SuperLearner

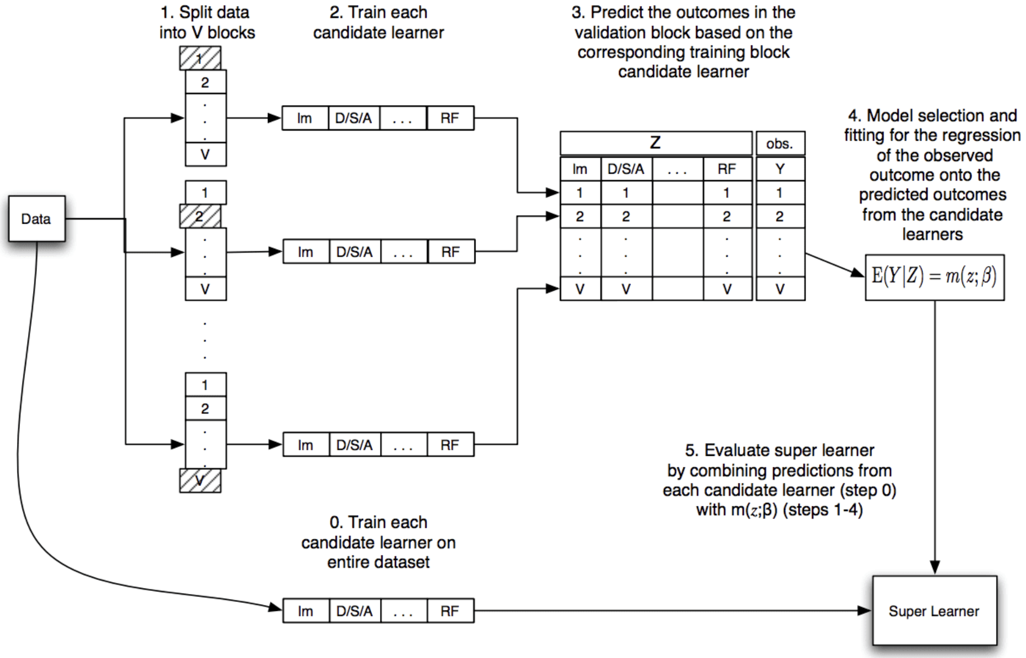
BAIARDI LORENZO

FOUNDATIONS OF STATISTICAL LEARNING

What is the SuperLearner?

The SuperLearner is a machine learning algorithm that combines different base models to achieve more accurate predictions. The base models are trained on training data and evaluated on their performance before being combined, with weights assigned based on their past performance.

The idea behind the SuperLearner is to leverage the advantages of the different base models used in order to reduce the risk of making incorrect predictions due to defects or specific limitations of a single model.



Steps of SL

- Selection of base models: linear regression, decision trees, SVM, …
- Training of base models.
- Evaluation of performance of base models.
- Creation of the SuperLearner model: assigning weights to the base models.
- Evaluation of the performance of the SuperLearner.

Pseudocode

1. Data: $D = \{(x_i, y_i)\}_{i=1}^{n}$, Library: $L = \{\Psi_k(X)\}_{k=1}^{K}$
2. Fit each algorithm in L on the entire data set D to estimate $\widehat{\Psi}_k(X)$
3. V-fold cross-validation on $D \rightarrow$ training samples $T(\nu)$ and the corresponding validation samples $V\,(\nu) = D\backslash T(\nu)\ (v\ =\ 1,\ldots,V)$
4. For the each v-fold:
   - $X_{T\,(\nu)} = (X_i\!: i \in T(\nu)), X_{V\,(\nu)} = (X_i\!: i \in V(\nu))$
   - Fit each learner in the library on $T(\nu) \rightarrow \widehat{\Psi}_{k,T(\nu)}\left(X_{T\,(\nu)}\right)$. Compute the predicted values on the corresponding validation set $\longrightarrow \widehat{\Psi}_{k,T(v)}\left(X_{V\,(\nu)}\right)$
5. Stack the predictions in a n × K matrix:

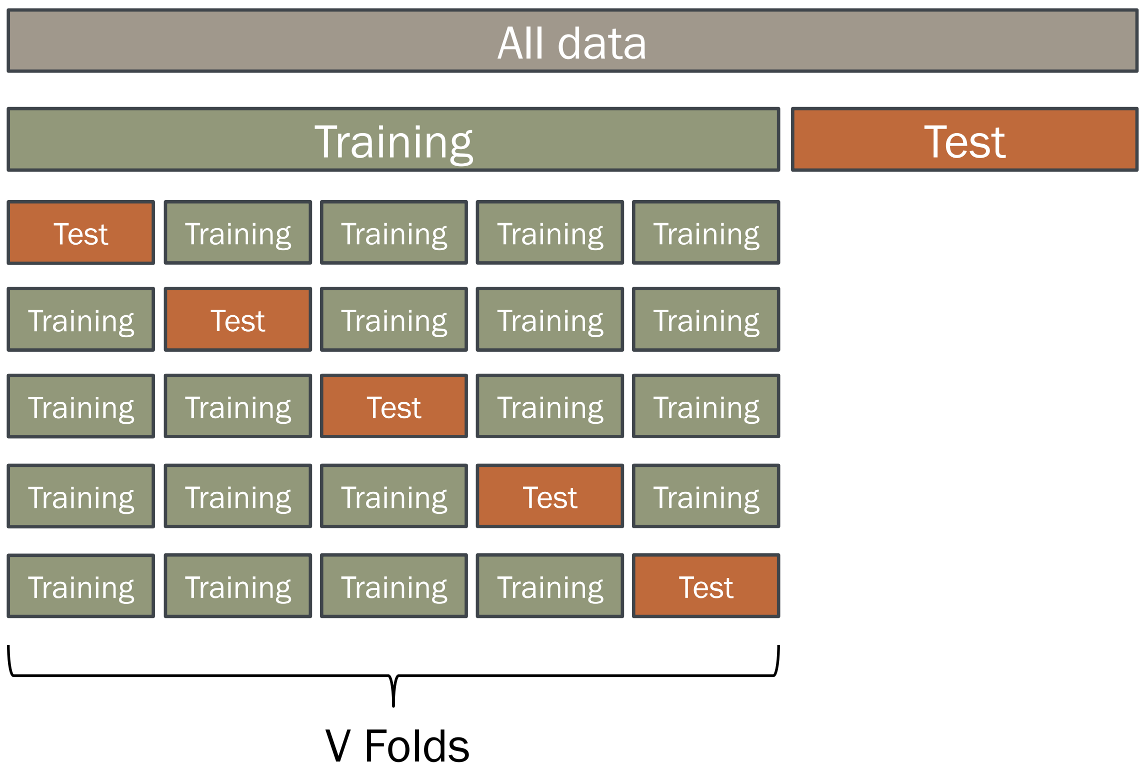
   $\mathrm{Z} = \{\widehat{\Psi}_{k,T(v)}\left(X_{V\,(\nu)}\right), v = 1,\ldots,V\ \&\ k = 1,\ldots,K\}$

Pseudocode

6. Combinations of the candidate learners, weighted by a vector α :
   - $m(z \mid \alpha) \; = \; \sum_{k=1}^{K} \alpha_k \widehat{\Psi}_{k,T(v)}\big(X_{V\,(v)}\big)$
   - $\alpha_k \geq \; 0 \; \forall k, \; \sum_{k=1}^{K} \alpha_k \; = 1$
7. Determine the α that minimizes the cross-validated risk over all allowed α-combinations:
   - $\hat{\alpha} = \arg\min_{\alpha} \sum_{i=1}^{n} (Y_i - m(z \mid \alpha))^2$
8. Combine $\hat{\alpha}$ with $\widehat{\Psi}_k$:
   - $\widehat{\Psi}_{SL} = \sum_{k=1}^{K} \hat{\alpha}_k \widehat{\Psi}_{k,T(v)}\big(X_{V\,(v)}\big)$

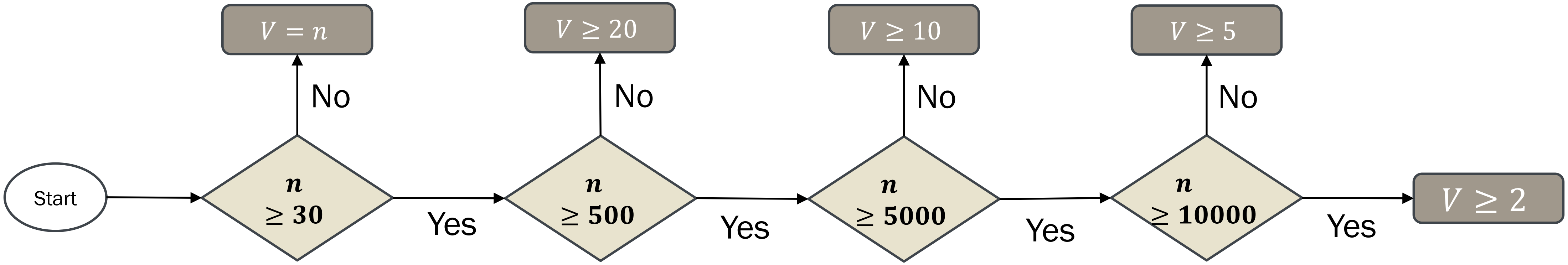
VFCV –
V Folds Cross Validation

VFCV involves splitting the dataset into V distinct validation sets and their corresponding training sets. For each fold, each candidate model is trained using the observations in the training set, and then predictions are made for the observations in the validation set.



Selecting the right number of folds

The choice of the number of folds, V, is important because it impacts the bias and variance of the predictive performance of cross-validation (CV Risk).



Note: Recommends LOOCV only for very small n_{eff} to **reduce the bias of the risk estimate.**

Meta Learner

NNLS

This is the main approach use on SL for the meta-level. The non-negative least squares is a type of constrained least squares problem where the coefficients are non-negative. After obtaining the weights through NNLS, they should be normalized.

OTHER

It’s possible to use other statistical learning algorithms to evaluate the weights of meta-level.

It is also possible to use Meta-Learners that can predict the output value directly from the predictions of the base learners without having to calculate weights.

Libraries in SL

When we are familiar with the DGP, it is important to choose libraries that best fit the problem.

Conversely, when we are not familiar with the DGP, the library should be as broad and diverse as possible.

- Base Learners: An algorithm that is not fully specified but defines a particular learning. A base learner is used as a building block to define one or more fully specified learners.
- Screener Couplings: A function that returns a subset of X. Covariate screening is essential when the dimensionality of the data is very large, and it can be practically useful in any SL or machine learning application.

Note: When the number of observations in the dataset is small, it is preferable to use generic base learners instead of neural networks, in fact neural networks require a large amount of data to optimize their parameters effectively.

Libraries in SL

When we are familiar with the DGP, it is important to choose libraries that best fit the problem.

Conversely, when we are not familiar with the DGP, the library should be as broad and diverse as possible.

```
All prediction algorithm wrappers in SuperLearner:

 [1] "SL.bartMachine"      "SL.bayesglm"         "SL.biglasso"
 [4] "SL.caret"            "SL.caret.rpart"      "SL.cforest"
 [7] "SL.earth"            "SL.extraTrees"       "SL.gam"
[10] "SL.gbm"              "SL.glm"              "SL.glm.interaction"
[13] "SL.glmnet"           "SL.ipredbagg"        "SL.kernelKnn"
[16] "SL.knn"              "SL.ksvm"             "SL.lda"
[19] "SL.leekasso"         "SL.lm"               "SL.loess"
[22] "SL.logreg"           "SL.mean"             "SL.nnet"
[25] "SL.nnls"             "SL.polymars"         "SL.qda"
[28] "SL.randomForest"     "SL.ranger"           "SL.ridge"
[31] "SL.rpart"            "SL.rpartPrune"       "SL.speedglm"
[34] "SL.speedlm"          "SL.step"             "SL.step.forward"
[37] "SL.step.interaction" "SL.stepAIC"          "SL.svm"
[40] "SL.template"         "SL.xgboost"
```

Libraries in SL

When we are familiar with the DGP, it is important to choose libraries that best fit the problem.

Conversely, when we are not familiar with the DGP, the library should be as broad and diverse as possible.

```
All screening algorithm wrappers in SuperLearner:

[1] "All"
[1] "screen.corP"           "screen.corRank"        "screen.glmnet"
[4] "screen.randomForest"   "screen.SIS"            "screen.template"
[7] "screen.ttest"          "write.screen.template"
```

Advantages of SL

- Improved accuracy: overall more precise predictions compared to using a single model.
- Reduction of overfitting risk: less accurate models can compensate for any limitations or biases present in a single model.
- Adaptability: The SuperLearner can be adapted to various machine learning problems.
- Flexibility: it can be used with a wide range of base models.

Disadvantages of SL

Computational complexity… but it is possible to parallelize it.

Dependency on the base models.

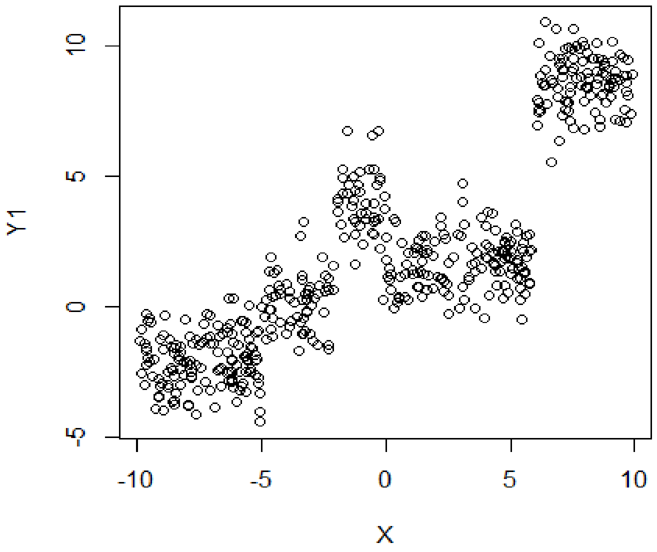
Difficulty in interpretation: decisions are influenced by multiple models with different weights.

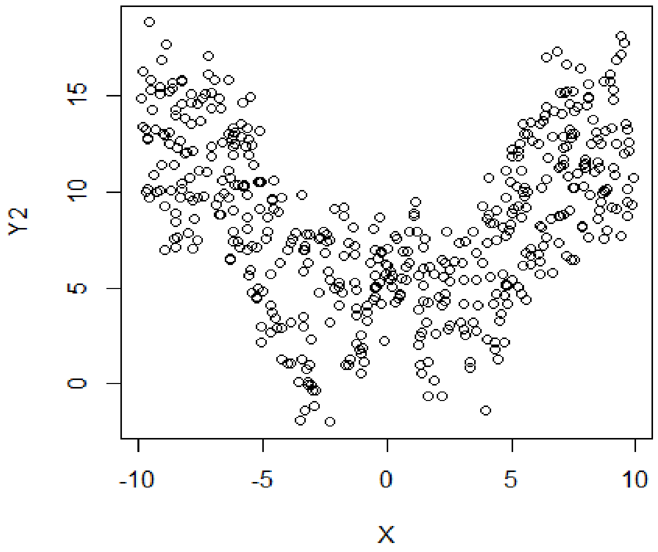
Experiments in R – Generated Data

22/06/2023

Generated Data

```
# Generated Data
n <- 500
noise <- rnorm(n)
X <- runif(n, -10, 10)

Y1 <- (-2*(ifelse(X<(-5),1,0))+4*(ifelse(X>(-2),1,0))
       -2.4*(ifelse(X>0,1,0))+7*(ifelse(X>6,1,0))+noise)

Y2 <- (4+2*cos(X)+0.125*(X^2)+4*sin(X^5)+noise)
```





Matrix of predictions

Each learner of the library generates a set of prediction during the VFCV.

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	1.44962716	1.335042139	1.6026027	1.712997107	2.167225	3.02935946
[2,]	9.08682442	9.294965245	7.9834497	8.567821120	2.047104	6.37856630
[3,]	1.39822936	1.512922351	4.9269234	1.676030338	2.270386	5.54621852
[4,]	-1.56702173	-1.644458735	-2.1802368	-1.976425244	2.228481	-1.75094368
[5,]	1.96659005	2.057738353	2.4144823	1.052881684	2.200852	2.36807584
[6,]	-1.71889007	-1.548508113	-2.2887798	-2.182648685	2.131630	-2.13723538
[7,]	2.09185958	2.066257528	3.2379742	1.893143745	2.047104	4.96248673
[8,]	8.62070847	8.501262053	8.7635890	8.506975749	2.167225	6.88051013
[9,]	7.89210844	7.704032993	7.8550387	8.694503754	2.078926	6.33560843
[10,]	-0.31757286	-0.693315060	0.6405515	0.288493641	2.228481	0.06221601
[11,]	1.84839880	1.989815672	4.3848789	1.381433172	2.227933	5.38142258
[12,]	-2.41663098	-2.757774682	-1.6361272	-1.839210727	2.270386	-1.12577796
[13,]	-1.35584724	-1.531667980	-2.1197716	-1.806723818	2.235165	-1.77877190
[14,]	-3.39173913	-3.161451033	-2.0501813	-2.206172508	2.228481	-3.30284823
[15,]	-2.10069036	-2.304639053	-2.0303068	-1.920992676	2.228481	-1.53860044
[16,]	-2.36859727	-2.543463208	-1.9783084	-2.257329489	2.078926	-1.42037307
[17,]	-2.25754929	-2.379853060	-2.1315136	-1.914445514	2.047104	-1.62546912
[18,]	-2.02295923	-1.810003842	-1.7766628	-1.975382225	2.212771	-1.26131762
[19,]	1.67720020	1.713015113	1.7774316	1.715567237	2.167225	2.87553250
[20,]	-3.03898883	-3.054974252	-2.1879818	-2.470473823	2.227933	-2.69245156
[21,]	2.19416547	2.217609921	1.7530105	2.148687492	2.167225	4.49083355
[22,]	0.11206632	0.199837753	0.9993592	0.097904671	2.047104	0.32118794
[23,]	7.68517017	7.676500646	7.6905530	8.543278297	2.131630	6.30430177
[24,]	0.42466328	0.223166670	-0.5394452	0.184035479	2.227933	-0.41399555
[25,]	-1.68644249	-1.835639597	-1.9363095	-1.538602606	2.078926	-3.33446308
[26,]	-1.59362423	-1.525141828	-1.7477597	-1.466990452	2.131630	-3.43385912
[27,]	-2.37672019	-2.561535813	-2.1242450	-2.242876099	2.235165	-2.74325514
[28,]	1.47637725	1.625788696	2.2330906	1.943659135	2.200852	4.65709334
[29,]	8.34613895	8.247364418	7.9133760	8.641699145	2.047104	6.35091567
[30,]	-1.92583263	-1.960769535	-2.1222202	-2.328413728	2.227933	-2.87044925

Results

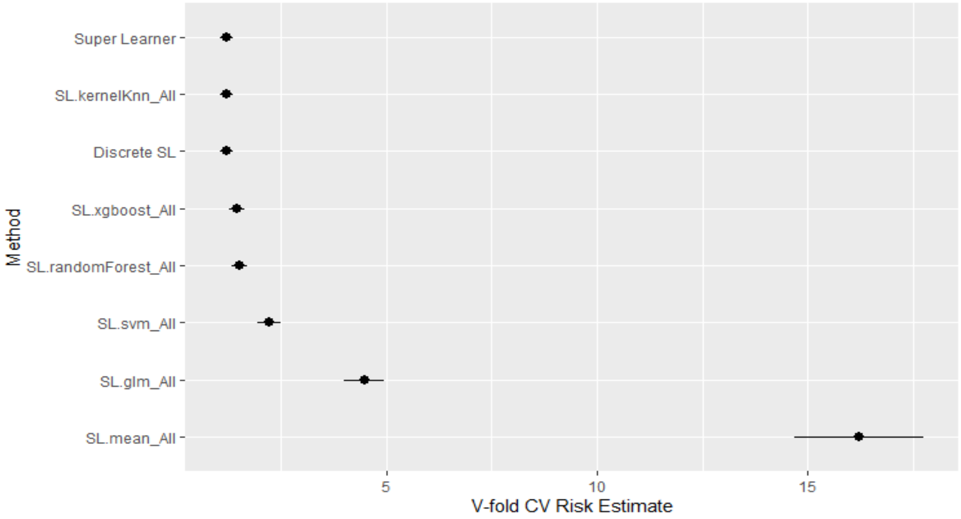
```r
sl_lib = c("SL.xgboost", "SL.randomForest", "SL.svm",
           "SL.kernelKnn", "SL.mean", "SL.glm")

SL.1 <- SuperLearner(Y=Y1, X=X, family=gaussian(), method="method.NNLS",
                     SL.library=sl_lib, verbose=TRUE)

SL.2 <- SuperLearner(Y=Y2, X=X, family=gaussian(), method="method.NNLS",
                     SL.library=sl_lib, verbose=TRUE)
```

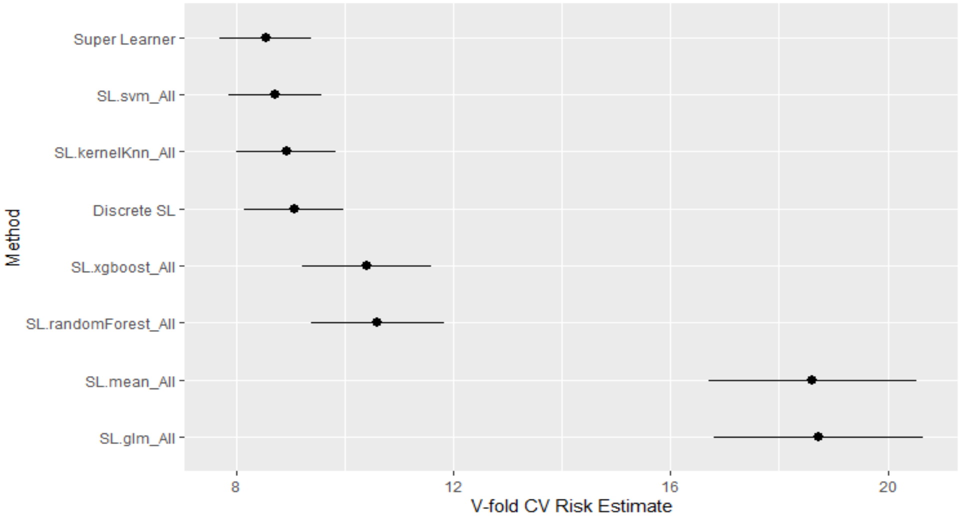
```
> SL.1

Call:
SuperLearner(Y = Y1, X = X, family = gaussian(),
SL.library = sl_lib, method = "method.NNLS",
    verbose = TRUE)


                          Risk       Coef
SL.xgboost_All        1.502276 0.00000000
SL.randomForest_All   1.555147 0.00000000
SL.svm_All            2.227123 0.06137821
SL.kernelKnn_All      1.193453 0.93862179
SL.mean_All          16.258294 0.00000000
SL.glm_All            4.471264 0.00000000
```

```
> SL.2

Call:
SuperLearner(Y = Y2, X = X, family = gaussian(),
SL.library = sl_lib, method = "method.NNLS",
    verbose = TRUE)


                          Risk        Coef
SL.xgboost_All       10.743335 0.000000000
SL.randomForest_All  10.983361 0.001971702
SL.svm_All            8.731799 0.597695021
SL.kernelKnn_All      8.985437 0.400333277
SL.mean_All          18.639758 0.000000000
SL.glm_All           18.763586 0.000000000
```

Results

```
#CV SuperLearner
CV.SL.1 <- CV.SuperLearner(Y=Y1, X=X, family=gaussian(), method="method.NNLS",
                           SL.library=sl_lib, verbose=TRUE)

CV.SL.2 <- CV.SuperLearner(Y=Y2, X=X, family=gaussian(), method="method.NNLS",
                           SL.library=sl_lib, verbose=TRUE)
```





Experiments in R – Dataset Adult

22/06/2023

Dataset Adult

Dataset Characteristics	**Subject Area**	**Associated Tasks**
Multivariate	Social	Classification
Attribute Type	**# Instances**	**# Attributes**
Categorical, Integer	48842	14

Task: Predict whether income exceeds $50K/yr based on census data

Dataset Adult

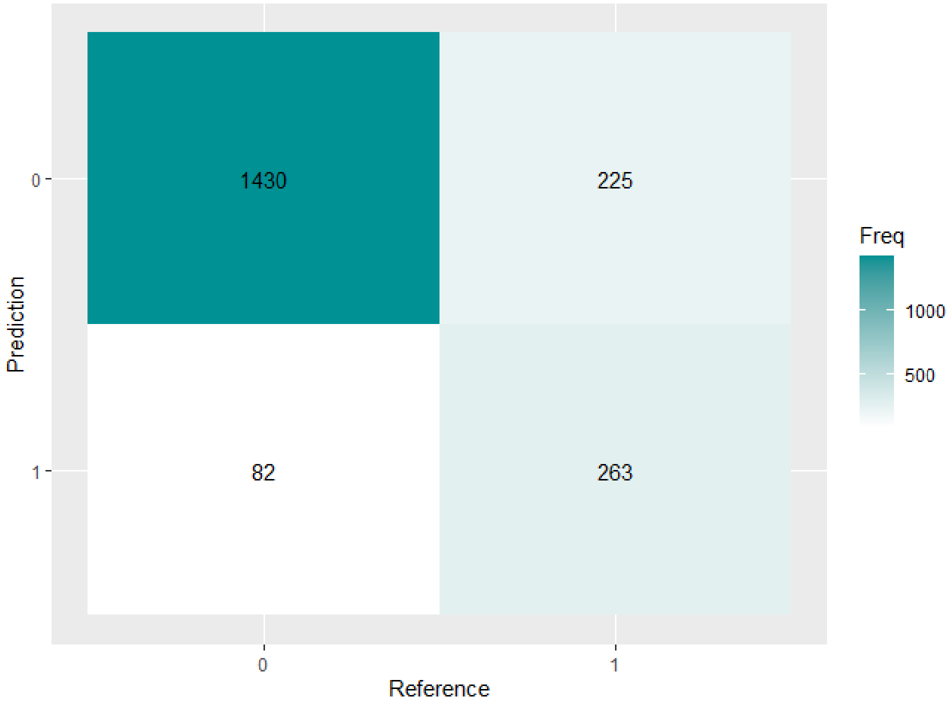
```
'data.frame':	32561 obs. of  15 variables:
 $ age           : int  39 50 38 53 28 37 49 52 31 42 ...
 $ workclass     : chr  "State-gov" "Self-emp-not-inc" "Private" "Private" ...
 $ fnlwgt        : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
 $ educatoin     : chr  "Bachelors" "Bachelors" "HS-grad" "11th" ...
 $ educatoin_num : int  13 13 9 7 13 14 5 9 14 13 ...
 $ marital_status: chr  "Never-married" "Married-civ-spouse" "Divorced" "Married-civ-spouse" ...
 $ occupation    : chr  "Adm-clerical" "Exec-managerial" "Handlers-cleaners" "Handlers-cleaners" ...
 $ relationship  : chr  "Not-in-family" "Husband" "Not-in-family" "Husband" ...
 $ race          : chr  "White" "White" "White" "Black" ...
 $ sex           : chr  "Male" "Male" "Male" "Male" ...
 $ capital_gain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
 $ capital_loss  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ hours_per_week: int  40 13 40 40 40 40 16 45 50 40 ...
 $ native_country: chr  "United-States" "United-States" "United-States" "United-States" ...
 $ income        : chr  "<=50K" "<=50K" "<=50K" "<=50K" ...
```

Results

```r
# SUPERLEARNER
tune = list(ntrees = c(50, 200),
            max_depth = 3:5,
            shrinkage = c(0.001, 0.01))

learners = create.Learner("SL.xgboost", tune=tune, detailed_names = TRUE,
                          name_prefix="xgb")

sl_lib = c("SL.xgboost", "SL.randomForest", "SL.mean", "SL.bartMachine",
           "SL.glmnet", learners$names)

SL.Adult = SuperLearner(Y=Y_train, X=X_train, family=binomial(),
                        SL.library=sl_lib, method="method.AUC",
                        verbose=TRUE)
```

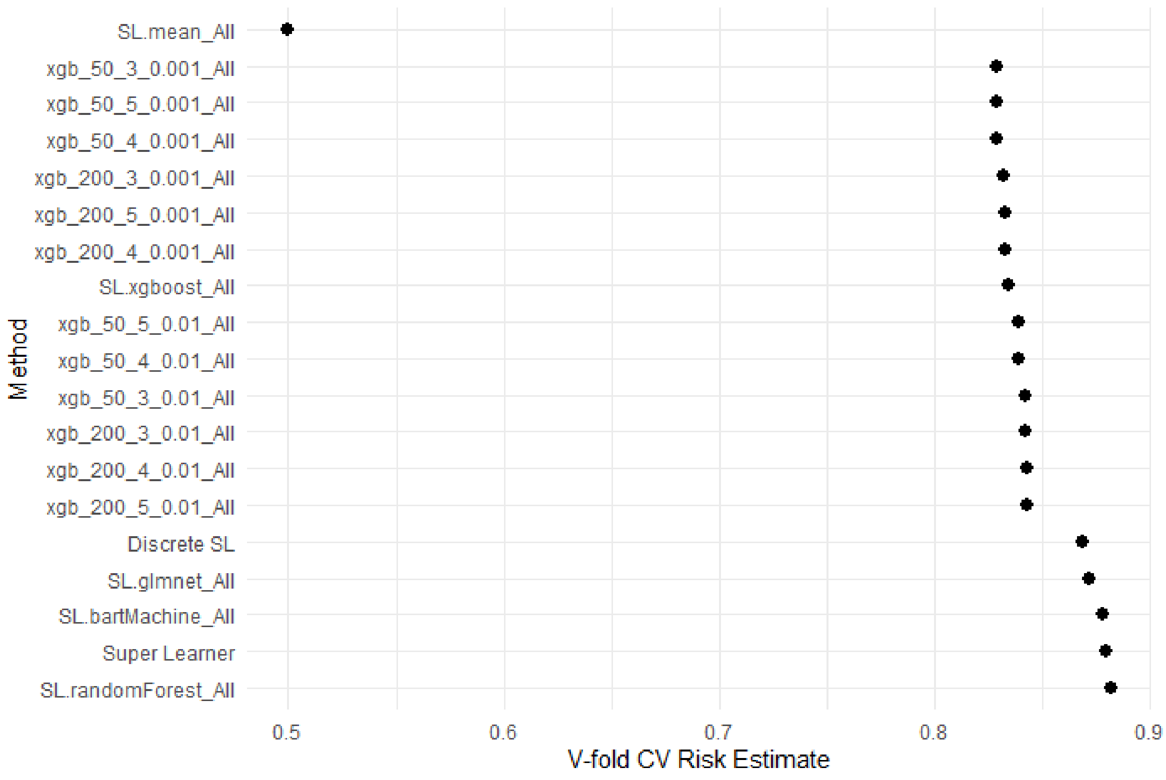
```
Call:
SuperLearner(Y = Y_train, X = X_train, family = binomial(),
SL.library = sl_lib,
    method = "method.AUC", verbose = TRUE)


                          Risk         Coef
SL.xgboost_All      0.11819582 0.0250874821
SL.randomForest_All 0.10503733 0.1554427564
SL.mean_All         0.53930502 0.0734794813
SL.bartMachine_All  0.09782539 0.1494141738
SL.glmnet_All       0.10206204 0.2691723055
xgb_50_3_0.001_All  0.17188327 0.0695545077
xgb_200_3_0.001_All 0.17100225 0.0525520325
xgb_50_4_0.001_All  0.15642931 0.0646014995
xgb_200_4_0.001_All 0.15028690 0.0262686732
xgb_50_5_0.001_All  0.15174346 0.0754267554
xgb_200_5_0.001_All 0.14690052 0.0238086477
xgb_50_3_0.01_All   0.16456037 0.0138582347
xgb_200_3_0.01_All  0.11880144 0.0010795748
xgb_50_4_0.01_All   0.14719263 0.0000000000
xgb_200_4_0.01_All  0.11815170 0.0000000000
xgb_50_5_0.01_All   0.14411106 0.0001294340
xgb_200_5_0.01_All  0.11777001 0.0001244415
```

Results




```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1430  225
         1   82  263

               Accuracy : 0.8465
                 95% CI : (0.8299, 0.862)
    No Information Rate : 0.756
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5381

 Mcnemar's Test P-Value : 5.302e-16

            Sensitivity : 0.9458
            Specificity : 0.5389
         Pos Pred Value : 0.8640
         Neg Pred Value : 0.7623
             Prevalence : 0.7560
         Detection Rate : 0.7150
   Detection Prevalence : 0.8275
      Balanced Accuracy : 0.7424

       'Positive' Class : 0
```

Results


SL.mean_All
xgb_50_3_0.001_All
xgb_50_5_0.001_All
xgb_50_4_0.001_All
xgb_200_3_0.001_All
xgb_200_5_0.001_All
xgb_200_4_0.001_All
SL.xgboost_All
xgb_50_5_0.01_All
xgb_50_4_0.01_All
xgb_50_3_0.01_All
xgb_200_3_0.01_All
xgb_200_4_0.01_All
xgb_200_5_0.01_All
Discrete SL
SL.glmnet_All
SL.bartMachine_All
Super Learner
SL.randomForest_All
Method
0.5
0.6
0.7
0.8
0.9
V-fold CV Risk Estimate

References

- ecpolley/SuperLearner: Current version of the SuperLearner R package (github.com)
- Practical considerations for specifying a super learner Rachael V. Phillips1 *, Mark J. van der Laan1 , Hana Lee2 , Susan Gruber3
- Super Learner In Prediction Eric C. Polley∗ Mark J. van der Laan†
- How to Develop Super Learner Ensembles in Python - MachineLearningMastery.com
- Guide to SuperLearner: Chris Kennedy, University of California, Berkeley
- Fundations of Statistical inference: Super Learner: Anna Gottard