BREAST CANCER CLASSIFICATION

GROUP 7:

ASTHA TIBREWAL
HARSHIT GUPTA
PAYAL DANGI
KAUSTAV DASGUPTA

OBJECTIVE

We aimed at building a breast cancer classification model that could accurately and efficiently detect breast cancer. A bigger motivation was to see how healthcare professionals make informed decisions for diagnosis and treatment, leading to improved patient outcomes and reduced mortality rates.
The goal was to create a model that can accurately distinguish between breast tumors based on input features extracted from medical imaging data, such as ultrasound scans.

DATASET USED

http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29

This has 569 training examples on which we trained our model.

OUR METHOD

First of all we imported the dataset from the SKlearn databases and converted the data into a dataframe. After that we cleaned the data by checking for any NULL entries and scaling the values using MinMaxScaler. Lastly we splitted the data into training and testing data and trained a logistic regression model based on the training data and tested the trained model on the testing data.
Finally we got an accuracy of 98% which shows that the model was quite accurate.

TECHNIQUES IMPLEMENTED

Firstly we used KNN model to test the data, the KNN model gave us an accuracy of 96%. So in order to increase the accuracy we tried to use different models.
SVM model gave an accuracy of 97%
And the logistic regression model gave an overall accuracy of 98%,
as the logistic model had the greatest accuracy we went with the logistic regression model.

CODE

In [25]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve,auc
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import matthews_corrcoef
import scikitplot as skplt
```

In [26]:

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()  # embeded dataset
df = pd.DataFrame(np.c_[cancer['target'], cancer['data']],
                  columns= np.append(['MB'], cancer['feature_names']))
df
```

Out[26]:

	MB	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	w compact
0	0.0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	...	25.380	17.33	184.60	2019.0	0.16220	0.6
1	0.0	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	...	24.990	23.41	158.80	1956.0	0.12380	0.1
2	0.0	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	...	23.570	25.53	152.50	1709.0	0.14440	0.4
3	0.0	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	...	14.910	26.50	98.87	567.7	0.20980	0.8
4	0.0	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	...	22.540	16.67	152.20	1575.0	0.13740	0.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	0.0	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	...	25.450	26.40	166.10	2027.0	0.14100	0.2
565	0.0	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	...	23.690	38.25	155.00	1731.0	0.11660	0.1
566	0.0	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	...	18.980	34.12	126.70	1124.0	0.11390	0.3

In [27]:

```python
df.info()
# Checking for null entries if present
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   MB                       569 non-null    float64
 1   mean radius              569 non-null    float64
 2   mean texture             569 non-null    float64
 3   mean perimeter           569 non-null    float64
 4   mean area                569 non-null    float64
 5   mean smoothness          569 non-null    float64
 6   mean compactness         569 non-null    float64
 7   mean concavity           569 non-null    float64
 8   mean concave points      569 non-null    float64
 9   mean symmetry            569 non-null    float64
 10  mean fractal dimension   569 non-null    float64
 11  radius error             569 non-null    float64
 12  texture error            569 non-null    float64
 13  perimeter error          569 non-null    float64
 14  area error               569 non-null    float64
 15  smoothness error         569 non-null    float64
 16  compactness error        569 non-null    float64
 17  concavity error          569 non-null    float64
 18  concave points error     569 non-null    float64
 19  symmetry error           569 non-null    float64
 20  fractal dimension error  569 non-null    float64
 21  worst radius             569 non-null    float64
 22  worst texture            569 non-null    float64
 23  worst perimeter          569 non-null    float64
 24  worst area               569 non-null    float64
 25  worst smoothness         569 non-null    float64
 26  worst compactness        569 non-null    float64
 27  worst concavity          569 non-null    float64
 28  worst concave points     569 non-null    float64
 29  worst symmetry           569 non-null    float64
 30  worst fractal dimension  569 non-null    float64
dtypes: float64(31)
memory usage: 137.9 KB
```

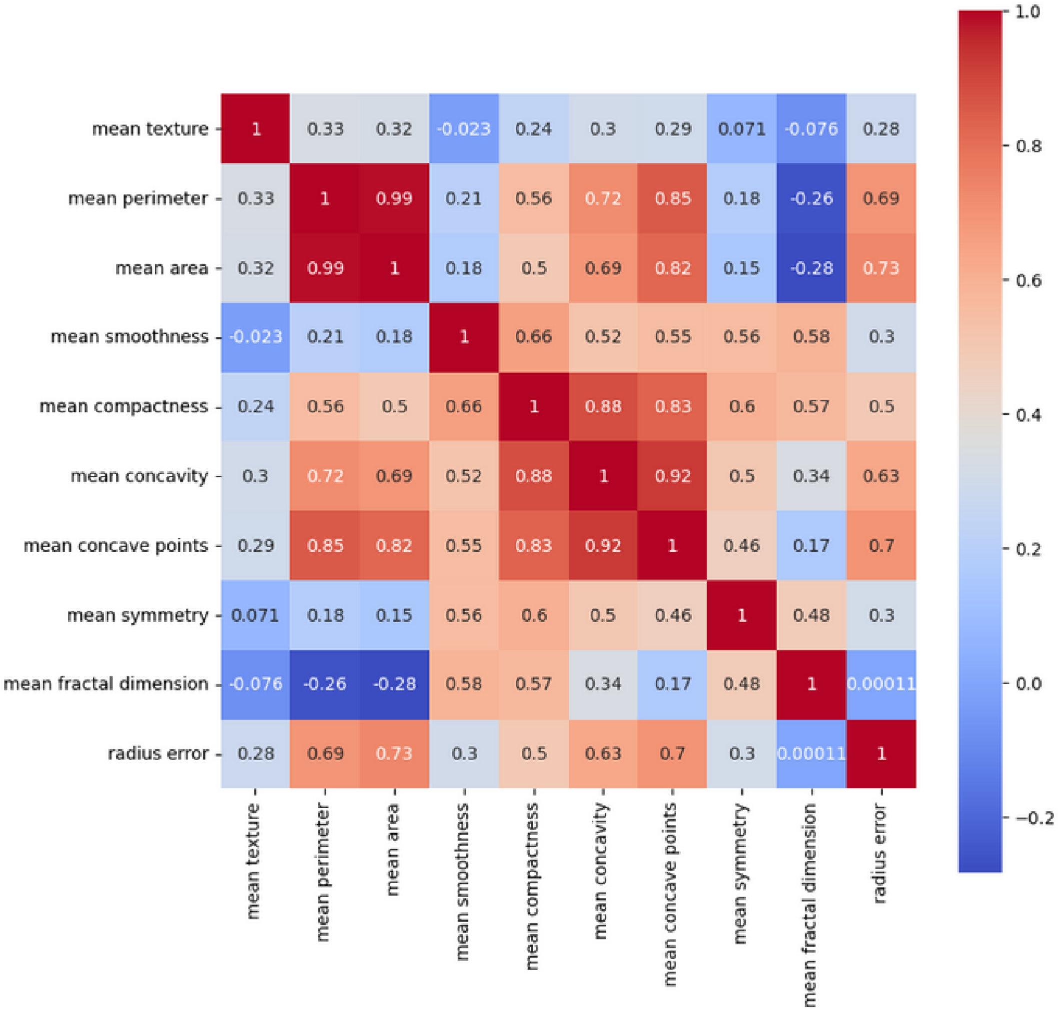
In [28]:

```
df.describe()
```

Out[28]:

	MB	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	...	worst radius	worst texture	peri
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.0
mean	0.627417	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...	16.269190	25.677223	107.2
std	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...	4.833242	6.146258	33.6
min	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...	7.930000	12.020000	50.4
25%	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...	13.010000	21.080000	84.1
50%	1.000000	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...	14.970000	25.410000	97.6
75%	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...	18.790000	29.720000	125.4
max	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...	36.040000	49.540000	251.2

8 rows × 31 columns



mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

In [30]:
```
X = df.drop('MB', axis = 1)
y = df['MB']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [31]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [32]:
```
model_accuracy = {}
```

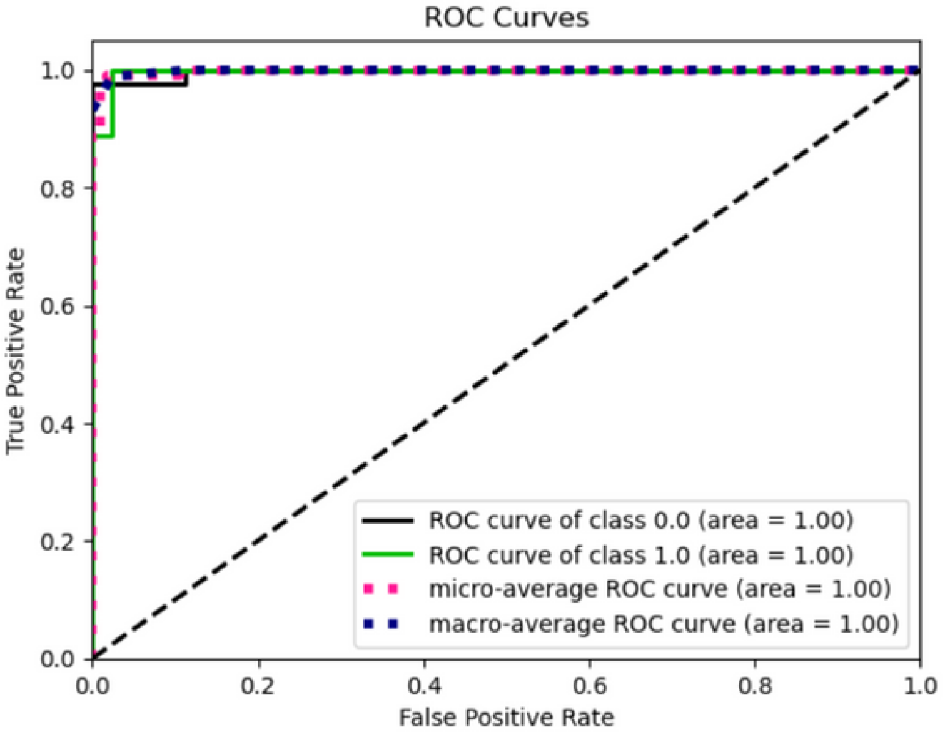
In [33]:
```
from sklearn.linear_model import LogisticRegression
```

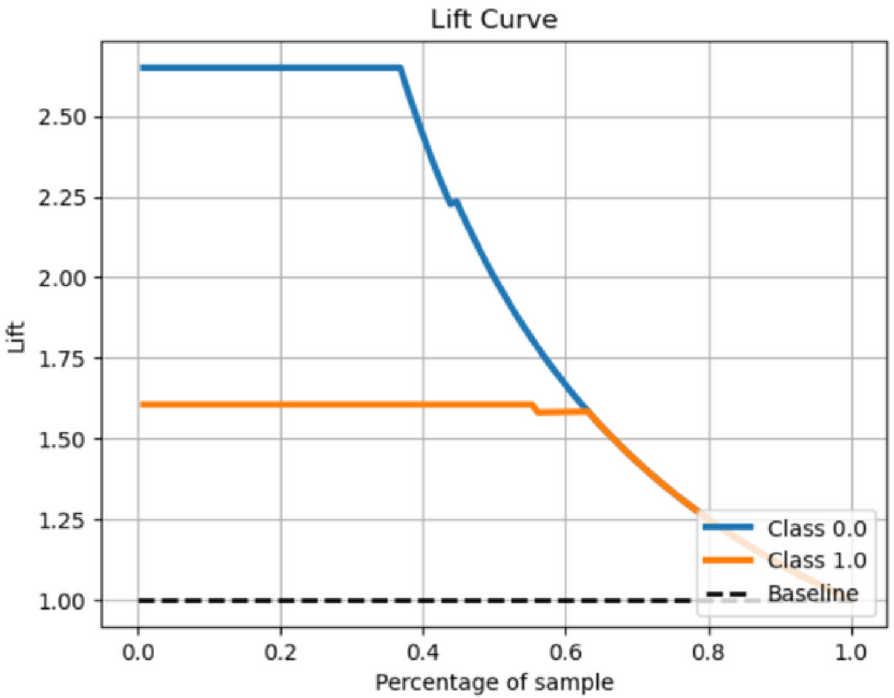
In [34]:
```
a = list(range(-5, 10))
complexity_values = [10**i for i in a]
#xticks = list(range(1, len(complexity_values)+1))
param_values = {'C':complexity_values, 'penalty': ['l1', 'l2', 'elasticnet', 'none'],'solver':['liblinear','newton-cg']}

clf = LogisticRegression()
m_lr = GridSearchCV(clf, param_grid=param_values, cv = 3, scoring = 'accuracy', return_train_score=True, n_jobs=-1)
m_lr.fit(X_train ,y_train)

y_pred = m_lr.predict(X_test)

model_accuracy['Logistic Regression'] = accuracy_score(y_test, y_pred)

print('\n')
print('Prediction Accuracy: ', accuracy_score(y_test, y_pred))
print('\n')
print('confusion matrix: ')
print(confusion_matrix(y_test, y_pred))
print('\n')
print('classification report: ')
print(classification_report(y_test, y_pred))
```

```python
fpr, tpr, threshold = roc_curve(y_test, y_pred)
auc = metrics.auc(fpr, tpr)
print("The AUC stats is: ", auc)
print("The kappa stats is: ", cohen_kappa_score(y_test, y_prec
print("The MCC stats is: ", matthews_corrcoef(y_test, y_pred))

# ROC curve
predicted_probas_lr = m_lr.predict_proba(X_test)
skplt.metrics.plot_roc(y_test, predicted_probas_lr)

# Lift curve
skplt.metrics.plot_lift_curve(y_test, predicted_probas_lr)
plt.show()
```

```
best score:  0.9647960962007668
best parameters:  {'C': 1000, 'penalty': 'l2', 'solver': 'newton-cg'}
best estimator:  LogisticRegression(C=1000, solver='newton-cg')


Prediction Accuracy:  0.9824561403508771


confusion matrix:
[[42  1]
 [ 1 70]]


classification report:
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98        43
         1.0       0.99      0.99      0.99        71

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114

The AUC stats is:  0.9813298395021289
The kappa stats is:  0.9626596790042581
The MCC stats is:  0.9626596790042581
```

```
The AUC stats is:  0.9813298395021289
The kappa stats is:  0.9626596790042581
The MCC stats is:  0.9626596790042581
```






```
[35]: model_accuracy
```

```
[35]: {'Logistic Regression': 0.9824561403508771}
```

RESULTS

We used regression to make the model and obtained an accuracy of 98%. The model was able to detect the type of breast cancer with 98% precision.

We also implemented the model using KNN and SVM. The accuracy we obtained for the KNN model was 96%.

CONCLUSION

This model aimed to develop an accurate and efficient system for distinguishing between benign and malignant breast tumors using machine learning techniques.

Different machine learning algorithms, such as logistic regression, support vector machines and KNNs were implemented and analyzed to give the best possible result. Our model was made by using regression techniques and further optimized to achieve the best possible performance.