In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder,StandardScaler
import warnings
warnings.filterwarnings('ignore')
from sklearn.tree import DecisionTreeRegressor
from sklearn.feature_selection import RFE
from sklearn.preprocessing import MinMaxScaler,StandardScaler
import joblib
import datetime
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import cross_val_score,GridSearchCV
from sklearn.linear_model import Ridge,Lasso
```

In [2]:
```
data = pd.read_csv("exams.csv")
```

In [3]:
```
data
```

Out[3]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group D	some college	standard	completed	59.0	70.0	78.0
1	male	group D	associate's degree	standard	none	96.0	93.0	87.0
2	female	group D	some college	free/reduced	none	57.0	76.0	77.0
3	male	group B	some college	free/reduced	none	70.0	70.0	63.0
4	female	group D	associate's degree	standard	none	83.0	85.0	86.0
...	...	...	...	...	...	...	...	...
995	male	group C	some college	standard	none	77.0	77.0	71.0
996	male	group C	some college	standard	none	80.0	66.0	66.0
997	female	group A	high school	standard	completed	67.0	86.0	86.0
998	male	group E	high school	standard	none	80.0	72.0	62.0
999	male	group D	high school	standard	none	58.0	47.0	45.0

1000 rows × 8 columns

In [4]:
```
data.head()
```

Out[4]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group D	some college	standard	completed	59.0	70.0	78.0
1	male	group D	associate's degree	standard	none	96.0	93.0	87.0
2	female	group D	some college	free/reduced	none	57.0	76.0	77.0
3	male	group B	some college	free/reduced	none	70.0	70.0	63.0
4	female	group D	associate's degree	standard	none	83.0	85.0	86.0

In [5]:
```
data.tail()
```

Out[5]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
995	male	group C	some college	standard	none	77.0	77.0	71.0
996	male	group C	some college	standard	none	80.0	66.0	66.0
997	female	group A	high school	standard	completed	67.0	86.0	86.0
998	male	group E	high school	standard	none	80.0	72.0	62.0
999	male	group D	high school	standard	none	58.0	47.0	45.0

In [6]:
```
data.shape
```

Out[6]:
```
(1000, 8)
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   gender                       1000 non-null   object 
 1   race/ethnicity               1000 non-null   object 
 2   parental level of education  1000 non-null   object 
 3   lunch                        1000 non-null   object 
 4   test preparation course      1000 non-null   object 
 5   math score                   975 non-null    float64
 6   reading score                993 non-null    float64
 7   writing score                989 non-null    float64
dtypes: float64(3), object(5)
memory usage: 62.6+ KB
```

In [8]:
```
data.describe()
```

Out[8]:

	math score	reading score	writing score
count	975.000000	993.000000	989.000000
mean	67.841026	70.379658	69.165824
std	15.210716	14.108946	14.999555
min	15.000000	25.000000	15.000000
25%	58.000000	61.000000	59.000000
50%	68.000000	70.000000	70.000000
75%	79.000000	80.000000	80.000000
max	100.000000	100.000000	100.000000

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
gender                          0
race/ethnicity                  0
parental level of education     0
lunch                           0
test preparation course         0
math score                     25
reading score                   7
writing score                  11
dtype: int64
```

In [10]:
```
data['math score'].mean()
```

Out[10]:
```
67.84102564102564
```

In [11]:
```
data['math score'].fillna(data['math score'].mean(),inplace = True)
```

In [12]:
```
data['reading score'].mean()
```

Out[12]:
```
70.37965760322255
```

In [13]:
```
data['reading score'].fillna(data['reading score'].mean(),inplace = True)
```

In [14]:
```
data['writing score'].mean()
```

Out[14]:
```
69.16582406471183
```

In [15]:
```
data['writing score'].fillna(data['writing score'].mean(),inplace = True)
```

In [16]:

```
data.isnull().sum()
```

Out[16]:

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

In [17]:

```
mode_value = data['math score'].mode().iloc[0]
print("Mode:", mode_value)
```

```
Mode: 62.0
```

In [18]:

```
mode_value = data['reading score'].mode().iloc[0]
print("Mode:", mode_value)
```

```
Mode: 72.0
```

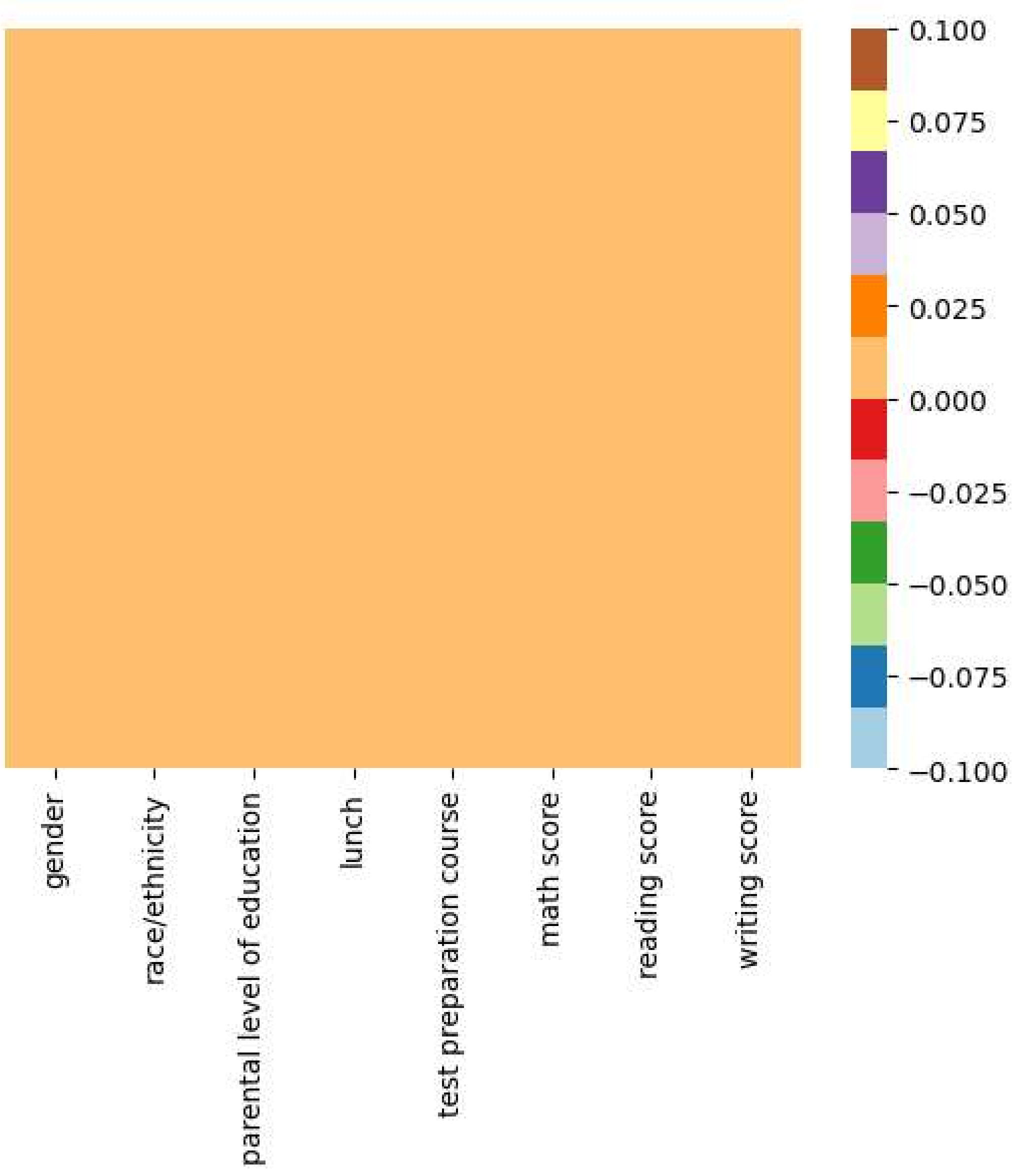
In [19]:

```
mode_value = data['writing score'].mode().iloc[0]
print("Mode:", mode_value)
```

```
Mode: 72.0
```

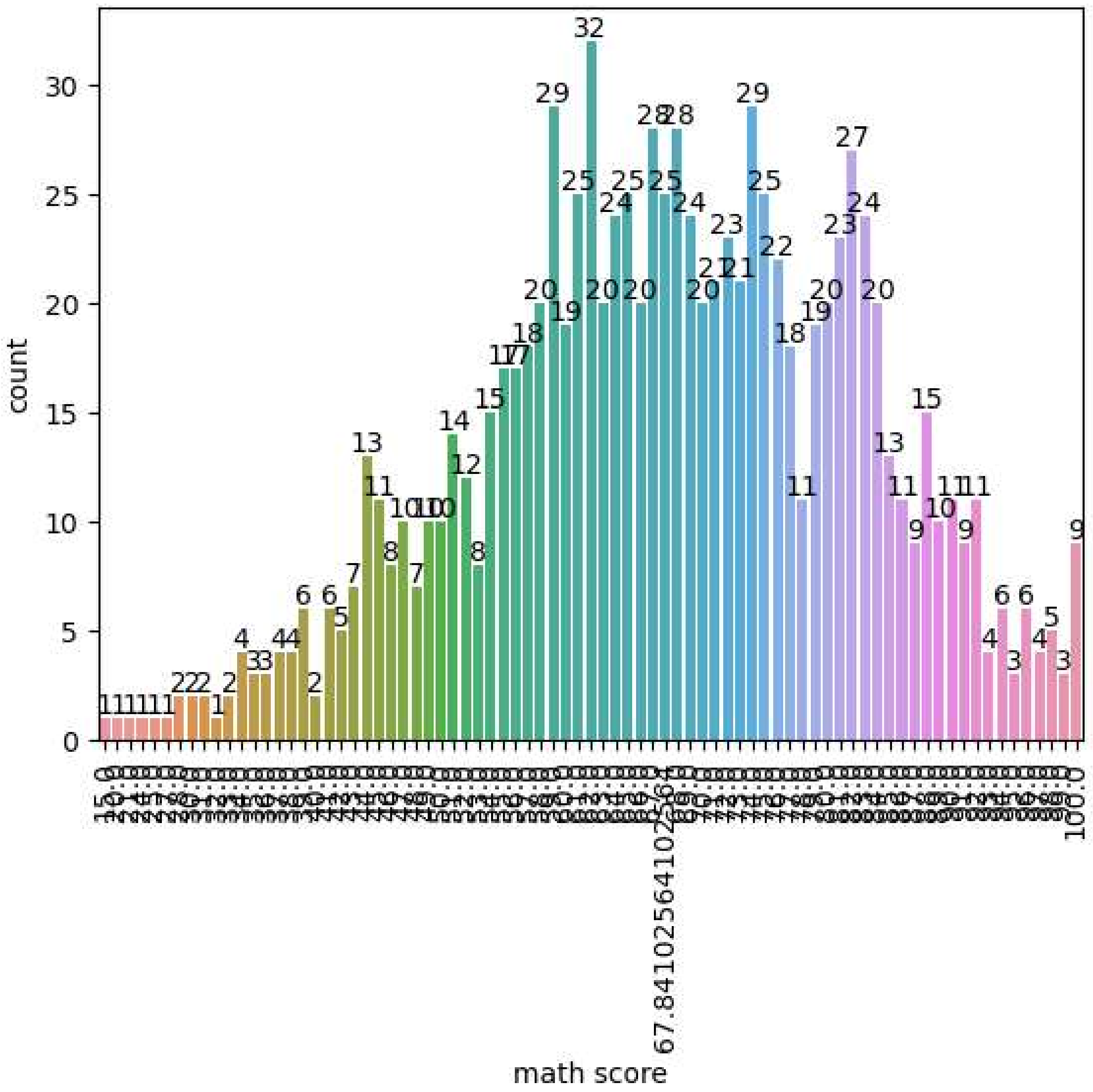
```
In [20]: sns.heatmap(data.isnull(),yticklabels=False,cmap="Paired")
```

```
Out[20]: <Axes: >
```



In [21]:

```
ax=sns.countplot(x='math score',data=data)
plt.xticks(rotation=90)
for bars in ax.containers:
 ax.bar_label(bars)
```

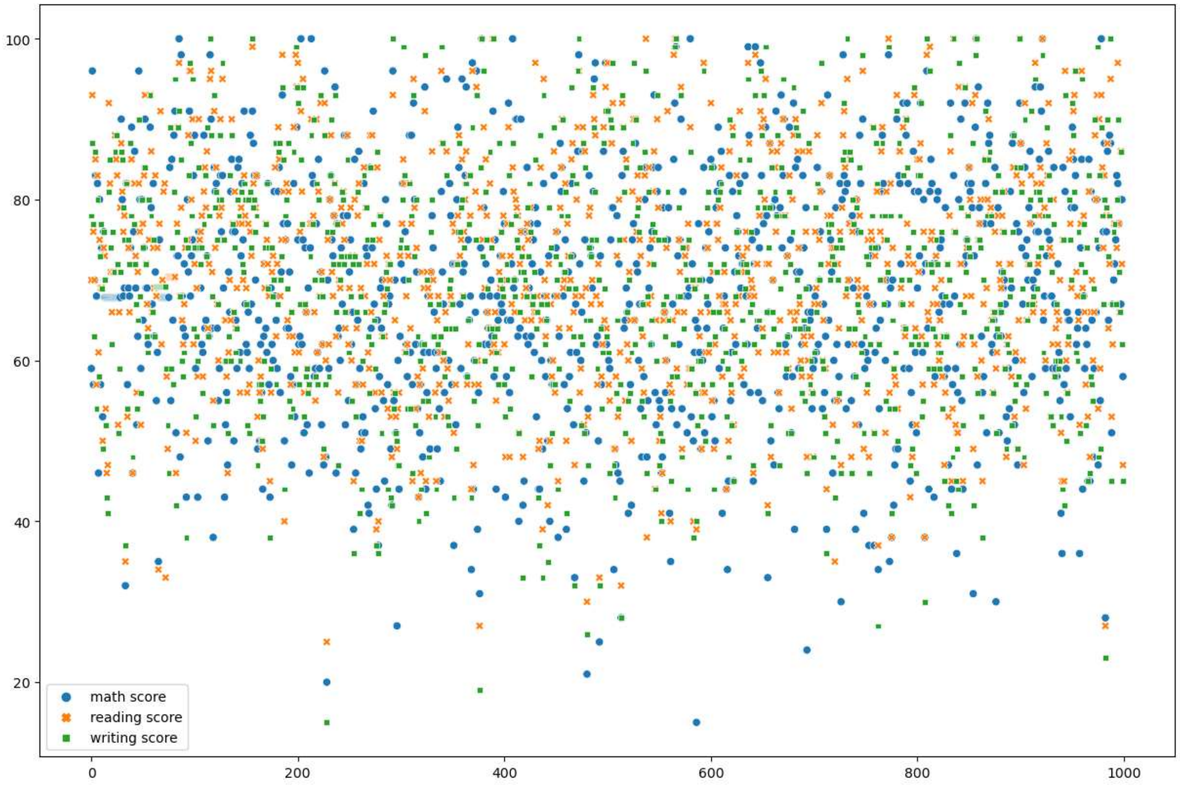


In [22]:

```
fig = plt.gcf();
fig.set_size_inches(15, 10);
sns.scatterplot(data)
```

Out[22]:

```
<Axes: >
```



In [23]:

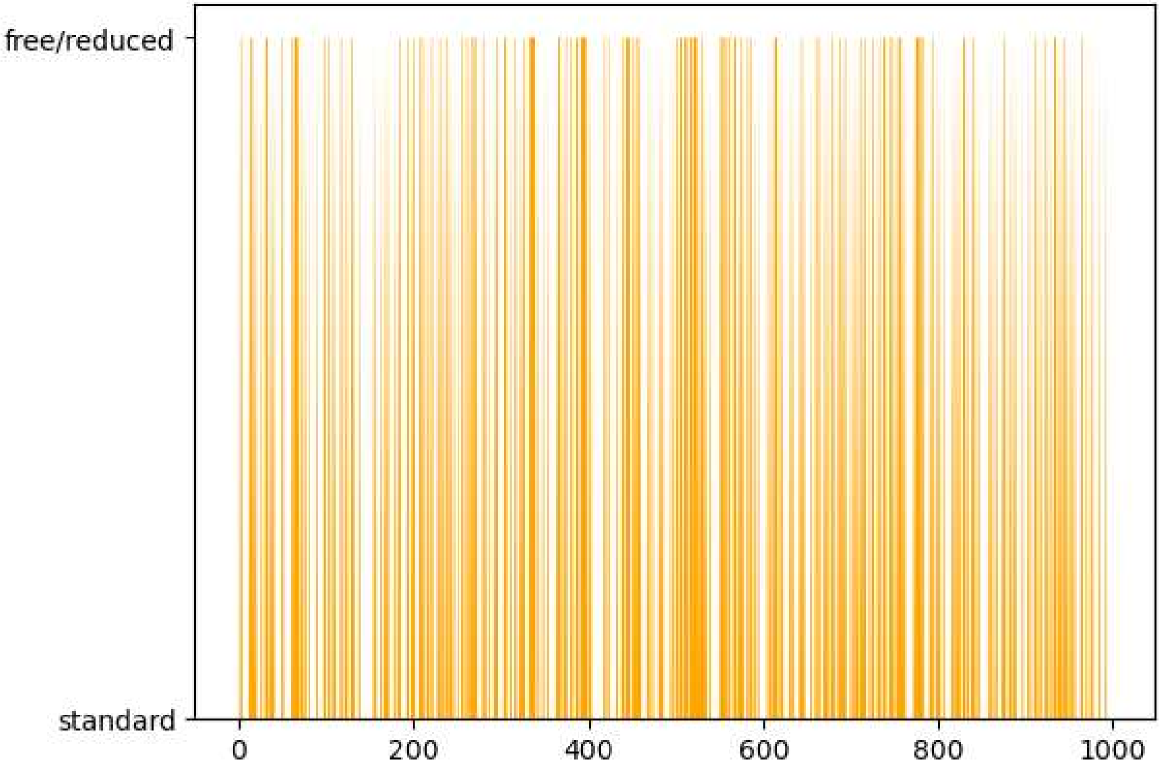
```
data
```

Out[23]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group D	some college	standard	completed	59.0	70.0	78.0
1	male	group D	associate's degree	standard	none	96.0	93.0	87.0
2	female	group D	some college	free/reduced	none	57.0	76.0	77.0
3	male	group B	some college	free/reduced	none	70.0	70.0	63.0
4	female	group D	associate's degree	standard	none	83.0	85.0	86.0
...	...	...	...	...	...	...	...	...
995	male	group C	some college	standard	none	77.0	77.0	71.0
996	male	group C	some college	standard	none	80.0	66.0	66.0
997	female	group A	high school	standard	completed	67.0	86.0	86.0
998	male	group E	high school	standard	none	80.0	72.0	62.0
999	male	group D	high school	standard	none	58.0	47.0	45.0

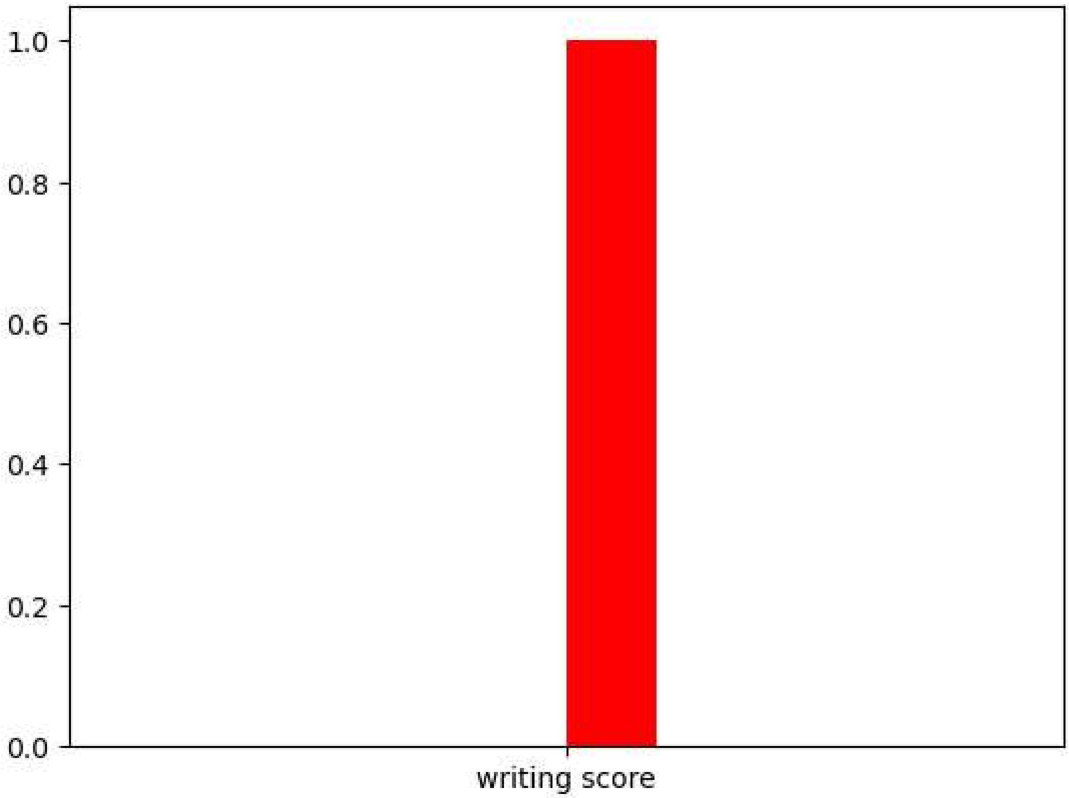
1000 rows × 8 columns

In [24]:
```
plt.stackplot(data.index,data.lunch,
 labels=['math score','reading score','writing score'],
 colors=['orange', 'green', 'red']);
```



In [25]:
```
plt.hist('writing score',color='r')
```

Out[25]:
```
(array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.]),
 array([-0.5, -0.4, -0.3, -0.2, -0.1,  0. ,  0.1,  0.2,  0.3,  0.4,  0.5]),
 <BarContainer object of 10 artists>)
```



In [26]:
```
encoder = LabelEncoder()
```

In [27]:
```
data['gender'] = encoder.fit_transform(data['gender'])
data['parental level of education'] = encoder.fit_transform(data['parental lev
data['lunch'] = encoder.fit_transform(data['lunch'])
```

In [28]:
```
data.shape
```

Out[28]:
```
(1000, 8)
```

In [29]:
```
categorical_col = ['gender','parental level of education','lunch']
encoder = OneHotEncoder(drop='first',sparse=False)
encoder_cols = pd.DataFrame(encoder.fit_transform(data[categorical_col]),colum
```

In [30]:

```
encoder_cols
```

Out[30]:

	gender_1	parental level of education_1	parental level of education_2	parental level of education_3	parental level of education_4	parental level of education_5	lunch_1
0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
1	1.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0
3	1.0	0.0	0.0	0.0	1.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...
995	1.0	0.0	0.0	0.0	1.0	0.0	1.0
996	1.0	0.0	0.0	0.0	1.0	0.0	1.0
997	0.0	0.0	1.0	0.0	0.0	0.0	1.0
998	1.0	0.0	1.0	0.0	0.0	0.0	1.0
999	1.0	0.0	1.0	0.0	0.0	0.0	1.0

1000 rows × 7 columns

In [31]:

```
data
```

Out[31]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	0	group D	4	1	completed	59.0	70.0	78.0
1	1	group D	0	1	none	96.0	93.0	87.0
2	0	group D	4	0	none	57.0	76.0	77.0
3	1	group B	4	0	none	70.0	70.0	63.0
4	0	group D	0	1	none	83.0	85.0	86.0
...	...	...	...	...	...	...	...	...
995	1	group C	4	1	none	77.0	77.0	71.0
996	1	group C	4	1	none	80.0	66.0	66.0
997	0	group A	2	1	completed	67.0	86.0	86.0
998	1	group E	2	1	none	80.0	72.0	62.0
999	1	group D	2	1	none	58.0	47.0	45.0

1000 rows × 8 columns

In [32]:
```
numerical_col = ['math score','reading score','writing score']
Scaled = StandardScaler()
Scaled= pd.DataFrame(Scaled.fit_transform(data[numerical_col]),columns=numeric
```

In [33]:
```
Scaled
```

Out[33]:

	math score	reading score	writing score
0	-0.588943	-0.027017	0.592528
1	1.875805	1.609714	1.196179
2	-0.722173	0.399956	0.525456
3	0.143820	-0.027017	-0.413556
4	1.009812	1.040416	1.129107
...	...	...	...
995	0.610123	0.471118	0.123022
996	0.809968	-0.311666	-0.212339
997	-0.056025	1.111578	1.129107
998	0.809968	0.115307	-0.480628
999	-0.655558	-1.663749	-1.620857

1000 rows × 3 columns

In [34]:
```
x = pd.concat([encoder_cols,Scaled],axis=1)
Y = data['gender']
```

In [35]:
```
x.shape
```

Out[35]:
```
(1000, 10)
```

In [36]:
```
Y
```

Out[36]:
```
0      0
1      1
2      0
3      1
4      0
      ..
995    1
996    1
997    0
998    1
999    1
Name: gender, Length: 1000, dtype: int32
```

In [37]:
```
X_train,X_test,Y_train,Y_test=train_test_split(x,Y,test_size=0.2,random_state=
```

In [38]:

```
model=LinearRegression()
model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
```

In [39]:

```
model = LinearRegression()
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)
```

In [40]:

```
from sklearn.metrics import r2_score, mean_squared_error
```

In [41]:

```
R2= r2_score(Y_test,y_pred)
```

In [42]:

```
mae = mean_absolute_error(Y_test,y_pred)
mse = mean_squared_error(Y_test,y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(Y_test,y_pred)
```

In [43]:

```
print('Mean Absolute Error',mae)
print('Mean Squared error',rmse)
print('Root Mean Squared Error',rmse)
print('R2 Score',r2)
```

```
Mean Absolute Error 7.844146439922244e-15
Mean Squared error 9.906270116153703e-15
Root Mean Squared Error 9.906270116153703e-15
R2 Score 1.0
```

In [ ]:

```
lr_model.fit(X_train,Y_train)
lr_prediction = lr_model.predict(X_test)
lr_mae = mean_absolute_error(Y_test,lr_prediction)
lr_mse = mean_squared_error(Y_test,lr_prediction)
lr_r2 = r2_score(Y_test,lr_prediction)
```

In [ ]:

```
print('Linear MAE',lr_mae)
print('Linear MSE',lr_mse)
print('Linear R2',lr_r2)
```

In [44]:

```
lr_model = LinearRegression()
lr_scores = cross_val_score(lr_model,X_train,Y_train,cv=5)
```

In [45]:

```
ridge_model= Ridge(alpha=1.0)
ridge_scores = cross_val_score(ridge_model , X_train , Y_train , cv = 5)
```

In [46]:

```
Lasso_model= Lasso(alpha=1.0)
Lasso_scores = cross_val_score(Lasso_model , X_train , Y_train , cv = 5)
```

In [47]:
```
Lasso_model.fit(X_train , Y_train)
Lasso_prediction = Lasso_model.predict(X_test)
Lasso_mae = mean_absolute_error(Y_test ,Lasso_prediction)
Lasso_mse = mean_squared_error(Y_test ,Lasso_prediction)
Lasso_r2 = r2_score(Y_test ,Lasso_prediction)
```

In [48]:
```
print('Lasso MAE',Lasso_mae)
print('Lasso MSE',Lasso_mse)
print('Lasso R2',Lasso_r2)
```

```
Lasso MAE 0.49981250000000005
Lasso MSE 0.24982656250000002
Lasso R2 -0.0018107769423560096
```

In [49]:
```
ridge_model.fit(X_train , Y_train)
ridge_prediction = ridge_model.predict(X_test)
ridge_mae = mean_absolute_error(Y_test ,ridge_prediction)
ridge_mse = mean_squared_error(Y_test ,ridge_prediction)
ridge_r2 = r2_score(Y_test ,ridge_prediction)
```

In [50]:
```
print('Lasso MAE',ridge_mae)
print('Lasso MSE',ridge_mse)
print('Lasso R2',ridge_r2)
```

```
Lasso MAE 0.002791417359187404
Lasso MSE 1.1679906764055893e-05
Lasso R2 0.9999531632811467
```

In [53]:
```
from sklearn.linear_model import HuberRegressor
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_scaled = scaler.fit_transform(X_test)
huber = HuberRegressor(epsilon=1.35)
huber.fit(X_scaled, Y_test)
huber_prediction = huber.predict(X_scaled)
huber_mae =mean_absolute_error(Y_test,huber_prediction)
huber_mse =mean_squared_error(Y_test,huber_prediction)
huber_rmse = np.sqrt(huber_mse)
huber_r2 = r2_score(Y_test,huber_prediction)
print('huber mae:',huber_mae)
print('huber mse:',huber_mse)
print('huber rmse:',huber_rmse)
print('huber r2:',huber_r2)
```

```
huber mae: 1.8375633237255328e-11
huber mse: 5.549402194005609e-22
huber rmse: 2.3557169172049533e-11
huber r2: 1.0
```

In [ ]: