Results

In this laboratory we tested multiple ML algorithms and implementations.

Below we can see the different results for all the datasets.

Default

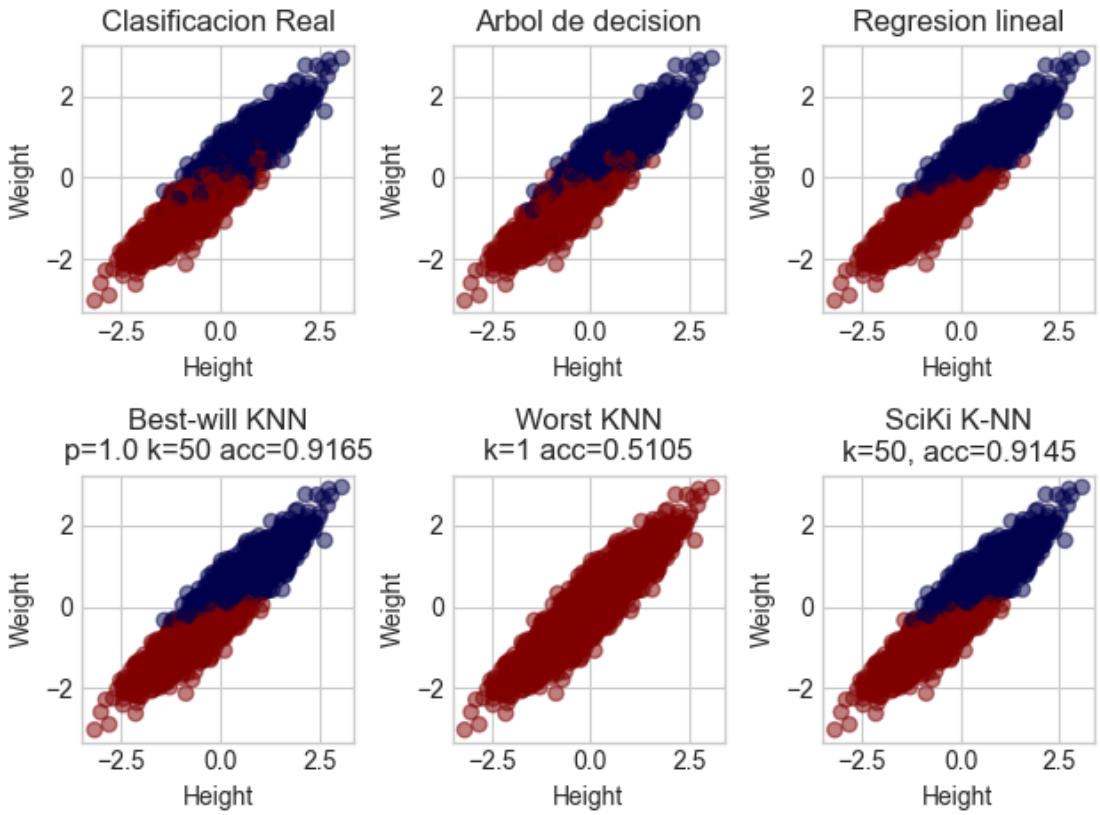
```
Scikit KNN (k=100): 0.975
William KNN best (k=100,p=2.0): 0.976
Decision Tree: 0.949
Logistic Regression: 0.974
William Logistic Regression: 0.974
```

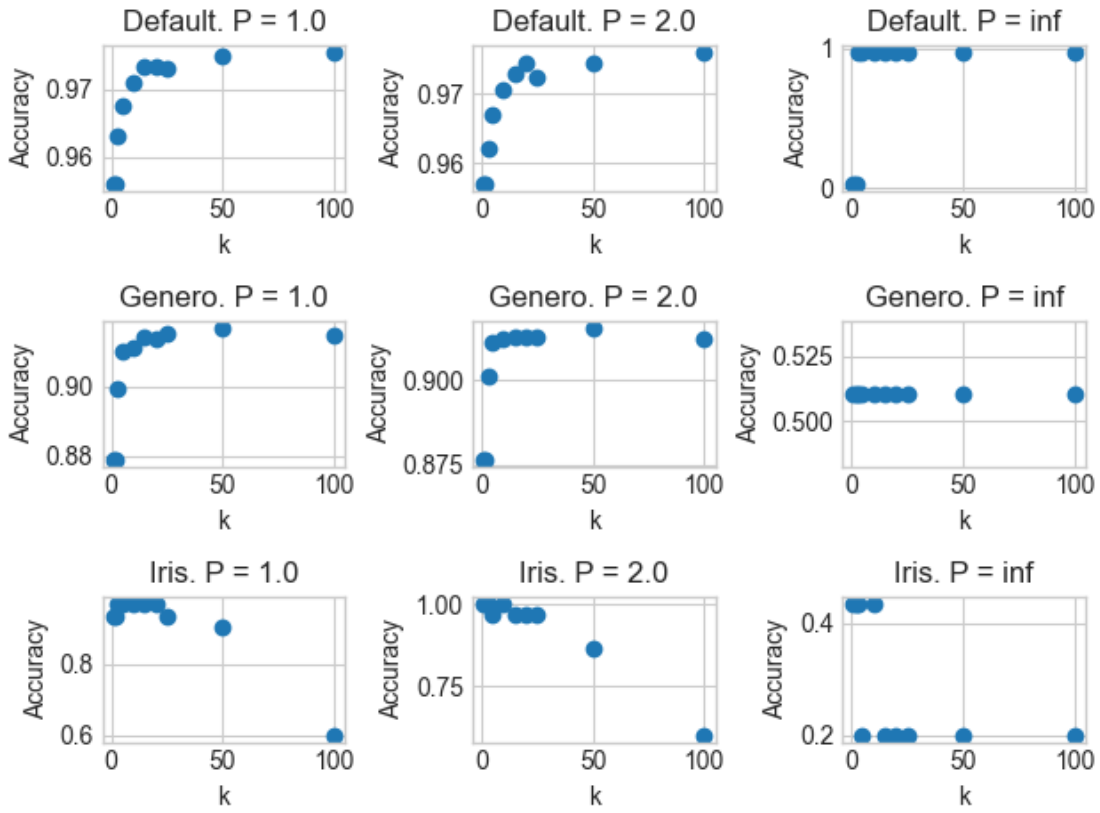
Genero

```
Scikit KNN (k=50): 0.9145
William KNN best (k=50,p=1.0): 0.9165
Decision Tree: 0.873
Logistic Regression: 0.9135
William Logistic Regression: 0.9135
```

Iris

```
Scikit KNN (k=1): 1.0
William KNN best (k=1,p=2.0): 1.0
Decision Tree: 0.9666666666666667
Logistic Regression: 0.9666666666666667
William Logistic Regression: 0.9666666666666667
```


Clasificacion Real
Arbol de decision
Regresion lineal
Best-will KNN
p=1.0 k=50 acc=0.9165
Worst KNN
k=1 acc=0.5105
SciKi K-NN
k=50, acc=0.9145
Weight
Height
2
0
−2
−2.5
0.0
2.5



Conclusions and reflections

- If the dataset gets larger, it is very difficult to train because the time it takes increases a lot.
- This is because KNN is a memory algorithm that checks/bruteforces every point with every other point.
- I would rather us a different algorith because, precisions are very similar and training time is way lower
- To make training faster, I implemented a multi-core KNN and seemded to work faster.
- The advantages that I see in using KNN is that it is very easy to implement and also very easy to understand, and despite that, it is very accurate.
- Regarding the second chart Accuracy vs K, I couldn't find any patter on which K is better, i think it depends on the dataset, however for Genero and Iris, P=Inf was the worst.

References:


```
 - https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
 - https://github.com/arseniyturin/logistic-regression
 - https://medium.com/@martinpella/logistic-regression-from-scratch-in-python-
124c5636b8ac
```

```
- https://medium.com/@lope.ai/logistic-regression-from-scratch-in-python-d1e9dd51f85d
- https://towardsdatascience.com/how-to-build-knn-from-scratch-in-python-5e22b8920bd2
```