

Actuaries Institute

YAP and YDAWG Data Analytics Case Study Preparation

R Workshop

Actuaries Institute

Agenda

- Introduction
- Transforming Data
- Exploratory Visualisation
- Machine Learning
- Model Visualisation
- Advanced Visualisation

Introduction

Why R for Data Analytics?

- Used by many companies
- Purpose built for statistics
- Great editor (R studio)
- Free packages for data, visualisation and machine learning

Goals

In this session you'll learn about:

- Transforming Data
- Exploring data with visualisations
- Machine learning
- Model visualisations

Focus on machine learning with the first two sections to get us there

Installation and Troubleshooting

- Follow installation instructions for Windows or Mac
- Download this code from the github link https://github.com/ActuariesInstitute/YAP-YDAWG-R-Workshop
- Or run in the web browser! http://mybinder.org/v2/gh/ActuariesInstitute/YAP-YDAWG-R-Workshop/master?urlpath=rstudio

Credits

YAP and YDAWG R Workshop team - Contributions from:

- Jonathan Shen
- Tim Lam
- Josh Jaroudy
- Cindy Vuong
- Arjun Sathasivam
- Heloísa Bete
- Michelle Ng
- Jacky Poon

Transforming Data

Transforming Data for Models – Variable Types

- Data will often come in many different types

Variable Type	Description
Categorical / nominal	• Discrete data that falls into categories • E.g. gender, month
Ordinal	• Discrete numerical data that can be put into a ranking • Has no meaning beyond its relativity to other data points • E.g. review ratings (1 star < 2 stars < 3 stars etc.)
Continuous	• Numerical, continuous data • E.g. claim severity, age
Unstructured	• Data without consistent form • Requires advanced techniques to create structure • E.g. text notes, images

Transforming Data for Models – Selecting Rows/Columns

- To select rows from a data frame, use the first index: df[4,]
- Alternatively, select a set of rows: df[c(1,5,7,10:20),]
- Or select rows according to a formula: df[State == "NSW",]
- Selecting columns works similarly, but using the second data frame index: df[, c("State", "Dayweek", "Time")]

Transforming Data for Models – Creating New Columns

- To create a new formula, simply define it:

```
df[, "New.Column"] <- df[, "Old.Column"] * 1.5
```

- Alternatively, use the data frame "$" notation:

```
df$New.Column <- df$Old.Column * 1.5
```

Transforming Data for Models – Merging Datasets

- Use the "merge" function to join two datasets together:

  merge(df1, df2, by = "col")

- This links data frames df1 and df2, using the common column "col"
- By default, this performs the equivalent of an inner join

Transforming Data for Models – Merging Datasets (Cont.)

- You can perform left join and right join merges as well
- Do this by setting the all.x or all.y arguments to TRUE

```
merge(df1, df2, by = "col", all.x = TRUE)
```

Transforming Data for Models – Rationale

- Data is often categorical.
- However underlying ML models often perform matrix calculations on numerical data.
- Some R packages have logic to transform categorical variables to numeric automatically, but others do not.
- So it is often good practice to convert factor variables to numeric data before modelling.

Transforming Data for Models – Categorical Variables

- R uses the factor data type to represent categorical variables.
- Factors are simply integer variables, but displayed using pre-defined categories.
- The categories can be inspected using the "levels" function:

```
levels(df1$factor.col)
```

Transforming Data for Models – Categorical Variables (Cont.)

- In our example, we can set flags manually for gender and each state.
- Setting one/zero flags is called "one-hot encoding" and is a type of "contrast".

	Female_fl	ACT_fl	NT_fl	QLD_fl	SA_fl	TAS_fl	VIC_fl	WA_fl	A
6	0	0	0	1	0	0	0	0	
7	0	0	0	0	0	0	0	1	
6	1	0	0	1	0	0	0	0	
0	0	0	0	0	1	0	0	0	
8	0	0	0	0	0	0	0	0	
6	0	0	0	1	0	0	0	0	
1	1	0	0	0	0	0	1	0	
0	0	0	0	0	1	0	0	0	
1	1	0	0	0	0	0	1	0	
6	0	0	0	1	0	0	0	0	
8	1	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	
8	1	0	0	0	0	0	0	0	
0	0	0	1	0	0	0	0	0	
7	1	0	0	0	0	0	0	1	
6	0	0	0	1	0	0	0	0	

Transforming Data for Models – Categorical Variables (Cont.)

- Or use model.matrix to set a "contrast" automatically:

X.Intercept.	State1	State2	State3	State4	State5	State6	State7
1	1	-1	-1	-1	-1	-1	-1
1	0	0	3	-1	-1	-1	-1
1	0	0	0	4	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1
1	0	0	3	-1	-1	-1	-1
1	0	0	3	-1	-1	-1	-1
1	0	0	0	4	-1	-1	-1
1	0	2	-1	-1	-1	-1	-1
1	0	0	0	0	0	6	-1
1	0	0	0	0	0	6	-1

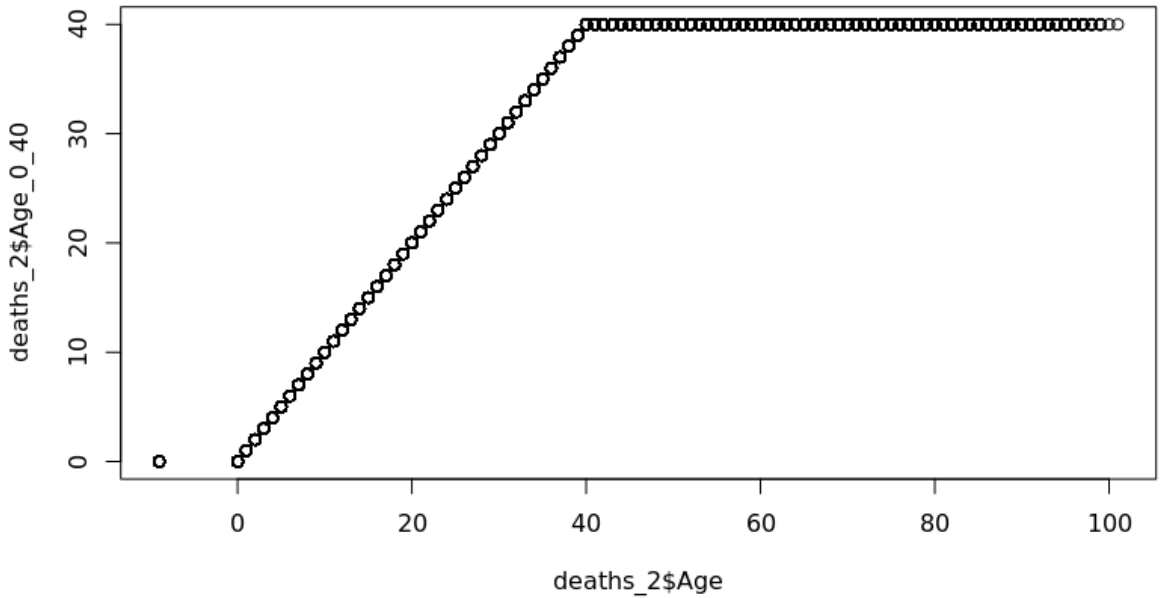
Transforming Data for Models – Categorical Variables (Cont.)

- Other common transformations

Transformation	How	Application Considerations
One-Hot Encoding	New variables that is e.g. 1 or 0 for each category [except for the first one]	• Works for most models • Most transparent option • But memory intensive
Integer Encoding	Number categories from 1,2,…n	• Moderate memory usage • Do NOT use for linear models! • Neural Network using embedding layer
Mean Encoding	Take the average of the response for each category and use that as the representation of the category	• May be better than integer for some models – e.g. likely to split better for decision trees
Binary / Hash Encoding	Binary e.g. three 1/0 columns can represent 8 levels (2^3) Category no. 5 = 1 | 0 | 1	• Limits memory usage • Potentially loses information

Transforming Data for Models – Continuous Variables

- Cap Age at a certain level using pmin and pmax:



Transforming Data for Models – Continuous Variables (Cont.)

- Common transformations used for various model types:

Model	Transformations
[Generalised] Linear Models	• Splines • Legendre Polynomials
Decision Trees, Random Forests and Boosting	• Generally not needed except missing (see below)
Neural Networks	• Standardise to range or mean/variance
Treatment of missing data (generally)	• Impute with mean value • Train another model to predict and fill missing data • Missing flag

Exercise

- What would happen if we just used min() and max()?
- “Do we get more pedestrian deaths on the weekend?”

Further learning – Transforming Data

- Dplyr
  - Query data frames and SQL databases with similar syntax
  - Part of the "tidyverse" of packages including next section's "ggplot2" (https://www.tidyverse.org/)
- Data.table
  - Fast data manipulation

Exploratory Visualisation

Basic Visualisation

- There are a number of basic visualisation functions available in base R e.g. plot(), hist(), barplot()
- These are fine for "quick and dirty" graphs when you're doing exploratory analysis.
- It's *possible* to make these look good, but can be a lot of effort!

Visualisation using ggplot2

- Based on the 'grammar of graphics': specify the dataset, the variable to plot (+ its characteristics) and the type of graph.
- Sample barchart: ggplot(data=X, aes(var)) + geom_bar()
  e.g. Plotting the 'Speed.Limit' variable on the 'deaths' dataset: ggplot(data=deaths, aes(deaths$Speed.Limit)) + geom_bar()
- Similarly, you can try geom_line() and geom_histogram()

Other Characteristics

- Incrementally build up graphs e.g. histogram characteristics, labels and titles, axis limits

```
ggplot(data=deaths, aes(deaths$claim_severity)) +
geom_histogram(breaks = c(seq(0, 9000000, 250000)), col="white", fill = "black")+
labs(title="Claim Severity Distribution", x="Claim Severity",y="Count")+
xlim(c(0, 10000000)) + ylim(c(0, 10000))
```

Summary Statistics

- Simple summary statistics can be directly calculated
- E.g. average claim_severity by age

```
ggplot(data=deaths, aes(x=factor(Age), y=claim_severity)) +
        stat_summary(fun.y="mean", geom="bar")
```

Exercise

- Open up the ggplot2 cheatsheet (Help > Cheatsheets > Data visualisation with ggplot2)
- Try visualizing Bus..Involvement and Articulated.Truck..Involvement. What do you notice?
- Try visualizing National_Road_Type and Year. What do you notice?
- Try one of the other chart types with another variable from the dataset. What story does it tell?

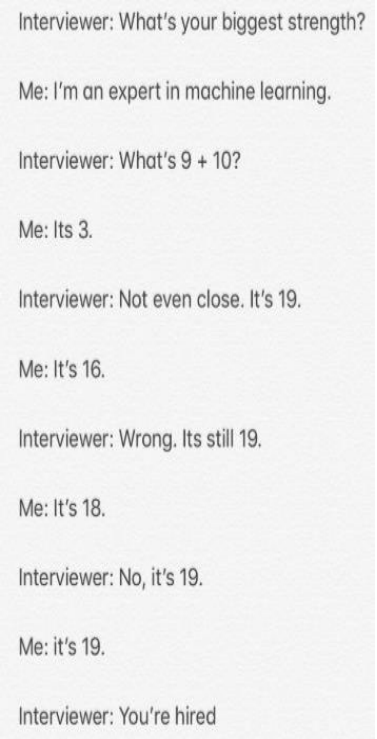
Further Learning – Exploratory Visualisation

- Graphing with base R: https://www.statmethods.net/graphs/index.html
- ggplot2 homepage: https://ggplot2.tidyverse.org/
- Graphing with ggplot2: http://www.cookbook-r.com/Graphs/

Machine Learning

What is Machine Learning?

- Deals with the problem of extracting **features** from data to make predictions:
  - Forecasting (e.g. sales, demand for insurance products)
  - Classifying (e.g. credit risk assessment)
  - Detecting anomalies (e.g. cases of insurance fraud)
- Find patterns that generalise beyond the data, and which can be used to predict outcomes for new data you haven't seen.
  - Machine learning models are built using the **training** data
  - Their suitability is tested against the **validation** data

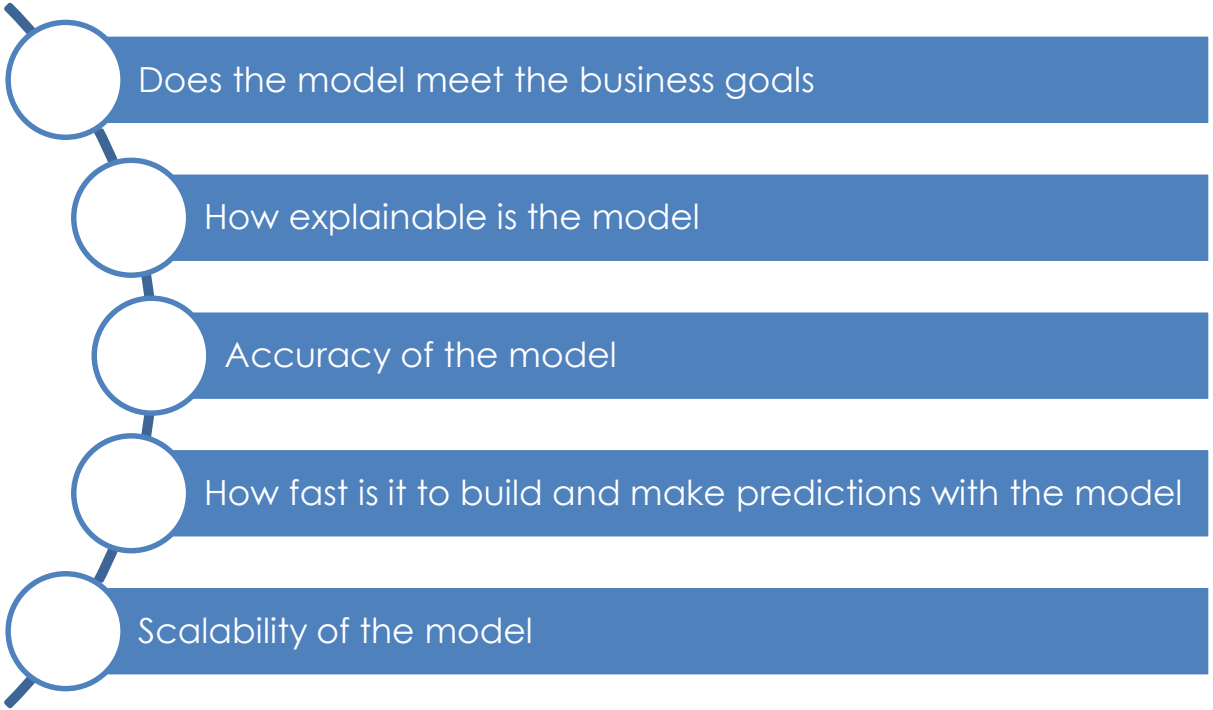


Pre-processing the Data

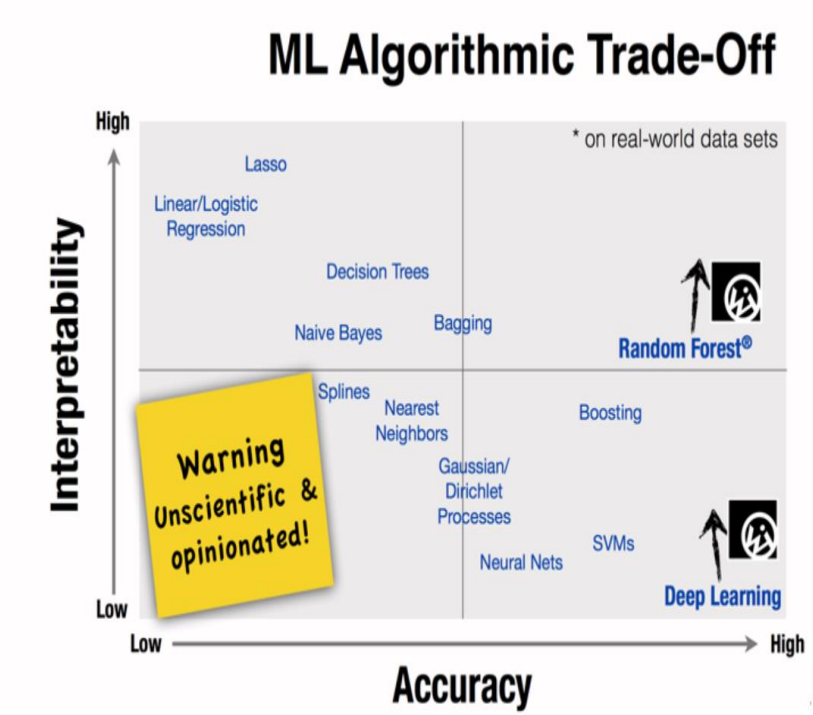
Have a clear understanding of:

- Data
  - Clean data for missing values; deal with outliers
  - Data types (e.g. Are the inputs labelled or unlabelled)
  - Feature Engineering – Process of going from raw data to ready to model data
  - Visualise data
- Constraints
  - Data storage
  - Do the predictions have to be fast – E.g. In autonomous driving, classifying road signs quickly is important
  - Does learning have to be fast – Is it necessary to train models quickly

Machine Learning Model Considerations



Machine Learning Model Considerations (Cont.)



Source: Choosing the right machine learning algorithm, https://hackernoon.com/choosing-the-right-machine-learning-algorithm-68126944ce1f

Algorithms Covered

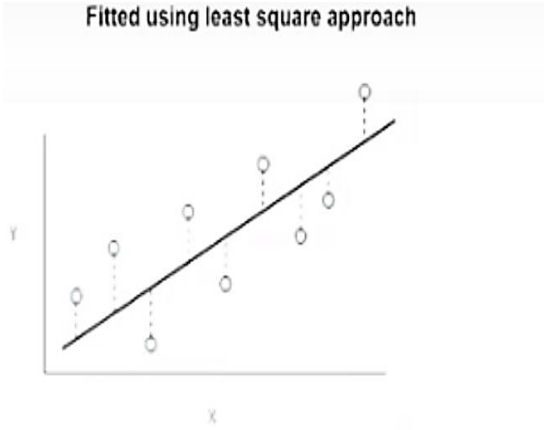
- Linear Regression
- K-Means Clustering
- Random Forest

Linear Regression

One-Variable:

Multiple-Variable:

- Optimal model minimises the SSEs (errors)
- Where possible one should remove variables that are very similar (correlated) and remove noise from your data



Objective Function

$$Error = \sum_{i}^{N} (Known_i - Predicted_i)^2$$

Linear Regression (Cont.)

Advantages

- Fast, simple and widely used
- Results are interpretable

Disadvantages

- Assumes a linear relationship
- Assumes independence within the dataset
- Sensitive to outliers
- Focuses on the relationship between the mean of the dependent variable and the independent variables. The mean is not a complete description (e.g. if you want to focus on the extremes).

Uses

- Good first algorithm to try to gain insights into the relationship between the response variable (Y) and predictor variables (X)

Linear Regression – R Exercise

It's time for the exercise. Here's some code to get your started:

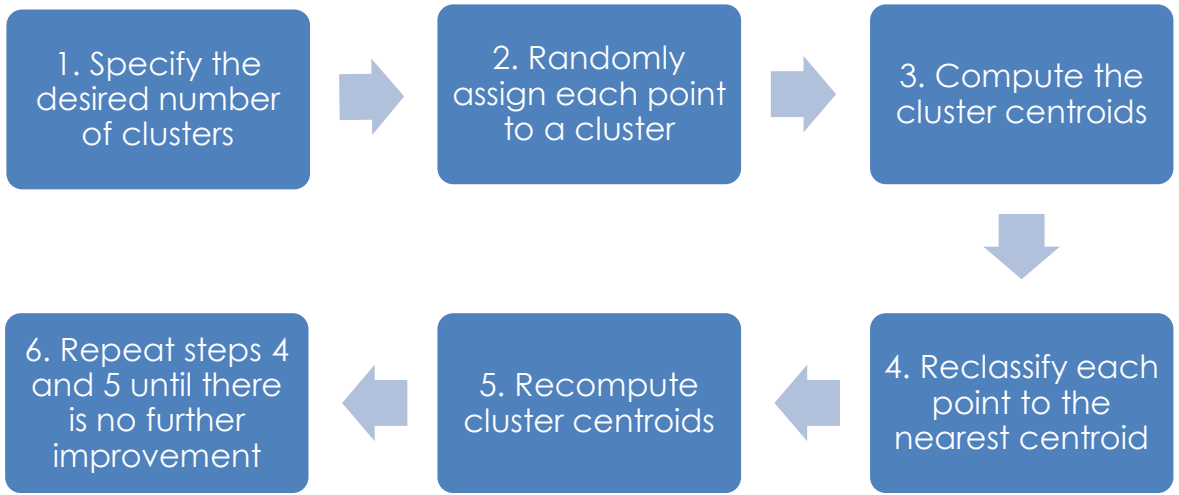
```
model <- lm(Y ~ X1 + X2 + …, data=mydata)
```

Other useful functions:

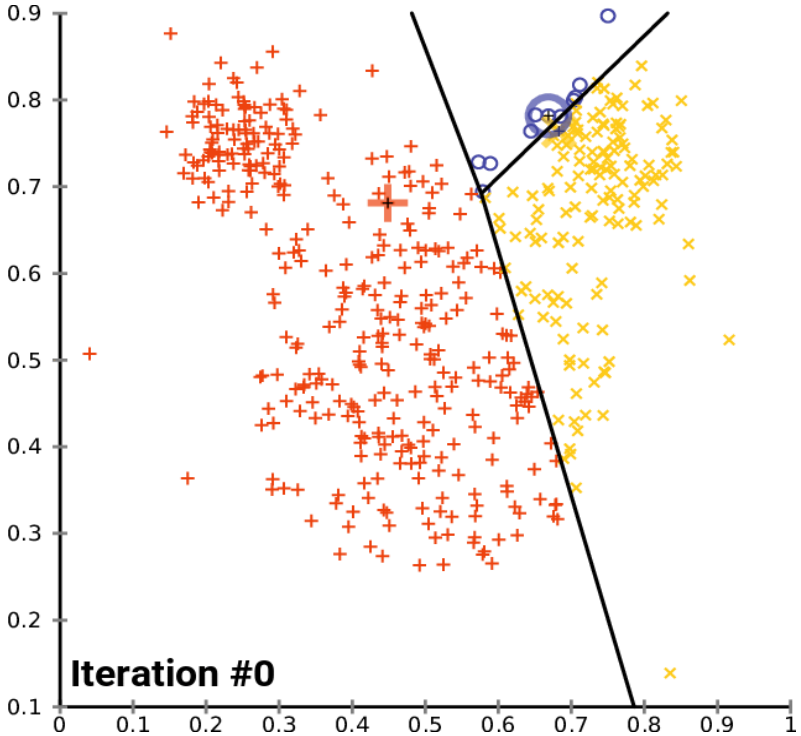
- summary(model) - Results on the beta coefficients, r-squared and residuals
- coefficients(model) - Model coefficients
- cofint(model, level = 0.95) - Confidence intervals for model parameters
- fitted(model) - Predicted values
- residuals(model) - Residuals
- anova(model) – Anova table
- influence(model) – Regression diagnostics

K-Means Clustering

- Unsupervised method (when you have unlabelled data) used to find groups within the data. Partition n observations into a fixed number of k clusters.
- Grouping is based on distance between points.



K-Means Clustering (Cont.)



K-Means Clustering (Cont.)

Advantages

- Easy to implement

Disadvantages

- Choice of centres requires judgement and is subjective
- Initial seeds has a strong impact on final results
- Sensitive to scale: Rescaling your dataset (normalisation) can change results

Example Uses

- Customer Segmentation – Targeted marketing
- Insurance Fraud Detection – Using past data on fraudulent claims, it's possible to isolate potential new fraudulent claims based on proximity to clusters with fraudulent patterns

K Means Clustering – R Exercise

It's time for the exercise. Here's some code to get your started:

kmeans(x, centres = c, nstart = n) where

- x - Data
- centres - Pre-defined number of clusters
- nstart – Model attempts n initial configurations

E.g. N = 25 generates 25 initial random centroids. The optimal configuration will be chosen for the algorithm.

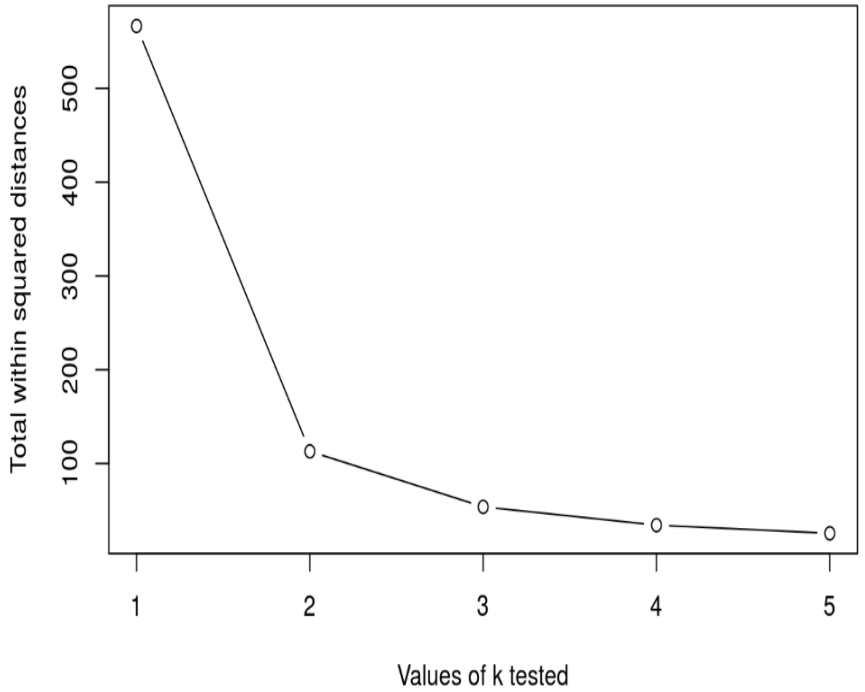
Issue 1: Model Selection

- Note that due to the random initialisation, one can obtain different clustering results.
- When k-means is run multiple times, the model that generates the smallest total within cluster sum of squares (SS)* is selected (i.e. a local minimum). There's no guarantee to obtain a global minimum.

*SS - For each observation, determine the squared euclidean distance from observation to centre of cluster and sum all the distances.

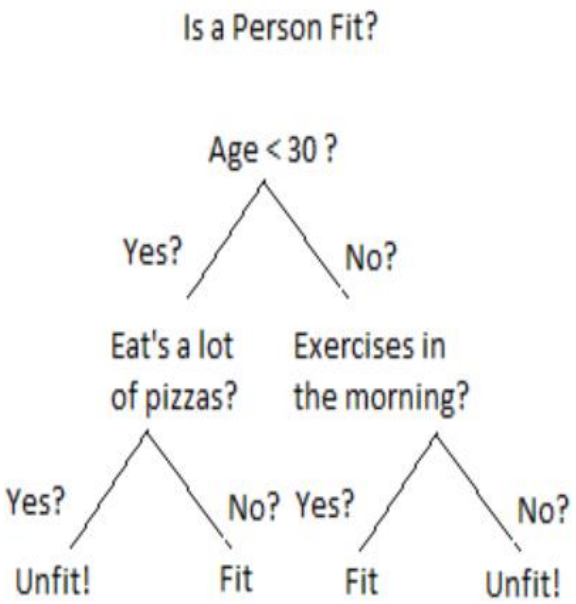
Issue 2: Cluster Selection

1. Run k-means with k=1, k=2, …, k=n
2. Record total within SS for each value of k.
3. Choose k at the elbow position, as illustrated below. Balance between over fitting and accuracy.



Decision Tree

- Uses a branching method to illustrate possible outcomes of a decision.
- Decision trees are a good choice when you have a large number of variables and aren't sure which ones are key or how they relate. The decision at the top of the model is the variable that most affects the outcome.
- Each node represents an input variable (x) and a split point on that variable.
- The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.



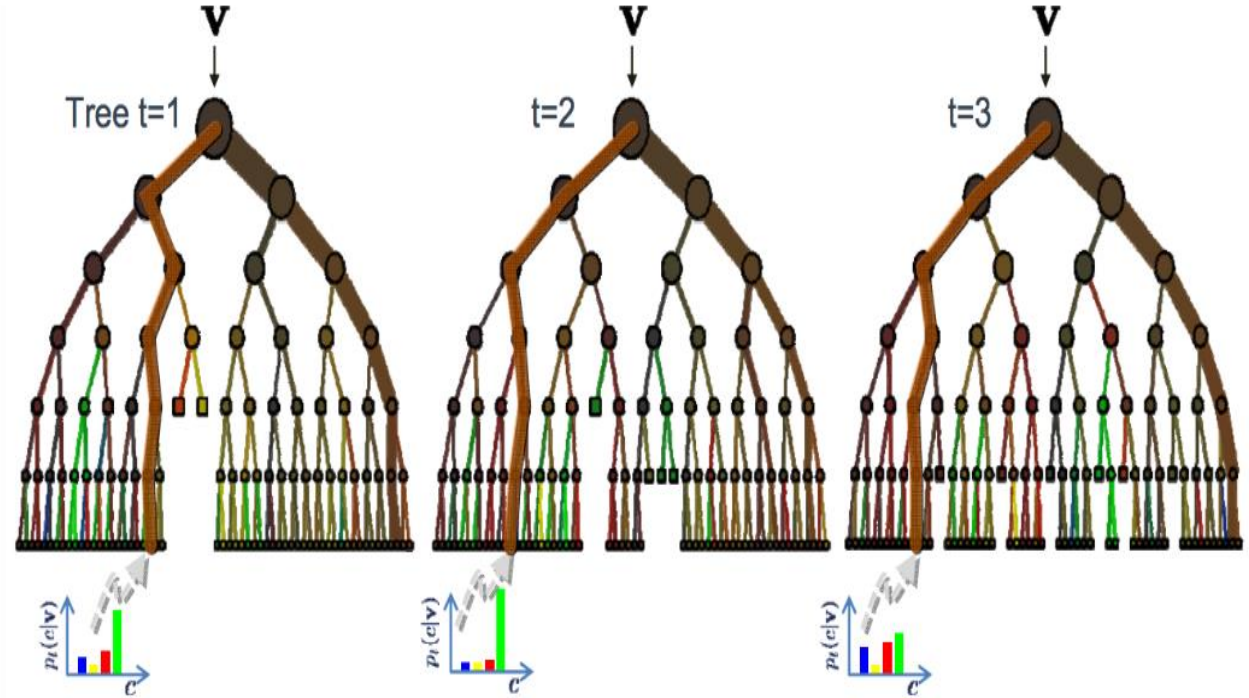
Random Forest

- Are an ensemble of decision trees.
- Multiple samples of your training data are taken and decision trees are constructed for each data sample. Each model makes a prediction with the new data and the predictions are averaged to give a better estimate of the true output.
- E.g. Suppose you want to watch a movie but are uncertain of its reviews. You ask 10 people and 7 say that the movie was bad. Since the majority didn't like the movie, you decide against watching it.

Random Forest (Cont.)

- Note that decision trees are created so that rather than selecting optimal split points, suboptimal splits are made by introducing randomness.
- The models created for each sample of the data are therefore more different than they otherwise would be, but still accurate in their unique and different ways. Combining their predictions results in a better estimate of the true underlying output value.

Random Forest (Cont.)



Random Forest (Cont.)

Advantages

- Are non-parametric; have no distribution assumptions
- Reliable and can handle non-linearity in the dataset
- Robust against outliers in the predictive variables
- Robust to correlated parameters
- Does not tend to overfit especially when cross validation is incorporated
- Highly scalable to any number of dimensions and has generally acceptable performance

Disadvantages

- Less interpretable and less easier to explain than regression/clustering
- Many parameters to adjust such as the number of trees and the tree depth although default parameters may work
- Not possible to iteratively improve the generated model
- Performance may be slow

Uses

- Can be used to help identify significant variables from thousands of inputs using its variable importance plot

Exercise – Random Forest

It's time for the exercise. Here's some code to get your started:

randomForest(Y ~ X1 + X2 + …, data, nodesize, ntree, mtry) where:

- data - Training dataset.
- nodesize - The minimum number of observations in the terminal node. The higher the number, the lower the tree depth.
- ntree - Number of decision trees to simulate. The larger, the more operationally intensive the algorithm becomes.
- mtry – Number of variables randomly sampled at each split.

Further Learning – Machine Learning

- A Tour of The Top 10 Algorithms for Machine Learning Newbies, https://towardsdatascience.com/a-tour-of-the-top-10-algorithms-for-machine-learning-newbies-dde4edffae11
- Introduction to Machine Learning with R - https://lgatto.github.io/IntroMachineLearningWithR/index.html
- Gradient Boosting – Catboost, XGBoost or LightGBM – e.g. Minh Phan's WOMD video: https://www.youtube.com/watch?v=G-HQ9HLV8HY or Jacky Poon's xgboost intro: https://www.youtube.com/watch?v=1cqTMqtun0c
- Neural Networks – Jacky Poon's Analytics Snippet https://www.actuaries.digital/2018/08/23/analytics-snippet-multitasking-risk-pricing-using-deep-learning/

Model Visualisation

DALEX – Model Interpretation

- We define an 'explainer' which contains information about the machine learning model we have built

```
expl <- explain(model, data, y)
```

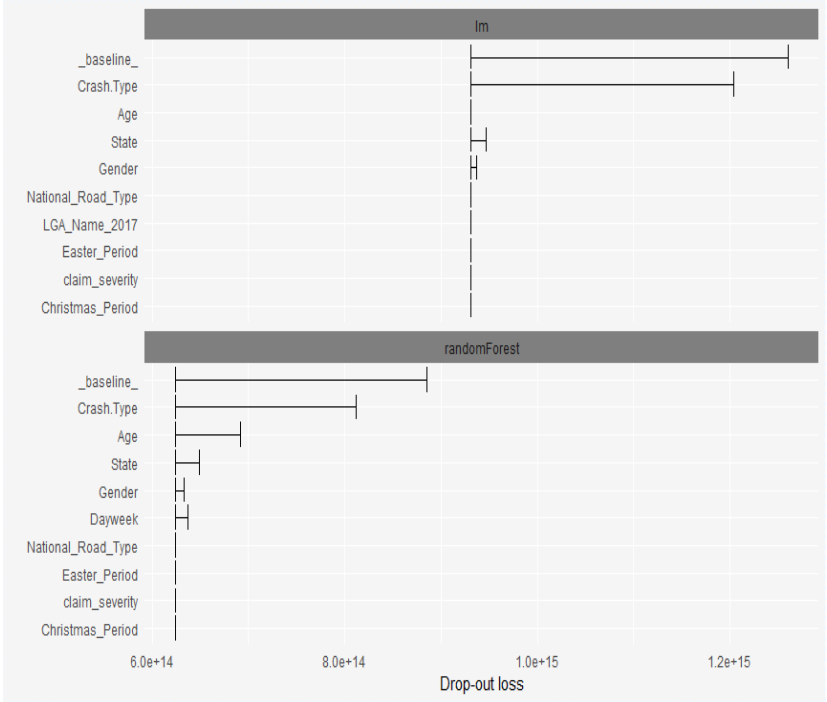
- model – a model object
- data – the dataset we built the model with
- y – the target variable
- In more complex cases, you will also need to specify a function to generate predictions and the link function

DALEX – Variable Importance

- Identifies the variables which contribute most to model predictions

```
vi1 <- variable_importance(explainer1)
vi2 <- variable_importance(explainer2)
plot(vi1, vi2)
```

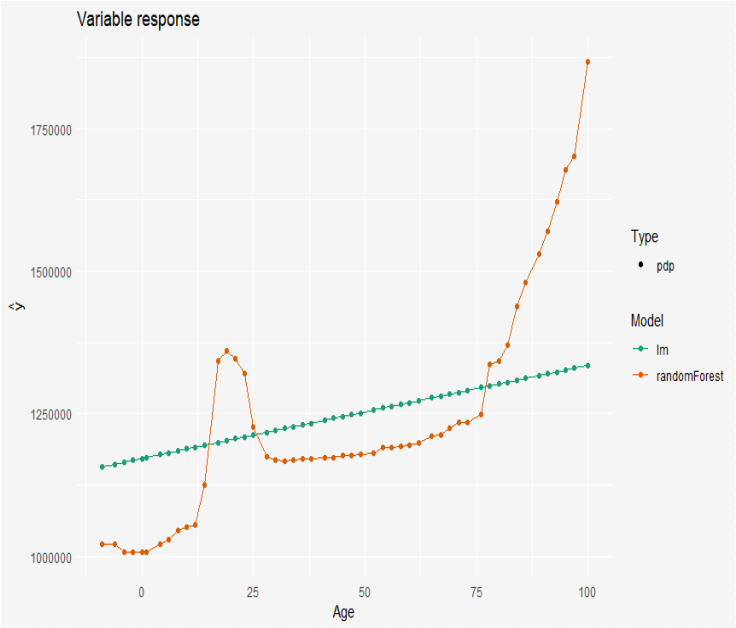
- E.g. in both models, Crash.Type is the most important feature



DALEX – Partial Dependence Plot

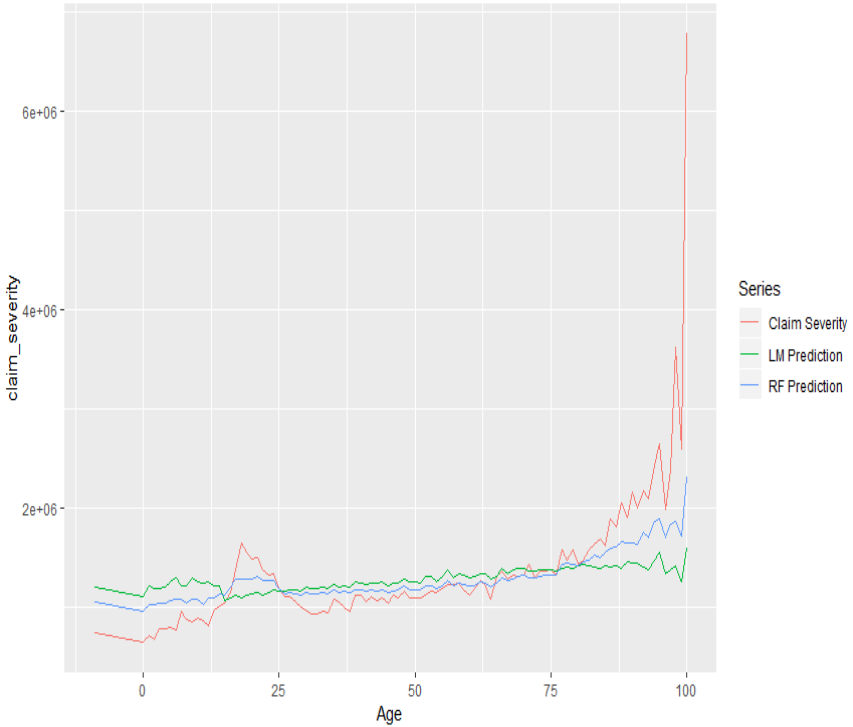
- Plots the relationship between the target and variable of interest, as inferred from the model structure
- E.g. the randomForest has fitted a more complex age curve

```
pd1 <- variable_response(explainer1, "age", "pdp")
```



Actual vs Model Predictions

```
age_avg <-
aggregate(deaths_m[,
c("claim_severity",
"lm_pred",
"rf_pred")], list(Age
= deaths_m$Age),
mean)
age_avg_long <-
melt(age_avg,
id="Age")
ggplot(data=age_avg_l
ong, aes(x=Age,
y=value,
colour=variable)) +
geom_line()
```

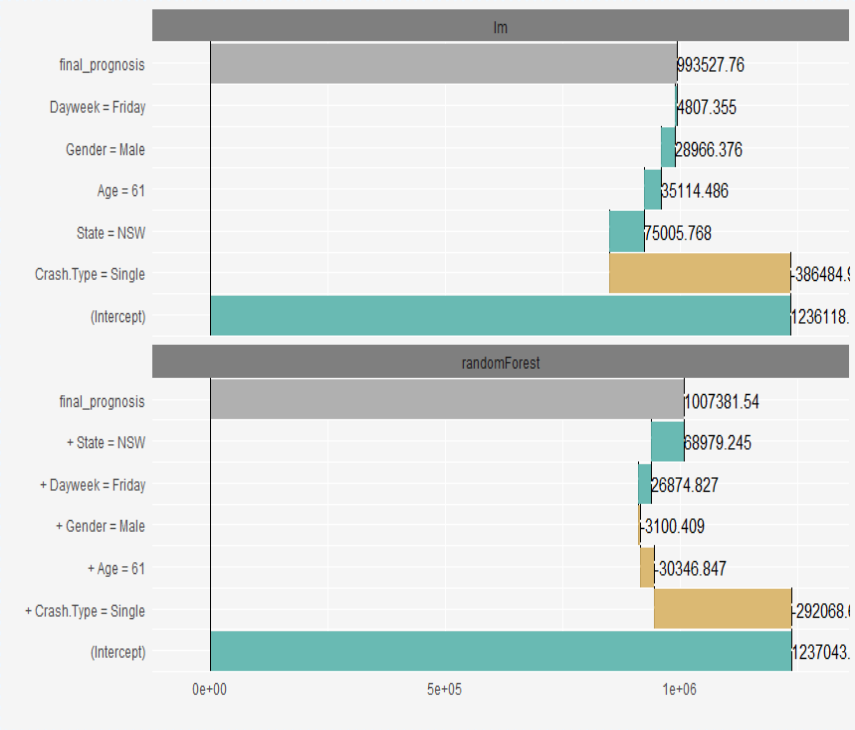


- Plots the target variable vs the model predictions
- Both models need some work.

The RF appears to be a slightly better fit.

DALEX: Explain individual predictions

```
pb1 <-
prediction_br
eakdown(expla
iner1, row1)
```



- Indicates the contribution of each variable to the model prediction

Exercise

- Can you improve the fit of the linear model?
- Can you improve the fit of the random forest?
- What relationships can you find in the data?

(e.g. consider interaction terms and other transformations)

Further Learning – Model Visualisations

- DALEX Documentation: https://www.rdocumentation.org/packages/DALEX/versions/0.2.4
- Josh Jaroudy's presentation from the 2018 DA Seminar: https://actuaries.logicaldoc.cloud/download-ticket?ticketId=88dc3a68-095f-4b4c-90a2-b308fa487146

Advanced Visualisation

Take Home Exercise – Advanced Visualisation

- Advanced visualisation (in the pack)
  - Learn about literate programming and R notebooks
  - Create an interactive dashboard using flexdashboard and R Shiny
- Can you make a dashboard that takes in inputs and creates a "DALEX: Explain individual predictions" chart from it?

Thank you – what's next?

Next Event – YAP and YDAWG Case Study

Date: Wednesday, 20th March 2019

Time: 3:30pm – 7:30pm

Venue: Actuaries Institute

Limited spots available

Registrations will open tomorrow