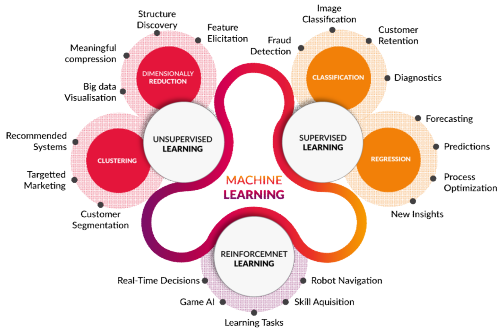
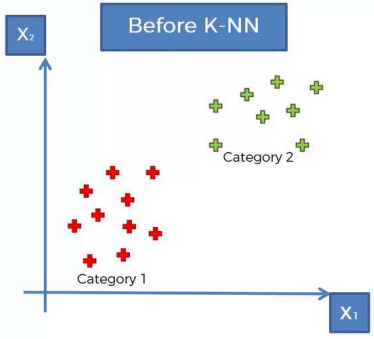
K-NN Classifier

Machine Learning

Dr. Adnan Abid



What K-NN does for you



How did it do that ?

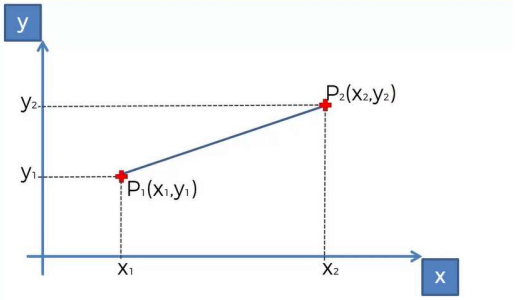
STEP 1: Choose the number K of neighbors

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

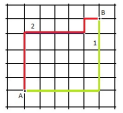
Euclidean Distance

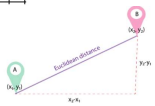


Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

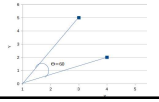
Similarity Measures

- Minkowski Distance $\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$
- Manhattan Distance → p = 1 in Minkowski Distance (used for high dimensional spaces) $d = \sum_{i=1}^{n} |x_i - y_i|$
- Euclidean Distance → 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 → 11011 10110 → Apply XOR → 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)



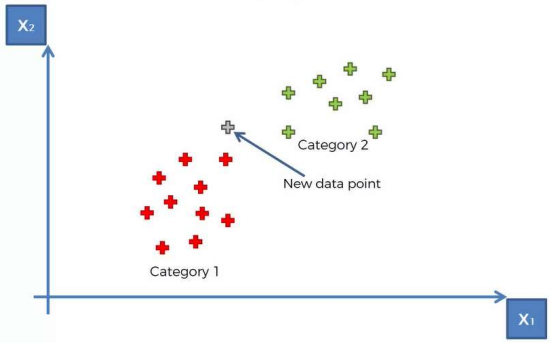


11011001 ⊕ 10011101 = 01000100. Since, this contains two 1s, the Hamming distance, d(11011001, 10011101) = 2.



K-NN algorithm

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



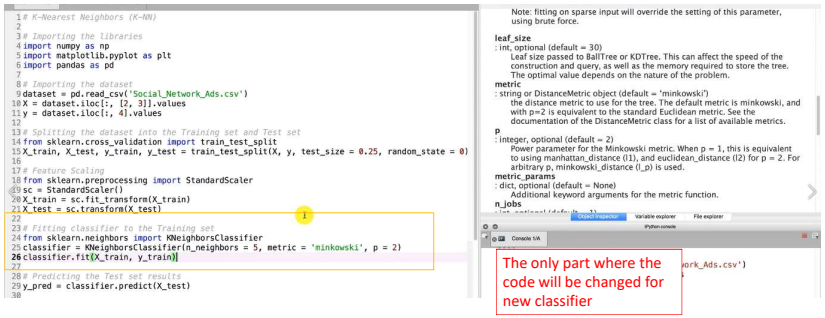
K-NN algorithm

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

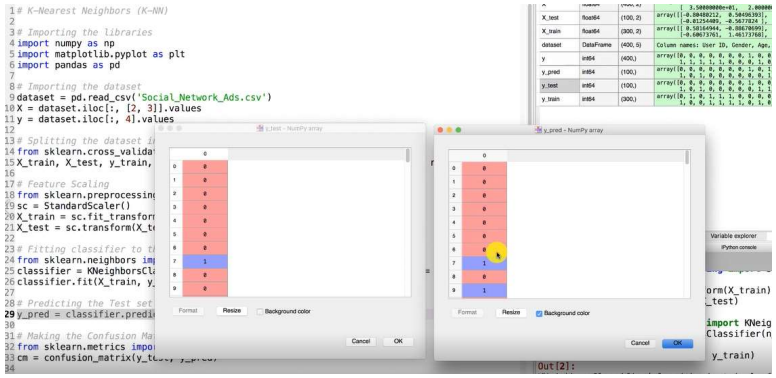
Sample Example and Implementation

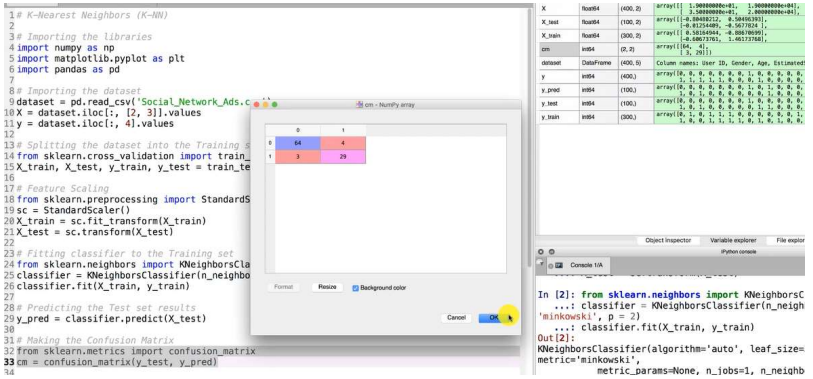
- Purchase SUV or Not
- Features
  - Age
  - Estimated Salary
- Disregard feature
  - User id
  - Gender

	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	15603246	Female	27	57000	0	
6	15804002	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	

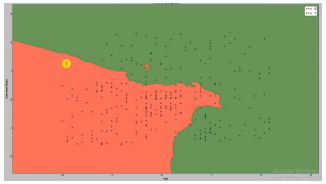


The parameters for KNeighborClassifier
n_neigbors = 5 //the default is 5, it is the number of neighbors we want to check proximity with.
Metric = minkowski
p = 1 or 2
Metric = Minkowski and p = 1 → Manhattan Distance
Metric = Minkowski and p = 2 → Eucledian Distance

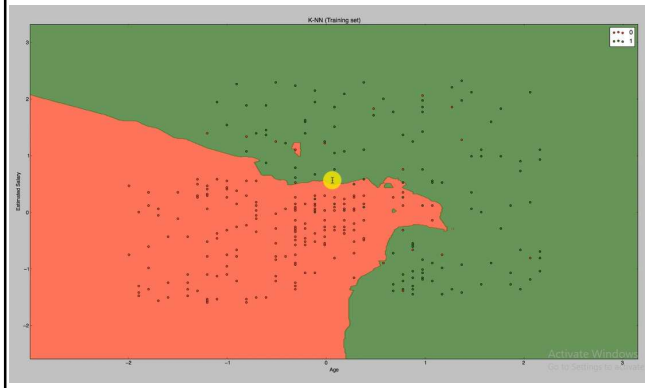




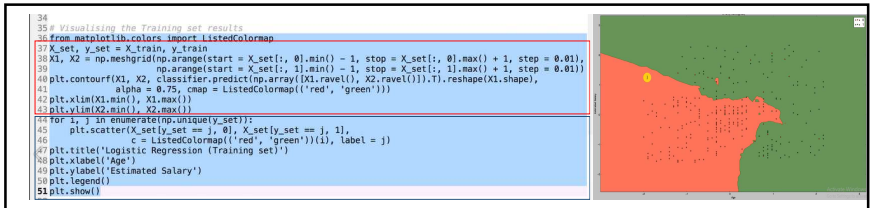
```python
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, s
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, s
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shap
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
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



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

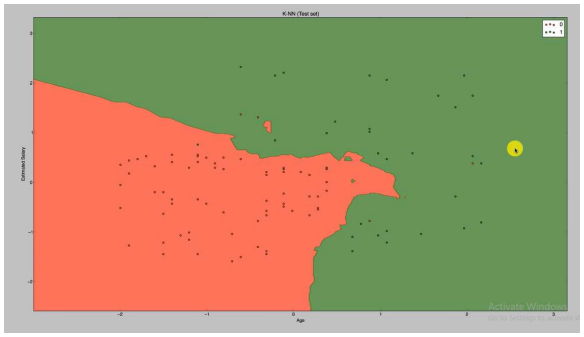
Lines 44-46: The loop plots all the data points in the form of a scatter plot.
Rest of the lines show labels on x and y axes, legend (red and green for 0 and 1 class), and display the plot.

```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, s
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, s
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shap
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, s
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, s
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shap
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
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.