Machine Learning Model

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

[2]:
```
data=pd.read_csv(r"C:\Users\Lenovo\Desktop\Machine Learning\student-mat.csv")
```

[3]:
```
data.head()
```

[3]:
```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  ... \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  ...
1     GP   F   17       U     GT3       T     1     1  at_home     other  ...
2     GP   F   15       U     LE3       T     1     1  at_home     other  ...
3     GP   F   15       U     GT3       T     4     2   health  services  ...
4     GP   F   16       U     GT3       T     3     3    other     other  ...

  famrel  freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4         3      4     1     1      3        6   5   6   6
1      5         3      3     1     1      3        4   5   5   6
2      4         3      2     2     3      3       10   7   8  10
3      3         2      2     1     1      5        2  15  14  15
4      4         3      2     1     2      5        4   6  10  10

[5 rows x 33 columns]
```

[4]:
```
data.shape
```

[4]:
```
(395, 33)
```

[5]:
```
data.describe()
```

[5]:
```
              age        Medu        Fedu  traveltime   studytime    failures  \
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000
mean    16.696203    2.749367    2.521519    1.448101    2.035443    0.334177
std      1.276043    1.094735    1.088201    0.697505    0.839240    0.743651
min     15.000000    0.000000    0.000000    1.000000    1.000000    0.000000
25%     16.000000    2.000000    2.000000    1.000000    1.000000    0.000000
```

```
50%     17.000000    3.000000    2.000000    1.000000    2.000000    0.000000
75%     18.000000    4.000000    3.000000    2.000000    2.000000    0.000000
max     22.000000    4.000000    4.000000    4.000000    4.000000    3.000000

           famrel    freetime       goout        Dalc        Walc      health  \
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000
mean     3.944304    3.235443    3.108861    1.481013    2.291139    3.554430
std      0.896659    0.998862    1.113278    0.890741    1.287897    1.390303
min      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
25%      4.000000    3.000000    2.000000    1.000000    1.000000    3.000000
50%      4.000000    3.000000    3.000000    1.000000    2.000000    4.000000
75%      5.000000    4.000000    4.000000    2.000000    3.000000    5.000000
max      5.000000    5.000000    5.000000    5.000000    5.000000    5.000000

         absences          G1          G2          G3
count  395.000000  395.000000  395.000000  395.000000
mean     5.708861   10.908861   10.713924   10.415190
std      8.003096    3.319195    3.761505    4.581443
min      0.000000    3.000000    0.000000    0.000000
25%      0.000000    8.000000    9.000000    8.000000
50%      4.000000   11.000000   11.000000   11.000000
75%      8.000000   13.000000   13.000000   14.000000
max     75.000000   19.000000   19.000000   20.000000
```

[6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

[7]:

```
data.corr()
```

[7]:

```
                 age      Medu      Fedu  traveltime  studytime  failures  \
age         1.000000 -0.163658 -0.163438    0.070641  -0.004140  0.243665
Medu       -0.163658  1.000000  0.623455   -0.171639   0.064944 -0.236680
Fedu       -0.163438  0.623455  1.000000   -0.158194  -0.009175 -0.250408
traveltime  0.070641 -0.171639 -0.158194    1.000000  -0.100909  0.092239
studytime  -0.004140  0.064944 -0.009175   -0.100909   1.000000 -0.173563
failures    0.243665 -0.236680 -0.250408    0.092239  -0.173563  1.000000
famrel      0.053940 -0.003914 -0.001370   -0.016808   0.039731 -0.044337
freetime    0.016434  0.030891 -0.012846   -0.017025  -0.143198  0.091987
goout       0.126964  0.064094  0.043105    0.028540  -0.063904  0.124561
Dalc        0.131125  0.019834  0.002386    0.138325  -0.196019  0.136047
Walc        0.117276 -0.047123 -0.012631    0.134116  -0.253785  0.141962
health     -0.062187 -0.046878  0.014742    0.007501  -0.075616  0.065827
absences    0.175230  0.100285  0.024473   -0.012944  -0.062700  0.063726
G1         -0.064081  0.205341  0.190270   -0.093040   0.160612 -0.354718
G2         -0.143474  0.215527  0.164893   -0.153198   0.135880 -0.355896
G3         -0.161579  0.217147  0.152457   -0.117142   0.097820 -0.360415

              famrel  freetime     goout      Dalc      Walc    health  \
age         0.053940  0.016434  0.126964  0.131125  0.117276 -0.062187
Medu       -0.003914  0.030891  0.064094  0.019834 -0.047123 -0.046878
Fedu       -0.001370 -0.012846  0.043105  0.002386 -0.012631  0.014742
traveltime -0.016808 -0.017025  0.028540  0.138325  0.134116  0.007501
studytime   0.039731 -0.143198 -0.063904 -0.196019 -0.253785 -0.075616
failures   -0.044337  0.091987  0.124561  0.136047  0.141962  0.065827
```

```
famrel      1.000000  0.150701  0.064568 -0.077594 -0.113397  0.094056
freetime    0.150701  1.000000  0.285019  0.209001  0.147822  0.075733
goout       0.064568  0.285019  1.000000  0.266994  0.420386 -0.009577
Dalc       -0.077594  0.209001  0.266994  1.000000  0.647544  0.077180
Walc       -0.113397  0.147822  0.420386  0.647544  1.000000  0.092476
health      0.094056  0.075733 -0.009577  0.077180  0.092476  1.000000
absences   -0.044354 -0.058078  0.044302  0.111908  0.136291 -0.029937
G1          0.022168  0.012613 -0.149104 -0.094159 -0.126179 -0.073172
G2         -0.018281 -0.013777 -0.162250 -0.064120 -0.084927 -0.097720
G3          0.051363  0.011307 -0.132791 -0.054660 -0.051939 -0.061335

            absences        G1        G2        G3
age         0.175230 -0.064081 -0.143474 -0.161579
Medu        0.100285  0.205341  0.215527  0.217147
Fedu        0.024473  0.190270  0.164893  0.152457
traveltime -0.012944 -0.093040 -0.153198 -0.117142
studytime  -0.062700  0.160612  0.135880  0.097820
failures    0.063726 -0.354718 -0.355896 -0.360415
famrel     -0.044354  0.022168 -0.018281  0.051363
freetime   -0.058078  0.012613 -0.013777  0.011307
goout       0.044302 -0.149104 -0.162250 -0.132791
Dalc        0.111908 -0.094159 -0.064120 -0.054660
Walc        0.136291 -0.126179 -0.084927 -0.051939
health     -0.029937 -0.073172 -0.097720 -0.061335
absences    1.000000 -0.031003 -0.031777  0.034247
G1         -0.031003  1.000000  0.852118  0.801468
G2         -0.031777  0.852118  1.000000  0.904868
G3          0.034247  0.801468  0.904868  1.000000
```

```
[ ]:
```

```
[8]: df=data[['Dalc','Walc','G1','G2','G3']]
```

```
[24]: print(df)
```

```
     Dalc  Walc  G1  G2  G3
0       1     1   5   6   6
1       1     1   5   5   6
2       2     3   7   8  10
3       1     1  15  14  15
4       1     2   6  10  10
..    ...   ...  ..  ..  ..
390     4     5   9   9   9
391     3     4  14  16  16
392     3     3  10   8   7
393     3     4  11  12  10
394     3     3   8   9   9
```

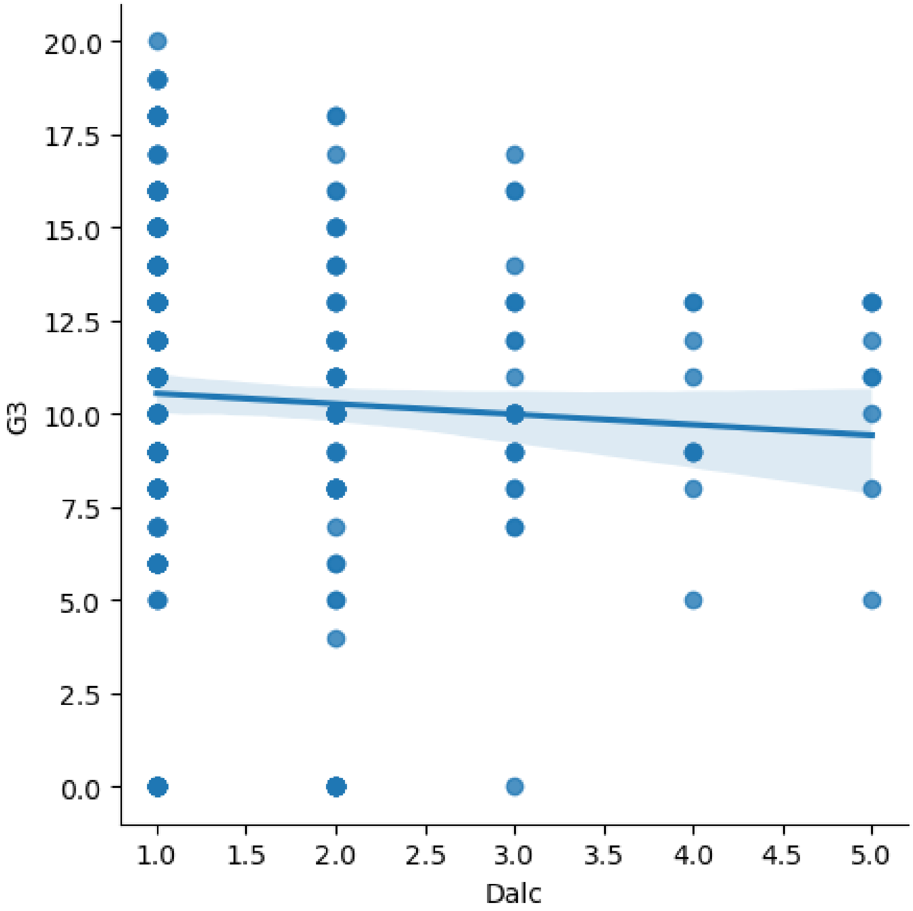
```
[395 rows x 5 columns]
```

[9]:

```
X=df.drop('G3',axis=1)
y=df['G3']
```

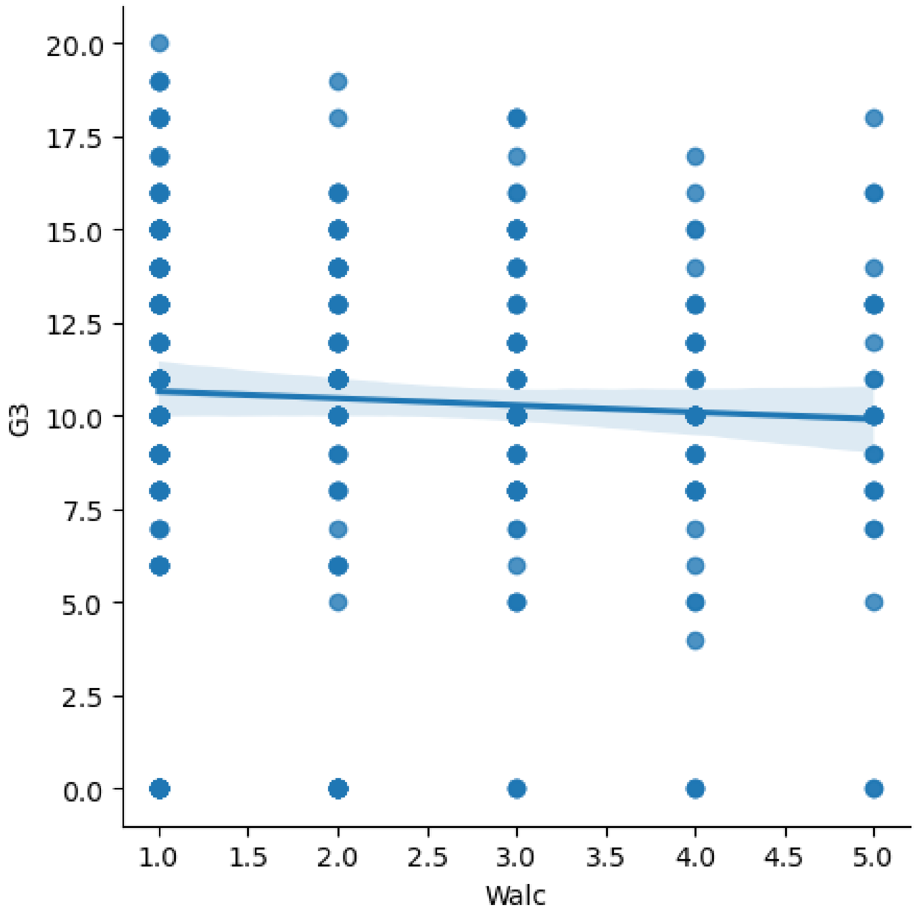
[10]:

```
sns.lmplot(x='Dalc',y='G3',data=df)
plt.show()
```



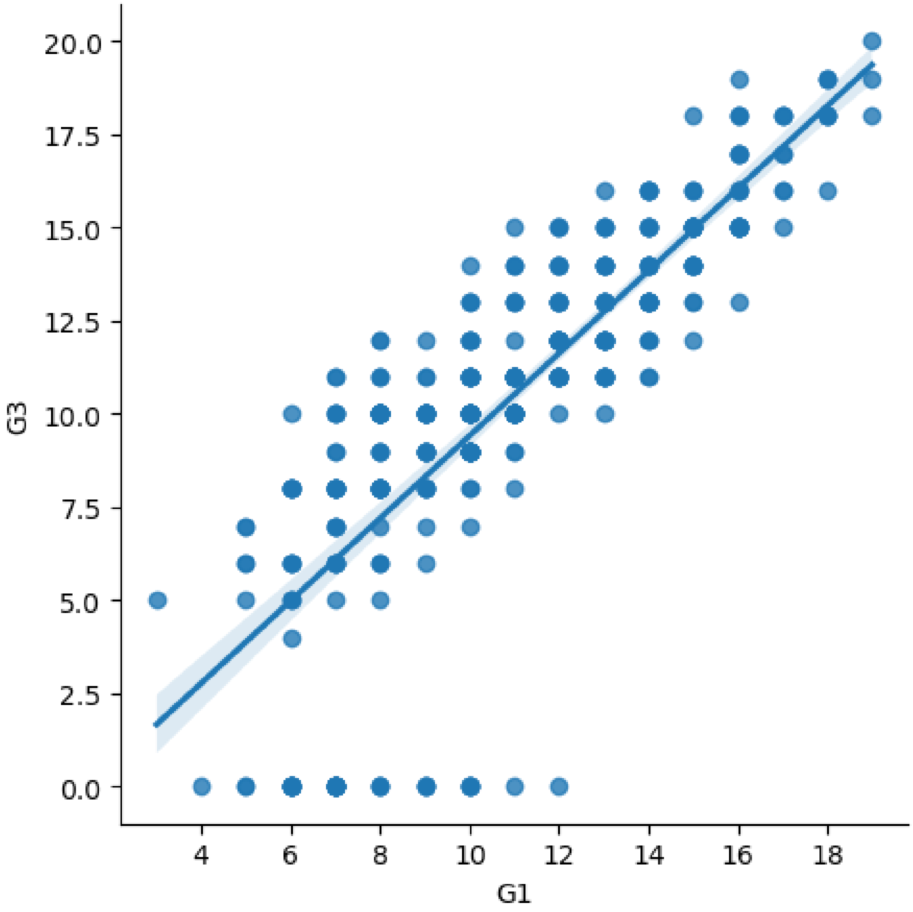
[14]:

```
sns.lmplot(x='Walc',y='G3',data=df)
plt.show()
```

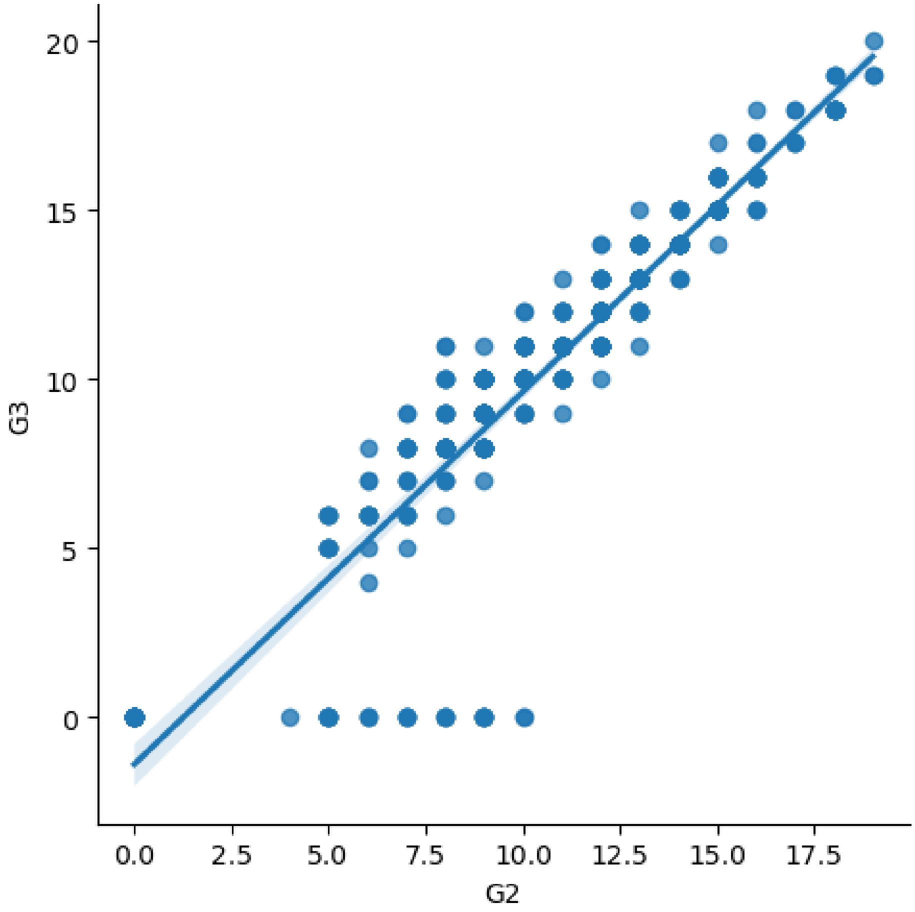


[11]:

```
sns.lmplot(x='G1',y='G3',data=df)
plt.show()
```



[12]:
```
sns.lmplot(x='G2',y='G3',data=df)
plt.show()
```



[13]:
```
from sklearn.model_selection import train_test_split
```

[14]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3, ␣
↪random_state=41)
```

[15]:
```
from sklearn.linear_model import LinearRegression
```

[16]:
```
model =LinearRegression()
```

[17]:
```
model.fit(X_train,y_train)
```

[17]:
```
LinearRegression()
```

[18]:
```
print(model.score(X_train,y_train))
```

```
0.8510204939162701
```

[19]:

```
prediction_test=model.predict(X_test)
print(y_test,prediction_test)
```

```
369    11
184    12
25      8
246    13
146     0
       ..
296     0
233    13
215    15
287    12
181    12
Name: G3, Length: 119, dtype: int64 [11.97373708 12.98392489  8.4341861
11.88388038  6.23409707  7.5430039
 10.87322383  9.95195525  9.36548347  7.44358714 15.33999223  8.85511952
 15.21730909  8.60019319  5.42548282 11.99653472 16.32417362 13.1066080З
 12.0827139   8.65307721 18.643737   13.1198456   9.68700013 15.30348827
 12.92101208  9.86256728 14.15055974  7.5329751  15.31351707  8.93126991
  9.75312172  7.44358714 11.34022125  7.74183742 14.18338617  7.90055577
 12.76550252  5.32285728 14.104027   11.77811234  8.68958116 15.01526678
  4.41482628 10.77380707 11.98329714  6.43293059 12.19536824  7.45682471
 10.77380707 10.79707344  8.65307721 15.42617141 14.0939982  -0.74108201
 10.99590696 10.952583   13.10660803  6.43293059 19.55497679 13.08334165
  6.33351383  7.73548615  6.55882251  6.52231856 15.5023218  11.98329714
 16.12534009 10.78704465 10.77380707 15.40290504 10.79707344 14.00461024
 10.78704465 12.88818565 12.79512016 16.26126081 11.97326834 14.20344376
 -1.0393323   9.54425941 14.19341496 15.40290504  7.68895341  8.44421489
 15.11468355  5.12402375  5.31282848  9.564317    8.50077643 15.5023218
 18.54432024  6.43293059 16.32417362 11.68825564  8.35482692 10.57497354
  5.62431635  4.19319512  7.3378191  12.79512016 15.30348827  8.83185314
  9.67697133 15.11468355 16.42679916  6.53234735 10.97264059 12.32808017
 -0.84049877  8.5901644   8.35482692  6.80051125 11.14138773  7.50335746
  8.76573155 13.29541276 15.29345948 11.98329714 12.98392489]
```

[20]:

```
print("Mean sq.error between y_test and prediction_test",np.
 ↪mean(prediction_test-y_test)**2)
```

```
Mean sq.error between y_test and prediction_test 0.2285913294902343
```

Save the Model

[21]:

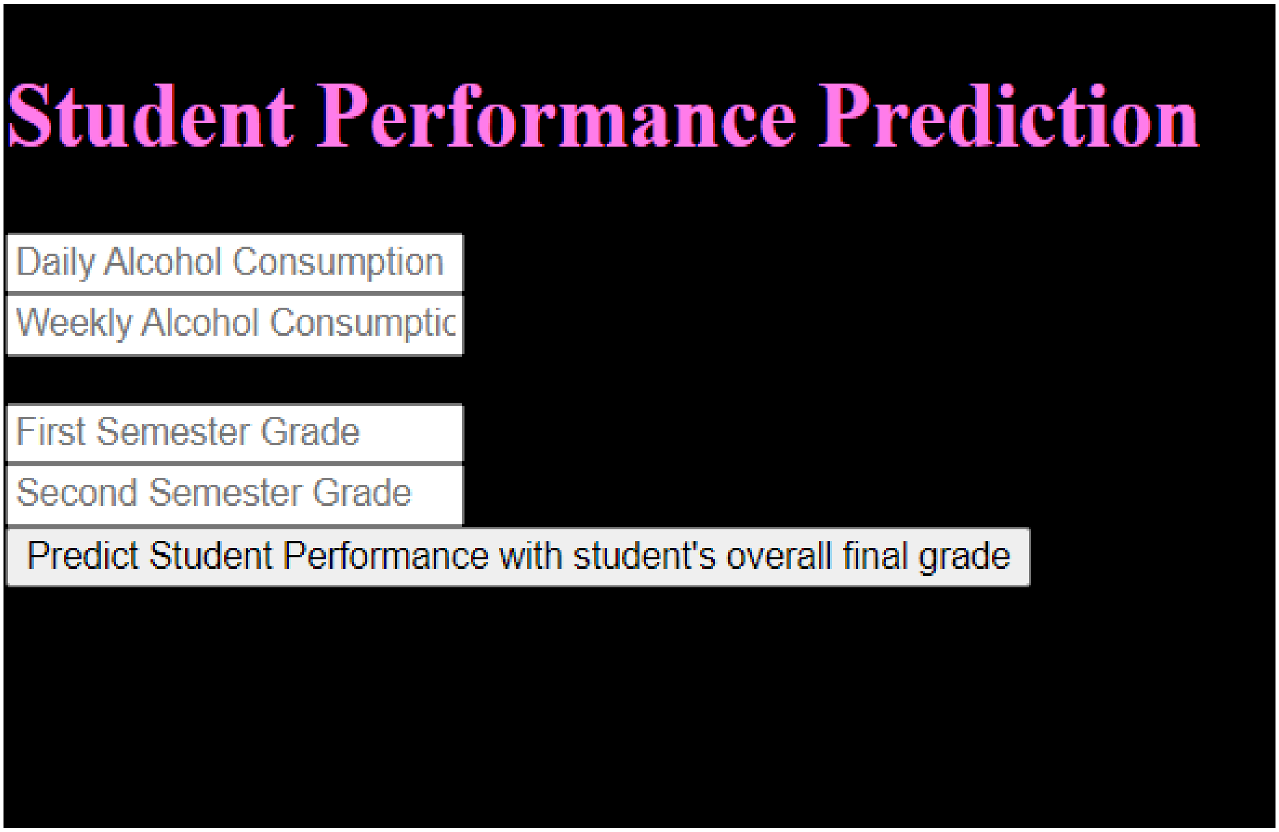
```
import pickle

pickle.dump(model, open('model.pkl', 'wb'))
```

Flask App

[ ]:
```
import numpy as np
from flask import Flask,request,render_template
import pickle

app = Flask(_name_)
model=pickle.load(open('model.pkl','rb'))
@app.route('/')
def home ():
    return render_template('index.html')
@app.route('/predict',methods=['POST'])
def predict() :
    int_features = [int(x) for x in request.form.values()]
    features=[np.array(int_features)]
    prediction=model.predict(features)

    output=prediction[0]

    return render_template('index.html',prediction_text="Predict Student␣
 ↪Performance with student's overall final grade   {}".format(output))
if___name__== '__main__':
    app.run()
```

```
 * Serving Flask app "_main_" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

[ ]:


Student Performance Prediction
Daily Alcohol Consumption
Weekly Alcohol Consumptic
First Semester Grade
Second Semester Grade
Predict Student Performance with student's overall final grade