Predicting the Diabetes of Patients

Data variables Preditctors

- Number of times pregnant (preg)
- Plasma glucose concentration a 2 hours in an oral glucose tolerance test (plas)
- Diastolic blood pressure in mm Hg (pres)
- Triceps skin fold thickness in mm (skin)
- 2-Hour serum insulin in mu U/ml (insu)
- Body mass index measured as weight in kg/(height in m)^2 (mass)
- Diabetes pedigree function (pedi)
- Age in years (age)
- Output is 0 or 1(0 for no diabetes and 1 for diabetes

Download Data https://www.kaggle.com/uciml/pima-indians-diabetes-database/data

Read Table

```
clear,clc,close all
aa=readtable("diabetes.csv");
```

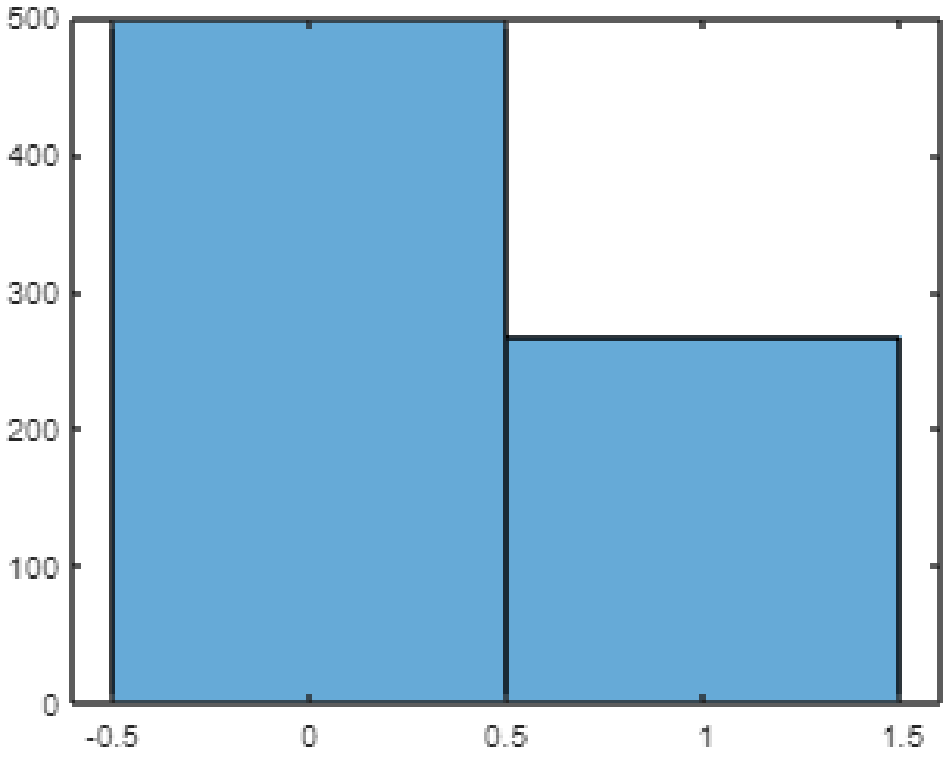
Find if there is missing data

```
idx=ismissing(aa);
sum(idx(:))
```

```
ans = 0
```

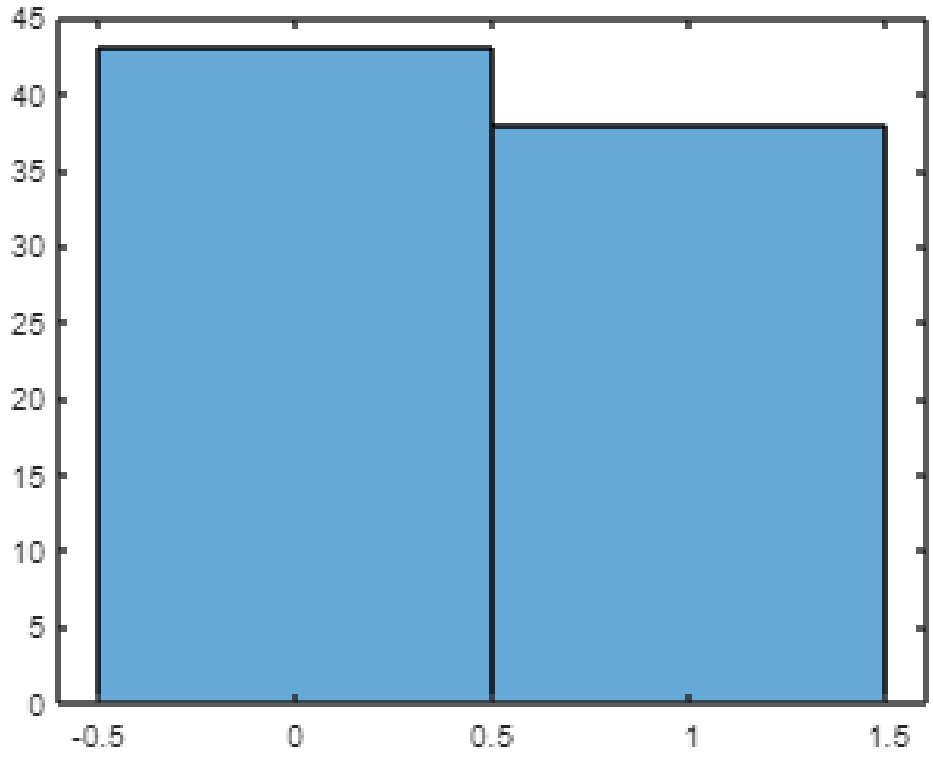
Count of diabetes and non diabetes Patients

```
histogram(aa.Outcome);
```



How many Patients have diabetes who are above 50 years old.

```
idx=find(aa.Age>50);
histogram(aa{idx,end});
```



Some Preprocessing

Convert output in category

```
aa.Outcome=categorical(aa.Outcome,[0,1],{'no','yes'})
```

aa = 768×9 table

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
1	6	148	72	35	0	33.6000
2	1	85	66	29	0	26.6000
3	8	183	64	0	0	23.3000
4	1	89	66	23	94	28.1000
5	0	137	40	35	168	43.1000
6	5	116	74	0	0	25.6000
7	3	78	50	32	88	31
8	10	115	0	0	0	35.3000
9	2	197	70	45	543	30.5000

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
10	8	125	96	0	0	0
11	4	110	92	0	0	37.6000
12	10	168	74	0	0	38
13	10	139	80	0	0	27.1000
14	1	189	60	23	846	30.1000

⋮

Average value of diabetes and non diabetes patients

```
varfun(@mean,aa,"InputVariables","Age",'GroupingVariables',"Outcome")
```

ans = 2×3 table

	Outcome	GroupCount	mean_Age
1	no	500	31.1900
2	yes	268	37.0672

Equalize Data

Count diabetes and non diabetes

```
sum(aa.Outcome=='no') %count of non diabetes
```

```
ans = 500
```

```
sum(aa.Outcome=='yes') % count of diabetes
```

```
ans = 268
```

Extract indices and apply random permutation

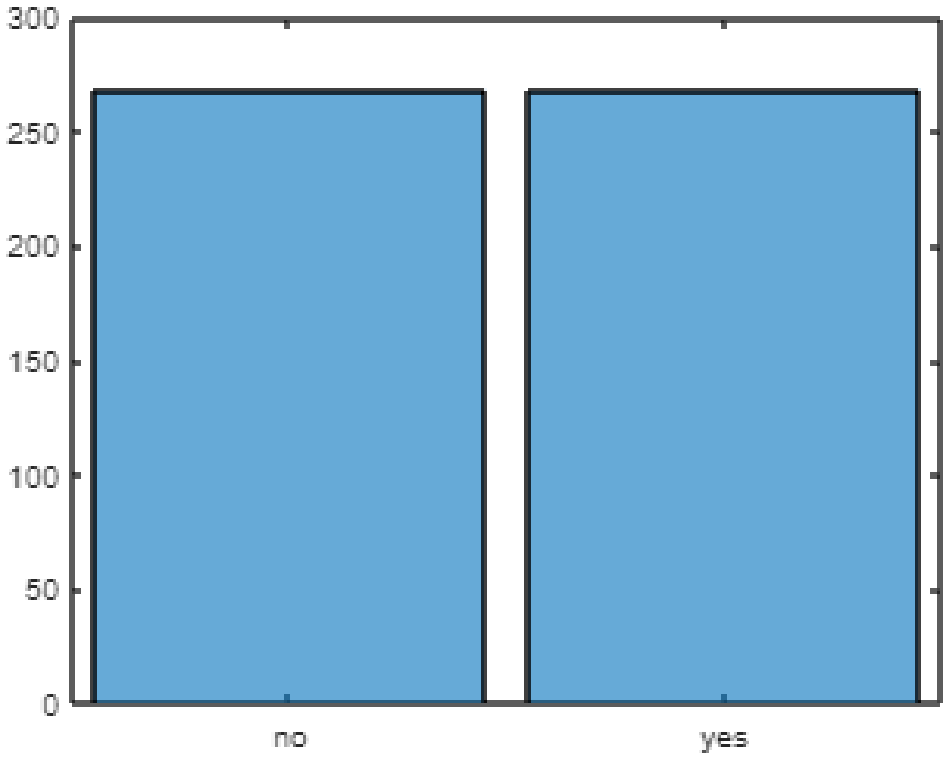
```
idxDiab=find(aa.Outcome=='yes'); % numeric index of diabetes
idxNonDiab=find(aa.Outcome=='no'); % numeric index of non diabetes;
idx=randperm(500,268); % random 268 index extraction of non diabetes
idxx=idxNonDiab(idx);
```

Concatenate both table

```
newTab=[aa(idxDiab,:);aa(idxx,:)]; % concatenate both table
```

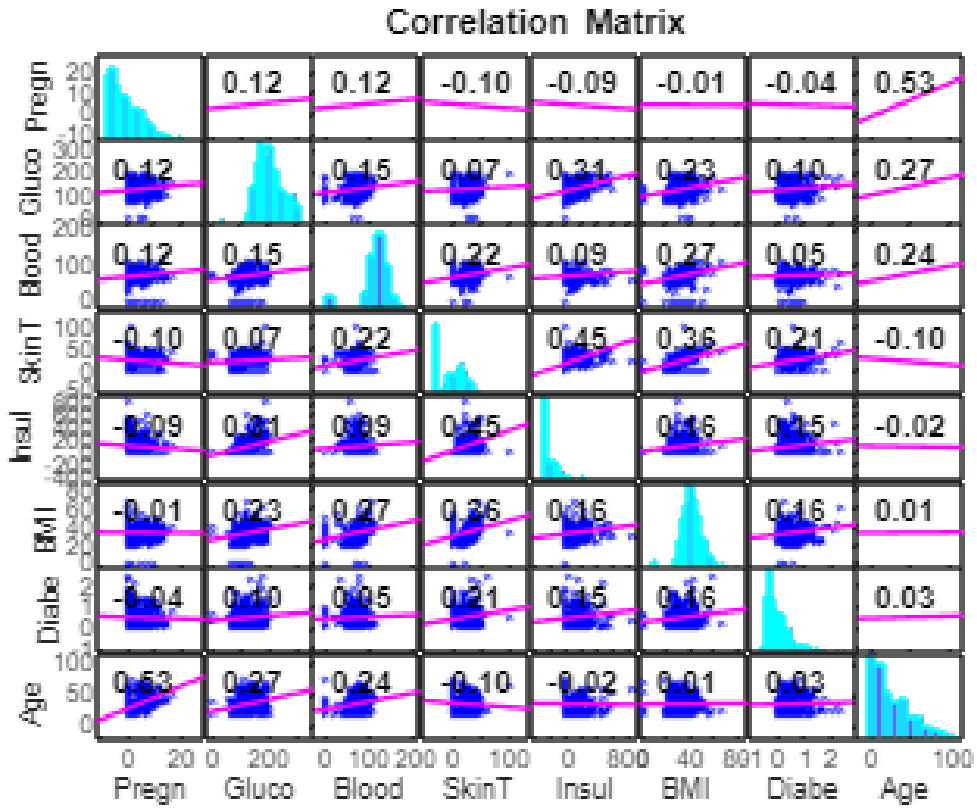
Histogram of modified data

```
figure,
histogram(newTab.Outcome)
```



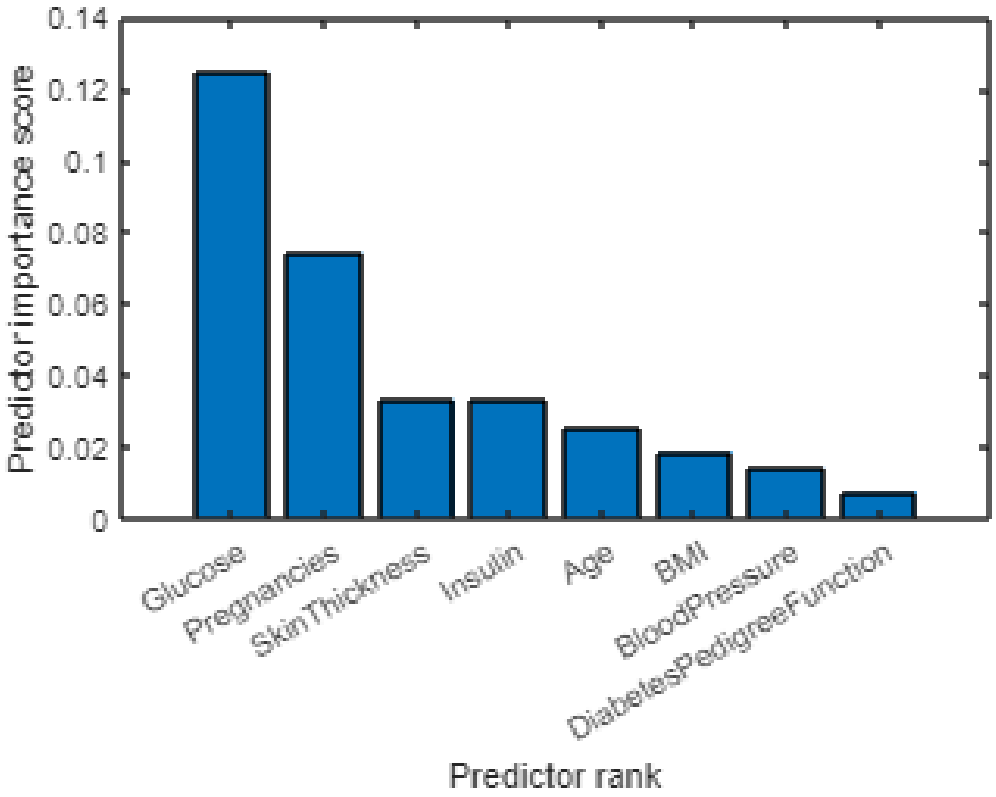
Correlation plots between variables

```
figure,
corrplot(newTab(:,1:end-1))
```



Rank features for classification using minimum redundancy maximum relevance (MRMR) algorithm

```
[idx,scores] = fscmrmr(newTab,'Outcome');
bar(scores(idx))
xlabel('Predictor rank')
ylabel('Predictor importance score')
xticklabels(strrep(aa.Properties.VariableNames(idx),'_','\_'))
```



https://www.mathworks.com/help/stats/feature-selection.html

https://www.mathworks.com/help/deeplearning/ug/extract-image-features-using-pretrained-network.html

Training and Predection

```
trainedClassifier=trainClassifier(newTab)
```

```
trainedClassifier = struct with fields:
           predictFcn: @(x)svmPredictFcn(featureSelectionFcn(predictorExtractionFcn(x)))
    RequiredVariables: {'Age'  'BMI'  'BloodPressure'  'DiabetesPedigreeFunction'  'Glucose'  'Insulin'  'Pregnancie
    ClassificationSVM: [1×1 ClassificationSVM]
                About: 'This struct is a trained model exported from Classification Learner R2020a.'
          HowToPredict: 'To make predictions on a new table, T, use: ↵  yfit = c.predictFcn(T) ↵replacing 'c' with
```

```
trainedClassifier.predictFcn(newTab(1,1:end-1))
```

```
ans = categorical
     yes
```

```
function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
```

```
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  Output:
%      trainedClassifier: A struct containing the trained classifier. The
%       struct contains various fields with information about the trained
%       classifier.
%
%      trainedClassifier.predictFcn: A function to make predictions on new
%       data.
%
%      validationAccuracy: A double containing the accuracy in percent. In
%       the app, the History list displays this overall accuracy score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 23-Jun-2020 13:38:15


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
predictors = inputTable(:, predictorNames);
response = inputTable.Outcome;
isCategoricalPredictor = [false, false, false, false, false, false, false, false];

% Data transformation: Select subset of the features
% This code selects the same subset of features as were used in the app.
includedPredictorNames = predictors.Properties.VariableNames([false true false false true true
predictors = predictors(:,includedPredictorNames);
isCategoricalPredictor = isCategoricalPredictor([false true false false true true true true]);

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
```

```
classificationSVM = fitcsvm(...
    predictors, ...
    response, ...
    'KernelFunction', 'gaussian', ...
    'PolynomialOrder', [], ...
    'KernelScale', 11, ...
    'BoxConstraint', 1, ...
    'Standardize', true, ...
    'ClassNames', categorical({'no'; 'yes'}));

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
featureSelectionFcn = @(x) x(:,includedPredictorNames);
svmPredictFcn = @(x) predict(classificationSVM, x);
trainedClassifier.predictFcn = @(x) svmPredictFcn(featureSelectionFcn(predictorExtractionFcn(x)

% Add additional fields to the result struct
trainedClassifier.RequiredVariables = {'Age', 'BMI', 'BloodPressure', 'DiabetesPedigreeFunction
trainedClassifier.ClassificationSVM = classificationSVM;
trainedClassifier.About = 'This struct is a trained model exported from Classification Learner
trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use: \n  yfit

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
predictors = inputTable(:, predictorNames);
response = inputTable.Outcome;
isCategoricalPredictor = [false, false, false, false, false, false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationSVM, 'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');

end
```