**Wine Quality Prediction using Machine Learning**

Wine industries nowadays use quality certificate to promote their products in the market. It is time-consuming and expensive for the human experts to check the quality of wine. With the advancement of technology in every field, machine learning algorithms can be used to predict the quality of wine. The major steps involved to predict the quality of wine using machine learning are

* Importing the required libraries
* Loading the wine quality dataset into a data frame.
* Exploratory Data Analysis on the dataset.
* Constructing the regression models (like Linear Regression Model, Support Vector Regressor Model, Random Forest Regressor Model and Gradient Boosting Regressor Model) using the dataset and comparing them.
* Evaluating the model.

**Description of the dataset:** The dataset that we use for analysis is red wine dataset taken from Kaggle. The dataset describes the amount of various chemicals present in wine and their effect on its quality.

**Input variables (based on physicochemical tests):**

1. **fixed acidity** - Non-volatile acids in wine are what cause fixed acidity. For instance, malic, citric, or tartaric acid. This kind of acid mixes the wine's taste's harmony and adds freshness.
2. **volatile acidity** - The portion of the acid in wine that may be detected by the nose is called volatile acidity. As opposed to those acids that can be tasted. One of the most typical flaws is volatile acidity, or the souring of wine.
3. **citric acid** - It can be used to collect wine, treat wine with acid to increase acidity, and clean filters of potential fungal and mould infections.
4. **residual sugar** - Grape sugar that has not undergone alcohol fermentation is referred to as residual sugar.
5. **chlorides** - The amount of minerals in the wine affects its structure as well. Their content is mostly influenced by climatic region, oenological procedures, wine storage, and ageing.
6. **free sulfur dioxide** - The antioxidant and antibacterial qualities of sulphur dioxide make it a useful preservative. It is a very powerful antibiotic that has a major impact on consumption and can result in wine deterioration.
7. **total sulfur dioxide** - It consists of the SO2 that is present in the wine both free and bound to other substances like aldehydes, pigments, or sugars.
8. **density** - Wine can have a density that is less or greater than that of water. Its value is mostly influenced by the alcohol and sugar content.
9. **pH** - Wine's pH is a gauge of its acidity. The optimal pH range for all wines is between 2.9 and 4.2. More acidic wines have lower pH values; less acidic wines have higher pH values.
10. **sulphates** - Sulfates are a byproduct of wine's sugar being fermented by yeast into alcohol.
11. **alcohol** - Wines' alcohol percentage is influenced by a wide range of factors, including grape varietal, berry sugar content, manufacturing methods, and growth environments.

**Question 1:**The following attitude helps to predict the quality of wine

1. **quality** - This is the variable we are after. This is a score between 0 and 10 where a higher score indicates that it was deemed to be a superior wine.

The following tasks will be performed in the project:

1. Performing EDA on the dataset
2. Fitting regression models (like Linear Regression Model, Support Vector Regressor Model, Random Forest Regressor Model and Gradient Boosting Regressor Model) to find the quality of wine using the 11 attributes of the wine dataset and comparing them.
3. Observe Error values and R2 score of different models.

**Question 2:**

In this project, we tried to predict the quality of red wine based on all the input features by fitting a supervised algorithms like Linear Regression Model, Support Vector Regressor Model, Random Forest Regressor Model and Gradient Boosting Regressor Model.

**Question 3:** To know the interesting insights from the dataset, the following EDA is done:

The following features are relatively correlated:

* total sulfur dioxide with free sulfur dioxide
* fixed acidity with density and citric acid
* alcohol with quality

The following features are inversely correlated:

* fixed acidity with pH
* citric acid with pH and volatile acidity

**Exploratory Data Analysis on Wine Quality Dataset:**

#Loading the required data

df = pd.read\_csv(r"/content/drive/MyDrive/Dataset/winequality-red.csv")

# Data Information

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1599 entries, 0 to 1598

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 fixed acidity 1599 non-null float64

1 volatile acidity 1599 non-null float64

2 citric acid 1599 non-null float64

3 residual sugar 1599 non-null float64

4 chlorides 1599 non-null float64

5 free sulfur dioxide 1599 non-null float64

6 total sulfur dioxide 1599 non-null float64

7 density 1599 non-null float64

8 pH 1599 non-null float64

9 sulphates 1599 non-null float64

10 alcohol 1599 non-null float64

11 quality 1599 non-null int64

dtypes: float64(11), int64(1)

memory usage: 150.0 KB

|  | **fixed acidity** | **volatile acidity** | **citric acid** | **residual sugar** | **chlorides** | **free sulfur dioxide** | **total sulfur dioxide** | **density** | **pH** | **sulphates** | **alcohol** | **quality** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 |
| **mean** | 8.319637 | 0.527821 | 0.270976 | 2.538806 | 0.087467 | 15.874922 | 46.467792 | 0.996747 | 3.311113 | 0.658149 | 10.422983 | 5.636023 |
| **std** | 1.741096 | 0.179060 | 0.194801 | 1.409928 | 0.047065 | 10.460157 | 32.895324 | 0.001887 | 0.154386 | 0.169507 | 1.065668 | 0.807569 |
| **min** | 4.600000 | 0.120000 | 0.000000 | 0.900000 | 0.012000 | 1.000000 | 6.000000 | 0.990070 | 2.740000 | 0.330000 | 8.400000 | 3.000000 |
| **25%** | 7.100000 | 0.390000 | 0.090000 | 1.900000 | 0.070000 | 7.000000 | 22.000000 | 0.995600 | 3.210000 | 0.550000 | 9.500000 | 5.000000 |
| **50%** | 7.900000 | 0.520000 | 0.260000 | 2.200000 | 0.079000 | 14.000000 | 38.000000 | 0.996750 | 3.310000 | 0.620000 | 10.200000 | 6.000000 |
| **75%** | 9.200000 | 0.640000 | 0.420000 | 2.600000 | 0.090000 | 21.000000 | 62.000000 | 0.997835 | 3.400000 | 0.730000 | 11.100000 | 6.000000 |
| **max** | 15.900000 | 1.580000 | 1.000000 | 15.500000 | 0.611000 | 72.000000 | 289.000000 | 1.003690 | 4.010000 | 2.000000 | 14.900000 | 8.000000 |

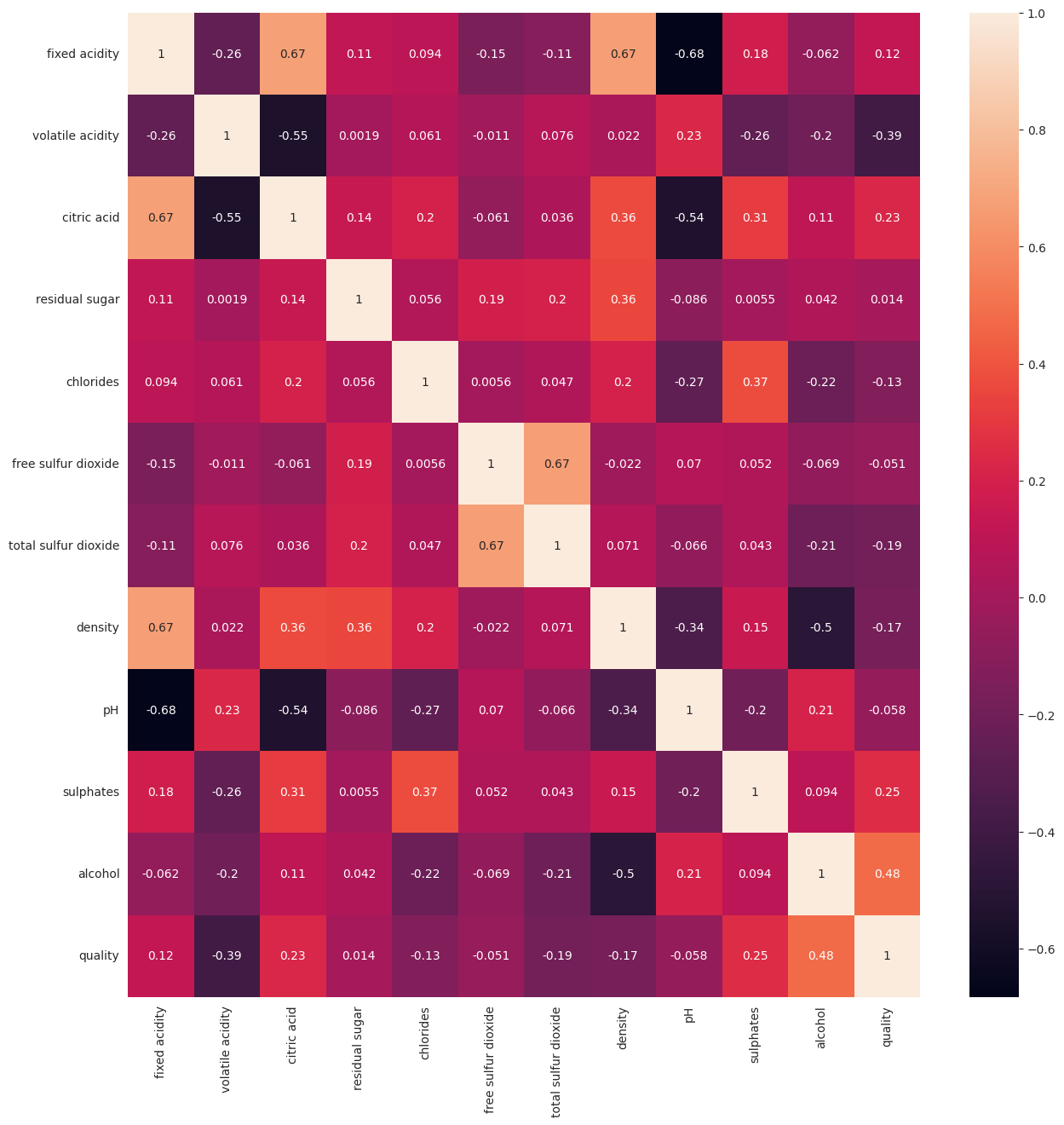
#Statistical summary

df.describe()

# **Correlation between different Features**

plt.figure(figsize=(15,15))

sns.heatmap(df.corr(),color = "k", annot=True)



The following features are relatively correlated:

total sulfur dioxide with free sulfur dioxide; fixed acidity with density and citric acid; alcohol with quality

The following features are inversely correlated:

fixed acidity with pH citric acid with pH and volatile acidity

# Selecting highly correlated features

relevant\_features= abs(df.corr()['quality'])[abs(df.corr()['quality'])>0.1]

relevant\_features.nlargest(10)

quality 1.000000

alcohol 0.476166

volatile acidity 0.390558

sulphates 0.251397

citric acid 0.226373

total sulfur dioxide 0.185100

density 0.174919

chlorides 0.128907

fixed acidity 0.124052

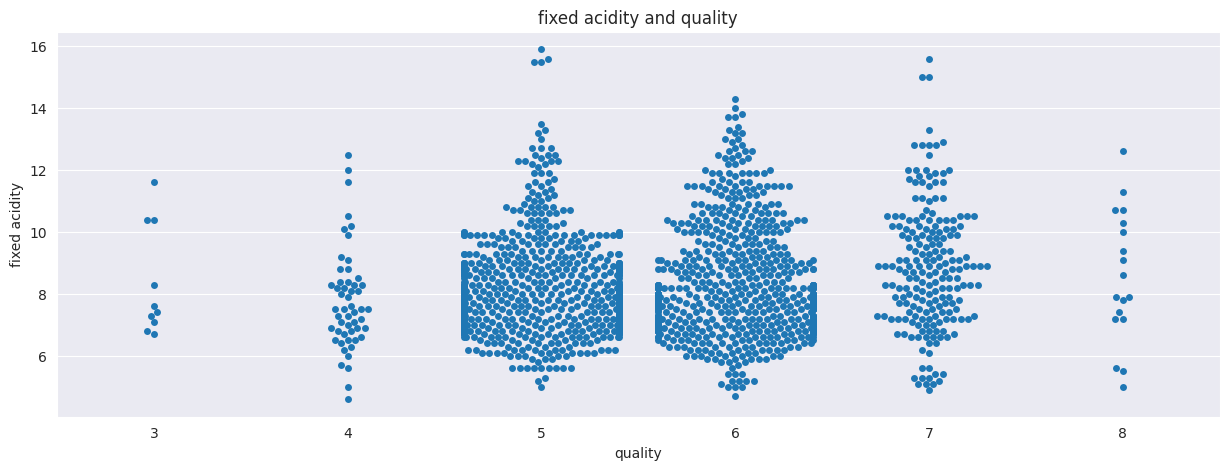
Name: quality, dtype: float64

# **Relations between fixed acidity and quality**

plt.figure(figsize=(15,5))

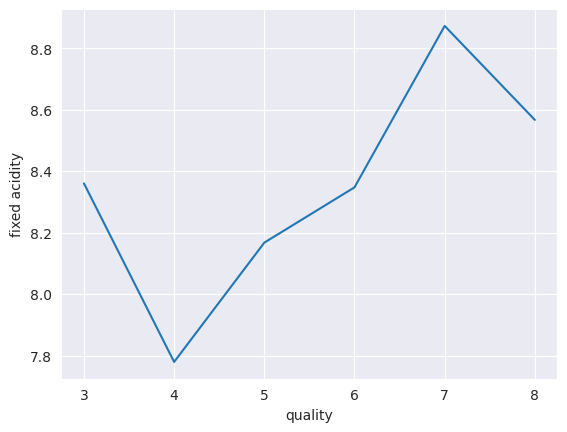
sns.swarmplot(x= "quality", y="fixed acidity" , data = df)

plt.title('fixed acidity and quality')



df.groupby('quality')['fixed acidity'].mean().plot.line()

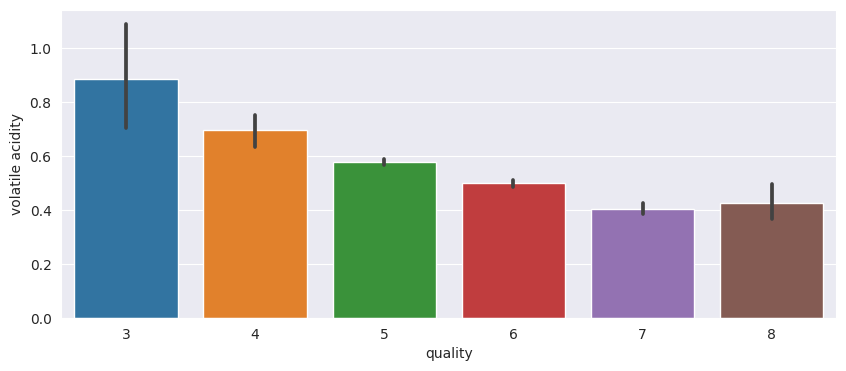
plt.ylabel("fixed acidity")



# **Relations between volatile acidity and quality**

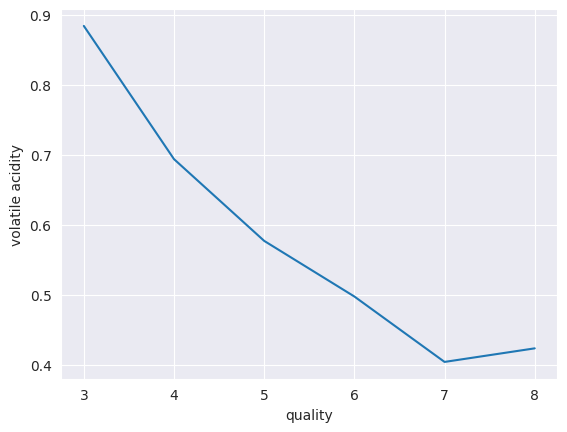
plt.figure(figsize=(10,4))

sns.barplot(x="quality", y="volatile acidity",   data=df )



df.groupby('quality')['volatile acidity'].mean().plot.line()

plt.ylabel("volatile acidity")



# **Data Distribution Skew**

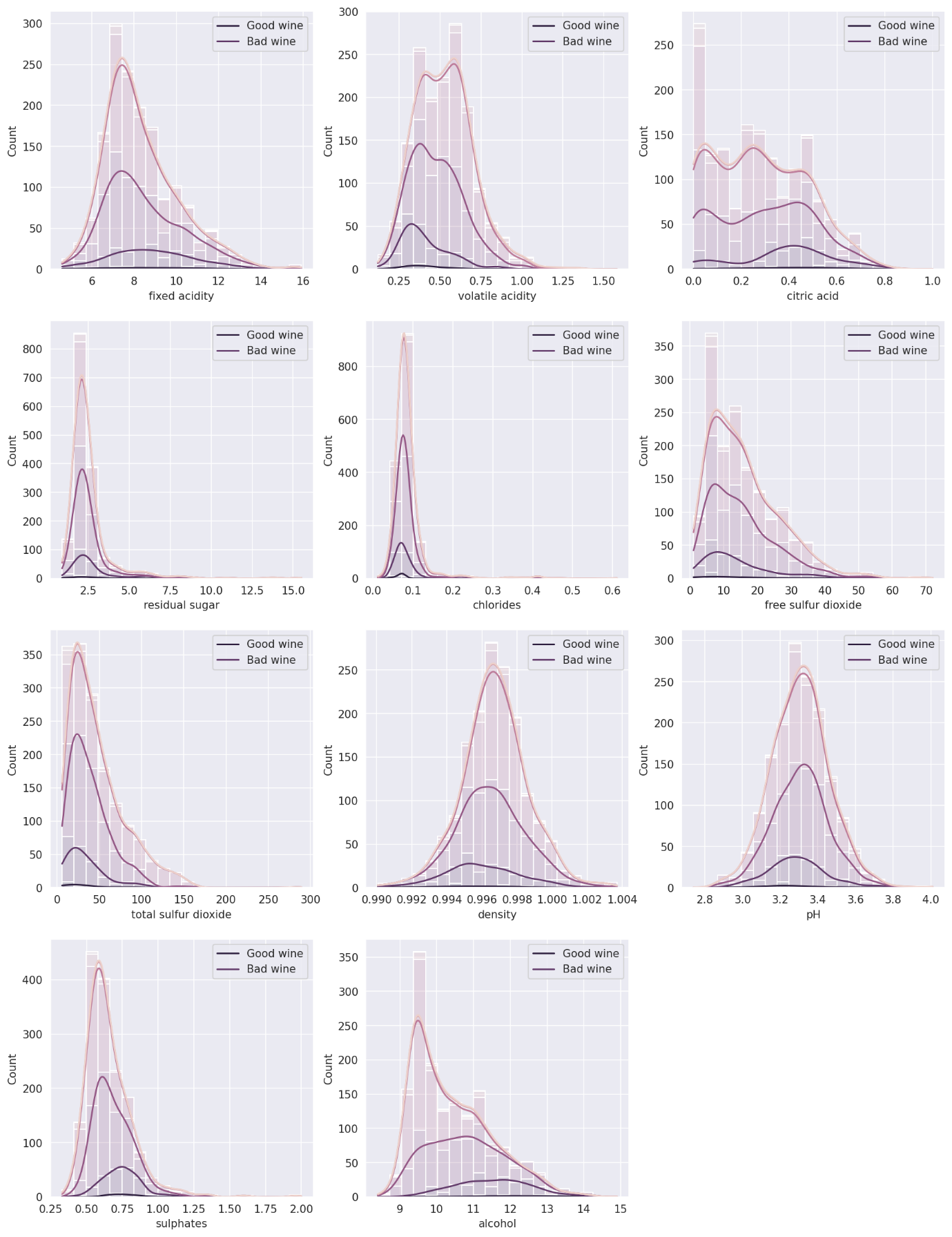
plt.figure(figsize=(15, 20),dpi=150)

for i,j in enumerate(df.drop('quality',axis=1).columns):

    plt.subplot(4, 3, i+1)

    sns.histplot(data=df, x=df[f"{j}"], hue="quality", kde=True, bins=20, multiple="stack", alpha=.2)

    plt.legend(['Good wine','Bad wine'])



# **Part 2: Modelling**

# **Splitting Data**

from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.preprocessing import StandardScaler  
from sklearn.pipeline import make\_pipeline  
from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error,r2\_score  
from numpy import round  
  
standarized\_df = round((df-df.mean())/df.std(), 3)  
standarized\_df  
  
X = standarized\_df.loc[:,'fixed acidity':'alcohol']  
y = standarized\_df['quality']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2,shuffle=True, stratify=df["quality"])

# **Model 1: Generating a linear regression model**

reg =  LinearRegression()  
reg.fit(X\_train, y\_train)  
  
#printing intercepts and slope  
print("intercept")  
print(reg.intercept\_)  
print()  
print("regression coeeficients")  
print(reg.coef\_)  
  
  
predictions = reg.predict(X\_test)  
  
print()  
print("Error values")  
print(f"Mean Squared Error(MSE): {mean\_squared\_error(y\_test, round(predictions, 0))}")  
print(f"Mean Absolute Error(MAE): {mean\_absolute\_error(y\_test, round(predictions, 0))}")  
print(f"R2 Score: {r2\_score(y\_test, round(predictions, 0))}")

intercept

0.01518705563026331

regression coeeficients

[ 0.06630889 -0.22596305 -0.03505179 0.02510622 -0.08923571 0.0770734

-0.15432789 -0.07010604 -0.08288477 0.19865599 0.34776039]

Error values

Mean Squared Error(MSE): 0.695866771875

Mean Absolute Error(MAE): 0.707240625

R2 Score: 0.2969754070620645

# **Model 2: Generating a SVM Model**

from sklearn.svm import SVR  
svm\_model = SVR()  
svm\_model.fit(X\_train, y\_train)  
  
predictions = svm\_model.predict(X\_test)  
  
print()  
print("Error values")  
print(f"Mean Squared Error(MSE): {mean\_squared\_error(y\_test, round(predictions, 0))}")  
print(f"Mean Absolute Error(MAE): {mean\_absolute\_error(y\_test, round(predictions, 0))}")  
print(f"R2 Score: {r2\_score(y\_test, round(predictions, 0))}")

Error values

Mean Squared Error(MSE): 0.675079271875

Mean Absolute Error(MAE): 0.665559375

R2 Score: 0.31797673133326054

# **Model 3: Generating a Random Forest Regressor Model**

from sklearn.ensemble import RandomForestRegressor  
  
rf\_model = RandomForestRegressor()  
rf\_model.fit(X\_train, y\_train)  
  
predictions = rf\_model.predict(X\_test)  
  
print()  
print("Error values")  
print(f"Mean Squared Error(MSE): {mean\_squared\_error(y\_test, round(predictions, 0))}")  
print(f"Mean Absolute Error(MAE): {mean\_absolute\_error(y\_test, round(predictions, 0))}")  
print(f"R2 Score: {r2\_score(y\_test, round(predictions, 0))}")

Error values

Mean Squared Error(MSE): 0.5352042718749999

Mean Absolute Error(MAE): 0.614240625

R2 Score: 0.4592905128093189

# **Model 4: Generating a Gradient Boosting Regressor Model**

from sklearn.ensemble import GradientBoostingRegressor  
gb\_model = GradientBoostingRegressor()  
gb\_model.fit(X\_train, y\_train)  
  
predictions = gb\_model.predict(X\_test)  
  
print()  
print("Error values")  
print(f"Mean Squared Error(MSE): {mean\_squared\_error(y\_test, round(predictions, 0))}")  
print(f"Mean Absolute Error(MAE): {mean\_absolute\_error(y\_test, round(predictions, 0))}")  
print(f"R2 Score: {r2\_score(y\_test, round(predictions, 0))}")

Error values

Mean Squared Error(MSE): 0.6300792718749999

Mean Absolute Error(MAE): 0.676434375

R2 Score: 0.3634396100923143

**Question 4: Comparison of Models**

The MAE, MSE and R2 Score values of different models is given below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of the Model** | **MAE** | **MSE** | **R2 Score** |
| Linear Regression | 0.707 | 0.696 | 0.297 |
| Support Vector Regressor | 0.665 | 0.675 | 0.318 |
| Random Forest Regressor | **0.614** | **0.535** | **0.459** |
| Gradient Boosting Regressor | 0.676 | 0.630 | 0.363 |

* It is observed from the above table that Random Forest Regressor is the best model to fit the given data.
* All the models are trained using all the input features.
* If we train the model using some important or relevant features then there would be a chance of improvement in the prediction.