Computational Task 1

ROSHAN RAJ RAMESH
209030148
rrr13@student.le.ac.uk

Question 1

The author is trying to solve a classification problem of breast cancer malignant or benign.

The author used inductive machine learning and logistic regression for classifying the cancer type. Inductive machine learning reported a 96.2% accuracy rate and by logistic regression, the author got a 97.5% accuracy rate.

To test the accuracy of the classifier, the author used 10-fold cross-validation repeated 100 times for the logistic regression algorithm and 10-fold cross-validation was done 5 times and the results were reported 3 times out of 5 times for the inductive machine learning classification.

Question 2

Ten real-valued features are computed for each cell nucleus. The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. So, in total there are 30 attributes.

Number of malignant and benign cases respectively

```
Count_M = 0
Count_B = 0
UnIndentified = 0
for i in answer:
    if(i=='M'):
        Count_M = Count_M+1
    elif(i=='B'):
        Count_B = Count_B+1
    else:
        UnIndentified = UnIndentified+1
print("Number of malignant cases: ", Count_M)
print("Number of benign cases: ", Count_B)
print("UnIdentified: ", UnIndentified)
```

```
Number of malignant cases:  212
Number of benign cases:  357
UnIdentified:  0
```

Number of malignant cases: 212

Number of Benign cases: 357

Following is the mean, variance and standard deviation of each attribute (Table 1) and mean, variance and standard deviation of each other in (Table 2), (Table 3), (Table 4) respectively.

	Mean	Variance	Standard Deviation
Attribute 1	14.127292	12.418920	3.524049
Attribute 2	19.289649	18.498909	4.301036
Attribute 3	91.969033	590.440480	24.298981
Attribute 4	654.889104	123843.554318	351.914129
Attribute 5	0.096360	0.000198	0.014064
Attribute 6	0.104341	0.002789	0.052813
Attribute 7	0.088799	0.006355	0.079720
Attribute 8	0.048919	0.001506	0.038803
Attribute 9	0.181162	0.000752	0.027414
Attribute 10	0.062798	0.000050	0.007060
Attribute 11	0.405172	0.076902	0.277313
Attribute 12	1.216853	0.304316	0.551648
Attribute 13	2.866059	4.087896	2.021855
Attribute 14	40.337079	2069.431583	45.491006
Attribute 15	0.007041	0.000009	0.003003
Attribute 16	0.025478	0.000321	0.017908
Attribute 17	0.031894	0.000911	0.030186
Attribute 18	0.011796	0.000038	0.006170
Attribute 19	0.020542	0.000068	0.008266
Attribute 20	0.003795	0.000007	0.002646
Attribute 21	16.269190	23.360224	4.833242
Attribute 22	25.677223	37.776483	6.146258
Attribute 23	107.261213	1129.130847	33.602542
Attribute 24	880.583128	324167.385102	569.356993
Attribute 25	0.132369	0.000521	0.022832
Attribute 26	0.254265	0.024755	0.157336
Attribute 27	0.272188	0.043524	0.208624
Attribute 28	0.114606	0.004321	0.065732
Attribute 29	0.290076	0.003828	0.061867
Attribute 30	0.083946	0.000326	0.018061

(Table 1)

	Mean of malignant	Variance of malignant \
Attribute 1	17.462830	10.265431
Attribute 2	21.604906	14.284393
Attribute 3	115.365377	477.625870
Attribute 4	978.376415	135378.355365
Attribute 5	0.102898	0.000159
Attribute 6	0.145188	0.002915
Attribute 7	0.160775	0.005628
Attribute 8	0.087990	0.001182
Attribute 9	0.192909	0.000764
Attribute 10	0.062680	0.000057
Attribute 11	0.609083	0.119052
Attribute 12	1.210915	0.233461
Attribute 13	4.323929	6.597427
Attribute 14	72.672406	3764.468961
Attribute 15	0.006780	0.000008
Attribute 16	0.032281	0.000338
Attribute 17	0.041824	0.000467
Attribute 18	0.015060	0.000030
Attribute 19	0.020472	0.000101
Attribute 20	0.004062	0.000004
Attribute 21	21.134811	18.348967
Attribute 22	29.318208	29.537095
Attribute 23	141.370330	867.718099
Attribute 24	1422.286321	357565.421850
Attribute 25	0.144845	0.000478
Attribute 26	0.374824	0.029027
Attribute 27	0.450606	0.032945
Attribute 28	0.182237	0.002144
Attribute 29	0.323468	0.005578
Attribute 30	0.091530	0.000465

(Table 2)

	Standard Deviation of malignant	Mean of benign \
Attribute 1	3.203971	12.146524
Attribute 2	3.779470	17.914762
Attribute 3	21.854653	78.075406
Attribute 4	367.937978	462.790196
Attribute 5	0.012608	0.092478
Attribute 6	0.053987	0.080085
Attribute 7	0.075019	0.046058
Attribute 8	0.034374	0.025717
Attribute 9	0.027638	0.174186
Attribute 10	0.007573	0.062867
Attribute 11	0.345039	0.284082
Attribute 12	0.483178	1.220380
Attribute 13	2.568546	2.000321
Attribute 14	61.355268	21.135148
Attribute 15	0.002890	0.007196
Attribute 16	0.018387	0.021438
Attribute 17	0.021603	0.025997
Attribute 18	0.005517	0.009858
Attribute 19	0.010065	0.020584
Attribute 20	0.002041	0.003636
Attribute 21	4.283569	13.379801
Attribute 22	5.434804	23.515070
Attribute 23	29.457055	87.005938
Attribute 24	597.967743	558.899440
Attribute 25	0.021870	0.124959
Attribute 26	0.170372	0.182673
Attribute 27	0.181507	0.166238
Attribute 28	0.046308	0.074444
Attribute 29	0.074685	0.270246
Attribute 30	0.021553	0.079442

(Table 3)

	Variance of benign	Standard Deviation of benign
Attribute 1	3.170222	1.780512
Attribute 2	15.961021	3.995125
Attribute 3	139.415582	11.807438
Attribute 4	18033.030100	134.287118
Attribute 5	0.000181	0.013446
Attribute 6	0.001139	0.033750
Attribute 7	0.001887	0.043442
Attribute 8	0.000253	0.015909
Attribute 9	0.000615	0.024807
Attribute 10	0.000046	0.006747
Attribute 11	0.012672	0.112570
Attribute 12	0.347133	0.589180
Attribute 13	0.594702	0.771169
Attribute 14	78.206998	8.843472
Attribute 15	0.000009	0.003061
Attribute 16	0.000267	0.016352
Attribute 17	0.001084	0.032918
Attribute 18	0.000033	0.005709
Attribute 19	0.000049	0.006999
Attribute 20	0.000009	0.002938
Attribute 21	3.925817	1.981368
Attribute 22	30.183536	5.493955
Attribute 23	182.982188	13.527091
Attribute 24	26765.425899	163.601424
Attribute 25	0.000401	0.020013
Attribute 26	0.008497	0.092180
Attribute 27	0.019703	0.140368
Attribute 28	0.001281	0.035797
Attribute 29	0.001743	0.041745
Attribute 30	0.000191	0.013804

(Table 4)

Attributes are not normalized to its unit variance as the mean is not equal to zero and standard deviation is not equal to one as pictured in the (Table 5)

	1	2	3	4	5 \
count	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360
std	3.524049	4.301036	24.298981	351.914129	0.014064
min	6.981000	9.710000	43.790000	143.500000	0.052630
25%	11.700000	16.170000	75.170000	420.300000	0.086370
50%	13.370000	18.840000	86.240000	551.100000	0.095870
75%	15.780000	21.800000	104.100000	782.700000	0.105300
max	28.110000	39.280000	188.500000	2501.000000	0.163400

(Table 5)

SkLearn's preprocessing library (StandardScaler method) is used to normalise the attributes to its unit variance. So after normalizing the attributes we get mean = 0 and standard deviation = 1 as shown in table 6.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
data_norm = sc.fit_transform(data_norm)
```

	radius	texture	perimeter	area	smoothness \
count	5.690000e+02	5.690000e+02	5.690000e+02	5.690000e+02	5.690000e+02
mean	-1.170710e-18	6.712069e-17	6.634022e-18	-4.292602e-18	-1.482899e-17
std	1.000880e+00	1.000880e+00	1.000880e+00	1.000880e+00	1.000880e+00
min	-2.029648e+00	-2.229249e+00	-1.984504e+00	-1.454443e+00	-3.112085e+00
25%	-6.893853e-01	-7.259631e-01	-6.919555e-01	-6.671955e-01	-7.109628e-01
50%	-2.150816e-01	-1.046362e-01	-2.359800e-01	-2.951869e-01	-3.489108e-02
75%	4.693926e-01	5.841756e-01	4.996769e-01	3.635073e-01	6.361990e-01
max	3.971288e+00	4.651889e+00	3.976130e+00	5.250529e+00	4.770911e+00

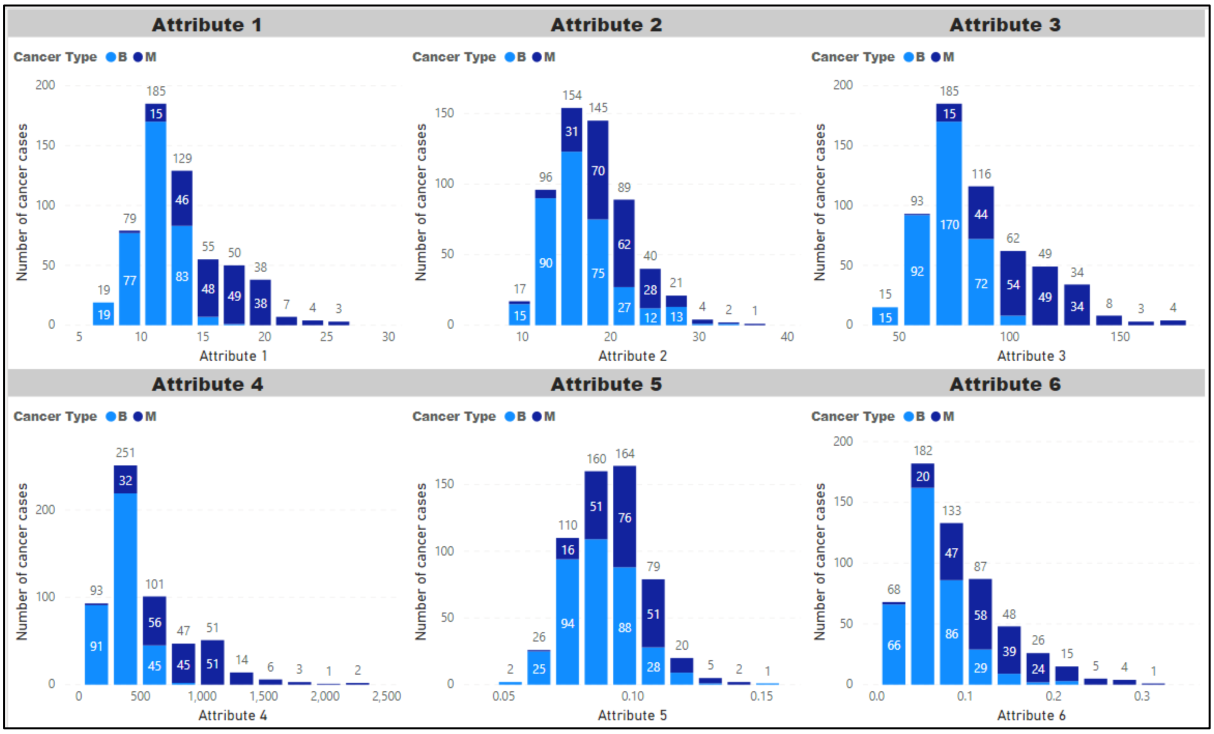
(Table 6)

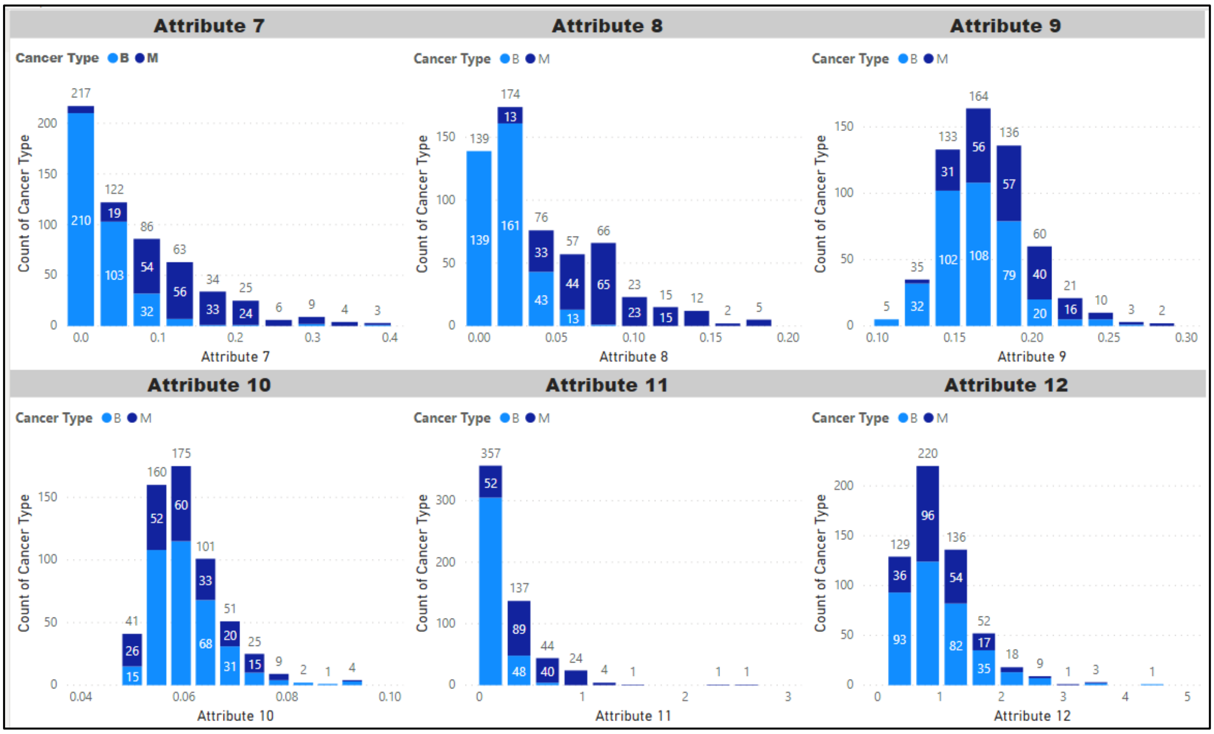
Question 3

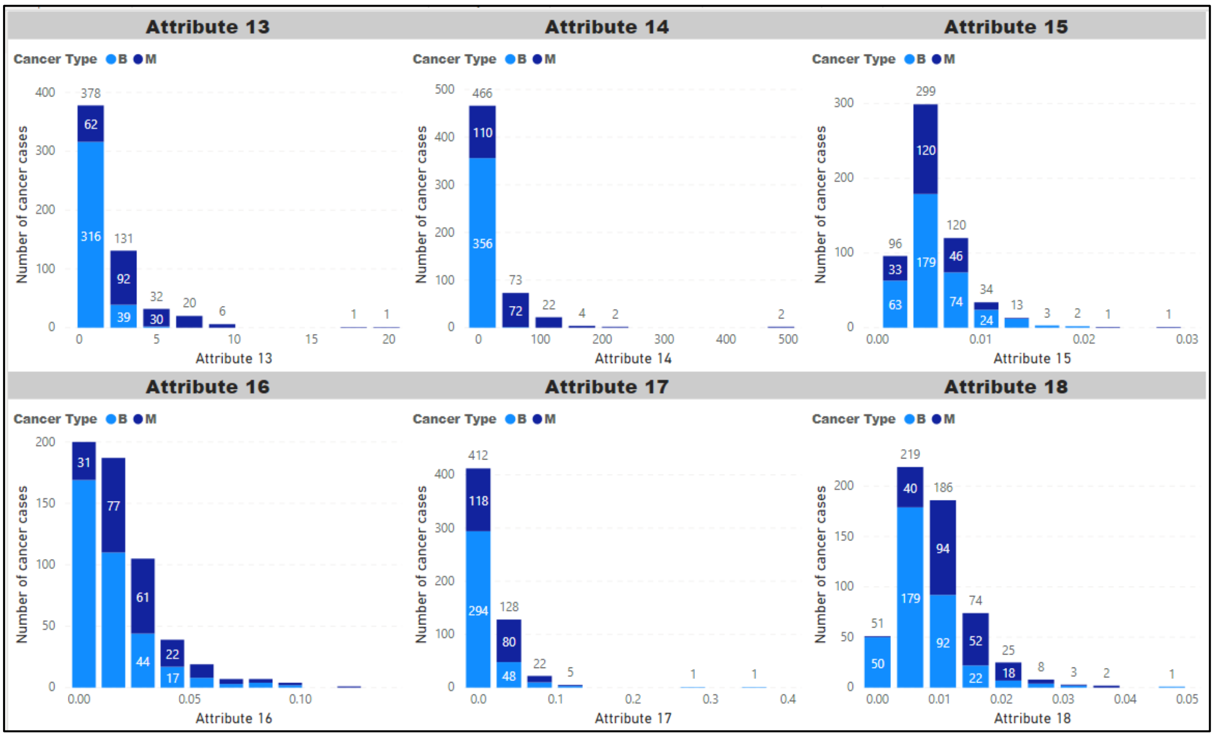
$$Error1 = \frac{Number\ wrongly\ classified\ Benign\ in\ Malignant}{Total\ cases\ of\ the\ Malignant}$$

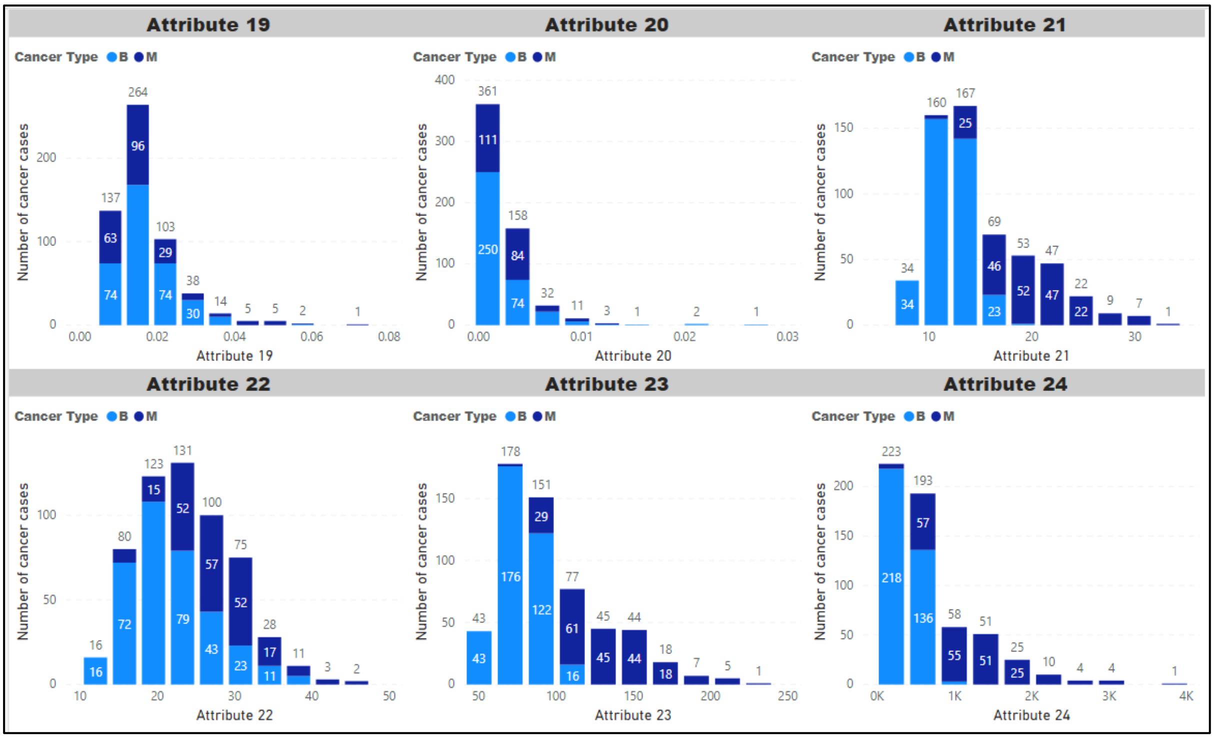
$$Error2 = \frac{Number\ wrongly\ classified\ in\ Benign}{Total\ cases\ of\ the\ Benign}$$

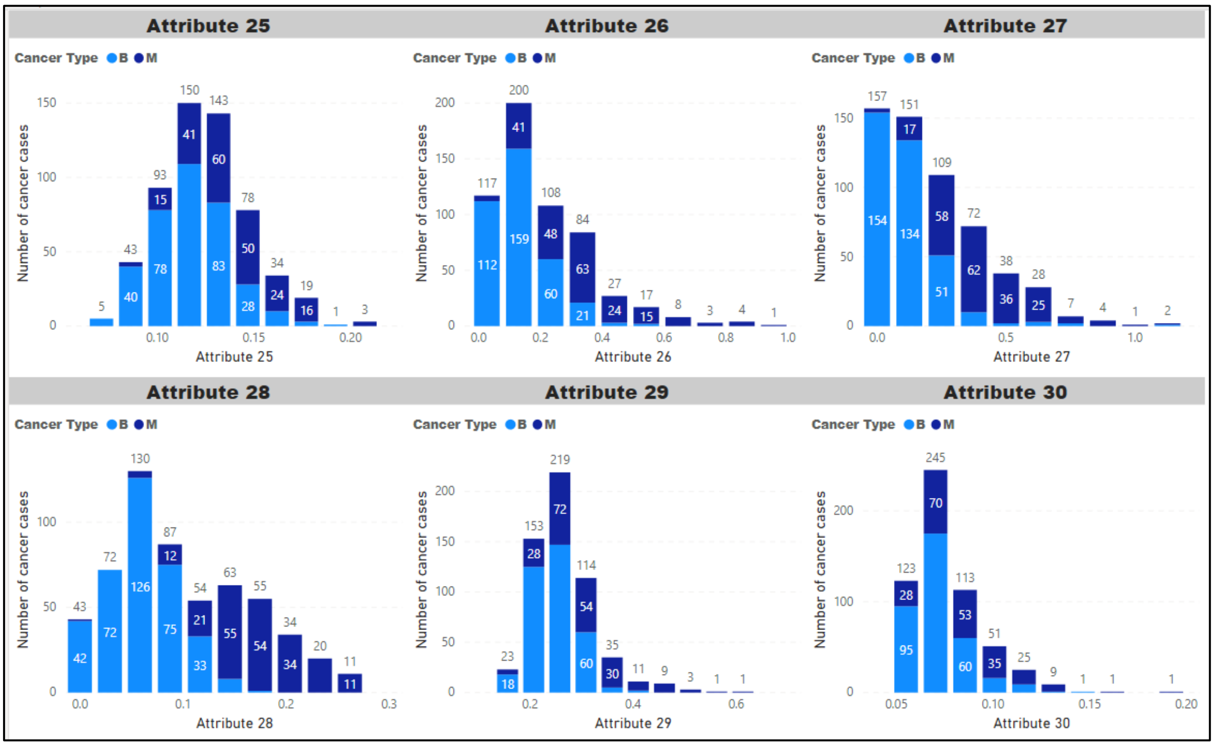
$$Error = \frac{Error1 + Error2}{2}$$




Attribute 7
Cancer Type B M
Count of Cancer Type
Attribute 7
Attribute 8
Cancer Type B M
Count of Cancer Type
Attribute 8
Attribute 9
Cancer Type B M
Count of Cancer Type
Attribute 9
Attribute 10
Cancer Type B M
Count of Cancer Type
Attribute 10
Attribute 11
Cancer Type B M
Count of Cancer Type
Attribute 11
Attribute 12
Cancer Type B M
Count of Cancer Type
Attribute 12


Attribute 13
Cancer Type B M
Number of cancer cases
Attribute 13
Attribute 14
Cancer Type B M
Number of cancer cases
Attribute 14
Attribute 15
Cancer Type B M
Number of cancer cases
Attribute 15
Attribute 16
Cancer Type B M
Number of cancer cases
Attribute 16
Attribute 17
Cancer Type B M
Number of cancer cases
Attribute 17
Attribute 18
Cancer Type B M
Number of cancer cases
Attribute 18


Attribute 19
Attribute 20
Attribute 21
Attribute 22
Attribute 23
Attribute 24
Cancer Type B M
Number of cancer cases


Attribute 25
Attribute 26
Attribute 27
Attribute 28
Attribute 29
Attribute 30
Cancer Type B M
Number of cancer cases

Attribute	Threshold	Error	Prediction Ability
1	13	28%	16th
2	19	41%	26th
3	85	9.83%	5th
4	380	15%	8th
5	0.10	33%	21st
6	0.08	20%	13th
7	0.04	13.5%	7th
8	0.04	9.8%	4th
9	0.19	29.5%	19th
10	Could not find one*	NA	NA
11	0.10	18.6%	12th
12	Could not find one*	NA	NA
13	0.75	18.4%	11th
14	6.5	11.8%	5th
15	Could not find one*	NA	NA
16	0.02	35.38%	24th
17	0	34.38%	22nd
18	0.01	34.56%	23rd
19	Could not find one*	NA	NA
20	0	40.85%	25th
21	13.5	16.26%	9th
22	27	29.43%	18th
23	90.5	8.22%	2nd
24	592	8.43%	3rd
25	0.13	29.26%	17th
26	0.23	20.08%	14th
27	0.13	16.46%	10th
28	0.12	7.38%	1st
29	0.30	21.45%	15th
30	0.09	31.03%	20th

Question 4

KNeighborsClassifier method from sklearn.neighbors library is used for classification. KNeighborsClassifier has a parameter 'n_neighbors' which takes the number of nearest neighbour. In this study, I used 1NN and 3NN classification rule as shown below.

```
KNN_1 = KNeighborsClassifier(n_neighbors=1)
```

```
KNN_3 = KNeighborsClassifier(n_neighbors=3)
```

Testing procedure: Leave-one-out cross validation (As suggested by Prof. Alexander)

LeaveOneOut method from sklearn.model_selection is used for doing the Leave-one-out Cross validation.

```
loo = LeaveOneOut()
```

Using the LeaveOneOut method the accuracy has been calculated as shown bellow.

```
accuracy_1nn = cross_val_score(KNN_1, X, Y, scoring='accuracy', cv = loo)
print('Accuracy of the 1NN classifier: %.3f (%.3f)' % (np.mean(accuracy_1nn), np.std(accuracy_1nn)))
```

```
Accuracy of the 1NN classifier: 0.951 (0.216)
```

```
accuracy_3nn = cross_val_score(KNN_3, X, Y, scoring='accuracy', cv = loo)
print('Accuracy of the 3NN classifier: %.3f (%.3f)' % (np.mean(accuracy_3nn), np.std(accuracy_3nn)))
```

```
Accuracy of the 3NN classifier: 0.965 (0.184)
```

As expected 3NN performs better than the 1NN algorithm

Model	AUC	CA	F1	Precision	Recall
1NN	0.945	0.951	0.951	0.951	0.951
3NN	0.982	0.965	0.965	0.965	0.965

Actual \ Predicted	B	M	Σ
B	345	12	357
M	16	196	212
Σ	361	208	569

[Confusion Matrix for 1NN]

Actual \ Predicted	B	M	Σ
B	354	3	357
M	17	195	212
Σ	371	198	569

[Confusion Matrix for 3NN]

From the results, we can conclude that 3NN is better than 1NN classification

Question 5

Fisher's Linear Discriminant projects multidimensional data points to a line in a way that each class is separable which makes it easy for classifying the data. Our objective is to find a projection in which the classes are well separated i.e the mean difference between each class should be maximum and the data points within each class should have a small variance. Fisher's algorithm does not have assumptions such as normally distributed classes or equal class covariances.

$$J(v) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\widetilde{s_1^2} + \widetilde{s_2^2}} \ldots (1)$$

Objective: To find the v which maximizes the function $J(v)$ by projecting the mean $(\tilde{\mu}_1 - \tilde{\mu}_2)^2$ far from each class and want scatter $(\widetilde{s_1^2} + \widetilde{s_2^2})$ inside both class 1 and class 2 to be small.

Consider projection on a line and the line direction be given by unit vector v.

$$S_1 = \sum_{x_i \in Class\ 1} (x_i - \mu_1)(x_i - \mu_1)^T \ldots (2)\ (Similarly\ we\ get\ S_2)$$

Within the class matrix

$$S_w = S_1 + S_2$$

We know that,

$$y_i = v^T x_i$$

$$\tilde{s}_1^2 = \sum_{y_i \in Class\ 1} (v^T x_i - v^T \mu_1)^2 = v^T S_1 v$$

$$\tilde{s}_2^2 = v^T S_2 v$$

Therefore,

$$\tilde{s}_1^2 + \tilde{s}_2^2 = v^T S_W v \ldots\ Substituting\ in\ equation\ 1$$

Defining between class scatter matrix $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$. This measures the separation between the mean of two classes (Before projection).

$$(\mu_1 - \mu_2)^2 = (v^T \mu_1 - v^T \mu_2)^2 = v^T S_B v \ldots\ Substituting\ in\ equation\ 2$$

$$J(v) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\widetilde{s_1^2} + \widetilde{s_2^2}} = \frac{v^T S_B v}{v^T S_W v}$$

Minimizing $J(v)$ with respect to the v and solving the derivative of $J(v)$ and setting it to zero, we get $S_B v = \lambda S_w v$. This is a generalized eigenvalue problem.

If S_w has full rank and is inversible, we can convert this to a standard eigenvalue problem. Further solving the equation, we end up with $v = S_W^{-1}(\mu_1 - \mu_2)$. We can easily get the new attribute by multiplying v with the matrix x.

Question 6 (Reference included)

Implementation of Fisher's Linear Discriminant

Finding the mean for each class $(\text{Mean})_{\text{class 1}}$ $[\mu_1]$ and $(\text{Mean})_{\text{class 2}}$ $[\mu_2]$

$$\mu = \mu_1 - \mu_2$$

```
mean = x_m.mean()-x_b.mean()
```

Finding the covariance matrix of class 1 (S_1) and class 2 (S_2)

$$S = S_1 + S_2$$

```
S_xmt = np.cov(x_mt)
```

```
S_ymt = np.cov(y_mt)
```

```
S = S_xmt+S_ymt
```

Taking the inverse of the covariance matrix S and multiplying it with the μ, we get the w.

```
Si = np.linalg.inv(S)
```

```
W = np.matmul(Si, mean)
```

```
Si.shape
```

```
(30, 30)
```

```
mean.shape
```

```
(30,)
```

```
W
```

```
array([ -2.59959375,  -1.02131711,  18.79920471, -11.53201769,
         1.57576641,  -1.29834852,  -2.1008193 ,   0.09086002,
        -0.65790628,  -0.56484001,   1.59307242,  -0.55563768,
        -2.71681955,   3.001523  ,  -0.33599296,   1.37420918,
        -0.92090242,   1.93124282,  -0.43886881,  -1.72572394,
         2.28475672,   2.23477337,   1.96367993,  -5.8976497 ,
         0.24390599,  -1.61197724,   1.30709359,   0.46140744,
         0.85924591,   1.48385997])
```

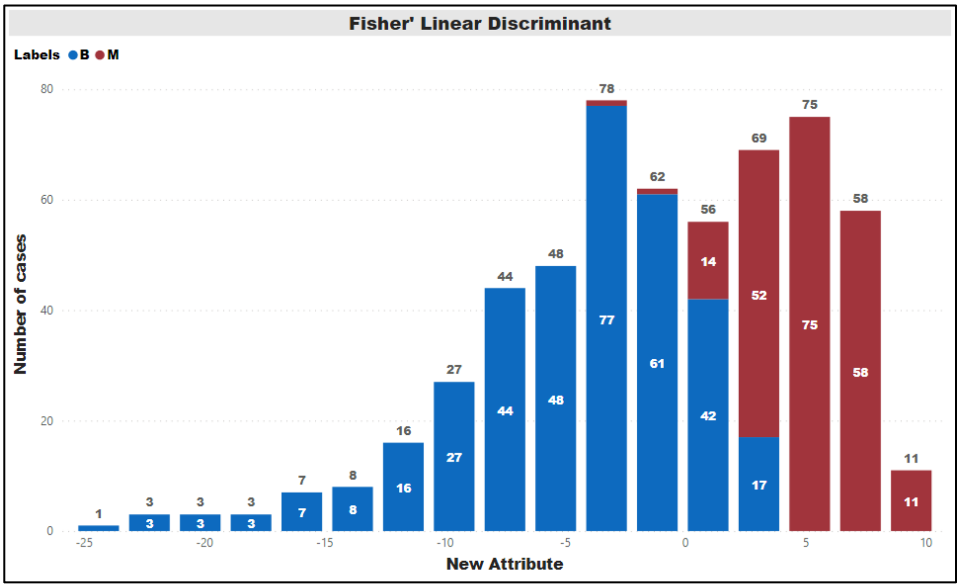
```
New_Attribute = np.matmul(x,W)
```

```
New_Attribute.shape
```

```
(569,)
```

```
                  0
count  5.690000e+02
mean   2.122887e-16
std    6.370808e+00
min   -2.437254e+01
25%   -3.941461e+00
50%    2.053414e-01
75%    5.344387e+00
max    1.152286e+01
```

The characteristics of the new attribute are attached above.



Bin size = 17

Confusion Matrix

B	M	569
340	17	B
16	196	M

Comparing 1NN, 3NN, Fisher's Linear Discriminant and Linear Discriminant:

Model	**Accuracy**	**Testing procedure**
1NN	95.1%	Leave one out cross-validation
3NN	96.5%	Leave one out cross-validation
Fisher's Linear Discriminant	93.76%	Simple histogram
LDA	96%	Train Test set (70% 30%)

Fisher's Linear Discriminant is a strong classifier as it makes the problem easy to solve by creating a new attribute that is capable of accurately classifying with help of a simple histogram as shown above.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=42)
```

```
clf = LinearDiscriminantAnalysis()
clf.fit(x_train, y_train)
```

```
LinearDiscriminantAnalysis()
```

```
pred = clf.predict(x_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

```
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
[[106   2]
 [  6  57]]
              precision    recall  f1-score   support

           B       0.95      0.98      0.96       108
           M       0.97      0.90      0.93        63

    accuracy                           0.95       171
   macro avg       0.96      0.94      0.95       171
weighted avg       0.95      0.95      0.95       171
```

Results from LinearDiscriminantAnalysis

Reference

Linear discriminant analysis - Wikipedia

https://www.csd.uwo.ca/~oveksler/Courses/CS434a_541a/Lecture8.pdf

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

https://scikitlearn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

Software Used:

1. Jupyter (Python)
2. Power BI (Creating visualizations - Histograms)
3. Orange

Note: This table denotes the real name of the attributes.

Attribute 1	Mean Radius
Attribute 2	Mean Area
Attribute 3	Mean Perimeter
Attribute 4	Mean Texture
Attribute 5	Mean Smoothness
Attribute 6	Mean Compactness
Attribute 7	Mean Concavity
Attribute 8	Mean Concave points
Attribute 9	Mean Symmetry
Attribute 10	Mean Fractal dimensions
Attribute 11	SE Radius
Attribute 12	SE Area
Attribute 13	SE Perimeter
Attribute 14	SE Texture
Attribute 15	SE Smoothness
Attribute 16	SE Compactness
Attribute 17	SE Concavity
Attribute 18	SE Concave points
Attribute 19	SE Symmetry
Attribute 20	SE Fractal dimensions
Attribute 21	Worst Radius
Attribute 22	Worst Area
Attribute 23	Worst Perimeter
Attribute 24	Worst Texture
Attribute 25	Worst Smoothness
Attribute 26	Worst Compactness
Attribute 27	Worst Concavity
Attribute 28	Worst Concave points
Attribute 29	Worst Symmetry
Attribute 30	Worst Fractal dimensions