

# ECGR\_4105\_HW3\_Problem\_3

October 17, 2024

```
[1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
[2]: file_url = 'https://raw.githubusercontent.com/lnguye782/ECGR-4105-Intro-to-ML/
↳refs/heads/main/HW3/cancer.csv'
data = pd.read_csv(file_url)

data.head()
```

```
[2]:
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

  

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	
1	0.07864	0.0869	0.07017	0.1812	
2	0.15990	0.1974	0.12790	0.2069	
3	0.28390	0.2414	0.10520	0.2597	
4	0.13280	0.1980	0.10430	0.1809	

  

	mean fractal dimension	...	worst texture	worst perimeter	worst area	\
0	0.07871	...	17.33	184.60	2019.0	
1	0.05667	...	23.41	158.80	1956.0	
2	0.05999	...	25.53	152.50	1709.0	
3	0.09744	...	26.50	98.87	567.7	
4	0.05883	...	16.67	152.20	1575.0	

  

	worst smoothness	worst compactness	worst concavity	worst concave points	\
0	0.1622	0.6656	0.7119	0.2654	
1	0.1238	0.1866	0.2416	0.1860	
2	0.1444	0.4245	0.4504	0.2430	
3	0.2098	0.8663	0.6869	0.2575	
4	0.1374	0.2050	0.4000	0.1625	

	worst symmetry	worst fractal dimension	target
0	0.4601	0.11890	0
1	0.2750	0.08902	0
2	0.3613	0.08758	0
3	0.6638	0.17300	0
4	0.2364	0.07678	0

[5 rows x 31 columns]

```
[3]: # Separate features and target variable (30 input features / 1 output target)
X = data.drop(columns=['target'])
Y = data['target']
```

```
[4]: # Split the data set into Training Data (80%) and Test Data (20%)
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)
```

```
[5]: # Scale the data between 0 and 1 to get better accuracy
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

```
[6]: # Fit a Naive Bayes model to the data
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
model.fit(X_train, Y_train)
```

```
[6]: GaussianNB()
```

```
[10]: # Predict on the test data
Y_pred = model.predict(X_test)
```

```
[12]: # Evaluate the model with L2 regularization using model evaluation metrics:↳
↳accuracy, precision, recall, and F1 score
from sklearn import metrics

print("Accuracy with Naive Bayes:",metrics.accuracy_score(Y_test, Y_pred))
print("Precision with Naive Bayes:",metrics.precision_score(Y_test, Y_pred))
print("Recall with Naive Bayes:",metrics.recall_score(Y_test, Y_pred))
print("F1 Score with Naive Bayes:",metrics.f1_score(Y_test, Y_pred))
```

Accuracy with Naive Bayes: 0.9649122807017544  
Precision with Naive Bayes: 0.958904109589041  
Recall with Naive Bayes: 0.9859154929577465  
F1 Score with Naive Bayes: 0.9722222222222222