CSE 4554
Machine Learning Lab

Experiment No: 2

Name of the experiment: Visualization, Preprocessing, and Linear Regression Modeling of a Dataset

Dr. Hasan Mahmud
Associate Professor, Department of CSE
Md. Shihab Shahriar
Lecturer, Department of CSE

September 7, 2023

Contents

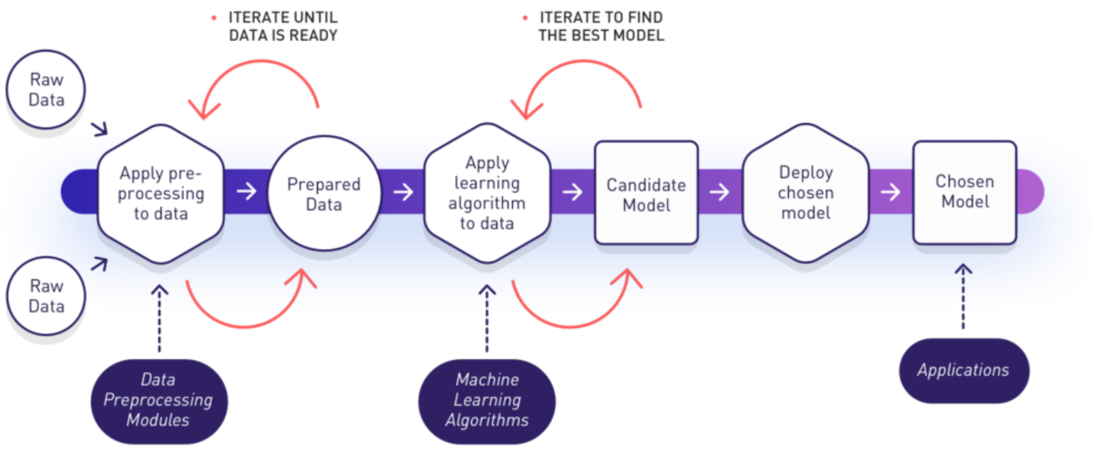
1 Objectives

- To know loading the dataset and split them into training and testing sets
- To know the basic Cross-validation task
- To use the Matplot Library for feature analysis and scores
- To know the use of KNN classifier
- To understand how to generate the Confusion Matrix for result analysis

2 Problem Discussion

2.1 Preprocessing and Visualization

In this lesson you will be implementing the Machine Learning steps which paves the path of training a model. You will be pre-processing the dataset and preparing the dataset. You will also have to visualize different features of the dataset and divide them into training and testing sets.


ITERATE UNTIL DATA IS READY
ITERATE TO FIND THE BEST MODEL
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

2.1.1 Train, Test and Validation sets

You don't want your model to over-learn from training data and perform poorly after being deployed in production. You need to have a mechanism to assess how well your model is generalizing. For this purpose, a testing dataset is usually separated from the data. Next, a validation dataset, while not strictly crucial, is quite helpful to avoid training your algorithm on the same type of data and to evaluate your model effectively.

1. **Training Dataset:** The sample of data used to fit the model.

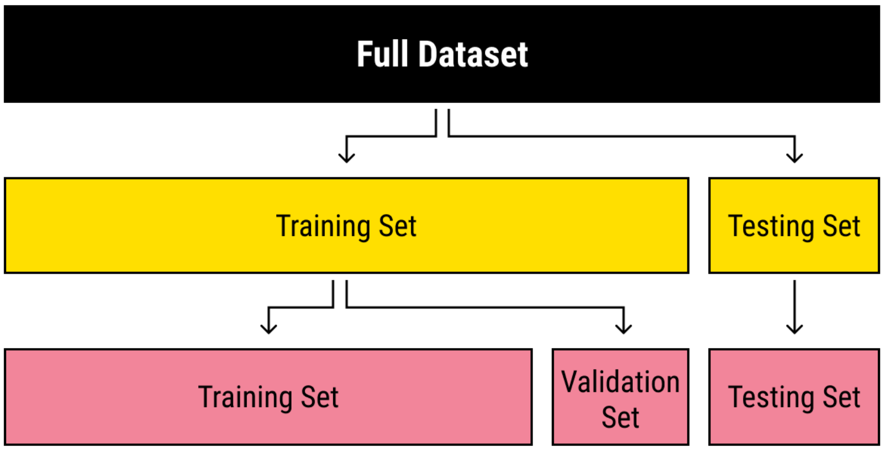
   The actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model sees and learns from this data.

2. **Validation Dataset:** The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

   The validation set is used to evaluate a given model, but this is for frequent evaluation. We, as machine learning engineers, use this data to fine-tune the model hyperparameters. Hence the model occasionally sees this data, but never does it "Learn" from this. We use the validation set results, and update higher level hyperparameters. So the validation set affects a model, but only indirectly. The validation set is also known as the Dev set or the Development set. This makes sense since this dataset helps during the "development" stage of the model.

3. **Test Dataset:** The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

   The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained(using the train and validation sets). The test set is generally what is used to evaluate competing models (For example on many Kaggle competitions, the validation set is released initially along with the training set and the actual test set is only released when the competition is about to close, and it is the result of the the model on the Test set that decides the winner). Many a times the validation set is used as the test set, but it is not good practice. The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world.



3 Linear Regression

At its most basic, linear regression means finding the best possible line to fit a group of data points that seem to have some kind of linear relationship. Linear Regression comes under the category of supervised machine learning algorithms. Regression problems try to predict results within a continuous output i.e. they try to map input variables to some continuous function. Some examples of regression problems are:

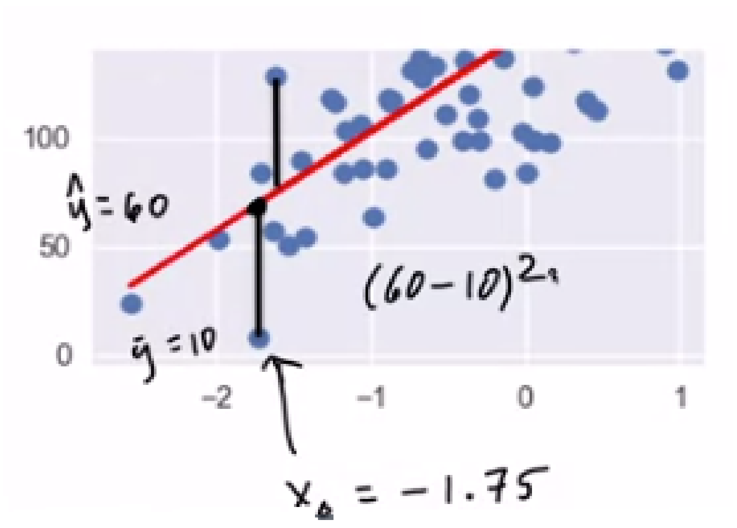
1. Predicting the price of houses given the area, number of rooms, etc.
2. Calculating fare for a taxi depending on the distance, traffic, etc.

In any supervised learning problem, our goal is simple: **Given a training set, we want to learn a function h: X →Y so that h(x) is a good prediction for the corresponding value of y**. Here h(x) is called the hypothesis function.

3.1 Define Hypothesis Funtion

For linear regression, our hypothesis is:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n \tag{1}$$



3.2 Define Cost Funtion

The cost function is defined from the error metric that we choose. There are several error metrics such as:

1. **The mean absolute error** (MAE) is the simplest regression error metric to understand. We'll calculate the residual for every data point, taking only the absolute value of each so that negative and positive residuals do not cancel out. We then take the average of all these residuals.
2. **The mean square error** (MSE) is just like the MAE, but squares the difference before summing them all instead of using the absolute value. We can see this difference in the equation below.

$$MAE = \overset{\text{Divide by the total number of data points}}{\boxed{\frac{1}{n}}} \underset{\text{Sum of}}{\Sigma} \underbrace{\left| \overset{\text{Actual output value}}{\boxed{y}} - \overset{\text{Predicted output value}}{\boxed{\widehat{y}}} \right|}_{\text{The absolute value of the residual}}$$

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

3. **R-squared** is a statistical measure of how close the data are to the fitted regression line. It is generally used to measure the accuracy of our linear model, mathematically :

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

SST is the total sum of squares and SSR is the total sum of squares of residuals.

The cost function is usually defined from MSE as:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cost function for linear regression with multiple features

3.3 Gradient Descent

Gradient descent in our context is an optimization algorithm that aims to adjust the parameters in order to minimize the cost function.

The main update step for gradient descent is:

$$\text{Repeat } \{$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\text{(simultaneously update } \theta_j \text{ for } j = 0, \ldots, n) \}$$

Gradient descent for linear regression with multiple features

So we multiply the derivative of the cost function with the learning rate(α) and subtract it from the present value of the parameters(θ) to get the new updated parameters(θ).

3.4 Plot Cost Function vs Epoch

We can record the cost function at each update step (epoch) to plot this graph.

3.5 Using Numerical Method to Find the Regression Line

The equation of the regression line can be represented as:

$$h(x_i) = \beta_0 + \beta_1 x_i$$

Here,

- h(x_i) represents the predicted response value for ith observation.
- b_0 and b_1 are regression coefficients and represent y-intercept and slope of regression line respectively.

To create our model, we must "learn" or estimate the values of regression coefficients b_0 and b_1. And once we've estimated these coefficients, we can use the model to predict responses!
We define the squared error or cost function, J as:

$$J(\beta_0, \beta_1) = \frac{1}{2n} \sum_{i=1}^{n} \varepsilon_i^2$$

And our task is to find the value of b_0 and b_1 for which J(b_0,b_1) is minimum! Without going into the mathematical details, we can use the equations here:

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

where SS_xy is the sum of cross-deviations of y and x:

$$SS_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} y_i x_i - n\bar{x}\bar{y}$$

and SS_xx is the sum of squared deviations of x:

$$SS_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n(\bar{x})^2$$

4 Data

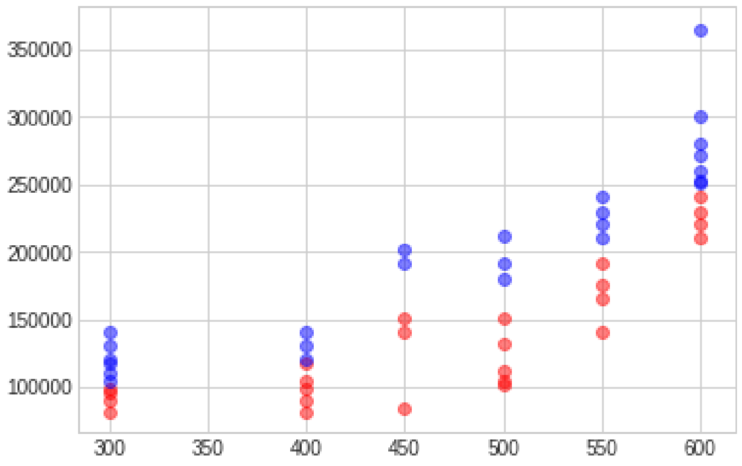
The first step for any machine learning problem is getting the data. There is no machine "learning" if there is nothing to "learn" from. So for this tutorial we will be using a very common dataset for linear regression i.e. the house-price prediction dataset. The dataset can be found here

This is a simple dataset containing housing prices in Portland, Oregon. The first column is the size of the house (in square feet), the second column is the number of bedrooms, and the third column is the price of the house. You might have noticed that we have more than one feature in our dataset (i.e. the house size(in sqft) and the number of rooms) hence we will be looking at multivariate linear regression and the label (y) will be the house price as that is what we are going to be predicting.

5 Tasks

- **Task 01:**
  Load the regression dataset. [Dataset = house_price_data.txt]. The dataset is given as a text file with no headers. You may add the headers and convert the text file to a csv file to load using pandas library.

- **Task 02:**
  Plot the each of the input features against the output feature. [Hint: Here, you can plot two graphs input_feature_1 vs output_feature and input_feature_2 vs output_feature. You may also plot both input features in the same graph may be showing input_feature_2 by color] [Hint: Use Scatterplot]



- **Task 03:**
  Display the statistical information for each of the input features (mean, median, standard deviation and variance) using a built in function of the pandas library.

- **Task 04:**
  Prepare the dataset by normalizing or scaling the feature set. You can use any kind of normalization here like:

  1. Min-Max normalization
  2. Z-score normalization
  3. Decimal Scaling

[Hint: Normalization range -1 to 1 or 0 to 1]
****Note:** Normalization is necessary for scaling a dataset which has largely varying data in order to reduce the time required for training.

- **Task 05:**
  Prepare the training and testing dataset. Split them into a 80-20% ratio size. You may use the built in function train_test_split provided by the sklearn library. You may also do it manually selecting the first 80% and put it in a different array named X_train and put the last 20% data in another array named X_test. Then do the same for Y.

- **Task 06:**
  Implement the hypothesis function.

```python
def h(x,theta):
    # your code here
```

- **Task 07:**
  Implement the cost function.

```python
def cost_function(x, y, theta):
    # your code here
```

- **Task 08:**
  Implement the gradient descent function. You should return the updated theta values and the value from the cost function at found from each epoch.

```python
def gradient_descent(x, y, theta, learning_rate=0.1, num_epochs
    =10):
    # your code here
```

- **Task 09:**
  Find the best fitted line with the gradient descent function

```python
num_epochs = 50
theta, J_all = gradient_descent(x, y, theta, learning_rate,
    num_epochs)
J = cost_function(x, y, theta)
print("Cost: ", J)
print("Parameters: ", theta)
```

- **Task 10:**
  Plot the best fitted line using matplotlib

- **Task 11:**
  Plot graph of cost function vs. epoch.

- **Task 12:**
  Implement the linear regression model with **sklearn** library.