milk-quality-prediction

October 30, 2023

[9]:
```
import pandas as pd #ifor data processing
import numpy as np #for mathematical operation and Linear algebra
import matplotlib.pyplot as plt
import seaborn as sns #for advance data visualization Library
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

1 Load Dataset

[12]:
```
df = pd.read_csv(r'E:\Projects File\Milk Quality Prediction\archive_6\milknew.
↪csv')
df.head()
```

[12]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
0	6.6	35	1	0	1	0	254	high
1	6.6	36	0	1	0	1	253	high
2	8.5	70	1	1	1	1	246	low
3	9.5	34	1	1	0	1	255	low
4	6.6	37	0	0	0	0	255	medium

[13]:
```
df.tail()
```

[13]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
1054	6.7	45	1	1	0	0	247	medium
1055	6.7	38	1	0	1	0	255	high
1056	3.0	40	1	1	1	1	255	low
1057	6.8	43	1	0	1	0	250	high
1058	8.6	55	0	1	1	1	255	low

[15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1059 entries, 0 to 1058
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
```

```
 0   pH           1059 non-null   float64
 1   Temprature   1059 non-null   int64
 2   Taste        1059 non-null   int64
 3   Odor         1059 non-null   int64
 4   Fat          1059 non-null   int64
 5   Turbidity    1059 non-null   int64
 6   Colour       1059 non-null   int64
 7   Grade        1059 non-null   object
dtypes: float64(1), int64(6), object(1)
memory usage: 66.3+ KB
```

2 Basic Statistic details about the data

[17]:
```
df.describe()
```

[17]:

	pH	Temprature	Taste	Odor	Fat	\
count	1059.000000	1059.000000	1059.000000	1059.000000	1059.000000	
mean	6.630123	44.226629	0.546742	0.432483	0.671388	
std	1.399679	10.098364	0.498046	0.495655	0.469930	
min	3.000000	34.000000	0.000000	0.000000	0.000000	
25%	6.500000	38.000000	0.000000	0.000000	0.000000	
50%	6.700000	41.000000	1.000000	0.000000	1.000000	
75%	6.800000	45.000000	1.000000	1.000000	1.000000	
max	9.500000	90.000000	1.000000	1.000000	1.000000	

	Turbidity	Colour
count	1059.000000	1059.000000
mean	0.491029	251.840415
std	0.500156	4.307424
min	0.000000	240.000000
25%	0.000000	250.000000
50%	0.000000	255.000000
75%	1.000000	255.000000
max	1.000000	255.000000

3 Data cleaning

[18]:
```
df.duplicated().sum()
```

[18]: 976

3.1 Removing the duplicates values

[20]:
```
df.drop_duplicates()
```

[20]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
0	6.6	35	1	0	1	0	254	high
1	6.6	36	0	1	0	1	253	high
2	8.5	70	1	1	1	1	246	low
3	9.5	34	1	1	0	1	255	low
4	6.6	37	0	0	0	0	255	medium
..	…	…	…	…	…	…	…	…
930	6.6	38	0	1	1	1	255	high
942	6.6	45	1	0	0	1	255	medium
957	6.8	41	1	1	1	0	255	high
985	6.5	45	1	0	0	0	246	medium
998	6.6	43	0	0	0	1	250	medium

[83 rows x 8 columns]

3.2 Check for null values in dataset

[25]:
```
df.isnull().sum()
```

[25]:
```
pH            0
Temprature    0
Taste         0
Odor          0
Fat           0
Turbidity     0
Colour        0
Grade         0
dtype: int64
```

[23]:
```
df.nunique()
```

[23]:
```
pH            16
Temprature    17
Taste          2
Odor           2
Fat            2
Turbidity      2
Colour         9
Grade          3
dtype: int64
```

3.3 Value count of different Columns

[37]:
```
df['pH'].value_counts()
```

[37]:
```
pH
6.8    249
```

```
6.5    189
6.6    159
6.7     82
3.0     70
9.0     61
8.6     40
7.4     39
4.5     37
9.5     24
8.1     24
5.5     23
8.5     22
4.7     20
5.6     19
6.4      1
Name: count, dtype: int64
```

[38]:
```
df['Temprature'].value_counts()
```

[38]:
```
Temprature
45    219
38    179
40    132
37     83
43     77
36     66
50     58
55     48
34     40
41     30
66     24
35     23
70     22
65     22
60     18
90     17
42      1
Name: count, dtype: int64
```

[39]:
```
df['Taste'].value_counts()
```

[39]:
```
Taste
1    579
0    480
Name: count, dtype: int64
```

[40]:
```
df['Odor'].value_counts()
```

[40]:
```
Odor
0    601
1    458
Name: count, dtype: int64
```

[43]:
```
df['Colour'].value_counts()
```

[43]:
```
Colour
255    628
250    146
245    115
247     48
246     44
240     32
248     23
253     22
254      1
Name: count, dtype: int64
```

[44]:
```
df['Grade'].value_counts()
```

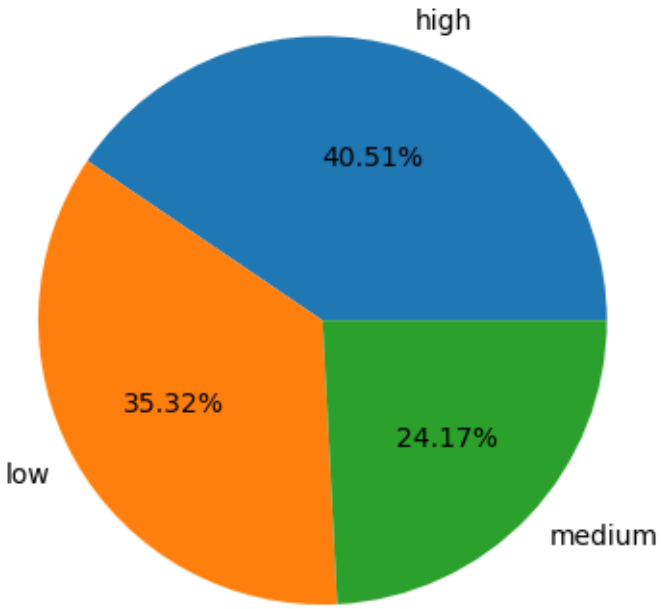
[44]:
```
Grade
low       429
medium    374
high      256
Name: count, dtype: int64
```

[45]:
```
df['Turbidity'].value_counts()
```

[45]:
```
Turbidity
0    539
1    520
Name: count, dtype: int64
```

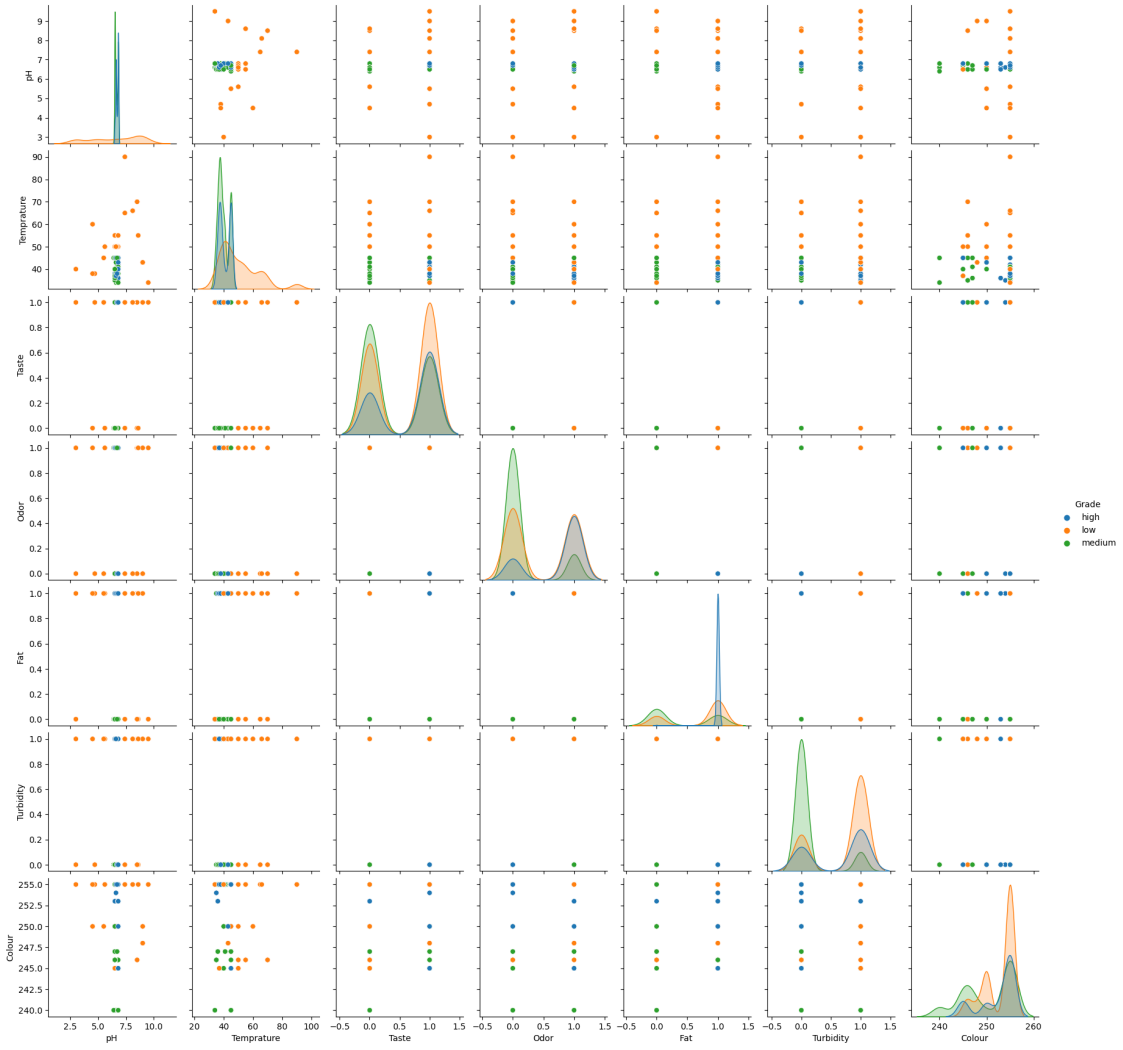
4 Data Visualization

[65]:
```
plt.pie(df['Grade'].value_counts(),autopct='%1.2f%%',labels=np.
  ↪unique(df['Grade']))
plt.show()
```



[68]:
```
sns.pairplot(df,hue='Grade')
```

[68]: <seaborn.axisgrid.PairGrid at 0x19dc0936620>



[73]:
```
plt.figure(figsize=(20,15))

plt.subplot(3,3,1)
sns.barplot(x = 'Grade', y = 'pH', data = df, palette='Reds')

plt.subplot(3,3,2)
sns.barplot(x = 'Grade', y = 'Temperature', data = df,palette ='Wistia')
```

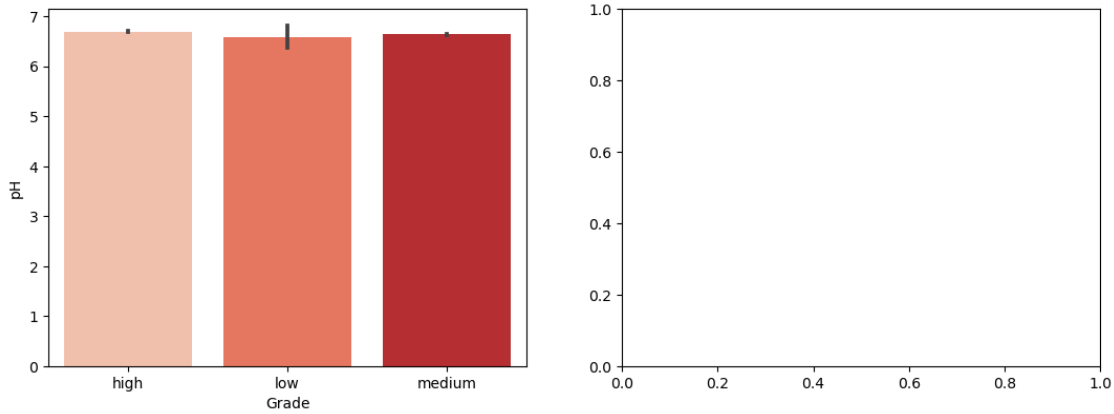
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[73], line 7
      4 sns.barplot(x = 'Grade', y = 'pH', data = df, palette='Reds')
      6 plt.subplot(3,3,2)
----> 7 sns.barplot(x = 'Grade', y = 'Temperature', data = df,palette ='Wistia'
```

```
File E:\PYTHON\Python Install\lib\site-packages\seaborn\categorical.py:2755, in barplot(data, x, y, hue, order, hue_order, estimator, errorbar, n_boot, units, seed, orient, color, palette, saturation, width, errcolor, errwidth, capsize, dodge, ci, ax, **kwargs)
   2752 if estimator is len:
   2753     estimator = "size"
-> 2755 plotter = _BarPlotter(x, y, hue, data, order, hue_order,
   2756                         estimator, errorbar, n_boot, units, seed,
   2757                         orient, color, palette, saturation,
   2758                         width, errcolor, errwidth, capsize, dodge)
   2760 if ax is None:
   2761     ax = plt.gca()

File E:\PYTHON\Python Install\lib\site-packages\seaborn\categorical.py:1530, in _BarPlotter.__init__(self, x, y, hue, data, order, hue_order, estimator, errorbar, n_boot, units, seed, orient, color, palette, saturation, width, errcolor, errwidth, capsize, dodge)
   1525 def __init__(self, x, y, hue, data, order, hue_order,
   1526              estimator, errorbar, n_boot, units, seed,
   1527              orient, color, palette, saturation, width,
   1528              errcolor, errwidth, capsize, dodge):
   1529     """Initialize the plotter."""
-> 1530     self.establish_variables(x, y, hue, data, orient,
   1531                              order, hue_order, units)
   1532     self.establish_colors(color, palette, saturation)
   1533     self.estimate_statistic(estimator, errorbar, n_boot, seed)

File E:\PYTHON\Python Install\lib\site-packages\seaborn\categorical.py:541, in _CategoricalPlotter.establish_variables(self, x, y, hue, data, orient, order, hue_order, units)
    539     if isinstance(var, str):
    540         err = f"Could not interpret input '{var}'"
--> 541         raise ValueError(err)
    543 # Figure out the plotting orientation
    544 orient = infer_orient(
    545     x, y, orient, require_numeric=self.require_numeric
    546 )

ValueError: Could not interpret input 'Temperature'
```



[85]:
```
word = "I love python programming"
word
```

[85]: 'I love python programming'

[86]:
```
word[-18:-12]
```

[86]: 'python'

[87]:
```
word[7:13]
```

[87]: 'python'

[ ]:
```

```