Прогнозування наявност і серцево-судинних хвороб у людини на основі медичних показників

Презентацію підготував
Студент групи ІП-11
Панченко Сергій

Мета

- Мета даної роботи полягає у визначенні найкращого методу для діагностування серцево-судинного захворювання (ССЗ) у пацієнта.
- Треба проаналізувати результати декількох алгоритмів на медичних даних пацієнтів різного віку, статі, рівню холестеролу та глюкози, шкідливих звичок (куріння, вживання алкоголю).
- Дане дослідження має на меті привернути увагу читачів до проблем, пов’ язаних з ССЗ.

- За даними Всесвітньої організації охорони здоров'я (ВООЗ), в 2019 році
- ССЗ стали причиною смерті для більше ніж 17 мільйонів людей, що становить
- більше 30% від загальної кількості смертей в світі. Від ІХС та ЦВЗ померли понад
- 13 мільйонів людей, що становить більше 80% від усіх смертей від ССЗ.

Використана література

1. Python — https://docs.python.org/3/
1. Pandas — https://pandas.pydata.org/docs/
2. Seaborn — https://seaborn.pydata.org/introduction.html
4. Matplotlib — https://matplotlib.org/stable/
5. Sklearn — https://scikit-learn.org/stable/user_guide.html
6. NumPy — https://numpy.org

Обгрунтування обраних методів

K-Nearest Neighbors

- Алгоритм лінивого навчання
- Легко інтерпритується
- Дані можна додавати в будь-який час

Random Forest

- Вирішує задачі регресії та класифікації
- Опрацьовує дискретні та неперервні змінні
- Стійкий до шуму
- Не потребує стандартизації та нормалізації

Logistic Regression

- Вирішує задачі бінарної класифікації
- Легко інтерпретується

SVM

- Стійкий до шуму
- Ефективний у багатовимірних просторах
- Вирішує задачі регресії та класифікації

Модель 1 - K-Nearest Neighbors

K-Nearest Neighbors, каже, що для заданого значення K алгоритм знайде K найближчих сусідів для невідомої точки даних, а потім призначить клас для невідомої точки даних, враховуючи клас, у якого найбільша кількість точок даних з усіх класів K сусідів.

In [20]:
```
classificator = KNeighborsClassifier()
params = {'n_neighbors': range(1, 60)}
grid_search = GridSearchCV(classificator, params, cv=10, verbose=1)
grid_search.fit(x_train, y_train)
knn = grid_search.best_estimator_
knn
```

```
Fitting 10 folds for each of 59 candidates, totalling 590 fits
```

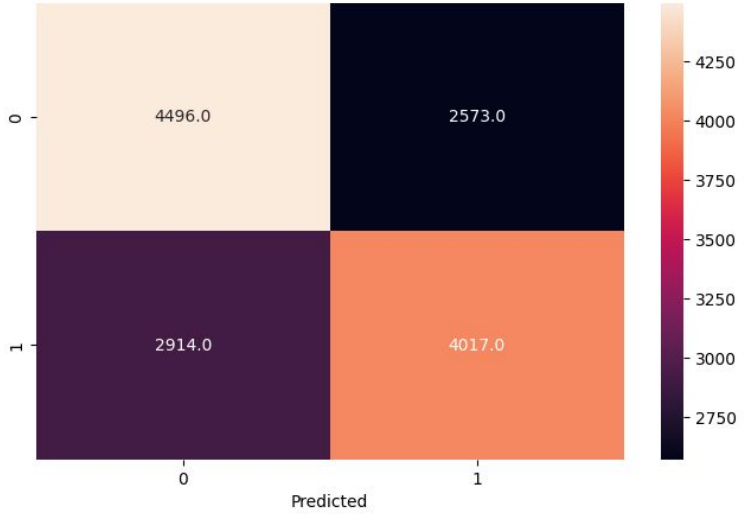
Out[20]:

```
▼        KNeighborsClassifier
KNeighborsClassifier(n_neighbors=58)
```

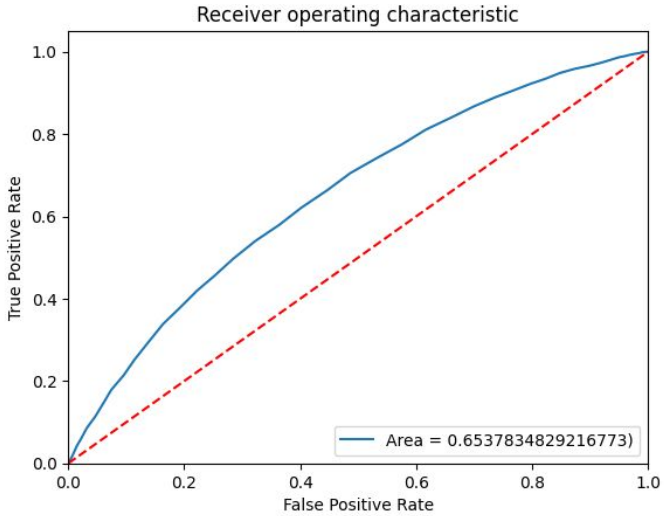
$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + \ldots + (x_n - x'_n)^2}$$

Модель 1 - K-Nearest Neighbors. Резултати.

In [24]: `conf_mat(knn, x_test, y_test)`



In [26]: `roc(knn, x_test, y_test)`

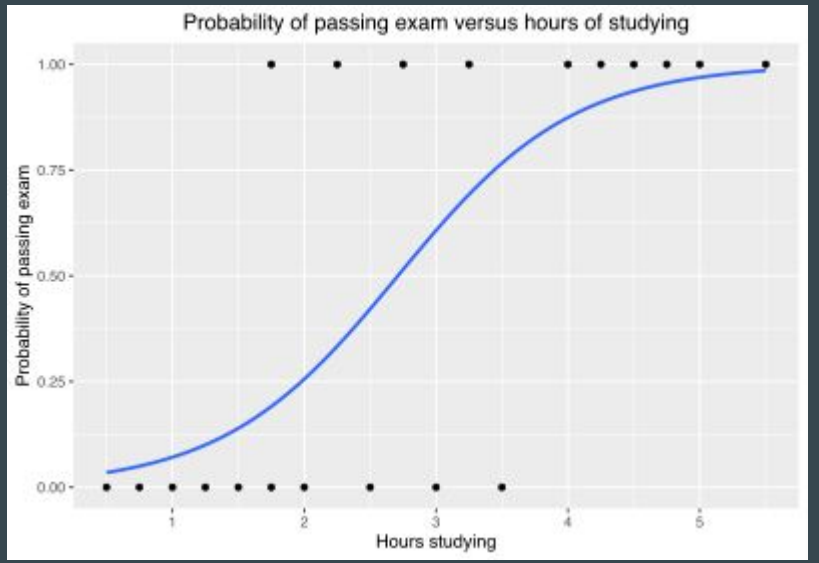


Модель 2 - Logistic Regression

Цей тип статистичної моделі (також відомий як логіт-модель) часто використовується для класифікації та прогнозної аналітики. Логістична регресія оцінює ймовірність події, наприклад проголосував або не проголосував, на основі заданого набору даних незалежних змінних. Оскільки результат є ймовірністю, залежна змінна обмежена між 0 і 1. У логістичній регресії до шансів застосовується логіт-перетворення, тобто ймовірність успіху, поділена на ймовірність невдачі. Це також широко відомо як логарифм шансів, або натуральний логарифм шансів, і ця логістична функція представлена такими формулами.

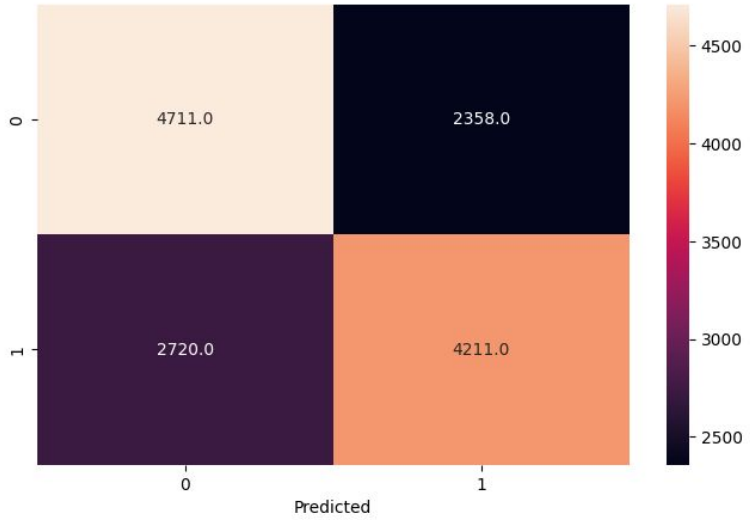
```
In [27]: import numpy as np
         from sklearn.linear_model import LogisticRegression
         from sklearn.pipeline import Pipeline
         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         logisticRegr = LogisticRegression()
         pipe = Pipeline(steps=[('sc', sc), ('logisticRegr', logisticRegr)])
         c = np.logspace(-4, 4, 60)
         penalty = ['l1', 'l2']
         params = dict(logisticRegr__C=c, logisticRegr__penalty=penalty)
         log_reg = GridSearchCV(pipe, params)
         log_reg.fit(x_train, y_train)
```

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

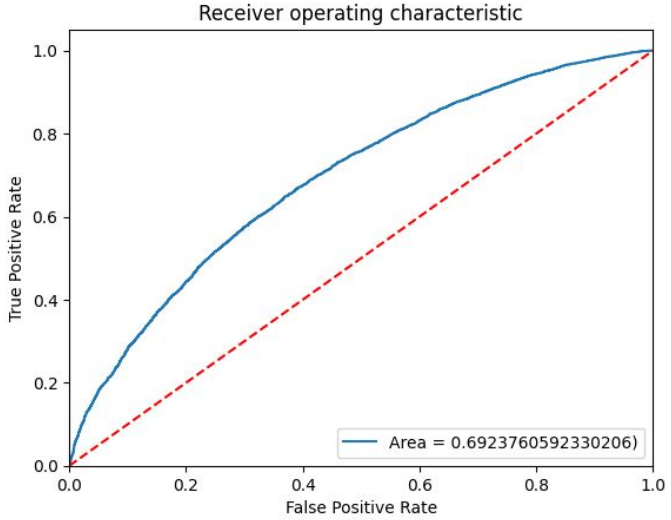


Модель 2 - Logistic Regression. Резултати.

```
In [29]: conf_mat(log_reg, x_test, y_test)
```




```
In [30]: roc(log_reg, x_test, y_test)
```



Модель 3 - Random Forest

Алгоритм навчання для випадкових лісів застосовує загальну техніку завантажувального агрегування або пакетування до тих, хто вивчає дерева. Враховуючи навчальний набір X = x1, ..., xn з відповідями Y = y1, ..., yn, пакетування повторно (B разів) вибирає випадкову вибірку із заміною навчального набору та підбирає дерева до цих зразків: Для b = 1, ..., B:

- Зразок, із заміною, n навчальних прикладів з X, Y; назвіть їх Xb, Yb.
- Навчіть класифікацію або дерево регресії fb на Xb, Yb.

Після навчання прогнози для невидимих зразків x' можна зробити шляхом усереднення прогнозів усіх окремих дерев регресії на x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

Крім того, оцінку невизначеності прогнозу можна зробити як стандартне відхилення прогнозів від усіх окремих дерев регресії на x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

In [31]:
```
from sklearn.ensemble import RandomForestClassifier
import numpy as np
# Кількість дерев
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 300, num = 60)]
params = {'n_estimators': n_estimators}
rf = RandomForestClassifier()
rf_random = GridSearchCV(rf, param_grid=params, cv=3, n_jobs=5)
rf_random.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarning: A worker stopped while some jobs were given to the executor. This can be caused by a too short worker timeout or by a memory leak.
  warnings.warn(
```

Out[31]:

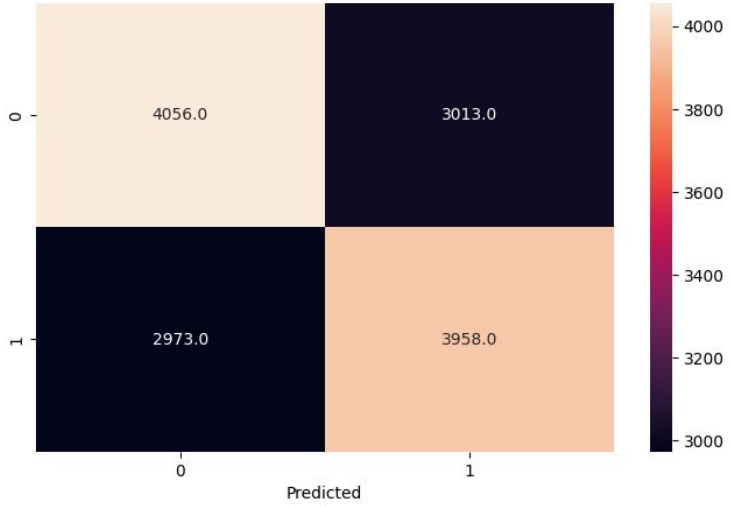
▸ GridSearchCV

▸ estimator: RandomForestClassifier

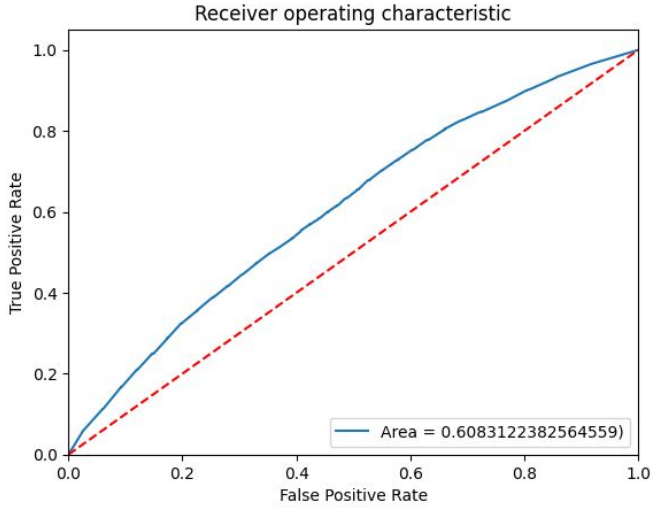
▸ RandomForestClassifier

Модель 3 -Random Forest. Резултати.

```
In [33]: conf_mat(rf_random, x_test, y_test)
```



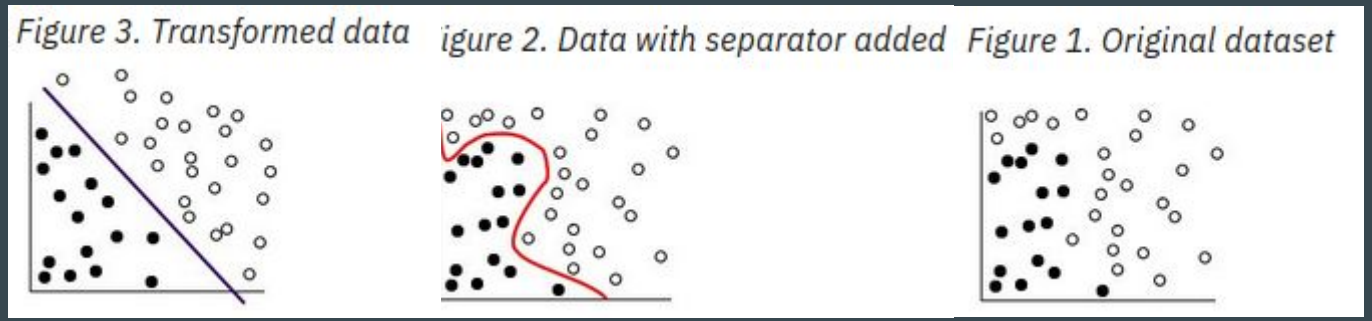

```
In [34]: roc(rf_random, x_test, y_test)
```



Модель 4 - SVM

SVM працює шляхом відображення даних у високовимірному просторі ознак, щоб точки даних можна було класифікувати, навіть якщо дані інакше не можна лінійно розділити. Знаходиться роздільник між категоріями, потім дані перетворюються таким чином, щоб роздільник можна було намалювати як гіперплощину. Точки розділені прямою до різних категорій(1), кривою(2). Після перетворення межу визначають за допомогою гіперплощини(3).

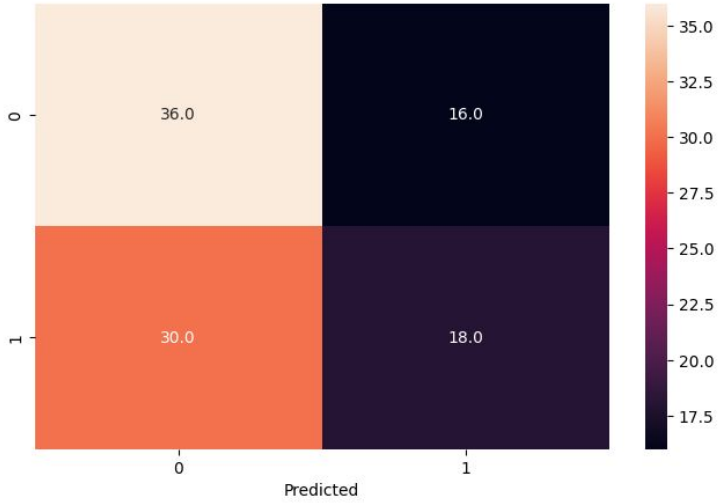
```
In [54]: train_score = round(svc_model.score(x_train, y_train), 5)
         test_score = round(svc_model.score(x_test, y_test), 5)
         results.append({'method': 'svm', 'score': train_score, 'type': 'train'})
         results.append({'method': 'svm', 'score': test_score, 'type': 'test'})
         print(f'Train accuracy: {train_score}')
         print(f'Test accuracy: {test_score}')

         Train accuracy: 0.62605
         Test accuracy: 0.62371
```



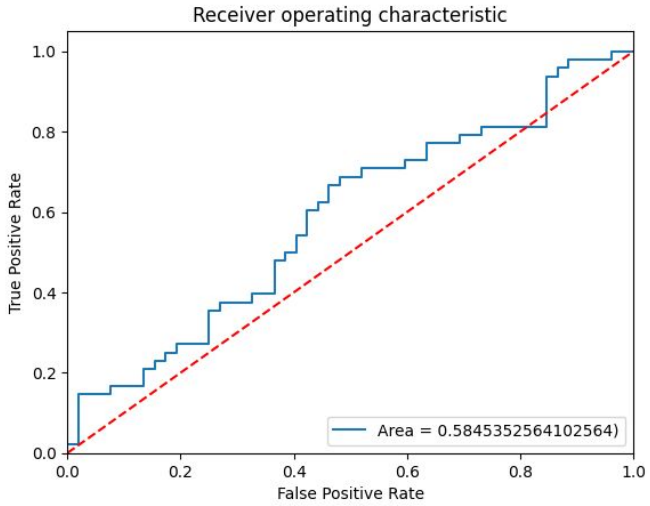
Математична функція, яка використовується для перетворення, відома як ядерна функція: лінійна, поліноміальна, радіальна базисна функція (RBF), сигмовидна.

Модель 4 -SVM. Резултати.

```
In [39]: conf_mat(svc_model, svm_x_test, svm_y_test)
```



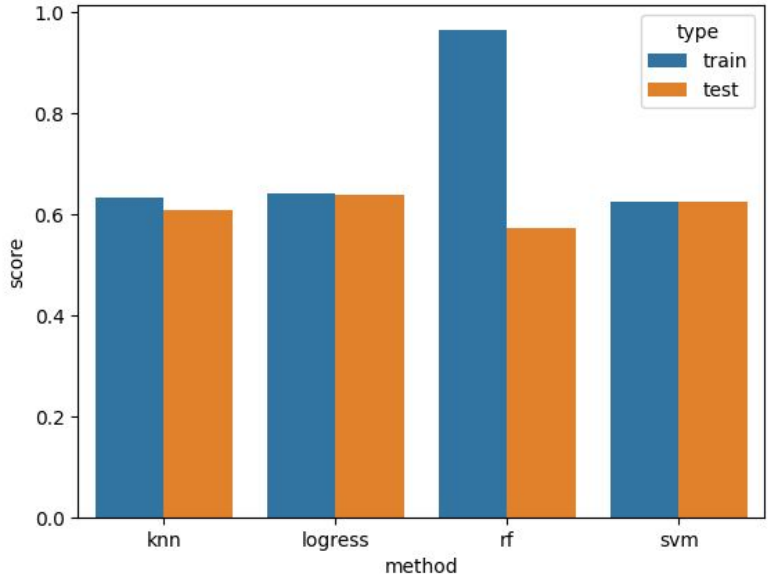

```
In [40]: roc(svc_model, x_test, y_test)
```



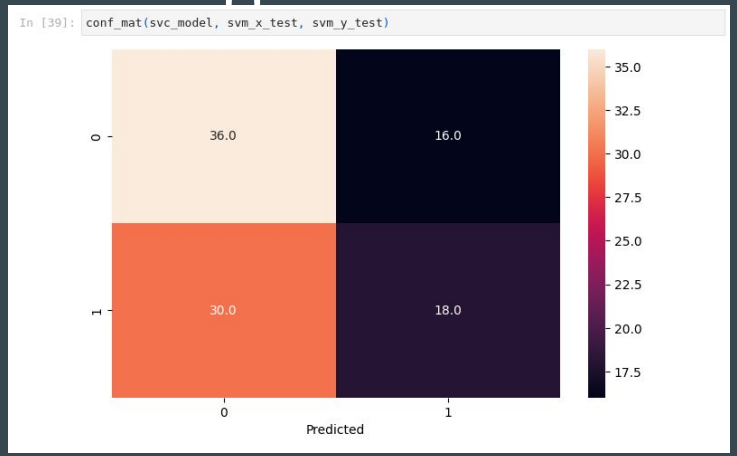
Порівняння моделей

```
In [56]: sns.barplot(x='method', y='score', hue='type', data=df_score)
```

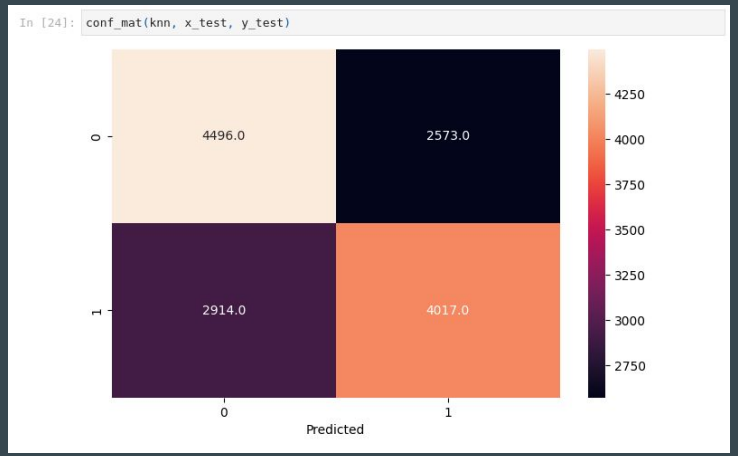
```
Out[56]: <AxesSubplot: xlabel='method', ylabel='score'>
```



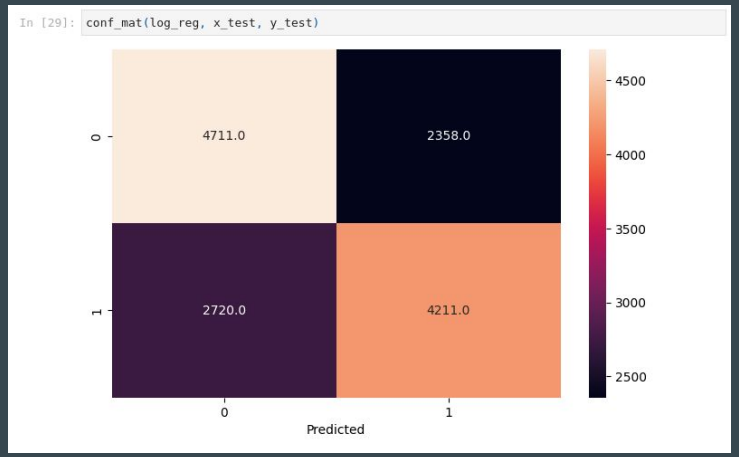
Порівняння моделей



SVM



K-Nearest Neighbors



Logistic Regression

Висновки

Аналіз результатів щодо серцево-судинних захворювань є критично важливим, оскільки він допомагає медичним працівникам краще зрозуміти фактори, які сприяють розвитку та прогресуванню серцево-судинних захворювань. Потім ці знання можна використати для розробки ефективніших стратегій профілактики та лікування.

Після аналізу даних встановлено, що метод Logistic Regression показав найкращі результати на тестових даних з точністю близько 63,73%. На тренувальних даних найкраще відпрацював Random Forest з точністю 96,56%, але на тестових даних його точність була найгіршою з результатом 57,24%. До того ж з матриць невідповідностей бачимо, що Logistic Regression виявив найбільше позитивно негативних результатів і найменше хибно негативних. Logistic Regression та SVM виявилися приблизно однаковими, проте через складність SVM було обрано набагато меншу вибірку ніж для Logistic Regression. Тому враховуючи те, що від діагностування хвороби залежить життя людини, я б обрав Logistic Regression.

Отже, висновок полягає в тому, що метод Logistic Regression є найбільш ефективним.