* + **Introduction-Background Review**
  + Review of the Current Model
  + **Load and check data**
* 2.1 load data
* 2.2 check for missing values
  + **Global data exploration**
  + **Detailed data exploration**
* 4.1 Normalisation and dimensionalty reduction
* 4.2 Global Radar Chart
* 4.3 Left and other features
* 4.4 Clustering analysis
* **5 Modeling**
* 5.1 Decision Tree
* Proposed Model
* Better Modifications
* Challenges
* Conclusions and Discussions

**Human Resources Analytics : Exploration Data**

**Analysis and modeling**

# Yassine Ghouzam, PhD

**17/08/2017**

I really enjoyed writing this notebook. If you like it or it helps you , you can upvote and/or leave a comment :).

* 1. **Introduction**
  2. **Load and check data**

2.1 load data

2.2 check for missing values

* 1. **Global data exploration**
  2. **Detailed data exploration**

4.1 Normalisation and dimensionalty reduction

4.2 Global Radar Chart

4.3 Left and other features

4.4 Clustering analysis

**5 Modeling**

5.1 Decision Tree

5.2 Random Forest

# 1. Introduction

The Human Resources Analytics is a dataset providing informations on the situation and work of several ten of thousands employees.

**In this kernel ill focus on one very important question : Why employees are leaving the company ?**

To tackle this question , this notebook combines data exploration analysis and modeling.

This dataset is perfect for this kind of detailed data exploration because it contains a few number of features a large number of individual, so we can perform robust statistics. Firstlty, ill globally explore the dataset, then ill focus on a detailed exploration analysis of the stayed/left employees and ill end by the data modeling.

This script follows three main parts:

**Global data exploration**

**Detailed data exploration**

**Data modeling**

In [3]:



[(https://bokeh.p BokehJS 1y.0.4 sucdata.orceg)](https://bokeh.pydata.org/)ssfully loaded.

# 2. Load and Check data

**2.1 Load the data** In [4]:

*# Load the data*

dataset

=

pd

.

read\_csv

(

"./kaggle\_hr\_analytics.csv"

)

In [5]:

dataset

.

shape

Out[5]:

(14999, 10)

In [6]:

*# Look at the train set*

dataset

.

head

()

Out[6]:

**satisfaction\_level last\_evaluation number\_project average\_montly\_hours time\_spend\_co**

1. 0.80 0.86 5 262
2. 0.11 0.88 7 272
3. 0.85 0.91 5 226
4. 0.11 0.93 7 308

This dataset contains 14999 rows described by 10 features.

There are 8 numerical features and 2 categorical features.

Sales is non nominal

Salary is ordinal

The feature of our interests is the 'left' feature, it encoded in {0,1} 0 for stayed employees and 1 if not.

**4**

0.10

0.95

6

244

**2.2 check for missing values** In [7]:

*# Check for missing values*

dataset

.

isnull

()

.

any

()

Out[7]:

satisfaction\_level False last\_evaluation False number\_project False average\_montly\_hours False time\_spend\_company False Work\_accident False left False promotion\_last\_5years False sales False salary False dtype: bool

The dataset is already clean, there is no missing values at all , great !!!

# 3. Global data exploration

Here, i display histograms of the 10 features for a global analysis.

In [8]:

fig

,

axs

=

plt

.

subplots

(

ncols

=

2

,

figsize

=

(

12

,

6

))

g

=

sns

.

countplot

(

dataset

[

"sales"

]

,

ax

=

axs

[

0

])

plt

.

setp

(

g

.

get\_xticklabels

()

,

rotation

=

45

)

g

=

sns

.

countplot

(

dataset

[

"salary"

,

]

ax

=

axs

[

1

])

plt

.

tight\_layout

()

plt

.

show

()

plt

.

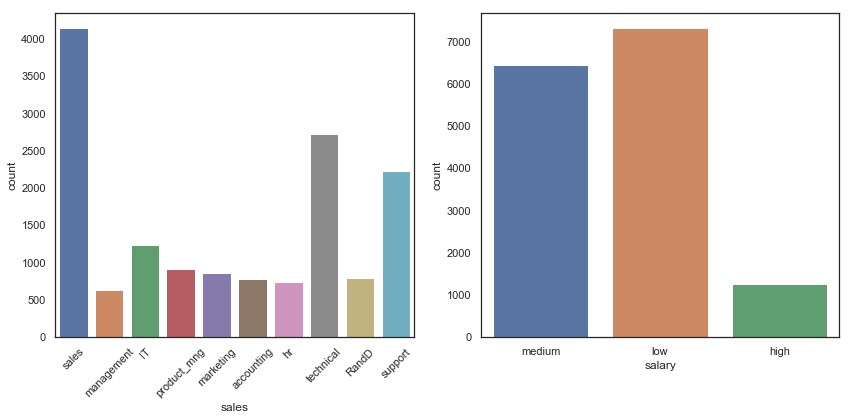
gcf

()

.

clear

()



<Figure size 432x288 with 0 Axes> In [9]:

fig

,

axs

=

plt

.

subplots

(

ncols

=

3

,

figsize

=

(

12

,

6

))

sns

.

countplot

(

dataset

[

"Work\_accident"

]

,

ax

=

axs

[

0

])

sns

.

countplot

(

dataset

[

"promotion\_last\_5years"

,

]

ax

=

axs

[

1

])

sns

.

countplot

(

dataset

[

"left"

,

]

ax

=

axs

[

2

])

plt

.

tight\_layout

()

plt

.

show

()

plt

.

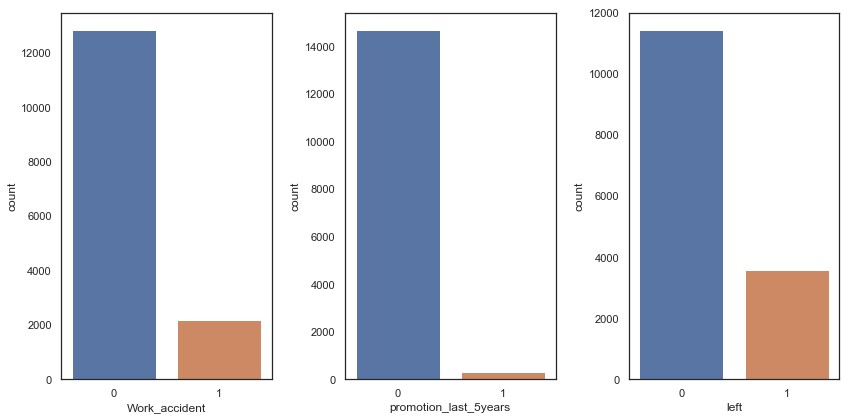
gcf

()

.

clear

()



<Figure size 432x288 with 0 Axes>

Our target variable (left) is unbalanced, since we have not more than 10x it is still reasonable.

In [10]:

fig

,

axs

=

plt

.

subplots

(

ncols

=

3

,

figsize

=

(

12

,

6

))

sns

.

distplot

(

dataset

[

"satisfaction\_level"

]

,

ax

=

axs

[

0

])

sns

.

distplot

(

dataset

[

"last\_evaluation"

,

]

ax

=

axs

[

1

])

sns

.

distplot

(

dataset

[

"average\_montly\_hours"

,

]

ax

=

axs

[

2

])

plt

.

tight\_layout

()

plt

.

show

()

plt

.

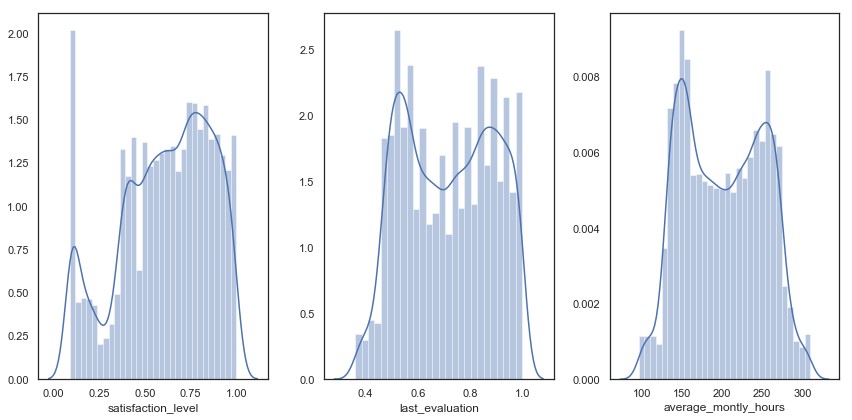
gcf

()

.

clear

()



<Figure size 432x288 with 0 Axes>

These distplots show something very interesting. It seems that there is two distributions mixed in satisfaction\_level, last\_evaluation and average\_montly\_hours data distributions.

**Is that corresponding to employees who stay and left ?**

In [11]:

fig

,

axs

=

plt

.

subplots

(

ncols

=

2

,

figsize

=

(

12

,

6

))

axs

[

0

]

.

hist

(

dataset

[

"number\_project"

]

,

bins

=

6

)

axs

[

0

]

.

set\_xlabel

(

"number\_project"

)

axs

[

0

]

.

set\_ylabel

(

"Count"

)

axs

[

1

]

.

hist

(

dataset

[

"time\_spend\_company"

]

,

bins

=

10

,

color

=

"r"

,

range

=

(

1

,

10

))

axs

[

1

]

.

set\_xlabel

(

"time\_spend\_company"

)

axs

[

1

]

.

set\_ylabel

(

"Count"

)

plt

.

tight\_layout

()

plt

.

show

()

plt

.

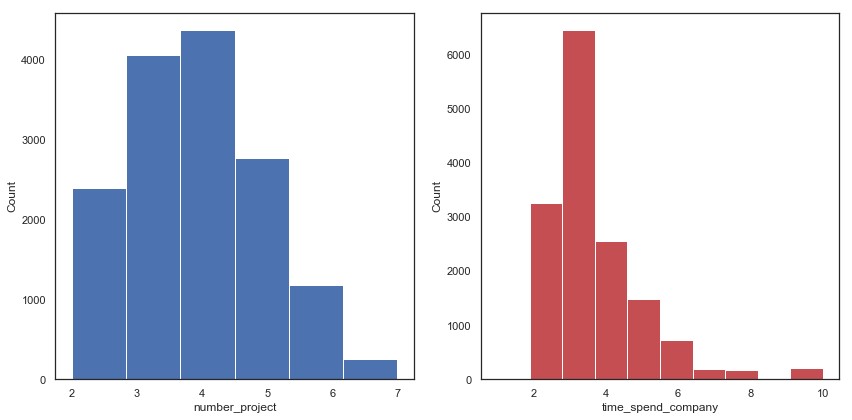
gcf

()

.

clear

()



<Figure size 432x288 with 0 Axes>

The number of projects and the time spend in company seem to follow an extrem value distribution (Gumbel distribution).

Time\_spend\_company is very positively skewed (right skewed).

In [12]:

g

=

sns

.

heatmap

(

dataset

.

corr

()

,

annot

=

**True**

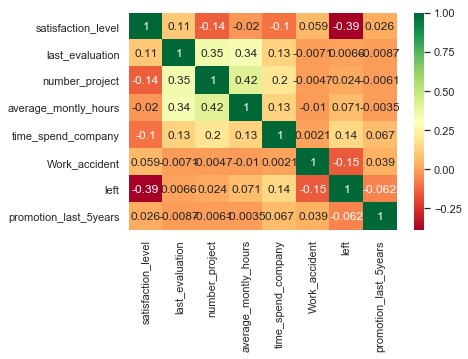
,

cmap

=

"RdYlGn"

)



It seems that employees working hard and with many projects have a better evaluation.

(corr(number\_project,last\_evaluation) : 0.35, corr(average\_montly\_hours,last\_evaluation) : 0.34 ).

The most important thing in this correlation matrix is the negative correlation between 'left' and 'satifaction\_level' (-0.39) : **employees leave because they are not happy at work ?**

**Is that the only main reason ?** **Is there employee patterns that can explained that ?**

To adress these questions, i performed a detailed analysis.

# 4. Detailed data exploration

Firslty, i will perform a dimensionality reduction in order to identify groups.

In [13]:

dataset

=

dataset

.

drop

(

labels

=

[

"sales"

]

,

axis

=

1

)

In [14]:

dataset["salary"] = dataset["salary"].astype("category",ordered=**True**, categories = ['low','medium','high']).cat.codes

## 4.1 Normalisation and dimensionalty reduction

In [15]:

*# pca/isomap analysis*

N

=

StandardScaler

()

N

.

fit

(

dataset

)

dataset\_norm

=

N

.

transform

(

dataset

)

Don't forget to normalize the data before the demensionality reduction.

In [16]: index = np.random.randint(0,dataset\_norm.shape[0],size=10000) In [17]:

dataset

.

head

()

Out[17]:

**satisfaction\_level last\_evaluation number\_project average\_montly\_hours time\_spend\_co**

1. 0.80 0.86 5 262
2. 0.11 0.88 7 272
3. 0.85 0.91 5 226
4. 0.11 0.93 7 308
5. 0.10 0.95 6 244

Because of the size of the dataset the isomap algorithm is very memory greedy. So i randomly choosed a 10 000 points in the dataset.

The isomap and pca maps are very similar to the ones obtained from the full dataset and are much faster to compute.

In [18]:

pca = PCA(n\_components=2) pca\_representation = pca.fit\_transform(dataset\_norm[index])

In [19]:

iso = Isomap(n\_components=2, n\_neighbors=40)

iso\_representation = iso.fit\_transform(dataset\_norm[index])

In [20]:

left\_colors = dataset["left"].map(**lambda** s : "g" **if** s==0 **else** "r") fig, axes = plt.subplots(1,2,figsize=(15,6))

axes[0].scatter(pca\_representation[:,0],pca\_representation[:,1],

c = left\_colors[index],alpha=0.5,s=20)

axes[0].set\_title("Dimensionality reduction with PCA") axes[0].legend(["Left employee"])

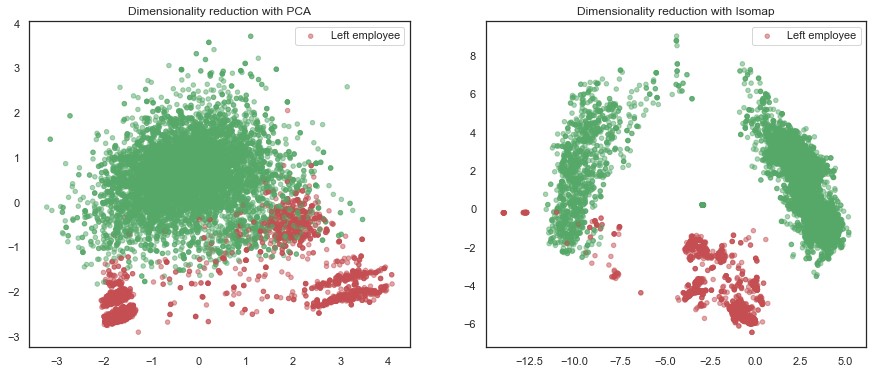
axes[1].scatter(iso\_representation[:,0],iso\_representation[:,1],

c = left\_colors[index],alpha=0.5,s=20)

axes[1].set\_title("Dimensionality reduction with Isomap") axes[1].legend(["Left employee"])

Out[20]:

<matplotlib.legend.Legend at 0x7f7e6c19fd30>



The red points correspond to employees who left. Here the PCA doesn't show a great separation between the left and stayed employees. PCA performs a linear demensionality reduction , the components produced by pca is a linear combinaison of the existing features. So it is very good when we have a linear relation between the points.

Here it seems that we need a non linear reduction like isomap does. We can see a great separation between the red and green points. An interesting fact is that we have two groups of employees who stayed (green points).

**Let's represent this with an interactive plot**

In [21]:

source\_dataset = ColumnDataSource(

data = dict(

x = iso\_representation[:2000,0], y = iso\_representation[:2000,1], desc = dataset.loc[index,"left"],

colors = ["#**%02x%02x%02x**" % (int(c\*255), int((1-c)\*255), 0) **for** c **in** dataset.loc[index,"left"]],

satisfaction\_level = dataset.loc[index,'satisfaction\_level'], last\_evaluation = dataset.loc[index,'last\_evaluation'], number\_project = dataset.loc[index,'number\_project'], time\_spend\_company = dataset.loc[index,'time\_spend\_company'], average\_montly\_hours = dataset.loc[index,'average\_montly\_hours']))

hover = HoverTool(tooltips=[("Left", "@desc"),

("Satisf. level", "@satisfaction\_level"),

("#projects", "@number\_project"),

("Last eval.", "@last\_evaluation"),

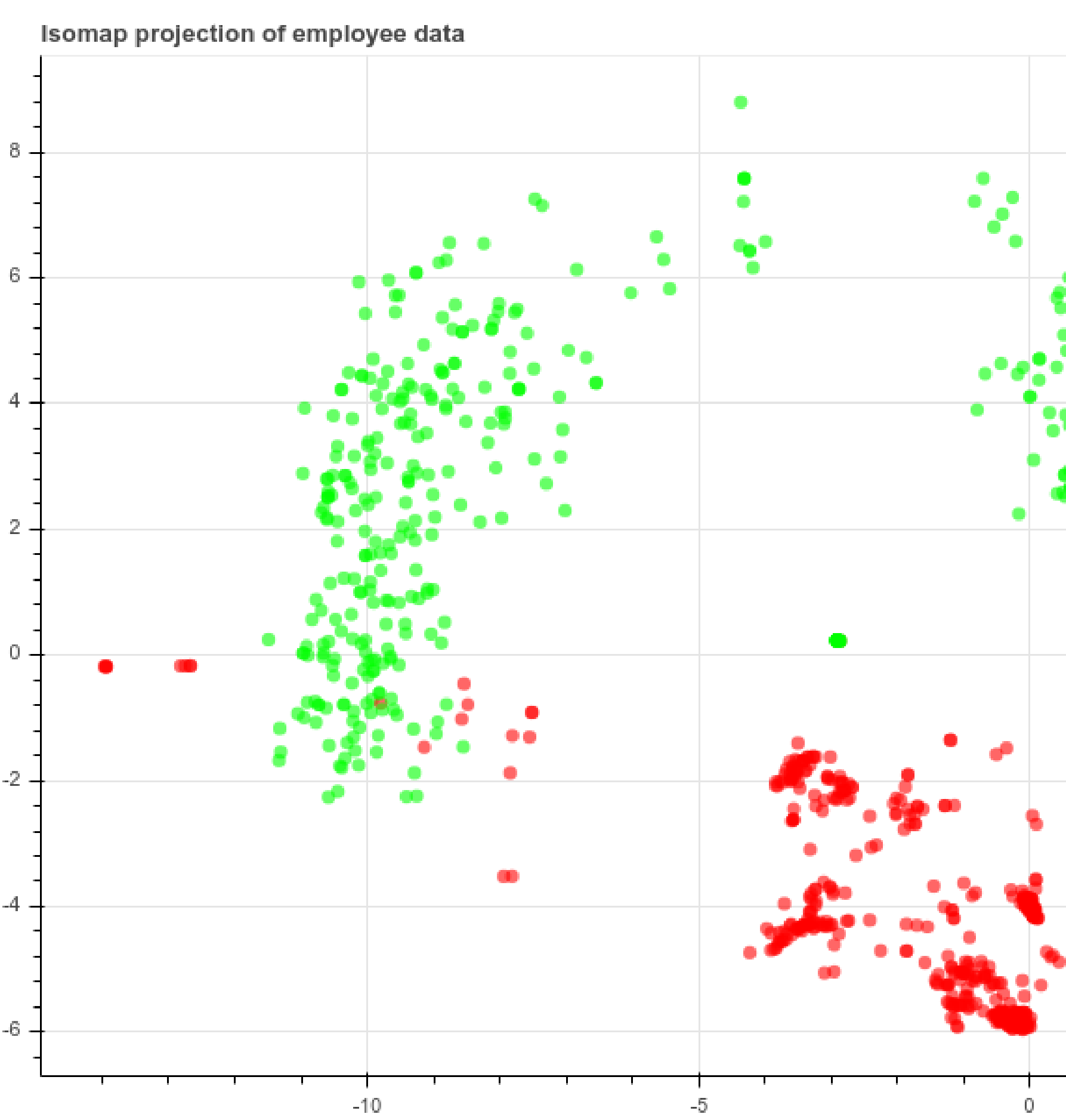
("Time in Company", "@time\_spend\_company"), ("Montly hrs", "@average\_montly\_hours")]) tools\_isomap = [hover, "box\_zoom",'pan', 'wheel\_zoom', 'reset']

plot\_isomap = figure(plot\_width= 800, plot\_height=600, tools=tools\_isomap,

title='Isomap projection of employee data')

plot\_isomap.scatter('x', 'y', size=7, fill\_color = "colors", line\_color = **None**,

fill\_alpha = 0.6, radius=0.1, alpha=0.5, line\_width=0, source=source\_dataset) show(plot\_isomap)



You can hover the data to see the major features.

**4.2 Global Radar Chart** In [22]:

data\_stay

=

dataset

[

dataset

[

"left"

]

==

0

]

data\_left

=

dataset

[

dataset

[

"left"

]

==

1

]

For practical reasons, i separate the left and stay data.

In [23]:

**def** \_scale\_data(data, ranges): (x1, x2) = ranges[0] d = data[0]

**return** [(d - y1) / (y2 - y1) \* (x2 - x1) + x1 **for** d, (y1, y2) **in** zip(data, r

anges)]

**class** **RadarChart**(): **def** \_\_init\_\_(self, fig, variables, ranges, n\_ordinate\_levels = 6): angles = np.arange(0, 360, 360./len(variables))

axes = [fig.add\_axes([0.1,0.1,0.8,0.8],polar = **True**, label = "axes**{}**".format(i)) **for** i **in** range(len(variables))] \_, text = axes[0].set\_thetagrids(angles, labels = variables)

**for** txt, angle **in** zip(text, angles):

txt.set\_rotation(angle - 90)

**for** ax **in** axes[1:]:

ax.patch.set\_visible(**False**) ax.xaxis.set\_visible(**False**) ax.grid("off")

**for** i, ax **in** enumerate(axes): grid = np.linspace(\*ranges[i],num = n\_ordinate\_levels) grid\_label = [""]+["**{:.1f}**".format(x) **for** x **in** grid[1:]] ax.set\_rgrids(grid, labels = grid\_label, angle = angles[i]) ax.set\_ylim(\*ranges[i])

self.angle = np.deg2rad(np.r\_[angles, angles[0]]) self.ranges = ranges self.ax = axes[0]

**def** plot(self, data, \*args, \*\*kw):

sdata = \_scale\_data(data, self.ranges)

self.ax.plot(self.angle, np.r\_[sdata, sdata[0]], \*args, \*\*kw)

**def** fill(self, data, \*args, \*\*kw):

sdata = \_scale\_data(data, self.ranges)

self.ax.fill(self.angle, np.r\_[sdata, sdata[0]], \*args, \*\*kw)

**def** legend(self, \*args, \*\*kw):

self.ax.legend(\*args, \*\*kw)

attributes = ['satisfaction\_level','last\_evaluation','number\_project', 'average\_montly\_hours','time\_spend\_company']

data\_stay\_mean = data\_stay[attributes].mean().values.reshape(1,-1) data\_left\_mean = data\_left[attributes].mean().values.reshape(1,-1) datas = np.concatenate((data\_stay\_mean,data\_left\_mean),axis = 0)

ranges = [[1e-2, dataset[attr].max()] **for** attr **in** attributes] colors = ["green","red"] left\_types = ["Stayed","Left"]

fig

=

plt

.

figure

(

figsize

=

(

8

,

8

))

radar

=

RadarChart

(

fig

,

attributes

,

ranges

)

**for**

data

,

color

,

left\_type

**in**

zip

(

datas

,

colors

,

left\_types

):

radar

.

plot

(

data

,

color

=

color

,

label

=

left\_type

,

linewidth

=

2.0

)

radar

.

fill

(

data

,

alpha

=

0.2

,

color

=

color

)

radar

.

legend

(

loc

=

1

,

fontsize

=

'medium'

)

plt

.

title

(

'Stats of employees who stayed and left'

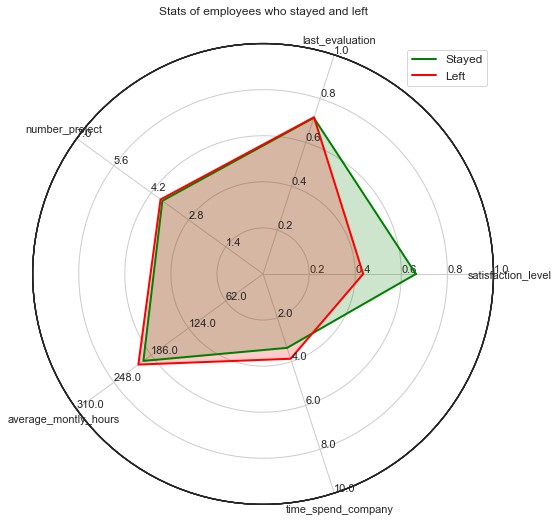
)

plt

.

show

()



This radar chart doesn't show so many diffrences between left and stayed employees. At first glance the main difference seems to be the satisfaction level.

As we demonstrate above, employees who left are less happy than the others.

However this radar chart is build on the mean of each feature, so it could hide some sub-distributions in the data.

Let's investigate this in further analysis.

## 4.3 Left and other features

In [24]:

fig

,

axs

=

plt

.

subplots

(

nrows

=

1

,

ncols

=

2

,

figsize

=

(

10

,

6

))

sns

.

factorplot

(

y

=

"satisfaction\_level"

,

x

=

"left"

,

data

=

dataset

,

kind

=

"box"

,

ax

=

axs

[

0

])

axs

[

1

]

.

hist

(

data\_stay

[

"satisfaction\_level"

]

,

bins

=

6

,

label

=

"Stay"

,

alpha

=

0.7

)

axs

[

1

]

.

hist

(

data\_left

[

"satisfaction\_level"

]

,

bins

=

6

,

label

=

"Left"

,

alpha

=

0.7

)

axs

[

1

]

.

set\_xlabel

(

"Satifaction level"

)

axs

[

1

]

.

set\_ylabel

(

"Count"

)

axs

[

1

]

.

legend

()

plt

.

tight\_layout

()

plt

.

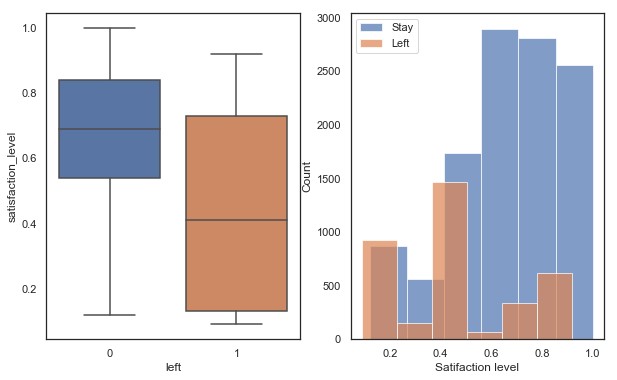
gcf

()

.

clear

()



<Figure size 360x360 with 0 Axes>

The satisfaction level is the most correlated feature with 'left'. Here we can see that employees who left have a lower satisfaction level that those who stayed.

We can also noticed the three sub-distributions of satisfaction levels with employees who left. Is that corresponding to 3 groups ?

One with a low satisfaction level

One with a medium satisfaction level

One with a high satisfaction level

In [25]:

salary\_counts

=

(

dataset

.

groupby

([

'left'

])[

'salary'

]

.

value\_counts

(

normalize

=

**True**

)

.

rename

(

'percentage'

)

.

mul

(

100

)

.

reset\_index

())

p

=

sns

.

barplot

(

x

=

"salary"

,

y

=

"percentage"

,

hue

=

"left"

,

data

=

salary\_counts

)

p

.

set\_ylabel

(

"Percentage"

)

p

=

p

.

set\_xticklabels

([

"Low"

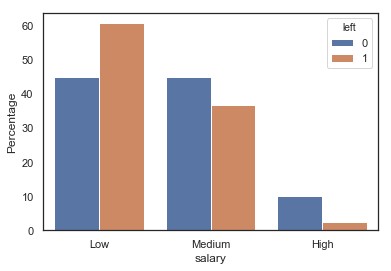
,

"Medium"

,

"High"

])



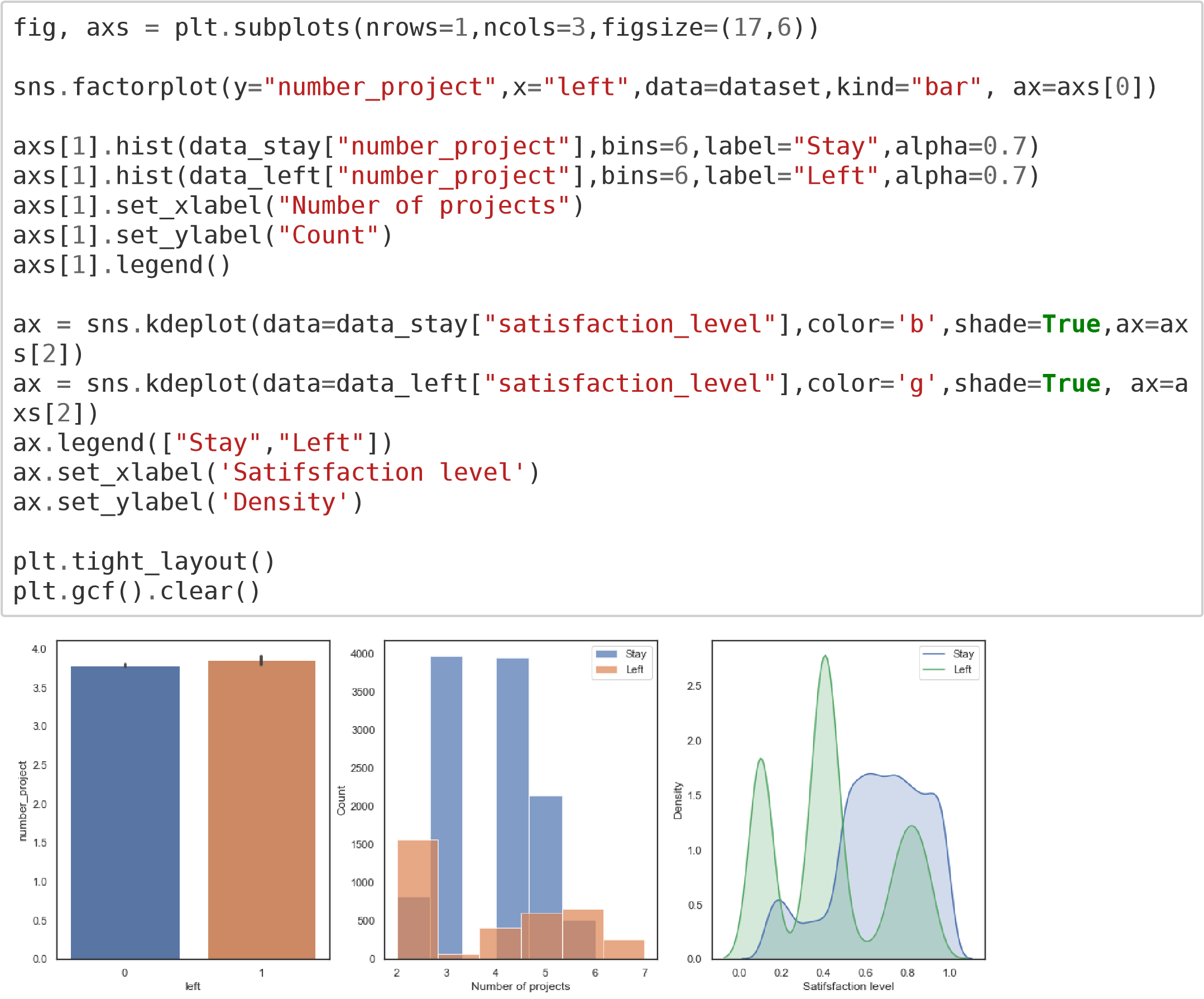
Let's invistigate for the salary of employees who left/stayed.

Here i show the percentage of the employees with a low/medium/high salary in the two categories.

Employees who left have a lower salary than other.

**Is that the reason why employees left ?**

In [26]:



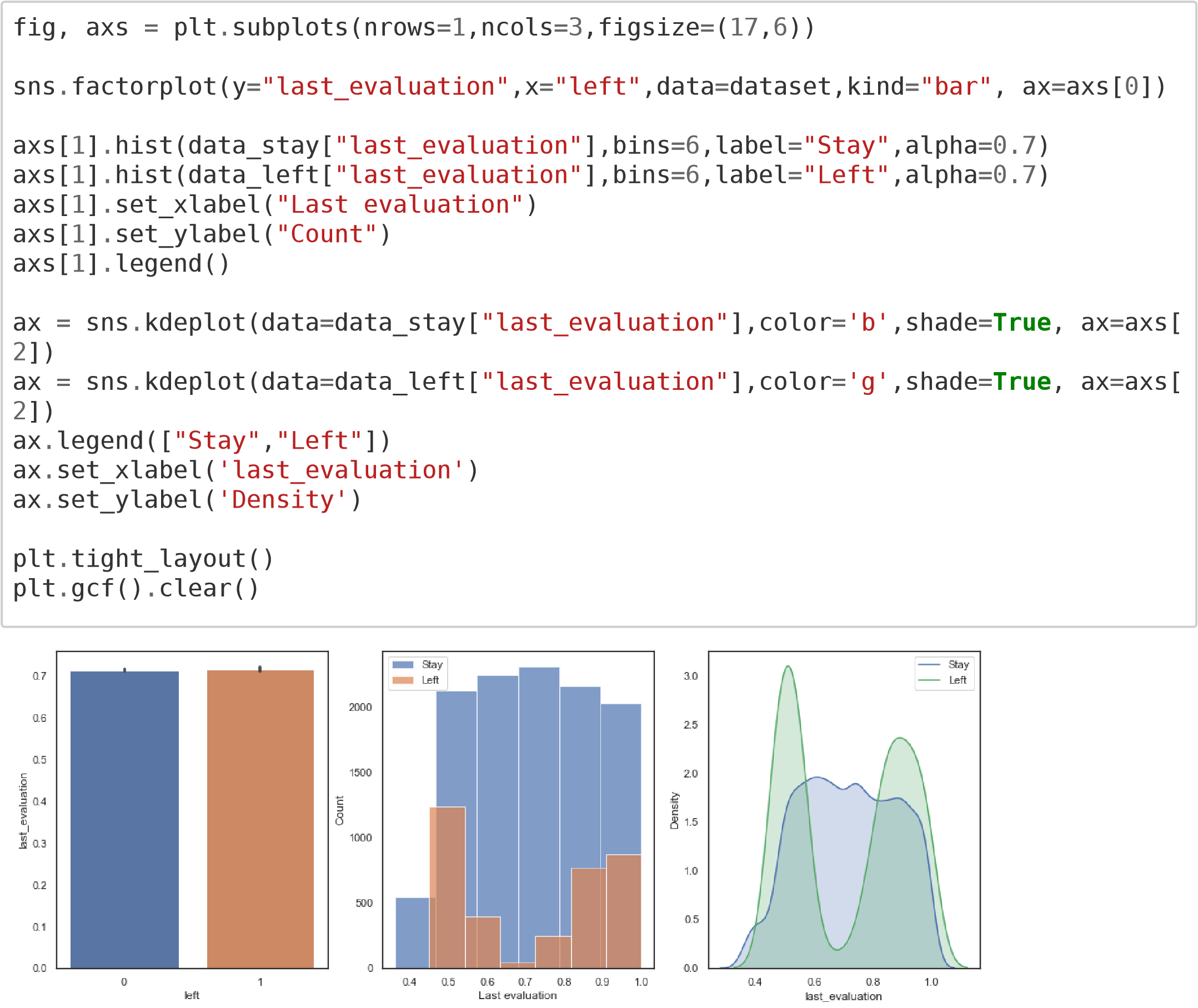
<Figure size 360x360 with 0 Axes>

Let's see now if they have more work than the others.

Left and stayed employees have a similar number of projects.

However, when we look in detail, there is two sub population in the employees who left. Those who have few projects and those who have a lot of projects.

In [27]:



<Figure size 360x360 with 0 Axes>

When we look at the last evaluation we still have two sub populations of left employees.

Those with a medium score and those with an high score, that's very interesting !

In [28]:

fig

,

axs

=

plt

.

subplots

(

nrows

=

1

,

ncols

=

2

,

figsize

=

(

10

,

6

))

sns

.

factorplot

(

y

=

"average\_montly\_hours"

,

x

=

"left"

,

data

=

dataset

,

kind

=

"box"

,

ax

=

axs

[

0

])

axs

[

1

]

.

hist

(

data\_stay

[

"average\_montly\_hours"

]

,

bins

=

6

,

label

=

"Stay"

,

alpha

=

0.7

)

axs

[

1

]

.

hist

(

data\_left

[

"average\_montly\_hours"

]

,

bins

=

6

,

label

=

"Left"

,

alpha

=

0.7

)

axs

[

1

]

.

set\_xlabel

(

"Average Montly Hours"

)

axs

[

1

]

.

set\_ylabel

(

"Count"

)

axs

[

1

]

.

legend

()

plt

.

tight\_layout

()

plt

.

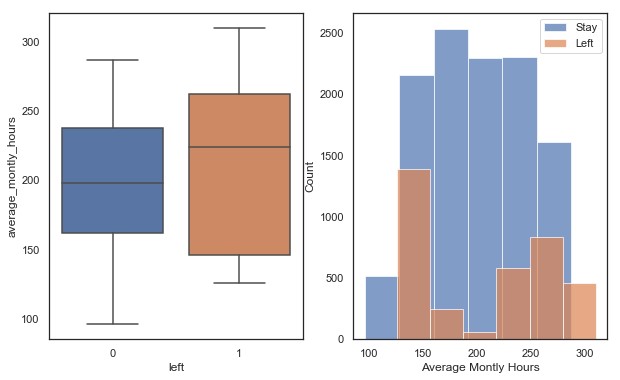
gcf

()

.

clear

()



<Figure size 360x360 with 0 Axes>

Similarly to the evaluation score and the number of projects. There is two sub populations of employees who left. Those who work less and those who work a lot.

Since the evaluation score, the number of projects and the average montly hours are correlated each other, we can make the hypothesis that there is two groups of employee who leaves. Those who work less because they have gets lower scores

In [29]:

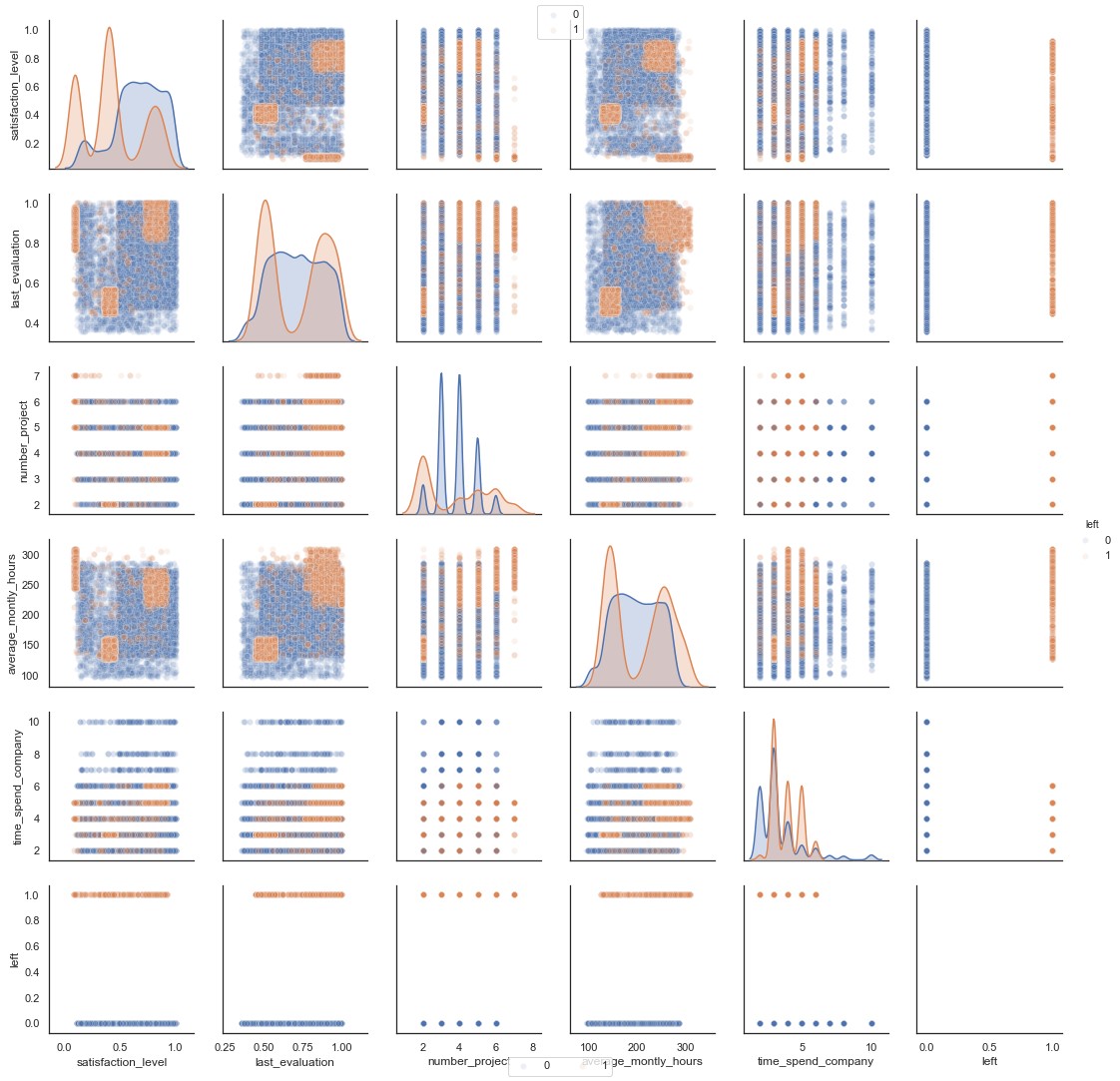
g = sns.pairplot(dataset.drop(labels=['promotion\_last\_5years','Work\_accident','s alary'],axis=1),hue="left",plot\_kws=dict(alpha=0.1)) handles = g.\_legend\_data.values() labels = g.\_legend\_data.keys()

g.fig.legend(handles=handles, labels=labels, loc='upper center', ncol=1)

g.fig.legend(handles=handles, labels=labels, loc='lower center', ncol=3)

Out[29]:

<matplotlib.legend.Legend at 0x7f7e723851d0>



The pairplot shows very interesting patterns when we plot the average montly hours against the satisfaction level or the satifaction level against the evaluation score.

It's like we still have 2/3 kind of employees who left.

Let's analyse these groups in detail.

In [30]:

*# Deeper in the analysis*

g = sns.FacetGrid(dataset, col="left", hue="left",size=5,aspect=1.2)

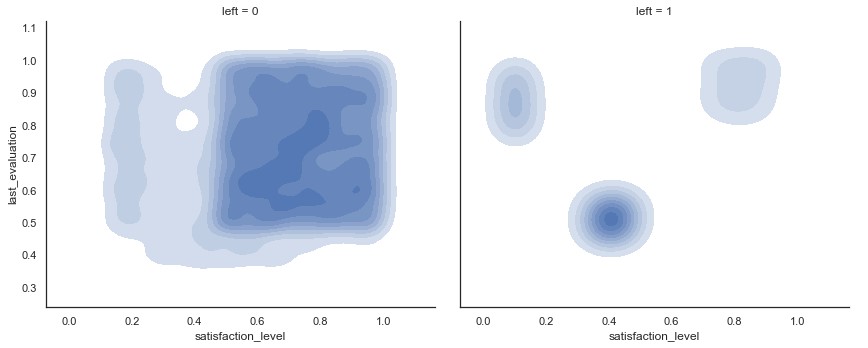
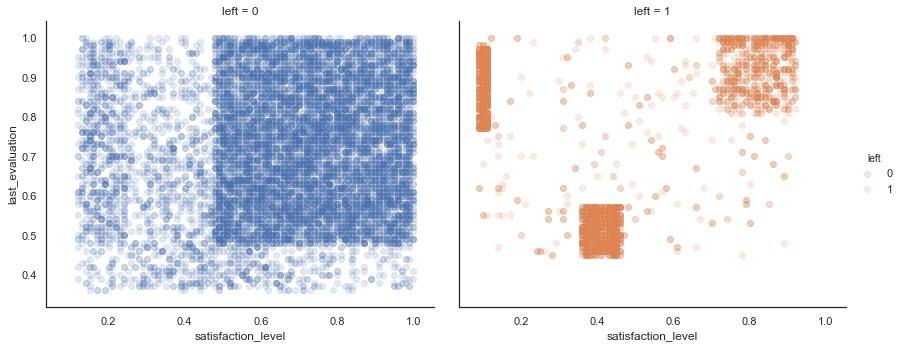
g.map(plt.scatter, "satisfaction\_level", "last\_evaluation",alpha=0.15) g.add\_legend()

g = sns.FacetGrid(dataset, col="left",size=5,aspect=1.2)

g.map(sns.kdeplot, "satisfaction\_level", "last\_evaluation",shade=**True**,shade\_lowe st=**False**)

g.add\_legend()

Out[30]: <seaborn.axisgrid.FacetGrid at 0x7f7e655ed668>



We have three groups of employees who left.

* Successfull but unhappy employees (top left)
* Successfull and happy employees (top right)
* Unsuccessfull and unhappy employees (bottom center)

Now we want to label the data with this tree groups.

## 4.4 Clustering analysis

In [31]:

*# Lets compare inside the 3 identified groups*

kmeans = KMeans(n\_clusters=3,random\_state=2) kmeans.fit(data\_left[["satisfaction\_level","last\_evaluation"]]) Out[31]:

KMeans(algorithm='auto', copy\_x=True, init='k-means++', max\_iter=30

0, n\_clusters=3, n\_init=10, n\_jobs=None, precompute\_distances='aut

o', random\_state=2, tol=0.0001, verbose=0)

I performed a kmean clustering to isolate these three groups.

In [32]:

kmeans\_colors

=

[

'red'

**if**

c

==

0

**else**

'orange'

**if**

c

==

2

**else**

'blue'

**for**

c

**in**

km

eans

.

labels\_

]

fig

=

plt

.

figure

(

figsize

=

(

10

,

7

))

plt

.

scatter

(

x

=

"satisfaction\_level"

,

y

=

"last\_evaluation"

,

data

=

data\_left

,

alpha

=

0.25

,

color

=

kmeans\_colors

)

plt

.

xlabel

(

"Satisfaction level"

)

plt

.

ylabel

(

"Last evaluation"

)

plt

.

scatter

(

x

=

kmeans

.

cluster\_centers\_

,

[:

0

]

,

y

=

kmeans

.

cluster\_centers\_

[:

,

1

]

,

color

=

"black"

,

marker

=

"X"

,

s

=

100

)

plt

.

title

(

"Clustering of the employed who left by Kmeans"

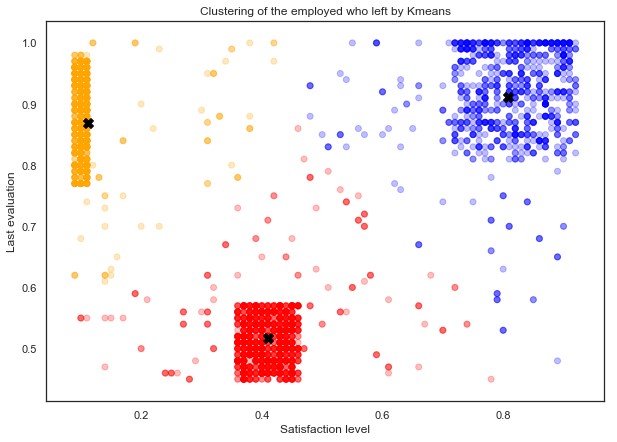
)

plt

.

show

()



The cluster center are the three black crosses.

We have three groups which are defined as :

* SuccessUnhappy (yellow) == Successfull but unhappy employees (top left)
* SuccessHappy (red) == Successfull and happy employees (top right)
* UnsuccessUnhappy (blue) == Unsuccessfull and unhappy employees (bottomcenter)

In [33]:

data\_left\_SuccessHappy = data\_left[kmeans.labels\_ == 0] data\_left\_UnsuccessUnhappy = data\_left[kmeans.labels\_ == 1] data\_left\_SuccessUnhappy = data\_left[kmeans.labels\_ == 2] In [34]:

data\_left\_SuccessUnhappy

.

shape

Out[34]:

(944, 9) In [35]:

data\_left\_SuccessHappy

.

shape

Out[35]:

(1650, 9) In [36]:

data\_left\_UnsuccessUnhappy

.

shape

Out[36]:

(977, 9)

The three groups have sufficient data to consider significant differences.

To demonstrate a significant diffrence we should perform statistical analysis . Such as comparison of means (Student test, Z-test or wilcoxon test if you don't want to make any assumption on the data distribution)

In [37]:

data\_left\_SuccessUnhappy\_mean = data\_left\_SuccessUnhappy[attributes].mean().valu es.reshape(1,-1)

data\_left\_SuccessHappy\_mean = data\_left\_SuccessHappy[attributes].mean().values.r eshape(1,-1) data\_left\_UnsuccessUnhappy\_mean = data\_left\_UnsuccessUnhappy[attributes].mean(). values.reshape(1,-1)

datas = np.concatenate((data\_stay\_mean,data\_left\_SuccessUnhappy\_mean,

data\_left\_SuccessHappy\_mean,data\_left\_UnsuccessUnhappy\_m ean),axis = 0)

In [38]:

colors

=

[

"green"

,

"red"

,

"blue"

,

"orange"

]

left\_types

=

[

"Stayed"

,

"Left Success-Unhappy"

,

"Left Success-Happy"

,

"Left Unsuc

ess-Unhappy"

]

fig

=

plt

.

figure

(

figsize

=

(

10

,

10

))

radar

=

RadarChart

(

fig

,

attributes

,

ranges

)

**for**

data

,

color

,

left\_type

**in**

zip

(

datas

,

colors

,

left\_types

):

radar

.

plot

(

data

,

color

=

color

,

label

=

left\_type

,

linewidth

=

2.0

)

radar

.

fill

(

data

,

alpha

=

0.1

,

color

=

color

)

radar

.

legend

(

loc

=

1

,

fontsize

=

'small'

)

plt

.

title

(

'Stats of employees who stayed and left'

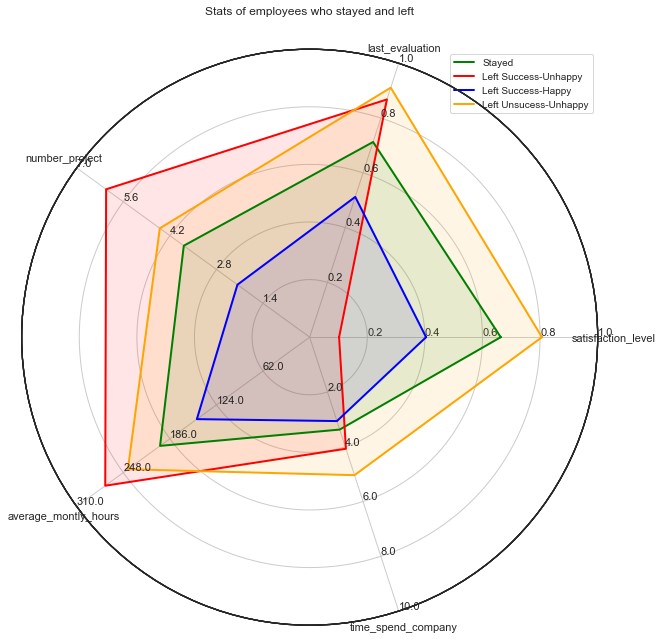
)

plt

.

show

()



When we compare the 3 groups and the group of employees who stayed we better understand the groups.

It appears that the SuccessUnhappy group of employees are those who work the most with 6 projects and more than 300 h/month. These employees left because they have too much work !

The UnsuccessUnhappy group left because they are not really involved in their company. They have few projects and work less than the other.

The last group (SuccessHappy group) is closest to the stay group except that they spent a long time in the company.

# 5. Modeling

I wanted to build a model that predicts the target variable with a good accuracy and explore the features weights and importance.

In [47]:

*## Prediction of the target variable (stay/left)*

X\_train

=

dataset

.

drop

(

labels

=

"left"

,

axis

=

1

)

Y\_train

=

dataset

[

"left"

]

train\_features

=

X\_train

.

columns

In [48]: kfold = StratifiedKFold(n\_splits = 10, random\_state = 2)

In [49]:

*#kfold = StratifiedKFold(Y\_train,n\_splits=10,random\_state=2)*

## 5.1 Decision Tree

I will consider the Decision tree algorithm which is perfect for studying the importance of features in data modeling.

In [50]:

DTC

=

DecisionTreeClassifier

(

max\_depth

=

3

)

cv\_results

=

cross\_val\_score

(

DTC

,

X\_train

,

Y\_train

,

cv

=

kfold

,

scoring

=

"accuracy"

)

cv\_results

.

mean

()

Out[50]:

0.9523944374790092

I have restricted the depth tree to 3, to build a simple tree for analysis.

Despite the simplicity of this tree, it gives us 95% of accuracy which is very good and enough.

In [51]:

DTC

.

fit

(

X\_train

,

Y\_train

)

Out[51]:

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_dep th=3,

max\_features=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_sta

te=None, splitter='best')

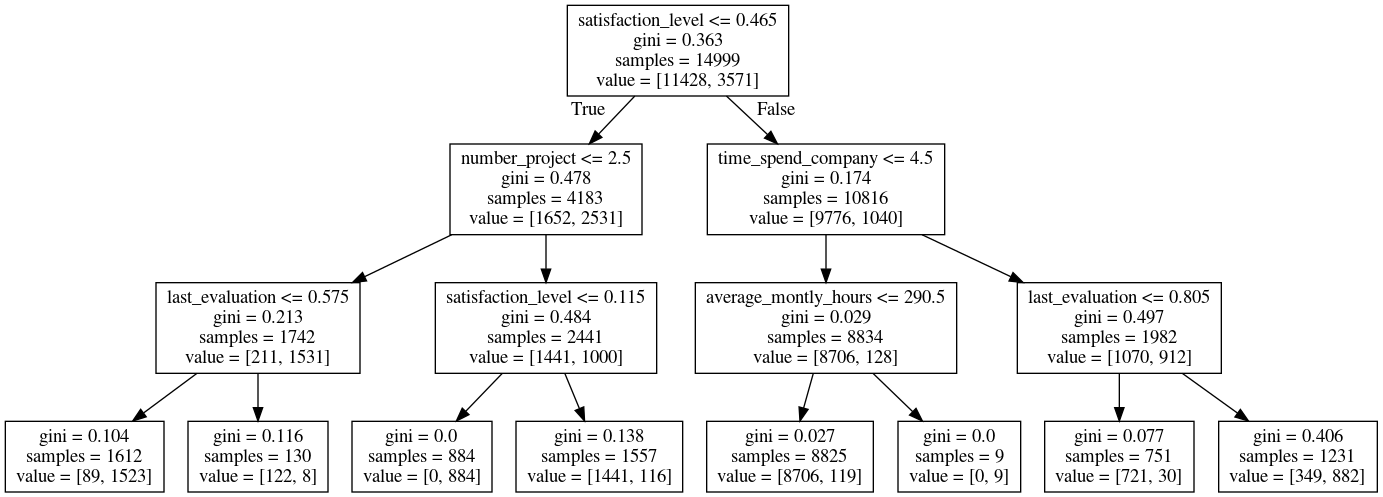
I plot the tree structure to look at the most important features.

In [55]:

dot\_data = tree.export\_graphviz(DTC, out\_file = **None**, feature\_names = train\_feat ures)

graph = pydotplus.graph\_from\_dot\_data(dot\_data) Image(graph.create\_png())

Out[55]:



In the Decision tree structure, the most important variable (for the prediction) is at the top.

So the satisfaction level is the most important feature.

What is very interesting is that we can found the 4 groups (SuccessHappy, UnsuccessUnhappy, SuccessUnhappy, stay) and their caracteristics .

**The left leaf (satisf.level <=0.465 [True] -> number\_project <= 2.5 [True] -> last\_evaluation <= 0.57 [True]) corresponds to the UnsuccessUnhappy group.**

**The path satisf.level <=0.465 [True] -> number\_project <= 2.5 [False] -> last\_evaluation <= 0.115 [True] corresponds to the SuccessUnhappy group.**

**The right leaf (satisf.level <=0.465 [False] -> time\_spend\_company <= 4.5 [False] -> last\_evaluation <= 0.805 [False])corresponds to the SuccessHappy group.**

**The path (satisf.level <=0.465 [False] -> time\_spend\_company <= 4.5 [True] ->**

**average\_monty\_hours <= 290.5 [True]) contains to the most important part of the stay group (8706/11428 = 76%).**

With this decision tree we can clearly explain why the employee left.

## 5.2 Random Forest

I made a Random Forest classifier for those who like the performances :) (99% of accuracy)

In [56]:

*# For those who like performance, 99% accuracy*

RFC

=

RandomForestClassifier

()

cv\_results

=

cross\_val\_score

(

RFC

,

X\_train

,

Y\_train

,

cv

=

kfold

,

scoring

=

"accuracy"

)

cv\_results

.

mean

()

Out[56]:

0.9899324878514761

In [57]:

RFC

.

fit

(

X\_train

,

Y\_train

)

Out[57]:

RandomForestClassifier(bootstrap=True, class\_weight=None, criterion

='gini', max\_depth=None, max\_features='auto', max\_leaf\_nodes=Non

e,

min\_impurity\_decrease=0.0, min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=10, n\_jobs=N

one,

oob\_score=False, random\_state=None, verbose=0, warm\_start=False)

In [58]:

indices

=

np

.

argsort

(

RFC

.

feature\_importances\_

)[::

-

1

][:

40

]

fig

=

plt

.

figure

(

figsize

=

(

7

,

10

))

g

=

sns

.

barplot

(

y

=

train\_features

[

indices

][:

40

]

,

x

=

RFC

.

feature\_importances\_

[

indi

ces

][:

40

]

,

orient

=

'h'

)

g

.

set\_xlabel

(

"Relative importance"

,

fontsize

=

12

)

g

.

set\_ylabel

(

"Features"

,

fontsize

=

12

)

g

.

tick\_params

(

labelsize

=

9

)

g

.

set\_title

(

"Random Forest classifier feature importance"

)

plt

.

show

()

plt

.

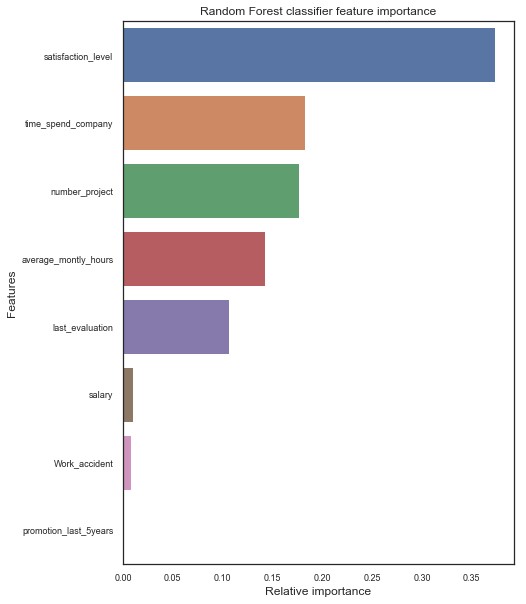
gcf

()

.

clear

()



<Figure size 432x288 with 0 Axes> It is now clear that satisfaction\_level average\_montly\_hours, number\_project, time\_spend\_company and last\_evaluation are the 5 most important features that explain why employees are leaving or staying in the company. Not the salary not the promotions.