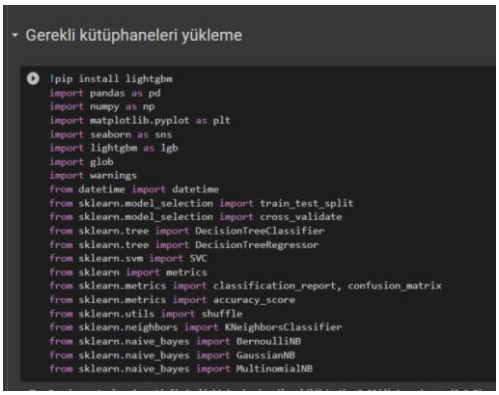
PROJE RAPORU

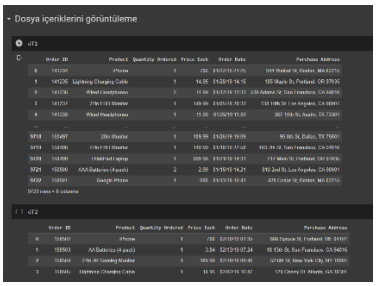
Rapor Tarihi: 02.07.2023 Proje Adı: Satış Verileri Tabanlı Ürün Sınıflandırma Hazırlayan: Elif Top

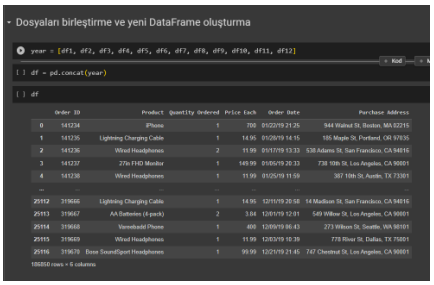
Yapılan İşlemler

1. İlk olarak, gerekli kütüphaneleri (pandas, numpy, matplotlib, seaborn, lightgbm, vb.) yüklüyoruz.

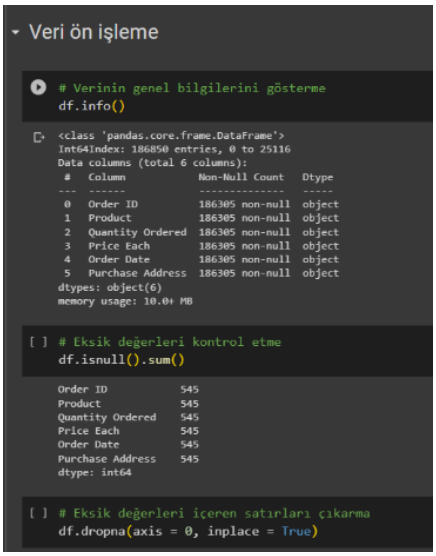


2. df1 ile df12 arasında 12 ayrı CSV dosyasını okuyarak yeni veri seti (DataFrame) oluşturuyoruz.

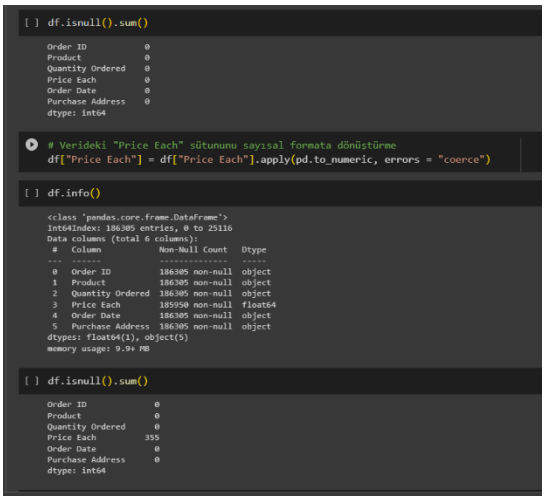




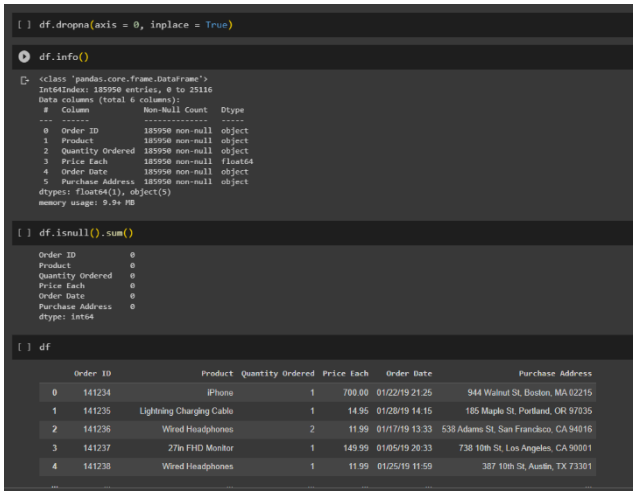
3. df veri setinin genel bilgilerini (info) ve eksik değerlerini kontrol ediyoruz ve eksik değerleri içeren satırları (NaN değerleri) dropna fonksiyonuyla veri setinden çıkarıyoruz.



4. "Price Each" sütunundaki verileri sayısal formata (numeric) dönüştürüyoruz, tekrar eksik değerleri kontrol ediyoruz ve eksik değer içermeyen bir veri elde ediyoruz.



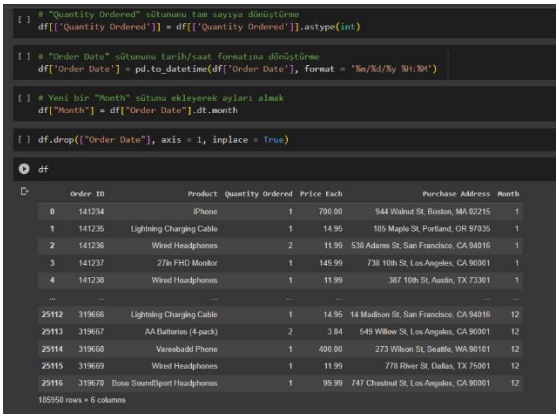
```
[ ] df.isnull().sum()

Order ID          0
Product           0
Quantity Ordered  0
Price Each        0
Order Date        0
Purchase Address  0
dtype: int64

# Verideki "Price Each" sütununu sayısal formata dönüştürme
df["Price Each"] = df["Price Each"].apply(pd.to_numeric, errors = "coerce")

[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 186305 entries, 0 to 25116
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          186305 non-null  object
 1   Product           186305 non-null  object
 2   Quantity Ordered  186305 non-null  object
 3   Price Each        185950 non-null  float64
 4   Order Date        186305 non-null  object
 5   Purchase Address  186305 non-null  object
dtypes: float64(1), object(5)
memory usage: 9.9+ MB

[ ] df.isnull().sum()

Order ID          0
Product           0
Quantity Ordered  0
Price Each      355
Order Date        0
Purchase Address  0
dtype: int64
```



```
[ ] df.dropna(axis = 0, inplace = True)

df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 25116
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          185950 non-null  object
 1   Product           185950 non-null  object
 2   Quantity Ordered  185950 non-null  object
 3   Price Each        185950 non-null  float64
 4   Order Date        185950 non-null  object
 5   Purchase Address  185950 non-null  object
dtypes: float64(1), object(5)
memory usage: 9.9+ MB

[ ] df.isnull().sum()

Order ID          0
Product           0
Quantity Ordered  0
Price Each        0
Order Date        0
Purchase Address  0
dtype: int64

[ ] df
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	141234	iPhone	1	700.00	01/22/19 21:25	944 Walnut St, Boston, MA 02215
1	141235	Lightning Charging Cable	1	14.95	01/28/19 14:15	185 Maple St, Portland, OR 97035
2	141236	Wired Headphones	2	11.99	01/17/19 13:33	538 Adams St, San Francisco, CA 94016
3	141237	27in FHD Monitor	1	149.99	01/05/19 20:33	738 10th St, Los Angeles, CA 90001
4	141238	Wired Headphones	1	11.99	01/25/19 11:59	387 10th St, Austin, TX 73301

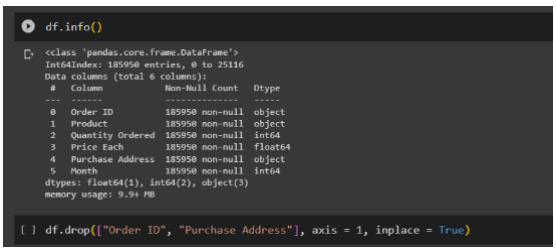
5. "Quantity Ordered" sütununu tam sayıya, "Order Date" sütununu tarih-saat formatına dönüştürüyoruz. Yeni bir "Month" sütunu ekliyoruz. Gereksiz sütunları ("Order Date", "Order ID", "Purchase Address") veri setinden çıkarıyoruz.



```
[ ] # "Quantity Ordered" sütununu tam sayıya dönüştürme
df['Quantity Ordered'] = df['Quantity Ordered'].astype(int)

[ ] # "Order Date" sütununu tarih/saat formatına dönüştürme
df['Order Date'] = pd.to_datetime(df['Order Date'], format = '%m/%d/%y %H:%M')

[ ] # Yeni bir "Month" sütunu ekleyerek ayları almak
df["Month"] = df["Order Date"].dt.month

[ ] df.drop(["Order Date"], axis = 1, inplace = True)

df
```

	Order ID	Product	Quantity Ordered	Price Each	Purchase Address	Month
0	141234	iPhone	1	700.00	944 Walnut St, Boston, MA 02215	1
1	141235	Lightning Charging Cable	1	14.95	185 Maple St, Portland, OR 97035	1
2	141236	Wired Headphones	2	11.99	538 Adams St, San Francisco, CA 94016	1
3	141237	27in FHD Monitor	1	149.99	738 10th St, Los Angeles, CA 90001	1
4	141238	Wired Headphones	1	11.99	387 10th St, Austin, TX 73301	1
...	...	...	...	...	...	...
25112	319668	Lightning Charging Cable	1	14.95	14 Madison St, San Francisco, CA 94016	12
25113	319667	AA Batteries (4-pack)	2	3.84	549 Willow St, Los Angeles, CA 90001	12
25114	319668	Vareebadd Phone	1	400.00	273 Wilson St, Seattle, WA 98101	12
25115	319669	Wired Headphones	1	11.99	778 River St, Dallas, TX 75001	12
25116	319670	Bose SoundSport Headphones	1	99.99	747 Chestnut St, Los Angeles, CA 90001	12

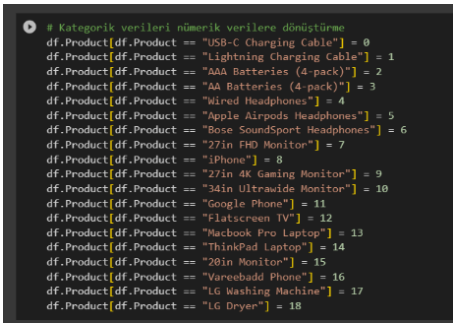
185950 rows × 6 columns



```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 25116
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          185950 non-null  object
 1   Product           185950 non-null  object
 2   Quantity Ordered  185950 non-null  int64
 3   Price Each        185950 non-null  float64
 4   Purchase Address  185950 non-null  object
 5   Month             185950 non-null  int64
dtypes: float64(1), int64(2), object(3)
memory usage: 9.9+ MB

[ ] df.drop(["Order ID", "Purchase Address"], axis = 1, inplace = True)
```

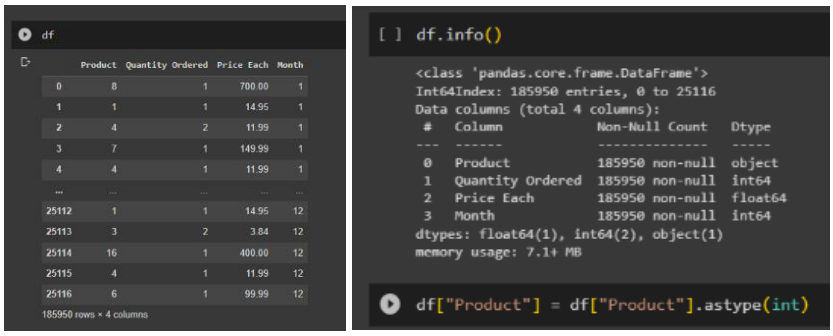
6. " Product" sütunundaki değerleri ve tekrar etme sayılarını gözlemliyoruz ve ürünlerin kategorik değerlerini sayısal değerlere eşliyoruz.

```
[ ] # "Product" sütunundaki değerleri ve ne kadar tekrar ettiklerini belirleme
df["Product"].value_counts()

USB-C Charging Cable          21903
Lightning Charging Cable      21658
AAA Batteries (4-pack)        20641
AA Batteries (4-pack)         20577
Wired Headphones              18882
Apple Airpods Headphones      15549
Bose SoundSport Headphones    13325
27in FHD Monitor               7507
iPhone                         6842
27in 4K Gaming Monitor         6230
34in Ultrawide Monitor         6181
Google Phone                   5525
Flatscreen TV                  4800
Macbook Pro Laptop             4724
ThinkPad Laptop                4128
20in Monitor                   4101
Vareebadd Phone                2065
LG Washing Machine              666
LG Dryer                        646
Name: Product, dtype: int64
```



```
# Kategorik verileri nümerik verilere dönüştürme
df.Product[df.Product == "USB-C Charging Cable"] = 0
df.Product[df.Product == "Lightning Charging Cable"] = 1
df.Product[df.Product == "AAA Batteries (4-pack)"] = 2
df.Product[df.Product == "AA Batteries (4-pack)"] = 3
df.Product[df.Product == "Wired Headphones"] = 4
df.Product[df.Product == "Apple Airpods Headphones"] = 5
df.Product[df.Product == "Bose SoundSport Headphones"] = 6
df.Product[df.Product == "27in FHD Monitor"] = 7
df.Product[df.Product == "iPhone"] = 8
df.Product[df.Product == "27in 4K Gaming Monitor"] = 9
df.Product[df.Product == "34in Ultrawide Monitor"] = 10
df.Product[df.Product == "Google Phone"] = 11
df.Product[df.Product == "Flatscreen TV"] = 12
df.Product[df.Product == "Macbook Pro Laptop"] = 13
df.Product[df.Product == "ThinkPad Laptop"] = 14
df.Product[df.Product == "20in Monitor"] = 15
df.Product[df.Product == "Vareebadd Phone"] = 16
df.Product[df.Product == "LG Washing Machine"] = 17
df.Product[df.Product == "LG Dryer"] = 18
```

7. Verimizi tekrar gözlemliyoruz ve veri içerisindeki "Product" sütununu tam sayıya dönüştürüyoruz.



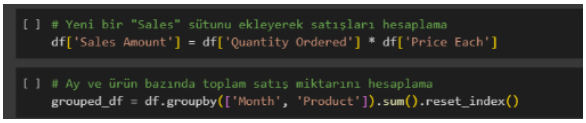
```
df
```

	Product	Quantity Ordered	Price Each	Month
0	8	1	700.00	1
1	1	1	14.95	1
2	4	2	11.99	1
3	7	1	149.99	1
4	4	1	11.99	1
...	...	...	...	...
25112	1	1	14.95	12
25113	3	2	3.84	12
25114	16	1	400.00	12
25115	4	1	11.99	12
25116	6	1	99.99	12

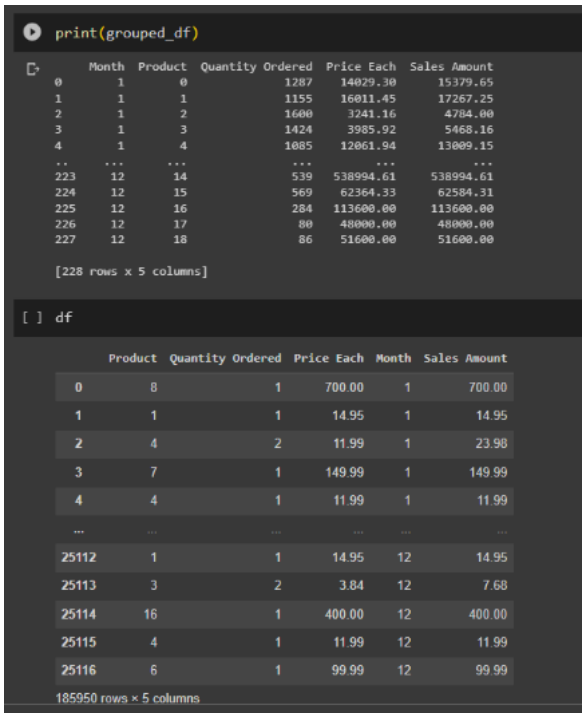
185950 rows × 4 columns

```
[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 25116
Data columns (total 4 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Product           185950 non-null  object
 1   Quantity Ordered  185950 non-null  int64
 2   Price Each        185950 non-null  float64
 3   Month             185950 non-null  int64
dtypes: float64(1), int64(2), object(1)
memory usage: 7.1+ MB

df["Product"] = df["Product"].astype(int)
```

8. "Quantity Ordered" ve "Price Each" sütunlarını çarparak "Sales Amount" adında yeni bir sütun ekliyoruz. "Month" ve "Product" sütunlarına göre veriyi gruplayarak toplam satış miktarını hesaplıyoruz.



```
[ ] # Yeni bir "Sales" sütunu ekleyerek satışları hesaplama
    df['Sales Amount'] = df['Quantity Ordered'] * df['Price Each']

[ ] # Ay ve ürün bazında toplam satış miktarını hesaplama
    grouped_df = df.groupby(['Month', 'Product']).sum().reset_index()
```



```
print(grouped_df)
```

	Month	Product	Quantity Ordered	Price Each	Sales Amount
0	1	0	1287	14029.30	15379.65
1	1	1	1155	16011.45	17267.25
2	1	2	1600	3241.16	4784.00
3	1	3	1424	3985.92	5468.16
4	1	4	1085	12061.94	13009.15
..	...	...	...	...	...
223	12	14	539	538994.61	538994.61
224	12	15	569	62364.33	62584.31
225	12	16	284	113600.00	113600.00
226	12	17	80	48000.00	48000.00
227	12	18	86	51600.00	51600.00

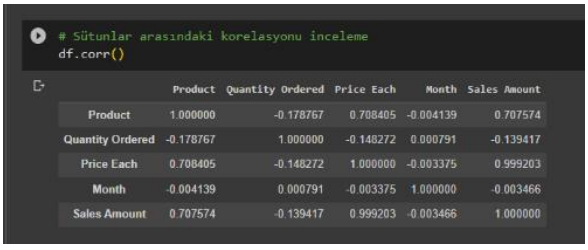
[228 rows x 5 columns]

```
[ ] df
```

	Product	Quantity Ordered	Price Each	Month	Sales Amount
0	8	1	700.00	1	700.00
1	1	1	14.95	1	14.95
2	4	2	11.99	1	23.98
3	7	1	149.99	1	149.99
4	4	1	11.99	1	11.99
...	...	...	...	...	...
25112	1	1	14.95	12	14.95
25113	3	2	3.84	12	7.68
25114	16	1	400.00	12	400.00
25115	4	1	11.99	12	11.99
25116	6	1	99.99	12	99.99

185950 rows × 5 columns

9. Verinin korelasyon matrisini hesaplıyoruz.

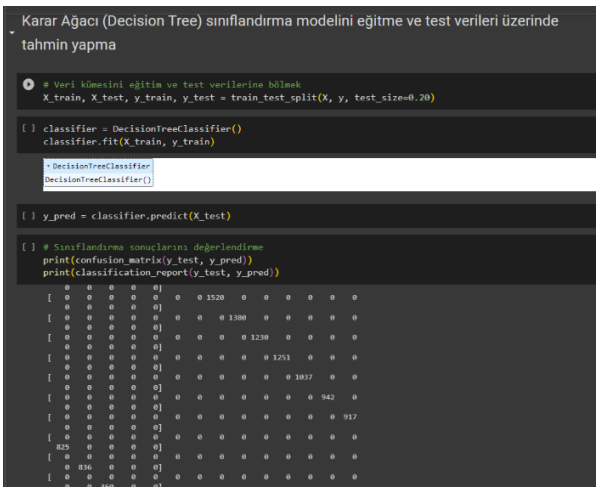


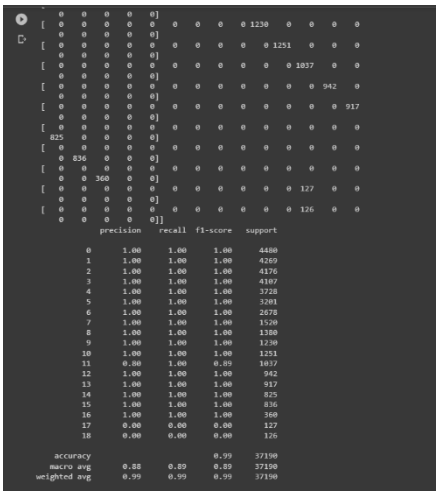
```
# Sütunlar arasındaki korelasyonu inceleme
df.corr()
```

	Product	Quantity Ordered	Price Each	Month	Sales Amount
Product	1.000000	-0.178767	0.708405	-0.004139	0.707574
Quantity Ordered	-0.178767	1.000000	-0.148272	0.000791	-0.139417
Price Each	0.708405	-0.148272	1.000000	-0.003375	0.999203
Month	-0.004139	0.000791	-0.003375	1.000000	-0.003466
Sales Amount	0.707574	-0.139417	0.999203	-0.003466	1.000000

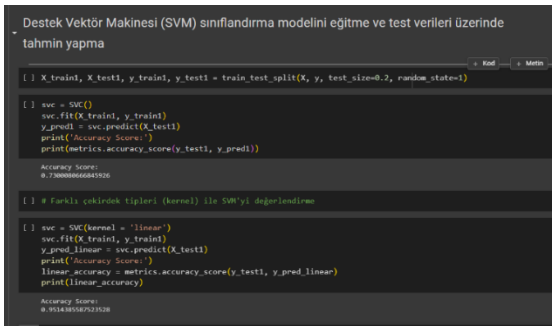
10. Hedef değişkenimizi "Product" olarak belirliyoruz ve geri kalan sütünları bağımsız değişken olarak atıyoruz.

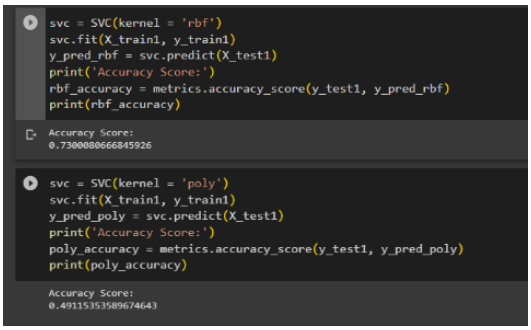
11. Karar ağacı (DecisionTreeClassifier) kullanarak sınıflandırma modelini eğitiyoruz ve test verileri üzerinde tahmin yapıyoruz. Ardından, sınıflandırma sonuçlarını değerlendiriyoruz.

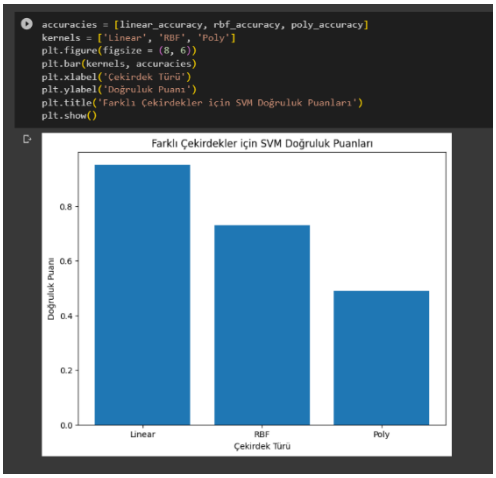




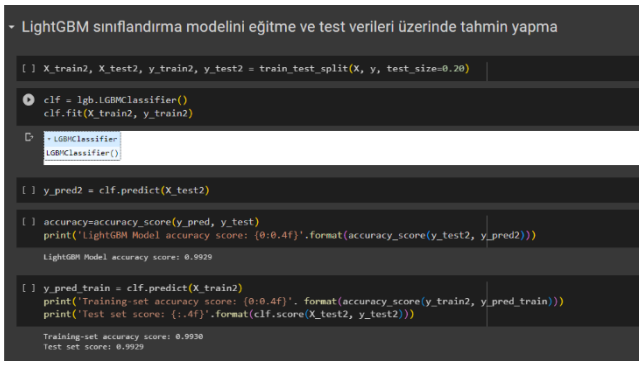
12. Destek Vektör Makinesi (SVM) kullanarak sınıflandırma modelini eğitiyoruz ve test verileri üzerinde tahmin yapıyoruz. Farklı çekirdek tipleri (kernel) ile SVM'yi değerlendiriyoruz.

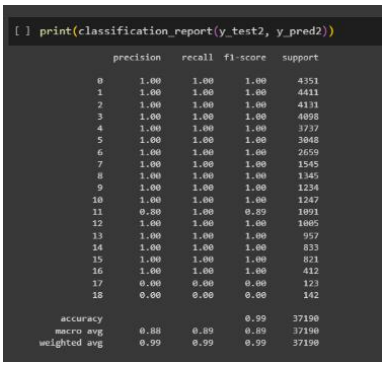




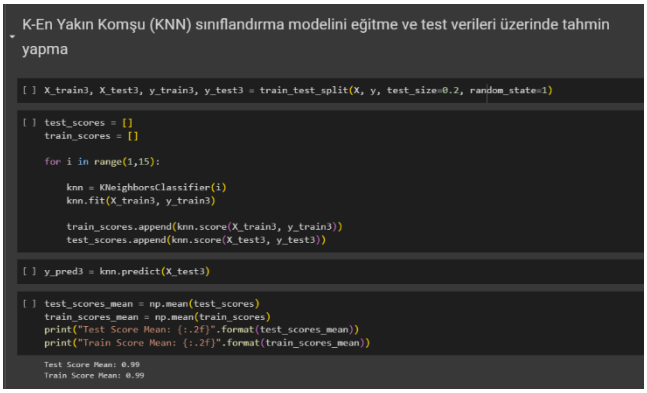


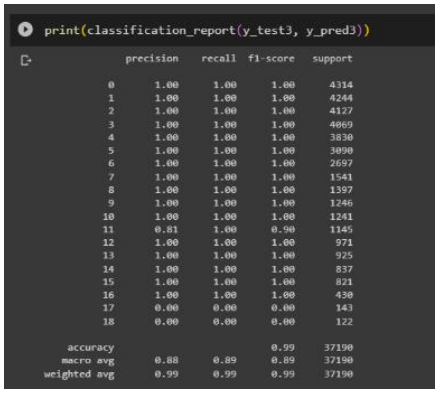
13. LightGBM sınıflandırma modelini eğitiyoruz ve test verileri üzerinde tahmin yapıyoruz. Doğruluk (accuracy) puanını hesaplıyoruz ve sınıflandırma sonuçlarını değerlendiriyoruz.



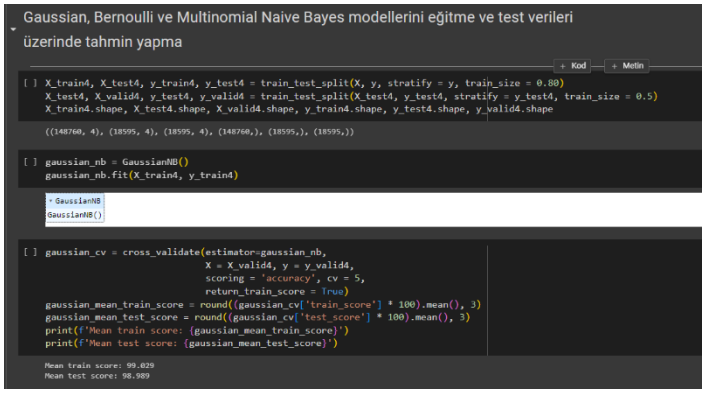


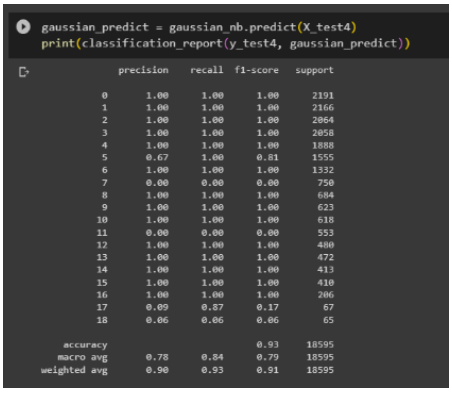
14. K-En Yakın Komşu (KNN) sınıflandırma modelini eğitiyoruz ve test verileri üzerinde tahmin yapıyoruz. Eğitim ve test skorlarını hesaplıyoruz ve sınıflandırma sonuçlarını değerlendiriyoruz.





15. Gaussian, Bernoulli ve Multinomial Naive Bayes modellerini eğitiyoruz ve test verileri üzerinde tahmin yapıyoruz. Eğitim ve test skorlarını hesaplıyoruz ve sınıflandırma sonuçlarını değerlendiriyoruz.





```
[ ] bernoulli_nb = BernoulliNB()
    bernoulli_nb.fit(X_train4, y_train4)

    BernoulliNB()
    BernoulliNB()

    bernoulli_cv = cross_validate(estimator = bernoulli_nb,
                                  X = X_valid4, y = y_valid4,
                                  cv=10, return_train_score=True)
    bernoulli_mean_train_score = round((bernoulli_cv['train_score'] * 100).mean(), 3)
    bernoulli_mean_test_score = round((bernoulli_cv['test_score'] * 100).mean(), 3)
    print(f'Mean train score: {bernoulli_mean_train_score}')
    print(f'Mean test score: {bernoulli_mean_test_score}')

    Mean train score: 11.777
    Mean test score: 11.777
```

```
bernoulli_predict = gaussian_nb.predict(X_test4)
print(classification_report(y_test4, bernoulli_predict))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2191
1	1.00	1.00	1.00	2166
2	1.00	1.00	1.00	2064
3	1.00	1.00	1.00	2058
4	1.00	1.00	1.00	1888
5	0.67	1.00	0.81	1555
6	1.00	1.00	1.00	1332
7	0.00	0.00	0.00	750
8	1.00	1.00	1.00	684
9	1.00	1.00	1.00	623
10	1.00	1.00	1.00	618
11	0.00	0.00	0.00	553
12	1.00	1.00	1.00	480
13	1.00	1.00	1.00	472
14	1.00	1.00	1.00	413
15	1.00	1.00	1.00	410
16	1.00	1.00	1.00	206
17	0.09	0.87	0.17	67
18	0.06	0.06	0.06	65
accuracy			0.93	18595
macro avg	0.78	0.84	0.79	18595
weighted avg	0.90	0.93	0.91	18595

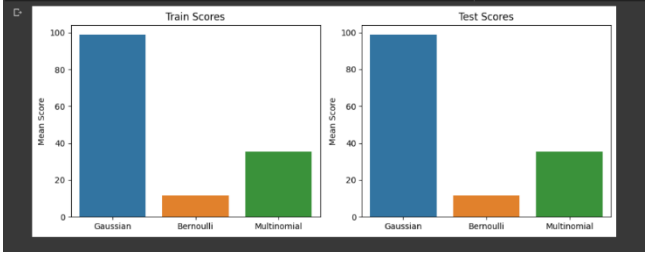
```
[ ] multinomial_nb = MultinomialNB()
    multinomial_nb.fit(X_train4, y_train4)

    MultinomialNB()
    MultinomialNB()

[ ] multinomial_cv = cross_validate(estimator = multinomial_nb,
                                    X = X_valid4, y = y_valid4,
                                    cv = 10, return_train_score = True)
    multinomial_mean_train_score = round((multinomial_cv['train_score'] * 100).mean(), 3)
    multinomial_mean_test_score = round((multinomial_cv['test_score'] * 100).mean(), 3)
    print(f'Mean train score: {multinomial_mean_train_score}')
    print(f'Mean test score: {multinomial_mean_test_score}')

    Mean train score: 35.328
    Mean test score: 35.284
```

```
multinomial_predict = multinomial_nb.predict(X_test4)
print(classification_report(y_test4, multinomial_predict))
```

	precision	recall	f1-score	support
0	0.35	0.61	0.45	2191
1	0.38	0.45	0.41	2166
2	0.61	0.62	0.62	2064
3	0.61	0.41	0.49	2058
4	0.00	0.00	0.00	1888
5	0.36	0.46	0.41	1555
6	0.32	0.53	0.40	1332
7	0.00	0.00	0.00	750
8	0.18	0.36	0.24	684
9	0.09	0.13	0.11	623
10	0.14	0.11	0.12	618
11	0.29	0.06	0.10	553
12	0.07	0.07	0.07	480
13	0.41	0.68	0.51	472
14	0.17	0.14	0.15	413
15	0.00	0.00	0.00	410
16	0.00	0.00	0.00	206
17	0.00	0.00	0.00	67
18	0.00	0.00	0.00	65
accuracy			0.36	18595
macro avg	0.21	0.24	0.21	18595
weighted avg	0.31	0.36	0.32	18595

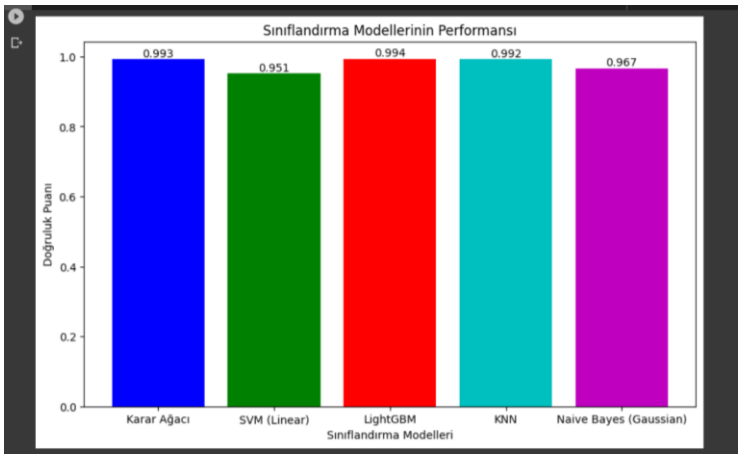
16. Naive Bayes modellerinin eğitim ve test skorlarını içeren bir görselleştirme yapıyoruz.

```
mean_train_scores = pd.DataFrame(data = [[gaussian_mean_train_score,
                                          bernoulli_mean_train_score,
                                          multinomial_mean_train_score]],
                                 columns = ['Gaussian', 'Bernoulli', 'Multinomial'])
mean_test_scores = pd.DataFrame(data = [[gaussian_mean_test_score,
                                         bernoulli_mean_test_score,
                                         multinomial_mean_test_score]],
                                columns = ['Gaussian', 'Bernoulli', 'Multinomial'])
fig, ax = plt.subplots(1, 2, figsize = (10, 4))
sns.barplot(data = mean_train_scores, ax = ax[0])
ax[0].set_title('Train Scores')
sns.barplot(data = mean_test_scores, ax=ax[1])
ax[1].set_title('Test Scores')
ax[0].set_ylabel('Mean Score')
ax[1].set_ylabel('Mean Score')
plt.tight_layout()
plt.show()
```



17. Son olarak kullandığımız modellerin performanslarını değerlendiren bir çubuk grafik çiziyoruz.

```
Sonuç

performanslar = []
performanslar.append(classifier.score(X_test, y_test))
performanslar.append(metrics.accuracy_score(y_test1, y_pred_linear))
performanslar.append(clf.score(X_test2, y_test2))
performanslar.append(test_scores_mean)
performanslar.append(gaussian_mean_test_score/100)
modeller = ['Karar Ağacı', 'SVM (Linear)', 'LightGBM', 'KNN', 'Naive Bayes (Gaussian)']
renkler = ['b', 'g', 'r', 'c', 'm']
plt.figure(figsize=(10, 6))
plt.bar(modeller, performanslar, color = renkler)
for i in range(len(modeller)):
    plt.text(i, performanslar[i], round(performanslar[i], 3), ha = 'center', va = 'bottom')
plt.xlabel('Sınıflandırma Modelleri')
plt.ylabel('Doğruluk Puanı')
plt.title('Sınıflandırma Modellerinin Performansı')
plt.show()
```



Genel olarak, bu sonuçlar bize modellerin başarılı olduğunu gösteriyor. Ancak, sadece doğruluk skorlarına dayanarak bir modelin performansını tam olarak değerlendirmek doğru olmayabilir. Diğer değerlendirme metriklerini (örneğin; hassasiyet, özgünlük, F1 skoru) ve cross-validation gibi yöntemleri de kullanarak modelleri daha kapsamlı bir şekilde değerlendirilmesi önerilebilir.

Bu kod parçacığı, bir yıl boyunca yapılan satış verilerini işleyerek farklı sınıflandırma modelleriyle ürün tahmini yapmaya odaklanmaktadır.