In [7]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split


# Modelling
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression, Ridge,Lasso
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.model_selection import RandomizedSearchCV
from catboost import CatBoostRegressor
from xgboost import XGBRegressor
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
df = pd.read_csv('data/stud.csv')
df.sample(2)
```

Out[4]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	math_score	reading
203	female	group B	associate's degree	standard	none	57	
357	female	group C	some college	free/reduced	completed	42	

In [5]:

```
X = df.drop(columns=['math_score'],axis=1)
y = df['math_score']
```

In [8]:

```
num_features = X.select_dtypes(exclude="object").columns
cat_features = X.select_dtypes(include="object").columns

numeric_transformer = StandardScaler()
oh_transformer = OneHotEncoder()

preprocessor = ColumnTransformer(
    [
        ("OneHotEncoder", oh_transformer, cat_features),
         ("StandardScaler", numeric_transformer, num_features),
    ]
)
```

In [9]:

```
X = preprocessor.fit_transform(X)
```

In [10]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
X_train.shape, X_test.shape
```

Out[10]:

```
((800, 19), (200, 19))
```

In [13]:

```
def evaluate_model(true, predicted):
    mae = mean_absolute_error(true, predicted)
    mse = mean_squared_error(true, predicted)
    rmse = np.sqrt(mean_squared_error(true, predicted))
```

```
    r2_square = r2_score(true, predicted)
    return mae, rmse, r2_square
```

In [14]:
```
models = {
    "Linear Regression": LinearRegression(),
    "Lasso": Lasso(),
    "Ridge": Ridge(),
    "K-Neighbors Regressor": KNeighborsRegressor(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest Regressor": RandomForestRegressor(),
    "XGBRegressor": XGBRegressor(),
    "CatBoosting Regressor": CatBoostRegressor(verbose=False),
    "AdaBoost Regressor": AdaBoostRegressor()
}
model_list = []
r2_list =[]

for i in range(len(list(models))):
    model = list(models.values())[i]
    model.fit(X_train, y_train) # Train model

    # Make predictions
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    # Evaluate Train and Test dataset
    model_train_mae , model_train_rmse, model_train_r2 = evaluate_model(y_train, y_train_pred)

    model_test_mae, model_test_rmse, model_test_r2 = evaluate_model(y_test, y_test_pred)


    print(list(models.keys())[i])
    model_list.append(list(models.keys())[i])

    print('Model performance for Training set')
    print("- Root Mean Squared Error: {:.4f}".format(model_train_rmse))
    print("- Mean Absolute Error: {:.4f}".format(model_train_mae))
    print("- R2 Score: {:.4f}".format(model_train_r2))

    print('----------------------------------')

    print('Model performance for Test set')
    print("- Root Mean Squared Error: {:.4f}".format(model_test_rmse))
    print("- Mean Absolute Error: {:.4f}".format(model_test_mae))
    print("- R2 Score: {:.4f}".format(model_test_r2))
    r2_list.append(model_test_r2)

    print('='*35)
    print('\n')
```

```
Linear Regression
Model performance for Training set
- Root Mean Squared Error: 5.3231
- Mean Absolute Error: 4.2667
- R2 Score: 0.8743
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 5.3940
- Mean Absolute Error: 4.2148
- R2 Score: 0.8804
===================================


Lasso
Model performance for Training set
- Root Mean Squared Error: 6.5938
- Mean Absolute Error: 5.2063
- R2 Score: 0.8071
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.5197
- Mean Absolute Error: 5.1579
- R2 Score: 0.8253
===================================


Ridge
Model performance for Training set
- Root Mean Squared Error: 5.3233
- Mean Absolute Error: 4.2650
- R2 Score: 0.8743
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 5.3904
- Mean Absolute Error: 4.2111
- R2 Score: 0.8806
===================================


K-Neighbors Regressor
Model performance for Training set
- Root Mean Squared Error: 5.7088
- Mean Absolute Error: 4.5177
- R2 Score: 0.8554
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 7.2494
- Mean Absolute Error: 5.6090
- R2 Score: 0.7840
===================================


Decision Tree
Model performance for Training set
- Root Mean Squared Error: 0.2795
- Mean Absolute Error: 0.0187
- R2 Score: 0.9997
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 8.0675
- Mean Absolute Error: 6.4550
- R2 Score: 0.7325
===================================
```

```
Random Forest Regressor
Model performance for Training set
- Root Mean Squared Error: 2.3225
- Mean Absolute Error: 1.8622
- R2 Score: 0.9761
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.0092
- Mean Absolute Error: 4.6202
- R2 Score: 0.8516
===================================


XGBRegressor
Model performance for Training set
- Root Mean Squared Error: 1.0073
- Mean Absolute Error: 0.6875
- R2 Score: 0.9955
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.4733
- Mean Absolute Error: 5.0577
- R2 Score: 0.8278
===================================


CatBoosting Regressor
Model performance for Training set
- Root Mean Squared Error: 3.0427
- Mean Absolute Error: 2.4054
- R2 Score: 0.9589
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.0086
- Mean Absolute Error: 4.6125
- R2 Score: 0.8516
===================================


AdaBoost Regressor
Model performance for Training set
- Root Mean Squared Error: 5.7967
- Mean Absolute Error: 4.7360
- R2 Score: 0.8510
----------------------------------
Model performance for Test set
- Root Mean Squared Error: 6.0402
- Mean Absolute Error: 4.6618
- R2 Score: 0.8501
===================================
```

In [15]:

```
pd.DataFrame(list(zip(model_list, r2_list)), columns=['Model Name', 'R2_Score']).sort_values(by=
```

Out[15]:

	Model Name	R2_Score
2	Ridge	0.880593
0	Linear Regression	0.880433
7	CatBoosting Regressor	0.851632
5	Random Forest Regressor	0.851603
8	AdaBoost Regressor	0.850071
6	XGBRegressor	0.827797
1	Lasso	0.825320
3	K-Neighbors Regressor	0.784030
4	Decision Tree	0.732533

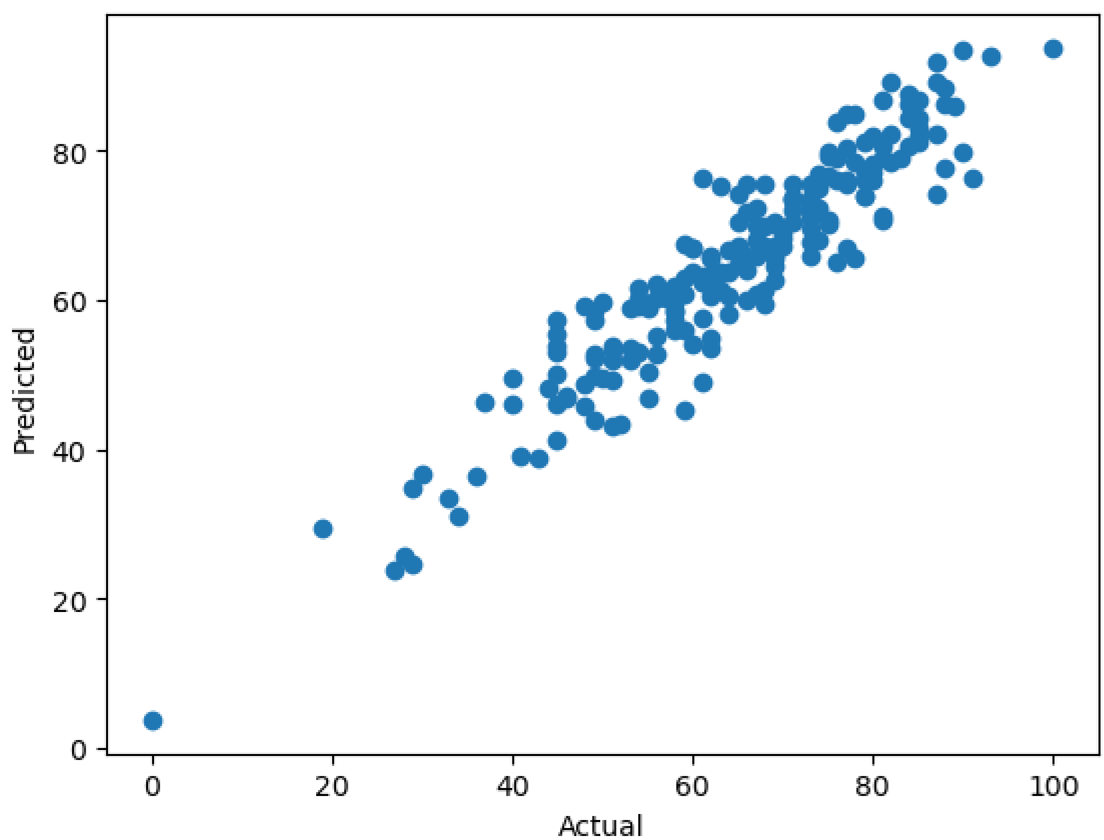
In [16]:

```
lin_model = LinearRegression(fit_intercept=True)
lin_model = lin_model.fit(X_train, y_train)
y_pred = lin_model.predict(X_test)
score = r2_score(y_test, y_pred)*100
print(" Accuracy of the model is %.2f" %score)
```

```
 Accuracy of the model is 88.04
```

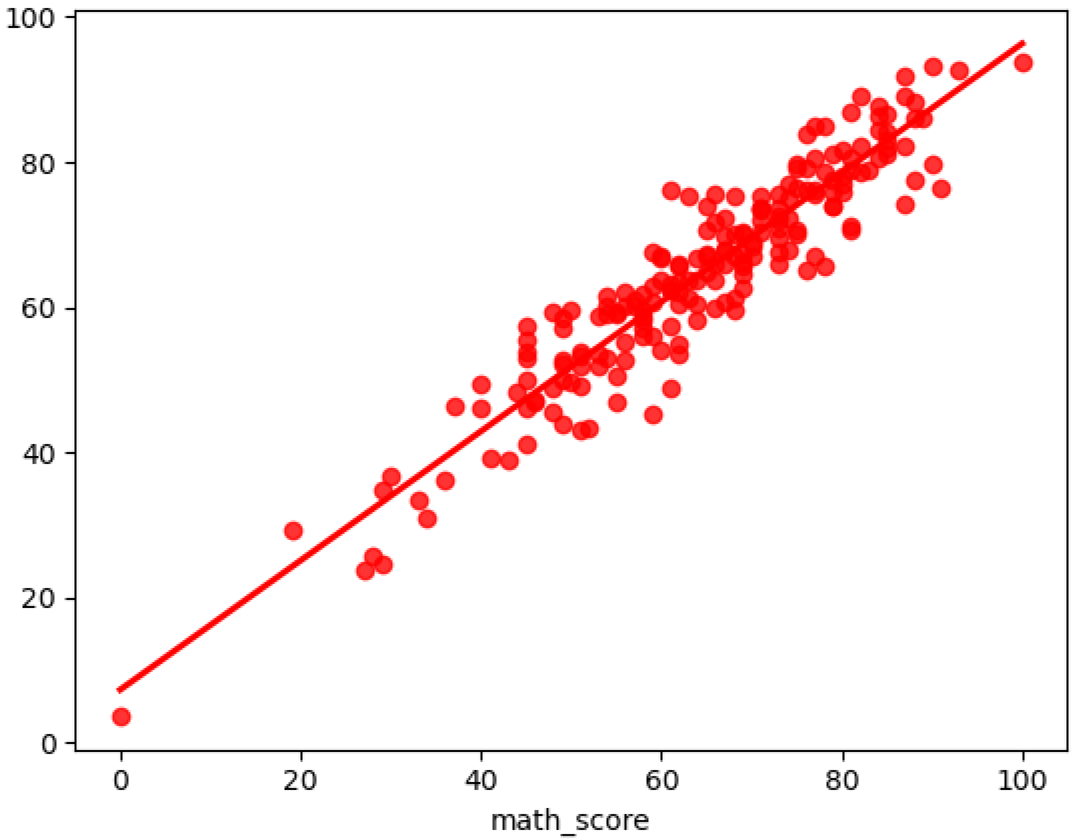
In [17]:

```
plt.scatter(y_test,y_pred);
plt.xlabel('Actual');
plt.ylabel('Predicted');
```



In [18]:

```
sns.regplot(x=y_test,y=y_pred,ci=None,color ='red');
```



In [19]:
```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred,'Difference':y_test-y_pred}
pred_df
```

Out[19]:

	Actual Value	Predicted Value	Difference
521	91	76.387970	14.612030
737	53	58.885970	-5.885970
740	80	76.990265	3.009735
660	74	76.851804	-2.851804
411	84	87.627378	-3.627378
...	...	...	...
408	52	43.409149	8.590851
332	62	62.152214	-0.152214
208	74	67.888395	6.111605
613	65	67.022287	-2.022287
78	61	62.345132	-1.345132

200 rows × 3 columns

In [ ]: