Department of Computer Science
Technical University of Cluj-Napoca

Intelligent Systems

Laboratory activity 2022-2023

Machine Learning
Project title: Chess Prediction

Name:Polman Marian, Cezar Zbughin
Group:30232
Email:marianpolman2000.mp@gmail.com, cezarzbughin32gmail.com

Course: Adrian Groza
Adrian.Groza@cs.utcluj.ro

Contents

Chapter 1

Descrierea proiectului

1.1 Introducere

Proiectul nostru se concentrează pe dezvoltarea unui Decision Tree (Arbore de Decizie) bazat pe un set de date care conține informații despre parcursul jocurilor de șah. Scopul sistemului nostru este să analizeze diferite caracteristici ale jocului, cum ar fi rating-ul jucătorilor, diferența de rating, deschiderea utilizată și numărul de mutări, și să prezică câștigătorul partidei respective.

Pentru a realiza acest lucru, vom utiliza un set de date colectat și prelucrat anterior, care conține informații detaliate despre jocuri de șah, inclusiv rating-ul jucătorilor implicați, deschiderea utilizată și numărul de mutări realizate. Vom utiliza aceste informații pentru a construi un arbore de decizie, un model bazat pe reguli, care va putea să facă predicții cu privire la câștigătorul partidelor de șah, pe baza caracteristicilor specifice ale fiecărei partide.

Prin utilizarea unui Decision Tree, vom explora relațiile dintre caracteristicile jocului de șah și rezultatul acestuia, identificând astfel pattern-uri și reguli care să ne ajute să facem predicții corecte. Arborele de decizie va fi antrenat pe setul nostru de date și va fi capabil să clasifice noi jocuri de șah în funcție de probabilitatea de câștig a fiecărui jucător.

Proiectul nostru reprezintă o oportunitate de a explora abordările de inteligență artificială în contextul jocurilor de strategie și de a dezvolta o soluție practică pentru a face predicții în acest domeniu. De asemenea, vom evalua performanța sistemului nostru și vom discuta posibile îmbunătățiri și extinderi ale acestuia.

1.2 Dataset

Cu ajutorul platformei Kaggle, avem acces la un dataset cuprinzător ce conține informații despre jocurile de șah online. Datasetul furnizează detalii referitoare la numărul de mutări, deschiderile utilizate și rezultatul fiecărei partide. Aceste informații oferă oportunități semnificative pentru analiza și înțelegerea jocului de șah, unul complex și apreciat de-a lungul timpului.

Indiferent de nivelul de experiență în jocul de șah sau de interesul în analiza datelor, acest dataset reprezintă o resursă valoroasă. Prin explorarea acestor informații, putem descoperi relații și modele între deschiderile utilizate și rezultatele obținute, aducând astfel beneficii în îmbunătățirea strategiilor și în creșterea performanței în jocul de șah.

Având la dispoziție acest set de date, avem posibilitatea de a explora și de a învăța mai multe despre dinamica jocului de șah și despre factorii care contribuie la rezultatele obținute în partidele online. Această abordare ne permite să obținem insights și să dezvoltăm metode mai eficiente de analiză și luare a deciziilor în cadrul acestui joc.

Astfel, datasetul de jocuri de șah de pe platforma Kaggle ne oferă o sursă bogată de informații pentru a analiza și înțelege mai bine acest joc fascinant. Prin explorarea datelor și prin utilizarea unui Decision Tree, putem dezvolta un sistem inteligent care să facă predicții cu privire la câștigătorul unei partide de șah pe baza caracteristicilor specifice ale fiecărei partide.

Chapter 2

Implementation details

2.1 Librarii

Pentru a implementa cu succes proiectul, am apelat la mai multe librării Python care ne-au oferit instrumentele necesare. Printre acestea se numără:

Pandas: Această librărie ne-a permis să manipulăm și să analizăm datele într-un mod eficient. Am putut citi setul de date și am extras informațiile relevante pentru construirea Decision Tree-ului.

NumPy: Am utilizat NumPy pentru operații numerice și lucrul cu matrice și vectori. Prin intermediul său, am putut pregăti caracteristicile și rezultatele din setul de date pentru antrenarea și testarea Decision Tree-ului.

Graphviz: Pentru a obține o reprezentare vizuală a Decision Tree-ului rezultat, am folosit biblioteca Graphviz. Aceasta ne-a oferit posibilitatea de a genera o diagramă grafică a structurii și deciziilor luate de arborelev de decizie.

2.2 Pasi implementare

În primul rând, am încărcat setul de date folosind biblioteca Pandas, utilizând funcția **read_csv**. Astfel, am creat un obiect DataFrame care conține informațiile din fișierul CSV specificat. De exemplu, în codul nostru, am citit fișierul "chess_games_2.csv" și am stocat datele în obiectul DataFrame numit **data**.

Pentru a putea utiliza datele în algoritmul Decision Tree, am trebuit să encodăm variabilele categorice. Pentru acest scop, am utilizat modulul **LabelEncoder** din biblioteca Scikit-learn. Am aplicat encodarea asupra coloanei "opening_code" din setul de date, transformând-o într-o formă numerică. De asemenea, am aplicat encodarea și pentru celelalte atribute din setul de date. Codul relevat este următorul: **le.fit(data['opening_code'])** și **data = data.apply(le.fit_transform)**.

Următorul pas a constat în împărțirea setului de date într-un set de antrenare și un set de testare. Pentru acest lucru, am utilizat funcția **train_test_split** din biblioteca Scikit-learn. Am specificat procentul de date alocat pentru setul de testare (în acest caz, 20%) și am setat și o valoare pentru generarea unor rezultate reproductibile. La final, am obținut patru variabile: **X_train, X_test, Y_train, Y_test**. Codul corespunzător este: **X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=100)**.

După pregătirea datelor, am construit arborele de decizie utilizând clasa **DecisionTreeClassifier** din biblioteca Scikit-learn. Am inițializat modelul și am folosit metoda **fit** pentru a ajusta

arborele de decizie la datele de antrenare. În codul nostru, am definit modelul astfel: **model = tree.DecisionTreeClassifier(criterion="entropy")**, iar apoi l-am antrenat cu datele de antrenare folosind **model.fit(X_train, Y_train)**.

Acești pași de implementare ne-au adus mai aproape de obținerea unui model de arbore de decizie pregătit pentru a efectua predicții pe datele de testare.

Chapter 3

Analiza rezultatelor

Pentru a evalua acuratețea modelului, am utilizat metrica "Accuracy" (acuratețe), care reprezintă proporția corectă de predicții în raport cu numărul total de instanțe. În cazul nostru, acuratețea obținută a fost de aproximativ 76.84%, ceea ce indică o performanță bună a modelului.

De asemenea, am generat matricea de confuzie, care ne oferă o imagine mai detaliată a performanței modelului pe fiecare clasă. Matricea de confuzie ne permite să vizualizăm numărul de predicții corecte și eronate pentru fiecare clasă. În rezultatele noastre, putem observa că modelul a avut o acuratețe ridicată pentru clasele 0 și 2, dar a avut o performanță mai scăzută pentru clasa 1.

Pentru a interpreta arborele de decizie generat, am vizualizat grafic structura arborelui utilizând biblioteca Graphviz. Acesta ne oferă o reprezentare vizuală a deciziilor luate de model și a ramificațiilor în funcție de diferitele atribute și valori. Prin examinarea arborelui de decizie, putem înțelege cum sunt luate deciziile și care sunt criteriile utilizate pentru predicții.

Prin urmare, evaluarea și interpretarea rezultatelor ne oferă o înțelegere mai clară a performanței modelului nostru de arbore de decizie. Acuratețea înaltă obținută și analiza matricei de confuzie ne arată că modelul are o capacitate bună de a prezice rezultatele jocurilor de șah pe baza datelor de intrare. Prin examinarea arborelui de decizie, putem obține și o înțelegere mai profundă a logicii și a factorilor importanți care influențează rezultatele jocului de șah.

În cadrul implementării noastre, am efectuat mai multe manipulări asupra setului de date pentru a îmbunătăți performanța modelului nostru de arbore de decizie. Am luat în considerare atât eliminarea anumitor coloane, cât și adăugarea altora pentru a obține rezultate mai precise.

Pentru a reduce potențialele interferențe ale unor atribute asupra procesului de luare a deciziilor, am eliminat coloanele care ar putea aduce zgomot sau să distorsioneze modelul. Am făcut acest lucru pe baza expertizei noastre și a cunoștințelor despre domeniul de aplicare al jocului de șah.

În același timp, am adăugat o nouă coloană numită "rating_diff", care reprezintă diferența de rang între cei doi jucători implicați într-un joc de șah. Am considerat că această diferență poate fi un factor semnificativ în determinarea rezultatului jocului și poate contribui la creșterea preciziei modelului nostru.

Astfel, prin manipularea setului de date și aplicarea acestor modificări, am reușit să obținem rezultate mai bune și să creștem performanța modelului nostru de arbore de decizie în predicția rezultatelor jocurilor de șah.

Chapter 4

Concluzie

În concluzie, experiența de implementare a acestui proiect a fost una valoroasă și plină de învățăminte. Procesul de construire a unui model de arbore de decizie pentru predicția rezultatelor jocurilor de șah ne-a permis să explorăm și să aplicăm diverse concepte și tehnici în domeniul machine learning.

Am avut oportunitatea de a utiliza biblioteci și module precum Pandas, NumPy, scikit-learn și Graphviz, care ne-au facilitat procesul de prelucrare a datelor, construire a modelului și vizualizare a rezultatelor. Înțelegerea și utilizarea acestor instrumente ne-au adus o mai mare încredere și familiaritate în lucrul cu proiecte de analiză și predicție.

De asemenea, manipularea setului de date, eliminarea și adăugarea de coloane ne-a adus o perspectivă mai detaliată asupra importanței atributelor în cadrul procesului de luare a deciziilor. Am învățat să evaluăm și să selectăm atributele relevante pentru a îmbunătăți performanța modelului nostru.

În ansamblu, acest proiect ne-a oferit oportunitatea de a aplica cunoștințele teoretice într-un context practic și de a explora modul în care algoritmul de arbore de decizie poate fi utilizat în rezolvarea problemelor de predicție. A fost o experiență captivantă și motivantă, care ne-a deschis noi orizonturi și ne-a adus o înțelegere mai profundă a domeniului machine learning.

```
import pandas as pd
import numpy as np
import graphviz
import os
import subprocess
from sklearn.tree import DecisionTreeClassifier, export_graphviz
#Citire data set
input_file = "chess_games_2.csv"
data = pd.read_csv(input_file, header = 0)
data_rows, data_cols = data.shape
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit(data['opening_code']) # only for one attribute
data_encoded = le.transform(data["opening_code"])

data_encoded

data = data.apply(le.fit_transform) # for all the attributes
array = data.values

#Features
X = array[:,0:data_cols-1]

#Target
Y = array[:,data_cols-1]
```

```
27
28      #Data exploration
29      import seaborn
30      import matplotlib.pyplot as plt
31
32      #seaborn.pairplot(data, hue = 'winner')
33      #plt.show()
34      #Data exploration
35      #1=nr bancnote falsificate
36      #0=nr bancnote autentice
37
38      from sklearn import tree
39      model = tree.DecisionTreeClassifier(criterion="entropy")
40      #Impartire dataset: train + test
41      #Fit
42      from sklearn.model_selection import train_test_split
43      X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =
    0.20, random_state = 100)
44      model.fit(X_train,Y_train)
45      DecisionTreeClassifier(criterion='entropy')
46      #Prezicere raspuns pentru dataset
47      #Predict
48      Y_prediction = model.predict(X_test)
49      print("Y: ",Y)
50      print("Y_prediction:",Y_prediction)
51      print("Y_test:",Y_test)
52      succ = 0;
53      fail = 0;
54      for i in range(0, len(Y_test)):
55          if Y_prediction[i] == Y_test[i]:
56              succ = succ + 1;
57          else:
58              fail = fail + 1;
59
60      print(succ / (succ+fail) * 100);
61      Y:  [2 0 2 ... 2 2 0]
62      Y_prediction: [2 2 0 ... 0 0 2]
63      Y_test: [2 2 0 ... 2 0 2]
64      76.8444666001994
65      import graphviz
66      Z= data[data.columns.drop('winner')]
67      dot_data = tree.export_graphviz(model, out_file = None, feature_names= Z
    .columns)
68      graph1 = graphviz.Source(dot_data)
69      graph1
70
71      #5 metrici pentru clasificatorul initial
72      from sklearn import metrics
73      from sklearn.metrics import confusion_matrix
74      from sklearn.metrics import recall_score
75      from sklearn.metrics import precision_score
76      from sklearn.metrics import f1_score
77
78
79      print("Accuracy:",metrics.accuracy_score(Y_test, Y_prediction))
80
81      confusion_mat = confusion_matrix(Y_test, Y_prediction)
82      print("Confusion matrix is : ", confusion_mat)
83      Accuracy: 0.768444666001994
84      Confusion matrix is :  [[1405    4  456]
```

```
    [   4  149    5]
    [ 455    5 1529]]
from sklearn.model_selection import cross_val_score
cross_val_score(model, X_train,Y_train,cv=5)
array([0.72523364, 0.67279526, 0.73106887, 0.74197569, 0.75070115])
#GridSearch pentru imbunatatire rezultate-cautarea celor mai bune valori pentru parametri
from sklearn import datasets
from sklearn.model_selection import GridSearchCV
parameters = {'criterion':('gini', 'entropy'),'splitter':('best', 'random') ,'max_depth':range(1,21),'min_samples_split':range(2,11)}
model3 = tree.DecisionTreeClassifier()
clf = GridSearchCV(model3, parameters)
clf.fit(X, Y)
clf.best_params_
clf.cv_results_
Schimbare parametrii pentru a imbunatati performanta
from sklearn.metrics import f1_score
#Schimbare parametrii pentru a imbunatati performanta
#Valorile pentru parametrii sunt cei gasiti de GridSearch

clf.fit(X_train,Y_train)
Y_prediction = clf.predict(X_test)
print("Y: ",Y)
print("Y_prediction:",Y_prediction)
print("Y_test:",Y_test)
print("Metricile dupa setarea parametrilor optimi:")
print("Accuracy:",metrics.accuracy_score(Y_test, Y_prediction))

confusion_mat = confusion_matrix(Y_test, Y_prediction)
print("Confusion matrix is : ", confusion_mat)

recall_show = recall_score(Y_test, Y_prediction)
print("Recall_score is :",recall_show)

precision_show = precision_score(Y_test, Y_prediction)
print("Precision Score is : ", precision_show)

f1_score = f1_score(Y_test, Y_prediction)
print("f1 Score is : ",f1_score)
#Cross-validation pentru modelul cu parametrii gasiti de GridSearch
from sklearn.model_selection import cross_val_score
cross_val_score(clf, X,Y,cv=2)
Vizualizare rezultate obtinute cu parametri diferiti
#Vizualizare rezultate obtinute cu parametri diferiti
%matplotlib inline
import matplotlib.pyplot as plt
scores = {}
scores_list = []
#Parametrul max_depth variaza intre 1-10
maxdepth=range(1,10)

for k in maxdepth:
    classifier = tree.DecisionTreeClassifier(max_depth=k)
    classifier.fit(X_train, Y_train)
    y_pred = classifier.predict(X_test)
    scores[k] = metrics.accuracy_score(Y_test,y_pred)
    scores_list.append(metrics.accuracy_score(Y_test,y_pred))

plt.plot(maxdepth,scores_list)
```

```
    plt.xlabel("Depth of the decision tree")
    plt.ylabel("Accuracy")
    %matplotlib inline
    import matplotlib.pyplot as plt
    scores = {}
    scores_list = []
    #Parametrul max_depth variaza intre 2-6
    criterion=["gini","entropy"]

    for k in criterion:
        classifier = tree.DecisionTreeClassifier(criterion=k)
        classifier.fit(X_train, Y_train)
        y_pred = classifier.predict(X_test)
        scores[k] = metrics.accuracy_score(Y_test,y_pred)
        scores_list.append(metrics.accuracy_score(Y_test,y_pred))

    plt.bar(criterion,scores_list,width=0.2,color='blue')
    plt.xlabel("Criterion")
    plt.ylabel("Accuracy")
    plt.ylim((0.97,1))
    plt.show()

    data.winner.value_counts().plot(kind="bar",color="blue",title="winning")
```

Bibliography

- https://moodle.cs.utcluj.ro/course/view.php?id=542
- https://www.kaggle.com/
- https://scikit-learn.org/
- https://tutorialspoint.com/
- https://geeksforgeeks.org/