SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Klasifikacija zvijezda i galaksija

Seminarski rad

Raspoznavanje uzoraka i strojno učenje

Lucija Azinović

Osijek, 2021.

Sadržaj

1. UVOD

U ovom projektu treba napisati programski kod nadziranog učenja nad podacima s obzirom na to pripadaju li podaci skupu zvijezda, galaksija ili kvazara. S obzirom da je puno manji postotak kvazara u odnosu na zvijezde i galaksije i zato što kvazar nije pravo svemirsko tijelo nego “lažna” zvijezda, njega će se eliminirati iz učenja. Podaci su dobiveni sa satelita i već je za svaki od njih 10000 obilježeno o kojoj se klasi radi – zvijezdi, galaksiji ili kvazaru.

2. NADZIRANO UČENJE

Nadzirano učenje je način strojnog učenja gdje je cilj odrediti nepoznatu funkcionalnu ovisnost između ulaznih veličina i izlazne veličine na temelju podatkovnih primjera. Pri tome su podatkovni primjeri parovi koji se sastoje od vektora ulaznih veličina i vrijednosti izlazne veličine. Broj ulaznih veličina označava se s *n* pa se vektor ulaznih veličina može zapisati u obliku:

$$x^{(i)} = [\, x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)} \,]^T$$

Raspoloživi podatkovni primjeri (mjerni uzorci) na temelju kojih se određuje nepoznata funkcionalna ovisnost primjenom algoritama strojnog učenja često se naziva i skup za učenje:

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(n_{train})}, (y^{(n_{train})})\}$$

Dvije su osnovne namjene dobivenog modela – predikcija i zaključivanje.

2.1. Klasifikacija

Ako je veličina *y* diskretna, tada se ovaj problem naziva klasifikacija, a ako je veličina *y* kontinuirana, tada se ovaj problem naziva regresija.

Ovaj projekt bavi se binarnom klasifikacijom koja je jednostavan primjer učenja klase na temelju pozitivnih i negativnih podatkovnih primjera odnosno u ovom slučaju zvijezda i galaksija.

Cilj je izgraditi model koji će za uzorak ulaznih veličina provesti predikciju izlazne veličine, tj. kojoj klasi pripada uzorak od mogućih klasa.

Kao pokazatelji vrjednovanja modela koriste se matrica zabune, točnost, učestalost pogrešne klasifikacije, preciznost, odziv i specifičnost.

Matrica zabune (eng. Confusion Matrix) je matrica koja pokazuje koliko je točno i netočno klasificiranih primjera određenog skupa podataka primjenom modela strojnog učenja. Kod binarne klasifikacije izgleda ovako:

	Predicted	
Actual	TN (true negative)	FP (false positive)
	FN (false negative)	TP (true positive)

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$missclassification\ rate = 1 - accuracy$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$specificity = \frac{TN}{TN + FP}$$

Točnost (engl. accuracy) je udio točno klasificiranih primjera u cijelom skupu. Učestalost pogrešne klasifikacije (engl. missclassification rate) definira se kao udio pogrešno klasificiranih primjera u cijelom skupu. Preciznost (engl. precision) je udio točno klasificiranih primjera u skupu koje model klasificira kao klasa +. Odziv (engl. recall) je udio točno klasificiranih primjera u skupu primjera koji pripadaju klasi +. Specifičnost (engl. specificity) je udio točno klasificiranih primjera u skupu svih primjera koji pripadaju klasi -.

2.2. Logistička regresija

Logistička regresija bavi se računanjem vjerojatnosti pripadnosti podataka određenoj klasi. Kako bi se povećala robusnost linearne regresije kao klasifikatora, koristi se zajedno s logističkom funkcijom:

$$h_\theta(x) = g(\Theta^T x) \qquad gdje\ je \qquad g(z) = \frac{1}{1 + e^{-z}}$$

Iako je model nelinearan, on predstavlja linearnu granicu u ulaznom prostoru. Funkcija $h_\theta(x)$ je derivabilna, ograničena na interval [0,1] i sigmoidalnog oblika.

Metoda gradijentnog spusta:

1. Odaberi početnu vrijednost vektora parametara Θ; definiraj duljinu koraka α

2. Simultano osvježi svaki element vektora Θ;

$$\Theta_j \coloneqq \Theta_j - \alpha \frac{\vartheta J(\Theta)}{\vartheta \Theta_j} \qquad za\ \ j = 1, \dots, m$$

3. Ako nije zadovoljen kriterij zaustavljanja tada idi na 2.; u suprotnom zaustavi optimiranje

3. RJEŠENJE PROBLEMA

3.1. Obrada podataka i implementacija algoritma

Podaci su “očišćeni” i pripremljeni za razvoj modela, što se vidi na slici 1. i 2.

```
# delete columns which don't have info for this classification (because it is a simple model)
data.drop(['objid', 'rerun', 'specobjid', 'plate'], axis=1, inplace=True)
```

Slika 1: Eliminirani su podaci koji ne sadrže informaciju za određivanje klase

```
# delete columns which don't have info for this classification (they are correlated with the telescope features and the time when the pictures were taken)
data.drop(['g', 'r', 'i', 'mjd'], axis=1, inplace=True)
```

Slika 2: Podaci vezani uz opis teleskopa nisu važni za klasifikaciju pa su eliminirani

Na slici 3. može se vidjeti u terminalu koliki je postotak zvijezda, galaksija i kvazara u skupu svih podataka, a na slici 4. je napravljen novi skup podataka koji ne sadrži kvazare.

```
GALAXY    49.98
STAR      41.52
QSO        8.50
```

Slika 3: Postoci koji govore koliko ima galaksija, zvijezda i kvazara

```
data1 = new_data[new_data['class']!='QSO']
```

Slika 4: Kvazari više nisu u novom skupu podataka

Na slici 5. može se vidjeti podjela podataka na skupove za trening i test:

```
# create train and data sets
from sklearn.model_selection import train_test_split
X = data1.drop('class', axis=1)
y = data1['class']
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.25, random_state=1551)
X_train.head() #print output
```

Slika 5: Podjela podataka na trening i test

Daljnja obrada podataka vidi se na slici 6.

```
# preprocessing --> the package thar provides several common utility functions and transformer classes to change raw feature vectors
# MinMaxScaler --> transforms features by scaling each feature to a given range
# fit --> compute the minimum and maximum to be used for later scaling
# transform --> scale features according to feature_range
# column_stack --> used to stack 1D arrays as columns into a 2D array
from sklearn.preprocessing import MinMaxScaler
train_scale = MinMaxScaler().fit(X_train[['ra', 'dec', 'u', 'z', 'run', 'field', 'redshift', 'fiberid']])
train_trans_data = train_scale.transform(X_train[['ra', 'dec', 'u', 'z', 'run', 'field', 'redshift', 'fiberid']])
X_train = np.column_stack([train_trans_data,X_train[[1,2,3,4,5,6]]])
```

Slika 6: Obrada podataka: transformiranje podataka za daljnju obradu

Algoritam prvo izračunava linearni model odnosno linearnu regresiju (slika 7), a pritom koristi sigmoidnu funkciju (slika 8.) za izračun vjerojatnosti pa zatim računa funkciju troška (slika 9.)

```
for i in range(0,iterations):
    val = np.matmul(X,theta) # calculate linear regression
    y_prob = sigmoid(val) # calculate probability

    # append --> adds an element to the end of the list
    J = -1/m * sum((y*np.log(y_prob)) + (1- y) * np.log(1-y_prob))
    cost.append(J) # append cost function values to list

    # Update theta values as a gradient of the cost function
    for t in range(0,n):
        temp = round(theta[t] - alpha/m * sum((y_prob - y)*X[:,t]),4)
        t_temp.append(temp)

    theta = []
    theta = t_temp
    t_temp = []
```

Slika 7: Linearna regresija

```
# Sigmoid function
def sigmoid(val):
    y = 1/(1 + np.exp(-val)) # math.exp() calculates e to the power of -(theta * X)
    return y
```

Slika 8: Definiranje sigmoidne funkcije

```
J = -1/m * sum((y*np.log(y_prob)) + (1- y) * np.log(1-y_prob))
```

Slika 9: Funkcija troška

Idući korak je korištenje granice odluke za odlučivanje kojoj klasi pripada podatak s obzirom na granicu odluke (slika 10).

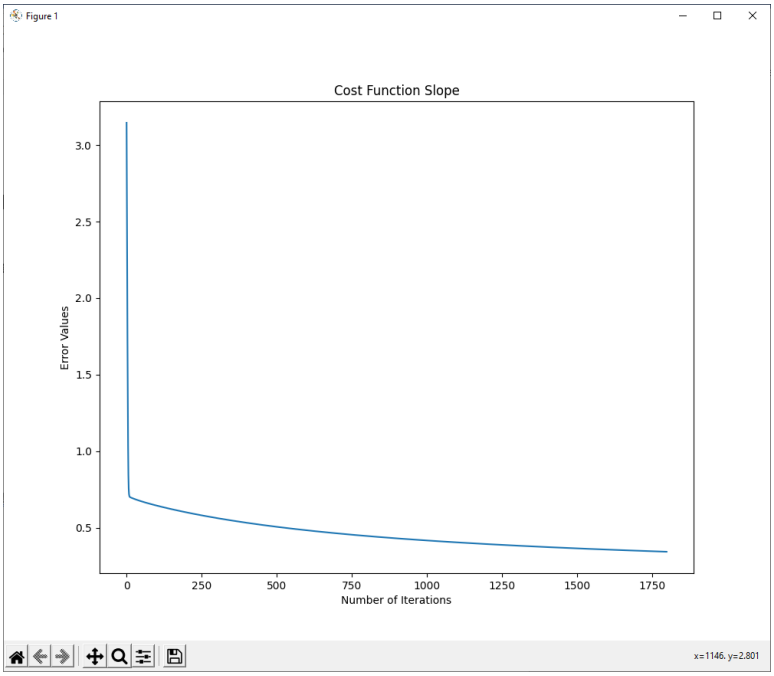
```
y_pred = [1 if y>0.5 else 0 for y in y_prob] # Convert probabilities to classes with 0.5 decision boundary
```

Slika 10: Granica odlučivanja

4. REZULTATI

Treniranje je provedeno na 10000 podataka s 1800 iteracija i koeficijentom učenja alpha = 0,5.

Na slici 11. može se vidjeti kako se smanjuje greška tijekom učenja; na početku funkcije troška događa se veliki pad jer je početni niz theta inicijaliziran na jedinice i zbog toga model ima veliku grešku u početnim iteracijama, a stoga i veliku promjenu koeficijenata theta (slika 12.). Kasnije se funkcija spušta u minimum.



Slika 11: Funkcija troška

```
The final Theta values are:  [1.0623, -0.4633, -0.1603, -5.7669, 7.0537, -0.359, 0.2272, -13.4247, 0.6712, 0.9924, 1.2012, 1.1012, 0.9385, 0.7954, 0.9916]
```

Slika 12: Niz koeficijenata theta

Na idućim slikama prikazan je opis modela; točnost (slika 13.), matrica zabune (slika 14.) te preciznost, odziv i težinski prosjek preciznosti i odziva (slika 15.)

```
The accuracy of the model is : 92.4 %
```

Slika 13: Točnost modela

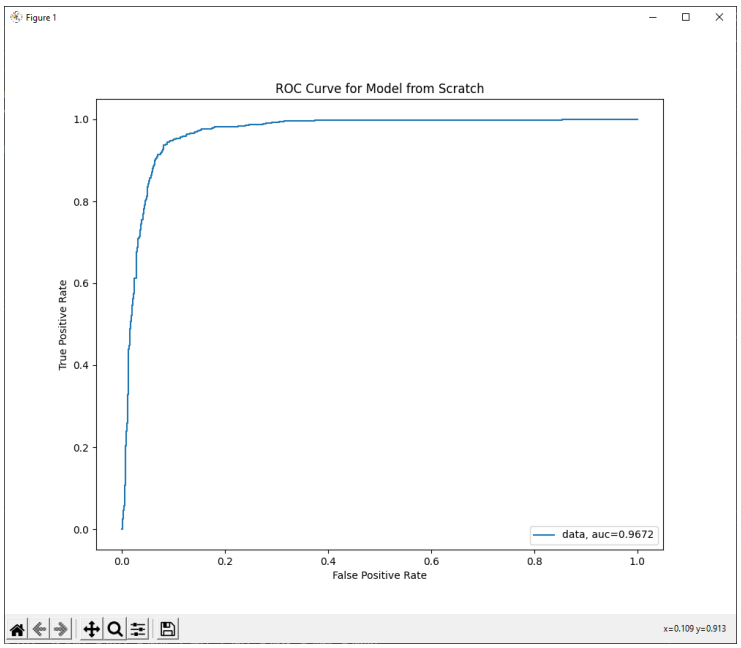
```
Confusion Matrix:
 [[1149  121]
 [  52  966]]
```

Slika 14: Matrica zabune

```
Precision =  0.8887
Recall =  0.9489
F-Score =  0.9178
```

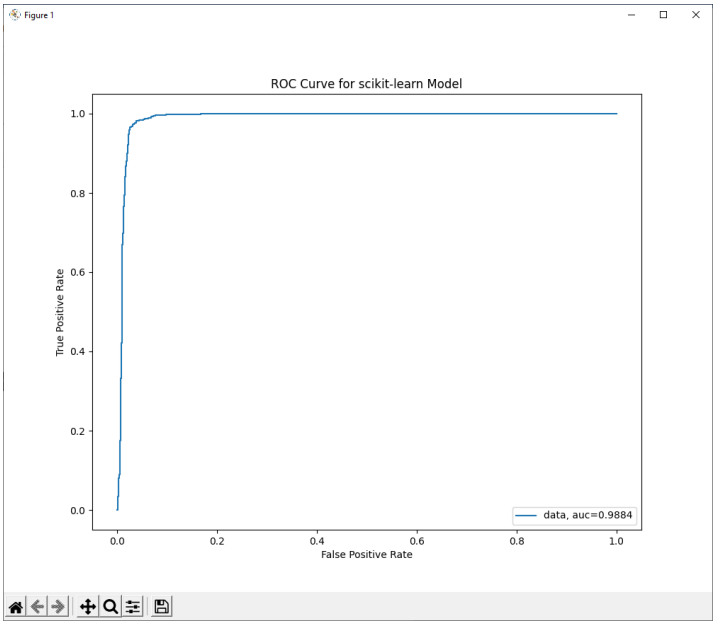
Slika 15: Preciznost, odziv i težinski prosjek preciznosti i odziva

ROC curve (eng. Receiver operating characteristic) je krivulja koja govori kako se odnose vrijednosti matrice zabune TN, FN, FP i TP kad se mijenja granica odluke na funkciji granice odlučivanja, slika 16. AUC (eng. Area under the ROC curve) je područje ispod ROC krivulje, a govori koliko je točan klasifikator. U ovom slučaju AUC iznosi 0,9672.



Slika 16: ROC curve

Kad se model učenja usporedi s ugrađenim SGD (eng. stochastic gradient descent) klasifikatorom iz paketa sklearn, dobije se ROC krivulja kao na slici 17. s točnošću, vrijednostima matrice zabune, preciznošću, odzivom te težinskim prosjekom preciznosti i odziva kao na slikama 18., 19. i 20.



Slika 17: ROC curve za ugrađeni model

```
The accuracy of the model is : 96.0 %
```

Slika 18: Točnost ugrađenog modela

```
Confusion Matrix:
 [[1198   72]
 [  13 1005]]
```

Slika 19: Matrica zabune za ugrađeni model

```
Precision =  0.9331
Recall =  0.9872
F-Score =  0.9594
```

Slika 20: Preciznost, odziv i težinski prosjek preciznosti i odziva za ugrađeni model

Iz rezultata se vidi da ugrađeni model ima bolju preciznost (96 %) od glavnog modela (92,4 %).

5. ZAKLJUČAK

U današnje vrijeme klasifikacija se sve više koristi za istraživanje svemira jer sateliti daju velike baze podataka pa to poprilično olakšava istraživanje svemira. Prikupljeni podaci se uvijek moraju obraditi prije učenja zbog postojućih anomalija, abnormalnosti i disbalansa podataka. Zatim se vrši skaliranje podataka za lakše i učinkovitije učenje. Algoritam koristi logističku regresiju koja vrlo brzo odredi dobre parametre, a zatim se učenje usporava i greška teži u lokalni minimum.

Napravljena je usporedba izrađenog i ugrađenog modela i pokazalo se da veću točnost ima ugrađeni model zbog kompliciranijeg algoritma učenja odnosno koristi stohastički spust gradijenta.