MACHINE LEARNING

IDEATORS

G.SRAVYA(IMT2015014)
DASARI SOWMYA(IMT2015507)
E.LAXMI NIHARIKA(IMT2014017)

Question 1:

DataPreparation

1)Read the data set.Then data.head() gives the first 5 data rows for observing the format of data.

2)Split the new data into train and test sets in 80:20 ratio.

3)We separated train set and test set into respective feature set and label set.

4)Data Cleaning : We then fill the missing values and Nan in train and test data using imputer function with median as strategy.

(i) Linear Regression using closed form :

The cost function J is

$$J = (XW - Y)^T.(XW - Y)$$

where X is the training data,W is the parameters and Y is the train labels.Equating gradient of J to zero gives the below result :

$$W = (X^T.X)^{-1}.X^T.Y$$

The above obtained W is used to predict the score of test data.

predicted_labels = test_set.W

The results obtained are :

parameters : W =
```
[[-3.57822423e+06]
 [-4.26323917e+04]
 [-4.24500719e+04]
 [ 1.18280965e+03]
 [-8.18797708e+00]
 [ 1.16260128e+02]
 [-3.84922131e+01]
 [ 4.63425720e+01]
 [ 4.05384044e+04]]
```

Training time : CPU times: user 8 ms, sys: 12 ms, total: 20 ms Wall time: 8.92 ms

Root-Mean-Square-Error : 71140.17328337362

(ii)Linear Regression using gradient-descent method :

The equation used in gradient descent is

$$\theta_{new} = \theta_{old} - \eta. \frac{\partial J}{\partial \theta} \Big|_{\theta = \theta old}$$

where J is the cost function for multivariate and θ is the parameters to be estimated.

$$J = (X\theta - Y)^T.(X\theta - Y)$$

where X is the training data,θ is the parameters and Y is the train labels.

Here we have assumed initial θ and the learning rate (η) and iterated it over to find the optimised paramaters θ_{opt}.

Then,we predicted the test score by applying the model onto the test set.

$$predicted\_score = test\_set.\theta_{opt}$$

From the given train data we obtained the below result : The below θ is obtained after 100 iteration and the initial η = 0.1.

parameters : θ_{opt} =

```
([[52398.75443688],
[23408.7678057 ],
[26726.04940337],
[28241.05591177],
[58181.68077111],
[30513.25621529],
[20400.85395255],
[32831.85390085],
[27523.47982497]])
```

Training time : CPU times: user 88 ms, sys: 116 ms, total: 204 ms Wall time: 74.4 ms

Root-Mean-Square-Error : 113785.83881909902

(iii)**Linear Regression using Newton's method :** Newton's method is similar to gradient-descent but here instead of learning rate(η) we use the hessian matrix to estimate the parameters.The equation used is

$$\theta_{new} = \theta_{old} - (H)^{-1}. \left.\frac{\partial J}{\partial \theta}\right|_{\theta=\theta old}$$

where J is the cost function for multivariate and θ is the parameters to be estimated.

$$J = (X\theta - Y)^T.(X\theta - Y)$$

and H is the hessian matrix.It is square matrix of the second order partial derivatives of the cost function.

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

$$\mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

It can be obtained by second order derivative of J.So,

$$H = (2/m)*(X^T.X)$$

Here we have assumed initial θ iterated it over to find the optimised paramaters θ_{opt}.

Then,we predicted the test score by applying the model onto the test set.

$$predicted\_score = test\_set.\theta_{opt}$$

From the given train data we obtained the below result : The below θ is obtained after 100 iteration.

paramters : θ_{opt} =
```
[[-3.57822423e+06],
[-4.26323917e+04],
[-4.24500719e+04],
[ 1.18280965e+03],
[-8.18797708e+00],
[ 1.16260128e+02],
[-3.84922131e+01],
[ 4.63425720e+01],
[ 4.05384044e+04]]
```

Training time : CPU times: user 244 ms, sys: 264 ms, total: 508 ms Wall time: 145 ms

Root-Mean-Square-Error : 71140.17328336267

(iv)**Ridge Regression :**Ridge is a regression analysis method that performs both parameter selection and regularisation in order to enhance the prediction accuracy and interpretability of the stastical model it produces.

Cost function J is:

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

where MSE is the mean square error and α is the reguralisation factor and θ is the parameter vector.

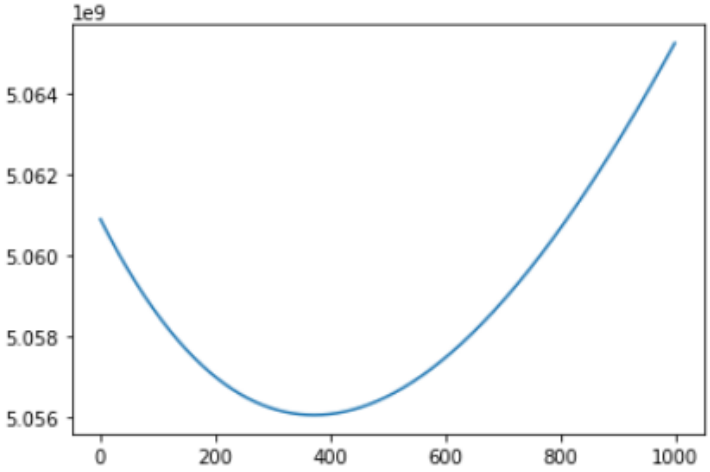
To estimate the parameters

$$\theta = (X^T.X + \alpha I)^{-1}.X^T.Y$$

We tried to predict minimal cost on varying lambda from 1 to 1000

Following are the observations:

In the below graph, X axis represents lambda, y axis represents root mean square error



Optimal lambda estimated: 372

Least root mean square error: 71105.9408285

Optimal parameters: [[-3.29475358e+06] [-3.93196062e+04] [-3.93570904e+04] [1.23538914e+03] [-8.44912825e+00] [1.11346091e+02] [-3.86477210e+01] [5.42026677e+01] [4.06827180e+04]]

Training time: CPU times: user 524 ms, sys: 24 ms, total: 548 ms Wall time: 294 ms

(v) **Lasso Regression :** Lasso is a regression analysis method that performs both variable selection and regularisation in order to enhance the prediction accuracy and interpretability of the stastical model it produces.

Cost function J is :

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

where MSE is the mean square error and α is the reguralisation factor and θ is the parameter vector.
We have used the library function Lasso() from sklearn.linear_model to estimate the parameters and predict the score.
The results obtained are :
parameters : coefficients =

```
[-4.26286808e+04, -4.24466049e+04,  1.18286158e+03,
-8.18829122e+00,1.16254615e+02, -3.84923721e+01,  4.63513818e+01,
4.05385728e+04])
```

Training time : CPU times: user 248 ms, sys: 0 ns, total: 248 ms Wall time: 249 ms

Root-Mean-Square-Error : 71140.0893237417

Question 2 :

(i)**Nearest Neighbour :** We first divided the whole dataset into train data and test data in 80:20 ratio.

We then modified class labels in train set as 1, 0 for Verginica and non verginica classes.

Then, for every test point in test data, we computed eucidean distance with every datapoint in train data. Then label of test point is assigned as label of least distance datapoint label.

Following are the observations:

Accuracy score: 1.0

Training time: CPU times: user 40 ms, sys: 0 ns, total: 40 ms`Wall time: 37.1 ms`

(ii)**Naive Bayes Classifier :** We use multivariate guassian to predict the probabilities P(x/Class = C_i).To predict the label of given test set,we calculate P(x/Class = C_i) for all the classes and assign the class with maximum probability.

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}$$

The results obtained are :

Training time: CPU times: user 8 ms, sys: 0 ns, total: 8 ms Wall time: 18.5 ms

Accuracy : 0.9666666666666667

(iii)**Logistic Regression(Gradient Descent):**The equation used in gradient descent is

$$\theta_{new} = \theta_{old} - \eta . \frac{\partial J}{\partial \theta} \Big|_{\theta=\theta old}$$

where J is the cost function for multivariate and θ is the parameters to be estimated.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{M}(H_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})^2$$

$$H_\theta(\boldsymbol{x}) = \theta^T\boldsymbol{x}.$$

Here we have assumed initial θ and the learning rate (η) and iterated it over to find the optimised paramaters θ_{opt}.

Then,we obtain the probabilities of the test set by substituting the estimated theta in the sigmoid function and classify them().

predicted_labels = probability>0.5

From the given train data we obtained the below result : The below θ is obtained after 100 iteration and the initial η = 0.1.

parameters : θ_{opt} =

```
[[-1.34221238]
[-2.77957351]
[-2.69385308]
[ 5.18784967]
[ 3.719677  ]]
```

Training time : CPU times: user 20 ms, sys: 0 ns, total: 20 ms Wall time: 18.6 ms

Accuracy-score: 0.7666666666666667

(iii)**Logistic Regression(Newton's Method):**The equation used in gradient descent is

$$\theta_{new} = \theta_{old} - H^{-1}. \left.\frac{\partial J}{\partial \theta}\right|_{\theta=\theta old}$$

where J is the cost function for multivariate and θ is the parameters to be estimated.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{M}(H_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})^2$$

$$H_\theta(\boldsymbol{x}) = \theta^T \boldsymbol{x}.$$

Here we have assumed initial θ and iterated it over to find the optimised paramaters θ_{opt}.

and H is the hessian matrix.It is square matrix of the second order partial derivatives of the cost function.

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

$$\mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

It can be obtained by second order derivative of J.

Then,we obtain the probabilities of the test set by substituting the estimated theta in the sigmoid function and classify them().

predicted_labels = probability>0.5

From the given train data we obtained the below result : The below θ is obtained after 100 iteration :

parameters : θ_{opt} =

```
[[ 4313.25679168]
 [ -148.99944752]
 [-1425.01564118]
 [ -947.20464262]
 [ 3439.09015461]]
```

Training time : CPU times: user 16 ms, sys: 0 ns, total: 16 ms Wall time: 16.4 ms

Accuracy-score: 0.8

(iv)**Logistic Regression(Library):**We have used the library function LogisticRegression() from sklearn.linear_model to estimate the parameters and predict the labels.

The results obtained are :

Training time : CPU times: user 4 ms, sys: 0 ns, total: 4 ms Wall time: 20.1 ms

Accuracy-score: 1.0

Question 3:

Preprocessing (Exploratory Data Analysis(EDA))

1)Read the data set.Then data.head() gives the first 5 data rows for observing the format of data.

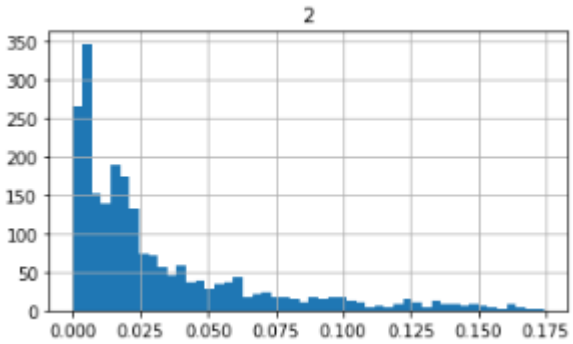
	Unnamed: 0	1	2	3	4	5	6	7	8	9
0	0	0.013409	0.014208	0.007272	0.022528	0.021640	0.003984	0.968127	0.000000	1
1	1	0.033941	0.032394	0.013957	0.009382	0.056784	0.000000	0.896414	0.023904	1
2	2	0.024826	0.016694	0.010657	0.006108	0.010441	0.000000	1.000000	0.000000	1
3	3	0.013235	0.008512	0.004536	0.009924	0.008956	0.000000	1.000000	0.007968	1
4	4	0.021368	0.015708	0.006477	0.009855	0.022992	0.000000	1.000000	0.000000	1

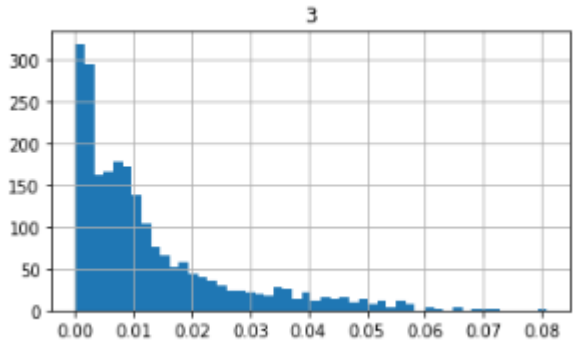
2)Then data.info() gives the number of non-null values for each attribute,the data type of attributes

3)data['9'].values_counts() gives the number of classes with their corresponding count of instances.

4)data.describe() summarize the central tendency, dispersion and shape of a dataset's distribution, excluding `NaN` values.

5)data.hist() displays histograms of all the attributes of data.





6)data.corr() displays the correlation matrix.It gives the correlation between features.

	1	2	3	4	5	6	7	8
1	1.000000	0.923179	0.936892	0.558479	0.560025	0.335002	-0.446284	0.284476
2	0.923179	1.000000	0.925403	0.553268	0.495780	0.309533	-0.503592	0.262611
3	0.936892	0.925403	1.000000	0.511949	0.506885	0.296677	-0.459527	0.275214
4	0.558479	0.553268	0.511949	1.000000	0.663467	0.447808	-0.338741	0.341691
5	0.560025	0.495780	0.506885	0.663467	1.000000	0.486263	-0.253466	0.445038
6	0.335002	0.309533	0.296677	0.447808	0.486263	1.000000	-0.075558	0.481211
7	-0.446284	-0.503592	-0.459527	-0.338741	-0.253466	-0.075558	1.000000	-0.028101
8	0.284476	0.262611	0.275214	0.341691	0.445038	0.481211	-0.028101	1.000000

7)scatter_matrix() plots the correlation between the attributes

8)Drop the first column i.e, Unnamed:0 since it just displays the serial number.

9)Split the new data into train and test sets in 80:20 ratio.

10)We separated train set and test set into respective feature set and label set.

11)Data Cleaning : We then fill the missing values and Nan in train and test data using imputer function with median as strategy.

12)From the above correlation matrix features 1,2 have correlation value of 0.922407 and features 1,3 have correlation value of 0.938654.So, features 1,2,3 are highly correlated.So,

we can drop features 2,3.But as our data set contains few features we are not dropping any of them because it doesnt affect much on the model performance.

13)Data Scaling and normalisation : Train and test data are scaled and normalised using scale_data(x) method to scale all the attributes to the same range.

We applied on data file D40.csv

(i)**Nearest Neighbour :**

Step1: Append train data and test data with their corresponding class labels.

Step2: Then, for every test point in test data we computed eucidean distance with every datapoint in train data. Then label of test point is assigned as label of least distance datapoint label.

Following are the observations:

accuracy score: `0.802660753880266`

precision score: `0.9017341040462428`

recall score: `0.8501362397820164`

auc score: 0.2773128324899442 (Area under curve for class 0)

0.7226871675100558 (Area under curve for class 1)

(ii)**Naive Bayes Classifier :** We use multivariate guassian to predict the probabilities P(x/Class = C_i).To predict the label of given test set,we calculate P(x/Class = C_i) for all the classes and assign the class with maximum probability.

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}$$

The results obtained are :

Training time: CPU times: user 4 ms, sys: 0 ns, total: 4 ms Wall time: 4.76 ms

Accuracy : 0.8270509977827051

Precision:0.9047619047619048

Recall: 0.8801089918256131

F_Measure:0.8922651933701659

Area under curve: 0.26232645646814584 (Area under curve for Class 0)

0.7376735435318541(Area under curve for Class 1)

(iii)**Logistic Regression(Gradient Descent):**The equation used in gradient descent is

$$\theta_{new} = \theta_{old} - \eta . \frac{\partial J}{\partial \theta} \Big|_{\theta = \theta old}$$

where J is the cost function for multivariate and θ is the parameters to be estimated.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{M}(H_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})^2$$

$$H_\theta(\boldsymbol{x}) = \theta^T \boldsymbol{x}.$$

Here we have assumed initial θ and the learning rate (η) and iterated it over to find the optimised paramaters θ_{opt}.

Then,we obtain the probabilities of the test set by substituting the estimated theta in the sigmoid function and classify them().

predicted_labels = probability>0.5

From the given train data we obtained the below result : The below θ is obtained after 100 iteration and the initial η = 0.1.

parameters : θ_{opt} =

```
[[ 1.69778614]
 [-0.17164114]
 [ 0.05799642]
 [-0.38128021]
 [-0.26646465]
 [-0.67045267]
 [-0.45979177]
 [-0.48785055]
 [-0.4159262 ]]
```

Training time：CPU times: user 84 ms, sys: 180 ms, total: 264 ms Wall time: 81.5 ms

Accuracy-score: 0.835920177383592

Precision: 0.8708860759493671

Recall: 0.9373297002724795

F_Measure: 0.9028871391076115

Area under curve:0.3349065784351888(Area under curve for Class 0)

0.6650934215648112(Area under curve for Class 1)

(iii)**Logistic Regression(Newton's Method):**The equation used in gradient descent is

$$\theta_{new} = \theta_{old} - H^{-1}. \left.\frac{\partial J}{\partial \theta}\right|_{\theta=\theta old}$$

where J is the cost function for multivariate and θ is the parameters to be estimated.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{M}(H_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})^2$$

$$H_\theta(\boldsymbol{x}) = \theta^T \boldsymbol{x}.$$

Here we have assumed initial θ and iterated it over to find the optimised paramaters θ_{opt}.

and H is the hessian matrix.It is square matrix of the second order partial derivatives of the cost function.

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

$$\mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

It can be obtained by second order derivative of J.

Then,we obtain the probabilities of the test set by substituting the estimated theta in the sigmoid function and classify them().

predicted_labels = probability>0.5

From the given train data we obtained the below result : The below θ is obtained after 100 iteration :

parameters : θ_{opt} =

```
[[ 5.44582326]
 [-4.47498883]
 [ 1.12554998]
 [ 3.78527106]
 [ 1.17027911]
 [-0.63480288]
 [ 2.44874708]
 [ 3.43631268]
 [ 1.77576075]]
```

Training time : CPU times: user 16 ms, sys: 20 ms, total: 36 ms Wall time: 16.6 ms

Accuracy-score: 0.811529933481153

Precision: 0.821917808219178

Recall: 0.9809264305177112

F_Measure:0.8944099378881987

Area under curve: 0.4738224990268587(Area under curve for Class 0)

0.5261775009731413(Area under curve for Class 1)

(iv)**Logistic Regression(Library):**We have used the library function LogisticRegression() from sklearn.linear_model to estimate the parameters and predict the labels.

The results obtained are :
parameters : coefficients =

```
[-4.26286808e+04, -4.24466049e+04,  1.18286158e+03,
-8.18829122e+00,1.16254615e+02, -3.84923721e+01,  4.63513818e+01,
4.05385728e+04])
```

Training time : CPU times: user 8 ms, sys: 0 ns, total: 8 ms Wall time: 5.75 ms

Accuracy-score: 0.8425720620842572

Precision: 0.8775510204081632

Recall: 0.9373297002724795

F_Measure:0.9064558629776021

Area under curve: 0.31704943557804594(Area under curve for Class 0)

0.6829505644219541(Area under curve for Class 0)