EN3150

PATTERN RECOGNITION

ASSIGNMENT 02

LEARNING FROM DATA AND RELATED CHALLENGES AND CLASSIFICATION

02.10.2024

AMARASURIYA G.N.

210036D

1) Logistic regression

1. Done in **Q1.ipynb**

2. The purpose of **y_encoded = le.fit_transform(df_filtered['species'])** is to convert the categorical variable species into a numerical format that can be used in the machine learning model.

The LabelEncoder from the sklearn.preprocessing module is used here to assign a unique integer to each category in the target variable species. This transformation is needed because most machine learning algorithms require numerical input rather than categorical data.

3. The purpose of **X = df_filtered.drop(['species', 'island', 'sex', 'class_encoded'], axis=1)**

This line splits the DataFrame into feature variables (X) by removing the columns that should not be used as predictors. It removes the non-numeric or irrelevant columns.

Thus, **X** will include only the numerical columns that are relevant for building the model.

4. Analyzing further can see why **island** and **sex** features cannot be used.

Based on model performance and the nature of the data we exclude the two above because:

1. **Categorical Nature**: Both are categorical variables. Most models require numerical input Sometimes the encoding process might not capture the relationships effectively mislead the model.
2. **Irrelevance to Target Variable:** For predicting penguin species the features island and sex may not provide significant predictive power. They can be excluded to simplify the model and reduce noise.
3. **Avoiding Overfitting**: Including too many features can lead to overfitting. It can negatively impact model performance on unseen data.

5. Trained the logistic regression model in **Q1.ipynb block 6,7 code**

6. The **random_state=42** is used in the **train_test_split** function for the following reasons

- **For reproducibility**: random_state is a seed for the random number generator used in the splitting process. It ensures that the data is split in the same way.
- **Consistency across runs**: Without setting a random_state, the split will differ every time run. It leads to slightly different training and test sets each time.]
- **42 is arbitrary**: It's a commonly used value. This reproducibility is essential for debugging and comparing models.

Different random_state values can be used where split is consistent with that seed, but the specific training and test data generated will differ. But excluding a random state is not ideal for reproducible research or testing due to variable results.

7. The accuracy of the LR model is low at approximately 58.14%. The model is under - performing, particularly when using the saga solver.

- **Inadequate training data**: There's only 214 rows of data in **X** and 171 rows of data used in **X_train** to train the model. The model is not trained on a sufficient amount of data to learn the underlying patterns effectively.
- **Inappropriate feature selection**: The 4 features used to train the model are not the most relevant to predict the target variable. It is better to perform feature engineering like creating new features from the existing ones. This can enhance the model's ability to capture the important patterns in the data.
- **Overfitting or underfitting**: The model may be overfitting due to overtraining noise. Also, it could underfit due to failing to capture the complexity of the underlying relationship. This requires adjustments in complexity, regularization, or hyperparameters to balance overfitting and underfitting.

The SAGA solver, a variant of SAG, supports non-smooth penalty L1 option, making it ideal for sparse multinomial logistic regression and large datasets [1]. It is not the most suitable choice for the current dataset.

- Solver misfit for small data
- Sensitivity to hyperparameters
- Gradient descent variability

Therefore, the saga solver performs poorly.

8. Trained the new logistic regression model in **Q1.ipynb block 9 code**

Classification accuracy of **1.0**

9. There is a dramatic increase in accuracy from approximately 58.14% with the saga solver to 100% with the liblinear solver. This highlights a significant impact of solver choice on model performance.

- **Data characteristics**: The "liblinear" solver is effective for smaller datasets and features with a coordinate descent algorithm, while "saga" is designed for larger datasets and high-dimensional data but may require more hyperparameter tuning for optimal performance [2].
- **Convergence behavior**: The "liblinear" solver converges faster in less complex scenarios and small datasets, while "saga" struggles with convergence if not tuned or the dataset doesn't leverage its strengths.

While both solvers have their strengths, "liblinear" often outperforms "saga" on smaller or less complex datasets. This is due to its faster convergence and lower computational requirements.

10. Done in **Q1.ipynb blocks 10-13 code**

Using sklearn standard scaler.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Obtained the following performance of the "liblinear" and "saga" solvers with and without feature scaling.

```
Accuracy with liblinear (without scaling): 1.0
Accuracy with liblinear (with scaling): 0.9767441860465116
Accuracy with saga (without scaling): 0.5813953488372093
Accuracy with saga (with scaling): 0.9767441860465116
```

Reasons for Differences in Accuracy

- **liblinear performs well without scaling**:
  liblinear uses a coordinate descent algorithm[1], which is less sensitive to feature scaling than gradient-based methods. It directly optimizes each coefficient, ensuring high accuracy even without scaling, as it is less affected by feature magnitudes. This approach improves numerical stability and overall fitting process.

- **Sensitivity of saga to Feature Scales**:
  The saga solver uses stochastic gradient descent [2] to update weights based on loss function gradients. Without scaling, larger features dominate gradient updates, leading to uneven learning and inefficient optimization. Scaling standardizes features to a similar range, ensuring equal contribution during training, improving convergence speed and accuracy, as observed with saga.

11. Done in **Q1.ipynb blocks 14-17 code**

When running listed code getting: **ValueError**: could not convert string to float: 'Dream'

This is because there's a string value 'Dream' present in one of the columns of the DataFrame **X**, which is being used as input for the logistic regression model. Logistic regression requires numerical data, so it cannot process string values directly.

To solve this can use one-hot encoding. It creates new binary columns for each unique value in a categorical column. Now the data is entirely numerical, and able to fit the logistic regression model without encountering the ValueError.

12. For these color categorical variables applying feature scaling methods like Standard Scaling or Min-Max Scaling directly to label-encoded categorical data is not correct.

Label encoding transforms values ('red', 'blue', 'green') into numerical labels. These numerical labels have no meaningful value/order. When scaling these encoded values, the algorithms treat the numbers as continuous.

It can lead to ordinal interpretation. This is where model assumes incorrectly about the relationships between categories, leading to misleading results.

Solution:

A better approach is to use One-Hot Encoding. It creates separate binary columns for each category. This can represent the categorical data without implying any order or distance between categories. After One-Hot Encoding, the resulting binary columns can be scaled if needed also.

Question 2

x_1 = number of hours studied

x_2 = undergraduate GPA

y = whether the student received an A+

$w_0 = -5.9, \quad w_1 = 0.06, \quad w_2 = 1.5$

1. Estimated probability of receiving an A^+
Studied for 50 hours and has an undergraduate GPA of 3.6

Using Logit transformation from lecture note;

$$\text{Logit}(p(x_i)) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$$

$$P(x_i) = \frac{e^{\text{Logit}(p(x_i))}}{1 + e^{\text{Logit}(p(x_i))}}$$

Thus

$$\text{Logit}(p(x_i)) = -5.9 + (0.06 \times 50) + (1.5 \times 3.6) = 2.5$$

Then

$$P(x_i) = \frac{e^{2.5}}{1 + e^{2.5}} = 0.92414$$

So, the estimated probability that a student who has studied for 50 hours and has a GPA of 3.6 will receive an A^+ is approximately 92.41%.

2. Hours of study needed
For a 60% chance of receiving an A^+

∴ Must find the number of study hours (x_1) given the probability p is given as 0.6

$$\text{Logit}(p(x_i)) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$$

$$P(x_i) = \frac{e^{\text{Logit}(p(x_i))}}{1 + e^{\text{Logit}(p(x_i))}}$$

Plugging the respective values

$$0.60 = \frac{e^{\text{Logit}(p(x_i))}}{1 + e^{\text{Logit}(p(x_i))}}$$

$$e^{\text{Logit}(p(x_i))} = \frac{0.6}{1 - 0.6}$$

$$\therefore \text{Logit}(p(x_i)) = \ln(1.5) \approx 0.4055$$

From Logit transformation in lecture note;

$$0.4055 = -5.9 + 0.06 \cdot x_1 + 1.5 \cdot 3.6$$

$$0.4055 = -0.5 + 0.06 \cdot x_1$$

$$\therefore x_1 = \frac{0.9055}{0.06} \approx 15.09\ hrs$$

2) Logistic regression on real world data

1. Choosing the **Breast Cancer Wisconsin** dataset where logistic regression is applicable. Dataset has no missing values.

2. Done in **Q2.ipynb blocks 3-4 code**

Obtained correlation matrix and pair plots for the features of the dataset as below

Will select the best 7 features from this correlation matrix. These features should be both highly correlated with the target variable. And have a low correlation with each other to avoid multicollinearity.

Then use pair plots to select up to 5 features for analysis.

	radius1	texture1	perimeter1	area1	smoothness1
radius1	1.000000	0.323782	0.997855	0.987357	0.170581
texture1	0.323782	1.000000	0.329533	0.321086	-0.023389
perimeter1	0.997855	0.329533	1.000000	0.986507	0.207278
area1	0.987357	0.321086	0.986507	1.000000	0.177028
smoothness1	0.170581	-0.023389	0.207278	0.177028	1.000000
compactness1	0.506124	0.236702	0.556936	0.498502	0.659123
concavity1	0.676764	0.302418	0.716136	0.685983	0.521984
concave_points1	0.822529	0.293464	0.850977	0.823269	0.553695
symmetry1	0.147741	0.071401	0.183027	0.151293	0.557775
fractal_dimension1	-0.311631	-0.076437	-0.261477	-0.283110	0.584792
radius2	0.679090	0.275869	0.691765	0.732562	0.301467
texture2	-0.097317	0.386358	-0.086761	-0.066280	0.068406
perimeter2	0.674172	0.281673	0.693135	0.726628	0.296092
area2	0.735864	0.259845	0.744983	0.800086	0.246552
smoothness2	-0.222600	0.006614	-0.202694	-0.166777	0.332375
compactness2	0.206000	0.191975	0.250744	0.212583	0.318943
concavity2	0.194204	0.143293	0.228082	0.207660	0.248396
concave_points2	0.376169	0.163851	0.407217	0.372320	0.380676
symmetry2	-0.104321	0.009127	-0.081629	-0.072497	0.200774
fractal_dimension2	-0.042641	0.054458	-0.005523	-0.019887	0.283607
radius3	0.969539	0.352573	0.969476	0.962746	0.213120
texture3	0.297008	0.912045	0.303038	0.287489	0.036072
perimeter3	0.965137	0.358040	0.970387	0.959120	0.238853
area3	0.941082	0.343546	0.941550	0.959213	0.206718
smoothness3	0.119616	0.077503	0.150549	0.123523	0.805324
compactness3	0.413463	0.277830	0.455774	0.390410	0.472468
concavity3	0.526911	0.301025	0.563879	0.512606	0.434926
concave_points3	0.744214	0.295316	0.771241	0.722017	0.503053
symmetry3	0.163953	0.105008	0.189115	0.143570	0.394309
fractal_dimension3	0.007066	0.119205	0.051019	0.003738	0.499316

	compactness1	concavity1	concave_points1	symmetry1
radius1	0.506124	0.676764	0.822529	0.147741
texture1	0.236702	0.302418	0.293464	0.071401
perimeter1	0.556936	0.716136	0.850977	0.183027
area1	0.498502	0.685983	0.823269	0.151293
smoothness1	0.659123	0.521984	0.553695	0.557775
compactness1	1.000000	0.883121	0.831135	0.602641
concavity1	0.883121	1.000000	0.921391	0.500667
concave_points1	0.831135	0.921391	1.000000	0.462497
symmetry1	0.602641	0.500667	0.462497	1.000000
fractal_dimension1	0.565369	0.336783	0.166917	0.479921
radius2	0.497473	0.631925	0.698050	0.303379
texture2	0.046205	0.076218	0.021480	0.128053
perimeter2	0.548905	0.660391	0.710650	0.313893
area2	0.455653	0.617427	0.690299	0.223970
smoothness2	0.135299	0.098564	0.027653	0.187321
compactness2	0.738722	0.670279	0.490424	0.421659
concavity2	0.570517	0.691270	0.439167	0.342627
concave_points2	0.642262	0.683260	0.615634	0.393298
symmetry2	0.229977	0.178009	0.095351	0.449137
fractal_dimension2	0.507318	0.449301	0.257584	0.331786
radius3	0.535315	0.688236	0.830318	0.185728
texture3	0.248133	0.299879	0.292752	0.090651
perimeter3	0.590210	0.729565	0.855923	0.219169
area3	0.509604	0.675987	0.809630	0.177193
smoothness3	0.565541	0.448822	0.452753	0.426675
compactness3	0.865809	0.754968	0.667454	0.473200
concavity3	0.816275	0.884103	0.752399	0.433721
concave_points3	0.815573	0.861323	0.910155	0.430297
symmetry3	0.510223	0.409464	0.375744	0.699826
fractal_dimension3	0.687382	0.514930	0.368661	0.438413

	fractal_dimension1	...	radius3	texture3	perimeter3
radius1	-0.311631	...	0.969539	0.297008	0.965137
texture1	-0.076437	...	0.352573	0.912045	0.358040
perimeter1	-0.261477	...	0.969476	0.303038	0.970387
area1	-0.283110	...	0.962746	0.287489	0.959120
smoothness1	0.584792	...	0.213120	0.036072	0.238853
compactness1	0.565369	...	0.535315	0.248133	0.590210
concavity1	0.336783	...	0.688236	0.299879	0.729565
concave_points1	0.166917	...	0.830318	0.292752	0.855923
symmetry1	0.479921	...	0.185728	0.090651	0.219169
fractal_dimension1	1.000000	...	-0.253691	-0.051269	-0.205151
radius2	0.000111	...	0.715065	0.194799	0.719684
texture2	0.164174	...	-0.111690	0.409003	-0.102242
perimeter2	0.039830	...	0.697201	0.200371	0.721031
area2	-0.090170	...	0.757373	0.196497	0.761213
smoothness2	0.401964	...	-0.230691	-0.074743	-0.217304
compactness2	0.559837	...	0.204607	0.143003	0.260516
concavity2	0.446630	...	0.186904	0.100241	0.226680
concave_points2	0.341198	...	0.358127	0.086741	0.394999
symmetry2	0.345007	...	-0.128121	-0.077473	-0.103753
fractal_dimension2	0.688132	...	-0.037488	-0.003195	-0.001000
radius3	-0.253691	...	1.000000	0.359921	0.993708
texture3	-0.051269	...	0.359921	1.000000	0.365098
perimeter3	-0.205151	...	0.993708	0.365098	1.000000
area3	-0.231854	...	0.984015	0.345842	0.977578
smoothness3	0.504942	...	0.216574	0.225429	0.236775
compactness3	0.458798	...	0.475820	0.360832	0.529408
concavity3	0.346234	...	0.573975	0.368366	0.618344
concave_points3	0.175325	...	0.787424	0.359755	0.816322
symmetry3	0.334019	...	0.243529	0.233027	0.269493
fractal_dimension3	0.767297	...	0.093492	0.219122	0.138957

	area3	smoothness3	compactness3	concavity3
radius1	0.941082	0.119616	0.413463	0.526911
texture1	0.343546	0.077503	0.277830	0.301025
perimeter1	0.941550	0.150549	0.455774	0.563879
area1	0.959213	0.123523	0.390410	0.512606
smoothness1	0.206718	0.805324	0.472468	0.434926
compactness1	0.509604	0.565541	0.865809	0.816275
concavity1	0.675987	0.448822	0.754968	0.884103
concave_points1	0.809630	0.452753	0.667454	0.752399
symmetry1	0.177193	0.426675	0.473200	0.433721
fractal_dimension1	-0.231854	0.504942	0.458798	0.346234
radius2	0.751548	0.141919	0.287103	0.380585
texture2	-0.083195	-0.073658	-0.092439	-0.068956
perimeter2	0.730713	0.130054	0.341919	0.418899
area2	0.811408	0.125389	0.283257	0.385100
smoothness2	-0.182195	0.314457	-0.055558	-0.058298
compactness2	0.199371	0.227394	0.678780	0.639147
concavity2	0.188353	0.168481	0.484858	0.662564
concave_points2	0.342271	0.215351	0.452888	0.549592
symmetry2	-0.110343	-0.012662	0.060255	0.037119
fractal_dimension2	-0.022736	0.170568	0.390159	0.379975
radius3	0.984015	0.216574	0.475820	0.573975
texture3	0.345842	0.225429	0.360832	0.368366
perimeter3	0.977578	0.236775	0.529408	0.618344
area3	1.000000	0.209145	0.438296	0.543331
smoothness3	0.209145	1.000000	0.568187	0.518523
compactness3	0.438296	0.568187	1.000000	0.892261
concavity3	0.543331	0.518523	0.892261	1.000000
concave_points3	0.747419	0.547691	0.801080	0.855434
symmetry3	0.209146	0.493838	0.614441	0.532520
fractal_dimension3	0.079647	0.617624	0.810455	0.686511

	concave_points3	symmetry3	fractal_dimension3
radius1	0.744214	0.163953	0.007066
texture1	0.295316	0.105008	0.119205
perimeter1	0.771241	0.189115	0.051019
area1	0.722017	0.143570	0.003738
smoothness1	0.503053	0.394309	0.499316
compactness1	0.815573	0.510223	0.687382
concavity1	0.861323	0.409464	0.514930
concave_points1	0.910155	0.375744	0.368661
symmetry1	0.430297	0.699826	0.438413
fractal_dimension1	0.175325	0.334019	0.767297
radius2	0.531062	0.094543	0.049559
texture2	-0.119638	-0.128215	-0.045655
perimeter2	0.554897	0.109930	0.085433
area2	0.538166	0.074126	0.017539
smoothness2	-0.102007	-0.107342	0.101480
compactness2	0.483208	0.277878	0.590973
concavity2	0.440472	0.197788	0.439329
concave_points2	0.602450	0.143116	0.310655
symmetry2	-0.030413	0.389402	0.078079
fractal_dimension2	0.215204	0.111094	0.591328
radius3	0.787424	0.243529	0.093492
texture3	0.359755	0.233027	0.219122
perimeter3	0.816322	0.269493	0.138957
area3	0.747419	0.209146	0.079647
smoothness3	0.547691	0.493838	0.617624
compactness3	0.801080	0.614441	0.810455
concavity3	0.855434	0.532520	0.686511
concave_points3	1.000000	0.502528	0.511114
symmetry3	0.502528	1.000000	0.537848
fractal_dimension3	0.511114	0.537848	1.000000

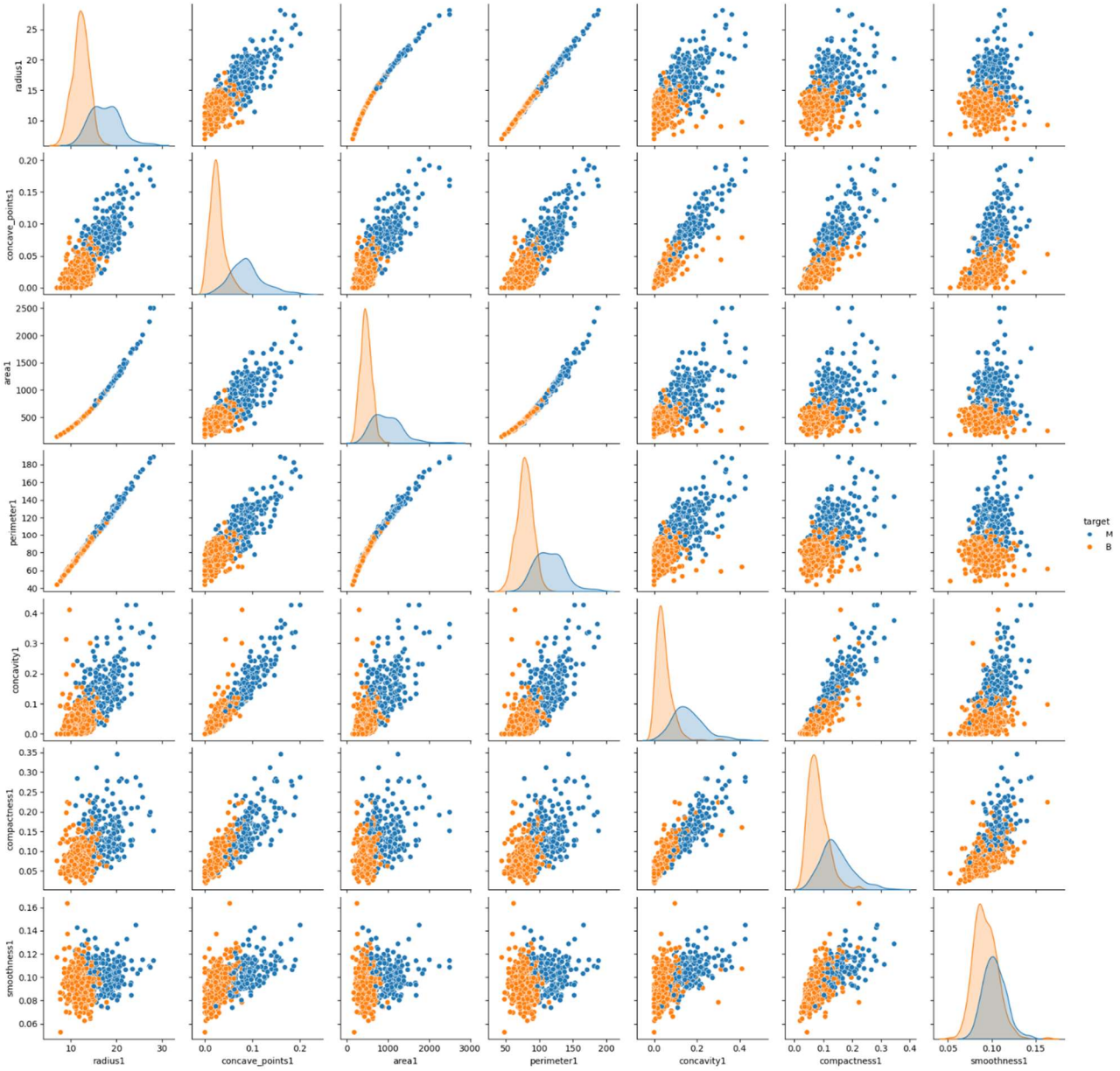
[30 rows x 30 columns]

From these results we can comment

radius1, concave_points1, area1, perimeter1, concavity1, compactness1 and smoothness1

These features are both strong indicators of important characteristics and provide a mix of size, shape, and texture properties, offering a comprehensive feature set.

Applying pair plots on these selected 7 obtained the following results.



- The features Radius1, Area1, and Perimeter1 show strong positive linear relationships.
- The pair plot also shows that Concave_points1 and Concavity1 have a moderate positive correlation.
- Smoothness1 shows some separation between classes, does not have a strong relationship with the other features

Features used: Radius1, Area1, Perimeter1, Concave_points1, Concavity1

These features capture a combination of size and shape making them powerful for discriminating between malignant and benign tumors.

3. Done in Q2.ipynb blocks 6-7 code

Fitted a logistic regression model and used accuracy_score and classification_report from sklearn.metrics to evaluate the model's performance

Obtained following results;

```
Accuracy: 0.9122807017543859
Classification Report:
              precision    recall  f1-score   support

           B       0.90      0.97      0.93       108
           M       0.94      0.81      0.87        63

    accuracy                           0.91       171
   macro avg       0.92      0.89      0.90       171
weighted avg       0.91      0.91      0.91       171
```

The logistic regression model achieved an accuracy of 91.23%.

For benign cases (B), it had a precision of 0.90 and a recall of 0.97, while for malignant cases (M), the precision was 0.94 and the recall was 0.81.

The F1-scores, balancing precision and recall, were 0.93 for benign and 0.87 for malignant, indicating the model's strong performance in classifying both categories effectively.

4. Done in Q2.ipynb blocks 8 code

Used statsmodels.Logit to obtain and interpret the p-values for the predictors.

Had to convert **y** which contains the column Diagnosis which has the value M and B as target variables. Converted the numerical values.

```
Optimization terminated successfully.
         Current function value: 0.186737
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:              Diagnosis   No. Observations:                  455
Model:                          Logit   Df Residuals:                      449
Method:                           MLE   Df Model:                            5
Date:                Mon, 30 Sep 2024   Pseudo R-squ.:                  0.7169
Time:                        16:57:22   Log-Likelihood:                -84.965
converged:                       True   LL-Null:                       -300.17
Covariance Type:            nonrobust   LLR p-value:                 8.261e-91
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.6693      7.930      0.337      0.736     -12.873      18.211
radius1       -0.4756      2.364     -0.201      0.841      -5.108       4.157
concave_points1  99.1998     20.109      4.933      0.000      59.786     138.613
area1          0.0258      0.014      1.907      0.057      -0.001       0.052
perimeter1    -0.1880      0.278     -0.675      0.500      -0.734       0.358
concavity1    -1.4491      6.477     -0.224      0.823     -14.143      11.245
==============================================================================
```

To determine if any features can be discarded using $p > 0.05$

- The constant has a high p-value of 0.736, so it is not statistically significant. The intercept alone does not provide useful information.
- Similarly, the feature radius1 has a p-value of 0.841, suggesting it is not significantly related to the target variable. It may not be necessary in the model.
- The most important feature is concave_points1, with a p-value of 0.000, making it highly significant.
- area1 with a p-value of 0.057, is only marginally significant, meaning it may contribute to the model.
- perimeter1 and concavity1 are not significant, they do not strongly influence the diagnosis prediction.

∴ **Retainable**: concave_points1, area1
Discardable: radius1, perimeter1, and concavity1

3) Logistic regression First/Second-Order Methods

1. Done in **Q3.ipynb block 1 code**

2. Done in **Q3.ipynb block 3 code**

Implemented batch gradient descent to update the weights for the given dataset over 20 iterations.

Method used to initialize the weights: Random initialization with small values scaled by 0.01

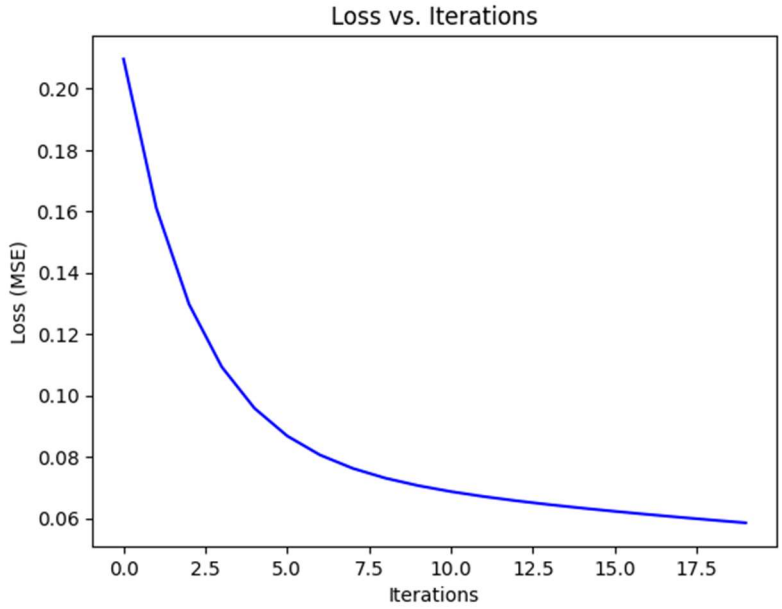
```
#Initialize weights and bias to small random values
weights = np.random.randn(n_features) * 0.01
bias = np.random.randn() * 0.01
```

Reason: Weights and bias to zero are initialized to to small random values. This is to prevent every feature weight in the model learning the same during training.

3. Loss Function: Mean Squared Error (MSE)

Reason MSE is an interpretable and a smooth gradient descent function. It is sensitive to outliers, allowing for better convergent gradient descent. Also it penalizes large errors more heavily, making it suitable for regression tasks.

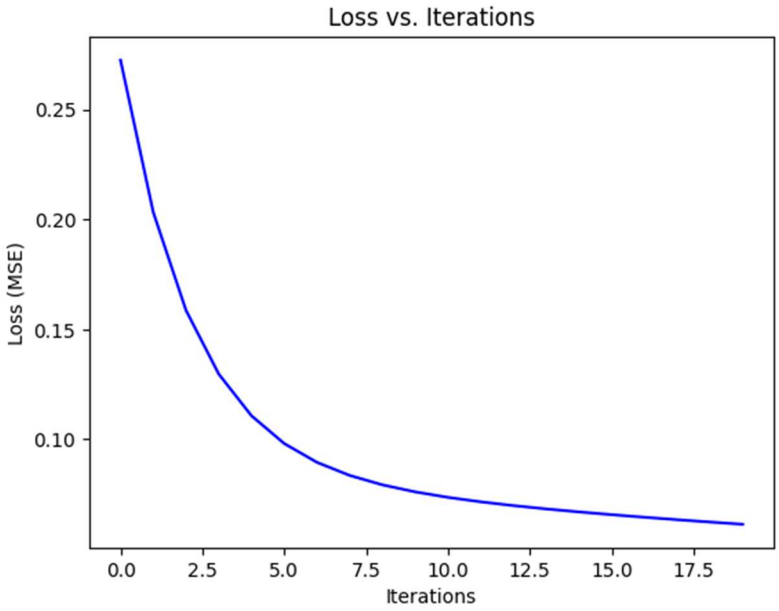
4. Plotted the loss vs number of iterations.




```
Weights are: [0.02193013 0.12359208]
Bias is: 0.08367596678437207
Minimum Loss: 0.058488730023383236
```

5. Done in **Q3.ipynb block 5 code**

Repeated with for Stochastic Gradient descent.

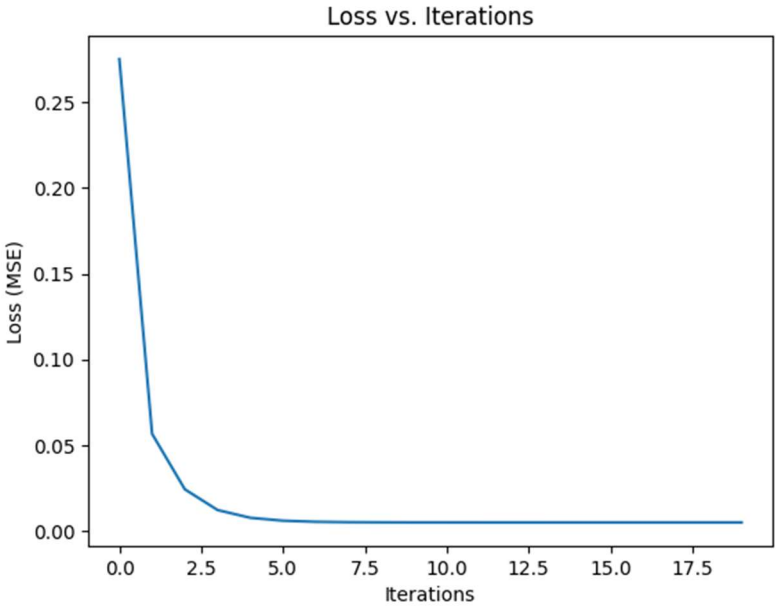



```
Weights are: [0.02794303 0.12104804]
Bias is: 0.07851895000409807
Minimum Loss: 0.060530166024089206
```

6. Done in **Q3.ipynb block 6 code**

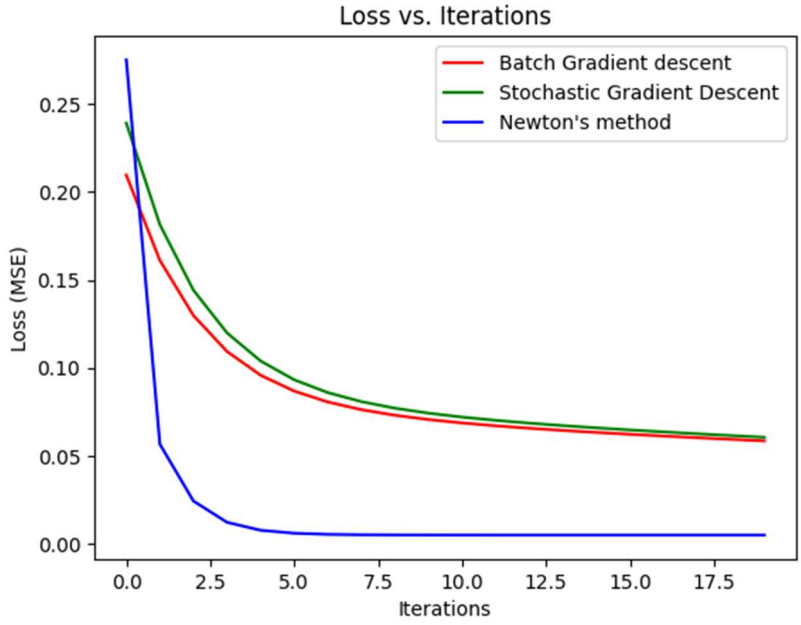
Implemented Newton's method to update the weights for the given dataset over 20 iterations.

7. Plotted the loss vs number of iterations.




```
Weights are: [0.13467162 0.0535865 ]
Bias is: 0.4900207455595825
Minimum Loss: 0.004946596991457199
```

8. Done in **Q3.ipynb block 7 code**.



Newton's Method shows fast convergence but takes more time to compute on Colab. **Batch Gradient Descent** is slower but stable convergence. **Stochastic Gradient Descent** has noisy convergence patterns

9. Approaches to decide number of iterations for Gradient descent and Newton's method.

1. **Convergence Criteria**

   To determine the number of iterations in Gradient Descent and Newton's Method, can select a convergence criterion based on loss function. Then can terminate the iteration loop when absolute difference falls below predefined or gradients are close to zero, This reduces computations and ensuring convergence.

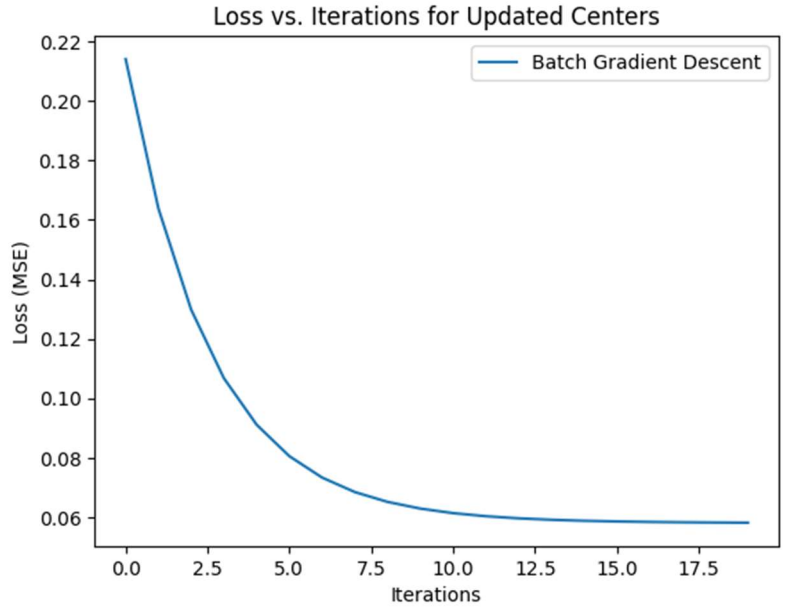
2. **Cross-Validation**

   Cross-validation techniques divide a dataset into training and validation sets. Then can run the algorithm for various iterations. The performance on the validation set is assessed using MSE. Therefore, can balance model performance and generalization.

10. Done in **Q2.ipynb blocks 8-10 code**

Changed to centers = [[3, 0], [5, 1.5]]

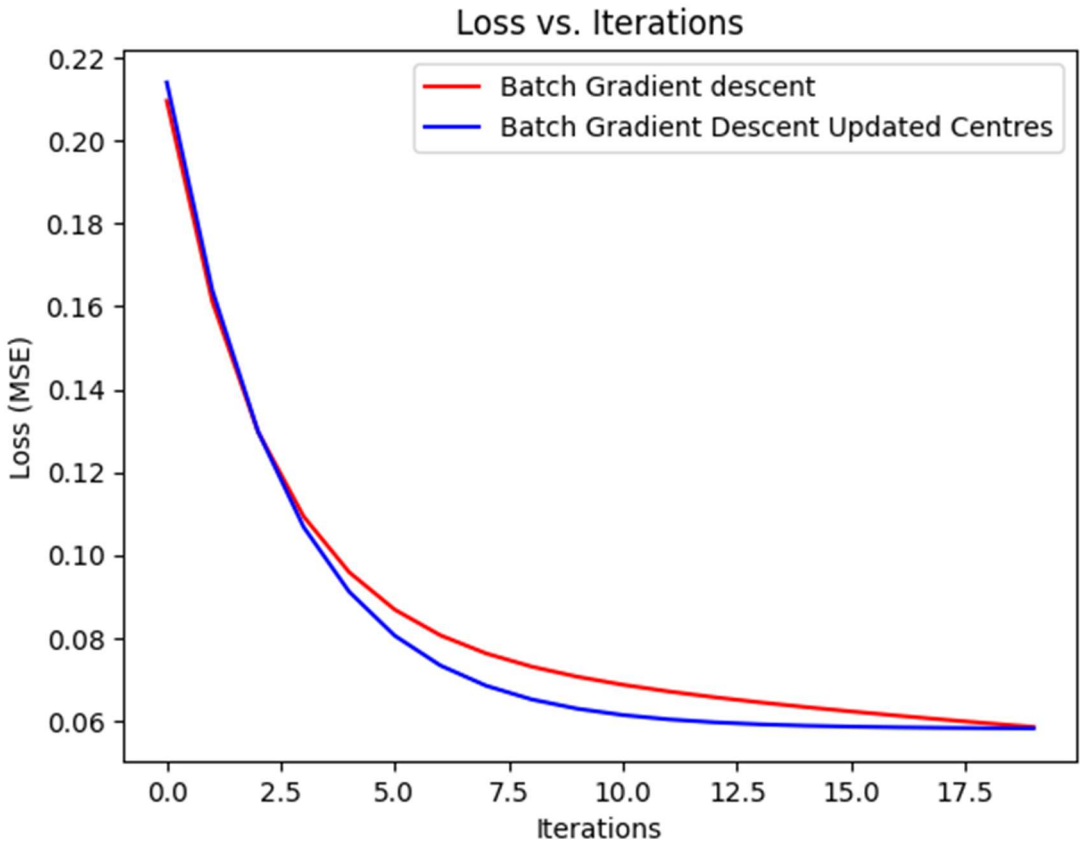
Used batch gradient descent to update the weights for this new configuration.




```
Weights are: [0.04908551 0.13527437]
Bias is: 0.017746202547842305
Minimum Loss: 0.05813945321707209
```

The updated centers have changed the distribution of the data points. Since the new centers are closer together, the model converges faster. This is because the gradient updates are more significant in the initial iterations.

Explanation for convergence behavior:



Data distribution influence - The original centers have a more favorable geometry for optimization. The updated centers could have resulted in a more challenging landscape for the Batch Gradient Descent algorithm to navigate.

Model's sensitivity - The model is more sensitive to the arrangement of data points, as indicated by the fluctuating loss in the early iterations with the updated centers**.**

References

[1] A. R, “Scikit-learn solvers explained - Arnav R - Medium,” Medium, Mar. 31, 2022. [Online]. Available: https://medium.com/@arnavr/scikit-learn-solvers-explained-780a17bc322d

[2] “Logistic regression python solvers’ definitions,” Stack Overflow. https://stackoverflow.com/questions/38640109/logistic-regression-python-solvers-definitions

THE END