```
import pandas as p
```

```
pro = p.read_excel('/content/wine_Training.xlsx')
pro.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulp
0	7.4	0.70	0.00	1.9	0.076	11.0	34	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67	0.9968	3.20	
2	7.8	0.76	0.04	2.3	0.092	15.0	54	0.9970	3.26	
3	11.2	0.28	0.56	1.9	0.075	17.0	60	0.9980	3.16	
4	7.4	0.70	0.00	1.9	0.076	11.0	34	0.9978	3.51	

```
pro['quality'].unique()
```

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

```
quality = {}
x = 0
for i in pro['quality'].unique():
    quality[i] = x
    x = x + 1
```

```
pro['quality'] = pro['quality'].map(quality)
```

```
pro.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulp
0	7.4	0.70	0.00	1.9	0.076	11.0	34	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67	0.9968	3.20	
2	7.8	0.76	0.04	2.3	0.092	15.0	54	0.9970	3.26	
3	11.2	0.28	0.56	1.9	0.075	17.0	60	0.9980	3.16	
4	7.4	0.70	0.00	1.9	0.076	11.0	34	0.9978	3.51	

```
import tensorflow as t
```

```
from tensorflow.keras.utils import to_categorical
```

```
ip = pro.drop('quality',axis = 1)
```

```
op = to_categorical(pro['quality'])
```

```
ip.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulp
0	7.4	0.70	0.00	1.9	0.076	11.0	34	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67	0.9968	3.20	
2	7.8	0.76	0.04	2.3	0.092	15.0	54	0.9970	3.26	
3	11.2	0.28	0.56	1.9	0.075	17.0	60	0.9980	3.16	
4	7.4	0.70	0.00	1.9	0.076	11.0	34	0.9978	3.51	

```
op
```

```
array([[1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0.],
       ...,
       [0., 0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0.]], dtype=float32)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
layer1 = Dense(1)
layer2 = Dense(100)
layer3 = Dense(100)
layer4 = Dense(6,activation='softmax')
```

```
model = Sequential()
model.add(layer1)
model.add(layer2)
model.add(layer3)
model.add(layer4)
```

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics='accuracy')
```

```
model.fit(ip,op,epochs=100)
```

```
Epoch 1/100
40/40 [==============================] - 1s 2ms/step - loss: 1.5103 - accuracy: 0.34
Epoch 2/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2806 - accuracy: 0.42
Epoch 3/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2485 - accuracy: 0.41
Epoch 4/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2269 - accuracy: 0.40
Epoch 5/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2100 - accuracy: 0.41
Epoch 6/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2090 - accuracy: 0.40
Epoch 7/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1967 - accuracy: 0.41
Epoch 8/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2059 - accuracy: 0.42
Epoch 9/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1991 - accuracy: 0.40
Epoch 10/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2089 - accuracy: 0.38
Epoch 11/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2011 - accuracy: 0.41
Epoch 12/100
40/40 [==============================] - 0s 2ms/step - loss: 1.2028 - accuracy: 0.40
Epoch 13/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1962 - accuracy: 0.40
Epoch 14/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1984 - accuracy: 0.41
Epoch 15/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1950 - accuracy: 0.39
Epoch 16/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1862 - accuracy: 0.42
Epoch 17/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1857 - accuracy: 0.42
Epoch 18/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1856 - accuracy: 0.40
Epoch 19/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1863 - accuracy: 0.42
Epoch 20/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1870 - accuracy: 0.44
Epoch 21/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1851 - accuracy: 0.44
Epoch 22/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1723 - accuracy: 0.45
Epoch 23/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1812 - accuracy: 0.43
Epoch 24/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1682 - accuracy: 0.46
Epoch 25/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1672 - accuracy: 0.46
Epoch 26/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1596 - accuracy: 0.46
Epoch 27/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1594 - accuracy: 0.46
Epoch 28/100
40/40 [==============================] - 0s 2ms/step - loss: 1.1560 - accuracy: 0.46
Epoch 29/100
```

```
40/40 [==============================] - 0s 2ms/step - loss: 1.1503 - accuracy: 0.47
Epoch 30/100
```

```
layer1.get_weights()
```

```
[array([[-0.02176535],
        [-1.3099122 ],
        [ 0.5902772 ],
        [ 0.02437038],
        [-1.1422433 ],
        [ 0.01081705],
        [-0.01120253],
        [-0.9274585 ],
        [ 0.13598995],
        [ 0.95154184],
        [ 0.47605044]], dtype=float32), array([-0.38201872], dtype=float32)]
```

```
import pandas as p
```

```
project = p.read_excel('/content/wine_Testing.xlsx')
```

```
project.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulp
0	9.8	0.300	0.39	1.7	0.062	3	9.0	0.99480	3.14	
1	7.1	0.460	0.20	1.9	0.077	28	54.0	0.99560	3.37	
2	7.1	0.460	0.20	1.9	0.077	28	54.0	0.99560	3.37	
3	7.9	0.765	0.00	2.0	0.084	9	22.0	0.99619	3.33	
4	8.7	0.630	0.28	2.7	0.096	17	69.0	0.99734	3.26	

```
project.dtypes
```

```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide       int64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

```
project['quality'].unique()
```

```
project['quality'].unique()

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

```
quality_dict = {}
x = 0
for i in project['quality'].unique():
    quality_dict[i] = x
    x = x + 1
```

```
project['quality'] = project['quality'].map(quality_dict)
```

```
project.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulp
0	9.8	0.300	0.39	1.7	0.062	3	9.0	0.99480	3.14	
1	7.1	0.460	0.20	1.9	0.077	28	54.0	0.99560	3.37	
2	7.1	0.460	0.20	1.9	0.077	28	54.0	0.99560	3.37	
3	7.9	0.765	0.00	2.0	0.084	9	22.0	0.99619	3.33	
4	8.7	0.630	0.28	2.7	0.096	17	69.0	0.99734	3.26	

```
ip_test = project.drop('quality',axis = 1)
```

```
ip_test.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulp
0	9.8	0.300	0.39	1.7	0.062	3	9.0	0.99480	3.14	
1	7.1	0.460	0.20	1.9	0.077	28	54.0	0.99560	3.37	
2	7.1	0.460	0.20	1.9	0.077	28	54.0	0.99560	3.37	
3	7.9	0.765	0.00	2.0	0.084	9	22.0	0.99619	3.33	
4	8.7	0.630	0.28	2.7	0.096	17	69.0	0.99734	3.26	

```
op_test = to_categorical(project['quality'])
```

```
op_test

     array([[1., 0., 0., 0., 0., 0.],
            [0., 1., 0., 0., 0., 0.],
            [0., 1., 0., 0., 0., 0.],
            ...,
            [0., 1., 0., 0., 0., 0.],
            [0., 0., 1., 0., 0., 0.],
            [0., 1., 0., 0., 0., 0.]], dtype=float32)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
layer1 = Dense(1)
layer2 = Dense(50)
layer3 = Dense(50)
layer4 = Dense(6,activation='softmax')
```

```
model = Sequential()
model.add(layer1)
model.add(layer2)
model.add(layer3)
model.add(layer4)
```

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics='accuracy')
```

```
model.fit(ip_test,op_test,epochs=100)
```

```
    Epoch 72/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1101 - accuracy: 0.51
    Epoch 73/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1123 - accuracy: 0.49
    Epoch 74/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1232 - accuracy: 0.44
    Epoch 75/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1087 - accuracy: 0.46
    Epoch 76/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1154 - accuracy: 0.49
    Epoch 77/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1152 - accuracy: 0.48
    Epoch 78/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1141 - accuracy: 0.47
    Epoch 79/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1069 - accuracy: 0.45
    Epoch 80/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1142 - accuracy: 0.46
    Epoch 81/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1339 - accuracy: 0.42
    Epoch 82/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1041 - accuracy: 0.47
    Epoch 83/100
    10/10 [==============================] - 0s 3ms/step - loss: 1.1050 - accuracy: 0.45
    Epoch 84/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1260 - accuracy: 0.47
    Epoch 85/100
    10/10 [==============================] - 0s 2ms/step - loss: 1.1250 - accuracy: 0.45
    Epoch 86/100
```

```
Epoch 86/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1658 - accuracy: 0.45
Epoch 87/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1454 - accuracy: 0.42
Epoch 88/100
10/10 [==============================] - 0s 2ms/step - loss: 1.2465 - accuracy: 0.42
Epoch 89/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1099 - accuracy: 0.49
Epoch 90/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1328 - accuracy: 0.43
Epoch 91/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1086 - accuracy: 0.47
Epoch 92/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1201 - accuracy: 0.47
Epoch 93/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1175 - accuracy: 0.45
Epoch 94/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1337 - accuracy: 0.43
Epoch 95/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1135 - accuracy: 0.48
Epoch 96/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1140 - accuracy: 0.49
Epoch 97/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1076 - accuracy: 0.45
Epoch 98/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1146 - accuracy: 0.48
Epoch 99/100
10/10 [==============================] - 0s 2ms/step - loss: 1.1058 - accuracy: 0.47

Epoch 100/100
10/10 [==============================] - 0s 3ms/step - loss: 1.1081 - accuracy: 0.48
<tensorflow.python.keras.callbacks.History at 0x7f185c43add0>
```

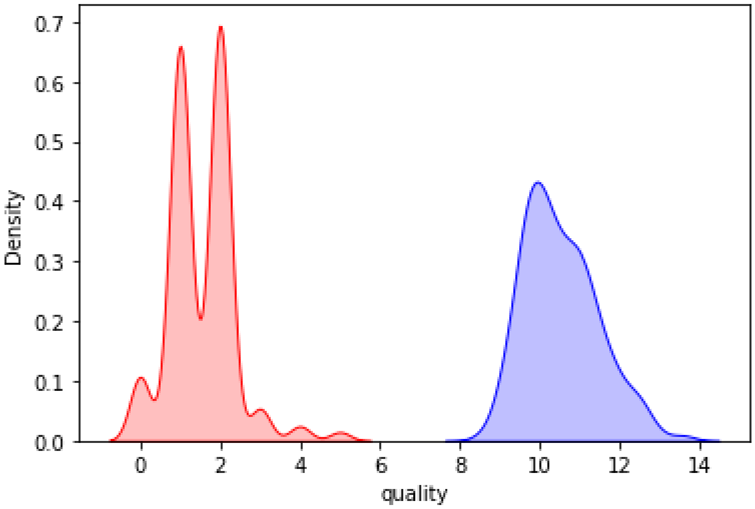
```
import seaborn as s
```

```
project.dtypes
```

```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide       int64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
dtype: object
```

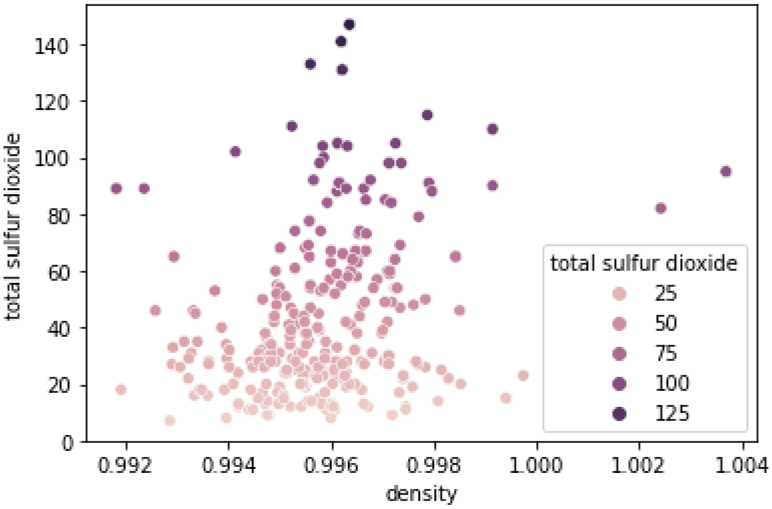
```
s.kdeplot(project['quality'], shade=True, color="r")
s.kdeplot(project['alcohol'], shade=True, color="b")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1847855790>
```



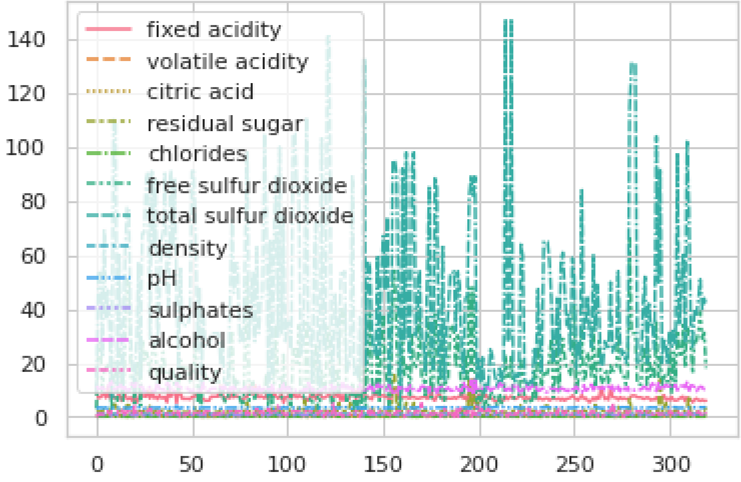
```
s.scatterplot(data=project, x="density", y="total sulfur dioxide", hue="total sulfur dioxide'
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1847737450>
```

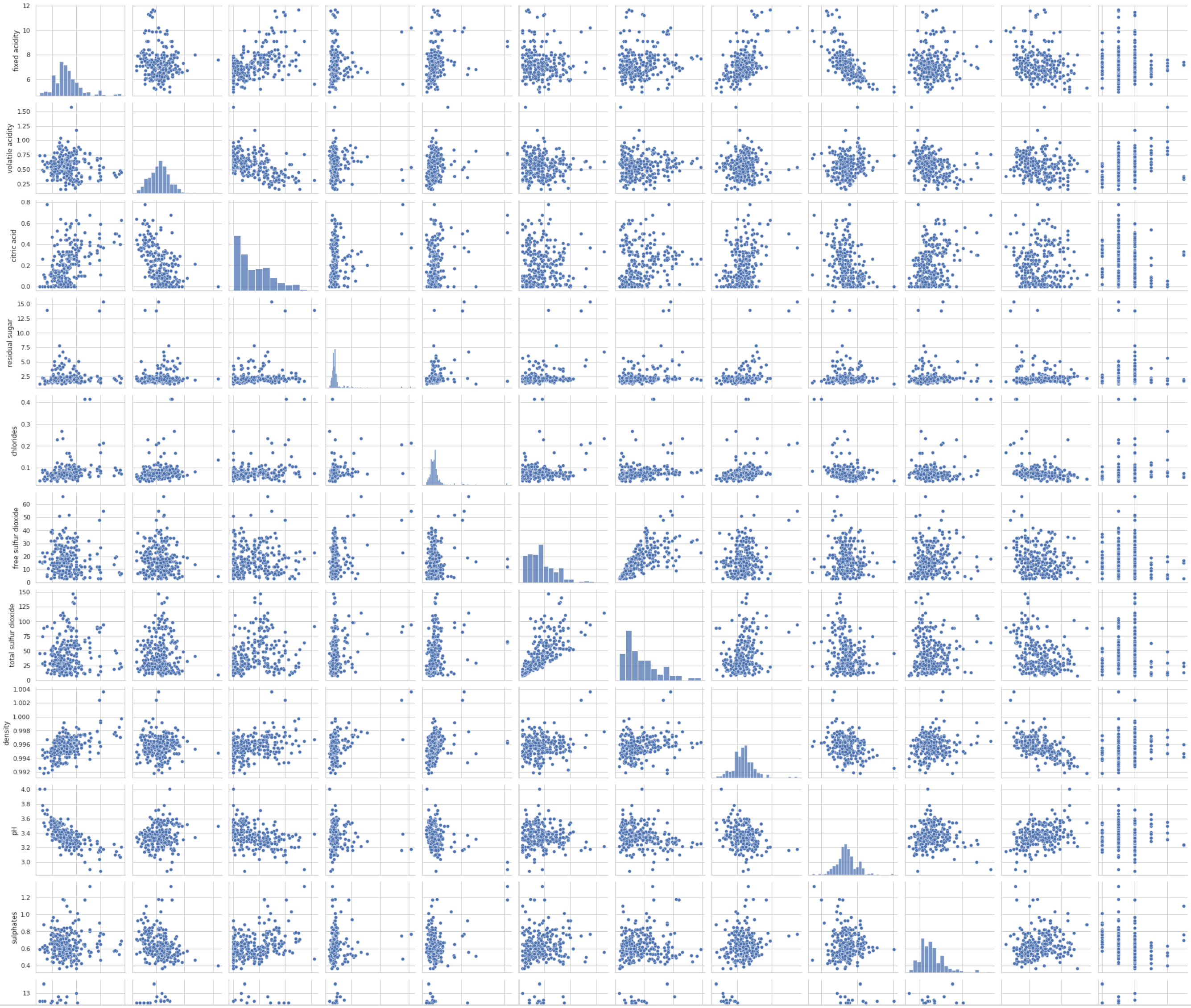


```
s.lineplot(data = project )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f18474a7710>
```



```
<seaborn.axisgrid.PairGrid at 0x7f1837c288d0>
```



✓ 1m 13s completed at 3:42 PM