Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра системотехніки

Дисципліна: «Методи та системи штучного інтелекту»

ПРАКТИЧНА РОБОТА № 2

«ЛОГІСТИЧНА РЕГРЕСІЯ»

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 методу машинного навчання, а саме логістичної регресії.

Теоретичні відомості:

Логістична регресія – це алгоритм класифікації машинного навчання, який використовується для прогнозування ймовірності категоріальної залежною змінною. У логістичної регресії залежна змінна є бінарною змінною, що містить дані, закодовані як 1 (так, успіх і т.п.) або 0 (немає, провал і т.п.).

Хід роботи:

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

```
In [1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import MinMaxScaler
```

Рисунок 1 – Імпорт необхідних пакетів

2. Імпортувати набір даних для подальшої роботи.

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

```
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]
 [ 0.8787462  -0.53878926]

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 ]
 [0.69047619 0.26666667]

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]
 [0.69047619 0.29481741]
```

Рисунок 7 – використання методу MinMax Scaling

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

7. Провести навчання моделі логістичної регресії на навчальному наборі даних.

Логістична регресія виводить прогнози про точки в бінарному масштабі, тобто нульовому або одиничному. Якщо значення чого-небудь одно або більше 0.5, то об'єкт класифікується в більшу сторону (до одиниці). Якщо значення менше 0.5 - в меншу (до нуля).

У кожної ознаки є своя мітка, що дорівнює лише 0 або тільки 1. Логістична регресія є лінійним класифікатором і тому використовується, коли в даних простежується якась лінійна залежність.

```
In [14]:  1 classifier = LogisticRegression(random_state = 0)
          2 classifier.fit(X_train, y_train)

Out[14]: LogisticRegression(random_state=0)
```

Рисунок 8 – навчання моделі логістичної регресії

8. Зробити прогнозування результатів на тестовому наборі даних.

Наведені нижче серії 0 і 1 в вихідних даних є прогнозованими значеннями.

```
In [15]:  1 predictions = classifier.predict(X_test)
          2 print(predictions)
          3 print(np.concatenate((predictions.reshape(len(predictions),1), y_test.reshape(len(y_test),1)),1))

[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0
 0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1]
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
```

Рисунок 9 – Прогнозовані значення

9. Перевірити точність прогнозування логістичної регресії.

Можемо знайти матрицю неточностей і точність побудови моделі на попередньому кроці, порівнявши два масиви, а саме y_test і predictions. Будемо використовувати функцію precision_score () для визначення точності.

```
In [16]:
cm = confusion_matrix(y_test, predictions)
print(cm)
print(accuracy_score(y_test,predictions))
```

```
[[67  1]
 [10 22]]
0.89
```

Рисунок 10 – Перевірка точності прогнозування

10. Отримати візуалізацію результатів.

```
In [20]:
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 = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
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('Logistic Regression (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.
```

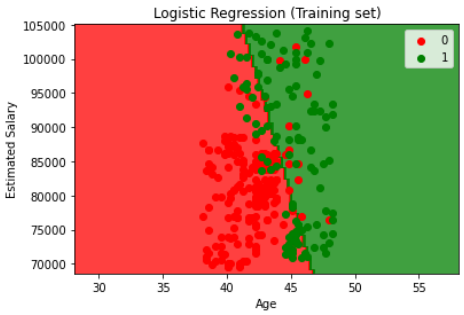

Рисунок 11 – Візуалізація тренувальної вибірки за допомогою логістичної регресії

In [17]:

```
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 = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
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('Logistic Regression (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.
```

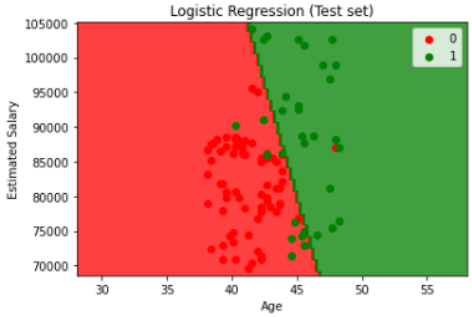

Рисунок 12 – Візуалізація тестової вибірки за допомогою логістичної регресії

Висновки:

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