Sean Wendlandt 5/16/23

COMP 4448: Data Science Tools II Assignment 6

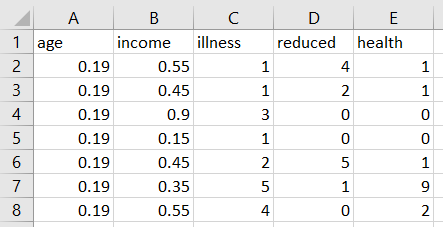
**Directions:** Do this assignment in Jupyter Notebook and provide screenshots of code and output in this word document wherever required. You will upload this word document containing screenshots of code and answers as well as your Jupyter Notebook to Canvas. All assignments will be submitted and graded through canvas and grades will be transferred to the 2U platform.

**Goal:** The goal of this assignment is to give you the opportunity to implement the Gradient Algorithm from scratch as well as using tools built into sklearn. You will also run regression analysis, evaluate, test assumptions and provide interpretations using your own data.

**Packages:** Core packages you may need for this assignment include numpy, pandas, sklearn, matplotlib.pyplot and/or seaborn.

**Question 1**

Use the entire health data provided on canvas for this question. Here are the first few rows of the data.



The variables are described as follows:

age = Age in years divided by 100.

income = Annual income in tens of thousands of dollars.

illness = Number of illnesses in past 2 weeks.

reduced = Number of days of reduced activity in past 2 weeks due to illness or injury.

health = General health questionnaire score using Goldberg's method. Note that higher health scores indicate worse health condition.

Use the **age, income, illness** and **reduced** variables as input variables and use the **health** variable as an output variable. You will find the parameters or coefficients (intercept and slopes) of a regression equation that models the relationship between the input and output variables. First use the **StandardScaler()** in sklearn to standardize the input data before you run your algorithm or fit your model.

1. Implement batch gradient descent from scratch to find the regression parameters as in the pseudo code below:

**= 0**

**= 0**

**Initialize all the b’s up to , where k = number of features**

**= 0.01**

**iterations = 1000**

**= np.array(x1\_values) # a vector**

**Initialize all the x’s up to**

**= np.array(x2\_values) # a vector**

**y = np.array(y\_values) # a vector**

**For i in range(iterations):**

**predicted\_y = + + # a vector**

**errors = y – predicted\_y # a vector**

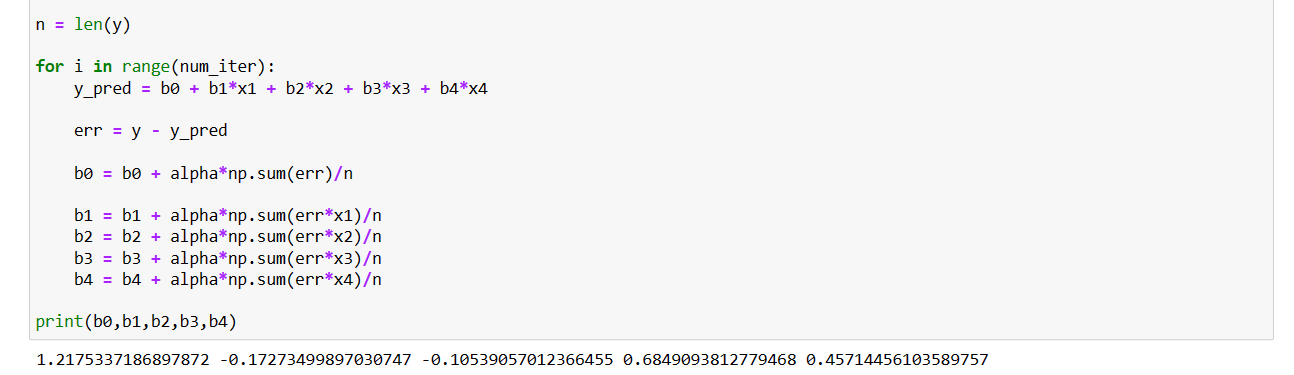
**# for all the j=1,…,k**

**print(,…,)# in this case, k=2**

1. Implement gradient descent using the

If you like, you can vectorize the implementation instead of having a line of code for each x or b. That mean, you can a create a vector of b’s and a feature matrix, X but you don’t have to. You can still implement it as described in the code above.





1. Use the scikit-learn package to implement the stochastic gradient descent to find the parameters of the regression equation. See the documentation here: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html#sklearn.linear_model.SGDRegressor>



Are you results in a) and b) similar? YES

**Question 2**

You will implement a multiple regression from a statistical perspective to explore the relationship between the input variables and the output variable. You will use the health data again.

1. Use the statsmodels package (<https://www.statsmodels.org/stable/regression.html#examples>) to implement a multiple linear regression using the entire health data to get a more detailed regression summary.

Below is some pseudo code for implementation, see the documentation for details.

**import** **numpy** **as** **np**

**import** **statsmodels.api** **as** **sm**

feature\_matrix = sm.add\_constant(X\_data, prepend=**False**)

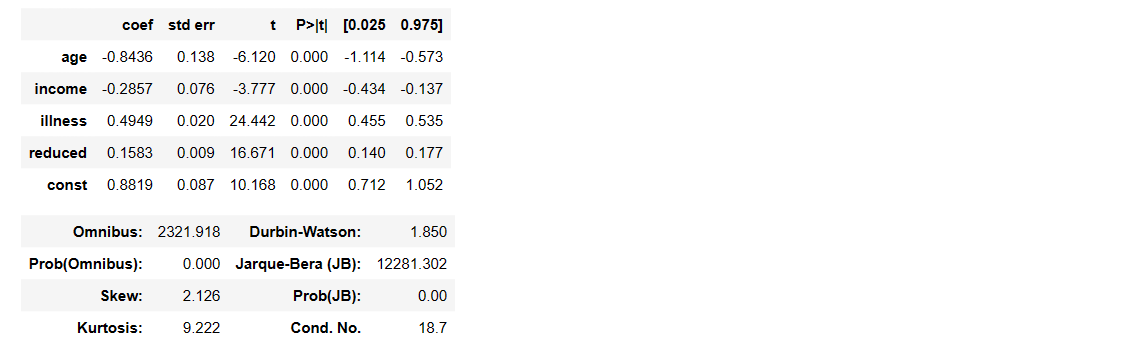
# Fit and summarize OLS model

model = sm.OLS(y\_data, feature\_matrix)

res = model.fit()

print(res.summary())





1. Does the model fit the data overall? That is, do all the x’s overall explain a good amount of variance in health outcome? Use information from your results to support your answer.

No, the R-squared value is .179, so the x's overall are only explains ~18% of the variance in health outcome.

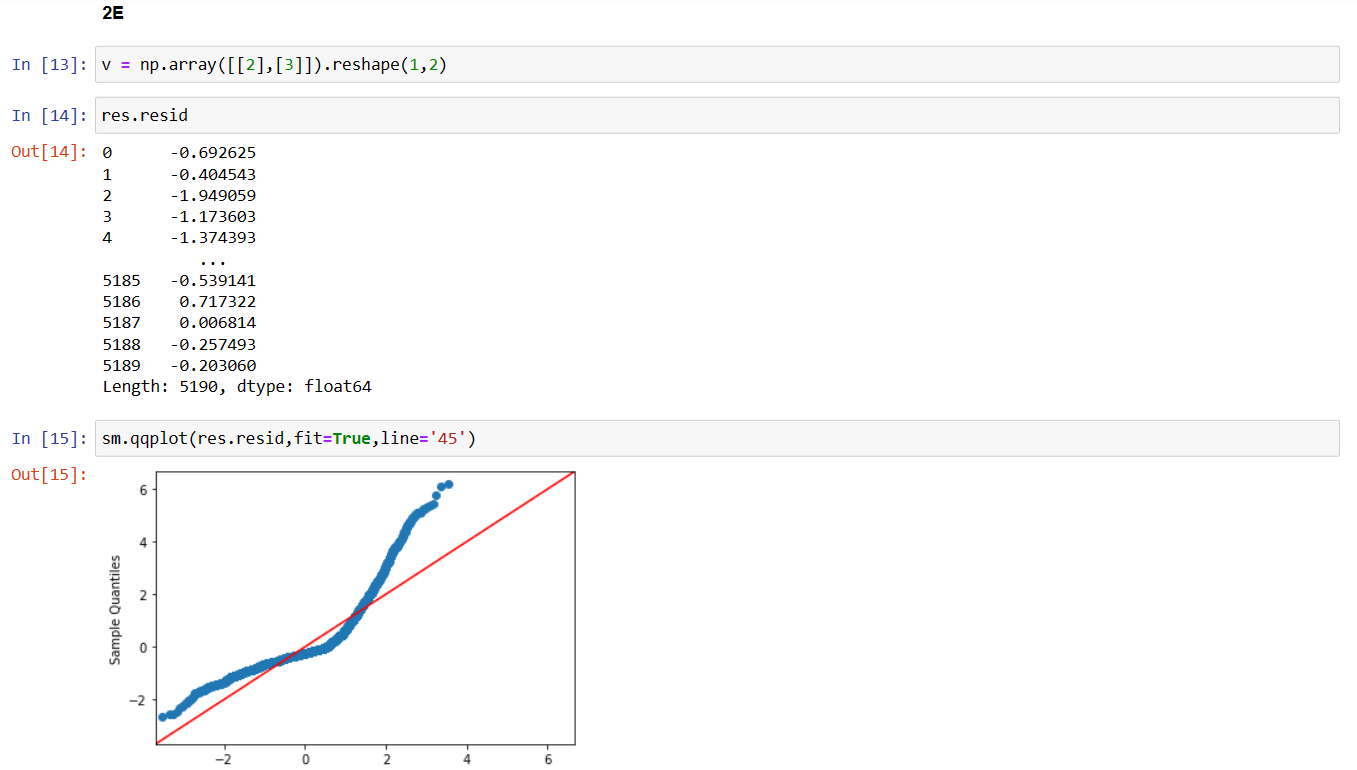
1. Do **age, income, illness** and **reduced** variables individually significantly predict health score? Use information from your summary of the regression results to support your answer.

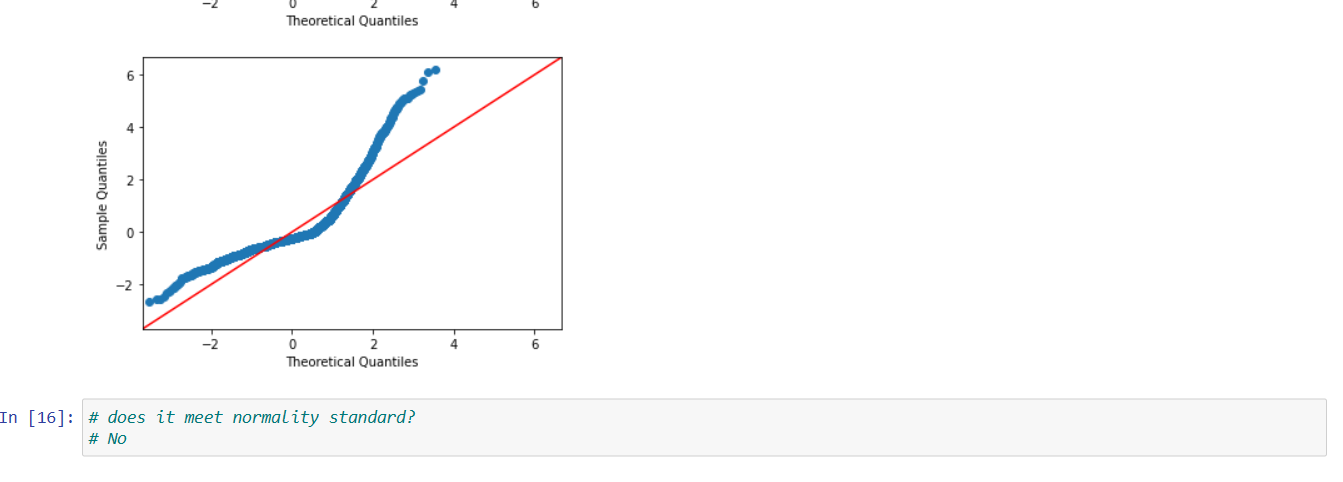
No, based on the coefficients of these predictors, they are not significant predictors

1. Which variable is the best predictor of health score? Why?

The illness variable is the best predictor of health score because it has the highest coefficient value of all the variables (0.4949)

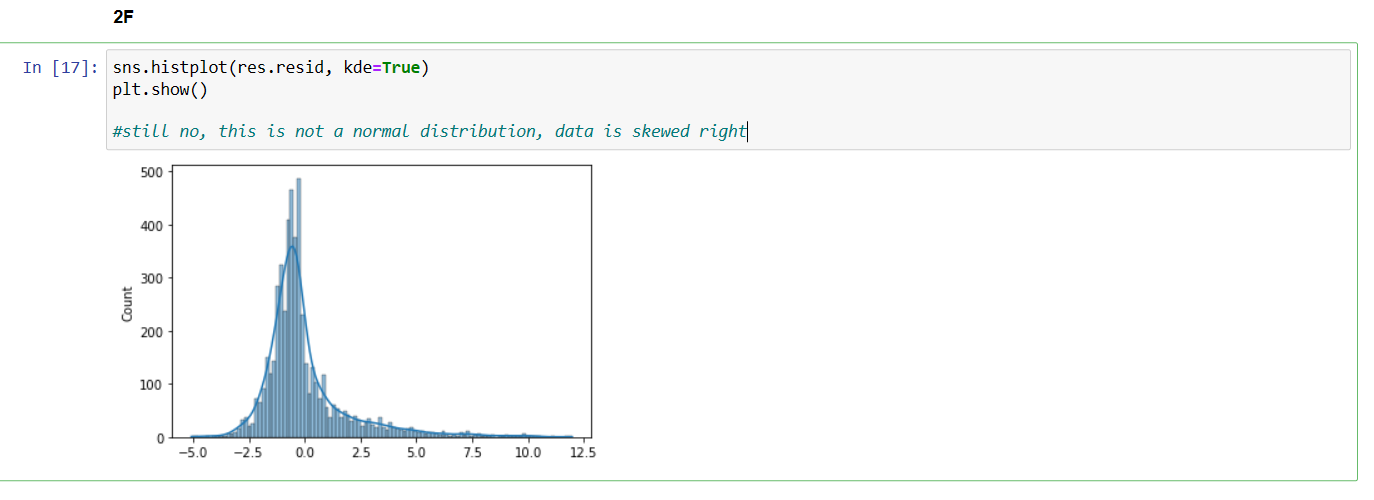
1. Extract the residuals of the model and test the assumptions normality and homogeneity of variance for the regression model. Are the assumptions met? Use **reg.resid** syntax to extract the residuals.



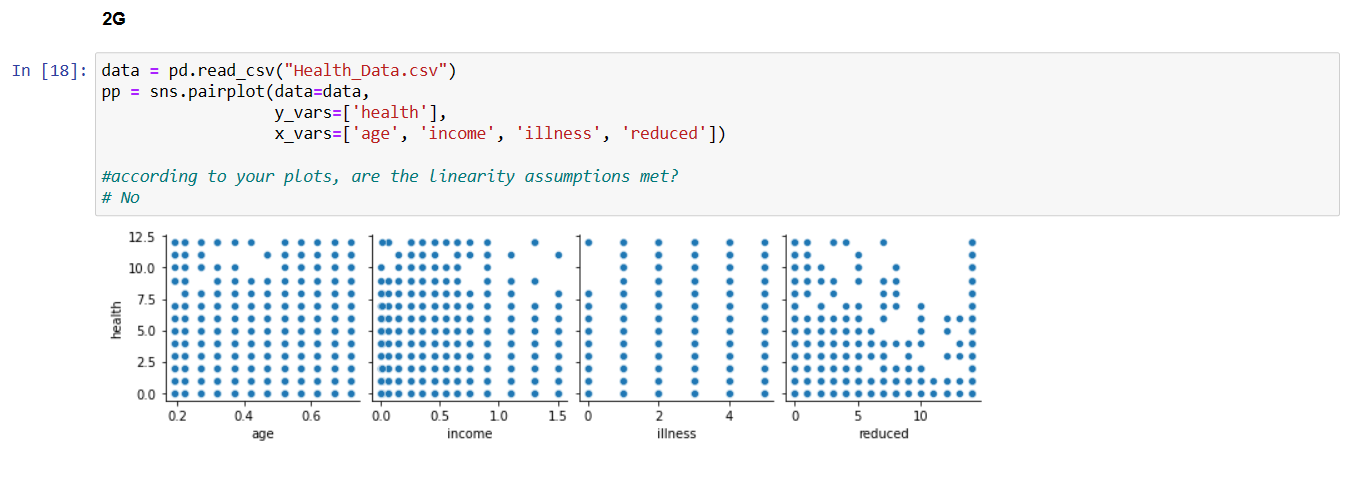


Not normal or homogenous based on plots.

1. To further verify the normality assumption, create a histogram for the residuals. Does the histogram look like a normal distribution?



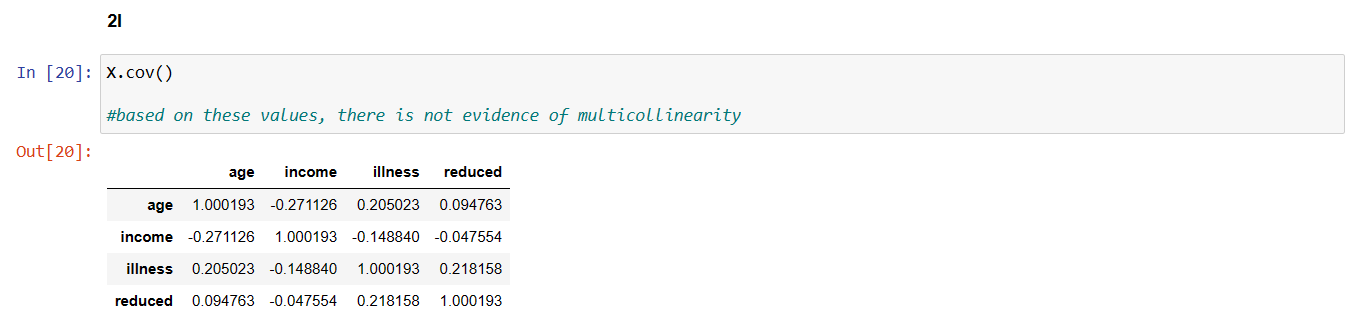
1. Test the linearity assumption by creating separate scatter plots for each input variable versus output variable. According to your plots, are the linearity assumptions met?



1. Extract the predicted output values from the model, then create a scatter plot of the predicted output versus actual output (from the data provided). This plot is also used to evaluate the accuracy of the model. According to the plot, is your model good for predicting health scores? Use the **reg.predict()** method to extract the predicted values.



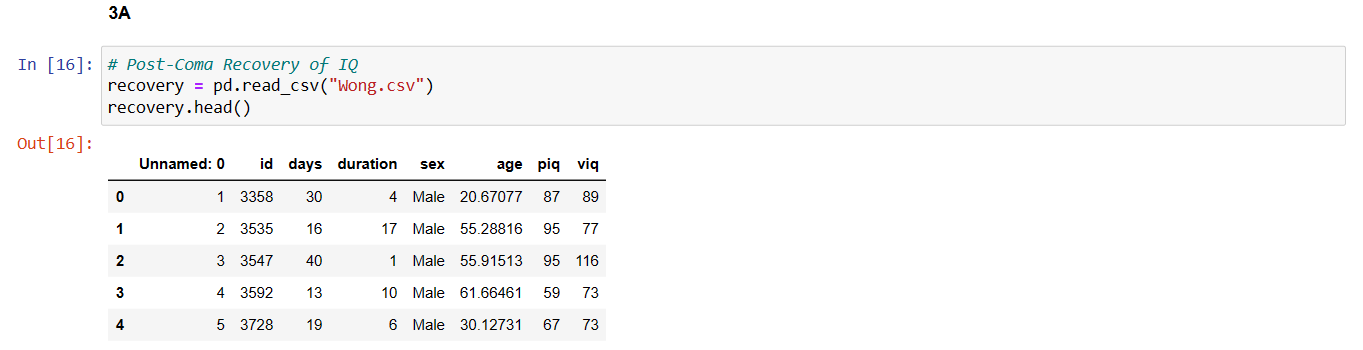
1. Check for multi-collinearity by generating a covariance matrix for the input data . You can use the .cov() method of pandas (<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.cov.html>). Do you think there is multicollinearity among the input variables?

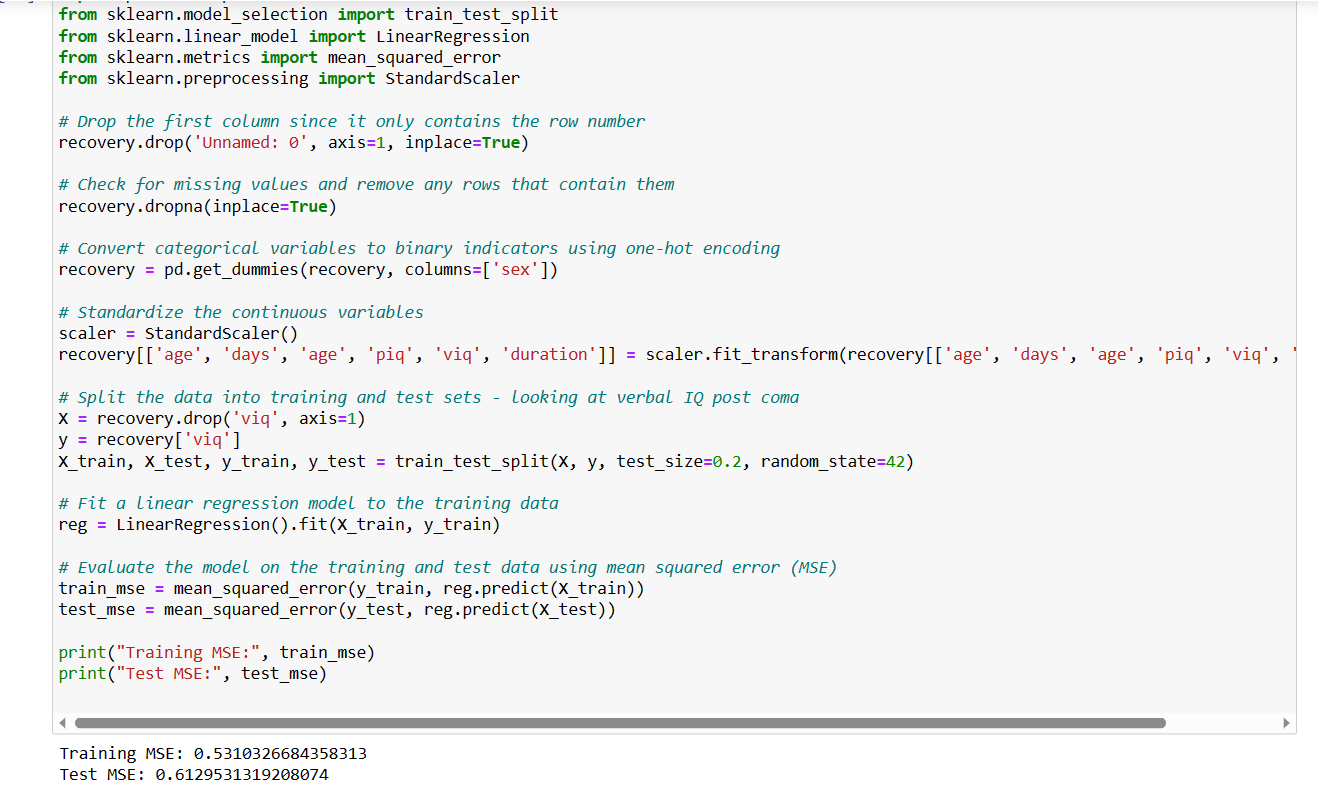


Question 3:

Find your own dataset from an online source with at least 3 input variables. Here is a suggested source of data but you don’t have to use it: <https://vincentarelbundock.github.io/Rdatasets/articles/data.html>. There should be one output variable of interest in the data. All the data used for analysis should be continuous. Clean the data as you find necessary, standardize the data and split it into training and test data using an appropriate split ratio.

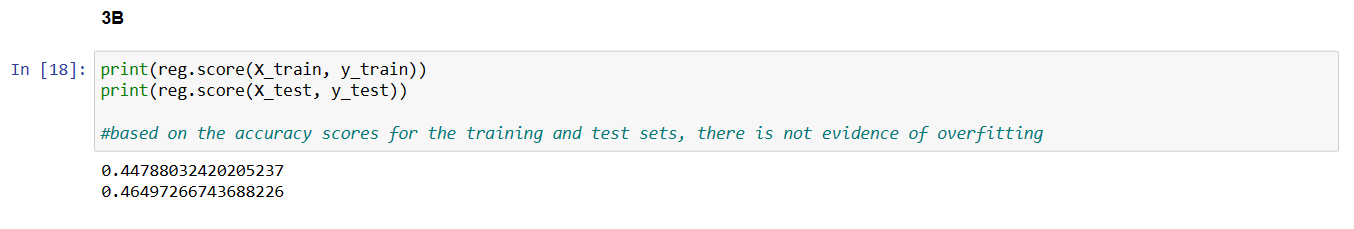
1. Construct a linear regression model using ordinary least squares method by applying the **.LinearRegression()** constructor in sklearn and find the training and test accuracy of this model using mean square error (mse). <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>



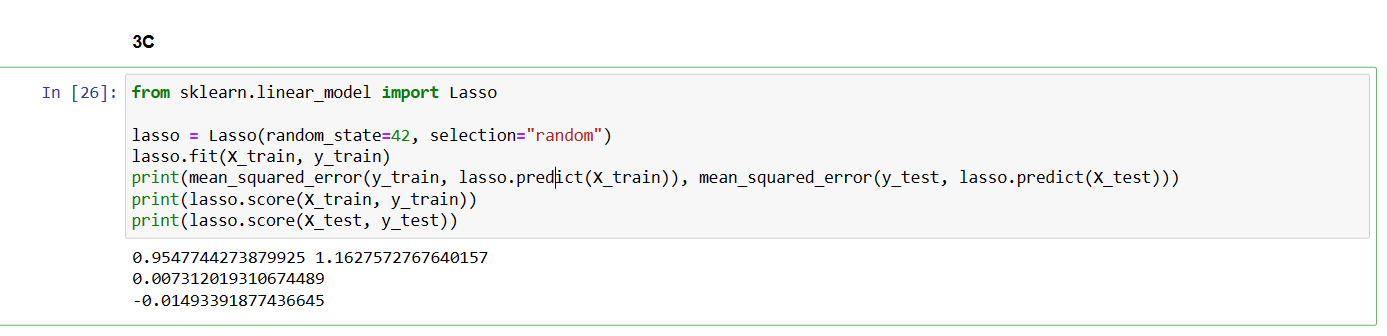


1. Check for overfitting. Is there overfitting? Support your answer with some results you generated.

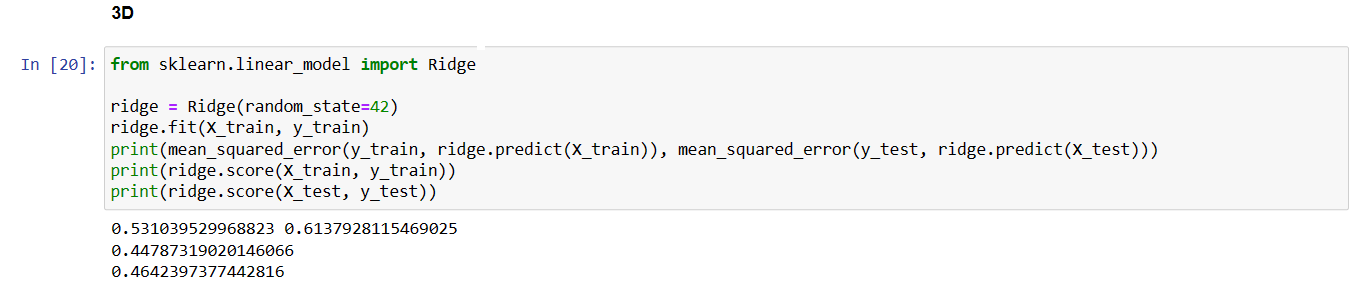
NO



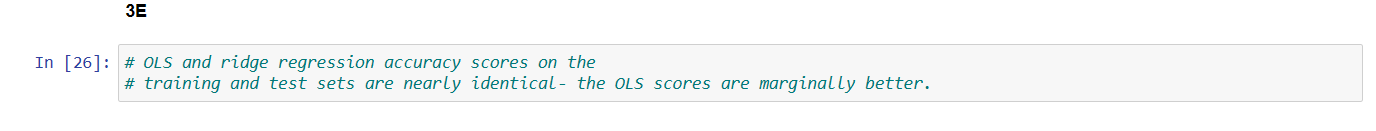
1. Fit a lasso regression on the data and check the training and test accuracy of the model using mse. Use the default alpha or penalty constant. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html>



1. Fit a ridge regression on the data and check the training and test accuracy of the model. Use the default alpha or penalty constant. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html>



1. Which model is better, OLS, Lasso, or Ridge regression?



1. Tune the alpha hyperparameters of the lasso and ridge regression using any tuning technique of your choice? What is the best alpha value for the lasso regression and what is the best alpha value for the ridge regression?

I performed a gridsearch for both models and found that lasso performs best with a alpha of 0.06 resulting in a train accuracy of 0.43308893503981005 and test accuracy of 0.4347267750481223. The results for the ridge regression was 0.44787319020146066 0.4642397377442816 with a alpha of 1.0 for the respective train and test sets. Please see my sceenshot below if you are interested in the code.

