Upload the big dataset (>250MB) to MATLAB Drive and read table

```
BitcoinData = readtable('bitcoin-dataset.csv');
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers
before creating variable names for the table. The original column headers are saved in the
VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.
```

```
Tablesize = size(BitcoinData)
```

```
Tablesize = 1×2
     4857377           8
```

```
head(BitcoinData,1)
```

```
ans = 1×8 table
```

...

	Timestamp	Open	High	Low	Close	Volume__BTC_
1	1.3253e+09	4.3900	4.3900	4.3900	4.3900	0.4556

Convert timestamp

```
datetest = BitcoinData(:,1);
datetest = table2array(datetest);
BitcoinData.Timestamp = datetime(datetest, 'ConvertFrom', 'posixtime');
```

Remove NaNs

```
% how many NaNs in each column
NaN2 = sum(isnan(BitcoinData.Open));
NaN3 = sum(isnan(BitcoinData.High));
NaN4 = sum(isnan(BitcoinData.Low));
NaN5 = sum(isnan(BitcoinData.Close));
NaN6 = sum(isnan(BitcoinData.Volume__BTC_));
NaN7 = sum(isnan(BitcoinData.Volume__Currency_));
NaN8 = sum(isnan(BitcoinData.Weighted_Price));

% remove NaNs
BitcoinData = BitcoinData(isnan(BitcoinData.High) == 0,:);
```

Select data

```
BitcoinDataTimetable = table2timetable(BitcoinData);

% March 2013
% S = timerange('17-Mar-2013 00:00:00','18-Mar-2013 23:59:00'); %2days
% S = timerange('17-Mar-2013 00:00:00','23-Mar-2013 23:59:00'); %7days
% S = timerange('17-Mar-2013 00:00:00','30-Mar-2013 23:59:00'); %14days

% March 2016
% S = timerange('17-Mar-2016 00:00:00','18-Mar-2016 23:59:00'); %2days
% S = timerange('17-Mar-2016 00:00:00','23-Mar-2016 23:59:00'); %7days
% S = timerange('17-Mar-2016 00:00:00','30-Mar-2016 23:59:00'); %14days

% March 2019
% S = timerange('17-Mar-2019 00:00:00','18-Mar-2019 23:59:00'); %2days
% S = timerange('17-Mar-2019 00:00:00','23-Mar-2019 23:59:00'); %7days
% S = timerange('17-Mar-2019 00:00:00','30-Mar-2019 23:59:00'); %14days

% March 2021

% S = timerange('17-Mar-2021 00:00:00','18-Mar-2021 23:59:00'); %2days
% S = timerange('17-Mar-2021 00:00:00','23-Mar-2021 23:59:00'); %7days
S = timerange('17-Mar-2021 00:00:00','30-Mar-2021 23:59:00'); %14days

BitcoinData2 = BitcoinDataTimetable(S,:);
Tablesize = size(BitcoinData2)
```

```
Tablesize = 1×2
       20108           7
```

Calculate price change

```
yconvert = diff(BitcoinData2.Close)
```

```
yconvert = 20107×1
   22.0000
  -45.7500
    4.8500
 -103.7700
  -79.3100
  -21.0600
  -49.2400
   85.5200
   23.2700
  -28.8300
      ⋮
```

```
% need to drop the top row
xconvert = BitcoinData2{[2:end], [1:3 5:7]};
```

Find ups and downs

```
Ysvm = double(yconvert >= 0) % Response data
```

```
Ysvm = 20107×1
     1
     0
     1
     0
     0
     0
     0
     1
     1
     0
```

⋮

```
sum(Ysvm)
```

```
ans = 9928
```

Create model - preprocessing

```
% standardization, and Initial predictor set (matrix)

Xsvm = zscore(xconvert);

BitcoinData_new = [Xsvm, Ysvm];
```

Split dataset into training and testing data

```
PD = 0.20;
cv = cvpartition(size(BitcoinData_new,1),'HoldOut',PD);
Xtrain = BitcoinData_new(cv.training,[1:end - 1]);
Ytrain = BitcoinData_new(cv.training,end);
Xtest = BitcoinData_new(cv.test,[1:end - 1]);
Ytest = BitcoinData_new(cv.test,end);
size(Xtrain)
```

```
ans = 1×2
       16086           6
```

```


Xtrain = zscore(Xtrain);
Xtest = zscore(Xtest); % To be more precise, this zscore should be calculated using the
```

SVM model with training-testing split

```
% svmModel = fitcsvm(Xtrain, Ytrain, 'BoxConstraint', 100, 'KernelScale', 1,"KernelFunc
```

```
% numSV1 = size(svmModel.SupportVectors,1)
```

```
% CVSVMModel = crossval(svmModel)
% classLoss = kfoldLoss(CVSVMModel)
```

Train SVM Model with Kernel Scales

```
Collect_F = [];     Collect_R = [];     Collect_P = [];     Collect_A = [];
KS = [0.1, 0.5, 3]; %100
for i = KS
    disp(['KS = ' num2str(i)])
    SVM = fitcsvm(Xtrain,Ytrain, 'KernelFunction', 'rbf', 'KernelScale', i, 'BoxConstra
    [labels score] = predict(SVM, Xtest);
    numSV = size(SVM.SupportVectors,1)
    [ClassPerformance, OverallAccuracy] = CFM_Stats(Ytest, labels)
    Collect_F = [Collect_F, ClassPerformance.Fscore];
    Collect_R = [Collect_R, ClassPerformance.Fscore];
    Collect_P = [Collect_P, ClassPerformance.Fscore];
    Collect_A = [Collect_A, OverallAccuracy];

end
```

```
KS = 0.1
numSV = 13903
Confusion Matrix:
        1433         587
         732        1269

Overall accuracy = 0.67197
ClassPerformance = 2×6 table
```

	accuracy	precision	recall	Fscore	sensitivity	specificity
1	0.6720	0.6619	0.7094	0.6848	0.7094	0.6342
2	0.6720	0.6837	0.6342	0.6580	0.6342	0.7094

```
OverallAccuracy = 0.6720
KS = 0.5
numSV = 14830
Confusion Matrix:
        1575         445
         717        1284

Overall accuracy = 0.71102
ClassPerformance = 2×6 table
```

	accuracy	precision	recall	Fscore	sensitivity	specificity
1	0.7110	0.6872	0.7797	0.7305	0.7797	0.6417
2	0.7110	0.7426	0.6417	0.6885	0.6417	0.7797

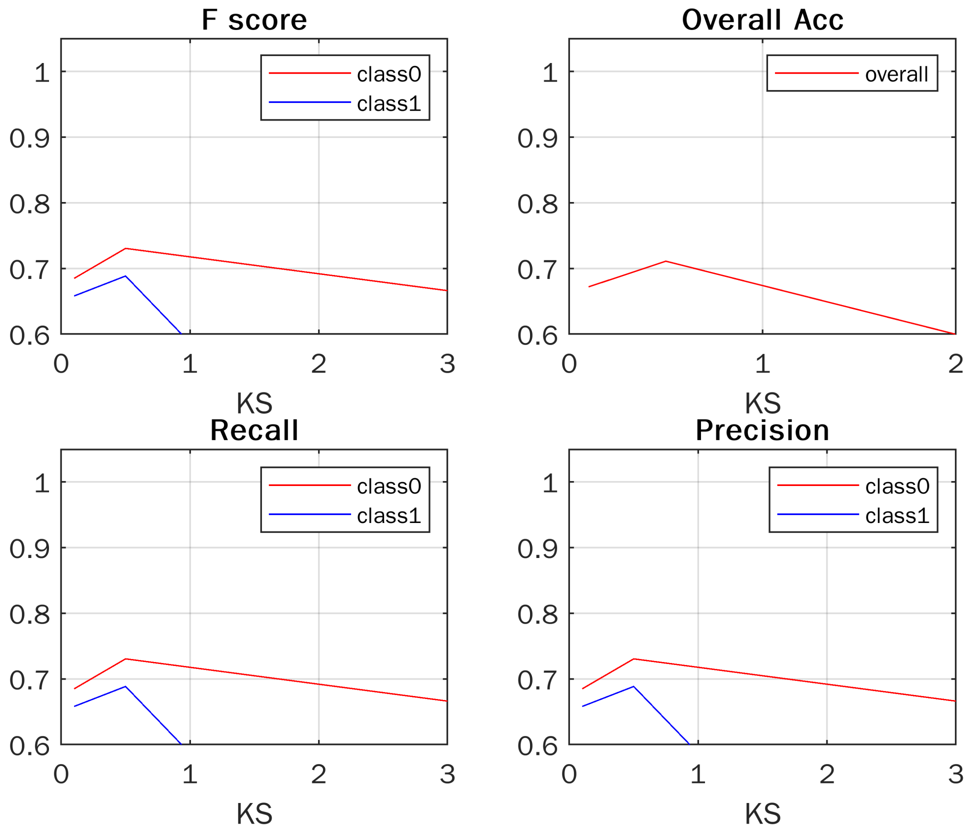
```
OverallAccuracy = 0.7110
KS = 3
numSV = 15816
Confusion Matrix:
        1904         116
        1792         209

Overall accuracy = 0.52549
ClassPerformance = 2×6 table
```

	accuracy	precision	recall	Fscore	sensitivity	specificity
1	0.5255	0.5152	0.9426	0.6662	0.9426	0.1044
2	0.5255	0.6431	0.1044	0.1797	0.1044	0.9426

```
OverallAccuracy = 0.5255
```

```
figure,
subplot(2,2,1), plot(KS, Collect_F(1, :), 'r', KS, Collect_F(2, :), 'b'),
title('F score'), grid on, ylim([0.6, 1.05]), legend({'class0', 'class1'}), xlabel('KS'
subplot(2,2,2), plot(KS, Collect_A(1, :), 'r'),
title('Overall Acc'), grid on, ylim([0.6, 1.05]), legend({'overall'}), xlabel('KS')
subplot(2,2,3), plot(KS, Collect_R(1, :), 'r', KS, Collect_R(2, :), 'b'),
title('Recall'), grid on, ylim([0.6, 1.05]), legend({'class0', 'class1'}), xlabel('KS')
subplot(2,2,4), plot(KS, Collect_P(1, :), 'r', KS, Collect_P(2, :), 'b'),
title('Precision'), grid on, ylim([0.6, 1.05]), legend({'class0', 'class1'}), xlabel('K
```



Train SVM Model with BoxConstraints

```
Collect_F = [];     Collect_R = [];     Collect_P = [];     Collect_A = [];      Colle
BoxCs = [0.1, 3, 5]; %100
for i = BoxCs
    disp(['BoxCs = ' num2str(i)])
    SVM = fitcsvm(Xtrain,Ytrain, 'KernelFunction', 'rbf', 'KernelScale', 1, 'BoxConstra
    [labels score] = predict(SVM, Xtest);
    numSV = size(SVM.SupportVectors,1)
    [ClassPerformance, OverallAccuracy] = CFM_Stats(Ytest, labels)
    Collect_F = [Collect_F, ClassPerformance.Fscore];
    Collect_R = [Collect_R, ClassPerformance.Fscore];
    Collect_P = [Collect_P, ClassPerformance.Fscore];
    Collect_A = [Collect_A, OverallAccuracy];
end
```

```
BoxCs = 0.1
numSV = 15873
Confusion Matrix:
        1977          43
        1937          64

Overall accuracy = 0.50759
ClassPerformance = 2×6 table
```

	accuracy	precision	recall	Fscore	sensitivity	specificity
1	0.5076	0.5051	0.9787	0.6663	0.9787	0.0320

	accuracy	precision	recall	Fscore	sensitivity	specificity
2	0.5076	0.5981	0.0320	0.0607	0.0320	0.9787

```
OverallAccuracy = 0.5076
BoxCs = 3
numSV = 13667
Confusion Matrix:
        1620         400
         516        1485

Overall accuracy = 0.7722
ClassPerformance = 2×6 table
```

	accuracy	precision	recall	Fscore	sensitivity	specificity
1	0.7722	0.7584	0.8020	0.7796	0.8020	0.7421
2	0.7722	0.7878	0.7421	0.7643	0.7421	0.8020

```
OverallAccuracy = 0.7722
BoxCs = 5
numSV = 12577
Confusion Matrix:
        1629         391
         476        1525

Overall accuracy = 0.78438
ClassPerformance = 2×6 table
```

	accuracy	precision	recall	Fscore	sensitivity	specificity
1	0.7844	0.7739	0.8064	0.7898	0.8064	0.7621
2	0.7844	0.7959	0.7621	0.7787	0.7621	0.8064

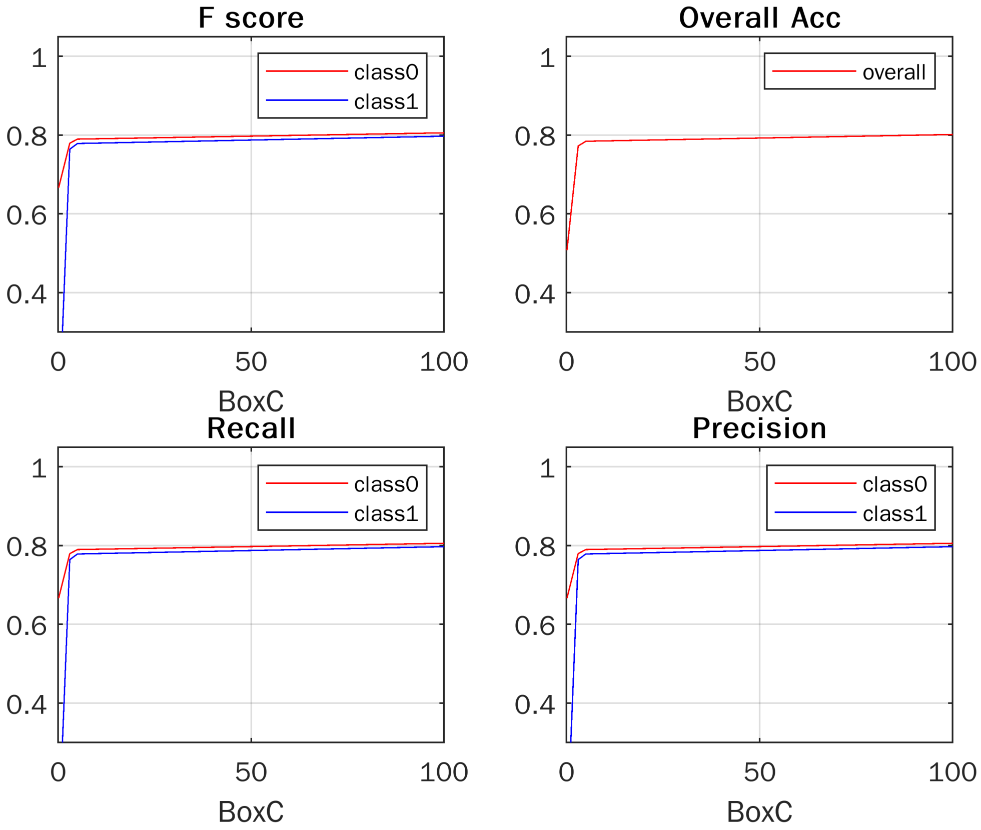
```
OverallAccuracy = 0.7844
BoxCs = 100
numSV = 8231
Confusion Matrix:
        1655         365
         433        1568

Overall accuracy = 0.80154
ClassPerformance = 2×6 table
```

	accuracy	precision	recall	Fscore	sensitivity	specificity
1	0.8015	0.7926	0.8193	0.8057	0.8193	0.7836
2	0.8015	0.8112	0.7836	0.7972	0.7836	0.8193

```
OverallAccuracy = 0.8015
```

```
figure,
subplot(2,2,1), plot(BoxCs, Collect_F(1, :), 'r', BoxCs, Collect_F(2, :), 'b'),
title('F score'), grid on, ylim([0.3, 1.05]), legend({'class0', 'class1'}), xlabel('Box
subplot(2,2,2), plot(BoxCs, Collect_A(1, :), 'r'),
title('Overall Acc'), grid on, ylim([0.3, 1.05]), legend({'overall'}), xlabel('BoxC')
subplot(2,2,3), plot(BoxCs, Collect_R(1, :), 'r', BoxCs, Collect_R(2, :), 'b'),
title('Recall'), grid on, ylim([0.3, 1.05]), legend({'class0', 'class1'}), xlabel('BoxC
subplot(2,2,4), plot(BoxCs, Collect_P(1, :), 'r', BoxCs, Collect_P(2, :), 'b'),
title('Precision'), grid on, ylim([0.3, 1.05]), legend({'class0', 'class1'}), xlabel('B
```



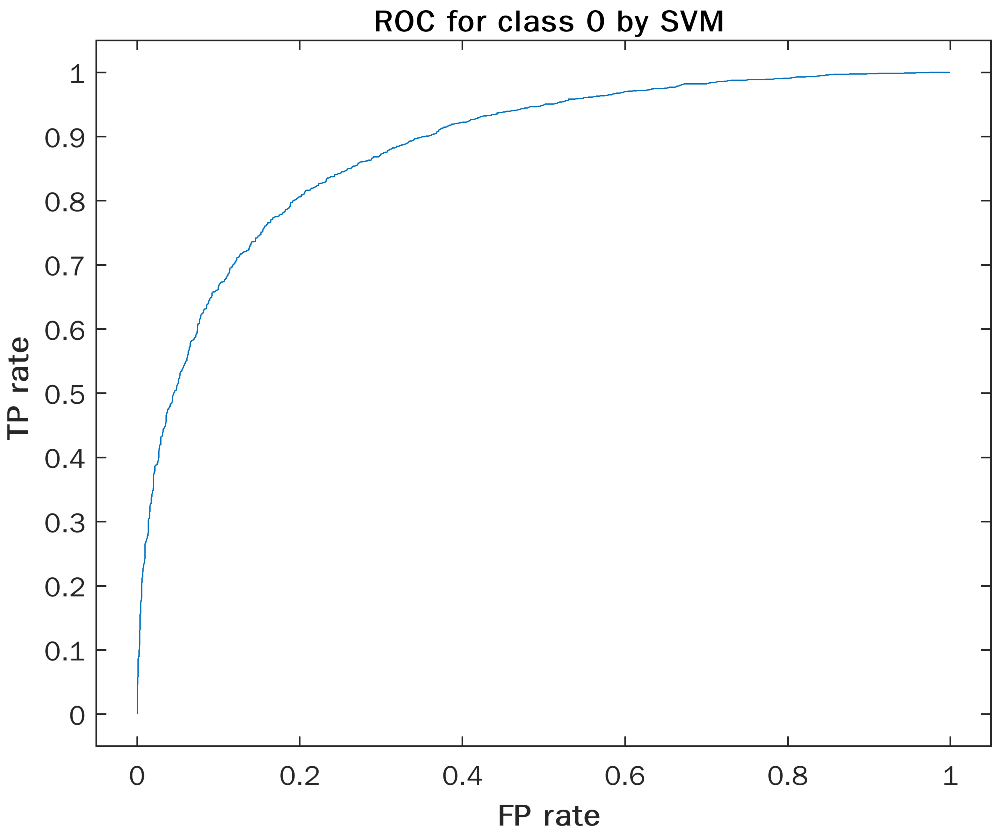
```
return
```

```
SVM.Alpha
SVM.Beta
```

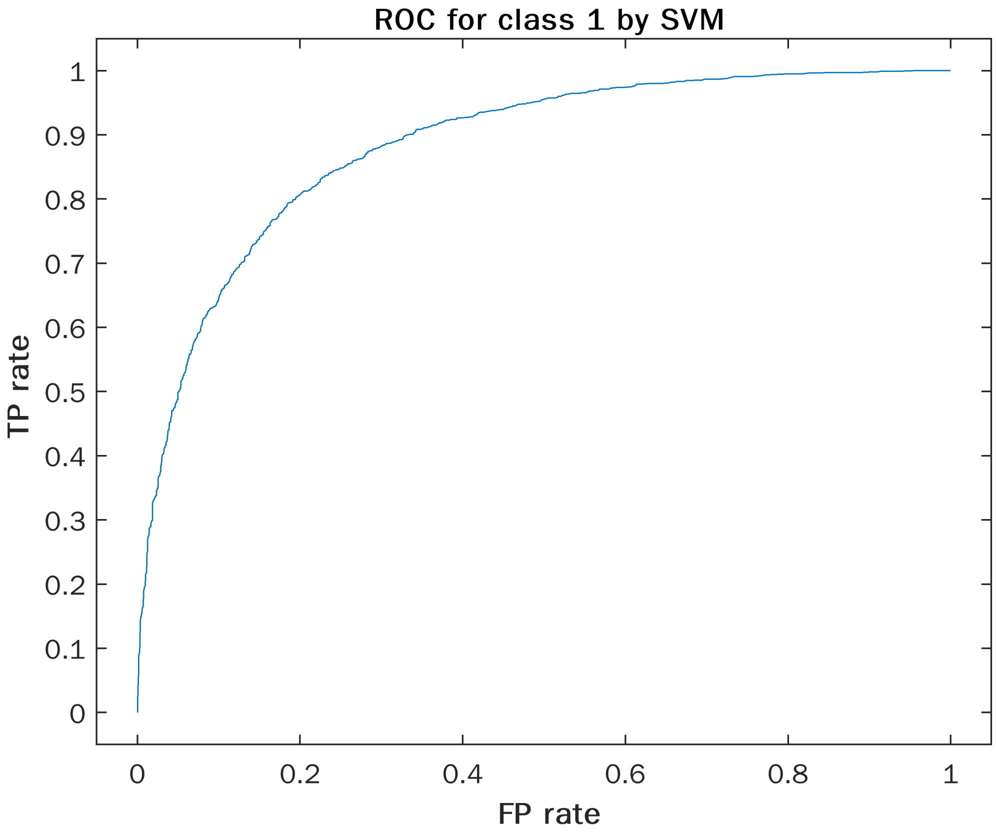
Plots

ROC - with Box Constraint – 5 ; Kernel Scale – 1.

```
[Xcurve0, Ycurve0, T, AUC] = perfcurve(Ytest, score(:,1),0);
figure,
plot(Xcurve0, Ycurve0)
xlim([-0.05 1.05]), ylim([-0.05 1.05]), xlabel('\bf FP rate'), ylabel('\bf TP rate')
title('\bf ROC for class 0 by SVM')
```



```
[Xcurve1, Ycurve1, T, AUC] = perfcurve(Ytest, score(:,2),1);
figure,
plot(Xcurve1, Ycurve1)
xlim([-0.05 1.05]), ylim([-0.05 1.05]), xlabel('\bf FP rate'),  ylabel('\bf TP rate')
title('\bf ROC for class 1 by SVM')
```


ROC for class 1 by SVM
TP rate
FP rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1