Machine Learning

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
import statsmodels.api as sm
```

In [ ]:
```
file_path = 'slr.csv'
df = pd.read_csv(file_path)
```

In [ ]:
```
df.head()
```

Out[ ]:

	Exam	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54
3	1685	2.74
4	1693	2.83

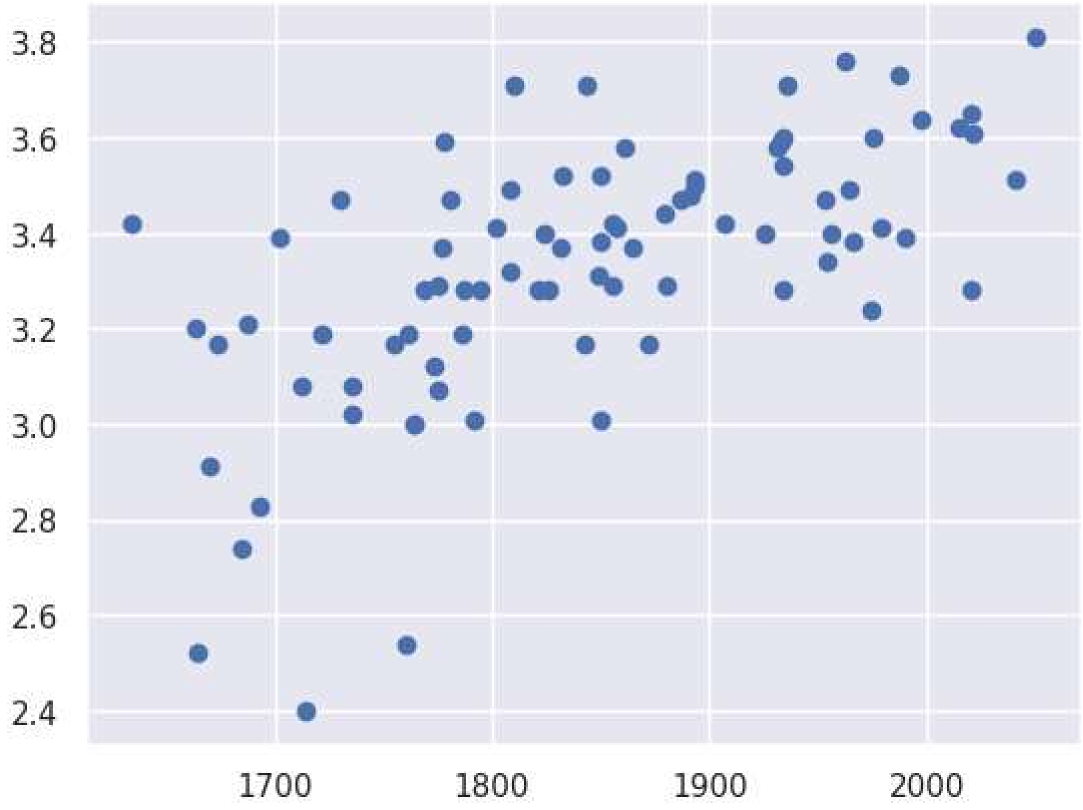
In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Exam    84 non-null     int64  
 1   GPA     84 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 1.4 KB
```

In [ ]:
```
# define independent and dependent variable       # step 2
```

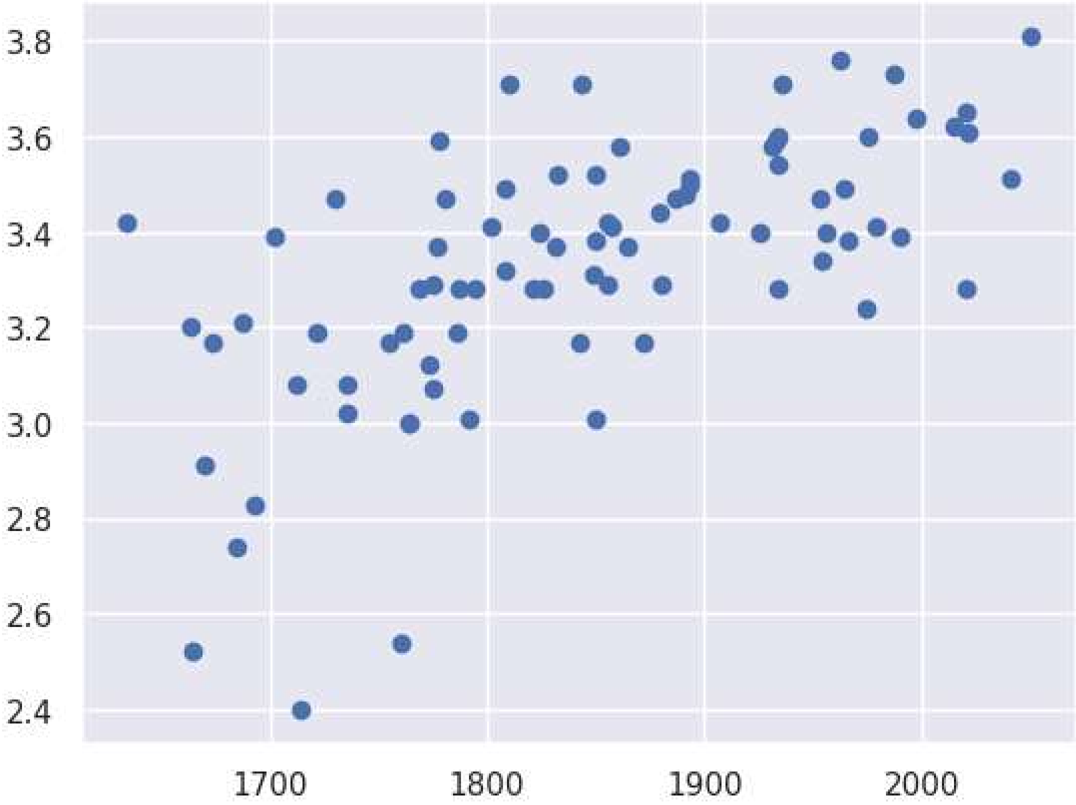
```
x1=df['Exam']  #independent
y=df['GPA']    #dependent
```

In [ ]:
```
sns.set()
plt.scatter(x1,y)
plt.show()
```



In [ ]:
```
sns.set()           # easy to understand background grid lines
plt.scatter(x1,y)   # show scatter
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x7df6c8f1c400>
```



In [ ]:
```
import statsmodels.api as sm
```

In [ ]:
```
x = sm.add_constant(x1) # create to constant value (1)
x
```

Out[ ]:

	const	Exam
0	1.0	1714
1	1.0	1664
2	1.0	1760
3	1.0	1685
4	1.0	1693
...	...	...
79	1.0	1936
80	1.0	1810
81	1.0	1987
82	1.0	1962
83	1.0	2050

84 rows × 2 columns

In [ ]:
```
model=sm.OLS(y,x)  # creating a model by using indepentant and dependent through OLS Methode
result=model.fit() # traing the model
```

In [ ]:
```
result.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	GPA	**R-squared:**	0.406
Model:	OLS	**Adj. R-squared:**	0.399
Method:	Least Squares	**F-statistic:**	56.05
Date:	Wed, 24 Apr 2024	**Prob (F-statistic):**	7.20e-11
Time:	04:22:50	**Log-Likelihood:**	12.672
No. Observations:	84	**AIC:**	-21.34
Df Residuals:	82	**BIC:**	-16.48
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.2750	0.409	0.673	0.503	-0.538	1.088
Exam	0.0017	0.000	7.487	0.000	0.001	0.002

Omnibus:	12.839	**Durbin-Watson:**	0.950
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	16.155
Skew:	-0.722	**Prob(JB):**	0.000310
Kurtosis:	4.590	**Cond. No.**	3.29e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.29e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Formula for linear line (or) Straight line Y = c + mx

Y --> Dependent

x --> Independent

c --> intercept (starting point)

m --> slope (line)

In [ ]:
```
yhat=0.275+0.0017*x1   # y=c+mx
yhat
```

Out[ ]:
```
0     3.1888
1     3.1038
2     3.2670
3     3.1395
4     3.1531
       ...
79    3.5662
80    3.3520
81    3.6529
82    3.6104
83    3.7600
Name: Exam, Length: 84, dtype: float64
```

In [ ]:
```
result.params
```

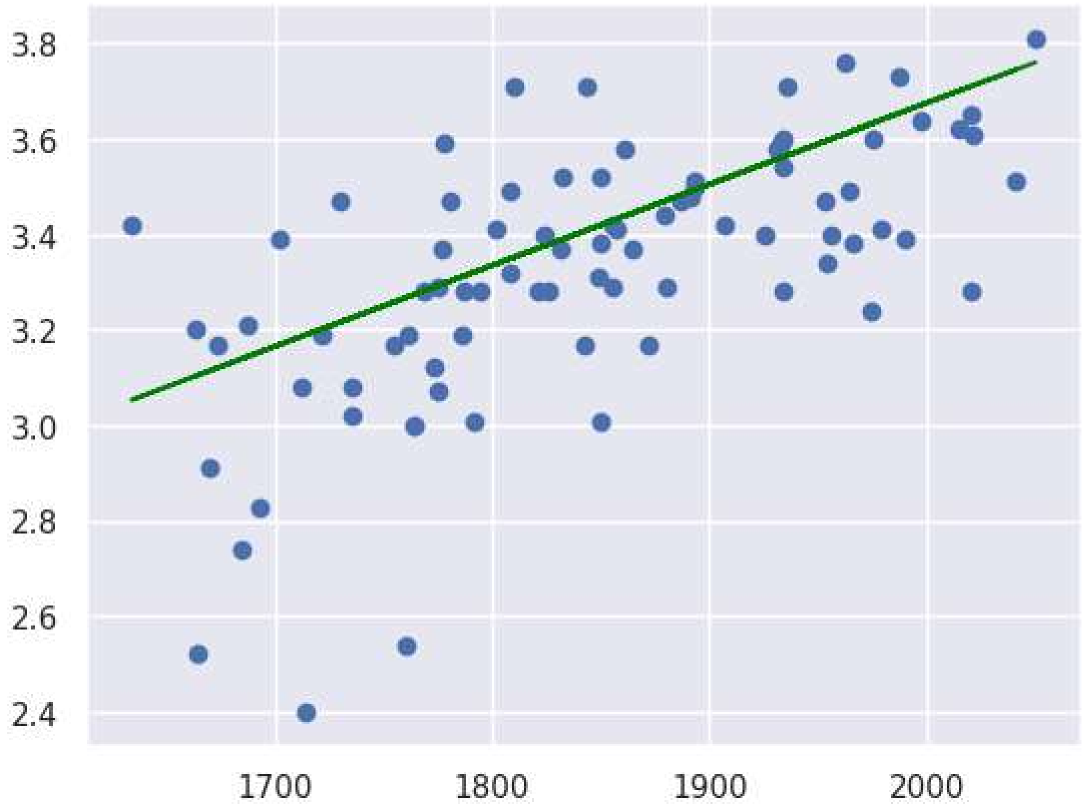
Out[ ]:
```
const    0.275040
Exam     0.001656
dtype: float64
```

In [ ]:
```
# predicting (if mark is 1987 means what is the output)
'''
y = mx + c
c = 0.275
m = 0017
x = 1987
'''


0.275+0.0017*1987  # c + m*x
```

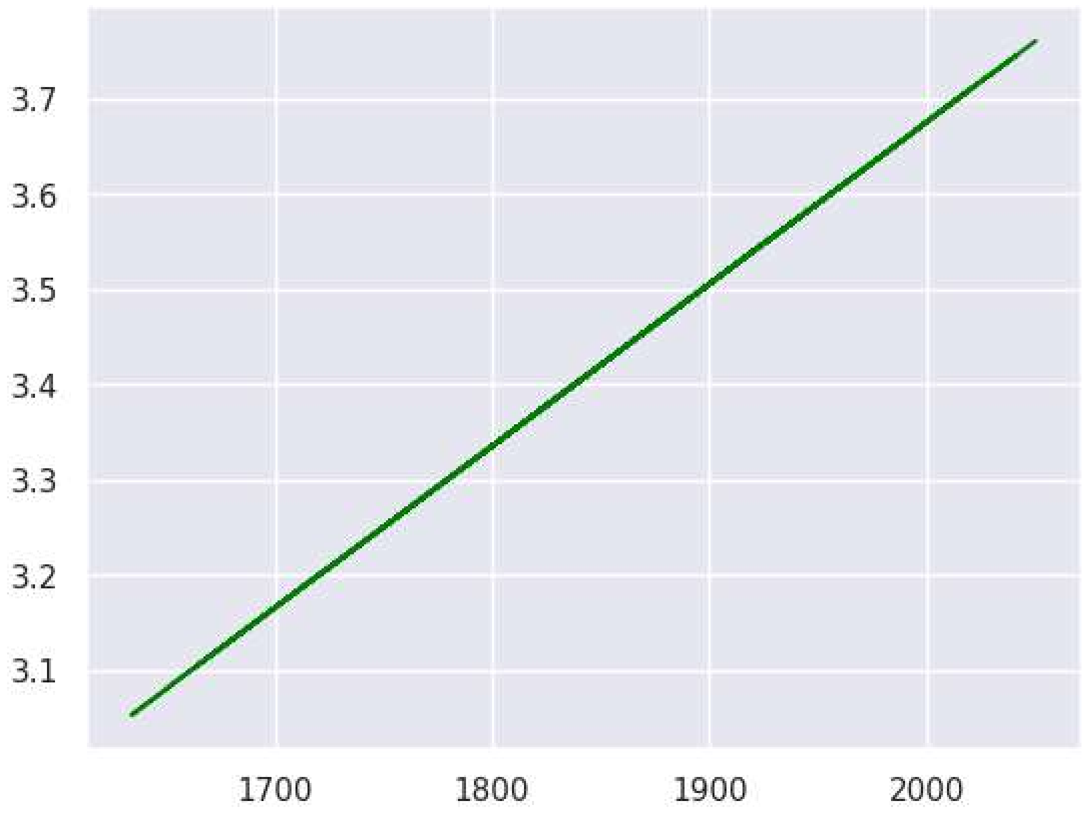
Out[ ]:
```
3.6529
```

In [ ]:
```
plt.scatter(x1,y)
plt.plot(x1,yhat,color="green") # Here yhat is dependent value
plt.show()
```




```
In [ ]: plt.plot(x1,yhat,color="green")
```

```
Out[ ]: [<matplotlib.lines.Line2D at 0x7df6c8dccfa0>]
```



In [ ]:

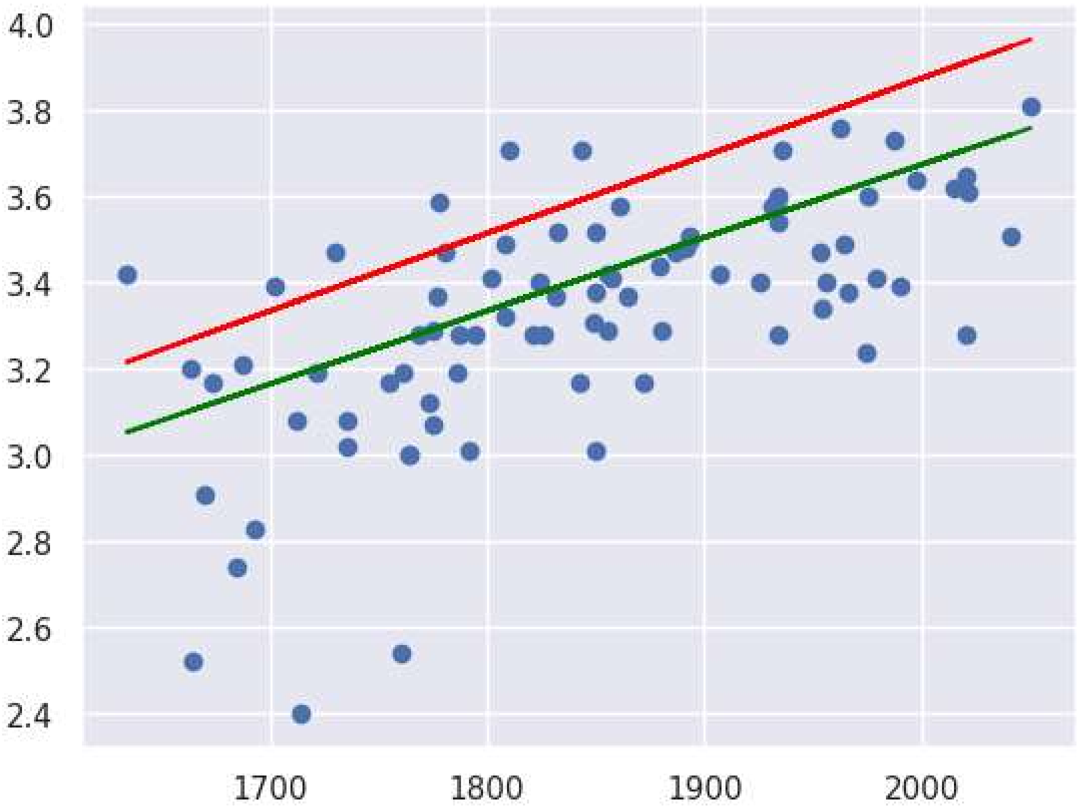
```
# for understanding 'm' value changed

yhat1=0.275+0.0018*x1  # y=mx+c
yhat1
```

Out[ ]:

```
0     3.3602
1     3.2702
2     3.4430
3     3.3080
4     3.3224
       ...
79    3.7598
80    3.5330
81    3.8516
82    3.8066
83    3.9650
Name: Exam, Length: 84, dtype: float64
```

In [ ]:
```
plt.scatter(x1,y)
plt.plot(x1,yhat,color="green")
plt.plot(x1,yhat1,color="red")
plt.show()
```



In [ ]:
```
result.params
```

Out[ ]:
```
const    0.275040
Exam     0.001656
dtype: float64
```

In [ ]:
```
result.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	GPA	**R-squared:**	0.406
Model:	OLS	**Adj. R-squared:**	0.399
Method:	Least Squares	**F-statistic:**	56.05
Date:	Wed, 24 Apr 2024	**Prob (F-statistic):**	7.20e-11
Time:	04:22:52	**Log-Likelihood:**	12.672
No. Observations:	84	**AIC:**	-21.34
Df Residuals:	82	**BIC:**	-16.48
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.2750	0.409	0.673	0.503	-0.538	1.088
Exam	0.0017	0.000	7.487	0.000	0.001	0.002

Omnibus:	12.839	**Durbin-Watson:**	0.950
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	16.155
Skew:	-0.722	**Prob(JB):**	0.000310
Kurtosis:	4.590	**Cond. No.**	3.29e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.29e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:

```
# ploting real value (other values are rounded off value)
result.params[0],result.params[1],result.params
```

Out[ ]:
```
(0.27504029966028876,
 0.0016556880500928112,
 const    0.275040
 Exam     0.001656
 dtype: float64)
```

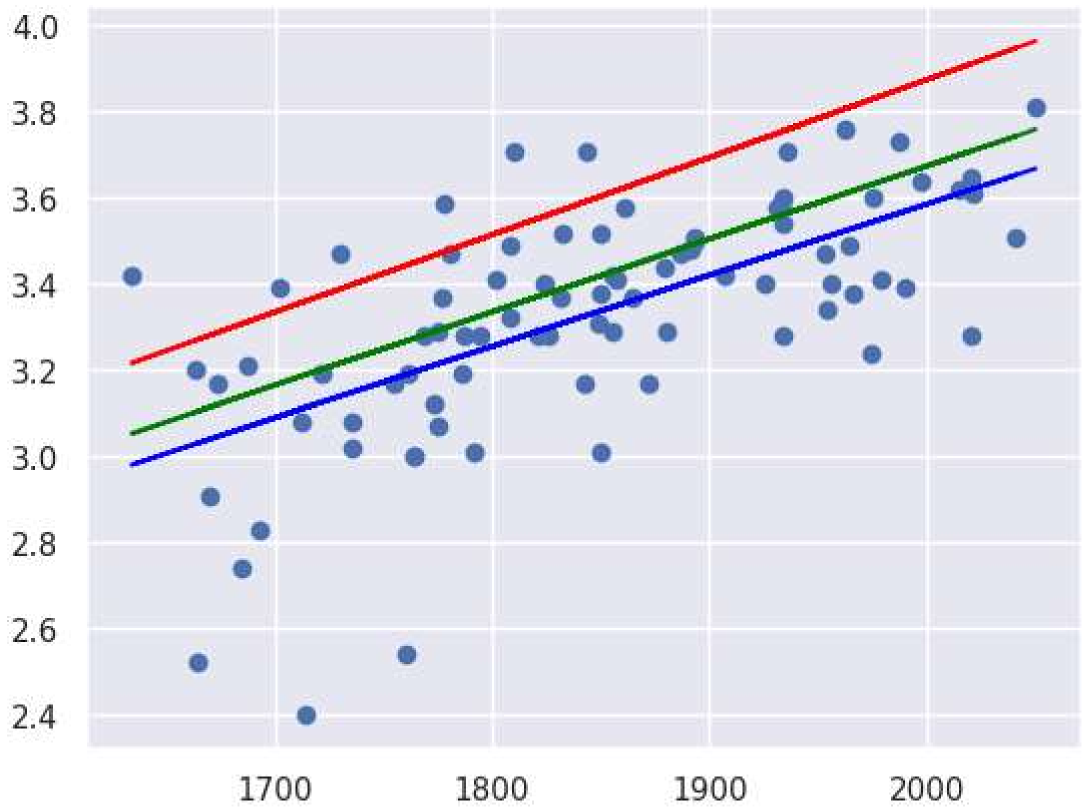
In [ ]:
```
result.params
```

Out[ ]:
```
const    0.275040
Exam     0.001656
dtype: float64
```

In [ ]:
```
yhat_org=result.params[0]+result.params[1]*x1
```

In [ ]:
```
# best fit line

plt.scatter(x1,y)
plt.plot(x1,yhat,color="green")
plt.plot(x1,yhat1,color="red")
plt.plot(x1,yhat_org,color="blue") # best fit line
plt.show()
```



In [ ]:

Regression Formulas

1. SST/TSS - Sum of squares of Total / Total Sum of Squares:

1. The difference between the actual point and the mean(average) point.
2. We are squaring the result to avoid the negative result.
3. To measure the total variability of the data set. (Mathematician perspective)

2. SSR/ESS - Sum of Squares of Regression / Explained Sum of Squares:

1. The difference between the predicted and the mean.
2. To measure the explained variability based on the line(best fit line) (Mathematician perspective)

3. SSE/RSS - Sum of Squares of Error / Residual Sum of Squares / Remaining Sum of Squares:

1. The difference between the actual and the predicted point.
2. To measure the unexplained variability by the equation. (Mathematician perspective)

Important:

SST = SSR + SSE

- If SSE is too low then it is advisable.

- If actual and predictor are same then SSE is zero.

In [ ]:

```

```

R-square = SSR / SST

R-square = 1, if SSR=SST Since SST = SSR + SSE, if SSE = 0 then SST = SSR Therfore, SST > SSR **R-square value will be always in the range of 0 and 1**. If the R-sqr value is towards 0 then it is more of error. If the R-sqr value if towards 1 then it is of very less error.