Classification with K-Nearest Neighbors

By Wilson Peguero Rosario

Machine learning algorithms help to provide instant predictive results for large amounts of data. One such algorithm, the k-nearest neighbor (kNN) is used for pattern classification, regression models, and is ideal for data mining. Some real-world examples of its use include determining credit card ratings, identifying who's likely to default on a loan, detecting unusual patterns in credit card usage, or predicting the future value of stocks.

To perform a classification using the k-nearest neighbors algorithm, complete the following:

1. Access the "UCI Machine Learning Repository," located in the topic Resources. Note: There are about 120 datasets that are suitable for use in a classification task. For this part of the exercise, you must choose one of these datasets, provided it includes at least 10 attributes and 10,000 instances.

Will use the following data set to obtain information on how likely a person is to be readmitted based on previous factors:

Diabetes 130-US hospitals for years 1999-2008

1. You may search for data in other repositories, such as Data.gov, Kaggle or Scikit Learn.
2. Discuss the origin of the data and assess whether it was obtained in an ethical manner.

For your selected dataset, build a classification model as follows:

1. Explain the dataset and the type of information you wish to gain by applying a classification method.
2. Explain the k-nearest neighbors algorithm and how you will be using it in your analysis (list the steps, the intuition behind the mathematical representation, and address its assumptions). Assume k = 5 using the Euclidian distance. Explain the value of k.
3. Import the necessary libraries, then read the dataset into a data frame and perform initial statistical exploration.
4. Clean the data and address unusual phenomena (e.g., normalization, outliers, missing data, encoding); use illustrative diagrams and plots and explain them.
5. Formulate two questions that can be answered by applying a classification method using the k-nearest neighbors method.
6. Split the data into 80% training and 20% testing sets.
7. Train the k-nearest neighbors classifier on the training set using the following parameters: k = 5, metric = 'minkowski', p = 2.
8. Make classification predictions.
9. Interpret the results in the context of the questions you asked.
10. Validate your model using a confusion matrix, accuracy score, ROC-AUC curves, and k-fold cross validation. Then explain the results.
11. Include all mathematical formulas used and graphs representing the final outcomes.

Prepare a comprehensive technical report as a markdown document or Jupyter notebook, including all code, code comments, all outputs, plots, and analysis. Make sure the project documentation contains a) problem statement, b) algorithm of the solution, c) analysis of the findings, and d) references.

Initial Impressions of the Data Set

The data set contains anonymized medical records, which pose no issues in terms of publishing the data set. The data collected is also related to routine check-ups done on people who suffer from diabetes in the first place, making the data set completely safe and ethical to use as it does not expose the patient's information directly.

Problem Statement

Diabetes is a chronic health condition where one is unable to produce the insulin required to store the sugar that inhabits the bloods. This disease is something that needs to be constantly monitored and is estimated to be ranked as the 7th prevalent mortality factor by 2030, making diabetes a priority for many countries. One way to properly monitor diabetes patients is through estimating whether they will require a readmission based on initial factors.

Algorithm of the Solution

Before any predictive modelling can be done, we must sift through all of the features associated with the data set and remove any feature that may be of least importance through exploratory data analysis.

In [ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

df__diabetes = pd.read_csv('dataset_diabetes/diabetic_data.csv')
#print(df__diabetes.dtypes)
df__diabetes['race'] = pd.Categorical(df__diabetes['race'])
df__diabetes['gender'] = pd.Categorical(df__diabetes['race'])
df__diabetes['age'] = pd.Categorical(df__diabetes['race'])
df__diabetes = df__diabetes.drop('weight', axis=1)
for (column, data) in df__diabetes.iteritems():
    coltype = df__diabetes[str(column)].dtype
    print(f"{column} contains {len(list(filter(lambda x:x in ['?'], data)))} that are '?' and is c
```

```
encounter_id contains 0 that are '?' and is of type int64
patient_nbr contains 0 that are '?' and is of type int64
race contains 2273 that are '?' and is of type category
gender contains 2273 that are '?' and is of type category
age contains 2273 that are '?' and is of type category
admission_type_id contains 0 that are '?' and is of type int64
discharge_disposition_id contains 0 that are '?' and is of type int64
admission_source_id contains 0 that are '?' and is of type int64
time_in_hospital contains 0 that are '?' and is of type int64
payer_code contains 40256 that are '?' and is of type object
medical_specialty contains 49949 that are '?' and is of type object
num_lab_procedures contains 0 that are '?' and is of type int64
num_procedures contains 0 that are '?' and is of type int64
num_medications contains 0 that are '?' and is of type int64
number_outpatient contains 0 that are '?' and is of type int64
number_emergency contains 0 that are '?' and is of type int64
number_inpatient contains 0 that are '?' and is of type int64
diag_1 contains 21 that are '?' and is of type object
diag_2 contains 358 that are '?' and is of type object
diag_3 contains 1423 that are '?' and is of type object
number_diagnoses contains 0 that are '?' and is of type int64
max_glu_serum contains 0 that are '?' and is of type object
```

```
A1Cresult contains 0 that are '?' and is of type object
metformin contains 0 that are '?' and is of type object
repaglinide contains 0 that are '?' and is of type object
nateglinide contains 0 that are '?' and is of type object
chlorpropamide contains 0 that are '?' and is of type object
glimepiride contains 0 that are '?' and is of type object
acetohexamide contains 0 that are '?' and is of type object
glipizide contains 0 that are '?' and is of type object
glyburide contains 0 that are '?' and is of type object
tolbutamide contains 0 that are '?' and is of type object
pioglitazone contains 0 that are '?' and is of type object
rosiglitazone contains 0 that are '?' and is of type object
acarbose contains 0 that are '?' and is of type object
miglitol contains 0 that are '?' and is of type object
troglitazone contains 0 that are '?' and is of type object
tolazamide contains 0 that are '?' and is of type object
examide contains 0 that are '?' and is of type object
citoglipton contains 0 that are '?' and is of type object
insulin contains 0 that are '?' and is of type object
glyburide-metformin contains 0 that are '?' and is of type object
glipizide-metformin contains 0 that are '?' and is of type object
glimepiride-pioglitazone contains 0 that are '?' and is of type object
metformin-rosiglitazone contains 0 that are '?' and is of type object
metformin-pioglitazone contains 0 that are '?' and is of type object
change contains 0 that are '?' and is of type object
diabetesMed contains 0 that are '?' and is of type object
readmitted contains 0 that are '?' and is of type object
```

Now that we know how many columns are missing values, we can then proceed to filter out those records with missing data while also converting columns of data type *object* to the *categorical* data type.

In [ ]:

```
for (col, data) in df__diabetes.iteritems():
    ctype = df__diabetes[str(col)].dtype
    df__diabetes = df__diabetes[df__diabetes[str(col)] != "?"]
    if ctype == 'object':
        df__diabetes[str(col)] = pd.Categorical(df__diabetes[str(col)])
    else:
        pass

dict__cat = {}
for (col, data) in df__diabetes.iteritems():
    ctype = df__diabetes[str(col)].dtype
    if str(ctype) == 'category':
        dict__cat[str(col)] = dict(enumerate(df__diabetes[str(col)].cat.categories))
        df__diabetes[str(col)] = df__diabetes[str(col)].cat.codes
print(len(df__diabetes))
```

```
26755
```

Now that we have our data processes and know that there are a total of 26,755 records that we currently have left after data processing, it is time to evaluate the correlation between the features and select the best possible features to represent the response variable (which is *readmitted*)

In [ ]:

```
corr_matrix = df__diabetes.corr()
corr_matrix.style.background_gradient(cmap='coolwarm', axis=None, vmin=-1, vmax=1).format(precisic
```

Out[ ]:

	encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id
encounter_id	1.00	0.45	0.20	0.20	0.20	0.03	0.03
patient_nbr	0.45	1.00	0.29	0.29	0.29	0.08	0.02

	encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id
race	0.20	0.29	1.00	1.00	1.00	0.14	0.01
gender	0.20	0.29	1.00	1.00	1.00	0.14	0.01
age	0.20	0.29	1.00	1.00	1.00	0.14	0.01
admission_type_id	0.03	0.08	0.14	0.14	0.14	1.00	0.02
discharge_disposition_id	0.03	0.02	0.01	0.01	0.01	0.02	1.00
admission_source_id	-0.13	-0.01	0.01	0.01	0.01	-0.22	0.03
time_in_hospital	-0.01	0.01	-0.03	-0.03	-0.03	-0.06	0.20
payer_code	0.04	0.09	0.06	0.06	0.06	0.01	-0.01
medical_specialty	-0.02	-0.08	-0.01	-0.01	-0.01	0.22	0.04
num_lab_procedures	0.06	-0.00	-0.02	-0.02	-0.02	-0.34	0.04
num_procedures	0.08	0.02	0.02	0.02	0.02	0.16	0.03
num_medications	0.09	-0.04	-0.03	-0.03	-0.03	0.08	0.18
number_outpatient	0.02	0.04	0.05	0.05	0.05	0.11	0.01
number_emergency	0.05	0.03	-0.02	-0.02	-0.02	-0.02	-0.00
number_inpatient	0.04	0.04	-0.01	-0.01	-0.01	-0.01	0.04
diag_1	0.02	0.04	0.05	0.05	0.05	0.05	0.04
diag_2	0.05	0.03	0.03	0.03	0.03	0.01	0.03
diag_3	0.03	0.02	0.01	0.01	0.01	0.01	0.02
number_diagnoses	0.31	0.32	0.14	0.14	0.14	0.00	0.14
max_glu_serum	0.02	0.01	0.00	0.00	0.00	-0.05	-0.01
A1Cresult	-0.02	-0.03	-0.00	-0.00	-0.00	0.05	0.01
metformin	0.03	0.03	0.02	0.02	0.02	0.04	-0.01
repaglinide	0.04	0.11	0.04	0.04	0.04	0.01	0.02
nateglinide	0.00	0.01	-0.02	-0.02	-0.02	-0.01	-0.00
chlorpropamide	0.00	0.01	0.01	0.01	0.01	0.01	0.01
glimepiride	0.02	0.04	0.03	0.03	0.03	-0.01	-0.00
acetohexamide	nan	nan	nan	nan	nan	nan	nan
glipizide	-0.03	-0.03	-0.01	-0.01	-0.01	-0.01	0.01
glyburide	-0.02	-0.01	0.02	0.02	0.02	0.01	-0.00
tolbutamide	nan	nan	nan	nan	nan	nan	nan
pioglitazone	0.01	0.01	0.02	0.02	0.02	0.03	0.00
rosiglitazone	-0.06	0.01	0.01	0.01	0.01	0.02	0.00
acarbose	0.01	0.03	0.02	0.02	0.02	0.01	-0.00
miglitol	-0.00	0.02	-0.01	-0.01	-0.01	0.00	0.01
troglitazone	nan	nan	nan	nan	nan	nan	nan
tolazamide	-0.01	-0.01	-0.00	-0.00	-0.00	0.00	-0.00

	encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id
examide	nan	nan	nan	nan	nan	nan	nan
citoglipton	nan	nan	nan	nan	nan	nan	nan
insulin	-0.03	-0.07	-0.06	-0.06	-0.06	-0.07	0.01
glyburide-metformin	0.02	0.05	0.03	0.03	0.03	0.04	0.02
glipizide-metformin	-0.00	-0.00	0.00	0.00	0.00	-0.01	0.00
glimepiride-pioglitazone	nan	nan	nan	nan	nan	nan	nan
metformin-rosiglitazone	nan	nan	nan	nan	nan	nan	nan
metformin-pioglitazone	0.02	-0.00	0.00	0.00	0.00	0.01	0.00
change	-0.08	-0.08	-0.01	-0.01	-0.01	-0.01	-0.05
diabetesMed	0.02	-0.00	-0.01	-0.01	-0.01	-0.03	0.02
readmitted	0.01	-0.12	-0.03	-0.03	-0.03	-0.01	-0.02

From the above, it seems that there are little to no correlation among any of the features, therefore. One will utilize all of the possible features to create the KNN model.

Let us now separate the data into it's training and testing set, and create the machine learning model with 5 clusters. KNN (K-Nearest Neighbors) algorithms are based on euclidean distance, which is the distance between points in space. Those of which are close to the centroid of the cluster are categorized to be the same as the rest of the cluster.

In [ ]:
```
X = df__diabetes.drop('readmitted', axis=1)
y = df__diabetes['readmitted']
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)
knn = KNeighborsClassifier(n_neighbors = 5, p = 2, metric = 'minkowski').fit(X_train, y_train)
```

Now we can create our model with 3 clusters. Let us make predictions on the test data set and compare some basic statistics to observe how well the algorithm classifies the test data.

In [ ]:
```
predictions = knn.predict(X_test)
ct01 = pd.crosstab(y_test, predictions, rownames=['Actual'], colnames=['Predictions'])
print(ct01)
```

```
Predictions    0    1     2
Actual
0             49  185   331
1            128  724   873
2            227  876  1958
```

Evaluation Measure	Formula	Value
Accuracy	$\frac{TN+TP}{GT}	0.5057
Error Rate	$1 - Accuracy$	0.4943

From the values above, The accuracy of the model is very bad, one might even state that predicting whether a person should be admitted or not is akin to a coin toss. Looking at the curve below, one can observe that

In [ ]:
```
y_score = knn.predict_proba(X_test)
kf = KFold(n_splits=5)
for tri, tei in kf.split(X):
    print(f"TRAIN: {len(tri)/len(X)}, TEST: {len(tei)/len(X)}")
```

```
TRAIN: 0.8, TEST: 0.2
TRAIN: 0.8, TEST: 0.2
TRAIN: 0.8, TEST: 0.2
TRAIN: 0.8, TEST: 0.2
TRAIN: 0.8, TEST: 0.2
```

Based on the above K-Fold Crossvalidation split, there seems to be no mistake related to the proportion between the training data set and the test data set. Majority of the error most likely comes from the fact that the data has been clustered beyond it's realistic categories (i.e. there are only supposed to be three clusters ideally). There is also some difficulties handling the data set as diabetes can be a condition that afflicts people later in their life or even from their birth. Narrowing down and classifying the data based on the type of diabetes may improve the clustering algorithm as an added feature. To Conclude, the two questions that were in my mind throughout this assignment were:

1. Can we predict whether a patient suffering diabetes should be readmitted or not using this data set?
    - The response to this is a clear no at the moment due to the lack of significant features in relation to the readmitted response variable.
2. Which Features are crucial to estimating whether a patient is to be readmitted or not?
    - The answer to this is surprisingly None, despite the fact that there are features from lab results that should indicate whether the patient's blood sugar levels are very unstable vs stable.

Sources:

- CDC. “What Is Diabetes?” Centers for Disease Control and Prevention, 11 June 2020, www.cdc.gov/diabetes/basics/diabetes.html.
- Ti'jay Goudjerkan, and Manoj Jayabalan. “Predicting 30-Day Hospital Readmission for Diabetes Patients Using Multilayer Perceptron.” IJACSA) International Journal of Advanced Computer Science and Applications, vol. 10, no. 2, 2019, thesai.org/Downloads/Volume10No2/Paper_36-Predicting_30_Day_Hospital_Readmission_for_Diabetes_Patients.pdf. Accessed 27 Jan. 2022.
- “UCI Machine Learning Repository.” Uci.edu, 2022, archive-beta.ics.uci.edu/ml/datasets/diabetes+130+us+hospitals+for+years+1999+2008. Accessed 27 Jan. 2022.