



دانشکده علوم ریاضی  
گروه علوم کامپیوتر

گزارش تمرین سری سوم

درس داده کاوی

جناب آقای دکتر فراهانی و جناب آقای دکتر خرد پیشه

دستیار محترم جناب آقای علی شریفی

زینب خسروی ۹۹۴۲۲۰۶۷

شبکه های پشتیبان در یادگیری ماشین با نظارت هستند. برای طبقه بندی داده ها استفاده می شوند. یکی از مهمترین روش های پیش بینی قوی هستند. مدلی را ایجاد می کنند که نمونه های جدید دسته بندی کند. با به وجود آوردن شکاف یا حاشیه بین دو دسته که هدف این هست که عرض شکاف بین دو دسته را به حداکثر برسانیم. تا با اطمینان بیشتری بگوییم داده در کدام کلاس هست. و سپس نمونه های جدید در همان فضای ترسیم شده پیش بینی کند.

بعضی وقت ها جدا می شوند ولی خطا هم دارند. در این صورت نرم رفتار کردیم یعنی چیزهایی وارد مارجین می شود که جریمه یا خطا در نظر می گیریم که خطاها نباید از حدی بیشتر شوند. باید خطا هم کم شود.

وقتی نمی شود داده ها را از هم جدا کرد توان بالا می بریم بعد فضای ویژگی بالا می رود. غیر خطی میشود ویژگی های جدید اضافه می شود. حالا داده هایی که داشتیم جدا نمی شدند با این روش توانایی بالا بردیم و از هم جدا می شوند.

کرنل ها مشابه این هست که فیچر اضافه می کردیم. کرنل ها ضابطه های متفاوتی دارند. برای اینکه با خط نمی توانستیم داده ها را از هم جدا کنیم از کرنل ها استفاده می کنیم که خط تبدیل میشود به رویه و داده ها را از هم جدا می کند. و داده ها را کلاس بندی می کند.

بردار های پشتیبان با کرنل های مختلف شکل های مختلف حاصل می شود. اگر بیش از دو کلاس داشته باشیم. از OVA, OVO استفاده می کنیم.

با استفاده از ترفند کرنل ی برای طبقه بندی غیر خطی انجان می دهند. که ورودی ها را در فضای ویژگی هایی با بعد بالا نگاشت می دهند.

وقتی داده ها بدون برچسب هستند یادگیری با نظارت امکان پذیر نیست. رویکرد بدون نظارت نیاز هست داده ها را خوشه بندی می کند. سپس داده های جدید برای این گروه ها ترسیم می کند. هدف این هست که داده جدید در کدام کلاس قرار خواهد گرفت . در SVM می خواهیم با ابر صفحه نقاط از هم جدا کنیم که به این روش طبقه بندی خطی می گویند. که ابر صفحه های زیادی وجود دارد که داده ها را طبقه بندی میکند. یکی از انتخاب های منطقی به عنوان ابر صفحه این هست که بیشترین حاشیه بین در کلاس نشان دهد. وظایف دیگرش تشخیص نقاط دور افتاده هست بزرگترین فاصله تا نزدیک ترین فاصله داده های آموزشی از هر کلاس بدست می آورد.

در حالی که مسئله اصلی ممکن است که در یک فضای متناهی محدود بیان شود. اغلب اتفاق می افتد که به طور خطی در آن فضا قابل تفکیک نیست. برای همین در فضایی با ابعاد بالاتر بر اساس عملکرد هسته یا کرنل ترسیم می شوند. تا تفکیک آن آسانتر شود. از مجموع هسته ها یا کرنل ها برای اندازه گیری نزدیکی نسبی نقاط به سایر مجموعه ها که تفکیک شده اند استفاده می کنند.

تابع کرنل خطا را به حداقل می رسانند و برای مسائل بهینه سازی مناسب هستند. پس کرنل مناسب باید انتخاب شود. که داده ها را به بدرستی طبقه بندی کند. و حاشیه حداکثر کند. و خطا به حداقل برساند

## توابع کرنل SVM

الگوریتم های SVM از مجموعه ای از توابع ریاضی که به عنوان کرنل تعریف می شوند استفاده می کند. وظیفه کرنل این هست که داده ها را به عنوان ورودی گرفته و آنها را به شکل مورد نیاز تبدیل کند. این توابع انواع مختلفی دارند مثال: خطی ، غیرخطی ، و چند جمله ای ، تابع پایه شعاعی RBF سیگموئید.

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.

### کرنل چند جمله ای

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

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$$

### کرنل گاوسی

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

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

### تابع پایه شعاعی گاوسی RBF

این کرنل برای اهداف عمومی کاربرد 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

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

## 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.

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

## کرنل تانژانت هیپربولیک tanh

می توانیم از آن در شبکه های عصبی استفاده کنیم. معادله به صورت زیر است.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i \cdot \mathbf{x}_j + c)$$

## کرنل سیگموئید

می توان این کرنل را در شبکه های عصبی مورد استفاده قرار داد. معادله به صورت زیر است.

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

## کرنل تابع بَسل

ما می توانیم از آن برای حذف مقطع عرضی در توابع ریاضی استفاده کنیم. معادله آن به صورت زیر است.

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

## کرنل پایه شعاعی ANOVA

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

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

### کرل خطی بصورت یک بعدی spline

این کرل، هنگام کار با بردارهای بزرگ داده پراکنده ، کاربرد زیادی دارد اغلب در دسته بندی متن مورد استفاده قرار می گیرد. در مسائل رگرسیون عملکرد خوبی دارد. معادله به صورت زیر است.

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

الگوریتم ماشین بردار پشتیبان در ابتدا برای ساخت کلاس بندی های باینری ایجاد شد سپس به عنوان الگوریتمی برای مسائل خوشه بندی و رگرسیون گسترش پیدا کرد.

یک الگوریتم جزئی برای روش های براساس کرل (هسته) است که ویژگی های برداری را به فضاهای با ابعاد بالاتر نگاشت می کند و این عمل با استفاده از عملکرد کرل و ساخت یک تابع بهینه خطی در این فضا یا فضایی چند بعدی که متناسب با داده های آموزشی است، صورت می گیرد.

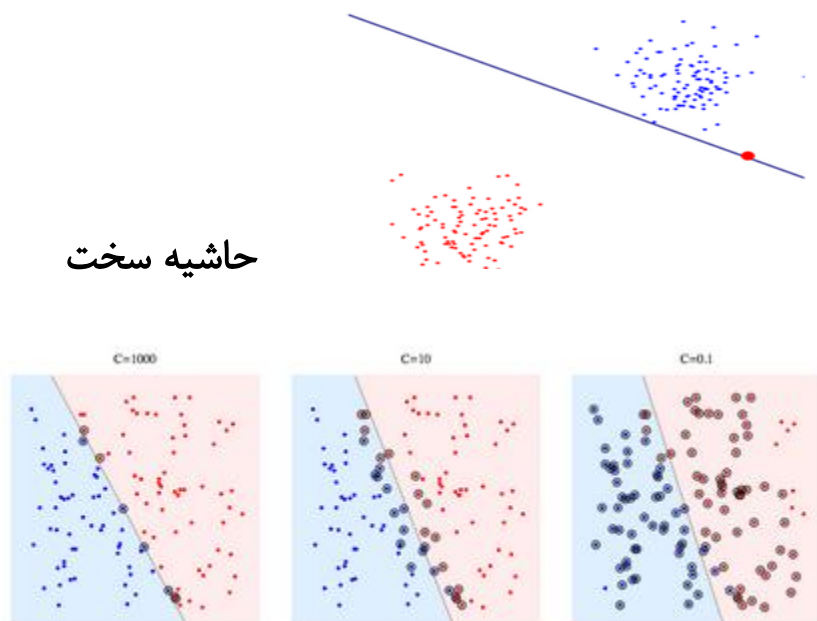
فضای مابین دو نقطه در فضاهای چند بعدی نیاز به تعریف کرل دارد. راه حل های ایجاد شده توسط این الگوریتم بهینه هستند. بردارهای ویژگی که به ابر صفحه نزدیک تر هستند، بردارهای پشتیبان نامیده می شوند

موقعیت بردارها بر عملکرد تصمیم گیری تاثیری نخواهد داشت

## Soft margin svm بهتر از hardmargin svm

با حاشیه نرم می توان یک مرز تصمیم انتخاب کرد که دارای خطای آموزش غیر صفر باشد حتی اگر داده ها به صورت خطی باشند که طبقه بندی شان قابل تفکیک هست.

حاشیه سخت



حاشیه سخت بیش از حد به نویز در داده حساس هست.

هدف به حداکثر رساندن حاشیه هست. با حاشیه سخت با یک محدودیت اضافی مواجه هستیم. در شکل بالا ضریب لاگرانژ هست. که وقتی ۱۰۰۰ می شود به حاشیه سخت نزدیک تر هست. که حاشیه در اینجا تقریباً صفر هست. با حاشیه نرم بهتری شود در طی مراحل آموزش متغیرهای جدید پیش بینی کرد. بنا براین حاشیه سخت نمی تواند تعمیم دهد یا نمی تواند داده های جدید پیش بینی کند. اما حاشیه نرم از قابلیت انعطاف پذیری بیشتری برخوردار هست.

برای کد نویسی پایتون

در ابتدا api token از کگل گرفتم خروجی یک فایل json به گوگل درایو منتقل کردم. بعد وارد کلب شدم درایو mount کردم. آدرس جایی که فایل قرار دادم تعریف کردم. و محل کار برایش مشخص کردم. در دیتاست کگل گزینه copy api commends زدم و بعد دستوری که در کگل کپی کردم علامت ! اولش گذاشتم و در کلب ران گرفتم. در آخر فایل هایی که زیپ بودند با کد دستوری زیر از حالت زیپ خارج کردم. داده ها را هم دانلود کردم و به صورت دستی در گوگل درایو آپلود کردم.

```
from google.colab import drive
drive.mount('/content/gdrive/')

Mounted at /content/gdrive/

[ ] import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content/gdrive/MyDrive/kaggle"

[ ] !kaggle datasets download -d iabhishekoofficial/mobile-price-classification

Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 5, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python2.7/dist-packages/kaggle/__init__.py", line 23, in <module>
    api.authenticate()
  File "/usr/local/lib/python2.7/dist-packages/kaggle/api/kaggle_api_extended.py",
    self.config_file, self.config_dir))
IOError: Could not find kaggle.json. Make sure it's located in /content/gdrive/MyDr

[ ] ! unzip/*.zip && rm *.zip

/bin/bash: unzip/*.zip: No such file or directory

[ ]
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('/content/gdrive/MyDrive/test.csv')
data = pd.read_csv('/content/gdrive/MyDrive/train.csv')
```

کتابخانه های مورد استفاده فراخوانی می کنیم.



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
%pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
[ ] import csv
import json
import re
import numpy as np
import pandas as pd
import altair as alt

from collections import Counter, OrderedDict
from IPython.display import HTML
```

ابتدا کتابخانه های مهم را برای اجرای بردار های پشتیبان در پروژه استفاده می کنیم.

در مرحله دوم برای پیاده سازی از مجموعه داده (`load_iris`) استفاده می کنیم.

فقط در این تحلیل از طول و عرض استفاده خواهیم کرد.

```

pylab.rcParams['figure.figsize'] = (10, 6)
iris_data = datasets.load_iris()
# We'll use the petal length and width only for this analysis
X = iris_data.data[:, [2, 3]]
y = iris_data.target
# Input the iris data into the pandas dataframe
iris_dataframe = pd.DataFrame(iris_data.data[:, [2, 3]],
                             columns=iris_data.feature_names[2:])
# View the first 5 rows of the data
print(iris_dataframe.head())
# Print the unique labels of the dataset
print('\n' + 'Unique Labels contained in this data are '
      + str(np.unique(y)))

```

```

petal length (cm)  petal width (cm)
0                1.4             0.2
1                1.4             0.2
2                1.3             0.2
3                1.5             0.2
4                1.4             0.2

```

Unique Labels contained in this data are [0 1 2]

داده ها را با استفاده از تابع `train_test_split()` به آموزش و تست تقسیم می کنیم.

```

45] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
print('The training set contains {} samples and the test set contains {} samples'.format(X_train.shape[0], X_test.shape[0]))

```

The training set contains 105 samples and the test set contains 45 samples

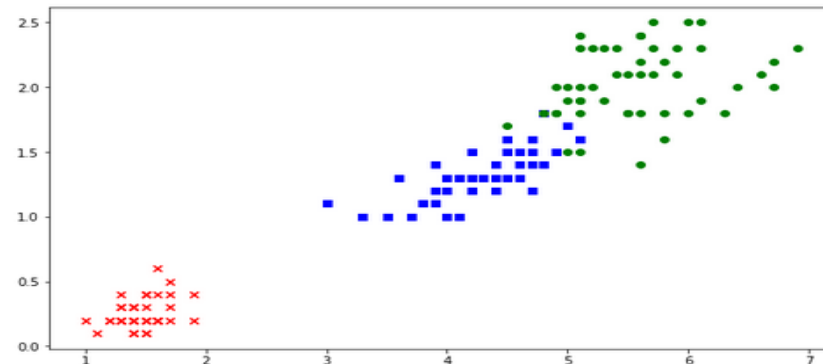
اکنون داده ها را تجسم می کنیم مشاهده می کنیم که یکی از کلاس ها به صورت خطی قابل تفکیک هست.

```

markers = ('x', 's', 'o')
colors = ('red', 'blue', 'green')
cmap = ListedColormap(colors[:len(np.unique(y_test))])
for idx, cl in enumerate(np.unique(y)):
    plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
               c=cmap(idx), marker=markers[idx], label=cl)

```

\*c\* argument looks like a single numeric RGB or RGBA sequence, which should be avoided  
\*c\* argument looks like a single numeric RGB or RGBA sequence, which should be avoided  
\*c\* argument looks like a single numeric RGB or RGBA sequence, which should be avoided



سپس داده ها را مقیاس گذاری کنیم مقیاس بندی این اطمینان میدهد که تمام مقادیر داده ها در یک محدوده مشترک قرار دارند.

```
[47] standard_scaler = StandardScaler()
      standard_scaler.fit(X_train)
      X_train_standard = standard_scaler.transform(X_train)
      X_test_standard = standard_scaler.transform(X_test)
      print('The first five rows after standardisation look like this:\n')
      print(pd.DataFrame(X_train_standard, columns=iris_dataframe.columns).head())
```

The first five rows after standardisation look like this:

	petal length (cm)	petal width (cm)
0	-0.182950	-0.293181
1	0.930661	0.737246
2	1.042022	1.638870
3	0.652258	0.350836
4	1.097702	0.737246

بعد از اینکه داده ها را پیش پردازش کردیم مرحله بعد پیاده سازی مدل بردار پشتیبان هست از کتابخانه sklearn استفاده می کنیم با کرنل rbf

```
[49] SVM = SVC(kernel='rbf', random_state=0, gamma=.10, C=1.0)
      SVM.fit(X_train_standard, y_train)

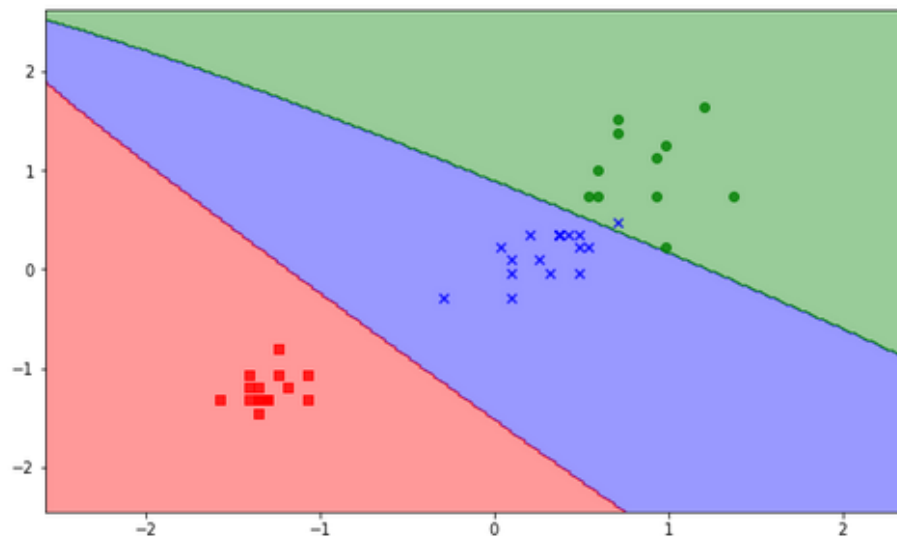
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
    max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001,
    verbose=False)
```

بعد از اینکه دقت خودش بدست آورد بهترین کار تجسم مدل هست با تابعی به نام Decision\_plot() انجام دهیم.

```
[50] import warnings
def versiontuple(version):
    return tuple(map(int, (version.split("."))))
def decision_plot(X, y, classifier, test_idx=None, resolution=0.02):
    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'green', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    # plot the decision surface
    x1min, x1max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2min, x2max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1min, x1max, resolution),
                           np.arange(x2min, x2max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                  alpha=0.8, c=cmap(idx),
                  marker=markers[idx], label=cl)
```

```
[51] decision_plot(X_test_standard, y_test, SVM)
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided a
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided a
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided a
```



داده فراخوانی می کنیم.

```
] data
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_w
0	842	0	2.2	0	1	0	7	0.6	18
1	1021	1	0.5	1	0	1	53	0.7	13
2	563	1	0.5	1	2	1	41	0.9	14
3	615	1	2.5	0	0	0	10	0.8	13
4	1821	1	1.2	0	13	1	44	0.6	14
...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	10
1996	1965	1	2.6	1	0	0	39	0.2	18
1997	1911	0	0.9	1	1	1	36	0.7	10
1998	1512	0	0.9	0	4	1	46	0.1	14
1999	510	1	2.0	1	5	1	45	0.9	16

2000 rows × 21 columns

< >

اطلاعات دیتا بدست می آوریم.

```
[ ] data.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
```

شمارش دیتا میانگین استاندارد مینیمم و ماکسیمم بدست می آوریم.

```
[ ] data.describe()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145700
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000

<

نوع دیتا را مشخص می کنیم.

```
[ ] data.dtypes
```

```
battery_power    int64
blue              int64
clock_speed      float64
dual_sim          int64
fc               int64
four_g           int64
int_memory        int64
m_dep            float64
mobile_wt         int64
n_cores           int64
pc               int64
px_height         int64
px_width          int64
ram              int64
sc_h              int64
sc_w              int64
talk_time         int64
three_g           int64
touch_screen      int64
wifi              int64
price_range       int64
dtype: object
```

داده های پوچ پیدا می کنیم.

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

```
battery_power    0
blue              0
clock_speed       0
dual_sim          0
fc                0
four_g            0
int_memory        0
m_dep             0
mobile_wt         0
n_cores           0
pc                0
px_height          0
px_width          0
ram               0
sc_h              0
sc_w              0
talk_time         0
three_g           0
touch_screen      0
wifi              0
price_range       0
dtype: int64
```

بعد اندازه دیتا بدست می آوریم

```
[ ] data.shape
```

```
(2000, 21)
```

نرمال سازی میکنیم تا داده های پرت حذف کند.بعد اندازه داده که می گیریم کم می شود.

```
[ ] for cols in data.columns:
    if data[cols].dtype == 'int64' or data[cols].dtype == 'float64':
        data[cols] = ((data[cols] - data[cols].mean())/(data[cols].std()))
```

```
[ ] for cols in data.columns:
    if data[cols].dtype == 'int64' or data[cols].dtype == 'float64':
        upper_range = data[cols].mean() + 3 * data[cols].std()
        lower_range = data[cols].mean() - 3 * data[cols].std()

        index = data[(data[cols]> upper_range)|(data[cols]< lower_range)].index
        data = data.drop(index)
```

```
[ ] data.shape
```

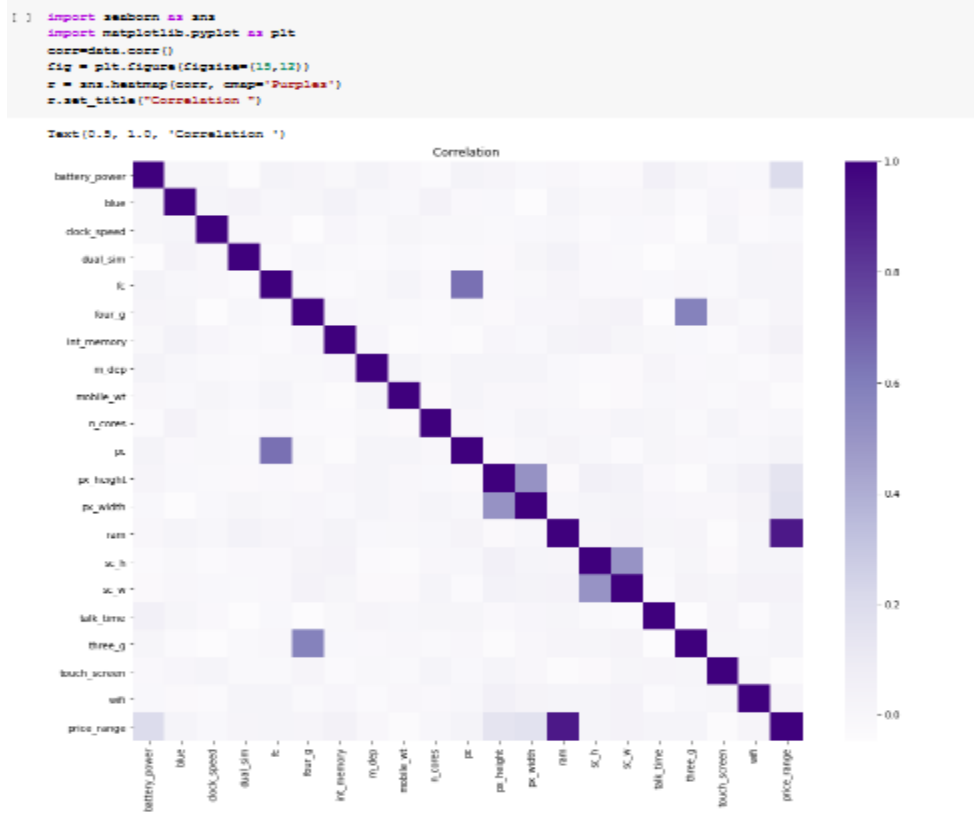
```
(1988, 21)
```

## عنوان ویژگی ها

```
[ ] data.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep
0	-0.902372	-0.989802	0.830572	-1.018929	-0.762304	-1.043705	-1.380298	0.340654
1	-0.495015	1.009798	-1.252751	0.980932	-0.992642	0.957646	1.154735	0.687376
2	-1.537302	1.009798	-1.252751	0.980932	-0.531966	0.957646	0.493422	1.380820
3	-1.418964	1.009798	1.198217	-1.018929	-0.992642	-1.043705	-1.214970	1.034098
4	1.325574	1.009798	-0.394912	-1.018929	2.001753	0.957646	0.658751	0.340654

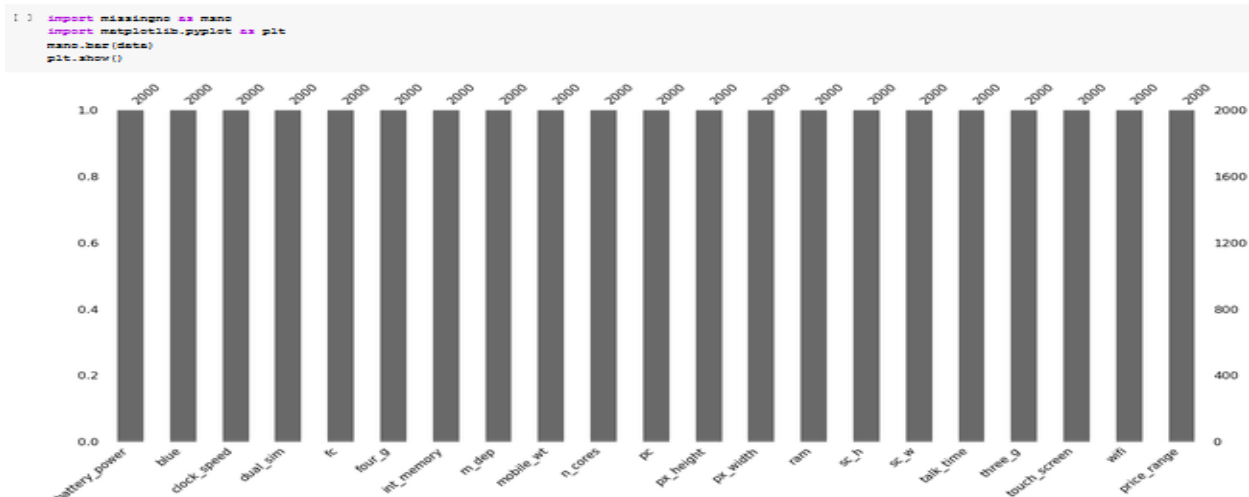
نمودار زیر همبستگی بین ویژگی ها را نشان می دهد.



با این نمودار پائین چک می کنیم که در دیتاست داده ها یا مقدارشون miss



شده یا نه یعنی مقدارش نداشته باشیم که در این جا همه داده ها هستند.



دیتاست کلاس بندی قیمت موبایل در کگل بود کد svm با استفاده از پکیج sklearn

```
[ ] import numpy as np
from sklearn.model_selection import train_test_split
labels = data['price_range']
features=data.drop('price_range',axis=1)
x_train,x_test,y_train,y_test = train_test_split(features,labels,test_size=0.2,random_sta

[ ] from sklearn import svm
clf = svm.SVC(kernel='linear', C=1,gamma=1)
```

مهندسی ویژگی

باید ویژگی های مناسبی از داده یا باز نمایی مناسب تری از داده داشته باشیم تا مدل عملکرد بهتری برای حل مسئله داشته باشد.و مدل بهتر و دقیق تر عمل کند.

در این قسمت باید خلاقیت زیاد باشد تا ویژگی ها را از داده ها بسازیم. برای هر کدام از داده ها تکنیک هایی برای مهندسی ویژگی هست.

مشکلی که هست این هست که بعد مسئله بالا می برد پیچیده گی مسئله هم بالا می برد .

راه کاربردی برای تبدیل ویژگی کتگوریکال به داده مقداری استفاده از:

## One hot encoding

یک ویژگی داریم k مقدار می تواند داشته باشد . از برداری به طول k استفاده می کنیم برای مدل هایی که خطی هستند استفاده می شود. فضا یا حافظه زیادی اشغال میکند. و اگر داده ای را نداشته باشیم مقدارش نداریم دیگه نمی توانیم از این روش استفاده کنیم. روش به این صورت هست که برای ویژگی مون برداری در نظر می گیریم به اندازه ی انواعی که داریم که مربوط به هر کدام بود یک می کنیم و بقیه صفر می شود. ویژگی های عددی بهبود ببخشیم. یعنی ویژگی های بیشتر به نفع مدل بدست آوریم. خود داده های عددی ممکنه به همان شکل که هستند به درد ما نخورند. مثلاً رند

```
chart=alt.Chart(data).mark_circle(size=20).encode(  
  x='ram',  
  y='battery_power',  
  color='price_range:N',  
  tooltip=["price_range", "ram", 'battery_power']  
)  
.interactive().properties(  
  width=400, height=300  
)
```

میکنیم. که متغیر عددی تبدیل می شود به کتگوریکال

کار دیگه ای که انجام می دهیم binning:

با بخش بخش کردن متغیر پیوسته تبدیل می کنیم به متغیر گسسته

داده ها را با فرایند کد کردن تغییر بدهیم تا در مدل خوب کار کند. و داده های پرت از بین ببریم. با این روش داده های پرت مدیریت می کنیم.

کریلیشن مقدار ویژگی price\_range نسبت به بقیه ویژگی ها

```
[ ] corr.sort_values(by=["price_range"],ascending=False).iloc[0].sort_values(ascending=
price_range      1.000000
ram               0.917046
battery_power     0.200723
px_width         0.165818
px_height        0.148858
int_memory       0.044435
sc_w             0.038711
pc               0.033599
three_g          0.023611
sc_h             0.022986
fc               0.021998
talk_time        0.021859
blue             0.020573
wifi             0.018785
dual_sim         0.017444
four_g           0.014772
n_cores          0.004399
m_dep            0.000853
clock_speed      -0.006606
mobile_wt        -0.030302
touch_screen     -0.030411
Name: price_range, dtype: float64
```

ویژگی PRICE RANGE AND RAM / Binning می کنیم

کلاس کم هزینه و کلاس هزینه متوسط و کلاس هزینه بالا و کلاس هزینه بسیار بالا

```
chart=alt.Chart(data).mark_bar().encode(
    alt.X('ram', bin=True),
    y='count(*) :Q',
    color='price_range:N',
).facet(column='price_range:N')
```

```
chart=alt.Chart(data).mark_bar().encode(
    alt.X('ram', bin=True),
    y='count(*) :Q',
    color='battery power:N',
).facet(column='battery power:N')
```

svm با کرنل RBF

```
[ ] from sklearn.svm import SVC
    model_svc = 'support vector classifier'
    svc = SVC(C=2, kernel='rbf')
```

به طور کلی برای مطابقت داده ها با نرمال بودن از log transform استفاده می کنند.

توزیع نرمال به شکل زنگوله ای هست. توزیع متقارن هست.

اغلب داده های بدست آمده در مطالعات واقعی کج و معوج هستند. که تحلیل آماری

استاندارد از این داده ها نتایج نامعتبری هستند.

داده های تبدیل شده log transform اگر در داده های اصلی از توزیع نرمال پیروی

کنند. داده های تبدیل شده هم از توزیع نرمال پیروی می کنند.

از هر روشی که استفاده می شود داده های انحرافی کنار می گذارند. برای وقتی کاربرد

دارد که به توزیع داده ها وابسته نیستند. برای مقابله با داده های انحرافی هست. اگر در

همه موارد استفاده می شود باید بسیار محتاطانه اعمال شود.

در این دیتاست نمی شود log transform استفاده کرد چون که داده های اصلی از

از توزیع نرمال پیروی نمی کنند.

برای اضافه کردن فیچر مساحت ویژگی طول را در ویژگی عرض ضرب می کنیم.

برای اضافه کردن فیچر حجم ویژگی طول در ویژگی ارتفاع ضرب می کنیم.

حاشیه نرم در ماشین بردار پشتیبان با استفاده sklwarn طبقه بندی ساده خطی از انجام می دهد. با بردار پشتیبان سعی می کند خطا به حداقل برساند. و حاشیه سختی ایجاد کند.

```
[ ] import numpy
    from scipy import optimize

    class SVM2C(object):
        def __init__(self,xdata,ydata,c=200.,learning_rate=0.01,
                      n_iter=5000,method='GD') :

            self.values=numpy.unique(ydata)
            self.xdata=xdata
            self.ydata=numpy.where(ydata==self.values[-1],1,-1)
            self.c=c
            self.lr=learning_rate
            self.n_iter=n_iter
            self.method=method

            self.m=len(xdata)
            self.theta=numpy.random.random(xdata.shape[1])-0.5

        def costFunc(self,theta,x,y):
            zs=numpy.dot(x,theta)
            j=numpy.maximum(0.,1.-y*zs).mean()*self.c+0.5*numpy.sum(theta**2)
            return j
```

```

def jac(self, theta, x, y):
    '''Derivative of cost function'''
    zs=numpy.dot(x, theta)
    ee=numpy.where(y*zs>=1., 0., -y)[: ,None]
    # multiply rows by ee
    dj=(ee*x).mean(axis=0)*self.c+theta
    return dj

def train(self):

    #-----Optimize using scipy.optimize-----
    if self.method=='optimize':
        opt=optimize.minimize(self.costFunc, self.theta, args=(self.xdata, self.ydata),
                               jac=self.jac, method='BFGS')
        self.theta=opt.x

    #-----Optimize using Gradient descent-----
    elif self.method=='GD':
        costs=[]
        lr=self.lr

        for ii in range(self.n_iter):
            dj=self.jac(self.theta, self.xdata, self.ydata)
            self.theta=self.theta-lr*dj
            cii=self.costFunc(self.theta, self.xdata, self.ydata)
            costs.append(cii)

        self.costs=numpy.array(costs)

    return self

```

```

def predict(self,xdata):

    yhats=[]
    for ii in range(len(xdata)):
        xii=xdata[ii]
        yhatii=xii.dot(self.theta)
        yhatii=1 if yhatii>=0 else 0
        yhats.append(yhatii)
    yhats=numpy.array(yhats)

    return yhats

#-----Main-----
if __name__=='__main__':

    xdata = numpy.array([[ -1, -1], [-2, -1], [1, 1], [2, 1]])
    ydata = numpy.array([1, 1, 2, 2])

    mysvm=SVM2C(xdata,ydata,method='GD')
    mysvm.train()

    from sklearn import svm
    clf=svm.SVC(C=50,kernel='linear')
    clf.fit(xdata,ydata)

    print ('mysvm.theta')
    print clf.coef_

#-----Plot-----

#-----Plot-----
import matplotlib.pyplot as plt
figure=plt.figure(figsize=(12,10),dpi=100)
ax=figure.add_subplot(111)

cmap=plt.cm.jet
nclasses=numpy.unique(ydata).tolist()
colors=[cmap(float(ii)/len(nclasses)) for ii in nclasses]

#-----Plot training data-----
for ii in range(len(ydata)):
    xii=xdata[ii][0]
    yii=xdata[ii][1]
    colorii=colors[nclasses.index(ydata[ii])]
    ax.plot(xii,yii,color=colorii,marker='o')

```

```

import numpy
from scipy import optimize

class SVM2C(object):
    def __init__(self, xdata, ydata, c=9000, learning_rate=0.001,
                  n_iter=600, method='GD'):

        self.values=numpy.unique(ydata)
        # Add 1 dimension for bias
        self.xdata=numpy.hstack([xdata, numpy.ones([xdata.shape[0],1])])
        self.ydata=numpy.where(ydata==self.values[-1],1,-1)
        self.c=c
        self.lr=learning_rate
        self.n_iter=n_iter
        self.method=method

        self.m=len(xdata)
        self.theta=numpy.random.random(self.xdata.shape[1])-0.5

    def costFunc(self, theta, x, y):
        zs=numpy.dot(x, theta)
        j=numpy.maximum(0., 1.-y*zs).mean()*self.c+0.5*numpy.sum(theta[:-1]**2)
        return j

    def jac(self, theta, x, y):
        '''Derivative of cost function'''
        zs=numpy.dot(x, theta)
        ee=numpy.where(y*zs>=1., 0., -y)[: ,None]
        dj=numpy.zeros(self.theta.shape)
        dj[:-1]=(ee*x[:, :-1]).mean(axis=0)*self.c+theta[:-1] # weights
        dj[-1]=(ee*self.c).mean(axis=0) # bias

    def train(self):
        #-----Optimize using scipy.optimize-----
        if self.method=='optimize':
            opt=optimize.minimize(self.costFunc, self.theta, args=(self.xdata, self.ydata),
                                  jac=self.jac, method='Nelder-Mead')
            self.theta=opt.x

        #-----Optimize using Gradient descent-----
        elif self.method=='GD':
            costs=[]
            lr=self.lr
            # ADAM parameteres
            beta1=0.9
            beta2=0.999
            epsilon=1e-8

            mt_1=0
            vt_1=0
            for ii in range(self.n_iter):
                t=ii+1
                dj=self.jac(self.theta, self.xdata, self.ydata)
                mt=beta1*mt_1+(1-beta1)*dj
                vt=beta2*vt_1+(1-beta2)*dj**2
                mt=mt/(1-beta1**t)
                vt=vt/(1-beta2**t)

```



یک ویژگی جدید به نام مساحت ساختم

```
[51]
df['square']=df['px_width']*df['px_height']
```

ماشین های بردار پشتیبان از روش های یادگیری نظارت شده هستند که برای طبقه بندی رگرسیون و تشخیص داده های پرت استفاده می شود.

از مزایای استفاده اش در فضا هایی با ابعاد بالا موثر هست برای عملکرد تصمیم گیری داده های آموزشی استفاده می شود از نظر حافظه کارآمد هست.

از توابع هسته برای عملکرد تصمیم گیری استفاده می شود. تنظیم هسته بسیار مهم هست که از کدام هسته برای کدام داده استفاده شود. تخمین های احتمالاتی مستقیم ارائه نمی دهد. از اعتبار سنجی برای بررسی مدل استفاده می شود.

اول داده های ترین و تست تقسیم می کنیم. بعد نحوه ی تقسیم بندی کلاس هارا بررسی می کنیم.

```
[ ] y = data["price_range"].values
x_data=data.drop(["price_range"],axis=1)
x = (x_data-np.min(x_data))/(np.max(x_data)-np.min(x_data))
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random
```

```
[ ] from yellowbrick.target import ClassBalance
visualizer = ClassBalance(labels=[0, 1, 2,3])
ClassBalance(y_train, y_test)
visualizer.poof()
```

تعداد کلاس ها تقریباً یکسان هست.

## مدل اول

```
[ 1 ] from sklearn.svm import SVC
      svm=SVC(random_state=1)
      SVC(x_train,y_train)

SVC(C=          battery_power  blue  clock_speed  ...  three_g  touch_screen  wifi
409      0.818303      1.0      0.16  ...      1.0      0.0      1.0
30       0.720107      1.0      0.00  ...      0.0      0.0      1.0
1150     0.251169      0.0      0.32  ...      1.0      0.0      0.0
1949     0.171009      1.0      0.16  ...      1.0      1.0      0.0
424      0.142953      0.0      0.72  ...      0.0      0.0      0.0
...      ...      ...      ...      ...      ...      ...
1800     0.572478      0.0      0.48  ...      1.0      0.0      0.0
1099     0.798931      0.0      0.00  ...      1.0      1.0      0.0
1944     0.028724      1.0      0.00  ...      1.0      1.0      0.0
237      0.812959      1.0      0.80  ...      1.0      1.0      1.0
1064     0.915832      0.0      0.00  ...      1.0      0.0      1.0

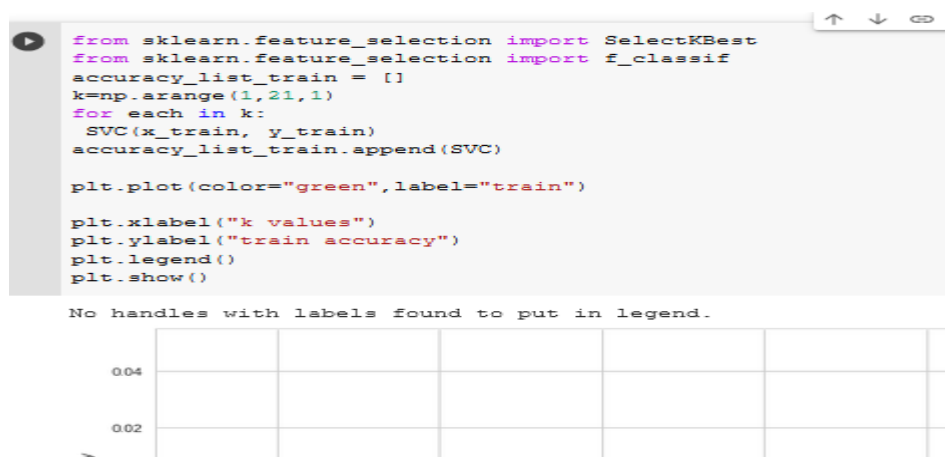
[1590 rows x 20 columns],
break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='scale',
kernel=array([ 0.44710178,  1.34130533,  0.44710178, ..., -1.34130533,
               1.34130533, -0.44710178]),
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

وقتی از کتابخانه اسکالرن دقت در مدل اول در داده های تست و ترین داریم.

حالا می خواهیم این دقت افزایش بدهیم اول ویژگی های غیر ضروری کم می کنیم.

تا دقت بهبود دهیم.بهترین ویژگی ها برای طبقه بندی انتخاب می کنیم.

از روش کلاس بندی آنوا برای تعیین بهترین ویژگی ها استفاده میکنیم.



از این ویژگی ها استفاده می کنیم.

```

df = {'best features number': k, 'train_score': accuracy_list_train}

[ ] selector = SelectKBest(f_classif, k = 5)
x_new = selector.fit_transform(x_train, y_train)
x_new_test=selector.fit_transform(x_test,y_test)
names_train = x_train.columns.values[selector.get_support()]
names_test = x_test.columns.values[selector.get_support()]
print("x train features:",names_train)
print("x test features:",names_test)

x train features: ['battery_power' 'px_height' 'px_width' 'ram' 'sc_h']
x test features: ['battery_power' 'px_height' 'px_width' 'ram' 'sc_h']

```

برای استفاده از الگوریتم بردار های پشتیبان باید برخی پارامتر هایش تنظیم شود.

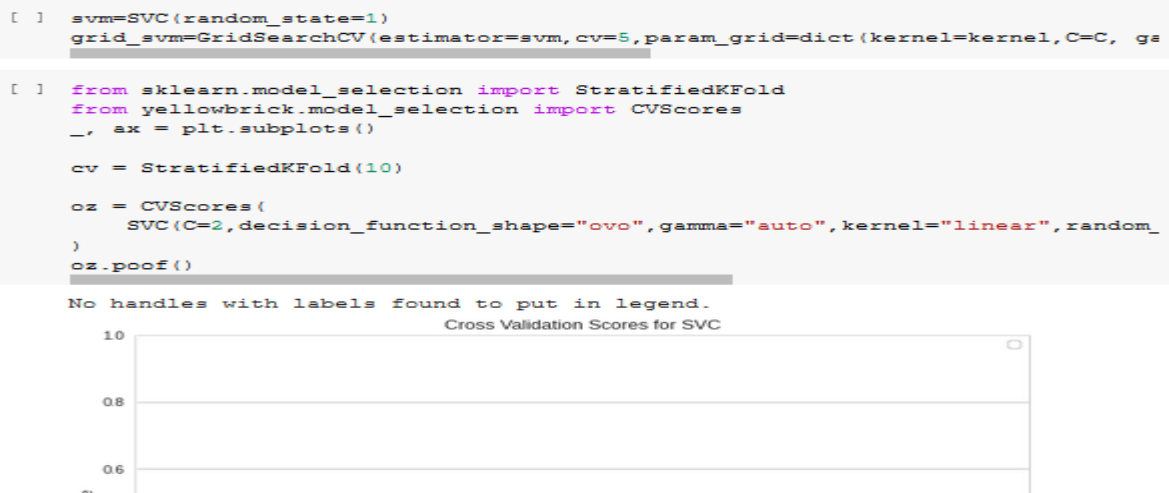
```

from sklearn.model_selection import GridSearchCV

C=[1,0.1,0.25,0.5,2,0.75]
kernel=["linear","rbf"]
gamma=["auto",0.01,0.001,0.0001,1]
decision_function_shape=["ovo","ovr"]

```

بعد از استفاده از بهترین پارامترها هسته rbf از هسته خطی هم استفاده می کنیم.



از اعتبار سنجی برای بررسی بردار های پشتیبان استفاده می کند.

```
[ ] svm_model=SVC(C=2,decision_function_shape="ovo",gamma="auto",kernel="linear",
```

```
[ ] from yellowbrick.classifier import ConfusionMatrix  
cm = ConfusionMatrix(  
    svm_model, classes=[0,1,2,3]  
)  
  
cm.poof()
```

```
[ ] svm_test=x_test[["battery_power","int_memory","px_height","px_width","ram"]]
```

```
svm_test.head()
```

	battery_power	int_memory	px_height	px_width	ram
1556	0.249833	0.661290	0.460204	0.313084	0.839123
1552	0.708751	0.854839	0.526531	0.851802	0.491716
1182	0.138945	0.177419	0.290816	0.615487	0.056387
112	0.244489	0.032258	0.035714	0.983979	0.142704
564	0.048764	0.580645	0.145918	0.447931	0.805184

تا مدل پیش بینی کند ویژگی ها را مقادیر واقعی با مقادیر پیش بینی شده تفاوت ها را بررسی می کنیم دقتش بدست می آورد.

با سپاس