نام درس : شناسایی آماری الگوریتم

تمرین شماره 3

نام و شماره دانشجویی : فاطمه توکلی , 400131016

.3

Preprocessing

Data:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pres
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	...	51.0	24.0	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	...	56.0	21.0	
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	...	53.0	24.0	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	...	51.0	24.0	
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	NaN	NaN	NaN	ESE	...	62.0	36.0	

145460 rows × 23 columns

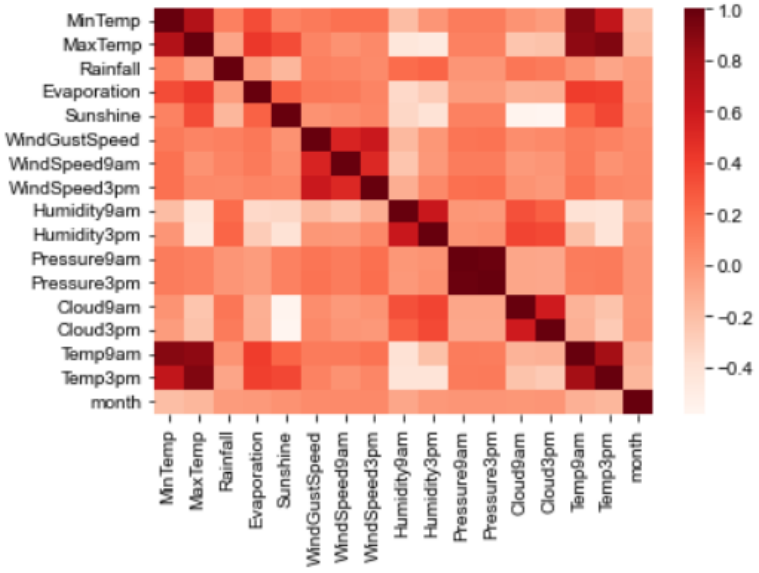
```
R = df[~df['RainTomorrow'].isnull()]
```

```
R['month'] = pd.DatetimeIndex(R['Date']).month
```

```
for i in R:
    R[i] = R[i].fillna(R['month'].median())
R
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pressure3pm
0	2008-12-01	Albury	13.4	22.9	0.6	6.0	6.0	W	44.0	W	...	22.0	1007.7	1007.1
1	2008-12-02	Albury	7.4	25.1	0.0	6.0	6.0	WNW	44.0	NNW	...	25.0	1010.6	1007.8
2	2008-12-03	Albury	12.9	25.7	0.0	6.0	6.0	WSW	46.0	W	...	30.0	1007.6	1008.7
3	2008-12-04	Albury	9.2	28.0	0.0	6.0	6.0	NE	24.0	SE	...	16.0	1017.6	1012.8
4	2008-12-05	Albury	17.5	32.3	1.0	6.0	6.0	W	41.0	ENE	...	33.0	1010.8	1006.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
145454	2017-06-20	Uluru	3.5	21.8	0.0	6.0	6.0	E	31.0	ESE	...	27.0	1024.7	1021.2
145455	2017-	Uluru	2.8	23.4	0.0	6.0	6.0	E	31.0	SE		24.0	1024.6	1020.3

a) ماتریس coefficient برای تمامی ویژگی ها:




```
cor_target = abs(coef)
relevant_features = cor_target[cor_target>0.95]
relevant_features
```

سپس جدولی به عنوان خروجی میدهد که بین دو ویژگی Pressure9am و Pressure3pm تصمیم به حذف pressure9am شد

b) در ابتدا نیاز داریم مقادیر یکتا را برای هر ستون در بیاوریم و مقدار عددی بدهیم تا بتوان از آنها استفاده کر و سپس 10 تا از بهترین ویژگی ها به شرح زیر است:

```
selector = SelectKBest(f_classif, k=10).fit(X_val, y_val)
cols = selector.get_support(indices=True)
new_feature = X_val.iloc[:,cols]
```

```
new_feature
```

	MaxTemp	Rainfall	Sunshine	WindGustSpeed	Humidity9am	Humidity3pm	Cloud9am	Cloud3pm	Temp3pm	RainToday
114766	22.5	0.0	5.4	50.0	48.0	32.0	7.0	7.0	21.8	2
56351	13.3	0.2	6.0	39.0	98.0	62.0	5.0	7.0	13.2	2
139537	32.4	0.0	11.1	33.0	62.0	50.0	1.0	2.0	31.1	2
80963	20.2	0.0	9.2	31.0	100.0	57.0	1.0	2.0	19.3	2
5369	20.3	0.0	6.0	20.0	86.0	51.0	6.0	6.0	19.6	2
...	...	...	...	...	...	...	...	...	...	...
7560	21.1	0.0	6.0	28.0	35.0	28.0	2.0	1.0	20.3	2
34419	15.2	0.0	5.1	69.0	63.0	71.0	3.0	7.0	10.3	2
90116	32.4	24.4	6.0	31.0	81.0	66.0	7.0	3.0	31.5	3
113639	17.9	0.0	6.0	46.0	61.0	63.0	6.0	6.0	15.9	2
75060	17.2	2.0	1.5	33.0	78.0	64.0	7.0	7.0	15.7	3

29092 rows × 10 columns

(c

```
knn = KNeighborsClassifier(n_neighbors=450)

knn.fit(new_feature, y_val)
```

```
print(knn.score(X_test, y_test))
```

```
0.8151016456921588
```

(d

```
a = pd.DataFrame([[10],[5],[0],[23],[40],[68],[10],[5],[1],[1]])
a = np.transpose(a)
print(knn.predict(a))
```

```
[1]
```

e) با استفاده از bulit-in method میتوان احتمال هرکدام از کلاس ها را به دست آورد مثلا در مثال رشت به احتمال 82 درصد 1 میشود یعنی بارندگی داریم و به احتمال 18 درصد جواب 0 است

Part e

```
predictions = knn.predict_proba(a)
predictions
```

```
array([[0.82, 0.18]])
```

.4

(a

```
def patch_extract(image):
    img = cv2.imread(image)
    (m , n, r)= img.shape
    blocks = np.array([img[i:i+8, j:j+8] for j in range(0,n-7,8) for i in range(0,m-7,8)])
    blocks_list=blocks.tolist()
    #convert t0 4200*64
    s=[]
    blocks_asli=[]
    for i in range(0,4200):
        for j in range(0,8):
            s=s+blocks_list[i][j]
        blocks_asli.append(s)
        s=[]
    my_array = np.array(blocks_asli)
    return my_array
```

```
my_array= patch_extract('donald.png')
```

```
my_array.shape
```

```
(4200, 64, 3)
```

(b

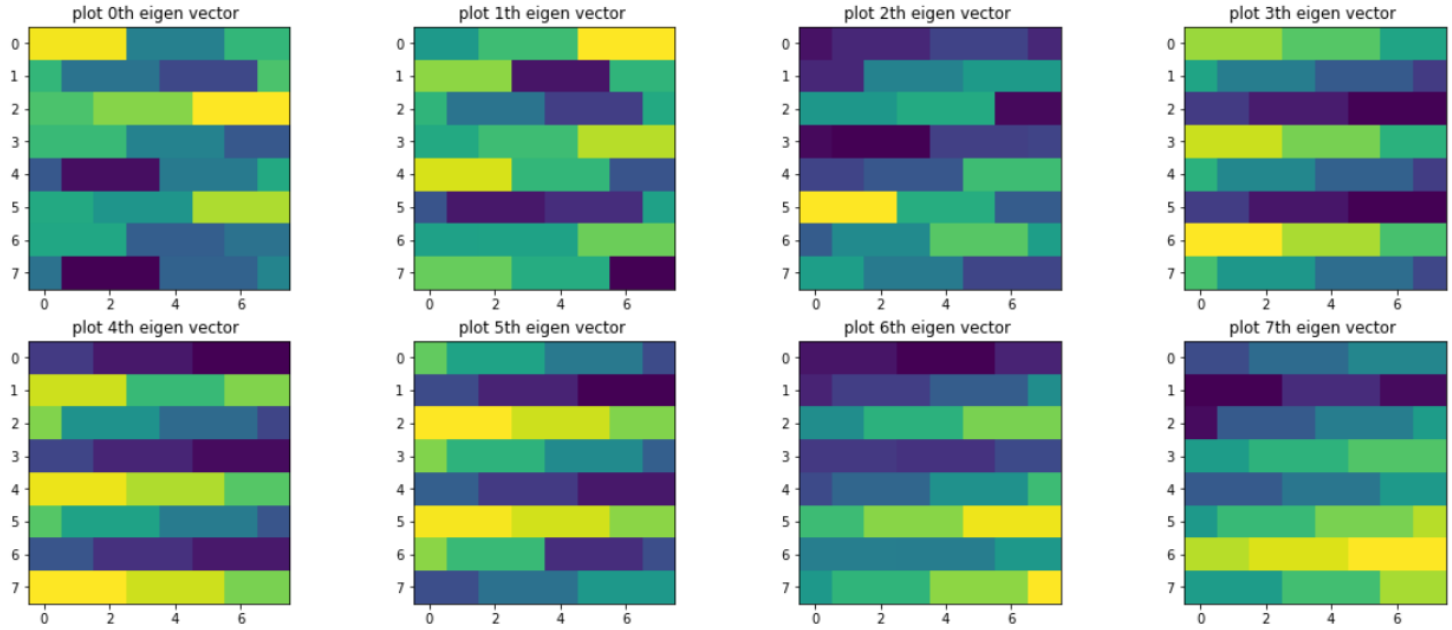
sorted eigenvectors:

```
[[-0.11947512 -0.12521405 -0.12282112 ... -0.12377336 -0.12528236
  -0.12724244]
 [ 0.09871062  0.05904384  0.00583904 ... -0.06135045 -0.01314881
   0.04496825]
 [-0.21238581 -0.22249308 -0.20501055 ...  0.19326025  0.18936895
   0.18403515]
 ...
 [-0.02738716  0.10884342  0.21212592 ... -0.12627434 -0.19496076
  -0.10714315]
 [-0.03703369  0.12974965  0.11625116 ... -0.08988038 -0.21186286
  -0.00395416]
 [-0.0722121  -0.04768545 -0.09604574 ...  0.03778346  0.07888795
   0.18803241]]
```

Sorted eigenvalue:

```
[2.84127601e+00 6.80682081e-02 5.27598138e-02 1.19248881e-02
9.41483323e-03 7.53740405e-03 4.47341870e-03 3.84582048e-03
2.99854980e-03 2.64588477e-03 2.00545061e-03 1.56427645e-03
1.39477843e-03 1.29048890e-03 9.82283824e-04 9.07493944e-04
8.59607113e-04 8.37389194e-04 7.06376493e-04 6.80689673e-04]
```

Plot 8 largest eigenvalue:



(c

```
projection.append (np.array(eigvectors_sort).T[1])
projection.append (np.array(eigvectors_sort).T[4])
projection.append (np.array(eigvectors_sort).T[9])
projection.append (np.array(eigvectors_sort).T[19])
project_mat=np.array(projection)
```

```
reduce_pic= (cv2.merge(project_mat))
reduce_pic.shape
```

```
(4200, 4, 3)
```

```
reduce_pic
```

```
array([[[-0.01941134, -0.01941134, -0.01941134],
        [ 0.02408792,  0.02408792,  0.02408792],
        [-0.01276486, -0.01276486, -0.01276486],
        [ 0.00151296,  0.00151296,  0.00151296]],

       [[-0.01430549, -0.01430549, -0.01430549],
        [ 0.02407012,  0.02407012,  0.02407012],
        [-0.01915991, -0.01915991, -0.01915991],
        [-0.00050212, -0.00050212, -0.00050212]],

       [[-0.01888378, -0.01888378, -0.01888378],
        [ 0.0262695 ,  0.0262695 ,  0.0262695 ],
        [-0.01208169, -0.01208169, -0.01208169],
        [-0.00087214, -0.00087214, -0.00087214]],

       ...,

       [[-0.0191938 , -0.0191938 , -0.0191938 ],
        [ 0.00035609,  0.00035609,  0.00035609],
        [-0.01920528, -0.01920528, -0.01920528],
        [-0.0003483 , -0.0003483 , -0.0003483 ]],

       [[-0.01894713, -0.01894713, -0.01894713],
        [-0.00193223, -0.00193223, -0.00193223],
        [-0.00609951, -0.00609951, -0.00609951],
        [ 0.00976245,  0.00976245,  0.00976245]],

       [[-0.00845418, -0.00845418, -0.00845418],
        [ 0.00321678,  0.00321678,  0.00321678],
        [-0.00903496, -0.00903496, -0.00903496],
        [-0.00170434, -0.00170434, -0.00170434]]])
```

(d

```
def patch_reconstruct(img,numrows,numcols,show = False):
    height = int(img.shape[0] / numrows)
    width = int(img.shape[1] / numcols)
    tiles = []
    for row in range(numrows):
        for col in range(numcols):
            y0 = row * height
            y1 = y0 + height
            x0 = col * width
            x1 = x0 + width
            tiles.append(img[y0:y1, x0:x1])
            if show:
                cv2.imshow("tile",img[y0:y1, x0:x1])
                cv2.rectangle(img,(x0,y0),(x1,y1),(255),1)
                cv2.imshow("Tile",img)
                cv2.waitKey(0)
                cv2.destroyAllWindows
    return tiles
```

(e

sorted eigenvector for joe image

```
[[-0.12307952 -0.12386087 -0.12460549 ... -0.12528621 -0.12561685
  -0.12527278]
 [ 0.16048943  0.11848753  0.066789   ... -0.07887508 -0.01399254
   0.04686398]
 [-0.1619492  -0.1809498  -0.19142003 ...  0.16907925  0.16995955
   0.16477516]
 ...
 [ 0.08725509  0.14289138  0.1667446  ...  0.0706638   0.03410883
  -0.0081821 ]
 [ 0.07487827 -0.08018896 -0.08344873 ...  0.18871512 -0.10097342
  -0.1752796 ]
 [-0.01383105  0.07275887  0.10354412 ...  0.15464991  0.2128355
   0.24964713]]
```

Sorted eigenvalue:

```
[4.67670749e+00 6.56936856e-02 4.15608819e-02 1.17494970e-02
9.68496745e-03 5.68315863e-03 3.79572449e-03 3.43428544e-03
2.58365317e-03 1.91717993e-03 1.60703942e-03 1.44397642e-03
1.15983038e-03 1.01128834e-03 9.86305720e-04 9.16397645e-04
8.28852439e-04 7.07719541e-04 6.42900495e-04 5.68725615e-04]
```

تاثیر پارامتر k به اینصورت است که زمانیکه شروع به افزایش تعداد مؤلفه ها می کنیم، دریافت تصویر واضح تری از شکل داده های اصلی می کنیم.

.5

(a

یکی از eigenvector ها به شرح زیر است:

```
Original number of features: 64
Reduced number of features: 15
[[ 2.22044605e-16 -1.73094658e-02 -2.23428838e-01 -1.35913306e-01
  -3.30323047e-02 -9.66340867e-02 -8.32944369e-03  2.26900124e-03
  -3.20516495e-04 -1.19308902e-01 -2.44451667e-01  1.48512736e-01
  -4.67319476e-02 -2.17740742e-01 -1.48136762e-02  4.47779617e-03
  -4.94136621e-05 -7.95419427e-02  8.33951497e-02  2.15915341e-01
  -1.72126791e-01 -1.63712103e-01  2.86444524e-02  4.23251873e-03
   9.85488464e-05  6.42319050e-02  2.54093316e-01 -3.56771020e-02
  -2.09462576e-01 -4.31311423e-02  5.13118739e-02  2.13422753e-04
   0.00000000e+00  1.59950877e-01  3.68690780e-01  1.64406821e-01
   8.52007942e-02  3.72982906e-02  2.15866872e-02  0.00000000e+00
   1.28865573e-03  1.06945278e-01  3.03067455e-01  2.47813041e-01
   2.09637295e-01  1.22325175e-02 -3.69458478e-02  1.61485010e-03
   6.93023483e-04 -8.35144581e-03 -5.58598952e-02  9.30534194e-02
   1.07387727e-01 -1.37734567e-01 -6.32879408e-02  9.61670117e-04
   9.55078542e-06 -1.40786852e-02 -2.35675491e-01 -1.41225592e-01
  -9.15964423e-03 -8.94184706e-02 -3.65977166e-02 -1.14684988e-02]]
```

(b

از روش grid search cross validation استفاده شده است:

```
params = {"bandwidth": np.logspace(-1, 1, 20)}
grid = GridSearchCV(KernelDensity(), params)
grid.fit(X_pca)
print("best bandwidth: {0}".format(grid.best_estimator_.bandwidth))
kde = grid.best_estimator_
```

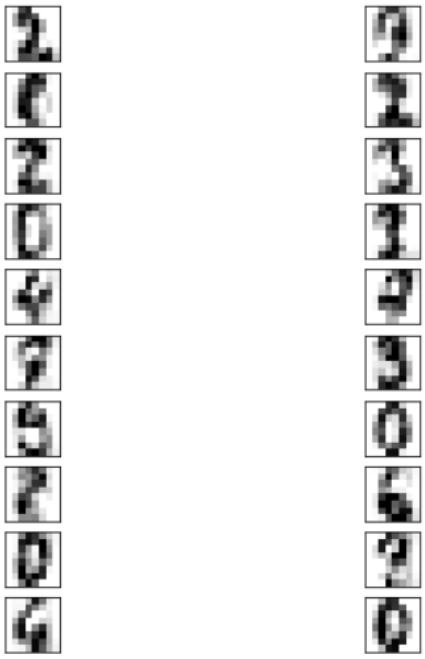
```
best bandwidth: 3.79269019073225
```

(c

```
def plot_digits(data):
    fig, ax = plt.subplots(10, 2, figsize=(8, 8),
                           subplot_kw=dict(xticks=[], yticks=[]))
    fig.subplots_adjust()
    for i, axi in enumerate(ax.flat):
        im = axi.imshow(data[i].reshape(8, 8), cmap='binary')
        im.set_clim(0, 16)
```

```
data_new = kde.sample(20, random_state=0)
data_new = pca.inverse_transform(data_new)
```

```
plot_digits(data_new)
```



d)دقت داده ها بهتر می شود و bandwidth= 1.8 است

تجزیه و تحلیل اجزای اصلی (PCA) برای الف) حذف نویز و ب) کاهش ابعاد استفاده می شود. نویز را از بین نمی برد، اما می تواند نویز را کاهش دهد.
اساساً یک تبدیل خطی متعامد برای یافتن پیش‌بینی همه داده‌ها به ابعاد k استفاده می‌شود، در حالی که این ابعاد k دارای بالاترین واریانس هستند.