Vaibhav Kumar

Rollno 19

Linear Regression

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.model_selection import train_test_split
```

In [2]:
```
dfgpa=pd.read_csv('D:\\vk\\TRIM 3\\ML\\DATASET\\LR1.csv')
```

In [3]:
```
dfgpa.head(10)
```

Out[3]:

	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54
3	1685	2.74
4	1693	2.83
5	1670	2.91
6	1764	3.00
7	1764	3.00
8	1792	3.01
9	1850	3.01

In [4]:
```
dfgpa.describe()
```

Out[4]:

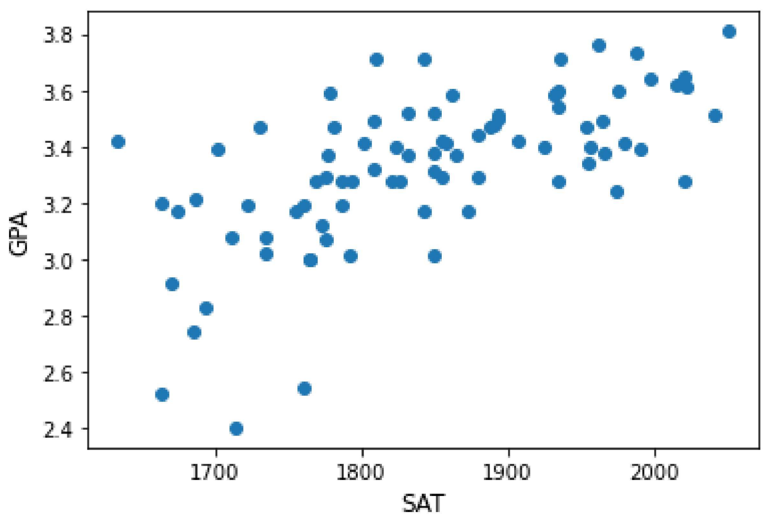
	SAT	GPA
count	84.000000	84.000000
mean	1845.273810	3.330238
std	104.530661	0.271617
min	1634.000000	2.400000
25%	1772.000000	3.190000
50%	1846.000000	3.380000
75%	1934.000000	3.502500
max	2050.000000	3.810000

In [5]:
```
dfgpa.shape
```

Out[5]: (84, 2)

In [6]:
```
x=dfgpa['SAT']
y=dfgpa['GPA']
```

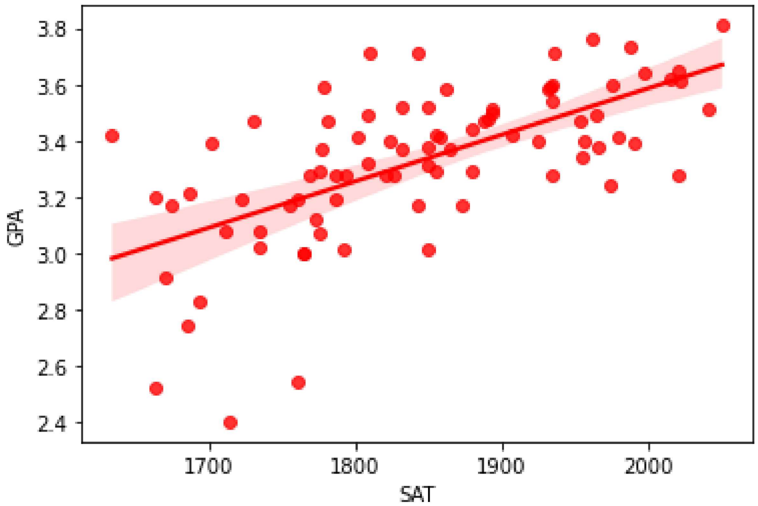
In [7]:
```
plt.scatter(x,y)
plt.xlabel('SAT',fontsize=12)
plt.ylabel('GPA',fontsize=12)
plt.show()
```



In [8]:
```
sns.regplot(x,y,color='red')
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]: <AxesSubplot:xlabel='SAT', ylabel='GPA'>



In [9]:
```
x.head()
```

Out[9]:
```
0    1714
1    1664
2    1760
3    1685
4    1693
Name: SAT, dtype: int64
```

In [10]:
```
X_=x.values.reshape(-1,1) # regression model only accepts the array value
```

In [11]:
```
x.shape
```

Out[11]:
```
(84,)
```

In [12]:
```
X_.shape
```

Out[12]:
```
(84, 1)
```

In [13]:
```
x
```

Out[13]:
```
0     1714
1     1664
2     1760
3     1685
4     1693
      ...
79    1936
80    1810
81    1987
82    1962
83    2050
Name: SAT, Length: 84, dtype: int64
```

In [14]:
```
X_
```

Out[14]:
```
array([[1714],
       [1664],
       [1760],
       [1685],
       [1693],
       [1670],
       [1764],
       [1764],
       [1792],
       [1850],
       [1735],
       [1775],
       [1735],
       [1712],
       [1773],
       [1872],
       [1755],
       [1674],
       [1842],
       [1786],
       [1761],
       [1722],
       [1663],
       [1687],
       [1974],
       [1826],
       [1787],
       [1821],
       [2020],
       [1794],
       [1769],
       [1934],
       [1775],
       [1855],
       [1880],
       [1849],
       [1808],
       [1954],
       [1777],
       [1831],
       [1865],
       [1850],
       [1966],
       [1702],
       [1990],
       [1925],
       [1824],
       [1956],
       [1857],
       [1979],
       [1802],
       [1855],
       [1907],
       [1634],
       [1879],
       [1887],
       [1730],
       [1953],
       [1781],
       [1891],
```

```
       [1964],
       [1808],
       [1893],
       [2041],
       [1893],
       [1832],
       [1850],
       [1934],
       [1861],
       [1931],
       [1933],
       [1778],
       [1975],
       [1934],
       [2021],
       [2015],
       [1997],
       [2020],
       [1843],
       [1936],
       [1810],
       [1987],
       [1962],
       [2050]], dtype=int64)
```

data cleaning done

Model

In [15]:
```
#this is important step for model devloping
#dividing the data into trainning and Testting
```

dividing

In [16]:
```
X_train,X_test,y_train,y_test=train_test_split(X_,y,test_size=0.1,random_state=69)
```

In [17]:
```
X_train.shape
```

Out[17]:
```
(75, 1)
```

In [18]:
```
X_test.shape
```

Out[18]:
```
(9, 1)
```

MODEL IS READY

NOW START THE *TRAINING*

In [19]:
```
#this is supervised traning technique
#that's why we are giving the input and label together
LR=LinearRegression()
LR.fit(X_train,y_train)
```

Out[19]:
```
LinearRegression()
```

In [20]:
```
# predict() is used for predicting
y_pred=LR.predict(X_test)
```

In [21]:
```
y_test
```

Out[21]:
```
67    3.54
15    3.17
36    3.32
25    3.28
10    3.02
50    3.41
51    3.42
31    3.28
40    3.37
Name: GPA, dtype: float64
```

In [22]:
```
y_pred
```

Out[22]:
```
array([3.48090482, 3.3774985 , 3.27075649, 3.30077768, 3.14900389,
       3.26074943, 3.34914515, 3.48090482, 3.36582359])
```

error

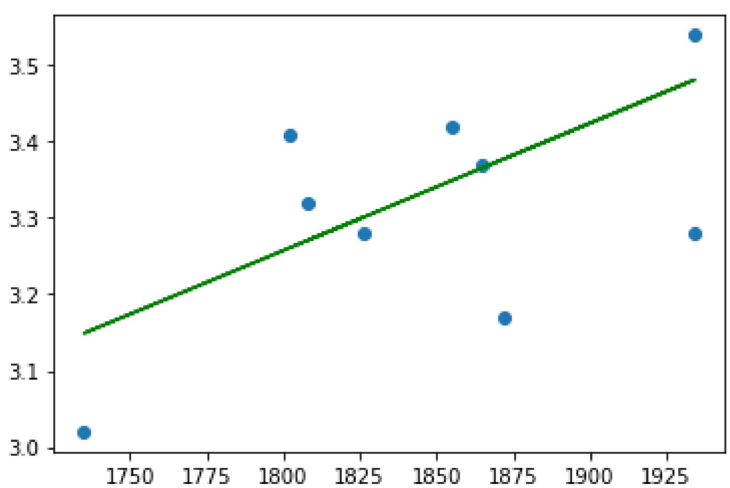
In [23]:
```
acc=mean_squared_error(y_test,y_pred)
acc
```

Out[23]:
```
0.014858092961980589
```

In [24]:
```
weights = LR.coef_
intercept = LR.intercept_
print(weights,intercept)
```

```
[0.00166784] 0.2552947901783029
```

In [25]:
```
plt.scatter(X_test, y_test)
plt.plot(X_test,y_pred, color='green')
plt.show()
```



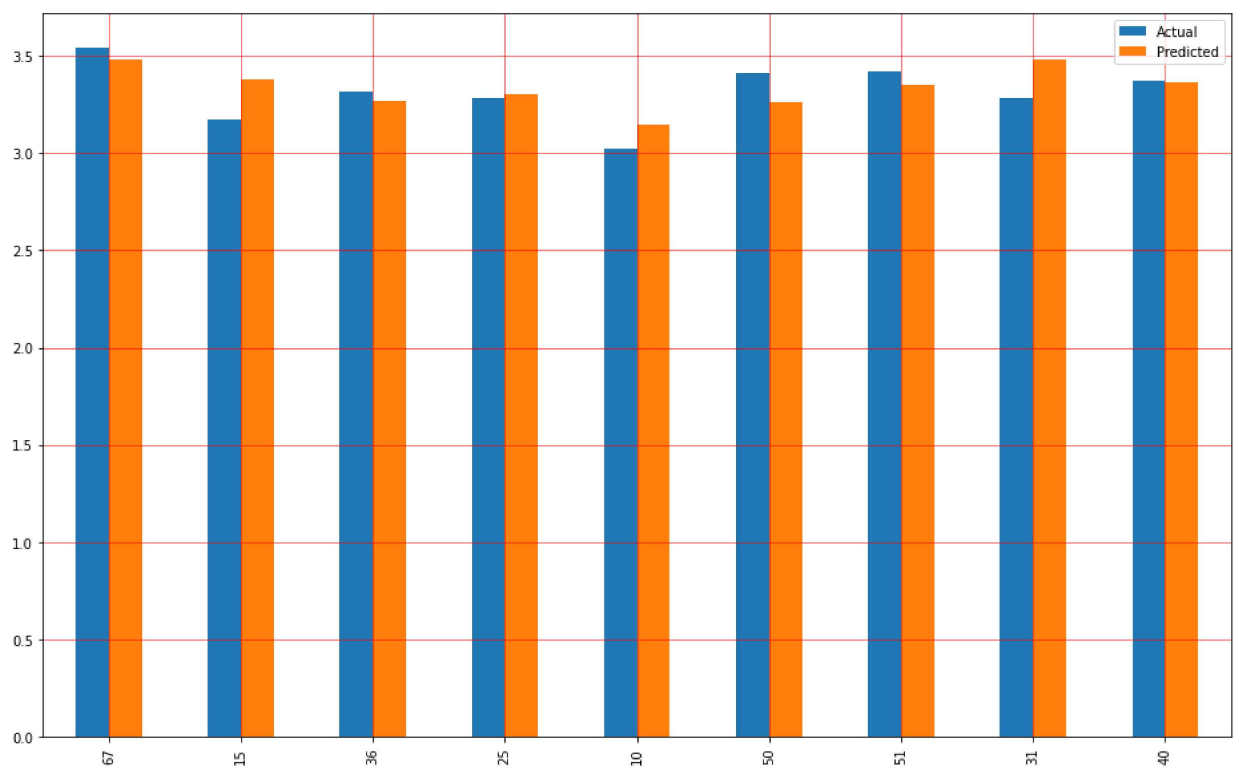
In [26]:
```
df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
df
```

Out[26]:

	Actual	Predicted
67	3.54	3.480905
15	3.17	3.377498
36	3.32	3.270756
25	3.28	3.300778
10	3.02	3.149004
50	3.41	3.260749
51	3.42	3.349145
31	3.28	3.480905
40	3.37	3.365824

In [27]:

```
df1=df.head(25)
df1.plot(kind='bar',figsize=(16,10))
plt.grid(which='major',linestyle='-',linewidth='0.5',color='red')
plt.grid(which='minor',linestyle=':',linewidth='0.5',color='green')
plt.show()
```



almost all the predicted and actual value are similar except some of the values

now we are changing the data or we can say giving new data to predict

this can be done only if the model is acceptable

In [28]:

```
new_data=pd.DataFrame([2115,900])
```

```
new_data
```

Out[28]:

	0
0	2115
1	900

In [29]:
```
LR.predict(new_data)
```

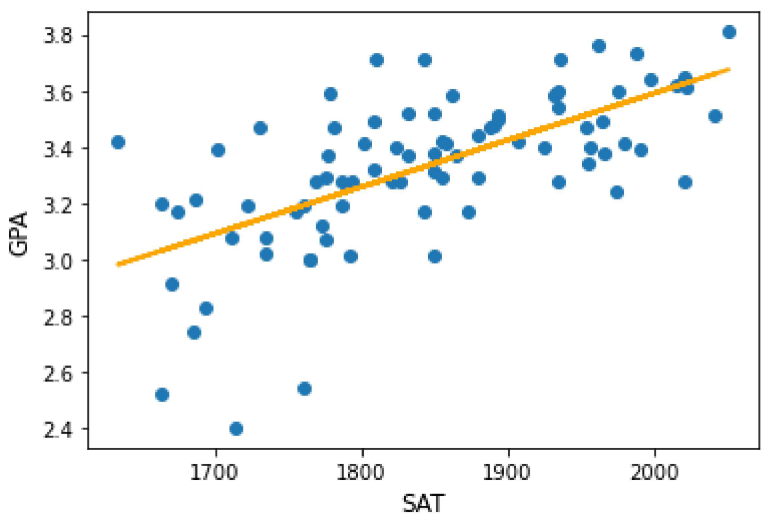
Out[29]:
```
array([3.78278456, 1.75635427])
```

preformance measure

In [30]:
```
print('Mean Absolute Error',mean_absolute_error(y_test,y_pred))
print('Mean Squared Error',mean_squared_error(y_test,y_pred))
print('Root Mean Sqaured Error',np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error 0.09897837838585355
Mean Squared Error 0.014858092961980589
Root Mean Sqaured Error 0.12189377737186008
```

In [31]:
```
plt.scatter(x,y)
yhat=LR.coef_*x+LR.intercept_ #yhat=0.275+0.0017x1 regression line
fig=plt.plot(x,yhat,lw=2,c='orange',label='Regression Line')
plt.xlabel('SAT',fontsize='12')
plt.ylabel('GPA',fontsize='12')
plt.show()
```



80 : 20 ---- Mean Absolute Error 0.14738307729471656 Mean Squared Error 0.039186315582133514 Root Mean Sqaured Error 0.1979553373418699

70:30---Mean Absolute Error 0.1469412968249975 Mean Squared Error 0.03697034973833757 Root Mean Sqaured Error 0.1922767529846954

90:10---Mean Absolute Error 0.09897837838585355 Mean Squared Error 0.014858092961980589 Root Mean Sqaured Error 0.12189377737186008

In [ ]: