به نام خدا

پروژه تشخیص نوع وسایل نقلیه از روی ابعاد آنها

محمدعلی افتخاری 9612358003

سید فرهاد حسینی 9612358016

نام درس : مبانی داده کاوی

استاد مربوطه : دکتر محرم منصوری زاده

مقدمه:

در این پروژه یک دیتاست با 10053 داده ، داده شده است که در آنها مختصات دو نقطه از وسایل نقلیه داده شده که ما باید با طبقه بند های مختلف نوع آنها را حدس بزنیم و دقت را ارزیابی کنیم.

مرحله اول:

در ابتدا فایل اکسل دیتاست را داخل colab اضافه کردیم وبا استفاده از pandas آن را میخوانیم.

```
from google.colab import files
uploaded = files.upload()
import pandas as pd
cars=pd.read_excel('cars.xlsx')
```

مرحله دوم:

در این مرحله اول دیتاست را به فرمت numpy بردیم و data و target را جدا کردیم.(صفات و مقادیر کلاس ها)

```
car=cars.values
data = car[:,0:4]        #(10053, 4)
target = car[: , 4]      # (10053,)
```

مرحله سوم:

در این مرحله برخی ویژگی های جدید مثل عرض و ارتفاع عکس و همچنین مساحت و محیط را استخراج میکنیم (مساحت و محیط در انجام محاسبات بی تاثیر بود) .
سپس ویژگی هارا بصورت افقی به همراه یک ستون 1 به هم متصل کردیم .

```
import numpy as np
height =  data[:,3] -  data[:,1]
width =  data[:,2] -  data[:,0]
s = width * height
p = 2*(width + height)

atts = np.hstack((np.ones((10052,1)) ,data, width.reshape((-1,1)) ,
height.reshape((-1,1)) , p.reshape((-1,1)) ,s.reshape((-1,1)) ))  #(10053,
5)

atts.shape #(bias  ,  x1 ,y1 ,x2 ,y2 , w , h , s ,p)
```

مرحله چهارم:

در این مرحله نوع خودرو را به شکل one hot تبدیل کردیم و آنها را به طور افقی به هم چسباندیم.

```
#------------------one hot encoding--------------------
bus = (target == "bus") * 1
microbus = (target == "microbus") * 1
sedan = (target == "sedan") * 1
minivan = (target == "minivan") * 1
suv = (target == "suv") * 1
truck = (target == "truck") * 1
classes = np.hstack((bus.reshape((-1,1)) , microbus.reshape((-1,1))
,sedan.reshape((-1,1)),minivan.reshape((-1,1)),suv.reshape((-1,1)),truck.r
eshape((-1,1))))     #(10053, 6)
np.set_printoptions(threshold=np.inf)

classes.shape

(bus ,  microbus  , sedan   ,  minivan   , suv   , truck)#
```

مرحله پنجم :

در این مرحله سطرهای دیتاست را شافل کرده تا برای جداکردن مجموعه های train و test آماده شود .

```
#------------------shuffle-----------------------
dataset=np.hstack((atts , classes))
from sklearn.utils import shuffle
shu=shuffle(dataset)

shu.shape   #(10053, 11)
```

مرحله ششم :

در این مرحله 80 درصد دیتا را برای train و 20 درصد باقیمانده را برای test اختصاص دادیم.

```
#----------------------test & train -----------------------------
train=shu[0:8042,:]     #(8042, 11)
test=shu[8042:,:]       #(2011, 11)
test.shape
```

مرحله هفتم (طبقه بند رگرسیون خطی):

دقت مدل:

در ابتدا با استفاده از sklearn یک طبقه بند رگرسیون خطی ساختیم و آن را با دیتای train ، fit کردیم و با reg.coef_ میتوانیم ضرایبی را که این طبقه بند برای ما ساخته را ببینیم.
سپس با استفاده از تابع score دقت طبقه بند را روی داده train به دست آوردیم که 0.39 به دست آمد.

```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(train[:,:7] ,train[: ,9:15])
reg.coef_.shape                          #(6, 7)
y_tr=train[: ,9:15]
y_tr=y_tr.astype('int')
reg.score(train[:,:7], y_tr)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/base.py
  "multioutput='uniform_average').", FutureWarning)
0.39853731380807184
```

سپس دقت را روی داده ی test امتحان کردیم که حدود 0.38 شد.

```
y_ts=test[: ,9:15]
y_ts=y_ts.astype('int')
reg.score(test[:,:7], y_ts)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/base.py:
  "multioutput='uniform_average').", FutureWarning)
0.3849572091860513
```

شباهت jaccard:

شباهت jaccard که معیار سختگیرانه تری برای شباهت است هم استفاده شده است.

```
u=train[:,0:7]@(reg.coef_[0,:].reshape(7,1))
an=(u > 0.25)*1
an.shape   #(8042, 1)
compare=np.hstack((u , train[:,9].reshape(8042,1) ))


tru = np.ones((8042,1))
#---------------------------------rate-------------------------------
# np.array([0,1,1,1,1,1,,1,1,1,1,,0,0,0,0,0,1,1,1,1,0,1])
# rate = (1 -  np.sum((train[:,4]-an[:,0])**2)/8042)
# f11 = np.sum(((train[:,4] == an[:,0]) and ( train[:,4] == tru ) )*1)
# rate=f11/(f)
from scipy.spatial import distance
1 - distance.jaccard(train[:,9] , an[:,0])
# compare

# pred1 = reg.predict(train[:,:7])

# # rate = (1 -  np.sum((y1-v11)**2)/150)
# # compare
# # rate
# pred1.shape   #(8042, 6)
```

```
0.3707414829659319
```

دلیل دقت کم :

دلیل دقت کم این است که داده از مدل خطی پیروی نمیکند.

مرحله هشتم(طبقه بند رگرسیون logestic):

دقت مدل:

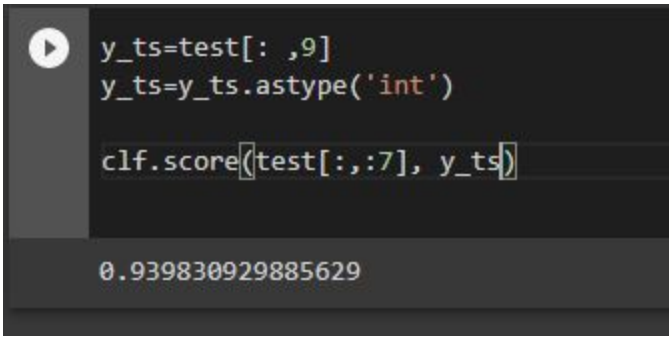
در ابتدا با استفاده از sklearn یک طبقه بند logestic ایجاد کردیم و آن را با دیتای train، fit کردیم و با reg.coef_ میتوانیم ضرایبی را که این طبقه بند برای ما ساخته را ببینیم.
سپس با استفاده از تابع score دقت طبقه بند را روی داده train به دست آوردیم که 0.944 بدست آمد.

logestic regression

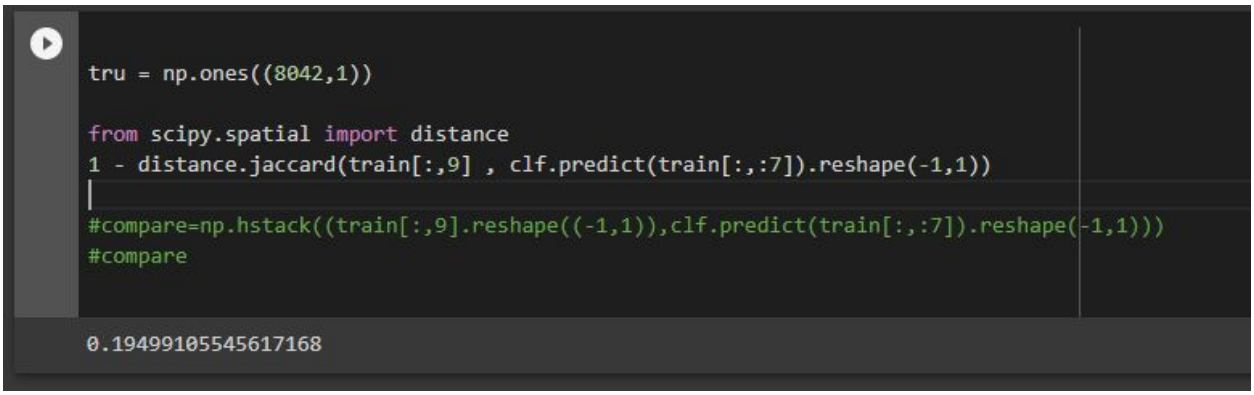
```
[10] from sklearn.linear_model import LogisticRegression
     clf = LogisticRegression(random_state=0)


     y_tr=train[: ,9]
     y_tr=y_tr.astype('int')


     clf.fit(train[:,:7] ,y_tr )

     # clf.predict(train[:,:7])

     # clf.predict_proba(X[:2, :])
     clf.score(train[:,:7], y_tr)
```

```
0.9440437702064163
```

سپس دقت را روی داده ی test امتحان کردیم که حدود 0.939 شد.



شباهت jaccard:



دلیل دقت خوب :

رگرسیون لجستیک برخلاف رگرسیون خطی میتواند داده ها را بر روی مدل خیر خطی هم مدل کند پس منطقی است که قدرت آن از رگرسیون خطی بالاتر باشد و همچنین نسبت به داده های پرت هم مقاومت نشان میدهد .

مرحله نهم(طبقه بند درخت تصمیم):

دقت مدل:

در ابتدا با استفاده از sklearn یک طبقه بند decisionTree ایجاد کردیم و آن را با دیتای train، fit کردیم و با reg.coef_ میتوانیم ضرایبی را که این طبقه بند برای ما ساخته را ببینیم. سپس با استفاده از تابع score دقت طبقه بند را روی داده train به دست آوردیم که 1.0 بدست آمد.

decision tree

```
[20] >>> from sklearn import tree
     X = [[0, 0], [1, 1]]
     Y = [0, 1]
     clft = tree.DecisionTreeClassifier()
     clft = clft.fit(train[:,:7],  y_tr)
     clft
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
tree.plot_tree(clft)
```

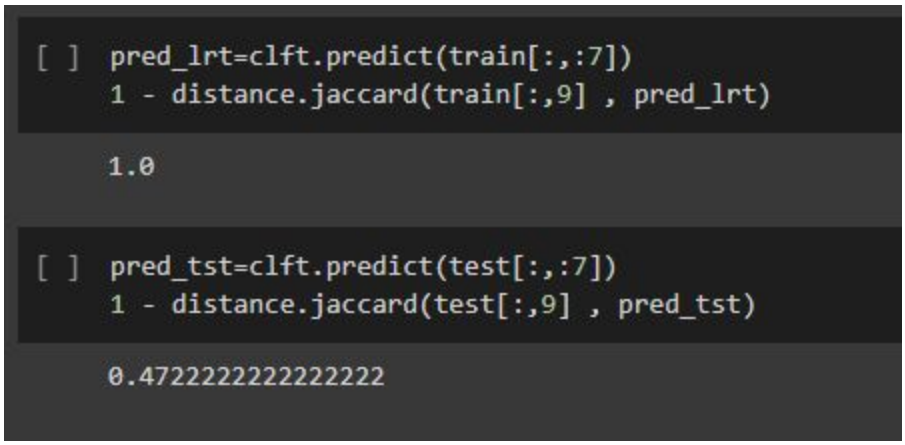
```
[ ] clft.score(train[:,:7], y_tr)

    1.0
```

سپس دقت را روی داده ی test امتحان کردیم که حدود 0.96 شد.

```
[ ] clft.score(test[:,:7], y_ts)

    0.9621890547263682
```

شباهت jaccard:

شباهت بر روی داده train همانطور که میدانیم 1 است و روی داده test هم 0.47 است.



دلیل دقت خوب :

همانطور که میدانیم درخت تصمیم توانایی این را دارد که با تک تک داده های آموزشی آموزش دیده و بازه ی تصمیم گیری خود را به مقدار خیلی زیادی دقیق کند . اما از طرفی معمولا مشکل بیش برازش در درخت تصمیم بوجود می آید یعنی مدل را بیش از حد پیچیده میکند که خوشبختانه در این مدل همچین مشکلی وجود ندارد و دقت برروی داده های test بالای 96 درصد میباشد .

مرحله دهم(شبکه عصبی پرسپترون):

در ابتدا با استفاده از sklearn یک طبقه بند شبکه عصبی پرسپترون ایجاد کردیم و آن را با دیتای train، fit کردیم و با reg.coef_ میتوانیم ضرایبی را که این طبقه بند برای ما ساخته را ببینیم.

سپس با استفاده از تابع score دقت طبقه بند را روی داده train به دست آوردیم که 0.94 بدست آمد.

multi level Perceptron

```
[ ] from sklearn.neural_network import MLPClassifier
    clfnn = MLPClassifier(solver='lbfgs', alpha=0.1, hidden_layer_sizes=(4, 2), random_state=1)
    clfnn.fit(train[:,:7],  y_tr)

    MLPClassifier(activation='relu', alpha=0.1, batch_size='auto', beta_1=0.9,
                  beta_2=0.999, early_stopping=False, epsilon=1e-08,
                  hidden_layer_sizes=(4, 2), learning_rate='constant',
                  learning_rate_init=0.001, max_fun=15000, max_iter=200,
                  momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
                  power_t=0.5, random_state=1, shuffle=True, solver='lbfgs',
                  tol=0.0001, validation_fraction=0.1, verbose=False,
                  warm_start=False)

[ ] clfnn.score(train[:,:7], y_tr)

    0.9424272569012684
```

سپس دقت را روی داده ی test امتحان کردیم که حدود 0.94 شد.

```
[29] clfnn.score(test[:,:7], y_ts)

     0.9448035803083044
```

لینک notebook:

https://colab.research.google.com/drive/1XxXxNfUMGLxgt54w-WrUqd0UZGVUPJy5?usp=sharing

پایان