Consulting Data Science project:

Consider the Combined Cycle Power Plant Dataset available on UCI machine learning repository.

https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant.

You have been contacted build a machine learning model to predict power output of a power plant given a set of readings from various sensors in a gas-fired power generation plant.

Dataset understanding:

The first step for building a machine learning application is understanding the business.

In our case, we want to predict the power output of a gas-fired power generation plant based on the readings from various sensors.

Peaking power plants are plants that supply power only occasionally. i.e. when there is high demand (peak demand) for electricity. Hence, the generated electricity is much more expensive than base load power plants. So it is important to understand and predict the power output to manage the plant connection to the power grid.

More information in Peaking power plants:
https://en.wikipedia.org/wiki/Peaking_power_plant

Data Set Variables:

Features consist of hourly average ambient variables

- Temperature (T) in the range 1.81°C and 37.11°C,
- Ambient Pressure (AP) in the range 992.89-1033.30 milibar,
- Relative Humidity (RH) in the range 25.56% to 100.16%
- Exhaust Vacuum (V) in teh range 25.36-81.56 cm Hg
- Net hourly electrical energy output (EP) 420.26-495.76 MW. The averages are taken from various sensors located around the plant that record the ambient variables every second. The variables are given without normalization.

This is a supervised machine learning problem since we have a labelled data set.

References:

- Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
- Pınar Tüfekci, Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods, International Journal of Electrical Power & Energy Systems, Volume 60, September 2014, Pages 126-140, ISSN 0142-0615
- Heysem Kaya, Pınar Tüfekci , Sadık Fikret Gürgen: Local and Global Learning Methods for Predicting Power of a Combined Gas & Steam Turbine, Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE 2012, pp. 13-18 (Mar. 2012, Dubai)

Part1: Linear Regression from Scratch

In this first part, you shall build a linear regression model by implementing Gradient Descent from scratch using linear algebra.

Please consider the following parameters:

- **m**: number of observations in the training set
- **n**: number of features (without the y offset)
- **w**: the vector of weights of your model (w_0, w_1, …., w_n)

$$w=\begin{bmatrix} w_0 \\ w_1 \\ w_1 \\ .. \\ .. \\ w_n \end{bmatrix}$$

- x: an observation vector (features and offset $x_0=1$) . Dimension : (n+1, 1)

$$x=\begin{bmatrix} x_0 = 1 \\ x_1 \\ x_1 \\ .. \\ .. \\ x_n \end{bmatrix}$$

- X: matrix of observations. Dimension : (m, n+1)

$$X=\begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \dots \\ \dots \\ \dots \\ (x^{(m)})^T \end{bmatrix}$$

- y: label (‘answers’) vector . Dimension : (m, 1)

$$y=\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

- **cost**: the cost function J

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2$$

- **delta**: the stop condition

delta= $J^{(k+1)} - J^{(k)}$

- **iterations**: max number of iterations of the gradient descent (default value = 1000)
- **alpha**: learning rate of the gradient descent (default value = 0.03)

<u>Gradient Descent:</u>

Repeat until convergence {

$$w_k = w_{k-} \frac{\propto}{m}\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)}){x_k}^{(i)}$$

}

Implement the following functions to build your model:

- notmalizeFeatures (params)
- costFunction(params)
- gradientDescent(params)

Part 2: Compare to Scikit-Learn

Once you build a good model with your from-scratch implementation, build a new model with Scikit-Learn and compare the obtained results.

- Add the following functions to your library to calculate R2, MAE, RMSE
  1. meanAbsoluteError()
  2. rootMeanSquaredError()
  3. r2()
- Use *LinearRegression* from *sklearn.linear_model* to build a new linear model
- Compare the obtained R2, MAE, RMSE

Part 3: Normal Equation

Write a function to calculate the optimal values of w_i using the normal equation