ML 13 - EDA With Red Wine Data By Virat Tiwari

November 25, 2023

1 EDA With Red Wine Dataset By Virat Tiwari

Dataset Information

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones). Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, we are not sure if all input variables are relevant. So it could be interesting to test feature selection methods.

Attribute Information

Input variables (based on physicochemical tests):

1 - fixed acidity,

2 - volatile acidity,

3 - citric acid,

4 - residual sugar,

5 - chlorides,

6 - free sulfur dioxide,

7 - total sulfur dioxide,

8 - density,

9 - pH,

10 - sulphates,

11 - alcohol

Output variable (based on sensory data):

12 - quality (score between 0 and 10)

NOTE - WHENEVER WE GET SUCH TYPE OF DATADET SO , AS A DATA SCIENTIST OUR MAIN AIM TO UNDERSTAND THE DATA THAT WHAT DATA IS TRY TO CONVEY THE MESSAGE AND WE ANALYSIS THE DATA SHARPELY AND UNDERSTAND EACH PARAMETER OF DATASET

[3]:
```
import pandas as pd
```

[4]:
```
df=pd.read_csv("winequality-red.csv")
df.head()
```

[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides
0	7.4	0.70	0.00	1.9	0.076
1	7.8	0.88	0.00	2.6	0.098
2	7.8	0.76	0.04	2.3	0.092
3	11.2	0.28	0.56	1.9	0.075
4	7.4	0.70	0.00	1.9	0.076

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	11.0	34.0	0.9978	3.51	0.56
1	25.0	67.0	0.9968	3.20	0.68
2	15.0	54.0	0.9970	3.26	0.65
3	17.0	60.0	0.9980	3.16	0.58
4	11.0	34.0	0.9978	3.51	0.56

	alcohol	quality
0	9.4	5
1	9.8	5
2	9.8	5
3	9.8	6
4	9.4	5

SUMMARY OF THE DATASET -

[6]:
```
# FOR CHECKING THE SUMMARY OF THE DATASET -
# . info ( ) - WE HAVE TO USE IN BUILT FUNCTION FOR THE SUMMARY

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
```

```
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

[7]:
```
# Descriptive summary of the datset
# describe ( )  function is used for getting the detailed decription of dataset

df.describe()
```

[7]:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	1599.000000	1599.000000	1599.000000	1599.000000	
mean	8.319637	0.527821	0.270976	2.538806	
std	1.741096	0.179060	0.194801	1.409928	
min	4.600000	0.120000	0.000000	0.900000	
25%	7.100000	0.390000	0.090000	1.900000	
50%	7.900000	0.520000	0.260000	2.200000	
75%	9.200000	0.640000	0.420000	2.600000	
max	15.900000	1.580000	1.000000	15.500000	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
count	1599.000000	1599.000000	1599.000000	1599.000000	
mean	0.087467	15.874922	46.467792	0.996747	
std	0.047065	10.460157	32.895324	0.001887	
min	0.012000	1.000000	6.000000	0.990070	
25%	0.070000	7.000000	22.000000	0.995600	
50%	0.079000	14.000000	38.000000	0.996750	
75%	0.090000	21.000000	62.000000	0.997835	
max	0.611000	72.000000	289.000000	1.003690	

	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000
mean	3.311113	0.658149	10.422983	5.636023
std	0.154386	0.169507	1.065668	0.807569
min	2.740000	0.330000	8.400000	3.000000
25%	3.210000	0.550000	9.500000	5.000000
50%	3.310000	0.620000	10.200000	6.000000
75%	3.400000	0.730000	11.100000	6.000000
max	4.010000	2.000000	14.900000	8.000000

[9]:
```
# Shape ( )  function is used for getting the datapints and columns of dataset
# (1599, 12) - ( datapoints , columns )

df.shape
```

[9]: (1599, 12)

[11]:
```
# List down the columns

df.columns
```

[11]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

[14]:
```
# This is how we check the unique values of quality
# Like quakity of wines are 5, 6, 7, 4, 8, 3 out of 10

df["quality"].unique()
```

[14]:
```
array([5, 6, 7, 4, 8, 3])
```

CHECK DATA IS BALANCED OR IMBALANCED

[15]:
```
# Here we check data is balanced or imbalanced
# We also check that how many datapoints lie in each category of quality outcome
# Like how many data points are lie in 5, 6, 7, 4, 8, 3 category of quality
# Most data points lie in category of 5 quality of wine and least datapoints␣
 ↪lie in 3 category of quality of wine
# OBIOUSLY THIS IS IMBALANCED DATASET DUE TO IRREGULAR DISTRIBUTION OF␣
 ↪DATAPINTS IN EACH CATEGORY OF WINE "QUALITY"

df["quality"].value_counts()
```

[15]:
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

CHECK MISSING VALUES IN DATASET

[16]:
```
# Whenever we dataset our first step to check the missing values in dataset
# we get 0's in front of each columns so in this dataset there is no missing␣
 ↪values

df.isnull().sum()
```

[16]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
```

```
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

CHECK DUPLICATES RECORDS OR VALUES

[17]:
```
# . duplicated ( )  function is used for check the duplicate records
# True shows the DUPLICATE values
# False shows the values that are not DUPLICATE

df.duplicated()
```

[17]:
```
0       False
1       False
2       False
3       False
4        True
        …
1594    False
1595    False
1596     True
1597    False
1598    False
Length: 1599, dtype: bool
```

[18]:
```
# For getting that duplicates records from the dataset
# df[df.duplicated()] is sued for get the duplicate records

df[df.duplicated()]
```

[18]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
4	7.4	0.700	0.00	1.90	0.076	
11	7.5	0.500	0.36	6.10	0.071	
27	7.9	0.430	0.21	1.60	0.106	
40	7.3	0.450	0.36	5.90	0.074	
65	7.2	0.725	0.05	4.65	0.086	
…	…	…	…	…	…	
1563	7.2	0.695	0.13	2.00	0.076	
1564	7.2	0.695	0.13	2.00	0.076	
1567	7.2	0.695	0.13	2.00	0.076	
1581	6.2	0.560	0.09	1.70	0.053	

```
1596            6.3             0.510         0.13             2.30      0.076

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
4                    11.0                  34.0  0.99780  3.51       0.56
11                   17.0                 102.0  0.99780  3.35       0.80
27                   10.0                  37.0  0.99660  3.17       0.91
40                   12.0                  87.0  0.99780  3.33       0.83
65                    4.0                  11.0  0.99620  3.41       0.39
…                     …                     …       …    …          …
1563                 12.0                  20.0  0.99546  3.29       0.54
1564                 12.0                  20.0  0.99546  3.29       0.54
1567                 12.0                  20.0  0.99546  3.29       0.54
1581                 24.0                  32.0  0.99402  3.54       0.60
1596                 29.0                  40.0  0.99574  3.42       0.75

      alcohol  quality
4         9.4        5
11       10.5        5
27        9.5        5
40       10.5        5
65       10.9        5
…         …        …
1563     10.1        5
1564     10.1        5
1567     10.1        5
1581     11.3        5
1596     11.0        6

[240 rows x 12 columns]
```

REMOVE THE DUPLICATE VALUES OR DATAPOINTS FROM THE DATASET -

[19]:
```
# REMOVE the DUPLICATE values
# drop_duplicates(inplace=True) function used for remove the duplicate values

df.drop_duplicates(inplace=True)
```

[20]:
```
# shape ( ) function is used for getting the records or datapoints with No of␣
 ↪Columns
# (1359, 12) - ( datapoints , columns )

df.shape
```

[20]:
```
(1359, 12)
```

TYPES OF ANALYSIS

[21]:
```
# Check the correlation ( cor ) between the features
# . corr ( ) function is used for finding the correlation

df.corr()
```

[21]:

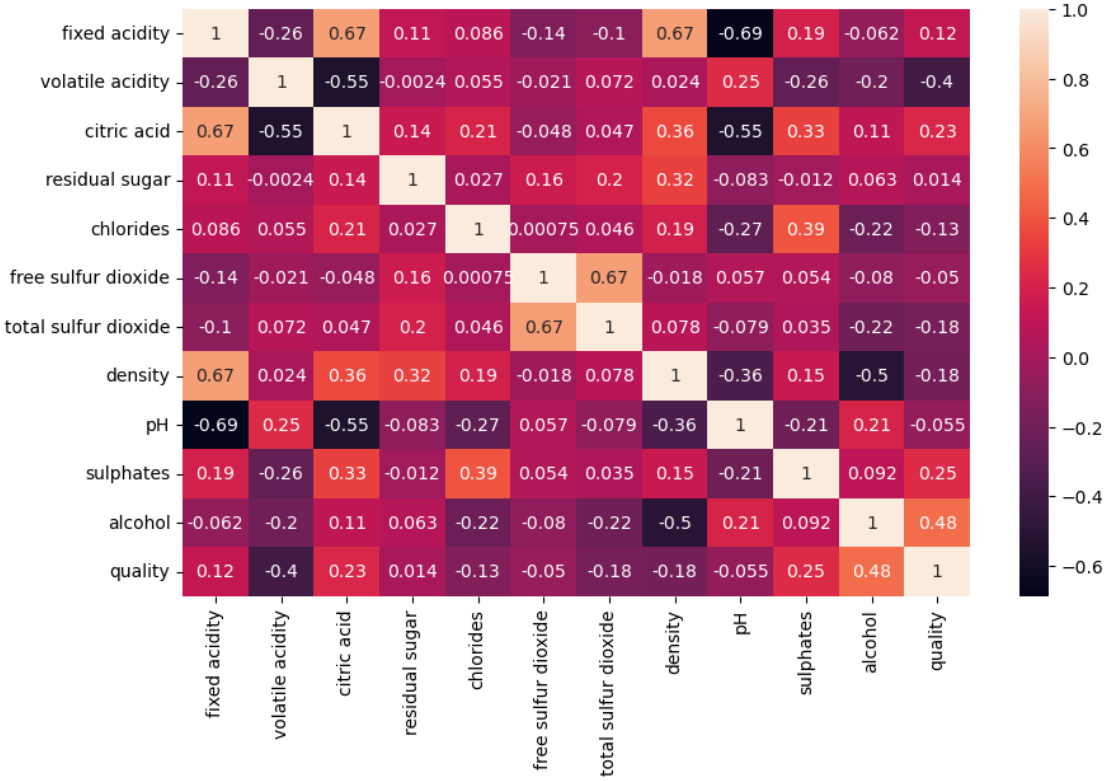
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.255124	0.667437	0.111025	0.085886	-0.140580	-0.103777	0.670195	-0.686685	0.190269	-0.061596	0.119024
volatile acidity	-0.255124	1.000000	-0.551248	-0.002449	0.055154	-0.020945	0.071701	0.023943	0.247111	-0.256948	-0.197812	-0.395214
citric acid	0.667437	-0.551248	1.000000	0.143892	0.210195	-0.048004	0.047358	0.357962	-0.550310	0.326062	0.105108	0.228057
residual sugar	0.111025	-0.002449	0.143892	1.000000	0.026656	0.160527	0.201038	0.324522	-0.083143	-0.011837	0.063281	0.013640
chlorides	0.085886	0.055154	0.210195	0.026656	1.000000	0.000749	0.045773	0.193592	-0.270893	0.394557	-0.223824	-0.130988
free sulfur dioxide	-0.140580	-0.020945	-0.048004	0.160527	0.000749	1.000000	0.667246	-0.018071	0.056631	0.054126	-0.080125	-0.050463
total sulfur dioxide	-0.103777	0.071701	0.047358	0.201038	0.045773	0.667246	1.000000	0.078141	-0.079257	0.035291	-0.217829	-0.177855
density	0.670195	0.023943	0.357962	0.324522	0.193592	-0.018071	0.078141	1.000000	-0.355617	0.146036	-0.504995	-0.184252
pH	-0.686685	0.247111	-0.550310	-0.083143	-0.270893	0.056631	-0.079257	-0.355617	1.000000	-0.214134	0.213418	-0.055245
sulphates	0.190269	-0.256948	0.326062	-0.011837	0.394557	0.054126	0.035291	0.146036	-0.214134	1.000000	0.091621	0.248835
alcohol	-0.061596	-0.197812	0.105108	0.063281	-0.223824	-0.080125	-0.217829	-0.504995	0.213418	0.091621	1.000000	0.476166
quality	0.119024	-0.395214	0.228057	0.013640	-0.130988	-0.050463	-0.177855	-0.184252	-0.055245	0.248835	0.476166	1.000000

```
                       alcohol   quality
fixed acidity        -0.061596  0.119024
volatile acidity     -0.197812 -0.395214
citric acid           0.105108  0.228057
residual sugar        0.063281  0.013640
chlorides            -0.223824 -0.130988
free sulfur dioxide  -0.080125 -0.050463
total sulfur dioxide -0.217829 -0.177855
density              -0.504995 -0.184252
pH                    0.213418 -0.055245
sulphates             0.091621  0.248835
alcohol               1.000000  0.480343
quality               0.480343  1.000000
```

Data Analysis Through The Visualization -

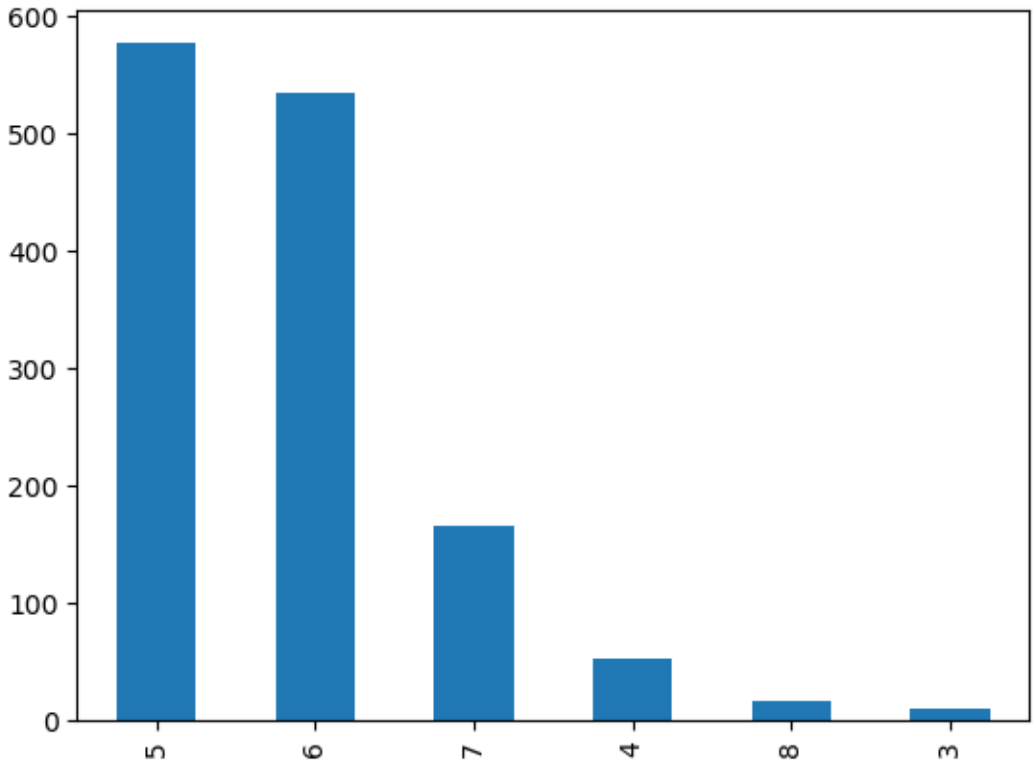
[31]:
```
# First of all we import the VISUALIZING LIBRARIES - matplotlib.pyplot and
 ↪seaborn

import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True)
```

[31]: <AxesSubplot: >



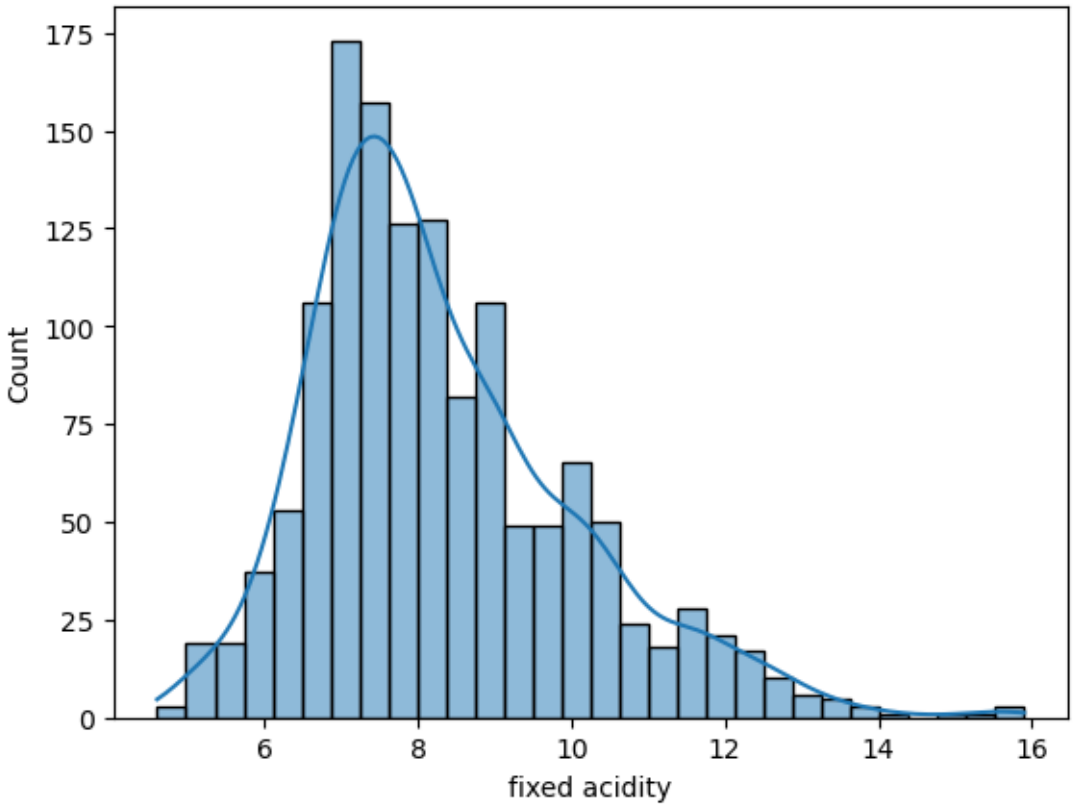
[33]:
```
# Easiest way to draw the chart
# .quality.value_counts().plot(kind="bar") - Syntax

df.quality.value_counts().plot(kind="bar")
```

[33]: <AxesSubplot: >

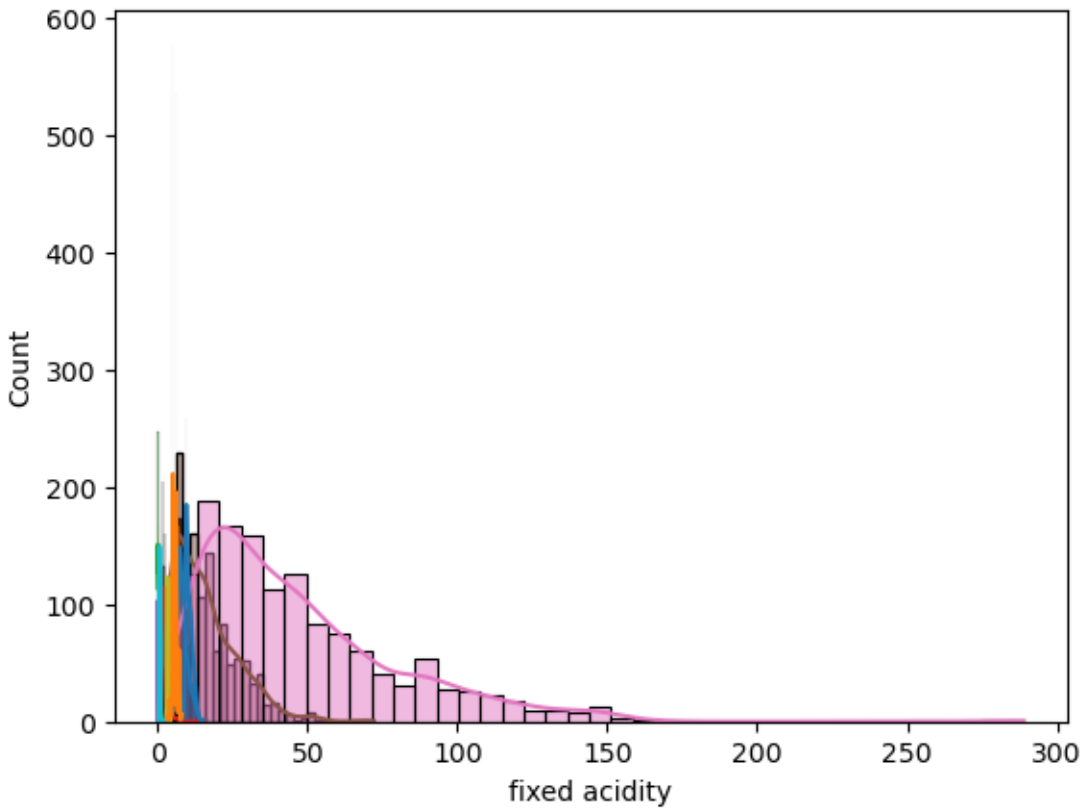


[36]:
```
# This histplot shows the "fixed acidity" features 's  values
# sns.histplot(df[" "],kde=True) - Synatx
# Kde - It is used for Draw the CURVE LINE in plot

sns.histplot(df["fixed acidity"],kde=True)
```

[36]: <AxesSubplot: xlabel='fixed acidity', ylabel='Count'>



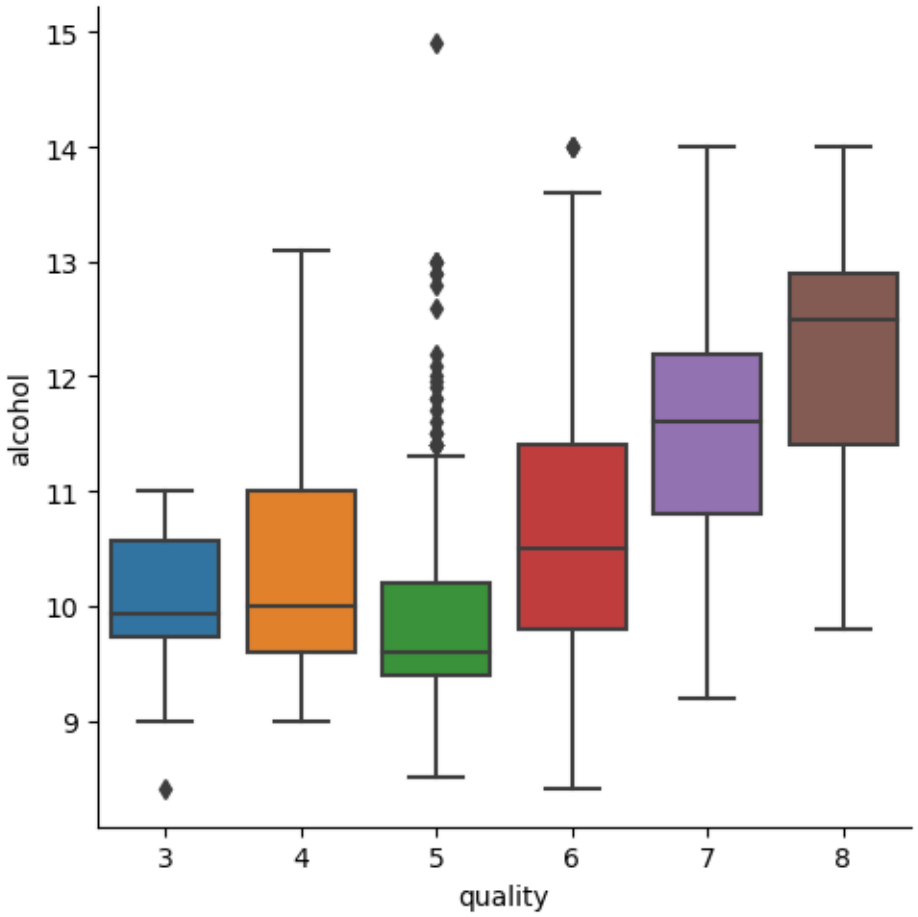
[38]:
```
# Additional way to draw the graph just for presenting in a beautiful manner

for i in df.columns:
    sns.histplot(df[i],kde=True)
```



CATEGORICAL PLOT

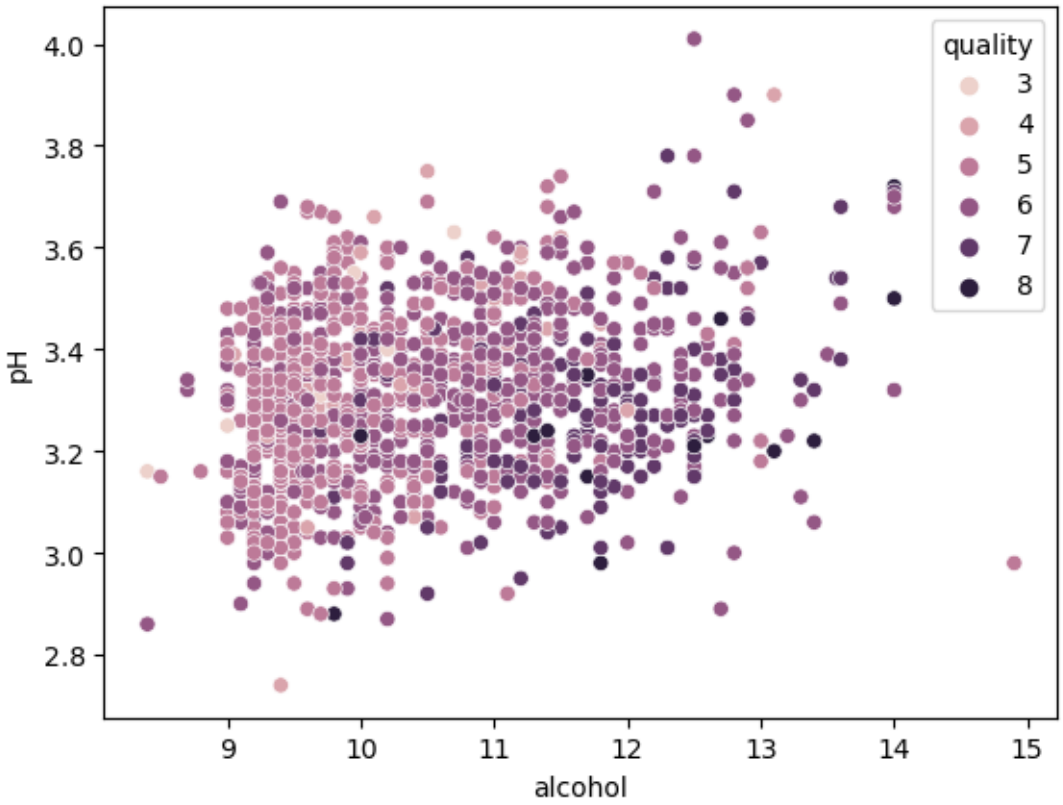
[41]:
```
# CATEGORICAL PLOT
# This plot shows the OUTLIERS THROUGH THE CHART
# sns.catplot(x=" ",y=" ",data=df,kind="box") - This is syntax for draw the␣
 ↪chart

sns.catplot(x="quality",y="alcohol",data=df,kind="box")
```

[41]:
```
<seaborn.axisgrid.FacetGrid at 0x7f7ab9e8eaa0>
```



[45]:
```
# Here we use the catter plot for understand the relation between alcohol and␣
↪quality
# sns.scatterplot(x=" ",y=" ",hue=" ", data=df) - This is our syntax

sns.scatterplot(x="alcohol",y="pH",hue="quality",data=df)
```

[45]: <AxesSubplot: xlabel='alcohol', ylabel='pH'>



AT THE END OF THE DAY WE ANALYSIS THE ENTIRE DATASET FROM DIFFERENT PARAMETRS AND DRAW THE CONCLUSION WITH THE HELP OF GRAPHS AND CHARTS . IN VERY SIMPLE TERMS WE JUST TRY TO UNDERSTAND WHAT DATA GIVES THE INFORMATION AND UNDERSTAND THE RELATIONS BETWEEN THE FEATURES THAT HELP IN DECISON MAKING PROCESS .

THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)