Linear Regression

We are required to create a linear regression model from scratch for this project, without using Scikitlearn or any other built-in libraries that already have the model implemented.

I created a linear regression model for this assignment that internally employs gradient descent and was built utilizing the oops and class-based ideas.

Libraries Used

```
import pandas as pd import numpy
as np import matplotlib.pyplot as
plt
%matplotlib inline from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split import seaborn
as sns
```

Although we can use Pandas for graph plotting as well, we have chosen to use two different libraries—Mat Plot and Seaborn—for improved visualization. Pandas was used for data loading and preprocessing.

Furthermore, we converted the categorical variables from sklearn to numbers using the Label Encoder so that we could utilize them in our model. We can accomplish it manually as well. For experimental purposes, I created a different implementation of the same function, but I won't go into that here because it was outside the preview of the study.

In addition, we generated training and testing data using train-test-split. We also considered conducting some hypothesis testing, but were unable to do so owing to time restrictions.

In this assignment we have done following operations -

1. Data loading and Data visualization
2. Data preprocessing
3. Model Development training and Model testing
4. Results viewing and comparing

Data loading and Data visualization

Data Visualization

In order to get insights on the data we have done data visualization we have done that in two ways –

- Graphical visualizations - In graphical visualization we plotted several graphs like ○

  Box Graph ○ Bar graph ○ Heat map ○ Histogram

  We discovered that most of the columns in the statistical visualization are independent, so in order to identify any bias in the datasets, we also created sets of all possible values and counted the frequency and most frequently occurring values in each column.

Data preprocessing –

This step involves modifying the categorical values and deleting dependent columns to make it easier for our model to be trained. We can encode using two approaches: one-hot encoding and twoapproach labels encoding. For the time being, we're using label encoding, but if we have time later, we'll also try one-hot encoding. We also attempted normalization, as well.

Model Development training and Model testing –

We used a class-based approach for developing the linear regression model.

```
class LinearRegression:
    def __init__(self, learning_rate=0.01,
n_iterations=1000):
        self.learning_rate           =           learning_rate
self.n_iterations = n_iterations          self.weights = None
self.bias = None         self.mse_history = []  # To store MSE
values during training
    #Based on weight bias and features, generating labels
def line_equation(self,X):
      return np.dot(X, self.weights) + self.bias
    #Calculating gradient of weight     def
grad_w(self,n,X,y):
      return (1 / n) * np.dot(X.T, (self.line_equation(X) -
y))
    #Calculating gradient of bias     def
grad_b(self,n,X,y):       return (1 / n) *
np.sum(self.line_equation(X) - y)
    #Fitting the model     def
fit(self, X, y):
        num_samples, num_features = X.shape
        self.weights = np.random.rand(num_features)  #
Initialize weights with random values
        self.bias = np.random.rand()
        #for each iteration adjusting weight and bias through
gradient descent approach and storing MSE for plotting of graph
for _ in range(self.n_iterations):
            self.weights -= self.learning_rate *
self.grad_w(num_samples,X,y)             self.bias -=
self.learning_rate * self.grad_b(num_samples,X,y)             mse
= np.mean((y - self.line_equation(X)) ** 2)
self.mse_history.append(mse)
    #Based on trained weight and bias, predicting labels for
feature vectors     def predict(self, X):
linear_model = np.dot(X, self.weights) + self.bias         return
linear_model
```

this class is having 3 functions and takes 2 parameters learning rates and number of iterations.

1. __init__() -This is the model's initialization function; it sets the model's weight, bias, learning rate, number of iterations, and a list that can store mean square error, which I used to plot the graph later.
2. Fit() – Internally, this function uses gradient descent. Basically, we randomly initialize the weights and bias, then run the training number of iterations using a for loop, and then compute or simply predict the output based on the linear relationship between the features. Afterward, we calculated the gradient of the weights and bias, got all the summation in order to find the means square error, and stored that into an array that we later used to predict the output.
3. Predict() – This is a straightforward mathematical function that uses the equation of line to predict y based on x weight and bias.

Results viewing and comparing –

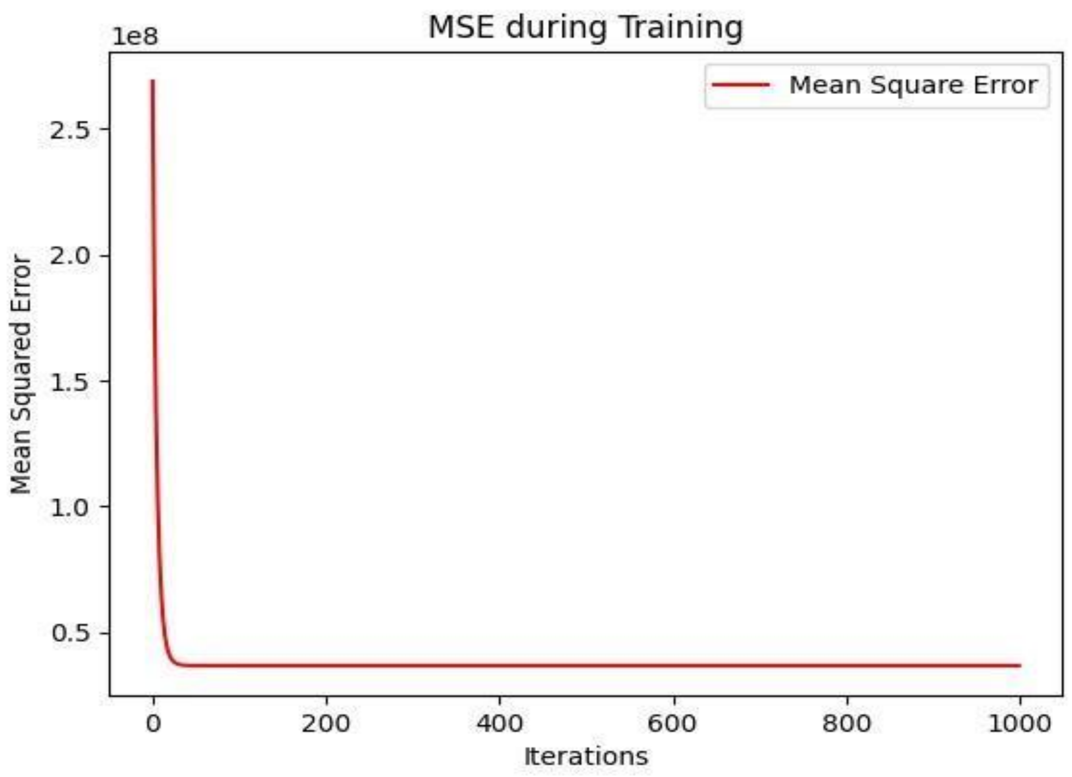
We run three sets of experiments –

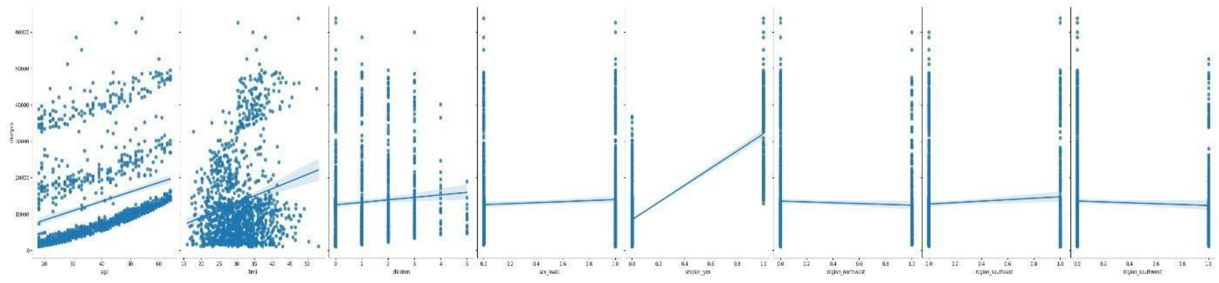
1. BMI vs charge
2. All features combined vs charge
3. Each individual features vs charges

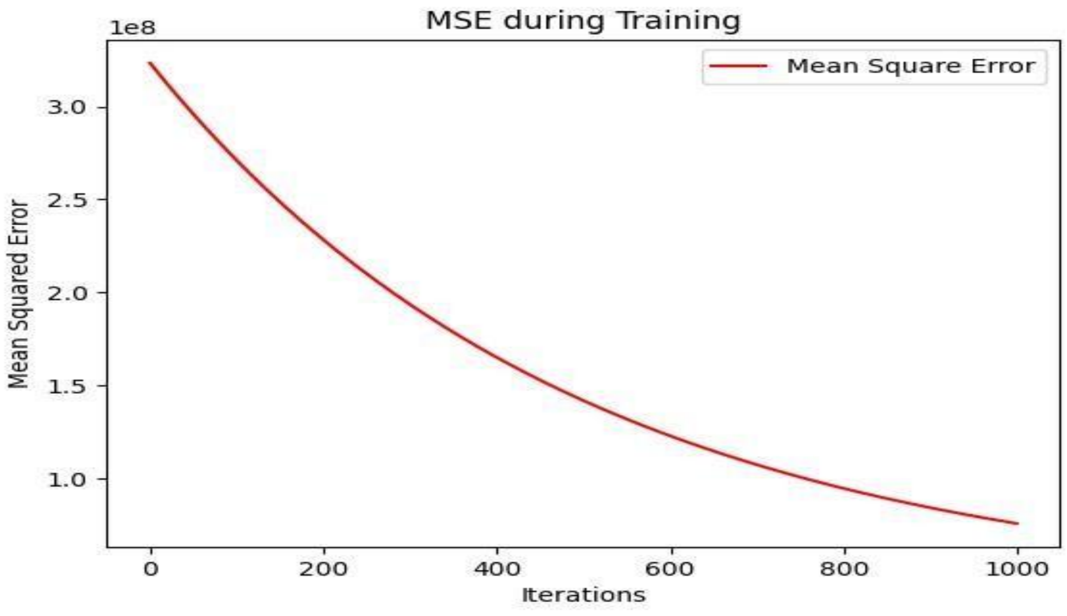
In addition to that we developed the code in such a way that it can be used with and sets of features and simple to experiment with also we experimented with different sets of learning rate we used total 3 sets of learning rates [0.1,0.01,0.001,0.0001] also with different number of iterations.

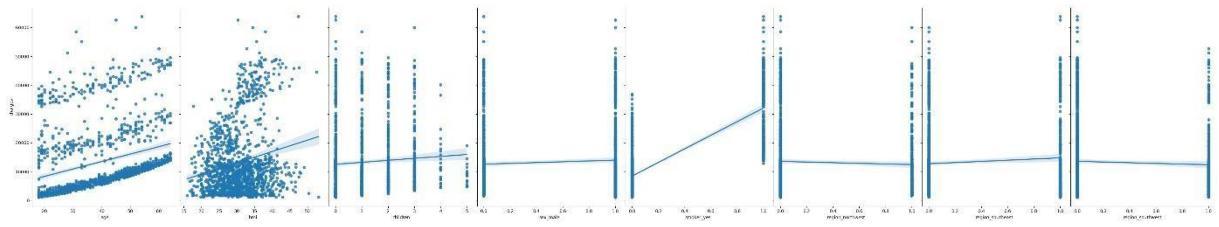
We printed the final weight, bias, and mean square error in addition to using a scatterplot with regression lines and an SNS pair plot to visualize the model's performance.

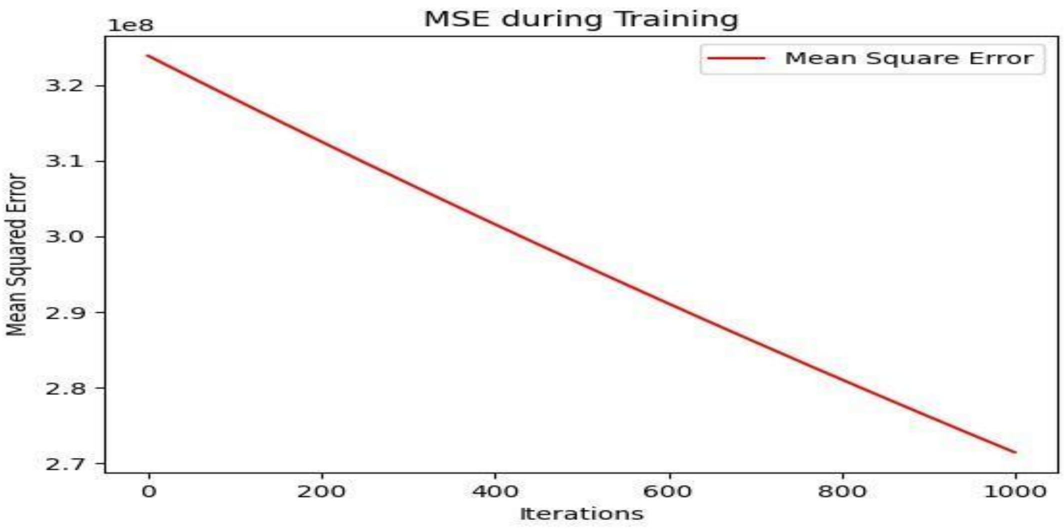
Conclusion – The model takes longer to converge the lower the learning rate; after a significant number of iterations, the mean square error stops changing, depending on the learning rate. To address this problem, numerous other gradient descent variations are being developed; to the best of my knowledge, Adam is a much more complex algorithm than this straightforward gradient descent.


MSE during Training
1e8
Mean Square Error
Mean Squared Error
Iterations
0
200
400
600
800
1000
0.5
1.0
1.5
2.0
2.5


charges
age
bmi
children
sex_male
smoker_yes
region_northwest
region_southeast
region_southwest


MSE during Training
1e8
Mean Square Error
Mean Squared Error
3.0
2.5
2.0
1.5
1.0
0
200
400
600
800
1000
Iterations




MSE during Training
1e8
Mean Square Error
Mean Squared Error
3.2
3.1
3.0
2.9
2.8
2.7
0
200
400
600
800
1000
Iterations