Lab # 5: Linear Regression

5.1 Objectives

32. Understand the Linear Regression
33. Apply the Linear Regression through python.

5.2 Pre-Lab

5.2.1.1 Linear Regression

Linear regression is a basic predictive analytics technique that uses historical data topredict an output variable. It is popular for predictive modeling because it is easilyunderstood and can be explained using plain English.

The basic idea is that if we can fit a linear regression model to observed data, wecan then use the model to predict any future values. For example, let us assume thatwe have found from historical data that the price (P) of a house is linearly dependentupon its size (S)—in fact, we found that a house's price is exactly 90 times its size.

The equation will look like this: P = 90*S.

With this model, we can then predict the cost of any house. If we have ahouse that is 1,500 square feet, we can calculate its price to be: P = 90*1500= $135,000

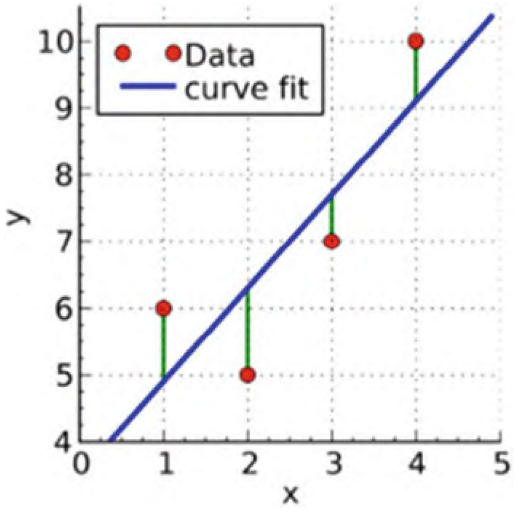
There are two kinds of variables in a linear regression model:

34. The input or predictor variable is the variable(s) that help predict the value of theoutput variable. It is commonly referred to as X.
35. The output variable is the variable that we want to predict. It is commonlyreferred to as Y.

To estimate Y using linear regression, we assume the equation: $Ye = \alpha + \beta X$where Ye is the estimated or predicted value of Y based on our linear equation. Ourgoal is to find statistically significant values of the parameters α and β that minimizethe difference between Y and Ye. If we are able to determine the optimum values ofthese two parameters, then we will have the line of best fit that we can use to predictthe values of Y, given the value of X. So, how do we estimate α and β? We can usea method called ordinary least squares.

The objective of the least squares method is to find values of α and β thatminimize the sum of the squared difference between Y and Ye.

Here, we notice that when E increases by 1, our Y increases by 0.175399. Also,when C increases by 1, our Y falls by 0.044160.



In this experiment and onwards following data set will be utilized:

Dataset: "Alumni Giving Regression (Edited).csv"

Following is the link to download dataset:

https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regression%20%28Edited%29.csv

5.3 In-Lab

In-Lab Task 1

Import the following libraries for Linear Regression. All of these should be properly working.

```
# Importing libraries needed
# Note that keras is generally used for deep learning as well
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
import pandas as pd
import csv

import matplotlib.pyplot as plt
```

In-Lab Task 2

Import Data Set

```
# ========================================================================
# Read Data and fix seed
# ========================================================================
# fix random seed for reproducibility
np.random.seed(7)
df = pd.read_csv("Alumni Giving Regression (Edited).csv", delimiter="," )
dd_df_1=df.head()
```

In-Lab Task 3

Write code to visualize the Data Pattern and Description of Data

Data Pattern

	A	B	C	D	E	F
0	24	0.42	0.16	0.59	0.81	0.08
1	19	0.49	0.04	0.37	0.69	0.11
2	18	0.24	0.17	0.66	0.87	0.31
3	8	0.74	0.00	0.81	0.88	0.11
4	8	0.95	0.00	0.86	0.92	0.28

Description of Data

	A	B	C	D	E	F
count	123.000000	123.000000	123.000000	123.000000	123.000000	123.000000
mean	17.772358	0.403659	0.136260	0.645203	0.841138	0.141789
std	4.517385	0.133897	0.060101	0.169794	0.083942	0.080674
min	6.000000	0.140000	0.000000	0.260000	0.580000	0.020000
25%	16.000000	0.320000	0.095000	0.505000	0.780000	0.080000
50%	18.000000	0.380000	0.130000	0.640000	0.840000	0.130000
75%	20.000000	0.460000	0.180000	0.785000	0.910000	0.170000
max	31.000000	0.950000	0.310000	0.960000	0.980000	0.410000

In-Lab Task 4

Compute the correlation of data and understand the relationship between variables.

The term “correlation” refers to a mutual relationship or association betweenquantities (numerical number). In almost any business, it is very helpful to expressone quantity in terms of its relationship with others. We are concerned aboutthis because business plans and departments are not isolated! For example, salesmight increase when the marketing department spends more on advertisements,or a customer’s average purchase amount on an online site may depend on hisor her characteristics. Often, correlation is the first step to understanding theserelationships and subsequently building better business and statistical models.

Compute the correlation of the data set using the following code and understand the meaning of correlation results.

```
# ================================================================================
# Commpute Correlation
# ================================================================================
corr=df.corr(method ='pearson')
corr
print (corr)
```

"D" and "E" have a strong correlation of 0.93, which means thatwhen D moves in the positive direction E is likely to move in that direction too.Here, notice that the correlation of `A` and `A` is 1. Of course, `A` would be perfectlycorrelated with `A`.

	A	B	C	D	E	F
A	1.000000	-0.691900	0.414978	-0.604574	-0.521985	-0.549244
B	-0.691900	1.000000	-0.581516	0.487248	0.376735	0.540427
C	0.414978	-0.581516	1.000000	0.017023	0.055766	-0.175102
D	-0.604574	0.487248	0.017023	1.000000	0.934396	0.681660
E	-0.521985	0.376735	0.055766	0.934396	1.000000	0.647625
F	-0.549244	0.540427	-0.175102	0.681660	0.647625	1.000000

In-Lab Task 5

Splitting of Data into Train and Test Segments

In general, we would need to test our model. `train_test_split` is a function in `Sklearn` model selection for splitting data arrays into two subsets for training data and for testing data. With this function, you do not need to divide the dataset manually. You can use from the function `train_test_split` using the following code `sklearn.model_selection import train_test_split`.

By default, Sklearn train_test_split will make random partitions for the two subsets. However, you can also specify a random state for the operation. Here, takes note that we will need to pass in the `X` and `Y` to the function. `X` refers to the `features` while `Y` refers to the `target` of the dataset.

```
# ================================================================================
# Train/ Test Split
Y_POSITION = 5

# ================================================================================
# Here, Y_POSITION is set to 5, indicating that the target variable is located
# in the 5th column of the DataFrame. model_1_features is then created as a l
# ist containing the indices [0, 1, 2, 3, 4], representing the columns from 0
# to 4 (excluding 5).
# ================================================================================
```

```
# ================================================================================

model_1_features = [i for i in range(0,Y_POSITION)]

# ================================================================================
# This code extracts the features (X) and the target variable (Y) from the
# DataFrame (df). The features are obtained by selecting all rows (:) and the
# columns specified in model_1_features. The target variable (Y) is obtained by
# selecting all rows and the column specified by Y_POSITION.
# ================================================================================

# x refers to as features of the data
X = df.iloc[:,model_1_features]

# target data
Y = df.iloc[:,Y_POSITION]
# create model
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20,
                                                    random_state=2020)

# ================================================================================
# This line uses the train_test_split function from scikit-learn to split the
# data into training and testing sets. The arguments are:

# X: Features.
# Y: Target variable.
# test_size: The proportion of the dataset to include in the test split (here,
# 20%).
# random_state: Seed for the random number generator to ensure reproducibility.
# The function returns four sets:

# X_train: Training features.
# X_test: Testing features.
# y_train: Training target variable.
# y_test: Testing target variable.
#
```

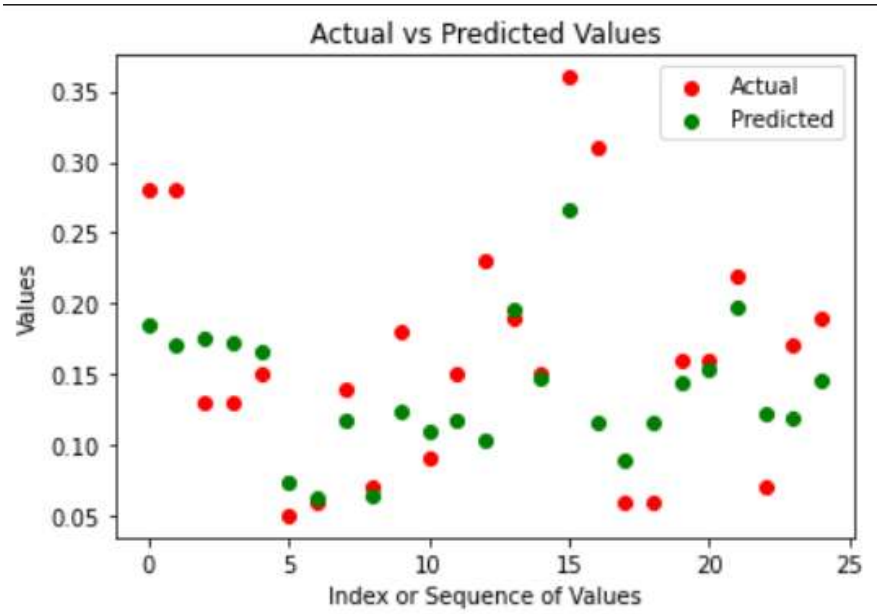
In-Lab Task 4

Perform Linear Regression and display result.

```
# ================================================================================
# #Model 1 : linear regression

#  This line creates an instance of the linear regression model using
# scikit-learn's LinearRegression class.
model1 = linear_model.LinearRegression()

# This line trains the linear regression model (model1) using the training
# features (X_train) and the corresponding target variable (y_train).
model1.fit(X_train, y_train)

# This line uses the trained model to make predictions on the training
# set (X_train) and stores the predicted values in y_pred_train1.
y_pred_train1 = model1.predict(X_train)
```

```
# The Root Mean Squared Error (RMSE) is a measure of the average magnitude of
# the errors between predicted and actual values. This line calculates and
# prints the RMSE for the training set.
print("Regression")
print("================================")
RMSE_train1 = mean_squared_error(y_train,y_pred_train1)

# This line uses the trained model to make predictions on the testing
# set (X_test) and stores the predicted values in y_pred1.
print("Regression Train set: RMSE {}".format(RMSE_train1))
print("================================")
y_pred1 = model1.predict(X_test)
RMSE_test1 = mean_squared_error(y_test,y_pred1)

print("Regression Test set: RMSE {}".format(RMSE_test1))
print("================================")
coef_dict = {}
for coef, feat in zip(model1.coef_,model_1_features):
    coef_dict[df.columns[feat]] = coef
print(coef_dict)
```

```
x_values = np.arange(len(y_test))
# Scatter plot of actual values (y_test) in red
plt.scatter(x_values, y_test, color='red', label='Actual')
# Scatter plot of predicted values (y_pred) in green
plt.scatter(x_values, y_pred1, color='green', label='Predicted')
plt.xlabel('Index or Sequence of Values')
plt.ylabel('Values')
plt.title('Actual vs Predicted Values')
plt.legend()
plt.show()
```



Use the results of the model coefficients to predict a value on paper and verify using code.

5.4 Post Lab

Post-Lab Task

Based on the linear regression analysis you performed, what steps would you take to improve the model's predictive accuracy, and how would you validate the model's performance on unseen data?

Select a data set from freely available datasets on internet and provide implementation of your suggested improvements.