# Problem Statement: -

A certain university wants to understand the relationship between students' SAT scores and their GPA. Build a Simple Linear Regression model with GPA as the target variable and record the RMSE and correlation coefficient values for different models.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("D:\\360Digi\Simple Resgression Ass\\SAT_GPA.csv")
df.isnull().sum()
```

Out[1]:

```
SAT_Scores    0
GPA           0
dtype: int64
```

# EDA

In [2]:

```
Eda = {"columns": df.columns,
       "mean": df.mean(),
       "median":df.median(),
       "strdrand deviation":df.std(),
       "variance": df.var(),
       "kurtosis": df.kurt()
       }
Eda
```

Out[2]:

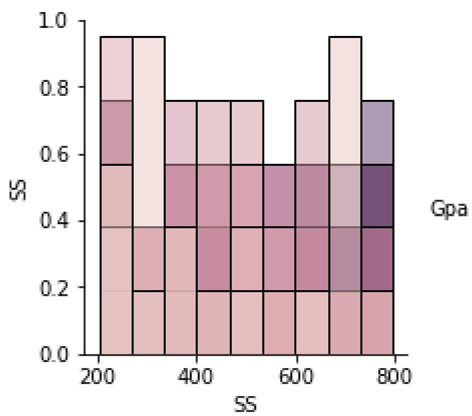
```
{'columns': Index(['SAT_Scores', 'GPA'], dtype='object'),
 'mean': SAT_Scores    491.8100
 GPA             2.8495
 dtype: float64,
 'median': SAT_Scores    480.5
 GPA             2.8
 dtype: float64,
 'strdrand deviation': SAT_Scores    174.893834
 GPA             0.541076
 dtype: float64,
 'variance': SAT_Scores    30587.853166
 GPA               0.292764
 dtype: float64,
 'kurtosis': SAT_Scores   -1.224188
 GPA          -1.040576
 dtype: float64}
```

In [3]:
```
df.columns.values[0] = "SS"
df.columns.values[1] = "Gpa"
df.columns
```

Out[3]:
```
Index(['SS', 'Gpa'], dtype='object')
```

In [4]:
```
plt.figure(figsize=(30, 30))
sns.pairplot(df, hue='Gpa', height=3, diag_kind='hist')
```

Out[4]:
```
<seaborn.axisgrid.PairGrid at 0x1fe2449c400>
<Figure size 2160x2160 with 0 Axes>
```

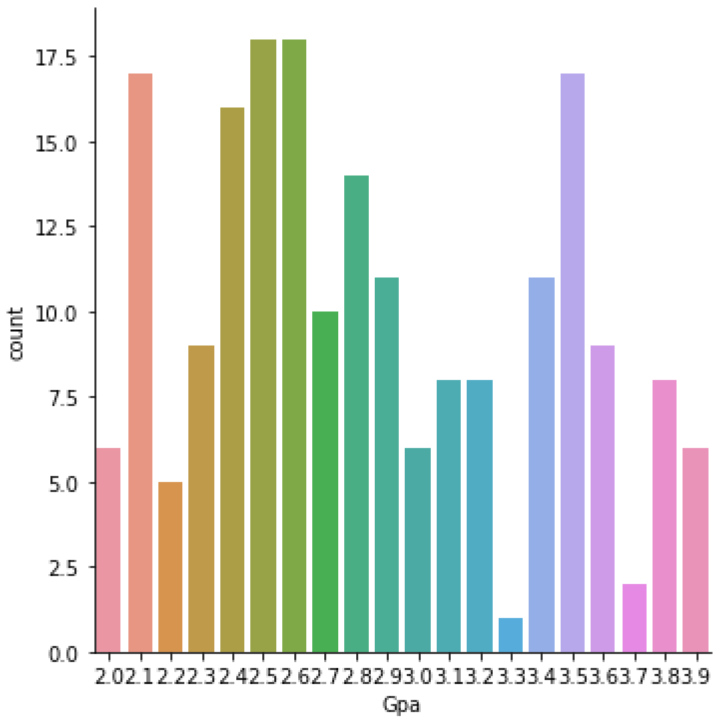


In [5]:

```
#yes or no count
sns.catplot('Gpa', data=df, kind='count')
```

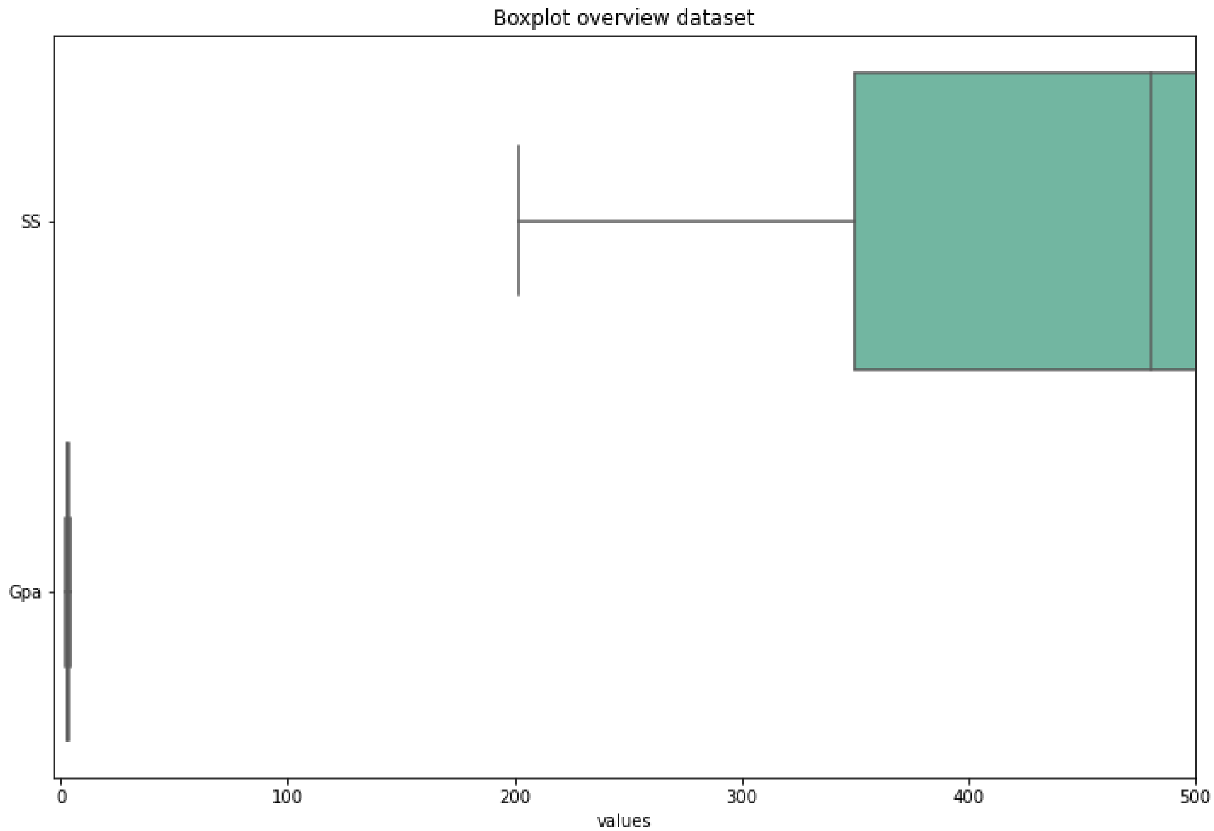
```
D:\anconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(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

Out[5]: <seaborn.axisgrid.FacetGrid at 0x1fe24485d00>

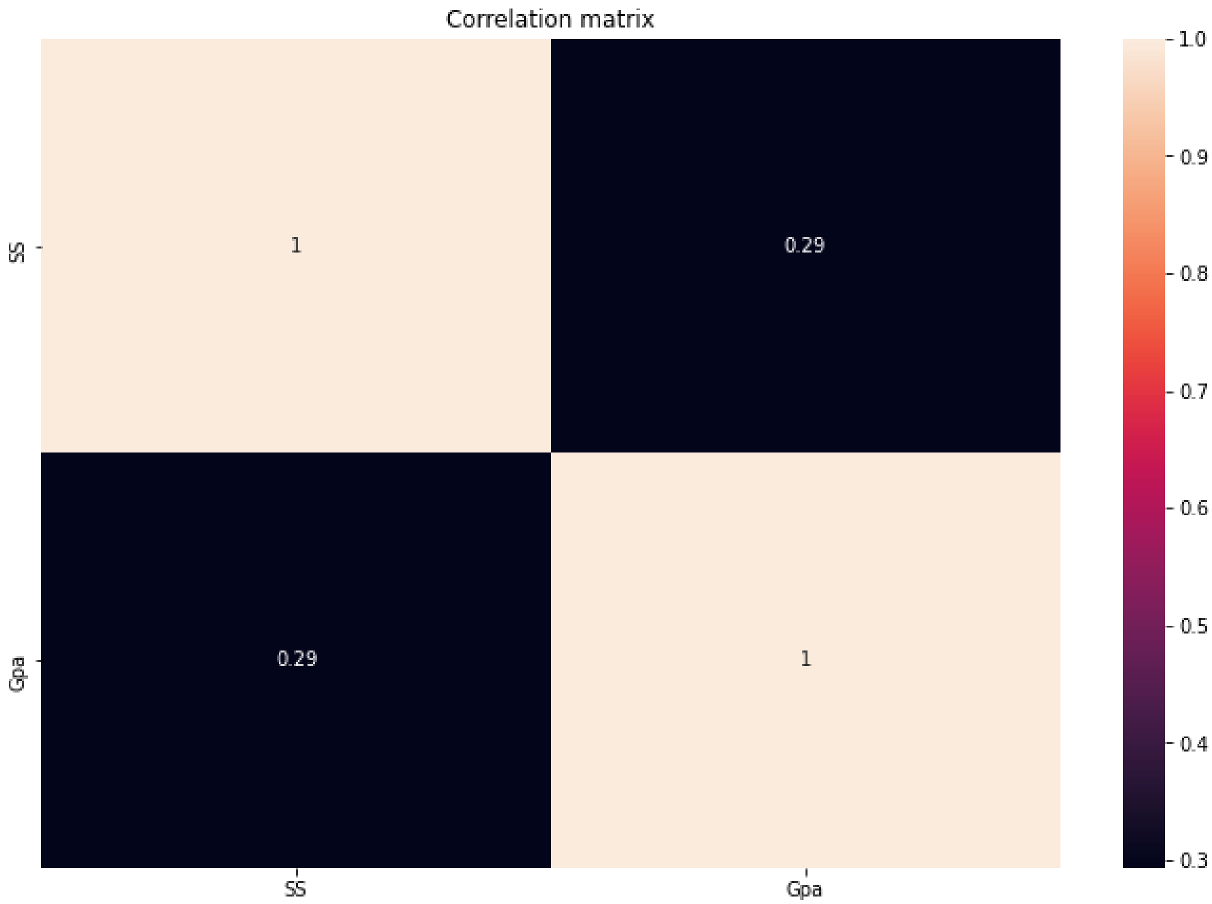


In [6]:

```
import matplotlib.pyplot as plt

plt.figure(figsize = (12, 8))
ax = sns.boxplot(data = df, orient = 'h', palette = 'Set2')
plt.title('Boxplot overview dataset')
plt.xlabel('values')
plt.xlim(-3, 500)
plt.show()
```



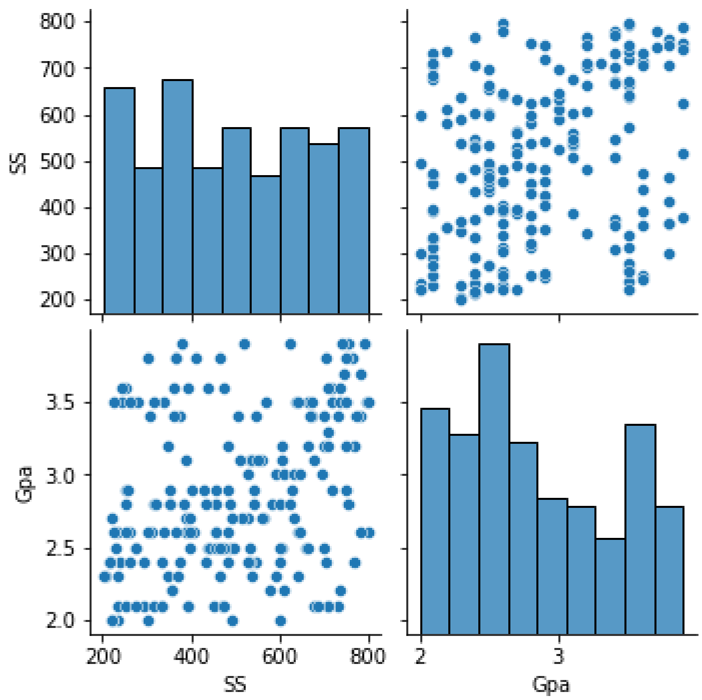
In [7]:
```
plt.figure(figsize = (12, 8))
sns.heatmap(df.corr(), annot = True)
plt.title('Correlation matrix')
plt.show()
```



In [8]:

```
sns.pairplot(df)
```

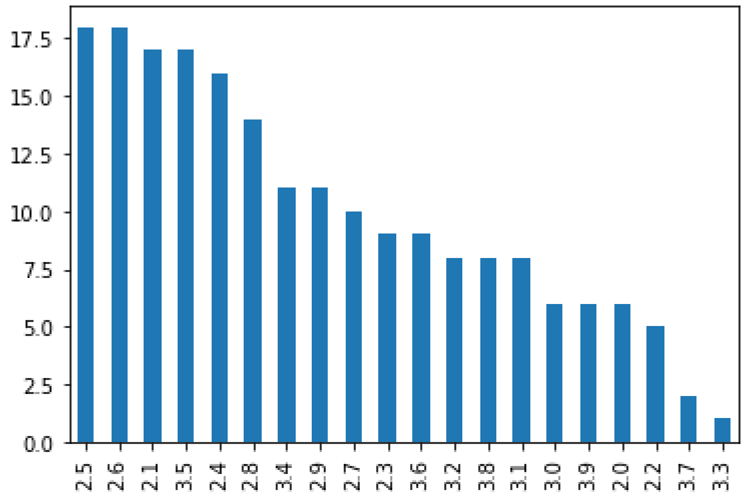
Out[8]: <seaborn.axisgrid.PairGrid at 0x1fe28c4b8b0>



In [9]:

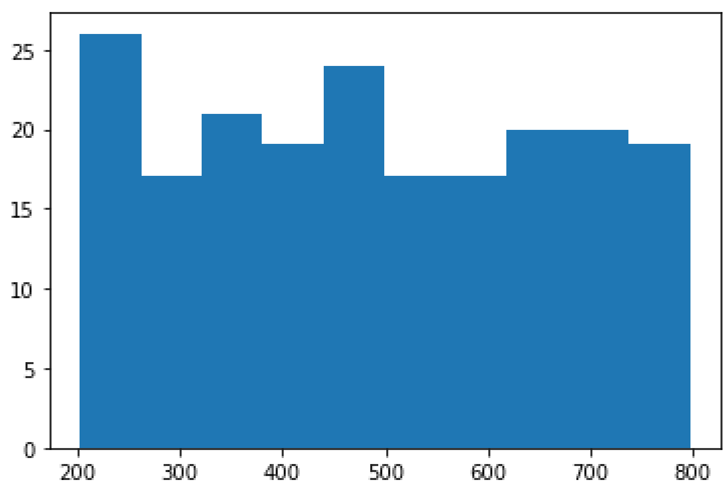
```
#Bar plot
df['Gpa'].value_counts().plot.bar()
```

Out[9]: <AxesSubplot:>



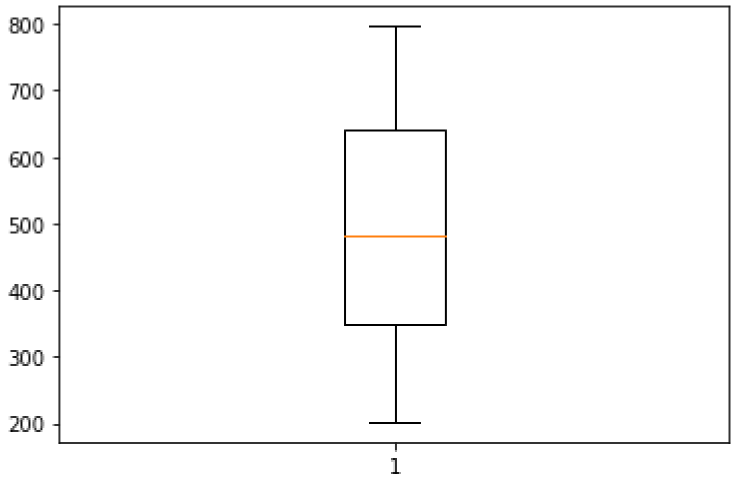
In [10]:
```
'''
# Normalization function using z std. all are continuous data.
def std_func(i):
    x = (i-i.mean())/(i.std())
    return (x)

# Normalized data frame (considering the numerical part of data)
cal = std_func(df)
cal.describe()
'''
cal = df

#Data Modeling

#Graphical Representation
import matplotlib.pyplot as plt # mostly used for visualization purposes

#plt.bar(height = cal.ST, x = np.arange(1, 110, 1))
plt.hist(cal.SS) #histogram
```

Out[10]:
```
(array([26., 17., 21., 19., 24., 17., 17., 20., 20., 19.]),
 array([202. , 261.5, 321. , 380.5, 440. , 499.5, 559. , 618.5, 678. ,
        737.5, 797. ]),
 <BarContainer object of 10 artists>)
```



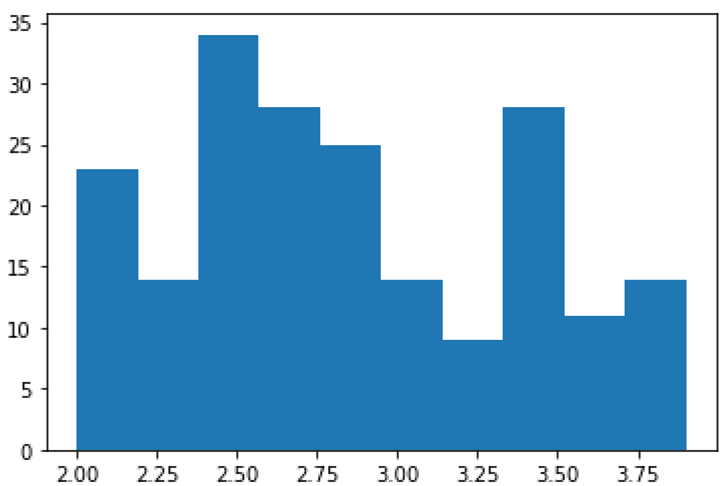
In [11]:
```
plt.boxplot(cal.SS) #boxplot
```

Out[11]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1fe2a306610>,
  <matplotlib.lines.Line2D at 0x1fe2a306970>],
 'caps': [<matplotlib.lines.Line2D at 0x1fe2a306cd0>,
  <matplotlib.lines.Line2D at 0x1fe2a311070>],
 'boxes': [<matplotlib.lines.Line2D at 0x1fe2a3062b0>],
 'medians': [<matplotlib.lines.Line2D at 0x1fe2a3113d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1fe2a311730>],
 'means': []}
```



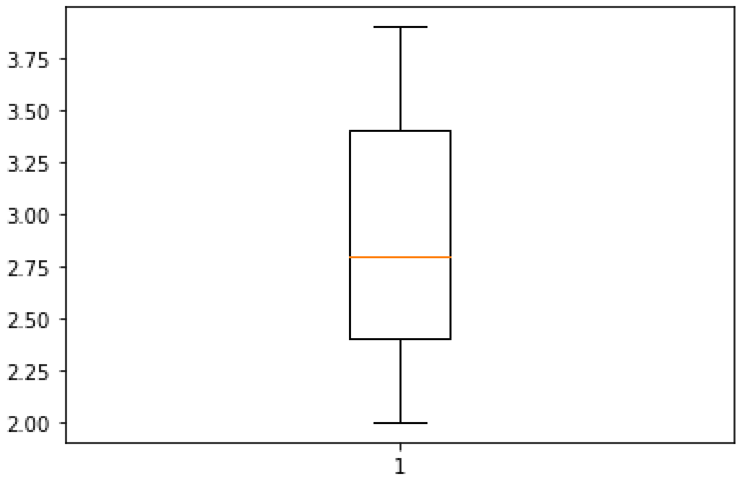

In [12]:
```
plt.hist(cal.Gpa) #histogram
```

Out[12]:
```
(array([23., 14., 34., 28., 25., 14.,  9., 28., 11., 14.]),
 array([2.  , 2.19, 2.38, 2.57, 2.76, 2.95, 3.14, 3.33, 3.52, 3.71, 3.9 ]),
 <BarContainer object of 10 artists>)
```



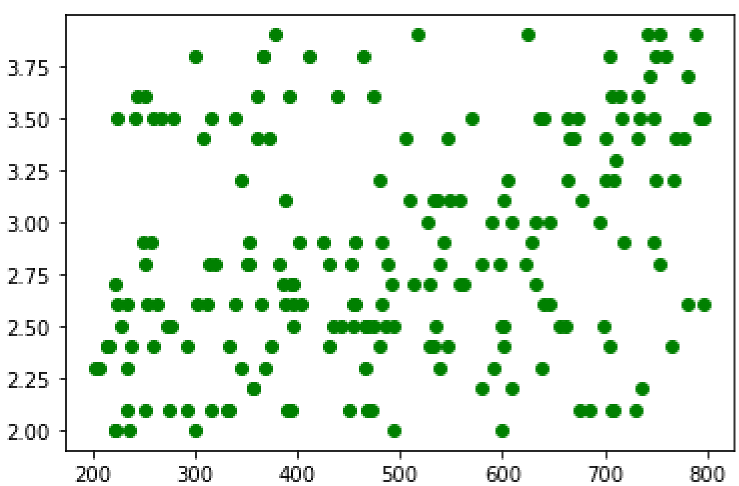
In [13]:
```
plt.boxplot(cal.Gpa) #boxplot
```

Out[13]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1fe2a3dc7f0>,
  <matplotlib.lines.Line2D at 0x1fe2a3dcb50>],
 'caps': [<matplotlib.lines.Line2D at 0x1fe2a3dceb0>,
  <matplotlib.lines.Line2D at 0x1fe2a3e8250>],
 'boxes': [<matplotlib.lines.Line2D at 0x1fe2a3dc490>],
 'medians': [<matplotlib.lines.Line2D at 0x1fe2a3e85b0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1fe2a3e8910>],
 'means': []}
```




In [14]:
```
# Scatter plot
plt.scatter(x = cal.SS, y = cal.Gpa, color = 'green')
```

Out[14]: <matplotlib.collections.PathCollection at 0x1fe2a43e400>



In [15]:
```
# correlation
np.corrcoef(cal.SS, cal.Gpa)
```

Out[15]:
```
array([[1.        , 0.29353828],
       [0.29353828, 1.        ]])
```

In [16]:
```
# Covariance
# NumPy does not have a function to calculate the covariance between two variable
# Function for calculating a covariance matrix called cov()
# By default, the cov() function will calculate the unbiased or sample covariance

cov_output = np.cov(cal.SS, cal.Gpa)[0, 1]
cov_output
```

Out[16]:
```
27.777793969849263
```

# DATA MODELING

In [17]:
```
# Import library
import statsmodels.formula.api as smf

# Simple Linear Regression
model = smf.ols('Gpa ~ SS', data = cal).fit()
model.summary()
```

Out[17]:

OLS Regression Results

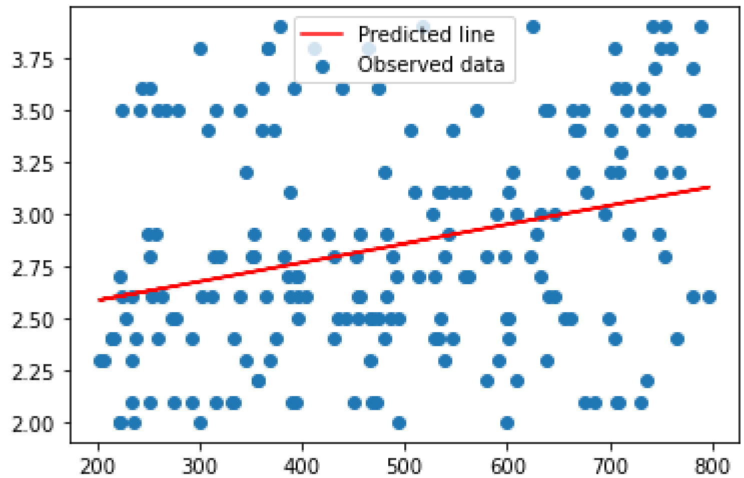
| | | | |
|---|---|---|---|
| **Dep. Variable:** | Gpa | **R-squared:** | 0.086 |
| **Model:** | OLS | **Adj. R-squared:** | 0.082 |
| **Method:** | Least Squares | **F-statistic:** | 18.67 |
| **Date:** | Sat, 19 Jun 2021 | **Prob (F-statistic):** | 2.46e-05 |
| **Time:** | 09:42:42 | **Log-Likelihood:** | -151.44 |
| **No. Observations:** | 200 | **AIC:** | 306.9 |
| **Df Residuals:** | 198 | **BIC:** | 313.5 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 2.4029 | 0.110 | 21.908 | 0.000 | 2.187 | 2.619 |
| **SS** | 0.0009 | 0.000 | 4.321 | 0.000 | 0.000 | 0.001 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 12.519 | **Durbin-Watson:** | 1.323 |
| **Prob(Omnibus):** | 0.002 | **Jarque-Bera (JB):** | 7.558 |
| **Skew:** | 0.317 | **Prob(JB):** | 0.0228 |
| **Kurtosis:** | 2.290 | **Cond. No.** | 1.56e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.56e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [18]:
```
pred1 = model.predict(pd.DataFrame(cal.SS))

# Regression Line
plt.scatter(cal.SS, cal.Gpa)
plt.plot(cal.SS, pred1, "r")
plt.legend(['Predicted line', 'Observed data'])
plt.show()

# Error calculation
res1 = cal.Gpa - pred1
res_sqr1 = res1 * res1
mse1 = np.mean(res_sqr1)
rmse1 = np.sqrt(mse1)
rmse1
```




Out[18]: 0.5159457227723684

In [19]:

```python
######### Model building on Transformed Data
# Log Transformation
# x = log(waist); y = at

plt.scatter(x = np.log(cal.SS), y = cal.Gpa, color = 'brown')
np.corrcoef(np.log(cal.SS), cal.Gpa) #correlation

model2 = smf.ols('Gpa ~ np.log(SS)', data = cal).fit()
model2.summary()
```

Out[19]:

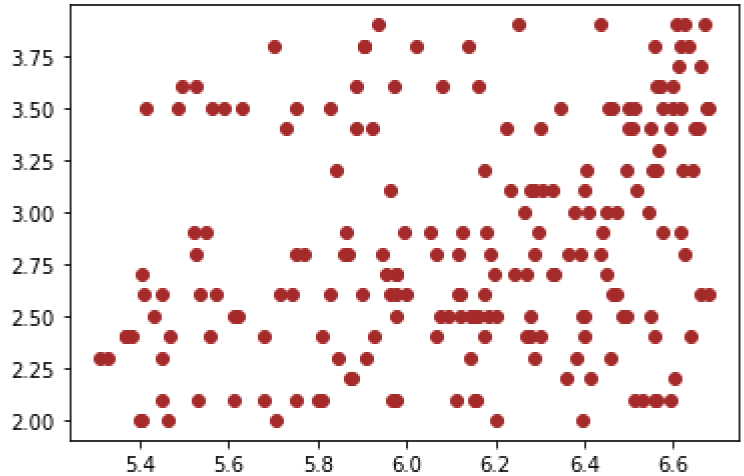
OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Gpa | **R-squared:** | 0.077 |
| **Model:** | OLS | **Adj. R-squared:** | 0.072 |
| **Method:** | Least Squares | **F-statistic:** | 16.55 |
| **Date:** | Sat, 19 Jun 2021 | **Prob (F-statistic):** | 6.85e-05 |
| **Time:** | 09:42:42 | **Log-Likelihood:** | -152.42 |
| **No. Observations:** | 200 | **AIC:** | 308.8 |
| **Df Residuals:** | 198 | **BIC:** | 315.4 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 0.4796 | 0.584 | 0.822 | 0.412 | -0.672 | 1.631 |
| **np.log(SS)** | 0.3868 | 0.095 | 4.068 | 0.000 | 0.199 | 0.574 |

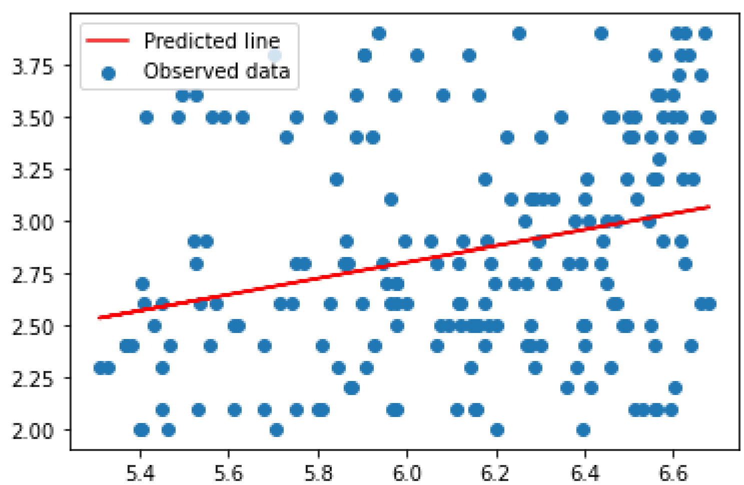
| | | | |
|---|---|---|---|
| **Omnibus:** | 15.866 | **Durbin-Watson:** | 1.333 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 8.435 |
| **Skew:** | 0.320 | **Prob(JB):** | 0.0147 |
| **Kurtosis:** | 2.224 | **Cond. No.** | 99.8 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



In [20]:

```
pred2 = model2.predict(pd.DataFrame(cal.SS))

# Regression Line
plt.scatter(np.log(cal.SS), cal.Gpa)
plt.plot(np.log(cal.SS), pred2, "r")
plt.legend(['Predicted line', 'Observed data'])
plt.show()

# Error calculation
res2 = cal.Gpa - pred2
res_sqr2 = res2 * res2
mse2 = np.mean(res_sqr2)
rmse2 = np.sqrt(mse2)
rmse2
```




Out[20]: 0.5184904101080668

In [21]:

```python
#### Exponential transformation
# x = waist; y = log(at)
#cal.columns

plt.scatter(x = cal.SS, y = np.log(cal.Gpa), color = 'orange')
np.corrcoef(cal.SS, np.log(cal.Gpa)) #correlation

model3 = smf.ols('np.log(Gpa) ~ SS', data = cal).fit()
model3.summary()
```

Out[21]:

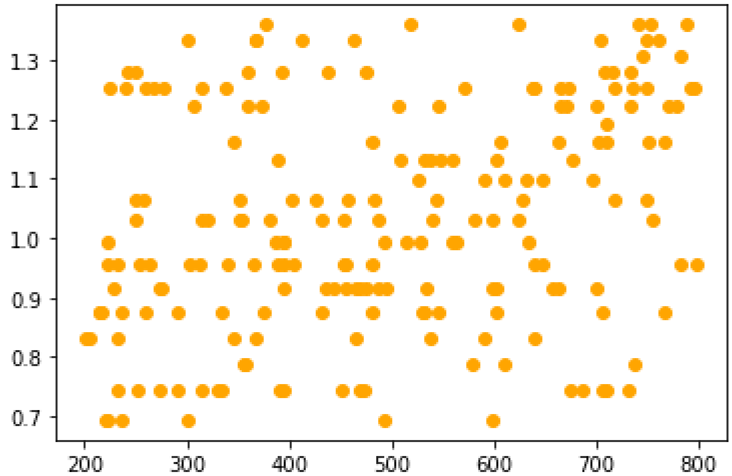
OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | np.log(Gpa) | **R-squared:** | 0.086 |
| **Model:** | OLS | **Adj. R-squared:** | 0.082 |
| **Method:** | Least Squares | **F-statistic:** | 18.75 |
| **Date:** | Sat, 19 Jun 2021 | **Prob (F-statistic):** | 2.37e-05 |
| **Time:** | 09:42:43 | **Log-Likelihood:** | 58.615 |
| **No. Observations:** | 200 | **AIC:** | -113.2 |
| **Df Residuals:** | 198 | **BIC:** | -106.6 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 0.8727 | 0.038 | 22.745 | 0.000 | 0.797 | 0.948 |
| **SS** | 0.0003 | 7.35e-05 | 4.330 | 0.000 | 0.000 | 0.000 |

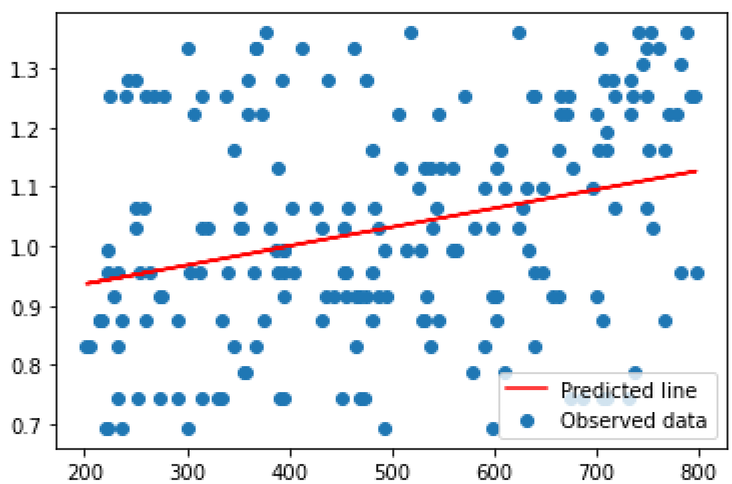
| | | | |
|---|---|---|---|
| **Omnibus:** | 11.046 | **Durbin-Watson:** | 1.375 |
| **Prob(Omnibus):** | 0.004 | **Jarque-Bera (JB):** | 4.816 |
| **Skew:** | 0.066 | **Prob(JB):** | 0.0900 |
| **Kurtosis:** | 2.251 | **Cond. No.** | 1.56e+03 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.56e+03. This might indicate that there are
strong multicollinearity or other numerical problems.




In [22]:

```
pred3 = model3.predict(pd.DataFrame(cal.SS))
pred3_at = np.exp(pred3)
pred3_at

# Regression Line
plt.scatter(cal.SS, np.log(cal.Gpa))
plt.plot(cal.SS, pred3, "r")
plt.legend(['Predicted line', 'Observed data'])
plt.show()

# Error calculation
res3 = cal.Gpa - pred3_at
res_sqr3 = res3 * res3
mse3 = np.mean(res_sqr3)
rmse3 = np.sqrt(mse3)
rmse3
```




Out[22]: 0.5175875893834132

In [23]:

```
#### Polynomial transformation
# x = waist; x^2 = waist*waist; y = log(at)

model4 = smf.ols('np.log(Gpa) ~ SS + I(SS*SS)', data = cal).fit()
model4.summary()
```

Out[23]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | np.log(Gpa) | **R-squared:** | 0.094 |
| **Model:** | OLS | **Adj. R-squared:** | 0.085 |
| **Method:** | Least Squares | **F-statistic:** | 10.23 |
| **Date:** | Sat, 19 Jun 2021 | **Prob (F-statistic):** | 5.95e-05 |
| **Time:** | 09:42:43 | **Log-Likelihood:** | 59.448 |
| **No. Observations:** | 200 | **AIC:** | -112.9 |
| **Df Residuals:** | 197 | **BIC:** | -103.0 |
| **Df Model:** | 2 | | |
| **Covariance Type:** | nonrobust | | |

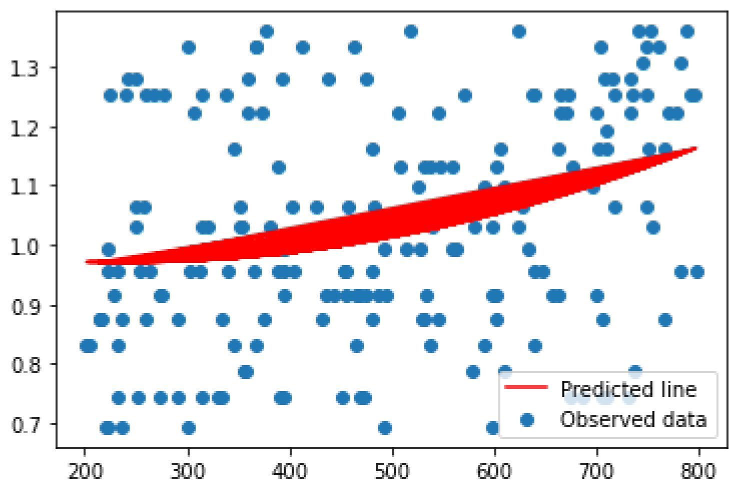
| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 1.0056 | 0.110 | 9.112 | 0.000 | 0.788 | 1.223 |
| **SS** | -0.0003 | 0.000 | -0.607 | 0.545 | -0.001 | 0.001 |
| **I(SS * SS)** | 6.142e-07 | 4.79e-07 | 1.284 | 0.201 | -3.3e-07 | 1.56e-06 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 8.598 | **Durbin-Watson:** | 1.357 |
| **Prob(Omnibus):** | 0.014 | **Jarque-Bera (JB):** | 4.118 |
| **Skew:** | 0.046 | **Prob(JB):** | 0.128 |
| **Kurtosis:** | 2.303 | **Cond. No.** | 2.79e+06 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.79e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [24]:

```
pred4 = model4.predict(pd.DataFrame(cal.SS))
pred4_at = np.exp(pred4)
pred4_at

# Regression line
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 2)
X = cal.iloc[:, 0:1].values
X_poly = poly_reg.fit_transform(X)
# y = wcat.iloc[:, 1].values


plt.scatter(cal.SS, np.log(cal.Gpa))
plt.plot(X, pred4, color = 'red')
plt.legend(['Predicted line', 'Observed data'])
plt.show()

# Error calculation
res4 = cal.Gpa - pred4_at
res_sqr4 = res4 * res4
mse4 = np.mean(res_sqr4)
rmse4 = np.sqrt(mse4)
rmse4
```




Out[24]: 0.5144912487746158

In [25]:

```
# Choose the best model using RMSE
data = {"MODEL":pd.Series(["SLR", "Log model", "Exp model", "Poly model"]), "RMSE
table_rmse = pd.DataFrame(data)
table_rmse
```

Out[25]:

| | MODEL | RMSE |
|---|---|---|
| **0** | SLR | 0.515946 |
| **1** | Log model | 0.518490 |
| **2** | Exp model | 0.517588 |
| **3** | Poly model | 0.514491 |

In [26]:

```
###################
# The best model

from sklearn.model_selection import train_test_split

train, test = train_test_split(cal, test_size = 0.3)

finalmodel = smf.ols('np.log(Gpa) ~ SS + I(SS*SS)', data = train).fit()
finalmodel.summary()
```

Out[26]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | np.log(Gpa) | **R-squared:** | 0.106 |
| **Model:** | OLS | **Adj. R-squared:** | 0.093 |
| **Method:** | Least Squares | **F-statistic:** | 8.113 |
| **Date:** | Sat, 19 Jun 2021 | **Prob (F-statistic):** | 0.000468 |
| **Time:** | 09:42:44 | **Log-Likelihood:** | 42.766 |
| **No. Observations:** | 140 | **AIC:** | -79.53 |
| **Df Residuals:** | 137 | **BIC:** | -70.71 |
| **Df Model:** | 2 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 0.9935 | 0.132 | 7.526 | 0.000 | 0.732 | 1.255 |
| **SS** | -0.0003 | 0.001 | -0.528 | 0.598 | -0.001 | 0.001 |
| **I(SS * SS)** | 6.43e-07 | 5.66e-07 | 1.135 | 0.258 | -4.77e-07 | 1.76e-06 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 3.861 | **Durbin-Watson:** | 2.230 |
| **Prob(Omnibus):** | 0.145 | **Jarque-Bera (JB):** | 2.566 |
| **Skew:** | 0.143 | **Prob(JB):** | 0.277 |
| **Kurtosis:** | 2.402 | **Cond. No.** | 2.78e+06 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.78e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [27]:
```
# Predict on test data
test_pred = finalmodel.predict(pd.DataFrame(test))
pred_test_AT = np.exp(test_pred)
pred_test_AT

# Model Evaluation on Test data
test_res = test.Gpa - pred_test_AT
test_sqrs = test_res * test_res
test_mse = np.mean(test_sqrs)
test_rmse = np.sqrt(test_mse)
test_rmse
```

Out[27]: 0.537987761257369

In [28]:
```


# Prediction on train data
train_pred = finalmodel.predict(pd.DataFrame(train))
pred_train_AT = np.exp(train_pred)
pred_train_AT


# Model Evaluation on train data
train_res = train.Gpa - pred_train_AT
train_sqrs = train_res * train_res
train_mse = np.mean(train_sqrs)
train_rmse = np.sqrt(train_mse)
train_rmse
```

Out[28]: 0.5082608011769393

# Summary

Model having highest R-Squared value is better i.e. (model=0.897 is not better than model1=0.960). There has good relationship>0.85

RMSE- lower the RMSE incidcate better fit. RMSE is a good measure of how accuaracy the model predict the reponse. In Linear regression RMSE value between 0.2 to 0.5

But in final model trainig and traing we choose Polynomial transformation np.log(Gpa) ~ SS + I(SS*SS) beacause the training rmse was show good result in Polynomial transformation rather than SLr,Log.

In [ ]: