**REQUEST LİB**

import requests

import json

#SİTEDEN VERİ ALMA

r=requests.get("https://api.exchangeratesapi.io/latest")

#json datası yoksa hata verecek satır

r.json()

#site içeriğini yazdırma

print(r.text)

#Json'dan dict'e dönüştürme

result=json.loads(r.text)

# Dict'den Json'a string olarak dönüştürme

# result = json.dumps(r.text)

print(type(result))

print(result["rates"])

#TARİHE GÖRE KUR ÇEKME UYGULAMASI

#Base urlyi aldık

url=("https://api.exchangeratesapi.io/")

#Tarihi aldık

yil=input("Yılı giriniz:")

ay= input("Ayı giriniz:")

gün=input("Günü giriniz:")

#Url'ye tarihi ekleyip veriyi aldık

result=requests.get(url+yil+"-"+ay+"-"+gün)

#Veriyi json metnine çevirdik

result=json.loads(result.text)

#Yazdırdık

print(result)

#Döviz Bozma Uygulaması

#Base URL Alındı

url=api\_url = "https://api.exchangeratesapi.io/latest?base="

#Döviz türleri ve miktarları alındı

bozdurulan=input("Bozulacak Döviz Türünü Giriniz:")

alinan=input("Alınacak Döviz Türünü Giriniz:")

miktar=int(input("Miktarı Giriniz:"))

#Siteden veriyi almak

result=requests.get(url+bozdurulan)

#Veriyi Json metnine dönüştürmek

result=json.loads(result.text)

#print(result)

#Kuru yazdırdık.

print("1 {0} = {1} {2}".format(bozdurulan,result["rates"][alinan],alinan))

#Döviz Bozdurma işlemini yaptık

print("{0} {1} karşılığında {2} {3} aldınız...".format(miktar,bozdurulan,miktar\*result["rates"][alinan],alinan))

BS4

html\_doc = """

 <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>İlk web sayfam</title>

</head>

<body>

    <h1 id="header">

        Python Kursu

    </h1>

    <div class="grup1">

        <h2>

            Programlama

        </h2>

        <ul>

            <li>Menü 1</li>

            <li>Menü 2</li>

            <li>Menü 3</li>

        </ul>

    </div>

    <div class="grup2">

        <h2>

            Modüller

        </h2>

        <ul>

            <li>Menü 1</li>

            <li>Menü 2</li>

            <li>Menü 3</li>

        </ul>

    </div>

    <div class="grup3">

        <h2>

            Django

        </h2>

        <ul>

            <li>Menü 1</li>

            <li>Menü 2</li>

            <li>Menü 3</li>

        </ul>

    </div>

    <img src="fred.jpg" alt="">

    <a class="sister" href="http://example1.com/elsie" id="link1">Elsie</a>

    <a class="sister" href="http://example2.com/elsie" id="link1">Elsie</a>

    <a class="sister" href="http://example3.com/elsie" id="link1">Elsie</a>

</body>

</html>

"""

from bs4 import BeautifulSoup

#Parse ettik. (html.parser-lxml-lxml-xml-html5lib)

soup=BeautifulSoup(html\_doc,"html.parser")

#Kodları daha anlaşılabilir hale getirdik.

result=soup.prettify()

#Karşılaştığımız ilk başlığı aldık

result=soup.title

#Karşılaştığımız ilk head'i aldık

result = soup.head

#Karşılaştığımız ilk body'i aldık

result = soup.body

#Title'ın adını aldık. Metnini değil. title1,title2 gibi.

result=soup.title.name

#Başlığın metnini aldık.

result=soup.title.string

#h1'i aldık

result = soup.h1

#h2'yi aldık

result = soup.h2

#h2'nin adını aldık. Metnini değil.

result = soup.h2.name

#h2 ve h1'in metnini aldık

result = soup.h2.string

result = soup.h1.string

#h2 başlığına sahip her şeyi aldık

result = soup.find\_all('h2')

#h2 başlığına sahip elemanları tek tek alma

result = soup.find\_all('h2')[0]

result = soup.find\_all('h2')[1]

#div için aynı işlemler

result = soup.div

result = soup.find\_all('div')[1]

#Seçtiğimizin içerisinden de seçim  yapabiliyoruz.

result = soup.find\_all('div')[1].ul.find\_all('li')

#ilk div'in altındakileri verir.

result = soup.div.findChildren()

#Sonraki ve önceki div'lere ulaşma

result = soup.div.findNextSibling().findNextSibling().findPreviousSibling()

#'a' içerenleri bulma

result = soup.find\_all('a')

#'a' içerenleri yazdırdık

for link in result:

    print(link)

#'a' içeren href'leri aldık

"""for link in result:

    print(link.get('href'))"""

#İçerdiği tüm metinleri alma

result=soup.get\_text()

#Metni değiştirme

soup.title.string="GALATASARAY"

result=soup.title.string

print(result)

Numpy Exercise 1

import numpy as np

#10 ile 70 arasında 5,10 boyutunda array oluşturma

array=np.random.randint(low=10,high=70, size=(5,10))

print(array)

#Array'in türünü öğrenme

print(type(array))

#Array'in kaç boyutlu olduğunu öğrenme

print(array.ndim)

#Array'in şeklini öğrenme

print(array.shape)

Numpy Exercise 2

import numpy as np

#Numpy array'i oluşturma

result=np.array([2,3,4,5,6])

#Sıralı array oluşturma

result=np.arange(1,22)

#50'ye kadar sıralı dizi oluşturma

result = np.arange(50)

#3'er atlayan sıralı array oluşturma

result=np.arange(27,107,3)

#3 elemanlı 0 dizisi oluşturma

result=np.zeros(3)

#5 elemanlı 1 dizisi oluşturma

result=np.ones(5)

#Aralığı eş parçaya bölüp, dizi oluşturma

result=np.linspace(0,33,4)

#Aralıkta rastgele sayılarla dizi oluşturma

result=np.random.randint(0,27,5)

#Rastgele sayi

result=np.random.randint(27)

#Rastgele 5 sayili dizi oluşturma(0 ile 1 arasında)

result = np.random.rand(5)

#Rastgele 5 sayili dizi oluşturma(-2 ile 2 arasında)

result = np.random.randn(5)

#Dizi oluşturma ve şekillendirme

array=np.arange(0,50)

array1=array.reshape(5,10)

print(array1)

#Arrayi soldan sağa toplama

print(array1.sum(axis=1))

#Arrayi baştan aşağı toplama

print(array1.sum(axis=0))

#Dizideki en büyük elemanı almak

result=array1.max()

#Dizideki en küçük elemanı almak

result=array1.min()

#Dizi ortalamasını almak

result=array.mean()

#En büyük elemanın indexini verir

result=array.argmax()

#En küçük elemanın indexini verir

result=array.argmin()

print(result)

Numpy Exercise 3

import numpy as np

# (3x5) boyutlarında (10-50) arasında rastgele bir matris oluşturma

array9=np.random.randint(10,50,(3,5))

result=array9

# Matrisin satır, sütun ve bütün toplamı

rowTotal=array9.sum(axis=1)

print("Satır Toplamı:",rowTotal)

colTotal=array9.sum(axis=0)

print("Sütun Toplamı:",colTotal)

Total=array9.sum()

print("Array Toplamı:",Total)

# (10-20) arasındaki sayıları içeren dizinin ilk 3 elemanını seçme

arr=np.arange(10,20)

result=arr[:3]

# Matrisi tersten yazırma

result=arr[::-1]

# Matrisin ilk satırını alma

result=array9[0]

# Matrisin 2. satırındaki 3. elemanı alma

result=array9[1,2]

# Matrisin tüm satırlarındaki ilk elemanı alma

result=array9[:,0]

# Matrisin karesini alma

result=array9\*\*2

# Matris elemanlarının çift ve pozitif olanlarını alma

ciftler=array9[array9%2==0]

pozitifciftler=ciftler[ciftler>0]

result=pozitifciftler

# arr2 = arr1 # referans

arr2 = arr1.copy()

print(result)

Numpy Exercise 4

import numpy as np

#Arraylerle matematik işlemleri yapılabilir

numbers1 = np.random.randint(10,100,6)

numbers2 = np.random.randint(10,100,6)

#Dizi elemanlarının trigonometrik değerlerini alma

result = np.sin(numbers1)

result = np.cos(numbers1)

result = np.sqrt(numbers1)

result = np.log(numbers1)

#Dizileri alt alta stackleme

result = np.vstack((numbers1,numbers2))

#Dizileri yan yana stackleme

result = np.hstack((numbers1,numbers2))

#Dizileri tekrar vektör haline getirme

array=np.random.randint(10,100,(3,5))

vector=np.resize(array,(15))

result=vector

#Dizileri birleştirme (integer olmaları gerekir)(sütun sayısı eşit olmalı)

AR=np.zeros((3,2), dtype=int)

AR2=np.ones((3,2), dtype=int)

result=np.concatenate((AR,AR2))

print(result)

SELENİUM EXERCİSE 1

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

import time

#Tarayıcı seçme (Eğer aynı klasörde değilse tarayıcının konumunu parantez içerisinde yazmalısın)

browser=webdriver.Chrome()

#Siteye Gitme

browser.get('http://google.com')

#Tarayıcı başlığının google olup, olmadığını kontrol eder. Eğer değilse hata verir.

assert 'Google' in browser.title

#Sayfa yüklenene kadar bekleme

time.sleep(2)

#Tarayıcıyı tam ekran yapma

browser.maximize\_window()

#Arama kutusunun ve arama butonunun yolunu bulduk

sorgukutusu=browser.find\_element\_by\_xpath("//\*[@id='tsf']/div[2]/div[1]/div[1]/div/div[2]/input")

aramabutonu=browser.find\_element\_by\_xpath("//\*[@id='tsf']/div[2]/div[1]/div[3]/center/input[1]")

#Aramaya gönderilecek harfleri gönderme ve arama yapma

sorgukutusu.send\_keys('Galatasaray'+Keys.ENTER)

#aramabutonu.click() ile arama butonuna da tıklayabiliriz

#Ekran Görüntüsü alma ve isimlendirme

#browser.save\_screenshot("googlearamasonucları.png")

#İleri Geri Gitme

# driver.back()

# driver.forward()

#Tarayıcıyı kapatma

time.sleep(2)

browser.close()

PANDAS EXERCİSE 1

import pandas as pd

#Dataset açma

df = pd.read\_csv("datasets/data.csv")

result=df

#Sütunları öğrenme

result=df.columns

#Sütun sayisini öğrenme

result=len(df.columns)

#İlk 5 veriyi alma

result=df.head()

#Baştan itibaren belirli sayıda veriyi alma

result=df.head(15)

#Sondan 5 veriyi alma

result=df.tail()

#Sondan itibaren belirli sayıda veriyi alma

result=df.tail(15)

#Toplam veri sayısını öğrenme

result=len(df.index)

#Belirli bir sütundan veriyi alma

result = df["Cinsiyet"]

#Belirli bir sütundan ilk 5 veriyi alma

result=df["Cinsiyet"].head()

#AKP'li olanları alma

result=df[df["parti"]=="AKP"]

#AKP'li olanların sayısını alma

result=len(df[df["parti"]=="AKP"].index)

#AKP'li olup, ekonominin iyi olduğunu düşünen lisans mezunları

result=df[(df['parti']=='AKP') & (df['soru1']=='Evet') & (df['Egitim']=='Lisans')]

#AKP'li olup, ekonominin iyi olduğunu düşünen lisans mezunlarının sayısı

result=len(df[(df['parti']=='AKP') & (df['soru1']=='Evet') & (df['Egitim']=='Lisans')].index)

print(result)

PANDAS EXERCİSE 2

import pandas as pd

df=pd.read\_csv("datasets/nba.csv")

#Maaş ortalamasını alma

result=df['Salary'].mean()

#En yüksek maaşı alma

result=df['Salary'].max()

#En düşük maaşı alma

result=df['Salary'].min()

#En yüksek maaşı alan oyuncuyunun sadece adını alma

result=df[df['Salary']==df['Salary'].max()]['Name'].iloc[0]

#20-25 yaş oyuncuların yaşlarına göre sıralı oynadıkları takım listesi

result=df[(df['Age']>= 20) & (df['Age']<=25)][['Name','Team','Age']].sort\_values('Age',ascending=False)

#Kobe Bryant'ın takımını bulma

result=df[df['Name']=='Kobe Bryant']['Team'].iloc[0]

#Takımlara göre ortalama maaş

result=df.groupby('Team').mean()['Salary']

#Kaç farklı takım olduğunu bulma

result=len(df.groupby('Team'))

#veya

result = df["Team"].nunique()

#Her takımda kaç oyuncu bulunmaktadır

result=df['Team'].value\_counts()

#Boş verileri düşürme(inplace ile yapılırsa df'ye kaydolur. Diğer türlü geçici işlem)

df.dropna(how='any',inplace=True)

#Any olursa herhangi bir eksik ise veri silinir

#All olursa tüm değerleri eksik ise veri silinir

#İsminde 'and' geçenleri alma

result=df[df['Name'].str.contains("and")]

#veya

def str\_bul(name):

    if 'and' in name.lower():

        return True

    return False

result=df[df['Name'].apply(str\_bul)]

print(result)

DICTIONARIES

#Sözlük oluşturma

thisdict = {

  "brand": "Ford",

  "model": "Mustang",

  "year": 1964

}

#Değer yazdırma

print(thisdict["model"])

#Değer değiştirme

thisdict["year"]=2020

#Tüm anahtar kelimeleri yazdırma

for x in thisdict:

  print(x)

#Tüm değerleri yazdırma

for x in thisdict:

  print(thisdict[x])

#veya

for x in thisdict.values():

  print(x)

#Tüm anahtar kelimeleri ve değerleri yazdırma

for x, y in thisdict.items():

    print(x, y)

#Sözlükte anahtar kelime var mı diye kontrol etmek

if "model" in thisdict:

    print("model anahtarı sözlükte var.")

#Sözlük uzunluğunu öğrenme

print(len(thisdict))

#Sözlüğe ekleme yapma

thisdict["color"]="red"

#Sözlükten silme

thisdict.pop("model")

#veya

#del thisdict["model"]

#Son ekleneni silme

thisdict.popitem()

#Sözlüğü silme

#del thisdict

#Sözlüğü kopyalama

mydict=thisdict.copy()

#Sözlüğün içini boşaltma

thisdict.clear()

#Sözlük yaratma 2

thisdict = dict(brand="Ford", model="Mustang", year=1964)

#Tüm her şeyi yazdırma

print(thisdict)

print(mydict)

NESTED DICTIONARIES

# #3 sözlük içeren bir sözlük yaratma

# myfamily = {

#   "child1" : {

#     "name" : "Emil",

#     "year" : 2004

#   },

#   "child2" : {

#     "name" : "Tobias",

#     "year" : 2007

#   },

#   "child3" : {

#     "name" : "Linus",

#     "year" : 2011

#   }

# }

#3 sözlük oluşturup, bir sözlüğe yerleştirme

child1 = {

  "name" : "Emil",

  "year" : 2004

}

child2 = {

  "name" : "Tobias",

  "year" : 2007

}

child3 = {

  "name" : "Linus",

  "year" : 2011

}

myfamily = {

  "child1" : child1,

  "child2" : child2,

  "child3" : child3

}

DOSYA İŞLEMLERİ

# 'r' modu: Dosyayı sadece okumak için açar. Bu mod varsayılan moddur.

# 'r+' modu: Dosyayı hem okumak hem de yazmak için açar. Eğer çağrılan dosya bulunamadıysa yeni bir dosya oluşturulmaz.

# 'w' modu: Dosyayı sadece yazmak için açar. Varolan dosyanın üzerine yazma işlemini yapar. Eğer çağrılan dosya bulunamadıysa yeni bir dosya oluşturur.

# 'w+' modu: Dosyayı hem okumak hem de yazmak için açar. Varolan dosyanın üzerine yazma işlemini yapar. Eğer çağrılan dosya bulunamadıysa yeni bir dosya oluşturur.

# 'a' modu: Dosyayı ekleme işlemi için açar. Eğer çağrılan dosya bulunursa, en sonundan eklemeye devam eder. Eğer dosya yoksa sadece yazma işlemi yapacak yeni bir dosya oluşturur.

# 'a+' modu: Dosyayı hem ekleme hem de okuma işlemi için açar. Eğer çağrılan dosya bulunursa, en sonundan eklemeye devam eder. Eğer dosya yoksa yazma ve okuma işlemleri yapacak yeni bir dosya oluşturur.

#Dosyaya ekleme yapmak

#Encoding türkçe karakterleri okuyabilmek için var.

dosya=open("dosyaislemleri1.txt","a+",encoding="utf-8")

dosya.write("\nEkleme başarılı")

dosya.close()

#Dosyayı sürekli kapatıp açmamak için with kullanılabilir

with open("dosyaislemleri1.txt","a+",encoding="utf-8") as dosya:

    dosya.write("\nDosyamızı with ile açtık")

#OS modülü

#önce import edilmeli

# os.path.abspath : Eğer bir dosyanın en genel biçimde dosya yolunu edinmek istiyorsanız os.path.abspath fonksiyonunu kullanabilirsiniz.

# os.path.exists: Eğer bir dosyanın belirttiğiniz dosya yolunda var olup olmadığını kontrol etmek istiyorsanız os.path.exists fonksiyonunu kullanabilirsiniz.

# os.path.isdir : Elde ettiğiniz metinsel verinin bir dosya yolu(Ör: C:/Metinler) olup olmadığını kontrol etmek isterseniz os.path.isdir fonksiyonunu kullanabilirsiniz.

# os.path.isfile : Elde ettiğiniz verinin bir dosya(Ör: C:/Metinler/deneme.txt) olup olmadığını kontrol etmek isterseniz os.path.isfile fonksiyonunu kullanabilirsiniz.

# os.listdir : Eğer belirli bir dosya yolundaki dosyaların isimlerini almak istiyorsanız os.listdir fonksiyonunu kullanabilirsiniz.

#Dosyayı okuma

dosya=open("dosyaislemleri1.txt","r",encoding="utf-8")

for i in dosya:

    print(i)

dosya.close()

MYSQL

**ALTER USER 'userName'@'localhost' IDENTIFIED BY 'New-Password-Here'**

# şeklinde mysql üzerinden şifre değiştirilebilir.

import mysql.connector

#Database'e bağlanma

mydb = mysql.connector.connect(

    host = "localhost", # 192.23.45.56

    user = "root",

    password = "1234",

    database= "school"

    #database = "node-app" şeklinde istediğimiz database'e bağlanabiliriz. Ayrıca database'in var olup, olmadığını kontrol edebiliriz.

)

print(mydb)

#Database'e cursoru koyma. buffered kullanılmazsa tek veri çekimlerinde fetchall sorun yaratır.

mycursor = mydb.cursor(buffered=True)

#Var olan Database'leri görme

mycursor.execute('SHOW DATABASES')

print("\n Var olan veri tabanları \n")

for x in mycursor:

    print(x)

#Tablo yaratma

mycursor.execute("CREATE TABLE sınıflar (name VARCHAR(50))")

#Var olan tabloya benzersiz numara atama

mycursor.execute("ALTER TABLE sınıflar ADD COLUMN id INT AUTO\_INCREMENT PRIMARY KEY")

#Tablodaki her veriye benzersiz numara atama

mycursor.execute("CREATE TABLE sınıflar2 (id INT AUTO\_INCREMENT PRIMARY KEY,name VARCHAR(50))")

mycursor.execute("ALTER TABLE sınıflar2 ADD COLUMN surname VARCHAR(50)")

#Var olan tabloları görme

mycursor.execute('SHOW TABLES')

print("\n Var olan tablolar \n")

for x in mycursor:

    print(x)

#Sütun'a veri girişi

sql="INSERT INTO sınıflar2 (name,surname) VALUES (%s,%s)"

val=[('Uğur','Al'),

('Ceyda', 'Tütüncü'),

('Ceren','Teksoy'),

('Asena','Turan')]

#Birden çok veri girdiğimiz için execute'un sonuna many ekledik ve tuple kullandık.

#many eklemezsek val=[(blabla)] da köşeli parantezler olmamalı. executemany ile bir tane veri de eklenebilir.

mycursor.executemany(sql,val)

mydb.commit()

#Eklenen verileri sayma

print(mycursor.rowcount,"tane veri eklendi.")

#Eklenen son verinin id'sini alma

print("Son verinin id'si:", mycursor.lastrowid)

#Tablodaki tüm verileri alma

mycursor.execute('SELECT \* FROM sınıflar2')

myresult=mycursor.fetchall()

for x in myresult:

    print(x)

#Tablodaki seçili sütunları alma

mycursor.execute('SELECT id,surname FROM sınıflar2')

myresult1=mycursor.fetchall()

for x in myresult1:

    print(x)

#Sadece tek satırdaki verileri çekmek

mycursor.execute('SELECT \* FROM sınıflar2')

myresult2=mycursor.fetchone()

print(myresult2)

#Database'de sorgu yapma

sql="SELECT \* FROM sınıflar2 WHERE name='Tansel' "

mycursor.execute(sql)

myresultname=mycursor.fetchall()

for x in myresultname:

    print(x)

#Database'de içereni bulma

sql="SELECT \* FROM sınıflar2 WHERE name LIKE '%ü%' "

mycursor.execute(sql)

myresultname2=mycursor.fetchall()

for x in myresultname2:

    print(x)

#Kullanıcıdan giriş alarak sorgu yapma

sql="SELECT \* FROM sınıflar2 WHERE name=%s"

#"Asena"dan sonra virgül olmak zorunda

name=("Asena",)

mycursor.execute(sql,name)

myresultname3=mycursor.fetchall()

for x in myresultname3:

    print(x)

#Sorguları sıralama

print("\nOrdering by name\n")

sql="SELECT \* FROM sınıflar2 ORDER BY name"

mycursor.execute(sql)

myresultnameorder=mycursor.fetchall()

for x in myresultnameorder:

    print(x)

#Sorguları azalan şekilde sıralama

print("\nOrdering by name as descending\n")

sql="SELECT \* FROM sınıflar2 ORDER BY name DESC"

mycursor.execute(sql)

myresultnameorder=mycursor.fetchall()

for x in myresultnameorder:

    print(x)

#Veritabanından veri silme

sql=("DELETE FROM sınıflar2 WHERE name='Uğur'")

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "tane veri silindi")

#Kullanıcıya Veri Sildirme

sql=("DELETE FROM sınıflar2 WHERE name=%s")

name=("Tansel",)

mycursor.execute(sql,name)

mydb.commit()

print(mycursor.rowcount, "tane veri silindi")

#Tablo silme

sql=("DROP TABLE sınıflar2")

mycursor.execute(sql)

#Eğer varsa tabloyu silme

sql=("DROP TABLE IF EXISTS sınıflar2")

mycursor.execute(sql)

#Veri değiştirme

sql=("UPDATE sınıflar2 SET name='Aylin' WHERE name='Ayda'")

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "tane veri değiştirildi")

#Kullanıcıya veri değiştirtme

sql=("UPDATE sınıflar2 SET name=%s WHERE name=%s")

val=("Aycan","Aylin")

mycursor.execute(sql,val)

mydb.commit()

print(mycursor.rowcount, "tane veri değiştirildi")

#İlk 5 kaydı alma

print("\n ilk 5 kaydı alma\n")

mycursor.execute("SELECT \* FROM sınıflar2 LIMIT 5")

myresult = mycursor.fetchall()

for x in myresult:

  print(x)

#Belirli kayıtları alma. OFFSET değeri index gibidir.

print("\n belirli kayıtları alma\n")

mycursor.execute("SELECT \* FROM sınıflar2 LIMIT 2 OFFSET 2")

myresult = mycursor.fetchall()

for x in myresult:

  print(x)

MYSQL FOREIGN KEY EKLEME

ALTER TABLE servis ADD FOREIGN KEY (servissahibino) REFERENCES müsteri(idmüsteri)

#INNER JOIN ÖRNEĞi

#INNER YERİNE RİGHT VEYA LEFT KULLANILABİLİR

    mycursor.execute("SELECT servis.idservis, servis.servistalep, müsteri.müsteriname, müsteri.müsterisurname, personel.personelname, personel.personelsurname FROM servis INNER JOIN müsteri ON servis.servissahibino=müsteri.idmüsteri INNER JOIN personel ON servis.servispersonel=personel.idpersonel")

TENSORFLOW

#Machine Learning’de de bulunabilir.

import numpy as np

import pandas as pd

import seaborn as sbn

import matplotlib.pyplot as plt

#Verilerimizi okuyup, değişkene atadık.

dataFrame=pd.read\_excel("merc.xlsx")

#Verilere göz atmak için ilk 5 veriyi çektik.

dataFrame.head()

#Verilerimizin matematiksel olarak incelemesini yapabilmek için bu rakamları getirdik.

dataFrame.describe()

#Datamızın içerisinde eksik bilgi var mı diye kontrol ediyoruz.

dataFrame.isnull().sum()

#Fiyat tahmini yapacağımız için fiyat grafiğimizi kontrol ediyoruz.

#Grafik düzgün gözüksün diye boyutlandırma yapıyoruz.

plt.figure(figsize=(7,5))

sbn.distplot(dataFrame["price"])

#Yine araç yıllarına göre satışları da grafiğe döktük ki uç kısımları çıkartıp, daha iyi sonuç elde edelim.

sbn.countplot(dataFrame["year"])

#Verilerin arasındaki korelasyona baktık

dataFrame.corr()

#Herhangi bir değişkenin diğer değişkenlerle arasındaki korelasyona sıralı şekilde bakabiliriz.

dataFrame.corr()["price"].sort\_values()

#mil ve fiyat ilişkisine baktık

sbn.scatterplot(x="mileage",y="price",data=dataFrame)

dataFrame.sort\_values("price",ascending = False).head(20)

#Amacımız fiyatı bulmak olduğu için ve fiyat grafiğinde az sayıda satılan çok pahalı araçlar olduğundan dolayı grafiğimizde küçülmeye gideceğiz

#Bu gibi durumlarda verinin işlevsiz kalan %1lik kısmını kullanmayabiliriz

print(len(dataFrame))

len(dataFrame)\*0.01

#Fiyatları azalan şekilde sıralayıp, en yüksek 131 satış fiyatını yeni verimize eklemedik.

yuzdeDoksanDokuzDf=dataFrame.sort\_values("price", ascending=False).iloc[131:]

yuzdeDoksanDokuzDf.describe()

yuzdeDoksanDokuzDf.corr()

#Şimdi iki dataframe'i yıllara göre fiyat değişimi açısından karşılaştıracağız.

#Arada ortalama 1000 sterlinlik farklar oluşmuş.

dataFrame.groupby("year").mean()["price"]

#Şimdi iki dataframe'i yıllara göre fiyat değişimi açısından karşılaştıracağız.

#Arada ortalama 1000 sterlinlik farklar oluşmuş.

dataFrame.groupby("year").mean()["price"]

#Dataframe'imizi güncelledik.

#Yıl olarak 1970 çok uçta kaldığından ve fiyat olarak da uç olduğundan bunun bir istisna olduğunu düşünüp, onu veri setimizden çıkardık.

dataFrame = yuzdeDoksanDokuzDf[yuzdeDoksanDokuzDf.year != 1970]

dataFrame.groupby("year").mean()["price"]

dataFrame.head()

#Vites türü sınıfsal olduğundan ve çok da etkili olmadığından veri setinden atıyoruz.

dataFrame = dataFrame.drop("transmission",axis=1)

# y değerimize fiyatı, x değerimize ise data setimizin fiyat olmayan halini veriyoruz.

# bunları öğrenme yaparken kullanmak için ayırdık. Fiyatı tahmin edeceğiz.

# bu nedenle y tek boyutluyken, x çok boyutlu bir array.

y=dataFrame["price"].values

x=dataFrame.drop("price", axis=1).values

# datamızı test, train olarak ayırmak için gerekli importu yaptık

from sklearn.model\_selection import train\_test\_split

#train ve test değişkenlerimi oluşturup, atamalarını yaptık.

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y, test\_size=0.3, random\_state=10)

#şimdi bu değişkenlerin boyutlarını kontrol ediyoruz.

print(len(x\_train))

print(len(x\_test))

print(len(y\_train))

print(len(y\_test))

#Verinin daha kolay işlenebilmesi için küçültme işlemi yapacağız. Bunun için import gerekiyor.

from sklearn.preprocessing import MinMaxScaler

#MinMaxScaler instance'ı oluşturuyoruz.

scaler=MinMaxScaler()

#Scaler ile x\_trainin değerlerini küçültüyoruz.

x\_train=scaler.fit\_transform(x\_train)

#Test datamızda sadece transform kullanıyoruz . x değerleri için fit daha önce hesaplandığından tekrar fitlememize gerek yok.

x\_test=scaler.transform(x\_test)

# Sequential ile model instance'ımızı oluşturacağız.

# Dense ile çekirdek yoğunluğunu ayarlayacağız.

# Early stopping ile epochs sayısı kaç olursa olsun test ve train arasındaki fark bizim belirlediğimiz değer kadar artmaya başlarsa öğrenme duracak.

# Dropout ise çekirdeklerin işlem sırasında kaç oranla kapatılacağını ayarlıyor.

# Activation ve optimizer için google araması yapmalısın.

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation, Dropout

from tensorflow.keras.callbacks import EarlyStopping

#Train shape'ine göre ilk dense değer vereceğiz.

#Bu zorunlu değil, ama başlangıçta yol göstermek için ideal

x\_train.shape

#modelimizi oluşturduk.

model=Sequential()

#Modelimize çekirdekleri ekledik.

model.add(Dense(12,activation="relu"))

#Dropout istediğimiz kadar eklenebilir

#model.add(Dropout(0.5))

model.add(Dense(12,activation="relu"))

model.add(Dense(12,activation="relu"))

model.add(Dense(12,activation="relu"))

#Çıkış dense'i 1 iyidir.

model.add(Dense(1))

#Loss test ve train datasındaki kaybı hesaplama yöntemi.

model.compile(optimizer="adam",loss="mse")

#verbose bilgi akışını sağlar. Genelde 0 tutarız.

#earlyStopping = EarlyStopping(monitor="val\_loss",mode="min",verbose=1,patience=25)

model.fit(x=x\_train, y = y\_train,validation\_data=(x\_test,y\_test),batch\_size=250,epochs=300)

#loss değerlerini grafik çizmek için alıyoruz.

modelKaybi = pd.DataFrame(model.history.history)

modelKaybi.plot()

modelKaybi.head()

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

#tahmin değer sapmasını kontrol ediyoruz.

tahminDizisi = model.predict(x\_test)

mean\_absolute\_error(y\_test,tahminDizisi)

plt.scatter(y\_test,tahminDizisi)

plt.plot(y\_test,y\_test,"g-\*")

#Rastgele bir veriyi dataframeden düşürüp, fiyat hesaplaması yaptırdık.

dataFrame.iloc[2]

yeniArabaSeries = dataFrame.drop("price",axis=1).iloc[2]

type(yeniArabaSeries)

yeniArabaSeries = scaler.transform(yeniArabaSeries.values.reshape(-1,5))

model.predict(yeniArabaSeries)

#Sınıflandırma işlemi yaparken farklılıklar var.

#from sklearn.metrics import classification\_report, confusion\_matrix

#print(classification\_report(y\_test,tahminlerimiz))

#print(confusion\_matrix(y\_test,tahminlerimiz))

#Gibi. relu, optimizer vs. de farklılık gösterebilir.