Linear regression

# Theory

* It is based on a breakthrough by Francis Galton in an experiment in the 1800s where he discovered that the height of a man was roughly equal to his fathers but tended towards the overall average height of all people.
* A regression line is a line that is as close to every point as possible.
* It is a predictive, supervised learning model.
* The goal is to minimize the distance between the line and the rest of the data points.
* To minimize this, different methods from least squares method are used to minimize the residuals (difference between observed data and fitted line prediction).

# Linear regression – 1

## Data exploration

* Import data analysis libraries, data visualization libraries, with matplotlib inline to see plots in the notebook.
* Create or import data
* Check data head
* Explore data with info, describe, and appropriate plot types from seaborn possibly a pair plot of the whole data frame.
* You could also plot the distribution of the column you are trying to predict with histplot.

## Training the Model

### Splitting data into test and train set

* Split data frame into features (X) and target variables (y).
* Split data into testing and training sets by importing “**from sklearn.model\_selection import train\_test\_split** “. Then, use tuple unpacking to split the instance e.g., X\_train, X\_test, y\_train, y\_test = train\_test\_split (X, y, test\_size = a, random\_state = b) where X and y are the features and target variables respectively, test size is the proportion of the data that would be randomly assigned to the test sample and random state is a seed to get the same random split every time.

Note: test size and random state are optional and test size by default is 0.33

### Training the model

* From sklearn.linear\_model import LinearRegression
* Instantiate the model by creating a model object e.g., lm = LinearRegression()
* Fit the training set with lm.fit(X\_train, y\_train)
* You can evaluate the model by checking out coefficients such as print(lm.intercept\_), lm.coef\_
* To better explore he lm.coef\_, instantiate it as a variable and transform the variable into a dat frame e.g., df1 = pd.DataFrame (lm.coef\_, X\_columns, columns = ‘axd’) where axd is column title for the coefficient data frame.
* The coefficients help construct the linear regression equation and imply that a one unit increase in that variable leads to an increase in the predicted variable equaling the coefficient provided al other variables are held constant.
* To import data from scikit learn library for practice, use, from sklearn.datasets import load\_boston where boston is the data set to be imported. Then, create a variable to instantiate e.g., boston = load\_boston(). Then, explore.

# Linear regression – 2

## Prediction

* You can run the predictions by instantiating a prediction object and passing in the test features e.g., predictions = lm.predict (X\_test)
* You can test the accuracy of the predictions by:

1. plotting a scatter plot of the expected test result and the predicted results of the model. If it lines up linearly, the model is probably right and vice versa.
2. A histplot of the residuals (expected results – predicted results). If the residuals are normally distributed, the model is probably a good fit for the data.

## Regression evaluation metrics

* Evaluation metrics are used to check the deviation from the expected value of the model’s prediction. The common methods are:
* Mean Absolute Error (MAE): easy. Simply averages out the absolute error.
* Mean Squared Error (MSE): is the mean of the squared errors. Is popular because it punishes larger deviations.
* Root Mean Squared Error (RMSE): is the most popular because it is directly interpretable in the predicted values. It is the square root of the MSE.
* To compute these, from sklearn import metrics. Then use metrics.mean\_absolute\_error(a,b), metrics.mean\_squared\_error(a,b) and np.sqrt (metrics.mean\_squared\_error(a,b)) to calculate the mean absolute error, mean squared error and root mean squared error respectively. Where a and b are expected values and predicted values respectively.