Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrare de laborator Nr.4

Disciplina:Sisteme de inteligență artificială

Tema: Procesul tehnologic în învățarea automată

A efectuat: st.gr. IA-182 ,

Ulmanu Cristian

A verificat : asist.univ. Stratulat Ștefan

Chișinău 2020

**Sarcina:** Studierea și elaborarea arhitecturii unui sistem cu mecanism de învățare automată pentru rezolvarea problemei X. Este necesar de parcurs etapele de bază din procesul tehnologic de învățare supervizată.

**Cuprins:**

1. Selectarea setului de date.
2. Importarea bibliotecilor si setului de date.
3. Studierea setului de date.
4. Setarea problemei si Pregatirea setului de date pentru antrenare.
5. Initializarea modelului de regresie liniara si aplicarea acestuia pe datele noastre.
6. Concluzii.
7. Selectarea setului de date.

Setul de date pentru lucrare a fost ales de pe pagina UCI http://archive.ics.uci.edu/ml/datasets/Abalone . UCI Machine Learning Repository este o colecție de baze de date, teorii de domenii și generatoare de date care sunt utilizate de comunitatea de învățare automată pentru analiza empirică a algoritmilor de învățare automată. Arhiva a fost creată ca o arhivă ftp în 1987 de către David Aha și colegi studenți de la UC Irvine. De atunci, a fost utilizat pe scară largă de către studenți, educatori și cercetători din întreaga lume ca sursă principală de seturi de date de învățare automată. Ca o indicație a impactului arhivei, a fost citată de peste 1000 de ori, făcându-l una dintre primele 100 de „lucrări” cele mai citate din toată informatica. Versiunea actuală a site-ului web a fost proiectată în 2007 de Arthur Asuncion și David Newman, iar acest proiect este în colaborare cu Rexa.infola Universitatea din Massachusetts Amherst. Sprijinul finanțat de Fundația Națională pentru Științe este recunoscut cu recunoștință.

1. Problema.

Problema care este propusă pentru rezolvarea prezintă: Crearea modelului capabil să prezică vârsta moluștei Abalon fără numărarea inelelor ei.

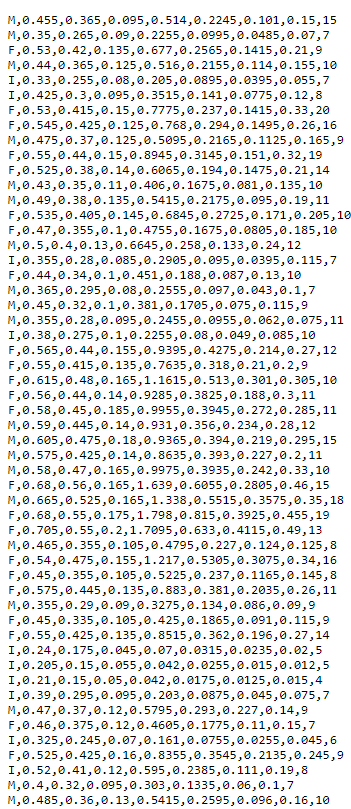
1. Studierea bibliotecilor ***numpy*** și ***pandas***.

Ideea fundamentală a NumPy este suportul pentru rețelele multidimensionale. Deci, NumPy poate fi considerat ca bază pentru calculul numeric în Python și a fost creat pentru a permite Python-ului să fie folosit în rezolvarea problemelor matematice și științifice. Modulul NumPy ne oferă sute de funcții matematice utile în plus față de constante cum ar fi baza logaritmilor naturali, pi.

Pandas este o bibliotecă software în Python pentru prelucrarea și analiza datelor. Manipularea datelor pandas este construită deasupra bibliotecii NumPy, care este un instrument de nivel inferior. Oferă structuri și operații de date speciale pentru manipularea tabelelor numerice și a seriilor temporale.

1. Studierea setului de date.

Setut de date prezintă un document csv care conține seria de date pentru procesare. Un fișier CSV (valori separate prin virgulă) este un tip special de fișier pe care îl puteți crea sau edita în Excel. În loc să stocați informații în coloane, fișierele CSV stochează informațiile separate prin virgule. Atunci când textul și numerele sunt salvate într-un fișier CSV, este simplu să le mutați de la un program la altul. De exemplu, puteți să exportați persoanele de contact din Google într-un fișier CSV, apoi să le importați în Outlook.



1. Procesarea setului de date.

% **matplotlib** inline

**import**  **matplotlib.pyplot**  **as**  **plt**

**import**  **numpy**  **as**  **np**

**import**  **panda**  **as**  **pd**

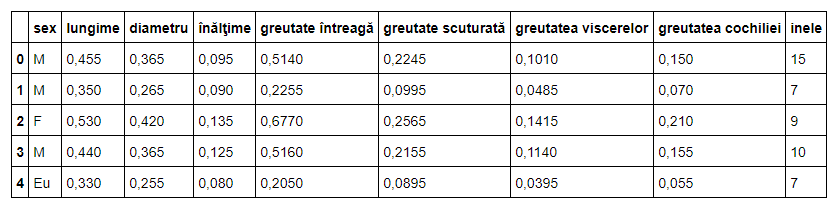
nume\_coloană = [ "sex" , "lungime" , "diametru" , "înălțime" , "greutate întreagă" ,

"greutate scuturată" , "greutate viscere" , "greutate coajă" , "inele" ]

date = pd . read\_csv ( "abalone.data" , names = nume\_coloana )

print ( "Număr de eșantioane:% d" % len ( date ))

date .

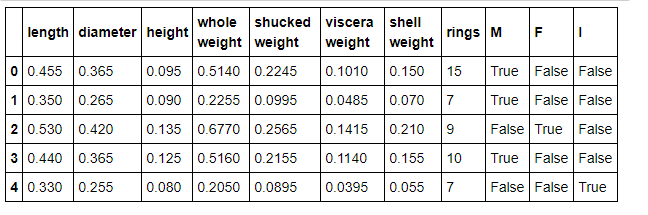


**for** label **in** "MFI":

data[label] = data["sex"] == label

**del** data["sex"]

data.head()



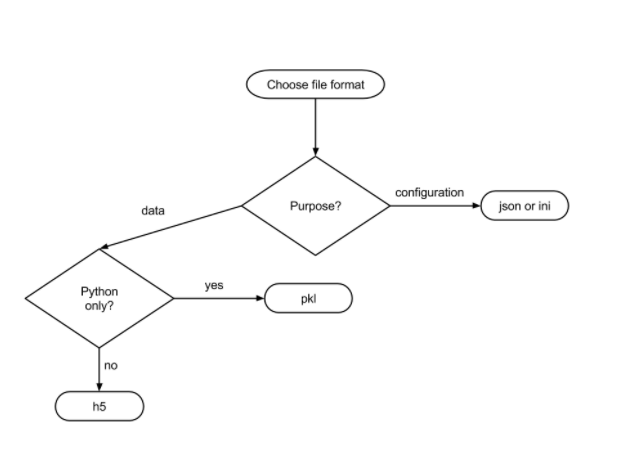
y = data.rings.values

**del** data["rings"] *# remove rings from data, so we can convert all the dataframe to a numpy 2D array.* X = data.values.astype(np.float)

Împărțiți datele în seturile de testare și testare

**din**  **sklearn**  **import**  cross\_validation

train\_X , test\_X , train\_y , test\_y = cross\_validation . train\_test\_split ( X , y ) *# se împarte în mod implicit 75% / 25%*



**from** **sklearn.tree** **import** DecisionTreeRegressor *# create an estimator, optionally specifying parameters* model = DecisionTreeRegressor() *# fit the estimator to the data* model.fit(train\_X, train\_y) *# apply the model to the test and training data* predicted\_test\_y = model.predict(test\_X) predicted\_train\_y = model.predict(train\_X)

**def** scatter\_y(true\_y, predicted\_y):

*"""Scatter-plot the predicted vs true number of rings*

*Plots:*

*\* predicted vs true number of rings*

*\* perfect agreement line*

*\* +2/-2 number dotted lines*

*Returns the root mean square of the error*

*"""*

fig, ax = plt.subplots(figsize=(8, 8))

ax.plot(true\_y, predicted\_y, '.k')

ax.plot([0, 30], [0, 30], '--k')

ax.plot([0, 30], [2, 32], ':k')

ax.plot([2, 32], [0, 30], ':k')

rms = (true\_y - predicted\_y).std()

ax.text(25, 3,

"Root Mean Square Error = %.2g" % rms,

ha='right', va='bottom')

ax.set\_xlim(0, 30)

ax.set\_ylim(0, 30)

ax.set\_xlabel('True number of rings')

ax.set\_ylabel('Predicted number of rings')

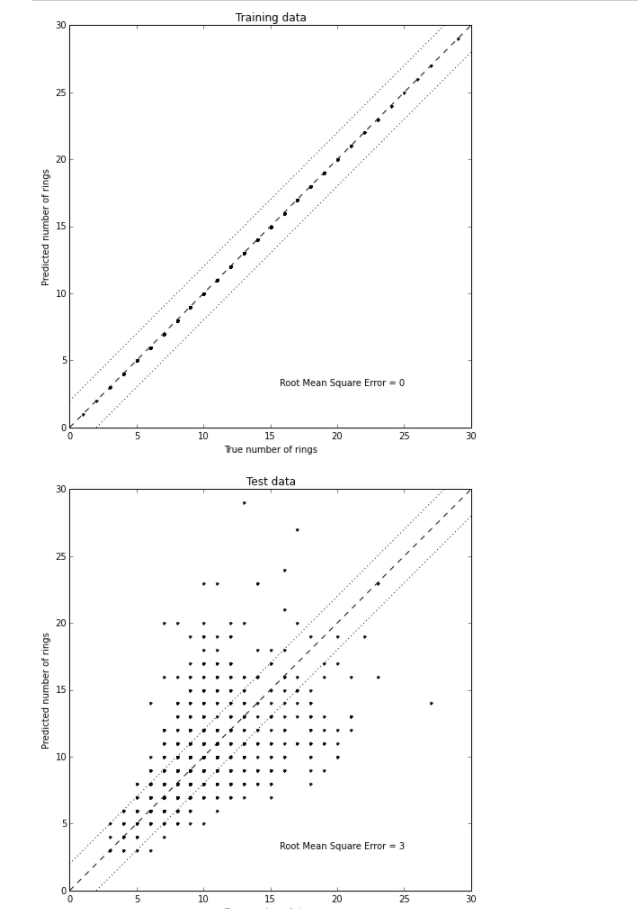
**return** rms

scatter\_y(train\_y, predicted\_train\_y)

plt.title("Training data")

scatter\_y(test\_y, predicted\_test\_y)

plt.title("Test data");



model = DecisionTreeRegressor(max\_depth=10)

*# fit the estimator to the data*

model.fit(train\_X, train\_y)

*# apply the model to the test and train data*

predicted\_test\_y = model.predict(test\_X)

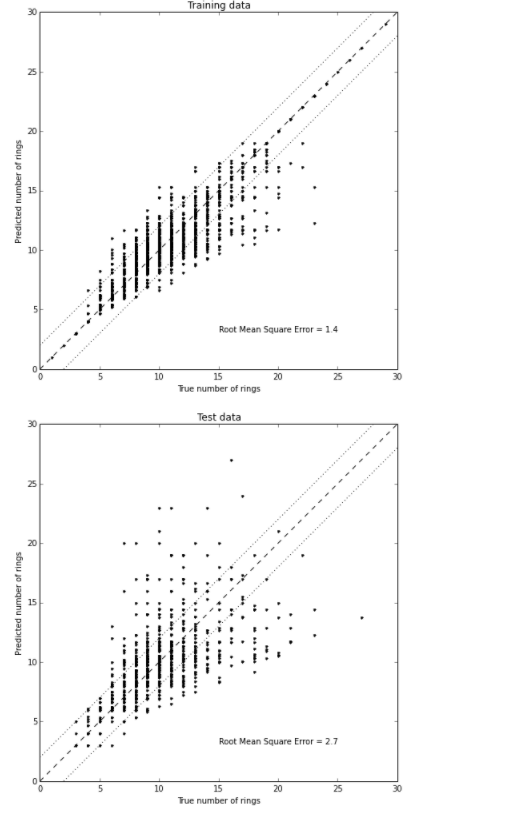
predicted\_train\_y = model.predict(train\_X)

scatter\_y(train\_y, predicted\_train\_y)

plt.title("Training data")

rms\_decision\_tree = scatter\_y(test\_y, predicted\_test\_y)

plt.title("Test data");



Trasăm curbele de învățare

data\_percentage\_array = np . linspace ( 10 , 100 , 10 )

train\_error = []

test\_error = []

**pentru**  data\_percentage **în**  data\_percentage\_array :

model = DecisionTreeRegressor ( max\_depth = 10 )

number\_of\_samples = int ( data\_percentage / 100. \* len ( train\_y ))

model . fit ( train\_X [: number\_of\_samples ,: ], train\_y [: number\_of\_samples ])

predicted\_train\_y = model . predict ( train\_X )

predicted\_test\_y = model . prezice ( test\_X )

tren\_error . append (( predicted\_train\_y - train\_y ) . std ())

test\_error . append (( predicted\_test\_y - test\_y ) . std ())

plt . plot ( data\_percentage\_array , train\_error , label = 'training' )

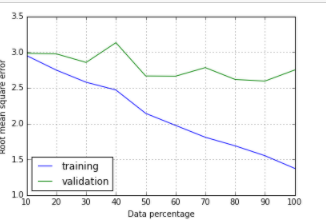
plt . plot ( data\_percentage\_array , test\_error , label = 'validare' )

plt . grilă ()

plt . legenda ( loc = 3 )

plt . xlabel ( „Procent de date” )

plt . ylabel ( „Rădăcina înseamnă eroare pătrată” );



1. Inițializarea modelului de învățare și implimentarea acestuia.

**din**  **sklearn.ensemble**  **import**  RandomForestRegressor

model = RandomForestRegressor ( n\_estimators = 5 )

model . fit ( train\_X , train\_y )

predicted\_test\_y = model . predict ( test\_X )

rms\_random\_forest = scatter\_y ( test\_y , predicted\_test\_y )

print("Root Mean Square error decreases from %.2g to %.2g." % (rms\_decision\_tree, rms\_random\_forest))

Optimizăm parametrii modelului

model = RandomForestRegressor ( n\_estimators = 100 )

n\_features = X . forma [ 1 ]

**from** **sklearn.grid\_search** **import** RandomizedSearchCV

grid = RandomizedSearchCV(model, n\_iter=20,

param\_distributions=dict(

max\_depth=np.arange(5,20+1),

max\_features=np.arange(1, n\_features+1)

)

)

grid.fit(X, y)

print(grid.best\_params\_)

model = RandomForestRegressor(max\_features=grid.best\_params\_["max\_features"], max\_depth=grid.best\_params\_["max\_depth"]) model.fit(train\_X, train\_y) predicted\_test\_y = model.predict(test\_X) rms\_optimized\_random\_forest = scatter\_y(test\_y, predicted\_test\_y

print("Root Mean Square error decreases from %.2g to %.2g." % (rms\_random\_forest, rms\_optimized\_random\_forest))

**Concluzii:**

În urma efectuării acestei lucrări de laborator am făcut cunoștință cu bibliotecile Pandas, sklearn și NumPy care sunt utilizate foarte des pentru rezolvarea problemelor în așa tip de învățare. Am aflat cum se încarcă datele în aplicație, de unde le putem lua, cum să afișăm datele preluate sau să le selectăm.

Deasemenea am înțeles cum se fac seturi de teste și cum merge în realitate învățarea automată și faptul că modelul inițial trebuie deseori optimizat pentru a micșora eroarea aplicației.