white_wine

April 24, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
white_wine = pd.read_csv('winequality-white.csv')
```

0.1 Understanding the structure of the dataset

[3]:
```
white_wine.head()
```

[3]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
3            7.2              0.23         0.32             8.5      0.058
4            7.2              0.23         0.32             8.5      0.058

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40

   alcohol  quality
0      8.8        6
1      9.5        6
2     10.1        6
3      9.9        6
4      9.9        6
```

[4]:
```
white_wine.tail()
```

[4]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
4893            6.2              0.21         0.29             1.6      0.039
4894            6.6              0.32         0.36             8.0      0.047
4895            6.5              0.24         0.19             1.2      0.041
```

```
4896            5.5              0.29         0.30              1.1       0.022
4897            6.0              0.21         0.38              0.8       0.020

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  
4893                 24.0                  92.0  0.99114  3.27       0.50   \
4894                 57.0                 168.0  0.99490  3.15       0.46
4895                 30.0                 111.0  0.99254  2.99       0.46
4896                 20.0                 110.0  0.98869  3.34       0.38
4897                 22.0                  98.0  0.98941  3.26       0.32

      alcohol  quality
4893     11.2        6
4894      9.6        5
4895      9.4        6
4896     12.8        7
4897     11.8        6
```

[5]:
```
white_wine.shape
```

[5]:
```
(4898, 12)
```

[6]:
```
white_wine.columns
```

[6]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

[7]:
```
white_wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64  
dtypes: float64(11), int64(1)
```

```
memory usage: 459.3 KB
```

[8]:
```python
white_wine.isnull().sum()
```

[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
dtype: int64
```

[9]:
```python
white_wine.describe()
```

[9]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    4898.000000       4898.000000  4898.000000     4898.000000
mean        6.854788          0.278241     0.334192        6.391415
std         0.843868          0.100795     0.121020        5.072058
min         3.800000          0.080000     0.000000        0.600000
25%         6.300000          0.210000     0.270000        1.700000
50%         6.800000          0.260000     0.320000        5.200000
75%         7.300000          0.320000     0.390000        9.900000
max        14.200000          1.100000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  4898.000000          4898.000000           4898.000000  4898.000000
mean      0.045772            35.308085            138.360657     0.994027
std       0.021848            17.007137             42.498065     0.002991
min       0.009000             2.000000              9.000000     0.987110
25%       0.036000            23.000000            108.000000     0.991723
50%       0.043000            34.000000            134.000000     0.993740
75%       0.050000            46.000000            167.000000     0.996100
max       0.346000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol      quality
count  4898.000000  4898.000000  4898.000000  4898.000000
mean      3.188267     0.489847    10.514267     5.877909
std       0.151001     0.114126     1.230621     0.885639
min       2.720000     0.220000     8.000000     3.000000
25%       3.090000     0.410000     9.500000     5.000000
50%       3.180000     0.470000    10.400000     6.000000
75%       3.280000     0.550000    11.400000     6.000000
```

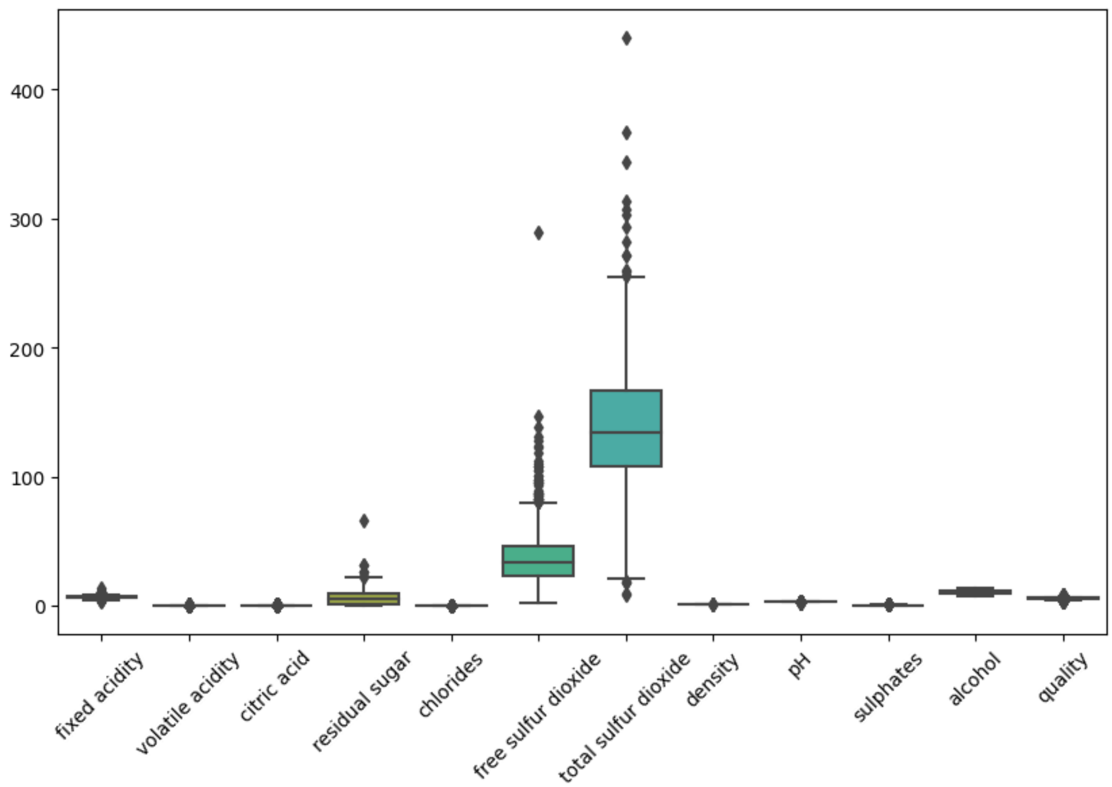
```
max        3.820000      1.080000     14.200000      9.000000
```

0.2 Outlier Detection

0.2.1 1. Boxplot

[10]:
```python
import seaborn as sns

plt.subplots(figsize=(10, 6))
plt.xticks(rotation=45)
sns.boxplot(data=white_wine)
```

[10]: <Axes: >

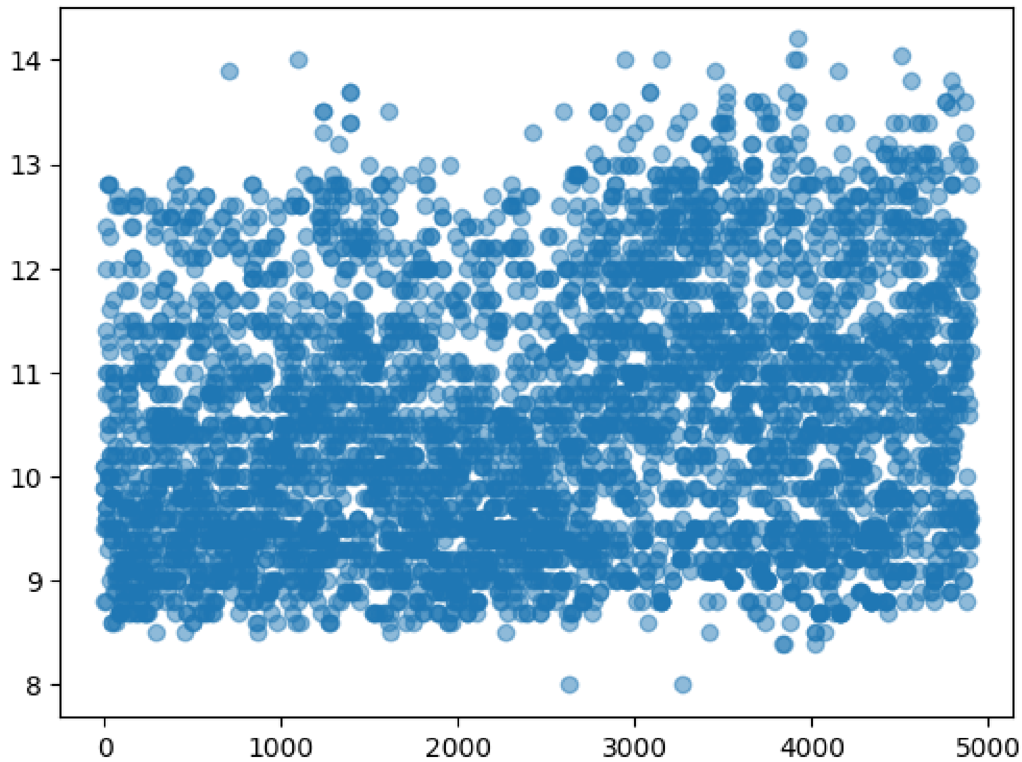


0.3 2. Zscore

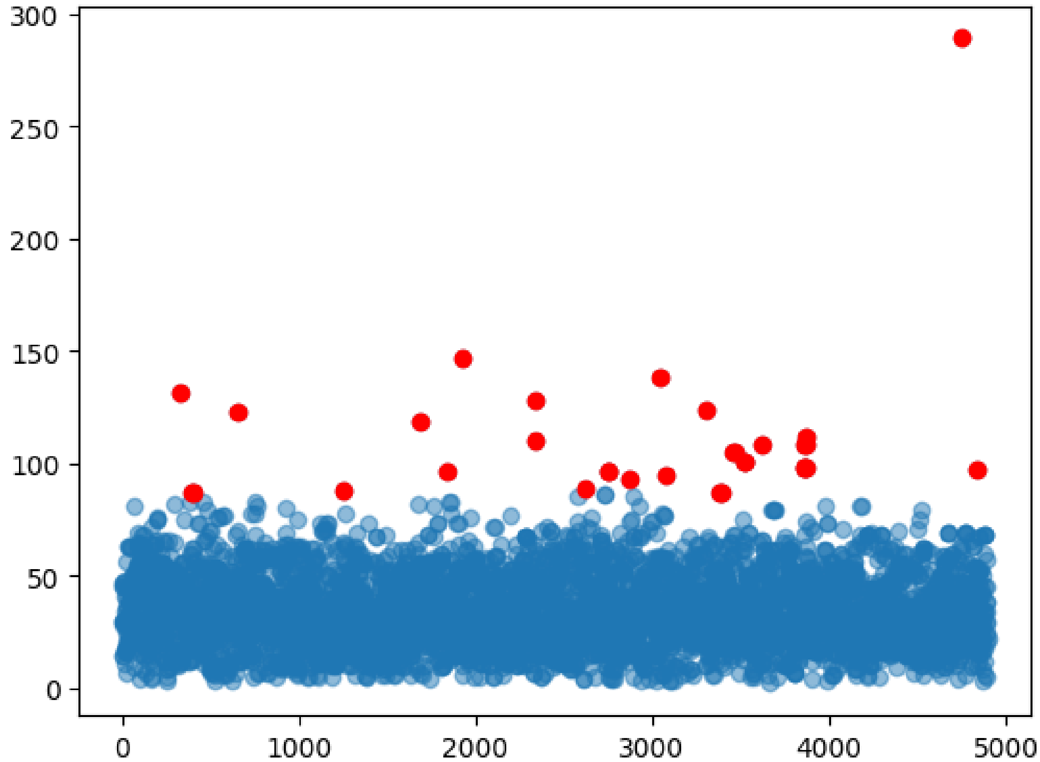
[11]:
```python
# In this code, the zscore() function is used to calculate the Z-scores for␣
↪each data point in the "alcohol" column.
# The threshold variable is set to 3, which means that any data point with a␣
↪Z-score greater than 3 or less than -3 is considered an outlier.
from scipy import stats
```

```
z_scores = stats.zscore(white_wine['alcohol'])
threshold = 3

outliers = white_wine[np.abs(z_scores) > threshold]
```

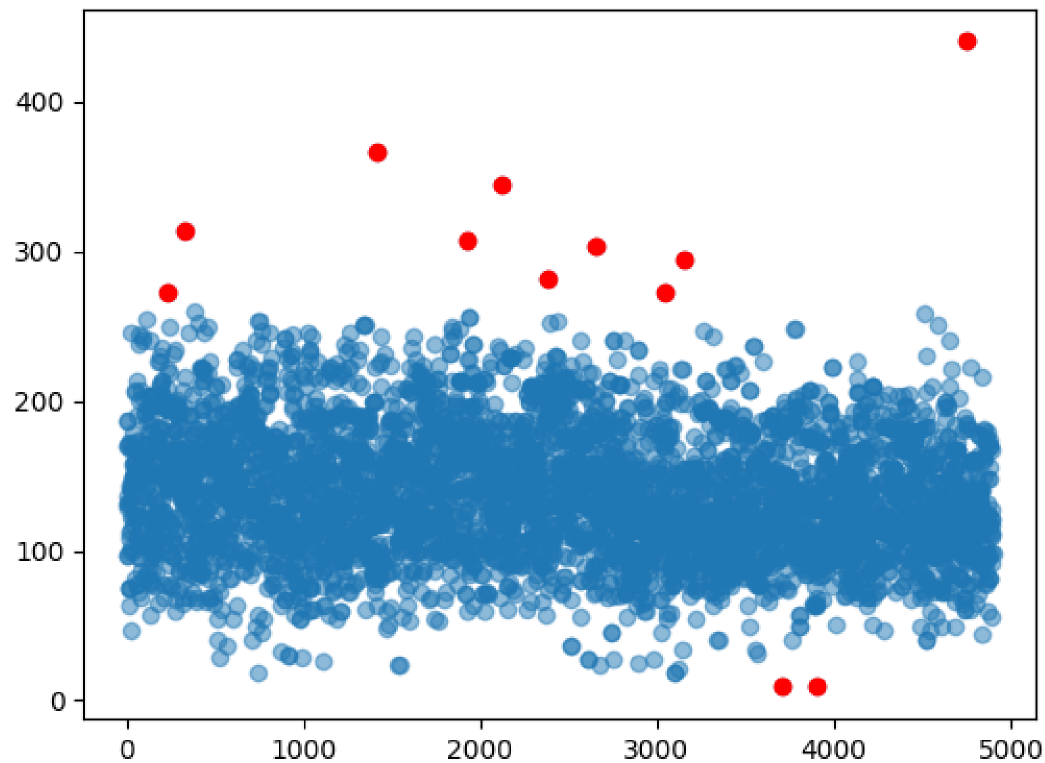
[12]:
```
import matplotlib.pyplot as plt

plt.scatter(white_wine.index, white_wine['alcohol'], alpha=0.5)
plt.scatter(outliers.index, outliers['alcohol'], color='r')
plt.show()
```



[13]:
```
z_scores = stats.zscore(white_wine['free sulfur dioxide'])
threshold = 3

outliers = white_wine[np.abs(z_scores) > threshold]

plt.scatter(white_wine.index, white_wine['free sulfur dioxide'], alpha=0.5)
plt.scatter(outliers.index, outliers['free sulfur dioxide'], color='r')
plt.show()
```

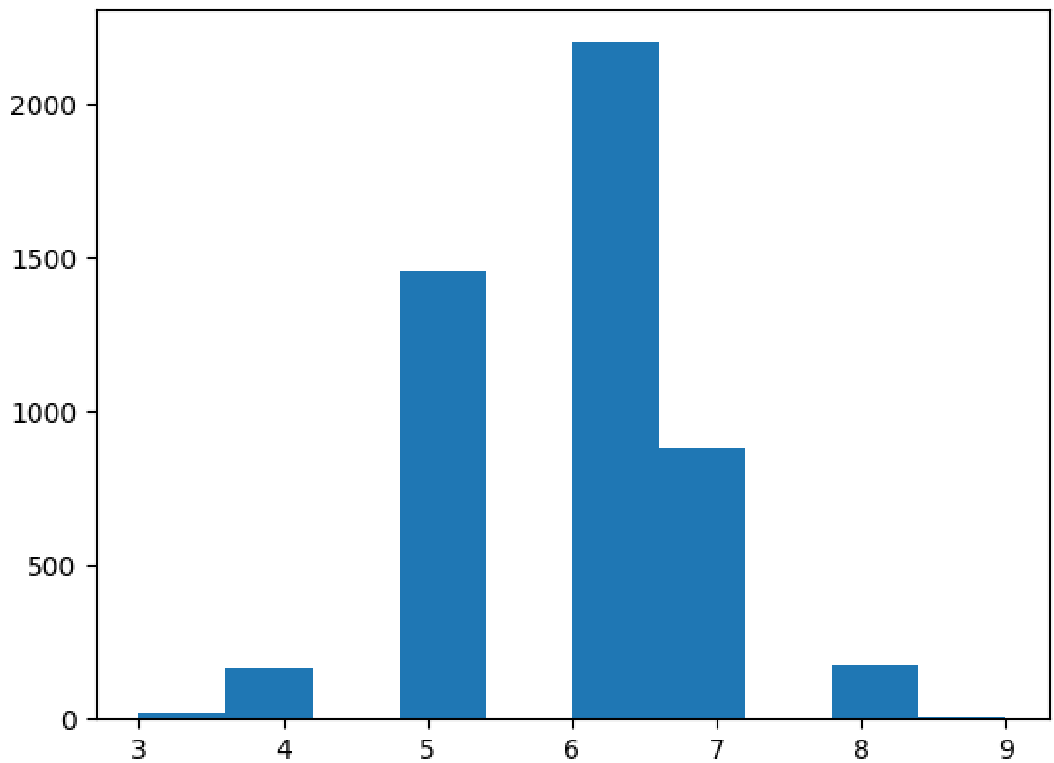


[14]:
```
z_scores = stats.zscore(white_wine['total sulfur dioxide'])
threshold = 3

outliers = white_wine[np.abs(z_scores) > threshold]

plt.scatter(white_wine.index, white_wine['total sulfur dioxide'], alpha=0.5)
plt.scatter(outliers.index, outliers['total sulfur dioxide'], color='r')
plt.show()
```



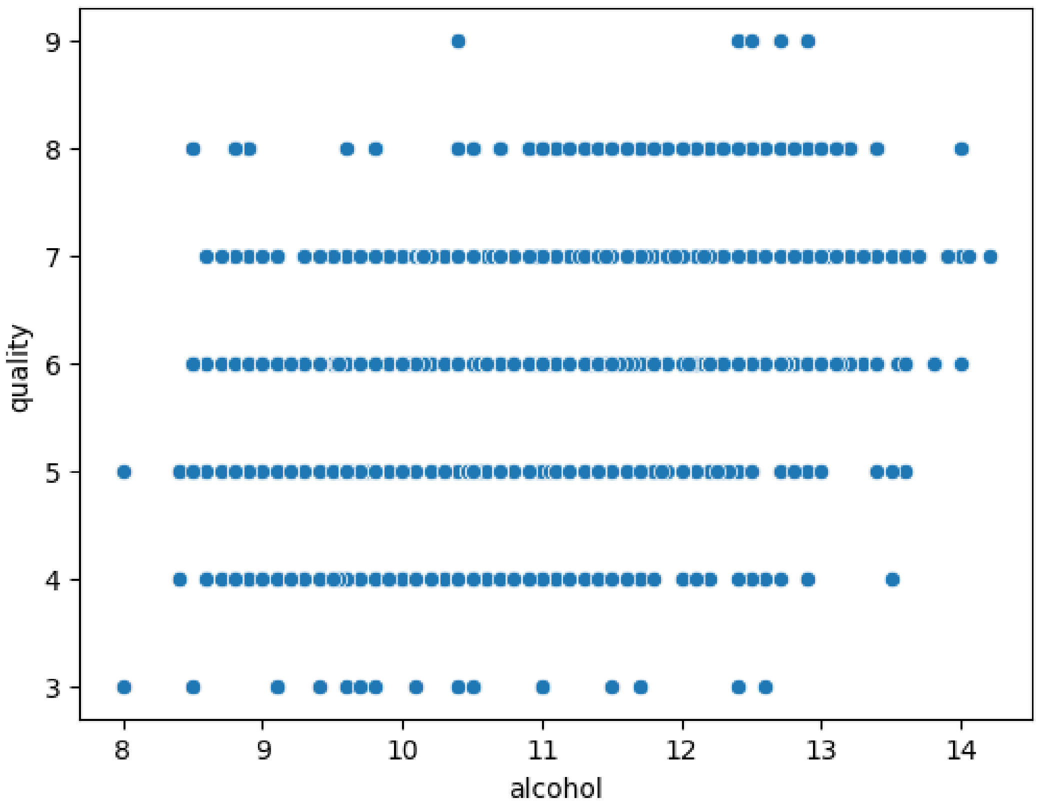
0.4 Data Visualization

[15]:
```
plt.hist(white_wine['quality'])
plt.show()
```



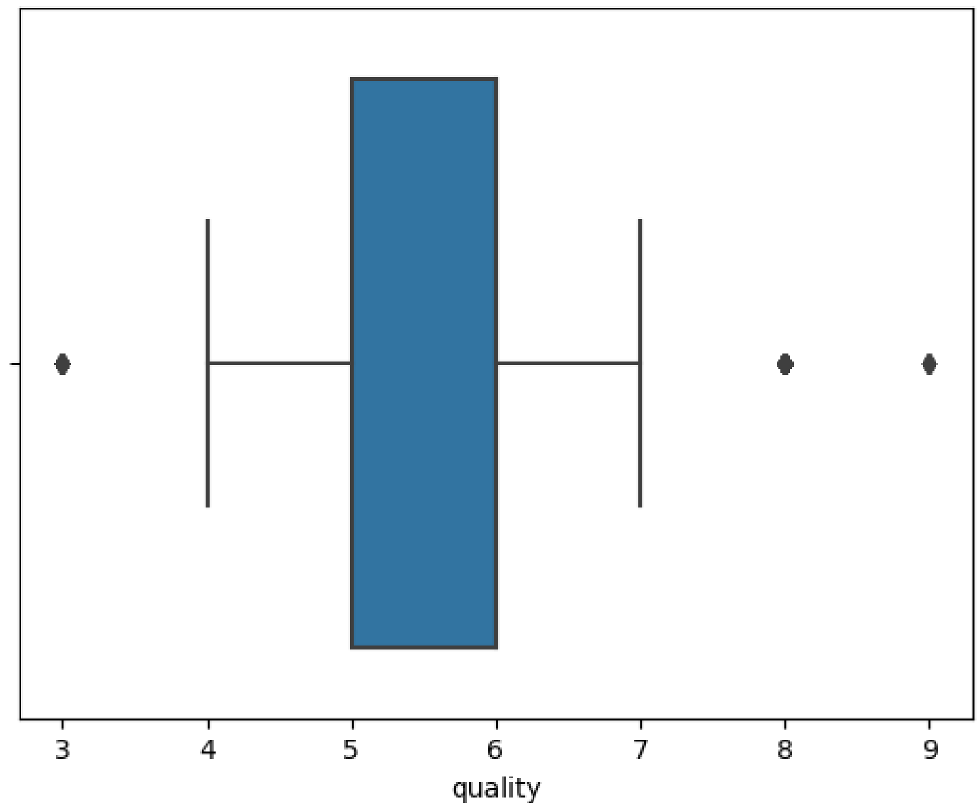
[16]:
```
import seaborn as sns

sns.scatterplot(x='alcohol', y='quality', data=white_wine)
```

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



[17]:
```
sns.boxplot(x='quality', data=white_wine)
```

[17]: <Axes: xlabel='quality'>



0.4.1 Splitting the data and creating regression model

[18]:
```
from sklearn.model_selection import train_test_split
```

[19]:
```
X = white_wine.drop('quality', axis=1) # Extract the input features
y = white_wine['quality'] # Extract the target variable
```

[20]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

0.4.2 1. Regression model

[21]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

[22]:
```
# Create a linear regression model
lr_model = LinearRegression()
```

[23]:
```
# Fit the model on the training set
lr_model.fit(X_train, y_train)
```

[23]: LinearRegression()

[24]:
```
# Make predictions on the testing set
y_pred = lr_model.predict(X_test)
```

[25]:
```
# Calculate RMSE, MAE, MSE, and R2 score
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

[26]:
```
print('RMSE:', rmse)
print('MAE:', mae)
print('MSE:', mse)
print('R-squared Score:', r2)
```

```
RMSE: 0.7543373063341975
MAE: 0.5862665383273417
MSE: 0.569024771727533
R-squared Score: 0.26527500421196626
```

0.5 2. XGboost

[27]:
```
from xgboost import XGBRegressor
```

[28]:
```
# Create an XGBoost model
xgb_model = XGBRegressor(objective='reg:squarederror', random_state=42)
```

[29]:
```
# Fit the model on the training set
xgb_model.fit(X_train, y_train)
```

[29]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=42, …)
```

[30]:
```
# Make predictions on the testing set
y_pred = xgb_model.predict(X_test)
```

[31]:
```
# Calculate RMSE, MAE, MSE, and R2 score
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

```python
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

[32]:
```python
print('RMSE:', rmse)
print('MAE:', mae)
print('MSE:', mse)
print('R-squared Score:', r2)
```

```
RMSE: 0.6195150148674802
MAE: 0.4430355848098288
MSE: 0.38379885364625416
R-squared Score: 0.5044387781702404
```

0.6 3. Artificial Nural Network

[33]:
```python
import keras
from keras.models import Sequential
from keras.layers import Dense
```

[34]:
```python
# Create a neural network model
ann_model = Sequential()
ann_model.add(Dense(128, input_dim=X_train.shape[1], activation='relu'))
ann_model.add(Dense(64, activation='relu'))
ann_model.add(Dense(32, activation='relu'))
ann_model.add(Dense(16, activation='relu'))
ann_model.add(Dense(1))
```

[35]:
```python
# Compile the model
ann_model.compile(loss='mean_squared_error', optimizer='adam')
```

[36]:
```python
# Fit the model on the training set
history = ann_model.fit(X_train, y_train, validation_split=0.2, epochs=100,␣
↪batch_size=32)
```

```
Epoch 1/100
98/98 [==============================] - 1s 3ms/step - loss: 1.5772 - val_loss:
1.0134
Epoch 2/100
98/98 [==============================] - 0s 2ms/step - loss: 0.7714 - val_loss:
0.6445
Epoch 3/100
98/98 [==============================] - 0s 2ms/step - loss: 0.7738 - val_loss:
1.3599
Epoch 4/100
98/98 [==============================] - 0s 2ms/step - loss: 0.7684 - val_loss:
0.6699
Epoch 5/100
```

```
98/98 [==============================] - 0s 2ms/step - loss: 0.7151 - val_loss:
0.6347
Epoch 6/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6236 - val_loss:
0.6178
Epoch 7/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6475 - val_loss:
0.6942
Epoch 8/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6665 - val_loss:
0.6143
Epoch 9/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6181 - val_loss:
0.6378
Epoch 10/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6476 - val_loss:
0.6540
Epoch 11/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6729 - val_loss:
0.6022
Epoch 12/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6668 - val_loss:
0.6102
Epoch 13/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6693 - val_loss:
0.8844
Epoch 14/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6205 - val_loss:
0.6398
Epoch 15/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6467 - val_loss:
0.6863
Epoch 16/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6384 - val_loss:
1.0206
Epoch 17/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6514 - val_loss:
0.6396
Epoch 18/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5889 - val_loss:
0.6208
Epoch 19/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6277 - val_loss:
0.6327
Epoch 20/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6191 - val_loss:
0.6051
Epoch 21/100
```

```
98/98 [==============================] - 0s 2ms/step - loss: 0.6138 - val_loss:
0.7263
Epoch 22/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6452 - val_loss:
0.7269
Epoch 23/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6596 - val_loss:
0.6082
Epoch 24/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5770 - val_loss:
0.5905
Epoch 25/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6100 - val_loss:
0.6073
Epoch 26/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5908 - val_loss:
0.5944
Epoch 27/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6352 - val_loss:
0.7148
Epoch 28/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6270 - val_loss:
0.6629
Epoch 29/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5820 - val_loss:
0.5950
Epoch 30/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5886 - val_loss:
0.5876
Epoch 31/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6046 - val_loss:
0.5861
Epoch 32/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5616 - val_loss:
0.5949
Epoch 33/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5818 - val_loss:
0.6110
Epoch 34/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6277 - val_loss:
0.8992
Epoch 35/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6109 - val_loss:
0.6235
Epoch 36/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6188 - val_loss:
0.5922
Epoch 37/100
```

```
98/98 [==============================] - 0s 2ms/step - loss: 0.5727 - val_loss:
0.6034
Epoch 38/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5847 - val_loss:
0.7017
Epoch 39/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5751 - val_loss:
0.5974
Epoch 40/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5869 - val_loss:
0.7480
Epoch 41/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6119 - val_loss:
0.5957
Epoch 42/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5658 - val_loss:
0.6221
Epoch 43/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5905 - val_loss:
0.6112
Epoch 44/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5833 - val_loss:
0.6001
Epoch 45/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5984 - val_loss:
0.6292
Epoch 46/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5746 - val_loss:
0.6207
Epoch 47/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5685 - val_loss:
0.5933
Epoch 48/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5777 - val_loss:
0.5975
Epoch 49/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5829 - val_loss:
0.6542
Epoch 50/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5810 - val_loss:
0.6127
Epoch 51/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5590 - val_loss:
0.6153
Epoch 52/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5478 - val_loss:
0.6798
Epoch 53/100
```

```
98/98 [==============================] - 0s 2ms/step - loss: 0.6385 - val_loss:
0.6448
Epoch 54/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5688 - val_loss:
0.6016
Epoch 55/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5876 - val_loss:
0.6400
Epoch 56/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5809 - val_loss:
0.5862
Epoch 57/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5632 - val_loss:
0.5878
Epoch 58/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5433 - val_loss:
0.7575
Epoch 59/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5608 - val_loss:
0.6100
Epoch 60/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5669 - val_loss:
0.5871
Epoch 61/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5586 - val_loss:
0.5762
Epoch 62/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5980 - val_loss:
0.6664
Epoch 63/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5984 - val_loss:
0.6979
Epoch 64/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5628 - val_loss:
0.6057
Epoch 65/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5548 - val_loss:
0.5800
Epoch 66/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5429 - val_loss:
0.6555
Epoch 67/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5552 - val_loss:
0.7153
Epoch 68/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5507 - val_loss:
0.5775
Epoch 69/100
```

```
98/98 [==============================] - 0s 2ms/step - loss: 0.5542 - val_loss:
0.6103
Epoch 70/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5500 - val_loss:
0.5982
Epoch 71/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5578 - val_loss:
0.5870
Epoch 72/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5458 - val_loss:
0.6093
Epoch 73/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5845 - val_loss:
0.7946
Epoch 74/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5451 - val_loss:
0.5812
Epoch 75/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5507 - val_loss:
0.6544
Epoch 76/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6082 - val_loss:
0.5852
Epoch 77/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5695 - val_loss:
0.5862
Epoch 78/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5653 - val_loss:
0.6422
Epoch 79/100
98/98 [==============================] - 0s 2ms/step - loss: 0.6223 - val_loss:
0.6057
Epoch 80/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5410 - val_loss:
0.6071
Epoch 81/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5485 - val_loss:
0.6327
Epoch 82/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5586 - val_loss:
0.5691
Epoch 83/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5592 - val_loss:
0.5789
Epoch 84/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5400 - val_loss:
0.5763
Epoch 85/100
```

```
98/98 [==============================] - 0s 2ms/step - loss: 0.6104 - val_loss:
0.5718
Epoch 86/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5445 - val_loss:
0.5960
Epoch 87/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5495 - val_loss:
0.6059
Epoch 88/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5561 - val_loss:
0.5839
Epoch 89/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5819 - val_loss:
0.6118
Epoch 90/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5748 - val_loss:
0.5779
Epoch 91/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5287 - val_loss:
0.5750
Epoch 92/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5387 - val_loss:
0.5785
Epoch 93/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5479 - val_loss:
0.7274
Epoch 94/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5545 - val_loss:
0.5648
Epoch 95/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5404 - val_loss:
0.5732
Epoch 96/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5806 - val_loss:
0.6400
Epoch 97/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5347 - val_loss:
0.5733
Epoch 98/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5560 - val_loss:
0.7660
Epoch 99/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5936 - val_loss:
0.5846
Epoch 100/100
98/98 [==============================] - 0s 2ms/step - loss: 0.5656 - val_loss:
0.6199
```

[38]:
```python
# Make predictions on the testing set
y_pred = ann_model.predict(X_test)
```

```
31/31 [==============================] - 0s 1ms/step
```

[39]:
```python
# Calculate RMSE, MAE, MSE, and R2 score
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

[40]:
```python
print('RMSE:', rmse)
print('MAE:', mae)
print('MSE:', mse)
print('R-squared Score:', r2)
```

```
RMSE: 0.7724077908358856
MAE: 0.6055406913465383
MSE: 0.5966137953439733
R-squared Score: 0.22965204671075723
```

[ ]:
```python

```