**Breast Cancer Wisconsin Data Analysis**

**Mike Sadowski**

**Assumptions**

Some key assumptions were made at the time of the analysis of the data set. The first major assumption was that the data set being used and analyzed was located at <https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/>, with the title of the file being “breast-cancer-wisconsin.data.”

Located inside of the file titled “breast-cancer-wisconsin.names” there was some information about the data. It described that the data had missing/unavailable entries and upon inspection of the data in the preprocessing phase of analysis, it was determined that 16 entries were missing. The assumption here was that the entire row of data was deemed unusable and was not included in the analysis (the 16 rows were removed).

The third and final assumption was also done during the data preprocessing phase, and the assumption made was that the “ID” feature was deemed unimportant to the analysis. The entire column was dropped from the data set and not included in the analysis.

**Conclusions**

After completing the analysis of the data set, it was apparent that of the 2 models (K-Nearest Neighbours and Logistic Regression) the kNN model outperformed the Logistic Regression model. The comparison was done by calculating accuracy scores for each model, generating a confusion matrix for each model, and calculating the mean squared error and the coefficient of determination for each model.

Performing hyperparameter optimization helped improve the performance of both the kNN and Logistic Regression models by about 3% or more depending on the metric. Interestingly, without the optimization of the hyperparameters the Logistic Regression model outperformed the kNN model. But with the optimization, the kNN model performed the best.

It was observed that with smaller sample sizes of the data set, the Logistic Regression model performed much worse than the kNN model. As sample size increased, so did the Logistic Regression model’s performance. When using the entire data set, the models are very close in performance. It was concluded that with a much larger sample size the Logistic Regression model would perform better and may even beat the kNN. This was done by selecting a smaller sample out of the data set, and gradually increasing the sample size and observing the accuracy of the models.

The kNN model performs well and is accurate, however this is highly dependent on the value of k given to the model. Lower values of k give a worse performance while increasing the value improves the performance.

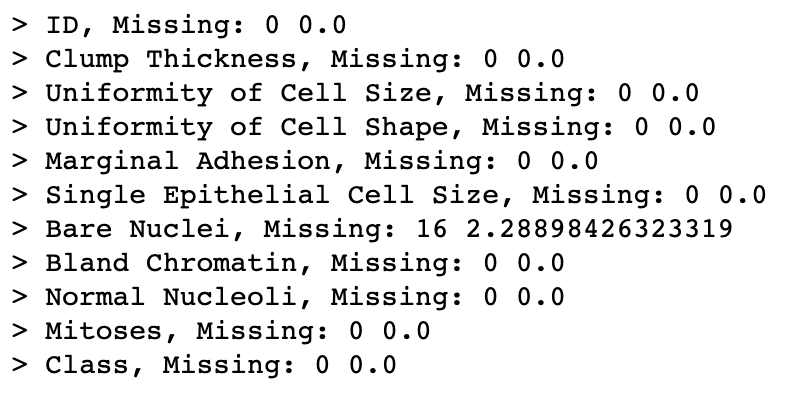
**Analysis**

1. **Data Preprocessing**

In this phase, data cleaning and data inspection was done to ensure that an accurate data set was provided when training the 2 models (kNN and Logistic Regression).

The data set was inspected by first examining the “breast-cancer-wisconsin.names” file. This file contains information about where the data originated from, who developed it, past usages, and most importantly information about the format of the data set. The data set contains 11 features: 'ID', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape', 'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin', 'Normal Nucleoli', 'Mitoses', and 'Class'.

After inspecting the data set, it was discovered that 16 rows were incomplete with missing values for the ‘Bare Nuclei’ feature, denoted as a ‘?’ character in the data set instead of a numerical value. These rows were corrected by loading the data set, changing the ‘?’ characters to a NaN value, then searching for the rows with NaN values and removing the row from the data set. The figure below shows the output of the features which are missing rows:



The ‘ID’ column provides no useful information. We do not need to identify any individual row for this analysis, we are only interested in providing the other features to the models and having the models classify whether a tumor is benign or malignant. As a result, the ‘ID’ column was dropped from the data set.

1. **Model Selection**

K-Nearest Neighbours was the first model selected and trained. The reason this model was chosen is because it is a supervised learning algorithm that is used for classification problems. In this case, we wanted to classify the tumors as benign or malignant based on several features. This model can identify patterns in the data, and group similar data points together based on their proximity to each other.

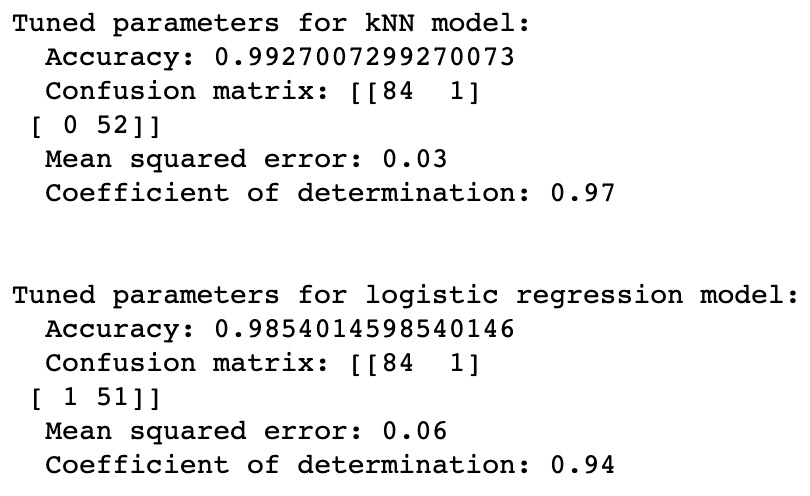
The kNN model was trained by fitting it with the training data (data was split 80/20 for training/test) from the data set. Next, the optimal hyperparameters were found by using 5-fold cross validation. The list of most optimal hyperparameters was found to be 'weights' = 'distance', 'n\_neighbors' = 6, 'leaf\_size' = 5 for this model.

Logistic Regression was the second model selected and trained. The reason this model was chosen is because it is used to estimate the relationship between a dependent variable and several independent variables. The result is the ability to predict the outcome of a dependent variable. In this case, we wanted to predict whether the tumors are benign or malignant based on several features.

The Logistic Regression model was trained by fitting it with the training data (data was split 80/20 for training/test) from the data set. Next, the optimal hyperparameters were found by using 5-fold cross validation. The list of most optimal hyperparameters was found to be 'solver' = 'lbfgs', 'penalty' = 'none', 'dual' = False for this model.

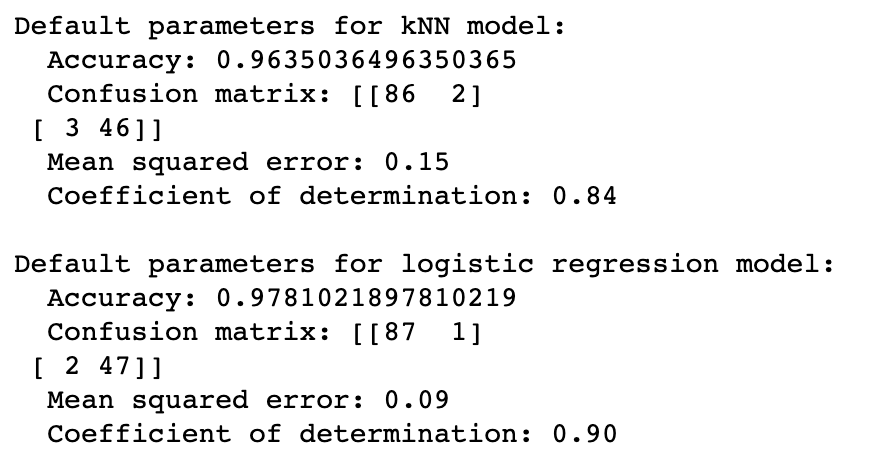
1. **Model Evaluation**

The two models were evaluated using the metrics accuracy scores, confusion matrices, mean squared error and coefficient of determination. The results are as seen below:



As can be seen from the figure above, the kNN model slightly outperforms the Logistic Regression model in every metric. The results are remarkably close.

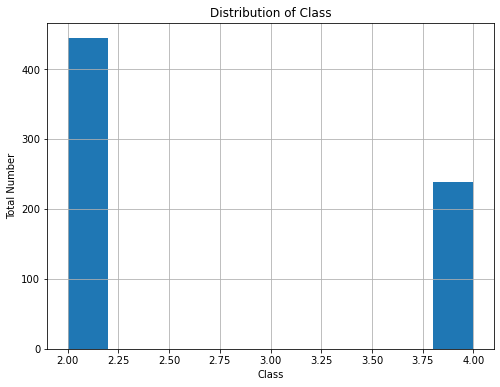
Performing hyperparameter optimization helped improve the performance of these models by about 3% accuracy. The performance of each model without hyperparameter optimization can be seen below:



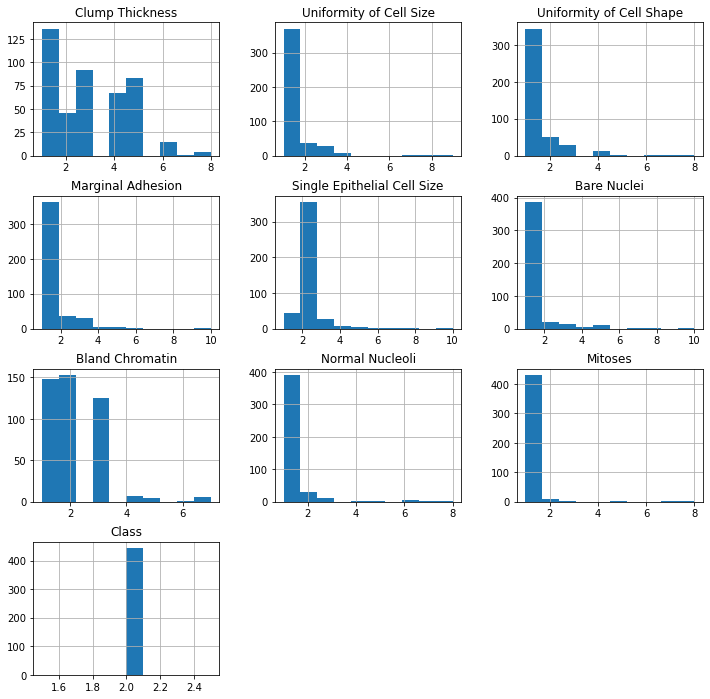
It is interesting to note that the Logistic Regression model outperforms the kNN model without optimization of the hyperparameters.

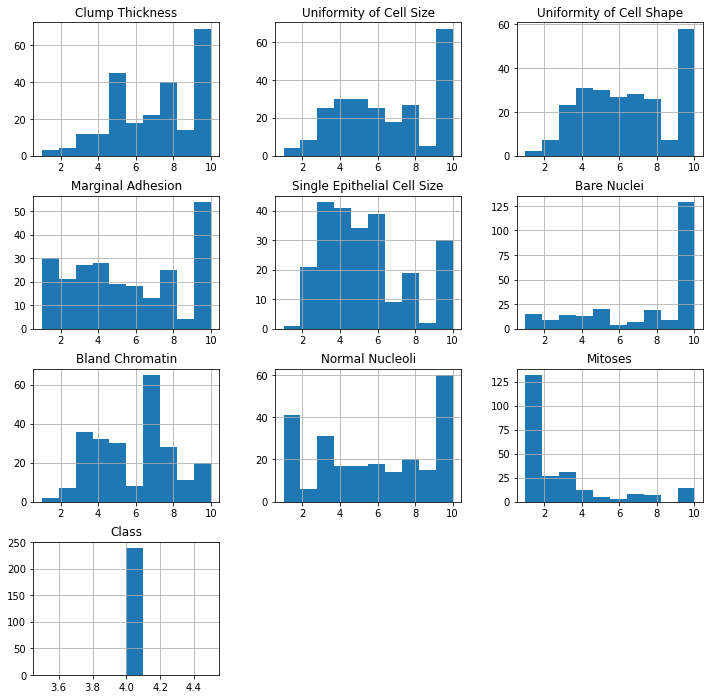
1. **Data Visualization**

Distribution of ‘Class’ feature:



Distribution of each feature vs ‘Class’:





**Code:**

# 1. download the dataset, clean the data, and do a stratified random split for generating the training/test sets.

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

feature\_names=['ID','Clump Thickness','Uniformity of Cell Size', 'Uniformity of Cell Shape', 'Marginal Adhesion', 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin', 'Normal Nucleoli', 'Mitoses', 'Class']

# used https://stackoverflow.com/questions/64957335/replace-question-mark-in-pandas-dataframe for na\_values='?'

df = pd.read\_csv('~/Desktop/git/Wilfrid-Laurier/CP610/project/data/breast-cancer-wisconsin.data', sep=',', names=feature\_names, engine='python', encoding='latin-1', na\_values='?', header=None)

# Took this function from Assignment #3 lesson/notes

def get\_missing\_counts(df: pd.DataFrame):

for i in df.columns:

# count number of rows with missing values

# get the count of missing values for the i column. isnull() set all NaN values to be True and others are False.

# and sum() counts the number of True values in this column

n\_miss = df[i].isnull().sum()

# df.shape[0] is the count of all values in a column, including NaN values.

# So we can get missing percentage of each column via the code below

perc = n\_miss / df.shape[0] \* 100

print(f'> {i}, Missing: {n\_miss} {perc}')

get\_missing\_counts(df)

# > ID, Missing: 0 0.0

# > Clump Thickness, Missing: 0 0.0

# > Uniformity of Cell Size, Missing: 0 0.0

# > Uniformity of Cell Shape, Missing: 0 0.0

# > Marginal Adhesion, Missing: 0 0.0

# > Single Epithelial Cell Size, Missing: 0 0.0

# > Bare Nuclei, Missing: 16 2.28898426323319

# > Bland Chromatin, Missing: 0 0.0

# > Normal Nucleoli, Missing: 0 0.0

# > Mitoses, Missing: 0 0.0

# > Class, Missing: 0 0.0

# There are 16 NaN values in the set (in the 'Bare Nuclei' column), remove these rows

# https://stackoverflow.com/questions/13413590/how-to-drop-rows-of-pandas-dataframe-whose-value-in-a-certain-column-is-nan

df = df[df['Bare Nuclei'].notna()]

# Cleaned data frame:

# > ID, Missing: 0 0.0

# > Clump Thickness, Missing: 0 0.0

# > Uniformity of Cell Size, Missing: 0 0.0

# > Uniformity of Cell Shape, Missing: 0 0.0

# > Marginal Adhesion, Missing: 0 0.0

# > Single Epithelial Cell Size, Missing: 0 0.0

# > Bare Nuclei, Missing: 0 0.0

# > Bland Chromatin, Missing: 0 0.0

# > Normal Nucleoli, Missing: 0 0.0

# > Mitoses, Missing: 0 0.0

# > Class, Missing: 0 0.0

# Drop unneeded columns:

# Drop 'ID' column, it is not needed for analysis as it doesn't provide any usefil insight

# Took this code from Assignment #2

drop\_columns = ['ID']

df.drop(drop\_columns, axis=1, inplace=True)

# 683 rows, 10 columns

df.shape

df.head(5)

# Took this code from Assignment #5

data = df.drop(['Class'], axis=1).to\_numpy()

target = df['Class'].to\_numpy()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(data, target, test\_size = 0.2)

# 2. apply at least 2 models to the dataset and use 5-fold cross-validation to find the optimal hyperparameters for each

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.linear\_model import LogisticRegression

# Model 1 - K-Nearest Neighbour

# Took this code from Assignment #9

knn\_model = KNeighborsClassifier()

knn\_model.fit(x\_train, y\_train)

random\_grid\_knn = {

'n\_neighbors': [2,3,4,5,6],

'weights': ['uniform','distance'],

'leaf\_size': [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],

}

knn\_model\_tuned = RandomizedSearchCV(estimator = knn\_model, param\_distributions = random\_grid\_knn, n\_iter = 10, cv = 5, n\_jobs = -1)

knn\_model\_tuned.fit(x\_train, y\_train)

# Model 2 - Logistic Regression

lr\_model = LogisticRegression(random\_state = 0)

lr\_model.fit(x\_train, y\_train)

random\_grid\_lr = {

'solver': ['newton-cg', 'lbfgs', 'liblinear'],

'penalty': ['l1', 'l2', 'elasticnet', 'none'],

'dual': [True, False]

}

lr\_model\_tuned = RandomizedSearchCV(estimator = lr\_model, param\_distributions = random\_grid\_lr, n\_iter = 10, cv = 5, n\_jobs = -1)

lr\_model\_tuned.fit(x\_train, y\_train)

# 3. apply the fine-tuned models on the test set and evaluate them on the test set

# Took this code from Assignment #5, #9

from sklearn.metrics import confusion\_matrix, accuracy\_score

# Model 1 - K-Nearest Neighbour

tuned\_pred\_knn = knn\_model\_tuned.best\_estimator\_.predict(x\_test)

print('Tuned parameters for kNN model:')

print(' Accuracy:', accuracy\_score(y\_test, tuned\_pred\_knn))

print(' Confusion matrix:', confusion\_matrix(y\_test, tuned\_pred\_knn))

print(" Mean squared error: %.2f" % mean\_squared\_error(y\_test, tuned\_pred\_knn))

print(" Coefficient of determination: %.2f" % r2\_score(y\_test, tuned\_pred\_knn))

print()

print('Best tuned kNN model info:')

print(' Score:', knn\_model\_tuned.best\_score\_)

print(' Estimator:', knn\_model\_tuned.best\_estimator\_)

print(' Params: ', knn\_model\_tuned.best\_params\_)

print()

# Model 2 - Logistic Regressions

tuned\_pred\_lr = lr\_model\_tuned.best\_estimator\_.predict(x\_test)

print('Tuned parameters for logistic regression model:')

print(' Accuracy:', accuracy\_score(y\_test, tuned\_pred\_lr))

print(' Confusion matrix:', confusion\_matrix(y\_test, tuned\_pred\_lr))

print(" Mean squared error: %.2f" % mean\_squared\_error(y\_test, tuned\_pred\_lr))

print(" Coefficient of determination: %.2f" % r2\_score(y\_test, tuned\_pred\_lr))

print()

print('Best tuned linear regression model info:')

print(' Score:', lr\_model\_tuned.best\_score\_)

print(' Estimator:', lr\_model\_tuned.best\_estimator\_)

print(' Params: ', lr\_model\_tuned.best\_params\_)

# Data Visualization

import matplotlib.pyplot as plt

figure\_size = (8, 6)

plt.figure(figsize = figure\_size)

plt.title('Distribution of Class')

df.Class.hist()

plt.xlabel('Class')

plt.ylabel('Total Number')

plt.show()

# https://www.codegrepper.com/code-examples/python/making+a+histogram+with+a+group+by+pandas

# https://stackoverflow.com/questions/19584029/plotting-histograms-from-grouped-data-in-a-pandas-dataframe

df.groupby('Class').hist(figsize=(12, 12))