Gabriel Van Dreel
800886655
October 23, 2021
ECGR 4105-C01

Homework 2

The source code developed for homework 2 was uploaded to the Github repository at the following link:

https://github.com/Gabrielvd616/ECGR4105/tree/main/Homework2

Problem 1

- The gradient descent training and cost evaluation for the logistic regression binary classifier predicting whether an individual had diabetes was developed with consideration of the Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age input variables. Using an 80% and 20% split of the labelled data between training and evaluation (test) sets, the model was trained and evaluated with both normalization (MIN MAX scaling) and standardization of the input variables. The logistic regression model was evaluated using the test data to obtain the confusion matrix as shown in section [1]. The accuracy, precision, and recall of the model were calculated and recorded in section [2]. The confusion matrix for the model was also plotted using a heat map as shown in section [3].

  Let class 0 be the classification that an instance of an individual does not have the diabetes outcome and class 1 be the classification that an instance of an individual does have the diabetes outcome.

Problem 2

- The analysis of the training data and cost evaluation for the Gaussian naïve Bayes binary classifier predicting whether an individual had diabetes was developed with consideration of the Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, and Age input variables. Using an 80% and 20% split of the labelled data between training and evaluation (test) sets, the model was trained and evaluated with both MIN MAX scaling and standardization of the input variables. The naïve Bayes model was evaluated using the test data to obtain the confusion matrix as shown in section [4]. The accuracy, precision, and recall of the model were calculated and recorded in section [5]. The confusion matrix for the model was also plotted using a heat map as shown in section [6].

  The accuracy and precision of the logistic regression model was observed to be higher than that of the naïve Bayes model while the recall of the two models was observed to be equal. The higher accuracy and precision of the logistic regression model is reflected in entry $(1,1)$ of the confusion matrices, which gives the number of correct predictions of the test data for membership to class 0 where it is 97 for the logistic regression model and 94 for the naïve

Bayes model. The performance of the logistic regression model is likely higher because it is a discriminative model that strictly analyzes trends in training data as opposed to the naïve Bayes model being a generative one that holistically analyzes the model features and evidently tends to be overfit as a result. For these reasons, the naïve Bayes model was observed to have lower performance and therefore be less generalizable than the logistic regression model.

Problem 3

- The gradient descent training and cost evaluation for the logistic regression binary classifier predicting whether an individual had diabetes was repeated with K-fold cross validation of the model using the labelled data. The logistic regression model was trained and evaluated for $K = 5$ and $K = 10$ folds of the labelled data with iterative evaluation of the model on each fold. The accuracy, precision, and recall of the model were calculated and recorded for each fold of $K = 5$ in section [7] and $K = 10$ in section [8].

  The accuracies of the logistic regression models trained and evaluated on each of the folds for $K = 5$ and $K = 10$ were marginally less than that of the logistic regression model trained and evaluated on the original 80% and 20% split of the labelled data. The precision of the model trained and evaluated on each of the folds for $K = 5$ and $K = 10$ was approximately equal to that of the model trained and evaluated on the original 80% and 20% split. Although it had the highest variation, the recall of the model trained and evaluated on each of the folds for $K = 5$ and $K = 10$ was approximately equal to that of the model trained and evaluated on the original 80% and 20% split. The variations in accuracy, precision, and recall may have resulted from the randomized ordering of the labelled data but, given that the evaluation metrics were consistent across all folds for $K = 5$ and $K = 10$, the K-fold cross validation was observed to support the evaluation metrics of the logistic regression model trained and evaluated on the original 80% and 20% split. As a result, the observed performance of the logistic regression model is probably accurate for prediction of a more generalized or unlabeled data set.

Problem 4

- The gradient descent training and cost evaluation for the Gaussian naïve Bayes binary classifier predicting whether an individual had diabetes was repeated with K-fold cross validation of the model using the labelled data. The naïve Bayes model was set to analyze the model features depicted in the training set and evaluated for each fold corresponding to $K = 5$ and $K = 10$ folds of the labelled data. The accuracy, precision, and recall of the model were calculated and recorded for each fold of $K = 5$ in section [9] and $K = 10$ in section [10].

  The accuracies, precisions, and recalls of the naïve Bayes models set to analyze the training model features and evaluated on each of the folds for $K = 5$ and $K = 10$ varied more widely than those of the logistic regression models trained and evaluated on each of the folds with values ranging from 0.458 to 0.833. The accuracy, precision, and recall of the naïve Bayes model set to analyze and be evaluated on the original 80% and 20% split of the labelled data was observed to be higher than the majority of the metric values obtained for the cross-validated naïve Bayes models. While the high variation and skew of evaluation metrics could partly be the

result of randomized ordering of labelled data, the metrics of the cross-validation primarily suggest that the original naïve Bayes model does not perform consistently across the K test sets and therefore would likely not perform well on a more generalized or unlabeled data set. Overall, the performance of the model was observed to reflect the generative nature of naïve Bayes classifiers that renders them unable to be effectively validated using K-fold cross validation, which likely results in part from their lack of training to analyze labelled data points.

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split, KFold, cross_validate
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Problem 1
# Reads labelled training data
df = pd.read_csv(r'diabetes.csv')

x = df.values[:, :7]
y = df.values[:, 8]

# Performs MIN MAX scaling
mms = MinMaxScaler()
x = mms.fit_transform(x)

# Performs standardization
ss = StandardScaler()
x = ss.fit_transform(x)

# Performs 80% and 20% split of the labelled data into training and test sets
np.random.seed(0)
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8,
                                                    test_size=0.2,
                                                    random_state=np.random)

# Performs logistic regression by instantiating LogisticRegression object
lr = LogisticRegression()
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

# Generates confusion matrix to evaluate accuracy of model
cnf_matrix = confusion_matrix(y_test, y_pred)
print(cnf_matrix)
```

```
[[97 10]
 [19 28]]
```

In [2]:

```
# Evaluates model using accuracy, precision, and recall evaluation metrics
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: 0.8116883116883117
```

```
Precision: 0.7368421052631579
Recall: 0.5957446808510638
```

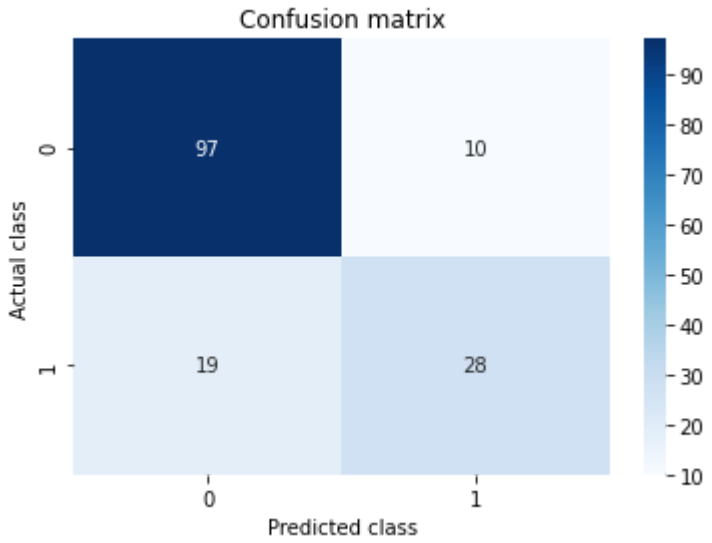
In [3]:

```
# Plots the results of the binary classifier model
class_names = [0, 1]
tick_marks = np.arange(len(class_names))
plt.subplots()
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

# Generates heat map
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap='Blues', fmt='g')
plt.title('Confusion matrix')
plt.xlabel('Predicted class')
plt.ylabel('Actual class')
```

Out[3]:

```
Text(33.0, 0.5, 'Actual class')
```



In [4]:

```
# Problem 2
# Generates Gaussian naive bayes model by instantiating GaussianNB object
gnb = GaussianNB()
gnb.fit(x_train, y_train)
y_pred = gnb.predict(x_test)

# Generates confusion matrix to evaluate accuracy of model
cnf_matrix = confusion_matrix(y_test, y_pred)
print(cnf_matrix)
```

```
[[94 13]
 [19 28]]
```

In [5]:

```
# Evaluates model using accuracy, precision, and recall evaluation metrics
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))
```

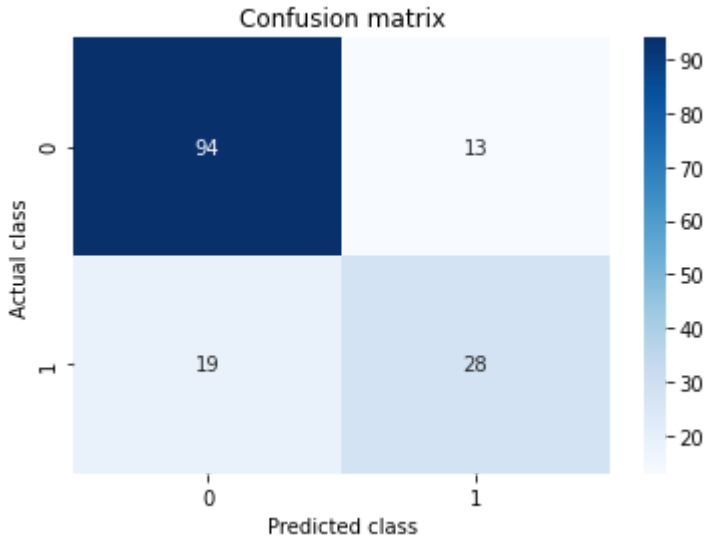
```
Accuracy: 0.7922077922077922
Precision: 0.6829268292682927
Recall: 0.5957446808510638
```

In [6]:

```
# Plots the results of the binary classifier model
tick_marks = np.arange(len(class_names))
plt.subplots()
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

# Generates heat map
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap='Blues', fmt='g')
plt.title('Confusion matrix')
plt.xlabel('Predicted class')
plt.ylabel('Actual class')
```

Out[6]:
```
Text(33.0, 0.5, 'Actual class')
```



In [7]:
```
# Problem 3
# Performs K-fold cross-validation of logistic regression for K = 5 folds
metrics = ['accuracy', 'precision', 'recall']
cv5 = KFold(n_splits=5, random_state=1, shuffle=True)
scores = cross_validate(lr, x, y, scoring=metrics, cv=cv5, n_jobs=-1)
print('Accuracy:', scores['test_accuracy'])
print('Precision:', scores['test_precision'])
print('Recall:', scores['test_recall'])
```

```
Accuracy: [0.77922078 0.74675325 0.77922078 0.79738562 0.73856209]
Precision: [0.75609756 0.68292683 0.72093023 0.74358974 0.69047619]
Recall: [0.56363636 0.51851852 0.58490566 0.58       0.51785714]
```

In [8]:
```
# Performs K-fold cross-validation of logistic regression for K = 10 folds
cv10 = KFold(n_splits=10, random_state=1, shuffle=True)
scores = cross_validate(lr, x, y, scoring=metrics, cv=cv10, n_jobs=-1)
print('Accuracy:', scores['test_accuracy'])
print('Precision:', scores['test_precision'])
print('Recall:', scores['test_recall'])
```

```
Accuracy: [0.79220779 0.76623377 0.79220779 0.7012987  0.80519481 0.71428571
 0.85714286 0.75324675 0.72368421 0.76315789]
Precision: [0.7826087  0.72222222 0.81818182 0.52631579 0.81818182 0.54545455
 0.8        0.7        0.63157895 0.75       ]
Recall: [0.62068966 0.5        0.6        0.41666667 0.62068966 0.5
 0.69565217 0.51851852 0.46153846 0.6        ]
```

In [9]:
```
# Problem 4
# Performs K-fold cross-validation of Gaussian naive bayes model for K = 5 folds
scores = cross_validate(gnb, x, y, scoring=metrics, cv=cv5, n_jobs=-1)
print('Accuracy:', scores['test_accuracy'])
print('Precision:', scores['test_precision'])
print('Recall:', scores['test_recall'])
```

```
Accuracy: [0.79220779 0.73376623 0.73376623 0.78431373 0.75163399]
Precision: [0.75555556 0.63829787 0.63043478 0.68085106 0.70454545]
Recall: [0.61818182 0.55555556 0.54716981 0.64       0.55357143]
```

In [10]:
```
# Performs K-fold cross-validation of Gaussian naive bayes model for K = 10 folds
scores = cross_validate(gnb, x, y, scoring=metrics, cv=cv10, n_jobs=-1)
print('Accuracy:', scores['test_accuracy'])
print('Precision:', scores['test_precision'])
print('Recall:', scores['test_recall'])
```

```
Accuracy: [0.83116883 0.75324675 0.81818182 0.67532468 0.80519481 0.68831169
 0.81818182 0.77922078 0.76315789 0.72368421]
Precision: [0.83333333 0.65217391 0.80769231 0.47826087 0.79166667 0.5
 0.69565217 0.72727273 0.68181818 0.69565217]
Recall: [0.68965517 0.57692308 0.7        0.45833333 0.65517241 0.45833333
 0.69565217 0.59259259 0.57692308 0.53333333]
```