# Report

Your goal is to be critical of the report that was given to you by another team member. You need to consider ways in which this analysis could be improved and provide your own interpretation of the situation before this report is shared with company leadership. The questions listed in each heading are suggestions. You do not need to answer each one, and your report can explore other questions not listed here.

* **Business Understanding**

### Abstract

In the last decades, having the best machines was enough to be competitive or to dominate an industrial sector. Nowadays, the company that has more engaged and productive employees will have a better chance of winning market competition. For this reason, companies can not lose important employees and when that begins to happen you need to understand why, to prevent this from happening. The Human Resorces Analytics dataset, is used to explain the first steps in the data analysis path. In this first part is presented how to get familiarize itself with the data set by performing the descriptive analysis. Techniques such as exploratory data analysis (EDA) allow us to present the data in a more meaningful way, applying general statistical methods and exploratory graphics, that allow a simpler interpretation before engage a machine learning algorithm.

* **Data Understanding.** 
  1. Is the data appropriate? What don’t we know from the data that would be helpful when understanding the results? What data should they have included that was missing?

○ How effective are the visualizations at building the narrative of the report? How could they be improved? What visualizations are missing that could help?

## 2. Exploratory Data Analysis (EDA)[¶](http://localhost:8888/notebooks/human-resources-analytics-a-descriptive-analysis.ipynb#2.-Exploratory-Data-Analysis-(EDA))

Exploratory data analysis employs a variety of techniques (mostly statistical graphics) before making inferences from data. It is essencial to examine all variables in the dataset to:

* Catch mistakes
* Generate hypotheses
* See patterns in the data
* Extract important variables
* Detect outliers and anomalies
* Gain deep familiarity with the dataset
* Refine selection of features that will be used to build the machine learning models.

Special attention to not skip the EDA process, because can generate inaccurate models or accurate models on the wrong data. This dataset contains 14999 objects and 10 attributes described below:

Variables | Descriptions

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

satisfaction\_level | Satisfaction Level

last\_evaluation | Last evaluation

number\_project | Number of projects

average\_montly\_hours | Average monthly hours

time\_spend\_company | Time spent at the company

Work\_accident | Whether they have had a work accident

left | Whether the employee has left

promotion\_last\_5years | Whether had a promotion in the last 5 years

sales | Departments (column sales)

salary | Salary

## 3. Preprossesing the dataset

Before starting the process, its important to answer if it's clear what kind of problem we are dealing with, because in many cases isn't so simple to identy it. A good understanding of the problem will help to choose the right data mining and machine learning techniques to make the right predictions.

Thus, the first step, is preprocessing the data to look for missing, incomplete or noise values, because, in real word, the raw datas can be collect from many sources like sensors, websites, public data and many others.

o start the step of preprossing the dataset is neccessary to import some useful Python libraries.

* Numpy: Is a fundamental package to use linear algebra and random number capabilities. See: [www.numpy.org/](http://www.numpy.org/)
* Pandas: Is a package to work with relacional data as tables. See: pandas.pydata.org/
* **Data Preparation** 
  1. Did the report appropriately explore all the different ways in which the data may

be corrupted? What were additional cleaning steps they could have considered? Should they have reshaped the data in any way? Do you trust the data? What would make you trust or distrust the data?

* **Modeling** 
  1. Were the analytics choices here appropriate? Did they apply them correctly? You don’t need to know the specifics of the code, but more about the general approach (e.g. was a decision tree a good choice, or is there another analysis that would have been better used?)

○ How were the models evaluated? How do you know that they fit the data appropriately? What approaches did they use to avoid overfitting? Do we know if these models will work on unknown data in the future? What metrics could they have used to assess the quality of the model?

○ What were the important variables in the models and how do you know they are important? Do we know how these variable impact the outcome? How could they have measured that impact?

* **Evaluation**
* **Deployment**

# Human Resources Analytics: A Descriptive Analysis

William Walter da Silva

### Abstract

In the last decades, having the best machines was enough to be competitive or to dominate an industrial sector. Nowadays, the company that has more engaged and productive employees will have a better chance of winning market competition. For this reason, companies can not lose important employees and when that begins to happen you need to understand why, to prevent this from happening. The Human Resorces Analytics dataset, is used to explain the first steps in the data analysis path. In this first part is presented how to get familiarize itself with the data set by performing the descriptive analysis. Techniques such as exploratory data analysis (EDA) allow us to present the data in a more meaningful way, applying general statistical methods and exploratory graphics, that allow a simpler interpretation before engage a machine learning algorithm.

## 1. The Human Resources Dataset

The Human Resources Analytics is a simulated dataset from [Kaggle](https://www.kaggle.com/ludobenistant/hr-analytics) and the focus is to understand why the best and most experienced employees is leaving the company. By the exploration of this dataset its possible to extract good insights of a problems that the Human Resource department deals daily. In many industries retain their best employees its a question of long term strategy, and can impact the companies growth or put in financial risk, mainly if the employees leave to work at the competitor.

## 2. Exploratory Data Analysis (EDA)

Exploratory data analysis employs a variety of techniques (mostly statistical graphics) before making inferences from data. It is essencial to examine all variables in the dataset to:

* Catch mistakes
* Generate hypotheses
* See patterns in the data
* Extract important variables
* Detect outliers and anomalies
* Gain deep familiarity with the dataset
* Refine selection of features that will be used to build the machine learning models.

Special attention to not skip the EDA process, because can generate inaccurate models or accurate models on the wrong data. This dataset contains 14999 objects and 10 attributes described below:

Variables | Descriptions

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

satisfaction\_level | Satisfaction Level

last\_evaluation | Last evaluation

number\_project | Number of projects

average\_montly\_hours | Average monthly hours

time\_spend\_company | Time spent at the company

Work\_accident | Whether they have had a work accident

left | Whether the employee has left

promotion\_last\_5years | Whether had a promotion in the last 5 years

sales | Departments (column sales)

salary | Salary

## 3. Preprossesing the dataset

Before starting the process, its important to answer if it's clear what kind of problem we are dealing with, because in many cases isn't so simple to identy it. A good understanding of the problem will help to choose the right data mining and machine learning techniques to make the right predictions. Thus, the first step, is preprocessing the data to look for missing, incomplete or noise values, because, in real word, the raw datas can be collect from many sources like sensors, websites, public data and many others.

To start the step of preprossing the dataset is neccessary to import some useful Python libraries.

* Numpy: Is a fundamental package to use linear algebra and random number capabilities. See: www.numpy.org/
* Pandas: Is a package to work with relacional data as tables. See: pandas.pydata.org/

In [1]:

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

#### Load the data

To load the dataset we use a Pandas method called **read\_csv** that read CSV(comma-separated) files and covert into DataFrame.

In [2]:

data = pd.read\_csv('../input/HR\_comma\_sep.csv')

Other useful method is **info** that shows a summary of the dataset, like number of observations, columns, variable type and the total memory usage. The dataset have 14999 observations, 10 columns and with no null values. The data types of the variables are divided in 2 float, 6 integer and 2 object.

In [3]:

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 14999 entries, 0 to 14998

Data columns (total 10 columns):

satisfaction\_level 14999 non-null float64

last\_evaluation 14999 non-null float64

number\_project 14999 non-null int64

average\_montly\_hours 14999 non-null int64

time\_spend\_company 14999 non-null int64

Work\_accident 14999 non-null int64

left 14999 non-null int64

promotion\_last\_5years 14999 non-null int64

sales 14999 non-null object

salary 14999 non-null object

dtypes: float64(2), int64(6), object(2)

memory usage: 1.1+ MB

Let's see the first 5 lines of the dataset. The **head** method list first N rows from the DataFrame and the method **tail**, returns the last N rows.

In [4]:

data.head(5)

Out[4]:

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **left** | **promotion\_last\_5years** | **sales** | **salary** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.38 | 0.53 | 2 | 157 | 3 | 0 | 1 | 0 | sales | low |
| **1** | 0.80 | 0.86 | 5 | 262 | 6 | 0 | 1 | 0 | sales | medium |
| **2** | 0.11 | 0.88 | 7 | 272 | 4 | 0 | 1 | 0 | sales | medium |
| **3** | 0.72 | 0.87 | 5 | 223 | 5 | 0 | 1 | 0 | sales | low |
| **4** | 0.37 | 0.52 | 2 | 159 | 3 | 0 | 1 | 0 | sales | low |

In [5]:

data.tail(5)

Out[5]:

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **left** | **promotion\_last\_5years** | **sales** | **salary** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **14994** | 0.40 | 0.57 | 2 | 151 | 3 | 0 | 1 | 0 | support | low |
| **14995** | 0.37 | 0.48 | 2 | 160 | 3 | 0 | 1 | 0 | support | low |
| **14996** | 0.37 | 0.53 | 2 | 143 | 3 | 0 | 1 | 0 | support | low |
| **14997** | 0.11 | 0.96 | 6 | 280 | 4 | 0 | 1 | 0 | support | low |
| **14998** | 0.37 | 0.52 | 2 | 158 | 3 | 0 | 1 | 0 | support | low |

**sample** is a easy way to get a few data quickly.

In [6]:

data.sample(10)

Out[6]:

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **left** | **promotion\_last\_5years** | **sales** | **salary** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6047** | 0.98 | 0.55 | 3 | 260 | 3 | 0 | 0 | 0 | sales | medium |
| **9153** | 0.41 | 0.45 | 3 | 236 | 2 | 0 | 0 | 0 | RandD | medium |
| **2870** | 0.75 | 0.79 | 4 | 145 | 3 | 0 | 0 | 0 | technical | medium |
| **3142** | 0.81 | 0.80 | 4 | 229 | 2 | 0 | 0 | 0 | sales | medium |
| **14786** | 0.39 | 0.49 | 2 | 129 | 3 | 0 | 1 | 0 | technical | medium |
| **11860** | 0.78 | 0.90 | 5 | 158 | 3 | 0 | 0 | 0 | sales | low |
| **13773** | 0.76 | 0.79 | 3 | 247 | 3 | 0 | 0 | 0 | sales | low |
| **3454** | 0.89 | 0.86 | 3 | 178 | 4 | 0 | 0 | 0 | sales | medium |
| **10941** | 0.81 | 0.92 | 5 | 258 | 3 | 0 | 0 | 1 | sales | medium |
| **12274** | 0.11 | 0.90 | 6 | 254 | 4 | 0 | 1 | 0 | technical | low |

#### Variables transformations

To plot some statistical graphics and for better understanding, we make some transformations in the variables:

* sales: Rename to department
* salary: Convert the type of the variable from categorical to numerical.

In [7]:

*# RENAME column sale to department*

data.rename(columns={'sales': 'department'}, inplace = **True**)

*# Convert salary variable type to numeric*

data['salary'] = data['salary'].map({'low':1, 'medium':2, 'high':3})

## 4. Descripitve Analysis

The descripitve Analysis is used to simplify and summarize the mainly characteristics of the dataset. In other words, show what kind of information the dataset has. The Pandas method **describe** generates a descriptive statistics that summarize the central tendency, dispersion and shape of the dataset. By using this method in Human Resource dataset important insights is possible to see:

* That approximately 24% os the employees left the company.
* The satisfaction level is around 62% and performance is around 72%.
* Employees work in average on 4 projects with 200 hours worked per month.

In [8]:

data.describe()

Out[8]:

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **left** | **promotion\_last\_5years** | **salary** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 |
| **mean** | 0.612834 | 0.716102 | 3.803054 | 201.050337 | 3.498233 | 0.144610 | 0.238083 | 0.021268 | 1.594706 |
| **std** | 0.248631 | 0.171169 | 1.232592 | 49.943099 | 1.460136 | 0.351719 | 0.425924 | 0.144281 | 0.637183 |
| **min** | 0.090000 | 0.360000 | 2.000000 | 96.000000 | 2.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **25%** | 0.440000 | 0.560000 | 3.000000 | 156.000000 | 3.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| **50%** | 0.640000 | 0.720000 | 4.000000 | 200.000000 | 3.000000 | 0.000000 | 0.000000 | 0.000000 | 2.000000 |
| **75%** | 0.820000 | 0.870000 | 5.000000 | 245.000000 | 4.000000 | 0.000000 | 0.000000 | 0.000000 | 2.000000 |
| **max** | 1.000000 | 1.000000 | 7.000000 | 310.000000 | 10.000000 | 1.000000 | 1.000000 | 1.000000 | 3.000000 |

### 4.1 How many employees works in each department?

Depending on how many employees work in each department, you can learn more about the type of company segment.

In [9]:

print(data['department'].value\_counts())

sales 4140

technical 2720

support 2229

IT 1227

product\_mng 902

marketing 858

RandD 787

accounting 767

hr 739

management 630

Name: department, dtype: int64

### 4.2 How many employees per salary range?

The employees salary is divided in Low (1), Medium (2) and High (3), distributed as follows:

In [10]:

print(data['salary'].value\_counts())

1 7316

2 6446

3 1237

Name: salary, dtype: int64

### 4.3 How many employees per salary range and department?

In [11]:

table = data.pivot\_table(values="satisfaction\_level", index="department", columns="salary",aggfunc=np.count\_nonzero)

table

Out[11]:

| **salary** | **1** | **2** | **3** |
| --- | --- | --- | --- |
| **department** |  |  |  |
| **IT** | 609.0 | 535.0 | 83.0 |
| **RandD** | 364.0 | 372.0 | 51.0 |
| **accounting** | 358.0 | 335.0 | 74.0 |
| **hr** | 335.0 | 359.0 | 45.0 |
| **management** | 180.0 | 225.0 | 225.0 |
| **marketing** | 402.0 | 376.0 | 80.0 |
| **product\_mng** | 451.0 | 383.0 | 68.0 |
| **sales** | 2099.0 | 1772.0 | 269.0 |
| **support** | 1146.0 | 942.0 | 141.0 |
| **technical** | 1372.0 | 1147.0 | 201.0 |

### 4.4 How plot graphics?

In descriptive analysis is very useful to use graphics to represent the data. For that, is necessary to import the libraries:

* Matplotlib: is a plotting lybrary, usefull to plot statistical graphics. See: www.matplotlib.org
* Seaborn: is a library based on matplotlib that can draw attrative statistical graphics. See: seaborn.pydata.org/index.html

In [12]:

%**matplotlib** inline

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

sns.set()

#### Boxplot

A boxplot is a way of summarizing a set of data measured on an interval scale. It is often used in exploratory data analysis. It is a type of graph which is used to show the shape of the distribution, its central value, and variability. The picture produced consists of the most extreme values in the data set (maximum and minimum values), the lower and upper quartiles, and the median. [Definition taken from Valerie J. Easton and John H. McColl's Statistics Glossary v1.1]

**Figure 1.** Boxplot example

Boxplot is a good statistical graphic to analyze the dataset and indentify outliers values. An outlier is as observation that lies an abnormal distance from other values, in this case the analyst have to decide what is considered abnormal.

The boxplots below, give the information about the data distributions:

* Satisfaction level and Last evaluation has a skewed left (negative) ditribuitions.
* Number of projects has a skewed right(positive)ditribution.
* Average monthly hours has a simetric ditribution.

Analyse de distribution of the variables is important due the fact that many statistical tests assume normal distribution.

In [13]:

f, axes = plt.subplots(2,2, figsize=(10,10), sharex=**True**)

plt.subplots\_adjust(wspace=0.5)*# adjust the space between the plots*

sns.despine(left=**True**)

*# plot a boxplot of satisfaction\_level to see if there is outliers*

sns.boxplot( x= 'satisfaction\_level', data=data, orient='v',ax=axes[0,0])

*# plot a boxplot of last\_evaluation to see if there is outliers*

sns.boxplot( x= 'last\_evaluation', data=data, orient='v',ax=axes[0,1])

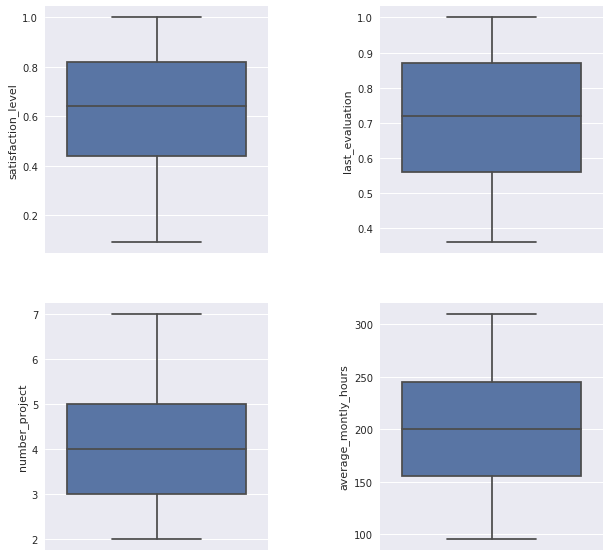
*# plot a boxplot of number\_project to see if there is outliers*

sns.boxplot( x= 'number\_project', data=data, orient='v',ax=axes[1,0])

*# plot a boxplot of average\_montly\_hours to see if there is outliers*

sns.boxplot( x= 'average\_montly\_hours', data=data, orient='v',ax=axes[1,1]);

*#Put a ; at the end of the last line to suppress the printing of output*



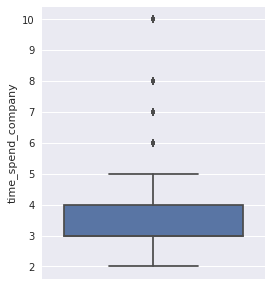
In the boxplots below it is possibel to see that only time\_spend\_company has outliers. Let's explain what kind of information is possible to conclude:

* The employees with more time in the company have 10 years, so is possible to say that is a relatively young company.
* Most of the employees have between 3 or 4 years in the company.

In [14]:

plt.figure(figsize=(4,5))

sns.boxplot( x= 'time\_spend\_company', data=data, orient='v');



### 4.5 Correlation Analysis

The correlation is a very useful statitiscal analysis that describes the degree of relationship between two variables. Let´s see the table below and the heat map to see what relationship are in the data.

In the heat map is possible to see:

* Negative correlation of (-0.39) between satisfaction\_level and the employees that left the company.
* The highest positive correlation is between number of projects and average monthly hours (0.42).
* Last\_evaluation is high correlated to number\_project(0.35)and average\_monthly\_hours(0.34).
* Work\_accident have a low negative correlation(-0.15)and salary (-0.16) with employees that left.

In [15]:

corr = data.corr()

corr

Out[15]:

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **left** | **promotion\_last\_5years** | **salary** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **satisfaction\_level** | 1.000000 | 0.105021 | -0.142970 | -0.020048 | -0.100866 | 0.058697 | -0.388375 | 0.025605 | 0.050022 |
| **last\_evaluation** | 0.105021 | 1.000000 | 0.349333 | 0.339742 | 0.131591 | -0.007104 | 0.006567 | -0.008684 | -0.013002 |
| **number\_project** | -0.142970 | 0.349333 | 1.000000 | 0.417211 | 0.196786 | -0.004741 | 0.023787 | -0.006064 | -0.001803 |
| **average\_montly\_hours** | -0.020048 | 0.339742 | 0.417211 | 1.000000 | 0.127755 | -0.010143 | 0.071287 | -0.003544 | -0.002242 |
| **time\_spend\_company** | -0.100866 | 0.131591 | 0.196786 | 0.127755 | 1.000000 | 0.002120 | 0.144822 | 0.067433 | 0.048715 |
| **Work\_accident** | 0.058697 | -0.007104 | -0.004741 | -0.010143 | 0.002120 | 1.000000 | -0.154622 | 0.039245 | 0.009247 |
| **left** | -0.388375 | 0.006567 | 0.023787 | 0.071287 | 0.144822 | -0.154622 | 1.000000 | -0.061788 | -0.157898 |
| **promotion\_last\_5years** | 0.025605 | -0.008684 | -0.006064 | -0.003544 | 0.067433 | 0.039245 | -0.061788 | 1.000000 | 0.098119 |
| **salary** | 0.050022 | -0.013002 | -0.001803 | -0.002242 | 0.048715 | 0.009247 | -0.157898 | 0.098119 | 1.000000 |

In [16]:

sns.set(style='white')

mask = np.zeros\_like(corr, dtype=np.bool)

mask[np.triu\_indices\_from(mask)] = **True**

*# Inserir a figura*

f, ax = plt.subplots(figsize=(13,8))

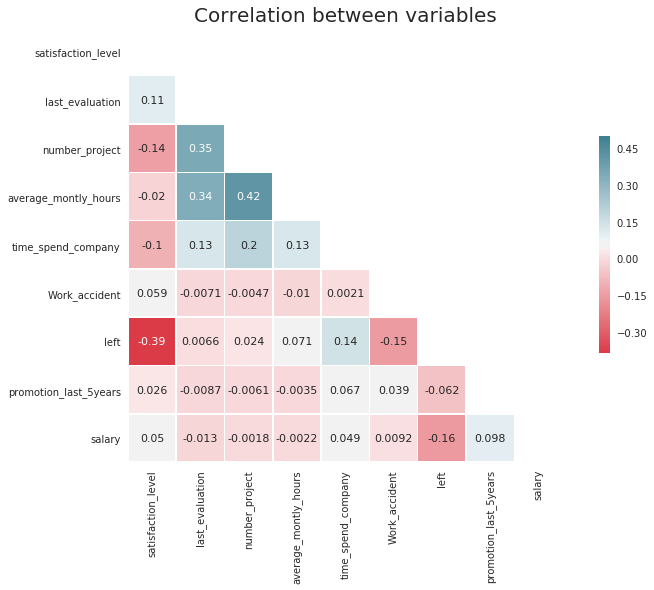
cmap = sns.diverging\_palette(10,220, as\_cmap=**True**)

*#Desenhar o heatmap com a máscara*

ax = sns.heatmap(corr, mask=mask, cmap=cmap, vmax= .5, annot=**True**, annot\_kws= {'size':11}, square=**True**, xticklabels=**True**, yticklabels=**True**, linewidths=.5,

cbar\_kws={'shrink': .5}, ax=ax)

ax.set\_title('Correlation between variables', fontsize=20);



## 5. Hypothesis

Now let's extract some more informations and testing some hypothesis

### 5.1 How many employees left the company?

In [17]:

print(data['left'].value\_counts()[1],"employees left the company")

3571 employees left the company

In [18]:

*# The plot show the amount o employees that stayed and left the company.*

plt.figure(figsize=(4,5))

ax = sns.countplot(data.left)

total = float(len(data))

**for** p **in** ax.patches:

height = p.get\_height()

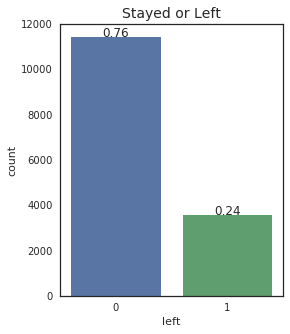
ax.text(p.get\_x()+p.get\_width()/2.,

height + 3,

'**{:1.2f}**'.format(height/total),

ha="center")

plt.title('Stayed or Left', fontsize=14);



### First Hypothesis

The first hypothesis is that salary is the reason why the employees left the company. Let's see if is this correct.

In [19]:

j = sns.factorplot(x='salary', y='left', kind='bar', data=data)

plt.title('Employees that left by salary level', fontsize=14)

j.set\_xticklabels(['High', 'Medium', 'Low']);



In the graphic Salaries by department is possible to see the distribuition of the salaries by department.

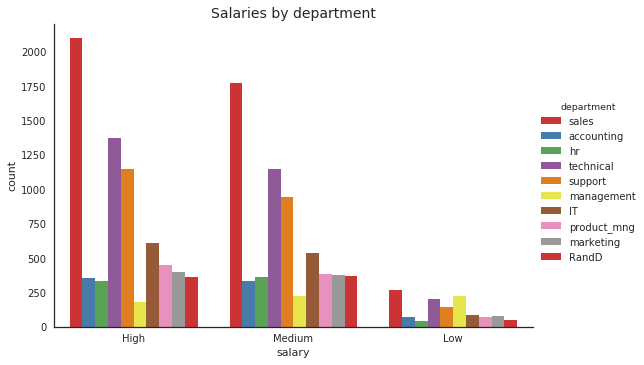
* Most of the employees of the sales department have low or medium salaries, this may be due that in some companies the sales commission is paid separately.
* Technical department is in the second place where most of the employees receives low and medium salaries.

In [20]:

h = sns.factorplot(x = 'salary', hue='department', kind ='count', size = 5,aspect=1.5, data=data, palette='Set1' )

plt.title("Salaries by department", fontsize=14)

h.set\_xticklabels(['High', 'Medium', 'Low']);



In the graphic(Salary Comparison):

* The manangement department has the biggest difference between the salary of the employees who stayed and those that left.
* It's not possible to see a huge difference in other departments.

The first hypothesis looks very weak to be the main reason why the employees left the company.

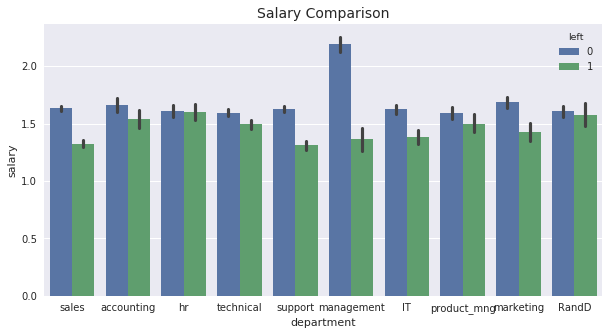
In [21]:

sns.set()

plt.figure(figsize=(10,5))

sns.barplot(x='department', y='salary', hue='left', data=data)

plt.title('Salary Comparison', fontsize=14);



### Second Hypothesis

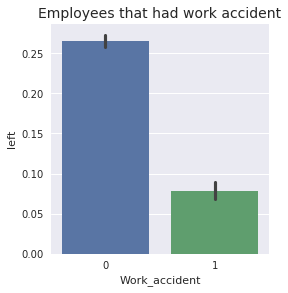
It is a dangerous job?

The second hypothesis is: employees leave the company because work is not safe.

In [22]:

sns.factorplot(x='Work\_accident', y='left', kind='bar', data=data)

plt.title('Employees that had work accident', fontsize=14);



About 14% of the employees had a work accident, although of the high number only of accidents only 169 employees data left the company had work a accident. Then this hypothesis is discarded.

In [23]:

print(data.Work\_accident.sum())

print(data.Work\_accident.mean())

print((data[data['left']==1]['Work\_accident']).sum())

2169

0.1446096406427095

169

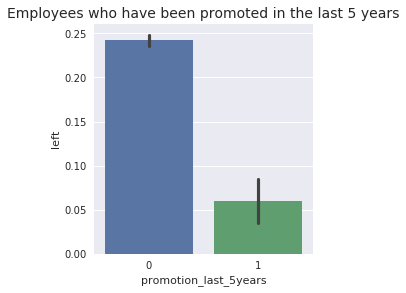
### Third Hypothesis

Is this company a good place to grow professionally?

In [24]:

sns.factorplot(x='promotion\_last\_5years', y='left', kind='bar', data=data)

plt.title('Employees who have been promoted in the last 5 years', fontsize=14);



In the last five years only 319 employees had promotion, this is equivalent to 2% of all employees. This may be a problem because if it is difficult to get promoted many employees become unmotivated and start looking for a new job.

In [25]:

print(data.promotion\_last\_5years.sum())

print(data.promotion\_last\_5years.mean())

319

0.021268084538969265

**Years in the company**

In the graphic 'Years in the company' we can identify an important characteristic.

* Employees with 7 or more years didn't left, maybe because with the passing of the years they are more confortable and not so interested in look for a new challenge in other company.
* The problem starts when the employees have more than 3 years and get worst when they achieve 5 years.
* It is too early to say that the difficult to get promoted is the main reason for the leaving of the employees, but more research is needed.

In [26]:

plt.figure(figsize =(7,5))

bins = np.linspace(1.0, 11,10)

plt.hist(data[data['left']==1]['time\_spend\_company'], bins=bins, alpha=1, label='Employees Left')

plt.hist(data[data['left']==0]['time\_spend\_company'], bins=bins, alpha = 0.5, label = 'Employee Stayed')

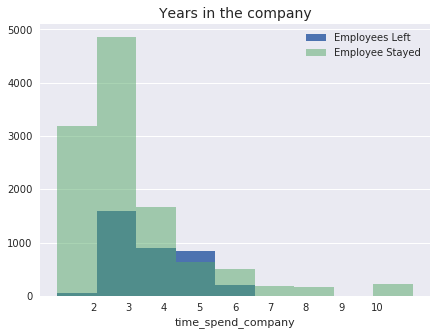
plt.grid(axis='x')

plt.xticks(np.arange(2,11))

plt.xlabel('time\_spend\_company')

plt.title('Years in the company', fontsize=14)

plt.legend(loc='best');



### Performance Analysis

There are 2 distincts groups of employees. A group with poor performance and other with high performance employees. It's natural that employees that don't work well leave the company, but the main problem is that the high performance employees is leaving too and it's necessary to understand why.

In [27]:

plt.figure(figsize =(7,7))

bins = np.linspace(0.305, 1.0001, 14)

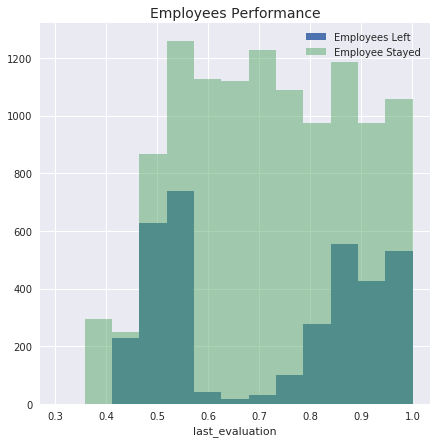
plt.hist(data[data['left']==1]['last\_evaluation'], bins=bins, alpha=1, label='Employees Left')

plt.hist(data[data['left']==0]['last\_evaluation'], bins=bins, alpha = 0.5, label = 'Employee Stayed')

plt.title('Employees Performance', fontsize=14)

plt.xlabel('last\_evaluation')

plt.legend(loc='best');



It is possible to see that 98% of employees with few projects that left also have poor performance.

And 95% of the employees with 5 or more projects that left the company had the highest performance.

3 or 4 are the best number of projects.

In [28]:

poor\_performance\_left = data[(data.last\_evaluation <= 0.62) & (data.number\_project == 2) & (data.left == 1)]

print('poor\_performance\_left:',len(poor\_performance\_left))

poor\_performance\_stayed = data[(data.last\_evaluation > 0.62) & (data.number\_project == 2) & (data.left == 1)]

print('poor\_performance\_stayed:',len(poor\_performance\_stayed))

print('**\n**')

high\_performance\_left= data[(data.last\_evaluation <= 0.62) & (data.number\_project >=5) & (data.left == 1)]

high\_performance\_stayed= data[(data.last\_evaluation > 0.8) & (data.number\_project >=5) & (data.left == 0)]

print('high\_performance\_left:',len(high\_performance\_left))

print('high\_performance\_stayed', len(high\_performance\_stayed))

plt.figure(figsize =(7,5))

bins = np.linspace(1.5,7.5, 7)

plt.hist(data[data['left']==1]['number\_project'], bins=bins, alpha=1, label='Employees Left')

plt.hist(data[data['left']==0]['number\_project'], bins=bins, alpha = 0.5, label = 'Employee Stayed')

plt.title('Number of projects', fontsize=14)

plt.xlabel('number\_ projects')

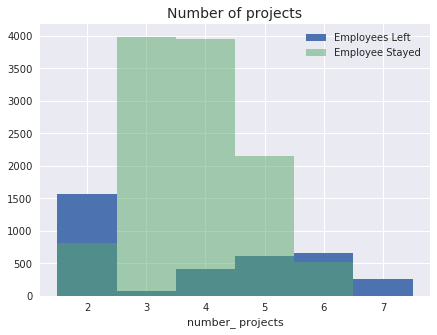
plt.legend(loc='best');

poor\_performance\_left: 1531

poor\_performance\_stayed: 36

high\_performance\_left: 47

high\_performance\_stayed 889



### Working hours

Again, there are 2 groups of employees. A group that works fewer hours and another that works more hours compared to the average hours worked.

In [29]:

plt.figure(figsize =(7,5))

bins = np.linspace(80,315, 15)

plt.hist(data[data['left']==1]['average\_montly\_hours'], bins=bins, alpha=1, label='Employees Left')

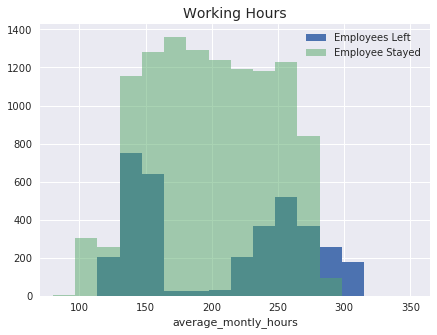
plt.hist(data[data['left']==0]['average\_montly\_hours'], bins=bins, alpha = 0.5, label = 'Employee Stayed')

plt.title('Working Hours', fontsize=14)

plt.xlabel('average\_montly\_hours')

plt.xlim((70,365))

plt.legend(loc='best');



Clearly is possible to see that the employees with 6 projects or more, work on average 20% more hours.

In [30]:

groupby\_number\_projects = data.groupby('number\_project').mean()

groupby\_number\_projects = groupby\_number\_projects['average\_montly\_hours']

print(groupby\_number\_projects)

plt.figure(figsize=(7,5))

groupby\_number\_projects.plot();

number\_project

2 160.342546

3 197.507522

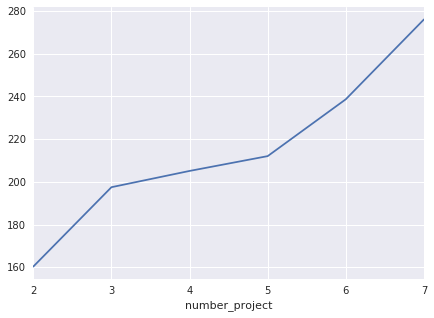
4 205.122108

5 212.061572

6 238.694208

7 276.078125

Name: average\_montly\_hours, dtype: float64



With the information above the employees that left the company are grouped as:

* Employees with 2 projects and worked less than the average of the company.
* Employees with 5 or more projects that worked at least 20% more than the average.

In [31]:

work\_less\_hours\_left = data[(data.average\_montly\_hours < 200) & (data.number\_project == 2) & (data.left == 1)]

print('work\_less\_hours\_left:',len(work\_less\_hours\_left))

work\_more\_hours\_left = data[(data.average\_montly\_hours > 240) & (data.number\_project >=5 ) & (data.left == 1)]

print('work\_more\_hours\_left:',len(work\_more\_hours\_left))

*#<p><font color="red">Aqui você fala sobre a relação entre horas de trabalho e quantidade de projetos, mas isso não é exibido no gráfico</font></p>*

work\_less\_hours\_left: 1535

work\_more\_hours\_left: 1225

### Satisfaction Level

It is possible to see 3 interesting peaks in the satisfaction levels of the employees that left the company.

* We have a peak of employees who are totally disappointed.
* Another peak at 0.4, representing another group with the satisfaction level below the average.
* And another amount in the range 0.7 and 0.9, with employees that left, although the high satisfaction.

In [32]:

plt.figure(figsize =(7,5))

bins = np.linspace(0.006,1.000, 15)

plt.hist(data[data['left']==1]['satisfaction\_level'], bins=bins, alpha=1, label='Employees Left')

plt.hist(data[data['left']==0]['satisfaction\_level'], bins=bins, alpha = 0.5, label = 'Employee Stayed')

plt.title('Employees Satisfaction', fontsize=14)

plt.xlabel('satisfaction\_level')

plt.xlim((0,1.05))

plt.legend(loc='best');



#### Average satisfaction for years in the company

In [33]:

groupby\_time\_spend = data.groupby('time\_spend\_company').mean()

groupby\_time\_spend['satisfaction\_level']

Out[33]:

time\_spend\_company

2 0.697078

3 0.626314

4 0.467517

5 0.610305

6 0.603440

7 0.635957

8 0.665062

10 0.655327

Name: satisfaction\_level, dtype: float64

#### When the employees becames unsatisfayed?

In next results it is clear the drop in satisfaction when employees are working on 6 or more projects.

In [34]:

sns.set()

sns.set\_context("talk")

ax = sns.factorplot(x="number\_project", y="satisfaction\_level", col="time\_spend\_company",col\_wrap=4, size=3, color='blue',sharex=**False**, data=data)

ax.set\_xlabels('Number of Projects');



Let´s see why the most valuable employees tend to leave.

From the employees that left with high performance, 4 or more years in the company and working on 5 or more project had:

* Low satisfaction level,
* Worked more hours,
* Haven´t been promoted in the last five years.

In [35]:

func\_living = data[(data.last\_evaluation >= 0.70) | (data.time\_spend\_company >=4) | (data.number\_project >= 5)]

corr2 = func\_living.corr()

sns.set(style='white')

mask = np.zeros\_like(corr2, dtype=np.bool)

mask[np.triu\_indices\_from(mask)] = **True**

*# Insert the graphic*

f, ax = plt.subplots(figsize=(13,8))

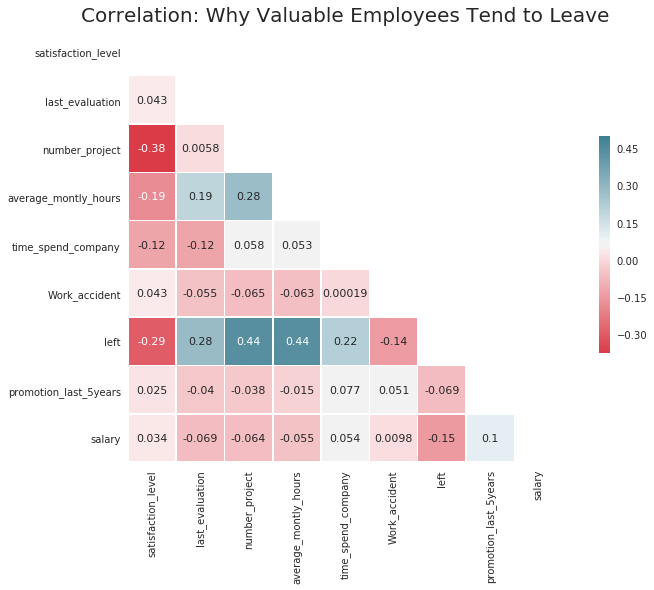
cmap = sns.diverging\_palette(10,220, as\_cmap=**True**)

*#Draw heat map mask*

ax = sns.heatmap(corr2, mask=mask, cmap=cmap, vmax= .5, annot=**True**, annot\_kws= {'size':11}, square=**True**, xticklabels=**True**, yticklabels=**True**, linewidths=.5,

cbar\_kws={'shrink': .5}, ax=ax)

ax.set\_title('Correlation: Why Valuable Employees Tend to Leave', fontsize=20);



### Summary of the Exploratory Data Analysis

* It is a relatively young company, on average, employees have 3 or 4 years in the company and the oldest employees are working 10 years.
* The biggest difference in the salary from who stayed and those who left, was found in the managemnet department, in the others departments although the salaries of who stayed be higher in average, it is not a big difference.
* The number of employees that had a work accident is about 14%, of which only 169 employees left the company, so don't seem to have a correlation with the employees leaving.
* In five years only 2% of the employees were promoted. Is possible that many employees get unmotivated and start planning to leave.
* Employees with 7 or longer in the company didn't left. Employees with 5 years have more chances to leaving.
* There are 2 distincts groups of employees performance that left. A group with poor performance with 2 projects and others with high performance with 5 or more projects. It is not necessary retain all the employees, the focus is on keeping employees with high performance.
* The employees with 4 years in the company have the lowest average satisfaction level of all the company with (0.47).
* The satisfaction drops when the employees are working in 5 or more projects. A number of 3 or 4 projects seems to be ideal independent of the time spend in the company.
* The employees with 5 or more projects that left also worked at least 20% more hours than the average of the company.
* The satisfaction level of the employees that left is grouped in totally disappointed, below the average satisfaction and satisfied.

# ALTERNATIVE METHOD

In [1]:

*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load in*

**import** **numpy** **as** **np** *# linear algebra*

**import** **pandas** **as** **pd** *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the "../input/" directory.*

*# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory*

**from** **subprocess** **import** check\_output

print(check\_output(["ls", "../input"]).decode("utf8"))

*# Any results you write to the current directory are saved as output.*

HR\_comma\_sep.csv

In [2]:

**from** **sklearn.cross\_validation** **import** train\_test\_split

**from** **sklearn.ensemble** **import** RandomForestRegressor

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn.ensemble** **import** AdaBoostClassifier

**from** **sklearn.tree** **import** export\_graphviz

**import** **seaborn** **as** **sns**

**import** **matplotlib.pyplot** **as** **plt**

/opt/conda/lib/python3.6/site-packages/sklearn/cross\_validation.py:44: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model\_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.

"This module will be removed in 0.20.", DeprecationWarning)

In [3]:

df= pd.read\_csv('../input/HR\_comma\_sep.csv')

df.head(10)

Out[3]:

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **left** | **promotion\_last\_5years** | **sales** | **salary** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.38 | 0.53 | 2 | 157 | 3 | 0 | 1 | 0 | sales | low |
| **1** | 0.80 | 0.86 | 5 | 262 | 6 | 0 | 1 | 0 | sales | medium |
| **2** | 0.11 | 0.88 | 7 | 272 | 4 | 0 | 1 | 0 | sales | medium |
| **3** | 0.72 | 0.87 | 5 | 223 | 5 | 0 | 1 | 0 | sales | low |
| **4** | 0.37 | 0.52 | 2 | 159 | 3 | 0 | 1 | 0 | sales | low |
| **5** | 0.41 | 0.50 | 2 | 153 | 3 | 0 | 1 | 0 | sales | low |
| **6** | 0.10 | 0.77 | 6 | 247 | 4 | 0 | 1 | 0 | sales | low |
| **7** | 0.92 | 0.85 | 5 | 259 | 5 | 0 | 1 | 0 | sales | low |
| **8** | 0.89 | 1.00 | 5 | 224 | 5 | 0 | 1 | 0 | sales | low |
| **9** | 0.42 | 0.53 | 2 | 142 | 3 | 0 | 1 | 0 | sales | low |

In [4]:

df.isnull().sum()

Out[4]:

satisfaction\_level 0

last\_evaluation 0

number\_project 0

average\_montly\_hours 0

time\_spend\_company 0

Work\_accident 0

left 0

promotion\_last\_5years 0

sales 0

salary 0

dtype: int64

In [5]:

df.dtypes

Out[5]:

satisfaction\_level float64

last\_evaluation float64

number\_project int64

average\_montly\_hours int64

time\_spend\_company int64

Work\_accident int64

left int64

promotion\_last\_5years int64

sales object

salary object

dtype: object

In [6]:

df['sales'].unique()

Out[6]:

array(['sales', 'accounting', 'hr', 'technical', 'support', 'management',

'IT', 'product\_mng', 'marketing', 'RandD'], dtype=object)

In [7]:

df['salary'].unique()

Out[7]:

array(['low', 'medium', 'high'], dtype=object)

In [8]:

*#Low=*

*#Medium=*

*#High=10*

df['salary'].replace({'low':1,'medium':5,'high':10},inplace=**True**)

In [9]:

satisfaction\_level=df['satisfaction\_level']

last\_evaluation=df['last\_evaluation']

number\_project=df['number\_project']

average\_montly\_hours=df['average\_montly\_hours']

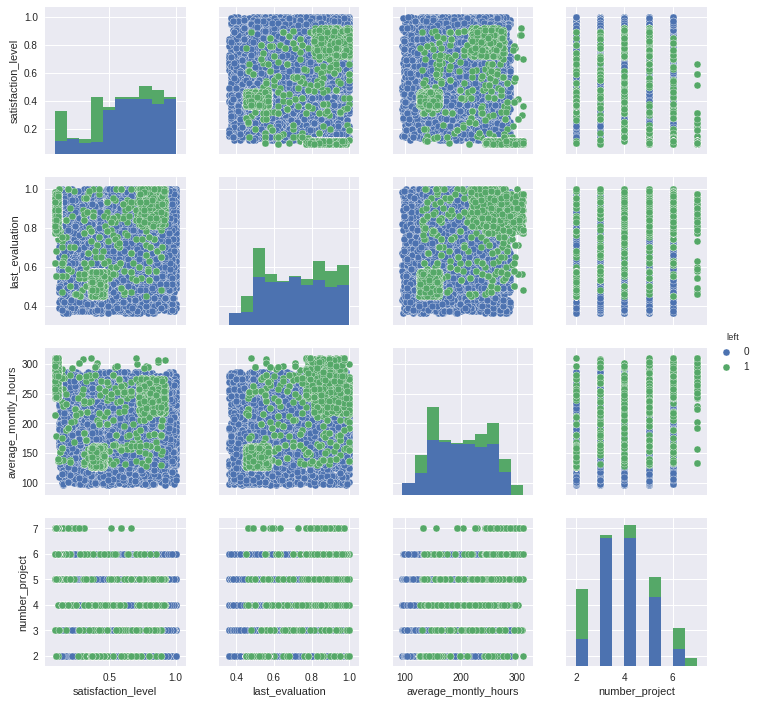
time\_spend\_company=df['time\_spend\_company']

Work\_accident=df['Work\_accident']

In [10]:

sns.pairplot(df, hue="left", vars=['satisfaction\_level', 'last\_evaluation', 'average\_montly\_hours','number\_project'])

plt.show()



In [11]:

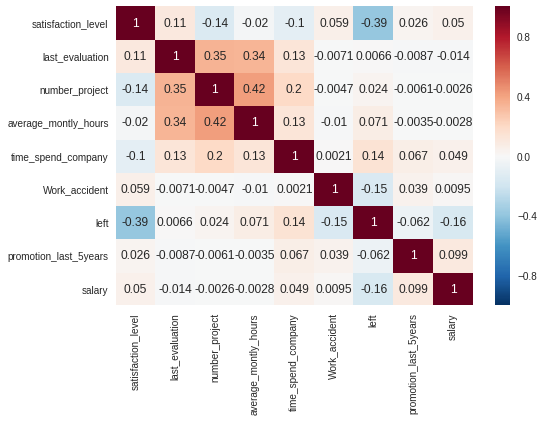
*#X\_train,y\_train,X\_test,y\_test=train\_test\_split(X,y,data\_size=0.2)*

corr=df.corr()

sns.heatmap(corr,annot=**True**)

Out[11]:

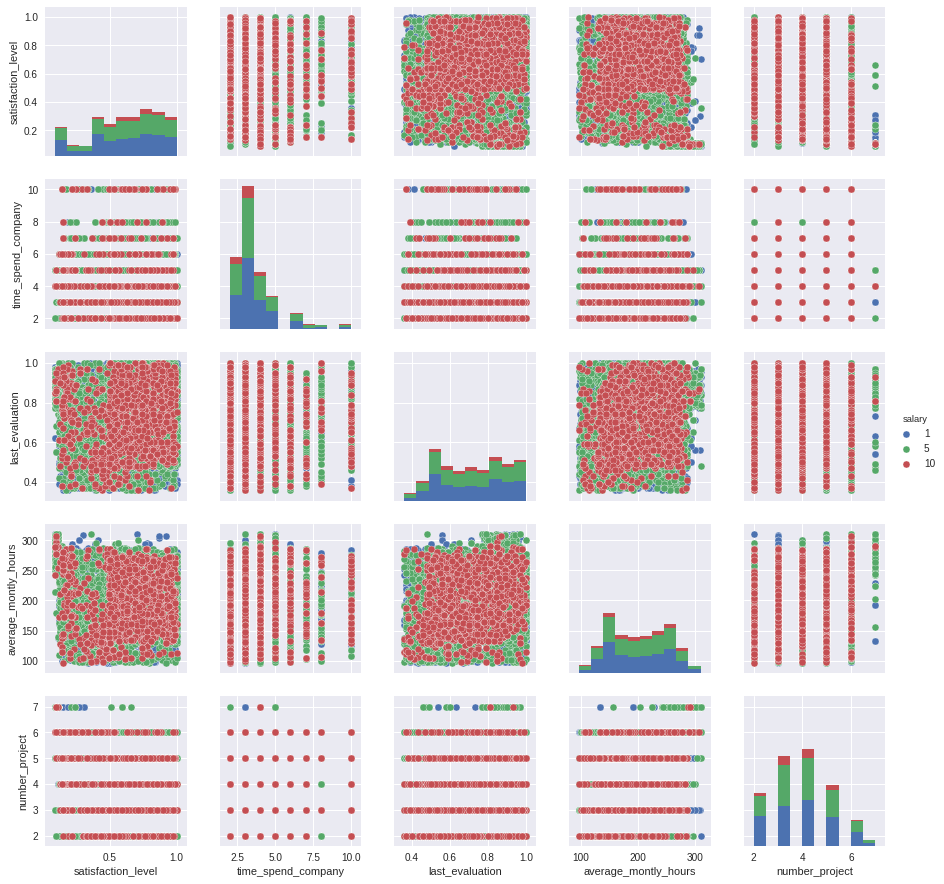
<matplotlib.axes.\_subplots.AxesSubplot at 0x7f41bf097198>



In [12]:

sns.pairplot(df, hue="salary", vars=['satisfaction\_level','time\_spend\_company', 'last\_evaluation', 'average\_montly\_hours','number\_project'])

plt.show()



In [13]:

*#Creating dummy variables for the column:sales*

dummies=pd.get\_dummies(df['sales'],prefix='sales')

df=pd.concat([df,dummies],axis=1)

df.drop(['sales'],axis=1,inplace=**True**)

df.head(10)

Out[13]:

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **left** | **promotion\_last\_5years** | **salary** | **sales\_IT** | **sales\_RandD** | **sales\_accounting** | **sales\_hr** | **sales\_management** | **sales\_marketing** | **sales\_product\_mng** | **sales\_sales** | **sales\_support** | **sales\_technical** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.38 | 0.53 | 2 | 157 | 3 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **1** | 0.80 | 0.86 | 5 | 262 | 6 | 0 | 1 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **2** | 0.11 | 0.88 | 7 | 272 | 4 | 0 | 1 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **3** | 0.72 | 0.87 | 5 | 223 | 5 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **4** | 0.37 | 0.52 | 2 | 159 | 3 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **5** | 0.41 | 0.50 | 2 | 153 | 3 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **6** | 0.10 | 0.77 | 6 | 247 | 4 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **7** | 0.92 | 0.85 | 5 | 259 | 5 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **8** | 0.89 | 1.00 | 5 | 224 | 5 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **9** | 0.42 | 0.53 | 2 | 142 | 3 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

# Parameters:

n\_estimators=No of trees in the forest

n\_jobs=The number of jobs to run in parallel for both fit and predict. If -1, then the number of jobs is set to the number of cores.

random\_state=seed used by random number generator,This is for reproductibility

max\_features=The number of features to consider when looking for the best split

1. 'auto'=All features(n)
2. 'sqrt'=sqrt(All features/n)
3. 'log2'=lo2(n)
4. None=n

min\_sample\_split=default=2

oob\_score:TRUE/FALSE Whether to use out-of-bag samples to estimate the generalization error.

In random forests there is no need of cross validation as this is internally done via oob(out of bag sampling).

In [14]:

*#Spilting data into test and train split:*

X=df.drop(['left'],axis=1)

y=df['left']

In [15]:

*#RandomForestClassifier*

model=RandomForestRegressor(n\_estimators=100,n\_jobs=-1,oob\_score=**True**,random\_state=19)

model.fit(X,y)

Out[15]:

RandomForestRegressor(bootstrap=True, criterion='mse', max\_depth=None,

max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=100, n\_jobs=-1, oob\_score=True, random\_state=19,

verbose=0, warm\_start=False)

In [16]:

*#RandomForestClassifier*

**from** **sklearn.metrics** **import** roc\_auc\_score

y\_pred=model.oob\_prediction\_

acc=roc\_auc\_score(y,y\_pred)

print('Accuracy :',acc)

Accuracy : 0.991462540433

In [17]:

*#K-Fold Validation*

*#Evaluating the model for a Ten-Fold Cross Validation*

acc=[]

**for** i **in** [50,100,150,200,250,300,350,400,500,600]:

*#Training the model*

clf=RandomForestRegressor(n\_estimators=i,n\_jobs=-1,oob\_score=**True**,random\_state=19)

clf.fit(X,y)

y\_pred=clf.oob\_prediction\_

k=roc\_auc\_score(y,y\_pred)

acc.append(k)

print(acc)

[0.99112293720258671, 0.99146254043309845, 0.99177373108364186, 0.99214341317737964, 0.99211786758478215, 0.99232524633792596, 0.99240608558011212, 0.99270432577915646, 0.99269741560446845, 0.99281477830542331]

In [18]:

**import** **matplotlib.pyplot** **as** **plt**

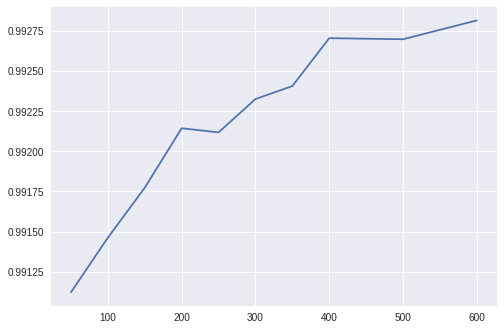
xx=[50,100,150,200,250,300,350,400,500,600]

yy=acc

plt.plot(xx,yy)

Out[18]:

[<matplotlib.lines.Line2D at 0x7f41bc90e2b0>]



In [19]:

model.n\_outputs\_

Out[19]:

1

In [20]:

model.n\_features\_

Out[20]:

18

In [21]:

*#Spilting data into test and train split:*

X=df.drop(['left'],axis=1)

y=df['left']

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2)

In [22]:

print(X\_train.shape,y\_train.shape,)

print(X\_test.shape,y\_test.shape)

(11999, 18) (11999,)

(3000, 18) (3000,)

In [23]:

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn.metrics** **import** roc\_auc\_score

*#DecisionTreeClassifier*

dt=DecisionTreeClassifier(max\_depth=3,min\_samples\_leaf=int(0.05\*len(X\_train)),random\_state=19)

boosted\_dt=AdaBoostClassifier(dt,algorithm='SAMME',n\_estimators=800,learning\_rate=0.5)

boosted\_dt.fit(X\_train,y\_train)

y\_predicted=boosted\_dt.predict(X\_test)

print ("Area under ROC curve: **%f**"%(roc\_auc\_score(y\_test, y\_predicted)))

Area under ROC curve: 0.955490