# Developer Salary Estimation

*Abdul Basit Hafeez*

*March 01, 2019*

## Problem

Stackoverflow is a platform where user asks or answer the programming related questions. It is the biggest platform having developers from different are present. From last couple of years Stackoverflow is running a survey for collecting information about the careers,salary and about theirself. This data is availiable for analysis to every one.

For applying to a new position in any stage of career and estimate how much he/she can earn, developer require to consider lot of things like years of expereince,formal education,coding skills etc. This project aims to model the salary prediction given these kind of features from the data

In this project I have targeted the Schengen countries where i want to go after this masters.

## Data Understanding

Data selected for this problem is freely available

https://insights.stackoverflow.com/survey

Number of Rows and colomuns in Data are below

```r
df = read.csv("stackdata.csv")
dim(df)
```

```
## [1] 98855   129
```

There are 98855 rows and 129 colomuns.

Since there are 100+ featues, I will will just show first 20 features.

```r
colnames(df)[1:20]
```

```
##  [1] "Respondent"        "Hobby"              "OpenSource"
##  [4] "Country"           "Student"            "Employment"
##  [7] "FormalEducation"   "UndergradMajor"     "CompanySize"
## [10] "DevType"           "YearsCoding"        "YearsCodingProf"
## [13] "JobSatisfaction"   "CareerSatisfaction" "HopeFiveYears"
## [16] "JobSearchStatus"   "LastNewJob"         "AssessJob1"
## [19] "AssessJob2"        "AssessJob3"
```

From these columns, there can be many potential features that can help us to predict the salary.

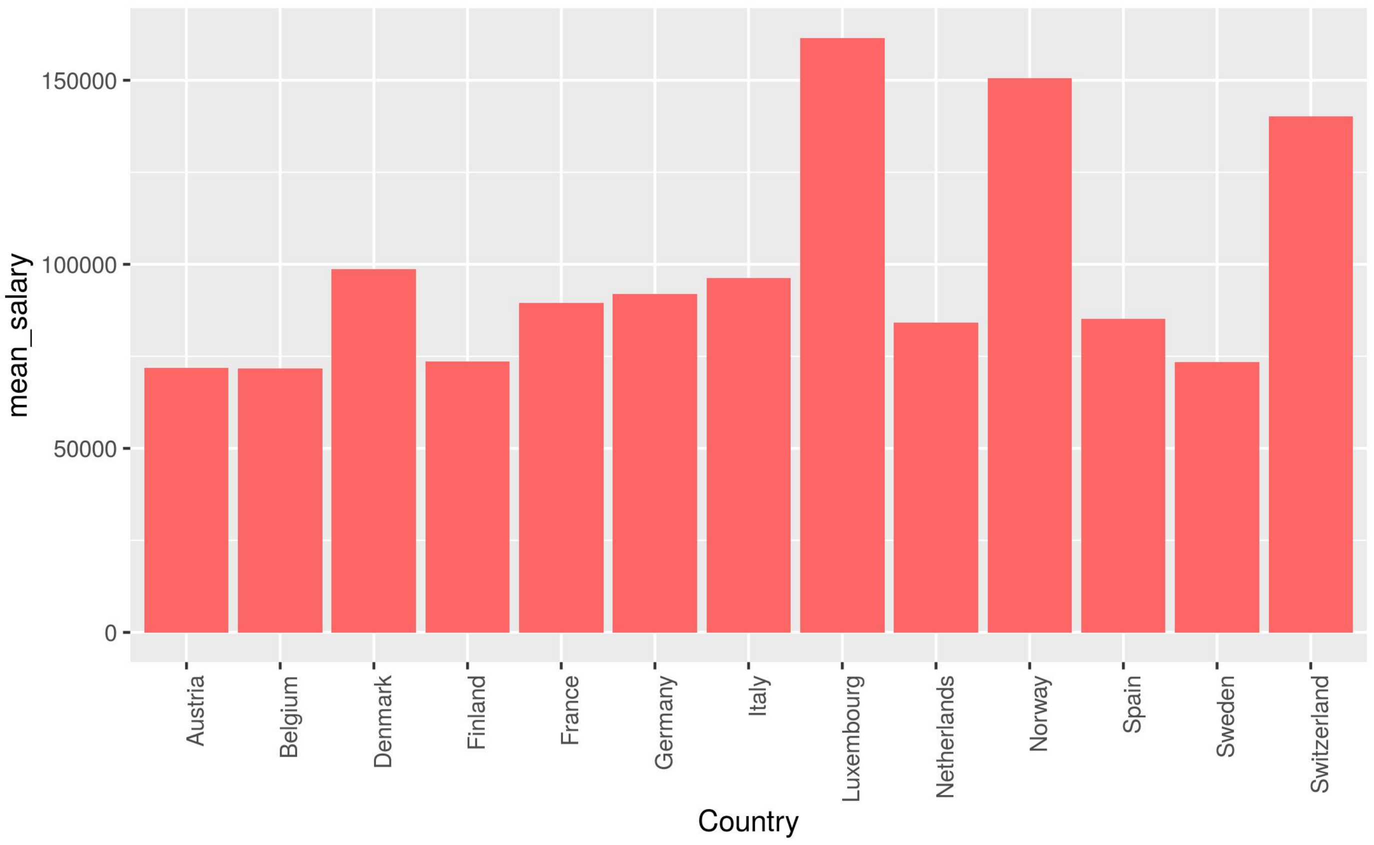
To understand data better, lets explore some of the features.

Data Contains a column for Formal education, table below summarizes the education level with respect to mean salary earned.

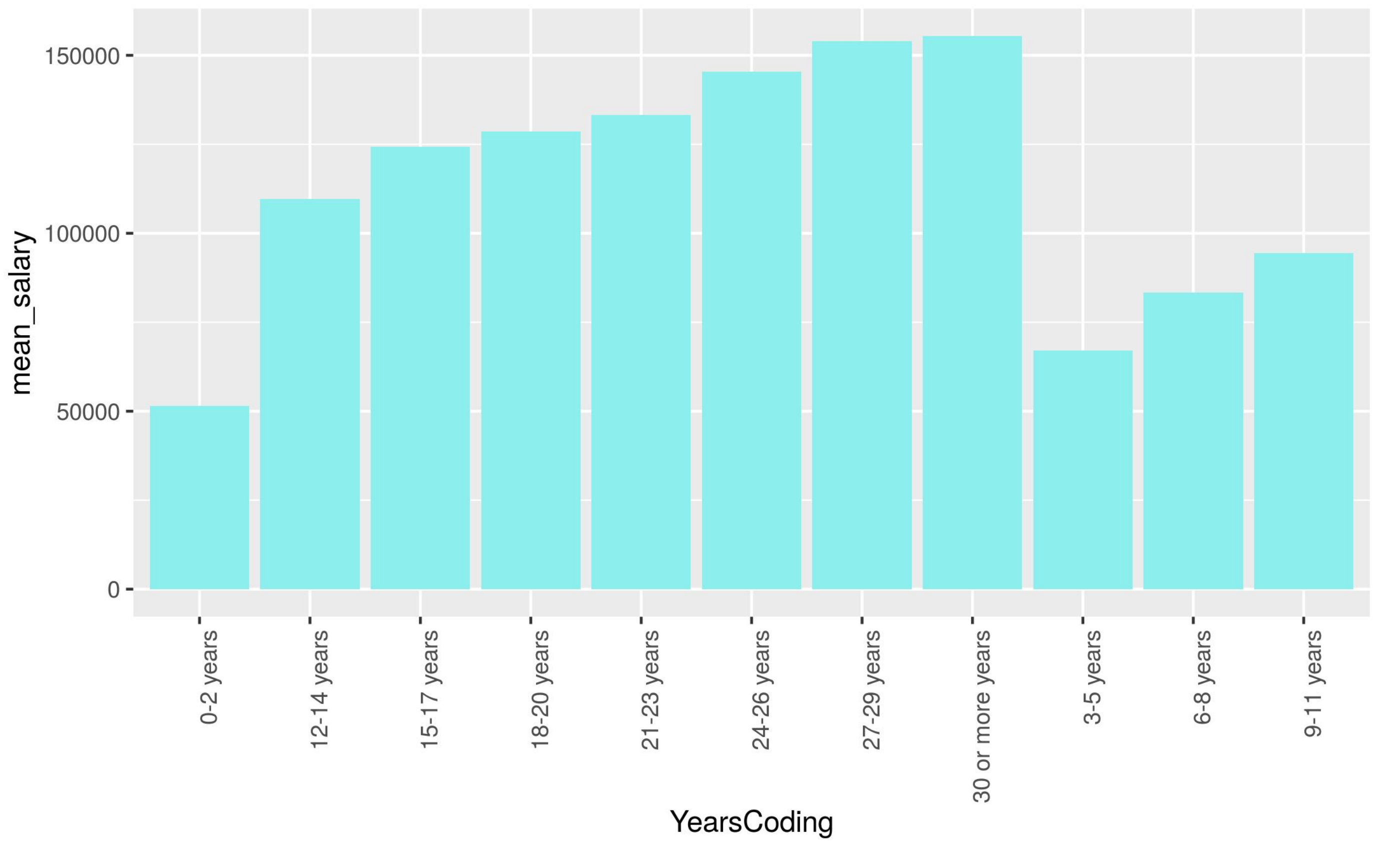
| FormalEducation | mean_salary |
|---|---:|
| Associate degree | 99178.01 |
| Bachelor’s degree (BA, BS, B.Eng., etc.) | 96462.51 |
| I never completed any formal education | 151409.60 |
| Master’s degree (MA, MS, M.Eng., MBA, etc.) | 100588.15 |
| Other doctoral degree (Ph.D, Ed.D., etc.) | 127154.87 |
| Primary/elementary school | 71942.71 |
| Professional degree (JD, MD, etc.) | 81716.98 |
| Secondary school (e.g. American high school, German Realschule or Gymnasium, etc.) | 69066.00 |
| Some college/university study without earning a degree | 92913.79 |

It makes sense that Doctoral degree earns more than master degree holder at average. For other,there could be some outliers in the Salary that we will remove in Data preparation section.

Lets see the mean average of Salary in nearest Schengen regions.



Now, lets explore the effect of “Number of years of coding” on salary. Graph below show that coding experience definitely increase your chances to earn.



## Data Preparation

First of all, we want to remove outliers w.r.t Salary, we will take into account the average minimum wage of Schegen region(1490/month) and will remove rocords belwo this thresold.

We will also remove outliers above 45% of Average Salary

```
df = df[df$ConvertedSalary>(1498*12),]
df = df[df$ConvertedSalary <250000,]
```

Let's see the missing value percentage of each coloumn. We will display top 6 missing value colomun by percentage

```
clmeans= sort(colMeans(is.na(df)))
tail(clmeans,4)
```

```
##  ErgonomicDevices  HackathonReasons        MilitaryUS TimeAfterBootcamp
##         0.5494384         0.6190010         0.7371299         0.9058671
```

So lets drop the coloumns with most missing values

```
df_sf = df[,!(names(df) %in% top_missing_cols)]
```

Most of the coloumns in the dataset have two many unique vallues, it will be dificult to fit a model with them.

Lets calculate columns with highest unique values.

```
unique_count = apply(df_sf, 2, function(x) length(unique(x)))
unique_count= sort(unique_count)
tail(unique_count,4)
#EducationTypes=222 , SelfTaughtTypes=265 , DevType=200
```

Remove coloumns with more than 200 unique values, except of Salary

```
csalary_temp = df_sf$ConvertedSalary
df_sf = df_sf[,unique_count<200]
df_sf$ConvertedSalary = csalary_temp
```

Now, if we see Salaries, the variable **Currency** has different currencies for example "Euro","US" etc. Also saaly varies across domographics. We can convert saalries according to conversion rates. But for sake of simplicity. we will stick to schegen region with following countries(filtered out)

```
df_sf = df_sf[df_sf$Country %in% c("France","Germany","Austria","Sweden","Switzerland","Belgium","Luxembourg"
                    "Netherlands","Spain","Italy","Finland","Norway","Denmark"),]
```

After the previous operations, we dont need following colomns any more

```
drops <- c("Salary","SalaryType" , "Country" , "Currency")
selected_features = colnames( df_sf[,!(names(df_sf) %in% drops)] )
df_sf = df_sf[,!(names(df_sf) %in% drops)]
```

Some of the coloumn(survey questions) are highly related to stackoverflow marketing or specific to benefits provided in company. We dont need them.

```
drop_final = c('AssessBenefits3', 'AssessBenefits4', 'AssessBenefits6',
        'AssessBenefits9', 'AssessBenefits10','AssessJob3', 'AssessJob5', 'AssessJob7')
df_sf = df_sf[,!(names(df_sf) %in% drop_final)]
```

Lets, remove missing values and see the total number of rows and coloums left now

```
df_sf = na.omit(df_sf)
dim(df_sf)
```

Lastly,we will need to encode categogrical value with one hot encoded, we will ot convert them into integars to avoid the wieghing biasedness.

```
dummyv<- dummyVars(" ~ .", data = df_sf)
hencoded <- data.frame(predict(dummyv, newdata = df_sf))
```

## Modelling

For modelling this problem, we will treat this as classification problem first. We will make 4 classes out of the Salary feature with help of binning and then apply 3 different types of classifiers.

### Binning

we will use make 4 classes of the salary

```
df_sf$group <- as.numeric(cut(df_sf$ConvertedSalary, 4))
df_sf$ConvertedSalary = NULL
```

Before modelling, we will split the data into two set of size 75% and 25% respectively. First one with the 75% of data will be called Training data on which model will be trained. and Remaining 25% data (Test Data) will be used for prediction evaulation.

Following code will split the data.

```
smp_size <- floor(0.75 * nrow(hencoded))
set.seed(123)
train_ind <- sample(seq_len(nrow(hencoded)), size = smp_size)
train <- hencoded[train_ind, ]
test <- hencoded[-train_ind, ]
```

After that, we wil separaate the label from the data

```
train_y<- factor(train$group)
train_X = train[,names(train) != "group"]
test_y<- factor(test$group)
test_X = test[,names(test) != "group"]
```

Lets train on different Classifiers

### SVM Classifier

SVM classifier can be used to separate classes with linear hyperlanes or non linear with help of kernel trick. SVM separates the examples of the different classes by margin that is as wide as possible.

For non linear boundaries, there can be different can be different type of kernel use, for exmaple polynomial kernel with varying degrees(2,3,4. . . n) or radial kernel.

Lets train a SVM model on training example with radial polynomial and degree 8.

```
svm_model_cat <- svm(train_cat_X,train_cat_y,kernel="polynomial"
                     ,degree=8,gamma=1,cost=1,probability = TRUE)
```

### RandomForest

RandomForect is algorithm that builds on top Of decision tree and extends it with bootstrap aggregating or tree learned.

Let us train a randomFoest model with value of ntree as 800

```
model1 <- randomForest(train_cat_X,train_cat_y,importance = TRUE)
```

### Support vector regressiom

For modeling problem as regression, we will take salary as class not bin. following will be the changes from the svm

```
train_y = train$ConvertedSalary
train_X = train[,names(train) != "ConvertedSalary"]
test_y = test$ConvertedSalary
test_X = test[,names(test) != "ConvertedSalary"]
svm_model_regression <- svm(train_X,train_y, kernel="polynomial",
                            degree=8,cost=1,gamma = 0.1,epsilon = 0.05,coef0 = 1)
```

## Evaluation

Lets check the accuracy or error of the models on the test data.

### Evaluation SVM Model

```
pred_cat <- predict(svm_model_cat, test_cat_X,decision.values = TRUE, probability = TRUE)
acc_table = table(pred_cat,test_cat_y)
cm_cat = confusionMatrix(acc_table)
cm_cat$overall['Accuracy']
```

```
## [1] "Accuracy 0.7615998"
```

we are getting 76.15 % accuracy on test set,lets try to change kernel to radial and following parameters parameter

kernel=radial ,cost= 10 , gamma=0.3 ,epsilon = 0.05

```
svm_model_cat2 <- svm(train_cat_X,train_cat_y,kernel="radial"
                    ,gamma = 0.3,cost=10,epsilon = 0.05,probability = TRUE)
```

Now see if Accuracy has increased or decreased using radial kernel and these parameters

```
pred_cat <- predict(svm_model_cat2, test_cat_X,decision.values = TRUE, probability = TRUE)
acc_table = table(pred_cat,test_cat_y)
cm_cat = confusionMatrix(acc_table)
cm_cat$overall['Accuracy']
```

```
## [1] "Accuracy 0.7669998"
```

There is very litle improvment, we have 76.69% accuracy with the setting. Instead of manually changing the paramters, we can use hyperparameter tunning method with cross validation to search for best hyperparameters

### Hyperparameter Tuning

Tune function can be used for searching various hyperparamers

Lets use this to find good values for "c" and "gamma"

```
svm_tuned = tune.svm(train_cat_x,train_cat_y,
                     kernel="polynomial",degree=8, gamma = c(0.1,0.3,0.5,1),cost=c(1,2,5,10))
svm_tuned$best.parameters
```

This gives us the same parameter values, but if we have different paramters we can use them to fit.

### Evaluation Random Forest

Lets evaluate random forest on test data

```
predTrain_rf <- predict(model1, test_X, type = "class")
acc_table_rf= table(predTrain_rf,test_y)
cm_cat_rf = confusionMatrix(acc_table_rf)
cm_cat_rf$overall['Accuracy']
```

```
## [1] "Accuracy 0.6156897"
```

We get 61% accuracy, which is quite bad. We can change some parameter to improve accuracy

Lets try with increasing ntree to 800

```
model2 <- randomForest(train_X,train_y, ntree = 800, mtry = 6, importance = TRUE)
```

Does this increase accuracy?

```
predTrain_rf <- predict(model2, test_X, type = "class")
acc_table_rf= table(predTrain_rf,test_y)
cm_cat_rf = confusionMatrix(acc_table_rf)
cm_cat_rf$overall['Accuracy']
```

```
## [1] "Accuracy 0.711985"
```

With this Accuracy of 71%, We reached pretty close to SVM.

### Evaluation Support Vector Regressor

```
predicted <- predict(svm_model_regression, test_X)
RMSE_Custom = sqrt(mean(predicted - test_y)^2)
RMSE_Custom
```

```
## [1] "98.03"
```

RMSE(98) is quite large for this model, this can be reduced by hyperparameter tuning in the same way as it was done for SVM above.