BME 499 - Final Project

David Chang and Mahmoud Komaiha

Due: April 24th, 2020

Task 1 - Processing Data

Reading in the table

```
cirrhosis = readtable('cirrhosis.csv');
% take out time temporarily, will add it back at end later
time = cirrhosis.time;
cirrhosis.time = [];
expandFigSize = [0 0 1000 700];
```

Add the extra variables

```
% Log some key variables
cirrhosis.logalb = log(cirrhosis.albumin);
cirrhosis.logbil = log(cirrhosis.bili);
cirrhosis.logpro = log(cirrhosis.protime);

% Rearrange data to put time as last variable
cirrhosis.time = time;
```

Store names and desc for later

```
% Make some variable descriptions
cirrhosis_desc = {'age','albumin','alkaline phosphatase','ascites','serum bilirubin','s
    'edema treatment','hepatomegaly','platelets','prothrombin time','sex','aspartate ar
    'stage','censoring','treatment','trigycerides','urine copper','log albumin','log se
    'log prothrombin time','time'};
% Label the Categorial Variables
categVar = {'ascites','edema','hepato','sex','spiders','stage','trt'};
% Subset key variables w/o log
subset = {'age','albumin','bili','alkphos','chol','ast','protime',...
    'platelet','spiders','hepato','ascites','edema'};
% Subset key variables w/ generated log features
subset_log = {'age','albumin','bili','alkphos','chol','ast','protime','platelet',...
    'spiders','hepato','ascites','edema','logalb','logbil','logpro'};
cirrhosis.Properties.VariableDescriptions = cirrhosis_desc;
cirrhosis_names = cirrhosis.Properties.VariableNames;
head(cirrhosis)
```

```
ans = 8×22 table
```

...

	age	albumin	alkphos	ascites	bili	chol	edema	hepato
1	58.7652	2.6000	1718	1	14.5000	261	1.0000	1
2	56.4463	4.1400	7.3948e+03	0	1.1000	302	0	1
3	70.0726	3.4800	516	0	1.4000	176	0.5000	0

	age	albumin	alkphos	ascites	bili	chol	edema	hepato
4	54.7406	2.5400	6.1218e+03	0	1.8000	244	0.5000	1
5	38.1054	3.5300	671	0	3.4000	279	0	1
6	66.2587	3.9800	944	0	0.8000	248	0	1
7	55.5346	4.0900	824	0	1.0000	322	0	1
8	53.0568	4.0000	4.6512e+03	0	0.3000	280	0	0

Task 2 - Exploratory Phase

Summarize data

```
cirr_summary = summary(cirrhosis);
N = numel(cirrhosis.Properties.VariableNames);
temp_table = struct('Min',zeros(N,1),'Max',zeros(N,1),'Range',zeros(N,1),...
                    'Median',zeros(N,1),'Mean',zeros(N,1),'NumMissing',zeros(N,1));
for i=1:N
    temp_struct              = cirr_summary.(cirrhosis_names{i});
    temp_table.Min(i)        = temp_struct.Min;
    temp_table.Max(i)        = temp_struct.Max;
    temp_table.Mean(i)       = mean(cirrhosis.(cirrhosis_names{i}));
    temp_table.Median(i)     = temp_struct.Median;
    temp_table.Range(i)      = temp_struct.Max - temp_struct.Min;
    temp_table.NumMissing(i) = temp_struct.NumMissing;
end
sumTable = table(cirrhosis_names', ...
      cirrhosis_desc', ...
      temp_table.Min, ...
      temp_table.Max, ...
      temp_table.Mean, ...
      temp_table.Median, ...
      temp_table.Range, ...
      temp_table.NumMissing, ...
      'VariableNames',{'VarName','VarDesc','Min','Max','Mean','Median','Range','NumMiss
```

sumTable = 22×8 table

	VarName	VarDesc	Min	Max	Mean	Median	Range	NumMissing
1	'age'	'age'	26.2779	78.4394	50.0190	49.7947	52.1615	0
2	'albumin'	'albumin'	1.9600	4.6400	3.5200	3.5500	2.6800	0
3	'alkphos'	'alkaline ph...	289.0000	1.3862e+04	1.9827e+03	1259	1.3573e+04	0
4	'ascites'	'ascites'	0	1	0.0769	0	1	0
5	'bili'	'serum bilir...	0.3000	28	3.2561	1.3500	27.7000	0
6	'chol'	'serum chole...	120.0000	1775	NaN	309.5000	1655	28
7	'edema'	'edema treat...	0	1	0.1106	0	1	0
8	'hepato'	'hepatomegaly'	0	1	0.5128	1	1	0

	VarName	VarDesc	Min	Max	Mean	Median	Range	NumMissing
9	'platelet'	'platelets'	62.0000	563	NaN	257	501	4
10	'protime'	'prothrombin...	9.0000	17.1000	10.7256	10.6000	8.1000	0
11	'sex'	'sex'	0	1	0.8846	1	1	0
12	'ast'	'aspartate a...	26.3500	457.2500	122.5563	114.7000	430.9000	0
13	'spiders'	'spiders'	0	1	0.2885	0	1	0
14	'stage'	'stage'	1.0000	4	3.0321	3	3	0

⋮

Clean up and setup data for unupervised and supervised learning

Time and status are the two dependent variables. This cleanup is done to make separate datasets so that the two dependent variables are not in the same dataset.

cirrhosis = original dataset with missing values removed

cirrhosis_log = log(time) and status is removed - for linear regression, lasso, stepwise and RF_regression models

cirrhosis_status = time is removed - for PCA, k-means clustering, classification, so logistic regression and RF_classification models

```
nMissingRows = sum(any(ismissing(cirrhosis),2))
```

```
nMissingRows = 36
```

```
cirrhosis = rmmissing(cirrhosis);
cirrhosis_log = cirrhosis;
cirrhosis_log.time = log(cirrhosis_log.time);
cirrhosis_log = removevars(cirrhosis_log,{'status'});
cirrhosis_log_names = cirrhosis_log.Properties.VariableNames;
cirrhosis_status = cirrhosis(:,1:end-1);
status = cirrhosis_status.status;
cirrhosis_status.status = [];
cirrhosis_status.status = status > 1;
cirrhosis_status_names = cirrhosis_status.Properties.VariableNames;
```

Standardize Data with Z score

Toggle on and off with comments

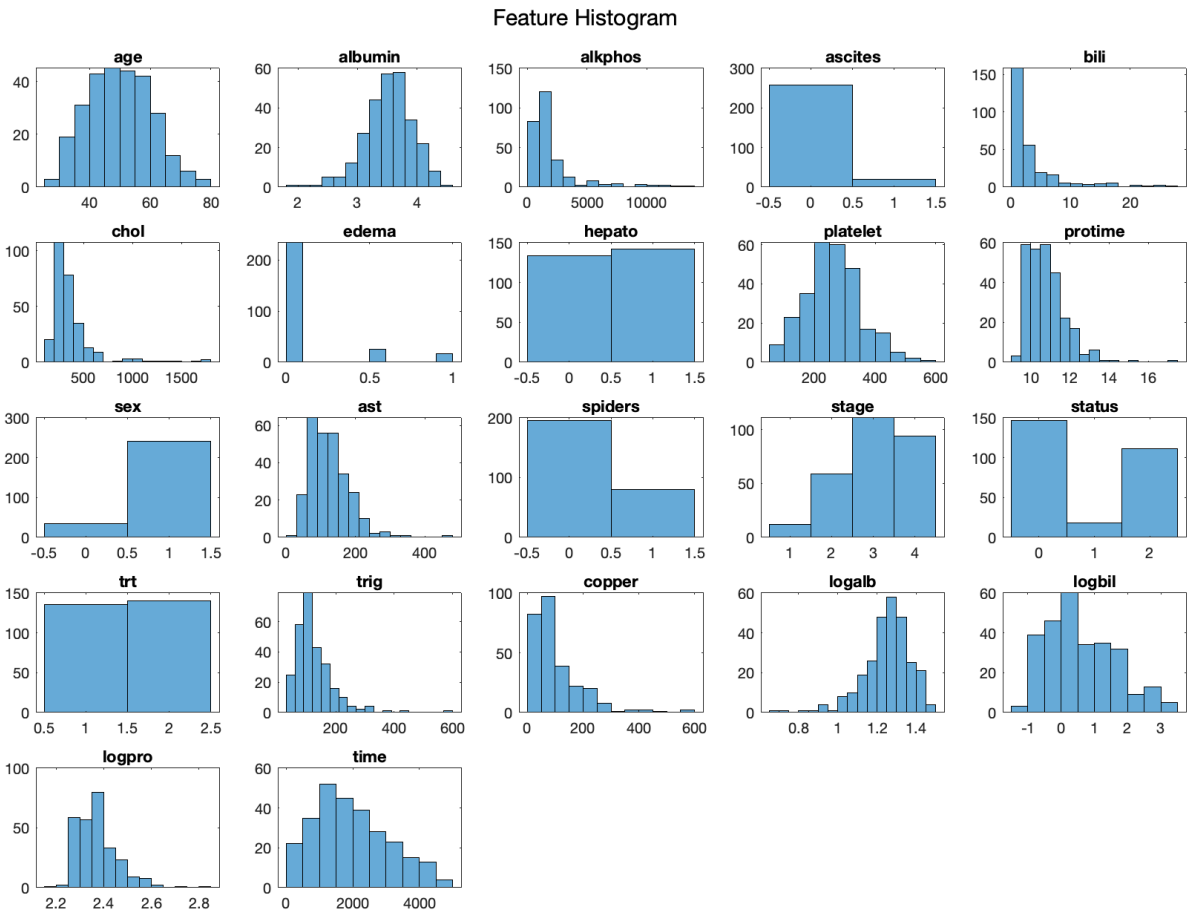
Rerun and save different version of output when doing this

```
% status = cirrhosis_status.status;
% cirrhosis = zscore(table2array(cirrhosis));
% cirrhosis = array2table(cirrhosis,"VariableNames",cirrhosis_names);
% cirrhosis_log = zscore(table2array(cirrhosis_log));
% cirrhosis_log = array2table(cirrhosis_log,"VariableNames",cirrhosis_log_names);
% array2table(cirrhosis_log,"VariableNames",cirrhosis_log_names);
% cirrhosis_status = zscore(table2array(cirrhosis_status));
```

```
% cirrhosis_status = array2table(cirrhosis_status,"VariableNames",cirrhosis_status_name
% cirrhosis_status.status = status;
```

Feature Histograms

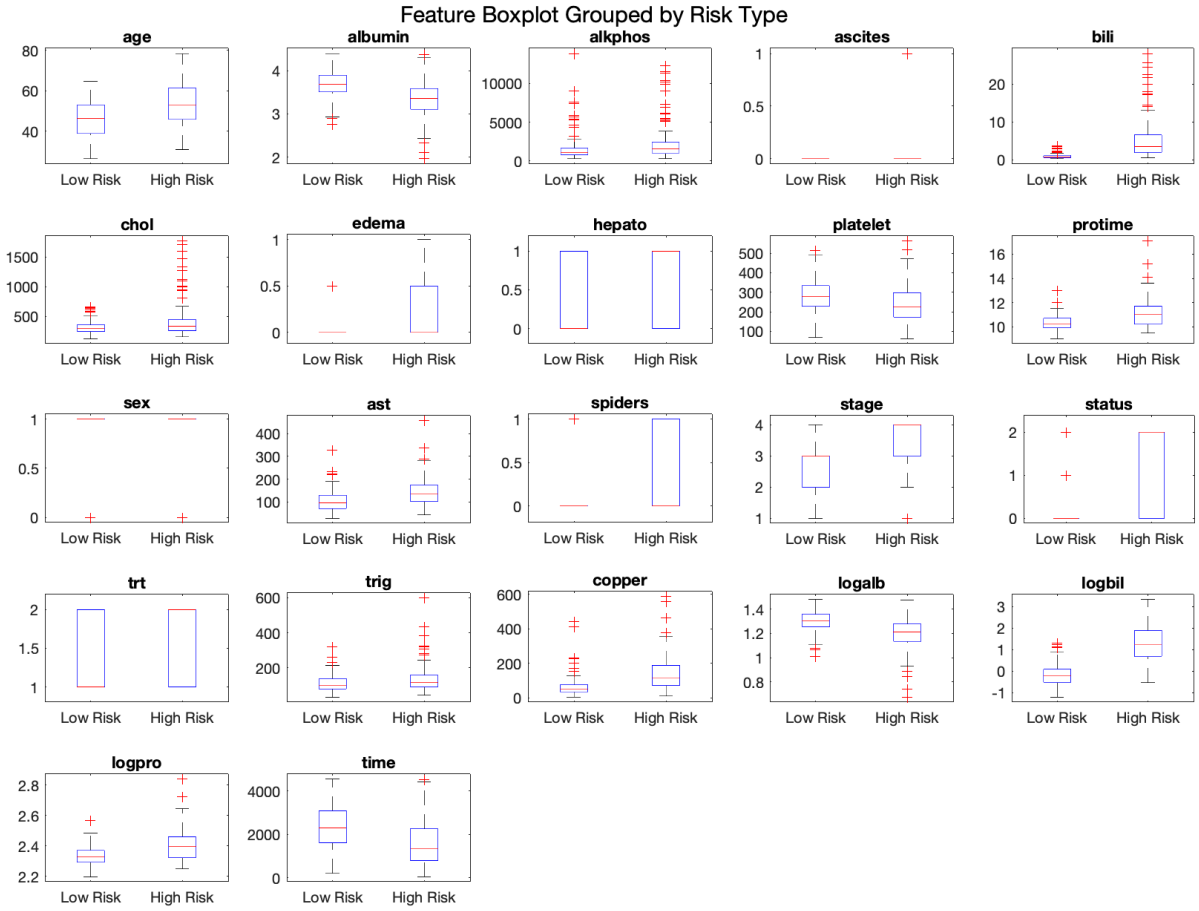
```
f = figure();
set(f, 'Position', expandFigSize)
for i=1:size(cirrhosis,2)
    subplot(5,5,i)
    histogram(cirrhosis.(i))
    title(cirrhosis_names{i})
end
sgtitle("Feature Histogram")
```



Boxplot

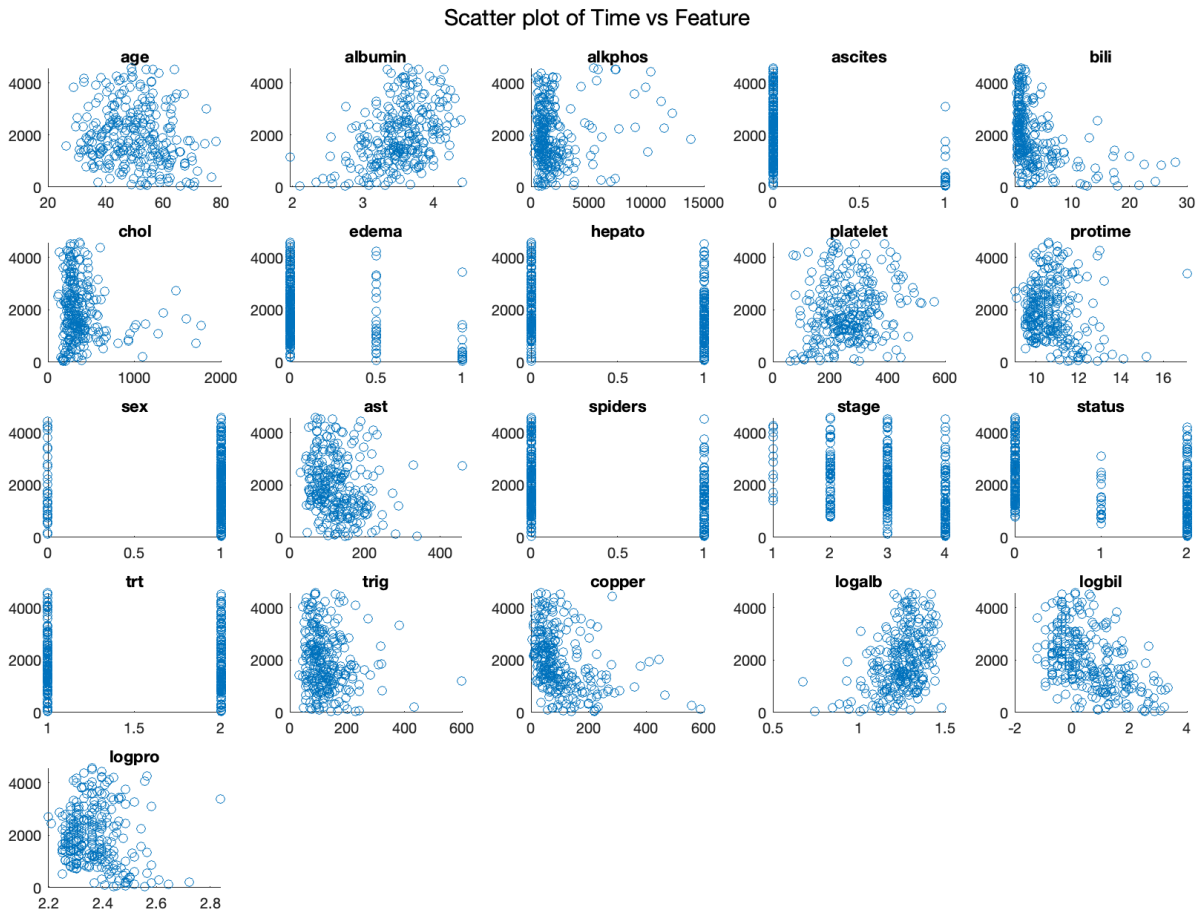
```
% Risk score computation as described in Cirrhosis paper
mayoFinalR = 0.871*cirrhosis.logbil ...
   - 2.53*cirrhosis.logalb ...
   + 0.039*cirrhosis.age ...
   + 2.38*cirrhosis.logpro ...
   + 0.859*cirrhosis.edema;
```

```
% Scatterplot subset by risk high risk -> decreased survivability
f = figure();
set(f, 'Position', expandFigSize)
for i=1:size(cirrhosis,2)
    subplot(5,5,i)
    boxplot(cirrhosis.(i),mayoFinalR >= median(mayoFinalR),'Labels',{'Low Risk','High
    title(cirrhosis_names{i})
end
sgtitle("Feature Boxplot Grouped by Risk Type")
```



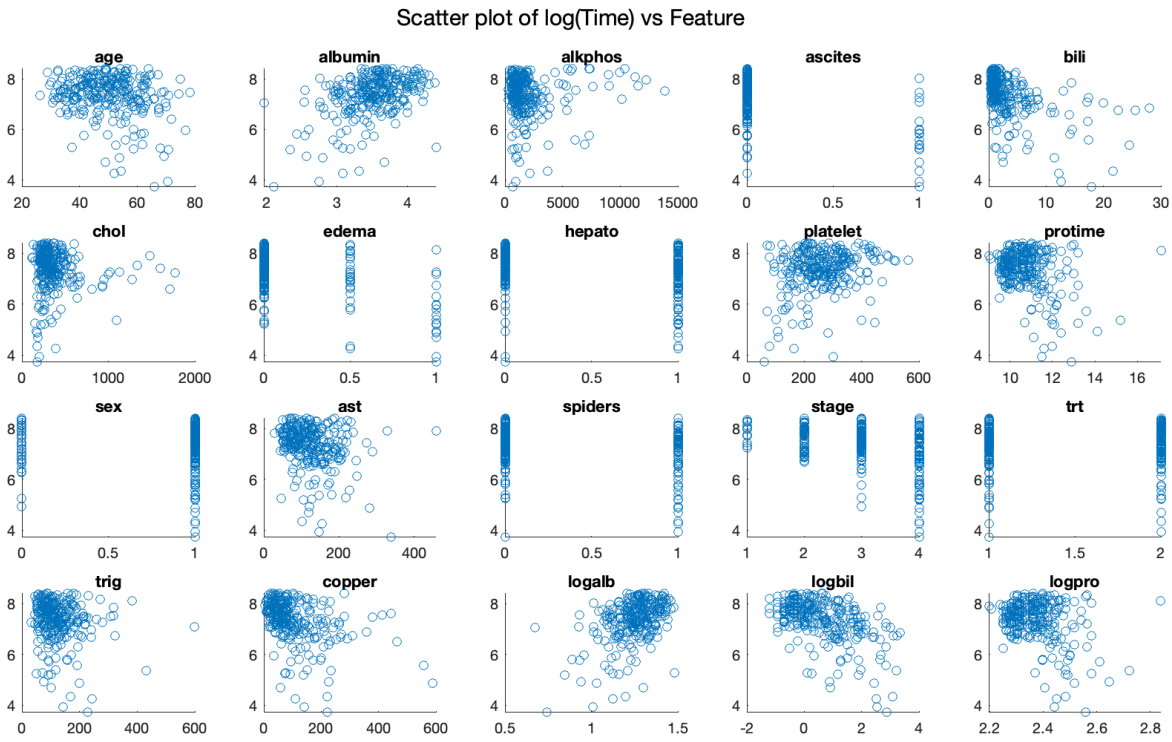
Scatter time vs. feature

```
f = figure();
set(f, 'Position', expandFigSize)
for i=1:size(cirrhosis,2)-1
    subplot(5,5,i)
    scatter(cirrhosis.(i),cirrhosis.time)
    title(cirrhosis_names{i})
end
sgtitle("Scatter plot of Time vs Feature")
```



Scatter log time vs. feature

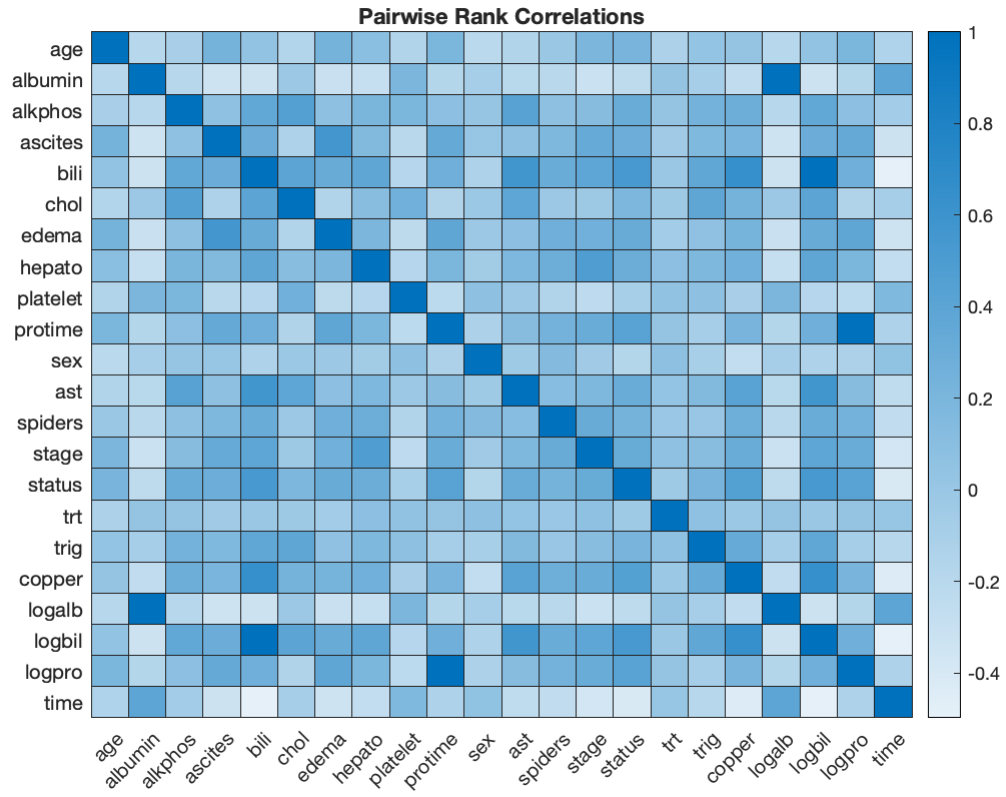
```
f = figure();
set(f, 'Position', expandFigSize)
for i=1:size(cirrhosis_log,2)-1
    subplot(5,5,i)
    scatter(cirrhosis_log.(i),cirrhosis_log.time)
    title(cirrhosis_log_names{i})
end
sgtitle("Scatter plot of log(Time) vs Feature")
```



Task 3 - Statistical Analysis

Pairwise spearman rank correlations

```
figure();
[R_rank,P_rank] = corr(table2array(cirrhosis),'Type','Spearman','rows','complete');
r_rank_map = heatmap(R_rank);
title('Pairwise Rank Correlations')
r_rank_map.XDisplayLabels = cirrhosis_names;
r_rank_map.YDisplayLabels = cirrhosis_names;
```




```
% Features sorted by Correlation
absR_rank_time = abs(R_rank(1:end-1,end));
[absR_rank_time,I] = sort(absR_rank_time,'descend');
P_rank_time = P_rank(1:end-1,end);
cirrhosis_names_table = cirrhosis_names(1:end-1);
cirrhosis_names_table = cirrhosis_names_table(I);
rankTable = table(cirrhosis_names_table',absR_rank_time,P_rank_time(I),...
    'VariableNames',{'Feature','Correlation','pValue'})
```

rankTable = 21×3 table

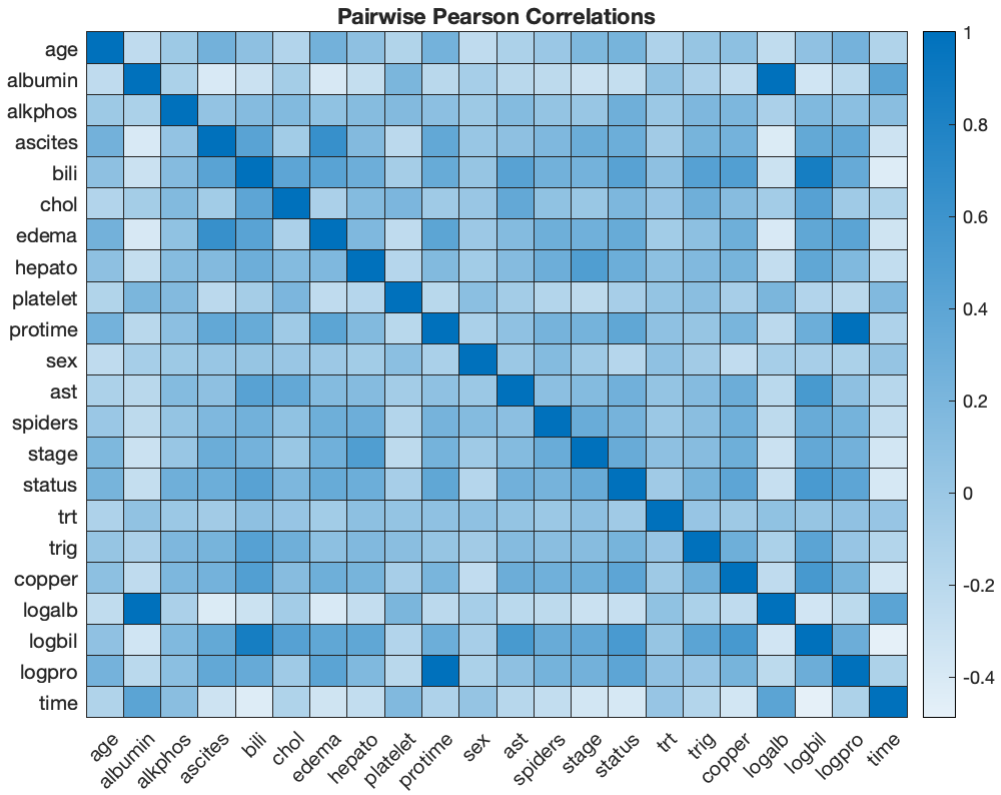
	Feature	Correlation	pValue
1	'bili'	0.4982	0.0000
2	'logbil'	0.4982	0.0000
3	'copper'	0.4359	0.0000
4	'status'	0.4059	0.0000
5	'albumin'	0.3910	0.0000
6	'logalb'	0.3910	0.0000
7	'stage'	0.3716	0.0000
8	'edema'	0.3375	0.0000
9	'ascites'	0.3337	0.0000
10	'hepato'	0.2666	0.0000

	Feature	Correlation	pValue
11	'spiders'	0.2567	0.0000
12	'ast'	0.2504	0.0000
13	'trig'	0.1916	0.0014
14	'platelet'	0.1622	0.0069

⋮

Pairwise pearson correlations

```
figure();
[R_pear,P_pear] = corr(table2array(cirrhosis),'Type','Pearson','rows','complete');
r_pear_map = heatmap(R_pear);
title('Pairwise Pearson Correlations')
r_pear_map.XDisplayLabels = cirrhosis_names;
r_pear_map.YDisplayLabels = cirrhosis_names;
```




```
% Features sorted by Correlation
absR_pear_time = abs(R_pear(1:end-1,end));
[absR_pear_time,I] = sort(absR_pear_time,'descend');
P_rank_time = P_pear(1:end-1,end);
cirrhosis_names_table = cirrhosis_names(1:end-1);
cirrhosis_names_table = cirrhosis_names_table(I);
pearsonTable = table(cirrhosis_names_table',absR_pear_time,P_rank_time(I),...
    'VariableNames',{'Feature','Correlation','pValue'})
```

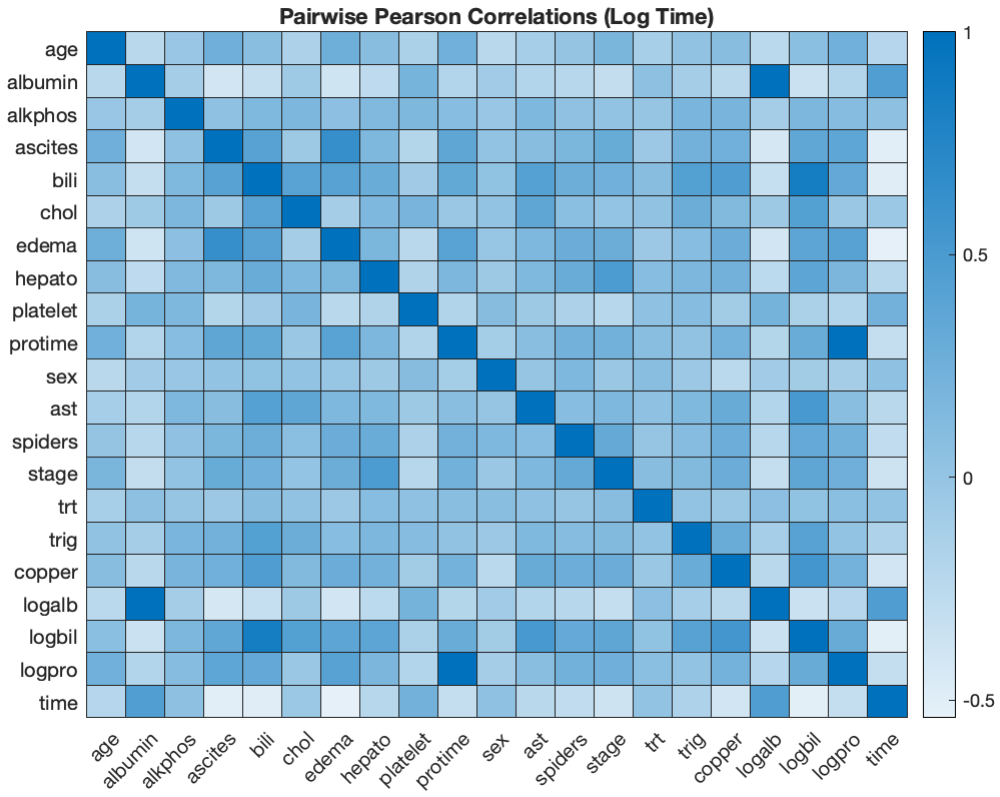
pearsonTable = 21×3 table

	Feature	Correlation	pValue
1	'logbil'	0.4882	0.0000
2	'bili'	0.4303	0.0000
3	'logalb'	0.4026	0.0000
4	'albumin'	0.4019	0.0000
5	'status'	0.3845	0.0000
6	'stage'	0.3639	0.0000
7	'copper'	0.3614	0.0000
8	'edema'	0.3471	0.0000
9	'ascites'	0.3294	0.0000
10	'hepato'	0.2610	0.0000
11	'spiders'	0.2606	0.0000
12	'ast'	0.1911	0.0014
13	'trig'	0.1639	0.0064
14	'platelet'	0.1591	0.0081

⋮

Pairwise pearson correlations with log time

```
figure();
[R_pear_log,P_pear_log] = corr(table2array(cirrhosis_log),'Type','Pearson','rows','com
r_pear_log_map = heatmap(R_pear_log);
title('Pairwise Pearson Correlations (Log Time)')
r_pear_log_map.XDisplayLabels = cirrhosis_log_names;
r_pear_log_map.YDisplayLabels = cirrhosis_log_names;
```




```
absR_pear_log_time = abs(R_pear_log(1:end-1,end));
[absR_pear_log_time,I] = sort(absR_pear_log_time,'descend');
P_rank_time = P_pear_log(1:end-1,end);
cirrhosis_names_table = cirrhosis_log_names(1:end-1);
cirrhosis_names_table = cirrhosis_names_table(I);
pearsonLogTable = table(cirrhosis_names_table',absR_pear_log_time,P_rank_time(I),...
    'VariableNames',{'Feature','Correlation','pValue'})
```

pearsonLogTable = 20×3 table

	Feature	Correlation	pValue
1	'edema'	0.5402	2.5921e-22
2	'logbil'	0.5242	6.9829e-21
3	'ascites'	0.5133	5.9590e-20
4	'bili'	0.5025	4.5844e-19
5	'logalb'	0.4600	7.4188e-16
6	'albumin'	0.4471	5.6923e-15
7	'copper'	0.4019	3.8594e-12
8	'stage'	0.3815	5.4442e-11
9	'logpro'	0.3164	7.8083e-08
10	'protime'	0.3150	9.0153e-08
11	'spiders'	0.2934	6.9635e-07

	Feature	Correlation	pValue
12	'ast'	0.2402	5.5398e-05
13	'platelet'	0.2279	1.3423e-04
14	'hepato'	0.2221	1.9906e-04

⋮

Task 4 - Unsupervised Learning

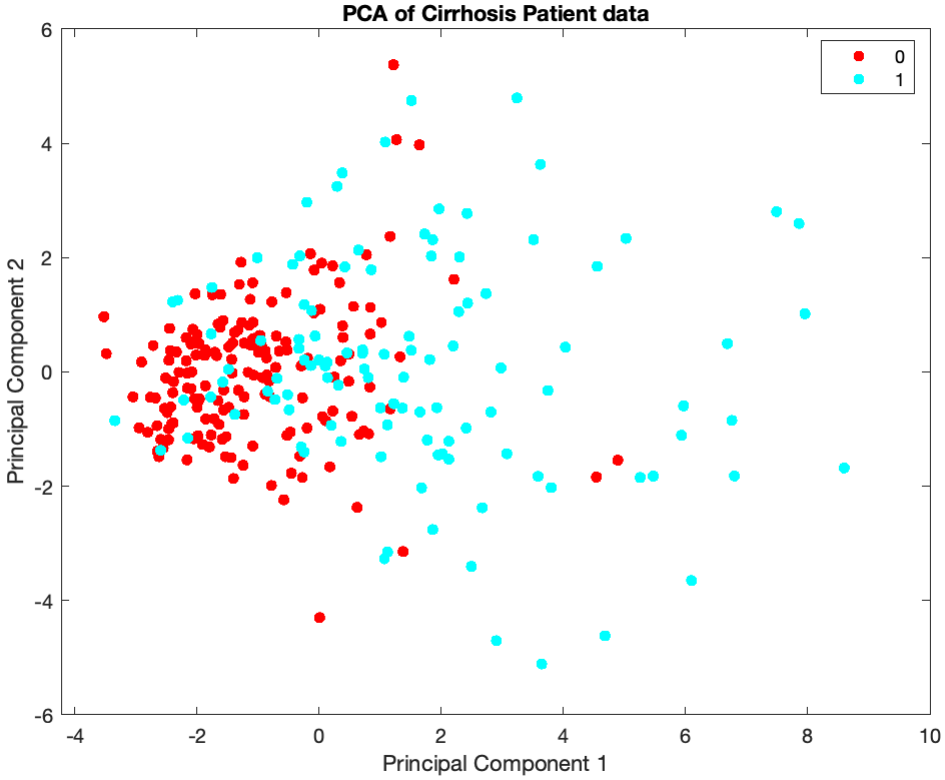
We did PCA and K-means clustering. We evaluated how the data groups with respect to the dependent categorical variable **status**.

PCA

```
% Remove status and try to find pca that detemines best features to predict status
cirrhosis_status_rm = cirrhosis_status;
cirrhosis_status_rm.status = [];
[coef, score, ~, ~, explained] = pca(zscore(table2array(cirrhosis_status_rm)));
% [coef, score, ~, ~, explained] = pca(table2array(cirrhosis_status_rm));  %Use this i
% %Variance explained by PC1 and PC2
explained(1:2)
```

```
ans = 2×1
   26.0297
   11.6807
```

```
% Plot PCA and color by status
figure();
gscatter(score(:,1), score(:,2), cirrhosis_status.status)
xlabel('Principal Component 1')
ylabel('Principal Component 2')
title('PCA of Cirrhosis Patient data')
```




```
PC1_coeff = coef(:,1);
PC2_coeff = coef(:,2);
% Table of PCA Coefficients
T = table(cirrhosis_status_rm.Properties.VariableNames', PC1_coeff, PC2_coeff)
```

T = 20×3 table

	Var1	PC1_coeff	PC2_coeff
1	'age'	0.1256	-0.2995
2	'albumin'	-0.2856	0.1006
3	'alkphos'	0.0900	0.1759
4	'ascites'	0.2788	-0.1801
5	'bili'	0.3298	0.2530
6	'chol'	0.0963	0.4533
7	'edema'	0.2865	-0.2080
8	'hepato'	0.2116	0.0657
9	'platelet'	-0.1308	0.2553
10	'protime'	0.2544	-0.2676
11	'sex'	-0.0329	0.0521
12	'ast'	0.1802	0.3177
13	'spiders'	0.1987	-0.0039

	Var1	PC1_coeff	PC2_coeff
14	'stage'	0.2399	-0.0663

⋮

```
% Table of PCA sorted by PC1
T_PC1 = sortrows(T,'PC1_coeff','descend')
```

T_PC1 = 20×3 table

	Var1	PC1_coeff	PC2_coeff
1	'logbil'	0.3459	0.2797
2	'bili'	0.3298	0.2530
3	'edema'	0.2865	-0.2080
4	'ascites'	0.2788	-0.1801
5	'logpro'	0.2589	-0.2677
6	'protime'	0.2544	-0.2676
7	'copper'	0.2531	0.1503
8	'stage'	0.2399	-0.0663
9	'hepato'	0.2116	0.0657
10	'spiders'	0.1987	-0.0039
11	'ast'	0.1802	0.3177
12	'trig'	0.1579	0.2934
13	'age'	0.1256	-0.2995
14	'chol'	0.0963	0.4533

⋮

```
% Table of PCA sorted by PC2
T_PC2 = sortrows(T,'PC2_coeff','descend')
```

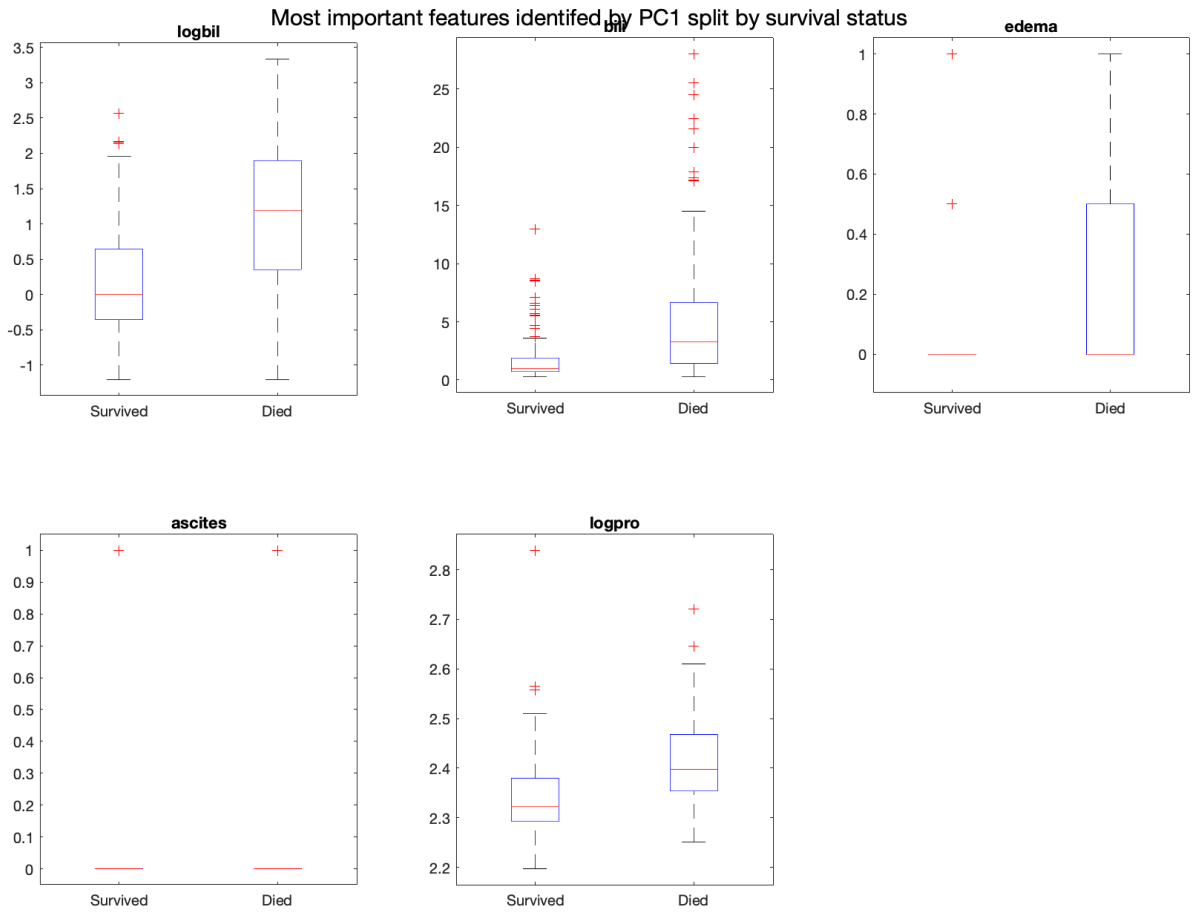
T_PC2 = 20×3 table

	Var1	PC1_coeff	PC2_coeff
1	'chol'	0.0963	0.4533
2	'ast'	0.1802	0.3177
3	'trig'	0.1579	0.2934
4	'logbil'	0.3459	0.2797
5	'platelet'	-0.1308	0.2553
6	'bili'	0.3298	0.2530
7	'alkphos'	0.0900	0.1759
8	'copper'	0.2531	0.1503

	Var1	PC1_coeff	PC2_coeff
9	'logalb'	-0.2895	0.1048
10	'albumin'	-0.2856	0.1006
11	'trt'	0.0060	0.0729
12	'hepato'	0.2116	0.0657
13	'sex'	-0.0329	0.0521
14	'spiders'	0.1987	-0.0039

⋮

```
%Box plots exploring top 5 features recognized as having most weight from from PC1
%Features split on status
f = figure();
set(f, 'Position', expandFigSize)
sgtitle('Most important features identifed by PC1 split by survival status')
features = T_PC1.Var1(1:5);
features_analyzed = cell(5,2);
features_analyzed(:,1) = features;
for i=1:length(features)
    subplot(2,3,i)
    boxplot(cirrhosis_status.(features{i}),cirrhosis_status.status, 'Labels',{'Survive
    title(features{i})
    [h,p] = ttest2(cirrhosis_status.(features{i})(cirrhosis_status.status == 0),cirrho
    features_analyzed{i,2} = p;
end
```




```
T_PC1_analyzed = cell2table(features_analyzed);
T_PC1_analyzed.Properties.VariableNames = {'Features','P values'}
```

T_PC1_analyzed = 5×2 table

	Features	P values
1	'logbil'	7.4316e-18
2	'bili'	3.5816e-13
3	'edema'	1.5362e-08
4	'ascites'	3.0548e-07
5	'logpro'	3.8126e-12

K-Means Clustering

```
figure();
kValues = 2:10;
n       = length(kValues);
s_score = zeros(n,1);

for i = 1:n
```

```
    idx = kmeans(table2array(cirrhosis_status),kValues(i)); % k-means clustering
    s = silhouette(table2array(cirrhosis_status),idx);      % silhouette values
    s_score(i) = mean(s);                                  % silhouette score
end
table(kValues',s_score)
```

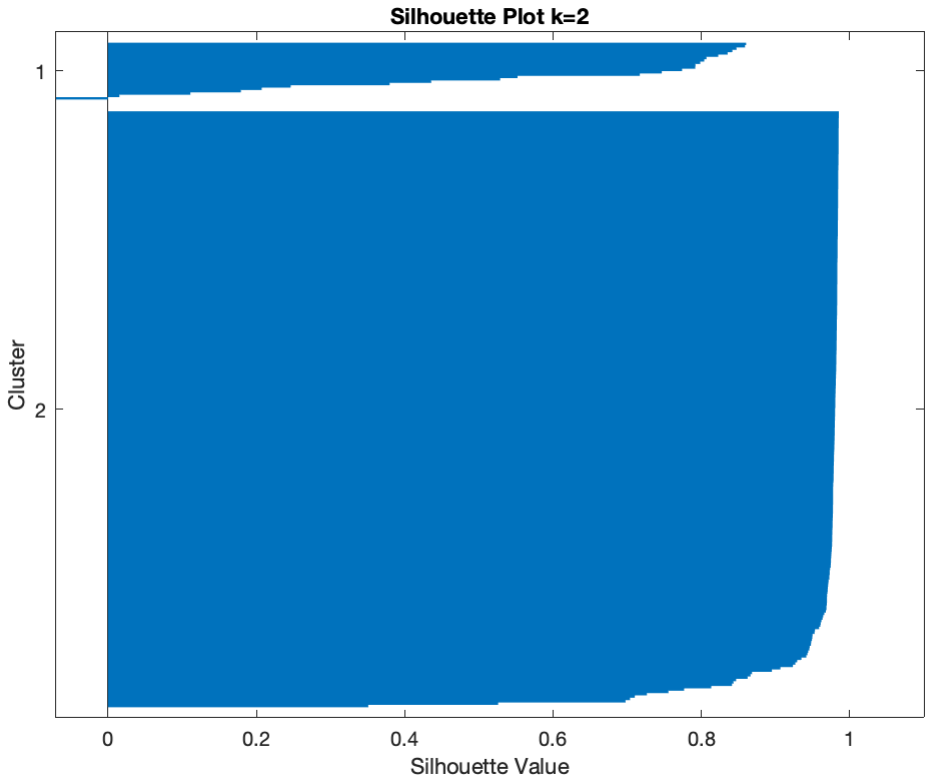
ans = 9×2 table

	Var1	s_score
1	2	0.9274
2	3	0.8810
3	4	0.7412
4	5	0.7236
5	6	0.6846
6	7	0.6478
7	8	0.6429
8	9	0.5772
9	10	0.5978

```
k = find(s_score==max(s_score))+1 % k = 2 is best
```

k = 2

```
% Plot silhouette for optimal cluster size
idx = kmeans(table2array(cirrhosis_status),k);
silhouette(table2array(cirrhosis_status),idx)
title('Silhouette Plot k=2')
```




```
%Predicting status for each cluster from k-means
cluster1 = (cirrhosis_status.status(idx == 1));
cluster1_survived = sum((cluster1 == 0))
```

```
cluster1_survived = 7
```

```
cluster1_died = sum((cluster1 == 1))
```

```
cluster1_died = 17
```

```
cluster2 = (cirrhosis_status.status(idx == 2));
cluster2_survived = sum((cluster2 == 0))
```

```
cluster2_survived = 158
```

```
cluster2_died = sum((cluster2 == 1))
```

```
cluster2_died = 94
```

Task 5 - Supervised Learning on Time

Linear regression using all the data and nonlog values of time

```
b_lin = fitlm(cirrhosis)
```

```
b_lin =
Linear regression model:
    time ~ 1 + age + albumin + alkphos + ascites + bili + chol + edema + hepato + platelet + protime + sex
```

```
Estimated Coefficients:
                   Estimate        SE         tStat        pValue
                   ________    ________    __________    __________

    (Intercept)       -26619       10653       -2.4988      0.013093
    age                1.4508      5.8132       0.24956       0.80313
    albumin            1083.3      1282.2       0.84487       0.39898
    alkphos           0.11516    0.027248        4.2264    3.3141e-05
    ascites           -180.94      296.96       -0.6093       0.54287
    bili              -24.179      25.017      -0.96651       0.33471
    chol            -0.075279     0.28404      -0.26503        0.7912
    edema             -368.75      280.94       -1.3126       0.19051
    hepato            -7.9454      128.19      -0.06198       0.95063
    platelet         0.029573     0.63797       0.046354       0.96306
    protime           -1366.2      631.37       -2.1638      0.031411
    sex               -1.3879      185.43    -0.0074847       0.99403
    ast                 1.427       1.159        1.2312        0.2194
    spiders           -82.574      134.67      -0.61314       0.54033
    stage             -183.65      79.029       -2.3238      0.020924
    status            -264.43      72.511       -3.6468    0.00032215
    trt                29.459      109.84       0.26819       0.78877
    trig               1.1024     0.96721        1.1398       0.25544
    copper            -1.4629      0.7702       -1.8994       0.05865
    logalb            -1787.2      4290.2      -0.41658       0.67733
    logbil            -214.87      123.76       -1.7362      0.083737
    logpro              17841      7293.8        2.4461       0.01512


Number of observations: 276, Error degrees of freedom: 254
Root Mean Squared Error: 872
R-squared: 0.433,  Adjusted R-Squared: 0.386
F-statistic vs. constant model: 9.24, p-value = 2.5e-21
```

```
% Show the significant coefficients
b_lin.Coefficients(b_lin.Coefficients.pValue < 0.05,:)
```

ans = 6×4 table

	Estimate	SE	tStat	pValue
1 (Intercept)	-2.6619e+04	1.0653e+04	-2.4988	0.0131
2 alkphos	0.1152	0.0272	4.2264	0.0000
3 protime	-1.3662e+03	631.3702	-2.1638	0.0314
4 stage	-183.6492	79.0288	-2.3238	0.0209
5 status	-264.4317	72.5114	-3.6468	0.0003
6 logpro	1.7841e+04	7.2938e+03	2.4461	0.0151

Lasso, Stepwise, and Linear Regression

Used cirrhosis_log because we used log time for all regression models

```
% Top 10 features for linear regression
[B,I] = maxk(abs(R_pear_log(end,1:end-1)),10);
linear_reg_names = [{'(Intercept)'},cirrhosis_log_names(I)];
% Generate crossvalidation groups
NFolds  = 5;
indices = crossvalind('kfold',size(cirrhosis_log,1),NFolds);
```

```
% Declare datasets for storing the outputs
regressionModel = struct();
subsets = {subset, subset_log};
% Run the analysis
for i=1:NFolds
    test = (indices == i);
    train = ~test;
    % Extract train rows from X and Y
    trainData = cirrhosis_log(train,:);
    Xtrain = table2array(trainData(:,1:end-1));
    Ytrain = table2array(trainData(:,end));
    % Extract test rows from X and Y
    testData = cirrhosis_log(test,:);
    Xtest = table2array(testData(:,1:end-1));
    Ytest = table2array(testData(:,end));

    % Lasso
    % Make model and predict
    [lassoB,lassoFit] = lasso(Xtrain,Ytrain,'CV',NFolds);
    coef0 = lassoFit.Intercept(lassoFit.IndexMinMSE);
    coef  = lassoB(:,lassoFit.IndexMinMSE);
    Ypred = Xtest*coef + coef0;
    % Get best coef
    ncoef = sum(coef ~= 0);
    [~,maxCoefIdx] = maxk(abs(coef),ncoef);
    % Save fit for later and evaluate the model
    regressionModel.lasso.fit{i} = lassoFit;
    regressionModel.lasso.coeffVals{i} = coef(maxCoefIdx);
    regressionModel.lasso.coeffNames{i} = cirrhosis_names(maxCoefIdx)';
    regressionModel.lasso.ncoef(i) = ncoef;
    regressionModel.lasso.absError(i) = mean(abs(Ypred-Ytest));
    [regressionModel.lasso.corrR(i),regressionModel.lasso.corrP(i)] = corr(Ytest,Ypred
    [regressionModel.lasso.rankCorrR(i),regressionModel.lasso.rankCorrP(i)] = corr(Ytes

    % Stepwise fit
    for j=1:3
        if j == 1
            stepFit = stepwiselm(trainData,'Upper','linear','ResponseVar','time','Cate
        else
            stepFit = stepwiselm(trainData,'Upper','linear','ResponseVar','time','Pred
        end
        Ypred = stepFit.predict(Xtest);

         [~,coefIdx] = sort(stepFit.Coefficients.pValue);

        % Save fit for later and evaluate the model
        regressionModel.step(j).fit{i} = stepFit;
        regressionModel.step(j).coeffNames{i}  = stepFit.Coefficients.Row(coefIdx);
        regressionModel.step(j).coeffVals{i}   = stepFit.Coefficients.Estimate(coefIdx
        regressionModel.step(j).ncoef(i)       = stepFit.NumPredictors;
        regressionModel.step(j).rSquared(i)    = stepFit.Rsquared.Ordinary;
        regressionModel.step(j).rSquaredAdj(i) = stepFit.Rsquared.Adjusted;
        regressionModel.step(j).absError(i)    = mean(abs(Ypred-Ytest));
        [regressionModel.step(j).corrR(i),regressionModel.step(j).corrP(i)] = corr(Ytes
```

```
        [regressionModel.step(j).rankCorrR(i),regressionModel.step(j).rankCorrP(i)] = c
    end

    % Linear
    % Make model and predict
    mdl = fitlm(Xtrain(:,I),Ytrain);
    Ypred = mdl.predict(Xtest(:,I));

    [~,coefIdx] = sort(mdl.Coefficients.pValue);
    % Evaluate the model
    regressionModel.linear.fit{i} = mdl;
    regressionModel.linear.features{i}    = linear_reg_names;
    regressionModel.linear.coeffNames{i}  = linear_reg_names(coefIdx)';
    regressionModel.linear.coeffVals{i}   = mdl.Coefficients.Estimate(coefIdx);
    regressionModel.linear.rSquared(i)    = mdl.Rsquared.Ordinary;
    regressionModel.linear.rSquaredAdj(i) = mdl.Rsquared.Adjusted;
    regressionModel.linear.ncoef(i)       = length(B);
    regressionModel.linear.absError(i)    = mean(abs(Ypred-Ytest));
    [regressionModel.linear.corrR(i),regressionModel.linear.corrP(i)] = corr(Ytest,Ypr
    [regressionModel.linear.rankCorrR(i),regressionModel.linear.rankCorrP(i)] = corr(Y
end
```

Lasso Analysis

```
rowNames = cellfun(@(s)sprintf('Subsample %d',s),num2cell(transpose(1:NFolds)),'Uniform
varNames = ["ncoeff","Average Absolute Error","Pearsons Correlation",...
    "Pearsons Correlation P-val","Rank Correlation", "Rank Correlation P-val"];
table(regressionModel.lasso.ncoef',regressionModel.lasso.absError',regressionModel.las
    regressionModel.lasso.corrP',regressionModel.lasso.rankCorrR',regressionModel.lasso
    'VariableNames', varNames,'RowNames', rowNames)
```

ans = 5×6 table

...

	ncoeff	Average Absolute Error	Pearsons Correlation
1 Subsample 1	9	0.4432	0.7036
2 Subsample 2	8	0.5350	0.7467
3 Subsample 3	10	0.3987	0.6856
4 Subsample 4	16	0.5324	0.6055
5 Subsample 5	14	0.4332	0.4938

```
maxRow = max(regressionModel.lasso.ncoef);
T = table();
for i=1:NFolds
    tempT = table([regressionModel.lasso.coeffNames{i},num2cell(regressionModel.lasso.
        'VariableNames',cellstr(sprintf("Subset %i",i)));
    T2Pad = [tempT;repmat({{NaN},{NaN}},maxRow-size(tempT,1),1)];
    T = [T T2Pad];
end
T
```

T = 16×5 table

...

	Subset 1		Subset 2	
1	'copper'	1.0398	'edema'	-0.8454
2	'ascites'	-0.5812	'copper'	0.5156
3	'edema'	-0.4742	'ascites'	-0.4198
4	'logalb'	-0.1906	'logalb'	-0.1689
5	'stage'	-0.0950	'stage'	-0.0834
6	'spiders'	-0.0741	'bili'	-6.2160e-04
7	'bili'	-0.0018	'trig'	-3.7997e-04
8	'trig'	-7.2035e-04	'alkphos'	3.4316e-05
9	'alkphos'	3.5227e-05	NaN	NaN
10	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN
12	NaN	NaN	NaN	NaN
13	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN

⋮

Linear Analysis

```
linVarNames = ["R-Squared","Adjusted R-Squared","ncoeff","Average Absolute Error","Pea
    "Pearsons Correlation P-val","Rank Correlation", "Rank Correlation P-val"];
table(regressionModel.linear.rSquared',regressionModel.linear.rSquaredAdj',regressionM
    regressionModel.linear.absError',regressionModel.linear.corrR',regressionModel.lin
    regressionModel.linear.rankCorrR',regressionModel.linear.rankCorrP',...
    'VariableNames', linVarNames,'RowNames', rowNames)
```

ans = 5×8 table

...

	R-Squared	Adjusted R-Squared	ncoeff	Average Absolute Error
1 Subsample 1	0.4958	0.4717	10	0.4621
2 Subsample 2	0.4516	0.4255	10	0.5266
3 Subsample 3	0.5043	0.4807	10	0.4358
4 Subsample 4	0.5061	0.4826	10	0.5072
5 Subsample 5	0.5341	0.5119	10	0.4403

```
maxRow = max(regressionModel.linear.ncoef);
T = table();
for i=1:NFolds
    tempT = table([regressionModel.linear.coeffNames{i},num2cell(regressionModel.linea
        'VariableNames',cellstr(sprintf("Subset %i",i)));
```

```
    T2Pad = [tempT;repmat({[{NaN},{NaN}]},maxRow-size(tempT,1),1)];
    T = [T T2Pad];
end
T
```

T = 11×5 table

...

	Subset 1		Subset 2	
1	'stage'	-0.1503	'edema'	-0.9025
2	'ascites'	-0.5725	'stage'	-0.1236
3	'edema'	-0.4901	'ascites'	-0.4427
4	'logalb'	9.2550	'logbil'	-0.1540
5	'logbil'	-0.1591	'logpro'	6.6627
6	'albumin'	-2.3638	'protime'	-0.5577
7	'copper'	-9.2122e-04	'copper'	-5.1500e-04
8	'protime'	-0.5301	'logalb'	2.5833
9	'logpro'	5.9515	'albumin'	-0.6051
10	'bili'	-0.0106	'(Intercept)'	-2.8779
11	'(Intercept)'	-3.5729	'bili'	-0.0045

Stepwise Analysis

```
stepVarNames = ["R-Squared","Adjusted R-Squared","Number of Predictors","Average Absolu
    "Pearsons Correlation P-val","Rank Correlation", "Rank Correlation P-val"];
stepTables = cell(3,1);
for i=1:3
    stepTables{i} = table(regressionModel.step(i).rSquared',regressionModel.step(i).rS
        regressionModel.step(i).ncoef',regressionModel.step(i).absError',regressionMod
        regressionModel.step(i).corrP',regressionModel.step(i).rankCorrR',regressionMo
        'VariableNames', stepVarNames,'RowNames', rowNames);
end
```

Stepwise Full

```
j = 1;
stepTables{j}
```

ans = 5×8 table

...

	R-Squared	Adjusted R-Squared	Number of Predictors
1 Subsample 1	0.4954	0.4787	6
2 Subsample 2	0.4481	0.4352	4
3 Subsample 3	0.5034	0.4871	6
4 Subsample 4	0.5252	0.5073	7

	R-Squared	Adjusted R-Squared	Number of Predictors
5 Subsample 5	0.5506	0.5337	7

```
maxRow = max(cellfun('length',regressionModel.step(j).coeffNames));
T = table();
for i=1:NFolds
    tempT = table([regressionModel.step(j).coeffNames{i},num2cell(regressionModel.step
        'VariableNames',cellstr(sprintf("Subset %i",i)));
    T2Pad = [tempT;repmat({[{NaN},{NaN}]},maxRow-size(tempT,1),1)];
    T = [T T2Pad];
end
T
```

T = 9×5 table

...

	Subset 1		Subset 2	
1	'(Intercept)'	5.8974	'(Intercept)'	6.3724
2	'logbil'	-0.2326	'edema_1'	-1.3755
3	'alkphos'	5.9631e-05	'logbil'	-0.2369
4	'logalb'	1.3461	'alkphos'	5.6260e-05
5	'ascites_1'	-0.6639	'edema_0.5'	-0.3729
6	'edema_1'	-0.5777	'logalb'	0.9083
7	'copper'	-0.0012	NaN	NaN
8	'edema_0.5'	-0.2519	NaN	NaN
9	NaN	NaN	NaN	NaN

Stepwise Subset

```
j = 2;
stepTables{j}
```

ans = 5×8 table

...

	R-Squared	Adjusted R-Squared	Number of Predictors
1 Subsample 1	0.4608	0.4456	6
2 Subsample 2	0.4237	0.4103	5
3 Subsample 3	0.4675	0.4551	5
4 Subsample 4	0.4599	0.4499	4
5 Subsample 5	0.5141	0.5028	5

```
maxRow = max(cellfun('length',regressionModel.step(j).coeffNames));
T = table();
for i=1:NFolds
    tempT = table([regressionModel.step(j).coeffNames{i},num2cell(regressionModel.step
        'VariableNames',cellstr(sprintf("Subset %i",i)));
```

```
    T2Pad = [tempT;repmat({[{NaN},{NaN}]},maxRow-size(tempT,1),1)];
    T = [T T2Pad];
end
T
```

T = 7×5 table

...

	Subset 1		Subset 2	
1	'(Intercept)'	5.9967	'(Intercept)'	6.5704
2	'bili'	-0.0477	'edema'	-0.9239
3	'albumin'	0.4463	'bili'	-0.0400
4	'ascites'	-0.7451	'alkphos'	4.8620e-05
5	'alkphos'	4.8464e-05	'albumin'	0.2744
6	'spiders'	-0.2315	'ascites'	-0.5059
7	'edema'	-0.4829	NaN	NaN

Stepwise Subset Log

```
j = 3;
stepTables{j}
```

ans = 5×8 table

...

	R-Squared	Adjusted R-Squared	Number of Predictors
1 Subsample 1	0.4850	0.4730	5
2 Subsample 2	0.4522	0.4394	5
3 Subsample 3	0.4905	0.4787	5
4 Subsample 4	0.4821	0.4700	5
5 Subsample 5	0.5365	0.5235	6

```
maxRow = max(cellfun('length',regressionModel.step(j).coeffNames));
T = table();
for i=1:NFolds
    tempT = table([regressionModel.step(j).coeffNames{i},num2cell(regressionModel.step
        'VariableNames',cellstr(sprintf("Subset %i",i)));
    T2Pad = [tempT;repmat({[{NaN},{NaN}]},maxRow-size(tempT,1),1)];
    T = [T T2Pad];
end
T
```

T = 7×5 table

...

	Subset 1		Subset 2	
1	'(Intercept)'	5.7683	'(Intercept)'	6.5915
2	'logbil'	-0.2823	'logbil'	-0.2273
3	'logalb'	1.3844	'edema'	-0.9090

	Subset 1		Subset 2	
4	'alkphos'	5.5215e-05	'alkphos'	5.3089e-05
5	'ascites'	-0.6648	'ascites'	-0.5026
6	'edema'	-0.5732	'logalb'	0.7486
7	NaN	NaN	NaN	NaN

Overall Averages Analysis

```
overSum = struct();
for i=1:3
    overSum.step(1,i) = mean(regressionModel.step(i).corrR);
    overSum.step(2,i) = mean(regressionModel.step(i).rankCorrR);
    overSum.step(3,i) = mean(regressionModel.step(i).absError);
    overSum.step(4,i) = mean(regressionModel.step(i).ncoef);
end
overSum.linear = [mean(regressionModel.linear.corrR)
    mean(regressionModel.linear.rankCorrR)
    mean(regressionModel.linear.absError)
    mean(regressionModel.linear.ncoef)];

overSum.lasso = [mean(regressionModel.lasso.corrR)
    mean(regressionModel.lasso.rankCorrR)
    mean(regressionModel.lasso.absError)
    mean(regressionModel.lasso.ncoef)];

sumVarNames = ["Stepwise Full", "Stepwise Subset", "Stepwise Subset Log", "Linear FitLr
sumRowNames = ["Pearsons Correlation","Rank Correlation","Mean Absolute Error","Number
table(overSum.step(:,1),overSum.step(:,2),overSum.step(:,3),overSum.linear,overSum.lass
        'VariableNames', sumVarNames,'RowNames', sumRowNames)
```

ans = 4×5 table

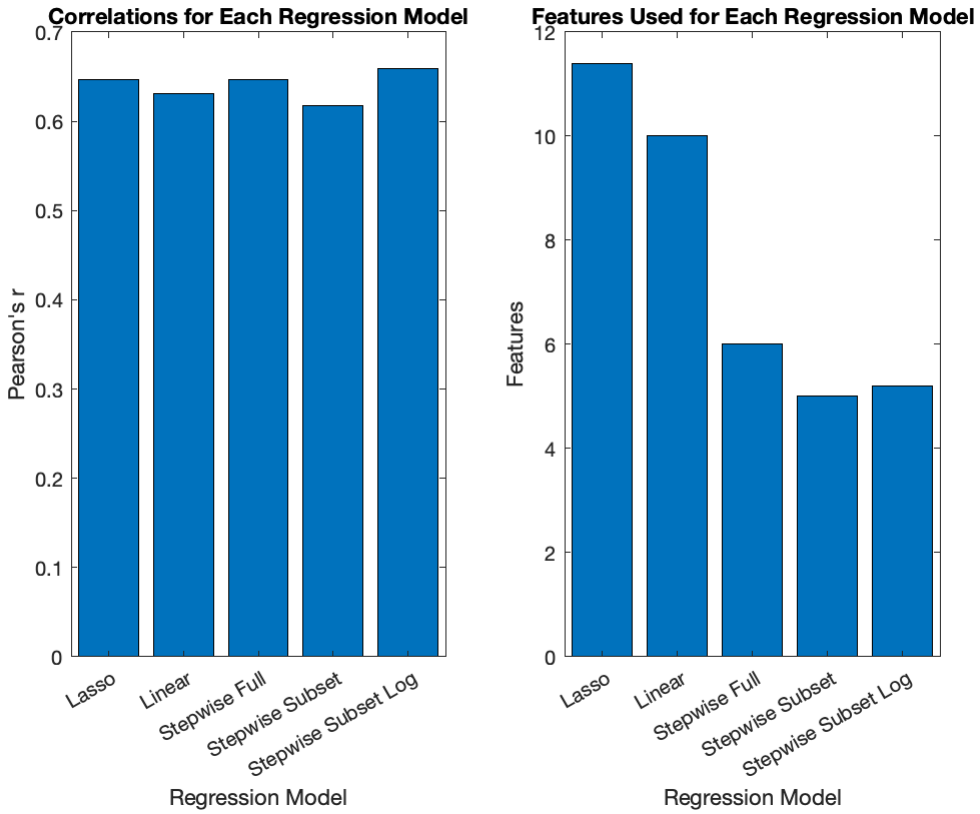
...

	Stepwise Full	Stepwise Subset	Stepwise Subset Log
1 Pearsons Correlation	0.6467	0.6170	0.6588
2 Rank Correlation	0.5632	0.5578	0.5627
3 Mean Absolute Error	0.4638	0.4835	0.4637
4 Number of Coefficients	6.0000	5.0000	5.2000

Method Comparision Plots

```
figure();
subplot(1,2,1)
X = categorical({'Lasso','Stepwise Full','Stepwise Subset','Stepwise Subset Log','Linea
bar(X,[overSum.lasso(1),overSum.step(1,:),overSum.linear(1)])
title('Correlations for Each Regression Model')
xlabel('Regression Model')
ylabel("Pearson's r")
subplot(1,2,2)
```

```
bar(X,[overSum.lasso(4),overSum.step(4,:),overSum.linear(4)])
title('Features Used for Each Regression Model')
xlabel('Regression Model')
ylabel("Features")
```



Task 6 - Supervised Learning on Status

Logistic Regression

```
% Number of features to select
NFeatures = 8;

cirrhosis_status_array = table2array(cirrhosis_status);

accuracyGuess = zeros(100,NFolds);
meanGuess = zeros(NFolds,1);
sigDiff = zeros(NFolds,2);

for i=1:NFolds
    test = (indices == i);
    train = ~test;

    % Extract train rows from X and Y
    Xtrain = cirrhosis_status_array(train,1:end-1);
    Ytrain = cirrhosis_status_array(train,end);

    % Extract test rows from X and Y
    Xtest = cirrhosis_status_array(test,1:end-1);
```

```
    Ytest = cirrhosis_status_array(test,end);

    % Create logistic model with train data.
    [b,dev,stats] = mnrfit(Xtrain, categorical(Ytrain));
    [p,I] = sort(stats.p(2:end,:));
    [p_min,min_idx] = min(p,[],2);
    regressionModel.logist.bestP{i} = p_min(1:NFeatures);
    regressionModel.logist.bestFeatures{i} = diag(I(1:NFeatures,min_idx(1:NFeatures)))
    regressionModel.logist.bestFeaturesNames{i} = cirrhosis_status_names(regressionMod
    regressionModel.logist.bestFeaturesCoeff{i} = b(I(1:NFeatures));

    % Predict Y values from model.
    Ypred = mnrval(b,Xtest);
    [~,Ypred] = max(Ypred,[],2);
    Ypred = double(Ypred - 1);

    %Comparing model to 100 random guesses
    for k = 1:100
        random_pred = Ytrain(randperm(length(Ytest)));
        accuracyGuess(k,i) = sum(random_pred == Ytest)/length(Ytest);
    end
    meanGuess(i) = mean(accuracyGuess(:,i));
    accuracy = sum(Ypred == Ytest)/length(Ypred);
    %T test comparing model accuracy to random guess accuracy
    [sigDiff(i,1),sigDiff(i,2)] = ttest2(accuracy,accuracyGuess(:,i));
    meanGuessError = meanGuess(i) - accuracy;

    % Evaluate Model
    regressionModel.logist.confusion{i} = confusionmat(Ytest,Ypred);
    regressionModel.logist.accuracy(i)  = accuracy;
    regressionModel.logist.precision(i) = sum(Ypred ~=0 & Ypred==Ytest)/sum(Ypred~=0);
    regressionModel.logist.recall(i)    = sum(Ypred ~=0 & Ytest ==Ypred)/sum(Ytest~=0)
    regressionModel.logist.absError(i)  = mean(abs(Ypred - Ytest));
    [regressionModel.logist.corrR(i),regressionModel.logist.corrP(i)] = corr(Ypred,Yte
    [regressionModel.logist.rankCorrR(i),regressionModel.logist.rankCorrP(i)] = corr(Y
    regressionModel.logist.meanGuessError(i) = meanGuessError;
    regressionModel.logist.compareGuess(i)   = sigDiff(i,2);
end
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
2.164030e-16.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
7.961029e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
6.517939e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
5.336438e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
4.369106e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.577122e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.436861e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.302100e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.172622e-17.
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.147343e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.122264e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.117273e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.112289e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.107313e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.102346e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.101353e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.101155e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.100956e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.100758e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.100559e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.100520e-17.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.100520e-17.
```

Logistic Regression Analysis

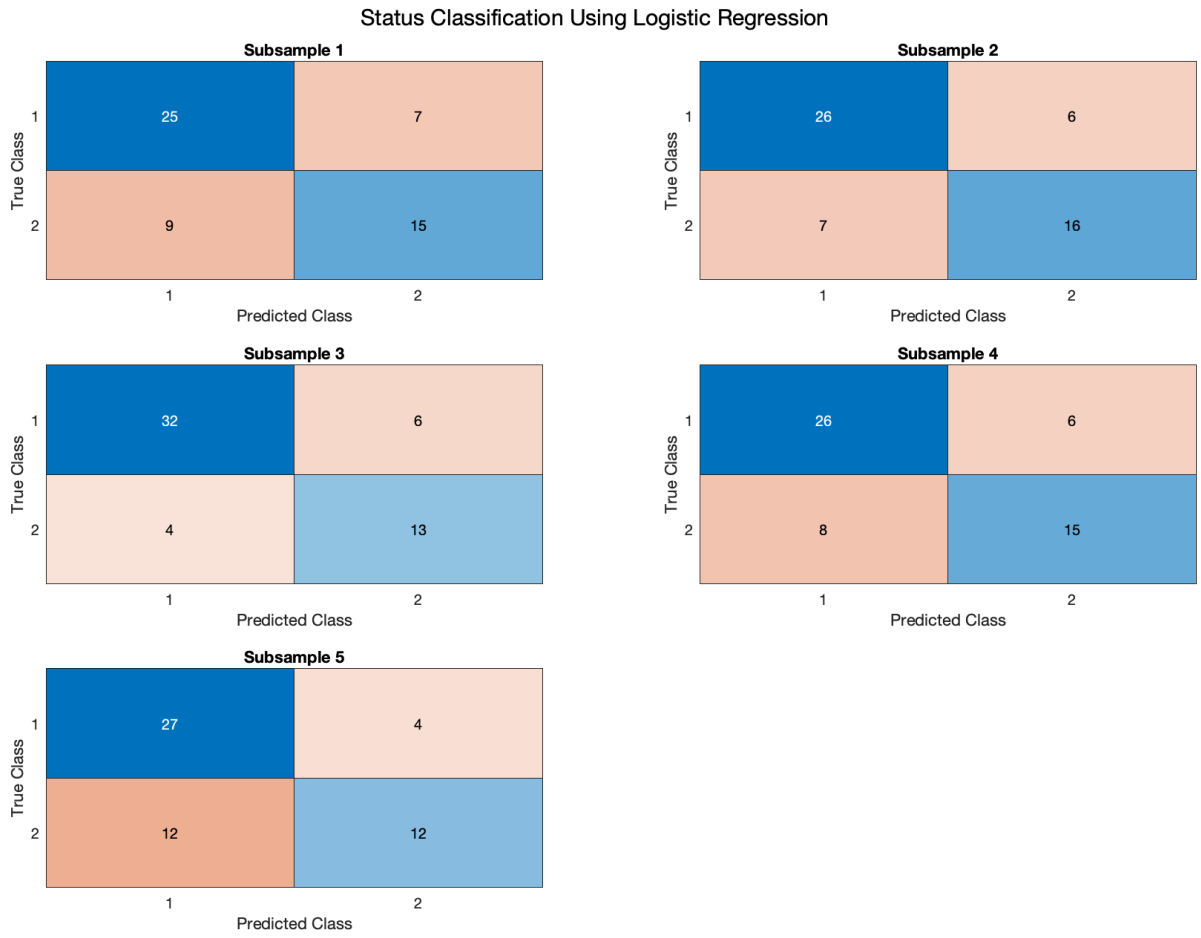
```
logVarNames = ["Accuracy","Precision","Recall","Average Absolute Error","Pearsons Corr
    "Pearsons Correlation P-val","Rank Correlation", "Rank Correlation P-val","Mean Gu
table(regressionModel.logist.accuracy',regressionModel.logist.precision',regressionMod
    regressionModel.logist.absError',regressionModel.logist.corrR',regressionModel.log
    regressionModel.logist.rankCorrR',regressionModel.logist.rankCorrP', regressionMod
    regressionModel.logist.compareGuess','VariableNames',logVarNames,'RowNames', rowNa
```

ans = 5×10 table

...

	Accuracy	Precision	Recall	Average Absolute Error
1 Subsample 1	0.7143	0.6818	0.6250	0.2857
2 Subsample 2	0.7636	0.7273	0.6957	0.2364
3 Subsample 3	0.8182	0.6842	0.7647	0.1818
4 Subsample 4	0.7455	0.7143	0.6522	0.2545
5 Subsample 5	0.7091	0.7500	0.5000	0.2909

```
f = figure();
set(f, 'Position', expandFigSize)
for i=1:NFolds
    subplot(ceil(NFolds/2),floor(NFolds/2),i)
    cm = confusionchart(regressionModel.logist.confusion{i});
    cm.Title = sprintf("Subsample %d",i);
end
sgtitle("Status Classification Using Logistic Regression")
```




```
T = table();
for i=1:NFolds
    tempT = table([regressionModel.logist.bestFeaturesNames{i},...
        num2cell(regressionModel.logist.bestFeaturesCoeff{i})],...
        'VariableNames',cellstr(sprintf("Subset %i",i)));
    T = [T tempT];
end
T
```

T = 8×5 table

...

	Subset 1		Subset 2	
1	'age'	88.8792	'alkphos'	-7.2926
2	'alkphos'	-3.9411	'age'	79.1033
3	'logpro'	-0.5764	'logpro'	-0.5537
4	'ast'	0.5772	'ast'	0.7510
5	'protime'	-0.0020	'trig'	0.1450
6	'logbil'	13.6599	'protime'	0.0013

	Subset 1		Subset 2	
7	'copper'	-2.9115e-05	'sex'	3.6249
8	'stage'	0.1487	'logbil'	24.6619

Task 7 - Random Forest Model

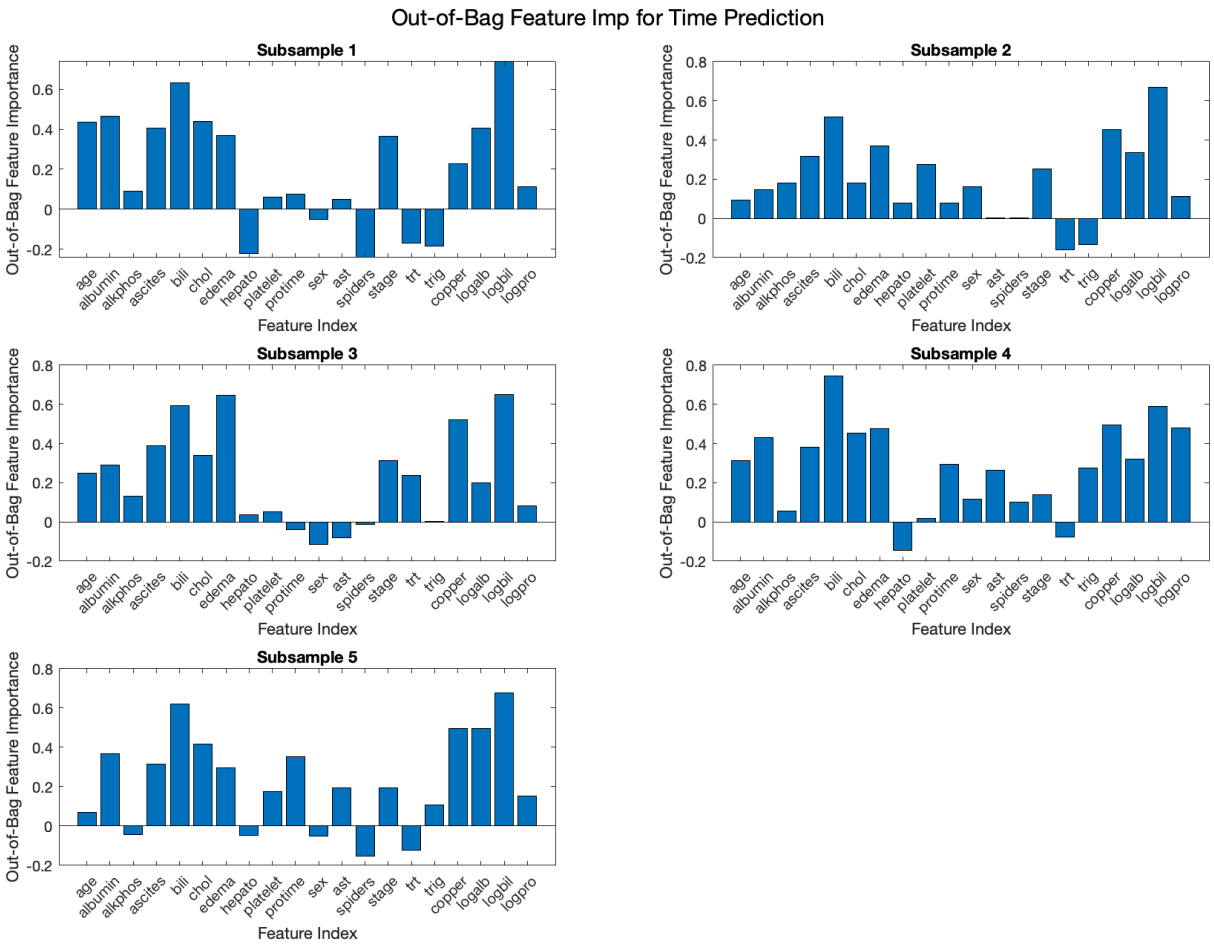
Tree Bagger Time Regression

Used cirrhosis_log again because it's regression.

```
for i=1:NFolds
    test = (indices == i);
    train = ~test;
    % Extract train rows from X and Y
    trainData = cirrhosis_log(train,:);
    Xtrain = table2array(trainData(:,1:end-1));
    Ytrain = table2array(trainData(:,end));
    % Extract test rows from X and Y
    testData = cirrhosis_log(test,:);
    Xtest = table2array(testData(:,1:end-1));
    Ytest = table2array(testData(:,end));

    % Create linear model with train data.
    rfMdlTime = TreeBagger(100,Xtrain,Ytrain,'Method','regression','OOBPrediction','On
    Ypred = rfMdlTime.predict(Xtest);

    % Evaluate the model
    regressionModel.rfMdlTime.fit{i} = rfMdlTime;
    regressionModel.rfMdlTime.absError(i) = mean(abs(Ypred-Ytest));
    [regressionModel.rfMdlTime.corrR(i),regressionModel.rfMdlTime.corrP(i)] = corr(Ytes
    [regressionModel.rfMdlTime.rankCorrR(i),regressionModel.rfMdlTime.rankCorrP(i)] = 
end
```

Tree Bagger Time Regression Analysis

```
f = figure();
set(f, 'Position', expandFigSize)
for i=1:NFolds
    subplot(ceil(NFolds/2),floor(NFolds/2),i)
    bar(regressionModel.rfMdlTime.fit{i}.OOBPermutedPredictorDeltaError)
    xlabel('Feature Index')
    ylabel('Out-of-Bag Feature Importance')
    title(sprintf('Subsample %i',i))
    xticks(1:length(cirrhosis_log_names)-1)
    xticklabels(cirrhosis_log_names(1:end-1))
    xtickangle(45)
end
sgtitle('Out-of-Bag Feature Imp for Time Prediction')
```




```
rfVarNames = ["Average Absolute Error","Pearsons Correlation",...
    "Pearsons Correlation P-val","Rank Correlation", "Rank Correlation P-val"];
table(regressionModel.rfMdlTime.absError',regressionModel.rfMdlTime.corrR',...
    regressionModel.rfMdlTime.corrP',regressionModel.rfMdlTime.rankCorrR',regressionMo
    'VariableNames', rfVarNames,'RowNames', rowNames)
```

```
ans = 5×5 table
```

...

	Average Absolute Error	Pearsons Correlation	Pearsons Correlation P-val
1 Subsample 1	0.4358	0.6918	0.0000
2 Subsample 2	0.4941	0.8378	0.0000
3 Subsample 3	0.3698	0.7372	0.0000
4 Subsample 4	0.5194	0.6111	0.0000
5 Subsample 5	0.4533	0.3848	0.0037

Tree Bagger Status Classification

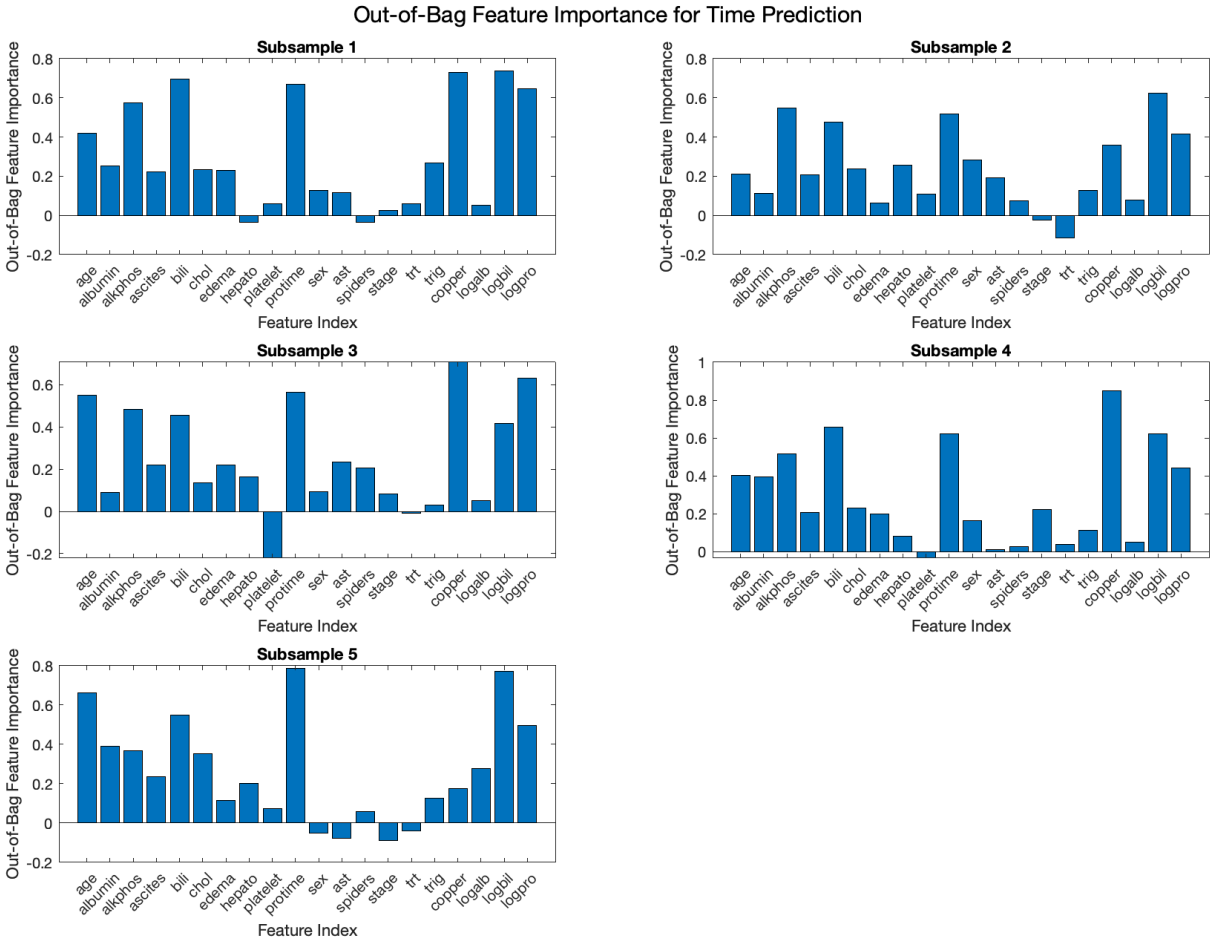
```
accuracyGuess = zeros(100,NFolds);
meanGuess = zeros(NFolds,1);
```

```
sigDiff = zeros(NFolds,2);

for i=1:NFolds
    test = (indices == i);
    train = ~test;
    % Extract train rows from X and Y
    trainData = cirrhosis_status(train,:);
    Xtrain = table2array(trainData(:,1:end-1));
    Ytrain = table2array(trainData(:,end));
    % Extract test rows from X and Y
    testData = cirrhosis_status(test,:);
    Xtest = table2array(testData(:,1:end-1));
    Ytest = double(table2array(testData(:,end)));

    % Create linear model with train data.
    rfMdlStatus = TreeBagger(100,Xtrain,Ytrain,'OOBPrediction','On','OOBPredictorImpor
    Ypred = double(categorical(rfMdlStatus.predict(Xtest))) - 1;

     %Comparing model to 100 random guesses
    for k = 1:100
        random_pred = Ytrain(randperm(length(Ytest)));
        accuracyGuess(k,i) = sum(random_pred == Ytest)/length(Ytest);
    end
    meanGuess(i) = mean(accuracyGuess(:,i));

    accuracy = sum(Ypred == Ytest)/length(Ypred);
    %T test comparing model accuracy to random guess accuracy
    [sigDiff(i,1),sigDiff(i,2)] = ttest2(accuracy,accuracyGuess(:,i));
    meanGuessError = meanGuess(i) - accuracy;

    % Evaluate the model
    regressionModel.rfMdlStatus.fit{i} = rfMdlStatus;
    regressionModel.rfMdlStatus.confusion{i} = confusionmat(Ytest,Ypred);
    regressionModel.rfMdlStatus.absError(i)  = mean(abs(Ypred-Ytest));
    regressionModel.rfMdlStatus.accuracy(i)  = accuracy;
    [regressionModel.rfMdlStatus.corrR(i),regressionModel.rfMdlStatus.corrP(i)] = corr
    [regressionModel.rfMdlStatus.rankCorrR(i),regressionModel.rfMdlStatus.rankCorrP(i)
    regressionModel.rfMdlStatus.meanGuessError(i) = meanGuessError;
    regressionModel.rfMdlStatus.compareGuess(i) = sigDiff(i,2);
end
```

Tree Bagger Status Classification Analysis

```
f = figure();
set(f, 'Position', expandFigSize)
for i=1:NFolds
    subplot(ceil(NFolds/2),floor(NFolds/2),i)
    bar(regressionModel.rfMdlStatus.fit{i}.OOBPermutedPredictorDeltaError)
    xlabel('Feature Index')
    ylabel('Out-of-Bag Feature Importance')
    title(sprintf('Subsample %i',i))
    xticks(1:length(cirrhosis_status_names)-1)
    xticklabels(cirrhosis_status_names(1:end-1))
    xtickangle(45)
```

```
end
sgtitle('Out-of-Bag Feature Importance for Time Prediction')
```




```
rfClassVarNames = ["Average Absolute Error","Accuracy","Pearsons Correlation",...
    "Pearsons Correlation P-val","Rank Correlation", "Rank Correlation P-val","Mean Gu
table(regressionModel.rfMdlStatus.absError',regressionModel.rfMdlStatus.accuracy',regr
    regressionModel.rfMdlStatus.corrP',regressionModel.rfMdlStatus.rankCorrR',regressi
    regressionModel.rfMdlStatus.meanGuessError',regressionModel.rfMdlStatus.compareGue
    'VariableNames',rfClassVarNames,'RowNames', rowNames)
```

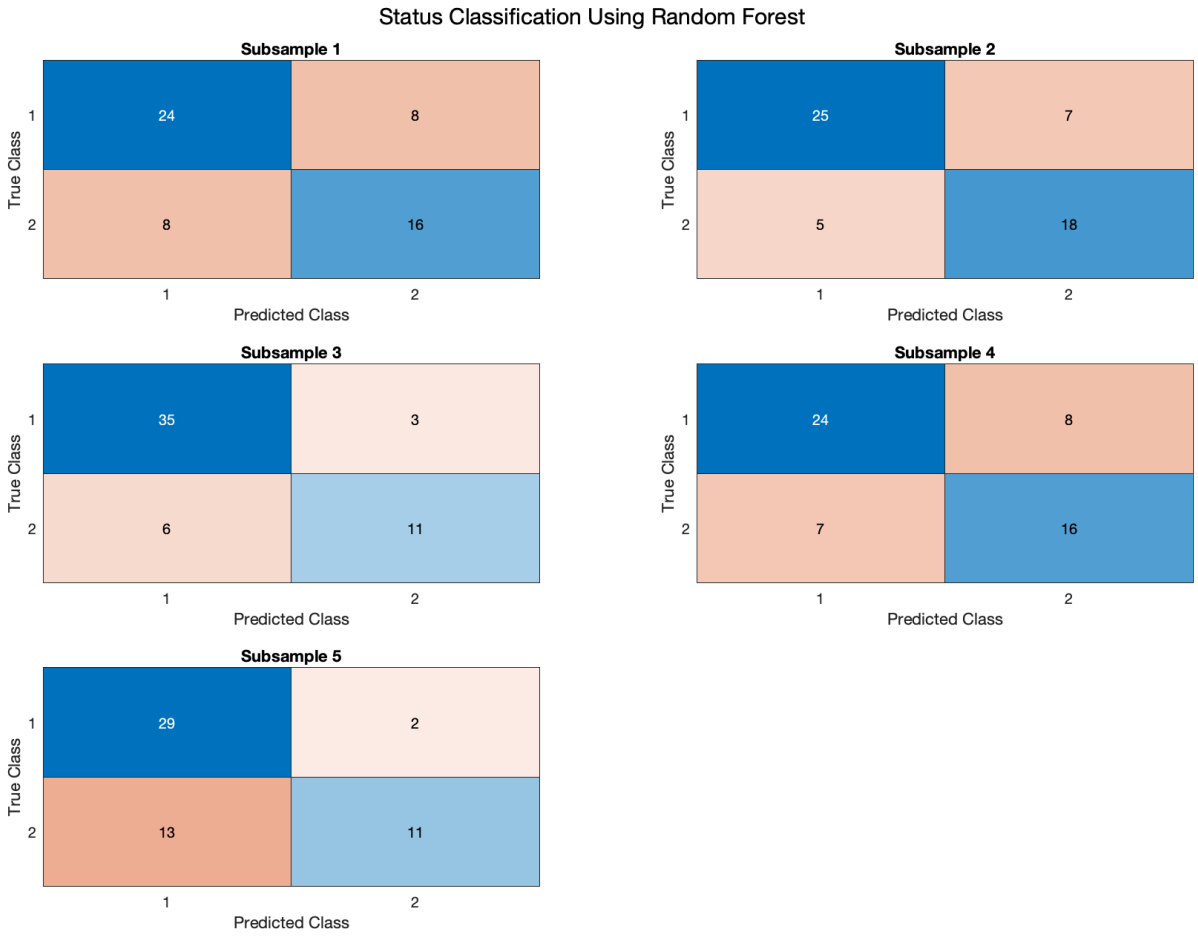
ans = 5×8 table

...

	Average Absolute Error	Accuracy	Pearsons Correlation
1 Subsample 1	0.2857	0.7143	0.4167
2 Subsample 2	0.2182	0.7818	0.5586
3 Subsample 3	0.1636	0.8364	0.6027
4 Subsample 4	0.2727	0.7273	0.4432
5 Subsample 5	0.2727	0.7273	0.4597

```
f = figure();
```

```
set(f, 'Position', expandFigSize)
for i=1:NFolds
    subplot(ceil(NFolds/2),floor(NFolds/2),i)
    cm = confusionchart(regressionModel.rfMdlStatus.confusion{i});
    cm.Title = sprintf("Subsample %d",i);
end
sgtitle("Status Classification Using Random Forest")
```



Task 8 - New Machine Learning Algorithm: Support Vector Machiine

For predicting status - binary classification

```
for i=1:NFolds
    test = (indices == i);
    train = ~test;
    % Extract train rows from X and Y
    trainData = cirrhosis_status(train,:);
    Xtrain = table2array(trainData(:,1:end-1));
    Ytrain = table2array(trainData(:,end));
    % Extract test rows from X and Y
    testData = cirrhosis_status(test,:);
    Xtest = table2array(testData(:,1:end-1));
```

```
    Ytest = double(table2array(testData(:,end)));

    svmMdlStatus = fitcsvm(cirrhosis_status,'status');
    Ypred = double(categorical(svmMdlStatus.predict(Xtest))) - 1;


     %Comparing model to 100 random guesses
    for k = 1:100
        random_pred = Ytrain(randperm(length(Ytest)));
        accuracyGuess(k,i) = sum(random_pred == Ytest)/length(Ytest);
    end
    meanGuess(i) = mean(accuracyGuess(:,i));

    accuracy = sum(Ypred == Ytest)/length(Ypred);

    %T test comparing model accuracy to random guess accuracy
    [sigDiff(i,1),sigDiff(i,2)] = ttest2(accuracy,accuracyGuess(:,i));
    meanGuessError = meanGuess(i) - accuracy;

    % Evaluate the model
    regressionModel.svmMdlStatus.fit{i} = svmMdlStatus;
    regressionModel.svmMdlStatus.confusion{i} = confusionmat(Ytest,Ypred);
    regressionModel.svmMdlStatus.absError(i)  = mean(abs(Ypred-Ytest));
    regressionModel.svmMdlStatus.accuracy(i)  = accuracy;
    [regressionModel.svmMdlStatus.corrR(i),regressionModel.svmMdlStatus.corrP(i)] = co
    [regressionModel.svmMdlStatus.rankCorrR(i),regressionModel.svmMdlStatus.rankCorrP(
    regressionModel.svmMdlStatus.meanGuessError(i) = meanGuessError;
    regressionModel.svmMdlStatus.compareGuess(i) = sigDiff(i,2);
end
```

```
svmClassVarNames = ["Average Absolute Error","Accuracy","Pearsons Correlation",...
    "Pearsons Correlation P-val","Rank Correlation", "Rank Correlation P-val","Mean Gu
table(regressionModel.svmMdlStatus.absError',regressionModel.svmMdlStatus.accuracy',re
    regressionModel.svmMdlStatus.corrP',regressionModel.svmMdlStatus.rankCorrR',regress
    regressionModel.svmMdlStatus.meanGuessError',regressionModel.svmMdlStatus.compareG
    'VariableNames',svmClassVarNames,'RowNames', rowNames)
```

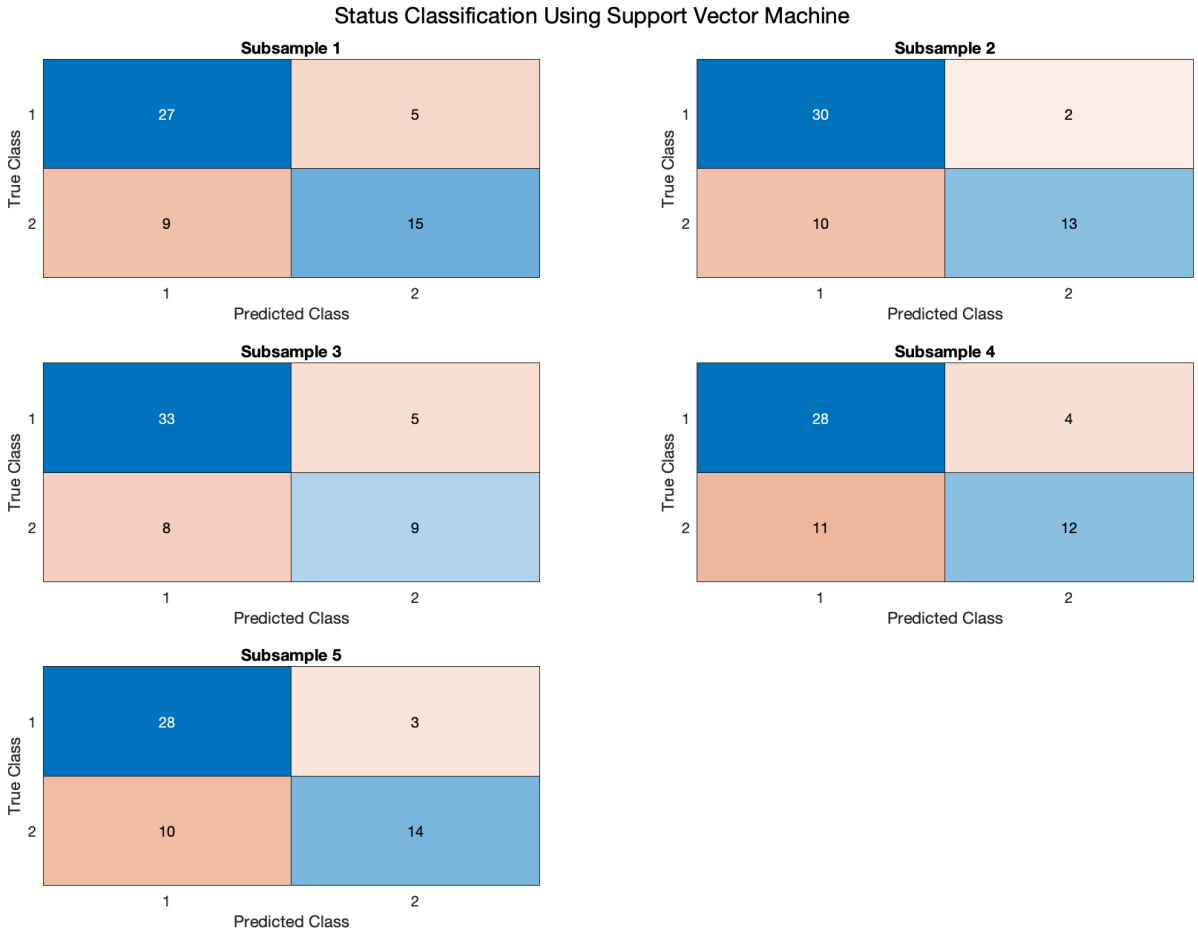
ans = 5×8 table

...

	Average Absolute Error	Accuracy	Pearsons Correlation
1 Subsample 1	0.2500	0.7500	0.4841
2 Subsample 2	0.2182	0.7818	0.5568
3 Subsample 3	0.2364	0.7636	0.4220
4 Subsample 4	0.2727	0.7273	0.4309
5 Subsample 5	0.2364	0.7636	0.5222

```
f = figure();
set(f, 'Position', expandFigSize)
for i=1:NFolds
    subplot(ceil(NFolds/2),floor(NFolds/2),i)
```

```
    cm = confusionchart(regressionModel.svmMdlStatus.confusion{i});
    cm.Title = sprintf("Subsample %d",i);
end
sgtitle("Status Classification Using Support Vector Machine")
```



Task 9 - Status Logistic vs Random Forest vs SVM

```
compVarNames = ["Logistic","Random Forest", "Support Vector Machine (SVM)"];
compRowNames = ["Pearsons Correlation","Mean Absolute Error","Accuracy"];
compData = [
    mean(regressionModel.logist.corrR),    mean(regressionModel.rfMdlStatus.corrR),
    mean(regressionModel.logist.absError), mean(regressionModel.rfMdlStatus.absError),
    mean(regressionModel.logist.accuracy), mean(regressionModel.rfMdlStatus.accuracy),
];
table(compData(:,1),compData(:,2), compData(:,3),...
        'VariableNames', compVarNames,'RowNames', compRowNames)
```

ans = 3×3 table

	Logistic	Random Forest	Support Vector Machine (SVM)
1 Pearsons Correlation	0.4780	0.4962	0.4832
2 Mean Absolute Error	0.2499	0.2426	0.2427

	Logistic	Random Forest	Support Vector Machine (SVM)
3 Accuracy	0.7501	0.7574	0.7573

```
f = figure();
set(f, 'Position', [0,0,1000,500])
subplot(1,3,1)
confusionchart(round(mean(cat(3,regressionModel.rfMdlStatus.confusion{:}),3)));
title("Random Forest Status Classification")
subplot(1,3,2)
confusionchart(round(mean(cat(3,regressionModel.logist.confusion{:}),3)));
title("Logistic Regression Status Classification")
subplot(1,3,3)
confusionchart(round(mean(cat(3,regressionModel.svmMdlStatus.confusion{:}),3)));
title("Support Vector Machine Status Classification")
```