wine data analysis useing logistic regression

June 30, 2023

1 using machine learning

2 Problem Statement:

You work in XYZ Company as a Python developer. The company officials want you to build a data science model. Tasks To Be Performed: 1. Using sklearn import the wine dataset 2. Split the data into train and test set 3. Train the model 4. Make Predictions 5. Check the performance of the model using r2_score

[ ]:

[1]:
```
## import the required library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
## import your dataset
data=pd.read_csv(r'C:/Users/Vikas/Downloads/wine.csv')
```

[3]:
```
data
```

[3]:
```
     Wine  Alcohol  Malic.acid   Ash   Acl   Mg  Phenols  Flavanoids  \
0       1    14.23        1.71  2.43  15.6  127     2.80        3.06
1       1    13.20        1.78  2.14  11.2  100     2.65        2.76
2       1    13.16        2.36  2.67  18.6  101     2.80        3.24
3       1    14.37        1.95  2.50  16.8  113     3.85        3.49
4       1    13.24        2.59  2.87  21.0  118     2.80        2.69
..    ...      ...         ...   ...   ...  ...      ...         ...
173     3    13.71        5.65  2.45  20.5   95     1.68        0.61
174     3    13.40        3.91  2.48  23.0  102     1.80        0.75
175     3    13.27        4.28  2.26  20.0  120     1.59        0.69
176     3    13.17        2.59  2.37  20.0  120     1.65        0.68
177     3    14.13        4.10  2.74  24.5   96     2.05        0.76

     Nonflavanoid.phenols  Proanth  Color.int   Hue    OD  Proline
0                    0.28     2.29       5.64  1.04  3.92     1065
1                    0.26     1.28       4.38  1.05  3.40     1050
```

```
2                    0.30     2.81        5.68  1.03  3.17     1185
3                    0.24     2.18        7.80  0.86  3.45     1480
4                    0.39     1.82        4.32  1.04  2.93      735
..                    …        …           …     …     …        …
173                  0.52     1.06        7.70  0.64  1.74      740
174                  0.43     1.41        7.30  0.70  1.56      750
175                  0.43     1.35       10.20  0.59  1.56      835
176                  0.53     1.46        9.30  0.60  1.62      840
177                  0.56     1.35        9.20  0.61  1.60      560

[178 rows x 14 columns]
```

[5]:
```
## i want to see the first five rows
data.head()
```

[5]:

	Wine	Alcohol	Malic.acid	Ash	Acl	Mg	Phenols	Flavanoids	\
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	

	Nonflavanoid.phenols	Proanth	Color.int	Hue	OD	Proline
0	0.28	2.29	5.64	1.04	3.92	1065
1	0.26	1.28	4.38	1.05	3.40	1050
2	0.30	2.81	5.68	1.03	3.17	1185
3	0.24	2.18	7.80	0.86	3.45	1480
4	0.39	1.82	4.32	1.04	2.93	735

[4]:
```
## i want to see the all col
data.columns
```

[4]:
```
Index(['Wine', 'Alcohol', 'Malic.acid', 'Ash', 'Acl', 'Mg', 'Phenols',
       'Flavanoids', 'Nonflavanoid.phenols', 'Proanth', 'Color.int', 'Hue',
       'OD', 'Proline'],
      dtype='object')
```

[6]:
```
## i want to see the shape of data
data.shape
```

[6]:
```
(178, 14)
```

[7]:
```
## i want to see all information of my dataset.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
```

```
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Wine                  178 non-null    int64
 1   Alcohol               178 non-null    float64
 2   Malic.acid            178 non-null    float64
 3   Ash                   178 non-null    float64
 4   Acl                   178 non-null    float64
 5   Mg                    178 non-null    int64
 6   Phenols               178 non-null    float64
 7   Flavanoids            178 non-null    float64
 8   Nonflavanoid.phenols  178 non-null    float64
 9   Proanth               178 non-null    float64
 10  Color.int             178 non-null    float64
 11  Hue                   178 non-null    float64
 12  OD                    178 non-null    float64
 13  Proline               178 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

[10]:
```
## i want to see the count,min max,std,
data.describe().T
```

[10]:

	count	mean	std	min	25%	\
Wine	178.0	1.938202	0.775035	1.00	1.0000	
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	
Acl	178.0	19.494944	3.339564	10.60	17.2000	
Mg	178.0	99.741573	14.282484	70.00	88.0000	
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	
Proanth	178.0	1.590899	0.572359	0.41	1.2500	
Color.int	178.0	5.058090	2.318286	1.28	3.2200	
Hue	178.0	0.957449	0.228572	0.48	0.7825	
OD	178.0	2.611685	0.709990	1.27	1.9375	
Proline	178.0	746.893258	314.907474	278.00	500.5000	

	50%	75%	max
Wine	2.000	3.0000	3.00
Alcohol	13.050	13.6775	14.83
Malic.acid	1.865	3.0825	5.80
Ash	2.360	2.5575	3.23
Acl	19.500	21.5000	30.00
Mg	98.000	107.0000	162.00
Phenols	2.355	2.8000	3.88
Flavanoids	2.135	2.8750	5.08

```
Nonflavanoid.phenols    0.340    0.4375     0.66
Proanth                 1.555    1.9500     3.58
Color.int               4.690    6.2000    13.00
Hue                     0.965    1.1200     1.71
OD                      2.780    3.1700     4.00
Proline               673.500  985.0000  1680.00
```

[8]:
```
##  i want to see the how much null values in my dataset
data.isnull().sum()
```

[8]:
```
Wine                  0
Alcohol               0
Malic.acid            0
Ash                   0
Acl                   0
Mg                    0
Phenols               0
Flavanoids            0
Nonflavanoid.phenols  0
Proanth               0
Color.int             0
Hue                   0
OD                    0
Proline               0
dtype: int64
```

[80]:
```

```

[ ]:
```

```

[81]:
```
## here we divide the dataset into independent(x) & dependent(y).
x=data.iloc[:,1:14].values
y=data.iloc[:,0].values
```

[45]:
```
x
```

[45]:
```
array([[1.423e+01, 1.710e+00, 2.430e+00, …, 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, …, 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, …, 1.030e+00, 3.170e+00,
        1.185e+03],
       …,
       [1.327e+01, 4.280e+00, 2.260e+00, …, 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, …, 6.000e-01, 1.620e+00,
        8.400e+02],
```

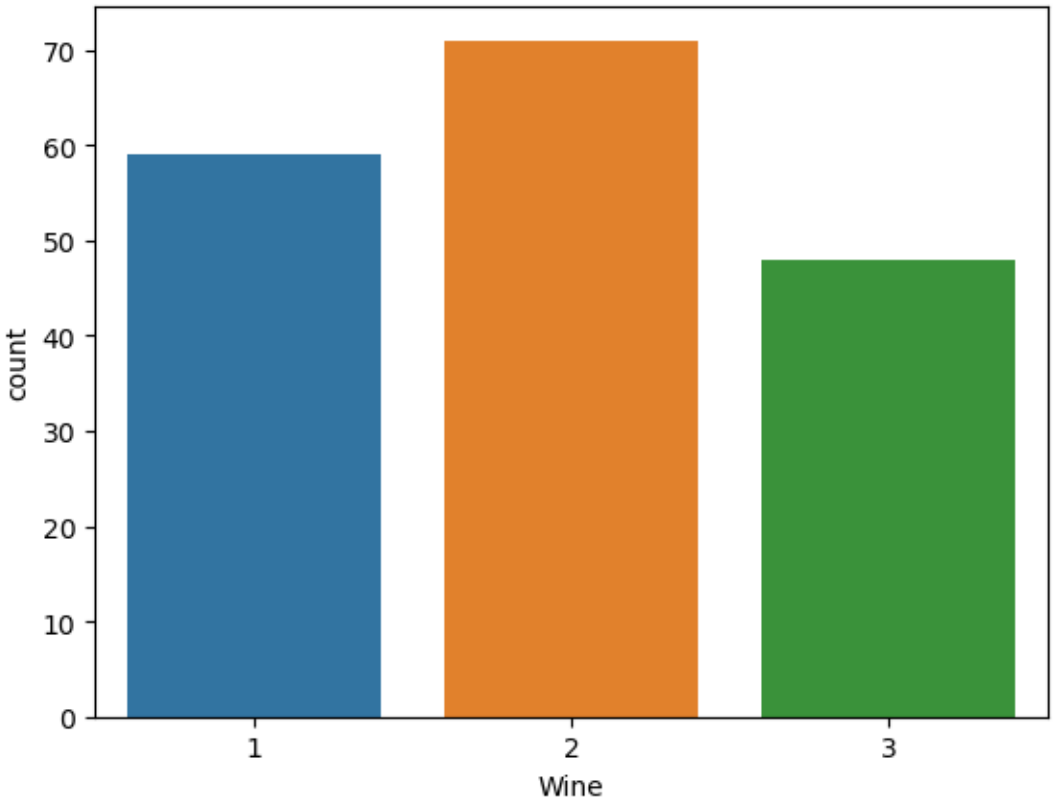
```
       [1.413e+01, 4.100e+00, 2.740e+00, …, 6.100e-01, 1.600e+00,
        5.600e+02]])
```

[46]:
```
y
```

[46]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3], dtype=int64)
```

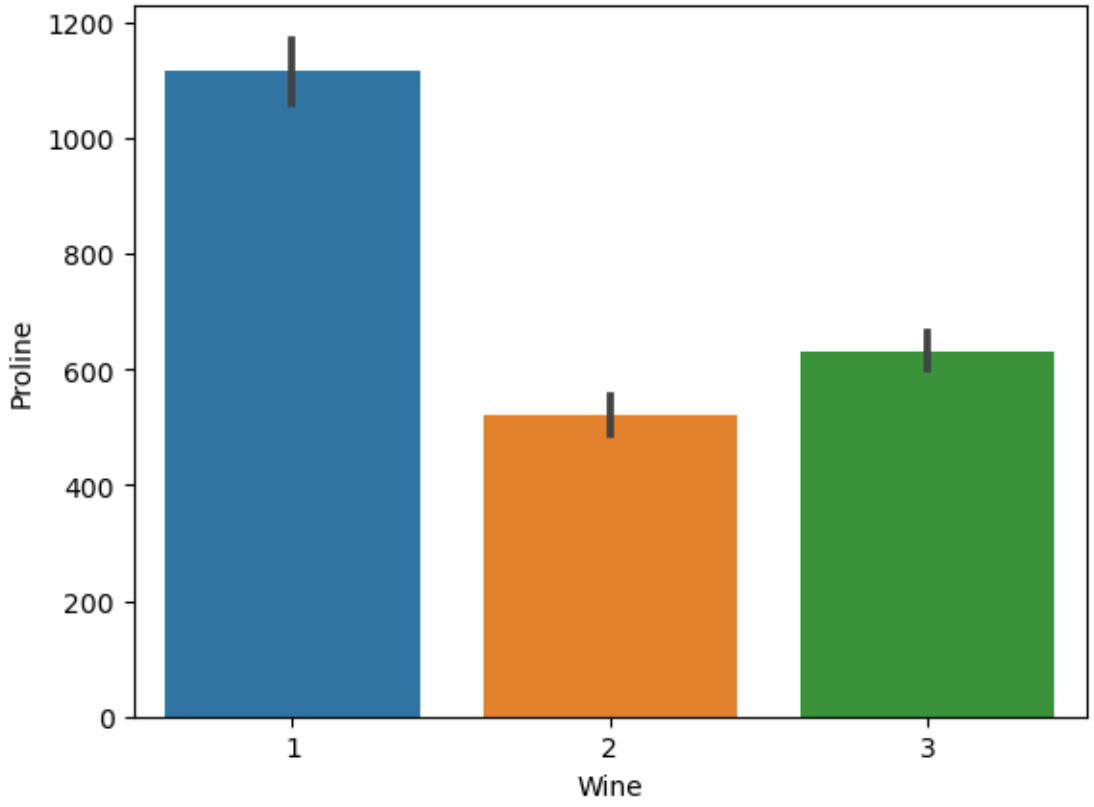
[48]:
```
## we can visualize the data,
sns.countplot(x='Wine',data=data)
```

[48]:
```
<Axes: xlabel='Wine', ylabel='count'>
```



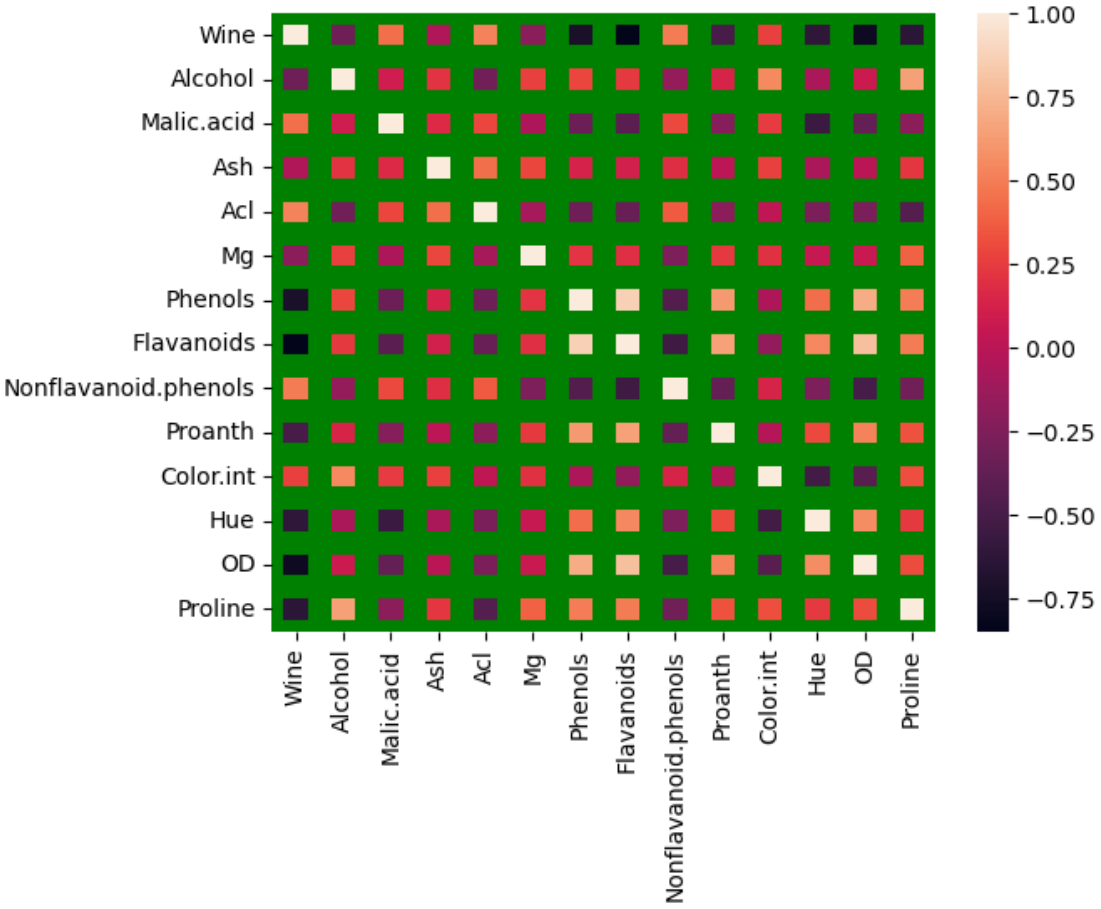
[50]:
```
## another way we can visualize the data
sns.barplot(x='Wine',y='Proline',data=data)
```

[50]: <Axes: xlabel='Wine', ylabel='Proline'>



[61]:
```
## another way we can visualize the data
sns.heatmap(data.corr(),linewidth=10,linecolor='g')
```

[61]: <Axes: >



3 what i mentioned above visualization is the 100% correlation

[82]:
```
##split the data into traning set and testing set,

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
x, y, test_size=0.33, random_state=42)
```

[83]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

[84]:
```
## fitting the LogisticRegression into training  set,
from sklearn.linear_model import LogisticRegression
logr=LogisticRegression(random_state=42)
logr.fit(x_train,y_train)
```

[84]:
```
LogisticRegression(random_state=42)
```

[66]:
```
## predicting the testing set,
y_pred=logr.predict(x_test)
y_pred
```

[66]:
```
array([1, 1, 3, 1, 2, 1, 2, 3, 2, 3, 1, 3, 1, 2, 1, 2, 2, 2, 1, 2, 1, 2,
       2, 3, 3, 3, 2, 2, 2, 1, 1, 2, 3, 1, 1, 1, 3, 3, 2, 3, 1, 2, 2, 3,
       3, 1, 2, 2, 3, 1, 2, 1, 1, 3, 3, 2, 2, 1, 2], dtype=int64)
```

[67]:
```
## i want to see the score,
logr.score(x_test,y_test)
```

[67]:
```
0.9830508474576272
```

[85]:
```

```

[ ]:
```

```