SPRAWOZDANIE

Zajęcia: Uczenie Maszynowe
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 1 Data 19.10.2024 Temat: Praktyczne zastosowanie regresji liniowej w analizie danych. Wariant drugi (2)	Bartosz Bieniek Informatyka II stopień, stacjonarne, 1 semestr, gr.A

1. Polecenie: Wariant drugi zadania

Opracować przepływ pracy uczenia maszynowego zagadnienia regresji (model regresji liniowej) oraz klasyfikacji binarnej (model SVM) na podstawie zbioru danych według wariantu zadania.

https://www.kaggle.com/datasets/thomaskonstantin/cpg-values-of-smoking-and-non-smoking-patients

2. Opis programu opracowanego

https://github.com/mindgoner/Studia/tree/master/Uczenie Maszynowe/Laboratorium 1

```
[14]  ✓ 0.0s
```

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVC
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, classification_report

# 1. Wczytanie danych, wstępne przetworzenie i kategoryzacja danych
data = pd.read_csv("smokers.csv")
data = data.dropna()
data['Smoking Status'] = data['Smoking Status'].map({'current': 1, 'former': 0, 'never': 0})
data['Gender'] = data['Gender'].map({'f': 0, 'm': 1})
```

Rys. 1. Fragment programu.

Na powyższym rysunku znajduje się fragment kodu. Pierwszą sekcję kodu stanowi import wszystkich wymaganych bibliotek.

```
[15]  ✓ 0.0s
```

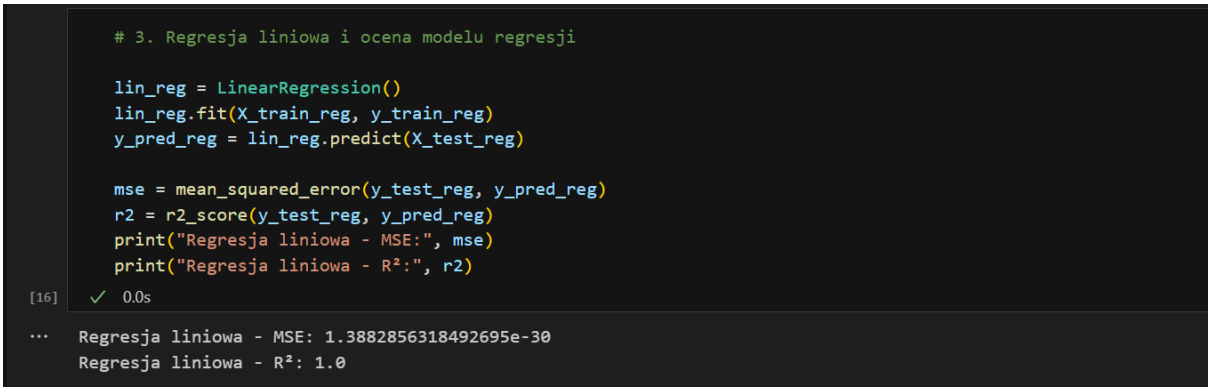
```python
# 2. Wyodrębnienie cech, zmiennych docelowych i podział na zbiór treningowy i testowy:

X = data.iloc[:, 4:]  # cechy epigenetyczne
y_regression = data['cg03683899']  # zmienna docelowa regresji
y_classification = data['Smoking Status']  # zmienna docelowa klasyfikacji

X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X, y_regression, test_size=0.2, random_state=42)
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X, y_classification, test_size=0.2, random_state=42)
```

Rys. 2. Fragment programu.

W kodzie na rysunku drugim wyodrębniono dane wejściowe (cechy epigenetyczne) oraz zmienne docelowe dla zadań regresji i klasyfikacji. Następnie dokonano podziału danych na zbiory treningowe i testowe w stosunku 80:20, aby przygotować je do uczenia modeli. Wykorzystano stałą wartość random_state dla zapewnienia powtarzalności wyników.



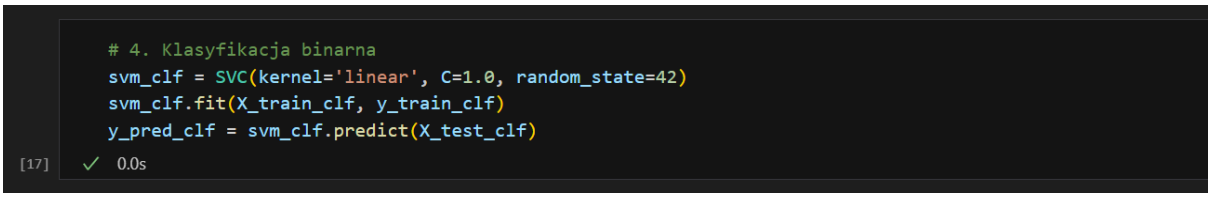

```
# 3. Regresja liniowa i ocena modelu regresji

lin_reg = LinearRegression()
lin_reg.fit(X_train_reg, y_train_reg)
y_pred_reg = lin_reg.predict(X_test_reg)

mse = mean_squared_error(y_test_reg, y_pred_reg)
r2 = r2_score(y_test_reg, y_pred_reg)
print("Regresja liniowa - MSE:", mse)
print("Regresja liniowa - R²:", r2)
```

Rys. 3. Fragment programu oraz wynik wykonania

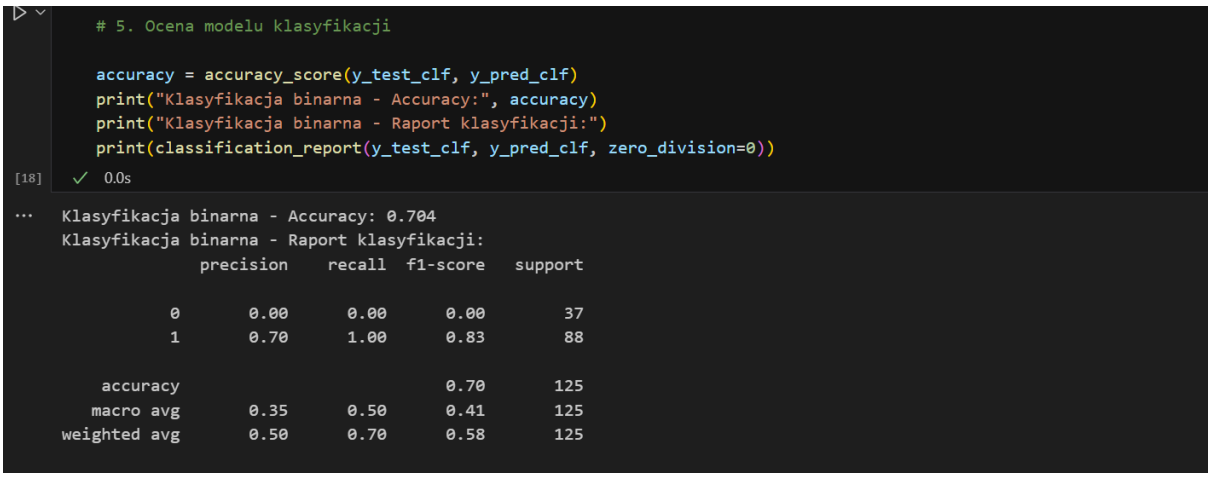
We fragmencie przedstawionym na rysunku trzecim utworzono i wytrenowano model regresji liniowej na zbiorze treningowym. Następnie za pomocą tego modelu dokonano predykcji na zbiorze testowym. Obliczono dwie miary jakości modelu: średni błąd kwadratowy (MSE) oraz współczynnik determinacji (R^2), które przedstawiono na ekranie.




```
# 4. Klasyfikacja binarna
svm_clf = SVC(kernel='linear', C=1.0, random_state=42)
svm_clf.fit(X_train_clf, y_train_clf)
y_pred_clf = svm_clf.predict(X_test_clf)
```

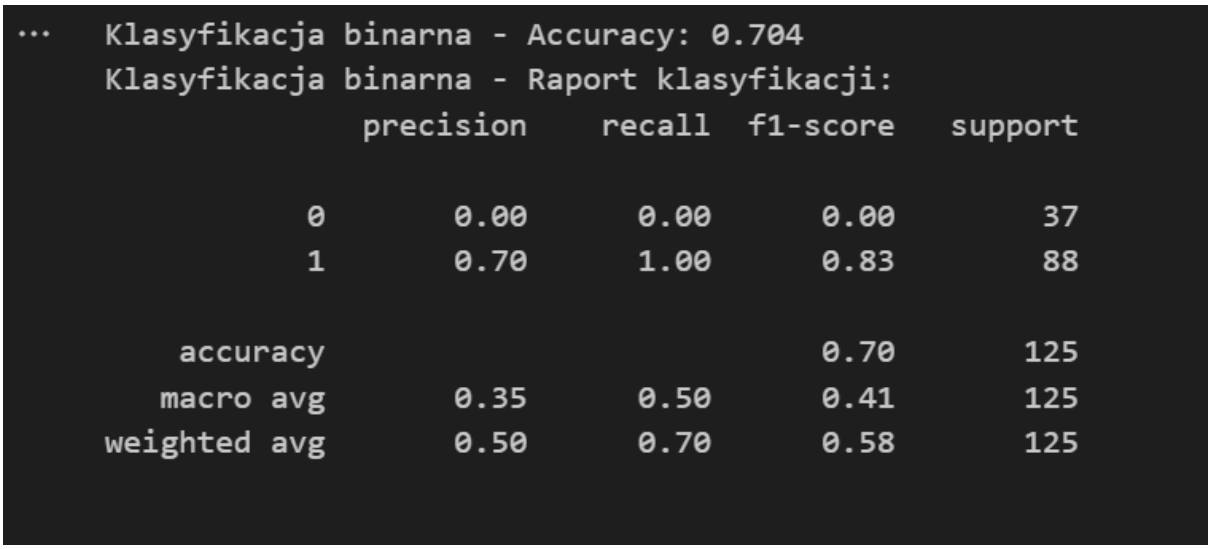
Rys. 4. Fragment programu.

Kolejnym krokiem było utworzenie modelu klasyfikacji SVM z liniowym jądrem oraz parametrem regularyzacji C=1.0. Model został wytrenowany na zbiorze treningowym, a następnie wykorzystano go do przewidywania etykiet klas na zbiorze testowym. Ustawiono random_state, aby wyniki były powtarzalne.



Rys. 5. Fragment programu.

Ostatnim punktem było obliczenie dokładności (accuracy) modelu klasyfikacji na zbiorze testowym, aby ocenić jego ogólną skuteczność. Dodatkowo wygenerowano szczegółowy raport klasyfikacji, zawierający miary takie jak precision, recall oraz F1-score dla każdej klasy. Parametr zero_division=0 zastosowano, w celu uniknięcia ostrzeżeń związanych z brakiem przewidywanych próbek w niektórych klasach.



Rys. 6. Fragment programu.

Wyniki klasyfikacji binarnej dla metody SVM wykazały dokładność równą 0.704. Model osiągnął wysoką precyzję dla klasy pozytywnej (klasa 1), wynoszącą 0.70, jednak z bardzo niskim współczynnikiem dla klasy zerowej, co przekłada się na F1-score równy 0.83 dla klasy pierwszej. Dla klasy zero wskaźniki precyzji, i F1 są równe 0, co wskazuje na nierozpoznanie tej klasy przez model. W wyniku zastosowania metryki macro avg, uzyskano średnie wartości miar, jednak są one również ograniczone przez zdominowanie wyników przez klasyfikację klasy 1. Wagi dla poszczególnych kla (weighted avg)

podniosły dokładność, lecz nie są wystarczające, do skutecznaego rozróżnienia wszystkich klas. Problem nierozpoznania klasy zero, stanowi główną słabość tego modelu.

3. Wnioski

Zbiór danych powinien być zrównoważony, aby model nie faworyzował jednej klasy. Można to osiągnąć poprzez zastosowanie technik takich jak oversampling (np. SMOTE) dla klasy mniejszościowej lub undersampling dla klasy większościowej.

Model SVM wymaga bardziej precyzyjnego dobrania parametrów, takich jak wartość C oraz inne hiperparametry, które wpływają na wagę błędów. Dostrojenie parametrów mogłoby pomóc w lepszym dobraniu modelu do danych.