Salary_predictions_ml

Sofia Gambirasio

```r
library(tidyverse)
library(tidymodels)
library(knitr)
```

Import the dataset elaborated in python

```r
sal = read_csv('C:\\Users\\sofia\\OneDrive\\Desktop\\Lavoro\\Portfolio\\machine learning py\\salary_mod
sal = sal %>%
  rename(index = '...1')
sal_ml = sal %>%
  select(!Gender & !index & !Edu_level)

# insert _ in place of spaces
names(sal_ml) <- gsub(" ", "_", names(sal_ml))
glimpse(sal_ml)
```

```
## Rows: 4,438
## Columns: 21
## $ Years_experience            <dbl> 5, 3, 2, 12, 1, 3, 16, 7, 13, 3, 7, 3, 22,~
## $ Salary                      <dbl> 90000, 65000, 55000, 120000, 45000, 75000,~
## $ J_Back_end_Developer        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Data_Analyst              <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Data_Scientist            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ~
## $ J_Financial_Manager         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ~
## $ J_Front_end_Developer       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Full_Stack_Engineer       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Human_Resources_Manager   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Junior_Sales_Associate    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Marketing_Analyst         <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ~
## $ J_Marketing_Coordinator     <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Marketing_Manager         <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ~
## $ J_Operations_Manager        <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, ~
## $ J_Product_Manager           <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ~
## $ J_Senior_Project_Engineer   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Senior_Software_Engineer  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ~
## $ J_Software_Developer        <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Software_Engineer         <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ J_Software_Engineer_Manager <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ Gender_e                    <dbl> 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, ~
```

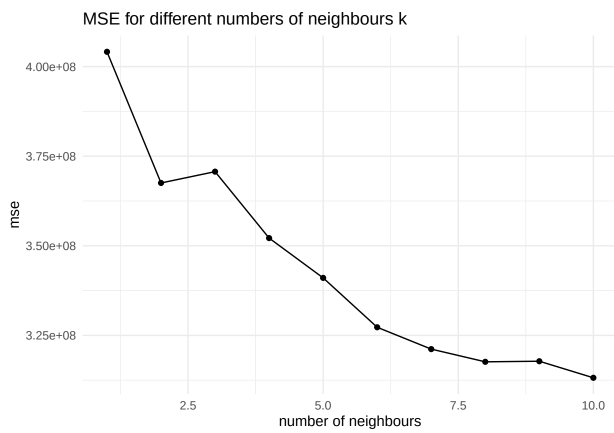
Data Splitting

Creating the training (70%), test(20%) and validation(10%) sets of data

```r
set.seed(12, sample.kind = 'Rejection')
group = sample(c(1, 2, 3),
               size = nrow(sal_ml),
               prob = c(0.7, 0.20, 0.10),
               replace = T)

train_sal = sal_ml[group == 1, ]
test_sal = sal_ml[group == 2, ]
val_sal = sal_ml[group == 3, ]
```

KNN

```r
library(kknn)

# tuning the k
k = 1:10
mse_vec = c()
for (i in k){
  knn_spec = nearest_neighbor(neighbors = i, weight_func = 'rectangular') %>%
    set_engine('kknn') %>%
    set_mode('regression')
  knn_fit = knn_spec %>% fit(Salary ~ ., data = train_sal)
  knn_pred = knn_fit %>% predict(new_data = val_sal)
  mse_vec[i] = mean((val_sal$Salary-knn_pred$.pred)^2)
}

# plotting the mse vs the k
data.frame(k, mse_vec) %>%
  ggplot(aes(x = k, y = mse_vec))+
  geom_line()+
  geom_point()+
  labs(title = 'MSE for different numbers of neighbours k',
       y = 'mse', x = 'number of neighbours')+
  theme_minimal()
```




```r
# save the k for the minimum mse
k_min = which.min(mse_vec)

# implement the model with the tuned value for k. k = 6
```

```
knn_spec = nearest_neighbor(neighbors = k_min, weight_func = 'rectangular') %>%
    set_engine('kknn') %>%
    set_mode('regression')
knn_fit = knn_spec %>% fit(Salary ~ ., data = train_sal)
knn_pred = knn_fit %>% predict(new_data = test_sal)

# save model assessment measures
knn = data.frame(
  mod = 'knn',
  mse = mean((test_sal$Salary-knn_pred$.pred)^2),
  cor = cor(test_sal$Salary,knn_pred$.pred)
)
```

Regression tree

```
library(rpart.plot)
library(randomForest)
library(parsnip)
```

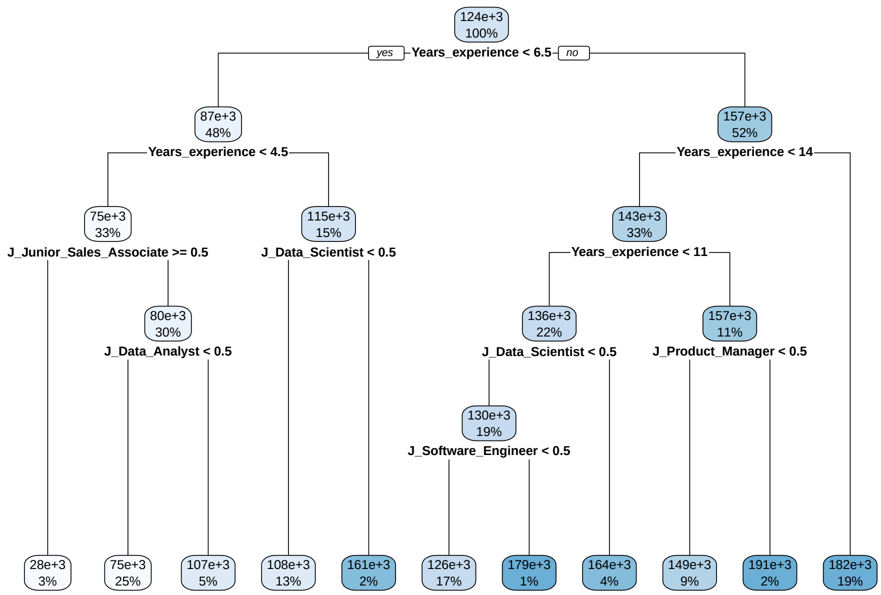
We generate the tree and then prune it using the 1-SE approach

```
# fit regression tree
set.seed(1,sample.kind = 'Rejection')
tree_spec = decision_tree() %>%
  set_engine('rpart')  %>%
  set_mode('regression')
tree_fit = tree_spec %>% fit(Salary~., data = train_sal)
tree_pred = tree_fit %>% predict(new_data = test_sal)

# save model assessment
tree = data.frame(
  mod = 'tree',
  mse = mean((test_sal$Salary-tree_pred$.pred)^2),
  cor = cor(test_sal$Salary,tree_pred$.pred)
)

# pruning the tree
cptable = data.frame(tree_fit$fit$cptable)
mincpindex = which.min(cptable[,"xerror"])
LL = cptable[mincpindex,"xerror"] - cptable[mincpindex,"xstd"]
UL = cptable[mincpindex,"xerror"] + cptable[mincpindex,"xstd"]
pos = which((cptable[,"xerror"] > LL) & (cptable[,"xerror"] < UL)) %>% min()
best_cp = cptable$CP[pos]
print(paste('The number of splits in the tree passes from',cptable$nsplit[10],'to',cptable$nsplit[pos]))
```

```
## [1] "The number of splits in the tree passes from 11 to 10"
```

```
# use the pruned tree to make predictions
set.seed(1,sample.kind = 'Rejection')
tree_spec_pr = decision_tree(cost_complexity = best_cp) %>%
  set_engine('rpart')  %>%
  set_mode('regression')
tree_fit_pr = tree_spec_pr %>% fit(Salary~., data = train_sal)
tree_pred_pr = tree_fit_pr %>% predict(new_data = test_sal)
```

```r
# plot pruned tree
rpart.plot(tree_fit_pr$fit, roundint = F)
```

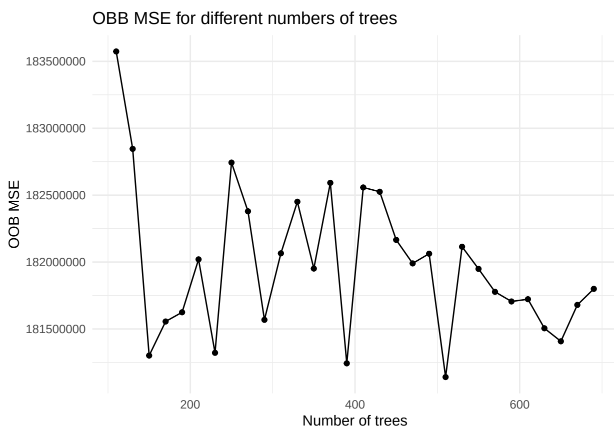



```r
# save assessment measures of the pruned tree
tree_pr = data.frame(
  mod = 'tree pruned',
  mse = mean((test_sal$Salary-tree_pred_pr$.pred)^2),
  cor = cor(test_sal$Salary,tree_pred_pr$.pred)
)
```

Bagging

```r
# tuning the number of trees
set.seed(1, sample.kind="Rejection")

ntree_vec = seq(110,700,by=20)
obb_mse = c()

for (i in 1:length(ntree_vec)){
  bag_spec = rand_forest(mtry = ncol(train_sal)-1,
                         trees = ntree_vec[i]) %>%
  set_engine('randomForest',importance = T) %>%
  set_mode('regression')
  bag_fit = bag_spec %>% fit(Salary ~ ., data = train_sal)
  obb_mse[i] = bag_fit$fit$mse[ntree_vec[i]]
}

# plotting the mse vs the number of trees
data.frame(ntree_vec,obb_mse) %>%
  ggplot(aes(ntree_vec,obb_mse)) +
  geom_line()+
```

```r
  geom_point()+
  labs(title = 'OBB MSE for different numbers of trees')+
  xlab("Number of trees")+
  ylab("OOB MSE")+
  theme_minimal()
```




```r
# saving the new optimal number of trees
n_trees = ntree_vec[which.min(obb_mse)]

# computing the model with the tuned number of trees 150
bag_spec = rand_forest(mtry = ncol(train_sal)-1,
                       trees = n_trees,
                       ) %>%
  set_engine('randomForest',importance = T) %>%
  set_mode('regression')
bag_fit = bag_spec %>% fit(Salary ~ ., data = train_sal)

# predictions
bag_pred = predict(bag_fit, new_data = test_sal)

bag = data.frame(
  mod = 'bagging',
  mse = mean((test_sal$Salary - bag_pred$.pred)^2),
  cor = cor(test_sal$Salary, bag_pred$.pred)
)
```

```r
# variable importance
tab_varimp = as.data.frame(bag_fit$fit$importance) %>%
  arrange(desc(`%IncMSE`))
kable(tab_varimp, align = 'l')
```

	%IncMSE	IncNodePurity
Years_experience	3609903921	5.422656e+12
J_Data_Scientist	371313291	3.589740e+11
J_Software_Engineer	240915022	2.533504e+11
J_Data_Analyst	215624234	2.540331e+11
J_Product_Manager	171021399	1.833043e+11
Gender_e	169104922	1.274426e+11
J_Senior_Project_Engineer	48935696	4.799577e+10

	%IncMSE	IncNodePurity
J__Junior__Sales__Associate	48168014	2.092335e+11
J__Software__Engineer__Manager	43307296	3.701877e+10
J__Full__Stack__Engineer	35812934	3.997379e+10
J__Marketing__Manager	31529667	3.994284e+10
J__Financial__Manager	29131947	3.001170e+10
J__Operations__Manager	28313725	5.323228e+10
J__Back__end__Developer	24570421	2.926590e+10
J__Senior__Software__Engineer	22195369	2.982893e+10
J__Human__Resources__Manager	20068549	3.293372e+10
J__Marketing__Coordinator	16249506	2.147008e+10
J__Front__end__Developer	14293032	1.830093e+10
J__Software__Developer	7549445	1.221680e+10
J__Marketing__Analyst	4959251	5.686242e+09

Random forest

```r
# tune the number of trees
set.seed(1, sample.kind="Rejection")

ntree_vec = seq(110,700,by=20)
obb_mse = c()

for (i in 1:length(ntree_vec)){
  rf_spec = rand_forest(mtry = (ncol(train_sal)-1)/3,
                       trees = ntree_vec[i]) %>%
  set_engine('randomForest',importance = T) %>%
  set_mode('regression')
  rf_fit = rf_spec %>% fit(Salary ~ ., data = train_sal)
  obb_mse[i] = rf_fit$fit$mse[ntree_vec[i]]
}
# select the numer of trees with the minum mse
n_trees = ntree_vec[which.min(obb_mse)]

# make predictions using the tuned number of trees = 570
set.seed(1, sample.kind="Rejection")
rf_spec = rand_forest(mtry = (ncol(train_sal)-1)/3,
                        trees = n_trees) %>%
  set_engine('randomForest',importance = T) %>%
  set_mode('regression')
rf_fit = rf_spec %>% fit(Salary ~ ., data = train_sal)
rf_pred = rf_fit %>% predict(new_data = test_sal)

# save model assessment
rf = data.frame(
  mod = 'random forest',
  mse = mean((test_sal$Salary-rf_pred$.pred)^2),
  cor = cor(test_sal$Salary,rf_pred$.pred)
)
```

Gradient boosting

```
library(gbm)

# tuning the learning rate lambda and the interaction depth d
hyper_grid = expand.grid(
  shrinkage = c(0.005, .01, .1),
  interaction.depth = c(1, 3, 5)
)
for(i in 1:nrow(hyper_grid)){
  set.seed(1, sample.kind="Rejection")
  gb_fit = gbm(formula = Salary ~ .,
                  data = train_sal,
                  distribution = "gaussian",
                  n.trees = 5000,
                  shrinkage = hyper_grid$shrinkage[i],
                  interaction.depth = hyper_grid$interaction.depth[i],
                  cv.folds = 5)
  hyper_grid$minMSE[i] = min(gb_fit$cv.error)
  hyper_grid$bestB[i] = which.min(gb_fit$cv.error)
}
# select the combination of parameters corresponding to the minimum mse
min_parameters = hyper_grid %>%
  filter(minMSE == min(minMSE))
print(min_parameters)
```

```
##   shrinkage interaction.depth    minMSE bestB
## 1       0.1                 5 186291847  4462
```

```
# make predictions with the tuned model
set.seed(1, sample.kind="Rejection")
gb_fit = gbm(formula = Salary ~ .,
                  data = train_sal,
                  distribution = "gaussian",
                  n.trees = min_parameters$bestB,
                  shrinkage = min_parameters$shrinkage,
                  interaction.depth = min_parameters$interaction.depth,
                  cv.folds = 5)
gb_pred = gb_fit %>% predict(test_sal, n.trees = min_parameters$bestB)

# save the model assessment
gb = data.frame(
  mod = 'gradient boosting',
  mse = mean((test_sal$Salary-gb_pred)^2),
  cor = cor(test_sal$Salary, gb_pred)
)
```

Model comparison

we compare the mean square error and the pearson correlation coefficient of the different models to see that bagging is the one that provides the best performance.

```
# table of comparisons of the models
assessment_df = rbind(knn, tree, tree_pr, bag, rf, gb) %>%
```

```r
  arrange(mse)
kable(assessment_df, align = 'l')
```

mod	mse	cor
bagging	172679435	0.9628075
gradient boosting	185898412	0.9599515
knn	248744296	0.9484078
random forest	253580491	0.9459170
tree	584313273	0.8680604
tree pruned	618990496	0.8598116

```r
# plot the bagging predictions vs the true values
data.frame(pred = gb_pred, true = test_sal$Salary) %>%
  ggplot()+
  geom_point(aes(x = pred, y = true))+
  geom_abline(slope = 1, intercept = 0, color = 'red')+
  labs(title = 'Bagging salary prediction vs true salaries',
       x = 'predicted values', y = 'true values')
```