НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КУРСОВА РОБОТА

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Прогнозування наявності цукрового діабету у людини на основі медичних показників. Методи K-Nearest Neighbors та Decision Tree Classifier»

Студентки 2 курсу групи ІП-02
Спеціальності: 121
«Інженерія програмного забезпечення»
Литвин Анастасії Вячеславівни

«ПРИЙНЯВ» з оцінкою

_____

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

_____
Підпис Дата

Київ - 2022 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІП-02 Семестр 4

ЗАВДАННЯ

на курсову роботу студентки

Литвин Анастасії Вячеславівни

1.Тема роботи Прогнозування наявності цукрового діабету у людини на основі медичних показників. Методи K-Nearest Neighbors та Decision Tree Classifier.

2.Строк здачі студентом закінченої роботи 15.06.2022

3. Вхідні дані до роботи методичні вказівки до курсової роботи, обрані дані з сайту https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

Вступ, постановка задачі, аналіз предметної області, робота з даними, інтелектуальний аналіз, висновки, перелік посилань, додаток А.

5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6.Дата видачі завдання 16.04.2022

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	16.04.2022	
2.	Визначення зовнішніх джерел даних	25.04.2022	
3.	Пошук та вивчення літератури з питань курсової роботи	10.05.2022	
4.	Обробка та аналіз даних	15.05.2022	
5.	Обґрунтування методів інтелектуального аналізу даних	25.05.2022	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	5.06.2022	
7.	Підготовка пояснювальної записки	10.06.2022	
8.	Здача курсової роботи на перевірку	15.06.2022	
9.	Захист курсової роботи	16.06.2022	

Студент _____ Литвин А. В.

(підпис) (прізвище, ім’я, по батькові)

Керівник _____ доц. Ліхоузова Т.А

(підпис) (прізвище, ім’я, по батькові)

Керівник _____ доц. Олійник Ю.О.

(підпис) (прізвище, ім’я, по батькові)

"26" червня 2022 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 21 сторінки, 16 рисунків, 10 посилань.

Об’єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз даних з подальшим прогнозуванням та графічним відображенням результатів.

Мета роботи: пошук, аналіз та обробка даних, реалізація ПЗ, що використовує отримані дані для подальшого аналізу та прогнозування результату.

Дана курсова робота включає в себе: постановку задачі, аналіз предметної області, роботу з даними, аналіз обраних методів для прогнозування та їх порівняння.

ДАТАСЕТ, ПРОГНОЗУВАННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, K-NEAREST NEIGHBORS, DECISION TREE CLASSIFIER.

ЗМІСТ

ВСТУП

Зараз все більше стає запитаною медична діагностика різних захворювань, що є дуже важливим аспектом для лікарів та здоров’я пацієнтів. Аналізуючи захворювання, особливо хочеться виділити цукровий діабет, адже з розвитком рівня життя цукровий діабет все частіше зустрічається в повсякденні. Саме через це швидкість та точність діагностування цукрового діабету – завдання, що потребує швидких та точних результатів.

У рамках даної курсової роботи були проаналізовані дані лікарень з різними важливими характеристиками для визначення цукрового діабету та на основі отриманих даних було використано декілька методів для прогнозування даного захворювання.

Дана курсова робота буде розроблена з використанням технологій та бібліотек Python 3[1], Pandas[2], Seaborn[3], Matplotlib[4], Sklearn[5].

1 ПОСТАНОВКА ЗАДАЧІ

Виконання даної курсової роботи потребує виконання декількох задач: аналізу предметної області; роботи з датасетом: завантаження, дослідження його структури та виправлення наявних помилок; вибір методів для прогнозування та обґрунтування даного вибору; аналіз отриманих результатів кожного з методів та порівняння отриманих результатів ефективності.

Створення застосунку, що поділяє отримані дані на тренувальні та тестові для перевірки декількох методів, у подальшому порівняння ефективності методів.

Для прогнозування буде використано методи K-Nearest Neighbors та Decision Tree Classifier. Для кожного методу проаналізувати результати та в кінці порівняти результати цих двох методів. Обрати найоптимальніший метод для прогнозування хвороби.

Виконане дослідження можна буде використовувати для прогнозування можливої хвороби цукрового діабету.

Вхідними даними будуть кількість вагітностей, що мала жінка, кількість глюкози в крові, кров’яний тиск, товщина шкіри, інсуліну в крові, ІМТ, функція, яка оцінює ймовірність діабету на основі сімейного анамнезу, вік та чи хворіє людина чи ні.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Цукровий діабет – це хронічне захворювання, яке характеризується високим рівнем цукру в крові. Воно може спричинити багато складних захворювань, таких як інсульт, ниркова недостатність, серцевий напад тощо. У 2014 році в усьому світі цукровим діабетом хворіло приблизно 422 мільйонів людей. У 2040 році ця цифра зможе сягнути 642 мільйонів, люди часто не замислюються про шкоду неправильного харчування, тому ми і отримуємо такі результати. Навіть випиті чашки кави та чаю з парою ложок цукру за день можуть шкідливо вплинути на ваше здоров’я, адже норма цукру на день знаходиться в межах 6-9 чайних ложок[6]. Так, дійсно, не лише цукор викликає діабет, але якщо мати раціон з високим вмістом цукру, то він буде збільшувати ризик розвитку діабету.

Для виявлення діабету пацієнту доводиться пройти кілька аналізів - визначення рівня глюкози в крові натще, толерантності до глюкози та випадкових рівнів глюкози в крові[7], потім фахівцям треба відстежувати безліч факторів під час процесу діагностики, що є досить важким і довгим заняттям, це у свою чергу може призвести до неточних результатів, що робить виявлення хвороби дуже складним та довгим процесом. Чим раніше буде поставлений діагноз, тим легше його зможуть контролювати.

Завдяки найбільш передовим технологіям, що використовуються для швидкого та точного прогнозування захворювання в галузі охорони здоров’я, цей процес можна прискорювати та покращувати, тому основною метою нашої роботи є зробити вклад у покращення здоров’я населення.

У програмному забезпеченні буде реалізовано наступну функціональність, що включає в себе:

- завантаження та дослідження структури датасету;
- використання декількох моделей прогнозування даних;
- прогнозування за характеристиками наявність діабету у людини;
- відображення отриманих результатів та їх аналіз;
- порівняння використаних методів.

3 РОБОТА З ДАНИМИ

3.1 Опис обраних даних

Для вирішення поставленної перед нами задачі був обраний «Pima Indians Diabetes Database». Даний набір даних складається з аналізів 768 пацієнтів, які є жінками віком від 21 року. Даний датасет містить в собі одну таблицю, що складається з 9 стовпців, а саме: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome. Дані стовпці несуть в собі наступну інформацію:

- Pregnancies – кількість вагітностей, що мала жінка;
- Glucose – кількість глюкози в крові;
- BloodPressure – кров'яний тиск;
- SkinThickness – товщина шкіри;
- Insulin – кількість інсуліну в крові;
- BMI – розшифровується, як Body Mass Index – величина, що рахується за формулою $\frac{\text{вага}}{\text{ріст}^2}$;
- DiabetesPedigreeFunction - вказує функцію, яка оцінює ймовірність діабету на основі сімейного анамнезу;
- Age – вік жінки;
- Outcome – вказує на те, чи хворіє жінка на діабет (0 – ні, 1 – так).

3.2 Перевірка даних

Для роботи з даними на мові Python ми використовуємо бібліотеку «pandas».

Для початку ми зчитуємо дані з файлу та виводимо основну інформацію про наш датафрейм (рис. 3.1).

Inputting data

```
[2]: dataset = pd.read_csv("data/diabetes.csv")

[3]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Рисунок 3.1 – Загальна інформація про датафрейм

Як можемо побачити, всі дані non-null та типи даних правильно визначилися, що вже дуже добре для нас.

Також переглянемо який вид мають наші дані отражу після завантаження їх в датафрейм. Виведемо перші 5 рядків (рис. 3.2).

```
[4]: dataset.head(5)
```

[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Рисунок 3.2 – Отриманий датафрейм

Наступним кроком перевіримо написанними нами функціями, чи зустрічаються неправильні дані. Будемо перевіряти, чи менші наші дані за нуль (рис. 3.3).

Checking data for the broken values (numbers below zero)

```
[4]: columnsToCheckUnderZero = list(dataset.columns.values)
     for i in columnsToCheckUnderZero:
         if not check_values_under_zero(dataset, i):
             print("Values under 0 not found in", i)
```

```
Values under 0 not found in Pregnancies
Values under 0 not found in Glucose
Values under 0 not found in BloodPressure
Values under 0 not found in SkinThickness
Values under 0 not found in Insulin
Values under 0 not found in BMI
Values under 0 not found in DiabetesPedigreeFunction
Values under 0 not found in Age
Values under 0 not found in Outcome
```

Рисунок 3.3 – Перевірка даних на те, чи зустрічаються числа менші за нуль

І знову нам щастить! Дані в гарному стані та не зустрічаються “зламані” числа. Наступним кроком перевіримо, чи зустрічаються дані, що рівні нулю, а саме

в колонках: Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedictionFunction, Age. Якщо зустрінеться нуль, то ми виводимо інформацію, що в даній колонці знайшовся нуль та відразу замінюємо його на середні значення (рис 3.4).

Check columns for presenting 0 in it and if it exists change it to average

```
[5]: columnsToCheckForZero = list(dataset.columns.values)[1:7]

[6]: for i in columnsToCheckForZero:
         dataset = zero_to_average(dataset, i)

Found 0 in column: Glucose
Found 0 in column: BloodPressure
Found 0 in column: SkinThickness
Found 0 in column: Insulin
Found 0 in column: BMI
```

Рисунок 3.4 – Перевірка на нуль та заміна на середні значення

Після виконаних нами маніпуляцій з даними отримаємо датафрейм з яким будемо працювати далі, перевіримо який вигляд він має зараз: виведемо перші 10 пацієнтів (рис 3.5).

Resulting dataframe

```
[7]: dataset.head(10)
```

[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	79	33.60	0.627	50	1
1	1	85	66	29	79	26.60	0.351	31	0
2	8	183	64	20	80	23.30	0.672	32	1
3	1	89	66	23	94	28.10	0.167	21	0
4	0	137	40	35	168	43.10	2.288	33	1
5	5	116	74	20	80	25.60	0.201	30	0
6	3	78	50	32	88	31.00	0.248	26	1
7	10	115	69	20	80	35.30	0.134	29	0
8	2	197	70	45	543	30.50	0.158	53	1
9	8	125	96	20	80	31.99	0.232	54	1

Рисунок 3.5 – Датафрейм для подальшої роботи

Переглянемо яка кількість пацієнтів з датасету хворіє на цукровий діабет, а яка кількість здорових пацієнтів (рис. 3.6).

Comparison of sick and healthy patients

```
[10]: sns.countplot(x="Outcome", data=dataset, palette = "Greens")
      plt.show()
```

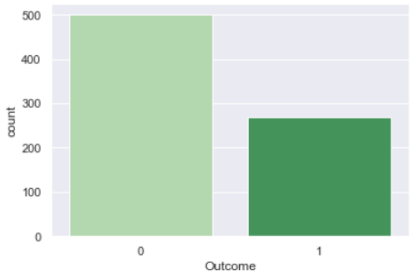

Рисунок 3.6 – Відображення кількості здорових та хворих пацієнтів

З графіка на рисунку 3.6 бачимо, що ми маємо більше здорових(0), ніж хворих(1) пацієнтів.

3.3 Поділ даних

Останнім кроком ми ділимо дані на тренувальні та тестові для подальшої роботи з методами класифікації (рис 3.7).

Now we need to split data for info and result

```
[8]: from sklearn.model_selection import train_test_split
     X = dataset.drop(columns='Outcome')
     y = dataset['Outcome']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=0)
```

Рисунок 3.7 – Поділ інформації на інформацію та результат

Щоб уникнути оверфіту, ми розділили наш набір даних на навчальні та тестові, а саме на 80% тренувальних та 20% даних, на яких буде проводитися тестування. Це дасть нам краще уявлення про те, як наші методи працюють на етапі тестування. Таким чином наші методи тестуються на невидимих даних, для кращого розуміння їх коректності використання для розв’язання поставленої нами задачі.

4 ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Обґрунтування вибору методів інтелектуального аналізу даних

Мною було обрано два методи для прогнозування а в подальшому і порівняні їх – це методи K-Nearest Neighbors та Decision Tree Classifier.

Метод K-Nearest Neighbors мене зацікавив спершу, бо це алгоритм лінивого навчання і тому не вимагає підготовки перед прогнозуванням в реальному часі. Це робить алгоритм KNN набагато швидшим, ніж інші алгоритми, які потребують навчання, наприклад, SVM, лінійна регресія, звідси алгоритм KNN не вимагає підготовки перед прогнозуванням, нові дані можна додавати без проблем.

Одним з завданням для вибору параметрів залишається лише вибір K-найближчих точок – сусідів, звідси метод призначає точку даних до класу, до якого належить більшість K точок даних. Саме ця робота методу зацікавила мене найбільше, звичайно, що всі люди різні та мають різні характеристики здоров’я, але, на мою думку, є досить багато людей у яких є схожі результати аналізів, звідси було б досить добре шукати схожі точки-даних, щоб прогнозувати наявність чи відсутність хвороби у заданого пацієнта.

Також мінусом методу KNN є погана робота з даними великої розмірності(при великій кількості вимірів алгоритму стає важко обчислити відстань у кожному вимірі) та високу вартість прогнозування для великих наборів даних(великих наборах даних вартість розрахунку відстані між новою точкою та кожною існуючою точкою стає вищою) та алгоритм KNN погано працює з даними великої розмірності, оскільки при великій кількості вимірів алгоритму стає важко обчислити відстань у кожному вимірі[8], у нашому ж випадку ми маємо 8 характеристик, що є досить невеликим числом та близько 600 прикладів, на яких буде натренована модель.

Щодо методу Decision Tree Classifier, то його я вже підбирала саме як метод для порівняння з KNN, бо мені було цікаво побачити який з методів буде більше підходити для нашого завдання.

Для кожного атрибута в наборі даних алгоритм дерева рішень формує вузол, де найважливіший атрибут розміщується в кореневому вузлі. Для оцінки ми починаємо з кореневого вузла і рухаємося вниз по дереву, дотримуючись відповідного вузла, який відповідає нашій умові або «рішенню». Цей процес триває до тих пір, поки не буде досягнутий листовий вузол, який містить прогноз. Так як ми маємо числові характеристики, то буде досить доцільно використати також і заданий метод для нашої задачі[9].

Також нам відомо, що дерева рішень добре працюють як для задач регресії, так і для задач класифікації, до яких відноситься і наша задача. Даний метод також дуже швидкі та ефективний, якщо порівнювати його з іншими алгоритмами класифікації, тому ми будемо його використовувати.

4.2 Аналіз отриманих результатів для методу K-Nearest Neighbors

Для початку, щоб аналізувати результати роботи методу KNN, необхідно обрати найкращу з можливих модельок, для цього ми використали модуль, що перебирає всі можливі варіанти параметрів та дає зрозуміти, які параметри найкраще вирішують поставлену задачу (рис. 4.1).

```
[15]: from sklearn.neighbors import KNeighborsClassifier
      from sklearn.model_selection import GridSearchCV
      #Пошук та підбір найкращого з параметрів для методу
      clasificator = KNeighborsClassifier()
      parameters={'n_neighbors': range(1, 25)}
      gridsearch=GridSearchCV(clasificator, parameters, cv=10, verbose=1)
      gridsearch.fit(X_train, y_train)
      print("Best parametrs - ", gridsearch.best_estimator_)

      Fitting 10 folds for each of 24 candidates, totalling 240 fits
      Best parametrs -  KNeighborsClassifier(n_neighbors=16)
```

Рисунок 4.1 – Пошук найкращих параметрів для методу KNN

Для заданого розміру даних для тренування найкращою кількістю модель з кількістю сусідів, що рівна 16, тому в подальшому будемо використовувати саме її.

Після цього підгонимо модель під тренувальні дані та перевіримо точність прогнозування отриманої модельки (рис 4.2).

```
[16]: #Присвоюємо найкращу модель для подальшої роботи з нею
knn = gridsearch.best_estimator_
knn.fit(X_train, y_train)
print("Accuracy of K-NN classifier on training data: ", round(knn.score(X_train, y_train), 5))
print("Accuracy of K-NN classifier on test data: ", round(knn.score(X_test, y_test), 5))

Accuracy of K-NN classifier on training data:  0.78013
Accuracy of K-NN classifier on test data:  0.77273
```

Рисунок 4.2 – Результати точності отриманої модельки для KNN

З результатів бачимо, що ми отримали достатньо високу точність прогнозування хвороби за використання методу KNN на тестових даних та низьку точність для тренувальних даних, якщо враховувати, що саме на цих даних навчалася модель.

Також перевіримо прогноз для яких інтервалів параметрів виявився не правильним побудувавши графік зображений на рисунку 4.3:

```
[13]: plt.rcParams["figure.figsize"] = (15,4)
plt.gca().axes.get_yaxis().set_visible(False)
plt.plot(X_test.index, y_test, "yx", label = "True result")
plt.plot(X_test.index, knn.predict(X_test), "b+", label = "Predict result")
plt.legend(loc="center right", shadow=True)
plt.show()
```

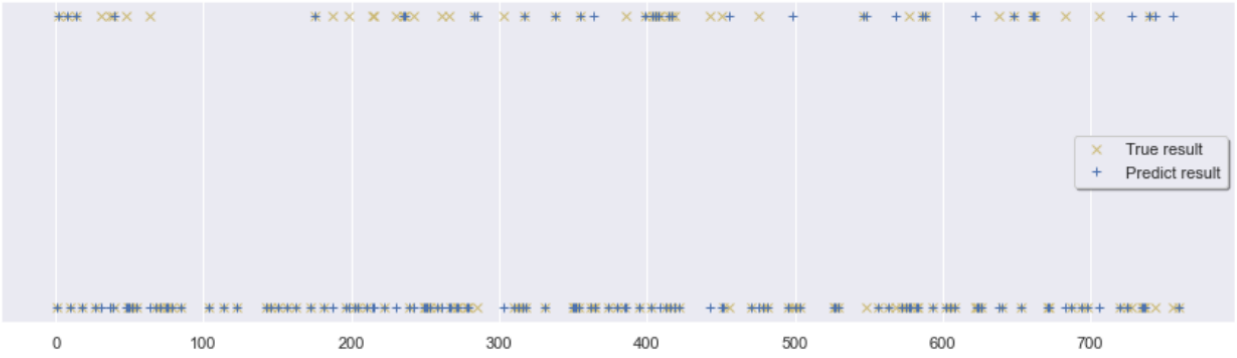

Рисунок 4.3 – Прогнози представлені методом KNN на інтервалах

З графіка добре видно сині плюси, які не наїжджають на жовті хрестики та жовті хрестики, що не закриваються синіми плюсами – це означає, що саме в цих точках модель допустила помилку.

Один з методів перевірки продуктивності роботи модельки є матриця невідповідностей[10]. Приклад такої матриця зображено на рисунку 4.4.

```
[14]: from sklearn.metrics import classification_report, plot_confusion_matrix
      plot_confusion_matrix(knn, X_test, y_test, cmap = "Greens")
      plt.grid(False)
```

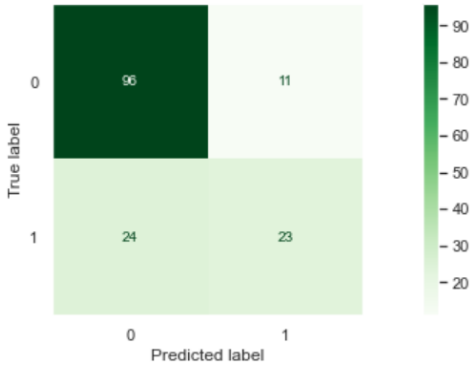

Рисунок 4.4 – Матриця невідповідностей для KNN

Дана матриця побудована за допомогою бібліотеки Sklearn. Верхня ліва клітинка відображає кількість істинно позитивних визначень моделі – прогнозувало, що людина здорова, коли вона і є здорова. Верхня права клітинка відображає кількість хибно позитивних визначень моделі - прогнозувало, що людина хвора, коли вона здорова. Нижня ліва клітинка відображає кількість хибно негативних визначень моделі - прогнозувало, що людина хвора, коли вона і здорова. Нижня права клітинка відображає кількість істинно негативних визначень моделі - прогнозувало, що людина хвора, коли вона і є хвора. Як видно з рисунку 4.4, модель в більшості випадків досить точно визначала наявність хвороби та її відсутність за заданими параметрами.

4.3 Аналіз отриманих результатів для методу Decision Tree Classifier

Для аналізу методу DTC підгонимо модель під тренувальні дані та перевіримо точність прогнозування отриманої модельки (рис 4.5).

```
[20]: from sklearn.tree import DecisionTreeClassifier
      dtc = DecisionTreeClassifier(random_state=0)
      dtc.fit(X_train, y_train)
      print("Accuracy of DT classifier on training data: ", round(dtc.score(X_train, y_train), 5))
      print("Accuracy of DT classifier on test data: ", round(dtc.score(X_test, y_test), 5))

      Accuracy of DT classifier on training data:  1.0
      Accuracy of DT classifier on test data:  0.78571
```

Рисунок 4.5 – Результати точності отриманої модельки для DTC

З результатів бачимо, що ми отримали достатньо високу точність прогнозування хвороби за використання методу DTC на тестових даних та ідеальну стовідсоткову точність для тренувальних даних.

Також перевіримо прогноз для яких інтервалів параметрів виявився не правильним побудувавши графік зображений на рисунку 4.6:

```
[16]: plt.rcParams["figure.figsize"] = (15,4)
      plt.gca().axes.get_yaxis().set_visible(False)
      plt.plot(X_test.index, y_test, "yx",  label = "True result")
      plt.plot(X_test.index, dtc.predict(X_test), "b+", label = "Predict result")
      plt.legend(loc="center right", shadow=True)
      plt.show()
```

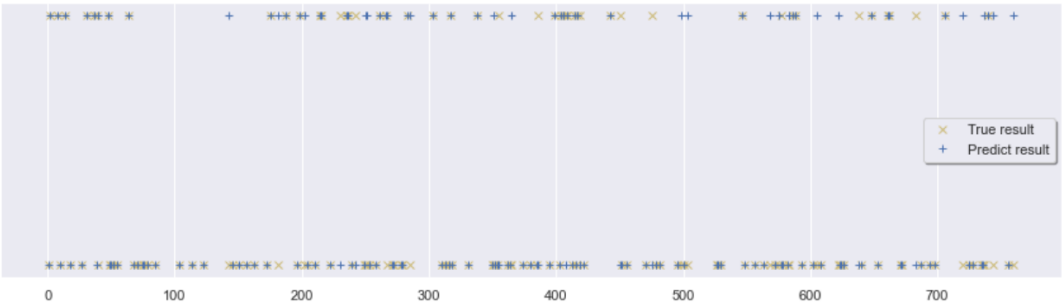

Рисунок 4.6 – Прогнози представлені методом DTC на інтервалах

З графіка добре видно сині плюси, які не наїжджають на жовті хрестики та жовті хрестики, що не закриваються синіми плюсами – це означає, що саме в цих точках модель допустила помилку.

Один з методів перевірки продуктивності роботи модельки є матриця невідповідностей. Ще один приклад такої матриця зображено на рисунку 4.7.

```
[17]: from sklearn.metrics import classification_report, plot_confusion_matrix
      plot_confusion_matrix(dtc, X_test, y_test, cmap = "Greens")
      plt.grid(False)
```

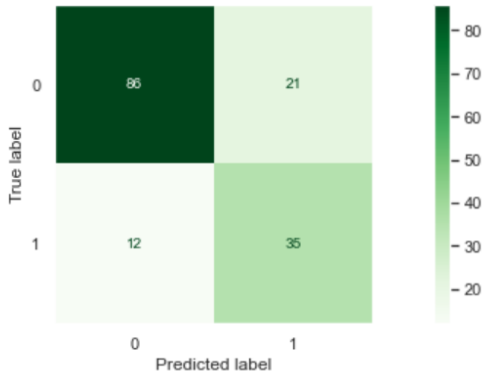

Рисунок 4.7 – Матриця невідповідностей для DTC

Дана матриця побудована за допомогою бібліотеки Sklearn. Верхня ліва клітинка відображає кількість істинно позитивних визначень моделі – прогнозувало, що людина здорова, коли вона і є здорова. Верхня права клітинка відображає кількість хибно позитивних визначень моделі - прогнозувало, що людина хвора, коли вона здорова. Нижня ліва клітинка відображає кількість хибно негативних визначень моделі - прогнозувало, що людина хвора, коли вона і здорова. Нижня права клітинка відображає кількість істинно негативних визначень моделі - прогнозувало, що людина хвора, коли вона і є хвора. Як видно з рисунку 4.7, модель в більшості випадків досить точно визначала наявність хвороби та її відсутність.

4.4 Порівняння отриманих результатів методів

Проаналізувавши окремо кожен із методів, що були мною використані під час прогнозування захворюваності на підготовлених даних, варто провести порівняння даних методів.

Порівняння точності зображене на рисунку 4.8:

Рисунок 4.8 – Порівняння точності методів KNN та DTC

Отже, маємо що максимальної точності прогнозування в даних, на яких тренувалися ці методи, досяг DTC – це 100%, порівнюючи його з методом KNN, що досяг лише 78%, то отримаємо, що метод KNN плутається навіть на даних, що були використані для його тренування.

Також якщо порівнювати вже роботу натренованих методів на даних для тестування, то знову бачимо, що хоча б на 1%, але метод DTC виявився краще методу KNN.

Щоб остаточно розуміти яку ж краще модель використовувати для подальшого прогнозування, то переглянемо результуючі матриці невідповідностей кожного з методів та порівняймо кожну комірку отриманих матриць (рис 4.9).

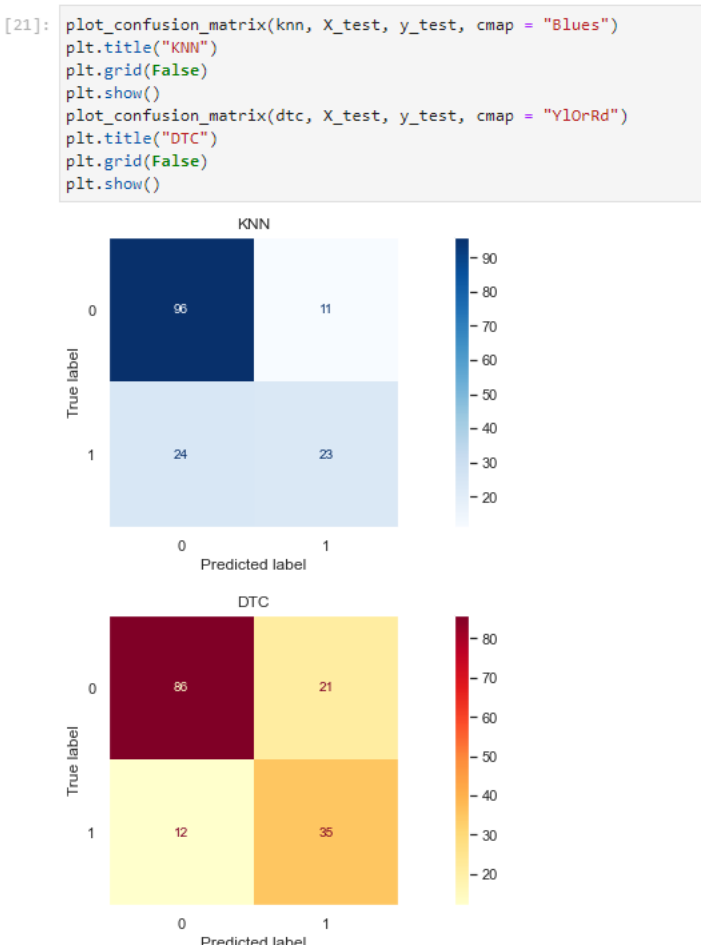

Рисунок 4.9 – Порівняння точності методів KNN та DTC за матрицями невідповідностей

Отже, маємо що метод краще KNN справляється з прогнозуванням для здорових пацієнтів, що вони здорові, ніж метод DTC, але DTC у свою чергу краще справляється з прогнозуванням хвороби для хворих пацієнтів. Також метод KNN досить часто вказує, що хворі пацієнти здорові, а DTC навпаки – що здорові пацієнти є хворими.

ВИСНОВКИ

Тема захворювання цукрового діабету досить важлива, як і тема будь-якого іншого серйозного захворювання, тому її варто підіймати та намагатися якомога швидше вирішувати.

Ми реалізували програмне забезпечення, що тренує моделі двох методів - K-Nearest Neighbors та Decision Tree Classifier, та під час порівняння результатів допомагає визначити який з цих двох методів краще обрати.

Нами була виконана робота з датасетом, у ході якої ми дослідили його структуру та виправили наявні помилки.

Також обґрунтовано вибір двох методів для прогнозування результату та проаналізовано результати кожного з методів окремо.

Аналізуючи отримані результати у кожному з пунктів, можна дійти висновків, що все таки кращою для прогнозування буде модель Decision Tree, її точність є ненабагато високою (лише на 1%), але також варто згадати результати, що ми отримали за допомогою матриць невідповідностей – з них видно, що метод KNN більше вказує, що хворі пацієнти здорові, а DTC– що здорові пацієнти є хворими, на мою думку, що краще, якщо неправильні результати здорових людей перевірять ще раз, ніж для неправильні результати хворих залишать без уваги.

Отже, для подальшого прогнозування захворювання цукрового діабету варто обрати метод Decision Tree Classifier, адже його характеристики є більш коректними для роботи з даними про здоров’я населення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація мови програмування Python. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/

2. Бібліотека Pandas. [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/docs/

3. Бібліотека Seaborn. [Електронний ресурс] – Режим доступу до ресурсу: https://seaborn.pydata.org/introduction.html

4. Бібліотека Matplotlib. [Електронний ресурс] – Режим доступу до ресурсу: https://matplotlib.org/stable/

5. Бібліотека Sklearn. [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/user_guide.html

6. American Heart Association Nutrition Committee of the Council on Nutrition, Physical Activity, and Metabolism and the Council on Epidemiology and Prevention. Dietary sugars intake and cardiovascular health [Електронний ресурс] / American Heart Association Nutrition Committee of the Council on Nutrition, Physical Activity, and Metabolism and the Council on Epidemiology and Prevention. – 2009. – Режим доступу до ресурсу: https://pubmed.ncbi.nlm.nih.gov/19704096/.

7. Iancu I. Method for the analysing of blood glucose dynamics in diabetes mellitus patients [Електронний ресурс] / Iancu I., Mota M., Iancu E.. – 2008. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/4588883.

8. K-Nearest Neighbors Algorithm in Python and Scikit-Learn [Електронний ресурс] – Режим доступу до ресурсу: https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/.

9. Decision Trees in Python with Scikit-Learn [Електронний ресурс] – Режим доступу до ресурсу: https://stackabuse.com/decision-trees-in-python-with-scikit-learn/.

10. Confusion Matrix in Machine Learning. [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду прогнозування наявності цукрового діабету у людини на основі медичних показників. Методи K-Nearest Neighbors та Decision Tree Classifier

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

(Обсяг програми (документа), арк.)

Студентки групи ІП-02 2 курсу

Литвин А. В.

main.py

```
#!/usr/bin
/env python
# coding: utf-8

# In[1]:


from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, plot_confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from functions import *
import warnings
warnings.filterwarnings(action='ignore', category=FutureWarning)
warnings.filterwarnings(action='ignore', category=DeprecationWarning)
sns.set(color_codes=True)


# # Inputting data

# In[2]:


dataset = pd.read_csv("data/diabetes.csv")


# In[3]:


dataset.info()


# In[4]:


dataset.head(5)


# #### Checking data for the broken values (numbers below zero)

# In[5]:


columnsToCheckUnderZero = list(dataset.columns.values)
for i in columnsToCheckUnderZero:
    if not check_values_under_zero(dataset, i):
        print("Values under 0 not found in", i)
```

```
# #### Check columns for presenting 0 in it and if it exists change it to average

# In[6]:


columnsToCheckForZero = list(dataset.columns.values)[1:7]


# In[7]:


for i in columnsToCheckForZero:
    dataset = zero_to_average(dataset, i)


# #### Resulting dataframe

# In[8]:


dataset.head(10)


# ## Now we need to split data for info and result

# In[9]:


X = dataset.drop(columns='Outcome')
y = dataset['Outcome']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, train_size=0.8, random_state=0)


# ##### Comparison of sick and healthy patients

# In[10]:


sns.countplot(x="Outcome", data=dataset, palette="Greens")
plt.show()


# ## K-Nearest Neighbors

# In[11]:


# Пошук та підбір найкращого з параметрів для методу
clasificator = KNeighborsClassifier()
```

```
parameters = {'n_neighbors': range(1, 25)}
gridsearch = GridSearchCV(clasificator, parameters, cv=10, verbose=1)
gridsearch.fit(X_train, y_train)
print("Best parametrs - ", gridsearch.best_estimator_)


# In[12]:


# Присвоюєму найкращу модель для подальшої роботи з нею
knn = gridsearch.best_estimator_
knn.fit(X_train, y_train)
print("Accuracy of K-NN classifier on training data: ",
      round(knn.score(X_train, y_train), 5))
print("Accuracy of K-NN classifier on test data: ",
      round(knn.score(X_test, y_test), 5))


# In[13]:


plt.rcParams["figure.figsize"] = (15, 4)
plt.gca().axes.get_yaxis().set_visible(False)
plt.plot(X_test.index, y_test, "yx", label="True result")
plt.plot(X_test.index, knn.predict(X_test), "b+", label="Predict result")
plt.legend(loc="center right", shadow=True)
plt.show()


# In[14]:


plot_confusion_matrix(knn, X_test, y_test, cmap="Greens")
plt.grid(False)


# ## Decision Tree Classifier

# In[15]:


dtc = DecisionTreeClassifier(random_state=0)
dtc.fit(X_train, y_train)
print("Accuracy of DT classifier on training data: ",
      round(dtc.score(X_train, y_train), 5))
print("Accuracy of DT classifier on test data: ",
      round(dtc.score(X_test, y_test), 5))


# In[16]:
```

```
plt.rcParams["figure.figsize"] = (15, 4)
plt.gca().axes.get_yaxis().set_visible(False)
plt.plot(X_test.index, y_test, "yx",  label="True result")
plt.plot(X_test.index, dtc.predict(X_test), "b+", label="Predict result")
plt.legend(loc="center right", shadow=True)
plt.show()


# In[17]:


plot_confusion_matrix(dtc, X_test, y_test, cmap="Greens")
plt.grid(False)


# ### Compare two methods

# In[18]:


print("Accuracy on training data for ")
print("K-NN classifier : ", int(knn.score(X_train, y_train)*100), '%',
      "\tDT classifier : ", int(dtc.score(X_train, y_train)*100), '%')
print("\nAccuracy on test data for ")
print("K-NN classifier : ", int(knn.score(X_test, y_test)*100), '%',
      "\tDT classifier : ", int(dtc.score(X_test, y_test)*100), '%')


# In[19]:


plot_confusion_matrix(knn, X_test, y_test, cmap="Blues")
plt.title("KNN")
plt.grid(False)
plt.show()
plot_confusion_matrix(dtc, X_test, y_test, cmap="YlOrRd")
plt.title("DTC")
plt.grid(False)
plt.show()
```

functions.py

```
def zero_to_average(dataset, column):
    res = False
    for i, row in dataset.iterrows():
        if row[column] == 0:
            dataset.at[i, column] = round(dataset[column].mean(), 2)
            res = True


    if res:
```

```
        print("Found 0 in column:", column)

    return dataset


def check_values_under_zero(dataset, column):
    for i, row in dataset.iterrows():
        if row[column] < 0:
            print("Found value below 0 in column ", column)
            return True
    return False
```