T.C.

FIRAT ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ

YAZILIM MÜHENDİSLİĞİNDE GÜNCEL KONULAR

5.Aşama 1.Rapor

GİZEM ÇOBAN

175541307

PROJE DANIŞMANI

DOÇ. DR. FATİH ÖZKAYNAK

2019-2020

Yapılan Çalışmalar

Geçen hafta React.js ile web arayüzü tasarlandı. Bu hafta ise geleceğe yönelik veri tahmini yapılacaktır. Bu tahmin sonuçlarını gösterebilmemiz için bir tane web servise ihtiyacımız bulunmaktadır. Web servisi yapabilmemiz için "Flask" adında bir framework kullanıldı.

Flask Nedir?

Flask, web uygulamalar geliştirmemizi sağlayan %100 Python programlama dili ile yazılmış olan bir frameworktur. Web ortamının Backend kısmında geliştirme yapaya yarayan Flask frameworkunun birçok hazır modülü bulunmaktadır.

Flask'ı projede kullanabilmemiz için öncellikle kurulumunu yapmamız gerekmektedir. Kurulum için Anaconda'nın komut ekranına aşağıdaki komutları yazmak gerekir;

```
pip install Flask
```

Kurulum tamamlanınca Şekil 1'deki gibi gerekli kütüphaneler projeye dahil edilmektedir.

```
from flask import Flask,render_template,redirect,url_for,request
from gevent.pywsgi import WSGIServer
import tensorflow.keras
from tensorflow.keras.models import model_from_json
import pandas as pd
import datetime
import random
import numpy as np
import json
from flask_cors import CORS

```

Şekil 1 Gerekli Kütüphanelerin İmport Edilmesi

Kütüphanelerin dahili bitince ilk olarak Flask sınıfından app adında bir nesne oluşturuldu. Bu nesne ile flask kütüphanesinin özelliklerini kullanacağız.

```
app=Flask(__name__)
CORS(app)
MODEL_PATH='model.h5'
```

Şekil 2 App Nesnesini Oluşturma

Şimdi nesnemiz üzerinden ilk web sayfamızı oluşturmak için Şekil 3'deki gibi kodlarımızı yazıyoruz.

```
@app.route ('/',methods=['GET'])
def index():

    return "Hello Word"
```

Şekil 3 Server'ı Oluşturma

Bu projede Yapay Sinir Ağları (YSA) modeli kullanılacaktır ve bu model üzerinden tahmin işlemi yapılacaktır. YSA modelimizi Şekil 4'deki gibi hazırlandı. Modeli tekrar tekrar eğitmemek için eğitim modelini model.json ve model.h5 dosyalarına kaydetme işlemi yapıldı. Tahmin işlemi yapılırken bu eğitilmiş model dosyası üzerinden işlemler yapılacaktır.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
veri=pd.read_csv("hki.csv")
label_encoder=LabelEncoder().fit(veri.HKI)
labels=label_encoder.transform(veri.HKI)
classes=list(label_encoder.classes_)
#x=veri.drop(["Tarih","HKI"], axis=1)
x=veri.drop(["PM10","PM10Debi","SO2","NO2","NOX","NO","O3","HavaSicakligi","RuzgarHizi","BagilNem","HavaBasinc","HKI"], axis=1)
y=labels
#kategorik verilerin kodlanması
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
#sayısal verilere dönüştürür.
labelencoder_X = LabelEncoder()
x['Tarih'] = labelencoder_X.fit_transform(x['Tarih'])

# sayısal verilere dönüştürür.
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
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 kategorilestirilmesi
from tensorflow.keras.utils import to_categorical
y_train=to_categorical(y_train)
y_test=to_categorical(y_test)

#YSA Modelinin olusturulması
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model=Sequential()
#girdi katmanı
model.add(Dense(16,input_dim=1,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"))

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)
score = model.evaluate(x_test, y_test, verbose=0)
#Model Kaydetme
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
#ağırlıkları HDF5'e seri hale getirme
model.save_weights("model.h5")
print("Model kaydedildi")
```

Şekil 4 YSA Modeli

Şekil 5'de YSA Modelinde oluşturmuş olduğumuz "model.json" ve "model.h5" dosyalarını Tahmin yapacağımız web servisimizin kodlanmış olduğu sayfaya yükleme işlemi yapılmıştır.

```
#kaydedilmiş modeli yükle
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
loaded_model.load_weights(MODEL_PATH)

```

Şekil 5 Modelin Yüklenmesi

Tahmin işlemlerini Şekil 6'daki url sayfasında yapılmaktadır.

```
@app.route ('/test',methods=['GET'])
def test():
    preds=model_predict(loaded_model)
    return preds
```

Şekil 6 Tahmin Url Sayfası

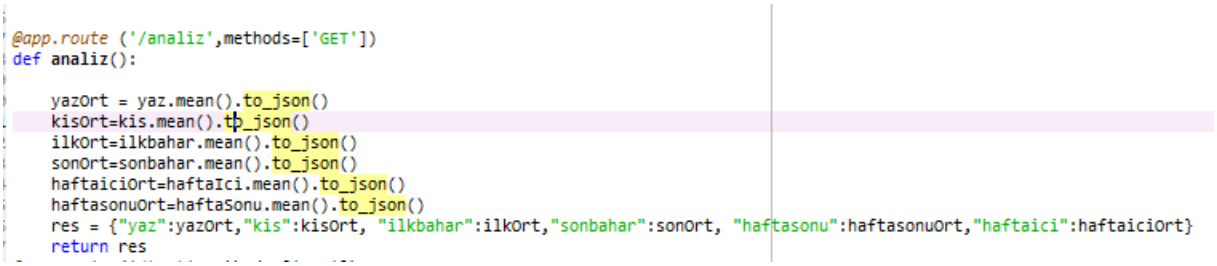
Projede "Analiz" sayfası bulunmaktadır. Bu sayfada mevsimsel, hafta sonu ve hafta içi için yapılmış ortalama analizlerin sonuçları gösterilmektedir. Bu analizleri gösterebilmek için Şekil 7'deki gibi öncelikle veri seti üzerinde mevsimsel, hafta sonu ve hafta içi için karakter analizinin yapılması gerekmektedir.

```
for i in range(len(dataset)):
    tarih = dataset["Tarih"][i].split("/")
    tarihNo= datetime.datetime(int(tarih[2]), int(tarih[0]), int(tarih[1])).weekday()
    #HKİ Sonucu
    hki=HKICalculate( dataset.iloc[i]["SO2"], dataset.iloc[i]["NO2"], dataset.iloc[i]["O3"], dataset.iloc[i]["PM10"])

    # Haftaici Haftasonu Kontrolü
    if(tarihNo<5):
        haftaIci.loc[dataset.index[i]] = dataset.iloc[i]
        haftaIci["HKI"][i] = hki
    else :
        haftaSonu.loc[dataset.index[i]] = dataset.iloc[i]
        haftaSonu["HKI"][i] = hki
    # Yaz Kış İlkbahar Sonbahar Kontrolü
    if(int(tarih[0]) in [12,1,2]):
        kis.loc[dataset.index[i]] = dataset.iloc[i]
        kis["HKI"][i] = hki
    elif(int(tarih[0]) in [3,4,5]):
        ilkbahar.loc[dataset.index[i]] = dataset.iloc[i]
        ilkbahar["HKI"][i] = hki
    elif(int(tarih[0]) in [6,7,8]):
        yaz.loc[dataset.index[i]] = dataset.iloc[i]
        yaz["HKI"][i] = hki
    else:
        sonbahar.loc[dataset.index[i]] = dataset.iloc[i]
        sonbahar["HKI"][i] = hki
```

Şekil 7 Karakter Analizi

Şekil 8'de ise karakter analizi yapılmış olan verilerin ortalaması alınarak ortalama bir analiz sonucu elde edilir. Bu sonuçlar tarayıcı üzerinde "127.0.0.1:5000/analiz" url adresi ile görülmektedir.



```
@app.route ('/analiz',methods=['GET'])
def analiz():

    yazOrt = yaz.mean().to_json()
    kisOrt=kis.mean().to_json()
    ilkOrt=ilkbahar.mean().to_json()
    sonOrt=sonbahar.mean().to_json()
    haftaiciOrt=haftaIci.mean().to_json()
    haftasonuOrt=haftaSonu.mean().to_json()
    res = {"yaz":yazOrt,"kis":kisOrt, "ilkbahar":ilkOrt,"sonbahar":sonOrt, "haftasonu":haftasonuOrt,"haftaici":haftaiciOrt}
    return res
```

Şekil 8 Analizleri