# Name: Tarun Tanmay Roll no: N049 MBA Tech CE 3rd Year

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import statsmodels.api as sm

 /usr/local/lib/python3.6/dist-packages/statsmodels/tools/\_testing.py:19: FutureWarni import pandas.util.testing as tm

from google.colab import files uploaded = files.upload()

 No file chosen Upload widget is only available when the cell has been

Choose Files

executed in the current browser session. Please rerun this cell to enable.

Saving sat cgpa csv to sat cgpa csv

data=pd.read\_csv("sat\_cgpa.csv") data.head()

data.describe()



|  |  |  |
| --- | --- | --- |
|  | **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 |

y=data["GPA"] x1=data["SAT"]

plt.scatter(x1,y)

plt.xlabel("SAT",fontsize=15) plt.ylabel("GPA",fontsize=15)

plt.title("Data Points",fontsize=30,color="green")

# Conclusion:

For the given dataset of 84 points stats model library of python is used to determine t- statistics

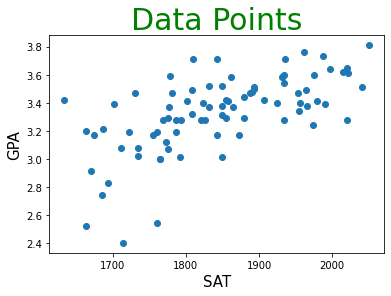
t-statistics is compare with p value for 99% confidence level

For the given dataset p value is < 0.001 therefore it can be concluded that slope is non zero Since t value is positve, slope of the given data points is also positive

ONS functionality of stats model also shows y intercept, slope and other statistical parameters for the given data set

Text(0.5, 1.0, 'Data Points')

x=sm.add\_constant(x1)



results=sm.OLS(y,x).fit() results.summary()

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:** Fri, 31 Jul 2020 **Prob (F-statistic):** 7.20e-11

**Time:** 11:25:38 **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

**SAT** 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 |

Warnings:

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.

plt.scatter(x1,y)

yhat=0.2750+(0.0017\*x1)

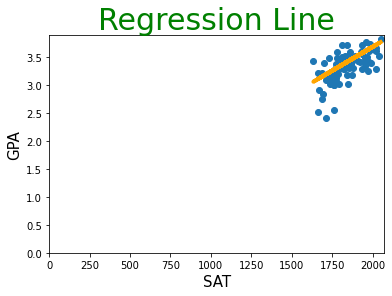
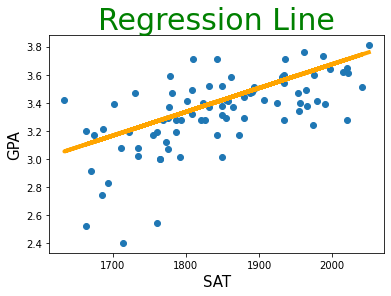
plt.plot(x1,yhat,lw=4,c='orange') plt.xlabel('SAT',fontsize=15)

plt.ylabel('GPA',fontsize=15)

plt.title('Regression Line',fontsize=30,color='green')

Text(0.5, 1.0, 'Regression Line')

plt.scatter(x1,y)



yhat=0.2750+(0.0017\*x1)

plt.plot(x1,yhat,lw=4,c='orange') plt.xlabel('SAT',fontsize=15)

plt.ylabel('GPA',fontsize=15) plt.xlim(0)

plt.ylim(0)

plt.title('Regression Line',fontsize=30,color='green') #Line for entire population

Text(0.5, 1.0, 'Regression Line')

# Confidence Interval

# c= (slope)+- (t\*SE/sqrt(n))

c\_u=0.0017+(7.487\*0.00) c\_l=0.0017+(7.487\*0.00)

Cs=([c\_l,c\_u])

print("CS=",Cs)

cm\_u=3.330238+(0.673\*0.271617/np.sqrt(84)) cm\_l=3.330238-(0.673\*0.271617/np.sqrt(84))

Cm=([cm\_l,cm\_u])

print("Cm=",Cm)

 CS= [0.0017, 0.0017]

Cm= [3.3102930767550647, 3.3501829232449354]