Lab Assignment 03

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_03.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 **Sunday, October 04, 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. To save your Jupyter notebook as a PDF file, go to `File > Export Notebook As > HTML` or `File > Download As > HTML` , open the HTML file and print it as a PDF file. Additionally, this assignment has a single question on Gradescope and **all pages of the PDF file** must be assigned to this question. A **0.5-point (5%) penalty** will be applied to submissions that do not follow these guidelines. For more instructions on how to submit assignments on Gradescope, see this guide.

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: Data Preparation

In [35]:
```
# 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 [36]:
```
# Load dataset and display the first five rows
data = pd.read_csv('data_lab_03.csv')
data.head()
```

Out[36]:

	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

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

In [37]:
```
x_train, x_test, y_train, y_test = train_test_split(data[['Account length', 'International plan', 'Voic
e mail plan',
                                                            'Number voice mail messages', 'Total day minutes','Tota
l day calls',
                                                            'Total eve minutes', 'Total eve calls', 'Total night mi
nutes','Total night calls',
                                                            'Total intl minutes', 'Total intl calls', 'Customer ser
vice calls']],
                                                                data['Total charge'], test_size=0.25, random_state=
0)
```

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

In [38]:
```
scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Part 2: Simple Linear Regression

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 [39]:
```
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]
```

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

In [40]:
```
predicted = fitted_model.predict(X = x_test_scaled[:,[4]].reshape(-1, 1))
```

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 [41]:
```
corr_coef = numpy.corrcoef(predicted, y_test.values.reshape(-1,1), rowvar=False)[1, 0]
R_squared = corr_coef ** 2
print(R_squared)
```

```
0.7858547147225989
```

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

Answer: Based on R_squared result, there are 78.59% of data are being represent by model, so this model is well fit the data

Part 3: 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 [42]:
```
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]
```

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

In [43]:
```
predicted = fitted_model.predict(X = 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 [44]:
```
corr_coef = numpy.corrcoef(predicted, y_test.values.reshape(-1,1), rowvar=False)[1, 0]
R_squared = corr_coef ** 2
print(R_squared)
```

```
0.9999997074154813
```

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

Answer: Based on R_squared result, there are 99.99% of data are being represent by model, the result of multiple predictor is higher than single predictor, so this model is better fit the data

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 [45]:
```
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]
```

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

In [46]:
```
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 [47]:
```
corr_coef = numpy.corrcoef(predicted, y_test, rowvar=False)[1, 0]
R_squared = corr_coef ** 2
print(R_squared)
```

```
0.9999997032282442
```

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

Answer: Based on R_squared result, there are 99.99% of data are being represent by the model, so this model is almost perfect fit the data

Part 4: Regularization

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

In [48]:
```
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 [49]:
```
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 [50]:
```
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 [51]:
```
corr_coef = numpy.corrcoef(predicted, y_test, rowvar=False)[1, 0]
R_squared = corr_coef ** 2
print(R_squared)
```

```
0.985441651550435
```

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

Answer: LASSO regression model made some coefficients become 0, it tells which predictor are not related with model, so the model can generate a well fit prediction

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: The most important variables to predict the total charge of a user are 'Total charge' with 'Total day minutes', 'Total eve minutes', 'Total night minutes', and 'Total intl minutes'.

Because when we use 'Total day minutes' as predictor, there are only 78.59% of data are being represent by model. When we use 'Total charge' with 'Total day minutes', 'Total eve minutes', 'Total night minutes', and 'Total intl minutes' as predictors, there are only 99.99% of data are being represent by model, which is a perfect model for predict the data. Even though the all features as predictors are also generate a 99.99% model, but that model has too many variance and could potentially cause overfitting.