Diabetes Detection System

Course Name: Data Mining

Course Code: SEN431

Student Name: Burak Kepüç

Student Number: B1805.090083

ABSTRACT

Data mining is the process of finding anomalies, patterns, and correlations within large data sets to predict outcomes. Using a broad range of techniques, you can use this information to increase revenues, cut costs, improve customer relationships, reduce risks, and more. In data mining Classification is a method that supervised machine learning method where the model tries to predict the correct label of a given input data. The K-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Keywords: Data mining,Predict diabetes,Classification, K-NN algorithm

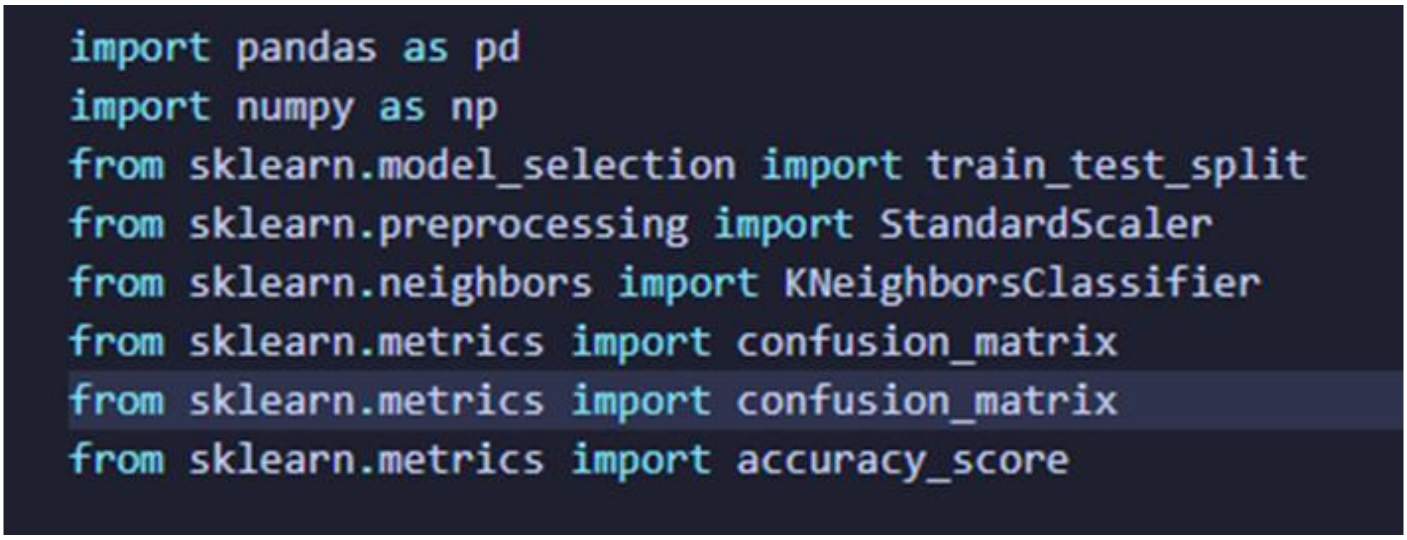
1. INTRODUCTION

Diabetes is a chronic (long-lasting) health condition that affects how your body turns food into energy. Your body breaks down most of the food you eat into sugar (glucose) and releases it into your bloodstream. When your blood sugar goes up, it signals your pancreas to release insulin.
To keep track of diabetes we need some data such as, 'Blood Pressure', 'SkinThickness', 'BMI', and 'Insulin'. In this project, we have some labeled data that defines these kinds of human data in CSV files. According to these data, we will use the Classification method so by using the K-NN algorithm to find how many people have diabetes risk.

2. STEPS

2.1 Import Libraries

In this step, we import important libraries, to use in our project. I used python because, in artificial intelligence, machine learning, and data mining area, Python has great libraries and functions to do its job. We imported libraries such as pandas, NumPy,testsplit,kneighbourclassifier and etc.



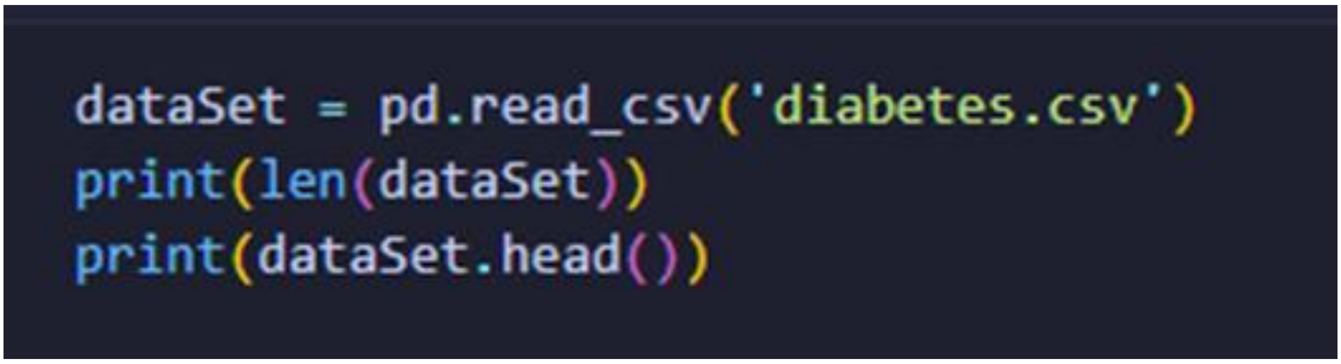
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Fig. 2. Kth-Nearest Neighbor Output

2.2 Load the dataset and have a look

With pd.read_csv method, we read the data by using pandas library and read_csv method, and assign it to dataSet variable.

Then we print the dataset length and we have look dataset itself.

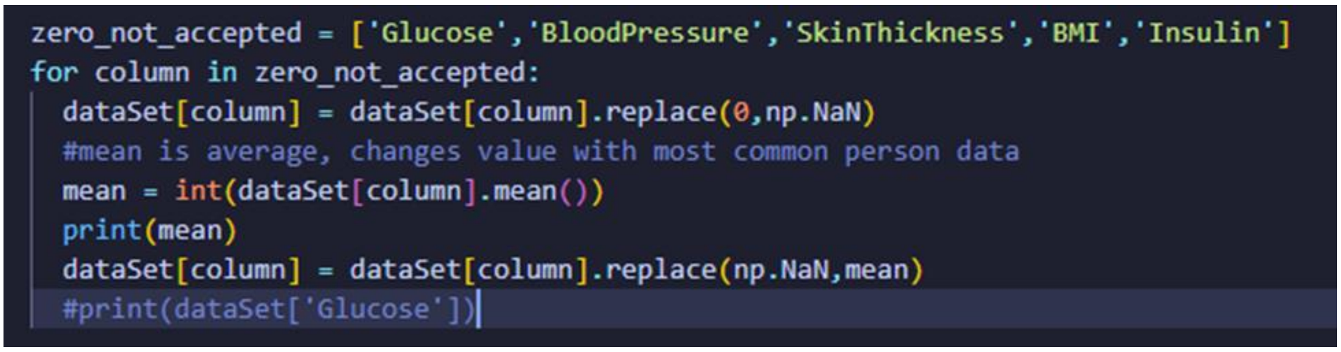


```
dataSet = pd.read_csv('diabetes.csv')
print(len(dataSet))
print(dataSet.head())
```

Fig. 3. Loading Dataset

2.3 Changing Zero Values

Values can't be zeros because it will affect the outcome, so we should replace zero values with the mean of that row and we must assign that means instead of 0 values.Basically, we chose firstly dataSet column, and assign it 0 to not a number values by using the python .replace method. Then, we take the mean of those rows by using the NumPy library and assign it a mean variable.Lastly, in the last line of code, we assign instead of np.NaN to mean

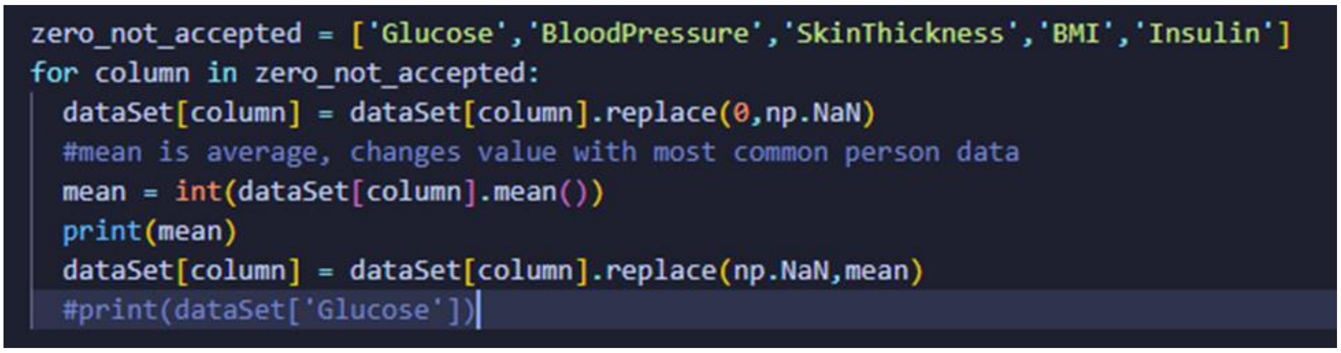


```
zero_not_accepted = ['Glucose','BloodPressure','SkinThickness','BMI','Insulin']
for column in zero_not_accepted:
    dataSet[column] = dataSet[column].replace(0,np.NaN)
    #mean is average, changes value with most common person data
    mean = int(dataSet[column].mean())
    print(mean)
    dataSet[column] = dataSet[column].replace(np.NaN,mean)
    #print(dataSet['Glucose'])
```

Fig. 4. Changing 0 values to mean of row

2.4 Train and test set

In this code, we split data set train and test. First 8 row is our training data set. And last row is our test set which is “y”. With random_state=0 , we get the same train and test sets across different executions. And test size is we will get and produce 20% of our data so we will keep our program state short.

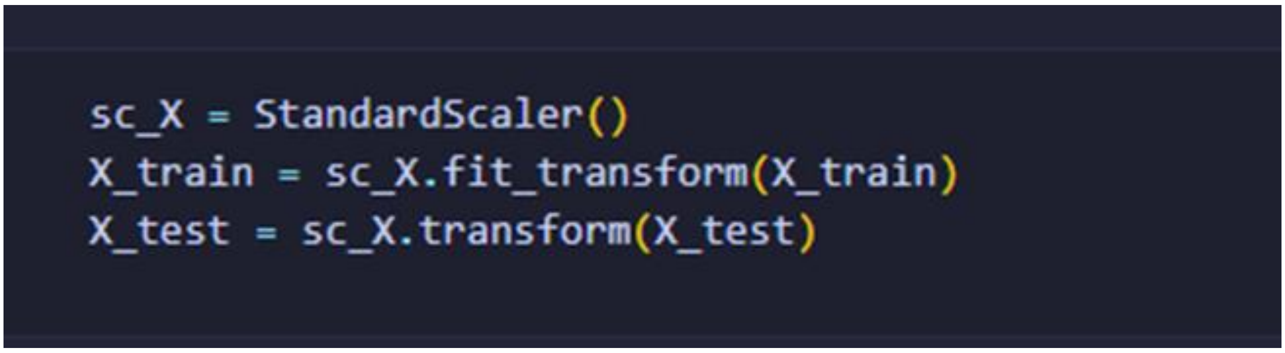


```
zero_not_accepted = ['Glucose','BloodPressure','SkinThickness','BMI','Insulin']
for column in zero_not_accepted:
    dataSet[column] = dataSet[column].replace(0,np.NaN)
    #mean is average, changes value with most common person data
    mean = int(dataSet[column].mean())
    print(mean)
    dataSet[column] = dataSet[column].replace(np.NaN,mean)
    #print(dataSet['Glucose'])
```

Fig. 5. Changing 0 values to mean of row

2.5 To standardize our data

Python's "sklearn" library offers us with StandardScaler() function to standardize the data values into a standard format. According to the above syntax, we initially create an object of the StandardScaler() function. Further, we use fit_transform() along with the assigned object to transform the data and standardize it.

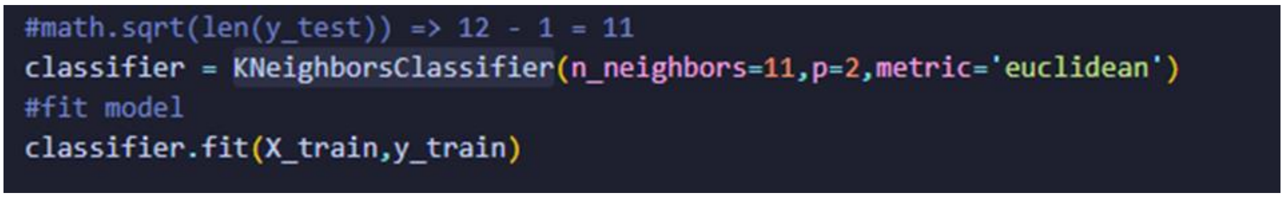


```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Fig. 6. Standard Scaler

2.6 Implementing K-NN Method

To use a classifier, we use KneighborsClassifier method. It takes some parameters. These are n_neighbour which is how the Number of neighbors to use by default for kneighbors queries, it should be an odd number. P = 2 is which method we will use, so p = 1 is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2.



```
#math.sqrt(len(y_test)) => 12 - 1 = 11
classifier = KNeighborsClassifier(n_neighbors=11,p=2,metric='euclidean')
#fit model
classifier.fit(X_train,y_train)
```

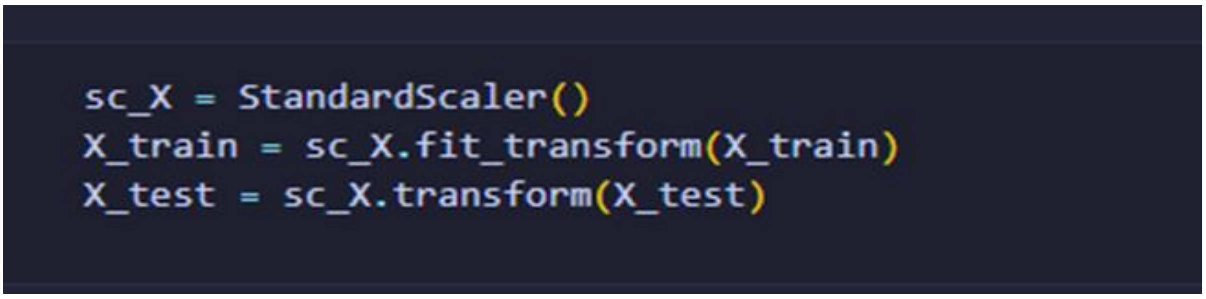
Fig. 7. K-NN method

Euclidian formula is :

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

2.7 To Predict test result

In this code we predict the data which one has diabetes and which one is not. Totally we have 154 data not all csv because we choose 20% of data, and it shows binary format. 0’s are negative, 1’s are positive.

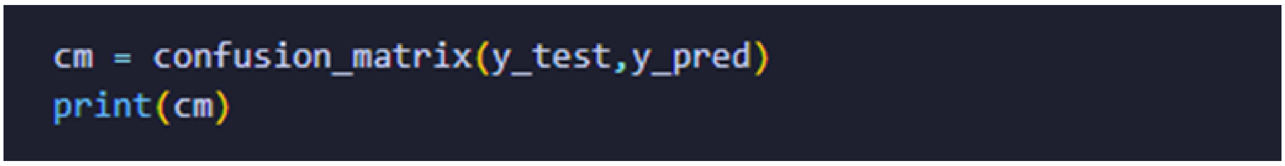


```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Fig. 8. Predict Test Result

2.8 Error Matrix

This matrix, also which known as the confusion matrix, shows how our model is evaluated. We see 94 people have no diabetes risk and 13 people have diabetes risk. Also 15 classified incorrect and 32 classified is correct. It is how classification works on our error matrix.

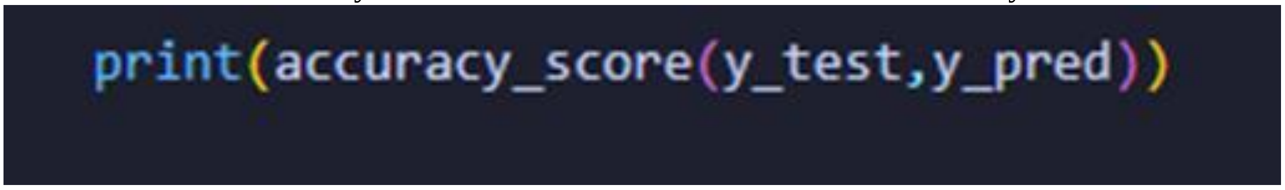


```
cm = confusion_matrix(y_test,y_pred)
print(cm)
```

Fig. 8. Error Matrix

2.9 Accuray score

Lastly, the higher the accuracy score, the better the data works and we can trust it. we see that our model works well with an accuracy score. We see that our model works 81% correctly.



```
print(accuracy_score(y_test,y_pred))
```

Fig. 10. Accuracy

3. CONCLUSION

K-Nearest Neighbors is one of the simplest supervised machine learning algorithms used for classification. It classifies a data point based on its neighbors' classifications. It stores all available cases and classifies new cases based on similar features. The KNN algorithm does not work well with large datasets. The cost of calculating the distance between the new point and each existing point is huge, which degrades performance. In the real world, the KNN algorithm has applications for both classification and regression problems. KNN is widely used in almost all industries, such as healthcare, financial services, eCommerce, political campaigns, etc. We use it to identify if a person diabetic or not according to some health data. We use some techniques and methods to analyse and clean data, then we standardized our data then used K-NN algorithms. At last, we had a look at the confusion matrix to see how to evaluate our model. Then we look at accuracy percentage., which has 80% accuracy.