|  |  |
| --- | --- |
| etf_logo.jpg | **Електротехнички факултет, Универзитет у Београду**  **Катедра за рачунарску технику и информатику** |
|  |  |

**Пројекат из предмета**

**Проналажење скривеног знања**

професор: Вељко Милутиновић

асистент: Бојан Фурлан

студент: Милан Бранковић

3129/11

[milan.brankovic@rocketmail](mailto:milan.brankovic@rocketmail).com

2012

Садржај

[1 Опис проблема 3](#_Toc314652364)

[2 Опис решења 4](#_Toc314652365)

[2.1 Задатак 1 4](#_Toc314652366)

[2.2 Задатак 2 4](#_Toc314652367)

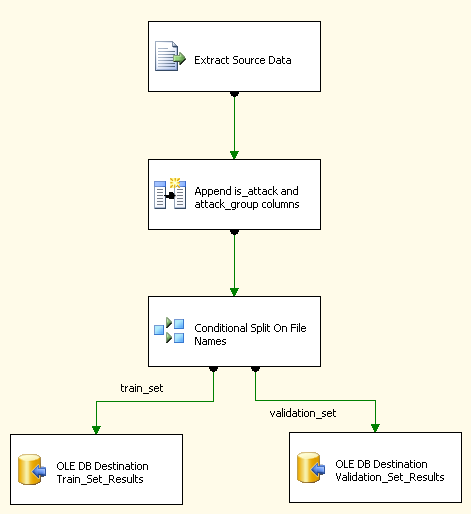
[2.3 Задатак 3 10](#_Toc314652368)

1 Опис проблема

Систем за детекцију упада (енг. *Intrusion Detection System - IDS*) је софтверски или хард-верски производ који прати догађаје и надгледа стање унутар рачунарског система или мреже. Главни циљ оваквих система је анализа и детекција неодговарајућих или неправил-них акција од стране корисника који се могу налазити како изван, тако и унутар самог сис-тема. Задатак учења/тренирања IDS-a огледа се у изградњи предиктивног модела (класи-фикатора) који може направити разлику између „лоших“ конекција (упада или напада) и „добрих“ нормалних конекција.

2 Опис решења

## 2.1 Задатак 1



Слика 1: Data Flow

На слици 1 је приказан изглед Data Flow прозора. Компоненте ће бити описане одозго – надоле и слева на десно. Прва компонента учитава податке из фолдера, следећа компоне-нта врши тражену трансформацију, на крају предпоследња компонента одабира где ће сместити податке у зависности од тога који се сет процесира. Последње две компоненте учитавају податке у базу података.

## 2.2 Задатак 2

На почетку израчунавања KNN алгоритма се израчунавају минималне удаљеноси и ранг (енг. range). Потом се ради нормализација обе табеле. Затим се пролази кроз све редове валидационог сета и рачунају се удаљености од чворова у тренинг сету, али само за оне чворове који имају исти сервис и протокол, као и flag. Три најближа чвора се убацују у помоћну табелу, а затим се у зависности од тога да ли је напад или не убацују израчунате (предиктоване) вредности за is\_attack и attack\_group, као и стварне вредности за тај улаз валидационог сета. Овде ће бити дат само приказ алгоритма за израчунавање Еуклидског растојања.

WHILE @@FETCH\_STATUS = 0

BEGIN

/\*

select all variables except attack\_type

from selected row of validation set

where id = @ID

\*/

SELECT @duration = duration,

@protocol\_type = protocol\_type,

@service = service,

@flag = flag,

@src\_bytes = src\_bytes,

@dst\_bytes = dst\_bytes,

@land = land,

@wrong\_fragment = wrong\_fragment,

@urgent = urgent,

@hot = hot,

@num\_failed\_logins = num\_failed\_logins,

@logged\_in = logged\_in,

@num\_compromised = num\_compromised,

@root\_shell = root\_shell,

@su\_attempted = su\_attempted,

@num\_root = num\_root,

@num\_file\_creations = num\_file\_creations,

@num\_shells = num\_shells,

@num\_access\_files = num\_access\_files,

@num\_outbound\_cmds = num\_outbound\_cmds,

@is\_host\_login = is\_host\_login,

@is\_guest\_login = is\_guest\_login,

@count = count,

@srv\_count = srv\_count,

@serror\_rate = serror\_rate,

@srv\_serror\_rate = srv\_serror\_rate,

@rerror\_rate = rerror\_rate,

@srv\_rerror\_rate = srv\_rerror\_rate,

@same\_srv\_rate = same\_srv\_rate,

@diff\_srv\_rate = diff\_srv\_rate,

@srv\_diff\_host\_rate = srv\_diff\_host\_rate,

@dst\_host\_count = dst\_host\_count,

@dst\_host\_srv\_count = dst\_host\_srv\_count,

@dst\_host\_same\_srv\_rate = dst\_host\_same\_srv\_rate,

@dst\_host\_diff\_srv\_rate = dst\_host\_diff\_srv\_rate,

@dst\_host\_same\_src\_port\_rate = dst\_host\_same\_src\_port\_rate,

@dst\_host\_srv\_diff\_host\_rate = dst\_host\_srv\_diff\_host\_rate,

@dst\_host\_serror\_rate = dst\_host\_serror\_rate,

@dst\_host\_srv\_serror\_rate = dst\_host\_srv\_serror\_rate,

@dst\_host\_rerror\_rate = dst\_host\_rerror\_rate,

@dst\_host\_srv\_rerror\_rate = dst\_host\_srv\_rerror\_rate,

@attack\_type = attack\_type,

@is\_attack = is\_attack,

@attack\_group = attack\_group

FROM MMN\_Validation\_Set

WHERE id = @ID

/\*

SELECT TOP(3)

from Euclidean distance

where service = @service

and protocolType = @protocolType

order by distance

and insert values in #Temp table

\*/

INSERT INTO #Temp(ID, isAttack, attackGroup, distance)

SELECT TOP(3) id, is\_attack, attack\_group,

SQRT(

SQUARE(duration - @duration) +

SQUARE(src\_bytes - @src\_bytes) +

SQUARE(dst\_bytes - @dst\_bytes) +

SQUARE(wrong\_fragment - @wrong\_fragment) +

SQUARE(urgent - @urgent) +

SQUARE(hot - @hot) +

SQUARE(num\_failed\_logins - @num\_failed\_logins) +

SQUARE(num\_compromised - @num\_compromised) +

SQUARE(num\_root - @num\_root) +

SQUARE(num\_file\_creations - @num\_file\_creations) +

SQUARE(num\_shells - @num\_shells) +

SQUARE(num\_access\_files - @num\_access\_files) +

SQUARE(num\_outbound\_cmds - @num\_outbound\_cmds) +

SQUARE(count - @count) +

SQUARE(srv\_count - @srv\_count) +

SQUARE(serror\_rate - @serror\_rate) +

SQUARE(srv\_serror\_rate - @srv\_serror\_rate) +

SQUARE(rerror\_rate - @rerror\_rate) +

SQUARE(srv\_rerror\_rate - @srv\_rerror\_rate) +

SQUARE(same\_srv\_rate - @same\_srv\_rate) +

SQUARE(diff\_srv\_rate - @diff\_srv\_rate) +

SQUARE(srv\_diff\_host\_rate - @srv\_diff\_host\_rate) +

SQUARE(dst\_host\_count - @dst\_host\_count) +

SQUARE(dst\_host\_srv\_count - @dst\_host\_srv\_count) +

SQUARE(dst\_host\_same\_srv\_rate - @dst\_host\_same\_srv\_rate) +

SQUARE(dst\_host\_diff\_srv\_rate - @dst\_host\_diff\_srv\_rate) +

SQUARE(dst\_host\_same\_src\_port\_rate - @dst\_host\_same\_src\_port\_rate) +

SQUARE(dst\_host\_srv\_diff\_host\_rate - @dst\_host\_srv\_diff\_host\_rate) +

SQUARE(dst\_host\_serror\_rate - @dst\_host\_serror\_rate) +

SQUARE(dst\_host\_srv\_serror\_rate - @dst\_host\_srv\_serror\_rate) +

SQUARE(dst\_host\_rerror\_rate - @dst\_host\_rerror\_rate) +

SQUARE(dst\_host\_rerror\_rate - @dst\_host\_rerror\_rate) +

SQUARE(dst\_host\_srv\_rerror\_rate - @dst\_host\_srv\_rerror\_rate) +

CASE WHEN land<>@land THEN 1 ELSE 0 END +

CASE WHEN logged\_in<>@logged\_in THEN 1 ELSE 0 END +

CASE WHEN root\_shell<>@root\_shell THEN 1 ELSE 0 END +

CASE WHEN su\_attempted<>@su\_attempted THEN 1 ELSE 0 END +

CASE WHEN is\_host\_login<>@is\_host\_login THEN 1 ELSE 0 END +

CASE WHEN is\_guest\_login<>@is\_guest\_login THEN 1 ELSE 0 END

) AS Distance\_Range

FROM MMN\_Train\_Set

WHERE (service = @service) AND (protocol\_type = @protocol\_type) AND (flag = @flag)

ORDER BY Distance\_Range

DECLARE @num\_of\_attacks INT

SET @num\_of\_attacks = 0;

/\*

insert into predictionTable values

\*/

SELECT @num\_of\_attacks = COUNT(ID)

FROM #Temp

WHERE isAttack = 1

IF @num\_of\_attacks >= 2

BEGIN

DECLARE @attackGroupDOS INT,

@attackGroupU2R INT,

@attackGroupR2L INT,

@attackGroupPROBE INT,

@attackGroup NVARCHAR(5)

SET @attackGroupDOS = (SELECT COUNT(ID) FROM #Temp WHERE attackGroup = 'dos')

SET @attackGroupU2R = (SELECT COUNT(ID) FROM #Temp WHERE attackGroup = 'u2r')

SET @attackGroupR2L = (SELECT COUNT(ID) FROM #Temp WHERE attackGroup = 'r2l')

SET @attackGroupPROBE = (SELECT COUNT(ID) FROM #Temp WHERE attackGroup = 'probe')

SET @attackGroup = ''

IF @attackGroupDOS >= 2 SET @attackGroup = 'dos'

ELSE IF @attackGroupU2R>= 2 SET @attackGroup = 'u2r'

ELSE IF @attackGroupR2L>= 2 SET @attackGroup = 'r2l'

ELSE IF @attackGroupPROBE>= 2 SET @attackGroup = 'probe'

ELSE SET @attackGroup = (SELECT TOP(1) attackGroup FROM #Temp)

INSERT INTO Result\_Table\_KNN\_Prediction3(ID, isAttack, isAttackPrediction, attackGroup, attackGroupPrediction)

VALUES (@ID, @is\_attack, 1, @attack\_group, @attackGroup)

END

ELSE

INSERT INTO Result\_Table\_KNN\_Prediction3(ID, isAttack, isAttackPrediction, attackGroup, attackGroupPrediction)

VALUES (@ID, @is\_attack, 0, @attack\_group, '')

/\*

delete from tempTable for next iteration

\*/

DELETE FROM #Temp

FETCH NEXT FROM db\_cursor INTO @ID

END

Напомена: предикција је рађена и за чворове који имају само исти сервис и протокол, и закључено је следеће:

1. Предикција без flag-а има боље резултате код погађања колоне is\_attack, када се у обзир узму сви чворови, и чворова који јесу напад, али лошије погађа уколико се у обзир узму само чворови који нису напад
2. Уколико се узме у обзир attack\_group, предикција без flag-а има лошије резултате код погађања колоне attack\_group, као и код погађања врсте напада dos, u2r, r2l, али боље погађа код врсте напад probe

Све ове разлике су јако мале ( < 1%), али пошто нас интересује бољи од ова два модела узет је модел који у обзир узима и flag, јер постиже мало боље резултате.

## 2.3 Задатак 3

Овај део задатка је исти као и задатак 2, с тим што је примењена функција косинусног ра-стојања.

WHILE @@FETCH\_STATUS = 0

BEGIN

/\*

select all variables except attack\_type

from selected row of validation set

where id = @ID

\*/

SELECT @duration = duration,

@protocol\_type = protocol\_type,

@service = service,

@flag = flag,

@src\_bytes = src\_bytes,

@dst\_bytes = dst\_bytes,

@land = land,

@wrong\_fragment = wrong\_fragment,

@urgent = urgent,

@hot = hot,

@num\_failed\_logins = num\_failed\_logins,

@logged\_in = logged\_in,

@num\_compromised = num\_compromised,

@root\_shell = root\_shell,

@su\_attempted = su\_attempted,

@num\_root = num\_root,

@num\_file\_creations = num\_file\_creations,

@num\_shells = num\_shells,

@num\_access\_files = num\_access\_files,

@num\_outbound\_cmds = num\_outbound\_cmds,

@is\_host\_login = is\_host\_login,

@is\_guest\_login = is\_guest\_login,

@count = count,

@srv\_count = srv\_count,

@serror\_rate = serror\_rate,

@srv\_serror\_rate = srv\_serror\_rate,

@rerror\_rate = rerror\_rate,

@srv\_rerror\_rate = srv\_rerror\_rate,

@same\_srv\_rate = same\_srv\_rate,

@diff\_srv\_rate = diff\_srv\_rate,

@srv\_diff\_host\_rate = srv\_diff\_host\_rate,

@dst\_host\_count = dst\_host\_count,

@dst\_host\_srv\_count = dst\_host\_srv\_count,

@dst\_host\_same\_srv\_rate = dst\_host\_same\_srv\_rate,

@dst\_host\_diff\_srv\_rate = dst\_host\_diff\_srv\_rate,

@dst\_host\_same\_src\_port\_rate = dst\_host\_same\_src\_port\_rate,

@dst\_host\_srv\_diff\_host\_rate = dst\_host\_srv\_diff\_host\_rate,

@dst\_host\_serror\_rate = dst\_host\_serror\_rate,

@dst\_host\_srv\_serror\_rate = dst\_host\_srv\_serror\_rate,

@dst\_host\_rerror\_rate = dst\_host\_rerror\_rate,

@dst\_host\_srv\_rerror\_rate = dst\_host\_srv\_rerror\_rate,

@attack\_type = attack\_type,

@is\_attack = is\_attack,

@attack\_group = attack\_group

FROM MMN\_Validation\_Set

WHERE id = @ID

/\*

SELECT TOP(3)

and insert values in #Temp table

\*/

INSERT INTO #Temp(ID, isAttack, attackGroup, distance)

SELECT TOP(3) id, is\_attack, attack\_group,

( 1 -

(((duration \* @duration) +

(src\_bytes \* @src\_bytes) +

(dst\_bytes \* @dst\_bytes) +

(wrong\_fragment \* @wrong\_fragment) +

(urgent \* @urgent) +

(hot \* @hot) +

(num\_failed\_logins \* @num\_failed\_logins) +

(num\_compromised \* @num\_compromised) +

(num\_root \* @num\_root) +

(num\_file\_creations \* @num\_file\_creations) +

(num\_shells \* @num\_shells) +

(num\_access\_files \* @num\_access\_files) +

(num\_outbound\_cmds \* @num\_outbound\_cmds) +

(count \* @count) +

(srv\_count \* @srv\_count) +

(serror\_rate \* @serror\_rate) +

(srv\_serror\_rate \* @srv\_serror\_rate) +

(rerror\_rate \* @rerror\_rate) +

(srv\_rerror\_rate \* @srv\_rerror\_rate) +

(same\_srv\_rate \* @same\_srv\_rate) +

(diff\_srv\_rate \* @diff\_srv\_rate) +

(srv\_diff\_host\_rate \* @srv\_diff\_host\_rate) +

(dst\_host\_count \* @dst\_host\_count) +

(dst\_host\_srv\_count \* @dst\_host\_srv\_count) +

(dst\_host\_same\_srv\_rate \* @dst\_host\_same\_srv\_rate) +

(dst\_host\_diff\_srv\_rate \* @dst\_host\_diff\_srv\_rate) +

(dst\_host\_same\_src\_port\_rate \* @dst\_host\_same\_src\_port\_rate) +

(dst\_host\_srv\_diff\_host\_rate \* @dst\_host\_srv\_diff\_host\_rate) +

(dst\_host\_serror\_rate \* @dst\_host\_serror\_rate) +

(dst\_host\_srv\_serror\_rate \* @dst\_host\_srv\_serror\_rate) +

(dst\_host\_rerror\_rate \* @dst\_host\_rerror\_rate) +

(dst\_host\_rerror\_rate \* @dst\_host\_rerror\_rate) +

(dst\_host\_srv\_rerror\_rate \* @dst\_host\_srv\_rerror\_rate) +

CASE WHEN land<>@land THEN 1 ELSE 0 END +

CASE WHEN logged\_in<>@logged\_in THEN 1 ELSE 0 END +

CASE WHEN root\_shell<>@root\_shell THEN 1 ELSE 0 END +

CASE WHEN su\_attempted<>@su\_attempted THEN 1 ELSE 0 END +

CASE WHEN is\_host\_login<>@is\_host\_login THEN 1 ELSE 0 END +

CASE WHEN is\_guest\_login<>@is\_guest\_login THEN 1 ELSE 0 END

) -- end of numerator

/

(SQRT(

SQUARE(duration) +

SQUARE(src\_bytes) +

SQUARE(dst\_bytes) +

SQUARE(wrong\_fragment) +

SQUARE(urgent) +

SQUARE(hot) +

SQUARE(num\_failed\_logins) +

SQUARE(num\_compromised) +

SQUARE(num\_root) +

SQUARE(num\_file\_creations) +

SQUARE(num\_shells) +

SQUARE(num\_access\_files) +

SQUARE(num\_outbound\_cmds) +

SQUARE(count) +

SQUARE(srv\_count) +

SQUARE(serror\_rate) +

SQUARE(srv\_serror\_rate) +

SQUARE(rerror\_rate) +

SQUARE(srv\_rerror\_rate) +

SQUARE(same\_srv\_rate) +

SQUARE(diff\_srv\_rate) +

SQUARE(srv\_diff\_host\_rate) +

SQUARE(dst\_host\_count) +

SQUARE(dst\_host\_srv\_count) +

SQUARE(dst\_host\_same\_srv\_rate) +

SQUARE(dst\_host\_diff\_srv\_rate) +

SQUARE(dst\_host\_same\_src\_port\_rate) +

SQUARE(dst\_host\_srv\_diff\_host\_rate) +

SQUARE(dst\_host\_serror\_rate) +

SQUARE(dst\_host\_srv\_serror\_rate) +

SQUARE(dst\_host\_rerror\_rate) +

SQUARE(dst\_host\_rerror\_rate) +

SQUARE(dst\_host\_srv\_rerror\_rate)

) \*

SQRT(

SQUARE(@duration) +

SQUARE(@src\_bytes) +

SQUARE(@dst\_bytes) +

SQUARE(@wrong\_fragment) +

SQUARE(@urgent) +

SQUARE(@hot) +

SQUARE(@num\_failed\_logins) +

SQUARE(@num\_compromised) +

SQUARE(@num\_root) +

SQUARE(@num\_file\_creations) +

SQUARE(@num\_shells) +

SQUARE(@num\_access\_files) +

SQUARE(@num\_outbound\_cmds) +

SQUARE(@count) +

SQUARE(@srv\_count) +

SQUARE(@serror\_rate) +

SQUARE(@srv\_serror\_rate) +

SQUARE(@rerror\_rate) +

SQUARE(@srv\_rerror\_rate) +

SQUARE(@same\_srv\_rate) +

SQUARE(@diff\_srv\_rate) +

SQUARE(@srv\_diff\_host\_rate) +

SQUARE(@dst\_host\_count) +

SQUARE(@dst\_host\_srv\_count) +

SQUARE(@dst\_host\_same\_srv\_rate) +

SQUARE(@dst\_host\_diff\_srv\_rate) +

SQUARE(@dst\_host\_same\_src\_port\_rate) +

SQUARE(@dst\_host\_srv\_diff\_host\_rate) +

SQUARE(@dst\_host\_serror\_rate) +

SQUARE(@dst\_host\_srv\_serror\_rate) +

SQUARE(@dst\_host\_rerror\_rate) +

SQUARE(@dst\_host\_rerror\_rate) +

SQUARE(@dst\_host\_srv\_rerror\_rate)

)

) -- end of denominator

) -- end of second member of expression

) AS Distance\_Range

FROM MMN\_Train\_Set

WHERE (service = @service) AND (protocol\_type = @protocol\_type) AND (flag = @flag)

ORDER BY Distance\_Range

DECLARE @num INT

SET @num = 0

SELECT @num = COUNT(ID) FROM #Temp

IF @num = 0

BEGIN

INSERT INTO #Temp(ID, isAttack, attackGroup)

SELECT TOP(3) id, is\_attack, attack\_group

FROM MMN\_Train\_Set

WHERE (service <> @service) OR (protocol\_type <> @protocol\_type) OR (flag <> @flag)

END

ELSE IF @num = 1

BEGIN

INSERT INTO #Temp(ID, isAttack, attackGroup)

SELECT TOP(2) id, is\_attack, attack\_group

FROM MMN\_Train\_Set

WHERE (service <> @service) OR (protocol\_type <> @protocol\_type) OR (flag <> @flag)

END

ELSE IF @num = 2

BEGIN

INSERT INTO #Temp(ID, isAttack, attackGroup)

SELECT TOP(1) id, is\_attack, attack\_group

FROM MMN\_Train\_Set

WHERE (service <> @service) OR (protocol\_type <> @protocol\_type) OR (flag <> @flag)

END

DECLARE @num\_of\_attacks INT

SET @num\_of\_attacks = 0

/\*

insert into predictionTable values

\*/

SELECT @num\_of\_attacks = COUNT(ID)

FROM #Temp

WHERE isAttack = 1

IF @num\_of\_attacks >= 2

BEGIN

DECLARE @attackGroupDOS INT,

@attackGroupU2R INT,

@attackGroupR2L INT,

@attackGroupPROBE INT,

@attackGroup NVARCHAR(5)

SET @attackGroupDOS = (SELECT COUNT(ID) FROM #Temp WHERE attackGroup = 'dos')

SET @attackGroupU2R = (SELECT COUNT(ID) FROM #Temp WHERE attackGroup = 'u2r')

SET @attackGroupR2L = (SELECT COUNT(ID) FROM #Temp WHERE attackGroup = 'r2l')

SET @attackGroupPROBE = (SELECT COUNT(ID) FROM #Temp WHERE attackGroup = 'probe')

IF @attackGroupDOS >= 2 SET @attackGroup = 'dos'

ELSE IF @attackGroupU2R>= 2 SET @attackGroup = 'u2r'

ELSE IF @attackGroupR2L>= 2 SET @attackGroup = 'r2l'

ELSE IF @attackGroupPROBE>= 2 SET @attackGroup = 'probe'

ELSE SET @attackGroup = (SELECT TOP(1) attackGroup FROM #Temp)

INSERT INTO Result\_Table\_KNN\_Prediction\_Cosine1(ID, isAttack, isAttackPrediction, attackGroup, attackGroupPrediction)

VALUES (@ID, @is\_attack, 1, @attack\_group, @attackGroup)

END

ELSE

INSERT INTO Result\_Table\_KNN\_Prediction\_Cosine1(ID, isAttack, isAttackPrediction, attackGroup, attackGroupPrediction)

VALUES (@ID, @is\_attack, 0, @attack\_group, '')

/\*

delete from tempTable for next iteration

\*/

DELETE FROM #Temp

FETCH NEXT FROM db\_cursor INTO @ID

END

Модел који даје боље резултате је предикција помоћу Еуклидског растојања, зато што предикција помоћу косинусног растојања узима у обзир три суседа који имају исти сер-вис, протокол и flag, а који су увек исти (топ 3), тако да уколико су прва три суседа са тим сервисом, протоколом и flag-ом напад или је већина напад, онда ће и нови чвор за који се врши предикција обележити као напад, без обзира што можда постоје ближи суседи који боље описују нови чвор.