Assignment-1 : Machine Learning
Group Name : Group61536

Raghavarapu Venkateswararao (IMT2015036)
Ghantasala Oormila (IMT2015015)
Alavala Deepika (IMT2015006)

March 18, 2018

Question 1 :

Objective : Given the Housing dataset, compute and compare the parameters learnt from various models - Linear Regression(closed form, Gradient Descent, Newton's Method), Ridge Regression, Lasso Regression.

- First the dataset is loaded and then the data is split into two sets - training set and test set using sklearn's train_test_split library (20% to the test set and remaining to train set).
- Looking at the data, it is observed that there are null values in the column total_bedrooms. So, data cleaning is required.
- The null values in the data are replaced with median values of the corresponding columns using sklearn's Imputer library. Therefore, both the train set and test set are transformed with this strategy.
- To predict median_house_value using the other attributes as inputs to the model (except ocean_proximity which is a categorical attribute).
- **Note** : The attribute ocean_proximity is avoided as an input feature since it is a verbal categorical attribute. We could also encode numerical values to the various categories and send them as input to the model.

1. Linear Regression - Closed form :
   - For fitting a hyperplane, $\mathbf{X}\Theta = \mathbf{Y}$
   - Here, Θ is the coefficient matrix, X is the [1, train_set columns] and Y is the median_house_value matrix.
   - Θ can be evaluated using the formula :

$$\Theta = (X^T X)^{-1} X^T Y$$

- From the Θ obtained, predicted values are computed by substituting in the hyperplane equation on the test set.
- Then, root mean squared error for the test values and the predicted values is calculated using sklearn's library mean_squared_error.
- Root mean squared error obtained for this model = **71140.17329483372**
- Training Time is : **2.19ms**

2. Linear Regression - Gradient Descent

- For fitting a hyperplane, $\mathbf{X}\Theta = \mathbf{Y}$
- Here, Θ is the coefficient matrix, X is the [1, train_set columns] and Y is the median_house_value matrix.
- Data is scaled using the below formula since the model seems to be sensitive to the scale of input features

$$X^{new} = \frac{(X^{old} + \mu_X)}{(X^{old} + \sigma_X)}$$

- Θ is evaluated by minimizing the cost function which is the Mean squared error (MSE) :

$$MSE(X, h_\Theta) = \frac{1}{m}\sum_{i=1}^{m}(\Theta^T.x_i - y_i)^2$$

$$\nabla_\theta MSE(\theta) = \frac{2}{m}X^T.(X.\Theta - Y)$$

- Using the gradient descent formula for finding optimal Θ :

$$\Theta^{new} = \Theta^{old} - \eta\nabla_\Theta MSE(\Theta)_{\Theta=\Theta^{old}}$$

  where η is the learning rate
- Assumed Θ^{old} = zero-matrix, $\eta = 0.1$ and no.of iterations = 1500.
- From the Θ obtained, predicted values are computed by substituting in the hyperplane equation on the test set.
- Then, root mean squared error for the test values and the predicted values is calculated using sklearn's library mean_squared_error.
- Root mean squared error obtained for this model = **110500.02368991509**
- Training Time is : **176ms**

3. Linear Regression - Newton's Method

- For fitting a hyperplane, $\mathbf{X}\Theta = \mathbf{Y}$

- Here, Θ is the coefficient matrix, X is the [1, train_set columns] and Y is the median_house_value matrix.
- Θ is evaluated by minimizing the cost function which is the Mean squared error (MSE) :

$$MSE(X, h_\Theta) = \frac{1}{m}\sum_{i=1}^{m}(\Theta^T.x_i - y_i)^2$$

$$\nabla_\theta MSE(\theta) = \frac{2}{m}X^T.(X.\Theta - Y)$$

- Using Newton's formula for finding optimal Θ :

$$\Theta^{new} = \Theta^{old} - H^{-1}\nabla_\Theta MSE(\Theta)_{\Theta=\Theta^{old}}$$

  where H is the Hessian matrix of the cost function and is computed by the second derivative of the cost function.
- Assumed Θ^{old} = zero-matrix and no.of iterations = 1000.
- It is observed that Hessian matrix (H) is indeed equal to $\frac{1}{m}X^TX$
- From the Θ obtained, predicted values are computed by substituting in the hyperplane equation on the test set.
- Then, root mean squared error for the test values and the predicted values is calculated using sklearn's library mean_squared_error.
- Root mean squared error obtained for this model = **71140.17328336269**
- Training Time is : **638ms**

4. Ridge Regression

- For fitting a hyperplane, $\mathbf{X\Theta = Y}$
- Here, Θ is the coefficient matrix, X is the [1, train_set columns] and Y is the median_house_value matrix.
- Ridge Regression is a regularized version of Linear Regression with a regularization term : $\alpha\sum_{i=1}^{m}\theta_i^2$($l_2$ norm) added to the cost function. Hence the cost function becomes :

$$J(\Theta) = MSE(\Theta) + \frac{\alpha}{2}\sum_{i=1}^{m}\theta_i^2$$

- Θ can be evaluated using the formula :

$$\Theta = (X^TX + \alpha A)^{-1}X^TY$$

  where α is a hyperparameter that controls how much the model has to be regularized. A is an identity matrix with the top left element as 0.

- From the Θ obtained, predicted values are computed by substituting in the hyperplane equation on the test set.
- Then, root mean squared error for the test values and the predicted values is calculated using sklearn's library mean_squared_error.
- To find optimal value for α, root mean squared error is calculated for α varying from 1 to 100. Optimum value is the one with least root mean squared error. Here, $\alpha_{opt} = 372$ with root mean squared error = 71105.94082000348
- Training time using α_{opt} : **2.82ms**
- It is observed that if α is very large, the weights tend to get close to 0 compared to the case where α is comparatively low and as α increases, both training time and mean squared error also increase. (Obtained this conclusion by considering $\alpha = 10000$ and 0.5)

5. Lasso Regression

- For fitting a hyperplane, $\mathbf{X}\Theta = \mathbf{Y}$
- Here, Θ is the coefficient matrix, X is the train set and Y is the median_house_value matrix.
- Lasso Regression is a regularized version of Linear Regression with a regularization term : $\alpha \sum_{i=1}^{m} |\theta_i|$ (l_1 norm) added to the cost function. Hence the cost function becomes :

$$J(\Theta) = MSE(\Theta) + \alpha \sum_{i=1}^{m} |\theta_i|$$

  where α is a hyperparameter.
- Θ and the predicted values for the test data are obtained using sklearn's Lasso library.
- Then, root mean squared error for the test values and the predicted values is calculated using sklearn's library mean_squared_error.
- To find optimal value for α, root mean squared error is calculated for α varying from 1 to 1000. Optimum value is the one with least root mean squared error. Here, $\alpha_{opt} = 847$ with root mean squared error = 71104.576572109
- Training time using α_{opt} : **175ms**
- Root mean squared error obtained for (α =10000) = **75144.06253730216**
  Root mean squared error obtained for (α =0.5) = **71140.13129109403**
- Training Time for $\alpha = 10000$ is : **183ms**
  Training Time for $\alpha = 0.5$ is : **300ms**

- It is observed that if α is very large, the weights tend to get close to 0 compared to the case where α is comparatively low and as α increases, training time decreases and mean squared error increases.

Observations : It is observed that the coefficient matrix Θ and the root mean squared error obtained in 1,3,4,5 is more or less the same whereas in 2, we get the highest error and a different coefficient matrix (might be because of the scaling). Training time is highest for Linear regression - Newton's method and least for Linear regression - closed form method.

Question 2 :

Objective : Given the Iris Dataset, perform Binary Classification, using the models - Nearest Neighbour, Naive-Bayes Classifier, Logistic Regression(Gradient Descent, Newton's method, using Library)

- First the dataset is loaded and then it is split into two sets - training set and test set using sklearn's train_test_split library (20% to the test set and remaining to train set).
- Looking at the data, it is observed that there are no null values. So, data cleaning is not required.
- To predict the class label - species (Virginica, non-virginica).
- X is the train set without the column - species and Y is the species columns encoded 1 for virginica and 0 for non-virginica.
- **Note :** Since it is a Binary Classification problem, we assumed that there are broadly two classes - virginica and non-virginica

1. Nearest Neighbour :
   - No training of the model is required
   - Given a test point, compute the euclidean distances to all the data points, find the point to which the distance is minimum (Nearest Neighbour), the label is predicted to be the label of the nearest neighbour of that point.
   - Euclidean distance between two points X_1, X_2

$$D = \sqrt{\sum_{i=1}^{m} (X_{(1,i)} - X_{(2,i)})^2}$$

   - Accuracy score for the test labels and the predicted labels is computed using sklearn's accuracy_score library

- Accuracy Score = **1.0**
- Training Time = **26.5ms**

2. Naive-Bayes Classifier :

   - The train set is split into two classes - Virginica and Non-Virginica.
   - To maximize the Likelihood Estimate, the μ and Σ of the Multivariate-Gaussian are computed as the sample mean and sample standard deviation of the data.
   - So, the mean and Covariance are computed for each of the classes using library functions.
   - Multivariate Gaussian :

$$G(\mu, \Sigma) = \frac{1}{\Sigma^{\frac{1}{2}}(2\pi)^{\frac{m}{2}}} e^{\frac{-(X-\mu)^T \Sigma^{-1}(X-\mu)}{2}}$$

   - The predicted label is the label of the class for which its Gaussian is maximum.
   - Accuracy Score = **0.6333333333333**
   - Training Time = **20.6ms**

3. Logistic Regression - Gradient Descent :

   - Here, Θ is the coefficient matrix, X is the [1, train_set columns] and Y is the species label matrix.
   - Sigmoid Curve :

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

   - In Logistic Regression, the probabilities of the train set are fit to a sigmoid curve i.e:

$$P(y_i|x_i; \Theta) = \sigma(x_i \Theta)$$

     Maximize the conditional probability :

$$P(Y|X; \Theta) = \prod_{i=1}^{m} P(y_i|x_i; \Theta)$$

     The cost function (Log-Likelihood) :

$$J(\Theta) = \frac{1}{m}(-Y^T.log(P(Y = 1|\Theta)) - (1 - Y^T.log(1 - P(Y = 1|\Theta))))$$

   - Θ is evaluated by maximizing the Likelihood :

$$\nabla_\theta J(\theta) = \frac{1}{m} X^T.(\sigma(X.\Theta) - Y)$$

- Using the gradient descent formula for finding optimal Θ :

$$\Theta^{new} = \Theta^{old} - \eta \nabla_{\Theta} J(\Theta)_{\Theta=\Theta^{old}}$$

  where η is the learning rate
- Assumed Θ^{old} = zero-matrix, $\eta = 0.1$ and no.of iterations = 1500.
- From the Θ obtained, compute $\sigma(X_{test}\Theta)$ and then rounding-off to 0 or 1 to obtain the predicted label.
- Accuracy Score = **1.0**
- Training Time = **76.5ms**

4. Logistic Regression - Newton's method :

   - Using Newton's formula for finding optimal Θ :

$$\Theta^{new} = \Theta^{old} - H^{-1} \nabla_{\Theta} J(\Theta)_{\Theta=\Theta^{old}}$$

     where H is the Hessian matrix of the cost function and is computed by the second derivative of the cost function.
   - Assumed Θ^{old} = zero-matrix and no.of iterations = 1000.
   - It is observed that Hessian matrix (H) is equal to

$$(\frac{1}{m} X^T X).diagonal(1 - \sigma(X\Theta)))$$

   - From the Θ obtained, compute $\sigma(X_{test}\Theta)$ and then rounding-off to 0 or 1 to obtain the predicted label.
   - Accuracy Score = **0.8**
   - Training Time = **139ms**

5. Logistic Regression - Using Library :

   - X is the train set and Y is the species label.
   - Using sklearn's LogisticRegression library, the predicted labels and their respective probabilities are obtained.
   - Accuracy Score = **1.0**
   - Training Time = **23.7ms**

Observations : Methods 3,4 give the highest training time since there are many iterations. Naive-Bayes has lowest training time and Accuracy score. Naive-Bayes is less accurate may be because of rounding off the predicted probabilities. High value for accuracy means the ratio of correctly predicted observations is high.

Question 3 :

Objective : Given D51 Dataset, perform Explanatory Data Analysis (EDA) and then Binary Classification using the models - Nearest Neighbour, Naive-Bayes Classifier, Logistic Regression(Gradient Descent, Newton's method, using Library)

EDA :

- First the dataset is loaded and looking at the head of the data, it is observed that each row represents data related to one instance. There are 14 attributes Unnamed:0,F0,F1,....F11,Label.
- There are 2254 instances in the dataset, comparably small for a machine learning problem.
- None of the attributes has null values since all the attributes have exactly 2254 non-null values.
- All attributes are numerical. But the Label attribute seems to be categorical since values in that column seem to be repetitive.
- The column Unnamed:0 seems to be the serial number row number for the data.
- Only two categories exist -1 and 0 and have count 1802 and 452 respectively.
- The columns F0,F1,....F11 seem to take value v where $|v| < 1$.
- Columns F0,F4,F3,F5 take both positive and negative values
- Columns F1,F6,F7,F8,F9,F10,F11 take only positive values
- Column F2 takes only negative values
- std shows standard deviation which measures how dispersed the values are.
- The 25th percentile, 50th percentile, 75th percentile that indicates the value below which a given percentage of observations in a group of observations falls.
- From the histograms (range vs instances) of each column : F10, F11, F6, F7, F8 seem to be tail heavy since they extend much farther to the right of the median than to the left. This makes detection of patterns difficult. Desired shape : bell-shaped distribution

- The correlation matrix gives the relation between each of the attributes.
- The correlation coefficient ranges from 1 to 1. When it is close to 1, it means that there is a strong positive correlation
- When the coefficient is close to 1, it means that there is a strong negative correlation. Coefficients close to zero mean that there is no linear correlation. For example, the correlation coefficient(F0,F2) is close to 1 which implies F0 increases linearly with an increase in F2 and vice-versa
- Plotting the scatter plots, there seems to be a linear correlation between some of the attributes. Other correlations seem to be non linear or no correlation at all

Classification Experiments :

- First the dataset is loaded and then it is split into two sets - training set and test set using sklearn's train_test_split library (20% to the test set and remaining to train set).
- Looking at the data, it is observed that there are no null values. So, data cleaning is not required.
- To predict the class label - Label (0, 1).
- X is the train set data without the columns 'Unnamed: 0' and 'Label' and Y is the 'Label' column
- **Note :** All the atributes are considered for classification. Neither of them are dropped if the correlation coefficient is low.

1. Nearest Neighbour :
   - No training of the model is required
   - Given a test point, compute the euclidean distances to all the data points, find the point to which the distance is minimum (Nearest Neighbour), the label is predicted to be the label of the nearest neighbour of that point.
   - Euclidean distance between two points X_1, X_2

$$D = \sqrt{\sum_{i=1}^{m} (X_{(1,i)} - X_{(2,i)})^2}$$

- Accuracy, precision, recall, and AUC for the test labels and the predicted labels is computed using sklearn's accuracy_score, precision_score, recall_score, metrics library repectively.
- Accuracy Score = **0.8713968957871396**
- Precision Score = **0.9054054054054054**
- Recall Score = **0.935754189944134**
- F_measure is computed using the formula :

$$F\_measure = 2 * (\frac{precision * recall}{precsion + recall})$$

  F_measure = **0.9203296703296703296703**
- AUC (Area under the curve) with label 0 = **0.22029494803868566**
  AUC (Area under the curve) with label 1 = **0.7797050519613143**
- Training Time = **6.44s**

2. Naive-Bayes Classifier :

- The train set is split into two classes - Label=0 and Label=1.
- To maximize the Likelihood Estimate, the μ and Σ of the Multivariate-Gaussian are computed as the sample mean and sample standard deviation of the data.
- So, the mean and Covariance are computed for each of the classes using library functions.
- Multivariate Gaussian :

$$G(\mu, \Sigma) = \frac{1}{\Sigma^{\frac{1}{2}}(2\pi)^{\frac{m}{2}}} e^{\frac{-(X-\mu)^T \Sigma^{-1}(X-\mu)}{2}}$$

- The predicted label is the label of the class for which its Gaussian is maximum.
- Accuracy Score = **0.8337028824833703**
- Precision Score = **0.924924924924925**
- Recall Score = **0.8603351955307262**
- F_measure is computed using the formula :

$$F\_measure = 2 * (\frac{precision * recall}{precsion + recall})$$

  F_measure = **0.8914616497829233**
- AUC (Area under the curve) with label 0 = **0.2042410043851745**
  AUC (Area under the curve) with label 1 = **0.7957589956148254**
- Training Time = **92.7ms**

3. Logistic Regression - Gradient Descent :

   - Here, Θ is the coefficient matrix, X is the [1, train_set columns] and Y is the Label matrix.
   - Sigmoid Curve :
   $$\sigma(x) = \frac{1}{1 + e^{-x}}$$
   - In Logistic Regression, the probabilities of the train set are fit to a sigmoid curve i.e:
   $$P(y_i|x_i; \Theta) = \sigma(x_i\Theta)$$
   Maximize the conditional probability :
   $$P(Y|X; \Theta) = \prod_{i=1}^{m} P(y_i|x_i; \Theta)$$
   The cost function (Log-Likelihood) :
   $$J(\Theta) = \frac{1}{m}(-Y^T.log(P(Y = 1|\Theta)) - (1 - Y^T.log(1 - P(Y = 1|\Theta))))$$
   - Θ is evaluated by maximizing the Likelihood :
   $$\nabla_\theta J(\theta) = \frac{1}{m} X^T.(\sigma(X.\Theta) - Y)$$
   - Using the gradient descent formula for finding optimal Θ :
   $$\Theta^{new} = \Theta^{old} - \eta \nabla_\Theta J(\Theta)_{\Theta=\Theta^{old}}$$
   where η is the learning rate
   - Assumed Θ^{old} = zero-matrix, $\eta = 0.1$ and no.of iterations = 1500.
   - From the Θ obtained, compute $\sigma(X_{test}\Theta)$ and then rounding-off to 0 or 1 to obtain the predicted label.
   - Accuracy Score = **0.8048780487804879**
   - Precision Score = **0.8026905829596412**
   - Recall Score = **1.0**
   - F_measure is computed using the formula :
   $$F\_measure = 2 * (\frac{precision * recall}{precsion + recall})$$
   F_measure = **0.8905472636815921**
   - AUC (Area under the curve) with label 0 = **0.4731182795698925**
   AUC (Area under the curve) with label 1 = **0.5268817204301075**

- Training Time = **1.72s**

4. Logistic Regression - Newton's method :
   - Using Newton's formula for finding optimal Θ :

     $$\Theta^{new} = \Theta^{old} - H^{-1}\nabla_{\Theta}J(\Theta)_{\Theta=\Theta^{old}}$$

     where H is the Hessian matrix of the cost function and is computed by the second derivative of the cost function.
   - Assumed Θ^{old} = zero-matrix and no.of iterations = 1000.
   - It is observed that Hessian matrix (H) is equal to

     $$(\frac{1}{m}X^{T}X).diagonal(1-\sigma(X\Theta)))$$

   - From the Θ obtained, compute $\sigma(X_{test}\Theta)$ and then rounding-off to 0 or 1 to obtain the predicted label.
   - Accuracy Score = **0.8691796008869179**
   - Precision Score = **0.8863049095607235**
   - Recall Score = **0.9581005586592178**
   - F_measure is computed using the formula :

     $$F\_measure = 2 * (\frac{precision * recall}{precsion + recall})$$

     F_measure = **0.9208053691275168**
   - AUC (Area under the curve) with label 0 = **0.25750886045533733**
     AUC (Area under the curve) with label 1 = **0.7424911395446627**
   - Training Time = **1.52s**

5. Logistic Regression - Using Library :
   - X is the train set and Y is the Label (0 or 1).
   - Using sklearn's LogisticRegression library, the predicted labels and their respective probabilities are obtained.
   - Accuracy Score = **0.8691796008869179**
   - Precision Score = **0.8863049095607235**
   - Recall Score = **0.9581005586592178**
   - F_measure is computed using the formula :

     $$F\_measure = 2 * (\frac{precision * recall}{precsion + recall})$$

     F_measure = **0.9208053691275168**

- AUC (Area under the curve) with label 0 = **0.25750886045533733**
  AUC (Area under the curve) with label 1 = **0.7424911395446627**
- Training Time = **40.7ms**

Observations : Accuracy score is more for methods 1,4,5. Precision Score is more for methods 1,2. This means these methods return more relavent results than irrelavent ones. Recall score is maximum for Logistic Regression - gradient descent. This means that the model returns most of the relavent results. F_measure is more for methods 1,4,5 which shows that they have more perfect values for precision and recall. Training time is highest for Nearest Neighbour since it is computationally inefficient and no training is done for the model. Logistic regression using sklearn's library is more efficent in terms of training time (least).

Note :

The datasets for respective questions should be packed in a folder named Assignment_data and should be put in the Group61536 folder.