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

In [2]:
```
wine = pd.read_csv(r'C:\Users\sushmitha\Downloads\WineQT.csv')
```

In [3]:
```
wine
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	al
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	

1143 rows × 13 columns

In [4]:
```
wine.head()
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9

In [5]:
```
pred_test = wine.iloc[3]
```

In [6]:
```
pred_test['type'] = 1
pred_test.drop(['quality','total sulfur dioxide'],inplace=True)
#pred_test.drop('total_sulfur_dioxide',inplace=True)
pred_test
```

Out[6]:
```
fixed acidity          11.200
volatile acidity        0.280
citric acid             0.560
residual sugar          1.900
chlorides               0.075
free sulfur dioxide    17.000
density                 0.998
pH                      3.160
sulphates               0.580
alcohol                 9.800
Id                      3.000
type                    1.000
Name: 3, dtype: float64
```

In [7]:
```
wine.shape
```

Out[7]:
```
(1143, 13)
```

In [8]:
```
wine.isnull().sum()
```

Out[8]:
```
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
Id                      0
dtype: int64
```

In [9]:
```
wine.describe()
```

Out[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sul diox
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000(
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45.9146
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32.7821
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000(
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21.000(
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37.000(
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61.000(
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289.000(

In [10]:
```
# One to remove na values is just by dropping them since they are very few
wine.dropna()
#another way is to impute let's say average value
#wine.update(wine.fillna(wine.mean()))
```

Out[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	al
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1138	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	
1139	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	
1140	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	
1141	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	
1142	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	

1143 rows × 13 columns

In [11]:
```
wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

In [12]:
```
wine.columns
```

Out[12]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual suga
r',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'densit
y',
       'pH', 'sulphates', 'alcohol', 'quality', 'Id'],
      dtype='object')
```

In [13]:
```
wine[("Type")].value_counts()
```

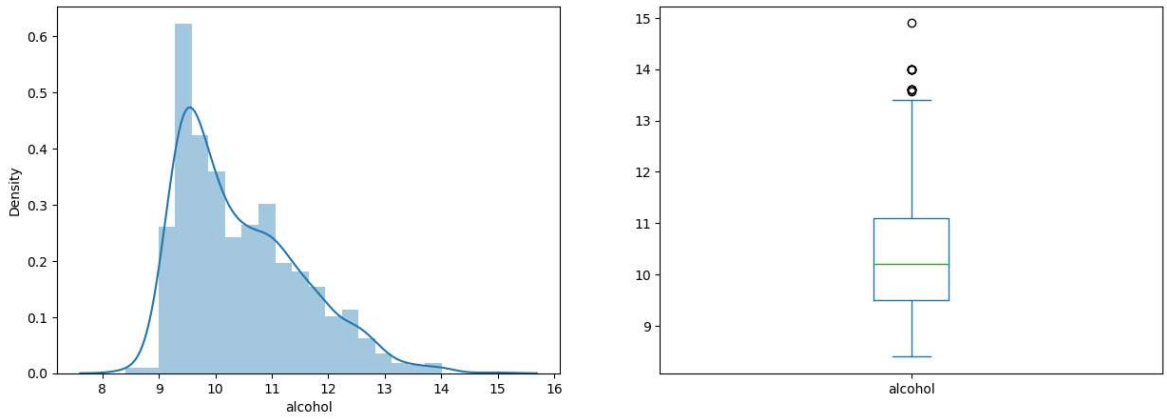
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self,
key, method, tolerance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.
IndexEngine.get_loc()

~\anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.
IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectH
ashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectH
ashTable.get_item()

KeyError: 'Type'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_37196\2522848030.py in <module>
----> 1 wine[("Type")].value_counts()

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, ke
y)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self,
key, method, tolerance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error
will raise

KeyError: 'Type'
```

In [14]:
```
sns.countplot(x="type", data=wine)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_37196\2937114997.py in <module>
----> 1 sns.countplot(x="type", data=wine)

~\anaconda3\lib\site-packages\seaborn\_decorators.py in inner_f(*args, **kw
args)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, arg
s)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

~\anaconda3\lib\site-packages\seaborn\categorical.py in countplot(x, y, hu
e, data, order, hue_order, orient, color, palette, saturation, dodge, ax, *
*kwargs)
   3596         raise ValueError("Cannot pass values for both `x` and `y`")
   3597
-> 3598     plotter = _CountPlotter(
   3599         x, y, hue, data, order, hue_order,
   3600         estimator, ci, n_boot, units, seed,

~\anaconda3\lib\site-packages\seaborn\categorical.py in __init__(self, x,
y, hue, data, order, hue_order, estimator, ci, n_boot, units, seed, orient,
color, palette, saturation, errcolor, errwidth, capsize, dodge)
   1582                  errwidth, capsize, dodge):
   1583         """Initialize the plotter."""
-> 1584         self.establish_variables(x, y, hue, data, orient,
   1585                                  order, hue_order, units)
   1586         self.establish_colors(color, palette, saturation)

~\anaconda3\lib\site-packages\seaborn\categorical.py in establish_variables
(self, x, y, hue, data, orient, order, hue_order, units)
    151                 if isinstance(var, str):
    152                     err = "Could not interpret input '{}'".format(v
ar)
--> 153                     raise ValueError(err)
    154
    155             # Figure out the plotting orientation

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

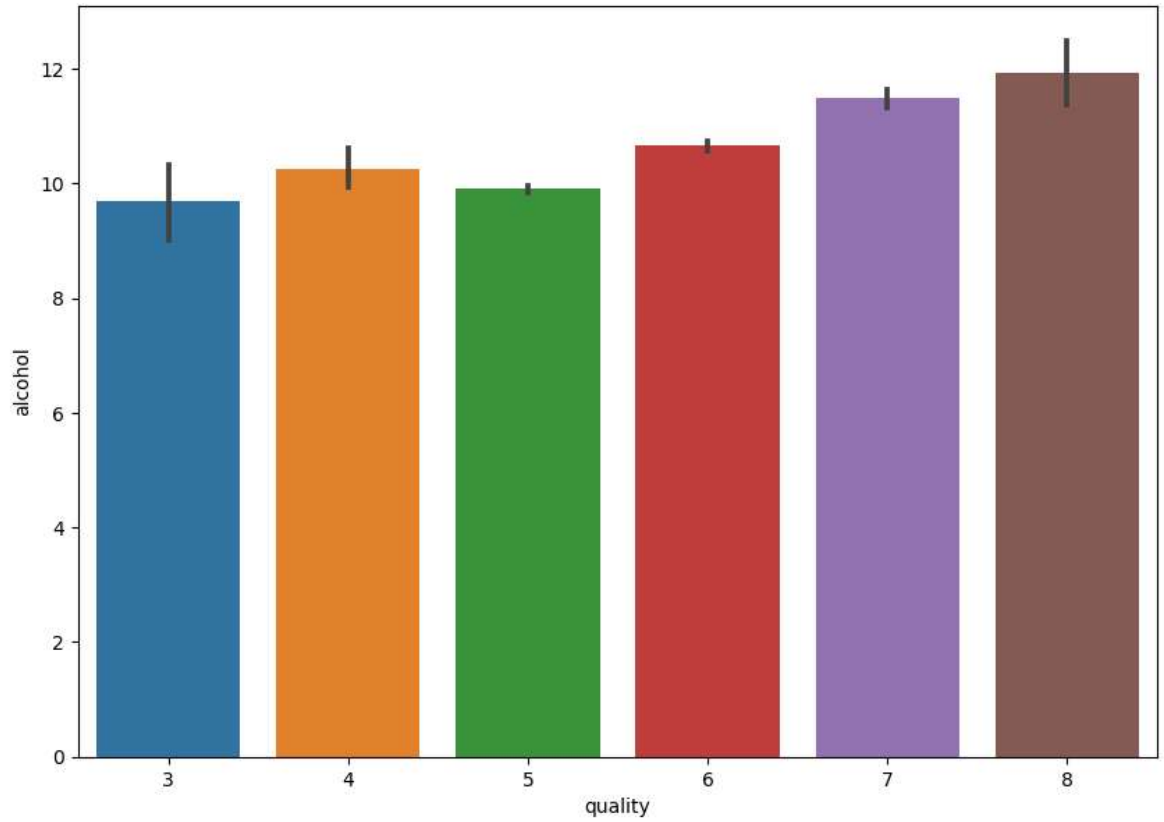
In [15]:
```
#Checking distribution and outlier for each variable
plt.figure(2)
plt.subplot(121)
sns.distplot(wine['alcohol'])
plt.subplot(122)
wine['alcohol'].plot.box(figsize=(15,5))
```

Out[15]: <AxesSubplot:>



In [16]:
```
#bivariate analysis to check quality with all the other variables
plt.figure(figsize=(10,7))
sns.barplot(x='quality',y='alcohol',data=wine)
```

Out[16]: <AxesSubplot:xlabel='quality', ylabel='alcohol'>



In [*]:
```
#Plotting all variables for their distribution and relation
sns.pairplot(wine)
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x1f2f8152eb0>
```

In [*]:
```
#checking correlation
wine.corr()
```

In [*]:
```
#buidling heatmap
plt.figure(figsize=(15,10))
sns.heatmap(wine.corr(), cmap='coolwarm')
```

In [*]:
```
#Dropping highly correlated variables - in this case total sulfur dioxide
wine_new = wine.drop('total sulfur dioxide',axis=1)
```

In [*]:
```
#Convert categorical value to dummies
wine_ml = pd.get_dummies(wine_new, drop_first=True)
wine_ml.head()
```

In [*]:
```
wine_ml.dtypes
```

In [*]:
```
wine_ml.dropna(inplace=True)
X = wine_ml.drop('quality',axis=1)
```

In [*]:
```
X.isnull().sum()
```

In [*]:
```
Y = wine_ml['quality'].apply(lambda y: 1 if y > 7 else 0)
Y
```

In [*]:
```
from sklearn.preprocessing import StandardScaler
```

In [*]:
```
scaler = StandardScaler()
scaler.fit(X)
x_standard = scaler.transform(X)
```

In [*]:
```
scaler = StandardScaler()
pred_test = np.asarray(pred_test).reshape(1,-1)
scaler.fit(pred_test)
pred_test_std = scaler.transform(pred_test)
```

In [*]:
```
X = x_standard


from sklearn.model_selection import train_test_split


X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=.2,random_s


from sklearn.linear_model import LogisticRegression


logreg = LogisticRegression()


logreg.fit(X_train, Y_train)
```

In [*]:
```
y_pred = logreg.predict(X_test)


pred_test_output = logreg.predict(pred_test_std)
pred_test_output
```

In [*]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_
accuracy_score(Y_test, y_pred)
```

In [*]:
```
print(classification_report(Y_test, y_pred))
```

In [*]:
```
confusion_matrix(Y_test, y_pred)
```

In [*]:
```
from sklearn.ensemble import RandomForestClassifier


rfc = RandomForestClassifier(n_estimators=200)


rfc.fit(X_train, Y_train)
```

In [*]:
```
rfc_pred = rfc.predict(X_test)


accuracy_score(Y_test, rfc_pred)
```

In [*]:
```
confusion_matrix(Y_test, rfc_pred)
```

In [*]:
```
classification_report(Y_test, rfc_pred)
```

In [*]:
```
print(classification_report(Y_test, rfc_pred))
```

In [*]:
```
rfc.feature_importances_
```

In [*]:
```
pd.Series(rfc.feature_importances_,index=wine_ml.drop('quality',axis=1).colum
```

In [ ]:
```

```