Table of Contents


```
close all; clear; clc;
```

Load dataset

```
datapath = '/Users/lynnhuang/Documents/Data Science/ML/CourseWork/
diabetes_binary_health_indicators_BRFSS2015.csv';
df = readtable(datapath);
```

Initial Analysis of the Data Set

```
target_col = categorical(df.Diabetes_binary);
continuous_col = [df.BMI,df.MentHlth,df.PhysHlth];
ordinal_col = [df.GenHlth,df.Age,df.Education,df.Income];
categorical_col = [df.HighBP,df.HighChol,df.CholCheck,df.Smoker,df.Stroke, ...
    df.HeartDiseaseorAttack,df.PhysActivity,df.Fruits,df.Veggies, ...
    df.HvyAlcoholConsump,df.AnyHealthcare,df.NoDocbcCost,df.DiffWalk,df.Sex];

% Target distribution
target = cell2table(tabulate(target_col),'VariableNames',
{'Diabetes_binary','Count','Percent'});

figure;
bar(target.Diabetes_binary,target.Count);
xticklabels({'Non-diabetes','Diabetes'});
title('Figure 1: Target Distribution');

% Categorical features count within two groups
cat_name = {'HighBP','HighChol','CholCheck','Smoker','Stroke', ...
    'HeartDiseaseorAttack','PhysActivity','Fruits','Veggies', ...
    'HvyAlcoholConsump','AnyHealthcare','NoDocbcCost','DiffWalk','Sex'};
for i = 1:14
```

```
    fprintf('Crosstab for %s with Diabetes:\n', cat_name{i});
    crosstab_result = crosstab(categorical_col(:,i), target_col);
    disp(crosstab_result);
end

% Histogram for coutinuous features within two groups
df0 = df(df.Diabetes_binary == 0, :);
df1 = df(df.Diabetes_binary == 1, :);

figure;
hm1 = histogram(df0.MentHlth,'DisplayName','Non-diabetes');
hold on
hm2 = histogram(df1.MentHlth,'DisplayName','Diabetes');
hm2.FaceColor = [0.8 0 0];
legend('show');
title('MentHlth');

figure;
hp1 = histogram(df0.PhysHlth,'DisplayName','Non-diabetes');
hold on
hp2 = histogram(df1.PhysHlth,'DisplayName','Diabetes');
hp2.FaceColor = [0.8 0 0];
legend('show');
title('PhysHlth');

figure;
hb1 = histogram(df0.BMI,25,'DisplayName','Non-diabetes');
hold on
hb2 = histogram(df1.BMI,25,'DisplayName','Diabetes');
hb2.FaceColor = [0.8 0 0];
legend('show');
title('BMI');

% Spearman correlation with continuous and ordinal features
spear_r = corr([continuous_col,ordinal_col],
[continuous_col,ordinal_col],'Type','Spearman');
figure;
h = heatmap(spear_r,'ColorLimits',[-1,1]);
CustomLabels =
({'BMI','MentHlth','PhysHlth','GenHlth','Age','Education','Income'});
title('Spearman correlation matrix')
colormap(flipud(jet));
h.XDisplayLabels = CustomLabels;
h.YDisplayLabels = CustomLabels;

% Make necessary change in the continuous features for model training
df.MentHlth_g2 = (df.MentHlth >= 1);
df.PhysHlth_g2 = (df.PhysHlth >= 1);
df.BMI_z = zscore(df.BMI);

df_clean = removevars(df, {'BMI','MentHlth','PhysHlth'});

Crosstab for HighBP with Diabetes:
      136109        8742
```

```
       82225       26604

Crosstab for HighChol with Diabetes:
      134429       11660
       83905       23686

Crosstab for CholCheck with Diabetes:
        9229         241
      209105       35105

Crosstab for Smoker with Diabetes:
      124228       17029
       94106       18317

Crosstab for Stroke with Diabetes:
      211310       32078
        7024        3268

Crosstab for HeartDiseaseorAttack with Diabetes:
      202319       27468
       16015        7878

Crosstab for PhysActivity with Diabetes:
       48701       13059
      169633       22287

Crosstab for Fruits with Diabetes:
       78129       14653
      140205       20693

Crosstab for Veggies with Diabetes:
       39229        8610
      179105       26736

Crosstab for HvyAlcoholConsump with Diabetes:
      204910       34514
       13424         832

Crosstab for AnyHealthcare with Diabetes:
       10995        1422
      207339       33924

Crosstab for NoDocbcCost with Diabetes:
      200722       31604
       17612        3742

Crosstab for DiffWalk with Diabetes:
      188780       22225
       29554       13121

Crosstab for Sex with Diabetes:
      123563       18411
       94771       16935
```

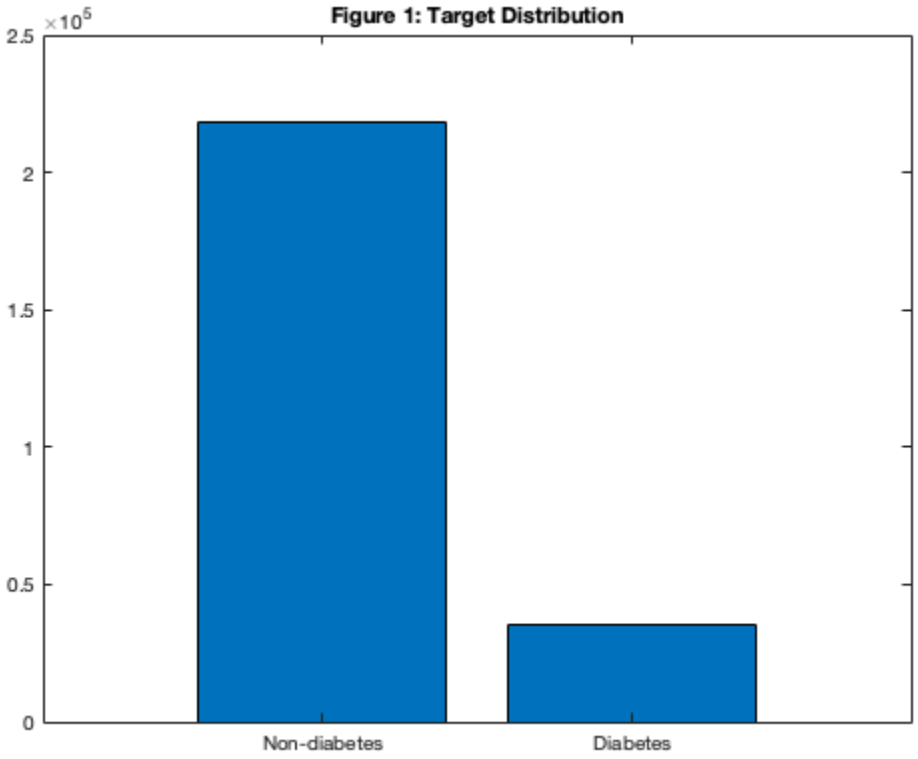
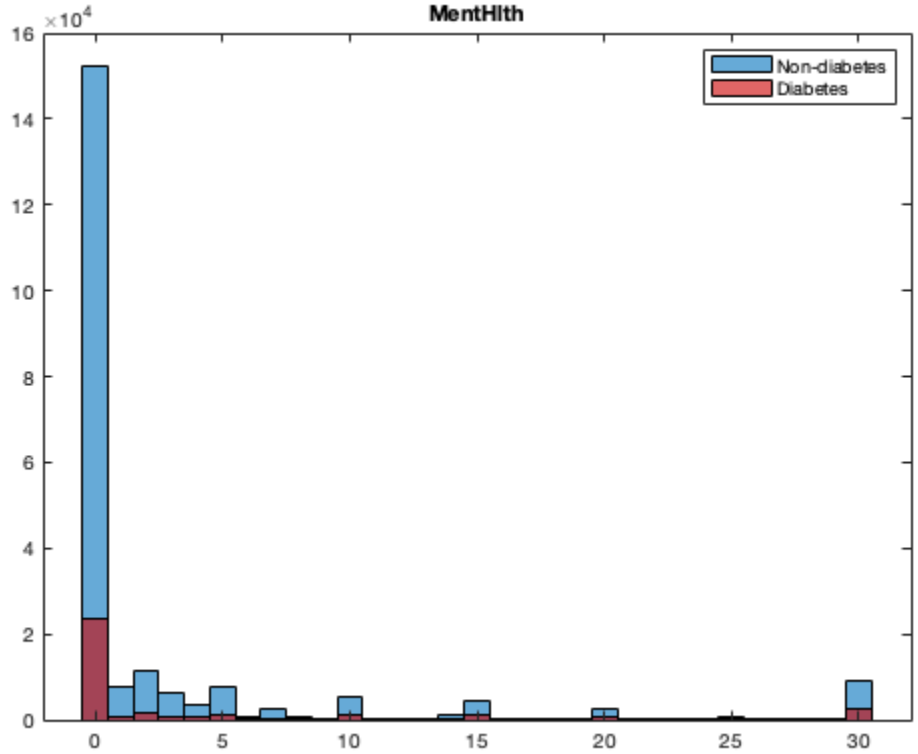
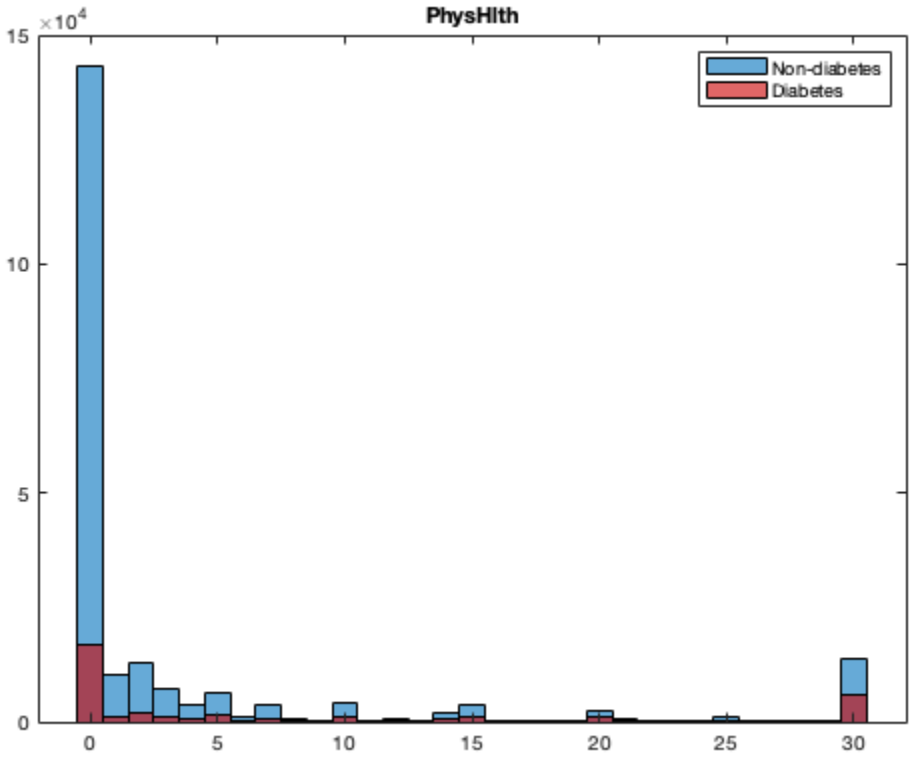
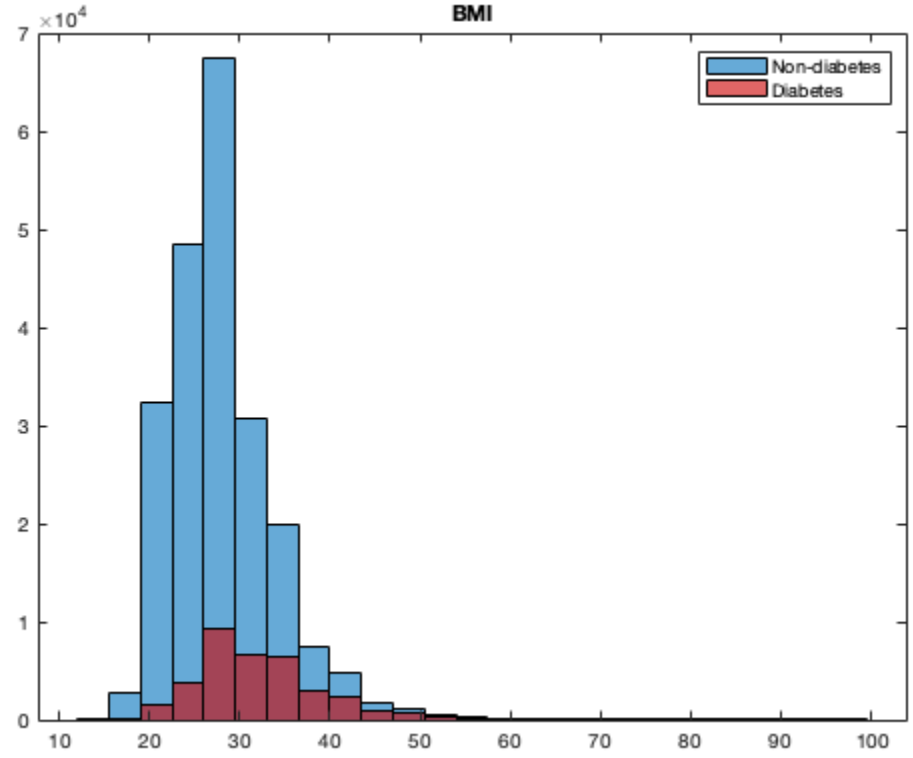
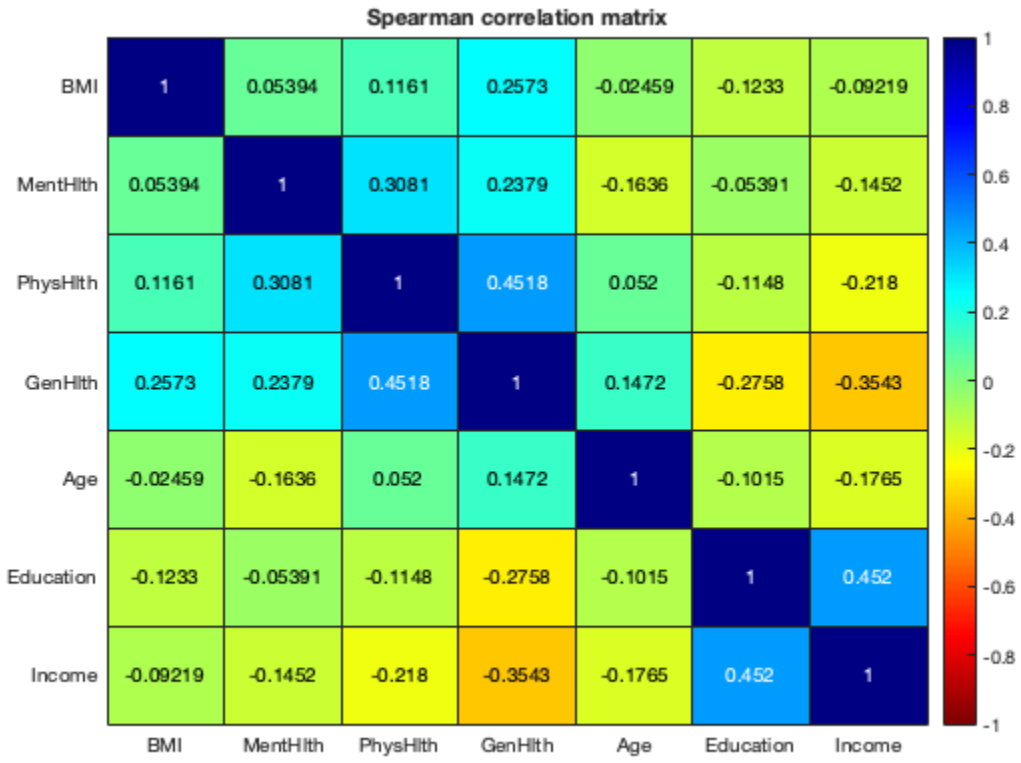


Figure 1: Target Distribution
×10^5
2.5
2
1.5
1
0.5
0
Non-diabetes
Diabetes


MentHlth
×10^4
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
Non-diabetes
Diabetes


PhysHlth
×10^4
15
10
5
0
0
5
10
15
20
25
30
Non-diabetes
Diabetes


BMI
×10^4
7
6
5
4
3
2
1
0
10
20
30
40
50
60
70
80
90
100
Non-diabetes
Diabetes



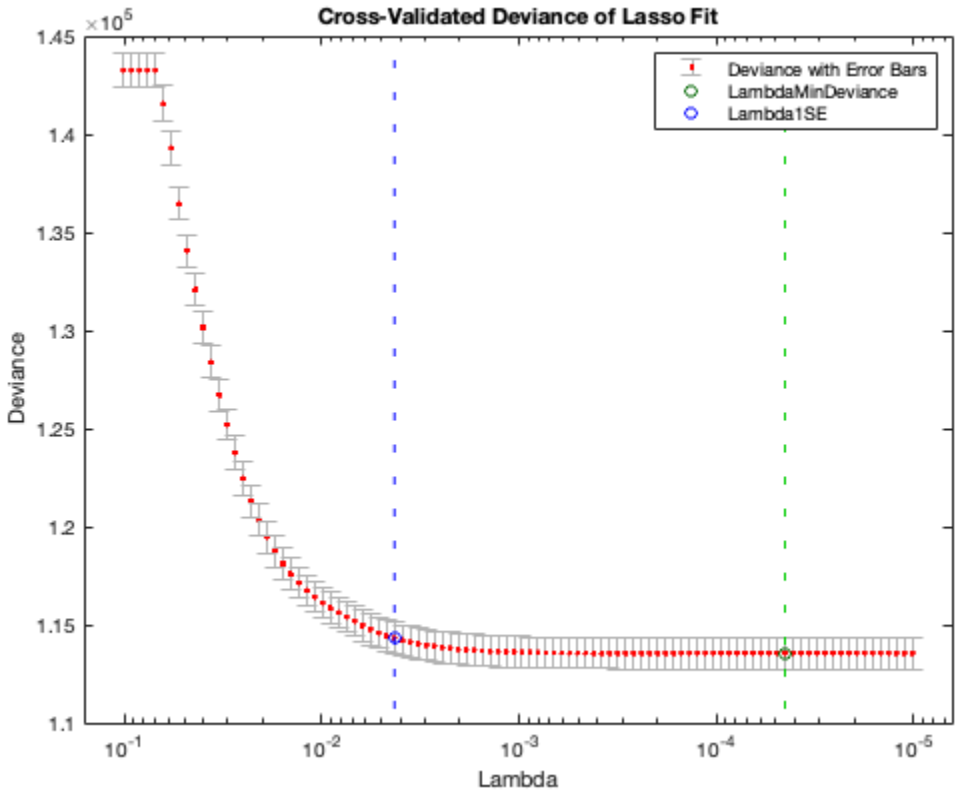
Split data into train and test datasets with 70% & 30%

```
rng(73);
c = cvpartition(size(df_clean,1),'HoldOut',0.3);
train_df = df_clean(c.training,:);

X_train = table2array(train_df(:,2:22));
y_train = train_df.Diabetes_binary;

% test_df will be blind until the model is trained
test_df = df_clean(c.test,:);
% output test set
writetable(test_df, 'test_df.csv');
```

Choose Logistic regression (LR) as a fist ML model

```
% Using L1 regularization to select the features in LR
rng(433);
[B,FitInfo] = lassoglm(X_train,y_train,'binomial','CV',3);

lassoPlot(B,FitInfo,PlotType="CV");
```

```
legend("show")

% Use MSE rule and 1SE rule to select features
idxLambdaMinDeviance = FitInfo.IndexMinDeviance;
mincoefs = find(B(:,idxLambdaMinDeviance));

idxLambda1SE = FitInfo.Index1SE;
min1coefs = find(B(:,idxLambda1SE));
```



Use MSE rule selected features as a LR model 1 with 10-fold cv

```
X_train_LR1 = X_train(:,mincoefs);

% 10-fold cv
rng(4);
K = 10;
cv = cvpartition(size(y_train, 1), 'KFold', K);

% Training results for LR model 1
Coeff_LR1 = zeros(length(mincoefs)+1,K);
TE_LR1 = zeros(K,1);
F1_LR1 = zeros(K,1);
Precision_LR1 = zeros(K,1);
Recall_LR1 = zeros(K,1);
```

```
Accuracy_LR1 = zeros(K,1);
AUC_LR1 = zeros(K,1);

% cutoff=0.5
TE_LR1_2 = zeros(K,1);
F1_LR1_2 = zeros(K,1);
Precision_LR1_2 = zeros(K,1);
Recall_LR1_2 = zeros(K,1);
Accuracy_LR1_2 = zeros(K,1);

% Start the timer
tic;
for fold = 1:K
    trainIdx = training(cv, fold);
    testIdx = test(cv, fold);

    X_train_cv = X_train_LR1(trainIdx, :);
    y_train_cv = y_train(trainIdx);

    X_test_cv = X_train_LR1(testIdx, :);
    y_test_cv = y_train(testIdx);

    % Train LR model for each fold
    Coeff_LR1(:, fold) =
glmfit(X_train_cv,y_train_cv,'binomial','link','logit');

    % Training set error in fold
    y_train_fit = glmval(Coeff_LR1(:, fold),X_train_cv,'logit');
    TE_LR1(fold) = sum((y_train_fit >= 0.14) ~= y_train_cv) /
length(y_train_cv);
    TE_LR1_2(fold) = sum((y_train_fit >= 0.5) ~= y_train_cv) /
length(y_train_cv);

    % F1 score, Precision, Recall, Accuracy with cutoff=0.14 for the test set
in fold
    y_fit = glmval(Coeff_LR1(:, fold),X_test_cv,'logit');
    tp = sum(((y_fit >= 0.14) == 1) & (y_test_cv == 1));
    fp = sum(((y_fit >= 0.14) == 1) & (y_test_cv == 0));
    tn = sum(((y_fit >= 0.14) == 0) & (y_test_cv == 0));
    fn = sum(((y_fit >= 0.14) == 0) & (y_test_cv == 1));

    F1_LR1(fold) = 2*tp / (2*tp + fp + fn);
    Precision_LR1(fold) = tp / (tp + fp);
    Recall_LR1(fold) = tp / (tp + fn);
    Accuracy_LR1(fold) = (tp + tn) / (tp + tn + fp + fn);

    % F1 score, Precision, Recall, Accuracy with cutoff=0.5 for the test set
in fold
    y_fit = glmval(Coeff_LR1(:, fold),X_test_cv,'logit');
    tp2 = sum(((y_fit >= 0.5) == 1) & (y_test_cv == 1));
    fp2 = sum(((y_fit >= 0.5) == 1) & (y_test_cv == 0));
    tn2 = sum(((y_fit >= 0.5) == 0) & (y_test_cv == 0));
    fn2 = sum(((y_fit >= 0.5) == 0) & (y_test_cv == 1));
```

```
    F1_LR1_2(fold) = 2*tp2 / (2*tp2 + fp2 + fn2);
    Precision_LR1_2(fold) = tp2 / (tp2 + fp2);
    Recall_LR1_2(fold) = tp2 / (tp2 + fn2);
    Accuracy_LR1_2(fold) = (tp2 + tn2) / (tp2 + tn2 + fp2 + fn2);

    % AUC for the test set in fold
    [~, ~, ~, AUC_LR1(fold)] = perfcurve(y_test_cv, y_fit, 1);

end
% Stop the timer
Time_LR1 = toc;

% Print the training results in LR model 1
disp('Train results in LR model 1');
disp(['Mean training error in LR model 1: ', num2str(mean(TE_LR1))]);
disp(['Mean validation F1 score in LR model 1: ', num2str(mean(F1_LR1))]);
disp(['Mean validation Precision in LR model 1: ',
num2str(mean(Precision_LR1))]);
disp(['Mean validation Recall in LR model 1: ', num2str(mean(Recall_LR1))]);
disp(['Mean validation Accuracy in LR model 1: ',
num2str(mean(Accuracy_LR1))]);
disp(['Mean validation AUC in LR model 1: ', num2str(mean(AUC_LR1))]);
disp(['Training time in LR model 1: ', num2str(Time_LR1)]);
disp(' ')
```

```
Train results in LR model 1
Mean training error in LR model 1: 0.27009
Mean validation F1 score in LR model 1: 0.44119
Mean validation Precision in LR model 1: 0.30961
Mean validation Recall in LR model 1: 0.76742
Mean validation Accuracy in LR model 1: 0.72959
Mean validation AUC in LR model 1: 0.82177
Training time in LR model 1: 3.4819
```

Use 1SE rule selected features as a LR model 2 with 10-fold cv

```
X_train_LR2 = X_train(:,min1coefs);

% 10-fold cv
rng(110);
K = 10;
cv = cvpartition(size(y_train, 1), 'KFold', K);

% Training results for LR model 2
Coeff_LR2 = zeros(length(min1coefs)+1,K);
TE_LR2 = zeros(K,1);
F1_LR2 = zeros(K,1);
Precision_LR2 = zeros(K,1);
Recall_LR2 = zeros(K,1);
Accuracy_LR2 = zeros(K,1);
```

```
AUC_LR2 = zeros(K,1);

% cutoff=0.5
TE_LR2_2 = zeros(K,1);
F1_LR2_2 = zeros(K,1);
Precision_LR2_2 = zeros(K,1);
Recall_LR2_2 = zeros(K,1);
Accuracy_LR2_2 = zeros(K,1);

% Start the timer
tic;
for fold = 1:K
    trainIdx = training(cv, fold);
    testIdx = test(cv, fold);

    X_train_cv = X_train_LR2(trainIdx, :);
    y_train_cv = y_train(trainIdx);

    X_test_cv = X_train_LR2(testIdx, :);
    y_test_cv = y_train(testIdx);

    % Train LR model for each fold
    Coeff_LR2(:, fold) =
glmfit(X_train_cv,y_train_cv,'binomial','link','logit');

    % Training set error in fold
    y_train_fit = glmval(Coeff_LR2(:, fold),X_train_cv,'logit');
    TE_LR2(fold) = sum((y_train_fit >= 0.14) ~= y_train_cv) /
length(y_train_cv);
    TE_LR2_2(fold) = sum((y_train_fit >= 0.5) ~= y_train_cv) /
length(y_train_cv);

    % F1 score, Precision, Recall, Accuracy with cutoff=0.14 for the test set
in fold
    y_fit = glmval(Coeff_LR2(:, fold),X_test_cv,'logit');
    tp = sum(((y_fit >= 0.14) == 1) & (y_test_cv == 1));
    fp = sum(((y_fit >= 0.14) == 1) & (y_test_cv == 0));
    tn = sum(((y_fit >= 0.14) == 0) & (y_test_cv == 0));
    fn = sum(((y_fit >= 0.14) == 0) & (y_test_cv == 1));

    F1_LR2(fold) = 2*tp / (2*tp + fp + fn);
    Precision_LR2(fold) = tp / (tp + fp);
    Recall_LR2(fold) = tp / (tp + fn);
    Accuracy_LR2(fold) = (tp + tn) / (tp + tn + fp + fn);

    % F1 score, Precision, Recall, Accuracy with cutoff=0.5 for the test set
in fold
    y_fit = glmval(Coeff_LR2(:, fold),X_test_cv,'logit');
    tp2 = sum(((y_fit >= 0.5) == 1) & (y_test_cv == 1));
    fp2 = sum(((y_fit >= 0.5) == 1) & (y_test_cv == 0));
    tn2 = sum(((y_fit >= 0.5) == 0) & (y_test_cv == 0));
    fn2 = sum(((y_fit >= 0.5) == 0) & (y_test_cv == 1));

    F1_LR2_2(fold) = 2*tp2 / (2*tp2 + fp2 + fn2);
```

```
    Precision_LR2_2(fold) = tp2 / (tp2 + fp2);
    Recall_LR2_2(fold) = tp2 / (tp2 + fn2);
    Accuracy_LR2_2(fold) = (tp2 + tn2) / (tp2 + tn2 + fp2 + fn2);

    % AUC for the test set in fold
    [~, ~, ~, AUC_LR2(fold)] = perfcurve(y_test_cv, y_fit, 1);

end
% Stop the timer
Time_LR2 = toc;

% Print the training results in LR model 2
disp('Train results in LR model 2');
disp(['Mean training error in LR model 2: ', num2str(mean(TE_LR2))]);
disp(['Mean validation F1 score in LR model 2: ', num2str(mean(F1_LR2))]);
disp(['Mean validation Precision in LR model 2: ',
num2str(mean(Precision_LR2))]);
disp(['Mean validation Recall in LR model 2: ', num2str(mean(Recall_LR2))]);
disp(['Mean validation Accuracy in LR model 2: ',
num2str(mean(Accuracy_LR2))]);
disp(['Mean validation AUC in LR model 2: ', num2str(mean(AUC_LR2))]);
disp(['Training time in LR model 2: ', num2str(Time_LR2)]);
disp(' ')

% output LR model 2 coefficient for testing data
B_LR2 = mean(Coeff_LR2,2);
```

```
Train results in LR model 2
Mean training error in LR model 2: 0.27021
Mean validation F1 score in LR model 2: 0.44158
Mean validation Precision in LR model 2: 0.30993
Mean validation Recall in LR model 2: 0.76791
Mean validation Accuracy in LR model 2: 0.72984
Mean validation AUC in LR model 2: 0.82178
Training time in LR model 2: 2.1081
```

Choose Naive Bayes as a second ML model

```
% Use chi-square and ANOVA test's p-value as a feature selection principle
pval = [transpose(1:21), zeros(21,1)];

for i = 1:20
    [~, ~, pval(i,2)] = crosstab(X_train(:,i),y_train);
end

[pval(21,2), ~, ~] = anova1(X_train(:,21),y_train, 'off');

pval(:,3) = pval(:,2) < 0.0024;
NB_feature1 = pval(pval(:,3)==1,1);

sorted_pval = sortrows(pval,2);
```

```
smallest_15p = sorted_pval(1:15,1);
NB_feature2 = sortrows(smallest_15p,1);
```

Using p-value < 0.0024 as a first model's features with 10-fold cv

```
X_train_NB1 = X_train(:,NB_feature1);

% 10-fold cv
rng(321);
K = 10;
cv = cvpartition(size(y_train, 1), 'KFold', K);

% Training results for NB model 1
TE_NB1 = zeros(K,1);
F1_NB1 = zeros(K,1);
Precision_NB1 = zeros(K,1);
Recall_NB1 = zeros(K,1);
Accuracy_NB1 = zeros(K,1);
AUC_NB1 = zeros(K,1);

Cat_index = 1:(length(NB_feature1)-1);

% Start the timer
tic;
for fold = 1:K
    trainIdx = training(cv, fold);
    testIdx = test(cv, fold);

    X_train_cv = X_train_NB1(trainIdx, :);
    y_train_cv = y_train(trainIdx);

    X_test_cv = X_train_NB1(testIdx, :);
    y_test_cv = y_train(testIdx);

    % Train NB model for each fold
    NB_class =
fitcnb(X_train_cv,y_train_cv,'CategoricalPredictors',Cat_index);

    % Training set error in fold
    y_train_fit = predict(NB_class,X_train_cv);
    TE_NB1(fold) = sum(y_train_fit ~= y_train_cv) / length(y_train_cv);

    % F1 score, Precision, Recall, Accuracy for the test set in fold
    [y_fit, Score, ~] = predict(NB_class,X_test_cv);
    tp = sum((y_fit == 1) & (y_test_cv == 1));
    fp = sum((y_fit == 1) & (y_test_cv == 0));
    tn = sum((y_fit == 0) & (y_test_cv == 0));
    fn = sum((y_fit == 0) & (y_test_cv == 1));

    F1_NB1(fold) = 2*tp / (2*tp + fp + fn);
    Precision_NB1(fold) = tp / (tp + fp);
```

```
    Recall_NB1(fold) = tp / (tp + fn);
    Accuracy_NB1(fold) = (tp + tn) / (tp + tn + fp + fn);

    % AUC for the test in fold
    [~,~,~,AUC_NB1(fold)] = perfcurve(y_test_cv,Score(:,2),1);

end
% Stop the timer
Time_NB1 = toc;

% Print the training results in NB model 1
disp('Train results in NB model 1');
disp(['Mean training error in NB model 1: ', num2str(mean(TE_NB1))]);
disp(['Mean validation F1 score in NB model 1: ', num2str(mean(F1_NB1))]);
disp(['Mean validation Precision in NB model 1: ',
num2str(mean(Precision_NB1))]);
disp(['Mean validation Recall in NB model 1: ', num2str(mean(Recall_NB1))]);
disp(['Mean validation Accuracy in NB model 1: ',
num2str(mean(Accuracy_NB1))]);
disp(['Mean validation AUC in NB model 1: ', num2str(mean(AUC_NB1))]);
disp(['Training time in NB model 1: ', num2str(Time_NB1)]);
disp(' ')

% output NB model for testing data
NB_class1 = fitcnb(X_train_NB1,y_train,'CategoricalPredictors',Cat_index);
```

```
Train results in NB model 1
Mean training error in NB model 1: 0.18433
Mean validation F1 score in NB model 1: 0.43093
Mean validation Precision in NB model 1: 0.37757
Mean validation Recall in NB model 1: 0.502
Mean validation Accuracy in NB model 1: 0.81556
Mean validation AUC in NB model 1: 0.80858
Training time in NB model 1: 2.6534
```

Using 15 smallest p-values as a second model's features with 10-fold cv

```
X_train_NB2 = X_train(:,NB_feature2);

% 10-fold cv
rng(15);
K = 10;
cv = cvpartition(size(y_train, 1), 'KFold', K);

% Training results for NB model 2
TE_NB2 = zeros(K,1);
F1_NB2 = zeros(K,1);
Precision_NB2 = zeros(K,1);
Recall_NB2 = zeros(K,1);
Accuracy_NB2 = zeros(K,1);
```

```
AUC_NB2 = zeros(K,1);

Cat_index = 1:(length(NB_feature2)-1);

% Start the timer
tic;
for fold = 1:K
    trainIdx = training(cv, fold);
    testIdx = test(cv, fold);

    X_train_cv = X_train_NB2(trainIdx, :);
    y_train_cv = y_train(trainIdx);

    X_test_cv = X_train_NB2(testIdx, :);
    y_test_cv = y_train(testIdx);

    % Train NB model for each fold
    NB_class =
fitcnb(X_train_cv,y_train_cv,'CategoricalPredictors',Cat_index);

    % Training set error in fold
    y_train_fit = predict(NB_class,X_train_cv);
    TE_NB2(fold) = sum(y_train_fit ~= y_train_cv) / length(y_train_cv);

    % F1 score, Precision, Recall, Accuracy for the test set in fold
    [y_fit, Score, ~] = predict(NB_class,X_test_cv);
    tp = sum((y_fit == 1) & (y_test_cv == 1));
    fp = sum((y_fit == 1) & (y_test_cv == 0));
    tn = sum((y_fit == 0) & (y_test_cv == 0));
    fn = sum((y_fit == 0) & (y_test_cv == 1));

    F1_NB2(fold) = 2*tp / (2*tp + fp + fn);
    Precision_NB2(fold) = tp / (tp + fp);
    Recall_NB2(fold) = tp / (tp + fn);
    Accuracy_NB2(fold) = (tp + tn) / (tp + tn + fp + fn);

    % AUC for the test in fold
    [~,~,~,AUC_NB2(fold)] = perfcurve(y_test_cv,Score(:,2),1);

end
% Stop the timer
Time_NB2 = toc;

% Print the training results in NB model 2
disp('Train results in NB model 2');
disp(['Mean training error in NB model 2: ', num2str(mean(TE_NB2))]);
disp(['Mean validation F1 score in NB model 2: ', num2str(mean(F1_NB2))]);
disp(['Mean validation Precision in NB model 2: ',
num2str(mean(Precision_NB2))]);
disp(['Mean validation Recall in NB model 2: ', num2str(mean(Recall_NB2))]);
disp(['Mean validation Accuracy in NB model 2: ',
num2str(mean(Accuracy_NB2))]);
disp(['Mean validation AUC in NB model 2: ', num2str(mean(AUC_NB2))]);
```

```
disp(['Training time in NB model 2: ', num2str(Time_NB2)]);
disp(' ')
```

```
Train results in NB model 2
Mean training error in NB model 2: 0.18422
Mean validation F1 score in NB model 2: 0.43016
Mean validation Precision in NB model 2: 0.37769
Mean validation Recall in NB model 2: 0.49969
Mean validation Accuracy in NB model 2: 0.81585
Mean validation AUC in NB model 2: 0.80754
Training time in NB model 2: 1.9106
```

Testing two best model, using F1-score as a evaluation matric

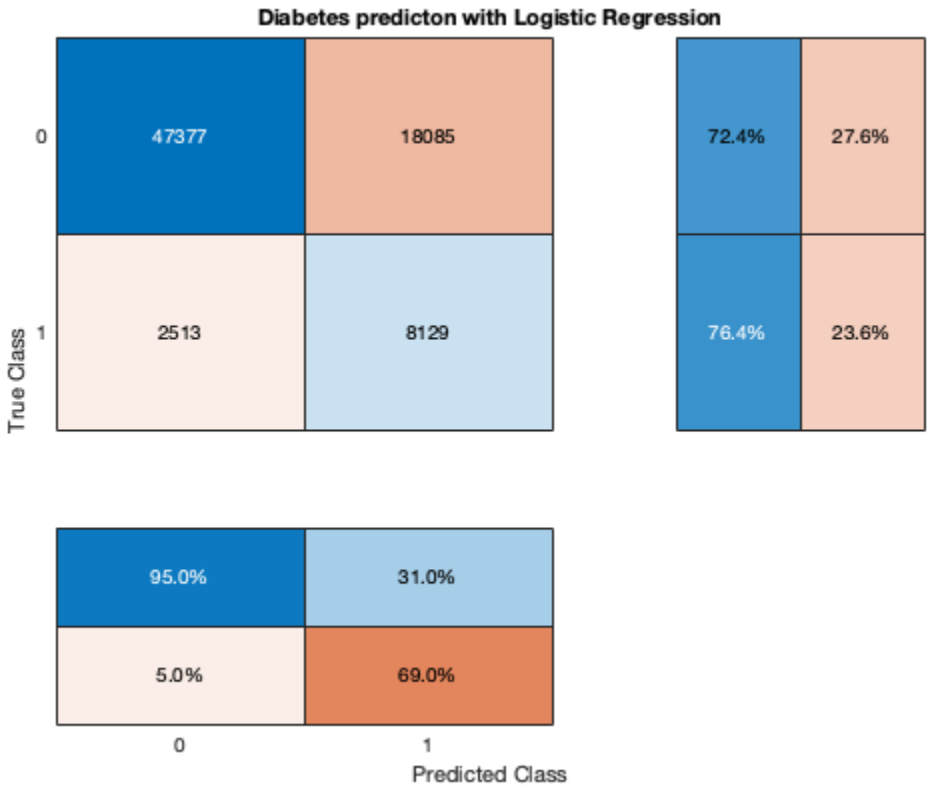
```
% Load test dataset
testpath = '/Users/lynnhuang/Documents/Data Science/ML/CourseWork/ML
Coursework_Ling-Yun, Huang/test_df.csv';
test_df = readtable(testpath);

X_test = table2array(test_df(:,2:22));
y_test = test_df.Diabetes_binary;
```

LR model 2 has a higher F1-score

```
% load best LR model's coefficient and features
load('LR_model')
load('LR_features')
% choose features in LR model2
X_test_LR = X_test(:,min1coefs);


% Start the timer
tic;
% predict test dataset with LR model 2
y_fit = glmval(B_LR2,X_test_LR,'logit');
% Stop the timer
Time_LR = toc;

% Testing results
tp = sum(((y_fit >= 0.14) == 1) & (y_test == 1));
fp = sum(((y_fit >= 0.14) == 1) & (y_test == 0));
tn = sum(((y_fit >= 0.14) == 0) & (y_test == 0));
fn = sum(((y_fit >= 0.14) == 0) & (y_test == 1));

% F1 score, Precision, Recall, Accuracy for the test dataset
F1_LR = 2*tp / (2*tp + fp + fn);
Precision_LR = tp / (tp + fp);
Recall_LR = tp / (tp + fn);
Accuracy_LR = (tp + tn) / (tp + tn + fp + fn);
```

```
% cutoff=0.5
tp2 = sum(((y_fit >= 0.5) == 1) & (y_test == 1));
fp2 = sum(((y_fit >= 0.5) == 1) & (y_test == 0));
tn2 = sum(((y_fit >= 0.5) == 0) & (y_test == 0));
fn2 = sum(((y_fit >= 0.5) == 0) & (y_test == 1));

F1_LR_2 = 2*tp2 / (2*tp2 + fp2 + fn2);
Precision_LR_2 = tp2 / (tp2 + fp2);
Recall_LR_2 = tp2 / (tp2 + fn2);
Accuracy_LR_2 = (tp2 + tn2) / (tp2 + tn2 + fp2 + fn2);

% AUC
[XLR, YLR, ~, AUC_LR] = perfcurve(y_test, y_fit, 1);

% print the testing results in LR model
disp('Test results in LR model');
disp(['F1 score in LR model: ', num2str(F1_LR)]);
disp(['Precision in LR model: ', num2str(Precision_LR)]);
disp(['Recall in LR model: ', num2str(Recall_LR)]);
disp(['Accuracy in LR model: ', num2str(Accuracy_LR)]);
disp(['AUC in LR model: ', num2str(AUC_LR)]);
disp(['Testing time in LR model: ', num2str(Time_LR)]);
disp(' ')

% confusion matrix chart in LR test result
y_predict = double(y_fit >= 0.14);
figure;
cm = confusionchart(y_test,y_predict);
cm.Title = 'Diabetes predicton with Logistic Regression';
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';
```

```
Test results in LR model
F1 score in LR model: 0.44112
Precision in LR model: 0.3101
Recall in LR model: 0.76386
Accuracy in LR model: 0.72934
AUC in LR model: 0.8221
Testing time in LR model: 0.0028169
```

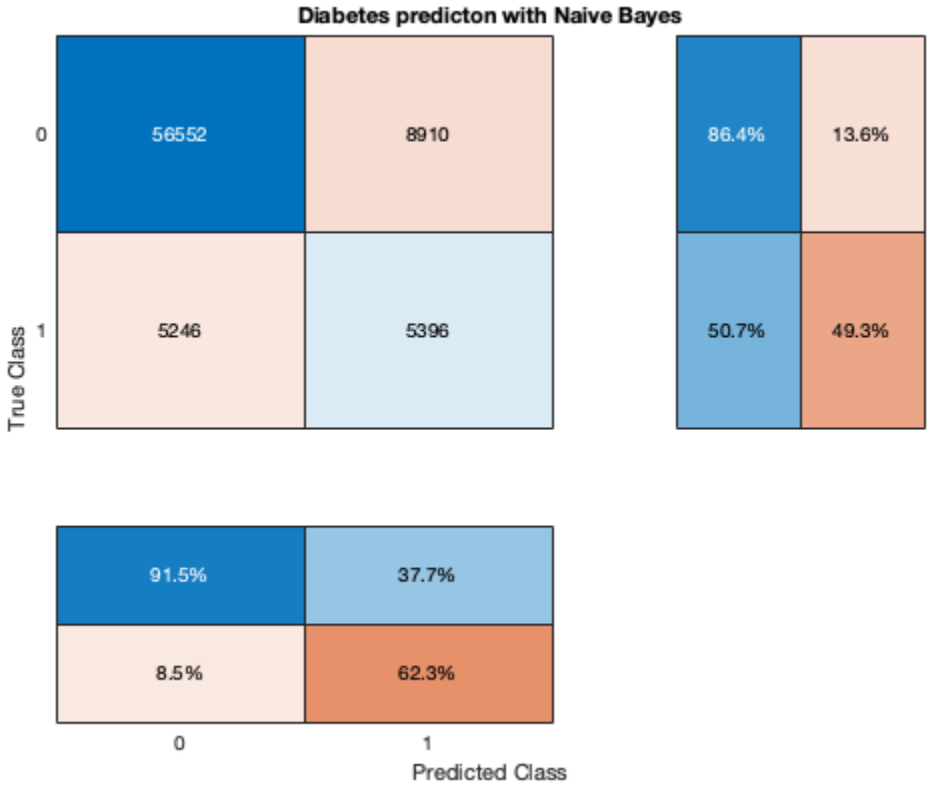


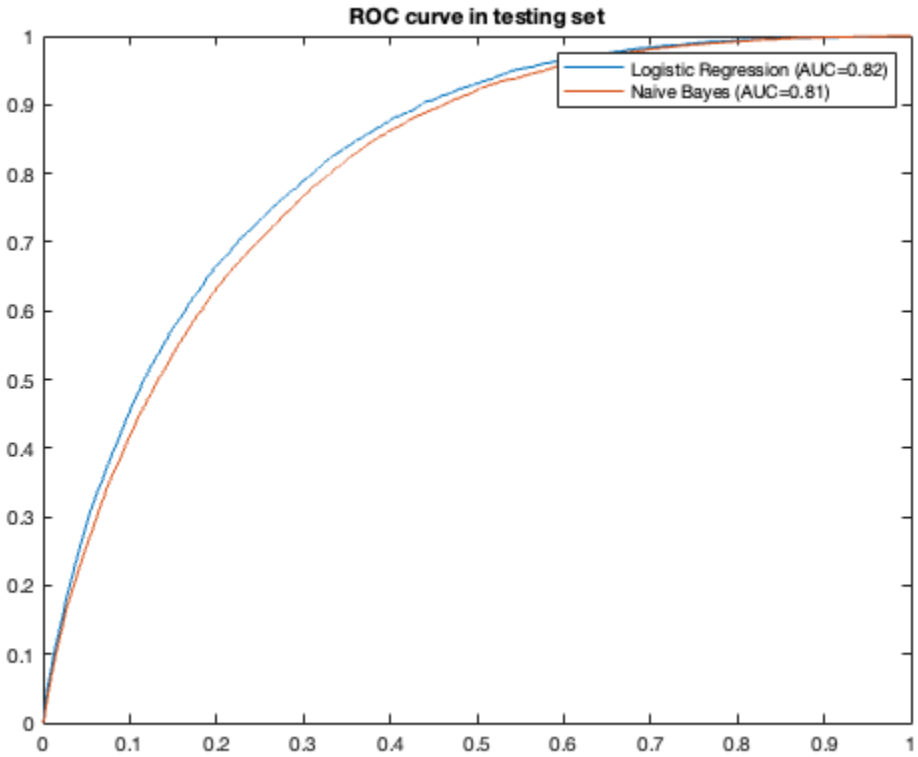
NB model 1 has a higher F1-score

```
% load best NB model
load('NB_model')

% Start the timer
tic;
% predict test dataset with NB model 1
[y_fit, Score, ~] = predict(NB_class1,X_test);
% Stop the timer
Time_NB = toc;

tp = sum((y_fit == 1) & (y_test == 1));
fp = sum((y_fit == 1) & (y_test == 0));
tn = sum((y_fit == 0) & (y_test == 0));
fn = sum((y_fit == 0) & (y_test == 1));

% F1 score, Precision, Recall, Accuracy for the test dataset
F1_NB = 2*tp / (2*tp + fp + fn);
Precision_NB = tp / (tp + fp);
Recall_NB = tp / (tp + fn);
Accuracy_NB = (tp + tn) / (tp + tn + fp + fn);

% AUC for test dataset
[XNB, YNB, ~, AUC_NB] = perfcurve(y_test, Score(:,2), 1);
```

```
% Print the testing results in NB model
disp('Test results in NB model');
disp(['F1 score in NB model: ', num2str(F1_NB)]);
disp(['Precision in NB model: ', num2str(Precision_NB)]);
disp(['Recall in NB model: ', num2str(Recall_NB)]);
disp(['Accuracy in NB model: ', num2str(Accuracy_NB)]);
disp(['AUC in NB model: ', num2str(AUC_NB)]);
disp(['Testing time in NB model: ', num2str(Time_NB)]);
disp(' ')

% confusion matrix chart in NB test result
figure;
cm = confusionchart(y_test,y_fit);
cm.Title = 'Diabetes predicton with Naive Bayes';
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';

% Plot AUC in two methods
figure;
p1 = plot(XLR, YLR,'DisplayName','Logistic Regression (AUC=0.82)');
hold on;
p2 = plot(XNB, YNB,'DisplayName','Naive Bayes (AUC=0.81)');
legend('show');
title('ROC curve in testing set');
```

```
Test results in NB model
F1 score in NB model: 0.43258
Precision in NB model: 0.37718
Recall in NB model: 0.50705
Accuracy in NB model: 0.81399
AUC in NB model: 0.80832
Testing time in NB model: 0.091101
```


Diabetes predicton with Naive Bayes
0
1
True Class
56552
8910
5246
5396
86.4%
13.6%
50.7%
49.3%
91.5%
37.7%
8.5%
62.3%
0
1
Predicted Class


ROC curve in testing set
Logistic Regression (AUC=0.82)
Naive Bayes (AUC=0.81)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Supplementary material results

```
% output cutoff=0.5 in Logistic regression models
disp('Train results in LR model 1 with cutoff=0.5');
disp(['Mean training error in LR model 1: ', num2str(mean(TE_LR1_2))]);
disp(['Mean validation F1 score in LR model 1: ', num2str(mean(F1_LR1_2))]);
disp(['Mean validation Precision in LR model 1: ',
num2str(mean(Precision_LR1_2))]);
disp(['Mean validation Recall in LR model 1: ', num2str(mean(Recall_LR1_2))]);
disp(['Mean validation Accuracy in LR model 1: ',
num2str(mean(Accuracy_LR1_2))]);
disp(' ')

disp('Train results in LR model 2 with cutoff=0.5');
disp(['Mean training error in LR model 2: ', num2str(mean(TE_LR2_2))]);
disp(['Mean validation F1 score in LR model 2: ', num2str(mean(F1_LR2_2))]);
disp(['Mean validation Precision in LR model 2: ',
num2str(mean(Precision_LR2_2))]);
disp(['Mean validation Recall in LR model 2: ', num2str(mean(Recall_LR2_2))]);
disp(['Mean validation Accuracy in LR model 2: ',
num2str(mean(Accuracy_LR2_2))]);
disp(' ')

disp('Test results in LR model with cutoff=0.5');
disp(['F1 score in LR model: ', num2str(F1_LR_2)]);
disp(['Precision in LR model: ', num2str(Precision_LR_2)]);
disp(['Recall in LR model: ', num2str(Recall_LR_2)]);
disp(['Accuracy in LR model: ', num2str(Accuracy_LR_2)]);
disp(' ')

% Use p-values=0 in NB feature selection
pval(:,4) = pval(:,2) == 0;
NB_feature3 = pval(pval(:,4)==1,1);
X_train_NB3 = X_train(:,NB_feature3);

rng(0);
K = 10;
cv = cvpartition(size(y_train, 1), 'KFold', K);

TE_NB3 = zeros(K,1);
F1_NB3 = zeros(K,1);
Precision_NB3 = zeros(K,1);
Recall_NB3 = zeros(K,1);
Accuracy_NB3 = zeros(K,1);
AUC_NB3 = zeros(K,1);

Cat_index = 1:(length(NB_feature3)-1);

% Start the timer
tic;
for fold = 1:K
    trainIdx = training(cv, fold);
    testIdx = test(cv, fold);
```

```
    X_train_cv = X_train_NB3(trainIdx, :);
    y_train_cv = y_train(trainIdx);

    X_test_cv = X_train_NB3(testIdx, :);
    y_test_cv = y_train(testIdx);

    % Train NB model for each fold
    NB_class =
fitcnb(X_train_cv,y_train_cv,'CategoricalPredictors',Cat_index);

    % Training set error in fold
    y_train_fit = predict(NB_class,X_train_cv);
    TE_NB3(fold) = sum(y_train_fit ~= y_train_cv) / length(y_train_cv);

    % F1 score, Precision, Recall, Accuracy for the test set in fold
    [y_fit, Score, ~] = predict(NB_class,X_test_cv);
    tp = sum((y_fit == 1) & (y_test_cv == 1));
    fp = sum((y_fit == 1) & (y_test_cv == 0));
    tn = sum((y_fit == 0) & (y_test_cv == 0));
    fn = sum((y_fit == 0) & (y_test_cv == 1));

    F1_NB3(fold) = 2*tp / (2*tp + fp + fn);
    Precision_NB3(fold) = tp / (tp + fp);
    Recall_NB3(fold) = tp / (tp + fn);
    Accuracy_NB3(fold) = (tp + tn) / (tp + tn + fp + fn);

    % AUC for the test in fold
    [~,~,~,AUC_NB3(fold)] = perfcurve(y_test_cv,Score(:,2),1);

end
% Stop the timer
Time_NB3 = toc;

disp('Train results in NB model 3');
disp(['Mean training error in NB model 3: ', num2str(mean(TE_NB3))]);
disp(['Mean validation F1 score in NB model 3: ', num2str(mean(F1_NB3))]);
disp(['Mean validation Precision in NB model 3: ',
num2str(mean(Precision_NB3))]);
disp(['Mean validation Recall in NB model 3: ', num2str(mean(Recall_NB3))]);
disp(['Mean validation Accuracy in NB model 3: ',
num2str(mean(Accuracy_NB3))]);
disp(['Mean validation AUC in NB model 3: ', num2str(mean(AUC_NB3))]);
disp(['Training time in NB model 3: ', num2str(Time_NB3)]);
```

```
Train results in LR model 1 with cutoff=0.5
Mean training error in LR model 1: 0.1361
Mean validation F1 score in LR model 1: 0.23918
Mean validation Precision in LR model 1: 0.53887
Mean validation Recall in LR model 1: 0.15376
Mean validation Accuracy in LR model 1: 0.86393

Train results in LR model 2 with cutoff=0.5
Mean training error in LR model 2: 0.13616
```

```
Mean validation F1 score in LR model 2: 0.23835
Mean validation Precision in LR model 2: 0.53457
Mean validation Recall in LR model 2: 0.15342
Mean validation Accuracy in LR model 2: 0.86362

Test results in LR model with cutoff=0.5
F1 score in LR model: 0.24064
Precision in LR model: 0.52513
Recall in LR model: 0.15608
Accuracy in LR model: 0.86225

Train results in NB model 3
Mean training error in NB model 3: 0.18268
Mean validation F1 score in NB model 3: 0.42828
Mean validation Precision in NB model 3: 0.37915
Mean validation Recall in NB model 3: 0.49247
Mean validation Accuracy in NB model 3: 0.81722
Mean validation AUC in NB model 3: 0.80698
Training time in NB model 3: 1.6897
```

Acknowledgement

The Matlab code used in this project is referenced from MathWorks official websites and course tutorials

Published with MATLAB® R2023b