Homework 2: This HW is based on the code for Multiple Variable Linear Regression

Instructions:

Place the answer to your code only in the area specified. Also, make sure to run all your code, meaning, press >> to "Restart Kernel and Run All Cells". This should plot all outputs including your answers to homework questions. After this, go to file (top left) and select "Print". Save your file as a PDF and upload the PDF to Canvas.

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

%matplotlib inline is a magic function that makes plots appear next to code and be stored in notebook: https://stackoverflow.com/questions/43027980/purpose-of-matplotlib-inline

Diabetes Data Example

To illustrate the concepts, we load the well-known diabetes data set. This dataset is included in the `sklearn.datasets` module and can be loaded as follows.

```
from sklearn import datasets, linear_model

# Load the diabetes dataset
diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target
```

We can print a description of the data as follows:

```
print(diabetes.DESCR)
```

```
    .. _diabetes_dataset:

    Diabetes dataset
    ----------------

    Ten baseline variables, age, sex, body mass index, average blood
    pressure, and six blood serum measurements were obtained for each of n =
    442 diabetes patients, as well as the response of interest, a
    quantitative measure of disease progression one year after baseline.

    **Data Set Characteristics:**

      :Number of Instances: 442

      :Number of Attributes: First 10 columns are numeric predictive values

      :Target: Column 11 is a quantitative measure of disease progression one year after baseline

      :Attribute Information:
          - age     age in years
          - sex
          - bmi     body mass index
          - bp      average blood pressure
          - s1      tc, total serum cholesterol
          - s2      ldl, low-density lipoproteins
          - s3      hdl, high-density lipoproteins
          - s4      tch, total cholesterol / HDL
          - s5      ltg, possibly log of serum triglycerides level
          - s6      glu, blood sugar level

    Note: Each of these 10 feature variables have been mean centered and scaled by the standard deviation times `n_samples` (i.e.

    Source URL:
    https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

    For more information see:
    Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (wi
    (https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```

The target values are stored in the vector `y`. The attributes for the diabetes data are stored in a data matrix, `X`. The size is is number of samples (442) x number of attributes (10).

```
nsamp, natt = X.shape
print("num samples={0:d}  num attributes={1:d}".format(nsamp,natt))
```

```
    num samples=442  num attributes=10
```

In the code above, we use the fromat method to help with output formatting. You use {} to indicate where the output would be substituted and you provide the variable to be used inside the format method, see more: https://docs.python.org/3/tutorial/inputoutput.html

Question 1:

Print the ages of the first five subjects?

```
import pandas as pd
dtx = pd.DataFrame(X, columns=diabetes.feature_names)
dty = pd.DataFrame(y)
print(dtx.age[:5])
```

```
    0    0.038076
    1   -0.001882
    2    0.085299
    3   -0.089063
    4    0.005383
    Name: age, dtype: float64
```

Question 2:

Print the attributes S1-S3 for subjects 10-15

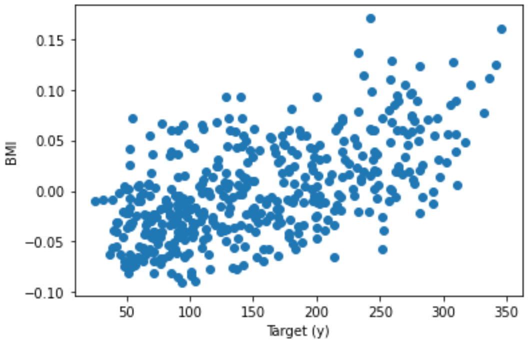
```
print(dtx.s1[9:15])
print(dtx.s2[9:15])
print(dtx.s3[9:15])
```

```
    9    -0.012577
    10   -0.103389
    11   -0.007073
    12   -0.004321
    13   -0.004321
    14    0.017694
    Name: s1, dtype: float64
    9    -0.034508
    10   -0.090561
    11    0.045972
    12   -0.009769
    13   -0.015719
    14   -0.000061
    Name: s2, dtype: float64
    9    -0.024993
    10   -0.013948
    11   -0.065491
    12    0.044958
    13   -0.002903
    14    0.081775
    Name: s3, dtype: float64
```

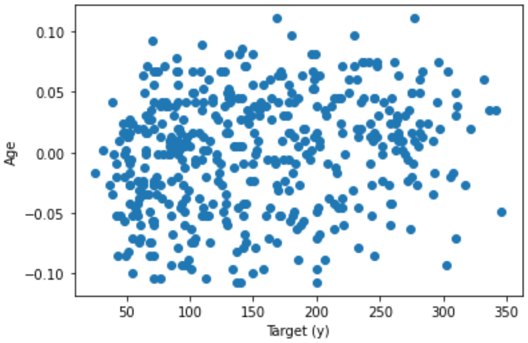
Question 3:

Create a scatter plot of the target variable, `y` vs. the BMI. Does there seem to be a relation? What about `y` vs. the age? Which is a better predictor?

```
plt.scatter(dty,dtx.bmi)
plt.xlabel('Target (y)')
plt.ylabel('BMI')
plt.show()
```



```
plt.scatter(dty,dtx.age)
plt.xlabel('Target (y)')
plt.ylabel('Age')
plt.show()
```



From the scatter plot of y vs. BMI, the relationship between the two variables, with a generally positive correlation (as BMI increases, target y also increases).

In the scatter plot of y vs. age, the relationship does not seem to be as strong, with more scatter and less of a clear relationship between the two variables.

Therefore,the above scatter plots show that BMI is a better predictor of target y than age.

Question 4:

You are given target values `y` and features `x1` and `x2` below. Fit the model on the first 4 data points and test the model on the fifth data point. You may want to use the following steps

- Construct the training training data `X_tr,y_tr`
- Create a regression object `regr = linear_model.LinearRegression()`
- Fit the model with the `regr.fit()` method
- Predict the value on the test value with the `regr.predict()`

```
x1 = np.array([0,1,3,5,4])
x2 = np.array([0,0.7, 4.3, 15.1, 13.2])
y = np.array([-2, -0.9, 1.5, 18, 13])

# Step 1: Construct the training data
X_tr = np.column_stack((x1[:4], x2[:4]))
y_tr = y[:4]

# Step 2: Create a regression object
regr = linear_model.LinearRegression()

# Step 3: Fit the model
regr.fit(X_tr, y_tr)

# Step 4: Predict the value
x_test = np.array([x1[4], x2[4]]).reshape(1,-1)
```

```
y_pred = regr.predict(x_test)

print("Predicted value: ", y_pred[0])
```

```
    Predicted value:  15.981708278580818
```

Question 5:

Describe the 1SE rule in cross validation and how the model order is selected based on the value of fitness score, i.e., whether a higher or lower firness score is desired and how the model order is determined.

The 1SE (One Standard Error) rule is a method to determine the optimal model complexity in cross-validation. It states that the optimal model is the one with the minimum number of parameters that is within one standard error of the best model.

The fitness score is a measure of how well the model fits the data. A higher fitness score indicates a better fit, and a lower fitness score indicates a poor fit. In cross-validation, the fitness score is usually calculated as the mean squared error (MSE) between the predicted and actual target values.

To determine the model order using the 1SE rule, the MSE is calculated for different model complexities, and a plot of MSE versus the number of parameters is created. The model order is then chosen as the one that is closest to the best model, but has the smallest number of parameters. The best model is defined as the one with the lowest MSE. The 1SE rule ensures that the chosen model is not over-fitting the data, i.e., it has a good balance between model complexity and performance.

✓ 0s completed at 11:32 AM ● ×