MOBIL PHONE PRICE

I-

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import warnings
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
warnings.filterwarnings('ignore')


def veriler_read():
    veriler = pd.read_csv('Mobile phone price.csv')
    print(veriler)

    #istatistik için veri temizleme ve düzenleme
    #Storage sütununu düzenleme
    storage = veriler["Storage "].astype(str)
    # "GB" ifadesini kaldır
    for i in range(len(storage)):
            if 'GB' in storage[i]:
                storage[i] = storage[i].replace('GB', '')
            elif ' ' in storage[i]:
                storage[i] = storage[i].replace(' ', '')
    veriler['Storage '] = storage
    #RAM sütununu düzenleme
    ram = veriler["RAM "].astype(str)
    #GB ifadesini kaldir
    for i in range(len(ram)):
        if 'GB' in ram[i]:
            ram[i] = ram[i].replace('GB','')
        elif ' ' in ram[i]:
            ram[i] = ram[i].replace(' ','')

    veriler['RAM '] = ram

    screen = veriler['Screen Size (inches)'].astype(str)
    for i in range(len(screen)):
        if ' ' in screen[i]:
            screen[i] = screen[i].replace(' ','')
        elif r'\(.*\)' in screen[i]:
            screen[i] = screen[i].replace(r'\(.*\)', '').str.strip()
    veriler['Screen Size (inches)'] = screen

    veriler[['Screen Size (inches 1)', 'Screen Size (inches 2)']] =
veriler['Screen Size (inches)'].str.split('+', expand=True)
    veriler['Screen Size (inches 1)'] = veriler['Screen Size (inches
1)'].str.replace(' ', '').astype(float)
    veriler['Screen Size (inches 2)'] = veriler['Screen Size (inches
2)'].str.replace(' ', '').astype(float)
    veriler['Screen Size (inches)'] = veriler.apply(
        lambda x: x['Screen Size (inches 1)'] + x['Screen Size (inches 2)']
if pd.notnull(x['Screen Size (inches 2)']) else
        x['Screen Size (inches 1)'], axis=1)
```

```
    # 'Screen Size (inches)' sütunu siliniyor
    veriler.drop('Screen Size (inches)', axis=1, inplace=True)

    # 'Camera (MP)' sütunu ayrıştırılıyor
    kamera = veriler['Camera (MP)'].str.split('+', expand=True)
    kamera.columns = [f"Camera {i+1}" for i in range(kamera.shape[1])]
    print(kamera.columns)

    # Kameranın megapiksel değerleri ayrıştırılıyor ve eksik değerler NaN
ile değiştiriliyor
    for col in kamera.columns:
        kamera[col] = kamera[col].str.strip().str.replace('MP','')
        kamera[col] = pd.to_numeric(kamera[col], errors='coerce')

    # 'Camera (MP)' sütunu siliniyor ve yeni sütunlar ekleniyor
    veriler.drop('Camera (MP)', axis=1, inplace=True)
    veriler = pd.concat([veriler, kamera], axis=1)

    veriler['Price ($)'] = veriler['Price ($)'].str.replace('$', '')  # $
işareti kaldırılır
    veriler['Price ($)'] = veriler['Price ($)'].str.replace(',',
'').astype(float)  # Virgül ile ayrılmış binlik haneler kaldırılır ve float
veri tipine dönüştürülür
    veriler = veriler.rename(columns={'Price ($)': 'Price'})  # Sütun adı
'Price' olarak değiştirilir

    veriler.fillna(0, inplace=True)
    # Verileri tablo halinde düzenleme
    print(veriler.to_string(index=False))

    le = LabelEncoder()
    veriler['Brand'] = le.fit_transform(veriler['Brand'])
    num_to_cat = dict(zip(le.transform(le.classes_), le.classes_))
    print(num_to_cat)
    veriler['Brand'] = veriler['Brand'].astype('category')
    veriler['Model'] = veriler['Model'].astype('category')
    veriler['Brand'] = veriler['Brand'].cat.codes
    veriler['Model'] = veriler['Model'].cat.codes



    # Verileri tablo halinde düzenleme
    print(veriler.to_string(index=False))

    #veri sutün analizi ve istatistiği
    print("Brand: ", veriler["Brand"].nunique(), " farklı marka var.")
    print("Model: ", veriler["Model"].nunique(), " farklı model var.")
    print("Storage: Ortalama=", veriler["Storage "].astype(int).mean(),
"GB, Standart Sapma=", veriler["Storage "].astype(int).std(), "GB.")
    print("RAM: Ortalama=", veriler["RAM "].astype(int).mean(), "GB,
Standart Sapma=", veriler["RAM "].astype(int).std(), "GB.")
    print("Screen Size (inches 1) : Ortalama=", veriler["Screen Size
(inches 1)"].astype(float).mean(), "inç, Standart Sapma=", veriler["Screen
Size (inches 1)"].astype(float).std(), "inç.")
    print("Screen Size (inches 2): Ortalama=", veriler["Screen Size (inches
2)"].astype(float).mean(), "inç, Standart Sapma=", veriler["Screen Size
(inches 2)"].astype(float).std(), "inç.")
    print("Camera (MP) 1: Ortalama=", veriler["Camera 1"].mean(), "MP,
Standart Sapma=", veriler["Camera 1"].std(), "MP.")
    print("Camera (MP) 2: Ortalama=", veriler["Camera 2"].mean(), "MP,
Standart Sapma=", veriler["Camera 2"].std(), "MP.")
```

```
    print("Camera (MP) 3: Ortalama=", veriler["Camera 3"].mean(), "MP,
Standart Sapma=", veriler["Camera 3"].std(), "MP.")
    print("Camera (MP) 4: Ortalama=", veriler["Camera 4"].mean(), "MP,
Standart Sapma=", veriler["Camera 4"].std(), "MP.")
    print("Battery Capacity (mAh): Ortalama=", veriler["Battery Capacity
(mAh)"].mean(), "mAh, Standart Sapma=", veriler["Battery Capacity
(mAh)"].std(), "mAh.")
    print("Price ($): Ortalama=", veriler["Price"].mean(), "$, Standart
Sapma=", veriler["Price"].std(), "$.")
    Linear_Regression(veriler)
    Brand_Lojistik_Regresyon(veriler)
    PCA_data(veriler)
```

Önce pandas kütüphanesi ile csv dosyası okunur. Daha sonra istatistik hesabı için verileri temizliyoruz ve düzenliyoruz. Bunun için önce Storage verilerindeki GB yazılarını silerek bu sütunu düzenliyoruz. Daha sonra aynı işlemi RAM için yapıyoruz. Sonra Screen Size sütununda bazı telefon modellerinde iki ekran boyutu verildiği için ve bu veriler arasında + işareti olduğundan dolayı + işaretinin solu ve sağı olarak 2 ayrı sütuna ayırıyoruz. Daha sonra Kameranın megapiksel değerleri ayrıştırılıyor ve eksik değerler NaN ile değiştiriliyor, birden fazla kamerası olan modeller için yeni sütunlar ekleniyor. Sonra Price sütunundaki $ işaretlerini kaldırıyoruz ve virgülle ayrılmış binlik veriler float tipine dönüştürülüyor. Daha sonra doğrusal regresyonun doğru çalışabilmesi için fillna fonksiyonu tablodaki tüm NaN değerler 0'a dönüştürülür. Marka ve model verileri string olduğu için labelEncoder fonksiyonu ile kategorik verileri sayısal verilere dönüştürüyoruz. Son olarak tablonun yeni hali ekrana print edilir.

2-I-

```
def Linear_Regression(veriler):
    # Özellikleri ve hedef değişkeni ayırma
    X = veriler.drop('Price', axis=1)
    y = veriler['Price']

    # Eğitim ve test setlerini ayırma
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    # Doğrusal regresyon modelini oluşturma ve eğitim
    lr_model = LinearRegression()
    lr_model.fit(X_train, y_train)

    # Test seti üzerinde tahmin yapma
    y_pred = lr_model.predict(X_test)

    # Model performansını değerlendirme
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    print("Test hatası (RMSE): {:.2f}".format(rmse))


    # Test seti üzerinde tahmin yapma
    y_pred = lr_model.predict(X_test)

    # Tahmin sonuçlarını yazdırma
    print('Tahmin edilen fiyatlar: ', y_pred)
    print('Gerçek fiyatlar: ', y_test)

    # Model performansını değerlendirme
```

```
    from sklearn.metrics import r2_score
    r2 = r2_score(y_test, y_pred)
    print('Model performansı (R^2): ', r2)   #1'e ne kadar yakınsa o kadar
iyi


    # Özellikleri ve hedef değişkeni ayırma
    X = veriler.drop('Price', axis=1)
    y = veriler['Price']

    # Eğitim ve test setlerini ayırma
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    # Doğrusal regresyon modelini oluşturma ve eğitim
    lr_model = LinearRegression()
    lr_model.fit(X_train, y_train)

    # Tahmini fiyat ile gerçek fiyat arasındaki farkı hesaplama
    y_diff = lr_model.predict(X_test) - y_test

    # En yakın fiyat farkını bulma
    min_diff = np.abs(y_diff).min()

    # En yakın fiyat farkına sahip modeli bulma
    closest_model_idx = np.where(np.abs(y_diff) == min_diff)[0][0]
    closest_model = X_test.iloc[[closest_model_idx]]

    # Sonuçları yazdırma
    print("En yakın model:")
    print(closest_model)
    print("Tahmini fiyatı:
${:.2f}".format(lr_model.predict(closest_model)[0]))
```

Bağımlı ve bağımsız değişkenler belirlenir. Sonra eğitim ve test setleri ayrılır. Doğrusal regresyon modeli oluşturulur ve lr_model.fit(X_train, y_train) ile eğitim gerçekleştirilir. Daha sonra test seti üzerinden tahmin yapılır. Tahmini fiyat ile gerçek fiyat arasındaki tahmin hesaplanır. En yakın model bulunur ve ekrana çıktı verilir.

2-II-

```
def Brand_Lojistik_Regresyon(veriler):
    #----------------markalara lojistik regresyon---------

    # Marka sütununu kategorik sütuna dönüştürün
    veriler['Brand'] = pd.Categorical(veriler['Brand'])
    veriler['Brand'] = veriler['Brand'].cat.codes

    # Samsung markasına ait verileri seçin
    samsung_veriler = veriler[veriler['Brand'] == 11]

    # Özellikleri ve hedef değişkeni ayırma
    X = samsung_veriler.drop('Model', axis=1)
    y = samsung_veriler['Model']

    # Eğitim ve test setlerini ayırma
    X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
test_size=0.2, random_state=42)

    # Lojistik regresyon modelini oluşturma ve eğitim
    lr_model = LogisticRegression()
    lr_model.fit(X_train, y_train)

    # Test setinde doğruluğu hesaplama
    y_pred = lr_model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    # Doğruluğu yazdırma
    print("Samsung için doğruluk:", accuracy)

    # Apple markasına ait verileri seçme
    apple_data = veriler[veriler['Brand'] == 0]

    # Özellikleri ve hedef değişkeni ayırma
    X_apple = apple_data.drop('Model', axis=1)
    y_apple = apple_data['Model']

    # Eğitim ve test setlerini ayırma
    X_train_apple, X_test_apple, y_train_apple, y_test_apple =
train_test_split(X_apple, y_apple, test_size=0.2, random_state=42)

    # Lojistik regresyon modelini oluşturma ve eğitim
    lr_model_apple = LogisticRegression()
    lr_model_apple.fit(X_train_apple, y_train_apple)

    # Test verileri ile tahmin yapma ve doğruluk hesaplama
    y_pred_apple = lr_model_apple.predict(X_test_apple)
    accuracy_apple = accuracy_score(y_test_apple, y_pred_apple)

    # Sonuçları yazdırma
    print("Apple markası için doğruluk:", accuracy_apple)

    # Xiaomi markasına ait verileri seçme
    xiaomi_data = veriler[veriler['Brand'] == 15]

    # Özellikleri ve hedef değişkeni ayırma
    X_xiaomi = xiaomi_data.drop('Model', axis=1)
    y_xiaomi = xiaomi_data['Model']

    # Eğitim ve test setlerini ayırma
    X_train_xiaomi, X_test_xiaomi, y_train_xiaomi, y_test_xiaomi =
train_test_split(X_xiaomi, y_xiaomi, test_size=0.2, random_state=42)

    # Lojistik regresyon modelini oluşturma ve eğitim
    lr_model_xiaomi = LogisticRegression()
    lr_model_xiaomi.fit(X_train_xiaomi, y_train_xiaomi)

    # Test verileri ile tahmin yapma ve doğruluk hesaplama
    y_pred_xiaomi = lr_model_xiaomi.predict(X_test_xiaomi)
    accuracy_xiaomi = accuracy_score(y_test_xiaomi, y_pred_xiaomi)

    # Sonuçları yazdırma
    print("Xiaomi markası için doğruluk:", accuracy_xiaomi)
```

Samsung, Xiaomi ve Apple için ayrı ayrı lojistik regresyon modeli oluşturulur. Eğitim ve test setleri ayrılır. Test setinde doğruluk hesaplanır.

3-

```
def PCA_data(veriler):

    #-----------------PCA--------------------
    from sklearn.preprocessing import StandardScaler
    from sklearn.decomposition import PCA
    import matplotlib.pyplot as plt


    # Yalnızca sayısal sütunları seçin
    numeric_cols = ['Storage ', 'RAM ', 'Battery Capacity (mAh)', 'Price']
    X = veriler[numeric_cols]

    # Verileri ölçeklendirin
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # PCA modelini oluşturun
    pca = PCA(n_components=1)

    # PCA modelini eğitin
    pca.fit(X_scaled)

    # PCA dönüşümünü uygulayın
    X_pca = pca.transform(X_scaled)

    # Elde edilen PCA bileşenini veri kümesine ekleyin
    veriler['PCA Component'] = X_pca

    # PCA bileşeninin fiyatla ilişkisine bakın
    veriler.plot.scatter(x='PCA Component', y='Price')
    plt.show()
    Linear_Regression(veriler)
    Brand_Lojistik_Regresyon(veriler)

def main():
    print(veriler_read())


main()
```

Bir veri kümesinde PCA ile boyut azaltma yaparak verilerin boyutunu azaltabiliriz. PCA veri kümesindeki varyansın büyük bir bölümünü açıklayan yeni bir özellik seti oluşturur. PCA'yı uygulamak için normal şartlarda öncelikle verilerin normalize edilmesi gerekir. Ama biz daha önce verilerimizi sayısal değerlere dönüştürdüğümüz için normalleştirme yapmamıza gerek yoktur. Yalnızca sayısal sütunları seçiyoruz ve verileri ölçeklendiriyoruz. PCA modelini oluşturuyoruz. Daha sonra modeli eğitiyoruz ve dönüşümü uyguluyoruz. Elde edilen PCA bileşenini veri kümesine ekliyoruz.

4-I-

```
# Model performansını değerlendirme
mse = mean_squared_error(y_test, y_pred)
```

```
rmse = np.sqrt(mse)
print("Test hatası (RMSE): {:.2f}".format(rmse))

# Model performansını değerlendirme
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print('Model performansı (R^2): ', r2)    #1'e ne kadar yakınsa o kadar iyi
```

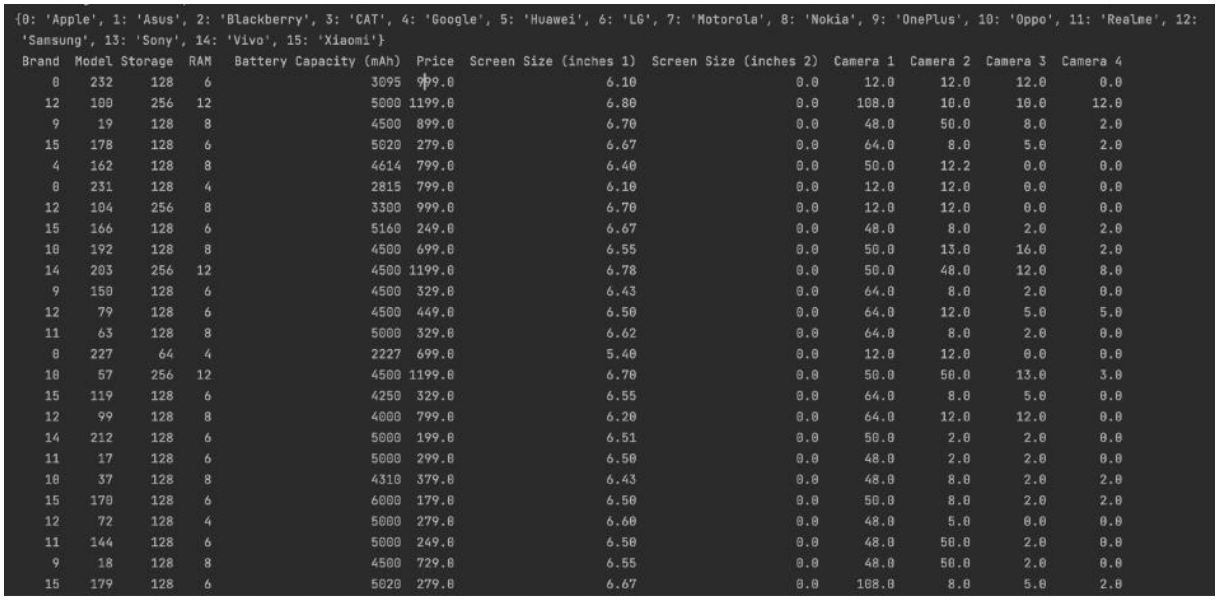
Model performansını değerlendirdik. Bu sonuç 1'e ne kadar yakınsa o kadar iyi bir performans elde etmiş oluruz. 2. Ve 3. Sorular için ayrı ayrı performans değerlendirme yapılır.

4-II-

```
veriler.plot.scatter(x='PCA Component', y='Price')
plt.show()
```

Verileri matplotlib ile görselleştirilir.

ÇIKTILAR



```
{0: 'Apple', 1: 'Asus', 2: 'Blackberry', 3: 'CAT', 4: 'Google', 5: 'Huawei', 6: 'LG', 7: 'Motorola', 8: 'Nokia', 9: 'OnePlus', 10: 'Oppo', 11: 'Realme', 12:
 'Samsung', 13: 'Sony', 14: 'Vivo', 15: 'Xiaomi'}
 Brand  Model Storage  RAM   Battery Capacity (mAh)  Price  Screen Size (inches 1)  Screen Size (inches 2)  Camera 1  Camera 2  Camera 3  Camera 4
     0    232     128    6                    3095  999.0                    6.10                     0.0      12.0      12.0      12.0       0.0
    12    100     256   12                    5000 1199.0                    6.80                     0.0     108.0      10.0      10.0      12.0
     9     19     128    8                    4500  899.0                    6.70                     0.0      48.0      50.0       8.0       2.0
    15    178     128    6                    5020  279.0                    6.67                     0.0      64.0       8.0       5.0       2.0
     4    162     128    8                    4614  799.0                    6.40                     0.0      50.0      12.2       0.0       0.0
     0    231     128    4                    2815  799.0                    6.10                     0.0      12.0      12.0       0.0       0.0
    12    104     256    8                    3300  999.0                    6.70                     0.0      12.0      12.0       0.0       0.0
    15    166     128    6                    5160  249.0                    6.67                     0.0      48.0       8.0       2.0       2.0
    10    192     128    8                    4500  699.0                    6.55                     0.0      50.0      13.0      16.0       2.0
    14    203     256   12                    4500 1199.0                    6.78                     0.0      50.0      48.0      12.0       8.0
     9    150     128    6                    4500  329.0                    6.43                     0.0      64.0       8.0       2.0       0.0
    12     79     128    6                    4500  449.0                    6.50                     0.0      64.0      12.0       5.0       5.0
    11     63     128    8                    5000  329.0                    6.62                     0.0      64.0       8.0       2.0       0.0
     0    227      64    4                    2227  699.0                    5.40                     0.0      12.0      12.0       0.0       0.0
    10     57     256   12                    4500 1199.0                    6.70                     0.0      50.0      50.0      13.0       3.0
    15    119     128    6                    4250  329.0                    6.55                     0.0      64.0       8.0       5.0       0.0
    12     99     128    8                    4000  799.0                    6.20                     0.0      64.0      12.0      12.0       0.0
    14    212     128    6                    5000  199.0                    6.51                     0.0      50.0       2.0       2.0       0.0
    11     17     128    6                    5000  299.0                    6.50                     0.0      48.0       2.0       2.0       0.0
    10     37     128    8                    4310  379.0                    6.43                     0.0      48.0       8.0       2.0       2.0
    15    170     128    6                    6000  179.0                    6.50                     0.0      50.0       8.0       2.0       2.0
    12     72     128    4                    5000  279.0                    6.60                     0.0      48.0       5.0       0.0       0.0
    11    144     128    6                    5000  249.0                    6.50                     0.0      48.0      50.0       2.0       0.0
     9     18     128    8                    4500  729.0                    6.55                     0.0      48.0      50.0       2.0       0.0
    15    179     128    6                    5020  279.0                    6.67                     0.0     108.0       8.0       5.0       2.0
```

```
Brand:  16  farklı marka var.
Model:  239  farklı model var.
Storage: Ortalama= 123.04668304668304 GB, Standart Sapma= 64.96315956786346 GB.
RAM: Ortalama= 5.837837837837838 GB, Standart Sapma= 2.4319797208860985 GB.
Screen Size (inches 1) : Ortalama= 6.471253071253072 inç, Standart Sapma= 0.32054241641833114 inç.
Screen Size (inches 2): Ortalama= 0.089582309582309581 inç, Standart Sapma= 0.1933158228676778 inç.
Camera (MP) 1: Ortalama= 43.319410319410316 MP, Standart Sapma= 24.66902470603544 MP.
Camera (MP) 2: Ortalama= 9.125798525798526 MP, Standart Sapma= 10.174802150734862 MP.
Camera (MP) 3: Ortalama= 3.6412776412776413 MP, Standart Sapma= 4.982906179400769 MP.
Camera (MP) 4: Ortalama= 0.9835380835380836 MP, Standart Sapma= 1.7194767398874728 MP.
Battery Capacity (mAh): Ortalama= 4676.4766584766585 mAh, Standart Sapma= 797.193713329488 mAh.
Price ($): Ortalama= 408.3144963144963 $, Standart Sapma= 299.6847676505315 $.
Test hatası (RMSE): 120.93
```

```
Tahmin edilen fiyatlar:  [ 163.88683393   99.35588452  406.34369932  326.31763666   95.05264359
  671.65661082   97.34576609  283.3237015   234.04320656  399.22499163
 1443.52039708  590.89423619  141.87062145  146.84439867  359.54076548
  289.65375484  434.51688694  460.75759794 1193.43669985  298.43445139
  251.71613449  984.418531    146.84439867  474.01252311  179.54211136
  334.3998123  1026.1307058   285.90806029  346.29180755  345.70652979
  476.47935275  467.0031704   927.7841528   287.94114876  187.28544458
  243.63589057  146.84439867  407.87791407  415.61679215  784.73863372
  262.2468545   275.01272748  236.92090993  480.19401791  246.31406422
  973.43368816  585.41612693  313.55550056  258.70494758  345.31618151
  138.95118013  586.84012514  159.92322413  395.01330375  369.51297407
  505.29473668  345.31618151  460.75759794  495.68873881  294.5686293
 1836.95495663  394.62295547  428.53228173  287.07910513 1823.62074082
  184.61841546  336.37992005 1637.45356573  210.77477369  172.06301908
  173.82578043  822.969786    138.51997983  389.25282733   71.96711683
  671.65661082  562.6846491   264.58030143  462.72993752  418.91602485
 1398.46057865  461.14794622]
```

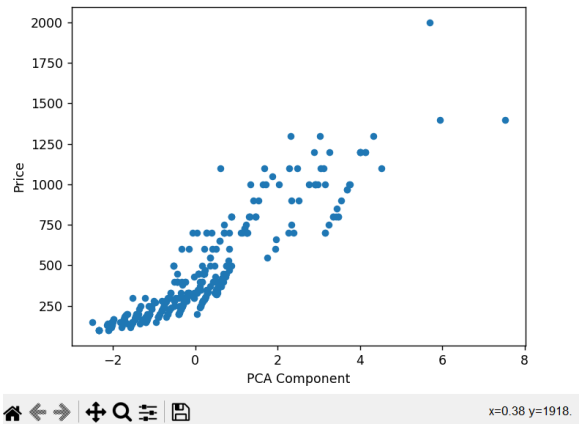
```
Gerçek fiyatlar:  70     169.0
218    139.0
258    349.0
33     259.0
42     189.0
       ...
90     399.0
379    449.0
341    399.0
376    999.0
357    399.0
Name: Price, Length: 82, dtype: float64
Model performansı (R^2):  0.8399818753567785
En yakın model:
    Brand  Model Storage  RAM   ...  Camera 1  Camera 2  Camera 3  Camera 4
76     11    145      64    4   ...      50.0       2.0       0.0       0.0
```

```
Tahmini fiyatı: $138.52
Samsung için doğruluk: 0.6666666666666666
Apple markası için doğruluk: 0.3333333333333333
Xiaomi markası için doğruluk: 0.07142857142857142
Test hatası (RMSE): 0.00
Tahmin edilen fiyatlar:  [ 169.  139.  349.  259.  189.  799.  179.  449.  199.  289. 1399.  799.
  129.  199.  249.  299.  469.  399. 1199.  299.  239. 1049.  229.  449.
  139.  299.  699.  269.  279.  329.  549.  349.  999.  269.  249.  159.
  199.  299.  399.  699.  199.  179.  219.  399.  299.  899.  599.  199.
  299.  399.  189.  249.  179.  369.  299.  699.  369.  329.  499.  199.
 1299.  499.  449.  249.  799.  189.  349. 1199.  179.  139.  149. 1099.
  139.  369.  149.  799.  699.  399.  449.  399.  999.  399.]
Gerçek fiyatlar:  70     169.0
218    139.0
258    349.0
33     259.0
42     189.0
       ...
90     399.0
379    449.0
341    399.0
376    999.0
357    399.0
Name: Price, Length: 82, dtype: float64
Model performansı (R^2):  1.0
En yakın model:
     Brand  Model Storage  RAM  ...  Camera 2  Camera 3  Camera 4  PCA Component
155      7    126      128    4  ...       8.0       2.0       0.0      -0.761211

[1 rows x 12 columns]
Tahmini fiyatı: $249.00
Samsung için doğruluk: 0.6666666666666666
Apple markası için doğruluk: 0.3333333333333333
Xiaomi markası için doğruluk: 0.21428571428571427
```

```
Model performansı (R^2):  0.8399818753567785
```

```
Model performansı (R^2):  1.0
```



Eda Menekşeyurt