Lab Assignment 05

The objective of this lab assignment is to build and evaluate regression models to predict total charge given information from customers of a telephone company (`data_lab_05.csv`).

Instructions:

Complete each task and question by filling in the blanks (`...`) with one or more lines of code or text. Each task and question is worth **0.5 points** (out of **10 points**).

Submission:

This assignment is due **Monday, November 18, at 11:59PM (Central Time)**.

This assignment must be submitted on Gradescope as a **PDF file** containing the completed code for each task and the corresponding output. Late submissions will be accepted within **0-12** hours after the deadline with a **0.5-point (5%) penalty** and within **12-24** hours after the deadline with a **2-point (20%) penalty**. No late submissions will be accepted more than 24 hours after the deadline.

This assignment is individual. Offering or receiving any kind of unauthorized or unacknowledged assistance is a violation of the University's academic integrity policies, will result in a grade of zero for the assignment, and will be subject to disciplinary action.

Part 1: Simple Linear Regression

In [24]:
```
# Load libraries
import pandas as pd
import numpy
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import linear_model
```

In [25]:
```
# Load dataset and display the first five rows
data = pd.read_csv('data_lab_05.csv')
data.head()
```

Out[25]:

	Account length	International plan	Voice mail plan	Number voice mail messages	Total day minutes	Total day calls	Total eve minutes	Total eve calls	Total night minutes	Total night calls	Total intl minutes	Total intl calls	Customer service calls	Total charge
0	128	0	1	25	265.1	110	197.4	99	244.7	91	10.0	3	1	75.56
1	107	0	1	26	161.6	123	195.5	103	254.4	103	13.7	3	1	59.24
2	137	0	0	0	243.4	114	121.2	110	162.6	104	12.2	5	0	62.29
3	84	1	0	0	299.4	71	61.9	88	196.9	89	6.6	7	2	66.80
4	75	1	0	0	166.7	113	148.3	122	186.9	121	10.1	3	3	52.09

In [26]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 14 columns):
Account length                3333 non-null int64
International plan            3333 non-null int64
Voice mail plan               3333 non-null int64
Number voice mail messages    3333 non-null int64
Total day minutes             3333 non-null float64
Total day calls               3333 non-null int64
Total eve minutes             3333 non-null float64
Total eve calls               3333 non-null int64
Total night minutes           3333 non-null float64
Total night calls             3333 non-null int64
Total intl minutes            3333 non-null float64
Total intl calls              3333 non-null int64
Customer service calls        3333 non-null int64
Total charge                  3333 non-null float64
dtypes: float64(5), int64(9)
memory usage: 364.6 KB
```

Task 01 (of 15): Partition the dataset into training set and test set. *Hint:* Use 75% of the data for training and 25% for testing and set parameter random_state to 0.

In [27]:
```
x_train, x_test, y_train, y_test = train_test_split(data.iloc[:,:13],data.iloc[:,13:],test_size = 0.25, rando
m_state=0)
```

In [28]:
```
y_test.head()
```

Out[28]:

	Total charge
405	70.06
118	43.41
710	66.70
499	55.99
2594	88.97

Task 02 (of 15): Standardize the training set and test set. *Hint:* Compute the mean and standard deviation using only the training set and then apply this transformation on the training set and test set.

In [29]:
```
scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Task 03 (of 15): Build a simple linear model to predict 'Total charge' with 'Total day minutes' as the predictor and print the coefficient of the model. *Hint:* `X` must be a 2D array.

In [30]:
```
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled[:,[4]].reshape(-1,1), y = y_train)
print(fitted_model.coef_)
```

```
[[9.30234225]]
```

In [31]:
```
from sklearn import linear_model
import statsmodels.api as sm
```

In [32]:
```
# Build model with 1 predictor (using statsmodels)
X = x_train_scaled[:,[4]]
Y = y_train
X = sm.add_constant(X)
results = sm.OLS(Y,X).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Total charge   R-squared:                       0.782
Model:                            OLS   Adj. R-squared:                  0.782
Method:                 Least Squares   F-statistic:                     8949.
Date:                Mon, 18 Nov 2019   Prob (F-statistic):               0.00
Time:                        18:21:22   Log-Likelihood:                -7524.5
No. Observations:                2499   AIC:                         1.505e+04
Df Residuals:                    2497   BIC:                         1.506e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         59.5751      0.098    605.838      0.000      59.382      59.768
x1             9.3023      0.098     94.598      0.000       9.110       9.495
==============================================================================
Omnibus:                        1.504   Durbin-Watson:                   1.860
Prob(Omnibus):                  0.471   Jarque-Bera (JB):                1.552
Skew:                          -0.043   Prob(JB):                        0.460
Kurtosis:                       2.914   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Task 04 (of 15): Use the model to predict 'Total charge' for the test set. *Hint:* `X` must be a 2D array.

In [33]:
```
predicted = fitted_model.predict(X = x_test_scaled[:,[4]])
#len(predicted)
```

Task 05 (of 15): Compute the coefficient of determination (R squared) of the model over the test set. *Hint:* First compute the correlation coefficient between the predicted y-values and the observed y-values.

In [34]:
```
corr_coef = numpy.corrcoef(predicted, y_test.values.reshape(-1,1),rowvar = False)[1,0]
R_squared = corr_coef**2
print(corr_coef)
print(R_squared)
```

```
0.8864844695326588
0.7858547147225995
```

Question 01 (of 05): What can you conclude about the performance of the model?

Answer: From the value of R2 = 0.786, we can conclude that nearly 78.6 percent of the variance in the "Total Charge" is explained by the model. And the model is good but not that good.

Part 2: Multiple Linear Regression

Task 06 (of 15): Build a multiple linear model to predict 'Total charge' with 'Total day minutes', 'Total eve minutes', 'Total night minutes', and 'Total intl minutes' as predictors and print the coefficients of the model.

In [35]:
```
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled[:,[4,6,8,10]],
                         y = y_train)
print(fitted_model.coef_)
```

```
[[9.23422081 4.34962707 2.2813772  0.75805126]]
```

In [36]:
```
# Build model with 1 predictor (using statsmodels)
X = x_train_scaled[:,[4,6,8,10]]
Y = y_train
X = sm.add_constant(X)
results = sm.OLS(Y,X).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Total charge   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 2.129e+09
Date:                Mon, 18 Nov 2019   Prob (F-statistic):               0.00
Time:                        18:21:27   Log-Likelihood:                 9370.2
No. Observations:                2499   AIC:                        -1.873e+04
Df Residuals:                    2494   BIC:                        -1.870e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         59.5751      0.000   5.23e+05      0.000      59.575      59.575
x1             9.2342      0.000    8.1e+04      0.000       9.234       9.234
x2             4.3496      0.000   3.81e+04      0.000       4.349       4.350
x3             2.2814      0.000      2e+04      0.000       2.281       2.282
x4             0.7581      0.000   6646.826      0.000       0.758       0.758
==============================================================================
Omnibus:                       18.599   Durbin-Watson:                   2.056
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               12.919
Skew:                           0.021   Prob(JB):                      0.00157
Kurtosis:                       2.650   Cond. No.                         1.04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Task 07 (of 15): Use the model to predict 'Total charge' for the test set.

In [16]:
```
predicted = fitted_model.predict(x_test_scaled[:,[4,6,8,10]])
```

Task 08 (of 15): Compute the coefficient of determination (R squared) of the model over the test set. *Hint:* First compute the correlation coefficient between the predicted y-values and the observed y-values.

In [17]:
```
corr_coef = numpy.corrcoef(predicted, y_test,rowvar = False)[1,0]
R_squared = corr_coef**2
print(corr_coef)
print(R_squared)
```

```
0.9999998537077304
0.9999997074154822
```

Question 02 (of 05): What can you conclude about the performance of the model?

Answer: As almost 100 percent of the variance of the "Total Charge" is explained by the model, we can conclude that its a perfect model.

Task 09 (of 15): Build a multiple linear model to predict 'Total charge' with all features as predictors and print the coefficients of the model.

In [38]:
```
model = linear_model.LinearRegression()
fitted_model = model.fit(X = x_train_scaled,y = y_train)
print(fitted_model.coef_)
```

```
[[ 1.86701207e-04 -5.41336738e-05  4.87160937e-04 -4.61769986e-04
   9.23422162e+00  2.04345920e-04  4.34963578e+00  8.54680666e-05
   2.28136883e+00  3.74275670e-05  7.58044203e-01  1.35159379e-04
  -2.43936617e-05]]
```

In [39]:

```
# Build model with 1 predictor (using statsmodels)
X = x_train_scaled
Y = y_train
X = sm.add_constant(X)
results = sm.OLS(Y,X).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Total charge   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 6.555e+08
Date:                Mon, 18 Nov 2019   Prob (F-statistic):               0.00
Time:                        18:24:05   Log-Likelihood:                 9375.4
No. Observations:                2499   AIC:                        -1.872e+04
Df Residuals:                    2485   BIC:                        -1.864e+04
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         59.5751      0.000   5.23e+05      0.000      59.575      59.575
x1             0.0002      0.000      1.633      0.103   -3.75e-05       0.000
x2         -5.413e-05      0.000     -0.473      0.636      -0.000       0.000
x3             0.0005      0.000      1.247      0.213      -0.000       0.001
x4            -0.0005      0.000     -1.182      0.237      -0.001       0.000
x5             9.2342      0.000   8.08e+04      0.000       9.234       9.234
x6             0.0002      0.000      1.788      0.074   -1.98e-05       0.000
x7             4.3496      0.000   3.81e+04      0.000       4.349       4.350
x8          8.547e-05      0.000      0.749      0.454      -0.000       0.000
x9             2.2814      0.000      2e+04      0.000       2.281       2.282
x10         3.743e-05      0.000      0.328      0.743      -0.000       0.000
x11            0.7580      0.000   6637.387      0.000       0.758       0.758
x12            0.0001      0.000      1.185      0.236   -8.85e-05       0.000
x13        -2.439e-05      0.000     -0.214      0.831      -0.000       0.000
==============================================================================
Omnibus:                       19.759   Durbin-Watson:                   2.059
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               13.585
Skew:                           0.024   Prob(JB):                      0.00112
Kurtosis:                       2.642   Cond. No.                         6.72
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Task 10 (of 15): Use the model to predict 'Total charge' for the test set.

In [19]:
```
predicted = fitted_model.predict(x_test_scaled)
```

Task 11 (of 15): Compute the coefficient of determination (R squared) of the model over the test set. *Hint:* First compute the correlation coefficient between the predicted y-values and the observed y-values.

In [20]:
```
corr_coef = numpy.corrcoef(predicted, y_test,rowvar = False)[1,0]
R_squared = corr_coef**2
print(corr_coef)
print(R_squared)
```

```
0.9999998516141111
0.9999997032282442
```

Question 03 (of 05): What can you conclude about the performance of the model?

Answer: Since R2 value is almost one, we can say that this is a very good model, but this model has many variables and is more complex.

Part 3: Regularization

Task 12 (of 15): Build a LASSO regression model to predict 'Total charge' with all features as predictors.

In [21]:
```
model = linear_model.Lasso(alpha = 1)
fitted_model = model.fit(X = x_train_scaled,y = y_train)
```

Task 13 (of 15): Print the coefficients of the model.

In [22]:
```
print(fitted_model.coef_)
```

```
[-0.          0.          0.          0.          8.24885419  0.
  3.3331111  -0.          1.25156405  0.          0.          0.
 -0.        ]
```

Task 14 (of 15): Use the model to predict 'Total charge' for the test set.

In [23]:
```
predicted = fitted_model.predict(x_test_scaled)
```

Task 15 (of 15): Compute the coefficient of determination (R squared) of the model over the test set. *Hint:* First compute the correlation coefficient between the predicted y-values and the observed y-values.

In [47]:
```
corr_coef = numpy.corrcoef(predicted, y_test,rowvar = False)[1,0]
R_squared = corr_coef**2
print(corr_coef)
print(R_squared)
```

```
0.9926941379651821
0.9854416515504361
```

Question 04 (of 05): What can you conclude about the coefficients and the performance of the model?

Answer: . . . LASSO regularization made some of the coefficients as zeroes and made the model less complex. This model is good but not that good as above models due to the decrease in the r and R2 values.

Question 05 (of 05): Based on all the results obtained, what are the most important variables to predict the total charge of a user? Justify your answer.

Answer: . . . Based on the above results, the variables: 'Total day minutes', 'Total eve minutes', 'Total night minutes', and 'Total intl minutes' are the most important ones to predict the total charge of a user. Because the model with these variables has the highest r and R2 value.

In [ ]:
```

```