# MScBMI 33200 – Machine Learning for Biomedical Informatics
# Assignment I
**<Troy Zhongyi Zhang>**

**Directions:**

1. Fill out below information (tables and methods)
2. Write as much detail as possible for the methods section for each problem. The idea is for you to have a record of all the steps you took for building each model.
3. Submit this document along with your code in an HTML/PDF format

## Section 1: EMR Bots 30-day Readmission study

**Using the training datasets, create the following models:**

**1. Naïve model: This model utilizes only patient characteristics (age, gender and race) to predict 30-day readmission in a logistic regression framework**

**2. Logistic Regression model : This model utilizes patient characteristics and most-recent lab recordings to predict 30-day admissions in a logistic regression framework.**

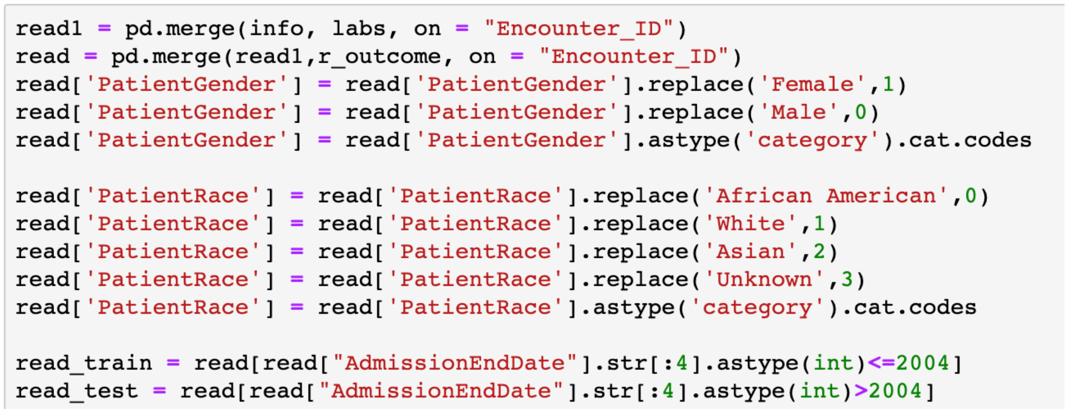
**Calculate AUC on the test dataset. Fill out the following Table.**

| | AUC (95% CI) |
|---|---|
| Naïve Model | 0.4909 (0.4105, 0.5714) |
| Logistic Regression | 0.4843 (0.3980, 0.5706) |

**Insert details (packages, parameter selection, etc.) on the models that were developed in the space given below.**

**Methods:**

Firstly I transferred all the texts data to categorical data for binary classification, and then I selected the training set (year<=2004) and testing set (year>2004). Since the Naïve Bayes model and logistic regression used different features, I have to prepare two data sets for modeling.



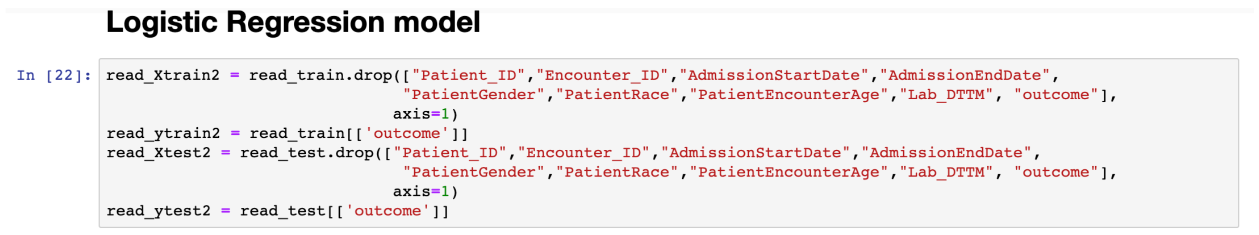
```
read1 = pd.merge(info, labs, on = "Encounter_ID")
read = pd.merge(read1,r_outcome, on = "Encounter_ID")
read['PatientGender'] = read['PatientGender'].replace('Female',1)
read['PatientGender'] = read['PatientGender'].replace('Male',0)
read['PatientGender'] = read['PatientGender'].astype('category').cat.codes

read['PatientRace'] = read['PatientRace'].replace('African American',0)
read['PatientRace'] = read['PatientRace'].replace('White',1)
read['PatientRace'] = read['PatientRace'].replace('Asian',2)
read['PatientRace'] = read['PatientRace'].replace('Unknown',3)
read['PatientRace'] = read['PatientRace'].astype('category').cat.codes

read_train = read[read["AdmissionEndDate"].str[:4].astype(int)<=2004]
read_test = read[read["AdmissionEndDate"].str[:4].astype(int)>2004]
```

--> Naïve

For logistic regression:

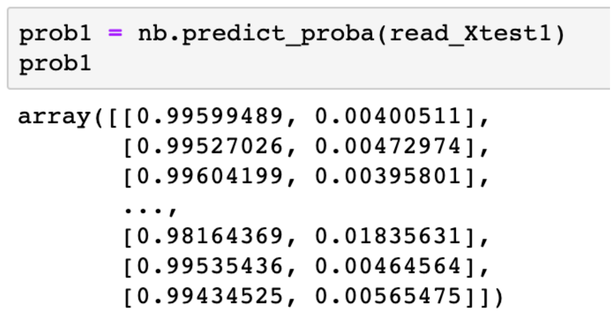


## Logistic Regression model

```
In [22]: read_Xtrain2 = read_train.drop(["Patient_ID","Encounter_ID","AdmissionStartDate","AdmissionEndDate",
                                         "PatientGender","PatientRace","PatientEncounterAge","Lab_DTTM", "outcome"],
                                        axis=1)
         read_ytrain2 = read_train[['outcome']]
         read_Xtest2 = read_test.drop(["Patient_ID","Encounter_ID","AdmissionStartDate","AdmissionEndDate",
                                       "PatientGender","PatientRace","PatientEncounterAge","Lab_DTTM", "outcome"],
                                      axis=1)
         read_ytest2 = read_test[['outcome']]
```

I used Python for modeling. I imported GaussianNB for the Naïve Model and LogisticRegression for the Logistic Rregression model from sklearn package.
from sklearn.naive_bayes import GaussianNB
I fitted the training set and predicted testing set
Then I used predict_proba to calculate the probability of my predictions.

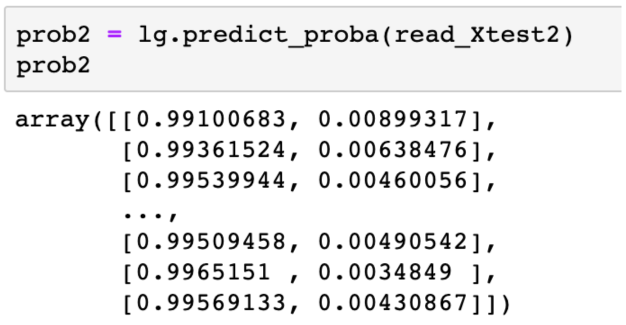
<u>nb – Naïve model:</u>



```
prob1 = nb.predict_proba(read_Xtest1)
prob1
```

```
array([[0.99599489, 0.00400511],
       [0.99527026, 0.00472974],
       [0.99604199, 0.00395801],
       ...,
       [0.98164369, 0.01835631],
       [0.99535436, 0.00464564],
       [0.99434525, 0.00565475]])
```

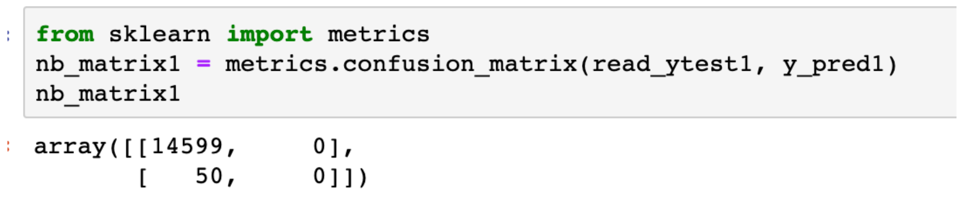
<u>lg – logistic regression:</u>



```
prob2 = lg.predict_proba(read_Xtest2)
prob2
```

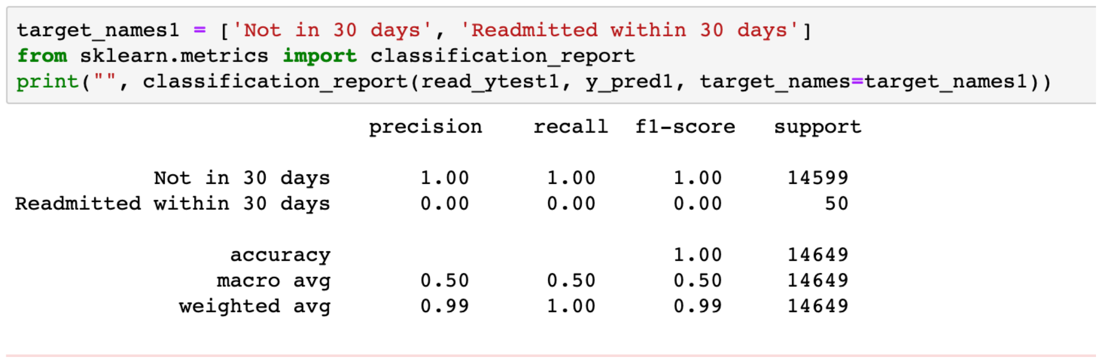
```
array([[0.99100683, 0.00899317],
       [0.99361524, 0.00638476],
       [0.99539944, 0.00460056],
       ...,
       [0.99509458, 0.00490542],
       [0.9965151 , 0.0034849 ],
       [0.99569133, 0.00430867]])
```

Both Naïve Model and Logistic Regression give me the same prediction results. All the predictions for testing set are “0”. I imported metrics and classification_report from “sklearn” packages for visualization on the accuracy scores by confusion matrix:



```
from sklearn import metrics
nb_matrix1 = metrics.confusion_matrix(read_ytest1, y_pred1)
nb_matrix1
```

```
array([[14599,     0],
       [   50,     0]])
```



```
target_names1 = ['Not in 30 days', 'Readmitted within 30 days']
from sklearn.metrics import classification_report
print("", classification_report(read_ytest1, y_pred1, target_names=target_names1))
```

```
                            precision    recall  f1-score   support

           Not in 30 days       1.00      1.00      1.00     14599
Readmitted within 30 days       0.00      0.00      0.00        50

                 accuracy                           1.00     14649
                macro avg       0.50      0.50      0.50     14649
             weighted avg       0.99      1.00      0.99     14649
```

By using the roc_auc_score from sklearn package, I got 49.09% for AUC of Naïve Model

```
from sklearn.metrics import roc_auc_score
nb_probs = nb.predict_proba(read_Xtest1)[:,1]
print(roc_auc_score(read_ytest1, nb_probs))
```

```
0.49090485649702037
```

I got 48.43% for AUC of Logistic Regression Model

```
lg_probs = lg.predict_proba(read_Xtest2)[:,1]
print(roc_auc_score(read_ytest2, lg_probs))
```

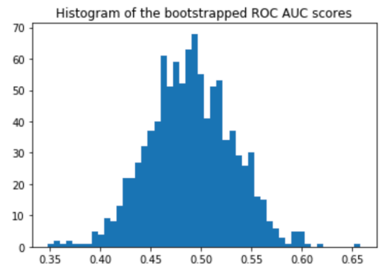
```
0.4842674155764093
```

Finally, since Python doesn't have the pROC package to calculate the confidence interval, I found two ways to obtain the 95% CI.
The first way is bootstrapping to draw the ROC curve and compute the CI, and the second way is to use the DeLong function transplanted from the pROC package in R.

<u>Naïve:</u>

```
print("Confidence interval for the score: [{:0.4f} - {:0.4}]".format(
    confidence_lower, confidence_upper))
```

```
Original ROC area: 0.4909
```

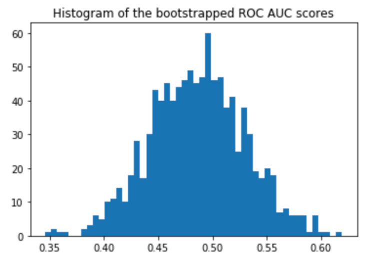


```
Confidence interval for the score: [0.4232 - 0.5578]
```

<u>Bootstrapping:</u>

```
print("Confidence interval for the score: [{:0.4f} - {:0.4}]".format(
    confidence_lower, confidence_upper))
```

```
Original ROC area: 0.4843
```



```
Confidence interval for the score: [0.4147 - 0.5574]
```

Naïve for transplanted pROC package:

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.49090485649702037
AUC COV: 0.0016849429801989745
95% AUC CI: [0.41045214 0.57135757]
```

Logistic Regression for pROC DeLong function:

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.4842674155764093
AUC COV: 0.0019394122536979157
95% AUC CI: [0.39795303 0.5705818 ]
```

Since bootstrapping method has a lot of uncertainties, I would choose the transplanted pROC package with DeLong function for my CI computation. The Confidence Intervals I filled in above form are the pROC calculation results.

## Section 2: Gusto Study

**Using the training datasets, create the following models:**

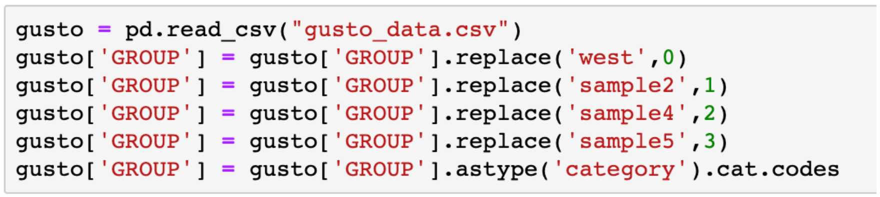
1. **GLM model : This model utilizes all features to predict 30-day mortality in a logistic regression framework.**
2. **Ridge Regression model : This model utilizes all features to predict 30-day mortality in a logistic regression framework with regularization. Utilize a 5 fold cross validation to build the parameters for your model.**

**Calculate AUC on the test dataset. Fill out the following Table.**

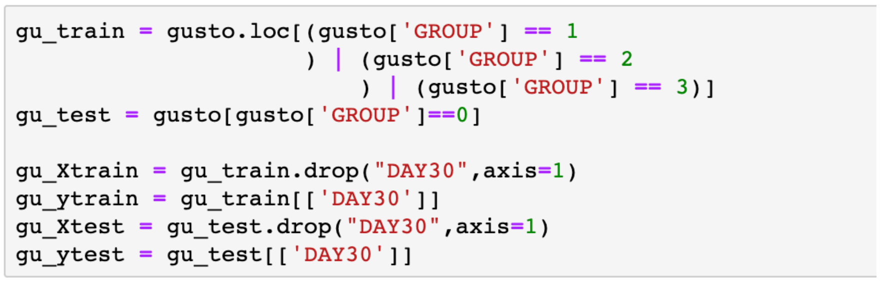
| | AUC (95% CI) |
|---|---|
| Logistic Regression | 0.8285 (0.7919, 0.8650) |
| Ridge Regression | 0.8280 (0.7912, 0.8647) |

**Insert details on the models that were developed in the space given below.**
**Methods:**

Firstly I transferred the text data into categorical int data for classification and split training dataset (GROUP = ‘sample2’, ‘sample4’, and ‘sample5’) and testing dataset(GROUP = ‘west’).



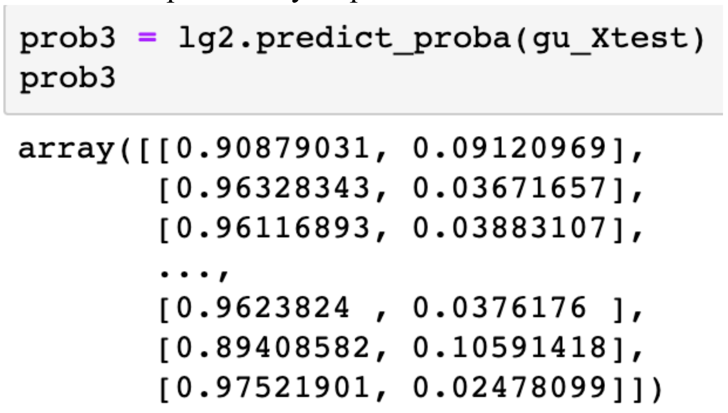
```
gusto = pd.read_csv("gusto_data.csv")
gusto['GROUP'] = gusto['GROUP'].replace('west',0)
gusto['GROUP'] = gusto['GROUP'].replace('sample2',1)
gusto['GROUP'] = gusto['GROUP'].replace('sample4',2)
gusto['GROUP'] = gusto['GROUP'].replace('sample5',3)
gusto['GROUP'] = gusto['GROUP'].astype('category').cat.codes
```



```
gu_train = gusto.loc[(gusto['GROUP'] == 1
                    ) | (gusto['GROUP'] == 2
                        ) | (gusto['GROUP'] == 3)]
gu_test = gusto[gusto['GROUP']==0]

gu_Xtrain = gu_train.drop("DAY30",axis=1)
gu_ytrain = gu_train[['DAY30']]
gu_Xtest = gu_test.drop("DAY30",axis=1)
gu_ytest = gu_test[['DAY30']]
```

Then I used LogisticRegression package from sklearn package to model the data for predictions. Below is the probability of predictions



```
prob3 = lg2.predict_proba(gu_Xtest)
prob3
```

```
array([[0.90879031, 0.09120969],
       [0.96328343, 0.03671657],
       [0.96116893, 0.03883107],
       ...,
       [0.9623824 , 0.0376176 ],
       [0.89408582, 0.10591418],
       [0.97521901, 0.02478099]])
```

Then I I imported metrics and classification_report from "sklearn" packages for visualization on the accuracy scores by confusion matrix for logistic regression:

```
lg_matrix3 = metrics.confusion_matrix(gu_ytest, gu_y_pred)
lg_matrix3
```

```
array([[2037,   16],
       [ 114,   21]])
```

```
target_names3 = ['Still alive at 30 day', 'Died in 30 days']
print("", classification_report(gu_ytest, gu_y_pred, target_names=target_names3))
```

```
                        precision    recall  f1-score   support

Still alive at 30 day       0.95      0.99      0.97      2053
      Died in 30 days       0.57      0.16      0.24       135

             accuracy                           0.94      2188
            macro avg       0.76      0.57      0.61      2188
         weighted avg       0.92      0.94      0.92      2188
```

By using the roc_auc_score from "sklearn" package, I got 82.85% for AUC of logistic regression for GUSTO dataset:

```
lg2_probs = lg2.predict_proba(gu_Xtest)[:,1]
print(roc_auc_score(gu_ytest, lg2_probs))
```

```
0.8284606086846712
```

I import the RidgeCV from the "sklearn.linear_model" package for regularization. I chose some alphas values to try some different parameters. Then I set "cv = 5" in the parenthesis according to the question requirement. By using the roc_auc_score from "sklearn" package, I got 82.80% for AUC of logistic regression for GUSTO dataset:

```
from sklearn.linear_model import RidgeCV
ridgecv = RidgeCV(alphas=[1e-3, 1e-2, 1e-1, 1, 10], cv=5, fit_intercept=True, scoring=None, normalize=False)
ridgecv=ridgecv.fit(gu_Xtrain,gu_ytrain)
gu_ridgecv = ridgecv.predict(gu_Xtest)
roc_auc_score(gu_ytest, gu_ridgecv)
```

```
0.8279590842669263
```

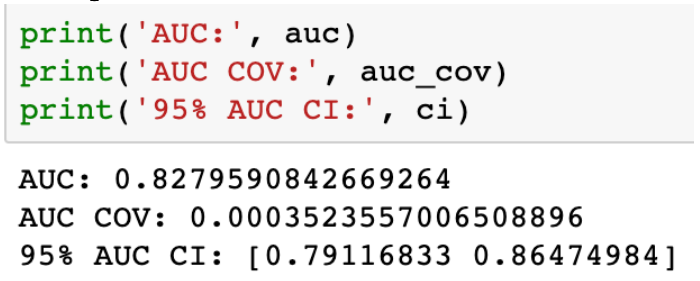
For the 95% confidence intervals, I also did both ways (bootstrapping and pROC) to compute the intervals for logistic regression and Ridge on the GUSTO dataset. I believe the pROC will give a better and more accurate result, I will just show the details of the pROC DeLong computation results.

For Logistic Regression on GUSTO dataset, I got the 95% CI of (0.7919, 0.8650).

```
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.8284606086846711
AUC COV: 0.0003479545222665671
95% AUC CI: [0.79190034 0.86502087]
```

For RidgeCV with 5 cross-validations on GUSTO dataset, I got 95% CI of (0.7912, 0.8647)



```python
print('AUC:', auc)
print('AUC COV:', auc_cov)
print('95% AUC CI:', ci)
```

```
AUC: 0.8279590842669264
AUC COV: 0.0003523557006508896
95% AUC CI: [0.79116833 0.86474984]
```

---

## Section 3: Short Answer Questions

Please answer the following questions briefly.

**a. What are the assumptions of a linear prediction model?**

Ans.

The regression has five key assumptions:

1. Linear relationship.
2. Multivariate normality.
3. No or little multicollinearity.
4. No auto-correlation
5. Homoscedasticity

Linear regression needs the relationship between the independent and dependent variables to be linear. The linear regression analysis requires all variables to be multivariate normal. Linear regression analysis requires that there is little or no autocorrelation in the data.

**b. Explain briefly what the interpretation of R2 is.**

Ans.

R-squared ($R^2$) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable. So, if the $R^2$ of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs.

**c. What is learning rate in the gradient descent algorithm? Explain what happens when it is too high. Explain what happened when it is too low.**

Ans.

The learning rate is the step size on how much the gradient descent model moves toward the global minimum. If the learning rate is too high, it will possibly pass over the minimum point. If the learning rate is too low, it will take the model too long to reach to the minimum point.

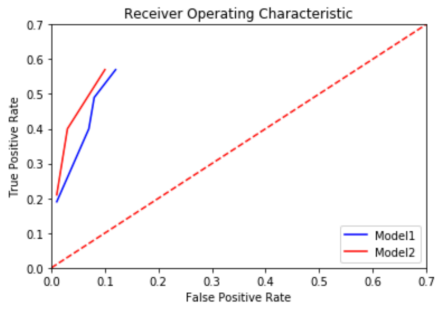
**d. Consider the following scenario. You developed two models for predicting mortality in the hospital using the same dataset. Model 1 had an AUC of 0.78 (95%CI 0.76-0.80). Model 2**

**had an AUC of 0.72 (95% CI 0.70-0.73). When you took a closer look at the sensitivity and specificity measures for various thresholds (predicted probabilities x 1000), you saw the following:**

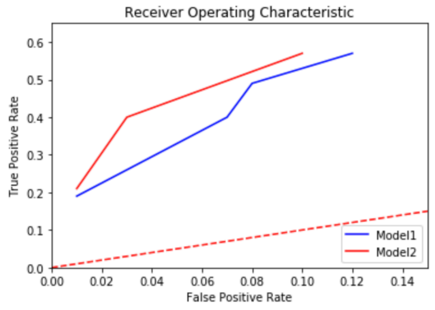
| Model Cutoff | Sensitivity (%, 95%CI) | Specificity (%, 95% CI) |
|---|---|---|
| Model 1 (Predicted Probability x1000) | | |
| ≥ 17 | 57 (55-59) | 88 (87-88) |
| ≥ 20 | 53 (51-55) | 90 (90-91) |
| ≥ 23 | 49 (47-51) | 92 (92-92) |
| ≥ 27 | 40 (38-41) | 93 (93-93) |
| ≥ 67 | 19 (18-21) | 99 (99-99) |
| Model 2 (Predicted Probability x 1000) | | |
| ≥ 16 | 57 (55-59) | 90 (90-91) |
| ≥ 20 | 40 (38-41) | 97 (97-97) |
| ≥ 30 | 21 (19-22) | 99 (99-99) |

**Which is a better model to operationalize? And why?**

Ans. The better model to operationalize is really depends, but I would like to choose Model 1 to be a better model.

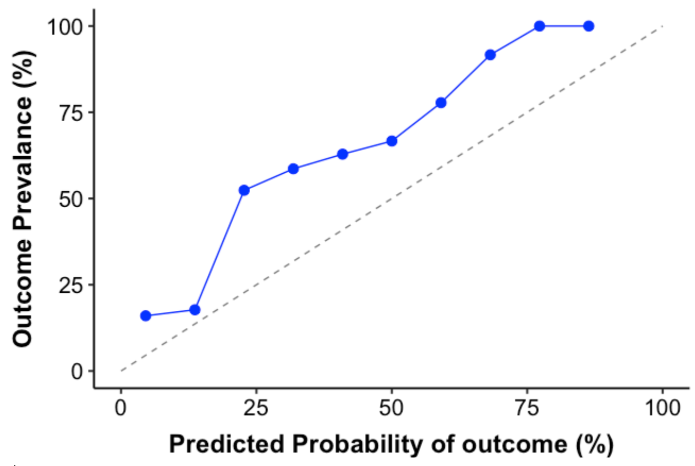


By magnifying the plot through modifying the x-axis and y-axis, I got a plot like



Although Model 2 (red line) is above the model 1 (blue line), the Model 1 has higher AUC_ROC than model2's AUC_ROC. If the data is too specific, the dataset is more likely overfitting. However, Model 1 has higher AUC compared with model 2, which means comprehensively speaking, the Model 1 could predict the True Positive more accurately. The two models have very similar sensitivity, but model 1 has higher specificity, which means model 1 could predict the True False more accurately. I would pick Model 1 to be a better model to operationalize.

**e. Consider the following calibration plot. What can you infer from this plot?**



Ans.
The y-axis is the relative frequency of what was observed, and the x-axis is the predicted probability frequency. Usually the closer the points appeared along the main diagonal from bottom left to upper right, the better calibrated or more reliable a forecast is. The line usually over-forecasts low probabilities and under-forecasts high probabilities. As the predicted probability of outcome increases, the outcome prevalence also increases. The plot line is always above the diagonal means the model has under-forecast and the probabilities are too small.

Section 4 (Optional): Watch the following talk: Yann Le Cunn: Who's afraid of non-convex loss functions? https://www.youtube.com/watch?v=8zdo6cnCW2w