# R Notebook

Data Science Job Salaries

The majority of information technology students are curious about knowing the salary levels in the field of data science. We will build a model for a data set of employees in the field of data science in different companies and study this data to reach results that help visualize the salaries of these employees and what are the factors affecting them.

Ruba Albnhar - Rema ALsharef - Maryam Alrawayah

1 Problem

described in the statement is the need to understand the factors that influence salaries in the rapidly expanding field of data science.

The primary objective of collecting this data set is to gain a comprehensive understanding of the factors that impact salaries within the rapidly expanding domain of data science. This understanding is of paramount importance for individuals, job seekers, and organizations. We intend to achieve this goal by conducting a thorough analysis of various job-related and employee-specific attributes, including but not limited to experience level, employment type, years of work, job title, salary, remote work prevalence, company location, and company size.

Our overarching aim is to develop a predictive model that can serve as a valuable tool for both individuals and businesses, aiding them in making well-informed decisions and staying abreast of prevailing market trends. In practical terms, this data set will empower organizations to establish equitable and competitive salary structures and promote meaningful dialogues during the hiring process.

2 Data mining task

The data mining task associated with this problem is regression or classification and clustering analysis. is used to predict a continuous variable (in this case, salary) based on various input variables (such as experience level, employment type, work years, job title, remote work ratio, company location, and company size). By analyzing the relationship between these input variables and the salary, a model can be built to predict salaries for new instances or understand the impact of each variable on the salary outcome.

3 data

source:https://www.kaggle.com/datasets/ruchi798/data-science-job-salaries

columns:12

rows:607

Work_year : The year the salary was paid.

experience_level : The experience level in the job during the year with the following possible values: EN Entry-level / Junior MI Mid-level / Intermediate SE Senior-level / Expert EX Executive-level / Director

employment_type : The type of employement for the role: PT Part-time FT Full-time CT Contract FL Freelance

job_title : The role worked in during the year.

salary: The total gross salary amount paid.

salary_currency: The currency of the salary paid as an ISO 4217 currency code

salary_in_usd : The salary in USD (FX rate divided by avg. USD rate for the respective year via fx-data.foorilla.com).

employee_residence: Employee's primary country of residence in during the work year as an ISO 3166 country code.

remote_ratio: The overall amount of work done remotely, possible values are as follows: 0 No remote work (less than 20%) 50 Partially remote 100 Fully remote (more than 80%)

company_location: The country of the employer's main office or contracting branch as an ISO 3166 country code.

company_size: The average number of people that worked for the company during the year: S less than 50 employees (small) M 50 to 250 employees (medium) L more than 250 employees (large)

---

**Installing Libraries**

```
install.packages("dplyr")
```

```
## package 'dplyr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("ggplot2")
```

```
## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("knitr")
```

```
## package 'knitr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("caret")
```

```
## package 'caret' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("rpart")
```

```
## package 'rpart' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("rpart.plot")
```

```
## package 'rpart.plot' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("rattle")
```

```
## package 'rattle' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("cluster")
```

```
## package 'cluster' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("factoextra")
```

```
## package 'factoextra' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

```
install.packages("fpc")
```

```
## package 'fpc' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Rema Alsharef\AppData\Local\Temp\RtmpcDdB1P\downloaded_packages
```

---

### Loading Libraries

```
library(dplyr)
library(ggplot2)
library(knitr)
library(caret)
```

```
library(rpart)
library(rpart.plot)
library(rattle)
library(cluster)
library(factoextra)
library(fpc)
```

---

## Import Data Set

We will employ the read.csv() function to import and process our data set [1]

```
#don't forget to replace the "C:/USers/AB/Downloads/salaries_data.csv" with the path where your data is
#[1]
job_data <- read.csv("salaries_data.csv")
```

### Explore Data

- We will look at the 1st 5 rows and last 5 rows of the data set
- (HEAD) is a convenient function for quickly inspecting the structure and content of your data.
- (TAIL) function is used to display the last few rows of a data ,is useful for quickly checking the end of a dataset

```
#kable(job_data) * we removed this because it takes a long space from the pdf
head(job_data)
```

```
##   X work_year experience_level employment_type                 job_title
## 1 0      2020               MI              FT             Data Scientist
## 2 1      2020               SE              FT Machine Learning Scientist
## 3 2      2020               SE              FT          Big Data Engineer
## 4 3      2020               MI              FT       Product Data Analyst
## 5 4      2020               SE              FT  Machine Learning Engineer
## 6 5      2020               EN              FT               Data Analyst
##   salary salary_currency salary_in_usd employee_residence remote_ratio
## 1  70000             EUR         79833                 DE            0
## 2 260000             USD        260000                 JP            0
## 3  85000             GBP        109024                 GB           50
## 4  20000             USD         20000                 HN            0
## 5 150000             USD        150000                 US           50
## 6  72000             USD         72000                 US          100
##   company_location company_size
## 1               DE            L
## 2               JP            S
## 3               GB            M
## 4               HN            S
## 5               US            L
## 6               US            L
```

```r
tail(job_data)
```

```
##       X work_year experience_level employment_type     job_title salary
## 602 601      2022               EN              FT  Data Analyst  52000
## 603 602      2022               SE              FT Data Engineer 154000
## 604 603      2022               SE              FT Data Engineer 126000
## 605 604      2022               SE              FT  Data Analyst 129000
## 606 605      2022               SE              FT  Data Analyst 150000
## 607 606      2022               MI              FT  AI Scientist 200000
##     salary_currency salary_in_usd employee_residence remote_ratio
## 602             USD         52000                 CA            0
## 603             USD        154000                 US          100
## 604             USD        126000                 US          100
## 605             USD        129000                 US            0
## 606             USD        150000                 US          100
## 607             USD        200000                 IN          100
##     company_location company_size
## 602               CA            M
## 603               US            M
## 604               US            M
## 605               US            M
## 606               US            M
## 607               US            L
```

## Stucture of the data set

- STR is used to display the structure and summary of an R will include str(job_data) information such as the total number of observations, the number of variables, and the data types of each variable. It will also provide a preview of the first few values of each variable.

```r
str(job_data)
```

```
## 'data.frame':    607 obs. of  12 variables:
##  $ X                 : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ work_year         : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ experience_level  : chr  "MI" "SE" "SE" "MI" ...
##  $ employment_type   : chr  "FT" "FT" "FT" "FT" ...
##  $ job_title         : chr  "Data Scientist" "Machine Learning Scientist" "Big Data Engineer" "Produ
##  $ salary            : int  70000 260000 85000 20000 150000 72000 190000 11000000 135000 125000 ...
##  $ salary_currency   : chr  "EUR" "USD" "GBP" "USD" ...
##  $ salary_in_usd     : int  79833 260000 109024 20000 150000 72000 190000 35735 135000 125000 ...
##  $ employee_residence: chr  "DE" "JP" "GB" "HN" ...
##  $ remote_ratio      : int  0 0 50 0 50 100 100 50 100 50 ...
##  $ company_location  : chr  "DE" "JP" "GB" "HN" ...
##  $ company_size      : chr  "L" "S" "M" "S" ...
```

The data set comprises 608 rows and 12 columns. The data set includes the following columns, each providing specific information:

- Work_year: This column represents the year in which the salary was disbursed.

- Experience_level: This field categorizes the job experience level during the year into four possible values:
  - EN: Entry-level / Junior
  - MI: Mid-level / Intermediate
  - SE: Senior-level / Expert
  - EX: Executive-level / Director
- Employment_type: This column describes the type of employment for the role, which can be classified as:
  - PT: Part-time
  - FT: Full-time
  - CT: Contract
  - FL: Freelance
- Job_title: This field specifies the job role held during the year.
- Salary: This column records the total gross salary amount paid.
- Salary_currency: It represents the currency of the salary paid, adhering to the ISO 4217 currency code standard.
- Salary_in_usd: This column indicates the equivalent salary in USD, calculated by applying the foreign exchange rate and dividing it by the average USD rate for the respective year.
- Employee_residence: This field records the employee's primary country of residence during the work year.
- Remote_ratio: It signifies the extent of remote work undertaken, with possible values as follows:
  - 0: No remote work (less than 20%)
  - 50: Partially remote
  - 100: Fully remote (more than 80%)
- Company_location: This column specifies the country of the employer's main office or contracting branch, using the ISO 3166 country code.
- Company_size: This field characterizes the average number of individuals employed by the company during the year, categorized as:
  - S: Less than 50 employees (small)
  - M: 50 to 250 employees (medium)
  - L: More than 250 employees (large)

**Data Cleaning**

**1.Rename column names** As evident, the first row contains default column names labeled from V1 to V12, while the actual column names are stored in the second row. To rectify this, we must eliminate the first row and convert the second row into the appropriate column names.

rename the column names in the job_data dataframe. However, there seems to be a syntax error in the column names assignment

This code assigns new column names to the job_data dataframe using the colnames() function. The correct column names are provided as a character vector. After renaming the column names, the first row is removed using the [-1, ] indexing. Finally, head(job_data) displays the first few rows of the updated job_data dataframe

```r
colnames(job_data) <- c("ID", "work_year", "experience_level", "employment_type", "job_title",
                        "salary", "salary_currency", "salary_in_usd", "employee_residence",
                        "remote_ratio", "company_location", "company_size")
job_data <- job_data[-1, ]
head(job_data)
```

```
##   ID work_year experience_level employment_type                 job_title
## 2  1      2020               SE              FT Machine Learning Scientist
## 3  2      2020               SE              FT          Big Data Engineer
## 4  3      2020               MI              FT       Product Data Analyst
## 5  4      2020               SE              FT  Machine Learning Engineer
## 6  5      2020               EN              FT               Data Analyst
## 7  6      2020               SE              FT        Lead Data Scientist
##   salary salary_currency salary_in_usd employee_residence remote_ratio
## 2 260000             USD        260000                 JP            0
## 3  85000             GBP        109024                 GB           50
## 4  20000             USD         20000                 HN            0
## 5 150000             USD        150000                 US           50
## 6  72000             USD         72000                 US          100
## 7 190000             USD        190000                 US          100
##   company_location company_size
## 2               JP            S
## 3               GB            M
## 4               HN            S
## 5               US            L
## 6               US            L
## 7               US            S
```

```r
table(is.na(job_data))
```

**2.Check NA Values**

```
##
## FALSE
##  7272
```

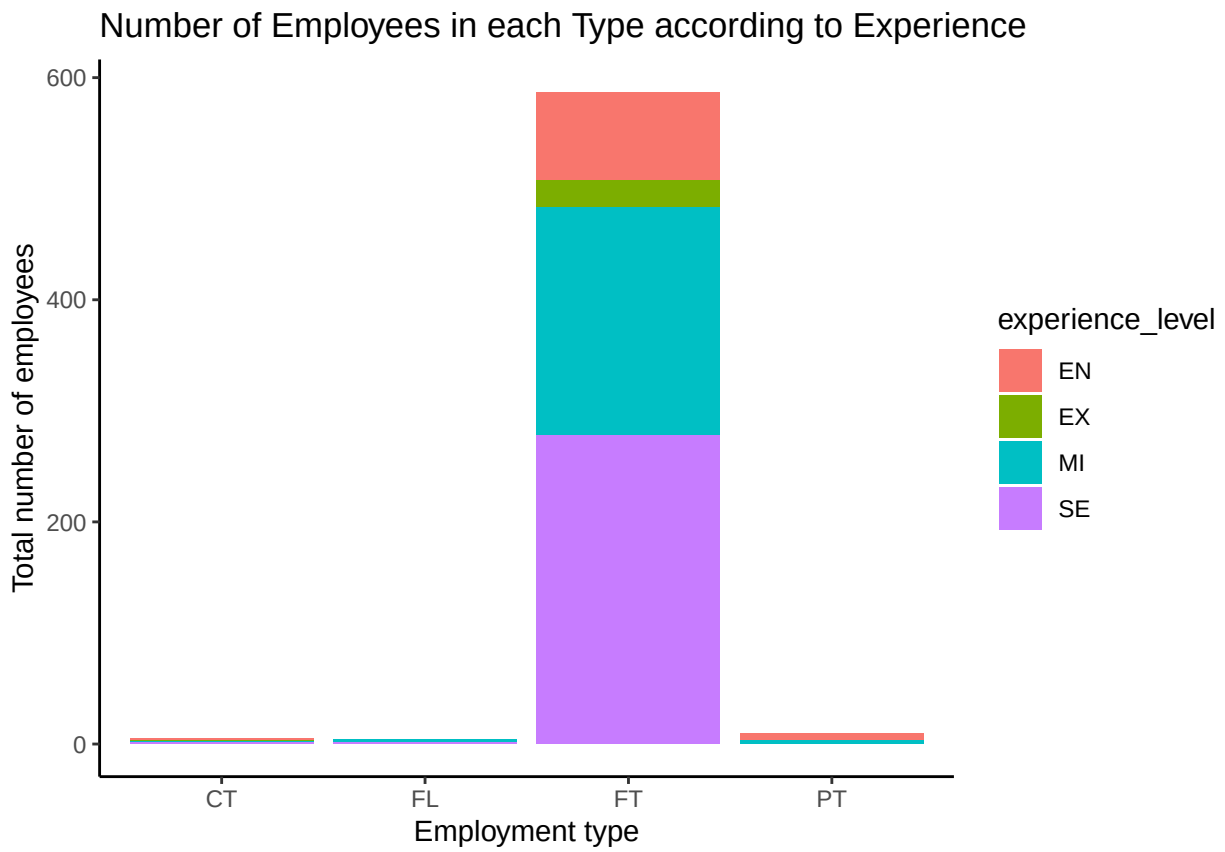
The data set is devoid of any missing or NA values.

**Number of exployees according to experience**

The code snippet you provided is using the ggplot2 package in R to create a bar plot showing the number of employees according to their experience level, categorized by employment type. Explanation:- ggplot(job_data): Initializes a new ggplot object with the job_data dataframe as the data source. - geom_bar(mapping = aes(employment_type, fill = experience_level)): Specifies a bar plot (`geom_bar()`) and maps the employment_type variable to the x-axis and the experience_level variable to the fill color of the bars. This will create stacked bars representing the number of employees based on their experience level within each employment type. - xlab("Employment type"): Sets the x-axis label as "Employment type". - ylab("Total number of employees"): Sets the y-axis label as "Total number of employees". - ggtitle("Number of Employees in each Type according to Experience"): Sets the title of the plot as "Number of

Employees in each Type according to Experience". - theme_classic(): Applies a classic theme to the plot, which provides a clean and minimalistic appearance.This code will generate a bar plot

```
ggplot(job_data)+ geom_bar(mapping = aes(employment_type, fill = experience_level)) +
  xlab("Employment type") + ylab("Total number of employees") +
  ggtitle("Number of Employees in each Type according to Experience") +
  theme_classic()
```



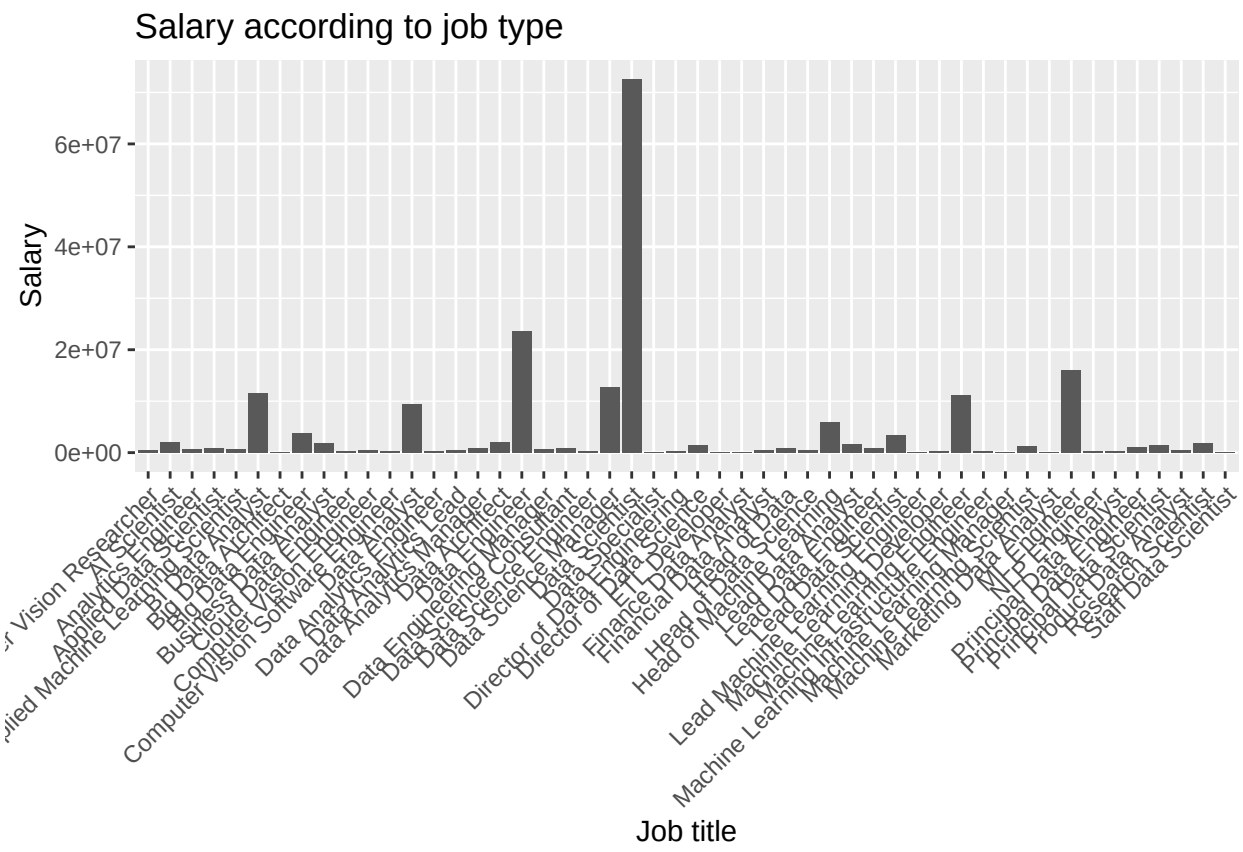

---

**Salary according to job type**

- ggplot(job_data, aes(job_title, salary)): Initializes a new ggplot object with the job_data dataframe as the data source and maps the job_title variable to the x-axis and the salary variable to the y-axis.
- geom_bar(stat = "identity"): Specifies a bar plot (`geom_bar()`) and uses the "identity" statistic to plot the bars directly from the data values. This is suitable for displaying the actual salary values.
- theme(axis.text.x = element_text(angle = 45, hjust = 1)): Adjusts the appearance of the x-axis labels by rotating them 45 degrees clockwise and aligning them to the right.
- xlab("Job title"): Sets the x-axis label as "Job title".
- ylab("Salary"): Sets the y-axis label as "Salary".
- ggtitle("Salary according to job type"): Sets the title of the plot as "Salary according to job type".

```
ggplot(job_data, aes(job_title, salary))+
        geom_bar(stat = "identity")+
```

```
        theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
        xlab("Job title") + ylab("Salary") +
        ggtitle("Salary according to job type")
```



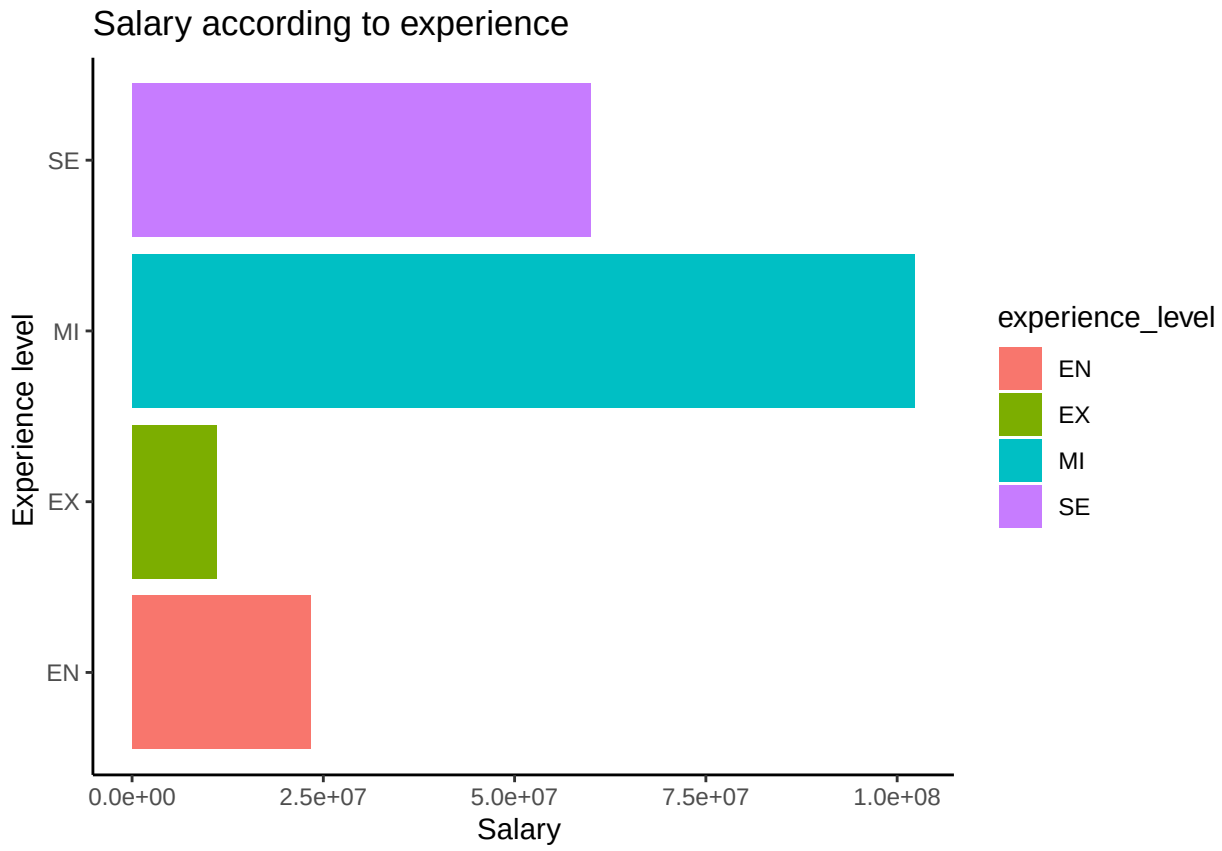

---

**Salary according to experience level**

ggplot(job_data, aes(experience_level, salary, fill = experience_level))‘: Initializes a new ggplot object with the job_data dataframe as the data source and maps the experience_level variable to the x-axis, the salary variable to the y-axis, and the experience_level variable to the fill aesthetic. This means that each bar will be filled based on the experience level.

- geom_bar(stat = “identity”): Specifies a bar plot (`geom_bar()`) and uses the “identity” statistic to plot the bars directly from the data values. This is suitable for displaying the actual salary values.
- xlab(“Experience level”): Sets the x-axis label as “Experience level”.
- ylab(“Salary”): Sets the y-axis label as “Salary”.
- ggtitle(“Salary according to experience”): Sets the title of the plot as “Salary according to experience”.
- theme_classic(): Applies a classic theme to the plot, which gives a clean and minimalistic appearance.
- coord_flip(): Flips the x and y axes, so that the bars are displayed horizontally. This is useful when you have long category names on the x-axis and want to improve readability.

```r
ggplot(job_data, aes(experience_level, salary, fill = experience_level))+
        geom_bar(stat = "identity") + xlab("Experience level ") + ylab("Salary ") +
        ggtitle("Salary according to experience ") + theme_classic() + coord_flip()
```




4 Data Preprossing part (1)

**3.Change some variables class** We will proceed to convert certain character variables, specifically "remote_ratio," "experience_level," "employment_type," and "company_size," into their numeric representations. This transformation will facilitate more efficient attribute encoding and calculations.

```r
job_data$remote_ratio <- factor(job_data$remote_ratio, levels = c("0","50","100"), labels = c(0,0.5,1))

job_data$experience_level <- factor(job_data$experience_level, levels = c("EN","MI","SE","EX"),
                                    labels = c(1 , 2 , 3 , 4))

job_data$employment_type <- factor(job_data$employment_type, levels = c('PT','FT','CT','FL'),
                                   labels = c(1,2,3,4))

job_data$company_size <- factor(job_data$company_size, levels = c('S','M','L'), labels = c(1,2,3))
job_data$salary_in_usd <- as.numeric(job_data$salary_in_usd)
job_data$job_title <- factor(job_data$job_title)
job_data$company_location <- factor(job_data$company_location)
str(job_data)
```

```
## 'data.frame':    606 obs. of  12 variables:
##  $ ID                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ work_year         : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ experience_level  : Factor w/ 4 levels "1","2","3","4": 3 3 2 3 1 3 2 2 3 1 ...
##  $ employment_type   : Factor w/ 4 levels "1","2","3","4": 2 2 2 2 2 2 2 2 2 2 ...
##  $ job_title         : Factor w/ 50 levels "3D Computer Vision Researcher",..: 41 8 48 38 13 35 23 9
##  $ salary            : int  260000 85000 20000 150000 72000 190000 11000000 135000 125000 45000 ...
##  $ salary_currency   : chr  "USD" "GBP" "USD" "USD" ...
##  $ salary_in_usd     : num  260000 109024 20000 150000 72000 ...
##  $ employee_residence: chr  "JP" "GB" "HN" "US" ...
##  $ remote_ratio      : Factor w/ 3 levels "0","0.5","1": 1 2 1 2 3 3 2 3 2 1 ...
##  $ company_location  : Factor w/ 50 levels "AE","AS","AT",..: 30 19 21 49 49 49 23 49 39 18 ...
##  $ company_size      : Factor w/ 3 levels "1","2","3": 1 2 1 3 3 1 3 3 1 1 ...
```

---

**4.Dimensionality Reduction** We will eliminate certain variables, which are:

- ID
- SALARY
- SALARY CURRENCY
- EMPLOYEE RESIDENCE

As they introduce redundancy to our data set and do not contribute valuable information to our primary objective, which is understanding the factors that influence salaries in the field of data science.

```
job_data <- subset(job_data, select = -c(ID))
job_data <- subset(job_data, select = -c(salary))
job_data <- subset(job_data, select = -c(salary_currency))
job_data <-subset(job_data, select = -c(employee_residence))
head(job_data)
```

```
##   work_year experience_level employment_type                  job_title
## 2      2020                3               2 Machine Learning Scientist
## 3      2020                3               2          Big Data Engineer
## 4      2020                2               2       Product Data Analyst
## 5      2020                3               2  Machine Learning Engineer
## 6      2020                1               2               Data Analyst
## 7      2020                3               2        Lead Data Scientist
##   salary_in_usd remote_ratio company_location company_size
## 2        260000            0               JP            1
## 3        109024          0.5               GB            2
## 4         20000            0               HN            1
## 5        150000          0.5               US            3
## 6         72000            1               US            3
## 7        190000            1               US            1
```

---

### Outliers Checking

We will now proceed to examine outliers within the **salary_in_usd** variable. The rationale for this outlier analysis is rooted in the recognition that outliers may signify inaccuracies in data entry or unusual data points. By identifying and rectifying these outliers, we can uphold the precision and quality of the dataset.

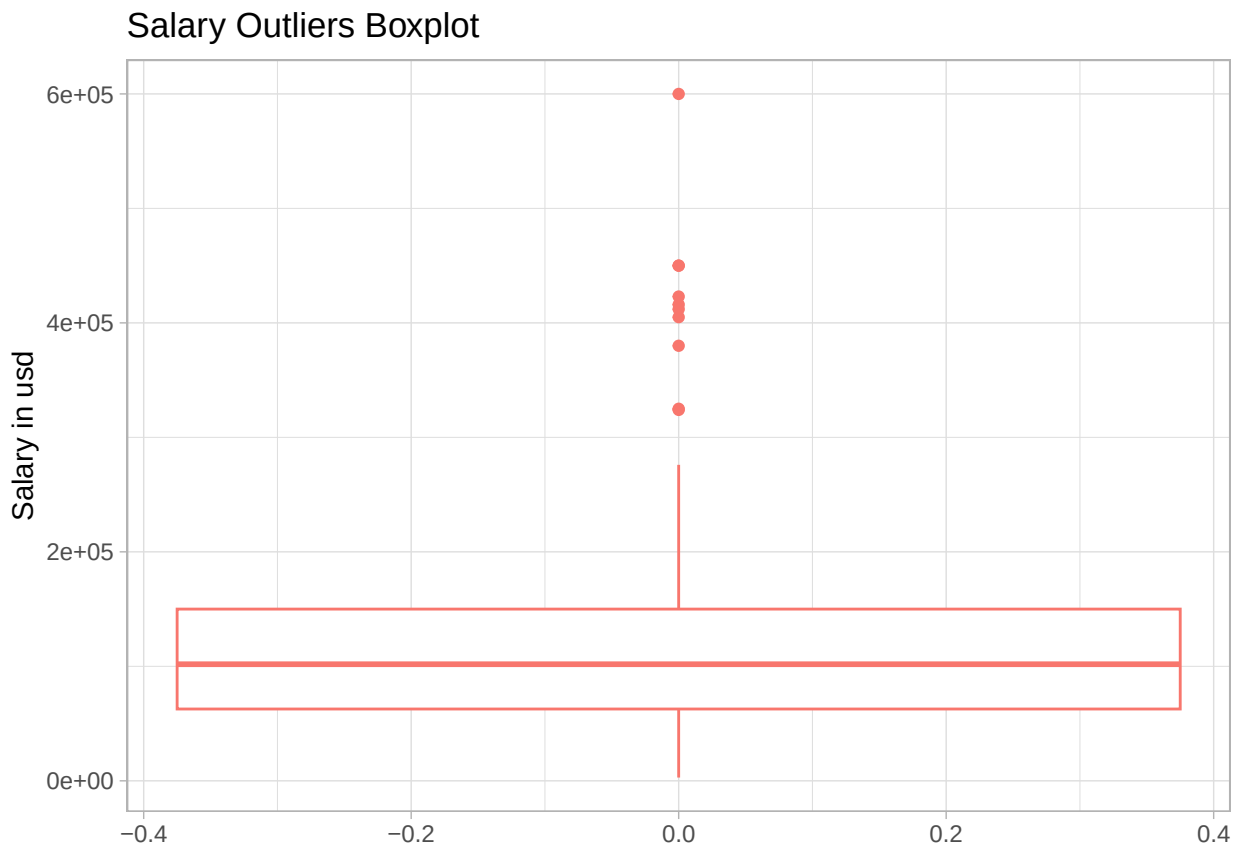
Furthermore, the detection of outliers offers valuable insights into the distribution of job salary data. It aids in the determination of whether the data adheres to a normal distribution or if there are exceptionally high or low values that could introduce bias into our analysis. Addressing outliers is of paramount importance since they have the potential to mislead results and impair the performance of predictive models, underscoring the need for their appropriate handling.

Additionally, outliers present in job salary data can shed light on compensation-related issues within an organization, possibly signaling instances of notably high or low salaries that warrant further investigation. This underscores the significance of our decision to scrutinize the **salary_in_usd** variable for potential outliers, given the potential presence of extreme values within this column.

```
outliers <- job_data %>%
  mutate(outlier = ifelse(salary_in_usd > quantile(salary_in_usd, 0.75) + 1.5 * IQR(salary_in_usd) |

table(outliers$outlier)
```

```
##
## Not Outlier     Outlier
##         596          10
```

```
ggplot(job_data, aes(y = job_data$salary_in_usd, color = "orange")) + geom_boxplot() +
  labs(title = "Salary Outliers Boxplot") + ylab("Salary in usd") +
  theme_light() + theme(legend.position = "none")
```




```
boxplot.stats(job_data$salary_in_usd)$out
```

```
##  [1] 325000 450000 412000 450000 423000 416000 600000 324000 380000 405000
```

the outlier in attribute **salary_in_usd** equal to 10 outliers.

---

### Normalization

The variable **salary_in_usd** exhibits a significant number of exceptionally large values. Consequently, we have opted to standardize this column using the min-max scaling method to ensure that the data falls within a more manageable and consistent range. This normalization process will enhance the interpretability and usability of the data for our analysis.

```
normalize_salary <- function (x) {return ((x-min(x))/ (max(x)))}
job_data$salary_in_usd <- normalize_salary(job_data$salary_in_usd)
head(job_data)
```

```
##   work_year experience_level employment_type                 job_title
## 2      2020                3               2 Machine Learning Scientist
## 3      2020                3               2          Big Data Engineer
## 4      2020                2               2       Product Data Analyst
```

```
## 5      2020                3               2  Machine Learning Engineer
## 6      2020                1               2               Data Analyst
## 7      2020                3               2       Lead Data Scientist
##   salary_in_usd remote_ratio company_location company_size
## 2    0.42856833            0               JP            1
## 3    0.17694167          0.5               GB            2
## 4    0.02856833            0               HN            1
## 5    0.24523500          0.5               US            3
## 6    0.11523500            1               US            3
## 7    0.31190167            1               US            1
```

---

5 Data mining technique

5.1 Regression technique

[2]

- Decision Tree part 1 ( this tree is different from the other tree )

**Applying Decision Tree Model**

```
partition_sizes <- c(0.6, 0.7, 0.8)
```

**Defining 3 different partition sizes**

```
attribute_selection_measures <- c("information", "gain", "gini")
```

**Defining 3 different attribute selection measures**

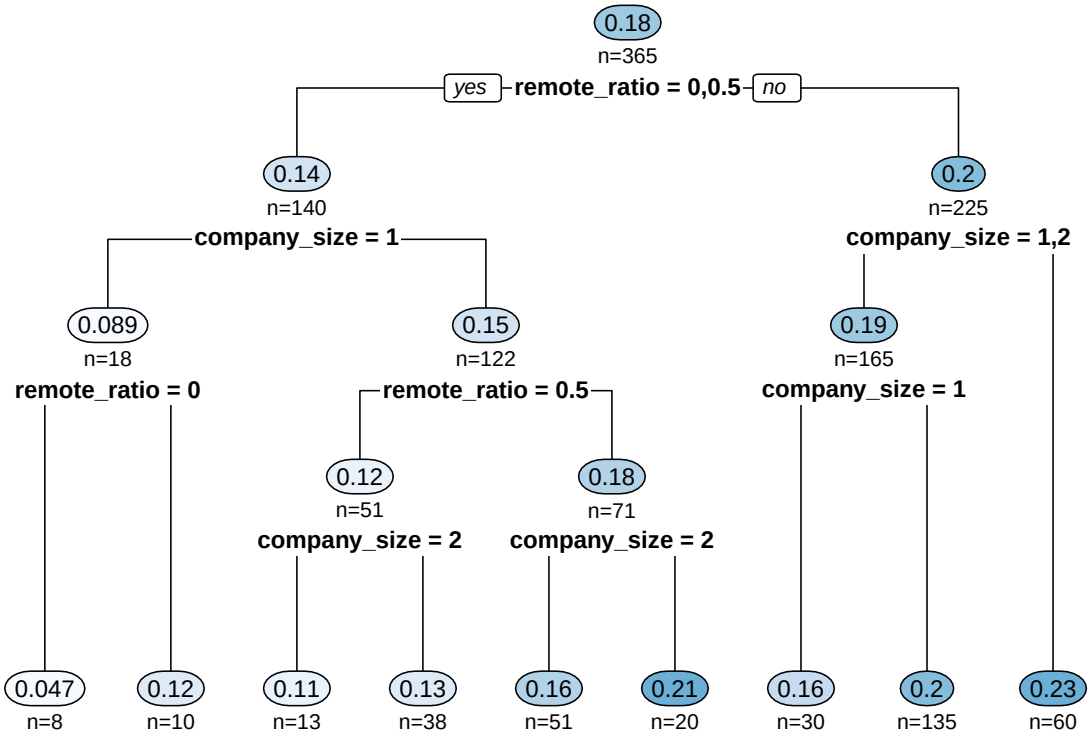
**Initializing results data frame for storing all 3 model results**

```
results <- data.frame(PartitionSize = numeric(),
                      AttributeSelection = character(),
                      RMSE = numeric())

job_data_1 <- job_data[, c(3, 5, 6, 8)]
```

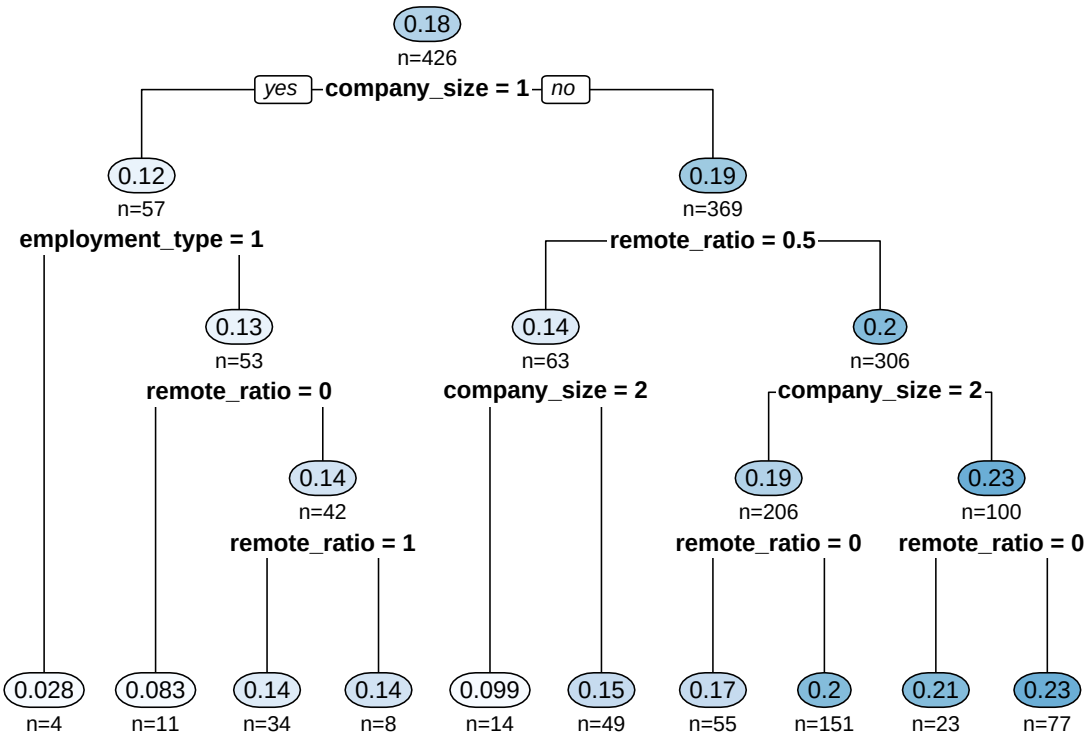
```
for (size in partition_sizes) {
  for (measure in attribute_selection_measures) {

    # Split the data into training and testing sets
```

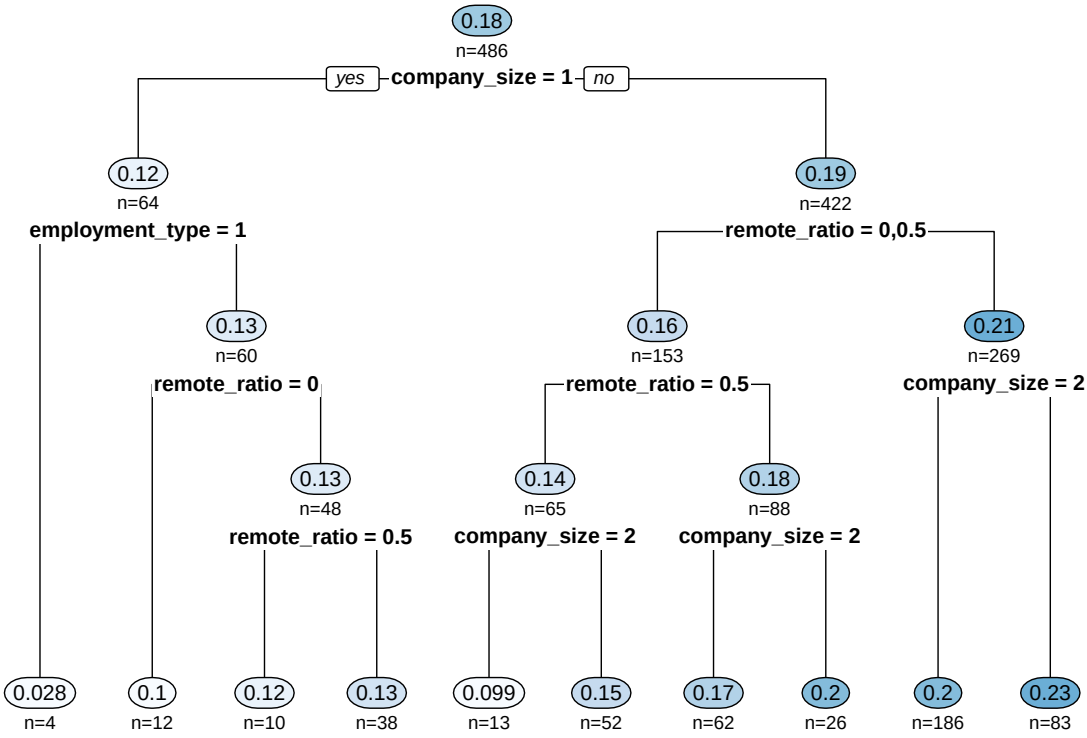
```
    set.seed(123)

    trainIndex <- createDataPartition(job_data_1$salary_in_usd, p = size, list = FALSE)
    trainData <- job_data_1[trainIndex, ]
    testData <- job_data_1[-trainIndex, ]

    # Train the decision tree regression model

    ctrl <- rpart.control(minsplit = 10, cp = 0)
    model <- rpart(salary_in_usd ~ ., data = trainData,
                   method = "anova", control = ctrl, parms = list(split = measure))

    # Visualize the model

    rpart.plot(model, type = 2, extra = 1, under = TRUE, fallen.leaves = TRUE)

    # Make predictions on the test set

    predictions <- predict(model, testData)

    # Calculate Root Mean Squared Error (RMSE) because our target variable is numeric that's
    #  why confusionMatrix will not work and in regression models we calculate RMSE for
    #  checking model performance and accuracy and in classification models we use
    #  confusionMatrix to calculate accuracy

    rmse <- sqrt(mean((testData$salary_in_usd - predictions)^2))

    # Add each model result to the data frame

    results <- rbind(results, data.frame(PartitionSize = size,
                                         AttributeSelection = measure, RMSE = rmse))
  }
}
```

**Training decision tree regression models with different partition sizes and attribute selection**




**measures**


0.18
n=426
yes
company_size = 1
no
0.12
n=57
0.19
n=369
employment_type = 1
remote_ratio = 0.5
0.13
n=53
0.14
n=63
0.2
n=306
remote_ratio = 0
company_size = 2
company_size = 2
0.14
n=42
0.19
n=206
0.23
n=100
remote_ratio = 1
remote_ratio = 0
remote_ratio = 0
0.028
n=4
0.083
n=11
0.14
n=34
0.14
n=8
0.099
n=14
0.15
n=49
0.17
n=55
0.2
n=151
0.21
n=23
0.23
n=77




```
print(results)
```

```
##   PartitionSize AttributeSelection      RMSE
## 1           0.6        information 0.1241714
## 2           0.6               gain 0.1241714
## 3           0.6               gini 0.1241714
## 4           0.7        information 0.1050549
## 5           0.7               gain 0.1050549
## 6           0.7               gini 0.1050549
## 7           0.8        information 0.1110464
## 8           0.8               gain 0.1110464
## 9           0.8               gini 0.1110464
```

Note: because we used Regression technique we can not calculate the Accuracy , precision , sensitivity and specificity we can calculate instead the Root Mean Square Error which is a measure of the average deviation between the predicted values of the model and the actual values in the dataset.

*another way to make a Regression technique with our dataset* - Decision Tree part2

Initialization:

```
library(rpart)
library(rpart.plot)
library(tidyverse)
data <- read.csv('salaries_data.csv')
```

1-categorize our features:

- company size

```
data <- data %>%
  mutate(
    company_size = case_when(
      company_size == "L" ~ "Large",
      company_size == "M" ~ "Medium",
      company_size == "S" ~ "Small",
      TRUE ~ "Uknown"
    )
  )
print(unique(data$company_size))
```

```
## [1] "Large"  "Small"  "Medium"
```

- employment type

```
uq_employment_type <- data %>%
  distinct(employment_type)
print(uq_employment_type)
```

```
##   employment_type
## 1              FT
## 2              CT
## 3              PT
## 4              FL
```

```
data <- data %>%
  mutate(
    employment_type = case_when(
      grepl("FT", (employment_type)) ~ "Full-Time",
      grepl("CT", (employment_type)) ~ "Contract",
      grepl("PT", (employment_type)) ~ "Part-Time",
      grepl("FL", (employment_type)) ~ "Flexible",
      TRUE ~ "Other"
    )
  )
print(unique(data$employment_type))
```

```
## [1] "Full-Time" "Contract"  "Part-Time" "Flexible"
```

- employee residence

```
print(unique(data$employee_residence))
```

```
##  [1] "DE" "JP" "GB" "HN" "US" "HU" "NZ" "FR" "IN" "PK" "PL" "PT" "CN" "GR" "AE"
## [16] "NL" "MX" "CA" "AT" "NG" "PH" "ES" "DK" "RU" "IT" "HR" "BG" "SG" "BR" "IQ"
## [31] "VN" "BE" "UA" "MT" "CL" "RO" "IR" "CO" "MD" "KE" "SI" "HK" "TR" "RS" "PR"
## [46] "LU" "JE" "CZ" "AR" "DZ" "TN" "MY" "EE" "AU" "BO" "IE" "CH"
```

```
data <- data %>%
  mutate(
    employee_residence = case_when(
      employee_residence %in% c("DE", "JP", "GB", "HN", "US", "HU", "NZ", "FR", "IN", "PK") ~ "Western C
      employee_residence %in% c("PL", "PT", "CN", "GR", "AE", "NL", "MX", "CA", "AT", "NG") ~ "Diverse E
      employee_residence %in% c("PH", "ES", "DK", "RU", "IT", "HR", "BG", "SG", "BR", "IQ") ~ "European
      employee_residence %in% c("VN", "BE", "UA", "MT", "CL", "RO", "IR", "CO", "MD", "KE", "SI") ~ "Div
      employee_residence %in% c("HK", "TR", "RS", "PR", "LU", "JE", "CZ", "AR", "DZ", "TN", "MY") ~ "Eur
      employee_residence %in% c("EE", "AU", "BO", "IE", "CH") ~ "European and Oceanian",
      TRUE ~ "Other"
    )
  )

print(unique(data$employee_residence))
```

```
## [1] "Western Countries"                   "Diverse European and Asian"
## [3] "European, Asian, and South American" "Diverse European and African"
## [5] "European and Asian"                  "European and Oceanian"
```

- job title

```
unique_job_titles <- data %>%
  distinct(job_title)

print(unique_job_titles)
```

```
##                                 job_title
## 1                          Data Scientist
## 2               Machine Learning Scientist
## 3                        Big Data Engineer
## 4                     Product Data Analyst
## 5                Machine Learning Engineer
## 6                           Data Analyst
## 7                     Lead Data Scientist
## 8                    Business Data Analyst
## 9                       Lead Data Engineer
## 10                      Lead Data Analyst
## 11                          Data Engineer
## 12                 Data Science Consultant
## 13                          BI Data Analyst
## 14                Director of Data Science
## 15                      Research Scientist
## 16                Machine Learning Manager
## 17                Data Engineering Manager
## 18 Machine Learning Infrastructure Engineer
## 19                              ML Engineer
## 20                           AI Scientist
## 21                Computer Vision Engineer
## 22                Principal Data Scientist
## 23                    Data Science Manager
## 24                           Head of Data
## 25           3D Computer Vision Researcher
```

```
## 26                  Data Analytics Engineer
## 27                   Applied Data Scientist
## 28                   Marketing Data Analyst
## 29                      Cloud Data Engineer
## 30                   Financial Data Analyst
## 31        Computer Vision Software Engineer
## 32             Director of Data Engineering
## 33                    Data Science Engineer
## 34                  Principal Data Engineer
## 35              Machine Learning Developer
## 36       Applied Machine Learning Scientist
## 37                  Data Analytics Manager
## 38                    Head of Data Science
## 39                         Data Specialist
## 40                          Data Architect
## 41                    Finance Data Analyst
## 42                   Principal Data Analyst
## 43                       Big Data Architect
## 44                    Staff Data Scientist
## 45                      Analytics Engineer
## 46                           ETL Developer
## 47                Head of Machine Learning
## 48                             NLP Engineer
## 49           Lead Machine Learning Engineer
## 50                     Data Analytics Lead
```

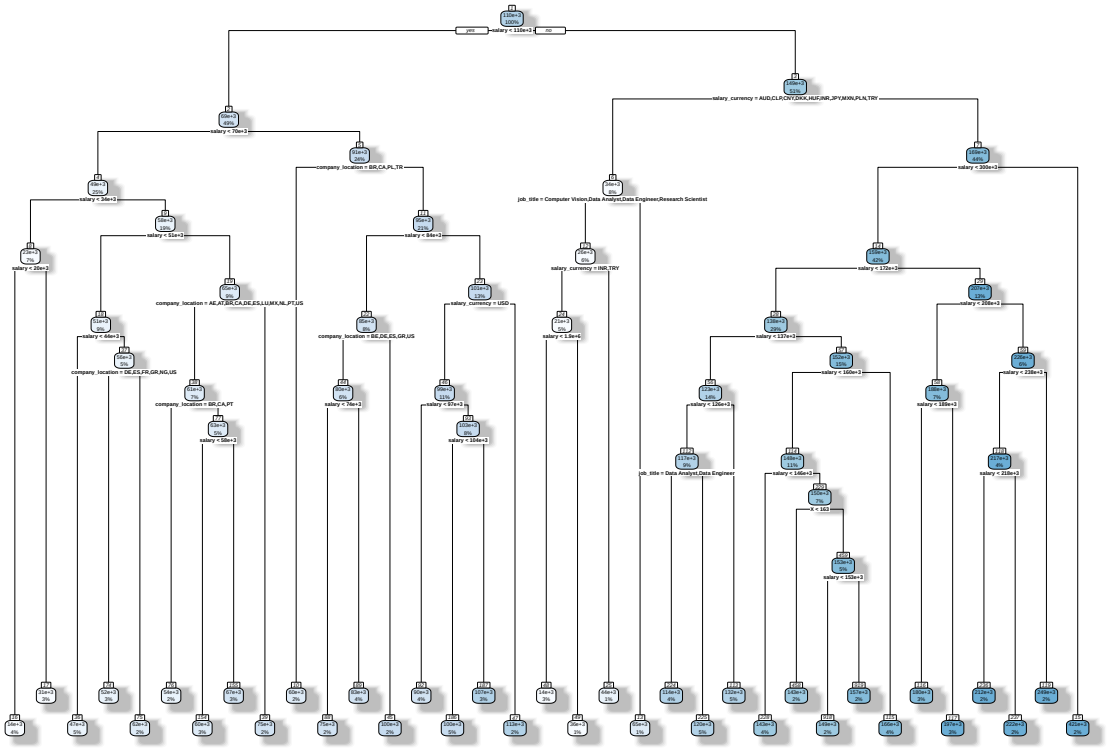
```
data <- data %>%
  mutate(
    job_title = case_when(
      str_detect(tolower(job_title), "data scientist|research scientist|scientist|science") ~ "Research
      str_detect(tolower(job_title), "machine learning engineer|ml engineer|ai scientist") ~ "ML Engine
      str_detect(tolower(job_title), "data engineer|etl|data engineering manager") ~ "Data Engineer",
      str_detect(tolower(job_title), "data analyst|bi|business data analyst|financial data analyst") ~
      str_detect(tolower(job_title), "director of data science|head of data science|data science manage
      str_detect(tolower(job_title), "computer vision|3d computer vision researcher|computer vision sof
      str_detect(tolower(job_title), "data architect|big data architect") ~ "Data Architect",
      str_detect(tolower(job_title), "machine learning developer|applied machine learning scientist") ~
      str_detect(tolower(job_title), "analytics engineer|data analytics manager") ~ "Analytics Engineer
      str_detect(tolower(job_title), "head of machine learning|lead machine learning engineer") ~ "Leade
      TRUE ~ "Other"
    ),

  )

# Print the unique new categories
print(unique(data$job_title))
```

```
##  [1] "Research Scientist" "Data Engineer"      "Data Analyst"
##  [4] "ML Engineer"        "Other"              "Computer Vision"
##  [7] "Analytics Engineer" "ML Developer"       "Data Architect"
## [10] "Leadership in ML"
```

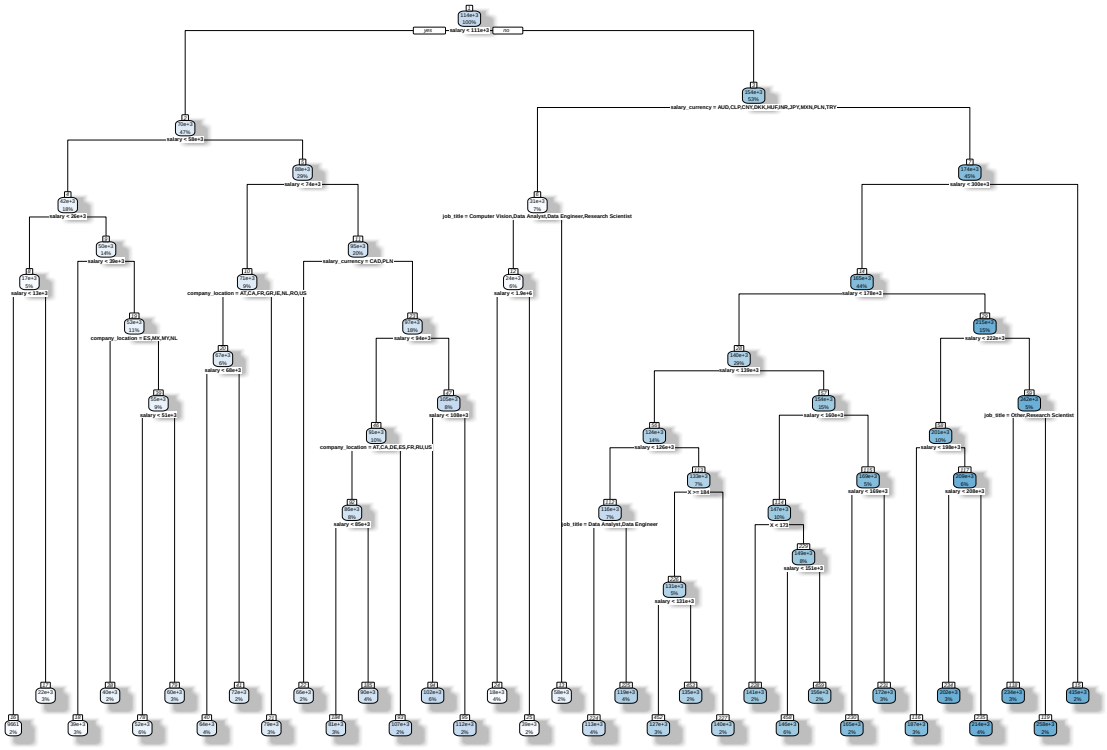
2- decision tree

```r
library(rpart)
library(rpart.plot)

train_and_plot_tree <- function(train_data, test_data, criterion, size, tolerance) {
  test_data$job_title <- factor(test_data$job_title, levels = levels(train_data$job_title))
  test_data$salary_currency <- factor(test_data$salary_currency, levels = levels(train_data$salary_curr
  test_data$company_location <- factor(test_data$company_location, levels = levels(train_data$company_l

  tree_model <- rpart(salary_in_usd ~ ., data = train_data, method = "anova", cp = 0.0001)

  cat(paste("\n\nDecision Tree for Partition Size: ", size, " and Criterion: ", criterion, "\n"))
  rpart.plot(tree_model, box.palette = "auto", shadow.col = "gray", nn = TRUE)

  predictions <- predict(tree_model, newdata = test_data)

  correct_predictions <- abs(predictions - test_data$salary_in_usd) <= tolerance
  accuracy <- sum(correct_predictions) / length(test_data$salary_in_usd)
  cat(paste("\nAccuracy for Partition Size: ", size, " and Criterion: ", criterion, " within $", tolera
}


set.seed(2024)
train_indices_80 <- sample(1:nrow(data), 0.8 * nrow(data), replace = FALSE)
train_data_80 <- data[train_indices_80, ]
test_data_80 <- data[-train_indices_80, ]

train_indices_70 <- sample(1:nrow(data), 0.7 * nrow(data), replace = FALSE)
train_data_70 <- data[train_indices_70, ]
test_data_70 <- data[-train_indices_70, ]

train_indices_60 <- sample(1:nrow(data), 0.6 * nrow(data), replace = FALSE)
train_data_60 <- data[train_indices_60, ]
test_data_60 <- data[-train_indices_60, ]

criteria <- c("anova", "gini", "dev")
tolerance <- 10000

for (criterion in criteria) {
  train_and_plot_tree(train_data_80, test_data_80, criterion, "80%", tolerance)
  train_and_plot_tree(train_data_70, test_data_70, criterion, "70%", tolerance)
  train_and_plot_tree(train_data_60, test_data_60, criterion, "60%", tolerance)
}
```

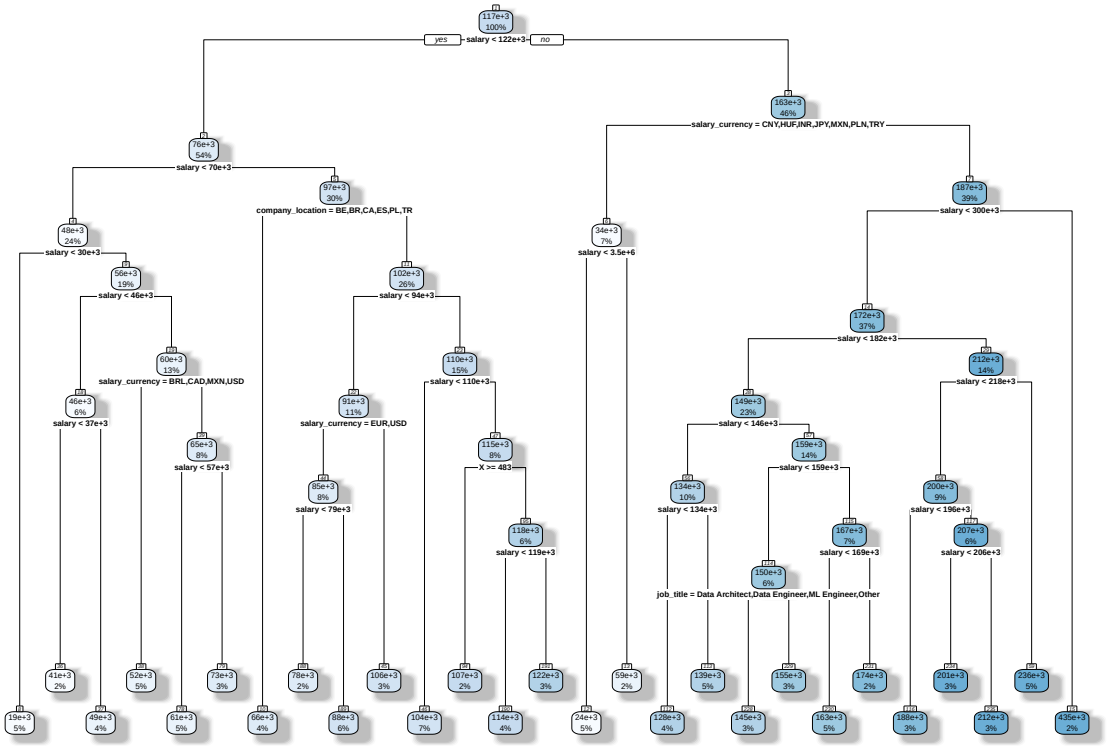
```
##
##
## Decision Tree for Partition Size:  80%  and Criterion:  anova
```



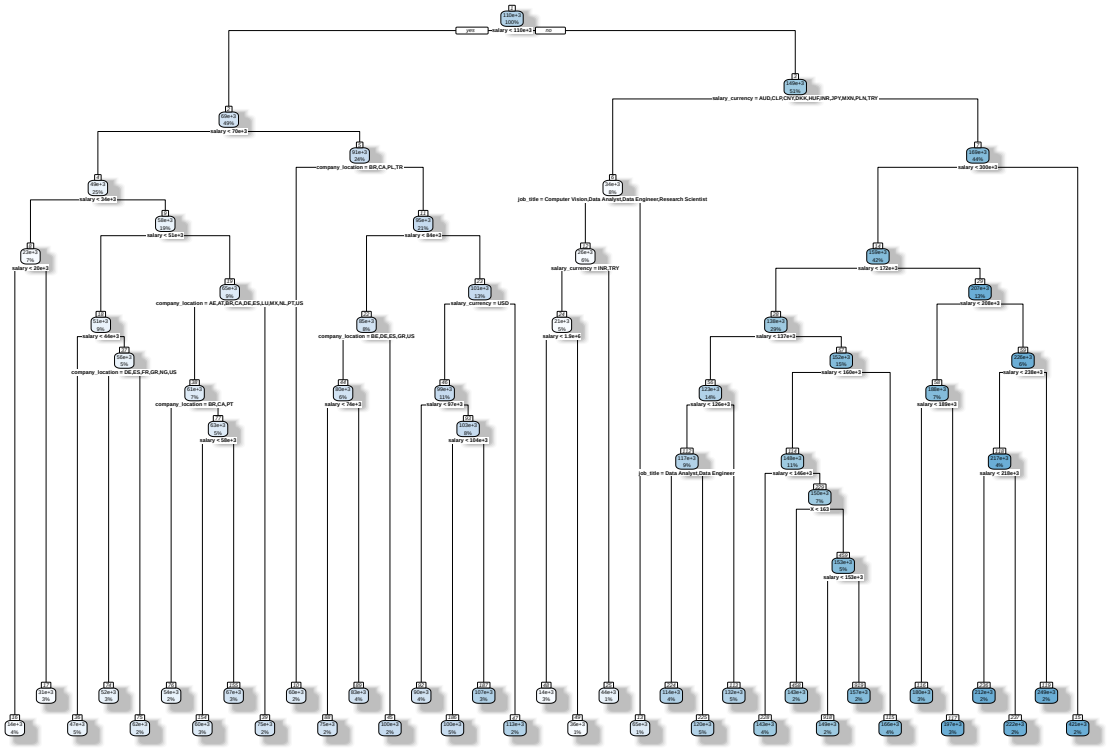

```
##
## Accuracy for Partition Size:  80%  and Criterion:  anova  within $ 10000  is  75.4098360655738 %
##
##
## Decision Tree for Partition Size:  70%  and Criterion:  anova
```



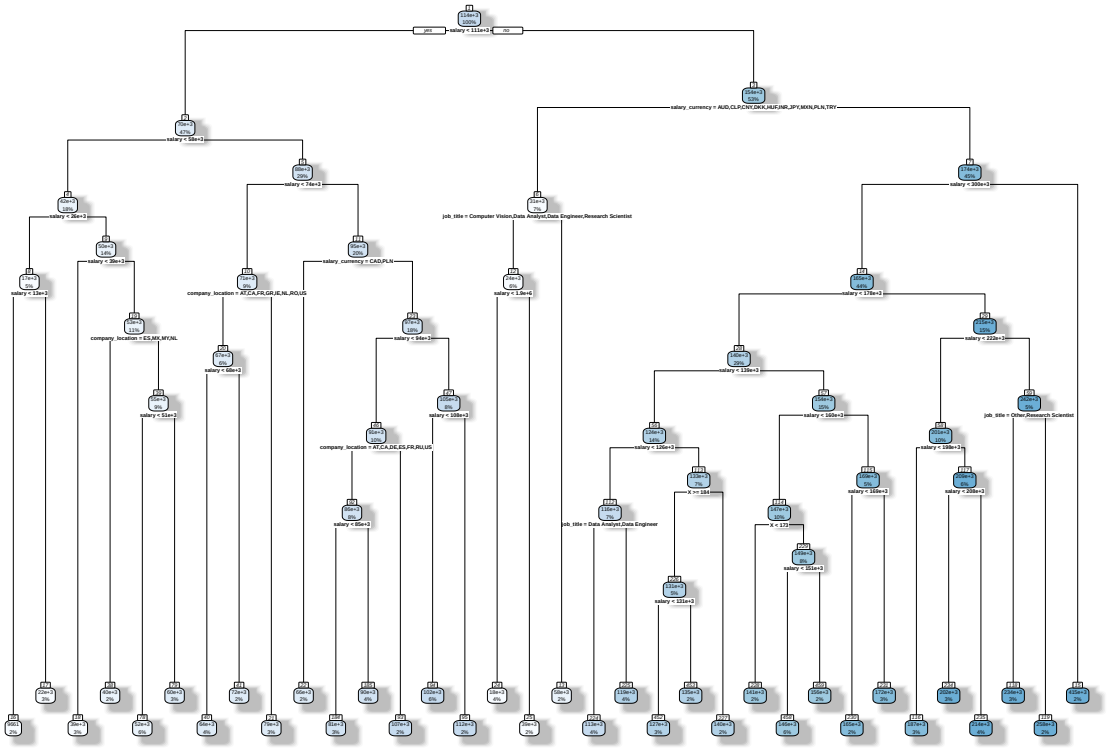
```
##
## Accuracy for Partition Size:  70%  and Criterion:  anova  within $ 10000  is  75.9562841530055 %
##
##
## Decision Tree for Partition Size:  60%  and Criterion:  anova
```



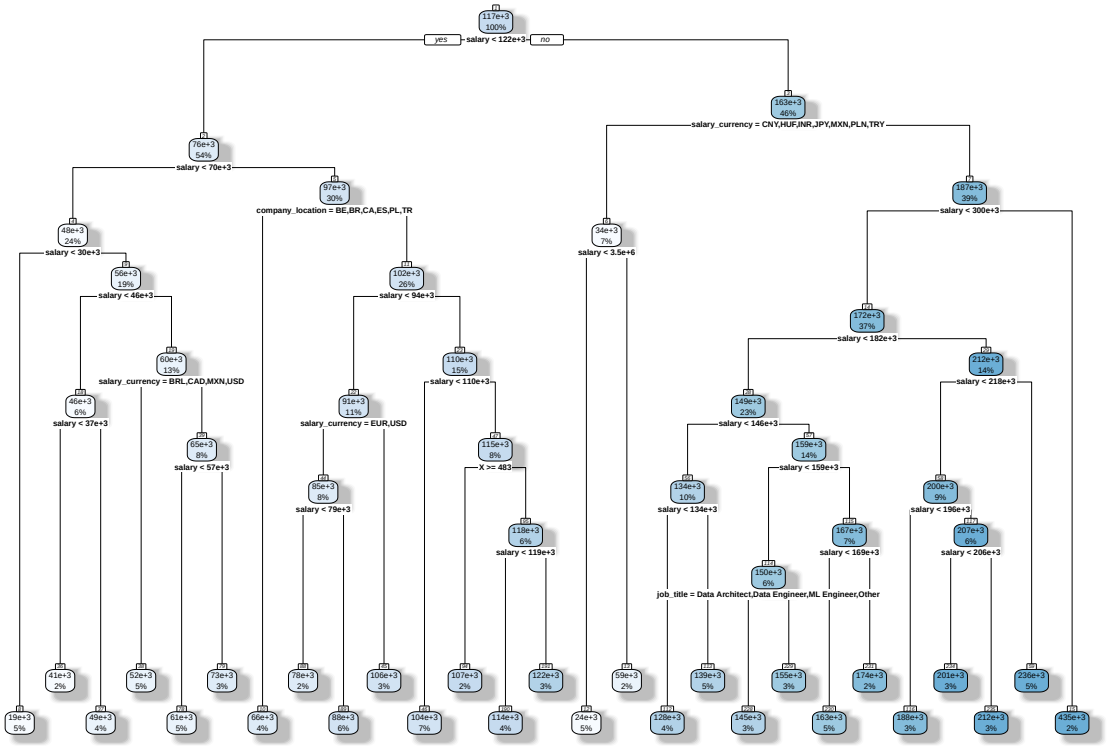

```
##
## Accuracy for Partition Size:  60%  and Criterion:  anova  within $ 10000  is  72.8395061728395 %
##
##
## Decision Tree for Partition Size:  80%  and Criterion:  gini
```



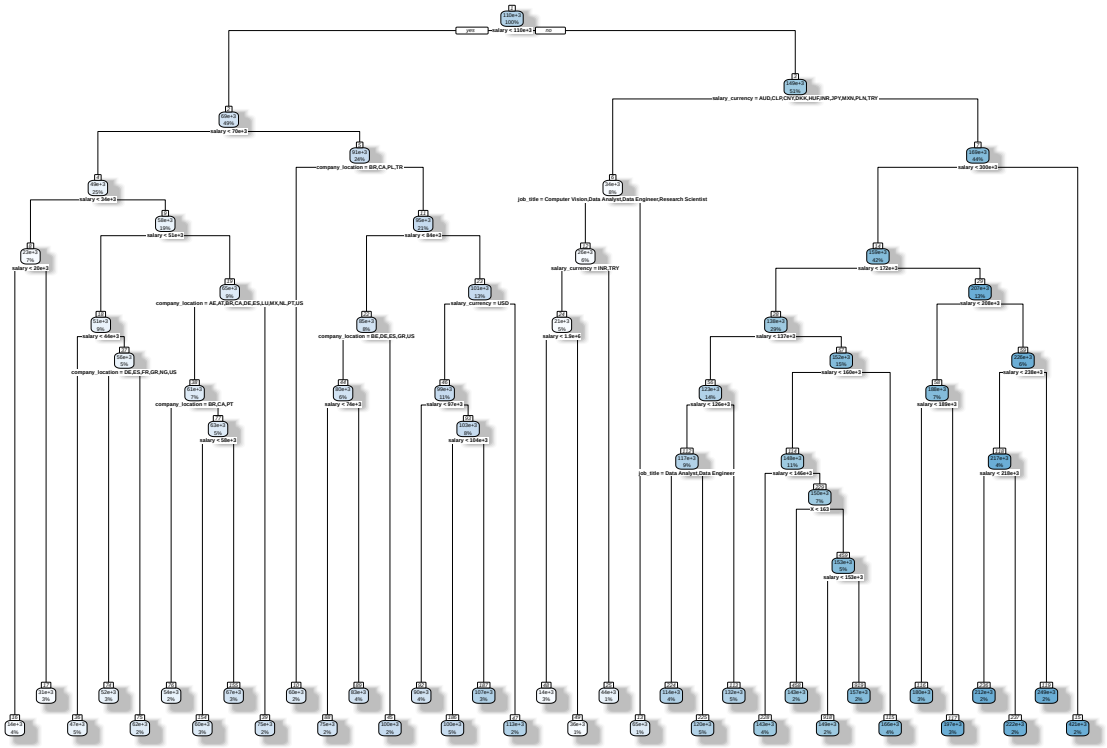

```
##
## Accuracy for Partition Size:  80%  and Criterion:  gini  within $ 10000  is  75.4098360655738 %
##
##
## Decision Tree for Partition Size:  70%  and Criterion:  gini
```



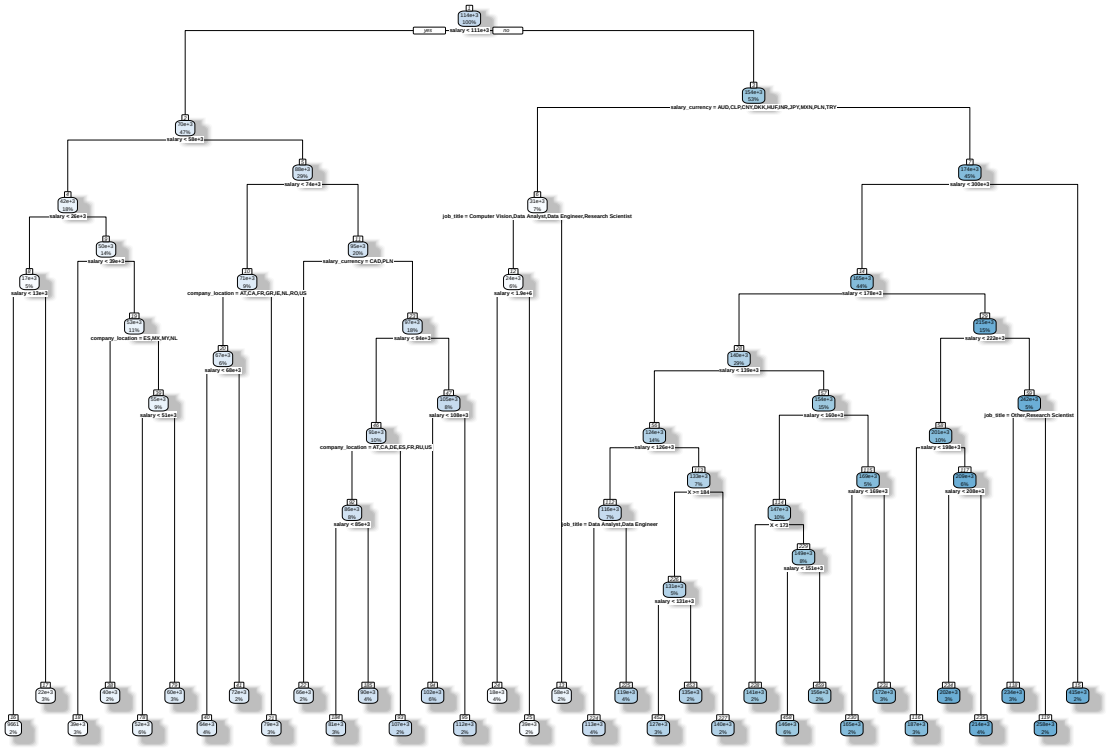
```
##
## Accuracy for Partition Size:  70%  and Criterion:  gini  within $ 10000  is  75.9562841530055 %
##
##
## Decision Tree for Partition Size:  60%  and Criterion:  gini
```



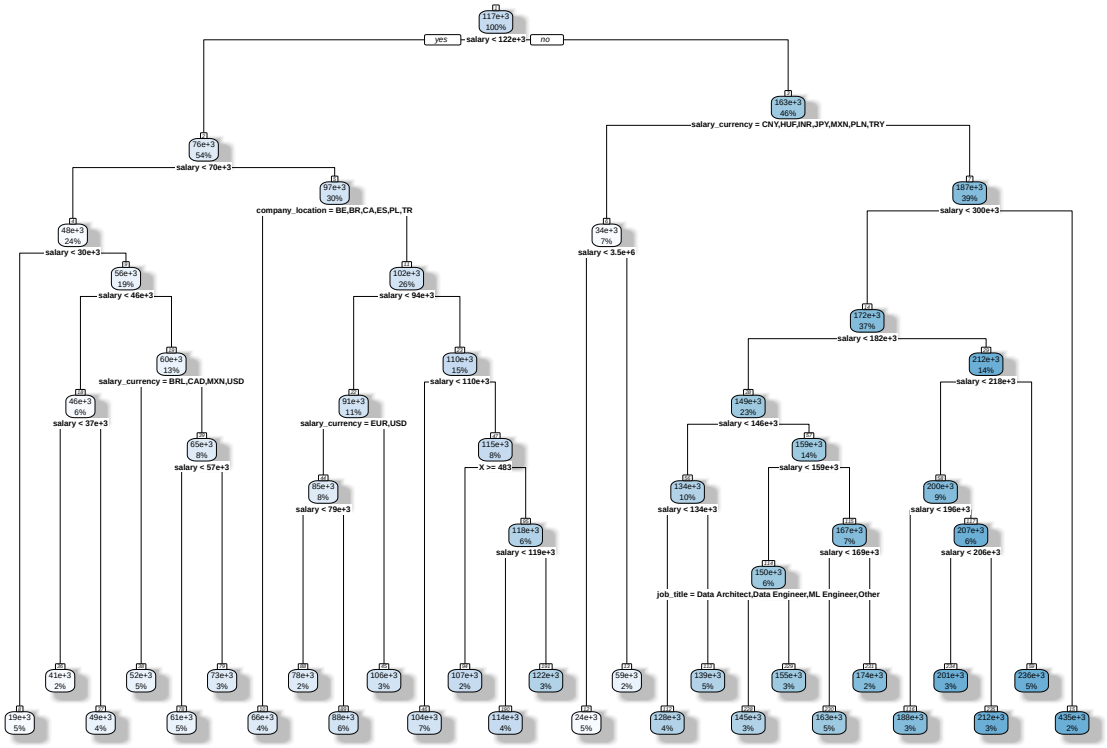

```
##
## Accuracy for Partition Size:  60%  and Criterion:  gini  within $ 10000  is  72.8395061728395 %
##
##
## Decision Tree for Partition Size:  80%  and Criterion:  dev
```




```
##
## Accuracy for Partition Size:  80%  and Criterion:  dev  within $ 10000  is  75.4098360655738 %
##
##
## Decision Tree for Partition Size:  70%  and Criterion:  dev
```



```
##
## Accuracy for Partition Size:  70%  and Criterion:  dev  within $ 10000  is  75.9562841530055 %
##
##
## Decision Tree for Partition Size:  60%  and Criterion:  dev
```




```
##
## Accuracy for Partition Size:  60%  and Criterion:  dev  within $ 10000  is  72.8395061728395 %
```

here is the accuracy for the decision tree 2 for each partioning size (60, 70, 80):

Decision Tree for Partition Size: 80% and Criterion: anova

Accuracy for Partition Size: 80% and Criterion: anova within $ 10000 is 75.4098360655738 %

Decision Tree for Partition Size: 70% and Criterion: anova

Accuracy for Partition Size: 70% and Criterion: anova within $ 10000 is 75.9562841530055 %

Decision Tree for Partition Size: 60% and Criterion: anova

Accuracy for Partition Size: 60% and Criterion: anova within $ 10000 is 72.8395061728395 %

Decision Tree for Partition Size: 80% and Criterion: gini

Accuracy for Partition Size: 80% and Criterion: gini within $ 10000 is 75.4098360655738 %

Decision Tree for Partition Size: 70% and Criterion: gini

Accuracy for Partition Size: 70% and Criterion: gini within $ 10000 is 75.9562841530055 %

Decision Tree for Partition Size: 60% and Criterion: gini

Accuracy for Partition Size: 60% and Criterion: gini within $ 10000 is 72.8395061728395 %

Decision Tree for Partition Size: 80% and Criterion: dev

Accuracy for Partition Size: 80% and Criterion: dev within $ 10000 is 75.4098360655738 %

Decision Tree for Partition Size: 70% and Criterion: dev

Accuracy for Partition Size: 70% and Criterion: dev within $ 10000 is 75.9562841530055 %

Decision Tree for Partition Size: 60% and Criterion: dev

Accuracy for Partition Size: 60% and Criterion: dev within $ 10000 is 72.8395061728395 %

---

5.2 clustring technique

- clustering model part 1

```
library(dplyr)
library(cluster)
library(factoextra)
```

```
df <- read.csv("salaries_data.csv")

# Assuming your dataframe is named df
# ... (any additional code for data cleaning or handling missing values)

# Select columns for clustering
selected_columns <- df[, c("work_year", "salary_in_usd", "remote_ratio", "experience_level", "employment

# Check for missing values
if (any(is.na(selected_columns))) {
  stop("There are missing values in the selected columns. Please handle them before clustering.")
}

# Check data types
str(selected_columns)
```

```
## 'data.frame':    607 obs. of  6 variables:
##  $ work_year       : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ salary_in_usd   : int  79833 260000 109024 20000 150000 72000 190000 35735 135000 125000 ...
##  $ remote_ratio    : int  0 0 50 0 50 100 100 50 100 50 ...
##  $ experience_level: chr  "MI" "SE" "SE" "MI" ...
##  $ employment_type : chr  "FT" "FT" "FT" "FT" ...
##  $ company_size    : chr  "L" "S" "M" "S" ...
```

```
# One-hot encode categorical variables
encoded_data <- model.matrix(~ . - 1, data = selected_columns)

# Scale the encoded data
scaled_data <- scale(encoded_data)

# Choose K values
k_values <- c(2, 3, 4)

# Initialize vectors to store results
silhouette_scores <- numeric(length = length(k_values))
```

```r
within_cluster_ss <- numeric(length = length(k_values))
results <- list()

# Apply K-means clustering
for (i in seq_along(k_values)) {
  k <- k_values[i]

  set.seed(2024)
  kmeans_result <- kmeans(scaled_data, centers = k, nstart = 50)

  # Check if the algorithm converged
  if (!is.null(kmeans_result$convergence) && kmeans_result$convergence != 0) {
    cat("K-means did not converge for K =", k, "\n")
  } else {
    results[[as.character(k)]] <- kmeans_result

    # Calculate silhouette score
    silhouette_scores[i] <- silhouette(kmeans_result$cluster, dist(scaled_data))

    # Calculate total within-cluster sum of squares
    within_cluster_ss[i] <- kmeans_result$tot.withinss
  }
}
```

```r
# Print silhouette scores and within-cluster sum of squares
cat("Silhouette Scores:", silhouette_scores, "\n")
```

```
## Silhouette Scores: 1 3 3
```

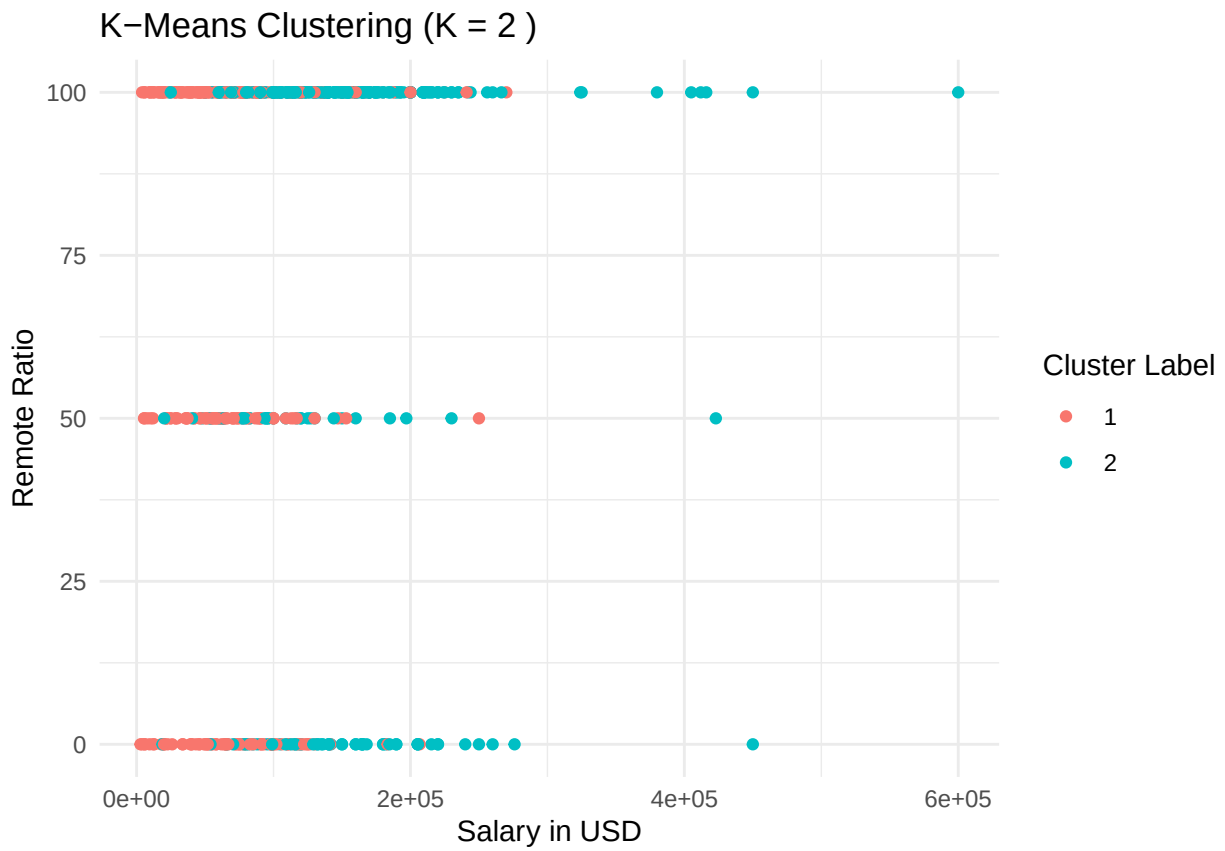
```r
cat("Within-Cluster Sum of Squares:", within_cluster_ss, "\n")
```

```
## Within-Cluster Sum of Squares: 6035.376 5106.642 4246.153
```

```r
# Choose the best K based on the silhouette score
best_k <- which.max(silhouette_scores)

# Assign cluster labels to the original dataframe
df$cluster_label <- as.factor(results[[as.character(best_k)]]$cluster)

# Print the number of clusters for the best K
cat("Number of clusters for the best K =", best_k, ":", length(unique(results[[as.character(best_k)]]$cl
```

```
## Number of clusters for the best K = 2 : 2
```

```r
# Plot the clusters
ggplot(df, aes(x = salary_in_usd, y = remote_ratio, color = cluster_label)) +
  geom_point() +
  labs(title = paste("K-Means Clustering (K =", best_k, ")"),
       x = "Salary in USD",
       y = "Remote Ratio",
       color = "Cluster Label") +
  theme_minimal()
```




**Applying K-means clustering**

**Specifying the different values of K number of clusters ,**

## and trying diffrent method for clustring

```
k_values <- c(3, 5, 7)
```

- clustering model part 2

```
silhouette_results <- data.frame(K = numeric(), Silhouette = numeric())
wss_results <- data.frame(K = numeric(), WSS = numeric())
precision_results <- data.frame(K = numeric(), Precision = numeric())
recall_results <- data.frame(K = numeric(), Recall = numeric())
```

```
data_1 <- job_data[, c(1, 2, 3, 5, 6, 8)]

data_1$work_year <- as.numeric(data_1$work_year)
```

```
data_1$experience_level <- as.numeric(data_1$experience_level)
data_1$employment_type <- as.numeric(data_1$employment_type)
data_1$salary_in_usd <- as.numeric(data_1$salary_in_usd)
data_1$remote_ratio <- as.numeric(data_1$remote_ratio)
data_1$company_size <- as.numeric(data_1$company_size)
str(data_1)
```

**Initializing the results data frames for storing all model results**

```
## 'data.frame':    606 obs. of  6 variables:
##  $ work_year       : num  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ experience_level: num  3 3 2 3 1 3 2 2 3 1 ...
##  $ employment_type : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ salary_in_usd   : num  0.4286 0.1769 0.0286 0.2452 0.1152 ...
##  $ remote_ratio    : num  1 2 1 2 3 3 2 3 2 1 ...
##  $ company_size    : num  1 2 1 3 3 1 3 3 1 1 ...
```

```
for (k in k_values) {

  # Create the cluster model

  kmeans_model <- kmeans(data_1, centers = k)

  # Calculate the Silhouette coefficient

  silhouette <- silhouette(kmeans_model$cluster, dist(data_1))


  # Calculate the Within-Cluster Sum of Squares (WSS)

  wss <- kmeans_model$tot.withinss

  # Add results to the data frames

  silhouette_results <- rbind(silhouette_results, data.frame(K = k,
                              Silhouette = mean(silhouette[, "sil_width"])))

  wss_results <- rbind(wss_results, data.frame(K = k, WSS = wss))

}

print("Silhouette Coefficient:")
```

**Loop over different K values and perform K-means clustering**

```
## [1] "Silhouette Coefficient:"
```

```
print(silhouette_results)
```

```
##   K Silhouette
## 1 3  0.3529762
## 2 5  0.3796629
## 3 7  0.4067098
```

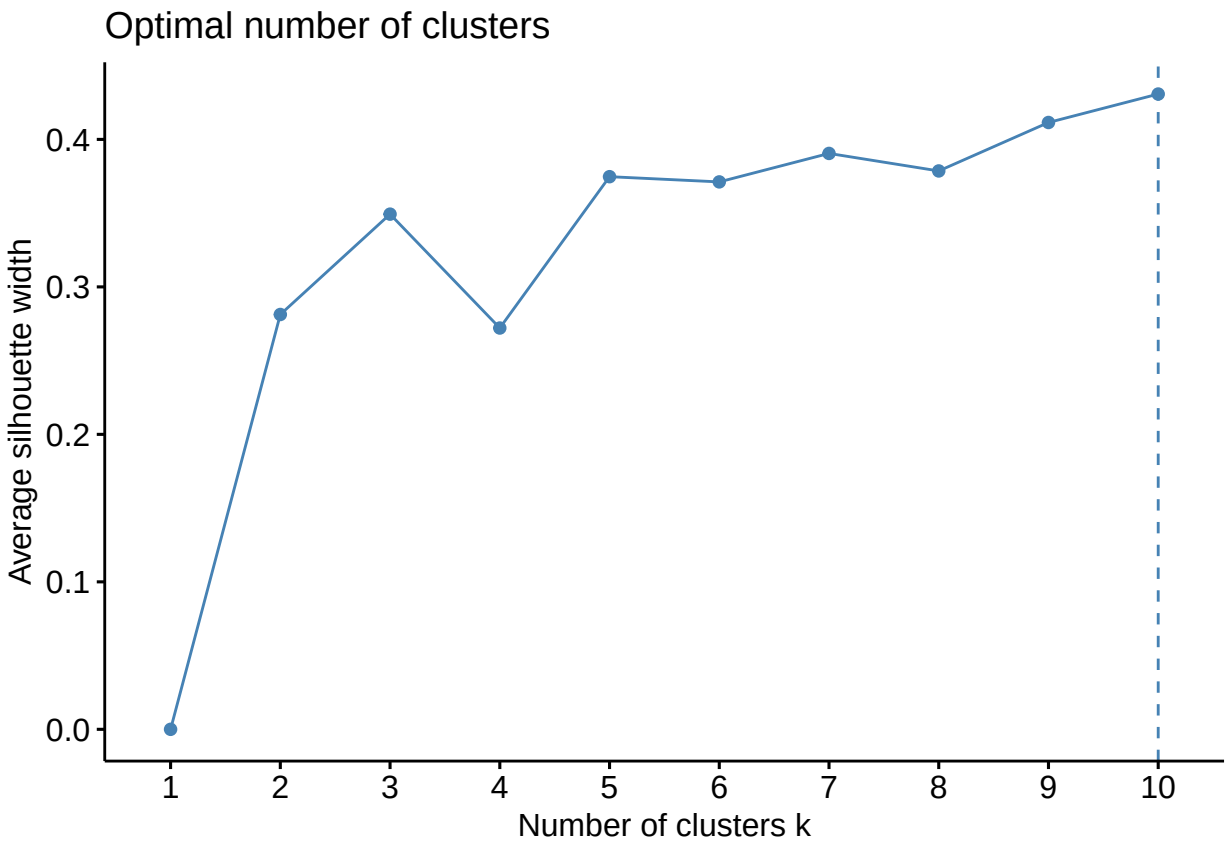
```
print("Within-Cluster Sum of Squares (WSS):")
```

```
## [1] "Within-Cluster Sum of Squares (WSS):"
```

```
print(wss_results)
```

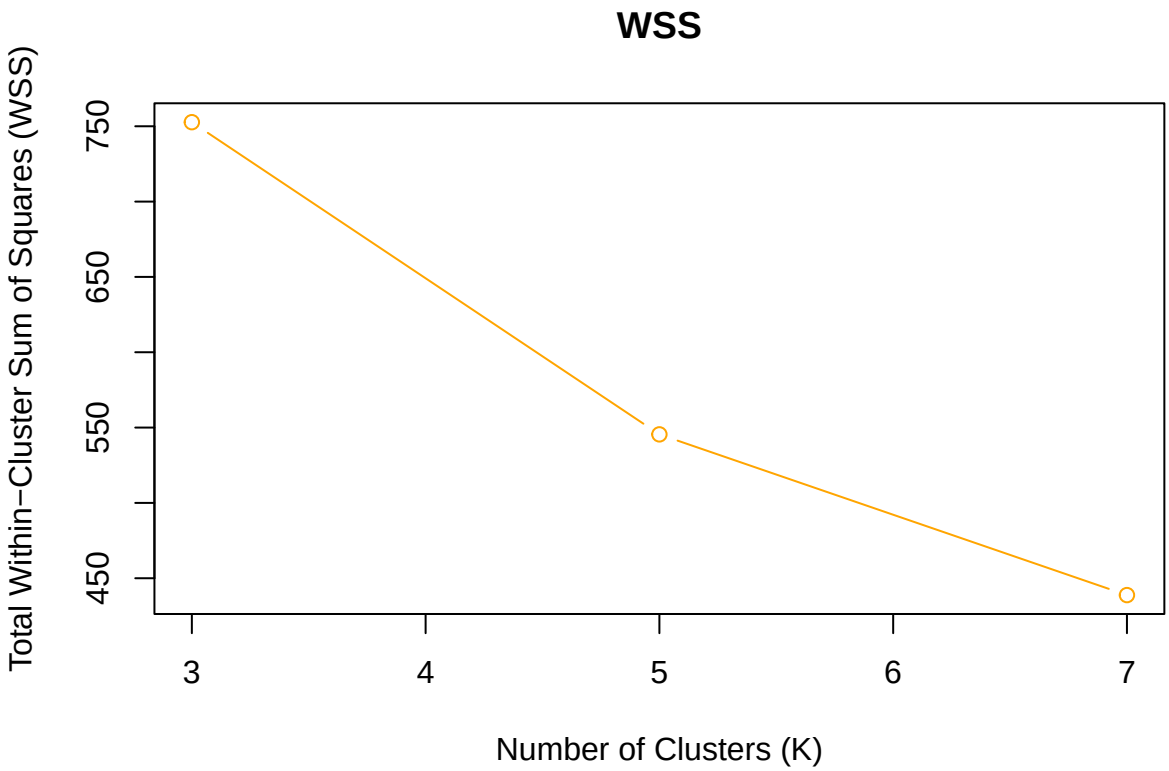
```
##   K      WSS
## 1 3 752.7008
## 2 5 545.4850
## 3 7 438.8104
```

```
# Silhouette Plot


fviz_nbclust(data_1, kmeans, method = "silhouette")
```




```
# Elbow Plot (WSS)

plot(wss_results$K, wss_results$WSS, type = "b", col = "orange",
```

```
    xlab = "Number of Clusters (K)",
    ylab = "Total Within-Cluster Sum of Squares (WSS)",
    main = "WSS")
```




6 Evaluation and Comparison

Note: because we used Regression technique we can not calculate the Accuracy , precision , sensitivity and specificity we can calculate instead the Root Mean Square Error which is a measure of the average deviation between the predicted values of the model and the actual values in the dataset.

| | 60% training set 40%testing set | 70% training set 30%testing set | 80% training set 20%testing set |
|---|---|---|---|
| Accuracy (Tree 2) | 72.8395061728395 % | 75.9562841530055 % | 75.4098360655738 % |
| RMSE (Tree 1) | 0.1241714 | 0.1050549 | 0.1110464 |

- Table for clustering part 1

| | k=2 | k=3 | k=4 |
|---|---|---|---|
| Average Silhouette width | 1 | 3 | 3 |

| | k=2 | k=3 | k=4 |
|---|---|---|---|
| total within-cluster sum of square | 6035.376 | 5106.642 | 4246.153 |
| Visualization | in the figures above | -- | -- |

- Table for clustering part 2

| | k=3 | k=5 | k=7 |
|---|---|---|---|
| Average Silhouette width | 0.3519878 | 0.3217594 | 0.4063718 |
| total within-cluster sum of square | 758.8789 | 652.4071 | 444.8279 |
| Visualization | in the figures above | -- | -- |

7 findings

Findings in Regression :

1- Choosing best decision tree model part 1

To identify the most suitable decision tree model, it's essential to assess the Root Mean Square Error (RMSE) values. Smaller RMSE values are indicative of superior model performance, and our aim is to opt for the model with the lowest RMSE. Upon reviewing the results in the decision tree model table provided, we observe that the RMSE values remain consistent across different attribute selection methods (information, gain, and gini) for each partition size.

In light of these findings, the optimal choice is to select Model 1, employing a partition size of 0.7 This decision is guided by the fact that Model 1 exhibits the most favorable RMSE value, which stands at 0.1050549

2- Choosing best decision tree model part 2

We made another tree that uses three splitting criterion used in the decision tree algorithm "anova", "gini", and "dev" and then we measure the accuracy of each criterion . each criterion means:

"anova": This method, called Analysis of Variance (ANOVA) splitting, uses the F-statistic to decide where to split. It measures how good the splits are and chooses the split that reduces the differences the most.

"gini": This method, known as Gini impurity splitting, looks at the chance of wrongly classifying something if it was labeled randomly based on the labels in a group. It calculates the Gini index for each possible split and picks the split with the smallest Gini index.

"dev": This method, called deviance splitting, uses deviance as a measure of how well the model fits. It's often used in logistic regression but can also be used in decision tree models. For each potential split, it calculates deviance, and the split with the lowest deviance is chosen. We set the tolerance value to be 1000 tolerance value implies that the model's predictions will be considered accurate if the absolute difference between the predicted salary and the actual salary is within 10000 units if the absolute difference exceeds 10000 units, the prediction will be considered inaccurate. In light of these findings, the optimal choice is to select Model 1, employing a partition size of 0.7 This decision is guided by the fact that Model 1 exhibits the most favorable accuracy value, which stands at 75.9562841530055 %

Findings in Clustring :

3- Choosing best K mean clustering model part 1

When choosing the best K value for K-means clustering, we should consider both the Silhouette score and the Total Within-Cluster Sum of Squares (WSS). A higher Silhouette score indicates better cluster separation and cohesion, while a lower WSS suggests tighter and more compact clusters.

in this clustering algorithm we've selected the attributes that we find more interesting in our study to see what the best K mean when we only do the clustering in these attributes which is ("work_year", "salary_in_usd", "remote_ratio", "experience_level", "employment_type", "company_size").

and here is the result for the selected attributes:

in the provided table:

Silhouette Scores:

- K = 2: Silhouette score = 1
- K = 3: Silhouette score = 3
- K = 4: Silhouette score = 3

Total WSS (Within-Cluster Sum of Squares):

- K = 2: WSS = 6035.376
- K = 3: WSS = 5106.642
- K = 4: WSS = 4246.153

4- Choosing best K mean clustering model part 2

When choosing the best K value for K-means clustering, we should consider both the Silhouette score and the Total Within-Cluster Sum of Squares (WSS). A higher Silhouette score indicates better cluster separation and cohesion, while a lower WSS suggests tighter and more compact clusters.

In the provided tables:

Silhouette Scores:

- K = 3: Silhouette score = 0.3519878
- K = 5: Silhouette score = 0.3217594
- K = 7: Silhouette score = 0.4063718

Total WSS (Within-Cluster Sum of Squares):

- K = 3: WSS = 758.8789
- K = 5: WSS = 652.4071
- K = 7: WSS = 444.8279

A good K value typically strikes a balance between a high Silhouette score and a low WSS. In this case, K = 7 has the highest Silhouette score (0.4063718), which indicates good cluster separation. Additionally, K = 7 has the lowest WSS (444.8279), indicating tight and compact clusters.

Based on these metrics, K = 7 appears to be a good choice for the number of clusters in your K-means clustering. It balances the trade-off between cluster separation and compactness. However, the choice of K should also consider the context and objectives of our analysis.

- Our Solution

  the salary levels in the field of data science, where salaries are crucial, we've created an easy-to-use prediction model. This model uses a simple way to figure out what affects the salaries of data scientists. It helps individuals and businesses understand market trends and supports fair pay practices.

Our approach involves making a model with a clear 70-30 split, focusing on important details like experience, job type, years worked and more. What makes our solution special , is regression model with using of the RMSE method, ensuring dependable and accurate predictions. We decided not to use classification in our method because it didn't fit well for dealing with continues class label values, which is a measure of the average deviation between the predicted values of the model and the actual values in the our dataset. This

makes sure our model is sharp and gives helpful insights for people deciding on their careers and helps companies set fair pay structures. With the 70-30 split and choosing the re regression grretion method, we're committed to making understanding data science salaries simple for everyone involved.

## 8 References

[1] https://www.kaggle.com/datasets/ruchi798/data-science-job-salaries [2] https://medium.com/nerd-for-tech/implementing-decision-trees-in-r-regression-problem-using-rpart-c74cbd9e0b7b