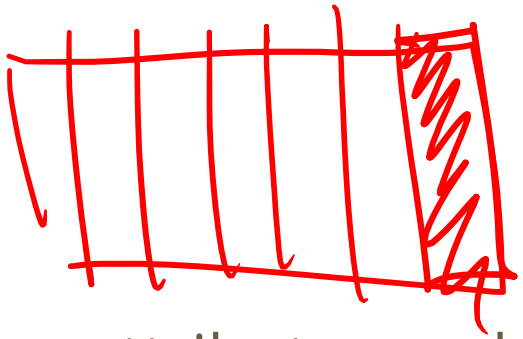
Wine Quality Prediction

SEIS 763: Machine Learning
Presenter: Garth Mortensen

Purpose



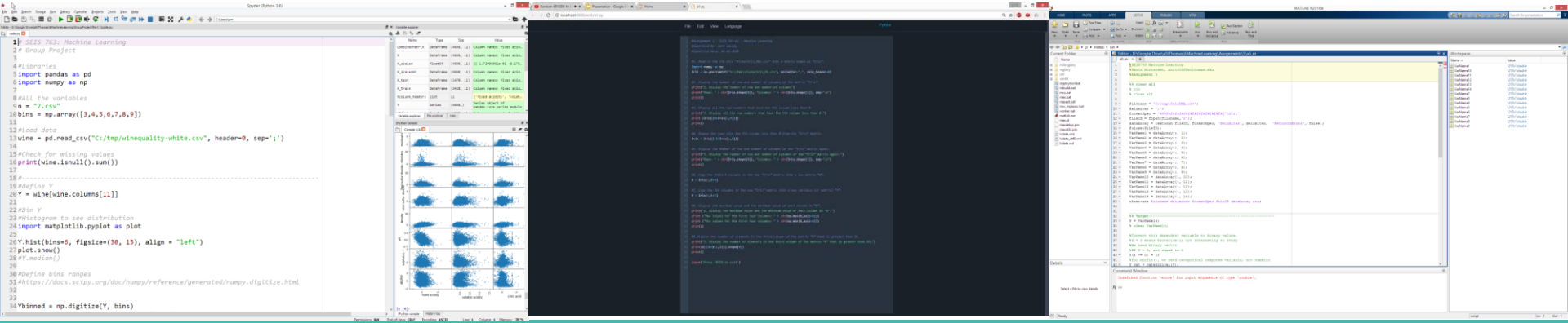
Given various attributes such as acidity, chlorides, pH and alcohol, identify which category (Good/Bad) a new wine bottle belongs to, using classification.

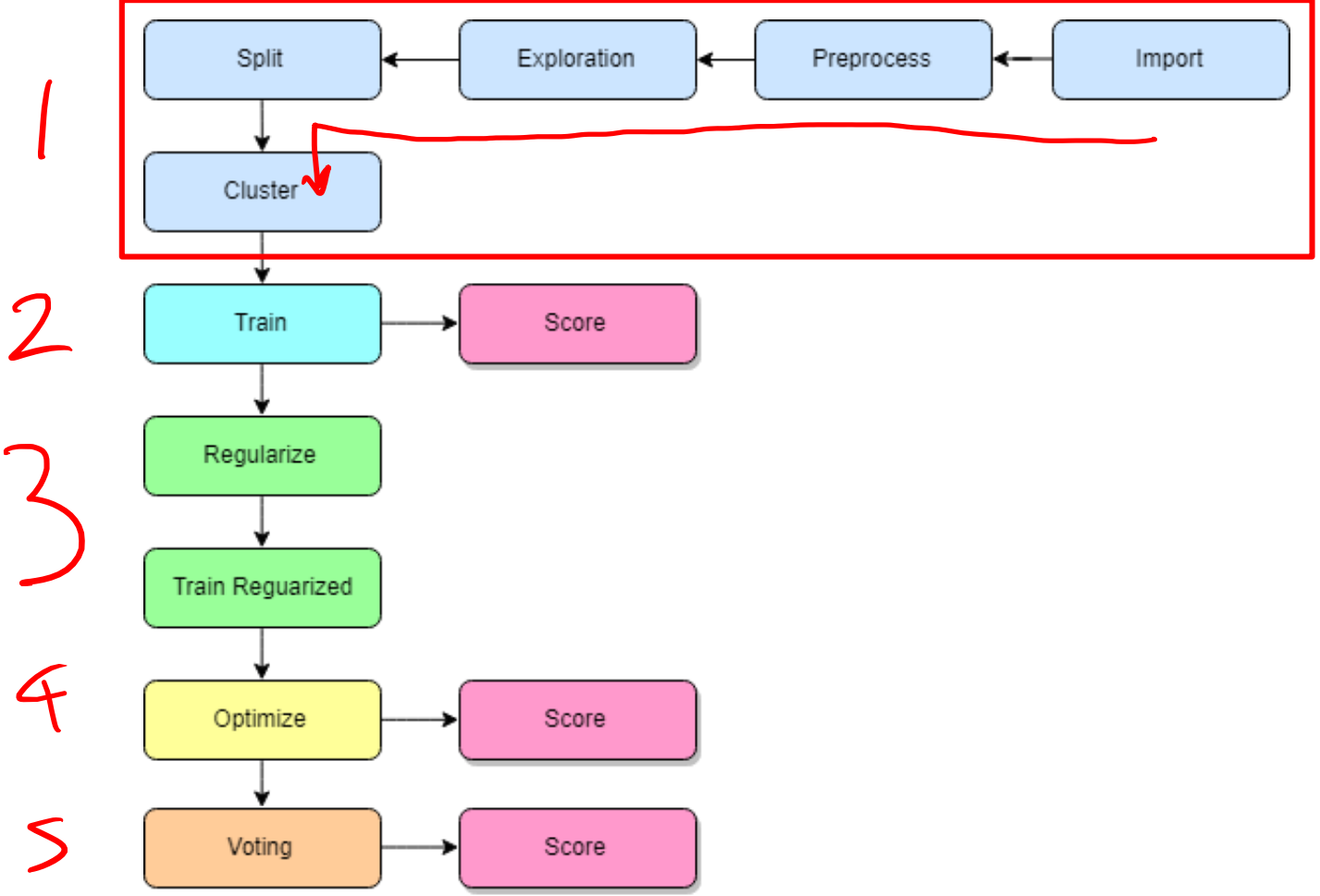
Find the best model, then optimize it.

Software/Tools Used

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

ΣLOC = 900+





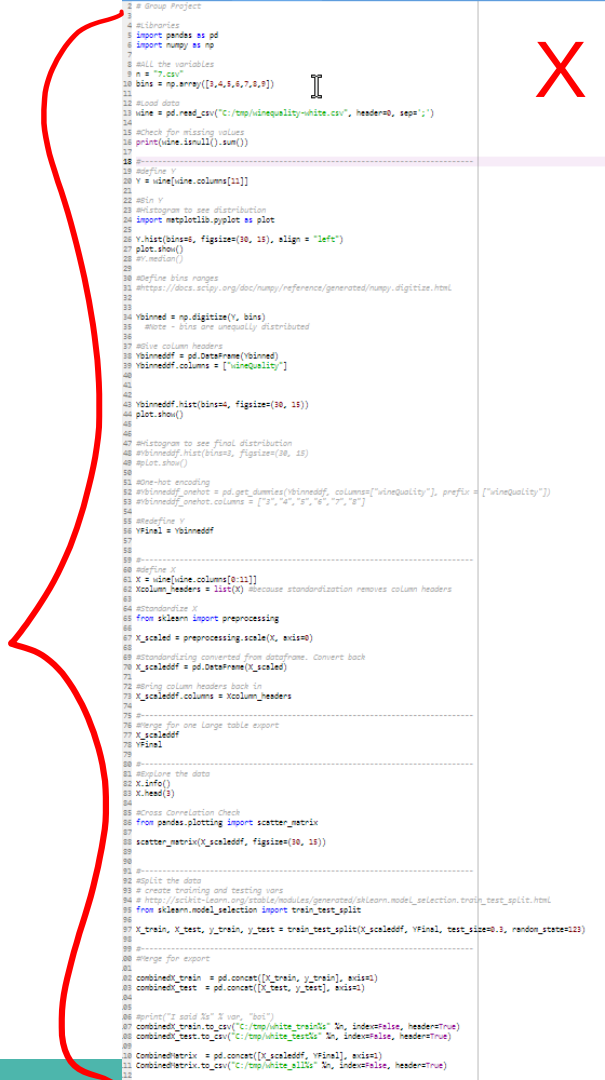
Adapted from: https://docs.microsoft.com/en-us/azure/machine-learning/studio/studio-overview-diagram

Data Set & Cleaning

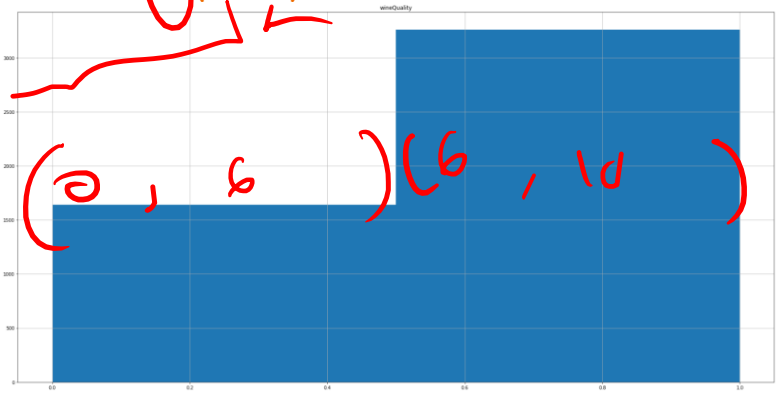
- Data **skewed** between Red and White wine
  - Red = 1,599 x 12
  - White = 4,898 x 12
- No NULL entries

- Predictors
  - All are **numerical**
  - All have been **standardized**

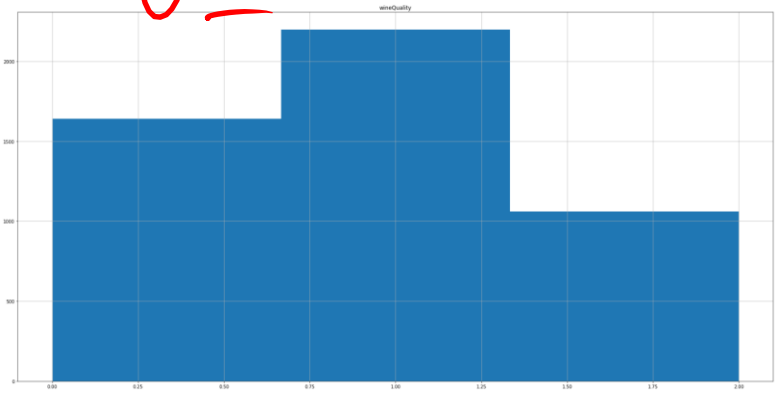
- **Target**
  - **Categorical (3, 9)**



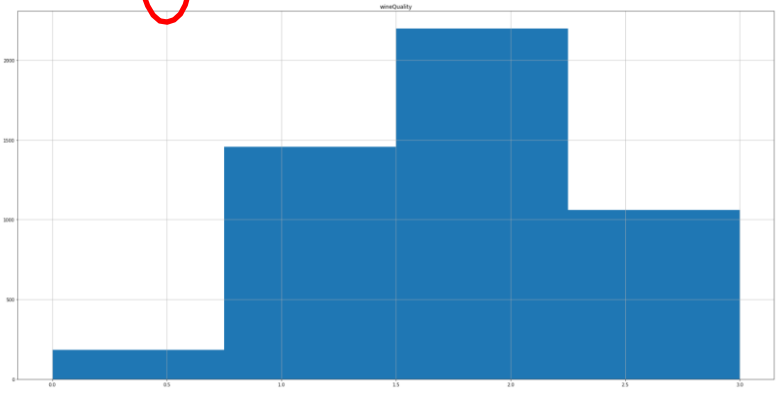
Y bins = 2 (6, 10)



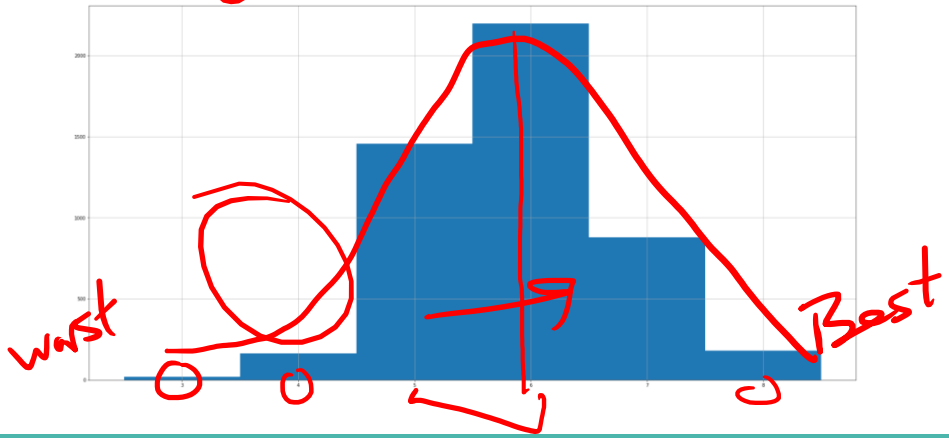
Y bins = 3 (6, 7, 10)



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



Y bins = 7 (default)



Correlation check

%numeric correlation

[correlation, pval] = corr([alcohol, chlorides, citricacid, density, fixedacidity, freesulfurdioxide, pH, residualsugar, sulphates, totalsulfurdioxide, volatileacidity]);

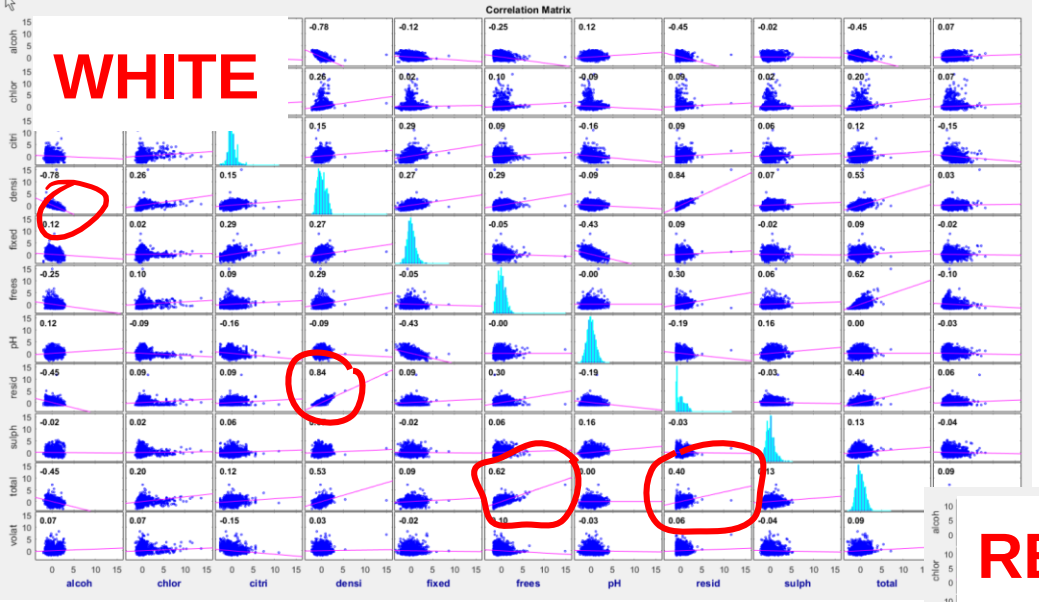
Max(abs(corr))

White 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
-0.360	1.000	0.114	0.257	0.023	0.101	-0.090	0.089	0.017	0.199	0.071
-0.076	0.114	1.000	0.150	0.289	0.094	-0.164	0.094	0.062	0.121	-0.149
-0.780	0.257	0.150	1.000	0.265	0.294	-0.094	0.839	0.074	0.530	0.027
-0.121	0.023	0.289	0.265	1.000	-0.049	-0.426	0.089	-0.017	0.091	-0.023
-0.250	0.101	0.094	0.294	-0.049	1.000	-0.001	0.299	0.059	0.616	-0.097
0.121	-0.090	-0.164	-0.094	-0.426	-0.001	1.000	-0.194	0.156	0.002	-0.032
-0.451	0.089	0.094	**0.839**	0.089	0.299	-0.194	1.000	-0.027	0.401	0.064
-0.017	0.017	0.062	0.074	-0.017	0.059	0.156	-0.027	1.000	0.135	-0.036
-0.449	0.199	0.121	0.530	0.091	**0.616**	0.002	**0.401**	0.135	1.000	0.089
0.068	0.071	-0.149	0.027	-0.023	-0.097	-0.032	0.064	-0.036	0.089	1.000

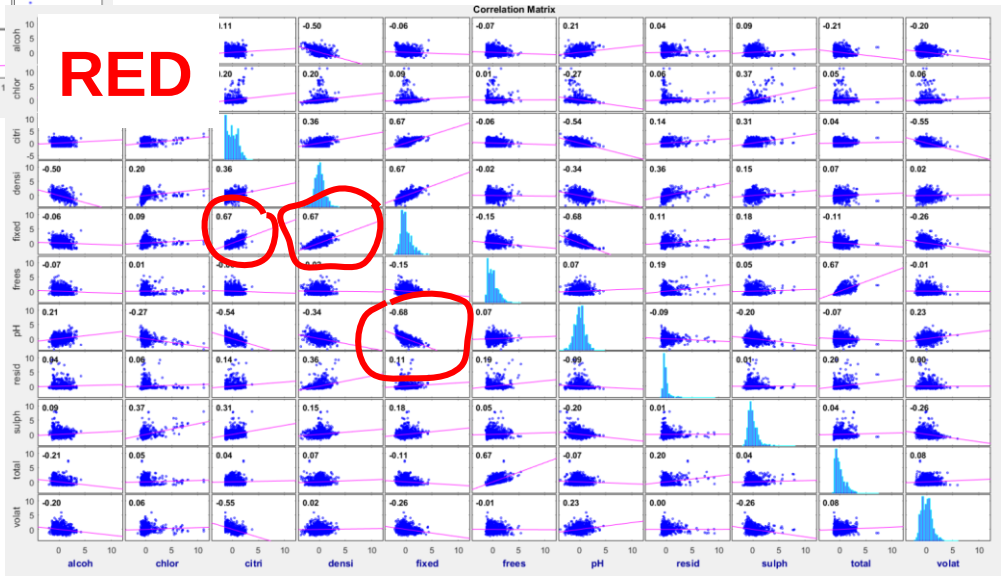
Red Wine Correlation Matrix

1.000	-0.221	0.110	-0.496	-0.062	-0.069	0.206	0.042	0.094	-0.206	-0.202
-0.221	1.000	0.204	0.201	0.094	0.006	-0.265	0.056	0.371	0.047	0.061
0.110	0.204	1.000	0.365	0.672	-0.061	-0.542	0.144	0.313	0.036	-0.552
-0.496	0.201	0.365	1.000	0.668	-0.022	-0.342	0.355	0.149	0.071	0.022
-0.062	0.094	**0.672**	**0.668**	1.000	-0.154	-0.683	0.115	0.183	-0.113	-0.256
-0.069	0.006	-0.061	-0.022	-0.154	1.000	0.070	0.187	0.052	0.668	-0.011
0.206	-0.265	-0.542	-0.342	-0.683	0.070	1.000	-0.086	-0.197	-0.066	0.235
0.042	0.056	0.144	0.355	0.115	0.187	-0.086	1.000	0.006	0.203	0.002
0.094	0.371	0.313	0.149	0.183	0.052	-0.197	0.006	1.000	0.043	-0.261
-0.206	0.047	0.036	0.071	-0.113	**0.668**	0.066	0.203	0.043	1.000	0.076
-0.202	0.061	-0.552	0.022	-0.256	-0.011	0.235	0.002	-0.261	0.076	1.000



sugar ↑
density ↑

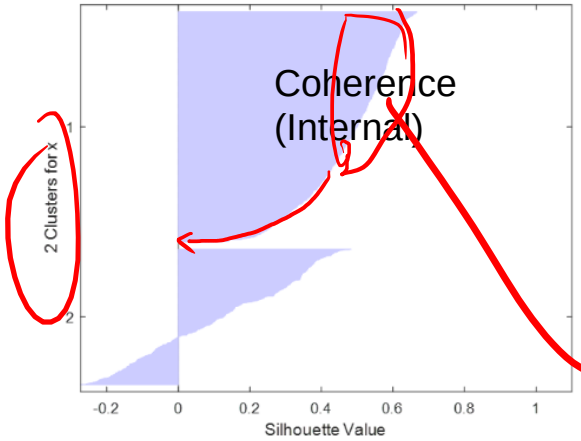
alcohol ↑
density ↓



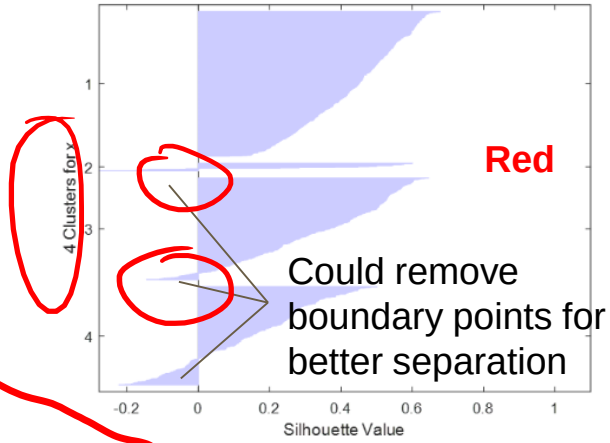
Randomized 70-30 Split

- 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

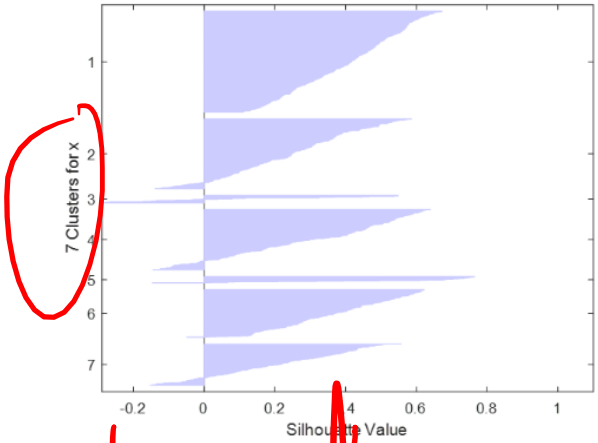


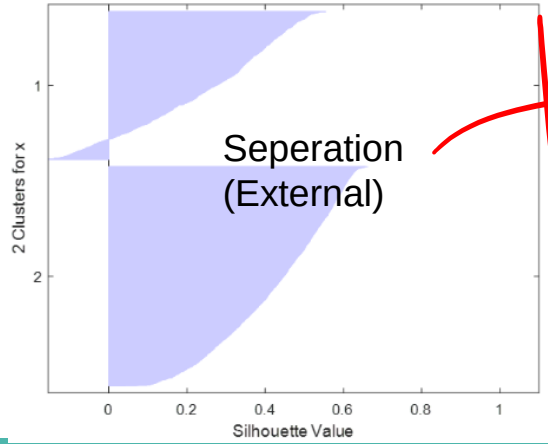
Coherence
(Internal)



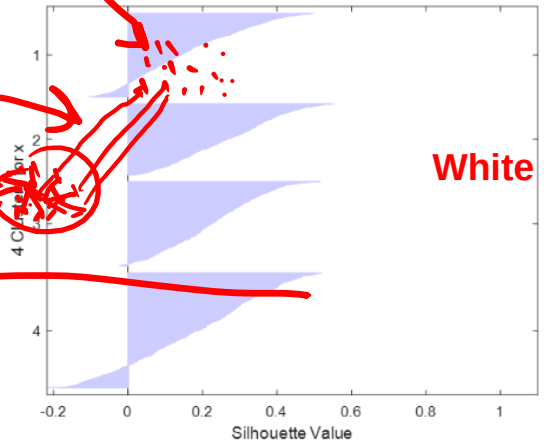
Red

Could remove
boundary points for
better separation

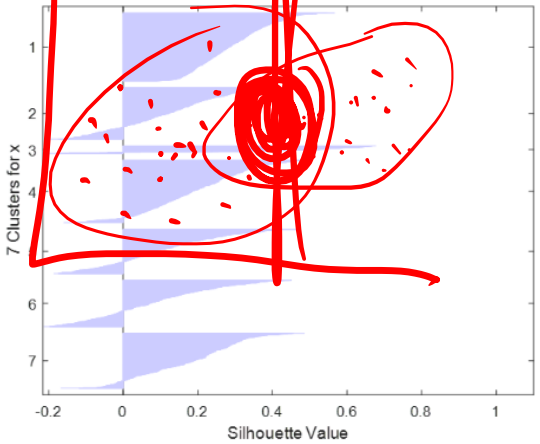


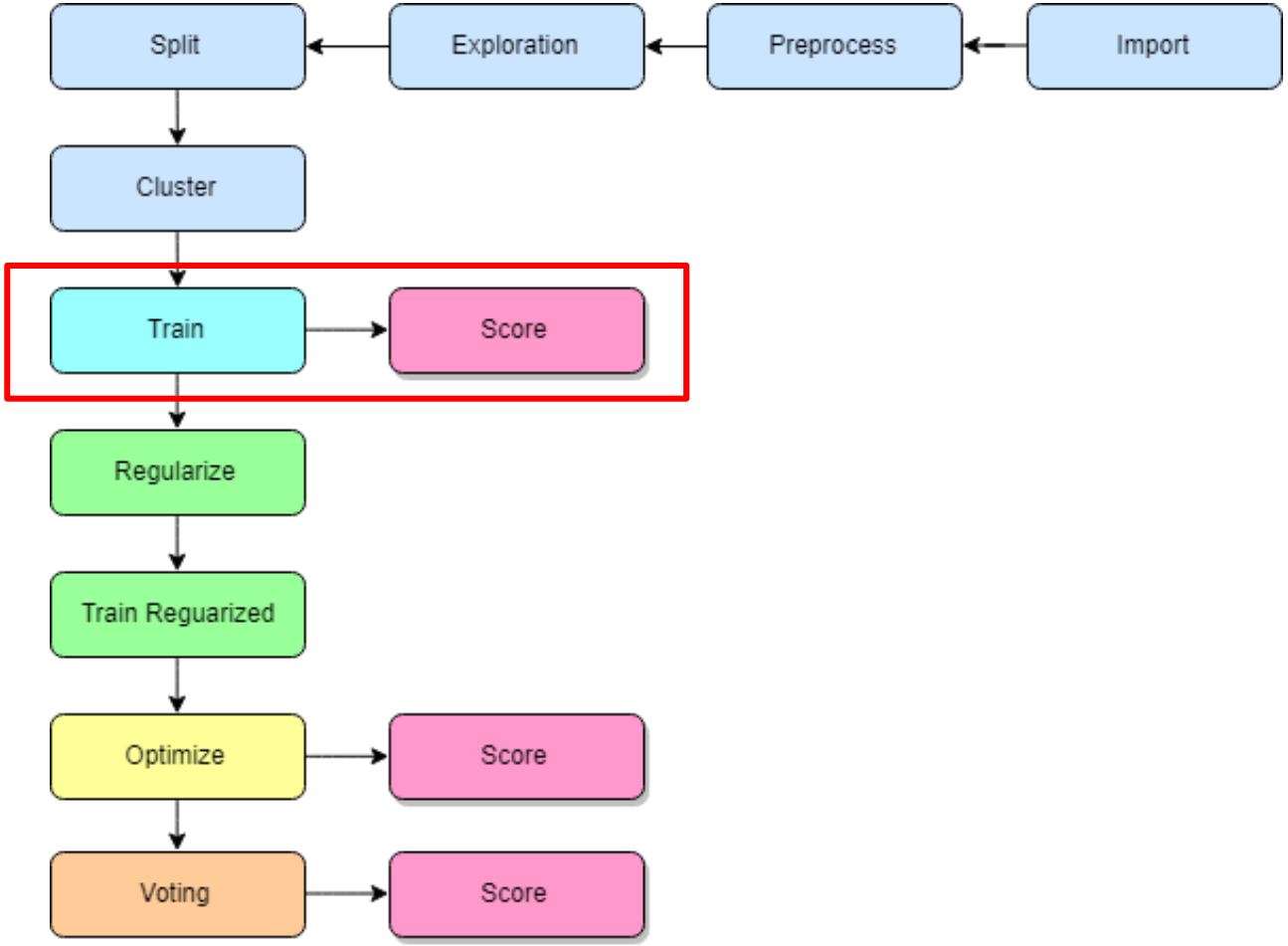


Seperation
(External)

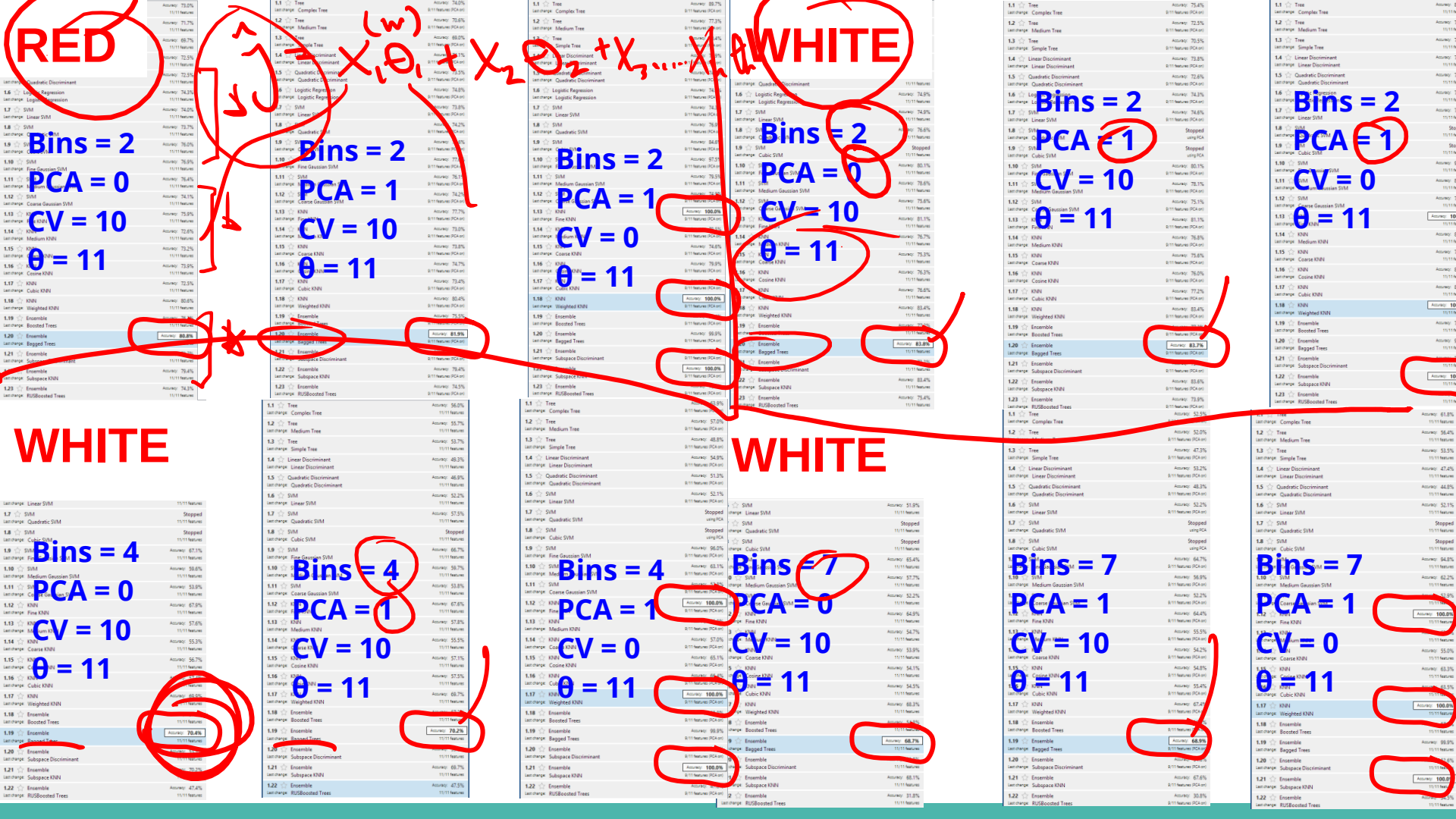


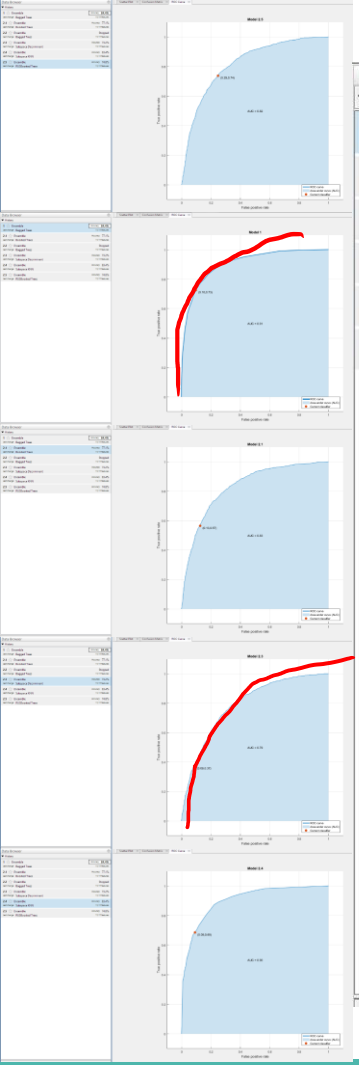
White





Adapted from: https://docs.microsoft.com/en-us/azure/machine-learning/studio/studio-overview-diagram


RED
WHITE
WHITE
WHITE
Bins = 2
PCA = 0
CV = 10
θ = 11
Bins = 2
PCA = 1
CV = 10
θ = 11
Bins = 2
PCA = 1
CV = 0
θ = 11
Bins = 2
PCA = 0
CV = 10
θ = 11
Bins = 2
PCA = 1
CV = 10
θ = 11
Bins = 2
PCA = 1
CV = 0
θ = 11
Bins = 4
PCA = 0
CV = 10
θ = 11
Bins = 4
PCA = 1
CV = 10
θ = 11
Bins = 4
PCA = 1
CV = 0
θ = 11
Bins = 7
PCA = 0
CV = 10
θ = 11
Bins = 7
PCA = 1
CV = 10
θ = 11
Bins = 7
PCA = 1
CV = 0
θ = 11



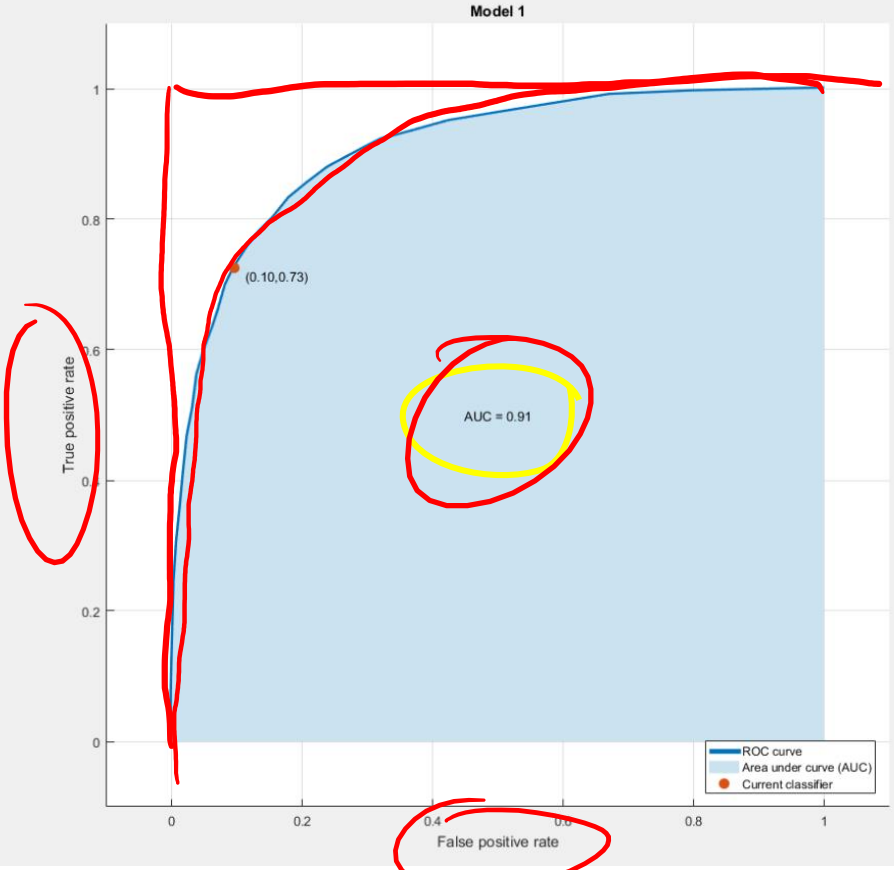
Data Browser

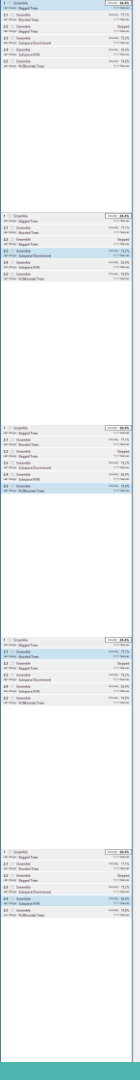
History

1	Ensemble	Accuracy: **84.4%**	
Last change:	Bagged Trees	11/11 features	
2.1	Ensemble	Accuracy: 77.1%	
Last change:	Boosted Trees	11/11 features	
2.2	Ensemble	Stopped	
Last change:	Bagged Trees	11/11 features	
2.3	Ensemble	Accuracy: 73.2%	
Last change:	Subspace Discriminant	11/11 features	
2.4	Ensemble	Accuracy: 83.4%	
Last change:	Subspace KNN	11/11 features	
2.5	Ensemble	Accuracy: 74.8%	
Last change:	RUSBoosted Trees	11/11 features	

Bins = 2
PCA = 0
CV = 10
θ = 11

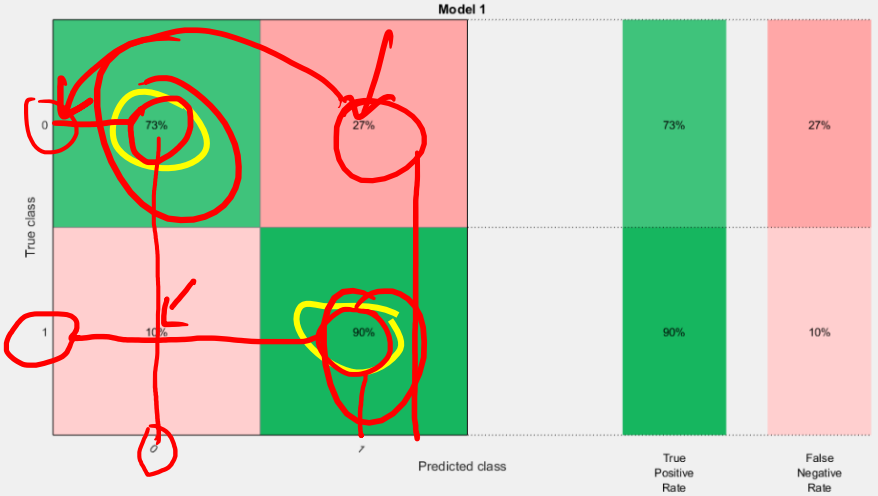
Scatter Plot | Confusion Matrix | ROC Curve





1	Ensemble	Accuracy:	84.4%
Last change:	Bagged Trees		11/11 features
2.1	Ensemble	Accuracy:	77.1%
Last change:	Boosted Trees		11/11 features
2.2	Ensemble		Stopped
Last change:	Bagged Trees		11/11 features
2.3	Ensemble	Accuracy:	73.2%
Last change:	Subspace Discriminant		11/11 features
2.4	Ensemble	Accuracy:	83.4%
Last change:	Subspace KNN		11/11 features
2.5	Ensemble	Accuracy:	74.8%
Last change:	RUSBoosted Trees		11/11 features

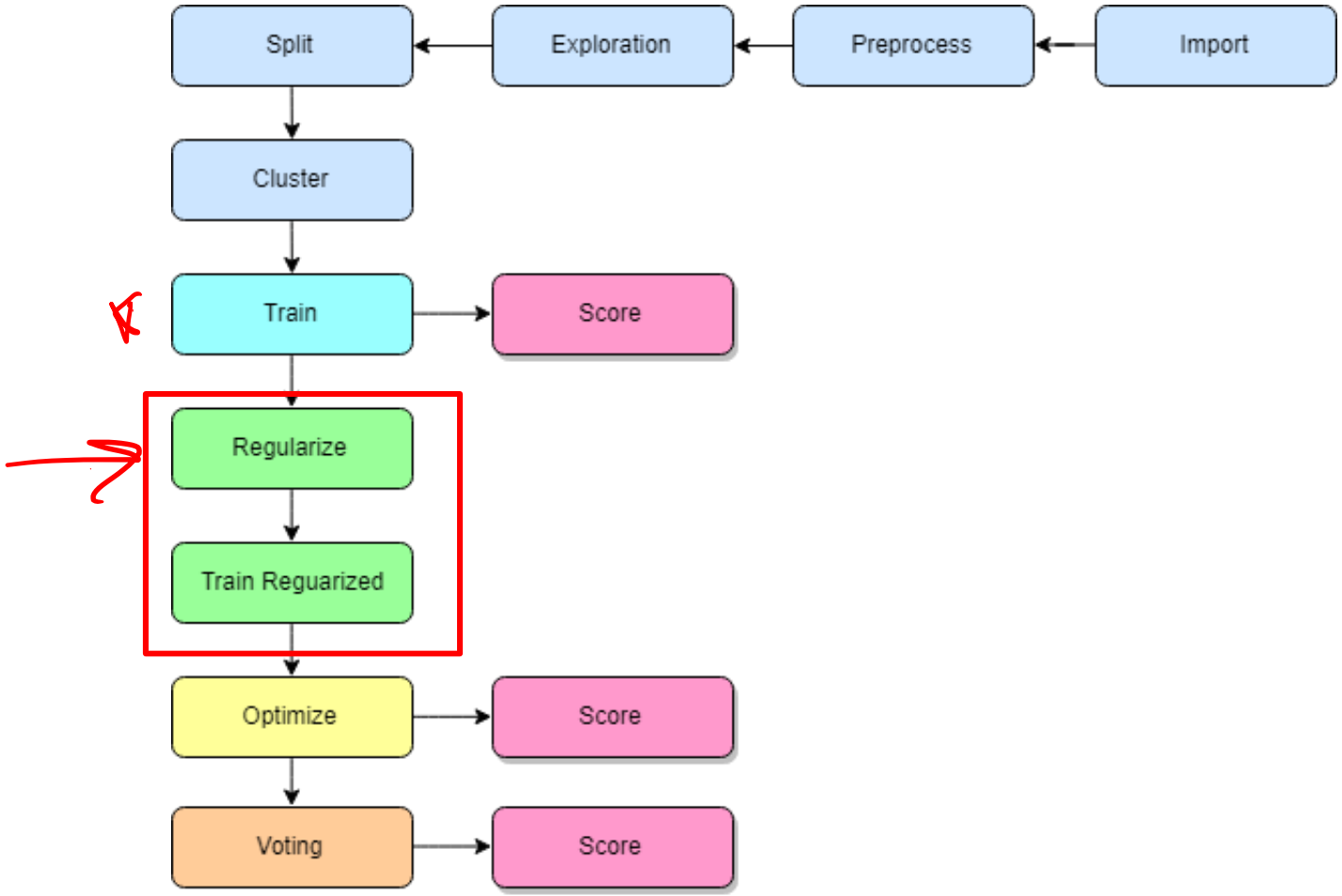
Bins = 2
PCA = 0
CV = 10
θ = 11



270 Pre-Regularized Models

- 2 bins
  - Ensemble > KNN > SVM
- 4 bins
  - Ensemble > KNN > SVM
- 7 bins
  - Ensemble > KNN > SVM

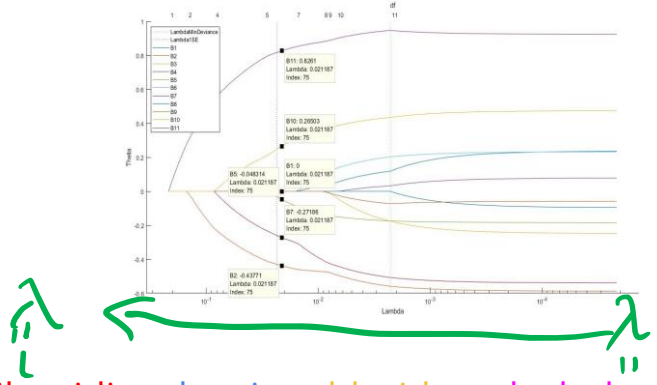
- *Bagged Tree **Ensemble** was most **Accurate***



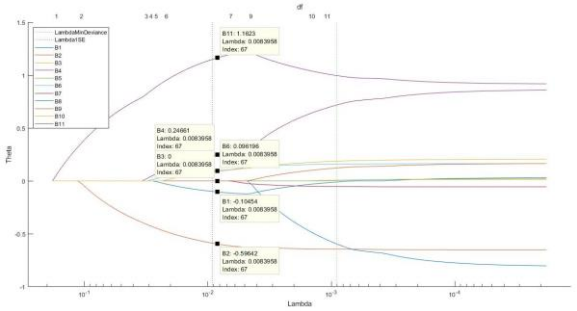
Adapted from: https://docs.microsoft.com/en-us/azure/machine-learning/studio/studio-overview-diagram

Lasso Regularization

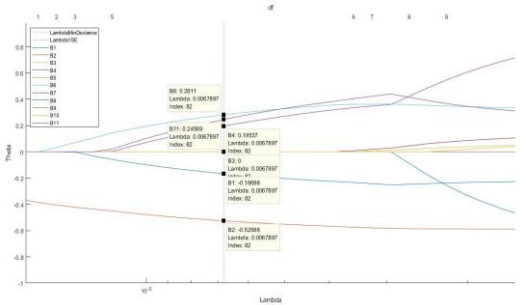
2 bin: volatileacidity, chlorides, totalsulfurdioxide, sulphates, alcohol



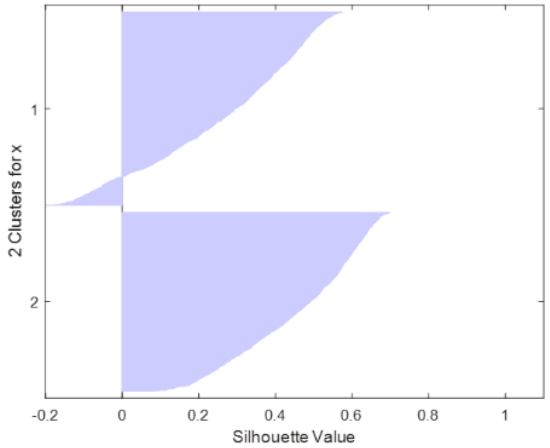
4 bin: volatileacidity, freesulfurdioxide, density, alcohol

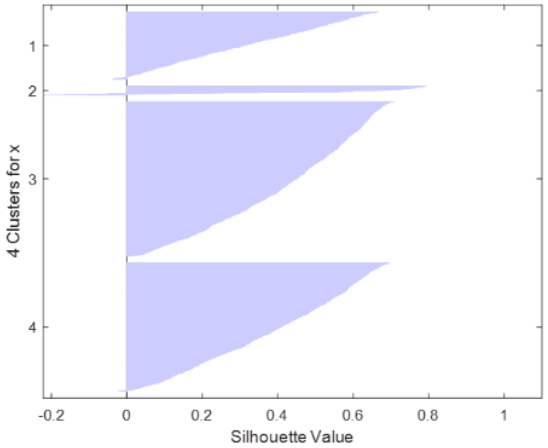


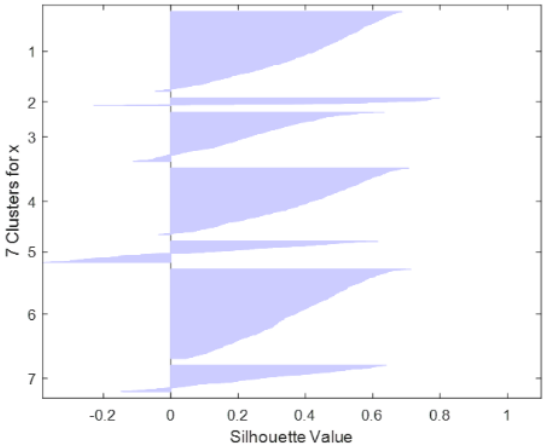
7 bin: freesulfurdioxide, volatileacidity, density, chlorides, alcohol



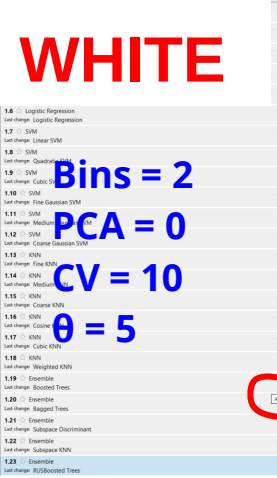
Again: Any Natural Cluster? K-Means

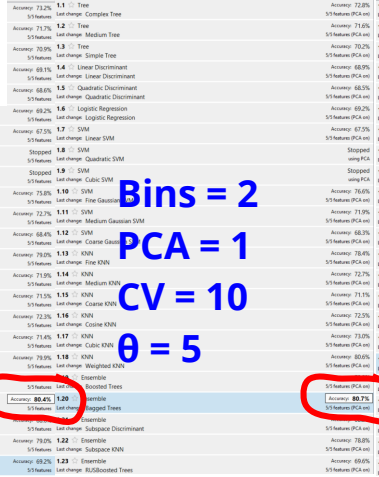


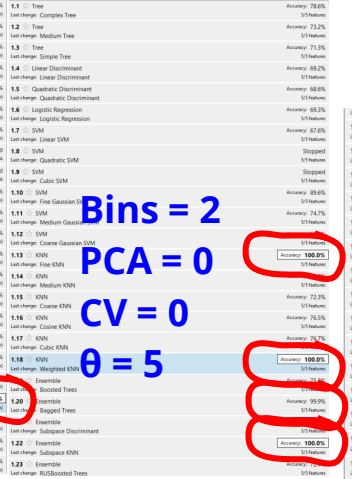


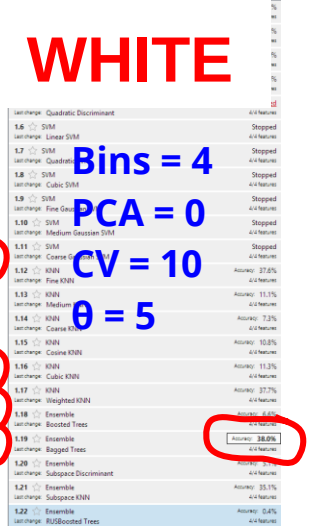


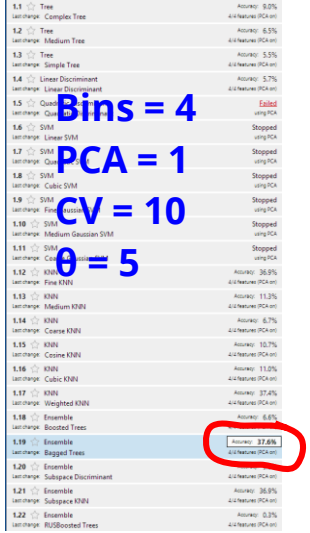
Before: cluster2 = 0.3533, cluster4 = 0.2012, cluster7 = 0.1741
After: cluster2 = 0.3562, cluster4 = 0.3895, cluster7 = 0.3344

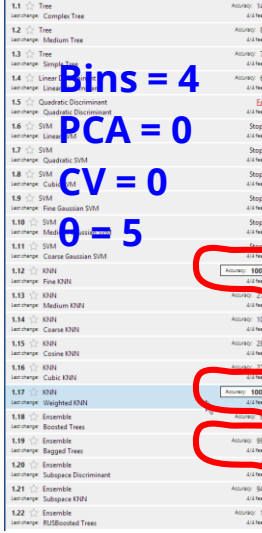

WHITE
Bins = 2
PCA = 0
CV = 10
θ = 5

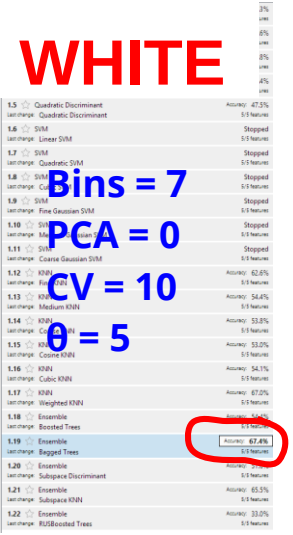

Bins = 2
PCA = 1
CV = 10
θ = 5

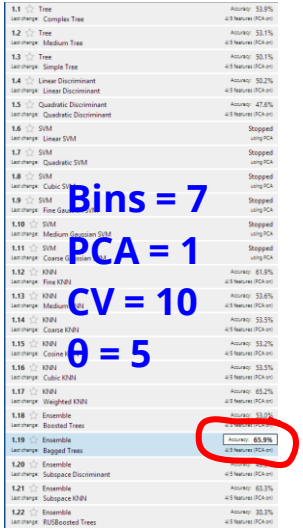

Bins = 2
PCA = 0
CV = 0
θ = 5

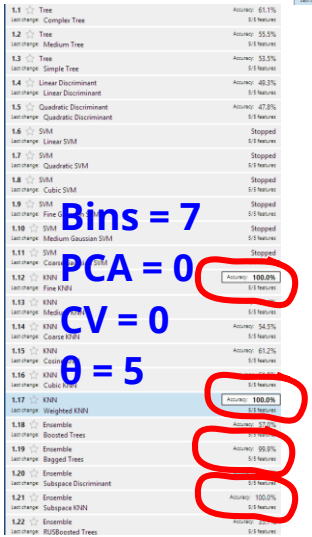

WHITE
Bins = 4
PCA = 0
CV = 10
θ = 5


Bins = 4
PCA = 1
CV = 10
θ = 5


Bins = 4
PCA = 0
CV = 0
θ = 5


WHITE
Bins = 7
PCA = 0
CV = 10
θ = 5


Bins = 7
PCA = 1
CV = 10
θ = 5


Bins = 7
PCA = 0
CV = 0
θ = 5

WHITE

Bins = 2
PCA = 0
CV = 10
θ = 5

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

Bins = 2
PCA = 1
CV = 10
θ = 5

1.1 Tree	Last change: Complex Tree	Accuracy: 72.8% 5/5 features (PCA on)
1.2 Tree	Last change: Medium Tree	Accuracy: 71.6% 5/5 features (PCA on)
1.3 Tree	Last change: Simple Tree	Accuracy: 70.2% 5/5 features (PCA on)
1.4 Linear Discriminant	Last change: Linear Discriminant	Accuracy: 68.9% 5/5 features (PCA on)
1.5 Quadratic Discriminant	Last change: Quadratic Discriminant	Accuracy: 68.5% 5/5 features (PCA on)
1.6 Logistic Regression	Last change: Logistic Regression	Accuracy: 69.2% 5/5 features (PCA on)
1.7 SVM	Last change: Linear SVM	Accuracy: 67.5% 5/5 features (PCA on)
1.8 SVM	Last change: Quadratic SVM	Stopped using PCA
1.9 SVM	Last change: Cubic SVM	Stopped using PCA
1.10 SVM	Last change: Fine Gaussian SVM	Accuracy: 76.6% 5/5 features (PCA on)
1.11 SVM	Last change: Medium Gaussian SVM	Accuracy: 71.9% 5/5 features (PCA on)
1.12 SVM	Last change: Coarse Gaussian SVM	Accuracy: 68.3% 5/5 features (PCA on)
1.13 KNN	Last change: Fine KNN	Accuracy: 78.4% 5/5 features (PCA on)
1.14 KNN	Last change: Medium KNN	Accuracy: 72.7% 5/5 features (PCA on)
1.15 KNN	Last change: Coarse KNN	Accuracy: 71.1% 5/5 features (PCA on)
1.16 KNN	Last change: Cosine KNN	Accuracy: 72.5% 5/5 features (PCA on)
1.17 KNN	Last change: Cubic KNN	Accuracy: 73.0% 5/5 features (PCA on)
1.18 KNN	Last change: Weighted KNN	Accuracy: 80.6% 5/5 features (PCA on)
1.19 Ensemble	Last change: Boosted Trees	5/5 features (PCA on)
1.20 Ensemble	Last change: Bagged Trees	Accuracy: **80.7%** 5/5 features (PCA on)
1.21 Ensemble	Last change: Subspace Discriminant	5/5 features (PCA on)
1.22 Ensemble	Last change: Subspace KNN	Accuracy: 78.8% 5/5 features (PCA on)
1.23 Ensemble	Last change: RUSBoosted Trees	Accuracy: 69.6% 5/5 features (PCA on)

Bins = 2
PCA = 0
CV = 0
θ = 5

1.1 Tree	Last change: Complex Tree	Accuracy: 78.6% 5/5 features
1.2 Tree	Last change: Medium Tree	Accuracy: 73.2% 5/5 features
1.3 Tree	Last change: Simple Tree	Accuracy: 71.3% 5/5 features
1.4 Linear Discriminant	Last change: Linear Discriminant	Accuracy: 69.2% 5/5 features
1.5 Quadratic Discriminant	Last change: Quadratic Discriminant	Accuracy: 68.6% 5/5 features
1.6 Logistic Regression	Last change: Logistic Regression	Accuracy: 69.3% 5/5 features
1.7 SVM	Last change: Linear SVM	Accuracy: 67.6% 5/5 features
1.8 SVM	Last change: Quadratic SVM	Stopped 5/5 features
1.9 SVM	Last change: Cubic SVM	Stopped 5/5 features
1.10 SVM	Last change: Fine Gaussian SVM	Accuracy: 89.6% 5/5 features
1.11 SVM	Last change: Medium Gaussian SVM	Accuracy: 74.7% 5/5 features
1.12 SVM	Last change: Coarse Gaussian SVM	5/5 features
1.13 KNN	Last change: Fine KNN	Accuracy: **100.0%** 5/5 features
1.14 KNN	Last change: Medium KNN	5/5 features
1.15 KNN	Last change: Coarse KNN	Accuracy: 72.3% 5/5 features
1.16 KNN	Last change: Cosine KNN	Accuracy: 76.5% 5/5 features
1.17 KNN	Last change: Cubic KNN	Accuracy: 76.7% 5/5 features
1.18 KNN	Last change: Weighted KNN	Accuracy: **100.0%** 5/5 features
1.19 Ensemble	Last change: Boosted Trees	Accuracy: 75.2% 5/5 features
1.20 Ensemble	Last change: Bagged Trees	Accuracy: 99.9% 5/5 features
1.21 Ensemble	Last change: Subspace Discriminant	5/5 features
1.22 Ensemble	Last change: Subspace KNN	Accuracy: **100.0%** 5/5 features
1.23 Ensemble	Last change: RUSBoosted Trees	5/5 features

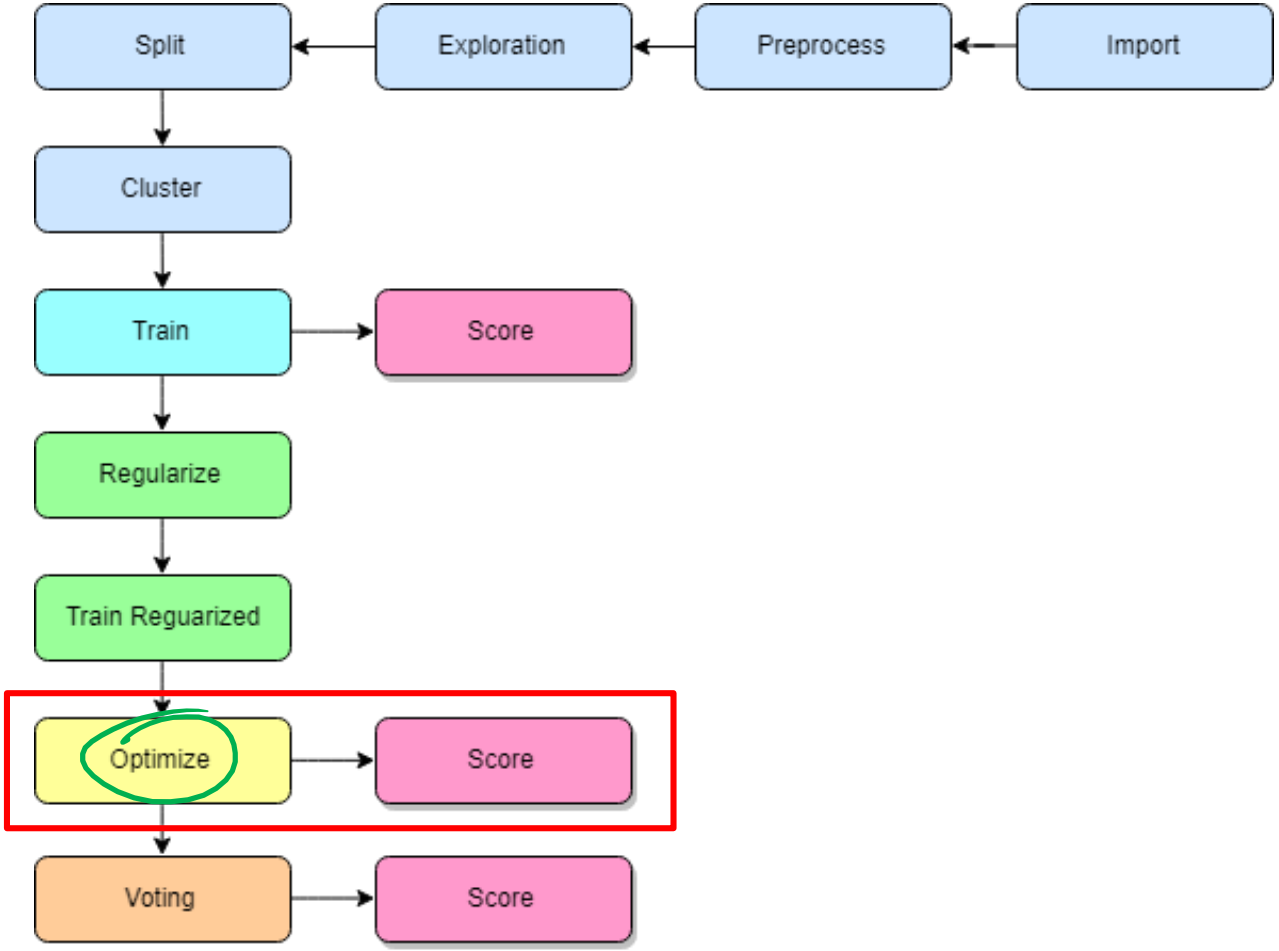
170 Post-Regularized Models

- 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**

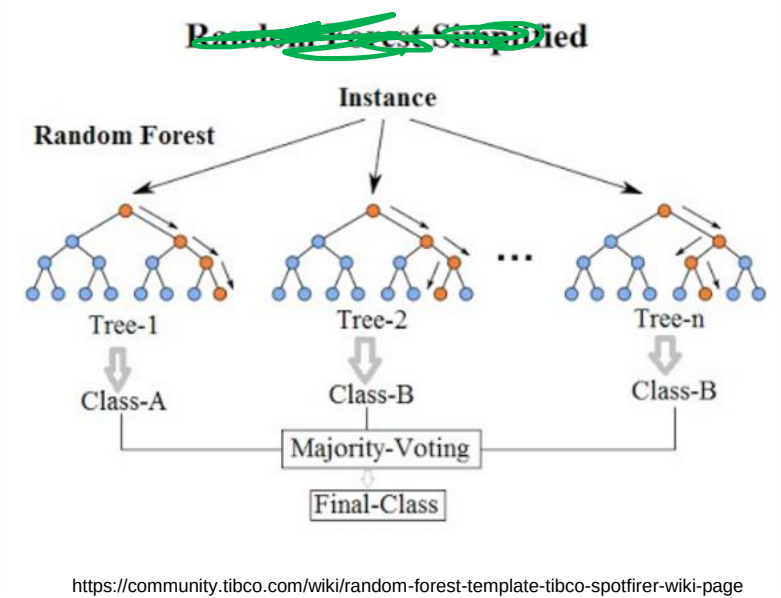


Adapted from: https://docs.microsoft.com/en-us/azure/machine-learning/studio/studio-overview-diagram

Best Individual Model - Ensemble Bagged Tree

```
classificationEnsemble0 = fitensemble(...
    predictors, ...
    response, ...
    'Bag', ...
    30, ...  % # trees
    'Tree', ...
    'Type', 'Classification', ...
    'ClassNames', [1; 2]);
```

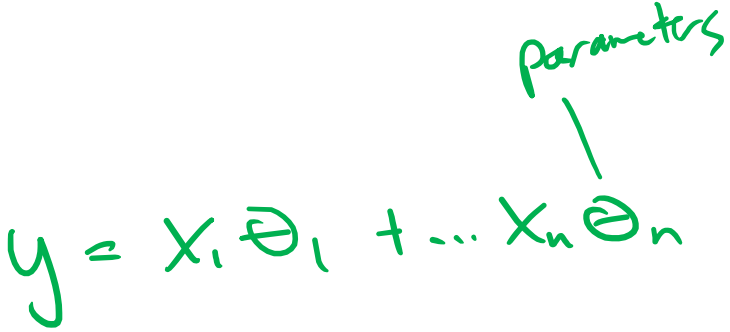
Training time = 15.68 seconds



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**



History

1 ☆ Ensemble — Accuracy: 67.1%

Last change: Bagged Trees — 5/5 features

trees

splits

Current model

Model number 1

Status: Trained

Accuracy: 67.1%

Prediction speed: ~7600 obs/sec

Training Time: 15.681 secs

primary

Classifier

Preset: Bagged Trees

Ensemble method: Bag

Learner type: Decision tree

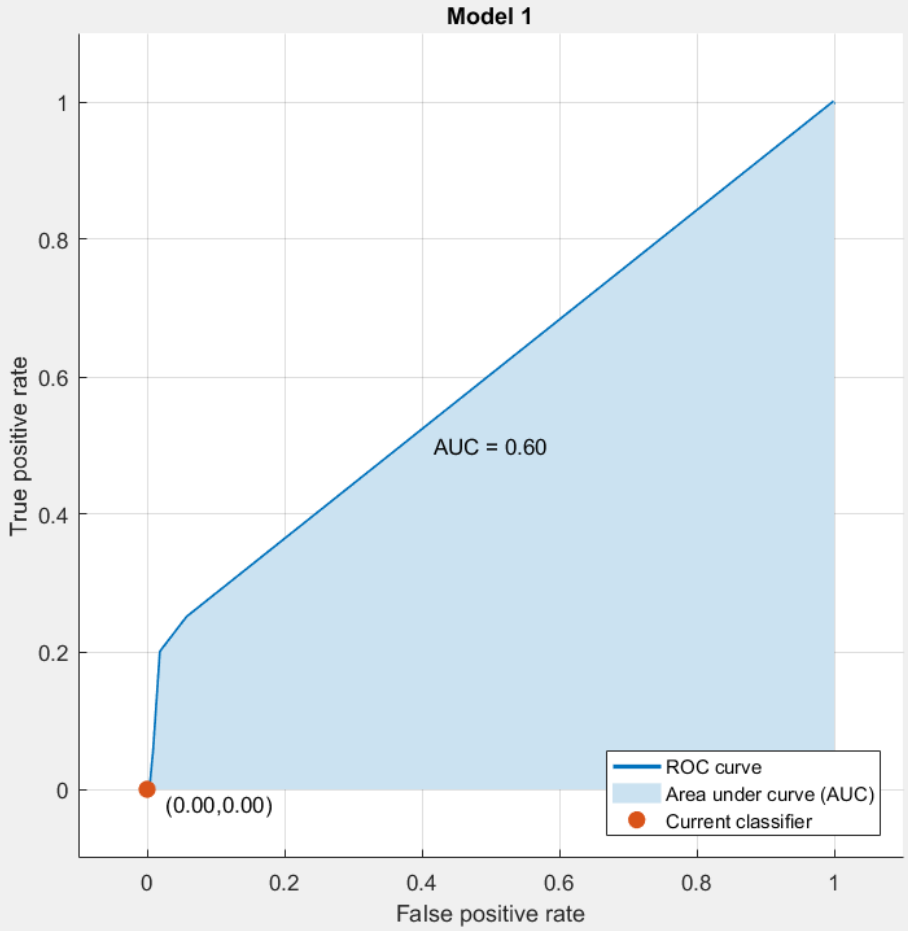
Number of learners: 30

Feature Selection

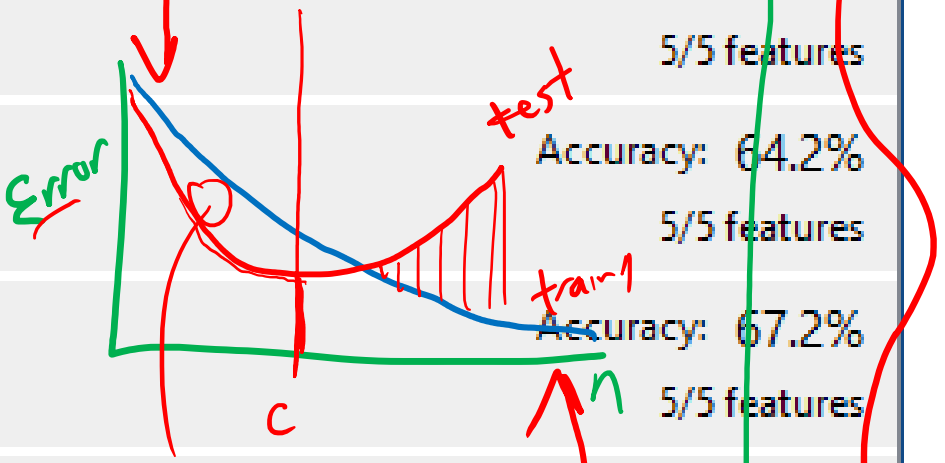
All features used in the model, before PCA

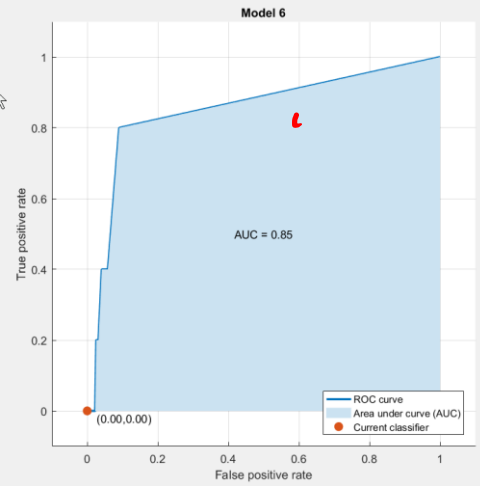
PCA

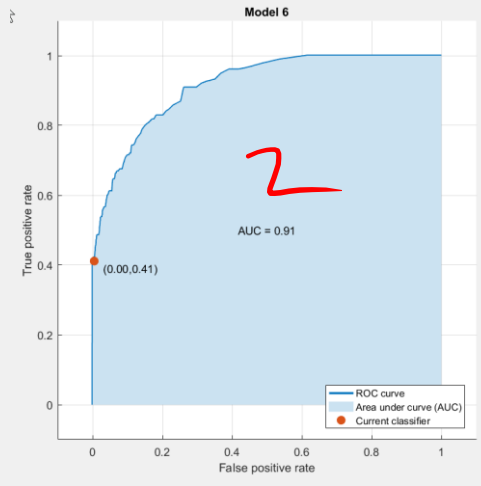
PCA disabled

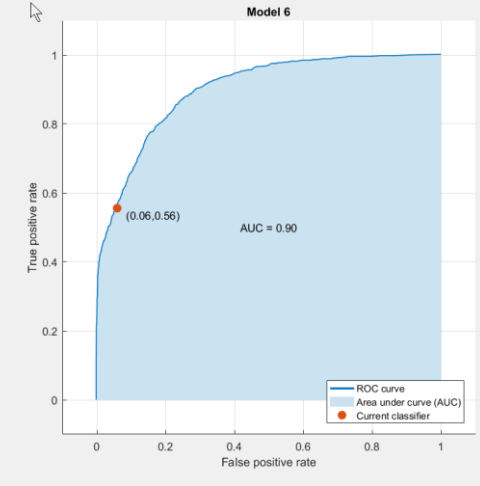


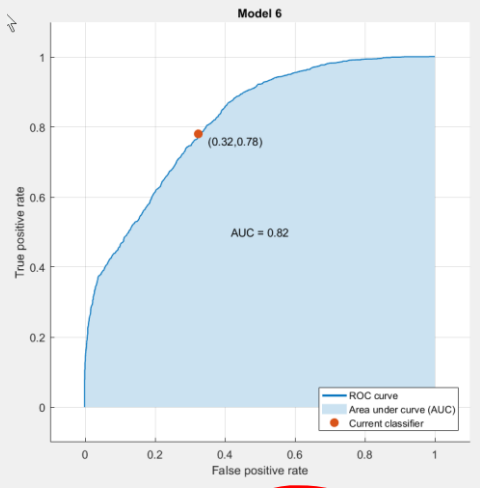
1	Ensemble	Last change: 'NumLearners' = '1'	Accuracy: 54.8% 5/5 features
2	Ensemble	Last change: 'NumLearners' = '5'	Accuracy: 63.5% 5/5 features
3	Ensemble	Last change: 'NumLearners' = '10'	Accuracy: 64.2% 5/5 features
4	Ensemble	Last change: 'NumLearners' = '50'	Accuracy: 67.2% 5/5 features
5	Ensemble	Last change: 'NumLearners' = '100'	Accuracy: 67.7% 5/5 features
6	Ensemble	Last change: 'NumLearners' = '500'	Accuracy: **68.0%** 5/5 features

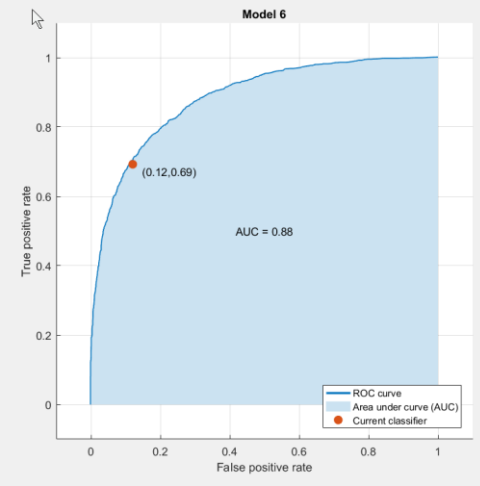


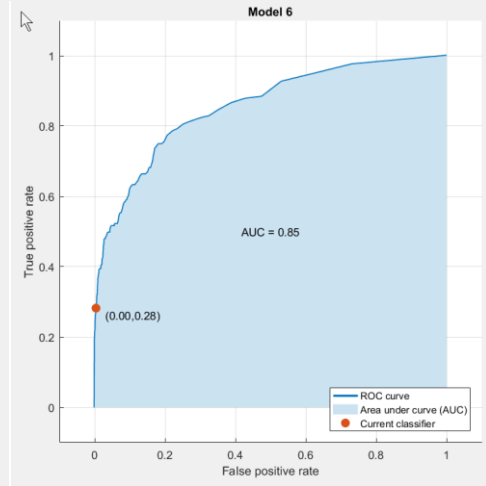

Model 6
AUC = 0.85
(0.00,0.00)
ROC curve
Area under curve (AUC)
Current classifier
True positive rate
False positive rate

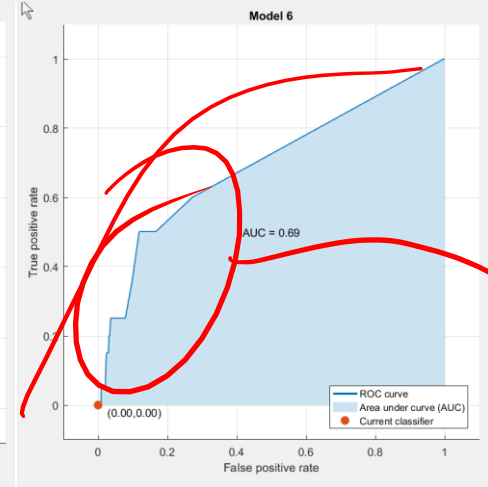

Model 6
AUC = 0.91
(0.00,0.41)
ROC curve
Area under curve (AUC)
Current classifier
True positive rate
False positive rate


Model 6
AUC = 0.90
(0.06,0.56)
ROC curve
Area under curve (AUC)
Current classifier
True positive rate
False positive rate


Model 6
AUC = 0.82
(0.32,0.78)
ROC curve
Area under curve (AUC)
Current classifier
True positive rate
False positive rate

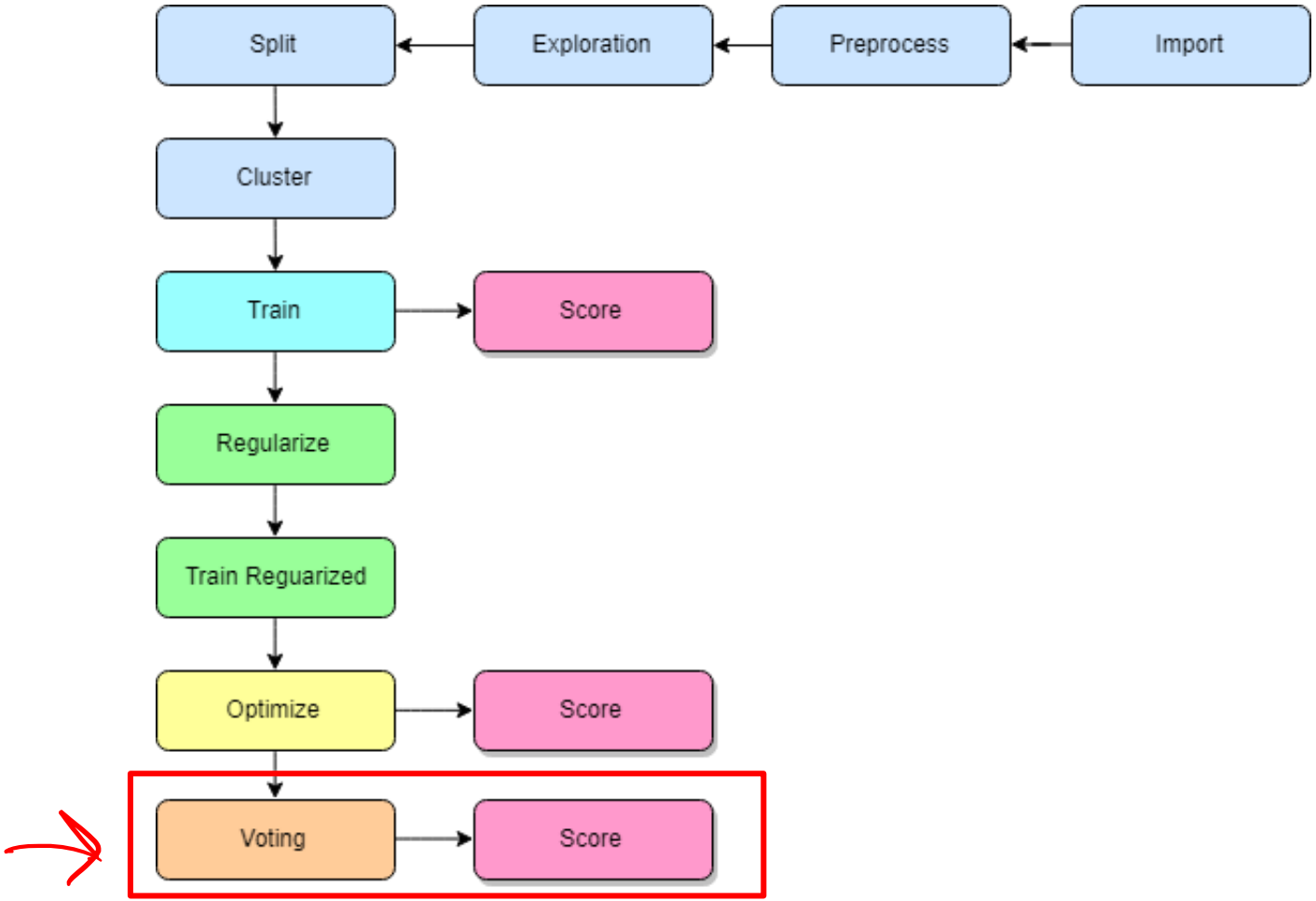

Model 6
AUC = 0.88
(0.12,0.69)
ROC curve
Area under curve (AUC)
Current classifier
True positive rate
False positive rate


Model 6
AUC = 0.85
(0.00,0.28)
ROC curve
Area under curve (AUC)
Current classifier
True positive rate
False positive rate


Model 6
AUC = 0.69
(0.00,0.00)
ROC curve
Area under curve (AUC)
Current classifier
True positive rate
False positive rate


Model 6
1 10 9
46 81 3 2
15 1009 410 21 2
2 302 1716 168 10
16 365 489 10
45 72
3 2
True class
Predicted class


Model 6
True class
Predicted class
1
2
3
4
5
6
7
5%
50%
45%
100%
28%
50%
21%
1%
28%
72%
1%
69%
28%
1%
<1%
69%
31%
<1%
14%
78%
8%
<1%
78%
22%
2%
41%
56%
1%
56%
44%
3%
30%
26%
41%
41%
59%
60%
40%
100%
True Positive Rate
False Negative Rate

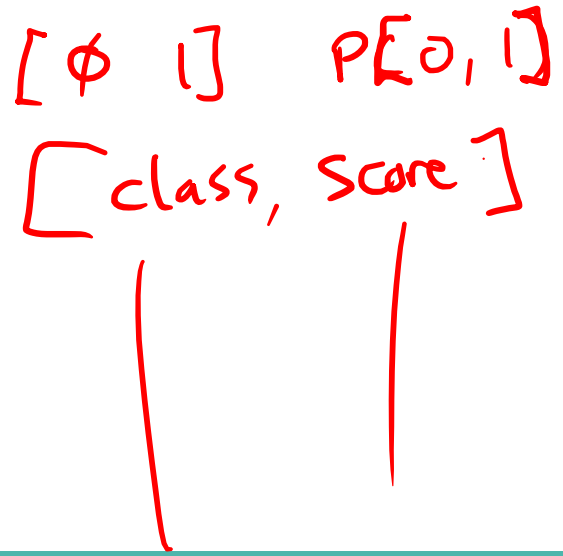


Adapted from: https://docs.microsoft.com/en-us/azure/machine-learning/studio/studio-overview-diagram

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



Hard Voting

```
class_2_matrix = [...
    class_2_baggedensemble...
    class_2_ComplexDT...
    class_2_fineGaussianSVM ...
    class_2_weightedKNN];

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

	1	2	3	4
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	2	3	4	5	6
1	0.1365	0.8635	0.0667	0.9333	0	1
2	0.1408	0.8592	0.0667	0.9333	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

Soft Voting

% 0 predictors
I(I == 1 | I == 3 | I == 5) = 8;

% 1 predictors
I(I == 2 | I == 4 | I == 6) = 9;

I(I == 8) = 0;
I(I == 9) = 1;

votesSoft_class_2 = I;

	1	2	3	4	5	6
1	0.1365	0.8635	0.0667	0.9333	0	1
2	0.1408	0.8592	0.0667	0.9333	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 vs Soft Comparison

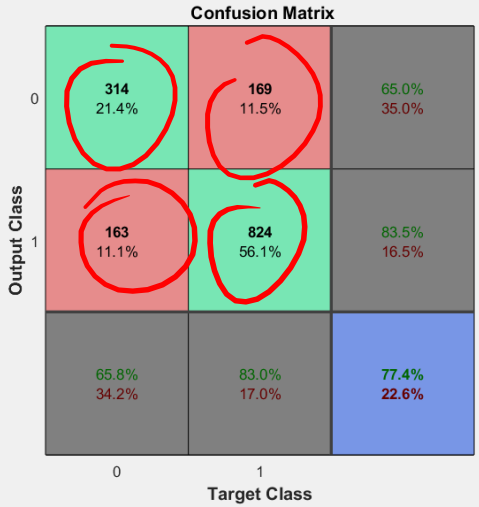
- 61 different
- **4.1% different**
- Soft voting predicts more class 1 (Good Wine)

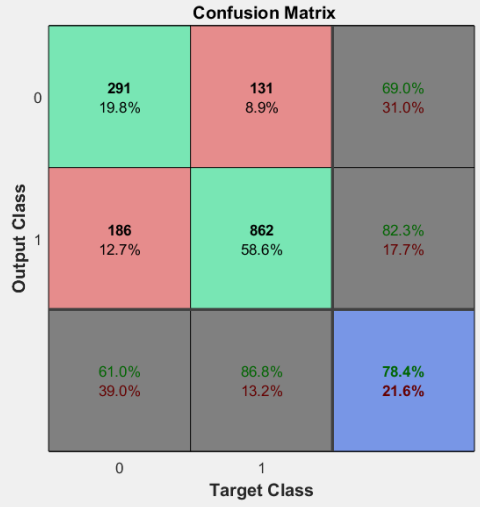
hard	soft
1	1
1	1
1	1
1	1
0	0
0	0
1	0
0	0
1	1
1	1
0	0
1	1
1	1
1	1
1	1
0	0
1	1
0	0
1	1
1	1
1	1
1	1
0	0
1	1
1	1
1	1

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





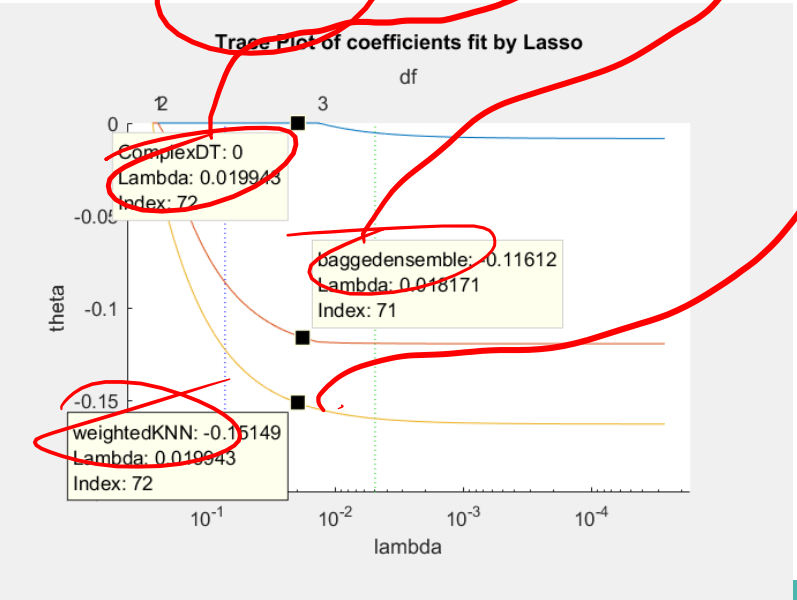
Model of Models

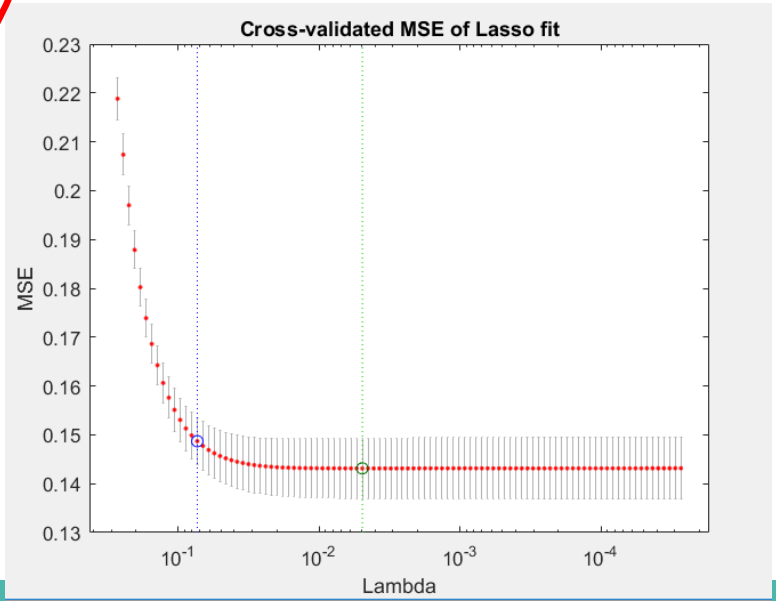
score

$y = X_1\theta_1 + X_2\theta_2 + X_3\theta_3$

weights weights

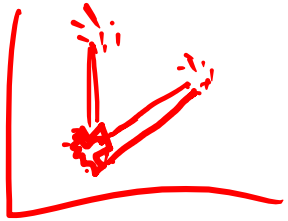
```
kfold = 10;
alpha = 1;
[B FitInfo] = lasso(X, Y, 'CV', kfold, 'Alpha', alpha);
```





Regrets

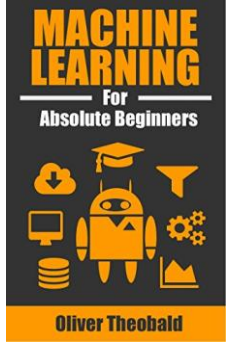
- Built Model of Models with **Test** data
- Standardized data **before** train-test split
- Train-Test Split **rearranged** categories
  - Only three class 9 observations in full dataset. Test didn't have a 9. Classes later indexed to 0, causing misalignment (suspicion).

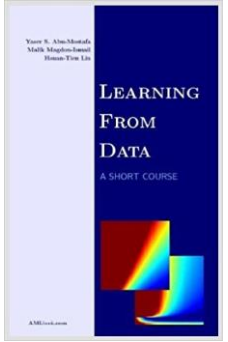


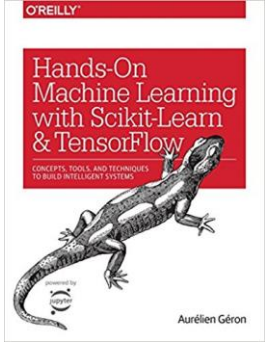
Wish List

- 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
- Explore **Red Wine** dataset

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!