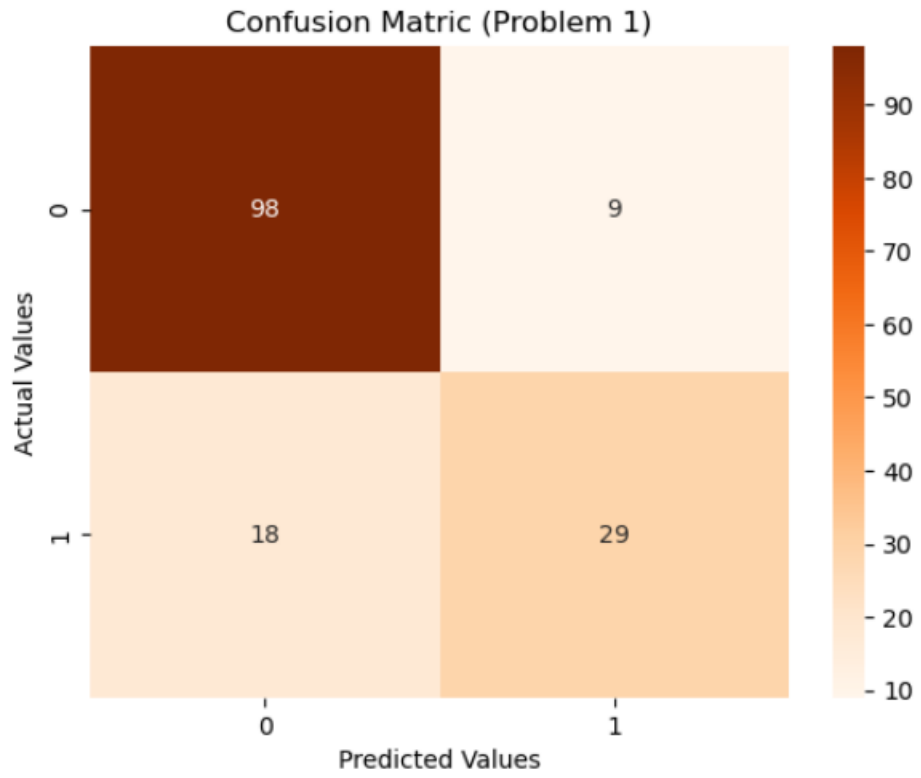
ECGR 4105 – HW # 1

Nahush D. Tambe - 801060297

https://github.com/Ntambe25

Problem 1:




```
print("Accuracy = ", Accuracy)
print("Precision = ", Precision)
print("Recall = ", Recall)
```

```
Accuracy =  0.8246753246753247
Precision =  0.7631578947368421
Recall =  0.6170212765957447
```

After performing the proper scaling using MinMax and Standard Scalar, the values for Accuracy, Precision, and Recall were generated using the right library call from Sklearn. With the three values printed, a confusion matrix was also generated. Figure, given above, is a confusion matrix for problem 1.

Problem 2:

```
# K-fold Cross-Validation for Training and Validation
Kfold = KFold(n_splits = 5, random_state = 0, shuffle = True)
model = LogisticRegression(solver = "liblinear")
results = cross_val_score(model, X, Y, cv = Kfold)

print("Accuracy = ", results.mean())
```

```
Accuracy =  0.7760037348272643
```

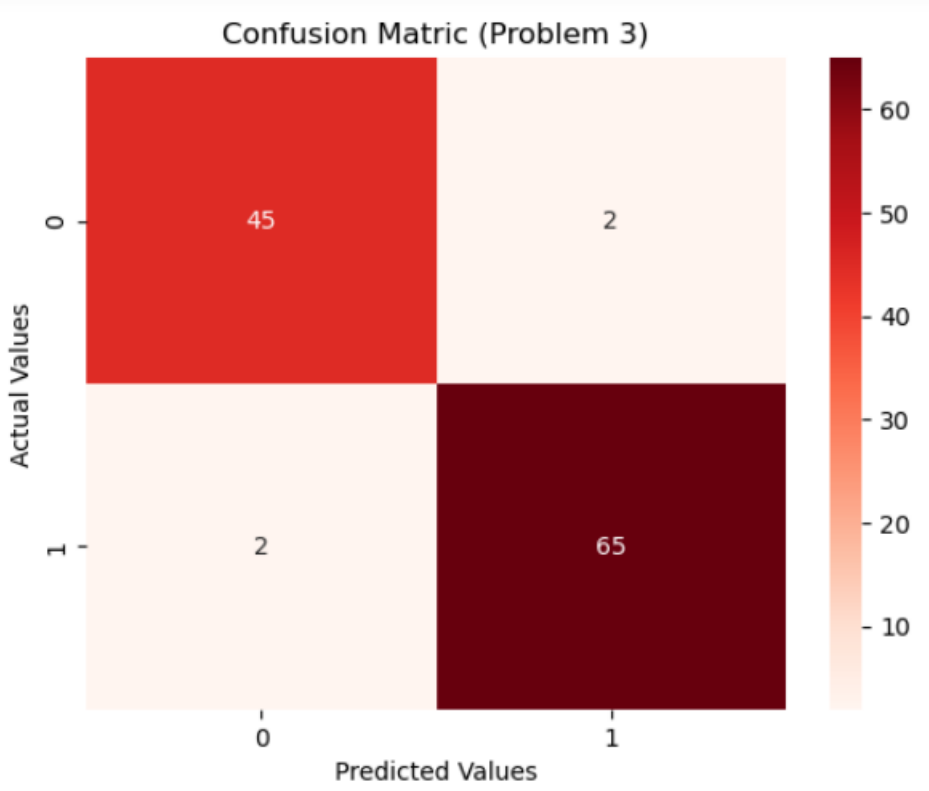
```
# K-fold Cross-Validation for Training and Validation
Kfold = KFold(n_splits = 10, random_state = 0, shuffle = True)
model = LogisticRegression(solver = "liblinear")
results = cross_val_score(model, X, Y, cv = Kfold);

print("Accuracy = ", results.mean())
```

```
Accuracy =  0.7760423786739576
```

For problem 2, K-fold Cross Validation was used, first with a value of K = 5, and then with a value of K = 10. As it can be seen, the accuracy of both the K values was almost the same, and slightly less than the accuracy for problem 1.

Problem 3:



Using the same techniques for earlier problems, a confusion matrix and the Accuracy, Precision, and Recall values were generated. Figure, given above, is the confusion matrix for problem 3. After the confusion matrix, penalty with a C value of 0.1 was added which increased the accuracy slightly.

```
print("Accuracy = ", Accuracy)
print("Precision = ", Precision)
print("Recall = ", Recall)
```

```
Accuracy =  0.9649122807017544
Precision =  0.9701492537313433
Recall =  0.9701492537313433
```

```
clf = LogisticRegression(penalty = "l1", C = 0.1, solver = "liblinear")
clf.fit(X_train, Y_train)
print("Training Accuracy = ", clf.score(X_train, Y_train))
print("Test Accuracy = ", clf.score(X_test, Y_test))
```

```
Training Accuracy =  0.9758241758241758
Test Accuracy =  0.9736842105263158
```

As it can be seen from above images, the accuracy with the penalty parameter added was slightly better. A parameter penalty value with a range from 0.001 to 10 was experimented with, and the value that gave the highest test accuracy was used, which was 0.1.

Problem 4:

```
# K-fold Cross-Validation for Training and Validation
Kfold = KFold(n_splits = 5, random_state = 0, shuffle = True)
model = LogisticRegression(solver = "liblinear")
results = cross_val_score(model, X_Cancer, Y_Cancer, cv = Kfold)

print("Accuracy = ", results.mean())
```

```
Accuracy =  0.9719298245614034
```

```
# K-fold Cross-Validation for Training and Validation
Kfold = KFold(n_splits = 10, random_state = 0, shuffle = True)
model = LogisticRegression(solver = "liblinear")
results = cross_val_score(model, X_Cancer, Y_Cancer, cv = Kfold);

print("Accuracy = ", results.mean())
```

```
Accuracy =  0.9789473684210526
```

```
clf = LogisticRegression(penalty = "l1", C = 0.1, solver = "liblinear")
clf.fit(X_train, Y_train)
print("Training Accuracy = ", clf.score(X_train, Y_train))
print("Test Accuracy = ", clf.score(X_test, Y_test))
```

```
Training Accuracy =  0.9758241758241758
Test Accuracy =  0.9736842105263158
```

As it can be seen from above image, the accuracy values for K-fold Cross validation for the cancer dataset with a K values of 5 and 10, along with an accuracy score with parameter penalty added were generated.
