Simple Linear Regression_CSAT_GPA

June 5, 2026

1 Objective: Given CSAT Score, find / predict GPA

[60]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[61]:
```
df = pd.read_csv("CSAT_GPA.csv")
```

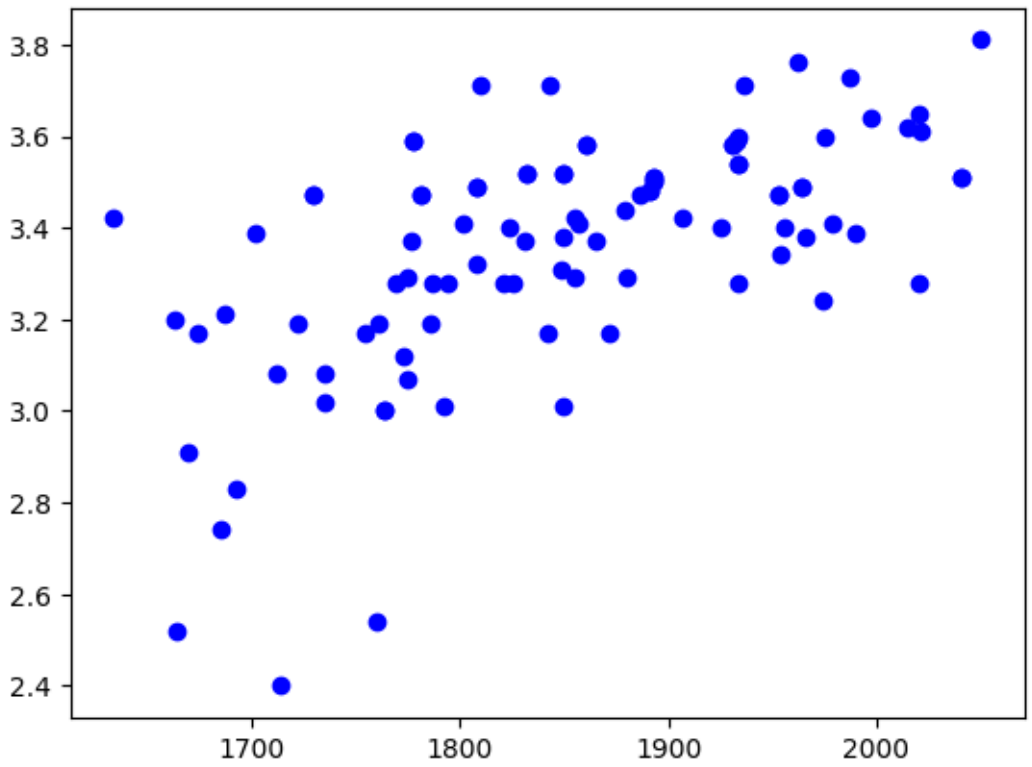
[80]:
```
df.head(10)
```

[80]:

	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

[63]:
```
plt.scatter(df['SAT'], df['GPA'], color='blue')
```

[63]:
```
<matplotlib.collections.PathCollection at 0x237074274d0>
```



[64]:
```
df.describe()
```

[64]:

	SAT	GPA
count	100.000000	100.000000
mean	1850.760000	3.360500
std	101.566331	0.258845
min	1634.000000	2.400000
25%	1776.500000	3.270000
50%	1850.000000	3.410000
75%	1934.000000	3.520000
max	2050.000000	3.810000

[65]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   SAT     100 non-null    int64
 1   GPA     100 non-null    float64
dtypes: float64(1), int64(1)
memory usage: 1.7 KB
```

1.0.1 Defining X and y {assigning one column to X and the other column to y}

[66]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

1.0.2 from sklearn get train_test_split

[67]:
```
from sklearn.model_selection import train_test_split
```

[69]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3,␣
↪random_state = 0)
```

[70]:
```
print(f"X_train contains {len(X_train)} items")
print(f"\ty_train contains {len(y_train)} items")

print(f"X_test contains {len(X_test)} items")
print(f"\ty_test contains {len(y_test)} items")
```

```
X_train contains 66 items
        y_train contains 66 items
X_test contains 34 items
        y_test contains 34 items
```

[71]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

[71]:
```
LinearRegression()
```

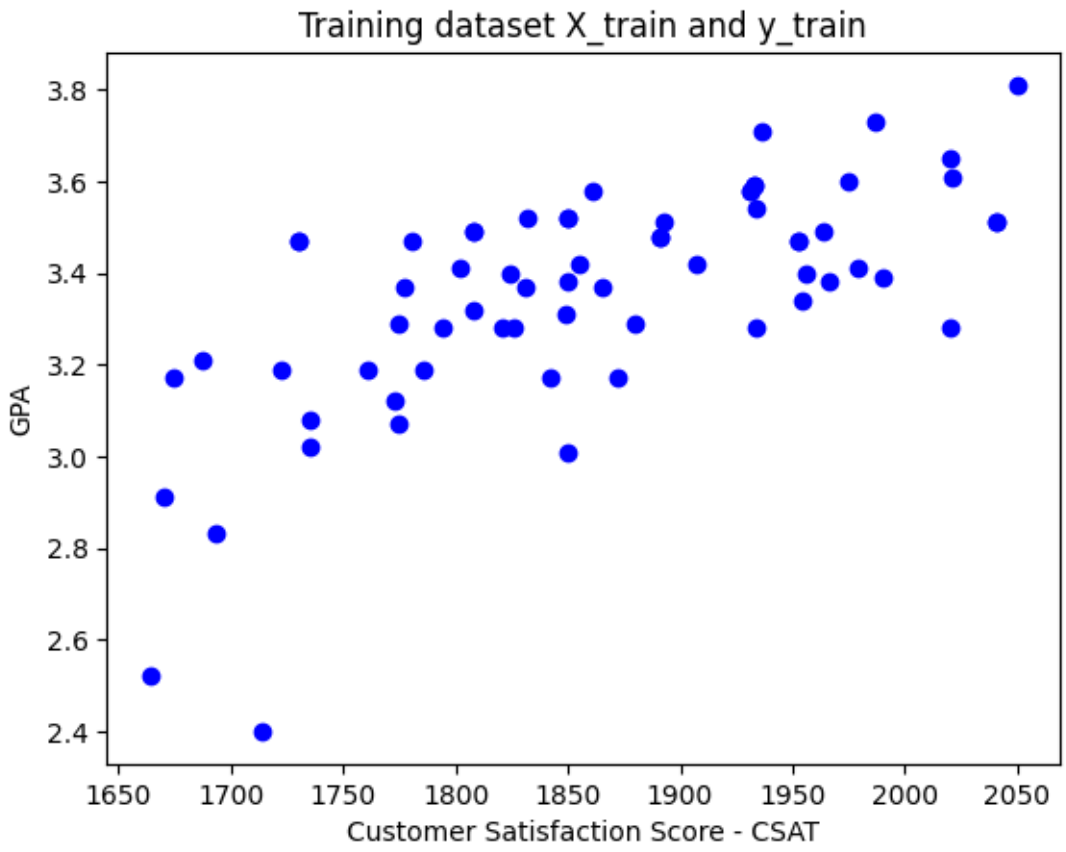
[72]:
```
y_pred = regressor.predict(X_test)
```

[73]:
```
y_pred
```

[73]:
```
array([3.23444279, 3.22458842, 3.19009815, 3.39868219, 3.60890863,
       3.30835052, 3.18188618, 3.47587471, 3.38554304, 3.47587471,
       2.9831565 , 3.40853655, 3.32641685, 3.11126323, 3.19666772,
       3.20487969, 3.03078592, 3.54157047, 3.34612558, 3.24265476,
       3.09483929, 3.40853655, 3.06691859, 3.21966124, 3.46109316,
       3.34941037, 3.19666772, 3.21966124, 3.52186174, 3.57934553,
       3.52514653, 3.27221785, 3.40853655, 3.35597994])
```

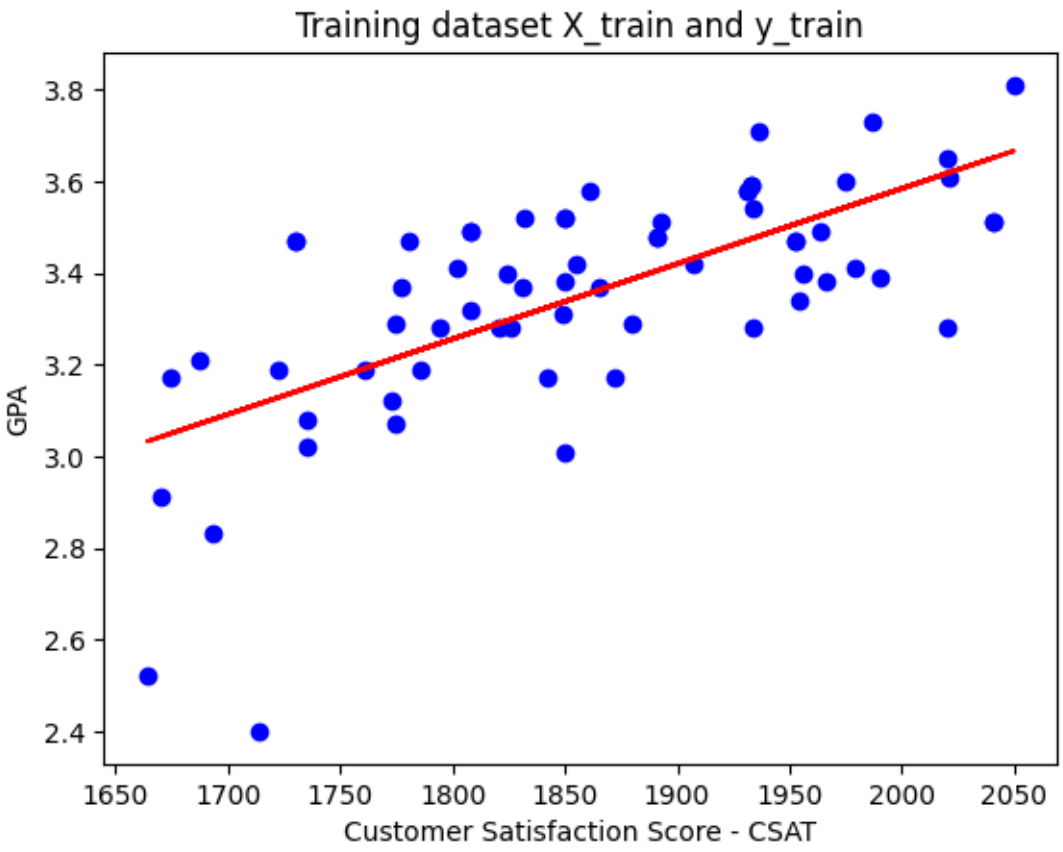
[74]:
```
plt.scatter(X_train, y_train, color = 'blue')
plt.title ("Training dataset X_train and y_train")
plt.xlabel("Customer Satisfaction Score - CSAT")
plt.ylabel ("GPA")
```

[74]:
```
Text(0, 0.5, 'GPA')
```



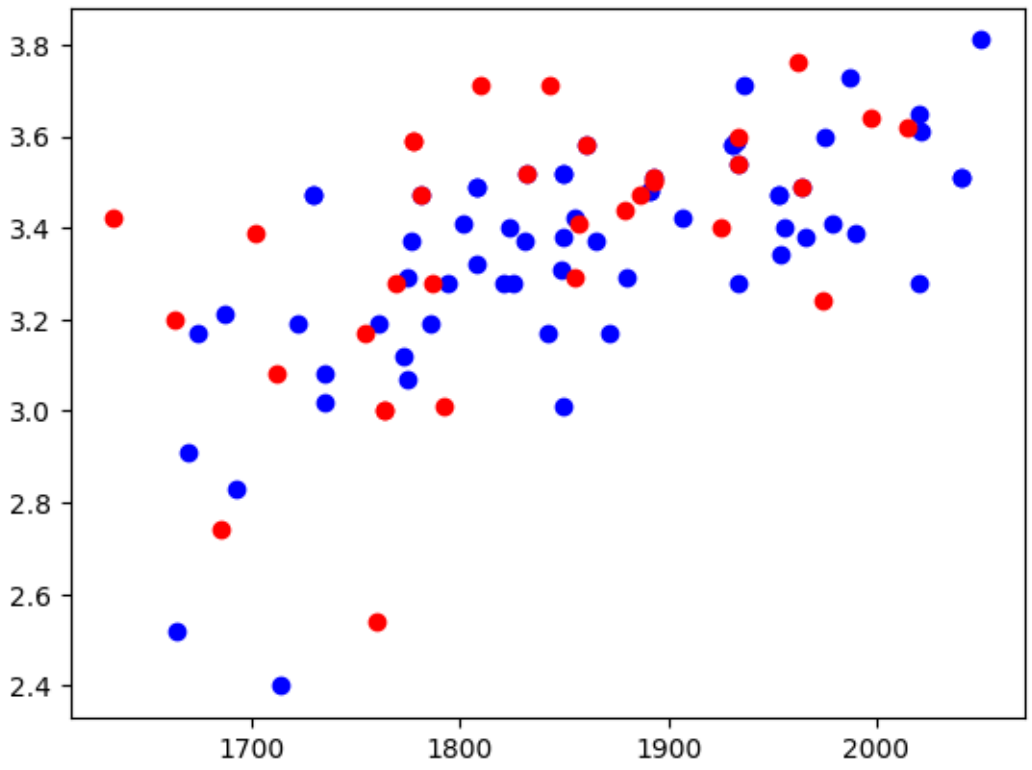
[75]:
```
plt.scatter(X_train, y_train, color = 'blue')
plt.plot(X_train, regressor.predict(X_train), color = 'red')

plt.title ("Training dataset X_train and y_train")
plt.xlabel("Customer Satisfaction Score - CSAT")
plt.ylabel ("GPA")
```

[75]:
```
Text(0, 0.5, 'GPA')
```



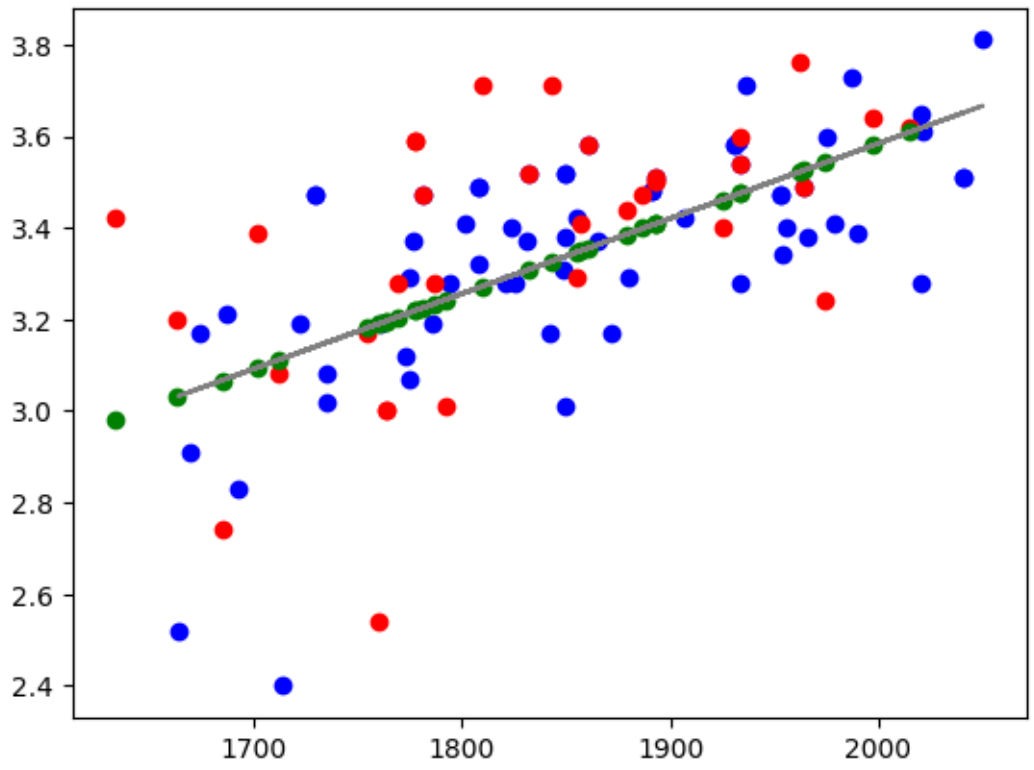
[76]:
```
plt.scatter(X_train, y_train, color = 'blue')
plt.scatter(X_test, y_test, color = "red")
```

[76]: <matplotlib.collections.PathCollection at 0x237076d7890>



[77]:
```
plt.scatter(X_train, y_train, color = 'blue')
plt.scatter(X_test, y_test, color = 'red')
plt.scatter(X_test, y_pred, color = 'green')
plt.plot(X_train, regressor.predict(X_train), color = 'grey')
```

[77]: [<matplotlib.lines.Line2D at 0x237077359d0>]



[82]:
```
print(regressor.predict([[100], [500], [1000], [2000], [3000]]))
```

```
[0.46372404 1.12068165 1.94187867 3.58427272 5.22666676]
```

[83]:
```
slope = f"{regressor.coef_[0]: .3f}"
yint = f"{regressor.intercept_: .3f}"
print(f"The slope of line is {slope} and its y-intercept is at {yint}")
```

```
The slope of line is  0.002 and its y-intercept is at  0.299
```

[87]:
```
df['pred GPA'] = float(slope) * df['GPA'] + float(yint)
df['pGPA - aGPA'] = df['pred GPA'] - df['GPA']
```

[88]:
```
df
```

[88]:

	SAT	GPA	pred GPA	pGPA - aGPA
0	1714	2.40	0.30380	-2.09620
1	1664	2.52	0.30404	-2.21596
2	1760	2.54	0.30408	-2.23592
3	1685	2.74	0.30448	-2.43552
4	1693	2.83	0.30466	-2.52534
..	…	…	…	…
95	1934	3.54	0.30608	-3.23392

```
96  1861  3.58   0.30616     -3.27384
97  1931  3.58   0.30616     -3.27384
98  1933  3.59   0.30618     -3.28382
99  1778  3.59   0.30618     -3.28382

[100 rows x 4 columns]
```

1.1 SLR - Accuracy Metrics calculations

Metric, Perfect Score, Units, What to look for

R2 Score, 1.0, None (Ratio), Higher is better.

MAE, 0.0, Same as target variable, Lower is better; tells you the average error.

RMSE, 0.0, Same as target variable, Lower is better; pay attention if it is much higher than MAE.

[91]:
```python
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import numpy as np

# y_test = actual values, y_pred = your model's predictions
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"\tR2 Score (Variance Explained): {r2:.2f}")
print(f"\tMean Absolute Error: {mae:.2f}")
print(f"\tRoot Mean Squared Error: {rmse:.2f}")
```

```
        R2 Score (Variance Explained): 0.23
        Mean Absolute Error: 0.19
        Root Mean Squared Error: 0.24
```

[ ]:
```python

```