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

In [2]:
```
df = pd.read_csv("winequality.csv")
df.head()
```

Out[2]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [3]:
```
df.describe()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH
count	6487.000000	6489.000000	6494.000000	6495.000000	6495.000000	6497.000000	6497.000000	6497.000000	6488.000000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	115.744574	0.994697	3.218395
std	1.296750	0.164649	0.145265	4.758125	0.035036	17.749400	56.521855	0.002999	0.160748
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   type                  6497 non-null   object 
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
type                  0
fixed acidity        10
volatile acidity      8
citric acid           3
residual sugar        2
chlorides             2
free sulfur dioxide   0
total sulfur dioxide  0
density               0
pH                    9
sulphates             4
alcohol               0
quality               0
dtype: int64
```

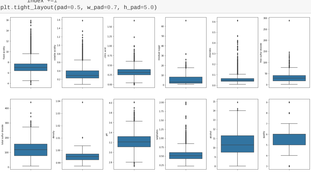
In [6]:
```
for col, value in df.items():
    if col !='type':
        df[col] = df[col].fillna(df[col].mean())
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
type                  0
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
```

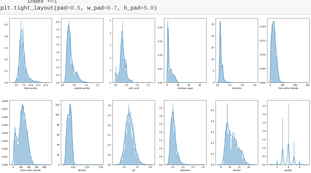
In [13]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(ncols=5, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()

for col, value in df.items():
    if col !='type':
        sns.boxplot(y=col, data = df, ax=ax[index])
        index +=1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



In [14]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(ncols=5, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()

for col, value in df.items():
    if col !='type':
        sns.distplot(value, ax=ax[index])
        index +=1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



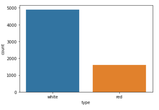
In [15]:
```
df['free sulfur dioxide'] = np.log(1+ df['free sulfur dioxide'])
```

In [16]:
```
sns.distplot(df['free sulfur dioxide'])
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1abbcd64ac0>
```

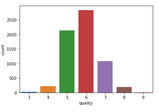
In [17]:
```
sns.countplot(df['type'])
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1abbceea850>
```



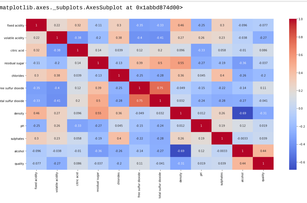
In [18]:
```
sns.countplot(df['quality'])
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1abbdd84940>
```



In [22]:
```
corr = df.corr()
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot =True, cmap='coolwarm')
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1abbd874d00>
```



In [27]:
```
X = df.drop(columns= ['type', 'quality'])
y = df['quality']
```

In [28]:
```
y.value_counts()
```

Out[28]:
```
6    2836
5    2138
7    1079
4     216
8     193
3      30
9       5
Name: quality, dtype: int64
```

In [30]:
```
from imblearn.over_sampling import SMOTE
oversample = SMOTE(k_neighbors=4)
X, y = oversample.fit_resample(X,y)
```

In [31]:
```
y.value_counts()
```

Out[31]:
```
9    2836
8    2836
7    2836
6    2836
5    2836
4    2836
3    2836
Name: quality, dtype: int64
```

In [36]:
```
from sklearn.model_selection import cross_val_score, train_test_split
def classify(model, X,y):
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=42
)
    model.fit(x_train, y_train)
    print("Accuracy:", model.score(x_test, y_test) * 100)

    score = cross_val_score(model, X,y, cv=5)
    print("CV Score", np.mean(score)*100)
```

In [37]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model, X, y)
```

```
Accuracy: 34.475115857344435
CV Score 32.545608051319164
```

In [39]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model, X,y)
```

```
Accuracy: 80.53596614950635
CV Score 74.97992688807456
```

In [42]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
classify(model, X,y)
```

```
Accuracy: 88.21277453153334
CV Score 82.52071726566727
```

In [ ]: