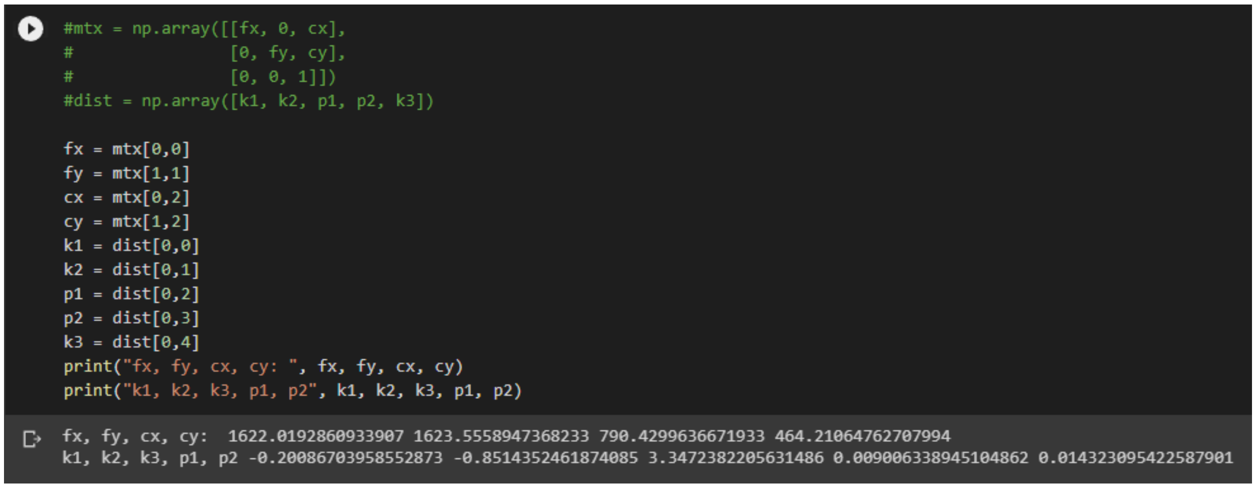
Kunnithi Treecharoensomboon ID 63070503461

CPE383 Machine Learning: Quiz5

1. Calibrate Camera. 2 hrs. Using the "Calibrate Camera by ChatGPT" program shown in class, calibrate your laptop or mobile phone camera to find its intrinsic parameters using 10-15 checkerboard images. Make sure you are not using mirror images. If the processing is slow, it may help to reduce the size of each image to a width of around 1,000.

   1.1. 10 points. Report the fx, fy, cx, cy, and lens distortion (k1, k2, k3, p1, p2) parameters found using left##.jpg, frame-##.png, and your camera's images.

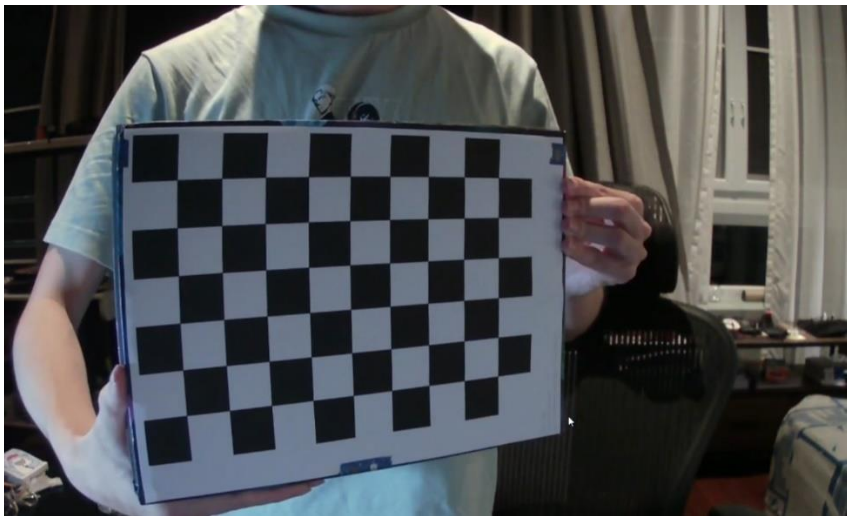


```
#mtx = np.array([[fx, 0, cx],
#                [0, fy, cy],
#                [0, 0, 1]])
#dist = np.array([k1, k2, p1, p2, k3])

fx = mtx[0,0]
fy = mtx[1,1]
cx = mtx[0,2]
cy = mtx[1,2]
k1 = dist[0,0]
k2 = dist[0,1]
p1 = dist[0,2]
p2 = dist[0,3]
k3 = dist[0,4]
print("fx, fy, cx, cy: ", fx, fy, cx, cy)
print("k1, k2, k3, p1, p2", k1, k2, k3, p1, p2)
```

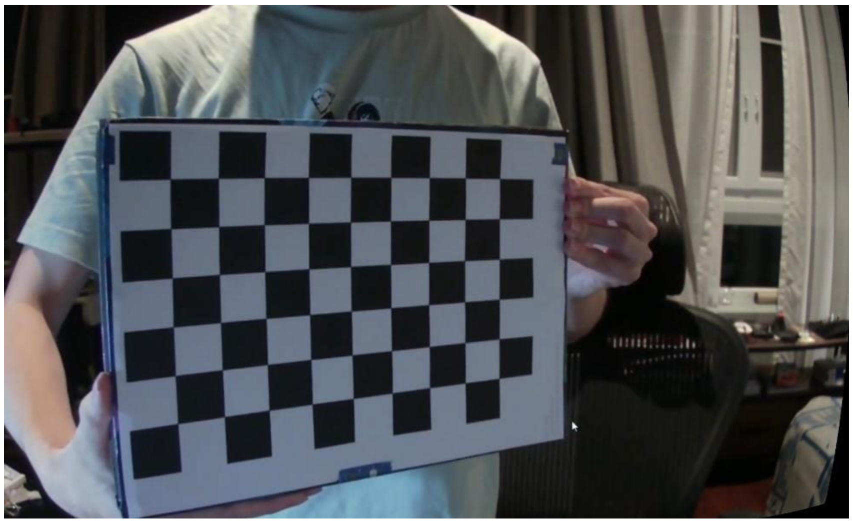
```
fx, fy, cx, cy:  1622.0192860933907 1623.5558947368233 790.4299636671933 464.21064762707994
k1, k2, k3, p1, p2 -0.20086703958552873 -0.8514352461874085 3.3472382205631486 0.009006338945104862 0.014323095422587901
```

1.2. 10 points. Show the Original and Undistorted image for one of your checkerboard images. Draw straight lines across the original image and undistorted image to see if the distortion has improved.

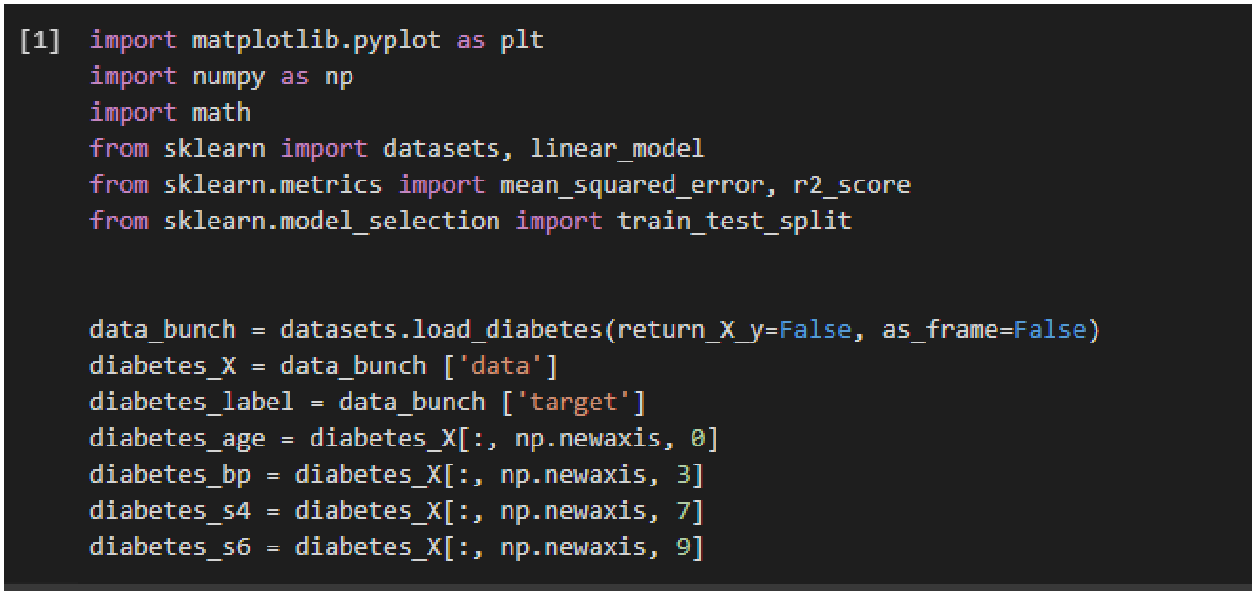
Distorted



Undistorted

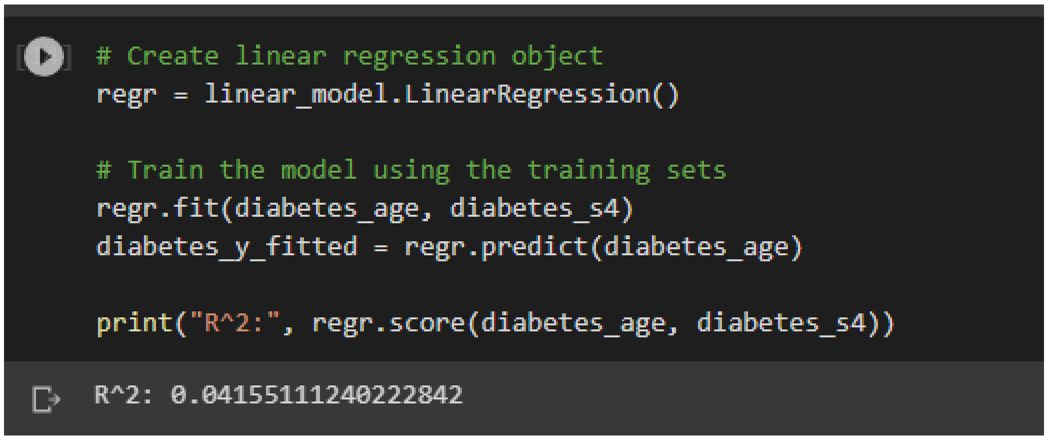


2. Using the Regression on diabetes data example:



```
[1]  import matplotlib.pyplot as plt
     import numpy as np
     import math
     from sklearn import datasets, linear_model
     from sklearn.metrics import mean_squared_error, r2_score
     from sklearn.model_selection import train_test_split


     data_bunch = datasets.load_diabetes(return_X_y=False, as_frame=False)
     diabetes_X = data_bunch ['data']
     diabetes_label = data_bunch ['target']
     diabetes_age = diabetes_X[:, np.newaxis, 0]
     diabetes_bp = diabetes_X[:, np.newaxis, 3]
     diabetes_s4 = diabetes_X[:, np.newaxis, 7]
     diabetes_s6 = diabetes_X[:, np.newaxis, 9]
```

2.1. 5 points. 1 hr. Is age highly correlated with total cholesterol / HDL (column ‘S4’)?

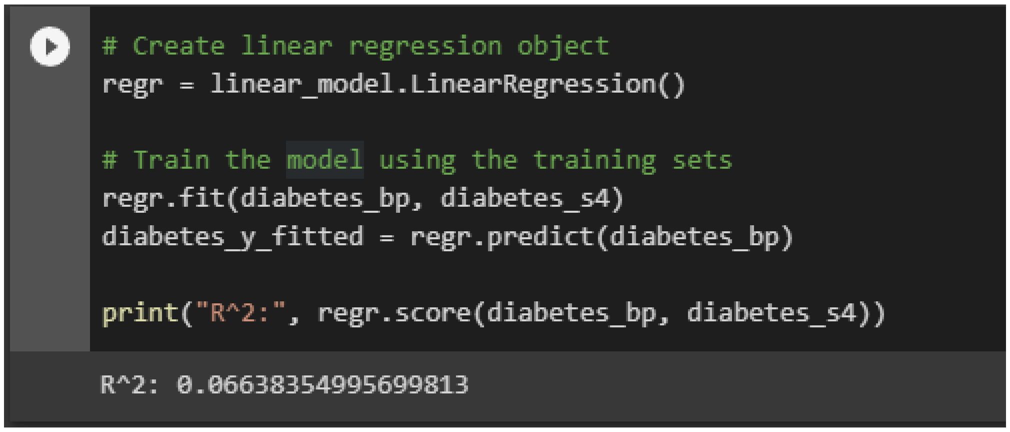


```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_age, diabetes_s4)
diabetes_y_fitted = regr.predict(diabetes_age)

print("R^2:", regr.score(diabetes_age, diabetes_s4))
```

```
R^2: 0.04155111240222842
```

ANS only a small proportion of the variation in diabetes_s4 can be explained by diabetes_age. Therefore, the correlation between the two variables is weak.

2.2. 5 points. 0.5 hr. Is blood pressure highly correlated with total cholesterol / HDL (column 'S4')?

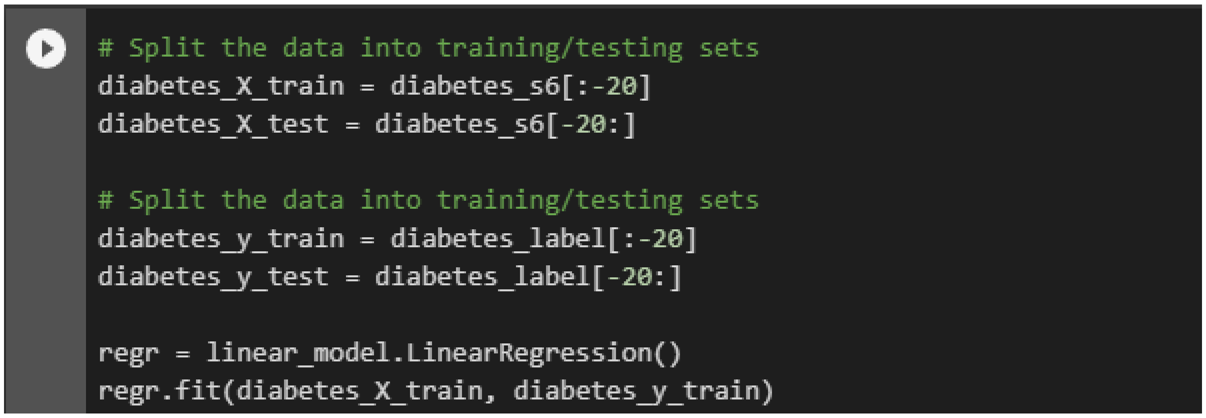


```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_bp, diabetes_s4)
diabetes_y_fitted = regr.predict(diabetes_bp)

print("R^2:", regr.score(diabetes_bp, diabetes_s4))
```

```
R^2: 0.06638354995699813
```

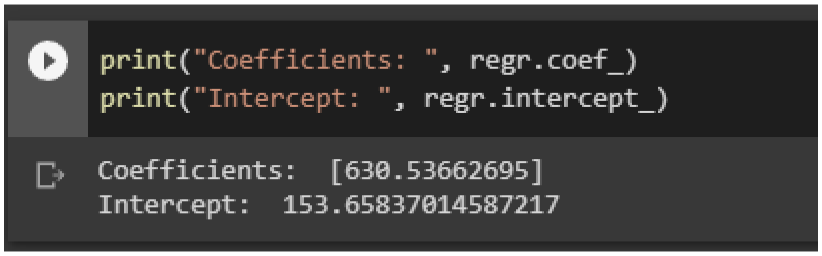
ANS only a small proportion of the variation in diabetes_s4 can be explained by diabetes_bp. Therefore, the correlation between the two variables is weak.

2.3. 15 points (4+3+3+5). 1 hr. Report Linear fit results for y = ax + b where x is the blood sugar level



```
# Split the data into training/testing sets
diabetes_X_train = diabetes_s6[:-20]
diabetes_X_test = diabetes_s6[-20:]

# Split the data into training/testing sets
diabetes_y_train = diabetes_label[:-20]
diabetes_y_test = diabetes_label[-20:]

regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
```

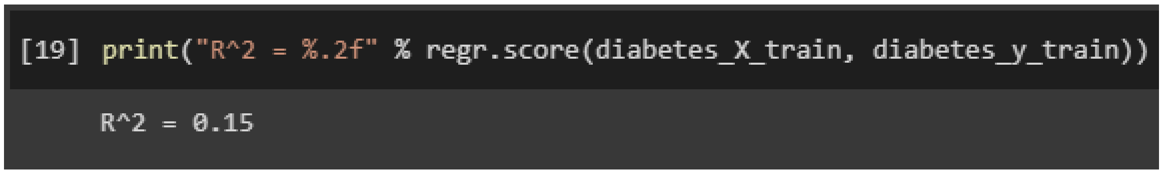
i. Linear fit coefficients and intercept of the training data



```
print("Coefficients: ", regr.coef_)
print("Intercept: ", regr.intercept_)
```

```
Coefficients:  [630.53662695]
Intercept:  153.65837014587217
```

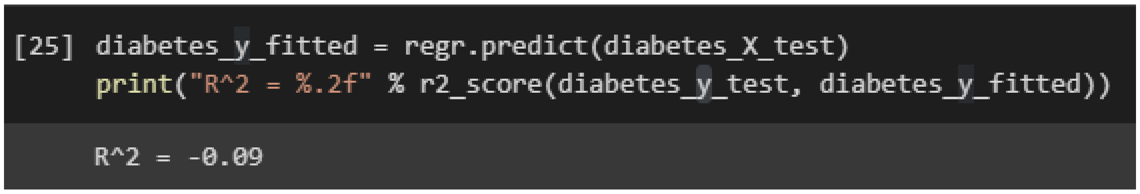
ii. What is the R^2 for the training data?



```
[19] print("R^2 = %.2f" % regr.score(diabetes_X_train, diabetes_y_train))
```

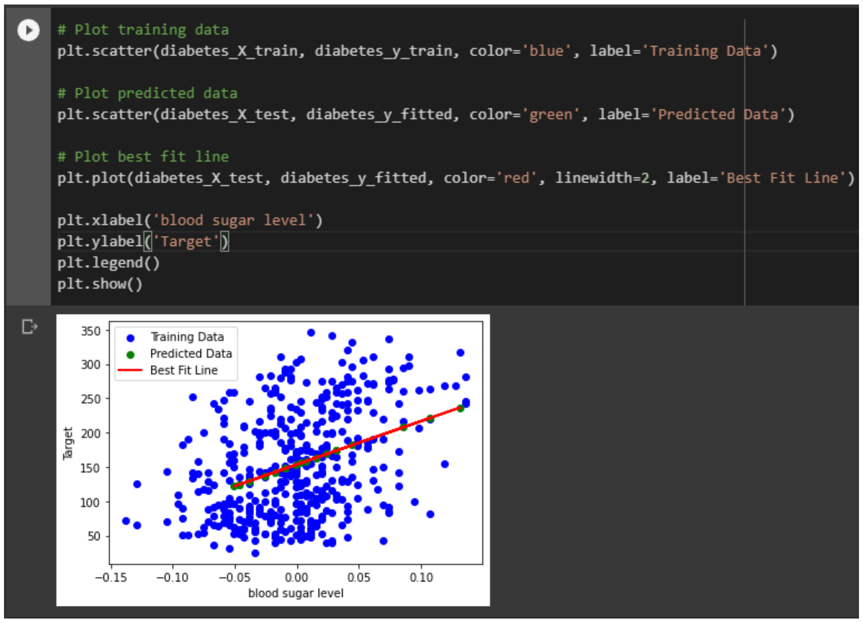
```
R^2 = 0.15
```

iii. What is the R^2 for the prediction of y based on blood sugar level for the test data?



```
[25] diabetes_y_fitted = regr.predict(diabetes_X_test)
     print("R^2 = %.2f" % r2_score(diabetes_y_test, diabetes_y_fitted))

     R^2 = -0.09
```

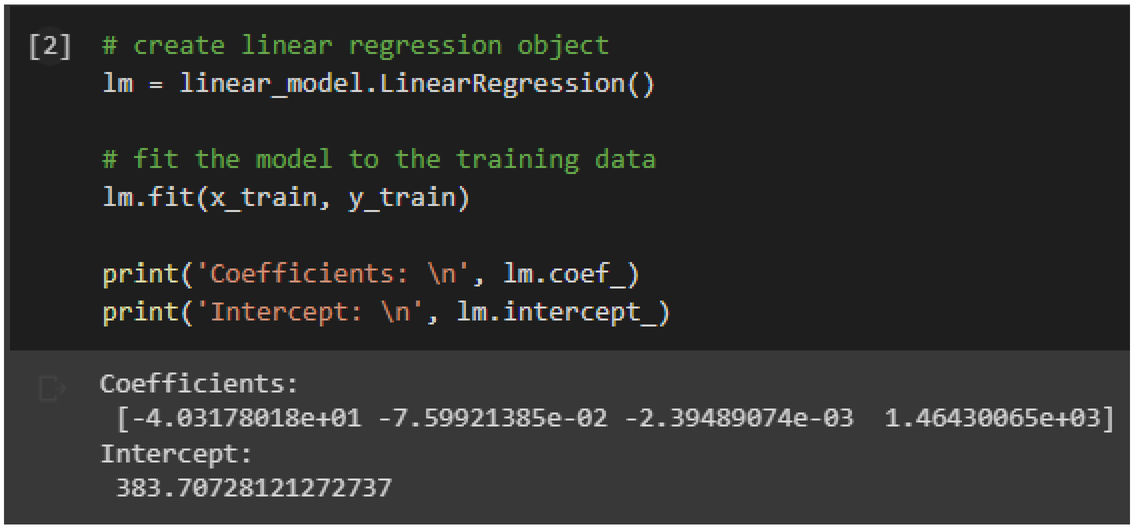
iv. Show a scatter plot of the train set (x, y) as blue circles and predicted (x, y) as green circles. Also show the best fit line in red.



```
# Plot training data
plt.scatter(diabetes_X_train, diabetes_y_train, color='blue', label='Training Data')

# Plot predicted data
plt.scatter(diabetes_X_test, diabetes_y_fitted, color='green', label='Predicted Data')

# Plot best fit line
plt.plot(diabetes_X_test, diabetes_y_fitted, color='red', linewidth=2, label='Best Fit Line')

plt.xlabel('blood sugar level')
plt.ylabel('Target')
plt.legend()
plt.show()
```

3. 2 hrs. Use the data provided in the shared file gasoline_use.txt to:

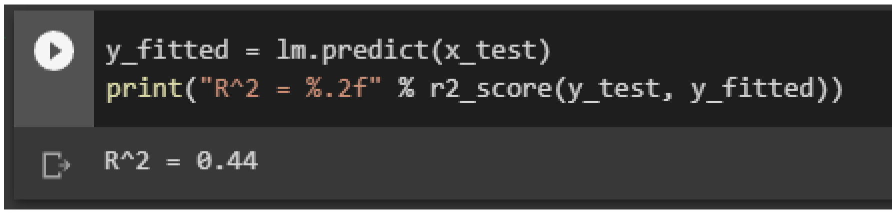
3.1. 10 points. Show the equation found by fitting the training data: y = f(x1, x2, x3, x4) = a0+ a1x1 + a2x2 + a3x3 + a4x4



```
[2] # create linear regression object
    lm = linear_model.LinearRegression()

    # fit the model to the training data
    lm.fit(x_train, y_train)

    print('Coefficients: \n', lm.coef_)
    print('Intercept: \n', lm.intercept_)

    Coefficients:
     [-4.03178018e+01 -7.59921385e-02 -2.39489074e-03  1.46430065e+03]
    Intercept:
     383.70728121272737
```

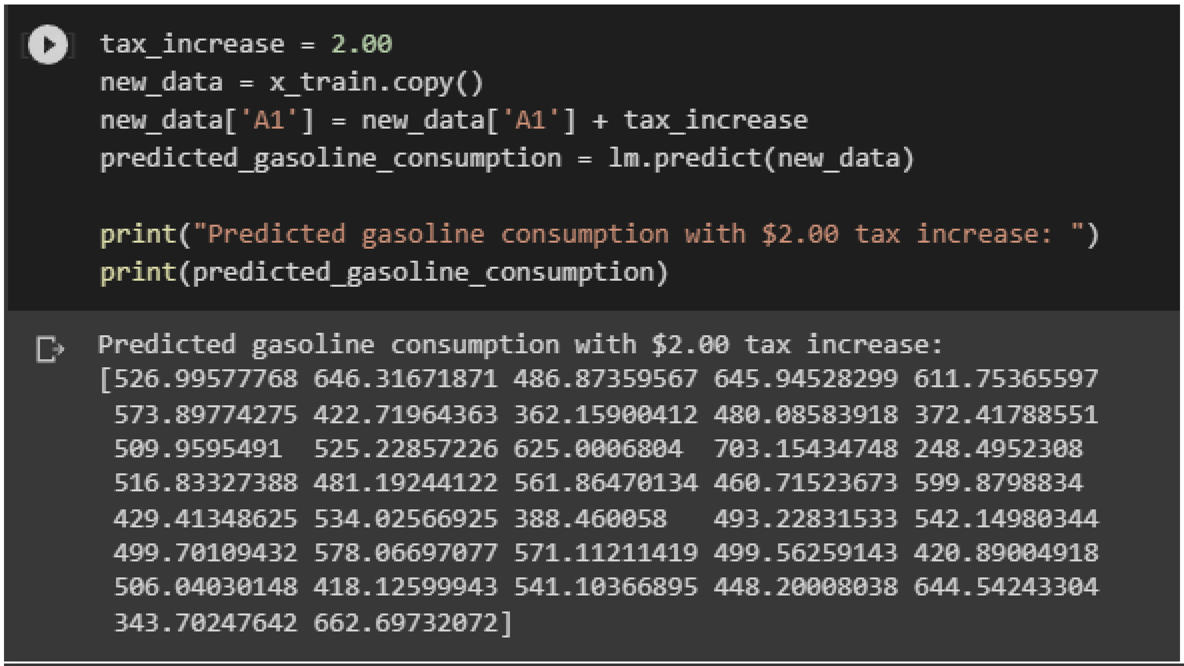
ANS y = -40.32x1 - 0.08x2 - 0.002x3 + 1464.30x4 + 383.71

3.2. 5 points. What is the R^2 for the prediction of y? Use testing data.



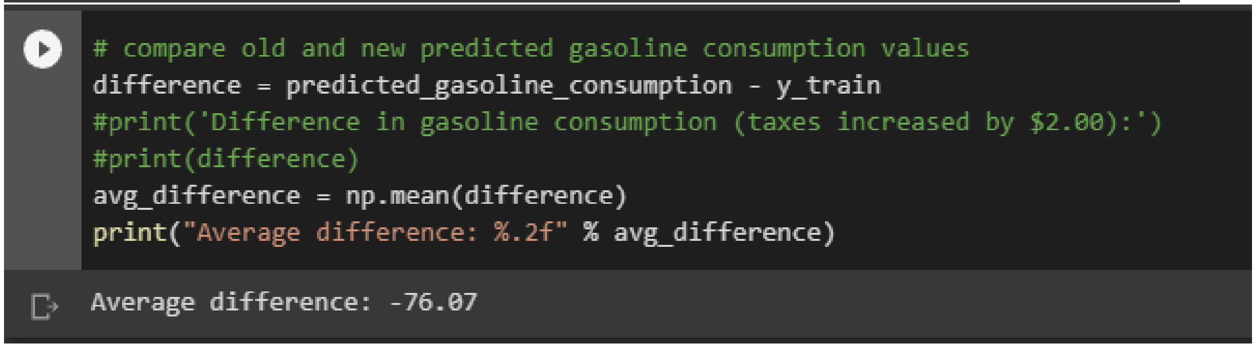
```
y_fitted = lm.predict(x_test)
print("R^2 = %.2f" % r2_score(y_test, y_fitted))

R^2 = 0.44
```

3.2. 5 points. What would happen to gasoline consumption if taxes are increased by $2.00? Use training data.



```python
tax_increase = 2.00
new_data = x_train.copy()
new_data['A1'] = new_data['A1'] + tax_increase
predicted_gasoline_consumption = lm.predict(new_data)

print("Predicted gasoline consumption with $2.00 tax increase: ")
print(predicted_gasoline_consumption)
```

```
Predicted gasoline consumption with $2.00 tax increase:
[526.99577768 646.31671871 486.87359567 645.94528299 611.75365597
 573.89774275 422.71964363 362.15900412 480.08583918 372.41788551
 509.9595491  525.22857226 625.0006804  703.15434748 248.4952308
 516.83327388 481.19244122 561.86470134 460.71523673 599.8798834
 429.41348625 534.02566925 388.460058   493.22831533 542.14980344
 499.70109432 578.06697077 571.11211419 499.56259143 420.89004918
 506.04030148 418.12599943 541.10366895 448.20008038 644.54243304
 343.70247642 662.69732072]
```



```python
# compare old and new predicted gasoline consumption values
difference = predicted_gasoline_consumption - y_train
#print('Difference in gasoline consumption (taxes increased by $2.00):')
#print(difference)
avg_difference = np.mean(difference)
print("Average difference: %.2f" % avg_difference)
```

```
Average difference: -76.07
```

ANS it’s tell that on average, the model predicts that gasoline consumption will decrease by approximately 76.07 units, but in my opinion tax is not relate to gasoline consumption at all.