Assignment-1

Supervisor: Prof. DineshBabu
Advisor: Chinchu Thomas

Prepared By- **Predators**

1. Mahesh Tolani(MT2017064)
2. Kaushal Mehta(MT2017059)
3. Shashikant(MT2017104)

Question 1:
Dataset: Housing Dataset

I) Linear Regression (Closed Form):

1. In this first we loaded the data set then we converted ocean proximity feature from string to integer.
2. Then we checked if any feature contains null value or not.
3. After that we performed Column Standardization.
4. Then Sliced data-frame into Features and Targets.
5. Then we converted data into Vector Form.
6. Then sliced data into training and test set.
7. Calculated coefficients using closed form solution

$$w = (X^T X)^{-1} X^T y$$

8. Destandardization of Labels
9. RMSE (root mean square error): 71316.5233825
10. Training Time: 0.005331 seconds

II) Linear Regression (Gradient Descent):

- Load the dataset

  Step 2 to 6 as closed form
- Calculate Gradient Descent

Formula1:

$$\theta_{new} = \theta_{old} - \eta \frac{\mathrm{d}}{\mathrm{d}\theta} \mathcal{J}(\theta)$$

Formula2:

$$\frac{\partial}{\partial w_i} \mathcal{J}(w) = \frac{2}{m} \sum_{i=1}^{i=N} (w^T . x^i - y_i) x_j^{(i)}$$

Cost Function:

$$\mathcal{J}(m,c) = \frac{1}{2}\sum_{i=1}^{i=N}[y_i - (mx_i + c)]^2$$

Error:

e=**X**w-y

- By taking Root mean Square Error: 70576.1753061
- Training Time: 0.7299243940 seconds

III) Linear Regression with Newton's Method:

- Step 1 to 6 as closed form method
- By using Newton's Method

Prediction:

$$y_{model} = Xw$$

Error:

e=**prediction-y**

cost = 1/(2*m) * (error(T), error)

Hessian Matrix:

$$\mathcal{H} = \begin{bmatrix} \frac{\partial \mathcal{J}^2}{\partial\theta_1\partial\theta_1} & \frac{\partial \mathcal{J}^2}{\partial\theta_1\partial\theta_2} \\ \frac{\partial \mathcal{J}^2}{\partial\theta_2\partial\theta_1} & \frac{\partial \mathcal{J}^2}{\partial\theta_2\partial\theta_2} \end{bmatrix}$$

Formula for new Theta:

$$\theta_{new} = \theta_{old} - \mathcal{H}^{-1}\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{J}(\theta)$$

- Pass the relevant variables to the function get new values back.
- RMSE: 71316.5234307
- Training Time: 0.0110297203 Seconds

IV) Ridge Regression:

- Step 1 to 6 as per the closed form
- Ridge Regression’s Cost Function

$$J(\theta) = \text{MSE}(\theta) + \alpha\frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

$$\hat{\theta} = (\mathbf{X}^T \cdot \mathbf{X} + \alpha\mathbf{A})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

- RMSE with Ridge Regression: 71230.8387212
- Training time: 0.008527755737 seconds

V) Lasso Regression:

- Step 1 to 6 as per the closed form
- Lasso Regression’s Cost Function

$$J(\theta) = \text{MSE}(\theta) + \alpha\sum_{i=1}^{n} | \theta_i |$$

$$\hat{\theta} = (\mathbf{X}^T \cdot \mathbf{X} + \alpha\mathbf{A})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

- RMSE with Lasso Regression:71316.42
- Training Time: 0.06814599037 seconds

Question 2:
Dataset: Iris Dataset

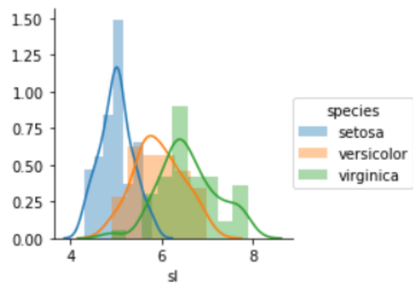
1) **Nearest Neighbor:**
- Load the dataset.
- Find the Euclidian Distance between one point to all other training points and set Minimum from all such distances.
- Compute distance from the test data to all training data.
- Find nearest Neighbor and assign a label accordingly.
- Accuracy: 100%
- Time Running: 0.022092103958 second

2) **Naive Bayes:**
- Load the dataset.
- Verify dataset is balanced or not.
- Check for null value presence in any feature.
- For Statistical Measures like mean, standard deviation, we created function to find mean and standard deviation of all features for given class
- To calculate Probability used Gaussian Function:

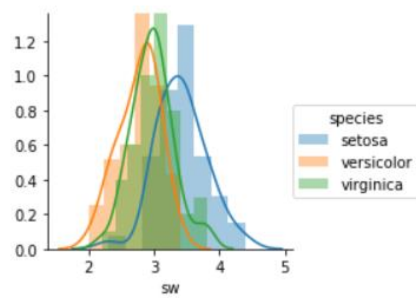
$$p(x\ ;\ \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Slice Training data and test data for verification
- Separate data by class.
- Define a function to predict the class label.
- Accuracy: 0.96
- Training Time: 0.003 seconds

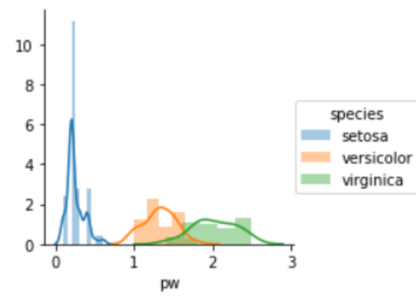
- Plotted the graph(Histogram) for each class
  For Sepal Length



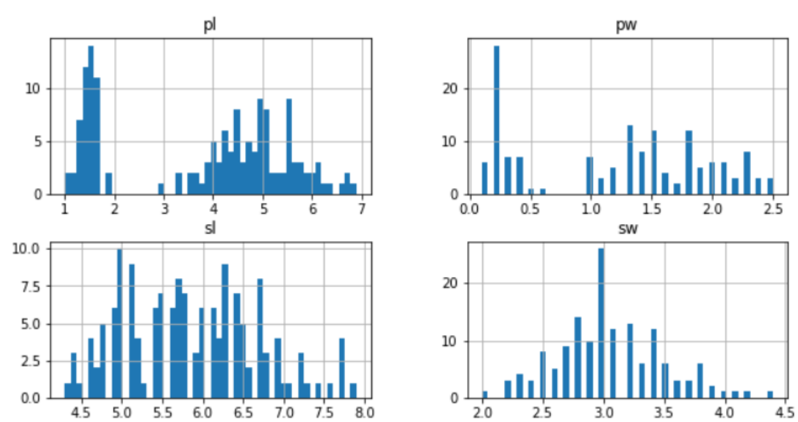
For Sepal Width



For Petal Width : Most Accurate result (Setosa is distinguished for others)



Histogram for Whole Dataset



Based on Evaluation of algorithm Accuracy Score is: 0.9399999

III) **Logistic Regression Gradient Descent (One vs All):**

1. Load the dataset.
2. Performed Binary Classification on 3 sets
   1) irisSet: (Versicolor and Virginica as one set and Setosa as another)
   2) irisVe: (Setosa and Virginica as one set and Versicolor as another)
   3) irisVi: (Versicolor and Setosa as one set and Virginica as another)
3. Slicing train data for setosa, versicolor and virginica and test data for common for all sets.
4. Usage of sigmoid Function for Logistic Regression:

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

   - ➔ For finding Log Likelihood we derived L(w) where w is a vector of parameters want to learn for which problem class label is maximize.
   - ➔ Where u=w(transpose)x=scores, Target is yi(labels) and P(xi)=Prediction
   - ➔ Equation for Log Likelihood:

$$l(w) = \log(p(Y \mid X; w)) = \sum_{i=1}^{N} \log\left(\frac{1}{1+e^{w^T x_i}}\right) + \sum_{i=1}^{N} y_i w^T x_i = -\sum_{i=1}^{N} \log(1+e^{w^T x_i}) + \sum_{i=1}^{N} y_i w^T x_i$$

   - ➔ By taking Derivative of log likelihood we get Gradient.

$$\frac{\partial}{\partial w_j} l(w) = \frac{\partial}{\partial w_j} \log(p(Y \mid X; w)) = -\sum_{i=1}^{N} p(x_i) x_{ij} + \sum_{i=1}^{N} y_i x_{ij} = \sum_{i=1}^{N} (y_i - p(x_i)) x_{ij}$$

   - ➔ Output Error= Target-Prediction
   - ➔ Applied Gradient Descent:

     Weights = Weights + Learning rate*Gradient

$$w_{new} = w_{old} + \eta \frac{\partial l(w)}{\partial w}$$

5. Call Functions of each classification
6. Accuracy of Logistic Regression with gradient Descent:96%
7. Training Time: 7.089 Seconds

IV) **Logistic Regression Newton's Method (One vs All):**

- Steps 1 to 3 same as Logistic Regression's Gradient Descent
- Usage of sigmoid Function for Logistic Regression:

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

➔ For finding Log Likelihood we derived L(w) where w is a vector of parameters want to learn for which problem class label is maximize.

➔ Where u=w(transpose)x=scores, Target is yi(labels) and P(xi)=Prediction

➔ Equation for Log Likelihood:

$$l(w) = \log(p(Y \mid X; w)) = \sum_{i=1}^{N} \log\left(\frac{1}{1+e^{w^T x_i}}\right) + \sum_{i=1}^{N} y_i w^T x_i = -\sum_{i=1}^{N} \log(1+e^{w^T x_i}) + \sum_{i=1}^{N} y_i w^T x_i$$

➔ Initialize the parameter values, score(h(x))=w(Transpose)x and prediction= sigmoid(scores)

➔ sigma=prediction*(1-prediction)

➔ Hessian=Features*(sigma*features)

➔ Then take inverse of hessian

➔ Apply Gradient Descent

$$w_{new} = w_{old} + \eta \frac{\partial l(w)}{\partial w}$$

Where $$\eta = \frac{1}{\mathcal{H}_{old}}$$

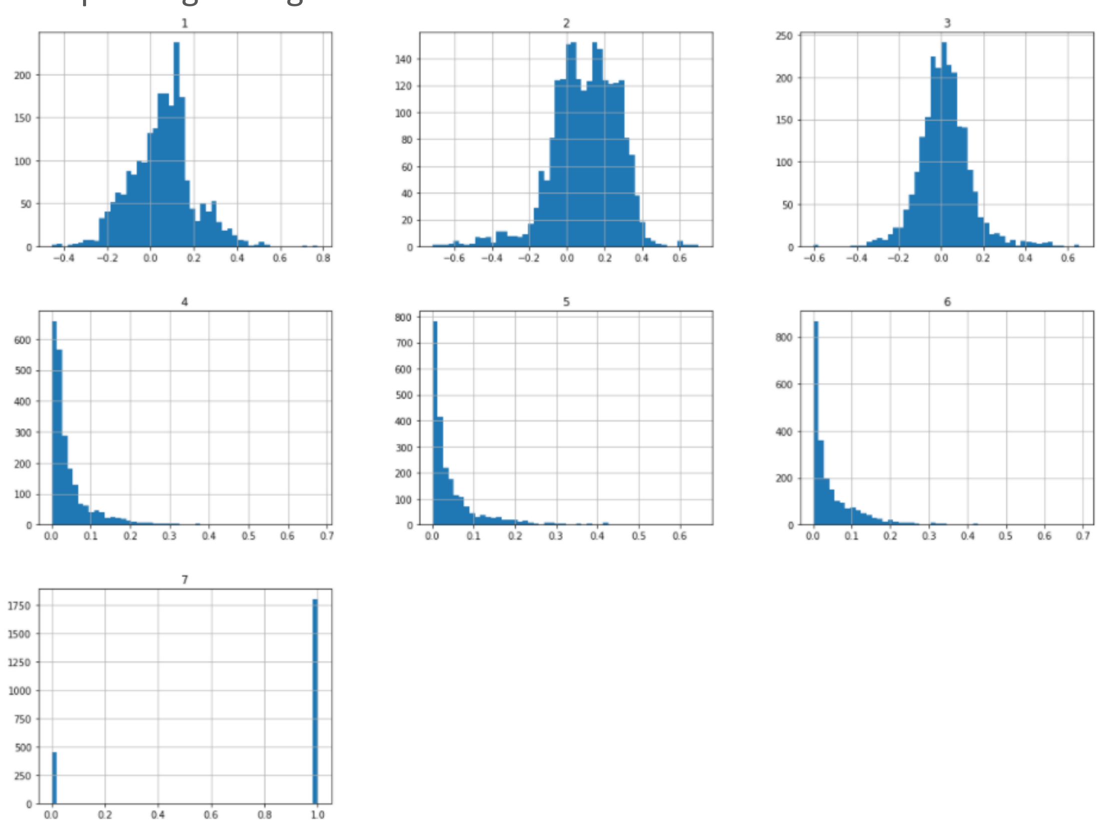
- Call Functions of each classification
- Accuracy of Logistic Regression with Newton's Method:96%
- Training Time:0.256 seconds

V) **Logistic Regression (Library):**

- Load the dataset.
- Split Training data and Test data.
- Apply scikit Library.
- Accuracy:100%
- Training Time:0.0020058155059 seconds

Question 3:
Exploratory Data Analysis

- In this part we will try to get insights in the data. We will also visualize it to see different relationships between feature.
- Load the data set.
- Delete the column because that is useless numbering.
- Count the number of labels
- Then plotting Histogram of features and labels



- Then We created correlation Matrix and sort the value of matrix in descending order.
- Features which are not correlated with any of the other features can be removed but we have kept them.
- Then performed for checking for null values
- Then divided data to features and labels
- Then split data into train and test data
- Logistic Regression using Library:

  Execution time: 0.0040111541748046875 seconds
  Accuracy using scikit learn Logistic Regression: 84.0354767184
  Precision: 0.825499664121
  Recall: 0.840354767184
  F1_score: 0.814330557418
  Roc_Auc_score: 0.642110000621

- **Using Nearest Neighbor Classification:**

  - ➔ Load the dataset.
  - ➔ Find the Euclidian Distance between one point to all other training points and set Minimum from all such distances.
  - ➔ Compute distance from the test data to all training data.

Find nearest Neighbor and assign a label accordingly.

  - ➔ Accuracy: 80.487%
  - ➔ Precision: 0.787838626987
  - ➔ Execution time: 3.940230 seconds
  - ➔ Recall: 0.804878804878
  - ➔ F1_score: 0.793996582985
  - ➔ Roc_Auc_score: 0.645431125458

- **Using Logistic Regression using gradient descent:**

  - ➔ Usage of sigmoid Function for Logistic Regression:

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

  - ➔ For finding Log Likelihood we derived L(w) where w is a vector of parameters want to learn for which problem class label is maximize.
  - ➔ Where u=w(transpose)x=scores, Target is yi(labels) and P(xi)=Prediction
  - ➔ Equation for Log Likelihood:

$$l(w) = \log(p(Y \mid X; w)) = \sum_{i=1}^{N} \log\left(\frac{1}{1+e^{w^T x_i}}\right) + \sum_{i=1}^{N} y_i w^T x_i = -\sum_{i=1}^{N} \log(1+e^{w^T x_i}) + \sum_{i=1}^{N} y_i w^T x_i$$

  - ➔ By taking Derivative of log likelihood we get Gradient.

$$\frac{\partial}{\partial w_j} l(w) = \frac{\partial}{\partial w_j} \log(p(Y \mid X; w)) = -\sum_{i=1}^{N} p(x_i) x_{ij} + \sum_{i=1}^{N} y_i x_{ij} = \sum_{i=1}^{N} (y_i - p(x_i)) x_{ij}$$

  - ➔ Output Error= Target-Prediction
  - ➔ Applied Gradient Descent:

    Weights = Weights + Learning rate*Gradient

$$w_{new} = w_{old} + \eta \frac{\partial l(w)}{\partial w}$$

  - ➔ Call the function that is defined
  - ➔ Accuracy: 84.0354767184 %
  - ➔ Precision: 0.825499664121
  - ➔ Execution time: 0.5254285335540771 seconds
  - ➔ Recall: 0.840354767184
  - ➔ F1_score: 0.814330557418
  - ➔ Roc_Auc_score: 0.642110000621

- **Logistic Regression using Newton's Method:**

  - ➔ Usage of sigmoid Function for Logistic Regression:

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

  - ➔ For finding Log Likelihood we derived L(w) where w is a vector of parameters want to learn for which problem class label is maximize.
  - ➔ Where u=w(transpose)x=scores, Target is yi(labels) and P(xi)=Prediction
  - ➔ Equation for Log Likelihood:

    $$l(w) = \log(p(Y \mid X; w)) = \sum_{i=1}^{N} \log\left(\frac{1}{1+e^{w^T x_i}}\right) + \sum_{i=1}^{N} y_i w^T x_i = -\sum_{i=1}^{N} \log(1+e^{w^T x_i}) + \sum_{i=1}^{N} y_i w^T x_i$$

  - ➔ By taking Derivative of log likelihood we get Gradient.

    $$\frac{\partial}{\partial w_j} l(w) = \frac{\partial}{\partial w_j} \log(p(Y \mid X; w)) = -\sum_{i=1}^{N} p(x_i) x_{ij} + \sum_{i=1}^{N} y_i x_{ij} = \sum_{i=1}^{N} (y_i - p(x_i)) x_{ij}$$

  - ➔ Initialize the parameter values, score(h(x))=w(Transpose)x and prediction= sigmoid(scores)
  - ➔ sigma=prediction*(1-prediction)
  - ➔ Hessian=Features*(sigma*features)
  - ➔ Then take inverse of hessian
  - ➔ Apply Gradient Descent

    $$w_{new} = w_{old} + \eta \frac{\partial l(w)}{\partial w}$$

    Where $$\eta = \frac{1}{\mathcal{H}_{old}}$$

  - ➔ Call Functions of each classification
  - ➔ Accuracy: 83.5920177384 %
  - ➔ Precision: 0.827418222724
  - ➔ Execution time: 0.13586044311523438 seconds
  - ➔ Recall: 0.835920177384
  - ➔ F1_score: 0.798542600875
  - ➔ Roc_Auc_score: 0.609690235272

- **Gaussian Naive Bayes Classifier:**

  - ➔ Load the dataset.
  - ➔ Delete features which are irrelevant.
  - ➔ Define a function to find mean and standard deviation of all features for class.
  - ➔ Verify dataset is balanced or not.
  - ➔ Check for null value presence in any feature.
  - ➔ For Statistical Measures like mean, standard deviation, we created function to find mean and standard deviation of all features for given class
  - ➔ To calculate Probability used Gaussian Function:

$$p(x\ ;\ \mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - ➔ Slice Training data and test data for verification
  - ➔ Separate data by class.
  - ➔ Define a function to predict the class label.
  - ➔ Accuracy: 81.9892473118 %
  - ➔ Precision: 0.818136564617
  - ➔ Execution time: 0.004064798355102539 seconds
  - ➔ Recall: 0.819892473118
  - ➔ F1_score: 0.818986399496
  - ➔ Roc_Auc_score: 0.716717107982