Breast Cancer Wisconsin (Diagnostic)

Mohammad Al-Ajlouni C0849924

Assignment 2

AML 1413

Instructor: Kritika Dahiya

Problem Statement

The purpose of the task is to classify breast cancer using classification and compare the result with statistical model.

Data Set Information

Features are computed from a digitized image of a breast mass's fine needle aspirate (FNA). They describe the characteristics of the cell nuclei present in the image.

The separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method that uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

Attribute Information

1) ID number

2) Diagnosis (M = malignant, B = benign)

3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from the center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" - 1)

Analysis

```
df = pd.read_csv('breast-cancer.csv')
```

```
df.isna().sum()
```

```
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave points_mean         0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst     0
Unnamed: 32               569
dtype: int64
```

```
df.drop(columns = ['id','Unnamed: 32'],inplace = True)
```

After reading the data we notice that there are no null values, and we will drop the Id.

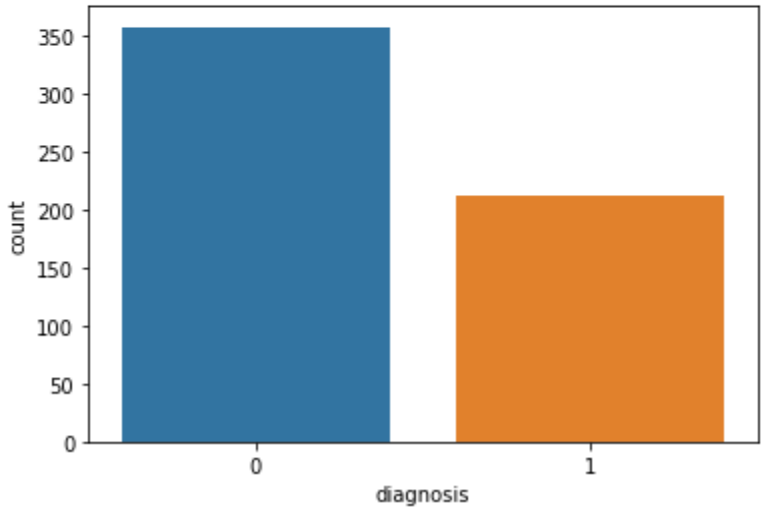
And this is how the data looks after converting the M and B into 1 and 0:

```
df['diagnosis'].unique()
```

```
array(['M', 'B'], dtype=object)
```

```
df['diagnosis'] = df['diagnosis'].map({"M":1,"B":0})
```

```
sns.countplot(x = 'diagnosis',data=df)
plt.show()
```




```
table_corr = df.corr()
```

```
table_corr_2 = pd.DataFrame(table_corr).query('diagnosis > 0.7')
```

```
table_corr_2.index
```

```
Index(['diagnosis', 'radius_mean', 'perimeter_mean', 'area_mean',
       'concave points_mean', 'radius_worst', 'perimeter_worst', 'area_worst',
       'concave points_worst'],
      dtype='object')
```

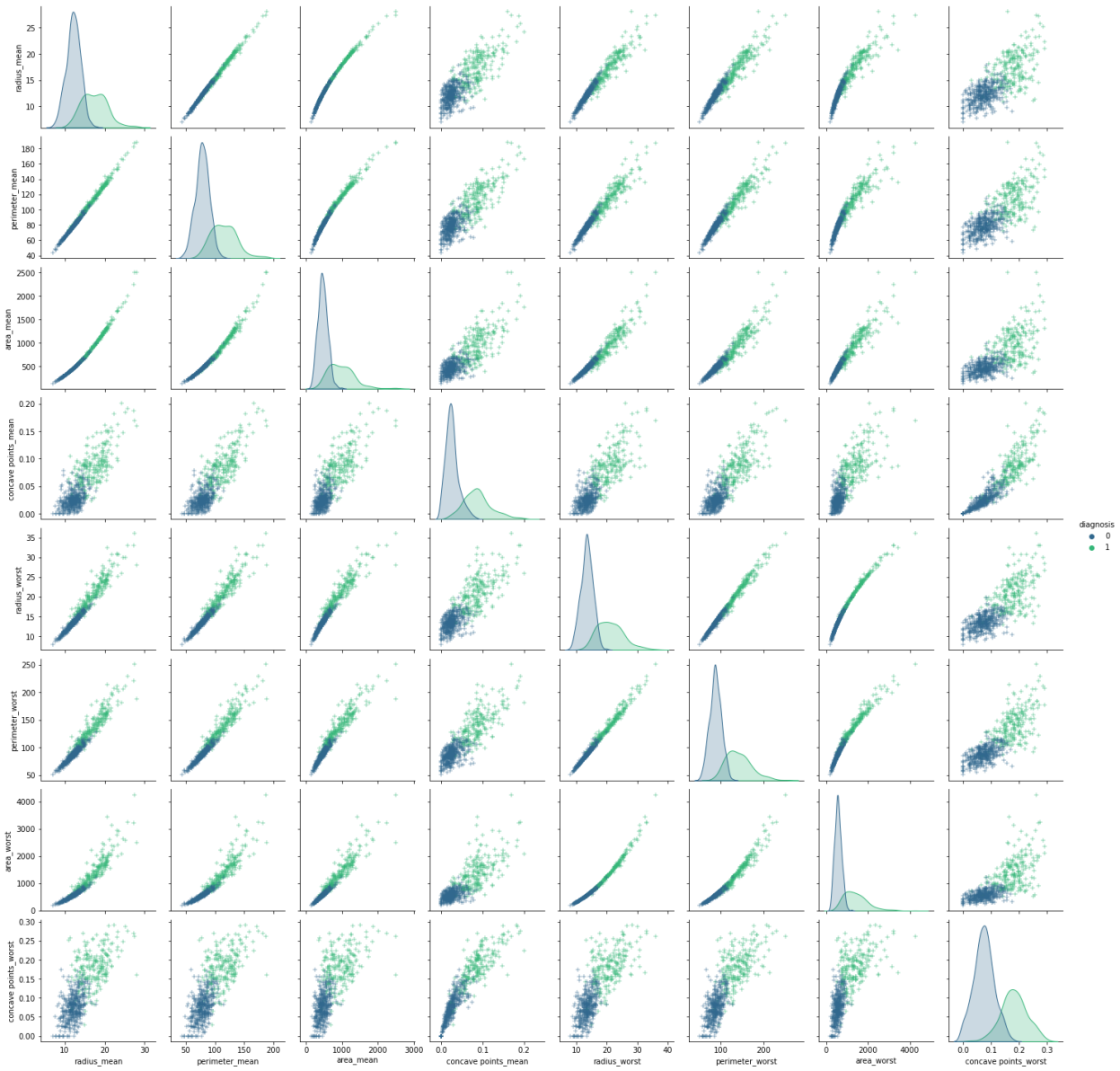
Anything with a correlation is less than 0.7 won't be used.

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactnes
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369
radius_se	0.567134	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473
texture_se	-0.008303	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205
perimeter_se	0.556141	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905
area_se	0.548236	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653
smoothness_se	-0.067016	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299
compactness_se	0.292999	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722
concavity_se	0.253730	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517
concave points_se	0.408042	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262
symmetry_se	-0.006522	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977
fractal_dimension_se	0.077972	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318
radius_worst	0.776454	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315
texture_worst	0.456903	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133
perimeter_worst	0.782914	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210
area_worst	0.733825	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604
smoothness_worst	0.421465	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541
compactness_worst	0.590998	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809
concavity_worst	0.659610	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275
concave points_worst	0.793566	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573
symmetry_worst	0.416294	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223
fractal_dimension_worst	0.323872	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382

Many of our features have linear dependence and normal distribution.

In [17]:

```
plt.figure(figsize=(20,14))
sns.pairplot(df,diag_kind = "kde", markers = "+", hue = "diagnosis", palette='viridis')
plt.show()
```



We clearly see that there are high indicators for malignant neoplasms and our classifier can determine benign and malignant neoplasms.

Machine Learning Algorithms

Four machine learning algorithms were used as follows:

1) Random Forest
2) Logistic Regression
3) XgBoost
4) Neural network (keras sequential)

```
clf_forest = RandomForestClassifier()
clf_log = LogisticRegression(solver = 'liblinear')
clf_xgb = XGBClassifier(objective ='binary:logistic')

clf = [clf_forest,clf_log,clf_xgb]

for i in clf:
    i.fit(X_train,y_train)
```

```
[17:22:26] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
```

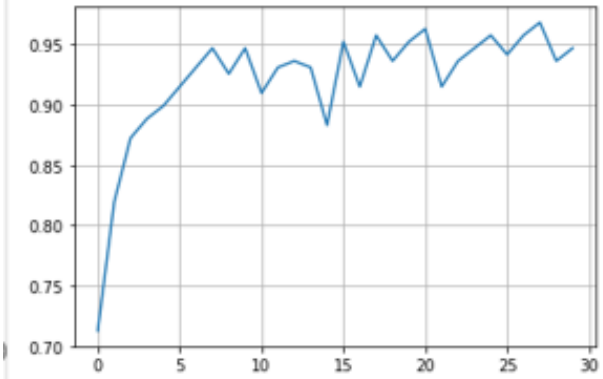
```
y_forest = clf_forest.predict(X_test)
y_logreg = clf_log.predict(X_test)
y_xbg = clf_xgb.predict(X_test)
```

```
m = ['random forest','logistic regression','xgboost']
f_forest = f1_score(y_test,y_forest)
f_logreg = f1_score(y_test,y_logreg)
f_xgb = f1_score(y_test,y_xbg)
f_score = pd.DataFrame({'f1 score' :[f_forest,f_logreg,f_xgb],'model': m})
```

```
model = keras.Sequential([layers.BatchNormalization()
                          ,layers.Dense(20, activation = 'relu', input_shape = [8]),
                          layers.BatchNormalization(),
                          layers.Dense(20, activation = 'relu'),
                          layers.BatchNormalization(),
                          layers.Dense(1, activation = 'sigmoid')])
model.compile(optimizer='adam',
              loss = 'binary_crossentropy',
              metrics = ['binary_accuracy'])
```

```
history = model.fit(X_test,y_test,
                    epochs=30)
```

```
history_score = pd.DataFrame(history.history)
plt.plot(history_score[['binary_accuracy']])
plt.grid()
plt.show()
```




```
predict = model.predict(X_test)
```

```
f_score.loc[len(f_score)] = [f1_score(y_test,np.rint(predict)),'Neural network']
```

```
f_score
```

	f1 score	model
0	0.919708	random forest
1	0.924242	logistic regression
2	0.900000	xgboost
3	0.851064	Neural network

```
y_forest = clf_forest.predict(X_test)
y_logreg = clf_log.predict(X_test)
y_xbg = clf_xgb.predict(X_test)
```

```
y_logreg
```

```
array([1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1], dtype=int64)
```

```
y_forest
```

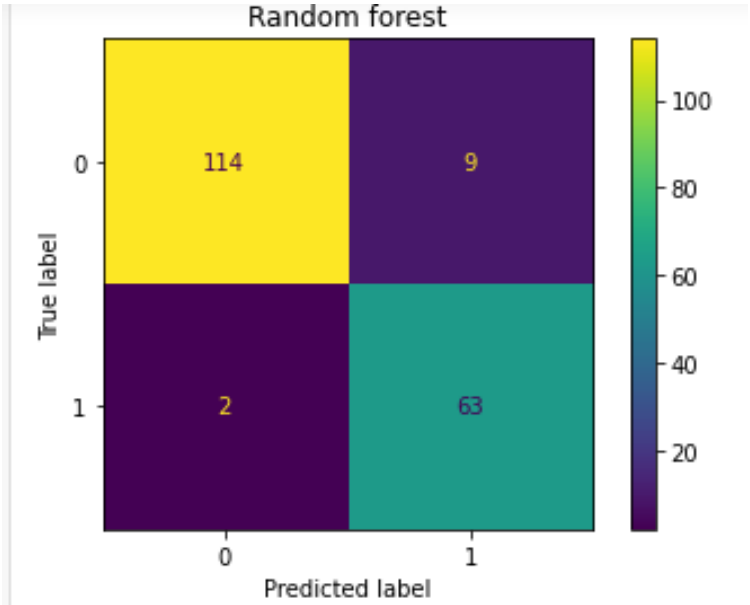
```
array([1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1], dtype=int64)
```

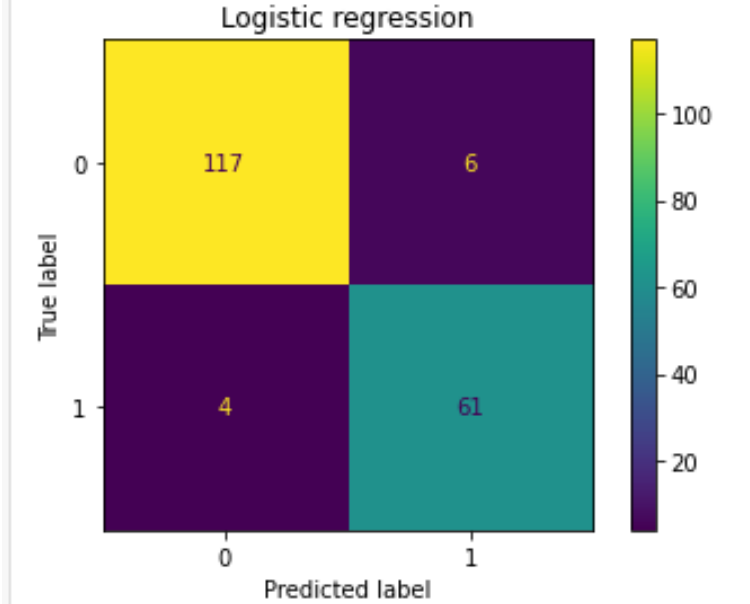
```
y_xbg
```

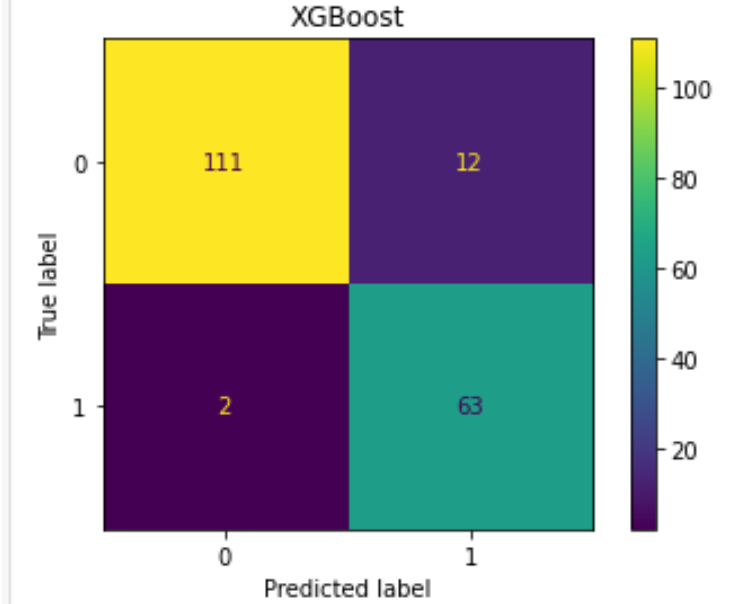
```
array([1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1], dtype=int64)
```

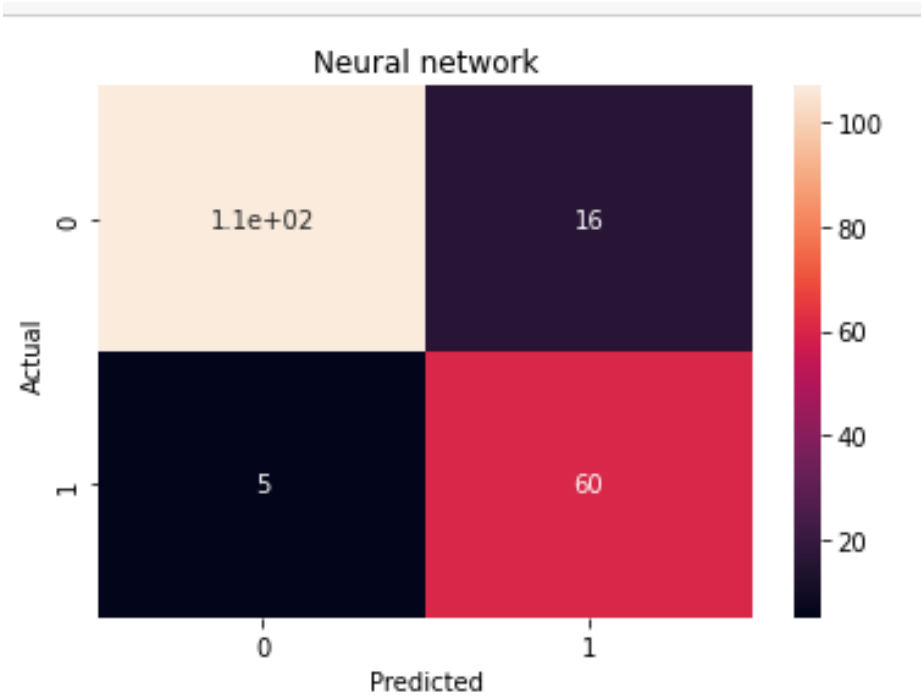
```
predict
```

```
array([[0.6415359 ],
       [0.1112051 ],
       [0.13520536],
       [0.377199  ],
       [0.46747422],
       [0.08948678],
       [0.12892291],
       [0.15521216],
       [0.18692917],
       [0.4840555 ],
       [0.95222604],
       [0.13963908],
       [0.18843445],
       [0.6095203 ],
       [0.5036391 ],
       [0.5075569 ],
       [0.5804636 ],
       [0.38034436],
       [0.16678724],
```


Random forest
114
9
2
63
0
1
True label
Predicted label
100
80
60
40
20


Logistic regression
117
6
4
61
0
1
True label
Predicted label
100
80
60
40
20


XGBoost
111
12
2
63
0
1
True label
Predicted label
100
80
60
40
20



Conclusion

1) F1-score is the highest for logistic regression. The neural network shows minimal results. It makes sense to evaluate these algorithms if you use machine learning algorithms without their default settings.
2) The roc-auc metric shows high results in a neural network. for binary classification, this metric shows itself best because it displays the true and constant ability of the model to predict.
3) Graphs of the importance of features are constructed and they have different weights for different models, this is due to the peculiarities of their mathematics. The graphs of the random forest and xgboost are similar, this is due to the fact that they are based on the construction of decision trees.

References

https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/code

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29