# CM3005WineQualityMT

January 4, 2025

# 1 Data Science Midterms: Wine Quality Analysis

# 2 1. Domain-specific area and objective of the project

This project's domain-specific area is wine, or the wine industry. Wine quality is arguably one of the most important factors about wine, other processes such as wine production is arguably just a means to an end. Factors such as price and production rate could also be important in balancing cost-effectiveness for both the manufacturer and consumer, but we will only focus on the wines' quality in this project.

Wine quality is affected by physicochemical properties such as levels of acidity, types of acidity, levels of sugar, types of sugar, alcohol content, amongst several others. Such physicochemical properties make up sensory properties used by wine sommeliers to grade wines. This means that we could develop a linear regression model to predict wine quality based on physicochemical properties. This model could then automate the process of wine grading, or aid sommeliers in the grading process.

The objectives of this project is to: 1. Develop a model using physicochemical properties to predict wine quality 2. Understand which of these physicochemical properties influence wine quality the most 3. Use these findings to provide winemakers a guideline or range of values of these physicochemical properties

We are able to use a linear regression model in this domain because: 1. Wine quality is influenced by phyisochemical properties and their levels, which values are numerical and can be modeled linearly. 2. Linear regression calculates how much weight or influence each physicochemical property has, and can provide a range or upper and lower limits of how much each property should be present in wine. 3. Linear regression is simple and straightforward to implement, while having been applied successfully to similar domains, such as coffee bean quality.

What am I trying to achieve? 1. Aiding sommeliers in the wine grading process or even automating it. This means that the model has to be able to make accurate and consistent predictions. 2. Aiding winemakers in wine production. Using the data and insights found, a guideline of physicochemical properties can be referenced in the wine production process.

# 3 2. Dataset description

The dataset is a combination of 2 datasets, 1 of red wines, with a sample size of 1599 and the other of white wines, with a sample size of 4898. In total, the combined dataset has a sample size of 6497. The datasets can be found on the UC Irvine Machine Learning Repository website (https://archive.ics.uci.edu/dataset/186/wine+quality). It contains the physicochemical properties of the wines, were rated by wine sommerliers and given a quality score. The ratings are based on sensory data, such as smell, taste, and texture. The wines are rated by calculating the median of at least 3 evaluations made by wine experts. These experts used a scale of 0 to 10 to grade the wine quality, with 0 being the worst and 10 being the best. As for the data format, each row represents an individual wine sample, and as the 12 features or attributes are measured in numerical values, linear regression can be done.

There are 11 physicochemical properties and 1 output attribute. The 11 properties were recorded in the float data type, while the output attribute, the quality score, was recorded as an integer. The 11 physicochemical properties are: 1. Fixed acidity 2. Volatile acidity 3. Citric acid 4. Residual sugar 5. Chlorides 6. Free sulfur dioxide 7. Total sulfur dioxide 8. Density 9. pH 10. Sulphates 11. Alcohol

Why does this dataset fit the above purpose? 1. The dataset includes features that are in numerical values that directly influence wine quality, making it ideal for this project's purpose of building, testing and evaluating a linear regression model. 2. The 11 physicochemical properties align with industry-standards and are acknowledged to affect wine quality. 3. The dataset's structure fits very well with doing regression analysis as it has continuous numerical features and a defined target.

# 4 3. Data preparation / preprocessing

[207]:
```
import pandas as pd

# load datasets with delimiter
red_wine = pd.read_csv("./wine+quality/winequality-red.csv", sep=";")
white_wine = pd.read_csv("./wine+quality/winequality-white.csv", sep=";")

# preview data
print(red_wine.head())
print(white_wine.head())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
```

```
2      9.8        5
3      9.8        6
4      9.4        5
   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
```

[250]:

```python
# combine the datasets into a dataframe
winesdf = pd.concat([red_wine, white_wine], axis=0)
print(winesdf.head())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

[209]:
```python
# check size, type and missing values
print(winesdf.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6497 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         6497 non-null   float64
 1   volatile acidity      6497 non-null   float64
 2   citric acid           6497 non-null   float64
 3   residual sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free sulfur dioxide   6497 non-null   float64
 6   total sulfur dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  quality               6497 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 659.9 KB
None
```

[210]:
```python
# check if there are duplicate index values as stated above "Index: 6497␣
 ↪entries, 0 to 4897"
print(winesdf.index.duplicated().sum())
```

```
1599
```

[211]:
```python
# reseting index
winesdf = winesdf.reset_index(drop=True)

# check again after reset
print(winesdf.index.duplicated().sum())
```

```
0
```

[212]:
```python
# double check for null values
print(winesdf.isnull().sum())
```

```
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
```

[251]:

```python
# preprocessing, normalising feature values
from sklearn.preprocessing import MinMaxScaler

# select columns for normalization excluding target variable
numeric_features = winesdf.select_dtypes(include=['float64', 'int64']).
 ↪drop('quality', axis=1).columns

# MinMax Scaling
scaler_minmax = MinMaxScaler()
winesdf[numeric_features] = scaler_minmax.
 ↪fit_transform(winesdf[numeric_features])
```

[214]:

```python
# check scaled values
print(winesdf.describe())
```

```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6497.000000       6497.000000  6497.000000     6497.000000
mean        0.282257          0.173111     0.191948        0.074283
std         0.107143          0.109758     0.087541        0.072972
min         0.000000          0.000000     0.000000        0.000000
25%         0.214876          0.100000     0.150602        0.018405
50%         0.264463          0.140000     0.186747        0.036810
75%         0.322314          0.213333     0.234940        0.115031
max         1.000000          1.000000     1.000000        1.000000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6497.000000          6497.000000           6497.000000  6497.000000
mean      0.078129             0.102518              0.252868     0.146262
std       0.058195             0.061630              0.130235     0.057811
min       0.000000             0.000000              0.000000     0.000000
25%       0.048173             0.055556              0.163594     0.100829
50%       0.063123             0.097222              0.258065     0.149990
75%       0.093023             0.138889              0.345622     0.190476
max       1.000000             1.000000              1.000000     1.000000

                pH    sulphates      alcohol      quality
count  6497.000000  6497.000000  6497.000000  6497.000000
mean      0.386435     0.174870     0.361131     5.818378
std       0.124641     0.083599     0.172857     0.873255
min       0.000000     0.000000     0.000000     3.000000
25%       0.302326     0.117978     0.217391     5.000000
50%       0.379845     0.162921     0.333333     6.000000
```

```
75%        0.465116     0.213483     0.478261     6.000000
max        1.000000     1.000000     1.000000     9.000000
```

# 5 4. Statistical analysis

[215]:

```python
# measures of central tendency and spread
print(winesdf.describe())
```

```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6497.000000       6497.000000  6497.000000     6497.000000
mean        0.282257          0.173111     0.191948        0.074283
std         0.107143          0.109758     0.087541        0.072972
min         0.000000          0.000000     0.000000        0.000000
25%         0.214876          0.100000     0.150602        0.018405
50%         0.264463          0.140000     0.186747        0.036810
75%         0.322314          0.213333     0.234940        0.115031
max         1.000000          1.000000     1.000000        1.000000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6497.000000          6497.000000           6497.000000  6497.000000
mean      0.078129             0.102518              0.252868     0.146262
std       0.058195             0.061630              0.130235     0.057811
min       0.000000             0.000000              0.000000     0.000000
25%       0.048173             0.055556              0.163594     0.100829
50%       0.063123             0.097222              0.258065     0.149990
75%       0.093023             0.138889              0.345622     0.190476
max       1.000000             1.000000              1.000000     1.000000

                pH    sulphates      alcohol      quality
count  6497.000000  6497.000000  6497.000000  6497.000000
mean      0.386435     0.174870     0.361131     5.818378
std       0.124641     0.083599     0.172857     0.873255
min       0.000000     0.000000     0.000000     3.000000
25%       0.302326     0.117978     0.217391     5.000000
50%       0.379845     0.162921     0.333333     6.000000
75%       0.465116     0.213483     0.478261     6.000000
max       1.000000     1.000000     1.000000     9.000000
```

[217]:

```python
# median
winesdfmedian = winesdf.median()
print(winesdfmedian)
```

```
fixed acidity           0.264463
volatile acidity        0.140000
citric acid             0.186747
residual sugar          0.036810
chlorides               0.063123
free sulfur dioxide     0.097222
```

```
total sulfur dioxide    0.258065
density                 0.149990
pH                      0.379845
sulphates               0.162921
alcohol                 0.333333
quality                 6.000000
dtype: float64
```

[216]:
```
# variance
winesdfvar = winesdf.var()
print(winesdfvar)
```

```
fixed acidity           0.011480
volatile acidity        0.012047
citric acid             0.007663
residual sugar          0.005325
chlorides               0.003387
free sulfur dioxide     0.003798
total sulfur dioxide    0.016961
density                 0.003342
pH                      0.015535
sulphates               0.006989
alcohol                 0.029879
quality                 0.762575
dtype: float64
```

[218]:
```
# skewness
print(winesdf.skew())
```

```
fixed acidity           1.723290
volatile acidity        1.495097
citric acid             0.471731
residual sugar          1.435404
chlorides               5.399828
free sulfur dioxide     1.220066
total sulfur dioxide   -0.001177
density                 0.503602
pH                      0.386839
sulphates               1.797270
alcohol                 0.565718
quality                 0.189623
dtype: float64
```

Chlorides is very positively skewed with a score of 5.4, while total sulfur dioxide is very negatively skewed with a score of -0.001. The rest of the features have a moderate or slight positive skew.

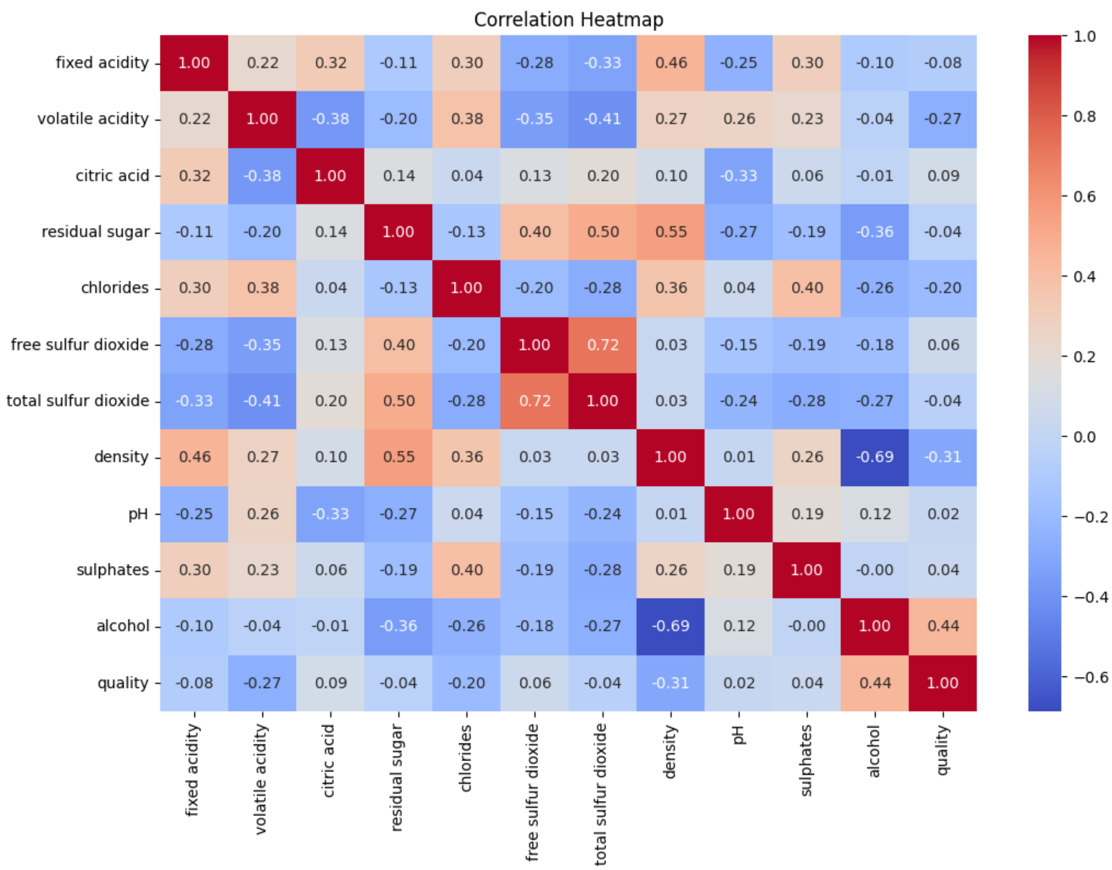
[219]:
```
# kurtosis
print(winesdf.kurtosis())
```

```
fixed acidity            5.061161
```

```
volatile acidity         2.825372
citric acid              2.397239
residual sugar           4.359272
chlorides               50.898051
free sulfur dioxide      7.906238
total sulfur dioxide    -0.371664
density                  6.606067
pH                       0.367657
sulphates                8.653699
alcohol                 -0.531687
quality                  0.232322
dtype: float64
```

Chlorides, sulphates, free sulfur dioxide, fixed acidity, and residual sugar are $> 3$, meaning they are leptokurtic (heavy tails and a sharp peak). While pH, total sulfur dioxide and alcohol are $< 3$, meaning that they are platykurtic (think tails and a flat peak). Volatile acidity and citric acid are very close to 3, and we can consider them as having a normal distribution.

[220]:
```
# correlation heat map
import matplotlib.pyplot as plt
import seaborn as sns

correlation_matrix = winesdf.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Heatmap")
plt.show()
```




### Correlation with quality

`Positive correlations:`

Alcohol shows a strong positive correlation with quality. Higher quality wines tend to have higher levels of alcohol. Seems to be a strong indicator of quality.

Citric acid shows a slight positive correlation quality. High quality wines may have slightly above average levels of citric acid. Seems to be an extremely weak indicator of quality.

`Negative correlations:`

Volatile acidity shows a strong negative correlation with quality. Higher quality wines tend to have lower levels of volatile acidity. Seems to be a decent indicator of quality.

Density shows a strong negative correlation with quality. Higher quality wines tend to have lower levels of density. This is probably because of strong correlations between density, fixed acidity, and residual sugar (dense components). Seems to be a weak indicator of quality as it is influenced by other features.

Chlorides shows a negative correlation with quality. Higher quality wines may have lower levels of chlorids. Seems to be a decent indicator of quality.

**Relationships among features**

`Alcohol and Density:`

They show a strong negative correlation. Alcohol and density are inversely related, leading density to be a weak indicator of quality as it will be influenced by alcohol levels.

`Total Sulfur Dioxide and Free Sulfur Dioxide:`

They show a strong positive correlation. These are closely related as free sulfur dioxide is related to total sulfur dioxide.

`Fixed Acidity and Citric Acid:`

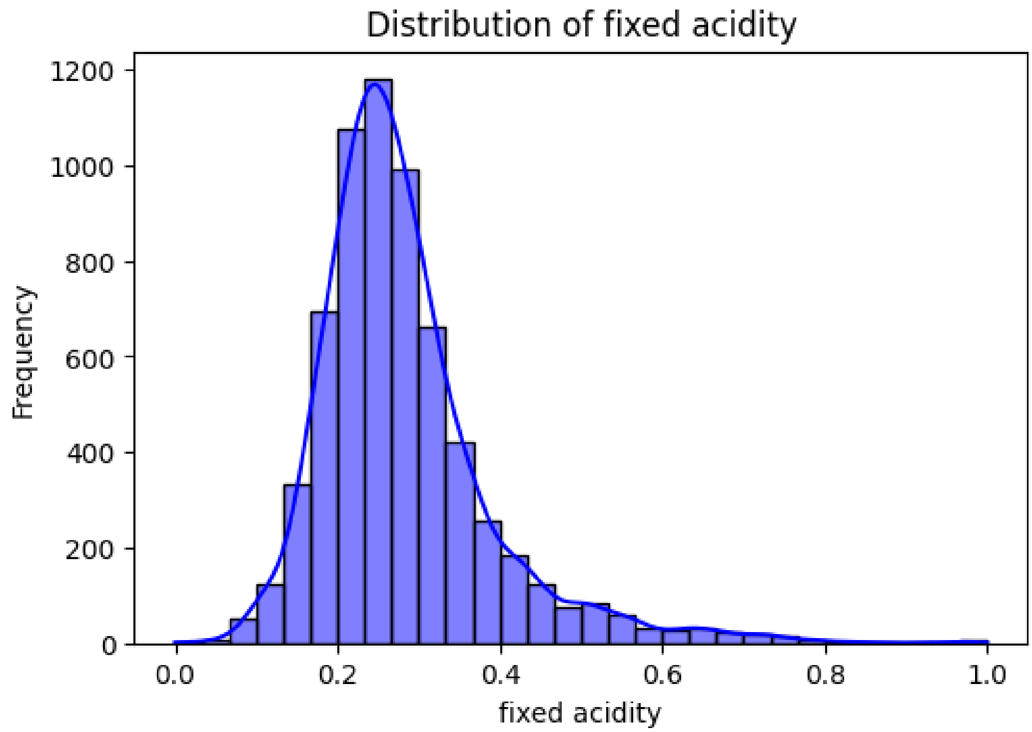
They show a moderate positive correlation. Fixed acidity and citric acid are both acids and are related to each other.

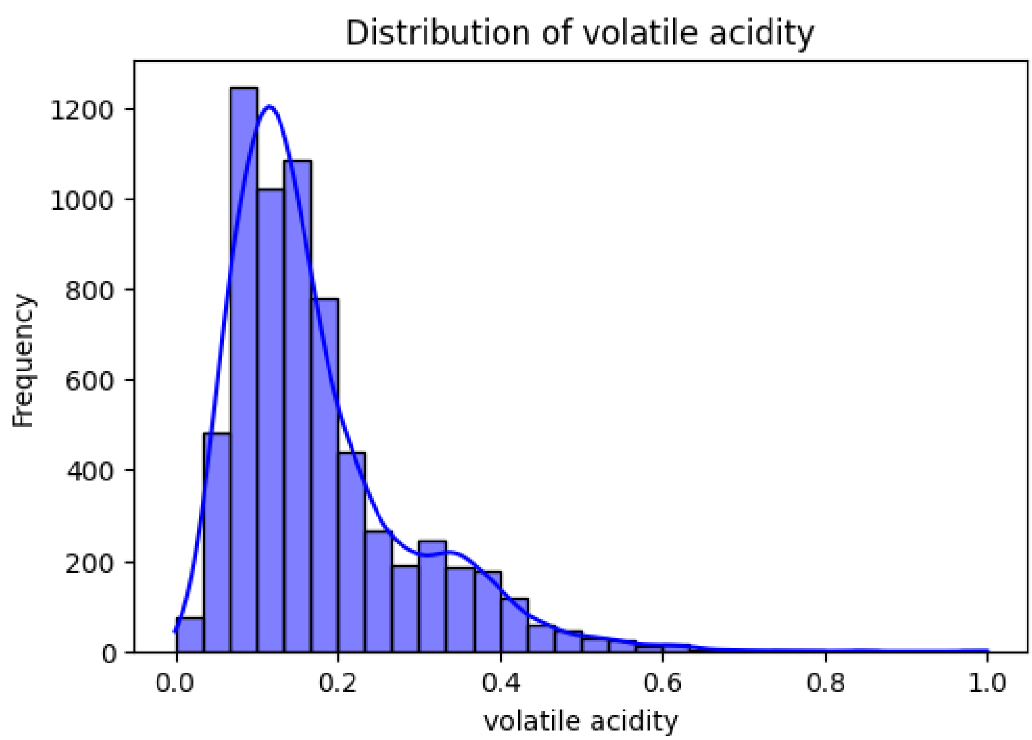
`Residual Sugar and Density:`

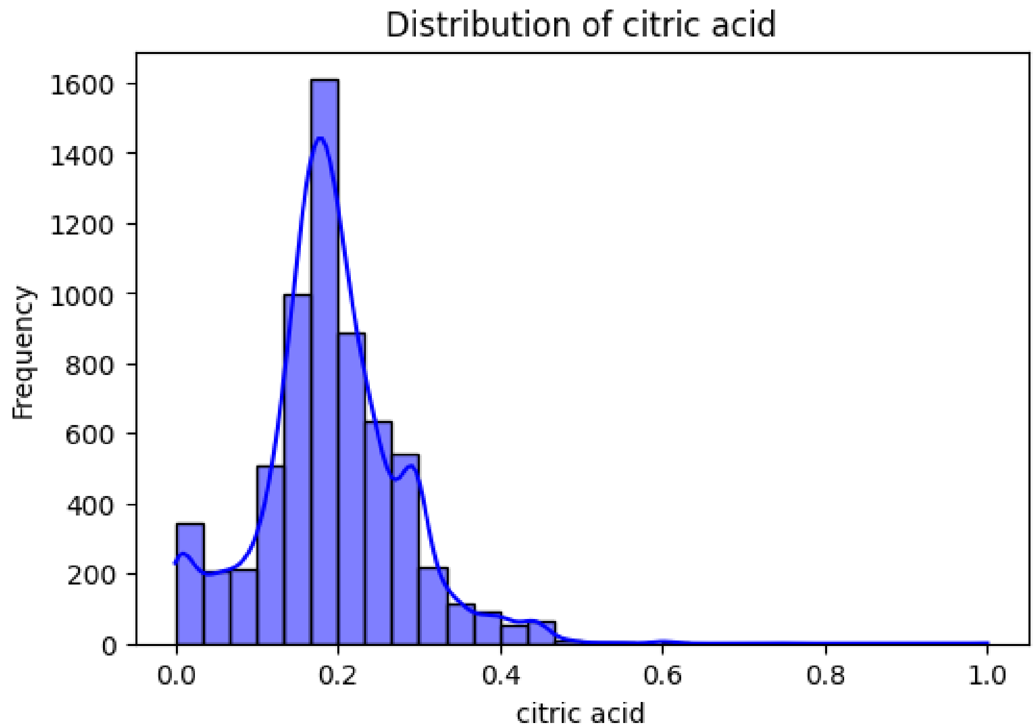
Strong positive correlation. Higher residual sugar increases the wine's density.

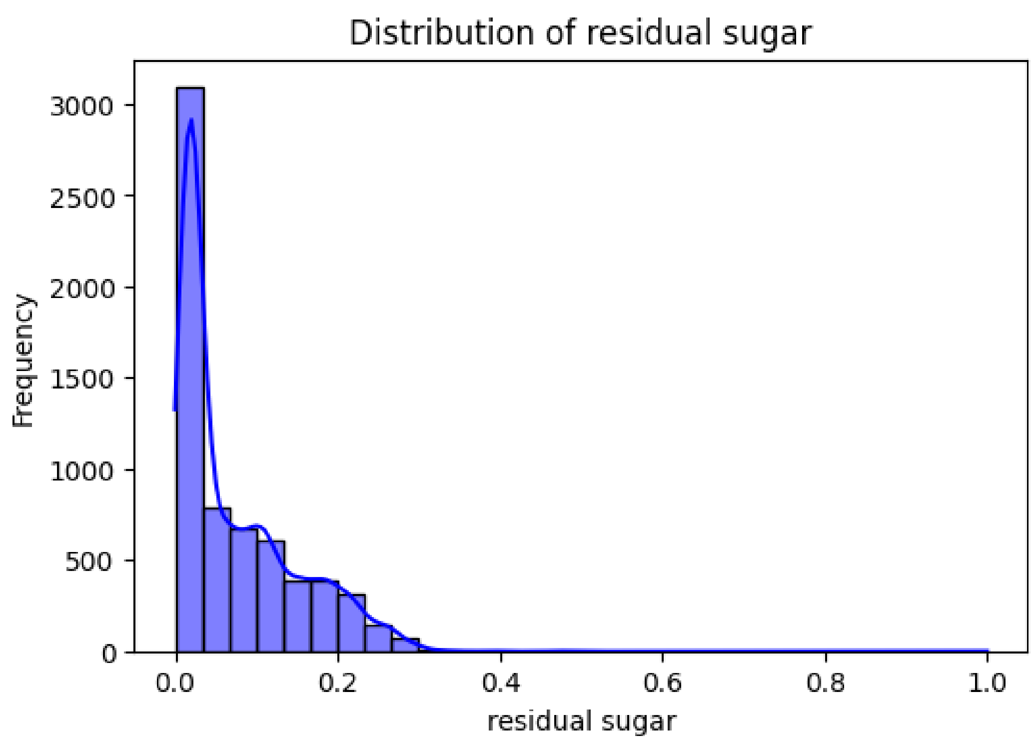
# 6 5. Visualisation

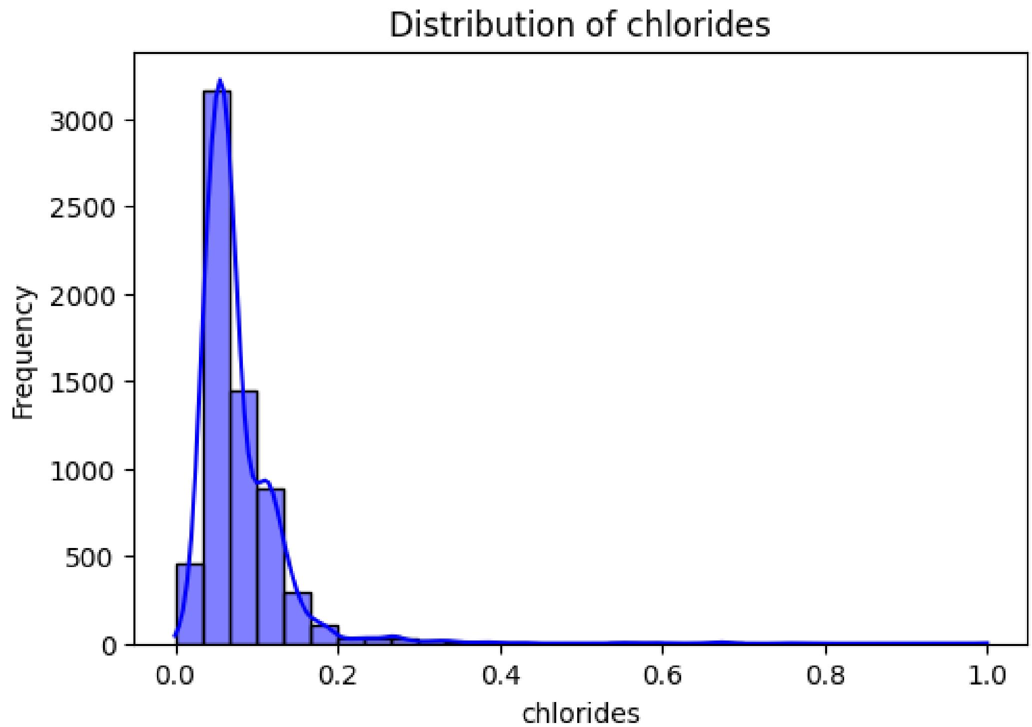
[256]:
```
# analysis of distribution of features
for column in numeric_features:
    plt.figure(figsize=(6, 4))
    sns.histplot(winesdf[column], kde=True, bins=30, color='blue')
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

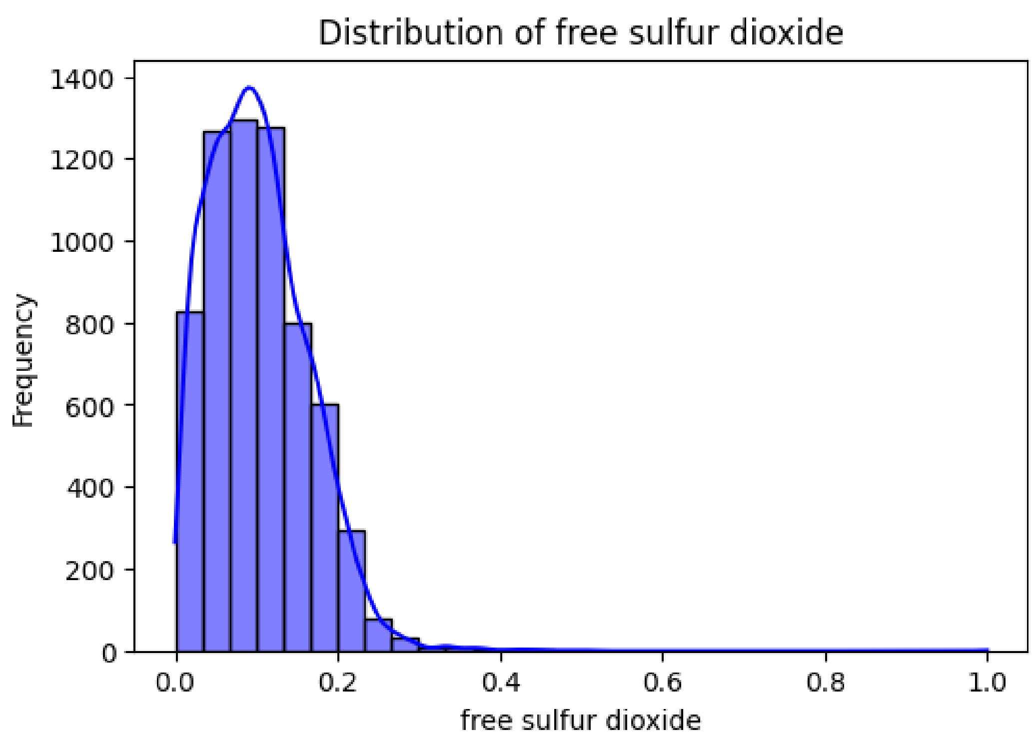

Distribution of fixed acidity
Frequency
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0
fixed acidity

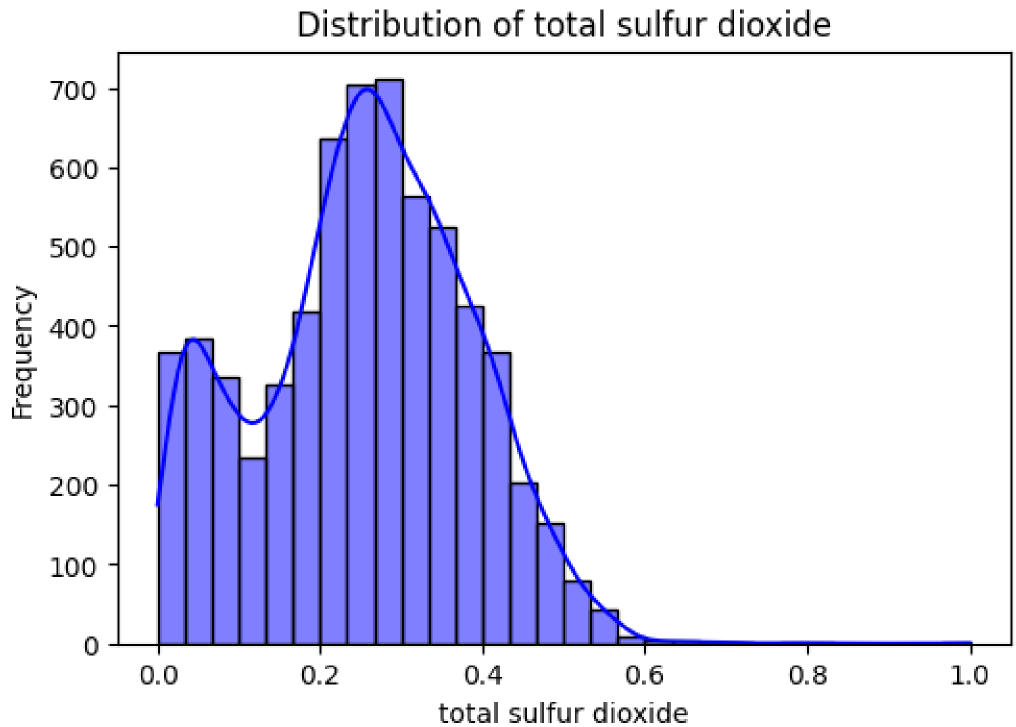

Distribution of volatile acidity
Frequency
0
200
400
600
800
1000
1200
0.0
0.2
0.4
0.6
0.8
1.0
volatile acidity

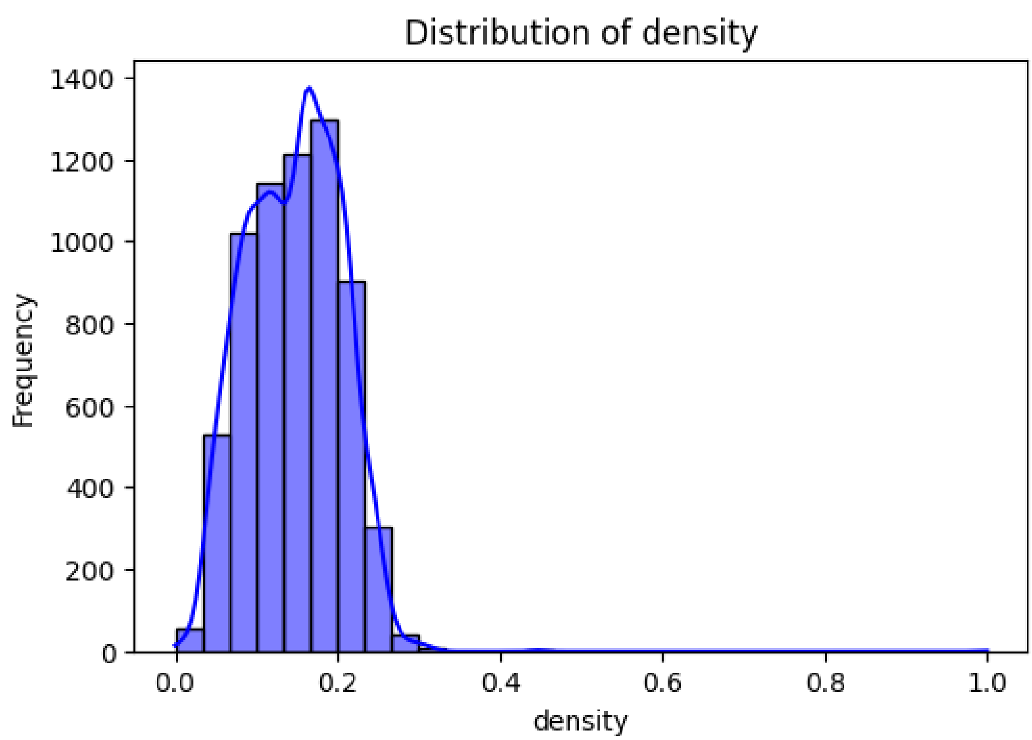

Distribution of citric acid
Frequency
0
200
400
600
800
1000
1200
1400
1600
0.0
0.2
0.4
0.6
0.8
1.0
citric acid

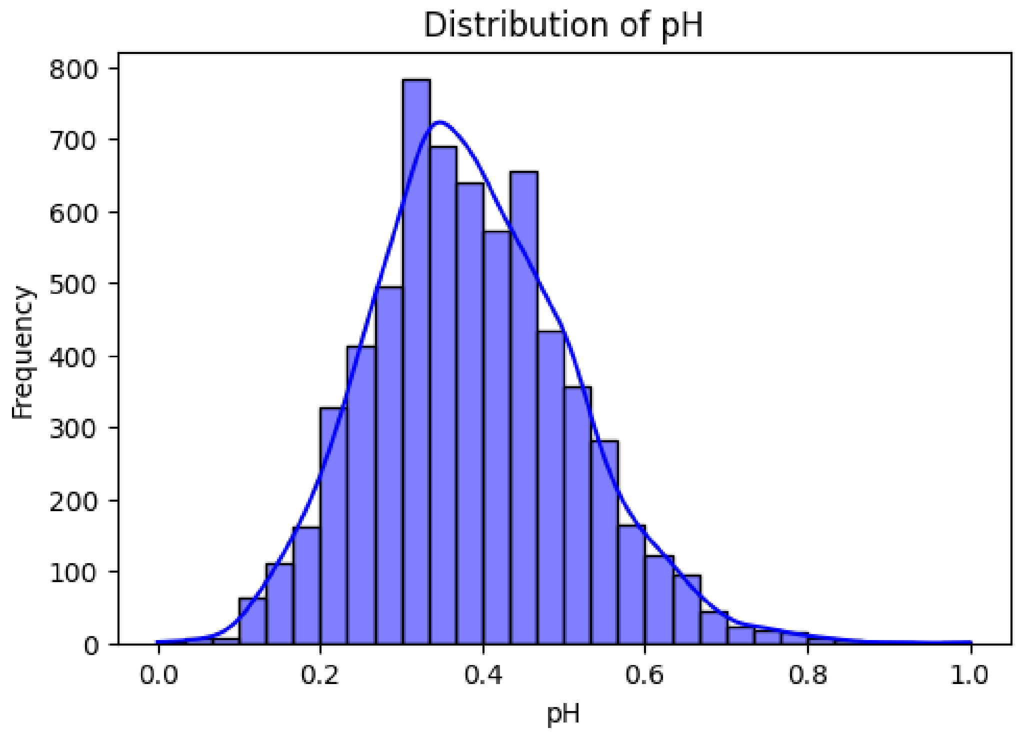

Distribution of residual sugar
Frequency
0
500
1000
1500
2000
2500
3000
0.0
0.2
0.4
0.6
0.8
1.0
residual sugar

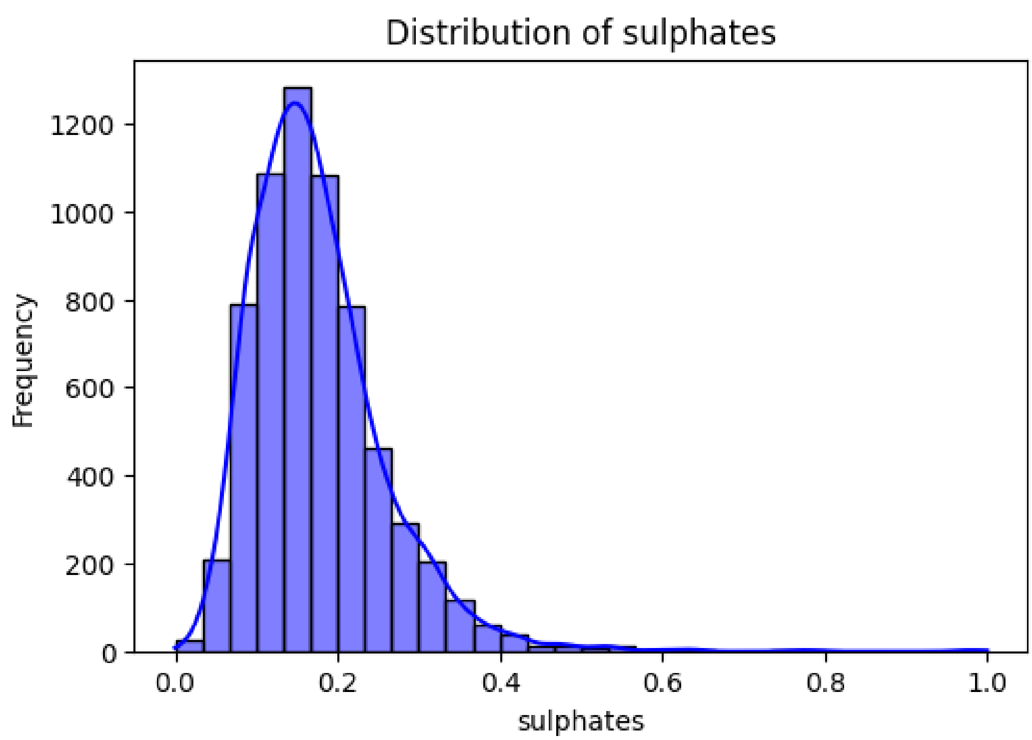

Distribution of chlorides
Frequency
3000
2500
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
chlorides

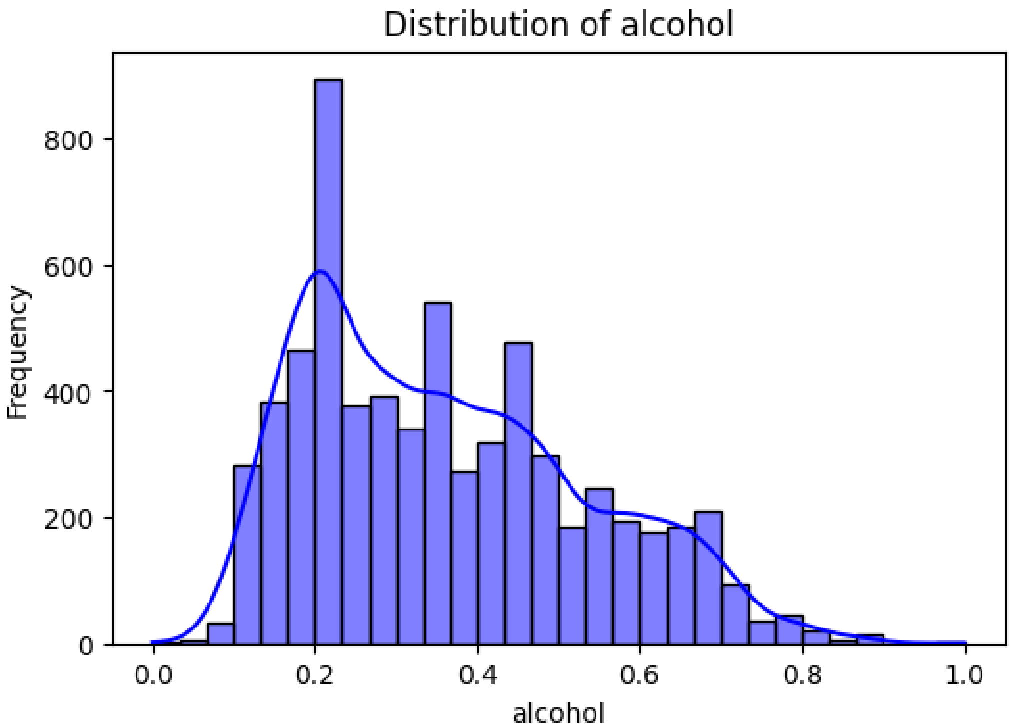

Distribution of free sulfur dioxide
Frequency
1400
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0
free sulfur dioxide


Distribution of total sulfur dioxide
Frequency
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0
total sulfur dioxide


Distribution of density
Frequency
0
200
400
600
800
1000
1200
1400
0.0
0.2
0.4
0.6
0.8
1.0
density

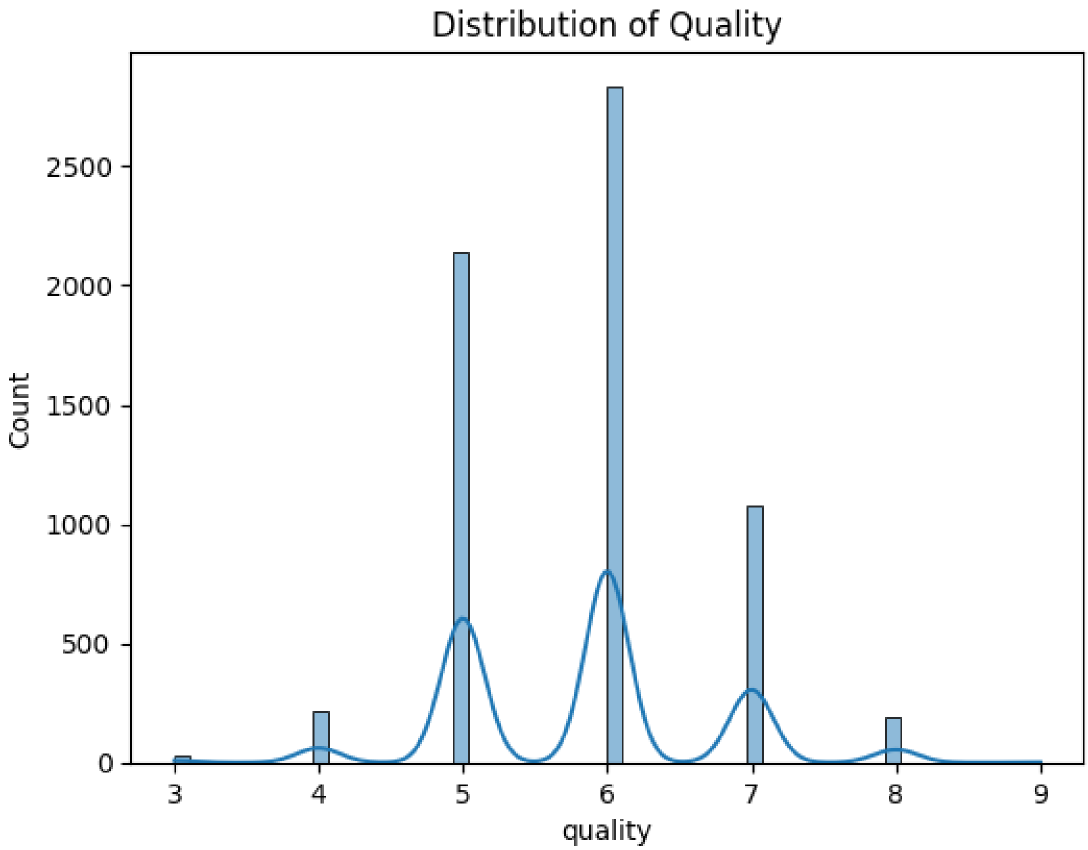

Distribution of pH
800
700
600
500
400
300
200
100
0
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
pH


Distribution of sulphates
1200
1000
800
600
400
200
0
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
sulphates



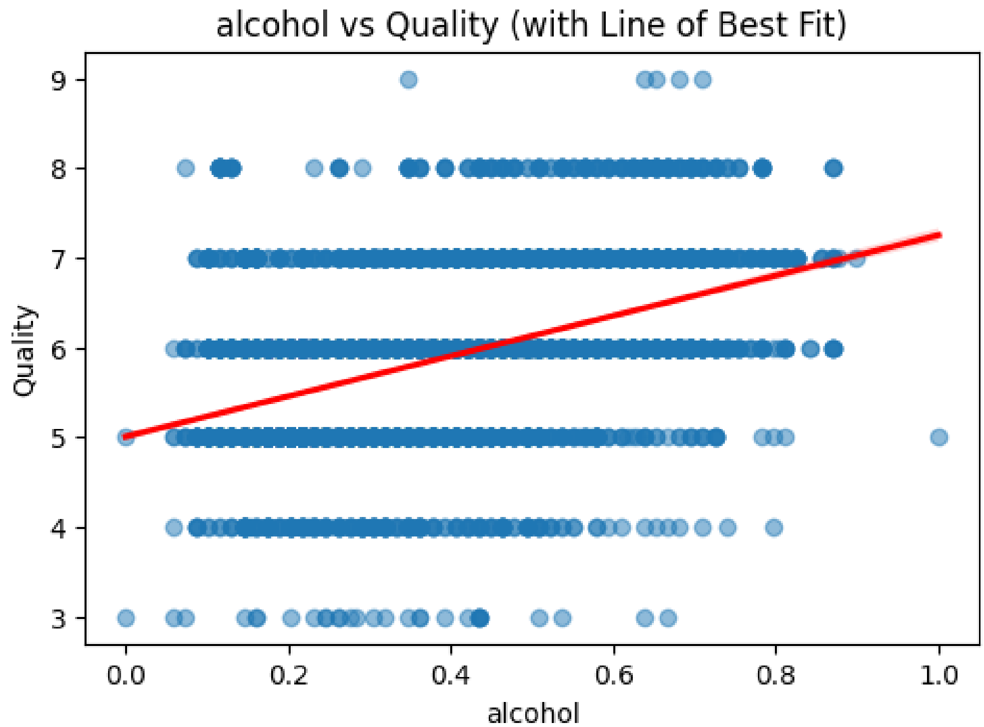

Visualisations match the analyses made earlier under skewness and kurtosis.

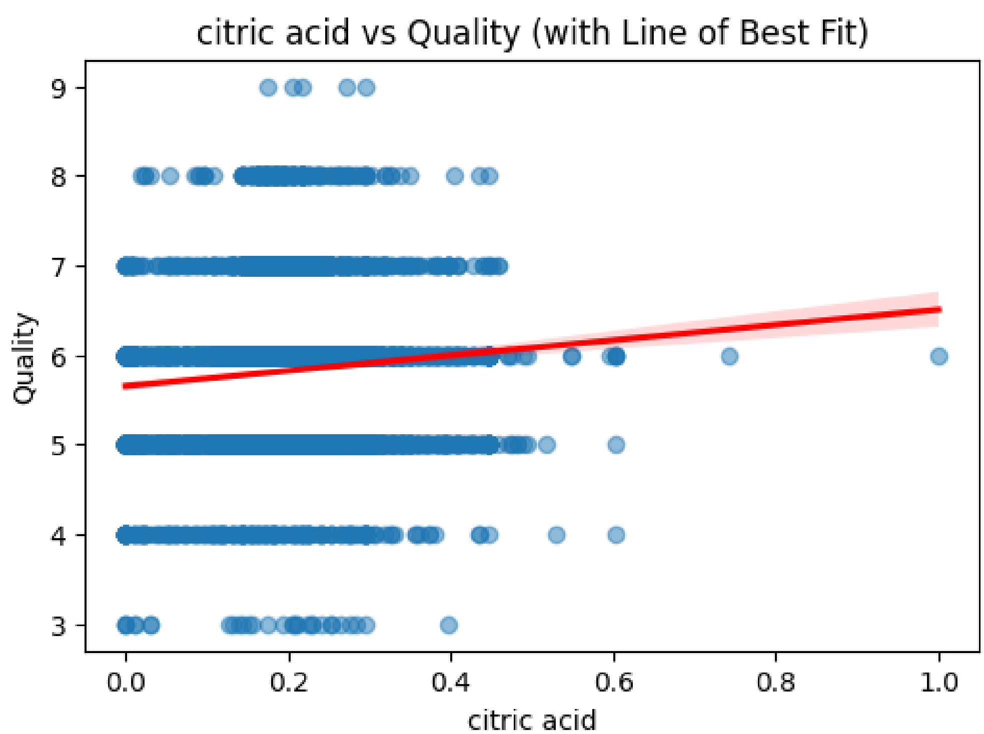
[247]:
```
# distribution of quality
sns.histplot(winesdf['quality'], kde=True)
plt.title("Distribution of Quality")
plt.show()
```

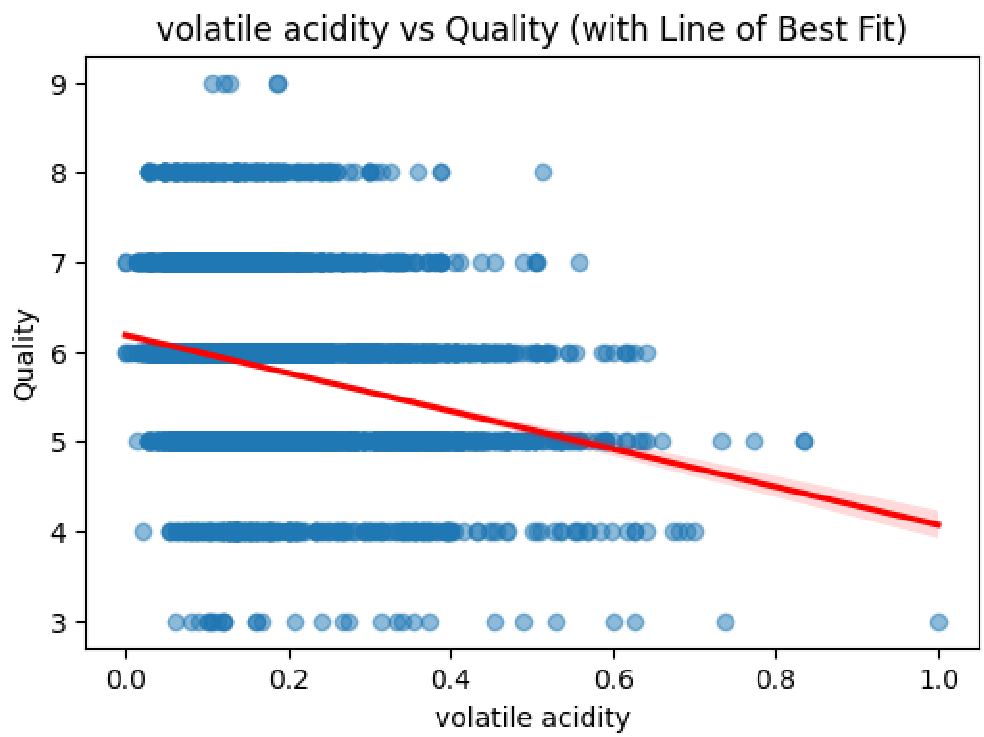


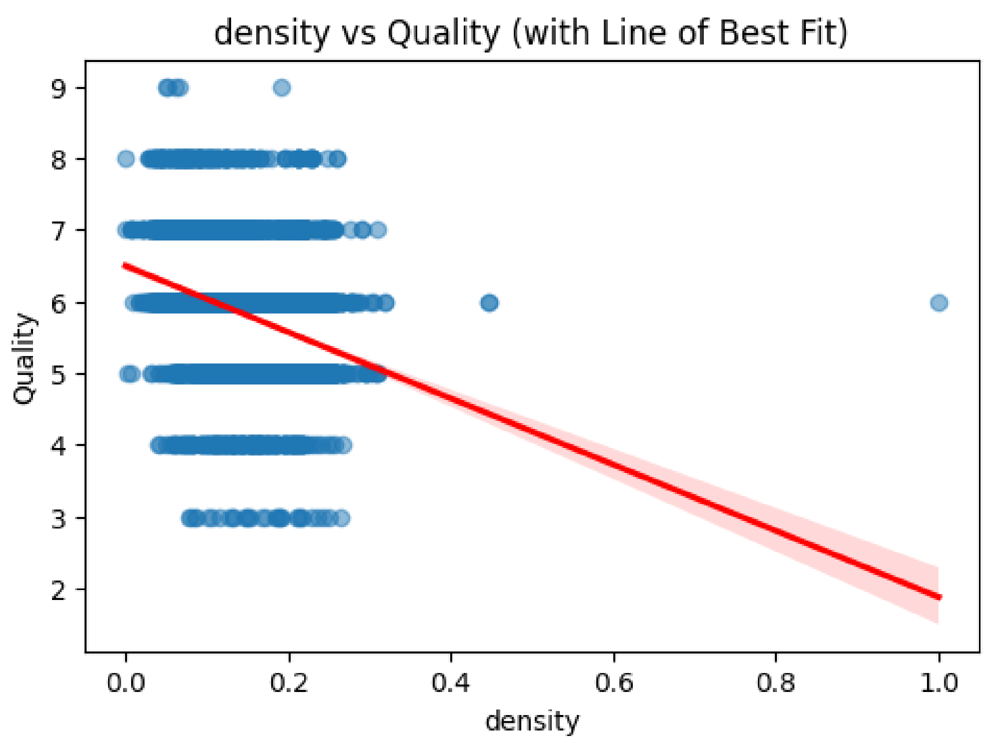

The bulk of the samples are of 5, 6, 7 qualities. While the rest of the quality levels can be seen as outliers. This leads me to believe that the linear regression model trained on this dataset will not be able to make accurate predictions.

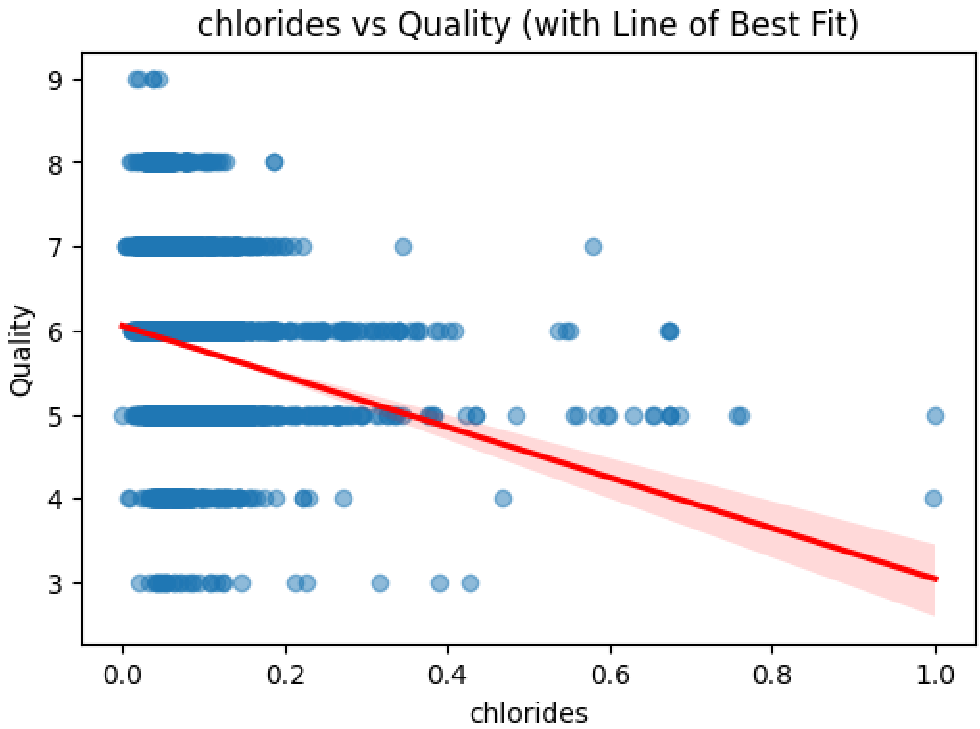
[259]:
```
# scatterplots of key features vs quality

# key features were pointed out with the analysis on the correlation heatmap
key_features = ['alcohol', 'citric acid', 'volatile acidity', 'density',␣
 ↪'chlorides']
for feature in key_features:
    plt.figure(figsize=(6, 4))
    sns.regplot(data=winesdf, x=feature, y='quality',
                scatter_kws={'alpha': 0.5},
                line_kws={'color': 'red'})
    plt.title(f'{feature} vs Quality (with Line of Best Fit)')
    plt.xlabel(feature)
    plt.ylabel('Quality')
    plt.show()
```


alcohol vs Quality (with Line of Best Fit)
Quality
9
8
7
6
5
4
3
0.0
0.2
0.4
0.6
0.8
1.0
alcohol


citric acid vs Quality (with Line of Best Fit)
Quality
9
8
7
6
5
4
3
0.0
0.2
0.4
0.6
0.8
1.0
citric acid


volatile acidity vs Quality (with Line of Best Fit)
Quality
9
8
7
6
5
4
3
0.0
0.2
0.4
0.6
0.8
1.0
volatile acidity


density vs Quality (with Line of Best Fit)
Quality
9
8
7
6
5
4
3
2
0.0
0.2
0.4
0.6
0.8
1.0
density




Visualisations prove our analyses made from the correlation heatmap.

Alcohol vs quality has a positive line of best fit, indicating that, to some extent, higher levels of alcohol do lead to higher quality wine.

Citric acid vs quality also has a positive line of best fit, but the line of best fit is almost flat that it would have very little impact on wine quality.

While the other 3 features show a negative line of best fit, indicating that, to some extent, lower levels of those features tend to lead to higher quality wine.

# 7 6. Building ML model

[236]:
```
from sklearn.model_selection import train_test_split

# building model with all features
X = winesdf.drop(columns=['quality'])
y = winesdf['quality']

# split data to train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[260]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# training model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# making predictions
y_pred = linear_model.predict(X_test)

# evaluating
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100

# display
print(f"MSE: {(mse):.04f}")
print(f"RMSE: {(rmse):.04f}")
print(f"MAE: {(mae):.04f}")
print(f"R-squared: {(r2):.04f}")
print(f"MAPE: {(mape):.04f}%")
```

```
MSE: 0.5467
RMSE: 0.7394
MAE: 0.5659
R-squared: 0.2598
MAPE: 10.0828%
```

# 8 7. Validation

[ ]:
```python
# cross validation
from sklearn.model_selection import cross_val_score

# mse
cv_mse = cross_val_score(linear_model, X_train, y_train, cv=5,␣
↪scoring='neg_mean_squared_error')
# convert negative MSE to positive
mse_scores = -cv_mse

# rmse
rmse_scores = np.sqrt(mse_scores)

# mae
```

```
cv_mae = cross_val_score(linear_model, X_train, y_train, cv=5,␣
 ↪scoring='neg_mean_absolute_error')
# convert negative MAE to positive
mae_scores = -cv_mae

# r2
cv_r2 = cross_val_score(linear_model, X_train, y_train, cv=5, scoring='r2')

# mape
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100

# display
print(f"Average MSE: {np.mean(mse_scores):.4f}")
print(f"Average RMSE: {np.mean(rmse_scores):.4f}")
print(f"Average MAE: {np.mean(mae_scores):.4f}")
print(f"Average R2: {np.mean(cv_r2):.4f}")
print(f"MAPE: {(mape):.04f}%")
```

```
Average MSE: 0.5411
Average RMSE: 0.7355
Average MAE: 0.5697
Average R2: 0.2952
MAPE: 10.0828%
```

From the above evaluation, we can tell that this model is not very good at predicting wine quality. Our mean MSE score is 0.54 and we want this score to be as low as possible, as this tells us the squared differences between predicted and actual values.

As my RMSE score is higher than my MAE score, we can tell that there are large errors due to outliers. We did anticipate this, having analysed the distribution of quality, with the counts of wine qualities 3, 4, 8, and 9 being relatively smaller than the counts of wine qualities 5, 6, and 7.

Our R2 score is also extremely low, with a score of 0.26. We want this score to be as close to 1 as possible, indicating a better fit. The low score suggests that the model is not capturing the relationships in the data.

With about 10% MAPE, we can tell that our model's predictions deviate by around 10% from the actual value, which may be acceptable for most cases in the real world. Especially if this model is only being used to aid wine sommelier's in grading wines, and not to fully decide wine quality scores as the only standard of judgement.

# 9 8. Feature engineering

[ ]:
```
# add polynomial features
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import cross_val_predict

poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)
```

```
# training model with polynomial features
model_poly = LinearRegression()
model_poly.fit(X_poly, y)

# evaluating
cv_poly_scores = cross_val_score(model_poly, X_poly, y, cv=5,␣
↪scoring='neg_mean_squared_error')
cv_poly_mse = -cv_poly_scores
cv_poly_rmse = np.sqrt(cv_poly_mse)
cv_poly_mae = -cross_val_score(model_poly, X_poly, y, cv=5,␣
↪scoring='neg_mean_absolute_error')
cv_poly_r2 = cross_val_score(model_poly, X_poly, y, cv=5, scoring='r2')

# cross-validated predictions for MAPE
y_pred_cv = cross_val_predict(model_poly, X_poly, y, cv=5)
mape = np.mean(np.abs((y - y_pred_cv) / y)) * 100

# display
print(f"Cross-validated MSE: {np.mean(cv_poly_mse):.4f}")
print(f"Cross-validated RMSE: {np.mean(cv_poly_rmse):.4f} (+/- {np.
↪std(cv_poly_rmse):.4f})")
print(f"Cross-validated MAE: {np.mean(cv_poly_mae):.4f}")
print(f"Cross-validated R²: {np.mean(cv_poly_r2):.4f}")
print(f"MAPE: {mape:.2f}%")
```

```
Cross-validated MSE: 0.5828
Cross-validated RMSE: 0.7633 (+/- 0.0145)
Cross-validated MAE: 0.5852
Cross-validated R²: 0.2143
MAPE: 10.39%
```

# 10 10. evaluation of model

Final performance metrics of the model: - MSE: 0.5828 - RMSE: 0.7633 (with a standard deviation of 0.0145) - MAE: 0.5852 - R2: 0.2143 - MAPE: 10.39%

Why the use of such metrics? 1. MSE - useful for detecting significant deviations in predictions and is easy to optimise during model training.

2. RMSE
   - penalises large prediction errors, ensuring that significant deviations in wine quality predictions are minimised.
3. MAE
   - is robust and gives an equal weight to all errors.
4. R2
   - determines the proportion of variance in the target variable eplained by the features, indicates goodness of fit.

5. MAPE
    - helps interpret model performance in relative terms as it is percentage based. Useful for understanding prediction error in proportion to actual target values.

Although the model's metrics has had minimal improvement throughout the project, we were able to find out which of the features were the most influential. The key features discovered are, alcohol, citric acid, volatile acidity, density, and chlorides. These features were found to be the most influential out of the rest of the features, be it inverse or not. These insights could help the industry's winemakers optimise wine production processes based on these data-driven findings. Although there is a mean deviation of about 10%, the model can still be used to aid wine sommelier's in their task to grade wines as a first degree of classification or sorting. This could reduce the time needed for the process of wine grading and tasting. The model could also be used to automate quality assessment systems in wineries.

Some limitations of this analysis, 1. Unbalanced dataset - The dataset's major bulk of quality values were in the 5, 6, and 7 range. This, added with the lack of samples with quality values in the 3, 4, 8, and 9 ranges, made it harder for the model to predict wines in the extreme ranges. Having a balanced dataset was necessary for this project's linear regression model to perform accurately.

2. Combined dataset
    - As the dataset was a combination of red and white wines, this leads me to believe that there could be greater correlation between features or accuracy of the model if it was trained on either red or white wine. There would be specific clusters in certain features belonging to either type of wine, lowering the influence and linearity of those features.

The developed model or steps taken to create the model can be applied to predicting the quality or researching on feature strength in other beverages such as beer or coffee. As I mentioned at the start of this notebook, linear regression has already been successfully applied to similar domains such as coffee bean quality. This is possible as the other beverages such as beer or coffee would also contain similar physicochemical properties such as alcohol content, pH level, acidity levels, residual sugar, etc.