Choose Your Own project - Using UK census data to predict occupations

Paul Ceely

26/08/2020

Contents

1 Executive summary

final project from the **EDX course HarvardX: PH125.9x Data Science: Capstone**.

machine learning. inspired by the ... UCI machine learning repository, Black lives matter movement and reporting. developed a new data set from the raw census data to enable answering the following question:

- predict the proportion of working age people each area that are one of the managerial and professional occupations

Focus was on the Middle layer Super Output Areas - 2011" (MSOA) which gives around 7000 areas to work with MSOA areas, but the models and methodology were tested on a subset of the SOA at the end.

For this project, approach was to look at a large number of models to identify which ones were most useful and then build an ensemble.

summary tables of the outcome

top 24 or so seemed most important

Good prediction accuracy for the final model for MSOA. SOA not so much. entir

The document is long, as it includes the majority of the code, which is in turn long due to the large amount of data wrangling to put the data set together from the ran census information, and additionally due to trying modelling on a large number of algorithms to identify the best for the final ensemble.

---

2 Introduction

This project is part of the **EDX course HarvardX: PH125.9x Data Science: Capstone**, which builds on the previous 8 covering using R, statistics, regression, machine learning among other data science topics, presented by Professor Rafael Irizarry, as described here https://www.edx.org/professional-certificate/harvardx-data-science.

The project is inspired by the UCI machine learning repository, to predict wages based on census data, here https://archive.ics.uci.edu/ml/datasets/Adult, and also the recent Black Lives Matter movements and recognition of the under-representation of certain groups in senior positions within society. https://www.theguardian.com/business/2020/jul/28/bame-representation-uk-top-jobs-colour-of-power-survey

Instead this project focuses on the UK, and uses the UK 2011 census data. This data does not have individual information, instead numbers within a geographical area, and in addition, it does not include pay, but does include occupation.

Therefore the challenge is to:

- predict the proportion of working age people each area that are one of the managerial and professional occupations

As the target it to predict the proportion in an area, this can be treated as continuous problem, even though the actual underlying decision is a categorical, with a binary outcome, professional/managerial or not.

The accuracy / error evaluation methodology will be developed through the work, but initially, as this is a continuous problem, it would be appropriate to use a loss function. The simplest is the root mean squared error (RMSE) for each geo location area.

As well as a machine learning exercise, the project aims to gain some understanding of the underlying causes of differences, and therefore there will be a preference for techniques that permit some interpretation.

2.1 Key steps

Key steps carried out:

1. Identify potentially useful data from the UK Census
2. Import data, clean up, create the values to predict
3. Create project data set and the validation set
4. Initial investigation and visualisation
5. Create a training and test set
6. Further data exploration including baseline of prediction
7. Training and testing a variety of models
8. Building an ensemble
9. Finalising a model and running on the validation set
10. Run on more detailed data set to confirm the models
11. Interpretation, and investigation in to feature importance

The methodology and code was developed using insight gained from the course, and also in the accompanying text book **An Introduction to Data Science, by Rafael A Irizarry, https://rafalab.github.io/dsbook/**.

3 Methodology and data preparation

3.1 Identification of data sources

The most recent census in the UK is in 2011. The information on the Office for National Statistics (ONS) was used as a starting point for the data, here https://www.ons.gov.uk/census/2011census/2011censusdata/2011censususerguide/variablesandclassifications.

This links to the Nomis web site, provided buy Durham University on behalf of the ONS, as the source of official labour market statistics. The following link provides a list of the tables from the 2011 UK census https://www.nomisweb.co.uk/census/2011/data_finder

The most granular data is provided for "super output area" (SOA) an area defined for the census, of at least 100 people, but aiming for larger than 125 households, which provides over 130,000 data points providing a good level of detail and variability. This is a very large data set, and will be challenging to develop a new methodology for it, and therefore this study will focus on a smaller data set, however, the SOA will be tested at the end to verify that the methodology works.

The Middle layer Super Output Areas - 2011" (MSOA) is a suitable level or granularity, with good detail from the various tables, with around 7000 areas to work with.

For the initial data processing and preparation, cleanup and cross checking the numbers between the tables the "regions" geographic areas will be used for ease of troubleshooting.

There appears to be more detailed data available for England and Wales, enabling further clean up (for example, to baseline on the same totals, working age and so on), and therefore this will be used.

The following are the data tables from the census used to construct the data. The focus is on indicators, trying to avoid information which may be due to the effects of having a good job such as size of house and so on. The UCI data set also provided guidance on suitable data to be developed in to the features.

Table 1: Census variables with name and location of the data tables

Census variable	URL of data table
Occupation, (1 to 9 options)	https://www.nomisweb.co.uk/census/2011/ks608ew
Age	https://www.nomisweb.co.uk/census/2011/ks102ew
Ethnicity and sex/gender	https://www.nomisweb.co.uk/census/2011/ks201ew
	https://www.nomisweb.co.uk/census/2011/lc2101ew
Qualifications - highest	https://www.nomisweb.co.uk/census/2011/ks501ew
	https://www.nomisweb.co.uk/census/2011/lc5102ew
Marital status	https://www.nomisweb.co.uk/census/2011/ks103ew
	https://www.nomisweb.co.uk/census/2011/lc1101ew
Religion	https://www.nomisweb.co.uk/census/2011/ks209ew
	https://www.nomisweb.co.uk/census/2011/lc2107ew
Household composition	https://www.nomisweb.co.uk/census/2011/ks105ew
	https://www.nomisweb.co.uk/census/2011/lc1109ew
Industry	https://www.nomisweb.co.uk/census/2011/ks605ew
	https://www.nomisweb.co.uk/census/2011/lc6110ew
Country of birth	https://www.nomisweb.co.uk/census/2011/qs203ew
	https://www.nomisweb.co.uk/census/2011/lc2103ew

Other potential predictors could be:

- Housing type
- Possibly related to how much money earned, so effect rather than cause?

It may be possible to map the occupation to earnings with other data, such as: https://www.ons.gov.uk/dat

asets/ashe-table-8-earnings/editions/time-series/versions/1#id-dimensions.

3.2 Data preparation

This section describes the source data and how it is converted in to the features used in the modelling, as well as initial data exploration.

In summary, this study will use the MSOA geographic areas for England and Wales, with data sourced from 9 tables from the nomis census data published online.

Working wih the occupation data initially, the target values will be created as a proportion of the residents in an area, so a fraction of 1. The other data sources will be addressed in a similar way to produce the features, developing values that are proportions of residents, trying to use a similar cohort as the occupation data.

3.2.1 Data import

A function was created to download the files from Nomis, which for a given geography type and table name, saved the csv files to the ~/data directory.

```
# test geo type
geographytype <- "TYPE480" #regions
# function to create download file from nomis
nomis_census_csv <- function(censusdata, geographytype){
  # generate filename
  filename <- paste(censusdata, geographytype, sep = "_") %>%
    paste0(., ".csv")
  destfile <- paste0(getwd(),"/data/", filename)
  # generate URL
  nomis_url_root <- "http://www.nomisweb.co.uk/census/2011/"
  nomis_url <- paste0(nomis_url_root, censusdata)
  # open http session
  nomis_session <- html_session(nomis_url)
  # get a copy of the form for the csv downloads
  nomis_form <- html_form(nomis_session)[[3]]
  # fill in the form, using the supplied details
  filled_form <- set_values(form = nomis_form, geography = geographytype)
  # request the data
  nomis_session2 <- submit_form(session = nomis_session,
                                form = filled_form)
  #download the file
  download.file(nomis_session2$url, destfile)
}
```

In order to download all the relevant census tables at once,a loop was used to cycle through a list of tables, downloading all the tables for a specified geography type (SOA, MSOA etc.).

```
# then TYPE297 'super output areas - middle layer 2011
geographytype <- "TYPE297" #super output areas - middle layer
# geographytype <- "TYPE499" #countries
#creating list of census tables to take
censusdata_list <- c("ks101ew", "ks608ew","ks102ew", "ks201ew","ks501ew",
                     "ks103ew", "ks209ew", "ks105ew", "ks605ew", "qs203ew")
# apply the list to download all of the data
tmp <- lapply(censusdata_list, function(censusdata){
  # pause for a few random seconds to avoid annoying nomis
  time <- sample(4:17,1)
  print(paste("pausing", time, "seconds"))
```

```
  Sys.sleep(time)
  #run the function to generate and download the file
  nomis_census_csv(censusdata, geographytype)
})
```

Finally a function was created to import the data in to R from the saved csv files.

```
ingest <- function(censustable, geography){
  print(paste("ingesting", censustable))
  # generate file name for loading
  filename <- paste(censustable, geography, sep = "_") %>%
    paste0(., ".csv") %>%
    paste0("data/", .)
  print(filename)
  #read the file to data
  # data <- read_csv(filename)
  return(read_csv(filename))
}
```

3.2.2 Creating the value to predict

This requires working with the occupation information in table ks608.

There are 9 Occupation types, and we are combining the first two and the remaining 7:

- Occupation: 1. Managers, directors and senior officials
- Occupation: 2. Professional occupations
- Occupation: 3. Associate professional and technical occupations
- Occupation: 4. Administrative and secretarial occupations; measures
- Occupation: 5. Skilled trades occupations
- Occupation: 6. Caring, leisure and other service occupations
- Occupation: 7. Sales and customer service occupations
- Occupation: 8. Process plant and machine operatives
- Occupation: 9. Elementary occupations

The y value is then simply the ratio of the sum of Occupations 1 and 2, and the sum of all occupations.

```
#load in occupation data
# https://www.nomisweb.co.uk/census/2011/ks608ew
# import the new table with occupation data
data <- ingest("ks608ew")
names(data)
#now creating a function for repeatability
occupation_target <- function(data){
  occupation <- data %>%
    rename("geo_code" = "geography code",
           "geo_name" = "geography",
           "geo_type" = "Rural Urban",
           "occupation_all" = "Sex: All persons; Occupation: All categories: Occupation; measures: Valu
    rowwise() %>%
    # calculate the ratio of managers and professionals
    mutate(y =
       (sum(across(contains("Sex: All persons; Occupation: 1. Managers"))) +
        sum(across(contains("Sex: All persons; Occupation: 2. Professional "))) )
          /occupation_all) %>%
    select(geo_name, geo_code, geo_type, y, occupation_all)
```

```
  #save file
  save(occupation, file='rda/occupation.rda')
}
#running the function on the data set
occupation_target(data)
#adding to main data set
load('rda/occupation.rda')
data_set <- occupation
```

A short function was created to save the data set with the geography area name in it.

```
# save data set with the geography type
data_set_save <- function(data_set_name){
  print(paste("saving", deparse(substitute(data_set_name))))
  # generate file name for saving
  filename <- paste(deparse(substitute(data_set_name)),
                    geographytype,
                    sep = "_") %>%
    paste0(., ".rda") %>%
    paste0("rda/", .)
  print(paste("saving as file", filename))
  # save the file to data
  save(data_set_name, file=filename)
}
```

3.2.3 Sex/gender feature

Starting with the Sex data set, which also includes the total number of people. It can be seen that the total number of residents listed in the Occupation table is roughly half of the total in this table. On the other hand, the Sex/gender data is fairly static, with more women then men everywhere. The female ratio calculated is for the full population rather than the Occupation group.

```
#load in residents and sex/gender data
data <- ingest("ks101ew", geographytype)
names(data)
sex <- data %>%
  # rename geo code column
  rename("geo_code" = "geography code") %>%
  rename("all_residents" = "Variable: All usual residents; measures: Value") %>%
  rename("females" = "Variable: Females; measures: Value") %>%
  mutate(female_ratio =  females/all_residents) %>%
  select(geo_code, all_residents, female_ratio)
names(sex)
#adding to main data set
data_set <- data_set %>%
  left_join(sex)
names(data_set)
```

3.2.3.1 Age feature There are many columns, for a predictor, Median will be used as it has more variation than mean.

In order to have the baseline for the features correct, it is necessary to understand the difference in numbers of residents in the occupation table, which may be working age.

Typically in the census, there is a breakdown of 16-74, as well as 25-64, but those age ranges are still larger than the Occupation total, and excerpt of the table is included.

```
# load in age data
# https://www.nomisweb.co.uk/census/2011/ks102ew
# import the new table with age data
data <- ingest("ks102ew", geographytype)
names(data)
#look at all of the ages, to find the right ones to work on
age <- data %>%
  rename_at(vars(contains("Age")), ~str_replace_all(., "; measures: Value", "")) %>%
  rename_at(vars(contains("Age")), ~str_replace_all(., "Age: Age ", "age_")) %>%
  rename_at(vars(contains("Age")), ~str_replace_all(., " to ", "_")) %>%
  rename_at(vars(contains("over")), ~str_replace_all(., " and ", "_")) %>%
  rename("geo_code" = "geography code",
         "all_residents" = "Age: All usual residents",
         "age_median" = "Age: Median Age") %>%
  mutate("under_16" = age_0_4 + age_5_7 + age_8_9 + age_10_14 + age_15) %>%
  mutate("over_74" = age_75_84 + age_85_89 + age_90_over) %>%
  mutate("age_16_to_74" = all_residents - over_74 - under_16) %>%
  mutate("age_25_to_64" = (age_16_to_74 - age_16_17 - age_18_19 - age_20_24 - age_65_74)) %>%
  select(geo_code, all_residents, age_16_to_74, age_25_to_64, age_median)
names(age)
data_set %>%
#comparing to the all occupation number in the data set
  left_join(age) %>%
  select(geo_name, occupation_all, all_residents, age_16_to_74, age_25_to_64) %>%
  knitr::kable(caption="Comparison of resident numbers")
age <- age %>%
  select(-age_16_to_74, -age_25_to_64)
```

Table 2: Comparison of resident numbers

geo_name	occupation_all	all_residents	age_16_to_74	age_25_to_64
North East	1152970	2596886	1924206	1362756
North West	3228744	7052177	5184216	3697881

Investigating further, there is an Economic activity table, https://www.nomisweb.co.uk/census/2011/ks601ew, which can be compared.

The number of residents in the Occupation table is between the "Economically Active" and the "Economically active: In employment" values. It may be possible to accurately baseline by extracting from combined tables, so for example, use a table with Economic activity and Sex. However, it may be that a resident had the potential to be a manager or professional although not currently economically active, and in addition, the combinations of data required may restrict other options for features. Therefore the project will use working age residents, preferring 25 to 64 where available, as it is closer to the value of the occupation table.

```
# checking economic activity
nomis_census_csv("ks601ew", geographytype)
data <- ingest("ks601ew")
names(data)
economic <- data %>%
  rename_at(vars(contains("Sex")),
            ~str_replace_all(., "; measures: Value", "")) %>%
  rename_at(vars(contains("Sex")),
            ~str_replace_all(., "Sex: All persons; Economic Activity: ", "")) %>%
```

```
  select(2:22) %>% select(-"Rural Urban") %>%
  rename("geo_code" = "geography code", "geo_name" = "geography")
# names(economic)
#table comparing the numbers
economic %>%
  left_join(data_set) %>%
  select(-geo_type, -y, -female_ratio, -occ_ratio, -age_median) %>%
  select(geo_name, geo_code, "All usual residents aged 16 to 74",
  "Economically active", "Economically active: In employment",
  "occupation_all", "all_residents") %>%
  knitr::kable()
```

Table 3: Comparison of resident numbers and employment

geo_name	all_res_16_74	econ_active	employed	occupation_all	all_residents
North East	1924206	1272082	1105909	1152970	2596886
North West	5184216	3515910	3089895	3228744	7052177

The code to create the age feature is very simple, using the median age column.

```
# load in age data
# https://www.nomisweb.co.uk/census/2011/ks102ew
# import the new table with age data
data <- ingest("ks102ew", geographytype)
# rewriting as a function for the median age only
age_predictors <- function(data){
  age <- data %>%
    rename("geo_code" = "geography code",
           "all_residents" = "Age: All usual residents; measures: Value",
           "age_median" = "Age: Median Age; measures: Value") %>% select(geo_code, all_residents, age_m
  save(age, file='rda/age.rda')
}
#running the function on the data set
age_predictors(data)
#adding to main data set
load('rda/age.rda')
data_set <- data_set %>%
  left_join(age)
# tidy
data_set_save(data_set)
```

3.2.4 Ethnicity feature

In order to choose just the working age people, a table was required with both age and ethnicity, with summarisation to the 25-64 age group. There are many subdivisions, the top 11 ethnicities, over of 0.5% population were chosen, with others aggregated following the structure used by the Census. This was chosed as there are reported differences for outcomes for Bangladeshi Britons which is the 11th ethnicity. The top ones are:

```
# reordering to identify which are the largest groups to retain
load('rda/ethicity_raw.rda')
# make the geo code a rowname - all data is now a value
ethnicity_ordered <- column_to_rownames(ethnicity_raw, var="geography_code")
```

```r
# str(ethnicity_ordered)
# add a new row with column sums, so totals for each ethnicity
ethnicity_ordered <- ethnicity_ordered %>%
  rbind("total" = colSums(ethnicity_ordered))
# reorder columns based on the values in the total row
ethnicity_ordered <- ethnicity_ordered[, order(-ethnicity_ordered[which(rownames(ethnicity_ordered) ==
# choose the first 11 columns, the first 11 ethnicities
data.frame("ethnicity categories" = names(ethnicity_ordered)[2:12]) %>%
  knitr::kable(caption="11 most prevalent ethnicities")
```

Table 4: 11 most prevalent ethnicities

ethnicity.categories
age25_64_All_categories_Ethnic_group
age25_64_White_English_Welsh_Scottish_Northern_Irish_British
age25_64_White_Other_White
age25_64_Asian_Asian_British_Indian
age25_64_Asian_Asian_British_Pakistani
age25_64_Black_African_Caribbean_Black_British_African
age25_64_Asian_Asian_British_Other_Asian
age25_64_Black_African_Caribbean_Black_British_Caribbean
age25_64_White_Irish
age25_64_Asian_Asian_British_Chinese
age25_64_Asian_Asian_British_Bangladeshi

This required the following aggregation which will be referred to throughout the rest of the document.

Table 5: Ethnicity features and census categories

Feature name	Census category
white_uk_25_64	White: English/Welsh/Scottish/Northern Irish/British
white_other_25_64	White: Other White
	White: Gypsy or Irish Traveller
white_irish_25_64	White: Irish
indian_25_64	Asian/Asian British: Indian
pakistani_25_64	Asian/Asian British: Pakistani
bangladeshi_25_64	Asian/Asian British: Bangladeshi
other_ethnicity_25_64	Other ethnic group: Any other ethnic group
	Mixed/multiple ethnic group: White and Black Caribbean
	Other ethnic group: Arab
	Mixed/multiple ethnic group: Other Mixed
	Mixed/multiple ethnic group: White and Asian
	Mixed/multiple ethnic group: White and Black African
chinese_25_64	Asian/Asian British: Chinese
asian_other_25_64	Asian/Asian British: Other Asian
black_other_25_6	Black/African/Caribbean/Black British: Caribbean
	Black/African/Caribbean/Black British: Other Black
black_african_25_64	Black/African/Caribbean/Black British: African

After this aggregation, the numbers of residents were divided by the sums of the rows, to have a normalised value for each of the ethnicities, as a ratio of the residents in the area. This created long code to tidy and

aggregate the data, as below.

```
# next to load in the ethnicity data
# https://www.nomisweb.co.uk/census/2011/lc2101ew
# updating with age and ethnicity
nomis_census_csv("lc2101ew", geographytype)
data <- ingest("lc2101ew", geographytype)
# creating a function for repeatability
ethnicity_predictors <- function(data){
  print("manipulating raw data")
  ethnicity_raw <- data %>%
    #keeping only the data relating to age 25 to 64
    select(contains("geography code") |
             contains("Age 25 to 49") |
             contains("Age 50 to 64") ) %>%
    # adding up the year groups
    rowwise() %>%
    mutate("all_25_64" =
             sum(across(contains("All categories: Ethnic group"))) ) %>%
    mutate("white_uk_25_64" =
             sum(across(contains("White: English/Welsh/Scottish/Northern Irish/British"))) ) %>%
    mutate("white_other_25_64" =
             sum(across(contains("White: Other White"))) +
             sum(across(contains("White: Gypsy or Irish Traveller"))) ) %>%
    mutate("white_irish_25_64" =
             sum(across(contains("White: Irish"))) ) %>%
    mutate("indian_25_64" =
             sum(across(contains("Asian/Asian British: Indian"))) ) %>%
    mutate("pakistani_25_64" =
             sum(across(contains("Asian/Asian British: Pakistani"))) ) %>%
    mutate("bangladeshi_25_64" =
             sum(across(contains("Asian/Asian British: Bangladeshi"))) ) %>%
    mutate("other_ethnicity_25_64" =
             sum(across(contains("Other ethnic group: Any other ethnic group"))) +
             sum(across(contains("Mixed/multiple ethnic group: White and Black Caribbean"))) +
             sum(across(contains("Other ethnic group: Arab"))) +
             sum(across(contains("Mixed/multiple ethnic group: Other Mixed"))) +
             sum(across(contains("Mixed/multiple ethnic group: White and Asian"))) +
             sum(across(contains("Mixed/multiple ethnic group: White and Black African")))) %>%
    mutate("chinese_25_64" =
             sum(across(contains("Asian/Asian British: Chinese"))) ) %>%
    mutate("asian_other_25_64" =
             sum(across(contains("Asian/Asian British: Other Asian"))) ) %>%
    mutate("black_other_25_64" =
             sum(across(contains("Black/African/Caribbean/Black British: Caribbean"))) +
             sum(across(contains("Black/African/Caribbean/Black British: Other Black"))) ) %>%
    mutate("black_african_25_64" =
             sum(across(contains("Black/African/Caribbean/Black British: African"))) ) %>%
    rename("geo_code" = "geography code") %>%
    # print(names(ethnicity)) %>%
    select(contains("geo_code") | contains("25_64")) %>%
    # print(names(ethnicity))
  # After this aggregation, need to normalise the value for each ethnicity
  # divide each value by the sums of the rows
```

```
  print("normalising the data")
  ethnicity <- sweep(ethnicity_raw[,3:13], 1, rowSums(ethnicity_raw[,3:13]), FUN = "/")  %>%
    cbind(ethnicity_raw[,1])
  print("saving the file")
  save(ethnicity, file='rda/ethnicity.rda')
}
#running the function on the data set
ethnicity_predictors(data)
#adding to main data set
load('rda/ethnicity.rda')
data_set <- data_set %>%
  left_join(ethnicity)
names(data_set)
```

3.2.5 Sex/gender feature in the working age population

The earlier feature for sex/gender was the ratio for the full population, but the data should be consistent, and therefore use the same 25 to 64 group as for ethnicity feature. This can be carried out using the same source data tables.

```
# next to load in the sex/gender data, for ages 25 to 64
# https://www.nomisweb.co.uk/census/2011/lc2101ew
# using the ethnicity data, which includes gender
data <- ingest("lc2101ew", geographytype)
#creating as a function for reuse
sex_predictors <- function(data){
  sex_25_64 <- data %>%
    select(contains("Ethnic Group: All categories:") |
             contains("geography code") ) %>%
    select(contains("Age 25 to 49") |
             contains("Age 50 to 64") |
             contains("geography code") ) %>%
    rowwise() %>%
    #create the ratio necessary
    mutate(female_ratio_25_64 = sum(across(contains("Sex: Female"))) /
             sum(across(contains("Sex: All Persons")))) %>%
    rename("geo_code" = "geography code")  %>%
    select(contains("female_ratio_25_64") | contains("geo_code") )
  save(sex_25_64, file='rda/sex_25_64.rda')
}
#running the function on the data set
sex_predictors(data)
#adding to main data set
load('rda/sex_25_64.rda')
data_set <- data_set %>%
  left_join(sex_25_64)
# show the data set values
names(data_set)
```

This reduces the ratio a little, still in favour of women but less so, presumably due to women living longer.

3.2.6 Qualification feature

Using the same methodology as before, ratios for each qualification type as a proportion of the 25 to 64 aged residents in each geo area were created. The census qualification categories were used, as follows, a total of 8:

- No qualifications;
- Level 1: 1-4 O Levels/CSE/GCSEs (any grades), Entry Level, Foundation Diploma, NVQ Level 1, Foundation GNVQ, Basic/Essential Skills;
- Level 2: 5+ O Level (Passes)/CSEs (Grade 1)/GCSEs (Grades A*-C), School Certificate, 1 A Level/ 2-3 AS Levels/VCEs, Intermediate/Higher Diploma, Welsh Baccalaureate Intermediate Diploma, NVQ level 2, Intermediate GNVQ, City and Guilds Craft, BTEC First/General Diploma, RSA Diploma;
- Apprenticeship;
- Level 3: 2+ A Levels/VCEs, 4+ AS Levels, Higher School Certificate, Progression/Advanced Diploma, Welsh Baccalaureate Advanced Diploma, NVQ Level 3; Advanced GNVQ, City and Guilds Advanced Craft, ONC, OND, BTEC National, RSA Advanced Diploma;
- Level 4 and above: Degree (for example BA, BSc), Higher Degree (for example MA, PhD, PGCE), NVQ Level 4-5, HNC, HND, RSA Higher Diploma, BTEC Higher level, Foundation degree (NI), Professional qualifications (for example teaching, nursing, accountancy);
- Other qualifications: Vocational/Work-related Qualifications, Foreign Qualifications (not stated/level unknown).

Source https://www.nomisweb.co.uk/census/2011/lc5102ew, clicking on the "Highest level of qualification" link.

```
# qualifications
# https://www.nomisweb.co.uk/census/2011/lc5102ew
# import the new table with qualifications
nomis_census_csv("lc5102ew", geographytype)
data <- ingest("lc5102ew", geographytype)
names(data)
#creating a function for repeatability
qualifications_predictors <- function(data){
  qualifications_raw <- data %>%
    #save only the 25 to 64 years
    select(contains("Age 25 to 34") | contains("Age 35 to 49") | contains("Age 50 to 64") | contains("g
    rename("geo_code" = "geography code") %>%
    #calculate the sums for the retained predictors
    rowwise() %>%
    mutate("all_qual_25_64"= sum(across(contains("All categories: ")))) %>%
    mutate("no_qual_25_64"= sum(across(contains("No qualifications")))) %>%
    mutate("level1_25_64"= sum(across(contains("Level 1 qualification")))) %>%
    mutate("level2_25_64"= sum(across(contains("Level 2 qualification")))) %>%
    mutate("level3_25_64"= sum(across(contains("Level 3 qualification")))) %>%
    mutate("level4_25_64"= sum(across(contains("Level 4 qualification")))) %>%
    mutate("other_qual_25_64"= sum(across(contains("Other qualifications")))) %>%
    mutate("apprentice_25_64"= sum(across(contains("Apprenticeship")))) %>%
    select( contains("geo_code") | contains("25_64"))
  # normalise per area
  qualifications <- sweep(qualifications_raw[,3:9], 1, rowSums(qualifications_raw[,3:9]), FUN = "/")  %
    cbind(qualifications_raw[,1])
  #save file
  save(qualifications, file='rda/qualifications.rda')
}
#running the function on the data set
qualifications_predictors(data)
#adding to main data set
load('rda/qualifications.rda')
data_set <- data_set %>%
  left_join(qualifications)
names(data_set)
```

3.2.7 Marital status feature

Using the same methodology as before, rations of the 25 to 64 residents, and normalising for each geo area.

The total numbers in this section were significantly less than the other categories, maybe two thirds of the total, so this may not be a good measure, having some uncertainty around it.

```
# next marital status
# https://www.nomisweb.co.uk/census/2011/lc1101ew
# import the new table with marital status
nomis_census_csv("lc1101ew", geographytype)
data <- ingest("lc1101ew", geographytype)
#creating a function for repeatability
marital_predictors <- function(data){
  marital_raw <- data %>%
    #only all genders
    select(contains("Sex: All persons;") | contains("geography code")) %>%
    #save only the 25 to 64 years
    select(contains("Age 25 to 34") | contains("Age 35 to 49") | contains("Age 50 to 64") | contains("g
    rename("geo_code" = "geography code") %>%
    #calculate for retained predictors
    rowwise() %>%
    mutate("all_marital_25_64"= sum(across(contains("All categories: ")))) %>%
    mutate("single_25_64"= sum(across(contains("Marital Status: Single")))) %>%
    mutate("married_25_64"= sum(across(contains("Marital Status: Married;")))) %>%
    mutate("civil_25_64"= sum(across(contains("Marital Status: In a registered same-sex civil")))) %>%
    mutate("separated_25_64"= sum(across(contains("Marital Status: Separated")))) %>%
    mutate("divorced_25_64"= sum(across(contains("Marital Status: Divorced")))) %>%
    mutate("widowed_25_64"= sum(across(contains("Marital Status: Widowed")))) %>%
    select(contains("geo_code") | contains("25_64"))
  # normalise per area
  marital <- sweep(marital_raw[,3:8], 1, rowSums(marital_raw[,3:8]), FUN = "/")  %>%
    cbind(marital_raw[,1])
  save(marital, file='rda/marital.rda')
}
#running the function on the data set
marital_predictors(data)
#adding to main data set
load('rda/marital.rda')
data_set <- data_set %>%
  left_join(marital)
```

3.2.8 Religion feature

As for the marital features. There are a number of religions with a small number of people, and it may make sense to aggregate them in to "other". Hoever, with the same cutoff of 0.5% as earlier, only one religion is summarised, "Jewish", therefore it can be left as it is.

```
# next religion
# https://www.nomisweb.co.uk/census/2011/lc2107ew
# import the new table with religion
nomis_census_csv("lc2107ew", geographytype)
data <- ingest("lc2107ew", geographytype)
#creating a function for repeatability
religion_predictors <- function(data){
  religion_raw <- data %>%
```

```
    # all genders
    select(contains("Sex: All persons;") | contains("geography code")) %>%
    #select the working age only
    select(contains("Age 25 to 34") | contains("Age 35 to 49") | contains("Age 50 to 64") | contains("g
    rename("geo_code" = "geography code") %>%
    #calculate the sums across ages
    rowwise() %>%
    mutate("all_religion_25_64"= sum(across(contains("All categories:")))) %>%
    mutate("christian_25_64"= sum(across(contains("Christian")))) %>%
    mutate("buddhist_25_64"= sum(across(contains("Buddhist")))) %>%
    mutate("hindu_25_64"= sum(across(contains("Hindu")))) %>%
    mutate("jewish_25_64"= sum(across(contains("Jewish")))) %>%
    mutate("muslim_25_64"= sum(across(contains("Muslim")))) %>%
    mutate("sikh_25_64"= sum(across(contains("Sikh")))) %>%
    mutate("other_25_64"= sum(across(contains("Other religion;")))) %>%
    mutate("no_religion_25_64"= sum(across(contains("No religion;")))) %>%
    mutate("religion_not_stated_25_64"= sum(across(contains("Religion not stated;")))) %>%
    select(contains("geo_code") | contains("25_64"))
  # normalise per area
  religion <- sweep(religion_raw[,3:11], 1, rowSums(religion_raw[,3:11]), FUN = "/")  %>%
    cbind(religion_raw[,1])
  save(religion, file='rda/religion.rda')
}
#running the function on the data set
religion_predictors(data)
#adding to main data set
load('rda/religion.rda')
data_set <- data_set %>%
  left_join(religion)
```

3.2.9 Household composition feature

In this case the 25 to 64 age group is not readily accessible, as the ages included in the predictors are:

- Age 25 to 34
- Age 35 to 49
- Age 50 and over

There is a subset which is called "One person household: Aged 65 and over" and "One family only: All aged 65 and over", which can be subtracted from the "Aged 50 and over" section. Not perfect as households will have a mixture of ages but an improvement to make the predictor closer to be relevant to the predictions and similar to the other features.

```
#next household composition
# https://www.nomisweb.co.uk/census/2011/lc1109ew
# import the new table with household
nomis_census_csv("lc1109ew", geographytype)
data <- ingest("lc1109ew", geographytype)
names(data)
# rerunning as a function for reuse
household_predictors <- function(data){
  household_raw <- data %>%
    # retain only the all sex (not female or male)
    select(contains("Sex: All persons;") | contains("geography code")) %>%
    # retain only the specific age groups
```

```
    select(contains("Age 25 to 34") |
             contains("Age 35 to 49") |
             contains("Age 50 and over") |
             contains("geography code")) %>%
    rename("geo_code" = "geography code") %>%
    # retain the totals, the total amount, and the over 65 columns
    select(contains("geo_code") |
             contains("All categories: Household composition") |
             contains(": Total") |
             contains("ged 65 and over")) %>%
    rowwise() %>%
    # combine the age categories to create the predictors
    mutate("all_household_25_64"= sum(across(contains("All categories:"))) -
             sum(across(contains("One person household: Aged 65 and over"))) -
             sum(across(contains("One family only: All aged 65 and over")))) %>%
    mutate("oneperson_25_64"= sum(across(contains("One person household: Total"))) -
             sum(across(contains("One person household: Aged 65 and over")))) %>%
    mutate("marriedfamily_25_64"= sum(across(contains("Married")))) %>%
    mutate("cohoabiting_25_64"= sum(across(contains("Cohabiting")))) %>%
    mutate("loneparent_25_64"= sum(across(contains("Lone parent")))) %>%
    mutate("otherhousehold_25_64"= sum(across(contains("Other household types: Total")))) %>%
    select(contains("geo_code") | contains("25_64")) %>%
    mutate(checksum_all = sum(across(contains("25_64"))) - all_household_25_64, .after = geo_code)
  # normalise per area
  household <- sweep(household_raw[,4:8], 1, rowSums(household_raw[,4:8]), FUN = "/")  %>%
    cbind(household_raw[,1])
  save(household, file='rda/household.rda')
}
#running the function on the data set
household_predictors(data)
#adding to main data set
load('rda/household.rda')
data_set <- data_set %>%
  left_join(household)
```

3.2.10 Industry feature

This used the same methodology as before, with the categories retained from the census. A summary list of the industries is as follows:

- A, B, D, E Agriculture, energy and water
- C Manufacturing
- F Construction
- G,I Distribution, hotels and restaurants
- H,J Transport and communication
- K,L,M,N Financial, Real Estate, Professional and Administrative activities
- O,P,Q Public administration, education and health
- R,S,T,U Other

```
#next Industry
# https://www.nomisweb.co.uk/census/2011/lc6110ew
# import the new table with Industry
nomis_census_csv("lc6110ew", geographytype)
data <- ingest("lc6110ew", geographytype)
```

```
names(data)
# rerunning as a function for reuse
industry_predictors <- function(data){
  industry_raw <- data %>%
    # retain only the specific age groups
    select(contains("Age 25 to 34") | contains("Age 35 to 49") | contains("Age 50 to 64") | contains("g
    rename("geo_code" = "geography code") %>%
    # combine the age categories to create the predictors
    rowwise() %>%
    mutate("all_industry_25_64"= sum(across(contains("All categories:")))) %>%
    mutate("industry_abde_25_64"= sum(across(contains("Industry: A, B, D, E ")))) %>%
    mutate("manufacturing_25_64"= sum(across(contains("Manufacturing")))) %>%
    mutate("construction_25_64"= sum(across(contains("Construction")))) %>%
    mutate("industry_gi_25_64"= sum(across(contains("Industry: G, I")))) %>%
    mutate("industry_hj_25_64"= sum(across(contains("Industry: H, J ")))) %>%
    mutate("industry_klmn_25_64"= sum(across(contains("Industry: K, L, M, N")))) %>%
    mutate("industry_opq_25_64"= sum(across(contains("Industry: O, P, Q ")))) %>%
    mutate("Industry_rstu_25_64"= sum(across(contains("Industry: R, S, T, U ")))) %>%
    select(contains("geo_code") | contains("25_64")) %>%
    mutate(checksum_all = sum(across(contains("25_64"))) - all_industry_25_64, .after = geo_code)
  # normalise per area
  industry <- sweep(industry_raw[,4:11], 1, rowSums(industry_raw[,4:11]), FUN = "/")  %>%
    cbind(industry_raw[,1])
  save(industry, file='rda/industry.rda')
}

#running the function on the data set
industry_predictors(data)
#adding to main data set
load('rda/industry.rda')
data_set <- data_set %>%
  left_join(industry)
```

3.2.11 Country of birth feature

The methodology is as before, but the data set is complex with many overlapping categories, for example:

- Other Europe: Total
- Other Europe: EU countries: Total
- Other Europe: EU countries: Member countries in March 2001
- Other Europe: EU countries: Accession countries April 2001 to March 2011
- Other Europe: Rest of Europe

The following were the categories used for features, again selecting by sizes of populations, and using the aggregation from the census data. One difference was that the Ireland category was combined with the rest of the EU, as there were not so many people:

- Europe: United Kingdom: Total
- Europe: Other Europe: EU countries: Member countries in March 2001
- Europe: Other Europe: EU countries: Accession countries April 2001 to March 2011
- Europe: Other Europe: Rest of Europe
- Africa
- Middle East and Asia
- The Americas and the Caribbean
- Antarctica, Oceania (including Australasia) and other

```
# next country of birth
# https://www.nomisweb.co.uk/census/2011/lc2103ew
# import the new table with country of birth
nomis_census_csv("lc2103ew", geographytype)
data <- ingest("lc2103ew", geographytype)
names(data)
# running as a function for reuse
country_predictors <- function(data){
  country_raw <- data %>%
    # retain only the all sex (not female or male)
    select(contains("Sex: All persons;") | contains("geography code")) %>%
    select(contains("Age 25 to 34") | contains("Age 35 to 49") | contains("Age 50 to 64") | contains("g
    rename("geo_code" = "geography code") %>%
    # combine the age categories to create the predictors
    rowwise() %>%
    mutate("all_country_25_64"= sum(across(contains("All categories:")))) %>%
    mutate("uk_25_64"= sum(across(contains("United Kingdom: Total")))) %>%
    mutate("eu_2001_25_64"= sum(across(contains("Other Europe: EU countries: Member countries in March 
    mutate("eu_rest_25_64"= sum(across(contains("Other Europe: EU countries: Accession countries April 
    mutate("europe_rest_25_64"= sum(across(contains("Other Europe: Rest of Europe")))) %>%
    mutate("africa_25_64"= sum(across(contains("Africa")))) %>%
    mutate("me_asia_25_64"= sum(across(contains("Middle East and Asia")))) %>%
    mutate("americas_25_64"= sum(across(contains("The Americas and the Caribbean")))) %>%
    mutate("other_country_25_64"= sum(across(contains("Antarctica, Oceania (including Australasia) and 
    select(contains("geo_code") | contains("25_64")) %>%
    mutate(checksum_all = sum(across(contains("25_64"))) - all_country_25_64, .after = geo_code)
  # normalise per area
  country <- sweep(country_raw[,4:11], 1, rowSums(country_raw[,4:11]), FUN = "/")  %>%
    cbind(country_raw[,1])
  save(country, file='rda/country.rda')
}
#running the function on the data set
country_predictors(data)
#adding to main data set
load('rda/country.rda')
data_set <- data_set %>%
  left_join(country)
```

3.2.12 Creating repeatable functions

The code is intended to support different area sizes, and therefore repeatable functions were created to create each set of features from the tables, as shown in the earlier section.

The results were confirmed by running the functions and comparing against the data set when run as separate commands, using the regional data for ease of handling. The full code includes both variants of the feature creation.

Then code was developed to carry out the full creation of the data set. Firstly, all necessary tables for a given geographic area are downloaded from Nomis.

```
####### repeatable code ##########
# geographytype <- "TYPE480" #regions
# geographytype <- "TYPE297" #super output areas - middle layer
# nomis_census_csv(censusdata, geographytype)
```

```
#creating list of census tables to take
censusdata_list <- c("ks608ew", "ks102ew", "lc2101ew", "lc2103ew","lc6110ew", "lc1109ew", "lc2107ew", "l

#create a function to download all the tables in the list
download_censusdata <- function(geotype, datalist){
  # apply the list to download all of the data
  tmp <- lapply(datalist, function(censusdata){
    # pause for a few random seconds to avoid annoying nomis
    time <- sample(4:17,1)
    print(paste("pausing", time, "seconds"))
    Sys.sleep(time)
    print(paste("downloading geography area", geotype, "for table", censusdata))
    #run the function to generate and download the file
    nomis_census_csv(censusdata, geotype)
  })
  rm(tmp)
}
```

The following code runs each of the feature functions in turn, appending to the data set until all the features are included.

```
# function to load in the data for a given geotype
import_data <- function(geotype){
  print(paste("importing", geotype))
  #load in occupation data
  print(paste("loading occupation data"))
  # https://www.nomisweb.co.uk/census/2011/ks608ew
  # import the new table with occupation data
  data <- ingest("ks608ew", geotype)
  #running the function on the data set
  occupation_target(data)
  #adding to main data set
  load('rda/occupation.rda')
  data_set <- occupation
  rm(occupation)

  # load in age data
  print(paste("loading age data"))
  # https://www.nomisweb.co.uk/census/2011/ks102ew
  # import the new table with age data
  data <- ingest("ks102ew", geotype)
  #running the function on the data set
  age_predictors(data)
  #adding to main data set
  load('rda/age.rda')
  data_set <- data_set %>%
    left_join(age)
  rm(age)

  # next to load in the ethnicity data
  print(paste("loading ethnicity data"))
  # https://www.nomisweb.co.uk/census/2011/lc2101ew
  # updating with age and ethnicity
  data <- ingest("lc2101ew", geotype)
```

```
#running the function on the data set
ethnicity_predictors(data)
#adding to main data set
load('rda/ethnicity.rda')
data_set <- data_set %>%
  left_join(ethnicity)
rm(ethnicity)

# next to load in the sex/gender data, for ages 25 to 64
print(paste("loading sex data"))
# https://www.nomisweb.co.uk/census/2011/lc2101ew
# using the ethnicity data, which includes gender
data <- ingest("lc2101ew", geotype)
#running the function on the data set
sex_predictors(data)
#adding to main data set
load('rda/sex_25_64.rda')
data_set <- data_set %>%
  left_join(sex_25_64)
rm(sex_25_64)

# qualifications
print(paste("loading qualifications data"))
# https://www.nomisweb.co.uk/census/2011/lc5102ew
# import the new table with qualifications
data <- ingest("lc5102ew", geotype)
#running the function on the data set
qualifications_predictors(data)
#adding to main data set
load('rda/qualifications.rda')
data_set <- data_set %>%
  left_join(qualifications)
rm(qualifications)

# next marital status
print(paste("loading marital data"))
# https://www.nomisweb.co.uk/census/2011/lc1101ew
# import the new table with marital
data <- ingest("lc1101ew", geotype)
#running the function on the data set
marital_predictors(data)
#adding to main data set
load('rda/marital.rda')
data_set <- data_set %>%
  left_join(marital)
rm(marital)

# next religion
print(paste("loading religion data"))
# https://www.nomisweb.co.uk/census/2011/lc2107ew
# import the new table with religion
data <- ingest("lc2107ew", geotype)
#running the function on the data set
```

```
religion_predictors(data)
#adding to main data set
load('rda/religion.rda')
data_set <- data_set %>%
  left_join(religion)
rm(religion)

#next household composition
print(paste("loading household data"))
# https://www.nomisweb.co.uk/census/2011/lc1109ew
# import the new table with household
data <- ingest("lc1109ew", geotype)
#running the function on the data set
household_predictors(data)
#adding to main data set
load('rda/household.rda')
data_set <- data_set %>%
  left_join(household)
rm(household)

#next Industry
print(paste("loading industry data"))
# https://www.nomisweb.co.uk/census/2011/lc6110ew
# import the new table with Industry
data <- ingest("lc6110ew", geotype)
#running the function on the data set
industry_predictors(data)
#adding to main data set
load('rda/industry.rda')
data_set <- data_set %>%
  left_join(industry)
rm(industry)
# names(data_set)

# next country of birth
print(paste("loading country data"))
# https://www.nomisweb.co.uk/census/2011/lc2103ew
# import the new table with country of birth
data <- ingest("lc2103ew", geotype)
#running the function on the country of birth data set
country_predictors(data)
#adding to main data set
load('rda/country.rda')
data_set <- data_set %>%
  left_join(country)
rm(country, data)

#save the data set
filename <- paste("data_set", geotype, sep = "_") %>%
  paste0(., ".rda")
destfile <- paste0("rda/", filename)
print(paste("saving", destfile))
save(data_set, file=destfile)
```

```
  #clean up
  rm(filename, destfile)
  return(data_set)
}
```

With the creation of these functions, the creation of the full data set for the middle layer super output areas (MSOA) is three lines, as below.

This produced a data set of 7201 observations with 62 variables, of which two are the name and the code for the area, and one is the y, the ratio of people in managerial and professional occupations. This file when saves as an R object is 3MB.

```
#creating list of census tables to take
censusdata_list <- c("ks608ew", "ks102ew", "lc2101ew", "lc2103ew","lc6110ew",
                     "lc1109ew", "lc2107ew", "lc1101ew", "lc5102ew")
# run the function to download the csv files
download_censusdata("TYPE297", censusdata_list)
# run the function on the MSOA imported data
data_set <- import_data("TYPE297")
```

3.3 Create main and validation MSOA data set

A validation set of 10% was created and saved for later, with the next phase of modelling being carried out on the retained "main" set.

```
# load the MSOA raw data from file
load("rda/data_set_TYPE297.rda")
# Validation set will be 10% of the data
set.seed(2011, sample.kind="Rounding")
test_index <- createDataPartition(y = data_set$y, times = 1, p = 0.1, list = FALSE)
main <- data_set[-test_index,]
validation <- data_set[test_index,]
```

3.4 Data exploration and visualisation of the MSOA data

The Middle layer Super Output Areas (MSOA), for England and Wales is a data set of 7201 observations with 62 variables, of which two are the name and the code for the area, the key, one is the outcome y, so 59 potential features/predictors.

Two of the variables relate to the size of the area. This is potentially interesting for comparison between the data sets, but not for predicting the outcome, as it is a function of the data capture. The ratio of people in the "occupation" category may have some predictability, but not the total size.

Similarly the "geo_code" is a key to link the data together. In addition, it does relate to the wider region, and it is likely that some regions of the UK have more senior roles present than others. There is a discussion at policy and political levels of left behind areas, and this may be a useful predictor. Therefore the data can be augmented with the geographic areas, replacing the "geo_name"

The "geo_type" is supposed to be separated in to Urban and Rural geographical areas, but in this data set all of the samples are "Total" which maybe indicates that they are too large and so contain both, or not recorded for some statistical gathering reason. It is not available for the Output Areas either and can be deleted.

```
table(main$geo_type)
```

```
##
## Total
##  6480
```

Following is the graph of the distribution of the outcome y, the proportion of residents in the area with Managerial or Professional occupation. This is not a normal distribution with a clear skew towards zero.

```
#make the occupation in an area a ratio
main <- main %>%
  mutate(occ_ratio = occupation_all/all_residents, .after = all_residents)
#graph the outcome y
main %>%  ggplot(aes(y))  +
  geom_histogram(bins = 30, color = "black") +
  ggtitle("Distribution of the outcome, y") +
  xlab("y, proportions of people in managerial or professional occupations") +
  ylab("count of numbers of MSOA areas")
```

The mean, maximum and minimum show a decent spread.

```
#table of key data points
main %>% group_by(geo_type) %>%
  summarise(mean = mean(y),
            sd = sd(y),
            max = max(y),
            min = min(y)) %>%
  select(-geo_type)  %>%
  knitr::kable(caption="Summary of the distributuion of the outcome y, for MSOA")
```

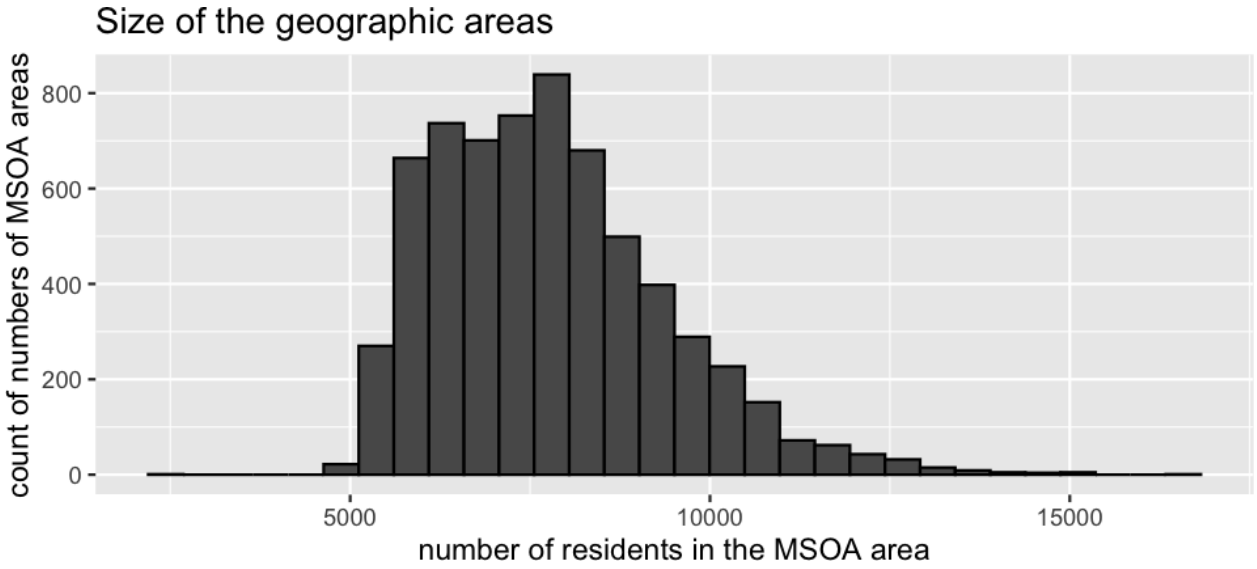
Table 6: Summary of the distributuion of the outcome y, for MSOA

mean	sd	max	min
0.275694	0.095921	0.661216	0.063305

Looking at the geographic areas, the distribution is strangely similar to the outcome, which is maybe something to watch. However, this may be that it they will both be right skewed or positively skewed as they are limited on the left had side to a minimum, either zero of the minimum MSOA size.

```
#graph the geographic areas
main %>%  ggplot(aes(all_residents))  +
  geom_histogram(bins = 30, color = "black") +
```

```
  ggtitle("Size of the geographic areas") +
  xlab("number of residents in the MSOA area") +
  ylab("count of numbers of MSOA areas")
```



This is also reflected in the table of information for the sizes.

```
#table of key data points of the sizes of the areas
main %>% group_by(geo_type) %>%
  summarise(mean = mean(all_residents),
            sd = sd(all_residents),
            max = max(all_residents),
            min = min(all_residents)) %>%
  select(-geo_type)  %>%

  knitr::kable(caption="Summary of the distributuion of the area sizes, for MSOA")
```

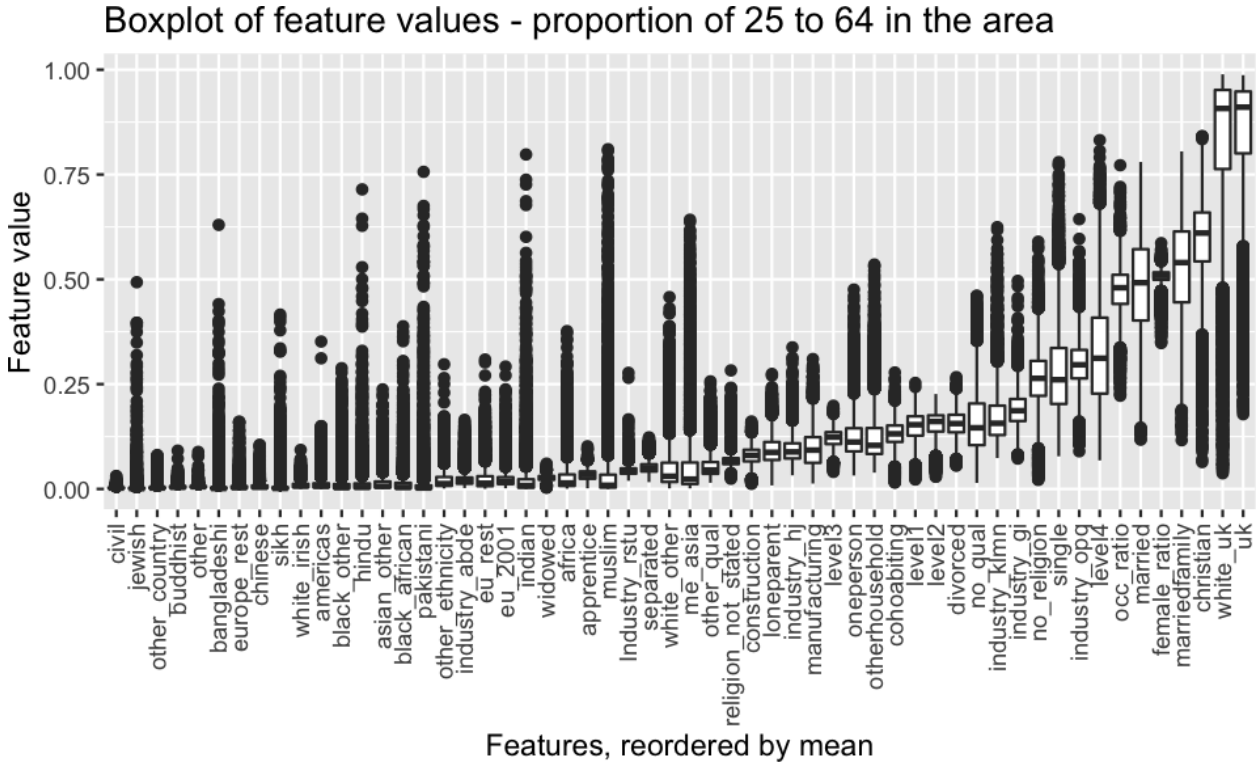
Table 7: Summary of the distributuion of the area sizes, for MSOA

mean	sd	max	min
7784.18	1605.54	16342	2203

Now looking at the features, aside from the median age, the others are proportions of the people aged 25 to 64 in an area. Making a boxplot of those proportions, if can be seen that a few are very high and a few very low. Of the ones with high proportions - UK nationals and White UK ethnicity, not surprising, but there are areas with much lower values to below 25%. On the other hand, there are a number with low values for all areas, which may indicate that they are not useful features.

```
#looking at the features
#box plot
#create a "tidy" long form version for the ggplot
main_tidy <- main %>%
  select(-occupation_all, -all_residents, -geo_type, -geo_name, -age_median) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  #pivot to a long version with a row per feature/value
```

```
  pivot_longer(cols = c(-geo_code, -y),
               names_to = "feature",
               values_to = "proportion")
#plot the box plot
main_tidy %>%
  ggplot(aes(x=reorder(feature, proportion, FUN=mean), y=proportion))  +
  geom_boxplot() +
  ggtitle("Boxplot of feature values - proportion of 25 to 64 in the area") +
  ylab("Feature value") +
  xlab("Features, reordered by mean") +
  theme(axis.text.x=element_text(angle = 90, hjust=1, vjust=0.5))
```



Investigating further, looking at a table of values, there are a number with low variability, which are probably candidates for removing when there are more computationally challenging models.

```
#choosing the 10 features with the lowest mean
main_tidy %>% group_by(feature) %>%
  summarise(mean=mean(proportion), sd=sd(proportion), max=max(proportion), min=min(proportion)) %>%
  arrange(mean) %>% head(10) %>%
  knitr::kable(caption="Summary of features with low mean proportions")
```

Table 8: Summary of features with low mean proportions

feature	mean	sd	max	min
civil	0.002649	0.002181	0.030985	0
jewish	0.004145	0.020514	0.493141	0
other_country	0.004386	0.007633	0.081054	0
buddhist	0.005476	0.004869	0.091413	0
other	0.005779	0.003788	0.088435	0
bangladeshi	0.006459	0.025925	0.629917	0

feature	mean	sd	max	min
europe_rest	0.006656	0.011704	0.160691	0
chinese	0.006858	0.008560	0.105263	0
sikh	0.007519	0.023988	0.415898	0
white_irish	0.009864	0.008715	0.093198	0

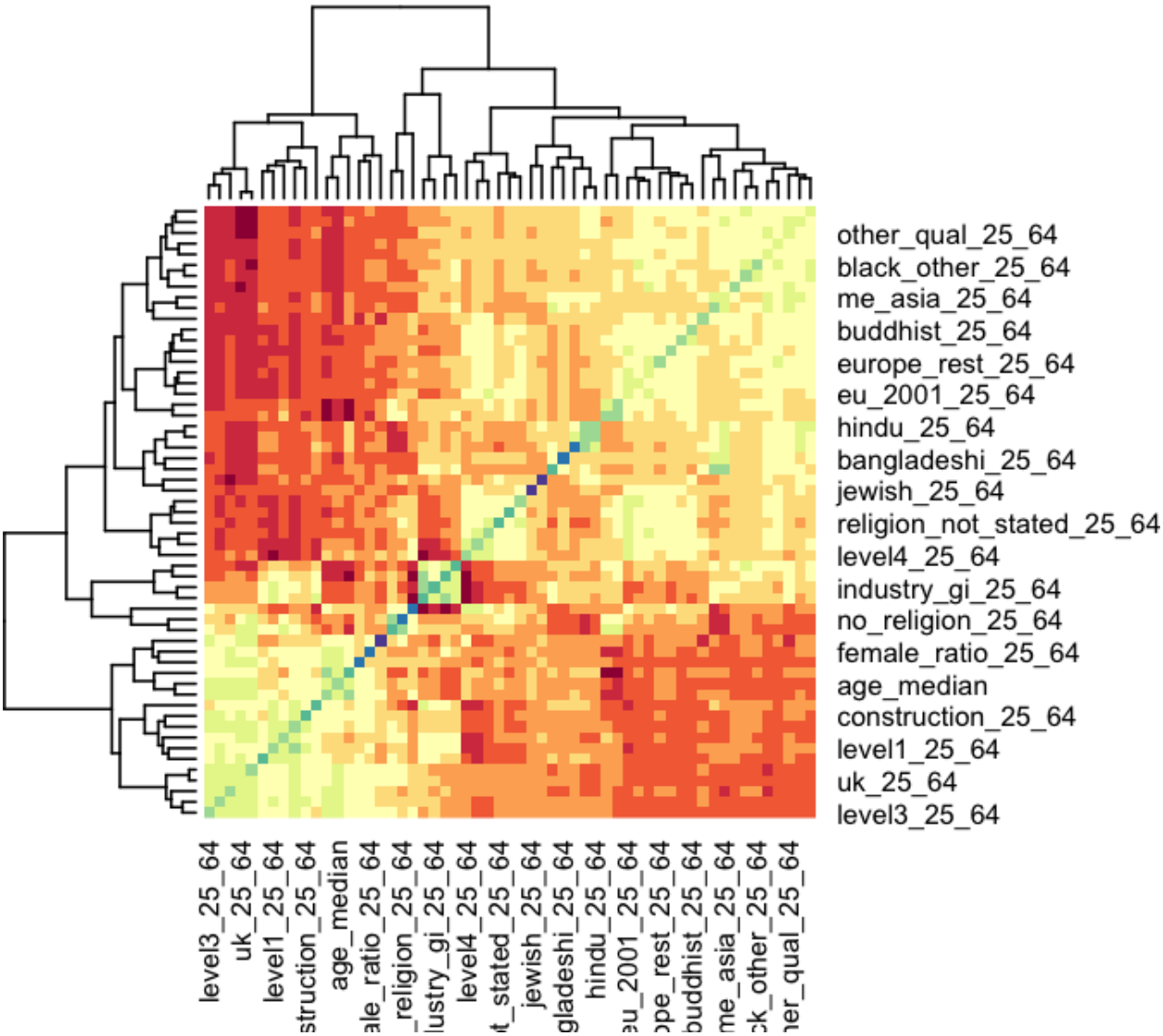
The actual selection of candidates for removing should be carried out with the training data, in the next section.

3.4.1 Correlation between features

There are a lot of features, there is almost certainly some duplication, for example, the "other" data is the remaining proportion left over, so will be fully explained by existing features.

Using the caret packages, it can be seen that a number of the features are highly correlated with each other.

```
### calculating the correlation between features
correlationmatrix <- cor(main[,7:63])
# image of correlation matrix
heatmap(x = correlationmatrix, col = RColorBrewer::brewer.pal(11, "Spectral"))
```



Following are the list of highly correlated features, which can be seen in a version of the image with the features reorded, with a number very similar to each other at the left and bottom rows.

```
# find attributes that are highly corrected
highlycorrelated <-
  findCorrelation(correlationmatrix, cutoff=0.8, exact = TRUE, names=TRUE)
highlycorrelated
```

```
##  [1] "uk_25_64"             "white_uk_25_64"       "otherhousehold_25_64"
##  [4] "level2_25_64"         "white_other_25_64"    "africa_25_64"
##  [7] "eu_2001_25_64"        "marriedfamily_25_64"  "me_asia_25_64"
## [10] "single_25_64"         "level1_25_64"         "muslim_25_64"
## [13] "level4_25_64"         "married_25_64"        "hindu_25_64"
```

Investigating further, one can see that “white_uk” (ethnicity) and “uk” (nationality) are 98% correlated, so only one of those need be retained. Some features are highly negatively correlated, as below, and as initially surmised, the “other” categories are within these, as candidates to remove.

```
#looking at the correlation for uk_25_64 specifically
data.frame(correlationmatrix) %>%
```

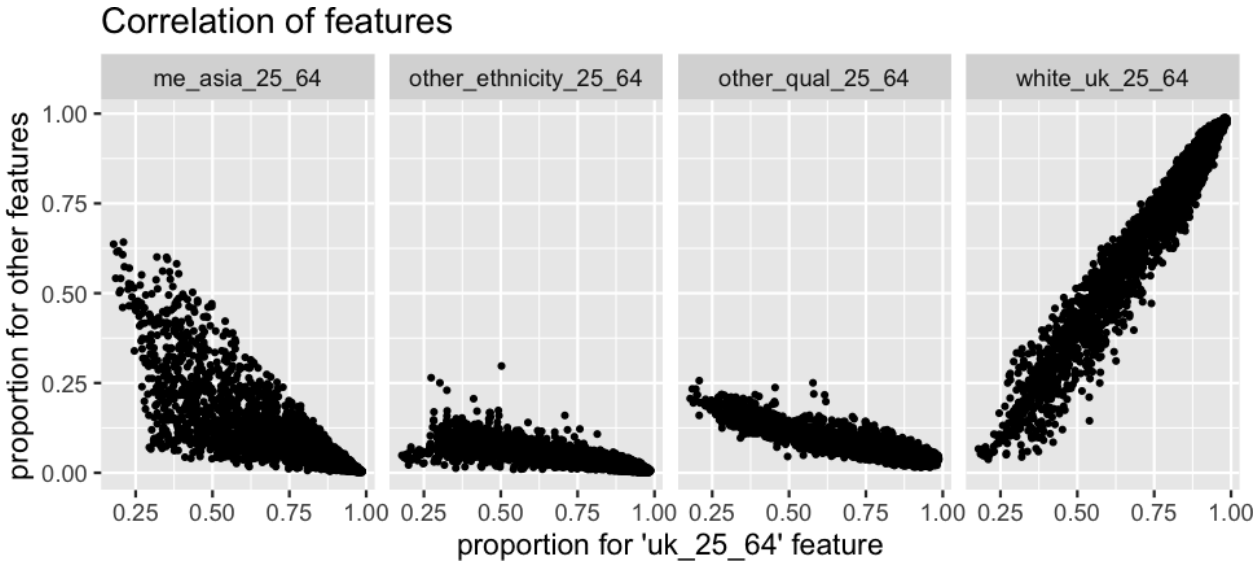
```
  select(uk_25_64) %>%
  filter(abs(uk_25_64) > 0.8) %>%
  knitr::kable(caption="Highly correlated features with 'uk_25_64'")
```

Table 9: Highly correlated features with 'uk_25_64'

	uk_25_64
white_uk_25_64	0.984635
other_ethnicity_25_64	-0.850962
asian_other_25_64	-0.814122
other_qual_25_64	-0.900515
otherhousehold_25_64	-0.900496
uk_25_64	1.000000
africa_25_64	-0.838491
me_asia_25_64	-0.840396

This is visible in scatter plots of the features against each other.

```
#graph of features against uk_25_64
main %>%  pivot_longer(
  cols = (!"uk_25_64" & contains("25_64") | contains("ratio") ),
  names_to = "feature",
  values_to = "proportion")  %>%
  select(-occupation_all, -all_residents, -geo_type, -geo_name) %>%
  filter(feature %in%
           c("white_uk_25_64", "me_asia_25_64",
             "other_ethnicity_25_64", "other_qual_25_64")) %>%
  ggplot(aes(uk_25_64, proportion))  +
  geom_point(size=0.7) +
  facet_grid(. ~feature) +
  ggtitle("Correlation of features") +
  xlab("proportion for 'uk_25_64' feature") +
  ylab("proportion for other features")
```



As earlier, any decisions on features to remove or not should be made with the training set.

3.4.2 Geographic distribution

There may be some geographic influence, maybe the roles being in a certain place, and therefore a feature would be useful. This section describes how the information on the locations and used to create a number of features relating to the geography.

The super output area (SOA) maps to the Lower layer Super Output Area (LSOA), on to the Middle Super Output Area (LSOA), and finally on to the Local Authority Districts (LAD). The detail of the mapping is described here:

- https://census.ukdataservice.ac.uk/use-data/guides/boundary-data.aspx
- https://www.ons.gov.uk/methodology/geography/ukgeographies/censusgeography#output-area-oa
- Geo mapping http://geoconvert.ukdataservice.ac.uk/
- https://borders.ukdataservice.ac.uk/easy_download.html

There is a lookup file provided which can be used to create the location feature. https://borders.ukdataservice.ac.uk/lut_download_data.html?data=oa11_lsoa11_msoa11_lad11_ew_lu.

Following is the code to carry out the mapping, which downloads the file, and then creates a lookup file between the area codes and the local areas, including over 7000 MSOAs and 130000 SOAs mapped to 348 local areas, with the area codes changed to be numerical rather than characters.

```
# getting the mapping for the locations
# https://borders.ukdataservice.ac.uk/lut_download_data.html?data=oa11_lsoa11_msoa11_lad11_ew_lu
#download file
geo_url <-
  "https://borders.ukdataservice.ac.uk/ukborders/lut_download/prebuilt/luts/engwal/OA11_LSOA11_MSOA11_LA
download.file(geo_url, 'data/OA11_LSOA11_MSOA11_LAD11_EW_LU.zip')
#unzip to data directory
unzip('data/OA11_LSOA11_MSOA11_LAD11_EW_LU.zip', exdir = "data/")
geo_mapping <-
  read.csv('data/OA11_LSOA11_MSOA11_LAD11_EW_LUv2.csv', fileEncoding="latin1")
save(geo_mapping, file='rda/geo_mapping.rda')
head(main$geo_code, 20)
head(geo_mapping)
# the columns of interest are MSOA11CD (MSOA), OA11CD (SOA) and LAD11CD
load("rda/geo_mapping.rda")
geo_lookup <- geo_mapping %>%
  #for the MSOA and SOA
  select(LAD11CD, OA11CD, MSOA11CD, ) %>%
  rename(local_area = LAD11CD, geo_code_soa=OA11CD, geo_code_msoa = MSOA11CD) %>%
  unique()
head(geo_lookup)
# however, for many models to work, the feature needs to be numeric
# I need to convert the character strings to numbers, so remove the "E"
# calculating a list of strings within the local area to be rewritten as numbers
# keeping the strings replaced as long as possible for smaller area_code
# retaining a 3 digit long area code
area_update <- levels(as.factor(str_sub(geo_lookup$local_area, 1, -3)))
#create a new column to be updated
geo_lookup <- geo_lookup %>%
  mutate(area_code = local_area)
head(geo_lookup)
#create a for loop to go through this list and update the local areas
for(i in 1:length(area_update)){
  #the old string to replace
  oldprefix <- area_update[i]
```

```
  # the new string, a number, but as a character to work with the existing character
  newprefix <- as.character(i)
  geo_lookup$area_code <- str_replace_all(geo_lookup$area_code, oldprefix, newprefix)
}
head(geo_lookup)
# change the character string to be numerical
geo_lookup$area_code <- as.integer(geo_lookup$area_code)
#checking
head(geo_mapping$local_area)
length(levels(as.factor(geo_mapping$local_area)))
head(levels(as.factor(geo_mapping$local_area)))
```

This will only be useful if the algorithms treat the number representing the geographic area as a category variable - the numbers are unrelated to underlying distributions and will not contain information if it is treated it as a continuous variable. A better way to do this would be to create a feature per region, and then give a 1 or 0 as to whether the MSOA/SOA is in that region.

This would be unwieldy with hundreds of areas as columns, and so the regions need to be larger. There is a further lookup from the local area (LAD) to a smaller number of regions in England (Wales is effectively a single region), as described here. https://geoportal.statistics.gov.uk/datasets/local-authority-district-to-region-december-2018-lookup-in-england

This lookup is then used to can use this to map each SOA/MSOA/LAD a specific region, which is in turn used to create columns. The region names are used, added to the geo_lookup created earlier, via an interim "region_lookup" file. Note that the MSOA in Wales had blank region names, and much of the complexity below is to add the wales name in, and to deal with a blank line in data.

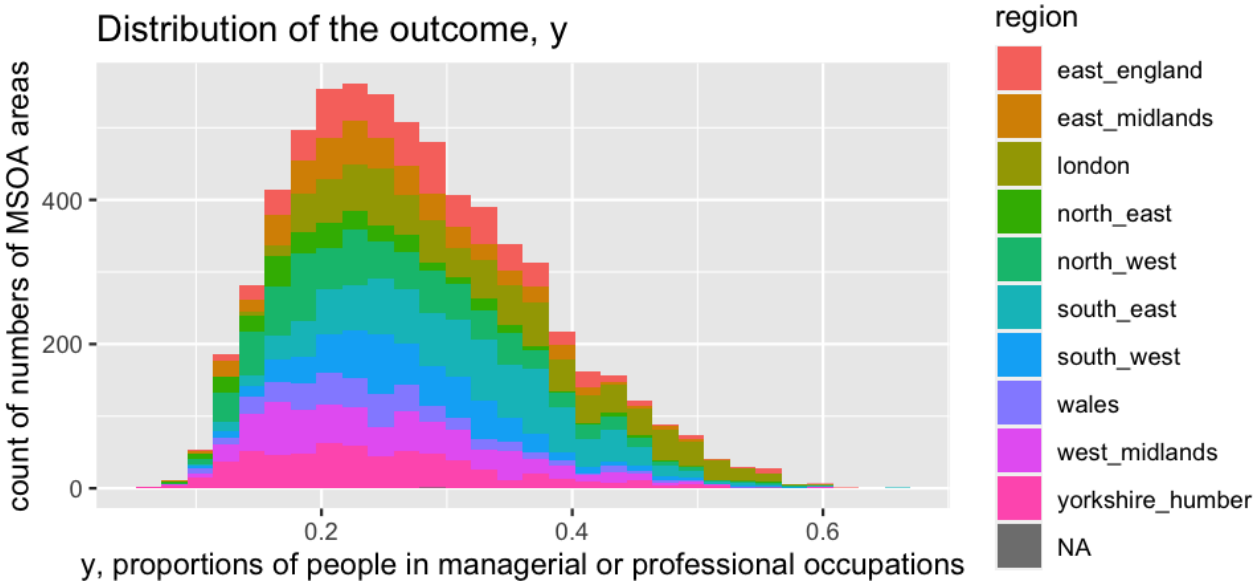
```
# in addition creating a column per region
# https://opendata.arcgis.com/datasets/0aac9db88cbb4f8d9db42ad20b03cb71_0.csv
#download file
geo_url <- "https://opendata.arcgis.com/datasets/0aac9db88cbb4f8d9db42ad20b03cb71_0.csv"
download.file(geo_url, 'data/0aac9db88cbb4f8d9db42ad20b03cb71_0.csv')
# load in file
region_mapping<- read.csv('data/0aac9db88cbb4f8d9db42ad20b03cb71_0.csv')
save(region_mapping, file='rda/region_mapping.rda')
save(region_mapping, file='rda/region_mapping.rda')
#checking the data
# load('rda/region_mapping.rda')
head(region_mapping)
levels(as.factor(region_mapping$RGN11NM))
# some have empty region name
# visual inspection shows that the unnamed regions are all Wales
# and the local area ids start with "W"
# there is a blank, for row 1, to be investigated
region_mapping %>%
  select(LAD11NM, LAD11CD, RGN11NM) %>%
  filter(RGN11NM == "") %>% unique()
# the blank row is the Isles of Scilly, which may or may not be in the data set
region_mapping %>%
  filter(LAD11CD == "")
#save just the local area and region
region_lookup <- region_mapping %>%
  rename(local_area = LAD11CD, region = RGN11NM) %>%
  select(local_area, region) %>%
  unique()
```

```r
# check the regions, showing the blank
levels(as.factor(region_lookup$region))
head(region_lookup)
# update empty fields to be "wales", where the local area starts with "WO"
# create an index of areas starting with W
index <- str_which(region_lookup$local_area, "WO")
# confirm it works
region_lookup[index,]
# add wales to the region column for those areas
region_lookup$region[index] <- "wales"
# check the regions, now including wales
levels(as.factor(region_lookup$region))
#removing the blank line (Isles of Scilly)
region_lookup <- region_lookup %>%
  filter(region != "")
#confirm that the regions are now correct
levels(as.factor(region_lookup$region))
# updating the names
region_lookup$region <- str_to_lower(region_lookup$region)
region_lookup$region <- str_replace_all(region_lookup$region, " and the ", "_")
region_lookup$region <- str_replace_all(region_lookup$region, " of ", "_")
region_lookup$region <- str_replace_all(region_lookup$region, " ", "_")
levels(as.factor(region_lookup$region))
#create a list of regions
region_list <- unique(region_lookup$region)
# create a table, with a column per region
# looping through, for each name in the region list
# if it matches the region assign a 1
region_table <- sapply(region_list, function(region){
  ifelse(str_detect(region_lookup$region, region), 1, 0)
})
# checking it works as intended
head(region_table, 20)
head(region_lookup, 20)
tail(region_table, 20)
tail(region_lookup, 20)
# adding the columns to the region lookup
region_lookup <- region_lookup %>% cbind(as.data.frame(region_table))
# save completed lookup for later
save(region_lookup, file='rda/region_lookup.rda')
# checking
head(region_lookup, 50)
tail(region_lookup, 20)
head(geo_lookup)
sum(is.na(region_lookup))
#add to the geo_lookup file
geo_lookup <- geo_lookup %>%
  left_join(region_lookup) %>%
  select(-region)
#this has created some NAs, so need to fix
geo_lookup[is.na(geo_lookup)] <- 0
sum(is.na(geo_lookup))
head(geo_lookup)
```

```
#number of local areas
length(levels(as.factor(geo_lookup$area_code)))
#tidy up
save(geo_lookup, file='rda/geo_lookup.rda')
```

Finally a graph to see if there is regional variation. The areas with higher proportion of managerial and professional roles tend to be in London and the South East.

```
#a quick visual check to see whether regions are relevant
#graph including regions, need the regions from earlier
load("rda/region_lookup.rda")
load("rda/geo_lookup.rda")
# use the geo lookup and region from the region_lookup
geo_lookup %>%
  mutate(geo_code=geo_code_msoa) %>%
  select(geo_code, local_area) %>%
  unique() %>%
  left_join(region_lookup, by = "local_area") %>%
  select(geo_code, region) %>%
  #add the main data with y
  right_join(main) %>%
  #plot...
  ggplot(aes(y))  +
  geom_histogram(aes(fill=region), bins = 30) +
  # geom_histogram(bins = 30) +
  ggtitle("Distribution of the outcome, y") +
  xlab("y, proportions of people in managerial or professional occupations") +
  ylab("count of numbers of MSOA areas")
```



3.4.3 Recommendations

This leads to the following insight and recommendations:

- create ratio of occupied/all residents
- create local area feature, both numeric and columns for regions
- remove occupation & all_residents population numbers, geo name & geo type

- some features have little variation, and could be removed if needed
- some features are highly correlated and could be removed if needed

3.5 Creating training and test data sets

From the "main" data set a training and test set was created for the modelling phase, in the same way as before, 10% of the data.

```
# test set will be 10% of the data
set.seed(2001, sample.kind="Rounding")
test_index <- createDataPartition(y = main$y, times = 1, p = 0.1, list = FALSE)
train_set <- main[-test_index,]
test_set <- main[test_index,]
```

3.5.1 Data cleanse

The final test and training data for the modelling was created with a function which can later be run on the main/validation data prior to the final result creation. The following was carried out:

- create ratio of occupied/all
- create geographic area features
- remove occupation & all_residents population numbers, geo name & geo type

THe geographic features were created using the lookup prepared in the previous section. In addition, the below includes if" statements to be able to handle the different area types, the SOA and MSOA, as they require slightly different lookups for the geographic region.

```
load("rda/geo_lookup.rda")
#currently working for MSOA and SOA
data_cleanse <- function(data_set_name, geosize){
  #generate geo mapping info
  load("rda/geo_lookup.rda")
  print(paste("working on", geosize))
  #choose the msoa geo_code
  if(geosize=="msoa"){
    print("msoa geolookup")
    geo_lookup <- geo_lookup %>%
      mutate(geo_code=geo_code_msoa)  %>%
      select(-geo_code_soa, -geo_code_msoa) %>%
      unique()
  }
  #choose the soa geo_code
  if(geosize=="soa"){
    print("soa geolookup")
    geo_lookup <- geo_lookup %>%
      mutate(geo_code=geo_code_soa)  %>%
      select(-geo_code_soa, -geo_code_msoa) %>%
      unique()
  }
  # modify the data
  tmp <- data_set_name %>%
    # add the occupation ratio
    mutate(occ_ratio = occupation_all/all_residents, .after = all_residents) %>%
    # remove the unnecessary columns
    select(-occupation_all, -all_residents, -geo_type, -geo_name) %>%
    # add the geographic region columns
```

```
    left_join(geo_lookup) %>%
    # remove the unnecessary geography columns
    select(-local_area, -geo_code)
  #return the new object
  return(tmp)
}

#run on the test set and train set
test_set_final <- data_cleanse(test_set, "msoa")
```

```
## [1] "working on msoa"
## [1] "msoa geolookup"
```

```
train_set_final <- data_cleanse(train_set, "msoa")
```

```
## [1] "working on msoa"
## [1] "msoa geolookup"
```

3.5.2 Initial baseline RMSE

The root mean squared error is:

$RMSE = \sqrt{\frac{1}{N}\sum(\hat{y_n} - y_n)^2}$

where y_n is the proportion of managerial and professional occupations in area n, and $\hat{y_n}$ is the prediction, and N the total number of areas in the sample.

The simplest model is to assume that all areas have a similar proportion, and all variation is by a random distribution

$y_n = \mu + \epsilon_n$

where the random variation is represented by ϵ_n, and μ is the average of all ratings.

from the training data μ is estimated with the following model:

$\hat{y_n} = \hat{\mu}$

and applying to the test data gives a value of 0.09647.

```
#load the test and train data
# load("rda/test_set_final.rda")
# load("rda/train_set_final.rda")
#create RMSE function
rmse <- function(true_proportions, predicted_proportions){
  sqrt(mean((true_proportions - predicted_proportions)^2))
}
# baseline RMSE with an average of all ratings
mu_hat <- mean(train_set_final$y)
rmse_ave <- rmse(test_set_final$y, mu_hat)
rmse_results <- tibble(method = "Mean of all locations", rmse = rmse_ave)
rmse_results %>% knitr::kable(caption="Initial baseline RMSE on an SOA with a mean estimate")
```

Table 10: Initial baseline RMSE on an SOA with a mean estimate

method	rmse
Mean of all locations	0.09647

4 Modelling

This section covers the machine learning modelling to develop the optimal model, and exploring the benefits of an ensemble model. In addition, it covers selection of optimal features for the models, selection of models, and some visualisation of errors.

4.1 Feature selection

As part of the modelling process, it is helpful to have smaller data sets to test models, and also to remove the potentially difficult to handle categorical data. Then the algorithms can be tested quickly before trying on larger data sets.

From the earlier investigation, some features had low variability, and some were highly correlated, and these will be removed from the smaller training sets.

4.1.1 Low variance features

The low variance candidates have a mean <=0.05, standard deviation <=0.015, and max <0.15, which are the following.

```
# low variation
# choosing the features as candidates to remove
low_variability <- train_set_final %>%
  #pivot to a long version with a row per feature/value
  pivot_longer(cols = (contains("25_64") | contains("ratio")),
               names_to = "feature",
               values_to = "proportion") %>%
  group_by(feature) %>%
  summarise(mean=mean(proportion),
            sd=sd(proportion),
            max=max(proportion),
            min=min(proportion)) %>%
  filter(mean <= "0.05" & sd <= "0.015" & max <= "0.15")
low_variability %>%
  arrange(mean) %>%
  knitr::kable(captions="Summary of features with low variability")
```

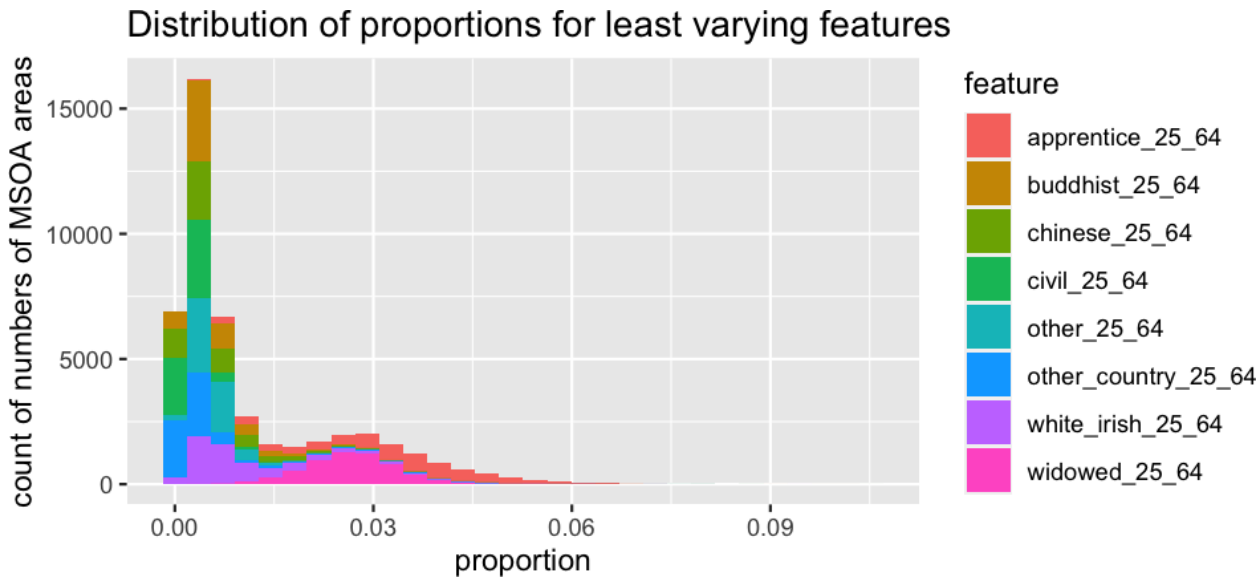
feature	mean	sd	max	min
civil_25_64	0.002660	0.002212	0.030985	0.000000
other_country_25_64	0.004394	0.007564	0.081054	0.000000
buddhist_25_64	0.005523	0.004948	0.091413	0.000000
other_25_64	0.005808	0.003877	0.088435	0.000000
chinese_25_64	0.006907	0.008631	0.105263	0.000000
white_irish_25_64	0.009883	0.008739	0.093198	0.000000
widowed_25_64	0.026470	0.006627	0.060459	0.003367
apprentice_25_64	0.032813	0.013375	0.101860	0.001608

```
# creating a list for later...
low_variability_list <- low_variability %>% .$feature
```

Putting together a graph, they look like they are not that interesting, with the exception perhaps of "widowed" and "apprentice".

```
# histogram of potential features to lose
train_set_final %>%
```

```
  #pivot to a long version with a row per feature/value
  pivot_longer(cols = (contains("25_64") | contains("ratio")),
               names_to = "feature",
               values_to = "proportion") %>%
  filter(feature %in% low_variability_list) %>%
  ggplot(aes(proportion, fill=feature))  +
  geom_histogram(bins = 30) +
  ggtitle("Distribution of proportions for least varying features") +
  ylab("count of numbers of MSOA areas")
```



4.1.2 Categorical features

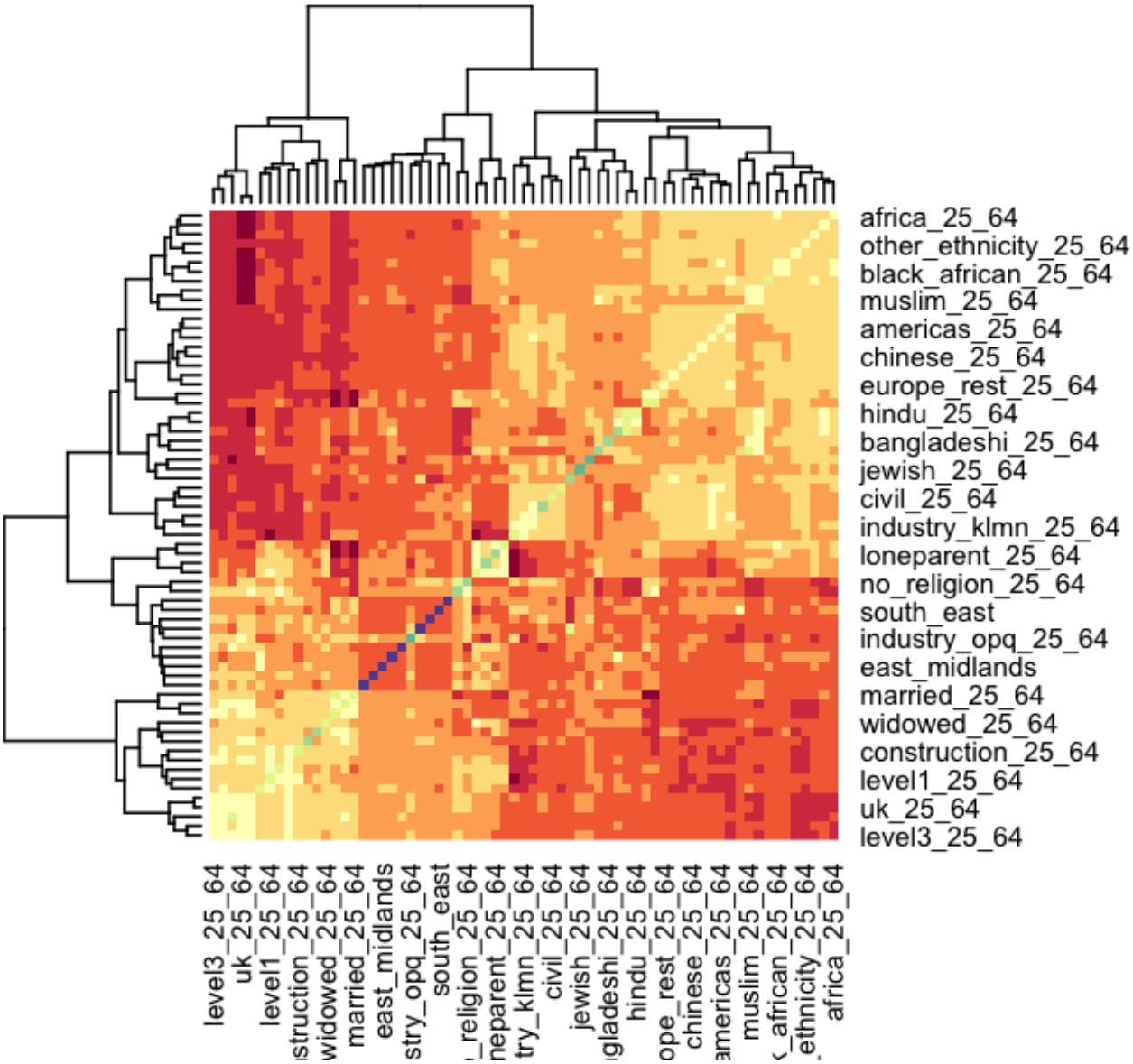
Some algorithms do not handle categorical features well. Although there is potentially useful information, as in, there is an association between the categories and the outcome, it is not possibly to calculate a correlation where there are few values. The training will use data sets without categorical features initially, and then test and bring them in later.

```
# categorical data
#this is the location data
load("rda/geo_lookup.rda")
categorical_features <- names(geo_lookup[4:13])
```

4.1.3 Highly correlated features

There are functions to calculate highly correlated features.

```
# load("rda/train_set_final.rda")
correlationmatrix <- cor(train_set_final[,3:69])
# image of correlation matrix
heatmap(x = correlationmatrix, col = RColorBrewer::brewer.pal(11, "Spectral"))
```



This generates a short list of features to remove. A cutoff of 0.8 was selected, with illustrations below.

```
# find attributes that are highly corrected
highlycorrelated <-
  findCorrelation(correlationmatrix, cutoff=0.8, exact = TRUE, names=TRUE)
#listing the correlations for a specific one of the selected features, say, 2, and 3
#create index for the feature, and show highly correlated other features
index <- str_which(names(data.frame(correlationmatrix)), highlycorrelated[2])
data.frame(correlationmatrix[,index][abs(correlationmatrix[,index]) > 0.7]) %>%
  rename("correlation" = starts_with("correlationmatri")) %>%
  rownames_to_column(var="feature name") %>%
  knitr::kable(caption="Highly correlated features with the 'white_uk' feature")
```

Table 12: Highly correlated features with the 'white_uk' feature

feature name	correlation
white_uk_25_64	1.000000
white_other_25_64	-0.720968
other_ethnicity_25_64	-0.851235
asian_other_25_64	-0.804588
black_other_25_64	-0.710204
level2_25_64	0.712417
level3_25_64	0.752717
other_qual_25_64	-0.888893
apprentice_25_64	0.728795
christian_25_64	0.752481
muslim_25_64	-0.765294
otherhousehold_25_64	-0.897115
uk_25_64	0.984678
africa_25_64	-0.832838
me_asia_25_64	-0.864764

4.1.4 Subset of data for initial testing

The data set for training has 5832 observations of 70 variables. Following are features that can be removed with limited impact, or to avoid categorical issues, in addition to removing the "geo-code" column as it is neither a predictor nor outcome:

Table 13: List of features to be removed for the smaller data sets

low variability	highly correlated	categorical
apprentice_25_64	uk_25_64	london
buddhist_25_64	white_uk_25_64	north_west
chinese_25_64	otherhousehold_25_64	yorkshire_humber
civil_25_64	level2_25_64	north_east
other_25_64	white_other_25_64	west_midlands
other_country_25_64	africa_25_64	east_midlands
white_irish_25_64	eu_2001_25_64	south_west
widowed_25_64	marriedfamily_25_64	east_england
	me_asia_25_64	south_east
	level1_25_64	wales
	single_25_64	
	muslim_25_64	
	level4_25_64	
	married_25_64	
	hindu_25_64	

This was used to create a new "train_small" with 5832 observations of 36 variables: 35 features and the outcome y.

And in addition, a further smaller training set was created with 1000 observations and 15 features, chosen by sampling.

```
## create smaller data set
#remove low variability, highly correlated and categorical columns
train_small <- train_set_final %>%
```

```
  select(-all_of(low_variability_list)) %>%
  select(-all_of(highlycorrelated)) %>%
  select(-all_of(categorical_features))

#make a smaller set, 15 columns and 1000 rows
#sampling the 1000 observations
set.seed(2001, sample.kind="Rounding")
index <- sample(1:nrow(train_small), 1000, replace = FALSE)
train_smaller <- train_small[index,]
# there are 39 features numbers 3:41 inclusive.
# creating the index, 15 from 39 plus 2
set.seed(1984, sample.kind="Rounding")
#create the sample of 15 features
index <- sample(1:(ncol(train_small)-1), 15, replace = FALSE)
#add 1 and add back in the y values
index2 <- append(1, (index+1))
train_smaller <- train_smaller[,index2]
```

In order to test the suitability of categorical data, a further three training sets were created, small and smaller with categorical, and the full set without categorical data. This made 6 training sets in total, full, small and smaller, with and without categorical features.

```
# then the same but with categorical variables
#remove low variability, highly correlated columns
train_small_cat <- train_set_final %>%
  select(-all_of(low_variability_list)) %>%
  select(-all_of(highlycorrelated))

#make a smaller set, 15 columns and 1000 rows
#sampling the 1000 observations
set.seed(2001, sample.kind="Rounding")
index <- sample(1:nrow(train_small_cat), 1000, replace = FALSE)
train_smaller_cat <- train_small_cat[index,]
# there are 35 features numbers 2:36 inclusive.
# creating the index, 15 from 35 plus 1
set.seed(1984, sample.kind="Rounding")
#create the sample of 15 features
# random, but a manual check shows two categorical variables were chosen
index <- sample(1:(ncol(train_small_cat)-1), 15, replace = FALSE)
#add one to include the y values
index2 <- append(1, (index+1))
train_smaller_cat <- train_smaller_cat[,index2]

#remove categorical columns
train_final_nocat <- train_set_final %>%
  select(-all_of(categorical_features))
```

4.2 Model selection

This project will be using the caret package within R to carry out the machine learning, testing out the results on a number of models. This section covers the selection of models to try out, the results per model and summarises the model performance.

This is a continuous rather than classification problem, therefore models suited to classification, such as Naive Bayes, Adaboost will not work.

4.2.1 Machine learning families

There are over a hundred different algorithms supported in the caret package, and instead of trying them all, examples of different types of models will be selected and tested, before focusing on the better performing types of models.

The model selection was carried out by grouping in to families, choosing one or two from each grouping. However, there does not appear to be a definitive list of algorithms and families, at least in part as there are many overlapping approaches. Below are the groupings used (with the caret package name in lower case), it is a matter of opinion as to where some algorithms fit, but this is not critical as the aim is to select a breadth of model types.

- Regression, regularisation
  - Linear Regression, Logistic Regression - glm
  - Generalized Additive Model using Splines - gam
  - Locally Estimated Scatterplot Smoothing - gamLoess
  - Least Absolute Shrinkage and Selection Operator - lasso
  - Support Vector Machines (linear kernel) - svmLinear
  - Ridge regression
  - Boosted Generalized Linear Model - glmboost
- Bayesian
  - Naive Bayes (classification)
  - Bayesian Generalized Linear Model - bayesglm
- Clustering, instance based
  - k-Nearest Neighbor - knn
  - Support Vector Machines (other kernel) - svmLinear, svmRadial, svmRadialCost, svmRadialSigma
  - Learning Vector Quantization (classification)
  - k-Means
- Trees / forests / gradient boosting
  - Classification and Regression Tree - rpart
  - Random Forest - rf, ranger, rborist
  - Weighted Subspace Random Forest - wsrf
  - gradient boosting ++ Stochastic Gradient Boosting - gbm ++ xgboost
  - adaboost (classification)
- Dimensionality reduction
  - Principal Component Regression - pcr
  - Polynomial Kernel Regularized Least Squares - krlsPoly
  - Linear Discriminant Analysis (Classification)
  - Quadratic Discriminant Analysis (Classification)
- Artificial Neural Network Algorithms, deep learning
  - Multilayer Perceptrons - mlp
  - Monotone Multi-Layer Perceptron Neural Network monmlp
  - Model Averaged Neural Network - avNNet

This was developed with input from various sources online, but the following being most important: https://topepo.github.io/caret/train-models-by-tag.html https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

4.2.2 Modelling function

A function was used to run the models without tuning, and store the data output in a list object, including the rmse.

```
# creating a function to run the models with default settings
# can set different algorithms and data sets
results_train_method <- function(traindata, algorithm){
  trainname <- paste("train",
```

```
                    deparse(substitute(algorithm)),
                    deparse(substitute(traindata)),
                    sep = "_")
  print(paste0("running ", trainname))
  #setting the random seed
  set.seed(2011, sample.kind="Rounding")
  #run the model training on the training data
  train_obj <- train(y ~ ., method = algorithm, data = traindata)
  # making a prediction on the test set
  yhat_obj <- predict(train_obj, test_set_final)
  # calculating the rmse
  rmse_obj <- rmse(test_set_final$y, yhat_obj)
  print(paste("rmse result is", rmse_obj))
  #adding to the RMSE list
  resultname <-
    paste0(deparse(substitute(algorithm)), " - ", deparse(substitute(traindata)))
  print(paste("creating result list for", resultname))
  rmse_results <- tibble(method = resultname, rmse = rmse_obj)
  #saving the results
  #returning the training model, y_hats, and rmse results
  my_list <- list("train" = train_obj, "y_hat" = yhat_obj, "results" = rmse_results)
  return(my_list)
  # return(rmse_results)
}
load("rda/train_small.rda")
load("rda/train_smaller.rda")
load("rda/train_small_cat.rda")
load("rda/train_smaller_cat.rda")
load("rda/train_final_nocat.rda")
load("rda/test_set_final.rda")
load("rda/train_set_final.rda")
```

4.2.3 Generalized linear model - glm

Starting with the simplest model, a linear model, using the Generalized linear model in the caret package, initally on the smallest training set, "train smaller" of 1000 observations of 15 features.

It takes no time, and gives a sensible order of importance for the variables, with "no qualification" first. And it provides a significant improvement in the rmse.

```
# initial the straightforward glm model
# with the smaller training set
result_glm_train_smaller <- results_train_method(train_smaller, "glm")
```

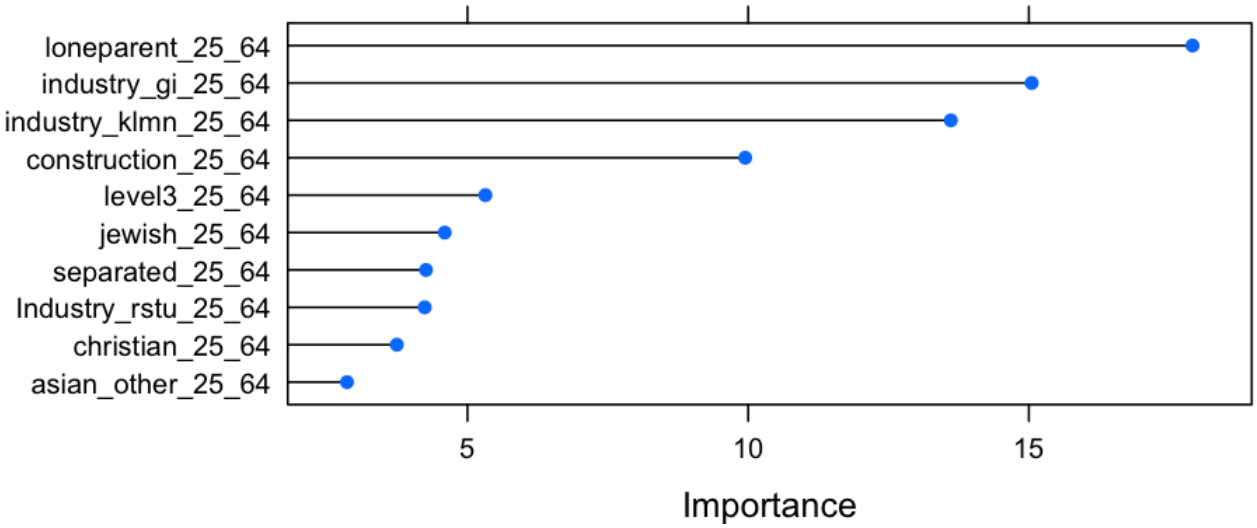
```
## [1] "running train_\"glm\"_train_smaller"
## [1] "rmse result is 0.039643386309209"
## [1] "creating result list for \"glm\" - train_smaller"
```

```
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_smaller$results, 1))
rmse_results %>% knitr::kable(caption="GLM model RMSE results for MSOA with the smallest data set")
```

Table 14: GLM model RMSE results for MSOA with the smallest data set

method	rmse
Mean of all locations	0.096470
"glm" - train_smaller	0.039643

```
#checking the variable importance
importance <- varImp(result_glm_train_smaller$train, scale=FALSE)
plot(importance, 10)
```



Increasing the size of the training set, the small set of 5832 by 35, with another jump up in performance. However there are a number of errors, although it worked, indicating some tuning opportunities. The highest predictors are all qualifications, and then age.

```
# with the small train set
result_glm_train_small <- results_train_method(train_small, "glm")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_small$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_glm_train_small$train, scale=FALSE)
plot(importance, 20)
```

With the full data, there is a further jump up in performance, and the level4 qualification is the most important feature.

```
# with the full train set, less categorical
result_glm_train_final_nocat <- results_train_method(train_final_nocat, "glm")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_final_nocat$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_glm_train_final_nocat$train, scale=FALSE)
plot(importance, 20)
```

The glm model does work with the categorical data, with a small improvement in performance.

```
# test if it will work with categorical feature the the smaller_cat
train_glm <- train(y ~ ., method = "glm", data = train_smaller_cat)
#glm does work...

# with the full train set
result_glm_train_set_final <- results_train_method(train_set_final, "glm")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_set_final$results, 1))
rmse_results %>% knitr::kable()
```

```
load("rda/result_glm_train_set_final.rda")
#checking the variable importance
importance <- varImp(result_glm_train_set_final$train, scale=FALSE)
plot(importance, 10)
```

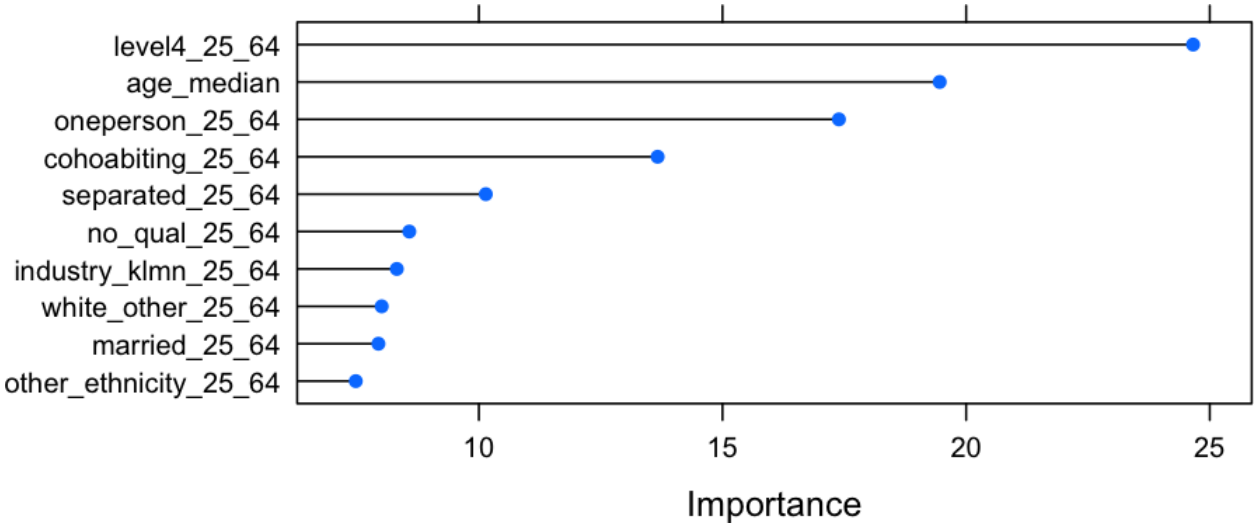


Table 15: GLM model RMSE results for MSOA

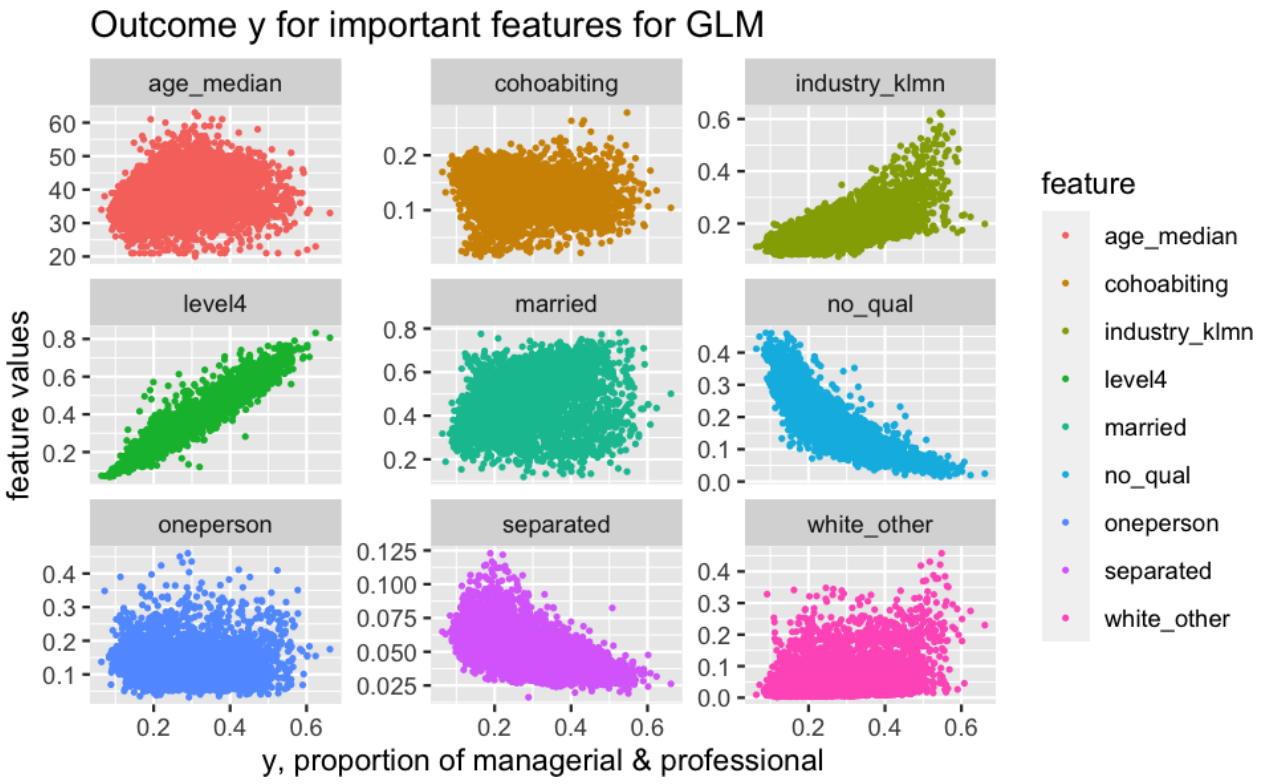
method	rmse
Mean of all locations	0.0964704
"glm" - train_set_final	0.0166809

4.2.3.1 Insights and visualisation The following were important features from the various versions of the model.

- all qualifications, level 4
- age_median
- oneperson_25_64
- loneparent_25_64
- cohoabiting_25_64
- construction_25_64
- eu_rest_25_64

Plotting the 9 most important features for GLM vs the outcome y, to visualise the links between the top features and the outcome, y, the proportion of residents with senior manager or professional occupation.

```r
#plotting important features vs y
important_features <- data.frame(importance$importance) %>%
  rownames_to_column(var="feature") %>%
  mutate(feature = str_replace_all(.$feature,"_25_64", "")) %>%
  arrange(desc(Overall)) %>%
  head(9) %>%
  pull(feature)
#taking the training data and making long form for a graph
train_set_final %>%
  #pivot to a long version with a row per feature/value
  rename_at(vars(ends_with("25_64")), ~str_replace_all(., "_25_64", "")) %>%
  pivot_longer(cols = !"y",
               names_to = "feature",
               values_to = "value") %>%
  filter(feature %in% important_features) %>%
  ggplot(aes(y, value, col=feature))  +
  geom_point(size=0.5) +
  facet_wrap(. ~feature,  scales = "free_y") +
  ggtitle("Outcome y for important features for GLM") +
  xlab("y, proportion of managerial & professional") +
  ylab("feature values")
```



4.2.4 K nearest neighbours - knn

The k nearest neighbours model, is another common model introduced during the course, a simple version of a cluster algorithm, "knn" in the caret package.

It took a few seconds with the smallest training set with a result a little worse than the GLM model, 0.038137. There is no ranking of feature importance. The model used 9 neighbours.

The larger training sets worked within a few minutes, however, the result was significantly worse than with a smaller training set, 0.070648. The number of features made no difference, but it was much better performing with a subset of the number of observations.

```
# with the smaller training set
result_knn_train_smaller <- results_train_method(train_smaller, "knn")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_knn_train_smaller$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_knn_train_smaller$train, scale=FALSE)
# check the model
result_knn_train_smaller$train$finalModel

# with the full train set, less categorical
result_knn_train_final_nocat <- results_train_method(train_final_nocat, "knn")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_knn_train_final_nocat$results, 1))
rmse_results %>% knitr::kable()
```

It is able to handle the categorical features, which makes sense, as it is not carrying out a regression analysis, however, the results were poor.

The algorithm is influenced by the absolute value of each feature and therefore the data needs to be normalised. The "smaller" training set does not include the age and location, which are the two features which have values over 1 and perhaps this is why the performance is better with the smaller set. As a quick test, the age and location features were from the small set, and the result was better than the smaller set at 0.029905, so on the right track, but still not as good as the GLM model. Using the same logic with the full data set, with no categorical data and no age or location gives a result of 0.023753.

```
# test if it will work with categorical feature the the smaller_cat
result_knn_train_smaller_cat<- results_train_method(train_smaller_cat, "knn")
# knn does work...

#making a new small training set, without the age and area
train_set_knn_small <- train_small %>%
  select(-age_median, -area_code)
#checking the max, should be less than 1
max(train_set_knn_small[1,])

#check with the new knn small training set
result_knn_train_set_knn_small <- results_train_method(train_set_knn_small, "knn")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_knn_train_set_knn_small$results, 1))
rmse_results %>% knitr::kable()

#making a new final training set, no categorical, without the age and area
train_set_knn_final_nocat <- train_final_nocat %>%
  select(-age_median, -area_code)
#checking the max, should be 1
max(train_set_knn_final_nocat[1,])

#check with the new training set
```

```
result_knn_train_set_knn_final_nocat <-
  results_train_method(train_set_knn_final_nocat, test_set_final, "knn")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_knn_train_set_knn_final_nocat$results, 1))
rmse_results %>% knitr::kable()
```

The features were then normalised to between 0 and 1, working on the full data set, less categorical data, using the calculation below:

$normalisedfeature = \frac{feature - min(feature)}{max(feature) - min(feature)}$

The summary function shows the variability of maximum, minimum and median.

```
#normalising the data, to be between 0 and 1
#look at some of the data, show the max and min
summary(train_final_nocat[,4:7])
```

```
##  white_uk_25_64   white_other_25_64 white_irish_25_64  indian_25_64
##  Min.   :0.0374   Min.   :0.00145   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.7602   1st Qu.:0.01686   1st Qu.:0.00423   1st Qu.:0.00304
##  Median :0.9069   Median :0.03071   Median :0.00699   Median :0.00817
##  Mean   :0.8116   Mean   :0.05294   Mean   :0.00988   Mean   :0.02583
##  3rd Qu.:0.9518   3rd Qu.:0.06295   3rd Qu.:0.01220   3rd Qu.:0.02326
##  Max.   :0.9891   Max.   :0.45761   Max.   :0.09320   Max.   :0.79785
```

```
#create a function to normalise
normalise <- function(feature) {
  return ((feature - min(feature)) / (max(feature) - min(feature))) }
#create a new training set, removing the y value and adding back in
train_final_norm <- train_final_nocat[1] %>%
  cbind(as.data.frame(lapply(train_final_nocat[2:59], normalise)))
#check the data again
# identical(names(train_final_norm), names(train_final_nocat))
print("after normalising")
```

```
## [1] "after normalising"
```

```
summary(train_final_norm[,4:7])
```

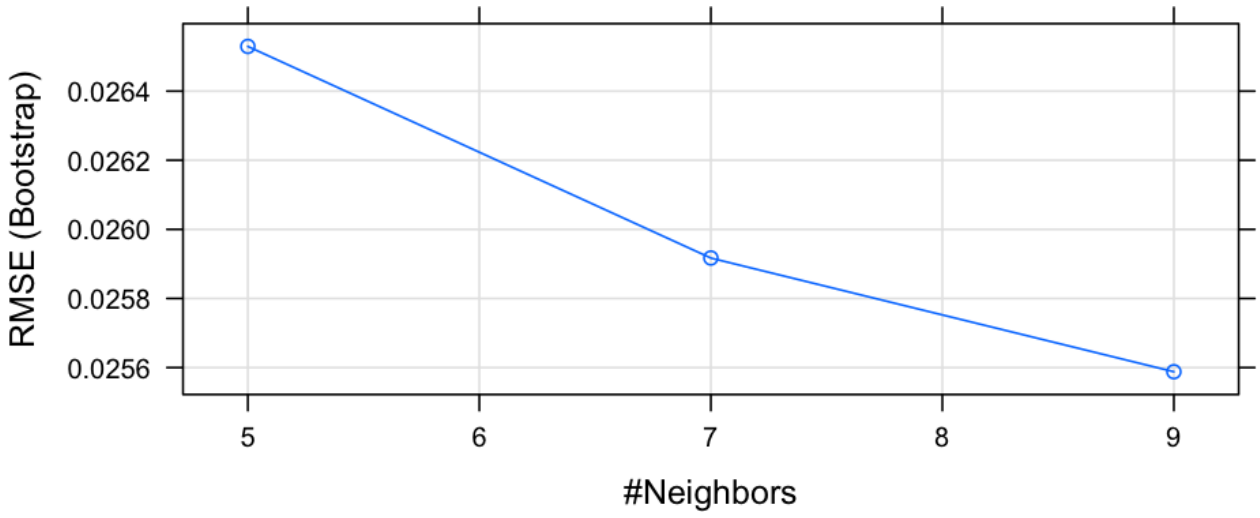
```
##  white_uk_25_64  white_other_25_64 white_irish_25_64  indian_25_64
##  Min.   :0.000   Min.   :0.0000    Min.   :0.0000    Min.   :0.00000
##  1st Qu.:0.760   1st Qu.:0.0338    1st Qu.:0.0454    1st Qu.:0.00381
##  Median :0.914   Median :0.0641    Median :0.0750    Median :0.01024
##  Mean   :0.813   Mean   :0.1129    Mean   :0.1060    Mean   :0.03238
##  3rd Qu.:0.961   3rd Qu.:0.1348    3rd Qu.:0.1309    3rd Qu.:0.02915
##  Max.   :1.000   Max.   :1.0000    Max.   :1.0000    Max.   :1.00000
```

This was run on the full data set, and did not perform well at all, worse than predicting the average at 0.102314. Perhaps with normalising, the algorithm is unable to discriminate features with little variation from those with big changes for the different areas.

```
#check with the new normalised training set
result_knn_train_final_norm <- results_train_method(train_final_norm, "knn")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_knn_train_final_norm$results, 1))
rmse_results %>% knitr::kable()
```

Overall KNN does not produce great results with this data set, however, it looks like the number of neighbours k can be improved.

```
load("rda/result_knn_train_set_knn_final_nocat.rda")
#checking the tuning for the best model
plot(result_knn_train_set_knn_final_nocat$train)
```



Running the model with more options for k gives a slightly improved value.

```
# trying more neighbours
train_knn <- train(y ~ ., method = "knn", data = train_set_knn_final_nocat, tuneLength=7)
# train_knnsmaller <- train(y ~ ., method = "knn", data = train_smaller, preProcess = c("center","scale
# 13 neighbours is best
plot(train_knn)
#making the prediction
yhat_knn <- predict(train_knn, test_set_final)
# calculating the rmse
rmse_knn <- rmse(test_set_final$y, yhat_knn)
#adding to the RMSE list
rmse_results <- bind_rows(rmse_results,
                          tibble(method='"knn" - train_set_knn_final_nocat, k=13',
                                 rmse = rmse_knn))
rmse_results %>% tail(5) %>% knitr::kable()
```

There may be ways to tune further, as the RMSE was improved to 0.023508 with the full data set less categorical features, age and location, using a k value of 13.

But given the gap to the leading model, there are many other algorithms to look at before any further tuning here.

Table 16: KNN model RMSE results for MSOA

method	rmse
Mean of all locations	0.0964704
“glm” - train_set_final	0.0166809
“knn” - train_set_knn_final_nocat, k=13	0.023508
“knn” - train_set_knn_final_nocat	0.0237531

4.2.5 Tree model CART - rpart

This is the simplest tree algorithm, and works with the smaller set with errors, and does not perform well, although it runs quickly, and worked on all the data sets.

The model trained on the small data set is based on the "no qualification" feature, with choices in the tree based on qualification proportions, visible in the importance plot as well. The full data set is based on the level4 qualification.

```
# uses the rpart package
if (!require('rpart')) install.packages('rpart'); library('rpart')
# with the smaller training set
result_rpart_train_smaller <- results_train_method(train_smaller, "rpart")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_rpart_train_smaller$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_rpart_train_smaller$train, scale=FALSE)
plot(importance, 20)
# looking at the model, it is simply based on the no_qualification feature
result_rpart_train_smaller$train$finalModel
# with the final set
result_rpart_train_set_final <- results_train_method(train_set_final, "rpart")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_rpart_train_set_final$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_rpart_train_set_final$train, scale=FALSE)
plot(importance, 20)
result_rpart_train_set_final$train$finalModel
```

Table 17: CART rpart model RMSE results for MSOA

method	rmse
Mean of all locations	0.0964704
"glm" - train_set_final	0.0166809
"rpart" - train_set_final	0.0507058

4.2.6 Support Vector Machines - svmLinear

This is a newer and popular model which is loosely in the clustering family, similar to KNN, but it also works by creating lines between groups, something a bit like linear regression related to the GLM. The documentation describes using a "kernel" to shift the data in to different dimensions allowing non-linear boundaries.

The model runs slowly, a few seconds with even the small set, but a good RMSE result, but does not provide an importance view. The model worked with all data sets, is able to handle categorical features, with the best results with the full data set.

```
# using the kernlab package
if (!require('kernlab')) install.packages('kernlab'); library('kernlab')
# with the smaller training set
result_svm_train_smaller <- results_train_method(train_smaller, "svmLinear")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
```

```
                          tail(result_svm_train_smaller$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance, doesn't work
# importance <- varImp(result_svm_train_smaller$train, scale=FALSE)
# with the full training set
result_svm_train_set_final <- results_train_method(train_set_final, "svmLinear")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_svm_train_set_final$results, 1))
rmse_results %>% knitr::kable()
```

Overall this is a good model, close to the GLM in performance, although requiring a lot of computation.

Table 18: SVM Linear model RMSE results for MSOA

method	rmse
Mean of all locations	0.0964704
"glm" - train_set_final	0.0166809
"svmLinear" - train_set_final	0.0167375

4.2.7 Stochastic Gradient Boosting - gbm

This is another popular style of model, where the "boosting" refers to iterative creation of models, trees, loosely in the tree/random forest group. In contrast to standard random forest models each new tree is created to minimise the errors from the previous result, "boosting" the performance

It runs well, on the all data sets, with results very similar to the GLM and SVM algorithms. It is not as slow as the SVM model to run, but the results are not quite as good.

There is no information on importance of variables, and it can handle the categorical features, with a small improvement.

```
# using the gbmm plyr, package
if (!require('gbm')) install.packages('gbm'); library('gbm')
if (!require('plyr')) install.packages('plyr'); library('plyr')
# with the full train set
result_gbm_train_set_final <- results_train_method(train_set_final, "gbm")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_gbm_train_set_final$results, 1))
rmse_results %>% knitr::kable()
```

Another potentially useful model.

Table 19: GBM model RMSE results for MSOA

method	rmse
Mean of all locations	0.096470
"glm" - train_set_final	0.016681
"gbm" - train_set_final	0.017467

4.2.8 Multilayer Perceptrons - mlp

This is part of the artifical neural network family so a different set of machine learning algorithms.

It works, but produces a fairly poor result with the smaller data set, taking a minute or so. And the small data set took a minute or two, with some errors, with a worse RMSE than the earlier result. The package had some interoperability issues with the caret package used throughout the rest of the analysis, so the analysis has not been used to avoid creating problems with reviewers' R installations.

4.2.9 Generalized Additive Model using Splines - gam

As GLM is still the leading model, and the linear version of SVM also performed well, another linear model was selected. GAM is another successful algorithm, which can be considered as part of the linear regression family. It operates by adding regression lines on top of each other (which additive in the name refers to), using a smoothing function, in this case a function called "splines".

It worked well with the smaller data set, and does provide the important variables, for the smaller set, being loneparent.

It runs slowly though, and did not complete the training with the "small" data set, with increased observations and features.

```
# this uses the mgcv and nlme packages
if (!require('mgcv')) install.packages('mgcv'); library('mgcv')
if (!require('nlme')) install.packages('nlme'); library('nlme')

# with the smaller training set
result_gam_train_smaller <- results_train_method(train_smaller, "gam")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_gam_train_smaller$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_gam_train_smaller$train, scale=FALSE)
plot(importance, 20)
result_gam_train_smaller$train$finalModel

# with the small training set
result_gam_train_small <- results_train_method(train_small, "gam")
# did not complete, took too long
```

The performance with the smaller training set is good, and there may be options to reduce the number of features evaluated, perhaps using another model to identify a small number of features to work with. This is a popular and successful type of model, so it probably worth looking at further.

Table 20: GAM model RMSE results for MSOA, with other results for smaller data sets

method	rmse
Mean of all locations	0.096470
"glm" - train_set_final	0.016681
"gbm" - train_smaller	0.033540
"gam" - train_smaller	0.036524
"glm" - train_smaller	0.039643

4.2.10 Least Absolute Shrinkage and Selection Operator - lasso

Again looking at linear regression models, related to GLM, another model is Lasso, a regularisation model, or logic regression. It produces decent results with the smaller training set, with quick results from the other

data set, handling categorical data. But the rmse results look identical to the GLM model.

Table 21: Lasso model RMSE results for MSOA

method	rmse
Mean of all locations	0.0964704
"glm" - train_small	0.0244498
"glm" - train_set_final	0.0166809
"lasso" - train_small	0.0244498
"lasso" - train_set_final	0.0166809

```
# this uses the lasso package
if (!require('elasticnet')) install.packages('elasticnet'); library('elasticnet')

# with the smaller training set
result_lasso_train_smaller <- results_train_method(train_smaller, "lasso")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_lasso_train_smaller$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance, doesn't work
# importance <- varImp(result_lasso_train_smaller$train, scale=FALSE)
result_lasso_train_smaller$train$finalModel

# with the small training set
result_lasso_train_small <- results_train_method(train_small, "lasso")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_lasso_train_small$results, 1))
rmse_results %>% knitr::kable()

# with the full train set, less categorical
# with the small training set
result_lasso_train_final_nocat <- results_train_method(train_final_nocat, "lasso")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_lasso_train_final_nocat$results, 1))
rmse_results %>% knitr::kable()

# does it work with categortical?
train_lasso <- train(y ~ ., method = "lasso", data = train_smaller_cat)
#yes it does

#compare glm and lasso predictions
result_lasso_train_set_final <- results_train_method(train_set_final, "lasso")
rmse_results <- bind_rows(rmse_results, tail(result_lasso_train_set_final$results, 1))
result_lasso_train_set_final$results$rmse
```

A visual inspection of the y_hat for glm and lasso models, they are the same, however they are not identical using the "identical" function, although this may be due to rounding. The "all.equal" function allows for some slight differences, and this shows that they are identical across the RMSE and $\hat{y}$ predictions, as can be seen from the plot.

```
load("rda/result_lasso_train_set_final.rda")
load("rda/result_glm_train_set_final.rda")
# check whether the RMSE are identical, or almost
identical(result_lasso_train_set_final$results$rmse,
```

```
          result_glm_train_set_final$results$rmse)
```

```
## [1] FALSE
```

```
all.equal(result_lasso_train_set_final$results$rmse,
          result_glm_train_set_final$results$rmse)
```

```
## [1] TRUE
```

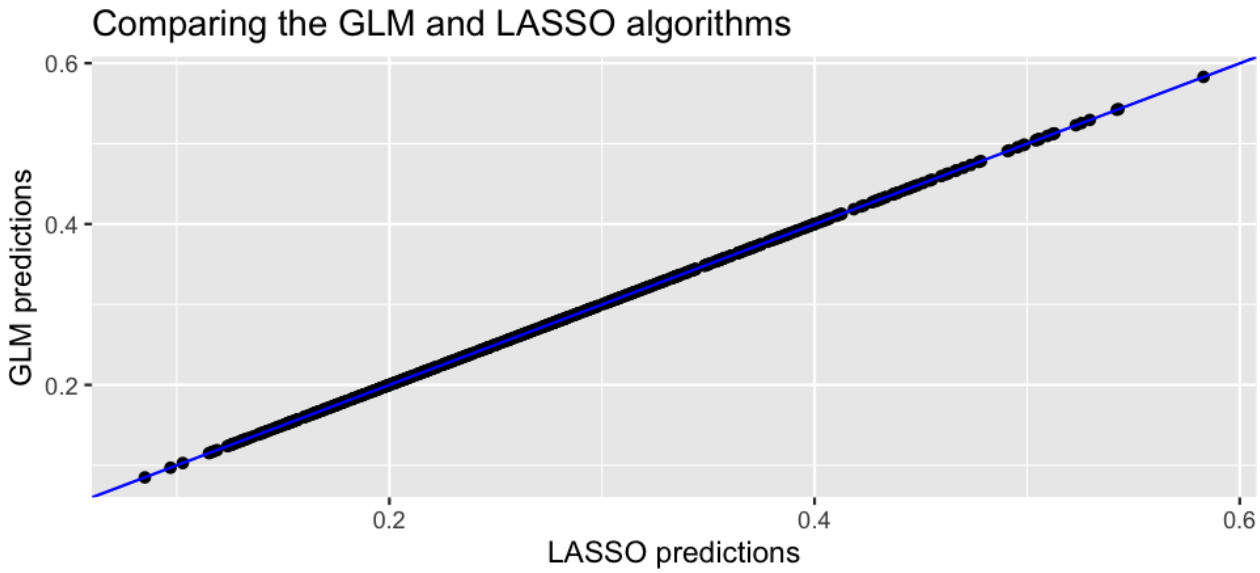
```
# check whether the y_hat are identical, or almost
identical(result_lasso_train_set_final$y_hat,
          result_glm_train_set_final$y_hat)
```

```
## [1] FALSE
```

```
all.equal(result_lasso_train_set_final$y_hat,
          result_glm_train_set_final$y_hat)
```

```
## [1] TRUE
```

```
# comparing glm and lasso
# plotting graphs
data.frame(result_lasso_train_set_final$y_hat,
           result_glm_train_set_final$y_hat) %>%
  ggplot(aes(result_lasso_train_set_final$y_hat,
             result_glm_train_set_final$y_hat)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col="blue")  +
  ggtitle("Comparing the GLM and LASSO algorithms") +
  xlab("LASSO predictions") +
  ylab("GLM predictions")
```



4.2.11 Principal Component Analysis - pcr

Now choosing a different type of model, this is a dimension reduction model, a family which has good results for recommendation problems with large data sets for example. It runs quickly, with no problems, but the performance is not good and gets worse with the full data set. It appears to have a problem with large number of features.

```r
# uses package "pls"
if (!require('pls')) install.packages('pls'); library('pls')
# with the smaller training set
result_pcr_train_smaller <- results_train_method(train_smaller, "pcr")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_pcr_train_smaller$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance, doesn't work
# importance <- varImp(result_pcr_train_smaller$train, scale=FALSE)
result_pcr_train_smaller$train$modelInfo

# with the small training set
result_pcr_train_small <- results_train_method(train_small, "pcr")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_pcr_train_small$results, 1))
rmse_results %>% knitr::kable()

#test the categorical
result_pcr_train_smaller_cat <- results_train_method(train_smaller_cat, "pcr")
# it works

# with the full train set
result_pcr_train_set_final <- results_train_method(train_set_final, "pcr")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_pcr_train_set_final$results, 1))
rmse_results %>% knitr::kable()
#performs badly though
```

Table 22: PCR model RMSE results for MSOA

method	rmse
Mean of all locations	0.0964704
“pcr” - train_smaller	0.0539450
“pcr” - train_small	0.0460373
“pcr” - train_set_final	0.0869626

4.2.12 Bayesian Generalized Linear Model - bayesglm

From another family, Bayes, but a regression model as most of the Bayes models are for categorical problems. It runs fine with the fist smaller data set, and with the small data set. No feature importance unfortunately, although the “finalmodel” does give detail of the coefficients used.

The results in terms of y_hat predictions are very similar to the GLM.

Table 23: BayesGLM model RMSE results for MSOA

method	rmse
Mean of all locations	0.0964704
“glm” - train_small	0.0244498
“glm” - train_set_final	0.0166809

method	rmse
“bayesglm” - train_smaller	0.0396911
“bayesglm” - train_small	0.0244478

```
# uses the arm package
if (!require('arm')) install.packages('arm'); library('arm')
# bayesglm
# with the smaller training set
result_bayesglm_train_smaller <- results_train_method(train_smaller, "bayesglm")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_bayesglm_train_smaller$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance, doesn't work
# importance <- varImp(result_bayesglm_train_smaller$train, scale=FALSE)
result_bayesglm_train_smaller$train$finalModel

# with the small training set
result_bayesglm_train_small <- results_train_method(train_small, "bayesglm")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_bayesglm_train_small$results, 1))
rmse_results %>% knitr::kable()
#checking the final model
result_bayesglm_train_small$train$finalModel

# with the final train set
result_bayesglm_train_set_final <- results_train_method(train_set_final, "bayesglm")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_bayesglm_train_set_final$results, 1))
rmse_results %>% knitr::kable()
```

A visual inspection of the y_hat for glm and Bayesglm models, they are the the same, however they are not identical using the “identical” function, and the “all.equal” gives a difference of 3.148678e-05 for the RMSE, and 0.0002200795 across the $\hat{y}$ predictions, so slightly different, but similar enough to only use one of the models, the GLM.

```
load("rda/result_bayesglm_train_set_final.rda")
load("rda/result_glm_train_set_final.rda")
# check whether the RMSE are identical, or almost
identical(result_bayesglm_train_set_final$results$rmse,
          result_glm_train_set_final$results$rmse)
```

```
## [1] FALSE
```

```
all.equal(result_bayesglm_train_set_final$results$rmse,
          result_glm_train_set_final$results$rmse)
```

```
## [1] "Mean relative difference: 3.14868e-05"
```

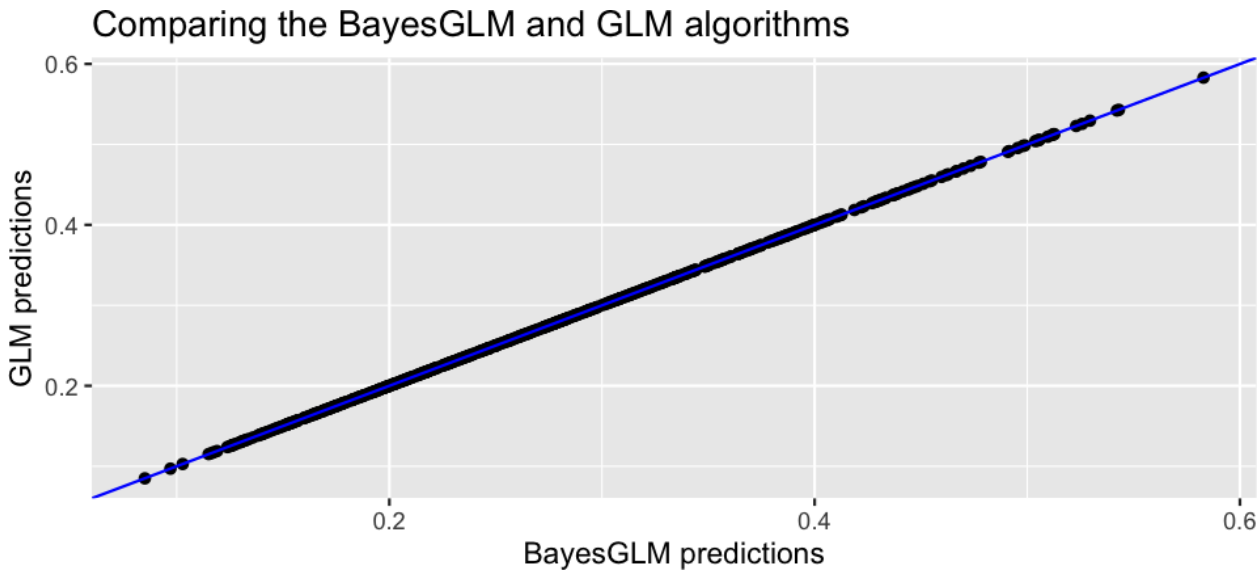
```
# check whether the y_hat are identical, or almost
identical(result_bayesglm_train_set_final$y_hat,
          result_glm_train_set_final$y_hat)
```

```
## [1] FALSE
```

```
all.equal(result_bayesglm_train_set_final$y_hat,
          result_glm_train_set_final$y_hat)
```

```
## [1] "Mean relative difference: 0.000220079"
```

```
# plotting graphs
data.frame(result_bayesglm_train_set_final$y_hat,
           result_glm_train_set_final$y_hat) %>%
  ggplot(aes(result_bayesglm_train_set_final$y_hat,
             result_glm_train_set_final$y_hat)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col="blue")  +
  ggtitle("Comparing the BayesGLM and GLM algorithms") +
  xlab("BayesGLM predictions") +
  ylab("GLM predictions")
```



4.2.13 Generalized Additive Model using LOESS - gamLoess

As the GAM model is in the linear regression family, and the GAM ran well on the smaller set, an alternative using LOESS for the additive regression models. However, it appears to suffer the same issues, being slow with lots of warnings, even for the smaller data set, and taking too long with the small data set.

With the smaller data set the important features were the industries, in contrast to most other models which focused on the lone parent, but the performance in terms of RMSE was not bad for the smaller set.

```
# uses the gam package
if (!require('gam')) install.packages('gam'); library('gam')
# with the smaller training set
result_gamloess_train_smaller <- results_train_method(train_smaller, "gamLoess")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_gamloess_train_smaller$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_gamloess_train_smaller$train, scale=FALSE)
plot(importance, 20)
```

```
# with the small training set
result_gamloess_train_small <- results_train_method(train_small, "gamLoess")
# did not complete
```

As with the GAM model, the as the performance with the smaller training set is good, it probably worth looking at further.

Table 24: GAM loess model RMSE results for MSOA, including other smaller data set results

method	rmse
Mean of all locations	0.096470
“glm” - train_set_final	0.016681
“gbm” - train_smaller	0.033540
“gamLoess” - train_smaller	0.038299
“glm” - train_smaller	0.039643

4.2.14 Random Forest - ranger

Now exploring the tree family further after the poorly performing simple tree model CART-rpart. The Ranger model is one of the random forest variants, and it was a big improvement, but slow, due to the many iterations to build the forest of trees.

It ran with all the data sets including the categorical, but slowly, and very high CPU. The RMSE performance was behind the leading models.

```
# uses the e1071, ranger, dplyr packages
if (!require('ranger')) install.packages('ranger'); library('ranger')
if (!require('e1071')) install.packages('e1071'); library('e1071')?

# with the full training set
result_ranger_train_set_final <- results_train_method(train_set_final, "ranger")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_ranger_train_set_final$results, 1))
rmse_results %>% knitr::kable()
```

Table 25: Ranger model RMSE results for MSOA

method	rmse
“glm” - train_set_final	0.016681
“ranger” - train_set_final	0.018276

4.2.15 Boosted Generalized Linear Model - glmboost

Trying a further variant of the GLM model, a boosted model, meaning that there are multiple iterations, each building on the next to improve the model.

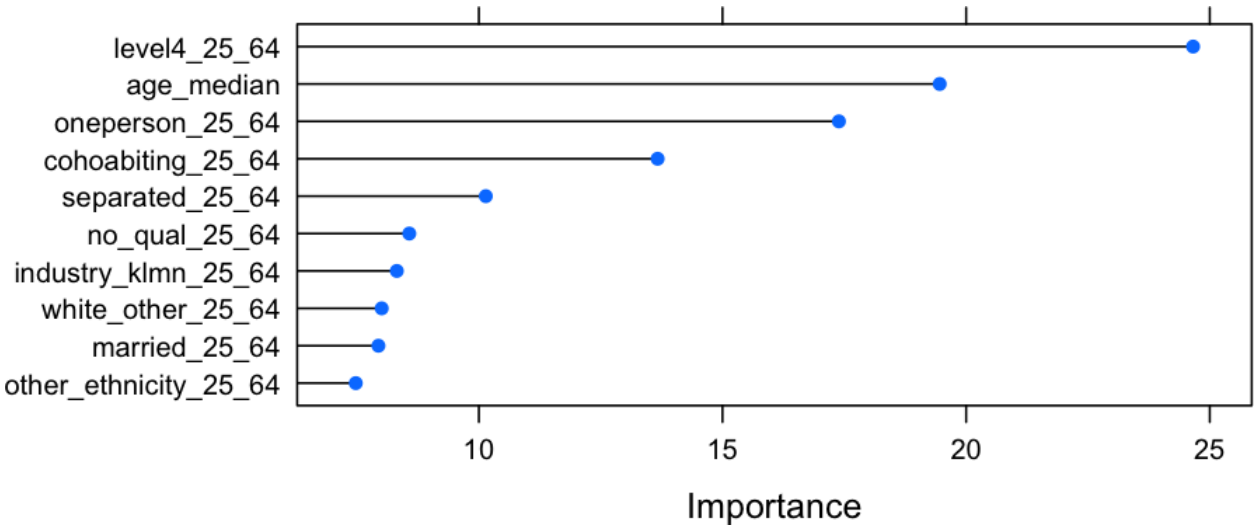
This runs fine on the smaller set, with importance, and is driven by the lone parent as for other models. It runs quickly, without errors on all the data sets, includint the categorical features, and with the final set it is marginally better.

```
# uses the plyr, mboost packages
if (!require('mboost')) install.packages('mboost'); library('mboost')
if (!require('plyr')) install.packages('plyr'); library('plyr')

# with the full training set
result_glmboost_train_set_final <- results_train_method(train_set_final, "glmboost")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glmboost_train_set_final$results, 1))
rmse_results %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_glmboost_train_set_final$train, scale=FALSE)
plot(importance, 20)
```

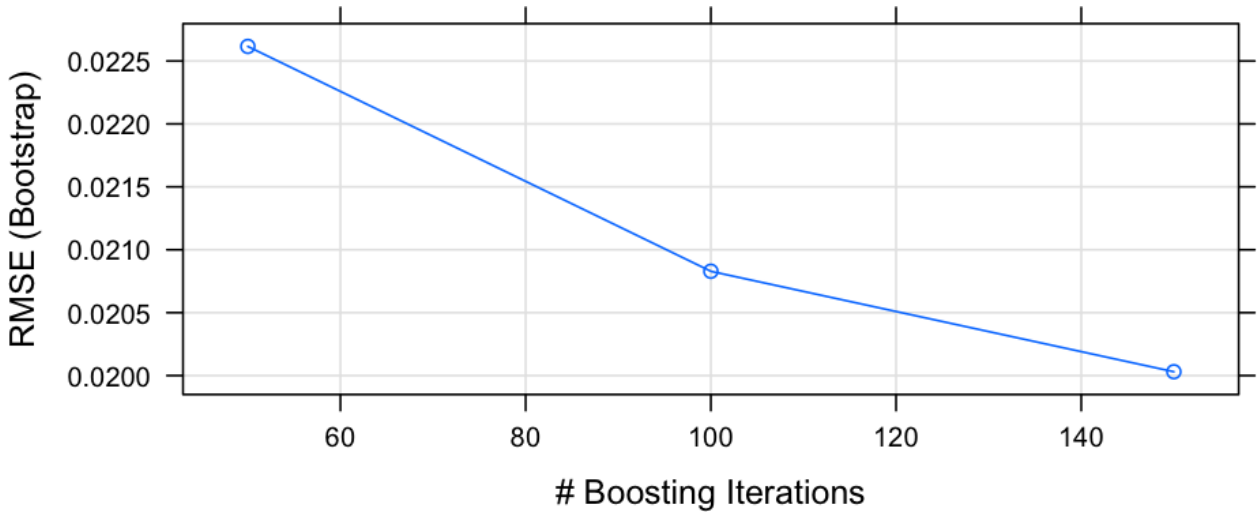
The model does provide the importance which will be helpful later. Again level4 qualifications are important, but also age, and household situations.

```
# important features
load("rda/importance_glmboost.rda")
plot(importance, 10)
```



Reviewing the model, it looks like the number of iterations of "boosting" could be increased which would increase the accuracy. However, it is far away from the leading models, and this would not close that large a gap, and also, that is likely to overfit the specific train/test data sets.

```
# tuning for glmboost
load("rda/result_glmboost_train_set_final.rda")
#checking tuning
plot(result_glmboost_train_set_final$train)
```



Overall, although fast, the performance of the GLMboost model is not that good. It does provide the importance of the features, which may be helpful later, particularly when it comes to interpretation.

Table 26: GLM Boost model RMSE results for MSOA

method	rmse
"glm" - train_set_final	0.016681
"glmboost" - train_set_final	0.020032

4.2.16 Model Averaged Neural Network - avNNet

This is a neural network model, the first attempted. There is little documentation, but it appears to operate as a deep learning algorithm with a hidden layer between the features and outcome, estimating weights between them, being a value describing the strength of the link between them. This is modelled a number of times with random initial values, averaged for the final result. For example, the initial model with the smaller data set is described as 15-5-1 (15 features, 1 outcome, 5 hidden points) with 86 weights between them.

With the full training set, the performance was good, not as good as the GLM, but better than most. There is no information on the importance of features, and it is quite slow due to the need to compute so many variables, so many times.

```
# uses the nnet package
if (!require('nnet')) install.packages('nnet'); library('nnet')

# with the full train set
result_avnnet_train_set_final <- results_train_method(train_set_final, "avNNet")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_avnnet_train_set_final$results, 1))
rmse_results %>% knitr::kable()
```

Table 27: avNNet model RMSE results for MSOA

method	rmse
"glm" - train_set_final	0.016681
"avNNet" - train_set_final	0.019568

4.2.17 Support Vector Machines with Linear Kernel - svmLinear2

Selecting another SVM model, as the SVM linear was the best so far other than GLM. As linear models have performed well, this is an attempt with a different linear model, using a different package.

It is slow, works with all data sets, with no feature importance, but providing good results.

```
# uses the e1071 package
if (!require('e1071')) install.packages('e1071'); library('e1071')

# compare svmlinear and svmlinear2 predictions
result_svm2_train_set_final <- results_train_method(train_set_final, "svmLinear2")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_svm2_train_set_final$results, 1))
rmse_results %>% knitr::kable()
```

In the end this has virtually identical results to svmlinear.

Table 28: SVM Linear2 model RMSE results for MSOA and comparison with SVM Linear

method	rmse
“svmLinear” - train_smaller	0.039519
“svmLinear” - train_set_final	0.016737
“svmLinear2” - train_smaller	0.039501
“svmLinear2” - train_set_final	0.016748

Visual inspection shows a small difference with the RMSE and also the y_hats seem very close, tracking each other, with a “Mean relative difference” through the all.equal function of 0.0003343818. The project will focus on the svmlinear and not explore this version any further.

```
load("rda/result_svm2_train_set_final.rda")
load("rda/result_svm_train_set_final.rda")
# result_svm2_train_set_final$results$rmse
# result_svm_train_set_final$results$rmse
# check whether the RMSE are identical, or almost
identical(result_svm2_train_set_final$results$rmse,
          result_svm_train_set_final$results$rmse)
```

```
## [1] FALSE
```

```
all.equal(result_svm2_train_set_final$results$rmse,
          result_svm_train_set_final$results$rmse)
```

```
## [1] "Mean relative difference: 0.000616187"
```

```
# check whether the y_hat are identical, or almost
identical(result_svm2_train_set_final$y_hat,
          result_svm_train_set_final$y_hat)
```

```
## [1] FALSE
```

```
all.equal(result_svm2_train_set_final$y_hat,
          result_svm_train_set_final$y_hat)
```

```
## [1] "Mean relative difference: 0.000334382"
```

4.2.18 Support Vector Machines with Radial Kernel - svmRadial

For a final model trying another variant of the SVM model as it is the best performing so far after the GLM, and a much discussed model type. There are a number of variants which use different "kernels" which create the abstracted relationships between the features. The SVM Radial was selected, which although using the same package, "kernlab", it handles the non-linear elements differently.

This model runs a little more quickly than the svmlinear and svmlinear2 models, works on all data sets, and is also much better performing, the leading model so far, by some distance. Interestingly the categorical data, so the regional data, makes a big improvement on the overall RMSE results.

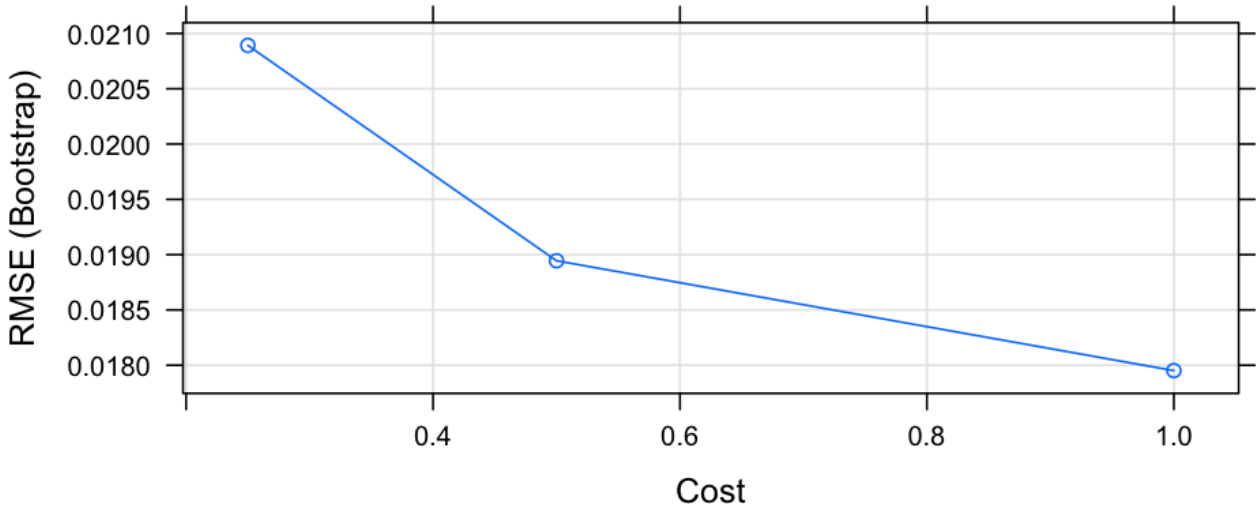
```
# uses package kernlab
if (!require('kernlab')) install.packages('kernlab'); library('kernlab')

# with the full train set, less categorical
result_svmradial_train_final_nocat <-
  results_train_method(train_final_nocat, test_set_final, "svmRadial")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_svmradial_train_final_nocat$results, 1))
rmse_results %>% knitr::kable()

# with the full train set
result_svmradial_train_set_final <- results_train_method(train_set_final, "svmRadial")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_svmradial_train_set_final$results, 1))
rmse_results %>% knitr::kable()
```

There is a tuning parameter which could make the model even better, the cost

```
load("rda/result_svmradial_train_set_final.rda")
# checking the tuning
plot(result_svmradial_train_set_final$train)
```




```
# result_svmradial_train_set_final$train
```

Table 29: SVM Radial model RMSE results for MSOA

method	rmse
“glm” - train_set_final	0.0166809
“svmLinear” - train_set_final	0.0167375
“svmRadial” - train_final_nocat	0.0157481
“svmRadial” - train_set_final	0.0145255

4.2.19 Model summary

The table below shows the leading models being SVM Radial, and behind that the GLM and SVM Linear. In addition the GAM and GAM Loess both performed well on the smaller data set, and there are a handful providing importance values.

Table 30: Summary of model RMSE performance for MSOA

Name	Family	Results	RMSE	Speed	Importance
svmRadial	clustering	very good	0.014525	medium	no
glm	linear regression	very good	0.016681	fast	yes
svmlinear	linear / clustering	very good	0.016737	slow	no
gbm	boosted trees	good	0.017467	medium	yes
ranger	tree / random forest	ok/good	0.018276	slow	no
avNNet	deep learning	ok/good	0.019568	medium	no
glmboost	boosted linear	ok	0.020032	fast	yes
gam	regression	good on smaller	0.036524	slow	no
gamLoess	regression	good on smaller	0.038299	slow	yes
knn	clustering	poor	0.023753	medium	no
pcr	dimension reduction	poor	0.046037	fast	no
rpart	tree	poor	0.050706	fast	yes
mlp	deep learning	poor		slow	no

svmLinear2 - similar to svmlinear
lasso - duplicate of the glm model
bayesglm - similar to the glm

4.3 Ensemble model

An ensemble will was created as the final model. As part of this further work was carried out to make use of the GAM and GAM Loess models, for use in the final ensemble.

As this is a regression problem, the ensemble prediction is the average of the y_hat predictions for the models.

Following are the leading models.

```
load("rda/rmse_results.rda")
# Identify the leading models, the top 5
rmse_results %>%
  arrange(rmse) %>%
  filter(str_detect(method, "train_set_final") ) %>%
  head(10) %>% knitr::kable(caption="Top 10 models by RMSE")
```

Table 31: Top 10 models by RMSE

method	rmse
“svmRadial” - train_set_final	0.014525
“bayesglm” - train_set_final	0.016680
“glm” - train_set_final	0.016681
“glm” - train_set_final	0.016681
“lasso” - train_set_final	0.016681
“svmLinear” - train_set_final	0.016737
“svmLinear2” - train_set_final	0.016748
“gbm” - train_set_final	0.017467
“ranger” - train_set_final	0.018276
“avNNet” - train_set_final	0.019568

Initially the 5 best models were used:

- Support Vector Machines with Radial Kernel - svmRadial
- Generalized linear model - glm
- Support Vector Machines - svmLinear
- Stochastic Gradient Boosting - gbm
- Random Forest - ranger

This produced a result of 0.015242, this is much worse than the svmRadial on its own at 0.0145255.

```
# create an ensemble y hats
load("rda/result_glm_train_set_final.rda")
load("rda/result_svm_train_set_final.rda")
load("rda/result_svmradial_train_set_final.rda")
load("rda/result_gbm_train_set_final.rda")
load("rda/result_ranger_train_set_final.rda")

# create table of y_hat
y_hat_ens_table <-
  data.frame(number = 1:length(result_glm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_glm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_gbm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_svm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_svmradial_train_set_final$y_hat)) %>%
  cbind(data.frame(result_ranger_train_set_final$y_hat)) %>%
  select(-number) %>%
  #calculate average of the y_hats
  mutate(y_hat_ave = rowMeans(.))
#test against the actual values
rmse_ens <- rmse(test_set_final$y, y_hat_ens_table$y_hat_ave)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Ensemble glm, gbm, svm, svmRadial, ranger",
                                 rmse = rmse_ens))
```

As the 2 of the models were quite a lot worse, a new ensemble was run with the leading 3 models (GLM, SVM Linear, SVM Radial), which provided a better still RMSE, 0.0149617, but still behind the svmRadial model.

```
#a smaller ensemble of the three best models
y_hat_ens_table2 <-
  data.frame(number = 1:length(result_glm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_glm_train_set_final$y_hat)) %>%
```

```
  cbind(data.frame(result_svm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_svmradial_train_set_final$y_hat)) %>%
  select(-number) %>%
  #calculate average of the y_hats
  mutate(y_hat_ave = rowMeans(.))
#test against the actual values
rmse_ens <- rmse(test_set_final$y, y_hat_ens_table2$y_hat_ave)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Ensemble2 glm, svm, svmRadial",
                                 rmse = rmse_ens))
rmse_results %>% tail(3) %>% knitr::kable(caption="Ensemble RMSE performance")
```

Table 32: Ensemble RMSE performance

method	rmse
Ensemble glm, gbm, svm, svmRadial, ranger	0.015242
Ensemble glm, gbm, svm, svmRadial, ranger	0.015242
Ensemble2 glm, svm, svmRadial	0.014962

4.3.1 Improving other models through feature selection

The GAM, GAMLoess models look promising but do not work on larger data sets, and the KNN had poor results with larger data sets. This may be due to the number of features, and given that from the models that show it, the importance drops off quite quickly we may be able to summarise the features to make use of these models.

```
rmse_results %>%
  filter(str_detect(method, "smaller") ) %>%
  arrange(rmse) %>% head(10) %>%
  knitr::kable(caption="Leading models with the smaller data set")
```

Table 33: Leading models with the smaller data set

method	rmse
“ranger” - train_smaller	0.033404
“gbm” - train_smaller	0.033540
“avNNet” - train_smaller	0.034365
“gam” - train_smaller	0.036524
“kknn” - train_smaller	0.036681
“svmRadial” - train_smaller	0.037099
“knn” - train_smaller	0.038137
“gamLoess” - train_smaller	0.038299
“gamLoess” - train_smaller	0.038299
“svmLinear2” - train_smaller	0.039501

Of the models that worked on the full set, there are three which provided information on the importance of features, GBM, GLM and GLMBoost. From the separate models, the importance was averaged using two methods, average of the ranking and average of the actual scores, normalised. The latter should provide the better result overall, and with testing it does so.

```
# using GLM, GBM, GLM boost to rank features
load("rda/importance_gbm.rda")
```

```
load("rda/importance_glm.rda")
load("rda/importance_glmboost.rda")
#create ranking for gbm algorithm, with normalised score
gbm_rank <- importance_gbm$importance %>%
  rownames_to_column(var = "feature") %>%
  arrange(desc(Overall)) %>%
  cbind(data.frame(gbm_rank = 1:length(importance_gbm$importance$Overall))) %>%
  mutate("gbm_overall" = Overall/max(Overall)) %>%
  select(-Overall)
# head(gbm_rank)
#create ranking for glm algorithm, with normalised score
glm_rank <- importance_glm$importance %>%
  rownames_to_column(var = "feature") %>%
  arrange(desc(Overall)) %>%
  cbind(data.frame(glm_rank = 1:length(importance_glm$importance$Overall))) %>%
  mutate("glm_overall" = Overall/max(Overall)) %>%
  select(-Overall)
# head(glm_rank)
#create ranking for glmboost algorithm, with normalised score
glmboost_rank <- importance_glmboost$importance %>%
  rownames_to_column(var = "feature") %>%
  arrange(desc(Overall)) %>%
  cbind(data.frame(glmboost_rank = 1:length(importance_glmboost$importance$Overall))) %>%
  mutate("glmboost_overall" = Overall/max(Overall)) %>%
  select(-Overall)

# combined ranking table, with mean, and reordered
feature_rank <- glm_rank %>%
  left_join(gbm_rank) %>%
  left_join(glmboost_rank) %>%
  mutate(mean_overall = (glm_overall+gbm_overall+glmboost_overall)/3, .after = feature) %>%
  mutate(mean_rank = (glm_rank+gbm_rank+glmboost_rank)/3, .after = feature)
# arrange by mean ranking and display top 9
feature_rank %>% select(feature, mean_rank, mean_overall) %>%
  arrange(mean_rank) %>% head(9) %>%
  knitr::kable(caption="Top 9 most important features by rank")
```

Table 34: Top 9 most important features by rank

feature	mean_rank	mean_overall
level4_25_64	1.00000	1.000000
no_qual_25_64	5.33333	0.177216
industry_klmn_25_64	6.66667	0.194686
jewish_25_64	7.66667	0.185855
age_median	8.00000	0.270555
industry_gi_25_64	12.66667	0.085941
eu_rest_25_64	16.66667	0.063826
indian_25_64	18.66667	0.096454
married_25_64	21.00000	0.107575

```
# arrange by mean score and display top 9
feature_rank %>% select(feature, mean_rank, mean_overall) %>%
```

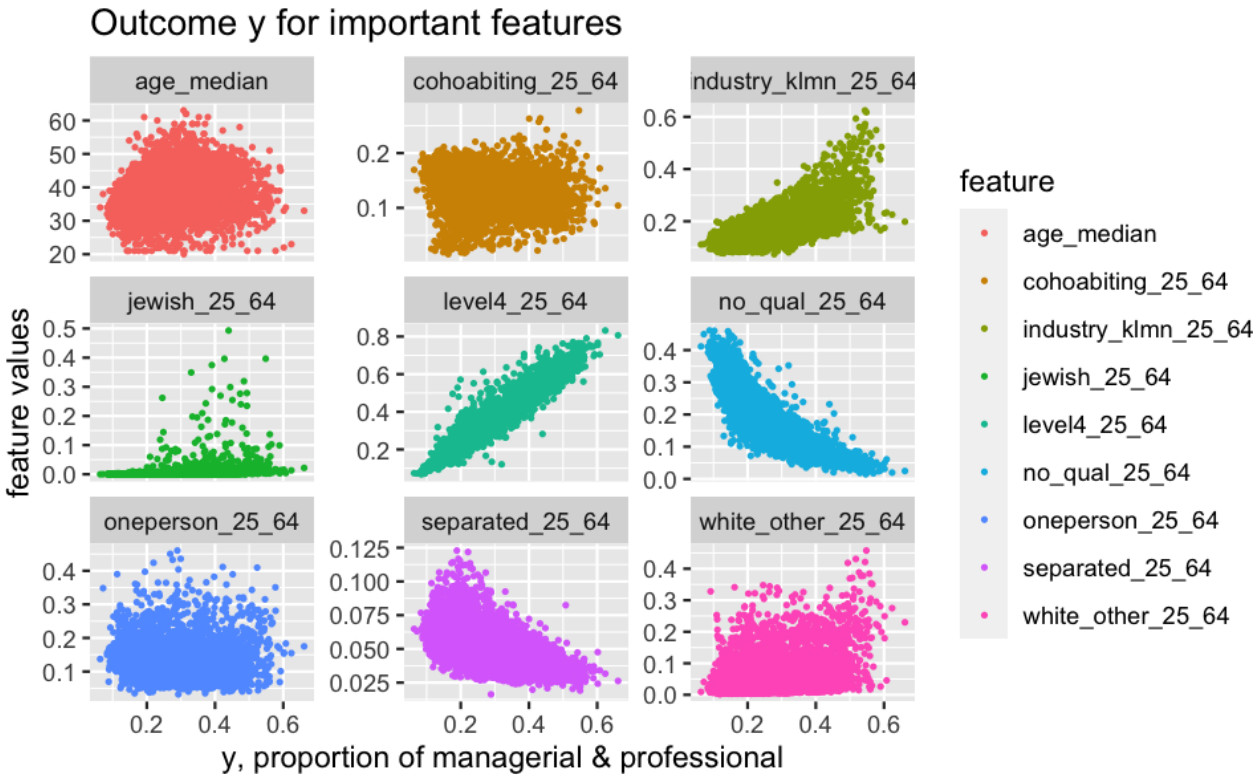
```
  arrange(desc(mean_overall)) %>% head(9) %>%
  knitr::kable(caption="Top 9 most important features by average of the model reported 'overall value'")
```

Table 35: Top 9 most important features by average of the model reported 'overall value'

feature	mean_rank	mean_overall
level4_25_64	1.00000	1.000000
age_median	8.00000	0.270555
oneperson_25_64	26.00000	0.235209
industry_klmn_25_64	6.66667	0.194686
jewish_25_64	7.66667	0.185855
cohoabiting_25_64	32.66667	0.184813
no_qual_25_64	5.33333	0.177216
separated_25_64	24.33333	0.137208
white_other_25_64	27.66667	0.108252

Plotting these features against the outcome y, the managerial and professional occupation ratio, you can see that there are varying distributions.

```
# plotting the top 9
feature_top9 <- feature_rank %>%
  select(feature, mean_rank, mean_overall) %>%
  arrange(desc(mean_overall)) %>% head(9) %>%
  pull(feature)
#create the plot
train_set_final %>%
  #pivot to a long version with a row per feature/value
  pivot_longer(cols = !"y",
               names_to = "feature",
               values_to = "value") %>%
  filter(feature %in% feature_top9) %>%
  ggplot(aes(y, value, col=feature))  +
  geom_point(size=0.5) +
  facet_wrap(. ~feature,  scales = "free_y") +
  ggtitle("Outcome y for important features") +
  xlab("y, proportion of managerial & professional") +
  ylab("feature values")
```



The following features are positively correlated:

- level4_25_64
- industry_klmn_25_64
- white_other_25_64
- jewish_25_64
- age_median

With details below extracted from the census information (http://www.ons.gov.uk/ons/guide-method/census/2011/census-data/2011-census-user-guide/information-by-variable/part-4--derived-variables.pdf).

- Level 4+ qualification: Degree (for example BA, BSc), Higher Degree (for example MA, PhD, PGCE), NVQ Level 45, HNC, HND, RSA Higher Diploma, BTEC Higher level, Foundation degree (NI), Professional qualifications (for example teaching, nursing, accountancy)
- Industry K - Financial and insurance activities
- Industry L - Real estate activities
- Industry M - Professional, scientific and technical activities
- Industry N - Administrative and support service activities
- white_other_25_64 - refers to all "white" ethnicities that are not "White: English/Welsh/Scottish/Northern Irish/British"

The following are negatively correlated:

- separated_25_64
- cohoabiting_25_64
- no_qual_25_64

```
#looking at correlation with y
data.frame(cor(train_set_final)) %>%
  select(y) %>%
  rownames_to_column(var="feature") %>%
  filter(feature %in% feature_top9) %>%
```

```
  arrange(desc(abs(y))) %>%
  knitr::kable(caption="Correlation with the outcome y, for MSOA")
```

Table 36: Correlation with the outcome y, for MSOA

feature	y
level4_25_64	0.953225
no_qual_25_64	-0.828072
industry_klmn_25_64	0.698728
separated_25_64	-0.535358
white_other_25_64	0.317770
jewish_25_64	0.225927
cohoabiting_25_64	-0.209523
age_median	0.190213
oneperson_25_64	-0.078622

4.3.2 Identifying the right number of features

Now we have the ranking order of priority, what is the optimal number for the models that can not handle many features?

In a similar way to calculating the performance of the models with the existing training sets (smaller, small, full etc.), the RMSE can be calculated for various numbers of the top features. This was carried out for the leading models, GLM, SVM Linear, SVM Radial.

The following function to create lists of the top n features based on the rankings, and uses the list to create the training sets.

```
#creating a function to choose the top n features, based on overall score
topfeature_overall <- function(topn, traindata){
  feature_n <- feature_rank %>%
    select(feature, mean_rank, mean_overall) %>%
    arrange(desc(mean_overall)) %>% head(topn) %>%
    pull(feature)
  train_topn <- traindata %>%
    select(y, all_of(feature_n))
  return(train_topn)
}
#creating a function to choose the top n features, based on rank
topfeature_rank <- function(topn, traindata){
  feature_n <- feature_rank %>%
    select(feature, mean_rank, mean_overall) %>%
    arrange(mean_rank) %>% head(topn) %>%
    pull(feature)
  train_topn <- traindata %>%
    select(y, all_of(feature_n))
  return(train_topn)
}

# load("rda/feature_rank.rda")
#create new training sets
train_top10 <- topfeature_overall(10, train_set_final)
# names(train_top10)
train_top10_rank <- topfeature_rank(10, train_set_final)
```

```
# names(train_top10_rank)
train_top15 <- topfeature_overall(15, train_set_final)
train_top20 <- topfeature_overall(20, train_set_final)
train_top30 <- topfeature_overall(30, train_set_final)
```

Using this a quick check shows that even though the importance of some of the features is low, they are making a significant improvement to the rmse, say for going from 15 to 20 features has a big drop. Also, it shows that prioritising by overall mean score is better than the mean rank, as expected.

```
# running on the GLM model
# with the top15 training set
result_glm_train_top15 <- results_train_method(train_top15, "glm")
```

```
## [1] "running train_\"glm\"_train_top15"
## [1] "rmse result is 0.0190818348261682"
## [1] "creating result list for \"glm\" - train_top15"
```

```
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_top15$results, 1))

# with the top20 training set
result_glm_train_top20 <- results_train_method(train_top20, "glm")
```

```
## [1] "running train_\"glm\"_train_top20"
## [1] "rmse result is 0.018539603602343"
## [1] "creating result list for \"glm\" - train_top20"
```

```
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_top20$results, 1))

# with the top10 training set
result_glm_train_top10 <- results_train_method(train_top10, "glm")
```

```
## [1] "running train_\"glm\"_train_top10"
## [1] "rmse result is 0.0191717079868583"
## [1] "creating result list for \"glm\" - train_top10"
```

```
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_top10$results, 1))

# with the top10 training set on rank
result_glm_train_top10_rank <- results_train_method(train_top10_rank, "glm")
```

```
## [1] "running train_\"glm\"_train_top10_rank"
## [1] "rmse result is 0.0203812503650401"
## [1] "creating result list for \"glm\" - train_top10_rank"
```

```
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_top10_rank$results, 1))
rmse_results %>%
  filter(str_detect(method, '"glm"') ) %>%
  arrange(rmse) %>%
  knitr::kable(caption="GLM model RMSE results with different feature sets")
```

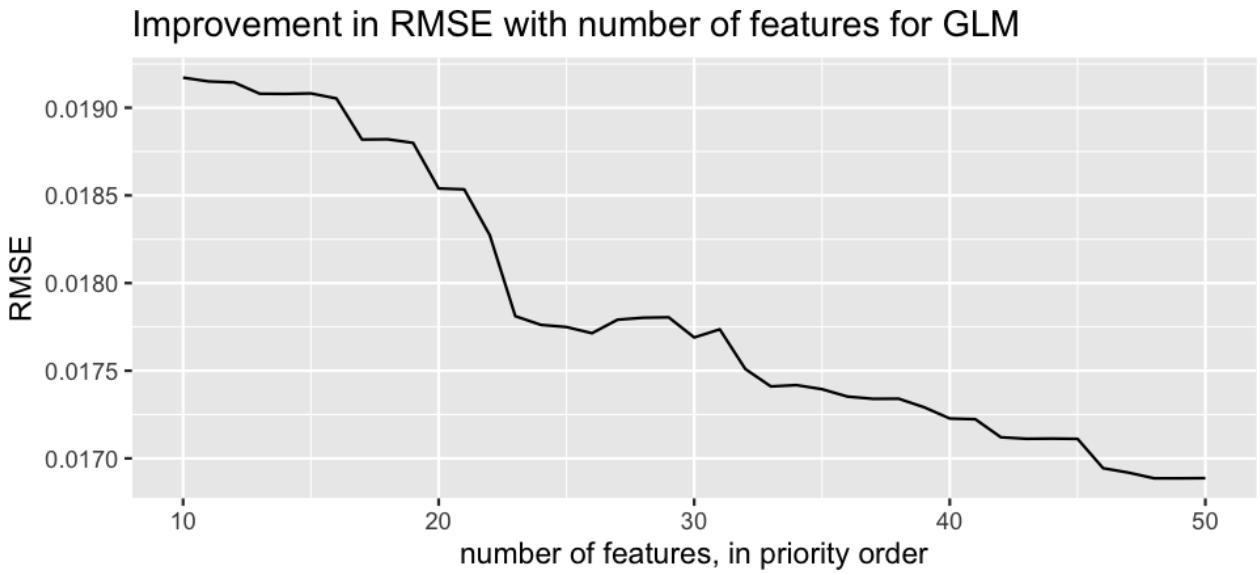
Table 37: GLM model RMSE results with different feature sets

method	rmse
"glm" - train_set_final	0.016681
"glm" - train_set_final	0.016681
"glm" - train_final_nocat	0.016900
"glm" - train_top20	0.018540
"glm" - train_top15	0.019082
"glm" - train_top10	0.019172
"glm" - train_top10_rank	0.020381
"glm" - train_small	0.024450
"glm" - train_smaller	0.039643

The model was run for the top 10 to top 50 features, with the respective RMSEs for the GLM model, and plotted you can see that there is a steady decrease in RMSE, with a big drop at 23. It is interesting to see that the overall trend is down, but adding a single feature can increase the RMSE - sometimes having more information makes the estimate/prediction worse.

```
# running with features from top 10 to top 50 say
#set up the RMSE table
load("rda/feature_rank.rda")
#apply to all of the top 10 to top 50 features
glm_top10_top50_tmp <- sapply(10:50, function(n){
  print(paste("training with", n, "features"))
  train_topn <- topfeature_overall(n)
  print("running results function")
  result_glm_train_topn <- results_train_method(train_topn, "glm")
  result_glm_train_topn$results[,2]
})
#extract the rmse values and put in a table
glm_top10_top50 <- t(data.frame(glm_top10_top50_tmp)) %>%
  cbind(feature_no = 10:50)
colnames(glm_top10_top50) <- c("rmse", "feature_no")
```

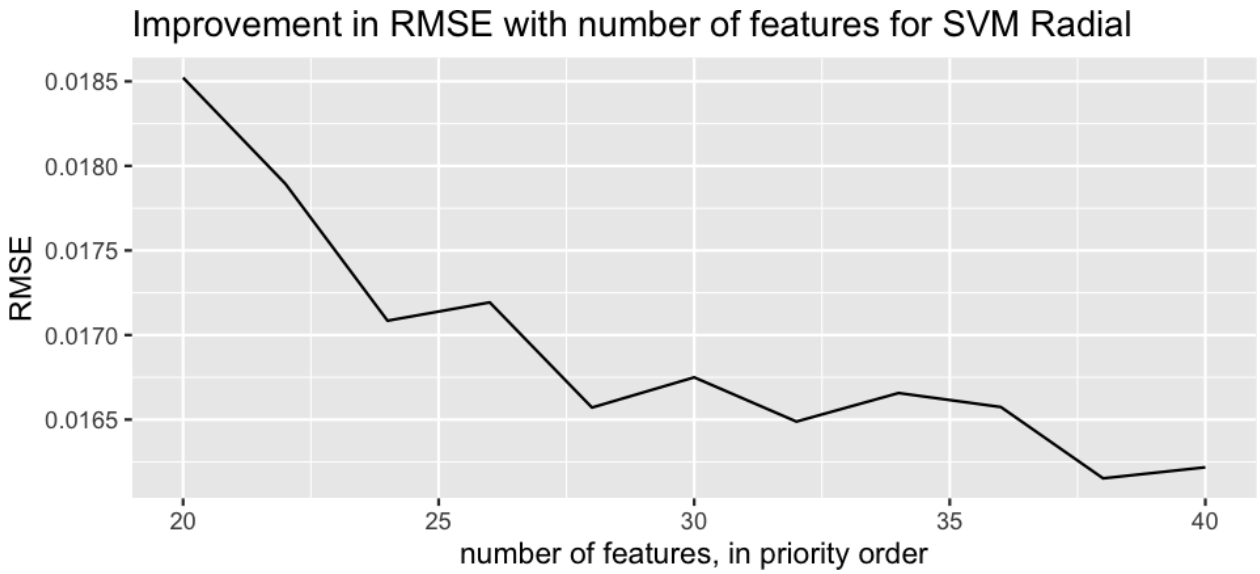
```
# load the data
load("rda/glm_top10_top50.rda")
#plot the values
data.frame(glm_top10_top50) %>%
  ggplot(aes(x=feature_no, y=rmse)) +
  geom_line() +
  ggtitle("Improvement in RMSE with number of features for GLM") +
  xlab("number of features, in priority order") +
  ylab("RMSE")
```



A similar process was carried out with the SVM Radial algorithm. As it is much slower, only the top 20 to 40 features were modelled, incrementing by 2. This also drops quickly after 20, but flattening a bit around the 28 number.

```
#not running here due to the time taken
# similar with svm radial
#apply to the top 20 to top 40 features
svmradial_top20_top40_tmp <- sapply(seq(20, 40, 2), function(n){
  print(paste("training with", n, "features"))
  train_topn <- topfeature_overall(n)
  print("running results function")
  result_svmradial_train_topn <- results_train_method(train_topn, "svmRadial")
  result_svmradial_train_topn$results[,2]
})
#extract the rmse values and put in a table
svmradial_top20_top40_tmp
svmradial_top20_top40 <- t(data.frame(svmradial_top20_top40)) %>%
  cbind(feature_no = seq(20, 40, 2))
colnames(svmradial_top20_top40) <- c("rmse", "feature_no")
svmradial_top20_top40
```

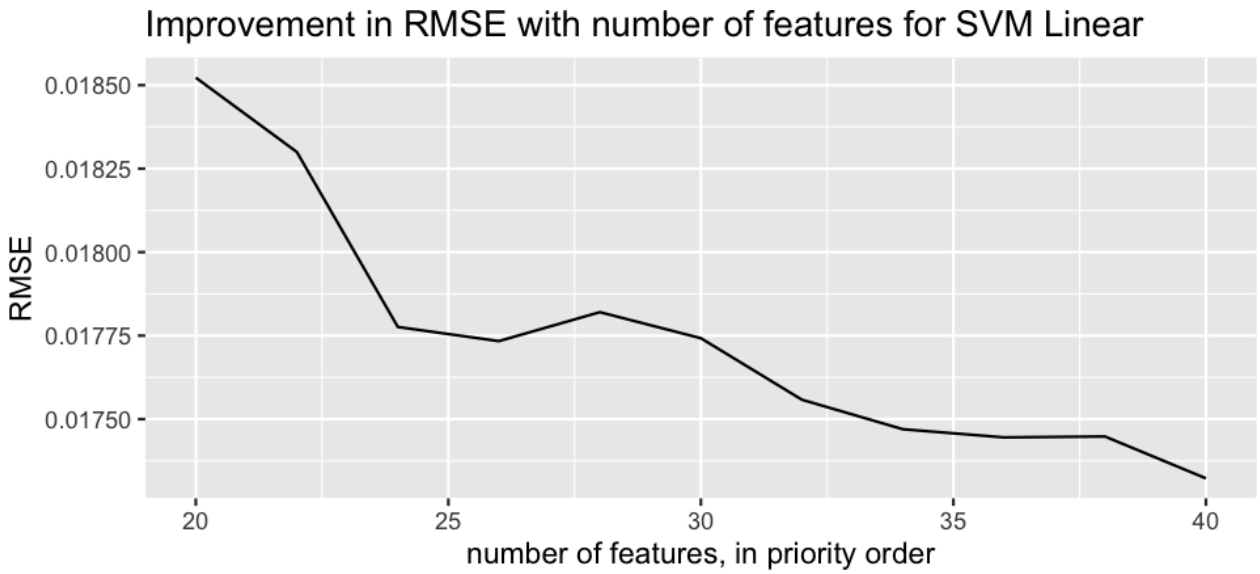
```
#load the calculation from file
load("rda/svmradial_top20_top40.rda")
#plot the values
data.frame(svmradial_top20_top40) %>%
  ggplot(aes(x=feature_no, y=rmse)) +
  geom_line() +
  ggtitle("Improvement in RMSE with number of features for SVM Radial") +
  xlab("number of features, in priority order") +
  ylab("RMSE")
```



Similarly for the SVM linear algorithm, focusing on the top 20 to 40 features, incrementing by 2 each time. This seems to have a drop around the 24 and then flattish until dropping again around 32/33.

```
# not running as it takes a while
#apply to the top 20 to top 40 features
svm_top20_top40_tmp <- sapply(seq(20, 40, 2), function(n){
  print(paste("training with", n, "features"))
  train_topn <- topfeature_overall(n)
  print("running results function")
  result_svm_train_topn <- results_train_method(train_topn, "svmLinear")
  result_svm_train_topn$results[,2]
})
#extract the rmse values and put in a table
svm_top20_top40_tmp
svm_top20_top40 <- t(data.frame(svm_top20_top40_tmp)) %>%
  cbind(feature_no = seq(20, 40, 2))
colnames(svm_top20_top40) <- c("rmse", "feature_no")
svm_top10_top40
```

```
#load the calculation from file
load("rda/svm_top20_top40.rda")
#plot the values
data.frame(svm_top20_top40) %>%
  ggplot(aes(x=feature_no, y=rmse)) +
  geom_line() +
  ggtitle("Improvement in RMSE with number of features for SVM Linear") +
  xlab("number of features, in priority order") +
  ylab("RMSE")
```



This indicates that the algorithms should use at least the top 24 features, as there is a bit drop just beforehand, and potentially around 28 to have the biggest impact on the overall RMSE.

4.3.3 Training GAM and GAM loess with fewer features

The GAM and GAM Loess can be run with an optimised smaller set of features. As the models did not work with the "small" set of 35 features, initially the top 25 features were used.

For the GAM model, this achieves an RMSE of 0.0161365, better than all but the SVM Radial model.

Trying with the top 28 features, and this was better still, achieving an RMSE of 0.015935. It does include a feature importance, and here level4 qualification has infinite importance!

```
# running on GAM with 25 features
train_top25 <- topfeature_overall(25)
# with the train_top25 training set
result_gam_train_top25 <- results_train_method(train_top25, "gam")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_gam_train_top25$results, 1))
rmse_results %>% knitr::kable()
# checking the variable importance
importance <- varImp(result_gam_train_top25$train, scale=FALSE)
plot(importance, 20)
# tidy
save(result_gam_train_top25, file="rda/result_gam_train_top25.rda")
rm(result_gam_train_top25)

# running on GAM with 28 features
train_top28 <- topfeature_overall(28)
# with the train_top28 training set
result_gam_train_top28 <- results_train_method(train_top28, "gam")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_gam_train_top28$results, 1))
rmse_results %>% knitr::kable()

#checking the overall performance
rmse_results %>%
  arrange(rmse)%>%
```

```
  head(10) %>%
  knitr::kable()
```

And next, doing the same with the GAM loess model, using the top 28 features, producing good results, with an RMSE of 0.016959, which puts it just behind the SVM model in 5th place.

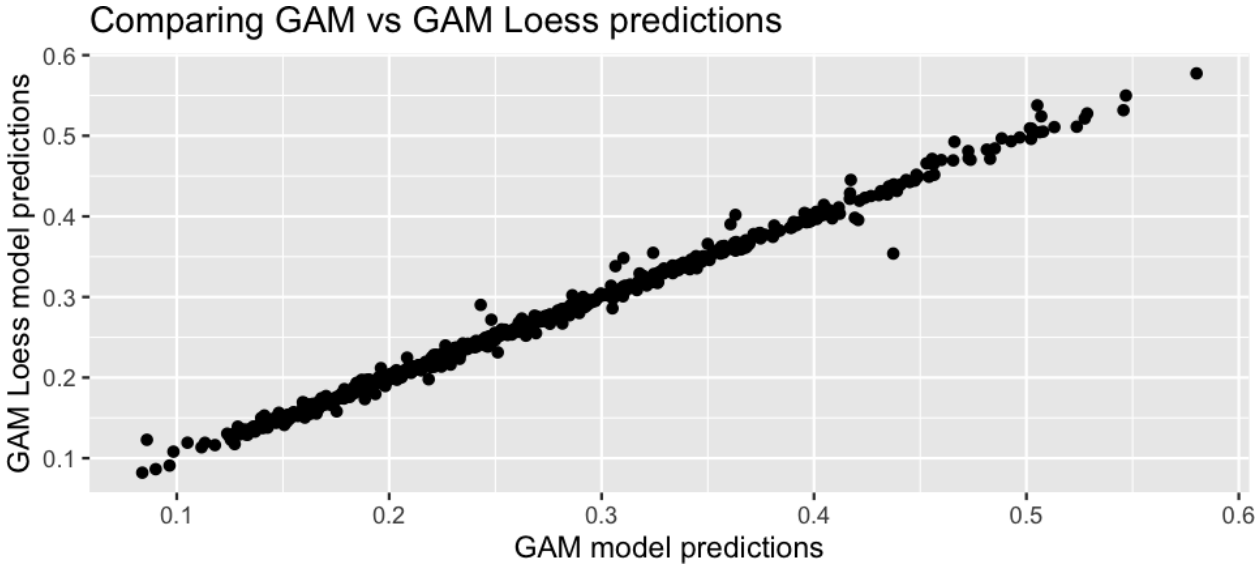
```
# running on GAMloess with 28 features
# with the train_top28 training set
result_gamloess_train_top28 <- results_train_method(train_top28, "gamLoess")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_gamloess_train_top28$results, 1))
rmse_results %>% knitr::kable()
```

Table 38: Leading models for the MSOA

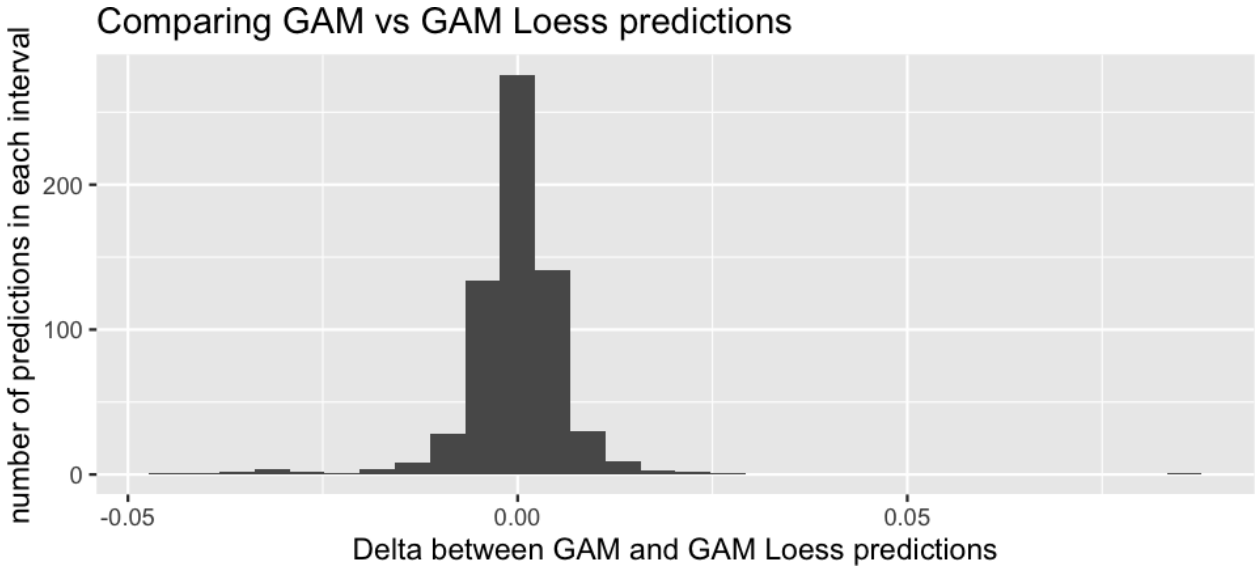
method	rmse
“svmRadial” - train_set_final	0.0145255
“gam” - train_top28	0.0159350
“glm” - train_set_final	0.0166809
“svmLinear” - train_set_final	0.0167375
“gamLoess” - train_top28	0.0169595

Although GAM and GAM Loess have similar source models using the gam package, their $\hat{y}$ estimates are quite different, so they can both be a useful addition to the ensemble.

```
load("rda/result_gamLoess_train_top28.rda")
load("rda/result_gam_train_top28.rda")
# comparing gam and gamloess
y_hat_gam <- result_gam_train_top28$y_hat
y_hat_gamLoess <- result_gamLoess_train_top28$y_hat
data.frame(y_hat_gam, y_hat_gamLoess) %>%
  ggplot(aes(y_hat_gam, y_hat_gamLoess)) +
  geom_point() +
  ggtitle("Comparing GAM vs GAM Loess predictions") +
  xlab("GAM model predictions") +
  ylab("GAM Loess model predictions")
```




```
data.frame(y_hat_gam, y_hat_gamLoess) %>%
  mutate(delta = (y_hat_gam-y_hat_gamLoess)) %>%
  ggplot(aes(delta)) +
  geom_histogram(bins = 30) +
  ggtitle("Comparing GAM vs GAM Loess predictions") +
  xlab("Delta between GAM and GAM Loess predictions") +
  ylab("number of predictions in each interval")
```



The smaller set was also used with th KNN model, but although it was the best it had produced it is still some distance behind the leading models.

```
# running on knn with 28 features, less age and area
train_top28_knn <- train_top28 %>%
  select(-age_median, -area_code)
# with the train_top28 training set
result_knn_train_top28_knn <- results_train_method(train_top28_knn, "knn")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_knn_train_top28_knn$results, 1))
```

```
rmse_results %>% knitr::kable()
```

4.3.4 Final ensemble model

Now looking at the top models, the best are SVM Radial, GAM, GLM and SVM Linear, with GAM Loess nearly as good.

```
# updating the ensemble with the new models
# Identify the leading models, the new top 5
rmse_results %>%
  filter(str_detect(method, 'Ensemble', negate=TRUE) ) %>%
  arrange(rmse) %>% head(15) %>%
  knitr::kable(caption="Leading individual models")
```

Table 39: Leading individual models

method	rmse
“svmRadial” - train_set_final	0.014525
“svmRadial” - train_final_nocat	0.015748
“gam” - train_top28	0.015935
“bayesglm” - train_set_final	0.016680
“glm” - train_set_final	0.016681
“glm” - train_set_final	0.016681
“lasso” - train_set_final	0.016681
“svmLinear” - train_set_final	0.016737
“svmLinear2” - train_set_final	0.016748
“glm” - train_final_nocat	0.016900
“gamLoess” - train_top28	0.016959
“gbm” - train_set_final	0.017467
“ranger” - train_set_final	0.018276
“glm” - train_top20	0.018540
“glm” - train_top15	0.019082

Creating an ensemble to include the GAM and GAM Loess just created, with five models (GLM, SVM Linear, SVM Radial, GAM, GAM Loess), averaging the prediction, the result is an RMSE of 0.0149383, vs the SVM Radial model of 0.0145255, and only slightly better than the last ensemble of three (GLM, SVM Linear, SVM Radial), which is maybe because the strong performance of the GAM is counterbalanced by the relatively poor GAM Loess.

```
#loading the separate model information
load("rda/result_glm_train_set_final.rda")
load("rda/result_svm_train_set_final.rda")
load("rda/result_svmradial_train_set_final.rda")
load("rda/result_gam_train_top28.rda")
load("rda/result_gamLoess_train_top28.rda")

# an ensemble of the best five models
y_hat_ens_table3 <-
  data.frame(number = 1:length(result_glm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_glm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_svm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_svmradial_train_set_final$y_hat)) %>%
  cbind(data.frame(result_gam_train_top28$y_hat)) %>%
```

```
  cbind(data.frame(result_gamLoess_train_top28$y_hat)) %>%
  select(-number) %>%
  #calculate average of the y_hats
  mutate(y_hat_ave = rowMeans(.))
# head(y_hat_ens_table3)
#test against the actual values
rmse_ens <- rmse(test_set_final$y, y_hat_ens_table3$y_hat_ave)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Ensemble3 glm, svm, svmRadial, gam, gamLoess",
                                 rmse = rmse_ens))
rmse_results %>%
  arrange(rmse) %>% head(10) %>%
  knitr::kable(caption="Leading models including the ensembles, with MSOA")
```

Table 40: Leading models including the ensembles, with MSOA

method	rmse
"svmRadial" - train_set_final	0.014525
Ensemble3 glm, svm, svmRadial, gam, gamLoess	0.014938
Ensemble2 glm, svm, svmRadial	0.014962
Ensemble glm, gbm, svm, svmRadial, ranger	0.015242
"gam" - train_top28	0.015935
"glm" - train_set_final	0.016681

Creating an ensemble with the best three models, (GLM, SVM Radial, GAM), the result is an RMSE of 0.014395, vs the SVM Radial model of 0.0145255, so improving and now the best result overall.

```
# an ensemble of the best three models
y_hat_ens_table4 <-
  data.frame(number = 1:length(result_glm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_glm_train_set_final$y_hat)) %>%
  cbind(data.frame(result_svmradial_train_set_final$y_hat)) %>%
  cbind(data.frame(result_gam_train_top28$y_hat)) %>%
  select(-number) %>%
  #calculate average of the y_hats
  mutate(y_hat_ave = rowMeans(.))
# head(y_hat_ens_table4)
#test against the actual values
rmse_ens <- rmse(test_set_final$y, y_hat_ens_table4$y_hat_ave)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Ensemble4 svmRadial, gam, glm",
                                 rmse = rmse_ens))
rmse_results %>%
  arrange(rmse) %>% head(5) %>% knitr::kable()
```

Finally, creating an ensemble with the best two models, (SVM Radial and GAM), averaging the prediction, the result is an RMSE of 0.013991 which is a significant improvement over the previous ensemble and over the SVM Radial and GAM separately. This is the final model for the best result for this project.

```
#ensemble of 2
y_hat_ens_table5 <-
  data.frame(svmradial_y_hat = result_svmradial_train_set_final$y_hat,
             gam_y_hat = result_gam_train_top28$y_hat) %>%
```

```
  #calculate average of the y_hats
  mutate(y_hat_ave = rowMeans(.))
#test against the actual values
rmse_ens <- rmse(test_set_final$y, y_hat_ens_table5$y_hat_ave)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Ensemble5 svmRadial, gam",
                                 rmse = rmse_ens))
rmse_results %>%
  arrange(rmse) %>% head(7) %>%
  knitr::kable(caption="Final ensemple model with MSOA")
```

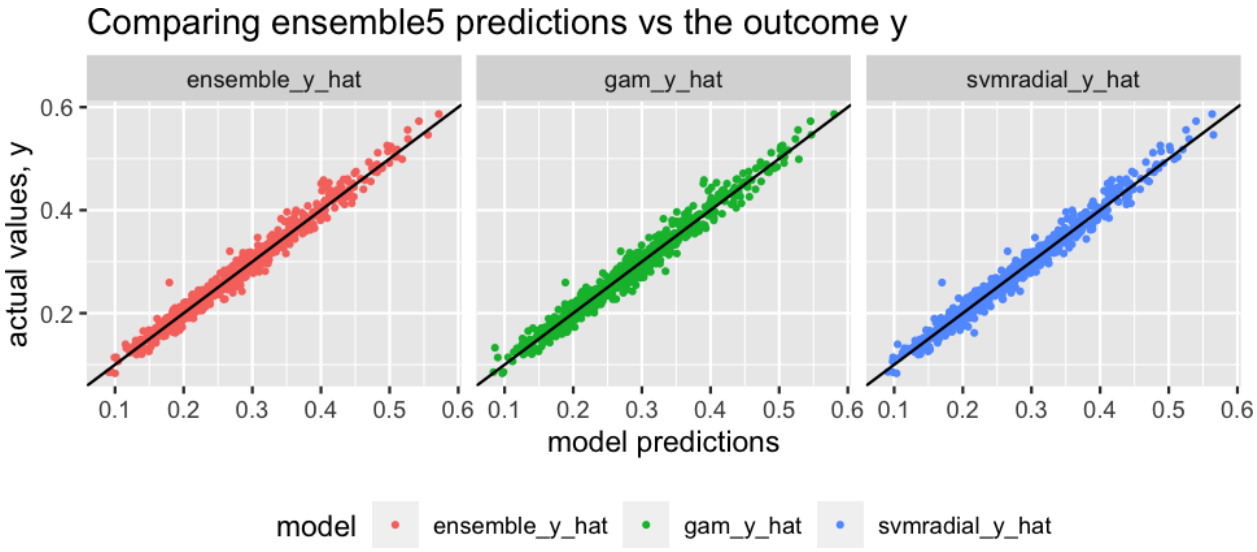
Table 41: Final ensemple model with MSOA

method	rmse
Ensemble5 svmRadial, gam	0.013991
Ensemble5 svmRadial, gam	0.013991
Ensemble5 svmRadial, gam	0.013991
Ensemble4 svmRadial, gam, glm	0.014395
"svmRadial" - train_set_final	0.014525
Ensemble3 glm, svm, svmRadial, gam, gamLoess	0.014938
Ensemble3 glm, svm, svmRadial, gam, gamLoess	0.014938

4.4 Analysis of errors

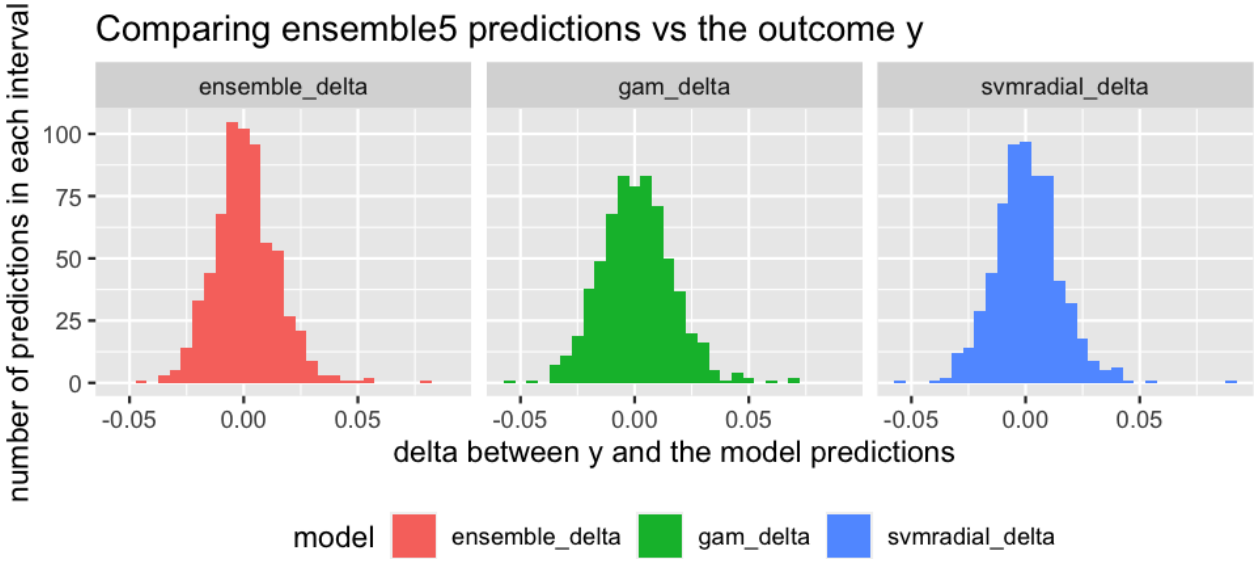
From the graphs of SVM Radial, GAM and their ensemble, it is hard to draw out too much conclusion. SVM Radial seems a bit tighter in general, but with a few outliers which are worse than the GAM model.

```
# plot of y against y_hat for all three
y_hat_ens_table5 %>%
  cbind(y = test_set_final$y) %>%
  rename(ensemble_y_hat=y_hat_ave) %>%
  pivot_longer(cols=contains("y_hat"), names_to="model", values_to="y_hat") %>%
  ggplot(aes(x=y_hat, y=y, col=model)) +
  geom_point(size=0.7) +
  geom_abline(slope = 1, intercept = 0, col="black") +
  facet_grid(. ~model) +
  ggtitle("Comparing ensemble5 predictions vs the outcome y") +
  xlab("model predictions") +
  ylab("actual values, y") +
  theme(legend.position = "bottom")
```



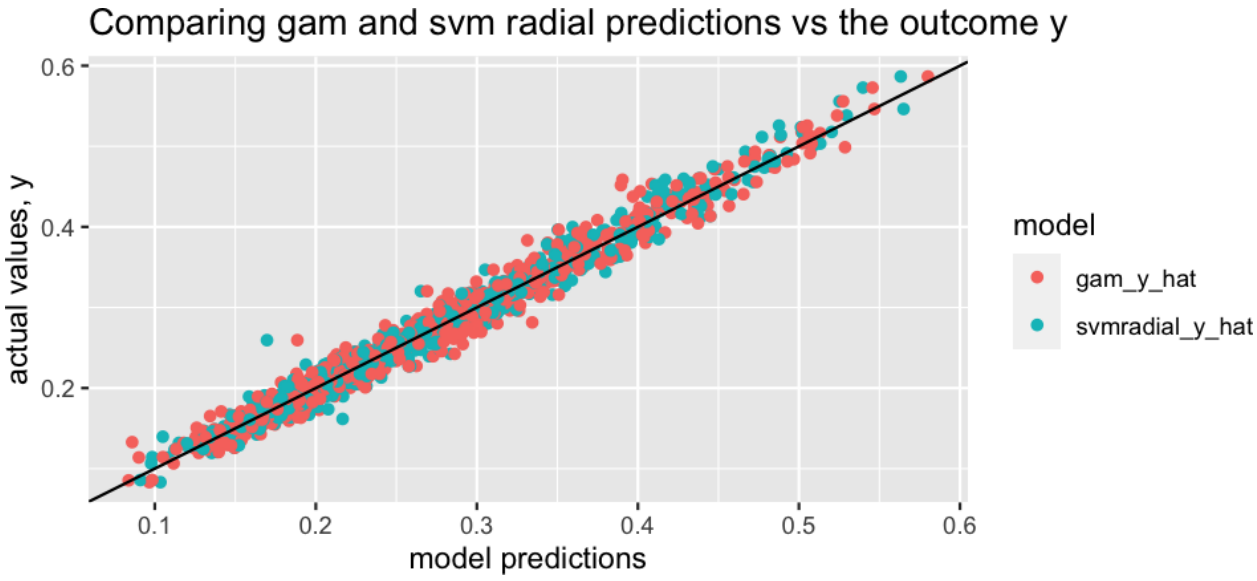
Looking at the histograms, it tells a similar story, which explains why the ensemble is better, as GAM is helping improve the outlier results from SVM Radial.

```
# histogram of deltas for all three
y_hat_ens_table5 %>%
  cbind(y = test_set_final$y) %>%
  mutate(svmradial_delta = (y-svmradial_y_hat)) %>%
  mutate(gam_delta = (y-gam_y_hat)) %>%
  mutate(ensemble_delta = (y-y_hat_ave)) %>%
  select(-svmradial_y_hat, -gam_y_hat, -y_hat_ave) %>%
  pivot_longer(cols=contains("delta"), names_to="model", values_to="delta") %>%
  ggplot(aes(delta, fill=model)) +
  geom_histogram(bins = 30) +
  facet_grid(. ~model) +
  ggtitle("Comparing ensemble5 predictions vs the outcome y") +
  xlab("delta between y and the model predictions") +
  ylab("number of predictions in each interval") +
  theme(legend.position = "bottom")
```



And this graph illustrates the same, with the colours representing the models, and GAM seems to surround SVM radial, apart from the outliers (large difference, and high y value).

```
# comparing the svmradial and gam models
y_hat_ens_table5 %>%
  cbind(y = test_set_final$y) %>%
  select(-y_hat_ave) %>%
  pivot_longer(cols=contains("y_hat"), names_to="model", values_to="y_hat") %>%
  ggplot(aes(x=y_hat, y=y, col=model)) +
  geom_point()  +
  geom_abline(slope = 1, intercept = 0, col="black") +
  ggtitle("Comparing gam and svm radial predictions vs the outcome y") +
  xlab("model predictions") +
  ylab("actual values, y")
```



5 Final model for MSOA

This section covers the calculation of the final results for the MSOA, using the main/validation set.

The final model is an ensemble of two algorithms, the SVM Radial, using all of the features and GAM using the top 28 features, where the features are selected based on the normalised importance scores from the GLM, GBM and GLM boost models.

Both of these require quite a lot of computation, with GAM particularly slow. This may be a problem with larger data sets, say, using the full SOA data, and may lead to needing to reduce the number of features further.

The final model developed with the train/test data will be used to train the final predictions.

Taking the main/validation data put together in the earlier section, and the geo_lookup table developed from the census information, cleanse data function was used to create the versions of the main and validation sets for training and testing.

```
#run on the test set and train set
# load the files
load("rda/geo_lookup.rda")
load("rda/main.rda")
load("rda/validation.rda")
#cleanse the data, add the geo info
main_clean <- data_cleanse(main, "msoa")
```

```
## [1] "working on msoa"
## [1] "msoa geolookup"
```

```
validation_clean <- data_cleanse(validation, "msoa")
```

```
## [1] "working on msoa"
## [1] "msoa geolookup"
```

Now training and testing on the data sets, calculating mean as the the baseline, GLM as a comparison, and SVM Radial, all at default. The mean is slightly better (0.091347 vs 0.096470), GLM worse (0.016939 vs 0.016681) and SVM Radial significantly worse (0.015654 vs 0.014525) than the performance using the train/test data.

```
# create baseline
mu_hat <- mean(main_clean$y)
rmse_ave <- rmse(validation_clean$y, mu_hat)
rmse_results_validation <- tibble(method = "Mean on validation", rmse = rmse_ave)

# run on glm radial
result_glm_main <-
  results_train_method(main_clean, validation_clean, "glm")
# extract the rmse from the results
rmse_results_validation <- bind_rows(rmse_results_validation,
                                     tail(result_glm_main$results, 1))

# run on svm radial
# uses package kernlab
if (!require('kernlab')) install.packages('kernlab'); library('kernlab')
# with the full train set
result_svmradial_main <-
  results_train_method(main_clean, validation_clean, "svmRadial")
# extract the rmse from the results
rmse_results_validation <- bind_rows(rmse_results_validation,
```

```
                                    tail(result_svmradial_main$results, 1))
rmse_results_validation %>% knitr::kable()
```

Table 42: GLM and SVM Radial model RMSE results on the main/validation data for MSOA

method	rmse
Mean on validation	0.091347
"glm" - main_clean	0.016939
"svmRadial" - main_clean	0.015654

Next, load the top28 features developed using the train/test data on GLM, GBM and GLM boost, and then run on the GAM model. Note, the top28 was developed earlier with information from the test as well as train data, but has not been developed with any data from the validation set.

The GAM result with 28 features is almost identical to the earlier RMSE with the train/ test data (0.015931 vs 0.015935), meaning that in this case, the performance of the GAM 28 is close to that of the SVM Radial.

```
# run on gam
# load top 28 features with the GLM, GBM and GLM boost models on the train/test data
load("rda/feature_rank.rda")
# put together the prioritised main data set subset
main_top28 <- topfeature_overall(28, main_clean)
# train the model with the train_top28 training set
result_gam_main_top28 <-
  results_train_method(main_top28, validation_clean, "gam")
#tmp
result_gam_main_top28 <- result_gam_train_top28
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results, tail(result_gam_main_top28$results, 1))
rmse_results %>% knitr::kable()
```

Running the final ensemble of SVM Radial and GAM on top 28, the result is a big step in accuracy, which is also closer to the results with the train/test set, (0.014253 vs 0.013991). Overall the model has performed well on the main/validation set.

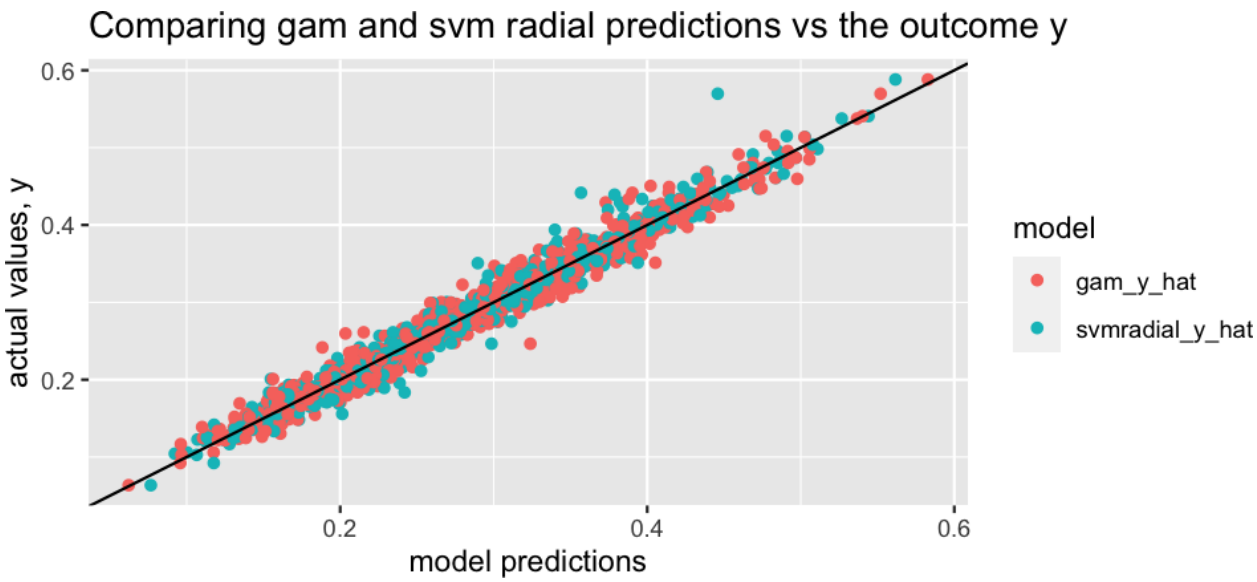
```
# ensemble of 2
load("rda/result_gam_main_top28.rda")
load("rda/result_svmradial_main.rda")
load("rda/rmse_results_validation.rda")
y_hat_ens_main <-
  data.frame(svmradial_y_hat = result_svmradial_main$y_hat,
             gam_y_hat = result_gam_main_top28$y_hat) %>%
  #calculate average of the y_hats
  mutate(y_hat_ave = rowMeans(.))
#test against the actual values
rmse_ens <- rmse(validation$y, y_hat_ens_main$y_hat_ave)
rmse_results_validation <- bind_rows(rmse_results_validation,
                          tibble(method="Ensemble svmRadial, gam28",
                                 rmse = rmse_ens))
rmse_results_validation %>%
  knitr::kable(caption="RMSE results with the main/validation data set on MSOA")
```

Table 43: RMSE results with the main/validation data set on MSOA

method	rmse
Mean on validation	0.091347
“glm” - main_clean	0.016939
“svmRadial” - main_clean	0.015654
“gam” - train_top28	0.015931
Ensemble svmRadial, gam28	0.014253
Ensemble svmRadial, gam28	0.014253

Comparing the SVM radial and GAM 28 models, once again there is a significant outlier for the SVM Radial which the GAM model will average out. This single outlier could be the reason the model is much worse this time around.

```
# comparing the svmradial and gam models
y_hat_ens_main %>%
  cbind(y = validation$y) %>%
  select(-y_hat_ave) %>%
  pivot_longer(cols=contains("y_hat"), names_to="model", values_to="y_hat") %>%
  ggplot(aes(x=y_hat, y=y, col=model)) +
  geom_point()  +
  geom_abline(slope = 1, intercept = 0, col="black") +
  ggtitle("Comparing gam and svm radial predictions vs the outcome y") +
  xlab("model predictions") +
  ylab("actual values, y")
```



6 Running the models with the SOA data

The analysis so far has been carried out using the MSOA data, around 7000 observations, nationwide. Now the methodology will be used with the SOA data, considerably more detailed and larger, to see how it behaves.

Rather than all 130,000 entries, two SOA areas were selected, London and Yorkshire and The Humber, as these will have some variety in the types of areas, roles and people. The London set had 25,053 observations, and the Yorkshire had 17,275, making 42,328 in total.

6.1 Data preparation and visualisation

6.1.1 Importing and creating the data set

In turn for London and Yorkshire, the code will download the csv data from Nomis, using the function created earlier, and then run the functions to extract the features and create the "data_set" object.

```
### downloading the SOA data
# choosing London and Yorkshire
# "2013265927TYPE299">output areas 2011 in London
# "2013265923TYPE299">output areas 2011 in Yorkshire and The Humber
#creating list of census tables to take
censusdata_list <- c("ks608ew", "ks102ew", "lc2101ew", "lc2103ew","lc6110ew",
                     "lc1109ew", "lc2107ew", "lc1101ew", "lc5102ew")

# London
# apply the list to download all of the data
download_censusdata("2013265927TYPE299", censusdata_list)
#import the data for London
data_set <- import_data("2013265927TYPE299")
#clean up
data_set_london <- data_set
rm(filename)
rm(geographytype)

# "2013265923TYPE299">output areas 2011 in Yorkshire and The Humber
geographytype <- "2013265923TYPE299" # output areas 2011 in Yorkshire and The Humber
# apply the list to download all of the data
download_censusdata("2013265923TYPE299", censusdata_list)
#import the data for Yorkshire
data_set <- import_data("2013265923TYPE299")
data_set_yorkshire <- data_set
# load("rda/data_set_2013265923TYPE299.rda")
```

These were combined in to a single data set (adding rows together) before creating the main/validation sets, and the train/test.

```
# load the yorkshire and humber data
load("rda/data_set_2013265923TYPE299.rda")
data_set_yorkshire <- data_set
# load the data for London
load("rda/data_set_2013265927TYPE299.rda")
data_set_london <- data_set
#combine in to one data set
data_set <- data_set_london %>%
  rbind(data_set_yorkshire)
```

```
#### create main and validation set
# load("rda/data_set_lon_york.rda")
# Validation set will be 10% of the data
set.seed(2011, sample.kind="Rounding")
test_index <- createDataPartition(y = data_set$y, times = 1, p = 0.1, list = FALSE)
main_soa <- data_set[-test_index,]
validation_soa <- data_set[test_index,]

#### create test and train set
```

```
# load("rda/main_soa.rda")
# load("rda/main_tidy.rda")
# test set will be 10% of the data
set.seed(2001, sample.kind="Rounding")
test_index <- createDataPartition(y = main_soa$y, times = 1, p = 0.1, list = FALSE)
train_soa <- main_soa[-test_index,]
test_soa <- main_soa[test_index,]
#check
# names(train_soa)
# sum(is.na(train_soa))

#data cleanse on the test set and train set
load("rda/geo_lookup.rda")
test_soa_final <- data_cleanse(test_soa, "soa")
```

```
## [1] "working on soa"
## [1] "soa geolookup"
```

```
train_soa_final <- data_cleanse(train_soa, "soa")
```

```
## [1] "working on soa"
## [1] "soa geolookup"
```

6.1.2 Initial baseline and visualisation

As before, an initial baseline RMSE based on the mean was created, which is 0.133419, much worse than the MSOA data which was 0.096470. This is as expected this, as there should be greated variability with the smaller areas. It could be that focusing on two more extreme areas in the UK could exacerbate this.

```
#### baseline rmse and rmse function
# load the test and train data
load("rda/test_soa_final.rda")
load("rda/train_soa_final.rda")
options(digits = 6)
# baseline RMSE with an average of all ratings
mu_hat <- mean(train_soa_final$y)
rmse_ave <- rmse(test_soa_final$y, mu_hat)
rmse_results_soa <- tibble(method = "Mean of all locations", rmse = rmse_ave)
rmse_results_soa %>% knitr::kable(caption="Initial baseline RMSE for SOA")
```

Table 44: Initial baseline RMSE for SOA

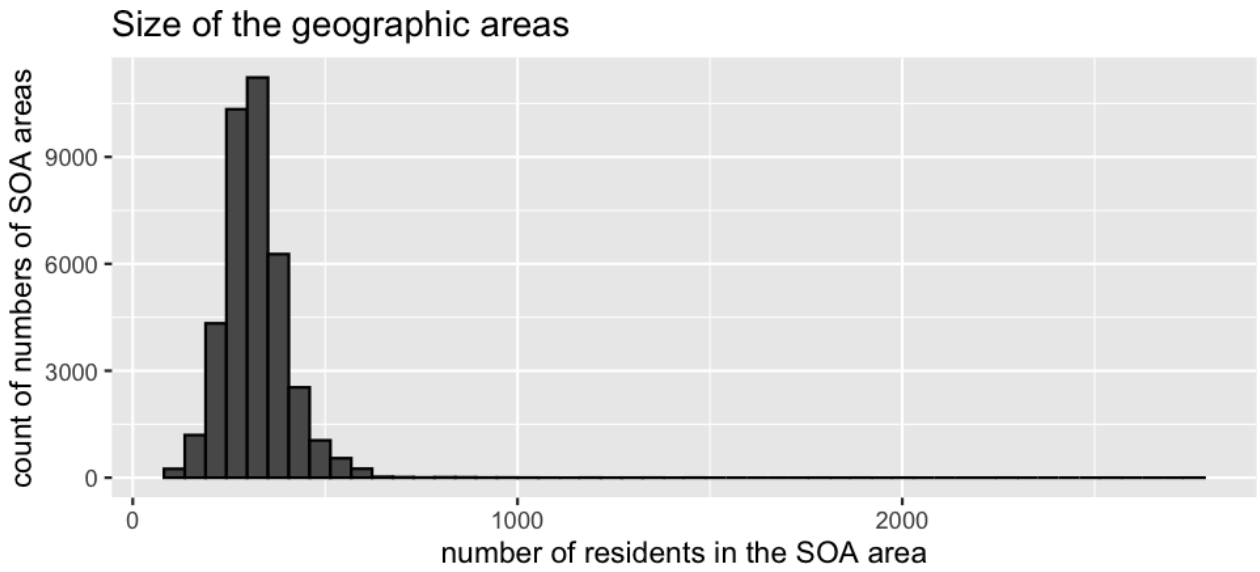
method	rmse
Mean of all locations	0.133419

Looking first at the sizes, the areas are roughly 25 times smaller than the MSOA, although the maximum is surprisingly large.

```
#graph the geographic areas
load("rda/main_soa.rda")
load("rda/main.rda")

main_soa %>%  ggplot(aes(all_residents))  +
  geom_histogram(bins = 50, color = "black") +
  ggtitle("Size of the geographic areas") +
```

```
  xlab("number of residents in the SOA area") +
  ylab("count of numbers of SOA areas")
```




```
#table of key data points of the sizes of the SOA areas
main_soa_sizes <- c(mean = mean(main_soa$all_residents),
                    sd=sd(main_soa$all_residents),
                    max = max(main_soa$all_residents),
                    min = min(main_soa$all_residents))
#table of key data points of the sizes of the MSOA areas
main_sizes <- c(mean = mean(main$all_residents),
                sd=sd(main$all_residents),
                max = max(main$all_residents),
                min = min(main$all_residents))
# displaying in a table for comparison
data.frame("Residents_in_SOA" = main_soa_sizes) %>%
  cbind(data.frame("Residents_in_MSOA" = main_sizes)) %>%
  knitr::kable(caption="Comparison of number of residents in SOA and MSOA")
```

Table 45: Comparison of number of residents in SOA and MSOA

	Residents_in_SOA	Residents_in_MSOA
mean	318.0385	7784.18
sd	85.7687	1605.54
max	2741.0000	16342.00
min	91.0000	2203.00

As expected, the histograms of the outcome y do show greater variability than the MSOA, with some areas with no one in the managerial and professional roles.

```
#graph the outcome y
tmp <- data.frame(SOA=train_soa_final$y) %>%
  pivot_longer(cols = SOA,
               names_to = "area_type",
               values_to = "y")
data.frame(MSOA=train_set_final$y) %>%
```

```
  pivot_longer(cols = MSOA,
               names_to = "area_type",
               values_to = "y") %>%
  rbind(tmp) %>%
  ggplot(aes(y))  +
  geom_histogram(bins = 30, color = "black") +
  facet_grid(area_type~., scales="free_y") +
  ggtitle("Distribution of the outcome, y for SOA and MSOA areas") +
  xlab("y, proportions of people in managerial or professional occupations") +
  ylab("count of numbers of areas")
```

And the table comparing demonstrates this, with a slightly higher mean for the smaller SOA areas, and larger standard deviation, with a much larger breadth of results.

```
#table of key data points of the outcome y
# summary info for the SOA data
train_soa_y <- c(mean = mean(train_soa_final$y),
                             sd=sd(train_soa_final$y),
                             max = max(train_soa_final$y),
                             min = min(train_soa_final$y))
# summary info for the MSOA data
train_msoa_y <- c(mean = mean(train_set_final$y),
                             sd=sd(train_set_final$y),
                             max = max(train_set_final$y),
                             min = min(train_set_final$y))
# displaying in a table for comparison
data.frame("y_SOA" = train_soa_y) %>%
  cbind(data.frame("y_MSOA" = train_msoa_y)) %>%
  knitr::kable(caption="Comparison of y, proportion of senior managers and professional occupations in S
```

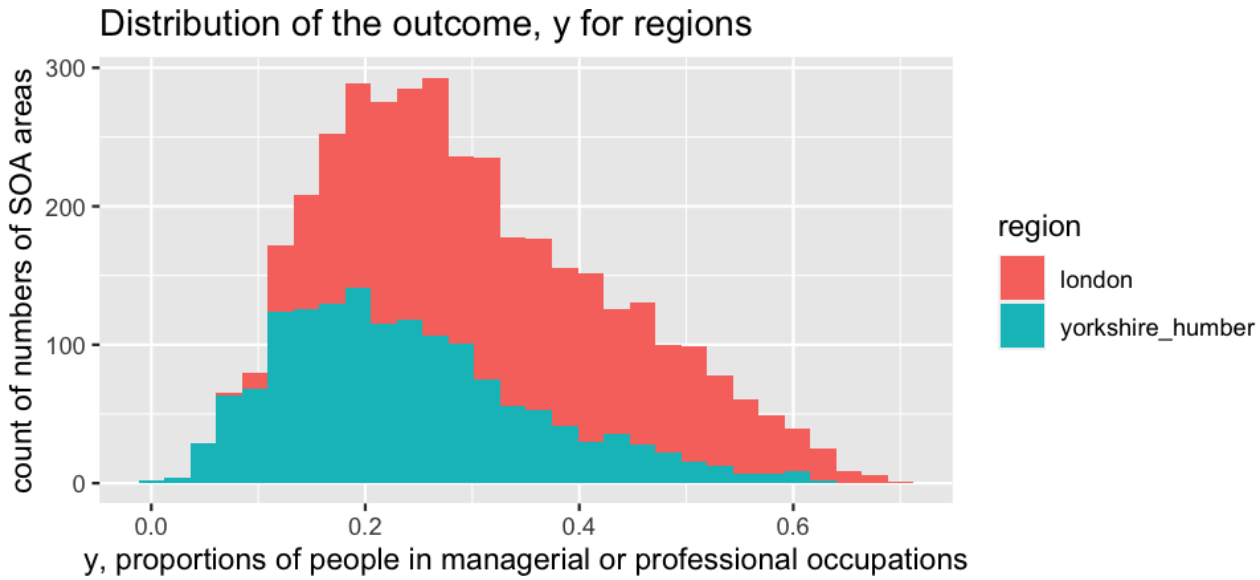
Table 46: Comparison of y, proportion of senior managers and professional occupations in SOA and MSOA

	y_SOA	y_MSOA
mean	0.295717	0.275763
sd	0.134452	0.095860

	y_SOA	y_MSOA
max	0.887850	0.661216
min	0.000000	0.063305

The areas London and Yorkshire and Humber are quite different as this graph illustrates. The London area tends to the right, with more managerial and professional roles, with very few places with very low proportion of managerial and professional. However, Yorkshire and Humber still has some concentrated areas with lots of senior roles.

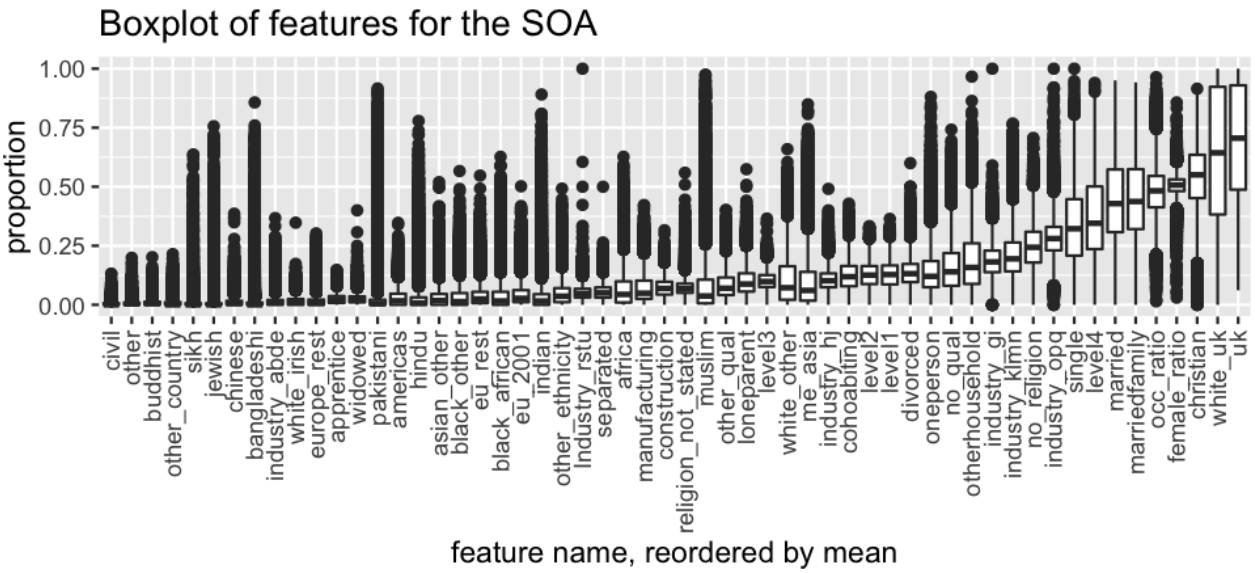
```
# graph illustrating the impact of London or Yorkshire
test_soa_final %>%
  select(y, london, yorkshire_humber) %>%
  pivot_longer(cols=c("london", "yorkshire_humber"),
               names_to="region", values_to="true") %>%
  filter(true=="1") %>%
  select(-true) %>%
  ggplot(aes(y, fill=region))  +
  geom_histogram(bins = 30) +
  ggtitle("Distribution of the outcome, y for regions") +
  xlab("y, proportions of people in managerial or professional occupations") +
  ylab("count of numbers of SOA areas")
```



Looking at the features, they are much broader as well, with many more outliers than the previous plot on the MSOA data. Note how the mean for UK nationals has dropped, due to half of the data coming from London which tends to be the initial destination for migrants to the UK.

```
#looking at the features
# box plot of features expressed as a ratio
# identify the geography features not necessary for this graph
load("rda/geo_lookup.rda")
# pull out only the places, not the geo and area columns
geography_features <-  intersect(
  str_subset(names(geo_lookup),  "area", negate = TRUE),
  str_subset(names(geo_lookup),  "geo", negate = TRUE))
# plot the graph
train_soa_final %>%
```

```r
# remove columns not needed, features that are not ratios
select(-age_median, -area_code, -all_of(geography_features)) %>%
#tidy the names, removing the _25_64
rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
#pivot to a long version with a row per feature/value
pivot_longer(cols = c(-y),
             names_to = "feature",
             values_to = "proportion") %>%
ggplot(aes(x=reorder(feature, proportion, FUN=mean), y=proportion))  +
geom_boxplot() +
ggtitle("Boxplot of features for the SOA") +
ylab("proportion") +
xlab("feature name, reordered by mean") +
theme(axis.text.x=element_text(angle = 90, hjust=1, vjust=0.5))
```



6.2 Modelling

Following the MSOA analysis, the top three algorithms which provide feature importance will be trained and tested, to develop the top 28 features for the GAM model:

- GLM
- GBM
- GLMBoost.

Then the top 3 algorithms from the MSOA modelling will be trained/tested, using a suitable number of features. It may be fewer than the MSOA model due to the large data set.

- GLM
- SVM Radial
- GAM

From this the best models will be used to build an ensemble.

6.2.1 GLM on the SOA data

GLM has a big improvement over the baseline, but much worse than the MSOA result, with 0.045790. It is quick, a minute, and will surely be able to handle the full SOA data set.

```
# GLM
# with the full train set
result_glm_train_soa <-
  results_train_method(train_soa_final, test_soa_final, "glm")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                          tail(result_glm_train_soa$results, 1))
rmse_results_soa %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_glm_train_soa$train, scale=FALSE)
plot(importance, 20)
importance$importance
```

Table 47: GLM model RMSE results for SOA

method	rmse
Mean of all locations	0.133419
"glm" - train_soa_final	0.045790

6.2.2 GBM on the SOA data

GBM took around fifteen minutes, and achieved a similar result to GLM, slightly better at 0.044361, but much worse than the result with the MSOA data. For importance, it was almost completely reliant on the level4 qualification.

```
# using the gbmm plyr, package
if (!require('gbm')) install.packages('gbm'); library('gbm')
if (!require('plyr')) install.packages('plyr'); library('plyr')
# with the full train set
result_gbm_train_soa <-
  results_train_method(train_soa_final, test_soa_final, "gbm")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                              tail(result_gbm_train_soa$results, 1))
rmse_results_soa %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_gbm_train_soa$train, scale=FALSE)
plot(importance, 20)
#tidy
importance_soa_gbm <- importance
```

Table 48: GBM model RMSE results for SOA

method	rmse
Mean of all locations	0.133419
"glm" - train_soa_final	0.045790
"gbm" - train_soa_final	0.044361

6.2.3 GLM Boost with the SOA data

And GLM Boost was very quick, and the worst of the three models with an RMSE of 0.046648.

```
#### Boosted Generalized Linear Model - glmboost
# uses the plyr, mboost packages
if (!require('mboost')) install.packages('mboost'); library('mboost')
if (!require('plyr')) install.packages('plyr'); library('plyr')
# with the full training set
result_glmboost_train_soa <-
  results_train_method(train_soa_final, test_soa_final, "glmboost")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                          tail(result_glmboost_train_soa$results, 1))
rmse_results_soa %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_glmboost_train_soa$train, scale=FALSE)
plot(importance, 20)
```

Table 49: GLM Boost model RMSE results for SOA

method	rmse
Mean of all locations	0.133419
"glmboost" - train_soa_final	0.046648

6.2.4 Feature prioritisation with the SOA data

With these three models the top 28 features for SOA were identified, and the data set with the top 28 features created.

```
# top 28 features
# using GLM, GBM, GLM boost to rank features
load("rda/importance_soa_gbm.rda")
load("rda/importance_soa_glm.rda")
load("rda/importance_soa_glmboost.rda")
#create ranking for gbm algorithm, with normalised score
gbm_soa_rank <- importance_soa_gbm$importance %>%
  rownames_to_column(var = "feature") %>%
  arrange(desc(Overall)) %>%
  cbind(data.frame(gbm_rank = 1:length(importance_soa_gbm$importance$Overall))) %>%
  mutate("gbm_overall" = Overall/max(Overall)) %>%
  select(-Overall)
head(gbm_soa_rank)
#create ranking for glm algorithm, with normalised score
glm_soa_rank <- importance_soa_glm$importance %>%
  rownames_to_column(var = "feature") %>%
  arrange(desc(Overall)) %>%
  cbind(data.frame(glm_rank = 1:length(importance_soa_glm$importance$Overall))) %>%
  mutate("glm_overall" = Overall/max(Overall)) %>%
  select(-Overall)
head(glm_soa_rank)
#create ranking for glmboost algorithm, with normalised score
glmboost_soa_rank <- importance_soa_glmboost$importance %>%
  rownames_to_column(var = "feature") %>%
  arrange(desc(Overall)) %>%
  cbind(data.frame(glmboost_rank = 1:length(importance_soa_glmboost$importance$Overall))) %>%
  mutate("glmboost_overall" = Overall/max(Overall)) %>%
```

```
  select(-Overall)
head(glmboost_soa_rank)
# combined ranking table, with mean, and reordered
feature_rank <- glm_soa_rank %>%
  left_join(gbm_soa_rank) %>%
  left_join(glmboost_soa_rank) %>%
  dplyr::mutate(mean_rank =
                  (glm_rank+gbm_rank+glmboost_rank)/3, .after = "feature") %>%
  dplyr::mutate(mean_overall =
                  (glm_overall+gbm_overall+glmboost_overall)/3, .after = "feature") %>%
  arrange(desc(mean_overall))
head(feature_rank)

# load("rda/feature_rank_soa.rda")
#create new training set, using the function from earlier
train_soa_top28 <- topfeature_overall(28, train_soa_final)
```

6.2.5 SVM Radial with the SOA data

The SVM Radial model was very slow and did not work well at all, by far the worst of the models so far at 0.052583. There were a lot of errors, and it is likely that it struggled with the size.

```
#### SVM radial
# uses package kernlab
if (!require('kernlab')) install.packages('kernlab'); library('kernlab')
# with the full train set
result_svmradial_train_soa <-
  results_train_method(train_soa_final, test_soa_final, "svmRadial")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                          tail(result_svmradial_train_soa$results, 1))
```

Therefore the top28 training set was next used, but this failed after taking much longer than the earlier training.

```
# with the top28 train set
result_svmradial_train_soa <-
  results_train_method(train_soa_top28, test_soa_final, "svmRadial")
```

This probably means that the algorithm struggles with the amount of data. In order to get some results, the same process as with the MSOA data was used, creating a data sets with subsets of features and observations. Trying a "small" data set with 5000 observations, and even with only 5 features, the result was pretty good, much better than with the full set earlier, with 0.0488267.

```
# training with 15 features
load("rda/feature_rank_soa.rda")
#create new training set, using the function from earlier
train_soa_top5 <- topfeature_overall(5, train_soa_final)
train_soa_top10 <- topfeature_overall(10, train_soa_final)
train_soa_top15 <- topfeature_overall(15, train_soa_final)
# with the top15 train set
result_svmradial_train_soa_top15 <-
  results_train_method(train_soa_top15, test_soa_final, "svmRadial")
# with the top5 train set
result_svmradial_train_soa_top5 <-
  results_train_method(train_soa_top5, test_soa_final, "svmRadial")
```

```
#sampling the 5000 observations
set.seed(2001, sample.kind="Rounding")
index <- sample(1:nrow(train_soa_final), 5000, replace = FALSE)
train_soa_5k <- train_soa_final[index,]
#create new training set, using the function from earlier
train_soa_5k_top5 <- topfeature_overall(5, train_soa_5k)
# with the 5k top5 train set
result_svmradial_train_soa_5k_top5 <-
  results_train_method(train_soa_5k_top5, test_soa_final, "svmRadial")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                              tail(result_svmradial_train_soa_5k_top5$results, 1))
```

Increasing the number of features to improve the SVM Radial predictions, 28 features and 5000 observations provided good results, 0.0452578, which is now second to the GBM model, with only a seventh of the observations available.

```
# with the 5k top10 train set
train_soa_5k_top10 <- topfeature_overall(10, train_soa_5k)
# training the model
result_svmradial_train_soa_5k_top10 <-
  results_train_method(train_soa_5k_top10, test_soa_final, "svmRadial")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                              tail(result_svmradial_train_soa_5k_top10$results, 1))
# with the 5k top15 train set
train_soa_5k_top15 <- topfeature_overall(15, train_soa_5k)
# training the model
result_svmradial_train_soa_5k_top15 <-
  results_train_method(train_soa_5k_top15, test_soa_final, "svmRadial")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                              tail(result_svmradial_train_soa_5k_top15$results, 1))
# with the 5k top28 train set
train_soa_5k_top28 <- topfeature_overall(28, train_soa_5k)
# training the model
result_svmradial_train_soa_5k_top28 <-
  results_train_method(train_soa_5k_top28, test_soa_final, "svmRadial")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                              tail(result_svmradial_train_soa_5k_top28$results, 1))
rmse_results_soa %>% knitr::kable()
```

Trying with more observations to see how far it can go. With 10,000 observations, it performs slightly better than GBM, at 0.044342, taking 15 minutes. And with 15,000 observations, taking 50 minutes, and better still at 0.0438095.

```
#sampling the 10000 observations
set.seed(2001, sample.kind="Rounding")
index <- sample(1:nrow(train_soa_final), 10000, replace = FALSE)
train_soa_10k <- train_soa_final[index,]
# with the 10k top28 train set
train_soa_10k_top28 <- topfeature_overall(28, train_soa_10k)
# training the model
result_svmradial_train_soa_10k_top28 <-
```

```
  results_train_method(train_soa_10k_top28, test_soa_final, "svmRadial")
# extract the rmse from the results
rmse_results_soa <-
  bind_rows(rmse_results_soa,
            tail(result_svmradial_train_soa_10k_top28$results, 1))
rmse_results_soa %>% knitr::kable()

# tidy
save(rmse_results_soa, file="rda/rmse_results_soa.rda")
save(result_svmradial_train_soa_10k_top28, file="rda/result_svmradial_train_soa_10k_top28.rda")
rm(result_svmradial_train_soa_10k_top28, index)

#sampling the 15000 observations
set.seed(2001, sample.kind="Rounding")
index <- sample(1:nrow(train_soa_final), 15000, replace = FALSE)
train_soa_15k <- train_soa_final[index,]
# with the 15k top28 train set
train_soa_15k_top28 <- topfeature_overall(28, train_soa_15k)
# training the model
result_svmradial_train_soa_15k_top28 <-
  results_train_method(train_soa_15k_top28, test_soa_final, "svmRadial")
# extract the rmse from the results
rmse_results_soa <-
  bind_rows(rmse_results_soa,
            tail(result_svmradial_train_soa_15k_top28$results, 1))
rmse_results_soa %>% knitr::kable()
```

Table 50: SVM Radial model RMSE results for SOA

method	rmse
Mean of all locations	0.1334185
"gbm" - train_soa_final	0.0443612
"svmRadial" - train_soa_final	0.0525831
"svmRadial" - train_soa_5k_top5	0.0488267
"svmRadial" - train_soa_5k_top28	0.0452578
"svmRadial" - train_soa_10k_top28	0.0443419
"svmRadial" - train_soa_15k_top28	0.0438095

6.2.6 GAM with the SOA data

Now working with the GAM model to see how that will perform. GAM on the top 28 features with all 34,282 observations produced a result of 0.046186, behind the GLM and GBM models, surprising.

```
# this uses the mgcv and nlme packages
if (!require('mgcv')) install.packages('mgcv'); library('mgcv')
if (!require('nlme')) install.packages('nlme'); library('nlme')
# with the soa top 28 training set
result_gam_train_soa_top28 <-
  results_train_method(train_soa_top28, test_soa_final, "gam")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                              tail(result_gam_train_soa_small_top28$results, 1))
```

Training with subsets of the data once more, top 28 with 5000 observations is not so good, but with 10,000 it

is better than with the full set of observations, at 0.045806 close to the GLM model with 0.045790. However, with 15,000 observations the performance was worse that the 10,000 sample.

```
# with the 5k top28 SOA training set
result_gam_train_soa_5k_top28 <-
  results_train_method(train_soa_5k_top28, test_soa_final, "gam")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                              tail(result_gam_train_soa_5k_top28$results, 1))
rmse_results_soa %>% knitr::kable()
#checking the variable importance
importance <- varImp(result_gam_train_soa_5k_top28$train, scale=FALSE)
plot(importance, 20)
# tidy
save(rmse_results_soa, file="rda/rmse_results_soa.rda")
save(result_gam_train_soa_top28, file="rda/result_gam_train_soa_top28.rda")
rm(result_gam_train_soa_top28)

# with the 10k top28 SOA training set
result_gam_train_soa_10k_top28 <-
  results_train_method(train_soa_10k_top28, test_soa_final, "gam")
# extract the rmse from the results
rmse_results_soa <- bind_rows(rmse_results_soa,
                              tail(result_gam_train_soa_10k_top28$results, 1))
rmse_results_soa %>% knitr::kable()
# tidy
save(rmse_results_soa, file="rda/rmse_results_soa.rda")
save(result_gam_train_soa_10k_top28,
     file="rda/result_gam_train_soa_10k_top28.rda")

# with the 15k top28 SOA training set
result_gam_train_soa_15k_top28 <-
  results_train_method(train_soa_15k_top28, test_soa_final, "gam")
# extract the rmse from the results
rmse_results_soa <-
  bind_rows(rmse_results_soa,
            tail(result_gam_train_soa_15k_top28$results, 1))
```

Table 51: GAM model RMSE results for SOA

method	rmse
Mean of all locations	0.1334185
“gbm” - train_soa_final	0.0443612
“gam” - train_soa_10k_top28	0.0458059
“gam” - train_soa_top28	0.0461864
“gam” - train_soa_15k_top28	0.0462241
“gam” - train_soa_5k_top28	0.0470019

6.3 Final ensemble for the SOA data

Previously for the MSOA model, the best result was an ensemble of two algorithms: SVM Radial and GAM with 28 features.

Looking at the leading models, the top two, SVM Radial and GBM have a gap to the rest, so would probably

be the best ensemble.

Table 52: Leading model RMSE results for SOA

method	rmse
"svmRadial" - train_soa_15k_top28	0.0438095
"gbm" - train_soa_final	0.0443612
"glm" - train_soa_final	0.0457900
"gam" - train_soa_10k_top28	0.0458059
"glmboost" - train_soa_final	0.0466483

The final model is an ensemble of two, which provided the best result of an RMSE of 0.0431886.

```
#ensemble of SVM Radial and GBM
load("rda/result_svmradial_train_soa_15k_top28.rda")
load("rda/result_gbm_train_soa.rda")
y_hat_ens_table6 <-
  data.frame(svmradial_y_hat = result_svmradial_train_soa_15k_top28$y_hat,
             gbm_y_hat = result_gbm_train_soa$y_hat) %>%
  #calculate average of the y_hats
  mutate(y_hat_ave = rowMeans(.))
#test against the actual values
rmse_ens <- rmse(test_soa_final$y, y_hat_ens_table6$y_hat_ave)
rmse_results_soa <- bind_rows(rmse_results_soa,
                          tibble(method="Ensemble svmRadial, gbm",
                                 rmse = rmse_ens))
rmse_results_soa %>%
  arrange(rmse) %>% head(10) %>%
  knitr::kable(caption="RMSE results for SOA")
```

Table 53: RMSE results for SOA

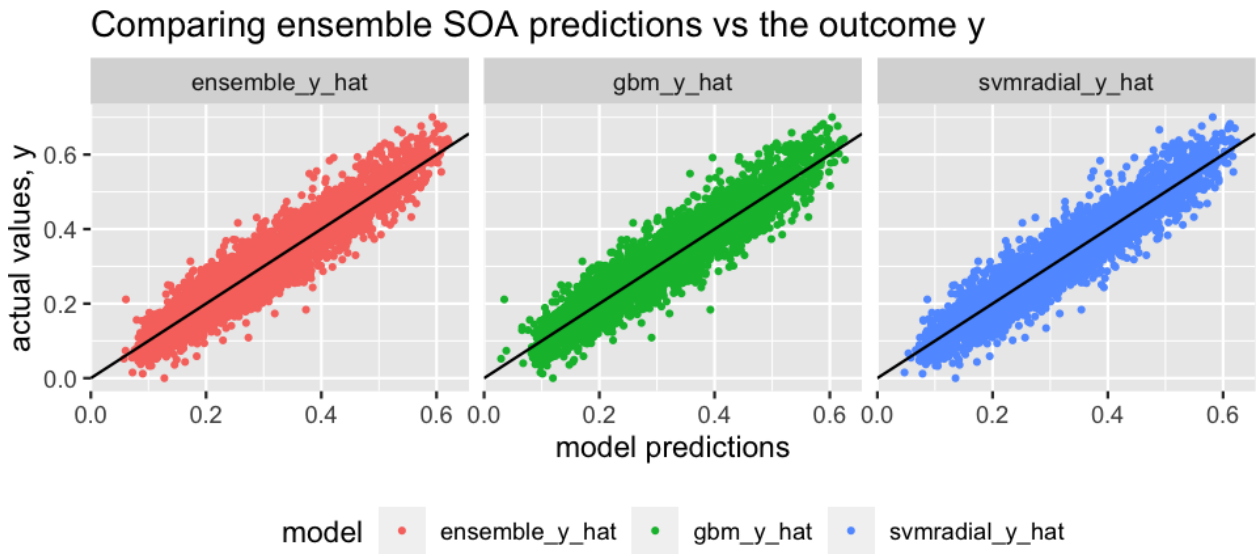
method	rmse
Ensemble svmRadial, gbm	0.043189
Mean of all locations	0.133419

6.4 Visualisation of the results

Looking at the results, it is not too illuminating. In general the distribution is a bit wider than the MSOA areas, nothing too obvious. Apart from the higher value areas, as the models a bit conservative and pretty much predict no proportions over 0.6, whereas the real data has values as high as 0.7. This could be the cause of the difference with the SOA results, as the larger MSOA areas would tend to average out and reduce these very high values, and then the weaknesses in the models would not show up.

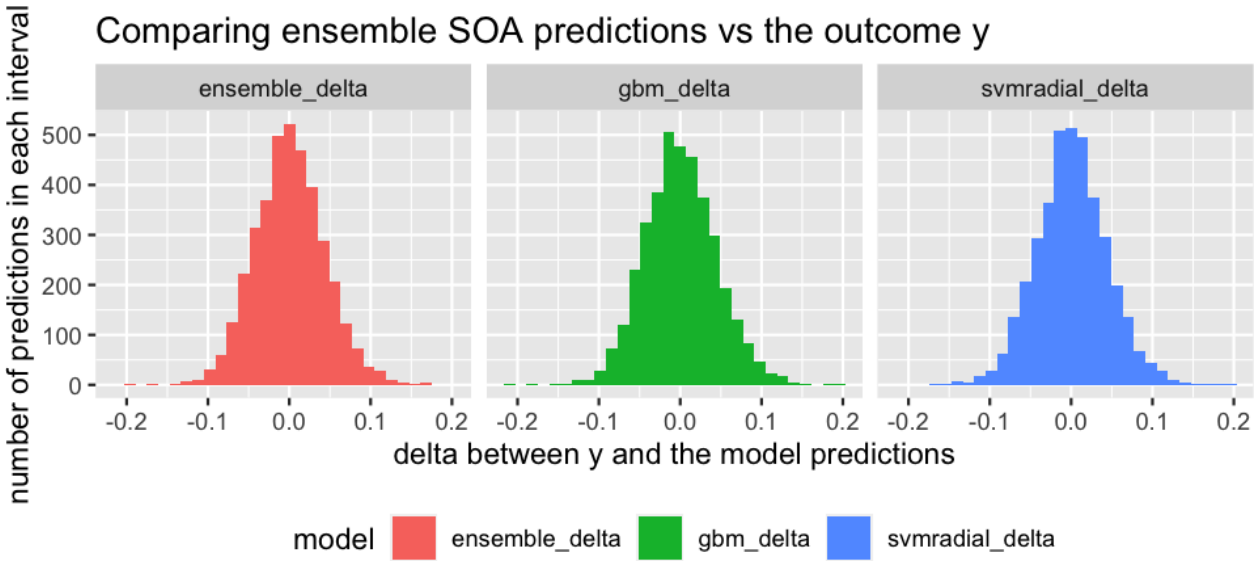
```
# looking at the best ensemble model, and constituent ones svmRadial and gbm
# plot of y against y_hat for all three
y_hat_ens_table6 %>%
  cbind(y = test_soa_final$y) %>%
  rename(ensemble_y_hat=y_hat_ave) %>%
  pivot_longer(cols=contains("y_hat"), names_to="model", values_to="y_hat") %>%
  ggplot(aes(x=y_hat, y=y, col=model)) +
  geom_point(size=0.7) +
  geom_abline(slope = 1, intercept = 0, col="black")  +
```

```
  facet_grid(. ~model) +
  ggtitle("Comparing ensemble SOA predictions vs the outcome y") +
  xlab("model predictions") +
  ylab("actual values, y") +
  theme(legend.position = "bottom")
```



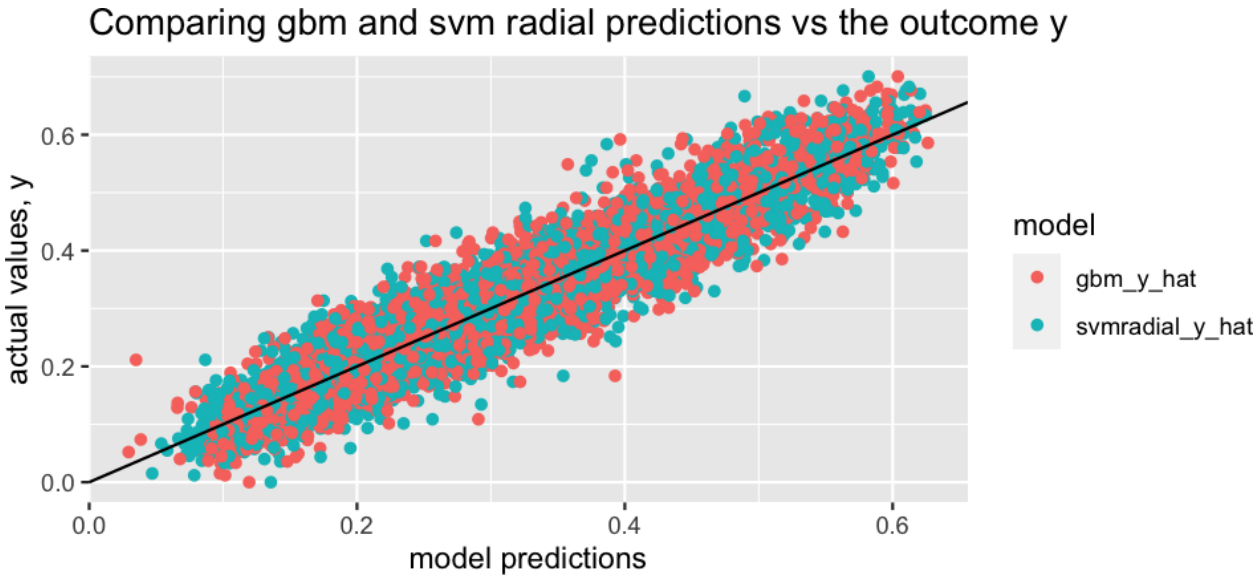
The graphs of the delta from the outcome y are all sensible shapes.

```
# histogram of deltas for all three
y_hat_ens_table6 %>%
  cbind(y = test_soa_final$y) %>%
  mutate(svmradial_delta = (y-svmradial_y_hat)) %>%
  mutate(gbm_delta = (y-gbm_y_hat)) %>%
  mutate(ensemble_delta = (y-y_hat_ave)) %>%
  select(-svmradial_y_hat, -gbm_y_hat, -y_hat_ave) %>%
  pivot_longer(cols=contains("delta"), names_to="model", values_to="delta") %>%
  ggplot(aes(delta, fill=model)) +
  geom_histogram(bins = 30) +
  facet_grid(. ~model) +
  ggtitle("Comparing ensemble SOA predictions vs the outcome y") +
  xlab("delta between y and the model predictions") +
  ylab("number of predictions in each interval") +
  theme(legend.position = "bottom")
```



There is nothing obvious from this graph on where the different models are better.

```
# comparing the svmradial and gam models
y_hat_ens_table6 %>%
  cbind(y = test_soa_final$y) %>%
  select(-y_hat_ave) %>%
  pivot_longer(cols=contains("y_hat"), names_to="model", values_to="y_hat") %>%
  ggplot(aes(x=y_hat, y=y, col=model)) +
  geom_point()  +
  geom_abline(slope = 1, intercept = 0, col="black") +
  ggtitle("Comparing gbm and svm radial predictions vs the outcome y") +
  xlab("model predictions") +
  ylab("actual values, y")
```



7 Model interpretation and feature importance

This section returns back to the motivation for this topic to consider achievements of top jobs in the context of the Black lives matter society wide debate.
https://www.theguardian.com/business/2020/jul/28/bame-representation-uk-top-jobs-colour-of-power-survey

Feature importance and correlation are analysed to see if there is data to enlighten the discussion, specifically focusing on the ethnicity features.

7.1 Feature importance

The earlier investigations in to important features for the MSOA areas found the following features are positively correlated:

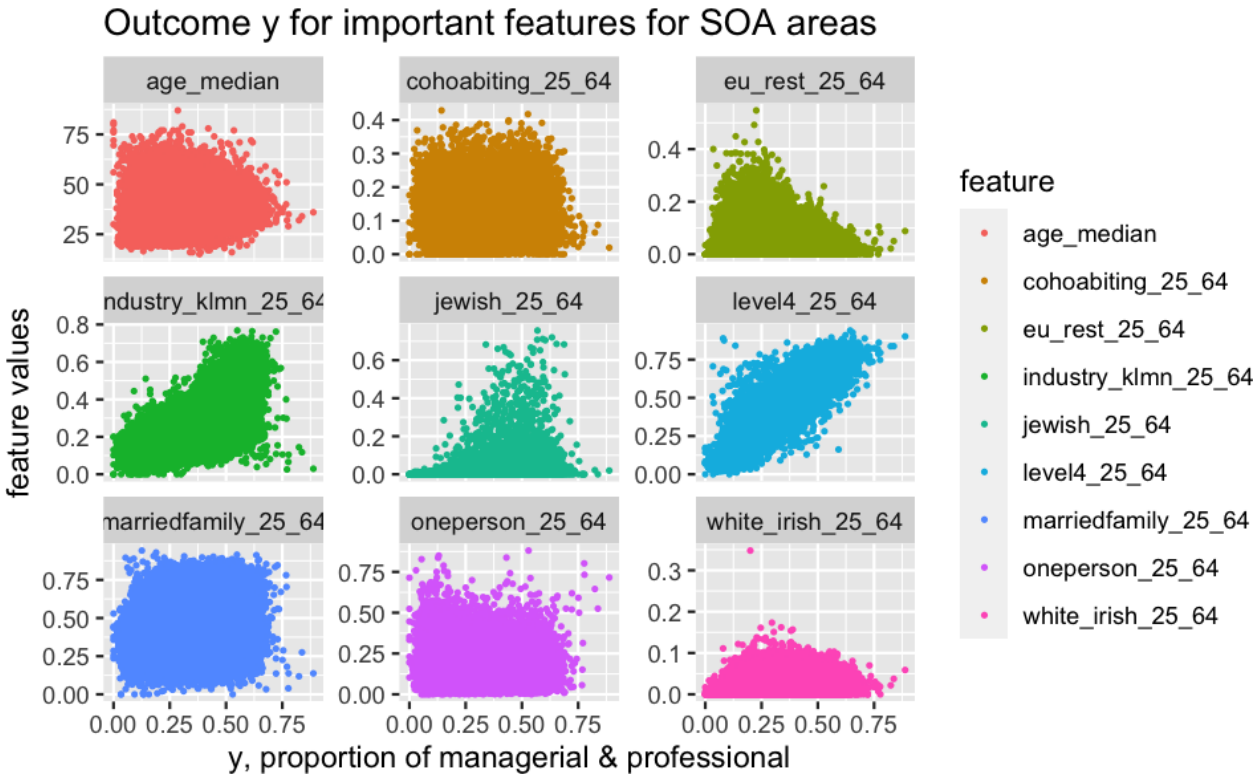
- level4_25_64
- industry_klmn_25_64
- white_other_25_64
- jewish_25_64
- age_median

The following are negatively correlated:

- separated_25_64
- cohoabiting_25_64
- no_qual_25_64

Carrying out the same analysis for the SOA area, the results have more variety as expected, with a slightly different set of features in the top 9. One interesting area is in the level4 qualifications, where for the MSOA there was a very clear correlation, but here there are a number of locations with high qualifications and low professional and managerial proportions.

```
# plotting the top 9 for SOA
load("rda/feature_rank_soa.rda")
feature_rank_soa <- feature_rank
feature_top9_soa <- feature_rank_soa %>%
  select(feature, mean_rank, mean_overall) %>%
  arrange(desc(mean_overall)) %>% head(9) %>%
  pull(feature)
#create the plot
train_soa_final %>%
  #pivot to a long version with a row per feature/value
  pivot_longer(cols = !"y",
               names_to = "feature",
               values_to = "value") %>%
  filter(feature %in% feature_top9_soa) %>%
  ggplot(aes(y, value, col=feature))  +
  geom_point(size=0.5) +
  facet_wrap(. ~feature,  scales = "free_y") +
  ggtitle("Outcome y for important features for SOA areas") +
  xlab("y, proportion of managerial & professional") +
  ylab("feature values")
```



The following are positively correlated:

- level4_25_64
- industry_klmn_25_64
- white_irish_25_64
- jewish_25_64
- age_median
- marriedfamily_25_64
- oneperson_25_64
- cohoabiting_25_64

With only one negatively correlated:

- eu_rest_25_64

One, cohabiting, has moved from negative to positive correlation which is interesting. This may be as the SOA only includes London and one other region, and it could be that cohabiting is positive in London and negative elsewhere.

```
#looking at correlation with y
data.frame(cor(train_soa_final)) %>%
  select(y) %>%
  rownames_to_column(var="feature") %>%
  filter(feature %in% feature_top9_soa) %>%
  arrange(desc(abs(y))) %>%
  knitr::kable(caption="Correlation with y for the 9 most important features, in SOA")
```

Table 54: Correlation with y for the 9 most important features, in SOA

feature	y
level4_25_64	0.905228
industry_klmn_25_64	0.697420
white_irish_25_64	0.344061
jewish_25_64	0.266607
age_median	0.084850
marriedfamily_25_64	0.077433
eu_rest_25_64	-0.073657
oneperson_25_64	0.055514
cohoabiting_25_64	0.033358

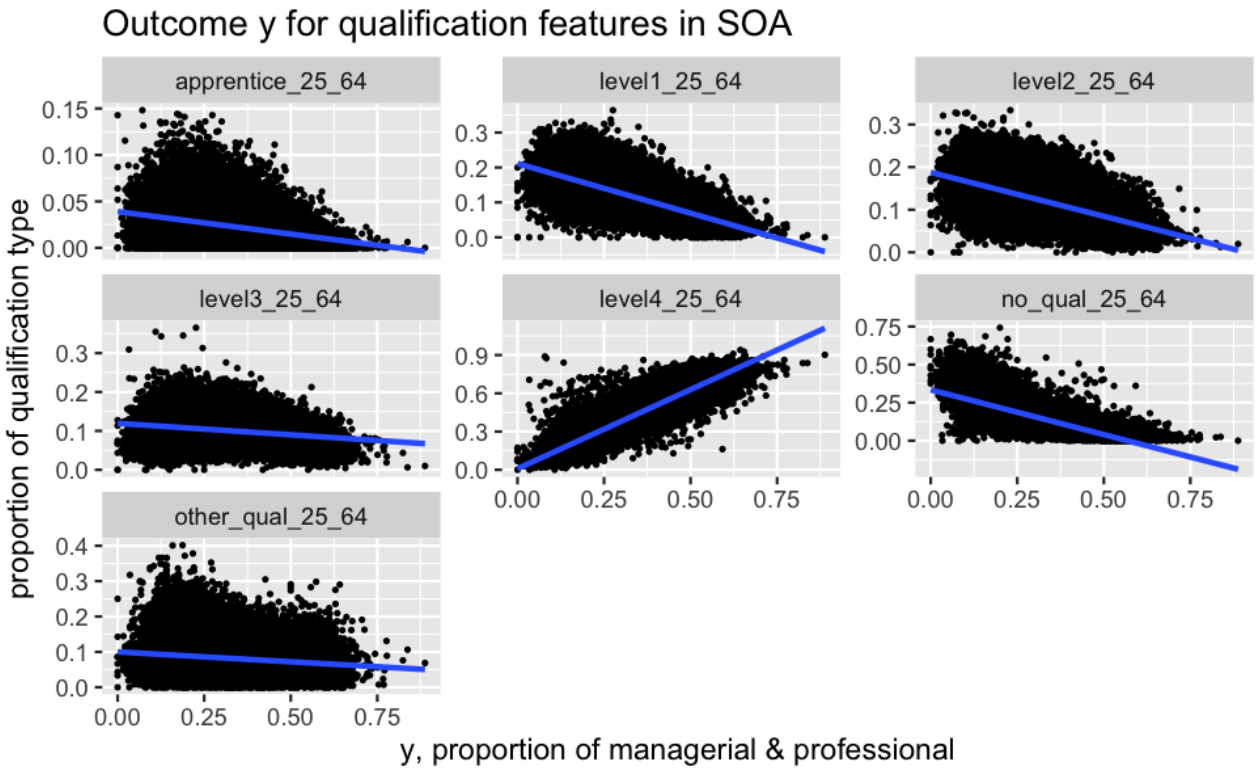
Given that they appear in the top lists it is worth looking in mode detail at the following:

- Qualification predictors
- Industry predictors
- Household status predictors
- Marital status predictors
- Religion predictors
- Ethnicity predictors

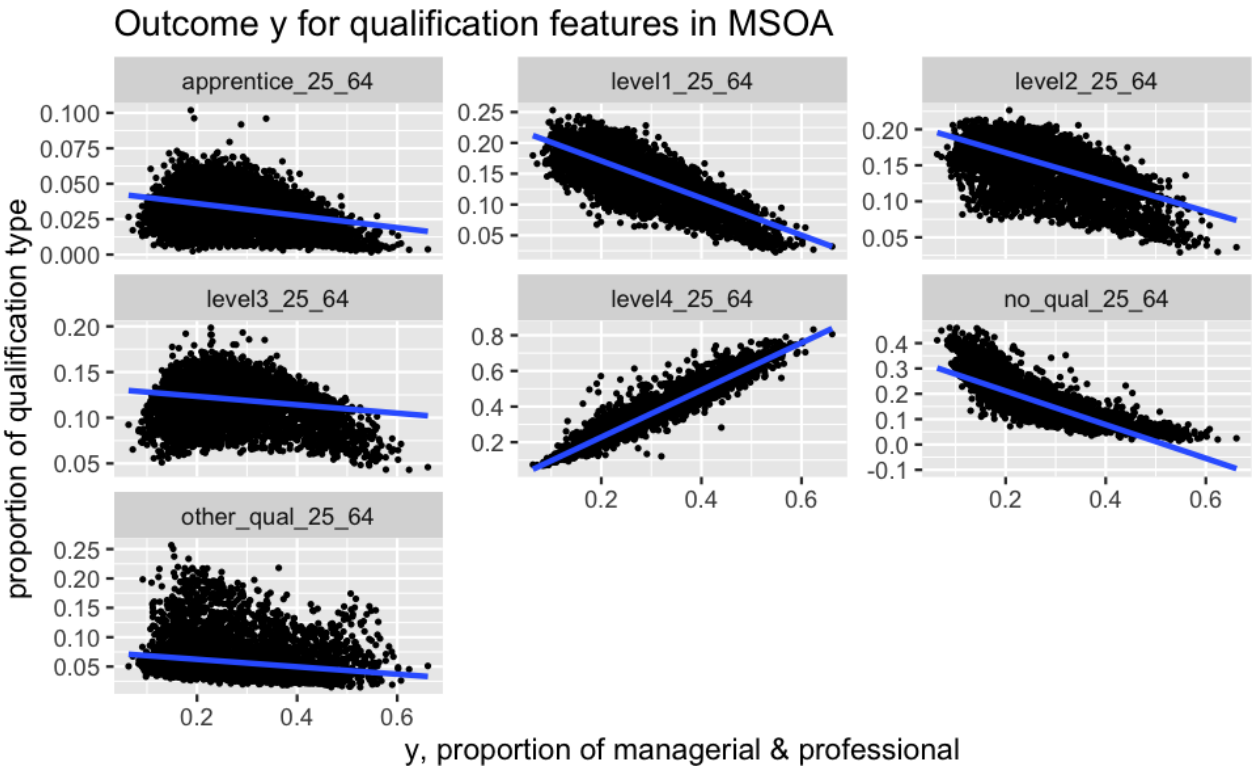
7.1.1 Qualification deep dive

Looking at all the qualification features, and mapping against the y values. Here only the Level4 and no qualification have values about 0.25, which is likely the reason for being important features in the models. In addition, all are negatively correlated apart from Level 4, with level 1 and no qualification clearly negative.

```
# qualification features SOA
load('rda/qualifications.rda')
qualifications_features <- names(qualifications) %>% str_subset("25_64")
train_soa_final %>%  select(y, all_of(qualifications_features)) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for qualification features in SOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of qualification type")
```




```
# qualification features MSOA
train_set_final %>%  select(y, all_of(qualifications_features)) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for qualification features in MSOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of qualification type")
```



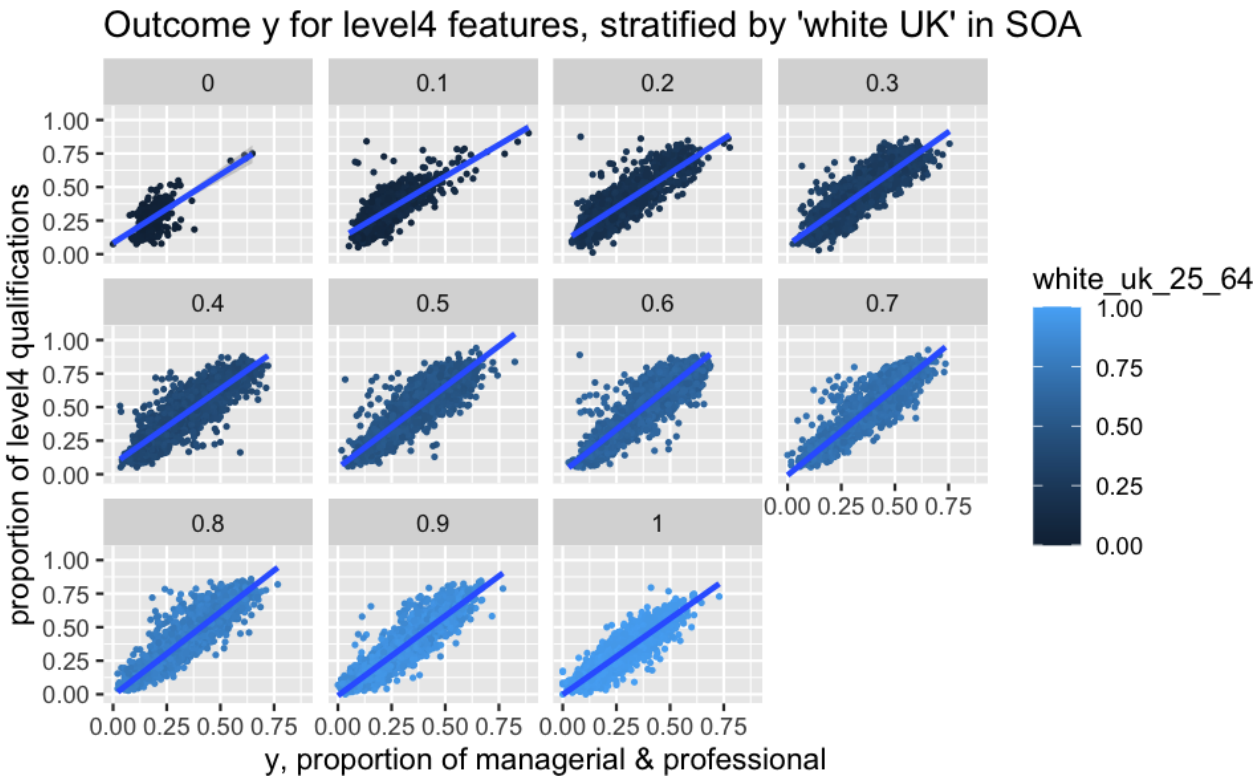
Given the motivation it may be interesting to look at ethnicity and qualifications together. As the qualifications are not broken down directly by ethnicity, and there are many ethnicities, this is possible indirectly, by visualising level 4 qualifications for MSOA and SOA areas with the proportion of white UK people. In both location types the graphs have a linear regression line, as expected as level4 qualification is the main driver of the models. Visually, it looks like the areas with a high white proportion tend to be closely correlated and the others areas less so, especially for the SOA areas where the lower proportions (darker blue) have both higher and lower senior managerial and professional proportions than might be expected.

```
#Plot of y vs level 4 with colour as proportion of white uk
# initial prep of the SOA data, retaining only the relevant columns
train_soa_final_excerpt <- train_soa_final %>%
  select(y, level4_25_64, white_uk_25_64) %>%
  rename(SOA = y) %>%
  # pivot longer with the lavel soa
  pivot_longer(cols = SOA,
               names_to = "area_type",
               values_to = "y")
train_set_final %>%
  select(y, level4_25_64, white_uk_25_64) %>%
  rename(MSOA = y) %>%
  # pivot longer with the label MSOA
  pivot_longer(cols = MSOA,
               names_to = "area_type",
               values_to = "y") %>%
  # adding the SOA data
  rbind(train_soa_final_excerpt) %>%
  ggplot(aes(y, level4_25_64, col=white_uk_25_64))  +
  facet_grid(.~area_type) +
  geom_point(size=0.7) +
  geom_smooth(method = "lm") +
```

```
  ggtitle("Outcome y for level4 features") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of level4 qualifications")
```

This can be illustrated further with stratification of the level4 vs y by white uk, where you can see that for lower proportions of white UK, the link with education is less important. Only SOA data is used here to provide more detail and variation.

```
# looking at stratification for the white uk, soa
train_soa_final %>%
  select(y, level4_25_64, white_uk_25_64) %>%
  # round to nearest 0.1
  mutate(white_uk_strata = round(white_uk_25_64, 1)) %>%
  ggplot(aes(y, level4_25_64, col=white_uk_25_64))  +
  facet_wrap(.~white_uk_strata) +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for level4 features, stratified by 'white UK' in SOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of level4 qualifications")
```



Listing the correlation values, it illustrates that below 60% the link between qualification and professional and managerial occupations is weaker, especially below 30%.

```
# stratification and correlation
train_soa_final %>%
  select(y, level4_25_64, white_uk_25_64) %>%
  # round to nearest 0.1
  mutate(white_uk_strata = round(white_uk_25_64, 1)) %>%
  group_by(white_uk_strata) %>%
  dplyr::summarize(correlation = cor(y, level4_25_64)) %>%
  knitr::kable(caption="Correlation between y and level4 qualifications, by strata of 'white uk'")
```

Table 55: Correlation between y and level4 qualifications, by strata of 'white uk'

white_uk_strata	correlation
0.0	0.645016
0.1	0.752867
0.2	0.830405
0.3	0.871952
0.4	0.873747
0.5	0.885605
0.6	0.905012
0.7	0.922077
0.8	0.915903
0.9	0.923779
1.0	0.913472

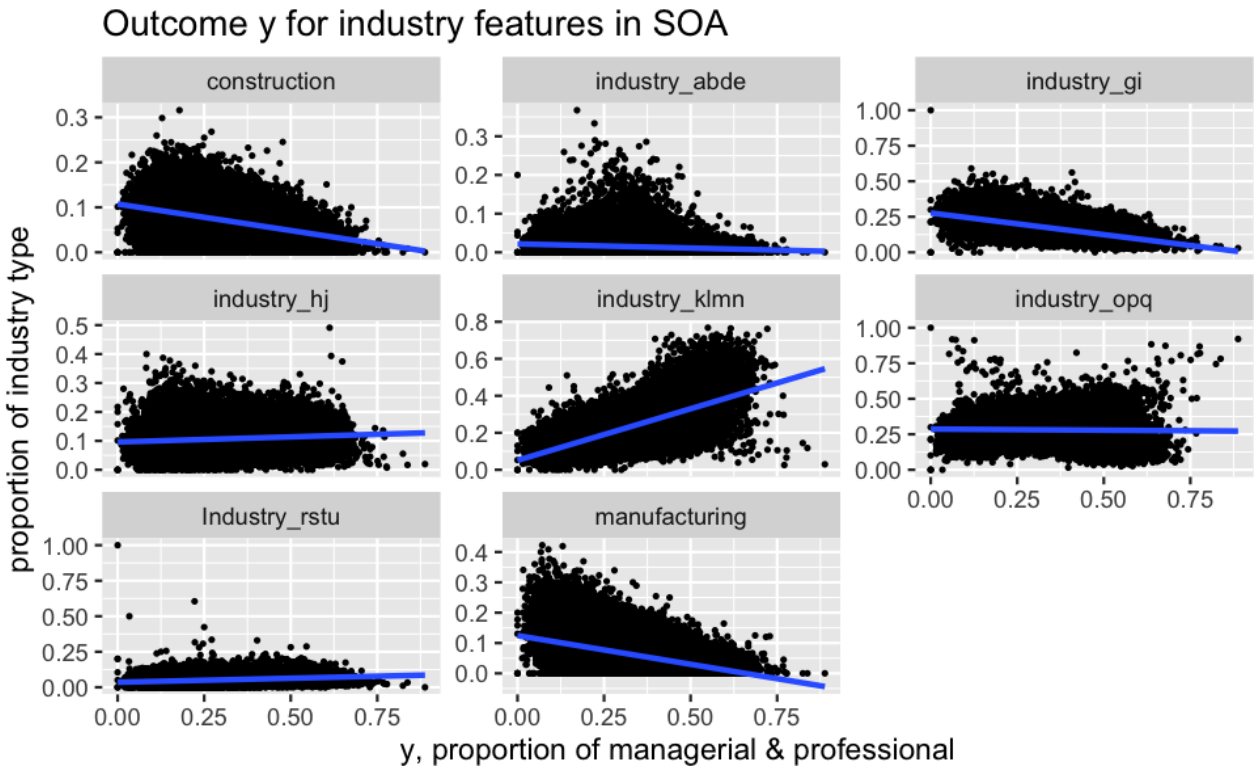
7.1.2 Industry predictors

Visualising the industry predictors in the same way to the qualifications, most of the features have flatish results, apart from Industry KLMN, positive, and Manufacturing, construction and GI, negative. Also, along with KLMN, OPQ is the only other industry which has high proportions in areas with high proportion of senior roles, even though overall it is flat. This matches the press reports of the reliance of the UK on the professional services sector, and also public services.

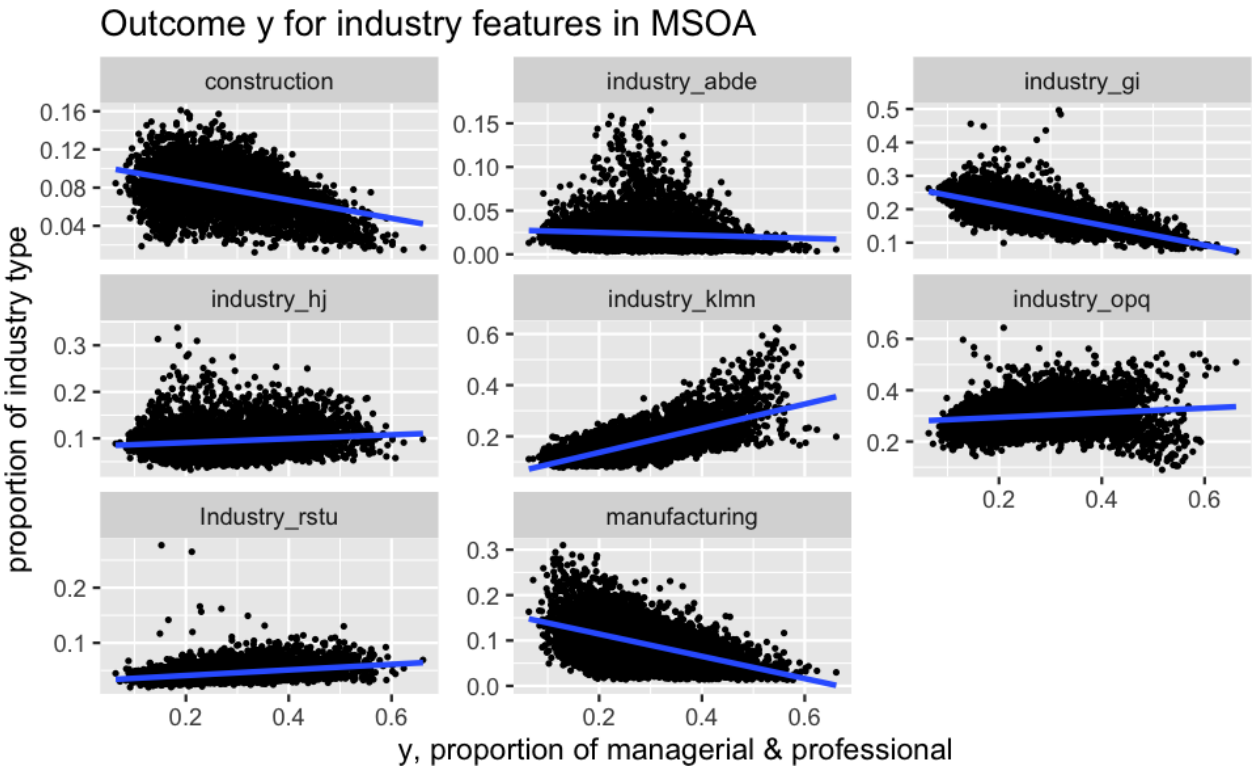
The key industries are:

- Industry KLMN
  - Industry K - Financial and insurance activities
  - Industry L - Real estate activities
  - Industry M - Professional, scientific and technical activities
  - Industry N - Administrative and support service activities
- Industry GI
  - Industry G - Wholesale and retail trade; repair of motor vehicles and motor cycles
  - Industry I - Accommodation and food service activities
- Industry OPQ
  - Industry O - Public administration and defence; compulsory social security
  - Industry P - Education
  - Industry Q - Human health and social work activities

```
# listing industry features
load('rda/industry.rda')
industry_features <- names(industry) %>% str_subset("25_64")
#industry features SOA
train_soa_final %>%  select(y, all_of(industry_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for industry features in SOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of industry type")
```




```
#industry features MSOA
train_set_final %>%  select(y, all_of(industry_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for industry features in MSOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of industry type")
```

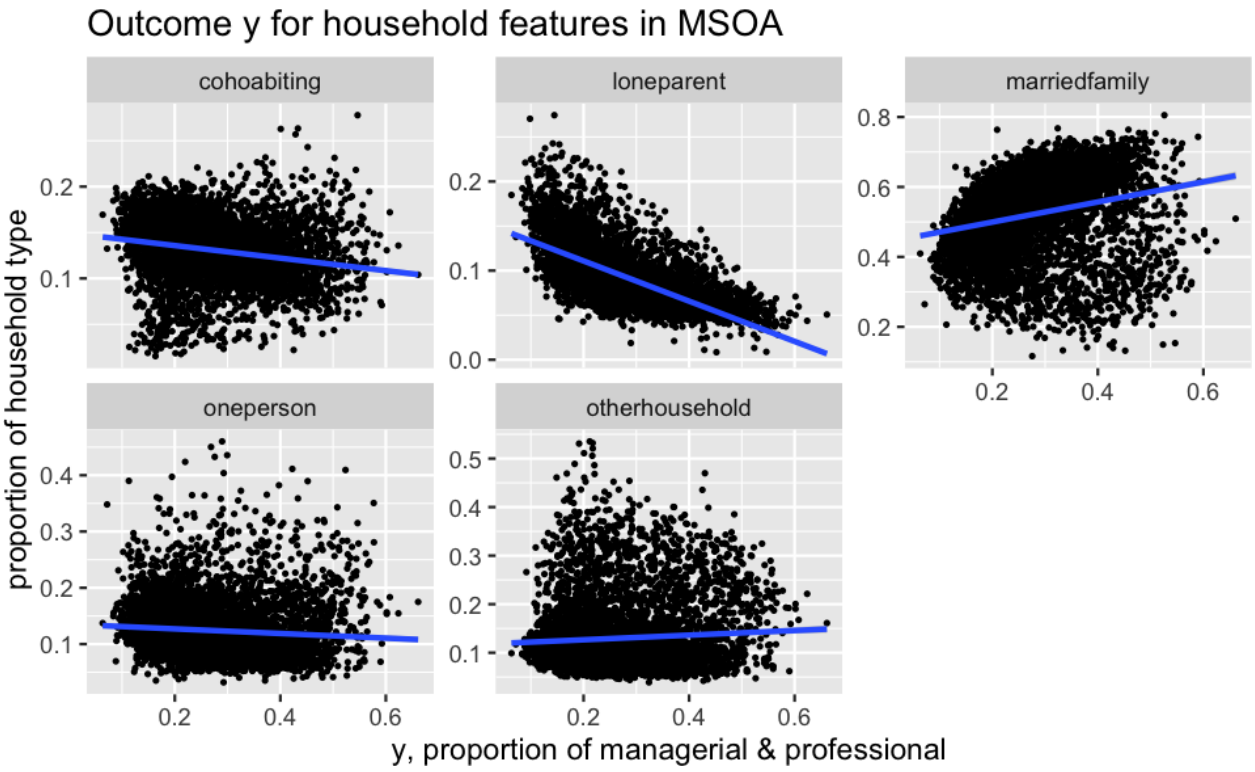


7.1.3 Household status predictors

In the household features, loneparent has a clear negative correlation, although it does not come out as an important feature, perhaps it is closely related to another feature, or that it, along with cohabiting have a lowish proportion, less than 0.4. Married family, one person and cohabiting appear on the important lists, but have only a slight gradient.

```
# household features soa
load('rda/household.rda')
household_features <- names(household) %>% str_subset("25_64")
train_soa_final %>%  select(y, all_of(household_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for household features in SOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of household type")
```

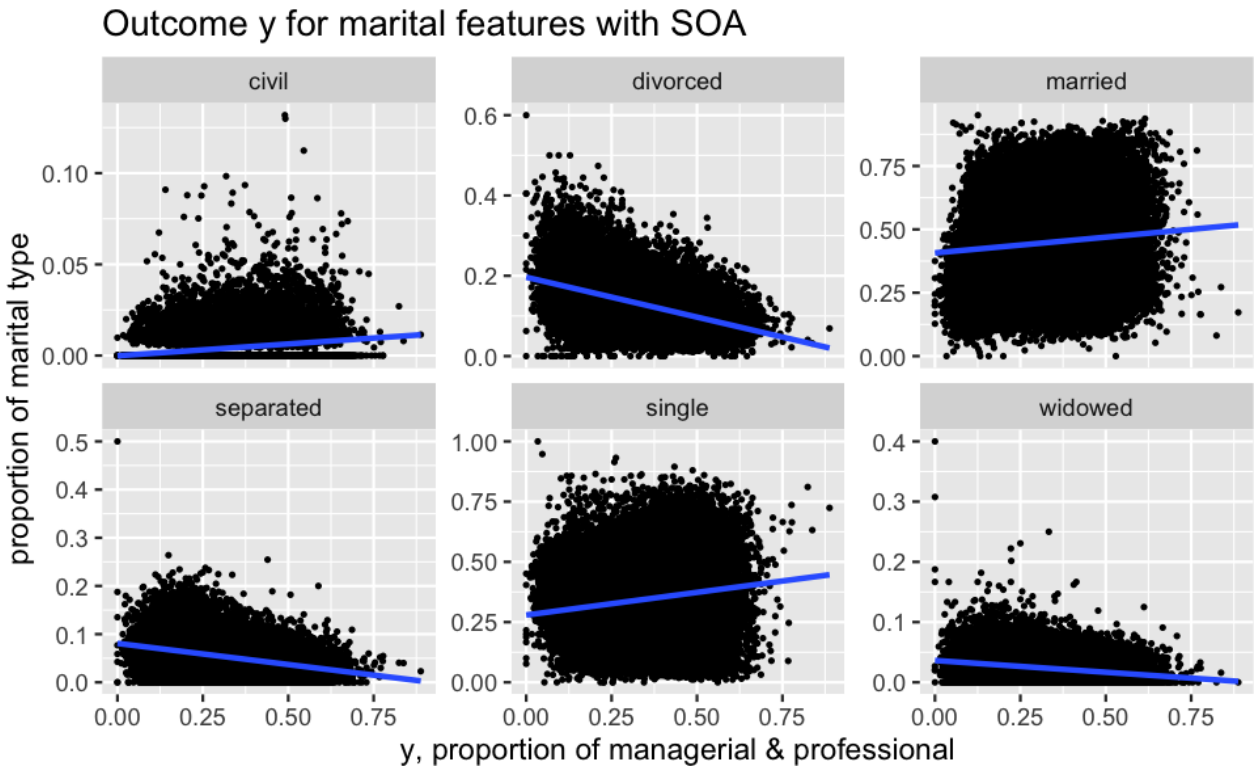

```
# household features msoa
load("rda/train_set_final.rda")
train_set_final %>%  select(y, all_of(household_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for household features in MSOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of household type")
```



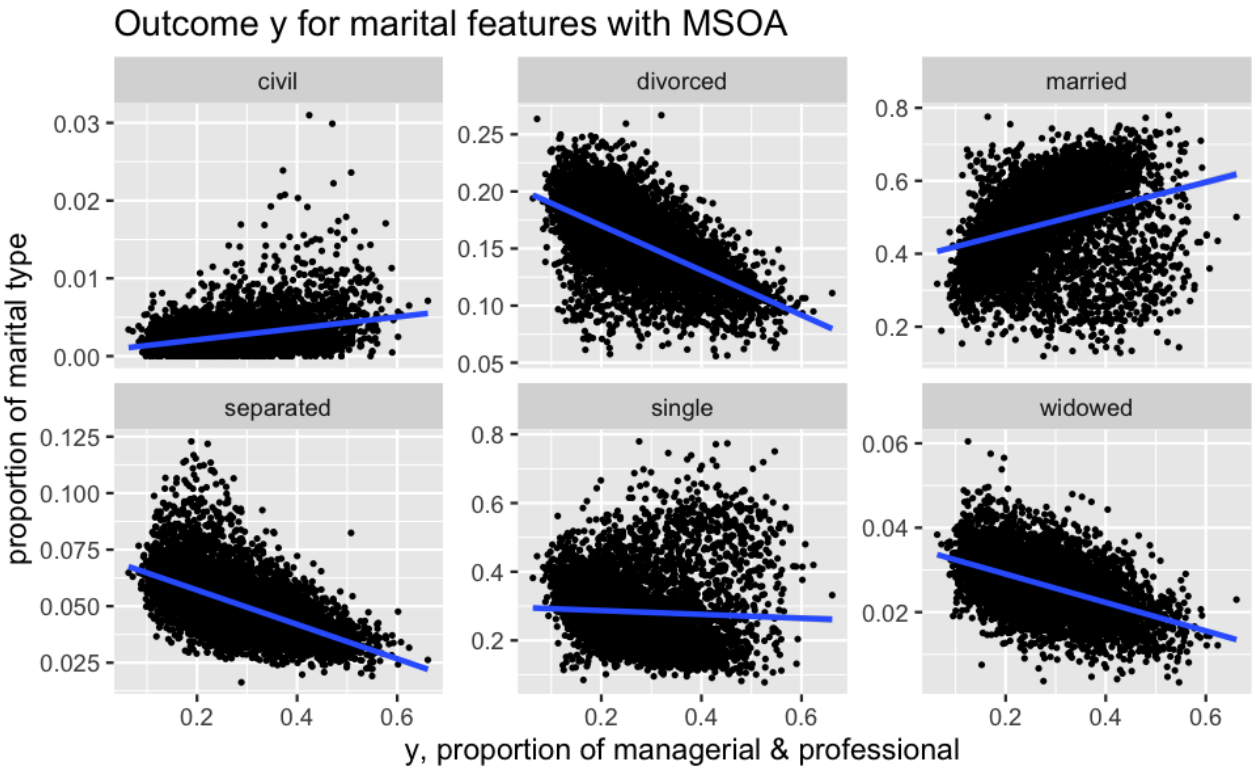
7.1.4 Marital status predictors

For Marital predictors, civil, separated and widowed have quite low values, and married and single both have a large spread, with married positively correlated. Divorced is also clearly negative.

```
#marital features soa
load('rda/marital.rda')
marital_features <- names(marital) %>% str_subset("25_64")
train_soa_final %>%  select(y, all_of(marital_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for marital features with SOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of marital type")
```




```
#marital features msoa
load("rda/train_set_final.rda")
train_set_final %>%  select(y, all_of(marital_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for marital features with MSOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of marital type")
```



7.2 Modelling without qualification features

As qualifications are so important to the models, in particular the Level4, it may be possible to identify other important features masked by the qualification features by modelling without those features. The GLM model will be used as it was high performing and fast, and provided information on feature importance.

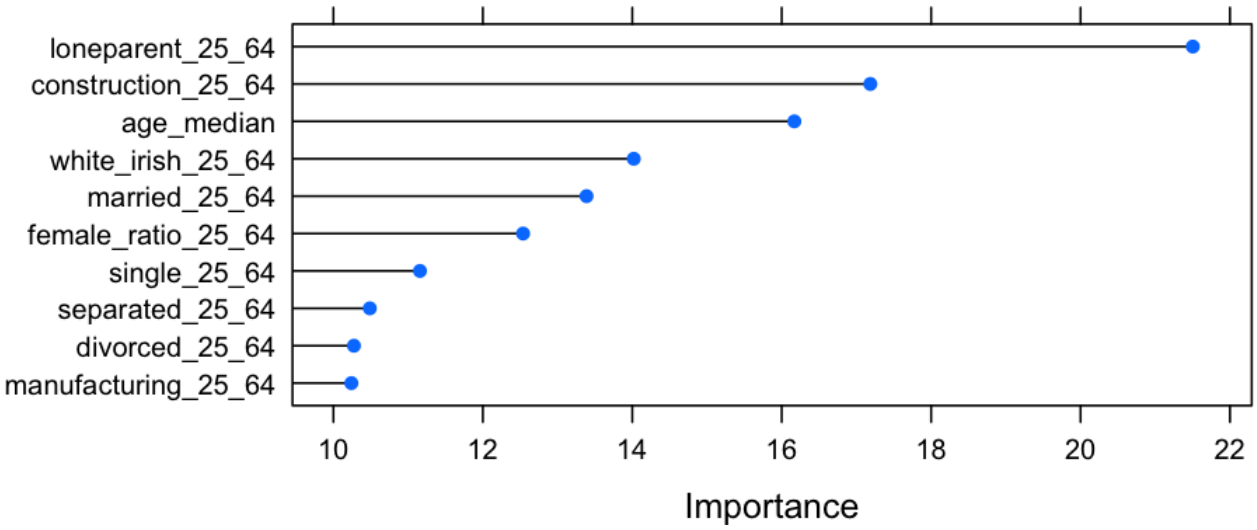
Starting with the MSOA data, removing the qualification features, the RMSE is 0.0314370 for GLM, vs 0.0166809 with the full data, significantly worse, and not so far from the baseline of the mean of 0.0965.

Of the features, loneparent is important, as is construction, both of which had strongly negative correlations. Ethnicities are not highlighted, with only white irish which was in the top list earlier. However, Age and female ratio are in the top 10.

```
## msoa data
# removing the qualification data
# get the names of the features
load('rda/qualifications.rda')
load("rda/test_set_final.rda")
load("rda/rmse_results.rda")
qualifications_features <- names(qualifications) %>% str_subset("25_64")
train_msoa_noqual <- train_set_final %>% select(-all_of(qualifications_features))
# train with the no qualification data set
result_glm_train_msoa_noqual <-
  results_train_method(train_msoa_noqual, test_set_final, "glm")
# extract the rmse from the results
rmse_results <- bind_rows(rmse_results,
                          tail(result_glm_train_msoa_noqual$results, 1))
rmse_results %>% knitr::kable()
```

```
load("rda/result_glm_train_msoa_noqual.rda")
#checking the variable importance
```

```
importance <- varImp(result_glm_train_msoa_noqual$train, scale=FALSE)
plot(importance, 10)
```



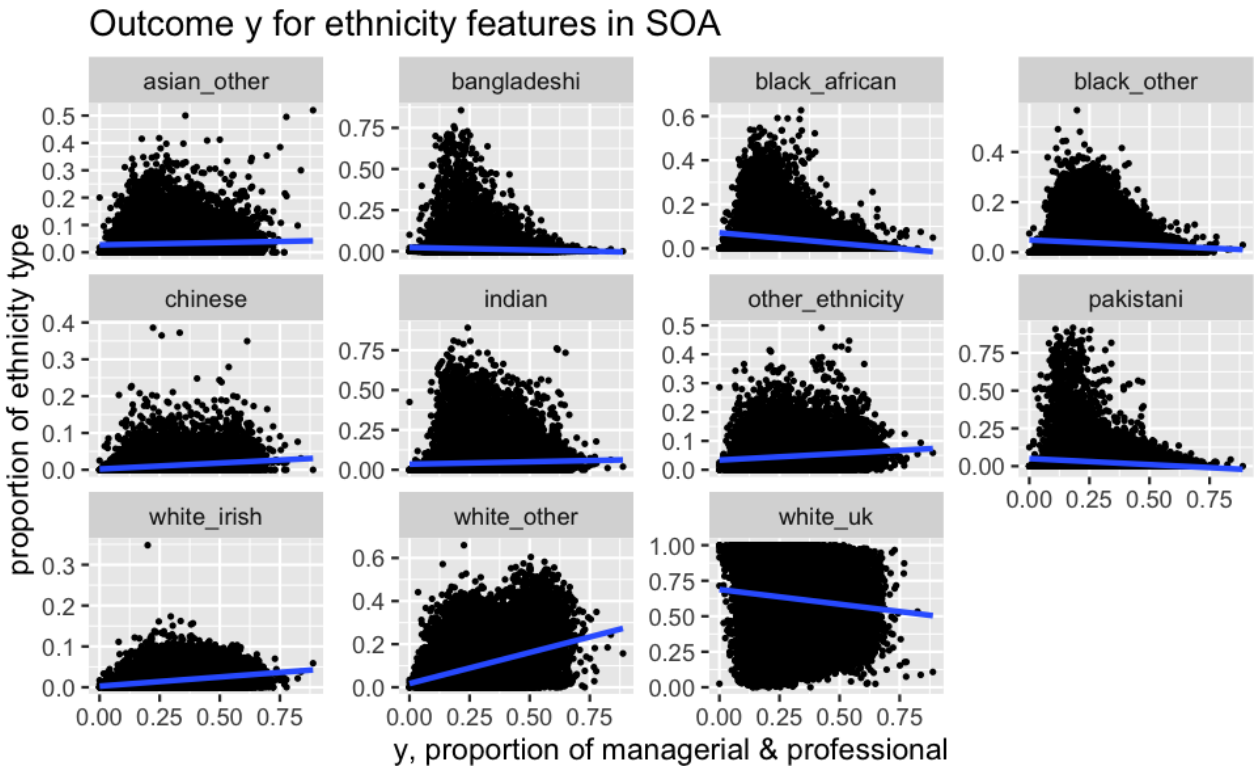
7.3 Ethnicity deep dive

Part of the original motivation for this project was the reported lack of opportunities for certain ethnic groups, and this section investigates ethnic features to see if the features indicate anything similar to the press reports.

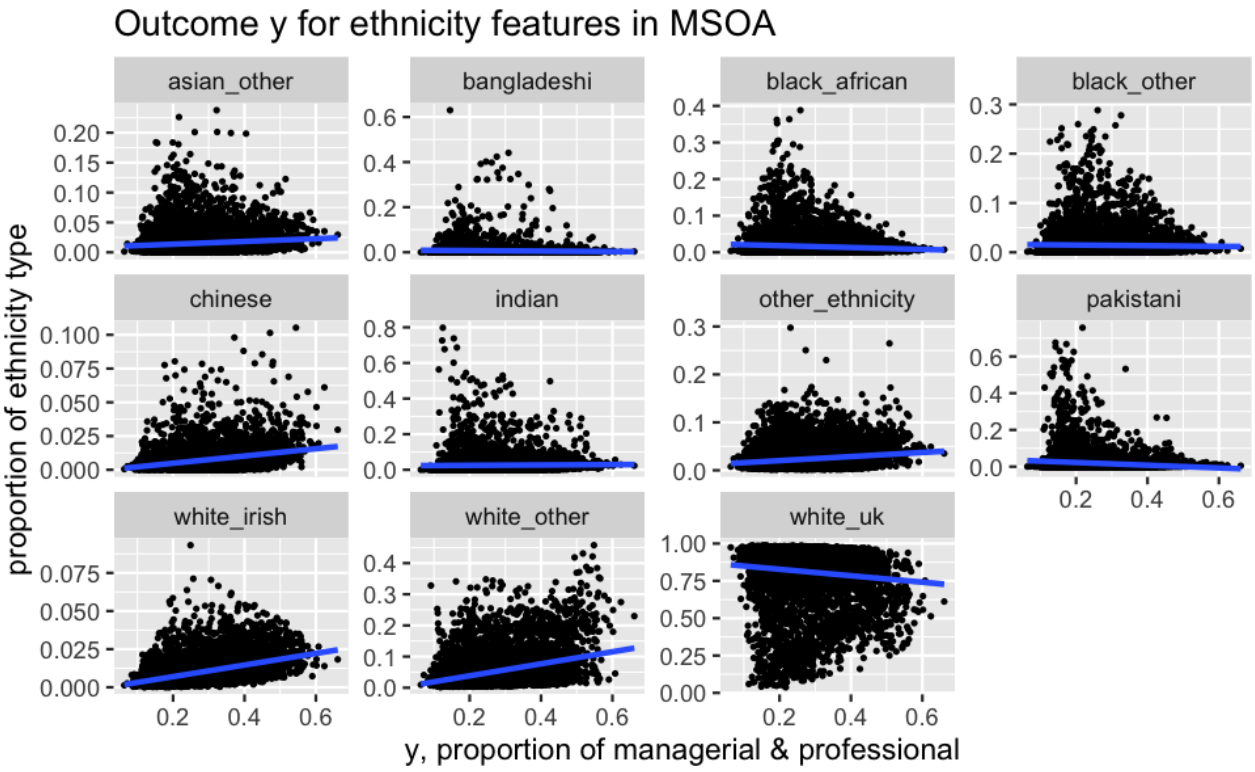
The machine learning model led analysis has not indicated that ethnicities are as significant factors, apart from positive correlation with white Irish and white other.

Starting with the visualisation of the distributions of proportions of ethnicities and y, proportion of managerial and professional roles, the fit lines are fairly flat apart from white other (positive), but there are lots of graphs which show a triangle shape with low proportions of senior roles in areas with high concentrations of that ethnicity, particularly Black african, Black other, Bangladeshi and Pakistani. These are the communities often highlighted in the press as having poor outcomes, backed up by the data here.

```
#ethnicity features soa
load('rda/ethnicity.rda')
ethnicity_features <- names(ethnicity) %>% str_subset("25_64")
train_soa_final %>%
  #select only the ethnicity features
  select(y, all_of(ethnicity_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  # str_replace_all(., "_25_64", "") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for ethnicity features in SOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of ethnicity type")
```




```
#ethnicity features msoa
load("rda/train_set_final.rda")
train_set_final %>%
  #select only the ethnicity features
  select(y, all_of(ethnicity_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -y,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(y, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for ethnicity features in MSOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of ethnicity type")
```



Looking at the correlations, and the p values, this matches the images, with white other, white Irish, Chinese and other positive, and Black African, Pakistani negative. The p-value for all for the SOA data is 0, indicating that there probability of these results given no relationship is low - there is an effect. For the MSOA, the p value is non-zero for Indian, black other and Bangladeshi, indicating a weaker effect, especially for Indian.

```
# correlation with p value
# uses the Hmisc package
if (!require('Hmisc')) install.packages('Hmisc'); library('Hmisc')
# extract the ethnicity features for SOA
# calculate the correlation matrix and p value matrix
train_soa_rcorr <- rcorr(as.matrix(train_soa_final))
# extract the p-value result for y
train_soa_ethnicity_p <-
  data.frame(train_soa_rcorr$P) %>%
  # select the y column
  select(y) %>%
  rownames_to_column(var="feature") %>%
  # choose only the ethnicity features
  filter(feature %in% ethnicity_features) %>%
  rename("p_value" = "y")
# extract correlation, and display with p-value result
data.frame(train_soa_rcorr$r) %>%
  # select the y column
  select(y) %>%
  rownames_to_column(var="feature") %>%
  # choose only the ethnicity features
  filter(feature %in% ethnicity_features) %>%
  rename("y_correlation_coefficient" = "y") %>%
  #include the p_value
  left_join(train_soa_ethnicity_p) %>%
```

```
  # reorder and display
  arrange(desc(abs(y_correlation_coefficient))) %>%
  knitr::kable(caption = "Correlation and P results for ethnicity and y in SOA")
```

Table 56: Correlation and P results for ethnicity and y in SOA

feature	y_correlation_coefficient	p_value
white_other_25_64	0.398507	0
white_irish_25_64	0.344061	0
chinese_25_64	0.238275	0
black_african_25_64	-0.197096	0
pakistani_25_64	-0.145979	0
other_ethnicity_25_64	0.133445	0
black_other_25_64	-0.109491	0
white_uk_25_64	-0.096776	0
bangladeshi_25_64	-0.084088	0
asian_other_25_64	0.053840	0
indian_25_64	0.048998	0

```
# extract the ethnicity features for MSOA
# calculate the correlation matrix and p valuer matrix
train_msoa_rcorr <- rcorr(as.matrix(train_set_final))
# extract the p-value result for y
train_msoa_ethnicity_p <-
  data.frame(train_msoa_rcorr$P) %>%
  # select the y column
  select(y) %>%
  rownames_to_column(var="feature") %>%
  # choose only the ethnicity features
  filter(feature %in% ethnicity_features) %>%
  rename("p_value" = "y")
# extract correlation, and display with p-value result
data.frame(train_msoa_rcorr$r) %>%
  # select the y column
  select(y) %>%
  rownames_to_column(var="feature") %>%
  # choose only the ethnicity features
  filter(feature %in% ethnicity_features) %>%
  rename("y_correlation_coefficient" = "y") %>%
  #include the p_value
  left_join(train_msoa_ethnicity_p) %>%
  # reorder and display
  arrange(desc(abs(y_correlation_coefficient))) %>%
  knitr::kable(caption = "Correlation and P results for ethnicity and y in MSOA")
```

Table 57: Correlation and P results for ethnicity and y in MSOA

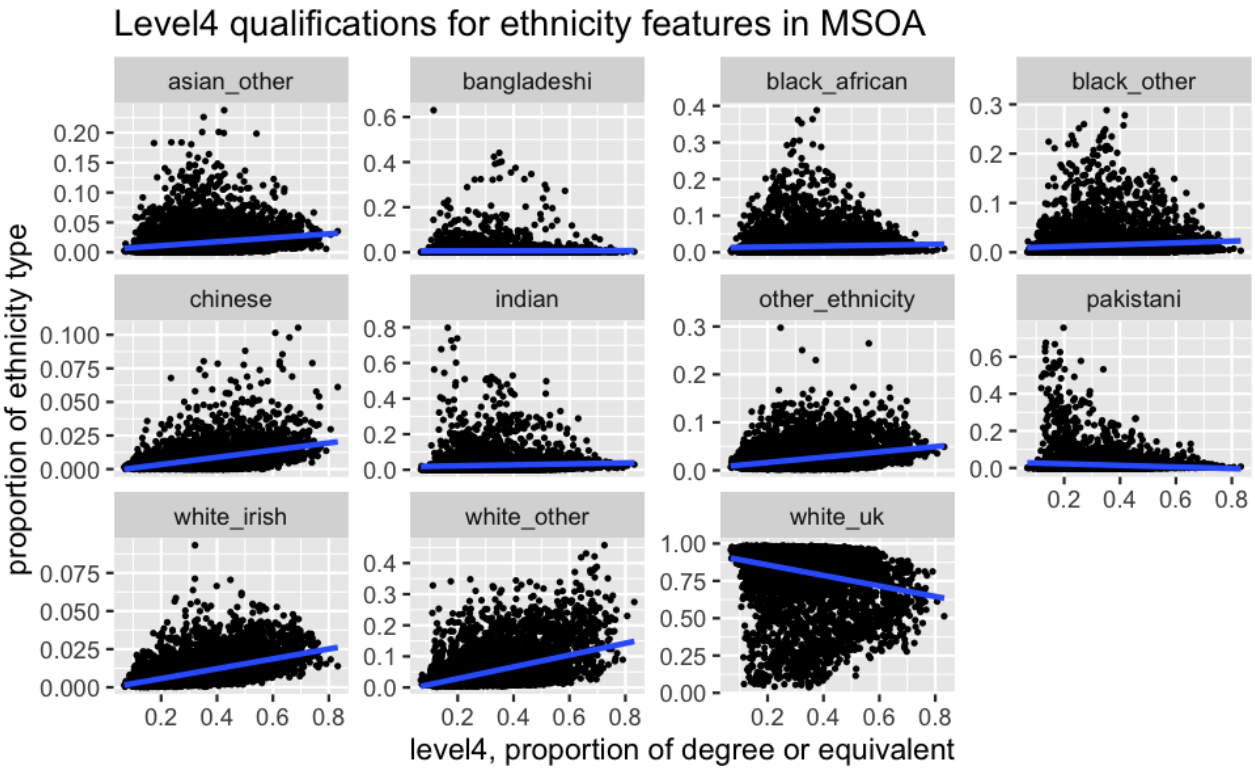
feature	y_correlation_coefficient	p_value
white_irish_25_64	0.417908	0.000000
white_other_25_64	0.317770	0.000000
chinese_25_64	0.298652	0.000000
other_ethnicity_25_64	0.158436	0.000000

feature	y_correlation_coefficient	p_value
pakistani_25_64	-0.129522	0.000000
asian_other_25_64	0.098927	0.000000
white_uk_25_64	-0.098790	0.000000
black_african_25_64	-0.073781	0.000000
bangladeshi_25_64	-0.037605	0.004077
black_other_25_64	-0.021688	0.097698
indian_25_64	0.016854	0.198112

Carrying out a similar analysis against the level4 qualifications, as expected the results are similar to the more senior managerial and professional occupations, with white other, white Irish and Chinese positive, and white UK and Pakistani negative.

```
#ethnicity features vs level4 soa
load('rda/ethnicity.rda')
ethnicity_features <- names(ethnicity) %>% str_subset("25_64")
train_soa_final %>%
  #select only the ethnicity features
  select(level4_25_64, all_of(ethnicity_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # rename(str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -level4,
               names_to = "feature",
               values_to = "proportion") %>%
  # str_replace_all(., "_25_64", "") %>%
  ggplot(aes(level4, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Level4 qualifications for ethnicity features in SOA") +
  xlab("level4, proportion of degree or equivalent") +
  ylab("proportion of ethnicity type")
```


```
#ethnicity features msoa
load("rda/train_set_final.rda")
train_set_final %>%
  #select only the ethnicity features
  select(level4_25_64, all_of(ethnicity_features)) %>%
  #tidy the names, removing the _25_64
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  # pivot longer with the features
  pivot_longer(cols = -level4,
               names_to = "feature",
               values_to = "proportion") %>%
  ggplot(aes(level4, proportion))  +
  facet_wrap(.~feature, scales = "free_y") +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Level4 qualifications for ethnicity features in MSOA") +
  xlab("level4, proportion of degree or equivalent") +
  ylab("proportion of ethnicity type")
```



However, there are differences, with Black African and Black Other significantly worse for their achievements in senior occupations than their educational achievements, this is more clear looking at the values, with correlation between areas with more Black African residents and Level4 qualification as -0.089220, vs -0.197096 for the senior occupations for SOA; for Black other residents, the level4 qualification has a correlation of -0.013246, whereas the occupation has a correlation of -0.109491 for SOA, with similar results in the MSOA, with a roughly -0.1 between the qualification and occupation.

```
# extract the ethnicity features for SOA
# extract the p-value result for level4
train_soa_level4_p <-
  data.frame(train_soa_rcorr$P) %>%
  # select the level4 columns
  select(level4_25_64) %>%
  rownames_to_column(var="feature") %>%
  # choose only the ethnicity features
  filter(feature %in% ethnicity_features) %>%
  rename("p_value" = "level4_25_64")
# extract correlation, and display with p-value result
data.frame(train_soa_rcorr$r) %>%
  # select the level4 columns
  select(level4_25_64) %>%
  rownames_to_column(var="feature") %>%
  # choose only the ethnicity features
  filter(feature %in% ethnicity_features) %>%
  rename("level4_correlation_coefficient" = "level4_25_64") %>%
  #include the p_value
  left_join(train_soa_level4_p) %>%
  # reorder and display
  arrange(desc(abs(level4_correlation_coefficient))) %>%
  knitr::kable(caption = "Correlation and P results for ethnicity and level4 in SOA")
```

Table 58: Correlation and P results for ethnicity and level4 in SOA

feature	level4_correlation_coefficient	p_value
white_other_25_64	0.511573	0.000000
white_irish_25_64	0.399894	0.000000
chinese_25_64	0.321996	0.000000
white_uk_25_64	-0.229121	0.000000
other_ethnicity_25_64	0.225433	0.000000
pakistani_25_64	-0.136642	0.000000
asian_other_25_64	0.125259	0.000000
black_african_25_64	-0.089220	0.000000
indian_25_64	0.085549	0.000000
bangladeshi_25_64	-0.054020	0.000000
black_other_25_64	-0.013246	0.014183

```r
# extract the ethnicity features for MSOA
# extract the p-value result for level4
train_msoa_level4_p <-
  data.frame(train_msoa_rcorr$P) %>%
  # select the level4 columns
  select(level4_25_64) %>%
  rownames_to_column(var="feature") %>%
  # choose only the ethnicity features
  filter(feature %in% ethnicity_features) %>%
  rename("p_value" = "level4_25_64")
# extract correlation, and display with p-value result
data.frame(train_msoa_rcorr$r) %>%
  # select the level4 columns
  select(level4_25_64) %>%
  rownames_to_column(var="feature") %>%
  # choose only the ethnicity features
  filter(feature %in% ethnicity_features) %>%
  rename("level4_correlation_coefficient" = "level4_25_64") %>%
  #include the p_value
  left_join(train_msoa_level4_p) %>%
  # reorder and display
  arrange(desc(abs(level4_correlation_coefficient))) %>%
  knitr::kable(caption = "Correlation and P results for ethnicity and level4 in MSOA")
```

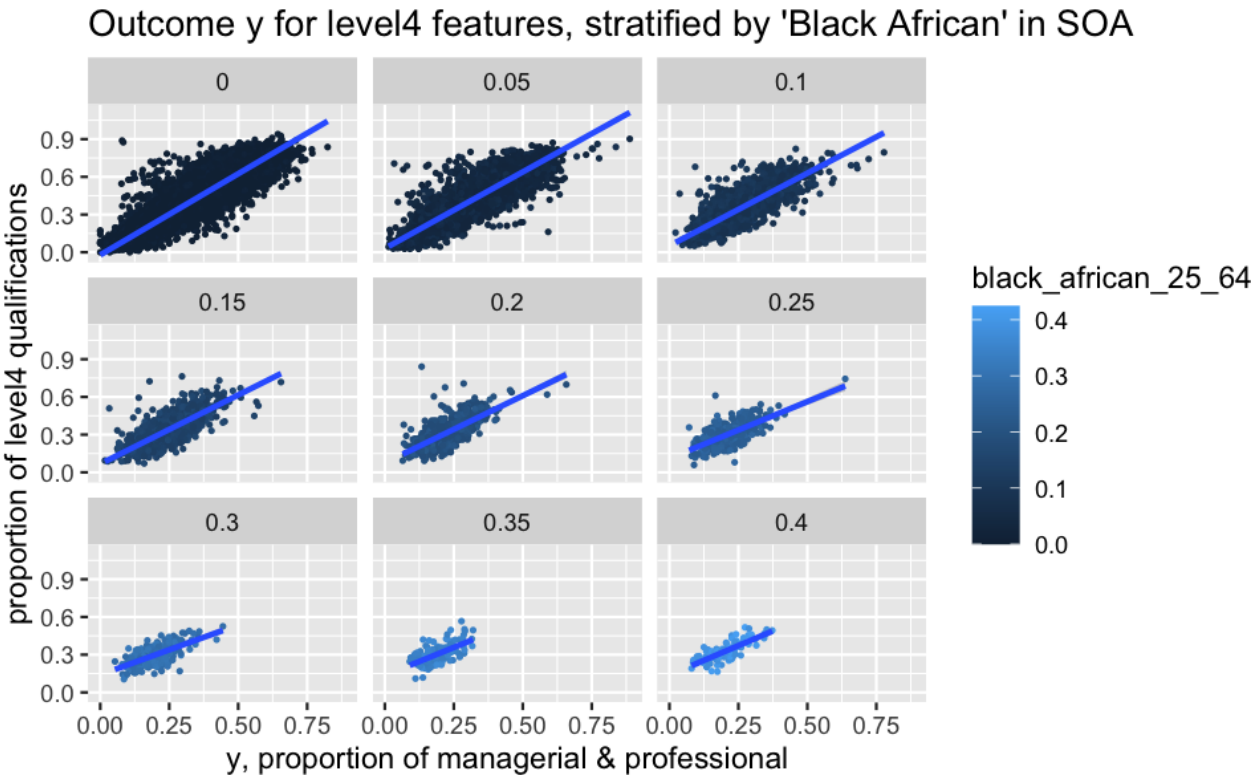
Table 59: Correlation and P results for ethnicity and level4 in MSOA

feature	level4_correlation_coefficient	p_value
white_irish_25_64	0.492428	0.000000
white_other_25_64	0.433157	0.000000
chinese_25_64	0.411622	0.000000
other_ethnicity_25_64	0.284806	0.000000
white_uk_25_64	-0.222841	0.000000
asian_other_25_64	0.197238	0.000000
pakistani_25_64	-0.103375	0.000000
black_other_25_64	0.081395	0.000000
indian_25_64	0.063336	0.000001
black_african_25_64	0.047700	0.000268

feature	level4_correlation_coefficient	p_value
bangladeshi_25_64	0.008269	0.527784

Stratifying the data can explore how the relationship between qualifications and senior occupations varies with the proportion of black african and black other residents. Limiting stratas to those with at least 50 samples, and using SOA only as there is more data, it is evident that the relation between occupation and qualification weakens with more Black African residents.

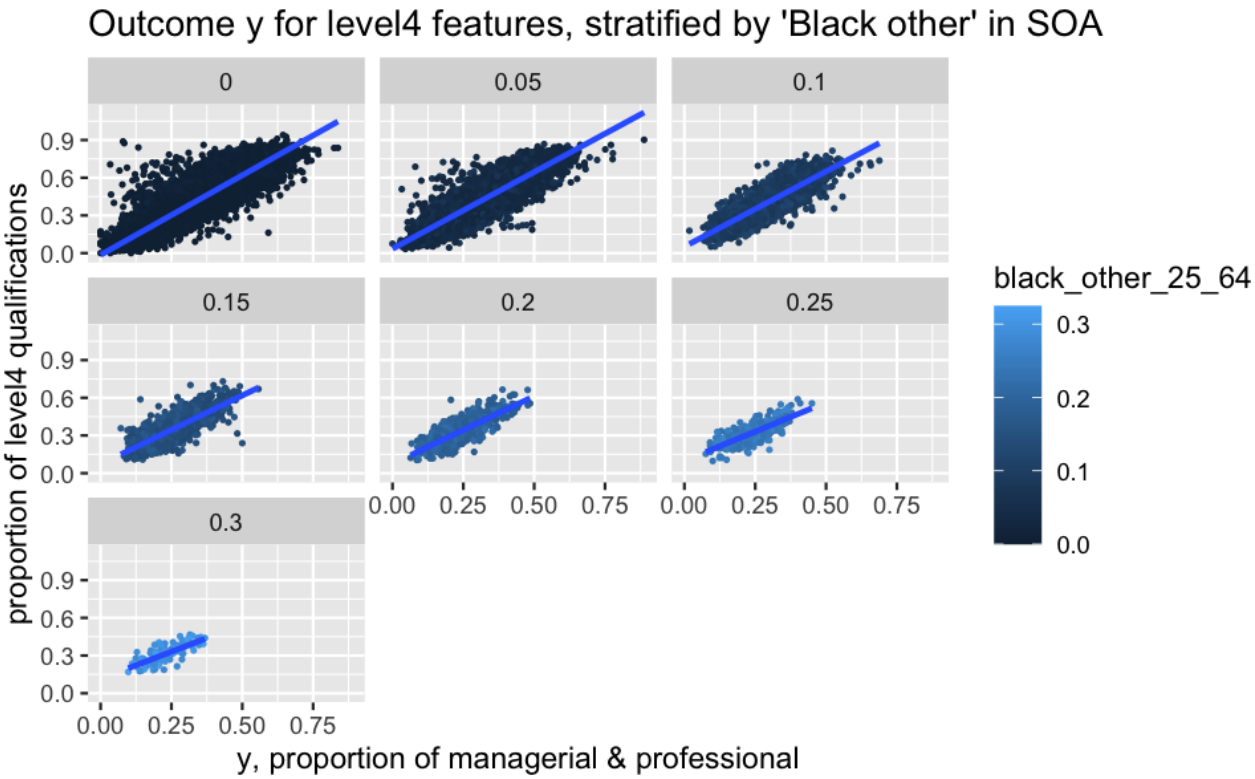
```
# looking at stratification for the black african, soa
train_soa_final %>%
  # select columns
  select(y, level4_25_64, black_african_25_64) %>%
  # round to nearest 0.5
  mutate(black_african_strata = (round(2*black_african_25_64, 1))/2) %>%
  # filter to those strata with more than 50 samples
  filter(black_african_strata <= 0.4) %>%
  ggplot(aes(y, level4_25_64, col=black_african_25_64))  +
  facet_wrap(.~black_african_strata) +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for level4 features, stratified by 'Black African' in SOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of level4 qualifications")
```



The results with Black other show a similar trend.

```
# looking at stratification for the black other, soa
train_soa_final %>%
  # select columns
```

```
  select(y, level4_25_64, black_other_25_64) %>%
  # round to nearest 0.5
  mutate(black_other_strata = (round(2*black_other_25_64, 1))/2) %>%
  # filter to those strata with more than 50 samples
  filter(black_other_strata <= 0.3) %>%
  ggplot(aes(y, level4_25_64, col=black_other_25_64))  +
  facet_wrap(.~black_other_strata) +
  geom_point(size=0.5) +
  geom_smooth(method = "lm") +
  ggtitle("Outcome y for level4 features, stratified by 'Black other' in SOA") +
  xlab("y, proportion of managerial & professional") +
  ylab("proportion of level4 qualifications")
```



Putting together a graph of all the ethnicities and their correlations, it is a confused picture, in part as the data does not have sufficient samples of areas for each strata, and also as the results are for areas with different proportions of each ethinicity, so we are unable to pick out the achievement of the residents of each ethinicity separately.

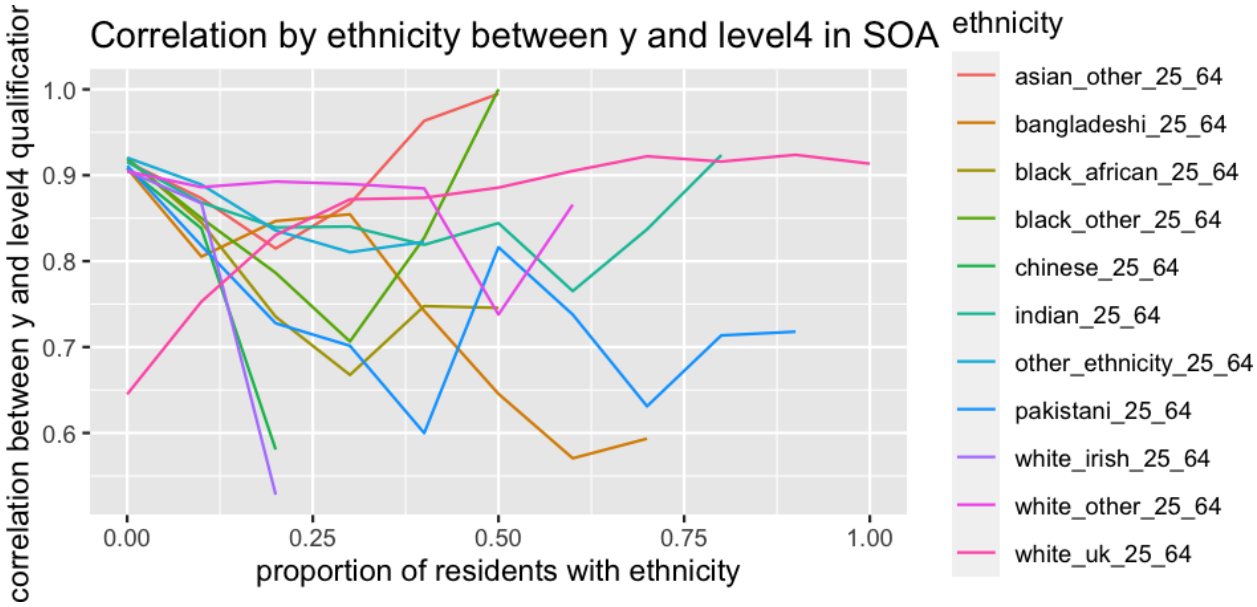
However, the overall correlation between level4 qualifications and the managerial and professional roles is 0.905, and this is only achieved in areas with white uk, white other, Asian other, and maybe Indian. It is clear that when there are more non-white ethnicities, the correlation between qualification and role drops off, perhaps indicating that even with similar high achievement in qualifications it is harder for non-white ethnicities to obtain the best roles.

```
# graphing the correlation for each ethnicity
# create a list of a correlation table for each ethnicity
eth_corr_list <- lapply(ethnicity_features, function(ethnicity){
  print(paste("working with", ethnicity, "feature"))
  tmp <- train_soa_final %>%
    # round to nearest 0.5
```

```
    mutate(strata = round(.data[[ethnicity]], 1)) %>%
    # filter to those strata with more than 50 samples
    filter(strata <= 1) %>%
    # group the data by strata
    group_by(strata) %>%
    dplyr::summarise(correlation = cor(y, level4_25_64))
  colnames(tmp) <- c("strata", ethnicity)
  tmp
})
```

```
## [1] "working with white_uk_25_64 feature"
## [1] "working with white_other_25_64 feature"
## [1] "working with white_irish_25_64 feature"
## [1] "working with indian_25_64 feature"
## [1] "working with pakistani_25_64 feature"
## [1] "working with bangladeshi_25_64 feature"
## [1] "working with other_ethnicity_25_64 feature"
## [1] "working with chinese_25_64 feature"
## [1] "working with asian_other_25_64 feature"
## [1] "working with black_other_25_64 feature"
## [1] "working with black_african_25_64 feature"
```

```
# create starter data frame
eth_corr <- data.frame(eth_corr_list[1])
# loop around and add each table to the data frame
for(i in 2:length(eth_corr_list)) {
  eth_corr <-
    eth_corr %>%
    left_join(data.frame(eth_corr_list[i]))
}
# graph the proportions
eth_corr %>%
  pivot_longer(cols = -strata,
               names_to = "ethnicity",
               values_to = "correlation") %>%
  filter(correlation >= 0.4) %>%
  ggplot(aes(strata, correlation, col=ethnicity))  +
  geom_line() +
  ggtitle("Correlation by ethnicity between y and level4 in SOA") +
  xlab("proportion of residents with ethnicity") +
  ylab("correlation between y and level4 qualifications")
```



Lastly it is possible to display the overall results in England and Wales (the detail is not available for the UK) for senior managerial and professional occupations and level 4 qualifications, with their ratio by ethnicity.

There are two tables with the necessary data:

- load in the Highest level of qualification by ethnic group by age
  - https://www.nomisweb.co.uk/census/2011/dc5202ew
- load in the Occupation by ethnic group by sex by age data
  - https://www.nomisweb.co.uk/census/2011/dc6213ew

Unfortunately one of them only has summarised data, which has a single figure for Asian, so masking the differences between Indian/Chinese, and Pakistani/Bangladeshi achievements seen in the earlier graphs.

The graph shows that the only group whose qualifications are below average are white UK residents. However, in terms of senior occupations Black, mixed and other do worse than would be expected. This is exacerbated by the fact that their qualifications are high compared to the normal, and therefore given that qualification is the most important factor (or machine learning feature) then this is doubly poor achievement in terms of occupations.

White Irish are also striking in terms of the qualifications and the seniority of their occupations.

As mentioned, the Asian group includes Chinese and Indian groups which are regularly reported as high performing academically and occupationally, as well as Pakistani and Bangladeshi which are often highlighted as struggling. It is likely that separated in to two, the sub Asian groups would diverge in terms of both qualifications and occupations.

```
# look at ratios of ethnicity and occupation and qualification in countries in the UK
## download the data
#  <option value="TYPE499">countries</option>
# load in the Highest level of qualification by ethnic group by age
# https://www.nomisweb.co.uk/census/2011/dc5202ew
# load in the Occupation by ethnic group by sex by age data
# https://www.nomisweb.co.uk/census/2011/dc6213ew
censusdata_list <- c("dc5202ew", "dc6213ew")
# run the function to download the csv files
download_censusdata("TYPE499", censusdata_list)

# load in the Highest level of qualification by ethnic group by age
```

```r
# https://www.nomisweb.co.uk/census/2011/dc5202ew
geographytype <- "TYPE499"
data <- ingest("dc5202ew", geographytype)
str(data)
# put together a table with the level4 qualifications by ethnicity
ethnicity_level4_25_64_raw <- data %>%
  #keeping only the data relating to age 25 to 64
  select(contains("geography") |
           contains("Age 25 to 34") |
           contains("Age 35 to 49") |
           contains("Age 50 to 64") ) %>%
  #keeping only the data relating to Highest Level of Qualification: Level 4 qualifications
  select(contains("geography") | contains("Highest Level of Qualification: Level 4 qualifications") ) %>
  # adding up the year groups
  rowwise() %>%
  mutate("all_25_64" =
           sum(across(contains("Ethnic Group: All categories"))) ) %>%
  mutate("white_uk_25_64" =
           sum(across(contains("Ethnic Group: White: English/Welsh/Scottish/Northern Irish/British"))) %
    mutate("white_irish_25_64" =
             sum(across(contains("Ethnic Group: White: Irish"))) ) %>%
    mutate("white_other_25_64" =
           sum(across(contains("Ethnic Group: White: Other White"))) ) %>%
    mutate("mixed_25_64" =
             sum(across(contains("Ethnic Group: Mixed/multiple ethnic group"))) ) %>%
    mutate("asian_25_64" =
           sum(across(contains("Ethnic Group: Asian/Asian British"))) ) %>%
    mutate("black_25_64" =
             sum(across(contains("Ethnic Group: Black/African/Caribbean/Black British"))) ) %>%
    mutate("other_25_64" =
           sum(across(contains("Ethnic Group: Other ethnic group"))) ) %>%
  rename("geo_name" = "geography") %>%
  select(contains("geo_name") | contains("25_64")) %>%
  filter(geo_name == "England and Wales")

# put together a table with all the 25 to 64 residents with the same data
ethnicity_all_25_64_raw <- data %>%
  #keeping only the data relating to age 25 to 64
  select(contains("geography") |
           contains("Age 25 to 34") |
           contains("Age 35 to 49") |
           contains("Age 50 to 64") ) %>%
  #keeping only the data relating to Highest Level of Qualification: All, to give all residents 25 to 6
  select(contains("geography") |
           contains("Highest Level of Qualification: All categories:") ) %>%
  # adding up the year groups
  rowwise() %>%
  mutate("all_25_64" =
           sum(across(contains("Ethnic Group: All categories"))) ) %>%
  mutate("white_uk_25_64" =
           sum(across(contains("Ethnic Group: White: English/Welsh/Scottish/Northern Irish/British"))) %
  mutate("white_irish_25_64" =
           sum(across(contains("Ethnic Group: White: Irish"))) ) %>%
```

```
  mutate("white_other_25_64" =
           sum(across(contains("Ethnic Group: White: Other White"))) ) %>%
  mutate("mixed_25_64" =
           sum(across(contains("Ethnic Group: Mixed/multiple ethnic group"))) ) %>%
  mutate("asian_25_64" =
           sum(across(contains("Ethnic Group: Asian/Asian British"))) ) %>%
  mutate("black_25_64" =
           sum(across(contains("Ethnic Group: Black/African/Caribbean/Black British"))) ) %>%
  mutate("other_25_64" =
           sum(across(contains("Ethnic Group: Other ethnic group"))) ) %>%
  rename("geo_name" = "geography") %>%
  select(contains("geo_name") | contains("25_64")) %>%
  filter(geo_name == "England and Wales")

# load in the Occupation by ethnic group by sex by age data
# https://www.nomisweb.co.uk/census/2011/dc6213ew
geographytype <- "TYPE499"
data <- ingest("dc6213ew", geographytype)
str(data) %>% head(50)

# put together a table with the level4 qualifications by ethnicity
ethnicity_y_25_64_raw <- data %>%
  #keeping only the data relating to all (not by gender)
  select(contains("geography") |
           contains("Sex: All persons") ) %>%
  #keeping only the data relating to age 25 to 64
  select(contains("geography") |
           contains("Age 25 to 49") |
           contains("Age 50 to 64") ) %>%
  #keeping only the data relating to Highest Level of Qualification: All, to give all residents 25 to 6
  select(contains("geography") |
           contains("Occupation: 1. Managers") |
           contains("Occupation: 2. Professional") ) %>%
  # adding up the year groups
  rowwise() %>%
  mutate("all_25_64" =
           sum(across(contains("Ethnic Group: All categories"))) ) %>%
  mutate("white_uk_25_64" =
           sum(across(contains("Ethnic Group: White: English/Welsh/Scottish/Northern Irish/British"))) )
  mutate("white_irish_25_64" =
           sum(across(contains("Ethnic Group: White: Irish"))) ) %>%
  mutate("white_other_25_64" =
           sum(across(contains("Ethnic Group: White: Other White"))) +
           sum(across(contains("Ethnic Group: White: Gypsy or Irish Traveller"))) ) %>%
  mutate("mixed_25_64" =
           sum(across(contains("Ethnic Group: Mixed/multiple ethnic group: Total"))) ) %>%
  mutate("asian_25_64" =
           sum(across(contains("Ethnic Group: Asian/Asian British: Total"))) ) %>%
  mutate("black_25_64" =
           sum(across(contains("Ethnic Group: Black/African/Caribbean/Black British: Total"))) ) %>%
  mutate("other_25_64" =
           sum(across(contains("Ethnic Group: Other ethnic group: Total"))) ) %>%
  rename("geo_name" = "geography") %>%
```

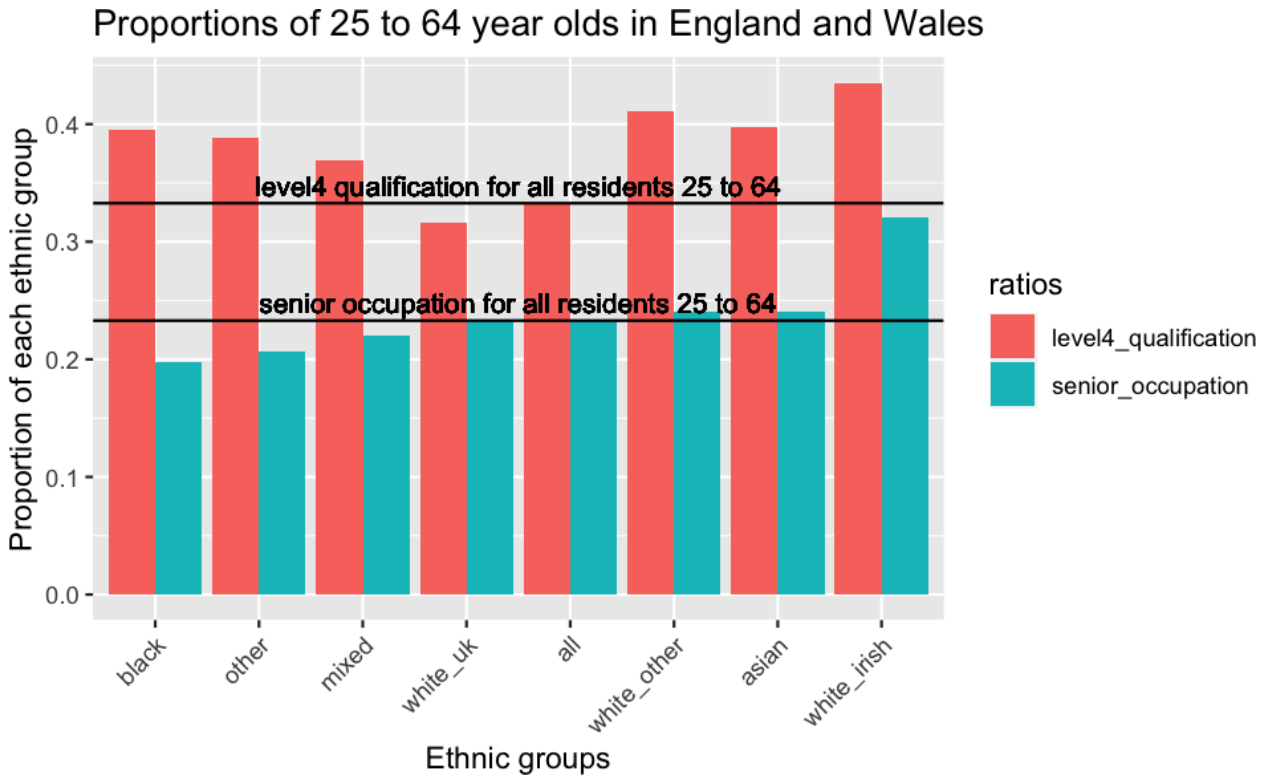
```
  select(contains("geo_name") | contains("25_64")) %>%
  filter(geo_name == "England and Wales")

# join together
# pivot to long form
ethnicity_y_25_64_Long <- ethnicity_y_25_64_raw %>%
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  pivot_longer(cols = -geo_name,
               names_to = "ethnicity",
               values_to = "occupation_25_64")
ethnicity_level4_25_64_long <- ethnicity_level4_25_64_raw %>%
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  pivot_longer(cols = -geo_name,
               names_to = "ethnicity",
               values_to = "qualification_25_64")
# join in to a single table
ethnicity_level4_25_64_long
ethnicity_25_64_raw <- ethnicity_all_25_64_raw %>%
  rename_at(vars(contains("_25_64")), ~str_replace_all(., "_25_64", "")) %>%
  pivot_longer(cols = -geo_name,
               names_to = "ethnicity",
               values_to = "all_25_64") %>%
  left_join(ethnicity_y_25_64_Long) %>%
  left_join(ethnicity_level4_25_64_long)

# tidy
save(ethnicity_25_64_raw, file="rda/ethnicity_25_64_raw.rda")
```

```
# graph
load("rda/ethnicity_25_64_raw.rda")
# calculate the ratio of the working population for each ethnicity
ethnicity_25_64 <- ethnicity_25_64_raw %>%
  mutate(senior_occupation = occupation_25_64 / all_25_64) %>%
  mutate(level4_qualification  = qualification_25_64 / all_25_64) %>%
  select(ethnicity, senior_occupation, level4_qualification)
# put together a list in order based on the senior_occupation value
ethnicity_graph_order <- ethnicity_25_64 %>%
  arrange(senior_occupation) %>% .$ethnicity
# ethnicity_25_64
# plot the graph
ethnicity_25_64 %>%
  # pivot longer to include the proportions in one columns
  pivot_longer(cols = -ethnicity,
               names_to = "ratios",
               values_to = "population_proportion") %>%
  ggplot(aes(ethnicity, population_proportion, fill=ratios)) +
  geom_bar(stat="identity", position="dodge") +
  geom_hline(aes(yintercept=ethnicity_25_64$senior_occupation[1])) +
  geom_text(aes(4.5, ethnicity_25_64$senior_occupation[1],
            label="senior occupation for all residents 25 to 64"),
           nudge_y=0.015, size = 3.5) +
  geom_hline(yintercept=ethnicity_25_64$level4_qualification[1]) +
  geom_text(aes(4.5, ethnicity_25_64$level4_qualification[1],
              label="level4 qualification for all residents 25 to 64"),
```

```
            nudge_y=0.015, size = 3.5) +
# using the list earlier for the order on the x axis
  scale_x_discrete(limits=ethnicity_graph_order) +
  theme(legend.position = "right") +
  theme(axis.text.x = element_text(angle = 45, vjust=1, hjust=1)) +
  ggtitle("Proportions of 25 to 64 year olds in England and Wales") +
  xlab("Ethnic groups") +
  ylab("Proportion of each ethnic group")
```



8 Results

8.1 MSOA areas

The optimal result for the MSOA data was from an ensemble of two models SVM Radial, and GAM, where the GAM model needed a reduction of numbers of features to 28 in order to complete the calculations.

Table 60: Summary of modelling results for MSOA areas

method	rmse
Mean on main & validation	0.091347
"svmRadial" - main_clean	0.015654
"gam" - train_top28	0.015931
Ensemble svmRadial, gam28	0.014253

Many models appear to struggle with a large number of features, but on the other hand, the more features the better the model performance for the leading models, and therefore the models should include as many features as possible.

The best models were linear regression, and variants of vector machines / gradient boosting. The latter seemed to work quite well, but computationally expensive.

The models show that the achievement of senior management and professional occupations is closely linked to the level4 qualification, which must be why linear regression models perform well.

8.2 SOA areas

The results for the SOA areas London and Yorkshire and Humber were not as good, although a larger improvement over the baseline of the mean of all locations. This reflects the greater variation of the observations.

Table 61: Summary of modelling results for MSOA areas

method	rmse
Mean of all locations	0.1334185
"svmRadial" - train_soa_15k_top28	0.0438095
"gbm" - train_soa_final	0.0443612
Ensemble svmRadial, gbm	0.0431886

9 Conclusion

This project developed a new data set from raw UK census data, which was then cleaned and prepared for machine learning to answer the following challenge:

- predict the proportion of working age people each area that are one of the managerial and professional occupations

The modelling was carried out by applying many different algorithms to focus down on the most effective ones, and with this building an ensemble.

In addition the project carried out detailed visualisations of the features, the important features and correlations to try and understand the drivers the for achievement of more senior roles.

9.1 Use of the census data

The data set worked well as a continuous machine learning problem, it behaved well, with sensible results and a variation in different results from different algorithms. The data put treated in this way also allowed for some interesting conclusions to be visualised from the features.

There were too many features, and some could probably be removed with little impact, but the imapct of removing the least 10 or so important features was not specifically tested.

The modelling with the MSOA, and the quick attempt at the SOA indicates that there is potential for a very large data set of 130,000 observations of 50 features, over 6 million pieces of publicly available data. This may be of interest to test algorithms for working with large numbers of observations and features.

The modelling of the SOA areas only used 2 regions, covering over 42 thousand observations out of the over 130 thousand. This was useful as a quick test of the process and algorithms, but it could be been extended to the full data set, particularly for the visualisation and attempts at understanding the implications.

9.2 Modelling conclusions

Overall the proportion of people with senior roles is quite predictable given the features provided. The root mean error of 0.013 is small given the range of 0.06 to 0.66, and even the less well performing SOA areas achieved 0.043 with a larger range of 0 to 0.887.

Table 62: Comparison of y, proportion of senior managers and professional occupations in SOA and MSOA

	train_soa_y	train_msoa_y
mean	0.295717	0.275763
sd	0.134452	0.095860
max	0.887850	0.661216
min	0.000000	0.063305

9.2.1 Model selection

By working with a lot of algorithms and the final ensemble the machine learning analysis was able to make a significant stepwise improvement during the project.

This study focused on the data and features, and to try many models, but each could be tuned, a huge amount of potential detailed investigation and improvement was not carried out.

In general the approach to the model selection was empirical: trying a variety of different types of model and then focusing down on to the ones that worked the best. This seemed to work, although there are a large number of models that have not been tried, maybe there are better ones - the best individual model was the very last tried, which could easily have been missed.

The best models seemed to be the fashionable ones, SVM and GAM, GBM, with the SVM Radial the best for both MSOA and SOA data sets.

9.2.2 Modelling very large data sets

The project only briefly worked with the larger data sets from the SOA to verify that it worked, and there were problems with algorithms being able to handle both large numbers of features and observations.

There will be many different approaches to large data sets, but these were not researched as the focus was the msoa. For example, not all the observations in the training data were used in the final result, and clearly using more data should improve, perhaps an ensemble of subsets of data?

9.2.3 Model interpretation

In general interpretability from the models was not very clear. Many have no indication of feature importance, or ways to get information out, without running many times with different sets of features. Furthermore, many of the popular ones for predictions using boosting or additive processes for iterative improvement which obfuscate the data an implications.

In addition, the investigation at the end appeared to show some implications from the ethnicity features which were not highlighted in model, where less intuitive features like lone parent came out as more important. This may be because there were a lot of related features, so each on its own is less important (such as ethnicity and country of birth), in contrast to age, female ratio, lone parent which are individual. This does make it hard to use machine learning models to make conclusions other than predictions.

The "high correlation" function in caret did not work well at helping feature selection, as some of the listed highly correlated features were the key ones. Using feature importance seemed to work, and would be better for future analysis.

9.3 Implications from the modelling

By far and away the most important element for high quality occupation is qualifications. As this is by area, it is not clear on cause and effect, that is, the highly qualified people all move to where the best jobs are, or that the best jobs are where there are highly qualified people. But the public policy implications are clear, ensuring all groups have access to education and qualifications is critical to ensure no one is left behind. Or another way, if any cohort has poor qualification achievements they will have poor achievements in work.

Beyond that, the conclusions from the models was not clear, other than the place of birth did was not highlighted as an issue at all, ethnicity, households and so on were much more important

There were implications from the features in the data set which were interesting, and align with recent political and press articles:

- there appeared poor achievements for white uk, both in occupations and qualifications. This is in contrast to other white ethnicities, "white Irish" and "white other" both highly qualified and with the best jobs - which could be the source of the anti-EU sentiment, and populism in general with the white UK population.
- poor occupation achievements for "black other" and "black African", despite good qualifications. This is again in contrast to non-UK white ethnicities - which could be the source of the UK variant of the Black Lives Matter movement, which is highlighting the predjudice and lack of opportunities.
- "lone parent" and "divorced" have negative impacts, and this could mean more help should be provided, probably around child care, and this probably impacts women more than men.

9.4 Further work

The following section describes a few potential ways to extend the work, both from a pure modelling and interpretation perspective.

9.4.1 Modelling and algorithms - MSOA

From a modelling perspective, the analysis could be extended by focusing on one or two algorithms and go into depth to optimise and improve the performance - can it outperform an ensemble of non-optimised models?

Also, there are many models which were not tried, another approach would be to move beyond the empirical model selection process and to use logic based on the specific data set to selecting specific models, and to find the best single model, to improve on the SVM Radial.

Finally, given that the best performing models are ones that are regularly highlighted, another alternative approach would be a literature review to identify the most popular algorithms today and focus on those.

9.4.2 Modelling and algorithms - SOA

The project could be extended by focusing on the SOA instead of the MSOA, and investigate working with methodologies to address large data sets, with large numbers of features and observations.

The analysis on numbers of features and observations could be improved, by taking a systematic approach, such as a measure of the computation expense vs RMSE performance, for each step in observations of features, to identify the optimal point. Similarly, computation expense can be used as part of model selection, RMSE improvement per CPU time. This could be particularly helpful to select models using smaller data sets with the intention to be used on the larger data sets.

9.4.3 Societal implications

Firstly, the SOA data looked useful, but it would be better to use the full 130,000 observations for the whole of England and Wales.

This study was to focus on machine learning, but it could be extended and change focus to analyse the statistical implications of the data, the studying correlations, confounding data and statistical significance of the different factors For example, “White other” is positively correlated with the high achievements in occupations, and “Pakistini” and “Bangladeshi” are negatively, and this could be due to education and qualifications (the press mentions the poor school outcomes for some ethnicities, and for working class white UK people), and it would be interesting to investigate causality in terms of drivers.

9.4.4 Further potential implications

https://www.theguardian.com/business/2020/jul/28/bame-representation-uk-top-jobs-colour-of-power-survey "The proportion of black, Asian and minority ethnic people in some of the 1,100 most powerful jobs in the UK has barely moved over the past three years, according to a study that highlights the lack of non-white representation across key roles.

Only 51 out of the 1097 most powerful roles in the country are filled by non-white individuals, an increase of only 1.2%, or 15 people, since 2017, the Colour of Power survey by consultants Green Park and not-for-profit organisation Operation Black Vote said.

That represented 4.7% of the total number compared with the 13% proportion of the UK population." https://thecolourofpower.com/

https://www.theguardian.com/fashion/2020/jul/29/numbers-dont-lie-how-data-reveals-fashions-inclusivity-problem https://www.ethnicity-facts-figures.service.gov.uk/work-pay-and-benefits/employment/employment-by-occupation/latest

https://www.ethnicity-facts-figures.service.gov.uk/education-skills-and-training/after-education/destinations-and-earnings-of-graduates-after-higher-education/latest

---