



FITE7410

Lecture 3

Lecturers: Dr. Vivien CHAN, Annie CHAN
Tutor: Ms. Yanan GONG

Department of Computer Science
The University of Hong Kong



Today's Agenda

01 Decision Tree

02 Ensemble Learning

03 Random Forest

04 Neural Network



01 Decision Trees

Dr. Vivien CHAN

Decision Trees - Basics

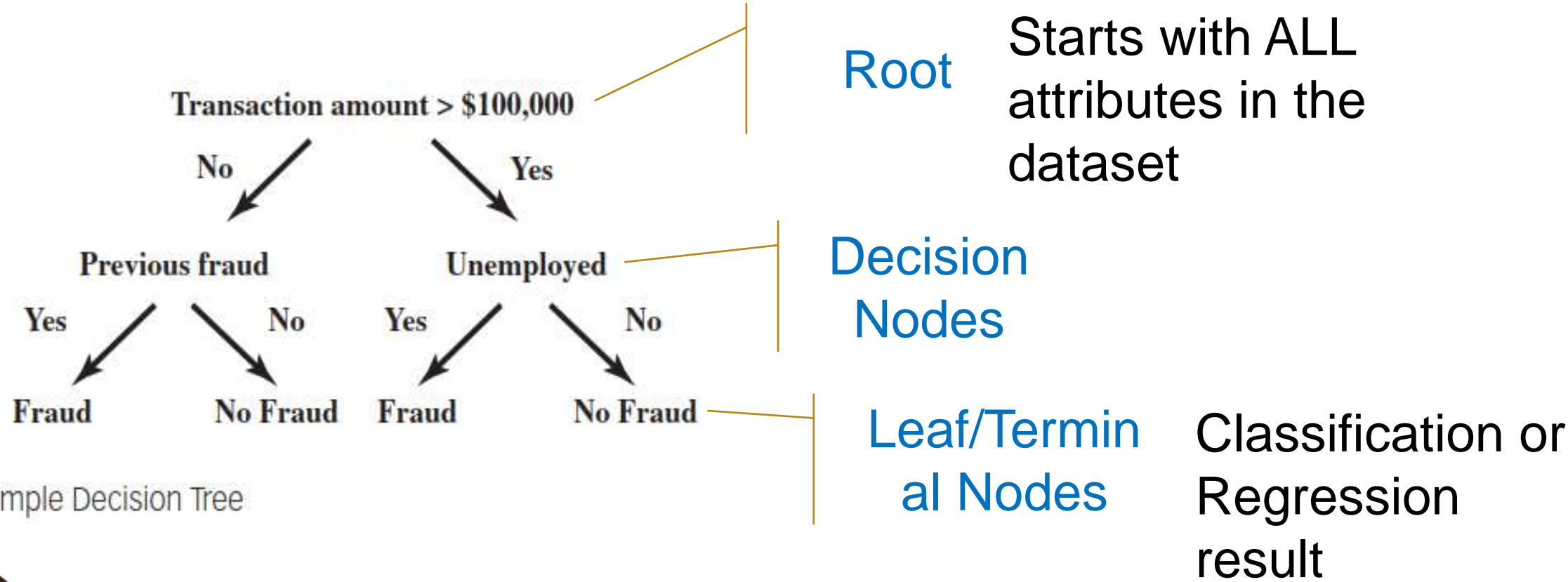


Figure 4.10 Example Decision Tree



Classification Decision Trees vs Regression Decision Trees

Decision Trees - Basics

- Splitting decision
 - Which attribute to split at what value?
 - e.g. Transaction amount is > \$100, 000 or not, Previous fraud is yes or no, unemployed is yes or no
- Stopping decision
 - When to stop adding nodes to the tree?
- Assignment decision
 - What class (e.g., fraud or no fraud) to assign to a leave node?

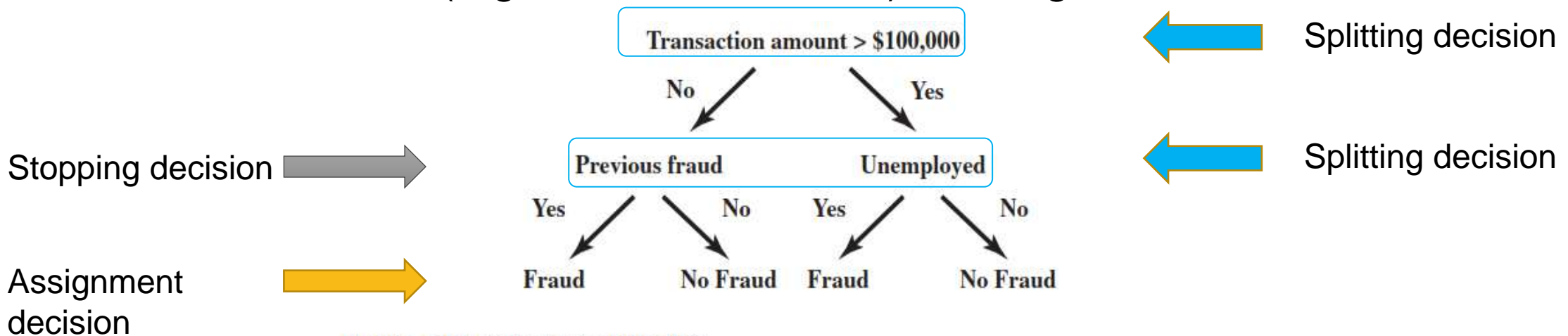


Figure 4.10 Example Decision Tree

Decision Tree

- Decision Tree aims at minimizing “impurity” in the data
- What is impurity?
 - The **node impurity** is a measure of the homogeneity of the labels at the **node**.
 - Two **impurity** measures for classification (Gini **impurity** and entropy) and one **impurity** measure for regression (variance).

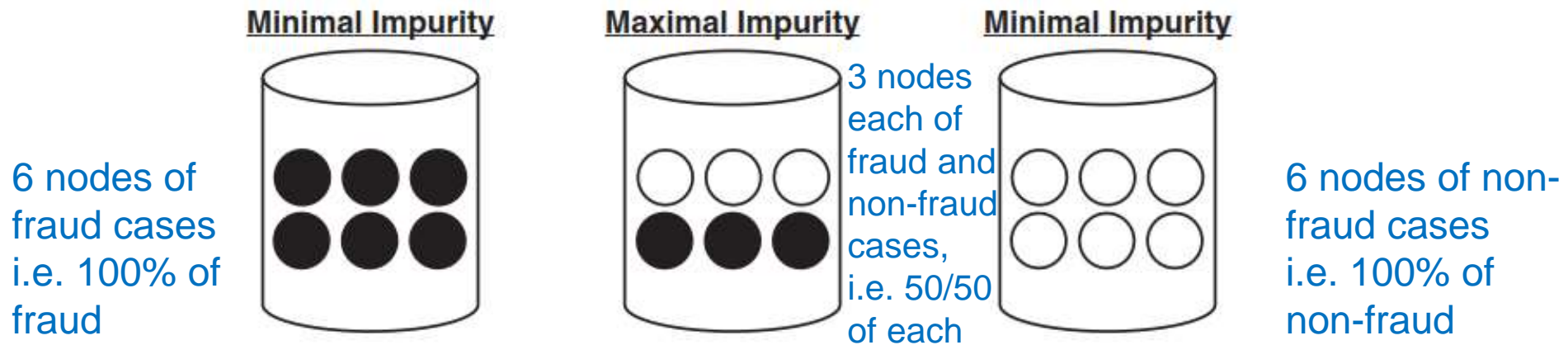


Figure 4.11 Example Data Sets for Calculating Impurity

Decision Tree – Splitting decision

1/ Categorical variables

Entropy: $E(S) = -p_G \log_2(p_G) - p_B \log_2(p_B)$ (C4.5/See5)

Gini: $Gini(S) = 2p_G p_B$ (CART)

Chi-squared analysis (CHAID)

Where

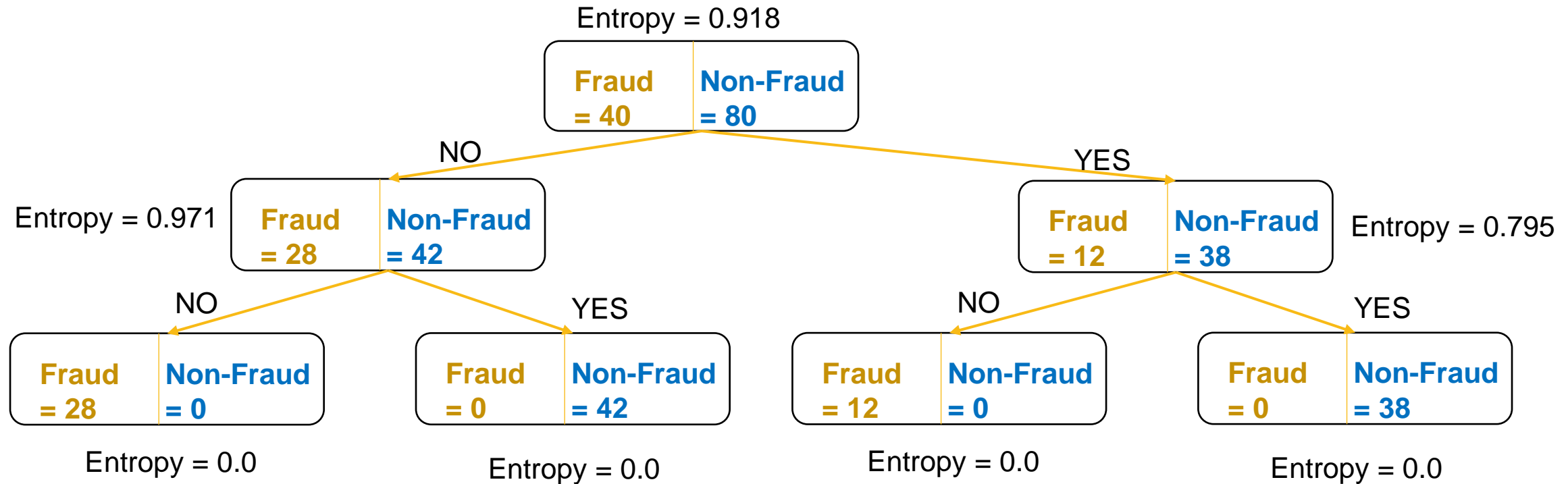
p_G = proportions of “good” or non-fraud observations

p_B = proportions of “bad” or fraud observations

2/ Continuous variables

- Use mean square error (MSE) or variance to predict the fraud percentage(FP)
- Also called Regression Tree

Decision Tree - example



Information Gain

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N} I(D_{left}) - \frac{N_{right}}{N} I(D_{right})$$

f: feature split on

D_p : dataset of the parent node

D_{left} : dataset of the left child node

D_{right} : dataset of the right child node

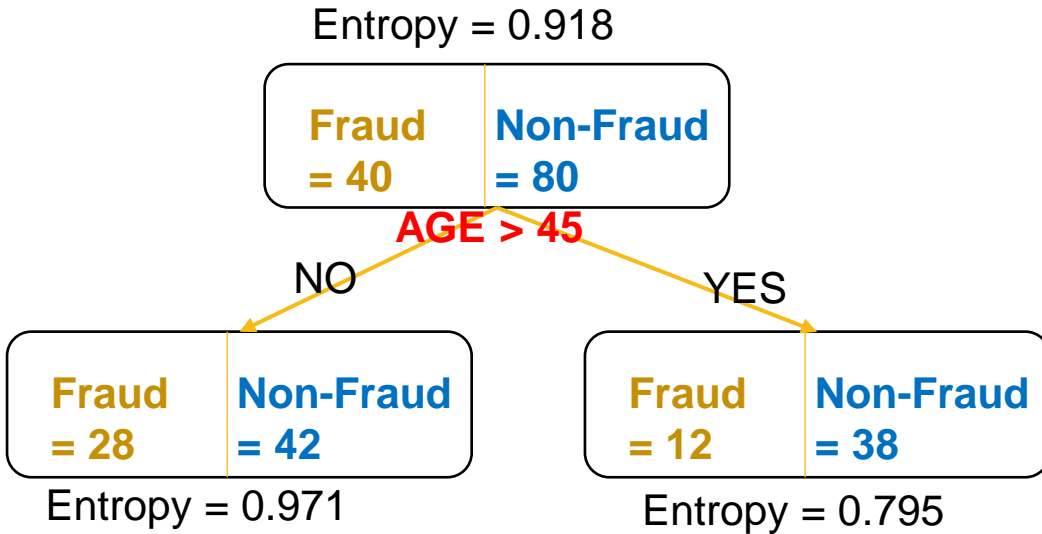
I: impurity criterion (Gini Index or Entropy)

N: total number of samples

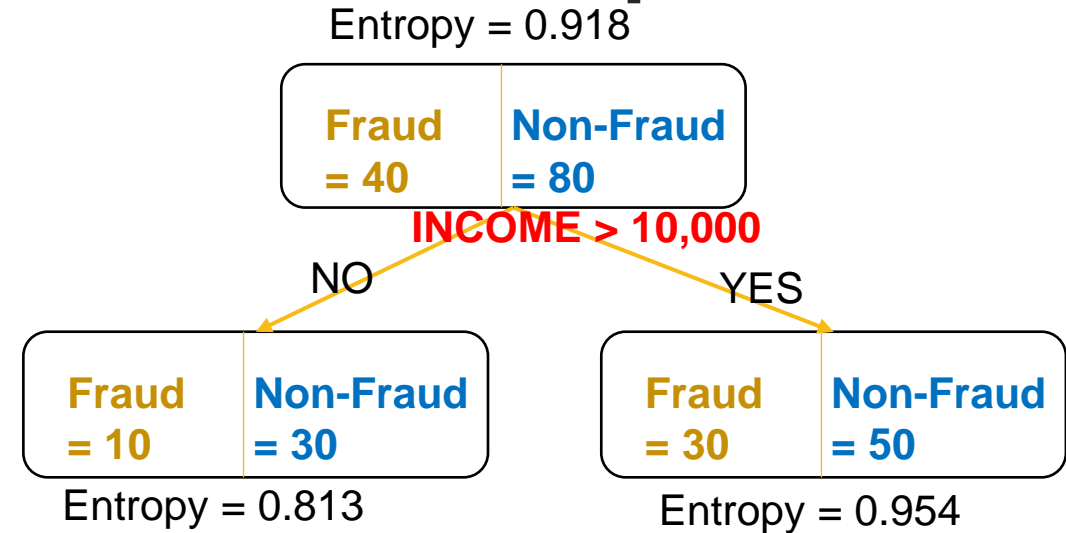
N_{left} : number of samples at left child node

N_{right} : number of samples at right child node

Decision Tree - example



$$\begin{aligned}
 IG(D_p, AGE) &= 0.918 - 70/120 \times 0.971 - 50/120 \times 0.795 \\
 &= 0.02
 \end{aligned}$$



$$\begin{aligned}
 IG(D_p, INCOME) &= 0.918 - 40/120 \times 0.813 - 80/120 \times 0.954 \\
 &= 0.01
 \end{aligned}$$

- Compare the IG for splitting decision using the attribute “AGE” and “INCOME”
- $IG(AGE) > IG(INCOME)$
=> use AGE as the splitting attribute

Regression Tree - Example

- Regression tree splits - favour homogeneity within node and heterogeneity between nodes
- E.g. favour low MSE in a leaf node

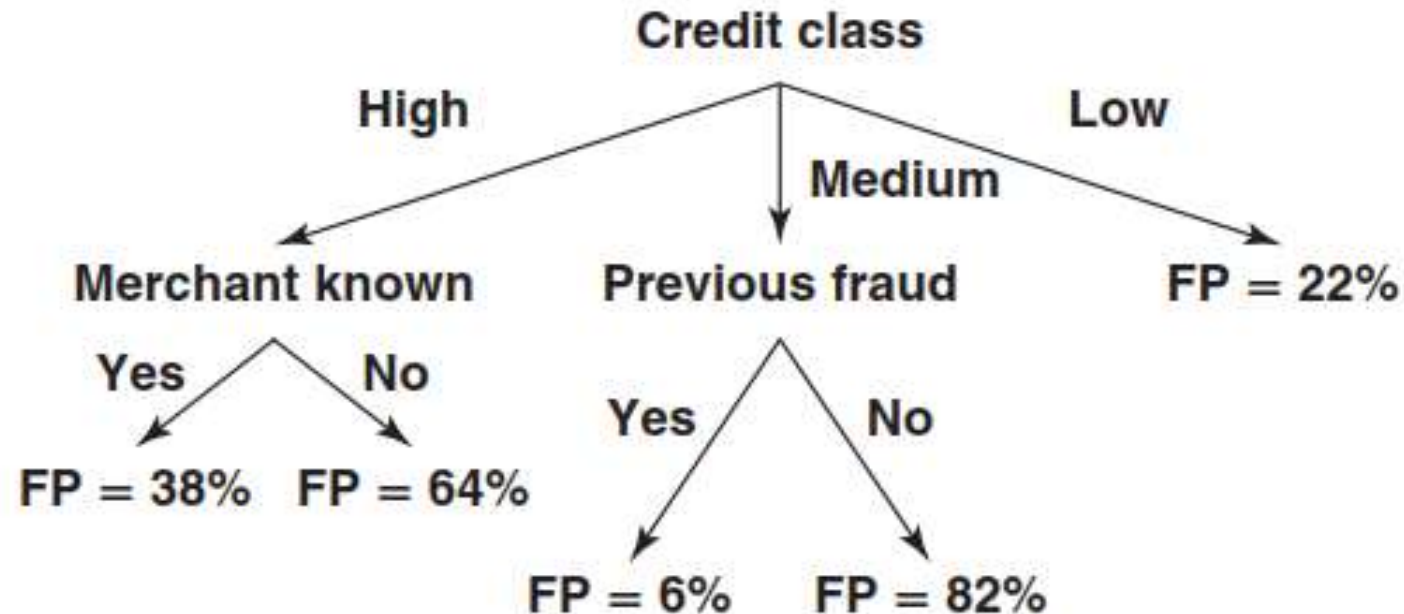


Figure 4.16 Example Regression Tree for Predicting the Fraud Percentage

Classification vs Regression tree

Classification Tree

used when dependent variable is categorical.

the value (class) obtained by terminal node in the training data is the mode of observations falling in that region

Regression Tree

used when dependent variable is continuous.

the value obtained by terminal nodes in the training data is the mean response of observation falling in that region

- Both the trees divide the predictor space (independent variables) into distinct and non-overlapping regions
- Both the trees follow a top-down greedy approach known as recursive binary splitting
- This splitting process is continued until a user defined stopping criteria is reached
- The splitting process results in fully grown trees until the stopping criteria is reached. But, the fully grown tree is likely to overfit data, leading to poor accuracy on unseen data.



What are the
possible problems
with Decision Tree
model?

Challenges of Decision Tree

- Problem: Overfitting
 - Overfitting - refers to the condition when the model completely fits the training data but fails to generalize the testing unseen data
 - Problem of overfitting - it continues to grow the tree, i.e. the tree has become too complex and fails to correctly model the noise free pattern or trend in the data
- Solutions:
 - A) Pruning
 - Instead of creating a FULLY grown tree, remove the sections of the tree that are not useful to the classification result
 - **Pre-pruning** : Pre-defined criteria to stop growing the tree at early stage, e.g. depth of tree, minimal number of records after the split, etc.
 - **Post-pruning** : FULLY grown a tree and use a validation dataset to decide the size of the tree
 - B) Ensemble methods

To be discussed later

Decision Tree – Post-pruning

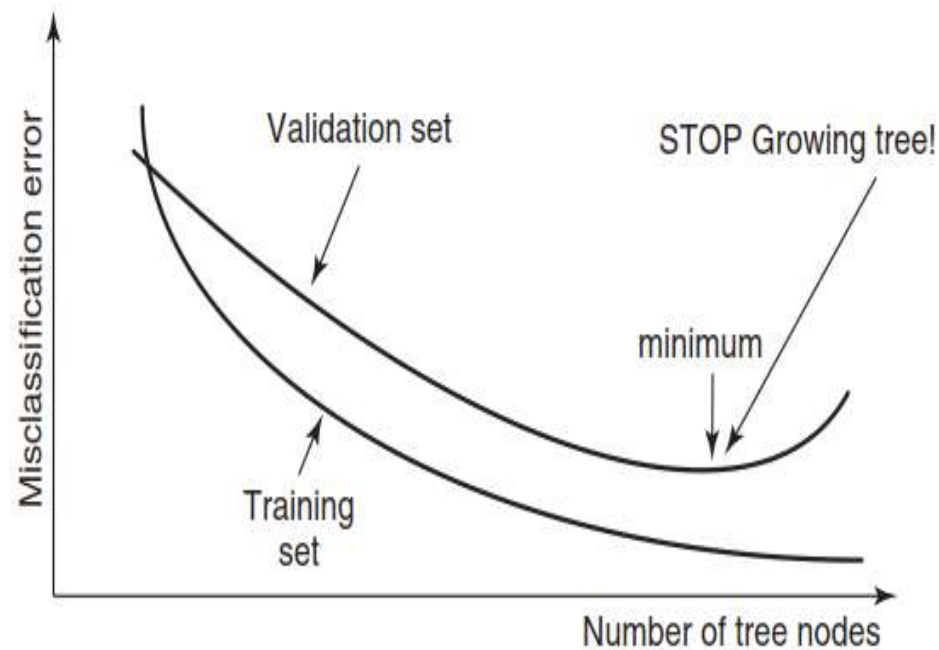


Figure 4.14 Using a Validation Set to Stop Growing a Decision Tree

- Steps for stopping decision:
 - Split the data into Training set and Validation set
 - Usually, 70% of sample data will be used as Training set and 30% of the Training set as Validation set
 - Stop growing the tree when the misclassification error for Validation set reaches its minimal value

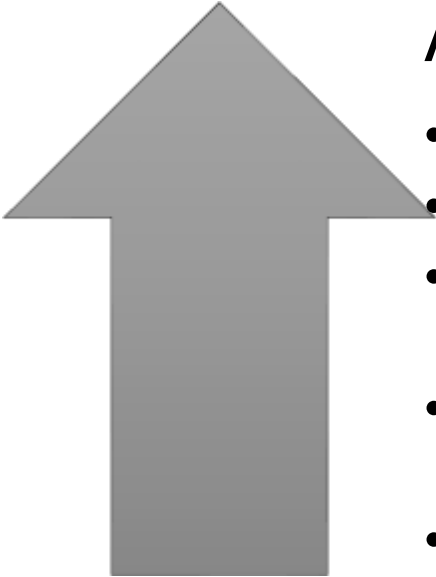


Which dataset should we use to evaluate the performance of the Decision Tree?

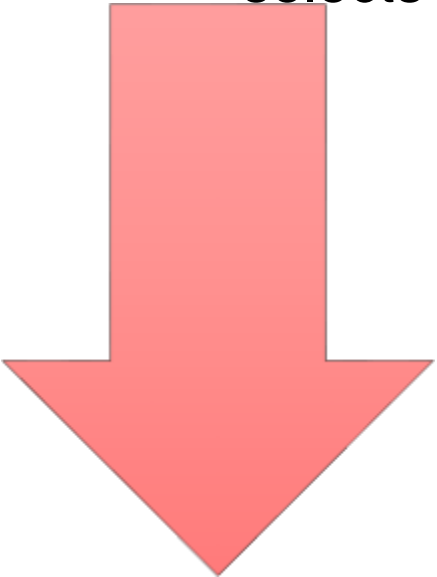
- a. Training Dataset
- b. Validation Dataset
- c. Testing Dataset

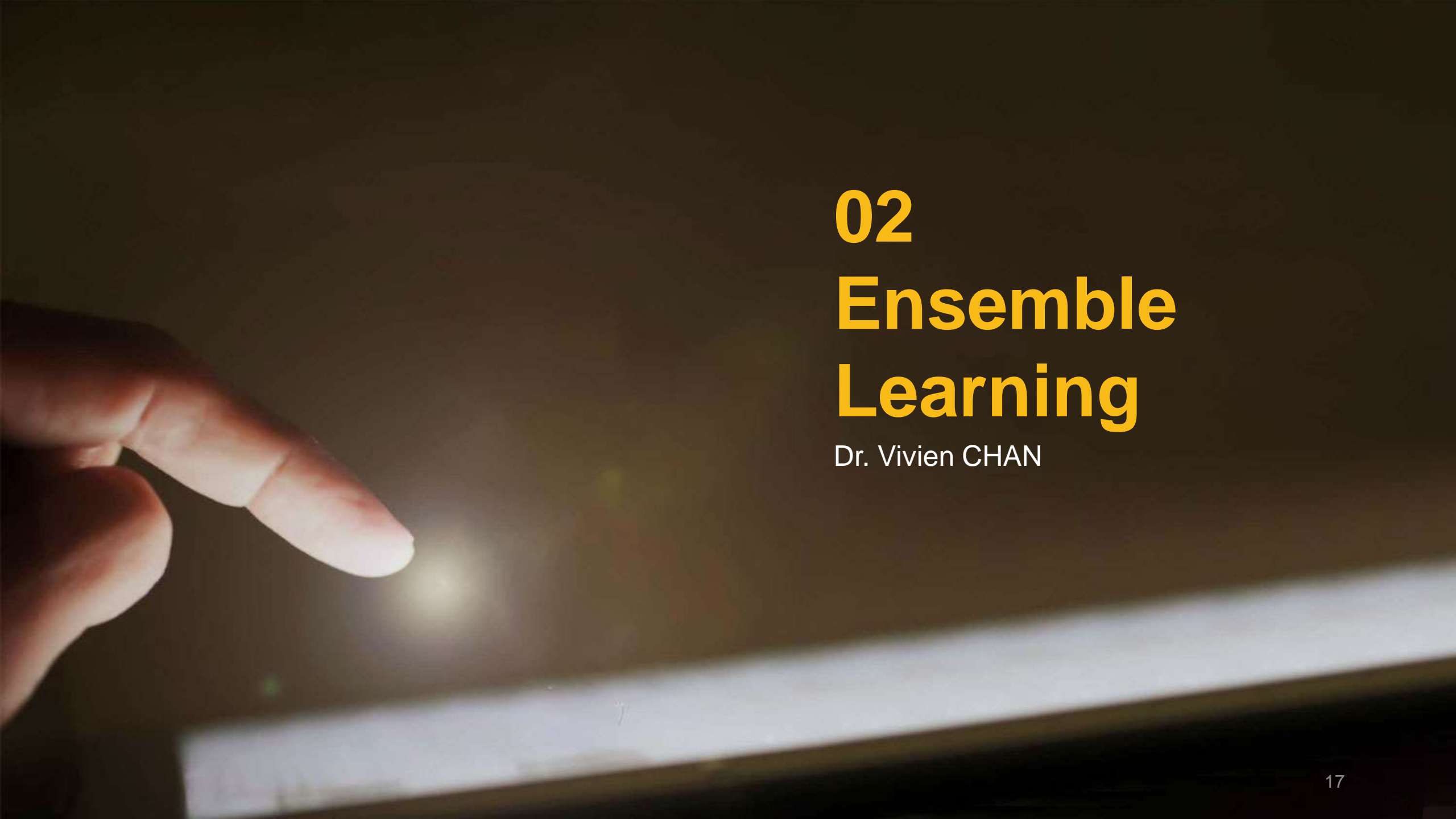
Decision Tree

Advantages

- 
- Easy to understand and interpret
 - Useful in data exploration
 - Less data clearing required – robust to outliers in inputs and no problem with missing values
 - Data type not a constraint – can handle both continuous and categorical data for both input and target data
 - Automatically detects interactions, accommodates nonlinearity and selects input variables

Disadvantages

- 
- Prone to overfitting
 - Splitting turns continuous input variables into discrete variables
 - Unstable fitted tree – small change in the data result in a very different series of splits

A hand is visible on the left side of the frame, with the index finger pointing towards the center. The background is a dark, out-of-focus presentation screen with a bright horizontal band of light near the bottom.

02 Ensemble Learning

Dr. Vivien CHAN

Ensemble Learning

- Ensemble learning is a machine learning model that involve a group of prediction models (or “base models” or “weak learners”).
- Reasons of using ensemble learning:
 - 1/ **Performance**: An ensemble can make better predictions and achieve better performance than any single contributing model.
 - 2/ **Robustness**: An ensemble reduces the spread or dispersion of the predictions and model performance.

Ensemble Learning

- Homogenous ensemble learning
 - Combining SAME base models (“weak learners”) for training
- Heterogenous ensemble learning
 - Combining DIFFERENT base models for training
 - NOTE: the choice of these different base models should be coherent with how you aggregation these base models

Ensemble - Combining base models

Bagging

- Used for homogeneous base models
- Learns the base models independently from each other and in parallel
- Combines the base models by some kind of deterministic averaging process
- Objective: reduce variance

Example:
Random
Forest

Boosting

- Used for homogeneous base models
- Learns the base models sequentially in a very adaptive way, i.e. base models depends on previous ones
- Combines the base models following a deterministic strategy
- Objective: reduce bias

Example:
AdaBoost,
GradientBoos
t

Stacking

- Used for heterogenous base models
- Learns the base models in parallel
- Combines the base models by training a meta-model to output a prediction based on the different base models predictions
- Objective : improve performance results

Bagging - Bootstrap sampling

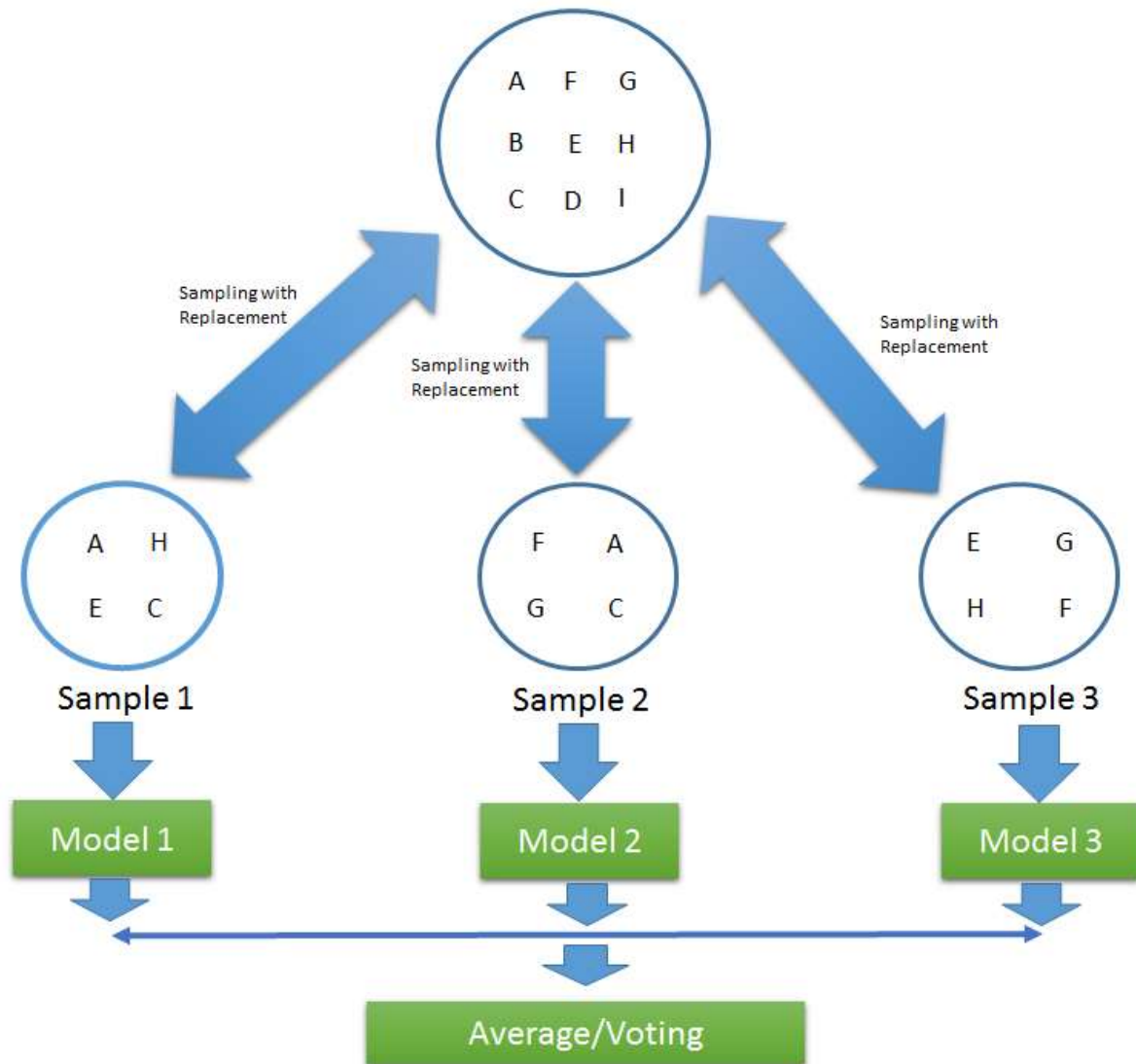
- Bootstrap sample is a statistic technique:
 - smaller sample of same size are repeatedly drawn, with replacement, from the larger original sample.
- Rationale behind:
 - The law of large numbers :
 - If you repeatedly sample again and again, it should approximate the large population.
 - Why replacement?
 - If the sample is a good representation of the large population, there will be more than one observation similar to another observation in the population

Bagging - Procedures of Bootstrapping

1. Choose a number of bootstrap samples to perform
2. Choose a sample size
3. For each bootstrap sample
 1. Draw a sample with replacement with the chosen size
 2. Calculate the statistic on the sample
4. Calculate the mean of the calculated sample statistics.

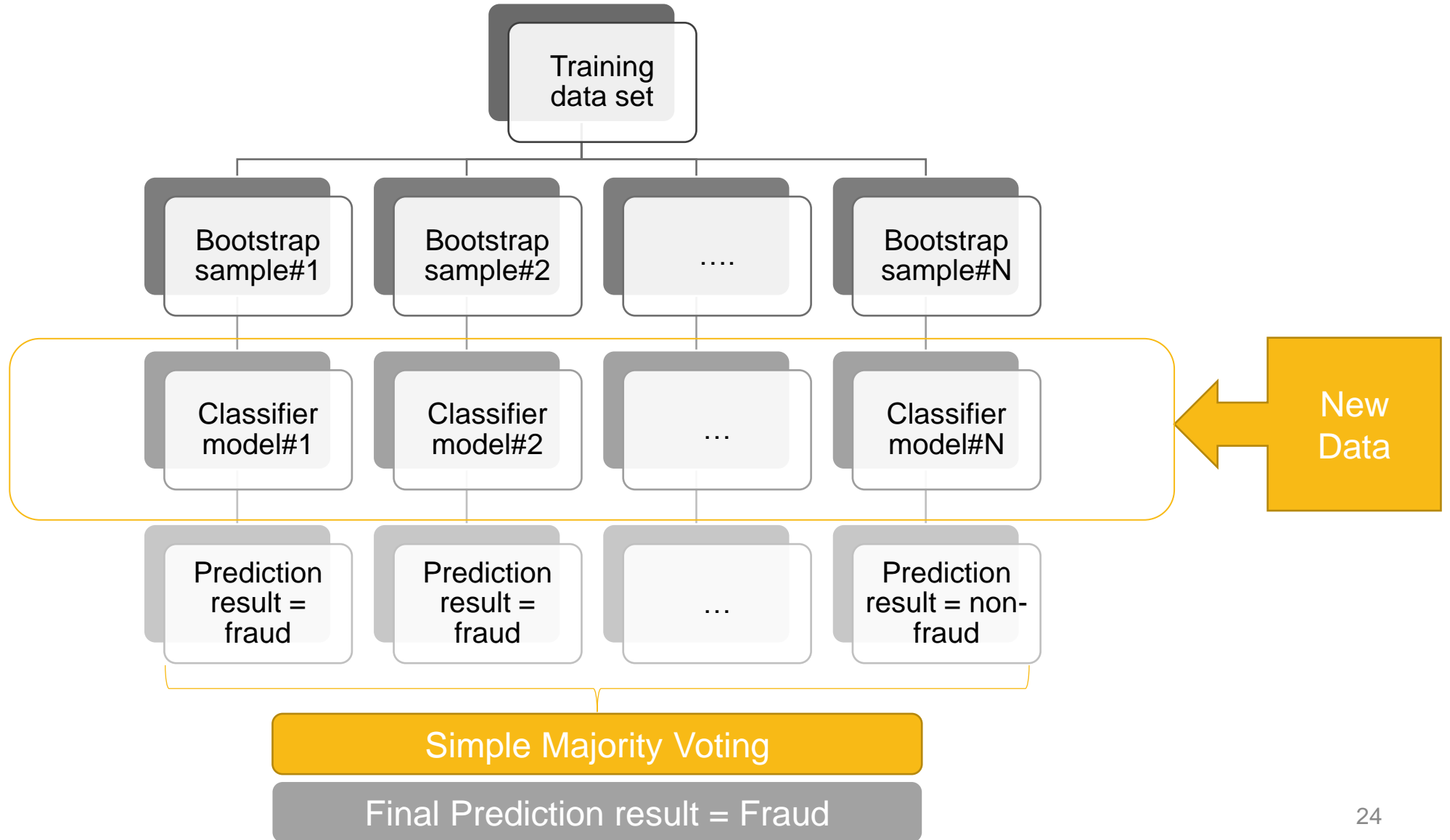


Bagging



- Bagging (Bootstrap aggregating) starts by taking N number of bootstraps from the underlying sample.
- The idea is to build a classifier for every bootstrap.
 - For classification : classified by letting all N classifiers vote and use a majority voting scheme
 - For regression : prediction by calculating the average of the outcome of the N prediction models

Bagging - example



A hand is visible on the left side of the frame, with the index finger pointing towards a bright, circular light spot on a dark, textured surface. The background is dark and out of focus, with a horizontal light band visible at the bottom.

03

Random Forest

Dr. Vivien CHAN

Decision Tree & Ensemble Learning (re-cap)

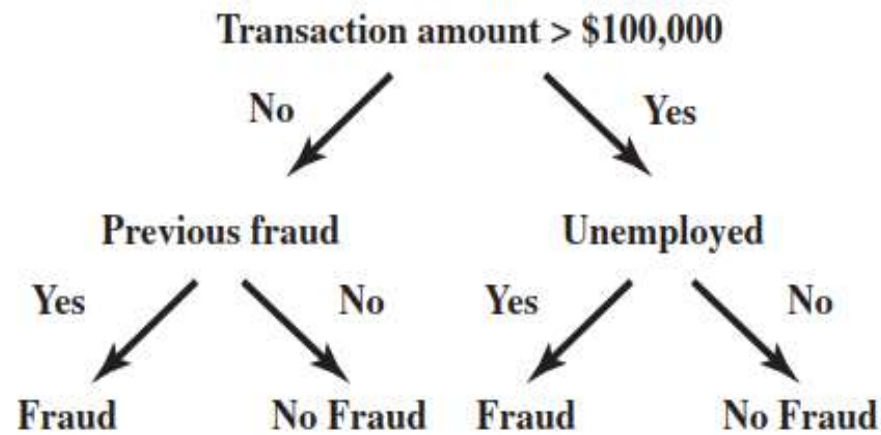
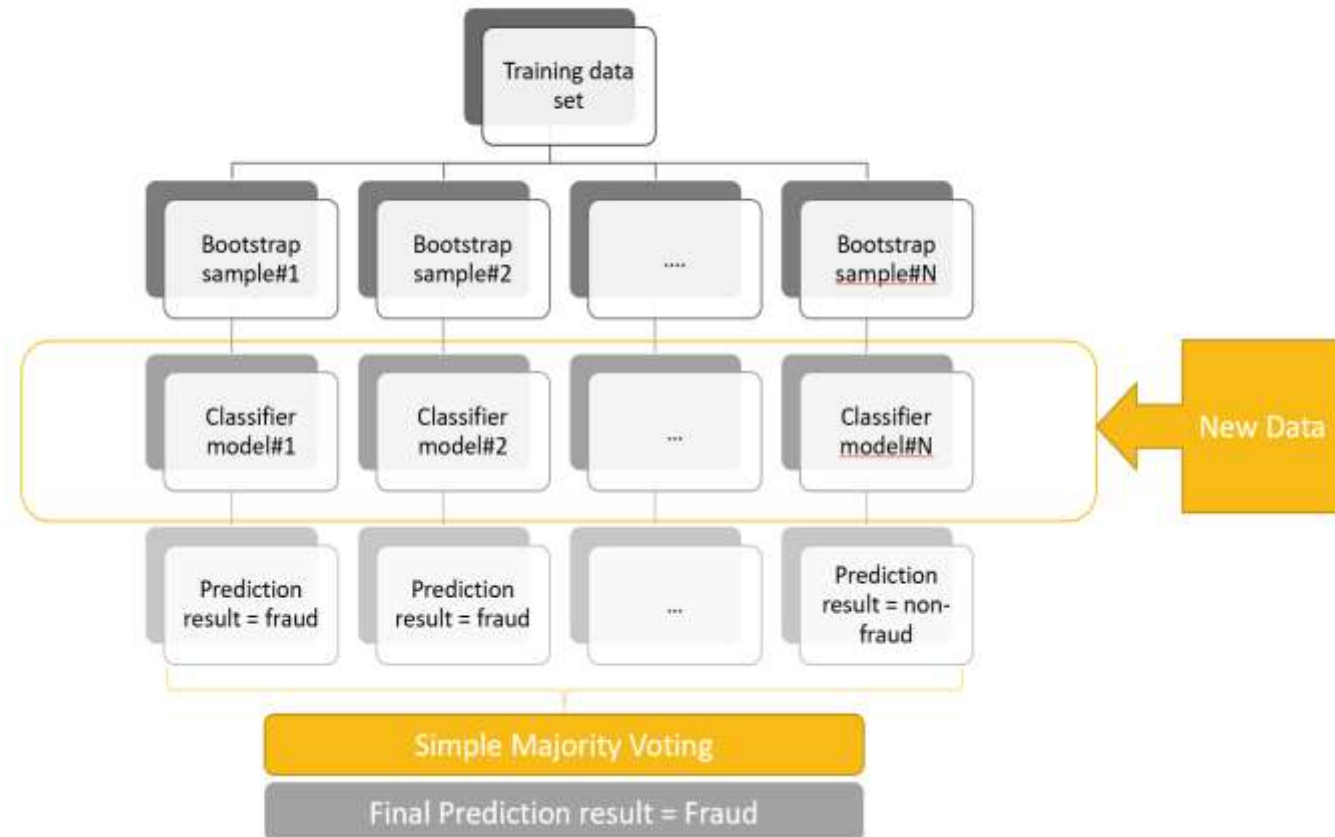


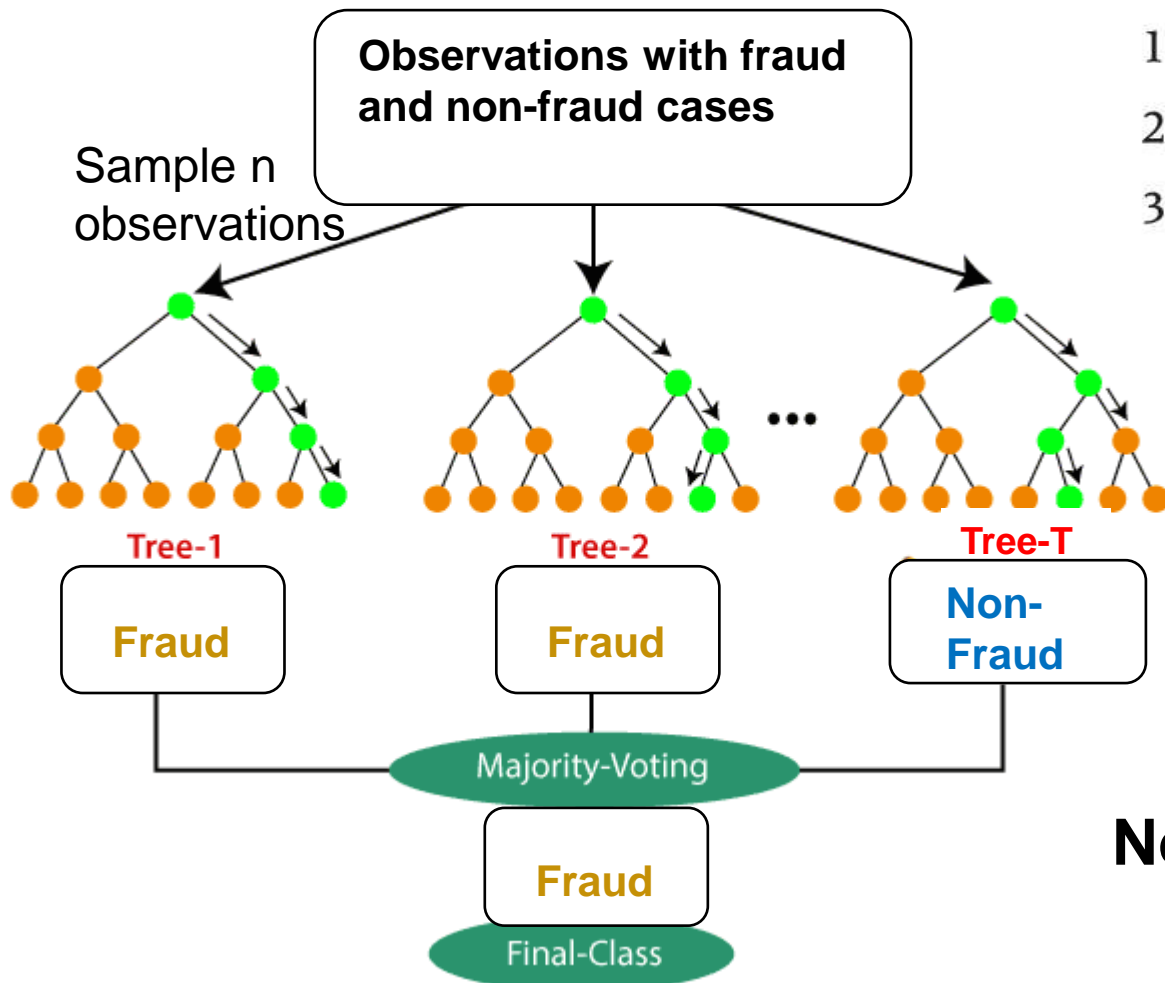
Figure 4.10 Example Decision Tree

Bagging - example



Random Forest

- Random Forest is a forest of Decision Trees



1. Given a data set with n observations and N inputs.
2. m = constant chosen on beforehand.
3. For $t = 1, \dots, T$
 - a. Take a bootstrap sample with n observations.
 - b. Build a decision tree whereby for each node of the tree, randomly choose m variables on which to base the splitting decision.
 - c. Split on the best of this subset.
 - d. Fully grow each tree without pruning.

Note: common choices of m is 1, 2 or $\log_2(N)+1$

Random Forest – Step1

- Step 1: Create a bootstrap sample from the original dataset
- Example: number of observations = 5, number of features/variables = 4 (i.e. $n=5$, $N=4$)

Original Dataset

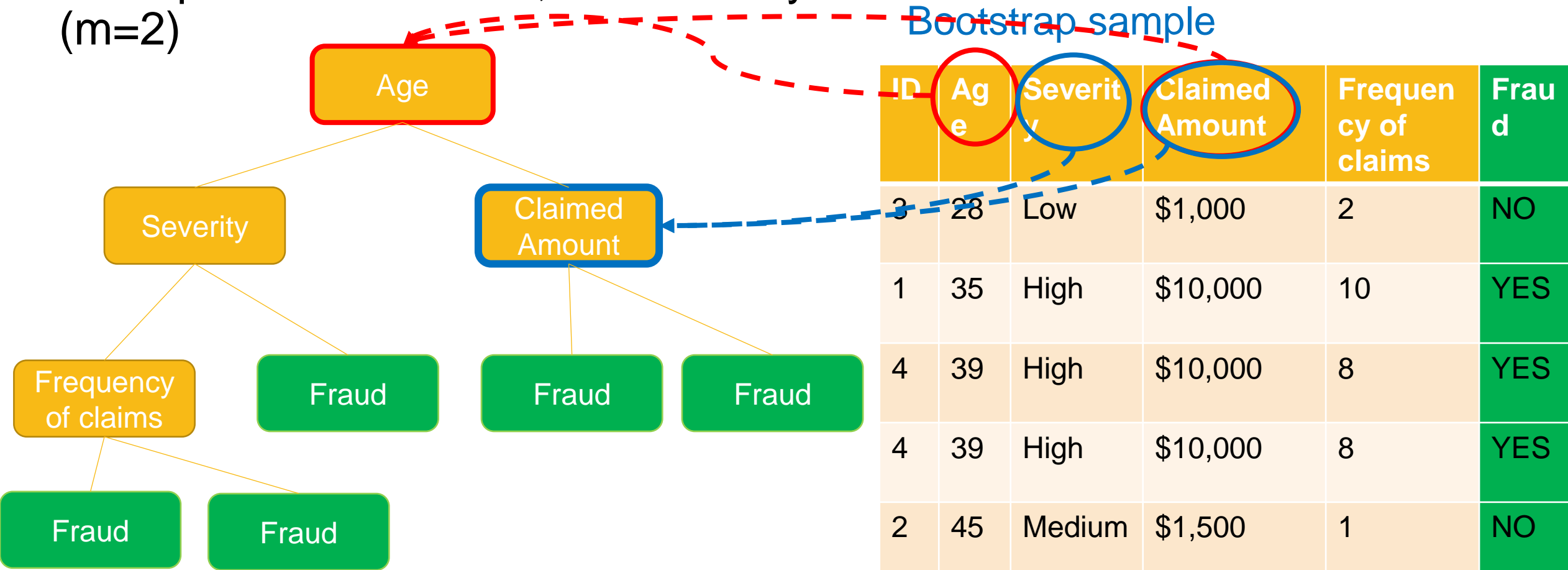
ID	Age	Severity	Claimed Amount	Frequency of claims	Fraud
1	35	High	\$10,000	10	YES
2	45	Medium	\$1,500	1	NO
3	28	Low	\$1,000	2	NO
4	39	High	\$10,000	8	YES
5	50	Low	\$1,000	1	NO

Bootstrap sample

ID	Age	Severity	Claimed Amount	Frequency of claims	Fraud
3	28	Low	\$1,000	2	NO
1	35	High	\$10,000	10	YES
4	39	High	\$10,000	8	YES
4	39	High	\$10,000	8	YES
2	45	Medium	\$1,500	1	NO

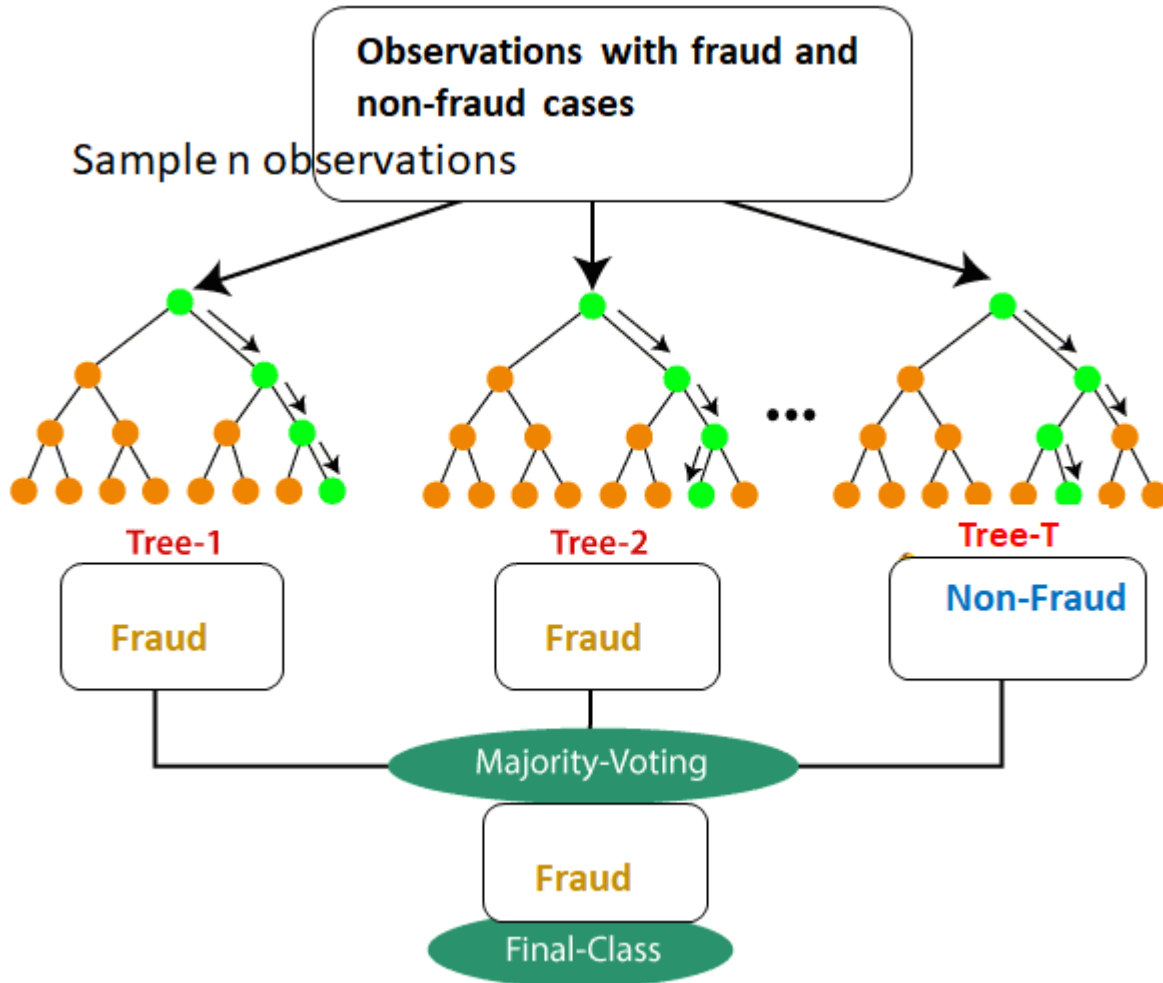
Random Forest – Step2

- Step 2: Create a decision tree with the bootstrap sample, BUT with a random subset of features/variables at each node
- Example: At each node, choose only 2 features at each node (m=2)



Random Forest – Step3

- Step 3: Repeats step 1-2 for T times
- Random forest can be used for both classification tree or regression tree



- Achieve dissimilarities among the decision trees by:
 - Adopting a bootstrap procedure to select training samples for each tree
 - Selecting a random subset of attributes at each node
 - Training different base models of decision trees
- Result of random forest is a model with better performance compared to a single decision tree model

Out-of-Bag (OOB) Sample

Original Dataset

ID	Age	Severity	Claimed Amount	Frequency of claims	Fraud
1	35	High	\$10,000	10	YES
2	45	Medium	\$1,500	1	NO
3	28	Low	\$1,000	2	NO
4	39	High	\$10,000	8	YES
5	50	Low	\$1,000	1	NO

Bootstrap sample

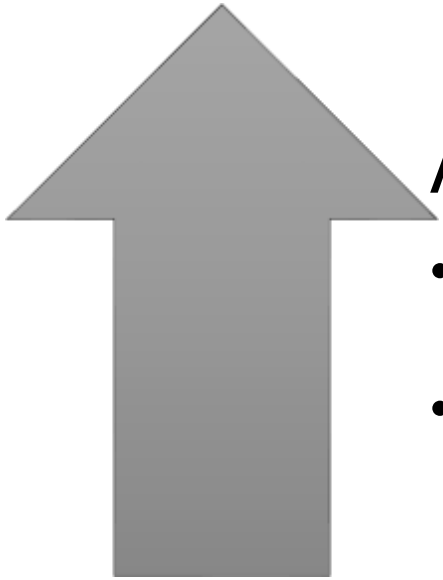
ID	Age	Severity	Claimed Amount	Frequency of claims	Fraud
3	28	Low	\$1,000	2	NO
1	35	High	\$10,000	10	YES
4	39	High	\$10,000	8	YES
4	39	High	\$10,000	8	YES
2	45	Medium	\$1,500	1	NO

- **Out-of-Bag (OOB)** samples are those observations not being selected for use in the bootstrap sample
- Each of the OOB sample rows is passed through every decision tree that did not contain the OOB sample row in its bootstrap training data and a majority prediction is noted for each row
- **OOB score** = the number of correctly predicted rows from OOB sample



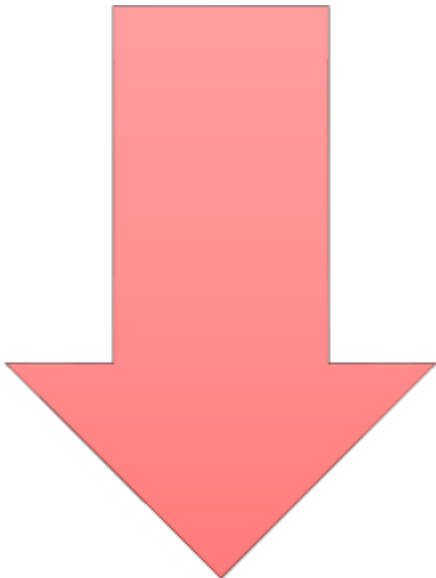
**Can we use OOB score
to evaluate the
performance of the
Random Forest model?
When is OOB score be
useful?**

Random Forest



Advantages

- Random Forest can achieve excellent predictive performance and suitable for the requirements of fraud detection
- Capable of dealing with data sets having only a few observations, but with lots of variables



Disadvantages

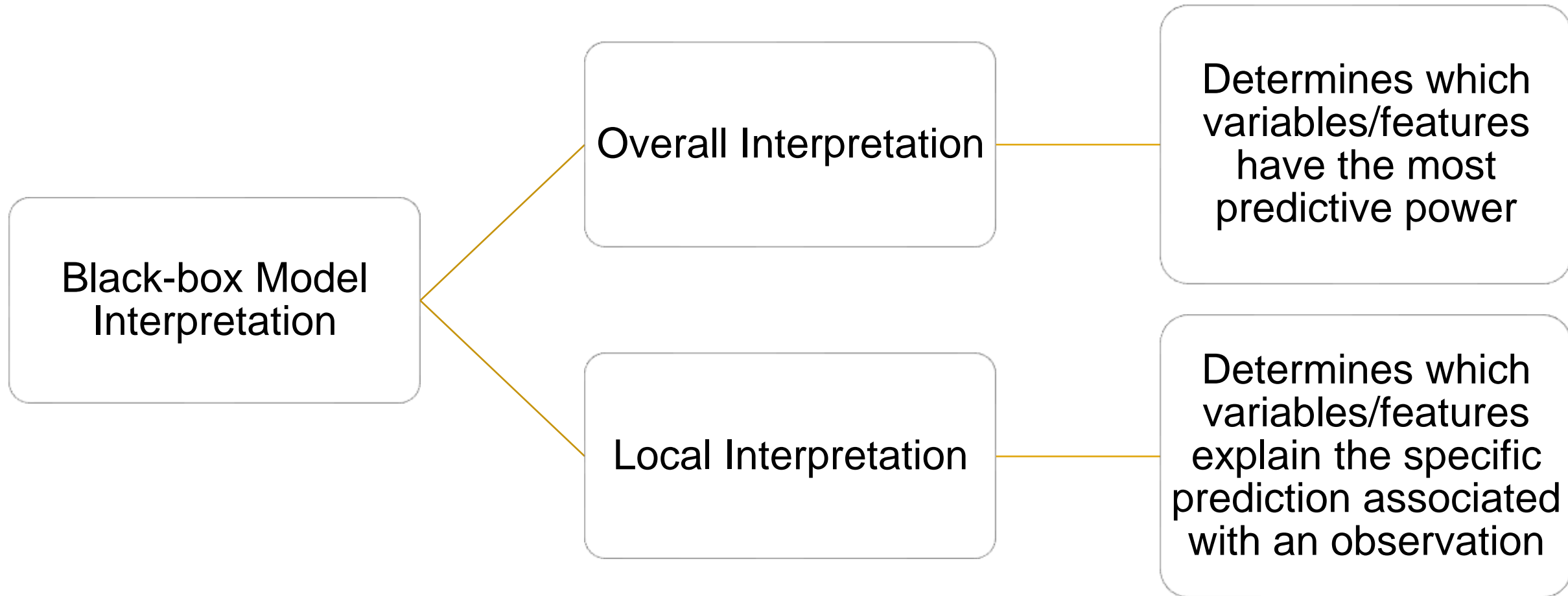
- It is a black-box model.
 - **Solution:** Variable importance can be used to understand the internal workings of Random Forest (or any ensemble model)

A hand is visible on the left side of the frame, with the index finger pointing towards the center. The background is a dark, out-of-focus presentation screen with some light streaks.

How to interpret Random Forest

Dr. Vivien CHAN

How to interpret a “black-box” model?



Overall Interpretation

- Variable/Feature Importance

Importance	Variable
10%	Var_5
8%	Var_1
6%	Var_2
3%	Var_8
3%	Var_6
2%	Var_22
1%	Var_13

Pic source:
https://miro.medium.com/max/324/1*k82O3uoBlO7ymkJ12pMQQ.png

- Variables with HIGH importance can be used for further analysis, while variables with LOW importance can be discarded
- Variables with TOO much importance compared to others might be bugs in the data or model
- Can be an indication of whether a “new” variable is relevant or useful to your model, e.g. compare the variable importance for the model with and without the new variables
- Compare variable importance for different models

Local Interpretation

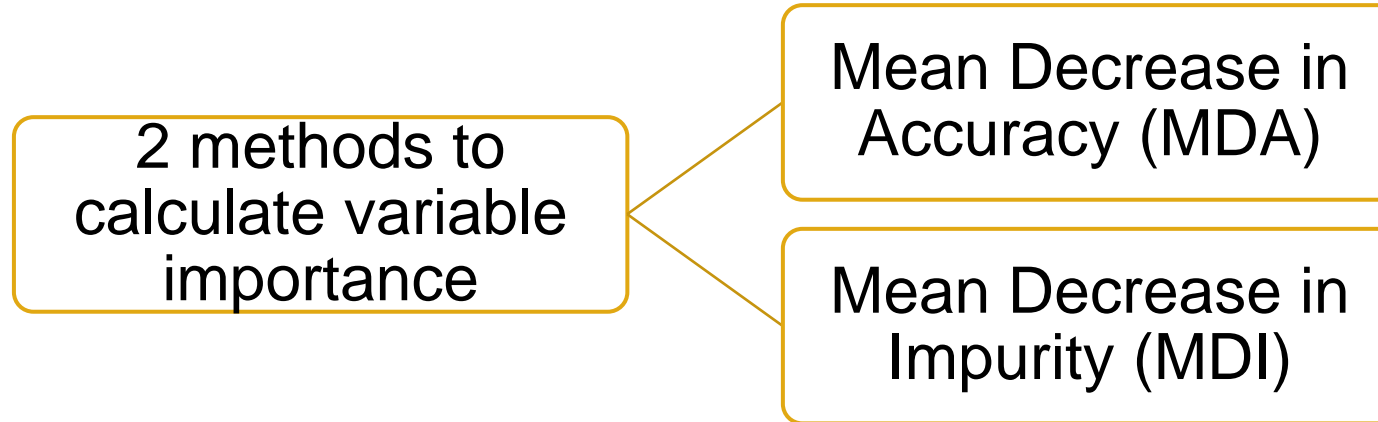
- To understand individual observations, i.e. reasons of the prediction, or which variables explains/predicts more for this specific observation
- To understand the most frequent reasons of the predictions. This might be particularly useful when you have an imbalanced dataset.

ID	Probability of default	Class predicted	3 variables with most contribution					
			1st variable		2nd variable		3rd variable	
			Name	Impact	Name	Impact	Name	Impact
12098321	95%	1	Var_1	+34%	Var_3	+19%	Var_9	+8%
12098322	88%	1	Var_1	+25%	Var_8	+14%	Var_3	+13%
12098323	35%	0	Var_7	-27%	Var_5	-23%	Var_1	-12%

	Frequency of being in the top 3 variables			Total
	1st	2nd	3rd	
Var_1	43	15	12	70
Var_3	11	24	30	65
Var_8	10	8	12	30
Var_9	2	4	45	51

Pic source: https://miro.medium.com/max/1400/1*joBp6qhBfXCE8c_9q2F71Q.png

How to calculate “Variable Importance”?



- Related package in R
 - `install.packages('caret')` *#package name*
 - `library(caret)` *#library name*
 - `varImp(object, ...)` *# function to list variable importance*
 - `varImpPlot` *# function to plot variable importance for Random Forest model*



Variable Importance not only applicable to Random Forrest, also applicable to other ensemble model

Variable Importance - MDI

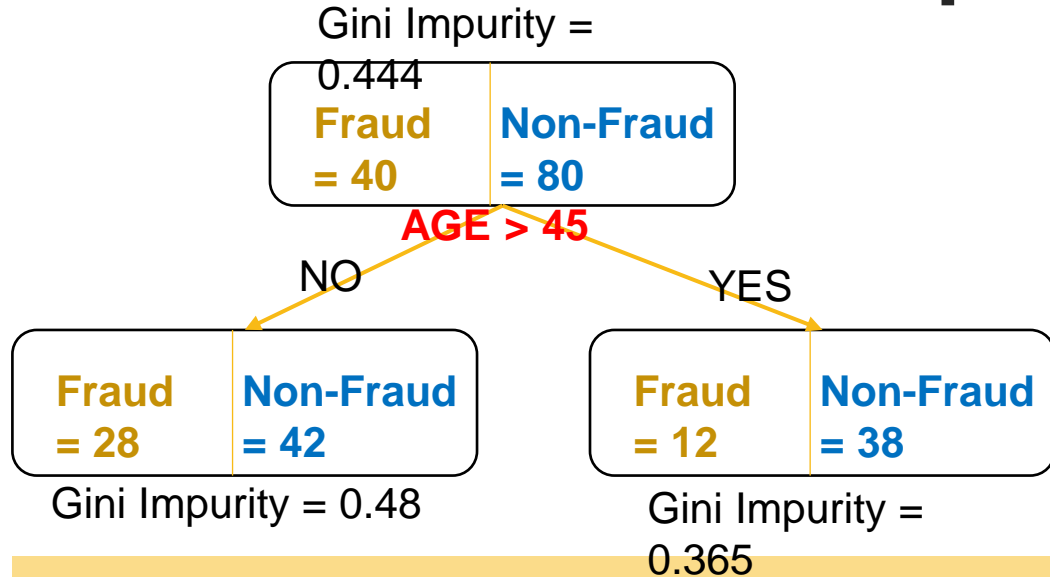
- Mean Decrease Impurity (MDI)
 - Measures the decrease in Gini impurity
 - What is Gini impurity for binary classification?
 - **Gini Impurity** is a measurement of the likelihood of an incorrect classification of a new instance of a random variable, if that new instance were randomly classified according to the distribution of class labels from the data set.

$$I_G = 1 - \sum_{j=1}^c p_j^2$$

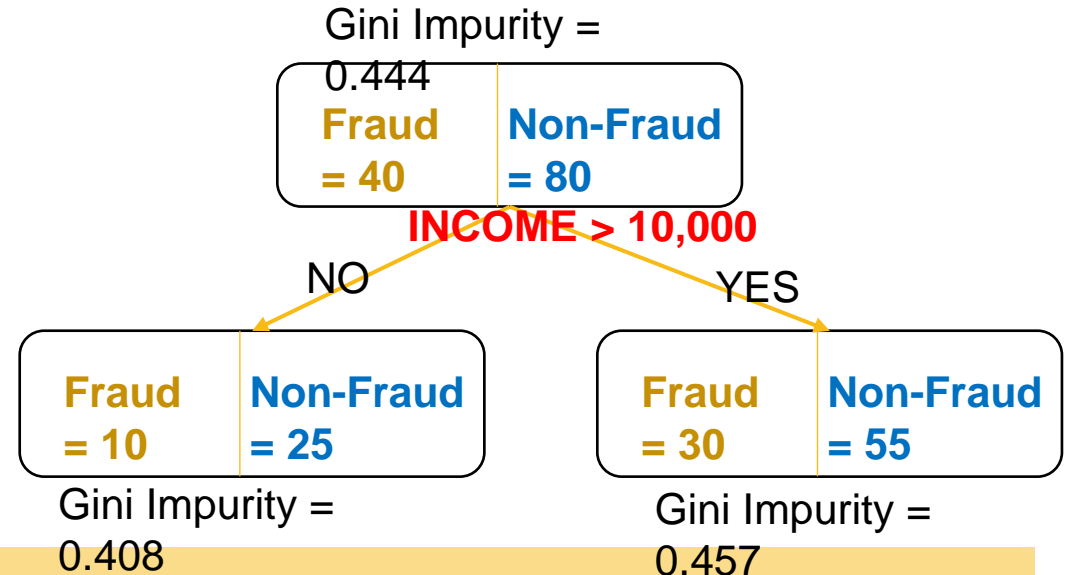
where c = number of class labels, p = proportions of samples belonging to class c in the node

Thus, for binary classification, $I_G = 1 - (p_1^2 + p_2^2)$

Gini Impurity - example



Gini impurity of "AGE" split
= weighted average of leaf nodes
= $70/120 \times 0.48 + 50/120 \times 0.365$
= 0.432



Gini impurity of "INCOME" split
= weighted average of leaf nodes
= $35/120 \times 0.408 + 85/120 \times 0.457$
= 0.443

- Compare the Gini Impurity for splitting decision using the attribute "AGE" and "INCOME"
- Gini Impurity(**AGE**) < Gini Impurity(**INCOME**)
=> use **AGE** as the splitting attribute

Variable Importance - MDI

- Mean Decrease Impurity (MDI)
 - Measures how effective the feature is at reducing impurity, or the total decrease in node impurity averaged over all trees of the ensemble
 - Counts the times a feature is used to split a node, weighted by the number of samples it splits
 - At each split in each tree, the improvement in the split-criterion is the importance measure attributed to the splitting variable, and is accumulated over all the trees in the forest separately for each variable
- Problems with MDI
 - Computationally efficient BUT biased towards continuous and high cardinality variables

Variable Importance - MDA

- Mean Decrease in Accuracy (MDA)
 - Measures the decrease in accuracy (Permutation Importance)
 - Steps of calculating MDA:
 - Use the out-of-bag (OOB) samples to construct variable importance, to measure the prediction strength of each variable
 - When the b^{th} tree is grown, the OOB samples are passed down the tree, and the prediction accuracy is recorded
 - Values for the j^{th} variable are randomly permuted in the OOB samples, and the accuracy is again computed
 - The decrease in accuracy as a result of this permutation is averaged over all trees, and is used as measure of the importance of variable j in the random forest.

Variable Importance

1. Permute the values of the variable under consideration (e.g., X_j) on the validation or test set.
2. For each tree, calculate the difference between the error on the original, unpermuted data and the error on the data with X_j permuted as follows:

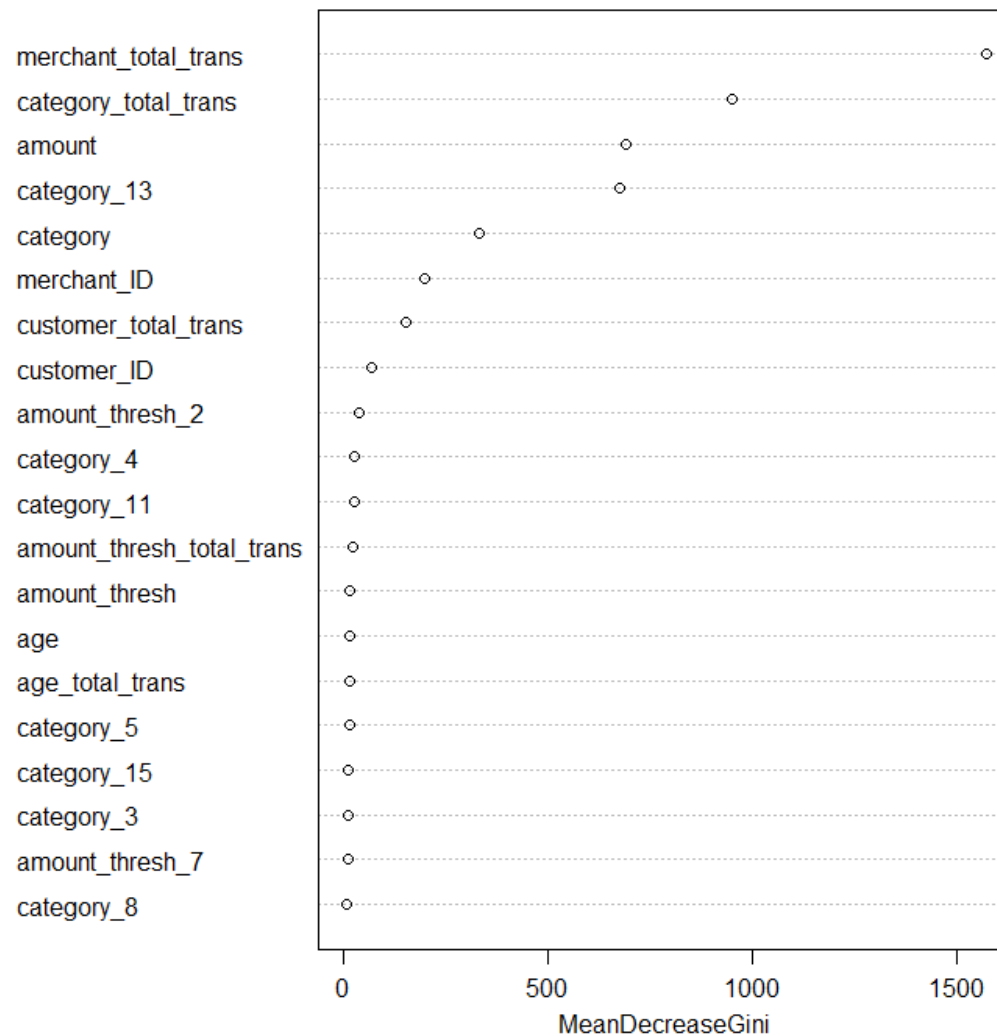
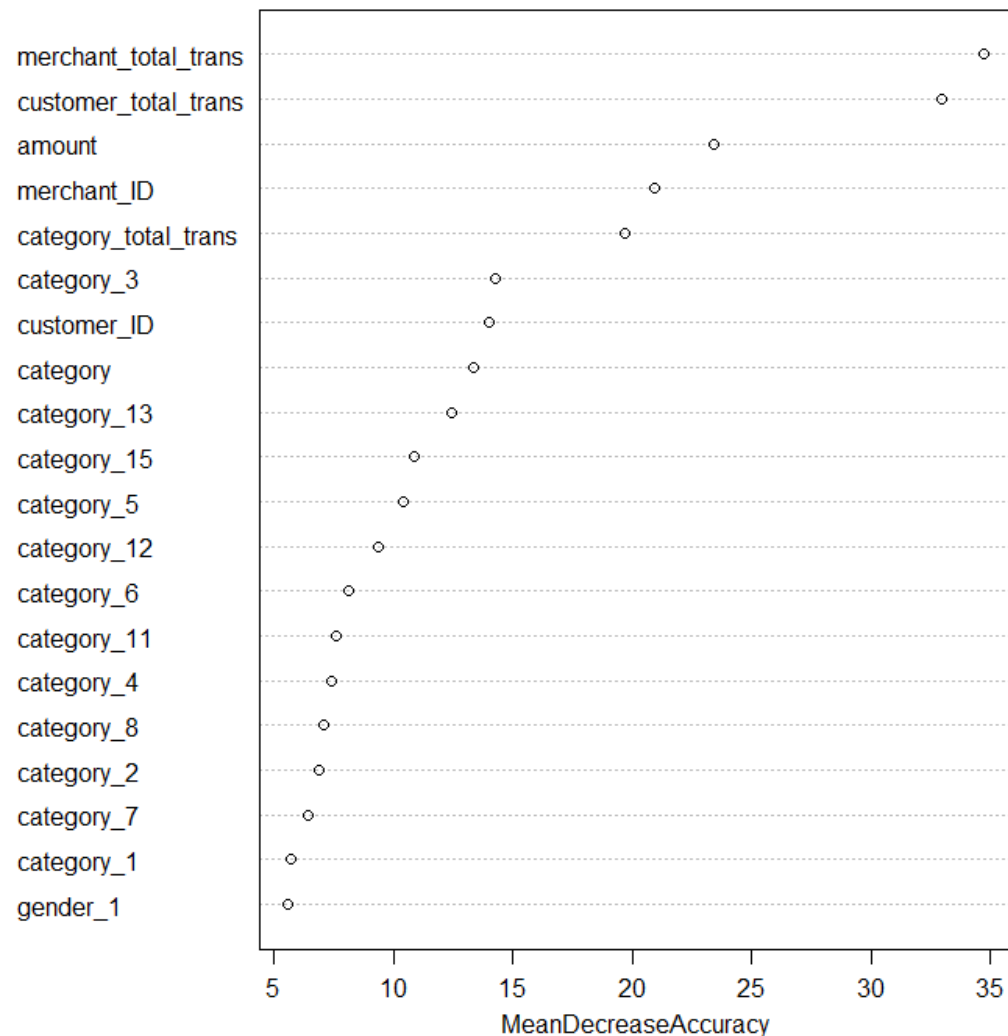
$$VI(X_j) = -\frac{1}{ntree} \sum_t (error_t(D) - error_t(\tilde{D}_j))$$

whereby *ntree* represents the number of trees in the ensemble, D the original data, and \tilde{D}_j the data with variable X_j permuted. In a regression setting, the error can be the mean squared error (MSE), whereas in a classification setting, the error can be the misclassification rate.

3. Order all variables according to their VI value. The variable with the highest VI value is the most important.

Variable Importance – example in R

model_rf_m12





Using R to build Random Forest

Dr. Vivien CHAN

- `library(tidyverse)` # for data manipulation
- `library(caret)` # for models
- `library(ROSE)` #for over-/under-sampling function
- `library(precrec)` #for Precision-Recall function
- `library(randomForest)`
- `library(NeuralNetTools)`
- `library(neuralnet)`

Example of EDA using R

- Dataset : <https://www.kaggle.com/mlg-ulb/creditcardfraud>
- The Credit Card Fraud Detection Dataset comprises transactions that European credit card holders made in September 2013. The dataset shows transactions that occurred in two days.
- The dataset has been collected and analyzed during a research collaboration of Worldline and the Machine Learning Group (<http://mlg.ulb.ac.be>) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

Random Forest

Preparing data for Random Forest

1. Imbalance Data Set
2. Random forest is affected by multicollinearity but not by outlier problem.
3. Impute missing values, if needed
4. Prepare the class label
 - Use the function “`as.factor`” to convert the class label to factor

```
cc_data <- read.csv('../input/creditcardfraud/creditcard.csv')
```

```
str(cc_data)
table(cc_data$Class)
```

	0	1
	284315	492

```
'data.frame': 284807 obs. of 31 variables:
 $ Time : num 0 0 1 1 2 2 4 7 7 9 ...
 $ V1 : num -1.36 1.192 -1.358 -0.966 -1.158 ...
 $ V2 : num -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
 $ V3 : num 2.536 0.166 1.773 1.793 1.549 ...
 $ V4 : num 1.378 0.448 0.38 -0.863 0.403 ...
 $ V5 : num -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...
 $ V6 : num 0.4624 -0.0824 1.8005 1.2472 0.0959 ...
 $ V7 : num 0.2396 -0.0788 0.7915 0.2376 0.5929 ...
 $ V8 : num 0.0987 0.0851 0.2477 0.3774 -0.2705 ...
 $ V9 : num 0.364 -0.255 -1.515 -1.387 0.818 ...
 $ V10 : num 0.0908 -0.167 0.2076 -0.055 0.7531 ...
 $ V11 : num -0.552 1.613 0.625 -0.226 -0.823 ...
 $ V12 : num -0.6178 1.0652 0.0661 0.1782 0.5382 ...
 $ V13 : num -0.991 0.489 0.717 0.508 1.346 ...
 $ V14 : num -0.311 -0.144 -0.166 -0.288 -1.12 ...
 $ V15 : num 1.468 0.636 2.346 -0.631 0.175 ...
 $ V16 : num -0.47 0.464 -2.89 -1.06 -0.451 ...
 $ V17 : num 0.208 -0.115 1.11 -0.684 -0.237 ...
 $ V18 : num 0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...
 $ V19 : num 0.404 -0.146 -2.262 -1.233 0.803 ...
 $ V20 : num 0.2514 -0.0691 0.525 -0.208 0.4085 ...
 $ V21 : num -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...
 $ V22 : num 0.27784 -0.63867 0.77168 0.00527 0.79828 ...
 $ V23 : num -0.11 0.101 0.909 -0.19 -0.137 ...
 $ V24 : num 0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...
 $ V25 : num 0.129 0.167 -0.328 0.647 -0.206 ...
 $ V26 : num -0.189 0.126 -0.139 -0.222 0.502 ...
 $ V27 : num 0.13356 -0.00898 -0.05535 0.06272 0.21942 ...
 $ V28 : num -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
 $ Amount: num 149.62 2.69 378.66 123.5 69.99 ...
 $ Class : int 0 0 0 0 0 0 0 0 0 0 ...
```

Step 1: Splitting the dataset into TRAINING and TESTING datasets

```
# convert class to factor for data modelling
cc_data$Class <- as.factor(cc_data$Class)
```

```
cc_data <- subset(cc_data, select = -c(Time))
#split the data into a training set and test set.
train_index <- createDataPartition(cc_data$Class, times = 1, p = 0.8, list = F)
cc_train <- cc_data[train_index,]
cc_test <- cc_data[-train_index,]
```

Remove attribute
"Time"

Split by
80%

	0	1
227452		394
	0	1
56863		98

```
# generating data using under-sampling
set.seed(123)
train_un <- ovun.sample(Class ~., data=cc_train, p=0.5, method="under")$data
table(train_un$Class)
```

	0	1
385		394

Step 3: Training the Random Forest data model

To include permutation importance in the “Variable Important” list

```
# train a random forest model
```

```
model_rf <- randomForest(Class ~., data=train_un, proximity=TRUE, importance=TRUE)  
model_rf
```

Call:

```
randomForest(formula = Class ~ ., data = train_un, proximity = TRUE, importance = TRUE)
```

```
  Type of random forest: classification
```

```
    Number of trees: 500
```

```
No. of variables tried at each split: 5
```

- 1.Number of trees used in the forest (ntree) and
- 2.Number of random variables used in each tree (mtry)

```
  OOB estimate of  error rate: 6.55%
```

```
Confusion matrix:
```

	0	1	class.error
0	373	12	0.03116883
1	39	355	0.09898477

OOB = Out of Bag error = Misclassification rate



What are the Precision, Recall & F-score?

Step 4a: Checking performance of the trained Random Forest data model

```
# evaluate performance of the trained model
rf_pred <- predict(model_rf, newdata = cc_test)
# show confusion matrix
confusionMatrix(data=rf_pred, reference=cc_test$Class, positive='1')
```

Confusion Matrix and Statistics

Prediction \ Reference	0	1
0	55299	9
1	1564	89

Accuracy : 0.9724
95% CI : (0.971, 0.9737)
No Information Rate : 0.9983
P-Value [Acc > NIR] : 1

Kappa : 0.0987

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.908163

Specificity : 0.972495

Pos Pred Value : 0.053842

Neg Pred Value : 0.999837

Prevalence : 0.001720

Detection Rate : 0.001562

Detection Prevalence : 0.029020

Balanced Accuracy : 0.940329

'Positive' Class : 1

Confusion Matrix

What does it mean when the performance results of training data is better than testing data?

Sensitivity = Recall

Pos Pred Value = Precision

F-score

$= 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$
 $= 0.10166$

Step 4b: Checking performance of the trained Random Forest data model

```
rf_pred <- as.numeric(rf_pred)
rf_prc <- evalmod(scores=rf_pred, labels = cc_test$class, mode="rocprc")
rf_prc
```

=== AUCs ===

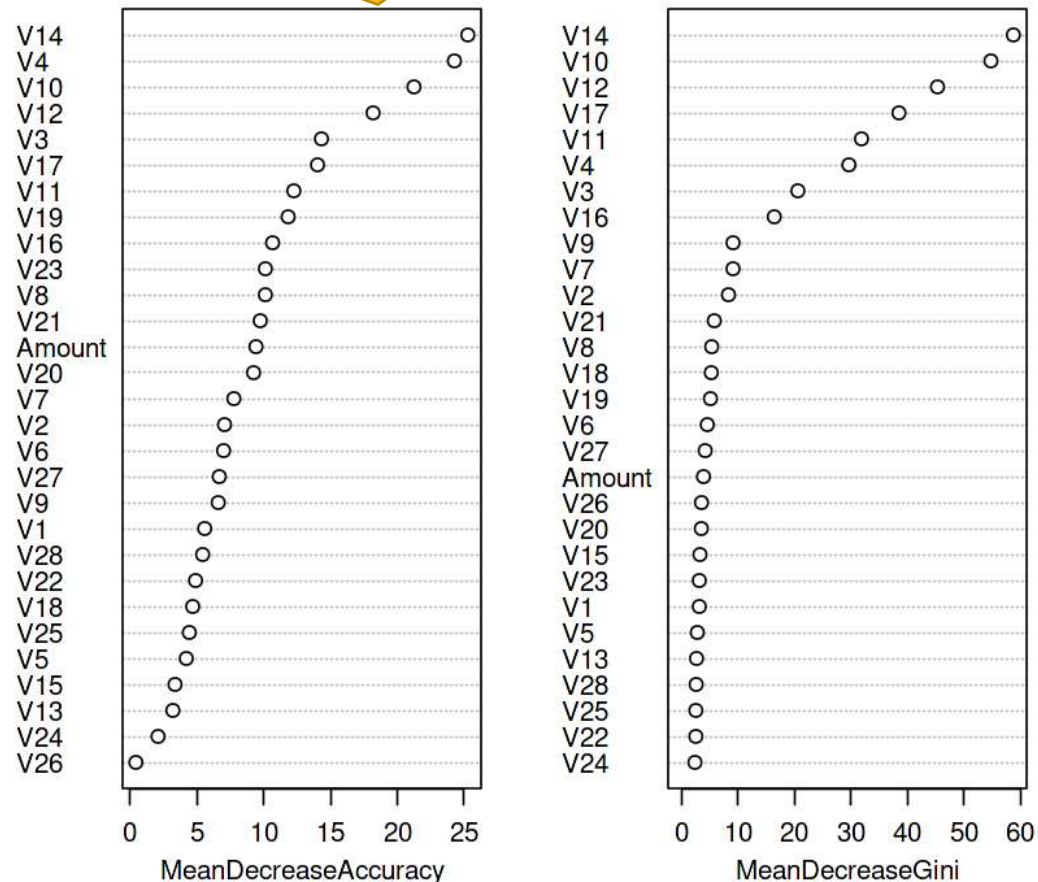
	Model name	Dataset ID	Curve type	AUC
1	m1	1	ROC	0.94032928
2	m1	1	PRC	0.04943335

=== Input data ===

	Model name	Dataset ID	# of negatives	# of positives
1	m1	1	56863	98

Step 5: Review Variable Importance

```
#show variable importance
importance(model_rf)
importance
varImpPlot(model_rf)
```



	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
V1	4.5921203	4.9468175	6.8575435	3.384981
V2	4.9936066	3.5285666	6.0635440	7.698307
V3	11.4013167	7.6939713	14.2903230	23.200712
V4	22.0438836	9.1942531	22.2793219	28.594252
V5	1.9578722	5.8621770	4.8517067	3.353066
V6	6.4942892	-0.4553901	6.4549112	3.515288
V7	7.2061349	2.3691728	8.2050448	12.683104
V8	5.8901841	5.8085948	7.7555559	3.962754
V9	5.9158525	2.8073788	7.0619939	7.353252
V10	17.6312119	8.0421629	18.2662423	47.708067
V11	12.7073988	7.0996781	14.7342521	32.112594
V12	14.7916614	10.6411195	16.4907279	43.438771
V13	3.0288844	1.9288234	3.5824662	2.835639
V14	24.8792319	11.9051437	24.5650494	72.838746
V15	0.2935782	2.0394105	1.2262737	2.369947
V16	8.1946868	4.8136973	9.5968023	11.628665
V17	12.7287641	7.1709769	13.5183432	35.111645
V18	3.0608862	6.1585267	6.2794732	4.965497
V19	11.5277185	6.4099497	13.3456113	6.618755
V20	11.7159107	4.8828517	12.0972327	3.933056
V21	8.7902857	1.4803732	9.4574618	5.277938
V22	6.9216574	3.2925022	7.1118904	2.913111
V23	11.2971544	0.5721250	11.0657407	3.321971
V24	-0.6701646	2.7083163	0.8611018	1.957743
V25	5.0715498	2.7915454	5.7152046	2.923475
V26	2.4965037	5.0403454	4.9019240	2.981296
V27	6.6950725	2.6927159	7.5506396	3.758955
V28	2.8730883	2.2119555	3.6873580	2.192275
Amount	13.0684958	4.1389750	13.0962399	6.287708



04 Neural Network

Dr. Vivien CHAN

ANN - Basic Concepts

- Artificial Neural Networks (ANN)
 - Deep learning was conceptualized by Geoffrey Hinton in the 1980s. He created the concept of “neural network”
 - Mathematical representations of the human brain, designed to recognize patterns
 - Can be used to perform tasks like classification, clustering, predictive analytics
 - Generalization of existing statistical models (Bishop 1995; Zurada 1992)

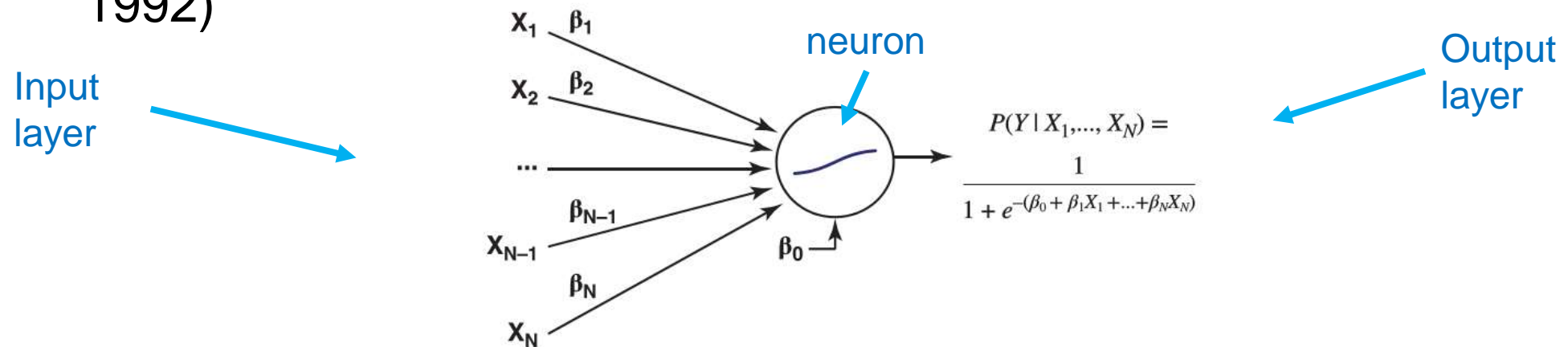
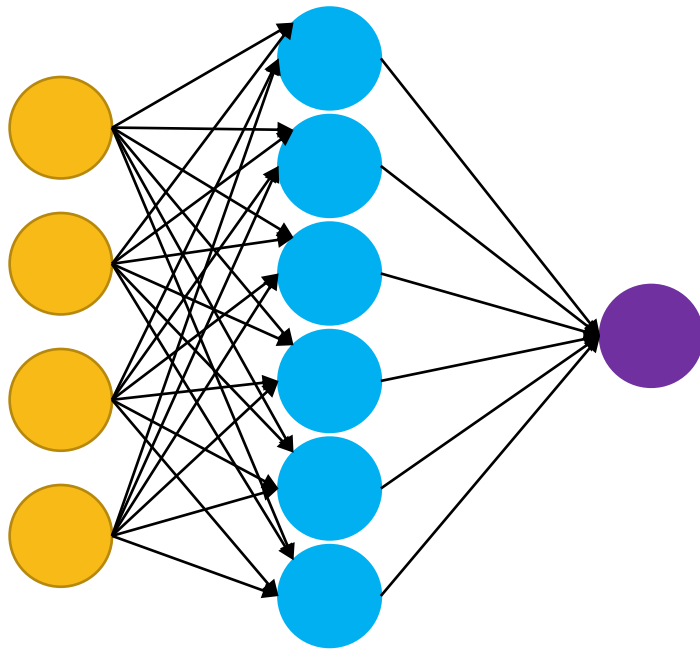


Figure 4.17 Neural Network Representation of Logistic Regression

ANN - Basic Concepts



Input layer

Hidden layer

Output layer

- Artificial neural networks (ANN) are composed of layers of **node/neurons**. Neural networks with more than one hidden layer are called deep neural networks (DNN)
- **Input layer** – initial input data for the neural network (visible layer)
- **Hidden layers** – between input and output layers, where the computations are done and passed on to other nodes deeper in the neural network
- **Output layer** – output result of the given input (visible layer)

ANN

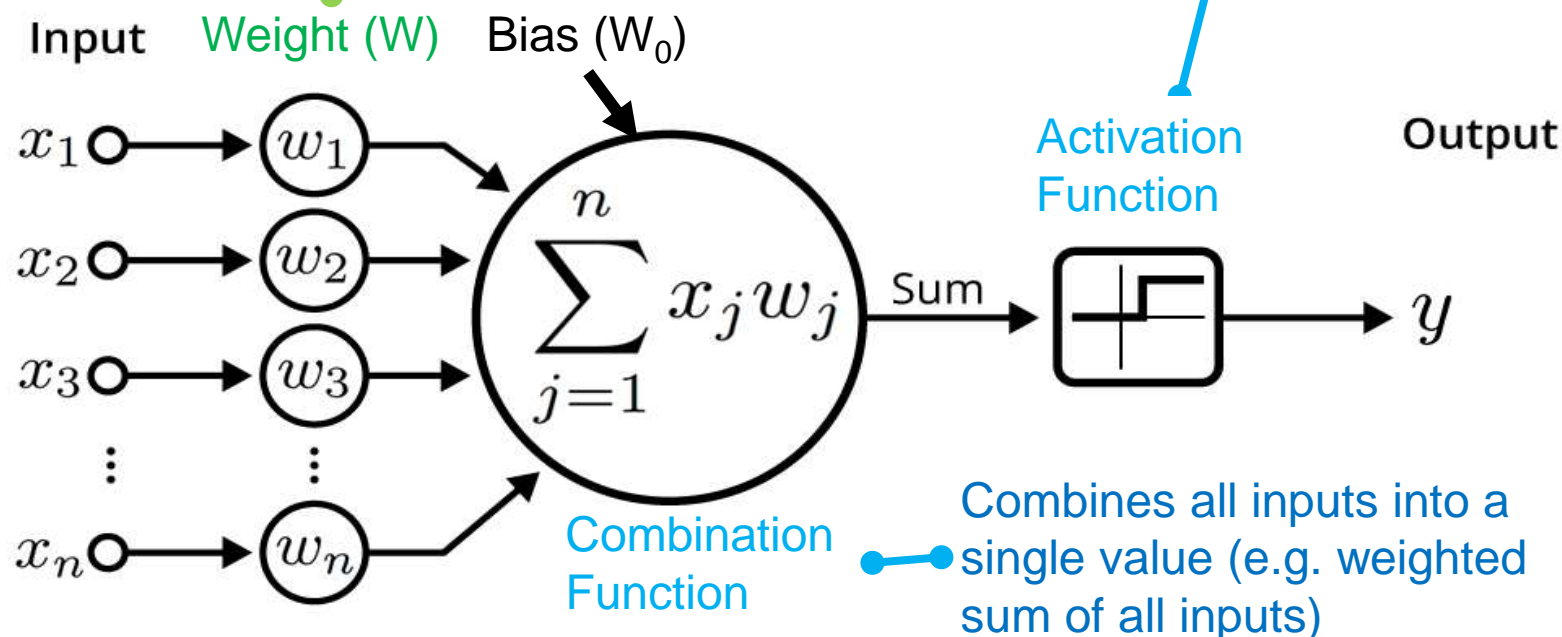
- Feedforward ANN
 - Not recursive
 - Neurons were only connected to neurons in the next layer, don't form a cycle
 - Signals travel in only one direction towards the output layer
- Feedback ANN
 - Recursive (also known as Recurrent Neural Network)
 - Contains cycles, the feedback cycles can cause the network's behaviour change over time based on its input
 - Signals travel in both directions by introducing loops in the network

Inside an artificial neuron

Weight (W) - impacts the preceding neuron's importance in the overall neural network

Transforms value of combination function to output value (e.g. linear, sigmoid, tanh, etc)

These input can come from either the raw data set or from neurons positioned at a previous layer of the neural network.



Output value is then passed on to the next layer of the neural network

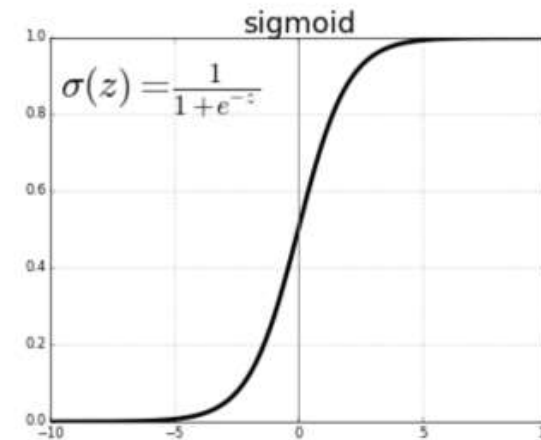
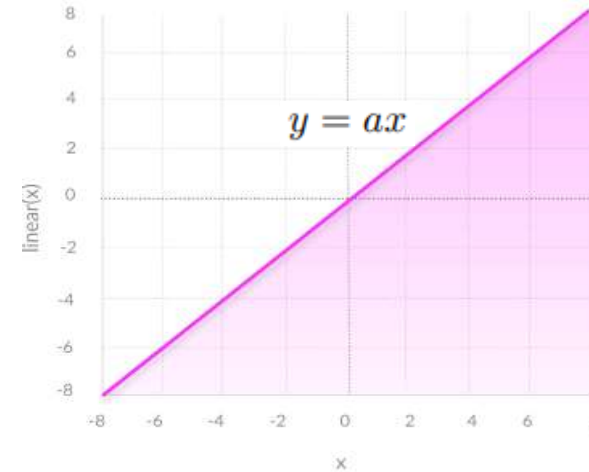
An illustration of an artificial neuron. Source: Becoming Human.

Activation functions

- What is Activation function?
- Mathematical equations that determine the output of a neural network
- Attached to each node and defines if the given node should be “activated”, based on whether each node’s input is relevant for the model’s prediction
- Normalize the output of each neuron to a range between 1 and 0 or between -1 and 1

Activation functions

- Examples of activation functions
 - Linear activation functions:
 - Threshold (unit step or sign) function
 - Linear function
 - ReLU function
 - Non-linear activation functions:
 - Sigmoid function
 - tanh (hyperbolic tangent) function
 - Softplus function

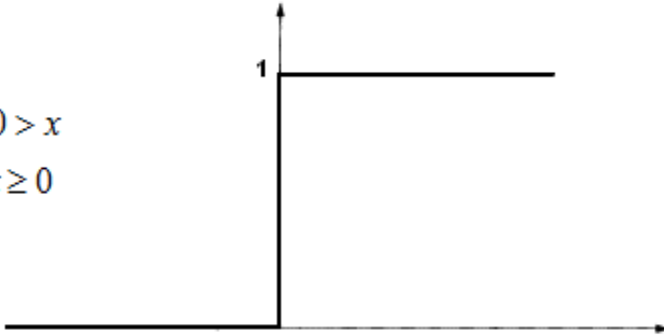


Activation function - linear

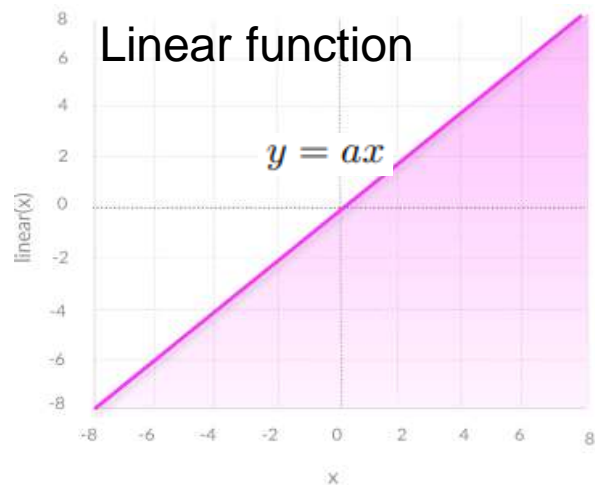
Threshold function

Unit step (threshold)

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$

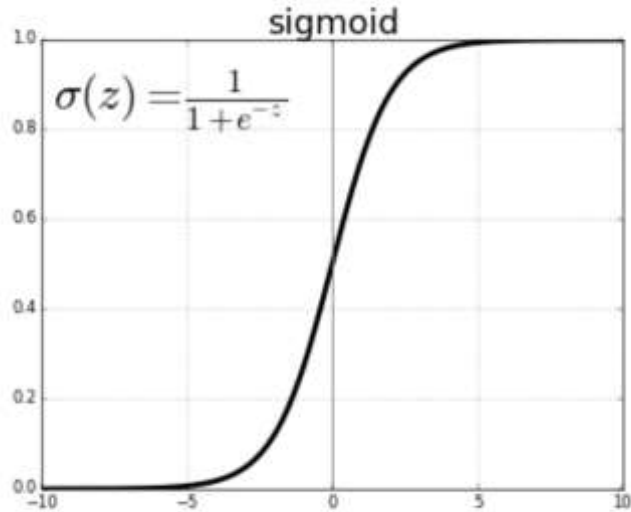


- Problems with linear functions
 - The problem with a threshold function is that it does not allow multi-value outputs—for example, it can only classify output as 0 or 1
 - The problem with linear function is that it transforms the output of the neural network into a linear function. Thus, unable to cater for non-linear data.



Activation function – non-linear

Sigmoid function

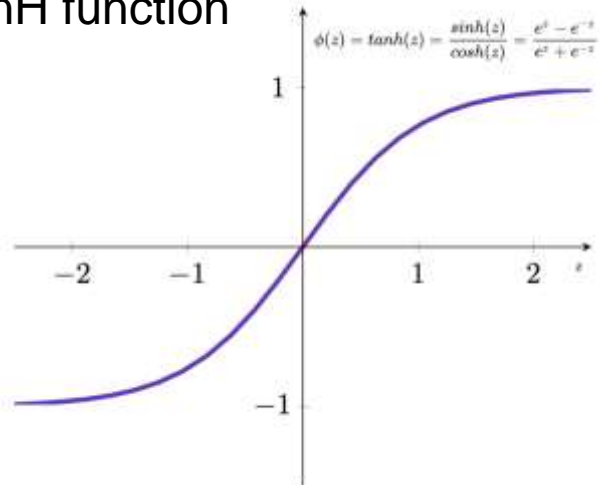


- Most neural networks use non-linear activation functions to for more complex mappings between inputs and outputs

- Sigmoid function

- Can accept any value and normalize output between 0 to 1
 - Smooth gradient and clear predictions

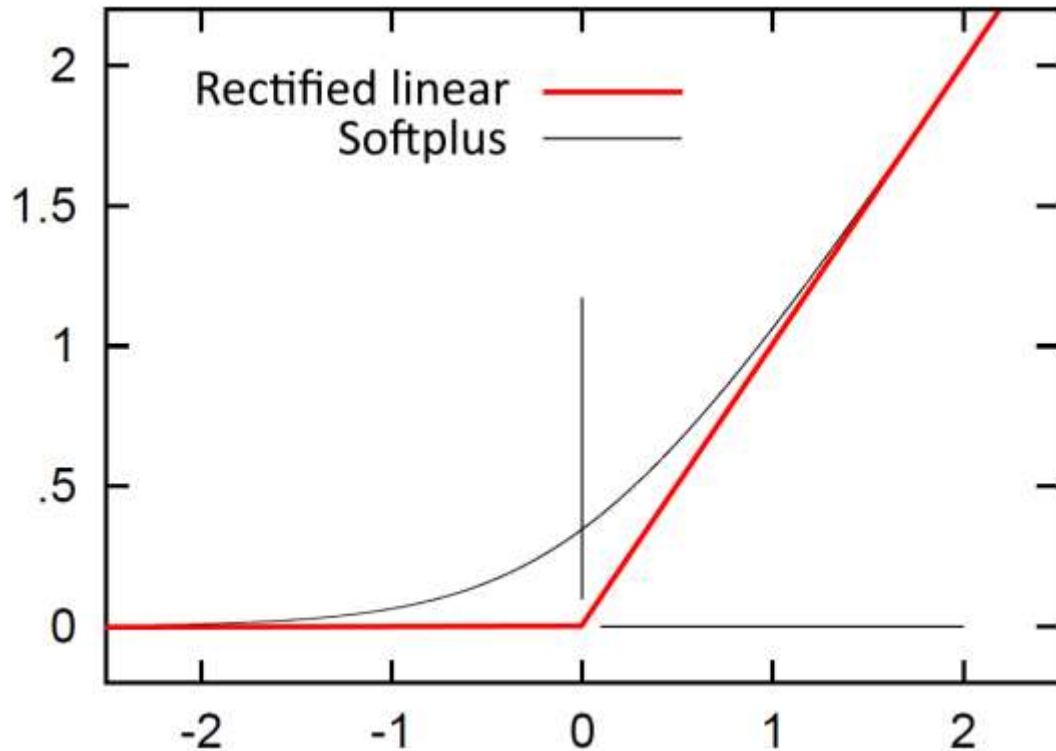
TanH function



- TanH function

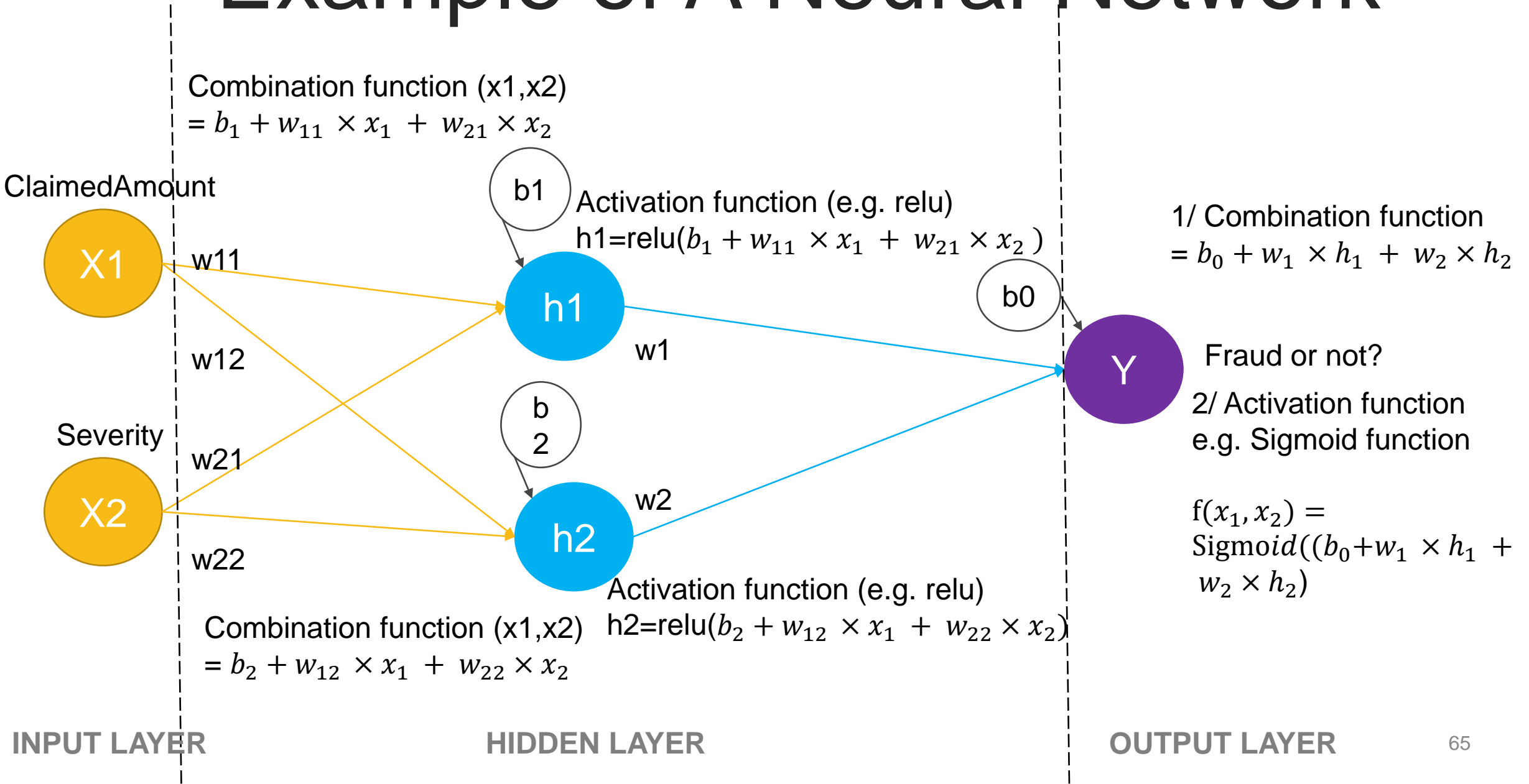
- Similar to Sigmoid function, but zero centered, i.e. easier to model inputs with strong negative, neutral and positive values

Most commonly used activation functions



- ReLU Function
 - $\text{ReLU}(x) = \max(0, x)$
 - Rectified Linear Unit (ReLU)
 - Outputs 0 for negative values of x
- Softplus Function
 - $f(x) = \ln(1 + e^x)$
 - Outputs of Sigmoid and tanh functions are bounded between upper and lower limits
 - Softplus function produces outputs in scale of $(0, +\infty)$. That's the essential difference.

Example of A Neural Network



A hand is visible on the left side of the frame, with the index finger pointing towards the center. The background is a dark, out-of-focus screen displaying the title and presenter information. The lighting is dim, with a bright spot where the finger points.

Training a Neural Network

Dr. Vivien CHAN

Cost / Loss function

- Neural network is trained based on a cost/loss function
 - To measure the error of the network's prediction
 - Or how good the neural network model is for a particular task
- Cost function – mean square error (MSE)
 - Difference between predicted output and actual output
 - Square the difference and
 - Calculate the mean

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

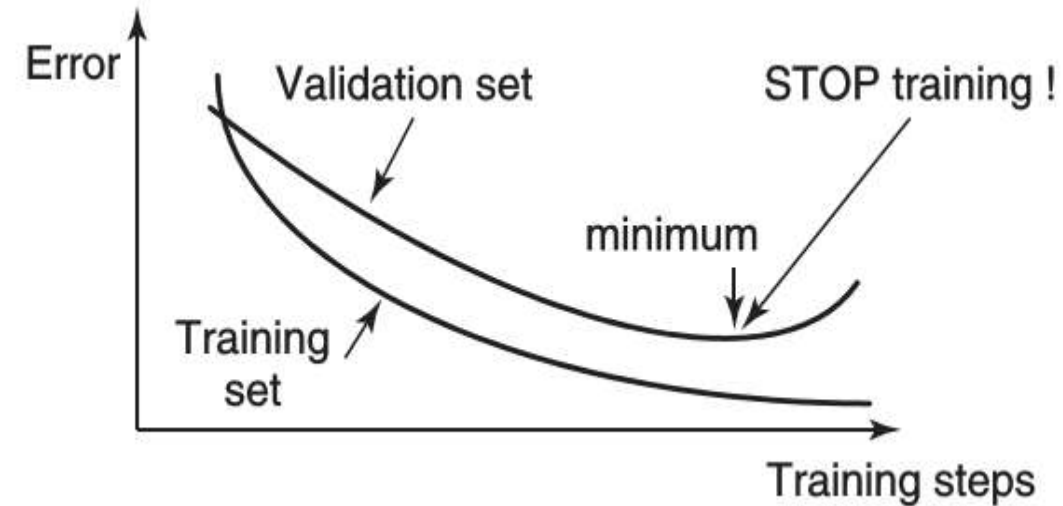
- If the cost function is big then our network doesn't perform very well, we want MSE as small number as possible

Training a neural network

1. Split the data into a training, validation, and test set.
2. Vary the number of hidden neurons from 1 to 10 in steps of one or more.
3. Train a neural network on the training set and measure the performance on the validation set (may be train multiple neural networks to deal with the local minimum issue).
4. Choose the number of hidden neurons with optimal validation set performance.
5. Measure the performance on the independent test set.
Note that for fraud detection, the number of hidden neurons typically varies between 6 and 12.

NOTE

- In terms of artificial neural networks, an epoch refers to one cycle through the full training dataset.
- Usually, training a neural network takes more than a few epochs.



Neural Network for Fraud Analytics

- Usually one hidden layer is enough for fraud analytics model
- Theoretically can use different activation function at each node, but usually this is fixed for each layer
- For hidden layers,
 - use non-linear functions, e.g. tanh function
- For output layers,
 - Classification target (fraud vs no-fraud): use sigmoid function
 - Regression target (e.g. amount of fraud): can use any function
- Data pre-processing
 - Normalize the continuous variables, e.g. using z-scores
 - Reduce the number of categories for categorical variables

A hand is visible on the left side of the frame, with the index finger pointing towards a bright, circular light spot on a dark, textured surface. The background is dark and out of focus, with a horizontal white line visible near the bottom.

How to interpret Neural Network

Dr. Vivien CHAN

Neural network

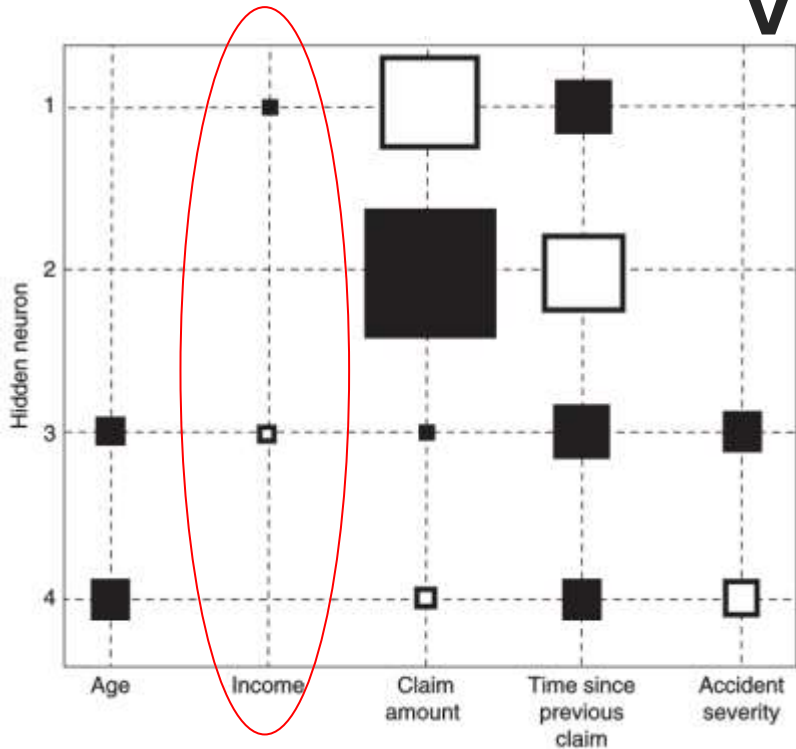
- Neural network is a useful tools to fraud analytics due to its modelling power
- It is known as black-box approach as the inputs and outputs are related in a mathematically complex, nontransparent and opaque way
- However, for fraud analytics, it is sometimes important to gain insight into the fraud behaviours. So, you need to be care when you choose to use neural network for fraud analytics.

Neural network

There are **3 ways** to interpret a neural network model:

- 1/ Variable Selection
 - Aim is to select those variables that actively contribute to the neural network output
 - Does not give insight about the internal workings of the neural network
- 2/ Rule extraction
 - Aim is to extract if-then classification rules
- 3/ Two-stage models
 - Combines 2 models with both interpretability and performance

Variable selection



Hinton diagram

- Visualizes the weights between inputs and hidden neurons as squares
- **Size** of square is proportional to **size** of the weight
- **Color** of the square represents **sign** of the weight (e.g. black = -ve, white = +ve)
- E.g. “Income” variable can be removed

(a) Variable selection procedures:

1. Inspect the Hinton diagram and remove the variable whose weights are closest to ZERO
2. Re-estimate the neural network with the variable removed. To speed up the convergence, it could be beneficial to start from the previous weights.
3. Continue with step 1 until a stopping criterion is met. The stopping criterion could be a decrease of predictive performance or a fixed number of steps.

Variable selection

(b) Backward Variable Selection procedures:

1. Build a neural network with all N variables.
2. Remove each variable in turn and reestimate the network. This will give N networks each having $N-1$ variables.
3. Remove the variable whose absence gives the best performing network (e.g. in terms of misclassification error, mean squared error).
4. Repeat this procedure until the performance decreases significantly.

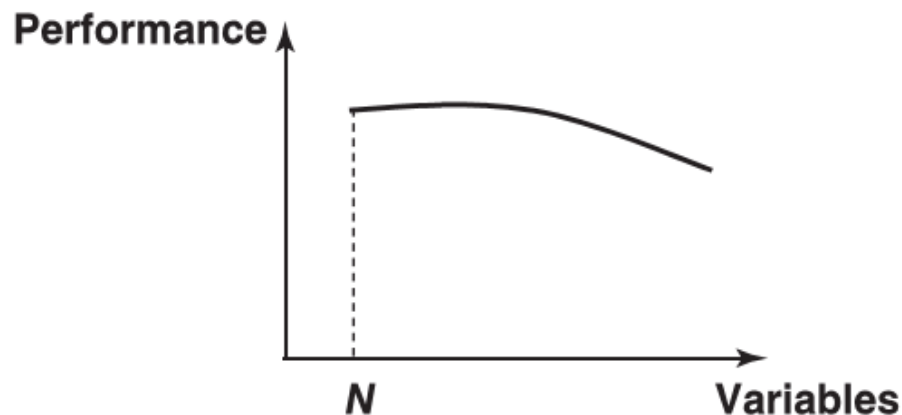


Figure 4.22 Backward Variable Selection

Rule extraction – Decompositional Approach

- Decompositional rule extraction approaches decompose the network's internal workings by inspecting weights and/or activation values.
- Procedures (Lu et al., 1995; Setiono et al. 2011):
 1. Train a neural network and do variable selection to make it as concise as possible.
 2. Categorize the hidden unit activation values by using clustering.
 3. Extract rules that describe the network output in terms of the categorized hidden unit activation values.
 4. Extract rules that describe the categorized hidden unit activation values in terms of the network inputs.
 5. Merge the rules obtained in step 3 and 4 to directly relate the inputs to outputs.

Rule extraction – Decompositional Approach

Customer	Age	Income	Known Customer	...	Fraud
Emma	28	1000	Y		No
Will	44	1500	N		Yes
Dan	30	1200	N		No
Bob	58	2400	Y		Yes

Step 1: Start from original data

Customer	Age	Income	Known Customer	h1	h2	h3	h1	h2	h3	Fraud
Emma	28	1000	Y	-1.20	2,34	0,66	1	3	2	No
Will	44	1500	N	0,78	1,22	0,82	2	3	2	Yes
Dan	30	1200	N	2,1	-0,18	0,16	3	1	2	No
Bob	58	2400	Y	-0,1	0,8	-2,34	1	2	1	Yes

Step 2: Build a neural network (e.g., 3 hidden neurons)

Step 3: Categorize hidden unit activations

If h1 = 1 **and** h2 = 3 **Then** Fraud = No

If h2 = 2 **Then** Fraud = Yes

Step 4: Extract rules relating network outputs to categorized hidden units

If Age < 28 **and** Income < 1000 **Then** h1 = 1

If Known Customer = Y **Then** h2 = 3

If Age > 34 **and** Income > 1500 **Then** h2 = 2

Step 5: Extract rules relating categorized hidden units to inputs

If Age < 28 **and** Income < 1000 **and** Known Customer = Y **Then** Fraud = No

If Age > 34 **and** Income > 1500 **Then** Fraud = Yes

Step 6: Merge both rule sets

Figure 4.23 Decompositional Approach for Neural Network Rule Extraction

Rule extraction – Pedagogical Approach

Customer	Age	Income	Known Customer	...	Fraud
Emma	28	1000	Y		No
Will	44	1500	N		Yes
Dan	30	1200	N		No
Bob	58	2400	Y		Yes

Step 1: Start from original data

Customer	Age	Income	Known Customer	Network Prediction	Fraud
Emma	28	1000	Y	No	No
Will	44	1500	N	Yes NO	Yes
Dan	30	1200	N	Yes	No
Bob	58	2400	Y	Yes	Yes

Step 2: Build a neural network

Step 3: Get the network predictions and add them to the data set

- Neural network as a black box
- Use the neural network predictions as input to a white-box analytical technique, e.g. decision tree

Step 4: Extract rules relating network predictions to original inputs. Generate additional data where necessary.

- The data set is labelled (e.g. classified or predicted) by the neural network
- The decision tree (or any other white-box models) can use the NN predicted labels as additional attributes for building the decision tree

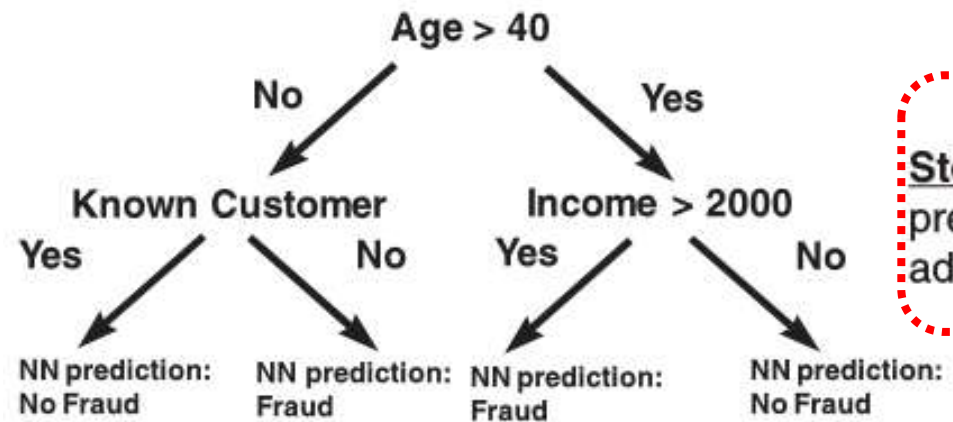


Figure 4.24 Pedagogical Approach for Rule Extraction

Performance evaluation of rule-extraction

- Rules extraction sets evaluated in terms of:
 - Accuracy
 - Conciseness (e.g. number of rules, number of conditions per rule)
 - Fidelity – measures to what extent the extracted rule set succeeds in mimicking the neural network

Rule Set Classification		Neural Network Classification	
		No Fraud	Fraud
	No Fraud	a	b
	Fraud	c	d

$$\text{Fidelity} = (a + d)/(b + c).$$

- Benchmark the extracted rules/trees with a tree built directly on the original data to see the benefit of going through the neural network

Two-stage models

- Stage one (model interpretability)
 - to estimate an easy-to-understand model (e.g. linear regression, logistic regression)
- Stage two (model performance)
 - to predict the errors made by the simple model using the same set of predictors

Customer	Age	Income	Known Customer	...	Fraud
Emma	28	1000	Y		No
Will	44	1500	N		Yes
Dan	30	1200	N		No
Bob	58	2400	Y		Yes

Step 1: Start from original data

Customer	Age	Income	Known Customer	...	Fraud	Logistic Regression output
Emma	28	1000	Y		No (=0)	0.44
Will	44	1500	N		Yes (=1)	0.76
Dan	30	1200	N		No (=0)	0.18
Bob	58	2400	Y		Yes(=1)	0.88

Step 2: Build Logistic Regression Model

Customer	Age	Income	Known Customer	...	Fraud	Logistic Regression output	Error
Emma	28	1000	Y		No (=0)	0.44	-0.44
Will	44	1500	N		Yes (=1)	0.76	0.24
Dan	30	1200	N		No (=0)	0.18	-0.18
Bob	58	2400	Y		Yes(=1)	0.88	0.12

Step 3: Calculate errors from Logistic Regression Model

Step 4: Build NN predicting errors from Logistic Regression Model

Customer	Age	Income	Known Customer	...	Logistic Regression output	NN output	Finaloutput
Bart	28	1000	Y		0.68	0.14	0.82

Step 5: Score new observations by adding up logistic regression and NN scores

Figure 4.25 Two-Stage Models

Neural Network

Advantages

- Handle different types of input variables and support multivariate targets
- Support flexible and nonlinear relations between input and target variables
- Due to flexibility of the model, it is applicable to many problems

Disadvantages

- Black-box model, not easy to interpret
- Prone to overfit
- Requires handling of input attributes, e.g. data imputation.



Using R to build Neural Network

Dr. Vivien CHAN

Neural Network

Preparing data for Neural Network

1. Handle imbalance Data Set
2. Impute missing values, if needed
3. Standardize or normalize all input attributes
 - Use the function “**scale**” to normalize the attributes

(Continue from Step 2) Step 3: Normalize the input attributes

```
# scale only the Amount attribute  
tmp <- scale(train_un[,c(29)])  
train_un[,c(29)] <- tmp  
str(train_un)
```

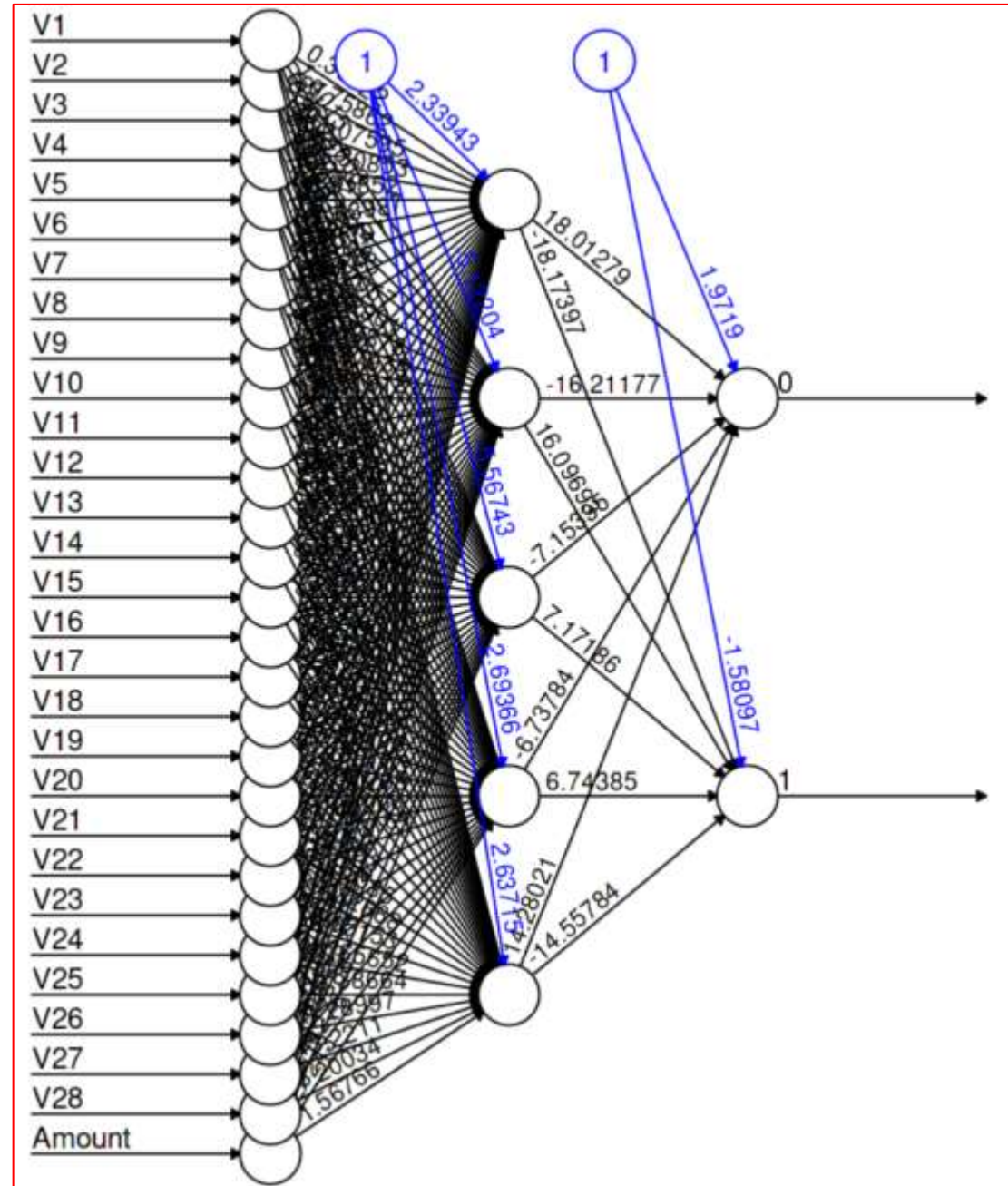
```
$ Amount: num  70.5 2849.94 4.61 1.29 32 ...
```



```
$ Amount: num [1:779, 1] -0.156 9.706 -0.39 -0.402 -0.293 ...  
..- attr(*, "scaled:center")= num 115  
..- attr(*, "scaled:scale")= num 282
```

Step 4: Train the neural network

```
# train a neuralnetwork model
set.seed(123)
model_nn <- neuralnet(Class ~., data=train_un,
                        hidden=5,
                        linear.output=FALSE)
plot(model_nn, rep = 'best')
```



Step 5: Check performance of the neural network model

```
# show performance results of TRAINING data
pred_nn_train <- predict(model_nn, train_un)
print("TRAINING data confusion matrix")
table(train_un$class == "1", pred_nn_train[, 1] > 0.5)

# show performance results of TESTING data
pred_nn <- predict(model_nn, cc_test)
print("TESTING data confusion matrix")
table(cc_test$class == "1", pred_nn[, 1] > 0.5)
```

```
[1] "TRAINING data confusion matrix"
```

	FALSE	TRUE
FALSE	392	2
TRUE	1	384

```
[1] "TESTING data confusion matrix"
```

	FALSE	TRUE
FALSE	5122	51741
TRUE	63	35



What is the performance of this neural network model?

What are the possible next steps to do?



QUESTIONS?

References

- Bart Baesens, Veronique Van Vlasselaer, Wouter Verbeke (2015). Fraud Analytics using Descriptive, Predictive, and Social Network Techniques, 1st ed, John Wiley & Sons Inc.
- Leonard W. Vona (2017). Fraud Data Analytics Methodology: The Fraud Scenario Approach to Uncovering Fraud in Core Business Systems, John Wiley & Sons, Inc.
- Spann, Delena D. (2013). Fraud Analytics : Strategies and Methods for Detection and Prevention, John Wiley & Sons, Incorporated, 2013. ProQuest Ebook Central, <http://ebookcentral.proquest.com/lib/hkuhk/detail.action?docID=1752695>
- G. Menardi and N. Torelli. (2014). Training and assessing classification rules with imbalanced data. Data Mining and Knowledge Discovery, 28(1):92–122.
- Gevrey, Muriel & Dimopoulos, Y. & Lek, Sövan. (2003). Review and comparison of methods to study the contribution of variables in Artificial Neural Networks models. Ecological Modelling. 160. 249-264. 10.1016/S0304-3800(02)00257-0.
- Severino, M.K., Peng Y.H. (2021). Machine learning algorithms for fraud prediction in property insurance: Empirical evidence using real-world microdata. Machine Learning with Applications 5 (2021) 100074. <https://doi.org/10.1016/j.mlwa.2021.100074>
- Xuan, S., Liu, G., & Li, Z. (2018). Refined Weighted Random Forest and Its Application to Credit Card Fraud Detection. CSoNet.
- Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2017). Gotcha! Network-based Fraud Detection for Social Security Fraud.