

# 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.

```
Gerekli kütüphaneleri yükleme

!pip install lightgbm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import lightgbm as lgb
import glob
import warnings

from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.utils import shuffle
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
```

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

```
Dosyaları içe aktarma

[ ] df1 = pd.read_csv("Sales_January_2019.csv")
df2 = pd.read_csv("Sales_February_2019.csv")
df3 = pd.read_csv("Sales_March_2019.csv")
df4 = pd.read_csv("Sales_April_2019.csv")
df5 = pd.read_csv("Sales_May_2019.csv")
df6 = pd.read_csv("Sales_June_2019.csv")
df7 = pd.read_csv("Sales_July_2019.csv")
df8 = pd.read_csv("Sales_August_2019.csv")
df9 = pd.read_csv("Sales_September_2019.csv")
df10 = pd.read_csv("Sales_October_2019.csv")
df11 = pd.read_csv("Sales_November_2019.csv")
df12 = pd.read_csv("Sales_December_2019.csv")

Dosya içeriklerini görüntüleme

df1
df2

Dosyaları birleştirme ve yeni DataFrame oluşturma

year = [df1, df2, df3, df4, df5, df6, df7, df8, df9, df10, df11, df12]
df = pd.concat(year)
```

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

```
Veri ön işleme

# Verinin genel bilgilerini gösterme
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          186305 non-null object
 4   Order Date          186305 non-null object
 5   Purchase Address    186305 non-null object
dtypes: object(6)
memory usage: 10.0+ MB

[ ] # Eksik değerleri kontrol etme
df.isnull().sum()

Order ID            545
Product             545
Quantity Ordered    545
Price Each          545
Order Date          545
Purchase Address    545
dtype: int64

[ ] # Eksik değerleri içeren satırları çıkarma
df.dropna(axis = 0, inplace = True)
```

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: 185958 entries, 0 to 25116
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  --
 0   Order ID           185958 non-null object
 1   Product            185958 non-null object
 2   Quantity Ordered   185958 non-null object
 3   Price Each         185958 non-null float64
 4   Order Date         185958 non-null object
 5   Purchase Address   185958 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.dropna(axis = 0, inplace = True)

df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 185958 entries, 0 to 25116
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  --
 0   Order ID           185958 non-null object
 1   Product            185958 non-null object
 2   Quantity Ordered   185958 non-null object
 3   Price Each         185958 non-null float64
 4   Order Date         185958 non-null object
 5   Purchase Address   185958 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 02115
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 94106
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 %I:%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 02115    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 94106    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  319658  Lightning Charging Cable        1      14.95    14 Madison St, San Francisco, CA 94106    12
25113  319657    AA Batteries (4-pack)        2       3.84    545 Wilcox St, Los Angeles, CA 90001    12
25114  319658    Vaseebadd Phone            1      400.00    273 Wilcox St, Seattle, WA 98101    12
25115  319658    Wired Headphones            1       11.99    778 River St, Dallas, TX 75001    12
25116  319678    Bose SoundSport Headphones        1      99.99    747 Choshad St, Los Angeles, CA 90001    12
105958 rows x 6 columns

df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 185958 entries, 0 to 25116
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  --
 0   Order ID           185958 non-null object
 1   Product            185958 non-null object
 2   Quantity Ordered   185958 non-null int64
 3   Price Each         185958 non-null float64
 4   Purchase Address   185958 non-null object
 5   Month              185958 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    21983
Lightning Charging Cable 21058
AAA Batteries (4-pack)  20641
AA Batteries (4-pack)   20577
Wired Headphones        18882
Apple AirPods Headphones 15549
Bose SoundSport Headphones 13325
27in FHD Monitor        7587
iPhone                  6842
27in 4K Gaming Monitor  6230
34in Ultrawide Monitor  6181
Google Phone            5525
Flatscreen TV           4800
Macbook Pro Laptop      4724
ThinkPad Laptop          4128
28in Monitor            4101
Vaseebadd 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 == "28in Monitor"] = 15
df.Product[df.Product == "Vaseebadd 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 x 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	8	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 x 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 sütunları bağımsız değişken olarak atıyoruz.

- Bağımsız değişkenleri ve hedef değişkeni belirleme

```
[ ] X = df.drop("Product", axis = 1)  
    y = df["Product"]
```

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

```
Karar Ağacı (Decision Tree) sınıflandırma modelini eğitime ve test verileri üzerinde tahmin yapma
```

```
# Veri kümesini eğitim ve test verilerine bölmek
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

```
[ ] classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

```
>DecisionTreeClassifier
DecisionTreeClassifier()
```

```
[ ] y_pred = classifier.predict(X_test)
```

```
[ ] # Sınıflandırma sonuçlarını değerlendirme
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4480
1	1.00	1.00	1.00	4269
2	1.00	1.00	1.00	4176
3	1.00	1.00	1.00	4167
4	1.00	1.00	1.00	2728
5	1.00	1.00	1.00	3201
6	1.00	1.00	1.00	2678
7	1.00	1.00	1.00	1320
8	1.00	1.00	1.00	1380
9	1.00	1.00	1.00	1230
10	1.00	1.00	1.00	1211
11	0.80	1.00	0.89	1617
12	1.00	1.00	1.00	942
13	1.00	1.00	1.00	917
14	1.00	1.00	1.00	825
15	1.00	1.00	1.00	836
16	1.00	1.00	1.00	360
17	0.00	0.00	0.00	127
18	0.00	0.00	0.00	126

	accuracy	macro avg	weighted avg
accuracy	0.99	0.99	0.99
macro avg	0.99	0.99	0.99
weighted avg	0.99	0.99	0.99

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

```
Destek Vektör Makinesi (SVM) sınıflandırma modelini eğitime ve test verileri üzerinde tahmin yapma

[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

[ ] svc = SVC()
    svc.fit(X_train, y_train)
    y_pred = svc.predict(X_test)
    print('Accuracy Score:')
    print(metrics.accuracy_score(y_test, y_pred))

Accuracy Score:
0.7000000000000000

[ ] # Farklı çekirdek tipleri (kernel) ile SVM'yi değerlendirme

[ ] svc = SVC(kernel = 'linear')
    svc.fit(X_train, y_train)
    y_pred_linear = svc.predict(X_test)
    print('Accuracy Score:')
    linear_accuracy = metrics.accuracy_score(y_test, y_pred_linear)
    print(linear_accuracy)

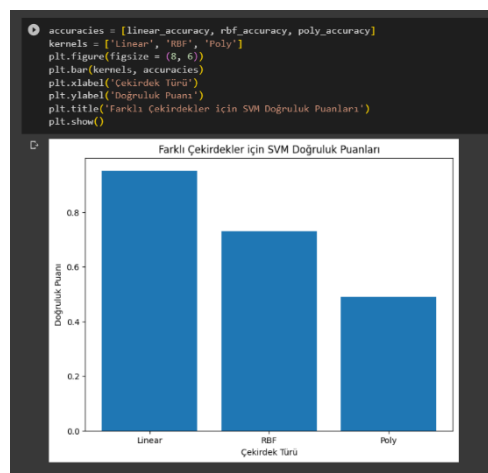
Accuracy Score:
0.9518018018018018

[ ] svc = SVC(kernel = 'rbf')
    svc.fit(X_train, y_train)
    y_pred_rbf = svc.predict(X_test)
    print('Accuracy Score:')
    rbf_accuracy = metrics.accuracy_score(y_test, y_pred_rbf)
    print(rbf_accuracy)

Accuracy Score:
0.7300000000000000

[ ] svc = SVC(kernel = 'poly')
    svc.fit(X_train, y_train)
    y_pred_poly = svc.predict(X_test)
    print('Accuracy Score:')
    poly_accuracy = metrics.accuracy_score(y_test, y_pred_poly)
    print(poly_accuracy)

Accuracy Score:
0.49115353589674643
```



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

## LightGBM sınıflandırma modelini eğitime ve test verileri üzerinde tahmin yapma

```
[ ] X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.20)

① clf = lgb.LGBMClassifier()
   clf.fit(X_train2, y_train2)

[ ] y_pred2 = clf.predict(X_test2)

[ ] accuracy=accuracy_score(y_pred2, y_test2)
   print('LightGBM Model accuracy score: {0:0.4f}'.format(accuracy_score(y_test2, y_pred2)))

   LightGBM Model accuracy score: 0.9929

[ ] y_pred_train = clf.predict(X_train2)
   print('Training set accuracy score: {0:0.4f}'.format(accuracy_score(y_train2, y_pred_train)))
   print('test set score: {0:0.4f}'.format(clf.score(X_test2, y_test2)))

   Training set accuracy score: 0.9930
   test set score: 0.9929
```

```
[ ] print(classification_report(y_test2, y_pred2))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4351
1	1.00	1.00	1.00	4411
2	1.00	1.00	1.00	4118
3	1.00	1.00	1.00	4095
4	1.00	1.00	1.00	3737
5	1.00	1.00	1.00	3848
6	1.00	1.00	1.00	2658
7	1.00	1.00	1.00	1545
8	1.00	1.00	1.00	1345
9	1.00	1.00	1.00	1234
10	1.00	1.00	1.00	1247
11	0.90	1.00	0.95	1091
12	1.00	1.00	1.00	1805
13	1.00	1.00	1.00	957
14	1.00	1.00	1.00	833
15	1.00	1.00	1.00	821
16	1.00	1.00	1.00	412
17	0.00	0.00	0.00	123
18	0.00	0.00	0.00	142
accuracy			0.99	37190
macro avg	0.88	0.89	0.89	37190
weighted avg	0.99	0.99	0.99	37190

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

K-En Yakın Komşu (KNN) sınıflandırma modelini eğitme ve test verileri üzerinde tahmin yapma

```
[ ] X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.2, random_state=1)

[ ] test_scores = []
    train_scores = []

    for i in range(1,15):
        knn = KNeighborsClassifier(i)
        knn.fit(X_train3, y_train3)

        train_scores.append(knn.score(X_train3, y_train3))
        test_scores.append(knn.score(X_test3, y_test3))

[ ] y_pred3 = knn.predict(X_test3)

[ ] test_scores_mean = np.mean(test_scores)
    train_scores_mean = np.mean(train_scores)
    print("Test Score Mean: {:.2f}".format(test_scores_mean))
    print("Train Score Mean: {:.2f}".format(train_scores_mean))

Test Score Mean: 0.99
Train Score Mean: 0.99
```

```
print(classification_report(y_test3, y_pred3))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	4314
1	1.00	1.00	1.00	4244
2	1.00	1.00	1.00	4127
3	1.00	1.00	1.00	4069
4	1.00	1.00	1.00	3830
5	1.00	1.00	1.00	3890
6	1.00	1.00	1.00	2697
7	1.00	1.00	1.00	1541
8	1.00	1.00	1.00	1397
9	1.00	1.00	1.00	1246
10	1.00	1.00	1.00	1241
11	0.81	1.00	0.90	1145
12	1.00	1.00	1.00	971
13	1.00	1.00	1.00	925
14	1.00	1.00	1.00	837
15	1.00	1.00	1.00	821
16	1.00	1.00	1.00	430
17	0.00	0.00	0.00	143
18	0.00	0.00	0.00	122
accuracy			0.99	37190
macro avg	0.88	0.89	0.89	37190
weighted avg	0.99	0.99	0.99	37190

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

Gaussian, Bernoulli ve Multinomial Naive Bayes modellerini eğitme ve test verileri üzerinde tahmin yapma

```
+ Kod + Metin

[ ] X_train4, X_test4, y_train4, y_test4 = train_test_split(X, y, stratify=y, train_size=0.80)
    X_test4, X_valid4, y_test4, y_valid4 = train_test_split(X_test4, y_test4, stratify=y_test4, train_size=0.5)
    X_train4.shape, X_test4.shape, X_valid4.shape, y_train4.shape, y_test4.shape, y_valid4.shape

((148760, 4), (18595, 4), (148760,), (18595,))

[ ] gaussian_nb = GaussianNB()
    gaussian_nb.fit(X_train4, y_train4)

+ GaussianNB
GaussianNB()

[ ] gaussian_cv = cross_validate(estimator=gaussian_nb,
                                X=X_valid4, y=y_valid4,
                                scoring='accuracy', cv=5,
                                return_train_score=True)
    gaussian_mean_train_score = round((gaussian_cv['train_score'] * 100).mean(), 3)
    gaussian_mean_test_score = round((gaussian_cv['test_score'] * 100).mean(), 3)
    print(f'Mean train score: {gaussian_mean_train_score}')
    print(f'Mean test score: {gaussian_mean_test_score}')

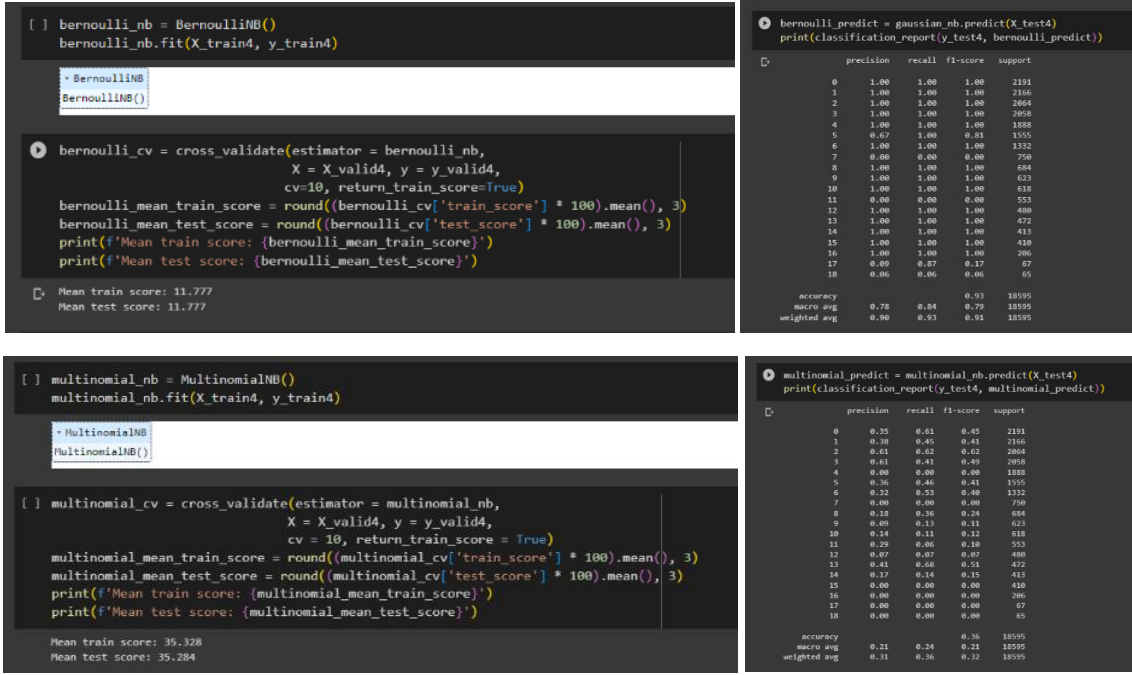
Mean train score: 99.029
Mean test score: 98.989
```

```
1 gaussian_predict = gaussian_nb.predict(X_test4)
2 print(classification_report(y_test4, gaussian_predict))

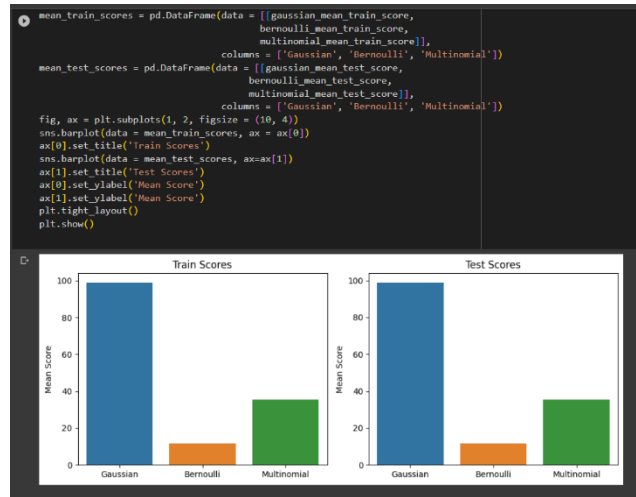
precision    recall  f1-score   support

0       1.00      1.00      1.00     2191
1       1.00      1.00      1.00     2166
2       1.00      1.00      1.00     2864
3       1.00      1.00      1.00     2858
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      400
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      286
17      0.09      0.87      0.17      67
18      0.06      0.06      0.06      65

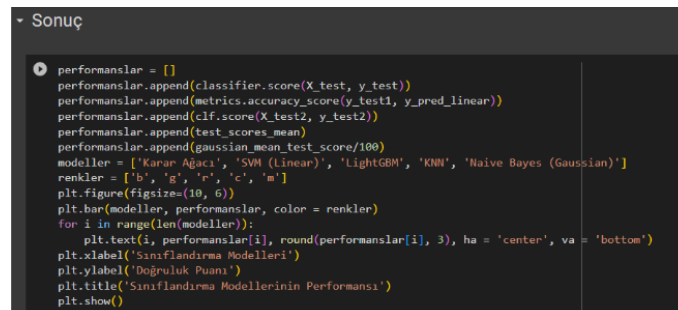
accuracy          0.93    18595
macro avg         0.78    0.84    0.79    18595
weighted avg      0.90    0.93    0.91    18595
```

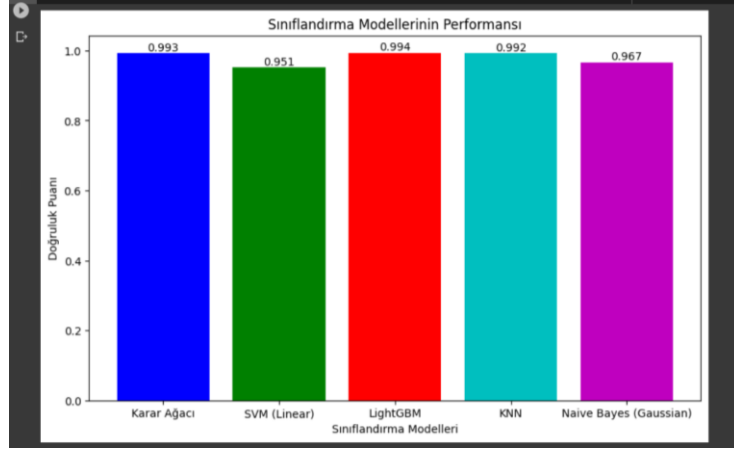


16. Naive Bayes modellerinin eğitim ve test skorlarını içeren bir görselleştirme yapıyoruz.



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





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 parçasığı, bir yıl boyunca yapılan satış verilerini işleyerek farklı sınıflandırma modelleriyle ürün tahmini yapmaya odaklanmaktadır.*