Лабораторна робота №2

Тема: Порівняння методів класифікації даних

Завдання 1: Класифікація за допомогою машин опорних векторів SVM

На початку роботи випишимо атрибути що надаються з набору даних та визначимо їх тип (таб. 1).

Таблиця 1. Опис атрибутів

|  |  |  |
| --- | --- | --- |
| Атрибут | Опис | Тип |
| Вік | Вік | Числова |
| Тип зайнятості | Тип зайнятості (ФОП, працює на невону ставку тощо) | Категоріальна |
| Вага | Коефіціент вани у файлах поточного опитування населення | Числова |
| Освіта | Рівень освіти (бакалавр, неповне середне тощо) | Категоріальна |
| Освітній номер | Номер рівня освіти | Числова |
| Подружній стан | Одруженний, розлученний тощо (поточний статус) | Категоріальна |
| Професійна діяльність | Рід зайнять | Категоріальна |
| Сімейний стан | Має дитину, розлученний тошо | Категоріальна |
| Раса | Расова приналежність | Категоріальна |
| Стать | Статева приналежність | Категоріальна |
| Кріїна походження | Країна проживання (поточна) | Категоріальна |
| Дохід | Середній дохід в рік | Категорівльна |
| Годин в неденю працює | Кількість робочих годин в неділю | Числова |
| Капітал зростання | Рівень росту доходу | Числова |

Після описання атрибутів напишимо программу для опису значення показника якості F1 за зразком доповнивши програму пошуком інших показників якості. Обрана точка позначення доходу не дивлячись на наявність чисел в нашому випадку скоріше виступає в якості категоріальної (визначає групи чий дохід вище або нижче за деяке значення)

*Лістинг програми:*

import numpy as np  
import matplotlib.pyplot as plt  
import sklearn.svm  
from sklearn import preprocessing  
from sklearn.svm import LinearSVC  
from sklearn.svm import SVC  
  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import accuracy\_score  
from sklearn.metrics import recall\_score  
from sklearn.metrics import precision\_score  
  
  
#Input Data  
input\_file = "income\_data.txt"  
  
X = []  
Y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 500  
  
with open(input\_file, "r") as f:  
 for line in f.readlines():  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:  
 break  
 if '?' in line:  
 continue  
 data = line[:-1].split(', ')  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
 if data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
  
X = np.array(X)  
  
print("---------[X After file reading]--------\n")  
print(X)  
  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
X = X\_encoded[:, :-1].astype(int)  
Y = X\_encoded[:, -1].astype(int)  
scaller = preprocessing.MinMaxScaler(feature\_range=(0,1))  
X = scaller.fit\_transform(X)  
  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0))  
  
classifier.fit(X=X, y=Y)  
  
X\_train, X\_test, y\_train, y\_test \  
 = train\_test\_split(X, Y, test\_size=0.2, random\_state=5)  
  
scaller = preprocessing.MinMaxScaler(feature\_range=(0,1))  
X\_train = scaller.fit\_transform(X\_train)  
  
classifier.fit(X=X\_train, y=y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
f1 = cross\_val\_score(classifier, X, Y, scoring="f1\_weighted", cv=3)  
accuracy\_values = cross\_val\_score(classifier, X, Y, scoring='accuracy', cv=3)  
print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")  
precision\_values = cross\_val\_score(classifier, X, Y, scoring='precision\_weighted', cv=3)  
print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")  
recall\_values = cross\_val\_score(classifier, X, Y, scoring='recall\_weighted', cv=3)  
print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")  
f1\_values = cross\_val\_score(classifier, X, Y, scoring='f1\_weighted', cv=3)  
print("F1: " + str(round(100 \* f1\_values.mean(), 2)) + "%")  
  
print("F1 score: " + str(round(100\*f1.mean(), 2)) + "%")  
  
input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners',  
 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']  
  
  
input\_data\_encoded = np.array([-1] \* len(input\_data))  
count = 0  
  
  
for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = item  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([item]))  
 count += 1  
  
input\_data\_encoded = input\_data\_encoded.astype(int)  
input\_data\_encoded = [ input\_data\_encoded ]  
  
print("-\_-\_-\_-\_-\_-")  
print(input\_data\_encoded)  
  
predicate\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicate\_class)[0])

A computer screen capture

Description automatically generated with medium confidence

Рисунок 1. Результати роботи программи

Завдання 2: Порівняння якості класифікаторів SVM з нелінійними ядрами

Наступним кроком обчислимо значення показників з викорситанням різних ядер класифікатора.

*Лістинг програми:*

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 2. Сигмоїдне ядро

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 3. Poly ядро

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 4. Rbf ядро

Як можна бачити rbf дало гарні результати дещо поступившись полі ядру, але набагато вигравши в швидкодії. Сигмоїдне ядро дало більш низькі результати, що дає нам основи вважати що rbf ядро, для нашого випадку, надає найкріші поєднання швидкодії та точності.

Завдання 3: Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Для виконання даного завдання викорстаємо набір даних що надаєтся бібліотекою sklearn. Нижче наведено повний лістинг программи.

*Лістинг програми:*

from sklearn.datasets import load\_iris  
iris\_dataset = load\_iris()  
  
  
print("Ключи Iris Dataset : \n{}".format(iris\_dataset.keys()))  
print(iris\_dataset["DESCR"][:193] + "\n...")  
print("Names answers: {}".format(iris\_dataset["target\_names"]))  
print("Names for description: \n{}".format(iris\_dataset["feature\_names"]))  
print("Type of data array: {}".format(type(iris\_dataset["data"])))  
print("Form of array data: {}".format(iris\_dataset["data"].shape))  
print("Тип масиву target: {}".format(type(iris\_dataset['target'])))  
print("Відповіді:\n{}".format(iris\_dataset['target']))  
  
# Завантаження бібліотек  
import numpy as np  
from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import StratifiedKFold  
from sklearn.metrics import classification\_report  
from sklearn.metrics import confusion\_matrix  
from sklearn.metrics import accuracy\_score  
from sklearn.linear\_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
  
  
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"  
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']  
dataset = read\_csv(url, names=names)  
  
# shape  
print(dataset.shape)  
  
# Зріз даних head  
print(dataset.head(20))  
  
# Стастичні зведення методом describe  
print(dataset.describe())  
  
# Розподіл за атрибутом class  
print(dataset.groupby('class').size())  
  
# Діаграма розмаху  
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)  
pyplot.show()  
  
  
# Гістограма розподілу атрибутів датасета  
dataset.hist()  
pyplot.show()  
  
  
  
#Матриця діаграм розсіювання  
scatter\_matrix(dataset)  
pyplot.show()  
  
  
# Розділення датасету на навчальну та контрольну вибірки  
array = dataset.values  
  
# Вибір перших 4-х стовпців  
X = array[:,0:4]  
  
# Вибір 5-го стовпця  
y = array[:,4]  
  
# Разделение X и y на обучающую и контрольную выборки  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X, y, test\_size=0.20, random\_state=1)  
  
  
#LOAD MODELS  
models = []  
models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))  
models.append(('LDA', LinearDiscriminantAnalysis()))  
models.append(('KNN', KNeighborsClassifier()))  
models.append(('CART', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', SVC(gamma='auto')))  
  
#Quality models  
results = []  
names = []  
  
for name, model in models:  
 kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)  
 cv\_results = cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring='accuracy')  
 results.append(cv\_results)  
 names.append(name)  
 print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))  
  
# Порівняння алгоритмів  
pyplot.boxplot(results, labels=names)  
pyplot.title('Algorithm Comparison')  
pyplot.show()  
  
# Створюємо прогноз на контрольній вибірці  
model = SVC(gamma='auto')  
model.fit(X\_train, Y\_train)  
predictions = model.predict(X\_validation)  
  
# Оцінюємо прогноз  
print(accuracy\_score(Y\_validation, predictions))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))  
  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
  
for name, model in models:  
 model.fit(X\_train, Y\_train)  
 prediction = model.predict(X\_new)  
 print("Прогноз\_\_\_: {}".format(prediction))  
 print(accuracy\_score(Y\_validation, predictions))  
 print(confusion\_matrix(Y\_validation, predictions))  
 print(classification\_report(Y\_validation, predictions))

При виконанні роботи оброробивши модель ми отримали наступну діаграмму розміху ірисів(рис.5)

Chart, box and whisker chart

Description automatically generated

Рисунок 5. Діаграмма розмаху

Гістограмму розподілу(рис.6)

Chart, box and whisker chart

Description automatically generated

Рисунок 6. Гістограмма розподілу атрибутів датасет

Та матрицю діагрми розсіювання (рис.7)

Chart

Description automatically generated

Рисунок 7. Матриця діаграми розсіювання

Сформувавши оцінки алгоритмів було отримано наступну діаграму для поточнох конфігурації

Chart, box and whisker chart

Description automatically generated

Рисунок 7. Порівняння алгоритмів

Як можна побачити з діаграми порівняння алгоритмів для конфігурації студента Найкраше себе проявила модель лінійного дискрімінінтного аналізу однак вона не досить стійка й під час тестування займала порівнянно більше часу на своє виконання. Досить непогані результати було отримано з алгоритму к сусідів, але найкраще поєднання передбачиваної точності й швидкої дає алгоритм дерева рішень.

Наступним кроком проведемо оцінки моделі та застосуємо модель для передбачення (код програми засновано на прикладі з методичних рекомендацій з дописом деяких частин)

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 9. Обробка якостімоделі

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 10. Використання моделі для передбачення

Квітка з кроку належить до класу setosa

Завдання 4: Порівняння якості класифікаторів для набору даних

Для виконання цього завдання адаптуємо з виокристанням багатьох моделей до використання на першому наборі даних.

*Лістинг програми:*

import numpy as np  
from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import StratifiedKFold  
from sklearn.metrics import classification\_report  
from sklearn.metrics import confusion\_matrix  
from sklearn.metrics import accuracy\_score  
from sklearn.linear\_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
  
  
import numpy as np  
import matplotlib.pyplot as plt  
import sklearn.svm  
from sklearn import preprocessing  
from sklearn.svm import LinearSVC  
from sklearn.svm import SVC  
  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.metrics import accuracy\_score  
from sklearn.metrics import recall\_score  
from sklearn.metrics import precision\_score  
  
  
#Input Data  
input\_file = "income\_data.txt"  
  
X = []  
Y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 500  
  
  
with open(input\_file, "r") as f:  
 for line in f.readlines():  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:  
 break  
 if '?' in line:  
 continue  
 data = line[:-1].split(', ')  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
 if data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
  
X = np.array(X)  
  
  
  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
X = X\_encoded[:, :-1].astype(int)  
Y = X\_encoded[:, -1].astype(int)  
  
  
scaller = preprocessing.MinMaxScaler(feature\_range=(0,1))  
X = scaller.fit\_transform(X)  
  
  
models = []  
models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))  
models.append(('LDA', LinearDiscriminantAnalysis()))  
models.append(('KNN', KNeighborsClassifier()))  
models.append(('CART', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', SVC(gamma='auto')))  
  
for name, model in models:  
 classifier = OneVsOneClassifier(model)  
 classifier.fit(X=X, y=Y)  
  
 X\_train, X\_test, y\_train, y\_test \  
 = train\_test\_split(X, Y, test\_size=0.2, random\_state=5)  
  
 scaller = preprocessing.MinMaxScaler(feature\_range=(0,1))  
 X\_train = scaller.fit\_transform(X\_train)  
  
 classifier.fit(X=X\_train, y=y\_train)  
 y\_test\_pred = classifier.predict(X\_test)  
  
  
 f1 = cross\_val\_score(classifier, X, Y, scoring="f1\_weighted", cv=3)  
 accuracy\_values = cross\_val\_score(classifier, X, Y, scoring='accuracy', cv=3)  
 print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")  
 precision\_values = cross\_val\_score(classifier, X, Y, scoring='precision\_weighted', cv=3)  
 print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")  
 recall\_values = cross\_val\_score(classifier, X, Y, scoring='recall\_weighted', cv=3)  
 print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")  
 f1\_values = cross\_val\_score(classifier, X, Y, scoring='f1\_weighted', cv=3)  
 print("F1: " + str(round(100 \* f1\_values.mean(), 2)) + "%")  
  
 print("F1 score: " + str(round(100\*f1.mean(), 2)) + "%")  
  
 input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners',  
 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']  
  
  
 input\_data\_encoded = np.array([-1] \* len(input\_data))  
  
 print("LABEL ENCODERS")  
 print(input\_data\_encoded)  
 count = 0  
  
  
 for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = item  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([item]))  
 count += 1  
  
 input\_data\_encoded = input\_data\_encoded.astype(int)  
 input\_data\_encoded = [ input\_data\_encoded ]  
  
 print("-\_-\_-\_-\_-\_-")  
 print(input\_data\_encoded)  
  
 predicate\_class = classifier.predict(input\_data\_encoded)  
 print(label\_encoder[-1].inverse\_transform(predicate\_class)[0])

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Рисунок 11. Отримані результати точності для різних класифікаторів

Завдання 5: Класифікація данних лінійним класифікатором Ridge

Наступним кроком проведемо класифікаю за допомогою лынійного класифікатора Ridge

*Лістинг програми:*

import numpy as np  
from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import StratifiedKFold  
from sklearn.metrics import classification\_report  
from sklearn.metrics import confusion\_matrix  
from sklearn.metrics import accuracy\_score  
from sklearn.linear\_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
  
#LAST TASK  
# ===================================================  
# Приклад класифікатора Ridge  
# ======================================================================  
import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn.linear\_model import RidgeClassifier  
  
iris = load\_iris()  
X, y = iris.data, iris.target  
Xtrain, Xtest, ytrain, ytest = train\_test\_split(X, y, test\_size = 0.3, random\_state = 0)  
clf = RidgeClassifier(tol = 1e-2, solver = "sag")  
clf.fit(Xtrain,ytrain)  
ypred = clf.predict(Xtest)  
  
from sklearn import metrics  
print('Accuracy:', np.round(metrics.accuracy\_score(ytest,ypred),4))  
print('Precision:', np.round(metrics.precision\_score(ytest,ypred,average = 'weighted'),4))  
print('Recall:', np.round(metrics.recall\_score(ytest,ypred,average = 'weighted'),4))  
print('F1 Score:', np.round(metrics.f1\_score(ytest,ypred,average = 'weighted'),4))  
print('Cohen Kappa Score:', np.round(metrics.cohen\_kappa\_score(ytest,ypred),4))  
print('Matthews Corrcoef:', np.round(metrics.matthews\_corrcoef(ytest,ypred),4))  
print('\t\tClassification Report:\n', metrics.classification\_report(ypred,ytest))  
  
from sklearn.metrics import confusion\_matrix  
from io import BytesIO  
  
#neded for plot  
import seaborn as sns;  
sns.set()  
import matplotlib.pyplot as plt  
  
mat = confusion\_matrix(ytest, ypred)  
sns.heatmap(mat.T, square = True, annot = True, fmt = 'd', cbar = False)  
plt.xlabel('true label')  
plt.ylabel('predicted label')  
plt.savefig("Confusion.jpg")  
# Save SVG in a fake file object.  
f = BytesIO()  
plt.savefig(f, format = "svg")

Використаємо два параметри налаштування класифікатора точність рішення (тол) та спосіб рішення що використовується при розрахунках (в нашому випадку це sag (Алгоритм середнього стохастичного градіенту)

Calendar

Description automatically generated with medium confidence

Рисунок 12. Отриманий результат

Як можна побачити з отриманного результату було використано показники якості точність, ф1, recall, коефиціент Коена Каппа (статистичне значення, що вимірую міжрегаональну згоду на категоріальні предмети, вважається більш надійним ніж розрахунок у відсотках бо враховує випадковість) та коефіціент кореляції Метьюза (використовується в машиному навчанні як міра якості бінарних мультикласних класифікацій, враховує істинні, хибні, позитивні та негативні результати, може бути використанний коли класи мають дуже різні розміри)

Отсанні кроком пояснимо що наведено на малюнку нижче (рис.13)

Це – Матриця невідповідності, згідно інформації отриманій з мережі інтернет це спеціальна матриця що використовується в галузі машинного навчання, й зокрема в задачі статистичної класифікації, ма́триця невідпові́дностей(англ. confusion matrix), також відома як матриця помилок (англ. error matrix) - це таблиця особливого компонування, що дає можливість унаочнювати продуктивність алгоритму, зазвичай керованого навчання (у спонтаннім навчанні її зазвичай називають ма́трицею допасо́ваності, англ. matching matrix). Кожен з рядків цієї матриці представляє зразки прогнозованого класу, тоді як кожен зі стовпців представляє зразки справжнього класу (або навпаки) Її назва походить від того факту, що вона дає можливість просто бачити, чи допускає система невідповідності між цими двома класами (наприклад, часто помилково маркуючи один як інший).

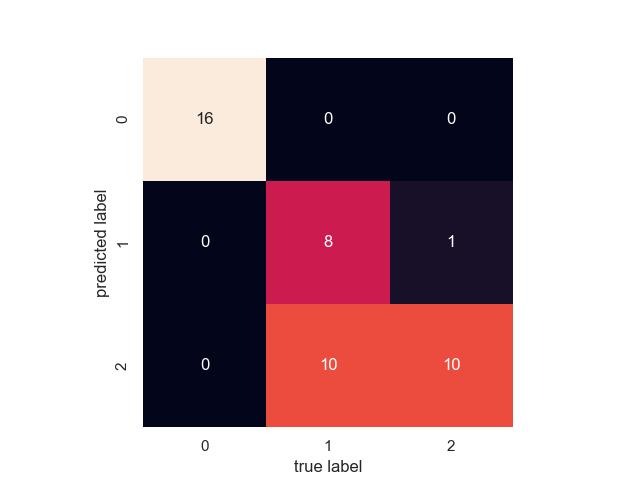


Рисунок 13. Матриця невідповідностей

Висновок: Розглянули особливості роботи з машинами векторів. Провели порівняльний аналіз різних методів класифікації даних.