# Machine Learning And Data Analysis

Identify characters from Google Street View images

Enrico Castelli S4502687

## 01

## The Challenge

## 01. The Challenge

The challenge is focus on the task of identifying characters from Google Street View images.

It differs from traditional character recognition because the data set contains different character fonts and the background is not the same for all images.

The challenge was taken from the Kaggle website.

kaggle

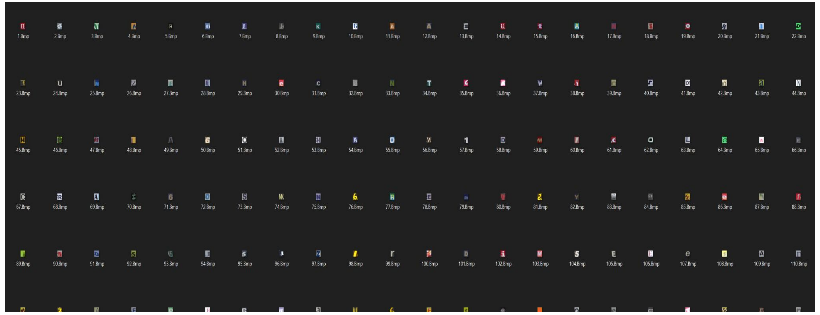
## 02

## Data Preprocessing

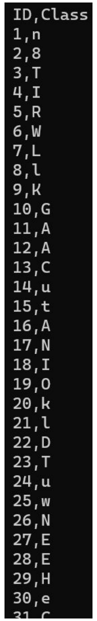
# 02. Data Processing

The training set consisted of:

- **Images** in 20×20 bmp format of upper and lower case letters and numbers from 0 to 9
- **csv file** indicating the correspondence between image and letter or number.


Bmp images of characters taken from Google Street View pictures.

Class label for each picture in the training data.



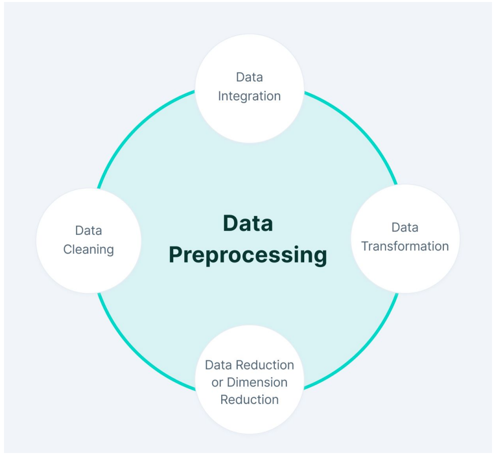
```
ID,Class
1,n
2,8
3,T
4,I
5,R
6,W
7,L
8,l
9,K
10,G
11,A
12,A
13,C
14,u
15,t
16,A
17,N
18,I
19,O
20,k
21,l
22,D
23,T
24,u
25,W
26,N
27,E
28,E
29,H
30,e
```

# 02. Data Processing

I have applied 2 main changes to the dataset to simplify and reduce the execution time of the algorithms:

1. The images were in color (3 channels: RGB) and I turned them to grayscale

2. I reduced the dataset from 6220 elements to 3000 elements by randomly choosing the images to remove



## 03

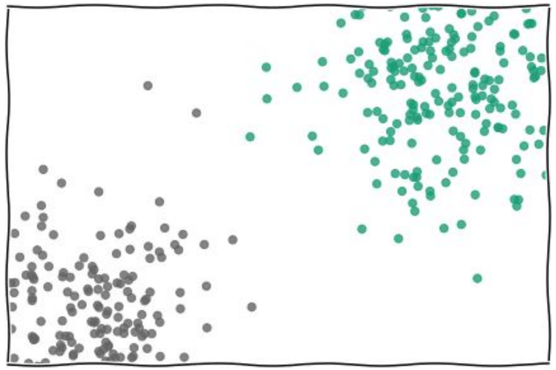
## k-NN Classifier

## 03. k-NN Classifier

k-nearest neighbors algorithm (k-NN) is a simple classification algorithm.

We can use it to assign a class to new data point, so the output is a class membership defined by data point neighbors.

*“Tell me who your neighbors are, and I’ll tell you who you are”.*



## 04

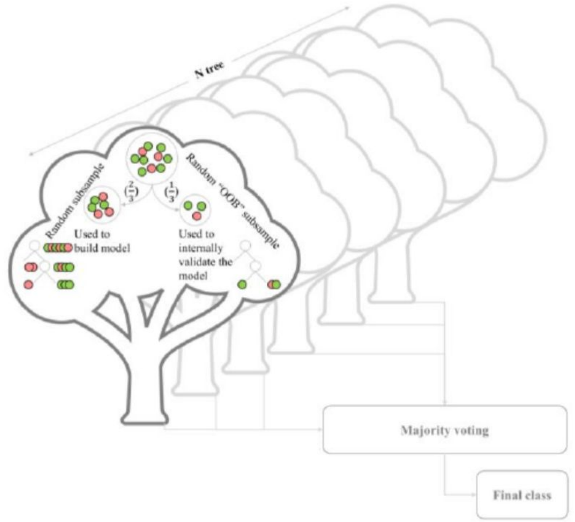
## Random Forest Classifier

## 04. Random Forest Classifier

The Random Forest Classifier is a learning method for classification composed of an **ensemble of decision tree**

Decision trees are classifiers that however have the problem of tending to overfit the data and therefore obtain excellent results in the train set but poor results in the test set

Random forests attempt to solve this problem using a **majority voting** algorithm



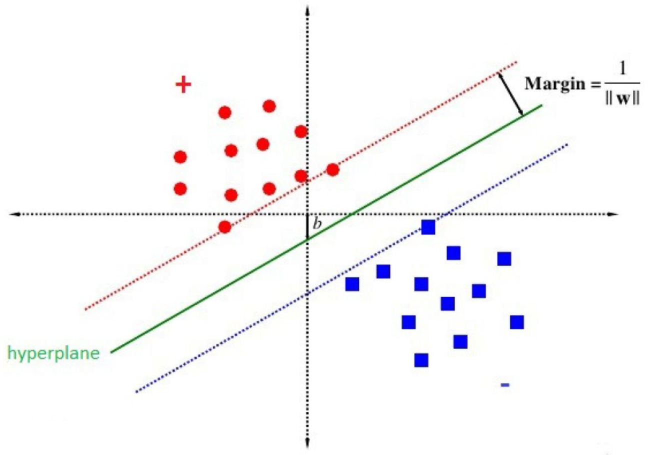
## 05

## Support Vector Classification

## 05. Support Vector Classification

The Support Vector Classification (SVC) is a learning model that analyze data for classification

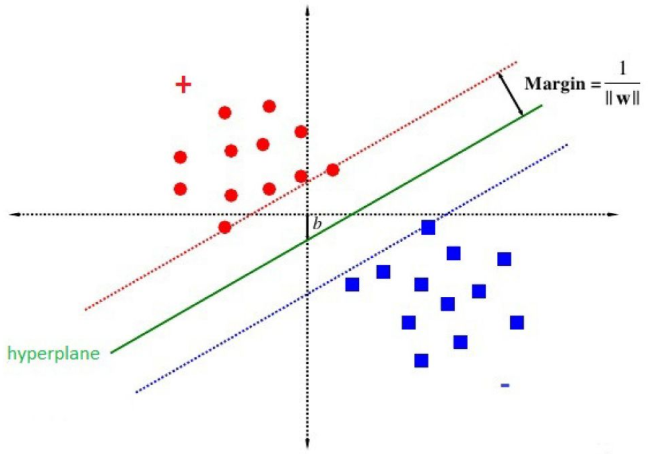
The model is mainly used in contexts where it is necessary to distinguish 2 different classes, in the case in which there are more than 2 classes what happens is that each class is compared with all the others.

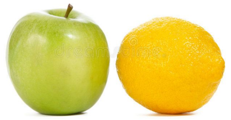


# 05. Support Vector Classification

The model identifies, among all the vectors at its disposal, a variable numbers of vectors (defined by the “C” parameter and the kernel) called support vectors, one per class that represent the characteristics most similar to the other class.

Then a line is drawn between the vectors





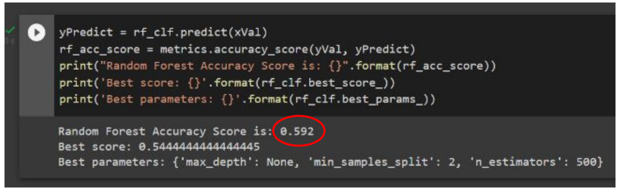
## 06

## Final Results and Considerations

## 06. Final Results and Considerations

I trained my model on the data using the 3 different algorithms discussed above and cross validating the parameters for each algorithm in order to find the best combination of parameters and, after hours of execution, I found that the best algorithm on my "validation" set turned out to be the random forest classifier with an accuracy of 59% and with parameters:

- “max_depth”:None
- “min_samples_split”:2
- “n_estimators”:500



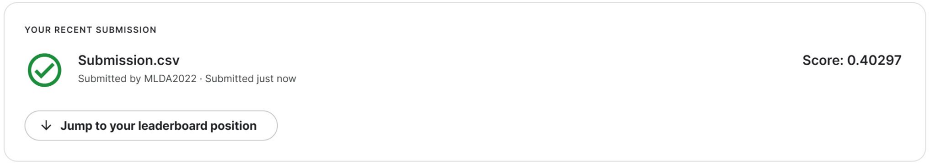
```
yPredict = rf_clf.predict(xVal)
rf_acc_score = metrics.accuracy_score(yVal, yPredict)
print("Random Forest Accuracy Score is: {}".format(rf_acc_score))
print('Best score: {}'.format(rf_clf.best_score_))
print('Best parameters: {}'.format(rf_clf.best_params_))
```

```
Random Forest Accuracy Score is: 0.592
Best score: 0.5444444444444445
Best parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 500}
```

## 06. Final Results and Considerations

After selecting the Random Forest Classifier and the best parameter configuration I performed the predict on the test set, I generated a csv file and after submitting it on Kaggle I obtained that my accuracy was 40%

One of the reasons why the accuracy was lower than the validation set is probably because the test set was found to be more "bad" with the images compared to the training set and in a context in which the random forest classifier already tends to overfit the data resulted in a 10% loss of accuracy


YOUR RECENT SUBMISSION

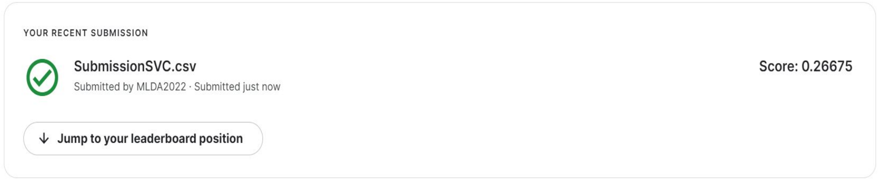
Submission.csv
Submitted by MLDA2022 · Submitted just now

Score: 0.40297

## 06. Final Results and Considerations

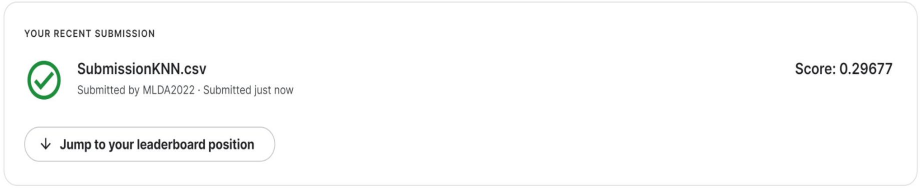
Accuracy is not that low anyway considering that the algorithm has to classify an image that can belong to 62 different classes so my model has really learned something and is not just choosing a random class otherwise it would have a much lower accuracy (less than 2%)

In order to get better results I also tried to run the other algorithms with their better parameters but I got lower accuracy


YOUR RECENT SUBMISSION

SubmissionSVC.csv
Submitted by MLDA2022 · Submitted just now

Score: 0.26675


YOUR RECENT SUBMISSION

SubmissionKNN.csv
Submitted by MLDA2022 · Submitted just now

Score: 0.29677