Въведение в Машинното обучение
Лабораторно упражнение №8

Класификация

Алгоритми за класификация в машинното обучение.

1. Цел на упражнението.

Целта на лабораторното упражнение е студентите да усвоят темата класификация, като я приложат при решаване на класификационни задачи.

2. Алгоритми за класификация.

Постановка на задачата:

Нека:

- $X \in R^n$ – множество на обектите (входни данни)
- $У \in R$ - множество на резултатите (изходни данни)

Ще разгледаме двойката (x, y), като реализация на (n+1) – мерни случайни величини (X,У), зададени във вероятностното пространство.

Законът на разпределение $P_{XУ}(x, y)$ не е известен, известна е само обучаваща извадка:

$$\{x^{(1)}, y^{(1)}, x^{(2)}, y^{(2)}, \ldots\ldots, x^{(N)}, y^{(N)}\}$$

където $\left(x^{(i)}, y^{(i)}\right),\ i = 1,2, \ldots, N$ се явяват независими реализации на случайна величина (X, Y).

Целта е да се намери функция $f: X \to Y$, която изхождайки от стойностите на x, да предскаже y. Функцията f се нарича решаваща функция или класификатор. Построяването на f наричаме обучение, настройки на модели и т.н.

2.1.Наивен Бейсов класификатор

Методът на Бейс, използван в машинното обучение се основава на известната формула на Бейс:

$$P(H_k \mid A) = \frac{P(H_k).P(A \mid H_k)}{P(A)},\ k = 1,2,\ldots,n$$

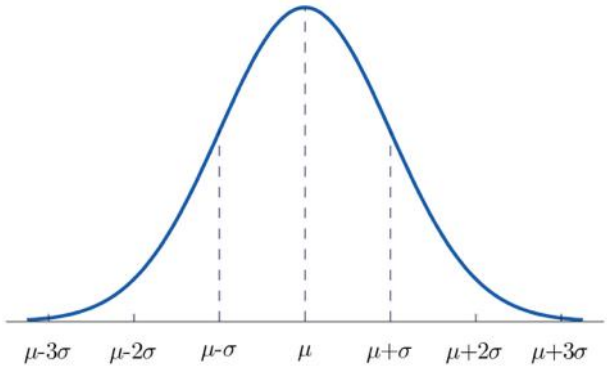
за определяне на постериорната вероятност за настъпването на събитието А съвместно с хипотезата $H_k H_k$.

Във формулата на Бейс:

- A е случайно събитие, настъпването на което винаги се съпътства с едно от събитията $H_1, H_2, \ldots, H_n$, наричани хипотези и образуващи пълна група от несъвместими събития;
- $P(H_k)\ \ k = 1,2,\ldots,n$ е априорната вероятност на хипотезата H_k ;
- $P(A \mid H_k)$ е условната вероятност за настъпване на събитието A при условие, че е настъпила хипотезата H_k;

2.1.1. Гаусов Бейсов класификатор

Гаусовият Наивен Бейсов класификатор е свързан с предположение за нормално разпределение на признаците.



Функцията на вероятностната плътност на нормално разпределена случайна величина е:

$$\mathrm{P}(x|\mu,\sigma) = normpdf(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}},$$

където

μ - математическо очакване:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)^2$$

σ - стандартното отклонение:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu)^2$$

Наивният аспект на алгоритъма е, че той третира всички входни величини като независими едни от други.

2.2.Метод на опорните вектори - SVM (Support Vector Mashine)

Методът на опорните вектори (SVM, Support Vector Machines) представя обучаващите примери като точки в n-мерно пространство. Примерите са проектират в пространството по такъв начин, че да бъдат линейно разделими. При работа с два класа се търси начин да се начертае линия, която да разделя данните от двата класа. Линията, която разделя данните, се нарича разделителна хипер равнина. Тази хипер равнина трябва да се избере по такъв начин, че да се намира възможно най-далеч от примерите и на двата класа.

Функцията f(x) на линейната класификация е във вида:

$$f(x) = w^T x + b$$

където: w^T *е* тегловен вектор, а b е отклонението

След като се определи, къде да бъде поставена разделителната линия, се калкулира допустимата границата:

$$label * (w^T x + b$$

Целта е да се намерят стойностите на w^T и b, които ще определят класификатора. За да се направи това, е необходимо да се намерят точките с най-малко отклонение, който трябва да се максимизира. Това може да се запише по този начин:

$$arg \max_{w,b} \left\{ \min_{n} label * (w^T + b). \frac{1}{\|w\|} \right\}$$

Функции на ядрото:

- Полиномиална функция на ядрото
- Гаусова радиална базисна функция
- Експоненциална радиална базова функция
- Сигмуид
- Многопластов перцептрон
- Би Сплайн функция
- Добавени ядра

2.3.Метод на К най-близкия съсед (KNN - k nearest neighbours)

Методът на К най-близкия съсед е метрически алгоритъм за класификация на обекти, основан на оценка на сходство на обекти. Класифицираният обект се отнася към този клас, към който принадлежат най-близките до него обекти на обучаващата извадка (KNN - k nearest neighbours).

Алгоритъм

- Определяне на К броя обекти, които ще участват в класификацията
- Изчисляване на дистанцията между всеки два обекта от обучаващата извадка, чрез използване на подходяща функция за измерване на разстояние между две точки
- Избор на К обекти от обучаващата извадка, разстоянието до което е минимално
- Класа на класифицирания обект – това е класа на новия обект, намерен въз основа на заключенията, направени по отношение на К най-близки съседи

Важен въпрос при работа с метода на К най-близкия съсед е изборът на числото К – това е броя на най-близките съседи. Евристични техники, като кръстосано валидиране могат да помогнат за получаването на подходящи стойности на К.

2.4.Невронни мрежи - Многослойните перцептрони

Многослойните перцептрони (Multilayer Perceptron - MLP) са вид невронни мрежи, която се използва за класификация и регресия. Те се състоят от поне един скрит слой на неврони между входния и изходния слой.

Невроните в скрития слой използват нелинейни активационни функции за да обработват входните данни и да генерират изходи, които се изпращат към следващия слой. Стойностите на изходите на последния слой от неврони представляват резултата от класификацията или регресията.

Обучението на MLP се извършва чрез обратно разпространение на грешката (backpropagation), която се основава на градиентния спуск. Този метод използва грешката

между предвидените и реалните изходи, за да коригира теглата на невроните в мрежата. Целта е да се минимизира грешката на обучение.

Перцептронът е модел за класификация, който се използва за разделяне на два класа по линейна граница. При този модел се използва само един слой и няма скрит слой.

Многослойният перцептрон (MLP) е по-мощен модел, тъй като има скрит слой, който му позволява да моделира по-сложни функции. Освен това MLP може да работи и с множество класове, като използва softmax функцията за активация в последния изходен слой.

3. Метрики използвани при класификация

Класификационният модел разпределя съответния обект в един от възможните класове вярно (True) или невярно (False). При този модел са възможни четири варианта на класификация:

- True Positive (TP) – действителен клас Positive и вярно класифициран като клас Positive;

$$TP\ Rate = \frac{TP}{TP + FN}$$

- True Negative (TN) – действителен е клас Negative и вярно класифициран като клас Negative;

$$TN\ Rate = \frac{TN}{TN + FP}$$

- False Positive (FP) - действителен е клас Negative, а невярно класифициран като клас Positive;

$$FP\ Rate = \frac{FP}{FP + TN}$$

- False Negative (FN) - действителен е клас Positive, а невярно класифициран като клас Negative.

$$FN\ Rate = \frac{FN}{FN + TP}$$

Прецизността (Precision) е мярка, която показва доколко е точен класификатора и дава процента на правилно класифицираните примери от съответния клас.

$$Precision = \frac{TP}{TP + FP}$$

Мярката чувствителност (Recall) показва пълнотата или възвращаемостта на класификатора, т.е. колко от всички примери от даден клас е успял да определи като принадлежащи на този клас. Обикновено се представя в проценти.

$$Recall = \frac{TP}{TP + FN}$$

Мярката F-measure, която комбинира Precision и Recall е тяхното претеглено хармонично средно. Обикновено се представя в проценти.

$$F\ measure = \frac{2}{{}^{1}/_{Recall} + {}^{1}/_{Precision}}$$

Мярката точност (Accuracy) се изчислява като отношението на правилно класифицирани примери към общия брой обекти от тестово множеството. Обикновено се представя в проценти.

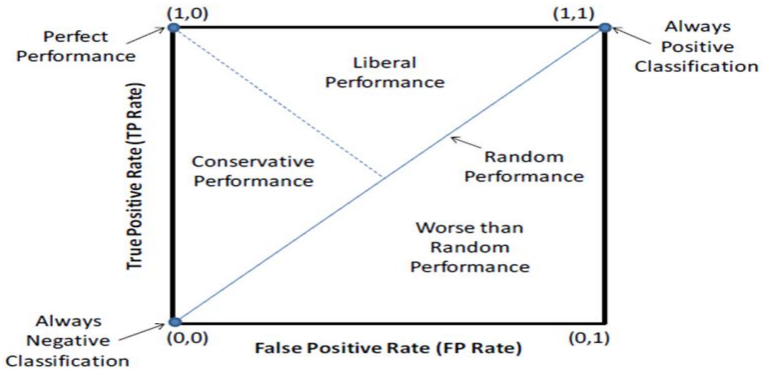
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Класификационната грешка (Error Rate) се изчислява като отношението на неправилно класифицирани примери към общия брой обекти от тестово множеството. Обикновено се представя в проценти.

$$Error\ Rate = \frac{FP + FN}{TP + FP + TN + FN}$$

ROC Area - Като количествена оценка на работата на класификационните модели се използва площта под ROC кривата - *ROC Area*. Максималната стойност е 1, а минималната е 0. Класификационният модел е по-добър, колкото по-голяма е площта над ROC кривата. Ако ROC Area<0.5, то оценяваният модел работи по-лошо от случайния класификатор, който е с ROC Area=0.5.

Receiver Operating Characteristic (ROC) кривите се използват за визуална оценка на точността на класификаторите, както и за сравняване на различни класификационни модели. ROC кривата представлява двумерна диаграма, която се получава чрез изобразяване на True Positive Rate (TP Rate) по вертикалната ос и False Positive Rate (FP Rate) по хоризонталната ос. Точката (0,1) на ROC кривата представлява идеалният класификатор, който не допуска грешка при предсказването (FP Rate=0 и TP Rate=1). Точка (0,0) съответства на класификатор, който предсказва клас Negative за всички обекти, а точка (1,1) съответства на класификатор, който предсказва клас Positive за всички обекти. С диагонала, свързващ долния ляв ъгъл и горния десен ъгъл на диаграмата, се изобразява работата на случаен класификатор, т.е. на класификационен модел, при който броят на правилно предсказаните обекти от клас Positive е равен на броя на неправилно предсказаните обекти от клас Positive (TP Rate=FP Rate).



Матрица на грешките (Confusion matrix) – квадратна матрица чиито редове и колони, отговарят на класовете. Редовете на матрицата представляват действителните екземпляри от съответния клас, а колоните класовете, получени след прилагането на класификационния модел върху тестовите данни.

Клас	Positive	Negative
Positive	True Positive TP	False Negative FN
Negative	False Positive FP	True Negative TN

4. Изграждане на класификационни модели с Pyhon

За да се изгради модел на класификатор в Python, може да се използва библиотеката scikit-learn (съкращение за "Scientific Kit for Learning"), която предлага различни модели за машинно обучение.

Класът SklearnClassifier от библиотеката scikit-learn в Python е предназначен да бъде използван за прилагане на различни класификационни модели в средата на NLTK (Natural Language Toolkit). Като основен метод, той използва метода fit() на обектите от класовете scikit-learn за обучение на моделите.

SklearnClassifier е наследник на базовия клас ClassifierI в NLTK и за да се използва, трябва да се импортира от модула nltk.classify.scikitlearn. След това може да се създаде обект от класа SklearnClassifier и да се подаде някой от класификационните модели на scikit-learn за обучение и предсказване на данните.

5. Методи на класа SklearnClassifier от библиотеката scikit-learn на Python

train(self, labeled_featuresets): Метод, който прилага обучение на предоставените етикетирани множества от признаци. Приема списък от множества от признаци и техните етикети, където етикетът е включен към множеството от признаци като последен елемент. Методът train() използва метода fit() на обекта на класификационния модел, който е предоставен като аргумент при създаването на обекта на SklearnClassifier. След

като методът train() приключи успешно, моделът ще бъде обучен и готов за предсказване на нови множества от признаци.

classify(self, featureset): Метод, който прилага обучения модел върху множеството от признаци и връща предсказания клас.

accuracy(self, gold): се използва за измерване на точността на класификационния модел, който е обучен с помощта на метода train().
Методът приема етикети на множеството от признаци, за които се знае точният клас (ground truth labels), като аргумент. Тези етикети са обикновено разделени на две групи - обучаващо множество и тестващо множество. Методът сравнява предсказаните етикети с верните етикети (ground truth labels) и измерва колко често предсказанията на модела съвпадат с верните етикети.

predict() се използва, когато моделът е вече обучен и се желае да се направят предсказания за класа на нови данни, които не са били виждани по време на обучението. Когато се извика методът predict() върху модел, той връща предсказани класове за новите данни, като тези предсказания са базирани на параметрите, научени по време на обучението на модела. Методът predict() не приема параметри, освен тези, които са налични в обекта на модела, който го използва. В общия случай, когато се използва модел от класа SklearnClassifier от библиотеката scikit-learn на Python, моделът има различни параметри, които могат да бъдат настроени преди обучението му (например броят на съседите при KNeighborsClassifier или броят на скритите слоеве при MLPClassifier), но тези параметри не се задават като аргументи на метода predict(). Вместо това, методът predict() използва параметрите, научени по време на обучението, за да направи предсказания за нови данни.

6. Алгоритми в Python използвани в задача 1

а) Метод на опорните вектори

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

б) Метод на К най-близкия съсед (KNN - k nearest neighbours)

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

в) Невронна мрежа - Многослоен класификатор Perceptron.

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

7. Допълнителни пояснения към кода на задача 1

а) за ред 9

9) digits = datasets.load_digits()

Наборът от данни MNIST е един от най-известните и широко използвани набори от данни за машинно обучение. Той включва изображения на ръкописни цифри, които са с размер

28x28 пиксела. Всеки пиксел може да има стойност от 0 до 255, като по-високата стойност означава по-тъмно пиксел.

В общия случай, наборът от данни MNIST се използва като задача за класификация. Идеята е да се построи модел за машинно обучение, който да може да разпознава, коя цифра се съдържа на изображението. Наборът от данни се състои от 60 000 обучаващи изображения и 10 000 тестови изображения.

За да се използва набора от данни за ръкописни цифри в Python, се използва библиотеката scikit-learn и функцията **datasets.load_digits()**, която връща като резултат обект, който съдържа изображенията и съответните им етикети (т.е. коя цифра се съдържа на съответното изображение).

б) за ред 10

10) X = digits.images.reshape((len(digits.images), -1)),

В ред 10) , наборът от данни за ръкописни цифри се преобразува, така че всеки елемент (т.е. изображение) да бъде представен като един единствен ред на матрица. Това се прави чрез метода .reshape() на digits.images, който променя формата на масива от изображения, като задава нова форма, която се определя от параметрите на метода.

len(digits.images) е броят на изображенията в набора от данни, а -1 означава, че първото измерение на масива (броят на изображенията) ще остане същото, а за второто измерение (броят на пикселите във всяко изображение) методът ще го изчисли автоматично, така че резултатът да е двумерна матрица с дължина len(digits.images) и ширина, която е равна на броя на пикселите в едно изображение.

в) за ред 11

11) y = digits.target,

В ред 11) се задават етикетите (т.е. класовете) на всеки елемент от набора от данни. digits.target е масив от числа, който показва коя цифра се съдържа на съответното изображение. В контекста на задачата за класификация, това е целевата променлива, която моделът за машинно обучение ще се опита да предскаже.

г) за ред 13

13) X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Кодът на ред 13) се използва, за да раздели набора от данни на тренировъчен и тестов набор.

train_test_split() е функция от библиотеката scikit-learn, която се използвае за случайно разделяне на набора от данни на две части - тренировъчен и тестов. При използване на train_test_split() функцията получава два аргумента - X и y, които представляват матрицата от признаци и целевата променлива, съответно.

test_size=0.3 определя какъв процент от данните ще бъдат използвани за тестов набор. В този случай, test_size=0.3 означава, че 30% от данните ще бъдат използвани за тест.

random_state=42 определя seed за генериране на случайни числа, която се използва при разделянето на данните. Това означава, че ако същият seed бъде използван отново, ще се получи същата разделяне на данните.

Като резултат от изпълнението на train_test_split(), се получават четири нови променливи - X_train, X_test, y_train и y_test, които съдържат съответно тренировъчните и тестовите данни за признаци и целева променлива.

д) за ред 15

15) classifiers = [KNeighborsClassifier(n_neighbors=3),

Кодът на ред 15) дефинира списък от класификатори, които ще бъдат използвани за обучение на модел за класификация на ръкописни цифри.

KNeighborsClassifier е класът за модела на метода на k-най-близките съседи (k-Nearest Neighbors - KNN) в scikit-learn библиотеката за машинно обучение.

n_neighbors=3 задава броя на най-близките съседи, които ще се вземат предвид при класификацията на дадено изображение.

е) за ред 21

21) MLPClassifier(hidden_layer_sizes=(64, 32)) ,

Кодът на ред 21) дефинира модела на класификатора на изкуствена невронна мрежа (Multi-Layer Perceptron - MLP) с два скрити слоя от съответно 64 и 32 неврона.

MLPClassifier е класът за модела на MLP в библиотеката scikit-learn за машинно обучение.

hidden_layer_sizes=(64, 32) параметърът задава броя на невроните във всеки от двата скрити слоя на MLP модела. Тук са зададени два слоя, съставени от 64 и 32 неврона съответно.

MLP е вид изкуствена невронна мрежа, която се състои от поне три слоя: входен слой, скрити слоеве и изходен слой. Всяка невронна мрежа е съставена от много неврони, като всеки неврон получава входни данни, ги обработва и генерира изходни данни. Скритите слоеве позволяват на MLP модела да извършва сложни нелинейни функции на входните данни.

ж) за ред 28

28) classifier.fit(X_train, y_train)

classifier.fit(X_train, y_train) се използва за обучаване на класификатора classifier, като използва данните за обучение X_train и етикетите на класовете y_train. Това означава, че моделът се тренира да прогнозира правилните класове на основата на входните данни X_train. Обучението на модела включва приспособяване на параметрите му към входните данни, така че да може да прави по-добри прогнози за данни, които не са използвани за обучение (например тестови данни). След като моделът е обучен, той може да бъде използван за прогнозиране на класовете на нови данни, които не са били виждани по време на обучението, като се използва метода predict().

з) за ред 53

53) plt.xticks(rotation=90)

Кодът на ред 53) задава ориентацията на етикетите на абсцисната ос в графика, използвайки библиотеката за визуализация на данни Matplotlib. Конкретно, той върти етикетите на абсцисната ос с 90 градуса по посока на часовниковата стрелка.

Задача 1. Класификация на ръкописни цифри

С помощта на класификаторите: Метод на опорните вектори, Метод на К най-близкия съсед, и Невронна мрежа да се изчисли и сравни точността на класификация на ръкописни цифри от набора данни MNIST. Да се визуализира точността на класификаторите с диаграма тип bar (стълбчеста). В програмата на задачата липсват части от кода, които трябва да въведете. Това са: ред 16 (при к =5), ред 17(при к =7), ред 19 (ядро - 'poly'), 20 ядро - rbf), ред 22 (скрити слоеве – 128, 64), ред 23 (скрити слоеве – 256, 128), редове от 34 до 41, ред 51, ред 52 и редове от 54 до 57. (3 точки).

```
1)  from sklearn import datasets
2)  from sklearn.neighbors import KNeighborsClassifier
3)  from sklearn.svm import SVC
4)  from sklearn.neural_network import MLPClassifier
5)  from sklearn.model_selection import train_test_split
6)  from sklearn.metrics import accuracy_score
7)  import matplotlib.pyplot as plt


8)  # Зареждане на данните
9)  digits = datasets.load_digits()
10) X = digits.images.reshape((len(digits.images), -1))
11) y = digits.target


12) # Разделяне на данните на тренировъчни и тестови
13) X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


14) # Използване на различни класификатори
15) classifiers = [KNeighborsClassifier(n_neighbors=3),
16)                                                  ,
17)                                                  ,
18) SVC(kernel='linear'),
19)                                ,
20)                                ,
21) MLPClassifier(hidden_layer_sizes=(64, 32)) ,
22)                                                  ,
```

```
23)                                                    ,
24)  ]


25)  # Обучение на моделите и оценка на точността им
26)  accuracies = []
27)  for classifier in classifiers:
28)  classifier.fit(X_train, y_train)
29)  y_pred = classifier.predict(X_test)
30)  accuracy = accuracy_score(y_test, y_pred)
31)  accuracies.append(accuracy)


32)  # Изобразяване на графика с точностите на различните класификатори
33)  classifiers_names = ['k-NN (k=3)',
34)                                ,
35)                                ,
36)                               ,
37)                              ,
38)                              ,
39)                               ,
40)                               ,
41)                               ,
42)  ]
43)  accuracies = []
44)  for i, classifier in enumerate(classifiers):
45)  classifier.fit(X_train, y_train)
46)  y_pred = classifier.predict(X_test)
47)  accuracy = accuracy_score(y_test, y_pred)
48)  accuracies.append(accuracy)
49)  print(f"{classifiers_names[i]} accuracy: {accuracy}")


50)  # Изобразяване на графика с точностите на различните класификатори
51)
52)
53)  plt.xticks(rotation=90)
54)
55)
56)
57)
```

Резултат:

k-NN (k=3) accuracy: 0.9888888888888889

k-NN (k=5) accuracy: 0.9925925925925926

k-NN (k=7) accuracy: 0.9907407407407407

SVM (linear) accuracy: 0.9796296296296296

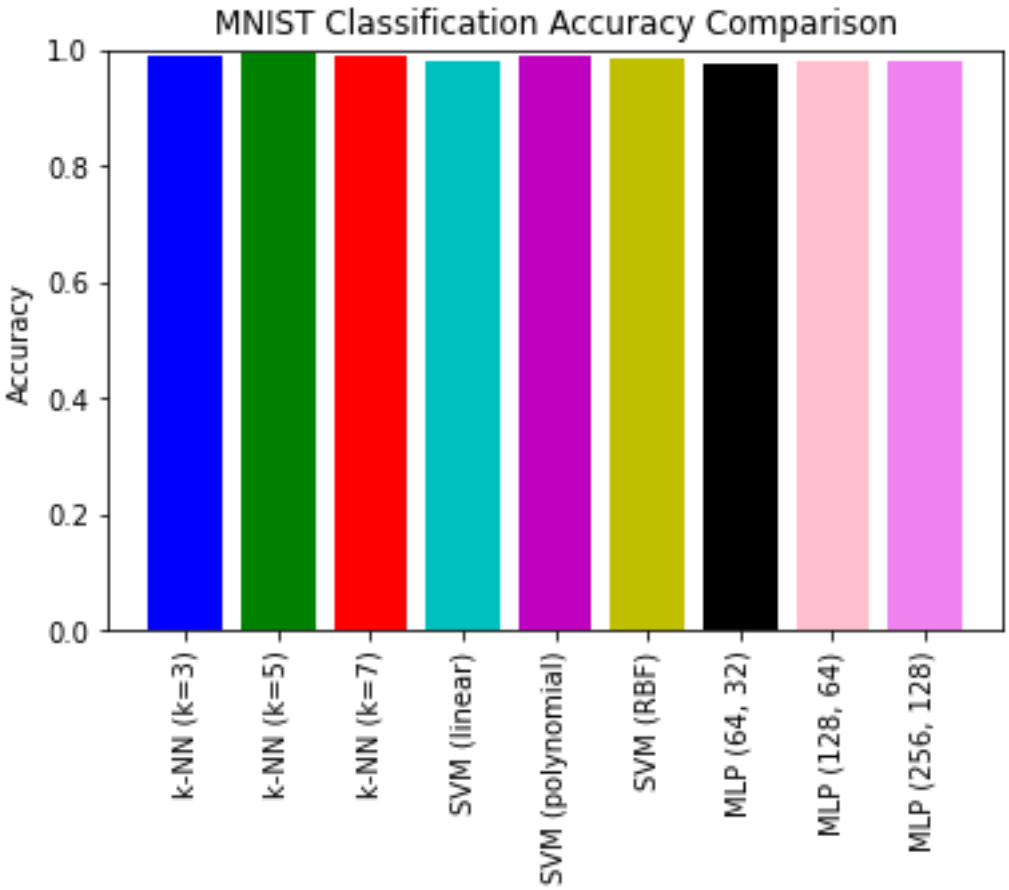
SVM (polynomial) accuracy: 0.9888888888888889

SVM (RBF) accuracy: 0.987037037037037

MLP (64, 32) accuracy: 0.9740740740740741

MLP (128, 64) accuracy: 0.9796296296296296

MLP (256, 128) accuracy: 0.9814814814814815



8. Алгоритъм в Python използван в задача 2

- Гаусов Наивен Бейсов класификатор

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

9. Допълнителни пояснения към кода на задача 2

а) за ред 10

10) data = load_breast_cancer()

На ред 10, методът load_breast_cancer() се извиква, за да се заредят данните за рак на гърдата в променливата data.

б) за ред 11

11) X = data.data

На ред 11, данните за атрибутите на пациентите (признаците) се извличат от data и се съхраняват в променливата X.

в) за ред 12

12) y = data.target

На ред 12, данните за целевата променлива (т.е. информация за това дали пациентът има рак на гърдата или не) се извличат от data и се съхраняват в променливата y.

г) от ред 14 до ред 19

14) stat, p = shapiro(X)

15) print('Shapiro-Wilk Test: Statistics=%.3f, p-value=%.3f' % (stat, p))

16) if p > 0.05:

17) print("Data has a normal distribution")

18) else:

19) print("Data does not have a normal distribution")

Ред 14 използва функцията **shapiro** от модула **scipy.stats** за да извърши тест на нормално разпределение върху данните в **X**. Този тест проверява дали образците в **X** са взети от нормално разпределение. Функцията връща два резултата - статистиката на теста (stat) и p-стойността (p) за теста. Резултатите се присвояват на променливите **stat** и **p**. Тестът се нарича "Shapiro-Wilk test" и се използва за проверка на хипотезата за нормално разпределение на данните. Конкретно в този случай, тестът се използва за проверка на нормалността на данните във вектора X. Резултатът от теста се изразява чрез статистика W и p-стойност. Ако p-стойността е по-малка от избрано ниво на значимост (в случая 0.05), това означава, че няма достатъчно доказателства, за да се отхвърли нулевата хипотеза, че данните имат нормално разпределение. Ако p-стойността е по-голяма от избраното ниво на значимост, то нулевата хипотеза се отхвърля и заключението е, че данните не са нормално разпределени.

Задача 2. Класификация на данни за рак на гърдата (Breast Cancer Wisconsin (Diagnostic)

С помощта на класификатора Гаусов Наивен Бейсов класификатор да се изчислят метриките: точност (accuracy), грешка (error), прецизност (precision), чувствителност (recall), F1-мярка (f1_score), матрица на грешките (confusion_matrix) и ROC AUC (Area

Under the Curve) при класификация на данни за рак на гърдата (Breast Cancer Wisconsin (Diagnostic). Да се визуализира ROC кривата. Да се направи проверка с теста на Тестът на Шапиро-Уилк (Shapiro-Wilk test) за нормалност на данните. В програмата на задачата липсват части от кода, които трябва да въведете. Това са редове: 33, 35, 37, 39, 41, от 47 до 52 ред (3 точки).

```
1) import matplotlib.pyplot as plt
2) import pandas as pd
3) import numpy as np
4) from sklearn.datasets import load_breast_cancer
5) from sklearn.model_selection import train_test_split
6) from sklearn.naive_bayes import GaussianNB
7) from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, f1_score, roc_curve, auc
8) from scipy.stats import shapiro


9) # Зареждане на данните
10) data = load_breast_cancer()
11) X = data.data
12) y = data.target


13) # Проверка за нормално разпределение на данните
14) stat, p = shapiro(X)
15) print('Shapiro-Wilk Test: Statistics=%.3f, p-value=%.3f' % (stat, p))
16) if p > 0.05:
17) print("Data has a normal distribution")
18) else:
19) print("Data does not have a normal distribution")


20) # Разделяне на данните на обучение и тест
21) X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)


22) # Обучение на модела
23) gnb = GaussianNB()
24) gnb.fit(X_train, y_train)


25) # Предсказване на тестовите данни
26) y_pred = gnb.predict(X_test)


27) # Изчисляване на метриките
```

28) #Точност – функция accuracy
29) accuracy = accuracy_score(y_test, y_pred)

30) #класификационна грешка
31) error = 1 - accuracy

32) #прецизност – функция precision_score (подобно като на ред 34)
33) precision =

34) #чувствителност - функция recall_score
35) recall =

36) #F1-мярка – функция f1_score
37) f1 =

38) #матрица на объркване – функция confusion_matrix
39) confusion_mat =

40) # ROC крива – функция roc_curve
41) fpr, tpr, _ =

42) # ROC area
43) roc_auc = auc(fpr, tpr)
44) # Извеждане на резултатите
45) print("\nРЕЗУЛТАТИ:")
46) print("Gaussian Naive Bayes Classifier Metrics:")
47)
48)
49)
50)
51)
52)
53) print("ROC AUC Score:", roc_auc)

54) # чертане на ROC кривата
55) plt.figure()

```
56) plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
57) plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
58) plt.xlim([0.0, 1.0])
59) plt.ylim([0.0, 1.05])
60) plt.xlabel('False Positive Rate')
61) plt.ylabel('True Positive Rate')
62) plt.title('Receiver operating characteristic example')
63) plt.legend(loc="lower right")
64) plt.show()
```

Резултати

Shapiro-Wilk Test: Statistics=0.295, p-value=0.000

Data does not have a normal distribution

РЕЗУЛТАТИ:

Gaussian Naive Bayes Classifier Metrics:

Accuracy: 0.9415204678362573

Classification Error: 0.05847953216374269

Precision: 0.9454545454545454

Recall: 0.9629629629629629

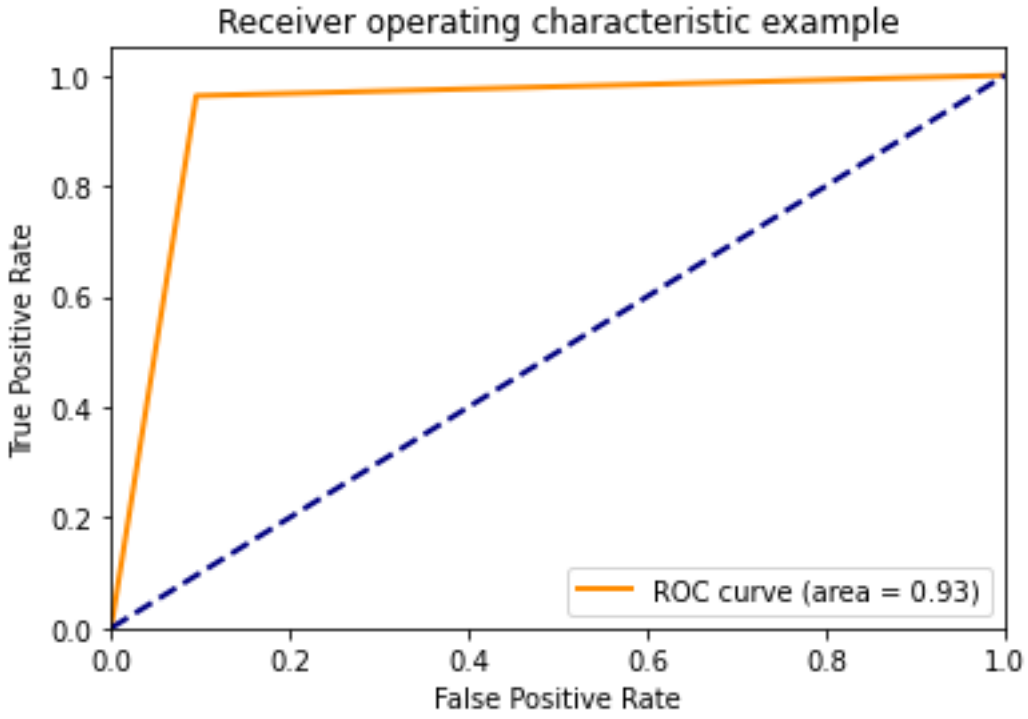
F1 Score: 0.9541284403669724

Confusion Matrix:

[[57 6]

[4 104]]

ROC AUC Score: 0.9338624338624338


Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.93)