Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни
«Прикладні задачі машинного навчання»

«Прикладна задача машинного навчання»

Варіант 7

Виконав студент: ІП-12 Васильєв Єгор Костянтинович

Перевірив: Нестерук Андрій Олександрович

Київ 2023

Лабораторна робота №6

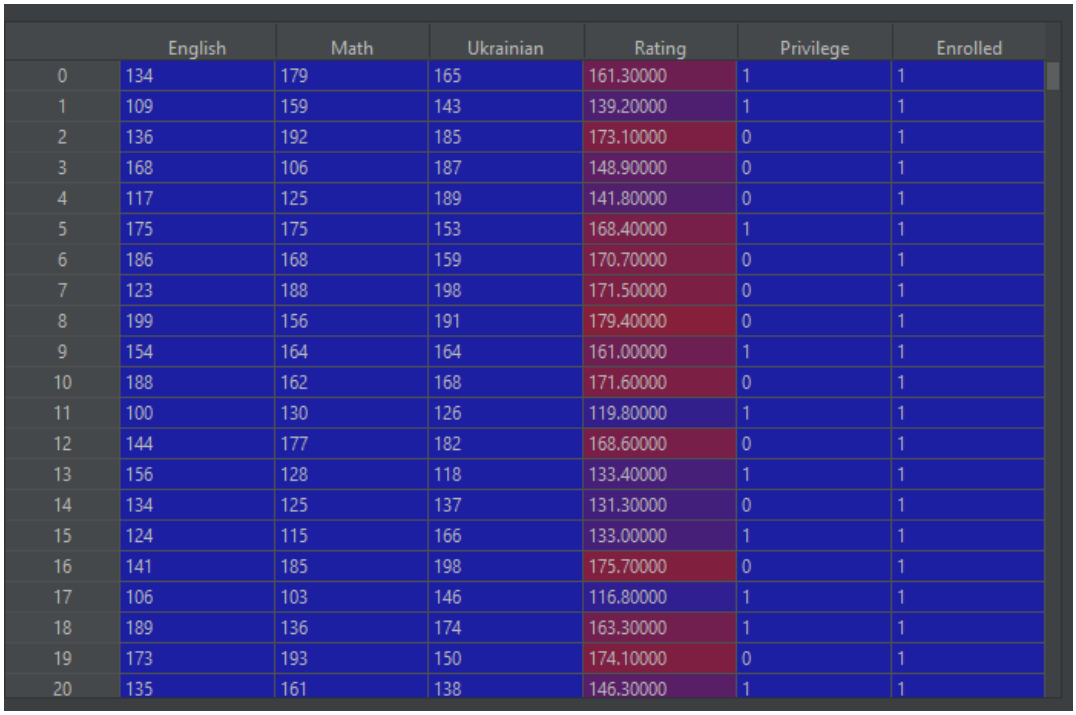
Тема: Прикладна задача машинного навчання

Постановка завдання:

1) Створення набору даних, що представляє собою оцінки студентів.
2) Поділ студентів на тих що вступили та тих, що ні.
3) Створення та тренування моделі.
4) Аналіз точності моделі.
5) Перевірка роботи моделі на нових випадкових даних.

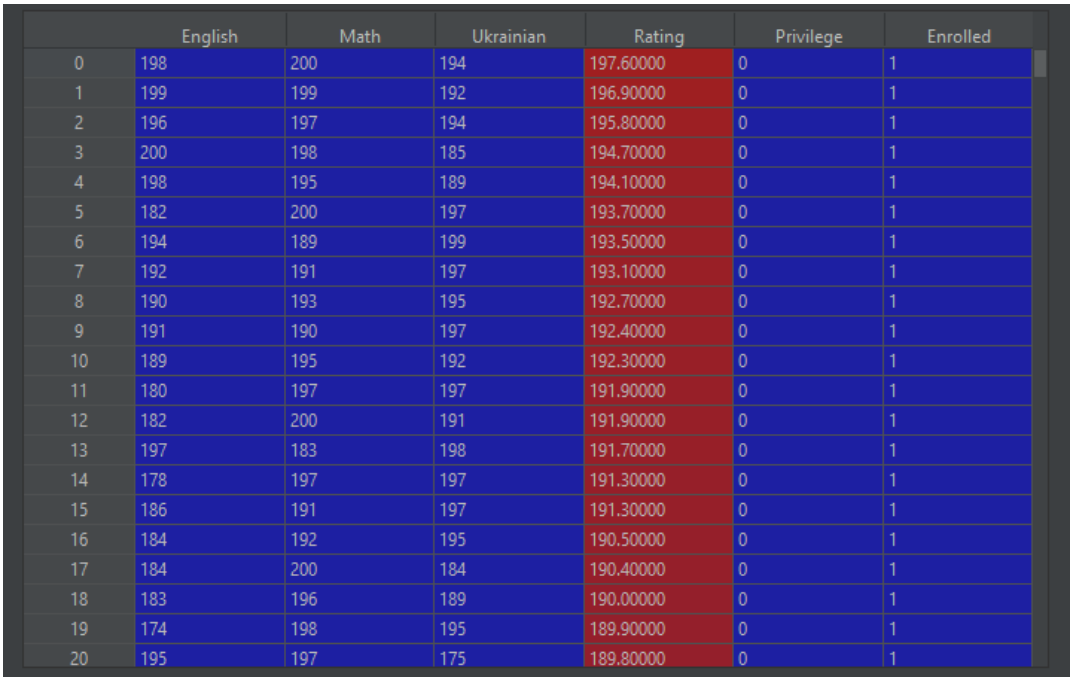
Хід роботи:

- Генерація даних



	English	Math	Ukrainian	Rating	Privilege	Enrolled
0	134	179	165	161.30000	1	1
1	109	159	143	139.20000	1	1
2	136	192	185	173.10000	0	1
3	168	106	187	148.90000	0	1
4	117	125	189	141.80000	0	1
5	175	175	153	168.40000	1	1
6	186	168	159	170.70000	0	1
7	123	188	198	171.50000	0	1
8	199	156	191	179.40000	0	1
9	154	164	164	161.00000	1	1
10	188	162	168	171.60000	0	1
11	100	130	126	119.80000	1	1
12	144	177	182	168.60000	0	1
13	156	128	118	133.40000	1	1
14	134	125	137	131.30000	0	1
15	124	115	166	133.00000	1	1
16	141	185	198	175.70000	0	1
17	106	103	146	116.80000	1	1
18	189	136	174	163.30000	1	1
19	173	193	150	174.10000	0	1
20	135	161	138	146.30000	1	1

- Розмітка даних



	English	Math	Ukrainian	Rating	Privilege	Enrolled
0	198	200	194	197.60000	0	1
1	199	199	192	196.90000	0	1
2	196	197	194	195.80000	0	1
3	200	198	185	194.70000	0	1
4	198	195	189	194.10000	0	1
5	182	200	197	193.70000	0	1
6	194	189	199	193.50000	0	1
7	192	191	197	193.10000	0	1
8	190	193	195	192.70000	0	1
9	191	190	197	192.40000	0	1
10	189	195	192	192.30000	0	1
11	180	197	197	191.90000	0	1
12	182	200	191	191.90000	0	1
13	197	183	198	191.70000	0	1
14	178	197	197	191.30000	0	1
15	186	191	197	191.30000	0	1
16	184	192	195	190.50000	0	1
17	184	200	184	190.40000	0	1
18	183	196	189	190.00000	0	1
19	174	198	195	189.90000	0	1
20	195	197	175	189.80000	0	1

- Тренування моделі

```
Epoch 1/200
563/563 [==============================] - 3s 2ms/step - loss: 2.3069 - accuracy: 0.6289
Epoch 2/200
563/563 [==============================] - 1s 2ms/step - loss: 0.5530 - accuracy: 0.7411
Epoch 3/200
563/563 [==============================] - 1s 2ms/step - loss: 0.5105 - accuracy: 0.7598
Epoch 4/200
563/563 [==============================] - 1s 2ms/step - loss: 0.4889 - accuracy: 0.7736
Epoch 5/200
563/563 [==============================] - 1s 2ms/step - loss: 0.4721 - accuracy: 0.7769
Epoch 6/200
563/563 [==============================] - 1s 2ms/step - loss: 0.4485 - accuracy: 0.7796
Epoch 7/200
563/563 [==============================] - 1s 2ms/step - loss: 0.4400 - accuracy: 0.7969
Epoch 8/200
563/563 [==============================] - 1s 2ms/step - loss: 0.4240 - accuracy: 0.8029
Epoch 9/200
563/563 [==============================] - 1s 2ms/step - loss: 0.4132 - accuracy: 0.8100
Epoch 10/200
563/563 [==============================] - 1s 2ms/step - loss: 0.4065 - accuracy: 0.8149
Epoch 11/200
563/563 [==============================] - 1s 2ms/step - loss: 0.3965 - accuracy: 0.8213
Epoch 12/200
563/563 [==============================] - 1s 2ms/step - loss: 0.4048 - accuracy: 0.8193
Epoch 13/200
563/563 [==============================] - 1s 2ms/step - loss: 0.3801 - accuracy: 0.8342
Epoch 14/200
563/563 [==============================] - 1s 2ms/step - loss: 0.3712 - accuracy: 0.8371
```

- Перевірка точності моделі

```
Epoch 200/200
563/563 [==============================] - 1s 2ms/step - loss: 0.1232 - accuracy: 0.9482
47/47 [==============================] - 0s 2ms/step - loss: 0.0829 - accuracy: 0.9740
47/47 [==============================] - 0s 1ms/step
Test accuracy: 97.40%
```

Для подальшого аналізу дані, на яких проводилось тестування було імпортовано до Excel файлу наступного вигляду:

	A	B	C	D	E	F	G	H
1	English	Math	Ukrainian	Rating	Privilege	Enrolled	predicted_enrollment	predict_confidance
2	125	136	116	126,7	0	0	0	0,000005
3	122	161	177	154,1	0	0	0	0,085649
4	107	161	106	128,3	0	0	0	0,000005
5	131	181	112	145,3	0	0	0	0,002086
6	179	130	109	138,4	0	0	0	0,000139
7	156	157	126	147,4	1	0	0	0,052000
8	150	140	166	150,8	0	0	0	0,040576
9	138	193	119	154,3	0	0	0	0,047377
10	124	106	151	124,9	0	0	0	0,000002
11	103	159	136	135,3	0	0	0	0,000042
12	155	154	104	139,3	0	0	0	0,000301
13	186	137	102	141,2	0	0	0	0,000115
14	105	179	167	153,2	0	0	0	0,000079
15	100	111	180	128,4	0	0	0	0,000000
16	163	181	103	152,2	0	0	0	0,000292
17	122	175	109	139,3	1	0	0	0,001697
18	119	125	145	129,2	0	0	0	0,000016
19	118	132	117	123,3	1	0	0	0,000010
20	169	103	138	133,3	0	0	0	0,000008
21	130	125	115	123,5	0	0	0	0,000002
22	118	167	194	160,4	0	0	0	0,480729
23	135	108	139	125,4	0	0	0	0,000002
24	100	173	195	157,7	0	0	0	0,000000
25	117	131	157	134,6	0	0	0	0,000105
26	101	149	172	141,5	1	0	0	0,000120
27	114	193	183	166,3	0	0	0	0,023234
28	157	169	136	155,5	0	0	0	0,114149

Більшість студентів з тих, яких нейромережа помилково не зарахувала до університету мали рейтинг на межі прохідного балу, а щодо абітурієнтів, які навпаки, були зараховані, хоч і не мали бути, непроходять по результату лише одного предмета.

191	174	121	163,2	0	1	0	0,47836
147	172	142	155,5	1	1	0	0,45484
170	121	191	156,7	1	1	0	0,37171
153	175	133	155,8	1	1	0	0,46566
120	200	173	167,9	0	1	0	0,24922
177	126	177	156,6	1	1	0	0,45670
198	151	123	156,7	1	1	0	0,32400
131	195	132	156,9	1	1	0	0,47714
175	132	192	162,9	0	1	0	0,41811
184	125	170	156,2	1	1	0	0,37619
200	128	190	168,2	0	0	1	0,60268
193	159	134	161,7	0	0	1	0,58831
154	148	187	161,5	0	0	1	0,67359
161	152	173	161	0	0	1	0,61085
199	187	118	169,9	0	0	1	0,75058
174	137	156	153,8	1	0	1	0,50803
184	138	190	167,4	0	0	1	0,80809
186	173	124	162,2	0	0	1	0,54053
160	144	181	159,9	0	0	1	0,55473
119	173	190	161,9	0	0	1	0,59120
172	152	164	161,6	0	0	1	0,66741
185	156	147	162	0	0	1	0,68455
175	137	192	164,9	0	0	1	0,67978
151	157	176	160,9	0	0	1	0,56929
175	137	160	155,3	1	0	1	0,63851
172	184	123	162,1	0	0	1	0,50055
148	148	189	160,3	0	0	1	0,56427
125	171	186	161,7	0	0	1	0,57838
130	177	175	162,3	0	0	1	0,60079
180	131	200	166,4	0	0	1	0,62610
163	143	186	161,9	0	0	1	0,66038
139	173	171	162,2	0	0	1	0,60892
194	149	144	161	0	0	1	0,60911
143	165	173	160,8	0	0	1	0,51837
119	176	199	165,8	1	0	1	0,69609
188	139	183	166,9	0	0	1	0,78986
177	165	143	162	0	0	1	0,63404
174	194	119	165,5	0	0	1	0,70972

- Тестування моделі на нових даних

```
Math score: 134, English score: 134, Ukrainian score: 154, rating: 140.0, privilege: 0
1/1 [==============================] - 0s 84ms/step
Student will not be able to enter a university, predicted value: 0.000
```

```
Math score: 133, English score: 166, Ukrainian score: 111, rating: 136.3, privilege: 0
1/1 [==============================] - 0s 91ms/step
Student will not be able to enter a university, predicted value: 0.000
```

```
Math score: 197, English score: 168, Ukrainian score: 127, rating: 167.3, privilege: 0
1/1 [==============================] - 0s 88ms/step
Student will be able to enter a university, predicted value: 0.829
```

```
Math score: 186, English score: 172, Ukrainian score: 196, rating: 184.8, privilege: 0
1/1 [==============================] - 0s 84ms/step
Student will be able to enter a university, predicted value: 1.000
```

Висновок

Було поглиблено знання бібліотеки keras, створену для побудови та навчання нейронних мереж; було згенеровано та розмічено набір даних, який представляє собою студентів з їх екзаменаційними балами та наявністю пільги, за встановленими правилами прийому для подальшого тренування моделі; було реалізовано нейронну мережу для ухвалення рішення про зарахування до університету абітурієнтів після складання іспитів та досліджено точність її роботи на тестових, та нових, випадкових даних; було експериментально встановлено найбільш прийнятний тип архітектури мережі та її тренувальних параметрів та досягнуто точності 98%; загалом, було вирішено досить практичну задачу передбачення можливості вступу абітурієнта до університету в залежності від результатів його іспитів, використовуючи машинне навчання.

```
Epoch 200/200
563/563 [==============================] - 1s 978us/step - loss: 0.0825 - accuracy: 0.9667
47/47 [==============================] - 0s 880us/step - loss: 0.0668 - accuracy: 0.9800
47/47 [==============================] - 0s 724us/step
Test accuracy: 98.00%
```

Вихідний код

Lab6.py

```python
import ...

AMOUNT_OF_APPLICANTS = 6000
MAX_ADMISSIBLE_STUDENTS = int((AMOUNT_OF_APPLICANTS * 70) / 300)  # 1400
MAX_ADMISSIBLE_PRIVILEGE_STUDENTS = int(0.1 * MAX_ADMISSIBLE_STUDENTS)  # 140
MAX_ADMISSIBLE_UNPRIVILEGE_STUDENTS = MAX_ADMISSIBLE_STUDENTS - MAX_ADMISSIBLE_PRIVILEGE_STUDENTS  # 1260


def generate_applicants():
    students = pd.DataFrame(columns=['English', 'Math', 'Ukrainian', 'Rating', 'Privilege', 'Enrolled'])
    for i in range(AMOUNT_OF_APPLICANTS):
        math_score = random.randint(100, 200)
        english_score = random.randint(100, 200)
        ukrainian_score = random.randint(100, 200)
        is_privilege = random.choices([0, 1], weights=[0.75, 0.25])[0]
        student = pd.DataFrame({'English': [english_score], 'Math': [math_score], 'Ukrainian': [ukrainian_score],
                                'Rating': [math_score * 0.4 + english_score * 0.3 + ukrainian_score * 0.3],
                                'Privilege': [is_privilege], 'Enrolled': [1]})
        students = pd.concat([students, student], ignore_index=True)
    return students


def cannot_apply(students):
    students.loc[(students.loc[:, 'Privilege'] == 0) & (students.loc[:, 'Rating'] < 160), 'Enrolled'] = 0
    students.loc[(students.loc[:, 'Privilege'] == 0) & (students.loc[:, 'Math'] < 140), 'Enrolled'] = 0
    students.loc[students.loc[:, 'English'] < 120, 'Enrolled'] = 0
    students.loc[students.loc[:, 'Math'] < 120, 'Enrolled'] = 0
    students.loc[students.loc[:, 'Ukrainian'] < 120, 'Enrolled'] = 0
    students.loc[(students.loc[:, 'Privilege'] == 1) & (students.loc[:, 'Rating'] < 144), 'Enrolled'] = 0
    students.sort_values(by=['Enrolled', 'Rating'], ascending=False, inplace=True)
    students.reset_index(drop=True, inplace=True)
    students.loc[:MAX_ADMISSIBLE_UNPRIVILEGE_STUDENTS, 'Privilege'] = 0  # privilege don't make any sense
    students.loc[(students['Privilege'] == 0) & (students.index >= MAX_ADMISSIBLE_UNPRIVILEGE_STUDENTS), 'Enrolled'] = 0
    students.sort_values(by=['Privilege', 'Enrolled', 'Rating'], ascending=False, inplace=True)
    students.reset_index(drop=True, inplace=True)
    students.loc[(students['Privilege'] == 1) & (students.index >= MAX_ADMISSIBLE_PRIVILEGE_STUDENTS), 'Enrolled'] = 0
    students.sort_values(by=['Enrolled', 'Rating'], ascending=False, inplace=True)
    students.reset_index(drop=True, inplace=True)
    # print(len(students[students.loc[:, 'Enrolled'] == 1]))
    # print(len(students[(students.loc[:, 'Enrolled'] == 1) & (students.loc[:, 'Privilege'] == 1)]))
    return students


def build_model(X_train, y_train):
    model = Sequential()
    model.add(Dense(16, input_dim=5, activation='relu'))
    model.add(Dense(8, activation='relu'))
    model.add(Dense(4, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=200, batch_size=8)
    model.save(r'F:\Egor\Уроки\Машинне навчання\Лаб6\university_model.h5')
    return model


applicants = generate_applicants()
applicants = cannot_apply(applicants)
train_data, test_data = train_test_split(applicants)

X_train = train_data.drop('Enrolled', axis=1).values.astype('float32')
y_train = train_data['Enrolled'].values.astype('float32')
X_test = test_data.drop('Enrolled', axis=1).values.astype('float32')
y_test = test_data['Enrolled'].values.astype('float32')

network = build_model(X_train, y_train)
# network = load_model(r'F:\Egor\Уроки\Машинне навчання\Лаб6\university_model.h5')
test_loss, test_acc = network.evaluate(X_test, y_test)

y_values = network.predict(X_test)
y_predicted = [1 if y >= 0.5 else 0 for y in y_values]
accuracy = sum(y_predicted == y_test) / len(y_test)
print(f'Test accuracy: {accuracy * 100:.2f}%')
print(len(y_test))

result_df = test_data.assign(predicted_enrollment=y_predicted)
result_df = result_df.assign(predict_confidance=y_values)
result_df.sort_values(by=['predicted_enrollment', 'Enrolled'], inplace=True)
result_df.to_excel(r'F:\Egor\Уроки\Машинне навчання\Лаб6\students.xlsx', index=False)
```

Testing Model.py

```python
import ...

model = load_model(r'F:\Egor\Уроки\Машинне навчання\Лаб6\university_model.h5')
math_score = random.randint(100, 200)
english_score = random.randint(100, 200)
ukrainian_score = random.randint(100, 200)
is_privilege = random.choices([0, 1], weights=[0.75, 0.25])[0]
rating = math_score * 0.4 + english_score * 0.3 + ukrainian_score * 0.3

print(f'Math score: {math_score}, English score: {english_score},'
      f' Ukrainian score: {ukrainian_score}, rating: {rating:.1f}, privilege: {is_privilege}')

student = [math_score, english_score, ukrainian_score, rating, is_privilege]
prediction = model.predict(np.array([student]))[0][0]
if prediction < 0.5:
    print(f'Student will not be able to enter a university, predicted value: {prediction:.3f}')
else:
    print(f'Student will be able to enter a university, predicted value: {prediction:.3f}')
```