Predictive Analytics & Data Mining on Campus Placements – Team 4

Final Assignment

Team Members:

Prashanth ASOK KUMAR

Annanya CHITRA KANNAN

Levon AVETISYAN

Vasanth MOHAN

Introduction

We are a group of 4, and we have split our task to work on both R-studio and Dataiku. Our data set is regarding campus placement of students at Jain University in Bangalore, India.

Link: https://www.kaggle.com/benroshan/factors-affecting-campus-placement

R-Studio Part

We are trying to try to predict using if the student will be placed or not and what will be his salary if placed. We are using the following 3 models for prediction.

- Linear Regression Model
- Knn classification Model
- Naive Bayes Model

Before starting with the prediction, lets see if there are any relation of the data points with status and salary.

Libraries used:

```
library(e1071)
library(ggplot2)
library(tidyr)
library(gmodels)
library(class)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Importing Data and Data Correction:

Removing the Serial Number Column and replacing the NA in salary column with 0

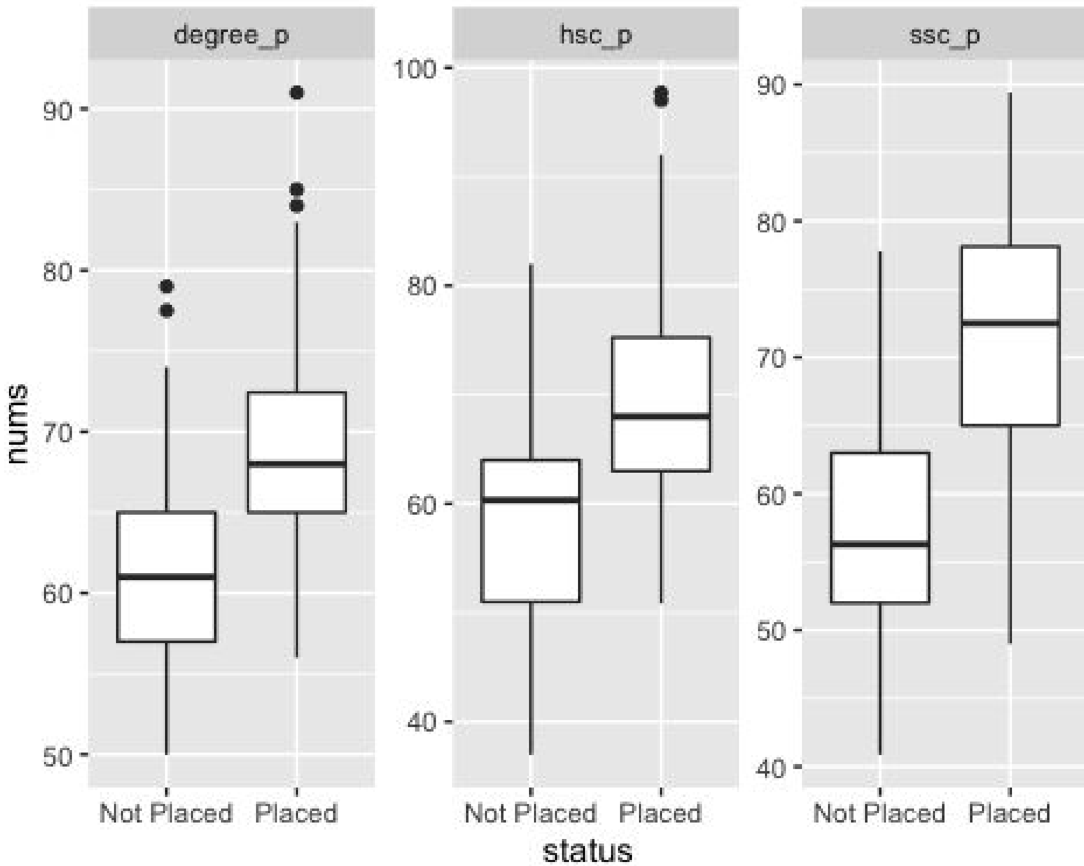
```r
placement<-read.csv("Placement_Data_Full_Class.csv", stringsAsFactors=TRUE)
placement<-placement[-1]
placement[is.na(placement)] <- 0
str(placement)
```

```
## 'data.frame':    215 obs. of  14 variables:
##  $ gender        : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 1 2 2 2 ...
##  $ ssc_p         : num  67 79.3 65 56 85.8 ...
##  $ ssc_b         : Factor w/ 2 levels "Central","Others": 2 1 1 1 1 2 2 1
1 1 ...
##  $ hsc_p         : num  91 78.3 68 52 73.6 ...
##  $ hsc_b         : Factor w/ 2 levels "Central","Others": 2 2 1 1 1 2 2 1
1 1 ...
##  $ hsc_s         : Factor w/ 3 levels "Arts","Commerce",..: 2 3 1 3 2 3 2
3 2 2 ...
##  $ degree_p      : num  58 77.5 64 52 73.3 ...
##  $ degree_t      : Factor w/ 3 levels "Comm&Mgmt","Others",..: 3 3 1 3 1 3
1 3 1 1 ...
##  $ workex        : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 1 ...
##  $ etest_p       : num  55 86.5 75 66 96.8 ...
##  $ specialisation: Factor w/ 2 levels "Mkt&Fin","Mkt&HR": 2 1 1 2 1 1 1 1
1 1 ...
##  $ mba_p         : num  58.8 66.3 57.8 59.4 55.5 ...
##  $ status        : Factor w/ 2 levels "Not Placed","Placed": 2 2 2 1 2 1 1
2 2 1 ...
##  $ salary        : num  270000 200000 250000 0 425000 0 0 252000 231000 0
...
```

Relation of data points with salary and status:

Relationship with status: The following have linearly positive relation with status:

```r
placement%>%gather(placement,nums,c("ssc_p","hsc_p","degree_p" ))%>%
  ggplot(aes(x=status,y=nums))+geom_boxplot() + facet_wrap(~ placement,
scales = "free_y")
```



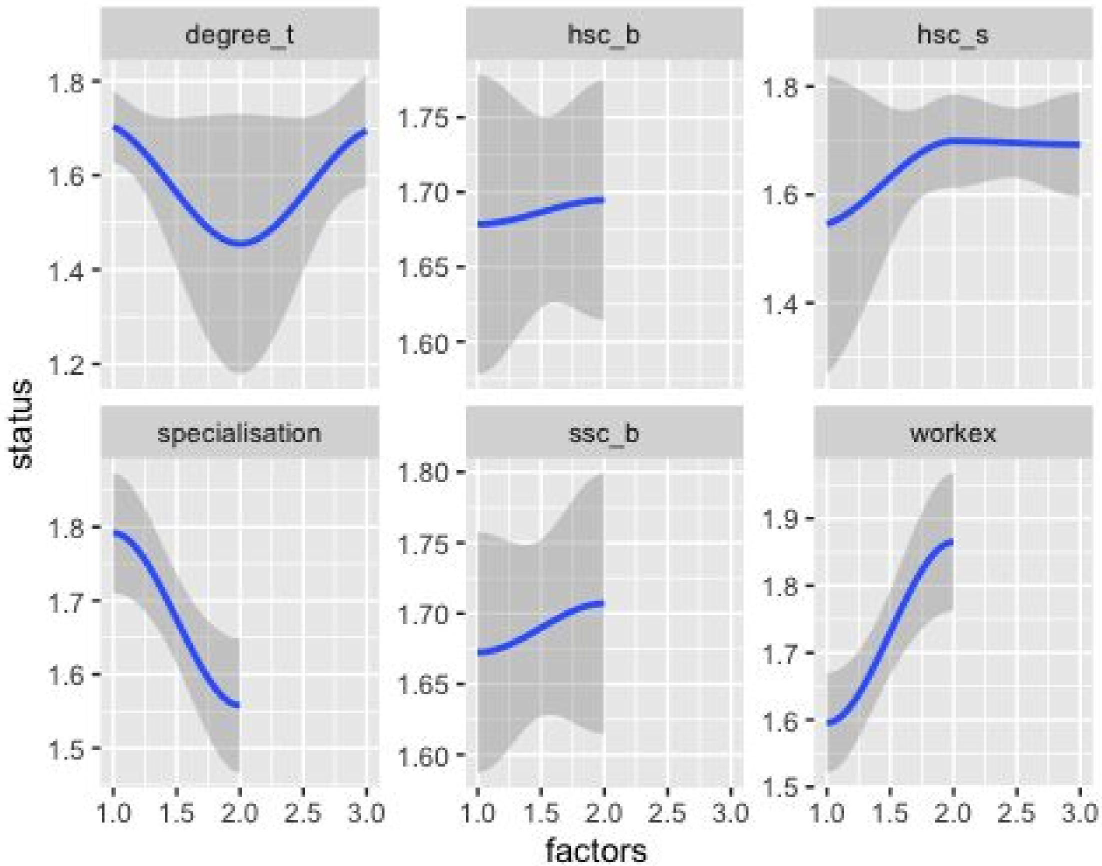
For better visualization of relationship with factor data, we are converting them to numerical

```
numPlacement<-placement
indx <- sapply(placement, is.factor)
numPlacement[indx] <- lapply(placement[indx], function(x) as.numeric(x))
```

Relationship with salary: Making use of the converted data to plot and show the relationship with salary

```
numPlacement%>%gather(numPlacement,factors,c("ssc_b","hsc_b",
"hsc_s","degree_t","workex","specialisation"))%>%
  ggplot(aes(x=factors,y=status))+geom_smooth()+ facet_wrap(~ numPlacement,
scales = "free_y")

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Linear Regression Model to predict the salary:

```
status_trainmodel_lm<-numPlacement[1:200,]
status_testmodel_lm<-numPlacement[201:215,]
status_trainmodel_lm_backup<-status_trainmodel_lm
status_trainmodel_lm<-lm(status~.,status_trainmodel_lm_backup)
status_pred<-predict(status_trainmodel_lm,status_testmodel_lm)
cor(status_testmodel_lm$status,status_pred)

## [1] 0.9480045
```

Comparing the status predicted with the test data: Rounding off the predicted value:

```
status_pred<-round(status_pred,digits = 0)
comparaison<-cbind(status_testmodel_lm$status,status_pred)
status_trainmodel_lm

##
## Call:
## lm(formula = status ~ ., data = status_trainmodel_lm_backup)
```

```
##
## Coefficients:
##    (Intercept)          gender           ssc_p           ssc_b
hsc_p
##      8.010e-01      -9.249e-03       9.720e-03       9.400e-03
3.266e-03
##          hsc_b           hsc_s        degree_p        degree_t
workex
##     -3.096e-04       5.450e-03       8.357e-03      -5.145e-02
5.478e-02
##        etest_p  specialisation           mba_p          salary
##     -2.217e-03       7.392e-03      -1.286e-02       2.027e-06
```

```
comparaison
```

```
##       status_pred
## 201 2           2
## 202 1           1
## 203 2           2
## 204 2           2
## 205 2           2
## 206 2           2
## 207 1           1
## 208 2           2
## 209 1           1
## 210 2           2
## 211 2           2
## 212 2           2
## 213 2           2
## 214 2           2
## 215 1           1
```

Labeling the status values to concerning labels:

```
status_pred[status_pred==2]<-"Placed"
status_pred[status_pred==1]<-"Not Placed"
status_testmodel_lm$status[status_testmodel_lm$status==1]<-"Not Placed"
status_testmodel_lm$status[status_testmodel_lm$status==2]<-"Placed"
comparaison<-cbind(status_testmodel_lm$status,status_pred)
status_trainmodel_lm
```

```
##
## Call:
## lm(formula = status ~ ., data = status_trainmodel_lm_backup)
##
## Coefficients:
##    (Intercept)          gender           ssc_p           ssc_b
hsc_p
##      8.010e-01      -9.249e-03       9.720e-03       9.400e-03
```

```
3.266e-03
##          hsc_b           hsc_s        degree_p        degree_t
workex
##     -3.096e-04       5.450e-03       8.357e-03      -5.145e-02
5.478e-02
##        etest_p  specialisation           mba_p          salary
##     -2.217e-03       7.392e-03      -1.286e-02       2.027e-06
```

```
comparaison
```

```
##                    status_pred
## 201 "Placed"     "Placed"
## 202 "Not Placed" "Not Placed"
## 203 "Placed"     "Placed"
## 204 "Placed"     "Placed"
## 205 "Placed"     "Placed"
## 206 "Placed"     "Placed"
## 207 "Not Placed" "Not Placed"
## 208 "Placed"     "Placed"
## 209 "Not Placed" "Not Placed"
## 210 "Placed"     "Placed"
## 211 "Placed"     "Placed"
## 212 "Placed"     "Placed"
## 213 "Placed"     "Placed"
## 214 "Placed"     "Placed"
## 215 "Not Placed" "Not Placed"
```

Predict the salary using the Linear Regression Method:

```
salary_trainmodel_lm<-placement[1:200,]
salary_testmodel_lm<-placement[201:215,]
salary_trainmodel_lm_backup<-salary_trainmodel_lm
salary_trainmodel_lm<-lm(salary~.,salary_trainmodel_lm_backup)
salary_pred<-predict(salary_trainmodel_lm,salary_testmodel_lm)
cor(salary_testmodel_lm$salary,salary_pred)
```

```
## [1] 0.950907
```

Comparing the salary predicted with the test values:

```
comparaison<-cbind(salary_testmodel_lm$salary,salary_pred)
comparaison
```

```
##             salary_pred
## 201 300000  284034.510
## 202      0  -16464.768
## 203 240000  295375.737
```

```
## 204 260000  292923.099
## 205 210000  296408.474
## 206 250000  284792.899
## 207      0   13395.782
## 208 300000  334151.869
## 209      0  -12407.543
## 210 216000  291125.733
## 211 400000  316476.131
## 212 275000  298198.751
## 213 295000  311757.438
## 214 204000  265419.231
## 215      0   -5348.827
```

Predict using knn classification and compare using Cross Table:

Status prediction:

```
normalize <- function(x) {return ((x - min(x)) / ((max(x) - min(x))))}

numPlacement_n <- as.data.frame(lapply(numPlacement[,1:12], normalize))

status_train_Knn<-numPlacement_n[1:200,]
status_test_Knn<-numPlacement_n[201:215,]

status_train_labels<-placement[1:200,13]
status_test_labels<-placement[201:215,13]

status_pred_knn<-knn(train = status_train_Knn, test = status_test_Knn,
cl=status_train_labels, k=15)

CrossTable(x=status_test_labels,y=status_pred_knn,prop.chisq=FALSE)

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  15
##
##
##                     | status_pred_knn
```

```
## status_test_labels | Not Placed |     Placed |  Row Total |
## -------------------|------------|------------|------------|
##         Not Placed |          3 |          1 |          4 |
##                    |      0.750 |      0.250 |      0.267 |
##                    |      0.750 |      0.091 |            |
##                    |      0.200 |      0.067 |            |
## -------------------|------------|------------|------------|
##             Placed |          1 |         10 |         11 |
##                    |      0.091 |      0.909 |      0.733 |
##                    |      0.250 |      0.909 |            |
##                    |      0.067 |      0.667 |            |
## -------------------|------------|------------|------------|
##       Column Total |          4 |         11 |         15 |
##                    |      0.267 |      0.733 |            |
## -------------------|------------|------------|------------|
##
##
```

Salary prediction:

```
numPlacement_salary_n <- as.data.frame(lapply(numPlacement[,1:13],
normalize))

salary_train_Knn<-numPlacement_salary_n[1:200,]
salary_test_Knn<-numPlacement_salary_n[201:215,]

salary_train_labels<-placement[1:200,14]
salary_test_labels<-placement[201:215,14]

salary_pred_knn<-knn(train = salary_train_Knn, test = salary_test_Knn,
cl=salary_train_labels, k=15)

CrossTable(x=salary_test_labels,y=salary_pred_knn,prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  15
##
##
```

```
##                    | salary_pred_knn
## salary_test_labels |         0 |    240000 |    250000 |    265000 |
3e+05 |     5e+05 | Row Total |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##                   0 |         4 |         0 |         0 |         0 |
0 |         0 |         4 |
##                    |     1.000 |     0.000 |     0.000 |     0.000 |
0.000 |     0.000 |     0.267 |
##                    |     0.800 |     0.000 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##                    |     0.267 |     0.000 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##             204000 |         0 |         0 |         0 |         1 |
0 |         0 |         1 |
##                    |     0.000 |     0.000 |     0.000 |     1.000 |
0.000 |     0.000 |     0.067 |
##                    |     0.000 |     0.000 |     0.000 |     0.500 |
0.000 |     0.000 |           |
##                    |     0.000 |     0.000 |     0.000 |     0.067 |
0.000 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##             210000 |         0 |         0 |         0 |         0 |
1 |         0 |         1 |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
1.000 |     0.000 |     0.067 |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
0.500 |     0.000 |           |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
0.067 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##             216000 |         0 |         0 |         0 |         0 |
1 |         0 |         1 |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
1.000 |     0.000 |     0.067 |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
0.500 |     0.000 |           |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
0.067 |     0.000 |           |
##
```

```
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##             240000 |         1 |         0 |         0 |         0 |
0 |         0 |         1 |
##                    |     1.000 |     0.000 |     0.000 |     0.000 |
0.000 |     0.000 |     0.067 |
##                    |     0.200 |     0.000 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##                    |     0.067 |     0.000 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##             250000 |         0 |         1 |         0 |         0 |
0 |         0 |         1 |
##                    |     0.000 |     1.000 |     0.000 |     0.000 |
0.000 |     0.000 |     0.067 |
##                    |     0.000 |     0.250 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##                    |     0.000 |     0.067 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##             260000 |         0 |         0 |         0 |         1 |
0 |         0 |         1 |
##                    |     0.000 |     0.000 |     0.000 |     1.000 |
0.000 |     0.000 |     0.067 |
##                    |     0.000 |     0.000 |     0.000 |     0.500 |
0.000 |     0.000 |           |
##                    |     0.000 |     0.000 |     0.000 |     0.067 |
0.000 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##             275000 |         0 |         0 |         0 |         0 |
0 |         1 |         1 |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
0.000 |     1.000 |     0.067 |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
0.000 |     1.000 |           |
##                    |     0.000 |     0.000 |     0.000 |     0.000 |
0.000 |     0.067 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##             295000 |         0 |         1 |         0 |         0 |
0 |         0 |         1 |
```

```
##                    |     0.000 |     1.000 |     0.000 |     0.000 |
0.000 |     0.000 |     0.067 |
##                    |     0.000 |     0.250 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##                    |     0.000 |     0.067 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##              3e+05 |         0 |         1 |         1 |         0 |
0 |         0 |         2 |
##                    |     0.000 |     0.500 |     0.500 |     0.000 |
0.000 |     0.000 |     0.133 |
##                    |     0.000 |     0.250 |     1.000 |     0.000 |
0.000 |     0.000 |           |
##                    |     0.000 |     0.067 |     0.067 |     0.000 |
0.000 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##              4e+05 |         0 |         1 |         0 |         0 |
0 |         0 |         1 |
##                    |     0.000 |     1.000 |     0.000 |     0.000 |
0.000 |     0.000 |     0.067 |
##                    |     0.000 |     0.250 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##                    |     0.000 |     0.067 |     0.000 |     0.000 |
0.000 |     0.000 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##       Column Total |         5 |         4 |         1 |         2 |
2 |         1 |        15 |
##                    |     0.333 |     0.267 |     0.067 |     0.133 |
0.133 |     0.067 |           |
##
-------------------|-----------|-----------|-----------|-----------|---------
--|-----------|-----------|
##
##
```

Naive_bayes algorithm

Naive bayes can only be used to predict Categorical data and not continuous data. So, we can only predict Status using it.

status Prediction:

```r
status_NB<-naiveBayes(as.factor(status)~.,data=placement)
status_prediction_NB<-predict(status_NB,placement[201:215,])

CrossTable(x=status_test_labels,y=status_prediction_NB,prop.chisq=FALSE)

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  15
##
##
##                    | status_prediction_NB
## status_test_labels | Not Placed |     Placed |  Row Total |
## -------------------|------------|------------|------------|
##         Not Placed |          4 |          0 |          4 |
##                    |      1.000 |      0.000 |      0.267 |
##                    |      0.333 |      0.000 |            |
##                    |      0.267 |      0.000 |            |
## -------------------|------------|------------|------------|
##             Placed |          8 |          3 |         11 |
##                    |      0.727 |      0.273 |      0.733 |
##                    |      0.667 |      1.000 |            |
##                    |      0.533 |      0.200 |            |
## -------------------|------------|------------|------------|
##       Column Total |         12 |          3 |         15 |
##                    |      0.800 |      0.200 |            |
## -------------------|------------|------------|------------|
##
##
```

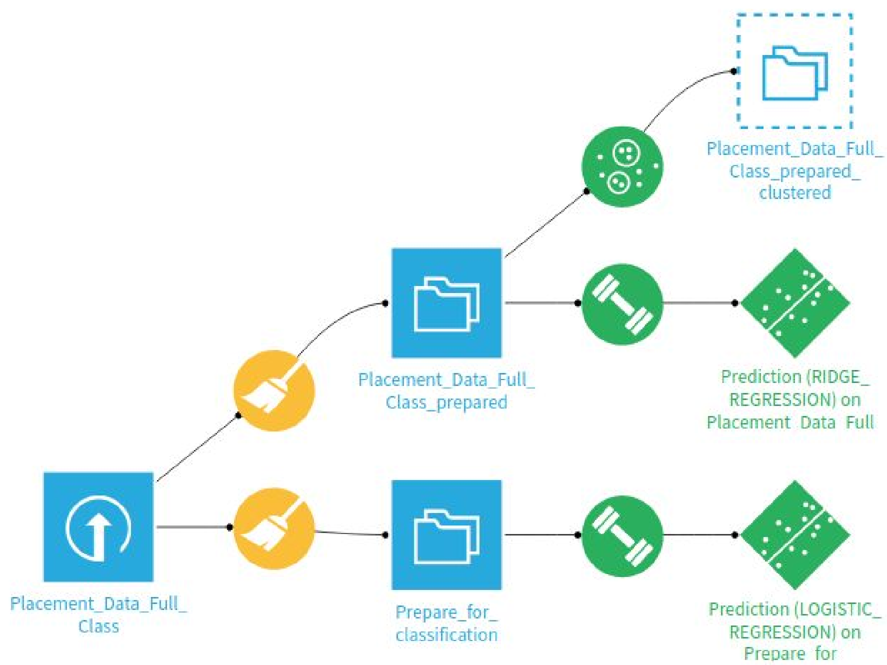
Dataiku:

The dataset was processed through different predictive models to determine the two different values of the dataset.

- Predict if the student was Placed or Not
- If the student was placed, what was the salary?

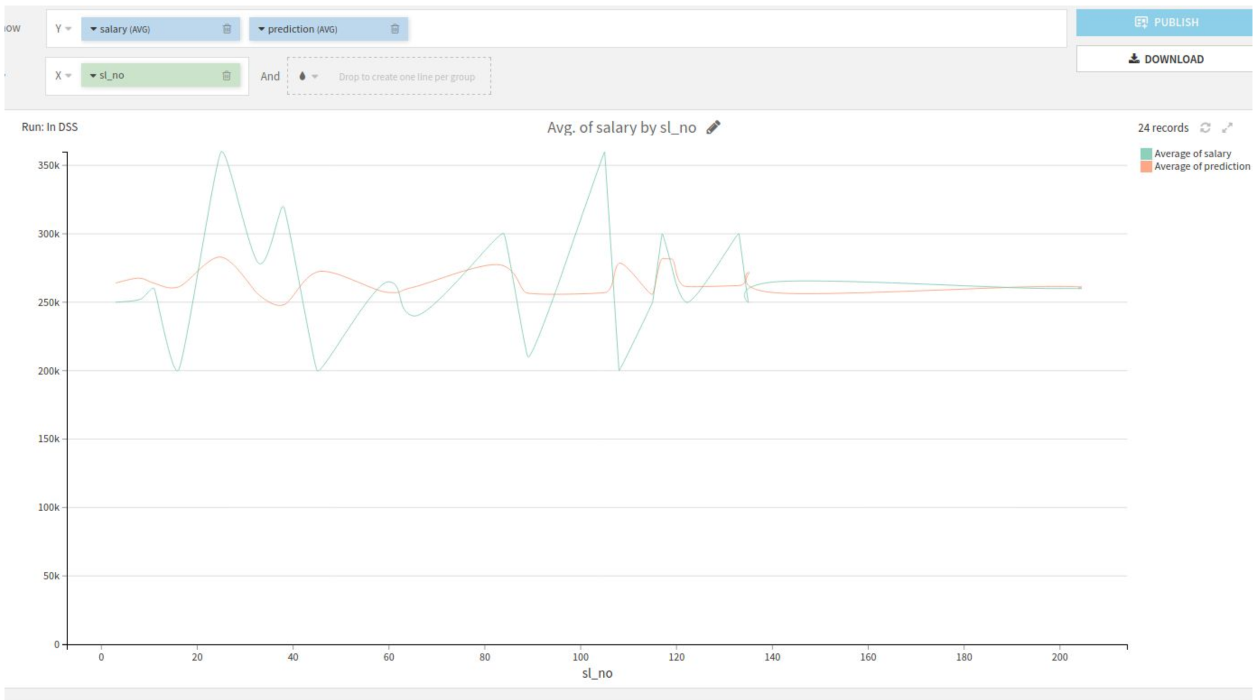
Flow:

Dataiku's intuitive interface makes it easier to describe the overall processing of data. You will find below, the overview of the data models designed and predictive analytics handled.



Regression to predict Salary:

Ridge (L2) Regression



This chart depicts a line chart comparing the predicted and actual values of salary for the test data

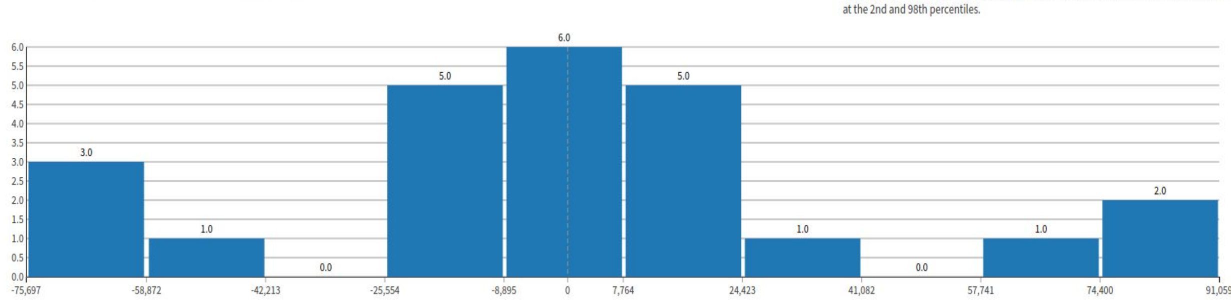
Error Distribution

Min. (raw)	Min.	25th perc.	Median	75th perc.	90th perc.	Max.	Max. (raw)
-78438	-75531	-16894	-2663.9	19450	61718	91059	1.0302e+5

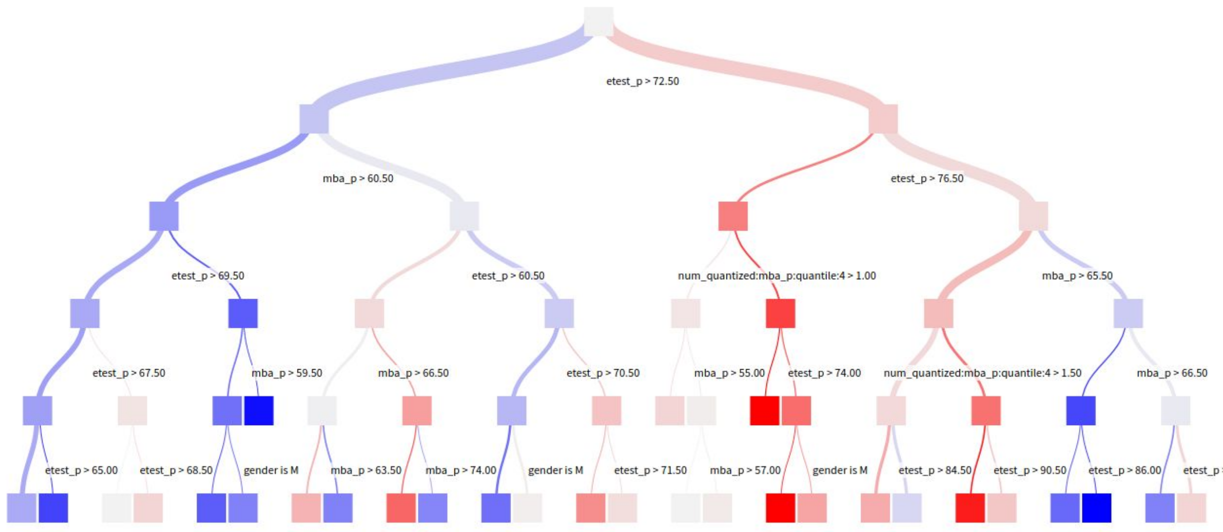
Average	281.75	Standard deviation	42097

The errors (difference between predicted and actual values) should be centered around zero, and the distribution should be "narrow", i.e the spread of the error should be limited. More generally, the errors should be "normally" distributed around zero (the curve should look like a bell).

To reduce the effect of possible spurious outliers, error distribution is winsorized (clipped) at the 2nd and 98th percentiles.

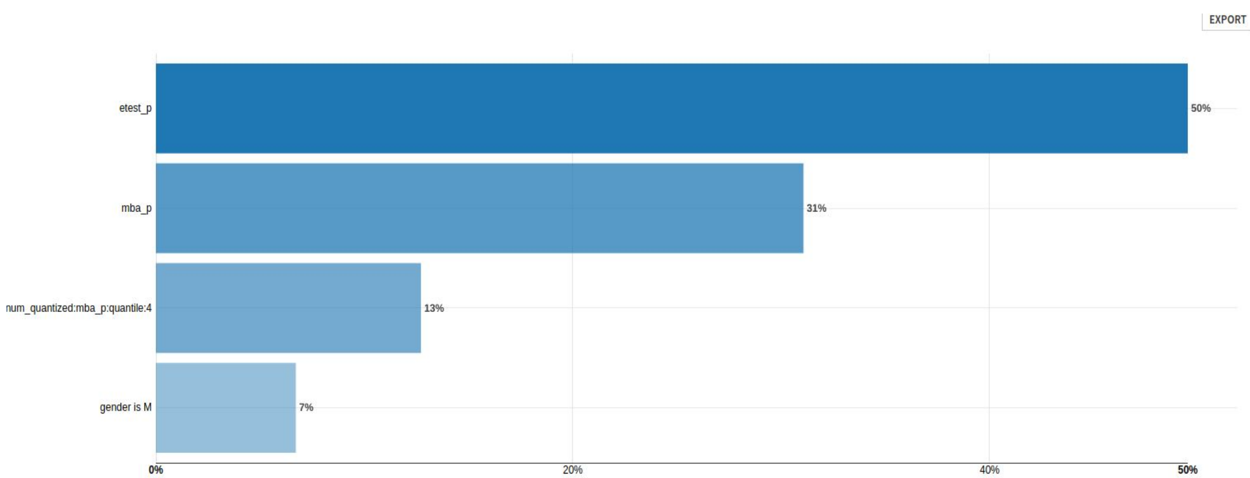


Decision Tree to predict Salary:



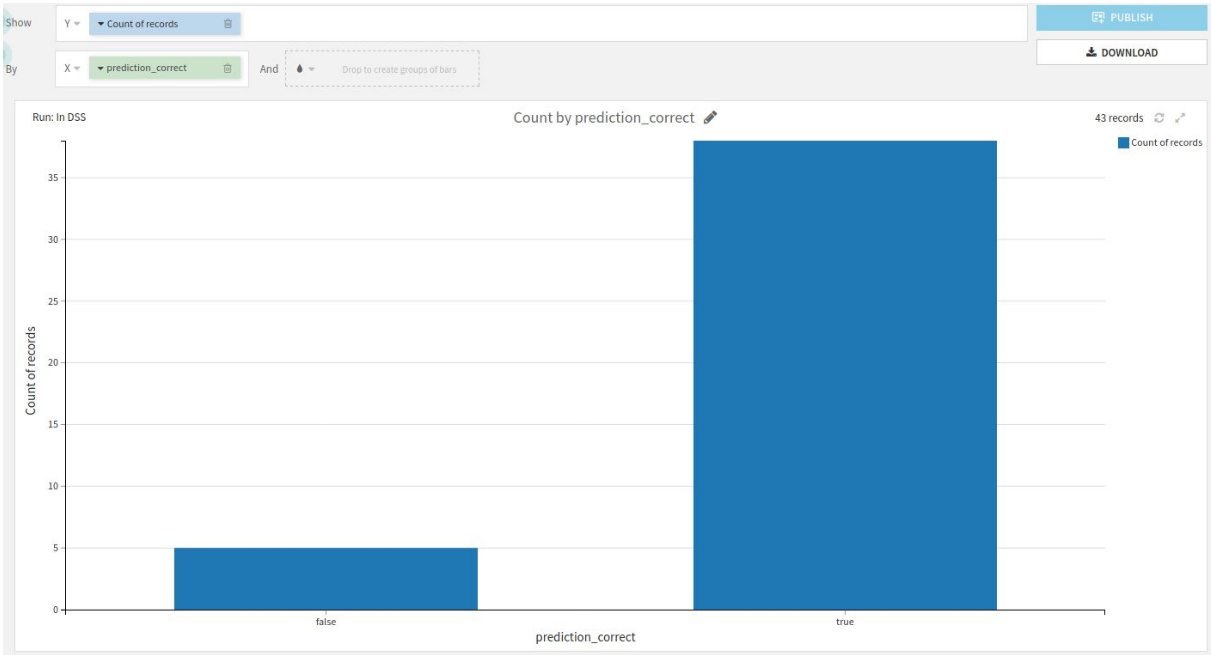
The Decision Tree was applied over the same dataset that was prepared for L2 regression.

Important columns that impact the prediction



Classification to predict Placed or not:

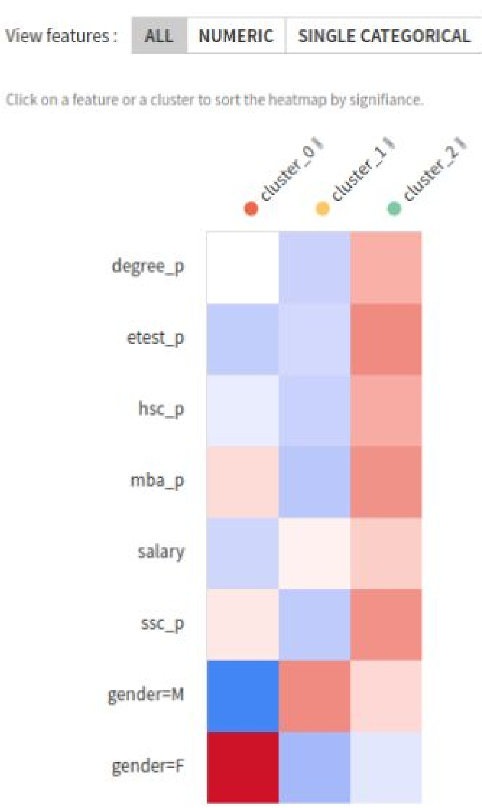
Logistic Regression



Threshold-dependent (current threshold = 0.8750)	
Accuracy Proportion of correct predictions (positive and negative) in the test set	0.8837
Precision Proportion of positive predictions that were indeed positive (in the test set)	0.7692
Recall Proportion of actual positive values found by the classifier	0.8333
F1 Score Harmonic mean between Precision and Recall	0.8000
Hamming loss Fraction of labels that are incorrectly predicted (the lower the better)	0.1163
Matthews Correlation Coefficient Correlation coefficient between actual and predicted values. +1 = perfect, 0 = no correlation, -1 = perfect anti-correlation	0.7194
Threshold-independent	
Log loss Error metric that takes into account the predicted probabilities (the lower the better)	0.5272
ROC - AUC Score Area under the ROC; from 0.5 (random model) to 1 (perfect model)	0.9382
Calibration loss Average distance between calibration curve and diagonal. From 0 (perfectly calibrated) up to 0.5.	0.2303

The Classification model was targeted to predict that "status" column of the dataset. From the model we can see that accuracy was very high and there is a distinct possibility of the data being overfitted

Clustering data to extract further info:



Here we can see that,

- Cluster_0 : Gender being Male is a significant factor for the datapoint
- Cluster_1 : The values of scores from ssc, hsc, mba and etest all are equally important. We also see that data points with Gender being *Female* is more likely to fall under this group
- Cluster_2 : There doesn't seem to be any significant associativity with any of the columns for this cluster

These clusters can be divided up into different datasets and can be used for cleaner and more accurate analysis of the patterns within them.