Naive Bayes (Iris)

October 30, 2023

0.0.1 MST IASD 2023-2024 (Département Génie Informatique)

Module "Apprentissage automatique" (M. AIT KBIR)

Bayes Naif appliqué la classification des Iris de Fisher

Dataset présenté en 1936 par Ronald Fisher

Fischer propose une méthode multiparamétrique pour distinguer trois classes de fleurs Iris (setosa, versicolor et virginica) à l'aide de quatre paramètres biométriques de détermination aisée (la longueur et la largeur des sépales ainsi que la longueur et la largeur des pétales, en cm), la base de données concerne 50 individus de chaque espèce.

Désactiver les commentaires pour voir les résultats intermédaires

[2]:
```
import pandas as pd
import numpy as np
from scipy.stats import norm
from sklearn import metrics
import seaborn as sns
```

[3]:
```
dataSet = sns.load_dataset("iris")
dataSet
```

[3]:

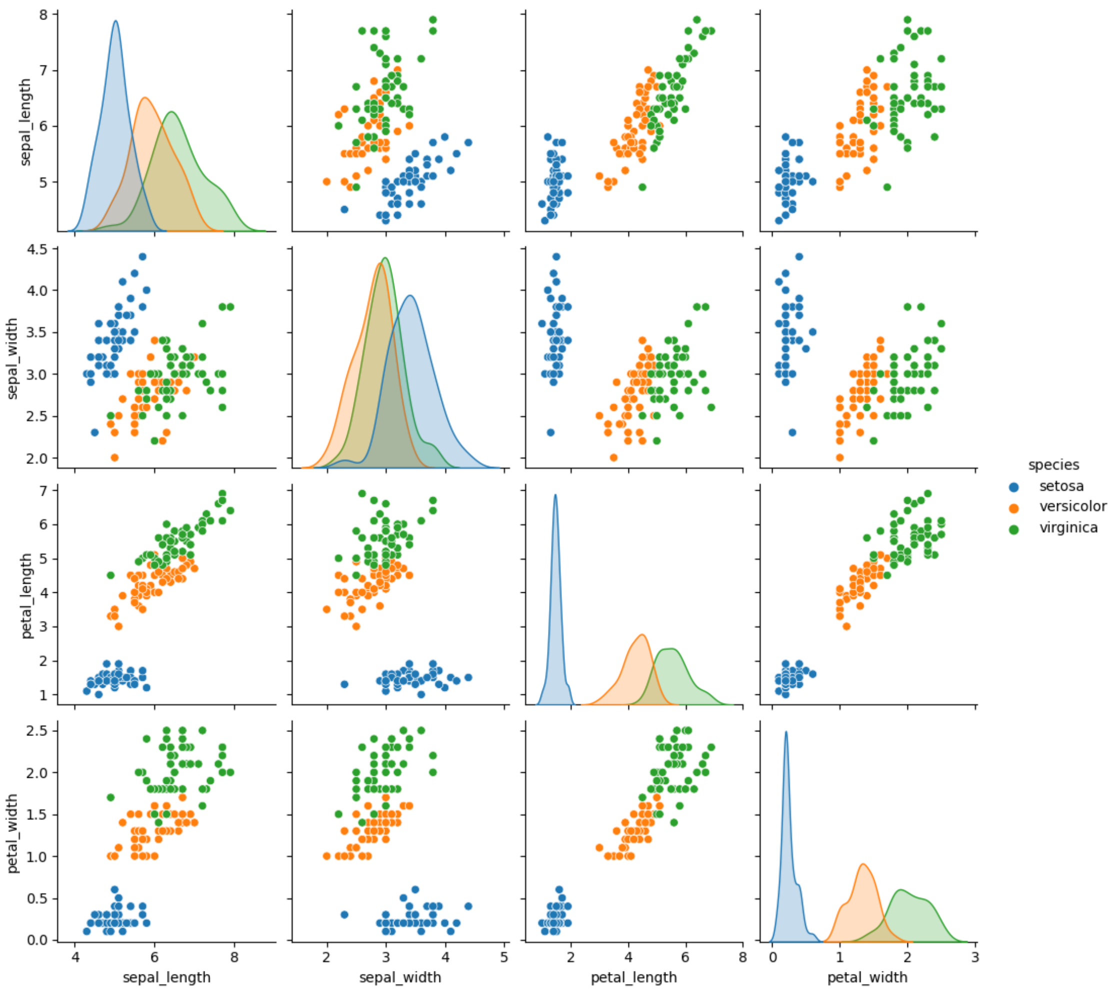
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
..	…	…	…	…	…
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

[150 rows x 5 columns]

0.0.2 Charger le fichier csv dans un objet du module pandas : DataFrame

[4]:
```python
sns.pairplot(dataSet, hue="species")
```

[4]:
```
<seaborn.axisgrid.PairGrid at 0x2199c5e1c10>
```



0.0.3 Créer une Base des exemples tests

[5]:
```python
nbreTest = 30; # 10 exemples par classe
# Elaborer les listes des indices des exemples utilisés dans l'apprentissage(trainingInd) et dans le test(testInd)
trainingInd = list(range(len(dataSet))); testInd=[]
for i in range(nbreTest):
    randIndex = int(np.random.uniform(0,len(trainingInd)))
    testInd.append(trainingInd[randIndex])
    del(trainingInd[randIndex])
```

```
del trainingInd

testSet = pd.concat([dataSet.loc[testInd]], axis=0)
dataSet.drop(dataSet.index[testInd], inplace=True) # retirer de la base des␣
 ↪exemples d'apprentissage
testSet
```

[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
48	5.3	3.7	1.5	0.2	setosa
23	5.1	3.3	1.7	0.5	setosa
104	6.5	3.0	5.8	2.2	virginica
121	5.6	2.8	4.9	2.0	virginica
42	4.4	3.2	1.3	0.2	setosa
82	5.8	2.7	3.9	1.2	versicolor
148	6.2	3.4	5.4	2.3	virginica
45	4.8	3.0	1.4	0.3	setosa
61	5.9	3.0	4.2	1.5	versicolor
53	5.5	2.3	4.0	1.3	versicolor
108	6.7	2.5	5.8	1.8	virginica
72	6.3	2.5	4.9	1.5	versicolor
128	6.4	2.8	5.6	2.1	virginica
116	6.5	3.0	5.5	1.8	virginica
24	4.8	3.4	1.9	0.2	setosa
79	5.7	2.6	3.5	1.0	versicolor
49	5.0	3.3	1.4	0.2	setosa
91	6.1	3.0	4.6	1.4	versicolor
110	6.5	3.2	5.1	2.0	virginica
129	7.2	3.0	5.8	1.6	virginica
60	5.0	2.0	3.5	1.0	versicolor
139	6.9	3.1	5.4	2.1	virginica
102	7.1	3.0	5.9	2.1	virginica
27	5.2	3.5	1.5	0.2	setosa
10	5.4	3.7	1.5	0.2	setosa
133	6.3	2.8	5.1	1.5	virginica
68	6.2	2.2	4.5	1.5	versicolor
31	5.4	3.4	1.5	0.4	setosa
119	6.0	2.2	5.0	1.5	virginica
111	6.4	2.7	5.3	1.9	virginica

0.0.4 Calcul des moyennes et des variances

[6]:
```
mean_dataSet = dataSet.groupby('species').mean()
std_dataSet = dataSet.groupby('species').std()
print(mean_dataSet)
print(std_dataSet)
```

```
            sepal_length  sepal_width  petal_length  petal_width
species
```

```
setosa          4.997561     3.436585      1.448780     0.241463
versicolor      5.959524     2.814286      4.283333     1.330952
virginica       6.624324     3.005405      5.594595     2.064865
            sepal_length  sepal_width  petal_length  petal_width
species
setosa          0.360200     0.406667      0.171933     0.104823
versicolor      0.533310     0.287607      0.464819     0.196913
virginica       0.694584     0.325701      0.607795     0.272046
```

0.0.5 Calcul des probabilités

[7]:
```
dataSet['species'].value_counts()
```

[7]:
```
versicolor    42
setosa        41
virginica     37
Name: species, dtype: int64
```

[8]:
```
# Probabilités à priori
nbresParClass = dataSet['species'].value_counts()
probApriori = {label: float(nbresParClass[label])/dataSet.shape[0] for label in␣
 ↪dataSet['species'].unique()}
probApriori
```

[8]:
```
{'setosa': 0.3416666666666667,
 'versicolor': 0.35,
 'virginica': 0.30833333333333335}
```

[9]:
```
# Des fonctions
# P(xi/wj)
def p_xi_wj(xi, wj, attrib):
    return norm.pdf(xi, loc = mean_dataSet.loc[wj, attrib], scale = std_dataSet.
 ↪loc[wj, attrib])

# P(X/wj) * P(wj)
def p_X_wj(exemple, wj):
    prob = probApriori[wj]
    for attrib, valeur in exemple.items():
        prob*=p_xi_wj(valeur, wj, attrib)
    return prob
```

0.0.6 Test

[10]:
```
labels   = dataSet['species'].unique()

# Extraire les colonnes sans classes
data = testSet.iloc[:,:-1]
```

```
# Ajouter des colonnes
testSet['classe calculée'] = np.nan
for i in range(len(labels)):
    testSet[labels[i]]     = np.nan

# Faire pour chaque exemple test
for index, row in data.iterrows():
    probCond = [p_X_wj(row, label) for label in labels]
    testSet.loc[index, 'classe calculée'] = labels[probCond.
 ↪index(max(probCond))]
    for i in range(len(labels)):
        testSet.loc[index, labels[i]] = probCond[i]/sum(probCond)
testSet
```

[10]:

```
     sepal_length  sepal_width  petal_length  petal_width     species  \
48            5.3          3.7           1.5          0.2      setosa
23            5.1          3.3           1.7          0.5      setosa
104           6.5          3.0           5.8          2.2   virginica
121           5.6          2.8           4.9          2.0   virginica
42            4.4          3.2           1.3          0.2      setosa
82            5.8          2.7           3.9          1.2  versicolor
148           6.2          3.4           5.4          2.3   virginica
45            4.8          3.0           1.4          0.3      setosa
61            5.9          3.0           4.2          1.5  versicolor
53            5.5          2.3           4.0          1.3  versicolor
108           6.7          2.5           5.8          1.8   virginica
72            6.3          2.5           4.9          1.5  versicolor
128           6.4          2.8           5.6          2.1   virginica
116           6.5          3.0           5.5          1.8   virginica
24            4.8          3.4           1.9          0.2      setosa
79            5.7          2.6           3.5          1.0  versicolor
49            5.0          3.3           1.4          0.2      setosa
91            6.1          3.0           4.6          1.4  versicolor
110           6.5          3.2           5.1          2.0   virginica
129           7.2          3.0           5.8          1.6   virginica
60            5.0          2.0           3.5          1.0  versicolor
139           6.9          3.1           5.4          2.1   virginica
102           7.1          3.0           5.9          2.1   virginica
27            5.2          3.5           1.5          0.2      setosa
10            5.4          3.7           1.5          0.2      setosa
133           6.3          2.8           5.1          1.5   virginica
68            6.2          2.2           4.5          1.5  versicolor
31            5.4          3.4           1.5          0.4      setosa
119           6.0          2.2           5.0          1.5   virginica
111           6.4          2.7           5.3          1.9   virginica

    classe calculée        setosa    versicolor     virginica
```

```
48       setosa   1.000000e+00  1.744349e-18  1.843061e-23
23       setosa   1.000000e+00  2.258513e-11  2.024727e-17
104   virginica  2.337726e-218  5.141155e-07  9.999995e-01
121   virginica  1.959558e-148  2.174954e-02  9.782505e-01
42       setosa   1.000000e+00  6.263519e-19  2.099499e-24
82   versicolor   8.476400e-64  9.999726e-01  2.737453e-05
148   virginica  8.521716e-200  4.055125e-07  9.999996e-01
45       setosa   1.000000e+00  1.822398e-16  1.680990e-22
61   versicolor   2.866325e-88  9.970978e-01  2.902229e-03
53   versicolor   3.826510e-71  9.999541e-01  4.594494e-05
108   virginica  7.367246e-192  1.033718e-03  9.989663e-01
72   versicolor  6.309595e-122  9.600349e-01  3.996515e-02
128   virginica  4.317706e-198  2.455291e-05  9.999754e-01
116   virginica  6.587197e-172  4.606437e-03  9.953936e-01
24       setosa   1.000000e+00  1.208785e-14  2.272228e-20
79   versicolor   1.242807e-42  9.999980e-01  1.969915e-06
49       setosa   1.000000e+00  3.310396e-18  9.302500e-24
91   versicolor  1.554265e-101  9.944058e-01  5.594171e-03
110   virginica  3.474417e-162  8.515264e-04  9.991485e-01
129   virginica  1.208998e-182  2.019375e-03  9.979806e-01
60   versicolor   2.517430e-41  9.999989e-01  1.072257e-06
139   virginica  1.325593e-188  1.270737e-05  9.999873e-01
102   virginica  7.965538e-221  4.153847e-07  9.999996e-01
27       setosa   1.000000e+00  6.137836e-18  2.873093e-23
10       setosa   1.000000e+00  2.835812e-18  3.149385e-23
133  versicolor  9.221692e-132  8.562367e-01  1.437633e-01
68   versicolor  6.865847e-103  9.948745e-01  5.125542e-03
31       setosa   1.000000e+00  1.969289e-14  4.090427e-20
119  versicolor  5.474972e-126  9.664764e-01  3.352364e-02
111   virginica  1.661143e-166  6.132511e-03  9.938675e-01
```

[11]:
```python
# Taux de la classification correcte
metrics.accuracy_score(testSet['species'], testSet['classe calculée'])
```

[11]: 0.9333333333333333

[12]:
```python
metrics.confusion_matrix(testSet['species'], testSet['classe calculée'])
```

[12]:
```
array([[ 9,  0,  0],
       [ 0,  8,  0],
       [ 0,  2, 11]], dtype=int64)
```