Untitled

December 4, 2023

[68]:
```python
import pandas as pd
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
# import sys
# !{sys.executable} -m pip install matplotlib==3.7.3
# !{sys.executable} -m pip install imblearn
```

[69]:
```python
df = pd.read_csv('wine-data-set .csv')
df
```

[69]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.0             0.270         0.36            20.7      0.045
1               6.3             0.300         0.34             1.6      0.049
2               8.1             0.280         0.40             6.9      0.050
3               7.2             0.230         0.32             8.5      0.058
4               7.2             0.230         0.32             8.5      0.058
…             …               …           …              …         …
6458            6.8             0.620         0.08             1.9      0.068
6459            6.2             0.600         0.08             2.0      0.090
6460            6.3             0.510         0.13             2.3      0.076
6461            5.9             0.645         0.12             2.0      0.075
6462            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    45.0                 170.0  1.00100  3.00       0.45
1                    14.0                 132.0  0.99400  3.30       0.49
2                    30.0                  97.0  0.99510  3.26       0.44
3                    47.0                 186.0  0.99560  3.19       0.40
4                    47.0                 186.0  0.99560  3.19       0.40
…                   …                    …      …    …        …
6458                 28.0                  38.0  0.99651  3.42       0.82
6459                 32.0                  44.0  0.99490  3.45       0.58
6460                 29.0                  40.0  0.99574  3.42       0.75
```

```
6461                    32.0                      44.0  0.99547  3.57        0.71
6462                    18.0                      42.0  0.99549  3.39        0.66

      alcohol  quality
0         8.8        6
1         9.5        6
2        10.1        6
3         9.9        6
4         9.9        6
…        …       …
6458      9.5        6
6459     10.5        5
6460     11.0        6
6461     10.2        5
6462     11.0        6

[6463 rows x 12 columns]
```

[70]:
```
df.describe()
```

[70]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6463.000000       6463.000000  6463.000000     6463.000000
mean        7.217755          0.339589     0.318758        5.443958
std         1.297913          0.164639     0.145252        4.756852
min         3.800000          0.080000     0.000000        0.600000
25%         6.400000          0.230000     0.250000        1.800000
50%         7.000000          0.290000     0.310000        3.000000
75%         7.700000          0.400000     0.390000        8.100000
max        15.900000          1.580000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6463.000000          6463.000000           6463.000000  6463.000000
mean      0.056056            30.516865            115.694492     0.994698
std       0.035076            17.758815             56.526736     0.003001
min       0.009000             1.000000              6.000000     0.987110
25%       0.038000            17.000000             77.000000     0.992330
50%       0.047000            29.000000            118.000000     0.994890
75%       0.065000            41.000000            156.000000     0.997000
max       0.611000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol      quality
count  6463.000000  6463.000000  6463.000000  6463.000000
mean      3.218332     0.531150    10.492825     5.818505
std       0.160650     0.148913     1.193128     0.873286
min       2.720000     0.220000     8.000000     3.000000
25%       3.110000     0.430000     9.500000     5.000000
50%       3.210000     0.510000    10.300000     6.000000
```

```
75%        3.320000      0.600000     11.300000      6.000000
max        4.010000      2.000000     14.900000      9.000000
```

[71]:
```
df.info()
```

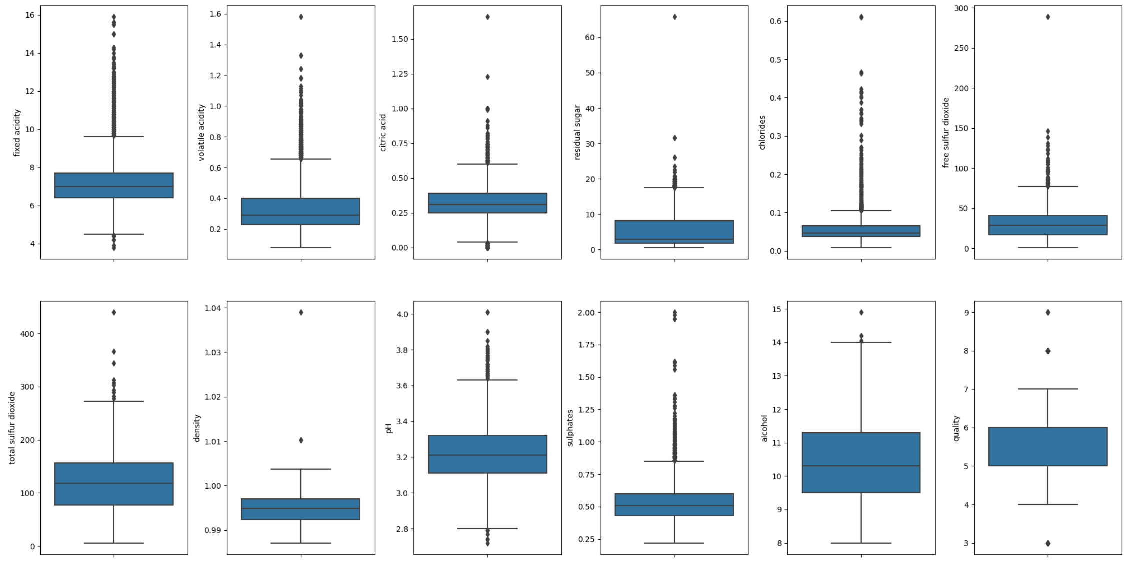
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6463 entries, 0 to 6462
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         6463 non-null   float64
 1   volatile acidity      6463 non-null   float64
 2   citric acid           6463 non-null   float64
 3   residual sugar        6463 non-null   float64
 4   chlorides             6463 non-null   float64
 5   free sulfur dioxide   6463 non-null   float64
 6   total sulfur dioxide  6463 non-null   float64
 7   density               6463 non-null   float64
 8   pH                    6463 non-null   float64
 9   sulphates             6463 non-null   float64
 10  alcohol               6463 non-null   float64
 11  quality               6463 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 606.0 KB
```

[72]:
```
df.isna().sum()
```

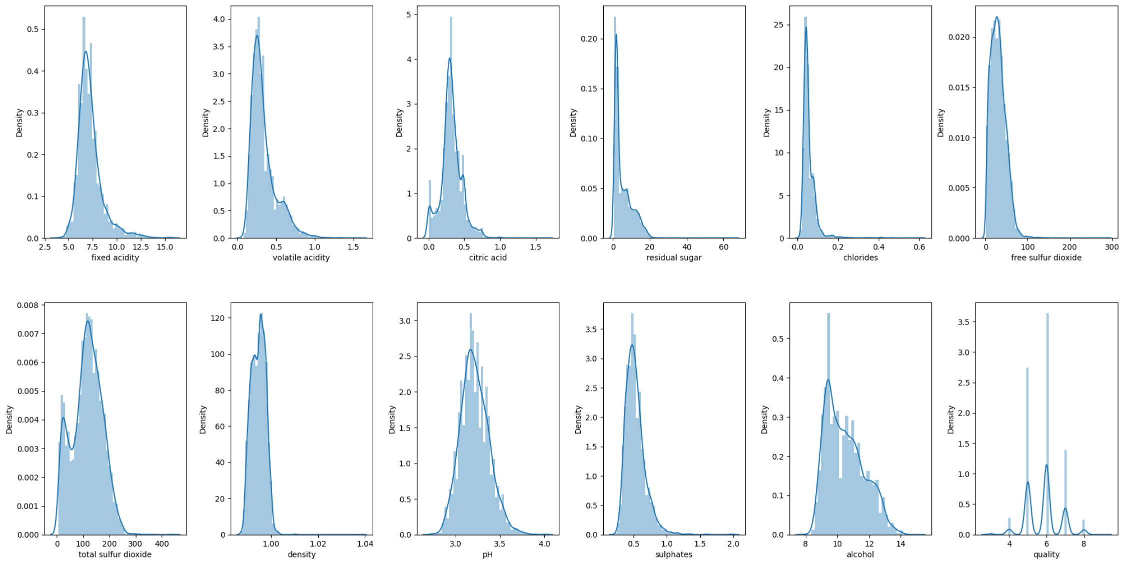
[72]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

[73]:
```
# create box plots
fig, ax = plt.subplots(ncols=6, 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)
```

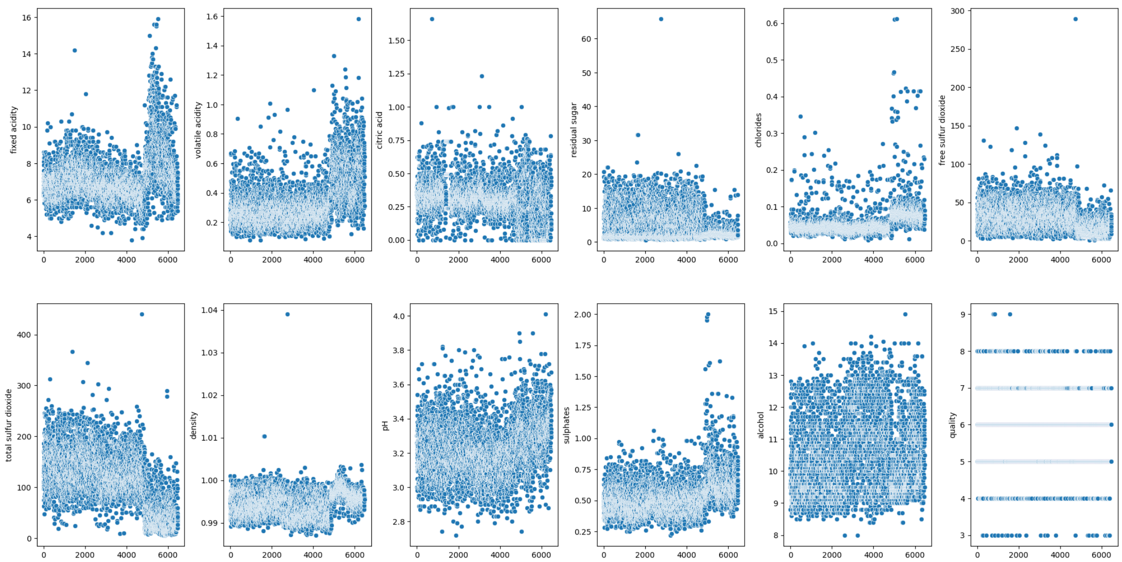


[74]:
```
# create dist plot
fig, ax = plt.subplots(ncols=6, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()

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)
```



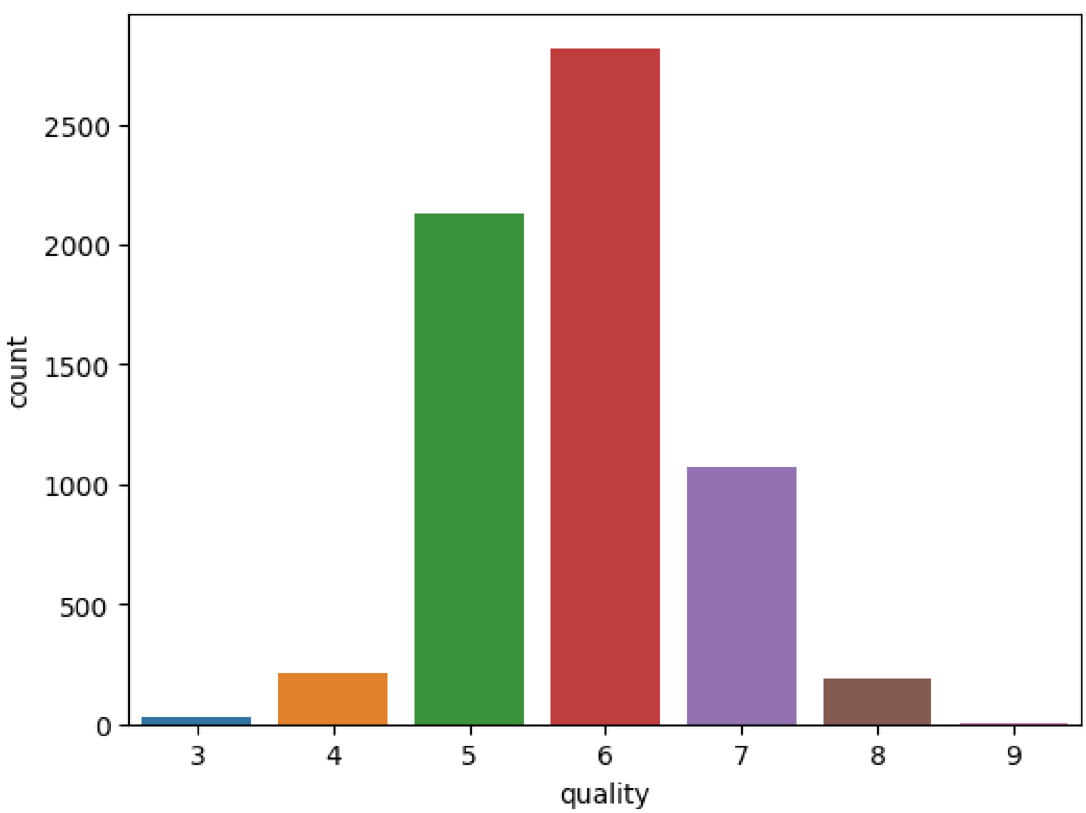
[75]:
```python
# # log transformation
# df['free sulfur dioxide'] = np.log(1 + df['free sulfur dioxide'])
# sns.distplot(df['free sulfur dioxide'])
```

[76]:
```python
# create dist plot
fig, ax = plt.subplots(ncols=6, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()

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



[77]:
```python
sns.countplot(x= df['quality'])
```

[77]: <Axes: xlabel='quality', ylabel='count'>



[78]:
```
corr = df.corr()
corr
```

[78]:

	fixed acidity	volatile acidity	citric acid
fixed acidity	1.000000	0.221066	0.323744
volatile acidity	0.221066	1.000000	-0.377512
citric acid	0.323744	-0.377512	1.000000
residual sugar	-0.113442	-0.196677	0.142324
chlorides	0.299104	0.377995	0.039412
free sulfur dioxide	-0.283485	-0.353402	0.132271
total sulfur dioxide	-0.330543	-0.414729	0.194398
density	0.459713	0.272101	0.097068
pH	-0.251121	0.260134	-0.327860
sulphates	0.301263	0.225656	0.059070
alcohol	-0.096190	-0.039528	-0.010056
quality	-0.076174	-0.266677	0.084926

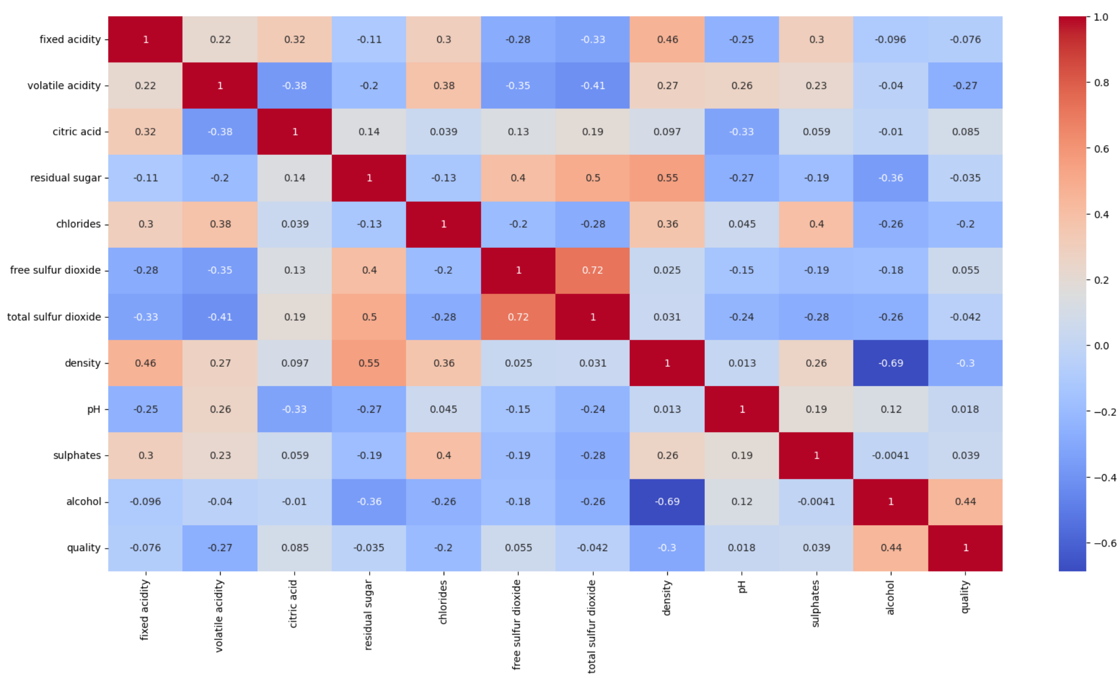
	residual sugar	chlorides	free sulfur dioxide
fixed acidity	-0.113442	0.299104	-0.283485
volatile acidity	-0.196677	0.377995	-0.353402
citric acid	0.142324	0.039412	0.132271
residual sugar	1.000000	-0.128814	0.403449
chlorides	-0.128814	1.000000	-0.195428
free sulfur dioxide	0.403449	-0.195428	1.000000
total sulfur dioxide	0.495684	-0.279602	0.721476
density	0.551494	0.363108	0.025113
pH	-0.266481	0.044653	-0.145164
sulphates	-0.185616	0.396240	-0.188947
alcohol	-0.359132	-0.257664	-0.179477
quality	-0.034654	-0.200553	0.054924

	total sulfur dioxide	density	pH	sulphates
fixed acidity	-0.330543	0.459713	-0.251121	0.301263
volatile acidity	-0.414729	0.272101	0.260134	0.225656
citric acid	0.194398	0.097068	-0.327860	0.059070
residual sugar	0.495684	0.551494	-0.266481	-0.185616
chlorides	-0.279602	0.363108	0.044653	0.396240
free sulfur dioxide	0.721476	0.025113	-0.145164	-0.188947
total sulfur dioxide	1.000000	0.031419	-0.237204	-0.275878
density	0.031419	1.000000	0.012525	0.260019
pH	-0.237204	0.012525	1.000000	0.190864
sulphates	-0.275878	0.260019	0.190864	1.000000
alcohol	-0.264385	-0.687432	0.120473	-0.004116
quality	-0.041598	-0.304447	0.018403	0.039054

```
                      alcohol   quality
fixed acidity        -0.096190 -0.076174
volatile acidity     -0.039528 -0.266677
citric acid          -0.010056  0.084926
residual sugar       -0.359132 -0.034654
chlorides            -0.257664 -0.200553
free sulfur dioxide  -0.179477  0.054924
total sulfur dioxide -0.264385 -0.041598
density              -0.687432 -0.304447
pH                    0.120473  0.018403
sulphates            -0.004116  0.039054
alcohol               1.000000  0.444637
quality               0.444637  1.000000
```

[79]:
```
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot=True, cmap='coolwarm' )
```

[79]: <Axes: >



[80]:
```
# Separate predictors (X) and target variable (y)
X = df.drop('quality', axis=1)
y = df['quality']
y.value_counts()
```

[80]:
```
6    2820
5    2128
7    1074
4     214
8     192
3      30
9       5
Name: quality, dtype: int64
```

[81]:
```
# classify function
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import accuracy_score
from imblearn.over_sampling import SMOTE

def classify(model, X, y):
    oversample = SMOTE(k_neighbors=4)
    # transform the dataset
    X, y = oversample.fit_resample(X, y)
    # print(y.value_counts())
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=42)
    # train the model
    model.fit(x_train, y_train)

    # Predict on the test set
    y_pred = model.predict(x_test)
    bottom_line_accuracy = accuracy_score(y_test, y_pred)
    print(f"Bottom-Line Accuracy: {bottom_line_accuracy*100:.4f}%")
    # print("Accuracy:", model.score(x_test, y_test) * 100)

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

[82]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(random_state=42)
print('Accuracy score without transforming any predictors:')
classify(model, X, y)
```

```
Accuracy score without transforming any predictors:
Bottom-Line Accuracy: 31.3070%
```

[83]:
```
# # log transformation
# df['free sulfur dioxide'] = np.log(1 + df['free sulfur dioxide'])
# sns.distplot(df['free sulfur dioxide'])
```

1 6. Applying Transformation on one predictor each time and getting accuracy

[84]:
```python
from sklearn.preprocessing import StandardScaler

scale_df = df.copy()
scaler = StandardScaler()
scaled_fixed_acidity = scaler.fit_transform(scale_df[['fixed acidity']])
scale_df['fixed acidity'] = scaled_fixed_acidity
X = df.drop('quality', axis=1)
y = df['quality']

print('Accuracy score with StandardScaler on predictor "fixed acidity":')
classify(model, X, y)
```

```
Accuracy score with StandardScaler on predictor "fixed acidity":
Bottom-Line Accuracy: 31.9149%
```

[85]:
```python
from sklearn.preprocessing import MinMaxScaler

scale_df = df.copy()
scaler = MinMaxScaler()
scaled_alcohol = scaler.fit_transform(scale_df[['alcohol']])
scale_df['alcohol'] = scaled_alcohol
X = df.drop('quality', axis=1)
y = df['quality']



print('Accuracy score with MinMaxScaler on predictor "alcohol":')
classify(model, X, y)
```

```
Accuracy score with MinMaxScaler on predictor "alcohol":
Bottom-Line Accuracy: 32.3202%
```

[86]:
```python
from sklearn.preprocessing import RobustScaler

scale_df = df.copy()
scaler = RobustScaler()
scaled_total_sulfur_dioxide = scaler.fit_transform(scale_df[['total sulfur␣
↪dioxide']])
scale_df['total sulfur dioxide'] = scaled_total_sulfur_dioxide
X = scale_df.drop('quality', axis=1)
y = scale_df['quality']

print('Accuracy score with RobustScaler on predictor "total sulfur dioxide":')
classify(model, X, y)
# X.describe()
```

```
Accuracy score with RobustScaler on predictor "total sulfur dioxide":
Bottom-Line Accuracy: 36.9605%
```

[87]:
```
from sklearn.preprocessing import MaxAbsScaler

scale_df = df.copy()
scaler = MaxAbsScaler()
scaled_free_sulfur_dioxide = scaler.fit_transform(scale_df[['free sulfur␣
↪dioxide']])
scale_df['free sulfur dioxide'] = scaled_free_sulfur_dioxide
X = scale_df.drop('quality', axis=1)
y = scale_df['quality']

# scaler = RobustScaler()
# X = scaler.fit_transform(X)

print('Accuracy score with MaxAbsScaler on predictor "free sulfur dioxide":')
classify(model, X, y)
# X.describe()
```

```
Accuracy score with MaxAbsScaler on predictor "free sulfur dioxide":
Bottom-Line Accuracy: 31.5299%
```

[88]:
```
from sklearn.preprocessing import QuantileTransformer

scale_df = df.copy()
scaler = QuantileTransformer()
scaled_residual_sugar = scaler.fit_transform(scale_df[['residual sugar']])
scale_df['residual sugar'] = scaled_residual_sugar

X = df.drop('quality', axis=1)
y = df['quality']

print('Accuracy score with QuantileTransformer on predictor "residual sugar":')
classify(model, X, y)
```

```
Accuracy score with QuantileTransformer on predictor "residual sugar":
Bottom-Line Accuracy: 32.2999%
```

2 8. Selecting subsets and getting accuracy

[89]:
```
# X = df.drop(columns =['quality', 'total sulfur dioxide', 'density'], axis=1)
X = df[['alcohol']]
y = df['quality']
# scaler = RobustScaler()
# X = scaler.fit_transform(X)
print('Selected subset ["alcohol"] based on correlation number closest to 1 for␣
↪"quality":')
```

```
classify(model, X, y)
```

```
Selected subset ["alcohol"] based on correlation number closest to 1 for
"quality":
Bottom-Line Accuracy: 28.7741%
```

[90]:
```
X = df[['alcohol', 'density', 'citric acid']]
y = df['quality']
# scaler = RobustScaler()
# X = scaler.fit_transform(X)
print("Selected subset [alcohol', 'density', 'citric acid'] on random:")
classify(model, X, y)
```

```
Selected subset [alcohol', 'density', 'citric acid'] on random:
Bottom-Line Accuracy: 30.1925%
```

[91]:
```
X = df[['alcohol', 'citric acid', 'free sulfur dioxide']]
y = df['quality']
# scaler = RobustScaler()
# X = scaler.fit_transform(X)
print("Selected subset ['alcohol', 'citric acid', 'free sulfur dioxide'] based␣
 ↪on correlation numbers closest to 1 for 'quality':")
classify(model, X, y)
```

```
Selected subset ['alcohol', 'citric acid', 'free sulfur dioxide'] based on
correlation numbers closest to 1 for 'quality':
Bottom-Line Accuracy: 33.1307%
```

[92]:
```
X = df[['density', 'volatile acidity', 'chlorides']]
y = df['quality']

print('Selected subset based on correlation numbers closest to negative 1 for␣
 ↪"quality":')
classify(model, X, y)
```

```
Selected subset based on correlation numbers closest to negative 1 for
"quality":
Bottom-Line Accuracy: 31.3070%
```

[93]:
```
X = df[['alcohol', 'density', 'volatile acidity']]
y = df['quality']
# scaler = RobustScaler()
# X = scaler.fit_transform(X)
print('Selected subset based on correlation numbers closest to 1 or negative 1␣
 ↪for "quality":')
classify(model, X, y)
```

```
Selected subset based on correlation numbers closest to 1 or negative 1 for
"quality":
Bottom-Line Accuracy: 29.5846%
```

Exploratory data analysis and initial model

To get The Bottom-Line Accuracy score of machine learning model named LogisticRegression I had to first look for null values in dataset. Then I looked for outliers using exploratory data analysis and plotting. I could have modified outlier data at this point in time. Since I was not asked to do it I moved on to the next step. Then I looked for correlation for all varibles which later became useful for selecting subsets. After that I split the variable into X and y, where y is the target variable quality. For model training I created classify function which takes in model, X and y. since y had severely imbalanced classes I used oversampling with SMOTE on X,y to generate new features from minority classes. this made sure all the classes in y have oversampled to the upper value. I used train_test_split from sklearn to split X and y in Train and test datasets. I kept 25% of the data in test datasets and and rest in train dataset. I fit x_train and y_train into model. then I get the model to predit on x_test to get y_preds. then comparint y_test and y_preds using accuracy score function I get bottom-line accuracy score. Initalially i use Logistic Recression model and passed it with X and y to classify function without transforming them.

Inital Bottom-Line Accuracy: 31.3070%

Transforming on predictor each time

Accuracy score with StandardScaler on predictor “fixed acidity”: 31.9149%, Here Bottom-Line Accuracy increases from original prediction

Accuracy score with MinMaxScaler on predictor “alcohol”: 32.3202%, Here Bottom-Line Accuracy increases from original prediction

Accuracy score with RobustScaler on predictor “total sulfur dioxide”: 36.9605%, Here Bottom-Line Accuracy increases from original prediction

Accuracy score with MaxAbsScaler on predictor “free sulfur dioxide”: 31.5299%, Here Bottom-Line Accuracy increases from original prediction

Accuracy score with QuantileTransformer on predictor “residual sugar”: 32.2999%, Here Bottom-Line Accuracy increases from original prediction

So, I found that all of these transformatons increased the accuracy score of the model. I also checked that if i used transformation of more than one predictor at each pass then accuracy improves further to close to 50% when RobustScaler is used on all predictors of X. Since this question restricts me to transform only one predictor at each pass I did not include it in the project.

Selecting different subsets each time

Selected subset [“alcohol”] based on correlation number closest to 1 for “quality”: 28.7741%, Here Bottom-Line Accuracy decreases from original prediction

Selected subset [alcohol’, ‘density’, ‘citric acid’] on random: 30.1925%, Here Bottom-Line Accuracy decreases from original prediction

Selected subset [‘alcohol’, ‘citric acid’, ‘free sulfur dioxide’] based on correlation numbers closest to 1 for ‘quality’: 33.1307%, Here Bottom-Line Accuracy increases from original prediction

Selected subset [‘density’, ‘volatile acidity’, ‘chlorides’] based on correlation numbers closest to negative 1 for “quality”: 31.3070%, Here Bottom-Line Accuracy is same from original prediction

Selected subset [‘alcohol’, ‘density’, ‘volatile acidity’] based on correlation numbers closest to 1 or negative 1 for “quality”: 29.5846%, Here Bottom-Line Accuracy decreases from original prediction

So I found that only subset [‘alcohol’, ‘citric acid’, ‘free sulfur dioxide’] increased the accuracy score of the model, Which is just as I expected since the 3 variables in this subset have the strongest positive correlation with quality variable. knowing the correlations was helpful to create this subset.

[ ]:

```
```