میانترم عملی درس هوش مصنوعی

مهدی عباسعلی پور

۹۸۱۳۳۱۰

ترتیب سوالات بر اساس پی دی اف سوالات موجود در فایل زیپ می باشد .

۱- خواندن دیتا و نمایش برخی اطلاعات دیتا مشاهده شد که دیتا داده ی از دست رفته ندارد :

```
data=pd.read_csv('./Fifa 23 Players Data.csv')
```

Python

```
data.head()
```

Python

	Known As	Full Name	Overall	Potential	Value(in Euro)	Positions Played	Best Position	Nationality	Image Link	Age	...	LM Rating
0	L. Messi	Lionel Messi	91	91	54000000	RW	CAM	Argentina	https://cdn.sofifa.net/players/158/023/23_60.png	35	...	91
1	K. Benzema	Karim Benzema	91	91	64000000	CF,ST	CF	France	https://cdn.sofifa.net/players/165/153/23_60.png	34	...	89
2	R. Lewandowski	Robert Lewandowski	91	91	84000000	ST	ST	Poland	https://cdn.sofifa.net/players/188/545/23_60.png	33	...	86
3	K. De Bruyne	Kevin De Bruyne	91	91	107500000	CM,CAM	CM	Belgium	https://cdn.sofifa.net/players/192/985/23_60.png	31	...	91
4	K. Mbappé	Kylian Mbappé	91	95	190500000	ST,LW	ST	France	https://cdn.sofifa.net/players/231/747/23_60.png	23	...	92

5 rows × 89 columns

```
data1=data[data['Overall']>=85]
```

Python

الف) آن هایی که امتیاز بالای ۸۵ را داشتند را در data1 ذخیره کردم . هم چنین کلاس model را ساختم که هریک از بخش های سوال را بتوان با آن اجرا کنم در این کلاس هنگام ایجادش تعیین می کنمی که چند خوشه نیاز داریم . در تابع fit یه pipline از یک نرمال کننده و pca و یک kmean می سازیم . در نهایت همه ی این لایه ها را با داده آموزش fit می کنیم و آموزش می دهیم .پس از آن متد های transorm, predict را برای خروجی تعریف می کنیم . در ابتدای همه ی این متد ها داده های غیر عددی را همانطور که سوال خواسته است است در نظر نمی گیریم (با _get_numeric_data) .

در مورد متد pca_transform فقط لایه آخر kmeans را درنظر نمی گیریم و فقط اعمال نرمال سازی و کاهش بعد را انجام می دهیم که این متد برای رسم نمودار کمک می کند .

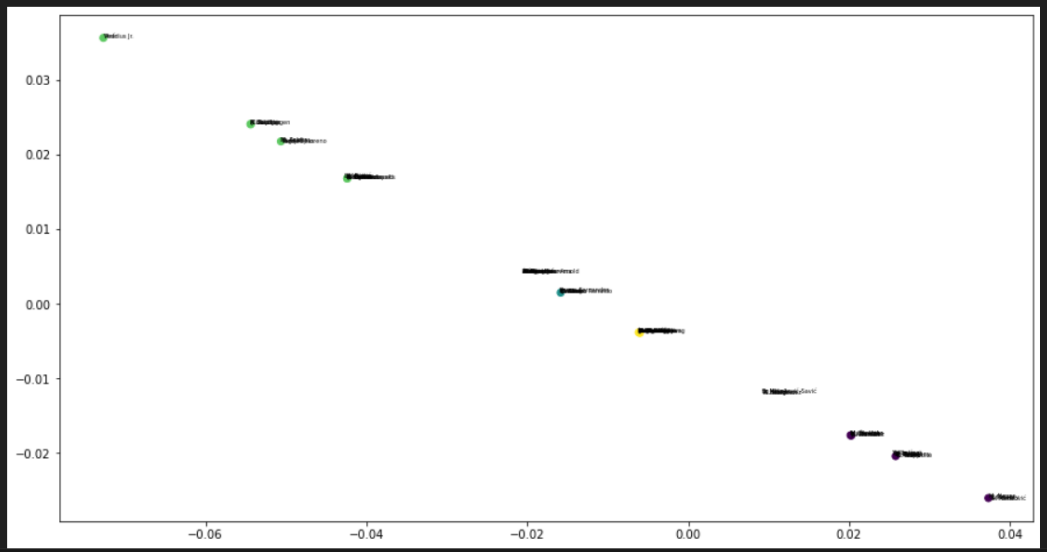
```python
from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.preprocessing import Normalizer
class Model():
    def __init__(self,n) -> None:
        self.k=n
    def fit(self,data):
        numeric_data=data._get_numeric_data()
        self.normalizer=Normalizer()
        self.pca=PCA(n_components=2)
        self.kmeans = KMeans(n_clusters=self.k, random_state=0, n_init="auto")
        self.pipe = Pipeline(steps=[("scaler", self.normalizer), ("pca", self.pca), ("kemans",self. kmeans)])
        self.pipe.fit(numeric_data)
    def pca_transform(self,data):
        numeric_data=data._get_numeric_data()
        normal_data=self.normalizer.transform(numeric_data)
        return self.pca.transform(normal_data)
    def transform(self,data):
        numeric_data=data._get_numeric_data()
        return self.pipe.transform(numeric_data)
    def predict(self,data):
        numeric_data=data._get_numeric_data()
        return self.pipe.predict(numeric_data)
```

ب)

به داده های آموزش مدل را فیت می کنیم k=5 و نمودار و اسم بازیکنان رانمایش می دهیم . این نمدار برحسب دو ویژگی که توسط pca استخراج شده اند رسم شده است و برای شلوغ نشدن نمودار برای فقط تعدادی از بازیکنان رسم شده است .

```python
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
fig.set_figwidth(15)
fig.set_figheight(8)
index=list(range(0,91,7))
ax.scatter(model.pca_transform(data1)[:,0][index], model.pca_transform(data1)[:,1][index],c=model.predict(data1)[index])
names=data1['Known As'].to_numpy()
for i, txt in enumerate(names):
    ax.annotate(txt, (model.pca_transform(data1)[:,0][i],model.pca_transform(data1)[:,1][i]),fontsize=5)
```

[8]



ج) در نتایج خوشه بندی مشاهده می کنید که مسی در کلاس۴ و رونالدو و امباپه و طارمی در کلاس دیگر ۲ قرار می گیرند .

```
model=Model(5)
model.fit(data1)
model.transform(messi)
print("Messi",model.predict(messi))
print("Ronaldo",model.predict(ronaldo))
print("Mbapeh",model.predict(mbapeh))
print("Taremi",model.predict(taremi))
# player3=np.array([messi,ronaldo,mbapeh])
# model.predict(player3)
```

[24] ✓ 0.0s

```
Messi [4]
Ronaldo [2]
Mbapeh [2]
Taremi [2]
```

د) به کار بردن سنجش kmeans با استفاده از صحت به علت این که امکان دارد لیبل کلاس های kmeansبا لیبل کلاس های واقعی تطابق نداشته باشد به صورت مستقیم با مشکل مواجه می شود . یعنی اگر مانند زیر صحت را بسنجیم آن نتیجه درست نمی باشد . اما طبق چیزی که من از یک بحث داخل گروه متوجه شدم می توان انواع مختلف اینکه هرخوشه کدام بهترین موقعیت باشد را در نظر گرفت و میانگین گرفت و باز هم طبق یکی از پیام ها یک حالت کلفی است و برای یک حالت ازشماره گذاری خوشه ها و کلاس ها صحت به صورت زیر است و کمتر از ۱۰ درصد است .

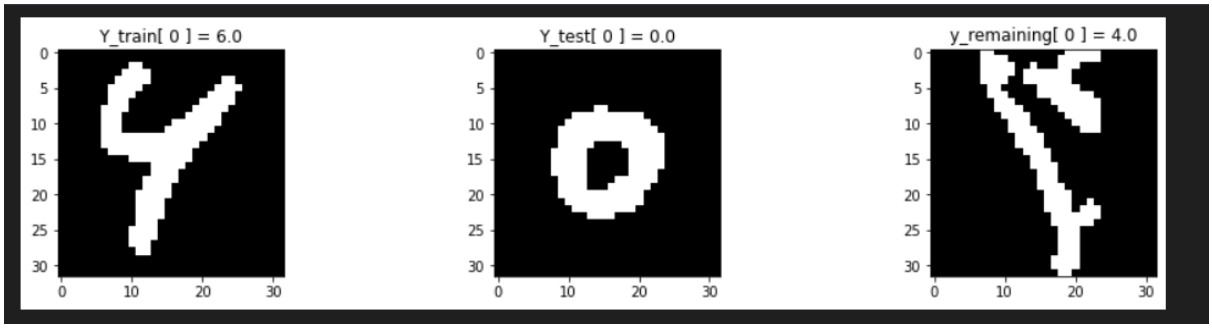
```
from sklearn.preprocessing import LabelEncoder
number_bestpos=data['Best Position'].unique().shape[0]
model2=Model(number_bestpos)
model2.fit(data)
pred=model2.predict(data)
label = LabelEncoder()
true_clusters=label.fit_transform(data['Best Position'])
sum(true_clusters==pred)/pred.shape[0]
```

[50] ✓ 0.3s

```
0.07449161227682184
```

۲- برای این سوال برای خواندن دیتا ست با استفاده از کدی که در منبع دیتا ست ذکر شده بود دیتا را خواندم (https://github.com/alirezafarzipour/Persian_Handwritten_Digit_Recognition).

در ادامه هم سه نمونه دیتا را نمایش دادم که این بخش را هم از کد منبع قرار دادم چون آماده بود .



پس از تخت کردن هرتصویر به صورت یک وکتور از فیچر های ورودی ، الگوریتم MDC را با کلاس MDC پیاده کردم . در این کلاس در متد fit پروتوتایپ هر خوشه را محاسبه می کنم و آن را درون بردار w آبجکت مدل MDC نگه داری می کنم

```
from sklearn.preprocessing import MinMaxScaler

class MDC():
    def __init__(self):
        self.w={}
    def distance(self,a,b):
        return np.sqrt(sum((a-b)**2))
    def fit(self,X,y):
        scaler = MinMaxScaler()
        x_scaled=scaler.fit_transform(X)
        classes=np.unique(y)
        for label in classes:
            a=x_scaled[y==label]
            self.w[label]=np.mean(a,axis=0)
```

در زیر پروتوتایپ کلاس ها را مشاهده می کنید:

```
c.w
[33] ✓ 0.0s
```

```
{0.0: array([0., 0., 0., ..., 0., 0., 0.], dtype=float32),
 1.0: array([0.        , 0.        , 0.00016667, ..., 0.00016667, 0.        ,
        0.        ], dtype=float32),
 2.0: array([0.        , 0.        , 0.00016667, ..., 0.        , 0.        ,
        0.        ], dtype=float32),
 3.0: array([0.        , 0.00066667, 0.0025    , ..., 0.0005    , 0.00016667,
        0.        ], dtype=float32),
 4.0: array([0.        , 0.00033333, 0.0005    , ..., 0.0015    , 0.00066667,
        0.00033333], dtype=float32),
 5.0: array([0.00066667, 0.00066667, 0.002     , ..., 0.00033333, 0.        ,
        0.        ], dtype=float32),
 6.0: array([0.00083333, 0.00233333, 0.0055    , ..., 0.00616667, 0.0025    ,
        0.00066667], dtype=float32),
 7.0: array([0.00066667, 0.00233333, 0.00733333, ..., 0.        , 0.        ,
        0.        ], dtype=float32),
 8.0: array([0.    , 0.    , 0.    , ..., 0.0205, 0.007 , 0.0015], dtype=float32),
 9.0: array([0.        , 0.        , 0.        , ..., 0.01      , 0.00383333,
        0.00066667], dtype=float32)}
```

در متد predict هم فاصله با پروتو تایپ ها سنجیده می شود و اندیس آن پروتوتایپ به عنوان خوشه ی پیش بینی شده برمی گردد .

```
def predict(self,X):
    y=np.array([])
    scaler =MinMaxScaler()
    x_scaled=scaler.fit_transform(X)
    for sample in x_scaled:
        v={}
        for i in self.w:
            v[i]=self.distance(self.w[i],sample)
        y=np.append(y,min(v,key=v.get))
    return y
```

در ادامه مدل را آموزش می دهیم و صحت را محاسبه می کنیم که تقریبا ۷۳ درصد می باشد .

```
c=MDC()
c.fit(X_train_flattened,y_train)
[7] ✓ 0.4s

y_pred=c.predict(X_test_flattened)
[8] ✓ 31.8s

np.sum(y_pred==y_test)/y_test.shape[0]
[10] ✓ 0.0s
... 0.73695
```

در ادامه هم ۵ نمونه از مواردی که اشتباه تشخیص داده شده است را به نمایش می گذاریم .

```
X_test_false=X_test[~(y_pred==y_test)]
y_pred_false=y_pred[~(y_pred==y_test)]
y_test_false=y_test[~(y_pred==y_test)]
ind=np.random.choice(X_test_false.shape[0],5)
y_test_sample=y_test_false[ind]
X_test_sample=X_test_false[ind]
y_pred_sample=y_pred[ind]
```

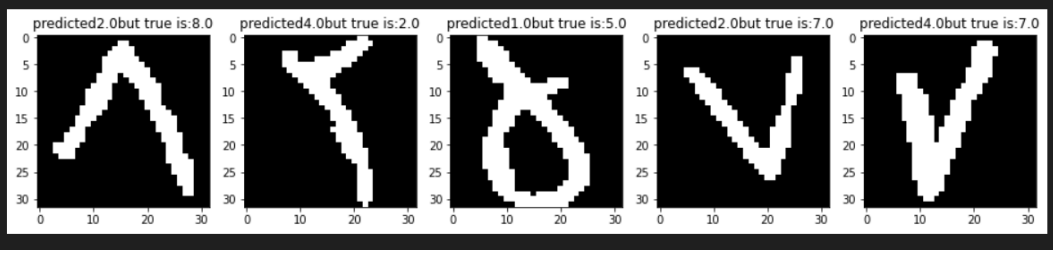
[12] ✓ 0.0s

```
fig = plt.figure(figsize=(16, X_test_sample.shape[0]))

for i in range(X_test_sample.shape[0]):
    fig.add_subplot(1,X_test_sample.shape[0], i+1)
    plt.title('predicted' + str(y_pred_sample[i])+'but true is:'+str(y_test_sample[i]))
    plt.imshow(X_test_sample[i].reshape([32, 32]), cmap='gray')
```

[13] ✓ 0.5s

...



۳-

الف)

```
import pandas as pd
df=pd.read_csv('./abc.csv')
N=2
df.groupby('A').head(N)['C'] # n top in order

df.groupby('A')['C'].nlargest(N)    # n top larget in feature C
```

[4] ✓ 0.0s

```
A
1  5     74
   1     22
2  4     36
   3     35
3  10    41
   2     15
Name: C, dtype: int64
```

ب)

```
import pandas as pd
data={'region':['A','B','C']*3,'month':['Feb','June','July','June','July','Feb','July','Feb','June'],'amount':[11,12,65,3,2,58,36,8,9]}
df=pd.DataFrame(data=data)
df['Percentage'] = df.groupby(
    ['month'])['amount'].transform(lambda x: (x/x.sum()) * 100)
df
```

[42] ✓ 0.0s

	region	month	amount	Percentage
0	A	Feb	11	14.285714
1	B	June	12	50.000000
2	C	July	65	63.106796
3	A	June	3	12.500000
4	B	July	2	1.941748
5	C	Feb	58	75.324675
6	A	July	36	34.951456
7	B	Feb	8	10.389610
8	C	June	9	37.500000

پ)

```
import scipy
import numpy as np
arr=np.arange(100).reshape(10,10)
scipy.ndimage.median_filter(arr,size=(3,3))
```

```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 90, 91, 92, 93, 94, 95, 96, 97, 98]])
```

د)

```
a=np.array([1,5,6,9,8,7,5,6,6,3,14,63])
b=np.array([12,5,6,9,8,5,5,5,5,64,7,14])
```

[55] ✓ 0.0s

```
np.argmin(np.abs(b-a[:,None]), axis=1)
```

[56] ✓ 0.0s

```
array([ 1,  1,  2,  3,  4, 10,  1,  2,  2,  1, 11,  9], dtype=int64)
```