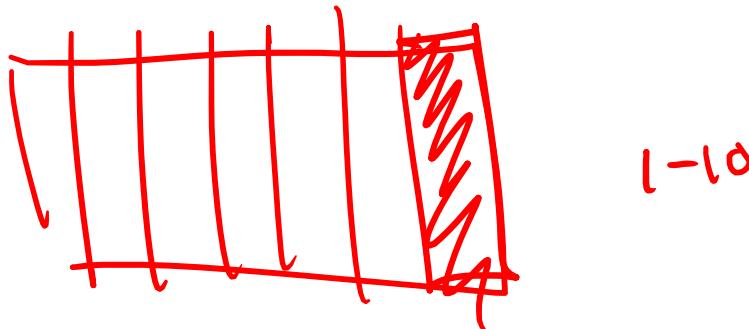




Wine Quality Prediction

SEIS 763: Machine Learning
Presenter: Garth Mortensen

Purpose



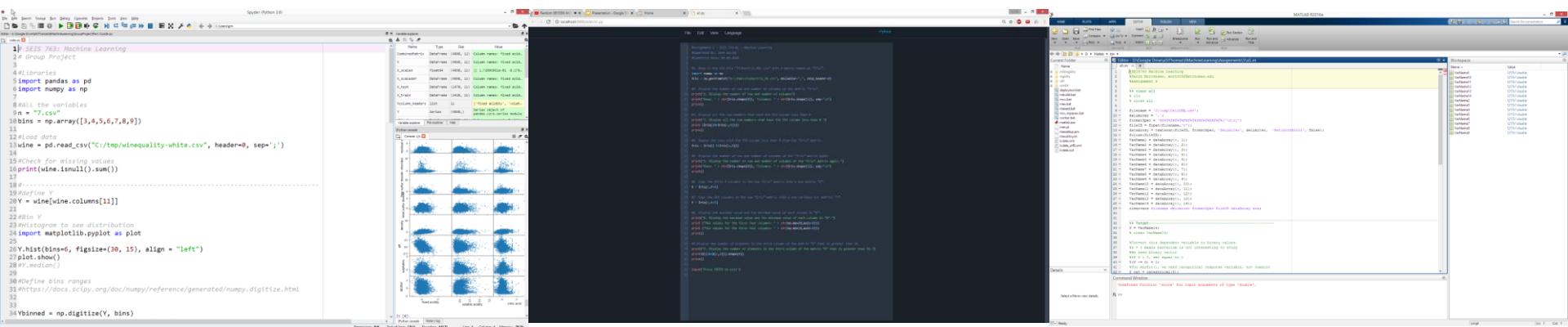
Given various attributes such as acidity,
chlorides, pH and alcohol, identify which
category (Good/Bad) a new wine bottle belongs
to, using classification.

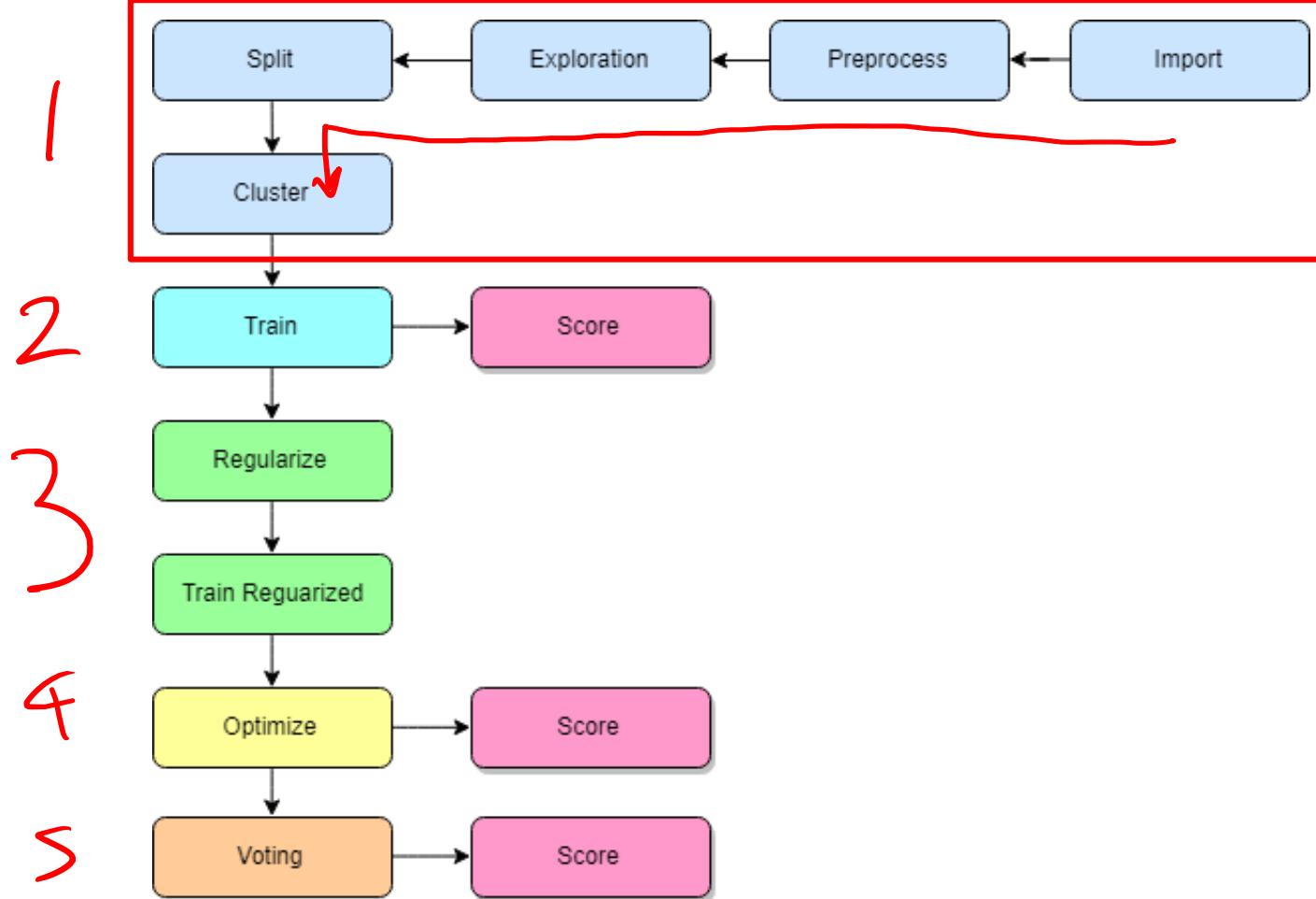
Find the best model, then optimize it.

Software/Tools Used

- Python (data loading, prep, splitting)
 - Spyder
 - Jupyter
- MatLab (modeling, scoring, optimizing)

$\Sigma\text{LOC} = 900+$





Data Set & Cleaning

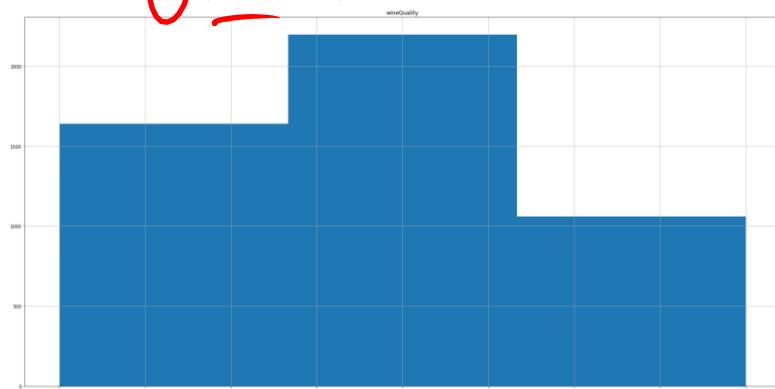
- Data **skewed** between Red and White wine
 - Red = $1,599 \times 12$
 - White $< 4,898 \times 12$
- No NULL entries
- Predictors
 - All are **numerical**
 - All have been **standardized**
- **Target**
 - **Categorical (3, 9)**

```
1 # Group Project
2
3 # Libraries
4 import pandas as pd
5 import numpy as np
6
7 # null the variables
8 n = "Wine.csv"
9 bins = np.array([3,4,5,6,7,8,9])
10
11 # load data
12 wine = pd.read_csv("C:/temp/winequality-white.csv", header=0, sep=';')
13
14 # check for missing values
15 print(wine.isnull().sum())
16
17 # drop
18 wine.dropna()
19 Y = wine[wine.columns[11]]
20
21 X = wine
22
23 # distribution to see distribution
24 import matplotlib.pyplot as plt
25
26 V_hist(bins, figsize(10, 10), align = "left")
27 plt.show()
28
29
30 # define bins ranges
31 #http://docs.scipy.org/doc/numpy/reference/generated/numpy.digitize.html
32
33 Vbinned = np.digitize(Y, bins)
34
35 # some bins are unequally distributed
36 # active column headers
37 Vbinneddf = pd.DataFrame(Vbinned)
38 Vbinneddf.columns = ["wineQuality"]
39
40
41
42 Vbinneddf.hist(bins=bins, figsize(10, 10))
43 plt.show()
44
45 # histogram to see final distribution
46 Vbinneddf.hist(bins=bins, figsize(10, 10))
47 plt.show()
48
49 # one-hot encoding
50 Vbinneddf_ohehot = pd.get_dummies(Vbinneddf, columns=["wineQuality"], prefix = ["wineQuality"])
51 Vbinneddf_ohehot.columns = ["3", "4", "5", "6", "7", "8"]
52
53 # afterdf Y
54 VFinal = Vbinneddf
55
56
57 #-----#
58 #-----#
59 #-----#
60 #-----#
61 X = wine[wine.columns[0:11]]
62 Xcolumn_headers = list(X) #because standardization removes column headers
63
64 #standardize X
65 from sklearn import preprocessing
66 X_scaled = preprocessing.scale(X, axis=0)
67
68 #standardizing converted from dataframe. Convert back
69 X_scaleddf = pd.DataFrame(X_scaled)
70
71 #bring column headers back in
72 X_scaleddf.columns = Xcolumn_headers
73
74 #-----#
75 #-----#
76 #export for one Large table export
77 X_scaleddf
78 VFinal
79
80 #-----#
81 #advise the data
82 X.info()
83 X.describe()
84
85 #Cross Correlation Check
86 from pandas.plotting import scatter_matrix
87 scatter_matrix(X_scaleddf, figsize(10, 10))
88
89 #-----#
90 #-----#
91 #split the data
92 # create training and testing vars
93 from sklearn.model_selection import train_test_split
94 from sklearn.model_selection import train_test_split
95
96 X_train, X_test, y_train, y_test = train_test_split(X_scaleddf, VFinal, test_size=0.3, random_state=123)
97
98 #-----#
99 # merge for export
100 combined_train = pd.concat([X_train, y_train], axis=1)
101 combinedTest = pd.concat([X_test, y_test], axis=1)
102
103
104 #print("X and Y's X var", "boil")
105 combined_train.to_csv("C:/temp/wine_train.csv", index=False, header=True)
106 combinedTest.to_csv("C:/temp/wine_test.csv", index=False, header=True)
107
108 CombinedMatrix = pd.concat([X_scaleddf, VFinal], axis=1)
109 CombinedMatrix.to_csv("C:/temp/wine_all.csv", index=False, header=True)
110 CombinedMatrix.to_csv("C:/temp/wine_all.csv", index=False, header=True)
```

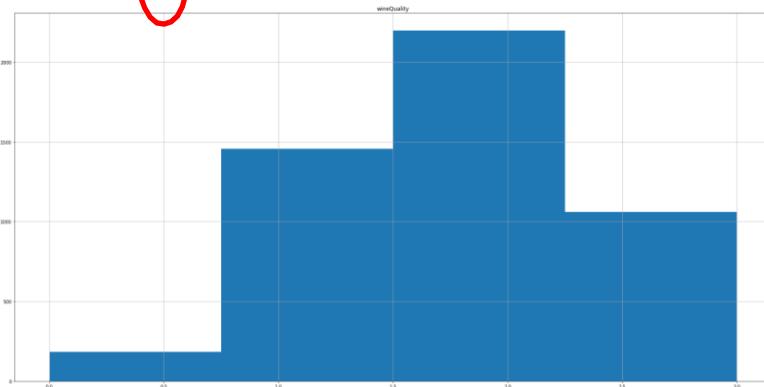
Y bins = 2 (6, 10)



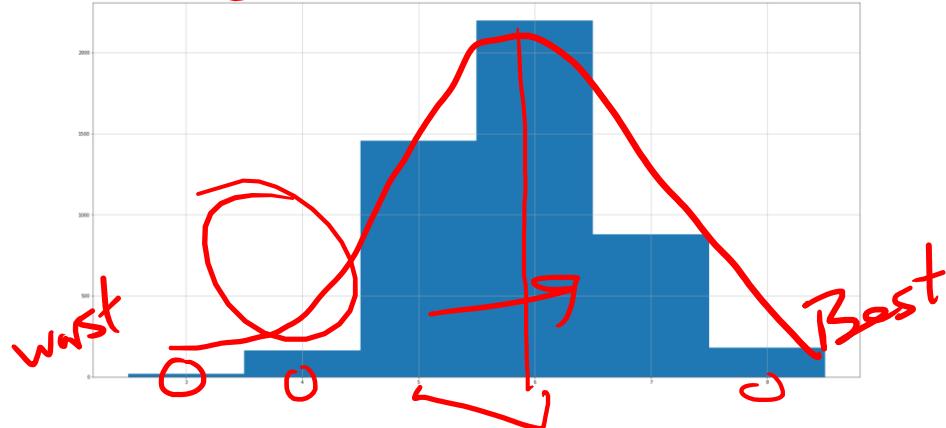
Y bins = 3 (6, 7, 10)



Y bins = 4 (5, 6, 7, 10)



Y bins = 7 (default)



Correlation check

%numeric correlation

[correlation, pval] = corr([alcohol, chlorides, citricacid, density, fixed acidity, free sulfur dioxide, pH, residual sugar, sulphates, total sulfur dioxide, volatile acidity]);

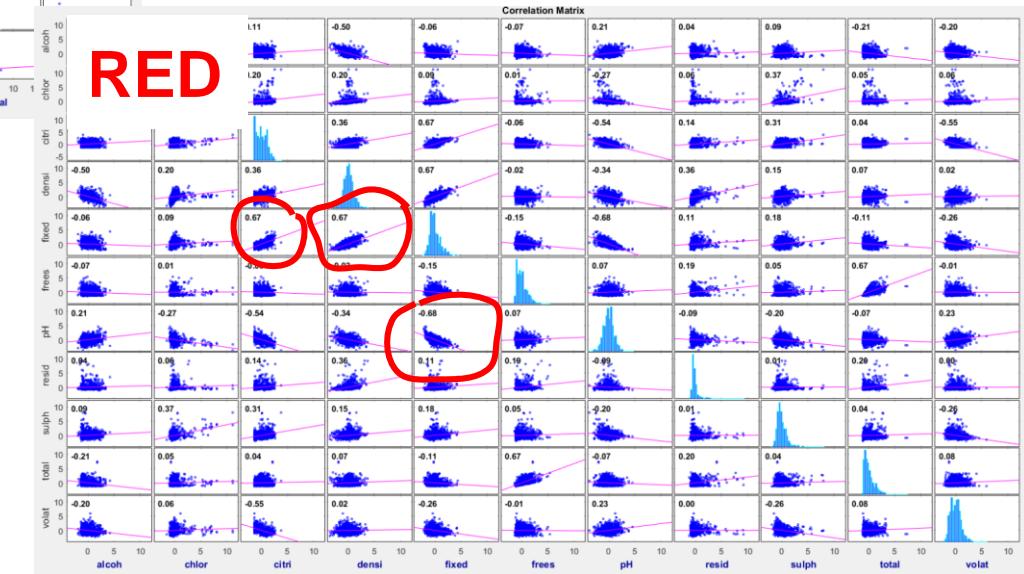
Max(abs(corr))

White Wine Correlation Matrix											Red Wine Correlation Matrix										
1.000	-0.360	0.076	-0.780	0.121	-0.250	0.121	0.451	-0.017	-0.449	0.068	1.000	-0.221	0.110	-0.496	-0.062	-0.069	0.206	0.042	0.094	-0.206	-0.202
-0.360	1.000	0.114	0.257	0.023	0.101	-0.090	0.083	0.017	0.199	0.071	-0.221	1.000	0.204	0.201	0.094	0.006	-0.265	0.056	0.371	0.047	0.061
0.076	0.114	1.000	0.150	0.289	0.094	-0.164	0.094	0.062	0.121	-0.149	0.110	0.204	1.000	0.365	0.672	-0.061	-0.542	0.144	0.313	0.036	-0.552
-0.780	0.257	0.150	1.000	0.265	0.294	-0.694	0.839	0.074	0.530	0.027	-0.496	0.201	0.365	1.000	0.668	-0.022	-0.342	0.355	0.149	0.071	0.022
0.121	0.023	0.289	0.265	1.000	-0.049	-0.426	0.089	-0.017	0.091	-0.023	-0.062	0.094	0.672	0.668	1.000	-0.154	-0.683	0.115	0.183	-0.113	-0.256
-0.250	0.101	0.094	0.294	-0.049	1.000	-0.001	0.293	0.059	0.616	-0.097	-0.069	0.006	-0.061	-0.022	-0.154	1.000	0.070	0.187	0.052	0.668	-0.011
0.121	-0.090	-0.164	-0.094	-0.426	-0.001	1.000	-0.194	0.156	0.002	-0.032	0.206	-0.265	-0.542	-0.342	-0.683	0.070	1.000	-0.086	-0.197	-0.066	0.235
-0.451	0.089	0.094	0.839	0.089	0.299	-0.194	1.000	-0.027	0.401	0.064	0.042	0.056	0.144	0.355	0.115	0.187	-0.086	1.000	0.006	0.203	0.002
-0.017	0.017	0.062	0.074	-0.017	0.059	0.156	-0.027	1.000	0.135	-0.036	0.094	0.371	0.313	0.149	0.183	0.052	-0.197	0.006	1.000	0.043	-0.261
-0.449	0.199	0.121	0.530	0.091	0.616	0.002	0.401	0.135	1.000	0.089	-0.206	0.047	0.036	0.071	-0.113	0.668	0.066	0.203	0.043	1.000	0.076
0.068	0.071	-0.149	0.027	-0.023	-0.097	-0.032	0.064	-0.036	0.089	1.000	-0.202	0.061	-0.552	0.022	-0.256	-0.011	0.235	0.002	-0.261	0.076	1.000



Sugar ↑
density ↑

alcohol ↑
density ↓

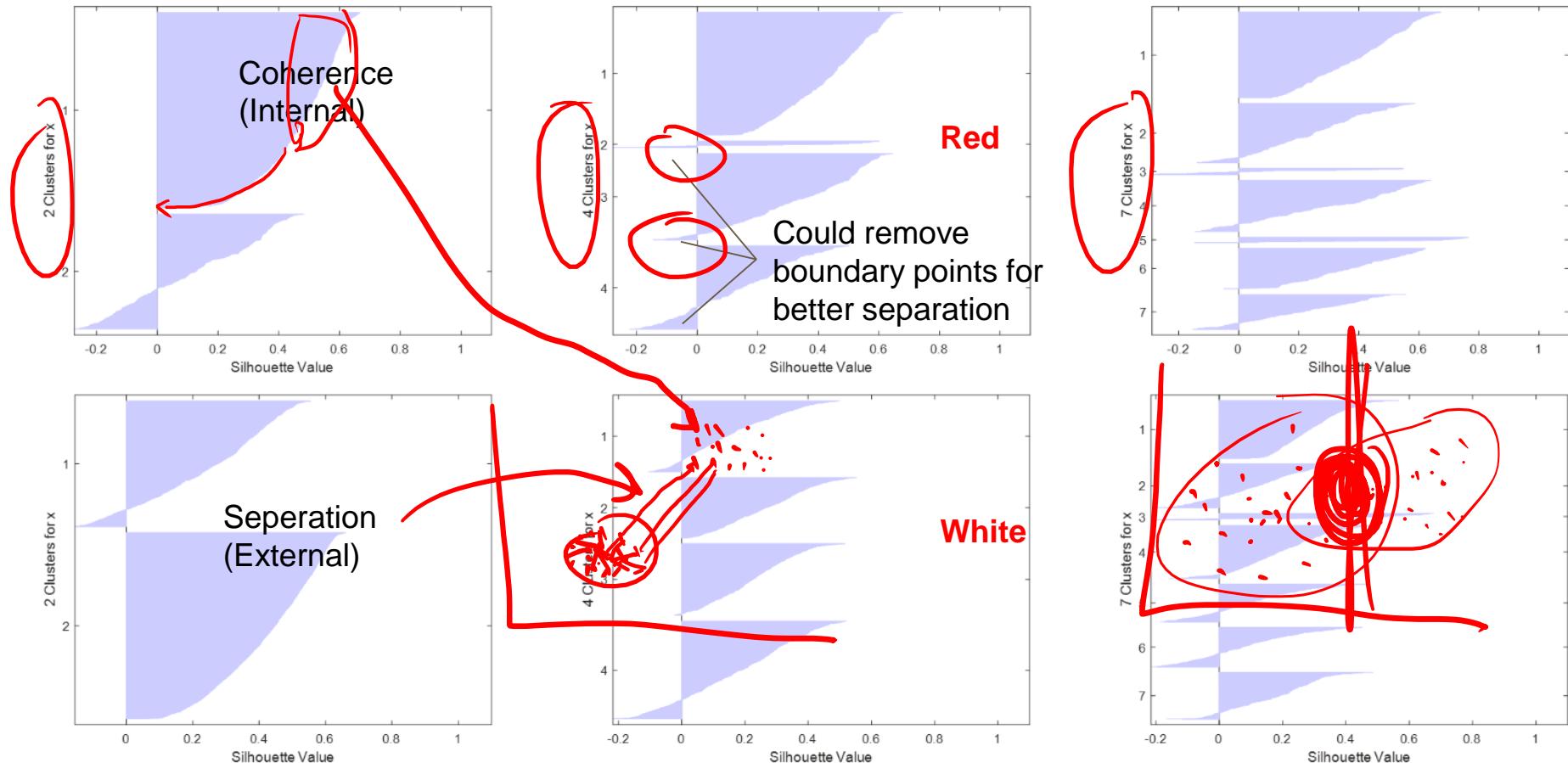


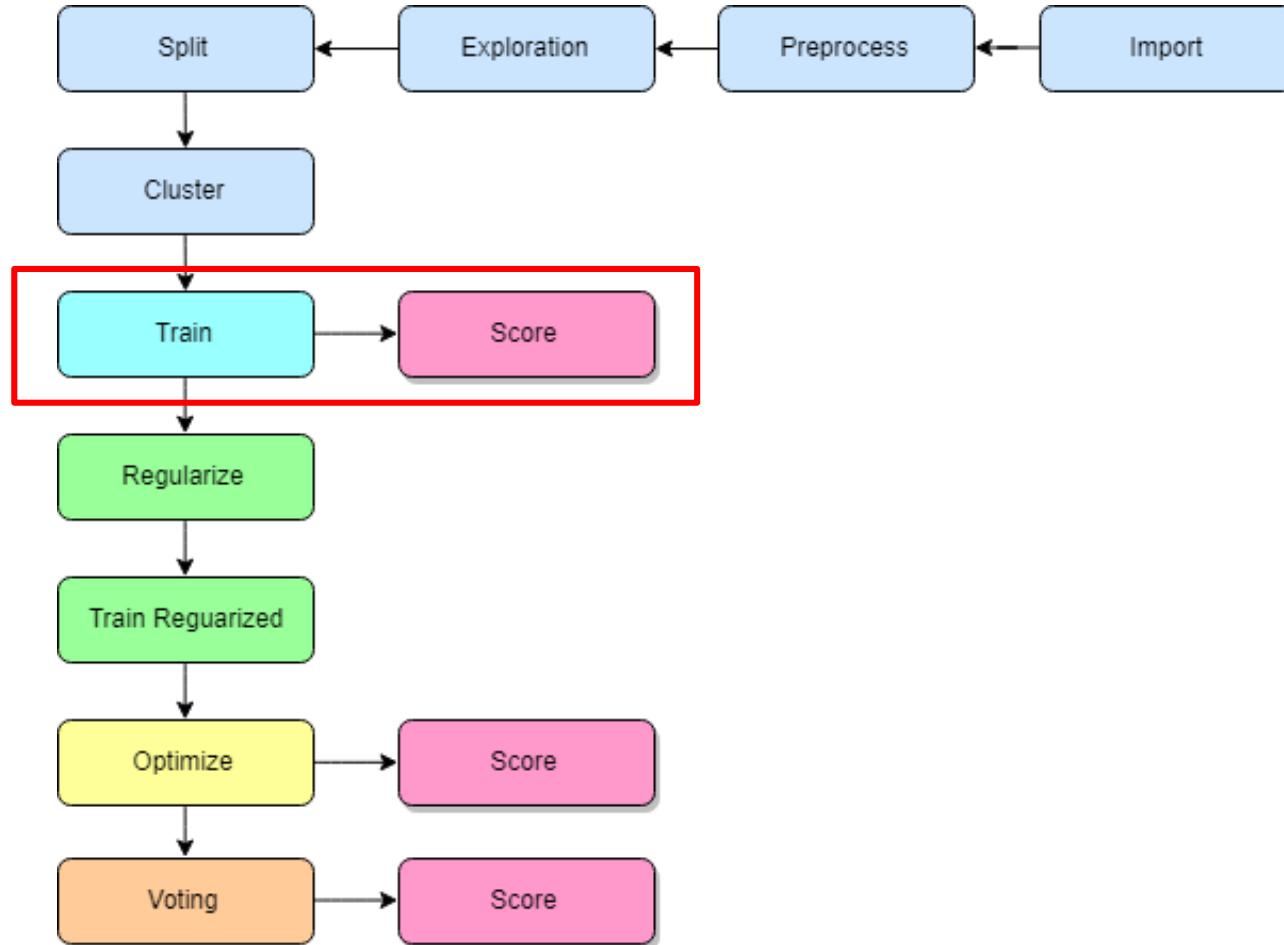
RED

Randomized 70-30 Split

- `X_train, X_test, y_train, y_test = train_test_split(X_scaleddf, YFinal, test_size=0.3, random_state=123)`
 - Data split **70-30** between training and testing
 - **Random** seed = 123

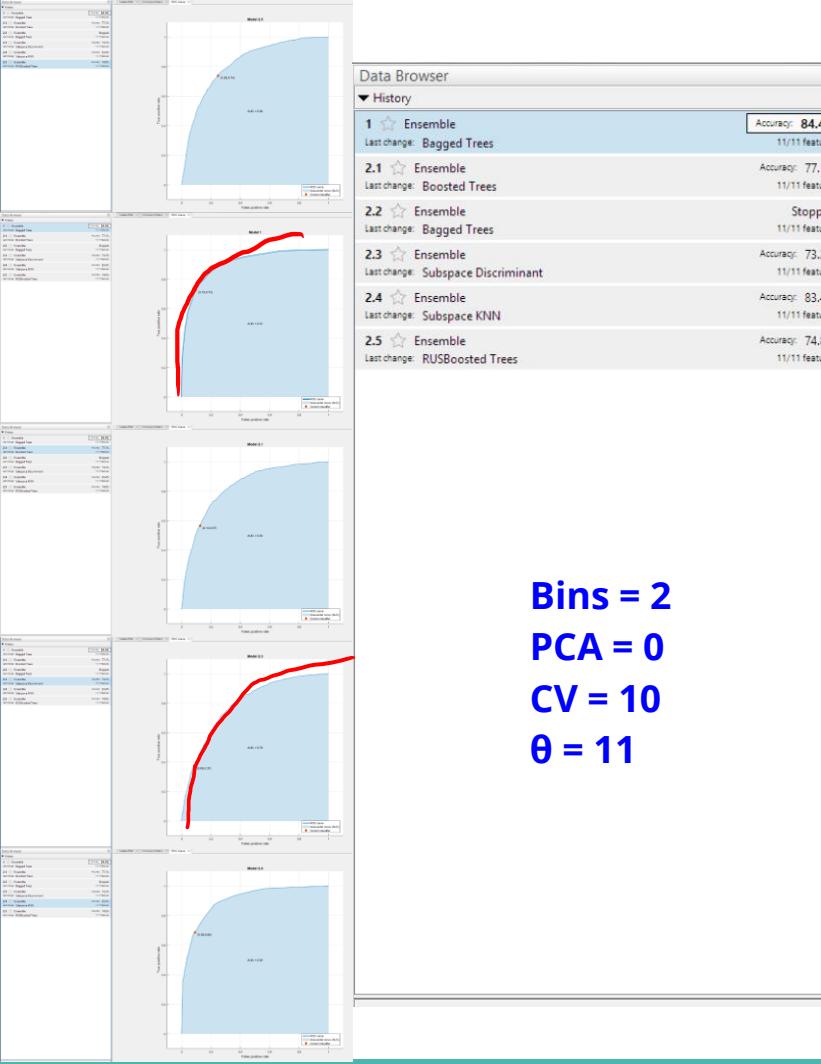
Any Natural Cluster? K-Means



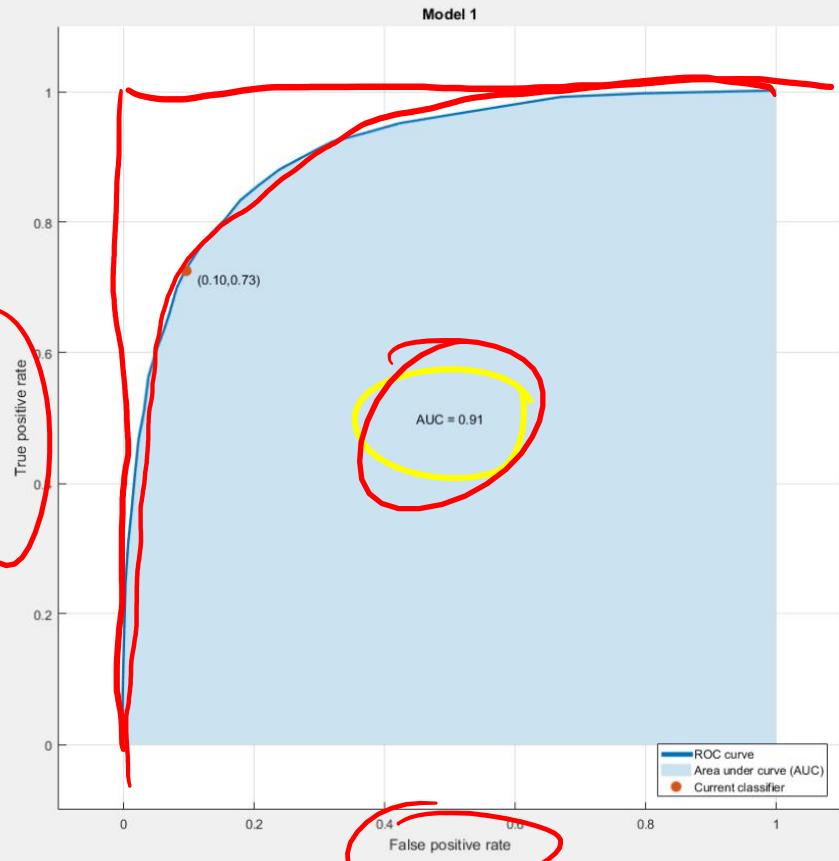
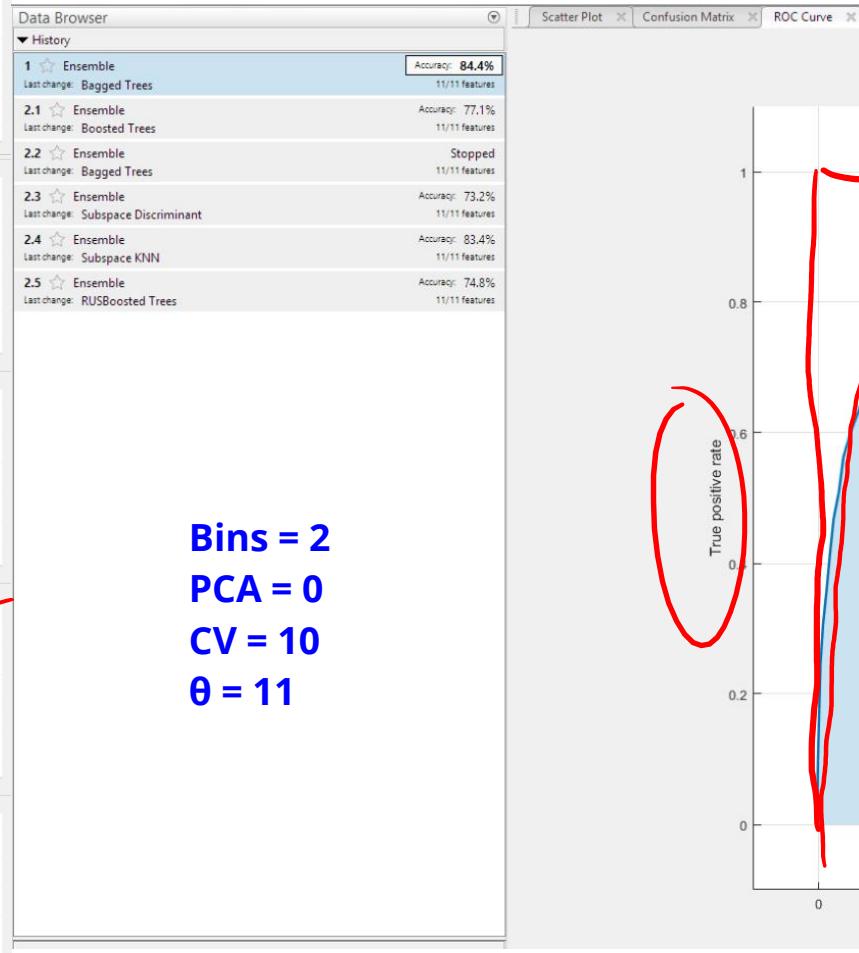


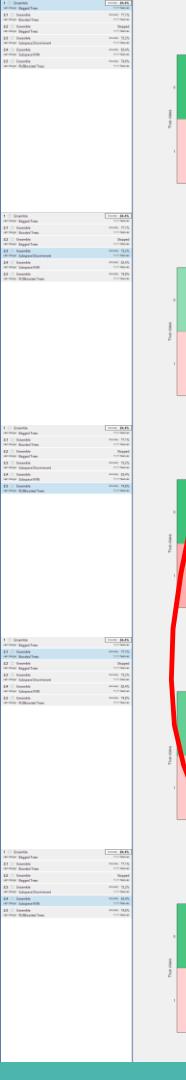
The figure displays a 10x10 grid of 100 decision tree models, likely from a random forest classifier. Each model's performance is summarized by four metrics: Bins, PCA, CV, and θ. The metrics are circled in red for emphasis.

Row	Column	Bins	PCA	CV	θ
1	1	2	0	10	11
1	2	2	1	10	11
1	3	2	1	10	11
1	4	2	1	10	11
1	5	2	1	10	11
1	6	2	1	10	11
1	7	2	1	10	11
1	8	2	1	10	11
1	9	2	1	10	11
1	10	2	1	10	11
2	1	4	0	10	11
2	2	4	1	10	11
2	3	4	1	10	11
2	4	4	1	10	11
2	5	4	1	10	11
2	6	4	1	10	11
2	7	4	1	10	11
2	8	4	1	10	11
2	9	4	1	10	11
2	10	4	1	10	11
3	1	7	0	10	11
3	2	7	0	10	11
3	3	7	0	10	11
3	4	7	0	10	11
3	5	7	0	10	11
3	6	7	0	10	11
3	7	7	0	10	11
3	8	7	0	10	11
3	9	7	0	10	11
3	10	7	0	10	11
4	1	7	0	10	11
4	2	7	0	10	11
4	3	7	0	10	11
4	4	7	0	10	11
4	5	7	0	10	11
4	6	7	0	10	11
4	7	7	0	10	11
4	8	7	0	10	11
4	9	7	0	10	11
4	10	7	0	10	11
5	1	7	0	10	11
5	2	7	0	10	11
5	3	7	0	10	11
5	4	7	0	10	11
5	5	7	0	10	11
5	6	7	0	10	11
5	7	7	0	10	11
5	8	7	0	10	11
5	9	7	0	10	11
5	10	7	0	10	11
6	1	7	0	10	11
6	2	7	0	10	11
6	3	7	0	10	11
6	4	7	0	10	11
6	5	7	0	10	11
6	6	7	0	10	11
6	7	7	0	10	11
6	8	7	0	10	11
6	9	7	0	10	11
6	10	7	0	10	11
7	1	7	0	10	11
7	2	7	0	10	11
7	3	7	0	10	11
7	4	7	0	10	11
7	5	7	0	10	11
7	6	7	0	10	11
7	7	7	0	10	11
7	8	7	0	10	11
7	9	7	0	10	11
7	10	7	0	10	11
8	1	7	0	10	11
8	2	7	0	10	11
8	3	7	0	10	11
8	4	7	0	10	11
8	5	7	0	10	11
8	6	7	0	10	11
8	7	7	0	10	11
8	8	7	0	10	11
8	9	7	0	10	11
8	10	7	0	10	11
9	1	7	0	10	11
9	2	7	0	10	11
9	3	7	0	10	11
9	4	7	0	10	11
9	5	7	0	10	11
9	6	7	0	10	11
9	7	7	0	10	11
9	8	7	0	10	11
9	9	7	0	10	11
9	10	7	0	10	11
10	1	7	0	10	11
10	2	7	0	10	11
10	3	7	0	10	11
10	4	7	0	10	11
10	5	7	0	10	11
10	6	7	0	10	11
10	7	7	0	10	11
10	8	7	0	10	11
10	9	7	0	10	11
10	10	7	0	10	11



Bins = 2
PCA = 0
CV = 10
 $\theta = 11$





1 ★ Ensemble
Last change: Bagged Trees Accuracy: 84.4%
11/11 features

2.1 ★ Ensemble
Last change: Boosted Trees Accuracy: 77.1%
11/11 features

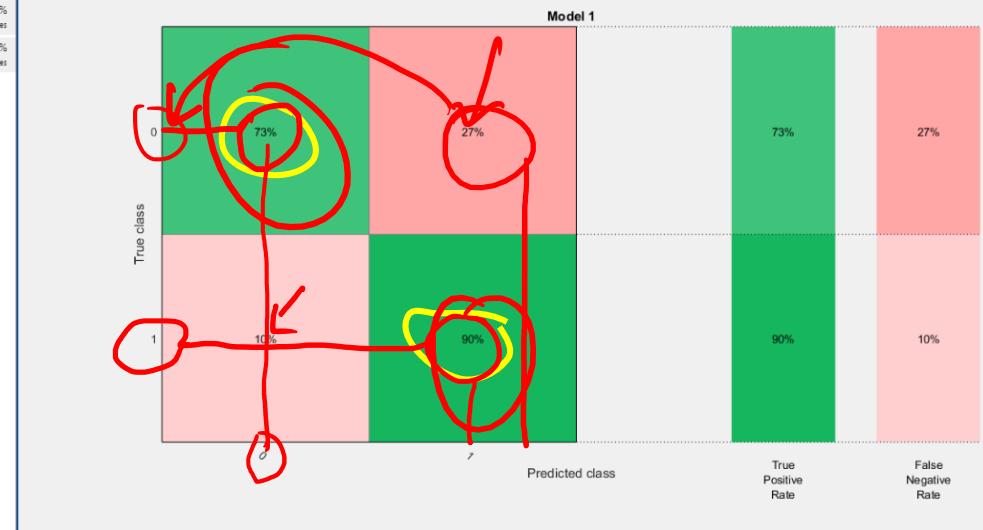
2.2 ★ Ensemble
Last change: Bagged Trees Stopped
11/11 features

2.3 ★ Ensemble
Last change: Subspace Discriminant Accuracy: 73.2%
11/11 features

2.4 ★ Ensemble
Last change: Subspace KNN Accuracy: 83.4%
11/11 features

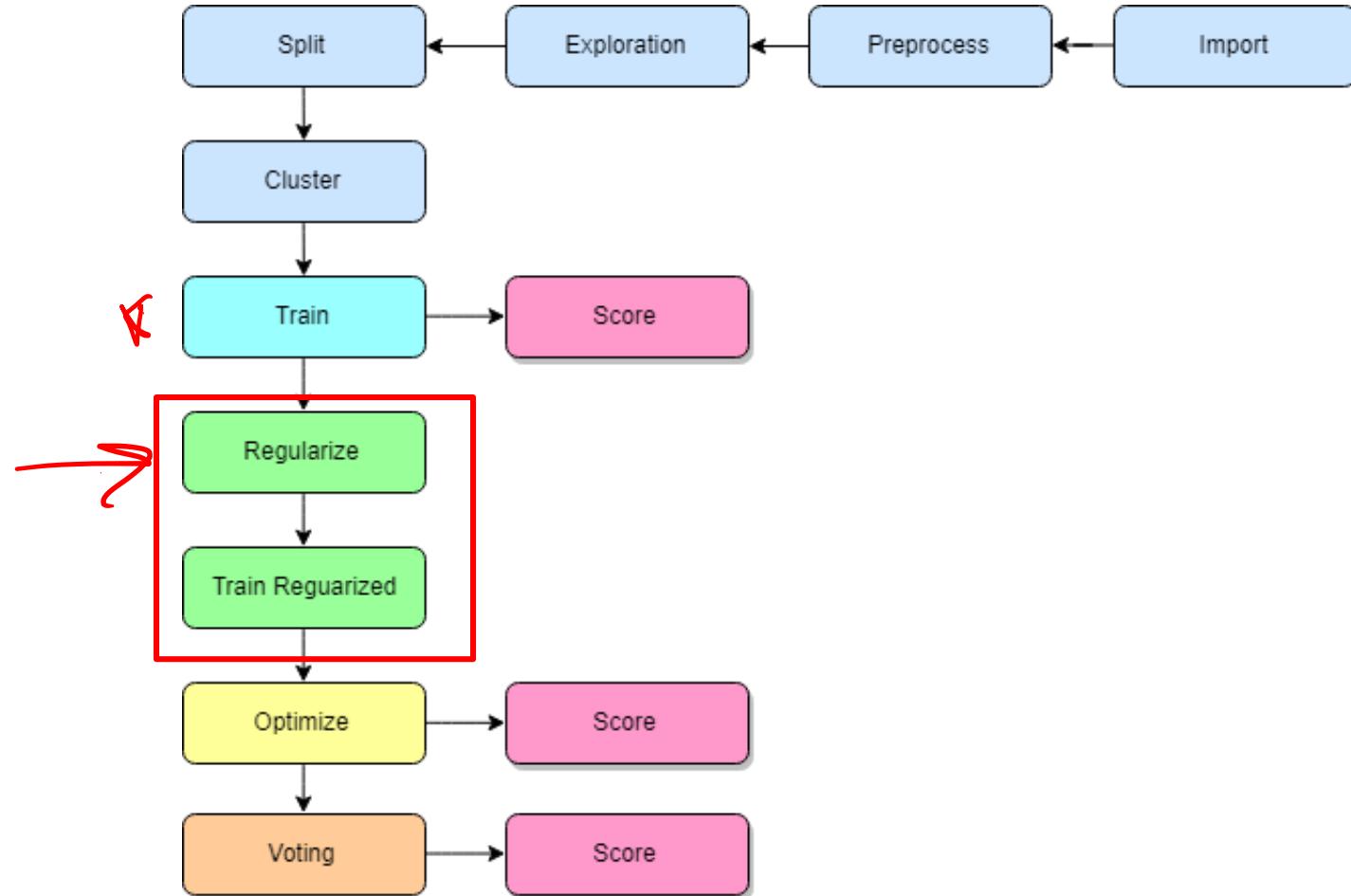
2.5 ★ Ensemble
Last change: RUSBoosted Trees Accuracy: 74.8%
11/11 features

Bins = 2
PCA = 0
CV = 10
 $\theta = 11$



270 Pre-Regularized Models

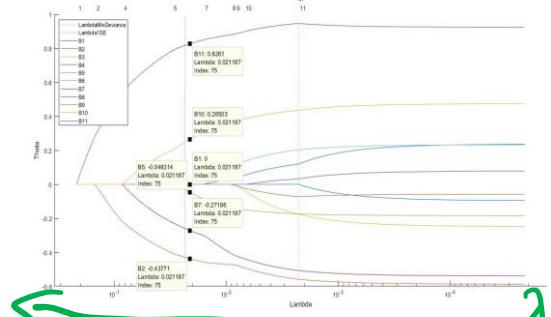
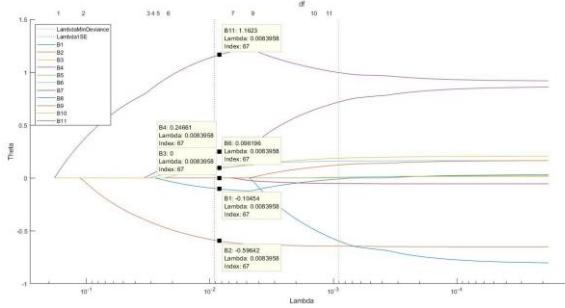
- 2 bins
 - Ensemble > KNN > SVM
- 4 bins
 - Ensemble > KNN > SVM
- 7 bins
 - Ensemble > KNN > SVM
- *Bagged Tree Ensemble was most Accurate*



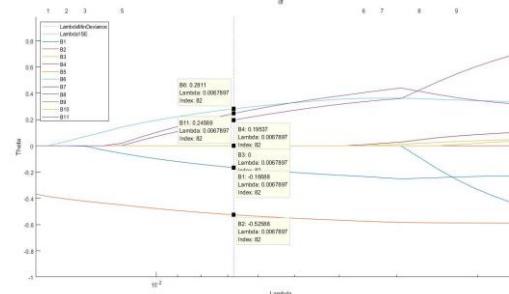
Lasso Regularization

2 bin: volatileacidity, chlorides, totalsulfurdioxide, sulphates, alcohol

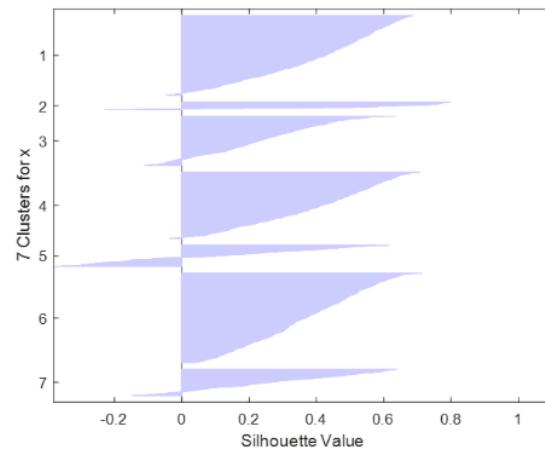
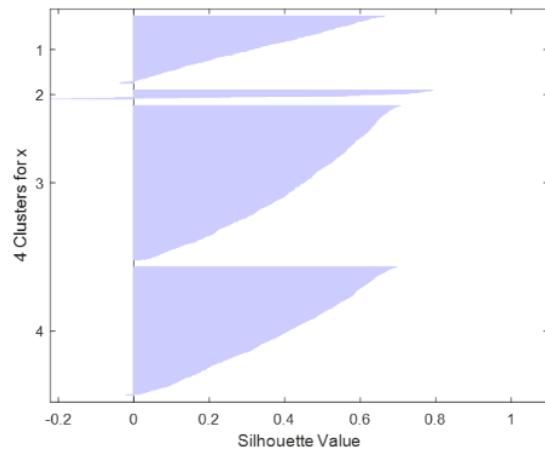
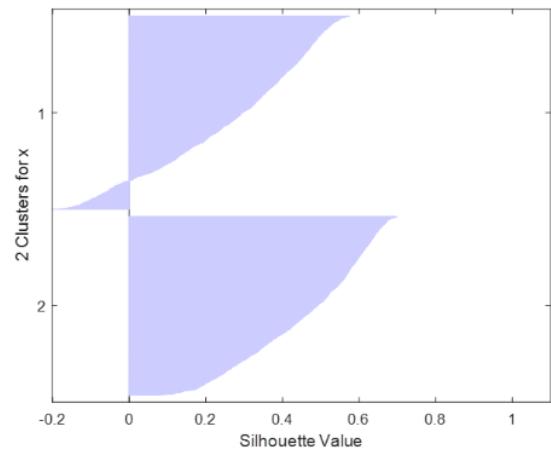
4 bin: volatileacidity, freesulfurdioxide, density, alcohol



7 bin: freesulfurdioxide, volatileacidity, density, chlorides, sulphates, alcohol



Again: Any Natural Cluster? K-Means



Before: cluster2 = 0.3533, cluster4 = 0.2012, cluster7 = 0.1741
After: cluster2 = 0.3562, cluster4 = 0.3895, cluster7 = 0.3344

WHITE

WHITE

- 1.6 Logistic Regression
- 1.7 Linear Regression
- 1.8 Linear SVM
- 1.9 Quadratic SVM
- 1.10 SVM
- 1.11 Gaussian SVM
- 1.12 Medium SVM
- 1.13 Decision Function SVM
- 1.14 KNN
- 1.15 Linear KNN
- 1.16 Cosine KNN
- 1.17 Cosine Similarity
- 1.18 Cosine KNN

Bins = 2
gaussian SVM
PCA = 1
CV = 10
θ = 5

ins = 2
CA = 0
V = 0
= 5

	PCA = 1	CV = 10	0 = 5
1.0 Natural	Linear SVM	Linear SVM	Linear SVM
1.1 Stopped	Linear SVM	Linear SVM	Linear SVM
1.2 A/k Natural	Linear SVM	Linear SVM	Linear SVM
1.3 Stopped	Linear SVM	Linear SVM	Linear SVM
1.4 A/k Natural	Linear SVM	Linear SVM	Linear SVM
1.5 Stopped	Linear SVM	Linear SVM	Linear SVM
1.6 A/k Natural	Linear SVM	Linear SVM	Linear SVM
1.7 Stopped	Linear SVM	Linear SVM	Linear SVM
1.8 A/k Natural	Linear SVM	Linear SVM	Linear SVM
1.9 Stopped	Cubic SVM	Cubic SVM	Cubic SVM
1.0 A/k Natural	Fit Host	Fit Host	Fit Host
1.1 Stopped	Medium Gaussian SVM	Medium Gaussian SVM	Medium Gaussian SVM
1.2 A/k Natural	Medium Gaussian SVM	Medium Gaussian SVM	Medium Gaussian SVM
1.3 Stopped	KNN	KNN	KNN
1.4 A/k Natural	KNN	KNN	KNN
1.5 Stopped	KNN	KNN	KNN
1.6 A/k Natural	KNN	KNN	KNN
1.7 Stopped	KNN	KNN	KNN
1.8 A/k Natural	KNN	KNN	KNN
1.9 Stopped	Coarse KNN	Coarse KNN	Coarse KNN

Bins = 4
PCA = 0
CV = 0
θ = 5

WHITE

- 1.8 SVM
- 1.9 Linear SVM
- 1.7 SVM
- 1.6 Quadratic SVM
- 1.5 Fine Gaussian SVM
- 1.4 Leverage Method
- 1.3 Coarse Gaussian SVM
- 1.2 KNN
- 1.1 Feature Selection
- 1.0 KNN
- 0.9 Coarse KNN
- 0.8 Linear KNN
- 0.7 Feature KNN
- 0.6 Quadratic KNN
- 0.5 Coarse KNN

Bins = 7
PCA = 1
CV = 10
θ = 5

Linear SVM
Sigmoid SVM
RBF SVM
Bins = 7
Gaussian SVM
PCA = 0
KNN
CV = 0
LR

WHITE

Bins = 2
PCA = 0
CV = 10
 $\theta = 5$

Last change: Simple Tree	1.1 ★ Tree Accuracy: 73.2% 5/5 features
1.4 ★ Linear Discriminant Last change: Linear Discriminant	1.2 ★ Tree Accuracy: 71.7% 5/5 features
1.5 ★ Quadratic Discriminant Last change: Quadratic Discriminant	1.3 ★ Tree Accuracy: 70.9% 5/5 features
1.6 ★ Logistic Regression Last change: Logistic Regression	1.4 ★ Linear Discriminant Accuracy: 69.1% 5/5 features
1.7 ★ SVM Last change: Linear SVM	1.5 ★ Quadratic Discriminant Accuracy: 68.6% 5/5 features
1.8 ★ SVM Last change: Quadratic SVM	1.6 ★ Logistic Regression Accuracy: 69.2% 5/5 features
1.9 ★ SVM Last change: Cubic SVM	1.7 ★ SVM Accuracy: 67.5% 5/5 features
1.10 ★ SVM Last change: Fine Gaussian SVM	1.8 ★ SVM Stopped 5/5 features
1.11 ★ SVM Last change: Medium Gaussian SVM	1.9 ★ SVM Stopped 5/5 features
1.12 ★ SVM Last change: Coarse Gaussian SVM	1.10 ★ SVM Accuracy: 75.8% 5/5 features
1.13 ★ KNN Last change: Fine KNN	1.11 ★ SVM Accuracy: 72.7% 5/5 features
1.14 ★ KNN Last change: Medium KNN	1.12 ★ SVM Accuracy: 68.4% 5/5 features
1.15 ★ KNN Last change: Coarse KNN	1.13 ★ KNN Accuracy: 79.0% 5/5 features
1.16 ★ KNN Last change: Cosine KNN	1.14 ★ KNN Accuracy: 71.9% 5/5 features
1.17 ★ KNN Last change: Cubic KNN	1.15 ★ KNN Accuracy: 71.5% 5/5 features
1.18 ★ KNN Last change: Weighted KNN	1.16 ★ KNN Accuracy: 72.3% 5/5 features
1.19 ★ Ensemble Last change: Boosted Trees	1.17 ★ KNN Accuracy: 71.4% 5/5 features
1.20 ★ Ensemble Last change: Bagged Trees	1.18 ★ KNN Accuracy: 79.9% 5/5 features
1.21 ★ Ensemble Last change: Subspace Discriminant	1.19 ★ Ensemble Accuracy: 80.4% 5/5 features
1.22 ★ Ensemble Last change: Subspace KNN	1.20 ★ Ensemble Last change: Bagged Trees Accuracy: 80.4% 5/5 features
1.23 ★ Ensemble Last change: RUSBoosted Trees	1.21 ★ Ensemble Last change: Subspace Discriminant Accuracy: 80.6% 5/5 features

Bins = 2
PCA = 1
CV = 10
 $\theta = 5$

1.1 ★ Tree Accuracy: 72.8% 5/5 features	1.1 ★ Tree Accuracy: 72.8% 5/5 features
1.2 ★ Tree Accuracy: 71.6% 5/5 features	1.2 ★ Tree Accuracy: 71.6% 5/5 features
1.3 ★ Tree Accuracy: 70.2% 5/5 features	1.3 ★ Tree Accuracy: 70.2% 5/5 features
1.4 ★ Linear Discriminant Last change: Linear Discriminant	1.4 ★ Linear Discriminant Last change: Linear Discriminant
1.5 ★ Quadratic Discriminant Last change: Quadratic Discriminant	1.5 ★ Quadratic Discriminant Last change: Quadratic Discriminant
1.6 ★ Logistic Regression Last change: Logistic Regression	1.6 ★ Logistic Regression Last change: Logistic Regression
1.7 ★ SVM Last change: Linear SVM	1.7 ★ SVM Last change: Linear SVM
1.8 ★ SVM Last change: Quadratic SVM	1.8 ★ SVM Stopped 5/5 features
1.9 ★ SVM Last change: Cubic SVM	1.9 ★ SVM Stopped 5/5 features
1.10 ★ SVM Last change: Fine Gaussian SVM	1.10 ★ SVM Last change: Fine Gaussian SVM
1.11 ★ SVM Last change: Medium Gaussian SVM	1.11 ★ SVM Last change: Medium Gaussian SVM
1.12 ★ SVM Last change: Coarse Gaussian SVM	1.12 ★ SVM Last change: Coarse Gaussian SVM
1.13 ★ KNN Last change: Fine KNN	1.13 ★ KNN Last change: Fine KNN
1.14 ★ KNN Last change: Medium KNN	1.14 ★ KNN Last change: Medium KNN
1.15 ★ KNN Last change: Coarse KNN	1.15 ★ KNN Last change: Coarse KNN
1.16 ★ KNN Last change: Cosine KNN	1.16 ★ KNN Last change: Cosine KNN
1.17 ★ KNN Last change: Cubic KNN	1.17 ★ KNN Last change: Cubic KNN
1.18 ★ KNN Last change: Weighted KNN	1.18 ★ KNN Last change: Weighted KNN
1.19 ★ Ensemble Last change: Boosted Trees	1.19 ★ Ensemble Last change: Boosted Trees Accuracy: 80.4% 5/5 features
1.20 ★ Ensemble Last change: Bagged Trees	1.20 ★ Ensemble Last change: Bagged Trees Accuracy: 80.7% 5/5 features
1.21 ★ Ensemble Last change: Subspace Discriminant	1.21 ★ Ensemble Last change: Subspace Discriminant Accuracy: 80.2% 5/5 features
1.22 ★ Ensemble Last change: Subspace KNN	1.22 ★ Ensemble Last change: Subspace KNN Accuracy: 78.8% 5/5 features
1.23 ★ Ensemble Last change: RUSBoosted Trees	1.23 ★ Ensemble Last change: RUSBoosted Trees Accuracy: 69.6% 5/5 features

Bins = 2
PCA = 0
CV = 0
 $\theta = 5$

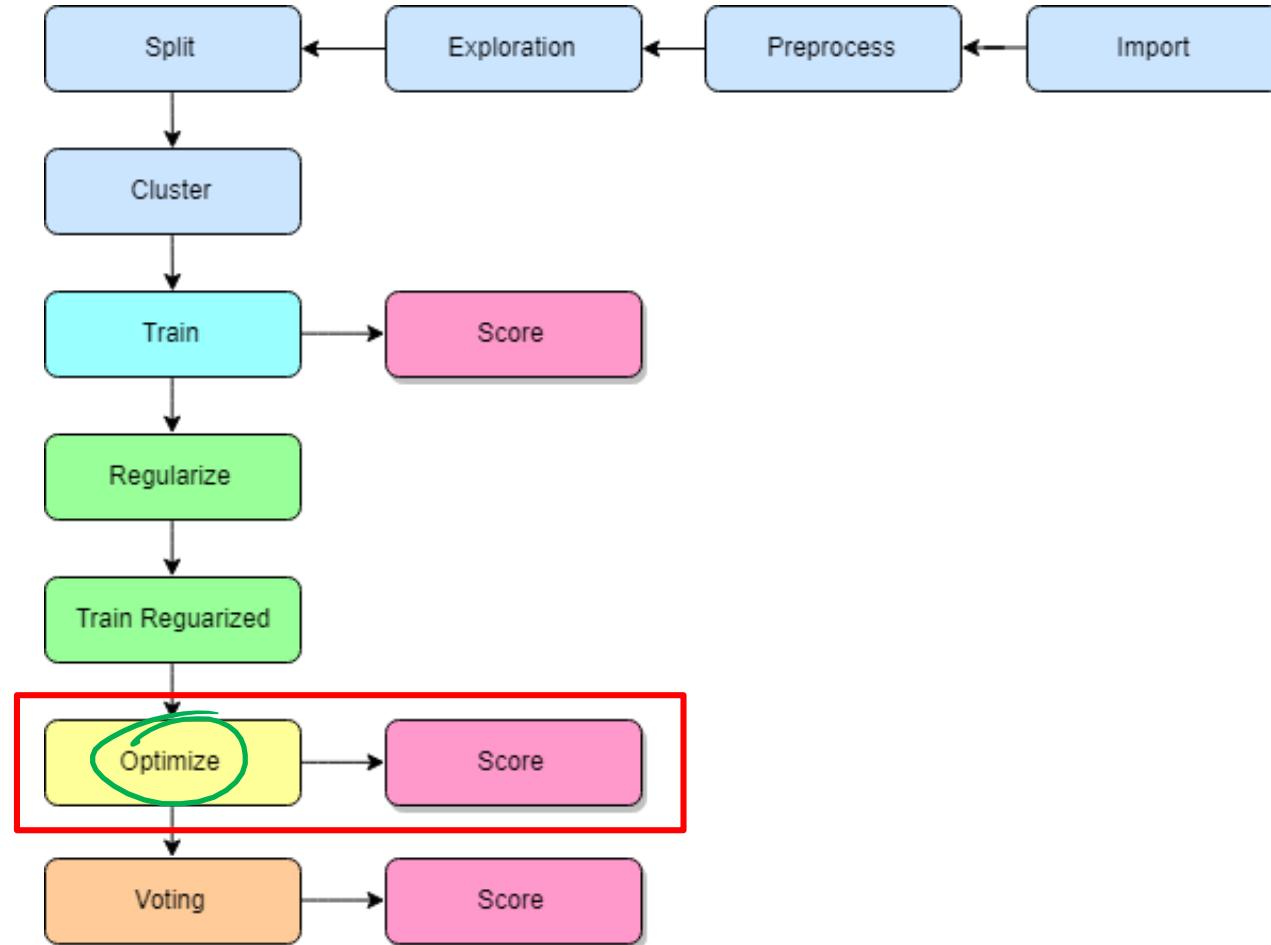
1.1 ★ Tree Accuracy: 78.6% 5/5 features	1.1 ★ Tree Accuracy: 78.6% 5/5 features
1.2 ★ Tree Accuracy: 73.2% 5/5 features	1.2 ★ Tree Accuracy: 73.2% 5/5 features
1.3 ★ Tree Accuracy: 71.3% 5/5 features	1.3 ★ Tree Accuracy: 71.3% 5/5 features
1.4 ★ Linear Discriminant Last change: Linear Discriminant	1.4 ★ Linear Discriminant Last change: Linear Discriminant
1.5 ★ Quadratic Discriminant Last change: Quadratic Discriminant	1.5 ★ Quadratic Discriminant Last change: Quadratic Discriminant
1.6 ★ Logistic Regression Last change: Logistic Regression	1.6 ★ Logistic Regression Last change: Logistic Regression
1.7 ★ SVM Last change: Linear SVM	1.7 ★ SVM Last change: Linear SVM
1.8 ★ SVM Last change: Quadratic SVM	1.8 ★ SVM Stopped 5/5 features
1.9 ★ SVM Last change: Cubic SVM	1.9 ★ SVM Stopped 5/5 features
1.10 ★ SVM Last change: Fine Gaussian SVM	1.10 ★ SVM Last change: Fine Gaussian SVM
1.11 ★ SVM Last change: Medium Gaussian SVM	1.11 ★ SVM Last change: Medium Gaussian SVM
1.12 ★ SVM Last change: Coarse Gaussian SVM	1.12 ★ SVM Last change: Coarse Gaussian SVM
1.13 ★ KNN Last change: Fine KNN	1.13 ★ KNN Last change: Fine KNN
1.14 ★ KNN Last change: Medium KNN	1.14 ★ KNN Last change: Medium KNN
1.15 ★ KNN Last change: Coarse KNN	1.15 ★ KNN Last change: Coarse KNN
1.16 ★ KNN Last change: Cosine KNN	1.16 ★ KNN Last change: Cosine KNN
1.17 ★ KNN Last change: Cubic KNN	1.17 ★ KNN Last change: Cubic KNN
1.18 ★ KNN Last change: Weighted KNN	1.18 ★ KNN Last change: Weighted KNN
1.19 ★ Ensemble Last change: Boosted Trees	1.19 ★ Ensemble Last change: Boosted Trees Accuracy: 100.0% 5/5 features
1.20 ★ Ensemble Last change: Bagged Trees	1.20 ★ Ensemble Last change: Bagged Trees Accuracy: 75.2% 5/5 features
1.21 ★ Ensemble Last change: Subspace Discriminant	1.21 ★ Ensemble Last change: Subspace Discriminant Accuracy: 99.9% 5/5 features
1.22 ★ Ensemble Last change: Subspace KNN	1.22 ★ Ensemble Last change: Subspace KNN Accuracy: 100.0% 5/5 features
1.23 ★ Ensemble Last change: RUSBoosted Trees	1.23 ★ Ensemble Last change: RUSBoosted Trees Accuracy: 72.4% 5/5 features

170 Post-Regularized Models

- 2 bins
 - Ensemble > KNN > SVM
- 4 bins
 - Ensemble > KNN > SVM
- 7 bins
 - Ensemble > KNN > SVM
- Bagged Trees **Ensemble** was most **Accurate**
- Ensemble is many Decision Trees
 - *Decision Trees were most accurate as ensembles, but not individually*
- Going forward, focus is on **2 bins**

170 Post-Regularized Models

- Regularizing the models slightly reduced accuracy



Best Individual Model - Ensemble Bagged Tree

classificationEnsemble0 = fitensemble(...

predictors, ...

response, ...

'Bag', ...

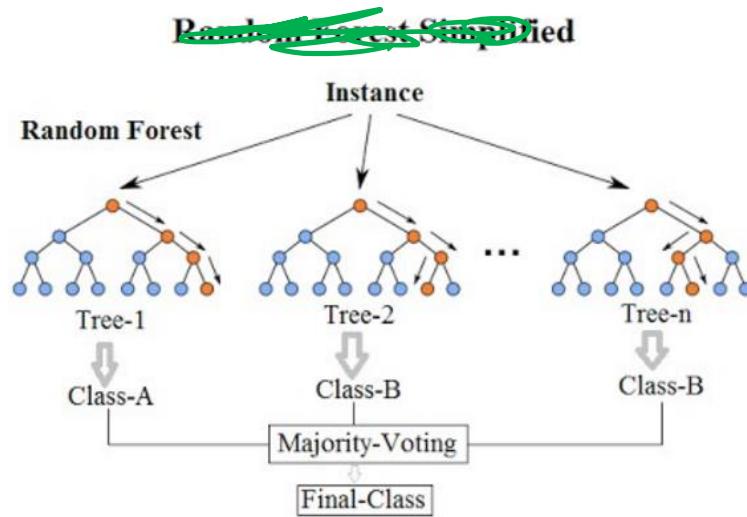
30, ... % # trees

'Tree', ...

'Type', 'Classification', ...

'ClassNames', [1; 2]);

Training time = 15.68 seconds



<https://community.tibco.com/wiki/random-forest-template-tibco-spotfirer-wiki-page>

Model Optimization

- Train Optimized Ensemble Bagged Tree Model
 - Optimize using **hyperparameters**
 - Increase **cross validation**

$$y = x_1 \theta_1 + \dots x_n \theta_n$$

parameters

▼ History

1 ★ Ensemble

Last change: Bagged Trees

Accuracy: 67.1%

5/5 features

trees
splits

pruned

▼ Current model

Model number 1

Status: Trained
Accuracy: 67.1%
Prediction speed: ~7600 obs/sec
Training Time: 15.681 secs

Classifier

Preset: Bagged Trees
Ensemble method: Bag
Learner type: Decision tree
Number of learners: 30

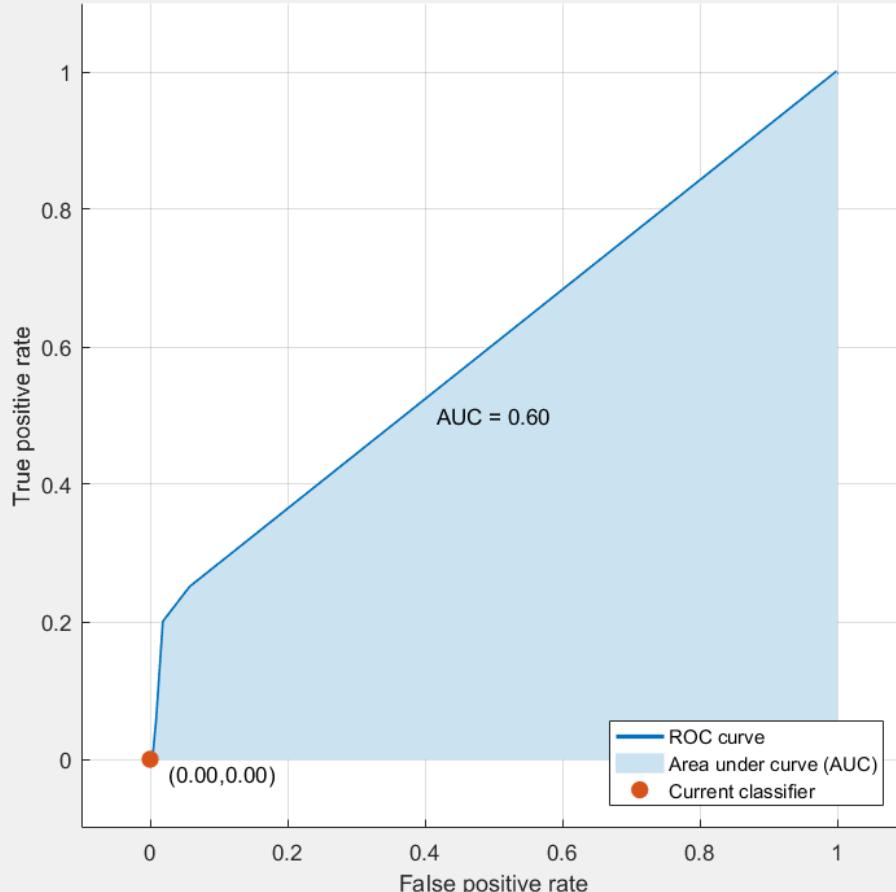
Feature Selection

All features used in the model, before PCA

PCA

PCA disabled

Model 1



1 ★ Ensemble

Last change: 'NumLearners' = '1'

2 ★ Ensemble

Last change: 'NumLearners' = '5'

3 ★ Ensemble

Last change: 'NumLearners' = '10'

4 ★ Ensemble

Last change: 'NumLearners' = '50'

5 ★ Ensemble

Last change: 'NumLearners' = '100'

6 ★ Ensemble

Last change: 'NumLearners' = '500'

Accuracy: 54.8%
5/5 features

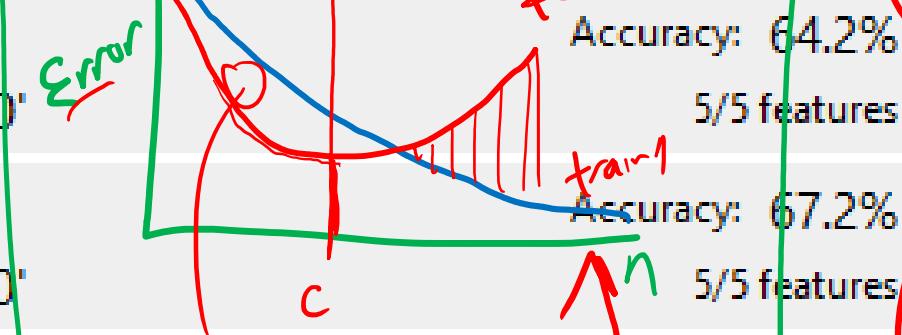
Accuracy: 63.5%
5/5 features

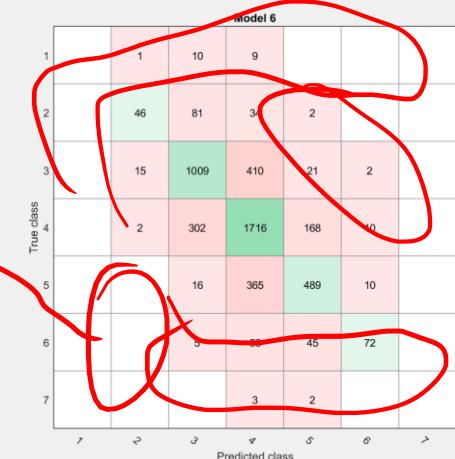
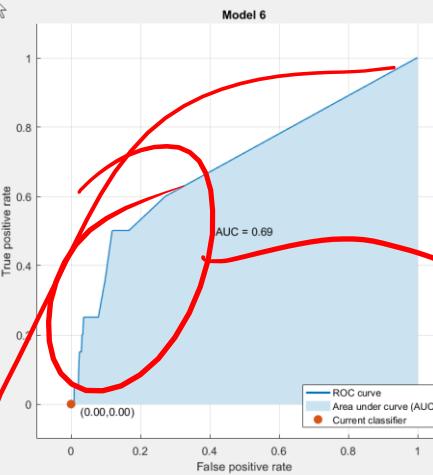
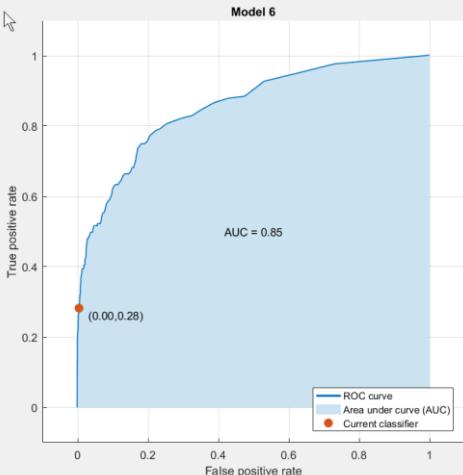
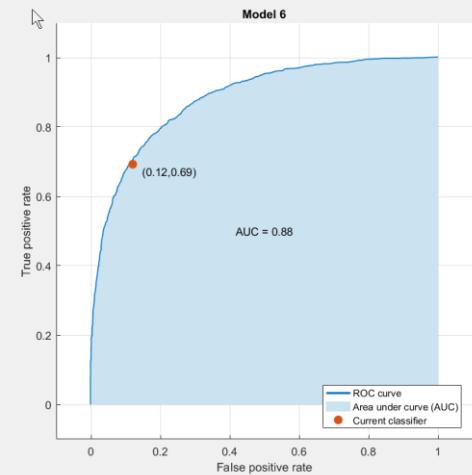
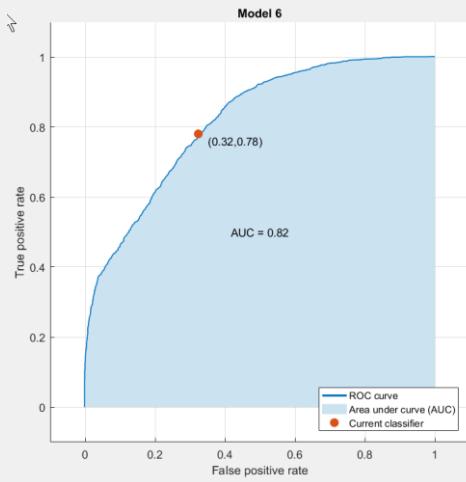
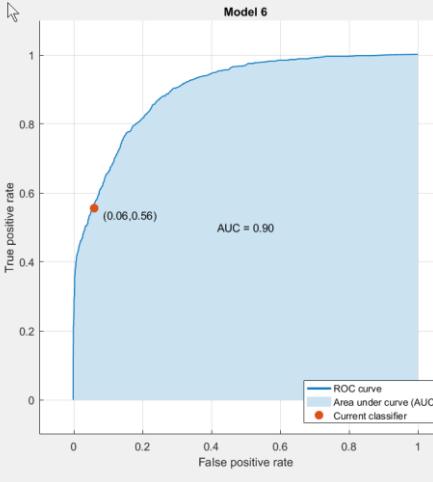
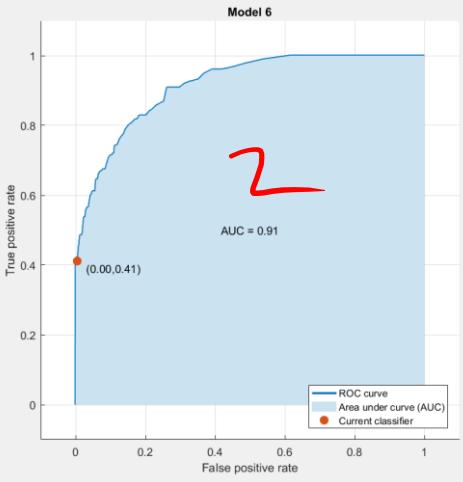
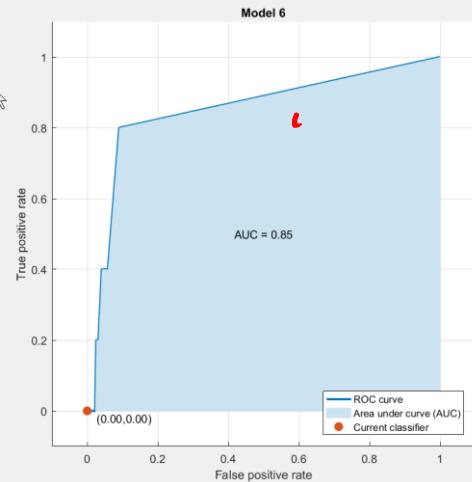
Accuracy: 64.2%
5/5 features

Accuracy: 67.2%
5/5 features

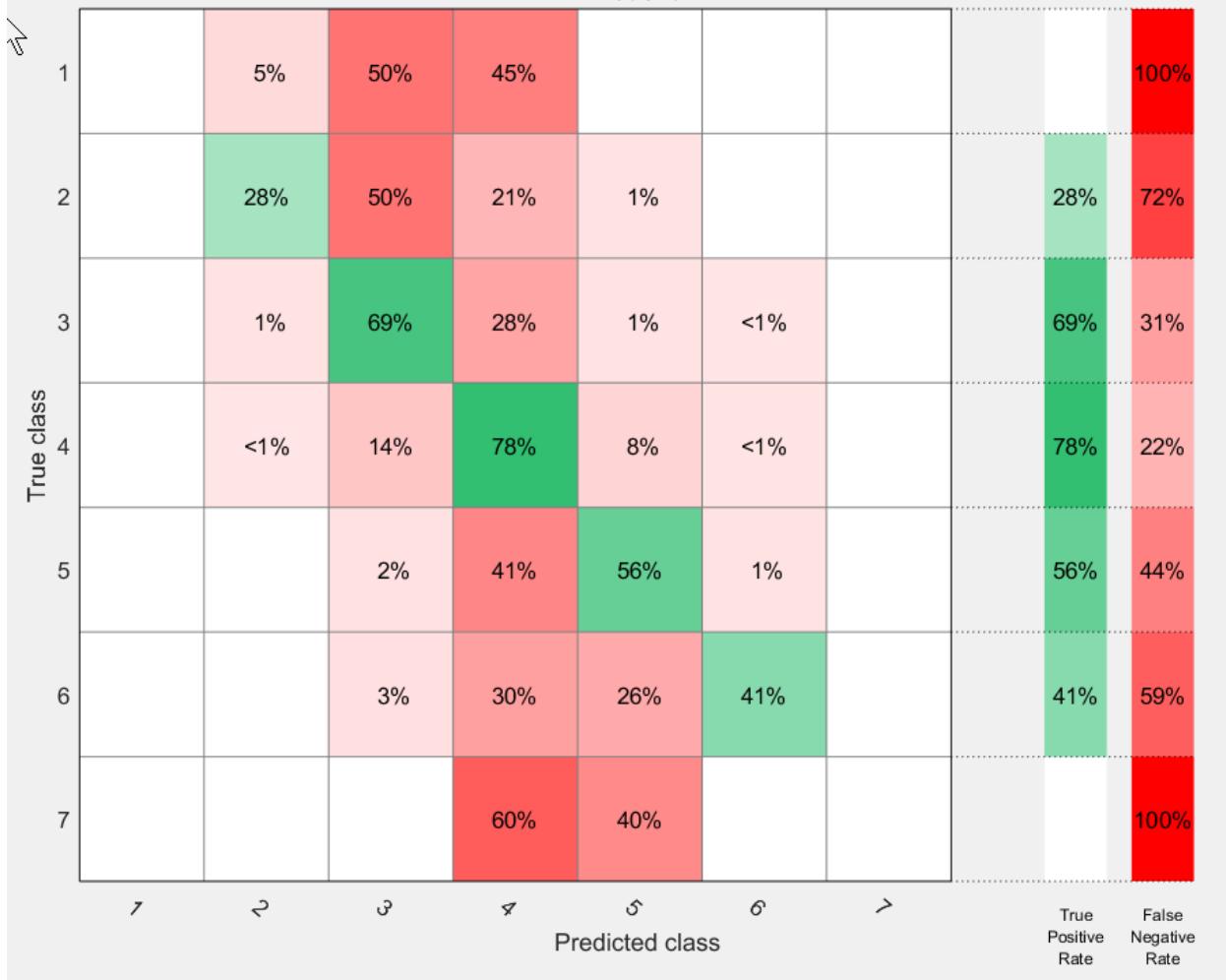
Accuracy: 67.7%
5/5 features

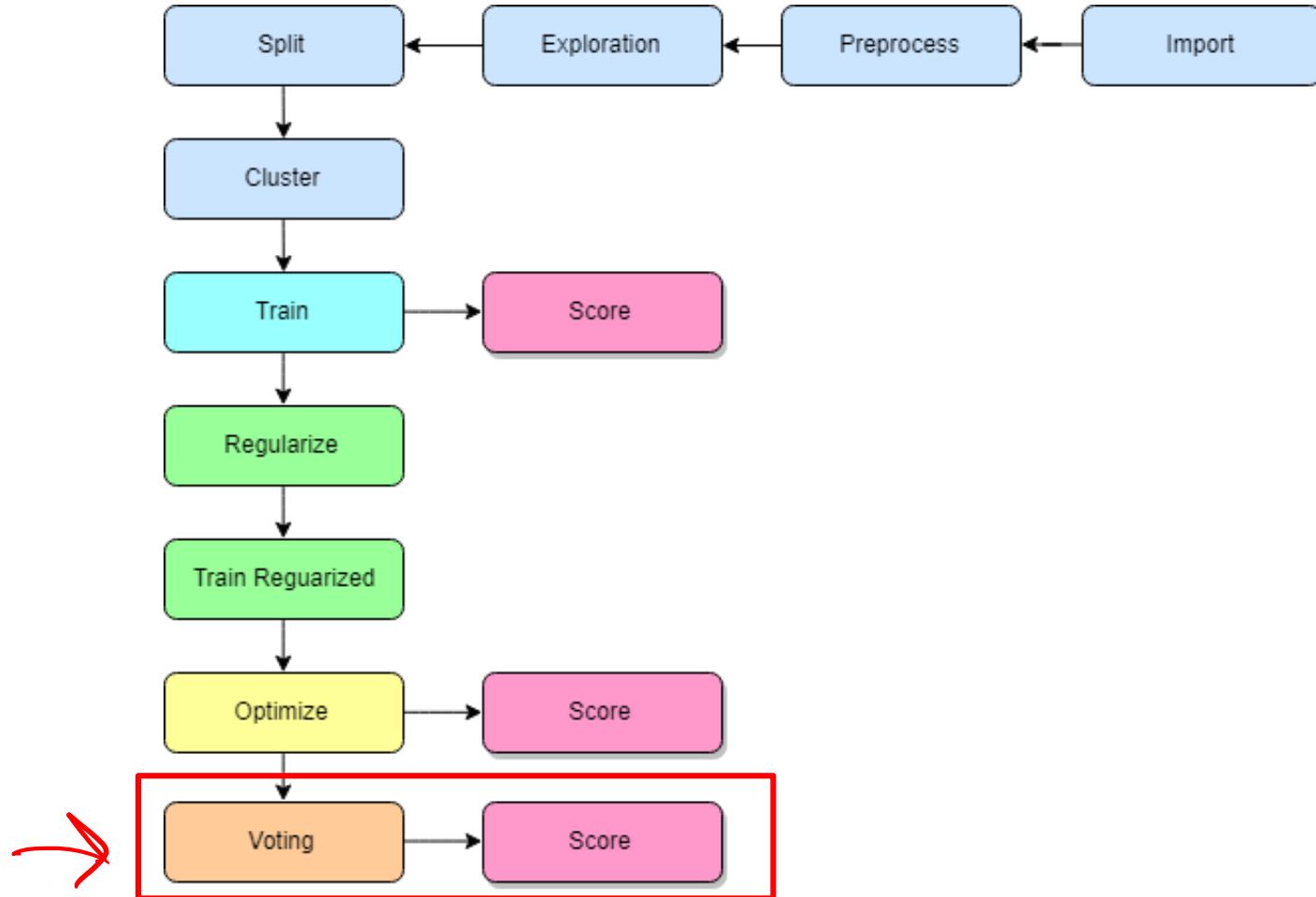
Accuracy: 68.0%
5/5 features





Model 6





Hard and Soft Voting

- Produce class and scores from **most accurate models**
 - **Classes** for hard voting
 - Probability **Scores** for soft voting

- Ensemble Bagged
- Weighted-KNN
- SVM Fine Gaussian
- Complex DT

[ϕ |] $P[\circ, 1]$

[class, score]

Hard Voting

```
class_2_matrix = [...  
    class_2_baggedense...  
    class_2_ComplexDT...  
    class_2_fineGaussianSVM ...  
    class_2_weightedKNN];
```

```
votesHard_class_2 = mode(class_2_matrix, 2);
```

1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	0	0	1	0
6	0	0	0	0
7	1	0	1	1
8	0	0	0	0
9	1	1	1	1
10	1	0	1	1
11	0	0	0	0
12	1	1	1	1
13	1	1	1	1
14	1	1	1	1
15	1	1	1	1
16	0	0	1	0
17	1	1	1	1
18	0	0	1	0
19	1	1	1	1

Soft Voting

Follow whichever model is *loudest*

threemodels = [...

[score_2_ComplexDT...
 score_2_baggedensemble.
 score_2_weightedKNN];

[M, I] = **max**(threemodels, [], 2);

	1 <i>M₁</i>	2 <i>M₂</i>	3 <i>M₃</i>	4	5	6	
1	0.1365	0.8635	0.0667	0.9333	0	0	1
2	-0.1408	+0.8592	0.0667	0.9333	0	0	1
3	0.1408	0.8592	0.1667	0.8333	0.1676	0.8324	
4	0.1408	0.8592	0.0333	0.9667	0	0	1
5	0.7419	0.2581	0.6667	0.3333	0.7549	0.2451	
6	0.7174	0.2826	0.7333	0.2667	0.9429	0.0571	
7	0.7500	0.2500	0.3000	0.7000	0.4284	0.5716	
8	0.8065	0.1936	0.7000	0.3000	0.7546	0.2454	
9	0.1365	0.8635	0	1	0	0	1
10	0.5940	0.4060	0.1667	0.8333	0	0	1
11	0.7500	0.2500	0.8333	0.1667	0.6349	0.3651	
12	0.1364	0.8636	0.1333	0.8667	0.0643	0.9357	
13	0.2626	0.7374	0.1667	0.8333	0.1961	0.8039	
14	0.1408	0.8592	0.2000	0.8000	0.1292	0.8708	
15	0.0417	0.9583	0.1667	0.8333	0.0507	0.9493	
16	1	0	0.7333	0.2667	0.9087	0.0913	
17	0.1408	0.8592	0.1333	0.8667	0.2486	0.7514	
18	0.7419	0.2581	0.9000	0.1000	0.8621	0.1379	
19	0.1408	0.8592	0.3333	0.6667	0.1545	0.8456	

Soft Voting

% 0 predictors

$\sum(I == 1 \mid I == 3 \mid I == 5) = 8;$

% 1 predictors

$\sum(I == 2 \mid I == 4 \mid I == 6) = 9;$

$\sum(I == 8) = 0;$

$\sum(I == 9) = 1;$

`votesSoft_class_2 = I;`

	1	2	3	4	5	6
1	0.1365	0.8635	0.0667	0.9333	0	0.1
2	0.1408	0.8592	0.0667	0.9333	0	0.1
3	0.1408	0.8592	0.1667	0.8333	0.1676	0.8324
4	0.1408	0.8592	0.0333	0.9667	0	1
5	0.7419	0.2581	0.6667	0.3333	0.7549	0.2451
6	0.7174	0.2826	0.7333	0.2667	0.9429	0.0571
7	0.7500	0.2500	0.3000	0.7000	0.4284	0.5716
8	0.8065	0.1936	0.7000	0.3000	0.7546	0.2454
9	0.1365	0.8635	0	1	0	1
10	0.5940	0.4060	0.1667	0.8333	0	1
11	0.7500	0.2500	0.8333	0.1667	0.6349	0.3651
12	0.1364	0.8636	0.1333	0.8667	0.0643	0.9357
13	0.2626	0.7374	0.1667	0.8333	0.1961	0.8039
14	0.1408	0.8592	0.2000	0.8000	0.1292	0.8708
15	0.0417	0.9583	0.1667	0.8333	0.0507	0.9493
16	1	0	0.7333	0.2667	0.9087	0.0913
17	0.1408	0.8592	0.1333	0.8667	0.2486	0.7514
18	0.7419	0.2581	0.9000	0.1000	0.8621	0.1379
19	0.1408	0.8592	0.3333	0.6667	0.1545	0.8456

hard	soft
1	1
1	1
1	1
1	1
0	0
0	0
1	0
0	0
1	1
1	1
0	0
1	1
1	1
1	1
1	1
0	0
1	1
1	1
1	1
0	0
1	1
0	0
1	1
1	1
1	1
0	0
1	1
1	1
1	1
1	1
0	0
1	1
1	1
1	1
1	1
0	0
1	1
1	1
1	1
1	1

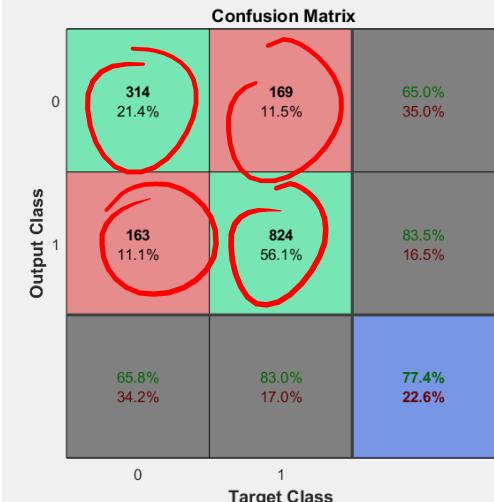
Hard vs Soft Comparison

- 61 different
- **4.1% different**
- Soft voting predicts more class 1 (Good Wine)

Score

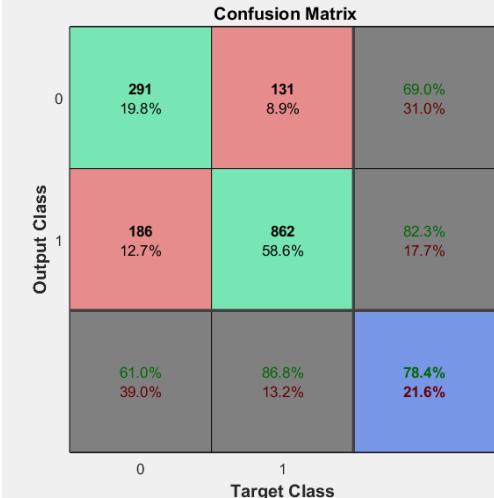
- Hard Voting - Good Wine Prediction

- Accuracy = 0.774
- Precision = 0.658
- Recall = 0.650
- FScore = 0.654



- Soft Voting - Good Wine Prediction

- Accuracy = 0.784
- Precision = 0.610
- Recall = 0.690
- FScore = 0.647



X

Model of Models

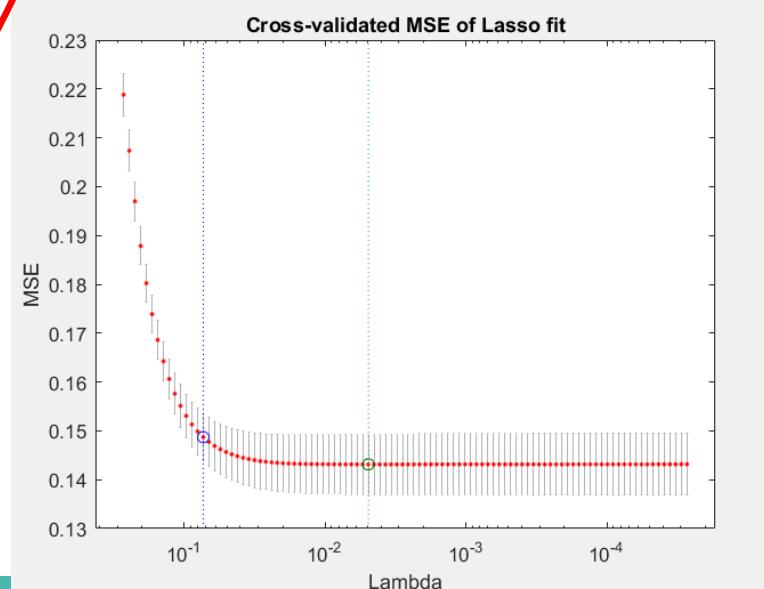
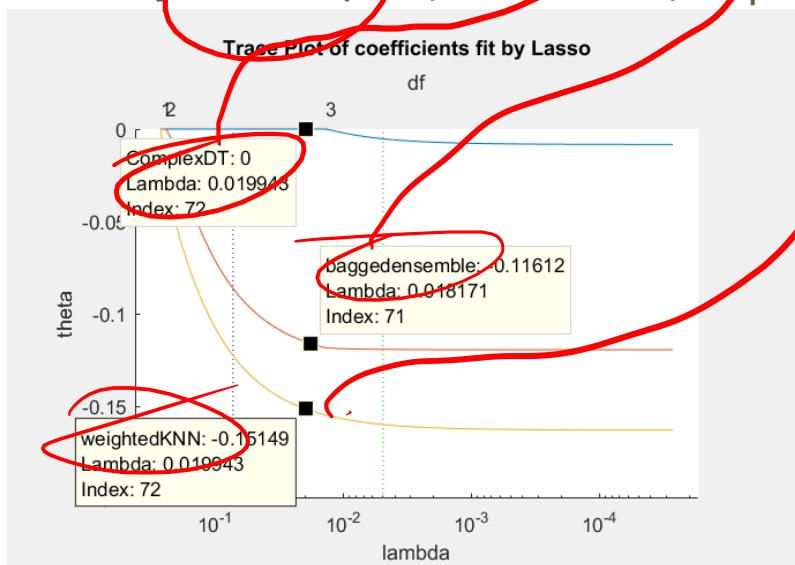
$$y = \underset{\text{weights}}{x_1 \theta_1} + \underset{\text{weights}}{x_2 \theta_2} + \underset{\text{weights}}{x_3 \theta_3}$$

score

```
kfold = 10;
```

```
alpha = 1;
```

```
[B FitInfo] = lasso(X, Y, 'CV', kfold, 'Alpha', alpha);
```



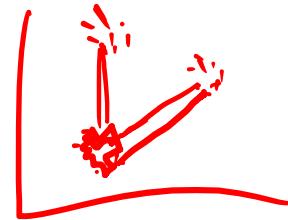
Regrets

- Built Model of Models with **Test** data
- Standardized data **before** train-test split
- Train-Test Split **rearranged** categories
 - Only three class 9 observations in full dataset. Test didn't have a 9. Classes later indexed to 0, causing misalignment (suspicion).



Wish List

- No **Minimum Spending Tree** for clustering
- Produce a **training-testing error** graph
- Compare models with more than just **accuracy**
- Adjust more hyperparameters dials
 - Via **parameter sweep**
- **Regularize** Bagged Tree Ensemble
- Include **more models** in voting → **odd**
- Explore **Red Wine** dataset



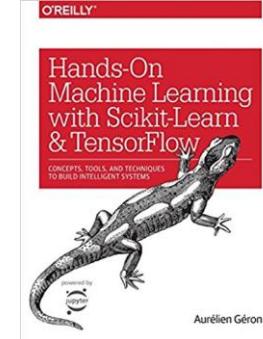
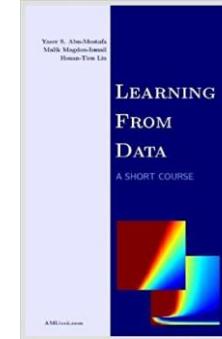
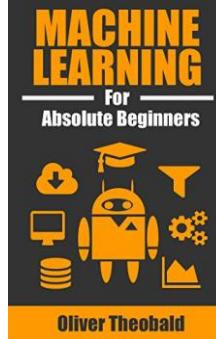
ansatz #

5
10
30
100
500 500



Sources/references

Professor Chih Lai



Data source: <http://archive.ics.uci.edu/ml/datasets/Wine+Quality>

Book 1 - <https://www.amazon.com/Machine-Learning-Absolute-Beginners-Introduction-ebook/dp/B06VXKBLNG>

Book 2 - <https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1491962291>

Book 3 - https://www.amazon.com/gp/product/1600490069/ref=oh_aui_detailpage_o09_s01?ie=UTF8&psc=1

Diagram - <https://docs.microsoft.com/en-us/azure/machine-learning/studio/studio-overview-diagram>

One hot - <https://docs.scipy.org/doc/numpy/reference/generated/numpy.digitize.html>

Classification Learner - <https://www.mathworks.com/help/stats/train-classification-models-in-classification-learner-app.html>

Ensemble - <https://stats.stackexchange.com/questions/349540/hard-voting-soft-voting-in-ensemble-based-methods>

Everything Else - <https://stackoverflow.com/>

Thank you!

