Faculty of Engineering
Cairo University

CDSS Project 1

Team 6

Breast Cancer Classification Using Different ML Models

Name	ID
Aya Amr	9202342
Sara Ayman	9202615
Shuaib Abdulsalam	9204062
Abdelrahman Saeed	9202839
Mahmoud Rabea	9203396

Table of Contents

Introduction About Breast Cancer

Breast cancer is a cancer that develops in the breast cells and progresses in stages. Few early symptoms may include new lump in the underarm or in breast, itching or discharge from the nipples, and skin texture change of the nipple or breast.

Cancer begins when healthy cells in the breast change and grow out of control, forming a mass or sheet of cells called a tumor. A tumor can be cancerous or benign. A cancerous tumor is malignant, meaning it can grow and spread to other parts of the body.

The exact cause of breast cancer is not known but risk factors include family history, hormonal changes, age (at more risk after 40 years of age), personal history of breast cancer, lifestyle including excess of alcohol consumption, environmental factors including exposure to radiations, obesity and overweight, menarche having periods at younger age and menopause at an older age, pregnancy becoming pregnant at an older age or never being pregnant, hormone use including long-term contraceptive use or postmenopausal hormone therapy.

Preventive measures involve healthy habits such as eating healthy and nutritious food, avoiding alcohol, practicing gentle exercises upon doctor's advice, visiting doctor for regular examination, preventive surgery may be recommended in women with high risk. To reduce the risk of developing cancer, get the pre-screening done.

Dataset Description

The dataset we used is used to predict the type of cancer either it is malignant or benign based on the input parameters like radius mean, texture mean, perimeter mean, area mean, smoothness mean and other parameters shown below.

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
 'smoothness_mean', 'compactness_mean', 'concavity_mean',
 'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
 'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
 'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
 'fractal_dimension_se', 'radius_worst', 'texture_worst',
 'perimeter_worst', 'area_worst', 'smoothness_worst',
 'compactness_worst', 'concavity_worst', 'concave points_worst',
 'symmetry_worst', 'fractal_dimension_worst'],
```

The dataset has 357 records with benign tumor and 212 records with malignant tumor, where 0 or M presents that the patient has malignant tumor while 1 or B represents that the patient has benign tumor.

Used ML Models

- **Logistic Regression**

```
X = df.drop(columns=['id','diagnosis','Unnamed: 32'])
Y = df['diagnosis']
Y = Y.ravel()
```
✓ 0.0s

```
x_train , x_test , y_train , y_test = train_test_split(X,Y,test_size=0.3,random_state=0)
```
✓ 0.0s

```
model = LogisticRegression(solver='liblinear', random_state=0)
result = model.fit(x_train,y_train)
```
✓ 0.0s

```
y_pred = model.predict(x_test)
```
✓ 0.0s

```
accuracy = sklearn.metrics.accuracy_score(y_test,y_pred)
accuracy_percentage = 100 * accuracy
accuracy_percentage
```
✓ 0.0s

```
96.49122807017544
```

```
print(classification_report(y_test, y_pred))
```
✓ 0.1s

```
              precision    recall  f1-score   support

           B       0.99      0.95      0.97       108
           M       0.93      0.98      0.95        63

    accuracy                           0.96       171
   macro avg       0.96      0.97      0.96       171
weighted avg       0.97      0.96      0.97       171
```

- **Decision Tree**

```
X = df.drop(columns=['diagnosis','Unnamed: 32']) #Unamed:32 contains Nan values so we drop it
y = df['diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20) #split the data
```

✓ 0.0s

```
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test) #make prediciton
```

✓ 0.0s

```
ac = accuracy_score(y_test,classifier.predict(X_test))
print('Accuracy is: ',ac*100)
```

✓ 0.0s

```
Accuracy is:  88.59649122807018
```

```
#print the classification report
print("Classification Report:\n",classification_report(y_test, y_pred))
```

✓ 0.0s

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

           B       0.86      0.95      0.90        62
           M       0.93      0.81      0.87        52

    accuracy                           0.89       114
   macro avg       0.89      0.88      0.88       114
weighted avg       0.89      0.89      0.88       114
```

- **Random Forest**

```
y=df.diagnosis
x = df.drop(columns=['Unnamed: 32', 'id','diagnosis'])
x.head()
```

✓ 0.1s Python

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean	...	radius_worst	texture_worst	perimeter_worst	area_worst	smoot
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	

5 rows × 30 columns

```
x.columns
```

✓ 0.0s

```
Index(['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
       'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

```
drop_list = ['perimeter_mean','radius_mean','compactness_mean','concave points_mean',
             'radius_se','perimeter_se','radius_worst','perimeter_worst','compactness_worst',
             'concavity_worst','compactness_se','concave points_se','texture_worst','area_worst']
x1 = x.drop(drop_list ,axis = 1 )
x1.head()
```

✓ 0.0s Python

	texture_mean	area_mean	smoothness_mean	concavity_mean	symmetry_mean	fractal_dimension_mean	texture_se	area_se	smoothness_se	concavity_se	symmetry_se	fractal_dimension_se	smoothness_worst	concave points_worst	symmetry_wo
0	10.38	1001.0	0.11840	0.3001	0.2419	0.07871	0.9053	153.40	0.006399	0.05373	0.03003	0.006193	0.1622	0.2654	0.46
1	17.77	1326.0	0.08474	0.0869	0.1812	0.05667	0.7339	74.08	0.005225	0.01860	0.01389	0.003532	0.1238	0.1860	0.27
2	21.25	1203.0	0.10960	0.1974	0.2069	0.05999	0.7869	94.03	0.006150	0.03832	0.02250	0.004571	0.1444	0.2430	0.36
3	20.38	386.1	0.14250	0.2414	0.2597	0.09744	1.1560	27.23	0.009110	0.05661	0.05963	0.009208	0.2098	0.2575	0.66
4	14.34	1297.0	0.10030	0.1980	0.1809	0.05883	0.7813	94.44	0.011490	0.05688	0.01756	0.005115	0.1374	0.1625	0.23

```
x_train, x_test, y_train, y_test = train_test_split(x1, y, test_size=0.2, random_state=10)

#n_estimators=10 (default)
clf_rf = RandomForestClassifier(random_state=42)
clr_rf = clf_rf.fit(x_train,y_train)

ac = accuracy_score(y_test,clf_rf.predict(x_test))
print('Accuracy is: ',ac*100)
```

✓ 0.3s

```
Accuracy is:  98.24561403508771
```

```
print(classification_report(y_test, clf_rf.predict(x_test)))
```

✓ 0.0s

```
              precision    recall  f1-score   support

           B       1.00      0.97      0.99        75
           M       0.95      1.00      0.97        39

    accuracy                           0.98       114
   macro avg       0.98      0.99      0.98       114
weighted avg       0.98      0.98      0.98       114
```

- **SVM (Support Vector Machine)**

```
df = df.drop(['Unnamed: 32','id'],axis=1)
df.info()
```

```
Output exceeds the size limit. Open the full output data in a text editor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   diagnosis                569 non-null    object
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
...
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

```
df = df.replace({'B':0,'M':1})
X = df.drop(('diagnosis'),axis = 1)
Y = df[['diagnosis']]
```

```
from sklearn.model_selection import train_test_split

# test set size of 20% of the data
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

from sklearn import svm



model = svm.SVC(kernel = 'linear', random_state = 0, C=1.0)

model.fit(X_train, y_train.values.ravel())

prediction=model.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
score =accuracy_score(prediction, y_test)
print(score*100,'%')
```

```
95.6140350877193 %
```

```
#classification report
from sklearn.metrics import classification_report
print('classification report')
print(classification_report(y_test,prediction))
```

```
classification report
              precision    recall  f1-score   support

           0       0.95      0.99      0.97        71
           1       0.97      0.91      0.94        43

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

SVM but with normalizing features between (-1,1)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVC

df = pd.read_csv('BreastCancer.csv')
df = df.drop(['Unnamed: 32','id'],axis=1)
df = df.replace({'B':0,'M':1})
X = df.drop(('diagnosis'),axis = 1)
Y = df[['diagnosis']]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)

model = svm.SVC(kernel = 'linear', random_state = 0, C=1.0)

model.fit(X_train_scaled, y_train.values.ravel())

X_test_scaled = scaler.transform(X_test)
prediction = model.predict(X_test_scaled)
```

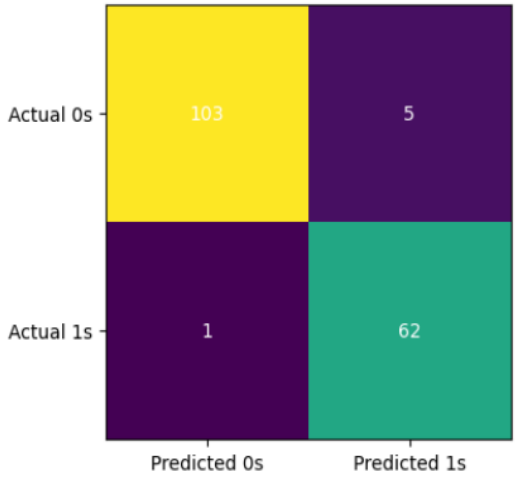
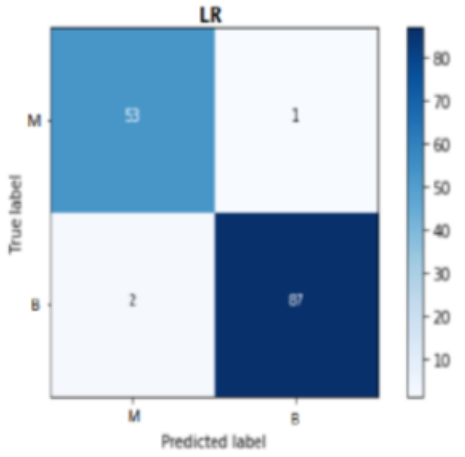
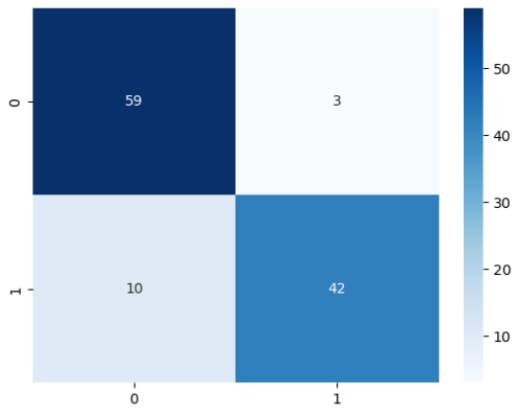
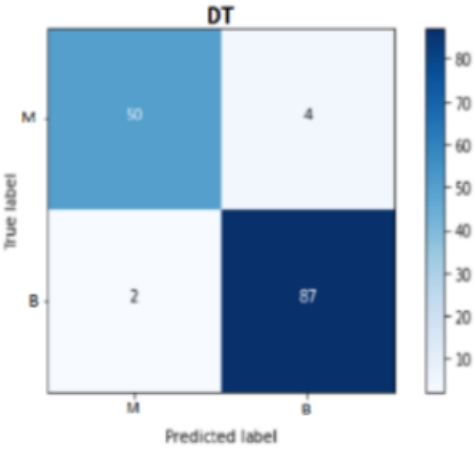
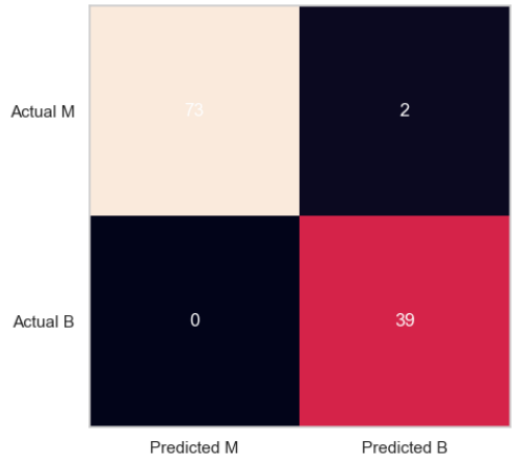
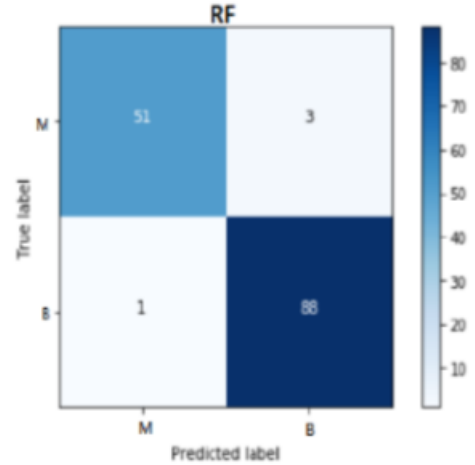
```
from sklearn.metrics import accuracy_score
score =accuracy_score(prediction, y_test)
print(score*100,'%')
```

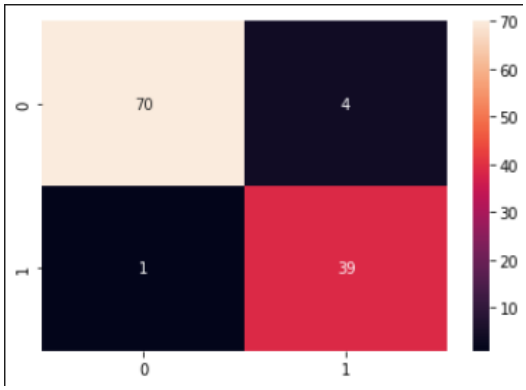
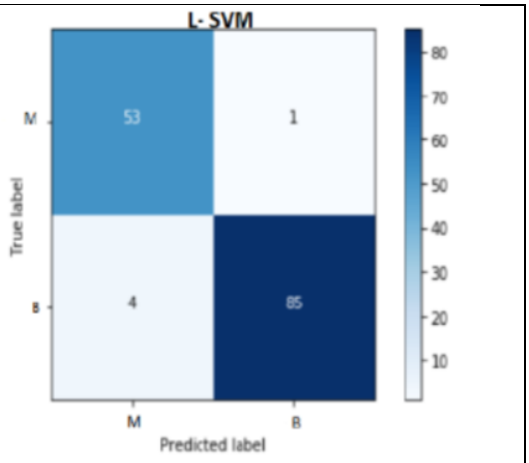
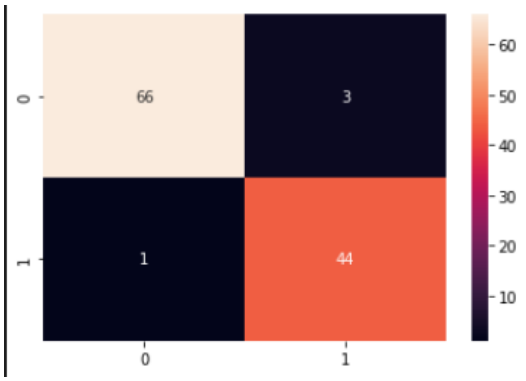
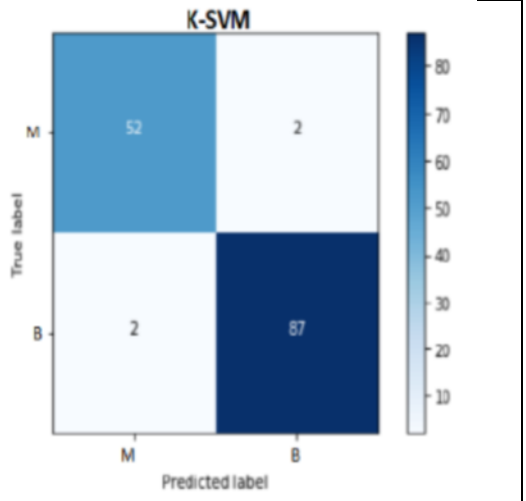
```
96.49122807017544 %
```

```
from sklearn.metrics import classification_report

print('classification report')
print(classification_report(y_test,prediction))
```

```
classification report
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        67
           1       0.98      0.94      0.96        47

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

Results Comparison

	Our results	Paper results
LR		
DT		
RF		

<table>
<tr><td rowspan="2">SVM</td><td>Without data normalization
 
</td><td> 
</td></tr>
<tr><td>With data normalization
 
</td><td> 
</td></tr>
<tr><td colspan="3">$$Accuarcy = \frac{TP + TN}{TP + TN + FP + FN}$$</td></tr>
</table>

Links

- **Paper link**
  https://www.researchgate.net/publication/346617710_Breast_cancer_classification_using_machine_learning_techniques_a_comparative_study

- **Dataset Link**
  https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data