T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 461 BÜYÜK VERİYE GİRİŞ

MODELING WEATHER DATA POINTS WITH PYTHON

B171210094 – SEFA EKİCİ
B181210391 – AHSEN DURMAZ
B171210064-TAYYİP GÜZEL

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Öğretim Görevlisi : DR.Öğr.Üyesi SÜMEYYE KAYNAK

1) GİRİŞ

Hava durumlarına ait veriler her yerde bulunabilir.Bu veri kümeleri hava durumu modellerini belirlemede ve geçmiş veri kümelerine dayalı hava durumu tahminlerinde faydalı olabilir.

Kaggle üzerinden eriştiğimiz veri kümesi (Ülke değerlerini tutan) CSV tipinde ve aşağıdaki veri alanlarına sahiptir;

- dt (Tarih ve Zaman)
- AverageTemperature (Ortalama Sıcaklık)
- AverageTemperatureUncertainty (Ortalama Sıcaklıktaki Belirsizlik)
- City (Şehir)
- Country (Ülke)
- Latitude (Enlem)
- Longitude (Boylam)

Bu sütünlarda tutulan verilerden yola çıkarak bir model geliştirmeye ve bu modeli iyileştirmeye çalışacağız.

2) TÜRKİYENİN SICAKLIK DEĞİŞİMİNİ TAHMİN ETMEK

- İlk olarak proje boyunca kullanacağımız kütüphaneleri projeye dahil ediyoruz.
Burada kitabın bölümünden farklı olarak sürüm değişiminden dolayı (pandastan datetime'ın kaldırılması.) datetime modülünü datetime üzerinden projeye dahil ettik.

In [2]:
```
#Gerekli Kütüphanelerin Projeye Dahil Edilmesi
import math
import pandas as pd
import numpy as np
import os
from pandas import DataFrame
from sklearn.cluster import KMeans
from sklearn import preprocessing
import matplotlib.pyplot as plt
from matplotlib import style
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets
style.use('ggplot')
import matplotlib.pylab as plt
from matplotlib.pylab import rcParams
from pandas import read_csv
from datetime import datetime
from matplotlib import pyplot
import sklearn
from sklearn.metrics import mean_squared_error
```

```
In [4]: data=pd.read_csv("GlobalLandTemperaturesByCountry.csv",index_col=None)
        data.head(10)
```

Out[4]:

	dt	AverageTemperature	AverageTemperatureUncertainty	Country
0	1743-11-01	4.384	2.294	Åland
1	1743-12-01	NaN	NaN	Åland
2	1744-01-01	NaN	NaN	Åland
3	1744-02-01	NaN	NaN	Åland
4	1744-03-01	NaN	NaN	Åland
5	1744-04-01	1.530	4.680	Åland
6	1744-05-01	6.702	1.789	Åland
7	1744-06-01	11.609	1.577	Åland
8	1744-07-01	15.342	1.410	Åland
9	1744-08-01	NaN	NaN	Åland

- Burada ise csv datası okunup data isimli değişkene atandı bu atama işleminden sonra dataya ait ilk 10 eleman ekrana bastırıldı.

```
In [6]: turkey=data.loc[data['Country']=="Turkey"]
        print(type(turkey))
        turkey.infer_objects()
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[6]:

	dt	AverageTemperature	AverageTemperatureUncertainty	Country
533683	1777-02-01	3.000	3.314	Turkey
533684	1777-03-01	NaN	NaN	Turkey
533685	1777-04-01	7.678	2.553	Turkey
533686	1777-05-01	15.293	3.189	Turkey
533687	1777-06-01	18.661	2.993	Turkey
...	...	...	...	...
536518	2013-05-01	17.522	0.360	Turkey
536519	2013-06-01	20.862	0.240	Turkey
536520	2013-07-01	23.335	0.281	Turkey
536521	2013-08-01	23.839	0.306	Turkey
536522	2013-09-01	NaN	NaN	Turkey

2840 rows × 4 columns

- Kodun bu tarafında datada sadece Türkiyeye ait olan kısımlar data içerisinden filtrelendi ve turkey isimli değişkenin içerisine atandı ve ekrana objenin tanımlayıcı nesneleri bastırıldı.Bu sayede sadece Türkiyeye ait olan dataları elde etmiş olduk.

```
In [7]: temperature=turkey['AverageTemperature']
        temperature=temperature.dropna(how='any')
```

- Yukarıdaki kod parçasında sıcaklık normalleştirilmiştir.Bunun dışında dropna methodu sayesinde NA olan dataların satırları ve sütünları tablodan düşürülmüş.Verilen how=’any’ parametresi yardımıyla da bu datalar içerisinde any olan datalarında düşürülmesi sağlanmıştır.

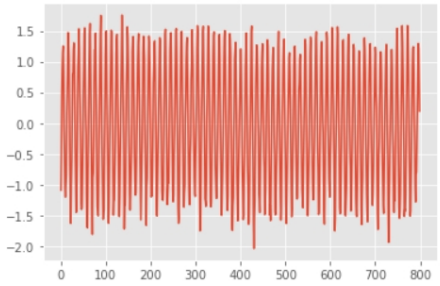
```
In [8]: scale=preprocessing.scale(temperature)
        plt.plot(scale)
        scale
```

```
Out[8]: array([-1.08922851, -0.49890571,  0.46204087, ...,  1.1648001 ,
                1.47687113,  1.54047153])
```

- Yukarıda yer alan kod parçasında sıcaklık değerleri ön işleme alınmış ve görselleştirilmiştir.Ve görselde görüldüğü gibi bir model elde edilmiştir.

```
In [13]: df=DataFrame(scale)
         sample=df[0:800]
         sample
         plt.plot(sample)
```

```
Out[13]: [<matplotlib.lines.Line2D at 0x283810dbfa0>]
```



- Kod parçasının yaptığı işlemi kısaca özetlemek gerekirse daha önceden ön işlem sonucunda elde edilen data üzerinden 0 dahil 800 dahil olmamak üzere ilk 800 data seçilmiştir.Bu dataların görselleştirmesi pyplot yardımıyla yapılmış ve bu şekilde bir görsel elde edilmiştir.

2.1)SÜREKLİLİK MODEL TAHMİNİ

Süreklilik model tahmini doğrusal artan bir zaman serisi için iyi bir temel tahmin yöntemidir.Süreklilik model tahmini bir önceki zaman adımındaki (t-1) gözlemin,mevcut zaman adımındaki (t) tahmin etmek için kullanıldığı yöntemdir.Bunu hava durumu için ileriye dönük tahminlerde geçmiş verileri kullanarak uygulayabiliriz.

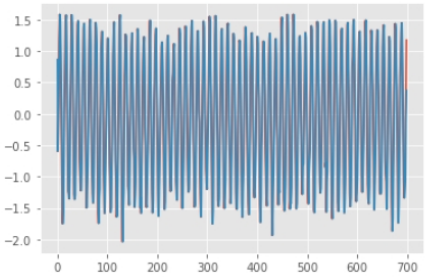
```
In [19]: X=sample.values
         X
         train, test = X[0:700], X[700:]
```

- Burada Turkiyeye ait tutulan ilk 1000 hava sıcaklığı verisinden (daha önceden 800 olarak alınan data 1000 olarak güncellenmiştir.) ilk 700 ü train geriye kalan 300 data ise test için kullanılmıştır.Burada bu şekilde bölünmesinin nedeni %70-%30 oranın yakalanmak istenmesidir.(Yapılan araştırmalar sonucu önerilen bölünme şekli.)

```
In [25]: history = [x for x in train]
         predictions = list()
         for i in range(len(test)):
          predictions.append(history[-1])
          history.append(test[i])

         rmse = math.sqrt(mean_squared_error(test, predictions))
         print('RMSE: %.3f' % rmse)
         pyplot.plot(test)
         pyplot.plot(predictions)
         pyplot.show()
         rmse

         RMSE: 0.548
```




```
Out[25]: 0.548342202169545
```

- Burada yapılan işlemi açıklamak gerekirse train dataları üzerinde for döngüsü yardımıyla gezilerek tarihi yani geçmiş veriler yardımıyla tahmin dataları oluşturulmuştur.Daha sonradan ise tahmin ve test dataları arasındaki kök ortalama kare hatası yani rmse bulunmuştur.Daha sonrasında ise train ve test dataları pyplot yardımıyla görselleştirilmiştir. Ve ekrana rmse yani kök ortalama kare hatası bastırılmıştır.

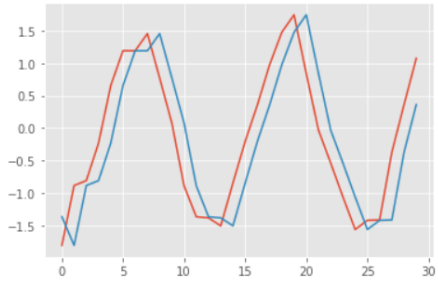
-Rmse değeri için değerlendirme aralığı 0 ile sonsuz aralığındadır.Model rmse değeri 0'a yaklaştıkça hata yapma oranı düşmektedir.

```
In [31]: sample2=sample[0:100]
         X2=sample2.values
         X2
         train2, test2 = X2[0:70], X2[70:]
```

- Kullandığımız datanın daha küçük veri seti ile rmse değerini ölçmek istediğimizden sample üzerinde yer alan ilk 100 datayı aldık bunun sonucunda ise %70-%30 oranı ile datamızı bölerek train ve test datalarımızı oluşturmuş olduk.

```
In [32]: history2 = [x for x in train2]
predictions2 = list()
for i in range(len(test2)):
    predictions2.append(history2[-1])
    history2.append(test2[i])
rmse2 = math.sqrt(mean_squared_error(test2, predictions2))
print('RMSE: %.3f' % rmse2)
pyplot.plot(test2)
pyplot.plot(predictions2)
pyplot.show()
rmse2
```

```
RMSE: 0.607
```




```
Out[32]: 0.6073635673763564
```

- İkinci örneğimize ait model eğitme işlemi tamamlandığında ise görüldüğü üzere küçük data topluluğuyla yapılan eğitim işlemi daha yüksek hata oranı ile karşımıza çıkmaktadır. Buradan yaptığımız çıkarım ise büyük verilerle çalışmanın modelimizi daha iyi eğittiğidir. Modeli eğitecek data seti büyüdükçe hata oranını sıfıra yaklaştırmak daha kolay olur.

3) ÜLKEYE GÖRE HAVA DURUMU İSTATİSTİKLERİ

```
temperatureByCountry=pd.read_csv('GlobalLandTemperaturesByCountry.csv')
countries=temperatureByCountry['Country'].unique()
```

- Şimdi en büyük sıcaklık farkına sahip ilk 20 ülkeyi gösteren bir model oluşturalım. İlk adım olarak csv dosyasını projeye dahil ediyoruz. Önceden oluşturduğumuz dataframeden farklı bir isimde yeni dataframemimizi oluşturuyoruz ve csv dosyasından eşşiz(unique) şekilde ülkeleri bir countries isimli dataya atıyoruz.

```
import seaborn as sns
```

- Bu aşamada görselleştirmek için farklı bir görselleştirme kütüphanesi kullanacağız bu kütüphanenin ismi seaborn.Kütüphane'yi sns ismiyle çağırmak için as işlecini kullanıyoruz.

```
max_min_list=[]
for country in countries:
    current_temps=temperatureByCountry[temperatureByCountry['Country'] ==country]['AverageTemperature']
    max_min_list.append((current_temps.max(), current_temps.min()))
```

- Ülkeleri dataframe üzerinden çektiğimiz data üzerinde her ülke için değerlere ulaşabilmek ve o ülkeye ait max ve min değerleri bulabilmek için gezdik.Bu for döngüsü esnasında max_min_list adında maximum ve minumum değerleri içerisinde barındırıcak bir diziye eklemeler yaptık.
Bunun dışında ise her ülkeye ait max ve min değerler bulunmuş oldu.

```
res_max_min_list = []
res_countries = []
for i in range(len(max_min_list)):
 if not np.isnan(max_min_list[i][0]):
     res_max_min_list.append(max_min_list[i])
     res_countries.append(countries[i])
```

- Bulduğumuz bu değerler üzerinde NaN value datalar var ise bu dataları temizlemek için nan olmayan değerler yeni dizilere atandı ve bu diziler içerisinde tutulmaya başlandı.

```
differences = []
for tpl in res_max_min_list:
 differences.append(tpl[0] - tpl[1])
```

- Ülkelerin sıcaklık farklarını bulmak için differences isimli bir dizi oluşturuldu.Bu dizi içerisine ülkelere ait max ve minmum değerlerin farkları eklendi.
Bu şekilde sıcaklık farkları bulunmuş olundu.

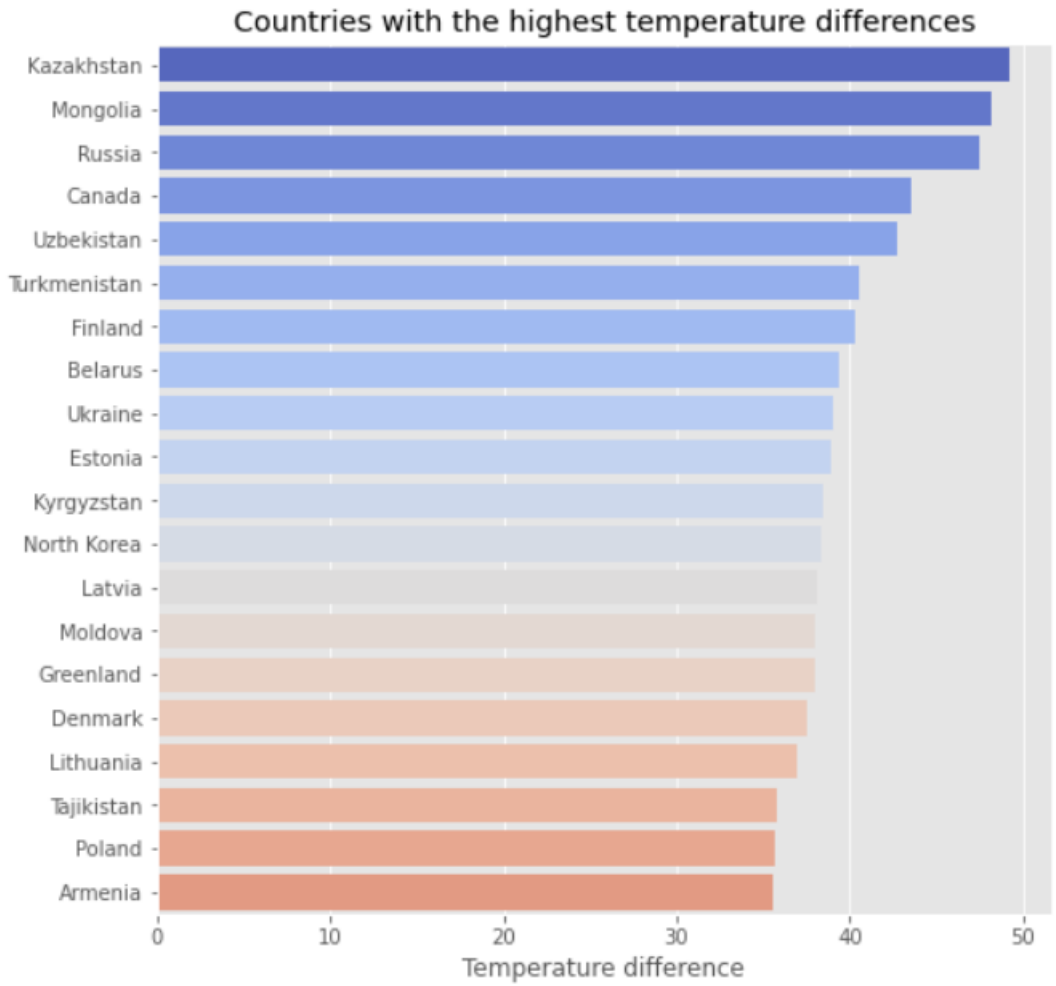
```
differences, res_countries = (list(x) for x in zip(*sorted(zip(differences,
res_countries), key=lambda pair: pair[0], reverse=True)))
```

- Diziler kendi içerisinde eşleştiği için örnek vermek gerekirse res_countries[0]'a ait değer ile differences[0] a ait değerleri arasında key-value ilişkisi vardır.Beraber şekilde sıralandı ve baz olarak sıcaklık farkları değerleri alındı.Görselleştirme esnasında sıcaklık farkı değeri en yüksek ülkenin yukarıda gösterilmesi istendiği için reverse bir şekilde bu sıralama yapıldı.

```
f, ax = plt.subplots(figsize=(8, 8))
sns.barplot(x=differences[:20], y=res_countries[:20],
palette=sns.color_palette("coolwarm", 25), ax=ax)
texts = ax.set(ylabel="", xlabel="Temperature difference", title="Countries with the highest temperature differences")
```

- Ülkelerin değerleri ve isimleri görselleştirmede gözükücek şekilde görselleştirildi.Soğuktan sıcağa doğru gidicek şekilde bir renk paleti seçildi.Bu seçimde sıcaklık farklı yüksek olanlar mavi tonlarda düşük olanlar ise kırmızı tonlara yaklaşıcak şekilde görselleştirildi.
Ve sadece sıralanan bu değerler üzerinden ilk 20 değerin ekrana bastırılması sağlandı.

-Görselleştirme sonucu da aşağıdaki şekilde yer almaktadır.



4)LİNEER REGRESYON İLE ANKARA HAVA DURUMU TAHMİNİ

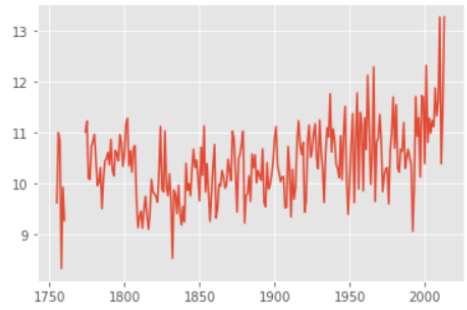
- Bu bölümde farklı bir csv datası ile çalışacağız.Bu data ülkelere ait başkentlerin sıcaklık verilerini içeriyor.Biz tercih olarak Ankarayı seçtik.

```
dframe= pd.read_csv('GlobalLandTemperaturesByMajorCity.csv')
df_an = dframe[dframe['City']=='Ankara']
df_an= df_an.iloc[:, :2]
df_an.head(10)
```

	dt	AverageTemperature
11893	1755-01-01	-3.657
11894	1755-02-01	-2.453
11895	1755-03-01	4.458
11896	1755-04-01	NaN
11897	1755-05-01	NaN
11898	1755-06-01	21.018
11899	1755-07-01	22.099
11900	1755-08-01	20.793
11901	1755-09-01	17.085
11902	1755-10-01	11.088

- Yukarıda yer alan kod parçasında yaptığımız işlemlerden bahsetmek gerekirse.İlk olarak csv dosyasını okuduk,sonrasında ise dataframe içerisinded City değeri Ankaraya eşit olan değerleri getirdik.Bu değerler içersinde sadece kullanıcacğımız kolonları yani ilk iki kolonu iloc yardımı ile tekrardan dataframememiz içerisine atadık.Ve dataları incelemek için ilk 10 elemanını ekrana bastırdık.

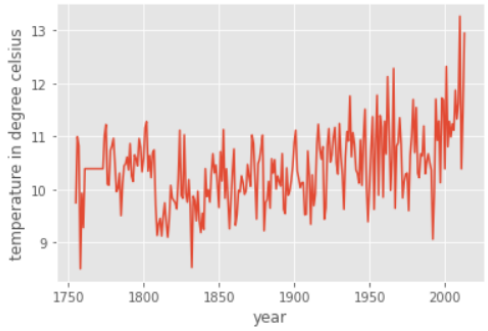
```
a = df_an['dt'].apply(lambda x: int(x[0:4]))
grouped = df_an.groupby(a).mean()
grouped.head(10)
plt.plot(grouped['AverageTemperature'])
plt.show()
```



- Yukarıda yer alan kod parçasında Ankara datalarını tarihlere göre grupladık ve gruplamalar içerisinde yer alan sıcaklıkların ortalamalarını görselleştirmede kullanmak üzere tuttuk.Ve sıcaklık değerleri üzerinden görselleştirdik.

```
df_an['AverageTemperature'] = df_an['AverageTemperature'].fillna(df_an['AverageTemperature'].mean())
grouped = df_an.groupby(a).mean()
plt.plot(grouped['AverageTemperature'])
plt.xlabel('year')
plt.ylabel('temperature in degree celsius')
```

```
Text(0, 0.5, 'temperature in degree celsius')
```



- Yukarıdaki kod parçasında kitap üzerinde Na değerler 0 değeri ile doldurulmasına karşın biz bu değerlerin 0 ile doldurulmasının tahminde hataya yol açacağını düşünmemizden dolayı tabloda yer alan ortalama değerler ile bu kısımları doldurduk.Bu sayede hatadan olabildiğince kaçmaya çalıştık. Bunun dışında bir üst tarafta yaptığımız gibi verileri yıllara göre sınıflandırarak ekranda sonucu görmek üzere görselleştirdik.

```
from sklearn.linear_model import LinearRegression as LinReg
x= grouped.index.values.reshape(-1,1)
y = grouped['AverageTemperature'].values
```

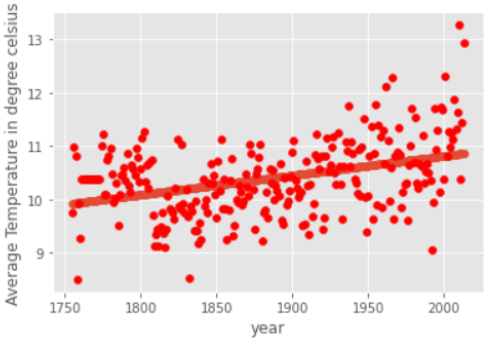
- Yukarıdaki kod parçasında öncelikli olarak sklearn kütüphanesinden LinearRegression paketi projeye dahil edildi.Bunun dışında gruplanmış verilere ait değerler x ve y değişkenlerine atandı.

```
reg = LinReg()
reg.fit(x,y)
y_preds = reg.predict(x)
Accuracy = str(reg.score(x,y))
print(Accuracy)
```

```
0.15253155211068858
```

- Yukarıda yapılan işlemler ise sırasıyla algoritmanın gerekli datalar için çalıştırılması ve doğruluk değerinin bu tahmin işlemi için hesaplanmasıdır.

```
plt.scatter(x=x, y=y_preds)
plt.scatter(x=x,y=y, c='r')
plt.ylabel('Average Temperature in degree celsius')
plt.xlabel('year')
plt.show()
```



- Datalar burada tahmin dataları ile görselleştirilmiştir.Ve ekrana plt yardımı ile çizdirilmiştir.

```
reg.predict([[2024],[2021],[3000]])
```

```
array([10.90303267, 10.89213733, 14.44764686])
```

- Yukarıdaki kod parçası üzerinde çeşitli yıllar için Sıcaklık tahminleri yapılmıştır.

SON