**CHAPTER- I**

**1. INTRODUCTION**

**ABSTRACT**

The data mining plays a role in Human Resource Management Systems (HRMS). A deep understanding of the knowledge hidden in Human Resource (HR) data is vital to a firm's competitive position and organisational decision making. Analysing the patterns and relationships in HR data is quite rare. The HR data is usually treated to answer queries. We show how data mining discovers and extracts useful patterns from this large data set to find observable patterns in HR. The project demonstrates the ability of data mining in improving the quality of the decision-making process in HRMS and gives propositions regarding whether data-mining capabilities should lead to increased performance to sustain competitive advantage.

The dataset we have choosen is about human resources. Human Resources are critical resources of any organiazation. Organizations spend huge amount of time and money to hire and nuture their employees. It is a huge loss for companies if employees leave, especially the key resources. So if HR can predict weather employees are at risk for leaving the company, it will allow them to identify the attrition risks and help understand and provide necessary support to retain those employees or do preventive hiring to minimize the impact to the orgranization.

**Data Description:**

We used [Human Resources Analytics Data](https://www.kaggle.com/ludobenistant/hr-analytics/data) from [kaggle](https://www.kaggle.com/). This HR data set is obtained from the results of a satisfaction survey the company has carried out on their employees in combination with other HR related records. It consists of 14999 rows and 10 columns. Each row is dedicated for a different employee. Out of 10, 8 columns are in numeric type, while the remaining 2 are in numeric values. Below you can find columns and their explanations, respectively.

* **1st Column:** Satisfaction level
* **2nd Column:** Last evaluation score
* **3rd Column:** Number of projects worked on (yearly basis)
* **4th Column:** Average monthly working hours
* **5th Column:** Time spent in the company (in years)
* **6th Column:** Whether they have had a work accident in the last 2 years
* **7th Column:** Whether they have had a promotion in the last 5 years
* **8th Column:** Departments
* **9th Column:** Salary
* **10th Column:** Whether the employee has left

All the data collected is 5 years whereas accident data belongs to 2 years. This HR database does not take into account the employees that have been fired, transferred or hired in the past year. Our objective is to make predictions about the probabilities that employees may leave their company and what to change to increase their satisfaction levels. We will try to give insights to make best employees more loyal.

**1.1 OVERVIEW:**

Here is an overview of what we are going to cover:

1. Installing the Python.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

* Separate out a validation dataset.
* Build 3 different models
* Select the best model.

**1.2 EXPECTED OUTCOME:**

The project focuses on weather employees are at risk for leaving the company, it will allow them to identify the attrition risks and help understand and provie necessary support to retain those employees or do preventive hiring to minimize the impact to the organization.by building models and finding the most important features which influence whether to leave the company

# 1.3 PROBLEM DEFINITION:

In data mining using humaran resource dataset we perform a different models to analyse which is best model to the hr to predict who will left the company. Here we perform the logistic regression model, decision tree classifier and random forest classifier.where predict which one is best and which feature plays a major role in different datas in the different models.

.

**CHAPTER-II**

**SYSTEM DESCRIPTION**

2.1 Data Mining:

**Data mining** is the process of discovering patterns in large [data sets](https://en.wikipedia.org/wiki/Data_set) involving methods at the intersection of [machine learning](https://en.wikipedia.org/wiki/Machine_learning), [statistics](https://en.wikipedia.org/wiki/Statistics), and [database systems](https://en.wikipedia.org/wiki/Database_system). Data mining is an [interdisciplinary](https://en.wikipedia.org/wiki/Interdisciplinary) subfield of [computer science](https://en.wikipedia.org/wiki/Computer_science) with an overall goal to extract information (with intelligent methods) from a data set and transform the information into a comprehensible structure for further use. Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD. Aside from the raw analysis step, it also involves database and [data management](https://en.wikipedia.org/wiki/Data_management) aspects, [data pre-processing](https://en.wikipedia.org/wiki/Data_pre-processing), [model](https://en.wikipedia.org/wiki/Statistical_model) and [inference](https://en.wikipedia.org/wiki/Statistical_inference) considerations, interestingness metrics, [complexity](https://en.wikipedia.org/wiki/Computational_complexity_theory) considerations, post-processing of discovered structures, [visualization](https://en.wikipedia.org/wiki/Data_visualization), and [online updating](https://en.wikipedia.org/wiki/Online_algorithm).

The term "data mining" is in fact a [misnomer](https://en.wikipedia.org/wiki/Misnomer), because the goal is the extraction of patterns and knowledge from large amounts of data, not the extraction (*mining*) of data itself It also is a [buzzword](https://en.wikipedia.org/wiki/Buzzword) and is frequently applied to any form of large-scale data or [information processing](https://en.wikipedia.org/wiki/Information_processing) ([collection](https://en.wikipedia.org/wiki/Data_collection), [extraction](https://en.wikipedia.org/wiki/Information_extraction), [warehousing](https://en.wikipedia.org/wiki/Data_warehouse), [analysis](https://en.wikipedia.org/wiki/Data_analysis), and statistics) as well as any application of [computer decision support system](https://en.wikipedia.org/wiki/Decision_support_system), including [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) (e.g., machine learning) and [business intelligence](https://en.wikipedia.org/wiki/Business_intelligence). The book *Data mining: Practical machine learning tools and techniques with Java* (which covers mostly machine learning material) was originally to be named just *Practical machine learning*, and the term *data mining* was only added for marketing reasons. Often the more general terms (*large scale*) [*data analysis*](https://en.wikipedia.org/wiki/Data_analysis)and [*analytics*](https://en.wikipedia.org/wiki/Analytics) – or, when referring to actual methods, *artificial intelligence* and *machine learning* – are more appropriate.

The actual data mining task is the semi-automatic or automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records ([cluster analysis](https://en.wikipedia.org/wiki/Cluster_analysis)), unusual records ([anomaly detection](https://en.wikipedia.org/wiki/Anomaly_detection)), and dependencies ([association rule mining](https://en.wikipedia.org/wiki/Association_rule_mining), [sequential pattern mining](https://en.wikipedia.org/wiki/Sequential_pattern_mining)). This usually involves using database techniques such as [spatial indices](https://en.wikipedia.org/wiki/Spatial_index). These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics). For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a [decision support system](https://en.wikipedia.org/wiki/Decision_support_system). Neither the data collection, data preparation, nor result interpretation and reporting is part of the data mining step, but do belong to the overall KDD process as additional steps.

1.Logistic regression :

Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score ([TRISS](https://en.wikipedia.org/wiki/TRISS)), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression. Many other medical scales used to assess severity of a patient have been developed using logistic regression. Logistic regression may be used to predict the risk of developing a given disease (e.g. [diabetes](https://en.wikipedia.org/wiki/Diabetes_mellitus); [coronary heart disease](https://en.wikipedia.org/wiki/Coronary_artery_disease)), based on observed characteristics of the patient (age, sex, [body mass index](https://en.wikipedia.org/wiki/Body_mass_index), results of various [blood tests](https://en.wikipedia.org/wiki/Blood_test), etc.). Another example might be to predict whether an Indian voter will vote BJP or Trinamool Congress or Left Front or Congress, based on age, income, sex, race, state of residence, votes in previous elections, etc. The technique can also be used in [engineering](https://en.wikipedia.org/wiki/Engineering), especially for predicting the probability of failure of a given process, system or product. It is also used in [marketing](https://en.wikipedia.org/wiki/Marketing) applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc. In [economics](https://en.wikipedia.org/wiki/Economics) it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application would be to predict the likelihood of a homeowner defaulting on a [mortgage](https://en.wikipedia.org/wiki/Mortgage). [Conditional random fields](https://en.wikipedia.org/wiki/Conditional_random_field), an extension of logistic regression to sequential data, are used in [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing).

2. Decision Tree:

Decision tree learning uses a [decision tree](https://en.wikipedia.org/wiki/Decision_tree) (as a [predictive model](https://en.wikipedia.org/wiki/Predictive_modelling)) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in [statistics](https://en.wikipedia.org/wiki/Statistics), [data mining](https://en.wikipedia.org/wiki/Data_mining) and [machine learning](https://en.wikipedia.org/wiki/Machine_learning). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, [leaves](https://en.wikipedia.org/wiki/Leaf_node) represent class labels and branches represent [conjunctions](https://en.wikipedia.org/wiki/Logical_conjunction) of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically [real numbers](https://en.wikipedia.org/wiki/Real_numbers)) are called regression trees.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and [decision making](https://en.wikipedia.org/wiki/Decision_making). In [data mining](https://en.wikipedia.org/wiki/Data_mining), a decision tree describes data (but the resulting classification tree can be an input for [decision making](https://en.wikipedia.org/wiki/Decision_making)). This page deals with decision trees in [data mining](https://en.wikipedia.org/wiki/Data_mining).

3.Random forest classifier:

Random forests or random decision forests are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks, that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of [overfitting](https://en.wikipedia.org/wiki/Overfitting) to their [training set](https://en.wikipedia.org/wiki/Test_set).

The first algorithm for random decision forests was created by [Tin Kam Ho](https://en.wikipedia.org/wiki/Tin_Kam_Ho) using the [random subspace method](https://en.wikipedia.org/wiki/Random_subspace_method), which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by [Leo Breiman](https://en.wikipedia.org/wiki/Leo_Breiman) and Adele Cutler, and "Random Forests" is their [trademark](https://en.wikipedia.org/wiki/Trademark). The extension combines Breiman's "[bagging](https://en.wikipedia.org/wiki/Bootstrap_aggregating)" idea and random selection of features, introduced first by Ho and later independently by Amit and [Geman](https://en.wikipedia.org/wiki/Donald_Geman) in order to construct a collection of decision trees with controlled variance.

**2.2: SYSTEM MODEL:**

System model basically specifies the detailed study of the various operations performed by the system and their relationship with in and outside the system. The key question in this phase is what all the problems there in the present system and what must be done to solve those problems. The success of the system depends largely on how clearly the problem is defined, thoroughly investigated and properly carried out. The analysis should provide the mechanism of problem understanding and also a frame work for its solution.

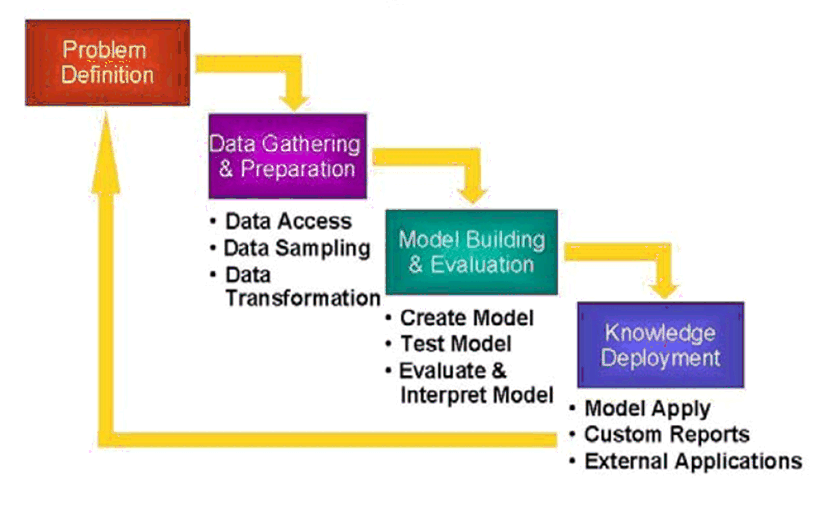
**Anaconda2-1.8.7 (Spyder 2.7)**

Anaconda® is a package manager, an environment manager, a Python distribution, and a collection of over 1,000+ open source packages. It is free and easy to install, and it offers free community support.

* Over 150 packages are automatically installed with Anaconda.
* Over 250 additional open source packages can be individually installed from the Anaconda repository with the conda install command.
* Thousands of other packages are available from Anaconda Cloud.
* You can download other packages using the pip install command that is installed with Anaconda.

You can also make your own custom packages using the conda build command, and you can share them with others by uploading them to Anaconda Cloud, PyPi or other repositories.

**2.3 WORKFLOW MODEL:**



**CHAPTER III**

**DEVELOPMENT ENVIRONMENT**

**3.1 HARDWARE CONFIGURATION:**

* Processor : INTEL CORE I3-5005U CPU @ 2.00GHz
* Hard Disk : 500 GB
* RAM : 4 GB
  1. **SOFTWARE CONFIGURATION:**
* Operating System : Windows 7
* Front End : Python 2.7\*
* IDE : Anaconda**2-1.8.7 (Spyder 3.2.8)**

**3.3. Language Specifications**

**3.3.1. Front End:**

**Spyder 3.2.8**

**Spyder is the Scientific Python Development Environment:**

* a powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features
* and a numerical computing environment thanks to the support of I Python (enhanced interactive Python interpreter) and popular Python libraries such as NumPy (linear algebra), SciPy (signal and image processing) or matplotlib (interactive 2D/3D plotting).

Spyder may also be used as a library providing powerful console-related widgets for your PyQt-based applications – for example, it may be used to integrate a debugging console directly in the layout of your graphical user interface.

Sublime Text 3 is one of the most complex integrated development environments (IDE) that can be used to build project involving software solutions, console apps, and graphical user interfaces. This IDE supports most of the programming languages frequently used by developers and webmasters: C/C++, ASP. NET, C#, Angular JS, PHP, Perl, Python, Ruby, Ruby on rails, SQL etc…

Users can develop Java script CSS, XML/XSLT, HTML/XHTML, PHP, Angular JS projects within visual studio. Not only can developers start their apps from scratch, but they can also modify their existing source code or look for syntax errors. Bugs can be fixed due to the built in debugger that can be used both as a source-level and as a machine level one, meant to support managed code ant native one developed in any supported programming language.

**CHAPTER-IV**

**APPENDIX**

**4.1 Source Code:**

**Import pandas as Pd**

**Import numpy as np**

**#loading the dataset:**

**hr\_df = pd.read\_csv ('F: \MINI PROJECT MCA\HR1.csv')**

**hr\_df.head ()**

satisfaction\_level last\_evaluation ... sales salary

0 0.38 0.53 ... sales low

1 0.80 0.86 ... sales medium

2 0.11 0.88 ... sales medium

3 0.72 0.87 ... sales low

4 0.37 0.52 ... sales low

[5 rows x 10 columns**]**

**hr\_df.columns**

Index ([u'satisfaction\_level', u'last\_evaluation', u'number\_project',

U’average\_montly\_hours', u'time\_spend\_company', u'Work\_accident',

u'left', u'promotion\_last\_5years', u'sales', u'salary'],

Dtype='object')

**#encoding categorical features:**

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

**categorical\_features = ['Work\_accident','promotion\_last\_5years', 'sales', 'salary']**

**#an utility function to create dummy variable**

**#def create\_dummies (df, colname):**

**def create\_dummies (df, colname) : #run it with full syntax to avoid eof**

**col\_dummies = pd.get\_dummies (df[colname], prefix=colname)**

**col\_dummies.drop (col\_dummies.columns [0], axis=1, inplace=True)**

**df = pd.concat([df, col\_dummies], axis=1)**

**df.drop (colname, axis = 1, inplace = True)**

**Return df**

**For c\_feature in categorical\_features:#run for fully**

**Hr\_df = create\_dummies ( hr\_df, c\_feature )**

**hr\_df.head()**

satisfaction\_level last\_evaluation ... salary\_low salary\_medium

0 0.38 0.53 ... 1 0

1 0.80 0.86 ... 0 1

2 0.11 0.88 ... 0 1

3 0.72 0.87 ... 1 0

4 0.37 0.52 ... 1 0

[5 rows x 19 columns]

**# Validating and Splitting the dataset**

**feature\_columns = hr\_df.columns.difference( ['left'] )**

**feature\_columns**

Index([u'Work\_accident\_1', u'average\_montly\_hours', u'last\_evaluation',

u'number\_project', u'promotion\_last\_5years\_1', u'salary\_low',

u'salary\_medium', u'sales\_RandD', u'sales\_accounting', u'sales\_hr',

u'sales\_management', u'sales\_marketing', u'sales\_product\_mng',

u'sales\_sales', u'sales\_support', u'sales\_technical',

u'satisfaction\_level', u'time\_spend\_company'],

dtype='object')

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

**train\_X, test\_X, train\_y, test\_y = train\_test\_split( hr\_df[feature\_columns],hr\_df['left'],test\_size = 0.2,random\_state = 42 )**

**hr\_left\_df = pd.DataFrame( hr\_df.left.value\_counts() )**

**hr\_left\_df**

left

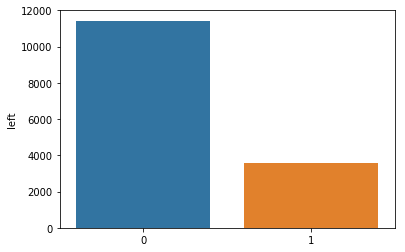
0 11428

1 3571

**import matplotlib as plt**

**import seaborn as sn**

**sn.barplot( hr\_left\_df.index, hr\_left\_df.left )**

****

**#Building Models**

**#Logistic Regression Model**

**from sklearn.linear\_model import LogisticRegression**

**logreg = LogisticRegression()**

**logreg.fit( train\_X, train\_y )**

**list( zip( feature\_columns, logreg.coef\_[0] ) )**

**logreg.intercept\_**

**#Predicting the test cases**

**hr\_test\_pred = pd.DataFrame( { 'actual': test\_y,'predicted': logreg.predict( test\_X ) } )**

**hr\_test\_pred = hr\_test\_pred.reset\_index()**

**#Comparing the predictions with actual test data**

**hr\_test\_pred.sample( n = 10 )**

Index actual predicted

125 13531 0 0

1048 4682 0 0

1726 5076 0 0

1862 2122 0 0

1888 14705 1 0

1032 749 1 0

2085 2348 0 0

911 4852 0 0

89 3429 0 0

2636 11662 0 0

**#Creating a confusion matrix**

**from sklearn import metrics**

**cm = metrics.confusion\_matrix( hr\_test\_pred.actual,hr\_test\_pred.predicted, [1,0] )**

**cm**

array([[ 225, 481],

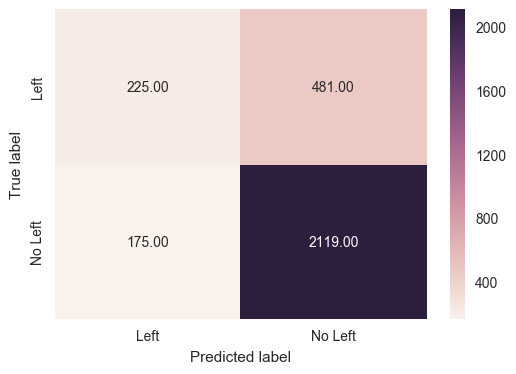
[ 175, 2119]], dtype=int64)

**import seaborn as sn**

**sn.heatmap(cm, annot=True, fmt='.2f', xticklabels = ["Left", "No Left"] , yticklabels = ["Left", "No Left"] )**

**plt.ylabel('True label')**

**plt.xlabel('Predicted label')**

****

**score = metrics.accuracy\_score( hr\_test\_pred.actual, hr\_test\_pred.predicted )**

**round( float(score), 2 )**

0.78

#### Observation:

* Overall test accuracy is 78%. But it is not a good measure. The result is very high as there are lots of cases which are no left and the model has predicted most of them as no left.
* The objective of the model is to indentify the people who will leave, so that the company can intervene and act.
* This might be the case as the default model assumes people with more than 0.5 probability will not leave the company.

**#Building Decision Tree**

**from sklearn import metrics**

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

**from sklearn.grid\_search import GridSearchCV**

**param\_grid = {'max\_depth': np.arange(3, 10)}**

**tree = GridSearchCV(DecisionTreeClassifier(), param\_grid, cv = 10)**

**tree.fit( train\_X, train\_y )**

**tree.best\_params\_**

**tree.best\_score\_**

0.98058171514292858

The accuracy is about 98%

**#Build Final Decision Tree Model**

**clf\_tree = DecisionTreeClassifier( max\_depth = 9 )**

**clf\_tree.fit( train\_X, train\_y, )**

**tree\_test\_pred = pd.DataFrame( { 'actual': test\_y,'predicted': clf\_tree.predict( test\_X ) } )**

**tree\_test\_pred.sample( n = 10 )**

actual predicted

14436 1 1

1053 1 1

5553 0 0

304 1 1

12129 1 1

13274 0 0

2214 0 0

10464 0 0

7932 0 0

13672 0 0

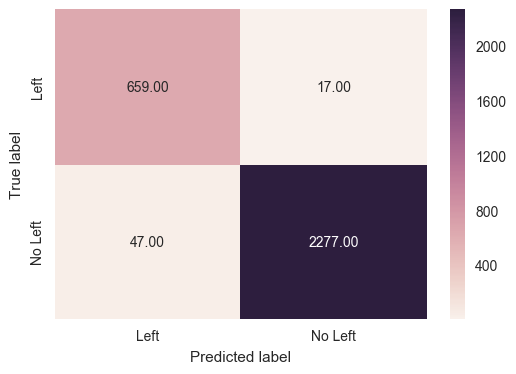
**metrics.accuracy\_score( tree\_test\_pred.actual, tree\_test\_pred.predicted )**

**tree\_cm = metrics.confusion\_matrix( tree\_test\_pred.predicted,tree\_test\_pred.actual,[1,0] )**

**sn.heatmap(tree\_cm, annot=True,fmt='.2f',xticklabels = ["Left", "No Left"] , yticklabels = ["Left", "No Left"] )**

**plt.ylabel('True label')**

**plt.xlabel('Predicted label')**

****

**#Generate rules from the decision tree**

**export\_graphviz( clf\_tree,out\_file = "hr\_tree.odt",feature\_names = train\_X.columns )**

**import pydotplus as pdot# install conda install -c conda-forge pydotplus,conda install -c conda-forge/label/gcc7 pydotplus**

**chd\_tree\_graph = pdot.graphviz.graph\_from\_dot\_file( 'hr\_tree.odt' )**

**chd\_tree\_graph.write\_jpg( 'hr\_tree.jpg' )#conda install -c anaconda graphviz**

**from IPython.display import Image**

**Image(filename='hr\_tree.jpg')**

#image stored as jpg file

**#Random Forest Model**

**from sklearn.ensemble import RandomForestClassifier**

**radm\_clf = RandomForestClassifier()**

**radm\_clf.fit( train\_X, train\_y )**

**radm\_test\_pred = pd.DataFrame( { 'actual': test\_y,'predicted': radm\_clf.predict( test\_X ) } )**

**metrics.accuracy\_score( radm\_test\_pred.actual, radm\_test\_pred.predicted )**

**tree\_cm = metrics.confusion\_matrix( radm\_test\_pred.predicted,radm\_test\_pred.actual,[1,0] )**

0.9853333333333333

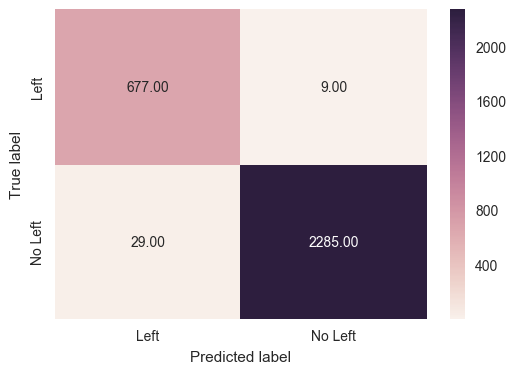
**Observation:**

It as (8 % accuracy).

**sn.heatmap(tree\_cm, annot=True,fmt='.2f',xticklabels = ["Left", "No Left"] , yticklabels = ["Left", "No Left"] )**

**plt.ylabel('True label')**

**plt.xlabel('Predicted label')**

****

**#Feature Importance from Random Forest Model**

**indices = np.argsort(radm\_clf.feature\_importances\_)[::-1]**

**feature\_rank = pd.DataFrame( columns = ['rank', 'feature', 'importance'] )**

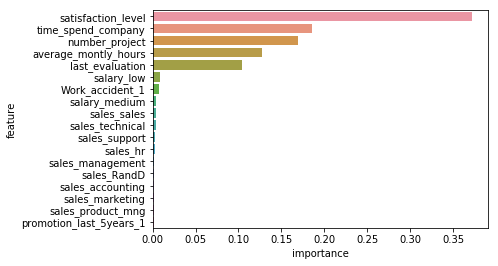
**for f in range(train\_X.shape[1]):# run fully for**

**feature\_rank.loc[f] = [f+1,**

**train\_X.columns[indices[f]],**

**radm\_clf.feature\_importances\_[indices[f]]]**

**sn.barplot( y = 'feature', x = 'importance', data = feature\_rank )**

****

**CHAPTER-V**

**CONCLUSION**

**5.1 SUMMARY OF THE PROJECT:**

As per the model, the most important features which influence whether to leave the company, in descending order, are

1. satisfaction\_level
2. number\_project
3. time\_spend\_company
4. last\_evaluation
5. average\_montly\_hours
6. work\_accident.

And the best model is the random forest classifier model compare to logistic regression and decision tree models by having the high accuracy of 98%.

**5.2 FUTURE ENCHANCEMENT:**

* Performing and building more models will help us to get the best results.
* Various data visualization methods will try to express the results**.**
* Having the more attributes in data helps us to make a predict of accuracy more and predict the expectations.

**5.3 REFERENCES:**

**>>>**[**https://machinelearningmastery.com/machine-learning-in-python-step-by-step/**](https://machinelearningmastery.com/machine-learning-in-python-step-by-step/)

**>>>**[**https://www.kaggle.com/ludobenistant/hr-analytics**](https://www.kaggle.com/ludobenistant/hr-analytics)