In [125…
```
'''
Patrick Ballou
ID: 801130521
ECGR 4105
Homework 2
Problem 3
'''
```

Out[125]:
```
'\nPatrick Ballou\nID: 801130521\nECGR 4105\nHomework 2\nProblem 3\n'
```

In [126…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

In [127…
```
breast = load_breast_cancer()
```

In [128…
```
breast_data = breast.data
breast_data.shape
```

Out[128]:
```
(569, 30)
```

In [129…
```
breast_input = pd.DataFrame(breast_data)
```

In [130…
```
breast_labels = breast.target
breast_labels.shape
```

Out[130]:
```
(569,)
```

In [131…
```
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data, labels],axis=1)
final_breast_data.shape
```

Out[131]:
```
(569, 31)
```

In [132…
```
breast_dataset = pd.DataFrame(final_breast_data)
features = breast.feature_names
features
```

Out[132]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [133…

```
features_labels = np.append(features, 'label')
breast_dataset.columns = features_labels
breast_dataset.head()
```

Out[133]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	dim
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	(
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	(
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	(
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	(
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	(

5 rows × 31 columns

In [134…

```
df_train, df_test = train_test_split(breast_dataset, train_size=.8, test_size=.2, ranc
```

In [96]:

```
#split into x and y, train and test
x_train = df_train[features]
Y_train = df_train['label']

x_test = df_test[features]
Y_test = df_test['label']
```

In [135…

```
#standard scaler is best here
scaler = StandardScaler()
#scaler = MinMaxScaler()
X_train = scaler.fit_transform(x_train)
X_test = scaler.fit_transform(x_test)
```

In [136…

```
#tested which C value is best with for loop and found .1 to perform best
'''
C = [10, 1, .1, .01, .001]

for c in C:
    classifier = LogisticRegression(random_state=7, C=c)
    classifier.fit(X_train, Y_train)
    print("C:", c)
    print("Training accuracy:", classifier.score(X_train, Y_train))
    print("Testing accuracy:", classifier.score(X_test, Y_test))

'''
#C=.1 is the best
classifier = LogisticRegression(random_state=7, C=.1)
classifier.fit(X_train, Y_train)
```

Out[136]:

```
▾ LogisticRegression
LogisticRegression(C=0.1, random_state=7)
```

In [137…

```
Y_pred = classifier.predict(X_test)
cnf_matrix = metrics.confusion_matrix(Y_test, Y_pred)
```

```
cnf_matrix
```

Out[137]:

```
array([[39,  1],
       [ 0, 74]], dtype=int64)
```

In [138…

```
print("Accuracy:", metrics.accuracy_score(Y_test, Y_pred))
print("Precision:", metrics.precision_score(Y_test, Y_pred))
print("Recall:", metrics.recall_score(Y_test, Y_pred))
```

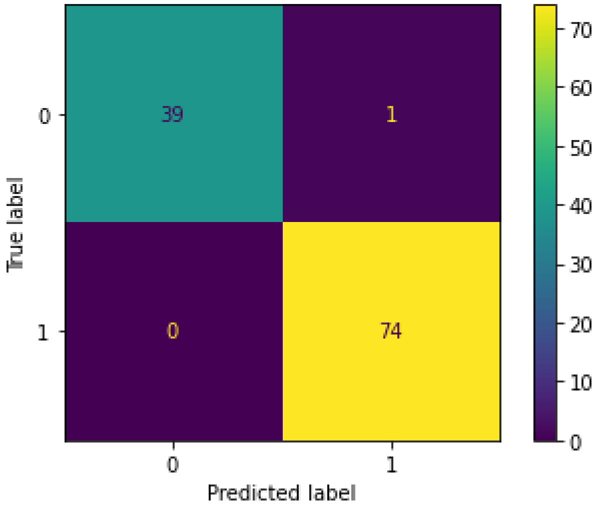
```
Accuracy: 0.9912280701754386
Precision: 0.9866666666666667
Recall: 1.0
```

In [139…

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cnf_matrix, display_label
cm_display.plot()
```

Out[139]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x20c37a84e20>
```



In [140…

```
#3b: add penalty
'''
C = [10, 1, .1, .01, .001]

for c in C:
    classifier = LogisticRegression(random_state=7, C=c, penalty='l2')
    classifier.fit(X_train, Y_train)
    print("C:", c)
    print("Training accuracy:", classifier.score(X_train, Y_train))
    print("Testing accuracy:", classifier.score(X_test, Y_test))
'''
#C=.1 is the best
classifier = LogisticRegression(random_state=7, C=.1)
classifier.fit(X_train, Y_train)
```

Out[140]:

```
▾ LogisticRegression
LogisticRegression(C=0.1, random_state=7)
```

In [141…

```
Y_pred = classifier.predict(X_test)
cnf_matrix = metrics.confusion_matrix(Y_test, Y_pred)
cnf_matrix
```

Out[141]:

```
array([[39,  1],
       [ 0, 74]], dtype=int64)
```

In [142…

```
print("Accuracy:", metrics.accuracy_score(Y_test, Y_pred))
print("Precision:", metrics.precision_score(Y_test, Y_pred))
print("Recall:", metrics.recall_score(Y_test, Y_pred))
```

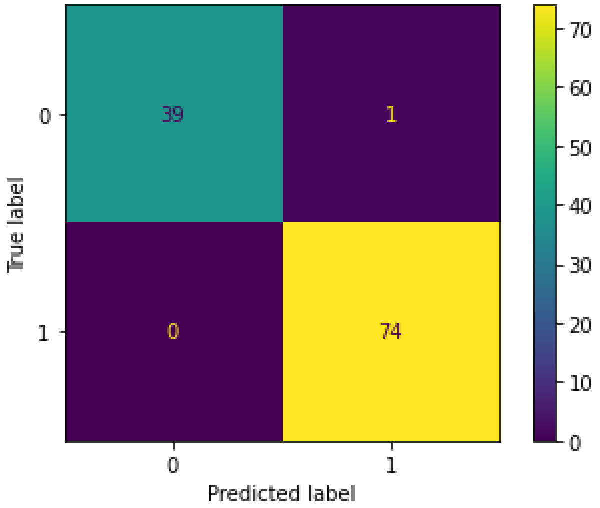
```
Accuracy: 0.9912280701754386
Precision: 0.9866666666666667
Recall: 1.0
```

In [143…

```
#this model is very good
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cnf_matrix, display_label
cm_display.plot()
```

Out[143]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x20c37b31ee0>
```


True label
Predicted label