Laporan Jobsheet 13

Teknologi Data (NAÏVE BAYES CLASSIFIER PYTHON)

Disusun oleh :

Nama :Dhuta Pamungkas Ibnusiqin

Nim : 1941723014

Class:TI-3D

Jurusan Teknologi Informasi

Politeknik Negeri Malang
2021

Praktikum 1 Berdasarkan materi
https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

Step 1: Separate By Class

```
# Split the dataset by class values, returns a dictionary
def separate_by_class(dataset):
  separated = dict()
  for i in range(len(dataset)):
    vector = dataset[i]
    class_value = vector[-1]
    if (class_value not in separated):
      separated[class_value] = list()
    separated[class_value].append(vector)
  return separated

# Test separating data by class
dataset = [[3.393533211,2.331273381,0],
  [3.110073483,1.781539638,0],
  [1.343808831,3.368360954,0],
  [3.582294042,4.67917911,0],
  [2.280362439,2.866990263,0],
  [7.423436942,4.696522875,1],
  [5.745051997,3.533989803,1],
  [9.172168622,2.511101045,1],
  [7.792783481,3.424088941,1],
  [7.939820817,0.791637231,1]]
separated = separate_by_class(dataset)
for label in separated:
  print(label)
  for row in separated[label]:
    print(row)
```

```
0
[3.393533211, 2.331273381, 0]
[3.110073483, 1.781539638, 0]
[1.343808831, 3.368360954, 0]
[3.582294042, 4.67917911, 0]
[2.280362439, 2.866990263, 0]
1
[7.423436942, 4.696522875, 1]
[5.745051997, 3.533989803, 1]
[9.172168622, 2.511101045, 1]
[7.792783481, 3.424088941, 1]
[7.939820817, 0.791637231, 1]
```

Step 2: Summarize Dataset

```
# Example of summarizing a dataset
```

```
from math import sqrt

# Calculate the mean of a list of numbers
def mean(numbers):
  return sum(numbers)/float(len(numbers))

# Calculate the standard deviation of a list of numbers
def stdev(numbers):
  avg = mean(numbers)
  variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
  return sqrt(variance)

# Calculate the mean, stdev and count for each column in a dataset
def summarize_dataset(dataset):
  summaries = [(mean(column), stdev(column), len(column)) for column in
zip(*dataset)]
  del(summaries[-1])
  return summaries

# Test summarizing a dataset
dataset = [[3.393533211,2.331273381,0],
  [3.110073483,1.781539638,0],
  [1.343808831,3.368360954,0],
  [3.582294042,4.67917911,0],
  [2.280362439,2.866990263,0],
  [7.423436942,4.696522875,1],
  [5.745051997,3.533989803,1],
  [9.172168622,2.511101045,1],
  [7.792783481,3.424088941,1],
  [7.939820817,0.791637231,1]]
summary = summarize_dataset(dataset)
print(summary)
```

```
[(5.178333386499999, 2.7665845055177263, 10), (2.9984683241, 1.218556343617447, 10)]
```

Step 3: Summarize Data By Class

```
# Example of summarizing data by class value
from math import sqrt

# Split the dataset by class values, returns a dictionary
def separate_by_class(dataset):
  separated = dict()
  for i in range(len(dataset)):
    vector = dataset[i]
```

```
    class_value = vector[-1]
    if (class_value not in separated):
      separated[class_value] = list()
    separated[class_value].append(vector)
  return separated

# Calculate the mean of a list of numbers
def mean(numbers):
  return sum(numbers)/float(len(numbers))

# Calculate the standard deviation of a list of numbers
def stdev(numbers):
  avg = mean(numbers)
  variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
  return sqrt(variance)

# Calculate the mean, stdev and count for each column in a dataset
def summarize_dataset(dataset):
  summaries = [(mean(column), stdev(column), len(column)) for column in
zip(*dataset)]
  del(summaries[-1])
  return summaries

# Split dataset by class then calculate statistics for each row
def summarize_by_class(dataset):
  separated = separate_by_class(dataset)
  summaries = dict()
  for class_value, rows in separated.items():
    summaries[class_value] = summarize_dataset(rows)
  return summaries

# Test summarizing by class
dataset = [[3.393533211,2.331273381,0],
  [3.110073483,1.781539638,0],
  [1.343808831,3.368360954,0],
  [3.582294042,4.67917911,0],
  [2.280362439,2.866990263,0],
  [7.423436942,4.696522875,1],
  [5.745051997,3.533989803,1],
  [9.172168622,2.511101045,1],
  [7.792783481,3.424088941,1],
  [7.939820817,0.791637231,1]]
summary = summarize_by_class(dataset)
for label in summary:
  print(label)
```

```
  for row in summary[label]:
    print(row)
```

```
0
(2.7420144012, 0.9265683289298018, 5)
(3.0054686692, 1.1073295894898725, 5)
1
(7.6146523718, 1.2344321550313704, 5)
(2.9914679790000003, 1.4541931384601618, 5)
```

Step 4: Gaussian Probability Density Function

```
# Example of Gaussian PDF
from math import sqrt
from math import pi
from math import exp

# Calculate the Gaussian probability distribution function for x
def calculate_probability(x, mean, stdev):
  exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
  return (1 / (sqrt(2 * pi) * stdev)) * exponent

# Test Gaussian PDF
print(calculate_probability(1.0, 1.0, 1.0))
print(calculate_probability(2.0, 1.0, 1.0))
print(calculate_probability(0.0, 1.0, 1.0))
```

```
0.3989422804014327
0.24197072451914337
0.24197072451914337
```

Step 5: Class Probabilities

```
# Example of calculating class probabilities
from math import sqrt
from math import pi
from math import exp

# Split the dataset by class values, returns a dictionary
def separate_by_class(dataset):
  separated = dict()
  for i in range(len(dataset)):
    vector = dataset[i]
    class_value = vector[-1]
    if (class_value not in separated):
      separated[class_value] = list()
    separated[class_value].append(vector)
```

```
  return separated

# Calculate the mean of a list of numbers
def mean(numbers):
  return sum(numbers)/float(len(numbers))

# Calculate the standard deviation of a list of numbers
def stdev(numbers):
  avg = mean(numbers)
  variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
  return sqrt(variance)

# Calculate the mean, stdev and count for each column in a dataset
def summarize_dataset(dataset):
  summaries = [(mean(column), stdev(column), len(column)) for column in
zip(*dataset)]
  del(summaries[-1])
  return summaries

# Split dataset by class then calculate statistics for each row
def summarize_by_class(dataset):
  separated = separate_by_class(dataset)
  summaries = dict()
  for class_value, rows in separated.items():
    summaries[class_value] = summarize_dataset(rows)
  return summaries

# Calculate the Gaussian probability distribution function for x
def calculate_probability(x, mean, stdev):
  exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
  return (1 / (sqrt(2 * pi) * stdev)) * exponent

# Calculate the probabilities of predicting each class for a given row
def calculate_class_probabilities(summaries, row):
  total_rows = sum([summaries[label][0][2] for label in summaries])
  probabilities = dict()
  for class_value, class_summaries in summaries.items():
    probabilities[class_value] = summaries[class_value][0][2]/float(tota
l_rows)
    for i in range(len(class_summaries)):
      mean, stdev, _ = class_summaries[i]
      probabilities[class_value] *= calculate_probability(row[i], mean,
stdev)
  return probabilities
```

```
# Test calculating class probabilities
dataset = [[3.393533211,2.331273381,0],
  [3.110073483,1.781539638,0],
  [1.343808831,3.368360954,0],
  [3.582294042,4.67917911,0],
  [2.280362439,2.866990263,0],
  [7.423436942,4.696522875,1],
  [5.745051997,3.533989803,1],
  [9.172168622,2.511101045,1],
  [7.792783481,3.424088941,1],
  [7.939820817,0.791637231,1]]
summaries = summarize_by_class(dataset)
probabilities = calculate_class_probabilities(summaries, dataset[0])
print(probabilities)
```

```
{0: 0.05032427673372076, 1: 0.0001155771837994576 5}
```

Praktikum 2 Berdasarkan Materi
https://towardsdatascience.com/implementing-naive-bayes-in-2-minutes-with-python-3ecd788803fe

1. Import Library Python dan Data Excel

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#memasukan data latih
datalatih = pd.read_excel("data testing.xlsx")
datalatih.head(11)
#Prestasi Siswa = 1 == Ya
#Prestasi Siswa = 2 == Tidak
#Prestasi Sekolah = 1 == Cukup
#Prestasi Sekolah = 2 == Baik
```

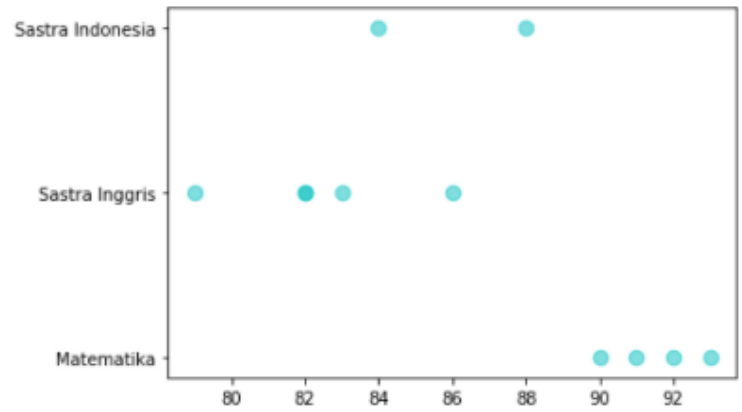
Hasil:

	Jurusan	Matematika	Bahasa Inggris	Bahasa Indonesia	Prestasi Siswa	Prestasi Sekolah
0	Matematika	92	86	88	1	1
1	Matematika	90	86	85	1	2
2	Sastra Inggris	82	87	92	2	1
3	Matematika	93	82	85	1	1
4	Sastra Inggris	86	85	88	1	1
5	Matematika	91	83	84	2	2
6	Sastra Indonesia	88	80	90	2	1
7	Sastra Indonesia	84	82	88	1	1
8	Sastra Inggris	79	89	85	1	1
9	Sastra Inggris	83	88	81	2	2
10	Sastra Inggris	82	85	84	2	1

2. Visualization

```
from sklearn.cluster import KMeans
#Memvisualkan persebaran data
plt.scatter(datalatih.Matematika, datalatih.Jurusan, s = 75, c = "c",
marker = "o", alpha = 0.5)
plt.show()
```

Hasil:



3.

```
x = datalatih.drop(["Jurusan"], axis=1)
x.head(11)
```

	Matematika	Bahasa Inggris	Bahasa Indonesia	Prestasi Siswa	Prestasi Sekolah
0	92	86	88	1	1
1	90	86	85	1	2
2	82	87	92	2	1
3	93	82	85	1	1
4	86	85	88	1	1
5	91	83	84	2	2
6	88	80	90	2	1
7	84	82	88	1	1
8	79	89	85	1	1
9	83	88	81	2	2
10	82	85	84	2	1

4.

```
y = datalatih["Jurusan"]
y.head(11)
```

```
0           Matematika
1           Matematika
2       Sastra Inggris
3           Matematika
4       Sastra Inggris
5           Matematika
6     Sastra Indonesia
7     Sastra Indonesia
8       Sastra Inggris
9       Sastra Inggris
10      Sastra Inggris
Name: Jurusan, dtype: object
```

5.

```
from sklearn.model_selection import train_test_split
# Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
# Mengaktifkan/memanggil/membuat fungsi klasifikasi Naive bayes
modelnb = GaussianNB()
# Memasukkan data training pada fungsi klasifikasi naive bayes
nbtrain = modelnb.fit(x, y)
datauji = pd.read_excel("data uji akurasi.xlsx")
datauji.head(11)
```

	Jurusan	Matematika	Bahasa Inggris	Bahasa Indonesia	Prestasi Siswa	Prestasi Sekolah
0	Matematika	92	82	85	1	1
1	Matematika	91	83	85	2	2
2	Sastra Inggris	82	80	88	1	1
3	Sastra Indonesia	88	80	90	2	1
4	Sastra Inggris	83	85	84	2	1
5	Sastra Indonesia	88	82	88	1	1

6.

```
x_test = datauji.drop(["Jurusan"], axis=1)
x_test.head(11)
```

	Matematika	Bahasa Inggris	Bahasa Indonesia	Prestasi Siswa	Prestasi Sekolah
0	92	82	85	1	1
1	91	83	85	2	2
2	82	80	88	1	1
3	88	80	90	2	1
4	83	85	84	2	1
5	88	82	88	1	1

7.

```
y_uji = datauji["Jurusan"]
y_uji.head(11)
```

```
0          Matematika
1          Matematika
2      Sastra Inggris
3    Sastra Indonesia
4      Sastra Inggris
5    Sastra Indonesia
Name: Jurusan, dtype: object
```

8.

```
Y_predict = nbtrain.predict(x_test)
print("Prediksi Naive Bayes : ",Y_predict)
```

```
Prediksi Naive Bayes :  ['Matematika' 'Matematika' 'Sastra Indonesia' 'Sastra Indonesia'
 'Sastra Inggris' 'Sastra Indonesia']
```

9.

```
from sklearn.model_selection import train_test_split
# Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
# Mengaktifkan/memanggil/membuat fungsi klasifikasi Naive bayes
modelnb = GaussianNB()
# Memasukkan data training pada fungsi klasifikasi naive bayes
nbtrain = modelnb.fit(x, y)
datauji = pd.read_excel("data uji akurasi.xlsx")
datauji.head(11)
```

	Jurusan	Matematika	Bahasa Inggris	Bahasa Indonesia	Prestasi Siswa	Prestasi Sekolah
0	Matematika	92	82	85	1	1
1	Matematika	91	83	85	2	2
2	Sastra Inggris	82	80	88	1	1
3	Sastra Indonesia	88	80	90	2	1
4	Sastra Inggris	83	85	84	2	1
5	Sastra Indonesia	88	82	88	1	1

10.

```
x_test = datauji.drop(["Jurusan"], axis=1)
x_test.head(11)
```

	Matematika	Bahasa Inggris	Bahasa Indonesia	Prestasi Siswa	Prestasi Sekolah
0	92	82	85	1	1
1	91	83	85	2	2
2	82	80	88	1	1
3	88	80	90	2	1
4	83	85	84	2	1
5	88	82	88	1	1

Naïve Bayes Prediction :

```
Y_predict = nbtrain.predict(x_test)
print("Prediksi Naive Bayes : ",Y_predict)
```

```
Prediksi Naive Bayes :  ['Matematika' 'Matematika' 'Sastra Indonesia' 'Sastra Indonesia'
 'Sastra Inggris' 'Sastra Indonesia']
```

```
from sklearn.metrics import accuracy_score
accuracy= accuracy_score(y_uji, Y_predict)
print("Akurasi Naive Bayes : ",accuracy)
```

	Akurasi Naive Bayes : 0.8333333333333334
	# Menghitung nilai akurasi dari klasifikasi naive bayes from sklearn.metrics import classification_report print(classification_report(y_uji, Y_predict))
	precision recall f1-score support Matematika 1.00 1.00 1.00 2 Sastra Indonesia 0.67 1.00 0.80 2 Sastra Inggris 1.00 0.50 0.67 2 accuracy 0.83 6 macro avg 0.89 0.83 0.82 6 weighted avg 0.89 0.83 0.82 6