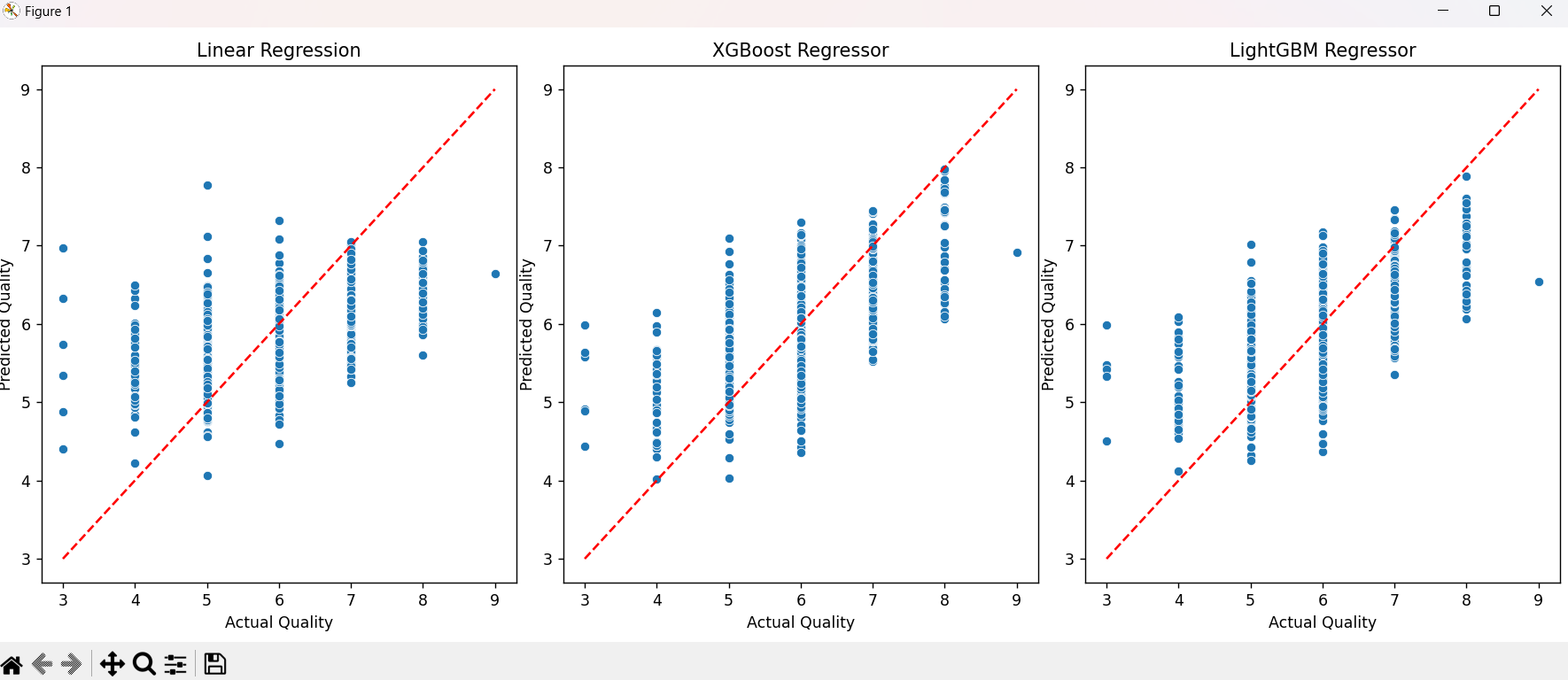
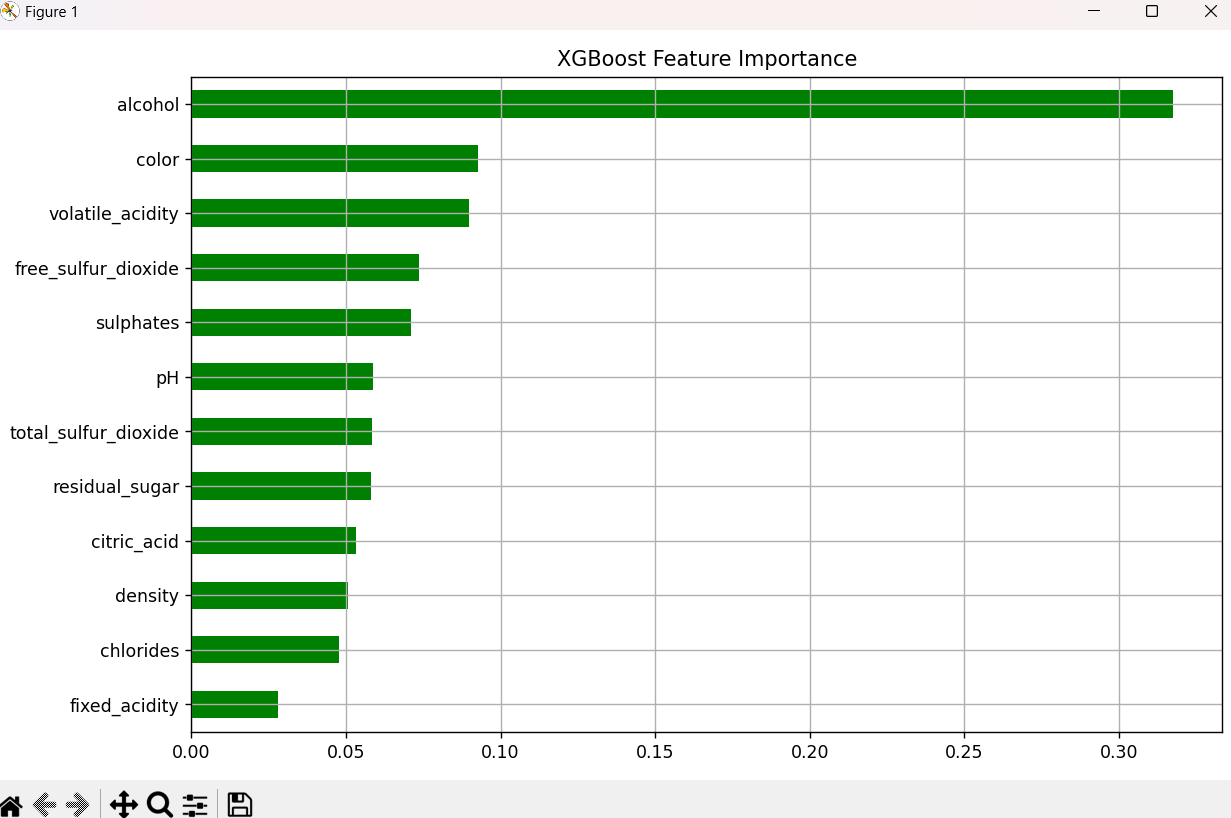
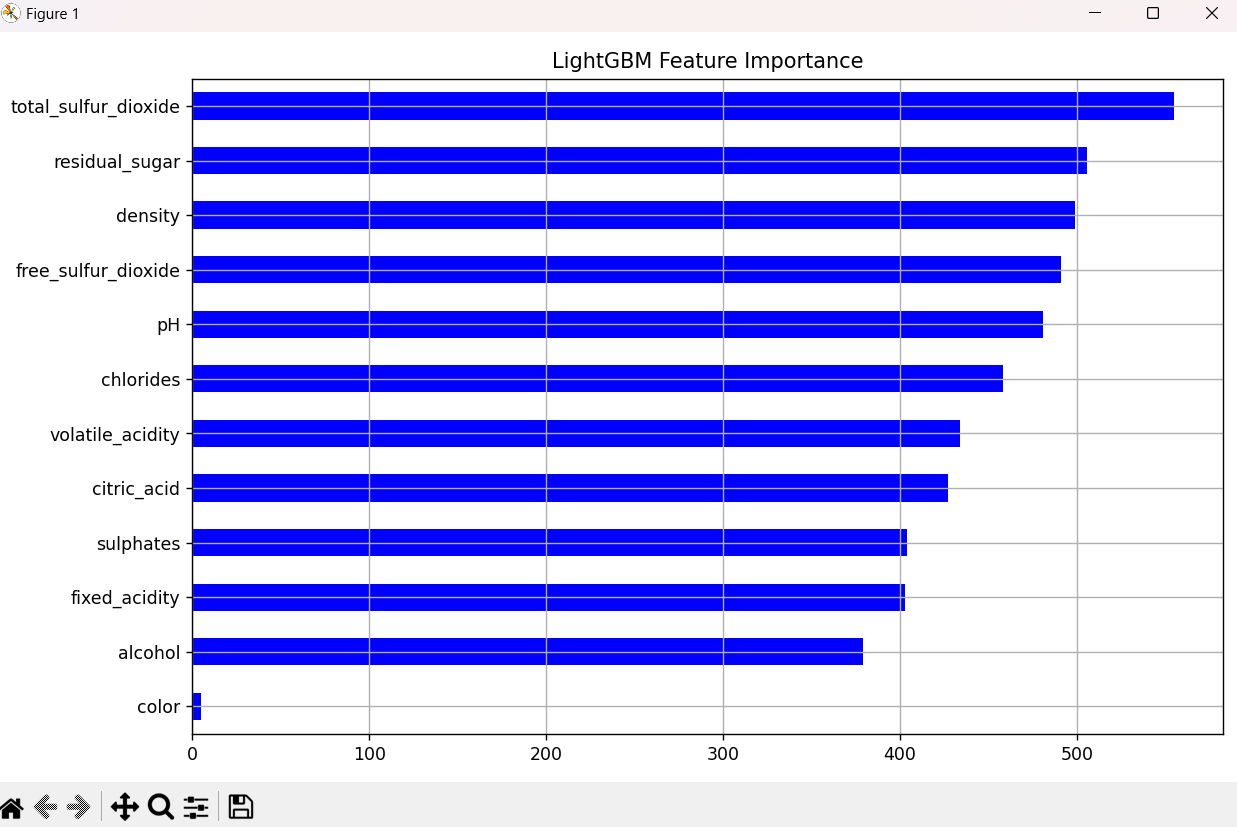
**Wine Quality Prediction using 3 models.**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.linear\_model import LinearRegression  
from sklearn.preprocessing import StandardScaler, LabelEncoder  
from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score  
from xgboost import XGBRegressor  
import lightgbm as lgb  
  
  
df = pd.read\_csv(r"C:\Users\Administrator\Downloads\Wine\_Quality\_Data.csv")  
  
  
if 'color' in df.columns:  
 df['color'] = LabelEncoder().fit\_transform(df['color'])  
  
  
X = df.drop(columns=['quality'])  
y = df['quality']  
  
  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
  
lr\_model = LinearRegression()  
lr\_model.fit(X\_train, y\_train)  
y\_pred\_lr = lr\_model.predict(X\_test)  
  
  
xgb\_model = XGBRegressor(objective='reg:squarederror', random\_state=42)  
xgb\_param\_grid = {  
 'learning\_rate': [0.01, 0.1, 0.2],  
 'max\_depth': [3, 5, 7],  
 'n\_estimators': [50, 100, 200]  
}  
xgb\_grid\_search = GridSearchCV(estimator=xgb\_model, param\_grid=xgb\_param\_grid, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')  
xgb\_grid\_search.fit(X\_train, y\_train)  
best\_xgb\_model = xgb\_grid\_search.best\_estimator\_  
y\_pred\_xgb = best\_xgb\_model.predict(X\_test)  
  
  
lgb\_model = lgb.LGBMRegressor(random\_state=42)  
lgb\_param\_grid = {  
 'learning\_rate': [0.01, 0.1, 0.2],  
 'max\_depth': [3, 5, 7],  
 'n\_estimators': [50, 100, 200]  
}  
lgb\_grid\_search = GridSearchCV(estimator=lgb\_model, param\_grid=lgb\_param\_grid, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')  
lgb\_grid\_search.fit(X\_train, y\_train)  
best\_lgb\_model = lgb\_grid\_search.best\_estimator\_  
y\_pred\_lgb = best\_lgb\_model.predict(X\_test)  
  
  
def evaluate\_model(name, y\_true, y\_pred):  
 print(f"\n{name} Evaluation:")  
 print("MAE:", mean\_absolute\_error(y\_true, y\_pred))  
 print("MSE:", mean\_squared\_error(y\_true, y\_pred))  
 print("R² Score:", r2\_score(y\_true, y\_pred))  
  
  
evaluate\_model("Linear Regression", y\_test, y\_pred\_lr)  
evaluate\_model("XGBoost Regressor", y\_test, y\_pred\_xgb)  
evaluate\_model("LightGBM Regressor", y\_test, y\_pred\_lgb)  
  
  
plt.figure(figsize=(18, 6))  
  
plt.subplot(1, 3, 1)  
sns.scatterplot(x=y\_test, y=y\_pred\_lr)  
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')  
plt.title("Linear Regression")  
plt.xlabel("Actual Quality")  
plt.ylabel("Predicted Quality")  
  
plt.subplot(1, 3, 2)  
sns.scatterplot(x=y\_test, y=y\_pred\_xgb)  
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')  
plt.title("XGBoost Regressor")  
plt.xlabel("Actual Quality")  
plt.ylabel("Predicted Quality")  
  
plt.subplot(1, 3, 3)  
sns.scatterplot(x=y\_test, y=y\_pred\_lgb)  
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')  
plt.title("LightGBM Regressor")  
plt.xlabel("Actual Quality")  
plt.ylabel("Predicted Quality")  
  
plt.tight\_layout()  
plt.show()  
  
  
plt.figure(figsize=(10,6))  
xgb\_importance = pd.Series(best\_xgb\_model.feature\_importances\_, index=X.columns)  
xgb\_importance.sort\_values().plot(kind='barh', color='green')  
plt.title("XGBoost Feature Importance")  
plt.grid(True)  
plt.tight\_layout()  
plt.show()  
  
  
plt.figure(figsize=(10,6))  
lgb\_importance = pd.Series(best\_lgb\_model.feature\_importances\_, index=X.columns)  
lgb\_importance.sort\_values().plot(kind='barh', color='blue')  
plt.title("LightGBM Feature Importance")  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

**Output:**  






**Eval results, final prediction output:**

