BM478 - Python İle Veri Bilimine Giriş Final Projesi

Ad: Samet Özer

Numara: 201001058

Normal Öğretim

Veri Seti Linki: https://www.kaggle.com/datasets/daniboy370/boston-marathon-2019

Veri Seti: Boston Marathon 2019

Kullandığım veri seti Boston Marathon 2019'a katılan ve bu maratonu tamamlayabilen insanların bilgilerini içermektedir.

Veri setinde 8 sütun bulunmaktadır bu sütunlar:

1. Rank_Tot: Maratonu tamamlayan kişinin genel sıralaması.
2. Age: Maratonu tamamlayan kişinin yaşı.
3. Gender: Maratonu tamamlayan kişinin cinsiyeti.
4. Country: Maratonu tamamlayan kişinin ülkesi.
5. Result_hr: Maratonu tamamlayan kişinin saat:dakika:saniye cinsinden bitirme süresi.
6. Result_sec: Maratonu tamamlayan kişinin saniye cinsinden bitirme süresi.
7. Rank_Gender: Maratonu tamamlayan kişinin kendi cinsiyeti içindeki sıralaması.
8. Country_code: Maratonu tamamlayan kişinin ülke kodu.

1. Verilerin İncelenmesi

Veri analizine başlamadan önce verilere bir göz gezdirelim.

1.1 Verilerin Tanımlanması

1. Rank_Tot: sayısal - oran
2. Age: sayısal - oran
3. Gender: kategorik - nominal
4. Country: kategorik - nominal
5. Result_hr: sayısal - oran
6. Result_sec: sayısal - oran
7. Rank_Gender: sayısal - oran
8. Country_code: kategorik - nominal

1.2 Veriye İlk Bakış

Pandas kütüphanesini kullanarak csv formatındaki veri seti dosyasını okuyup inceleyelim.

In [1]:
```
import pandas as pd
import numpy as np
data = pd.read_csv("Dataset-Boston-2019.csv", sep = ",")
```

Veri setinin ilk 10 verisine bakalım.

In [2]:
```
data.head(10)
```

Out[2]:

	Rank_Tot	Age	Gender	Country	Result_hr	Result_sec	Rank_Gender	Country_code
0	1	30	M	Kenya	2:07:57	7677	1	KEN
1	2	29	M	Ethiopia	2:07:59	7679	2	ETH
2	3	34	M	Kenya	2:08:07	7687	3	KEN
3	4	32	M	Kenya	2:08:54	7734	4	KEN
4	5	26	M	Kenya	2:08:55	7735	5	KEN
5	6	28	M	Kenya	2:08:57	7737	6	KEN
6	7	27	M	United States	2:09:09	7749	7	USA
7	8	30	M	United States	2:09:25	7765	8	USA
8	9	24	M	Kenya	2:09:25	7765	9	KEN
9	10	28	M	Kenya	2:09:53	7793	10	KEN

Veri setinin son 10 verisine bakalım.

In [3]:
```
data.tail(10)
```

Out[3]:

	Rank_Tot	Age	Gender	Country	Result_hr	Result_sec	Rank_Gender	Country_code
26641	26632	52	F	Indonesia	6:16:20	22580	11972	INA
26642	26634	32	F	United States	6:16:21	22581	11973	USA
26643	26635	54	F	United States	6:16:40	22600	11974	USA
26644	26636	28	F	United States	6:16:56	22616	11975	USA
26645	26637	33	F	United States	6:18:13	22693	11976	USA
26646	26640	44	F	United States	6:21:19	22879	11977	USA
26647	26641	50	F	United States	6:22:27	22947	11978	USA
26648	26648	32	F	United States	6:35:50	23750	11979	USA
26649	26650	55	F	United States	6:53:38	24818	11980	USA
26650	26652	46	F	United States	6:59:57	25197	11981	USA

Veri setinin infosuna bakalım.

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26651 entries, 0 to 26650
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Rank_Tot      26651 non-null  int64 
 1   Age           26651 non-null  int64 
 2   Gender        26651 non-null  object
 3   Country       26651 non-null  object
 4   Result_hr     26651 non-null  object
 5   Result_sec    26651 non-null  int64 
 6   Rank_Gender   26651 non-null  int64 
 7   Country_code  26651 non-null  object
dtypes: int64(4), object(4)
memory usage: 1.6+ MB
```

Veri setindeki sütunların unique değerlerini inceleyelim.

In [5]:
```
unique_degerler = data.Age.unique()
unique_degerler.sort()
print(unique_degerler)
```

```
[18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65
 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83]
```

18 ve 83 aralığında her yaştan insan maratona katılmıştır.

In [6]:
```
unique_degerler = data.Gender.unique()
unique_degerler.sort()
print(unique_degerler)
```

```
['F' 'M']
```

Maratonda hem kadınlar ve erkekler bulunmaktadır.

In [7]:
```
unique_degerler = data.Country.unique()
unique_degerler.sort()
print(unique_degerler)
```

```
['Afghanistan' 'Albania' 'Algeria' 'Andorra' 'Antigua and Barbuda'
 'Argentina' 'Australia' 'Austria' 'Barbados' 'Belarus' 'Belgium'
 'Bermuda' 'Bhutan' 'Bolivia' 'Bosnia and Herzegovina' 'Brazil' 'Bulgaria'
 'Cambodia' 'Canada' 'Cayman Islands' 'Chile' 'China' 'Colombia'
 'Costa Rica' 'Croatia' 'Cuba' 'Cyprus' 'Czech Republic' 'Denmark'
 'Dominican Republic' 'Ecuador' 'Egypt' 'El Salvador' 'Eritrea' 'Estonia'
 'Ethiopia' 'Finland' 'France' 'Georgia' 'Germany' 'Greece' 'Guatemala'
 'Guyana' 'Honduras' 'Hong Kong' 'Hungary' 'Iceland' 'India' 'Indonesia'
 'Ireland' 'Israel' 'Italy' 'Jamaica' 'Japan' 'Jordan' 'Kazakhstan'
 'Kenya' 'Korea, Republic of' 'Kuwait' 'Latvia' 'Lebanon' 'Lithuania'
 'Luxembourg' 'Macao' 'Malaysia' 'Malta' 'Mexico' 'Moldova' 'Monaco'
 'Morocco' 'Namibia' 'Nepal' 'Netherlands' 'New Zealand' 'Nicaragua'
 'Norway' 'Pakistan' 'Palestine' 'Panama' 'Paraguay' 'Peru' 'Philippines'
 'Poland' 'Portugal' 'Puerto Rico' 'Romania' 'Russia'
 'Serbia and Montenegro' 'Singapore' 'Slovakia' 'Slovenia' 'South Africa'
 'Spain' 'Sri Lanka' 'Sweden' 'Switzerland' 'Taiwan' 'Thailand'
 'Trinidad and Tobago' 'Turkey' 'Uganda' 'Ukraine' 'United Arab Emirates'
 'United Kingdom' 'United States' 'Uruguay' 'Uzbekistan' 'Venezuela'
 'Vietnam']
```

Maratona farklı ülkelerden katılan birsürü insan vardır.

Describe ile veri setinin özet analizine göz gözdirelim.

In [8]:
```
data.describe()
```

Out[8]:

	Rank_Tot	Age	Result_sec	Rank_Gender
count	26651.000000	26651.000000	26651.000000	26651.000000
mean	13326.800946	42.799895	13980.057184	6731.454242
std	7693.901503	11.537523	2697.850346	3962.252174
min	1.000000	18.000000	7677.000000	1.000000
25%	6664.500000	34.000000	12025.500000	3332.000000
50%	13327.000000	43.000000	13534.000000	6664.000000
75%	19989.500000	51.000000	15538.500000	9995.000000
max	26652.000000	83.000000	25197.000000	14671.000000

2. Analiz Öncesi Ön İşleme

Analize başlamadan önce verilerdeki gerekli düzenlemeleri yapalım.

2.1 Sütunların Düzenlenmesi

Veri setimde aynı değerleri taşıyan 2 farklı sütun vardır.

Bunlar Country, Country_code ve Result_hr, Result_sec sütunları.

Bu sutünlar aynı değerlerin farklı versiyonlarını taşısada veri setimden silmeyeceğim çünkü veri setini görselleştirirken farklı gösterimlerin yararlı olabileceklerini düşünüyorum.

2.2 Veri Tiplerinin Düzenlenmesi

Veri tiplerine tekrar bakalım.

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26651 entries, 0 to 26650
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Rank_Tot      26651 non-null  int64 
 1   Age           26651 non-null  int64 
 2   Gender        26651 non-null  object
 3   Country       26651 non-null  object
 4   Result_hr     26651 non-null  object
 5   Result_sec    26651 non-null  int64 
 6   Rank_Gender   26651 non-null  int64 
 7   Country_code  26651 non-null  object
dtypes: int64(4), object(4)
memory usage: 1.6+ MB
```

Object tipinde görünen Gender, Country ve Country_code sütunlarını Kategori tipine dönüştüreceğim. Object tipinde görünen Result_hr sütununu Datetime object tipine dönüştüreceğim.

In [10]:
```
data.Gender = pd.Categorical(data.Gender)
data.Country = pd.Categorical(data.Country)
data.Country_code = pd.Categorical(data.Country_code)
data.Result_hr = pd.to_datetime(data.Result_hr, format= '%H:%M:%S').dt.time
data.info()
```

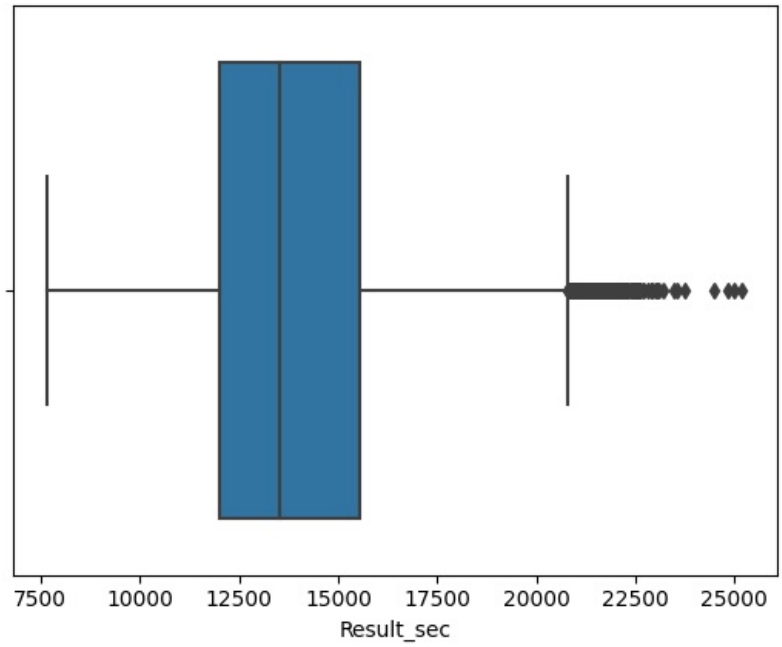
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26651 entries, 0 to 26650
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype   
---  ------        --------------  -----   
 0   Rank_Tot      26651 non-null  int64   
 1   Age           26651 non-null  int64   
 2   Gender        26651 non-null  category
 3   Country       26651 non-null  category
 4   Result_hr     26651 non-null  object  
 5   Result_sec    26651 non-null  int64   
 6   Rank_Gender   26651 non-null  int64   
 7   Country_code  26651 non-null  category
dtypes: category(3), int64(4), object(1)
memory usage: 1.1+ MB
```

2.3 Aykırı Değerler

Veri setindeki aykırı değerleri inceleyelim.

In [11]:
```
import seaborn as sns

data_table = data["Result_sec"]
```

In [12]:
```
sns.boxplot(x = data_table);
```



In [13]:
```
Q1 = data_table.quantile(0.25)
Q3 = data_table.quantile(0.75)
IQR = Q3-Q1
print(Q1)
print(Q3)
```

```
print(IQR)
```

```
12025.5
15538.5
3513.0
```

In [14]:
```
alt_sinir = Q1- 1.5*IQR
ust_sinir = Q3 + 1.5*IQR
print(alt_sinir)
print(ust_sinir)
```

```
6756.0
20808.0
```

In [15]:
```
(data_table < alt_sinir) | (data_table > ust_sinir)
```

Out[15]:
```
0        False
1        False
2        False
3        False
4        False
         ...
26646     True
26647     True
26648     True
26649     True
26650     True
Name: Result_sec, Length: 26651, dtype: bool
```

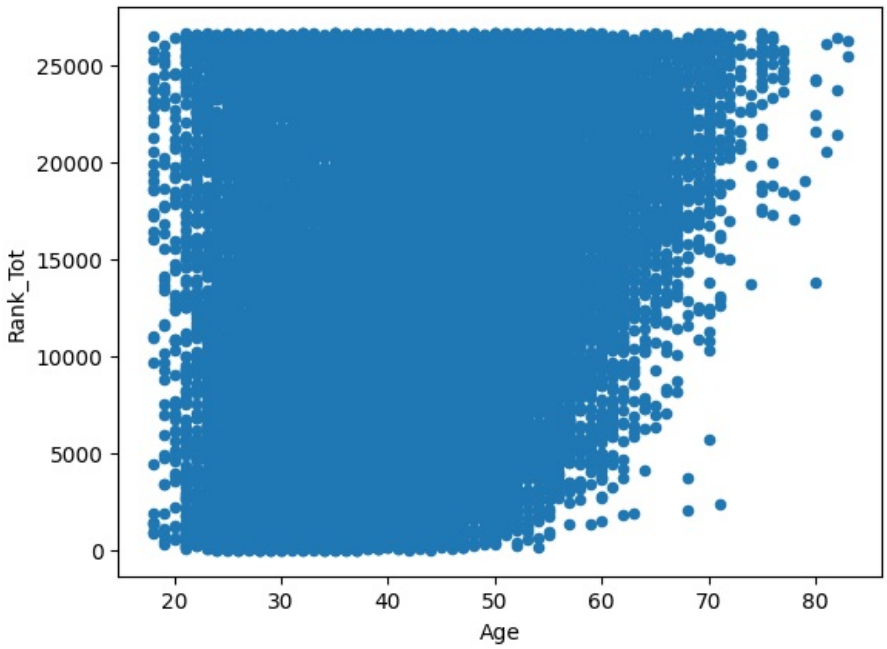
In [16]:
```
aykiri_data = (data_table < alt_sinir) | (data_table > ust_sinir)
data_table[aykiri_data].index
```

Out[16]:
```
Index([14455, 14456, 14457, 14458, 14459, 14460, 14461, 14462, 14463, 14464,
       ...
       26641, 26642, 26643, 26644, 26645, 26646, 26647, 26648, 26649, 26650],
      dtype='int64', length=497)
```

Görüldüğü üzere aykırı değerler vardır bunlara birde grafik üzerinden bakalım

In [17]:
```
from matplotlib import pyplot as plt
data.plot.scatter("Age","Rank_Tot")
```

Out[17]:
```
<Axes: xlabel='Age', ylabel='Rank_Tot'>
```



Grafikte de görüldüğü üzere topluluktan ayrı düşen noktalar vardır.

Veri setinde aykırı değerler olmasına rağmen ben bunları silmeyeceğim veya değiştirmeyeceğim çünkü aykırı değer içeren veriyi sildiğimde veya değiştirdiğimde maratona katılan kişilerin sıralaması değişecektir,çakışmalar olacaktır ve sıralamada boşluklar oluşacaktır bu da veri setinin bütünlüğünü bozacaktır.

2.4 Eksik Veri Analizi

Veri setinde eksik değer var mı diye bakalım.

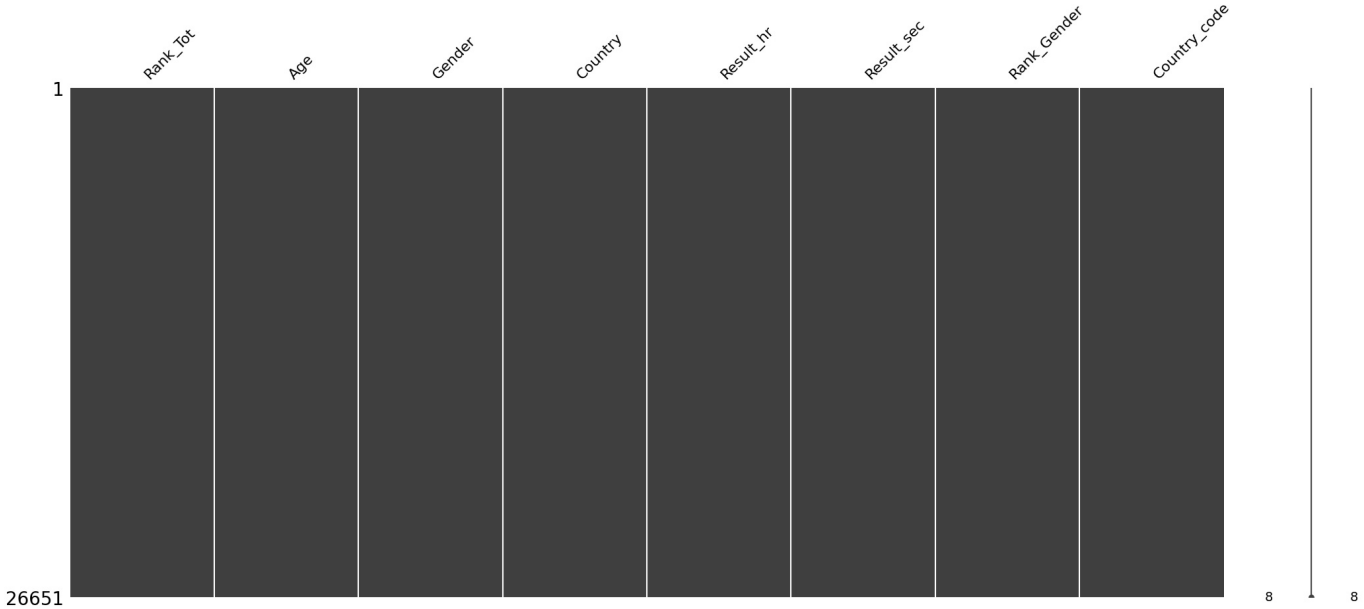
In [18]:
```
#eksik veri varmı?
data.isnull().values.any()
```

Out[18]: False

In [19]:
```
#sütunlardaki eksik veri sayısı
data.isnull().sum()
```

Out[19]:
```
Rank_Tot        0
Age             0
Gender          0
Country         0
Result_hr       0
Result_sec      0
Rank_Gender     0
Country_code    0
dtype: int64
```

In [20]:
```
#sütunlara grafik üzerinden bakalım
import missingno as msno
msno.matrix(data);
```



Görüldüğü üzere veri setinde hiç eksik veri yok bu yüzden bu konuda bir düzenlemeye gerek yoktur.

Sonuç: Veri setinin bütünlüğünün bozulmaması için herhangi bir değerde değişikliğe sebep olacak işlemleri uygulamadım. Sadece sütunların veri tiplerinde gerekli düzenlemeleri yaptım.

3. Verilerin Analizi

Veri setini öncelikle sayısal olarak analiz edelim sonrasında grafikler üzerinden inceleyeceğiz.

In [21]:
```
data.shape[0] + 1
#indis değeri 0 dan başladığı için 1 ile topluyoruz
```

Out[21]: 26652

Veri setinde maratonu tamamlayan toplam 26652 kişinin verileri bulunmaktadır.

In [22]:
```
data.count()
```

Out[22]:
```
Rank_Tot        26651
Age             26651
Gender          26651
Country         26651
Result_hr       26651
Result_sec      26651
Rank_Gender     26651
Country_code    26651
dtype: int64
```

Eksik veri olmadığı için tüm sütunlarda aynı sayıda veri bulunmaktadır.

In [23]:
```
data[['Age', 'Result_sec']].min()
```

Out[23]:
```
Age            18
Result_sec    7677
dtype: int64
```

In [24]:
```
data[['Age', 'Result_sec']].max()
```

In [24]:
```
data[['Age', 'Result_sec']].max()
```

Out[24]:
```
Age              83
Result_sec    25197
dtype: int64
```

Maratona katılan en küçük kişi 18 yaşındadır.
Maratonu en kısa sürede bitiren kişi 7677 saniyede bitirmiştir.

Maratona katılan en büyük kişi 83 yaşındadır.
Maratonu en uzun sürede bitiren kişi 25197 saniyede bitirmiştir.

In [25]:
```
data[['Age', 'Result_sec']].mean()
```

Out[25]:
```
Age              42.799895
Result_sec    13980.057184
dtype: float64
```

Maratona katılan kişilerin yaş ortalaması 42.79'dur.
Maratonu katılan kişilerin bitirme süresi ortalama 13980 saniyedir.

In [26]:
```
data.groupby("Gender")[['Age', 'Result_sec']].mean()
```

Out[26]:

Gender	Age	Result_sec
F	40.545530	14717.804858
M	44.641036	13377.538105

Grupları kadın erkek olarak ayırırsak, Maratona katılan kadınların ortalama yaşı 40,54, bitirme süresi 14714. Maratona katılan erkeklerin ortalam yaşı 44.64, bitirme süresi 13377.

Bu sonuçlara göre maratona katılan kadınlar erkeklerden daha genç iken erkekler maratonu kadınlardan daha kısa sürede bitirmiştir.

In [27]:
```
data.groupby("Gender")[[ 'Result_sec']].min()
```

Out[27]:

Gender	Result_sec
F	8611
M	7677

Maratonu en kısa sürede bitiren erkek ve kadına bakıldığında erkek kadından 934 saniye daha erken bitirmiştir.

In [28]:
```
data.groupby("Country")[['Result_sec']].mean().sort_values(by='Result_sec',)
```

Out[28]:

Country	Result_sec
Eritrea	7985.000000
Jordan	8805.000000
Ethiopia	9286.000000
Palestine	9663.000000
Kenya	9693.157895
...	...
Cayman Islands	16988.000000
Uganda	18436.000000
Monaco	18462.000000
Indonesia	18592.818182
United Arab Emirates	18882.000000

109 rows × 1 columns

Ülkelerin ortalama bitirme süreleri karşılaştırıldığında maratonu en kısa sürede bitiren katılımcılara sahip ülke Eritre iken en uzun sürede bitiren katılımcılara sahip ülke Birleşik Arap Emirlikleri'dir.

In [29]:
```
data.groupby("Country")[['Age']].mean().sort_values(by='Age',)
```

Out[29]:

Country	Age
Uzbekistan	26.000000
Antigua and Barbuda	26.000000
Palestine	29.000000
Georgia	29.000000
Jordan	30.000000
...	...
Morocco	52.235294
Korea, Republic of	53.064103
Algeria	57.000000
Afghanistan	60.000000
Barbados	67.000000

109 rows × 1 columns

Ülkelerin ortalama yaşlarına bakıldığında en düşük ortalamaya sahip ülke Özbekistan iken en büyük ortalamaya sahip ülke Barbadostur.

In [30]:
```
yas_araliklari = [0, 20, 40, 60, float('inf')]
yas_kategori = pd.cut(data['Age'], bins=yas_araliklari, labels=['0-20', '21-40', '41-60', '61+'], include_lowes
gruplu_veri = data.groupby(yas_kategori)[['Result_sec']]
gruplu_veri.mean()
```

Out[30]:

Age	Result_sec
0-20	14050.672000
21-40	13376.877933
41-60	14213.815737
61+	16086.020574

In [31]:
```
gruplu_veri.min()
```

Out[31]:

Age	Result_sec
0-20	9569
21-40	7677
41-60	8336
61+	10560

Yaş gruplarına göre bir analiz yaptığımızda 21-40 yaş grubu hem ortalamada hemde minimum bitirme süresinde en kısa süreyi elde etmiştir buna göre en başarılı grup 21-40 yaş grubu diyebiliriz.

In [32]:
```
data["Age"].value_counts()
```

Out[32]:
```
Age
45    1053
40     925
46     924
50     865
47     826
      ...
82       3
78       2
81       2
83       2
79       1
Name: count, Length: 66, dtype: int64
```

Maratona en çok 45 yaşında insan bulunurken sadece 1 tane 79 yaşında kişi bulunmaktadır.

In [33]:
```
data["Country"].value_counts()
```

Out[33]:
```
Country
United States     18902
Canada             1753
United Kingdom      705
China               529
Mexico              426
                   ...
Monaco                1
Namibia               1
Nicaragua             1
Pakistan              1
Jordan                1
Name: count, Length: 109, dtype: int64
```

Maratonda ezici bir üstünlükle en çok amerikadan katılan insan vardır.

In [34]:
```
data["Gender"].value_counts()
```

Out[34]:
```
Gender
M    14670
F    11981
Name: count, dtype: int64
```

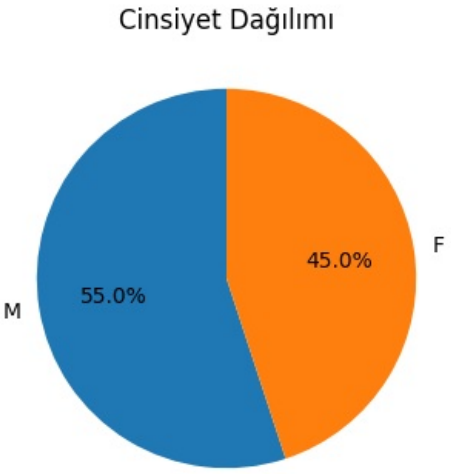
Erkekler kadınlara göre daha çok katılım göstermiştir.

4.Verilerin Görselleştirilmesi

Bir önceki başlıkta verilerin analizini yapmıştık. Şimdi grafikler üzerinden inceleme yapacağız.

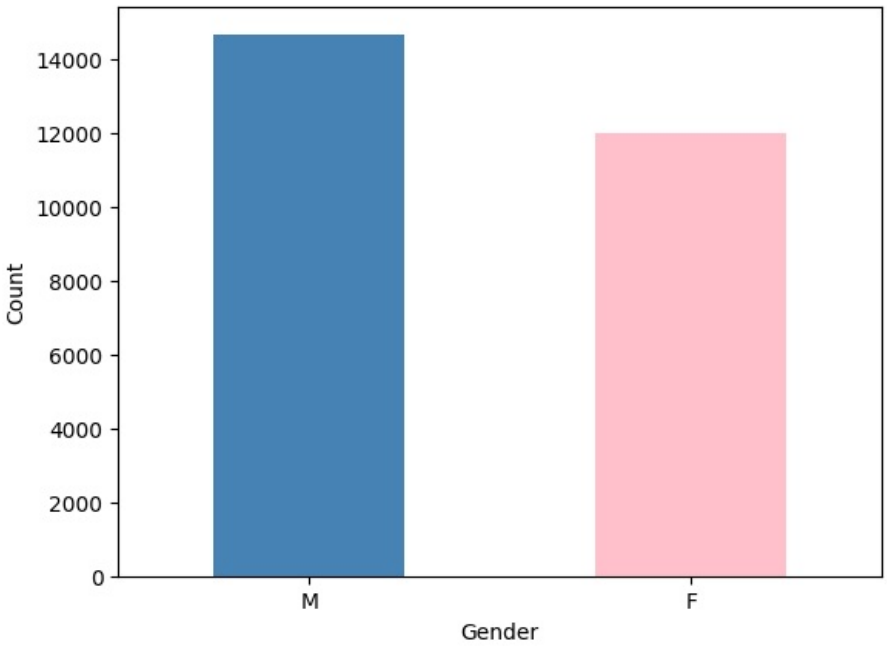
4.1 Cinsiyet Grafikleri

In [35]:
```
cinsiyet_seri = data['Gender'].value_counts()

plt.figure(figsize=(4, 4))
cinsiyet_seri.plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Cinsiyet Dağılımı')
plt.ylabel('')
plt.show()
```



Dağılımı pie grafik üzerinde incelediğimizde maratona erkeklerin daha çok katıldığı görülüyor fakat arada devasa bir fark yok.

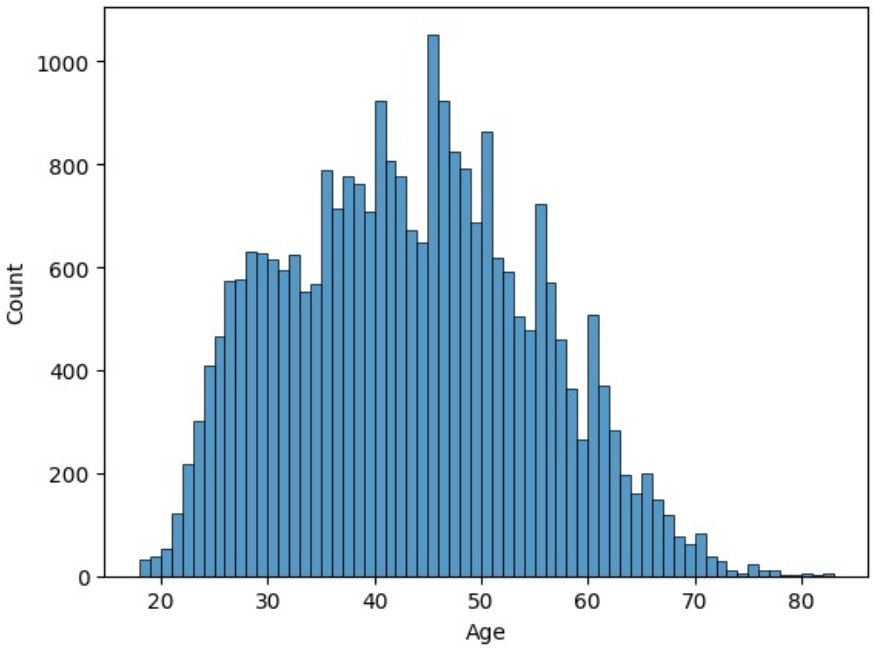
In [36]:
```
colors = ['steelblue','pink']
data['Gender'].value_counts().plot.bar(color=colors)
plt.xlabel('Gender')
plt.ylabel('Count')
plt.tick_params(axis='x', rotation=0)
plt.show()
```



Sütun grafik üzerinden incelediğimizde arada yaklaşık 2500 kişi fark görülmektedir.

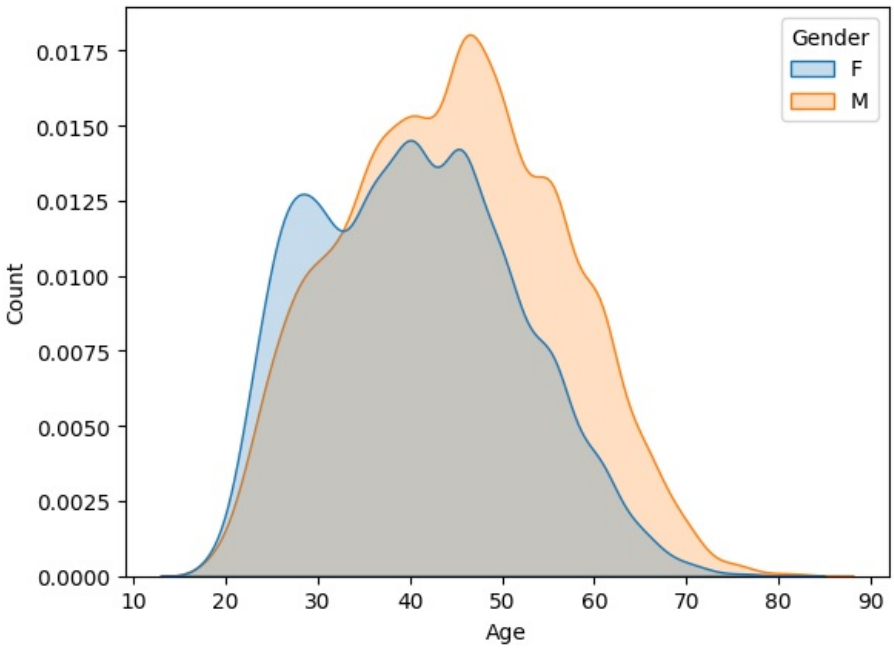
4.2 Yaş Grafikleri

In [37]:
```
sns.histplot(data.Age, bins = 65);
```



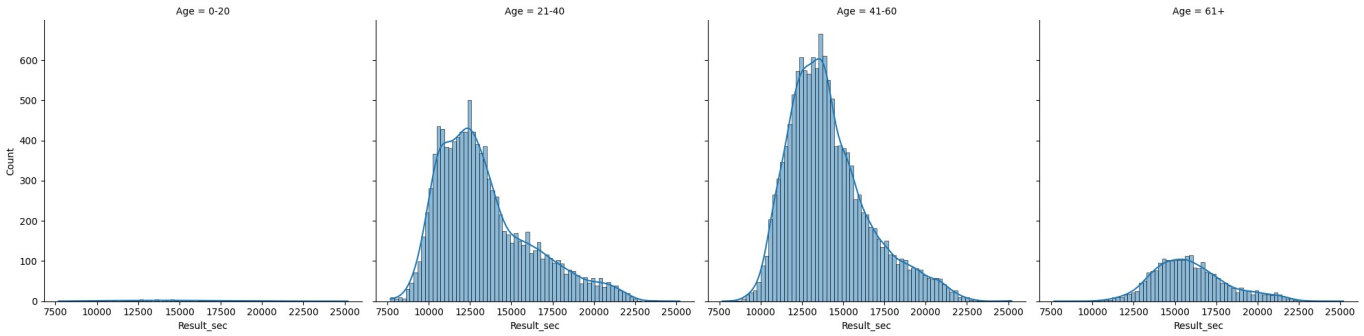
Katılımcıların yaş aralıklarını incelediğimizde 20-30 yaş aralığında 30 yaşa doğru bir artış vardır. En çok 30-50 yaş aralığından katılım olmuştur ve 40-50 yaş aralığında yüksek katılım vardır. Katılım 50-80 aralığında 80 yaşa doğru gittikçe azalmaktadır.

In [38]:
```
sns.kdeplot(data=data, x='Age', hue='Gender', fill=True)
plt.ylabel('Count')
plt.show()
```



Yaş dağılımını yoğunluk grafiği üzerinden cinsiyet ayrımına göre incelediğimizde erkekler 40-50 yaş aralığına kadar artan bir katılım gösterip 50 yaşından sonra 80 yaşa doğru azalan bir katılım göstermiştir. Kadınlar 25 yaşa kadar artan bir katılım gösterirken 25-50 yaş aralığında hafif artan sabit bir katılım göstermektedir. Kadınlarda 50 yaşından sonra azalan bir katılım sergilemektedir. Katılımlar iki cinsiyette de 50 yaşından sonra düşmüştür.

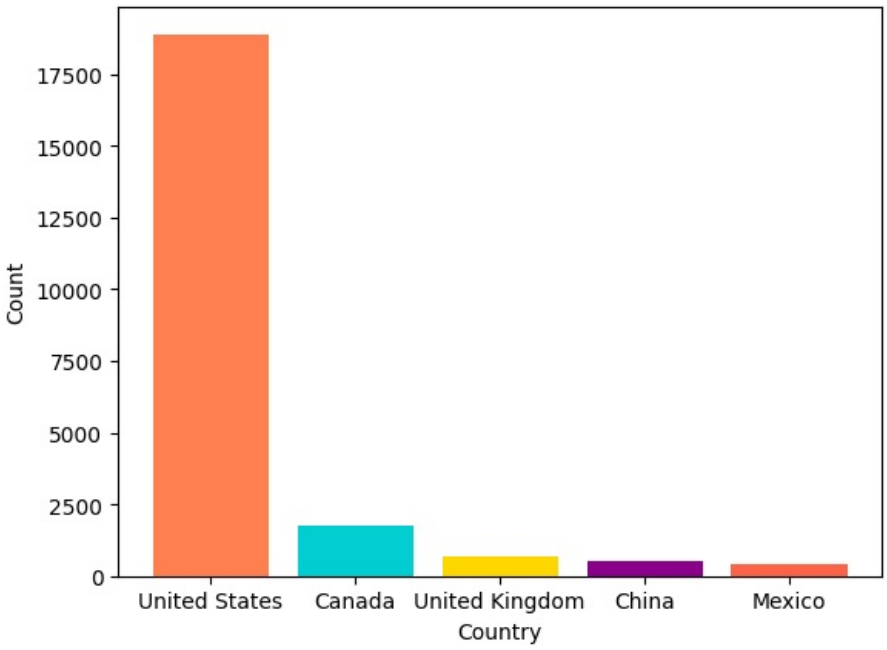
In [39]:
```
yas_aralik = [0, 20, 40, 60, float('inf')]
age_cut = pd.cut(data['Age'], bins=yas_aralik, labels=['0-20', '21-40', '41-60', '61+'], right= False )
sns.displot(data = data, x = 'Result_sec', col = age_cut, kde = True )
plt.show()
```



0-20 yaş aralığında çok az katılımcı olduğundan grafikte yorumlanacak bir veri yoktur. 21-40 yaş aralığında katılımcıların maratonu en çok 12000 saniye civarında bitirdiği görülmektedir,katılımcılar genellikle 10000-15000 saniye aralığında maratonu bitirmiştir. 41-60 yaş aralığında katılımcıların maratonu en çok 13000 saniye civarında bitirdiği görülmektedir. 61+ katılımcıların maratonu en çok 15000 civarında bitirdiği görülmektedir. Grafiklere göre tahmin edilebileceği gibi yaş arttıkça bitirme süresi artmaktadır.

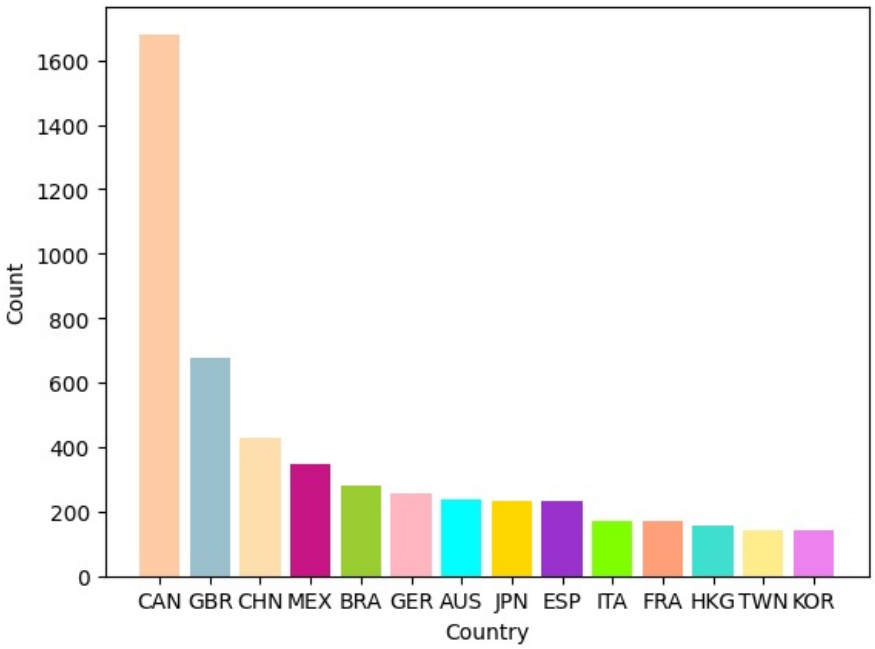
4.3 Ülke Grafikleri

In [40]:
```
ulkeler_data = data['Country'].value_counts().reset_index()
ulkeler_data.columns = ['Country', 'Count']
ulkeler_data5 = ulkeler_data.head(5)

colors5 = ['#FF7F50', '#00CED1', '#FFD700', '#8B008B', '#FF6347']

plt.bar(ulkeler_data5['Country'], ulkeler_data5['Count'], color=colors5)
plt.xlabel('Country')
plt.ylabel('Count')
plt.show()
```



Maraton Amerikada yapıldığı için Amerikadan katılımların ezici bir şekilde fazla olduğunu görüyoruz.

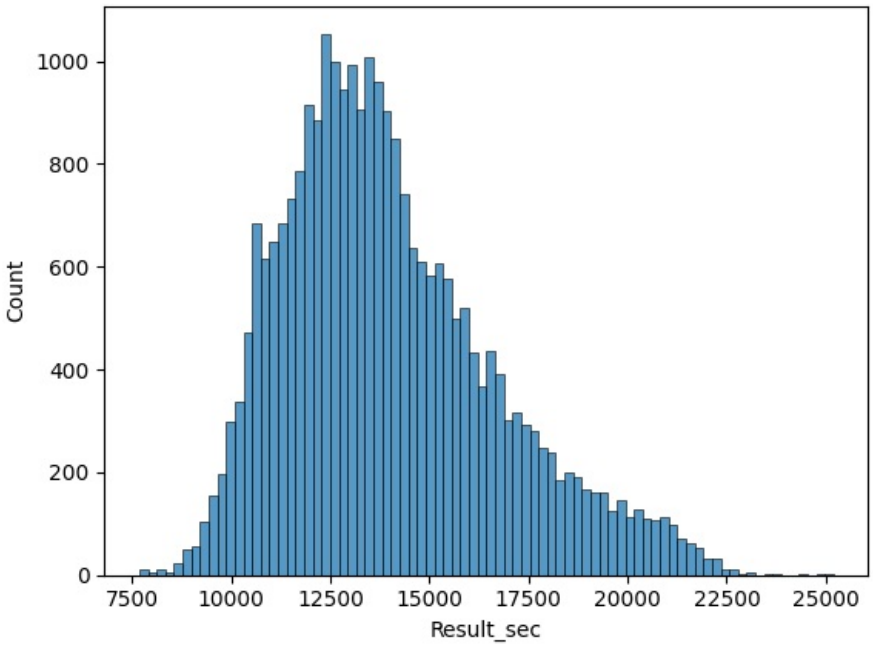
In [41]:
```
ulkeler_data = data['Country_code'].value_counts().reset_index()
ulkeler_data.columns = ['Country_code', 'Count']
ulkeler_data15 = ulkeler_data.head(15)
ulkeler_data15 = ulkeler_data15.iloc[1:]


colors15 = ['#FFCBA4', '#9AC0CD', '#FFDEAD', '#C71585', '#9ACD32',
          '#FFB6C1', '#00FFFF', '#FFD700', '#9932CC', '#7FFF00',
          '#FFA07A', '#40E0D0', '#FFEC8B', '#EE82EE', '#00FF7F']

plt.bar(ulkeler_data15['Country_code'], ulkeler_data15['Count'], color=colors15)
plt.xlabel('Country')
plt.ylabel('Count')
plt.show()
```



USA dışında katılım gösteren katılımcıların grafiğini inceleyelim çünkü USA'yı dahil ettiğimizde grafikte yorumlanmayacak kadar küçük kalıyorlar. Amerikadan sonra en çok Kanada, İngiltere, Çin , Meksika ve Brezilyadan olmuştur. Genellikle Amerikaya yakın ülkelerden katılım fazladır.

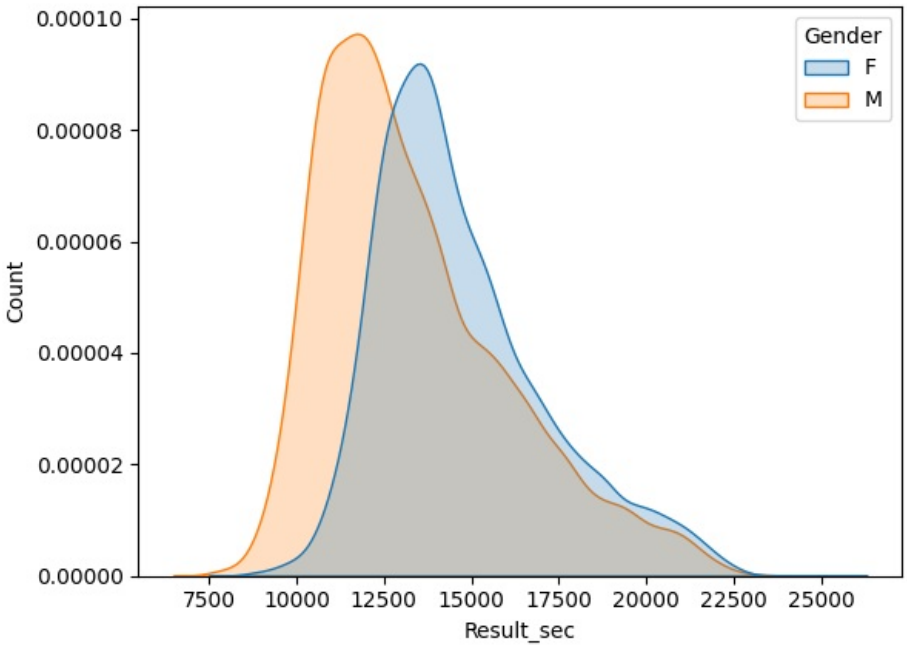
4.4 Maratonu Bitirme Süresi Grafikleri

In [42]:
```
sns.histplot(data.Result_sec, bins = 80);
```



Histogram grafik üzerinden incelediğimizde 12000 saniyeye kadar hızlı bir yükseliş vardır ve katılımcılar en çok 12000 - 14000 saniye aralığında maratonu tamamlamıştır. 14000 saniyeden sonra 22500 saniyeye doğru yavaş bi düşüş vardır.

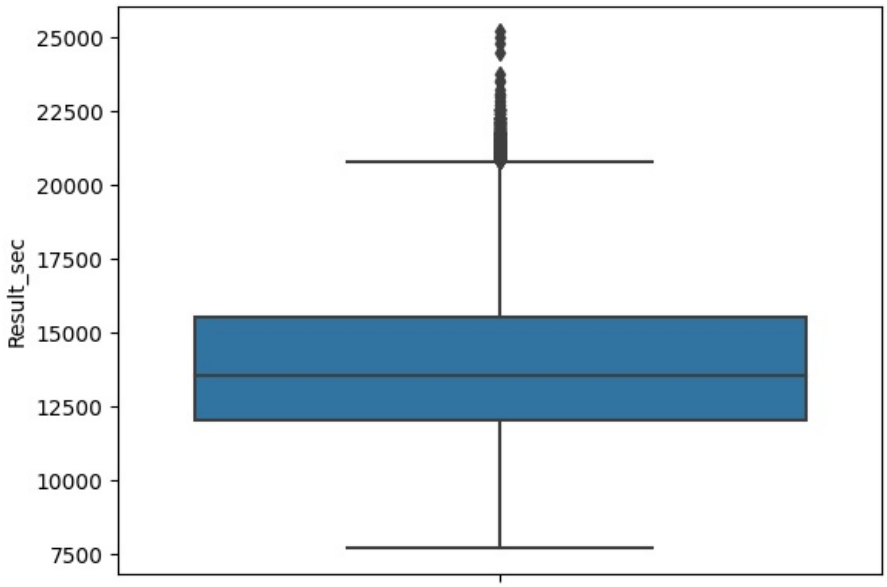
In [43]:
```
sns.kdeplot(data=data, x='Result_sec', hue='Gender', fill=True)
plt.ylabel('Count')
plt.show()
```



Bitirme sürelerini yoğunluk grafiği üzerinden cinsiyet ayrımına göre incelediğimizde erkeklerin maratonu daha kısa sürede tamamladığı görüyoruz. Erkeklerde 12000 saniyeye kadar hızlı bir yükseliş varken sonrasında yavas yavas 22500 e kadar azalmaktadır. Kadınlarda 13500 saniyeye kadar hızlı bir yükseliş varken 22500 saniyeye erkeklere göre daha hızlı inmektedir.

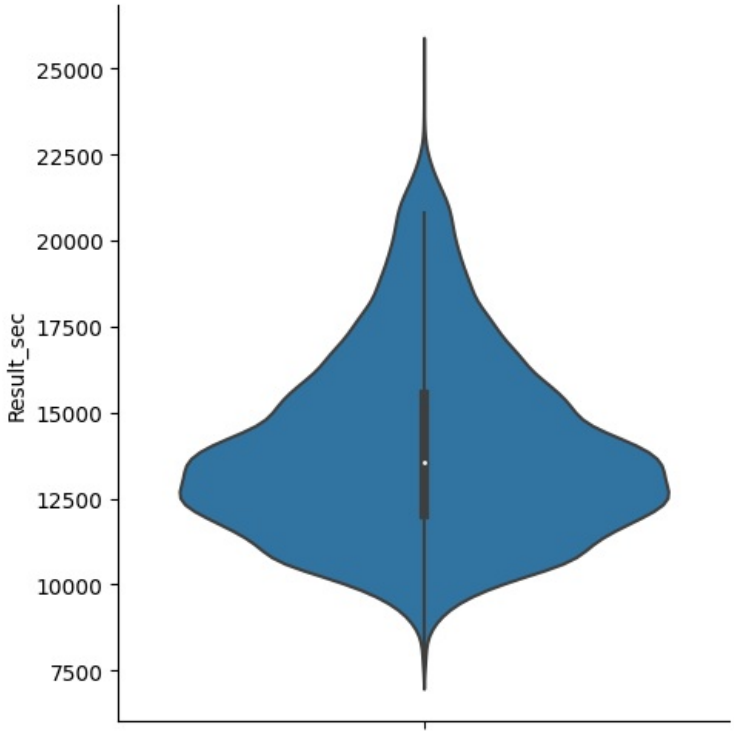
4.5 Kutu ve Violin Grafikler

In [44]:
```
sns.boxplot(y = "Result_sec",  data = data);
```



Bitirme süresinin kutu grafiğini inceleyelim.

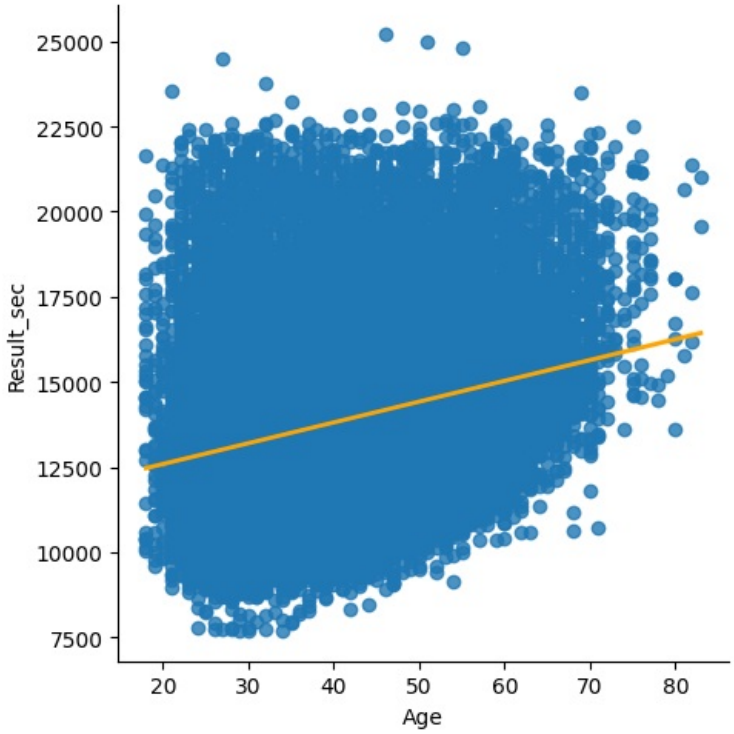
In [45]:
```
sns.catplot(y = "Result_sec", kind = "violin", data = data);
```



Bitirme süresinin violin grafiğini inceleyelim.

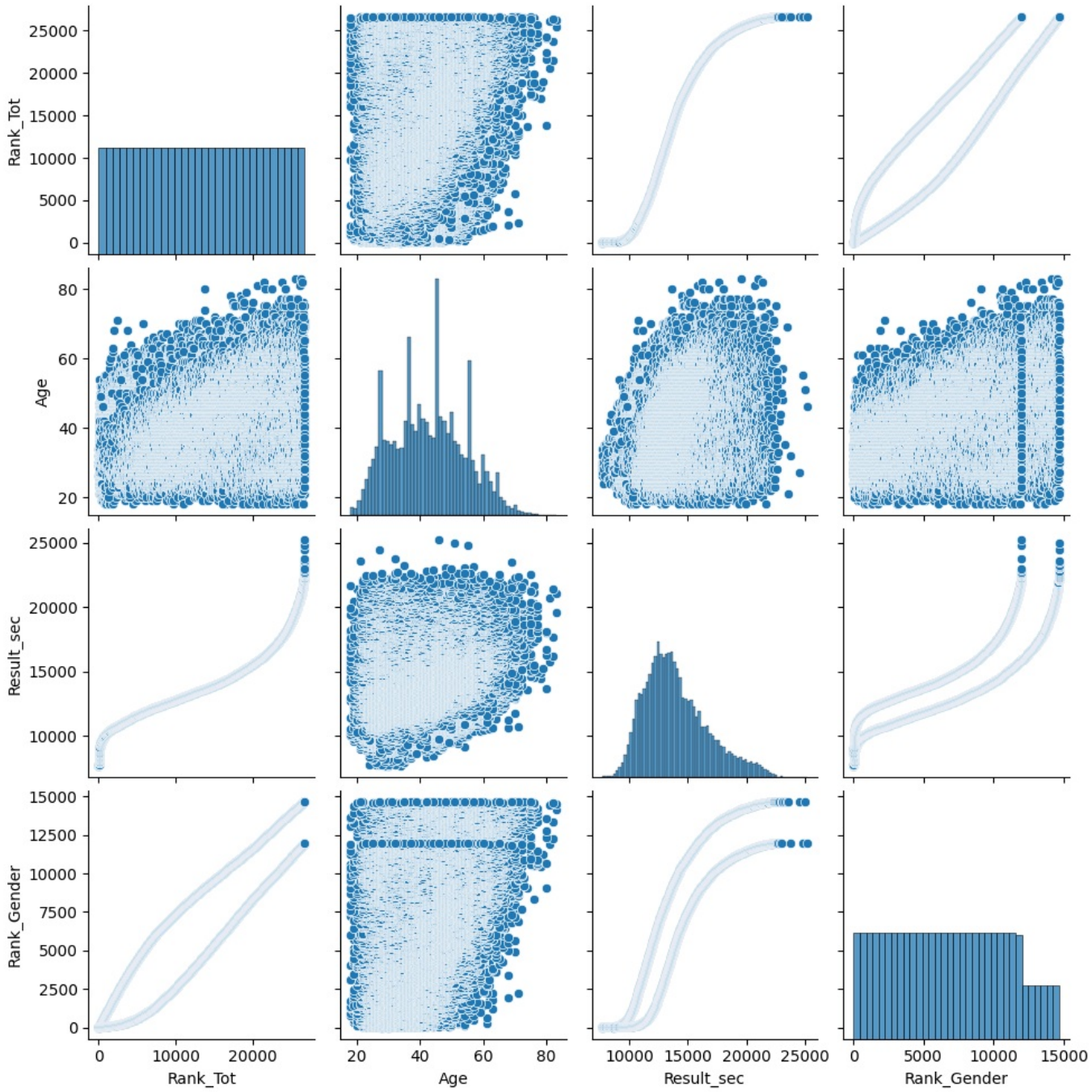
4.6 Kolerasyon Grafikleri

In [46]:
```
sns.lmplot(x = "Age", y = "Result_sec", data = data,line_kws={"color": "orange", "linewidth": 2});
```



Grafiği incelediğimizde yaş ve bitirme süresi arasında pozitif yönlü zayıf kolerasyon olduğunu görüyoruz.

In [47]:
```
sns.pairplot(data);
```

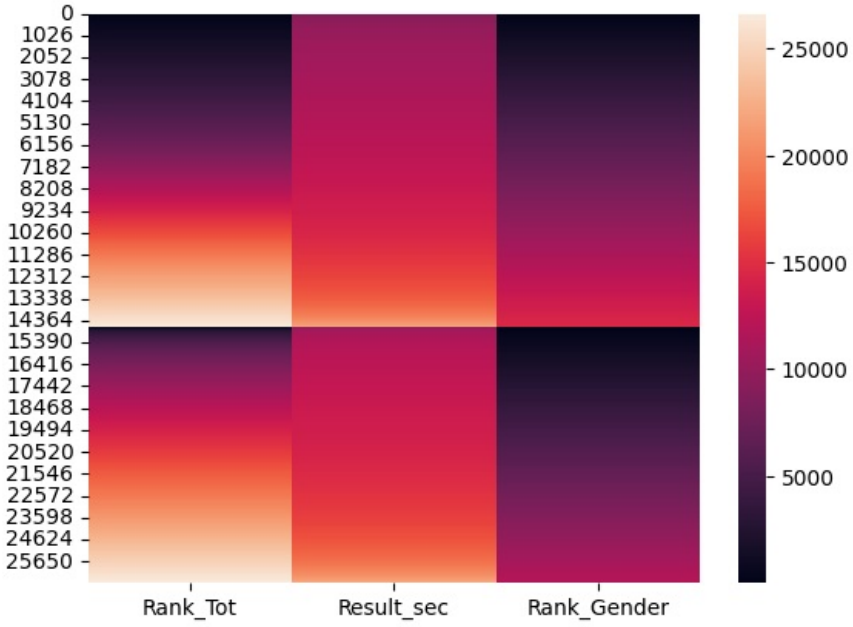


Veri setinin scatter plot matrisini inceleyelim.

4.7 Isı Haritası

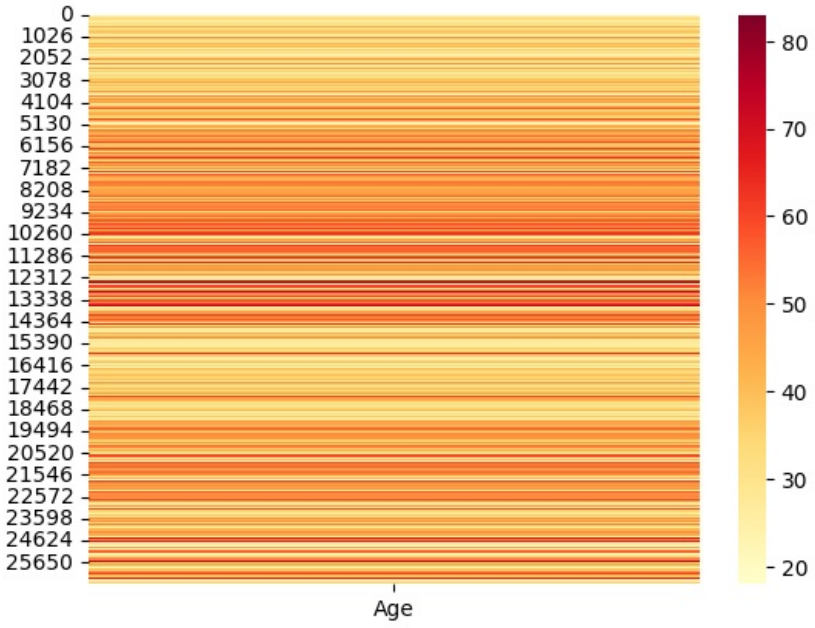
In [48]:
```
int_columns = data.select_dtypes(include=[int]).columns
int_columns = int_columns.drop('Age')
int_data = data[int_columns]
sns.heatmap(int_data)
```

Out[48]: <Axes: >



In [49]:
```
sns.heatmap(data[['Age']], cmap='YlOrRd')
```

Out[49]: <Axes: >

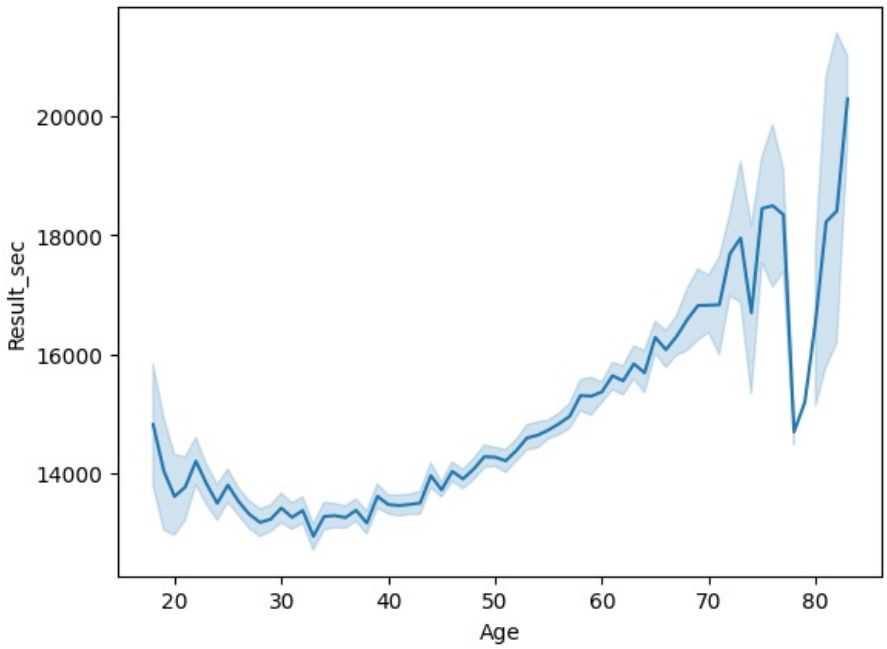


Veri setindeki sayısal sütunların ısı haritasını inceleyelim.

4.8 Çizgi Grafikleri

In [50]:
```
sns.lineplot(x = "Age", y = "Result_sec", data = data)
```

Out[50]: <Axes: xlabel='Age', ylabel='Result_sec'>



Yaş ve Bitirme süresinin çizgi grafiğini incelediğimizde 20-30 yaş aralığında yaş arttıkça bitirme süresi çok hafif bir şekilde düşsede 30-80 yaş aralığında yaş arttıkça bitirme süresininde arttığı belirgin bir şekilde görülmektedir.

4. Verilerin İstatiksel Analizi

In [51]:
```
import researchpy as rp
from scipy.stats import shapiro
from scipy.stats import spearmanr
from scipy.stats import pearsonr
from scipy.stats import kendalltau
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
from sklearn.linear_model import LinearRegression
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from sklearn.cross_decomposition import PLSRegression, PLSSVD
from sklearn import model_selection
from sklearn.linear_model import Lasso
from sklearn.linear_model import LassoCV
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import ElasticNetCV
from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV
```

4.1 Betimsel İstatistik

Sayısal sütunların istatiksel özetine bakalım

In [52]:
```
age_series = data['Age']
result_sec_series = data['Result_sec']

summary = rp.summary_cont(age_series)
print(summary)
summary = rp.summary_cont(result_sec_series)
print(summary)
```

```
  Variable        N     Mean       SD      SE  95% Conf.  Interval
0      Age  26651.0  42.7999  11.5375  0.0707    42.6614   42.9384


     Variable        N        Mean         SD       SE   95% Conf.    Interval
0  Result_sec  26651.0  13980.0572  2697.8503  16.5257  13947.6658  14012.4485
```

Güven aralığını hesaplayalım.

In [53]:
```
a=(1.96*(2697.85034/pow(26651,1/2)))
```

```
a
b=13980.0572-a
c=13980.0572+a
print(b)
print(c)
```

```
13947.666735899857
14012.447664100142
```

Maratonun bitirme süreleri için güven aralığı 13947 - 14012'dir.

Yaş ve Maratonu bitirme süresi arasındaki kovaryansa bakalım.

In [54]:
```
data[["Age","Result_sec"]].cov()
```

Out[54]:

	Age	Result_sec
Age	133.114441	8.124637e+03
Result_sec	8124.637334	7.278396e+06

Yaş ve Maratonu bitirme süresi arasındaki korelasyona bakalım.

In [55]:
```
data[["Age","Result_sec"]].corr()
```

Out[55]:

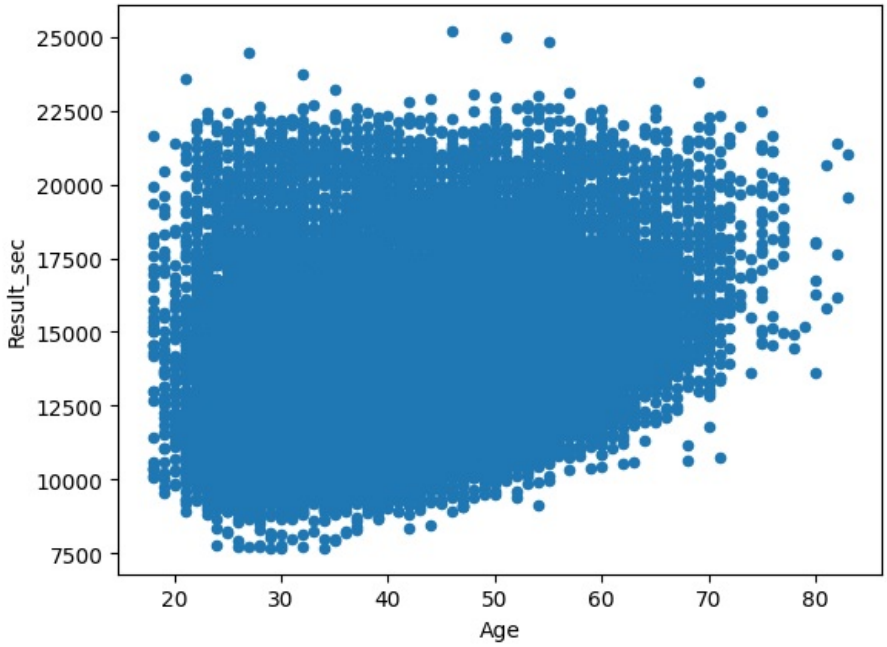
	Age	Result_sec
Age	1.00000	0.26102
Result_sec	0.26102	1.00000

4.2 Korelasyon Analizi

Öncelikle korelasyonu grafik üzerinden inceleyelim.

In [56]:
```
data.plot.scatter("Age","Result_sec")
```

Out[56]:
```
<Axes: xlabel='Age', ylabel='Result_sec'>
```



Grafikte net bir korelasyon görülmemektir. Grafiğin alt kısımlarında yaş arttıkça maratonu bitirme süresinin arttığı gözlemlenmektedir. Beklenildiği üzere yaşlı insanların maratonu bitirme süresi daha uzundur. Bu, yaş ile maratonu bitirme süresi arasında bir korelasyon olduğu anlamına gelebilir değerleri inceleyelim.

4.2.1 Shapiro-Wilk Testi

İki değişkenin normallik varsayımını sağlayıp sağlamadığına bakalım.

In [57]:
```
test_istatistigi, pvalue = shapiro(data["Age"])
print('Test İstatistiği = %.4f, p-değeri = %.4f' % (test_istatistigi, pvalue))

test_istatistigi, pvalue = shapiro(data["Result_sec"])
print('Test İstatistiği = %.4f, p-değeri = %.4f' % (test_istatistigi, pvalue))
```

```
Test İstatistiği = 0.9868, p-değeri = 0.0000
Test İstatistiği = 0.9613, p-değeri = 0.0000
C:\Users\samet\AppData\Local\Programs\Python\Python311\Lib\site-packages\scipy\stats\_morestats.py:1816: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

İki değişkeninde p değeri 0.05'ten küçük olduğu için normallik varsayımını sağlamamaktadır.
Diğer metodları denememiz gerekir.

4.2.3 Spearman Metodu

Varsayım sağlanmadığı için korelasyon katsayısını hesaplamak için spearman metodunu kullanmamız gerekir.

In [58]:
```
spearman_corr, p_value = spearmanr(data["Age"], data["Result_sec"])
print('Spearman Korelasyon Katsayısı = %.4f, p-değeri = %.4f' % (spearman_corr, p_value))
```

```
Spearman Korelasyon Katsayısı = 0.2946, p-değeri = 0.0000
```

Korelasyon katsayısı 0.2946 çıkmıştır buna göre Age ve Result_sec Değişkenleri arasında pozitif yönde düşük kuvvetli bir korelasyon vardır.

4.2.2 Pearson Metodu

Korelasyonu birde pearson metodu ile inceleyelim.

In [59]:
```
test_istatistigi, pvalue = pearsonr(data["Age"],data["Result_sec"])
print('Korelasyon Katsayısı = %.4f, p-değeri = %.4f' % (test_istatistigi, pvalue))
```

```
Korelasyon Katsayısı = 0.2610, p-değeri = 0.0000
```

4.2.4 Kendall Metodu

Korelasyonu birde kendall metodu ile inceleyelim.

In [60]:
```
corr_coef, p_value = kendalltau(data["Age"], data["Result_sec"])
print('Korelasyon Katsayısı = %.4f, p-değeri = %.4f' % (corr_coef, p_value))
```

```
Korelasyon Katsayısı = 0.2084, p-değeri = 0.0000
```

5. Makine Öğrenmesi

5.1 Doğrusal Regresyon

5.1.1 Basit Doğrusal Regresyon

sklearn ile modelleme

In [61]:
```
X = data[["Age"]]
y = data["Result_sec"]
reg = LinearRegression()
model = reg.fit(X, y)
print(model.intercept_)
print(model.coef_)
```

```
11367.766413504596
[61.0349809]
```

In [62]:
```
y_pred = model.predict(X)
basit_dogrusal_rmse = np.sqrt(mean_squared_error(y, y_pred))
print("RMSE değeri:", basit_dogrusal_rmse)
```

```
RMSE değeri: 2604.2762737604457
```

In [63]:
```
model.predict(X)[0:5]
```

Out[63]:
```
array([13198.81584051, 13137.78085961, 13442.95576411, 13320.88580231,
       12954.67591691])
```

In [64]:
```
y[0:5]
```

Out[64]:
```
0    7677
1    7679
2    7687
3    7734
4    7735
Name: Result_sec, dtype: int64
```

In [65]:
```
print("Model skoru: " + str(model.score(X, y)))
```

```
Model skoru: 0.06813136455618829
```

5.1.2 Çoklu Doğrusal Regresyon

scikit-learn ile modelleme

In [66]:
```
X = data[['Age', 'Gender', 'Country']]
y = data['Result_sec']
X_encoded = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 20, random_state= 42)

lm = LinearRegression()
model = lm.fit(X_train, y_train)

#sabit katsayı
print(model.intercept_)
#bağımsız deşişken katsayıları
print(model.coef_)
```

```
36366005526795.63
[ 7.50587731e+01 -3.63586008e+13 -3.63586008e+13 -7.40469185e+09
 -7.40469274e+09 -7.40469397e+09 -7.40469408e+09 -7.40468846e+09
 -7.40469287e+09 -7.40469311e+09 -7.40469230e+09 -7.40469074e+09
 -7.40469384e+09 -7.40469271e+09 -7.40469341e+09 -7.40469394e+09
 -7.40469102e+09 -7.40469447e+09 -7.40469315e+09 -7.40469356e+09
 -7.40469189e+09 -7.40469331e+09 -7.40469103e+09 -7.40469394e+09
 -7.40469236e+09 -7.40469370e+09 -7.40469372e+09 -7.40469038e+09
 -7.40469287e+09 -7.40469380e+09 -7.40469376e+09 -7.40469260e+09
 -7.40469434e+09 -7.40469287e+09 -7.40469226e+09 -7.40469464e+09
 -7.40469691e+09 -7.40469383e+09 -7.40469643e+09 -7.40469326e+09
 -7.40469291e+09 -7.40469260e+09 -7.40469235e+09 -7.40469140e+09
 -7.40469400e+09 -7.40469510e+09 -7.40469365e+09 -7.40469231e+09
 -7.40469266e+09 -7.40469195e+09 -7.40469123e+09 -7.40468745e+09
 -7.40469260e+09 -7.40469227e+09 -7.40469176e+09 -7.40469358e+09
 -7.40469283e+09 -7.40469601e+09 -7.40469490e+09 -7.40469616e+09
 -7.40469132e+09 -7.40468963e+09 -7.40469370e+09 -7.40469123e+09
 -7.40469435e+09 -7.40469206e+09 -7.40469135e+09 -7.40469111e+09
 -7.40469067e+09 -7.40469315e+09 -7.40469329e+09 -7.40468801e+09
 -7.40469184e+09 -7.40469197e+09 -7.40469088e+09 -7.40469271e+09
 -7.40469258e+09 -7.40469206e+09 -7.40469285e+09 -7.40469010e+09
 -7.40469508e+09 -7.40469339e+09 -7.40469366e+09 -7.40469362e+09
 -7.40469110e+09 -7.40469344e+09 -7.40469266e+09 -7.40469383e+09
 -7.40469121e+09 -7.40469399e+09 -7.40469284e+09 -7.40469136e+09
 -7.40469261e+09 -7.40469314e+09 -7.40469209e+09 -7.40469253e+09
 -7.40469329e+09 -7.40469244e+09 -7.40469236e+09 -7.40469197e+09
 -7.40469090e+09 -7.40469435e+09 -7.40469325e+09 -7.40468728e+09
 -7.40469318e+09 -7.40468831e+09 -7.40469308e+09 -7.40469239e+09
 -7.40469448e+09 -7.40469269e+09 -7.40469325e+09 -7.40469160e+09]
```

In [67]:
```
print("Model skoru: " + str(model.score(X_train, y_train)))

coklu_dogrusal_egitim_rmse = np.sqrt(mean_squared_error(y_train, model.predict(X_train)))
print("Eğitim RMSE değeri:", coklu_dogrusal_egitim_rmse)
coklu_dogrusal_test_rmse = np.sqrt(mean_squared_error(y_test, model.predict(X_test)))
print("Test RMSE değeri:", coklu_dogrusal_test_rmse)
```

```
Model skoru: 0.18138858768180954
Eğitim RMSE değeri: 2440.9414152683553
Test RMSE değeri: 2556.184482087839
```

model tuning

In [68]:
```
X = data[['Age', 'Gender', 'Country']]
y = data['Result_sec']
X_encoded = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 20, random_state=90)

lm = LinearRegression()
model = lm.fit(X_train, y_train)
coklu_dogrusal_egitim_rmse_mt=np.sqrt(mean_squared_error(y_train, model.predict(X_train)))
print("Model Tuning Eğitim RMSE değeri:",coklu_dogrusal_egitim_rmse_mt)
coklu_dogrusal_test_rmse_mt=np.sqrt(mean_squared_error(y_test, model.predict(X_test)))
print("Model Tuning Test RMSE değeri:",coklu_dogrusal_test_rmse_mt)


print("Model skoru: " + str(model.score(X_train, y_train)))
```

```
Model Tuning Eğitim RMSE değeri: 2440.1766340143786
Model Tuning Test RMSE değeri: 3392.326421414406
Model skoru: 0.1815674089417627
```

5.1.3 PCR (Temel Bileşen Regresyonu - Principal Component Regression)

In [69]:
```
X = data[['Age', 'Gender', 'Country']]
```

```
y = data['Result_sec']
X_encoded = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 0.25, random_state= 42)

pca = PCA()
X_reduced_train = pca.fit_transform(scale(X_train))
X_reduced_train[0:1,:]

lm = LinearRegression()
pcr_model = lm.fit(X_reduced_train, y_train)
print(pcr_model.intercept_)
print(pcr_model.coef_)
```

```
13990.79336574965
[-3.32593385e+02  1.84447502e+02  5.69342092e+02 -2.96438668e+02
 -1.17768582e+01  5.88380087e+01  1.75908198e+01  4.07977345e+01
  6.18487549e+00 -4.17656250e+01  1.45000000e+01  2.27500000e+01
  8.30803223e+01  5.61817198e+01 -3.56329613e+01  4.97772352e+01
  2.69904836e+01  2.33554638e+01 -6.94729188e+01 -1.20654497e+01
  7.32382381e+00 -1.21123718e+02 -3.37343750e+01  4.25000000e+00
  7.06250000e+00 -3.83414001e+01  1.78919744e+01  1.05559128e+01
 -1.93645927e+01 -3.43434794e+00 -4.26613918e+01 -1.85009378e+00
  3.94934270e+01  1.78134766e+01 -1.71464844e+01  1.78359375e+01
 -3.42197266e+01  1.39550781e+00  5.56946976e+01  5.13700128e+01
 -3.14160294e+01 -3.67830977e+01  5.59761719e+01  2.92926543e+01
 -9.59642334e+01  1.37982178e+01 -9.26184082e+00 -2.64018555e+01
 -3.56192017e+01 -1.19512754e+01  2.42661246e+01  1.45840877e+01
  1.64494348e+01  1.28039849e+02 -1.86781397e+01  2.37480469e+01
  2.62031250e+01  1.88339844e+01  5.17651367e+00 -2.60028836e+00
 -5.39481071e+00 -6.87315786e+00  3.91711773e+01 -3.96339877e+01
  2.23414974e+01 -3.71235352e+01  3.48260498e+01  1.78066406e+01
 -7.87184525e+00 -2.81670743e+01  1.13140629e+01  2.17358122e+01
  1.15454998e+01 -1.22077030e+00 -7.14245605e+00  1.41250000e+01
  1.22705078e+01  1.15627526e+01 -2.66278259e+01 -1.97442993e+01
  1.08362908e+01  7.20908481e+00 -2.07021484e+01  2.90429688e+00
  9.03031737e-01 -4.69111132e+01 -2.32506411e+01 -8.99819183e+00
 -5.01953125e+00  4.66357422e+00 -7.31019050e-01 -1.61551931e+01
 -2.54181352e+01 -1.39681667e+01  1.22205936e+01  1.00584859e+01
 -1.77658195e+02 -7.31024583e+02  1.45207530e+15  1.54269887e+15
  2.33886817e+15 -2.70469935e+16 -1.96365596e+15 -1.09768095e+13
  1.73109293e+14  1.49949363e+15 -7.08665791e+13 -2.35549343e+15
  4.10277883e+14 -1.23941901e+14 -1.99140488e+15 -7.09745548e+14]
```

In [70]:
```
y_pred = pcr_model.predict(X_reduced_train)

pce_egitim_rmse = np.sqrt(mean_squared_error(y_train, y_pred))
print("Eğitim RMSE değeri:", pce_egitim_rmse)
pce_r2skor = r2_score(y_train, y_pred)
print("R-kare skoru:", pce_r2skor)
```

```
Eğitim RMSE değeri: 2449.5136048942427
R-kare skoru: 0.18202167449072282
```

In [71]:
```
pca2 = PCA()
X_reduced_test = pca2.fit_transform(scale(X_test))
y_pred = pcr_model.predict(X_reduced_test)
pce_test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Test RMSE değeri:", pce_test_rmse)
```

```
Test RMSE değeri: 2804.268190566341
```

model tuning

In [72]:
```
lm = LinearRegression()
pcr_model = lm.fit(X_reduced_train, y_train)
y_pred = pcr_model.predict(X_reduced_test)
pce_egitim_rmse2 = np.sqrt(mean_squared_error(y_test, y_pred))
print("Eğitim RMSE değeri:", pce_egitim_rmse2)
```

```
Eğitim RMSE değeri: 2804.268190566341
```

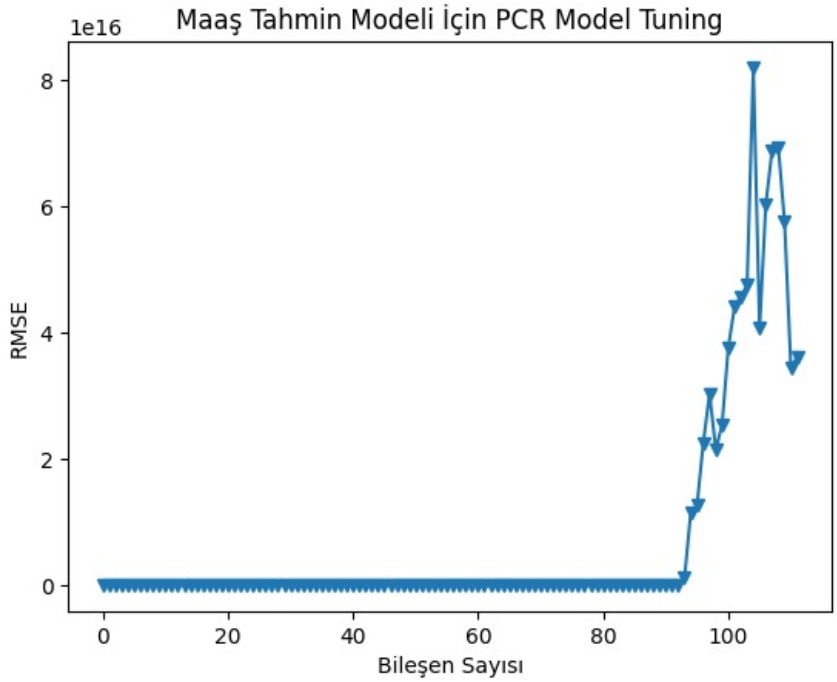
In [73]:
```
lm = LinearRegression()
pcr_model = lm.fit(X_reduced_train[:,0:15], y_train)
y_pred = pcr_model.predict(X_reduced_test[:,0:15])
pce_egitim_rmse_mt = np.sqrt(mean_squared_error(y_test, y_pred))
print("Model Tuning RMSE değeri:", pce_egitim_rmse_mt)
```

```
Model Tuning RMSE değeri: 2787.2921529452424
```

In [74]:
```
cv_10 = model_selection.KFold(n_splits = 10,
                              shuffle = True,
                              random_state = 1)
lm = LinearRegression()
RMSE = []
```

```
for i in np.arange(1, X_reduced_train.shape[1] + 1):

    score = np.sqrt(-1*model_selection.cross_val_score(lm,
                                                       X_reduced_train[:,:i],
                                                       y_train.ravel(),
                                                       cv=cv_10,
                                                       scoring='neg_mean_squared_error').mean())
    RMSE.append(score)
```

In [75]:
```
plt.plot(RMSE, '-v')
plt.xlabel('Bileşen Sayısı')
plt.ylabel('RMSE')
plt.title('Maaş Tahmin Modeli İçin PCR Model Tuning');
```



Anlamlı bir grafik çıkmamıştır 100 bileşenden sonra sonuçlar daha kötü olmaktadır.

In [76]:
```
lm = LinearRegression()
pcr_model = lm.fit(X_reduced_train[:,0:50], y_train)

y_pred = pcr_model.predict(X_reduced_train[:,0:50])
pce_egitim_rmse_mt2 = np.sqrt(mean_squared_error(y_train, y_pred))
print("Model Tuning Eğitim RMSE değeri:", pce_egitim_rmse_mt2)

y_pred = pcr_model.predict(X_reduced_test[:,0:50])
pce_test_rmse_mt2 = np.sqrt(mean_squared_error(y_test, y_pred))
print("Model Tuning Test RMSE değeri:", pce_test_rmse_mt2)
```

```
Model Tuning Eğitim RMSE değeri: 2551.0483256544094
Model Tuning Test RMSE değeri: 2797.514277289227
```

5.1.4 PLS (Kısmi En Küçük Kareler Regresyonu)

In [77]:
```
X = data[['Age', 'Gender', 'Country']]
y = data['Result_sec']
X_encoded = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 0.25, random_state= 42)

pls_model = PLSRegression().fit(X_train, y_train)
```

In [78]:
```
y_pred = pls_model.predict(X_train)
pls_egitim_rmse = np.sqrt(mean_squared_error(y_train, y_pred))
print("Eğitim RMSE değeri:", pls_egitim_rmse)
```

```
Eğitim RMSE değeri: 2451.115605080199
```

In [79]:
```
pls_r2skor = r2_score(y_train, y_pred)
print("R-kare skoru:", pls_r2skor)
```

```
R-kare skoru: 0.180951396756307
```

In [80]:
```
y_pred = pls_model.predict(X_test)
pls_test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

```
print("Test RMSE değeri:", pls_test_rmse)
```

```
Test RMSE değeri: 2425.9155528093315
```

model tuning

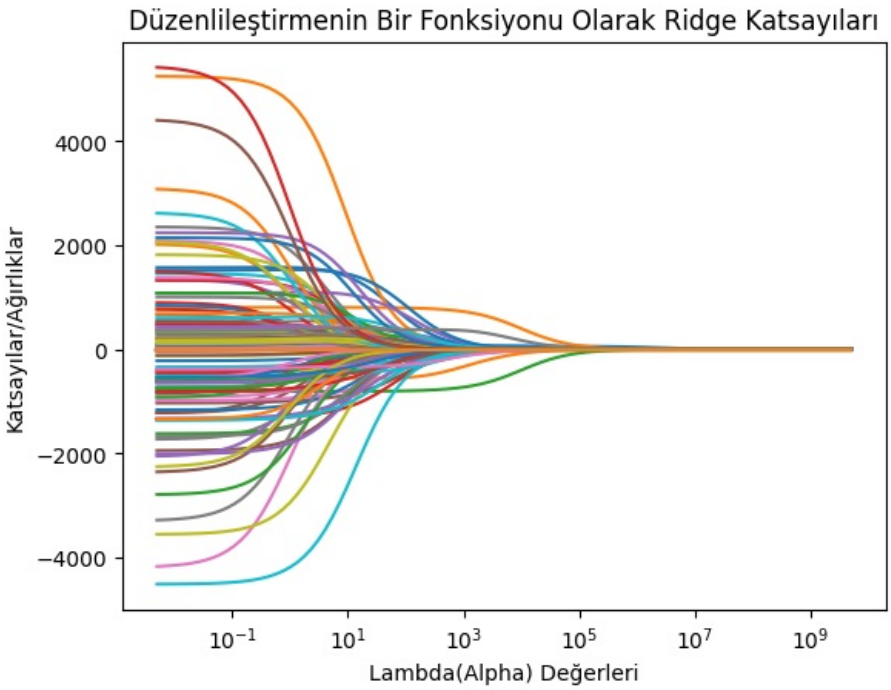
In [81]:
```
pls_model = PLSRegression(n_components = 2).fit(X_train, y_train)

y_pred = pls_model.predict(X_test)

pls_test_rmse_mt = np.sqrt(mean_squared_error(y_test, y_pred))
print("Model Tuning Test RMSE değeri:", pls_test_rmse_mt)
```

```
Model Tuning Test RMSE değeri: 2425.9155528093315
```

5.1.5 Ridge Regresyon

In [82]:
```
X = data[['Age', 'Gender', 'Country']]
y = data['Result_sec']

X_encoded = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 0.25, random_state= 42)

ridge_model = Ridge(alpha = 0.1).fit(X_train, y_train)
```

In [83]:
```
lambdalar = 10**np.linspace(10,-2,100)*0.5

ridge_model = Ridge()
katsayilar = []

for i in lambdalar:
    ridge_model.set_params(alpha = i)
    ridge_model.fit(X_train, y_train)
    katsayilar.append(ridge_model.coef_)


import matplotlib.pyplot as plt
ax = plt.gca()
ax.plot(lambdalar, katsayilar)
ax.set_xscale('log')

plt.xlabel('Lambda(Alpha) Değerleri')
plt.ylabel('Katsayılar/Ağırlıklar')
plt.title('Düzenlileştirmenin Bir Fonksiyonu Olarak Ridge Katsayıları');
```



In [84]:
```
y_pred = ridge_model.predict(X_train)
ridge_egitim_rmse = np.sqrt(mean_squared_error(y_train, y_pred))
print("Eğitim RMSE değeri:", ridge_egitim_rmse)
```

```
Eğitim RMSE değeri: 2449.5999259733744
```

In [85]:
```
ridge_r2skor = r2_score(y_train, y_pred)
print("R-kare skoru:", ridge_r2skor)
```

```
R-kare skoru: 0.1819640222157951
```

In [86]:
```
y_pred = ridge_model.predict(X_test)
ridge_test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Test RMSE değeri:", ridge_test_rmse)
```

```
Test RMSE değeri: 2422.7749705756128
```

model tuning

In [87]:
```
ridge_cv = RidgeCV(alphas=lambdalar, scoring="neg_mean_squared_error")
ridge_cv.fit(X_train, y_train)
ridge_tuned = Ridge(alpha=ridge_cv.alpha_).fit(X_train, y_train)

ridge_test_rmse_mt = np.sqrt(mean_squared_error(y_test, ridge_tuned.predict(X_test)))
print("Model Tuning Test RMSE değeri:", ridge_test_rmse_mt)
```

```
Model Tuning Test RMSE değeri: 2419.649197358269
```

5.1.6 Lasso Regresyon

In [88]:
```
X = data[['Age', 'Gender', 'Country']]
y = data['Result_sec']
X_encoded = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 0.25, random_state= 42)

lasso_model = Lasso(alpha = 0.1).fit(X_train, y_train)
```

In [89]:
```
y_pred = lasso_model.predict(X_train)
lasso_egitim_rmse = np.sqrt(mean_squared_error(y_train, y_pred))
print("Eğitim RMSE değeri:", lasso_egitim_rmse)
```

```
Eğitim RMSE değeri: 2450.431082230728
```

In [90]:
```
lasso_r2skor = r2_score(y_train, y_pred)
print("R-kare skoru:",lasso_r2skor)
```

```
R-kare skoru: 0.18140880414680427
```

In [91]:
```
y_pred = lasso_model.predict(X_test)
lasso_test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Test RMSE değeri:", lasso_test_rmse)
```

```
Test RMSE değeri: 2420.55892628749
```

model tuning

In [92]:
```
lasso_cv_model = LassoCV(alphas=None, cv=10, max_iter=10000)
lasso_cv_model.fit(X_train, y_train)
lasso_tuned = Lasso(alpha=lasso_cv_model.alpha_)
lasso_tuned.fit(X_train, y_train)
y_pred = lasso_tuned.predict(X_test)
lasso_test_rmse_mt = np.sqrt(mean_squared_error(y_test, y_pred))
print("Model Tuning Test RMSE değeri:", lasso_test_rmse_mt)
```

```
Model Tuning Test RMSE değeri: 2427.856547723586
```

5.1.7 ElasticNet (eNet) Regresyonu

In [93]:
```
X = data[['Age', 'Gender', 'Country']]
y = data['Result_sec']

X_encoded = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 0.25, random_state= 42)

lasso_model = Lasso(alpha = 0.1).fit(X_train, y_train)
enet_model = ElasticNet().fit(X_train, y_train)
```

In [94]:
```
y_pred = enet_model.predict(X_train)
eNet_egitim_rmse = np.sqrt(mean_squared_error(y_train, y_pred))
print("Eğitim RMSE değeri:", eNet_egitim_rmse)
```

```
Eğitim RMSE değeri: 2516.0055908947493
```

In [95]:
```
eNet_r2skor = r2_score(y_train, y_pred)
print("R-kare skoru:", eNet_r2skor)
```

```
R-kare skoru: 0.13701094474298592
```

In [96]:
```
y_pred = enet_model.predict(X_test)
eNet_test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Test RMSE değeri:", eNet_test_rmse)
```

```
Test RMSE değeri: 2464.920036922978
```

model tuning

In [97]:
```
enet_cv_model = ElasticNetCV(cv = 10, random_state = 0).fit(X_train, y_train)
enet_tuned = ElasticNet(alpha = enet_cv_model.alpha_).fit(X_train,y_train)
y_pred = enet_tuned.predict(X_test)
eNet_test_rmse_mt = np.sqrt(mean_squared_error(y_test, y_pred))
print("Model Tuning Test RMSE değeri:", eNet_test_rmse_mt)
```

```
Model Tuning Test RMSE değeri: 2551.879030807876
```

5.2 Doğrusal Olmayan Regresyon

5.2.1 K En Yakın Komşu (KNN)

In [98]:
```
X = data[['Age', 'Gender', 'Country']]
y = data['Result_sec']

X_encoded = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.25, random_state=42)

knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)

y_pred = knn_model.predict(X_test)
```

In [99]:
```
RMSE = []
for k in range(1, 11):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(X_train, y_train)
    y_pred = knn_model.predict(X_train)
    knn_rmse = np.sqrt(mean_squared_error(y_train, y_pred))
    RMSE.append(knn_rmse)
    print("k =", k, "için RMSE değeri:", knn_rmse)
```

```
k = 1 için RMSE değeri: 3141.8382739876884
k = 2 için RMSE değeri: 2929.829894793801
k = 3 için RMSE değeri: 3116.998903472465
k = 4 için RMSE değeri: 3331.58714702181
k = 5 için RMSE değeri: 3407.190027380276
k = 6 için RMSE değeri: 3528.0211293022726
k = 7 için RMSE değeri: 3577.331156598783
k = 8 için RMSE değeri: 3608.735345668486
k = 9 için RMSE değeri: 3656.0720557828513
k = 10 için RMSE değeri: 3712.7978701465477
```

5.2.2 Çok Katmanlı Algılayıcı (Yapay Sinir Ağları YSA)

In [100…
```
X = data[["Age", "Gender", "Country"]]
y = data["Result_sec"]

le = LabelEncoder()
X.loc[:, "Gender"] = le.fit_transform(X.loc[:, "Gender"])
X.loc[:, "Country"] = le.fit_transform(X.loc[:, "Country"])

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=42)

model = Sequential()
model.add(Dense(32, input_dim=3, activation="relu"))
model.add(Dense(16, activation="relu"))
model.add(Dense(1))

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

model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=1)

y_pred = model.predict(X_test)

comparison = pd.DataFrame({"Gerçek": y_test, "Tahmin": y_pred.flatten()})
print(comparison)
```

```
Epoch 1/50
625/625 [==============================] - 1s 654us/step - loss: 200137104.0000
Epoch 2/50
625/625 [==============================] - 0s 641us/step - loss: 160503088.0000
Epoch 3/50
625/625 [==============================] - 0s 662us/step - loss: 78195304.0000
Epoch 4/50
625/625 [==============================] - 0s 644us/step - loss: 21208792.0000
```

```
Epoch 5/50
625/625 [==============================] - 0s 644us/step - loss: 8020820.5000
Epoch 6/50
625/625 [==============================] - 0s 636us/step - loss: 6869911.5000
Epoch 7/50
625/625 [==============================] - 0s 643us/step - loss: 6519985.5000
Epoch 8/50
625/625 [==============================] - 0s 641us/step - loss: 6320588.5000
Epoch 9/50
625/625 [==============================] - 0s 636us/step - loss: 6211078.0000
Epoch 10/50
625/625 [==============================] - 0s 636us/step - loss: 6151366.5000
Epoch 11/50
625/625 [==============================] - 0s 639us/step - loss: 6116618.0000
Epoch 12/50
625/625 [==============================] - 0s 639us/step - loss: 6094910.0000
Epoch 13/50
625/625 [==============================] - 0s 639us/step - loss: 6076922.0000
Epoch 14/50
625/625 [==============================] - 0s 641us/step - loss: 6061699.5000
Epoch 15/50
625/625 [==============================] - 0s 643us/step - loss: 6049714.5000
Epoch 16/50
625/625 [==============================] - 0s 643us/step - loss: 6040425.5000
Epoch 17/50
625/625 [==============================] - 0s 638us/step - loss: 6029347.5000
Epoch 18/50
625/625 [==============================] - 0s 638us/step - loss: 6022555.5000
Epoch 19/50
625/625 [==============================] - 0s 651us/step - loss: 6015096.5000
Epoch 20/50
625/625 [==============================] - 0s 681us/step - loss: 6009971.0000
Epoch 21/50
625/625 [==============================] - 0s 682us/step - loss: 6005470.5000
Epoch 22/50
625/625 [==============================] - 0s 672us/step - loss: 6003660.5000
Epoch 23/50
625/625 [==============================] - 0s 658us/step - loss: 5998878.0000
Epoch 24/50
625/625 [==============================] - 0s 646us/step - loss: 5998015.5000
Epoch 25/50
625/625 [==============================] - 0s 674us/step - loss: 5994500.5000
Epoch 26/50
625/625 [==============================] - 0s 654us/step - loss: 5995456.5000
Epoch 27/50
625/625 [==============================] - 0s 686us/step - loss: 5992120.0000
Epoch 28/50
625/625 [==============================] - 0s 649us/step - loss: 5991033.0000
Epoch 29/50
625/625 [==============================] - 0s 649us/step - loss: 5988296.5000
Epoch 30/50
625/625 [==============================] - 0s 649us/step - loss: 5987062.0000
Epoch 31/50
625/625 [==============================] - 0s 646us/step - loss: 5987504.5000
Epoch 32/50
625/625 [==============================] - 0s 666us/step - loss: 5987697.5000
Epoch 33/50
625/625 [==============================] - 0s 650us/step - loss: 5982973.0000
Epoch 34/50
625/625 [==============================] - 0s 649us/step - loss: 5984158.5000
Epoch 35/50
625/625 [==============================] - 0s 651us/step - loss: 5983766.0000
Epoch 36/50
625/625 [==============================] - 0s 674us/step - loss: 5982317.0000
Epoch 37/50
625/625 [==============================] - 0s 663us/step - loss: 5980257.5000
Epoch 38/50
625/625 [==============================] - 0s 646us/step - loss: 5982846.5000
Epoch 39/50
625/625 [==============================] - 0s 641us/step - loss: 5979888.5000
Epoch 40/50
625/625 [==============================] - 0s 644us/step - loss: 5980216.0000
Epoch 41/50
625/625 [==============================] - 0s 641us/step - loss: 5980038.5000
Epoch 42/50
625/625 [==============================] - 0s 677us/step - loss: 5976744.5000
Epoch 43/50
625/625 [==============================] - 0s 639us/step - loss: 5978112.5000
Epoch 44/50
625/625 [==============================] - 0s 644us/step - loss: 5977913.0000
Epoch 45/50
625/625 [==============================] - 0s 651us/step - loss: 5977602.5000
Epoch 46/50
```

```
625/625 [==============================] - 0s 651us/step - loss: 5978258.5000
Epoch 47/50
625/625 [==============================] - 0s 641us/step - loss: 5976891.0000
Epoch 48/50
625/625 [==============================] - 0s 644us/step - loss: 5975563.0000
Epoch 49/50
625/625 [==============================] - 0s 639us/step - loss: 5976641.5000
Epoch 50/50
625/625 [==============================] - 0s 643us/step - loss: 5976019.5000
209/209 [==============================] - 0s 527us/step
       Gerçek        Tahmin
11728   15631  12631.559570
4990    11820  11773.446289
13212   17383  12814.240234
5337    11938  14239.313477
1530    10475  12872.903320
...       ...           ...
19972   13874  15395.631836
8843    13507  14552.091797
24575   17007  15512.726562
19786   13798  14815.564453
2498    10818  13670.840820

[6663 rows x 2 columns]
```

In [101…
```
y_pred = model.predict(X_train)
ysa_egitim_rmse = np.sqrt(mean_squared_error(y_train, y_pred))
print("Eğitim RMSE değeri:", ysa_egitim_rmse )
```

```
625/625 [==============================] - 0s 516us/step
Eğitim RMSE değeri: 2445.515145229996
```

In [102…
```
ysa_r2skor = r2_score(y_train, y_pred)
print("R-kare skoru:", ysa_r2skor)
```

```
R-kare skoru: 0.184689946192778
```

In [103…
```
y_pred = model.predict(X_test)
ysa_test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Test RMSE değeri:", ysa_test_rmse)
```

```
209/209 [==============================] - 0s 518us/step
Test RMSE değeri: 2394.226167023456
```

5.3 Sınıflandırma Problemleri

5.3.1 Gaussian Naive Bayes

In [104…
```
X = data[['Age', 'Gender', 'Country']]
y = data['Result_sec']
X_encoded = pd.get_dummies(X)


X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 0.25, random_state= 42)
nb = GaussianNB()
nb_model = nb.fit(X_train, y_train)
y_pred = nb_model.predict(X_test)
gnba_test_rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Ortalama Karesel Hata (RMSE):", gnba_test_rmse)
```

```
Ortalama Karesel Hata (RMSE): 4074.635814175297
```

In [105…
```
accuracy_score(y_test, y_pred)
```

Out[105]: 0.00030016509079993996

Makine öğrenmesi için sonuç:

In [106…
```
metodlar = ["Basit Doğrusal Regresyon", "Çoklu Doğrusal Regresyon","Temel Bileşen Regresyonu","Kısmi En Küçük Ka
metod_degerleri = [basit_dogrusal_rmse,coklu_dogrusal_test_rmse_mt,pce_test_rmse_mt2,pls_test_rmse_mt,ridge_tes

for index in range(10):
    print(metodlar[index],"için RMSE değeri: ",metod_degerleri[index])
    print("")
    if(index==6 or index==8):
        print("-"*55)
        print("")
```

```
Basit Doğrusal Regresyon için RMSE değeri:  2604.2762737604457

Çoklu Doğrusal Regresyon için RMSE değeri:  3392.326421414406

Temel Bileşen Regresyonu için RMSE değeri:  2797.514277289227

Kısmi En Küçük Kareler Regresyonu için RMSE değeri:  2425.9155528093315

Ridge Regresyon için RMSE değeri:  2419.649197358269

Lasso Regresyon için RMSE değeri:  2427.856547723586

ElasticNet Regresyon için RMSE değeri:  2551.879030807876

-----------------------------------------------------

K En Yakın Komşu için RMSE değeri:  3712.7978701465477

Yapay Sinir Ağları için RMSE değeri:  2445.515145229996

-----------------------------------------------------

Gaussian Naive Bayes için RMSE değeri:  4074.635814175297
```

En düşük RMSE değerini Ridge Regresyon metodu vermiştir fakat diğer metodlarla arasında çok fark yoktur diğer doğrusal regresyon metodları da ortalama 2500 RMSE değeri vermiştir.

Doğrusal Regresyon, Doğrusal Olmayan Regresyon ve Sınıflandırma problemlerini koşucuların maratonu bitirme sürelerini tahmin etmek için kullandım. Doğrusal Regresyon diğer ikisine kıyasla daha başarılı sonuçlar verdiği için Doğrusal Regresyon metodlarını veri setimde uyguladım fakat diğer ikisine kıyasla daha iyi olsada yinede tahminler başarısızdır. Model tuningle de sonuçlarda pek bir değişiklik olmamaktadır. Sonuç olarak koşucunun yaş, cinsiyet ve doğduğu ülke bilgileriyle maratonu bitirme süresi tahminleri başarılı şekilde yapılmamaktadır başka bilgilerde gereklidir.