Linear Regression Application

An Insurance company provides Health Insurance to its customers. Various details of the customers are stored amongst which, *age, gender, bmi, children, smoker, and region* are to be considered. The *insurance charges* born by the company for each of their customers are also provided. It is required to build a model to predict the insurance charges for a customer given the features, *age, gender, bmi, children, smoker, and region.*

1. Give a formal description of this application in terms of Task, Experience, and Performance. (Make it a well posed problem)

   A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

   For example:

   a) ***Task*** *(T): Classify a tweet that has not been published as going to get retweets or not.*
   b) ***Experience*** *(E): A corpus of tweets for an account where some have retweets and some do not.*
   c) ***Performance*** *(P): Classification accuracy, the number of tweets predicted correctly out of all tweets considered as a percentage.*

2. Do exploratory data analysis on the data. (As given in Lab-0)

3. Do the required preprocessing on the dataset to make it suitable to apply gradient descent algorithm.

4. Find the relationship of each feature with the target feature *insurance charge* by using a scatter plot.

   For example, you can plot a graph with samples (Ai,Bi) with A on the X-axis and B on the Y-axis. A is one of the features, say, *age,* and B is the target feature, *insurance charges*. Repeat it for other features too such as *sex, bmi, children, smoker, region.*

5. Implement linear regression using inbuilt package of python scikit. This will help to estimate the function f(X)=Y where X is one of the input features and Y is the target feature, *insurance charges.*

   (Pl. note, this is univariate since only 1 feature needs to be considered)

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test) ]
```

   Fit the curve obtained by gradient descent algorithm on the graph plotted in Q.4.

6. Implement gradient descent algorithm with the function prototype

def gradient_descent(alpha, x, y, max_iter=1500):

where alpha is the learning rate, x is the input feature vector. y is the target feature. Convergence criteria: when no: of iterations exceed max_iter.

The output of the function is the parameter θ, where θ_j is the parameter corresponding to feature *j* as given in the following equation.

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad \text{(simultaneously update } \theta_j \text{ for all } j\text{)}.$$

   $h_\theta(x) = \theta^T x$; *x* is the input feature vector and *y* is the target feature. x_j^i is the jth feature value of ith sample. m is the number of training samples.

7. Vary learning rate from 0.1 to 0.9 and observe the learned parameter.

8. Draw a contour plot of cost function and simulate the steps of gradient descent.

   The cost function is.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^i\right) - y^i\right)$$

   where m is the number of training samples.

The following is a small example contour for a function.

```
xmesh, ymesh = np.mgrid[-2:2:50j,-2:2:50j]

fmesh = f(np.array([xmesh, ymesh]))

plt.contour(xmesh, ymesh, fmesh)

def f(x):

    return 0.5*x[0]**2 + 2.5*x[1]**2
```

9. Compute the error metrics MSE, RMSE and compare.