# 1 Introduction

You will model **Task #1’s Cleaned and Feature selected Dataset** with **Linear and Multivariable regressions**. In Python, you will use the Python Library Scikit-learn (Sklearn)

**Modeling and Prediction**

* from sklearn.linear\_model and import the LinearRegression() Class for both Linear and Multivariable regression modeling. It works by the ordinary least square error method as shown in the class Week 1 generating matrix and matrix inversion to estimate the optimized coefficients and intercept.

LinearRegression () class has .fit () and .predict () functions to perform modeling and prediction.

* + .fit(X\_train, y\_train) to generate a model by using 70 % of the dataset
  + .predict(X\_test) to predict, y\_predict value by using 30% X\_test(Vs, DEN, NEU)

**Model Performance Evaluation**

* Finally, you will evaluate the model accuracy performance with R2 value by using the model predicted (y\_predict) and the y\_test (true data).

The models you are going to generate are:

1. **For Linear Regression**, you will generate three models 1) Vp vs Vs 2) Vp vs DEN, 3) Vp vs NEU
2. **For Multivariable Regression,** you will generate four models 1) Vp vs (Vs, DEN) 2) Vp vs (Vs, NEU), 3) Vp vs (DEN, NEU) and 4) Vp vs (Vs, DEN , NEU)

# 2 How to perform a curve fitting with sklearn.linear\_model library with LinearRegression()?

**Here are the main steps:**

* **Step 1: Import libraries.**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split # For data splitting

from sklearn.linear\_model import LinearRegression # For modelling split dataset

from sklearn.metrics import mean\_squared\_error, r2\_score # For model performance analysis

* **Step 2: Import file**

df = pd.read\_excel (‘ Filename.xlsx’)

* **Step 3: Define x and y variables and splitting.**
  + # By convention, we used the Capital letter X for input features and the small letter y, for the target variable

#Step 3a: **Feature and Target variables selection** for **linear regression**:

#X= df[['Vs']] # Here use double parenthesis for the input feature

#y = df['Vp'] # Target variable

#Step 3b: **Feature and Target variables selection** for **Multivariable regression**

#X= df[['Vs','DEN']] # Here use double parenthesis for input feature

#y = df['Vp'] # Target variable

#Step 3c: **split** the data for training and testing with train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

print ("Training Data",X\_train.shape)

print ("Testing Data",X\_test.shape)

print ("Total Data",df.shape)

* **Step 4 Machine Learning Modeling and Prediction**

#Modeling: We use a lineraRegression() to perform training and prediction.

regressor = LinearRegression()

**Step 4a: Modeling**

regressor.fit(X\_train, y\_train) #Fit function. Generate optimized coefficients.

# Step 4b: Prediction

y\_pred = regressor.predict(X\_test) # Using the optimized coefficients

# Print optimized coefficients,

print (regressor.coef\_,regressor.intercept\_)

* **Step 5 Model Performance Accuracy Evaluations**

# Function to calculate the R^2 for predicted and true dataset:

mse = mean\_squared\_error(y\_test, y\_pred) # Mean square sum

rmser = np.sqrt(mean\_squared\_error(y\_test, y\_pred)) # Root means square error

r2 = r2\_score(y\_test, y\_pred) # Regression coefficient

print('Mean Squared Error:', mse)

print('Root Mean Squared Error:',rmse)

print('R-Square:',r2 )

* Finally, **plot** the curve fitting model (line) with the scatter (data) plot providing the equations.
* Compare Co based on True data (Vp) vs **your Model**, **Gardner, and Castagna** (Here, compute the R2 and Plot in a bar graph. In addition, your best model with True Vp, Castagana, and Gardner)
* Gardner r = 0.23(Vp)4
  + Vp (km/s) from Gardner in terms of r (g/cc): (Vp (km/s) = 0.0003048 (r/0.23)4
* Castagna Vp = 1.16\*Vs + 1.36 (Vp = km/s and Vs = km/s)