**Assessment Task 2: Individual Problem solving task**

**Part-1: Binary Classification**

For this problem, we will use a subset of the Wisconsin Breast Cancer dataset. Note that this dataset has some information missing.

* 1. **Data Munging:**
* Read the training and testing data. Print the number of features in the dataset.

train=pd.read\_csv("train\_wbcd.csv")

test=pd.read\_csv("test\_wbcd.csv")

print(train.shape)

print(test.shape)

**Output:**

(100, 32)

(20, 32)

* For the data label, print the total number of B's and M's in the training and testing data. Comment on the class distribution. Is it balanced or unbalanced?

train.Diagnosis.value\_counts()

Output:

B 58

M 42

Name: Diagnosis, dtype: int64

test.Diagnosis.value\_counts()

Output:

B 14

M 6

Name: Diagnosis, dtype: int64

Here Data is imbalanced but not highly. So we can take class\_weight="balanced" while applying logistic regression.

• **Print the number of features with missing entries**

print(train.columns[train.isnull().any()].tolist())

print(test.columns[test.isnull().any()].tolist())

Output:

['f21']

['f21']

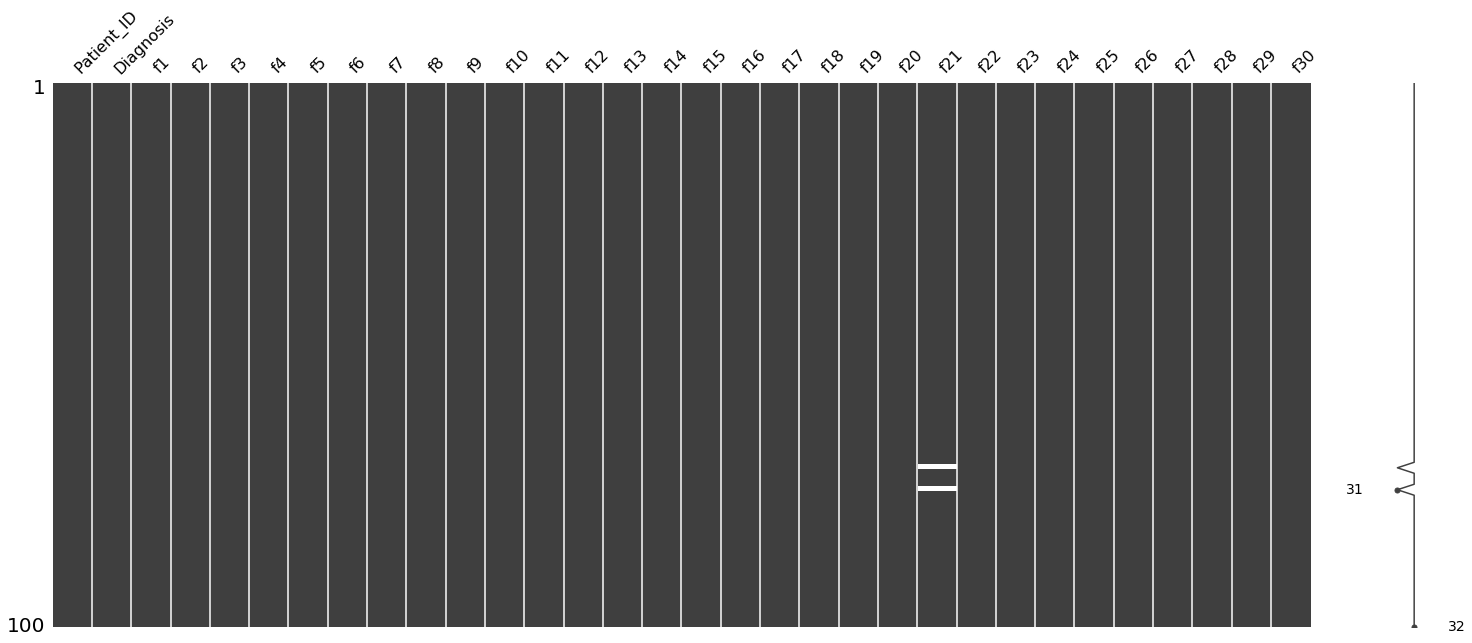
Missing Values in train dataset:



Missing values in test dataset:



**Just to visualize. No missing value:**

****

**For filling any feature, we can use r mean or median value of the feature values from observed entries. I have choose here median because I feel median imputation will work better because it is a number that is already present in the data set and is less susceptible to outlier errors as compared to mean imputation. Filled missing values with median because of good outliers.**

Code:

train.f21.fillna(train.f21.median(),inplace=True)

test.f21.fillna(test.f21.median(),inplace=True)

After missing values imputation by median data has cleaned.

• **Normalize the training and testing data**

from sklearn.preprocessing import StandardScaler

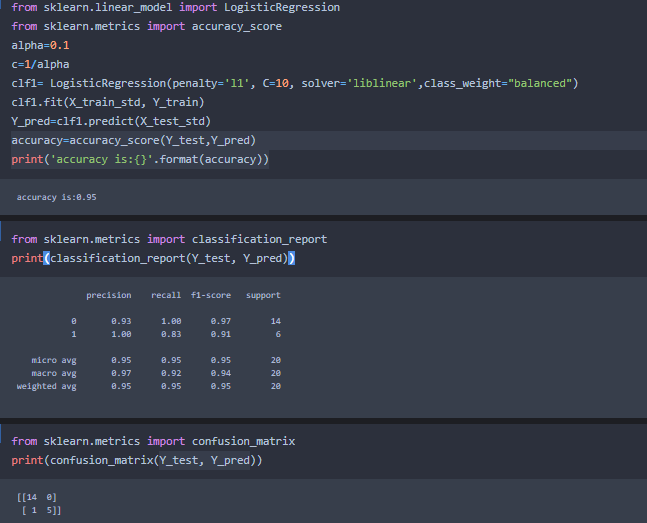
sc = StandardScaler()

X\_train\_std = sc.fit\_transform(X\_train)

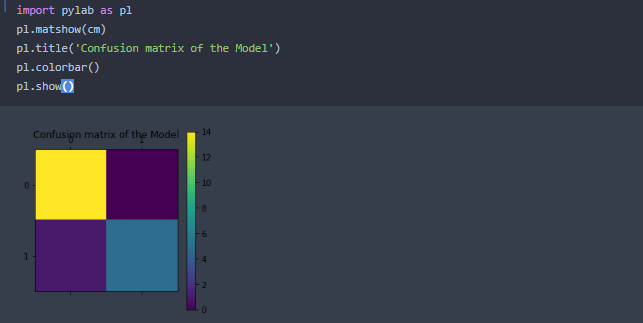
X\_test\_std = sc.transform(X\_test)

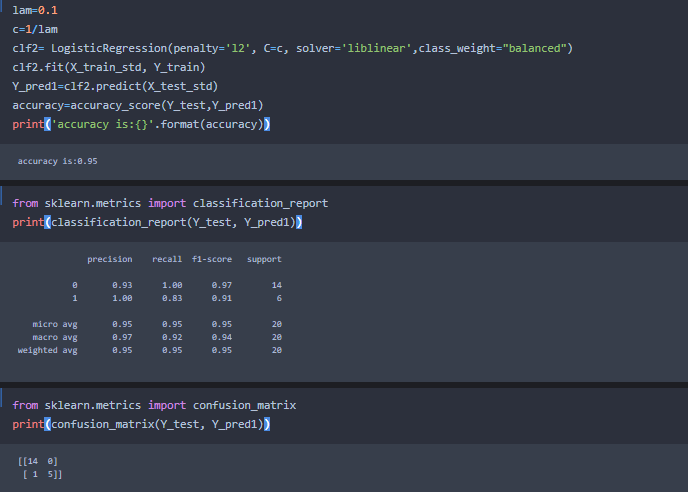
* 1. **Logistic Regression :**

Train logistic regression models with L1 regularization and L2 regularization using alpha = 0.1 and lambda = 0.1. Report accuracy, precision, recall, f1-score and print the confusion matrix.



**Confusion Matrix for model:**





* 1. **Choosing the best hyper-parameter**

1. **For L1 model, choose the best alpha value from the following set:** **{0.1,1,3,10,33,100,333,1000, 3333, 10000, 33333}** based on parameter P.

**Output:**

C: 10.0

Coefficient of each feature: [[0. 0.00969812 0.06688608 1.06381363 0. 0.

0.19197898 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 3.04502186 3.92248715 6.9011608 0.

2.42829482 0. 0. 0. 0.25285979 0.92329334]]

Training accuracy: 1.0

Test accuracy: 0.95

C: 1.0

Coefficient of each feature: [[0. 0. 0. 0. 0. 0.

0. 0.42560668 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 1.05479703 1.66323848 3.1060092 0.

0.82307393 0. 0.41592794 0.17235238 0.18235674 0.21261723]]

Training accuracy: 1.0

Test accuracy: 0.95

C: 0.3333333333333333

Coefficient of each feature: [[0. 0. 0. 0. 0. 0.

0. 0.31251968 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0.66371671 0.90777485 1.34590332 0.

0.24868754 0. 0.30274797 0.73635048 0.17645588 0. ]]

Training accuracy: 1.0

Test accuracy: 0.95

C: 0.1

Coefficient of each feature: [[0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0.4016646 0.28169495 0.31852308 0.

0. 0. 0.02443281 1.0535884 0. 0. ]]

Training accuracy: 0.97

Test accuracy: 0.95

C: 0.030303030303030304

Coefficient of each feature: [[0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 0.30353059 0. 0. ]]

Training accuracy: 0.93

Test accuracy: 0.95

C: 0.01

Coefficient of each feature: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.]]

Training accuracy: 0.58

Test accuracy: 0.7

C: 0.003003003003003003

Coefficient of each feature: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.]]

Training accuracy: 0.58

Test accuracy: 0.7

C: 0.001

Coefficient of each feature: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.]]

Training accuracy: 0.58

Test accuracy: 0.7

C: 0.00030003000300030005

Coefficient of each feature: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.]]

Training accuracy: 0.58

Test accuracy: 0.7

C: 0.0001

Coefficient of each feature: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.]]

Training accuracy: 0.58

Test accuracy: 0.7

C: 3.000030000300003e-05

Coefficient of each feature: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.]]

Training accuracy: 0.58

Test accuracy: 0.7

1. **For L2 model, choose the best lambda value from the following set: {0.001, 0.003, 0.01, 0.03, 0.1,0.3,1,3,10,33} based on parameter P.**

**Output:**

C: 1000.0

Coefficient of each feature is: [[ 2.30800719 2.43603174 2.28150424 2.30946564 -0.97998022 -0.43804591

1.94481012 2.19145209 -0.07893154 -2.48942637 1.14624381 0.56194399

1.16399016 1.29055453 1.61409959 -1.33316104 -0.96277373 -0.63697561

-1.90772279 0.5126348 2.49280621 4.00124548 2.49441845 2.36709597

2.85701292 0.66676139 1.13792138 1.46044474 2.28973769 2.68531559]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 333.3333333333333

Coefficient of each feature is: [[ 1.96772945 2.13623168 1.94679901 1.96234383 -0.8197337 -0.35639264

1.63702691 1.8761428 -0.03229645 -2.09840808 0.97972397 0.45561315

1.01704256 1.1073311 1.32825889 -1.13018393 -0.81289511 -0.49514134

-1.63735739 0.39945697 2.12784322 3.45907263 2.13097962 2.01594754

2.43267803 0.59624352 1.00828154 1.30060553 1.97894283 2.26719255]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 100.0

Coefficient of each feature is: [[ 1.64569431 1.85053918 1.62929788 1.63926744 -0.65277646 -0.2397139

1.39160314 1.55582136 -0.00425076 -1.72088392 0.83242249 0.3177467

0.84581578 0.94825939 1.0663428 -0.90986123 -0.59983073 -0.3596244

-1.3413562 0.35707319 1.77564464 2.87429838 1.76938987 1.68151829

1.97570298 0.50552261 0.88929898 1.09135917 1.62763774 1.88232178]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 33.333333333333336

Coefficient of each feature is: [[ 1.33588851 1.56390759 1.32359421 1.32289095 -0.48342484 -0.17473437

1.11078118 1.26699427 0.05868637 -1.35215819 0.69135332 0.23648718

0.7167646 0.7912307 0.78659874 -0.74313973 -0.4838862 -0.25024144

-1.08865915 0.22784736 1.44870701 2.37514499 1.44484382 1.366771

1.60157862 0.44225156 0.77912162 0.95455118 1.34349074 1.49987958]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 10.0

Coefficient of each feature is: [[ 1.0228753 1.27605721 1.01394622 1.00487885 -0.29394434 -0.09338654

0.84728958 0.96618586 0.12590074 -0.96802739 0.55350256 0.15272193

0.56777428 0.64026238 0.49639807 -0.57518011 -0.35393395 -0.15644913

-0.81739609 0.10240669 1.12056193 1.85247695 1.11465012 1.05178336

1.21125903 0.37525407 0.67928339 0.80040129 1.03831545 1.11577269]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 3.3333333333333335

Coefficient of each feature is: [[ 0.76895249 1.02292105 0.76297689 0.74464552 -0.13300593 -0.02539899

0.63460077 0.73076184 0.15223785 -0.6537823 0.43506208 0.10137161

0.44351898 0.50332696 0.25302689 -0.42463273 -0.25548386 -0.09410593

-0.58323151 -0.00426309 0.85299704 1.42487543 0.84929857 0.79205718

0.89478652 0.33540227 0.5916868 0.67224942 0.79276424 0.80474422]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 1.0

Coefficient of each feature is: [[ 0.53805401 0.76523978 0.53482054 0.51036871 0.01238161 0.04804396

0.45059218 0.52105796 0.14934614 -0.38224289 0.31998334 0.05932625

0.31474588 0.36774977 0.05006717 -0.27465802 -0.16101069 -0.04796616

-0.36345647 -0.08636901 0.60542384 1.01432733 0.60298482 0.55281164

0.60703393 0.29998128 0.49535618 0.53486997 0.56472908 0.52118502]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 0.3333333333333333

Coefficient of each feature is: [[ 0.3812236 0.5468125 0.37998259 0.35547582 0.08398914 0.09748305

0.32781087 0.37718752 0.12360469 -0.22421488 0.23056487 0.03115966

0.21544488 0.26514527 -0.04802497 -0.16067444 -0.09199337 -0.01620813

-0.22085586 -0.11487729 0.43084458 0.70021363 0.42895042 0.38778457

0.41547366 0.26401215 0.3995753 0.41513366 0.40419894 0.32823167]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 0.1

Coefficient of each feature is: [[ 0.26075797 0.33577999 0.26063623 0.24032043 0.09588997 0.11446669

0.22828436 0.26117399 0.08949784 -0.12710087 0.15510007 0.00873709

0.13904277 0.17929381 -0.07613135 -0.06860093 -0.03588469 0.01233301

-0.12775267 -0.1004014 0.29194494 0.42620356 0.2906206 0.26043249

0.27023102 0.21364194 0.29163719 0.29882805 0.27239469 0.18834212]]

Training accuracy is: 1.0

Test accuracy is: 0.95

C: 0.030303030303030304

Coefficient of each feature is: [[ 0.17321972 0.18042736 0.17350687 0.15980959 0.07467759 0.10042448

0.15311911 0.17379902 0.06115292 -0.06880019 0.1034912 -0.00600308

0.09225893 0.11994563 -0.06216168 -0.013638 0.00027252 0.03000166

-0.07440925 -0.06233554 0.19079685 0.2313775 0.18991466 0.17107906

0.16726245 0.15449357 0.19354664 0.2002098 0.1713238 0.10665241]]

Training accuracy is: 0.99

Test accuracy is: 0.95

1. **Use the best alpha and lambda parameter to re-train your final L1 and L2 regularized model. Evaluate the prediction performance on the test data and report the following:**

**• Precision and Accuracy**

### L1 regularization

**With L1 penalty**

best\_alpha=0.1

clf5= LogisticRegression(penalty='l1', C=best\_alpha, solver='liblinear',class\_weight="balanced")

clf5.fit(X\_train\_std, Y\_train)

Y\_pred2=clf5.predict(X\_test\_std)

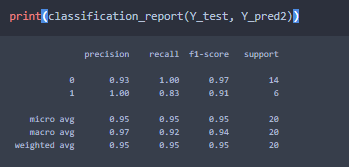
print(accuracy\_score(Y\_test,Y\_pred2))

Output :

0.95

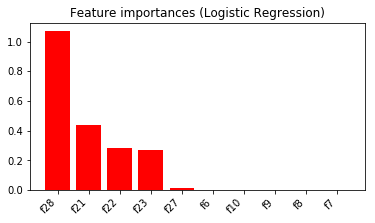
print(classification\_report(Y\_test, Y\_pred2))

**output:**

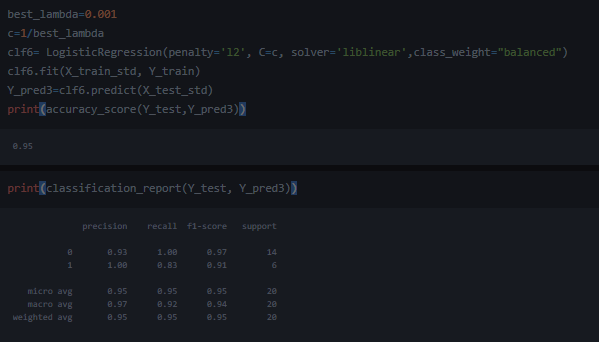


* **The top features selected in decreasing order of feature weights**

**L1 Feature importances (Logistic Regression):**

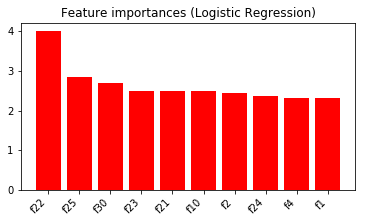
****

### L2 regularization

**L2 penalty** 

* The top features selected in decreasing order of feature weights

**L2 penalty:**



**• Confusion matrix**

**For L1 regularization:**

[[14 0]

[ 1 5]]

**For L2 Regularization:**

[[14 0]

[ 1 5]]

References

1. "P.C. Mahalanobis Memorial Lectures, 7th series". P.C. Mahalanobis Memorial Lectures, Indian Statistical Institute. Archived from the original on 29 October 2013. Retrieved 18 July 2017.Last Name, F. M. (Year). *Book Title.* City Name: Publisher Name.
2. Data Science Journal. (2012, April). Available Volumes. Retrieved from Japan Science and Technology Information Aggregator, Electronic: http://www.jstage.jst.go.jp/browse/dsj/\_vols [Archived](https://web.archive.org/web/20120403153707/http:/www.jstage.jst.go.jp/browse/dsj/_vols) 3 April 2012 at the Wayback Machine
3. R. Kohavi and F. Provost, "Glossary of terms," Machine Learning, vol. 30, no. 2–3, pp. 271–274, 1998
4. Le Roux, Nicolas; Bengio, Yoshua; Fitzgibbon, Andrew (2012). "Improving First and Second-Order Methods by Modeling Uncertainty". In Sra, Suvrit; Nowozin, Sebastian; Wright, Stephen J. (eds.). *Optimization for Machine Learning*. MIT Press. p. 404.
5. Jesse; Martino, Luca; Olmos, Pablo M.; Luengo, David (2015-06-01). "Scalable multi-output label prediction: From classifier chains to classifier trellises". *Pattern Recognition*. **48** (6): 2096–2109
6. Riemenschneider, M; Senge, R; Neumann, U; Hüllermeier, E; Heider, D (2016). ["Exploiting HIV-1 protease and reverse transcriptase cross-resistance information for improved drug resistance prediction by means of multi-label classification"](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4772363)