In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import seaborn as sns

from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from pathlib import Path
from matplotlib.gridspec import GridSpec
from scipy.signal import savgol_filter
```

In [2]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
id_1 = train["id"]
id_t = test["id"]
Xt = test["study_hours"]
X2t = test["class_attendance"]
train.columns
```

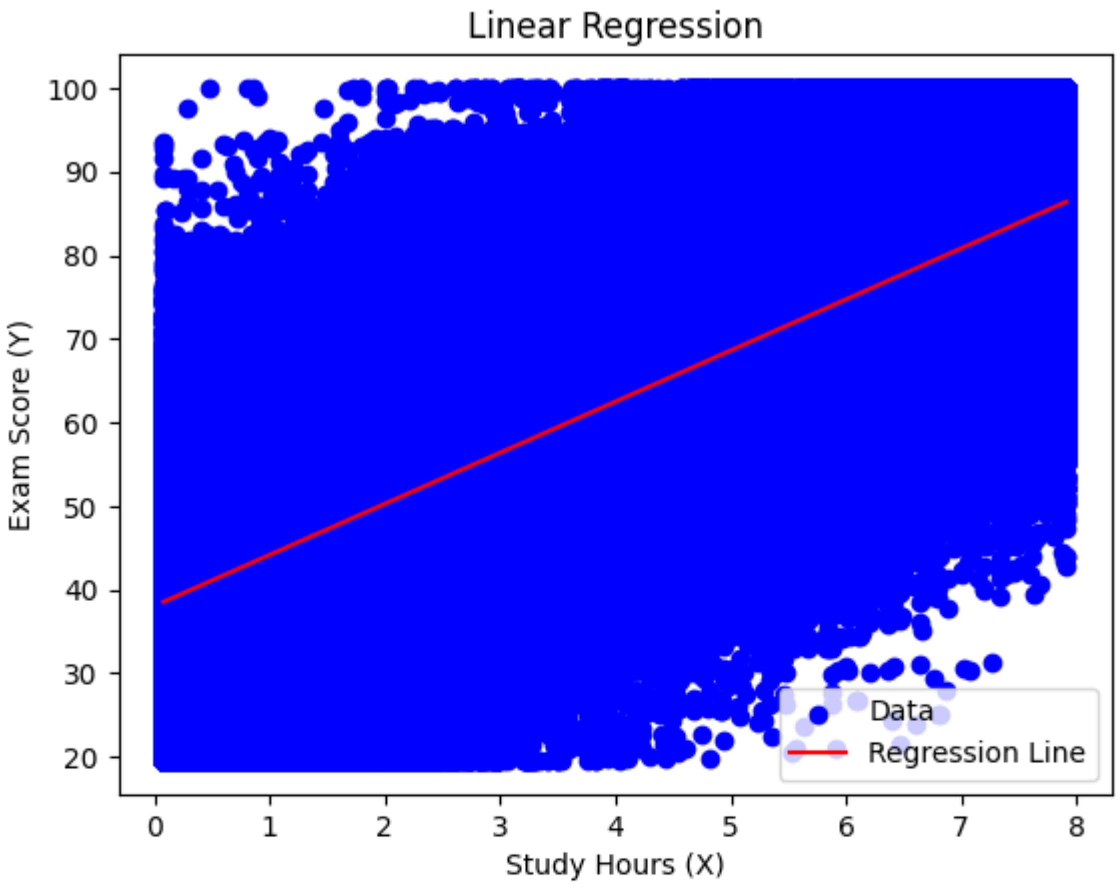
Out[2]:
```
Index(['id', 'age', 'gender', 'course', 'study_hours', 'class_attendanc
e',
       'internet_access', 'sleep_hours', 'sleep_quality', 'study_metho
d',
       'facility_rating', 'exam_difficulty', 'exam_score'],
      dtype='str')
```

In [3]:
```
train[["study_hours", "class_attendance", "exam_score"]].corr()["exam_sco
```

Out[3]:
```
study_hours         0.762267
class_attendance    0.360954
exam_score          1.000000
Name: exam_score, dtype: float64
```

In [4]:
```
X = train["study_hours"].values.reshape(-1, 1)
Y = train["exam_score"]

model = LinearRegression()
model.fit(X, Y)

Y_pred_1 = model.predict(X)

sorted_idx = np.argsort(X.flatten())
X_sorted = X.flatten()[sorted_idx]
Y_pred_sorted = Y_pred_1[sorted_idx]

plt.scatter(X, Y, color='blue', label='Data')
plt.plot(X_sorted, Y_pred_sorted, color='red', label='Regression Line')

plt.title("Linear Regression")
plt.xlabel("Study Hours (X)")
plt.ylabel("Exam Score (Y)")
plt.legend()
plt.show()
```

```
/home/avveer-singh-khurana/anscombe-project/venv/lib/python3.12/site-packa
ges/IPython/core/pylabtools.py:170: UserWarning: Creating legend with loc=
"best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
In [5]: pred_df = np.column_stack((X.flatten(), Y_pred_1))
```

```
In [6]: def predict_grade(study_hours):
            study_hours = np.array(study_hours).reshape(-1, 1)
            y_pred = model.predict(study_hours)
            y_pred = np.round(y_pred, 1)
            return float(y_pred[0])

        Y_pred_2 = []
        for i in range(len(id_t)):
            j = Xt[i]
            Y_pred_2.append(predict_grade(j))
```

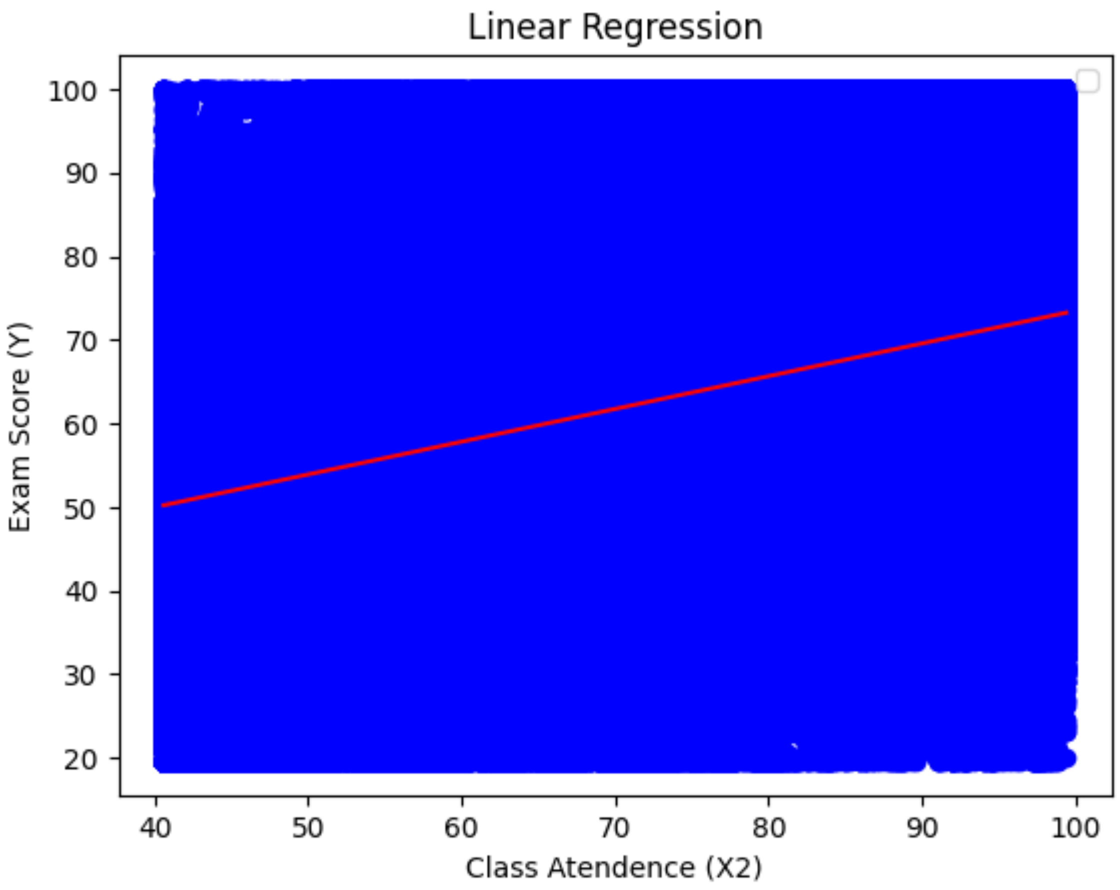
```
In [7]: print(Y_pred_2[:20])
```

```
[79.9, 78.4, 78.4, 56.6, 50.5, 76.1, 60.0, 57.2, 77.7, 85.2, 49.2, 47.9, 4
0.7, 65.8, 40.4, 78.3, 73.1, 64.8, 73.4, 66.5]
```

```
In [8]: X2 = train["class_attendance"].values.reshape(-1, 1)
        Y = train["exam_score"]

        model = LinearRegression()
        model.fit(X2, Y)

        Y_pred_3 = model.predict(X2)

        sorted_idx2 = np.argsort(X2.flatten())
        X_sorted2 = X2.flatten()[sorted_idx2]
```

```
Y_pred_sorted2 = Y_pred_3[sorted_idx2]

plt.scatter(X2, Y, color='blue')
plt.plot(X_sorted2, Y_pred_sorted2, color='red')

plt.title("Linear Regression")
plt.xlabel("Class Atendence (X2)")
plt.ylabel("Exam Score (Y)")
plt.legend()
plt.show()
```

```
/tmp/ipykernel_52932/1333310726.py:19: UserWarning: No artists with labels
found to put in legend.  Note that artists whose label start with an under
score are ignored when legend() is called with no argument.
  plt.legend()
```



In [9]:
```
def predict_grade2(class_attendence):
    class_attendence = np.array(class_attendence).reshape(-1, 1)
    y_pred4 = model.predict(class_attendence)
    y_pred4 = np.round(y_pred4, 1)
    return float(y_pred4[0])

Y_pred_5 = []
for i in range(len(id_t)):
    j = X2t[i]
    Y_pred_5.append(predict_grade(j))
```

In [10]:
```
print(Y_pred_5[:20])
```

```
[59.8, 51.9, 72.9, 60.3, 50.9, 56.0, 70.2, 66.7, 68.0, 71.8, 65.9, 54.7, 6
7.2, 61.9, 71.3, 57.1, 65.7, 66.4, 61.6, 60.6]
```

In [19]:
```
Y_pred_67 = []
```

```
for i in range(len(id_t)):
    if Y_pred_5[i] <= 75:
        _67 = ((Y_pred_2[i] * 0.762) + (Y_pred_5[i] * 0.238))
        Y_pred_67.append(_67)
    else:
        _67 = ((Y_pred_2[i] * 0.9) + (Y_pred_5[i] * 0.1))
        Y_pred_67.append(_67)
Y_pred_67 = np.round(Y_pred_67,1)
```

In [20]:
```
print(Y_pred_67[:20])
```

```
[75.1 72.1 77.1 57.5 50.6 71.3 62.4 59.5 75.4 82.  53.2 49.5 47.  64.9
 47.8 73.3 71.3 65.2 70.6 65.1]
```

In [21]:
```
lock_in_submission = pd.DataFrame({
    'id': id_t,
    'exam_score': Y_pred_67
})

lock_in_submission.to_csv('lock_in_submission.csv', index=False)
lock_in_submission.head
```

Out[21]:
```
<bound method NDFrame.head of              id  exam_score
0        630000        75.1
1        630001        72.1
2        630002        77.1
3        630003        57.5
4        630004        50.6
...         ...         ...
269995   899995        56.7
269996   899996        43.7
269997   899997        75.0
269998   899998        61.0
269999   899999        70.0

[270000 rows x 2 columns]>
```

In [ ]: