Using Ensemble models in Machine learning

Myfanwy Hopkins, PhD

Data Scientist, Intel IT Business Values Pathfinding

Ensemble models

- Why Ensembles?
- Bias/variance tradeoff
- Details of Random Forest Ensemble machine learning model
- Super-learning with stacked models and K-fold cross validation
- Evaluate stacked models

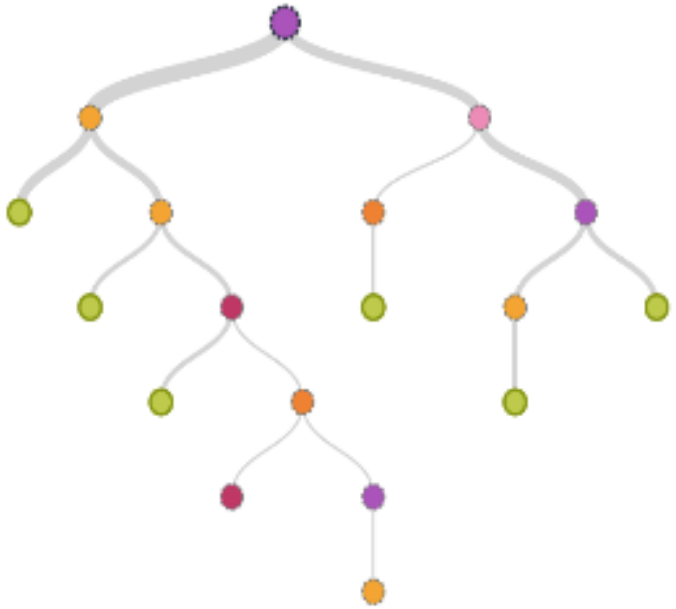
Ensemble models

"All models are wrong, but some models are useful"

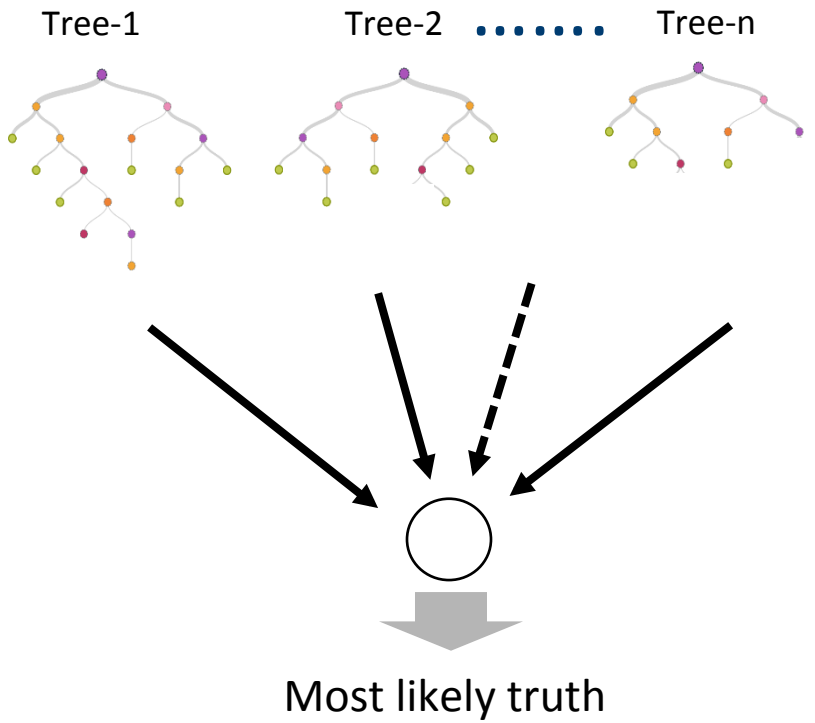
- Models built on historical data always have a limitation when applied to data in real time (prediction error rises with increased model complexity)
- Ensemble methods involve multiple models combined by averaging (regression models) of voting (classification models)
- Kaggle: winners use ensemble methods with K-fold cross validation to achieve high accuracy even on hidden test sets
- Diversity of models is Key!
  - Multiple models with high correlation may not improve accuracy significantly
  - Diverse models when combined together can help to balance Bias and Variance to find the global minimum for reduced test set error

Random Forest Ensemble Example (Level 1 ensemble)

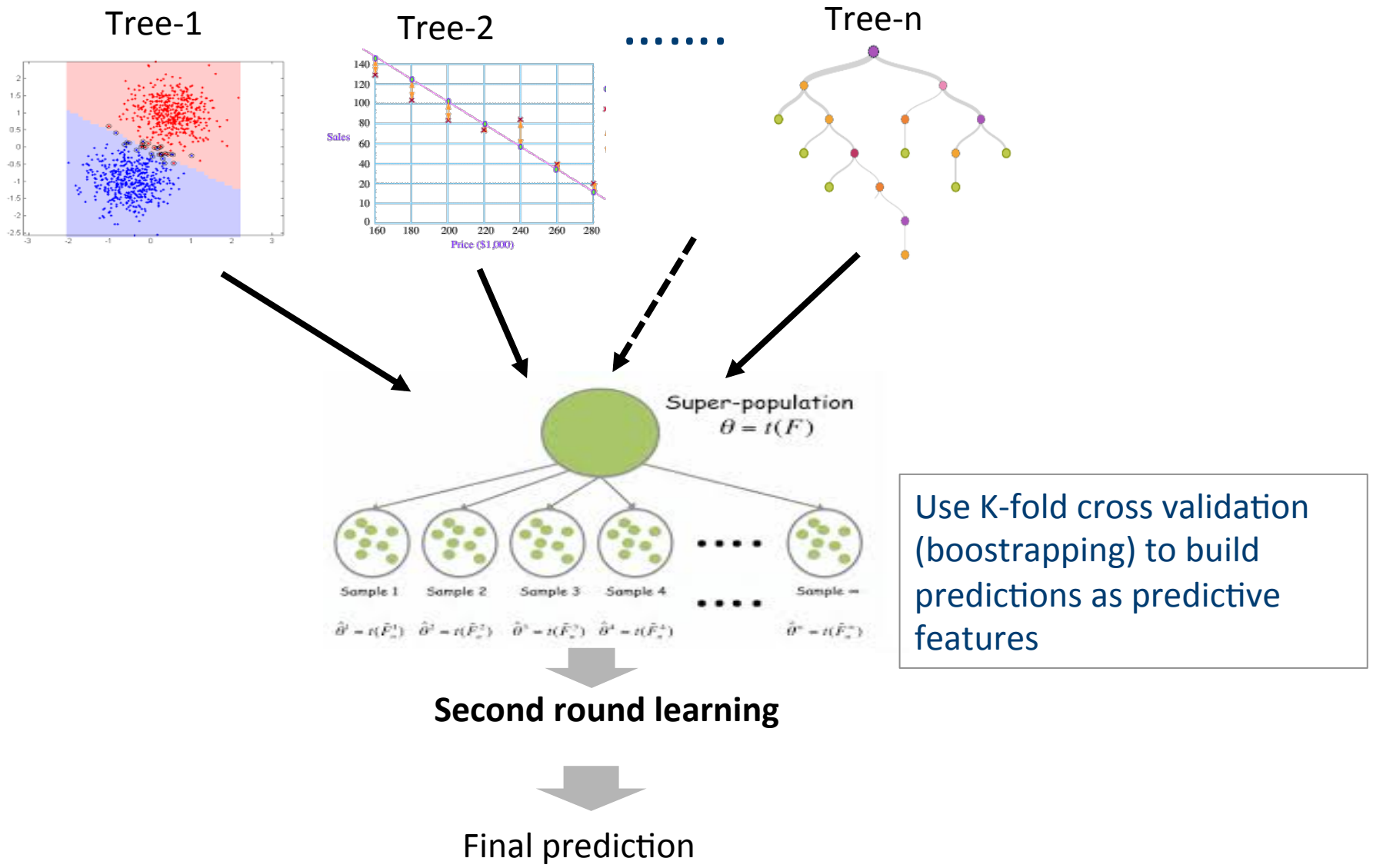
A Single Tree model
(One simple read on reality)



Random Forest
(Multiple reads on reality)

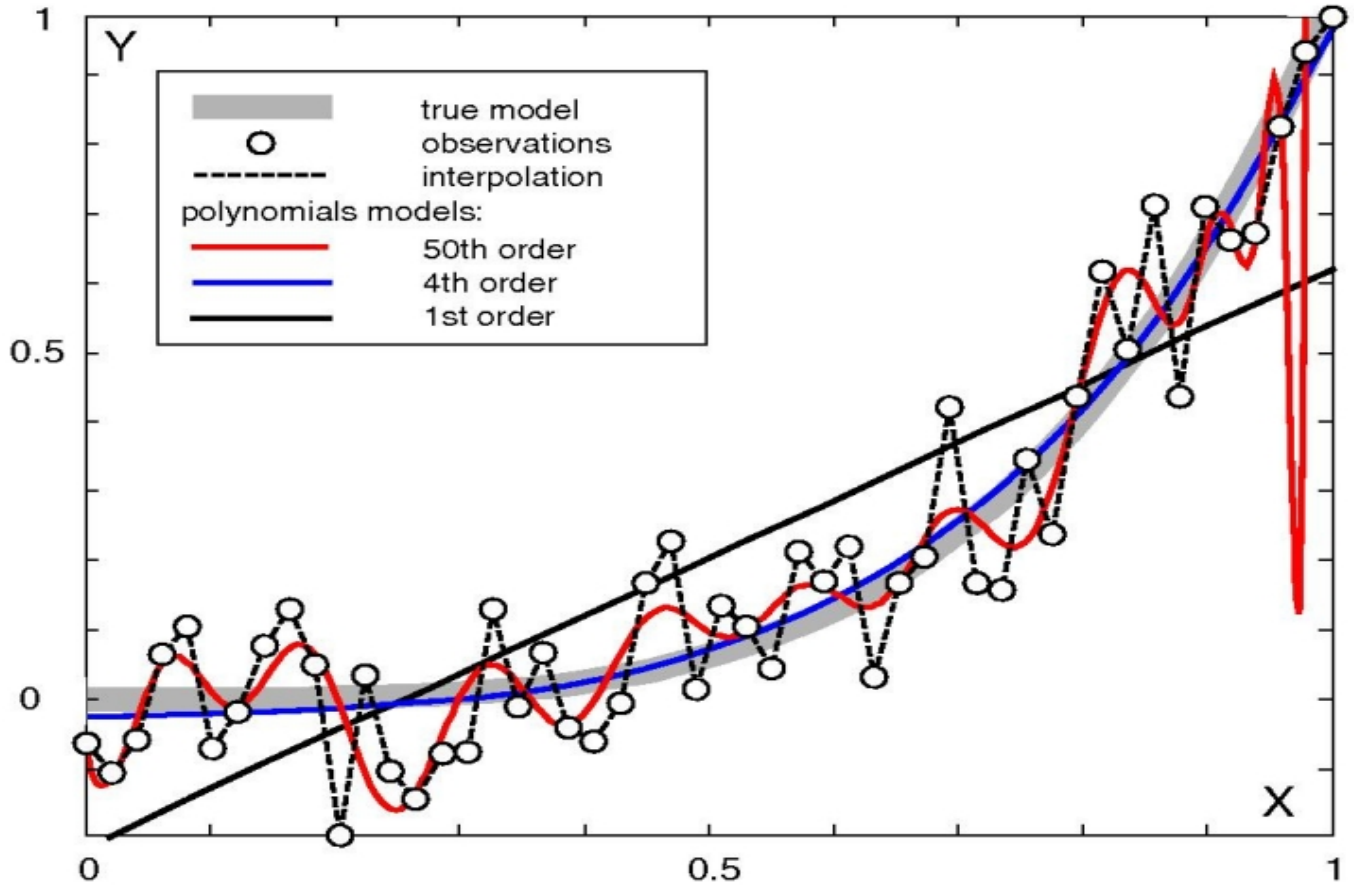


Stacked Ensemble Example (level 2 ensemble)



Use K-fold cross validation (boostrapping) to build predictions as predictive features

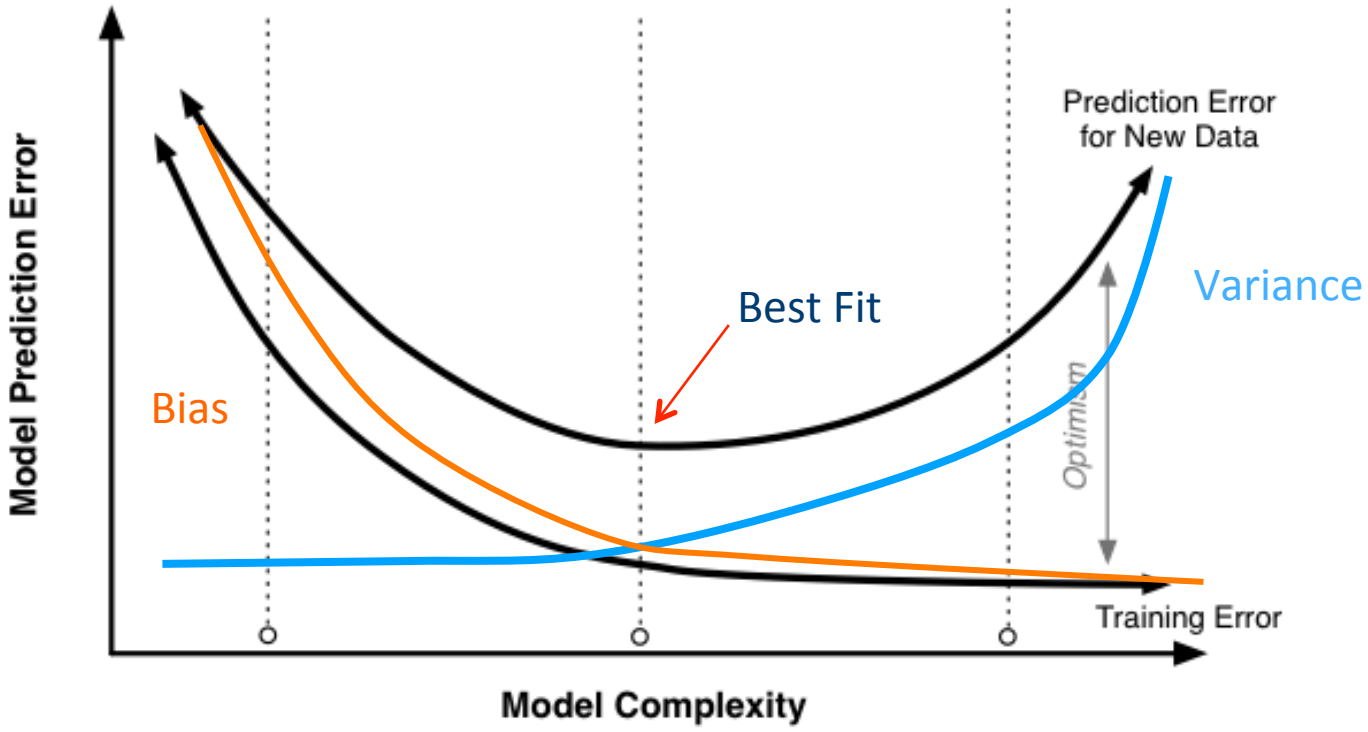
Bias vs. variance tradeoff



- Bias – The snow globe effect
  - -Under- fit
    - (1st order polynomial)
- Variance – the universe effect, or infinite possibilities
  - -Over-fit
    - (nth order polynomial)

Avoid over-parameterization, use ensemble to achieve high performance

- A main challenge with Big Data, or massive compute power is over-fitting, or models with high variance
- Some variance is good, but to prevent over-fitting, ensembling can better than simply hyper-tuning parameters to fit your training or cross validation set



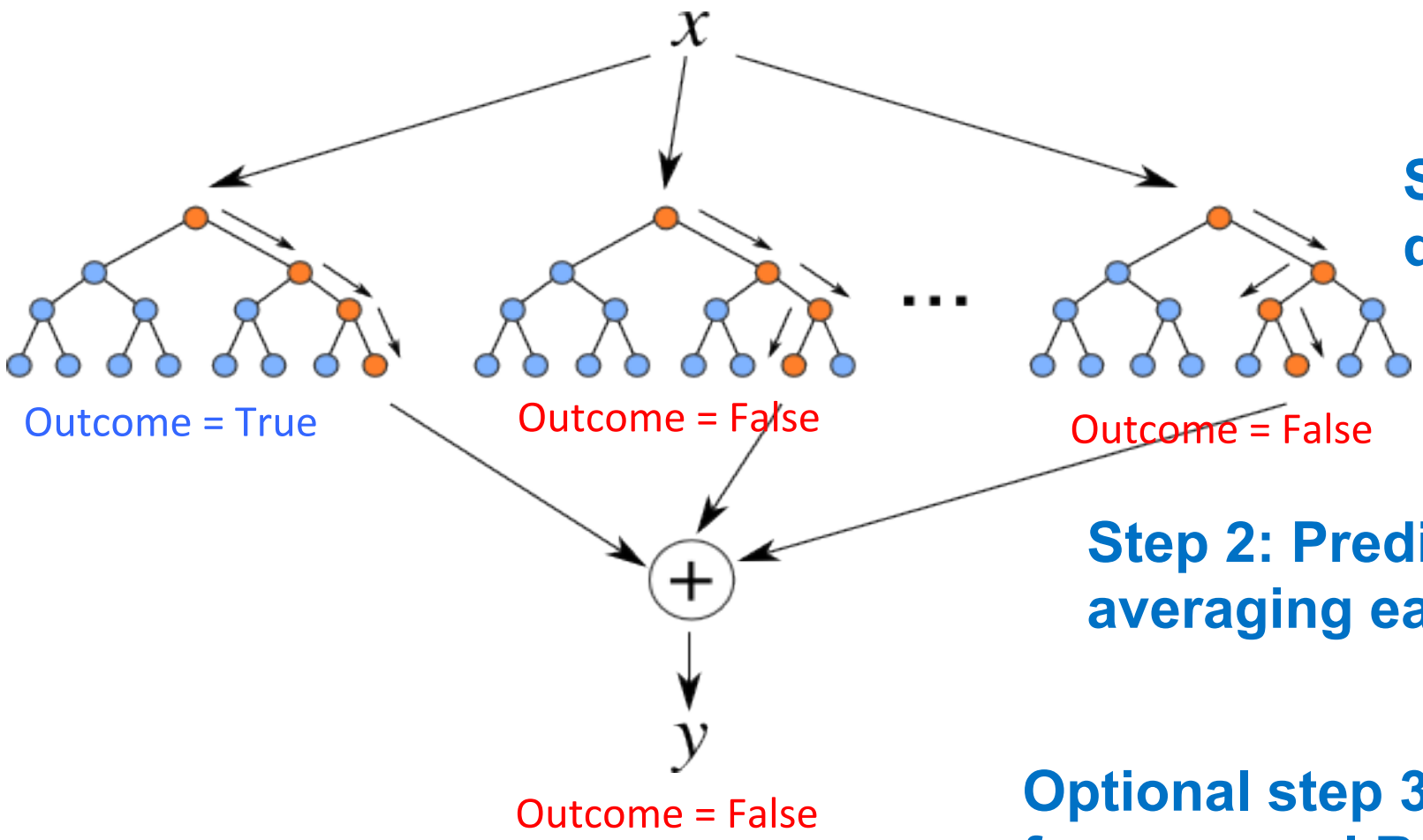
One basic model

Many basic models with some tuning voting together

One highly parameterized hyper-tuned model

Fortmann-Roe, Scott. Accurately Measuring Model Prediction Error. May 2012
http://scott.fortmann-roe.com/docs/MeasuringError.html

Random Forest Ensemble: many simple models make one powerful algorithm



Step 1: build trees with data subset

Step 2: Predict new values by averaging each tree prediction

Optional step 3: Average outcome form several Random forests for final prediction

Random Forest Ensemble uses Bootstrap aggregating

- Parameters to tune: number of variables to include, and number of trees in the forest

ID	A	B	C	D	E	F	G
2C Cl	303.676	407.138	412.847	480.138	315.512	310.89	142.147
2C Cl	0	157.809	309.137	184.062	216.139	290.478	303.665
2C Cl	0	0	1	6	4	3.72	9.201
2C Cl	163.033	226.12	53.256	17.33	15.904	5.205	1.41
2C Cl	0	0	84.012	79.504	102.438	84.188	75.215
2C Cl	60.302	55.012	40.47	72.647	0	0	0
2C Cl	990.059	783.368	1238.352	491.371	0	0	0
2C Cl	4.234	0.361	1.888	0.742	0	0	0
2C Cl	30.168	40.633	52.361	71.313	0	0	0
2C Cl	393.056	391.536	343.202	220.767	0	0	0
E740(	90.931	77.859	1.63	0.05	0	0	0
E740(	188.617	52.573	8.33	0.42	0.675	0.225	0
E740(	18.225	5.788	0.055	0	0	0	0
E740(	78.698	50.21	0.011	0.02	0	0	0
E740(	1.311	0.254	0.02	0.015	0	0	0
E740(	0.447	0.02	0.018	0	0	0	0
C2Q (	1.413	0.088	0.01	0	0	0	0
C2Q (	0.013	0.03	0	0.03	0	0	0
C2Q (	0.286	0.306	0	0	0	0	0
C2Q (	1.46	0.02	0.005	0	0	0	0
C2Q (	16.514	1.8	0.005	0	0.002	0.05	0
C2Q (	0.315	0.013	0.001	0	0	0	0
C2Q (	0.002	0	0	0	0	0	0
C2Q (	31.252	28.357	0.005	0.225	0	0	0
C2Q (	0	0.005	0	0	0	0	0

Random Forest
(multiple reads on reality)

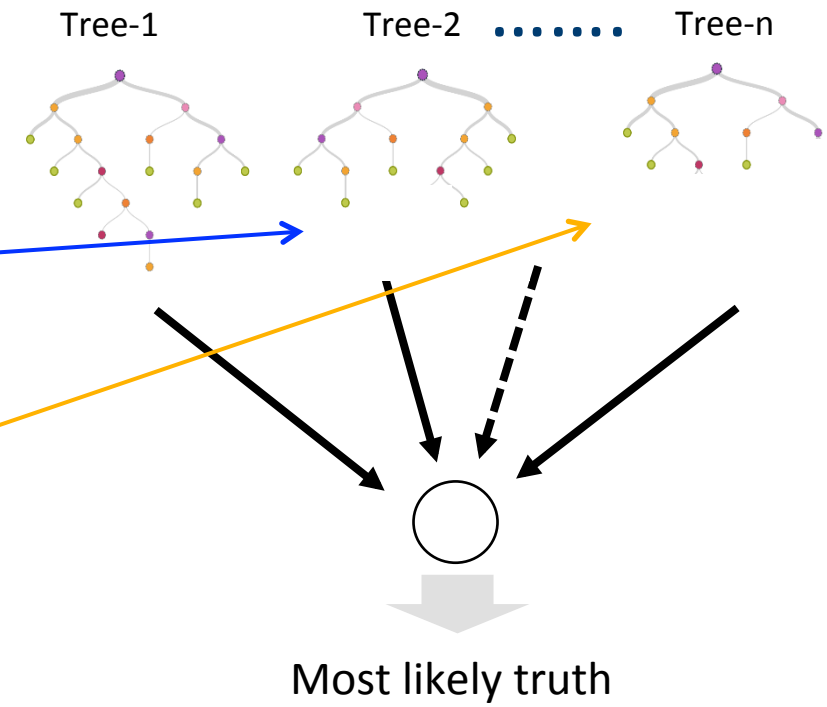
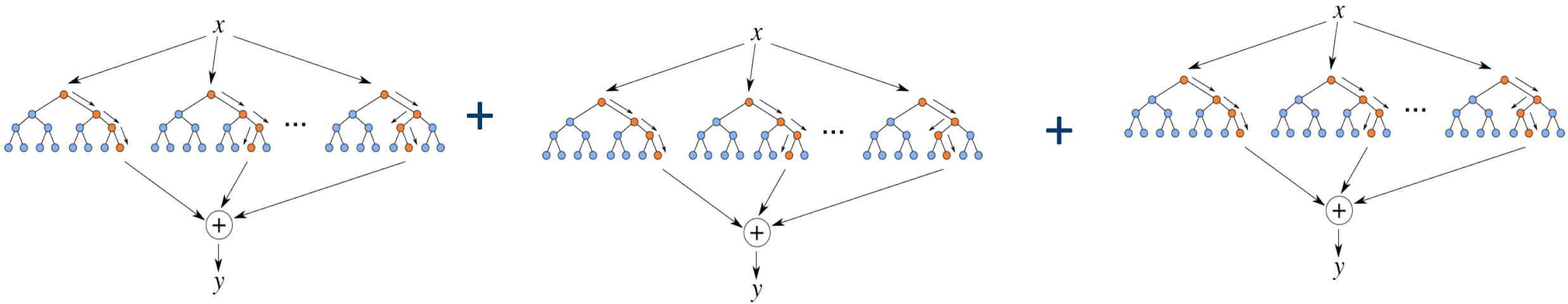


*Image for illustration. Variable selection is really non-contiguous and random

Average results form Random Forest gives a high accuracy model



library(randomForest) set.seed(1) RF1 = randomForest(Outcome ~., data=train, mtry=3,ntree=100) Pred_RF1 <- predict(RF, newdata=Test)	library(randomForest) set.seed(2) RF2 = randomForest(Outcome ~., data=train, mtry=3,ntree=100) Pred_RF2 <- predict(RF, newdata=Test)	library(randomForest) set.seed(3) RF3 = randomForest(Outcome ~., data=train, mtry=3,ntree=100) Pred_RF3 <- predict(RF, newdata=Test)

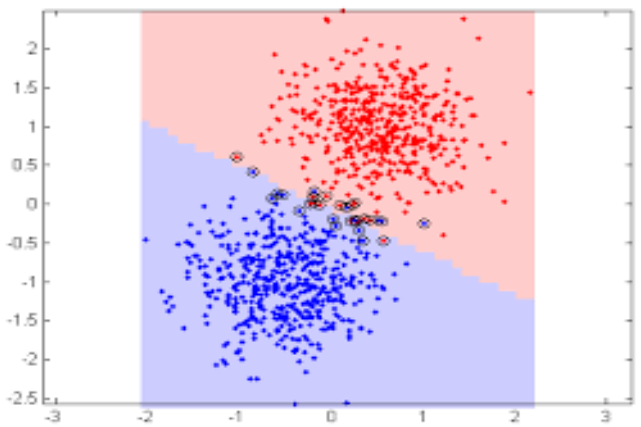
Bind predictions in data table: model_agg <- data.table(train,Pred_RF1,Pred_RF2,Pred_RF3)
Average predictions for final prediction: model_agg$ensemble<- (Pred_RF1+Pred_RF2+Pred_RF3)/3

Level 2 Ensemble: Stacked modeling and K-fold cross validation

- Stacked ensemble model is often built with random forest in combination with other model types, for even more divergent or diverse views of the data

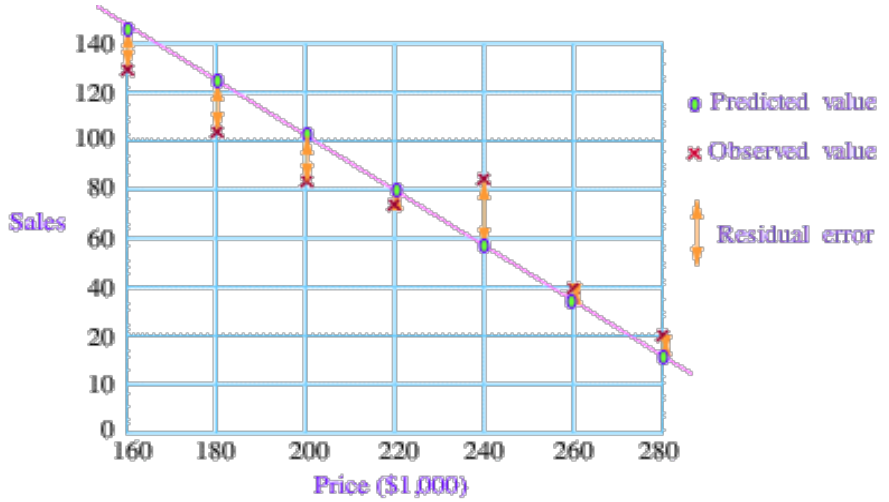
- Stacking is essential with K-fold validation sets
  - The less correlated the models, the better the overall prediction will perform on a real-life test set.
  - Helps to reduce the effects of over-parameterization, finding a perfect balance between bias and variance.

Building an orthogonal ensemble with Bootstrapping



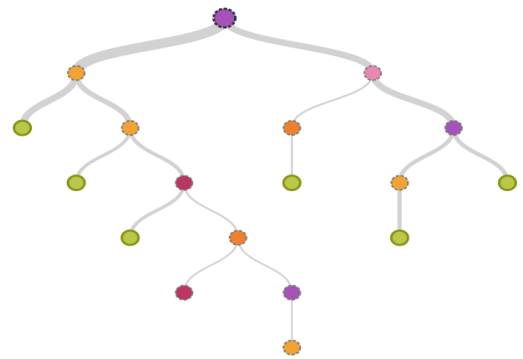
Support vector machine

+



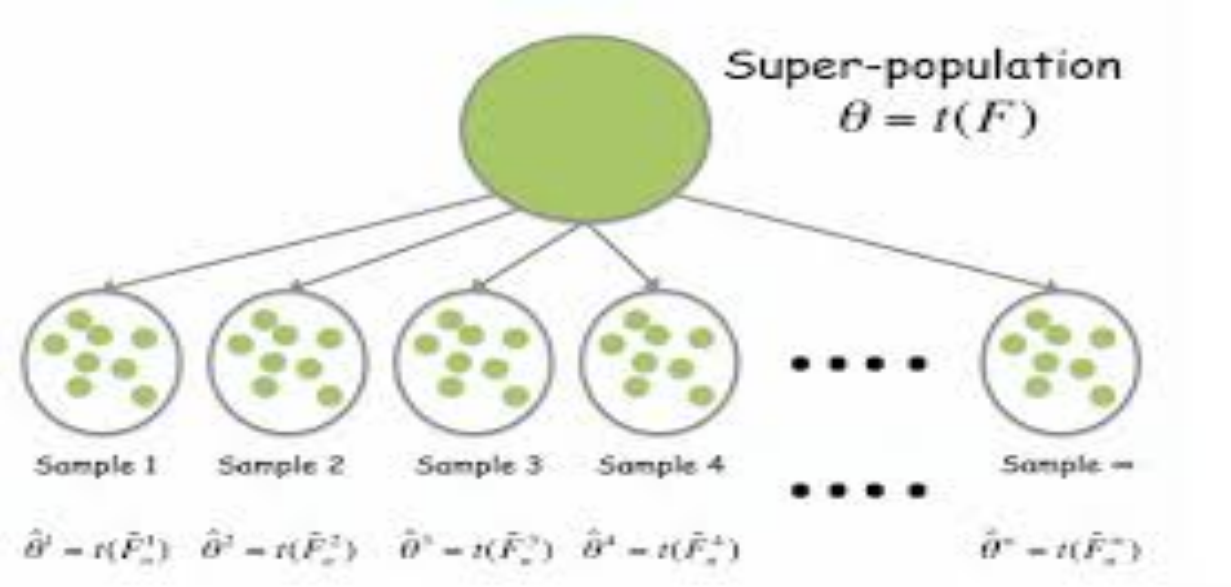
Regression with residuals

+

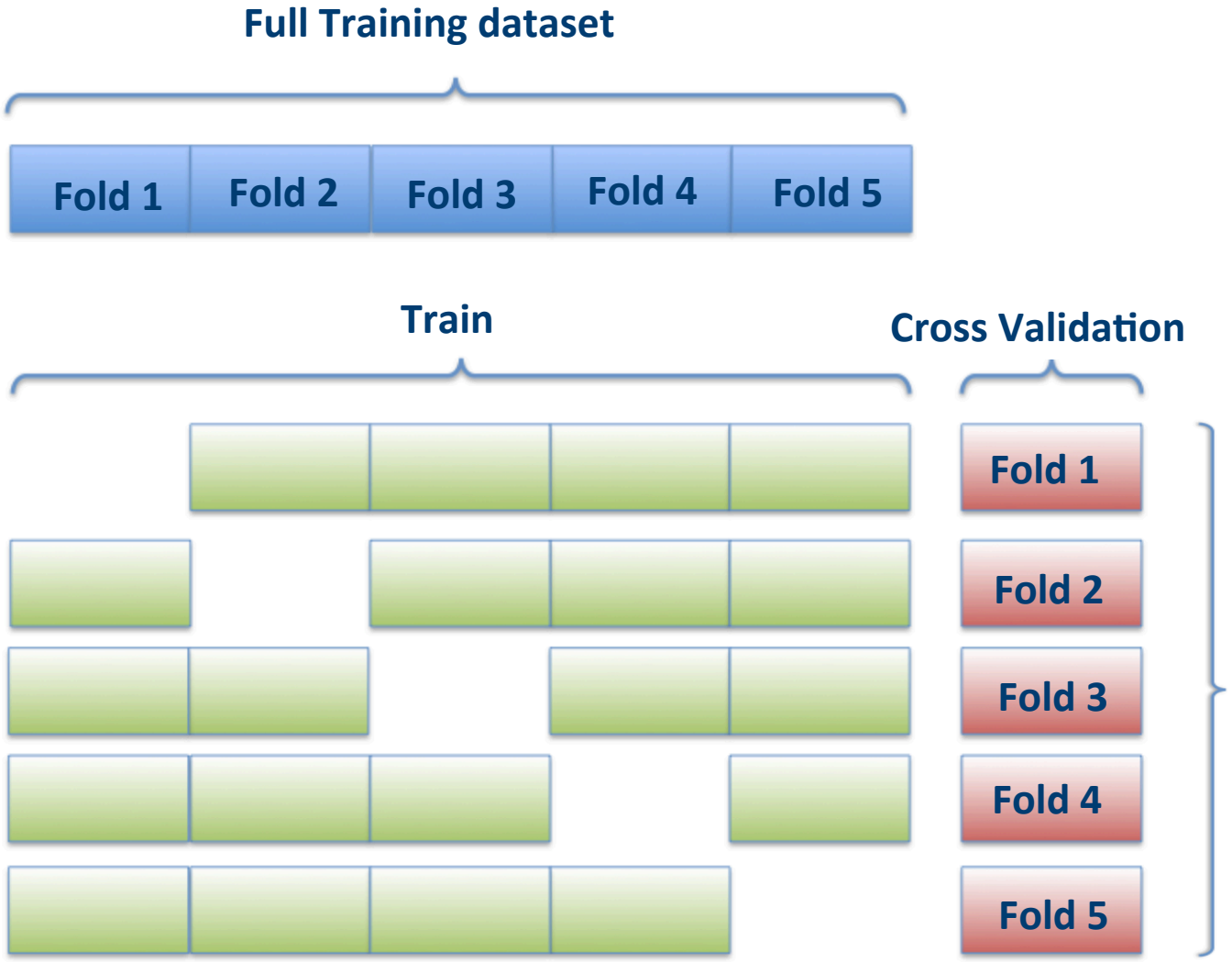


Random Forest Tree

Ensembling can be used with bootstrapping or K-fold cross validation, so each model is build with a slightly different data subset, making sure each model is unique solution to for the data



K-Fold cross validation or Bootstrapping



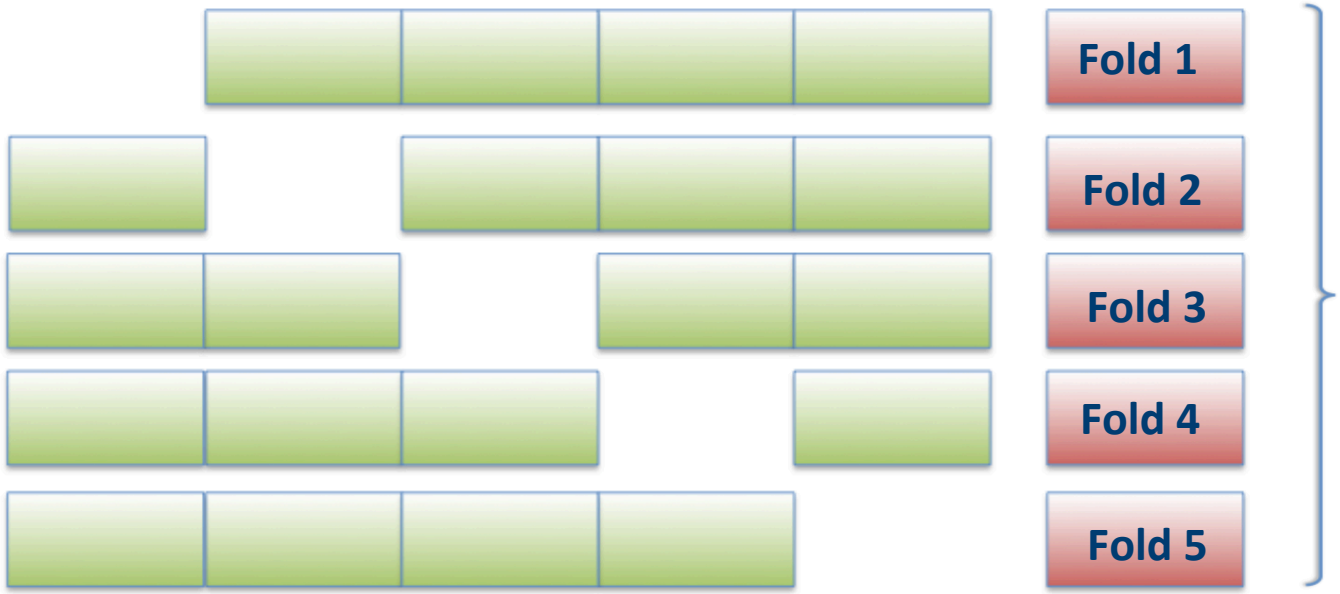
Feature 1	Feature 2	Feature 3
Model 1	Model 2	Model 3
Model 1	Model 2	Model 3
Model 1	Model 2	Model 3
Model 1	Model 2	Model 3
Model 1	Model 2	Model 3

K-Fold cross validation or Bootstrapping

Test Reporting Holdout

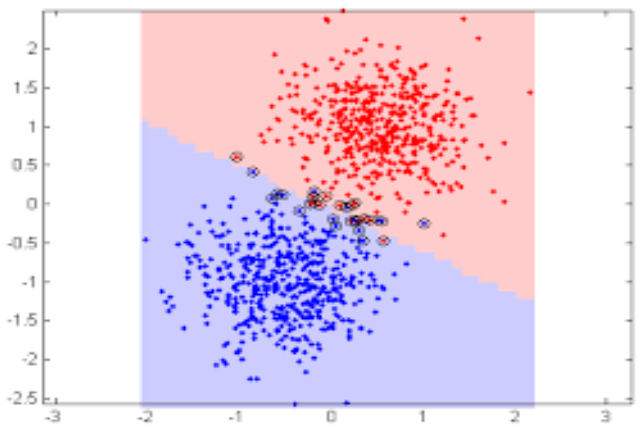
Test Holdout gives the best approximation of how model will hold up in real life

New training set

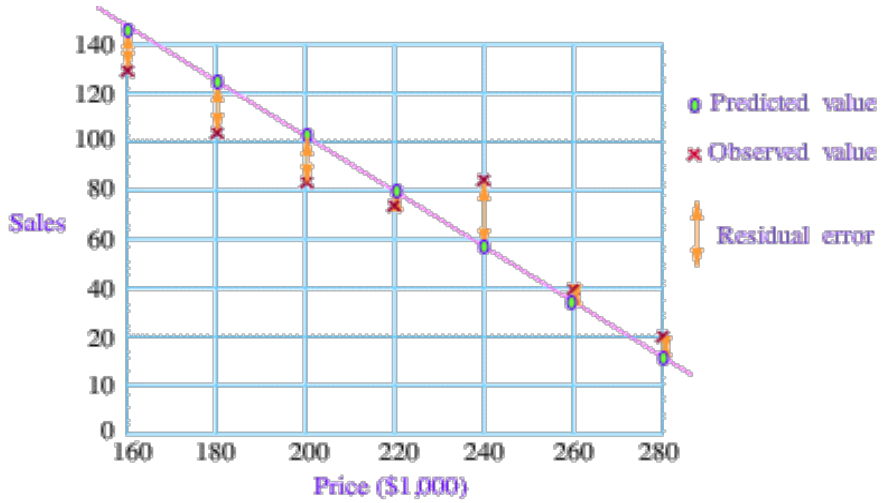


Feature 1	Feature 2	Feature 3
Model 1	Model 2	Model 3
Model 1	Model 2	Model 3
Model 1	Model 2	Model 3
Model 1	Model 2	Model 3
Model 1	Model 2	Model 3

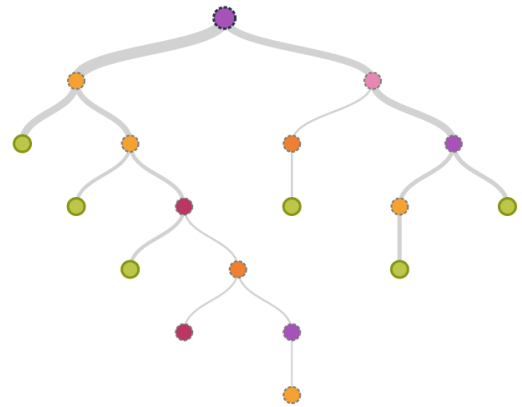
Building an orthogonal ensemble with features from ML models



+



+



Support vector machine

Regression with residuals

Random Forest Tree

```
library(e1071)
#split data into 5 fold sets, 5 train, 5 test
SVM <- svm(Outcome ~., data = train1)
Pred_SVM <- predict(SVM,
                    test1$Outcome)
```

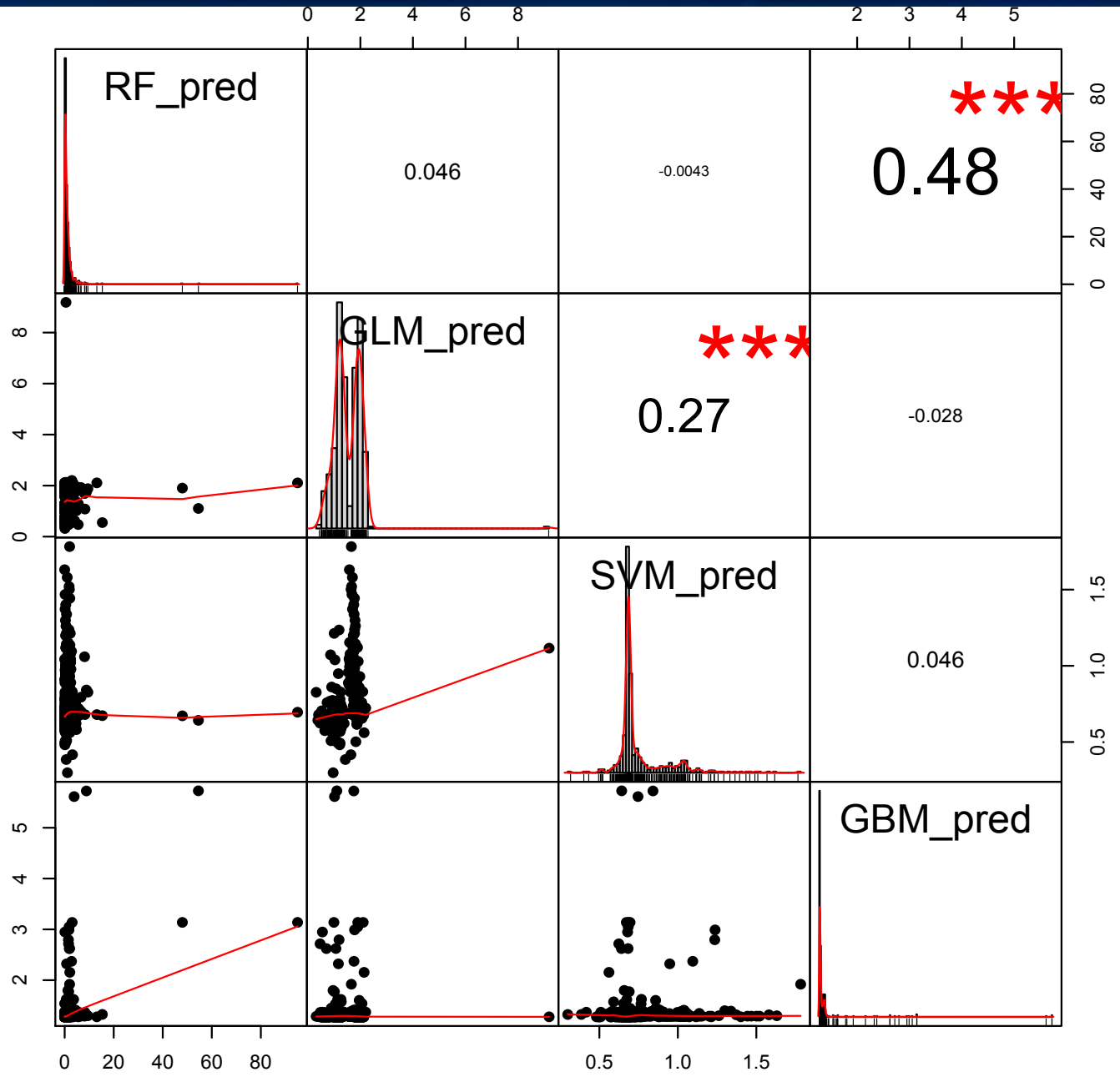
```
library(gmodels)
#use same 5 train and test sets
GLM <- glm(cash_on_cash ~.,
           data = train1)
Pred_glm <- predict(GLM, test1)
```

```
library(randomForest)
#use same 5 train and test sets
RF = randomForest(Outcome ~.,
      data=train1, mtry=3,ntree=100)
Pred_RF <- predict(RF, newdata=tes1t)
```

```
test1<-rbind(Pred_SVM,Pred_glm, Pred_RF) #repeat for test2 to test5
model_stack<- cbind(test1,test2,test3,test4,test5) #stack the crossfold sets
```

How correlated are the models? Testing with correlation plots

```
library(PerformanceAnalytics)
chart.Correlation(data, histogram=TRUE, pch=19)
```



Superlearning in R

```
install.packages("devtools")

library("devtools")

install_github("ecpolley/SuperLearner")

SL.library <- c("SL.knn", "SL.glm", "SL.randomForest") method <-
"method.NNLS" family <- "binomial”

fit <- SuperLearner(Y = Y, X = X, family = family, SL.library = SL.library,
method = method)

pred <- predict(fit, newdata = newX)
```

Summary

- Benefits
  - Can increase accuracy on the holdout test set and new data points
  - Finds balance of bias and variance
  - Conceptually understandable, i.e. not a total black box
- Limitations
  - Increases overall modeling time, many extra steps
  - Computationally expensive
  - More parameters to tune
  - May be difficult to explain all parameter choices to customers

Thank you!

Questions?

Find me at

Myfanwy.h.hopkins@intel.com

https://www.linkedin.com/in/myfanwyhopkins/

https://github.com/Myffy