Insights into the UK Data job market: Analysing salaries, Skills demand and data scientist roles

The job market for data-related positions is dynamic and constantly evolving however it is very competitive and hard to get into. To earn high-end competitive salaries, you need high end data-analysis skills. This portfolio aims to provide a comprehensive analysis of job-adverts on leading job hiring websites such as: LinkedIn, Indeed, and Glassdoor. Using data analytical tools, we will offer insights into key trends and characteristics of these roles.

The portfolio includes three specific areas of questioning we aim to analyse:

1. Predicting the Expected Maximum Salary: Using linear regression, we will explore the main factors and predicators of salary ranges.
2. Skills in demand: Through association rule mining and Market Basket Analysis, we dive into the most sought-after technical and non-technical skills employers look for
3. Characterising Data scientist roles: We use classification methods to identify whether the job position is for a data scientist job, or a job related to data, and which programmes are best suited for which.

Together, these analyses contribute to a clearer understanding of the job landscape for data professionals in the UK.

Predicting the maximum expected salary Q1

Preparing the Data

The first thing we will do is load in our data, then begin to explore its characteristics and structure. Throughout this we have used the libraries: “arules”, “arulesViz”, “rpart”, “rpart.plot”

```
#packages needed
install.packages("arules")
install.packages("arulesViz")

library(arules)
library(arulesViz)
library(rpart)
library(rpart.plot)

# Loading the data from the csv file saved in the working directory.
data <- read.csv("Jobs.csv")
attach(data)
```

We then also prepare the transactional data set to make it easier to perform methods on and read/use

```
# Preparing the transactional data set
tr_matrix = cbind(data[,4:21])
names=names(tr_matrix)
tr_matrix=as.matrix(tr_matrix)
dimnames(tr_matrix) =  list(
  paste("Tr",c(1:1765), sep = ""),
  names)
tr =  as(tr_matrix, "transactions")
```

After doing this we begin to look at the data structure and most importantly if any data from any rows are missing. As there is a big output, I’ve truncated the output, but will outline the most important information given from this

```
summary(data)
str(data)
any(is.na(data))
```

```
> any(is.na(data))
[1] FALSE
```

```
> str(data)
'data.frame':   1765 obs. of  28 variables:
 $ company         : chr  "BT" "BT" "BT" "Barclays" ...
 $ job_title       : chr  "data scientist" "data scientist" "data scientist" "data
 $ City            : chr  "Birmingham" "London" "Birmingham" "Glasgow" ...
 $ Python          : int  1 1 1 1 1 1 1 1 0 1 ...
 $ R               : int  1 0 0 0 0 0 1 1 0 1 ...
 $ SAS             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Azure           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Devops          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ AWS             : int  0 1 1 0 1 1 0 0 0 0 ...
 $ GCP             : int  0 1 1 0 1 1 0 0 0 0 ...
 $ PySpark         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Databricks      : int  0 0 0 0 0 0 0 0 0 0 ...
```

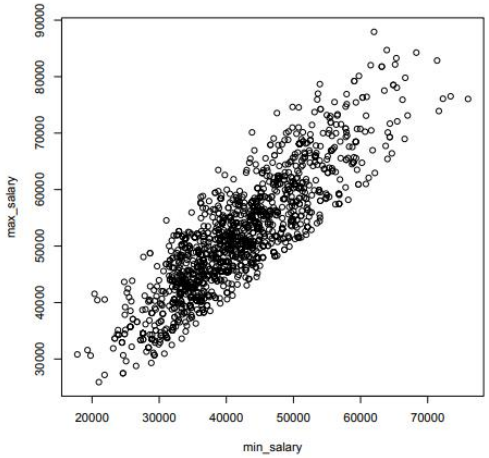
Here we see that there is no missing data, and so because we are working with pre-cleaned data, there is no need to perform anymore cleaning techniques. Secondly, we see that the data frame consists of 1765 records of input meaning we have taken data from 1765 unique job listings. Out of the 28, 5 of them have character data types whilst the remaining have integer types. Apart from salary columns, the integer columns are all 1 or 0 indicating a binary representation of a form of yes or no response to a skill/requirement.

Analysing the relationship between minimum and maximum salaries Q1a

Many job opportunities promote different ranges of salaries offered. The first thing we will look at is the relationship between the minimum and maximum salaries. This could give us potential insights into whether there is a chance for growth within the company. A low minimum salary but high maximum salary could incentivise employees for better work and loyalty reflected through a growing salary.

```
plot(data$min_salary, data$max_salary)
salary_corr = cor(data$min_salary, data$max_salary, method = "pearson" ,use ="pairwise.complete.obs")
```

From the plot of the graph itself we can see how there is a strong, positive, linear relationship between minimum and maximum salaries. The plot follows a general straight line with most job offers around the £40k and £60k margin as seen through the dark cluster of circles concentrated around that part of the graph. There doesn't seem to be any outliers.



We also managed to compute Pearsons's correlation coefficient. We used this as the variables were continuous. The coefficient came to be 0.87 reinforcing the strong, positive, linear relationship. Squaring this gives us a value of around 76%. This means that around 76% of the variance in the dependent variable can be explained by the independent variable, which supports the trend we see from the graph. In other words, it can be said that jobs that tend to offer high minimum salaries also offer high maximum salaries with little to none offering low minimum salaries and high maximum salaries. Most of the salary ranges are consistent with most of the variation which highlights how there is a consistent/agreed upon pay for the type of jobs being offered around data. Most of the variation seems to come at either end of the plot. At the lower end, it could be that it's a startup company with not much money to give, but as we go to the higher end the variance could be due to the fact it's a bigger company looking to train and grow their employees allowing for a bigger maximum salary.

One limitation of this analysis and for most of the analysis performed throughout is that we don't know when these job listings were taken, what we could be seeing is the effect of time on salaries, such as inflation over the years, rather than the relationship between salary ranges.

Using a linear regression line to make predictions Q1b

We will assume the linear regression model takes the form

$$Y = \beta_0 + \beta_1 X + U$$

where the dependent variable is the maximum expected salary (Y), the predictor is the minimum expected salary (X), and U is a normal random error. In this model, if B1>0, it would mean that for each pound the minimum salary increases by, the maximum salary would increase by more than a pound. Vice-versa if B1<0 for every pound increase in the minimum salary, we would see an increase of less than 1 pound. The following code to form a regression model is seen below along with its output.

```
fit = lm(data$max_salary ~ data$min_salary)
summary(fit)
Call:
lm(formula = data$max_salary ~ data$min_salary)

Residuals:
   Min     1Q Median     3Q    Max
-10026  -3360   -243   2843  16633

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     9.123e+03  5.749e+02   15.87   <2e-16 ***
data$min_salary 1.012e+00  1.364e-02   74.22   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4830 on 1763 degrees of freedom
Multiple R-squared:  0.7576,    Adjusted R-squared:  0.7574
F-statistic:  5509 on 1 and 1763 DF,  p-value: < 2.2e-16
```

From the output, we see that B0 (the intercept) has a value of £9,123, meaning that this is the predicted maximum salary for someone that has no expected minimum salary. B1 is that independent variable (minimum salary) which has a value of around 1.012, which indicates that on average, for every pound increase in minimum salary, the maximum salary increases by 1.012 pounds. The R-squared value is once again 0.76 which means 76% of the maximum salary's variance is linearly explained through the minimum salary.

Q1c

We are going to see if having past experience and knowledge of how to use R along with minimum salary can further explain maximum salary by using a Multiple linear regression analysis.

```
new_fit = lm(data$max_salary ~ data$min_salary + data$R + data$past_experience)
summary(new_fit)
```

```
Call:
lm(formula = data$max_salary ~ data$min_salary + data$R + data$past_experience)

Residuals:
     Min       1Q   Median       3Q      Max
-11877.6  -3332.0    -79.7   3097.5  15477.8

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         6.253e+03  6.296e+02   9.932  < 2e-16 ***
data$min_salary     1.053e+00  1.417e-02  74.340  < 2e-16 ***
data$R              2.239e+03  2.762e+02   8.108 9.55e-16 ***
data$past_experience 1.088e+03  2.279e+02   4.772 1.98e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4699 on 1761 degrees of freedom
Multiple R-squared:  0.7707,    Adjusted R-squared:  0.7703
F-statistic:  1973 on 3 and 1761 DF,  p-value: < 2.2e-16
```

```
sum(residuals(new_fit))
```

As we can see here, the intercept is £6,253 and the estimated coefficients for minimum salary, R-experience and past job experience is £1.053, £2,239 and £1,088 respectively. We also find the sum of the residuals to be 1.134993e-09 which is almost negligible as we would expect it to be so we can ignore it in the equation. The final equation for the regression model is

$$\max \text{salary} = 6.253 + 1.053(\min salary) + 2239(R\ experience) + 1088(past\ experience)$$

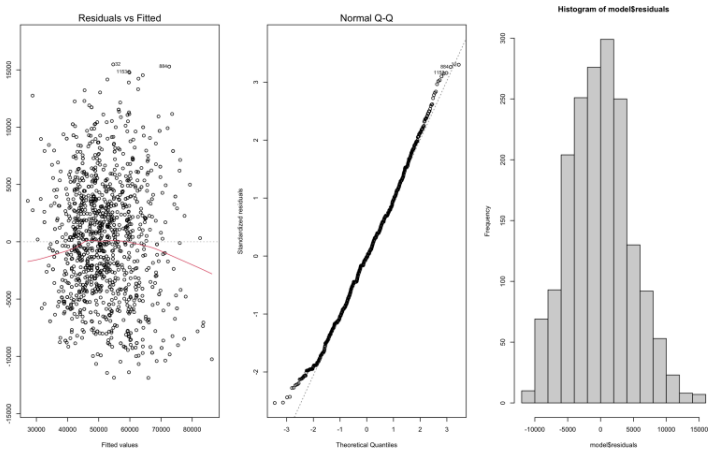
As R experience and past experience can only take values 1 or 0 showing the absence or presence of it, it expectedly has a bigger impact on that equation that minimum salary which can take multiple values. Having either experience will increase your salary flat out. To find out the predicted maximum salary of a job with a minimum salary of £48,000 but required past job experience and R-experience we can use the following code seen below.

```
> 6253+1.053*(48000)+2239*(1)+1088*(1)
[1] 60124
```

Therefore, the predicted maximum salary is £60,124.

Assumptions of a linear regression model Q1d

When applying a linear regression model, there are some assumptions we make such as, the random error is normally distributed, homoscedasticity, errors are independent but most importantly that the relationship is linear between all the predicators and response. As the model is non-deterministic until we make sure the relationship is linear, the model may give us misleading information. However, we can confirm linearity through the following graphs.



From the residuals vs fitted on the very left, we see a constant variance for the most part and looks to be randomly scattered. The main idea is, is that there is no clear cluster or pattern and so we from this we can say the first plot indicates a linear relationship. This is further supported by the Normal Q-Q plot. We see that residuals follow the straight diagonal line, especially throughout at the centre. There are no large deviations showing a good fit for a normal distribution. However, to confirm this we also plot a histogram of the residuals. We see an overall bell-curve shape indicating a good normal distribution fit. There is a slight left-skewness in the histogram and slight tail at the ends of the Q-Q plot indicating that in general the model seems to be overestimating the dependent variable at the low and high ends. However overall, the assumptions of the linear regression model are satisfied.

Skills in demand Q2

There are a lot of organizations and businesses that seek out all sorts of technical skills. However, universities also seek out these types of skills in accordance with what employers want, so they can update their curriculum. This part of the analysis will look at understanding the most sought out skills using market basket analysis.

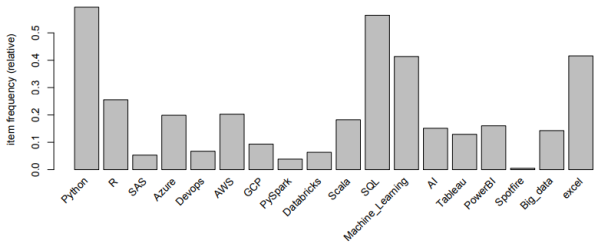
Top 5 most desirable skills Q2a

By using the code below we can find out the 5 most sought out skills in the job market

```
head(sort(itemFrequency(tr), decreasing = TRUE))
```

```
head(sort(itemFrequency(tr), decreasing = TRUE))
    Python        SQL      excel Machine_Learning          R
 0.5937677  0.5643059  0.4158640       0.4135977  0.2555241
```

From the analysis, we conclude that Python, SQL, Excel, Machine Learning and R are the most wanted technical skills in job listings. In order, these skills can be seen in 59%, 56%, 42%, 41% and 26% in job recruitment requirements. This is also backed up from the bar chart given on the right



Finding interesting rules Q2b

The code below shows the code and output of applying the apriori algorithm.

```
rules=apriori(tr, parameter=list(support=0.01, confidence=0.01, minlen=2))
summary(rules)
```

```
> summary(rules)
set of 4373 rules

rule length distribution (lhs + rhs):sizes
   2    3    4    5    6    7
 242 1059 1748 1095  222    7

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   3.000   4.000   4.004   5.000   7.000

summary of quality measures:
    support          confidence        coverage            lift             count
 Min.   :0.01020   Min.   :0.0327   Min.   :0.01020   Min.   : 0.5697   Min.   : 18.00
 1st Qu.:0.01190   1st Qu.:0.3231   1st Qu.:0.02153   1st Qu.: 1.3783   1st Qu.: 21.00
 Median :0.01586   Median :0.4943   Median :0.03513   Median : 1.6438   Median : 28.00
 Mean   :0.02328   Mean   :0.5425   Mean   :0.06075   Mean   : 2.1320   Mean   : 41.09
 3rd Qu.:0.02436   3rd Qu.:0.8140   3rd Qu.:0.06402   3rd Qu.: 2.4178   3rd Qu.: 43.00
 Max.   :0.42380   Max.   :1.0000   Max.   :0.59377   Max.   :11.0896   Max.   :748.00

mining info:
 data ntransactions support confidence
   tr          1765    0.01       0.01
                                                                                call
 apriori(data = tr, parameter = list(support = 0.01, confidence = 0.01, minlen = 2))
```

Here we see that with a minimum support and confidence of 0.01, we obtain a set of 4373 rules. To find the most interesting rules we should define a minimum limit for the support, which is the percentage of transactions that contain both item sets A and B, the confidence, which is the percentage of transactions containing the itemset A that also contain B and the lift being ratio between the confidence and support. Here it would not be reasonable to set a minimum support of 50%, as the maximum support of any observed rule is 42%. Setting a minimum confidence of 80% would be a good idea as it captures the upper third quartile of data. So you would get a good amount of rules to perform analysis on without compromising on the quality/confidence of them. As the lift represents how confident we are that someone will have B given an itemset A, we want rules with a high lift. We see that the max lift is 11, however that is most likely to be an outlier as the median and mean is around 1.8. Having a lift of 1 however would be too low as it would include more than 75% of data which can be computationally expensive to explore. A good lift requirement would be above 2.

Assessing rules Q2c

```
inspect(head(rules, n=3, by="lift"))
```

By running the code seen above, we get the following output that shows us the 3 highest rules according to lift which are,

```
> inspect(head(rules, n=3, by="lift"))
    lhs                                                rhs          support    confidence coverage
[1] {Python, Azure, Devops, PowerBI}                => {Databricks} 0.01076487 0.7037037  0.01529745
[2] {Python, Azure, Devops, SQL, PowerBI}           => {Databricks} 0.01076487 0.7037037  0.01529745
[3] {Python, Azure, Machine_Learning, PowerBI, excel} => {Databricks} 0.01133144 0.6896552  0.01643059
    lift     count
[1] 11.08962 19
[2] 11.08962 19
[3] 10.86823 20
```

Changing the parameter of "by" to confidence we can obtain top 3 rules in terms of support and confidence. Thus:

By Support

Rule	**Support**	**Confidence**	**Lift**
{SQL} => {Python}	0.423	0.751	1.25
{Python} => {SQL}	0.423	0.713	1.26
{Machine_Learning} => {Python}	0.357	0.864	1.45

By Confidence

Rule	**Support**	**Confidence**	**Lift**
{PySpark, Databricks} => {SQL}	0.0101	1.00	1.77
{PySpark, PowerBI} => {SQL}	0.0124	1.00	1.77
{PySpark, Scala} => {Python}	0.0164	1.00	1.68

Analysing the rule, {SQL} => {Python}, we see that it has a support of 0.423 meaning 42.3% of job listings that require SQL as a skill also require Python. A confidence of 0.751 reveals that 75.1% of our rows that have SQL have SQL and python as a requirement. Finally, a lift of 1.25 shows that there is a 25% higher chance of python appearing as a valued skill when SQL is listed rather than just appearing by chance.

Q2d

We will now increase the minimum thresholds for support and confidence to 15% and 80% so we can find the most frequent requirements that appear together and create strong rules. We change the code as follows and decide to rank it by lift. I made this choice as we are already inputting minimum thresholds for support and confidence and so from the required thresholds, we want the rules with the highest lift to predict the skills jobs want from candidates

```
new_rules=apriori(tr, parameter=list(support=0.15, confidence=0.8), minlen=2)
inspect(head(new_rules, n =3, by="lift"))
```

And obtain the following rules

```
lhs                          rhs      support   confidence coverage  lift     count
{SQL, Machine_Learning} => {Python} 0.2260623 0.9279070  0.2436261 1.562744 399
{R, SQL}                => {Python} 0.1654391 0.9096573  0.1818697 1.532009 292
{Machine_Learning}      => {Python} 0.3575071 0.8643836  0.4135977 1.455760 631
```

These were the top three out of the 6 obtained rules. The most interesting rules would be the one with the highest support and confidence. This would seem to be {SQL} => {Python} and {Python} => {SQL}. The have high proportions for both whilst their lift is only around 0.3 - 0.25 below that of rules with the highest lift as seen above. Universities can use this to their advantage and make sure that students have apt knowledge in both SQL and PYTHON as both are in the two most interesting rules in my opinion. In doing so, they open their students up to around 40% of the job market according to our data, as we predict and conclude that jobs that have one will require or requires the other.

Characterising Data Scientists Roles Q3

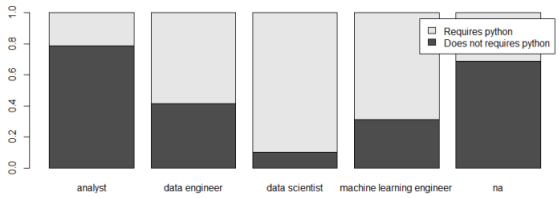
As Data science jobs are in high demand, many universities are looking to investigate which are the common features of job adverts for these type of job roles. They are particularly interested in which type of programme languages are sought after. With this aim in mind, we will apply a classification method to filter out jobs into one of two categories: Data scientist or data related jobs. We will use a subset of independent variables discussed below as predictors.

Q3a

We will explore the data further here to examine the proportion of job titles that require python as a skill or if it doesn't. In the dataset this is represented in binary with 1 being yes and 0 being no. The code following gives us the graph shown.

```
prop.table(table(data$Python, data$job_title), 2)
barplot(prop.table(table(data$Python, data$job_title), 2),
        legend.text = c("Does not require python", "Requires python"))
```

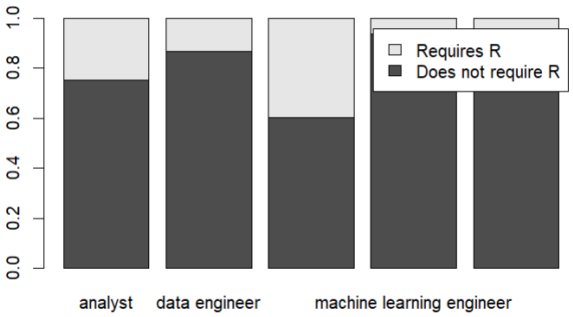
```
      job_title
Python   analyst data engineer data scientist machine learning engineer        na
     0 0.7858881     0.4162349      0.1031250                 0.3125000 0.6890756
     1 0.2141119     0.5837651      0.8968750                 0.6875000 0.3109244
```



Here we can see that over half of jobs labelled as: data engineer, data scientist or machine learning engineer require python knowledge. In particular, around 90% of jobs labelled as data scientist require python. This graph and table indicate that its desirable to hire candidates with python knowledge. We will know also look at how desirable it is to have R as a programming language skill.

```
prop.table(table(data$R, data$job_title), 2)
barplot(prop.table(table(data$R, data$job_title), 2),
        legend.text = c("Does not require R", "Requires R"))
```

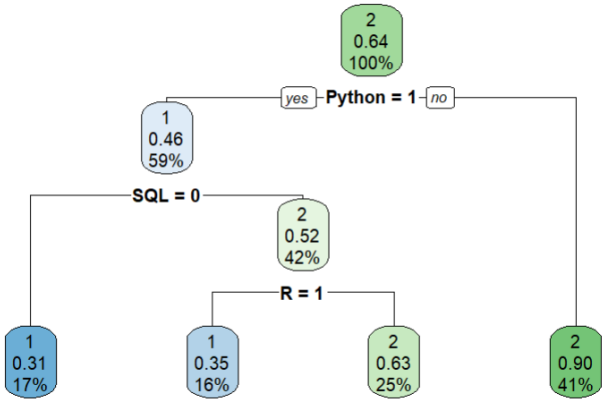
```
    analyst data engineer data scientist machine learning engineer        na
0 0.7542579     0.8670121      0.6031250                 0.9375000 0.8487395
1 0.2457421     0.1329879      0.3968750                 0.0625000 0.1512605
```



Here we have also produced a proportions table and the corresponding graph. This draws a completely opposite conclusion from the tables associated with python. The highest percentage job classification that requires R is once again data scientist with around 40%. From both plots we can begin to hypothesize that that python is generally a commonly used programme within a lot of jobs, but that's not the case with R. We see how the distribution of jobs that require R is very within most of the fields. We can conclude that having R as a skill isn't as necessary as having python. If one was to only have R experience, he would only be considered for around 13% of jobs according to our data

Building a classification model Q3b

```
n=length(data$class)
set.seed(1200) #the last 4 digits of my ID
index=sample(1:n,round(n * 0.80),replace=FALSE)
train=data[index,]
test=data[-index,]
mytree <- rpart(
  class ~ past_experience+Python+SQL+R,
  data = train,
  method = "class"
)
rpart.plot(mytree)
```



The code above shows us splitting the class column into two parts. The training data which has 80% of the data and training data which includes the remaining 20%. We then built a decision tree to help us classify what a

job listing is asking for depending on their past experience and their knowledge of python, SQL and R. It then also shows the tree seen above.

We have two initial classes, 1 represents jobs labelled “data scientist” and 2 as “other jobs related to data”. The tree we created shows us how most of the jobs are assumed to be of class 2 as seen from the root node. This is because 64% of jobs are class 2 already and at this stage, we have begun classifying data and so 100% of the training data is still there. On the other hand, we see smaller values on leaf nodes such as the one on the very right. This is because we have classified most of the data already into a class. The one on the very right suggest that if you have python and SQL experience there is a 31% chance you will be labelled class 1. The tree has classified 17% of data this way.

We can get the biggest split by of data by first asking if python is required, SQL and then R. This reinforces the reoccurring theme we have found so far that python and SQL are the most desirable skills, and R is not as important. The tree also does not evaluate whether past experience is required as we get pure classes from just evaluating the afore-mentioned programming skills.

Using this tree, we will classify two jobs

- Job 1: the job is in London, requires knowledge of Python, machine learning, SQL, big data, excel and to have previous job experience.
    - Starting from the root node we go left as python is required. We then evaluate if SQL is required, and it is so we go left once more and end up at the terminal node predicting us to be a data scientist
- Job 2: the job is in London, requires knowledge of R, SQL, SAS, Scala, excel and to have a Master degree qualification.
    - Starting from the root node, as we do not need knowledge of python we follow the tree to the right. This leads us straight to the terminal node where we are classed as class 2, “other jobs related to data”

Knowing that Job 1 was looking for a data scientist and job 2 was looking for a data analyst our tree correctly predicts the job classifications

Evaluating our model Q3c

We will now evaluate our model by constructing a confusion matrix and looking at the proportion of correct and incorrect classifications.

```
train_pred <-predict(mytree, train, type = "class")
train_CM <- table(class[index], train_pred)
train_CM
prop.table(train_CM)
```

```
train_CM
 train_pred
   1   2
1 319 190
2 155 748
prop.table(train_CM)
 train_pred
          1         2
1 0.2259207 0.1345609
2 0.1097734 0.5297450
```

The confusion matrix shown shows us that 319 data analyst jobs were correctly classified and 748 other related data jobs were also assigned the correct class. We see that around 75% of job listings were correctly classified

Using the formulas seen below we also obtain the accuracy, precision, sensitivity and specificity of the model

```
accuracy <- sum(diag(train_CM)) / sum(train_CM)
precision <- train_CM["2", "2"] / (train_CM["2", "2"] + train_CM["1", "2"])
sensitivity <- train_CM["2", "2"] / (train_CM["2", "2"] + train_CM["2", "1"])
specificity <- train_CM["1", "1"] / (train_CM["1", "1"] + train_CM["1", "2"])
```

```
 print(paste("Accuracy:", round(accuracy, 4)))
L] "Accuracy: 0.7557"
 print(paste("Precision:", round(precision, 4)))
L] "Precision: 0.7974"
 print(paste("Sensitivity:", round(sensitivity, 4)))
L] "Sensitivity: 0.8283"
 print(paste("Specificity:", round(specificity, 4)))
L] "Specificity: 0.6267"
```

We will now use the same methods to predict the classes on the remaining test data.

```
test_pred <-predict(mytree, test, type = "class")
test_CM <- table(class[-index], test_pred)
test_CM
prop.table(test_CM)
```

```
> print(paste("Accuracy:", round(accuracy, 4)))
[1] "Accuracy: 0.7557"
> print(paste("Precision:", round(precision, 4)))
[1] "Precision: 0.7957"
> print(paste("Sensitivity:", round(sensitivity, 4)))
[1] "Sensitivity: 0.8423"
> print(paste("Specificity:", round(specificity, 4)))
[1] "Specificity: 0.6336"
```

```
> test_CM
   test_pred
      1   2
  1  83  48
  2  35 187
> prop.table(test_CM)
   test_pred
             1          2
  1 0.23512748 0.13597734
  2 0.09915014 0.52974504
```

We see that both yield similar proportions in terms of incorrect and correctly identifying data meaning our model is generalizing well and its under or over fitted to specific data. Accuracy is the proportion of correctly predicted classes, precision is the proportion of positives that we actually positive, sensitivity is the proportion of actual positives that were correctly identified, and specificity is the proportion of negatives that were correctly identified. All our values are relatively high which emphasis the goodness of our model. From the results our models biggest flaw is that its slightly more unsuccessful when identifying actual negatives. In this case it would be class 2.

Conclusion

The portfolio explored and examined the UK job market for fields related to data science and data as a whole. We predominantly focused on salaries, skills in demand and characterising data scientists using a plethora of statistical and analytical methods. We yielded the following key insights.

1. **Predicting the Expected Maximum Salary**

- There is a strong, linear, positive relationship between minimum and maximum salaries, with a Pearson correlation coefficient of 0.87
- The developed regression model indicates that for every £1 increase in minimum salary, the maximum salary increases by approximately £1.012.
- Factors like R experience and prior job experience further heighten the predicted maximum salary

2. **Skills in Demand**

- Python and SQL are the most sought-after skills, appearing in 59% and 56% of listings respectively. They are frequently listed together, as shown through our association rule mining
- The strongest two rules are {SQL} => {Python} and {Python} => {SQL}. It highlights a synergy between these skills, suggesting that universities should prioritize equipping students with both skills to increase employability.

3. **Characterising Data Scientists Roles**

- Decision tree classification highlighted Python and SQL as the strongest indicators of whether a job is listed as “data scientist” with R being comparatively less critical.
- This model achieved around a 62-82% rate on the main metric we use for evaluating models. Specifically, a 75% accuracy rate from distinguishing data scientist jobs from other data-related jobs

These insights help many people at a lot of different levels. For job seekers, the data suggests that they should focus on developing Python and SQL expertise as these skills are in high demand. Universities should also incorporate these two programming languages into curriculums while also lightly exposing R to students as a supplementary.

As stated before that whilst this gives us important insights, there are also important limitations to consider. The data frame contains no information on when the information was scaped and so the trends and patters could be false or misleading. For example, the minimum salary 20 years ago was much lower than it would be today but relatively high in that time period. This data is limited in how long it may be useful for, other on-the-rise programming languages may supersede python and SQL rendering it obsolete. So whilst we predict from the data these are important skills, in a few years the data may not continue to reflect present trends.