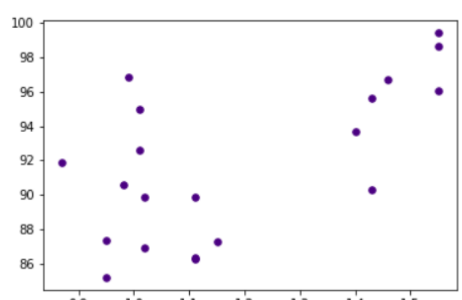
Creating the dataframe

I did my homework on Google Colab using Python. I began by importing the libraries I would use, and by creating the dataframe of the given data. By using the method `describe()` I was able to quickly find certain statistical data of the dataframe as seen below. I also plotted a scatter chart of the values to get an initial look, and it didn't look very linear to me.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import math
from scipy.stats import t
import copy

data = {"Purity":
        [86.91, 89.85, 90.28, 86.34, 92.58, 87.33, 86.29, 91.86, 95.61, 89.86,
         96.73, 99.42, 98.66, 96.07, 93.65, 87.31, 95.00, 96.85, 85.20, 90.56],
        "Hydrocarbon":
        [1.02 , 1.11 , 1.43 , 1.11 , 1.01 , 0.95 , 1.11 , 0.87 , 1.43 , 1.02 ,
         1.46 , 1.55 , 1.55 , 1.55 , 1.40 , 1.15 , 1.01 , 0.99 , 0.95 , 0.98 ]}
df = pd.DataFrame(data)
df.describe()
```

	Purity	Hydrocarbon
count	20.000000	20.000000
mean	91.818000	1.182500
std	4.478882	0.236752
min	85.200000	0.870000
25%	87.325000	1.005000
50%	91.210000	1.110000
75%	95.725000	1.430000
max	99.420000	1.550000

```
plt.scatter(df["Hydrocarbon"], df["Purity"],
            color = "indigo", marker = "o", s = 30)
```



Part a – Simple Linear Regression

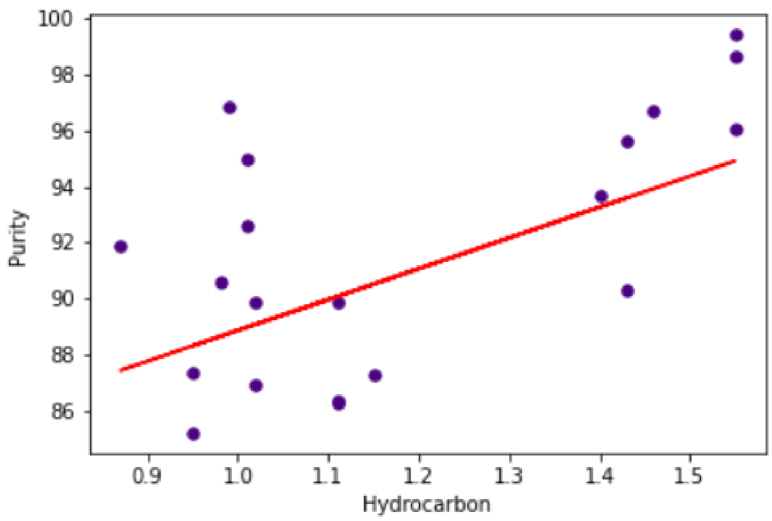
In this part I first calculated the sum of squares (SS_{xx} and SS_{xy}), their respective least square estimators β_0 and β_1, as well as the predicted y values & the residual values (e). The latter two I've added to the dataframe. The results I calculated (see below) are correct, as they are the same values we can obtain by using the built-in regression functions of Python libraries (*shown in my .ipynb file online*). On the next page you can see the plotted regression model.

```
n = len(df["Hydrocarbon"])  # n=20
x = df["Hydrocarbon"]
y = df["Purity"]

# Calculating Sxx and Sxy
Sxx = np.sum(x*x) - np.sum(x)**2 / n
Sxy = np.sum(x*y) - np.sum(x) * np.sum(y) / n

# Calculating B1 and B0 estimators
B1 = Sxy / Sxx              # slope
B0 = P_mean - B1*H_mean     # intercept

# Regression Line & Residual e
y_hat = B0 + B1*x
e = y - y_hat      # sum is equal to 0

df2 = df
df2["y_hat"], df2["Residual e"] = y_hat, e

print("Sxx =",Sxx,"\nSxy =",Sxy)
print("B\u2080 =", B0,"\nB\u2081 =", B1)
print("\u0177  =", round(B0,2), "\u2212", round(B1,2), "x")
df2.head()
```

```
Sxx = 1.064975000000004
Sxy = 12.567799999999806
B₀ = 77.8632841616003
B₁ = 11.801028193149849
ŷ  = 77.86 − 11.8 x
```

	Purity	Hydrocarbon	y_hat	Residual e
0	86.91	1.02	89.900333	-2.990333
1	89.85	1.11	90.962425	-1.112425
2	90.28	1.43	94.738754	-4.458754
3	86.34	1.11	90.962425	-4.622425
4	92.58	1.01	89.782323	2.797677

```
# plotting the regression graph
plt.scatter(x, y, color = "indigo", marker = "o", s = 30)
plt.plot(x, y_hat, color = "r")
plt.xlabel("Hydrocarbon"), plt.ylabel("Purity")
plt.show()
```



Part b – Testing the hypothesis $H_0 : \beta_1 = 0$

I did a hypothesis test for the slope to see the linear relationship between the response and the regressor. For this test I took alpha to be 0.5 and found $t_{\alpha/2,n-2}$ as ~2.101. When I did this, I found that we should reject H_0 because the value of $t_0 > t_{\alpha/2,n-2}$. Since the p-value (*on the file*) is also less than the alpha level of 0.05, we can reject the null hypothesis. This would imply that there is a linear relationship between y and x. I did the same for the intercept (*on file*).

```
alpha = 0.05
tt = (t.ppf(1 - (alpha/2), df=n-2))

MSres = sum(e**2)/(n-2)
se_B1 = math.sqrt(MSres / Sxx)
t0 = (B1-0)/se_B1
print("t\u2080 =",round(t0,3),"and t =",round(tt,3),"\n")
print("t\u2080 > t\nReject H\u2080") if t0 > tt else print(("t\u2080 < t\nFail to reject H0"))
```

```
t₀ = 3.386 and t = 2.101
t₀ > t
Reject H₀
```

Part c – Calculating R^2

Here I calculated the coefficient of determination R^2 using the Sum of Squares values. The result shows that only 38.9% of the data actually fits the regression model. That is not a very high amount, so we can say our model is in fact not very reliable.

```
SSr = B1 * Sxy
SSres = MSres * (n-2)
SSt = SSr + SSres
R2 = SSr / SSt  # or R2 = 1 - (SSres/SSt)
print("R\u00b2 =", round(R2,3),"or",round(R2*100,1),"%")
```

```
R² = 0.389 or 38.9 %
```

Part d – 95% CI of the Slope

I calculated the two-sided Confidence Interval of the slope with alpha being 0.05 (because it is 95%). Since we're calculating for the slope, I used the least square estimator β_1 for my calculation because, along with the standard error of the slope.

```
CI_low = B1 - tt*se_B1
CI_up = B1 + tt*se_B1
print("B\u2081 \t± t \t* se(B\u2081)")
print(round(B1,3),"\t±",round(tt,3),"*",round(se_B1,4))
print("\nCI: ",round(CI_low,2),"< B1 <",round(CI_up,2))
```

```
B₁      ± t      * se(B₁)
11.801 ± 2.101 * 3.4851

CI:  4.48 < B1 < 19.12
```

Part e – 95% CI of mean Purity when Hydrocarbon = 1%

For this Confidence Interval I did calculations in a similar way, except that I recalculated the y values, the sum of squares, the residuals, and the mean square of the residuals. I used the following formula to find the upper and lower limits of the confidence interval:

$$\hat{\mu}_{y|x_0} \pm t_{\alpha/2,n-2}\sqrt{MS_{\mathrm{Res}}\left[\frac{1}{n}+\frac{(x_0-\bar{x})^2}{S_{xx}}\right]}$$

```
H_perc = 1.00
y_hat1 = B0 + B1 * H_perc
Sxx1 = np.sum(1*1) - np.sum(1)**2 / n
e1 = copy.deepcopy(e)
e1.add(y.mean() - y_hat1)
MSres1 = round(sum(e1**2)/(n-2),1)
var = MSres1 * (1/n + round((H_perc - x.mean())**2,2) / Sxx1)

print("\u0177 \t ±  t \t\t* √( MSres*(1/n + (x\u2080-x̄)\u00b2/Sxx) )")
print(round(y_hat1,3),"\t ± ",round(tt,3),"\t*",round(math.sqrt(var),4))
CI_low2 = y_hat1 - tt * math.sqrt(var)
CI_up2 = y_hat1 + tt * math.sqrt(var)
print("\nCI: ",round(CI_low2,2),"< µ <",round(CI_up2,2))
```

```
ŷ        ±  t      * √( MSres*(1/n + (x₀-x̄)²/Sxx) )
89.664   ±  2.101  * 1.0259

CI:  87.51 < µ < 91.82
```

Links to homework project

Homework Folder

https://drive.google.com/drive/folders/1sDywCgZRuEaekqlPFMLErjcPkYkP3dIr?usp=sharing

.ipynb code file

https://colab.research.google.com/drive/1T1aKljcx7uTItavir0zHgeCI-lXbbpYd?usp=sharing