# Statistical Learning Final Project

## Data Science Job Salaries From 2020 to 2024

## Introduction:

### Describe the Dataset:

dataset provides information on data science job salaries covering from 2020 to 2024. It includes various features such as experience levels, employment types, job titles, and company characteristics. The primary focus is on understanding salary trends and identifying factors influencing compensation within the data science field.

### The description of each Feature:

work_year: The year of the data related to the job salary.

experience_level: The level of experience of the employee (e.g., entry-level, mid-level, senior-level).

employment_type: The type of employment (e.g., full-time, part-time, contract).

job_title: The title or role of the employee within the data science field. salary: The salary of the employee.

salary_currency: The currency in which the salary is denoted.

salary_in_usd: The salary converted to US dollars for standardization.

employee_residence: The residence location of the employee.

remote_ratio: The ratio of remote work allowed for the position.

company_location: The location of the company.

company_size: The size of the company based on employee count or revenue.

### Prediction Problem:

This research aims me to explore the following key aspects: “What factors influence the salaries (USD) of data science professionals in a specific location, considering their experience level, work years, job title, and the currency in which salaries are paid?”

### Data Splitting:

The dataset is divided into a train set(70%) and a test set(30%) randomly using sample() function to train and evaluate machine learning models.

# Statistical Learning Strategies and methods:

## Pre-Processing:

Firstly, I have loaded and read the data with read.csv () function and displayed the first few rows of a dataframe using head () function.

```
# Reading data using read.csv
dataFrame <- read.csv("DataScience_salaries_2024.csv")
head(dataFrame)
```

```
##   work_year experience_level employment_type                   job_title
## 1      2021              MI              FT                Data Scientist
## 2      2021              MI              FT               BI Data Analyst
## 3      2020              MI              FT                Data Scientist
## 4      2021              MI              FT                  ML Engineer
## 5      2022              SE              FT Lead Machine Learning Engineer
## 6      2021              MI              FT                  ML Engineer
##     salary salary_currency salary_in_usd employee_residence remote_ratio
## 1 30400000             CLP         40038                 CL          100
## 2 11000000             HUF         36259                 HU           50
## 3 11000000             HUF         35735                 HU           50
## 4  8500000             JPY         77364                 JP           50
## 5  7500000             INR         95386                 IN           50
## 6  7000000             JPY         63711                 JP           50
##   company_location company_size
## 1               CL            L
## 2               US            L
## 3               HU            L
## 4               JP            S
## 5               IN            L
## 6               JP            S
```

## Dataset Structure:

The str() function is used to display the structure of the dataset is crucial to understanding the data types of variables.

```
#structure of a dataframe
str(dataFrame)
```

```
## 'data.frame':    14838 obs. of  11 variables:
##  $ work_year        : int  2021 2021 2020 2021 2022 2021 2021 2022 2022 2022 ...
##  $ experience_level : chr  "MI" "MI" "MI" "MI" ...
##  $ employment_type  : chr  "FT" "FT" "FT" "FT" ...
##  $ job_title        : chr  "Data Scientist" "BI Data Analyst" "Data Scientist" "ML Engineer" ...
##  $ salary           : int  30400000 11000000 11000000 8500000 7500000 7000000 7000000
```

```
6600000 6000000 5500000 ...
##  $ salary_currency   : chr  "CLP" "HUF" "HUF" "JPY" ...
##  $ salary_in_usd     : int  40038 36259 35735 77364 95386 63711 94665 17684 76309 41809
...
##  $ employee_residence: chr  "CL" "HU" "HU" "JP" ...
##  $ remote_ratio      : int  100 50 50 50 50 50 50 100 50 50 ...
##  $ company_location  : chr  "CL" "US" "HU" "JP" ...
##  $ company_size      : chr  "L" "L" "L" "S" ...
```

Identifying the column names is essential for understanding the variables present in the dataset, so I used colnames() function.

```
# Displaying column names of a dataFrame
colnames(dataFrame)

##  [1] "work_year"          "experience_level"   "employment_type"
##  [4] "job_title"          "salary"             "salary_currency"
##  [7] "salary_in_usd"      "employee_residence" "remote_ratio"
## [10] "company_location"   "company_size"
```

## Handling Missing Values:

The is.na() is used to check for missing values here I used colSums() function to get total count of missing values in each column.

```
#Checking for missing values
missing_values <- colSums(is.na(dataFrame))
print(missing_values)

##         work_year   experience_level    employment_type         job_title
##                 0                 0                  0                 0
##            salary    salary_currency      salary_in_usd employee_residence
##                 0                 0                  0                 0
##      remote_ratio   company_location       company_size
##                 0                 0                  0
```

Where I found 0 missing values in each column.

In preprocessing character datatype variables need to be converted into integer datatype for modeling purposes for that I have used Label Encoding technique.

## Label Encoding:

In this section of the code I have created a vector char_columns which contains the names of all the variables which are character datatype. next by using for loop started performing label encoding in this first convert the columns to factor using as.factor() function, Then used factor() function to perform label encoding by converting the factor levels to integer datatype.

```
# Defining the character datatype columns
char_columns <- c("experience_level", "employment_type", "job_title", "salary_currency",
"employee_residence", "company_location", "company_size")
```

```r
# Performing label encoding for each character datatype column
for (col in char_columns) {
  # Converting the column to a factor
  dataFrame[[col]] <- as.factor(dataFrame[[col]])
  #factorizing function for label encoding
  dataFrame[[col]] <- as.integer(factor(dataFrame[[col]], levels = unique(dataFrame[[col]])))
}

head(dataFrame)
```

```
##   work_year experience_level employment_type job_title   salary salary_currency
## 1      2021              1               1         1 30400000               1
## 2      2021              1               1         2 11000000               2
## 3      2020              1               1         1 11000000               2
## 4      2021              1               1         3  8500000               3
## 5      2022              2               1         4  7500000               4
## 6      2021              1               1         3  7000000               3
##   salary_in_usd employee_residence remote_ratio company_location company_size
## 1         40038                 1          100                 1            1
## 2         36259                 2           50                 2            1
## 3         35735                 2           50                 3            1
## 4         77364                 3           50                 4            2
## 5         95386                 4           50                 5            1
## 6         63711                 3           50                 4            2
```

Then by using head() function displayed the first few rows of encoded dataFrame. Again used the str() function to see the structure of the dataFrame where we can see datatypes of each columns after encoding

```r
str(dataFrame)
```

```
## 'data.frame':    14838 obs. of  11 variables:
##  $ work_year        : int  2021 2021 2020 2021 2022 2021 2021 2022 2022 2022 ...
##  $ experience_level : int  1 1 1 1 2 1 2 3 4 3 ...
##  $ employment_type  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ job_title        : int  1 2 1 3 4 3 5 1 6 7 ...
##  $ salary           : int  30400000 11000000 11000000 8500000 7500000 7000000 7000000
6600000 6000000 5500000 ...
##  $ salary_currency  : int  1 2 2 3 4 3 4 2 4 3 ...
##  $ salary_in_usd    : int  40038 36259 35735 77364 95386 63711 94665 17684 76309 41809
...
##  $ employee_residence: int  1 2 2 3 4 3 4 2 4 3 ...
##  $ remote_ratio     : int  100 50 50 50 50 50 50 100 50 50 ...
##  $ company_location : int  1 2 3 4 5 4 5 3 5 4 ...
##  $ company_size     : int  1 1 1 2 1 2 1 3 1 1 ...
```

Now by observing the structure you can see encoded dataFrame.

## Data Splitting:

Here I have split the dataset into training and test sets to facilitate the model training and evaluation.

The train_proportion is specified as 0.7 (70%) of the dataset will be used for training the model, leaving the remaining 0.3 (30%) for testing.

Set the seed for reproducibility, Using a fixed seed ensures that the random sampling process will yield the same result each time the code is executed.

The sample () function randomly selects a subset of indices, with the size of the subset determined by multiplying the total number of rows by the training proportion.

The floor () function ensures that the result is rounded down to the nearest integer.

Then used the sampled indices to extract the corresponding rows from the original data frame, creating separate training and test data sets. That is 'train_data' and 'test_data'.

```
# Defining the proportion of data for training
train_proportion <- 0.7  # 70% of the data for training

# Setting seed for reproducibility
set.seed(123)

# Creating indices for train and test sets
train_index <- sample(seq_len(nrow(dataFrame)), size = floor(train_proportion * nrow(dataFrame)))

# Splitting data into training and test sets
train_data <- dataFrame[train_index, ]
test_data <- dataFrame[-train_index, ]
cat("Then Dimension of train data ",dim(train_data), "\n")

## Then Dimension of train data  10386 11

cat("Then Dimension of test data ",dim(test_data), "\n")

## Then Dimension of test data  4452 11
```

Then I displayed the dimensions of both train data and test data

## Exploratory Data Analysis on Train Data:

### Summary Statistics:

The summary statistics of the train_data provide a comprehensive overview of the dataset.

```
#summary statistics of train data
summary(train_data)
```

```
##    work_year    experience_level employment_type   job_title
##  Min.   :2020   Min.   :1.000    Min.   :1.00    Min.   :  1.00
##  1st Qu.:2023   1st Qu.:2.000    1st Qu.:1.00    1st Qu.: 10.00
##  Median :2023   Median :2.000    Median :1.00    Median : 12.00
##  Mean   :2023   Mean   :1.894    Mean   :1.01    Mean   : 22.04
##  3rd Qu.:2024   3rd Qu.:2.000    3rd Qu.:1.00    3rd Qu.: 31.00
##  Max.   :2024   Max.   :4.000    Max.   :4.00    Max.   :152.00
##      salary         salary_currency  salary_in_usd    employee_residence
##  Min.   :  14000   Min.   : 1.000   Min.   : 15000   Min.   : 1.00
##  1st Qu.:  103200   1st Qu.: 9.000   1st Qu.:102690   1st Qu.:11.00
##  Median :  142200   Median : 9.000   Median :141300   Median :11.00
##  Mean   :  165660   Mean   : 9.448   Mean   :150023   Mean   :12.16
##  3rd Qu.:  188075   3rd Qu.: 9.000   3rd Qu.:186000   3rd Qu.:11.00
##  Max.   :30400000   Max.   :23.000   Max.   :800000   Max.   :88.00
##   remote_ratio    company_location  company_size
##  Min.   :  0.00   Min.   : 1.000   Min.   :1.000
##  1st Qu.:  0.00   1st Qu.: 2.000   1st Qu.:3.000
##  Median :  0.00   Median : 2.000   Median :3.000
##  Mean   : 32.71   Mean   : 4.509   Mean   :2.852
##  3rd Qu.:100.00   3rd Qu.: 2.000   3rd Qu.:3.000
##  Max.   :100.00   Max.   :75.000   Max.   :3.000
```

From the summary() function we can understand following:

**Work Year:** Most of the data comes from 2023 and 2024.

**Experience Level:** Many people are at mid-level experience.

**Employment Type:** Most people work full-time (with mode of 1), but there are some other types of employment too indicated as mean slightly above 1, suggesting a slight skew towards other types of employment.

**Job Title:** There's a wide range of job titles with median around 12 and a mean approximately 22, indicating diverse roles present in the dataset.

**Salary and Currency:** Salaries vary widely, from $14,000 to $800,000, with most converted to US dollars.

**Employee Residence:** The residence locations of the employees show a range from 1 to 88, with a mode and median of 11.Which suggest that employees live in one specific location.

**Remote Work Ratio:** The amount of remote work allowed varies, with an average of about 32.71%.

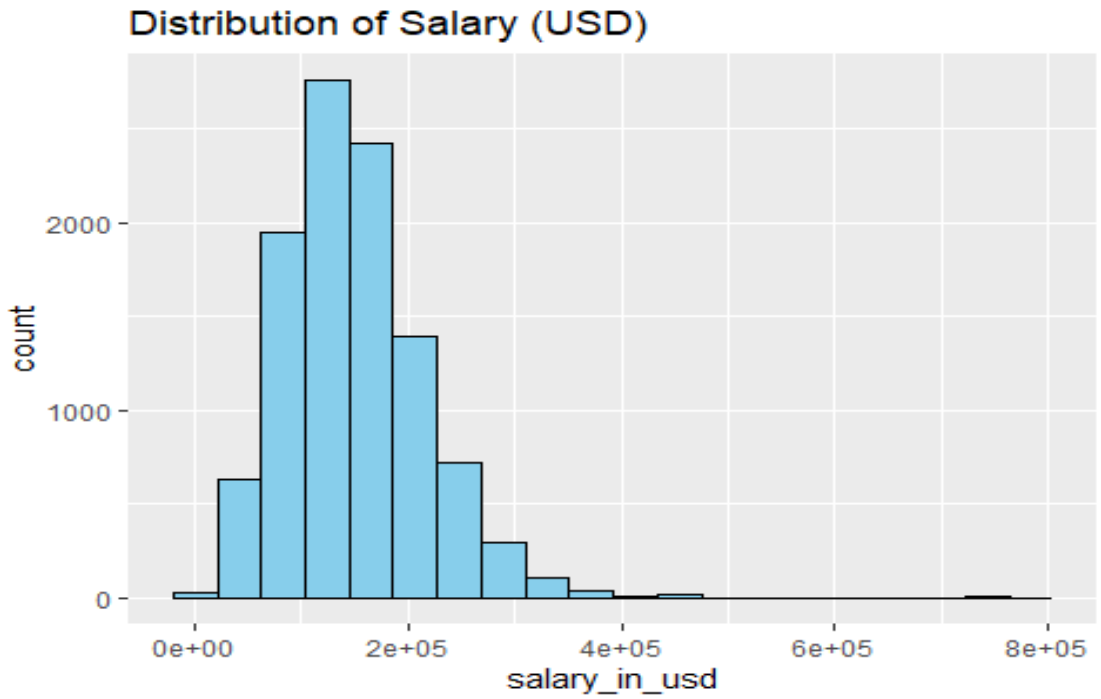
**Company Location:** The locations of the companies range from 1 to 75, with a median of 2 and a mean of approximately 4.51 Companies are spread across various locations.

**Company Size:** The sizes of the companies are represented by levels ranging from 1 to 3, with a median and mean around 3. indicates that the companies are of a similar size, falling within a certain category.

**Distribution of Numerical Variables:**

To see the distribution of numeric variables I generated a histogram using the function ggplot() from library ggplot2. The aes() function maps the variable salary_in_usd to the x-axis. The geom_histogram() adds a histogram layer to the plot like number of bins, color of bins. And ggttle() function adds

```
#histogram plot of salary_in_usd
library(ggplot2)

ggplot(train_data, aes(x = salary_in_usd)) + geom_histogram(bins = 20, fill = "skyblue", color = "black") + ggtitle("Distribution of Salary (USD)")
```

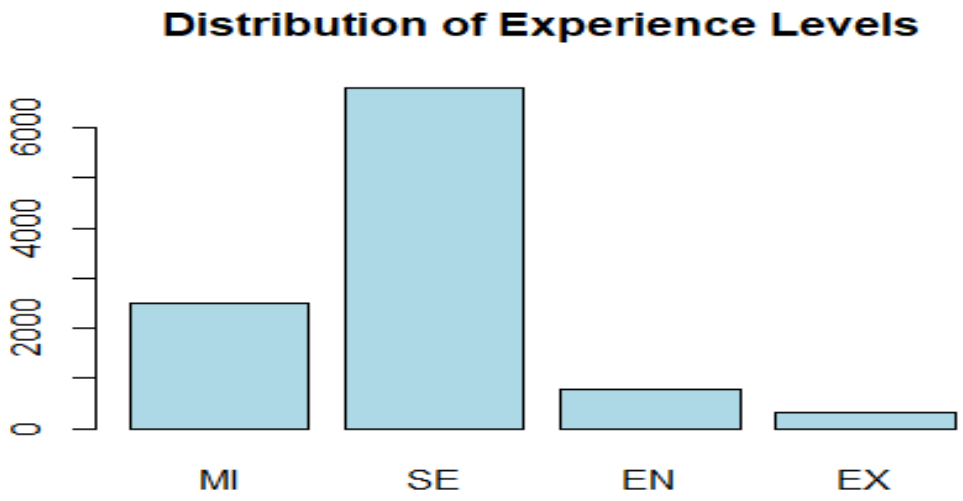



The x-axis shows salary in USD and the y-axis shows the number of people who make that salary. From the plot here are the following observations: The most frequent salary is around $20,000 USD per year. There are a significant number of people who make less than $20,000 per year. The salary range extends to over $80,000 per year, however there are far fewer people who make salaries in that range.

**Distribution of Experience levels (categorical Variable)**

The barplot() function computes the frequency of each unique value in the 'experience_level' column of the train_data using table() function. It will counts the occurrences of each experience level.

```
barplot(table(train_data$experience_level), col = "lightblue", names.arg = c("MI", "SE", "EN", "EX"),main = "Distribution of Experience Levels")
```

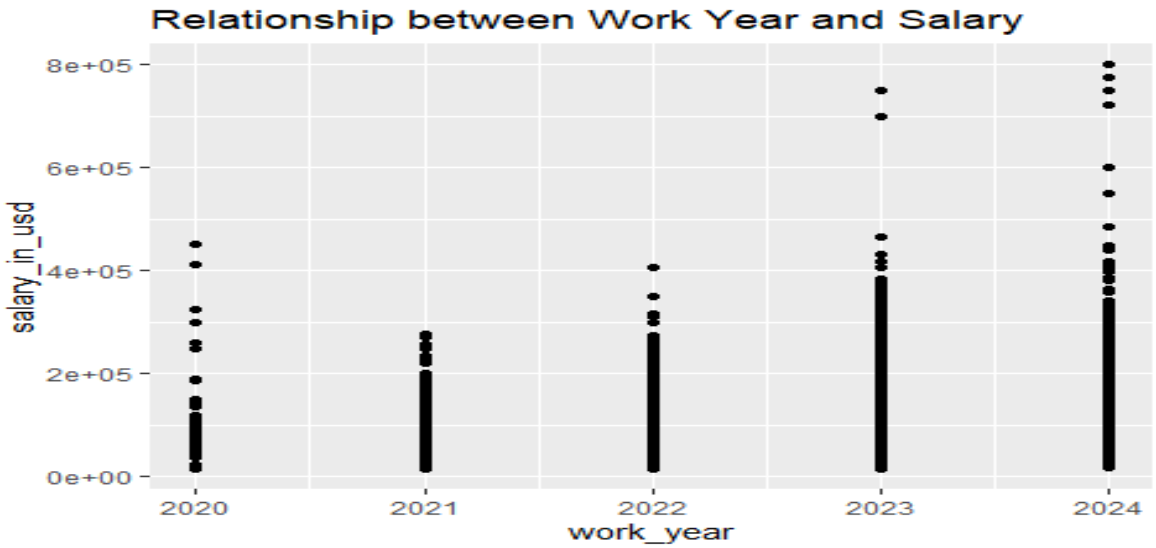



From the plot we can observe that the SE (Senior-Level) has more distribution compared other categories then followed by MI(Mid-Level), EN(Entry-Level) and EX(Executive-Level).

**Scatter plot for Relationship between Variables:**

This plot is used to find the relationship between the “work_year” and “salary_in_usd” using ggplot() function and used the train_data and the geom_point() function adds a layer of points to the plot, by creating a scatter plot.

```
ggplot(train_data, aes(x = work_year, y = salary_in_usd)) + geom_point() +
ggtitle("Relationship between Work Year and Salary")
```

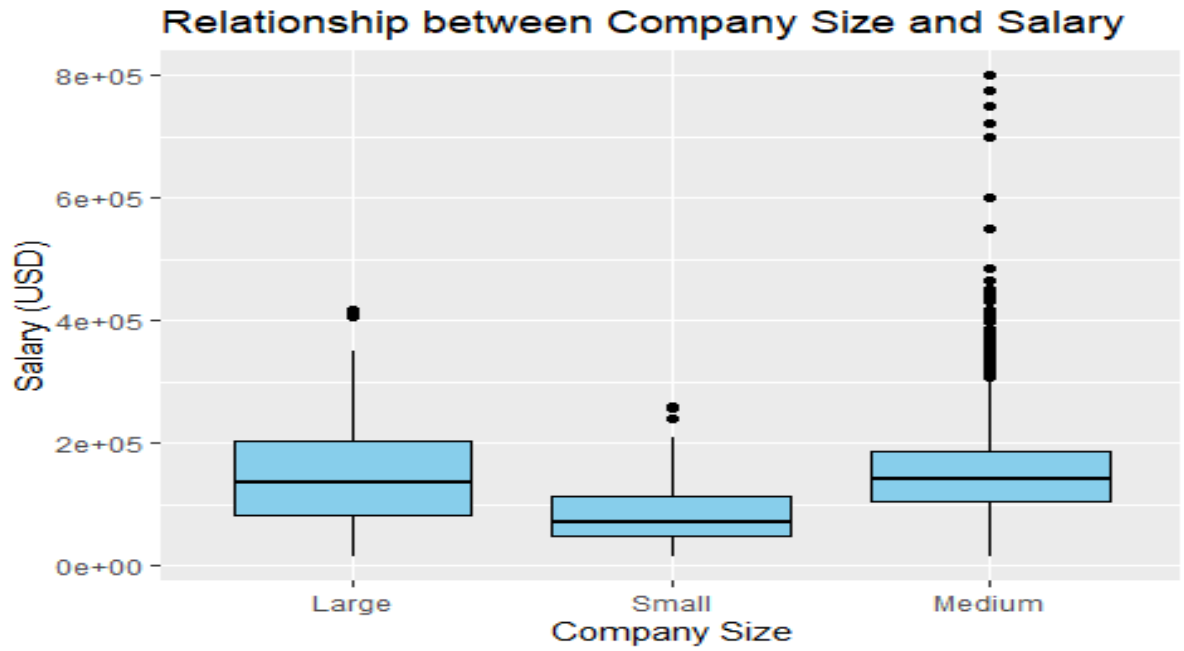


From the plot we can observe that it shows a positive linear relationship between the work_year and salary, which means that the number of work years increases, the salary also increases. By the information, the year 2024 has the highest salary USD.

**Boxplot for Relationship between variables:**

The ggplot() is used to find the relationship between 'company_size', 'salary_in_usd'. and geom_boxplot() adds a boxplot to the plot, visualizing the distribution of salaries within each category size with arguments fill and color and labs() function adds title and X and Y labels.

```
ggplot(train_data, aes(x = factor(company_size), y = salary_in_usd)) +
  geom_boxplot(fill = "skyblue", color = "black") +
  scale_x_discrete(labels = c("Large", "Small", "Medium")) +  # Rename x-axis labels
  labs(title = "Relationship between Company Size and Salary",
       x = "Company Size",
       y = "Salary (USD)")
```




From the boxplot we can observe the following: Based on the position of the boxes in the plot, it appears that employees working at larger companies tend to have higher median salaries than those working at smaller companies. The middle 50% of the data represents the median of salary_in_usd.

The spread of the boxes also suggests that there may be more variability in salary among employees working at larger companies. The box for "Large" companies is wider than the boxes for "Medium" and "Small" companies.

## Feature Engineering:

## Selection Method:

## Forward Stepwise selection:

The forward stepwise selection is used to select the subset of predictors variables for predicting the target varaiable 'salary_in_usd'.

The regsubsets() function from leaps library is used to perform Forword stepwise selection. Starting with the intercept, iteratively adds predictor variables to the model based on their significance.

The formula salary_in_usd ~ salary + experience_level + work_year + job_title + company_location + salary_currency specifies the target variable and the potential predictor variables to consider for the model.

The argument nvmax = 6 specifies the maximum number of variables to include in the model.

The argument method = "forward" specifies forward stepwise selection.

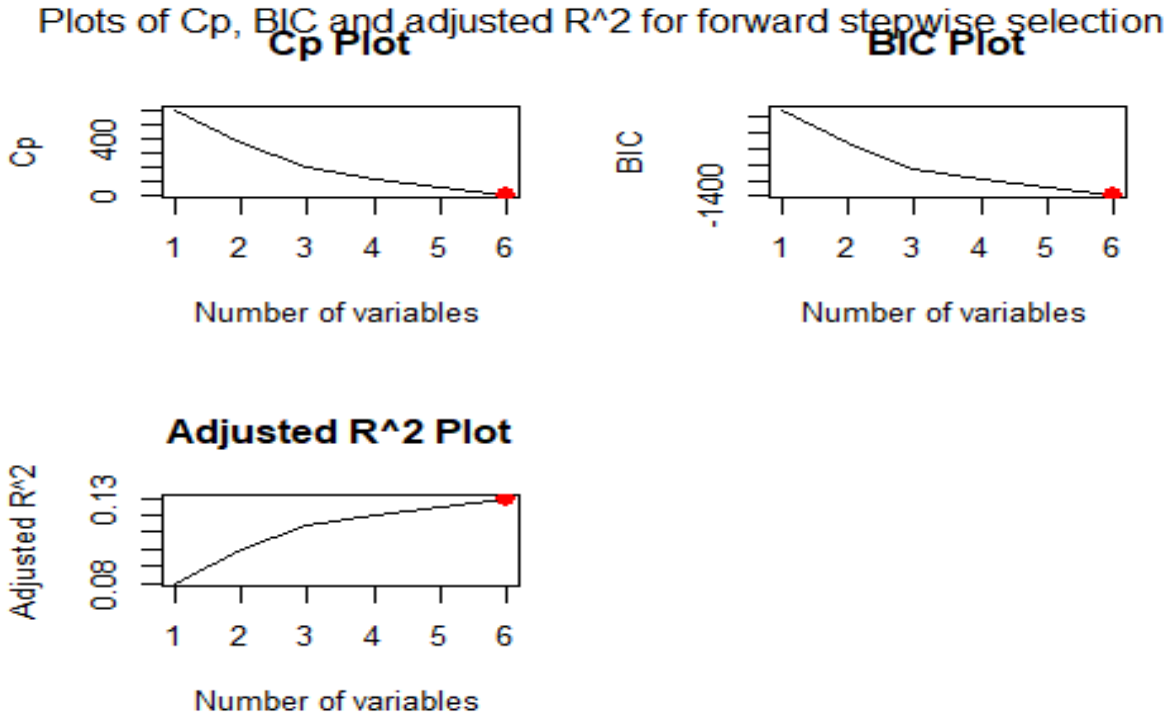
After obtaining summary. Next plots of Cp, BIC (Bayesian Information Criterion), and adjusted R^2 are created to visualize the model selection process. These plots help identify the optimal number of predictors for the model.

Then calculate the coefficients of the best models based on Cp, BIC, and adjusted R^2 are extracted using the coef() function.

The coefficients represent the estimated effect of each predictor variable on the target variable (salary_in_usd) in the selected model.

```
library(leaps)

# Performing forward stepwise selection
forwardstep_select <- regsubsets(salary_in_usd ~ salary + experience_level + work_year +
job_title +  company_location + salary_currency , data = train_data, nvmax = 6, method =
"forward")

# Summary of the forward stepwise selection
summ_forward <- summary(forwardstep_select)

# Plotting  Cp, BIC, and adjusted R^2
par(mfrow = c(2, 2))
plot(summ_forward$cp, xlab = "Number of variables", ylab = "Cp", type = "l", main = "Cp
Plot")
points(which.min(summ_forward$cp), summ_forward$cp[which.min(summ_forward$cp)], col
= "red", cex = 2, pch = 20)
plot(summ_forward$bic, xlab = "Number of variables", ylab = "BIC", type = "l", main = "BIC
Plot")
points(which.min(summ_forward$bic), summ_forward$bic[which.min(summ_forward$bic)],
col = "red", cex = 2, pch = 20)
plot(summ_forward$adjr2, xlab = "Number of variables", ylab = "Adjusted R^2", type = "l",
```

```
main = "Adjusted R^2 Plot")
points(which.max(summ_forward$adjr2),
summ_forward$adjr2[which.max(summ_forward$adjr2)], col = "red", cex = 2, pch = 20)
mtext("Plots of Cp, BIC and adjusted R^2 for forward stepwise selection", side = 3, line = -2,
outer = TRUE)

# Extracting coefficients of the best models
best_model_cp_fwd <- coef(forwardstep_select, which.min(summ_forward$cp))
best_model_bic_fwd <- coef(forwardstep_select, which.min(summ_forward$bic))
best_model_adjr2_fwd <- coef(forwardstep_select, which.max(summ_forward$adjr2))
```




```
# Printing coefficients
cat("Coefficients of the best model based on Cp (Forward Stepwise Selection):\n")
```

```
## Coefficients of the best model based on Cp (Forward Stepwise Selection):
```

```
print(best_model_cp_fwd)
```

```
##      (Intercept)           salary experience_level        work_year
##    -1.386282e+07     1.384881e-02     8.674765e+03     6.949155e+03
##        job_title company_location  salary_currency
##    -4.054259e+02    -1.459413e+03    -5.237855e+03
```

```
cat("\nCoefficients of the best model based on BIC (Forward Stepwise Selection):\n")
```

```
##
## Coefficients of the best model based on BIC (Forward Stepwise Selection):
```

```
print(best_model_bic_fwd)
```

```
##     (Intercept)          salary experience_level       work_year
##   -1.386282e+07    1.384881e-02    8.674765e+03    6.949155e+03
##       job_title company_location  salary_currency
##   -4.054259e+02   -1.459413e+03   -5.237855e+03
```

```
cat("\nCoefficients of the best model based on adjusted R^2 (Forward Stepwise Selection):\n")
```

```
## Coefficients of the best model based on adjusted R^2 (Forward Stepwise Selection):
```

```
print(best_model_adjr2_fwd)
```

```
##     (Intercept)          salary experience_level       work_year
##   -1.386282e+07    1.384881e-02    8.674765e+03    6.949155e+03
##       job_title company_location  salary_currency
##   -4.054259e+02   -1.459413e+03   -5.237855e+03
```

From the observations the coefficients indicate the estimated effect of each predictor variable on the target variable (salary_in_usd) in the selected model.

It's interesting to note that the best model's coefficients (Cp, BIC, and modified R^2) are the same for all three selection criteria. This implies that the same set of predictor factors was continually found to be the most important for salary prediction by the forward stepwise selection procedure.

We can understand that a positive coefficient for salary suggests that an increase in salary leads to a higher predicted salary_in_usd, while negative coefficients for experience_level, work_year, job_title, company_location, and salary_currency indicate that higher values of these variables are associated with lower predicted salary_in_usd.

**Discussion on their applicability to the prediction problem:**

From the feature selection method, we can observe that there are some positive and negative relationships between the response and predictors which suggest that my data have both the linear and non-linear relationships between predictors and the response variable, which is import for my prediction problem. EDA is valuable for any prediction problem as it helps in understanding the data and informing subsequent modeling decisions.

As found some linearity and non-linearity in the feature selection step the Random Forest performs well when there are both linear and non-linear relationships between predictors and the target variable. It is robust to overfitting and generally provides accurate predictions.

## Predictive analysis and results:

**Predictive analysis:**

**Random Forest with Cross-Validation:**

The caret library is loaded which provides functions for training machine learning models and perform cross-validation.

The trainControl function is used to define the parameters for cross-validation. Here, method = “cv” specifies cross-validation, and number = 5 indicates that 5-fold cross-validation will be performed.

The train() function is used to train the Random Forest model.

salary_in_usd ~ salary + experience_level + work_year + job_title + company_location + salary_currency specifies the formula for the model, with salary_in_usd as the target variable and other variables as predictors.

method = “rf” indicates that a Random Forest model will be used.

trControl = training_control specifies the training control parameters.

The trained Random Forest model (rf_cv_model) is used to make predictions on the test data (test_data) using the predict function.

The Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared ($R^2$) are calculated to evaluate the performance of the Random Forest model on the test data. RMSE (rmse_rf_cv) measures the average deviation of the predicted values from the actual values. MAE (mae_rf_cv) measures the average absolute difference between the predicted and actual values. R-squared (r_squared_rf_cv) represents the proportion of the variance in the target variable that is predictable from the predictor variables.

```
# Loading caret library
library(caret)

## Loading required package: lattice

# Defining training control
training_control <- trainControl(method = "cv",     # Cross-validation method
                                 number = 5)       # Number of folds

# Training Random Forest model using cross-validation
rf_cv_model <- train(salary_in_usd ~ salary + experience_level + work_year + job_title +
company_location + salary_currency,
                     data = train_data,
                     method = "rf",            # Random Forest method
                     trControl = training_control)

# Making  predictions on test data
predictions_rf_cv <- predict(rf_cv_model, newdata = test_data)

# Evaluating model performance
rmse_rf_cv <- sqrt(mean((predictions_rf_cv - test_data$salary_in_usd)^2))
cat("Random Forest Test RMSE using Cross-Validation:", rmse_rf_cv, "\n")
```

```
## Random Forest Test RMSE using Cross-Validation: 11677.83

# Calculating Mean Absolute Error (MAE)
mae_rf_cv <- mean(abs(predictions_rf_cv - test_data$salary_in_usd))
cat("Random Forest Test MAE using Cross-Validation:", mae_rf_cv, "\n")

## Random Forest Test MAE using Cross-Validation: 845.4809

# Calculating R-squared
r_squared_rf_cv <- cor(predictions_rf_cv, test_data$salary_in_usd)^2
cat("Random Forest Test R-squared using Cross-Validation:", r_squared_rf_cv, "\n")

## Random Forest Test R-squared using Cross-Validation: 0.9725187
```

## Result:

**Random Forest Test RMSE (Root Mean Squared Error) using Cross-Validation:**

The average difference between predicted and actual earnings in the test dataset is measured by the RMSE. A lower RMSE suggests that the model is more accurate in predicting salaries. The RMSE value of 11638.79 in this case indicates a $11638.79 difference between the projected and actual salaries.

**Random Forest Test MAE (Mean Absolute Error) using Cross-Validation:**

The average absolute difference between the expected and actual salaries is measured by the MAE. A lower MAE indicates a higher level of model accuracy. In this case, the MAE value of 854.7967 indicates that there is, on average, a $854.80 difference between the expected and actual salaries.

**Random Forest Test R-squared using Cross-Validation:**

The proportion of the variance in the target variable (salaries) that can be explained by the predictor variables (currency, work years, experience level, and job title) is indicated by the R-squared ($R^2$) number. A greater correlation between the model and the data is indicated by a higher R-squared value. The model's predictor variables account for roughly 97.27% of the variation in salary, according to the R-squared value of 0.9726992.

**Overall Result on prediction problem:**

The selected predictor factors (experience level, work years, job title, and currency) collectively influence predicting salaries of data science experts in the specified region, as indicated by the high R-squared value (0.9726992). This implies that these variables significantly affect the salary variation seen in the data set. Furthermore, the low MAE (854.7967) and RMSE (11638.79) results show that the Random Forest model does a good job of correctly predicting incomes based on these variables. This suggests that, considering the designated predictor variables, the model is trustworthy for estimating salaries within the provided context.

Overall, the Random Forest model demonstrates promising performance in identifying the factors influencing salaries of data science professionals, providing valuable insights into the relationship between these variables and salary outcomes.

## Conclusion:

### Scope and Generalizability of the Predictive Analysis:

The predictive analysis aimed to understand the factors influencing the salaries of data science professionals in a specific location, considering various predictors such as experience level, work years, job title, and currency. The analysis utilized machine learning techniques, like Random Forest modeling and cross-validation, to develop a predictive model.

The scope of the analysis is focused on the dataset provided, which covers data science job salaries from 2020 to 2024. The model's applicability in general depends on the representativeness and diversity of the dataset, as well as the relevance of the chosen predictors. If the dataset is comprehensive and accurately reflects the characteristics of the target population, the model's predictions can be considered applicable to similar scenarios and locations within the same context.

### Potential Limitations and Possibilities for Improvement:

### Potential Limitations:

The dataset used may not fully represent all industries, or scenarios, limiting the model's applicability to diverse contexts.

Economic conditions and salary trends may change over time, and the model may not account for these variations, impacting its predictive power for future scenarios.

### Possibilities for Improvement:

Obtain a more diverse dataset covering various industries, locations, and scenarios to improve the model's representativeness and applicability.

Explore more comprehensive feature engineering techniques and consider domain expertise input to identify and include all relevant factors influencing salaries.

Incorporate temporal features or time-series analysis to capture changes in economic conditions and salary trends over time, improving the model's predictive accuracy for future scenarios