In [31]:
```
#import libraries
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.metrics import accuracy_score

from sklearn import pipeline
```

In [32]:
```
#import libraries
#import csv_dataset
#check for missing values
#declare axes
#split
#create mode and fit
#test
#accuracy
#optimization
```

In [33]:
```
df = pd.read_csv("C:\\Users\\DAVIS\\Desktop\\Pandas\\winequality\\winequality.csv")
df[:5]
```

Out[33]:

	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

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
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
```

In [ ]:
```

```

In [5]:
```
x=df.drop(['quality'],axis=1)
```

In [6]:
```
y=df['quality']
y
```

Out[6]:
```
0       5
1       5
2       5
3       6
4       5
       ..
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64
```

In [7]:
```
x_train, x_test, y_train, y_test= train_test_split(x,y, test_size=0.2, random_state=42)
scaler = preprocessing.StandardScaler().fit(x_train)

x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled=scaler.transform(x_test)
```

In [8]:
```
logistic_model= LogisticRegression()
logistic_model.fit(x_train_scaled, y_train)
predictions=logistic_model.predict(x_test_scaled)
predictions
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/s
table/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (http
s://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[8]:
```
array([5, 5, 6, 5, 6, 5, 5, 5, 6, 6, 6, 5, 6, 5, 5, 7, 5, 5, 7, 5, 5, 5,
       6, 6, 5, 5, 7, 5, 5, 6, 5, 5, 6, 5, 6, 5, 6, 6, 6, 6, 5, 5, 6, 5,
       6, 6, 7, 5, 5, 6, 5, 5, 6, 6, 5, 5, 6, 5, 6, 5, 5, 6, 5, 5, 7, 5,
       7, 5, 6, 5, 7, 5, 6, 6, 6, 5, 7, 6, 6, 7, 5, 7, 5, 6, 6, 6, 5, 6,
       6, 5, 6, 5, 6, 6, 5, 6, 5, 6, 5, 6, 5, 5, 6, 6, 6, 6, 6, 5, 6, 5,
       7, 5, 6, 5, 6, 6, 6, 5, 5, 6, 6, 5, 6, 5, 5, 5, 6, 6, 5, 6, 6, 5,
       5, 6, 6, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 5, 6, 5, 6, 5, 6, 6, 5, 6,
       6, 6, 5, 6, 5, 6, 6, 6, 6, 5, 5, 6, 5, 5, 5, 5, 5, 5, 6, 5, 5, 6
```

In [9]:
```
accuracy= accuracy_score(y_test, predictions)
accuracy
```

Out[9]:
```
0.575
```

In [29]:
```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
x.shape
```

Out[29]:
```
(1599, 11)
```

In [11]:
```
mse = mean_squared_error(y_test, predictions)
mae = mean_absolute_error(y_test, predictions)
mse, mae
```

Out[11]:
```
(0.490625, 0.446875)
```

optimizing

In [12]:
```
logistic_model=LogisticRegression()
```

In [13]:
```
x=df.drop(['quality'],axis=1)
y=y=df['quality']

x_train, x_test, y_train, y_test= train_test_split(x,y, test_size=0.2, random_state=42)
scaler = preprocessing.StandardScaler().fit(x_train)

x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled=scaler.transform(x_test)

logistic_model= LogisticRegression()
logistic_model.fit(x_train_scaled, y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[13]:
```
LogisticRegression
LogisticRegression()
```

In [14]:
```
#paramater grid
param_grid = {
#     'C' [1.5]
     'penalty':['l2','l1','elasticnet'],
    'dual':[False, True],
    'fit_intercept':[True,False],
#     intercept_scaling=1,
    'solver':['liblinear','sag','saga','lbfgs', 'newton-cg'],
    'n_jobs':[-1]

}
```

In [15]:
```
grid_search=GridSearchCV(logistic_model,param_grid,cv=5)
grid_search.fit(x_train,y_train)
```

```
  File "C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py", lin
e 61, in _check_solver
    raise ValueError(
ValueError: Solver sag supports only dual=False, got dual=True

--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "C:\ProgramData\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.p
y", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py", line 1151, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py", lin
e 1168, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py", lin
e 61, in _check_solver
```

In [16]:
```
best_params= grid_search.best_params_
print(best_params)
```

```
{'dual': False, 'fit_intercept': True, 'n_jobs': -1, 'penalty': 'l2', 'solver': 'newton-cg'}
```

In [17]:
```
best_model=LogisticRegression(**best_params)
best_model.fit(x_train, y_train)
```

Out[17]:
```
LogisticRegression(n_jobs=-1, solver='newton-cg')
```

In [18]:
```
y_pred=best_model.predict(x_test)
accuracy=accuracy_score(y_test, y_pred)
accuracy
```

Out[18]:
```
0.571875
```

LinearRegression

In [34]:
```
#import libraries
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import f1_score, mean_squared_error, mean_absolute_error, r2_score
from sklearn import preprocessing
```

In [ ]:
```

```

In [35]:
```
df = pd.read_csv("C:\\Users\\DAVIS\\Desktop\\Pandas\\winequality\\winequality.csv")
```

In [38]:
```
scaler=MinMaxScaler()
scaler.fit(df)
dataset=pd.DataFrame(scaler.transform(df),columns=df.columns)
dataset.head()
```

Out[38]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qualit
0	0.247788	0.397260	0.00	0.068493	0.106845	0.140845	0.098940	0.567548	0.606299	0.137725	0.153846	0.
1	0.283186	0.520548	0.00	0.116438	0.143573	0.338028	0.215548	0.494126	0.362205	0.209581	0.215385	0.
2	0.283186	0.438356	0.04	0.095890	0.133556	0.197183	0.169611	0.508811	0.409449	0.191617	0.215385	0.
3	0.584071	0.109589	0.56	0.068493	0.105175	0.225352	0.190813	0.582232	0.330709	0.149701	0.215385	0.
4	0.247788	0.397260	0.00	0.068493	0.106845	0.140845	0.098940	0.567548	0.606299	0.137725	0.153846	0.

In [40]:
```
#split dataset
X=df.drop('quality', axis=1)
Qual= df['quality']

X_train, X_test, Qual_train,Qual_test= train_test_split(X, Qual, test_size=0.2, random_state=4
```

In [41]:
```
lr = LinearRegression()
lr.fit(X_train, Qual_train)
```

Out[41]:
```
LinearRegression
LinearRegression()
```

In [42]:
```
#test
predictions = lr.predict(X_test)

mse = mean_squared_error(Qual_test, predictions)
mae= mean_absolute_error(Qual_test, predictions)
r2 = r2_score(Qual_test, predictions)
```

In [47]:
```
print('MEAN SQUARFED ERROR:',mse, 'MEAN ABS ERROR:', mae, 'R2 SCORE:', r2)
print('Before Optimization')
```

```
MEAN SQUARFED ERROR: 0.490625 MEAN ABS ERROR: 0.503530441552466 R2 SCORE: 0.40318034127906854
Before Optimization
```

```
Fine Tunning
```

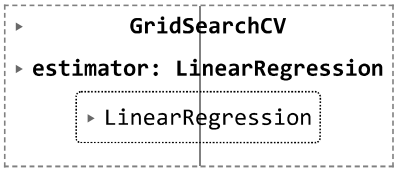
In [24]:
```
#parameters for tuning
para_grid = {
    'fit_intercept': [True, False],  # Whether to fit an intercept term
    'positive': [True, False],
    'copy_X': [True, False],
    'n_jobs': [None],


}
```

In [25]:
```
# Create GridSearchCV object
grid_search = GridSearchCV(lr, para_grid, cv=5, scoring='neg_mean_squared_error')  # Use negat

# Train the model with different hyperparameter combinations
grid_search.fit(X_train, Qual_train)
```

Out[25]:



In [26]:
```

# Get the best model with the lowest mean squared error
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
```

In [27]:
```
print(best_params)
```

```
{'copy_X': True, 'fit_intercept': False, 'n_jobs': None, 'positive': False}
```

In [50]:
```
# Make predictions on test set
y_pred = best_model.predict(X_test)

# Calculate mean squared error (MSE)
msee = mean_squared_error(Qual_test, y_pred)
mae = mean_absolute_error(Qual_test, y_pred)
rmse = np.sqrt(mse)  # Root Mean Squared Error (RMSE)
r2 = r2_score(Qual_test, y_pred)
# Print results
print('Best Parameters:', best_params)
print('MAE', mae)
print(f"Test MSE: {msee:.2f}")
print(f"Test RMSE: {rmse:.2f}")
print(r2_score)
```

```
Best Parameters: {'copy_X': True, 'fit_intercept': False, 'n_jobs': None, 'positive': False}
MAE 0.503363927068237
Test MSE: 0.39
Test RMSE: 0.70
<function r2_score at 0x0000026EFB1F00E0>
```

In [ ]:

In [ ]: