Logistic Regression

```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
from sklearn.linear_model import LogisticRegression

# Import necessary modules
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report,confusion_matrix
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
```

```
df = pd.read_csv('breast_cancer_data.csv')
df.head()
```

	id	diagnosis	radius_1ean	texture_1ean	peri1eter_1ean	area_1ean	s1o
0	842302	1	17.99	10.38	122.80	1001.0	
1	842517	1	20.57	17.77	132.90	1326.0	
2	84300903	1	19.69	21.25	130.00	1203.0	
3	84348301	1	11.42	20.38	77.58	386.1	
4	84358402	1	20.29	14.34	135.10	1297.0	

```
print(df.shape)
df.describe().transpose()
```

id	569.0	3.037183e+07	1.250206e+08	8670.000000	869218.0000(
diagnosis	569.0	3.725835e-01	4.839180e-01	0.000000	0.0000(
radius_1ean	569.0	1.412729e+01	3.524049e+00	6.981000	11.7000(
texture_1ean	569.0	1.928965e+01	4.301036e+00	9.710000	16.1700(
peri1eter_1ean	569.0	9.196903e+01	2.429898e+01	43.790000	75.1700(
area_1ean	569.0	6.548891e+02	3.519141e+02	143.500000	420.3000(
s1oothness_1ean	569.0	9.636028e-02	1.406413e-02	0.052630	0.0863(
co1pactness_1ean	569.0	1.043410e-01	5.281276e-02	0.019380	0.0649(
concavity_1ean	569.0	8.879932e-02	7.971981e-02	0.000000	0.0295(
concave points_1ean	569.0	4.891915e-02	3.880284e-02	0.000000	0.0203(
sy11etry_1ean	569.0	1.811619e-01	2.741428e-02	0.106000	0.1619(
fractal_di1ension_1ean	569.0	6.279761e-02	7.060363e-03	0.049960	0.0577(
radius_se	569.0	4.051721e-01	2.773127e-01	0.111500	0.2324(
texture_se	569.0	1.216853e+00	5.516484e-01	0.360200	0.8339(
peri1eter_se	569.0	2.866059e+00	2.021855e+00	0.757000	1.6060(
area_se	569.0	4.033708e+01	4.549101e+01	6.802000	17.8500(
s1oothness_se	569.0	7.040979e-03	3.002518e-03	0.001713	0.0051(
co1pactness_se	569.0	2.547814e-02	1.790818e-02	0.002252	0.0130(
concavity_se	569.0	3.189372e-02	3.018606e-02	0.000000	0.0150(
concave points_se	569.0	1.179614e-02	6.170285e-03	0.000000	0.0076(
sy11etry_se	569.0	2.054230e-02	8.266372e-03	0.007882	0.0151(
fractal_di1ension_se	569.0	3.794904e-03	2.646071e-03	0.000895	0.0022(
radius_worst	569.0	1.626919e+01	4.833242e+00	7.930000	13.0100(
texture_worst	569.0	2.567722e+01	6.146258e+00	12.020000	21.0800(
peri1eter_worst	569.0	1.072612e+02	3.360254e+01	50.410000	84.1100(
area_worst	569.0	8.805831e+02	5.693570e+02	185.200000	515.3000(
s1oothness_worst	569.0	1.323686e-01	2.283243e-02	0.071170	0.1166(
co1pactness_worst	569.0	2.542650e-01	1.573365e-01	0.027290	0.1472(

```
target_column = ['diagnosis']
predictors = list(set(list(df.columns))-set(target_column))
df[predictors] = df[predictors]/df[predictors].max()
df.describe().transpose()
```

id	569.0	0.033327	0.137186	0.000010	0.000954	0.000994	0.00
diagnosis	569.0	0.372583	0.483918	0.000000	0.000000	0.000000	1.00
radius_1ean	569.0	0.502572	0.125366	0.248346	0.416222	0.475631	0.56
texture_1ean	569.0	0.491081	0.109497	0.247200	0.411660	0.479633	0.55
peri1eter_1ean	569.0	0.487899	0.128907	0.232308	0.398780	0.457507	0.55
area_1ean	569.0	0.261851	0.140709	0.057377	0.168053	0.220352	0.31
s1oothness_1ean	569.0	0.589720	0.086072	0.322093	0.528580	0.586720	0.64
co1pactness_1ean	569.0	0.302087	0.152903	0.056109	0.187956	0.268182	0.37
concavity_1ean	569.0	0.208058	0.186785	0.000000	0.069260	0.144189	0.30
concave points_1ean	569.0	0.243137	0.192857	0.000000	0.100944	0.166501	0.36
sy11etry_1ean	569.0	0.595927	0.090179	0.348684	0.532566	0.589474	0.64
fractal_di1ension_1ean	569.0	0.644475	0.072459	0.512726	0.592159	0.631568	0.67
radius_se	569.0	0.141028	0.096524	0.038810	0.080891	0.112844	0.16
texture_se	569.0	0.249100	0.112927	0.073736	0.170706	0.226817	0.30
peri1eter_se	569.0	0.130394	0.091986	0.034440	0.073066	0.104049	0.15
area_se	569.0	0.074395	0.083901	0.012545	0.032921	0.045242	0.08
s1oothness_se	569.0	0.226180	0.096451	0.055027	0.166046	0.204947	0.26
co1pactness_se	569.0	0.188169	0.132261	0.016632	0.096603	0.151034	0.23
concavity_se	569.0	0.080540	0.076227	0.000000	0.038106	0.065379	0.10
concave points_se	569.0	0.223454	0.116884	0.000000	0.144686	0.207047	0.27
sy11etry_se	569.0	0.260194	0.104704	0.099835	0.192020	0.237239	0.29
fractal_di1ension_se	569.0	0.127175	0.088675	0.029987	0.075335	0.106803	0.15
radius_worst	569.0	0.451420	0.134108	0.220033	0.360988	0.415372	0.52
texture_worst	569.0	0.518313	0.124067	0.242632	0.425515	0.512919	0.59
peri1eter_worst	569.0	0.426995	0.133768	0.200677	0.334833	0.388774	0.49
area_worst	569.0	0.207001	0.133840	0.043535	0.121133	0.161378	0.25
s1oothness_worst	569.0	0.594648	0.102572	0.319721	0.523810	0.589847	0.65
co1pactness_worst	569.0	0.240326	0.148711	0.025794	0.139130	0.200284	0.32
concavity_worst	569.0	0.217403	0.166633	0.000000	0.091454	0.181070	0.30
concave points_worst	569.0	0.393836	0.225884	0.000000	0.223127	0.343402	0.55

```
X = df[predictors].values
y = df[target_column].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=40)
print(X_train.shape);
print(X_test.shape)
```

```
    (398, 31)
```

```
     (171, 31)


model = LogisticRegression(random_state=0,solver='lbfgs')
model.fit(X_train,y_train.ravel())

predict_train = model.predict(X_train)
predict_test = model.predict(X_test)


print("Confustion Matrix For Training Data")
print("-------------------------------------")
print(confusion_matrix(y_train,predict_train))
print("-------------------------------------")
print("Accuracy:", accuracy_score(y_train,predict_train))
print("Sensitivity/Recall:",metrics.recall_score(y_train,predict_train))
tn, fp, fn, tp = confusion_matrix(y_train,predict_train).ravel()
specificity = tn / (tn+fp)
print("Specificity:", specificity)
print("Precision:", metrics.precision_score(y_train,predict_train))
print("F-Score:", metrics.f1_score(y_train,predict_train))
print("Mens Squre Error:", mean_squared_error(y_test,predict_test))
print("Root Mens Squre Error:", np.sqrt(mean_squared_error(y_test,predict_test)))
print("ROC_AUC scores:",metrics.roc_auc_score(y_train,predict_train, average="macro"))

# Compute fpr, tpr, thresholds and roc auc
fpr, tpr, thresholds = roc_curve(y_train,predict_train)
roc_auc = auc(fpr,tpr)

# Plot ROC curve
plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')  # random predictions curve
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate or (1 - Specifity)')
plt.ylabel('True Positive Rate or (Sensitivity)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

```
Confustion Matrix For Training Data
-----------------------------------
[[240   2]
 [ 16 140]]
-----------------------------------
Accuracy: 0.9547738693467337
Sensitivity/Recall: 0.8974358974358975
Specificity: 0.9917355371900827
Precision: 0.9859154929577465
F-Score: 0.9395973154362417
```

```
print("Confustion Matrix For Testing Data")
print("-------------------------------------")
print(confusion_matrix(y_test,predict_test))
print("-------------------------------------")
print("Accuracy:", accuracy_score(y_test,predict_test))
print("Sensitivity/Recall:",metrics.recall_score(y_test,predict_test))
tn, fp, fn, tp = confusion_matrix(y_test,predict_test).ravel()
specificity = tn / (tn+fp)
print("Specificity:", specificity)
print("Precision:", metrics.precision_score(y_test,predict_test))
print("F-Score:", metrics.f1_score(y_test,predict_test))
print("Mens Squre Error:", mean_squared_error(y_test,predict_test))
print("Root Mens Squre Error:", np.sqrt(mean_squared_error(y_test,predict_test)))
print("ROC_AUC scores:",metrics.roc_auc_score(y_test,predict_test, average="macro"))

# Compute fpr, tpr, thresholds and roc auc
fpr, tpr, thresholds = roc_curve(y_test,predict_test)
roc_auc = auc(fpr,tpr)

# Plot ROC curve
plt.plot(fpr, tpr, label='ROC curve (area = %0.3f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')  # random predictions curve
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate or (1 - Specifity)')
plt.ylabel('True Positive Rate or (Sensitivity)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

```
Confustion Matrix For Testing Data
-----------------------------------
[[114   1]
 [  2  54]]
-----------------------------------
Accuracy: 0.9824561403508771
Sensitivity/Recall: 0.9642857142857143
Specificity: 0.991304347826087
Precision: 0.9818181818181818
F-Score: 0.972972972972973
```

```
<matplotlib.legend.Legend at 0x7fb1fcdc27d0>
```