Mathematics for Big Data course
Davide Rigoni project

ISTRUTTORE: Isabel Serra
iserra@crm.cat

Homework

Consider any points cloud you may be interested in. Examples:

- http://graphics.stanford.edu/data/3Dscanrep/
- https://archive.ics.uci.edu/ml/index.php
- Any other you are interested in

Enumerate one (or more) objectives and analyze the data with any of the techniques:

- Trees
- NN
- Random Forest

Write a report with the conclusions: answering the objective(s) and describing the validation procedure

Using this environment, provide a compressed file including:

- An explanation of the software you have used.
- An explanation of the data you have used.
- The code you have written to analyse the data.
- The data set or a link to the data set.
- The conclusions: which information does the technique you have used provide you.

Deadline: June 15th, 2018. Attach file or download link to iserra@crm.cat

Software e files

For this project i used *python* with the help of the *sklearn* library.

The following four files include the code that i used for the analysis:

1. *wineTree.py*: wine dataset with tree model
2. *wineNN.py:* wine dataset with neural network
3. *exoTree.py*: exoplanet dataset with tree model
4. *exoNN.py*: exoplanet dataset with neural network

Data

At the beginning i started with the dataset **wine** present in the sklearn library, but after i wanted to tried with a more complicated dataset. At the end i apply the same techniques to the real case dataset **Exoplanet Hunting in Deep Space** present in the famous website Kaggle (https://www.kaggle.com).

What is ***Kaggle***? ***Kaggle*** is a platform for predictive modelling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and users. This crowdsourcing approach relies on the fact that there are countless strategies that can be applied to any predictive modelling task and it is impossible to know beforehand which technique or analyst will be most effective.

Dataset Wine

Explanation

The wine dataset is a multi-class classification dataset in which we want to predict the kind of wine based on the other features like alcohol, magnesium, and so on. There is only three different kind of wine ([0,1,2])

Data Format

Here's the list of all the features:

- 1) Alcohol
- 2) Malic acid
- 3) Ash
- 4) Alcalinity of ash
- 5) Magnesium
- 6) Total phenols
- 7) Flavanoids
- 8) Non flavanoid phenols
- 9) Proanthocyanins
- 10)Color intensity
- 11)Hue
- 12)OD280/OD315 of diluted wines
- 13)Proline

All of the features are numeric and there are 178 elements.

Solution

The objective of this work is to find the best model that is able to predict with more accuracy the kind of wine. So this is a multi-class classification problem.

I applied two different techniques in order to find the best way to fit this data. The first technique is the classification tree and the second one is the neural network.

I steps i have followed to get the current results are:

1. Preprocessing of the data
2. Finding the best model

Preprocessing

Tree

In this case there is no need to elaborate the data before building the tree, because it is still able to fit well the data. In fact we have the same result even if we scale the data.

Neural Networks

In this case i have standardized all the features values because the neural network is sensible to the scale of the data. In order to do that i used the *scale* function present in the *package sklearn.preprocessing.*

Finding the best model

In both the case, in order to find the best structure for the models, i fit more than one time the model with different initial params, using the leave one out cross validation.

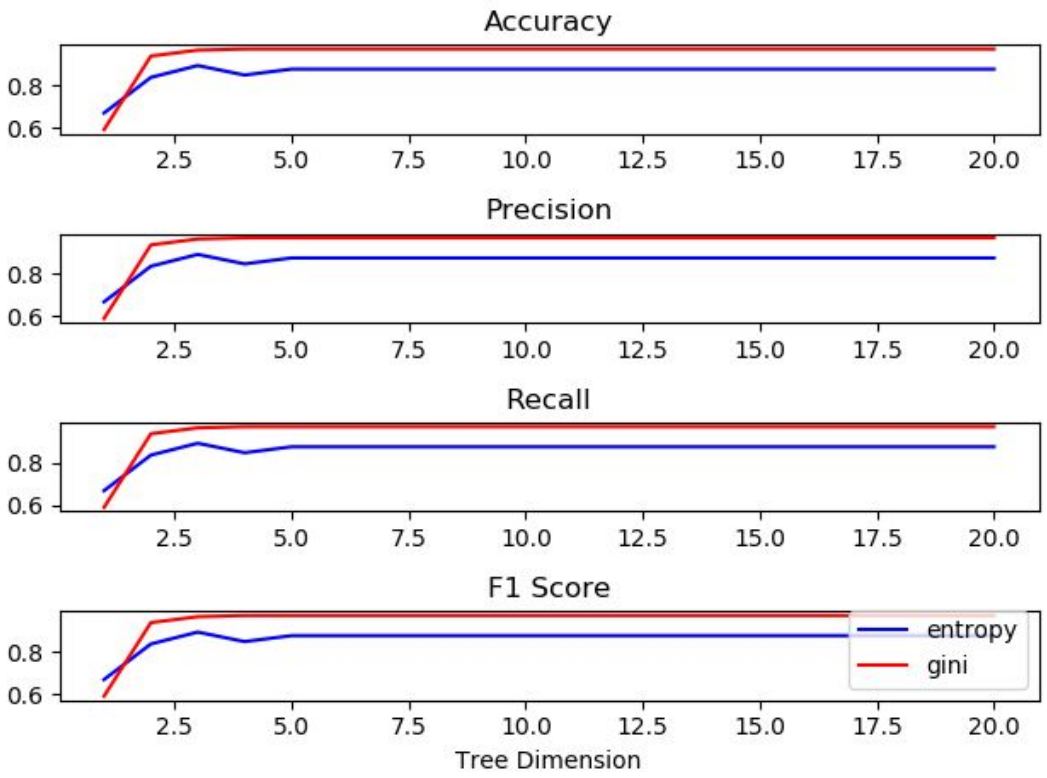
The metrics I used to compare different models among them are:

1. Accuracy
2. Precision using the macro average
3. Recall using the macro average
4. F1 score using the macro average

Tree

I tried different models with increasing max depth value from 1 to 20 and using *gini* and *entropy* criteria. The results are the following:

```
Max Accuracy Entropy:  0.8932584269662921  at depth:  3
Max Accuracy Gini:  0.9719101123595506  at depth:  4
Max Precision Entropy:  0.8932584269662921  at depth:  3
Max Precision Gini:  0.9719101123595506  at depth:  4
Max Recall Entropy:  0.8932584269662921  at depth:  3
Max Recall Gini:  0.9719101123595506  at depth:  4
Max F1 Score Entropy:  0.8932584269662921  at depth:  3
Max F1 Score Gini:  0.9719101123595506  at depth:  4
```



How we can see we are able to reach the best results using the *gini* criteria, and we are able to have the best values with the max depth 4. There is no need to use a three with higher depth than 4. We have reached same values for all the gini scores: the 97.19%.

Neural Network

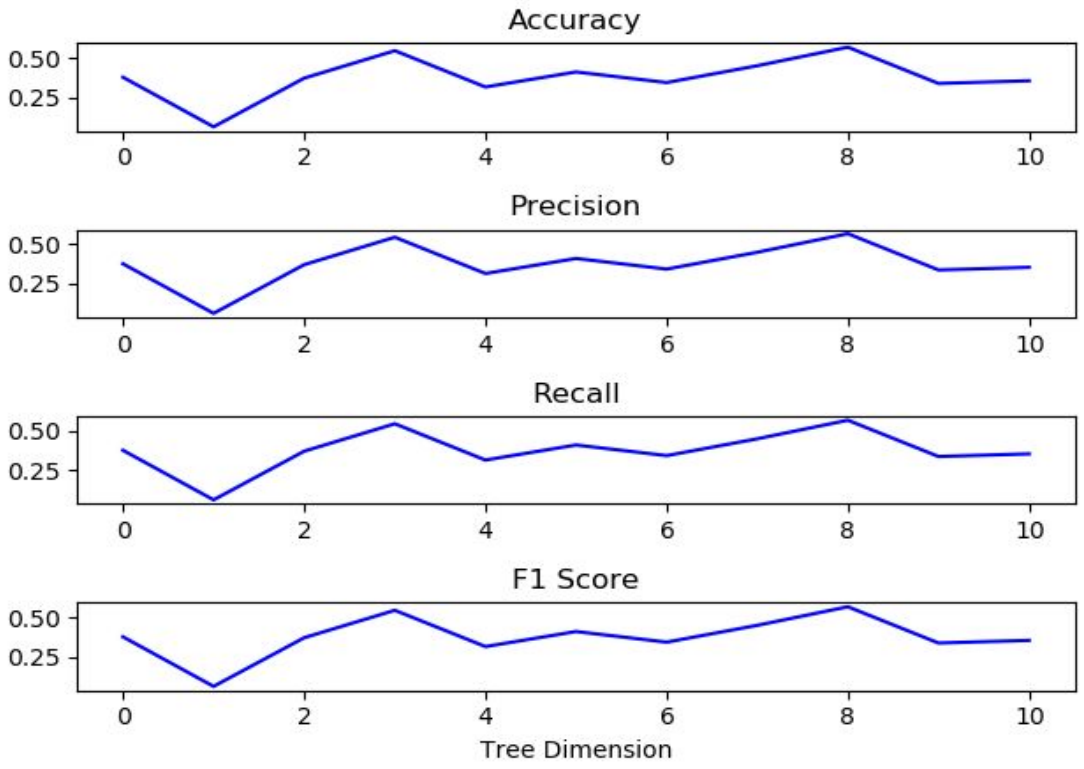
The neural network model is made with:

- 13 input neurons (one for each feature)
- 1 output neuron (classification problem)
- 1 hidden layer

In order to find the best model that is able to fit this data, i tried to see if incrementing the number of neurons from 3 to 13 it was able to perform better.

In fact we can see that the best result is reached with 11 (3+ 8) neurons in the hidden layer.

```
Max Accuracy:  0.5674157303370787  at iteration:  8
Max Precision:  0.5674157303370787  at iteration:  8
Max Recall:  0.5674157303370787  at iteration:  8
Max F1 Score:  0.5674157303370787  at iteration:  8
```



However the result is not so good and the tree is able to classify better the data. This probably is due to the large amount of data that the neural network need to learn properly the weights. In this case we don’t have enough data. A solution could be resampling but in this way it can lead to overfitting the data.

Conclusion

Looking to the best model founded for both the tree model and the neural network model, we see that is better to use the tree in this case. This because it allow us to reach more accuracy in predicting the data.

Dataset Exoplanet

Explanation

The data describe the change in flux (light intensity) of several thousand stars. Each star has a binary label of `2` or `1`. 2 indicated that that the star is confirmed to have at least one exoplanet in orbit; some observations are in fact multi-planet systems.

As you can imagine, planets themselves do not emit light, but the stars that they orbit do. If said star is watched over several months or years, there may be a regular 'dimming' of the flux (the light intensity). This is evidence that there may be an orbiting body around the star; such a star could be considered to be a 'candidate' system. Further study of our candidate system, for example by a satellite that captures light at a different wavelength, could solidify the belief that the candidate can in fact be 'confirmed'

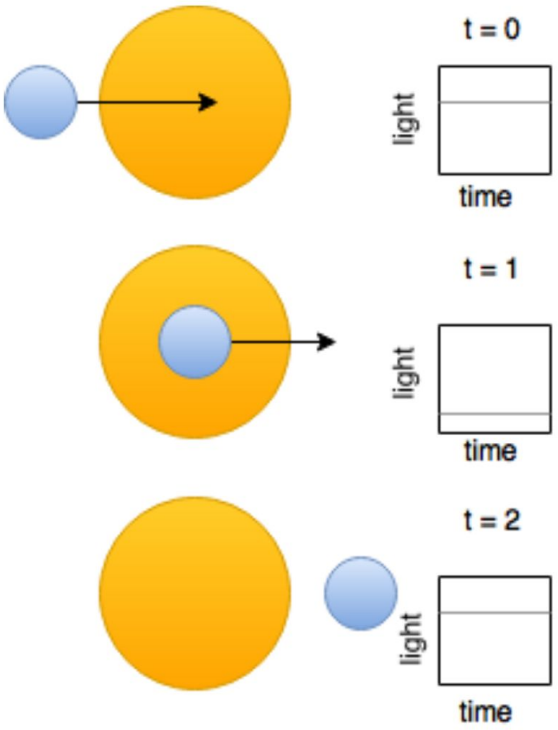


Figure 1: Explanation about more or less light

In the Figure 1 , a star is orbited by a blue planet. At t = 1, the starlight intensity drops because it is partially obscured by the planet, given our position. The starlight rises back to its original value at t = 2. The graph in each box shows the measured flux (light intensity) at each time interval. At the end we want to predict if given a star there is an exoplanet.

Data Format

Trainset (exoTrain.cvs):

- 5087 rows or observations.
- 3198 columns or features.
- Column 1 is the label vector. Columns 2 - 3198 are the flux values over time.
- 37 confirmed exoplanet-stars and 5050 non-exoplanet-stars.

Testset (exoTest.cvs): ***Used only to test the final result***

- 570 rows or observations.
- 3198 columns or features.
- Column 1 is the label vector. Columns 2 - 3198 are the flux values over time.
- 5 confirmed exoplanet-stars and 565 non-exoplanet-stars.

Acknowledgements

The data presented here are cleaned and are derived from observations made by the NASA Kepler space telescope. The Mission is ongoing - for instance data from Campaign 12 was released on 8th March 2017. Over 99% of this dataset originates from Campaign 3. To boost the number of exoplanet-stars in the dataset, confirmed exoplanets from other campaigns were also included.

To be clear, all observations from Campaign 3 are included. And in addition to this, confirmed exoplanet-stars from other campaigns are also included.

The datasets were prepared late-summer 2016.

Campaign 3 was used because 'it was felt' that this Campaign is unlikely to contain any undiscovered (i.e. wrongly labelled) exoplanets.

NASA open-sources the original Kepler Mission data and it is hosted at the Mikulski Archive. After being beamed down to Earth, NASA applies de-noising algorithms to remove artefacts generated by the telescope. The data - in the .fits format - is stored online. And with the help of a seasoned astrophysicist, anyone with an internet connection can embark on a search to find and retrieve the datafiles from the Archive.

The cover image is copyright © 2011 by Dan Lessmann

Solution

The objective of this work is to find the best model that is able to predict with more precision if at least one exoplanet exist around the star. So this is a binary classification problem.

I applied two different techniques in order to find the best way to fit this data. The first technique is the classification tree and the second one is the neural network.

I steps i have followed to get the current results are:

1. Preprocessing of the data
2. Find the best model with 5-fold cross validation
3. Fit the best model with all the training set
4. Predict test set and result

Preprocessing

Tree

In this case there is no need to elaborate the data before building the tree, because it is still able to fit well the data. In fact the results are the same.

Neural Networks

In this case i have standardized all the features values because the neural network is sensible to the scale of the data. In order to do that i used the *scale* function present in the *package sklearn.preprocessing.*

Finding the best model

In both the case, in order to find the best structure for the models, i executed more than one times them with different initial params, using the 5-fold cross validation for trees and test set validation for the neural network.
I used the 5-fold cross validation and the test set validation because fit the models require a lot of time and computational power. (More than 8 hours in order to find the best neural network)

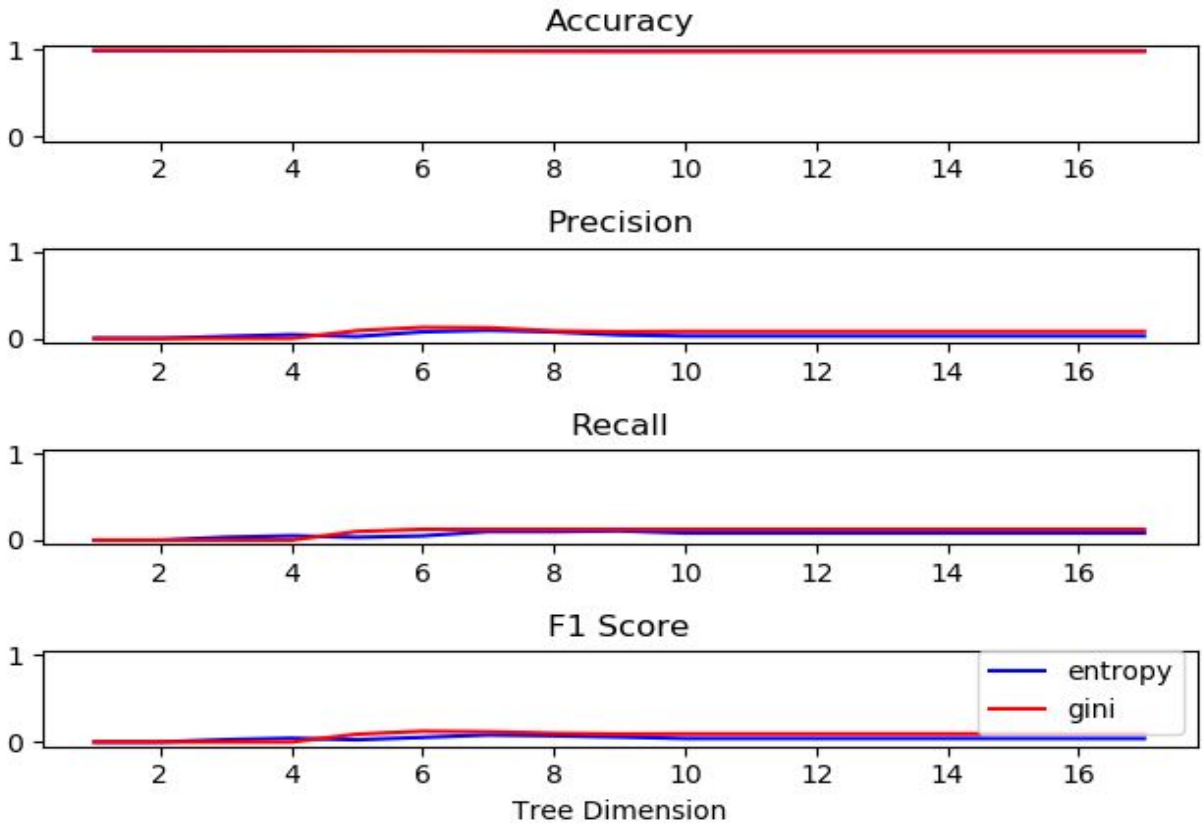
The metrics I used to compare different models among them are:

1. Accuracy
2. Precision of class labeled 2 : to see how the model perform with this class
3. Recall of class labeled 2 : to see how the model perform with this class
4. F1 score of class labeled 2 : to see how the model perform with this class

Tree

I try different models with increasing max depth value from 1 to 17 and using gini and entropy criteria. The results are the following:

```
Max Accuracy Entropy:  0.9915474265579454  at depth:  1
Max Accuracy Gini:  0.992726594842491  at depth:  1
Max Precision Entropy:  0.09318181818181819  at depth:  7
Max Precision Gini:  0.1269047619047619  at depth:  6
Max Recall Entropy:  0.10833333333333332  at depth:  9
Max Recall Gini:  0.125  at depth:  6
Max F1 Score Entropy:  0.08186274509803922  at depth:  7
Max F1 Score Gini:  0.12444444444444444  at depth:  6
```



As we can see, it is strange that with the most simple tree we have the model with the best accuracy. This is due to the data set, there are 5087 elements and only 37 have an exoplanet. So if we make only one model classifying every elements to the class labeled 1, we could still reach an accuracy of 0,9927265578926676 ((1087 - 37) / 5087).

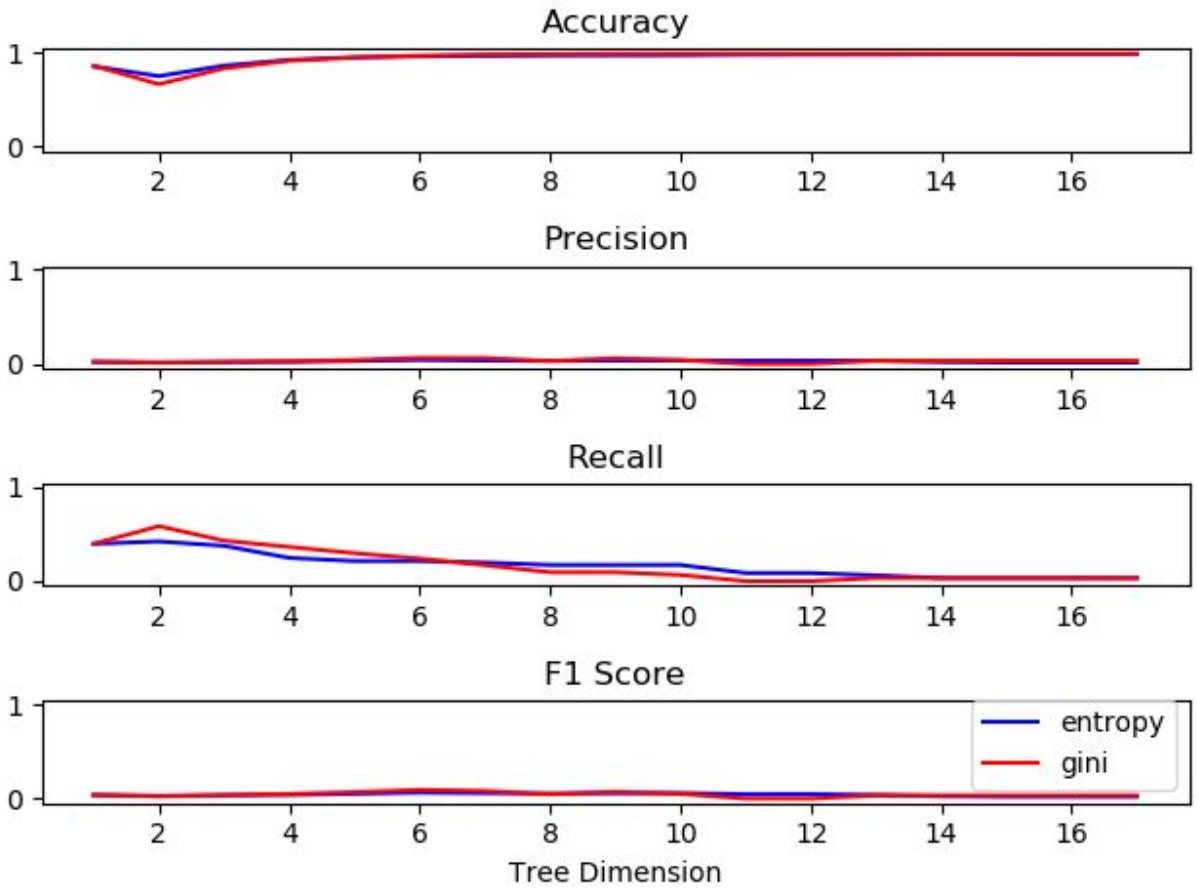
In order to understand better what is happen, i used others metrics focused on the class labeled 2. Thanks to these, we are able to see that gini usually performs better than entropy and that the tree reach the best precision at depth 6.

The problems of having unbalanced datasets is very common and every solution has a consequence:

1. Delete items from the largest class: with this solution we could lose very important details in the data which could improve a lot the model.
2. Sampling from the smallest class: with this solution we are copying exactly the elements in the dataset and this could lead to overfitting.

In this case i decided to apply the second strategy and in order to do that, i used the balanced parameter in the tree model. The following are the results obtained with the model where weights are automatically adjust inversely proportional to class frequencies in the input data as *n_samples / (n_classes * np.bincount(y))*.

```
Max Accuracy Entropy:  0.9807353574691927  at depth:  17
Max Accuracy Gini:  0.986436087494905  at depth:  13
Max Precision Entropy:  0.041105836486641435  at depth:  6
Max Precision Gini:  0.06294237207077953  at depth:  6
Max Recall Entropy:  0.4226190476190476  at depth:  2
Max Recall Gini:  0.5845238095238096  at depth:  2
Max F1 Score Entropy:  0.0645567731356804  at depth:  6
Max F1 Score Gini:  0.09211846229995051  at depth:  6
```



In this case we see that the model with the best accuracy is that one founded with the highest depth, but however the other metrics are very bad and the first tree model was better both in accuracy and in precision.

Neural Network

The neural network model is made with:

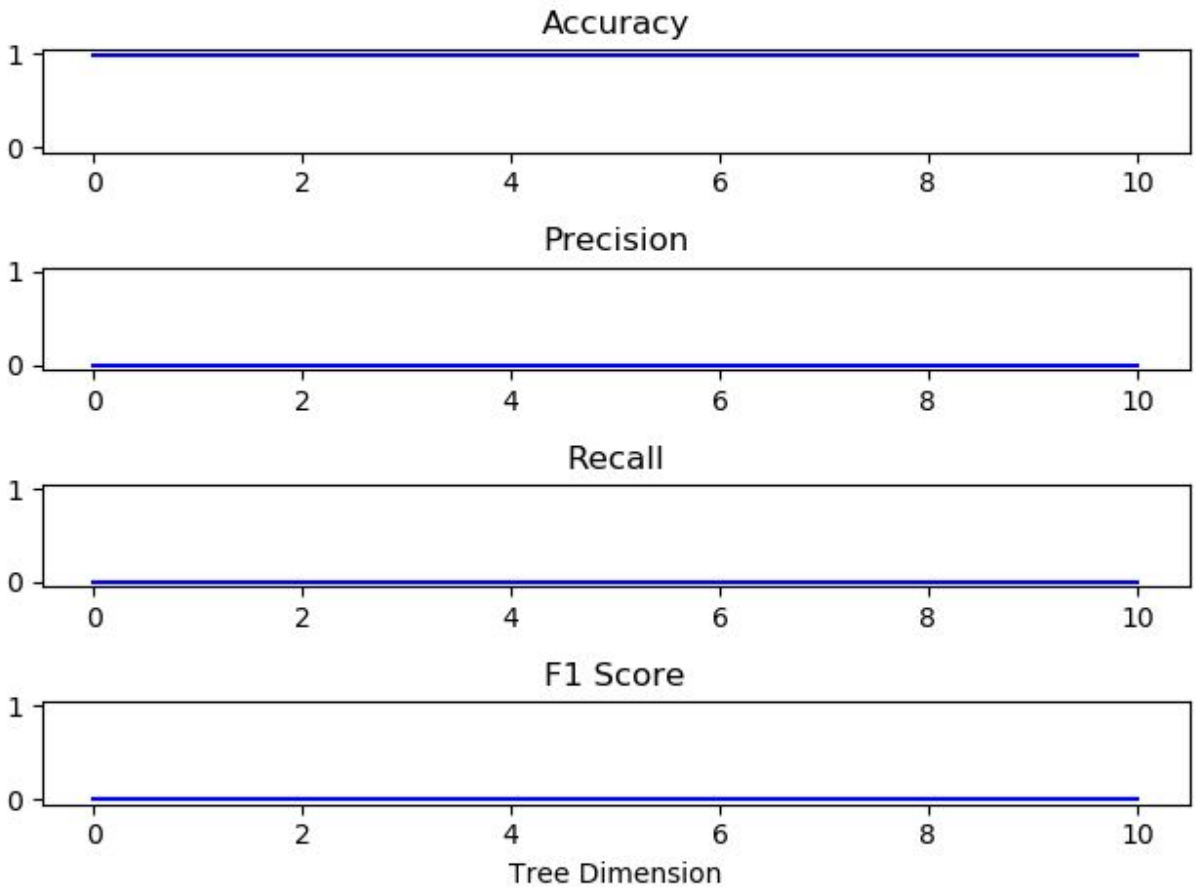
- 3197 input neurons (one for each feature)
- 1 output neuron (classification problem)
- 2 hidden layers: for many practical problems, there is no reason to use any more than one hidden layer. However in order to be sure to represent even the most complicated function i used 2 layers. Notice that using more layers and neurons we need more data to fit the model.

But how much neurons in this two hidden layers? Keeping in mind that the best number of neurons range from 1 to the same number of neurons in the input layer (there is no theoretical proof, but it seems to be so from empirical results) i tried the following configuration. In order to look the performance i divided the data in training set (77%) and test set (33%):

- For the first hidden layer: from 1066 to 2066 with 10 iterations (a step of 100). This goes from more or less ⅓ to ⅔ of the input neurons numbers.
- For the second hidden layer: from 355 to 655 with 10 iterations (a step of 30). This goes from more or less ⅓ to ⅔ of the previously layers neurons numbers.

After a lot of hours of computation i had the following results:

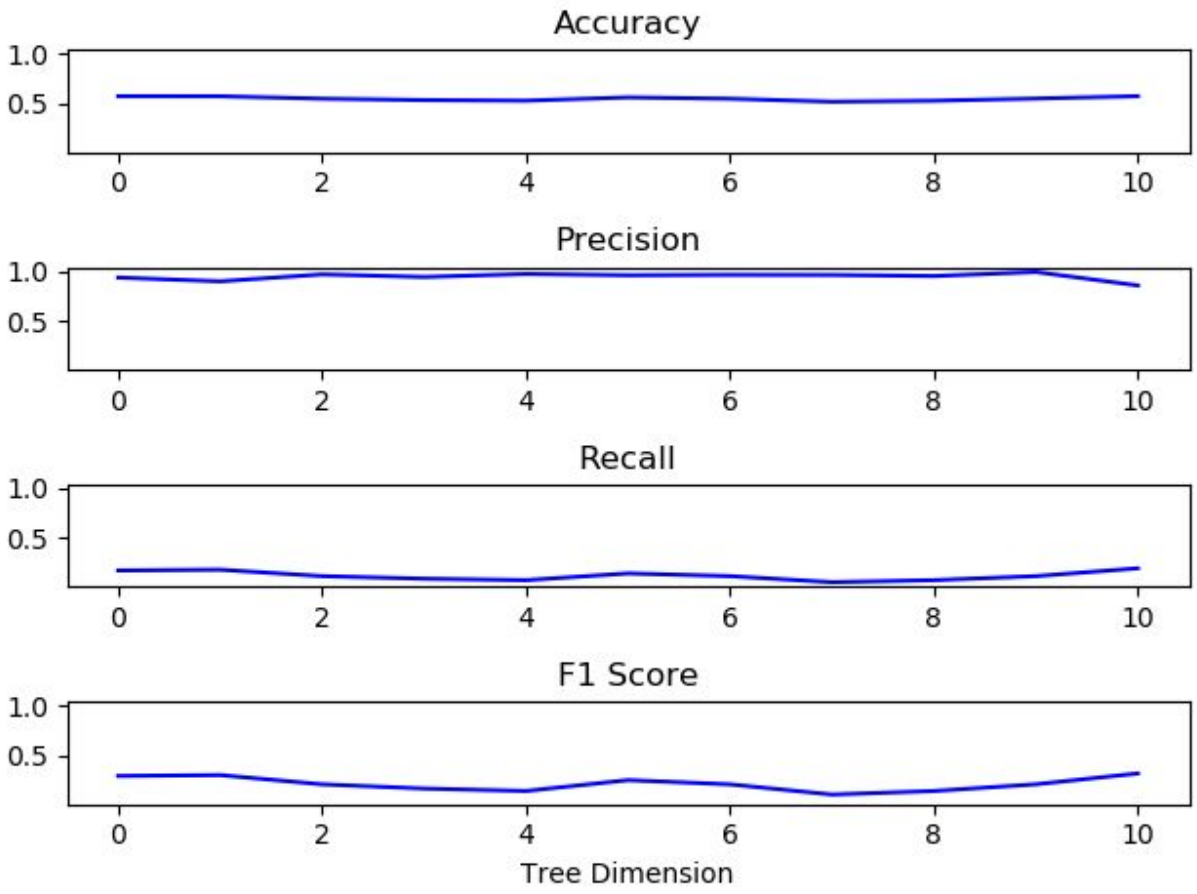
```
Max Accuracy:  0.9922572960095295  at iteration:  0
Max Precision:  0.0  at iteration:  0
Max Recall:  0.0  at iteration:  0
Max F1 Score:  0.0  at iteration:  0
```



The data Neural network is not good, it has the same precision of the naive model that predict all with label 1.

For the same reason i explained before in the case of the tree, we have the unbalanced data problem. This time i resample the elements labeled 2 in order to have a final balanced dataset where there are 50% of examples belonging to class 1 and 50% belonging to class 2.

```
Max Accuracy:  0.5721572157215722  at iteration:  10
Max Precision:  0.9901477832512315  at iteration:  9
Max Recall:  0.1965962441314554  at iteration:  10
Max F1 Score:  0.3196564885496183  at iteration:  10
```



In this case we was able to reach a good precision, Recall and F1 score, but the accuracy is not so good. Moreover we see that the best result is obtained for most of them with iteration equal to 10, it means that the hidden neurons for the hidden layers are 2066 for the first one and 655 for the second one.

Fit the best model and predict

Tree

From the step before we learn that the best model is that one that doesn't weight the differently the elements and that use the gini criteria with max depth 6.

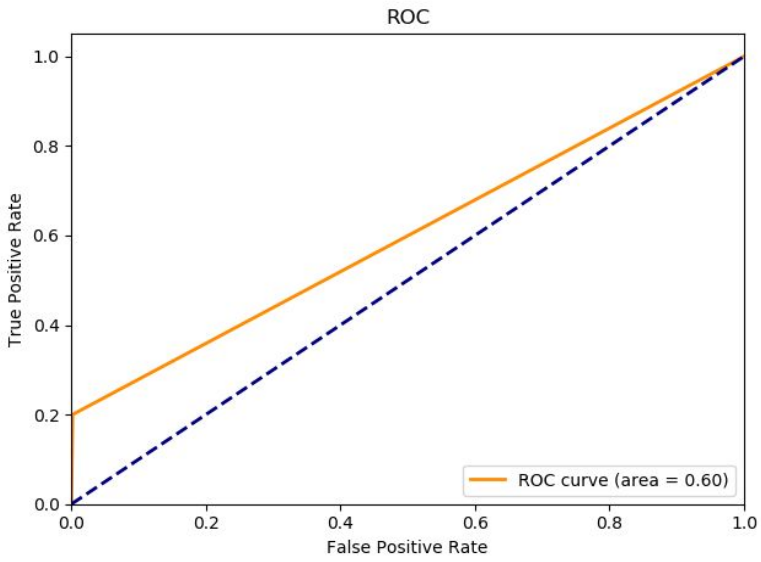
These are the result when it is trained with all the train set and it is used to predict the test set:

```
Accuracy:  0.9912280701754386
```

```
Precision:  0.5

Recall:  0.2

F1 score:  0.28571428571428575
```



How we can see, the results is not good. We obtained the same accuracy that we could have with the model that classify everything with label 1. In fact:

(570-5)/570 = 0.9912280702

But in this case we was able to obtain more precision, exactly 0.5 that means that we was able to classify right half of the elements with label equal to 2. However the recall is still low.

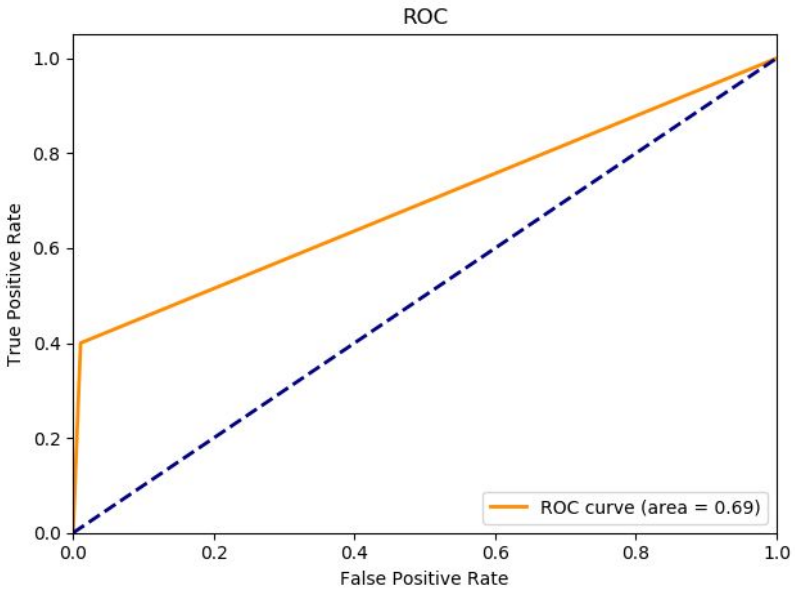
The following result belong to the model without a limit in the depth, and we see that it worse than the one founded previously. However it has more area under the curve.

```
Accuracy:  0.9842105263157894

Precision:  0.25

Recall:  0.4

F1 score:  0.3076923076923077
```



Neural Network

Using the neural network that perform better with resampling, we have the following result:

```
Accuracy:  0.19473684210526315
Precision:  0.006521739130434782
Recall:  0.6
F1 score:  0.012903225806451613
```

As expected it doesn't perform well in the test set, because it overfits the data due to the resampling.

Using the best model found without resampling the result is the following:

```
Accuracy:  0.9912280701754386
Precision:  0.0
Recall:  0.0
F1 score:  0.0
```

This because trained neural network is classifying everything with label 1.

Conclusion

In conclusion applying only these two techniques we are not able to find a model that is able to predict well the presence of an exoplanet. There is the need to use different approach for this complicated dataset, and searching online from the best results (99.8% accuracy) the solution could be implemented a more effective preprocessing and a convolutional neural network.