Veri Madenciliği Proje 2

Halil İbrahim KOÇ – 160401025

Projede geliştirilmesi istenen ilgili modelleri geliştirmek için Anaconda Navigator içerisinde bulunan Spyder IDE, Python programlama dili kullanılmıştır. Verileri çekmek için pandas kütüphanesi, modelleri oluşturmak için ise Sci Kit Learn kütüphanesi kullanılmıştır.

Veri Önişleme

Proje 1'de verilen veri kümesinde polarite özniteliğinde eksik değerler toplam veri adedinin %50'sinden fazla olduğu için polarite özniteliğinin silinip analizlerde değerlendirmeye alınmamasına karar verilmiştir. Diğer özniteliklerdeki eksik veriler ise özniteliğin ortalama değeri ile doldurulmuştur. Verileri projeye aktarmak için pandas kütüphanesi kullanılmıştır. Verilen veri kümesi dosyası .csv uzantılı başka bir dosyaya dönüştürülerek pandas kütüphanesi ile içeriye aktarılmıştır.

Test

Verilen veri kümesinin %66'sı Modeli eğitmek için ve %33'ü ise test etmek için kullanılmıştır.

K En Yakın Komşu

Modeli oluşturmak için yazılan python kodu aşağıda verilmiştir. Test için ise test başlığı altında verilen oranlar kullanılmıştır.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jan  3 01:30:54 2020


@author: kochalilibrahim
"""
# pandas kutuphanesini import ediyoruz (importing pandas library for read the data)
import pandas as pd



# Veri yukleme: .csv uzantili hale getirdigimiz veri kumemizi pandas kutuhanesi kullanarak yukluyoruz
# Reading the data with pandas
veriler = pd.read_csv("Proje1VeriKumesi.csv", encoding='utf-8')
classless = pd.read_csv("Proje1Sinifsiz.csv", encoding='utf-8')
```

```
# Veri Onisleme (Data preprocessing)
# Eksik verileri doldurmak icin sütunların ortalamasını kullanıyoruz.
# We are using the mean for missing values.
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values="NaN", strategy="mean", axis=0)


x = veriler.iloc[:, 1:10].values  # bagimsiz degiskenler (independent variables)
y = veriler.iloc[:, 10:].values  # bagimli degisken (dependent variable)
xClassless = classless.iloc[:, 1:10].values  # bagimsiz degisken classless (independent variables for
 classles data)


imputer = imputer.fit(x[:, 1:10])
x[:, 1:10] = imputer.transform(x[:, 1:10])


# Verilerin egitim ve test icin bölünmesi, verilerin 1/3 lük kismini test icin ayiriyoruz.
# Seperating  the data to 3 piece for training and test. 1 out of 3 for test, 2 out of 3 for training.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=0)


# verileri ölcekliyoruz. (scaling transaction for data)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
xClassless = sc.transform(xClassless)


# KNN
from sklearn.neighbors import KNeighborsClassifier


knn = KNeighborsClassifier(n_neighbors=15, metric='manhattan')
knn.fit(X_train, y_train)  # Training


y_pred = knn.predict(X_test)  # predicts for test class
predClassless = knn.predict(xClassless)  # predicts for classless data


# Confusion matrix for KNN
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print("---------------------------------KNN----------------------------------------\n")
print("Confusion Matrix: ")
print(cm)
print("\nKNN predicts for classless datas: ")
```

```
print(" ID = 24: "+predClassless[0] + ", ID = 44: "+predClassless[1] + ", ID = 54:
 "+predClassless[2]+"\n")
```

Modelde uzaklık Minkowski , Euclidean ve Manhattan uzaklıkları için k değeri 1-20 arası degerler üzerinde test edilmiştir ve Minkowski ve Euclidean uzaklıkları için optimum k değeri 13 olarak saptanmıştır ve confusion matrisi [14, 0, 0, 36] şeklinde %100 ile en yüksek accuracy oranına ulaşmıştır. Manhattan uzaklığı için ise optimum k değeri 15 olarak bulunmuştur. Confusion matrisi [14, 0, 0, 36] şeklinde ve accuracy, sensitivity ve specificity değerleri %100'dür. Mamhattan uzaklığı daha detaysız bir formüle dayandığı için daha çok komşuya ihtiyaç duyduğu düşünülebilir.

Desicion Tree(Karar Ağacı)

Modeli oluşturmak için projeye eklenen python kodu aşağıda verilmiştir. Test için ise test başlığı altında verilen oranlar kullanılmıştır.

```
# Desicion Tree
from sklearn import tree
dtc = tree.DecisionTreeClassifier(criterion="entropy")
dtc.fit(X_train, y_train)  # Training

y_pred = dtc.predict(X_test)  # predicts for test class
predClassless = dtc.predict(xClassless)  # predicts for classless data

# Confusion Matrix for DSCTREE
cm = confusion_matrix(y_test, y_pred)
print("--------------------------------DSCTREE---------------------------------------\n")
print("Confusion Matrix: ")
print(cm)
print("\nDSCTREE predicts for classless datas: ")
print(" ID = 24: "+predClassless[0] + ", ID = 44: "+predClassless[1] + ", ID = 54:
 "+predClassless[2]+"\n")
```

Desicion Tree model testinde optimum kriterin entropi olduğu bulunmuş ve kullanılmıştır ve hesaplamalar bu şekilde yapılmıştır. Hesaplama sonucunda confusion matrisi [14, 0, 3, 33] şeklinde oluşmuş ve %94 accuracy oranı ile sonuçlanmıştır.

Neural Network Models (Multi-layer Perceptron)

Modeli oluşturmak için projeye eklenen python kodu aşağıda verilmiştir. Test için ise test başlığı altında verilen oranlar kullanılmıştır.

```
# Neural Network Models Multi-layer Perceptron
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(solver='lbfgs', alpha=1e-5)
mlp.fit(X_train, y_train)  # Training

y_pred = mlp.predict(X_test)  # predicts for test class
predClassless = mlp.predict(xClassless)  # predicts for classless data

# Confusion Matrix for MLP
cm = confusion_matrix(y_test, y_pred)
print("-------------------------NEURAL_NETWORK_MLP--------------------------------\n")
print("Confusion Matrix: ")
print(cm)
print("\nNEURAL_NETWORK_MLP predicts for classless datas: ")
print(" ID = 24: "+predClassless[0] + ", ID = 44: "+predClassless[1] + ", ID = 54:
 "+predClassless[2]+"\n")
```

Solver olarak Limited-memory Broyden-Fletcher-Goldfarb-Shanno algoritması ve alpha parametresi için 1e-5 formülü kullanılmıştır. Hesaplama sonucunda confusion matrisi [12, 2, 0, 36] şeklinde oluşmuş ve %96 accuracy oranı ile sonuçlanmıştır.

Sınıfı Belli Olmayan Veriler

Sınıfı belli olmayan ID = 24, ID = 44 ve ID =53 verileri için modeller tek tek uygulandığında

KNN için:
ID = 24: yaşanamaz, ID = 44: yaşanamaz, ID = 54: yaşanamaz

DESICION TREE için:
ID = 24: yaşanabilir, ID = 44: yaşanamaz, ID = 54: yaşanamaz

MULTI-LAYER PERCEPTRON için:
ID = 24: yaşanamaz, ID = 44: yaşanamaz, ID = 54: yaşanamaz
sonuçları elde edilmektedir.

SONUÇ

Modelleri karşılaştırdığımızda karşımıza çıkan tablo ise şu şekilde.

KNN: Modelde optimum k sayısını ve optimum uzaklık formülünü bulabilmek kolay değil fakat bulunduktan sonra elde edilen sonuçların özellikle confusion matris incelendiğinde tüm oranların diğer modellere göre daha kesin ve kararlı olduğu görülüyor.

DESICION TREE: Modelde seçilen kritere göre farklı sonuçlar çıkabiliyor. KNN ‘ye göre bu veri kümesinde daha az kararlı ve az daha az kesinlikte bir sonuç elde ettik. Ancak yine de Neural Network Multi-layer Perceptron modeline göre confusion matrisi üzerinden göreceğimiz üzere sensitivity oranında daha yüksek yüzde sağlıyor.

NEURAL NETWORK MULTI-LAYER PERCEPTRON: Modelde seçilen Limited-memory Broyden-Fletcher-Goldfarb-Shanno algoritması ve alpha parametresi için 1e-5 formülü sonucunda KNN ‘ye göre bu veri kümesinde daha az kesinlikte bir sonuç elde ettik. DESICION TREE modeline göre confusion matrisi üzerinden daha az bir sensitivity oranı ancak daha yüksek bir specificity oranı elde ettiğimizi gördük.