PORTFOLIO

KA MAN CHENG Kabay

kabay@live.hk
+44 07 393939 726
linkedin.com/in/kabayc
MSc in Data-Intensive Analysis, University of St Andrews

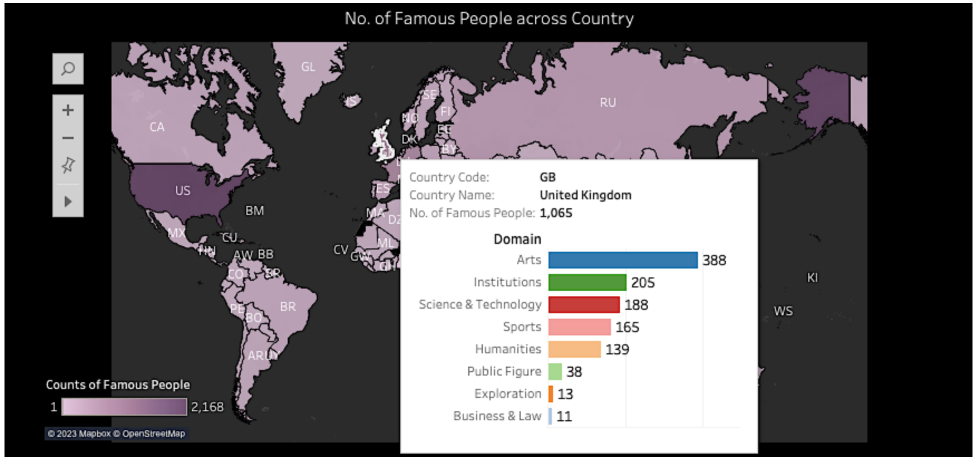
#DataVisualisation

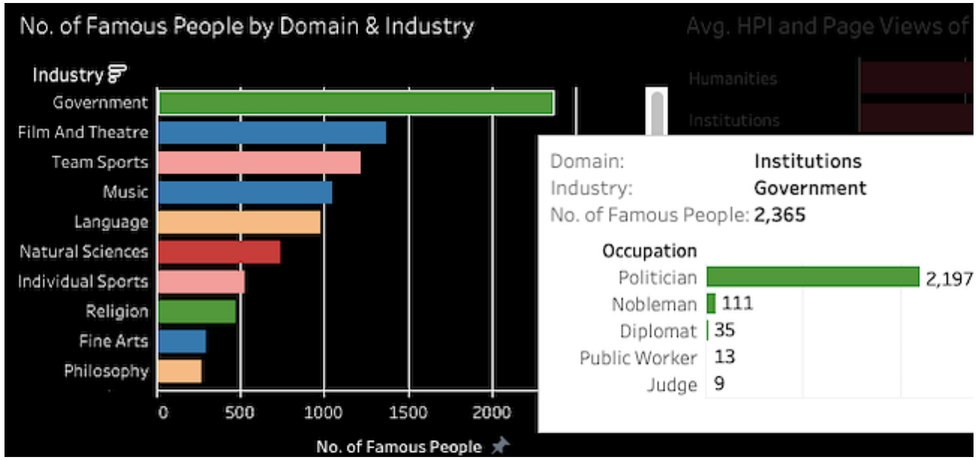
Tableau Dashboard for Pantheon /February 2022

This Tableau dashboard was built for a practical exercise in an information visualization course. It was designed to visualise the Pantheon dataset, which is a manually verified dataset that includes 11,341 globally famous biographies.

The visualization facilitated the exploration of the data and helped extract insights to answer the following questions:
1. *What are the demographic characteristics and occupational backgrounds of the famous biographies?*
2. *How do the relationships between HPI and page-views vary by biographies' expertise and birthplace?*

The work combines different types of charts into one dashboard and utilises a geographic map, hover-over functions, and some interactive elements such as a slider for filtering, providing more dynamic views of the data.





Total No. of Famous People
11,341
-3500 - -3001
-3000 - -2501
-2500 - -2001
-2000 - -1501
-1500 - -1001
-1000 - -501
-500 - -1
0 - 499
500 - 999
1000 - 1499
1500 - 2005
Birth Year
-3500
2005
Gender
(All)
Female
Male
Continent
(All)
Africa
Asia
Europe
North America
Oceania
South America
9,390
1,481
Female
Male
No. of Famous People across Country
Counts of Famous People
1
2,169
© 2023 Mapbox © OpenStreetMap
Domain
Arts
Business & Law
Exploration
Humanities
Institutions
Public Figure
Science & Technology
Sports
No. of Famous People by Domain & Industry
Industry
Government
Film And Theatre
Team Sports
Music
Language
Natural Sciences
Individual Sports
Religion
Fine Arts
Philosophy
0
500
1000
1500
2000
No. of Famous People
Avg. HPI and Page Views of Famous People by Domain
Humanities
Institutions
Exploration
Science & Technology
Public Figure
Business & Law
Arts
Sports
0M
2M
4M
6M
8M
Avg. Page Views
By Domain / Continent
Domain
Avg. HPI
17.709
24.327

(Google Data Studio?)

#R #StatisticalComputing

X /March 2022

X

Solution

Setup

To predict `price` from other given variables, first load relevant libraries and dataset for modelling with Random Forest.

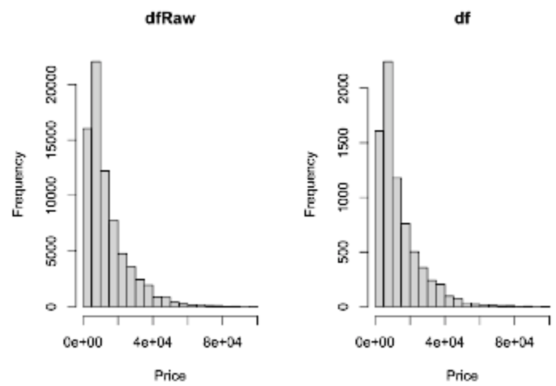
```r
# Load relevant libraries and data
library(tidyverse)
library(car) # Anova(.)
library(randomForest) #randomforest(.)
library(caret)

vehicles <- read.csv("vehicles.csv", na.strings=c("NA", "NaN", ""))
```

Before start using the dataset, some data cleaning is needed.

```r
# Remove irrelevant columns and rows that with NA
dfRaw <- na.omit(select(vehicles, -url, -region_url, -VIN, -image_url, -description,
                        -county, -lat, -long, -posting_date))
# Filter out rows with extreme high / low price
dfRaw <- dfRaw %>% filter(price > 999, price < 100000)
# Set covariate "year" as factor
dfRaw$year <- as.factor(dfRaw$year)

# Use systematic sampling method to get a subset of the dataset for a faster run of model
df <- dfRaw %>% filter(row_number() %% 10 == 1)
head(df)
```

```
##           id     region price year manufacturer      model condition   cylinders
## 1 7316356412     auburn 15000 2013         ford  f-150 xlt excellent 6 cylinders
## 2 7316868220 birmingham 10950 2009        lexus      rx350 excellent 6 cylinders
## 3 7316031977 birmingham 45000 2017    chevrolet  silverado excellent 8 cylinders
## 4 7315259946 birmingham 98900 2001      ferrari 360 modena      good 8 cylinders
## 5 7314833347 birmingham  4000 2002       toyota      camry  like new 4 cylinders
## 6 7314334186 birmingham  6950 2011   volkswagen      jetta excellent 5 cylinders
##   fuel odometer transmission drive      size        type paint_color state
## 1  gas   128000    automatic   rwd full-size       truck       black    al
## 2  gas   191955    automatic   4wd full-size         SUV       white    al
## 3  gas    92000    automatic   4wd full-size       truck         red    al
## 4  gas    20187    automatic   rwd  mid-size convertible         red    al
## 5  gas   160000    automatic   fwd full-size       sedan       white    al
## 6  gas   116000    automatic   fwd full-size       sedan      silver    al
```

```r
# Compare the price distribution for the original dataset and its subset
par(mfrow=c(1, 2)); hist(dfRaw$price, main = "dfRaw", xlab = "Price")
hist(df$price, breaks = 20,  main = "df", xlab = "Price"); par(mfrow=c(1, 1))
```



As the two distributions looks similar, the subset is trustworthy and be used for modelling.

Exploratory Data Analysis

Fitting simple linear model to pick 4 most important covariates for modelling with Random Forest.

```
# Fitting simple linear model with all covariates (dropping off those with large df)
lm <- lm(price ~ . -region -year -model, data = df)
Anova(lm)
```

```
## Anova Table (Type II tests)
##
## Response: price
##                   Sum Sq   Df  F value    Pr(>F)
## id             8.5642e+08    1  10.9968 0.0009172 ***
## manufacturer 2.3701e+10   40   7.6083 < 2.2e-16 ***
## condition    6.8811e+10    5 176.7126 < 2.2e-16 ***
## cylinders    4.4352e+09    7   8.1357 6.714e-10 ***
## fuel         4.8408e+10    4 155.3954 < 2.2e-16 ***
## odometer     2.7980e+10    1 359.2758 < 2.2e-16 ***
## title_status 2.8971e+09    5   7.4401 5.686e-07 ***
## transmission 7.4234e+07    2   0.4766 0.6209115
## drive        1.2501e+10    2  80.2567 < 2.2e-16 ***
## size         1.3698e+09    3   5.8629 0.0005398 ***
## type         2.3365e+10   12  25.0014 < 2.2e-16 ***
## paint_color  1.0545e+10   11  12.3094 < 2.2e-16 ***
## state        1.7612e+10   50   4.5228 < 2.2e-16 ***
## Residuals    5.6260e+11 7224
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pick the 4 most important covariates based on the least *p*-values. Here, `condition`, `fuel`, `odometer`, `drive` will be used for the following modelling.

Random Forest

Prepare a training data and a test data (80/20 split) from the dataset, then modelling the data with Random Forest. Start with fewer trees using `ntree = 25`. Test for the possible no. of variables randomly sampled as candidates at each split, i.e. `mtry = 3`, `mtry = 2` and `mtry = 1`, and also try on using all selected covariates, i.e. Bagging - `mtry = 4`.

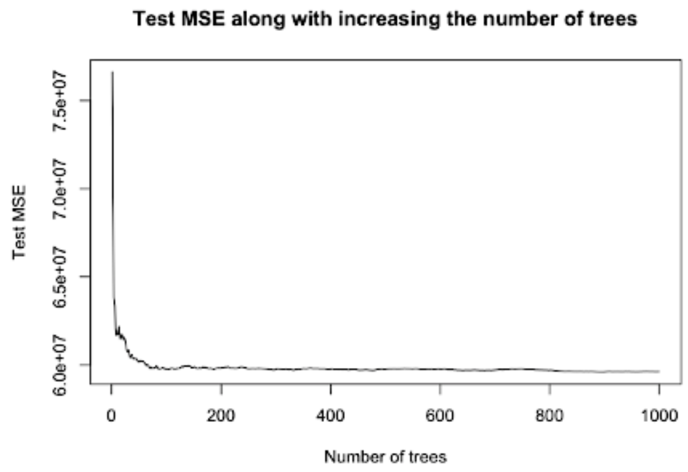
```
set.seed(5986)
# Separate out training and test data (80/20 split)
trainIndex <- sample(1:nrow(df), round(nrow(df)*0.8))
testResponse <- df[-trainIndex, "price"]

# Start with fewer trees, and randomly sample m (with m < p) covariates as splitting candidates
vehiclesRF = list(); predict = list(); mse = list()
for(i in 1:4) {
  rf <- randomForest(price ~ condition + fuel + odometer + drive, data = df, subset = trainIndex,
                     mtry = i, ntree = 25)
  vehiclesRF[[length(vehiclesRF) + 1]] <- rf

  # Predict to the test data
  pred <- predict(vehiclesRF[i], newdata = df[-trainIndex, ])
  predict[[length(predict) + 1]] <- pred
  # Test MSE
  testMSE <- mean((unlist(predict[i]) - testResponse)^2)
  mse[[length(mse) + 1]] <- c("mtry", i, testMSE)
}
mse
```

```
## [[1]]
## [1] "mtry"              "1"                 "68743831.3706561"
## [[2]]
## [1] "mtry"              "2"                 "60921466.9568478"
## [[3]]
## [1] "mtry"              "3"                 "63514437.8664374"
## [[4]]
## [1] "mtry"              "4"                 "68504852.8010765"
```

Result shows that the model of Random Forest using `mtry = 2` returns with the lowest MSE.

With sampling the covariates using `mtry = 2`, use out-of-bag (a generalization error) and store the test errors to see the how it changes with increasing the size of trees.

```
# Use OOB to pass test set in to store test errors as progress
testCov <- df[-trainIndex, ] %>% dplyr::select(-price, -id, -region, -year, -manufacturer, -model, -cylinders,
                                               -title_status, -transmission, -size, -type, -paint_color, -state)
# Try with more trees, say 1,000
vehiclesRF2 <- randomForest (price ~ condition + fuel + odometer + drive, data = df, subset = trainIndex,
                             xtest = testCov, ytest = testResponse, mtry = 2, ntree = 1000)
vehiclesRF2TestError <- vehiclesRF2$test$mse
plot(vehiclesRF2TestError, type = "l", ylab = "Test MSE", xlab = "Number of trees",
     main = "Test MSE along with increasing the number of trees")
```



The graph indicates that should be having around 100 trees for the test error to plateau, here the following will be using 500 trees.

```
vehiclesRF2 <- randomForest(price ~ condition + fuel + odometer + drive,  data = df, subset = trainIndex,
                            mtry = 2, ntree = 500)
# Predict to the test data
pred <- predict(vehiclesRF2, newdata = df[-trainIndex, ])
head(setNames(data.frame(df[-trainIndex, "id"], df[-trainIndex, "price"], pred), c("id", "price", "prediction")))
```

```
##             id price prediction
## 2  7316868220 10950   9967.264
## 4  7315259946 98900  42926.945
## 7  7313889508 11500  12396.244
## 12 7311505593  1800   3449.779
## 13 7311387137 24990  13929.614
## 28 7305217525  5500   8033.186
```

```
# test MSE
mean((pred - testResponse)^2)
```

```
## [1] 59514971
```

Result shows that the model performance has been improved, with a better (lower) test error when increasing the size of trees.

#HTML #CSS #JavaScript

Websites – CityU (link), St Andrews (movie and card)

#DigitalMarketing

Google/Bing/FB/IG