|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Seat Number:10097 Sign: Date: / /** |
| **Student Name: Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q1. Find the correlation matrix.** |

# Program:

import numpy as np

ModuleNotFoundError Traceback (most recent call last)Input In [2], in <cell line: 1>()

----> 1 import numpy as np

ModuleNotFoundError: No module named 'numpy' x = np.array([3,5,11,21,28,35,56,61,72,88]

)

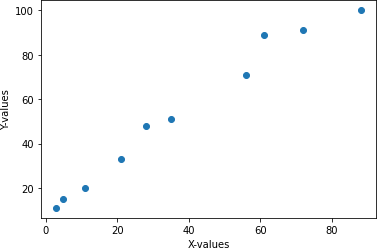
y = np.array([11,15,20,33,48,51,71,89,91,100])

z = np.array([104,100,89,81,76,66,69,43,17,11])

type(x) numpy.ndarr ay import matplotlib.pyplot as plt

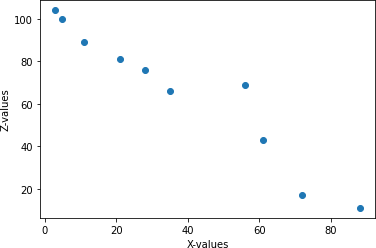
plt.xlabel('X-values') plt.ylabel('Y-values') plt.scatter(x, y)

<matplotlib.collections.PathCollection at 0x7f153834ffa0>



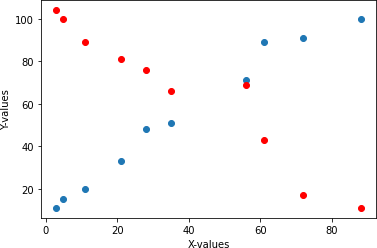
plt.xlabel('X-values') plt.ylabel('Z-values') plt.scatter(x, z)

<matplotlib.collections.PathCollection at 0x7f15362423d0>



plt.xlabel('X-values') plt.ylabel('Y-values') plt.scatter(x, y) plt.scatter(x, z, color = 'r')

<matplotlib.collections.PathCollection at 0x7f153

61aa130

np.corrcoef(x, y)

array([[1. , 0.98757408],

[0.98757408, 1. ]])

np.corrcoef(x, z)

array([[ 1. , -0.96149075],

[-0.96149075, 1. ]])

np.corrcoef(z, y)

array([[ 1. , -0.95002927],

[-0.95002927, 1. ]])

import scipy.stats as st st.pearsonr(x, y)[0] 0.9875740814243562

st.pearsonr(x, z)[0]

-0.9614907503686154

st.pearsonr(z, y)[0]

-0.9500292724815624

import pandas as pd

x1 = pd.Series([3,5,11,21,28,35,56,61,72,88])

y1 = pd.Series([11,15,20,33,48,51,71,89,91,100]) z1 = pd.Series([104,100,89,81,76,66,69,43,17,11])

x1.corr(y1) 0.98757408142435

62

y1.corr(z1)

-0.9500292724815624

df =

})

pd.DataFrame(

{'x': x,

'y': y,

'z': z

|  |  |  |  |
| --- | --- | --- | --- |
| df |  |  |  |
|  | x | y | z |
| 0 | 3 | 11 | 104 |
| 1 | 5 | 15 | 100 |
| 2 | 11 | 20 | 89 |
| 3 | 21 | 33 | 81 |
| 4 | 28 | 48 | 76 |
| 5 | 35 | 51 | 66 |
| 6 | 56 | 71 | 69 |
| 7 | 61 | 89 | 43 |
| 8 | 72 | 91 | 17 |
| 9 | 88 | 100 | 11 |

df.corr()

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 1.000000 | 0.987574 | -0.961491 |
| y | 0.987574 | 1.000000 | -0.950029 |
| z | -0.961491 | -0.950029 | 1.000000 |

df.corrwith(x1) x

1.00000

0

y 0.987574

z -0.961491

dtype: float64 st.spearmanr(x, y)[0] 0.9999999999999999

st.spearmanr(x, z)[0]

-0.9878787878787878

st.spearmanr(z, y)[0]

-0.9878787878787878

df.corr(method='spearman')

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 1.000000 | 1.000000 | -0.987879 |
| y | 1.000000 | 1.000000 | -0.987879 |
| z | -0.987879 | -0.987879 | 1.000000 |

st.kendalltau(x, y)[0] 0.9999999999999999

st.kendalltau(x, z)[0]

-0.9555555555555554

st.kendalltau(z, y)[0]

-0.9555555555555554

df.corr(method='kendall')

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 1.000000 | 1.000000 | -0.955556 |
| y | 1.000000 | 1.000000 | -0.955556 |
| z | -0.955556 | -0.955556 | 1.000000 |

df.corrwith(x1, method='kendall')x 1.000000

y 1.000000

z -0.955556

dtype: float64

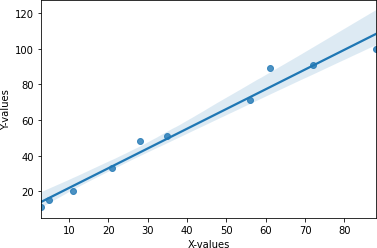
cor = df.corr (method='kendall')cor.values

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| array([[ 1. | , | 1. | , | -0.95555556], | |
| [ 1. | , | 1. | , | -0.95555556], | |
| [-0.95555556, |  | -0.95555556, |  | 1. | ]]) |

import seaborn as sns

plt.xlabel('X-values') plt.ylabel('Y-values') sns.regplot(x=x, y=y, data=df)

<AxesSubplot:xlabel='X-values', ylabel='Y

-values'>

|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number: 10097 Sign: Date: / /** |
| **Student Name:Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.2 Plot the correlation plot on dataset and visualize giving an overview ofrelationships among data on iris data.** |

# Program:

*# Data Import*

import pandas as pd

x = ['slength','swidth','plength','pwidth','species']

df = pd.read\_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data', names=x)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| df |  |  |  |  |  |
|  | slength | swidth | plength | pwidth | species |
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| .. | ... | ... | ... | ... | ... |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

[150 rows x 5 columns]

df = pd.read\_csv('iris.csv') *# Local data import*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| df |  |  |  |  |  |  |
|  | 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] | |  |  |  |

import seaborn as sns

sns.get\_dataset\_names()

['anagrams', 'anscombe', 'attention', 'brain\_networks', 'car\_crashes', 'diamonds', 'dots', 'exercise', 'flights', 'fmri', 'gammas',

'geyser',

'iris',

'mpg', 'penguins', 'planets', 'taxis',

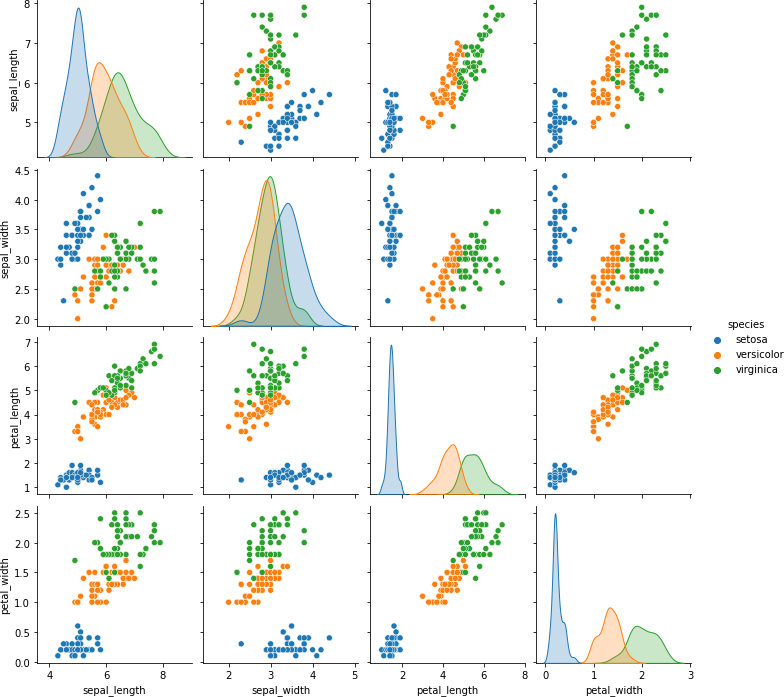
'tips', 'titanic']

iris = sns.load\_dataset('iris')

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| iris |  |  |  |  |  |  |
|  | 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] | |  |  |  |

sns.pairplot(df, hue='species')

<seaborn.axisgrid.PairGrid at 0x7fbe5dfb5df0>

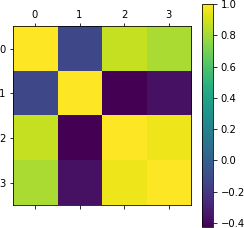


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| import matplotlib.pyplot as df.corr()  sepal\_length | | plt  sepal\_width | petal\_length | petal\_width |
| sepal\_length | 1.000000 | -0.117570 | 0.871754 | 0.817941 |
| sepal\_width | -0.117570 | 1.000000 | -0.428440 | -0.366126 |
| petal\_length | 0.871754 | -0.428440 | 1.000000 | 0.962865 |
| petal\_width | 0.817941 | -0.366126 | 0.962865 | 1.000000 |

plt.figure(figsize=(16,9)) plt.matshow(df.corr()) plt.colorbar()

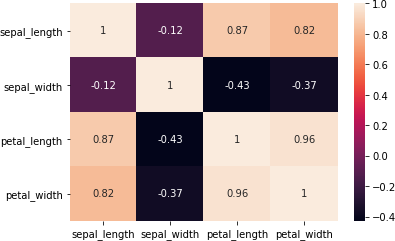
<matplotlib.colorbar.Colorbar at 0x7fbe556e8a30>

<Figure size 1152x648 with 0 Axes>



sns.heatmap(df.corr(), annot=True)

<AxesSubplot:>



|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number: 10097 Sign: Date: / /** |
| **Student Name: Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.3 Analysis of covariance: variance (ANOVA), if data have ategorical variables on iris data.** |

import pandas as pd

df = pd.read\_csv("data.txt",sep='\t') df.head()

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id gender |  | bdate educ | jobcat | salary | salbegin | jobtime prevexp | \0 | 1 |
| m | 2/3/1952 | | 15 | 3 | 57000 | 27000 | 98 | 144 |
| 1 2 | m 5/23/1958 | | 16 | 1 | 40200 | 18750 | 98 | 36 |
| 2 3 | f 7/26/1929 | | 12 | 1 | 21450 | 12000 | 98 | 381 |
| 3 4 | f 4/15/1947 | | 8 | 1 | 21900 | 13200 | 98 | 190 |
| 4 5 | m 2/9/1955 | | 15 | 1 | 45000 | 21000 | 98 | 138 |
| minority jobcat\_name | | |  |  |  |  |  |  |
| 0 | 0 | Manager |  |  |  |  |  |  |
| 1 | 0 | Clerical |  |  |  |  |  |  |
| 2 | 0 | Clerical |  |  |  |  |  |  |
| 3 | 0 | Clerical |  |  |  |  |  |  |
| 4 | 0 | Clerical |  |  |  |  |  |  |

df[['jobcat\_name','prevexp']].groupby('jobcat\_name').mean()

|  |  |
| --- | --- |
|  | prevexp |
| jobcat\_name |  |
| Clerical | 85.038567 |
| Custodial | 298.111111 |
| Manager | 77.619048 |

mgr = df[df.jobcat\_name=='Manager']['prevexp'] cle = df[df.jobcat\_name=='Clerical']['prevexp'] cust = df[df.jobcat\_name=='Custodial']['prevexp']

from scipy import stats

f\_statistic, p\_value = stats.f\_oneway(mgr, cle, cust) print("F\_Statistic: {0}, P- Value: {1}".format(f\_statistic,p\_value))

F\_Statistic: 69.19167101209159, P-Value: 4.515360685161322e-27

from statsmodels.formula.api import ols

model\_name = ols('prevexp ~ C(jobcat\_name)', data=df).fit() model\_name.summary()

<class 'statsmodels.iolib.summary.Summary'>"""

OLS Regression Results

===================================================================

==========

=

Dep. Variable: prevexp R- squared:0.227

|  |  |  |  |
| --- | --- | --- | --- |
| Model:  0.224 |  | OLS | Adj. R-squared: |
| Method: | Leas | t Squares | F-statistic: |
| 69.19  Date: | Fri, 11 | Mar 2022 | Prob (F-statistic): |
| 27 |  |  |  |
| Time: 2815.1 |  | 13:16:41 | Log-Likelihood: |
| No. Observations: |  | 474 | AIC: |
| 5636.  Df Residuals: |  | 471 | BIC: |
| 5649. |  |  |  |
| Df Model:  Covariance Type: |  | 2  nonrobust |  |

4.52e-

-

===================================================================

==========

==================

coef std err t P>|t|

[0.025 0.975]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| - |  |  |  |  |  |
| Intercept | 85.0386 | 4.836 | 17.584 | 0.000 |  |
| 75.535 94.542  C(jobcat\_name)[T.Custodial] | 213.0725 | 18.380 | 11.592 | 0.000 |  |
| 176.955 249.190 |  |  |  |  |  |
| C(jobcat\_name)[T.Manager] 29.342 14.503 | -7.4195 | 11.156 | -0.665 | 0.506 | - |
| =====================================================================  ========  = | | | | | |
| Omnibus: | 133.381 | Durbin-Watson: | |  |  |
| 1.817 |  |  |  |  |  |
| Prob(Omnibus): 277.084 | 0.000 | Jarque-Bera (JB): | |  |  |
| Skew: | 1.525 | Prob(JB): |  | 6.79e- | |
| 61  Kurtosis: | 5.175 | Cond. No. |  |  |  |
| 4.46 |  |  |  |  |  |
| =====================================================================  ========  = | | | | | |

Notes:

[1] Standard Errors assume that the covariance matrix of the errors iscorrectly specified. """

|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number: 10097 Sign: Date: / /** |
| **Student Name: Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.4 Apply linear regression Model techniques to predict the data on anydataset.** |

*# Apply linear regression Model techniques to predict the# data on any dataset.*

*# Import pandas package*

import pandas as pd

*# Find the current working directory*

import os os.getcwd ()

'/home/mitu'

*# Import dataset*

df = pd.read\_csv('Salary\_Data.csv')

|  |  |  |
| --- | --- | --- |
| df | YearsExperience |  |
|  |  | Salary |
| 0 | 1.1 | 39343 |
| 1 | 1.3 | 46205 |
| 2 | 1.5 | 37731 |
| 3 | 2.0 | 43525 |
| 4 | 2.2 | 39891 |
| 5 | 2.9 | 56642 |
| 6 | 3.0 | 60150 |
| 7 | 3.2 | 54445 |
| 8 | 3.2 | 64445 |
| 9 | 3.7 | 57189 |
| 10 | 3.9 | 63218 |
| 11 | 4.0 | 55794 |
| 12 | 4.0 | 56957 |
| 13 | 4.1 | 57081 |
| 14 | 4.5 | 61111 |
| 15 | 4.9 | 67938 |
| 16 | 5.1 | 66029 |
| 17 | 5.3 | 83088 |
| 18 | 5.9 | 81363 |
| 19 | 6.0 | 93940 |
| 20 | 6.8 | 91738 |
| 21 | 7.1 | 98273 |
| 22 | 7.9 | 101302 |
| 23 | 8.2 | 113812 |
| 24 | 8.7 | 109431 |
| 25 | 9.0 | 105582 |
| 26 | 9.5 | 116969 |
| 27 | 9.6 | 112635 |
| 28 | 10.3 | 122391 |
| 29 | 10.5 | 121872 |

df.sha pe(30, 2)

df.columns

Index(['YearsExperience', 'Salary'], dtype='object')

*# Input*

x = df['YearsExperience'].values

*# Output*

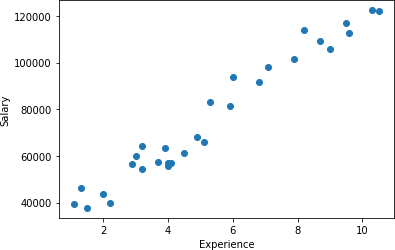
y = df['Salary'].values

|  |  |  |
| --- | --- | --- |
| df.corr() |  |  |
| YearsExperience | | Salary |
| YearsExperience | 1.000000 | 0.978242 |
| Salary | 0.978242 | 1.000000 |

import matplotlib.pyplot as plt

plt.xlabel('Experience') plt.ylabel('Salary') plt.scatter(x, y)

<matplotlib.collections.PathCollection at 0x7f97a5137310>



*# Import LR class*

from sklearn.linear\_model import LinearRegression

*# Create the object*

regressor = LinearRegression() x = x.reshape(-1,1)x

|  |  |
| --- | --- |
| array([[ | 1.1], |
| [ | 1.3], |
| [ | 1.5], |
| [ | 2. ], |
| [ | 2.2], |
| [ | 2.9], |
| [ | 3. ], |
| [ | 3.2], |
| [ | 3.2], |
| [ | 3.7], |
| [ | 3.9], |
| [ | 4. ], |
| [ | 4. ], |
| [ | 4.1], |
| [ | 4.5], |
| [ | 4.9], |
| [ | 5.1], |
| [ | 5.3], |
| [ | 5.9], |
| [ | 6. ], |
| [ | 6.8], |
| [ | 7.1], |
| [ | 7.9], |
| [ | 8.2], |
| [ | 8.7], |
| [ | 9. ], |
| [ | 9.5], |
| [ | 9.6], |

[10.3],

[10.5]])

*# Train the algorithm with data*

regressor.fit(x, y) LinearRegression()

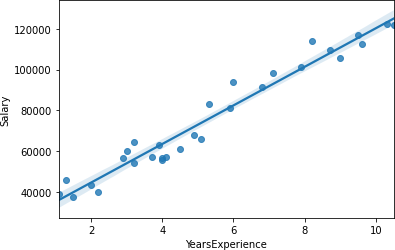
*# Prediction*

regressor.predict([[5]]) array([73042.01180594])

y\_pred = regressor.predict(x)

import seaborn as sns sns.regplot(x='YearsExperience', y='Salary', data=df)

<AxesSubplot:xlabel='YearsExperience', ylabel='Salary'>



result = pd.DataFrame({ 'Actual': y, 'Predicted': y\_pred

})

result

Actual Predicted0 39343

36187.158752

1 46205

38077.151217

2 37731

39967.143681

3 43525

44692.124842

4 39891

46582.117306

5 56642

53197.090931

6 60150

54142.087163

7 54445

56032.079627

8 64445

56032.079627

9 57189

60757.060788

10 63218

62647.053252

11 55794

63592.049484

12 56957

63592.049484

13 57081

64537.045717

14 61111

68317.030645

15 67938

72097.015574

16 66029

73987.008038

17 83088

75877.000502

18 81363

81546.977895

19 93940 82491.974127

20 91738 90051.943985

21 98273 92886.932681

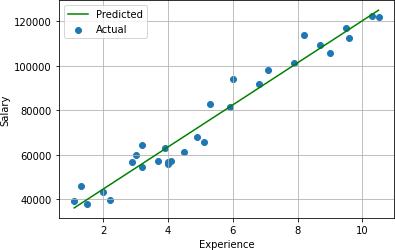
|  |  |
| --- | --- |
| 22 101302 | 100446.902538 |
| 23 113812 | 103281.891235 |
| 24 109431 | 108006.872395 |
| 25 105582 | 110841.861092 |
| 26 116969 | 115566.842252 |
| 27 112635 | 116511.838485 |
| 28 122391 | 123126.812110 |
| 29 121872 | 125016.804574 |

plt.xlabel('Experience') plt.ylabel('Salary') plt.grid()

plt.scatter(x, y, label = 'Actual')

plt.plot(x, y\_pred, label = 'Predicted', color='g')plt.legend()

<matplotlib.legend.Legend at 0x7f97a618dd30>



regressor.coef\_ *# Slope of line*

array([9449.96232146])

regressor.intercept\_ *# y-intercept of line*

25792.200198668696

5 \* 9449.96232146 + 25792.200198668696

73042.0118059687

*# r2 score*

regressor.score(x, y) 0.95695666100975086

from sklearn.metrics import r2\_score, mean\_absolute\_error,mean\_absolute\_percentage\_error r2\_score(y, y\_pred)

0.956956661009750

86

mean\_absolute\_error(y, y\_pred) 4644.2012894435375

mean\_absolute\_percentage\_error(y, y\_pred) 0.07048034398306606

df = pd.read\_csv('mtcars.csv')

df.shape (32, 11)

*# input*

x = df[['disp','hp','wt']]

*# Output*

y = df['mpg']

|  |  |  |  |
| --- | --- | --- | --- |
| x |  |  |  |
|  | disp | hp | wt |
| 0 | 160.0 | 110 | 2.620 |
| 1 | 160.0 | 110 | 2.875 |
| 2 | 108.0 | 93 | 2.320 |
| 3 | 258.0 | 110 | 3.215 |
| 4 | 360.0 | 175 | 3.440 |

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | 225.0 | 105 | 3.460 |
| 6 | 360.0 | 245 | 3.570 |
| 7 | 146.7 | 62 | 3.190 |
| 8 | 140.8 | 95 | 3.150 |
| 9 | 167.6 | 123 | 3.440 |
| 10 | 167.6 | 123 | 3.440 |
| 11 | 275.8 | 180 | 4.070 |
| 12 | 275.8 | 180 | 3.730 |
| 13 | 275.8 | 180 | 3.780 |
| 14 | 472.0 | 205 | 5.250 |
| 15 | 460.0 | 215 | 5.424 |
| 16 440.0 | | 230 5.345 | |
| 17 78.7 | | 66 2.200 | |
| 18 75.7 | | 52 1.615 | |
| 19 71.1 | | 65 1.835 | |
| 20 120.1 | | 97 2.465 | |
| 21 318.0 | | 150 3.520 | |
| 22 304.0 | | 150 3.435 | |
| 23 350.0 | | 245 3.840 | |
| 24 400.0 | | 175 3.845 | |
| 25 79.0 | | 66 1.935 | |
| 26 120.3 | | 91 2.140 | |
| 27 95.1 | | 113 1.513 | |
| 28 351.0 | | 264 3.170 | |
| 29 145.0 | | 175 2.770 | |
| 30 301.0 | | 335 3.570 | |
| 31 121.0 | | 109 2.780 | |

regressor = LinearRegression() regressor.fit(x, y) LinearRegression() regressor.intercept\_ 37.10550526903182

regressor.coef\_

array([-9.37009081e-04, -3.11565508e-02, -3.80089058e+00])

*# r2 Score*

regressor.score(x, y) 0.8268361424946447

*# prediction*

new = [[221, 102, 3.81]]

regressor.predict(new) array([19.23906496]) new = [[211, 134, 2.81]]

regressor.predict(new) array([22.052316])

x.corrwith(y)

|  |  |
| --- | --- |
| disp | -0.847551 |
| hp | -0.776168 |
| wt | -0.867659 |
| dtype: | float64 |

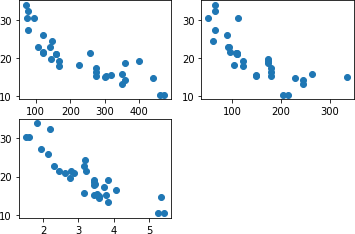
y\_pred = regressor.predict(x) mean\_absolute\_error(y, y\_pred) 1.9070264019715124

r2\_score(y, y\_pred) 0.82683614249464

47

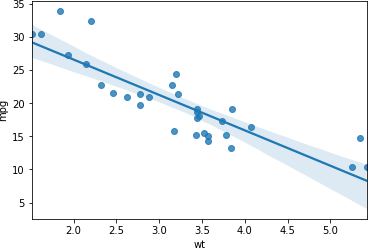
*# Data Visualization* plt.subplot(2,2,1) plt.scatter(x['disp'], y) plt.subplot(2,2,2) plt.scatter(x['hp'], y) plt.subplot(2,2,3) plt.scatter(x['wt'], y)

<matplotlib.collections.PathCollection at 0x7f97a6219970>



sns.regplot(x='wt', y='mpg', data=pd.read\_csv('mtcars.csv'))

<AxesSubplot:xlabel='wt', ylabel='mpg'>



|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number: 10097 Sign: : Date: / /** |
| **Student Name: Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.5 Apply logical regression Model techniques to predict the data on any dataset.** |

# Program:

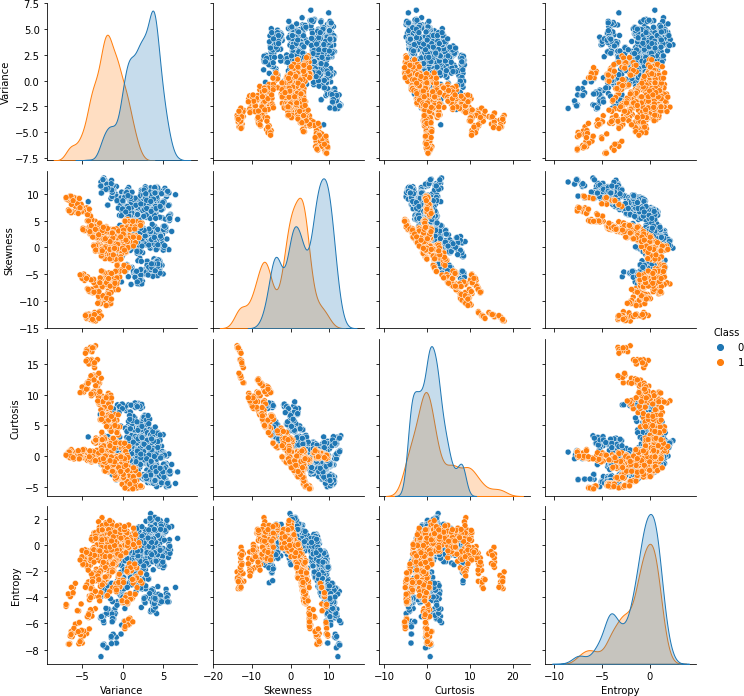
import pandas as pd

*# Data import*

df = pd.read\_csv('banknotes.csv')

import seaborn as sns sns.pairplot(df, hue='Class')

<seaborn.axisgrid.PairGrid at 0x7f42a29c9310>



*# output*

y = df['Class'] x.shape (1372, 4)

*# Cross validation*

from sklearn.model\_selection import train\_test\_splitx\_train, x\_test, y\_train, y\_test = train\_test\_split(

x, y, random\_state=0, test\_size=0.25) x\_train.head()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Variance | Skewness | Curtosis | Entropy |
| 662 | 2.9736 | 8.7944 | -3.6359 | -1.375400 |
| 512 | 2.6648 | 10.7540 | -3.3994 | -4.168500 |
| 1193 | -3.7573 | -8.2916 | 10.3032 | 0.380590 |
| 682 | 3.7321 | -3.8840 | 3.3577 | -0.006049 |
| 1313 | -1.5078 | -7.3191 | 7.8981 | 1.228900 |

x\_train.shape (1029, 4)

*# Import the class*

from sklearn.linear\_model import LogisticRegression

*# Create the object*

classifier = LogisticRegression()

*# Train the algorithm*

classifier.fit(x\_train, y\_train) LogisticRegression() x\_test.shape (343, 4)

*# Predict on the test data*

y\_pred = classifier.predict(x\_test)set(y)

{0, 1}

y.value\_counts()0 762

1 610

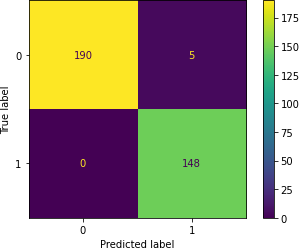
Name: Class, dtype: int64

result = pd.DataFrame({ 'Actual': y\_test, 'Predicted': y\_pred

})

|  |  |  |
| --- | --- | --- |
| result |  |  |
| Actual | | Predicted |
| 1023 | 1 | 1 |
| 642 | 0 | 0 |
| 1196 | 1 | 1 |
| 31 | 0 | 0 |
| 253 | 0 | 0 |
| ... | ... | ... |
| 866 | 1 | 1 |
| 361 | 0 | 0 |
| 703 | 0 | 0 |
| 328 | 0 | 0 |
| 530 | 0 | 0 |

[343 rows x 2 columns]

fromsklearn.metricsimport plot\_confusion\_matrix, accuracy\_score plot\_confusion\_matrix(classifier, x\_test, y\_test); y\_test.value\_counts()

0 195

1 148

Name: Class, dtype: int64 accuracy\_score(y\_test, y\_pred)0.9854227405247813 new1 = [[0.7057,- 5.4981,8.3368,-2.8715]]

new2 = [[-0.4665,2.3383,-2.9812,-1.0431]]

classifier.predict(new1) array([0]) classifier.predict\_proba(new1) array([[0.99724553, 0.00275447]]) classifier.predict(new2) array([1]) classifier.predict\_proba(new2) array([[6.24842128e-04, 9.99375158e-01]])

|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **RollNumber:10097 Sign:**  **Date: / /** |
| **StudentName:Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.6 Clustering algorithms for unsupervised classification.** |

**Program:**

import pandas as pd

df = pd.read\_csv('/home/mitu/Mall\_Customers.csv')df.shape (200, 5) list(df.columns)

['CustomerID', 'Genre', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']

*# Input data*

x = df.iloc[:,3:]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x |  |  |  |  |  |
| Annual | Incom e | (k$) | Spending | Scor e | (1-  100) |
| 0 |  | 15 |  |  | 39 |
| 1 |  | 15 |  |  | 81 |
| 2 |  | 16 |  |  | 6 |
| 3 |  | 16 |  |  | 77 |
| 4 |  | 17 |  |  | 40 |
| .. 195 |  | ... 120 |  |  | ... 79 |
| 196 |  | 126 |  |  | 28 |
| 197 |  | 126 |  |  | 74 |
| 198 |  | 137 |  |  | 18 |
| 199 |  | 137 |  |  | 83 |

[200 rows x 2 columns]

*# Summerize*

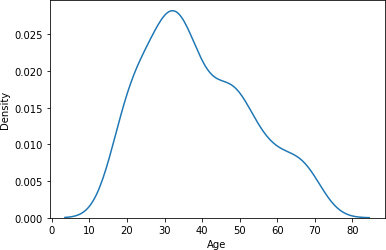
df.describe()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | CustomerID | Age | Annual | Income (k$) | Spending | Score (1- |
| count | 200.000000 | 200.000000 |  | 200.000000 |  | 100) |
|  |  |  |  |  |  | 200.000000 |
| mean | 100.500000 | 38.850000 |  | 60.560000 |  | 50.200000 |
| std | 57.879185 | 13.969007 |  | 26.264721 |  | 25.823522 |
| min | 1.000000 | 18.000000 |  | 15.000000 |  | 1.000000 |
| 25% | 50.750000 | 28.750000 |  | 41.500000 |  | 34.750000 |
| 50% | 100.500000 | 36.000000 |  | 61.500000 |  | 50.000000 |
| 75% | 150.250000 | 49.000000 |  | 78.000000 |  | 73.000000 |

*# import seaborn package*

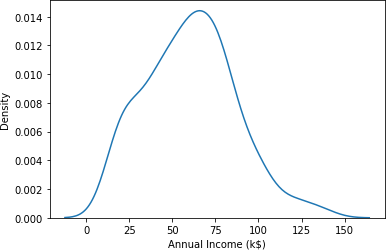
import seaborn as sns sns.kdeplot(df['Age'])

<AxesSubplot:xlabel='Age', ylabel='Density'>



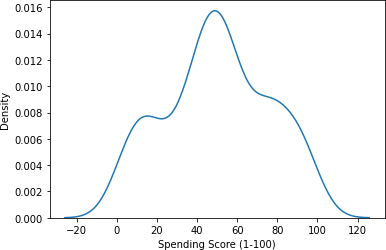
sns.kdeplot(df['Annual Income (k$)'])

<AxesSubplot:xlabel='Annual Income (k$)', ylabel='Density'

>

sns.kdeplot(df['Spending Score (1-100)'])

<AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Density'>



sns.boxplot(df['Age'])

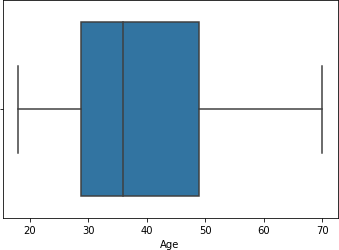
/home/mitu/.local/lib/python3.8/site-packages/seaborn/\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version

0.12, the only valid positional argument will be `data`, and passing otherarguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

(

<AxesSubplot:xlabel='Age'>

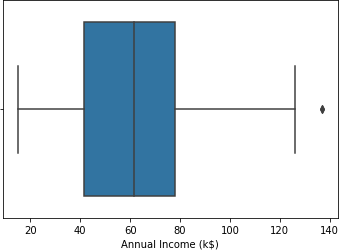


sns.boxplot(df['Annual Income (k$)'])

/home/mitu/.local/lib/python3.8/site-packages/seaborn/\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

<AxesSubplot:xlabel='Annual Income (k$)'>

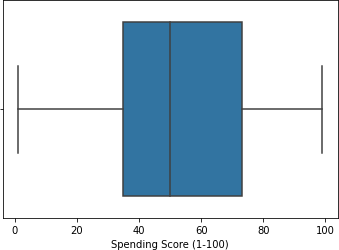


sns.boxplot(df['Spending Score (1-100)'])

/home/mitu/.local/lib/python3.8/site-packages/seaborn/\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

<AxesSubplot:xlabel='Spending Sc

ore (1-100)

*# Import the class*

from sklearn.cluster import KMeans

*# Create the object*

km = KMeans(n\_clusters=12, random\_state=0)

*# Train the algorithm*

labels = km.fit\_predict(x)

*# Sum of squared errors*

km.inertia\_ 15810.838613705504

*# elbow method*

sse = []

**for** k **in** range(1,41):

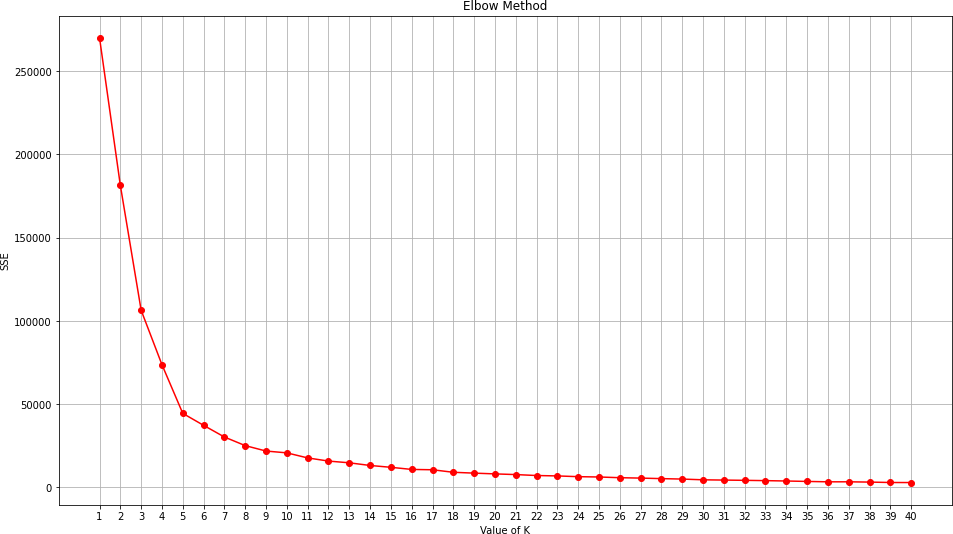
km = KMeans(n\_clusters=k, random\_state=0) labels = km.fit\_predict(x) sse.append(km.inertia\_)

import matplotlib.pyplot as plt plt.figure(figsize=(16,9)) plt.title('Elbow Method') plt.xlabel('Value of K') plt.ylabel('SSE')

plt.grid() plt.xticks(range(1,41))

plt.plot(range(1,41), sse, marker='o', color='r')

[<matplotlib.lines.Line2D at 0x7f9dac6b3070>]



*# Silhoutte method*

from sklearn.metrics import silhouette\_score

silh = []

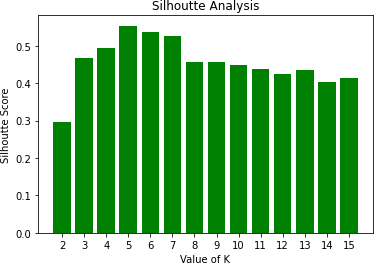
**for** k **in** range(2,16):

km = KMeans(n\_clusters=k, random\_state=0) labels = km.fit\_predict(x)

score = silhouette\_score(x, labels) silh.append(score)

*# plot the silhoutte scores* plt.title('Silhoutte Analysis') plt.xlabel('Value of K') plt.ylabel('Silhoutte Score') plt.xticks(range(2,16)) plt.bar(range(2,16), silh, color='g')

<BarContainer object of 14 ar tists>



*# Create the object*

km = KMeans(n\_clusters=5, random\_state=0)

*# Train the algorithm*

labels = km.fit\_predict(x)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| labels |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| array([4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, |
| 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 1, |
| 4, | 3, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 2, | 0, | 2, | 1, | 2, | 0, | 2, | 0, | 2, |
| 1, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 1, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, |
| 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, |
| 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, |

0, 2], dtype=int32)

*# Cluster labels*

km.labels\_

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| array([4  , | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, |
| 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 1, |
| 4, | 3, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 2, | 0, | 2, | 1, | 2, | 0, | 2, | 0, | 2, |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 1, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, |
|  |  |  |  |  |  |  | 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,  0, 2, 0,  0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,  0, 2, 0,  0, 2], dtype=int32) | | | | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 0, | 2, 0, 2, 0, 2, 0, 2, 0, 2, | 0, 2, 0, 2, | 0, 2, 0, 2, 0, 2, 0, 2, |
| 0, | 2, 0, 2, 0, 2, 0, 2, 0, 2, | 0, 2, 0, 2, | 0, 2, 0, 2, 0, 2, 0, 2, |
| 0, | 2], dtype=int32) |  |  |

*# SSE*

km.inertia\_ 44448.45544793369

*# Centroids*

km.cluster\_centers\_

array([[88.2 , 17.11428571],

|  |  |
| --- | --- |
| [55.2962963 , | 49.51851852], |
| [86.53846154, | 82.12820513], |
| [25.72727273, | 79.36363636], |
| [26.30434783, | 20.91304348]]) |

*# Extract the clusters*

df[labels==2] *# Boolean filtering*

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)123

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 124 | Male | 39 | 69 | 91 |
| 125 | 126 | Female | 31 | 70 | 77 |
| 127 | 128 | Male | 40 | 71 | 95 |
| 129 | 130 | Male | 38 | 71 | 75 |
| 131 | 132 | Male | 39 | 71 | 75 |
| 133 | 134 | Female | 31 | 72 | 71 |
| 135 | 136 | Female | 29 | 73 | 88 |
| 137 | 138 | Male | 32 | 73 | 73 |
| 139 | 140 | Female | 35 | 74 | 72 |
| 141 | 142 | Male | 32 | 75 | 93 |
| 143 | 144 | Female | 32 | 76 | 87 |
| 145 | 146 | Male | 28 | 77 | 97 |
| 147 | 148 | Female | 32 | 77 | 74 |
| 149 | 150 | Male | 34 | 78 | 90 |
| 151 | 152 | Male | 39 | 78 | 88 |
| 153 | 154 | Female | 38 | 78 | 76 |
| 155 | 156 | Female | 27 | 78 | 89 |
| 157 | 158 | Female | 30 | 78 | 78 |
| 159 | 160 | Female | 30 | 78 | 73 |
| 161 | 162 | Female | 29 | 79 | 83 |
| 163 | 164 | Female | 31 | 81 | 93 |
| 165 | 166 | Female | 36 | 85 | 75 |
| 167 | 168 | Female | 33 | 86 | 95 |
| 169 | 170 | Male | 32 | 87 | 63 |
| 171 | 172 | Male | 28 | 87 | 75 |
| 173 | 174 | Male | 36 | 87 | 92 |
| 175 | 176 | Female | 30 | 88 | 86 |
| 177 | 178 | Male | 27 | 88 | 69 |
| 179 | 180 | Male | 35 | 93 | 90 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 181 | 182 | Female | 32 | 97 | 86 |
| 183 | 184 | Female | 29 | 98 | 88 |
| 185 | 186 | Male | 30 | 99 | 97 |
| 187 | 188 | Male | 28 | 101 | 68 |
| 189 | 190 | Female | 36 | 103 | 85 |
| 191 | 192 | Female | 32 | 103 | 69 |
| 193 | 194 | Female | 38 | 113 | 91 |
| 195 | 196 | Female | 35 | 120 | 79 |
| 197 | 198 | Male | 32 | 126 | 74 |
| 199 | 200 | Male | 30 | 137 | 83 |

one = df[labels==1] one.shape

(81, 5)

*# Export the cluster*

one.to\_csv('one.csv')

print('Cluster-0:', len(df[labels==0]))

print('Cluster-1:', len(df[labels==1]))

print('Cluster-2:', len(df[labels==2]))

print('Cluster-3:', len(df[labels==3])) print('Cluster-4:', len(df[labels==4]))

Cluster-0: 35

Cluster-1: 81

Cluster-2: 39

Cluster-3: 22

Cluster-4: 23

*# Prediction*

new = [[45, 76]]

km.predict(new)[0]3

*# Prediction*

new = [[25, 36]]

km.predict(new)[0]4

*# Prediction*

new = [[85, 76]]

km.predict(new)[0]2

*# Prediction*

new = [[45, 47]]

km.predict(new)[0]1

from sklearn.preprocessing import LabelEncoderle = LabelEncoder()

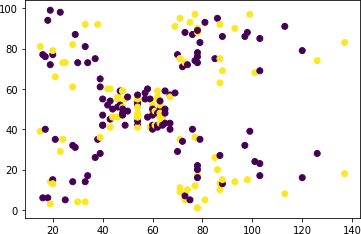
df['Genre'] = le.fit\_transform(df['Genre'])import matplotlib.pyplot as plt

df['Genre']

|  |  |
| --- | --- |
| 0 | 1 |
| 1 | 1 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 195 | .. 0 |
| 196 | 0 |
| 197 | 1 |
| 198 | 1 |
| 199 | 1 |
| Name: | Genre, Length: 200, dtype: int64 |

plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'],c=df['Genre'])

<matplotlib.collections.PathCollection at 0x7f9dac6e2fd0>



|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number:10097 Sign: Date: / /** |
| **Student Name: Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.7 Association algorithms for supervised classification on any dataset.** |

# Program:

dataset = [['Apple', 'Beer', 'Rice', 'Chicken'],

['Apple', 'Beer', 'Rice'],

['Apple', 'Beer'],

['Apple', 'Pear'],

['Milk', 'Beer', 'Rice', 'Chicken'],

['Milk', 'Beer', 'Rice'],

['Milk', 'Beer'],

['Apple', 'Pear']]

dataset

[['Apple', 'Beer', 'Rice', 'Chicken'],

['Apple', 'Beer', 'Rice'],

['Apple', 'Beer'],

['Apple', 'Pear'],

['Milk', 'Beer', 'Rice', 'Chicken'],

['Milk', 'Beer', 'Rice'],

['Milk', 'Beer'],

['Apple', 'Pear']]

*# Import the transaction encoder*

from mlxtend.preprocessing import TransactionEncoder

*# Create the object*

trans = TransactionEncoder()

*# Apply the operation*

df\_t = trans.fit\_transform(dataset)df\_t

array([[ True, True, True, False, False, True],[ True, True, False, False, False, True],[ True, True, False, False, False, False],[ True, False, False, False, True, False], [False, True, True, True, False, True], [False, True, False, True, False, True],[False, True, False, True, False, False],[ True, False, False, False, True, False]])

trans.columns\_

['Apple', 'Beer', 'Chicken', 'Milk', 'Pear', 'Rice']import pandas as pd

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| df |  |  |  |  |  |  |
|  | Apple | Beer | Chicken | Milk | Pear | Rice |
| 0 | True | True | True | False | False | True |
| 1 | True | True | False | False | False | True |
| 2 | True | True | False | False | False | False |
| 3 | True | False | False | False | True | False |
| 4 | False | True | True | True | False | True |
| 5 | False | True | False | True | False | True |
| 6 | False | True | False | True | False | False |
| 7 | True | False | False | False | True | False |
| *#* | *Support* | *count* |  |  |  |  |

sum(df['Rice']) / len(df)0.5

*# Generate frequent itemsets*

from mlxtend.frequent\_patterns import apriori

freq\_itemset = apriori(df, min\_support=0.25, use\_colnames=True)freq\_itemset support itemsets

0 0.625 (Apple)

1 0.750 (Beer)

2 0.250 (Chicken)

3 0.375 (Milk)

4 0.250 (Pear)

5 0.500 (Rice)

1. 0.375 (Beer, Apple)
2. 0.250 (Pear, Apple)
3. 0.250 (Rice, Apple)
4. 0.250 (Beer, Chicken)
5. 0.375 (Beer, Milk)
6. 0.500 (Beer, Rice)
7. 0.250 (Rice, Chicken)
8. 0.250 (Rice, Milk)
9. 0.250 (Beer, Rice, Apple)
10. 0.250 (Beer, Rice, Chicken)
11. 0.250 (Beer, Rice, Milk)

*# Generate strong association rules*

from mlxtend.frequent\_patterns import association\_rules

rules = association\_rules(freq\_itemset,

metric='confidence', min\_threshold=0.5)

rules

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | antecedents | consequents ... leverage conviction0 | | | | |
|  | (Beer) |  | (Apple) ... -0.09375 | | | 0.750 |
| 1 | (Apple) |  | (Beer) ... -0.09375 | | | 0.625 |
| 2 | (Pear) |  | (Apple) ... 0.09375 | | | inf |
| 3 | (Rice) |  | (Apple) ... -0.06250 | | | 0.750 |
| 4 | (Chicken) |  | (Beer) ... 0.06250 | | | inf |
| 5 | (Beer) |  | (Milk) ... 0.09375 | | | 1.250 |
| 6 | (Milk) |  | (Beer) ... 0.09375 | | | inf |
| 7 | (Beer) |  | (Rice) ... 0.12500 | | | 1.500 |
| 8 | (Rice) |  | (Beer) ... 0.12500 | | | inf |
| 9 | (Rice) | (Chicken) ... 0.12500 | | | | 1.500 |
| 10 | (Chicken) |  | (Rice) ... 0.12500 | | | inf |
| 11 | (Rice) |  | (Milk) ... 0.06250 | | | 1.250 |
| 12 | (Milk) |  | (Rice) ... 0.06250 | | | 1.500 |
| 13 | (Beer, Rice) |  | (Apple) | ... -0.06250 0.750 | | |
| 14 | (Beer, Apple) |  | (Rice) | ... 0.06250 1.500 | | |
| 15 | (Rice, Apple) |  | (Beer) | ... 0.06250 inf | | |
| 16 | (Rice) | (Beer, Apple) .. | |  | 0.06250 | 1.250 |
| 17 (Beer, Rice) | | (Chicken) ... | | | 0.12500 | 1.500 |
| 18 (Beer, Chicken) | |  | (Rice) ... | | 0.12500 | inf |
| 19 (Rice, Chicken) | |  | (Beer) ... | | 0.06250 | inf |
| 20 (Rice) (Beer, Chicken) ... | | | 0.12500 |  | 1.500 |  |
| 21 | (Chicken) | (Beer, Rice) ... | |  | 0.12500 | inf |
| 22 | (Beer, Rice) |  | (Milk) ... | | 0.06250 | 1.250 |
| 23 | (Beer, Milk) |  | (Rice) ... | | 0.06250 | 1.500 |
| 24 | (Rice, Milk) |  | (Beer) ... | | 0.06250 | inf |
| 25 | (Rice) | (Beer, Milk) ... | |  | 0.06250 | 1.250 |
| 26 | (Milk) | (Beer, Rice) ... | |  | 0.06250 | 1.500 |

[27 rows x 9 columns]

rules = rules[['antecedents','consequents','support','confidence']]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| rules |  |  |  |  |
|  | antecedents | consequents | support | confidence |
| 0 | (Beer) | (Apple) | 0.375 | 0.500000 |
| 1 | (Apple) | (Beer) | 0.375 | 0.600000 |
| 2 | (Pear) | (Apple) | 0.250 | 1.000000 |
| 3 | (Rice) | (Apple) | 0.250 | 0.500000 |
| 4 | (Chicken) | (Beer) | 0.250 | 1.000000 |
| 5 | (Beer) | (Milk) | 0.375 | 0.500000 |
| 6 | (Milk) | (Beer) | 0.375 | 1.000000 |
| 7 | (Beer) | (Rice) | 0.500 | 0.666667 |
| 8 | (Rice) | (Beer) | 0.500 | 1.000000 |
| 9 | (Rice) | (Chicken) | 0.250 | 0.500000 |
| 10 | (Chicken) | (Rice) | 0.250 | 1.000000 |
| 11 | (Rice) | (Milk) | 0.250 | 0.500000 |
| 12 | (Milk) | (Rice) | 0.250 | 0.666667 |
| 13 | (Beer, Rice) | (Apple) | 0.250 | 0.500000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 14 | (Beer, | Apple) | (Rice) | 0.250 | 0.666667 |
| 15 | (Rice, | Apple) | (Beer) | 0.250 | 1.000000 |
| 16 |  | (Rice) | (Beer, Apple) | 0.250 | 0.500000 |
| 17 | (Beer, Rice) | | (Chicken) | 0.250 | 0.500000 |
| 18 | (Beer, Chicken) | | (Rice) | 0.250 | 1.000000 |
| 19 | (Rice, Chicken) | | (Beer) | 0.250 | 1.000000 |
| 20 |  | (Rice) | (Beer, Chicken) | 0.250 | 0.500000 |
| 21 | (Chicken) | | (Beer, Rice) | 0.250 | 1.000000 |
| 22 | (Beer, Rice) | | (Milk) | 0.250 | 0.500000 |
| 23 | (Beer, Milk) | | (Rice) | 0.250 | 0.666667 |
| 24 | (Rice, Milk) | | (Beer) | 0.250 | 1.000000 |
| 25 |  | (Rice) | (Beer, Milk) | 0.250 | 0.500000 |
| 26 |  | (Milk) | (Beer, Rice) | 0.250 | 0.666667 |

rules['antecedent\_len'] = rules['antecedents'].apply(**lambda** x: len(x))

<ipython-input-24-514ef6b1bde9>:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.Try using

.loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas- docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy rules['antecedent\_len'] = rules['antecedents'].apply(lambda x: len(x))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| rules  antecedents | consequents | support | confidence | antecedent\_len |
| 0 (Beer) | (Apple) | 0.375 | 0.500000 | 1 |
| 1 (Apple) | (Beer) | 0.375 | 0.600000 | 1 |
| 2 (Pear) | (Apple) | 0.250 | 1.000000 | 1 |
| 3 (Rice) | (Apple) | 0.250 | 0.500000 | 1 |
| 4 (Chicken) | (Beer) | 0.250 | 1.000000 | 1 |
| 5 (Beer) | (Milk) | 0.375 | 0.500000 | 1 |
| 6 (Milk) | (Beer) | 0.375 | 1.000000 | 1 |
| 7 (Beer) | (Rice) | 0.500 | 0.666667 | 1 |
| 8 (Rice) | (Beer) | 0.500 | 1.000000 | 1 |
| 9 (Rice) | (Chicken) | 0.250 | 0.500000 | 1 |
| 10 (Chicken) | (Rice) | 0.250 | 1.000000 | 1 |
| 11 (Rice) | (Milk) | 0.250 | 0.500000 | 1 |
| 12 (Milk) | (Rice) | 0.250 | 0.666667 | 1 |
| 13 (Beer, Rice) | (Apple) | 0.250 | 0.500000 | 2 |
| 14 (Beer, Apple) | (Rice) | 0.250 | 0.666667 | 2 |
| 15 (Rice, Apple) | (Beer) | 0.250 | 1.000000 | 2 |
| 16 (Rice) | (Beer, Apple) | 0.250 | 0.500000 | 1 |
| 17 (Beer, Rice) | (Chicken) | 0.250 | 0.500000 | 2 |
| 18 (Beer, Chicken) | (Rice) | 0.250 | 1.000000 | 2 |
| 19 (Rice, Chicken) | (Beer) | 0.250 | 1.000000 | 2 |
| 20 (Rice) | (Beer, Chicken) | 0.250 | 0.500000 | 1 |
| 21 (Chicken) | (Beer, Rice) | 0.250 | 1.000000 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 24 | (Rice, Milk) | (Beer) | 0.250 | 1.000000 | 2 |
| 25 | (Rice) | (Beer, Milk) | 0.250 | 0.500000 | 1 |
| 26 | (Milk) | (Beer, Rice) | 0.250 | 0.666667 | 1 |

nrules = rules[(rules['antecedent\_len'] == 1) &

(rules['support'] > 0.30)]

nrules

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | antecedents  (Beer) | consequents  (Apple) | support 0.375 | confidence 0.500000 | antecedent\_len  1 |
| 1 | (Apple) | (Beer) | 0.375 | 0.600000 | 1 |
| 5 | (Beer) | (Milk) | 0.375 | 0.500000 | 1 |
| 6 | (Milk) | (Beer) | 0.375 | 1.000000 | 1 |
| 7 | (Beer) | (Rice) | 0.500 | 0.666667 | 1 |
| 8 | (Rice) | (Beer) | 0.500 | 1.000000 | 1 |

*# Prediction / Suggestion / Recommendation*

nrules[nrules['antecedents'] == {'Apple'}]['consequents'][1] frozenset({'Beer'})

rules.sort\_values(by='confidence', ascending=False)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 18 | antecedents (Beer, Chicken) | consequents  (Rice) | support 0.250 | confidence 1.000000 | antecedent\_len  2 |
| 2 | (Pear) | (Apple) | 0.250 | 1.000000 | 1 |
| 21 | (Chicken) | (Beer, Rice) | 0.250 | 1.000000 | 1 |
| 4 | (Chicken) | (Beer) | 0.250 | 1.000000 | 1 |
| 24 | (Rice, Milk) | (Beer) | 0.250 | 1.000000 | 2 |
| 6 | (Milk) | (Beer) | 0.375 | 1.000000 | 1 |
| 15 | (Rice, Apple) | (Beer) | 0.250 | 1.000000 | 2 |
| 8 | (Rice) | (Beer) | 0.500 | 1.000000 | 1 |
| 19 | (Rice, Chicken) | (Beer) | 0.250 | 1.000000 | 2 |
| 10 | (Chicken) | (Rice) | 0.250 | 1.000000 | 1 |
| 12 | (Milk) | (Rice) | 0.250 | 0.666667 | 1 |
| 14 | (Beer, Apple) | (Rice) | 0.250 | 0.666667 | 2 |
| 26 | (Milk) | (Beer, Rice) | 0.250 | 0.666667 | 1 |
| 7 | (Beer) | (Rice) | 0.500 | 0.666667 | 1 |
| 23 | (Beer, Milk) | (Rice) | 0.250 | 0.666667 | 2 |
| 1 | (Apple) | (Beer) | 0.375 | 0.600000 | 1 |
| 22 | (Beer, Rice) | (Milk) | 0.250 | 0.500000 | 2 |
| 25 | (Rice) | (Beer, Milk) | 0.250 | 0.500000 | 1 |
| 20 | (Rice) | (Beer, Chicken) | 0.250 | 0.500000 | 1 |
| 0 | (Beer) | (Apple) | 0.375 | 0.500000 | 1 |
| 17 | (Beer, Rice) | (Chicken) | 0.250 | 0.500000 | 2 |
| 16 | (Rice) | (Beer, Apple) | 0.250 | 0.500000 | 1 |
| 11 | (Rice) | (Milk) | 0.250 | 0.500000 | 1 |
| 9 | (Rice) | (Chicken) | 0.250 | 0.500000 | 1 |
| 5 | (Beer) | (Milk) | 0.375 | 0.500000 | 1 |

|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number: 10097 Sign: Date: / /** |
| **Student Name: Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.8 Developing and implementing Decision Tree model on the dataset** |

# Program:

import pandas as pd

*# Data import*

df = pd.read\_csv('Social\_Network\_Ads.csv') df.shape

(400, 5)

*# input*

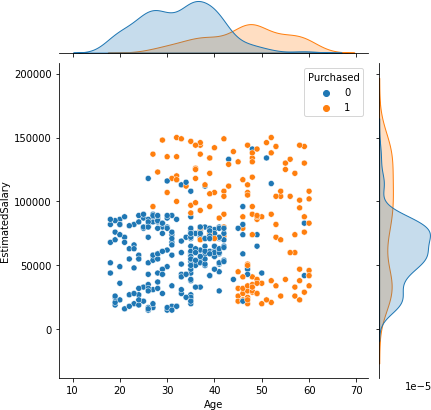
x = df[['Age','EstimatedSalary']]

*# output*

y = df['Purchased'] import seaborn as sns

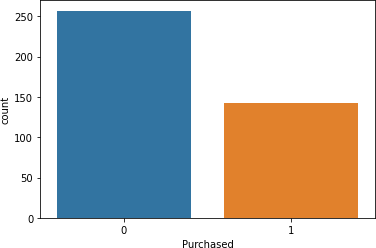
sns.jointplot(x='Age', y='EstimatedSalary', hue='Purchased', data=df)

<seaborn.axisgrid.JointGrid at 0x7fb1b1c5e9a0>



sns.countplot(x=y)

<AxesSubplot:xlabel='Purchased', ylabel='count'



y.value\_counts() 0 257

1 143

Name: Purchased, dtype: int64

*# Cross-validation*

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=0, test\_size=0.25)

x\_train.shape (300, 2)

x\_test.shape (100, 2)

*# Import the class*

from sklearn.tree import DecisionTreeClassifier

*# Create the object*

classifier = DecisionTreeClassifier(random\_state=0)

*# Train the algorithm with data*

*# Predictions*

y\_pred = classifier.predict(x\_test)

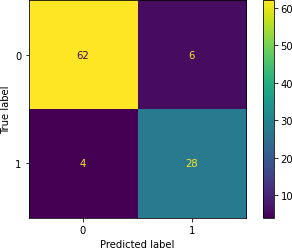
*# Combine the data*

result = pd.DataFrame({ 'Actual': y\_test, 'Predicted': y\_pred

})

result

|  |  |  |
| --- | --- | --- |
|  | Actual | Predicted |
| 132 | 0 | 0 |
| 309 | 0 | 0 |
| 341 | 0 | 0 |
| 196 | 0 | 0 |
| 246 | 0 | 0 |
| .. | ... | ... |
| 146 | 1 | 1 |
| 135 | 0 | 0 |
| 390 | 1 | 1 |
| 264 | 1 | 1 |
| 364 | 1 | 1 |
| [100 | rows x | 2 columns] |

from sklearn.metrics import plot\_confusion\_matrix, accuracy\_score plot\_confusion\_matrix(classifier, x\_test, y\_test);

accuracy\_score(y\_test, y\_pred)0.9

*# Single prediction*

new1 = [[34, 123000]]

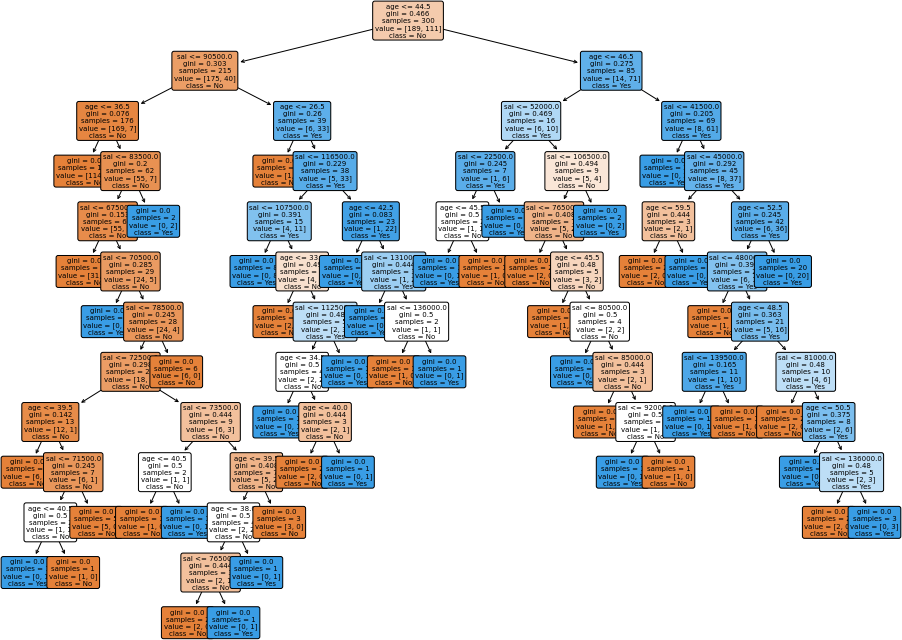
new2 = [[25, 48900]]

classifier.predict(new1)array([1]) classifier.predict(new2)array([0])

from sklearn.tree import plot\_treeimport matplotlib.pyplot as plt

plt.figure(figsize=(16,12))

plot\_tree(classifier, fontsize=7, feature\_names=['age','sal'],class\_names=['No','Yes'], filled=True, rounded=True);



|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number:10097 Sign: Date: / /** |
| **Student Name:Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title:Q.9 Bayesian classification on any dataset.** |

# Program:

*# Import packages* import pandas as pd import seaborn as sns

*# Data import*

df = pd.read\_csv('iris.csv')

*# The data shape*

df.shape (150, 5)

*# The columns names*

list(df.columns)

['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'species']

*# Let's describe*

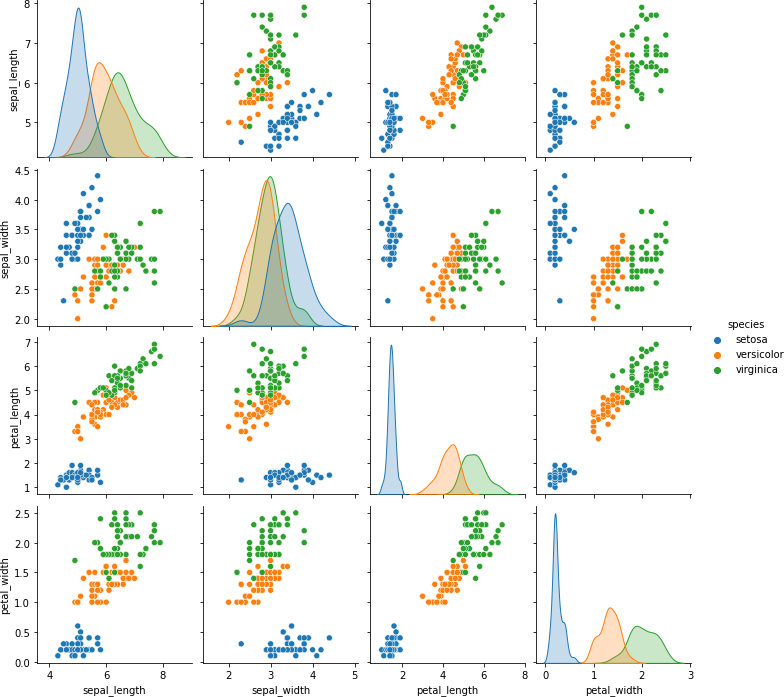
df.describe()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| count | sepal\_length 150.000000 | sepal\_width 150.000000 | petal\_length 150.000000 | petal\_width 150.000000 |
| mean | 5.843333 | 3.057333 | 3.758000 | 1.199333 |
| std | 0.828066 | 0.435866 | 1.765298 | 0.762238 |
| min | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| 25% | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| 50% | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| 75% | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| max | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

*# Check the clusters*

sns.pairplot(df, hue='species')

<seaborn.axisgrid.PairGrid at 0x7fa2f1e0df40>



*# input data*

x = df.drop('species', axis = 1)

*# output data*

y = df['species'] x.shape

(150, 4)

sns.countplot(x = y)

<AxesSubplot:xlabel='species', ylabel='count'>



y.value\_counts()

setosa 50

virginica 50

versicolor 50

Name: species, dtype: int64

*# Cross validation -> hold out method*

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=0, train\_size=0.75)

x\_train.shape (112, 4)

x\_test.shape (38, 4)

*# Import the class*

from sklearn.naive\_bayes import GaussianNB

*# Create the object*

classifier = GaussianNB()

*# Train the algorithm with data*

*# Predictions*

y\_pred = classifier.predict(x\_test)

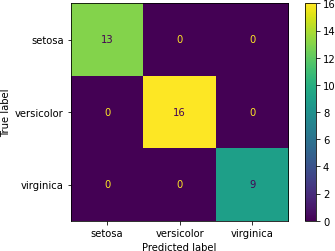
*# Import all functions*

from sklearn.metrics import plot\_confusion\_matrix, accuracy\_scorefrom sklearn.metrics import classification\_report

*# Plot the confusion matrix*

plot\_confusion\_matrix(classifier, x\_test, y\_test)

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at0x7fa2e177e6d0>



*# Accuracy*

accuracy\_score(y\_test, y\_pred)1.0

*# Classification report*

print(classification\_report(y\_test, y\_pred))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| setosa | 1.00 | 1.00 | 1.00 | 13 |
| versicolor | 1.00 | 1.00 | 1.00 | 16 |
| virginica | 1.00 | 1.00 | 1.00 | 9 |
| accuracy |  |  | 1.00 | 38 |
| macro avg | 1.00 | 1.00 | 1.00 | 38 |

weighted avg 1.00 1.00 1.00 38

*# Print the probabilities*

classifier.predict\_proba(x\_test)

array([[2.05841140e-233, 1.23816844e-006, 9.99998762e-001],

|  |  |  |
| --- | --- | --- |
| [1.76139943e-084, | 9.99998414e-001, | 1.58647449e-006], |
| [1.00000000e+000, | 1.48308613e-018, | 1.73234612e-027], |
| [6.96767669e-312, | 5.33743814e-007, | 9.99999466e-001], |
| [1.00000000e+000, | 9.33944060e-017, | 1.22124682e-026], |
| [4.94065646e-324, | 6.57075840e-011, | 1.00000000e+000], |
| [1.00000000e+000, | 1.05531886e-016, | 1.55777574e-026], |
| [2.45560284e-149, | 7.80950359e-001, | 2.19049641e-001], |
| [4.01160627e-153, | 9.10103555e-001, | 8.98964447e-002], |
| [1.46667004e-094, | 9.99887821e-001, | 1.12179234e-004], |
| [5.29999917e-215, | 4.59787449e-001, | 5.40212551e-001], |
| [4.93479766e-134, | 9.46482991e-001, | 5.35170089e-002], |
| [5.23735688e-135, | 9.98906155e-001, | 1.09384481e-003], |
| [4.97057521e-142, | 9.50340361e-001, | 4.96596389e-002], |
| [9.11315109e-143, | 9.87982897e-001, | 1.20171030e-002], |
| [1.00000000e+000, | 7.81797826e-019, | 1.29694954e-028], |
| [3.86310964e-133, | 9.87665084e-001, | 1.23349155e-002], |
| [2.27343573e-113, | 9.99940331e-001, | 5.96690955e-005], |
| [1.00000000e+000, | 1.80007196e-015, | 9.14666201e-026], |
| [1.00000000e+000, | 1.30351394e-015, | 8.42776899e-025], |
| [4.66537803e-188, | 1.18626155e-002, | 9.88137385e-001], |
| [1.02677291e-131, | 9.92205279e-001, | 7.79472050e-003], |
| [1.00000000e+000, | 6.61341173e-013, | 1.42044069e-022], |
| [1.00000000e+000, | 9.98321355e-017, | 3.50690661e-027], |
| [2.27898063e-170, | 1.61227371e-001, | 8.38772629e-001], |
| [1.00000000e+000, | 2.29415652e-018, | 2.54202512e-028], |
| [1.00000000e+000, | 5.99780345e-011, | 5.24260178e-020], |
| [1.62676386e-112, | 9.99340062e-001, | 6.59938068e-004], |
| [2.23238199e-047, | 9.99999965e-001, | 3.47984452e-008], |
| [1.00000000e+000, | 1.95773682e-013, | 4.10256723e-023], |
| [3.52965800e-228, | 1.15450262e-003, | 9.98845497e-001], |
| [3.20480410e-131, | 9.93956330e-001, | 6.04366979e-003], |
| [1.00000000e+000, | 1.14714843e-016, | 2.17310302e-026], |
| [3.34423817e-177, | 8.43422262e-002, | 9.15657774e-001], |
| [5.60348582e-264, | 1.03689515e-006, | 9.99998963e-001], |
| [7.48035097e-091, | 9.99950155e-001, | 4.98452400e-005], |
| [1.00000000e+000, | 1.80571225e-013, | 1.83435499e-022], |

[8.97496247e-182, 5.65567226e-001, 4.34432774e-001]]) new1 = [[5.1,3.7,1.5,0.4]]

new2 = [[6.8,2.8,4.8,1.4]]

new3 = [[7.7,2.6,6.9,2.3]]

*# Predictions*

classifier.predict(new1)[0]

'setosa' classifier.predict(new2)[0] 'versicolor' classifier.predict(new3)[0] 'virginica'

|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number: 10097 Sign: Date: / /** |
| **Student Name: Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.10 SVM classification on any dataset** |

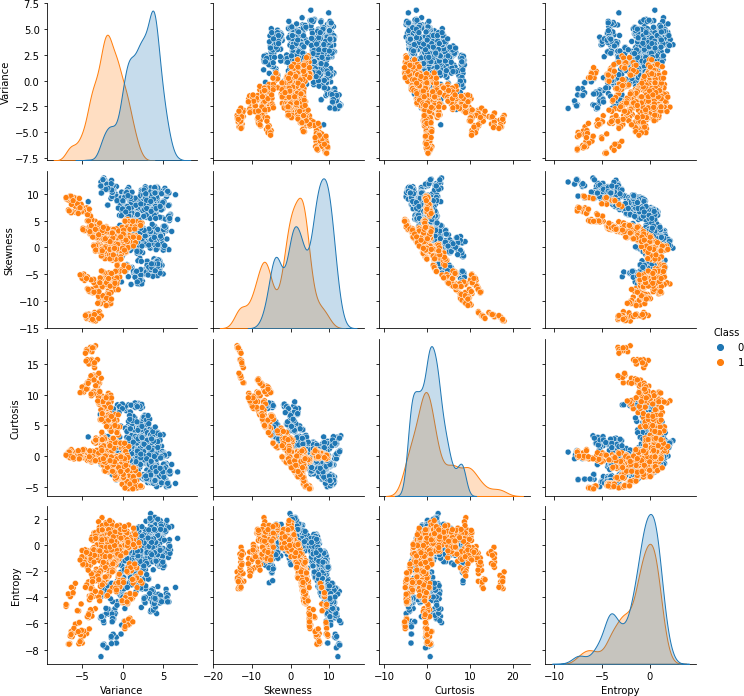
# Program:

import pandas as pd

*# Data import*

df = pd.read\_csv('banknotes.csv')import seaborn as sns sns.pairplot(df, hue='Class')

<seaborn.axisgrid.PairGrid at 0x7f03ccecc490>



*# Input data*

x.shape (1372, 4)

*# Cross - validation -> hold out method*

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, random\_state=0, test\_size=0.25)

x\_train.shape (1029, 4)

x\_test.shape (343, 4)

x\_train

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Variance | Skewness | Curtosis | Entropy |
| 662 | 2.97360 | 8.794400 | -3.635900 | -1.375400 |
| 512 | 2.66480 | 10.754000 | -3.399400 | -4.168500 |
| 1193 | -3.75730 | -8.291600 | 10.303200 | 0.380590 |
| 682 | 3.73210 | -3.884000 | 3.357700 | -0.006049 |
| 1313 | -1.50780 | -7.319100 | 7.898100 | 1.228900 |
| ... | ... | ... | ... | ... |
| 763 | 0.39012 | -0.142790 | -0.031994 | 0.350840 |
| 835 | -0.94255 | 0.039307 | -0.241920 | 0.315930 |
| 1216 | 0.60050 | 0.999450 | -2.212600 | 0.097399 |
| 559 | 2.01650 | -0.252460 | 5.170700 | 1.076300 |
| 684 | -2.07590 | 10.822300 | 2.643900 | -4.837000 |

[1029 rows x 4 columns] sns.countplot(x=y)

<AxesSubplot:xlabel='Class', ylabel='count'>



y.value\_counts()

0 762

1 610

Name: Class, dtype: int64 y\_train.value\_counts()

0 567

1 462

Name: Class, dtype: int64 y\_test.value\_counts()

0 195

1 148

Name: Class, dtype: int64

*# Import the SVM class*

from sklearn.svm import SVC

*# Create the object of SVC*

classifier = SVC(random\_state=0, kernel='sigmoid')

*# Train the algorithm*

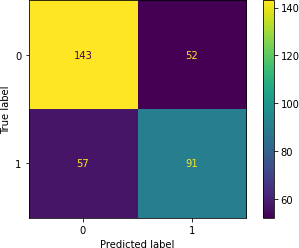
classifier.fit(x\_train, y\_train) SVC(kernel='sigmoid', random\_state=0)

*# Predictions*

y\_pred = classifier.predict(x\_test)

from sklearn.metrics import plot\_confusion\_matrix, classification\_reportfrom sklearn.metrics import accuracy\_score plot\_confusion\_matrix(classifier, x\_test, y\_test)

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at0x7f03a41518e0>



print(classification\_report(y\_test, y\_pred))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.71 | 0.73 | 0.72 | 195 |
| 1 | 0.64 | 0.61 | 0.63 | 148 |
| accuracy |  |  | 0.68 | 343 |
| macro avg | 0.68 | 0.67 | 0.67 | 343 |
| weighted avg | 0.68 | 0.68 | 0.68 | 343 |

accuracy\_score(y\_test, y\_pred) 0.6822157434402333

new1 = [[3.73210,-3.884000,3.357700,-0.006049]]

classifier.predict(new1)array([1])

*# Linear - 0.9854227405247813*

*# Polynomial - 0.967930029154519#*

*RBF - 0.9970845481049563*

*# Sigmoid - 0.6822157434402333*

|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number: 10097 Sign: Date: / /** |
| **Student Name:Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.11 Text Mining algorithms on unstructured dataset** |

# Program:

import pandas as pd

df = pd.read\_csv('SMSSpamCollection', sep='\t',

names = ['class','body\_text']) df

class body\_text

1. ham Go until jurong point, crazy.. Available only ...
2. ham Ok lar... Joking wif u oni...
3. spam Free entry in 2 a wkly comp to win FA Cup fina...
4. ham U dun say so early hor... U c already then say...
5. ham Nah I don't think he goes to usf, he lives aro...

... ... ...

5567 spam This is the 2nd time we have tried 2 contact u...5568 ham

Will ü b going to esplanade fr home?5569 ham Pity, \* was in mood for that. So...any other s...5570 ham The guy did some bitching but I acted like i'd...5571 ham Rofl. Its true to its name

[5572 rows x 2 columns] import string string.punctuation

'!"#$%&\'()\*+,-./:;<=>?@[\\]^\_`{|}~'

*# Function to count the punctuation symbols*

**def** count\_punct(text):

count = sum([1 **for** x **in** text **if** x **in** string.punctuation])

**return**(round(count/(len(text)-text.count(' '))\*100,2))s = 'Hello, friends! How are you? Welcome to Pune.!!!' count\_punct(s)

17.07

*# Add feature of punctuation percentages*

df['punct%'] = df['body\_text'].apply(**lambda** x: count\_punct(x))

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| df |  |  |  |  |  |  |
|  | class |  |  |  | body\_text | punct% |
| 0 | ham | Go | until jurong | point, | crazy.. Available only ... | 9.78 |
| 1 | ham |  |  | Ok | lar... Joking wif u oni... | 25.00 |
| 2 | spam | Free entry in 2 | | a wkly | comp to win FA Cup fina... | 4.69 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 ham U dun say so early hor... U c already then say... 15.38 | | | | | | | | | | |
| 4 | ham | Nah I | don't | think he | goes | to | usf, he | lives | aro... | 4.08 |
| ... | ... |  |  |  |  |  |  |  | ... | ... |
| 5567 spam This is the 2nd time we have tried 2 contact u... 6.11 | | | | | | | | | | |
| 5568 | ham |  |  | Will ü b going to esplanade fr home? | | | | |  | 3.45 |
| 5569 | ham | Pity, \* | was | in mood for that. So...any other s... | | | | |  | 14.58 |
| 5570 | ham | The guy | did | some bitching but I acted like i'd... | | | | |  | 1.00 |
| 5571 | ham |  |  |  | Rofl. Its true to its name | | | |  | 4.76 |

*# Add the column body length to it*

df['body\_len'] = df['body\_text'].apply(**lambda** x: len(x) - x.count(" ")) df

class body\_text punct% \

0 ham Go until jurong point, crazy.. Available only ... 9.78

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | ham | Ok lar... Joking wif u oni... | 25.00 |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | 4.69 |

3 ham U dun say so early hor... U c already then say... 15.38

4

...

ham

...

Nah I don't think he goes to usf, he lives aro...

...

4.08

...

5567 spam This is the 2nd time we have tried 2 contact u... 6.11

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5568 | ham |  | Will ü b going to esplanade fr home? | | | | |  | 3.45 |
| 5569 | ham | Pity, \* was | in mood for that. So...any other s... | | | |  |  | 14.58 |
| 5570 | ham | The guy did | some bitching but I acted like i'd... | | | |  |  | 1.00 |
| 5571 | ham |  | Rofl. | Its | true | to | its | name | 4.76 |

body\_len

|  |  |
| --- | --- |
| 0 | 92 |
| 1 | 24 |
| 2 | 128 |
| 3 | 39 |
| 4 | 49 |
| ... | ... |
| 5567 | 131 |
| 5568 | 29 |
| 5569 | 48 |
| 5570 | 100 |
| 5571 | 21 |

[5572 rows x 4 columns]

from nltk.corpus import stopwords s\_words

= stopwords.words('english')s\_words;

from nltk.stem import PorterStemmerps = PorterStemmer()

*# analyzer function*

**def** clean\_text(text):

data = [x **for** x **in** text **if** x **not in** string.punctuation]data = "".join(data)

data = [ps.stem(x) **for** x **in** data.split() **if** x **not in** s\_words]

**return** data clean\_text(s)

['hello', 'friend', 'how', 'welcom', 'pune']

|  |  |  |  |
| --- | --- | --- | --- |
| *#*  X  y  X | *Seperate the input and output*  = df.drop('class', axis = 1)  = df['class'] |  |  |
|  | body\_text | punct% | body\_len |
| 0 | Go until jurong point, crazy.. Available only ... | 9.78 | 92 |
| 1 | Ok lar... Joking wif u oni... | 25.00 | 24 |
| 2 | Free entry in 2 a wkly comp to win FA Cup fina... | 4.69 | 128 |
| 3 | U dun say so early hor... U c already then say... | 15.38 | 39 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4  ... | Nah I | don't | think he | goes | to | usf, he | lives | aro...  ... | 4.08  ... | 49  ... |
| 5567 This is the 2nd time we have tried 2 contact u... 6.11 131 | | | | | | | | | | |
| 5568 |  |  | Will ü b going to esplanade fr home? | | | | |  | 3.45 | 29 |
| 5569 | Pity, \* | was | in mood for that. So...any other s... | | | | |  | 14.58 | 48 |
| 5570 | The guy | did | some bitching but I acted like i'd... | | | | |  | 1.00 | 100 |
| 5571 |  |  |  | Rofl. Its true to its name | | | |  | 4.76 | 21 |

[5572 rows x 3 columns]

*# Import tfidf vectorizer*

from sklearn.feature\_extraction.text import TfidfVectorizertfidf = TfidfVectorizer(analyzer=clean\_text)

X\_trans = tfidf.fit\_transform(X['body\_text'])

X\_trans.shape (5572, 8277)

X\_vect = pd.concat([X[['body\_len', 'punct%']]

.reset\_index(drop=True), pd.DataFrame(X\_trans.toarray())], axis=1)

X\_vect.shape

(5572, 8279)

y.value\_counts()

ham 4825

spam 747

Name: class, dtype: int64 X\_vect.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 5572 entries, 0 to 5571

Columns: 8279 entries, body\_len to 8276dtypes:

float64(8278), int64(1) memory usage: 351.9 MB

*# Cross validation*

from sklearn.model\_selection import train\_test\_splitX\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_vect, y, stratify=y, random\_state=0)

X\_train.shape (4179, 8279)

from sklearn.ensemble import RandomForestClassifierclf = RandomForestClassifier(random\_state=0) clf.fit(X\_train, y\_train) RandomForestClassifier(random\_state=0)

y\_pred = clf.predict(X\_test)

from sklearn.metrics import accuracy\_score, classification\_reportaccuracy\_score(y\_test, y\_pred) 0.9662598707824839

print(classification\_report(y\_test, y\_pred))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| ham | 0.96 | 1.00 | 0.98 | 1206 |
| spam | 1.00 | 0.75 | 0.86 | 187 |
| accuracy |  |  | 0.97 | 1393 |
| macro avg | 0.98 | 0.87 | 0.92 | 1393 |
| weighted avg | 0.97 | 0.97 | 0.96 | 1393 |

new = pd.read\_csv("sample.csv",

names=['body\_text'], sep='\t')

new

body\_text

1. Ok lar i double check wif da hair dresser alre...
2. As a valued customer, I am pleased to advise y...
3. Today is "song dedicated day.." Which song wil...

new['body\_len'] = new['body\_text'].apply(**lambda** x: len(x) - x.count(" "))new['punct%'] = new['body\_text'].apply(**lambda** x: count\_punct(x))

new\_vect = tfidf.transform(new['body\_text']) sample\_vect = new

sample\_vect = pd.concat([new[['body\_len', 'punct%']].reset\_index(drop=True), pd.DataFrame(new\_vect.toarray())], axis=1)

sample\_vect.shape(3, 8279)

sample\_vect

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| body\_len | | | punct  % | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 ... 8267 | 8268 |
| \ |  |  |  |  |  |  |  |  |  |  |  |
| 0 |  | 89 | 4.49 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 1 | 125 | | 2.40 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
| 2 | 102 | | 9.80 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 ... 0.0 | 0.0 |
|  | 8269 | 8270 | 8271 | 8272 | 8273 | 8274 | 8275 | | 8276 |  |  |  |
| 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | | 0.0 |  |  |  |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | | 0.0 |  |  |  |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | | 0.0 |  |  |  |

[3 rows x 8279 columns] clf.predict(sample\_vect)

array(['ham', 'spam', 'ham'], dtype=object)

|  |
| --- |
| **PIRENS Institute of Business Management and Administration, Loni BK.** |
| **Roll Number: 10097 Sign: Date: / /** |
| **Student Name: Yash Bora** |
| **Subject Name: Knowledge Representation and Artificial Intelligence, ML, DL** |
| **Program Title: Q.12 Plot the cluster data using python visualizations.** |

# Program:

*# Import packages*

import pandas as pd

*# Import the dataset*

df = pd.read\_csv(‘Mall\_Customers.csv’) df.shape

(200, 5)

list(df.columns)

[‘CustomerID’, ‘Genre’, 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']

*# Input data*

x = df.iloc[:,3:]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x |  |  |  |  |  |
| Annual | Income | (k$) | Spending | Score | (1-100) |
| 0 |  | 15 |  |  | 39 |
| 1 |  | 15 |  |  | 81 |
| 2 |  | 16 |  |  | 6 |
| 3 |  | 16 |  |  | 77 |
| 4 |  | 17 |  |  | 40 |
| .. 195 |  | ... 120 |  |  | ... 79 |
| 196 |  | 126 |  |  | 28 |
| 197 |  | 126 |  |  | 74 |
| 198 |  | 137 |  |  | 18 |
| 199 |  | 137 |  |  | 83 |

[200 rows x 2 columns]

*# Summerize*

df.describe()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | CustomerID 200.000000 | Age 200.000000 | Annual | Income (k$) 200.000000 | Spending | Score (1-100)  200.000000 |
| count |  |  |
| mean | 100.500000 | 38.850000 |  | 60.560000 |  | 50.200000 |
| std | 57.879185 | 13.969007 |  | 26.264721 |  | 25.823522 |
| min | 1.000000 | 18.000000 |  | 15.000000 |  | 1.000000 |
| 25% | 50.750000 | 28.750000 |  | 41.500000 |  | 34.750000 |

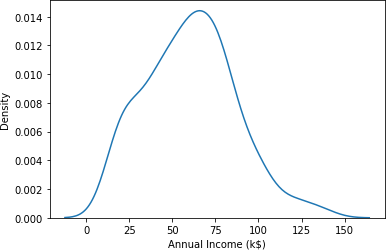
*import seaborn package*

import seaborn as sns sns.kdeplot(df['Age'])

<AxesSubplot:xlabel='Age', ylabel='Density'>

sns.kdeplot(df['Annual Income (k$)'])

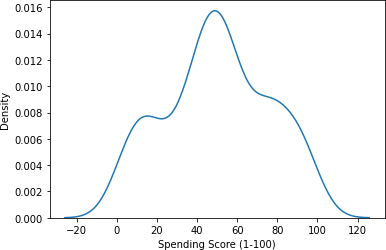
<AxesSubplot:xlabel='Annual Income

(k$)',

ylabel='Density'>

sns.kdeplot(df['Spending Score (1-100)'])

<AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Density'>



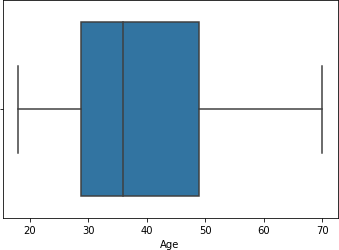
sns.boxplot(df['Age'])

/home/mitu/.local/lib/python3.8/site-packages/seaborn/\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version

0.12, the only valid positional argument will be `data`, and passing otherarguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

<AxesSubplot:xlabel='Age'>

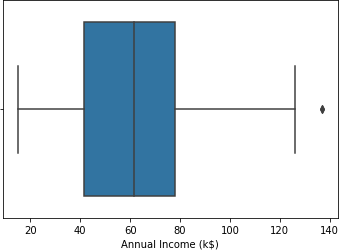


sns.boxplot(df['Annual Income (k$)'])

/home/mitu/.local/lib/python3.8/site-packages/seaborn/\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

<AxesSubplot:xlabel='Annual Income (k

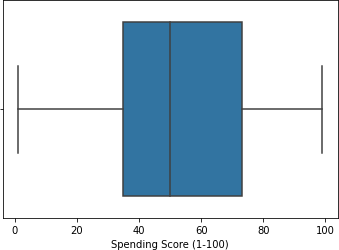
'>

sns.boxplot(df['Spending Score (1-100)'])

/home/mitu/.local/lib/python3.8/site-packages/seaborn/\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

<AxesSubplot:xlabel='Spending Score (1-100)'>



*# Import the class*

from sklearn.cluster import KMeans

*# Create the object*

km = KMeans(n\_clusters=12, random\_state=0)

*# Train the algorithm*

labels = km.fit\_predict(x)

*# Sum of squared errors*

km.inertia\_ 15810.838613705504

*# elbow method*

sse = []

**for** k **in** range(1,41):

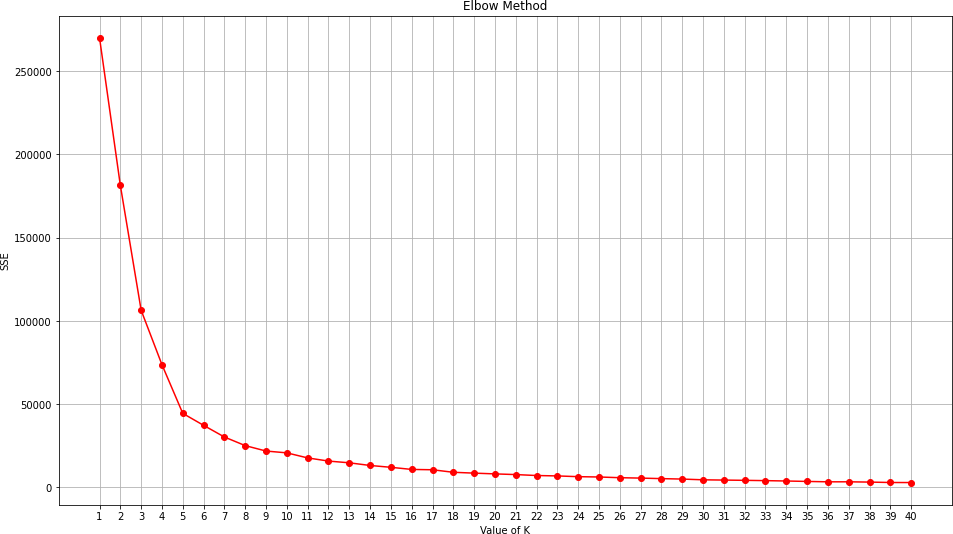
km = KMeans(n\_clusters=k, random\_state=0) labels = km.fit\_predict(x) sse.append(km.inertia\_)

import matplotlib.pyplot as plt plt.figure(figsize=(16,9)) plt.title('Elbow Method') plt.xlabel('Value of K') plt.ylabel('SSE')

plt.grid() plt.xticks(range(1,41))

plt.plot(range(1,41), sse, marker='o', color='r')

[<matplotlib.lines.Line2D at 0x7fb5f259fa60>]



*# Silhoutte method*

from sklearn.metrics import silhouette\_score silh = []

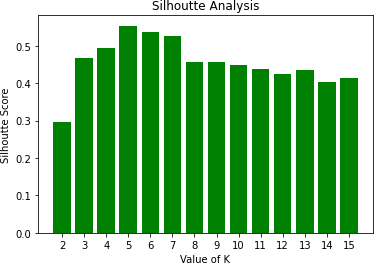
**for** k **in** range(2,16):

km = KMeans(n\_clusters=k, random\_state=0) labels = km.fit\_predict(x)

score = silhouette\_score(x, labels) silh.append(score)

*# plot the silhoutte scores* plt.title('Silhoutte Analysis') plt.xlabel('Value of K') plt.ylabel('Silhoutte Score') plt.xticks(range(2,16)) plt.bar(range(2,16), silh, color='g')

<BarContainer object of 14 artists>



*# Create the object*

km = KMeans(n\_clusters=5, random\_state=0)

*# Train the algorithm*

labels = km.fit\_predict(x)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| labels |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| array([4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, |
| 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 1, |
| 4, | 3, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 2, | 0, | 2, | 1, | 2, | 0, | 2, | 0, | 2, |
| 1, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 1, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, |
| 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, |
| 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, | 0, | 2, |

0, 2], dtype=int32)

*# Cluster labels*

km.labels\_

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| array([4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, |
| 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 3, | 4, | 1, |
| 4, | 3, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |
| 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, | 1, |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1, 1, 1, 1, 1, 1, 1, 1, 1, | 1, | 1, | 1, 1, 2, | 0, | 2, | 1, 2, 0, 2, 0, 2, |
| 1, 2, 0, 2, 0, 2, 0, 2, 0, | 2, | 1, | 2, 0, 2, | 0, | 2, | 0, 2, 0, 2, 0, 2, |
| 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, |  |  | 0, 2, 0, 2, |  |  | 0, 2, 0, 2, 0, 2, 0, 2, |
| 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, |  |  | 0, 2, 0, 2, |  |  | 0, 2, 0, 2, 0, 2, 0, 2, |
| 0, 2], dtype=int32) |  |  |  |  |  |  |

*# SSE*

km.inertia\_ 44448.45544793369

*# Centroids*

km.cluster\_centers\_

array([[88.2 , 17.11428571],

|  |  |
| --- | --- |
| [55.2962963 , | 49.51851852], |
| [86.53846154, | 82.12820513], |
| [25.72727273, | 79.36363636], |
| [26.30434783, | 20.91304348]]) |

*# Extract the clusters*

df[labels==2] *# Boolean filtering*

CustomerID Genre Age Annual Income (k$) Spending Score (1-100)123

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 124 | Male | 39 | 69 | 91 |
| 125 | 126 | Female | 31 | 70 | 77 |
| 127 | 128 | Male | 40 | 71 | 95 |
| 129 | 130 | Male | 38 | 71 | 75 |
| 131 | 132 | Male | 39 | 71 | 75 |
| 133 | 134 | Female | 31 | 72 | 71 |
| 135 | 136 | Female | 29 | 73 | 88 |
| 137 | 138 | Male | 32 | 73 | 73 |
| 139 | 140 | Female | 35 | 74 | 72 |
| 141 | 142 | Male | 32 | 75 | 93 |
| 143 | 144 | Female | 32 | 76 | 87 |
| 145 | 146 | Male | 28 | 77 | 97 |
| 147 | 148 | Female | 32 | 77 | 74 |
| 149 | 150 | Male | 34 | 78 | 90 |
| 151 | 152 | Male | 39 | 78 | 88 |
| 153 | 154 | Female | 38 | 78 | 76 |
| 155 | 156 | Female | 27 | 78 | 89 |
| 157 | 158 | Female | 30 | 78 | 78 |
| 159 | 160 | Female | 30 | 78 | 73 |
| 161 | 162 | Female | 29 | 79 | 83 |
| 163 | 164 | Female | 31 | 81 | 93 |
| 165 | 166 | Female | 36 | 85 | 75 |
| 167 | 168 | Female | 33 | 86 | 95 |
| 169 | 170 | Male | 32 | 87 | 63 |
| 171 | 172 | Male | 28 | 87 | 75 |
| 173 | 174 | Male | 36 | 87 | 92 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 175 | 176 Female | | 30 | 88 | 86 |
| 177 | 178 Male | | 27 | 88 | 69 |
| 179 | 180 Male | | 35 | 93 | 90 |
| 181 | 182 | Female | 32 | 97 | 86 |
| 183 | 184 | Female | 29 | 98 | 88 |
| 185 | 186 | Male | 30 | 99 | 97 |
| 187 | 188 | Male | 28 | 101 | 68 |
| 189 | 190 | Female | 36 | 103 | 85 |
| 191 | 192 | Female | 32 | 103 | 69 |
| 193 | 194 | Female | 38 | 113 | 91 |
| 195 | 196 | Female | 35 | 120 | 79 |
| 197 | 198 | Male | 32 | 126 | 74 |
| 199 | 200 | Male | 30 | 137 | 83 |

one = df[labels==1] one.shape

(81, 5)

*# Export the cluster*

one.to\_csv('one.csv')

print('Cluster-0:', len(df[labels==0]))

print('Cluster-1:', len(df[labels==1]))

print('Cluster-2:', len(df[labels==2]))

print('Cluster-3:', len(df[labels==3]))

print('Cluster-4:', len(df[labels==4]))

Cluster-0: 35

Cluster-1: 81

Cluster-2: 39

Cluster-3: 22

Cluster-4: 23

*# Prediction*

new = [[45, 76]]

km.predict(new)[0]3

*# Prediction*

new = [[25, 36]]

km.predict(new)[0]4

*# Prediction*

new = [[85, 76]]

km.predict(new)[0]2

*# Prediction*

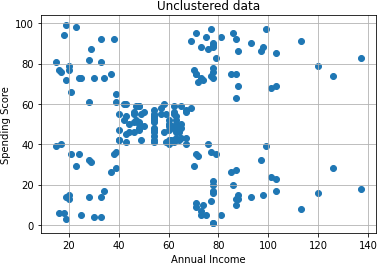
new = [[45, 47]]

km.predict(new)[0]1

*# Visualization of clusters* plt.title('Unclustered data')plt.xlabel('Annual Income') plt.ylabel('Spending Score') plt.grid()

plt.scatter(x['Annual Income (k$)'], x['Spending Score (1-100)'])

<matplotlib.collections.PathCollection at 0x7fb5f14582e0>



*# Save the centroids*

cent = km.cluster\_centers\_cent array([[88.2 , 17.11428571],

|  |  |
| --- | --- |
| [55.2962963 , | 49.51851852], |
| [86.53846154, | 82.12820513], |
| [25.72727273, | 79.36363636], |
| [26.30434783, | 20.91304348]]) |

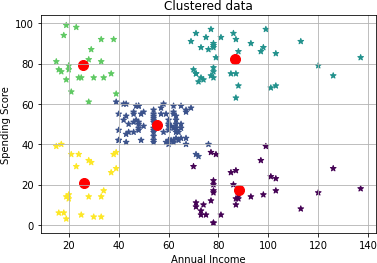
*# Visualization of clusters* plt.title('Clustered data') plt.xlabel('Annual Income') plt.ylabel('Spending Score') plt.grid()

plt.scatter(x['Annual Income (k$)'], x['Spending Score (1-100)'],

c = labels, marker='\*')

plt.scatter(cent[:,0], cent[:,1], s=100, marker='o', color='r')

<matplotlib.collections.PathCollection at 0x7fb5f0ed0b80>



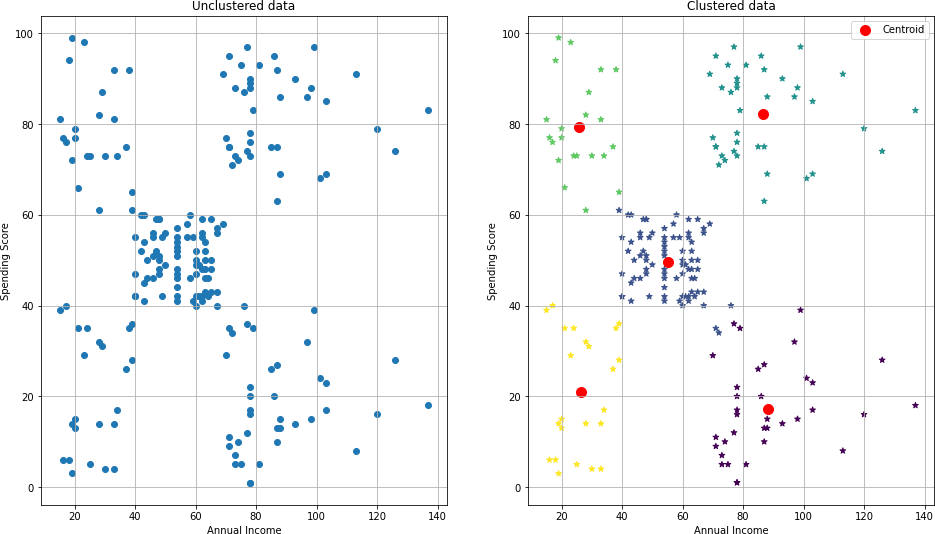
*# Combined plot* plt.figure(figsize=(16,9)) plt.subplot(1,2,1) plt.title('Unclustered data') plt.xlabel('Annual Income') plt.ylabel('Spending Score') plt.grid()

plt.scatter(x['Annual Income (k$)'], x['Spending Score (1-100)'])plt.subplot(1,2,2) plt.title('Clustered data')

plt.xlabel('Annual Income') plt.ylabel('Spending Score') plt.grid()

plt.scatter(x['Annual Income (k$)'], x['Spending Score (1-100)'],c = labels, marker='\*')

plt.scatter(cent[:,0], cent[:,1], s=100, marker='o', color='r',label = 'Centroid') plt.legend() plt.savefig('Clusters.png')



import seaborn as sns

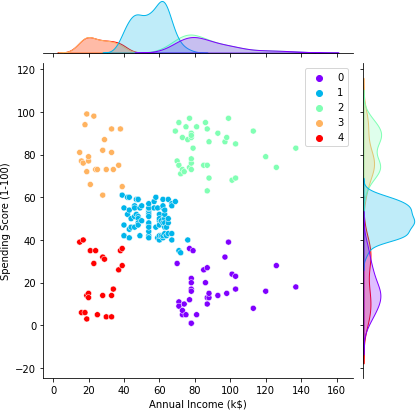
*# Visualization using joint plot*

p = sns.jointplot(x=x['Annual Income (k$)'],

y=x['Spending Score (1-100)'], hue = labels,palette='rainbow', )

*# sns.jointplot(x=cent[:,0], y=cent[:,1])*

p.savefig('seaborn\_clusters.png')



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 131 | [D | loss: | 0.038602, | acc.: | 100.00%] [G loss: 4.238760] |
| 132 | [D | loss: | 0.066584, | acc.: | 98.44%] [G loss: 4.631004] |
| 133 | [D | loss: | 0.064235, | acc.: | 98.44%] [G loss: 4.729104] |
| 134 | [D | loss: | 0.057679, | acc.: | 100.00%] [G loss: 4.399063] |
| 135 | [D | loss: | 0.038678, | acc.: | 100.00%] [G loss: 3.980439] |
| 136 | [D | loss: | 0.070430, | acc.: | 96.88%] [G loss: 4.184968] |
| 137 | [D | loss: | 0.269052, | acc.: | 85.94%] [G loss: 3.930685] |
| 138 | [D | loss: | 0.071771, | acc.: | 98.44%] [G loss: 4.210067] |
| 139 | [D | loss: | 0.175595, | acc.: | 92.19%] [G loss: 3.578120] |
| 140 | [D | loss: | 0.057856, | acc.: | 96.88%] [G loss: 4.090517] |
| 141 | [D | loss: | 0.091329, | acc.: | 98.44%] [G loss: 3.495711] |
| 142 | [D | loss: | 0.074046, | acc.: | 98.44%] [G loss: 3.672240] |
| 143 | [D | loss: | 0.067564, | acc.: | 100.00%] [G loss: 3.488506] |
| 144 | [D | loss: | 0.097541, | acc.: | 96.88%] [G loss: 3.927138] |
| 145 | [D | loss: | 0.189200, | acc.: | 92.19%] [G loss: 3.607136] |
| 146 | [D | loss: | 0.069164, | acc.: | 100.00%] [G loss: 4.224221] |
| 147 | [D | loss: | 0.577445, | acc.: | 79.69%] [G loss: 2.658618] |
| 148 | [D | loss: | 0.192502, | acc.: | 92.19%] [G loss: 3.820522] |
| 149 | [D | loss: | 0.084979, | acc.: | 98.44%] [G loss: 4.757998] |
| 150 | [D | loss: | 0.261661, | acc.: | 92.19%] [G loss: 2.996725] |
| 151 | [D | loss: | 0.188527, | acc.: | 89.06%] [G loss: 4.621965] |
| 152 | [D | loss: | 0.151155, | acc.: | 93.75%] [G loss: 3.851809] |
| 153 | [D | loss: | 0.136393, | acc.: | 93.75%] [G loss: 4.189128] |
| 154 | [D | loss: | 0.083352, | acc.: | 100.00%] [G loss: 4.461646] |
| 155 | [D | loss: | 0.206723, | acc.: | 89.06%] [G loss: 4.497554] |
| 156 | [D | loss: | 0.241861, | acc.: | 89.06%] [G loss: 4.464531] |
| 157 | [D | loss: | 0.319591, | acc.: | 82.81%] [G loss: 3.933166] |
| 158 | [D | loss: | 0.078051, | acc.: | 100.00%] [G loss: 3.995445] |
| 159 | [D | loss: | 0.258115, | acc.: | 89.06%] [G loss: 3.682753] |
| 160 | [D | loss: | 0.068538, | acc.: | 98.44%] [G loss: 3.920011] |
| 161 | [D | loss: | 0.137065, | acc.: | 95.31%] [G loss: 2.958877] |
| 162 | [D | loss: | 0.092553, | acc.: | 95.31%] [G loss: 3.897508] |
| 163 | [D | loss: | 0.243603, | acc.: | 89.06%] [G loss: 3.506659] |
| 164 | [D | loss: | 0.044570, | acc.: | 100.00%] [G loss: 4.298730] |
| 165 | [D | loss: | 0.274047, | acc.: | 89.06%] [G loss: 3.803701] |
| 166 | [D | loss: | 0.216394, | acc.: | 90.62%] [G loss: 4.244328] |
| 167 | [D | loss: | 0.938720, | acc.: | 57.81%] [G loss: 1.454402] |
| 168 | [D | loss: | 0.281417, | acc.: | 85.94%] [G loss: 3.043864] |
| 169 | [D | loss: | 0.071866, | acc.: | 100.00%] [G loss: 4.173522] |
| 170 | [D | loss: | 0.167514, | acc.: | 95.31%] [G loss: 3.013133] |
| 171 | [D | loss: | 0.095101, | acc.: | 96.88%] [G loss: 3.071562] |
| 172 | [D | loss: | 0.062486, | acc.: | 98.44%] [G loss: 3.801221] |
| 173 | [D | loss: | 0.169537, | acc.: | 96.88%] [G loss: 3.312897] |
| 174 | [D | loss: | 0.098783, | acc.: | 96.88%] [G loss: 4.142616] |
| 175 | [D | loss: | 0.244112, | acc.: | 92.19%] [G loss: 3.173460] |
| 176 | [D | loss: | 0.129209, | acc.: | 96.88%] [G loss: 5.158587] |
| 177 | [D | loss: | 0.785221, | acc.: | 67.19%] [G loss: 2.247335] |
| 178 | [D | loss: | 0.319861, | acc.: | 81.25%] [G loss: 3.888173] |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 179 | [D | loss: | 0.074654, | acc.: | 96.88%] [G loss: 5.345549] |
| 180 | [D | loss: | 0.378398, | acc.: | 84.38%] [G loss: 2.330404] |

181 [D loss: 0.144777, acc.: 90.62%] [G loss: 3.041365]

182 [D loss: 0.095836, acc.: 95.31%] [G loss: 4.223273]

183 [D loss: 0.157615, acc.: 96.88%] [G loss: 3.565648]

184 [D loss: 0.109397, acc.: 98.44%] [G loss: 4.065246]

185 [D loss: 0.226231, acc.: 92.19%] [G loss: 3.359378]

186 [D loss: 0.151613, acc.: 95.31%] [G loss: 4.360668]

187 [D loss: 0.582917, acc.: 70.31%] [G loss: 2.666638]

188 [D loss: 0.080962, acc.: 100.00%] [G loss: 4.300864]

189 [D loss: 0.176439, acc.: 95.31%] [G loss: 3.181917]

190 [D loss: 0.107121, acc.: 98.44%] [G loss: 3.637481]

191 [D loss: 0.209021, acc.: 92.19%] [G loss: 4.648886]

192 [D loss: 0.334682, acc.: 85.94%] [G loss: 2.255054]

193 [D loss: 0.154234, acc.: 95.31%] [G loss: 4.317871]

194 [D loss: 0.288475, acc.: 90.62%] [G loss: 2.890252]

195 [D loss: 0.113874, acc.: 98.44%] [G loss: 3.731670]

196 [D loss: 0.272280, acc.: 90.62%] [G loss: 3.698488]

197 [D loss: 0.375167, acc.: 81.25%] [G loss: 5.970434]

198 [D loss: 1.642656, acc.: 42.19%] [G loss: 1.831249]

199 [D loss: 0.910615, acc.: 62.50%] [G loss: 1.924973]