دانشگاه تهران

دانشکده علوم و فنون نوین

پروژه درس

هوش مصنوعی

استاد درس: دکتر رضایی

آرمان بختیاری

830499019

مقدمه

بیماری دیابت که به آن بیماری قند هم گفته می شود از جمله ی بیماری های بسیار شایع در کشور ماست که اگر به موقع و درست کنترل نشود میتواند منجر به قطع عضو یا حتی مرگ شود. در این بیماری توانایی تولید هورمون انسولین در بدن از بین می‌رود یا بدن در برابر انسولین مقاوم شده و بنابراین انسولین تولیدی نمی‌تواند عملکرد طبیعی خود را انجام دهد. نقش اصلی انسولین پایین آوردن قند خون توسط سازوکارهای مختلف است.

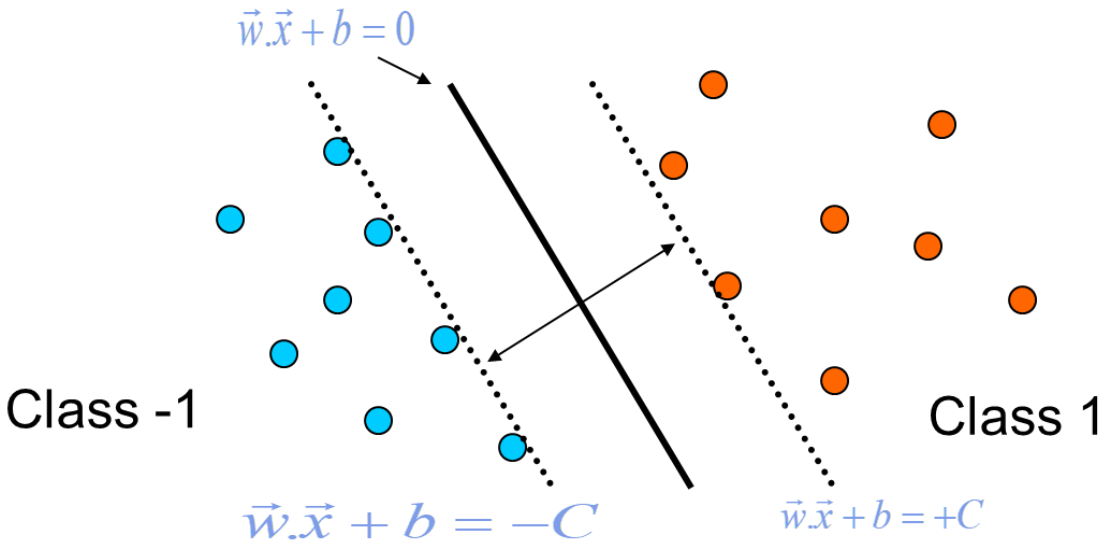
هدف از این پروژه طراحی سیستمی به وسیله هوش مصنوعی است که با آن بتوان با دادن برخی مشخصه های حیاتی فرد به سیستم، ابتلای فرد به دیابت را پیش بینی نمود. به دلیل آنکه هدف ما دسته بندی افراد به دو گروه فرد مبتلا به دیابت و فرد سالم است (classification)، میتوانیم از الگوریتم هایی که بدین منظور طراحی شده اند استفاده کنیم. از جمله این الگوریتم ها SVM (Support Vector Machine) میباشد که در ادامه به معرفی آن می پردازیم و سپس کد مربوطه را توضیح میدهیم.

SVM

SVM دسته بندی کننده ای است که جزو شاخه های Kernel Methods در یادگیری ماشین محسوب میشود که در سال 1992 توسط Vapnik معرفی شده و بر پایه statistical learning theory بنا گردیده است. شهرت SVM به خاطر قدرت تشخیص آن در حروف دست نویس است.

SVM با فرض آنکه دسته ها به صورت خطی جدا پذیر باشند، ابر صفحه هایی با حداکثر حاشیه (maximum margin) را بدست می آورد که دسته ها را جدا کنند. در صورتی که داده ها به

شکل خطی جداپذیر نباشند، داده ها به فضای با ابعاد بیشتر نگاشت شده تا بتوان آنها را در این فضای جدید به صورت خطی جدا نمود.



در پایتون با دستور `from sklearn import svm` می توان آن را از کتابخوانه فراخواند و استفاده کرد.

کد

قسمت اول کد را به فراخوانی دستوراتی که در بخش های مختلف کد به آن ها نیاز پیدا میکنیم اختصاص میدهیم. خود دستورات را در ادامه و در بخش مربوطه بیشتر توضیح خواهیم داد.

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
     from sklearn import svm
     from sklearn.preprocessing import StandardScaler
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score
```

ابتدا باید داده های خود را بخوانیم که برای این کار از دستور pd.read_csv استفاده میکنیم. سپس برای اینکه اطلاعاتی کلی از داده ها داشته باشیم با دستور ()head. 5 سطر ابتدایی داده ها را مشاهده کرده و با ()describe. اطلاعاتی درباره ماکزیمم مینیمم مقدار ویژگی های مختلف داده، میانگین آنها و ... به دست می آوریم.

```
#getting the dataset
datas = pd.read_csv('/content/diabetes.csv')
datas.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
[ ] datas.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

همینطور با استفاده از دستور ()corr. کورلیشن هر ویژگی از داده را با ویژگیOutcome که خروجی صفر و یکی ماست بدست می آوریم. همانطور که میبینیم glucose بیشترین کورلیشن را دارد که منطقی هم هست.

```
[ ]  #seeking correlations between features
     corr_matrix = datas.corr()
     corr_matrix["Outcome"].sort_values(ascending=False)
```

```
Outcome                     1.000000
Glucose                     0.466581
BMI                         0.292695
Age                         0.238356
Pregnancies                 0.221898
DiabetesPedigreeFunction    0.173844
Insulin                     0.130548
SkinThickness               0.074752
BloodPressure               0.065068
Name: Outcome, dtype: float64
```

حال باید داده های خود را آماده پردازش کنیم. برای این امر ابتدا باید ویژگی ها و خروجی این ویژگی ها را از هم جدا کنیم. دستور drop. این امکان را به ما میدهد که ستون یا سطری را که مدنظر داریم از داده حذف کنیم. اگر ستونی را حذف میکنیم axis=1 و اگر سطری را حذف میکنیم axis=0 قرار میدهیم. در اینجا ما میخواهیم ستون Outcome که خروجی ما هست را از باقی اطلاعات داده جدا کنیم و آنرا مستقلا در متغیری دیگر که به عنوان label از آن استفاده میکنیم ذخیره کنیم.

```
[ ]  #preparing the dataset
     data_features = datas.drop(columns = 'Outcome', axis=1)
     data_labels = datas['Outcome']
     data_features.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

به دلیل آنکه واحد ویژگی ها یکسان نیست و مقدار ویژگی های مختلف هم در یک بازه نیستند باید داده ها را نورمالیزه کنیم و متناسب با هر ویژگی مقداری بین -1 تا 1 به آن اختصاص دهیم تا فرآیند یادگیری به درستی صورت گیرد. برای این کار از دستور StandardScaler استفاده میکنیم.

```
[ ]  #preparing the dataset
     #normalizing the features
     normalizer = StandardScaler()
     norm_features = normalizer.fit_transform(data_features)
     norm_features
```

```
array([[ 0.63994726,  0.84832379,  0.14964075, ...,  0.20401277,
         0.46849198,  1.4259954 ],
       [-0.84488505, -1.12339636, -0.16054575, ..., -0.68442195,
        -0.36506078, -0.19067191],
       [ 1.23388019,  1.94372388, -0.26394125, ..., -1.10325546,
         0.60439732, -0.10558415],
       ...,
       [ 0.3429808 ,  0.00330087,  0.14964075, ..., -0.73518964,
        -0.68519336, -0.27575966],
       [-0.84488505,  0.1597866 , -0.47073225, ..., -0.24020459,
        -0.37110101,  1.17073215],
       [-0.84488505, -0.8730192 ,  0.04624525, ..., -0.20212881,
        -0.47378505, -0.87137393]])
```

یک کار مرسوم در فرآیند یادگیری تقسیم داده ها به داده های train و test است تا با داده های train فرآیند یادگیری انجام شود و با داده های test کارآمدی یادگیری انجام شده سنجیده شود. همچنین این کار از پدیده overfitting هم جلو گیری میکند. ما اینجا 20 درصد داده ها را به عنوان داده test در نظر میگیریم. این امر به کمک دستور train_test_split انجام میگیرد. برای اینکه همه 1 ها برای مثال در y_test هم قرار نگیرند از stratify=data_labels استفاده میکنیم که data_labels 0 و 1 هایی هستند که خود پیشتر جدا کرده بودیم.

```
[ ]  #spliting the dataset into train and test datas
     #using 80% of the datas for training the model and 20% to test it
     x_train, x_test, y_train, y_test = train_test_split(norm_features, data_labels, test_size=0.2, stratify=data_labels, random_state=43)
     y_test.shape
```

```
(154,)
```

حال مدل خود را بر پایه SVM خطی با استفاده از داده های train میسازیم.

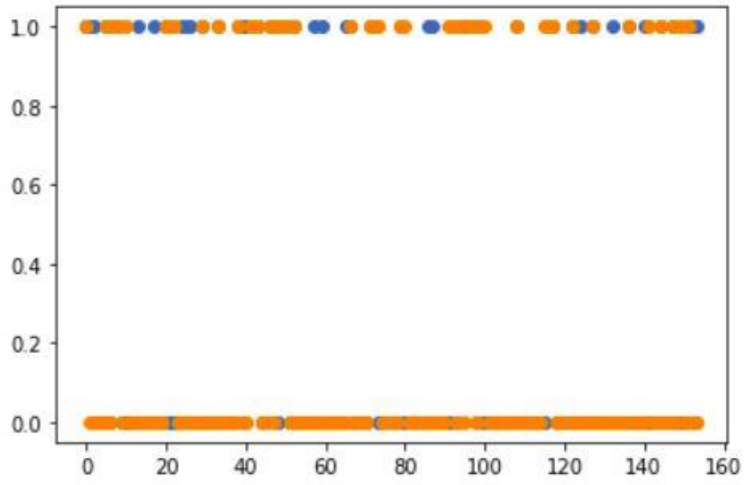
```
[ ]  #building a linear svm model
     clf = svm.SVC(kernel='linear')
     clf.fit(x_train,y_train)
```

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

حال با استفاده از داده های test مدل ساخته شده را ارزیابی میکنیم. ابتدا داده های x_test را به عنوان ورودی به مدل داده و خروجی آن را با استفاده از دستور predict ساخته و با y_test مقایسه میکنیم. در اینجا هم از رسم مقدار های خروجی با رنگ های مختلف استفاده شده که گویا نیست و هم از دستور accuracy_score برای سنجیدن دقیق صحت مدل استفاده شده است که نتیجه آن صحت 79 درصدی است.

```
[ ]  #testing the model on test datas
     predicts = clf.predict(x_test)
     plt.scatter(np.arange(0,154), y_test)
     plt.scatter(np.arange(0,154), predicts)
```

```
<matplotlib.collections.PathCollection at 0x7fa6aadc6110>
```




```
[ ]  #testing the model on test datas
     accuracy = accuracy_score(predicts, y_test)
     accuracy
```

```
0.7922077922077922
```

نهایتا ما باید نمونه ی همین ویژگی ها را از فرد جدیدی که قصد دارد تا ابتلای خود به دیابت را ارزیابی کند، گرفته و ابتلای او را به بیماری پیش بینی کنیم. به این منظور ابتدا ویژگی های بیمار جدید که به صورت list به ما داده میشود را به شکل یک آرایه که قابل پردازش باشد در می آوریم و سپس برای آنکه به عنوان ورودی بتوانیم آن را به مدل خود بدهیم نورمالیزه میکنیم. خروجی ما به صورت 0 و 1 است. برای سهولت در فهم آن از عبارت Diabetes= - برای خروجی 0 (فرد سالم) و از Diabetes= + برای خروجی 1 (فرد بیمار) استفاده میکنیم.

```
[ ]  #getting features from a hypothetical patient and predrict if he/she has diabetes or not
     patient_features = (0,187,50,33,392,33.9,0.826,34)
     pat_features = np.array(patient_features).reshape(1,-1)
     norm_pat_features = normalizer.transform(pat_features)
     pred = clf.predict(norm_pat_features)
     pred
```

```
array([1])
```

```
[ ]  if (pred[0] == 0):
       print('Diabetes: - ')
     if (pred[0] == 1):
       print('Diabetes: + ')
```

```
Diabetes: +
```