

Random Forest Learning

Dermateology data set

Kuldeep sharma | Machine Learning | 5 March 2019

Introduction

The project is based on Classification based Supervised Learning Problem of the diagnosis of erythemato-squamous diseases through Random forest Learning.

Database used: - Title: Dermatology Database

Relevant Information:

This database contains 34 attributes, 33 of which are linear

valued and one of them is nominal.

The differential diagnosis of erythemato-squamous diseases is a real

problem in dermatology. They all share the clinical features of

erythema and scaling, with very little differences. The diseases in

this group are psoriasis, seboreic dermatitis, lichen planus,

pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris.

Usually a biopsy is necessary for the diagnosis but unfortunately

these diseases share many histopathological features as

well. Another difficulty for the differential diagnosis is that a

disease may show the features of another disease at the beginning

stage and may have the characteristic features at the following

stages.

Patients were first evaluated clinically with 12 features.

Afterwards, skin samples were taken for the evaluation of 22

histopathological features. The values of the histopathological

features are determined by an analysis of the samples under a

microscope.

In the dataset constructed for this domain, the family history

Feature has the value 1 if any of these diseases has been observed

in the family, and 0 otherwise. The age feature simply represents

the age of the patient. Every other feature (clinical and

histopathological) was given a degree in the range of 0 to 3. Here, 0

indicates that the feature was not present, 3 indicates the largest

amount possible, and 1, 2 indicate the relative intermediate values.

The names and id numbers of the patients were recently

removed from the database.

Number of Instances: 366

Number of Attributes: 34

Attribute Information:

-- Complete attribute documentation:

Clinical Attributes: (take values 0, 1, 2, 3, unless otherwise indicated)

1: erythema

2: scaling

3: definite borders

4: itching

5: koebner phenomenon

6: polygonal papules

7: follicular papules

8: oral mucosal involvement

9: knee and elbow involvement

10: scalp involvement

11: family history, (0 or 1)

34: Age (linear)

Histopathological Attributes: (take values 0, 1, 2, 3)

12: melanin incontinence

13: eosinophils in the infiltrate

14: PNL infiltrate

15: fibrosis of the papillary dermis

16: exocytosis

17: acanthosis

18: hyperkeratosis

19: parakeratosis

20: clubbing of the rete ridges

21: elongation of the rete ridges

22: thinning of the suprapapillary epidermis

23: spongiform pustule

24: munro microabcess

25: focal hypergranulosis

26: disappearance of the granular layer

27: vacuolisation and damage of basal layer

28: spongiosis

29: saw-tooth appearance of retes

30: follicular horn plug

31: perifollicular parakeratosis

32: inflammatory monoluclear inflitrate

33: band-like infiltrate

Missing Attribute Values: 8 (in Age attribute). Distinguished with '?'.

Class Distribution:

Database: Dermatology

Class code: Class: Number of instances:

1 psoriasis 112

2 seboreic dermatitis 61

3 lichen planus 72

4 pityriasis rosea 49

5 cronic dermatitis 52

6 pityriasis rubra pilaris 20

RANDOM FOREST

**Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it’s simplicity and the fact that it can be used for both classification and regression tasks.**

Random Forest is a supervised learning algorithm. Like you can already see from its name, it creates a forest and makes it somehow random. The forest it builds, is an ensemble of Decision Trees, most of the time trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

**Important Hyperparameters:**

The Hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster.

**1. Increasing the Predictive Power**

Firstly, there is the **n\_estimators** hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

Another important hyperparameter is **max\_features**, which is the maximum number of features Random Forest considers splitting a node. Sklearn provides several options.

The last important hyper-parameter in terms of speed, is **min\_sample\_leaf**. This determines, like its name already says, the minimum number of leafs that are required to split an internal node.

**2. Increasing the Models Speed**

The **n\_jobs** hyperparameter tells the engine how many processors it can use. If it has a value of 1, it can only use one processor. A value of “-1” means that there is no limit.

**random\_state** makes the model’s output replicable. The model will always produce the same results when it has a definite value of random\_state and if it has been given the same hyperparameters and the same training data.

Lastly, there is the **oob\_score** (also called oob sampling), which is a random forest cross validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance.

**Advantages and Disadvantages:**

An advantage of random forest is that it can be used for both regression and classification tasks and that it’s easy to view the relative importance it assigns to the input features.

Random Forest is also considered as a very handy and easy to use algorithm, because it’s default hyperparameters often produce a good prediction result. The number of hyperparameters is also not that high and they are straightforward to understand.

One of the big problems in machine learning is overfitting, but most of the time this won’t happen that easy to a random forest classifier. That’s because if there are enough trees in the forest, the classifier won’t overfit the model.

The main limitation of Random Forest is that a large number of trees can make the algorithm to slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. In most real-world applications the random forest algorithm is fast enough, but there can certainly be situations where run-time performance is important and other approaches would be preferred.

And of course, Random Forest is a predictive modeling tool and not a descriptive tool.

CODE

#Linear Algebra

import numpy as np

#data processing

import pandas as pd

#Algorithms

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

df= pd.read\_csv("C:/Users/sharm/OneDrive/Desktop/Machine Learning pro/prodata.data", names = ["erythema", "scaling", "definite borders", "itching", "koebner phenomenon", "polygonal papules", "follicular papules", "oral mucosal involvement", "knee and elbow involvement", "scalp involvement", "family history", "melanin incontinence", "eosinophils in the infiltrate", "PNL infiltrate", "fibrosis of the papillary dermis", "exocytosis", "acanthosis", "hyperkeratosis", "parakeratosis", "clubbing of the rete ridges", "thinning of the suprapapillary epidermis", "spongiform pustule", "Age", "class"])

print(df.info())

print(df.describe())

print(df.head(15))

#Dealing with the missing values

print("Missing values")

print((df["Age"] == "?").sum())

df[["Age"]] = df[["Age"]].replace("?", np.NaN)

# drop rows with missing values

df.dropna(inplace=True)

# summarize the number of rows and columns in the dataset

print(df.shape)

print("Missing values")

print((df["Age"] == "?").sum())

#Preparing data for Training

data = df.iloc[:, 0:-1].values

target = df.iloc[:, -1].values

#Spliting the data in test and train data

data\_train,data\_test,target\_train,target\_test = train\_test\_split(data,target,test\_size = 0.3, random\_state = 0)

#feature Scaling

sc = StandardScaler()

data\_train = sc.fit\_transform(data\_train)

data\_test = sc.fit\_transform(data\_test)

#Training the model

random\_forest = RandomForestClassifier(n\_estimators = 3)

random\_forest.fit(data\_train, target\_train)

prediction\_forest = random\_forest.predict(data\_test)

accuracy = round( random\_forest.score(data\_train, target\_train)\*100 , 2)

print("Accuracy of Random forest with 3 estimators")

print(round(accuracy, 2), "%")

#

max=0

for i in range (1,101):

random\_forest = RandomForestClassifier(n\_estimators = i)

random\_forest.fit(data\_train, target\_train)

prediction\_forest = random\_forest.predict(data\_test)

accuracy = round( random\_forest.score(data\_train, target\_train)\*100 , 2)

print("Accuracy of Random forest with",i,"estimators")

print(round(accuracy, 2), "%")

if max < round(accuracy,2):

max = round(accuracy,2)

no = i

print("Best no. of Estimators are",no)

conclusion

Random forest is a great learning model. It is one of the best algorithms as it gives the best result most of the time.

In this Project I managed to get accuracy from 95% to 98% and which is also a great prediction which can really help in the medical world today considering the type of threat disease possess today its important for us to use the power of machine Learning to best of its core as it can really help us to save around million patients all around the world.