Decision Trees

Practical work n° 9

toumi mustapha abderahmane
11 november 2019

keywords : machine learning, python , data sets,sklearn, pandas, decision trees,

1 Abstract

A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter

2 Methods and Materials

1. **Materials:**

1. **software :**

- MySQL dbms
- Navicat for MySql

2.hardware :

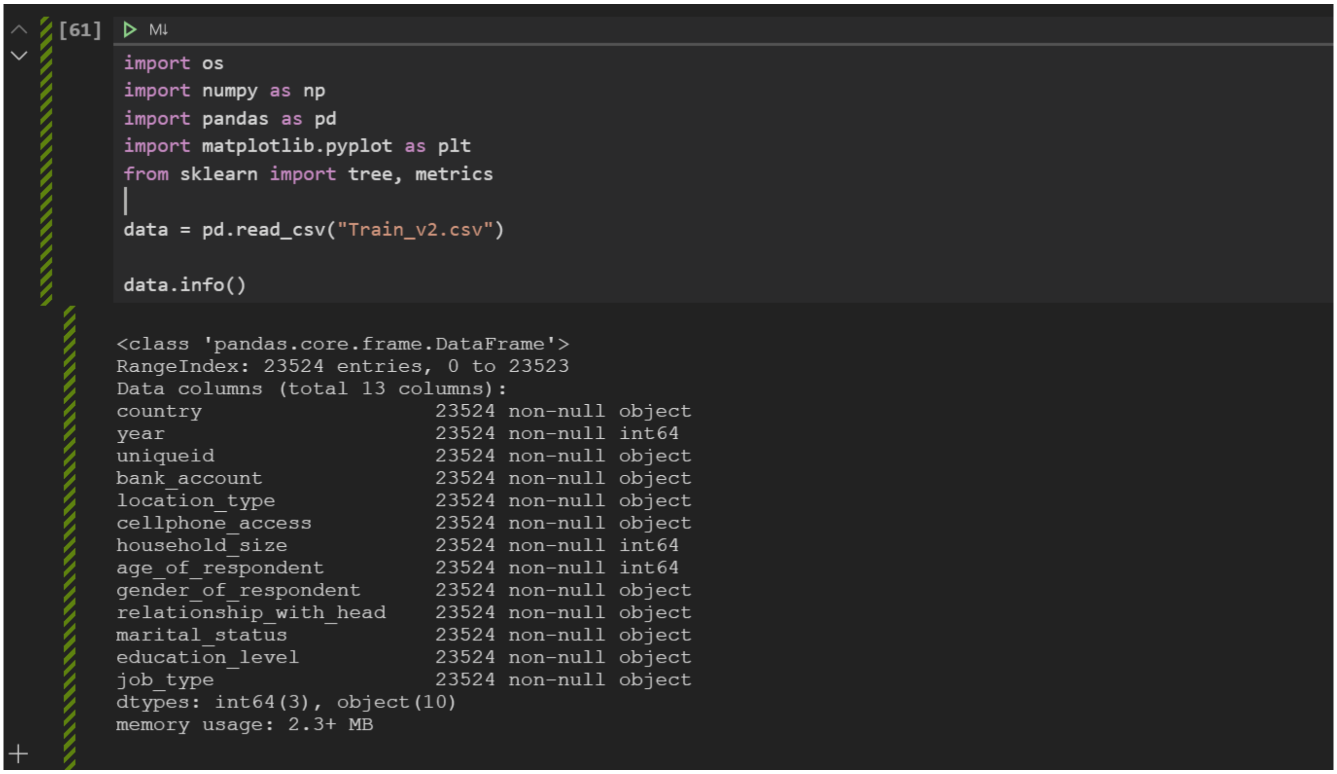
- processeur : i3 VI
- ram : 8mb
- hard disk : 125 gb

3 Results

In this work we have the data set “Train_v2” which represent a financial info about each interviewee

So we are trying to classify this interviewee our output is “bank_account “ define whether this interviewee have or doesn’t have a bank account , the input is the rest of columns except “unique_id”

First lets use pandas to read our Train_v2 .csv and see what we have , we can get info and shape too

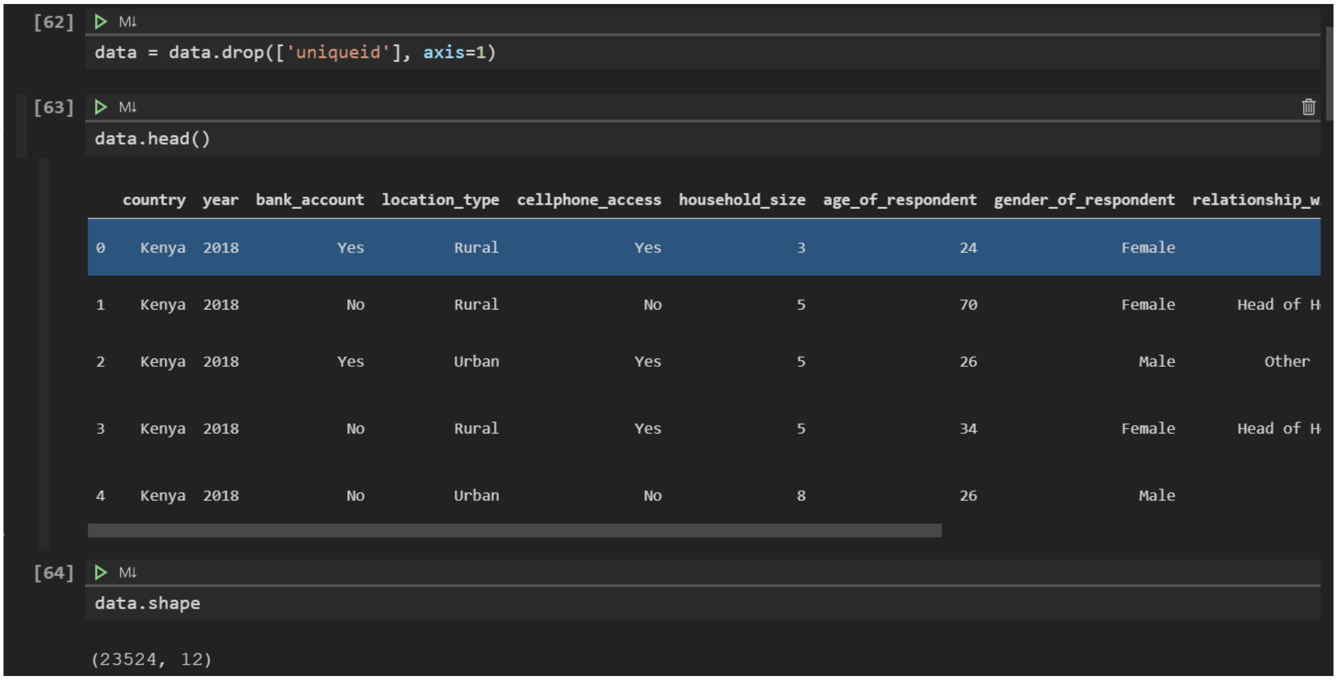


```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree, metrics

data = pd.read_csv("Train_v2.csv")

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23524 entries, 0 to 23523
Data columns (total 13 columns):
country                   23524 non-null object
year                      23524 non-null int64
uniqueid                  23524 non-null object
bank_account              23524 non-null object
location_type             23524 non-null object
cellphone_access          23524 non-null object
household_size            23524 non-null int64
age_of_respondent         23524 non-null int64
gender_of_respondent      23524 non-null object
relationship_with_head    23524 non-null object
marital_status            23524 non-null object
education_level           23524 non-null object
job_type                  23524 non-null object
dtypes: int64(3), object(10)
memory usage: 2.3+ MB
```

Figure 1: read csv and get some info

Now lets drop the unique id and see how the data look like again



```
data = data.drop(['uniqueid'], axis=1)
```

```
data.head()
```

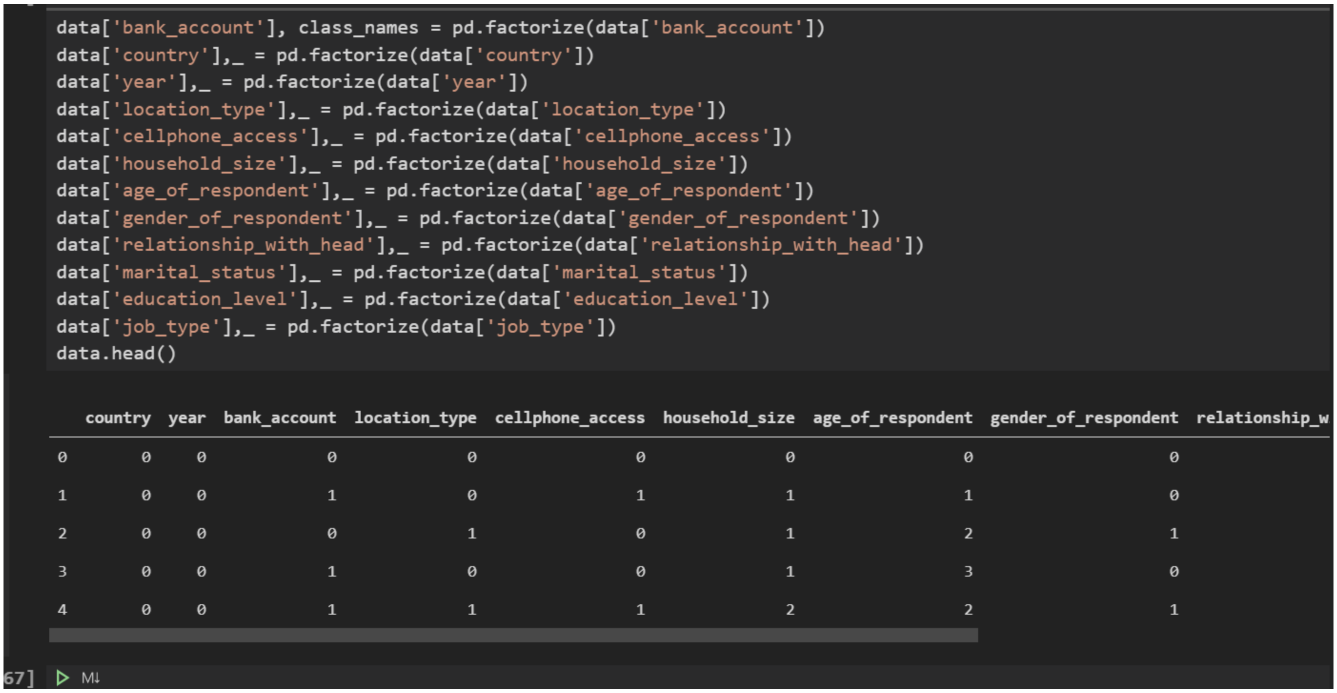
	country	year	bank_account	location_type	cellphone_access	household_size	age_of_respondent	gender_of_respondent	relationship_w
0	Kenya	2018	Yes	Rural	Yes	3	24	Female	
1	Kenya	2018	No	Rural	No	5	70	Female	Head of H
2	Kenya	2018	Yes	Urban	Yes	5	26	Male	Other
3	Kenya	2018	No	Rural	Yes	5	34	Female	Head of H
4	Kenya	2018	No	Urban	No	8	26	Male	

```
data.shape
```

```
(23524, 12)
```

Figure 2: drop unique_id and display the head and shape of the data

Now we are ready to convert our columns to a unique numeric types using factorize

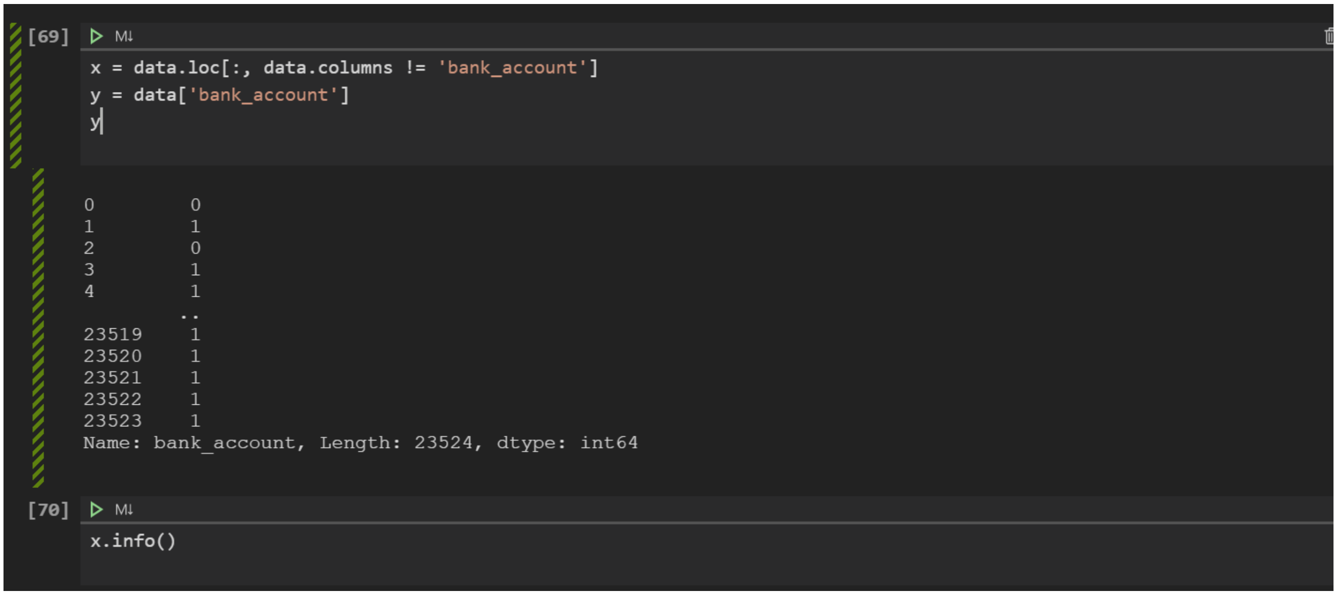


```
data['bank_account'], class_names = pd.factorize(data['bank_account'])
data['country'],_ = pd.factorize(data['country'])
data['year'],_ = pd.factorize(data['year'])
data['location_type'],_ = pd.factorize(data['location_type'])
data['cellphone_access'],_ = pd.factorize(data['cellphone_access'])
data['household_size'],_ = pd.factorize(data['household_size'])
data['age_of_respondent'],_ = pd.factorize(data['age_of_respondent'])
data['gender_of_respondent'],_ = pd.factorize(data['gender_of_respondent'])
data['relationship_with_head'],_ = pd.factorize(data['relationship_with_head'])
data['marital_status'],_ = pd.factorize(data['marital_status'])
data['education_level'],_ = pd.factorize(data['education_level'])
data['job_type'],_ = pd.factorize(data['job_type'])
data.head()
```

	country	year	bank_account	location_type	cellphone_access	household_size	age_of_respondent	gender_of_respondent	relationship_w
0	0	0	0	0	0	0	0	0	
1	0	0	1	0	1	1	1	0	
2	0	0	0	1	0	1	2	1	
3	0	0	1	0	0	1	3	0	
4	0	0	1	1	1	2	2	1	

Figure 3: Convert columns to numeric types

Separate data into an input ‘x’ and output ‘y’



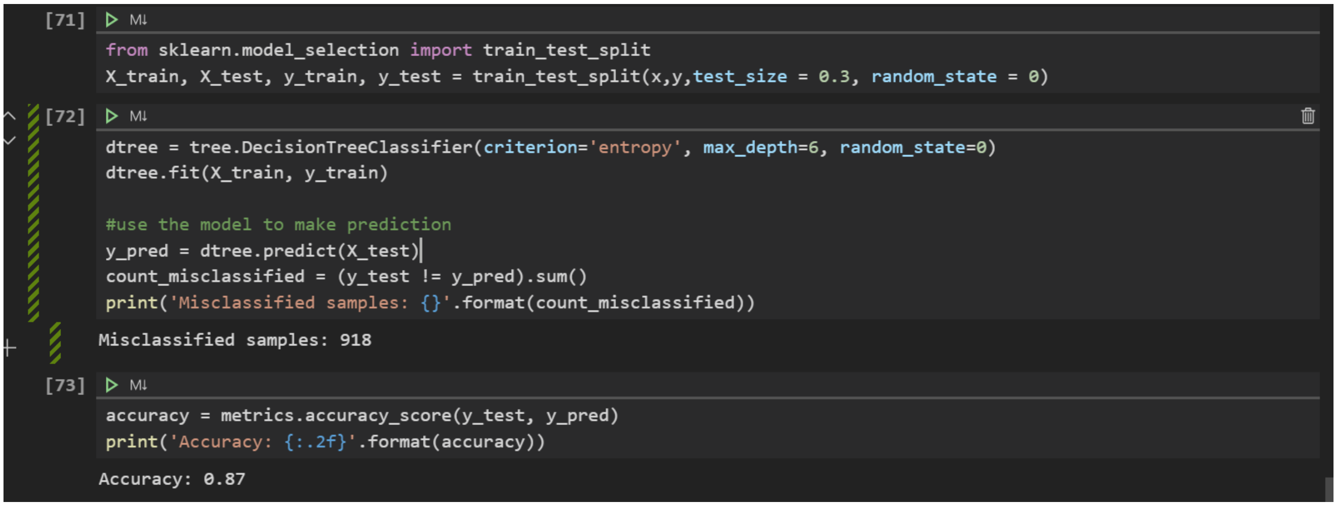
```
x = data.loc[:, data.columns != 'bank_account']
y = data['bank_account']
y
```

```
0        0
1        1
2        0
3        1
4        1
        ..
23519    1
23520    1
23521    1
23522    1
23523    1
Name: bank_account, Length: 23524, dtype: int64
```

```
x.info()
```

Figure 4: input is x and the output is y

Now we are ready to split the data into training and testing , and start the training and the prediction phase after that we will calculate the miss classified samples and the accuracy



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size = 0.3, random_state = 0)
```

```
dtree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=6, random_state=0)
dtree.fit(X_train, y_train)

#use the model to make prediction
y_pred = dtree.predict(X_test)
count_misclassified = (y_test != y_pred).sum()
print('Misclassified samples: {}'.format(count_misclassified))
```

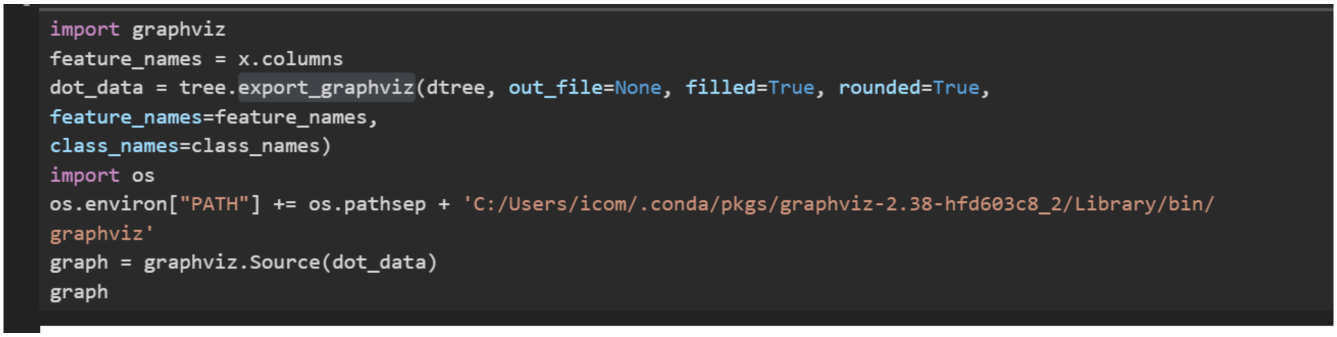
```
Misclassified samples: 918
```

```
accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: {:.2f}'.format(accuracy))
```

```
Accuracy: 0.87
```

Figure 5: Train, Split, Predict

Display the tree using “graphviz”



```
import graphviz
feature_names = x.columns
dot_data = tree.export_graphviz(dtree, out_file=None, filled=True, rounded=True,
feature_names=feature_names,
class_names=class_names)
import os
os.environ["PATH"] += os.pathsep + 'C:/Users/icom/.conda/pkgs/graphviz-2.38-hfd603c8_2/Library/bin/
graphviz'
graph = graphviz.Source(dot_data)
graph
```

Figure 6: code to visualize decision tree using graphviz

i Visualissation of the tree is in this file below “Source.gv”

4 Conclusion

The accuracy was 87% with 918 miss classified samples this was a good result