DAV INSTITUTE OF ENGINEERING AND TECHNOLOGY

SIX WEEKS INDUSTRIAL PROJECT REPORT

ON

HR ANALYTICS

SUBMITTED BY

Abhay Puri

B.Tech CSE

University Roll No.- 1604517

1. PREFACE

HR analytics, also called talent analytics, is the application of considerable data mining and business analytics techniques to human resources data. The goal of human resources analytics is to provide an organization with insights for effectively managing employees so that business goals can be reached quickly and efficiently. The challenge of human resources analytics is to identify what data should be captured and how to use the data to model and predict capabilities so the organization gets an optimal return on investment on its human capital.

HR analytics does not only deal with gathering data on employee efficiency. Instead, it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve the processes.

So how can companies use HR data analytics to make strategic personnel decisions? First, they'll need software, which isn't hard to find. Huge vendors such as Oracle, IBM and SAP compete with many smaller vendors to deliver the best HR analytics software as a service in the market.

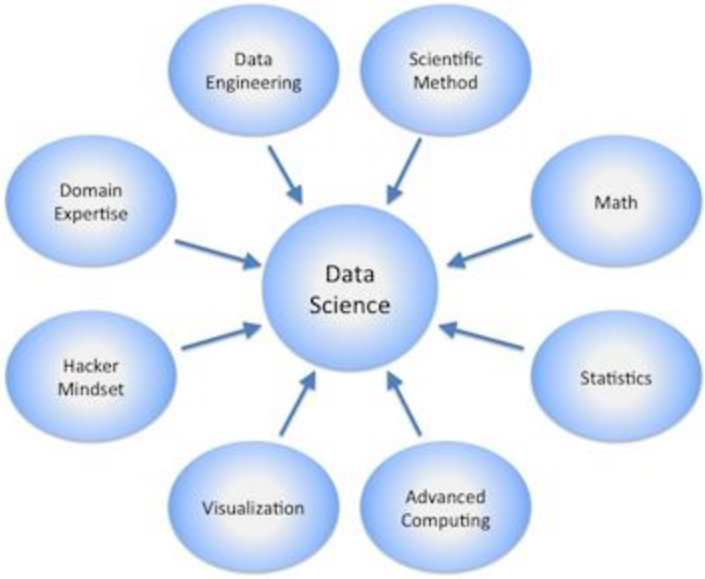
This Project focuses on solving this problem by applying a machine learning approach to understand the people data and bring up useful insights from the same.

1. INTRODUCTION TO DATA SCIENCE AND MACHINE LEARNING

WHAT IS DATA SCIENCE?

Data science is a multidisciplinary blend of **data inference, algorithmm development, and technology** in order to solve analytically complex problems.

At the core is data. Troves of raw information, streaming in and stored in enterprise data warehouses. Much to learn by mining it. Advanced capabilities we can build with it. Data science is ultimately about using this data in creative ways to generate business value:



Data science – discovery of data insight

This aspect of data science is all about uncovering findings from data. Diving in at a granular level to mine and understand complex behaviors, trends, and inferences. It's about surfacing hidden insight that can help enable companies to make smarter business decisions. For example:

- Netflix data mines movie viewing patterns to understand what drives user interest, and uses that to make decisions on which Netflix original series to produce.
- Target identifies what are major customer segments within it's base and the unique shopping behaviors within those segments, which helps to guide messaging to different market audiences.
- Proctor & Gamble utilizes time series models to more clearly understand future demand, which help plan for production levels more optimally.

How do data scientists mine out insights? It starts with data exploration. When given a challenging question, data scientists become detectives. They investigate leads and try to understand pattern or characteristics within the data. This requires a big dose of analytical creativity.

Then as needed, data scientists may apply quantitative technique in order to get a level deeper – e.g. inferential models, segmentation analysis, time series forecasting, synthetic control experiments, etc. The intent is to scientifically piece together a forensic view of what the data is really saying.

This data-driven insight is central to providing strategic guidance. In this sense, data scientists act as consultants, guiding business stakeholders on how to act on findings.

Data science – development of data product

A "data product" is a technical asset that: (1) utilizes data as input, and (2) processes that data to return algorithmically-generated results. The classic example of a data product is a recommendation engine, which ingests user data, and makes personalized recommendations based on that data. Here are some examples of data products:

- Amazon's recommendation engines suggest items for you to buy, determined by their algorithms. Netflix recommends movies to you. Spotify recommends music to you.
- Gmail's spam filter is data product – an algorithm behind the scenes processes incoming mail and determines if a message is junk or not.
- Computer vision used for self-driving cars is also data product – machine learning algorithms are able to recognize traffic lights, other cars on the road, pedestrians, etc.

This is different from the "data insights" section above, where the outcome to that is to perhaps provide advice to an executive to make a smarter business decision. In contrast, a data product is technical functionality that encapsulates an algorithm, and is designed to integrate directly into core applications. Respective examples of applications that incorporate data product behind the scenes: Amazon's homepage, Gmail's inbox, and autonomous driving software.

Data scientists play a central role in developing data product. This involves building out algorithms, as well as testing, refinement, and technical deployment into production systems. In this sense, data scientists serve as technical developers, building assets that can be leveraged at wide scale.

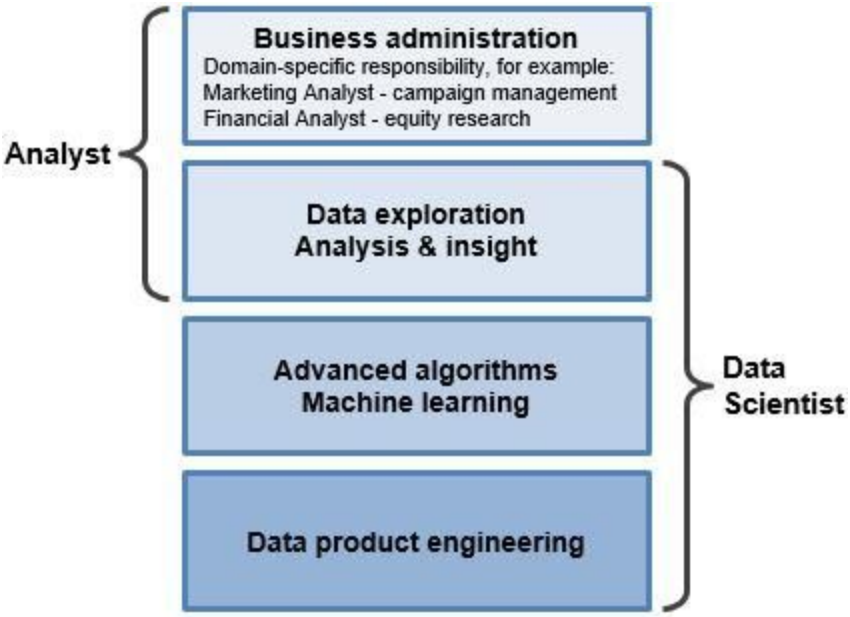
Analytics and machine learning – how it ties to data science

There are a slew of terms closely related to data science.

What is Analytics?

Analytics has risen quickly in popular business lingo over the past several years; the term is used loosely, but generally meant to describe critical thinking that is quantitative in nature. Technically, analytics is the "science of analysis" — put another way, the practice of analyzing information to make decisions.

Is "analytics" the same thing as data science? Depends on context. Sometimes it is synonymous with the definition of data science that we have described, and sometimes it represents something else. A data scientist using raw data to build a predictive algorithm falls into the scope of analytics. At the same time, a non-technical business user interpreting pre-built dashboard reports (e.g. GA) is also in the realm of analytics, but does not cross into the skill set needed in data science. Analytics has come to have fairly broad meaning. At the end of the day, as long as you understand beyond the buzzword level, the exact semantics don't matter much.



What is Machine Learning?

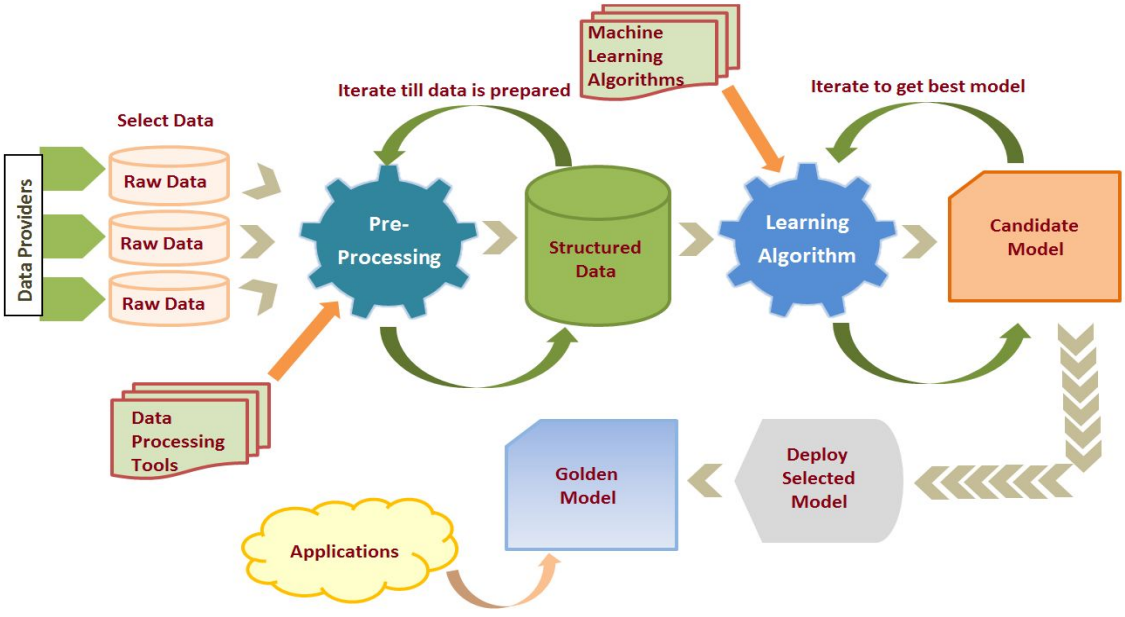
Machine learning is a term closely associated with data science. It refers to a broad class of methods that revolve around data modeling to algorithmically make predictions, and algorithmically decipher patterns in data.

Machine learning for making predictions — Core concept is to use tagged data to train predictive models. Tagged data means observations where ground truth is already known.

Training models means automatically characterizing tagged data in ways to predict tags for unknown data points. E.g. a credit card fraud detection model can be trained using a historical record of tagged fraud purchases. The resultant model estimates the likelihood that any new purchase is fraudulent. Common methods for training models range from basic regressions to complex neural nets. All follow the same paradigm known as supervised learning.

Machine learning for pattern discovery — Another modeling paradigm known as unsupervised learning tries to surface underlying patterns and associations in data when no existing ground truth is known (i.e. no observations are tagged). Within this broad category of methods, the most commonly used are clustering techniques, which algorithmically detect what are the natural groupings that exist in a data set. For example, clustering can be used to programmatically learn the natural customer segments in a company's user base. Other unsupervised methods for mining underlying characteristics include: principal component analysis, hidden markov models, topic models, and more.

Not all machine learning methods fit neatly into the above two categories. For example, collaborative filtering is a type of recommendations algorithm with elements related to both supervised and unsupervised learning. Contextual bandits are a twist on supervised learning where predictions get adaptively modified on-the-fly using live feedback.



This wide-ranging breadth of machine learning techniques comprises an important part of the data science toolbox. It is up to the data scientist to figure out which tool to use in different

circumstances (as well as how to use the tool correctly) in order to solve analytically open-ended problems.

2. INTRODUCTION TO THE PROJECT

HR ANALYSIS

WHAT IS HR ANALYTICS?

Human resource analytics (HR analytics) is an area in the field of analytics that refers to applying analytic processes to the human resource department of an organization in the hope of improving employee performance and therefore getting a better return on investment. HR analytics does not just deal with gathering data on employee efficiency. Instead, it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve these processes.

What HR analytics does is correlate business data and people data, which can help establish important connections later on. The key aspect of HR analytics is to provide data on the impact the HR department has on the organization as a whole. Establishing a relationship between what HR does and business outcomes – and then creating strategies based on that information – is what HR analytics is all about.

HR has core functions that can be enhanced by applying processes in analytics. These are acquisition, optimization, paying and developing the workforce of the organization. HR analytics can help to dig into problems and issues surrounding these requirements, and using analytical workflow, guide the managers to answer questions and gain insights from information at hand, then make relevant decisions and take appropriate actions.

WHY IS ANALYTICS IMPORTANT TO HR?

HR analytics, also called talent analytics, is the application of considerable data mining and business analytics techniques to human resources data. The goal of human resources analytics is to provide an organization with insights for effectively managing employees so that business goals can be reached quickly and efficiently. The challenge of human resources analytics is to identify what data should be captured and how to use the data to model and predict capabilities so the organization gets an optimal return on investment on its human capital.

HR analytics does not only deal with gathering data on employee efficiency. Instead, it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve the processes.

So how can companies use HR data analytics to make strategic personnel decisions? First, they'll need software, which isn't hard to find. Huge vendors such as Oracle, IBM and SAP compete with many smaller vendors to deliver the best HR analytics software as a service in the market.

This Project focuses on solving this problem by applying a machine learning approach to understand the people data and bring up useful insights from the same.

Some typical benefits and use cases of analytics are as follows:

- Improve organizational performance through high quality talent related decisions
- Forecast workforce requirements and utilization for improved business performance.
- Optimization of talents through development and planning.
- Identify the primary reasons for attrition and identify high-value employees for leaving.
- Provide the source of competitive platform for the organizations
- Manages applicants in better way on basis of qualification for a specific position.
- Recognize the factors which turn the employee satisfaction and productivity.
- To determine the individuals KPIs on the business.
- Enabling HR to demonstrate its benefaction to achieving corporate goals.

PROJECT DESCRIPTION

The focus of the project is to understand why the best and most experienced employees are leaving the company. The project explores good insights of problems that the Human Resource department deals daily. In many industries retaining their best employees is a question of long term strategy, and can impact the company's growth or put in financial risk, mainly if the employees leave to work at the competitor.

The project studies various aspects of employees leaving the company, hence helping the companies to know factors responsible and retain their employees.

Data has been very consciously preprocessed and visualized. Each factor has been visualized to perceive its consequence on the employee's behavior.

We have two goals: first, we want to understand why valuable employees leave, and second, we want to predict who will leave next.

Therefore, we propose to work with the HR department to gather relevant data about the employees and to communicate the significant effect that could explain and predict employees' departure.

3. OBJECTIVE OF THE PROJECT

We have two goals: first, we want to understand why valuable employees leave, and second, we want to predict who will leave next.

Therefore, we propose to work with the HR department to gather relevant data about the employees and to communicate the significant effect that could explain and predict employees' departure.

1. Exploring data and differentiating between employees who are leaving the company and those who have chosen to stay till now.

    This part mainly consists of visualization to get a feel of the distribution for different groups of employees.

2. Find out the important factors which dictate the turnover of the employees.

3. Fit prediction models and find out which employees might leave next.

4. LIBRARIES REQUIRED

1. Numpy:

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

2. Pandas:

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

3. OS:

This module provides a portable way of using operating system dependent functionality.

Notes on the availability of these functions:

- The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function os.stat (path) returns stat information about *path* in the same format (which happens to have originated with the POSIX interface).
- Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.
- All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.

4. Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

5. Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Here is some of the functionality that seaborn offers:

A dataset-oriented API for examining relationships between multiple variables

Specialized support for using categorical variables to show observations or aggregate statistics

Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data

Automatic estimation and plotting of linear regression models for different kinds dependent variables

Convenient views onto the overall structure of complex datasets

High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations

Concise control over matplotlib figure styling with several built-in themes

Tools for choosing color palettes that faithfully reveal patterns in your data

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

6. Scikit-Learn:

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis

Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

Some popular groups of models provided by scikit-learn include:

- **Clustering**: for grouping unlabeled data such as KMeans.
- **Cross Validation**: for estimating the performance of supervised models on unseen data.
- **Datasets**: for test datasets and for generating datasets with specific properties for investigating model behavior.

- **Dimensionality Reduction**: for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
- **Ensemble methods**: for combining the predictions of multiple supervised models.
- **Feature extraction**: for defining attributes in image and text data.
- **Feature selection**: for identifying meaningful attributes from which to create supervised models.
- **Parameter Tuning**: for getting the most out of supervised models.
- **Manifold Learning**: For summarizing and depicting complex multi-dimensional data.
- **Supervised Models**: a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

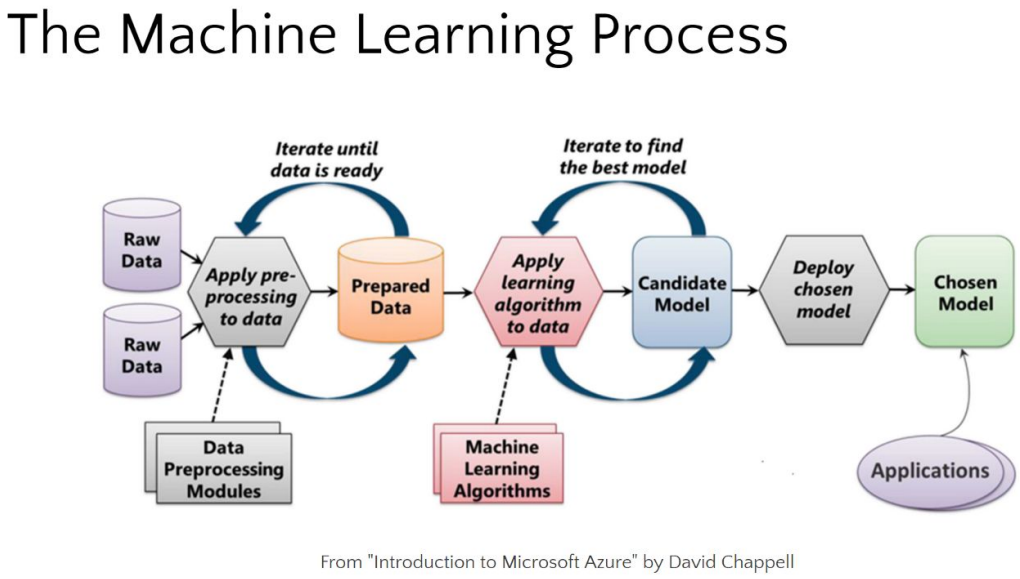
6. DATA PREPROCESSING

What is Data Preprocessing?

Data preprocessing is a **data** mining technique that involves transforming raw **data** into an understandable format. Real-world **data** is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. **Data preprocessing** is a proven method of resolving such issues.

Why we use Data Preprocessing?

In Real world **data** are generally incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate **data**. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names.



From "Introduction to Microsoft Azure" by David Chappell

Steps in Data Preprocessing

Step 1 : Import the libraries

Step 2 : Import the data-set

Step 3 : Check out the missing values

Step 4 : See the Categorical Values

Step 5 : Splitting the data-set into Training and Test Set

Step 6 : Feature Scaling

STEP1: Import the libraries

```
In [3]: # Import the Libraries
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')
```

Import the libraries

This is how we import libraries in Python using import keyword

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

1. A powerful N-dimensional array object

2. Sophisticated (broadcasting) functions

3. Tools for integrating C/C++ and FORTRAN code

4. Useful linear algebra, Fourier transform, and random number capabilities

Pandas is for data manipulation and analysis. Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Pandas is a NumFOCUS sponsored project. This will help ensure the success of development of pandas as a world-class open-source project, and makes it possible to donate to the project.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn't warrant raising an exception and terminating the program. For example, one might want to issue a warning when a program uses an obsolete module.

STEP 2: Import the dataset

```
In [3]: data=pd.read_csv('HR_comma_sep.csv')

In [4]: data.head(5)
```

Out[4]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promot
0	0.38	0.53	2	157	3	0	1	
1	0.80	0.86	5	262	6	0	1	
2	0.11	0.88	7	272	4	0	1	
3	0.72	0.87	5	223	5	0	1	
4	0.37	0.52	2	159	3	0	1	

```
In [5]: data.tail(5)
```

Out[5]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promo
4994	0.40	0.57	2	151	3	0	1	

Import the dataset

```
In [5]: data.shape
Out[5]: (14999, 10)

In [7]: data.index
Out[7]: RangeIndex(start=0, stop=14999, step=1)

In [8]: data.columns
Out[8]: Index(['satisfaction_level', 'last_evaluation', 'number_project',
               'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',
               'promotion_last_5years', 'sales', 'salary'],
              dtype='object')

In [7]: data.describe()
Out[7]:
```

Data info

By using Pandas we import our data-set and the file I used here is .csv file. However, to access and to use fastly we use CSV files because of their light weights. After importing the dataset, head function has been usd (This function returns the first n rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it. By default it returns 5 rows.)

STEP 3: Check out the missing values

The concept of missing values is important to understand in order to successfully manage data. If the missing values are not handled properly by the researcher, then he/she may end up drawing an inaccurate inference about the data. Due to improper handling, the result obtained by the researcher will differ from ones where the missing values are present.

```
In [9]: data.isnull().sum()

Out[9]: 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
```

There are no null values encountered in the dataset. But in case we have null values in the dataset we can do the following steps to remove the null values:

1. Deleting Rows

This method commonly used to handle the null values. Here, we either delete a particular row if it has a null value for a particular feature and a particular column if it has more than 70-75% of missing values. This method is advised only when there are enough samples in the data set. One has to make sure that after we have deleted the data, there is no addition of bias. Removing the data will lead to loss of information which will not give the expected results while predicting the output.

```
data.dropna(inplace = True)
data.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

Pros:

- Complete removal of data with missing values results in robust and highly accurate model
- Deleting a particular row or a column with no specific information is better, since it does not have a high weightage

Cons:

- Loss of information and data
- Works poorly if the percentage of missing values is high (say 30%), compared to the whole dataset

2. Replacing With Mean/Median/Mode

This strategy can be applied on a feature which has numeric data like the age of a person or the ticket fare. We can calculate the mean, median or mode of the feature and replace it with the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns. Replacing with the above three approximations are a statistical approach of handling the missing values. This method is also called as leaking the data while training. Another way is to approximate it with the deviation of neighbouring values. This works better if the data is linear.

```
data['Age'].isnull().sum()
```

```
177
```

```
data['Age'].mean()
```

```
29.69911764705882
```

```
data['Age'].replace(np.NaN , data['Age'].mean()).head(15)
```

```
0     22.000000
1     38.000000
2     26.000000
3     35.000000
4     35.000000
5     29.699118
6     54.000000
7      2.000000
8     27.000000
9     14.000000
10     4.000000
11    58.000000
12    20.000000
13    39.000000
14    14.000000
Name: Age, dtype: float64
```

Replaced with mean

To replace it with median and mode we can use the following to calculate the same:

```
data['Age'].median()
```

```
28.0
```

```
data['Age'].mode()
```

```
0    24.0
dtype: float64
```

Pros:

- This is a better approach when the data size is small
- It can prevent data loss which results in removal of the rows and columns

Cons:

- Imputing the approximations add variance and bias
- Works poorly compared to other multiple-imputations method

3. Assigning An Unique Category

A categorical feature will have a definite number of possibilities, such as gender, for example. Since they have a definite number of classes, we can assign another class for the missing values. Here, the features Cabin and Embarked have missing values which can be replaced with a new category, say, U for 'unknown'. This strategy will add more information into the dataset which will result in the change of variance. Since they are categorical, we need to find one hot encoding to convert it to a numeric form for the algorithm to understand it. Let us look at how it can be done in Python:

```
data['Cabin'].head(10)
```

```
0     NaN
1     C85
2     NaN
3    C123
4     NaN
5     NaN
6     E46
7     NaN
8     NaN
9     NaN
Name: Cabin, dtype: object
```

```
data['Cabin'].fillna('U').head(10)
```

```
0       U
1     C85
2       U
3    C123
4       U
5       U
6     E46
7       U
8       U
9       U
Name: Cabin, dtype: object
```

Pros:

- Less possibilities with one extra category, resulting in low variance after one hot encoding — since it is categorical
- Negates the loss of data by adding an unique category

Cons:

- Adds less variance
- Adds another feature to the model while encoding, which may result in poor performance

4. Predicting The Missing Values

Using the features which do not have missing values, we can predict the nulls with the help of a machine learning algorithm. This method may result in better accuracy, unless a missing value is expected to have a very high variance. We will be using linear regression to replace the nulls in the feature 'age', using other available features. One can experiment with different algorithms and check which gives the best accuracy instead of sticking to a single algorithm.

```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()

data_with_null = data[['PassengerId','Pclass','Survived',
                       'SibSp','Parch','Fare','Age']].dropna()
data_without_null = data_with_null.dropna()
#All features except AGE
train_data_x = data_without_null.iloc[:,:6]
#Only AGE
train_data_y = data_without_null.iloc[:,6]
```

```
# Training with the available data
linreg.fit(train_data_x,train_data_y)

# Predict for the whole dataset and replace only the missing values later
test_data = data_with_null.iloc[:,:6]
age_predicted['Age'] = pd.DataFrame(linreg.predict(test_data))

#Lets replace only the missing values
data_with_null.Age.fillna(age_predicted.Age,inplace=True)
```

Pros:

- Imputing the missing variable is an improvement as long as the bias from the same is smaller than the omitted variable bias
- Yields unbiased estimates of the model parameters

Cons:

- Bias also arises when an incomplete conditioning set is used for a categorical variable

- Considered only as a proxy for the true values

5. Using Algorithms Which Support Missing Values

KNN is a machine learning algorithm which works on the principle of distance measure. This algorithm can be used when there are nulls present in the dataset. While the algorithm is applied, KNN considers the missing values by taking the majority of the K nearest values. In this particular dataset, taking into account the person's age, sex, class etc, we will assume that people having same data for the above mentioned features will have the same kind of fare.

Unfortunately, the SciKit Learn library for the K – Nearest Neighbour algorithm in Python does not support the presence of the missing values.

Another algorithm which can be used here is RandomForest. This model produces a robust result because it works well on non-linear and the categorical data. It adapts to the data structure taking into consideration of the high variance or the bias, producing better results on large datasets.

Pros:

- Does not require creation of a predictive model for each attribute with missing data in the dataset
- Correlation of the data is neglected

Cons:

- Is a very time consuming process and it can be critical in data mining where large databases are being extracted
- Choice of distance functions can be Euclidean, Manhattan etc. which is do not yield a robust result

Conclusion

Every dataset we come across will almost have some missing values which need to be dealt with. But handling them in an intelligent way and giving rise to robust models is a challenging task. We have gone through a number of ways in which nulls can be replaced. It is not necessary to handle a particular dataset in one single manner. One can use various methods on different features depending on how and what the data is about. Having a small domain knowledge about the data is important, which can give you an insight about how to approach the problem.

STEP 4: See the categorical values

```
In [8]: data.sample(10)
```

Out[8]:

number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	sales	salary
2	108	2	0	0	0	technical	medium
4	274	3	0	0	0	sales	low
4	262	6	0	1	0	accounting	medium
2	176	5	0	0	1	marketing	low
3	184	2	0	0	0	marketing	low
3	260	4	0	0	0	technical	high
5	265	3	0	0	0	IT	low
4	156	3	0	0	0	product_mng	medium
2	198	4	0	0	0	technical	medium
3	163	2	0	0	0	sales	high

```
In [ ]:
```

Since, machine learning models are based on Mathematical equations and you can intuitively understand that it would cause some problem if we can keep the Categorical data in the equations because we would only want numbers in the equations.

So, we need to encode the Categorical Variable…..

```
In [9]: # now we have to change the sales to the department and the salary to the numerical values.
        data.rename(columns={'sales':'department'},inplace=True)

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

```
In [10]: data.head()
```

Out[10]:

n	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	department	salary
53	2	157	3	0	1	0	sales	1
86	5	262	6	0	1	0	sales	2
88	7	272	4	0	1	0	sales	2
87	5	223	5	0	1	0	sales	1
52	2	159	3	0	1	0	sales	1

```
In [11]: print(data['department'].value_counts())
```

```
sales          4140
```

```
In [38]: data['department']=data['department'].map({'IT':1,'RandD':2,'accounting':3,'hr':4,'management':5,'marke
         index = data.index
         columns = data.columns
```

Step 5 : Splitting the data-set into Training and Test Set

In any Machine Learning model is that we're going to split data-set into two separate sets

1. Training Set

2. Test Set

Why we need splitting ?

Well here it's your algorithm model that is going to learn from your data to make predictions. Generally we split the data-set into 70:30 ratio or 80:20 what does it mean, 70 percent data take in train and 30 percent data take in test. However, this Splitting can be varies according to the data-set shape and size.

```
print('Shape of x:',x.shape,' Shape of y:', y.shape)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25, random_state=0)
print('Shape of x:',x_train.shape,' Shape of y:', y_train.shape)
```

```
Shape of x: (14999, 9)  Shape of y: (14999,)
Shape of x: (11249, 9)  Shape of y: (11249,)
```

6. DATA VISUALIZATION

Data visualization is the presentation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns. With interactive visualization, you can take the concept a step further by using technology to drill down into charts and graphs for more detail, interactively changing what data you see and how it's processed.

THE IMPORTANCE OF DATA VISUALIZATION

Without a doubt, data visualization is a crucial tool in today's data-focused business world. Here are nine ways they help:

1. ABSORB INFORMATION QUICKLY.

A picture is worth a thousand words. Or, in this case, a picture is worth thousands of lines of data. As data volume inevitably increases, visualization manages influxes of new information and makes it easy to find trends.

2. UNDERSTAND YOUR NEXT STEPS.

From these visual trends, you can easily understand your best next steps with less time and energy dedicated to data analysis. You can save hours of time by looking at the big picture instead of a thousand puzzle pieces.

3. CONNECT THE DOTS.

Data visualization doesn't just show patterns and trends. It also brings important but subtle correlations and relationships between business conditions into focus.

4. HOLD YOUR AUDIENCE'S INTERESTS LONGER.

Graphics built with your data replay a message quickly, before you lose interest. As people now have an attention span shorter than that of goldfish keeping interest is a crucial goal when sharing insights.

5. KICK THE NEED FOR DATA SCIENTISTS.

Data visualization makes data more accessible and less confusing. Just years ago, the only professionals who could understand company data worked in the IT department. Now, finance, sales, and marketing teams not only have an avid interest in what they're data tells them, but they have the means to actually go after the answers.

6. SHARE YOUR INSIGHTS WITH EVERYONE.

It makes data more shareable. Visualizations can be distributed among teams easily, and your teams will be much more receptive to an attractive visual than a massive Excel spreadsheet.

7. FIND THE OUTLIERS.

Data visualization quickly reveals the outliers in your data. As outliers tend to drag down data averages in the wrong direction, it's crucial to find and eliminate them from your analysis when they skew the results. Graphics quickly shed light on them, allowing you to understand why they're there and ignore them when necessary.

8. MEMORIZE THE IMPORTANT INSIGHTS.

Visuals help commit important concepts to memory. It's easier to remember and memorize a concept if we have a **graphic to focus on, not just words or line items.**

9. ACT ON YOUR FINDINGS QUICKLY.

Most importantly, data visualization allows you to make decisions faster. Using them, you can review your strategies quickly and make updates efficiently, helping you achieve success with fewer mistakes and greater speed.

Even as data gets more complicated, the key to understanding it isn't always more complex formulas (although that can help you get to actionable insights faster), it's sometimes the simplification of its presentation. Here are ten steps to setting your data visualization up for success:

10 ELEMENTS OF A SUCCESSFUL DATA VISUALIZATION

Here are ten components of a successful data visualization:

1. IT TELLS A VISUAL STORY.

Remember, graphics rose in popularity because they simplify a complicated message. You turn to data visualization when there's too much data to follow with words. Make sure your visualization conveys the message in an easily-understood way. It should tell a story.

2. IT'S EASY TO UNDERSTAND.

If your audience can't find the end message, your graphic isn't effective. If they start trying to follow your methods, you've made it too confusing. Make it so simple that they trust you.

3. IT'S TAILORED FOR YOUR TARGET AUDIENCE.

Some people prefer specific visuals. Finance may find comfort in traditional scatterplots and bar graphs. Sales and marketing may need something more creative, such as an infographic. Build what your audience favors to help them better understand the information you're presenting.

4. IT ANSWERS SPECIFIC QUESTIONS.

You analyze data, because you have questions. You make data visualizations because you want to get to these answers faster. If you skip that crucial first step, your graphics serve no purpose.

5. IT'S USER-FRIENDLY.

Data visualizations shouldn't exist in a vacuum. The best visuals can be quickly and easily updated with new data to keep analysis fresh and ongoing. You shouldn't have to build from the ground up every time.

4. IT'S USEFUL.

Don’t make data pretty just for the sake of making something pretty. It needs to have a core functionality. In the business world, data visualizations typically help professionals defend or improve upon their strategies.

5. IT’S HONEST.

Data can easily be twisted to show a false story. Take care with your methods and data scaling to responsibly and honestly represent what’s going on in your business.

6. IT’S SUCCINCT.

Avoid information overload. Show only what is important. Don’t try to stuff too many messages in one visual. Remember, the aim of data visualization is simplicity. You can always create more visuals if you need to tell a multi-faceted story.

7. IT PROVIDES CONTEXT.

Your graphic should include insights as to why the story is important. Data visualization in a vacuum is useless. Make sure you explain why your visual story matters.

8. IT DOESN’T REQUIRE IT.

IT is busy enough managing, cleaning, and protecting your company’s data. They shouldn’t be responsible for making it visual as well. Your typical sales, marketing, or finance professional should be able build data visualizations with your preferred tool.

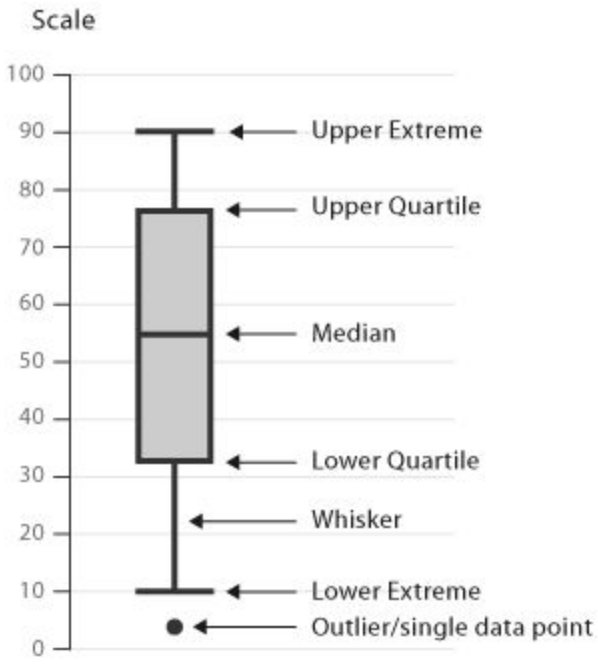
BOX PLOT VISUALIZATION

Description

A Box and Whisker Plot (or Box Plot) is a convenient way of visually displaying the data distribution through their quartiles.

The lines extending parallel from the boxes are known as the “whiskers”, which are used to indicate variability outside the upper and lower quartiles. Outliers are sometimes plotted as individual dots that are in-line with whiskers. Box Plots can be drawn either vertically or horizontally.

Although Box Plots may seem primitive in comparison to a Histogram or Density Plot, they have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets.



Here are the types of observations one can make from viewing a Box Plot:

What the key values are, such as: the average, median 25th percentile etc.

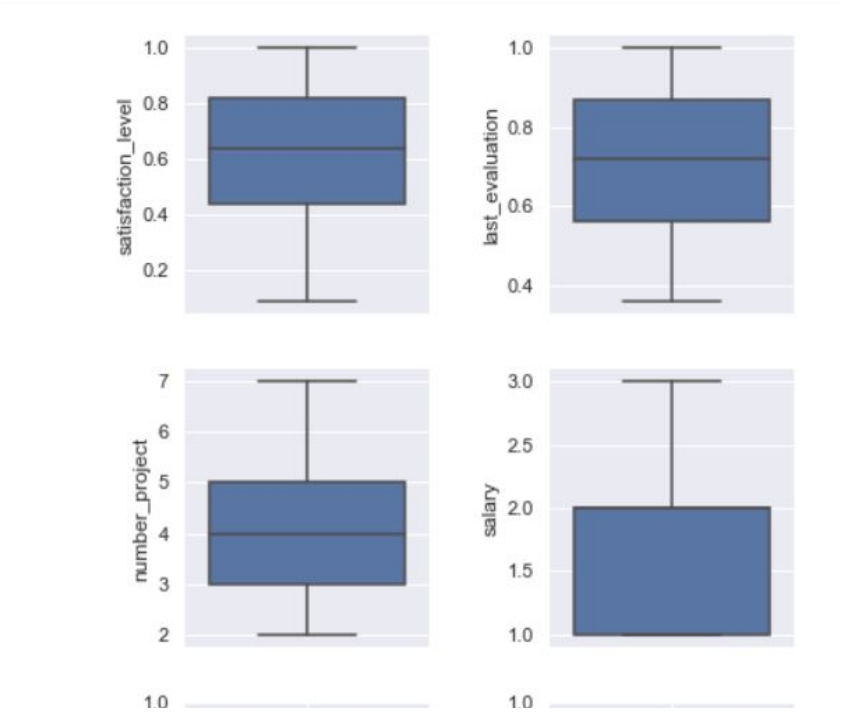
If there are any outliers and what their values are.

Is the data symmetrical.

How tightly is the data grouped.

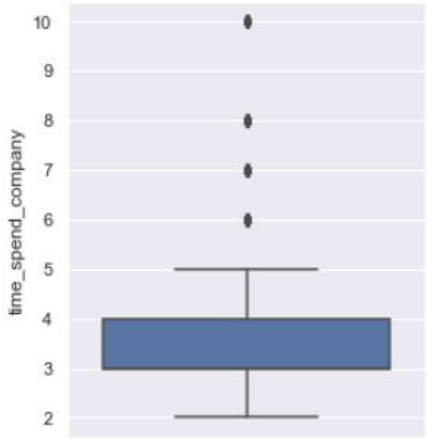
If the data is skewed and if so, in what direction.

Two of the most commonly used variation of Box Plot are: variable-width Box Plots and notched Box Plots.



Boxplot of salary, satisfaction_level ,last_evaluation ,number_projects

```
In [15]: plt.figure(figsize=(4,5))
         sns.boxplot( x= 'time_spend_company',  data=data, orient='v');
```



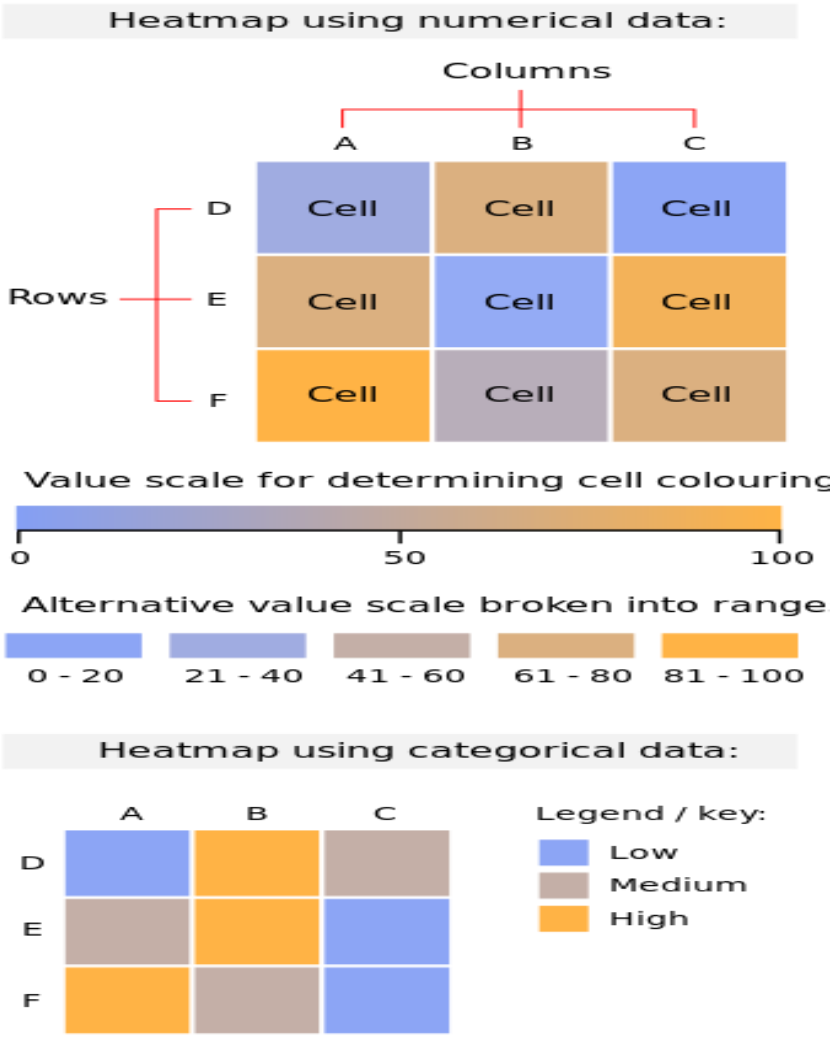
Box plot of time_spend_company

HEAT MAP VISUALIZATION

Description

Heatmaps visualise data through variations in colouring. When applied to a tabular format, Heatmaps are useful for cross-examining multivariate data, through placing variables in the rows and columns and colouring the cells within the table. Heatmaps are good for showing variance across multiple variables, revealing any patterns, displaying whether any variables are similar to each other, and for detecting if any correlations exist in-between them.

Typically, all the rows are one category (labels displayed on the left or right side) and all the columns are another category (labels displayed on the top or bottom). The individual rows and columns are divided into the subcategories, which all match up with each other in a matrix. The cells contained within the table either contain color-coded categorical data or numerical data, that is based on a color scale. The data contained within a cell is based on the relationship between the two variables in the connecting row and column.



A legend is required alongside a Heat map in order for it to be successfully read. Categorical data is color-coded, while numerical data requires a color scale that blends from one color to another, in order to represent the difference in high and low values. A selection of solid colors can be used to represent multiple value ranges (0-10, 11-20, 21-30, etc) or you can use a gradient scale for a single range (for example 0 - 100) by blending two or more colors together.

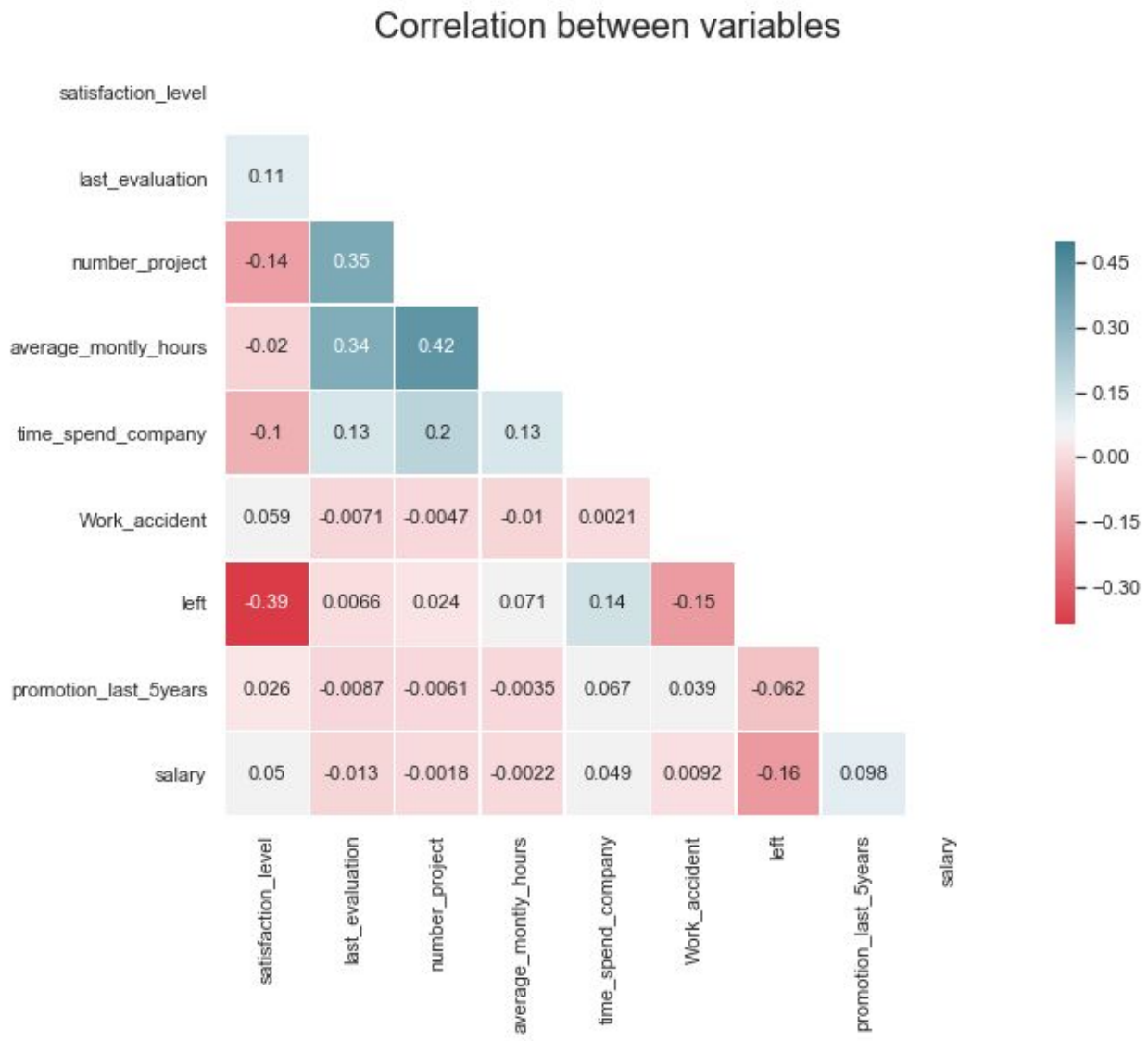
Because of their reliance on color to communicate values, Heatmaps are a chart better suited to displaying a more generalized view of numerical data, as it's harder to accurately tell the differences between color shades and to extract specific data points from (unless of course, you include the raw data in the cells).

Heatmaps can also be used to show the changes in data over time if one of the rows or columns are set to time intervals. An example of this would be to use a Heatmap to compare the temperature changes across the year in multiple cities, to see where's the hottest or coldest places. So the rows could list the cities to compare, the columns contain each month and the cells would contain the temperature values.

Heat maps represent values in a matrix as colors. Traditionally, heat maps have been used to indicate the level of activity in different systems. For example, a load test result can represent requests to different parts of the application as a heat map. The heat map appears as a mass of colors chosen from a color scheme with gradients from one color to the other.

Visualizing the data using heat map

```
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, yticklab
           cbar_kws={'shrink': .5}, ax=ax)
ax.set_title('Correlation between variables', fontsize=20);
```



Heat map visualization

COUNT PLOT VISUALIZATION

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for bar plot, so you can compare counts across nested variables.

Input data can be passed in a variety of formats, including:

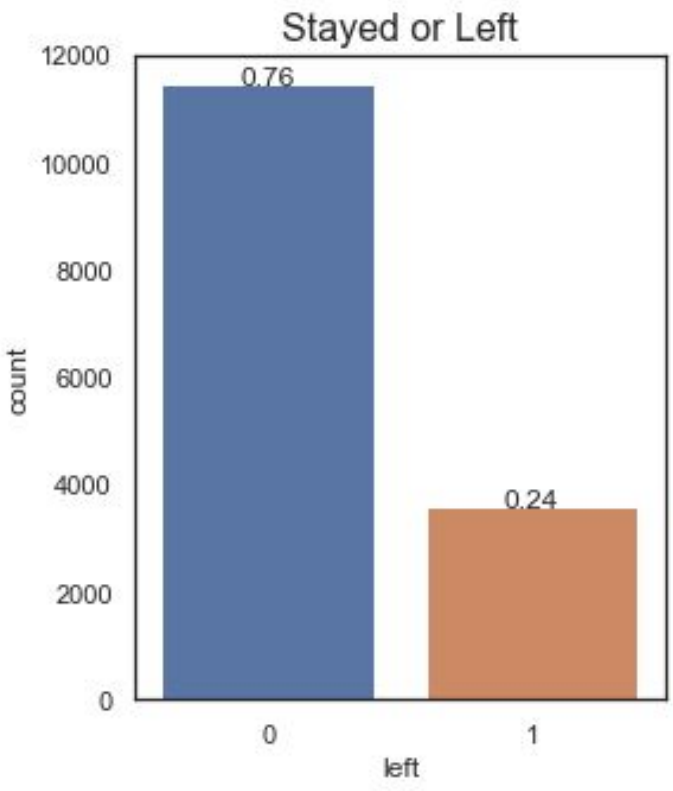
- Vectors of data represented as lists, numpy arrays, or pandas Series objects passed directly to the x, y, and/or hue parameters.

- A “long-form” DataFrame, in which case the x, y, and hue variables will determine how the data are plotted.
- A “wide-form” DataFrame, such that each numeric column will be plotted.
- An array or list of vectors.

In most cases, it is possible to use numpy or Python objects, but pandas objects are preferable because the associated names will be used to annotate the axes. Additionally, you can use Categorical types for the grouping variables to control the order of plot elements.

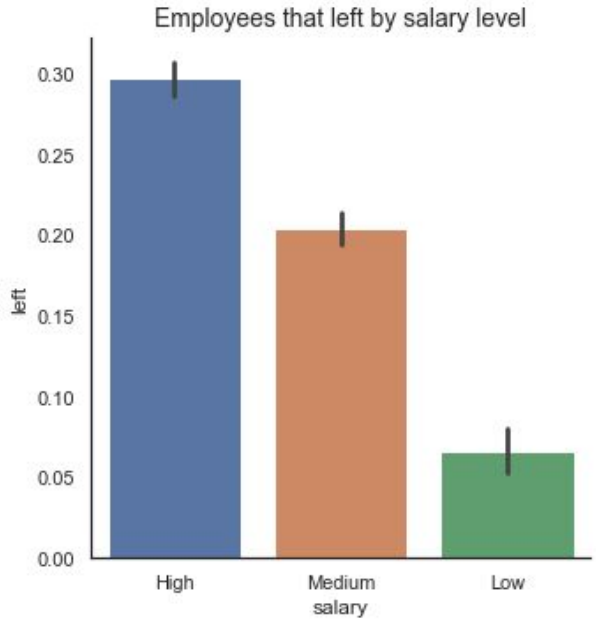
This function always treats one of the variables as categorical and draws data at ordinal positions (0, 1, … n) on the relevant axis, even when the data has a numeric or date type.

In [19]:
```
# the plot show the amount of employes that staysed 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=16)
```

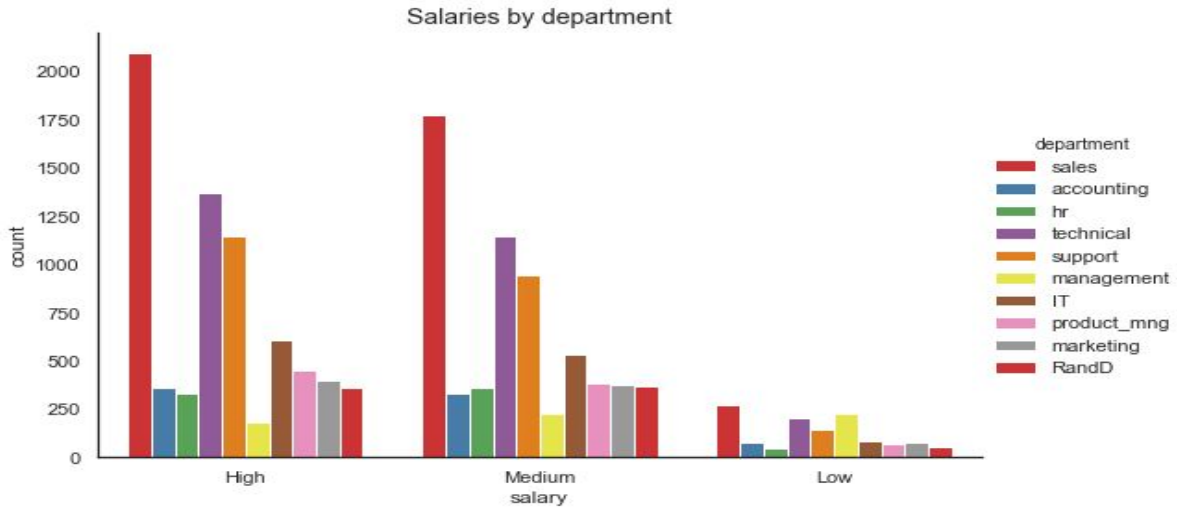


Count plot of employees who stayed and left the company

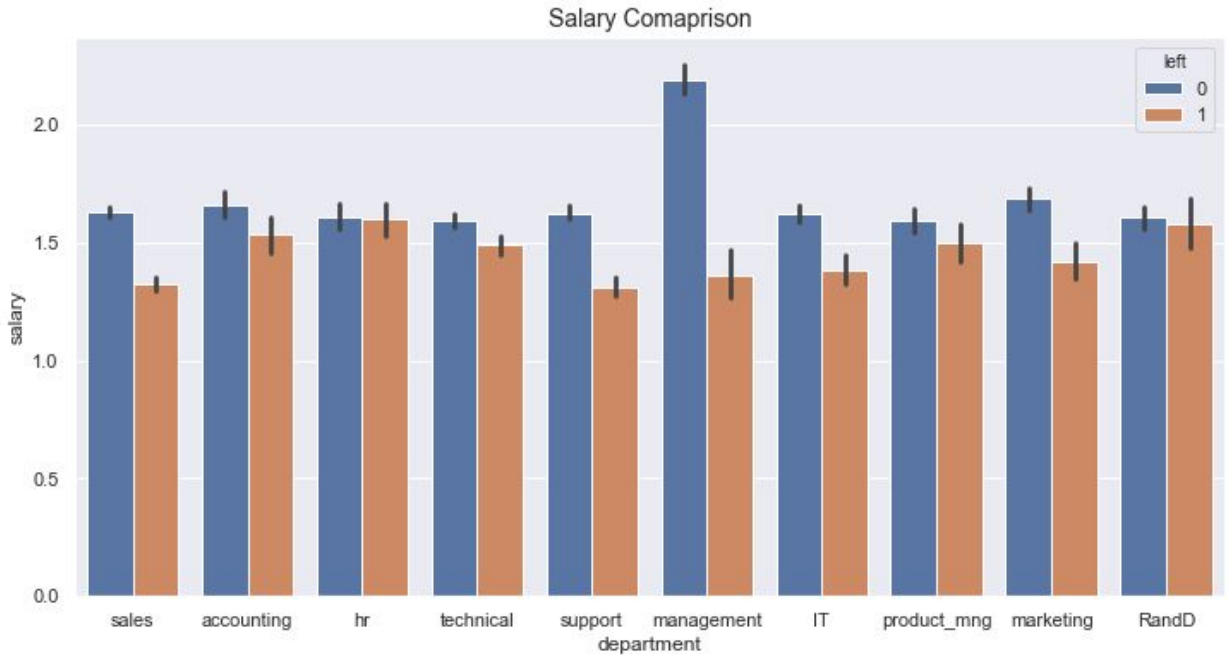
```
In [20]: j = sns.factorplot(x='salary', y='left', kind='bar', data=data)
         plt.title('Employees that left by salary level', fontsize=14)
         j.set_xticklabels(['High', 'Medium', 'Low']);
```



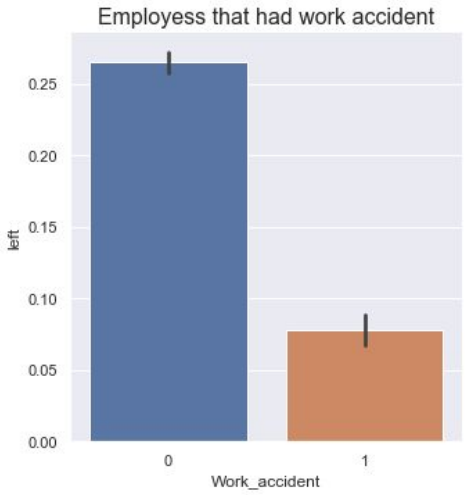

```
In [21]: h = sns.factorplot(x = 'salary', hue='department', kind ='count', size = 5,aspect=1.5, data=data, palette='Set1' )
         plt.title("Salaries by department", fontsize=14)
         h.set_xticklabels(['High', 'Medium', 'Low']);
```



In [22]:
```
sns.set()
plt.figure(figsize=(12,6))
sns.barplot(x='department',y='salary',hue='left', data=data)
plt.title('Salary Comaprison', fontsize=14);
```



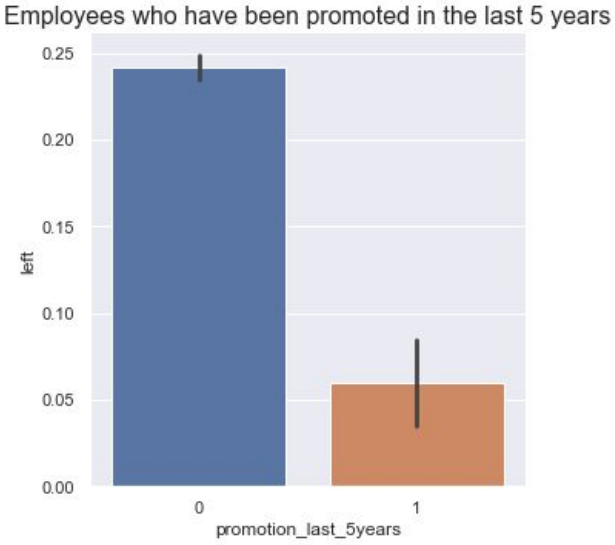
In [23]:
```
#second hypothesis

sns.factorplot(x='Work_accident',y='left',kind='bar', data=data)
plt.title('Employess that had work accident',fontsize=16);
```



In [24]:
```
print(data.Work_accident.sum())
print(data.Work_accident.mean())
print((data[data['left']==1]['Work_accident']).sum())
```

```
2169
0.1446096406427095
169
```

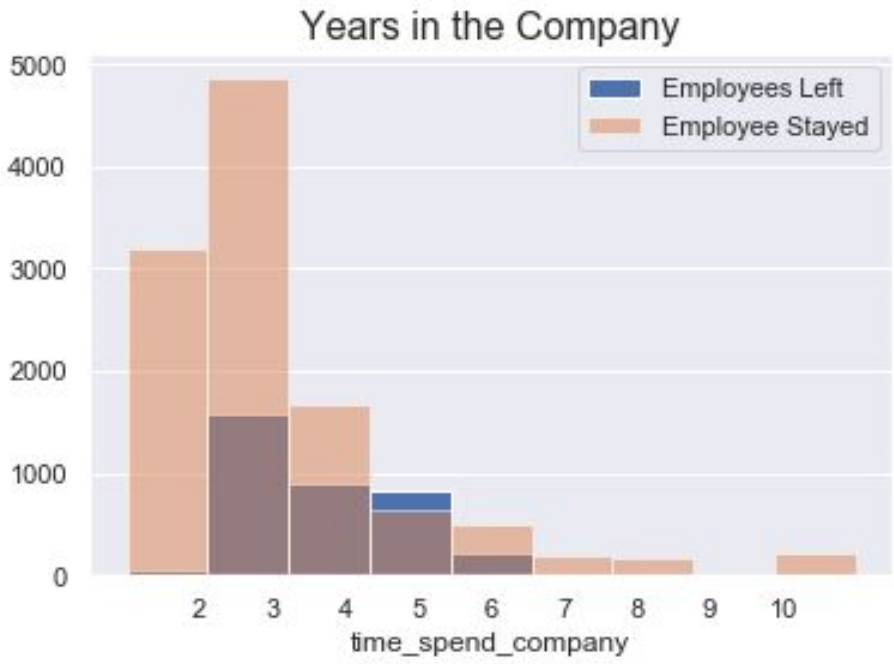
In [25]:
```
#third hypothesis
sns.factorplot(x='promotion_last_5years', y='left', kind='bar', data=data)
plt.title('Employees who have been promoted in the last 5 years', fontsize=16);
```



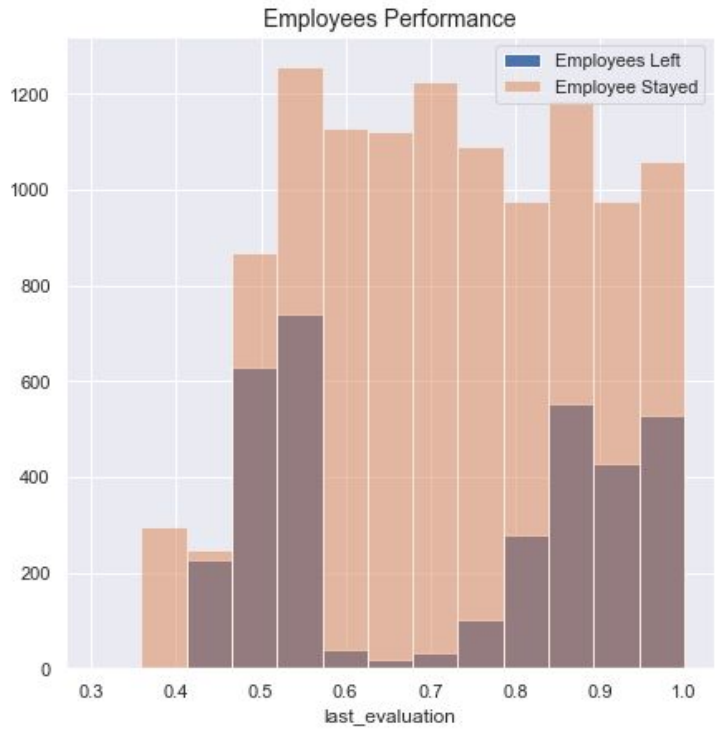
In [26]:
```
print(data.promotion_last_5years.sum())
print(data.promotion_last_5years.mean())
```

```
319
0.021268084538969265
```

In [27]:
```
plt.figure()
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=16)
plt.legend(loc='best');
```

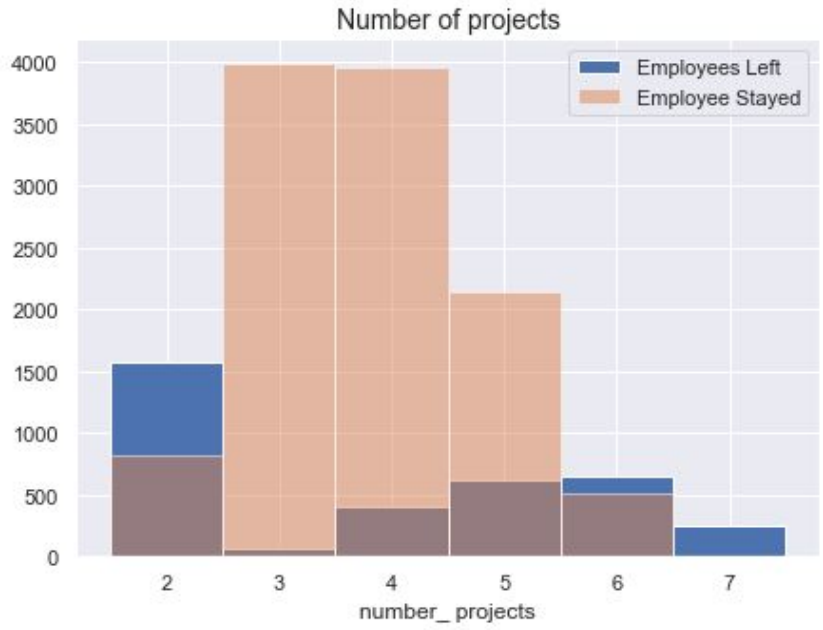


In [28]:
```
plt.figure(figsize =(7,7))
bins = np.linspace(0.305, 1.0001, 14)
plt.hist(data[data['left']==1]['last_evaluation'], bins=bins, alpha=1, label='Employees Left')
plt.hist(data[data['left']==0]['last_evaluation'], bins=bins, alpha = 0.5, label = 'Employee Stayed')
plt.title('Employees Performance', fontsize=14)
plt.xlabel('last_evaluation')
plt.legend(loc='best');
```

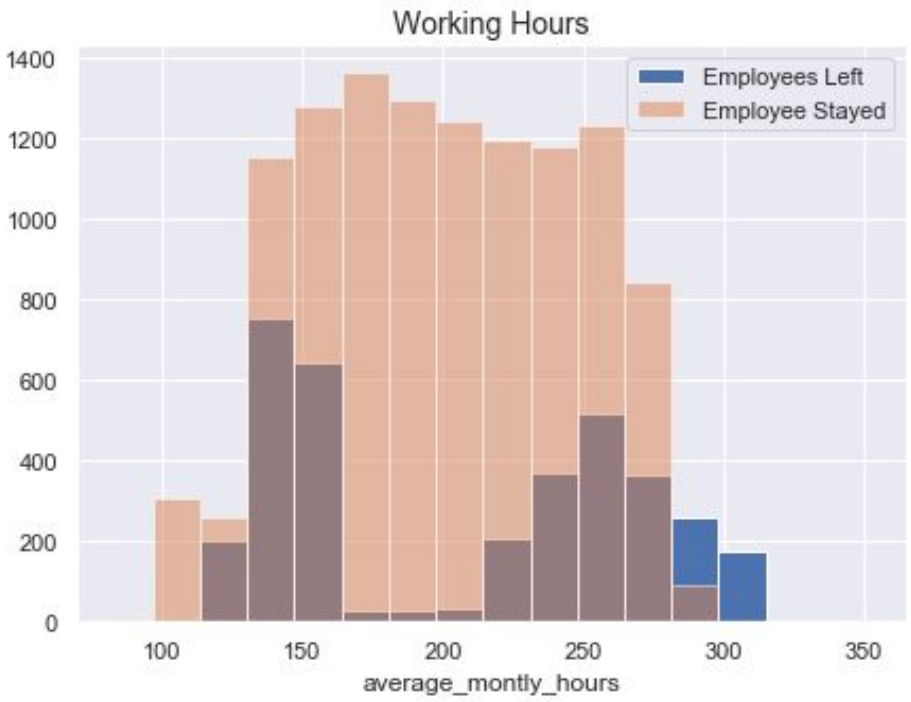


In [29]:
```
poor_performance_left = data[(data.last_evaluation <= 0.62) & (data.number_project == 2) & (data.left == 1)]
print('poor_performance_left:',len(poor_performance_left))

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
```

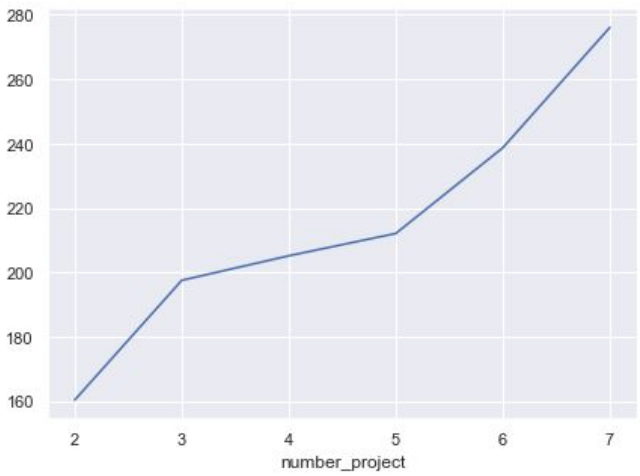


In [30]:
```
plt.figure(figsize =(7,5))
bins = np.linspace(80,315, 15)
plt.hist(data[data['left']==1]['average_montly_hours'], bins=bins, alpha=1, label='Employees Left')
plt.hist(data[data['left']==0]['average_montly_hours'], bins=bins, alpha = 0.5, label = 'Employee Staye
plt.title('Working Hours', fontsize=14)
plt.xlabel('average_montly_hours')
plt.xlim((70,365))
plt.legend(loc='best');
```



In [31]:
```
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
```



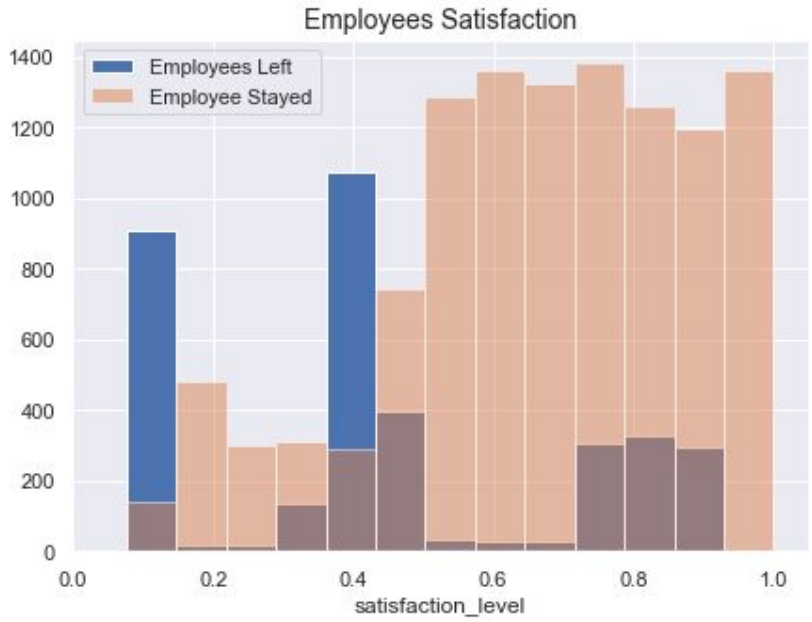
In [32]:

```
work_less_hours_left = data[(data.average_montly_hours < 200) & (data.number_project == 2) & (data.left
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
print('work_more_hours_left:',len(work_more_hours_left))
```

```
work_less_hours_left: 1535
work_more_hours_left: 1225
```

In [33]:

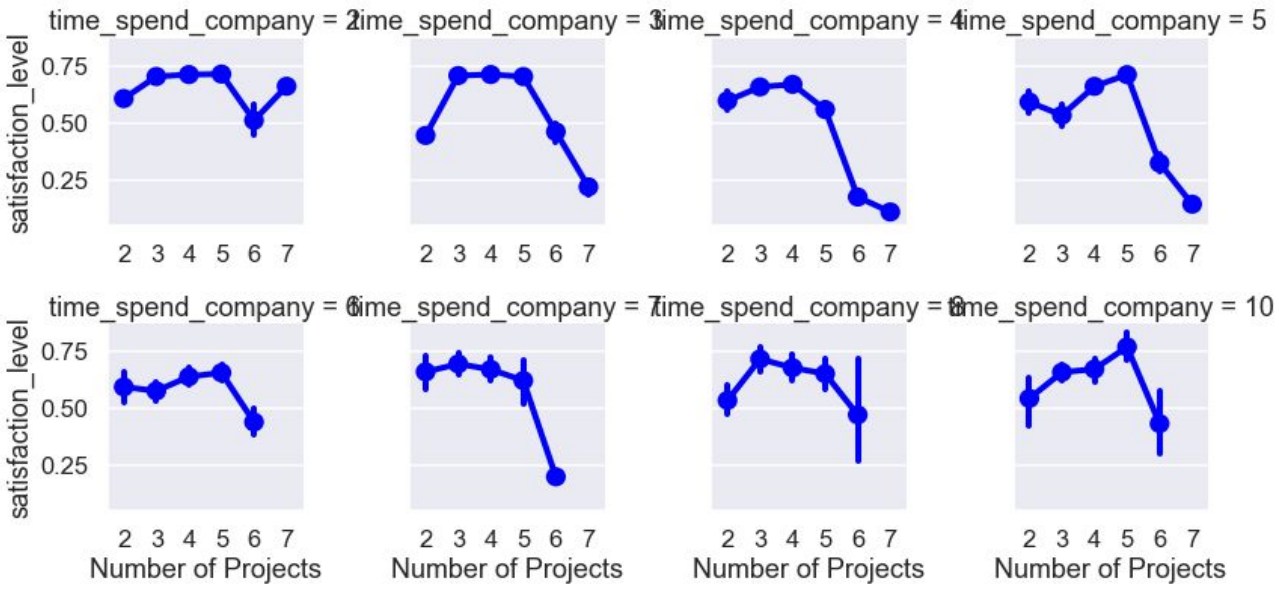
```
plt.figure(figsize =(7,5))
bins = np.linspace(0.006,1.000, 15)
plt.hist(data[data['left']==1]['satisfaction_level'], bins=bins, alpha=1, label='Employees Left')
plt.hist(data[data['left']==0]['satisfaction_level'], bins=bins, alpha = 0.5, label = 'Employee Stayed'
plt.title('Employees Satisfaction', fontsize=14)
plt.xlabel('satisfaction_level')
plt.xlim((0,1.05))
plt.legend(loc='best');
```



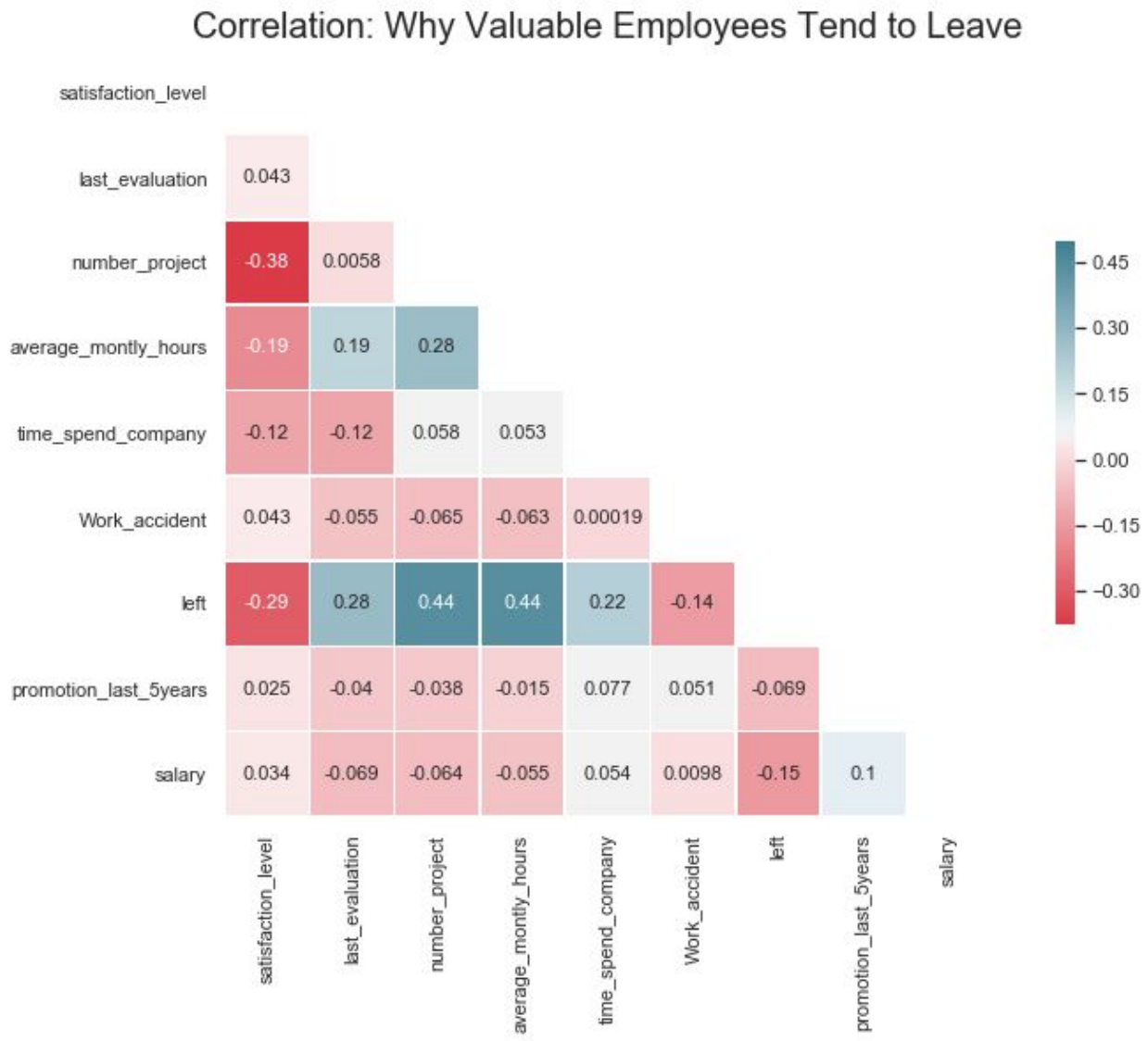
In [34]:
```
groupby_time_spend = data.groupby('time_spend_company').mean()
groupby_time_spend['satisfaction_level']
```

Out[34]:
```
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
```

In [35]:
```
sns.set()
sns.set_context("talk")
ax = sns.factorplot(x="number_project", y="satisfaction_level", col="time_spend_company",col_wrap=4, si
ax.set_xlabels('Number of Projects');
```



In [36]:
```
func_living = data[(data.last_evaluation >= 0.70) | (data.time_spend_company >=4) | (data.number_projec

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
           cbar_kws={'shrink': .5}, ax=ax)
ax.set_title('Correlation: Why Valuable Employees Tend to Leave', fontsize=20);
```



So we can now see the problem with highly evaluated employees who leave.

1. They have lower satisfaction level

2. They have more number of projects

3. They have higher monthly hours

4. They have also spent more time in company

5. They have lower salary

6. They have not been promoted in the last 5 years

8. EVALUATION OF ALGORITHMS

To make predictions on whether in future an employee will leave the company or not, the dataset has been trained using various algorithms.

The algorithms have been evaluated using confusion matrix, classification report and ROC curve.

CONFUSION MATRIX

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Here is an example of confusion matrix for a binary classifier (though it can easily be extended to the case of more than two classes):

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
- In reality, 105 patients in the sample have the disease, and 60 patients do not.

Let's now define the most basic terms, which are whole numbers (not rates):

- true positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.
- true negatives (TN): We predicted no, and they don't have the disease.
- false positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- false negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

- Accuracy: Overall, how often is the classifier correct?
    - (TP+TN)/total = (100+50)/165 = 0.91
- Misclassification Rate: Overall, how often is it wrong?
    - (FP+FN)/total = (10+5)/165 = 0.09
    - equivalent to 1 minus Accuracy
    - also known as "Error Rate"
- True Positive Rate: When it's actually yes, how often does it predict yes?
    - TP/actual yes = 100/105 = 0.95

  - also known as "Sensitivity" or "Recall"
- False Positive Rate: When it's actually no, how often does it predict yes?
  - FP/actual no = 10/60 = 0.17
- True Negative Rate: When it's actually no, how often does it predict no?
  - TN/actual no = 50/60 = 0.83
  - equivalent to 1 minus False Positive Rate
  - also known as "Specificity"
- Precision: When it predicts yes, how often is it correct?
  - TP/predicted yes = 100/110 = 0.91
- Prevalence: How often does the yes condition actually occur in our sample?
  - actual yes/total = 105/165 = 0.64

CLASSIFICATION REPORT

Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model. For our model, we have got 0.803 which means our model is approx. 80% accurate.

Accuracy = TP+TN/TP+FP+FN+TN

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Precision = TP/TP+FP

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? We have got recall of 0.631 which is good for this model as it’s above 0.5.

Recall = TP/TP+FN

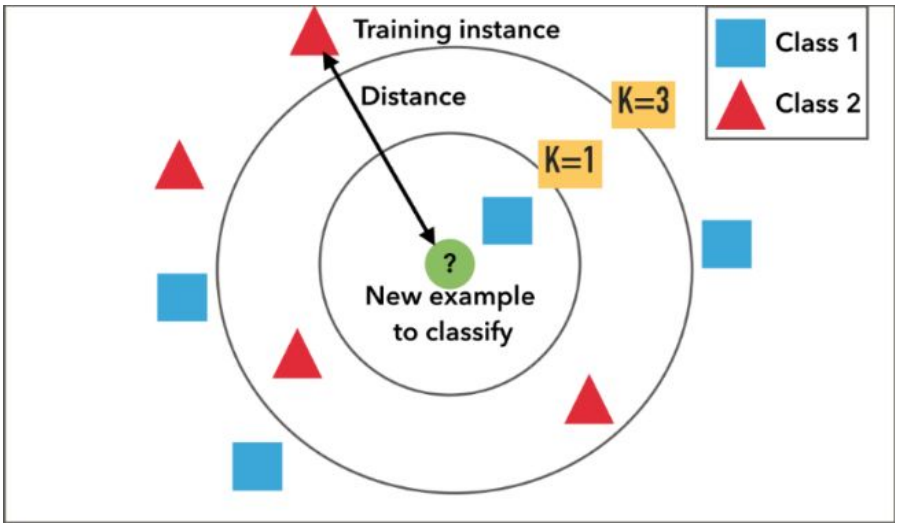
F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. In our case, F1 score is 0.701.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

1. K Nearest Neighbor

KNN is also a lazy algorithm (as opposed to an *eager* algorithm). What this means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase* or it is very minimal. This also means that the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. To be more exact, all (or most) the training data is needed during the testing phase.

KNN Algorithm is based on feature similarity: How closely out-of-sample features resemble our training set determines how we classify a given data point:



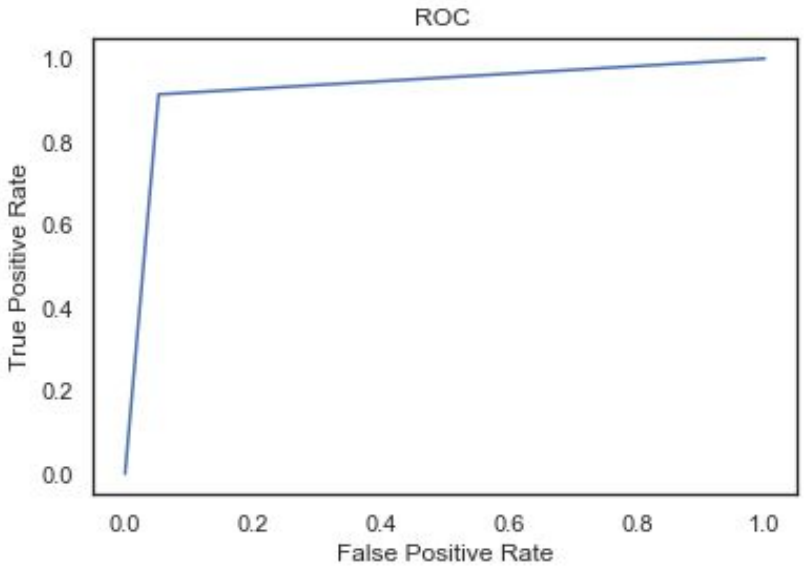
Example of k-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner

circle. If, for example k = 5 it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle).

KNN can be used for classification — the output is a class membership (predicts a class — a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for regression — output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

```
[1 0 0 ... 1 0 0]
Confusion Matrix :
[[2730  151]
 [  75  794]]
report              precision    recall  f1-score   support

          0       0.97      0.95      0.96      2881
          1       0.84      0.91      0.88       869

avg / total       0.94      0.94      0.94      3750

Accuracy 0.9397333333333333
```

Classification report for KNN



ROC curve for KNN

2. Random Forest Classifier

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object.

In Laymen's term,

Suppose training set is given as : [X1, X2, X3, X4] with corresponding labels as [L1, L2, L3, L4], random forest may create three decision trees taking input of subset for example,

1. [X1, X2, X3]
2. [X1, X2, X4]
3. [X2, X3, X4]

So finally, it predicts based on the majority of votes from each of the decision trees made.

This works well because a single decision tree may be prone to a noise, but aggregate of many decision trees reduce the effect of noise giving more accurate results.

The subsets in different decision trees created may overlap

Alternative implementation for voting

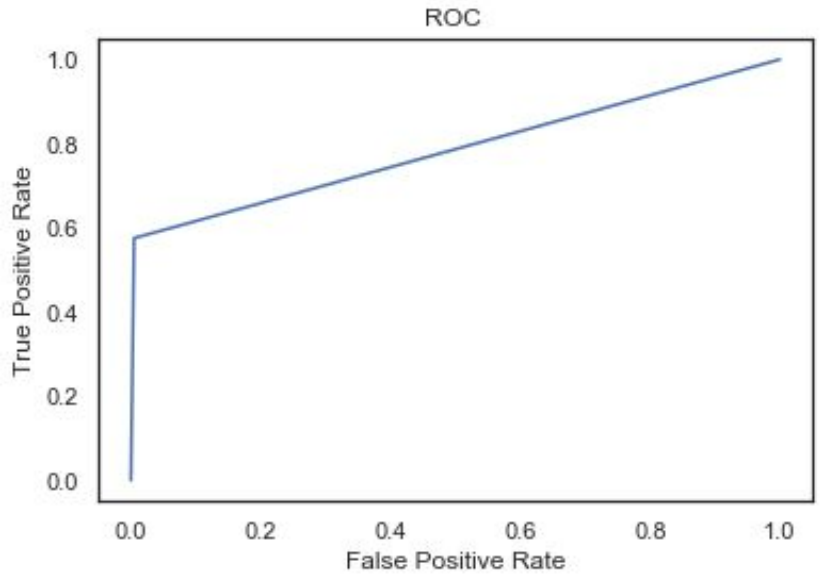
Alternatively, the random forest can apply weight concept for considering the impact of result from any decision tree. Tree with high error rate are given low weight value and vise versa. This would increase the decision impact of trees with low error rate.

Basic Parameters

Basic parameters to Random Forest Classifier can be total number of trees to be generated and decision tree related parameters like minimum split, split criteria etc.

```
[1 0 0 ... 0 0 0]
Confusion Matrix :
[[2867   14]
 [ 369  500]]
report              precision    recall  f1-score   support

          0       0.89      1.00      0.94      2881
          1       0.97      0.58      0.72       869

avg / total       0.91      0.90      0.89      3750

Accuracy 0.8978666666666667
```

Classification Report for Random Forest Classifier



ROC Curve for random forest Classifier

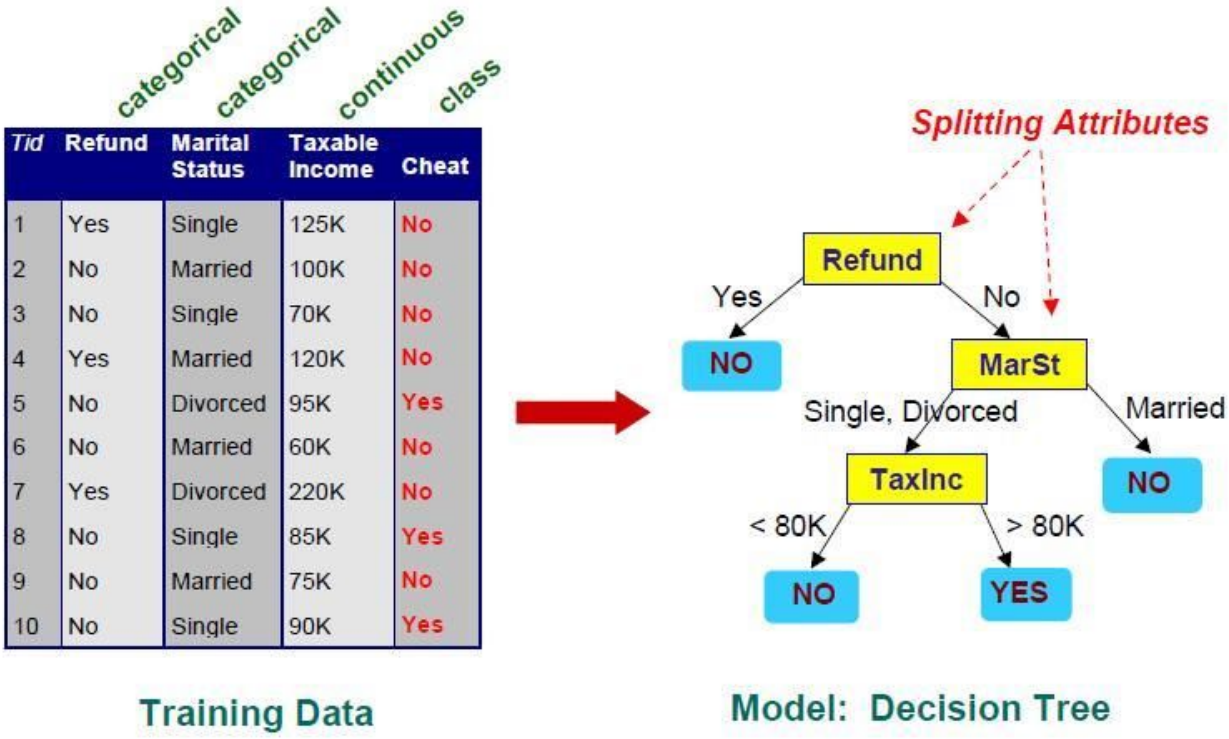
3. Decision Tree Classifier

The classification technique is a systematic approach to build classification models from an input dat set. For example, decision tree classifiers, rule-based classifiers, neural networks, support vector machines, and naive Bayes classifiers are different technique to solve a classification problem. Each technique adopts a learning algorithm to identify a model that best fits the relationshio between the attribute set and class label of the input data. Therefore, a key objective of the learning algorithm is to build prdictive model that accurately predict the class labels of previously unkonw records.

Decision Tree Classifier is a simple and widely used classification technique. It applies a straitforward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time time it receive an answer, a follow-up question is asked until a conclusion about the calss label of the record is reached.

Decision Tree Based Method

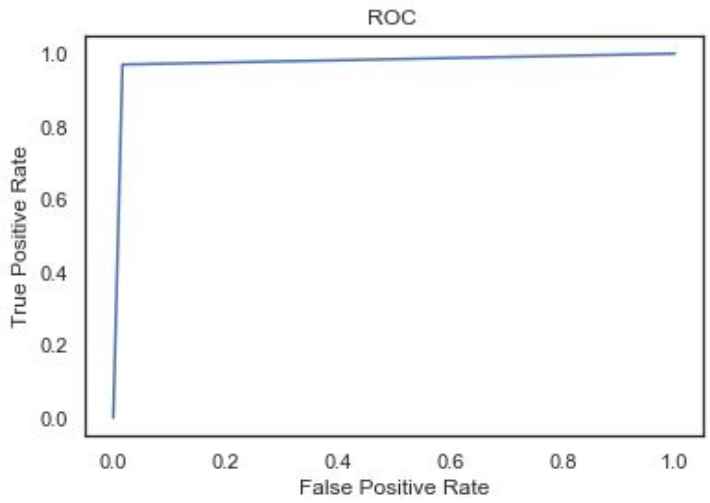
The decision tree classifiers organized a series of test questions and conditions in a tree structure. The following figure shows a example decision tree for predictin whether the person cheats. In the decision tree, the root and internal nodes contain attribute test conditions to separate recordes that have different characteristics. All the terminal node is assigned a class lable Yes or No.



Once the decision tree has been constructed, classifying a test record is straightforward. Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test. It then lead us either to another internal node, for which a new test condition is applied, or to a leaf node. When we reach the leaf node, the class lable associated with the leaf node is then assigned to the record, As shown in the follwoing figure [1], it traces the path in the decision tree to predict the class label of the test record, and the path terminates at a leaf node labeled NO.

```
[1 0 0 ... 1 0 0]
Confusion Matrix :
[[2834   47]
 [  26  843]]
report              precision    recall  f1-score   support

          0       0.99      0.98      0.99      2881
          1       0.95      0.97      0.96       869

avg / total       0.98      0.98      0.98      3750

Accuracy 0.9805333333333334
```

Classification Report for Decision tree Classifier



ROC Curve for Decision tree classifier

4. Gradient Boosting Classifier

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Gradient boosting involves three elements:

1. A loss function to be optimized.

2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

1. Loss Function

The loss function used depends on the type of problem being solved.

It must be differentiable, but many standard loss functions are supported and you can define your own.

For example, regression may use a squared error and classification may use logarithmic loss.

A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

2. Weak Learner

Decision trees are used as the weak learner in gradient boosting.

Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and "correct" the residuals in the predictions.

Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss.

Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels.

It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes.

This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

3. Additive Model

Trees are added one at a time, and existing trees in the model are not changed.

A gradient descent procedure is used to minimize the loss when adding trees.

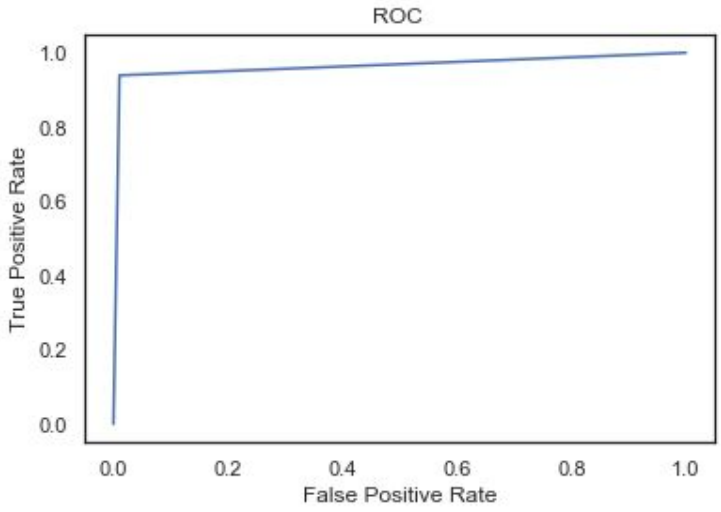
Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.

Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss.

Generally this approach is called functional gradient descent or gradient descent with functions.

```
[1 0 0 ... 1 0 0]
Confusion Matrix :
[[2851   30]
 [  53  816]]
report              precision    recall  f1-score   support

          0       0.98      0.99      0.99      2881
          1       0.96      0.94      0.95       869

avg / total       0.98      0.98      0.98      3750

Accuracy 0.9778666666666667
```

Classification Report for Gradient Boosting Classifier

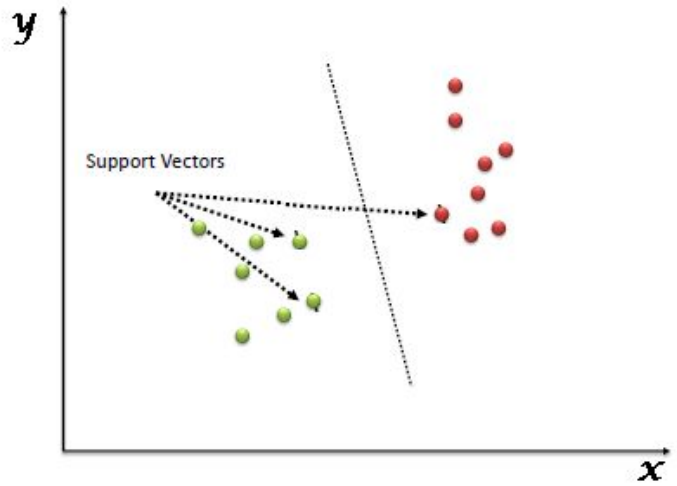


ROC Curve for Gradient Boosting Classifier

5. SVM Classifier

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular

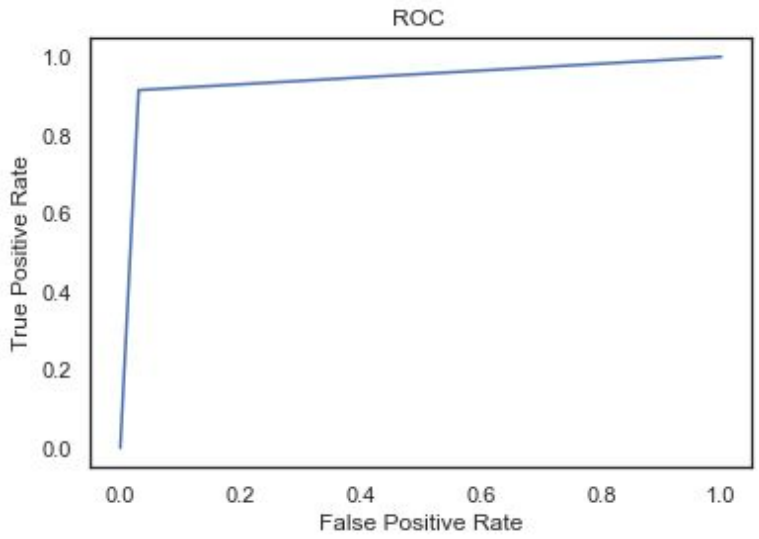
coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).



Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

```
[1 0 0 ... 1 0 0]
Confusion Matrix :
[[2793   88]
 [  74  795]]
report              precision    recall  f1-score   support

          0       0.97      0.97      0.97      2881
          1       0.90      0.91      0.91       869

avg / total       0.96      0.96      0.96      3750

Accuracy 0.9568
```

Classification Report for SVM



ROC Curve for SVM

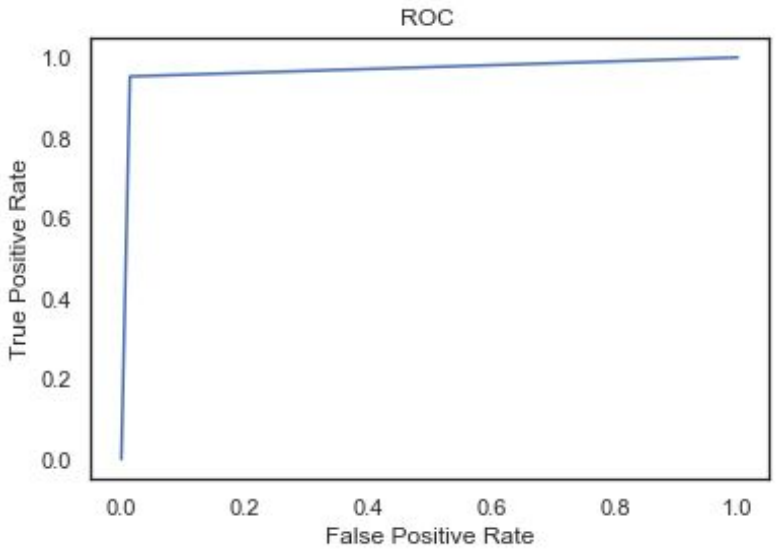
6. **Ada Boost Classifier**

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gödel Prize for their work. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset, AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier.[1][2] When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

```
[1 0 0 ... 1 1 0]
Confusion Matrix :
[[2841   40]
 [  41  828]]
report              precision    recall  f1-score   support

          0       0.99      0.99      0.99      2881
          1       0.95      0.95      0.95       869

avg / total       0.98      0.98      0.98      3750

Accuracy 0.9784
```

Classification Report for Adaboost Classifier



ROC Curve for AdaBoost Classifier

9. CONCLUSION

After visualizing and analyzing the data it can be inferred that

The problem with highly evaluated employees who leave are:

1. They have lower satisfaction level

2. They have more number of projects

3. They have higher monthly hours

4. They have also spent more time in company

5. They have lower salary

6. They have not been promoted in the last 5 years

Comparison of the algorithms:

ALGORITHM	ACCURACY	F1-SCORE	PRECISION	RECALL
KNN CLASSIFIER	0.9397	0.94	0.94	0.94
RANDOM FOREST CLASSIFIER	0.8978	0.89	0.90	0.91
DECISION TREE CLASSIFIER	0.9805	0.98	0.98	0.98
GRADIENT BOOSTING CLASSIFIER	0.978	0.98	0.98	0.98
SVM CLASSIFIER	0.9568	0.96	0.96	0.96
ADABOOST CLASSIFIER	0.9784	0.98	0.98	0.98

FUTURE SCOPE

Data and technology enablement has advanced rapidly in the last decade and will continue to do so. Given the huge amount of workforce data available in organizations today HR analytics has huge potential to deliver clear business benefits, but many HR leaders are at a loss as to where to begin.

A recent survey of CHROs (Chief Human Resource Officers) of companies found that the next big investment in the coming 6-18 months in HR will be in HR Analytics. Most companies feel that workforce analytics is their strategic priority.

Globally, 78 percent companies (employing 1000 and above employees) rated HR analytics as urgent, but only 19 percent companies felt they were equipped to handle this as compared to 81 percent in finance, 78 percent in operations and 58 percent in marketing and sales (Deloitte, 2014). Thus, HR is still playing catch up in the Analytics arena, though the scope is huge.

Currently, if we look at the statistics, 39% of the companies worldwide have data to understand the strengths and weaknesses of the employees which can be used for customizing their development programs, 38% use quantitative metrics for benchmarking but only 42% know how to extract meaningful insights from the data available to them.

In 3-5 years, 50-60% workforce in top companies would be temporary. Talent management would be one of the major challenges for the organizations. This includes the whole value chain from recruiting the right talent to nurturing and retaining it. Analytics allows organizations to maximize the investments they make in human capital.

According to these reports, this project can be further expanded on a web platform and can be used as an analytical tool to analyze various aspects of the people data which will benefit the company and enhance its productibility.

10. BIBLIOGRAPHY

- https://www.kaggle.com/rhuebner/human-resources-data-set
- https://en.wikipedia.org/wiki/AdaBoost
- https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
- https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/
- http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/decisionTree.html
- https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7
- https://www.slideshare.net/Centerline_Digital/the-importance-of-data-visualization