T.C.

FIRAT ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ

YAZILIM MÜHENDİSLİĞİNDE GÜNCEL KONULAR

4.Aşama 2.Rapor

GİZEM ÇOBAN

175541307

PROJE DANIŞMANI

DOÇ. DR. FATİH ÖZKAYNAK

2019-2020

1. **Giriş**

   Bu hafta 4. Aşamada daha önceden belirlemiş olduğumuz veri seti için araştırılan modellerin Python dilinde kodlanıp başarı ve F1-Score değerlerine bakıldı.

2. **Yapılan Çalışmalar**

   Proje uygun olduğunu düşünülen modellerimiz şu şekildedir;

DecisionTreeClassifier

Bir Karar Ağacı, örnekleri sınıflandırmak için basit bir temsildir. Verilerin belirli bir parametreye göre sürekli olarak bölündüğü bir Makine Öğrenimidir. Karar ağacında yani DecisionTreeClassifier algoritmasında öncelikle bir ağaç oluşturulur.

Python'daki kodları Görsel 1'de yer almaktadır.

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 15 18:11:23 2020

@author: Gizem Coban

"""

# DecisionTreeClassifier

# Kütüphanelerin import edilmesi
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Veri setinin okunması ve bağımsız-bağımlı değişkenlerin ayrılması
dataset = pd.read_csv('hki.csv')


X=dataset.drop(["Tarih","HKI"], axis=1)
y=dataset["HKI"]
# Feature Scaling (Özellik ölçekleme)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
# Veri setinin eğitim ve test olarak ayrılması
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Eğitim veri setine Decision Tree sınıflandırıcısının uygulanması
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)


# Test verileri ile tahmin yapılması
y_pred = classifier.predict(X_test)


# Confusion matrisinin oluşturulması
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

#Accuracy (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print("Accuracy", acc)


#f1_score (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import  f1_score
f1_score = f1_score(y_test, y_pred)
print("f1 score", f1_score)
```

Görsel 1: DecisionTreeClassifier

Çıkan sonuç Görsel 2'de yer almaktadır.

```
In [6]: runfile('C:/Users/ASUS/Desktop/VeriBilimi_eski/
DecisionTreeClassifier.py', wdir='C:/Users/ASUS/Desktop/
VeriBilimi_eski')
Accuracy 1.0
f1 score 1.0
```

Görsel 2: DecisionTreeClassifier Sonucu

Bu algoritma eğitim sonucunda ezber yapmıştır. Bu sebepten dolayı projede kullanılması uygun değildir.

KNeighborsClassifier (KNN)

K En Yakın Komşu (KNN) çok basit, anlaşılması kolay, çok yönlü ve en üst düzey makine öğrenme algoritmalarından biridir. KNN algoritması hem sınıflandırma hem de regresyon problemleri için kullanılır. Özellik benzerliği yaklaşımına dayalı bir algoritmadır. KNN finans, sağlık, siyaset bilimi, el yazısı algılama, görüntü tanıma ve video tanıma gibi çeşitli uygulamalarda kullanılır.

Python'daki kodları Görsel 3'de yer almaktadır.

```
# K-Nearest Neighbors (K-NN)

# Kütüphanelerin import edilmesi
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Veri setinin okunması ve bağımsız-bağımlı değişkenlerin ayrılması
dataset = pd.read_csv('hki.csv')


X=dataset.drop(["Tarih","HKI"], axis=1)
y=dataset["HKI"]
# Feature Scaling (Özellik ölçekleme)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
# Veri setinin eğitim ve test olarak ayrılması
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)



# Eğitim veri setine KNN sınıflandırıcısının uygulanması
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)

# Test verileri ile tahmin yapılması
y_pred = classifier.predict(X_test)


# Confusion matrisinin oluşturulması
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

#Accuracy (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print("Accuracy", acc)


#f1_score (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import  f1_score
f1_score = f1_score(y_test, y_pred)
print("f1 score", f1_score)

```

Görsel 3: KNeighborsClassifier (KNN)

Çıkan sonuç Görsel 4'de yer almaktadır.

```
In [1]: runfile('C:/Users/ASUS/Desktop/VeriBilimi_eski/Bayes.py',
wdir='C:/Users/ASUS/Desktop/VeriBilimi_eski')

In [2]: runfile('C:/Users/ASUS/Desktop/VeriBilimi_eski/KNN.py',
wdir='C:/Users/ASUS/Desktop/VeriBilimi_eski')
Accuracy 0.9239130434782609
f1 score 0.9542483660130718

In [3]:
```

Görsel 4: KNeighborsClassifier (KNN) Sonucu

GaussianNB (Naive Bayes)

Naive Bayes sınıflandırıcısının temeli Bayes teoremine dayanır. lazy (tembel) bir öğrenme algoritmasıdır aynı zamanda dengesiz veri kümelerinde de çalışabilir. Algoritmanın çalışma şekli bir eleman için her durumun olasılığını hesaplar ve olasılık değeri en yüksek olana göre sınıflandırır. Az bir eğitim verisiyle çok başarılı işler çıkartabilir. Test kümesindeki bir değerin eğitim kümesinde gözlemlenemeyen bir değeri varsa olasılık değeri olarak 0 verir yani tahmin yapamaz. Bu durum genellikle Zero Frequency (Sıfır Frekans) adıyla bilinir. Bu durumu çözmek için düzeltme teknikleri kullanılabilir. En basit düzeltme tekniklerinden biri Laplace tahmini olarak bilinir. Kullanım alanlarına örnek olarak gerçek zamanlı tahmin, çok sınıflı tahmin, metin sınıflandırması, spam filtreleme, duyarlılık analizi ve öneri sistemleri verilebilir

Python'daki kodları Görsel 5'de yer almaktadır.

```python
# Navie Bayes

# Kütüphanelerin import edilmesi
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Veri setinin okunması ve bağımsız-bağımlı değiskenlerin ayrılması
dataset = pd.read_csv('hki.csv')


X=dataset.drop(["Tarih","HKI"], axis=1)
y=dataset["HKI"]
# Feature Scaling (Özellik ölcekleme)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
# Veri setinin eğitim ve test olarak ayrılması
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)


 # Eğitim veri setine Naive Bayes sınıflandırıcısının uygulanması
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Test verileri ile tahmin yapılması
y_pred = classifier.predict(X_test)


# Confusion matrisinin olusturulması
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

#Accuracy (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print("Accuracy", acc)


#f1_score (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import  f1_score
f1_score = f1_score(y_test, y_pred)
print("f1 score", f1_score)

```

Görsel 5: Naive Bayes

Çıkan sonuç Görsel 6'da yer almaktadır.

```
In [3]: runfile('C:/Users/ASUS/Desktop/VeriBilimi_eski/Bayes.py',
wdir='C:/Users/ASUS/Desktop/VeriBilimi_eski')
Accuracy 0.9130434782608695
f1 score 0.9459459459459458
```

Görsel 6: Naive Bayes Sonucu

RandomForestClassifier

Karar ağaçlarının en büyük problemlerinden biri aşırı öğrenme-veriyi ezberlemedir (overfitting). Random Forest modeli bu problemi çözmek için hem veri setinden hem de öznitelik setinden Rastgele olarak 10'larca 100'lerce farklı alt-setler seçiyor ve bunları eğitiyor. Bu yöntemle 100'lerce karar ağacı oluşturuluyor ve her bir karar ağacı bireysel olarak tahminde bulunuyor.
Python'daki kodları Görsel 7'de yer almaktadır.

```python
# -*- coding: utf-8 -*-
"""
Created on Fri May 15 18:11:23 2020

@author: Gizem Coban

"""

#Random Forest

# Kütüphanelerin import edilmesi
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Veri setinin okunması ve bağımsız-bağımlı değişkenlerin ayrılması
dataset = pd.read_csv('hki.csv')


X=dataset.drop(["Tarih","HKI"], axis=1)
y=dataset["HKI"]
# Feature Scaling (Özellik ölçekleme)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
# Veri setinin eğitim ve test olarak ayrılması
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Eğitim veri setine Random Forest sınıflandırıcısının uygulanması
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)


# Test verileri ile tahmin yapılması
y_pred = classifier.predict(X_test)


# Confusion matrisinin oluşturulması
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

#Accuracy (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print("Accuracy", acc)


#f1_score (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import  f1_score
f1_score = f1_score(y_test, y_pred)
print("f1 score", f1_score)

```

Görsel 7: RandomForestClassifier

Çıkan sonuç Görsel 8'de yer almaktadır.

```
In [8]: runfile('C:/Users/ASUS/Desktop/VeriBilimi_eski/Random
Forest.py', wdir='C:/Users/ASUS/Desktop/VeriBilimi_eski')
Accuracy 0.9891304347826086
f1 score 0.9934640522875817
```

Görsel 8: RandomForestClassifier Sonucu

SVC

Support Vector Machine sınıflandırma için kullanılan yöntemlerden birisidir. Temel olarak iki sınıfı bir doğru veya düzlem ile birbirinden ayırmaya çalışır. Bu ayırmayı da sınırdaki elemanlara göre yapar. SVC sınıfı scikit-learn kütüphanesi SVM modülünün altından projeye dahil edilmektedir.

Python'daki kodları Görsel 9'da yer almaktadır.

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 15 18:11:23 2020

@author: Gizem Coban

"""

#SVC

# Kütüphanelerin import edilmesi
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Veri setinin okunması ve bağımsız-bağımlı değişkenlerin ayrılması
dataset = pd.read_csv('hki.csv')


X=dataset.drop(["Tarih","HKI"], axis=1)
y=dataset["HKI"]
# Feature Scaling (Özellik ölçekleme)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
# Veri setinin eğitim ve test olarak ayrılması
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Eğitim veri setine SVM sınıflandırıcısının uygulanması
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)

# Test verileri ile tahmin yapılması
y_pred = classifier.predict(X_test)


# Confusion matrisinin oluşturulması
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

#Accuracy (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print("Accuracy", acc)


#f1_score (sınıflandırma için başarım metriği) ölçülmesi
from sklearn.metrics import  f1_score
f1_score = f1_score(y_test, y_pred)
print("f1 score", f1_score)
```

Görsel 9: SVC

Çıkan sonuç Görsel 10'da yer almaktadır.

```
In [9]: runfile('C:/Users/ASUS/Desktop/VeriBilimi_eski/SVC.py',
wdir='C:/Users/ASUS/Desktop/VeriBilimi_eski')
Accuracy 0.9456521739130435
f1 score 0.9668874172185431
```

Görsel 10: SVC Sonucu

Yapay Sinir Ağı (YSA)

Yapay sinir ağları; insan beyninden esinlenerek geliştirilmiş, ağırlıklı bağlantılar aracılığıyla birbirine bağlanan ve her biri kendi belleğine sahip işlem elemanlarından oluşan paralel ve dağıtılmış bilgi isleme yapıları; bir başka deyişle, biyolojik sinir ağlarını taklit eden bilgisayar programlarıdır.

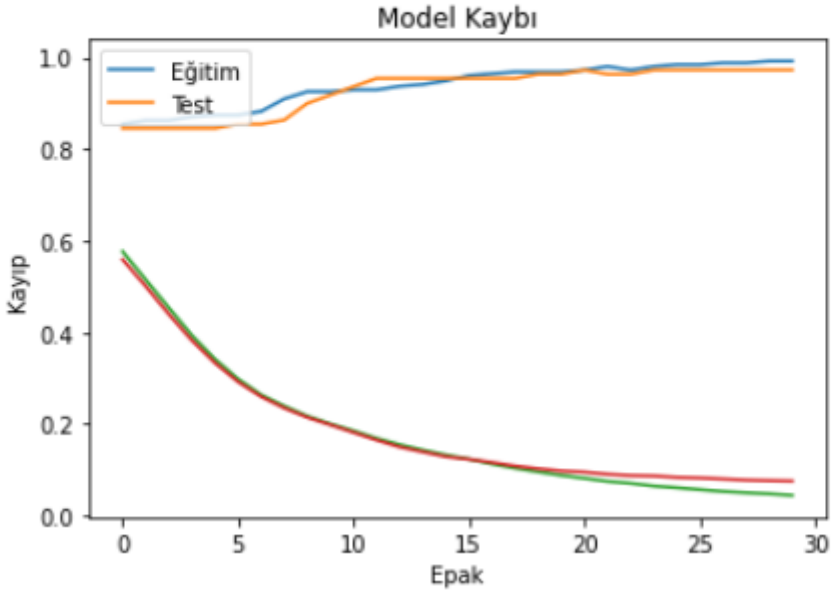
Python'daki kodları Görsel 11'da yer almaktadır.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder

#verisetinin yüklenmesi
veri=pd.read_csv("hki.csv")
#sınıf sayısını belirle
label_encoder=LabelEncoder().fit(veri.HKI)
labels=label_encoder.transform(veri.HKI)
classes=list(label_encoder.classes_)

x=veri.drop(["Tarih","HKI"], axis=1)
y=labels

#verilerin standartlaştırılması
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)

#eğitim ve test verilerinin hazırlanması
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3) #%20sini testte kullanma


#çıktı değerlerinin kategorileştirilmesi
from tensorflow.keras.utils import to_categorical
y_train=to_categorical(y_train)
y_test=to_categorical(y_test)


#YSA Modelinin oluşturulması
from tensorflow.keras.models import Sequential

#Dense ysa oluşturacağımız kısımdır.
from tensorflow.keras.layers import Dense

#16 tane nöron olsun. 20 tane alanımız bulunmakta. activasyonumuzda relu olsun
model=Sequential()
#girdi katmanı
model.add(Dense(16,input_dim=11,activation="relu"))
#ara katmanlar 3 tane eklendi
model.add(Dense(12,activation="relu"))
model.add(Dense(14,activation="relu"))
model.add(Dense(10,activation="relu"))
#çıktı katmanı
#çıktı sayısı kaçsa o kadar nöron olmak zorunda.
#activasyonu softmax olmak zorunda çünkü sınıflandırma yapılmaktadır.
model.add(Dense(2,activation="softmax"))
model.summary()


#Modelin Derlenmesi
model.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"])

#modelin eğitilmesi epochs süresi40 verildi
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=30)
```

Görsel 11: Yapay Sinir Ağı Modeli

Çıkan sonuç Görsel 12’da yer almaktadır.

```
Ortalama Eğitim Kaybı: 0.18049750563999017
Ortalama Eğitim Başarımı: 0.9398692846298218
Ortalama Doğrulama Kaybı: 0.18550443674127262
Ortalama Doğrulama Başarımı: 0.9290909151236216
<function matplotlib.pyplot.show>
```



Görsel 12: Yapay Sinir Ağı Modelinin Sonucu