Міністерство освіти і науки України

Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Програмування інтелектуальних інформаційних систем

**ЗВІТ**

До лабораторної роботи №4.

Тема: Прогнозування

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | ІТ-92, Бондаренко Дмитро Сергійович |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | ас. Баришич Л.М. |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2021

# **Завдання лабораторної роботи**

Завданням цієї лабораторної роботи є виконання передбачення на основі створених даних про подальші результати гри. Для цього потрібно:

* Обрати залежну змінну.
* Привести датасет до виду, що можна використати для регресії.
* Розбити датасет на тренувальну та контрольну частини.
* Натренувати модель.

Прогнозування буде виконуватися на основі лінійної регресії.

# **Опис використаних технологій**

У цій роботі буде використана мова програмування Python та LISP, оскільки ці мови гарно підходять для такого роду задач.

# **Опис програмного коду**

За залежну зміну буде обрана кількість очок. За незалежну зміну буде обраний час гри. За допомогою мови LISP ми розділимо наші дані на тестові та навчальні вибірки з який 75% - навчальні дані, та 25% - тренувальні відповідно:

(require "cl-csv")

(setq elems (cl-csv:read-csv #P"stats.csv"))

(setq header (NTH 0 elems))

(setq expectimax\_list ())

(setq expectimax\_test\_list ())

(setq expectimax\_train\_list ())

(setq alphabeta\_list ())

(setq alphabeta\_test\_list ())

(setq alphabeta\_train\_list ())

(defun nshuffle (sequence)

  (loop for i from (length sequence) downto 2

        do (rotatef (elt sequence (random i))

                    (elt sequence (1- i))))

  sequence)

(loop for a in elems

   do (if (string-equal (NTH 0 a) "ExpectimaxAgent")

             (push a expectimax\_list)))

(loop for a in elems

   do (if (string-equal (NTH 0 a) "AlphaBetaAgent")

             (push a alphabeta\_list)))

(nshuffle expectimax\_list)

(nshuffle alphabeta\_list)

(setq expectimax\_list\_length (length expectimax\_list))

(setq alphabeta\_list\_length (length alphabeta\_list))

(loop for a from 0 to (/ (- expectimax\_list\_length 1) 4)

   do (push (NTH a expectimax\_list) expectimax\_test\_list)

)

(loop for a from (+ (/ (- expectimax\_list\_length 1) 4) 1) to (- expectimax\_list\_length 1)

   do (push (NTH a expectimax\_list) expectimax\_train\_list)

)

(loop for a from 0 to (/ (- alphabeta\_list\_length 1) 4)

   do (push (NTH a alphabeta\_list) alphabeta\_test\_list)

)

(loop for a from (+ (/ (- alphabeta\_list\_length 1) 4) 1) to (- alphabeta\_list\_length 1)

   do (push (NTH a alphabeta\_list) alphabeta\_train\_list)

)

(push header alphabeta\_train\_list)

(push header alphabeta\_test\_list)

(push header expectimax\_train\_list)

(push header expectimax\_test\_list)

(with-open-file (stream "stats\_expectimax\_test.csv"

                     :direction :output

                     :if-exists :supersede

                     :if-does-not-exist :create)

                     (cl-csv:write-csv expectimax\_test\_list :stream stream))

(with-open-file (stream "stats\_expectimax\_train.csv"

                     :direction :output

                     :if-exists :supersede

                     :if-does-not-exist :create)

                     (cl-csv:write-csv expectimax\_train\_list :stream stream))

(with-open-file (stream "stats\_alphabeta\_train.csv"

                     :direction :output

                     :if-exists :supersede

                     :if-does-not-exist :create)

                     (cl-csv:write-csv alphabeta\_train\_list :stream stream))

(with-open-file (stream "stats\_alphabeta\_test.csv"

                     :direction :output

                     :if-exists :supersede

                     :if-does-not-exist :create)

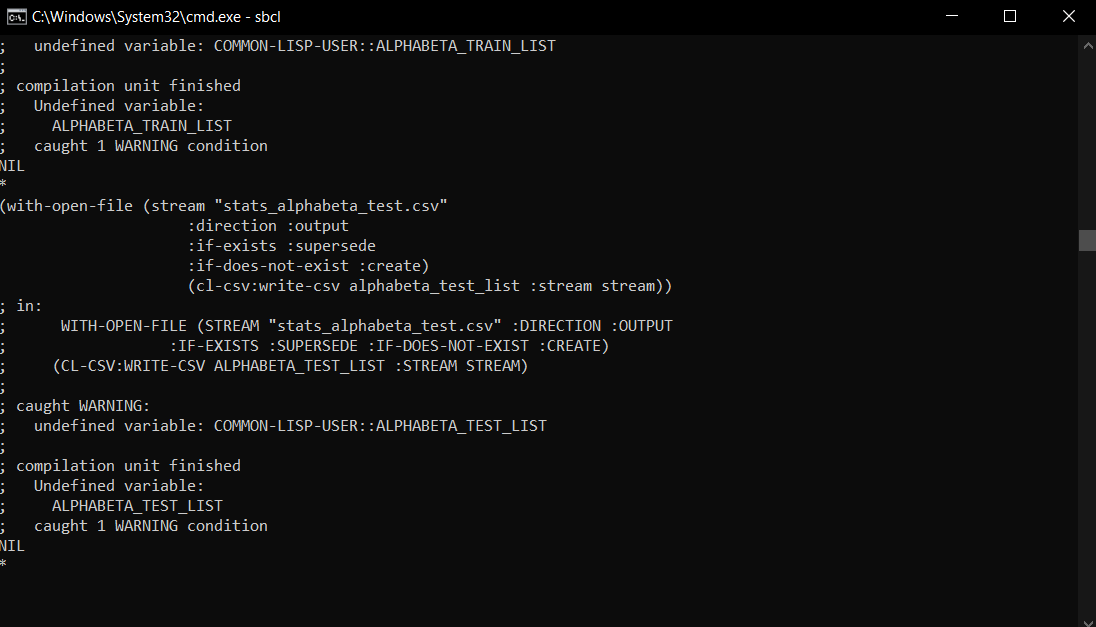
                     (cl-csv:write-csv alphabeta\_test\_list :stream stream))

Прогнозування та тренування моделі ми будемо робити використовуючи мову Python:

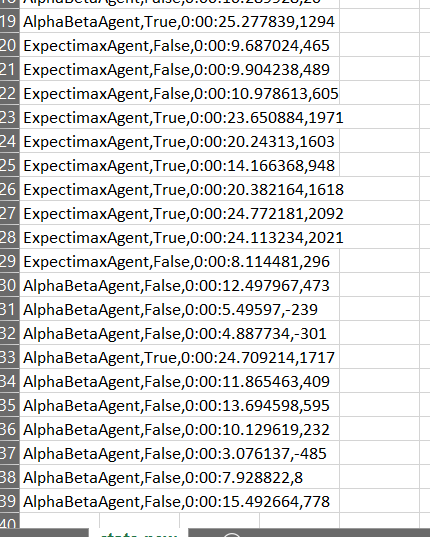
import pandas as pd  
import numpy as np  
from numpy import random  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import r2\_score  
  
  
# x - time  
# y - points  
  
def forecast(test, train, file\_to\_continue):  
 df\_new\_csv = pd.read\_csv(file\_to\_continue)  
 df\_test = pd.read\_csv(test)  
 df\_train = pd.read\_csv(train)  
 y\_train = df\_train['Points'].tolist()  
 x\_train = list()  
 times\_train = df\_train['GameTime'].tolist()  
 x\_test\_values = list()  
 y\_test\_values = df\_test['Points'].tolist()  
 times\_test = df\_test['GameTime'].tolist()  
 for time in times\_train:  
 x\_train.append(float(time.replace("0:00:", "")))  
 for time in times\_test:  
 x\_test\_values.append(float(time.replace("0:00:", "")))  
 x\_test = np.array(x\_test\_values).reshape((-1, 1))  
 x\_train = np.array(x\_train).reshape((-1, 1))  
 model = LinearRegression()  
 model.fit(x\_train, y\_train)  
 test = model.predict(x\_test)  
 print("Score:" + str(r2\_score(y\_test\_values, test)))  
 min\_value\_x = df\_test['GameTime'].min()  
 max\_value\_x = df\_test['GameTime'].max()  
 min\_value\_x = float(min\_value\_x.replace("0:00:", ""))  
 max\_value\_x = float(max\_value\_x.replace("0:00:", ""))  
 random\_times = list()  
 for i in range(10):  
 random\_time = round(random.uniform(max\_value\_x, min\_value\_x), 6)  
 random\_times.append(random\_time)  
 random\_times\_array = np.array(random\_times).reshape((-1, 1))  
 predicts = model.predict(random\_times\_array).tolist()  
 for i in range(len(predicts)):  
 is\_win = True  
 if predicts[i] < 900:  
 is\_win = False  
 print("{},{},{},{}".format(df\_test.values[0][0],  
 is\_win, "0:00:" + str(random\_times[i]), int(predicts[i])))  
 entity = pd.DataFrame({"AlgorithmAgent": [df\_test.values[0][0]],  
 "IsWon": [is\_win],  
 "GameTime": ["0:00:" + str(random\_times[i])],  
 "Points": [int(predicts[i])]})  
 df\_new\_csv = df\_new\_csv.append(entity)  
 df\_new\_csv.to\_csv('stats\_new.csv', index=False)  
  
  
def main():  
 forecast(r'stats\_expectimax\_test.csv', r'stats\_expectimax\_train.csv', r'stats.csv')  
 forecast(r'stats\_alphabeta\_test.csv', r'stats\_alphabeta\_train.csv', r'stats\_new.csv')  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

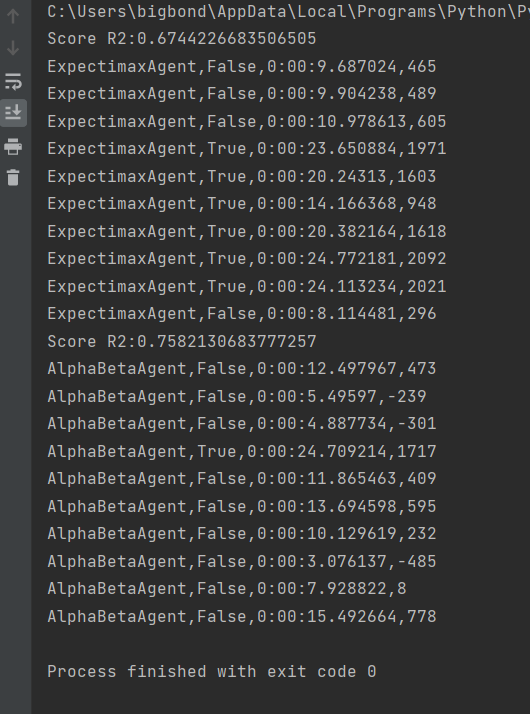
Результатом виконання цього коду буде файл із 10 спрогнозованними результатами для ExpectiMax та 10 спрогнозованними результатами гри для AlphaBeta.

# **Скріншоти роботи програмного застосунку**



Робота коду LISP





Результат виконання коду на Python.

Для двох алгоритмів був вирахуваний коефіціент кореляції R2. Так, для ExpectiMax це коефіцієнт дорівнює 0.67, а для AlphaBeta – 0.75, тобто, зв’язок між часом та очками у моїй реалізації цих алгоритмів доволі висока. Оскільки для вирахування коефіцієнтів використовувалася тестові вибірки, можна зробити висновок, що моделі натреновані якісно.

**Висновок:** У цій роботі ми навчилися прогнозувати дані з використанням мов програмування Python та Lisp. Ми вивчили, що таке лінійна регресія, як і де її можна застосувати. Навчилися будувати та тренувати моделі для прогнозування. В результаті роботи, ми спрогнозували результат 20 ігор та записали ці значення у файл.