درس داده کاوی

گزارش تمرین سری سوم

استادان:

جناب آقای دکتر فراهانی

جناب آقای دکتر خردپیشه

استاد حل تمرین:

جناب آقای شریفی

گرداورنده:

مجید محمدزمانی

شماره دانشجویی:

۹۹۴۲۲۱۷۲

فهرست مطالب

مقدمه

مجموعه داده مربوط به داده های کلاس بندی قیمت موبایل می باشد.

برای تحلیل داده از زبان پایتون و کتابخانه های مرتبط آن در Google Colab استفاده نموده ایم.

پاسخ سوال اول

توابع کرنل SVM

الگوریتم های SVM از مجموعه ای از توابع ریاضی که به عنوان کرنل تعریف می شوند، استفاده می کنند. وظیفه کرنل این است که داده ها را به عنوان ورودی گرفته و آن ها را به شکل مورد نیاز تبدیل کند. الگوریتم های مختلف SVM ، از انواع مختلف توابع کرنل استفاده می کنند. این توابع می توانند انواع متفاوتی داشته باشند. به عنوان مثال خطی ، غیرخطی ، چند جمله ای ، تابع پایه شعاعی (RBF) و سیگموئید.

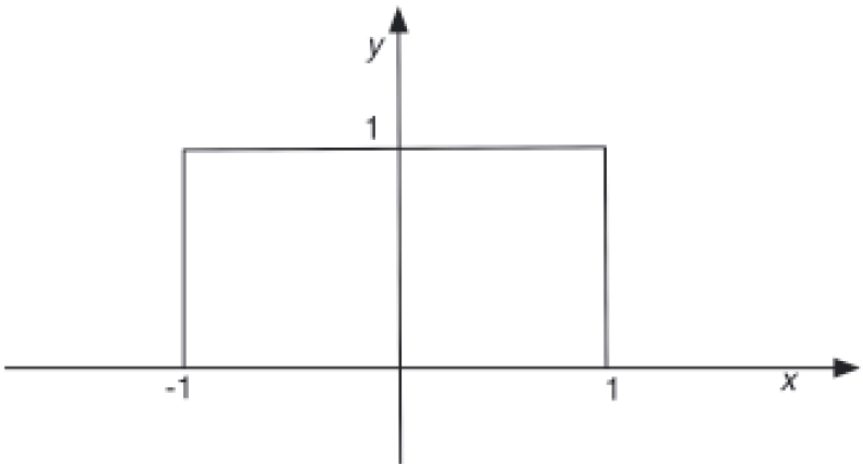
توابع کرنل ، برای داده های ترتیبی ، نمودار ها ، متن ها ، تصاویر و همچنین بردار ها معرفی می شوند. پرکاربردترین نوع تابع کرنل، RBF است. زیرا دارای پاسخ محلی و متناهی در کل بازه محور x است. توابع کرنل، ضرب داخلی بین دو نقطه در یک فضای ویژگی مناسب را برمی گردانند. بنابراین ، با هزینه محاسباتی کم، حتی در فضاهای با ابعاد بالا، مفهومی از شباهت را تعریف می کنند.

قواعد کرنل

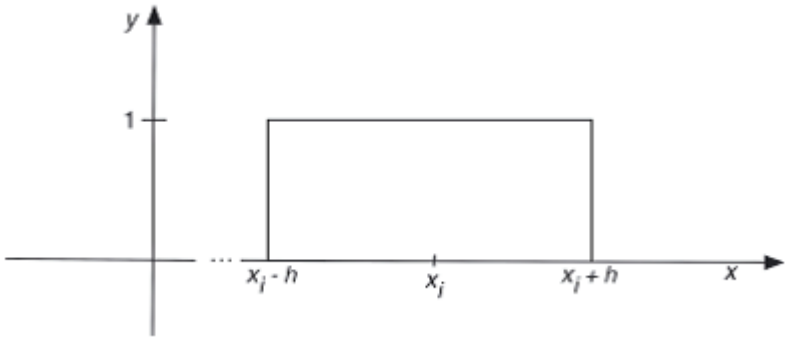
تعریف کرنل یا یک تابع پنجره به شرح زیر است:

$$K\left(\overline{x}\right) = \begin{matrix} 1 & \text{if} \left\|\overline{x}\right\| \le 1 \\ 0 & \text{otherwise} \end{matrix}$$

مقدار این تابع، در داخل یک شکل بسته به دامنه ۱ و مرکز مبدا مختصات برابر ۱ و در غیر این صورت ۰ است. همانطور که در شکل زیر نشان داده شده است:



برای xi ثابت، در داخل شکل بسته با دامنه h و مرکز xi ، تابع برابر است با $K(z - xi / h) = 1$ و در غیر این صورت ۰ می باشد. همانطور که در شکل زیر نشان داده شده است:



بنابراین ، با انتخاب آرگومان $K(0)$ ، پنجره را حرکت داده اید تا با دامنه h در مرکز xi قرار گیرد.

نمونه هایی از کرنل های SVM

برخی از کرنل های رایج مورد استفاده در SVM ها و کاربرد های آن ها به شرح زیر می باشد:

۱ -کرنل چند جمله ای

این کرنل در پردازش تصویر پرکاربرد است. معادله آن به صورت زیر است:

$$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$$

که در آن d درجه چند جمله ای است.

۲ -کرنل گاوسی

این یک کرنل برای اهداف عمومی است و هنگامی که هیچ دانش پیشینی در مورد داده ها وجود ندارد استفاده می شود. معادله آن به صورت زیر است:

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

۳ -تابع پایه شعاعی گاوسی (RBF)

این کرنلی برای اهداف عمومی کلربرد دارد. و هنگامی که هیچ دانش پیشینی در مورد داده ها وجود نداشته باشد، مورد استفاده قرار می گیرد. معادله آن به صورت زیر است:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

و برای

$$\gamma > 0$$

گاهی اوقات با استفاده از پارامتر زیر استفاده می شود:

$$\gamma = 1/2\sigma^2$$

۴ -کرنل RBF لاپلاس

این هم یک کرنل برای اهداف عمومی است. و هنگامی که هیچ دانش پیشینی در مورد داده ها وجود ندارد استفاده می شود. معادله آن به صورت زیر است:

$$k(x, y) = \exp\left(-\frac{\|x - y\|}{\sigma}\right)$$

۵ -کرنل تانژانت هیپربولیک(tanh)

می توانیم از آن در شبکه های عصبی استفاده کنیم. معادله مربوط به آن عبارت است از:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa\mathbf{x_i} \cdot \mathbf{x_j} + c)$$

در برخی موارد (نه همیشه) $k > 0$ **و** $c < 0$.

۶ -کرنل سیگموئید

می توان این کرنل را در شبکه های عصبی مورد استفاده قرار داد. معادله مربوط به آن عبارت است از :

$$k(x, y) = \tanh(\alpha x^T y + c)$$

۷ -کرنل تابع بِسِل (Bessel) از نوع اول

ما می توانیم از آن برای حذف مقطع عرضی در توابع ریاضی استفاده کنیم. معادله آن عبارت است از:

$$k(x,y) = \frac{J_{v+1}(\sigma\|x-y\|)}{\|x-y\|^{-n(v+1)}}$$

که J تابع بسل از نوع اول است.

۸ -کرنل پایه شعاعی ANOVA

ما می توانیم از آن در مسائل رگرسیون استفاده کنیم. معادله مربوط به آن عبارت است از:

$$k(x,y) = \sum_{k=1}^{n} \exp(-\sigma(x^k - y^k)^2)^d$$

۹ -کرنل spline خطی بصورت یک بعدی

این کرنل، هنگام کار با بردارهای بزرگ داده پراکنده ، کاربرد زیادی دارد. این کرنل اغلب در دسته بندی متن مورد استفاده قرار می گیرد. کرنل spline همچنین در مسائل رگرسیون عملکرد خوبی دارد.

معادله آن عبارت است از:

$$k(x,y) = 1 + xy + xy\ min(x,y) - \frac{x+y}{2}\ min(x,y)^2 + \frac{1}{3}\min(x,y)^3$$

پاسخ سوال دوم

در ابتدا کتابخانه های مورد نیاز را وارد پروژه می نماییم.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import urllib
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from matplotlib.colors import ListedColormap
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import FunctionTransformer
```

ابتدا API های کاگل را تعریف می کنیم.

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.8.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle) (5.0.2)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.41.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2020.12.5)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python-slugify->kaggle)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (3.0.4)
```

سپس فایل دسترسی را آپلود می نماییم.

```
from google.colab import files
files.upload()
```

```
Choose Files kaggle.json
• kaggle.json(application/json) - 68 bytes, last modified: 6/5/2021 - 100% done
Saving kaggle.json to kaggle.json
```

حال خواندن فایل دسترسی

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

وسپس دانلود فایل دیتای مورد نظر

```
!kaggle datasets download -d iabhishekofficial/mobile-price-classification
```

```
Downloading mobile-price-classification.zip to /content
  0% 0.00/70.6k [00:00<?, ?B/s]
100% 70.6k/70.6k [00:00<00:00, 26.3MB/s]
```

نمایش فایل ها

```
!ls
```

```
kaggle.json  mobile-price-classification.zip  sample_data
```

Extract کردن فایل ها

```
import zipfile
zip_ref = zipfile.ZipFile('mobile-price-classification.zip', 'r')
zip_ref.extractall('files')
zip_ref.close()
```

و پس از خواندن داده ها.

```
test=pd.read_csv('/content/files/test.csv')
train=pd.read_csv('/content/files/train.csv')
```

حالا اطلاعات اولیه را از داده ها بدست می آوریم.

```
train.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_wi
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

و اطلاعات کلی درباره تعداد رکورد و ستون ها و تعداد رکودهای null و ...

```
print('Rows     :',train.shape[0])
print('Columns  :',train.shape[1])
print('\nFeatures :\n     :',train.columns.tolist())
print('\nMissing values    :',train.isnull().values.sum())
print('\nUnique values :  \n',train.nunique())
```

```
Rows     : 2000
Columns  : 21

Features :
     : ['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g', 'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc',

Missing values    : 0

Unique values :
 battery_power    1094
blue                 2
clock_speed         26
dual_sim             2
fc                  20
four_g               2
int_memory          63
m_dep               10
mobile_wt          121
n_cores              8
pc                  21
px_height         1137
px_width          1109
ram               1562
sc_h                15
sc_w                19
talk_time           19
three_g              2
touch_screen         2
wifi                 2
price_range          4
dtype: int64
```

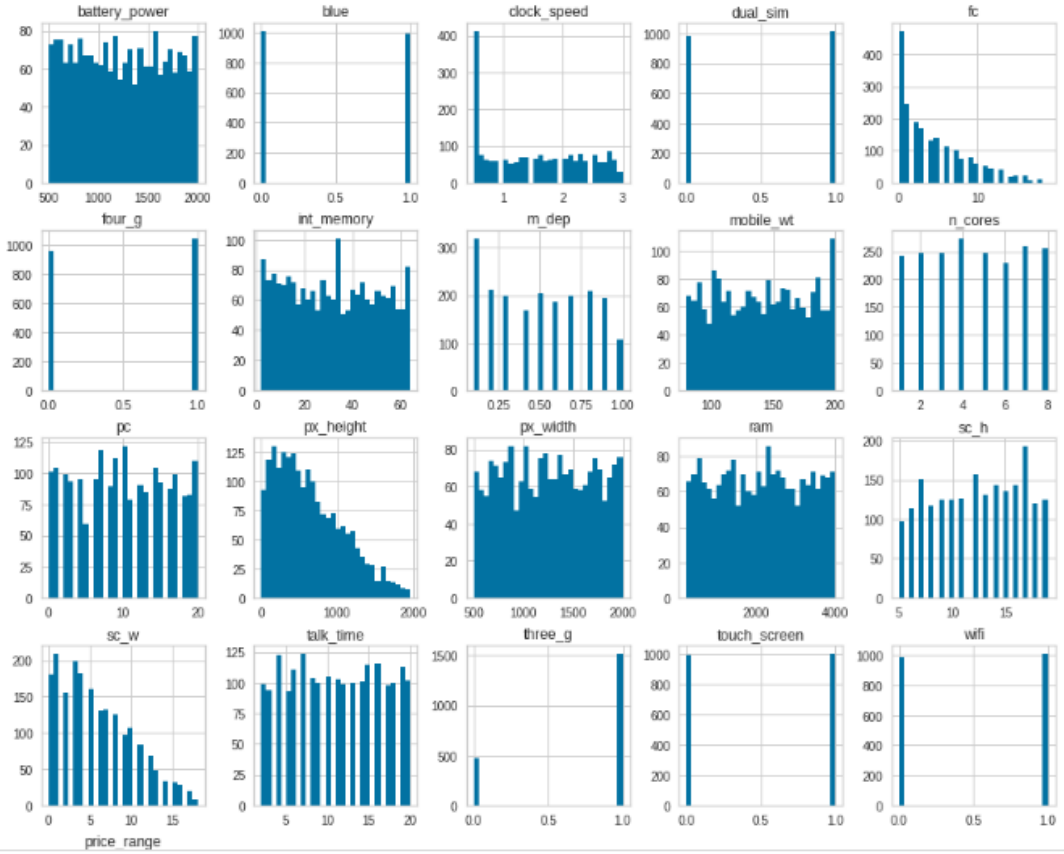
```
train.describe().T.round(decimals=1)
```

	count	mean	std	min	25%	50%	75%	max
battery_power	2000.0	1238.5	439.4	501.0	851.8	1226.0	1615.2	1998.0
blue	2000.0	0.5	0.5	0.0	0.0	0.0	1.0	1.0
clock_speed	2000.0	1.5	0.8	0.5	0.7	1.5	2.2	3.0
dual_sim	2000.0	0.5	0.5	0.0	0.0	1.0	1.0	1.0
fc	2000.0	4.3	4.3	0.0	1.0	3.0	7.0	19.0
four_g	2000.0	0.5	0.5	0.0	0.0	1.0	1.0	1.0
int_memory	2000.0	32.0	18.1	2.0	16.0	32.0	48.0	64.0
m_dep	2000.0	0.5	0.3	0.1	0.2	0.5	0.8	1.0
mobile_wt	2000.0	140.2	35.4	80.0	109.0	141.0	170.0	200.0
n_cores	2000.0	4.5	2.3	1.0	3.0	4.0	7.0	8.0
pc	2000.0	9.9	6.1	0.0	5.0	10.0	15.0	20.0
px_height	2000.0	645.1	443.8	0.0	282.8	564.0	947.2	1960.0
px_width	2000.0	1251.5	432.2	500.0	874.8	1247.0	1633.0	1998.0
ram	2000.0	2124.2	1084.7	256.0	1207.5	2146.5	3064.5	3998.0
sc_h	2000.0	12.3	4.2	5.0	9.0	12.0	16.0	19.0
sc_w	2000.0	5.8	4.4	0.0	2.0	5.0	9.0	18.0
talk_time	2000.0	11.0	5.5	2.0	6.0	11.0	16.0	20.0
three_g	2000.0	0.8	0.4	0.0	1.0	1.0	1.0	1.0
touch_screen	2000.0	0.5	0.5	0.0	0.0	1.0	1.0	1.0
wifi	2000.0	0.5	0.5	0.0	0.0	1.0	1.0	1.0
price_range	2000.0	1.5	1.1	0.0	0.8	1.5	2.2	3.0

اطلاعات اولیه داده ها را مشاهده می نماییم.

```
train.hist(bins=30, figsize=(15, 15))
```

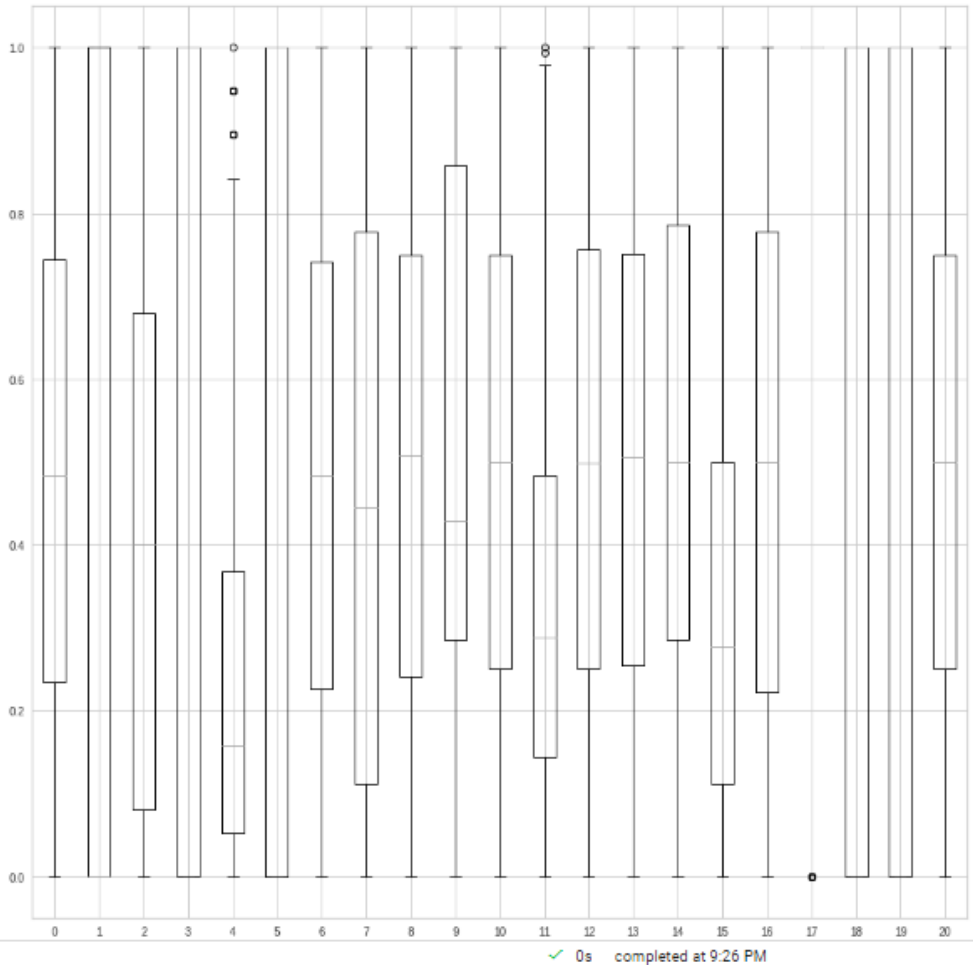
```
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f41c5ea4b10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f41c5e690d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f41c5e1e650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f41c5e50bd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f41c5d95190>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f41c5dc9710>]],
      dtype=object)
```



و نمودار اولیه داده ها.

```
x = train.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
data_scaled = pd.DataFrame(x_scaled)
data_scaled.boxplot(figsize=(15, 15))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f41c6985f50>
```



✓ 0s completed at 9:26 PM

نرمال سازی میانگین داده ها.

```
print(train.corr()['price_range'].sort_values())
```

```
touch_screen    -0.030411
mobile_wt       -0.030302
clock_speed     -0.006606
m_dep            0.000853
n_cores          0.004399
four_g           0.014772
dual_sim         0.017444
wifi             0.018785
blue             0.020573
talk_time        0.021859
fc               0.021998
sc_h             0.022986
three_g          0.023611
pc               0.033599
sc_w             0.038711
int_memory       0.044435
px_height        0.148858
px_width         0.165818
battery_power    0.200723
ram              0.917046
price_range      1.000000
Name: price_range, dtype: float64
```

ارتباط داده ها نسبت به رنج قیمت.

```
lowCorrs = list(train.corr()['price_range'].sort_values().keys()[:16]); print("Drop:", lowCorrs)

X = train.drop(['price_range', *lowCorrs], axis=1);print(X.shape)
y = train['price_range'];print(y.shape)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=22, shuffle=True)
```

```
Drop: ['touch_screen', 'mobile_wt', 'clock_speed', 'm_dep', 'n_cores', 'four_g', 'dual_sim', 'wifi', 'blue', 'talk_time',
(2000, 4)
(2000,)
```

آماده سازی داده ها برای SVM

```
model_svm = SVC()
model_svm.fit(X_train, y_train)
model_svm.score(X_test, y_test)
```

```
y_pred = model_svm.predict(X_test)
CM = confusion_matrix(y_test, y_pred)
print(CM)
CR = classification_report(y_test, y_pred)
print(CR)
```

```
[[141   3   0   0]
 [  6 137   2   0]
 [  0   5 148   5]
 [  0   0   4 149]]
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       144
           1       0.94      0.94      0.94       145
           2       0.96      0.94      0.95       158
           3       0.97      0.97      0.97       153

    accuracy                           0.96       600
   macro avg       0.96      0.96      0.96       600
weighted avg       0.96      0.96      0.96       600
```

و اجرای SVM بر روی داده و نتایج آن

پاسخ سوال سوم

اجرای SVM با کرنل linear بر روی داده و نتایج آن

```
model_svm = SVC(kernel='linear')
model_svm.fit(X_train, y_train)
model_svm.score(X_test, y_test)
```

```
0.965
```

```
y_pred = model_svm.predict(X_test)
CM = confusion_matrix(y_test, y_pred)
print(CM)
CR = classification_report(y_test, y_pred)
print(CR)
```

```
[[141   3   0   0]
 [  2 141   2   0]
 [  0   5 151   2]
 [  0   0   7 146]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       144
           1       0.95      0.97      0.96       145
           2       0.94      0.96      0.95       158
           3       0.99      0.95      0.97       153

    accuracy                           0.96       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.96      0.97       600
```

اجرای SVM با کرنل sigmoid بر روی داده و نتایج آن

```
model_svm = SVC(kernel='sigmoid')
model_svm.fit(X_train, y_train)
model_svm.score(X_test, y_test)
```

```
0.018333333333333333
```

```
y_pred = model_svm.predict(X_test)
CM = confusion_matrix(y_test, y_pred)
print(CM)
CR = classification_report(y_test, y_pred)
print(CR)
```

```
[[ 11   0   0 133]
 [ 53   0   0  92]
 [131   9   0  18]
 [153   0   0   0]]
              precision    recall  f1-score   support

           0       0.03      0.08      0.04       144
           1       0.00      0.00      0.00       145
           2       0.00      0.00      0.00       158
           3       0.00      0.00      0.00       153

    accuracy                           0.02       600
   macro avg       0.01      0.02      0.01       600
weighted avg       0.01      0.02      0.01       600
```

که نتیجه دقیقی نمی دهد.

اجرای SVM با کرنل poly بر روی داده و نتایج آن

```
model_svm = SVC(kernel='poly')
model_svm.fit(X_train, y_train)
model_svm.score(X_test, y_test)
```

```
0.9616666666666667
```

```
y_pred = model_svm.predict(X_test)
CM = confusion_matrix(y_test, y_pred)
print(CM)
CR = classification_report(y_test, y_pred)
print(CR)
```

```
[[141   3   0   0]
 [  1 140   4   0]
 [  0   4 150   4]
 [  0   0   7 146]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       144
           1       0.95      0.97      0.96       145
           2       0.93      0.95      0.94       158
           3       0.97      0.95      0.96       153

    accuracy                           0.96       600
   macro avg       0.96      0.96      0.96       600
weighted avg       0.96      0.96      0.96       600
```

اجرای SVM با کرنل rbf بر روی داده و نتایج آن

```
[239] model_svm = SVC(kernel='rbf')
      model_svm.fit(X_train, y_train)
      model_svm.score(X_test, y_test)
```

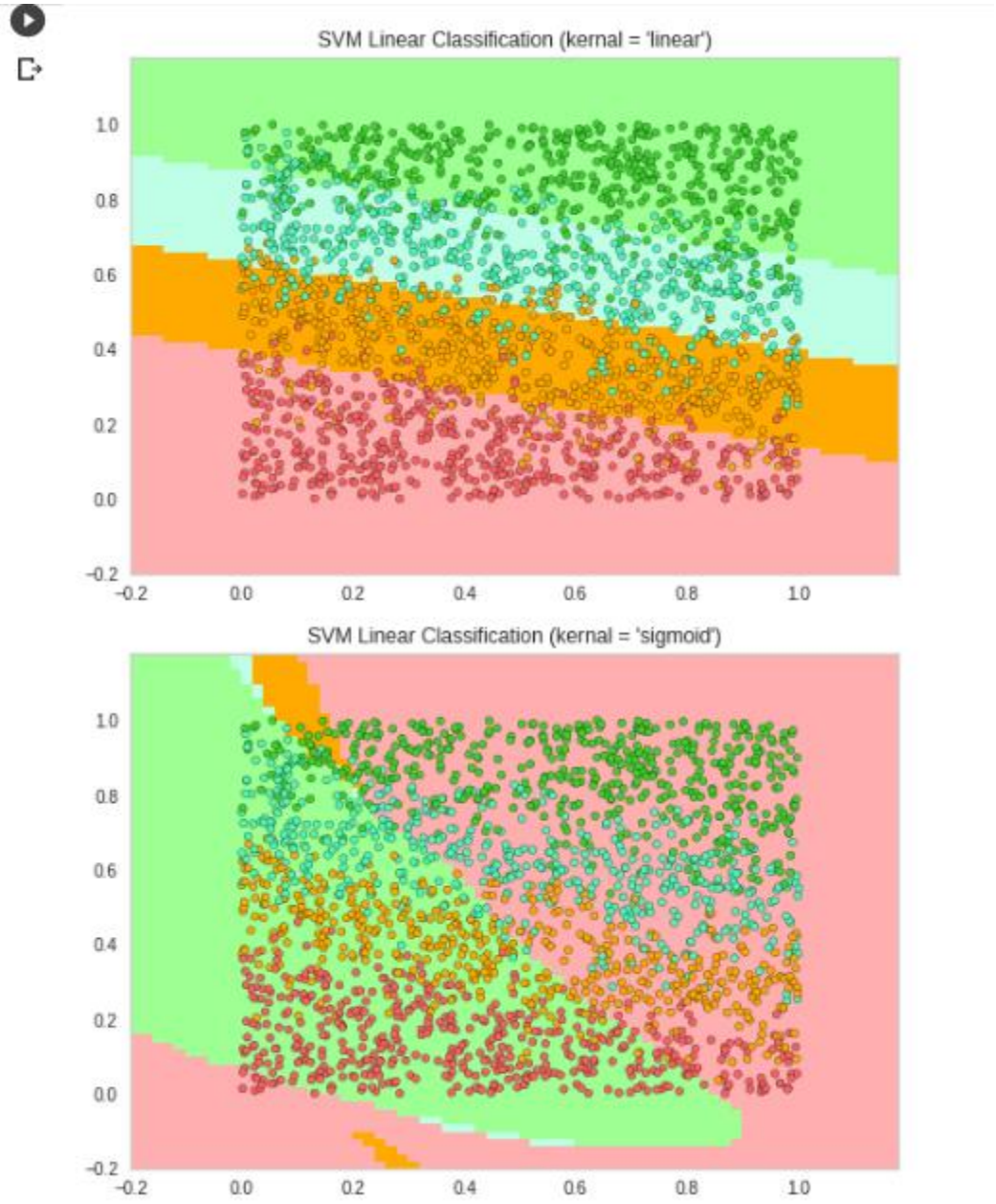
```
0.9583333333333334
```

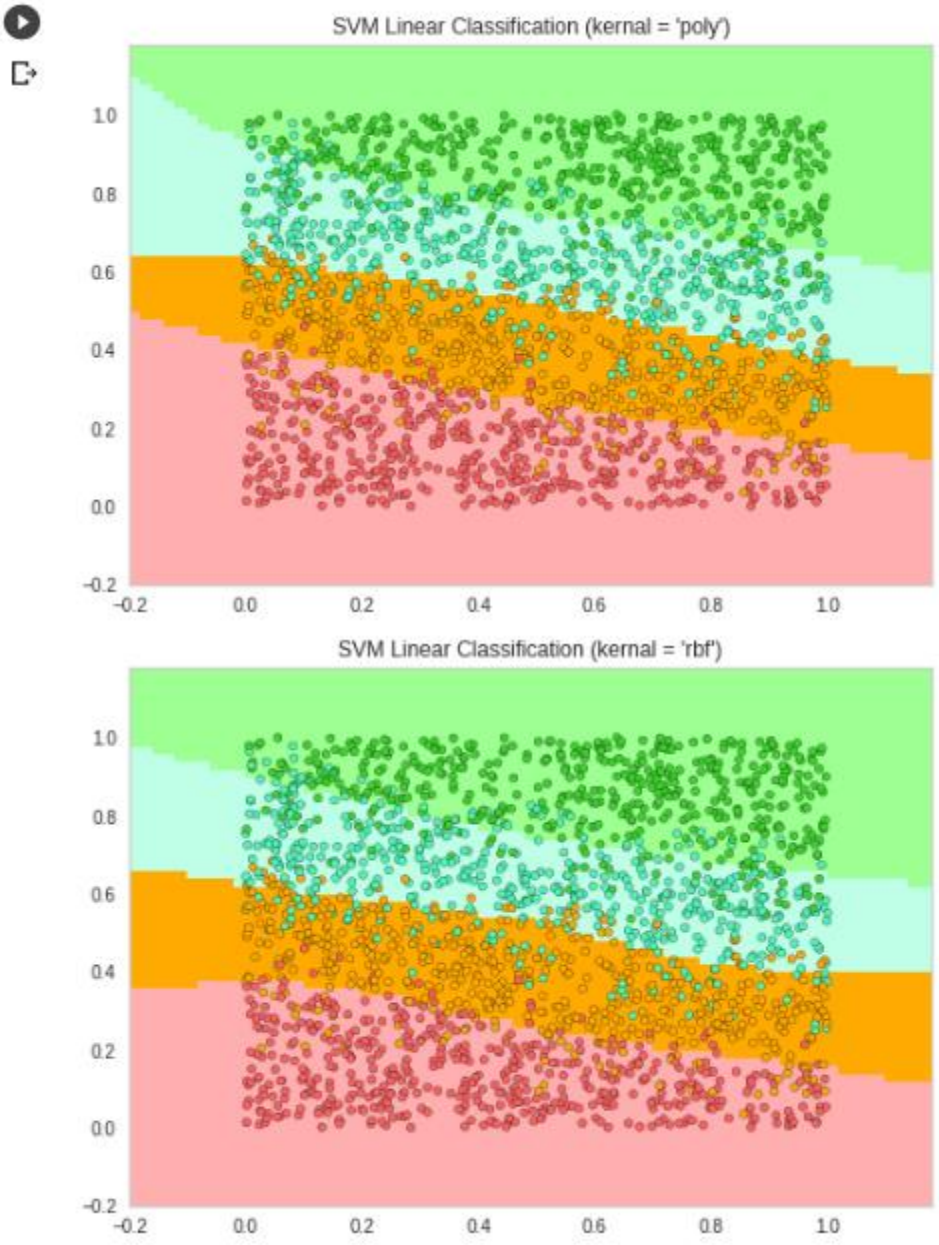
```
y_pred = model_svm.predict(X_test)
CM = confusion_matrix(y_test, y_pred)
print(CM)
CR = classification_report(y_test, y_pred)
print(CR)
```

```
[[141   3   0   0]
 [  6 137   2   0]
 [  0   5 148   5]
 [  0   0   4 149]]
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       144
           1       0.94      0.94      0.94       145
           2       0.96      0.94      0.95       158
           3       0.97      0.97      0.97       153

    accuracy                           0.96       600
   macro avg       0.96      0.96      0.96       600
weighted avg       0.96      0.96      0.96       600
```

و نمایش نمای تمامی کرنل ها و مقایسه آنها

```
cm_dark = ListedColormap(['#ff6060', '#8282ff','#ffaa00','#fff244','#4df9b9','#76e8fc','#3ad628'])
cm_bright = ListedColormap(['#ffafaf', '#c6c6ff','#ffaa00','#ffe2a8','#bfffe7','#c9f7ff','#9eff93'])

scaler = MinMaxScaler()

X = np.array(train.iloc[:,[0,13]])
y = np.array(train['price_range'])
X = scaler.fit_transform(X)
h = .02
C_param = 1
for kernel in ('linear','sigmoid', 'poly', 'rbf'):

    clf1 = SVC(kernel=kernel,C=C_param)
    clf1.fit(X, y)

    x_min, x_max = X[:, 0].min()-.20, X[:, 0].max()+.20
    y_min, y_max = X[:, 1].min()-.20, X[:, 1].max()+.20
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = clf1.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cm_bright)

    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cm_dark,
                edgecolor='k', s=20)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("SVM Linear Classification (kernal = '%s')"% (kernel))

plt.show()
```

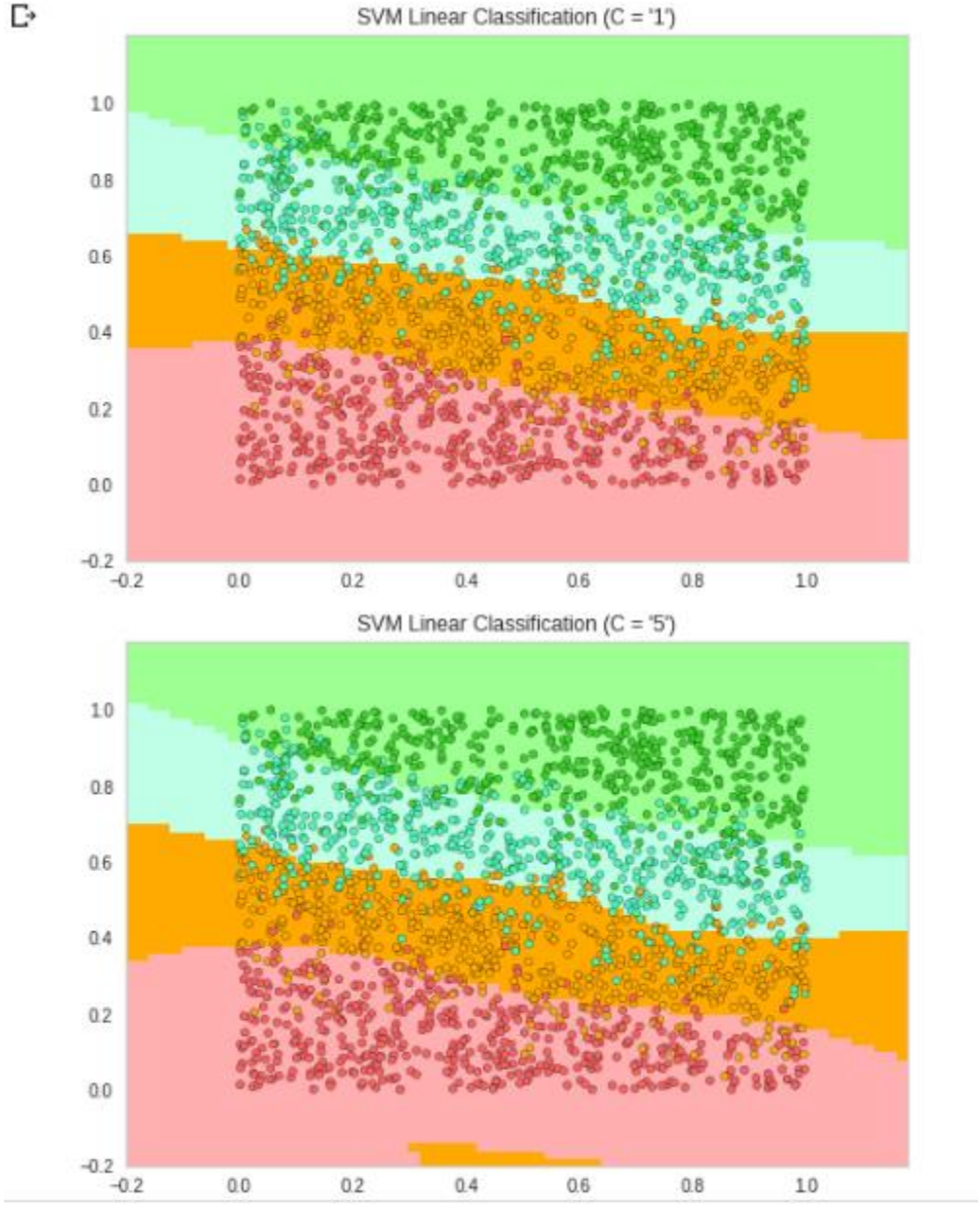

SVM Linear Classification (kernal = 'linear')
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
SVM Linear Classification (kernal = 'sigmoid')
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.2
0.0
0.2
0.4
0.6
0.8
1.0

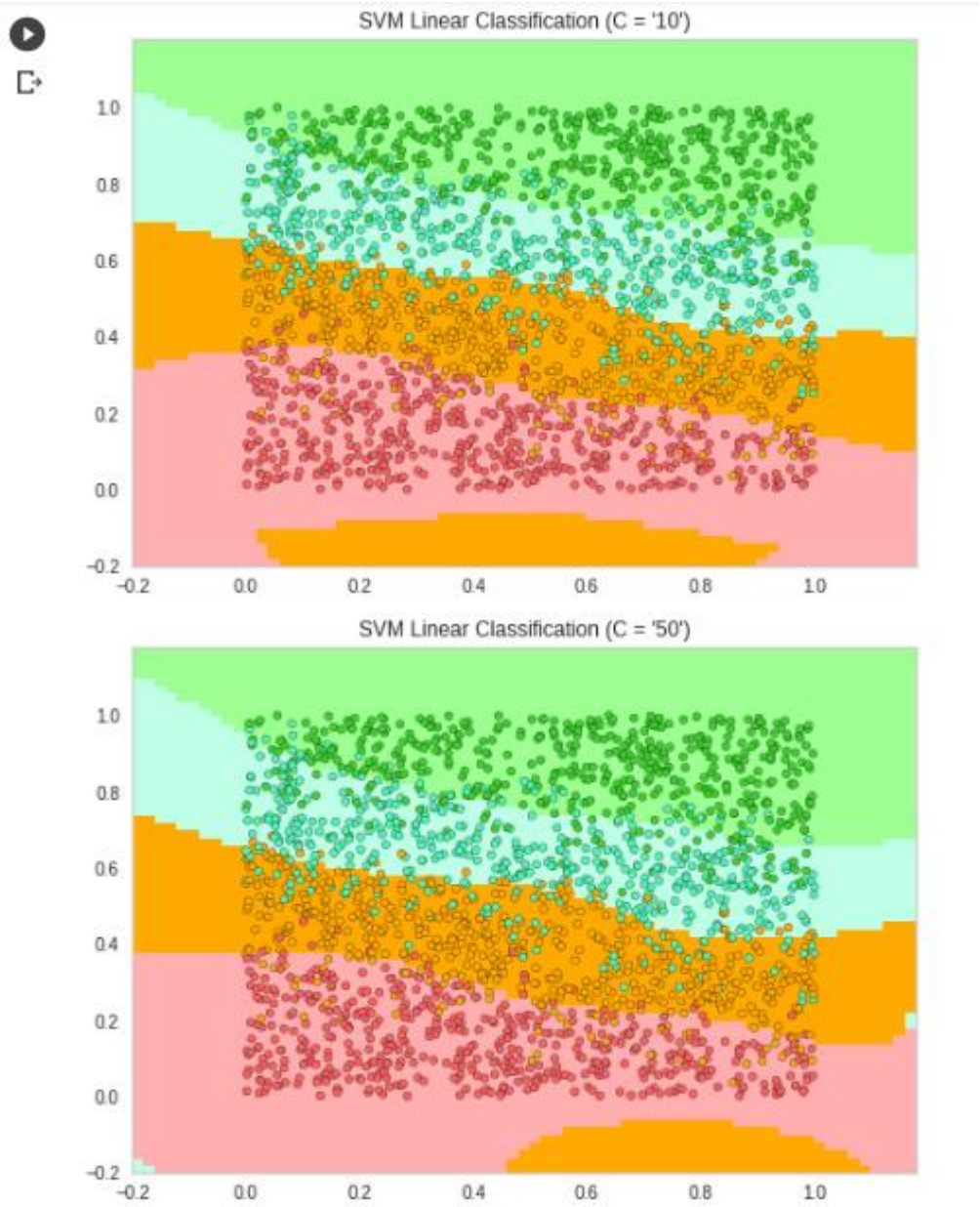


پاسخ سوال چهارم

و نمایش margin **ها با اندازه های متفاوت برای** soft margin **و** hard margin

```
C_param = 1
for C_param in (1,5,10,50):

    clf1 = SVC(C=C_param)
    clf1.fit(X, y)

    x_min, x_max = X[:, 0].min()-.20, X[:, 0].max()+.20
    y_min, y_max = X[:, 1].min()-.20, X[:, 1].max()+.20
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = clf1.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cm_bright)

    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cm_dark,
                edgecolor='k', s=20)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("SVM Linear Classification (C = '%s')"% (C_param))

plt.show()
```


SVM Linear Classification (C = '1')
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.2
0.0
0.2
0.4
0.6
0.8
1.0
SVM Linear Classification (C = '5')
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.2
0.0
0.2
0.4
0.6
0.8
1.0

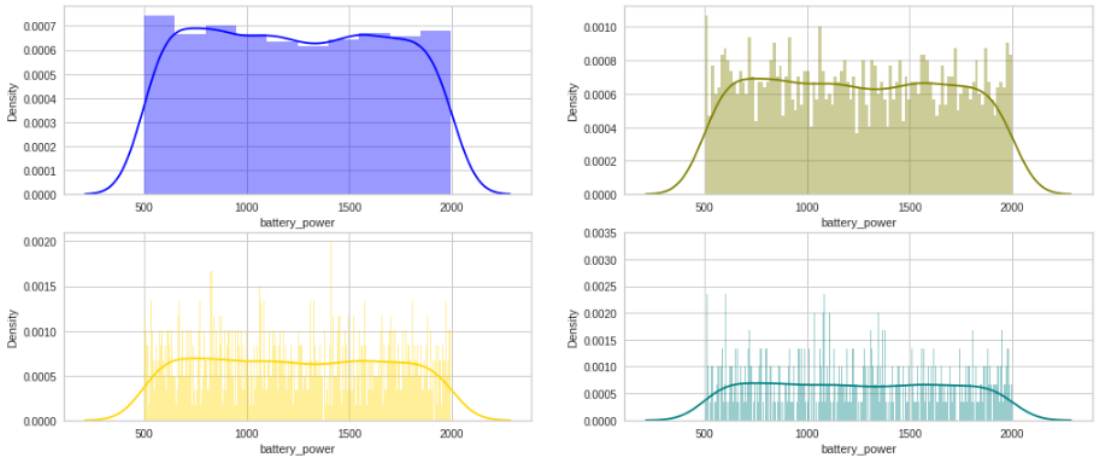


پاسخ سوال پنجم

بخش آ

تفاوت binning **های متفاوت بر روی فیچر** battery power

```
train_new = train["battery_power"]
f, axes = plt.subplots(2, 2, figsize=(17, 7), sharex=False)
sns.distplot( train["battery_power"], bins=10, color="blue", ax=axes[0, 0])
sns.distplot( train["battery_power"], bins=100, color="olive", ax=axes[0, 1])
sns.distplot( train["battery_power"], bins=500, color="gold", ax=axes[1, 0])
sns.distplot( train["battery_power"], bins=1000, color="teal", ax=axes[1, 1])
```



بخش ب

```
from sklearn.preprocessing import OneHotEncoder

X_OHE = train[['price_range','dual_sim','four_g','three_g','touch_screen','wifi']]
X_OHE = pd.get_dummies(X_OHE,columns=['price_range','dual_sim','four_g','three_g','touch_screen','wifi'])
X_OHE.head(16)
```

	price_range_0	price_range_1	price_range_2	price_range_3	dual_sim_0	dual_sim_1	four_g_0	four_g_1	three_g_0	three_g_1	touch_screen_
0	0	1	0	0	1	0	1	0	1	0	
1	0	0	1	0	0	1	0	1	0	1	
2	0	0	1	0	0	1	0	1	0	1	
3	0	0	1	0	1	0	1	0	0	1	
4	0	1	0	0	1	0	0	1	0	1	
5	0	1	0	0	0	1	1	0	0	1	
6	0	0	0	1	1	0	0	1	0	1	
7	1	0	0	0	0	1	1	0	0	1	
8	1	0	0	0	1	0	1	0	0	1	
9	1	0	0	0	0	1	0	1	0	1	
10	0	0	0	1	0	1	1	0	1	0	
11	0	0	0	1	1	0	0	1	0	1	
12	0	1	0	0	1	0	1	0	0	1	
13	0	0	1	0	1	0	1	0	0	1	
14	1	0	0	0	1	0	0	1	0	1	
15	1	0	0	0	1	0	1	0	0	1	

بخش ج

```
transformer = FunctionTransformer(np.log1p, validate=True)
LT = train
transformer.transform(LT)
print(LT)
```

```
      battery_power  blue  clock_speed  ...  touch_screen  wifi  price_range
0               842     0          2.2  ...             0     1            1
1              1021     1          0.5  ...             1     0            2
2               563     1          0.5  ...             1     0            2
3               615     1          2.5  ...             0     0            2
4              1821     1          1.2  ...             1     0            1
...             ...   ...          ...  ...           ...   ...          ...
1995            794     1          0.5  ...             1     0            0
1996           1965     1          2.6  ...             1     1            2
1997           1911     0          0.9  ...             1     0            3
1998           1512     0          0.9  ...             1     1            0
1999            510     1          2.0  ...             1     1            3

[2000 rows x 21 columns]
```

بخش د

```
train_area=train[['battery_power','px_height','px_width','price_range']]
train_area['area'] = train['px_height']*train['px_width']
train_area.head()
```

	battery_power	px_height	px_width	price_range	area
0	842	20	756	1	15120
1	1021	905	1988	2	1799140
2	563	1263	1716	2	2167308
3	615	1216	1786	2	2171776
4	1821	1208	1212	1	1464096

پاسخ سوال ششم

```
X_OHE = pd.get_dummies(train,columns=['price_range','dual_sim','four_g','three_g','touch_screen','wifi'])
print(X_OHE.shape)
y1 = X_OHE['price_range_0']
print(y1.shape)

X1_train, X1_test, y1_train, y1_test = train_test_split(X_OHE, y1, test_size=0.3, random_state=22, shuffle=True)

model_svm = SVC()
model_svm.fit(X1_train, y1_train)
model_svm.score(X1_test, y1_test)
```

```
(2000, 29)
(2000,)
0.9866666666666667
```

```
y1_pred = model_svm.predict(X1_test)
CM = confusion_matrix(y1_test, y1_pred)
print(CM)
CR = classification_report(y1_test, y1_pred)
print(CR)
```

```
[[450   6]
 [  2 142]]
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       456
           1       0.96      0.99      0.97       144

    accuracy                           0.99       600
   macro avg       0.98      0.99      0.98       600
weighted avg       0.99      0.99      0.99       600
```

و بالاتر رفتن فیت بودن SVM در One Hot Encoding