Name	**Rahma abdelkader mohmed**
Bn	**1**
Sec	**31**

Packages used:

- Numpy (both notebooks)
- Pandas (both notebooks)
- Scikit- learn (in first notebook)

Problem definition:

The main problem we aim to address is to accurately predict insurance costs for individuals based on a set of features that describe their personal characteristics and lifestyle choices.

EDA and processing summary: (EDA represented in EDA for insurance cost notebook)

- Data doesn't contain any missing values.
- Data only contain one duplication.
- Most of people ages ranges from 18-23
- Most of people bmi ranges from 26.0-31.0 as their mean is 31.0.
- The above observation declares that most of these people are overweight.
- Most of people don't have children
- The distribution of target is right-skewed
- most of people charges range from 1100 to 10000
- The distribution of sex is slightly equal but men are more
- Most of people don't smoke

Documentation for the used functions (they are already documented in notebook)

- **Normalization:**
  - Normalizing by mean and standard deviation before modeling helps to center the data and equalize feature scales, improving model performance and convergence.
  - Equation:

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

- **Denormalization:**
  - We inverse the normalization at the end to represent the graph by the real value of data

- **Linear reg:**

  - This function, linear_reg, performs linear regression using gradient descent. It takes input data x and target values y, and iteratively updates model weights and bias for a specified number of iterations using a given learning rate (alpha).
  - Equations:
    1. Predicted values (y_predict) are calculated using the linear model:
       y_predict = x * weights + bias

    2. The derivatives of the loss function with respect to weights (dw) and bias (db) are computed as follows:
       dw = (1 / num_samples) * x.T * (y_predict - y)
       db = (1 / num_samples) * Σ(y_predict - y)

    3. Weights and bias are updated using gradient descent:
       weights = weights - (alpha * dw)
       bias = bias - (alpha * db)

- **Predict:**

  - The function "predict" takes input data x, model weights, and bias as input and returns the predicted values (y_pred) using a linear regression model.
  - Equations:
    Predicted values (y_pred) are calculated using the linear model:
    y_pred = x * weights + bias

- **encoding:**

  - We encode the categoral features and change them into numeric to fit in the linear regression model.

- **r2_score:**

  - For model evaluation
  - Equation:

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

- **mean_square_error:**

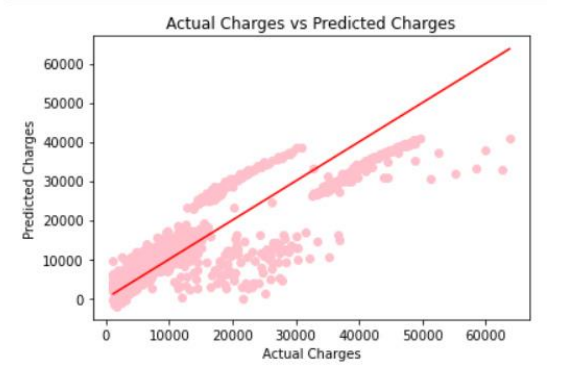
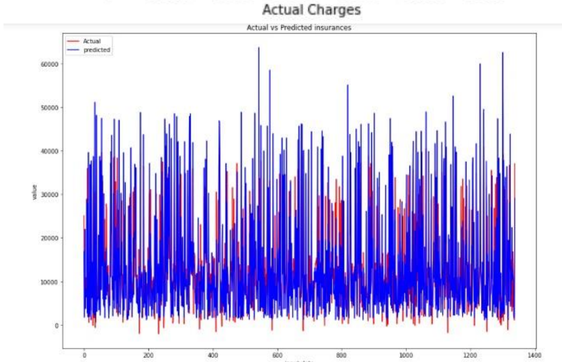
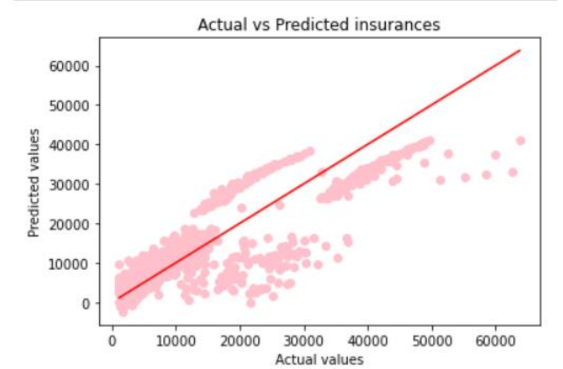
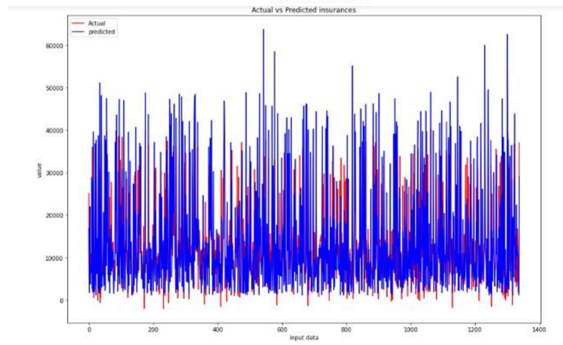
  - Equation:

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

- **Root_mean_square_error:**

  - Root the upper equation , this considered as an evaluation parameter for linear regression models

Comparison between both methods of implementation:

POC	Scikit-learn	From scratch
Graph		
R2 score	0.75	0.749
Extra functions needed	• Standard scaling • Label encoding	• Normalization • Denormalization • Manual encoding for categoral features