**Theory**

Regression is a technique in machine learning and statistics that circulates around finding out the strength, relationship and deviation between two *variables* or *features*; one of which is considered to be independent and the other a dependent variable.

Regression has further many types:

* Linear Regression
* Logistic Regression
* Ridge Regression
* Lasso Regression
* Polynomial Regression [1]

We will mainly look into Linear Regression as it is the most relevant to our work.

**What is Linear Regression?**

Being one of the most frequently asked questions regarding data analysis, this question holds great importance as it introduces the most efficiently and commonly used data analysis technique *linear regression*.

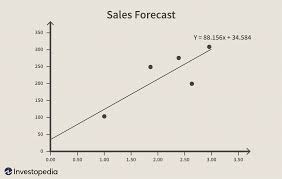
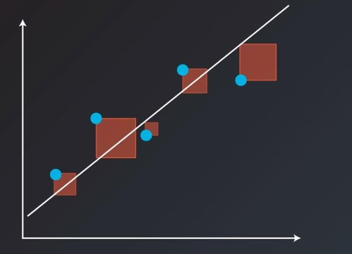
Linear Regression is a methodology that results in predicting unknown value(s) depending on their relationship with the independent variable(s) related to each other in a linear way.

* The dependent features/variables are called the ***dependent variables, outputs***, or ***responses***.
* The independent features/variables are called the ***independent variables, inputs***, or ***predictors***. [2]

For instance, consider the weight and height of a person two variables. A person’s weight varies according to their height. Here, the first variable depends on the latter thus making ‘weight’ the *dependent* variable that needs to be predicted and ‘height’ as the independent variable also known as the *predictor*.

Linear Regression models usually have a best-fitting line that passes straight through the points. This line is called a *Regression Line.*

A *Regression Line* describes how a dependent variable (say x) changes with respect to the independent variable (say y). It also helps us out to predict the value of the dependent variable.



Two separately Visualized Regression Lines

Python and R are the most efficient and commonly used languages for prediction through linear regression of data sets.

We will be using Python codes in Jupyter Notebook for the implementation of linear regression algorithms on our data set.

**Linear Regression using Python**

Our code includes three main libraries for the visualization and analysis of the data set:

* Numpy
* Scikit-learn
* Pandas

The package ***NumPy*** is an open source fundamental Python scientific package that allows many high-performance operations on single and multi-dimensional arrays. It also offers many mathematical routines.

Just like NumPy, the package ***scikit-learn*** is another widely used open source Python library for machine learning, built on top of NumPy and some other packages. It provides the means for preprocessing data, reducing dimensionality, implementing regression, classification, clustering, and more. [2]

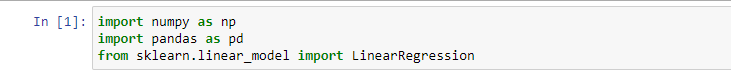
*Pandas* is also another game changer open source Python library used for data manipulation through series and data frames.

Importing these libraries into our data set makes it easier for us to deal with our data set. For example; to use our data set in the Jupyter Notebook, first we have to load the file through importing Pandas with an alliance. Secondly, we will read our data file via the Pandas alliance variable that can be whatever we have coded it to be. This is a much easier approach than writing the whole data set in Jupyter Notebook. We can now choose to do whatever we want with this data including preprocessing, manipulation, analysis and visualization.

The basic procedure of finding Linear Regression in Python is given below.

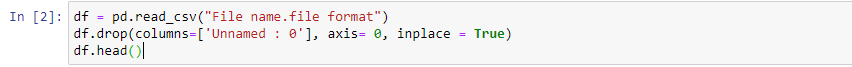
1. **Importing Modules/Libraries**

Libraries must be imported as per our needs before everything else in the compiler for Python.



1. **Providing the data or loading the data set file**

The second step is providing the data that we wish to work with. We can add data directly in the code or we can load our data set file by reading it through our imported libraries.



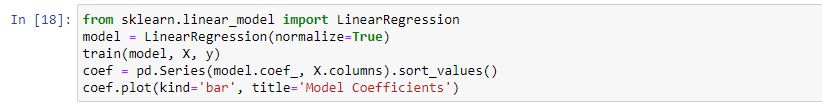
1. **Preprocessing the data**

The third step of the procedure is to pre-process or clean the data. In other words, this steps removes the unwanted, missing or null values from the data set.



1. **Creating a model for regression**

What we have to do next is, creating a regression model and fitting the data set in it via an instance of the LinearRegression class from the sci-kit learn library.



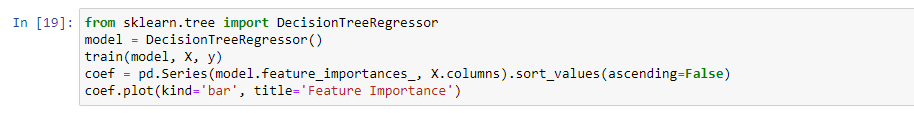
1. **Getting results from the model**

The final step is to get results from the linear regression model that we just created to see whether the model is working satisfactorily. The model can be predicted through manual prediction Statsmodels prediction or Sci-kit learn prediction.

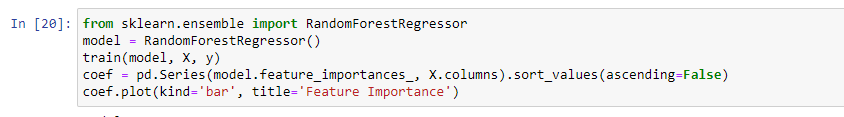
We can apply many other operations on our data set according to our wishes and demands.

For example;

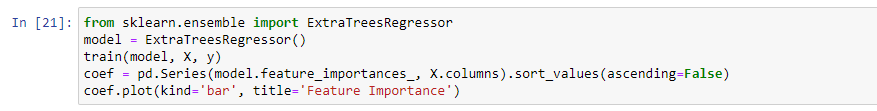
* using DecisionTreeRegressor class from the sci-kit learn library helps us to break down our data set into smaller subsets according to the attribute/features chosen.



* importing the RandomForestRegressor, a meta estimator class of sci-kit learn library that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [3]



* ExtraTreesRegressor is another meta estimator from the class sci-kit learn that fits many randomized decision trees or extra-trees on sub-samples of the data set and uses averaging for accurate predictions and over-fitting control.



# References

|  |  |
| --- | --- |
| [1] | "5 Types of Regression Analysis," Appier.com, 15 January 2021. [Online]. Available: https://www.appier.com/blog/5-types-of-regression-analysis-and-when-to-use-them/. [Accessed 29 October 2021]. |
| [2] | M. Stojiljkovic, "RealPython.com," Real Python, [Online]. Available: https://realpython.com/linear-regression-in-python/. [Accessed 30 October 2021]. |
| [3] | "Sci-kit Learn," scikit-learn.org, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed 31 October 2021]. |