Module 6 Homework

Problem 3 Tutorial

3) The file model_assessment_data.csv contains the output of a classification model for 100 observations. The first column – P(Y) contains the probability that the model assigned to that observation that it contained the event. The second column indicates whether it actually contained the event.

a) Create a dataframe with the following lift curve data as columns

- Percentile: values from 5 to 100 by 5 (5, 10, 15, ..., 100)
- Model lift (value at the corresponding percentile)
- Best lift (value at the corresponding percentile)

In [51]:

```python
import pandas as pd
import numpy as np
```

Step 1: Read in model_assessment_data and sort by P(Y) descending

- Use df.sort_values() method
- Reset index using df.reset_index(drop = True) method

In [ ]:

Step 2: Create lift_data dataframe with three columns: "Percentile", "Model", and "Best"

- Multiple ways to creat a dataframe. I created a dictionary first and then created the dataframe from the dictionary.
- Populate the dictionary "Percentile" list with Percentile values using the list(range()) function
- Populate the dictionary "Model" and "Best" lists with the Python function [0.0] * 20. (Note - use 0.0, not 0 so that it creates real datatypes, not integers.

In [ ]:

Step 3: Calculate the value for index_increment (the number of ovservations in each 5% "bucket"

In [ ]:

Step 4: Create the "best model" series. A series of 20 integers each indicating how many 1s are in each 5% "bucket"

In [ ]:

Step 5: Calculate "random_model_per_bin" value with the number of 1s in a randomly selected 5% bucket.

In [ ]:

Step 6: Populate the "Model" and "Best" columns in your lift_data dataframe.

- Hint: use a for loop

In [ ]:

b) Create a lift curve graphic using this data

- Hint: this is actually easier with matplotlib than with seaborne.

In [ ]:

c) Calculate and display the confusion matrix, sensitivity, and specificity with a classification threshold of 0.5

Hints:

- Add a 'Y_Hat' classification prediction to your model_assessment_data dataframe
    - Several ways to do this. Easy way is with the np.where function
- Use metrics.confusion_matrix from sklearn and sns.heatmap from seaborn
- Create variables true_negatives, true_positives, false_negatives, false_positives from output of your metrics.confusion_matrix function call and use these values to calculate sensitivity and specificity.

In [52]:

```python
from sklearn import metrics
import seaborn as sns
```

d) Create an ROC chart (using 1% intervals for the threshold)

Step 1: Create roc_data dataframe containing three columns, each with length 99: "Threshold", "FPR", and "TPR". Populate "Threshold" with the values

1 through 99 and populate "FPR" and "TPR" with 0.0 using the same approach as step 2 of part 3A.

In [ ]:

Step 2: Create a loop to step through each value of "Threshold"

- Calculate the conf_matrix, true_negatives, true_positives, false_positives, and false_negatives in the same way for each value of Threshold
- Populate the FPR and TPR variables of your roc_data dataframe using these four values

In [ ]:

Step 3: Plot the ROC curve using Matplotlib plt.plot function.

- Hint: create the diagonal line with the following code:
    - ident = [0.0, 1.0]
    - plt.plot(ident, ident)

In [ ]:

e) What value of threshold yields the largest ROC separation?

- Calculate a 99-element "separation" list
- Find the largest value in the list using the np.argmax() function

In [ ]:

f) Recalculate the confusion matrix, sensitivity, and specificity using the optimal threshold you found in part (e):

Repeat part 3C using your new optimal_threshold value

In [ ]: