514-Assignment-2

Muhammad Somaan

April 2024

1 Preprocessing

First of all, I encoded all the categorical data to 0 and 1. This included the Gender column and the Lung_Cancer column. After this, I checked for any missing data and then I checked the maximum and minimum values of all the columns. To improve the efficiency of my model, I normalized each column to between 0 and 1.

1.1 Encoding Data

1.1.1 Gender

Value of 1 was assigned to M for male and value of 0 was assigned to F for female.

```
# Converting the categorical data to numerical data.
le = preprocessing.LabelEncoder()
# 1 is Male, 0 is Female
data['GENDER'] = le.fit_transform(data['GENDER'])
```

Figure 1: Gender Encoding

1.1.2 Lung_Cancer

Value of 1 was assigned to Yes and value of 0 was assigned to No.

```
# Converting the categorical data to numerical data.
le = preprocessing.LabelEncoder()
# 1 is Yes, 0 is No
data['LUNG_CANCER'] = le.fit_transform(data['LUNG_CANCER'])
```

Figure 2: Lung Cancer Encoding

1.2 Data Preperation

1.2.1 Missing Data

No missing data was found.

```
# Checking for missing data.
for column in data.columns:
    if data[column].isnull().sum() > 0:
        print(column, "has missing values")
```

Figure 3: Missing Data

1.2.2 Data Ranges

All of the columns data was found to be within their respectable ranges.

```
# Checking for missing data.
for column in data.columns:
    # Check if data is an integer.
    if data[column].dtype == 'int64' or data[column].dtype == 'int32':
        # print max value in the column
        print(column, "max value is ", data[column].max(), "and min value is ", data[column].min())
```

Figure 4: Data Ranges Code

```
GENDER max value is  1 and min value is  0
AGE max value is  87 and min value is  21
SMOKING max value is  2 and min value is  1
YELLOW_FINGERS max value is  2 and min value is  1
ANXIETY max value is  2 and min value is  1
CHRONIC DISEASE max value is  2 and min value is  1
FATIGUE  max value is  2 and min value is  1
ALLERGY  max value is  2 and min value is  1
WHEEZING max value is  2 and min value is  1
ALCOHOL max value is  2 and min value is  1
COUGHING max value is  2 and min value is  1
SHORTNESS OF BREATH max value is  2 and min value is  1
SWALLOWING DIFFICULTY max value is  2 and min value is  1
CHEST PAIN max value is  2 and min value is  1
LUNG_CANCER max value is  1 and min value is  0
```

Figure 5: Data Ranges Output

1.2.3 Normalizing Data

Data was normalized mainly because of the age column since the data in it was not binary, and normalizing it would help the most in model calculations. Normalizing other columns also made sure that the whole data is in one format.

```
# Standardize the data
scaler = StandardScaler()
train_x = scaler.fit_transform(train_x)
test_x = scaler.transform(test_x)
```

Figure 6: Data Normalization

2 Method and Experiments

The classification method I chose is k-NN. I applied 10-fold cross-validation to evaluate the results. To determine the best k value for k-NN, I used the elbow method[1] for k values from 1 to 31.

2.1 Cross Validation

I'm using 10-fold cross-validation, which divides the data into ten parts, each of which is used as a test part for the model. This way ten models are generated and their combined average accuracy is taken to provide a more reliable assessment of how well a model is. This is used to prevent overfitting.

2.2 K Nearest Neighbour

K-NN works by clustering similarly valued data by taking their Euclidean distance. This is a type of lazy learner, and no model is created. In this model, the training data is stored and kept, and processing is only done when test data is given. When labeling the test data, the k value determines the number of close data points to consider in the voting.

2.2.1 K-NN Code

The function written for the k-NN takes the data and the labels as arguments and applies 10-fold cross-validation using k-NN on it and then returns the true labels of the data and the predicted labels of the data by the model.

```
def evaluate_model(data_x, data_y, k):
    k_fold = KFold(10, shuffle=True, random_state=1)

    predicted_targets = np.array([])
    actual_targets = np.array([])

    for train_ix, test_ix in k_fold.split(data_x):
        train_x, train_y, test_x, test_y = data_x[train_ix], data_y[train_ix], data_x[test_ix], data_y[test_ix]

        # Standardize the data
        scaler = StandardScaler()
        train_x = scaler.fit_transform(train_x)
        test_x = scaler.transform(test_x)

        # Fit the classifier
        knn = KNeighborsClassifier(n_neighbors=k)
        classifier = knn.fit(train_x, train_y)

        # Predict the labels of the test set samples
        predicted_labels = classifier.predict(test_x)

        predicted_targets = np.append(predicted_targets, predicted_labels)
        actual_targets = np.append(actual_targets, test_y)

    return predicted_targets, actual_targets
```

Figure 7: K-NN Code

2.3 Determining best k-value

The k-NN method is applied to k values from 1 to 31, and their accuracy scores are stored, after which they are plotted, and the best k-value is determined using the elbow method[1] . The best k-value that I got is 5.

```
k_values = [i for i in range(1, 31)]
acc_scores = []

for k in k_values:
    predicted_targets, actual_targets = evaluate_model(X.values, y.values, k)
    acc_scores.append(accuracy_score(actual_targets, predicted_targets))

error = np.array(1) - np.array(acc_scores)

sns.lineplot(x=k_values, y=error, marker='o')
plt.xlabel("K Values")
plt.ylabel("Error")
plt.show()
```

Figure 8: Determining best k-value code

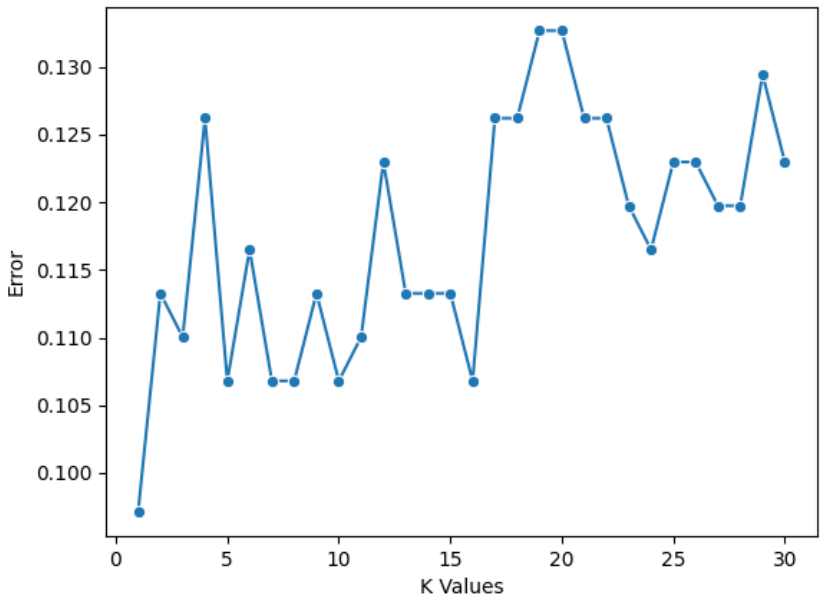

Figure 9: Plot for Elbow Method

3 Evaluation and Results

Using the best k-value of 5, the confusion matrix and other scores are calculated. The scores calculated represent a good model. The confusion matrix is plotted using the code provided by this article[2].

Table 1: Scores

Specificity:	0.3589
Sensitivity:	0.9703
Accuracy:	0.8932
F1 Score:	0.9407
Precision:	0.9128

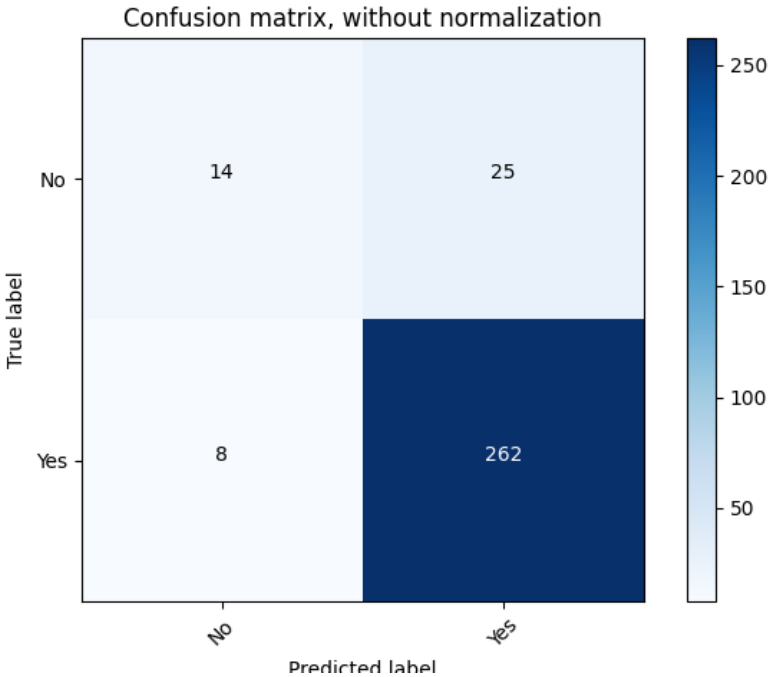

Figure 10: Confusion Matrix

3.1 Results Code

Most of the scores were obtained using internal functions provided by sklearn. Specificity was calculated based on values from the confusion matrix.

```
best_k = 5
predicted_target, actual_target = evaluate_model(X.values, y.values, best_k)
cnf_matrix = confusion_matrix(actual_target, predicted_target)
plot_confusion_matrix(cnf_matrix)

tn, fp, fn, tp = cnf_matrix.ravel()

print("Specificity: ", tn / (tn + fp))
print("Sensitivity: ", recall_score(actual_target, predicted_target))
print("Accuracy: ", accuracy_score(actual_target, predicted_target))
print("F1 Score: ", f1_score(actual_target, predicted_target))
print("Precision: ", precision_score(actual_target, predicted_target))
```

Figure 11: Results Code

4 Acknowledgements

References

[1] Soni, P. (2022) One stop for Knn, Medium. https://medium.com/@priyanshsoni761/k-nearest-neighbors-knn-1606989b7ee0#:~:text=The%20Elbow%20Method%2C%20is%20a,error%20on%20the%20test%20set.

[2] Somarathna, R. (2021) Generation of a concatenated confusion matrix in cross-validation, Medium. https://medium.com/analytics-vidhya/generation-of-a-concatenated-confusion-matrix-in-cross-validation-912485c4a972