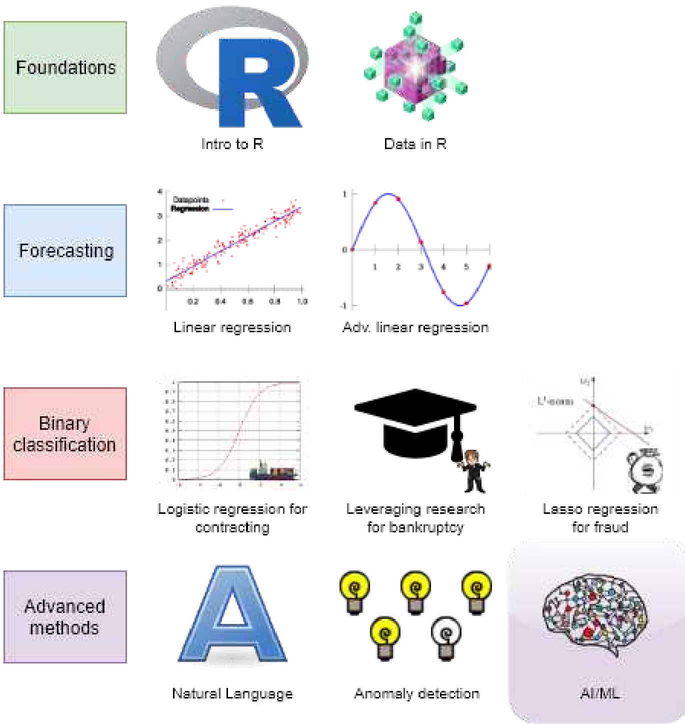
Forecasting and Forensic Analytics

Session 10: Machine Learning and AI

Dr. Wang Jiwei

Preface

Learning objectives

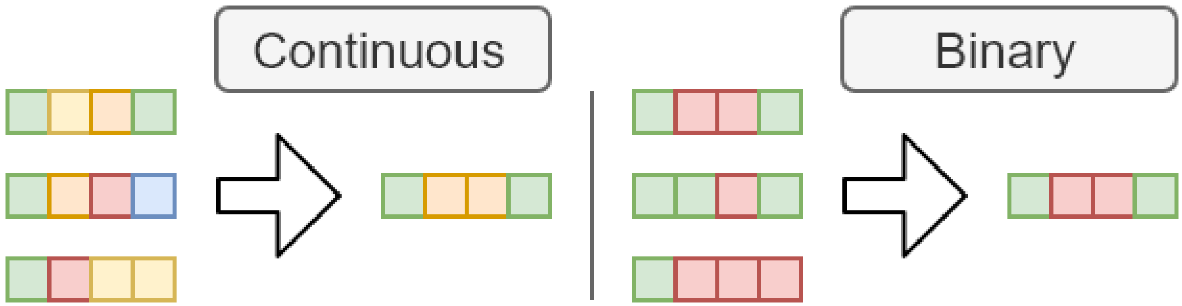


- **Theory:**
  - Ensembling
  - Ethics
  - Data security
- **Application:**
  - Loan application
  - Varied applications for ethics and data security
- **Methodology:**
  - Varied ensembling algos
  - Automated ML with H2O
- **Bonus:**
  - R interface to Python
  - TPOT AutoML

Ensembles

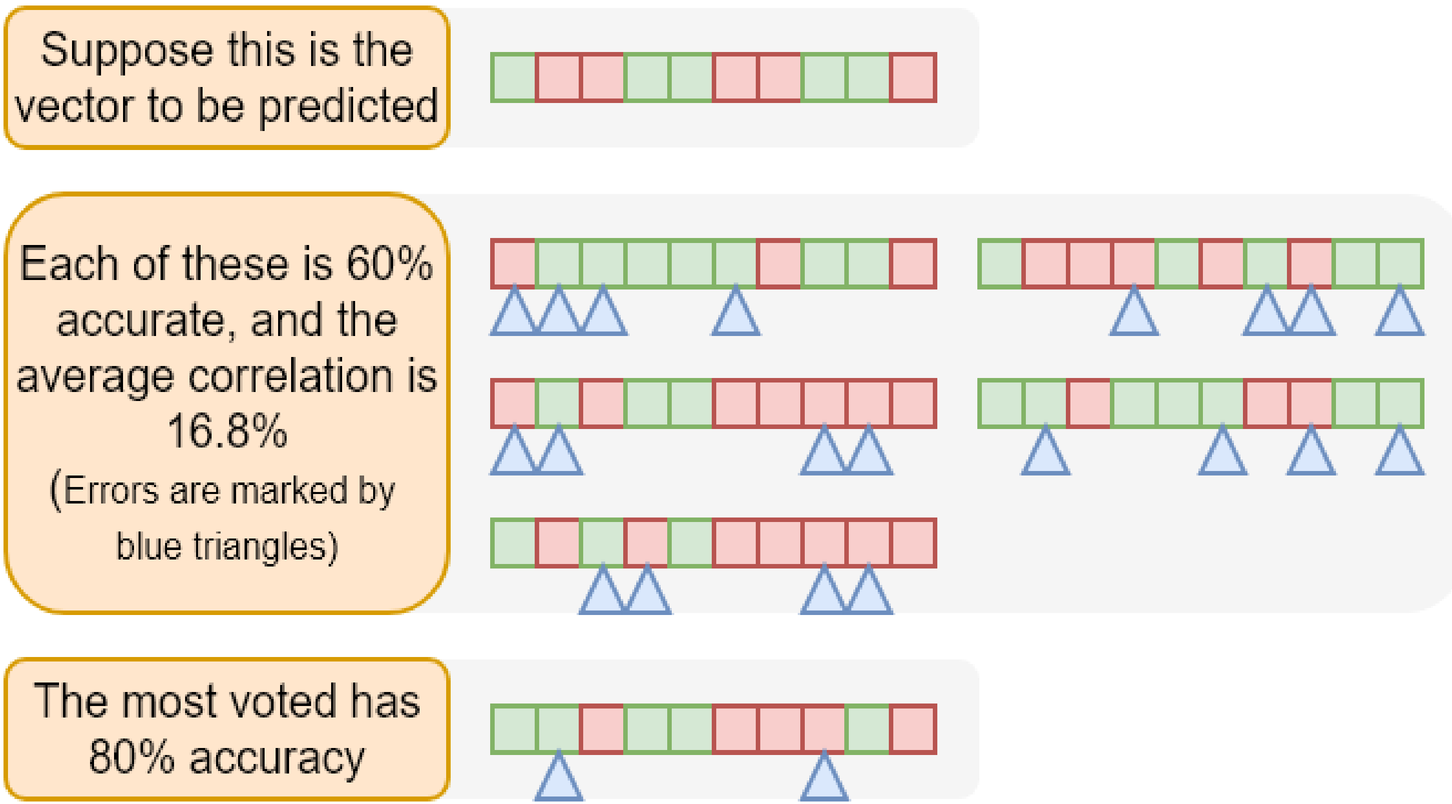
What are ensembles?

- Ensembles are models made out of models: You train multiple models using different techniques, and each seems to work well in certain cases and poorly in others
  - If you use the models in isolation, then any of them would do an OK (but not great) job
  - If you make a model using all models, you can get better performance if their strengths all shine through
- Ensembles range from simple to complex
  - Simple: a (weighted) average of a few models' predictions



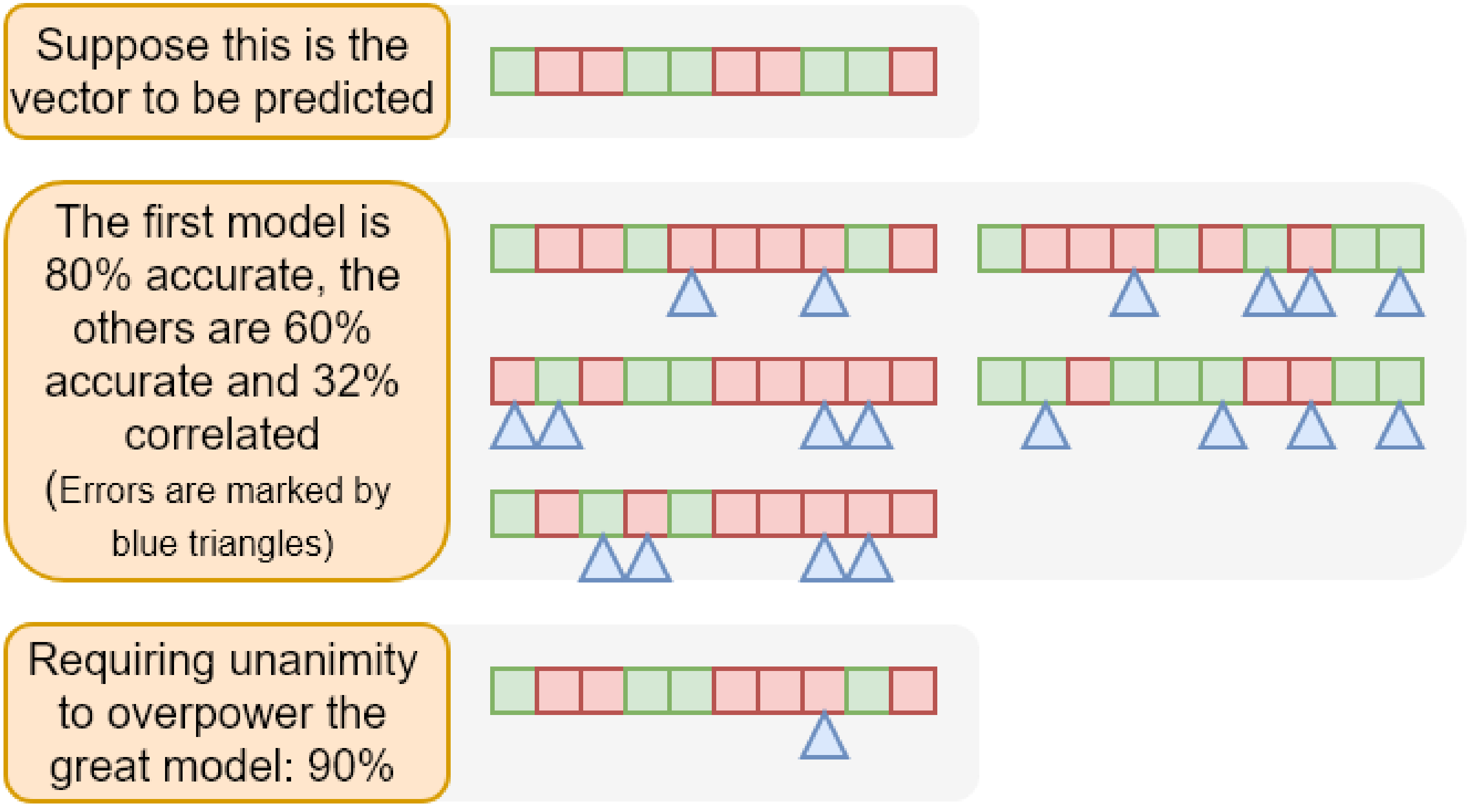
When are ensembles useful?

- (1). You have multiple models that are all decent, but none are great
  - And, ideally, the models' predictions are not highly correlated



When are ensembles useful?

- (2). You have a really good model and a bunch of mediocre models
  - And, ideally the mediocre models are not highly correlated

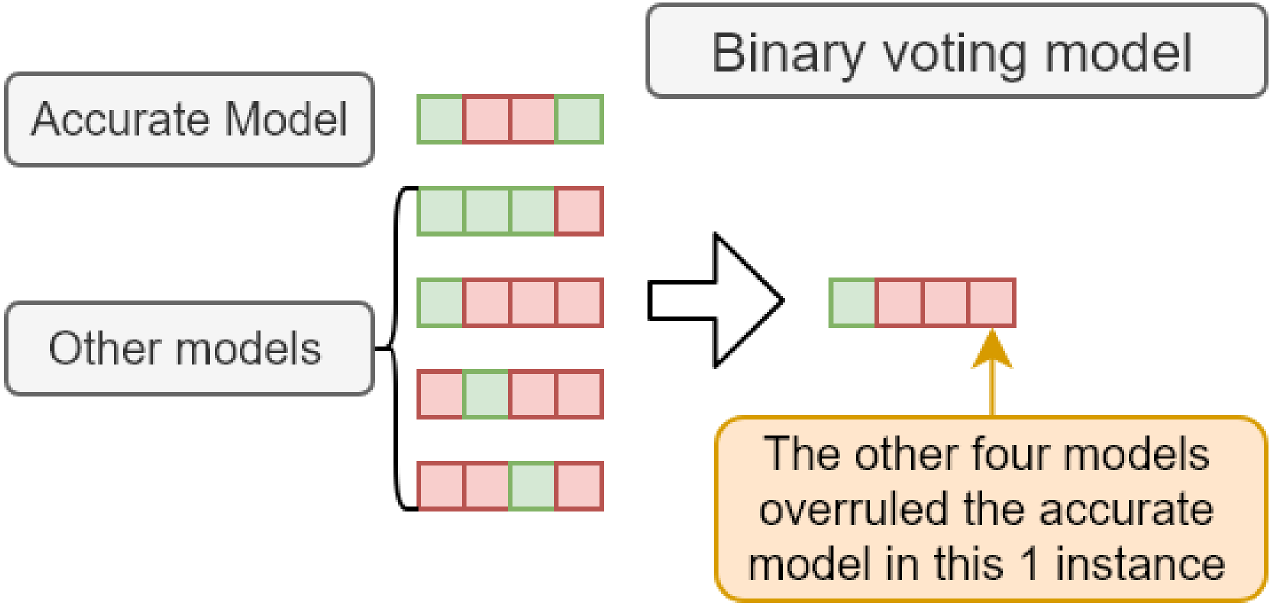


When are ensembles useful?

- (3). You really need to get just a bit more accuracy/less error out of the model, and you have some other models lying around
- (4). We want a more stable model
  - It helps to stabilize predictions by limiting the effect that errors or outliers produced by any 1 model can have on our prediction
  - Think: Diversification (like in finance)

A simple ensemble

- Simple averaging is the easiest
- Weighted averaging: You may want to weight the best model a bit higher
- Voting: vote by the majority rule



Simple ensemble in R

- Classification And Regression Training `package:caret`: training and plotting 238 models
- Prepare data: `preProcess()` and `predict()`
- Partition data: `createDataPartition()`
- Training parameters: `trainControl()`
- Running model: `train()`

```
library(caret); set.seed(123)
data <- read.csv('../../Data/Session_10_ensemble.csv')
str(data)
```

```
## 'data.frame':    614 obs. of  13 variables:
##  $ Loan_ID          : chr  "LP001002" "LP001003" "LP001005" "LP001006" ...
##  $ Gender           : chr  "Male" "Male" "Male" "Male" ...
##  $ Married          : chr  "No" "Yes" "Yes" "Yes" ...
##  $ Dependents       : chr  "0" "1" "0" "0" ...
##  $ Education        : chr  "Graduate" "Graduate" "Graduate" "Not Graduate" ...
##  $ Self_Employed    : chr  "No" "No" "Yes" "No" ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
##  $ Property_Area    : chr  "Urban" "Rural" "Urban" "Urban" ...
##  $ Loan_Status      : chr  "Y" "N" "Y" "Y" ...
```

Training control for base models

```
# Imputing missing values using median and normalize the data
# by substracting the mean and divided by standard deviation
preProcValues <- preProcess(data, method = c("medianImpute","center","scale"))
data_processed <- predict(preProcValues, data)
sum(is.na(data_processed))
```

```
## [1] 0
```

```
#Spliting dataset into train and test based on outcome: 75% and 25%
#List=FALSE returns an integer matrix which could be used for index
#Stratified sampling: randomly draw p% from each group of Loan_Status)
index <- createDataPartition(data_processed$Loan_Status, p=0.75, list=FALSE)
trainSet <- data_processed[ index, ]
testSet  <- data_processed[-index, ]

#Defining the training controls for multiple models
fitControl <- trainControl(
  method = "cv", # cross-validation
  number = 5, # k-fold
  savePredictions = 'final', # save the predictions for the optimal parameters
  classProbs = T) # probabilities be computed for classification models

#Defining the predictors and outcome
predictors <- c("Credit_History", "LoanAmount", "Loan_Amount_Term",
                "ApplicantIncome", "CoapplicantIncome")
outcomeName <- "Loan_Status"
```

Base model 1: Random Forest

```r
# Training the random forest model
# The tuneLength specifies default values for the main parameter
# the main para for RF is the number of random features
model_rf <- train(trainSet[, predictors], trainSet[, outcomeName], method='rf',
                  trControl = fitControl, tuneLength = 3)

#Predicting using random forest model
testSet$pred_rf <- predict(object = model_rf, testSet[, predictors])

#Checking the accuracy of the random forest model
confusionMatrix(factor(testSet$Loan_Status), testSet$pred_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 26 22
##          Y  9 96
##
##                Accuracy : 0.7974
##                  95% CI : (0.7249, 0.858)
##     No Information Rate : 0.7712
##     P-Value [Acc > NIR] : 0.25338
##
##                   Kappa : 0.4921
##
##  Mcnemar's Test P-Value : 0.03114
##
##             Sensitivity : 0.7429
##             Specificity : 0.8136
```

Base model 2: kNN

```r
# Training the knn model
# the main para for kNN is k
model_knn <- train(trainSet[, predictors], trainSet[, outcomeName],
                   method = 'knn', trControl = fitControl, tuneLength = 3)

#Predicting using knn model
testSet$pred_knn <- predict(object = model_knn, testSet[, predictors])

#Checking the accuracy of the knn model
confusionMatrix(factor(testSet$Loan_Status), testSet$pred_knn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  24  24
##          Y   4 101
##
##                Accuracy : 0.817
##                  95% CI : (0.7465, 0.8748)
##     No Information Rate : 0.817
##     P-Value [Acc > NIR] : 0.5502895
##
##                   Kappa : 0.5208
##
##  Mcnemar's Test P-Value : 0.0003298
##
##             Sensitivity : 0.8571
##             Specificity : 0.8080
##          Pos Pred Value : 0.5000
```

Base model 3: Logit

```r
# Training the Logistic regression model
# The tuneLength is no applicable for logit
model_lr <- train(trainSet[, predictors], trainSet[, outcomeName],
                  method = 'glm', trControl = fitControl)

#Predicting using logit model
testSet$pred_lr <- predict(object = model_lr, testSet[, predictors])

#Checking the accuracy of the logit model
confusionMatrix(factor(testSet$Loan_Status), testSet$pred_lr)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  25  23
##          Y   2 103
##
##                Accuracy : 0.8366
##                  95% CI : (0.7683, 0.8914)
##     No Information Rate : 0.8235
##     P-Value [Acc > NIR] : 0.3832
##
##                   Kappa : 0.5694
##
##  Mcnemar's Test P-Value : 6.334e-05
##
##             Sensitivity : 0.9259
##             Specificity : 0.8175
##          Pos Pred Value : 0.5208
```

Ensemble by average

```
#Predicting the probabilities
testSet$pred_rf_prob <- predict(object = model_rf, testSet[, predictors],
                                type = 'prob')
testSet$pred_knn_prob <- predict(object = model_knn, testSet[, predictors],
                                 type = 'prob')
testSet$pred_lr_prob <- predict(object = model_lr, testSet[, predictors],
                                type = 'prob')

#Taking average of predictions
testSet$pred_avg <- (testSet$pred_rf_prob$Y + testSet$pred_knn_prob$Y +
                     testSet$pred_lr_prob$Y) / 3

#Splitting into binary classes at 0.5
testSet$pred_avg <- as.factor(ifelse(testSet$pred_avg > 0.5, 'Y', 'N'))

confusionMatrix(factor(testSet$Loan_Status), testSet$pred_avg)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    N   Y
##          N  25  23
##          Y   2 103
##
##                Accuracy : 0.8366
##                  95% CI : (0.7683, 0.8914)
##     No Information Rate : 0.8235
##     P-Value [Acc > NIR] : 0.3832
##
##                   Kappa : 0.5694
```

Ensemble by weighted average

```r
#Taking weighted average of predictions
testSet$pred_weighted_avg <-
  (testSet$pred_rf_prob$Y * 0.25) + (testSet$pred_knn_prob$Y * 0.25) +
  (testSet$pred_lr_prob$Y * 0.5)

#Splitting into binary classes at 0.5
testSet$pred_weighted_avg <-
  as.factor(ifelse(testSet$pred_weighted_avg > 0.5, 'Y', 'N'))

confusionMatrix(factor(testSet$Loan_Status), testSet$pred_weighted_avg)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  25  23
##          Y   2 103
##
##                Accuracy : 0.8366
##                  95% CI : (0.7683, 0.8914)
##     No Information Rate : 0.8235
##     P-Value [Acc > NIR] : 0.3832
##
##                   Kappa : 0.5694
##
##  Mcnemar's Test P-Value : 6.334e-05
##
##             Sensitivity : 0.9259
##             Specificity : 0.8175
##          Pos Pred Value : 0.5208
```

Ensemble by voting

```r
#The majority vote
testSet$pred_majority <-
  as.factor(ifelse(testSet$pred_rf == 'Y' & testSet$pred_knn== 'Y', 'Y',
  ifelse(testSet$pred_rf == 'Y' & testSet$pred_lr == 'Y', 'Y',
  ifelse(testSet$pred_knn =='Y' & testSet$pred_lr == 'Y', 'Y', 'N'))))

confusionMatrix(factor(testSet$Loan_Status), testSet$pred_majority)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  25  23
##          Y   2 103
##
##                Accuracy : 0.8366
##                  95% CI : (0.7683, 0.8914)
##     No Information Rate : 0.8235
##     P-Value [Acc > NIR] : 0.3832
##
##                   Kappa : 0.5694
##
##  Mcnemar's Test P-Value : 6.334e-05
##
##             Sensitivity : 0.9259
##             Specificity : 0.8175
##          Pos Pred Value : 0.5208
##          Neg Pred Value : 0.9810
##              Prevalence : 0.1765
```

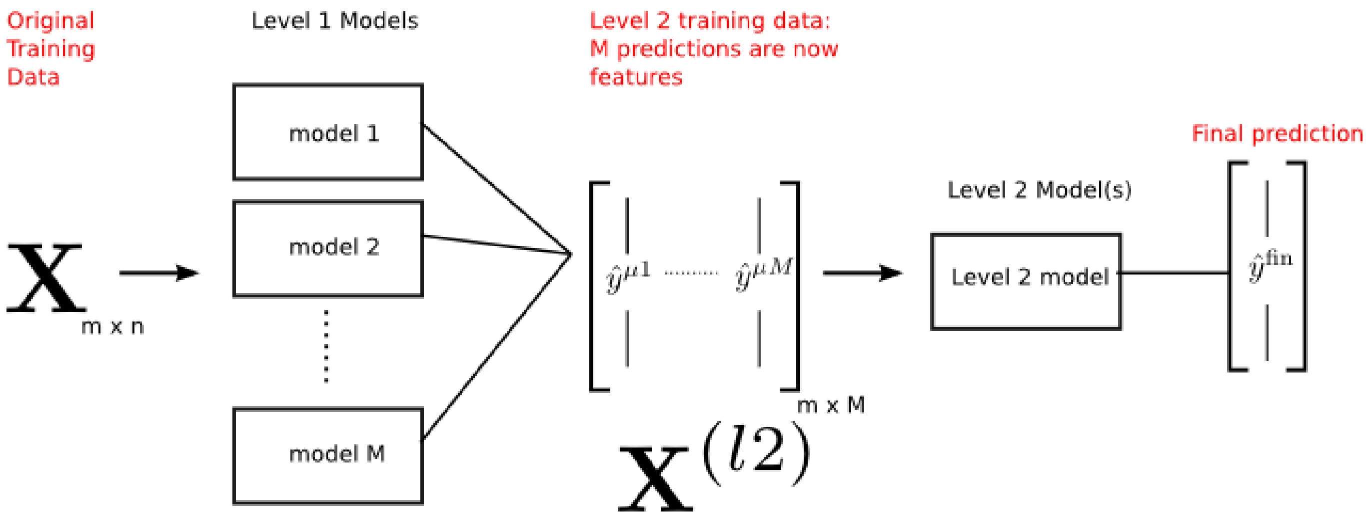
Why ensembling not improved?

```r
#Correlation matrix for the 3 base model predictions
testSet$pred_rf1 <- as.numeric(ifelse(testSet$pred_rf == 'Y', 1, 0))
testSet$pred_knn1 <- as.numeric(ifelse(testSet$pred_knn == 'Y', 1, 0))
testSet$pred_lr1 <- as.numeric(ifelse(testSet$pred_lr == 'Y', 1, 0))
cor(testSet[ , c("pred_rf1", "pred_knn1", "pred_lr1")])
```

```
##            pred_rf1 pred_knn1  pred_lr1
## pred_rf1  1.0000000 0.7885379 0.8499700
## pred_knn1 0.7885379 1.0000000 0.9337366
## pred_lr1  0.8499700 0.9337366 1.0000000
```

A more complex ensemble

- In stead of using simple functions such as average and voting, we can train a new model on the predictions from all the other models
  - The new model could be a new algo or the same algo as the other models
  - We can also build multi layers of models and predict based on predictions from models in the lower layer models.
  - It is called stacking or blending models (typically 2 layers)
  - Read here for more on stacking in R



A more complex ensemble

```r
#Get the predicted outcome for training data
trainSet$pred_rf_prob <- model_rf$pred$Y[order(model_rf$pred$rowIndex)]
trainSet$pred_knn_prob <- model_knn$pred$Y[order(model_knn$pred$rowIndex)]
trainSet$pred_lr_prob <- model_lr$pred$Y[order(model_lr$pred$rowIndex)]

#Predicting probabilities for the test data
testSet$pred_rf_prob <- predict(model_rf, testSet[, predictors], type='prob')$Y
testSet$pred_knn_prob <- predict(model_knn, testSet[, predictors], type='prob')$Y
testSet$pred_lr_prob <- predict(model_lr, testSet[, predictors], type='prob')$Y
```

Name	Type	Value
model_rf	list [20] (S3: train)	List of length 20
method	character [1]	'rf'
modelInfo	list [15]	List of length 15
modelType	character [1]	'Classification'
results	list [3 x 5] (S3: data.frame)	A data.frame with 3 rows and 5 columns
pred	list [461 x 7] (S3: data.frame)	A data.frame with 461 rows and 7 columns
mtry	double [461]	2 2 2 2 2 2 ...
pred	factor	Factor with 2 levels: "N", "Y"
obs	factor	Factor with 2 levels: "N", "Y"
N	double [461]	0.846 0.456 0.446 0.042 0.328 0.082 ...
Y	double [461]	0.154 0.544 0.554 0.958 0.672 0.918 ...
rowIndex	integer [461]	314 317 361 362 330 359 ...
Resample	character [461]	'Fold1' 'Fold1' 'Fold1' 'Fold1' 'Fold1' 'Fold1' ...

A more complex ensemble

```
#Predictors for the top layer model
predictors_top<-c('pred_rf_prob', 'pred_knn_prob', 'pred_lr_prob')

#Stochastic Gradient Boosting Model (GBM) as the top layer model
model_gbm <- train(trainSet[, predictors_top], trainSet[, outcomeName],
                   method='gbm', trControl = fitControl)
```

A more complex ensemble

```
#Predicting using GBM model
testSet$pred_gbm_ens <- predict(model_gbm, testSet[, predictors_top])

#Checking the accuracy of the GBM model
confusionMatrix(factor(testSet$Loan_Status), testSet$pred_gbm_ens)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  26  22
##          Y   2 103
##
##                Accuracy : 0.8431
##                  95% CI : (0.7757, 0.8968)
##     No Information Rate : 0.817
##     P-Value [Acc > NIR] : 0.2353527
##
##                   Kappa : 0.5893
##
##  Mcnemar's Test P-Value : 0.0001052
##
##             Sensitivity : 0.9286
##             Specificity : 0.8240
##          Pos Pred Value : 0.5417
##          Neg Pred Value : 0.9810
##              Prevalence : 0.1830
##          Detection Rate : 0.1699
##    Detection Prevalence : 0.3137
```

How about our fraud detection models?

- Seven models from Session 7:
  - pred_F: fit_2011
  - pred_S: fit_2000s
  - pred_FS: fit_2000f
  - pred_BCE: fit_BCE
  - pred_lmin: fit_lasso
  - pred_l1se: fit_lasso
  - pred_xgb: fit4 (XGBoost)
- Ensemble the seven models by XGBoost

```
df <- readRDS('../../Data/Session_10_models.rds')
head(df) %>% select(-pred_F, -pred_S) %>% slice(1:2) %>% html_df()
```

Test	AAER	pred_FS	pred_BCE	pred_lmin	pred_l1se	pred_xgb
0	0	0.0395418	0.0661011	0.0301550	0.0296152	0.0478672
0	0	0.0173693	0.0344585	0.0328011	0.0309861	0.0616048

How about our fraud detection models?

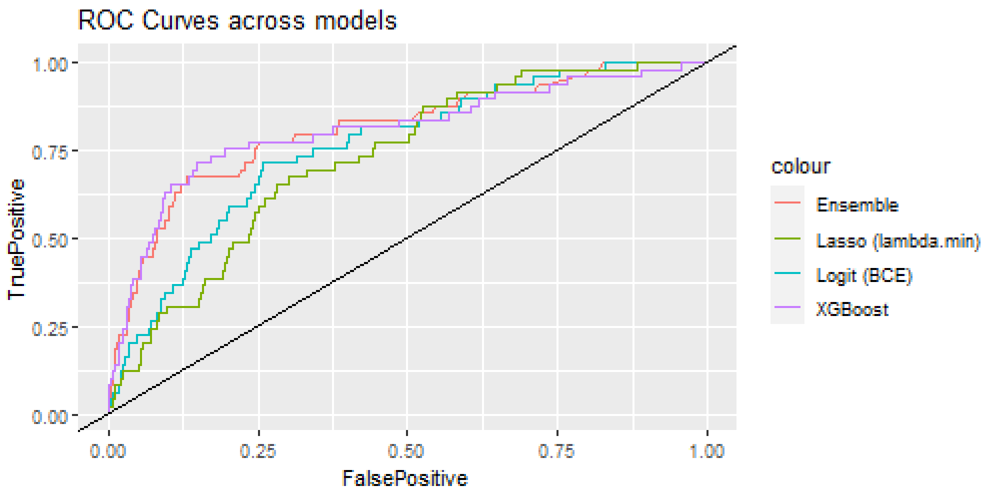
```r
library(xgboost)

# Prep data
# -1 to remove the first column: Test or AAER
train_x <- model.matrix(AAER ~ ., data = df[df$Test == 0, -1])[, -1]
train_y <- model.frame(AAER ~ ., data = df[df$Test == 0, ])[, "AAER"]
test_x <- model.matrix(AAER ~ ., data = df[df$Test == 1, -1])[, -1]
test_y <- model.frame(AAER ~ ., data = df[df$Test == 1, ])[, "AAER"]

set.seed(468435)  #for reproducibility
xgbCV <- xgb.cv(max_depth = 5, eta = 0.10, gamma = 5, min_child_weight = 4,
                subsample = 0.57, objective = "binary:logistic", data = train_x,
                label = train_y, nrounds = 100, eval_metric = "auc", nfold = 10,
                stratified = TRUE, verbosity = 0)

fit_ens <- xgboost(params = xgbCV$params, data = train_x, label = train_y,
                   nrounds = which.max(xgbCV$evaluation_log$test_auc_mean))
```

How about our fraud detection models?

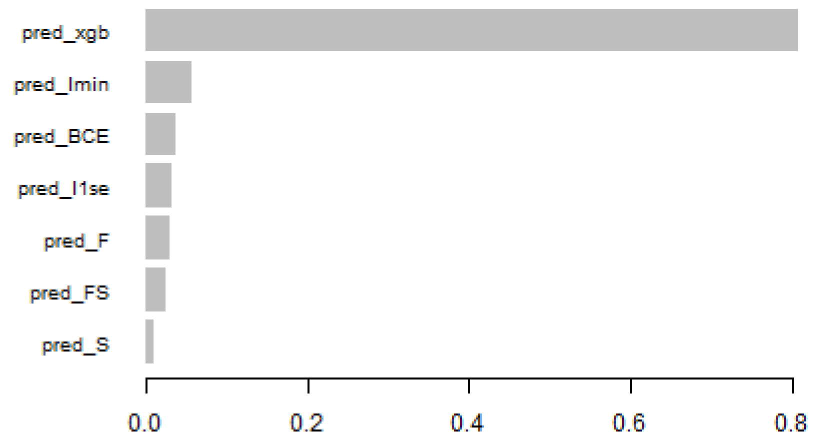



```
aucs  # Out of sample
```

```
##           Ensemble        Logit (BCE) Lasso (lambda.min)            XGBoost
##          0.8115824          0.7599594          0.7290185          0.8083503
```

What drives the ensemble?

```r
xgb.train.data = xgb.DMatrix(train_x, label = train_y, missing = NA)
col_names = attr(xgb.train.data, ".Dimnames")[[2]]
imp = xgb.importance(col_names, fit_ens)
# Variable importance
xgb.plot.importance(imp)
```



Practicalities

- Methods like stacking or blending are much more complex than a simple averaging or voting based ensemble
  - But in practice they perform slightly better

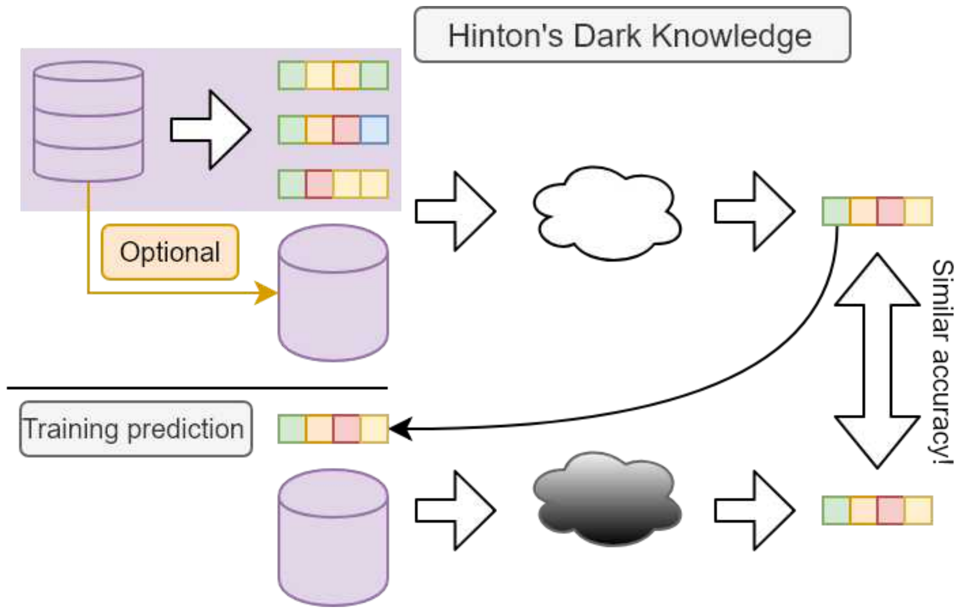
> Recall the tradeoff between complexity and accuracy!

- As such, we may not prefer the complex ensemble in practice, unless we only care about accuracy

> Example: In 2009, Netflix awarded a $1M prize to a team for beating Netflix's own user preference algorithm by >10%. The alogorithm was so complex that Netflix never used it. It instead used a simpler algorithm with an 8% improvement.

[Geoff Hinton's] Dark Knowledge

- Complex ensembles work well but exceedingly computationally intensive
  - This is bad for running on small or constrained devices (like phones)
  - How long are you willing to take when grabbing a taxi?
- We can always create a simple model that approximates the complex model
- Train the simple model not on the actual training data, but on the best algorithm's prediction for the training data
- Somewhat surprisingly, this new, simple algorithm can work almost as well as the full thing!



Learning more about Ensembling

- Geoff Hinton's Dark Knowledge slides
  - For more details on *dark knowledge*, applications, and the softening transform
  - His interesting (though highly technical) Reddit AMA
- Stacking Models for Improved Predictions
  - A short guide on stacking with nice visualizations
- Kaggle Ensembling Guide
  - A comprehensive list of ensembling methods with some code samples and applications discussed
- Ensemble Learning to Improve Machine Learning Results
  - Nicely covers bagging and boosting (two other techniques)

> There are many ways to ensemble, and there is no specific guide as to what is best. It may prove useful in the group project, however.

Ethics: Fairness

In class reading with case

- From Datarobot's Colin Preist:
  - Four Keys to Avoiding Bias in AI
- The four points:
  1. Data can be correlated with features that are illegal to use
  2. Check for features that could lead to ethical or reputational problems
  3. "An AI only knows what it is taught"
  4. Entrenched bias in data can lead to biased algorithms

> What other ethical issues might we encounter?

Examples of reputational damage

- Microsoft's Tay and their response
- Coca-Cola: Go make it happy
- Google: Google Photos mistakenly labels black people 'gorillas'
- Machine Bias
  - "blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend."
  - The number of true positives divided by the number of all positives is more or less equal across ethnicities

Prediction Fails Differently for Black Defendants

	WHITE	AFRICAN AMERICAN
Labeled Higher Risk, But Didn't Re-Offend	23.5%	44.9%
Labeled Lower Risk, Yet Did Re-Offend	47.7%	28.0%

Overall, Northpointe's assessment tool correctly predicts recidivism 61 percent of the time. But blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend. It makes the opposite mistake among whites: They are much more likely than blacks to be labeled lower risk but go on to commit other crimes. (Source: ProPublica analysis of data from Broward County, Fla.)

Perils of Machine Learning

But what about...

Commonsense knowledge

Logical reasoning

Linguistic phenomena

Intuitive physics

...

MACHINE LEARNING DOESN'T CARE

Examining ethics for algorithms

1. Understanding the problem and its impact
    - Think about the effects the algorithm will have!
        - Will it drastically affect lives? If yes, exercise more care!
    - Think about what you might expect to go wrong
        - What biases might you expect?
        - What biases might be in the data?
        - What biases do people doing the same task exhibit?
2. Manual inspection
    - Check association between model outputs and known problematic indicators
    - Test the algorithm before putting it into production
3. Methods like SHAP (SHapley Additive exPlanations) to explain models
4. Use purpose-built tools
    - Facebook's Fairness Flow
    - Accenture's Fairness Tool
    - Microsoft's Fairness Tool

Areas where ethics is particularly important

- Anything that impacts people's livelihoods
  - `Legal systems`
  - `Healthcare/Insurance systems`
  - `Hiring and HR systems`
  - `Finance systems like credit scoring`
  - `Education`
- Anything where failure is catastrophic
  - Voting systems
  - Engineering systems
  - Transportation systems
    - Such as the Joo Koon MRT Collision in 2017
    - Self driving cars (Results summary) Try it!

> A good article of examples of the above: Algorithms are great and all, but they can also ruin lives

Algorithms assurance

- Algorithms assurance: The future of auditing?
  - Singapore to establish AI framework for 'fairness' credit scoring metrics
  - The EU's General Data Protection Regulation (GDPR) requires that organizations be able to explain their algorithmic decisions
  - In France, all algorithms developed for government use will be made publicly available
  - Big 4
    - Deloitte's assurance over machine learning and algorithms
    - EY's assurance in the age of AI
    - PwC's Managing AI risk with confidence
    - KPMG's In AI we trust?

Other references

- Kate Crawford's NIPS 2017 Keynote: "The Trouble with Bias" (video)
- Google's Responsible AI
  - Watch the video
- Detecting Data Bias Using SHAP and ML
  - A nice article showing off the extra interpretability coming from Python's shap library in the context of programmer's salaries

Ethics: Data security

Recall the Walmart data

- 45 stores from different regions
  - Each stores' revenue by department
  - **Each store is anonymized with an ID**
    - A number between 1 and 45
- Other data: CPI, labor, weather, gas price

> What if I said I could tell you which store ID 32 and 11 were located?

- Walmart would likely not be happy about this
  - Competitors could use this to extract private information and strategize against Walmart
- They did a bad job anonymizing the data, making this possible

How did I back this out?

- Walmart provides data on certain macro factors in the region around the store
  - They don't give much detail on exactly how these are calculated though
- Temperature
  - Unclear if this is average or high temperature
  - Provided weekly
- Cross reference with monthly temperature data from >1000 weather stations in the US
  - Check error (RMSE) for every station vs every store

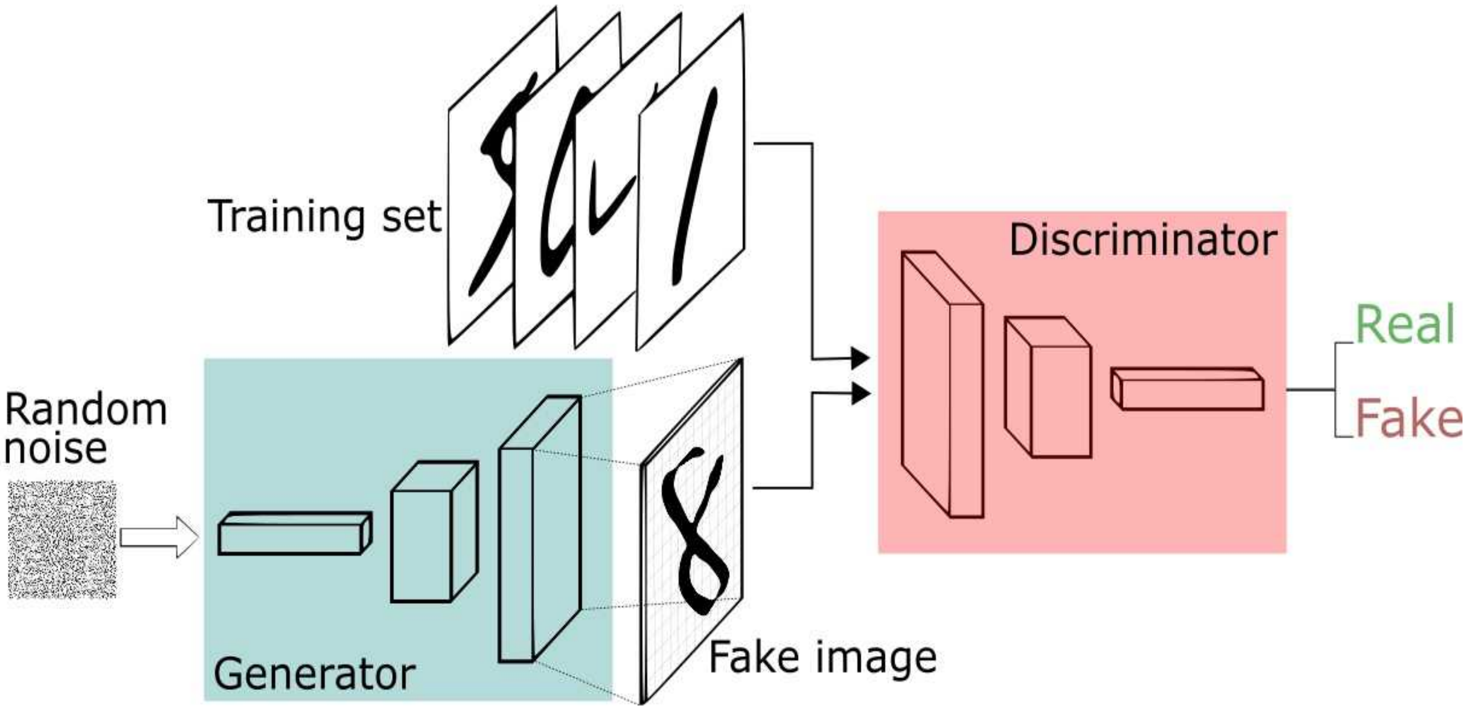
Anomymizing data is tricky

- Generally we anonymize data because, while the data itself is broadly useful, providing full information could harm others or oneself

> "...... warn against relying on the anonymisation of data since *deanonymisation techniques are often surprisingly powerful*. Robust anonymisation of data is difficult, particularly when it has high dimensionality, as the anonymisation is likely to lead to an unacceptable level of data loss." -- TPHCB 2017

A novel approach using a GAN

- Source: Learning Anonymized Representations with Adversarial Neural Networks
- On handwriting classification, cases that can be deanonymized drop from 40% to 3.3%
    - Accuracy drops from ~98% down to 95%, a much smaller drop



Responsibilities generating data

- Keep users as unidentifiable as feasible
- If you need to record people's private information, **make sure they know**
  - This is called *informed consent*
- If you are recording sensitive information, consider not keeping identities at all
  - Create a new, unique identifier (if needed)
  - Maintain as little identifying information as necessary
  - Consider using encryption if sensitive data is retained
  - Can unintentionally lead to infringements of *human rights* if the data is used in unintended ways

> Also, note the existence of the PDPA law in Singapore
>
> For experiments, see The Belmont Report
>
> For electronic data, see The Menlo Report

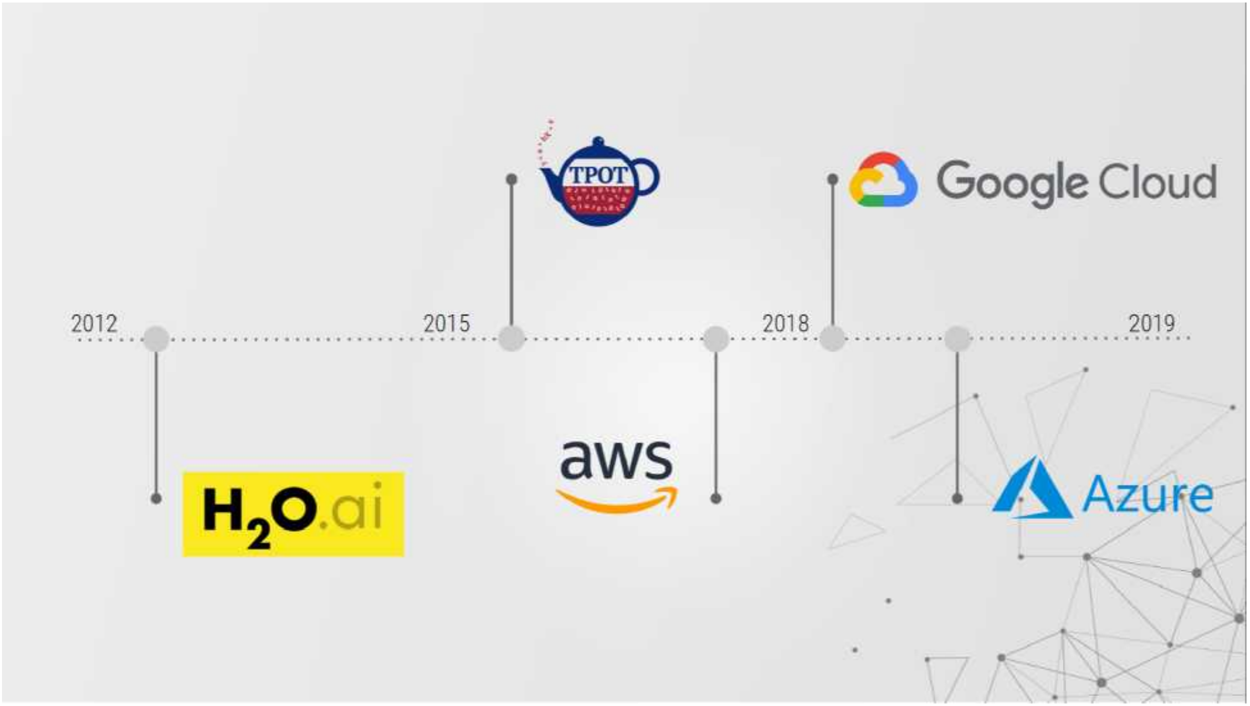
Informed consent

- When working with data about *people*, they should be informed of this and consent to the research, unless the data is publicly available
- From SMU's IRB Handbook: (2017 SEP 18 version)
- "*Informed consent*: Respect for persons requires that participants, to the degree that they are capable, be given the opportunity to make their own judgments and choices. When researchers seek participants' participation in research studies, they provide them the opportunity to make their own decisions to participate or not by ensuring that the following adequate standards for informed consent are satisfied:
  - *Information*: Participants are given sufficient information about the research study, e.g., research purpose, study procedures, risks, benefits, confidentiality of participants' data.
  - *Comprehension*: The manner and context in which information is conveyed allows sufficient comprehension. The information is organized for easy reading and the language is easily comprehended by the participants.
  - *Voluntariness*: The manner in which researchers seek informed consent from the participants to participate in the research study must be free from any undue influence or coercion. Under such circumstances, participants are aware that they are not obliged to participate in the research study and their participation is on a voluntary basis."

Automated Machine Learning

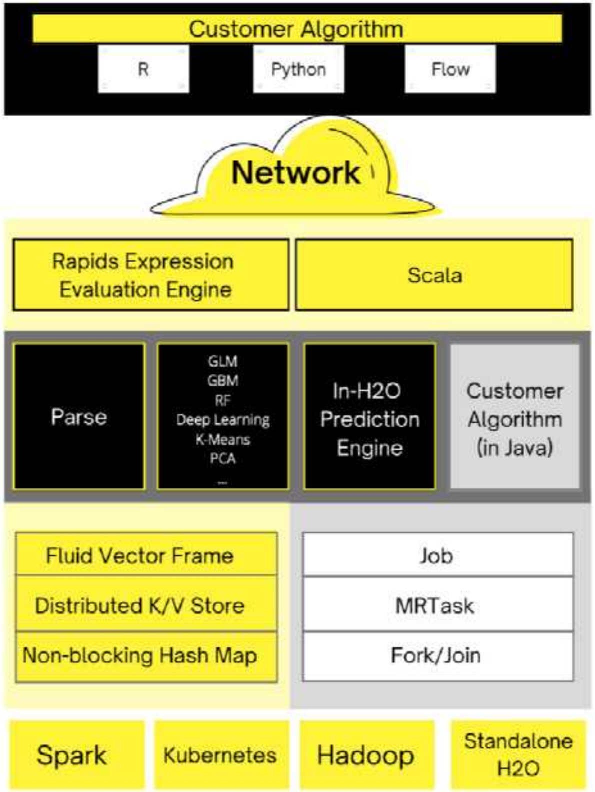
The Future of Data Scientists?

- With automated machine learning service, we can focus on the most accurate models and avoid testing a large range of less valuable models.



H2O AutoML

- H2O.ai provides a free automated ML platform which has R/Python interface
- AutoML with R



Initiate H2O AutoML in R

```r
# The easiest way is to install h2o within RStudio through CRAN
# But the CRAN version is typically one version behind the most recent version
# It is fine and you may ignore the warning messages about Java and H2o cluster
# install.packages("h2o")

# If you want to install the most recent version of H2O AutoML
# Go to http://h2o-release.s3.amazonaws.com/h2o/rel-zermelo/4/index.html
# and click "INSTALL IN R" tab

library(h2o)
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         4 hours 56 minutes
##     H2O cluster timezone:       Asia/Singapore
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.4
##     H2O cluster version age:    1 month and 12 days
##     H2O cluster name:           H2O_started_from_R_jwwang_awh027
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   7.23 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
```

Load the data

```
# use data.table package to handle large files
# you may need to install "bit64" package to activate data.table
options("h2o.use.data.table" = TRUE)
# Import the loan .csv data into H2O
# You may also convert an R dataframe to h2o format directly
data.h2o <- h2o.importFile("../../Data/Session_10_ensemble.csv")
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
summary(data.h2o)
```

```
##  Loan_ID Gender       Married  Dependents        Education         Self_Employed
##          Male   :489  Yes:398  Min.    :0.0000   Graduate     :480  No :500
##          Female:112   No :213  1st Qu.:0.0000   Not Graduate:134  Yes: 82
##          NA     : 13  NA :  3  Median :0.0000                     NA : 32
##                                Mean    :0.5547
##                                3rd Qu.:1.0000
##                                Max.    :2.0000
##                                NA's    :66
##  ApplicantIncome CoapplicantIncome LoanAmount      Loan_Amount_Term
##  Min.    :  150  Min.    :     0   Min.    :  9.0  Min.    : 12
##  1st Qu.: 2818   1st Qu.:     0   1st Qu.:100.0  1st Qu.:360
##  Median : 3788   Median : 1188    Median :128.0  Median :360
##  Mean    : 5403  Mean    : 1621   Mean    :146.4  Mean    :342
```

Prepare the data

```r
data.h2o <- data.h2o[, c("Credit_History", "LoanAmount", "Loan_Amount_Term",
                         "ApplicantIncome", "CoapplicantIncome", "Loan_Status")]
# Construct test and train sets using sampling
h2o.impute(data.h2o, method = "median")
```

```
## [1]    0.8421986  146.4121622  342.0000000 5403.4592834 1621.2457980
## [6]    1.0000000
```

```r
data.h2o.split = h2o.splitFrame(data = data.h2o, ratios = 0.75)
data.h2o.train = data.h2o.split[[1]]
data.h2o.test = data.h2o.split[[2]]

# Identify predictors and response
y <- "Loan_Status"
x <- setdiff(names(data.h2o.train), y)

# For binary classification, response should be a factor
data.h2o.train[, y] <- as.factor(data.h2o.train[, y])
data.h2o.test[, y] <- as.factor(data.h2o.test[, y])
```

Run the AutoML model

- Use the `h2o.automl()` function
- Currently XGBoost is not available on Windows machines

```
# Run AutoML for 20 models (limited to 1 hour max_runtime_secs by default)
# use include_algos and exclude_algos to choose models.
# It will take some time, you may reduce the max_models number to save time
aml <- h2o.automl(x = x, y = y, training_frame = data.h2o.train,
                  max_models = 20, max_runtime_secs = 120,
                  #exclude_algos = c(""), include_algos = c(""),
                  seed = 123) # Set a seed for reproducibility
```

```
##
  |
  |                                                                      |   0%
## 15:14:30.662: AutoML: XGBoost is not available; skipping it.
  |
  |=========                                                             |  12%
  |
  |============                                                          |  17%
  |
  |==================                                                    |  25%
  |
  |====================                                                  |  29%
  |
  |==========================                                            |  38%
  |
  |======================================                                |  55%
  |
```

AutoML leaderboard

```r
# View the AutoML Leaderboard
DT::datatable(as.data.frame(aml@leaderboard[, c("model_id", "auc")]),
              options = list(pageLength = 3))
```

Show 3 entries   Search:

	model_id	auc
1	DeepLearning_grid__2_AutoML_20210313_151430_model_2	0.762050653594771
2	GBM_4_AutoML_20210313_151430	0.758748638344227
3	DRF_1_AutoML_20210313_151430	0.756910403050109

Showing 1 to 3 of 22 entries

Previous 1 2 3 4 5 … 8 Next

```r
# Make predictions using the best model @leader
# pred <- h2o.predict(aml@leader, data.h2o.test)
```

How about financial detection models?

```
# Convert the detection model data into H2O format
data.h2o <- as.h2o(df)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
summary(data.h2o)
```

```
##  Test             AAER               pred_F              pred_S
##  Min.   :0.0000   Min.   :0.00000   Min.   :1.291e-07   Min.   :2.220e-16
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:7.633e-03   1st Qu.:1.552e-02
##  Median :0.0000   Median :0.00000   Median :1.437e-02   Median :1.995e-02
##  Mean   :0.1971   Mean   :0.02045   Mean   :2.188e-02   Mean   :2.187e-02
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:2.784e-02   3rd Qu.:2.549e-02
##  Max.   :1.0000   Max.   :1.00000   Max.   :2.993e-01   Max.   :3.695e-01
##  pred_FS             pred_BCE            pred_lmin           pred_l1se
##  Min.   :2.220e-16   Min.   :2.220e-16   Min.   :0.0004233   Min.   :0.001315
##  1st Qu.:6.274e-03   1st Qu.:4.651e-03   1st Qu.:0.0093053   1st Qu.:0.010798
##  Median :1.344e-02   Median :1.137e-02   Median :0.0160267   Median :0.017005
##  Mean   :2.233e-02   Mean   :2.202e-02   Mean   :0.0219844   Mean   :0.021760
##  3rd Qu.:2.689e-02   3rd Qu.:2.584e-02   3rd Qu.:0.0280294   3rd Qu.:0.027696
##  Max.   :4.481e-01   Max.   :5.168e-01   Max.   :0.2404756   Max.   :0.173738
```

How about financial detection models?

```r
# split the data based on Test indicator
data.h2o.train = data.h2o[data.h2o[, "Test"] == 0, ]
data.h2o.test = data.h2o[data.h2o[, "Test"] == 1, ]

# Identify predictors and response
y <- "AAER"
x <- c("pred_F", "pred_S", "pred_FS", "pred_BCE", "pred_lmin",
       "pred_l1se", "pred_xgb")

# For binary classification, response should be a factor
data.h2o.train[, y] <- as.factor(data.h2o.train[, y])
data.h2o.test[, y] <- as.factor(data.h2o.test[, y])
```

How about financial detection models?

- Use the `h2o.automl()` function
- Currently XGBoost is not available on Windows machines

```
# Run AutoML for 20 models (limited to 1 hour max_runtime_secs by default)
# use include_algos and exclude_algos to choose models.
# It will take some time, you may reduce the max_models number to save time
aml <- h2o.automl(x = x, y = y, training_frame = data.h2o.train,
                  max_models = 20, max_runtime_secs = 120,
                  #exclude_algos = c(""), include_algos = c(""),
                  seed = 123) # Set a seed for reproducibility
```

How about financial detection models?

```r
# View the AutoML Leaderboard
DT::datatable(as.data.frame(aml@leaderboard[, c("model_id", "auc")]),
              options = list(pageLength = 3))
```

Show 3 entries Search:

	model_id	auc
1	GLM_1_AutoML_20210313_151557	0.975893355246835
2	GBM_5_AutoML_20210313_151557	0.975293889029912
3	DeepLearning_grid__1_AutoML_20210313_151557_model_2	0.974049243717

Showing 1 to 3 of 20 entries

Previous 1 2 3 4 5 6 7 Next

Bonus: AutoAL with TPOT

TPOT

- TPOT is a Python Automated Machine Learning tool.
- No R interface yet
- I will use TPOT as an example to show you how to run both R and Python within R Markdown

Setup R Interface to Python

- `package:reticulate` provides an R interface to Python which can
    - call Python from R in a variety of ways including R Markdown
    - translate between R and Python objects (for example, between R and Pandas data frames, or between R matrices and NumPy arrays)
    - bind to different versions of Python including virtual environments and Conda environments.

```r
# install the reticulate package directly
# install.packages("reticulate")

# launch the package
library(reticulate)

# specify the Python version to use
# https://rstudio.github.io/reticulate/articles/versions.html
# I assume you use the Anaconda which is easier to manage
use_python("C:\\ProgramData\\Anaconda3\\")
```

Install TPOT

- The simplest way is to use `conda-forge`
- Run `Anaconda Powershell Prompt` as Administrator
- Run `conda install -c conda-forge tpot`, you are ready to try TPOT
- If you want to install additional dependencies such as xgboost
    - Run `conda install -c conda-forge tpot xgboost dask dask-ml scikit-mdr skrebate`

TPOT preparation

```python
# This is a python code, you can only run it directly in RMarkdown
# If you want to run it together with your R session, please visit
# https://github.com/rstudio/reticulate for more information
import pandas as pd
from tpot import TPOTClassifier
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
# Read the data
tpot_data = pd.read_csv('../../Data/Session_10_ensemble.csv')
tpot_data = tpot_data[["Credit_History", "LoanAmount", "Loan_Amount_Term",
                "ApplicantIncome", "CoapplicantIncome", "Loan_Status"]]
# Create a dataframe with all features (ie, all X)
features = tpot_data.drop('Loan_Status', axis = 1) # drop column axis = 1
# Create training and test datasets
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    tpot_data['Loan_Status'],
                                                    train_size = 0.75,
                                                    test_size = 0.25)
```

Implement TPOT

```
# This is a python code, you can only run it directly in RMarkdown
# If you want to run it together with your R session, please visit
# https://github.com/rstudio/reticulate for more information

# Setup TPOT optimizer parameters
# generations: number of iterations, population_size: number of models
# cv: k-fold cross validation; random_state: for reproducibility
# verbosity: How much information TPOT communicates while it is running
pipeline_optimizer = TPOTClassifier(generations = 5,
                                    population_size = 20,
                                    cv = 5,
                                    random_state = 123,
                                    verbosity = 0)
# Fit the models
pipeline_optimizer.fit(X_train, y_train)
# evaluate and print out the testing set
print(pipeline_optimizer.score(X_test, y_test))
# Output the best model Python code
pipeline_optimizer.export('tpot_exported_pipeline.py')
```

Summary of Session 10

Recap

Today, we:

- Learned about combining models to create an even better model
- Discussed the potential ethical issues surrounding:
  - AI algorithms
  - Data creation
  - Data usage
  - Data security
- Use H2O.ai's AutoML to automate machine learning
- Bonus: R interface to Python
  - TPOT AutoML with Python within R Markdown

For next week

- We will talk about neural networks and deep learning
  - These are important tools underpinning a lot of recent advancements
  - We will take a look at some of the advancements, and the tools that underpin them
- If you would like to be well prepared, there is a nice introductory article at here (8 parts though)
  - We have covered Part 1 and Part 2 is useful for next week
  - For those very interested in machine learning and deep learning, parts 3 through 8 are also great, but more technical and targeted at specific applications like facial recognition and machine translation
- Keep working on the group project, try an ensembling of your models
  - You may also try the automated ML.

Fun machine learning examples

- Interactive:
  - Semantris
    - A game based on the Universal Sentence Encoder
  - Draw together with a neural network
    - click the images to try it out yourself!
  - Google's Quickdraw
  - Google's Teachable Machine
  - Four experiments in handwriting with a neural network
- Non-interactive
  - Predicting e-sports winners with Machine Learning

R packages used in this slide

This slide was prepared on 2021-03-13 from Session_10s.Rmd with R version 4.0.3 (2020-10-10) Bunny-Wunnies Freak Out on Windows 10 x64 build 18362 😀.

The attached packages used in this slide are:

```
##         h2o        ROCR     xgboost       caret    lattice     forcats     stringr
## "3.32.0.4"   "1.0-11"  "1.3.2.1"    "6.0-86"  "0.20-41"     "0.5.1"     "1.4.0"
##       dplyr       purrr       readr       tidyr     tibble     ggplot2   tidyverse
##     "1.0.4"     "0.3.4"     "1.4.0"     "1.1.2"    "3.0.6"     "3.3.3"     "1.3.0"
## kableExtra       knitr
##     "1.1.0"      "1.31"
```