weatheripynb

February 27, 2024

[1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score
from sklearn.preprocessing import PowerTransformer
from sklearn.model_selection import cross_val_score
from scipy.cluster.vq import whiten, kmeans, vq
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv('weather.csv')
```

[3]:
```
df
```

[3]:
```
     MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine WindGustDir  \
0        8.0     24.3       0.0          3.4       6.3          NW
1       14.0     26.9       3.6          4.4       9.7         ENE
2       13.7     23.4       3.6          5.8       3.3          NW
3       13.3     15.5      39.8          7.2       9.1          NW
4        7.6     16.1       2.8          5.6      10.6         SSE
..       …        …        …           …        …           …
361      9.0     30.7       0.0          7.6      12.1         NNW
362      7.1     28.4       0.0         11.6      12.7           N
363     12.5     19.9       0.0          8.4       5.3         ESE
364     12.5     26.9       0.0          5.0       7.1          NW
365     12.3     30.2       0.0          6.0      12.6          NW

     WindGustSpeed WindDir9am WindDir3pm  WindSpeed9am  …  Humidity3pm  \
```

```
0             30.0          SW          NW          6.0  ...        29
1             39.0           E           W          4.0  ...        36
2             85.0           N         NNE          6.0  ...        69
3             54.0         WNW           W         30.0  ...        56
4             50.0         SSE         ESE         20.0  ...        49
..             ...         ...         ...          ...  ...       ...
361           76.0         SSE          NW          7.0  ...        15
362           48.0         NNW         NNW          2.0  ...        22
363           43.0         ENE         ENE         11.0  ...        47
364           46.0         SSW         WNW          6.0  ...        39
365           78.0          NW         WNW         31.0  ...        13

     Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  \
0         1019.7       1015.0         7         7     14.4     23.6
1         1012.4       1008.4         5         3     17.5     25.7
2         1009.5       1007.2         8         7     15.4     20.2
3         1005.5       1007.0         2         7     13.5     14.1
4         1018.3       1018.5         7         7     11.1     15.4
..           ...          ...       ...       ...      ...      ...
361       1016.1       1010.8         1         3     20.4     30.0
362       1020.0       1016.9         0         1     17.2     28.2
363       1024.0       1022.8         3         2     14.5     18.3
364       1021.0       1016.2         6         7     15.8     25.9
365       1009.6       1009.2         1         1     23.8     28.6

     RainToday  RISK_MM RainTomorrow
0           No      3.6          Yes
1          Yes      3.6          Yes
2          Yes     39.8          Yes
3          Yes      2.8          Yes
4          Yes      0.0           No
..         ...      ...          ...
361         No      0.0           No
362         No      0.0           No
363         No      0.0           No
364         No      0.0           No
365         No      0.0           No

[366 rows x 22 columns]
```

[4]:
```
obj_df = df.select_dtypes(include=['object'])
num_df = df.select_dtypes([np.number])
```

[5]:
```
obj_df.fillna(obj_df.mode().iloc[0], inplace=True)
```

[6]:
```
obj_df_label = obj_df.copy()
```

[7]:
```
le = LabelEncoder()
obj_df_label = obj_df.apply(le.fit_transform)
```

[8]:
```
obj_df_label
```

[8]:
```
     WindGustDir  WindDir9am  WindDir3pm  RainToday  RainTomorrow
0              7          12           7          0             1
1              1           0          13          1             1
2              7           3           5          1             1
3              7          14          13          1             1
4             10          10           2          1             0
..           …           …           …         …           …
361            6          10           7          0             0
362            3           6           6          0             0
363            2           1           1          0             0
364            7          11          14          0             0
365            7           7          14          0             0

[366 rows x 5 columns]
```

[9]:
```
obj_df_onehot = obj_df.copy()
```

[10]:
```
obj_df_onehot = pd.get_dummies(obj_df_onehot, columns=obj_df_onehot.columns)
```

[11]:
```
obj_df_onehot
```

[11]:
```
     WindGustDir_E  WindGustDir_ENE  WindGustDir_ESE  WindGustDir_N  \
0            False            False            False          False
1            False             True            False          False
2            False            False            False          False
3            False            False            False          False
4            False            False            False          False
..             …              …                …              …
361          False            False            False          False
362          False            False            False           True
363          False            False             True          False
364          False            False            False          False
365          False            False            False          False

     WindGustDir_NE  WindGustDir_NNE  WindGustDir_NNW  WindGustDir_NW  \
0             False            False            False            True
1             False            False            False           False
2             False            False            False            True
3             False            False            False            True
4             False            False            False           False
..              …              …                …              …
361           False            False             True           False
```

```
362          False            False            False           False
363          False            False            False           False
364          False            False            False            True
365          False            False            False            True

     WindGustDir_S  WindGustDir_SE  …  WindDir3pm_SSE  WindDir3pm_SSW  \
0            False           False  …           False           False
1            False           False  …           False           False
2            False           False  …           False           False
3            False           False  …           False           False
4            False           False  …           False           False
..             …             …  …             …             …
361          False           False  …           False           False
362          False           False  …           False           False
363          False           False  …           False           False
364          False           False  …           False           False
365          False           False  …           False           False

     WindDir3pm_SW  WindDir3pm_W  WindDir3pm_WNW  WindDir3pm_WSW  \
0            False         False           False           False
1            False          True           False           False
2            False         False           False           False
3            False          True           False           False
4            False         False           False           False
..             …           …             …             …
361          False         False           False           False
362          False         False           False           False
363          False         False           False           False
364          False         False            True           False
365          False         False            True           False

     RainToday_No  RainToday_Yes  RainTomorrow_No  RainTomorrow_Yes
0            True          False            False              True
1           False           True            False              True
2           False           True            False              True
3           False           True            False              True
4           False           True             True             False
..            …            …              …               …
361          True          False             True             False
362          True          False             True             False
363          True          False             True             False
364          True          False             True             False
365          True          False             True             False

[366 rows x 52 columns]
```

[12]:
```
obj_df_binary = obj_df.copy()
```

[13]:
```
obj_df.isnull().sum()
```

[13]:
```
WindGustDir     0
WindDir9am      0
WindDir3pm      0
RainToday       0
RainTomorrow    0
dtype: int64
```

[14]:
```
obj_df_binary.fillna(obj_df_binary.mode().iloc[0], inplace=True)
```

[15]:
```
obj_df_binary
```

[15]:
```
    WindGustDir WindDir9am WindDir3pm RainToday RainTomorrow
0            NW         SW         NW        No          Yes
1           ENE          E          W       Yes          Yes
2            NW          N        NNE       Yes          Yes
3            NW        WNW          W       Yes          Yes
4           SSE        SSE        ESE       Yes           No
..          …          …          …         …           …
361         NNW        SSE         NW        No           No
362           N        NNW        NNW        No           No
363         ESE        ENE        ENE        No           No
364          NW        SSW        WNW        No           No
365          NW         NW        WNW        No           No

[366 rows x 5 columns]
```

[16]:
```
# lb = LabelBinarizer()
# obj_df_binary = lb.fit_transform(obj_df_binary)
```

[17]:
```
num_df.isnull().sum()
```

[17]:
```
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         3
WindGustSpeed    2
WindSpeed9am     7
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
```

```
Temp9am          0
Temp3pm          0
RISK_MM          0
dtype: int64
```

[18]:
```
num_df
```

[18]:
```
     MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0        8.0     24.3       0.0          3.4       6.3           30.0
1       14.0     26.9       3.6          4.4       9.7           39.0
2       13.7     23.4       3.6          5.8       3.3           85.0
3       13.3     15.5      39.8          7.2       9.1           54.0
4        7.6     16.1       2.8          5.6      10.6           50.0
..       …        …         …            …         …              …
361      9.0     30.7       0.0          7.6      12.1           76.0
362      7.1     28.4       0.0         11.6      12.7           48.0
363     12.5     19.9       0.0          8.4       5.3           43.0
364     12.5     26.9       0.0          5.0       7.1           46.0
365     12.3     30.2       0.0          6.0      12.6           78.0

     WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  \
0             6.0            20           68           29       1019.7
1             4.0            17           80           36       1012.4
2             6.0             6           82           69       1009.5
3            30.0            24           62           56       1005.5
4            20.0            28           68           49       1018.3
..            …             …            …            …            …
361           7.0            50           38           15       1016.1
362           2.0            19           45           22       1020.0
363          11.0             9           63           47       1024.0
364           6.0            28           69           39       1021.0
365          31.0            35           43           13       1009.6

     Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  RISK_MM
0         1015.0         7         7     14.4     23.6      3.6
1         1008.4         5         3     17.5     25.7      3.6
2         1007.2         8         7     15.4     20.2     39.8
3         1007.0         2         7     13.5     14.1      2.8
4         1018.5         7         7     11.1     15.4      0.0
..           …         …         …        …        …        …
361       1010.8         1         3     20.4     30.0      0.0
362       1016.9         0         1     17.2     28.2      0.0
363       1022.8         3         2     14.5     18.3      0.0
364       1016.2         6         7     15.8     25.9      0.0
365       1009.2         1         1     23.8     28.6      0.0

[366 rows x 17 columns]
```

[19]:
```
num_df_mean = num_df.copy()
```

[20]:
```
num_df_mean.fillna(num_df_mean.mean().iloc[0], inplace=True)
```

[21]:
```
num_df_mean.isnull().sum()
```

[21]:
```
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustSpeed    0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RISK_MM          0
dtype: int64
```

[22]:
```
obj_df_label
```

[22]:

	WindGustDir	WindDir9am	WindDir3pm	RainToday	RainTomorrow
0	7	12	7	0	1
1	1	0	13	1	1
2	7	3	5	1	1
3	7	14	13	1	1
4	10	10	2	1	0
..	…	…	…	…	…
361	6	10	7	0	0
362	3	6	6	0	0
363	2	1	1	0	0
364	7	11	14	0	0
365	7	7	14	0	0

[366 rows x 5 columns]

[23]:
```
# Q1 = num_df_mean.quantile(0.25)
# Q3 = num_df_mean.quantile(0.75)
# IQR = Q3 - Q1
# lower_fence = Q1 - 1.5 * IQR
# upper_fence = Q3 + 1.5 * IQR
```

```
# num_df_mean = num_df_mean[~((num_df_mean < lower_fence) | (num_df_mean >␣
 ↪upper_fence)).any(axis=1)]
```

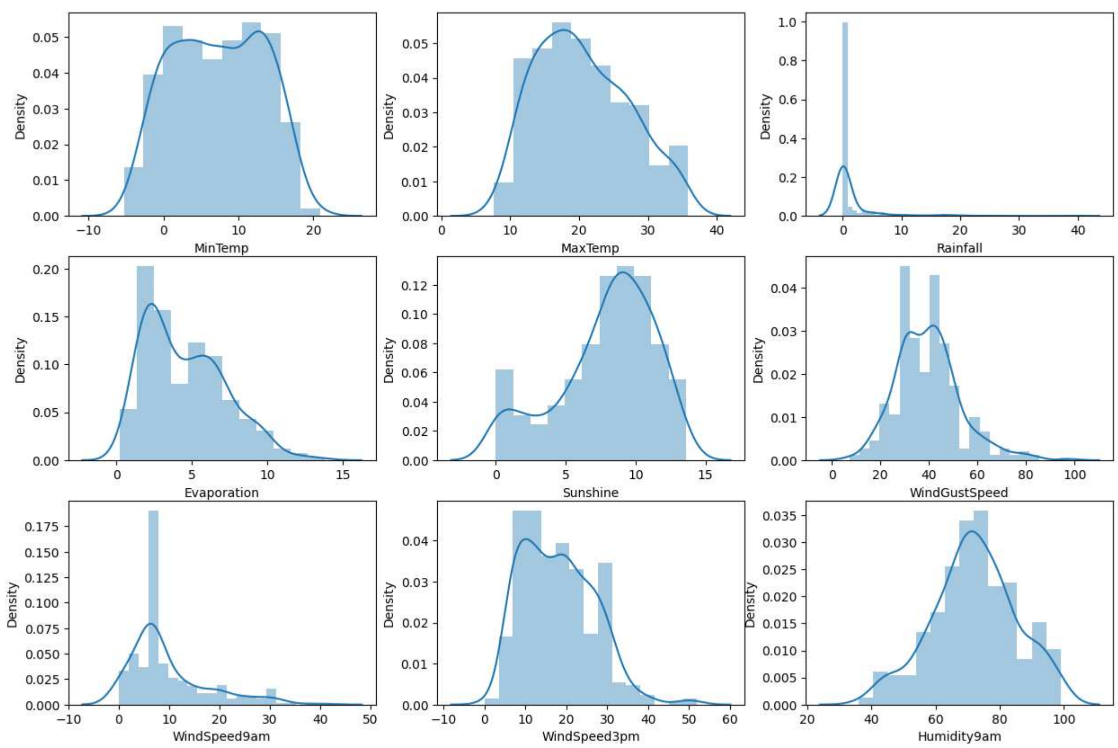
[24]:
```
processed_df = pd.concat([num_df_mean, obj_df_label], axis=1)
```

[25]:
```
processed_df
```

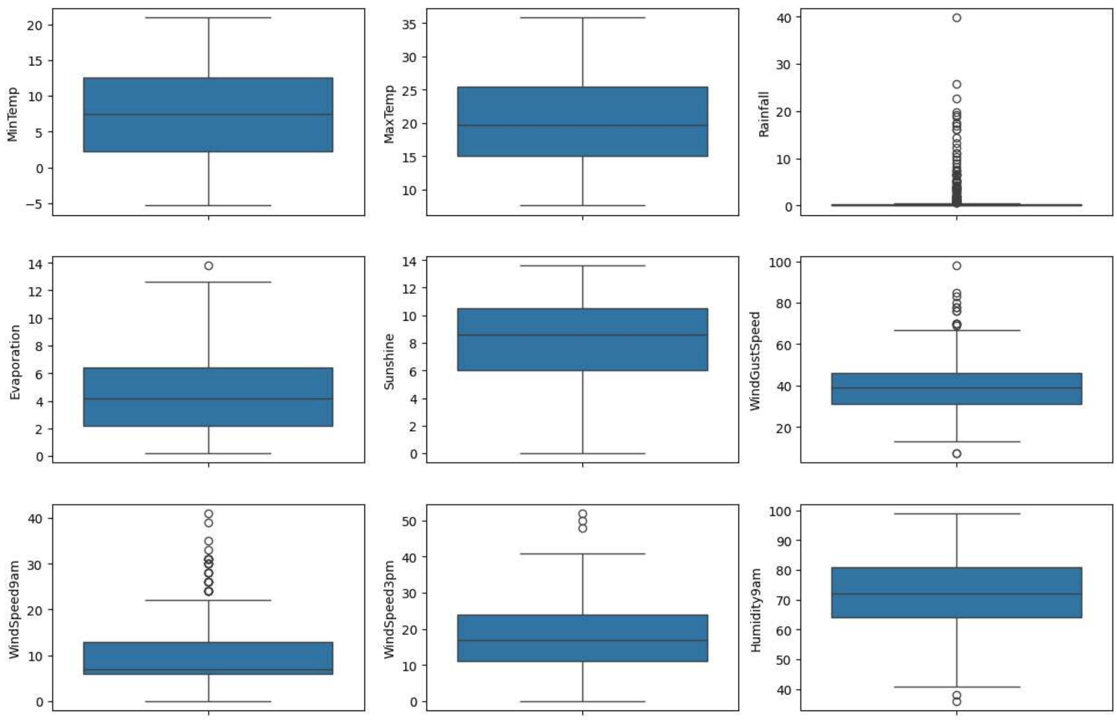
[25]:
```
     MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0        8.0     24.3       0.0          3.4       6.3           30.0
1       14.0     26.9       3.6          4.4       9.7           39.0
2       13.7     23.4       3.6          5.8       3.3           85.0
3       13.3     15.5      39.8          7.2       9.1           54.0
4        7.6     16.1       2.8          5.6      10.6           50.0
..       …        …        …          …        …            …
361      9.0     30.7       0.0          7.6      12.1           76.0
362      7.1     28.4       0.0         11.6      12.7           48.0
363     12.5     19.9       0.0          8.4       5.3           43.0
364     12.5     26.9       0.0          5.0       7.1           46.0
365     12.3     30.2       0.0          6.0      12.6           78.0

     WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  …  Cloud9am  \
0             6.0            20           68           29  …         7
1             4.0            17           80           36  …         5
2             6.0             6           82           69  …         8
3            30.0            24           62           56  …         2
4            20.0            28           68           49  …         7
..            …             …            …            …  …       …
361           7.0            50           38           15  …         1
362           2.0            19           45           22  …         0
363          11.0             9           63           47  …         3
364           6.0            28           69           39  …         6
365          31.0            35           43           13  …         1

     Cloud3pm  Temp9am  Temp3pm  RISK_MM  WindGustDir  WindDir9am  WindDir3pm  \
0           7     14.4     23.6      3.6            7          12           7
1           3     17.5     25.7      3.6            1           0          13
2           7     15.4     20.2     39.8            7           3           5
3           7     13.5     14.1      2.8            7          14          13
4           7     11.1     15.4      0.0           10          10           2
..        …       …       …       …          …          …          …
361         3     20.4     30.0      0.0            6          10           7
362         1     17.2     28.2      0.0            3           6           6
363         2     14.5     18.3      0.0            2           1           1
364         7     15.8     25.9      0.0            7          11          14
365         1     23.8     28.6      0.0            7           7          14

     RainToday  RainTomorrow
```

```
0      0      1
1      1      1
2      1      1
3      1      1
4      1      0
..     …      …
361    0      0
362    0      0
363    0      0
364    0      0
365    0      0

[366 rows x 22 columns]
```

[81]:
```python
fig, ax = plt.subplots(3, 3, figsize=(15, 10))
for i, subplot in zip(processed_df.columns, ax.flatten()):
    sns.distplot(processed_df[i], ax=subplot)
plt.show()
```



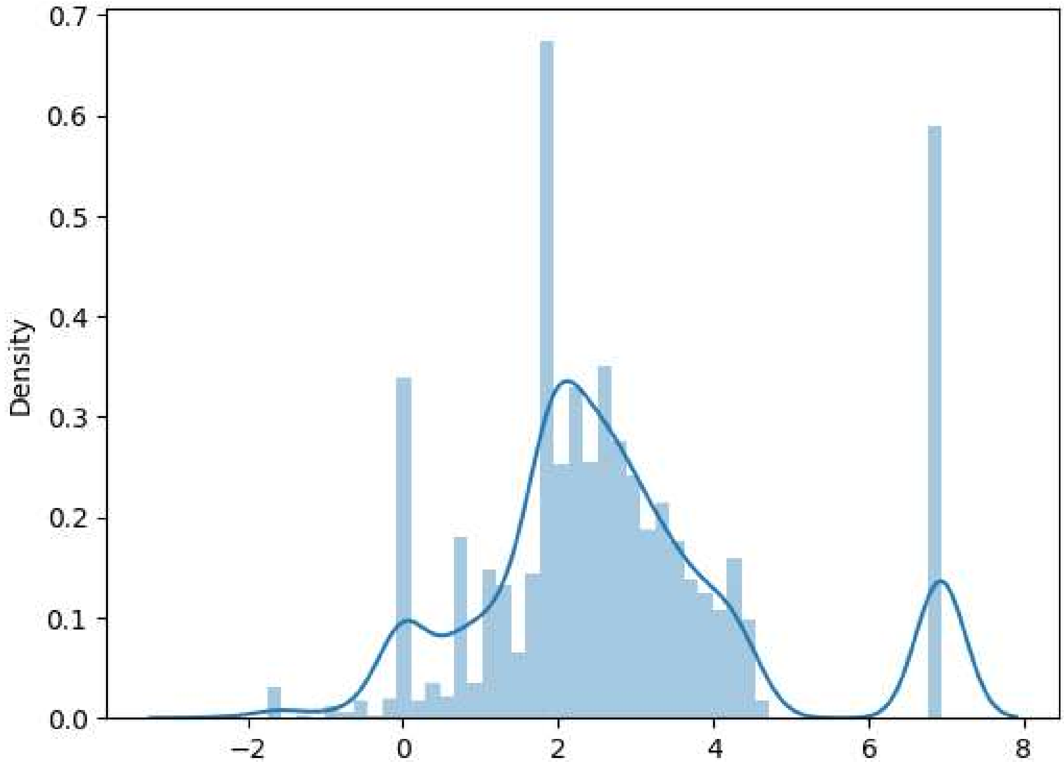
[80]:
```python
fig, ax = plt.subplots(3, 3, figsize=(15, 10))
for i, subplot in zip(processed_df.columns, ax.flatten()):
    sns.boxplot(processed_df[i], ax=subplot)
plt.show()
```



[28]:
```
processed_df_log = processed_df.copy()
```

[29]:
```
processed_df_log = np.log(processed_df_log)
processed_df_log = processed_df_log.replace([np.inf, -np.inf], np.nan)
np.seterr(divide = 'ignore')
warnings.filterwarnings('ignore')
```

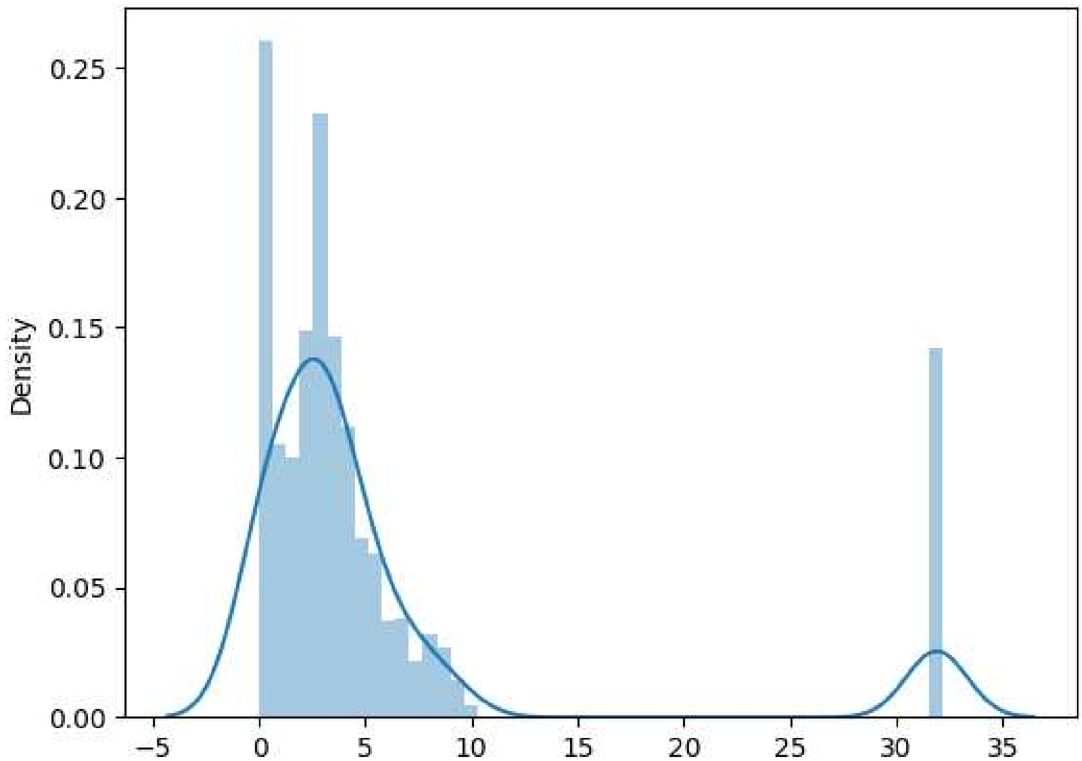
[30]:
```
sns.distplot(processed_df_log)
```



[31]:
```
processed_df_sqrt = processed_df.copy()
```

[32]:
```
processed_df_sqrt = np.sqrt(processed_df_sqrt)
```

[33]:
```
sns.distplot(processed_df_sqrt)
warnings.filterwarnings('ignore')
```

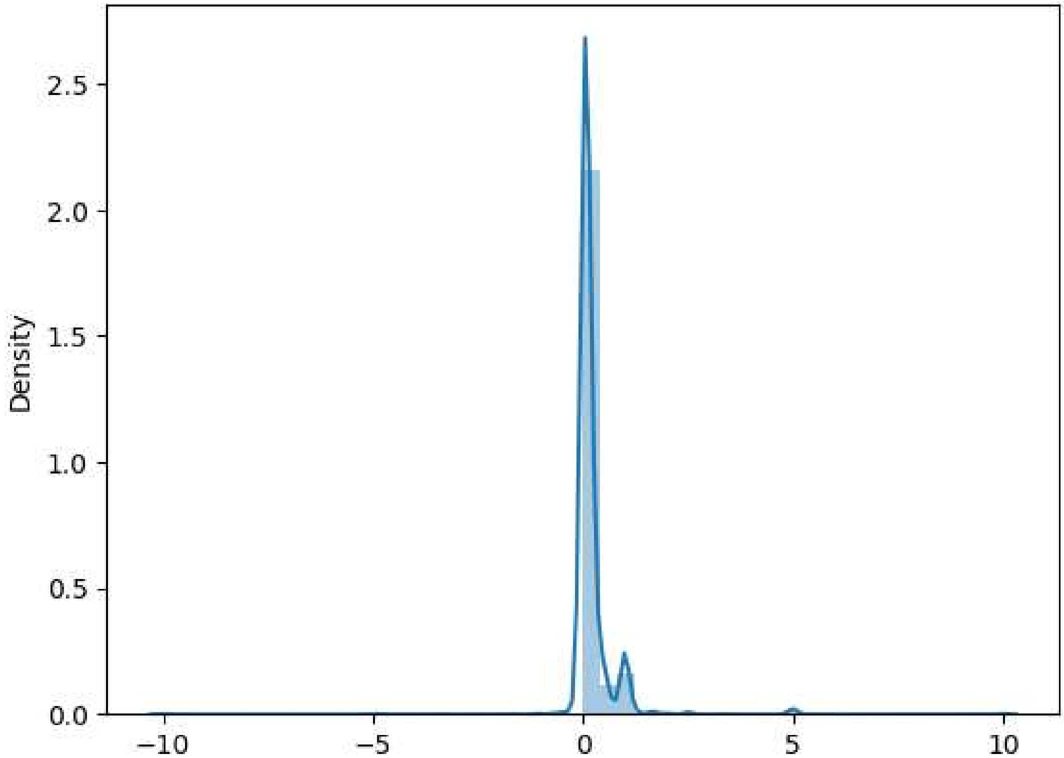


[34]:
```
processed_df_reciprocal = processed_df.copy()
```

[35]:
```
processed_df_reciprocal = 1/processed_df_reciprocal
processed_df_reciprocal = processed_df_reciprocal.replace([np.inf, -np.inf], np.
 ↪nan)
```

[36]:
```
sns.distplot(processed_df_reciprocal)
```

[36]: <Axes: ylabel='Density'>



[86]:
```
processed_df.head()
```

[86]:
```
   MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0      8.0     24.3       0.0          3.4       6.3           30.0
1     14.0     26.9       3.6          4.4       9.7           39.0
2     13.7     23.4       3.6          5.8       3.3           85.0
3     13.3     15.5      39.8          7.2       9.1           54.0
4      7.6     16.1       2.8          5.6      10.6           50.0

   WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  …  Cloud9am  \
0           6.0            20           68           29  …         7
1           4.0            17           80           36  …         5
2           6.0             6           82           69  …         8
3          30.0            24           62           56  …         2
4          20.0            28           68           49  …         7

   Cloud3pm  Temp9am  Temp3pm  RISK_MM  WindGustDir  WindDir9am  WindDir3pm  \
0         7     14.4     23.6      3.6            7          12           7
1         3     17.5     25.7      3.6            1           0          13
2         7     15.4     20.2     39.8            7           3           5
3         7     13.5     14.1      2.8            7          14          13
4         7     11.1     15.4      0.0           10          10           2
```

```
   RainToday  RainTomorrow
0          0             1
1          1             1
2          1             1
3          1             1
4          1             0

[5 rows x 22 columns]
```

[38]:
```python
lr = LinearRegression()
```

[39]:
```python
X = processed_df.iloc[:,:-5]
y = processed_df.iloc[:, -5:]
```

[40]:
```python
y
```

[40]:
```
     WindGustDir  WindDir9am  WindDir3pm  RainToday  RainTomorrow
0              7          12           7          0             1
1              1           0          13          1             1
2              7           3           5          1             1
3              7          14          13          1             1
4             10          10           2          1             0
..           …           …           …          …           …
361            6          10           7          0             0
362            3           6           6          0             0
363            2           1           1          0             0
364            7          11          14          0             0
365            7           7          14          0             0

[366 rows x 5 columns]
```

[41]:
```python
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

[42]:
```python
lr.fit(X_train,y_train)

y_pred = lr.predict(X_test)

r2_score(y_test,y_pred)
```

[42]: 0.27407628542111473

[43]:
```python
pt1 = PowerTransformer()

X_train_transformed2 = pt1.fit_transform(X_train)
X_test_transformed2 = pt1.transform(X_test)
```

```
lr = LinearRegression()
lr.fit(X_train_transformed2,y_train)

y_pred3 = lr.predict(X_test_transformed2)

print(r2_score(y_test,y_pred3))

pd.DataFrame({'cols':X_train.columns,'Yeo_Johnson_lambdas':pt1.lambdas_})
```

0.36108092627846095

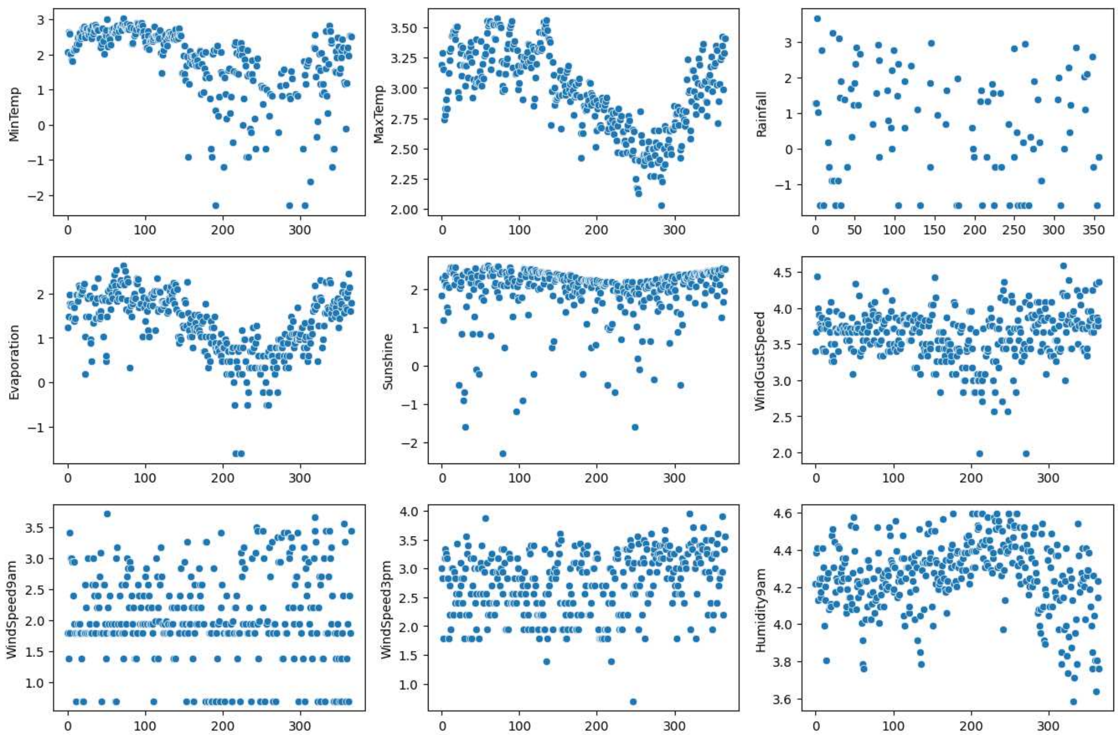
[43]:

	cols	Yeo_Johnson_lambdas
0	MinTemp	0.845582
1	MaxTemp	0.258255
2	Rainfall	-2.494539
3	Evaporation	0.146360
4	Sunshine	1.432938
5	WindGustSpeed	0.519691
6	WindSpeed9am	0.255951
7	WindSpeed3pm	0.452736
8	Humidity9am	1.266156
9	Humidity3pm	0.372768
10	Pressure9am	25.954197
11	Pressure3pm	22.063945
12	Cloud9am	0.299128
13	Cloud3pm	0.394528
14	Temp9am	0.811665
15	Temp3pm	0.426617
16	RISK_MM	-2.305938

[44]:
```
pt = PowerTransformer()
X_transformed2 = pt.fit_transform(X)

lr = LinearRegression()
np.mean(cross_val_score(lr,X_transformed2,y,scoring='r2'))
```

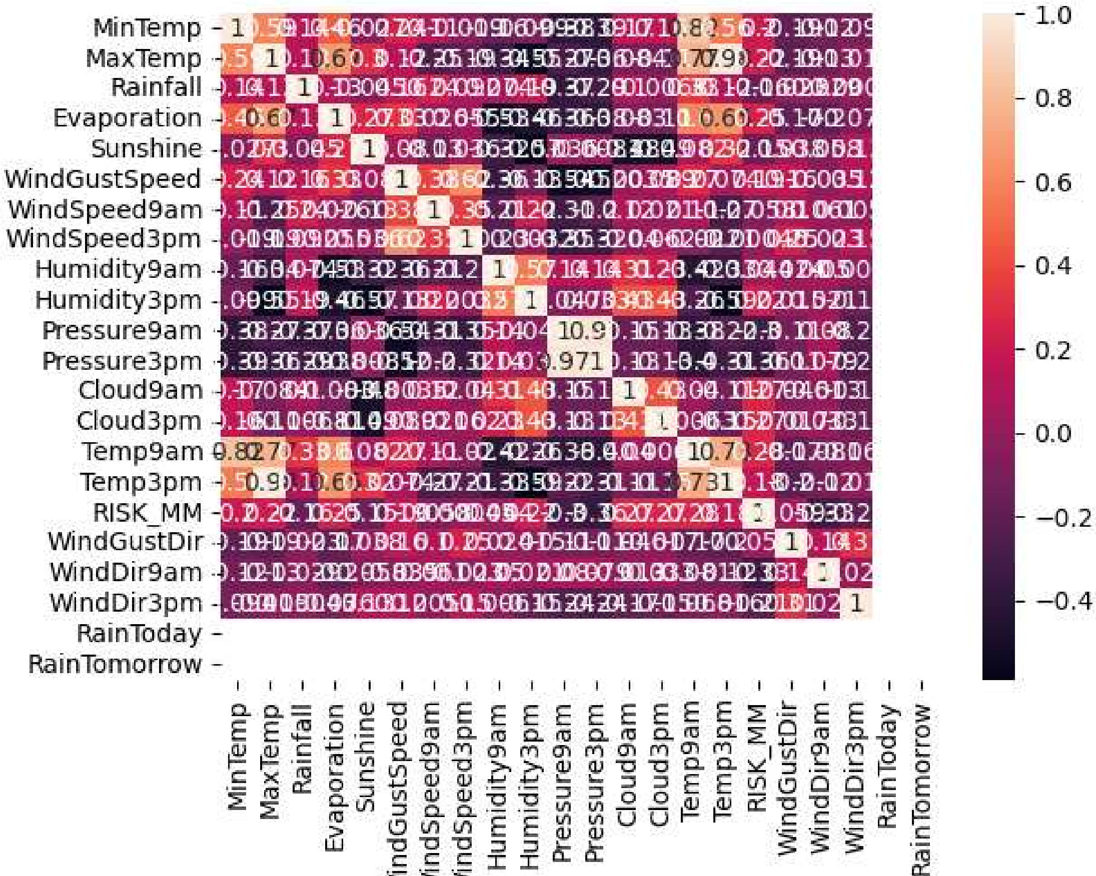
[44]: 0.30544757696266495

[79]:
```
fig, ax = plt.subplots(3, 3, figsize=(15, 10))
for i, subplot in zip(processed_df_log.columns, ax.flatten()):
    sns.scatterplot(processed_df_log[i], ax=subplot)
plt.show()
```

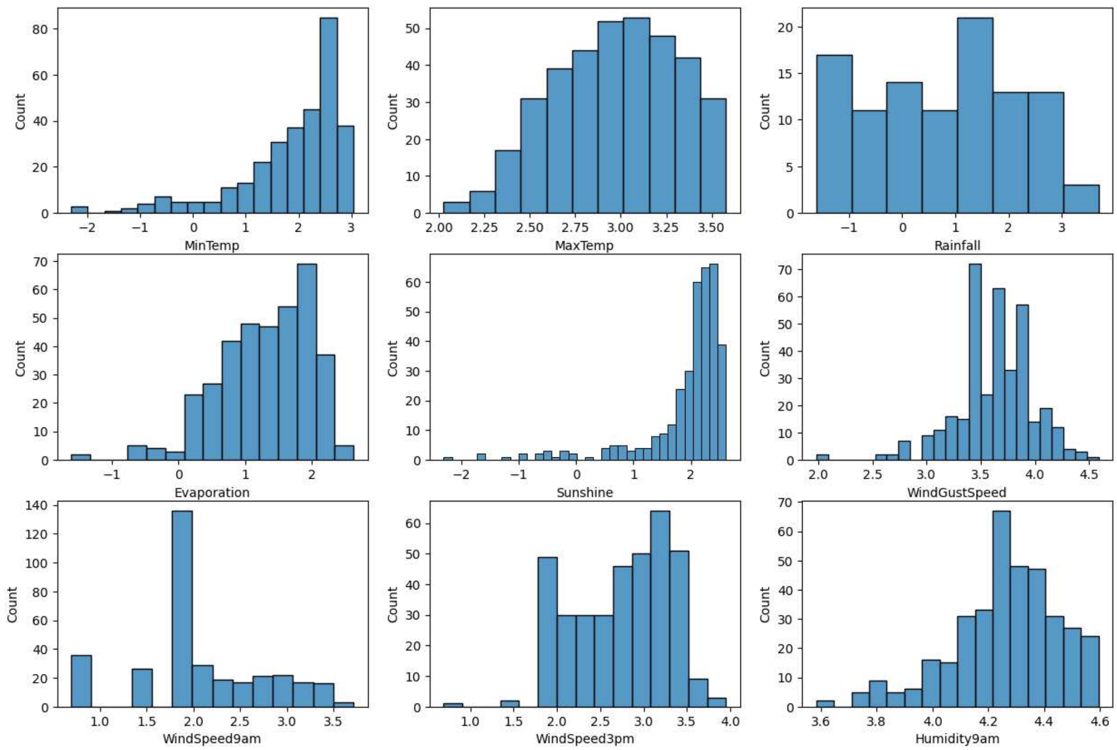



```
[46]: sns.heatmap(processed_df_log.corr(), annot=True)
```

```
[46]: <Axes: >
```

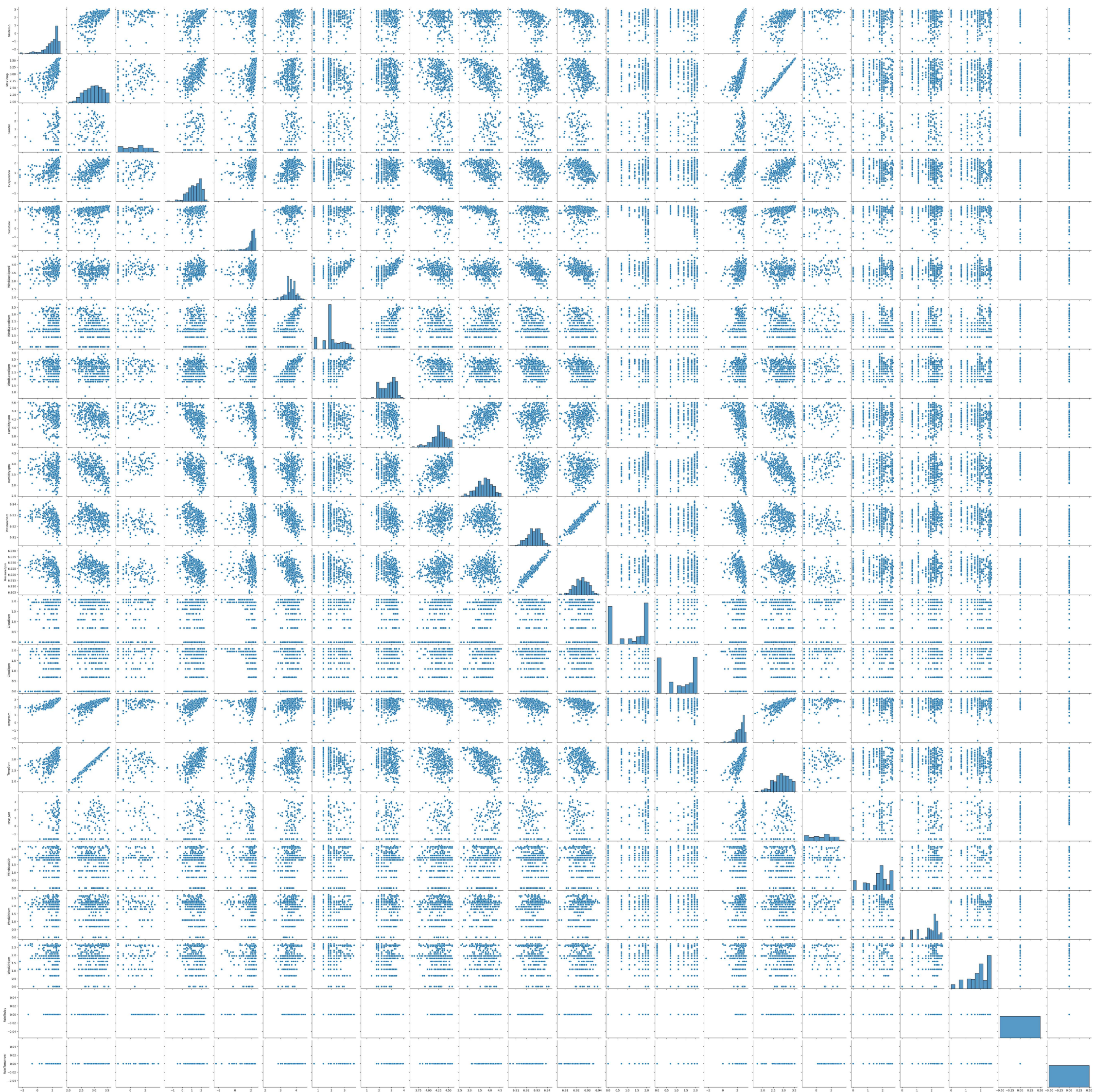


[72]:
```python
fig, ax = plt.subplots(3, 3, figsize=(15, 10))
for i, subplot in zip(processed_df_log.columns, ax.flatten()):
    sns.histplot(processed_df_log[i], ax=subplot)
plt.show()
```



[48]:
```
sns.pairplot(processed_df_log)
```

[48]: <seaborn.axisgrid.PairGrid at 0x708591164b10>



[49]:
```
processed_df_log.columns
```

[49]:
```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
       'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm',
       'Temp9am', 'Temp3pm', 'RISK_MM', 'WindGustDir', 'WindDir9am',
       'WindDir3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

[50]:
```
# sns.pairplot(pairplot_last, diag_kind= 'kde')
```

[51]:
```
processed_df_log
```

[51]:

```
      MinTemp   MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0    2.079442  3.190476       NaN     1.223775  1.840550       3.401197
1    2.639057  3.292126  1.280934     1.481605  2.272126       3.663562
2    2.617396  3.152736  1.280934     1.757858  1.193922       4.442651
3    2.587764  2.740840  3.683867     1.974081  2.208274       3.988984
4    2.028148  2.778819  1.029619     1.722767  2.360854       3.912023
..        …         …         …           …         …              …
361  2.197225  3.424263       NaN     2.028148  2.493205       4.330733
362  1.960095  3.346389       NaN     2.451005  2.541602       3.871201
363  2.525729  2.990720       NaN     2.128232  1.667707       3.761200
364  2.525729  3.292126       NaN     1.609438  1.960095       3.828641
365  2.509599  3.407842       NaN     1.791759  2.533697       4.356709

     WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  …  Cloud9am  \
0        1.791759      2.995732     4.219508     3.367296  …  1.945910
1        1.386294      2.833213     4.382027     3.583519  …  1.609438
2        1.791759      1.791759     4.406719     4.234107  …  2.079442
3        3.401197      3.178054     4.127134     4.025352  …  0.693147
4        2.995732      3.332205     4.219508     3.891820  …  1.945910
..            …             …            …            …  …       …
361      1.945910      3.912023     3.637586     2.708050  …  0.000000
362      0.693147      2.944439     3.806662     3.091042  …       NaN
363      2.397895      2.197225     4.143135     3.850148  …  1.098612
364      1.791759      3.332205     4.234107     3.663562  …  1.791759
365      3.433987      3.555348     3.761200     2.564949  …  0.000000

     Cloud3pm   Temp9am   Temp3pm   RISK_MM  WindGustDir  WindDir9am  \
0    1.945910  2.667228  3.161247  1.280934     1.945910    2.484907
1    1.098612  2.862201  3.246491  1.280934     0.000000         NaN
2    1.945910  2.734368  3.005683  3.683867     1.945910    1.098612
3    1.945910  2.602690  2.646175  1.029619     1.945910    2.639057
4    1.945910  2.406945  2.734368       NaN     2.302585    2.302585
..        …         …         …         …           …           …
361  1.098612  3.015535  3.401197       NaN     1.791759    2.302585
362  0.000000  2.844909  3.339322       NaN     1.098612    1.791759
363  0.693147  2.674149  2.906901       NaN     0.693147    0.000000
364  1.945910  2.760010  3.254243       NaN     1.945910    2.397895
365  0.000000  3.169686  3.353407       NaN     1.945910    1.945910

     WindDir3pm  RainToday  RainTomorrow
0      1.945910        NaN           0.0
1      2.564949        0.0           0.0
2      1.609438        0.0           0.0
3      2.564949        0.0           0.0
4      0.693147        0.0           NaN
..          …          …            …
361    1.945910        NaN           NaN
```

```
362    1.791759       NaN          NaN
363    0.000000       NaN          NaN
364    2.639057       NaN          NaN
365    2.639057       NaN          NaN

[366 rows x 22 columns]
```

[52]:
```
processed_df_log_new = processed_df_log.copy()
for i in processed_df_log_new.columns:
    processed_df_log_new[i] = processed_df_log_new[i].replace(np.nan,␣
 ↪processed_df_log_new[i].mean())
```

[82]:
```
# processed_df_log_new
```

[54]:
```
W = processed_df_log_new.iloc[:,:-1]
V = processed_df_log_new.iloc[:,-1]
```

[85]:
```
# print("Data :\n", W, "\n")
```

[56]:
```
W = whiten(W)
```

[57]:
```
centroids, mean_dist = kmeans(W, 2)
print("Code-book :\n", centroids, "\n")
```

```
Code-book :
 [[2.52210223e+00 9.53420945e+00 1.01201283e+00 2.56172339e+00
  2.75176729e+00 1.05582082e+01 2.96989569e+00 5.13882326e+00
  2.17590122e+01 9.07017914e+00 1.05621531e+03 1.08825750e+03
  1.35172903e+00 1.38487484e+00 4.33229197e+00 8.53975666e+00
  1.06884954e+00 2.21303459e+00 2.95462803e+00 2.71024468e+00
  0.00000000e+00]
 [1.33264334e+00 7.97570784e+00 7.86389251e-01 1.12030518e+00
  2.39969657e+00 9.90075228e+00 2.96679099e+00 5.14015655e+00
  2.24817321e+01 9.78932981e+00 1.05716151e+03 1.08928294e+03
  1.31231602e+00 1.32296791e+00 2.94173335e+00 7.03639851e+00
  7.22282725e-01 2.65915202e+00 3.41978484e+00 2.81204609e+00
  0.00000000e+00]]
```

[58]:
```
clusters, dist = vq(W, centroids)
print("Clusters :\n", clusters, "\n")
```

```
Clusters :
 [0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
```

```
1 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1
1 1 1 0 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 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 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0]
```

[59]:
```python
cluster1 = list(clusters).count(0)

cluster2 = list(clusters).count(1)
```

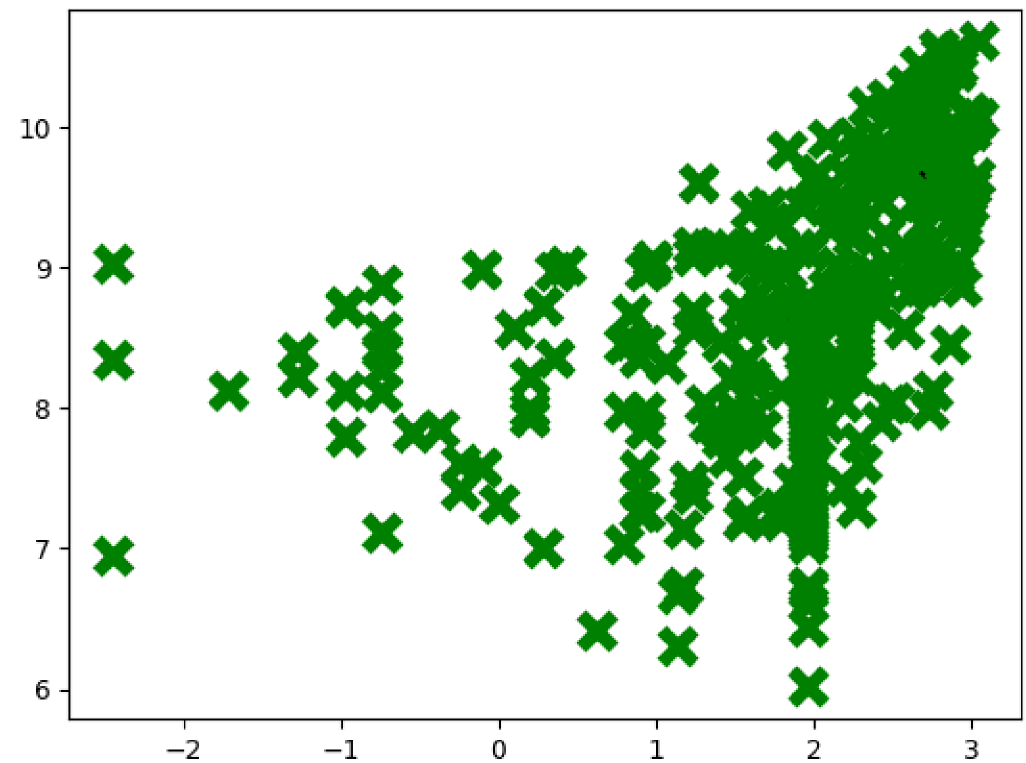
[60]:
```python
kmeanss = KMeans(n_clusters=2, random_state=42)
```

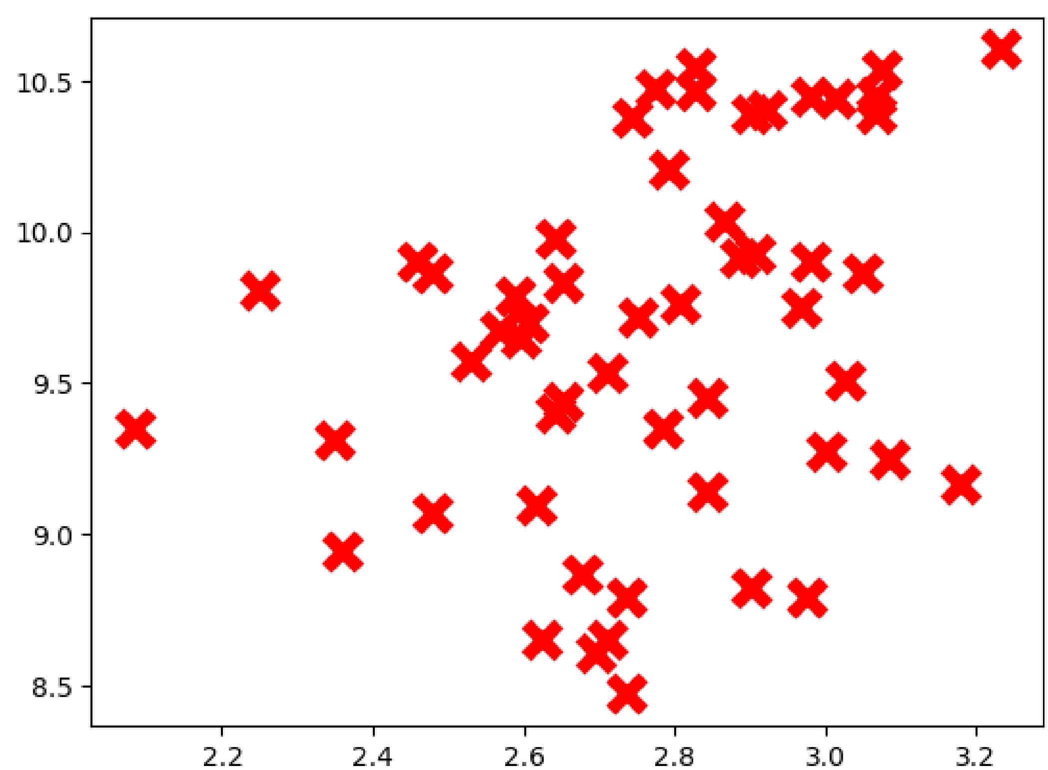
[61]:
```python
silhouette_score(W, kmeanss.fit_predict(W))
```

[61]: 0.16255911532588155

[62]:
```python
colors = 10*["g","r","c","b","k"]

class K_Means:
    def __init__(self, k=2, tol=0.001, max_iter=300):
        self.k = k
        self.tol = tol
        self.max_iter = max_iter
    def fit(self,data):

        self.centroids = {}

        for i in range(self.k):
            self.centroids[i] = data[i]
        for i in range(self.max_iter):
            self.classifications = {}

            for i in range(self.k):
                self.classifications[i] = []

            for featureset in data:
                distances = [np.linalg.norm(featureset-self.
↪centroids[centroid]) for centroid in self.centroids]
                classification = distances.index(min(distances))
                self.classifications[classification].append(featureset)

            prev_centroids = dict(self.centroids)

            for classification in self.classifications:
```

```
                self.centroids[classification] = np.average(self.classifications[classification],axis=0)

            optimized = True

            for c in self.centroids:
                original_centroid = prev_centroids[c]
                current_centroid = self.centroids[c]
                if np.sum((current_centroid-original_centroid)/original_centroid*100.0) > self.tol:
                    print(np.sum((current_centroid-original_centroid)/original_centroid*100.0))
                    optimized = False

            if optimized:
                break

    def predict(self,data):
            distances = [np.linalg.norm(data-self.centroids[centroid]) for centroid in self.centroids]
            classification = distances.index(min(distances))
            return classification

clf = K_Means()
clf.fit(W)

for centroid in clf.centroids:
    plt.scatter(clf.centroids[centroid][0], clf.centroids[centroid][1],
    marker="o", color="k", s=150, linewidths=5)

for classification in clf.classifications:
    color = colors[classification]
    for featureset in clf.classifications[classification]:
        plt.scatter(featureset[0], featureset[1], marker="x", color=color, s=150, linewidths=5)
    plt.show()
```


10
9
8
7
6
−2
−1
0
1
2
3


10.5
10.0
9.5
9.0
8.5
2.2
2.4
2.6
2.8
3.0
3.2

day-5

February 27, 2024

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_wine
import warnings
warnings.filterwarnings('ignore')
```

[6]:
```
# Load the data
data = load_wine()
```

[7]:
```
data.keys()
```

[7]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names'])
```

[8]:
```
# Description of the data
print(data.DESCR)
```

```
.. _wine_dataset:

Wine recognition dataset
------------------------

**Data Set Characteristics:**

:Number of Instances: 178
:Number of Attributes: 13 numeric, predictive attributes and the class
:Attribute Information:
    - Alcohol
    - Malic acid
    - Ash
    - Alcalinity of ash
    - Magnesium
    - Total phenols
    - Flavanoids
    - Nonflavanoid phenols
    - Proanthocyanins
```

```
    - Color intensity
    - Hue
    - OD280/OD315 of diluted wines
    - Proline
    - class:
        - class_0
        - class_1
        - class_2

:Summary Statistics:

============================= ==== ===== ======= =====
                                Min   Max   Mean     SD
============================= ==== ===== ======= =====
Alcohol:                      11.0  14.8    13.0   0.8
Malic Acid:                   0.74  5.80    2.34  1.12
Ash:                          1.36  3.23    2.36  0.27
Alcalinity of Ash:            10.6  30.0    19.5   3.3
Magnesium:                    70.0 162.0    99.7  14.3
Total Phenols:                0.98  3.88    2.29  0.63
Flavanoids:                   0.34  5.08    2.03  1.00
Nonflavanoid Phenols:         0.13  0.66    0.36  0.12
Proanthocyanins:              0.41  3.58    1.59  0.57
Colour Intensity:              1.3  13.0     5.1   2.3
Hue:                          0.48  1.71    0.96  0.23
OD280/OD315 of diluted wines: 1.27  4.00    2.61  0.71
Proline:                       278  1680     746   315
============================= ==== ===== ======= =====

:Missing Attribute Values: None
:Class Distribution: class_0 (59), class_1 (71), class_2 (48)
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

This is a copy of UCI ML Wine recognition datasets.
https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

The data is the results of a chemical analysis of wines grown in the same
region in Italy by three different cultivators. There are thirteen different
measurements taken for different constituents found in the three types of
wine.

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies,
```

```
Via Brigata Salerno, 16147 Genoa, Italy.

Citation:

Lichman, M. (2013). UCI Machine Learning Repository
[https://archive.ics.uci.edu/ml]. Irvine, CA: University of California,
School of Information and Computer Science.

|details-start|
**References**
|details-split|

(1) S. Aeberhard, D. Coomans and O. de Vel,
Comparison of Classifiers in High Dimensional Settings,
Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
Mathematics and Statistics, James Cook University of North Queensland.
(Also submitted to Technometrics).

The data was used with many others for comparing various
classifiers. The classes are separable, though only RDA
has achieved 100% correct classification.
(RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
(All results using the leave-one-out technique)

(2) S. Aeberhard, D. Coomans and O. de Vel,
"THE CLASSIFICATION PERFORMANCE OF RDA"
Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
Mathematics and Statistics, James Cook University of North Queensland.
(Also submitted to Journal of Chemometrics).

|details-end|
```

[9]:
```python
# Features
data.feature_names
```

[9]:
```
['alcohol',
 'malic_acid',
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
```

```
 'od280/od315_of_diluted_wines',
 'proline']
```

[10]:
```
# Create a dataframe with the data
df = pd.DataFrame(data.data, columns=data.feature_names)
df.head()
```

[10]:
```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80   
1    13.20        1.78  2.14               11.2      100.0           2.65   
2    13.16        2.36  2.67               18.6      101.0           2.80   
3    14.37        1.95  2.50               16.8      113.0           3.85   
4    13.24        2.59  2.87               21.0      118.0           2.80   

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04   
1        2.76                  0.26             1.28             4.38  1.05   
2        3.24                  0.30             2.81             5.68  1.03   
3        3.49                  0.24             2.18             7.80  0.86   
4        2.69                  0.39             1.82             4.32  1.04   

   od280/od315_of_diluted_wines  proline  
0                          3.92   1065.0  
1                          3.40   1050.0  
2                          3.17   1185.0  
3                          3.45   1480.0  
4                          2.93    735.0  
```

[11]:
```
# Add the target and target names to the dataframe
df['target'] = data.target
df['target_names'] = df.target.apply(lambda x: data.target_names[x])
```

[12]:
```
df.head().T
```

[12]:
```
                                  0      1      2      3      4
alcohol                       14.23   13.2  13.16  14.37  13.24
malic_acid                     1.71   1.78   2.36   1.95   2.59
ash                            2.43   2.14   2.67    2.5   2.87
alcalinity_of_ash              15.6   11.2   18.6   16.8   21.0
magnesium                     127.0  100.0  101.0  113.0  118.0
total_phenols                   2.8   2.65    2.8   3.85    2.8
flavanoids                     3.06   2.76   3.24   3.49   2.69
nonflavanoid_phenols           0.28   0.26    0.3   0.24   0.39
proanthocyanins                2.29   1.28   2.81   2.18   1.82
color_intensity                5.64   4.38   5.68    7.8   4.32
hue                            1.04   1.05   1.03   0.86   1.04
od280/od315_of_diluted_wines   3.92    3.4   3.17   3.45   2.93
```

```
proline                          1065.0   1050.0   1185.0   1480.0    735.0
target                                0        0        0        0        0
target_names                    class_0  class_0  class_0  class_0  class_0
```

[13]:
```
# Check for null values
df.isnull().sum()
```

[13]:
```
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
target                          0
target_names                    0
dtype: int64
```

[14]:
```
# Check the distribution of the target
df.target_names.value_counts()
```

[14]:
```
target_names
class_1    71
class_0    59
class_2    48
Name: count, dtype: int64
```

[15]:
```
# Check the distribution of the features
df.describe().T
```

[15]:
```
                              count       mean        std    min      25%  \
alcohol                       178.0  13.000618   0.811827  11.03  12.3625
malic_acid                    178.0   2.336348   1.117146   0.74   1.6025
ash                           178.0   2.366517   0.274344   1.36   2.2100
alcalinity_of_ash             178.0  19.494944   3.339564  10.60  17.2000
magnesium                     178.0  99.741573  14.282484  70.00  88.0000
total_phenols                 178.0   2.295112   0.625851   0.98   1.7425
flavanoids                    178.0   2.029270   0.998859   0.34   1.2050
nonflavanoid_phenols          178.0   0.361854   0.124453   0.13   0.2700
proanthocyanins               178.0   1.590899   0.572359   0.41   1.2500
color_intensity               178.0   5.058090   2.318286   1.28   3.2200
```

```
hue                           178.0    0.957449    0.228572    0.48    0.7825
od280/od315_of_diluted_wines  178.0    2.611685    0.709990    1.27    1.9375
proline                       178.0  746.893258  314.907474  278.00  500.5000
target                        178.0    0.938202    0.775035    0.00    0.0000

                                  50%       75%      max
alcohol                        13.050   13.6775    14.83
malic_acid                      1.865    3.0825     5.80
ash                             2.360    2.5575     3.23
alcalinity_of_ash              19.500   21.5000    30.00
magnesium                      98.000  107.0000   162.00
total_phenols                   2.355    2.8000     3.88
flavanoids                      2.135    2.8750     5.08
nonflavanoid_phenols            0.340    0.4375     0.66
proanthocyanins                 1.555    1.9500     3.58
color_intensity                 4.690    6.2000    13.00
hue                             0.965    1.1200     1.71
od280/od315_of_diluted_wines    2.780    3.1700     4.00
proline                       673.500  985.0000  1680.00
target                          1.000    2.0000     2.00
```

[16]:

```python
df.dtypes
```

[16]:

```
alcohol                         float64
malic_acid                      float64
ash                             float64
alcalinity_of_ash               float64
magnesium                       float64
total_phenols                   float64
flavanoids                      float64
nonflavanoid_phenols            float64
proanthocyanins                 float64
color_intensity                 float64
hue                             float64
od280/od315_of_diluted_wines    float64
proline                         float64
target                            int64
target_names                     object
dtype: object
```

[17]:

```python
obj_df = df.select_dtypes(include=['object']).copy()
num_df = df.select_dtypes(include=['float64', 'int64']).copy()
```

[18]:

```python
#label encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
label_df = obj_df.apply(le.fit_transform)
```

[19]:
```
label_df.head()
```

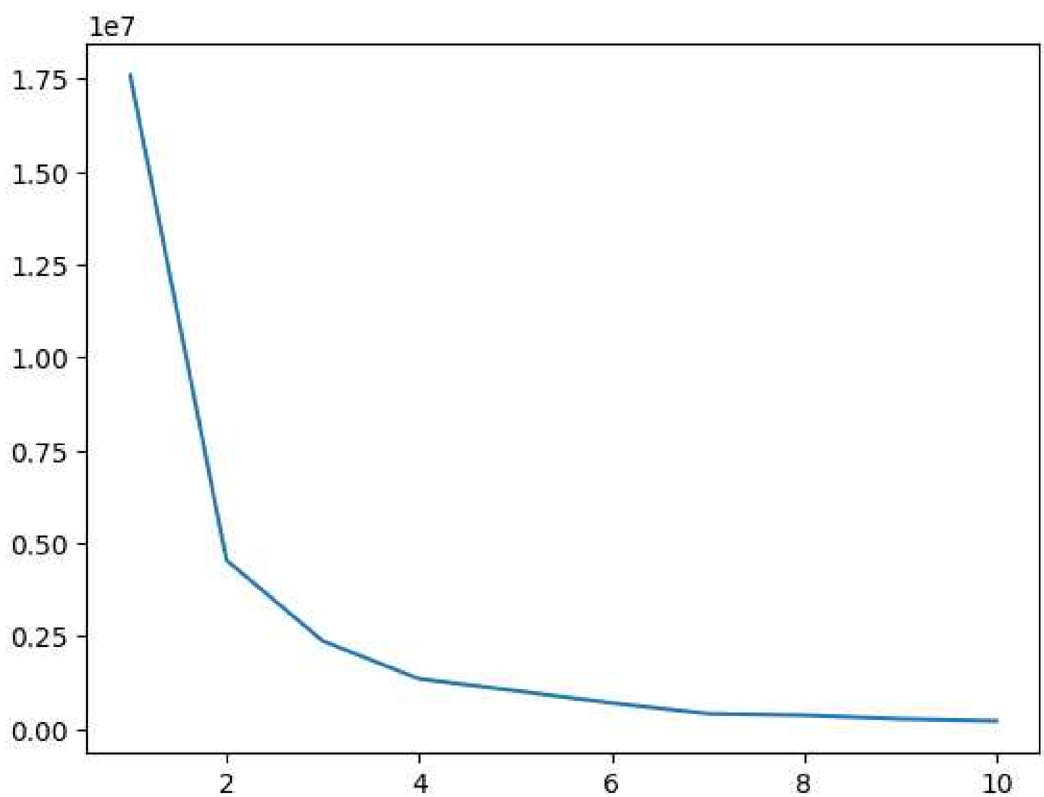
[19]:

	target_names
0	0
1	0
2	0
3	0
4	0

[20]:
```
#concatenate the label and numerical dataframes
new_df = pd.concat([num_df, label_df], axis=1)
```

[21]:
```
#findin optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(new_df)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
```

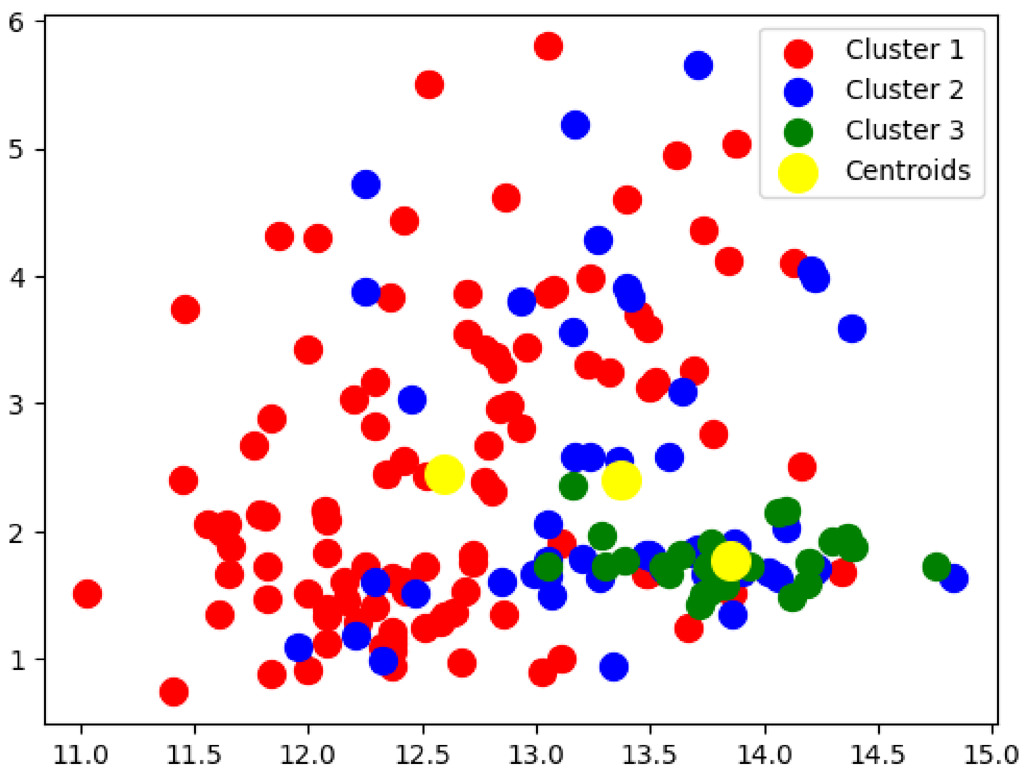
[21]: [<matplotlib.lines.Line2D at 0x76a4d016a250>]



[22]:
```
#fitting kmeans to the dataset
kmeans = KMeans(n_clusters=3)
y_kmeans = kmeans.fit_predict(new_df)
```

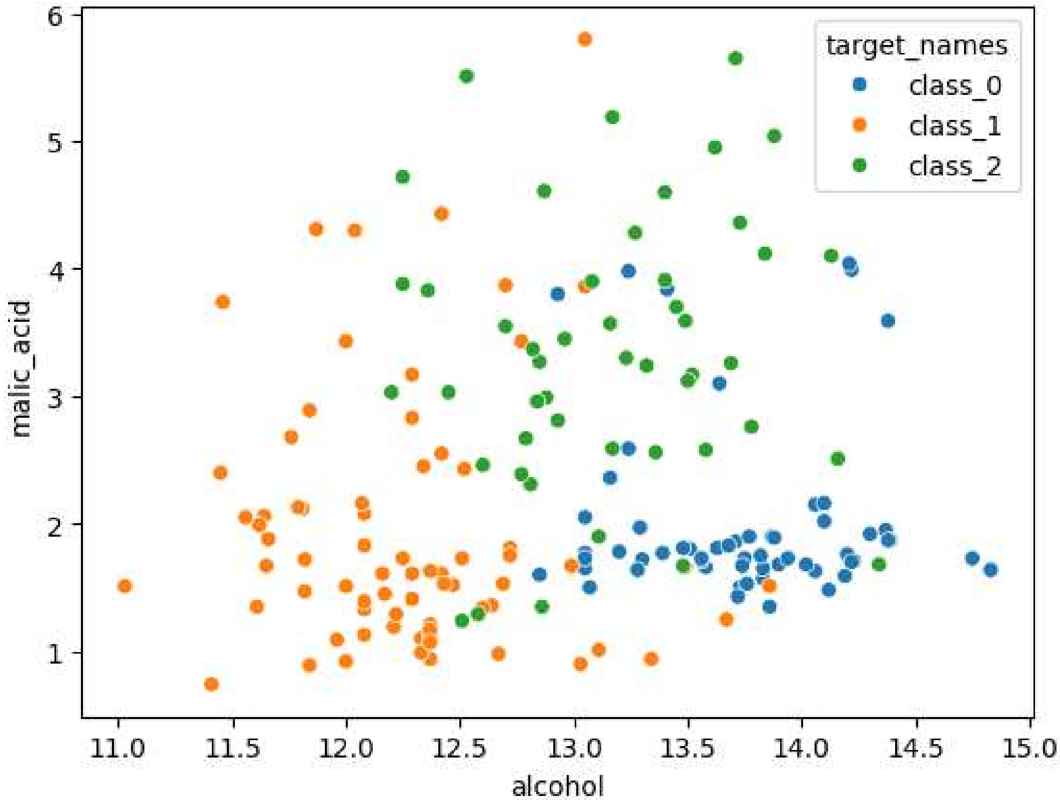
[23]:
```
plt.scatter(new_df.iloc[y_kmeans == 0, 0], new_df.iloc[y_kmeans == 0, 1], s =
 ↪100, c = 'red', label = 'Cluster 1')
plt.scatter(new_df.iloc[y_kmeans == 1, 0], new_df.iloc[y_kmeans == 1, 1], s =
 ↪100, c = 'blue', label = 'Cluster 2')
plt.scatter(new_df.iloc[y_kmeans == 2, 0], new_df.iloc[y_kmeans == 2, 1], s =
 ↪100, c = 'green', label = 'Cluster 3')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s =
 ↪200, c = 'yellow', label = 'Centroids')
plt.legend()
```

[23]:
```
<matplotlib.legend.Legend at 0x76a4cdb67ed0>
```



[24]:
```
sns.scatterplot(x='alcohol', y='malic_acid', hue='target_names', data=df)
```

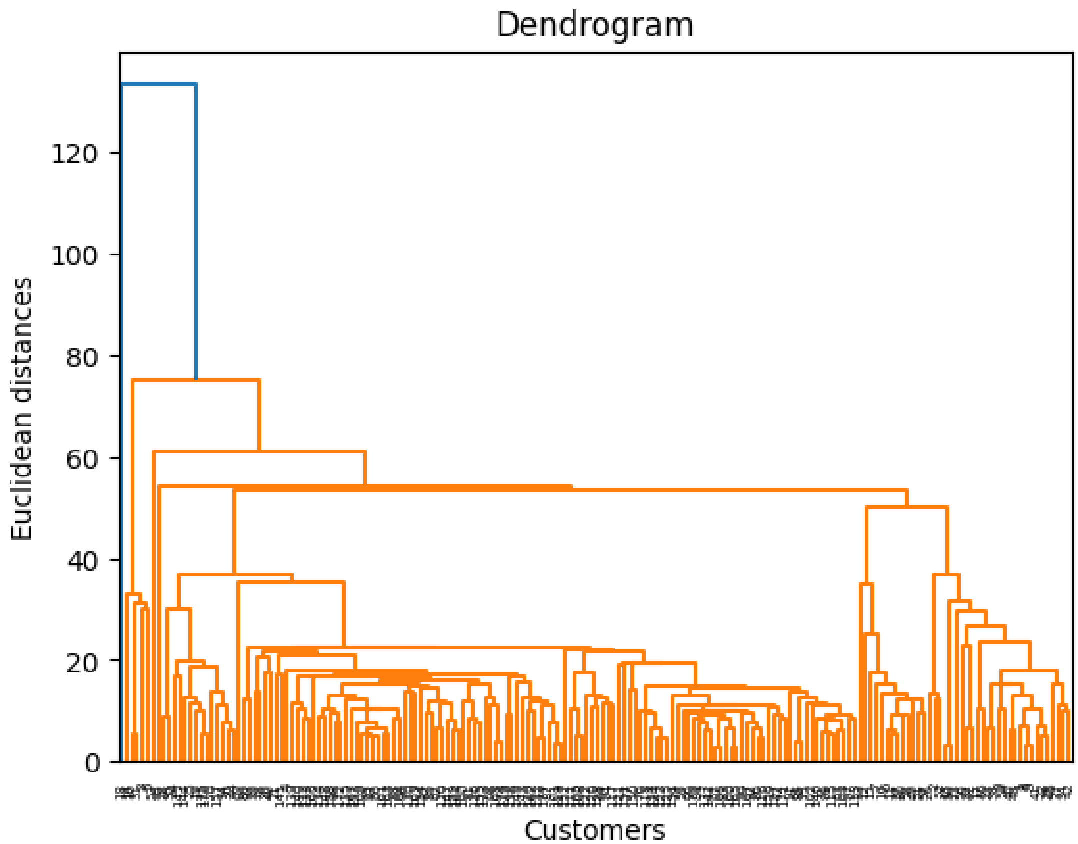
[24]:
```
<Axes: xlabel='alcohol', ylabel='malic_acid'>
```



[25]:
```python
#shilouette score
from sklearn.metrics import silhouette_score
silhouette_score(new_df, y_kmeans)
```

[25]: 0.55956126417174

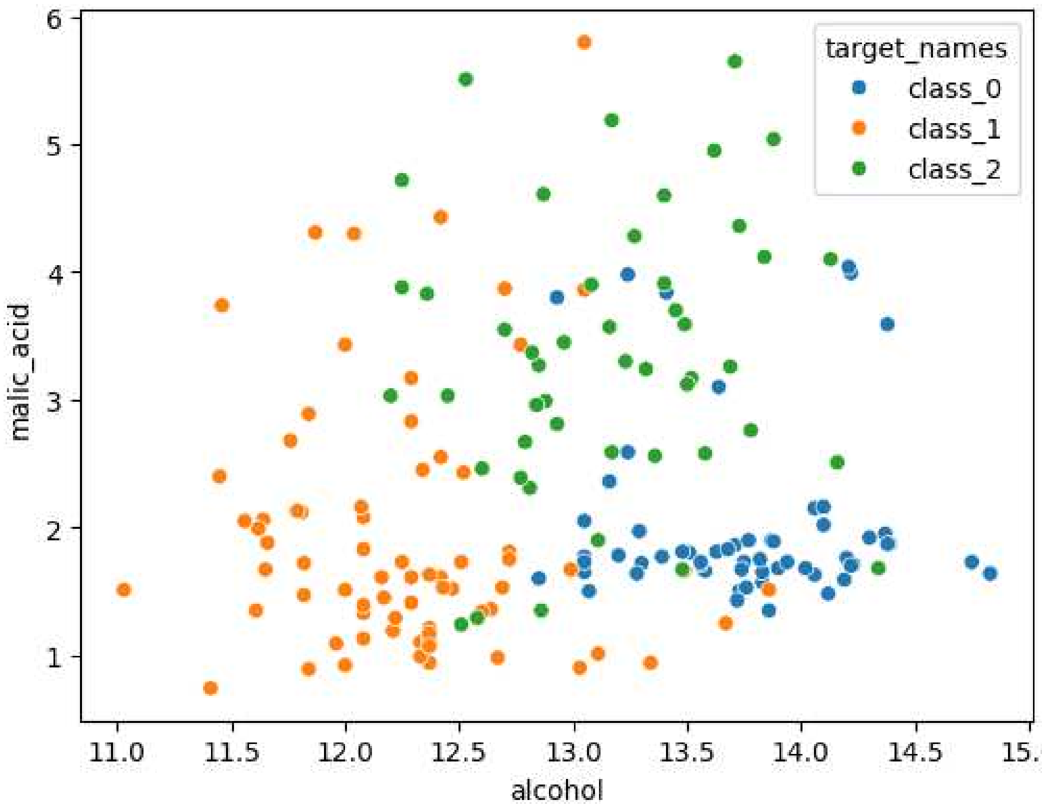
[26]:
```python
#hierarchical clustering
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(new_df, method='single'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



[27]:
```
#DBSCAN
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=3, min_samples=4)
y_dbscan = dbscan.fit_predict(new_df)
```

[28]:
```
sns.scatterplot(x='alcohol', y='malic_acid', hue='target_names', data=df)
```

[28]: <Axes: xlabel='alcohol', ylabel='malic_acid'>



[29]:
```
# #shilouette score
# from sklearn.metrics import silhouette_score
# silhouette_score(new_df, y_dbscan)
```

[ ]:
```

```

day-6

February 27, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv('placement.csv')
```

[3]:
```
df.isna().sum()
```

[3]:
```
cgpa       0
package    0
dtype: int64
```

[4]:
```
df.dtypes
```

[4]:
```
cgpa       float64
package    float64
dtype: object
```

[5]:
```
def outliers(df, ft):
    q1 = df[ft].quantile(0.25)
    q3 = df[ft].quantile(0.75)
    iqr = q3 - q1
    ub = q3 + 1.5 * iqr
    lb = q1 - 1.5 * iqr
    ls = df.index[(df[ft] > ub) | (df[ft] < lb)]
    return ls

def replace_outliers_with_mean(df, ft):
    q1 = df[ft].quantile(0.25)
    q3 = df[ft].quantile(0.75)
    iqr = q3 - q1
    ub = q3 + 1.5 * iqr
    lb = q1 - 1.5 * iqr
```

```
    outliers_indices = df.index[(df[ft] >= ub) | (df[ft] <= lb)]


    df.loc[outliers_indices, ft] = df[ft].mean()

    return df

index_ls = []

for i in df.columns:
    index_ls.extend(outliers(df, i))
print(index_ls)

for i in df.columns:
    df = replace_outliers_with_mean(df, i)
```

```
[]
```

[6]:
```
# #standard scaling
# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# df = scaler.fit_transform(df)
```

[7]:
```
#linear regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X = df.iloc[:, :-1]
y = df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[8]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

[8]:
```
LinearRegression()
```

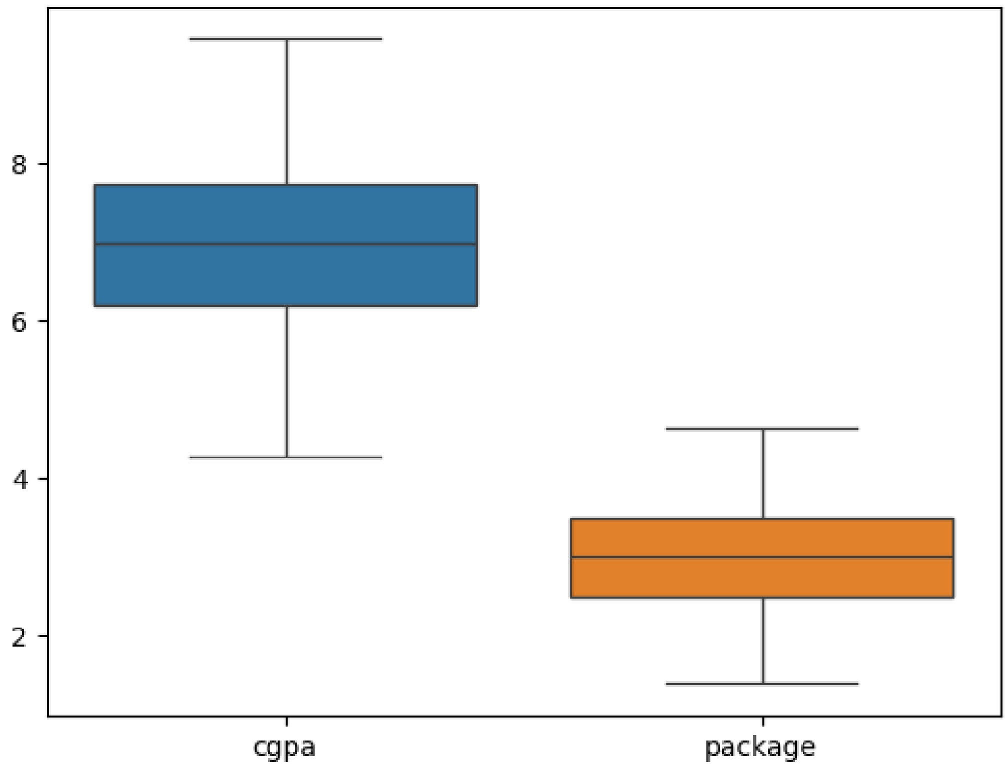
[9]:
```
y_pred = regressor.predict(X_test)
```

[10]:
```
#r2 score
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print(r2)
```

```
0.7730984312051673
```

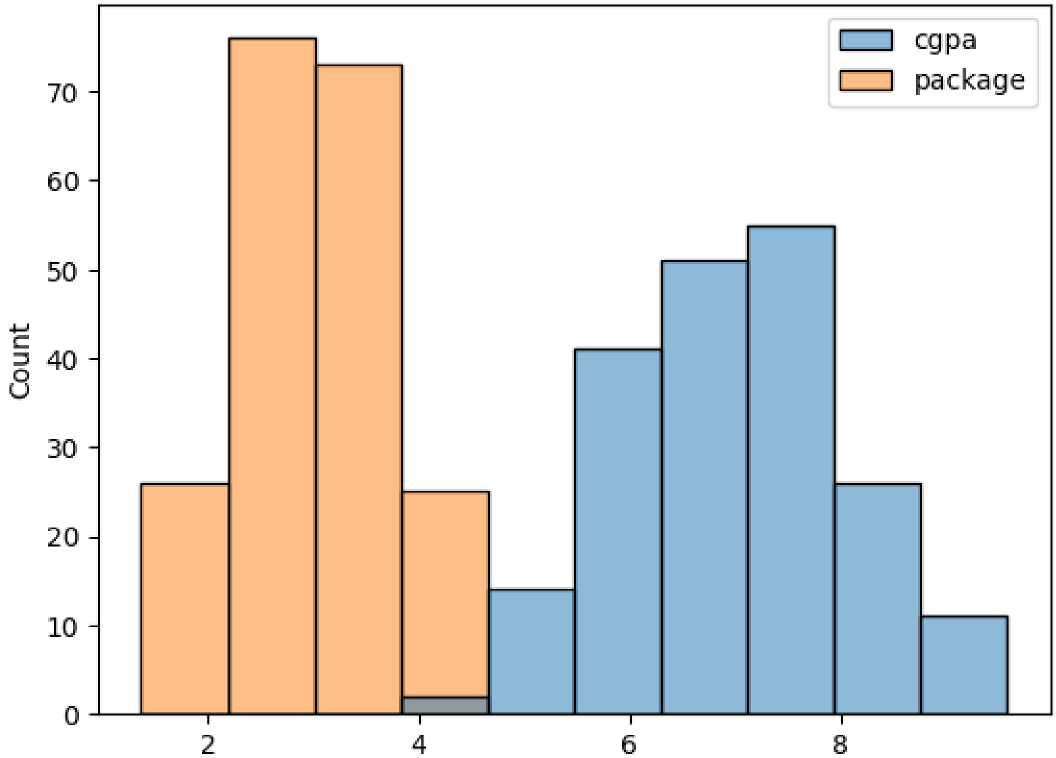
[11]:
```
#boxplot
sns.boxplot(data=df)
```

[11]: <Axes: >



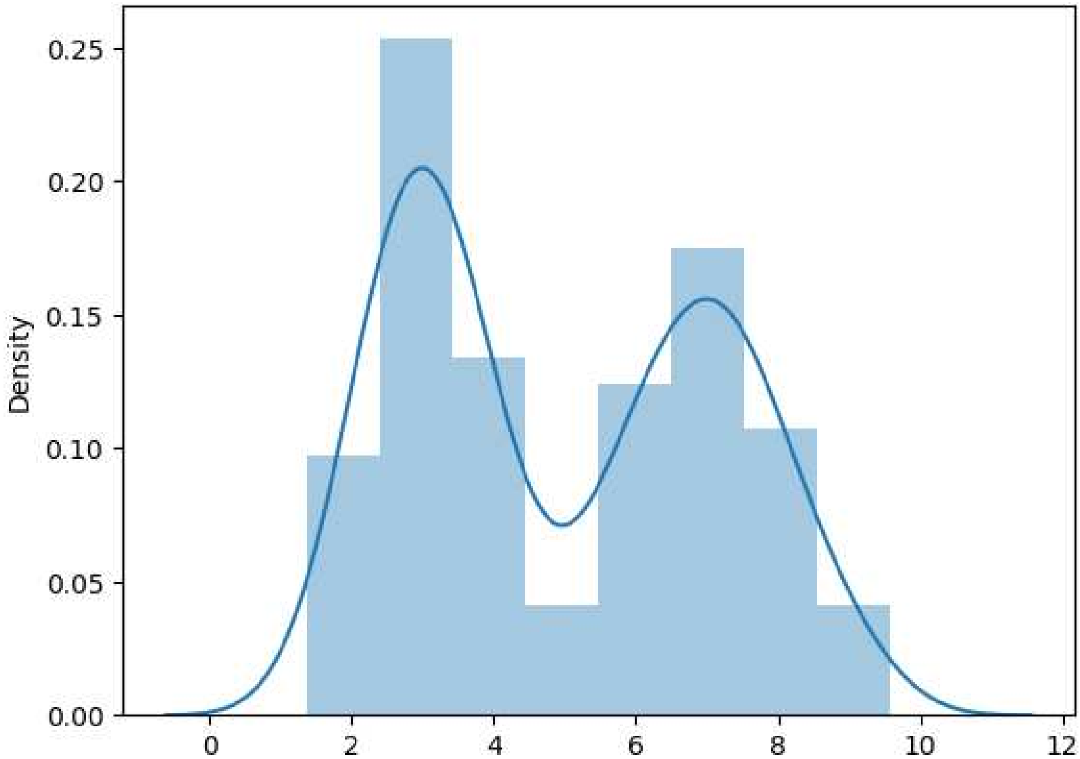
[12]:
```
#distplot
sns.histplot(df)
```

[12]: <Axes: ylabel='Count'>



[13]:
```
sns.distplot(df)
```

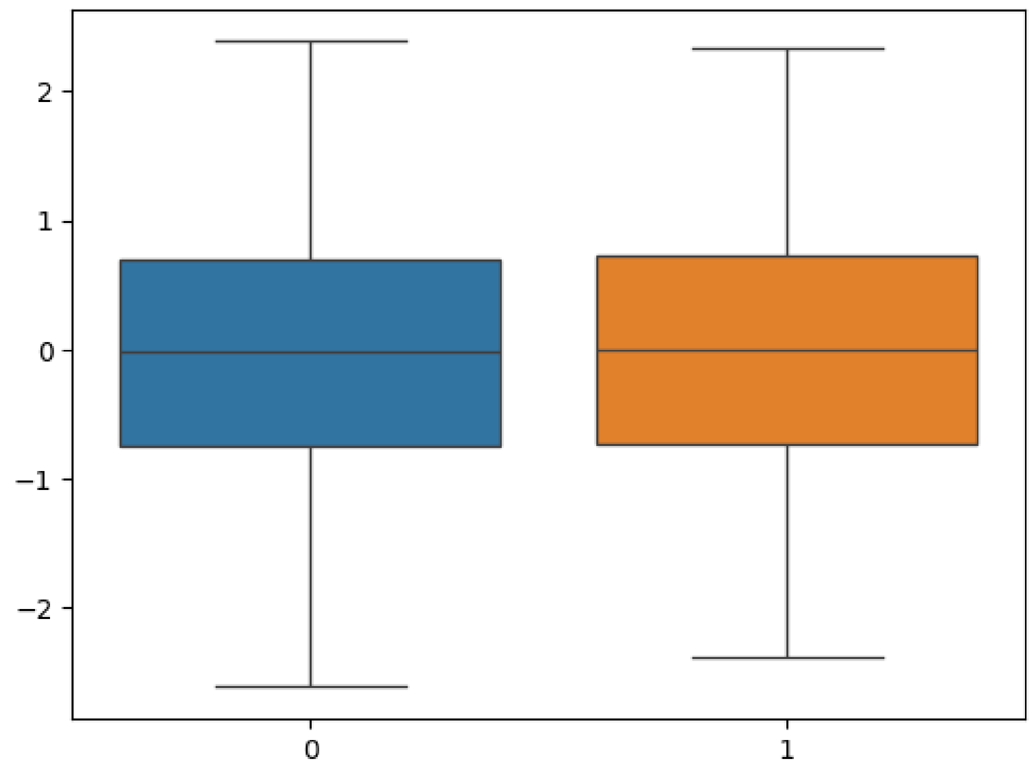
[13]: <Axes: ylabel='Density'>



[14]:
```
#yeo-johnson transformation
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
new_df = pd.DataFrame(pt.fit_transform(df))
```

[15]:
```
#boxplot
sns.boxplot(data=new_df)
```

[15]: `<Axes: >`



[16]:
```
W = new_df.iloc[:, :-1]
z = new_df.iloc[:, -1]

W_train, W_test, z_train, z_test = train_test_split(W, z, test_size=0.2,␣
↪random_state=0)
```

[17]:
```
regressor = LinearRegression()
regressor.fit(W_train, z_train)

z_pred = regressor.predict(W_test)

#r2 score
r2 = r2_score(z_test, z_pred)
print(r2)
```

0.7307234904168096

[18]:
```
#random forest regression
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=10, random_state=42)
regressor.fit(W_train, z_train)
```

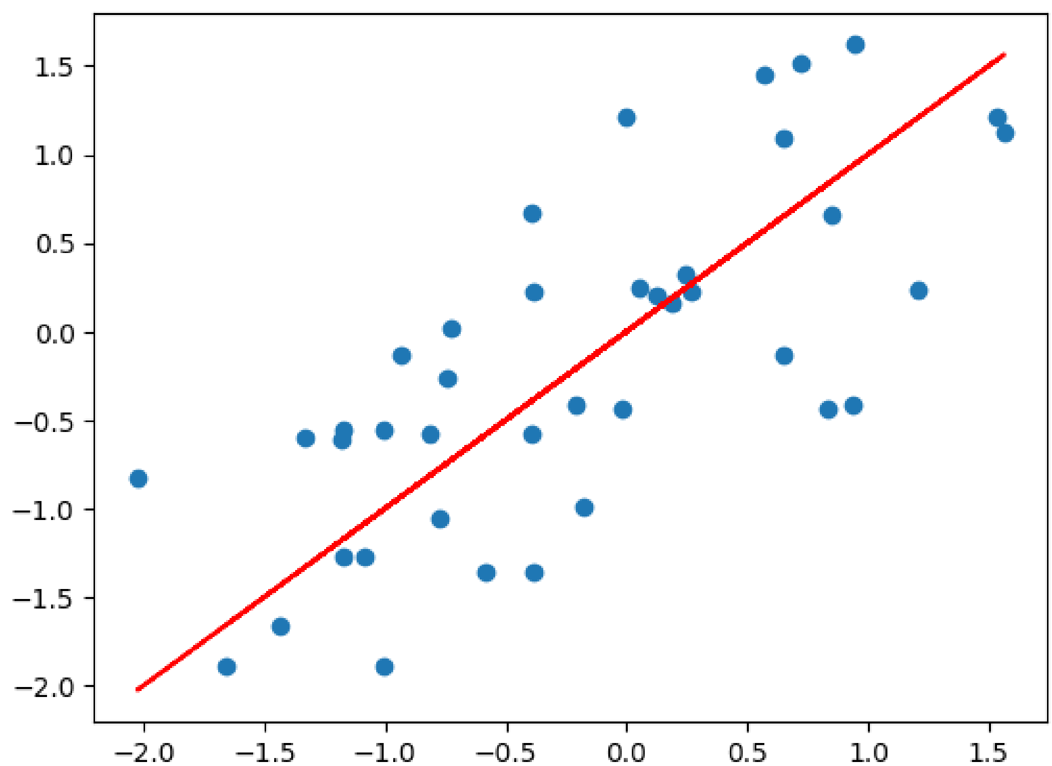
[18]: RandomForestRegressor(n_estimators=10, random_state=42)

[19]:
```
z_pred = regressor.predict(W_test)

#r2 score
r2 = r2_score(z_test, z_pred)
print(r2)
```

```
0.4254801167712502
```

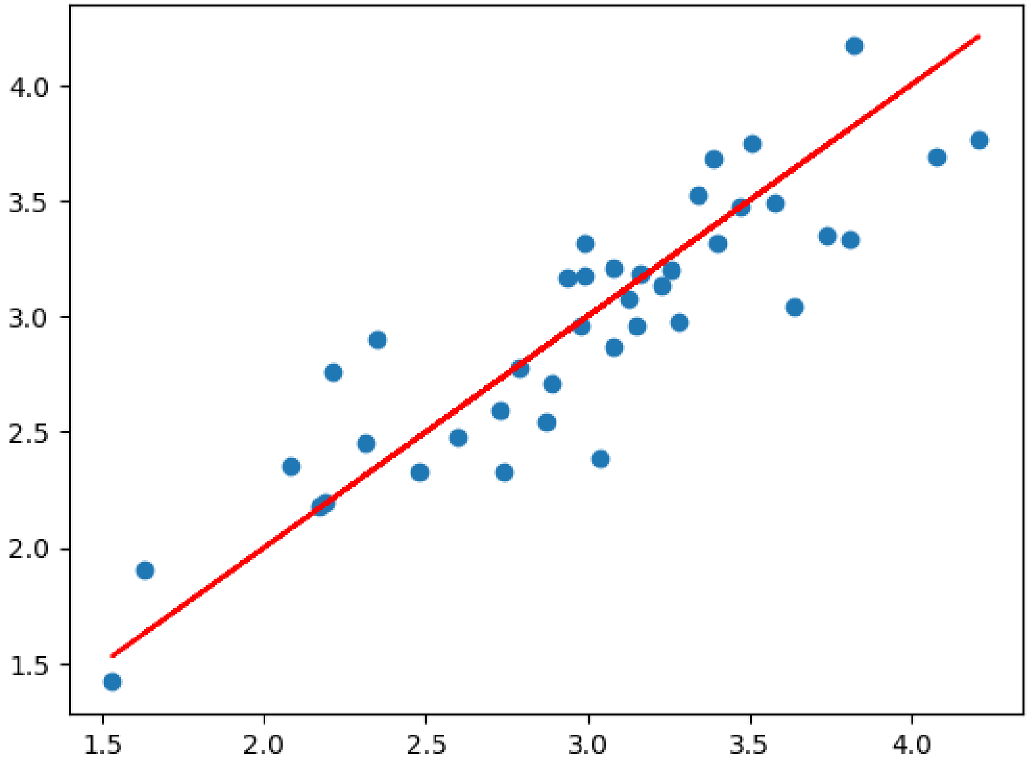
[20]:
```
#scatter plot with slope
plt.scatter(z_test, z_pred)
plt.plot(z_test, z_test, color='red')
```

[20]: [<matplotlib.lines.Line2D at 0x77b65363f790>]



[21]:
```
#scatter plot with slope
plt.scatter(y_test, y_pred)
plt.plot(y_test, y_test, color='red')
print('Coefficient of determination: %.2f' % r2_score(y_test, y_pred))
```

```
Coefficient of determination: 0.77
```


4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0

```
[ ]:
```

day-7

February 27, 2024

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```python
df = pd.read_csv('auto-mpg.csv')
```

[3]:
```python
df.head()
```

[3]:
```
    mpg  cylinders  displacement horsepower  weight  acceleration  model year  \
0  18.0          8         307.0        130    3504          12.0          70
1  15.0          8         350.0        165    3693          11.5          70
2  18.0          8         318.0        150    3436          11.0          70
3  16.0          8         304.0        150    3433          12.0          70
4  17.0          8         302.0        140    3449          10.5          70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

[4]:
```python
df.dtypes
```

[4]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

[5]:
```
obj_df = df.select_dtypes(include=['object']).copy()
num_df = df.select_dtypes(include=['float64', 'int64']).copy()
```

[6]:
```
obj_df.head().T
```

[6]:
```
                                  0                  1                    2  \
horsepower                      130                165                  150
car name    chevrolet chevelle malibu  buick skylark 320  plymouth satellite

                        3            4
horsepower            150          140
car name    amc rebel sst  ford torino
```

[7]:
```
num_df.head().T
```

[7]:
```
                   0       1       2       3       4
mpg             18.0    15.0    18.0    16.0    17.0
cylinders        8.0     8.0     8.0     8.0     8.0
displacement   307.0   350.0   318.0   304.0   302.0
weight        3504.0  3693.0  3436.0  3433.0  3449.0
acceleration    12.0    11.5    11.0    12.0    10.5
model year      70.0    70.0    70.0    70.0    70.0
origin           1.0     1.0     1.0     1.0     1.0
```

[8]:
```
#IQR method
Q1 = num_df.quantile(0.25)
Q3 = num_df.quantile(0.75)
IQR = Q3 - Q1

print(IQR)
```

```
mpg               11.50
cylinders          4.00
displacement     157.75
weight          1384.25
acceleration       3.35
model year         6.00
origin             1.00
dtype: float64
```

[9]:
```
print((num_df < (Q1 - 1.5 * IQR)) | (num_df > (Q3 + 1.5 * IQR)))
```

```
       mpg  cylinders  displacement  weight  acceleration  model year  origin
0    False      False         False   False         False       False   False
1    False      False         False   False         False       False   False
2    False      False         False   False         False       False   False
3    False      False         False   False         False       False   False
4    False      False         False   False         False       False   False
```

```
..   …          …            …       …           …          …       …
393  False      False          False   False         False        False   False
394  False      False          False   False          True        False   False
395  False      False          False   False         False        False   False
396  False      False          False   False         False        False   False
397  False      False          False   False         False        False   False

[398 rows x 7 columns]
```

[10]:
```
#Removing outliers
num_df_out = num_df[~((num_df < (Q1 - 1.5 * IQR)) | (num_df > (Q3 + 1.5 *␣
 ↪IQR))).any(axis=1)]
```

[11]:
```
obj_df.head().T
```

[11]:
```
                                   0                  1                    2  \
horsepower                       130                165                  150
car name    chevrolet chevelle malibu  buick skylark 320  plymouth satellite

                        3            4
horsepower            150          140
car name    amc rebel sst  ford torino
```

[12]:
```
#label encoding
from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()
label_obj_df = obj_df.apply(lb_make.fit_transform)
```

[13]:
```
label_obj_df.head().T
```

[13]:
```
             0   1    2   3    4
horsepower  15  33   27  27   22
car name    49  36  231  14  161
```

[14]:
```
new_df = pd.concat([num_df_out, label_obj_df], axis=1)
```

[15]:
```
new_df.head().T
```

[15]:
```
                   0       1       2       3       4
mpg             18.0    15.0    18.0    16.0    17.0
cylinders        8.0     8.0     8.0     8.0     8.0
displacement   307.0   350.0   318.0   304.0   302.0
weight        3504.0  3693.0  3436.0  3433.0  3449.0
acceleration    12.0    11.5    11.0    12.0    10.5
model year      70.0    70.0    70.0    70.0    70.0
origin           1.0     1.0     1.0     1.0     1.0
```

```
horsepower      15.0    33.0    27.0    27.0    22.0
car name        49.0    36.0   231.0    14.0   161.0
```

[16]:
```
new_df.isna().sum()
```

[16]:
```
mpg             8
cylinders       8
displacement    8
weight          8
acceleration    8
model year      8
origin          8
horsepower      0
car name        0
dtype: int64
```

[17]:
```
mena_num_df = new_df.fillna(new_df.mean())
```

lasso

[18]:
```
#lasso regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error

X = mena_num_df.drop('mpg', axis=1)
y = mena_num_df['mpg']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

[19]:
```
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)
```

[19]:
```
Lasso(alpha=0.1)
```

[20]:
```
y_pred = lasso.predict(X_test)
```

[21]:
```
print(mean_squared_error(y_test, y_pred))
```

```
10.834734885814798
```

[22]:
```
#r2 score
from sklearn.metrics import r2_score

print(r2_score(y_test, y_pred))
```

```
0.8258432055490966
```

Ridge

[23]:
```
#ridge regression
from sklearn.linear_model import Ridge

ridge = Ridge(alpha=1)
ridge.fit(X_train, y_train)
```

[23]:
```
Ridge(alpha=1)
```

[24]:
```
y_pred = ridge.predict(X_test)
```

[25]:
```
print(mean_squared_error(y_test, y_pred))
```

```
10.746788369431064
```

[26]:
```
print(r2_score(y_test, y_pred))
```

```
0.8272568518946632
```

ElasticNet

[27]:
```
# elastic net
from sklearn.linear_model import ElasticNet

elastic = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic.fit(X_train, y_train)
```

[27]:
```
ElasticNet(alpha=0.1)
```

[28]:
```
y_pred = elastic.predict(X_test)
```

[29]:
```
print(mean_squared_error(y_test, y_pred))
```

```
10.830856032105528
```

[30]:
```
print(r2_score(y_test, y_pred))
```

```
0.825905553980809
```

[ ]:
```

```

day7-2

February 27, 2024

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PowerTransformer
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
df = pd.read_csv('train .csv')
```

[4]:
```
df.head().T
```

[4]:
```
                                      0            1                  2  \
employee_id                       65438        65141               7513
department            Sales & Marketing   Operations  Sales & Marketing
region                         region_7    region_22          region_19
education             Master's & above   Bachelor's         Bachelor's
gender                                f            m                  m
recruitment_channel            sourcing        other           sourcing
no_of_trainings                       1            1                  1
age                                  35           30                 34
previous_year_rating                5.0          5.0                3.0
length_of_service                     8            4                  7
KPIs_met >80%                         1            0                  0
awards_won?                           0            0                  0
avg_training_score                   49           60                 50
is_promoted                           0            0                  0

                                      3            4
employee_id                        2542        48945
department            Sales & Marketing   Technology
region                        region_23    region_26
education                    Bachelor's   Bachelor's
gender                                m            m
```

```
recruitment_channel                  other        other
no_of_trainings                          2            1
age                                     39           45
previous_year_rating                   1.0          3.0
length_of_service                       10            2
KPIs_met >80%                            0            0
awards_won?                              0            0
avg_training_score                      50           73
is_promoted                              0            0
```

[5]:
```
df.dtypes
```

[5]:
```
employee_id               int64
department               object
region                   object
education                object
gender                   object
recruitment_channel      object
no_of_trainings           int64
age                       int64
previous_year_rating    float64
length_of_service         int64
KPIs_met >80%             int64
awards_won?               int64
avg_training_score        int64
is_promoted               int64
dtype: object
```

[6]:
```
obj_df = df.select_dtypes(include=['object']).copy()
num_df = df.select_dtypes(include=['int64']).copy()
```

[7]:
```
obj_df.head().T
```

[7]:
```
                                     0           1                  2  \
department           Sales & Marketing  Operations  Sales & Marketing
region                        region_7   region_22          region_19
education            Master's & above  Bachelor's         Bachelor's
gender                               f           m                  m
recruitment_channel           sourcing       other           sourcing

                                     3           4
department           Sales & Marketing  Technology
region                       region_23   region_26
education                   Bachelor's  Bachelor's
gender                               m           m
recruitment_channel              other       other
```

[8]:
```
num_df.head().T
```

[8]:

	0	1	2	3	4
employee_id	65438	65141	7513	2542	48945
no_of_trainings	1	1	1	2	1
age	35	30	34	39	45
length_of_service	8	4	7	10	2
KPIs_met >80%	1	0	0	0	0
awards_won?	0	0	0	0	0
avg_training_score	49	60	50	50	73
is_promoted	0	0	0	0	0

[9]:
```
obj_df.isnull().sum()
```

[9]:
```
department               0
region                   0
education             2409
gender                   0
recruitment_channel      0
dtype: int64
```

[10]:
```
num_df.isnull().sum()
```

[10]:
```
employee_id           0
no_of_trainings       0
age                   0
length_of_service     0
KPIs_met >80%         0
awards_won?           0
avg_training_score    0
is_promoted           0
dtype: int64
```

[11]:
```
#IQR
Q1 = num_df.quantile(0.25)
Q3 = num_df.quantile(0.75)

IQR = Q3 - Q1
print(IQR)
```

```
employee_id           39060.75
no_of_trainings           0.00
age                      10.00
length_of_service         4.00
KPIs_met >80%             1.00
awards_won?               0.00
avg_training_score       25.00
```

```
is_promoted              0.00
dtype: float64
```

[12]:
```
#Outliers
num_df_out = num_df[~((num_df < (Q1 - 1.5 * IQR)) |(num_df > (Q3 + 1.5 * IQR))).
 ↪any(axis=1)]
```

[13]:
```
num_df_out.shape
```

[13]:
```
(36477, 8)
```

[14]:
```
#label encoding
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
label_df = obj_df.apply(labelencoder.fit_transform)
```

[15]:
```
label_df.head().T
```

[15]:

	0	1	2	3	4
department	7	4	7	7	8
region	31	14	10	15	18
education	2	0	0	0	0
gender	0	1	1	1	1
recruitment_channel	2	0	2	0	0

[16]:
```
new_df = pd.concat([label_df, num_df], axis=1)
```

[17]:
```
new_df.isna().sum()
```

[17]:
```
department            0
region                0
education             0
gender                0
recruitment_channel   0
employee_id           0
no_of_trainings       0
age                   0
length_of_service     0
KPIs_met >80%         0
awards_won?           0
avg_training_score    0
is_promoted           0
dtype: int64
```

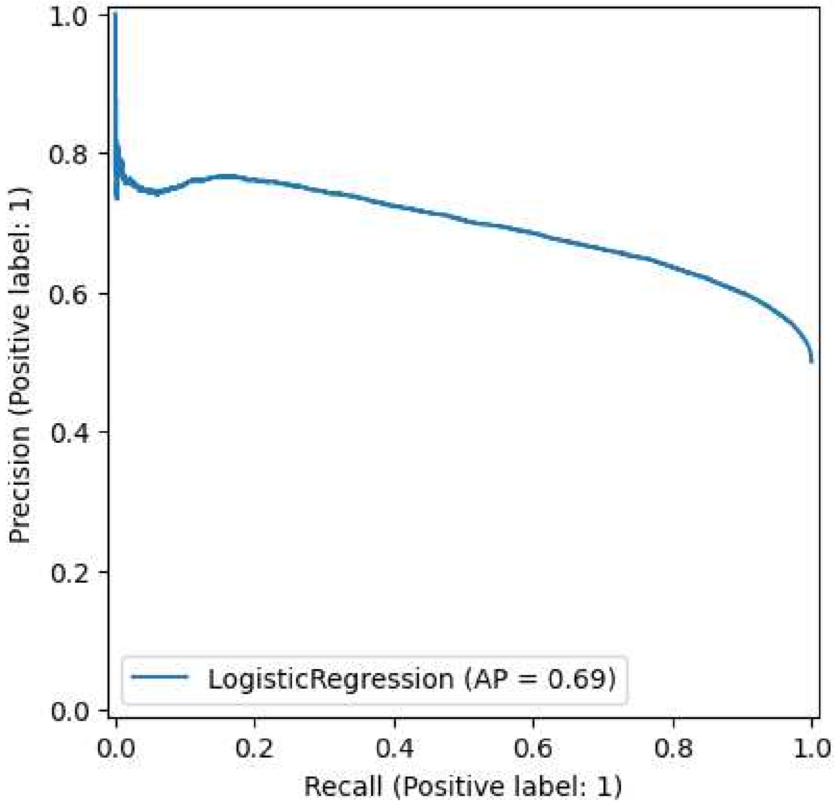
[18]:
```
new_df.columns
```

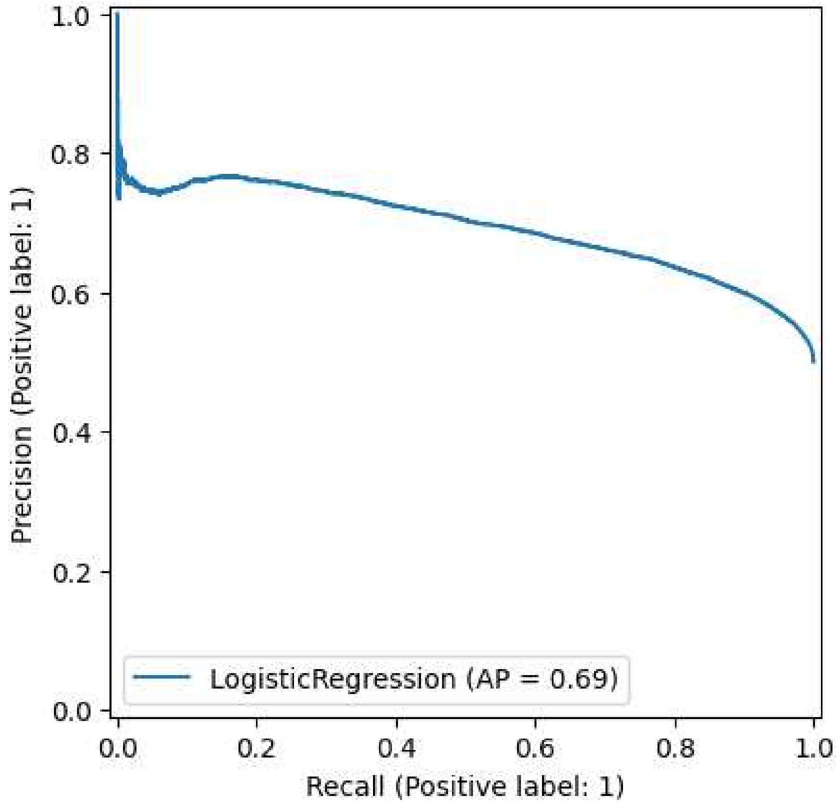
[18]: Index(['department', 'region', 'education', 'gender', 'recruitment_channel',
        'employee_id', 'no_of_trainings', 'age', 'length_of_service',
        'KPIs_met >80%', 'awards_won?', 'avg_training_score', 'is_promoted'],
       dtype='object')

[19]:
```
X = new_df.drop(['is_promoted'], axis=1)
y = new_df['is_promoted']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
sm = SMOTE(random_state=42)
X_train, y_train = sm.fit_resample(X_train, y_train)
```

[20]:
```
#ROS
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import PrecisionRecallDisplay
from sklearn import linear_model

oversampler = RandomOverSampler(sampling_strategy='minority')
X_over, y_over = oversampler.fit_resample(X_train, y_train)

lr = linear_model.LogisticRegression()
lr.fit(X_over, y_over)
y_pred = lr.predict(X_test)
pr = PrecisionRecallDisplay.from_estimator(lr, X_over, y_over)
pr.plot()
```

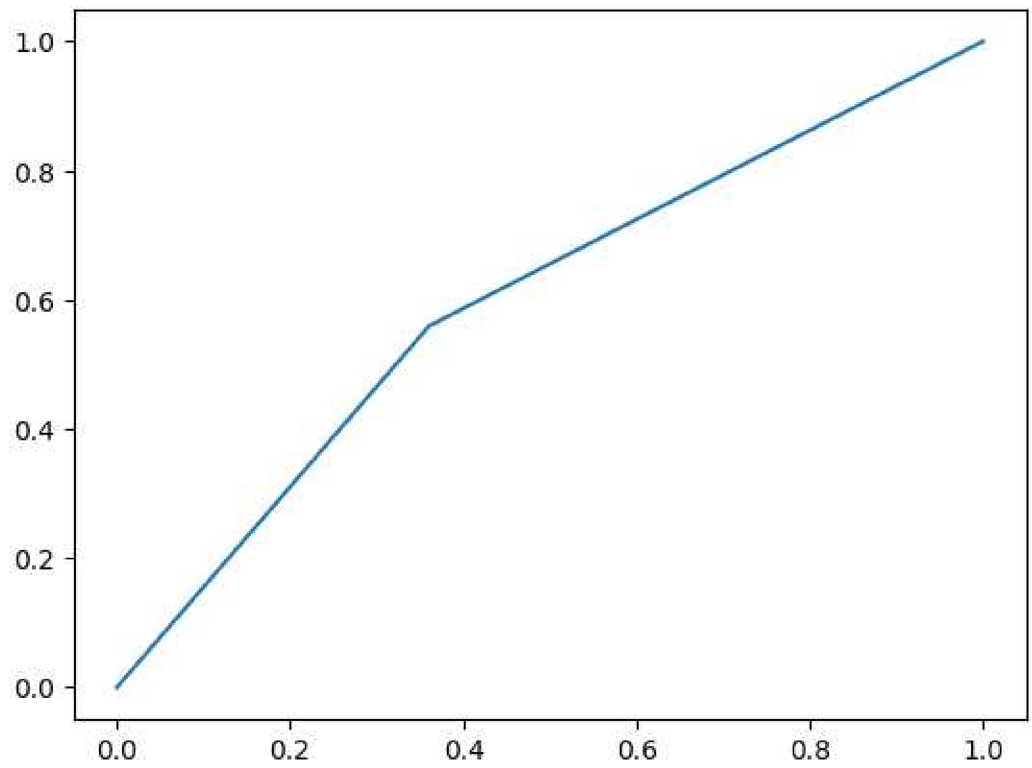
[20]: <sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at
0x76d9650cb410>


1.0
0.8
0.6
0.4
0.2
0.0
Precision (Positive label: 1)
LogisticRegression (AP = 0.69)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)



[21]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score,␣
↪roc_auc_score

print(mean_absolute_error(y_test, y_pred))
print(mean_squared_error(y_test, y_pred))
print(r2_score(y_test, y_pred))
print(roc_auc_score(y_test, y_pred))

#Draw ROC curve
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr, tpr)
plt.show()
```

```
0.36717752234993617
0.36717752234993617
-3.833157557121062
0.5994591759564432
```



[22]:

```
print(y_over.value_counts())
```

```
is_promoted
0    40086
1    40086
Name: count, dtype: int64
```

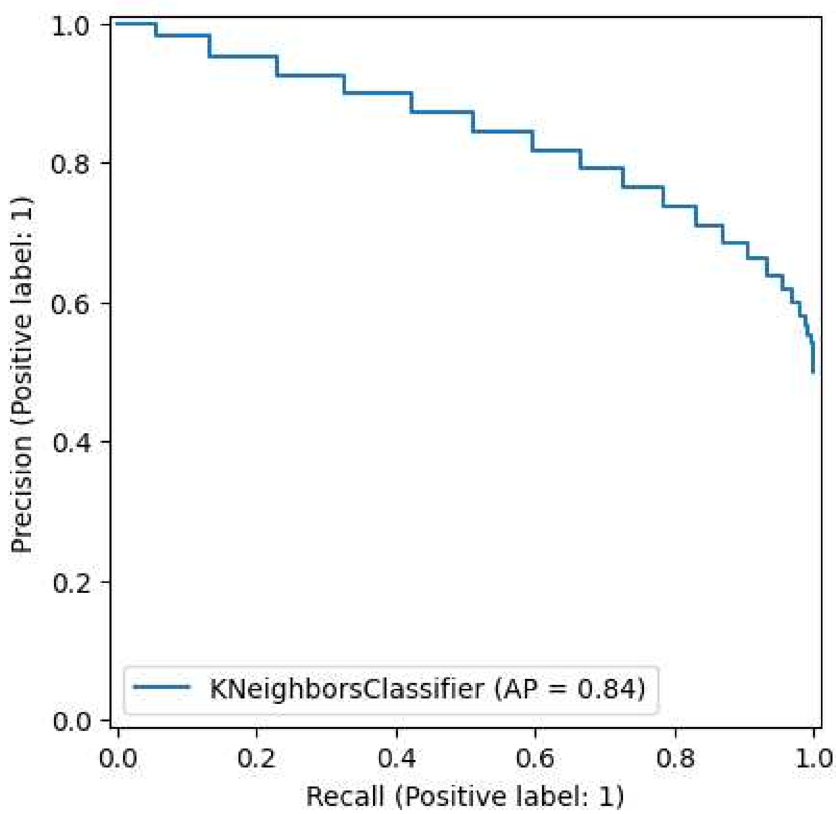
[23]:

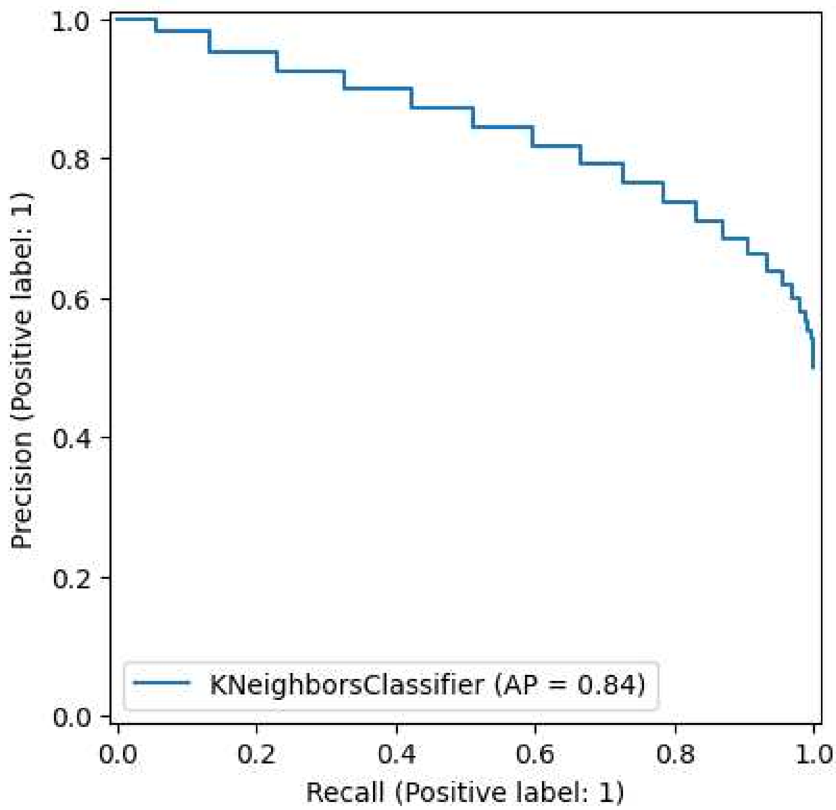
```
#find the best model
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix

knn = KNeighborsClassifier()
param_grid = {'n_neighbors': np.arange(1, 25)}
knn_gscv = GridSearchCV(knn, param_grid, cv=5)
knn_gscv.fit(X_over, y_over)
knn_gscv.best_params_

knn_gscv = KNeighborsClassifier(n_neighbors=24)
knn_gscv.fit(X_over, y_over)
y_pred = knn_gscv.predict(X_test)
print(confusion_matrix(y_test, y_pred))
```

```python
print(classification_report(y_test, y_pred))
pr = PrecisionRecallDisplay.from_estimator(knn_gscv, X_over, y_over)
pr.plot()
```

```
[[6144 3910]
 [ 500  408]]
              precision    recall  f1-score   support

           0       0.92      0.61      0.74     10054
           1       0.09      0.45      0.16       908

    accuracy                           0.60     10962
   macro avg       0.51      0.53      0.45     10962
weighted avg       0.86      0.60      0.69     10962
```

[23]: <sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x76d964b34cd0>





1.0
0.8
0.6
0.4
0.2
0.0
Precision (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)
KNeighborsClassifier (AP = 0.84)

[ ]: