KNN for face recognition

Developed By :

Ouerghi Firas

Belaidi Siwar

Supervised By :

Mme.MARRAKCHI Linda

National Engineering School Of Tunis

November 4, 2019

Plan I

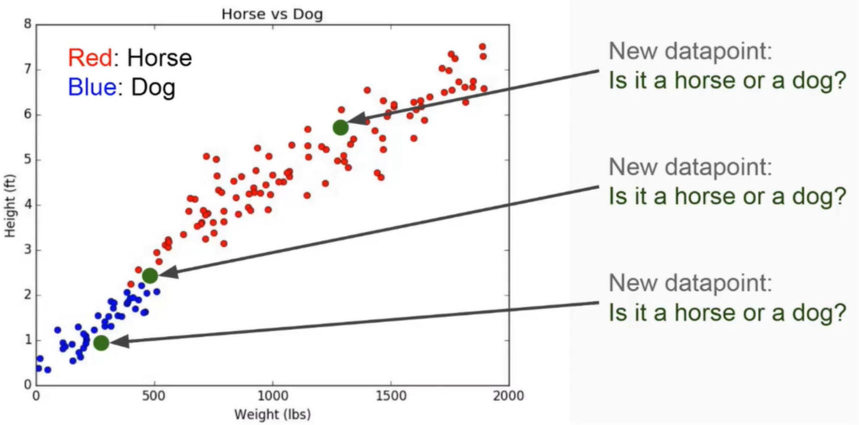
1. Theory and Background
   - Background
   - KNN Algorithm
2. Data Science Life Cycle
   - 1-Business Understanding :
   - 2-Data Mining :
   - 3-Data cleaning :
   - 4-Data exploration :
   - 5-Feature Engineering :
   - 6-Predictive Modeling :
3. Implementation && discussion :
   - Implementation Steps :
   - Evaluate the Model performance
   - Discussion : K-value Selection and probable ways to boost the algorithm

K Nearest Neighbors

K Nearest Neighbors is a classification algorithm that operates on a very simple principle.
It is best shown through an example:

- Imagine we had some imaginary data on Dogs and Horses with heights and weights.

K Nearest Neighbors



New datapoint:
Is it a horse or a dog?

New datapoint:
Is it a horse or a dog?

New datapoint:
Is it a horse or a dog?

K Nearest Neighbors

Training Algorithm:

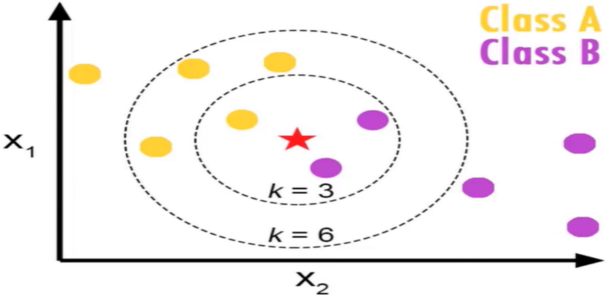
- Store all the Data

Prediction Algorithm:

- Calculate the distance from x to all points in our data
- Sort the points in your data by increasing distance from x
- Predict the majority label of the “k” closest points

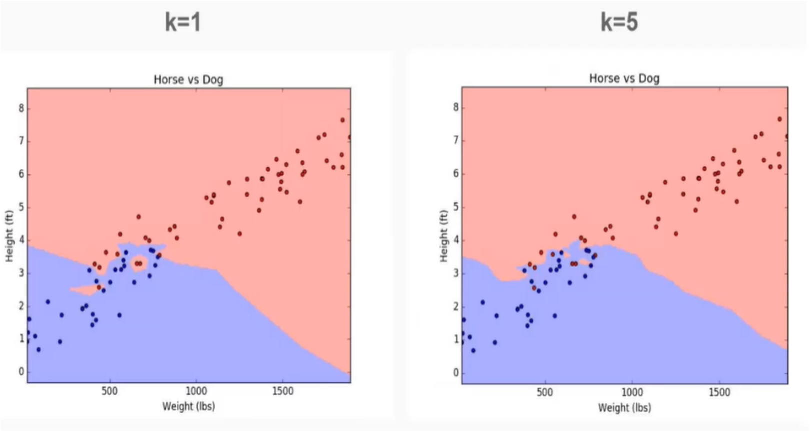
K Nearest Neighbors

Choosing a K will affect what class a new point is assigned to:



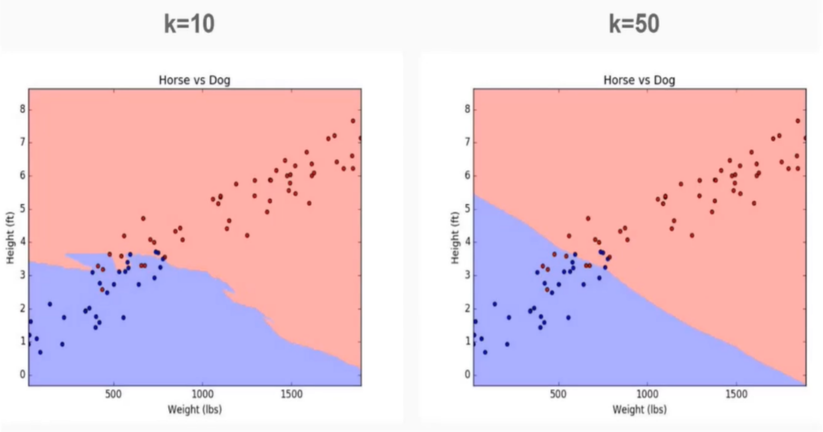
K Nearest Neighbors

Choosing a K will affect what class a new point is assigned to:



K Nearest Neighbors

Choosing a K will affect what class a new point is assigned to:



K Nearest Neighbors

Pros :

- Very simple
- Training is trivial
- Works with any number of classes
- Easy to add more data
- Few parameters: K & Distance Metric

K Nearest Neighbors

Cons :

- High prediction cost (worse for large data sets)
- Not good with high dimensional data
- Categorical features don't work well

1-Business Understanding

01

BUSINESS UNDERSTANDING

Ask relevant questions and define objectives for the problem that needs to be tackled.

1. Banking
2. Access and Security
3. Healthcare
4. Criminal identification
5. Advertising

2-Data Mining :

1. **Gathering pictures from different sources**
   - Using appropriate hardware like (webcam, phone ....)
   - From Social media (Facebook, Linked-in, Instagram...) we could automate this work using a web-scrapping python script.

3-Data cleaning :

03

DATA CLEANING

Fix the inconsistencies within the data and handle the missing values.

1. Resize the images and keep only face pixels using the Cascade Classifier.
2. Eliminate the noise by applying filters.
3. Store the data into a csv File.

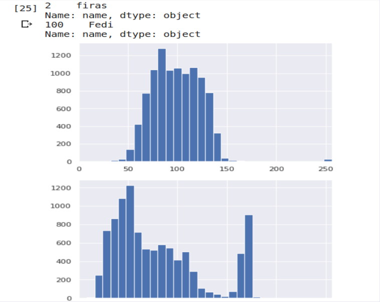
4-Data exploration :

04

DATA EXPLORATION

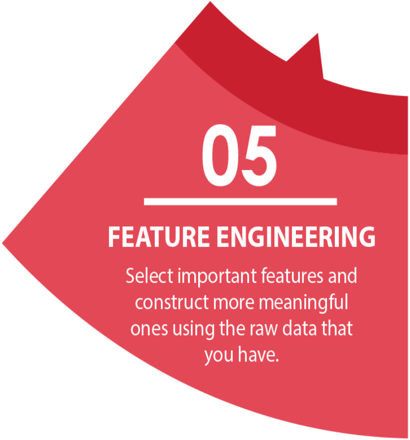
Form hypotheses about your defined problem by visually analyzing the data.

1. Draw pictures histograms to see pixels distribution.



```
[25] 2      firas
     Name: name, dtype: object
     100      Fedi
     Name: name, dtype: object
```

5-Feature Engineering :



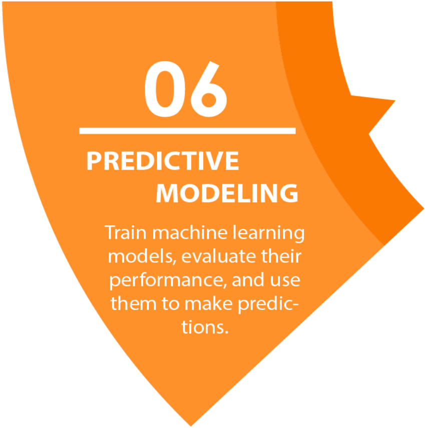
Select important features and construct more meaningful ones using the raw data that you have.

Many Features can be used:

1. Eyes, nose or lips positions
2. Eyes, nose or lips size
3. Face Edge
4. Skin texture

In our case, we have used Texture as a feature.

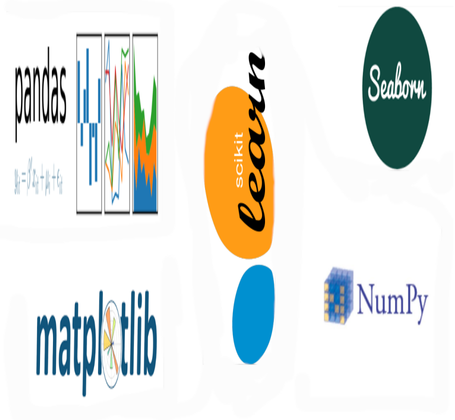
6-Predictive Modeling :


06
PREDICTIVE
MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.

Train the model using KNeighbors Classifier and model.fit function

1. Knn function calling with k=5
   model = KNeighborsClassifier(n_neighbors=4)
2. fdtraining of model
   model.fit(X, Y)

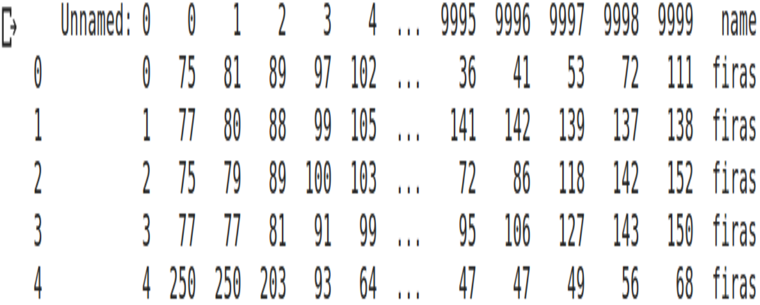
load the needed libraries :



1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt
4. from sklearn import :
    - KNeighborsClassifier
    - confusion_matrix
    - classification_report
    - train_test_split

Load the cleaned data :

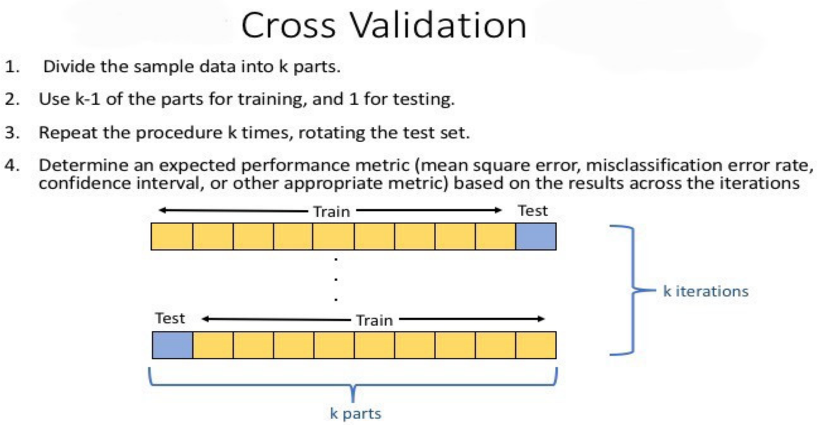
- **data=pd.read_csv("/content/face_data.csv")**
- **print(data.head())**
- **First data split :**
  1. Target=data['name']
  2. features=data.drop('name',axis=1)



```
   Unnamed: 0    0    1    2    3    4  ...  9995  9996  9997  9998  9999   name
0           0   75   81   89   97  102  ...    36    41    53    72   111  firas
1           1   77   80   88   99  105  ...   141   142   139   137   138  firas
2           2   75   79   89  100  103  ...    72    86   118   142   152  firas
3           3   77   77   81   91   99  ...    95   106   127   143   150  firas
4           4  250  250  203   93   64  ...    47    47    49    56    68  firas
```

Cross Validation process

- X_train, X_test, y_train, y_test = train_test_split(features, Target, test_size=0.3, random_state=101)



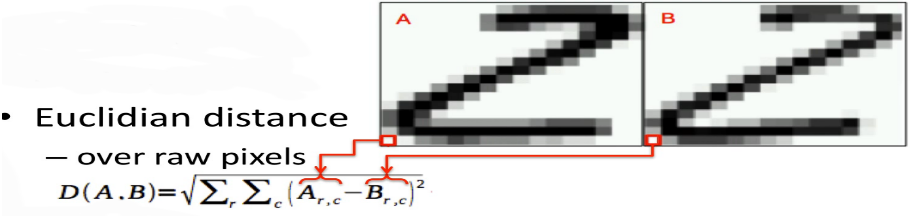
Cross Validation

1. Divide the sample data into k parts.
2. Use k-1 of the parts for training, and 1 for testing.
3. Repeat the procedure k times, rotating the test set.
4. Determine an expected performance metric (mean square error, misclassification error rate, confidence interval, or other appropriate metric) based on the results across the iterations

Create and train the model :

- **KNeighborsClassifier(n_neighbors=5, weights=’uniform’, algorithm=’auto’, p=2, metric=’minkowski’)**
    1. n_neighbors : Number of neighbors to use for the election process.
    2. weights : "uniform" or "distance".
    3. algorithm : ‘auto’,‘kd_tree’, ‘brute’
    4. metric: the distance metric to use.
- **knn.fit(X_train,y_train)**
- **pred=knn.predict(X_test)**

Behind the scene



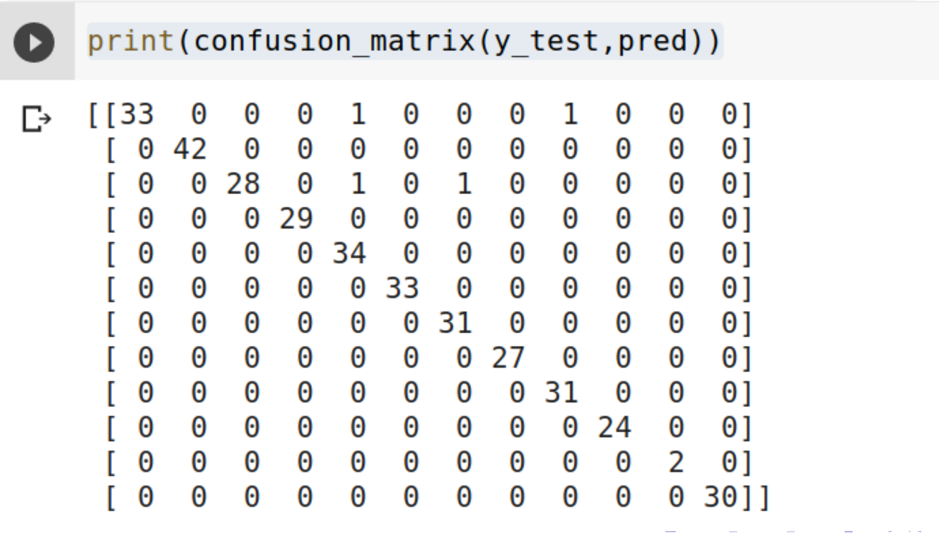
- Euclidian distance
  - over raw pixels

$$D(A.B)=\sqrt{\sum_r \sum_c (\overbrace{A_{r,c}}-\overbrace{B_{r,c}})^2}$$

1. Generate distance vector [d1,d2,d3,............,dn].
2. Sort the distances.
3. Choose the K first distances.
4. Map the distances to their corresponded images.
5. Proceed to the election process.

Confusion matrix :

- **print(confusion_matrix(y_test,pred))**

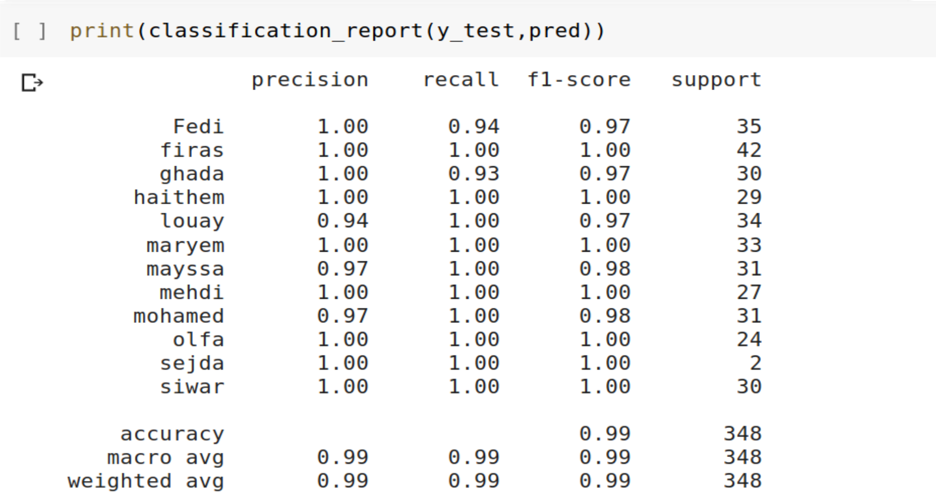


```
print(confusion_matrix(y_test,pred))
```

```
[[33  0  0  0  1  0  0  0  1  0  0  0]
 [ 0 42  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 28  0  1  0  1  0  0  0  0  0]
 [ 0  0  0 29  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 34  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 33  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 31  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 27  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 31  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 24  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  2  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 30]]
```

Classification report :

- **print(classification_report(y_test,pred))**



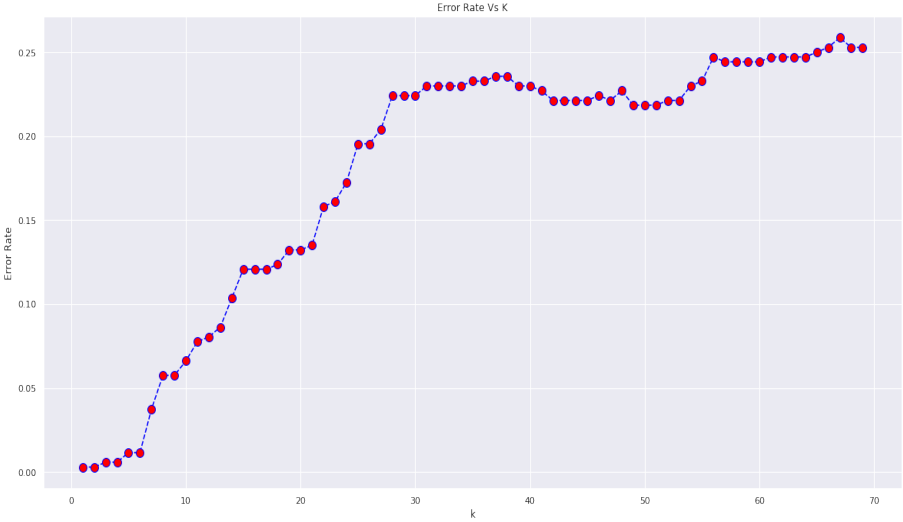
```
[ ]  print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

        Fedi       1.00      0.94      0.97        35
       firas       1.00      1.00      1.00        42
       ghada       1.00      0.93      0.97        30
     haithem       1.00      1.00      1.00        29
       louay       0.94      1.00      0.97        34
      maryem       1.00      1.00      1.00        33
      mayssa       0.97      1.00      0.98        31
       mehdi       1.00      1.00      1.00        27
     mohamed       0.97      1.00      0.98        31
        olfa       1.00      1.00      1.00        24
       sejda       1.00      1.00      1.00         2
       siwar       1.00      1.00      1.00        30

    accuracy                           0.99       348
   macro avg       0.99      0.99      0.99       348
weighted avg       0.99      0.99      0.99       348
```

K-value Selection :

```
erro_rate=[]
for i in range (1,70):
    knn=KNeighborsClassifier(n_neighbors=i,weights='distance')
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    erro_rate.append(np.mean(pred_i != y_test ))
plt.figure(figsize=(20,10))
plt.plot(range(1,70),erro_rate,color='blue',linestyle='–
',marker='o',markerfacecolor='red',markersize=10)
plt.title("Error Rate Vs K")
plt.ylabel("Error Rate")
plt.xlabel('k')
```

K-value Selection :



Probable ways to boost the algorithm :

1. Use the KD Tree algorithm.
2. Choose more relevant features by applying The PCA algorithm.

References

1. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifi
2. https://scikit-learn.org/stable/modules/neighbors.html
3. https://www.youtube.com/watch?v=ZD_tfNpKzHY