CS584_Assigment1_Spring2023_A20516822

March 10, 2023

1 Assignment 1

1.1 Bhavesh Rajesh Talreja (A20516822)

Due by 11:59pm, Feb. 9, 2022

1.2 Theory Questions (Question 4 is worth 10 points, other question is worth 5 points)

1. Explain the differences between supervised, semi-surpervised, weakly-supervised and unsupervised learning.

Answer:

Supervised learning: a) In this type of learning, we deal with labelled data and the algorithm is trained to learn via this labelled data to make the prediction on unseen/test data. b) As we have the features values and target value as input/training data, the algorithm tries to learn the mapping relationship between the feature values and the target value. c) Post learning, we use this model on the test data and compare the predicted values and the ground truth/actual values, and if necessary, we tune the model to minimize the error if the error between the predicted and actual values is high. d) Example: Regression, Classification, Decision trees

Unsupervised learning: a) In this type of learning, we deal with unlabelled data and the algorithm is left to learn on its own without any supervision. b) The goal is to discover patterns or relationships in the data and group similar data together. c) In the real world, most of the times, we have to deal with unlabelled data which means unsupervised learning is commonly used. d) Example: Clustering, Anomaly detection, Dimensionality reduction

Semi-supervised learning: a) Supervised learning algorithms requires labelled data which in real life is both time and economically expensive. On the other hand, unsupervised learning algorithms handle unlabelled data and have to discover the patterns without any supervision involved. b) To overcome these extremes, a common ground concept was introduced which is semi-supervised learning. In this we deal with a limited amount of labelled data and large amount of unlabelled data. The reason is that with the limited labelled data we are providing little supervision and with large amount of unlabelled data we are forcing the algorithm to learn with better efficiency and accuracy. c) The algorithm uses the unlabelled data to cluster or group the similar data together and then use the available labelled data to classify the grouped data. d) Example: Text classification, Speech analysis, Protein sequence classification.

Weakly-supervised learning: a) Weakly-supervised learning deals with partially labelled data. Just like semi-supervised learning, it falls between supervised and unsupervised learning. b) The concept

of this learning is that the model is trained using the partial labelled data and is used to make prediction on complete unlabelled data. c) Both semi-supervised and weakly-supervised learning work on making the most out of partial/limited labelled data so that the model can perform well when it comes to unseen/test data. d) Example: Image classification, Sentiment analysis.

Reference: https://rstudio-pubs-static.s3.amazonaws.com/559023_f62f3bce1be14fb9b248127194c0c1e3.html https://www.geeksforgeeks.org/ml-semi-supervised-learning/

2. Explain the difference between regression and classification.

Answer:

Regression and Classification are two common examples where supervised learning is used.

Regression: a) To regress in general sense means to go back, since we regress back to the mean, this is called Regression. b) As it is an example of supervised learning, we have the target value data as well and if the target value is of continuous data type, regression is used for this use case. c) The idea of regression is to find the underlying relationship/mapping between the input features and target value using the labelled training input data and then use the model to predict the target value on test/unseen data. d) Example: predicting the sales of a product due to various advertising channels (like television, newspaper, online ads).

Classification: a) To classify in general sense means to identify and group similar things, here we do the same with similar data points, hence this is called Classification. b) In the classification problem, the target value is of discrete data type. c) The idea of classification is to learn the mapping between input features data and its respective class so as to identify the unseen data into appropriate bucket/class. d) Example: predict the class of flower based on its sepal length, sepal width, petal length, petal width, etc.

3. Explain how to perform N-fold cross-validation.

Answer:

N-fold cross validation, as the name suggests it is a validation technique. It is used for evaluating machine learning models and estimate their performance on test data.

It is a parametrized approach, the parameter N decides the number of folds in which the data will be divided for validation.

Based on N, we divide data into N folds out of which N-1 folds are used for training the model and the remaining fold is used for testing. This process is iterated N times; that means every fold acts as the test data once. The overall metric performed is obtained by averaging the individual iterations performances collected.

There are various types of N-fold cross validation. I have heard about Holdout cross validation Leave one out cross validation

In Holdout cross validation, the data is divided into 80% training and 20% testing data as a general rule of thumb. While in the Leave one out cross validation, only one sample row is used for testing and rest of the whole data is the training data set. LOOC method is susceptible to overfitting and is computationally expensive.

General process for N-fold cross validation: 1. The very first step is to randomly shuffle the data so as to avoid any contiguous chunk of data which can act as a bias or worse situation will be the model will be trained on only data belongin to one class. 2. Then the whole dataset is partitioned into

N parts of equal size and each partition is called as a fold. 3. Choose the first fold as testing data and use the N-1 folds as training data and fit the model. 4. Get all the performance metrics based on the first fold as the testing dataset. 5. Repeat the process considering the second fold as testing dataset & remaining folds as training data, get all the performance metrics for that test dataset. 6. This process will be repeated till the N^{th} fold will become the test dataset and its previous N-1 folds will be used for training the model and fitting the model to get all the performance metrics. 7. Finally, an average of the performance metrics will be taken to get the overall performance metric. 8. For example: In 3-fold cross validation, it will be iterated 3 times.

Advantage: It helps in overcoming the problem of overfitting because the algorithm is trained and tested on multiple datasets which makes the overall estimate more robust.

4. Bayes rule:

John owns a retail store for selling phones. The phones are manufactured at three different factories, A, B, C, where factory A, B, and C produces 20%, 30%, and 50% of the phone being sold at John's store. The probabilities of the defective phones from stores A, B, and C are 2%, 1%, and 0.6%, respectively. The total number of phones being sold at John's store is 10,000. One day, a customer walks up to John's store, and ask for a refund for a defective phone. 1. What is the probability of a phone being defective? 2. What is the probability that this defective phone is manufactured at factory A? 3. What is the probability that this defective phone is manufactured at factory B? 4. What is the probability that this defective phone is manufactured at factory C?

Answer:

Let the P(DP) be the probability of phone being defective and probability of defective phone from store A, store B, store C be P(DP|A), P(DP|B), and P(DP|C) respectively.

P(A) be the probability that the phone was manufactured at factory A.

Here, P(DP|A) = 0.02, P(DP|B) = 0.01, P(DP|C) = 0.006

1. Probability of a phone being defective:

In order to calculate this value, the probability of a phone being defective from each factory and the proportion of phones produced by each factory is needed.

20% phones are produced by factory A, 30% phones are produced by factory B, and 50% phones are produced by factory C.

P(Phone being defective) = (P(A) * P(DP|A)) + (P(B) * P(DP|B)) + (P(C) * P(DP|C)) = (0.2 * 0.02) + (0.3 * 0.01) + (0.5 * 0.006) = (0.004) + (0.003) + (0.003) = 0.01

This tells us that the probability of a phone being defective is 1%.

2. Probability that this defective phone is manufactured at factory A:

To calcuate P(A|DP), we need to use Baye's rule

P(A|DP) = P(A) * P(DP|A) / P(DP) = 0.2 * 0.02 / 0.01 = 0.004 / 0.01 = 0.4

The probability that the defective phone manufactured at factory A is 40%.

3. Probability that this defective phone is manufactured at factory B:

Now, let P(DP) be the probability of phone being defective and let P(B) be the probability that the phone was manufactured at factory B.

To calcuate P(B|DP), we need to use Baye's rule

P(B|DP) = P(B) * P(DP|B) / P(DP) = 0.3 * 0.01 / 0.01 = 0.003 / 0.01 = 0.3

The probability that the defective phone manufactured at factory B is 30%.

4. Probability that this defective phone is manufactured at factory C:

Now, let P(DP) be the probability of phone being defective and let P(C) be the probability that the phone was manufactured at factory C.

To calcuate P(C|DP), we need to use Baye's rule

P(C|DP) = P(C) * P(DP|C) / P(DP) = 0.5 * 0.006 / 0.01 = 0.003 / 0.01 = 0.3

The probability that the defective phone manufactured at factory C is 30%.

5. Explain the standard steps to perform machine learning.

Answer:

In order to perform machine learning, one must follow these steps:

1. Understanding the problem at hand: It is important to understand the problem we are trying to solve via machine learning and we should be able to define it in terms of machine learning domain.
2. Data collection: Collecting relevant data is crucial for us to perform machine learning tasks. A lot of research and domain expertise is needed to be able to collect relevant data and it can be a time consuming and expensive task as quality and quantity of data will be one of the deciding factor as to how the model will perform.
3. Data cleaning/preprocessing: Even though a lot of research and domain expertise is used for data collection process, the real world data contains some missing values, null values, unordered data, irrelevant data may slip in which is why we must ensure that we don't feed this data to the model as it is one the make or break steps in machine learning. A lot of time is spent on this cruical step which ensures the accuracy of the model in the long run.
4. Exploratory data analysis: Before modelling the data, it is important that we understand the data we have and we have set of common practices to analyze the data via visualization of the raw data. It gives us a basic idea of how the output should turn out. For example: a simple plot of the data can inform us that the data will be divided into 3 clusters by the model which will be the 3 classes in terms of classification.
5. Feature engineering: Often times, we deal with data which consists of different scales and we need to either scale the data or normalize it for using it for modelling purpose to make the sense out of it.
6. Defining the model: Based on the application, we need to decide which machine learning model we want to use. Various models can be used in ensemble as well. For example: Random forests can make use of multiple decision tree models, for prediction, we can use linear regression or decision trees, for classification tasks, we can use k-nearest neighbouring algorithm.
7. Train the model: We use the data collected and divide it among three types training data, cross validation data, test data. Using the training data, we train the model to understand

the relationship between the features, find the patterns, find similar set of data, etc.

8. Model evaluation: Once the model has been trained, we use the cross validation data to tune the model if the model is not performing as per the expectations. Finally, the test data which is never seen by the model is used to evaluate the performance of the model is as per the needs of the application or not.
9. Model deployment: The final step in this process is to deploy the model to face the real world data. If the model performs well on the real world data as well, it is considered as a successful deployment and further updates are made over the time. If it doesn't perform well, it is tuned and retrained on totally new data so as to meet the performance criteria needed.

6. Let $(x^{(i)}, y^{(i)})_{i=1}^{m}$ be a set of training examples where $y_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^n$, write a linear regression model using model parameter θ_j and features $x_1, ..., x_n$.

Answer:

To define a linear regression model, we can define it in terms of a simple linear regression model as well as multi variate linear regression model.

Considering simple linear regression model, it has one independent feature variable (x), one dependent target variable (y), two model parameters (θ_0 and θ_1), and irreducible error term (ϵ)

$$\hat{y}_i = \theta_0 + \theta_1 * x_i + \epsilon$$

Considering multi variate linear regression model, it has multiple independent feature variables (x_1 to x_n), one dependent target variable (y), n+1 model parameters (θ_0 to θ_n), and irreducible error term (ϵ)

$$\hat{y}_i = \theta_0 + \theta_1 * x_{i1} + \theta_2 * x_{i2} + ... + \theta_n * x_{in} + \epsilon$$

7. Write the objective function that can be used to determine the regression model parameters. How is this objective function will be used to find model parameters?

Answer:

Consider the linear regression model for our reference

$y_i = w_0 + w_1 * x_{i1} + w_2 * x_{i2} + \ldots + w_n * x_{in} + \epsilon_i$

where w_i are the parameters of linear regression, ϵ_i is the random error component

The objective function that we commonly use for regression is mean squared error which is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

The goal of an objective function is to minimize the error by reducing the difference between the true and predicted values. In each iteration, the weights are updated so as to minimize the error.

We use gradient descent algorithm for this, we differentiate the objective function with respect to the model parameters to get the gradient. Then, we update the model parameters in the direction opposite to the gradient, using a constant step size known as the learning rate (η).

$$w_i = w_i - \eta * \frac{\partial MSE}{\partial w_i}$$

The process is repeated until the gradient becomes very small, indicating that the parameters have converged to a minimum of the objective function. The final values of the parameters obtained after convergence are the optimal parameters for the regression model.

8. Let θ =[1,2,3] be the model parameters obtained for linear regression. Let x =[-4,5] be a feature vector. Find the value y that will be obtained by the linear regression for the feature vector x.

Answer:

From the data θ = [1,2,3] and x = [-4,5]

In order to find the value y:

$y = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2 = 1 + 2 * (-4) + 3 * (5) = 1 - 8 + 15 = 8$

9. Suppose an artifial neuron has weight W = [0.4, 0.6], bias b = 0.1 and a sigmoid activation function. If the input data = [3,4], what is the output value of the neuron?

Answer:

From the data, W = [0.4, 0.6], b = 0.1 and x = [3,4]

In order to find the predicted value y from the artificial neuron:

y = sigmoid($w_1 * x_1 + w_2 * x_2$ + b) = sigmoid(0.4 * 3 + 0.6 * 4 + 0.1) = sigmoid(1.2 + 2.4 + 0.1) = sigmoid(3.7) = $1/(1+e^{-3.7})$ = 0.9758 = 0.98

1.3 Programming Questions (50 points, each part is worth 10 points)

In this assignment, you are going to implement your own Simple Linear Regression function.

Please notice: **No library versions of linear regression are allowed**.

[1]:
```
# Do not edit the codes in this cell
# load required library
from sklearn.datasets import load_diabetes
import matplotlib.pyplot as plt
import numpy as np
# load dataset
X, y = load_diabetes(return_X_y=True)
X = X[:, 2]
```

Part I _____

Using the given dataset above, draw a scatter plot. 1. Plot title and xlabel, ylabel are required; 2. Show the plot in the Output.

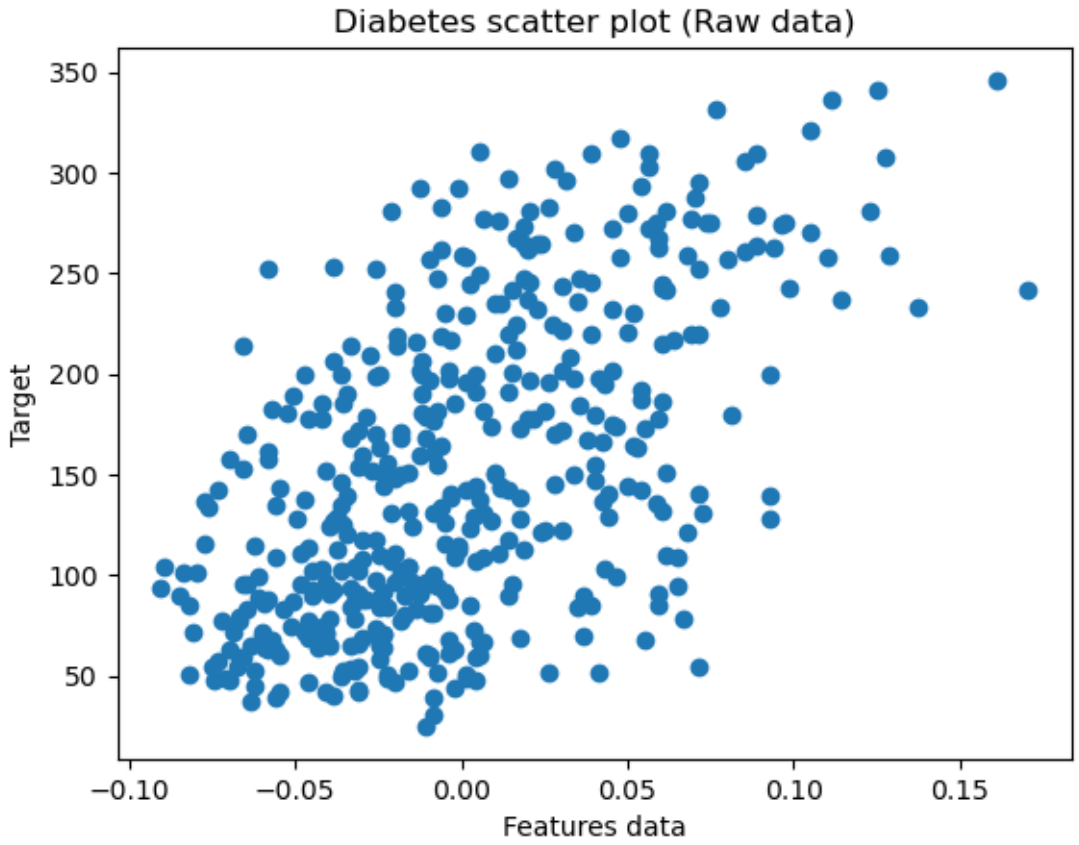
Example plot

[2]:
```
# draw the raw data plot
# TODO
plt.title("Diabetes scatter plot (Raw data)")

#Reference for xlabel and ylabel: https://scikit-learn.org/stable/modules/
 ↪generated/sklearn.datasets.load_diabetes.html

plt.xlabel("Features data")
plt.ylabel("Target")

plt.scatter(X,y)
plt.show()
```



Part II _____

Initialize theta and construct cost function for the upcoming gradient decent. 1. Initialize theta, print it out; 2. Construct Mean Square Error loss function; 3. Calcualte the cost using initial theta,

X and y, and print the result out.

[3]:
```
# As a common practice in Linear regression, adding a column of ones to provide␣
 ↪the intercept term.
X = np.c_[np.ones(X.size), X]
```

[4]:
```
# initialize theta
# TODO

# initializing the theta with size of number of columns in X and with random␣
 ↪values
# i.e., the parameters of linear regression.

theta = np.random.rand(X.shape[1])
print("Initial theta:", theta)
```

```
Initial theta: [0.18003749 0.52762876]
```

[5]:
```
# define cost function
# TODO

# Mean square error for linear regression: Mean Square of y (true value) - dot␣
 ↪product(X, theta)
def mean_square_error(X, y, theta):
    return np.mean((y-X.dot(theta))**2)
```

[6]:
```
# calculate cost by calling the function
# TODO

# initial cost using initial theta, X, and y
initial_cost = mean_square_error(X, y, theta)
print(initial_cost)
```

```
29017.468742367004
```

Part III _____

Gradient descent to find the optimal fit. 1. Initialize learning rate and epoch, try to explain your reasons for the values chosen; 2. Construct gradient descent function, which updates the theta and meanwhile records all the history cost; 3. Call the function for the optimal fit. Print out the final theta and final cost.

Question: How did you choose your lr and epoch number?

Answer:

[7]:
```
# gradient descent to find the optimal fit
# TODO
def gradient_descent(X, y, theta, no_of_epochs, eta):
```

```
    cost_history = np.zeros(no_of_epochs) #defining a numpy array for storing␣
↪the cost for each iteration
    #for each epoch, update the weights and calculate the error cost.
    for i in range(no_of_epochs):
        theta = theta - eta * (2 * X.T.dot(X.dot(theta) - y)/len(y)) #eta is␣
↪the learning rate
        cost_history[i] = mean_square_error(X, y, theta)
    return theta, cost_history
```

[8]:
```
#Trying out different values for no of epochs and learning rate.
#theta, cost_history = gradient_descent(X, y, theta, 10000, 0.1)
```

[9]:
```
#Trying out different values for no of epochs and learning rate.
#theta, cost_history = gradient_descent(X, y, theta, 15000, 0.1)
```

[10]:
```
#Trying out different values for no of epochs and learning rate.
#theta, cost_history = gradient_descent(X, y, theta, 15000, 0.01)
```

[11]:
```
#Trying out different values for no of epochs and learning rate.
theta, cost_history = gradient_descent(X, y, theta, 20000, 0.01)
```

[12]:
```
print("Final Theta:", theta)
print("Final Cost:", cost_history[-1])
```

```
Final Theta: [152.13348416 565.56155327]
Final Cost: 4223.848040241246
```

By the above trial and error method, I chose the values for no. of epochs and learning rate. These two values have an impact on the convergence of the gradient descent algorithm.

In general, we want the number of epochs to be a large number so that the algorithm converges to the optimal solution. Care should be taken to make sure that the number is not too large that it becomes computationally expensive. Hence, after few trials, I decided to go with N=20000 epochs. The reason is that it is large enough to capture the convergence of the algorithm and not too large enough for it to be computationally expensive.

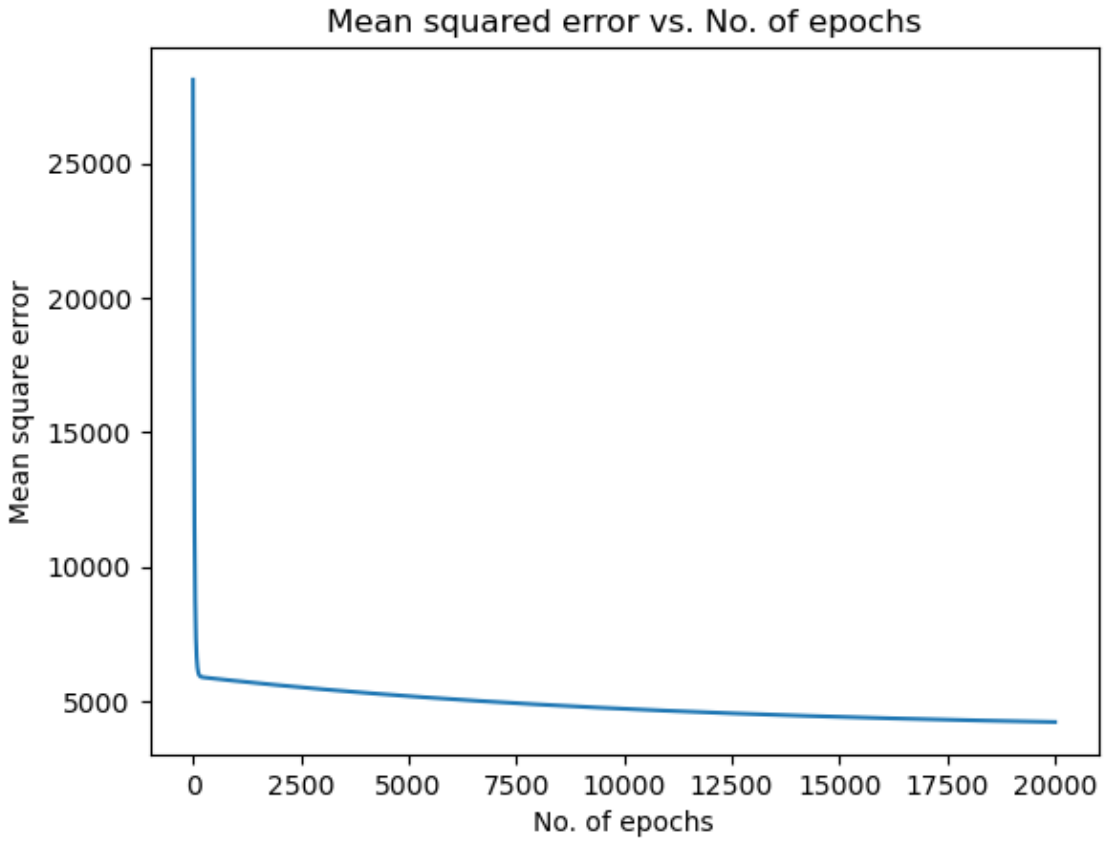
For learning rate, the value I chose is 0.01, the reason is that with a small learning rate the algorithm makes small updates to the weights and it can help us observe the progress.

Part IV _____

Plot the cost for each iteration. 1. Plot title and xlabel, ylabel are required; 2. Show the plot in the Output.

[13]:
```
# draw the cost change
# TODO
no_of_epochs = 20000
plt.plot(list(range(no_of_epochs)), cost_history)
plt.xlabel("No. of epochs")
```

```
plt.ylabel("Mean square error")
plt.title("Mean squared error vs. No. of epochs")
plt.show()
```

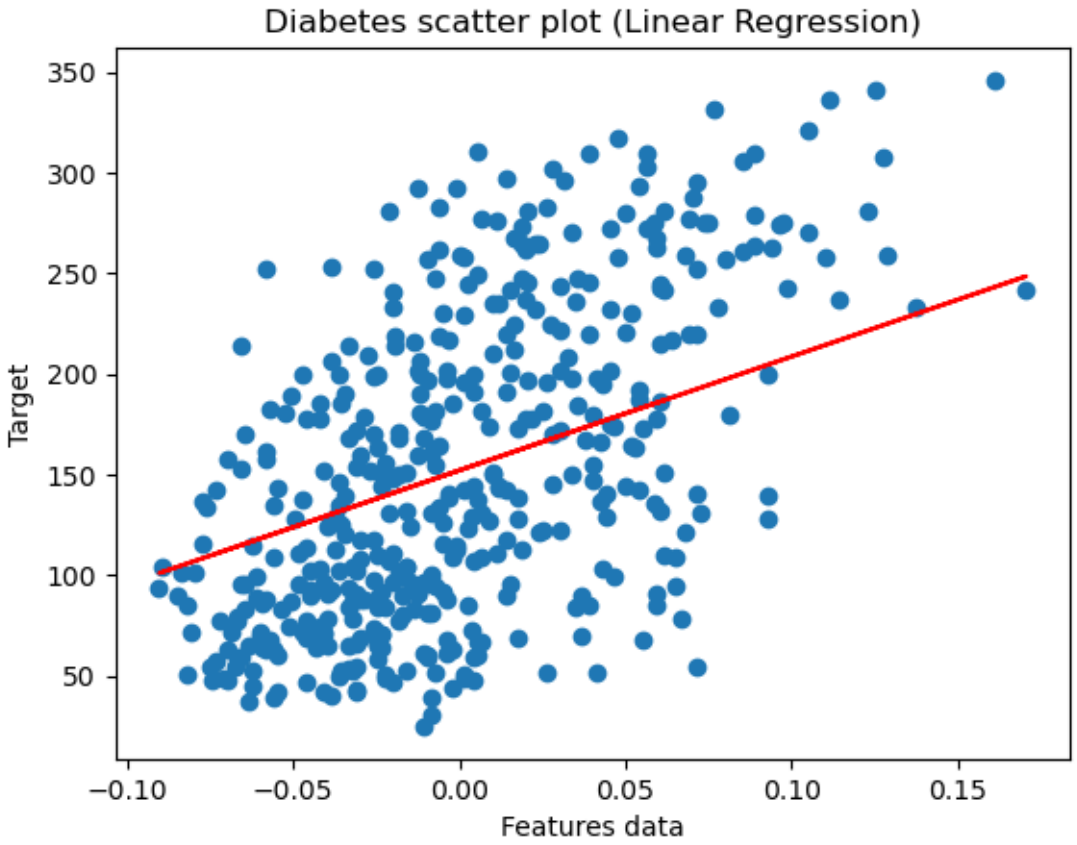


Part V _____

Based on the raw data plot you already drawn, plot the final linear regression model using the final version of theta. 1. Plot title and xlabel, ylabel are required; 2. Show the plot in the Output.

Example plot

[14]:
```
# draw the final linear model
# TODO
plt.title("Diabetes scatter plot (Linear Regression)")
plt.xlabel("Features data")
plt.ylabel("Target")
plt.scatter(X[:,1],y)
plt.plot(X[:,1], X.dot(theta), color='red')
plt.show()
```



References:

[1] Pattern Recognition and Machine Learning, Christopher M. Bishop, ISBN: 978-0387-31073-2 [2] The Elements of Statistical Learning Data Mining, Inference, and Prediction, Trevor Hastie Robert Tibshirani Jerome Friedman, Second Edition [3] An Introduction to Statistical Learning, Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, 2nd Edition [4] Introduction to Data Mining, PANG-NING TAN, MICHAEL STEINBACH, ANUJ KARPATNE, VIPIN KUMAR, 2nd Edition