8_wine_quality_prediction

吴清柳

July 31, 2023

1 红酒质量预测

2 数据集字段

- 固定酸度: 固定酸度对应于一组低挥发性有机酸，例如苹果酸、乳酸、酒石酸或柠檬酸，并且是样品特性所固有的。
- 挥发性酸度：挥发性酸度是一个重要的感官参数，含量越高表明葡萄酒腐败。
- 柠檬酸：柠檬酸通常添加到葡萄酒中以增加酸度、补充特定风味或防止铁雾。它可以添加到成品葡萄酒中以增加酸度并赋予新鲜的风味。
- 残糖：残糖是酒精发酵完成后葡萄酒中残留的天然葡萄糖。
- 氯化物：它们是葡萄酒咸味的主要来源。
- 游离二氧化硫：二氧化硫 (SO2) 可以保存葡萄酒，防止氧化和褐变。
- 总二氧化硫：总二氧化硫 (TSO2) 是葡萄酒中游离 SO2 的部分加上与葡萄酒中其他化学物质（如醛、色素或糖）结合的部分。
- 密度：保持酒精含量恒定，密度对葡萄酒的质量影响很小，因为其他因素也会影响密度。
- pH 值：酿酒师使用 pH 值来衡量与酸度相关的成熟度。低 pH 值的葡萄酒尝起来又酸又脆，而高 pH 值的葡萄酒更容易受到细菌生长的影响。
- 硫酸盐：另一种硫酸盐的存在被认为有助于去除葡萄酒中的多种细菌（好细菌和坏细菌）。这似乎也会降低葡萄酒的质量，因为它会减弱葡萄酒的发酵过程。
- 酒精：酒精含量会影响葡萄酒的酒体，因为酒精比水更粘稠。酒精含量较高的葡萄酒，酒体会更加饱满、丰富，而酒精含量较低的葡萄酒，口感会更加清淡、细腻。
- 品质：葡萄酒的品质主要取决于酿造过程和葡萄的地理原产地，但也很大程度上取决于葡萄的品种组成。

[84]:
```
# Import libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.offsetbox import AnnotationBbox, OffsetImage
import matplotlib.image as mpimg

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import (
    RandomForestClassifier,
    GradientBoostingClassifier,
    AdaBoostClassifier,
)

from catboost import CatBoostClassifier
from xgboost import XGBClassifier

import lightgbm as lgbm

import sklearn.metrics as metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score

import optuna

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras import Input
from tensorflow.keras.layers import Dense,Dropout,Conv2D,Flatten,MaxPooling2D
from tensorflow.keras.optimizers import Adam
from keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.utils import plot_model

import warnings
warnings.filterwarnings('ignore')
```

[58]:
```
# Set a color scheme
custom_colors=['#a70000','#cc0001','#ff0000','#ff5252','#ff7b7b','#ffbaba']
custom_palette=sns.set_palette(sns.color_palette(custom_colors))
sns.palplot(sns.color_palette(custom_colors),size=1)
plt.tick_params(axis='both',labelsize=0,length=0)
```

[59]:
```
# Reading the data
df=pd.read_csv('../dataset/winequality-red.csv')
```

```
df.head()
```

[59]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

[60]:
```
# Taking a look at the missing values of the dataset
print(df.isna().sum())
print('-'*10)
print('Total Missing Values = {}'.format(df.isna().sum().sum()))
print('-'*10)
```

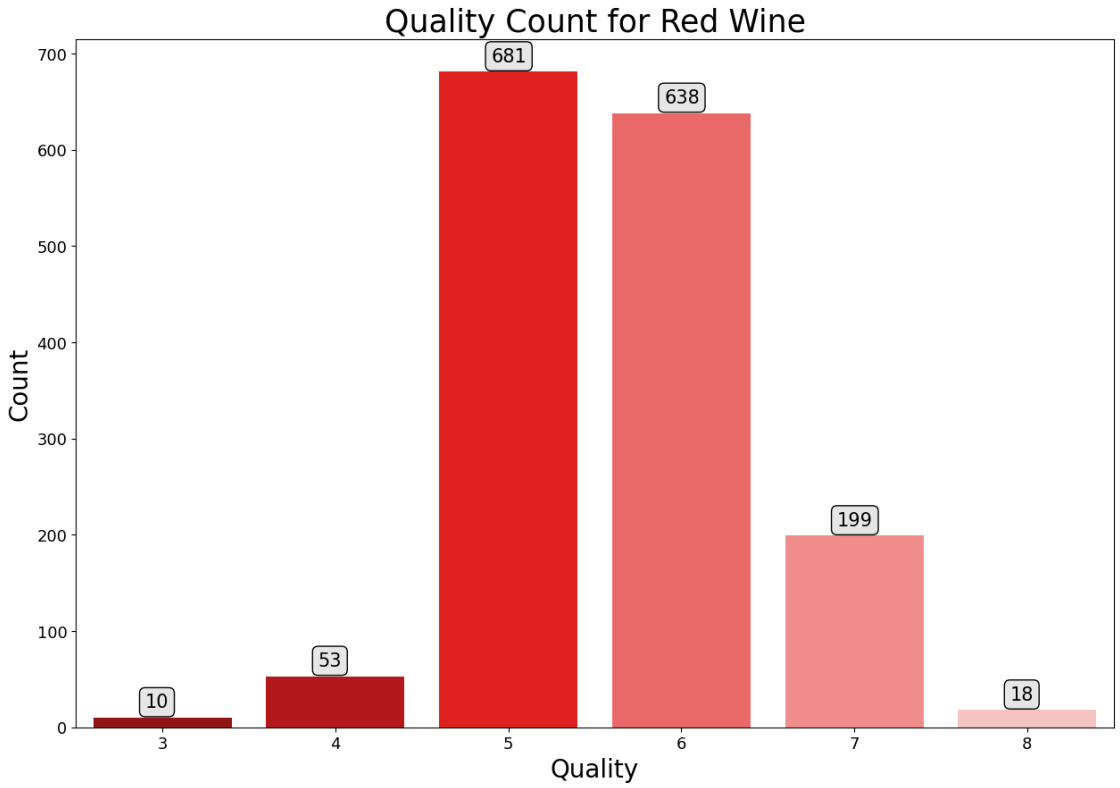
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
----------
Total Missing Values = 0
----------
```

[61]:
```
# Taking a look at the statistical summary of the dataset
summary = pd.DataFrame(df.describe())
summary = (
    summary.style.background_gradient(cmap="Reds")
    .set_table_attributes('style="display: inline"')
    .set_caption("Statistics of the Dataset")
    .set_table_styles(
        [{"selector": "caption", "props": [("font-size", "16px")]}]
    )
)
summary
```

[61]:
```
<pandas.io.formats.style.Styler at 0x7f370a520640>
```

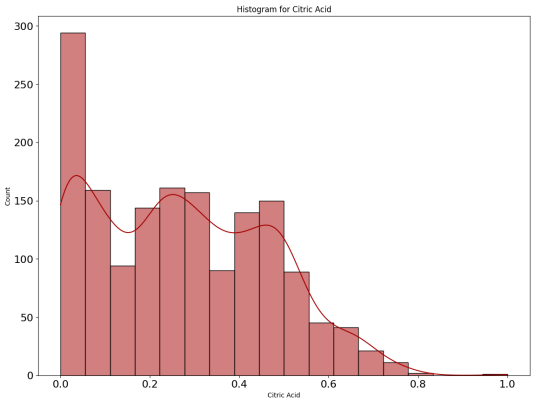
3 探索性数据分析 (EDA)

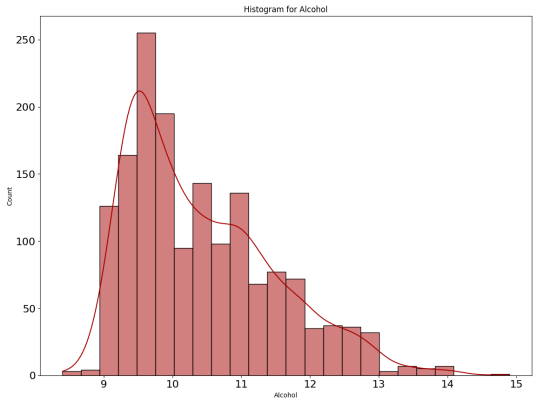
[62]:
```
plt.figure(figsize=(15, 10))
ax = sns.countplot(
    data=df,
    x="quality",
    palette=[
        custom_colors[0],
        custom_colors[1],
        custom_colors[2],
        custom_colors[3],
        custom_colors[4],
        custom_colors[5],
    ],
)

bbox_args = dict(boxstyle="round", fc="0.9")
for p in ax.patches:
    ax.annotate(
        "{:.0f}".format(p.get_height()),
        (p.get_x() + 0.3, p.get_height() + 10.5),
        color="black",
        bbox=bbox_args,
        fontsize=15,
    )

plt.title("Quality Count for Red Wine", fontsize=25)
plt.xlabel("Quality", fontsize=20)
plt.ylabel("Count", fontsize=20)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.show()
```



观察: 质量为 **5** 的红酒是最多的, 质量为 **3** 的红酒是最少的. 在数据集中有**类不平衡**, 在开始训练机器学习模型之前需要对其进行修复

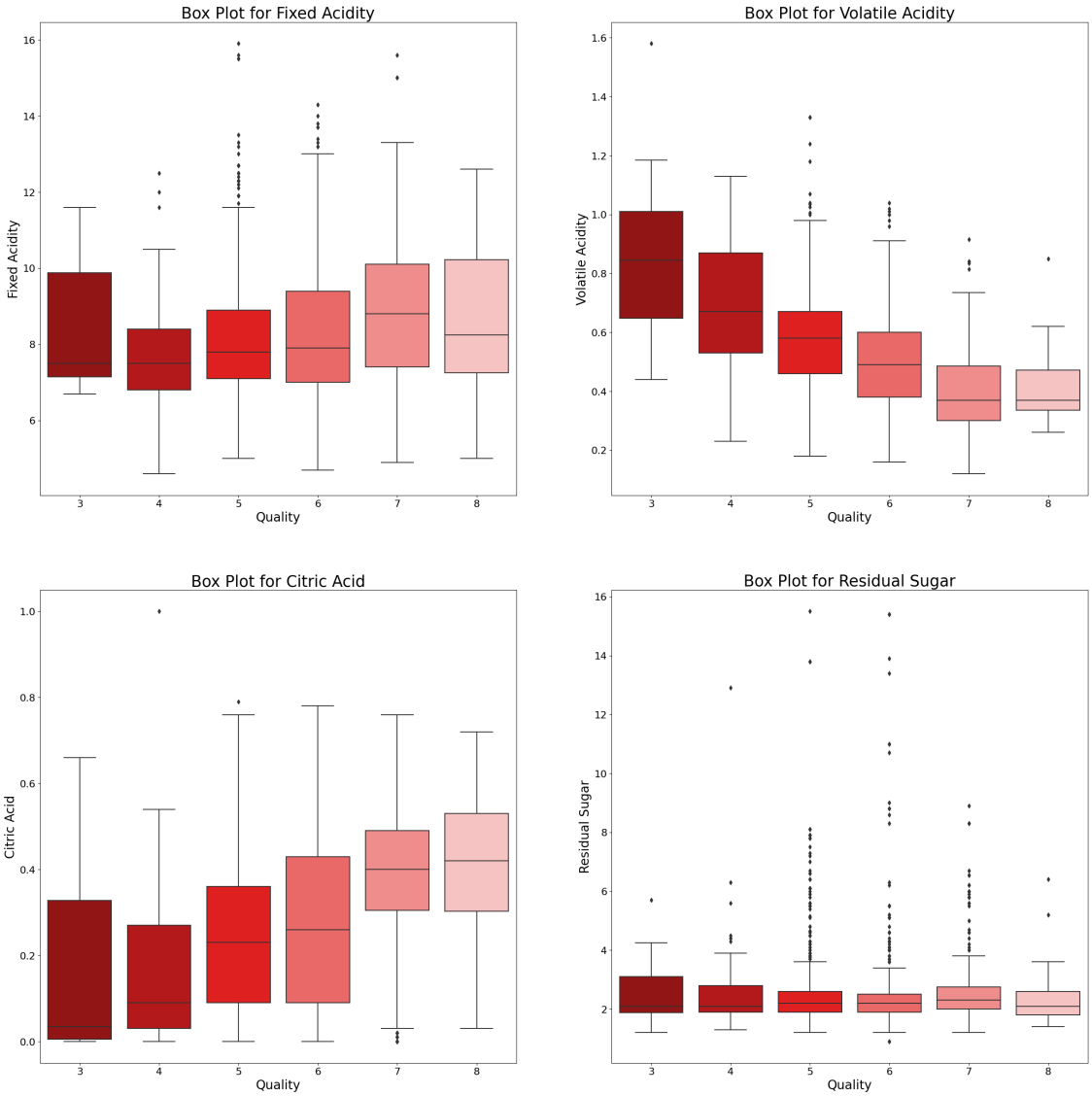
[63]:
```
fig, axes = plt.subplots(ncols=2, figsize=(30, 10))

sns.histplot(data=df, x="citric acid", kde=True, ax=axes[0])
axes[0].set_title("Histogram for Citric Acid")
axes[0].set_xlabel("Citric Acid")
axes[0].set_ylabel("Count")
axes[0].xaxis.set_tick_params(labelsize=16)
axes[0].yaxis.set_tick_params(labelsize=16)

sns.histplot(data=df, x="alcohol", kde=True, ax=axes[1])
axes[1].set_title("Histogram for Alcohol")
axes[1].set_xlabel("Alcohol")
axes[1].set_ylabel("Count")
axes[1].xaxis.set_tick_params(labelsize=16)
axes[1].yaxis.set_tick_params(labelsize=16)

plt.show()
```





观察: 绘制柠檬酸和酒精的直方图以查看其分布的偏度。酒精直方图呈现右偏态，其中众数 $<$ 中值 $<$ 平均值。

[64]:
```
# BoxPlots
plt.figure(figsize=(30,30))

def create_boxplot(feature):
    sns.boxplot(data=df,x=df['quality'],y=feature)
    plt.title('Box Plot for '+feature.title(),fontsize=25)
    plt.xlabel('Quality',fontsize=20)
    plt.ylabel(feature.title(),fontsize=20)
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)

plt.subplot(221)
create_boxplot('fixed acidity')

plt.subplot(222)
create_boxplot('volatile acidity')

plt.subplot(223)
create_boxplot('citric acid')

plt.subplot(224)
create_boxplot('residual sugar')
```

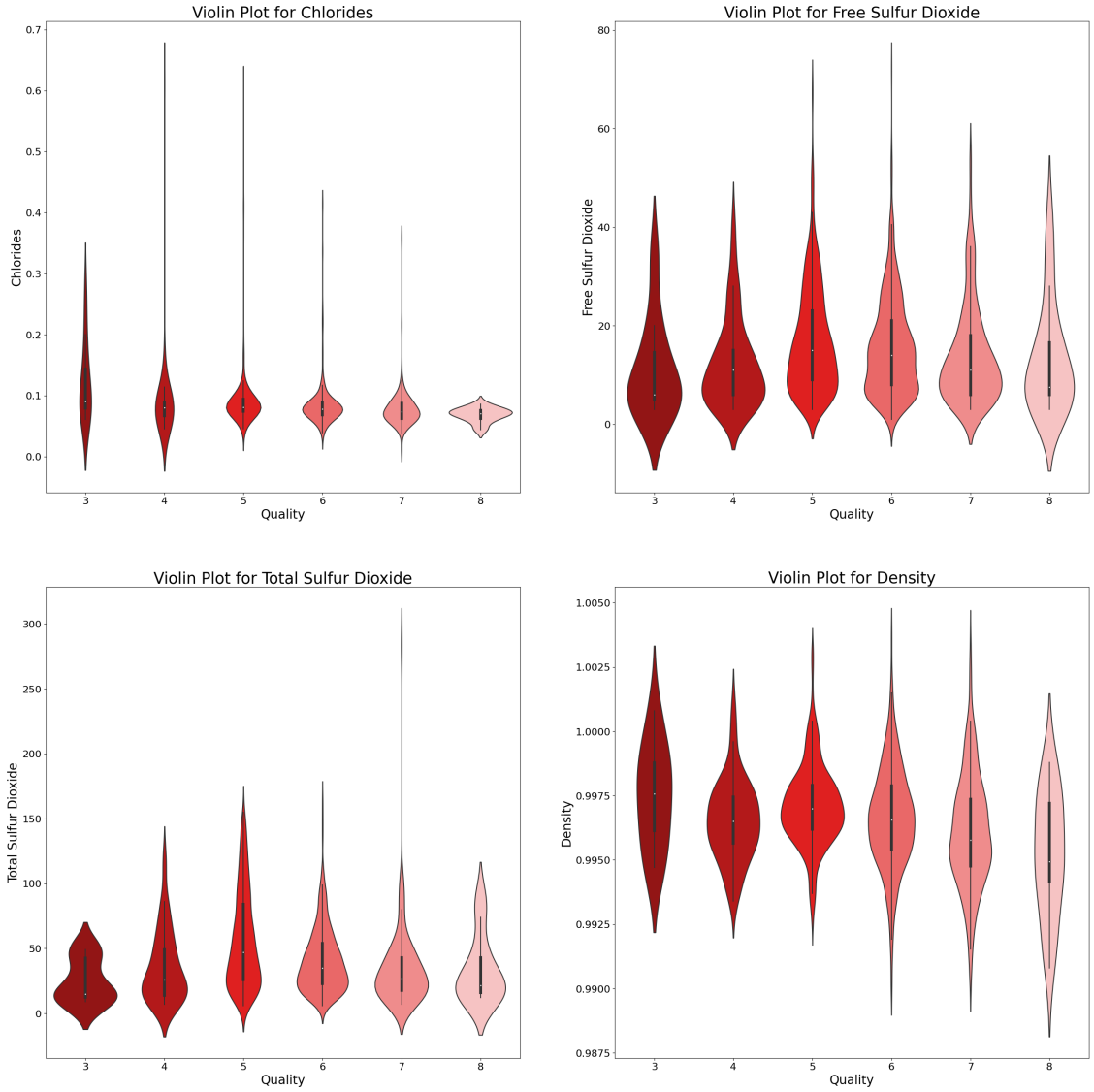


观察:

- 固定酸度：对于不同品质的葡萄酒，固定酸度的箱线图具有大致相同的中值。质量为 5 的葡萄酒的异常值最高。
- 挥发酸度：随着葡萄酒质量的提高，我们可以观察到葡萄酒挥发酸度的中值下降。
- 柠檬酸：随着葡萄酒质量的提高，我们可以观察到葡萄酒柠檬酸的中值增加。这与我们在分析葡萄酒的挥发酸度时获得的观察结果完全相反。
- 残糖：不同品质的葡萄酒，残糖的中值几乎相同。质量为 5 或 6 的葡萄酒的异常值数量最多。

[65]:
```
# Violin plots
plt.figure(figsize=(30,30))

def create_violinplot(feature):
    sns.violinplot(data=df,x=df['quality'],y=feature)
```

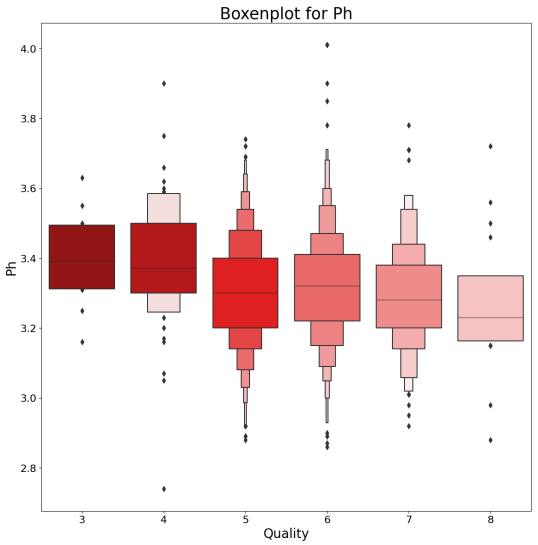
```
    plt.title('Violin Plot for '+feature.title(),fontsize=25)
    plt.xlabel('Quality',fontsize=20)
    plt.ylabel(feature.title(),fontsize=20)
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)

plt.subplot(221)
create_violinplot('chlorides')

plt.subplot(222)
create_violinplot('free sulfur dioxide')

plt.subplot(223)
create_violinplot('total sulfur dioxide')

plt.subplot(224)
create_violinplot('density')
```

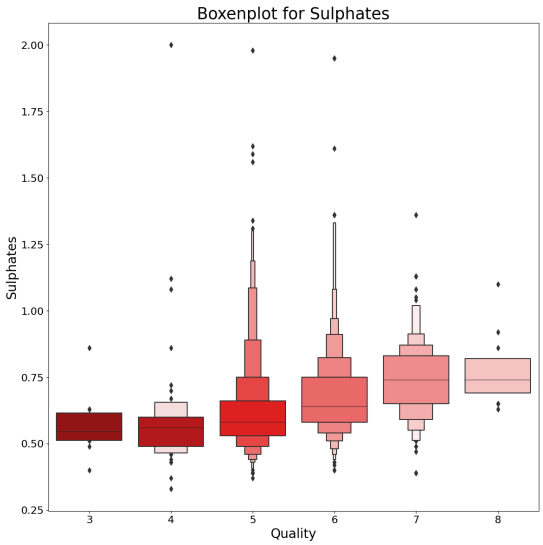


观察 - 氯化物：不同类型葡萄酒品质的氯化物中值是相同的。- 游离二氧化硫：品质为 5 的葡萄酒中游离二氧化硫的中值最高。- 总二氧化硫：总二氧化硫 IQR 最高的是品质为 5 的葡萄酒。- 密度：品质为 3 的葡萄酒具有最高的密度中值。

[66]:
```
# boxen plots
plt.figure(figsize=(30, 30))


def create_boxenplot(feature):
    sns.boxenplot(data=df, x=df["quality"], y=feature)
    plt.title("Boxenplot for " + feature.title(), fontsize=25)
    plt.xlabel("Quality", fontsize=20)
    plt.ylabel(feature.title(), fontsize=20)
```

```
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)

plt.subplot(221)
create_boxenplot('pH')

plt.subplot(222)
create_boxenplot('sulphates')
```

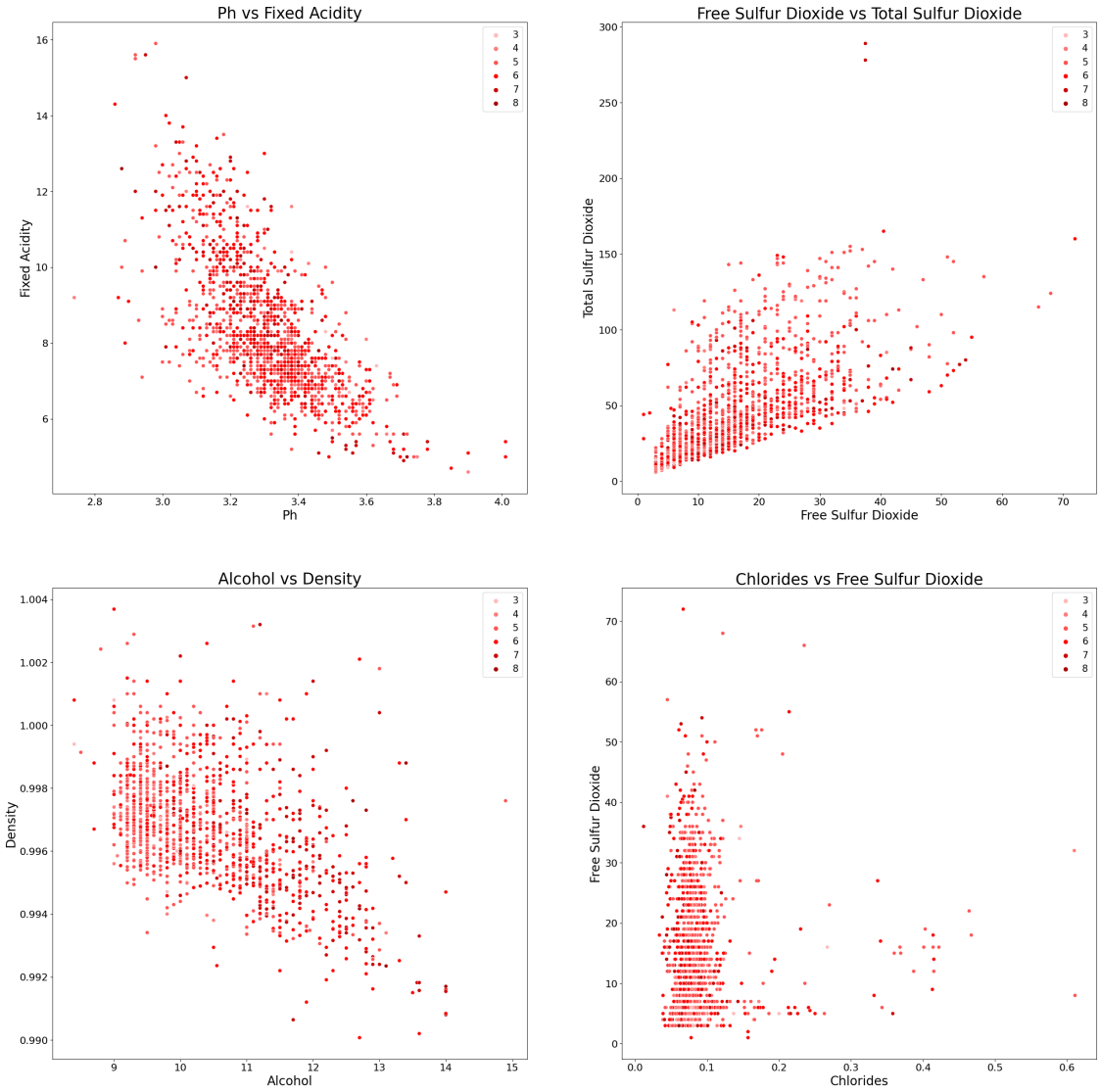




观察 - pH: pH 的中位数随着品质上升而下降. 具有质量 5 和 6 的红酒相比其他有更长的头和尾 - 硫酸盐: 随着品质上升, 硫酸盐含量上升. 和 pH 类似, 品质为 5 和 6 的红酒有更长的头和尾

[67]:
```
# scatter plots
plt.figure(figsize=(30, 30))


def create_scatterplot(feature1, feature2):
    sns.scatterplot(
        data=df,
        x=feature1,
        y=feature2,
        hue=df["quality"],
        palette=[
            custom_colors[-1],
            custom_colors[-2],
            custom_colors[-3],
            custom_colors[-4],
            custom_colors[-5],
            custom_colors[-6],
```

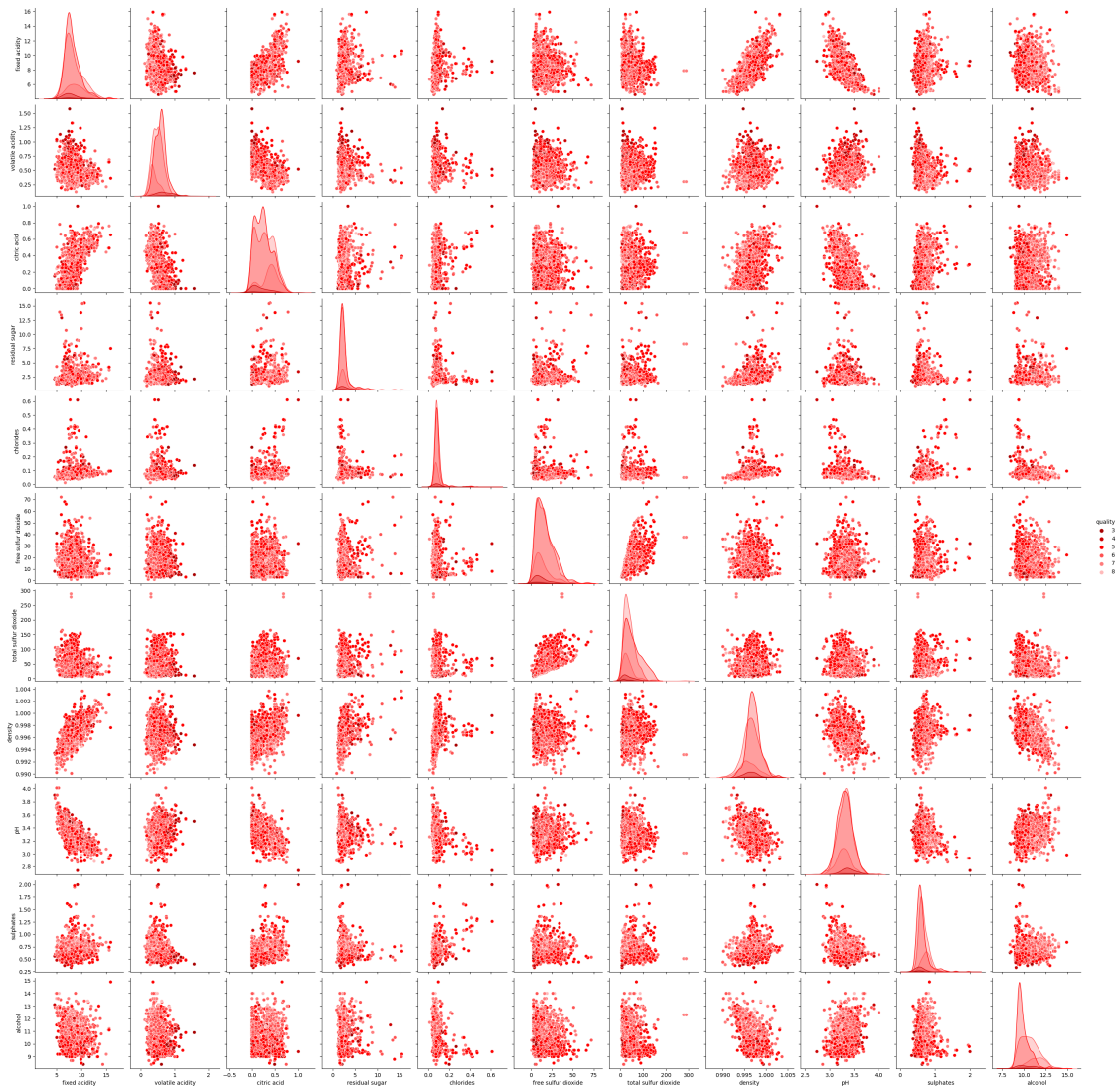
```
        ],
    )
    plt.title(feature1.title()+' vs '+feature2.title(),fontsize=25)
    plt.legend(fontsize=15)
    plt.xlabel(feature1.title(),fontsize=20)
    plt.ylabel(feature2.title(),fontsize=20)
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)

plt.subplot(221)
create_scatterplot('pH','fixed acidity')

plt.subplot(222)
create_scatterplot('free sulfur dioxide','total sulfur dioxide')

plt.subplot(223)
create_scatterplot('alcohol','density')

plt.subplot(224)
create_scatterplot('chlorides','free sulfur dioxide')

plt.show()
```



观察 - pH vs 固定酸度：pH 和固定酸度之间有强负相关性 - 游离二氧化硫 vs 总二氧化硫：有一定的正相关性 - 酒精 vs 密度：负相关性 - 氯化物 vs 二氧化硫：没有可以分辨的相关性

[68]:
```
# Pair plots
sns.pairplot(
    data=df,
    hue="quality",
    palette=[
        custom_colors[0],
        custom_colors[1],
        custom_colors[2],
        custom_colors[3],
        custom_colors[4],
```

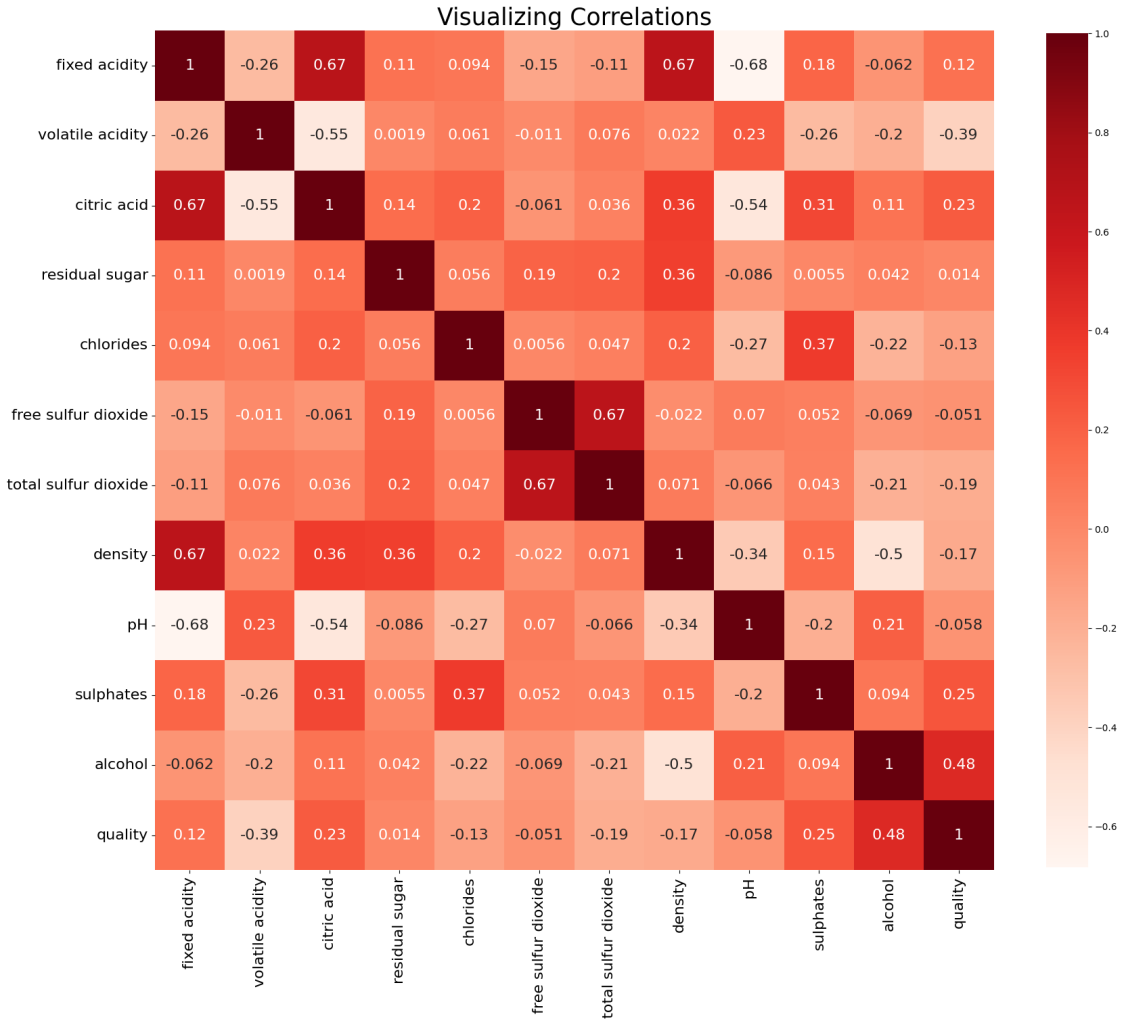
```
        custom_colors[5],
    ],
)
plt.show()
```



4 检查相关性

[69]:
```
# correlation matrix
plt.figure(figsize=(20, 20))
sns.heatmap(
    df.corr(),
    cmap="Reds",
    square=True,
```

```
    annot=True,
    annot_kws={"size": 16},
    cbar_kws={"shrink": 0.80},
)
plt.title('Visualizing Correlations',size=25)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.show()
```

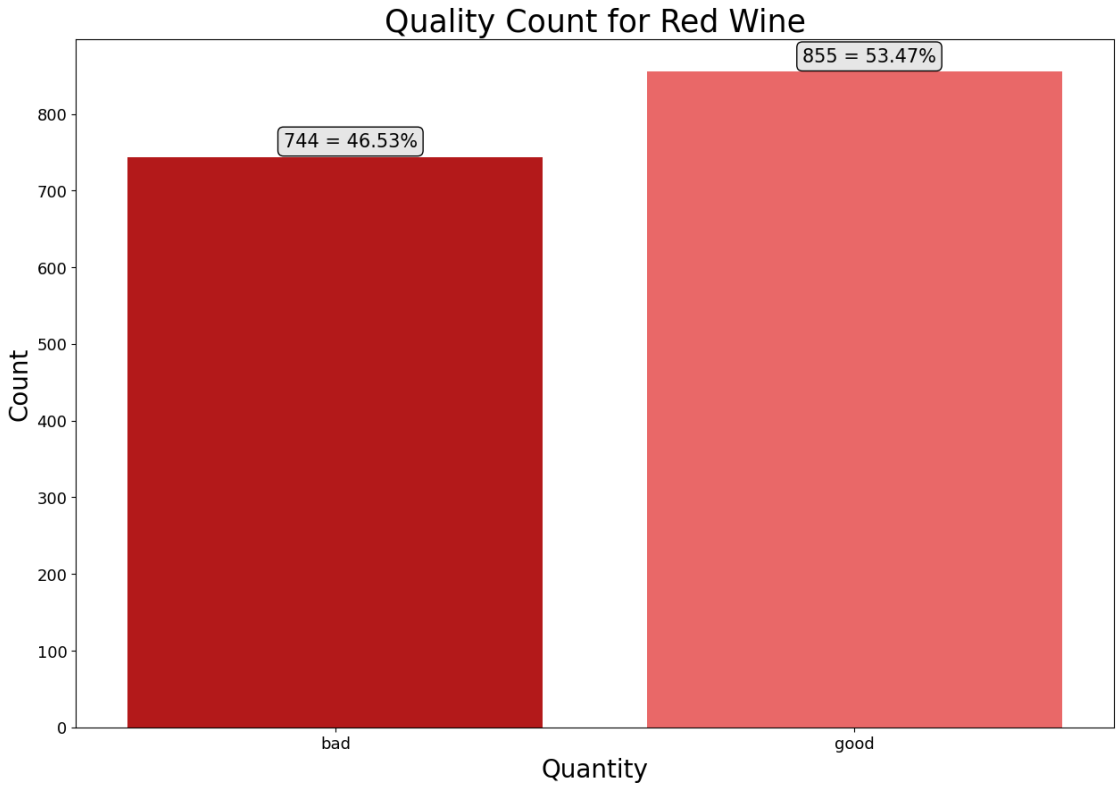


5 修复类不平衡

为了纠正数据集中的类不平衡, 需要使用 binning. 将数据集区分为坏和好两种品质的酒, 使得坏品质的酒喝好品质的酒的数量大致相同

[70]:
```
print(df.head())
df["quality"] = pd.cut(df["quality"], bins=[1, 5, 10], labels=["bad", "good"])

plt.figure(figsize=(15, 10))
ax = sns.countplot(
    x="quality", data=df, palette=[custom_colors[1], custom_colors[-3]]
)
bbox_args = dict(boxstyle="round", fc="0.9")
for p in ax.patches:
    ax.annotate(
        "{:.0f} = {:.2f}%".format(
            p.get_height(), (p.get_height() / len(df["quality"])) * 100
        ),
        (p.get_x() + 0.3, p.get_height() + 13),
        color="black",
        bbox=bbox_args,
        fontsize=15,
    )

plt.title("Quality Count for Red Wine", fontsize=25)
plt.xlabel("Quantity", fontsize=20)
plt.ylabel("Count", fontsize=20)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.show()
```

```
   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
```



6 对标签进行编码

[72]:
```
label_encoder=LabelEncoder()
df['quality']=label_encoder.fit_transform(df['quality'])
df['quality'].value_counts()
```

[72]:
```
1    855
0    744
Name: quality, dtype: int64
```

7 对数据缩放

StandardScaler 可以通过减去平均值, 并缩放导单位方差内对一个特征进行缩放.

$$z = \frac{x-\mu}{\theta}\mu = \text{Mean}\theta = \text{Standard Deviation}$$

[75]:
```
print(df.head())
scaler=StandardScaler()
features=[features for features in df.columns if df[features].dtype!=int]
df[features]=scaler.fit_transform(df[features])
```

```
df
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0      -0.528360          0.961877    -1.391472       -0.453218  -0.243707
1      -0.298547          1.967442    -1.391472        0.043416   0.223875
2      -0.298547          1.297065    -1.186070       -0.169427   0.096353
3       1.654856         -1.384443     1.484154       -0.453218  -0.264960
4      -0.528360          0.961877    -1.391472       -0.453218  -0.243707

   free sulfur dioxide  total sulfur dioxide   density        pH  sulphates  \
0            -0.466193             -0.379133  0.558274  1.288643  -0.579207
1             0.872638              0.624363  0.028261 -0.719933   0.128950
2            -0.083669              0.229047  0.134264 -0.331177  -0.048089
3             0.107592              0.411500  0.664277 -0.979104  -0.461180
4            -0.466193             -0.379133  0.558274  1.288643  -0.579207

    alcohol  quality
0 -0.960246        0
1 -0.584777        0
2 -0.584777        0
3 -0.584777        1
4 -0.960246        0
```

[75]:

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0         -0.528360          0.961877    -1.391472       -0.453218  -0.243707
1         -0.298547          1.967442    -1.391472        0.043416   0.223875
2         -0.298547          1.297065    -1.186070       -0.169427   0.096353
3          1.654856         -1.384443     1.484154       -0.453218  -0.264960
4         -0.528360          0.961877    -1.391472       -0.453218  -0.243707
…               …                 …            …               …          …
1594      -1.217796          0.403229    -0.980669       -0.382271   0.053845
1595      -1.390155          0.123905    -0.877968       -0.240375  -0.541259
1596      -1.160343         -0.099554    -0.723916       -0.169427  -0.243707
1597      -1.390155          0.654620    -0.775267       -0.382271  -0.264960
1598      -1.332702         -1.216849     1.021999        0.752894  -0.434990

      free sulfur dioxide  total sulfur dioxide   density        pH  \
0               -0.466193             -0.379133  0.558274  1.288643
1                0.872638              0.624363  0.028261 -0.719933
2               -0.083669              0.229047  0.134264 -0.331177
3                0.107592              0.411500  0.664277 -0.979104
4               -0.466193             -0.379133  0.558274  1.288643
…                     …                     …         …         …
1594             1.542054             -0.075043 -0.978765  0.899886
1595             2.211469              0.137820 -0.862162  1.353436
1596             1.255161             -0.196679 -0.533554  0.705508
1597             1.542054             -0.075043 -0.676657  1.677400
```

```
1598             0.203223             -0.135861 -0.666057  0.511130

      sulphates   alcohol  quality
0     -0.579207 -0.960246        0
1      0.128950 -0.584777        0
2     -0.048089 -0.584777        0
3     -0.461180 -0.584777        1
4     -0.579207 -0.960246        0
…           …       …        …
1594  -0.461180  0.072294        0
1595   0.601055  0.729364        1
1596   0.542042  0.541630        1
1597   0.305990 -0.209308        0
1598   0.010924  0.541630        1

[1599 rows x 12 columns]
```

[76]:
```
# Split Features and Target
X=df.drop('quality',axis=1)
y=df['quality']
print(X,'\n\n\n',y)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0         -0.528360          0.961877    -1.391472       -0.453218  -0.243707
1         -0.298547          1.967442    -1.391472        0.043416   0.223875
2         -0.298547          1.297065    -1.186070       -0.169427   0.096353
3          1.654856         -1.384443     1.484154       -0.453218  -0.264960
4         -0.528360          0.961877    -1.391472       -0.453218  -0.243707
…               …                 …            …               …          …
1594      -1.217796          0.403229    -0.980669       -0.382271   0.053845
1595      -1.390155          0.123905    -0.877968       -0.240375  -0.541259
1596      -1.160343         -0.099554    -0.723916       -0.169427  -0.243707
1597      -1.390155          0.654620    -0.775267       -0.382271  -0.264960
1598      -1.332702         -1.216849     1.021999        0.752894  -0.434990

      free sulfur dioxide  total sulfur dioxide   density        pH  \
0               -0.466193             -0.379133  0.558274  1.288643
1                0.872638              0.624363  0.028261 -0.719933
2               -0.083669              0.229047  0.134264 -0.331177
3                0.107592              0.411500  0.664277 -0.979104
4               -0.466193             -0.379133  0.558274  1.288643
…                     …                     …         …         …
1594             1.542054             -0.075043 -0.978765  0.899886
1595             2.211469              0.137820 -0.862162  1.353436
1596             1.255161             -0.196679 -0.533554  0.705508
1597             1.542054             -0.075043 -0.676657  1.677400
1598             0.203223             -0.135861 -0.666057  0.511130
```

```
      sulphates   alcohol
0     -0.579207 -0.960246
1      0.128950 -0.584777
2     -0.048089 -0.584777
3     -0.461180 -0.584777
4     -0.579207 -0.960246
…           …         …
1594  -0.461180  0.072294
1595   0.601055  0.729364
1596   0.542042  0.541630
1597   0.305990 -0.209308
1598   0.010924  0.541630

[1599 rows x 11 columns]
```

```
 0       0
1        0
2        0
3        1
4        0
        ..
1594     0
1595     1
1596     1
1597     0
1598     1
Name: quality, Length: 1599, dtype: int64
```

8 训练集和测试集划分

[77]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪2,random_state=0,stratify=y)
```

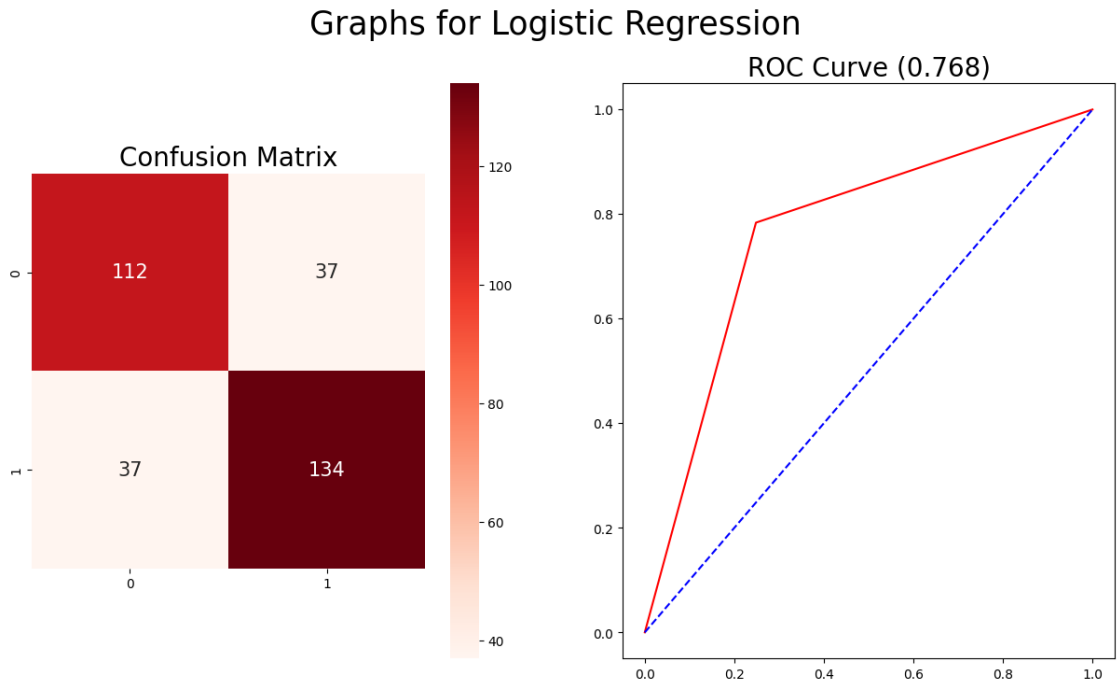
9 训练机器学习模型

[91]:
```
algo_name = []
accuracy = []


def display_results_and_graphs(algorithm_name, model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc_model = model.score(X_test, y_test)

    algo_name.append(algorithm_name)
```

```
    accuracy.append(acc_model)

    print(f"---For {algorithm_name}---")
    print(
        "Training Accuracy: {}%\nTesting Accuracy: {}%\nF1 Score: {}".format(
            model.score(X_train, y_train) * 100,
            model.score(X_test, y_test) * 100,
            f1_score(y_test, y_pred),
        )
    )
    print("\n")

    fig, axes = plt.subplots(1, 2, figsize=(15, 8))

    fig.suptitle("Graphs for " + algorithm_name, fontsize=25)

    sns.heatmap(
        confusion_matrix(y_test, y_pred),
        annot=True,
        cmap="Reds",
        annot_kws={"size": 15},
        square=True,
        fmt=".0f",
        ax=axes[0],
    )
    axes[0].set_title("Confusion Matrix", fontsize=20)

    fpr, tpr, threshould = metrics.roc_curve(y_test, y_pred)
    roc_auc = metrics.auc(fpr, tpr)
    sns.lineplot(x=fpr, y=tpr, ax=axes[1], color="red")
    axes[1].set_title("ROC Curve (" + str(round(roc_auc, 3)) + ")", fontsize=20)
    axes[1].plot(
        [0, 1],
        [
            0,
            1,
        ],
        "b--",
    )
    plt.show()


display_results_and_graphs("Logistic Regression", LogisticRegression())
display_results_and_graphs(
    "K Nearest Neighbors", KNeighborsClassifier(n_neighbors=13)
)
display_results_and_graphs("Support Vector Classifier", SVC())
```

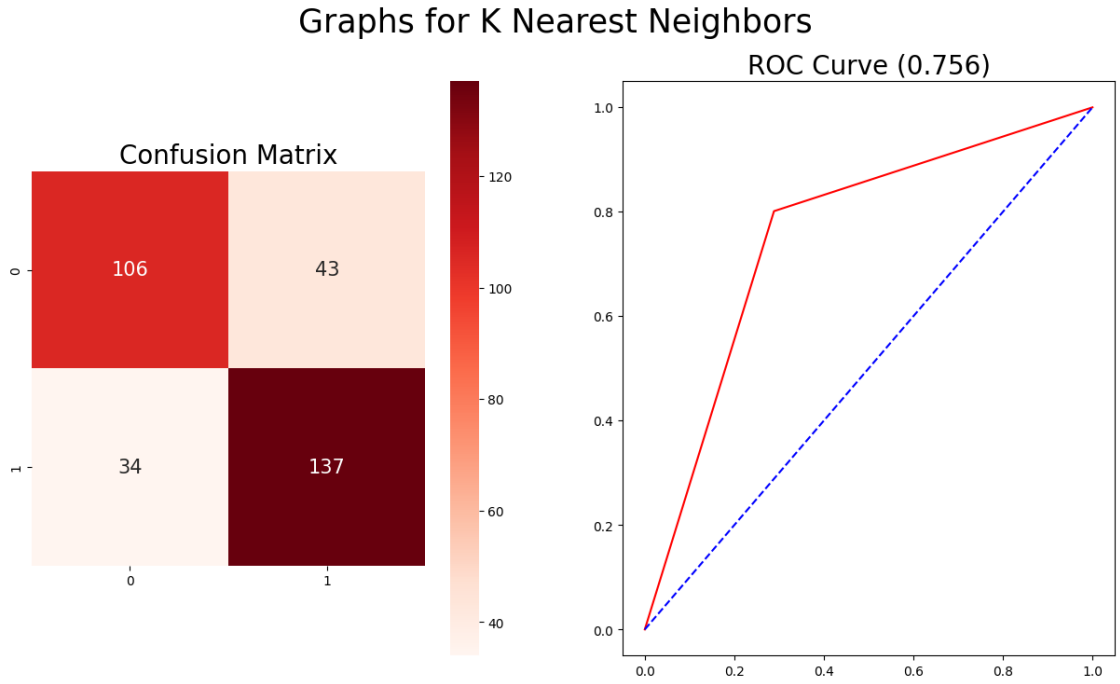
```
display_results_and_graphs(
    "Decision Tree Classifier", DecisionTreeClassifier(random_state=10)
)
display_results_and_graphs("Random Forest Classifier", RandomForestClassifier())
display_results_and_graphs(
    "Gradient Boosting Classifier", GradientBoostingClassifier(random_state=10)
)
display_results_and_graphs(
    "Ada Boost Classifier", AdaBoostClassifier(random_state=0)
)
display_results_and_graphs(
    "Cat Boost Classifier", CatBoostClassifier(verbose=0)
)
display_results_and_graphs("XGBoost Classifier", XGBClassifier())
display_results_and_graphs(
    "Light Gradient Boosting Machine", lgbm.LGBMClassifier()
)
```

```
---For Logistic Regression---
Training Accuracy: 73.88584831899921%
Testing Accuracy: 76.875%
F1 Score: 0.783625730994152
```



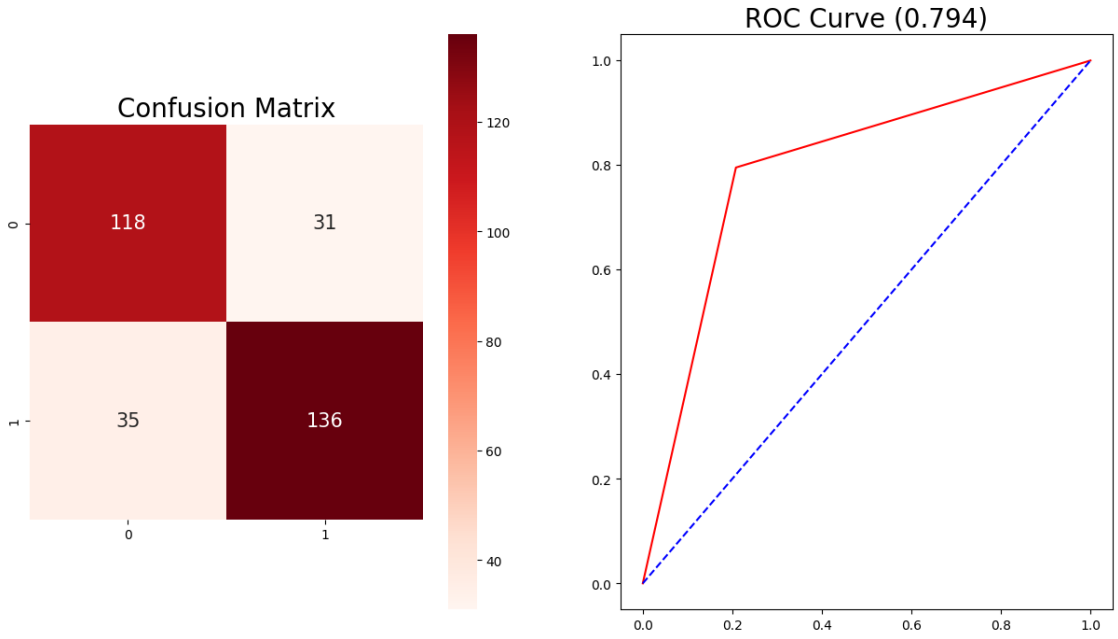

```
---For K Nearest Neighbors---
```

```
Training Accuracy: 77.79515246286161%
Testing Accuracy: 75.9375%
F1 Score: 0.7806267806267805
```



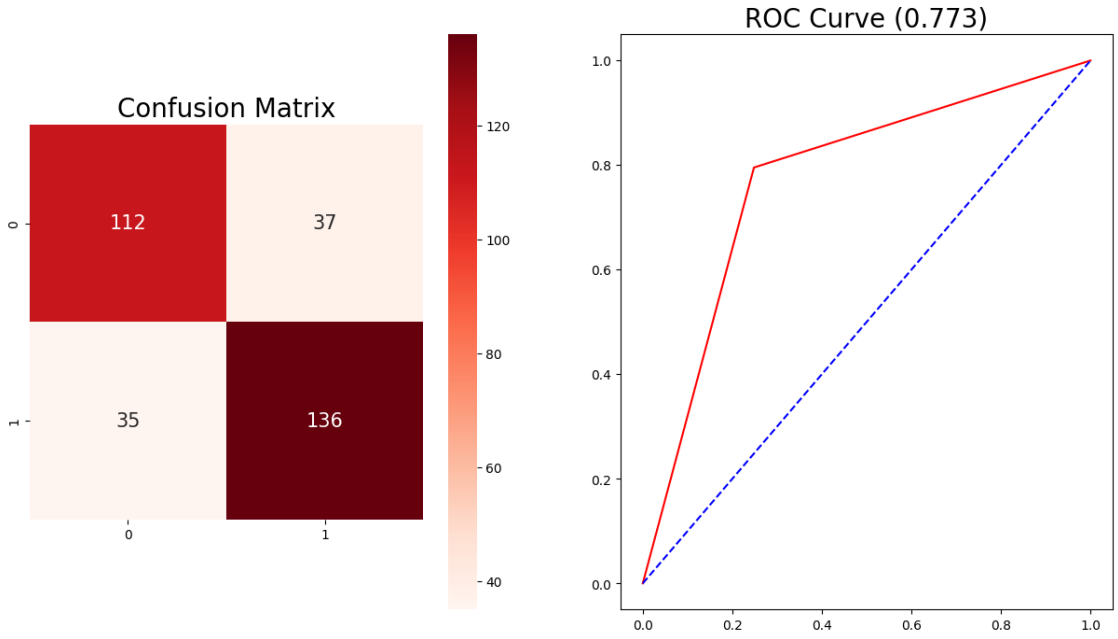

```
---For Support Vector Classifier---
Training Accuracy: 78.65519937451134%
Testing Accuracy: 79.375%
F1 Score: 0.8047337278106509
```

Graphs for Support Vector Classifier

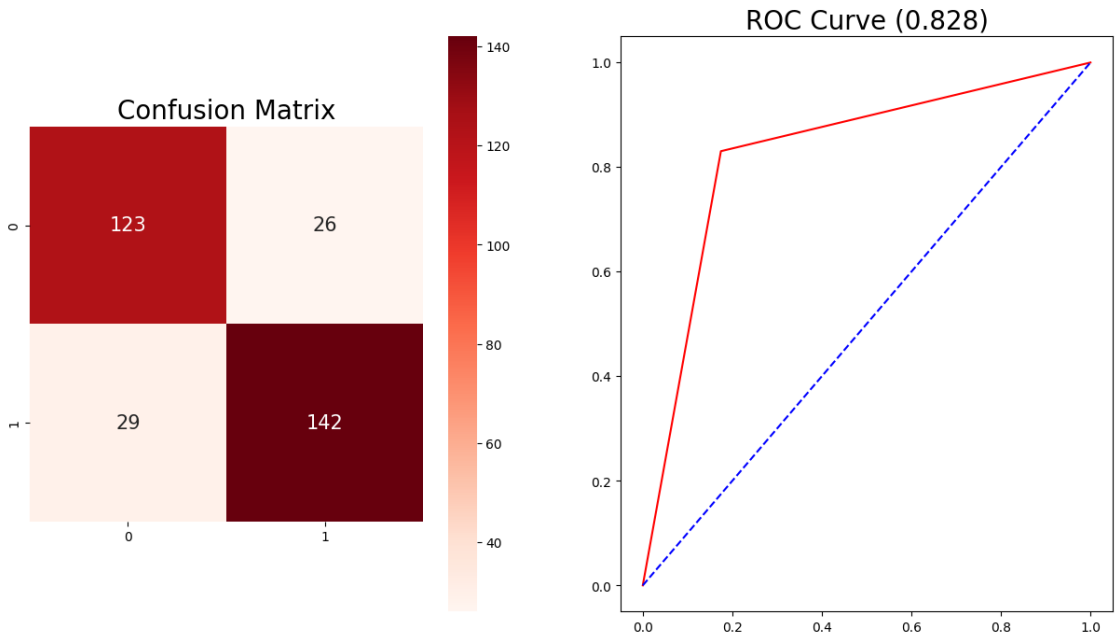



```
---For Decision Tree Classifier---
Training Accuracy: 100.0%
Testing Accuracy: 77.5%
F1 Score: 0.7906976744186047
```

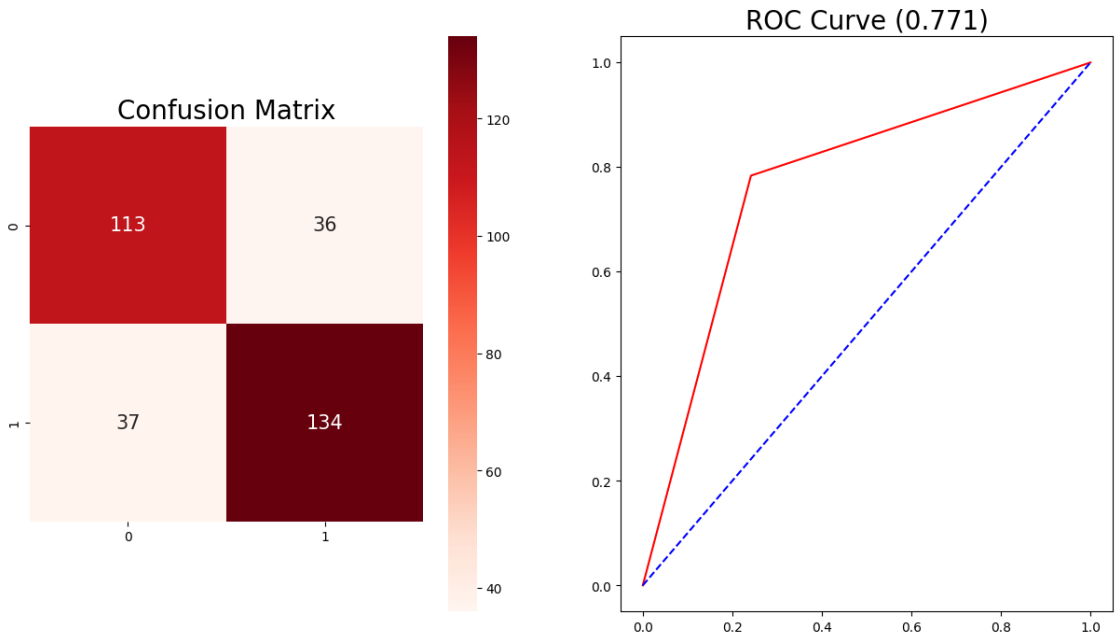
Graphs for Decision Tree Classifier




```
---For Random Forest Classifier---
Training Accuracy: 100.0%
Testing Accuracy: 82.8125%
F1 Score: 0.8377581120943952
```

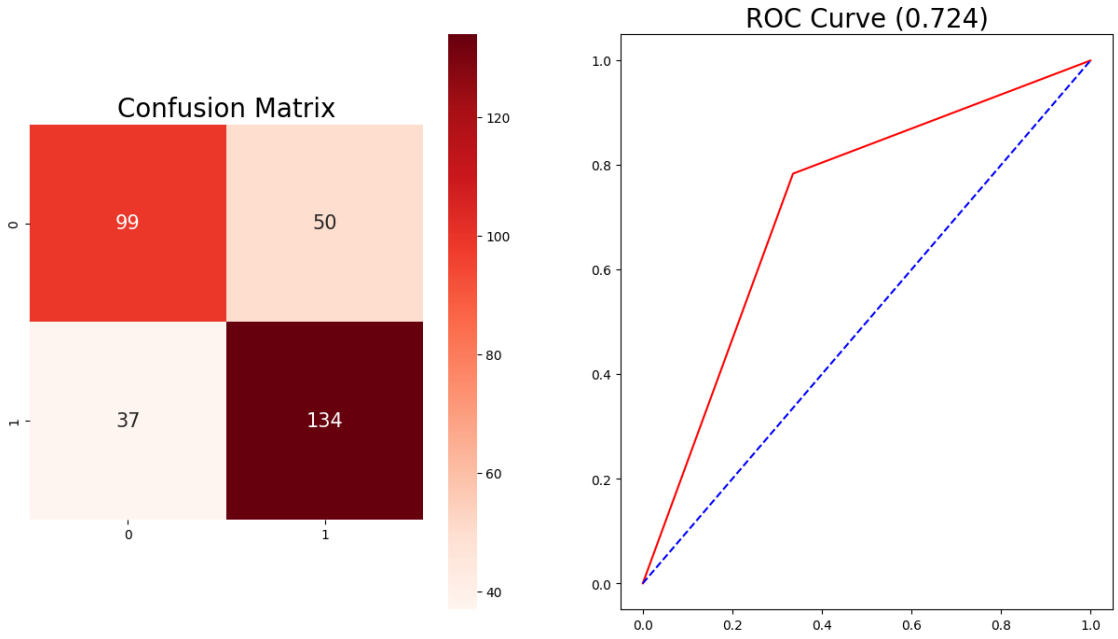



```
---For Gradient Boosting Classifier---
Training Accuracy: 87.33385457388584%
Testing Accuracy: 77.1875%
F1 Score: 0.7859237536656891
```

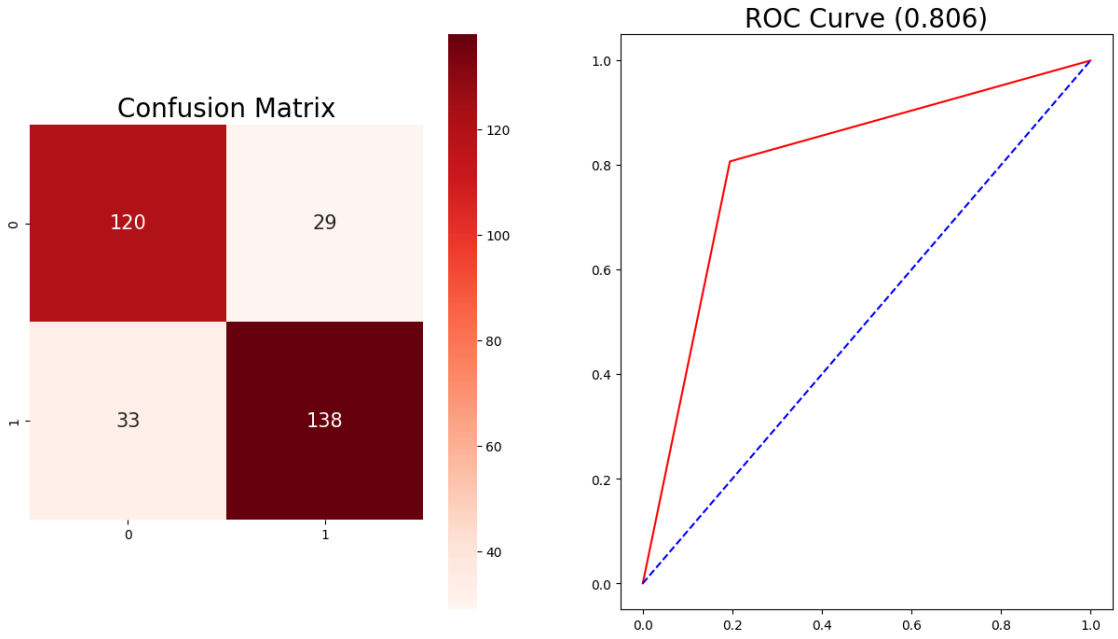



```
---For Ada Boost Classifier---
Training Accuracy: 78.18608287724786%
Testing Accuracy: 72.8125%
F1 Score: 0.7549295774647887
```

Graphs for Ada Boost Classifier



---For Cat Boost Classifier---
Training Accuracy: 93.58874120406567%
Testing Accuracy: 80.625%
F1 Score: 0.8165680473372781


Graphs for Cat Boost Classifier
Confusion Matrix
120
29
33
138
0
1
0
1
120
100
80
60
40
ROC Curve (0.806)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

[ ]: