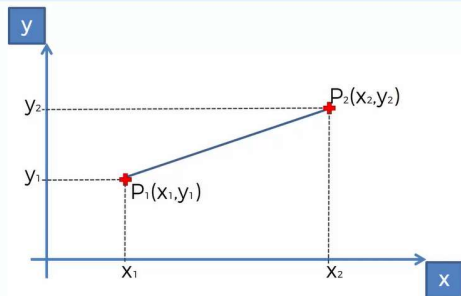


STEP 3: Among these K neighbors, count the number of data points in each category



STEP 4: Assign the new data point to the category where you counted the most neighbors

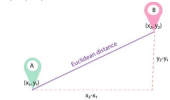
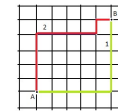
## Euclidean Distance



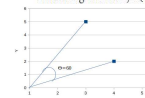
Euclidean Distance between  $P_1$  and  $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

## Similarity Measures

- Minkowski Distance  $\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$
- Manhattan Distance  $\rightarrow p = 1$  in Minkowski Distance (used for high dimensional spaces)  $d = \sum_{i=1}^n |x_i - y_i|$
- Euclidean Distance  $\rightarrow p = 2$  in Minkowski Distance (used for low dimensional spaces)  $d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$
- Hamming Distance  $\rightarrow 11011 \ 10110 \rightarrow$  Apply XOR  $\rightarrow 01101$  and count 1's, in this case Hamming Distance is 3 (used for categorical values)
- Cosine Distance: used for text, collaborative filtering etc.  $1 - \cos(\theta)$

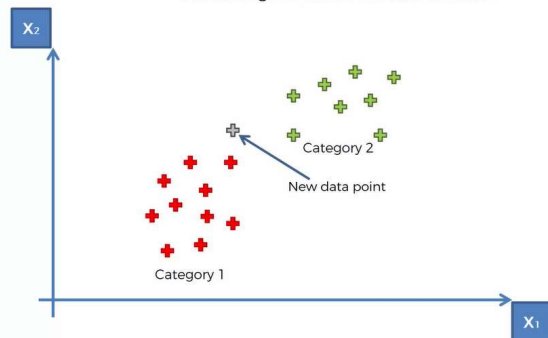


$11011001 \oplus 10011101 = 01000100$ . Since, this contains two 1s, the Hamming distance,  $d(11011001, 10011101) = 2$ .



## K-NN algorithm

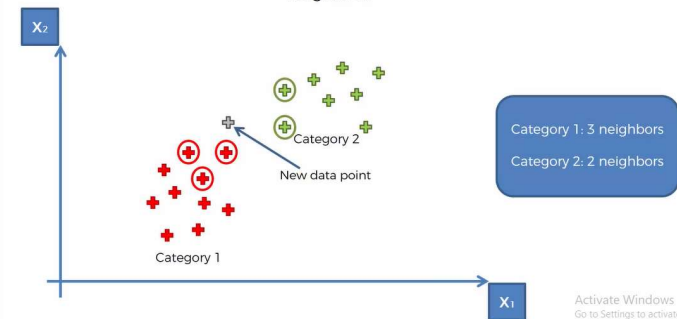
STEP 2: Take the K = 5 nearest neighbors of the new data point, according to the Euclidean distance



Activ  
Go to

## K-NN algorithm

STEP 4: Assign the new data point to the category where you counted the most neighbors



Activate Windows  
Go to Settings to activate Windows

## Sample Example and Implementation

- Purchase SUV or Not
- Features
  - Age
  - Estimated Salary
- Disregard feature
  - User id
  - Gender

	A	B	C	D	E	F
1	User ID	Gender	Age	Estimated Purchased		
2	15624510	Male	19	19000	0	
3	15810944	Male	35	20000	0	
4	15668575	Female	26	43000	0	
5	15601246	Female	27	57000	0	
6	15804022	Male	19	76000	0	
7	15728773	Male	27	58000	0	
8	15598044	Female	27	84000	0	
9	15694829	Female	32	150000	1	
10	15600575	Male	25	33000	0	
11	15727311	Female	35	65000	0	
12	15570769	Female	26	80000	0	
13	15606274	Female	26	52000	0	
14	15746139	Male	20	86000	0	
15	15704987	Male	32	18000	0	
16	15628972	Male	18	82000	0	
17	15697686	Male	29	80000	0	
18	15733883	Male	47	25000	1	
19	15617482	Male	45	26000	1	
20	15704583	Male	46	28000	1	

```

1 # K-Nearest Neighbors (K-NN)
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Social_Network_Ads.csv')
10 X = dataset.iloc[:, 2: 3].values
11 y = dataset.iloc[:, 4].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 # Fitting classifier to the Training set
24 from sklearn.neighbors import KNeighborsClassifier
25 classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
26 classifier.fit(X_train, y_train)
27
28 # Predicting the Test set results
29 y_pred = classifier.predict(X_test)
30

```

Note: fitting on sparse input will override the setting of this parameter, using brute force.

**leaf\_size**  
int, optional (default = 30)  
Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.

**metric**  
string or DistanceMetric object (default = 'minkowski')  
The distance metric to use for the tree. The default metric is minkowski, and with p=2 is equivalent to the standard Euclidean metric. See the documentation of the DistanceMetric class for a list of available metrics.

**p**  
integer, optional (default = 2)  
Power parameter for the Minkowski metric. When p = 1, this is equivalent to using Manhattan distance (l1), and euclidean distance (l2) for p = 2. For arbitrary p, minkowski\_distance (Lp) is used.

**metric\_params**  
dict, optional (default = None)  
Additional keyword arguments for the metric function.

**n\_jobs**  
int, optional (default = -1)  
The number of parallel jobs to run. This parameter is ignored when using the 'brute' method.

The only part where the code will be changed for new classifier

The parameters for KNeighborClassifier  
 n\_neighbors = 5 //the default is 5, it is the number of neighbors we want to check proximity with.  
 Metric = minkowski  
 p = 1 or 2  
 Metric = Minkowski and p = 1 → Manhattan Distance  
 Metric = Minkowski and p = 2 → Euclidian Distance

```

1 # K-Nearest Neighbors (K-NN)
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Social_Network_Ads.csv')
10 X = dataset.iloc[:, 2: 3].values
11 y = dataset.iloc[:, 4].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 # Fitting classifier to the Training set
24 from sklearn.neighbors import KNeighborsClassifier
25 classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
26 classifier.fit(X_train, y_train)
27
28 # Predicting the Test set results
29 y_pred = classifier.predict(X_test)
30
31 # Making the Confusion Matrix
32 from sklearn.metrics import confusion_matrix
33 cm = confusion_matrix(y_test, y_pred)
34

```

Variable explorer

Python console

Import Weib Classifier in

Out [2]:

```

1 # K-Nearest Neighbors (K-NN)
2
3 # Importing the libraries
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 # Importing the dataset
9 dataset = pd.read_csv('Social_Network_Ads.csv')
10 X = dataset.iloc[:, 2: 3].values
11 y = dataset.iloc[:, 4].values
12
13 # Splitting the dataset into the Training set and Test set
14 from sklearn.cross_validation import train_test_split
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
16
17 # Feature Scaling
18 from sklearn.preprocessing import StandardScaler
19 sc = StandardScaler()
20 X_train = sc.fit_transform(X_train)
21 X_test = sc.transform(X_test)
22
23 # Fitting classifier to the Training set
24 from sklearn.neighbors import KNeighborsClassifier
25 classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
26 classifier.fit(X_train, y_train)
27
28 # Predicting the Test set results
29 y_pred = classifier.predict(X_test)
30
31 # Making the Confusion Matrix
32 from sklearn.metrics import confusion_matrix
33 cm = confusion_matrix(y_test, y_pred)
34

```

Variable explorer

Python console

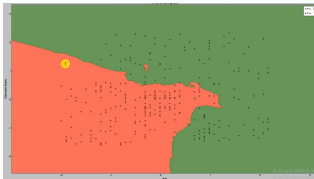
Out [2]:

In [2]: from sklearn.neighbors import KNeighborsClassifier  
 ...: classifier = KNeighborsClassifier(n\_neighbors=5, metric='minkowski', p=2)  
 ...: classifier.fit(X\_train, y\_train)  
 Out [2]:  
 KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',  
 metric\_params=None, n\_jobs=1, n\_neighbors=5, p=2, radius=None)

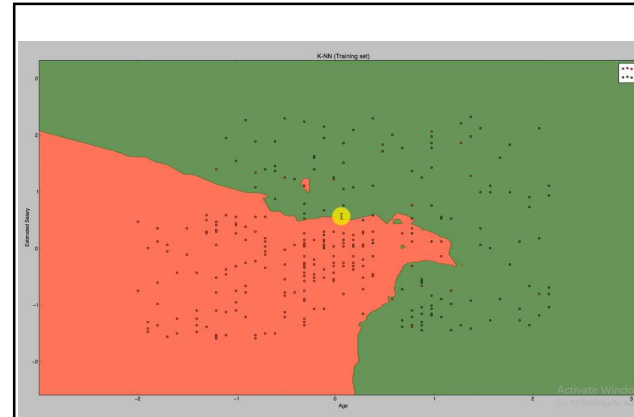
```

35 # Visualising the Training set results
36 from matplotlib.colors import ListedColormap
37 X_set, y_set = X_train, y_train
38 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
39                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
40 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
41             alpha = 0.75, cmap = ListedColormap(['red', 'green']))
42 plt.xlim(X1.min(), X1.max())
43 plt.ylim(X2.min(), X2.max())
44 for i, j in enumerate(np.unique(y_set)):
45     plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
46               c = ListedColormap(['red', 'green'])(i), label = j)
47 plt.title('K-NN (Training set)')
48 plt.xlabel('Age')
49 plt.ylabel('Estimated Salary')
50 plt.legend()
51 plt.show()

```



TRAINING SET PLOT



**TRAINING SET PLOT**  
Boundary is not straight line. K-NN is a non-linear classifier. The regions are very fitted according to the training data set, though we have some incorrect plots. This is because we want to avoid over fitting

```

34
35 # Visualising the Training set results
36 from matplotlib.colors import ListedColormap
37 X_set, y_set = X_train, y_train
38 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
39                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
40 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
41             alpha = 0.75, cmap = ListedColormap(['red', 'green']))
42 plt.xlim(X1.min(), X1.max())
43 plt.ylim(X2.min(), X2.max())
44 for i, j in enumerate(np.unique(y_set)):
45     plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
46               c = ListedColormap(['red', 'green'])(i), label = j)
47 plt.title('Logistic Regression (Training set)')
48 plt.xlabel('Age')
49 plt.ylabel('Estimated Salary')
50 plt.legend()
51 plt.show()

```



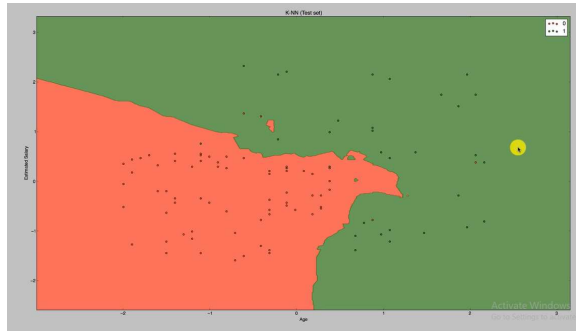
Line 37: X\_set, y\_set are two local variables which help using the same code for plotting the test set data with line X\_set, y\_set = X\_test, y\_test.  
Lines 38-39: Plots pixel points with resolution 0.01 (step variable). Actually, with this code it is using the classifier to plot all the pixel points with different values of Age and Estimated Salary (the two variables) and draws the red and green regions for all the hypothetical values with 0.01 pixel density. Thus the grid is prepared.  
Min -1 and Max +1 are used for both variables Age and Estimated Salary so that the plotted points should have some distance from the grid boundary.  
Lines 40-41: Contour function is used to draw this contour line that is marking the boundary between the two regions.  
Lines 42-43: Plots the limits of the x and y axis. (X -> age, y -> estimated Salary)  
Lines 44-46: The loop plots all the data points in the form of a scatter plot.  
Rest of the lines show labels on x and y axes, legend (red and green for 0 and 1 class), and display the plot.

```

38
39 # Making the Confusion Matrix
40 from sklearn.metrics import confusion_matrix
41 cm = confusion_matrix(y_test, y_pred)
42
43 # Visualising the Training set results
44 from matplotlib.colors import ListedColormap
45 X_set, y_set = X_train, y_train
46 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
47                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
48 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
49             alpha = 0.75, cmap = ListedColormap(['red', 'green']))
50 plt.xlim(X1.min(), X1.max())
51 plt.ylim(X2.min(), X2.max())
52 for i, j in enumerate(np.unique(y_set)):
53     plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
54               c = ListedColormap(['red', 'green'])(i), label = j)
55 plt.title('K-NN (Training set)')
56 plt.xlabel('Age')
57 plt.ylabel('Estimated Salary')
58 plt.legend()
59 plt.show()
60
61 # Visualising the Test set results
62 from matplotlib.colors import ListedColormap
63 X_set, y_set = X_test, y_test
64 X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
65                      np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
66 plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
67             alpha = 0.75, cmap = ListedColormap(['red', 'green']))
68 plt.xlim(X1.min(), X1.max())
69 plt.ylim(X2.min(), X2.max())
70 for i, j in enumerate(np.unique(y_set)):
71     plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
72               c = ListedColormap(['red', 'green'])(i), label = j)
73 plt.title('K-NN (Test set)')
74 plt.xlabel('Age')
75 plt.ylabel('Estimated Salary')
76 plt.legend()
77 plt.show()

```

TEST SET PLOT

**TEST SET PLOT**

The plot shows that most of the test set points have been plotted in the correct regions.

It is just the same as that of the confusion matrix. We had 11 wrong results while using Logistic Regression, where now we have reduced them to 7.

	0	1
0	94	4
1	3	29