```
#1. Uyarıları kapatınız
```

```
#2.   0 ile 20 arasında rasgele sayılar alan 10 satır ve 2 sutundan oluşan bir
# matris oluşturun
import numpy as np
matris = np.random.randint(0, 21, size=(10, 2))
print(matris)
```

```
[[ 6  8]
 [17 10]
 [ 1  0]
 [11  0]
 [ 5  0]
 [16  4]
 [13  6]
 [ 7 19]
 [ 3 10]
 [ 6 11]]
```

```
#3.  Aşağıdaki komut sonrası oluşturulan ikiboyutlu matrisi 2 ve 4 nolu
# sutunlardan ayrıştırarak 3 adet a, b ve c martislerine dönüştürün.
import numpy as np
m = np.arange(30).reshape(5,6)
print(m)
a = m[:,:2].reshape(-1, 2)
b = m[:, 2:4].reshape(-1,2)
c = m[:,4:6].reshape(-1,2)

print("Matris a:")
print(a)

print("\nMatris b:")
print(b)

print("\nMatris c:")
print(c)
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]]
Matris a:
[[ 0  1]
 [ 6  7]
 [12 13]
 [18 19]
 [24 25]]

Matris b:
[[ 2  3]
 [ 8  9]
 [14 15]
 [20 21]
 [26 27]]

Matris c:
[[ 4  5]
 [10 11]
 [16 17]
 [22 23]
 [28 29]]
```

```
#4.   Aşağıdaki komut sonrası oluşturulan ikiboyutlu matrisi 3 nolu satırdan
# ayrıştırarak 2 adet a ve b martislerine dönüştürün
import numpy as np
m = np.arange(30).reshape(5,6)
print(m)
a= m[0:2,:].reshape(2,-1)
print("\n")
print(a)
b = m[2:,:].reshape(3,-1)
print("\n")
print(b)
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]]


[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]


[[12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]]
```

```
#5.  Aşağıdaki komutlar sonrası oluşturulan a matrisini colon adları “sutun1”,
#  ”sutun2” olan dataframe'e dönüştürün.
import numpy as np
import pandas as pd
a=np.random.randint(0,20,(10,2))
print(a)

x=pd.DataFrame(data=a,index=range(10),columns=["sutun1","sutun2"])
print(x)
```

```
[[12 18]
 [ 6  3]
 [11  5]
 [ 1 13]
 [16  2]
 [19 18]
 [ 1 10]
 [ 8  7]
 [ 9  6]
 [17 16]]
   sutun1  sutun2
0      12      18
1       6       3
2      11       5
3       1      13
4      16       2
5      19      18
6       1      10
7       8       7
8       9       6
9      17      16
```

```
#6. Önceki soruda oluşturulan dataframe’in index’lerini “i1”,”i2”,...,”i10”
# olarak değiştiriniz
import pandas as pd
import numpy as np
x=pd.DataFrame(data=a,index=range(1,11),columns=["sutun1","sutun2"])
x.index=(f"i{index}" for index in range(1,11))
print(x)
```

```
     sutun1  sutun2
i1       12      18
i2        6       3
i3       11       5
i4        1      13
i5       16       2
i6       19      18
i7        1      10
i8        8       7
```

```
i9        9       6
i10      17      16
```

```
#7. Aşağıdaki sözlük yapısındaki 99 değerini 100 olarak değiştirmek için
# komutu yazınız.

import pandas as pd

sozluk = {"A" : {"D":[1,2]},  "B" : {"E": [3,99,5]},  "C" : ["F",6]}

sozluk["B"]["E"][1] = 100
print(sozluk)
```

```
{'A': {'D': [1, 2]}, 'B': {'E': [3, 100, 5]}, 'C': ['F', 6]}
```

```
#8. 1 ile 10 dahil ardışık tam sayılardan oluşan bir numpy dizisi oluşturun
# ve bu diziyi 5 satır ve 2 sutundan oluşan bir matrise dönüştürün.
import numpy as np
m = np.arange(1,11).reshape(5,2)
print(m)
```

```
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]]
```

```
#9. Önceki soruda oluşturulan matrisi bir dataframe!e dönüştürün ve bu dataframe'e
# ilk 2 sutun toplamlarından oluşan yeni bir "toplam" adlı sutun ekleyin.
import numpy as np
import pandas as pd
m = np.arange(1,11).reshape(5,2)
y=pd.DataFrame(data=m,index=range(5),columns=["A","B"])

y['Toplam'] = y["A"] + y["B"]
print(y)
```

```
   A   B  Toplam
0  1   2       3
1  3   4       7
2  5   6      11
3  7   8      15
4  9  10      19
```

```
#10. İki adet a = [1,2,3,4] ve b = [5,6,7,8] numpy dizisi oluşturun ve bu dizileri
# içeren yeni bir 2 boyutlu c numpy matrisi oluşturun.
import numpy as np
m = np.arange(1,5).reshape(1,4)
n=np.arange(5,9).reshape(1,4)
c = np.array([m, n])

print(c)
```

```
[[[1 2 3 4]]

 [[5 6 7 8]]]
```

```
#11. 10 ile 20 arasında 10 adet rasgele tam sayıdan oluşan bir numpy dizisi oluşturun
# ve bu dizinin ortalamasından büyük olan sayılardan oluşan listeyi yazdırınız.
import numpy as np


numpy_dizi = np.random.randint(10, 21, 10)


ortalama = np.mean(numpy_dizi)
buyuk_sayilar = [sayi for sayi in numpy_dizi if sayi > ortalama]

print("Oluşturulan Numpy Dizisi:", numpy_dizi)
print("Dizinin Ortalaması:", ortalama)
print("Ortalamadan Büyük Sayılar:", buyuk_sayilar)
```

```
Oluşturulan Numpy Dizisi: [13 11 15 18 10 12 20 19 14 20]
Dizinin Ortalaması: 15.2
Ortalamadan Büyük Sayılar: [18, 20, 19, 20]
```

```
#12. -50 ile 50 arası rasgele sayılardan oluşan 10 elemanlık bir numpy dizisi oluşturun.
# Bu elemanların Z-score normalleştirilmiş değerlerinden oluşan b dizisi oluşturunuz
from scipy.stats import zscore
import numpy as np
matris = np.random.randint(-50, 51, 10) #aralığı ve kaç eleman olacağını belirledik.
b= zscore(matris)
print("Orjinal Matris:")
print(matris)
print("\nZ-score Normalleştirilmiş Matris:")
print(b)
```

```
Orjinal Matris:
[ 19  46  48 -39 -27 -19  17   4   8  -4]

Z-score Normalleştirilmiş Matris:
[ 0.50038339  1.48654042  1.55958909 -1.61802803 -1.17973601 -0.88754133
  0.42733472 -0.04748164  0.0986157  -0.33967631]
```

```
#13. Aşağıdaki şekilde bir dataframe oluşturulmuştur. Bu dataframe'in gruplara göre
# veri ortalamalarını yansıtan tabloyu yazdıran komutu yazınız.
import pandas as pd
df = pd.DataFrame({'gruplar': ['A', 'B', 'C', 'A', 'B', 'C'],
                   'veri': [10,11,52,23,43,55]}, columns=['gruplar', 'veri'])

ortalama_tablo = df.groupby('gruplar')['veri'].mean().reset_index() #groupby fonksiyonunu kullandık

print(ortalama_tablo)
```

```
  gruplar  veri
0       A  16.5
1       B  27.0
2       C  53.5
```

Düzenlemek için çift tıklayın (veya enter tuşuna basın)

```
#14. ";" karakterleri ile ayrılmış veriler içeren dış kaynaklı "veri.csv" dosyasını
# bir dataframe'e okumak için komutu yazın
import pandas as pd
eray=pd.read_csv("veri.csv",sep=";")  #sep parametresi ile ayırmak istediğimiz yolu belirledik.

#Burada hata vermesinin sebebi "veri.csv" adında bir csv dosyasının olmayıp varmış gibi yapmamız!
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[14], line 4
      1 #14. ";" karakterleri ile ayrılmış veriler içeren dış kaynaklı "veri.csv"
dosyasını
      2 # bir dataframe'e okumak için komutu yazın
      3 import pandas as pd
----> 4 eray=pd.read_csv("veri.csv",sep=";")  #sep parametresi ile ayırmak
istediğimiz yolu belirledik.
      6 #Burada hata vermesinin sebebi "veri.csv" adında bir csv dosyasının
olmayıp varmış gibi yapmamız!

File /opt/conda/lib/python3.10/site-packages/pandas/io/parsers/readers.py:912, in
read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols,
dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows,
skipfooter, nrows, na_values, keep_default_na, na_filter, verbose,
skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser,
date_format, dayfirst, cache_dates, iterator, chunksize, compression, thousands,
decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment,
encoding, encoding_errors, dialect, on_bad_lines, delim_whitespace, low_memory,
memory_map, float_precision, storage_options, dtype_backend)
    899 kwds_defaults = _refine_defaults_read(
    900     dialect,
    901     delimiter,
   (...)
    908     dtype_backend=dtype_backend,
    909 )
    910 kwds.update(kwds_defaults)
--> 912 return _read(filepath_or_buffer, kwds)

File /opt/conda/lib/python3.10/site-packages/pandas/io/parsers/readers.py:577, in
_read(filepath_or_buffer, kwds)
    574 _validate_names(kwds.get("names", None))
    576 # Create the parser.
--> 577 parser = TextFileReader(filepath_or_buffer, **kwds)
    579 if chunksize or iterator:
    580     return parser

File /opt/conda/lib/python3.10/site-packages/pandas/io/parsers/readers.py:1407, in
TextFileReader.__init__(self, f, engine, **kwds)
   1404     self.options["has_index_names"] = kwds["has_index_names"]
   1406 self.handles: IOHandles | None = None
-> 1407 self._engine = self._make_engine(f, self.engine)

File /opt/conda/lib/python3.10/site-packages/pandas/io/parsers/readers.py:1661, in
TextFileReader._make_engine(self, f, engine)
   1659     if "b" not in mode:
   1660         mode += "b"
-> 1661 self.handles = get_handle(
   1662     f,
   1663     mode,
   1664     encoding=self.options.get("encoding", None),
   1665     compression=self.options.get("compression", None),
   1666     memory_map=self.options.get("memory_map", False),
   1667     is_text=is_text,
   1668     errors=self.options.get("encoding_errors", "strict"),
   1669     storage_options=self.options.get("storage_options", None),
   1670 )
   1671 assert self.handles is not None
   1672 f = self.handles.handle

File /opt/conda/lib/python3.10/site-packages/pandas/io/common.py:859, in
get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors,
```

```
#15. Parametre olarak aldığı bir kelimeyi tümüyle büyük karakterlere dönüştürerek
# geri döndüren "funk" adlı bir fonksiyonun tanımını yazınız.
def funk(a: str):
    buyuk_harfler = a.upper()
    return buyuk_harfler

sonuc = funk("eray")
print(sonuc)
```

```
#16. Python'da "seaborn" kütüphanesinden "mpg.csv" veri setini indiriniz,
# veri seti bilgilerini yazdırınız (info),
# veri setinin ilk 10 satırını ekrana yazdırınız
import pandas as pd
import seaborn as sns
dosya_yolu="/kaggle/input/mpg-verisi/mpg_veri.csv"
x=pd.read_csv(dosya_yolu)
x.info()
x.head(10)
```

```
#17. veri setinin uygun atributlraına göre ortalama, varyans, min, max, vs.
# değerleri yazdırınız,
import pandas as pd
a=x.describe()
print(a)
#Varyansı yazdırmamızın iki farklı yolu 1)Ekstra bir varyans olarak yazdırdım.
statistics = a
print("\nVaryans:\n", statistics.var())
#2)Varyan standart sapmanın karesi olduğundan dolayı ikinci yolu da böyle yaptım.
a.loc['variance']=a.loc['std']**2
print(a)


#18. “horsepower” ve “weight” atributları için, “origin” kırılımına (hue)
# göre “scatter plot” grafiğini oluşturunuz
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

sns.scatterplot(x='horsepower', y='weight', hue='origin', data=x)
plt.show()


#19. “mpg” ve “acceleration” değişkenleri arasında Kendall
# korelasyon katsayısını hesaplamalı ve yazdırmalı.
from scipy.stats import kendalltau  #scipy kütüphanesi kullanıyorum


kendall_corr, p_value = kendalltau(x['mpg'],x['acceleration'])

print(f"Kendall Korelasyon Katsayısı: {kendall_corr:.4f}")




#20. “horsepower”, “acceleration” ve “weight” değişkenlerine göre
# “mpg” değerinin tahmini için
# LinearRegression() modeli oluşturunuz
# Modelin intercept ve katsayılarını yazdırınız
# NOT: kayıp veriler olabilir. Gerekirse kayıp verileri siliniz.

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

dosya_yolu="/kaggle/input/mpg-verisi/mpg_veri.csv"
mpg_data=pd.read_csv(dosya_yolu)
mpg_data.dropna(subset=['horsepower', 'acceleration', 'weight', 'mpg'], inplace=True)

X = mpg_data[['horsepower', 'acceleration', 'weight']] #b'sız değişken
y = mpg_data['mpg'] #b'lı değişken

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)

print("Intercept (Düşey Kesim):", model.intercept_)
print("Katsayılar:", model.coef_)
```

```
#21. Oluşturulan modelin R-squared değerini yazdırınız
from sklearn.metrics import r2_score

y_pred = model.predict(X_test)
r_squared = r2_score(y_test, y_pred)
print("R-squared Değeri:", r_squared)
```

```
#22. horsepower = 130, acceleration=13, weight=3500 olan bir
# otomobilin “mpg” değerini tahmin ediniz.
# Verilen özellik değerleri
import numpy as np
from sklearn.linear_model import LinearRegression

dosya_yolu="/kaggle/input/mpg-verisi/mpg_veri.csv"
df=pd.read_csv(dosya_yolu)

df.dropna(inplace=True)

X=df[['horsepower','acceleration','weight']]
y = df['mpg']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Eğitim Veri Seti NaN Kontrolü:\n", X_train.isnull().sum())
print("Test Veri Seti NaN Kontrolü:\n", X_test.isnull().sum())

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

horsepower_value = 130
acceleration_value = 13
weight_value = 3500

new_data_point = [[horsepower_value, acceleration_value, weight_value]]

mpg_prediction = model.predict(new_data_point)

print("Tahmin edilen MPG değeri:", mpg_prediction[0])
```

```
#23. “mpg.csv” veri setini, “mpg”, “displacement”,”horsepower”,”weight”
```

```
#24. Oluşturulan modelin test seti üzerinde doğruluk (accuracy) skorunu hesaplayınız,
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print("Test seti doğruluk oranı:", accuracy)
```

```
#25.Değişkenlerin önem derecelerini (Importance) yazdırınız.
# Gerekli kütüphaneleri yükleyin
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


feature_importances = rf_model.feature_importances_
feature_names = X.columns
for feature, importance in zip(feature_names, feature_importances):
    print(f"{feature}: {importance}")
```