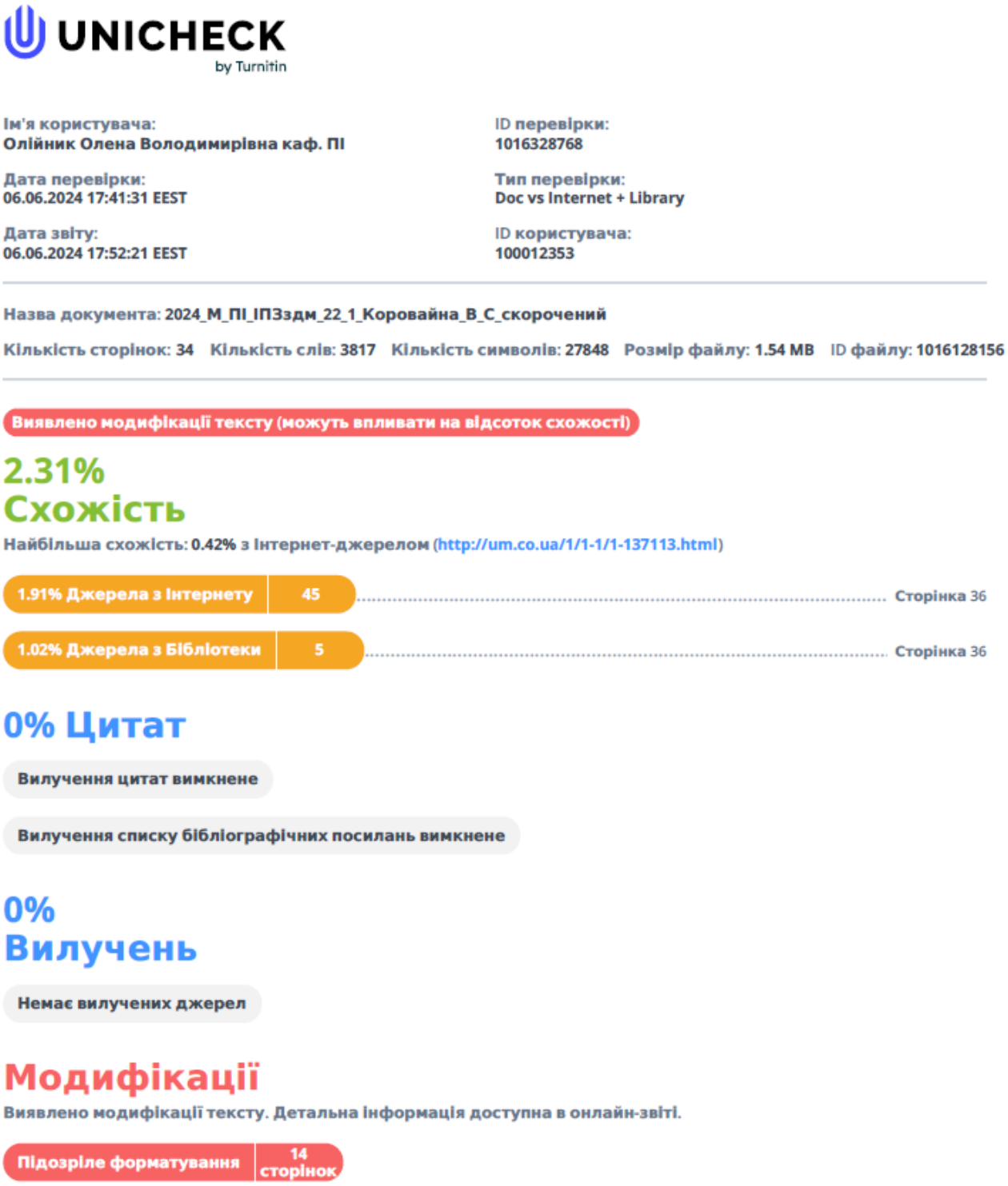
Додаток А.

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016328768

Дата перевірки:
06.06.2024 17:41:31 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2024 17:52:21 EEST

ID користувача:
100012353

Назва документа: 2024_М_ПІ_ІПЗздм_22_1_Коровайна_В_С_скорочений

Кількість сторінок: 34 Кількість слів: 3817 Кількість символів: 27848 Розмір файлу: 1.54 MB ID файлу: 1016128156

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

2.31%
Схожість

Найбільша схожість: 0.42% з Інтернет-джерелом (http://um.co.ua/1/1-1/1-137113.html)

1.91% Джерела з Інтернету 45 ..... Сторінка 36

1.02% Джерела з Бібліотеки 5 ..... Сторінка 36

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 14 сторінок

Додаток Б. Лістинг коду з попередньої підготовки даних

```
import pandas as pd import seaborn as sns

students_df = pd.read_excel('ID_Vlada.xlsx')

students_df.head(10)
# функція віднесення студента до групи залежно від кількості боргів
def label_student (n_fails):
     if n_fails == 0:
        return("успішно")
     if n_fails > 0 and n_fails <= 6:
         return("борги")
     if n_fails > 6:
         return("багато боргів")

# створення додаткового стовпця в дату сеті, де вказується група студента залежно від
кількості боргів
students_df['група боргів'] = students_df.apply(lambda row:
label_student(row['Незадовільних']), axis=1)

students_df.info()
students_df.isna().any()

 import matplotlib.pyplot as plt
def draw_pie_diagram(name):
     print(name)
     name.plot(kind='pie')
     plt.axis('equal')
     plt.show()

 def draw_pie_diagram_advanced(name, lst=[], exp=None):
     print(name)
     if exp != None:
            exp = exp
    fig1, ax1 = plt.subplots(figsize=(12,7))
    ax1.pie(name,explode=exp,labels=lst,autopct='%1.1f%%',shadow=True,startangle=90)
# Однакове співвідношення сторін гарантує, що пиріг буде намальований у вигляді кола
    ax1.axis('equal')
    plt.tight_layout()
    plt.legend()
    plt.show()

print(students_df.groupby(['Незадовільних'])['ID'].count())
print(students_df.groupby(['Незадовільних'])['ID'].count()/len(students_df.index)*100)
f,ax = plt.subplots(figsize=(15,10))
sns.countplot(students_df['Незадовільних'])

stud_sex = students_df.groupby(['Стать'])['ID'].count()
draw_pie_diagram_advanced(stud_sex, ['Жіноча', 'Чоловіча'], [0.1, 0])

# sns.countplot(students_df['Форма навчання'])
print(students_df.groupby(['Форма навчання'])['ID'].count())

print(students_df.groupby([ 'Кваліфікація', 'Курс'])['ID'].count())
print(students_df.groupby([ 'Кваліфікація', 'Курс'])['ID'].count()/
len(students_df.index)*100)
f,ax = plt.subplots(figsize=(15,10))
sns.countplot(students_df['Кваліфікація'],hue=students_df['Курс'])
```

```
print(students_df.groupby([ 'Форма навчання', 'Кваліфікація'])['ID'].count())
print(students_df.groupby([ 'Форма навчання', 'Кваліфікація'])['ID'].count()/
len(students_df.index)) f,ax = plt.subplots(figsize=(15,10))
sns.countplot(students_df['Кваліфікація'],hue=students_df['Форма навчання'])

print(students_df.groupby([ 'Форма навчання', 'Кваліфікація', 'група боргів'])['ID']
.count())
print(students_df.groupby([ 'Форма навчання', 'Кваліфікація', 'група боргів'])['ID']
.count()/len(students_df.index))

print(students_df.groupby([ 'група боргів'])['ID'].count())
print(students_df.groupby([ 'група боргів'])['ID'].count()/len(students_df.index))
f,ax=plt.subplots(figsize=(15,10))
sns.countplot(students_df[ 'група боргів'])

def draw_diagramm(kval, form):
     print("Діаграма відсоткового співвідношення студентів з боргами і без для групи:
"+ str(kval)+ ', ' + str(form) +'Форма навчання')
     stud_debts = students_df[(students_df['Форма навчання ']==form) & (students_df
['Кваліфікація']==kval)].groupby(['група боргів'])['ID'].count()
   draw_pie_diagram_advanced(stud_debts, ['борги', 'багато боргів', 'успішно'])

# kval_list = ["Магістр"]
# form_list = ['Очна', "Заочно-дист."]
# for kval in kval_list:
#     for form in form_list:
#         draw_diagramm(kval, form)

# print("Діаграма відсоткового співвідношення студентів з боргами та без групи:
Магістр, Заочно-дист. форма навчання")
# stud_debts = students_df[(students_df['Форма навчання']=='Заочно-дист.') &
(students_df['Кваліфікація']=='Магістр')].groupby(['група боргів'])['ID'].count()
# draw_pie_diagram_advanced(stud_debts, ['борги', 'багато боргів'])

stud_debts = students_df[students_df.Кваліфікація=='Бакалавр'].groupby(['група боргів'])
['ID'].count()
draw_pie_diagram_advanced(stud_debts, ['борги', 'багато боргів', 'успішно'])

pr1 = students_df.groupby([ 'Спеціальність'])['Незадовільних'].median()
# print(pr1)
# type(pr1)
pr1.nlargest(15)
# pr1.nsmallest(10)

f,ax = plt.subplots(figsize=(15,10))
sns.countplot(students_df['Кафедра'],hue=students_df['група боргів'])

students_df = students_df[students_df['Усього годин аудиторних занять у семестрі'].
notnull()]

Y = students_df['Незадовільних']

X = students_df[['Форма навчання', 'Кваліфікація', 'Курс', 'Спеціальність', 'Академ.
відпустка (діюча) - так / ні', 'Усього годин пропусків у семестрі', 'Усього годин
аудиторних занять у семестрі']]
Y = students_df['група боргів']

from sklearn.preprocessing import LabelEncoder
le_f = LabelEncoder()
X['Форма навчання'] = le_f.fit_transform(X['Форма навчання'].values)
le_k = LabelEncoder()
X['Кваліфікація'] = le_k.fit_transform(X['Кваліфікація'].values)
```

```
le_s = LabelEncoder()
X['Спеціальність'] = le_s.fit_transform(X['Спеціальність'].values)
le_a = LabelEncoder()
X['Академ. відпустка (діюча) - так / ні'] = le_a.fit_transform(X['Академ. відпустка
(діюча) - так / ні'].values)

X.info()
sns.set(font_scale= 1)
hm = sns.heatmap(X.corr(), cbar=True, annot=True)
plt.show()

X.head()

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=1000.0, random_state=0 )
lr.fit(x_train,y_train)

print("Тренувальна: {0}".format(lr.score(x_train, y_train)))
print("Тестова: {0}".format(lr.score(x_test, y_test)))

pred_y = lr.predict(x_test)
from sklearn import metrics
# Крос-валідація моделі
print(metrics.classification_report(pred_y, y_test))

  from sklearn import svm
clf = svm.SVC()
clf.fit(x_train,y_train)

print("Тренувальна: {0}".format(clf.score(x_train, y_train)))
print("Тестова: {0}".format(clf.score(x_test, y_test)))

pred_y = clf.predict(x_test)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print(metrics.classification_report(pred_y, y_test))

from sklearn.ensemble import RandomForestClassifier
clas = RandomForestClassifier(max_depth=15, random_state=0)
clas.fit(x_train,y_train)

print(clas.feature_importances_)

print(clas.score(x_train, y_train))
print(clas.score(x_test, y_test))

pred_y = clas.predict(x_test)
from sklearn import metrics
# Точність моделі, як часто класифікатор є правильним?
print(metrics.classification_report(pred_y, y_test))

from sklearn.model_selection import train_test_split
X = students_df[['Курс']].values.astype(int)
y = students_df['Незадовільно'].values.astype(int)

import numpy as np
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X, y)
```

```
y_pred = model.predict(X)
print('Slope: {:.2f}'.format(model.coef_[0]))
print('Intercept: {:.2f}'
.format(model.intercept_))

plt.scatter(X, y)
plt.plot(X, model.predict(X), color='red', linewidth=2);

model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print('Slope: {:.2f}'.format(model.coef_[0]))
print('Intercept: {:.2f}'.format(model.intercept_))

print(model.score(x_train, y_train))
print(model.score(x_test, y_test))

test_pred_y = model.predict(x_test)
train_pred_y = model.predict(x_train)
from sklearn import metrics
# Точність моделі, як часто класифікатор є правильним?
# print(metrics.classification_report(pred_y, y_test))

from sklearn.metrics import mean_absolute_error, mean_squared_error,
median_absolute_error, r2_score

print('MSE train: {:.3f}, test: {:.3f}'.format( mean_squared_error(y_train, train_pred_y),
mean_squared_error(y_test, test_pred_y)))
print('R^2 train: {:.3f}, test: {:.3f}'.format( r2_score(y_train, train_pred_y),
r2_score(y_test, test_pred_y)))

X.info()

xx = X[['Курс', 'Спеціальність', 'Усього годин пропусків у семестрі', 'Усього годин
аудиторних занятій у семестрі']]

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(xx, Y, test_size=0.3, random_state=0)

from sklearn.ensemble import RandomForestClassifier
clas = RandomForestClassifier(max_depth=15, random_state=0)
clas.fit(x_train,y_train)

print(clas.feature_importances_)

print(clas.score(x_train, y_train))
print(clas.score(x_test, y_test))

pred_y = clas.predict(x_test)
from sklearn import metrics
# Точність моделі, як часто класифікатор є правильним?
print(metrics.classification_report(pred_y, y_test))

super_df = students_df
from sklearn.preprocessing import LabelEncoder
le_f = LabelEncoder()
super_df['Форма навчання'] = le_f.fit_transform(super_df['Форма навчання'].values)
le_k = LabelEncoder()
super_df['Кваліфікація'] = le_k.fit_transform(super_df['Кваліфікація'].values)
le_s = LabelEncoder()
super_df['Спеціальність'] = le_s.fit_transform(super_df['Спеціальність'].values)
le_a = LabelEncoder()
```

```
for i in range(1, 6):
    print("Для {0} курса ".format(i))
    u_df = super_df[students_df ['Курс']==i]
    u_df['Академ. відпустка (діюча) - так / ні'] = le_a.fit_transform(u_df['Академ.
відпустка (діюча) - так / ні'].values)
    x1 = u_df[['Форма навчання', 'Кваліфікація', 'Курс', 'Спеціальність', 'Академ.
відпустка (діюча) - так / ні', 'Усього годин пропусків у семестрі', 'Усього годин
аудиторних занять у семестрі']]
    Y = u_df['група боргів']
    x_train,x_test,y_train,y_test=train_test_split(x1, Y, test_size=0.3, random_state=0)
    clas = RandomForestClassifier(max_depth=15, random_state=0)
    clas.fit(x_train,y_train)
    print(clas.score(x_train, y_train))
    print(clas.score(x_test, y_test))
    pred_y = clas.predict(x_test)
    from sklearn import metrics
    # Точність моделі, як часто класифікатор є правильним?
    print(metrics.classification_report(pred_y, y_test))

    for i in le_k.transform(list(le_k.classes_)):
    print("Для {0}".format(le_k.inverse_transform(i)))
    u_df = super_df[students_df['Кваліфікація']==i]
    u_df['Академ. Відпустка (діюча) - так / ні'] = le_a.fit_transform(u_df['Академ.
відпустка (діюча) - так / ні'].values)
    x1 = u_df[['Форма навчання', 'Кваліфікація', 'Курс', 'Спеціальність', 'Академ.
відпустка (діюча) - так / ні', 'Усього годин пропусків у семестрі', 'Усього годин
аудиторних занять у семестрі']]
    Y = u_df['група боргів']
    x_train,x_test,y_train,y_test=train_test_split(x1, Y, test_size=0.3, random_state=0)
    clas = RandomForestClassifier(max_depth=15,random_state=0)
    clas.fit(x_train,y_train)
    print(clas.score(x_train, y_train))
    print(clas.score(x_test, y_test))
    pred_y = clas.predict(x_test)
    from sklearn import metrics
    # Точність моделі, як часто класифікатор є правильним?
    print(metrics.classification_report(pred_y, y_test))

x_train, x_test, y_train, y_test = train_test_split(x1, Y, test_size=0.3, random_state=0)
clas = RandomForestClassifier(max_depth=15, random_state=0)
clas.fit(x_train,y_train)
print(clas.score(x_train, y_train))
print(clas.score(x_test, y_test))
pred_y = clas.predict(x_test) from sklearn import metrics
# Точність моделі, як часто класифікатор є правильним?
print(metrics.classification_report(pred_y, y_test))

students_df.head(10)

subj_df = students_df[students_df['Дисципліни, за якими отримані незадовільні оцінки']
.notnull()]
subj_df = subj_df[subj_df['Спеціальність']=='121 ІПЗ']

subj_df.head()
 subject_dict = {} for i in subj_df['Дисципліни, за якими отримані незадовільні оцінки']:
clear_data = i.replace(', ', ',').split(',')
     for j in clear_data:
         if j in subject_dict:
            subject_dict[j] = (subject_dict[j]+1)
```

```
        else:
            subject_dict[j] = 1
print(subject_dict)

from collections import OrderedDict
d_sorted_by_value = OrderedDict(sorted(subject_dict.items(), key=lambda x: x[1]))

from beautifultable import BeautifulTable
table = BeautifulTable()
table.column_headers = ['Назва дисципліни', 'Кількість боржників']

count = 0
for key in reversed(d_sorted_by_value):
     table.append_row([key, d_sorted_by_value[key]])
     count += 1
     if count == 15:
        break

print(table)

# функція віднесення студента у групу в залежності від кількості боржників
 def label_student1(missings, lessons):
     p = missings/lessons
     return p

students_df['пропуски'] = students_df.apply(lambda row: label_student1(row['Усього годин
пропусків у семестрі'], row['Усього аудіторних занятий в семестре']), axis=1)

j_df = students_df[students_df['пропуски'] < 0.1]

j_df.head(15)

# plt.scatter(students_df['пропуски'], students_df['Незадовільних'])
plt.plot(students_df['Незадовільних'], students_df['пропуски'], color='red', linewidth=2);
```

Додаток В. Апробація результатів роботи

PREDICTIVE MODEL FOR ASSESSING PERFORMANCE OF STUDENTS

Vladyslava Korovaina[1], Dmytro Kolesnykov[2], Oleksii Nazarov[3], Nataliia Nazarova[4]
[1]Master's student of the department of Software Engineering, NURE, Ukraine, vladyslava.korovaina.cpe@nure.ua
[2]Associate professor of the department of Software Engineering, NURE, Ukraine, dmytro.kolesnykov@nure.ua,
ORCID iD: 0000-0002-4901-6869
[3]Associate professor of the department of Software Engineering, NURE, Ukraine, oleksii.nazarov1@nure.ua,
ORCID iD: 0000-0001-8682-5000
[4]Assistant of the Department of Higher Mathematics, NURE, Ukraine, nataliia.nazarova@nure.ua,
ORCID iD: 0009-0007-7816-7088

The object of research is the process of developing a predictive model for assessing the performance of university students based on the results of current studies. The purpose of the study is to build a predictive model of students' session results depending on the estimated parameters of current performance. In the course of the study, the main problems in this area were analyzed, and goals were set for their direct implementation. We also conducted preliminary data processing to build a machine learning model. After that, various machine learning models were built, and the qualitative indicators of each model were evaluated. After selecting the optimal model, a graphical user interface for the predictive model was created. As a result of the study, a predictive model of university students academic performance was created, as well as a graphical interface for its use. The most significant factors in predicting student performance have been identified.
PREDICTING STUDENT PERFORMANCE, MACHINE LEARNING MODEL, DATA PREPARATION, DATA SET, CLASSIFICATION.

Коровайна В.С., Колесников Д.О., Назаров О.С., Назарова Н.В. Прогностична модель оцінювання успішності студентів. Об'єктом дослідження є процес розробки прогнозної моделі для оцінки успішності студентів університету за підсумками поточного навчання. Мета роботи – побудова прогнозної моделі підсумків сесії студентів залежно від оціночних параметрів поточної успішності. У процесі дослідження було проведено аналіз основних проблем у цій галузі, поставлено цілі для їх безпосереднього виконання. Також було проведено попередню обробку даних для побудови моделі машинного навчання. Після цього було побудовано різні моделі машинного навчання, а також оцінено якісні показники кожної моделі. Після вибору оптимальної моделі було створено графічний інтерфейс користувача прогнозної моделі. У результаті дослідження було створено прогнозну модель успішності студентів вишу, а також графічний інтерфейс для її використання. Визначено найбільш значущі фактори під час прогнозування успішності у студентів.
ПРОГНОЗУВАННЯ УСПІШНОСТІ СТУДЕНТІВ, МОДЕЛЬ МАШИННОГО НАВЧАННЯ, ПІДГОТОВКА ДАНИХ, НАБІР ДАНИХ, КЛАСИФІКАЦІЯ.

1. Introduction

The level of student success in higher education is a form of diagnosis and prediction of the level of commitment of a future specialist. In turn, student success is an indicator of the performance of the higher education institution in solving educational tasks. In order to solve these tasks as efficiently as possible, constant objective evaluation, adjustment and management are required. However, management is impossible without forecasting. Therefore, it is necessary to predict the performance of students at all stages of education.

Having information about those students who are most likely to have academic debt by the end of the semester if they do not change the current trend, we can influence students and thereby improve their academic performance.

The purpose of this thesis is to create a predictive model of the academic performance of NURE students. The availability of such a model will allow us to pay more attention to students who are at risk of having a large number of debts in academic disciplines and, as a result, will be candidates for expulsion. Early identification of such students will allow for more detailed and personalized work with them to help them manage their academic workload.

Based on the objective, this work includes the following tasks: review the literature on the subject, study the methods and algorithms used, clean and prepare the initial data, develop a predictive model, test the results, create a graphical user interface for the module for predicting student performance.

2. Subject area analysis

Education plays one of the most important roles in any country. The quality of education in a given society largely determines the pace of its economic and political development and its moral state.

The rapid development of information technologies makes it possible to automate many areas of human activity and increase their efficiency, and education is no exception. This paper will focus on creating a predictive model of student performance based on current grades using data mining technologies.

The measure of the quality of education received by a particular student is his or her grades in the subjects passed. When we talk about an educational institution, one of the

Рисунок В.1 – Перша сторінка статті у журналі «Біоніка інтелекту»

measures of the quality of education it provides is the aggregate of its students' grades. Timely measures to help students who cannot cope with the academic load are one of the main parts of educational work in higher education institutions that affect the quality of education in the institution.

Recently, many different changes have been made to improve the quality of education in higher education institutions. For example, the transition from a traditional grading system to a point system eliminates the possibility that a student who has not attended classes throughout the semester will simply come and take an exam. To be allowed to take the exam, they must earn a certain number of points, which in turn requires them to attend classes and complete current assignments.

This approach has an effective impact on the understanding of the study material. According to many studies, information that has been studied over a long period of time is retained for a long time, while "cramming" the night before an exam can only lead to a good result on the exam, which will eventually be passed, but the student will have no residual knowledge of the subject. In addition, there is a milestone control, a date set in the middle of the semester when a certain part of the material must be passed.

However, knowing the peculiarities of student life, many students still leave everything until the last minute. Some students manage to pass the course, while others are left with a debt for another semester. The problem is that the single dean's office of Tomsk Polytechnic University starts its control activities only after the student has already incurred debts.

Therefore, these measures cannot be called preventive. The main task of the predictive model is to identify such students and conduct certain talks with them before the problem of academic debt arises. At present, this measure is partly implemented by the fact that each group of freshmen has a tutor, and this tutor accompanies each group until the second year, and the schedule includes such an event as the "tutor's hour", where he analyzes the students' progress.

This is an excellent practice and should not be abandoned, but the human factor comes into play. It is not always possible for a tutor to convey to students the importance of attending classes. The introduction of a new system for predicting end-of-semester debt based on current academic performance is an important step in automating the educational process.

In this way, a single dean's office will be able to take control measures against at-risk students much earlier than a real problem arises. In this way, it will be possible to help students successfully complete the semester and develop professional skills, and to expel those who are not interested in studying earlier, freeing up places for those who really want and are ready to receive knowledge.

The main focus is on the forecasting system, because it is not only about monitoring the attendance of students. The data analysis shows that the final result of the semester is influenced not only by one factor of attendance, but by a whole range of different data. For example, there is a whole category of students who are already working in their field, mostly masters students and final year students. These students do not always have the opportunity to attend classes, and yet they show excellent results at the end of the session because they understand many aspects of the profession at a higher level than their classmates.

In fact, a huge number of factors affect a student's performance, and some of the most important are motivation to study, morale, relationships with classmates, and so on, but thanks to the fact that the system will identify problem students and will be able to work with each of them in detail, it will be possible to identify what problems exist with this particular student who risks ending the semester with a large amount of debt. It is possible to identify such parameters as motivation, determination, and psychological data, but this requires a large number of tests to be conducted on a regular basis. These methods are not very effective due to the complexity of their implementation, as well as the verification and interpretation of the results.

Once it is clear that this problem is indeed relevant, it is worth considering the existing methods for solving it.

At present, there are no publicly available materials devoted to the implementation and use of a system for predicting student performance in higher education institutions. Although this idea is not innovative, since about 2010 there have been articles about predicting the performance of applicants, predicting the performance of students in a particular course, where the following parameters are usually taken as initial data: the level of current knowledge of the subject, the number of absences, intermediate control, grades in the courses that provide the discipline.

Next, we will consider machine learning algorithms used to solve similar problems in the area of student performance prediction. The most commonly used methods and algorithms will be presented, as well as a description of existing works with specific tasks for which these methods are used.

Some sources use the clustering method, in our opinion this is not quite correct, since we already have the right classes, namely the number of debts, so in this case we should use classification methods. Clustering refers to unsupervised machine learning methods, and classification and regression refer to supervised learning, since they take markers for each class as input. Cluster analysis methods for estimating the final grade in a subject have been applied in [1].

3. K-Nearest Neighbors Algorithm

The K-nearest neighbor algorithm (KNN) is a type of supervised machine learning algorithm that can be used for both classification and regression prediction tasks. It is one of the easiest algorithms to understand, but it has been proven effective in a variety of tasks and is not only used for educational purposes. The Nearest Neighbors algorithm is also

Рисунок В.2 – Друга сторінка статті у журналі «Біоніка інтелекту»

called a "lazy" classifier because it does not build a model during training, but simply stores data. All computations are started only when it is necessary to classify new data.

The essence of this algorithm is that the prediction of new data values is based on their proximity to already labeled data in the training set. In other words, if a new data point has 4 points of class A and 1 point of class B among its nearest neighbors, then this new point will be defined as class A. Thus, the k-nearest neighbors algorithm has 2 most important parameters, namely the distance metric (Euclidean, Manhattan, or Hamming) and the number of neighbors we will consider. A visual representation of the algorithm is shown in Figure 1.

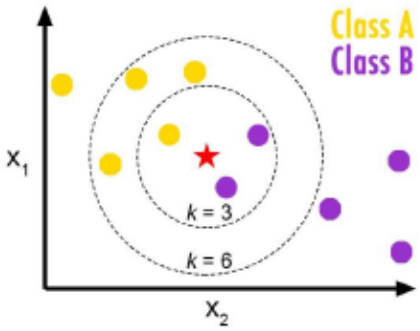


Fig. 1. KNN Algorithm Operation

This figure shows an array of source points color-coded according to their class membership. The asterisk marks the point to be classified. All points are plotted in two dimensions, along the X1 and X2 axes. If the set number of nearest neighbors is 3, then the unlabeled object belongs to class B, if it is 6, then it belongs to class A.

This algorithm has many nuances, for example, we can add weights to the voting data depending on the proximity to our unlabeled object. It is these nuances that make the KNN algorithm relevant for solving a wide range of tasks.

The benefits of this algorithm are:

- Easy to understand and interpret.

- Works well with non-linear data.

- It is a universal algorithm, suitable for both classification and regression tasks.

- It has relatively high accuracy.

Disadvantages of this algorithm:

- Large memory consumption to store all the data, unlike algorithms that use model building.

- Sensitivity to data size.

- Slow prediction for large amounts of data.

- High sensitivity to data noise.

In [2], this algorithm is used to classify the grade of an individual student in each subject based on his or her previous grades and the grades of students of previous courses with the most similar parameters in these subjects. This article presents a fairly high accuracy of the algorithm for this task, with a maximum grade prediction error of 0.55 points. However, only 307 students of one subject were considered, and there was no question of implementing this methodology in the university system.

This algorithm is also used in [3], where a student's grade for an exam in a given subject is predicted based on his or her grades in previous subjects, attendance, and midterm grades.

4. Support Vector Method

Support Vector Machines (SVM) are a family of similar learning algorithms for solving classification and regression problems. It is one of the most common learning methods belonging to the family of linear classifiers. One of the characteristics of the support vector method is that it consistently reduces the empirical classification error and increases the variance. Therefore, this method is also called the maximum distance classification method.

The basic idea of the support vector method can be illustrated by an example: there are points on a plane that are labeled into 2 classes that are linearly separated. In this case, the resulting function will be the plane that separates these classes. However, it is possible to draw many hyperplanes that separate these classes. To find the optimal hyperplane, you need to find the maximum sum of the normal vectors from class A and class B. A visual representation of this method can be seen in Figure 2. In this figure, the reference vectors are perpendicular to the normals.

The formal description of this method is as follows - suppose we have an instructive example:

$$\{(X_1, C_1), (X_2, C_2), \ldots, (X_l, C_l)\},$$

where

X_l – is a p-dimensional real vector;

C_l – the value 1 or -1 that the class takes.

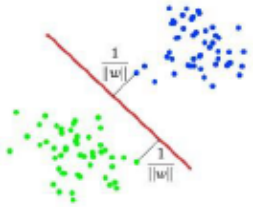


Fig. 2. Support Vector Method

Рисунок В.3 – Третя сторінка статті у журналі «Біоніка інтелекту»

The support vector method builds a classification function in the form of:

$$F(x)=sign([w,x]+b), \quad (1)$$

where

[,] – scalar multiplication;

w – is a normal vector to the separating hyperplane;

b – auxiliary parameter.

So we can write it all down in the form of an optimization problem that has one solution, and only one:

$$\begin{cases} \|\mathbf{w}\|^2 \to \min \\ c_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1, \quad 1\leq i\leq n. \end{cases} \quad (2)$$

This problem is solved by quadratic programming and using Lagrange multipliers.

The case when there are 2 separate classes has been considered. In practice, the classes are almost always not linearly separable, and the task is to classify more than 2 classes. To solve the problem with linearly inseparable classes, we allow the classifier to make an error on the training set. Let's write the equation for this assumption.

$$\begin{cases} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \to \min_{w,b,\xi_i} \\ c_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geq 1-\xi_i, \quad 1\leq i\leq n \\ \xi_i\geq 0, \quad 1\leq i\leq n \end{cases} \quad (3)$$

where

C – method setting parameter,

$\mathcal{E}_i$ – is the value of the allowable error.

To solve multi-class problems, the generalized support vector method is used, since the transition to classification into many classes is made by splitting into 2 classes, such as the corresponding class and the non-corresponding class. This strategy is also called "One vs. All" and is used to apply binary classifiers to multi-class tasks.

Advantages of the algorithm:

- The problem is well studied and has a single solution.

- The principle of the optimal separating hyperplane leads to a reliable classification.

- Equivalent to a two-layer neural network, where the number of neurons in the hidden layer is automatically determined as the number of support vectors.

Disadvantages:

- Instability to noise, outliers in the initial data directly affect the construction of the separating hyperplane.

- No feature selection.

- It is necessary to select methods for constructing kernels and rectifying spaces separately for each task.

This algorithm is used in [3], where a student's grade for an exam in a given subject is predicted based on his grades in previous subjects, attendance, and midterm grades.

5. Neural Networks

In addition to the above methods, neural networks are also used to predict student performance. There are a number of references to the possibility of using neural networks to solve this problem, but there is no information about the actual implementation of such predictive models.

Neural networks are mathematical models based on the organizational and functional principles of biological neural networks. Neural networks are trained rather than programmed in the conventional sense. During training, neural networks are able to recognize and generalize complex relationships between input and output data. Once trained, the network is able to predict future values for a given sequence based on a series of past values.

An illustration of how a neural network works is shown in Figure 3. A neural network consists of neurons, layers, and synapses. Neurons are shown as nodes of different colors. All nodes of the same color belong to the same layer of the neural network. Synapses are lines that connect neurons in one layer to neurons in another layer. Synapses have only one parameter, the weight.

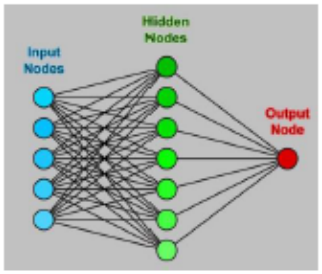


Fig. 3. Neural Network

Each neuron performs a specific mathematical function, so it receives a set of values as input and a single value as output. Thus, the output is a specific value produced by a previously trained neural network.

Advantages:

- Resistance to input noise.

- Self-learning and creativity. Ability to solve tasks that cannot be solved by other algorithms.

- Adaptation to changes, retraining.

Disadvantages:

- For large networks, it is impossible to estimate the network training time even approximately in advance.

- Difficulty in interpreting the result.

- Approximation of the obtained answer.

Рисунок В.4 – Четверта сторінка статті у журналі «Біоніка інтелекту»

Let's take a closer look at the use of a neural network to solve the problem of predicting student performance. In [4], a neural network model is trained to predict whether a student will be promising. The task of binary classification of applicants is solved using input data such as school number, grades in physics and math, and parents' occupations. In [5], a neural network is used to predict grades in a computer science course. Article [6] discusses the task of classifying students based on NMT results.

Thus, we have reviewed the algorithms most commonly used to solve the problem of predicting student performance based on different initial data and solving different problems, whether it is performance in a particular subject or a general picture of performance in all disciplines. The considered examples of prediction are not systematic, but are only attempts to come closer to solving this problem. Obviously, to solve this problem successfully, it is necessary to apply several methods and compare their results.

6. Data preprocessing

The quality of machine learning models is highly dependent on the quality of the underlying data. However, real-world data is often poorly structured, with missing values, noise, and incorrect values. If the data is not well prepared, no amount of tuning of machine learning algorithms can ensure high predictive accuracy of these models. Data preparation before analysis takes up about 80 percent of a machine learning specialist's time, but this work is necessary.

In this paper, the methodology described in [7] is used to prepare the data.

To familiarize ourselves with the data and prepare it for further analysis, we will use the Python programming language and its libraries. The choice of the Python programming language is due to its high performance in data processing, simplicity, and a large number of libraries for machine learning. Python is one of the best languages for working with data. The following Python libraries are used in this paper.

- NumPy. This library adds support for large multidimensional arrays and matrices, as well as high-level commands for mathematical functions with very high performance on these arrays [7].

- Pandas. A data processing and analysis software library.

- Seaborn. A data visualization library based on another Python library, Matplotlib.

- Scikit - learn. An open source library for machine learning.

- PyQt5 is a set of extensions to the Qt graphical framework for the Python programming language, made in the form of a Python extension. This library implements almost all the capabilities of Qt and allows you to create a graphical interface for programs written in Python.

First of all, in order to work with data using Python, you need to convert it into a format that this language and its libraries can work with. In this case, this format is DataFrame from the Pandas library.

After the conversion, you can familiarize yourself with the main characteristics of the original dataset.

You can see that some attributes in the tuple have null values, for greater clarity you should see which attributes they are in.

You can see that the attribute "Disciplines in which unsatisfactory grades were received" itself suggests the presence of "missings", while "Profile" and "Country" are most likely missing values as a result of filling the database.

Next, we will analyze the data in the simplest terms in order to understand the initial vector of data preparation.

First of all, let's look at the number of debtors (and therefore the quality, which we will evaluate by the number of debts).

In order to take into account this attribute (the number of unsatisfactory grades), it is necessary to further divide this range of attribute values into groups, since the difference between a student with no debts and one debt is greater than between students with 18 and 17 debts, where it is not essential.

It was decided to divide students into 3 groups according to the number of debts:

- Group 1 "successful" - 0 debts

- Group 2 "debts" - from 1 to 6 debts

- Group 3 "many debts" - from 7 and more (maximum 18 debts)

In order to facilitate the analysis of this criterion, an additional column was created in the dataset indicating the group to which the student belongs.

7. Machine learning model

Thus, after analyzing all the parameters, the following were identified as having a greater impact on student performance, namely

1. Type of study

2. Qualification

3. Course of studies

4. Specialty

5. Academic leave (valid) - and no / /.

6. Total hours of absence in the semester

7. Total hours of classes in the semester

These parameters will be used in machine learning with the teacher to classify the student according to the number of debts.

Let's look at the correlations between the selected parameters.

Рисунок В.5 – П’ята сторінка статті у журналі «Біоніка інтелекту»

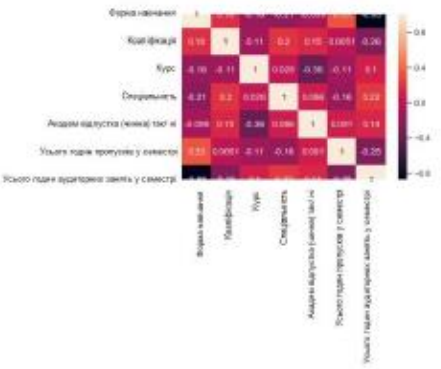


Fig. 4. Feature Correlation

To start training the model, we need to convert all the parameters into numerical representations. To do this, we will use the LabelEncoder function, which is already available in Python from the sklearn.preprocessing library.

After the conversion, we get the following: X is the data frame of all parameters, already converted to a numeric value, Y is the labels.

To start building the model, we divide our data into training and test samples.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
```

The first classifier used is logistic regression.

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=1000.0, random_state=0 )
lr.fit(x_train,y_train)
```

```
LogisticRegression(C=1000.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
print(lr.score(x_train, y_train))
print(lr.score(x_test, y_test))
```

```
0.6668915500084331
0.6336088154269972
```

```
pred_y = lr.predict(x_test)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print(metrics.classification_report(pred_y, y_test))
```

```
              precision    recall  f1-score   support

           0       0.84      0.60      0.70      1732
           1       0.48      0.74      0.58       433
           2       0.48      0.69      0.57       376

    accuracy                           0.63      2541
   macro avg       0.57      0.67      0.59      2541
weighted avg       0.71      0.63      0.65      2541
```

This method showed that there is some dependency and that the parameters were chosen correctly. Next, we will try other types of classification and compare the results.

Let's use the support vector method.

```
from sklearn import svm
clf = svm.SVC()
clf.fit(x_train,y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
print(clf.score(x_train, y_train))
print(clf.score(x_test, y_test))
```

```
0.8794865829777366
0.7197953561589925
```

```
pred_y = clf.predict(x_test)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print(metrics.classification_report(pred_y, y_test))
```

```
              precision    recall  f1-score   support

           0       0.86      0.68      0.76      1564
           1       0.54      0.85      0.66       421
           2       0.64      0.74      0.69       556

    accuracy                           0.72      2541
   macro avg       0.68      0.76      0.70      2541
weighted avg       0.76      0.72      0.73      2541
```

Here we can see that the support vector method is prone to overfitting, as there is a large 10 percent difference between the training and test sets in label detection. However, the result on the test set is almost 10 percent higher than the classifier based on logistic regression.

Using the 3rd Classifier "Random Forest"

```
from sklearn.ensemble import RandomForestClassifier
clas = RandomForestClassifier(max_depth=15, random_state=0)
clas.fit(x_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=15, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

```
print(clas.feature_importances_)
```

```
[0.01858244 0.03035815 0.12210830 0.2179611  0.07045300 0.31382953
 0.21770641]
```

```
print(clas.score(x_train, y_train))
print(clas.score(x_test, y_test))
```

```
0.8775510204081632
0.7898465171192444
```

```
pred_y = clas.predict(x_test)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print(metrics.classification_report(pred_y, y_test))
```

```
              precision    recall  f1-score   support

           0       0.82      0.78      0.80      1311
           1       0.74      0.85      0.79       579
           2       0.77      0.76      0.77       651

    accuracy                           0.79      2541
   macro avg       0.78      0.80      0.79      2541
weighted avg       0.79      0.79      0.79      2541
```

Рисунок В.6 – Шоста сторінка статті у журналі «Біоніка інтелекту»

Random Forest showed the best result, although the spread between the training and test samples is quite large (10 The accuracy of the test sample prediction is 79 percent. This is quite a high accuracy rate. Also, random forests with different maximum depths were tested, and experience showed that a depth greater than 15 does not significantly affect the test set prediction, which affects the accuracy.

Thus, we can conclude that the task of determining student performance based on such parameters as

- Type of study
- Qualification
- Course of study
- specialization
- Academic leave (valid) - and not /
- Total hours of absences in the semester
- Total hours of classes in the semester

The analysis also showed that the most significant contribution to the model is made by 3 parameters, namely

- Total absenteeism hours in the semester
- Total classroom hours in the semester
- Course.

In the future, additional parameters, such as the student's hobbies, participation in the social life of the university, etc., can also improve the accuracy of the model.

Conclusions

As a result of the research practice, the following tasks were accomplished:

1. Analysis of methods used to solve similar problems in the field of education.

2. Preliminary data processing was carried out.

3. Built a machine learning model capable of predicting student performance at the end of the semester.

4. Identified the most significant features in determining student performance.

5. Developed a graphical user interface for the student performance prediction model.

Since the preprocessing of the data used labels to indicate "student performance," the type of machine learning was supervised. Three supervised machine learning models were tested: logistic regression, support vector machine, and random forest. The random forest classifier performed best on the test sample. This algorithm is optimal because it has the following advantages.

In addition, a graphical interface was created for the module to predict student performance at the end of the semester based on current grades.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] В.О. Шевченко. Прогнозування успішності студентів на основі методів кластерного аналізу // Вісник ХНАДУ. 2015. №68. – С. 15-18.

[2] В.О. Шевченко. Інформаційна технологія формування індивідуальних траєкторій самостійної роботи студентів // Вісник НТУ "ХПІ» 2015. №21(1130). – С. 76-83.

[3] В.О. Шевченко, А.І. Кудін. Алгоритмізація процедури кластерного аналізу для прогнозування успішності студентів // Автомобіль і електроніка. Сучасні технології, 11/2017. – С. 64-67.

[4] O. Haitan, O. Nazarov. Hybrid approach to solving of the automated timetabling problem in higher educational institution // Системи управління, навігації та зв'язку, 2020, випуск 2(60). – С. 60-69.

[5] Al-Shehri H. et al. Student performance prediction using support vector machine and k-nearest neighbor //2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE). – IEEE, 2017. – С. 1-4.

[6] Breazley, D. Python Cookbook, Third Edition / D. Breasley, B. K. Jones. – USA: O'Reilly Media, 2013. – 688 p.

[7] McKinney, W. Python for Data Analysis. – USA: O'Reilly Media, 2013. – 453 p.

Рисунок В.7 – Сьома сторінка статті у журналі «Біоніка інтелекту»

Додаток Г

Слайди презентації

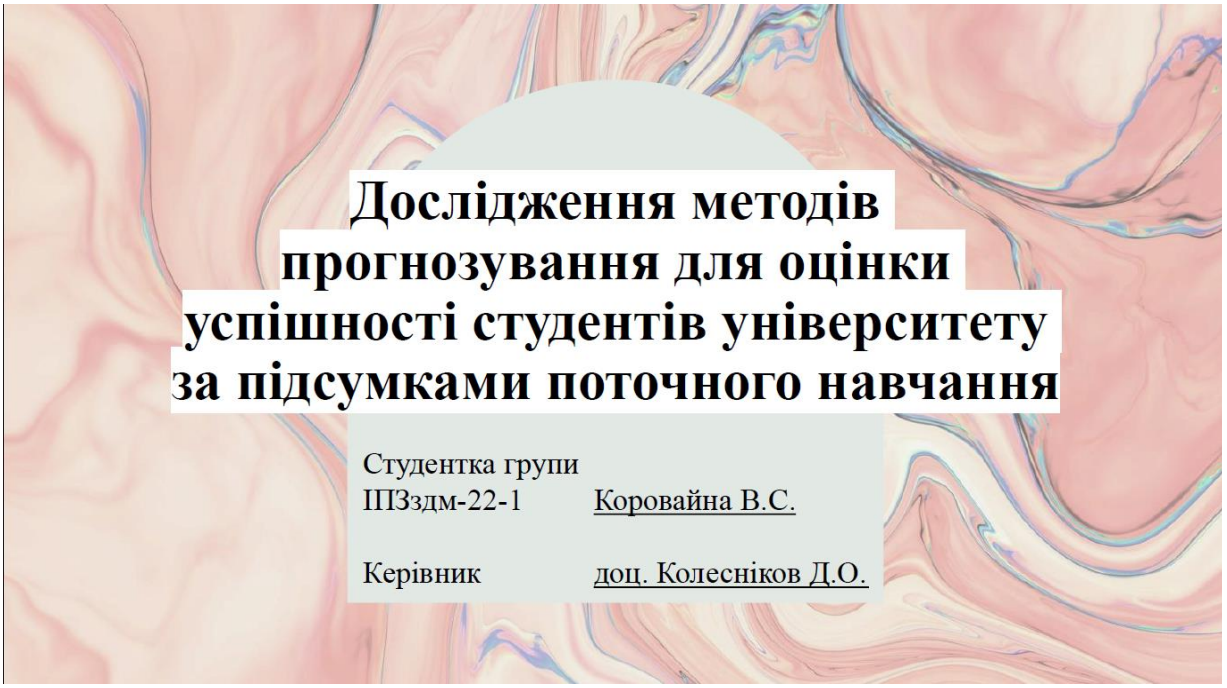


Рисунок Г.1 – Слайд 1 «Титульний»

Рисунок Г.2 – Слайд 2 «Аналіз предметної області»

ПОСТАНОВКА ЗАДАЧІ

Об'єктом дослідження є процес розробки прогнозної моделі для оцінки успішності студентів університету за підсумками поточного навчання.

Мета роботи – побудова прогнозної моделі підсумків сесії студентів залежно від оціночних параметрів поточної успішності.

У процесі дослідження необхідно провести аналіз основних проблем у цій галузі, поставити цілі для їх безпосереднього виконання. Також буде проведено попередню обробку даних для побудови моделі машинного навчання. Після цього буде побудовані різні моделі машинного навчання, а також оцінені якісні показники кожної моделі. Після вибору оптимальної моделі буде створено графічний інтерфейс користувача прогнозної моделі.

У результаті дослідження буде створено прогнозну модель успішності студентів університету, а також графічний інтерфейс для її використання. Крім того, будуть визначені найбільш значущі фактори під час прогнозування успішності у студентів.

Рисунок Г.3 – Слайд 3 «Постановка задачі»

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

Основні характеристики початкового датасета

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8551 entries, 0 to 8550
Data columns (total 24 columns):
Форма навчання                          8551 non-null object
Кваліфікація                            8551 non-null object
Курс                                    8551 non-null int64
Спеціальність                           8551 non-null object
Профіль                                 6676 non-null object
Випуск. відділ.                         8551 non-null object
Випуск. школа                           7988 non-null object
Група                                   8551 non-null object
Навч. підрозділ.                        8551 non-null object
ID                                      8551 non-null int64
Форма фінансування                      8551 non-null object
Країна                                  8533 non-null object
Громадянство                            8551 non-null object
Стать                                   8551 non-null object
Дата народження                         8551 non-null object
Академвідпустка (чинна) так / ні        8551 non-null object
Усього                                  8551 non-null int64
Позитивних                              8551 non-null int64
Незадовільних                           8551 non-null int64
Дисципліни з яких отримано незадовільні оцінки    6370 non-null object
Пропусків з дисциплін за якими отримано незадовільні оцінки    8551 non-null int64
Усього годин з дисциплін з яких отримано незадовільні оцінки    8539 non-null float64
dtypes: float64(2), int64(7), object(15)
```

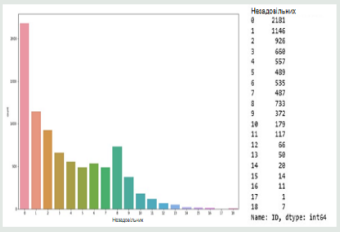
Наявність нульових значень у параметрів

```
Форма навчання                          False
Кваліфікація                            False
Курс                                    False
Спеціальність                           False
Профіль                                 True
Випуск. відділ.                         False
Випуск. школа                           True
Група                                   False
Навч. підрозділ.                        False
ID                                      False
Форма фінансування                      False
Країна                                  True
Громадянство                            False
Стать                                   False
Дата народження                         False
Академвідпустка (чинна) так / ні        False
Усього                                  False
Позитивних                              False
Незадовільних                           False
Дисципліни з яких отримано незадовільні оцінки    True
Пропусків з дисциплін з яких отримано незадовільні оцінки    False
Усього годин з дисциплін з яких отримано незадовільні оцінки    True
Усього годин пропусків у семестрі       False
Усього годин аудиторних занять у семестрі    True
```

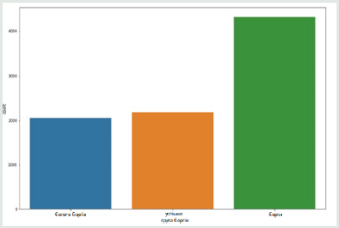
Рисунок Г.4 – Слайд 4 «Попередня обробка даних»

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

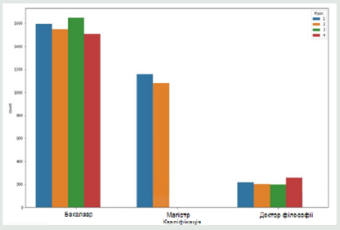
Кількість незадовільних оцінок



Розподіл студентів по "групах боргів"



Розподіл по кваліфікації і курсу навчання



Розподіл студентів по "групах боргів"

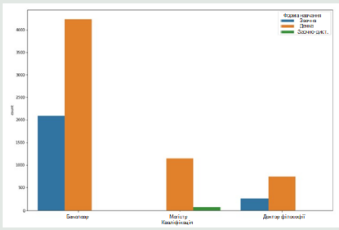


Рисунок Г.5 – Слайд 5 «Попередня обробка даних-2»

Рисунок Г.6 – Слайд 6 «Розподіл студентів по «групах боргів»»

Рисунок Г.7 – Слайд 7 «Розподіл студентів по «групах боргів»-2»

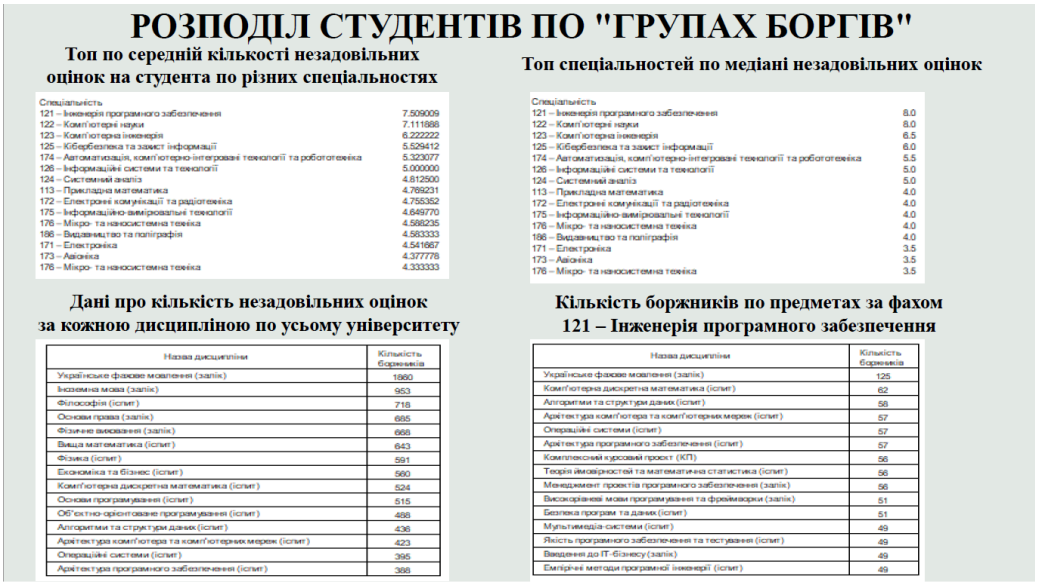


Рисунок Г.8 – Слайд 8 «Розподіл студентів по «групах боргів»-3»

Рисунок Г.9 – Слайд 9 «Модель машинного навчання»

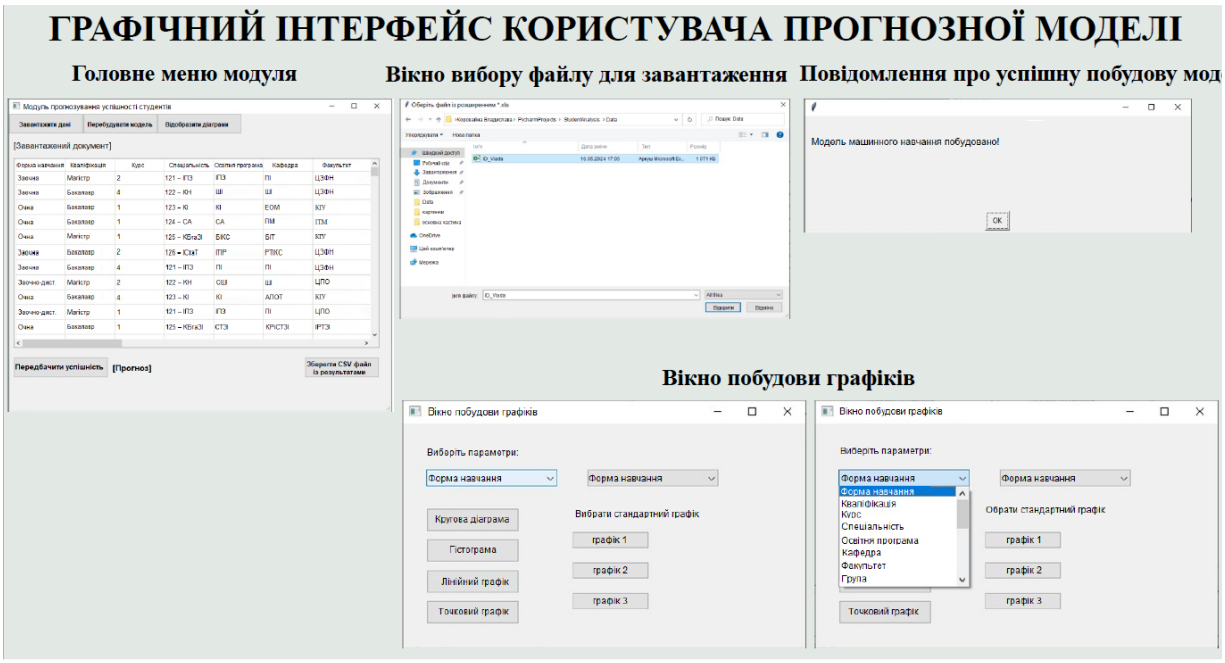


Рисунок Г.10 – Слайд 10 «Графічний інтерфейс користувача прогнозної моделі»

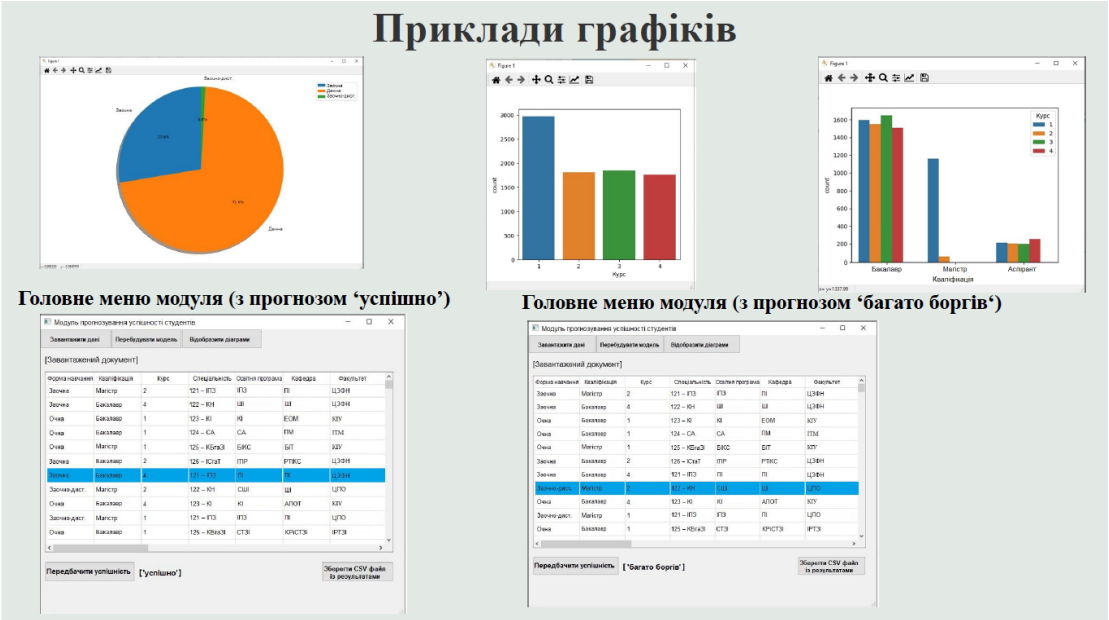


Рисунок Г.11 – Слайд 11 «Приклади графіків»

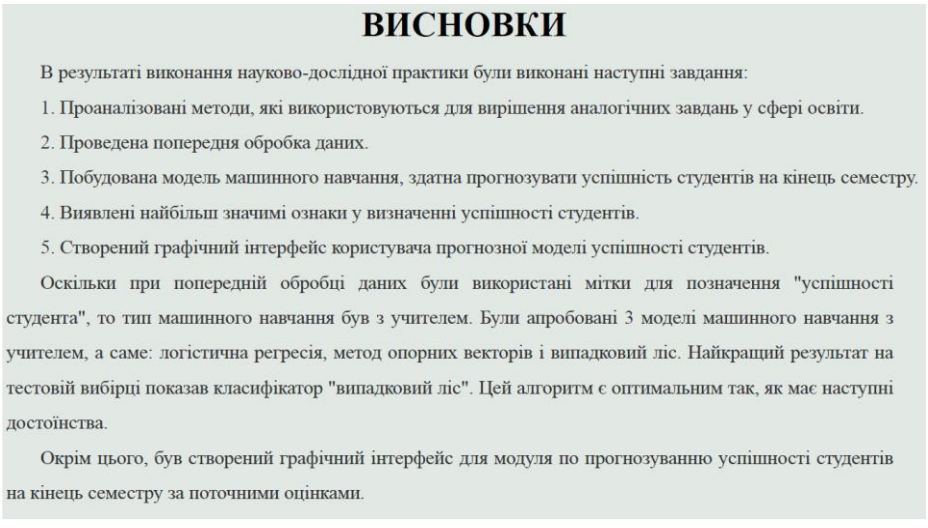


Рисунок Г.12 – Слайд 12 «Висновки»

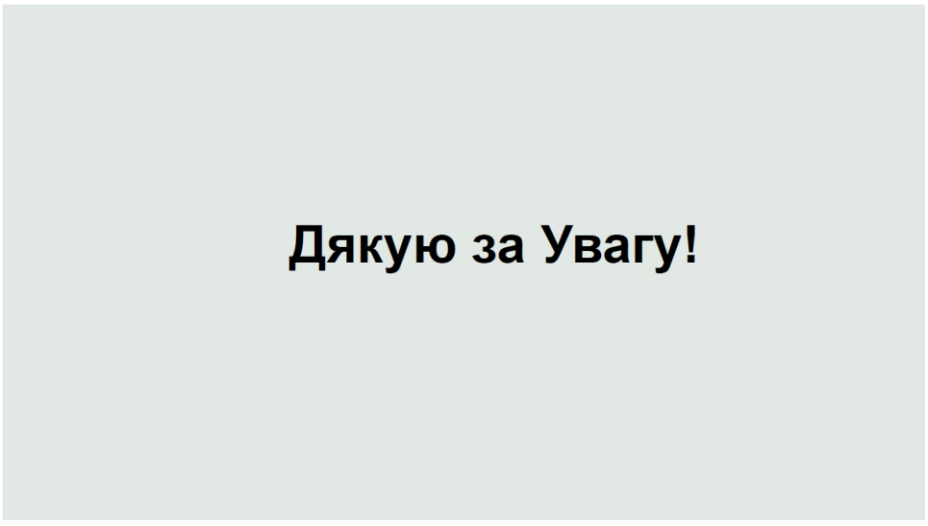


Рисунок Г.13 – Слайд 13 «Дякую за Увагу!»

Додаток Д

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗздм-22-1 (група)

Коровайна Владислава Сергіївна
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

08.06.2024