Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра системотехніки

Дисципліна: «Методи та системи штучного інтелекту»

ПРАКТИЧНА РОБОТА № 3

«Порівняння методів класифікації»

Виконав:
ст. гр. ІТКН-18-5
Левченко А.С.

Прийняв:
к.т.н., ст. викл. каф.СТ
Жернова П.Є.

Харків 2020

Мета роботи:

Отримання практичних навичок при аналізі даних при використанні методів машинного навчання, а саме методу k-nearest neighbors та Support vector machine

Теоретичні відомості:

Хід роботи:

1. Необхідно імпортувати необхідні пакети і класи: Pandas, Numpy, Sklearn, matplotlib.pyplot.

```
In [1]:
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.impute import SimpleImputer
6 from sklearn.preprocessing import StandardScaler
7 from sklearn.neighbors import KNeighborsClassifier
8 from sklearn.metrics import confusion_matrix, accuracy_score
9 from matplotlib.colors import ListedColormap
```

Рисунок 1 – Імпорт необхідних пакетів

2. Імпортувати набір даних для подальшої роботи.

```
In [19]:
1 dataset = pd.read_csv('Social_Network_Ads.csv')
2 dataset
```

Out[19]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
...	...	...	...
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

400 rows × 3 columns

Рисунок 2 – Імпорт набору даних

3. Розділити набір даних, щоб відповіді були окремо від основних даних.

```
In [3]: X = dataset.iloc[:, :-1].values
        y = dataset.iloc[:, -1].values
```

Рисунок 3 – Розподіл набору даних

4. Перевірити на наявність пропущених даних та заповнити їх.

Пропущених даних в наборі немає.

```
In [4]:  1 dataset.info()
         2 dataset.isnull().sum()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 400 entries, 0 to 399
         Data columns (total 3 columns):
          #   Column           Non-Null Count  Dtype
         ---  ------           --------------  -----
          0   Age              400 non-null    int64
          1   EstimatedSalary  400 non-null    int64
          2   Purchased        400 non-null    int64
         dtypes: int64(3)
         memory usage: 9.5 KB

Out[4]:  Age                0
         EstimatedSalary    0
         Purchased          0
         dtype: int64
```

Рисунок 4 – Перевірка наявності пропущених значень

5. Розділити набір даних на тестову та тренувальну вибірку даних.

Одним із критеріїв якості роботи будь-якої системи МН є якість відповіді на тестувальній вибірці.

Якісна модель на тестувальній вибірці дає досить близькі результати до міток. Погана модель може показувати дуже хороші результати на тренувальній вибірці (відповіді будуть один в один співпадати з прикладами, які ми їй надавали), але на тестувальній вибірці результати будуть не такими хорошими.

```
In [5]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state = 0)
```

Рисунок 5 – розподіл на тренувальну та тестову вибірки

6. Виконати масштабування даних.

Найпростіша трансформація - це Standart Scaling (вона ж Z-score normalization).

$$z = \frac{x - \mu}{\sigma}$$

Перший метод StandartScaling хоч і не робить розподіл нормальним в строгому сенсі слова, але в якійсь мірі захищає від викидів.

```
In [10]: sc = StandardScaler()
         X_train = sc.fit_transform(X_train)
         X_test = sc.transform(X_test)

In [11]: print(X_train)

         [[ 0.58164944 -0.88670699]
          [-0.60673761  1.46173768]
          [-0.01254409 -0.5677824 ]
          [-0.60673761  1.89663484]
          [ 1.37390747 -1.40858358]
          [ 1.47293972  0.99784738]
          [ 0.08648817 -0.79972756]
          [-0.01254409 -0.24885782]
          [-0.21060859 -0.5677824 ]
          [-0.21060859 -0.19087153]
          [-0.30964085 -1.29261101]
          [-0.30964085 -0.5677824 ]
          [ 0.38358493  0.09905991]
          [ 0.8787462  -0.59677555]
          [ 2.06713324 -1.17663843]
          [ 1.07681071 -0.13288524]
          [ 0.68068169  1.78066227]
          [-0.70576986  0.56295021]
          [ 0.77971394  0.35999821]

In [12]: print(X_test)

         [[-0.80480212  0.50496393]
          [-0.01254409 -0.5677824 ]
          [-0.30964085  0.1570462 ]
          [-0.80480212  0.27301877]
          [-0.30964085 -0.5677824 ]
          [-1.10189888 -1.43757673]
          [-0.70576986 -1.58254245]
          [-0.21060859  2.15757314]
```

Рисунок 6 – використання методу StandartScaling

Інший досить популярний варіант - MinMax Scaling, який переносить всі точки на заданий відрізок (зазвичай (0, 1)).

```
In [17]: X_train = MinMaxScaler().fit_transform(X_train)
         X_test = MinMaxScaler().fit_transform(X_test)
         print(X_train)

[[0.61904762 0.17777778]
 [0.33333333 0.77777778]
 [0.47619048 0.25925926]
 [0.33333333 0.88888889]
 [0.80952381 0.04444444]
 [0.83333333 0.65925926]
 [0.5        0.2       ]
 [0.47619048 0.34074074]
 [0.42857143 0.25925926]
 [0.42857143 0.35555556]
 [0.4047619  0.07407407]
 [0.4047619  0.25925926]
 [0.57142857 0.42962963]
 [0.69047619 0.25185185]
 [0.97619048 0.1037037 ]
 [0.73809524 0.37037037]
 [0.64285714 0.85925926]
 [0.30952381 0.54814815]
 [0.66666667 0.4962963 ]

In [18]: print(X_test)

[[0.28571429 0.53333333]
 [0.47619048 0.25925926]
 [0.4047619  0.44444444]
 [0.28571429 0.47407407]
 [0.4047619  0.25925926]
 [0.21428571 0.03703704]
 [0.30952381 0.        ]
 [0.42857143 0.95555556]
 [0.         0.39259259]
```

Рисунок 7 – використання методу MinMax Scaling

StandartScaling і MinMax Scaling мають схожі області застосовності і часто скільки-небудь синоніми.

7. Провести навчання моделі на навчальному наборі даних.

Для навчання класифікатора необхідно мати набір об'єктів, для яких заздалегідь визначені класи. Це безліч називається навчальною вибіркою, її розмітка проводиться вручну.

```
In [20]: 1 classif = KNeighborsClassifier(n_neighbors = 8)
         2 classif.fit(X_train, y_train)

Out[20]: KNeighborsClassifier(n_neighbors=8)
```

Рисунок 8 – навчання моделі за допомогою класу sklearn класу KNeighborsClassifier

Спочатку ми створимо новий класифікатор k-NN і встановимо для n_neighbors значення 8. Нова точка даних позначається більшістю з 3 найближчих точок. Але цей параметр можно замінювати, за допомогою параметра гіпертюнінга.

Параметри гіпертюнінга – це коли ви проходите процес, щоб знайти оптимальні параметри для вашої моделі для підвищення точності. У нашому випадку ми будемо використовувати GridSearchCV, щоб знайти оптимальне значення для n_neighbors.

Параметр гипертюнинга

In [15]:
```
from sklearn.model_selection import GridSearchCV#create new a knn model
knn2 = KNeighborsClassifier()#create a dictionary of all values we want to test for n_neighbors
param_grid = {'n_neighbors': np.arange(1, 25)}#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=5)#fit model to data
knn_gscv.fit(X, y)
```

Out[15]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24])})
```

In [16]:
```
#check top performing n_neighbors value
knn_gscv.best_params_

```

Out[16]:
```
{'n_neighbors': 1}
```

Рисунок 8 – Визначення показника n_neighbors за допомогою параметра гіпертюнінга

8. Зробити прогнозування результатів на тестовому наборі даних.

Можемо знайти матрицю неточностей і точність побудови моделі

In [25]:
```
y_pred = classif.predict(X_test)
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,y_pred)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[64  4]
 [ 3 29]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95        68
           1       0.88      0.91      0.89        32

    accuracy                           0.93       100
   macro avg       0.92      0.92      0.92       100
weighted avg       0.93      0.93      0.93       100

Accuracy: 0.93
```

Рисунок 9 – Прогнозовані значення

9. Отримати візуалізацію результатів.

Візуалізація результату класифікації тренувальних даних методом KNN

Для побудови моделі на навчальному наборі, ми викликаємо метод fit об'єкта knn, який приймає в якості аргументів масив NumPy X_train, що містить навчальні дані, і масив NumPy y_train, відповідний навчальним мітках.

In [66]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 1),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 1))
plt.contourf(X1, X2, classif.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```

Рисунок 11 – Візуалізація тренувальної вибірки методом KNN

In [67]:

```
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_test), y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 1),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 1))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```

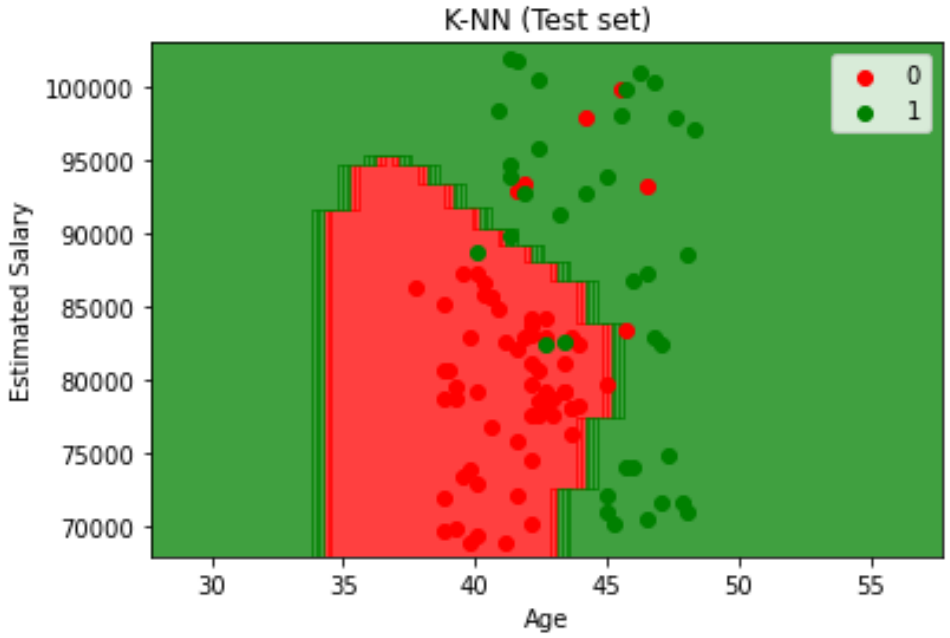

Рисунок 12 – Візуалізація тестової вибірки методом KNN

Візуалізація результату класифікації тренувальних даних методом SVM

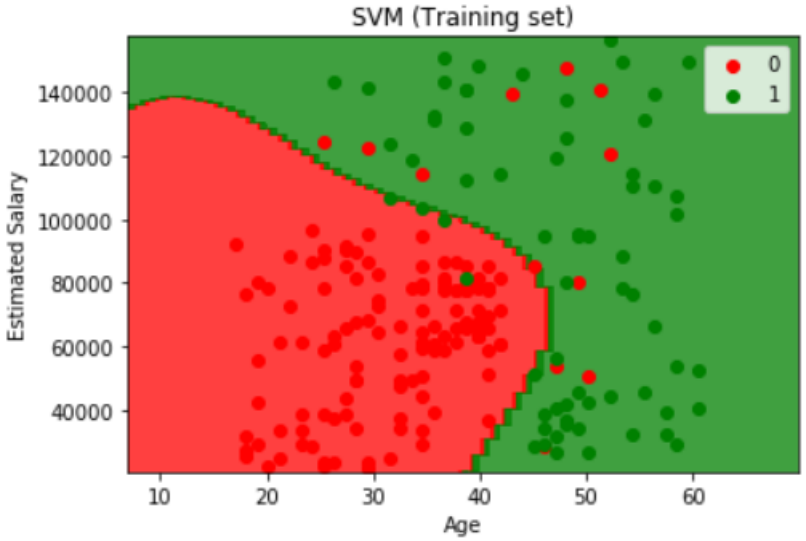

Рисунок 19 – Візуалізація результатів SVM (train)

Рисунок 19 – Візуалізація результатів SVM (train)

В даному випадку SVM дає дуже гладку і нелінійну (непряму) кордон. Тут ми скорегували параметр C і параметр gamma, які зараз детально обговоримо.

Параметр gamma задає ступінь близькості розташування точок. Параметр C являє собою параметр регуляризації, аналогічний тому, що використовувався в лінійних моделях. Він обмежує важливість кожної точки (точніше, її dual_coef_).

Після деякої кількості тестів при різних значеннях параметрів C та gamma виявили що оптимальна комбінація 1000 та ‘auto’ відповідно. Таким чином, оптимальні настройки обох параметрів, як правило, сильно

взаємопов'язані між собою і тому C і gamma повинні бути відрегульовані разом.

10. Порівняти точність класифікації обраних методів.

Часто точність класифікації KNN може бути значно покращена, якщо метрику відстані вивчається за допомогою спеціалізованих 39 алгоритмів, таких як аналіз великих граничних сусідів або компонентів сусідства. Недолік основної класифікації "більшості" відбувається тоді, коли розподіл класів є перекісним. Тобто, приклади більш частого класу мають тенденцію домінувати в прогнозуванні нового прикладу, оскільки вони мають тенденцію бути спільними серед найближчих сусідів через їх велику кількість.

SVM можуть ефективно виконувати нелінійну класифікацію, використовуючи так зване перетворення ядра

Більш формально, SVM створює гіперплощину або набір гіперплощин, який може бути використаний для класифікації, регресії або інших задач, таких як виявлення викидів.

За результатами проведених експериментальних досліджень можна зробити висновок про тому, що використання запропонованого двоетапного методу підвищує якість результатів класифікації, так як застосування kNNклассіфікатора до об'єктів, розташованих поблизу гиперплоскости, що розділяє класи і певної SVM-класифікатором, зменшує число помилково класифікованих об'єктів. Пропонований двоетапний метод класифікації дозволяє приймати високоточні рішення по класифікації складноорганізованих багатовимірних даних.

Висновки:

У ході виконання практичного заняття було отримано практичні навички при аналізі даних при використанні методу машинного навчання, а саме логістичної регресії.

За допомогою алгоритмів KNN ми можемо класифікувати потенційного виборця по різних класах, таким як «Буду голосувати», «Не буду голосувати», «Буду голосувати за партію« Конгрес »,« Буду голосувати за партію «BJP».

Іншими областями, в яких може використовуватися алгоритм KNN, є розпізнавання мови, виявлення почерку, розпізнавання зображень і розпізнавання відео.

SVM дозволяє будувати складні вирішальні кордону, навіть якщо дані містять лише кілька ознак. Вони добре працюють на низькорозмірних і високорозмірних даних (тобто коли у нас мало або, навпаки, багато ознак), проте погано масштабуються з зростанням обсягу даних. Запуск SVM на наборі даних об'ємом 10000 спостережень не складає проблем, однак робота з наборами даних об'ємом 100000 спостережень і більше може стати складним завданням з точки зору часу обчислень і використання пам'яті.