Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра системотехніки

Дисципліна: «Методи та системи штучного інтелекту»

ПРАКТИЧНА РОБОТА № 4

«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 Naive bayes та Random forest classification

Теоретичні відомості:

Хід роботи:

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

In [1]:
```
import sklearn as sk
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn. metrics import confusion_matrix, accuracy_score
from matplotlib.colors import ListedColormap
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
```

Рисунок 1 – Імпорт необхідних пакетів

2. Імпортувати набір даних для подальшої роботи.

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

Out[2]:

	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.

Якщо зустрінемо пропущені дані, то заповнимо середнім по значенням по стовпцю. Пропущених даних в наборі немає.

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

```
<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	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 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]: 1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=1, stratify=y)
```

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 [9]: 1 sc = StandardScaler()
        2 X_train = sc.fit_transform(X_train)
        3 X_test = sc.transform(X_test)
```

Рисунок 6 – використання методу StandartScaling

7. Провести навчання моделі на навчальному наборі даних.

7.1 Метод Naive bayes

Наївний баесовский алгоритм - це алгоритм класифікацій, заснований на теоремі Баєса з допущенням про незалежність признак. Інші слова, НБА передбачає, що наявність якого-небудь визнання в класі не пов'язане з наявністю якого-небудь іншого визнання.

Моделі на основі НБА досить прості та крайні польові при роботі з дуже великими наборами даних. При своєму простоті НБА здатний перейти навіть до деяких складних алгоритмів класифікації.

В основі НБА лежить, як ви вже могли догадатися, теорема Байеса.

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

Крок 1. Екземпляр класифікатора Naive Bayes

```
In [12]:  1 classifier = GaussianNB()
          2 classifier.fit(X_train, y_train)

Out[12]: GaussianNB()
```

Рисунок 7 - Створення класифікатора та навчання моделі

Крок 2. Прогнозування результатів

```
In [21]:  1 y_pred = classifier.predict(X_test)
          2 print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

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

Рисунок 8 – Прогнозування результатів за допомогою методу Naive Bayes

Крок 3. Перевірити точність класифікації

In [22]:
```
conf_matrix = confusion_matrix(y_test, y_pred)
print("Матриця відповідей:")
print(conf_matrix)
print('Оцінка точності на тестовому наборі:')
print("{:.0%}".format(accuracy_score(y_test,y_pred)))
```

```
Матриця відповідей:
[[59  5]
 [ 6 30]]
Оцінка точності на тестовому наборі:
89%
```

Рисунок 9 – Перевірка точності класифікації за допомогою метода Naive Bayes

Крок 4. Отримати візуалізацію результатів

In [14]:
```
from matplotlib.colors import ListedColormap
from sklearn.preprocessing import StandardScaler
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, 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('Naive Bayes (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 pr
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
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 pr
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
u intend to specify the same RGB or RGBA value for all points.
```

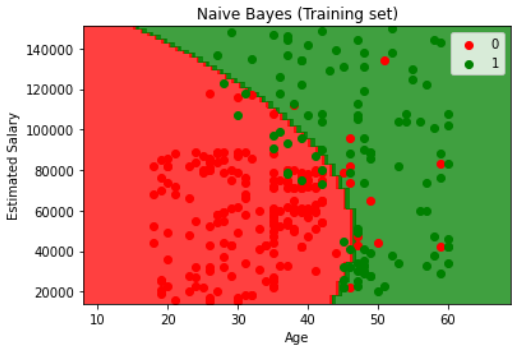

Рисунок 10 – Візуалізація тренувальної вибірки

In [15]:

```
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('Naive Bayes (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 pr
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
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 pr
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
u intend to specify the same RGB or RGBA value for all points.
```

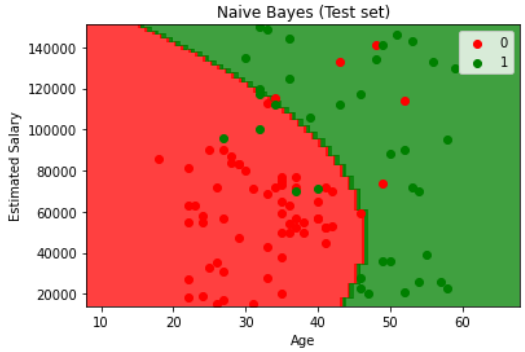

Рисунок 11 – Візуалізація тестової вибірки

7.2 Метод Random forest

Алгоритм Random Forest – ансамблевый метод машинного обучения, который использует ансамбль деревьев решений. Он основывается на основных подходахГбэггинга и выбора случайных подмножествФпризнаков. Этот алгоритм позволяет достичь высокой точности классификации. Деревья в ансамбле строятся друг от друга независимо.

Крок 1. Екземпляр класифікатора Random forest

In [16]:

```
rf = RandomForestClassifier(n_estimators=20)
rf.fit(X_train, y_train)
```

Out[16]:

```
RandomForestClassifier(n_estimators=20)
```

Рисунок 12 - Створення класифікатора та навчання моделі

Крок 2. Прогнозування результатів

```
In [23]:
1 y_pred = rf.predict(X_test)
2 print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

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

Рисунок 13 – Прогнозування результатів за допомогою методу Random forest

Крок 3. Перевірити точність класифікації

```
In [17]:
1 conf_matrix = confusion_matrix(y_test, y_pred)
2 print("Матриця відповідей:")
3 print(conf_matrix)
4 print('Оцінка точності на тестовому наборі:')
5 print("{:.0%}".format(accuracy_score(y_test,y_pred)))
```

```
Матриця відповідей:
[[60  4]
 [ 5 31]]
Оцінка точності на тестовому наборі:
91%
```

Рисунок 14 – Перевірка точності класифікації за допомогою метода Random forest

Крок 4. Отримати візуалізацію результатів

In [18]:
```
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, rf.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('Random forest (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 15 – Візуалізація тестової вибірки

In [19]:
```
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, rf.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('Random forest(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.
```

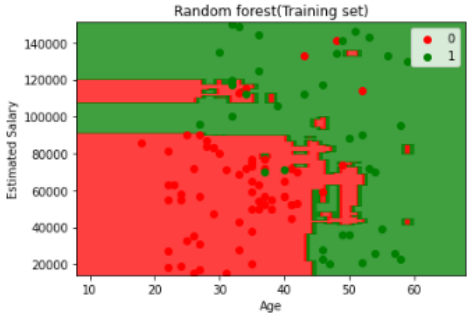

Рисунок 16 – Візуалізація тестової вибірки

8. Порівняти точність класифікації обраних методів.

Метод Байєса має високу швидкість роботи і простотою математичної моделі. Цей метод часто використовується в якості базового методу при порівнянні різних методів машинного навчання.

Класифікація методом Байеса показала себе гірше, ніж метод RF, точність такого методу становить 89%, а точність методу RF 91%.

```
In [24]:
1 Accuracy = [Accuracy_RF, Accuracy_NB]
2 Methods = ['Random_Forest', 'Naive_Bayes']
3 Accuracy_pos = np.arange(len(Methods))
4 plt.bar(Accuracy_pos, Accuracy)
5 plt.xticks(Accuracy_pos, Methods)
6 plt.title('Порівняння точності кожної моделі')
7 plt.show()
8
```

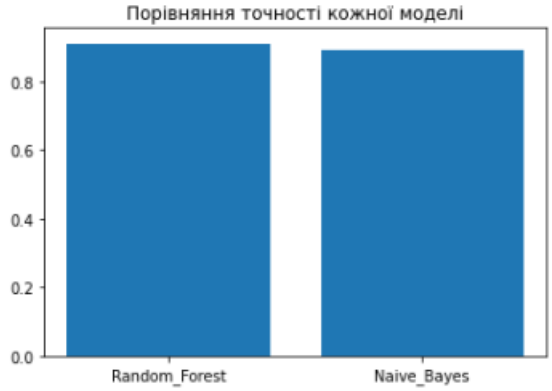

Рисунок 16 – Порівняння точності методу Random forest та Naive Bayes

Висновки:

У ході виконання практичного заняття було отримано практичні навички при аналізі даних при використанні методу машинного навчання, а саме, Naive bayes та Random forest classification.

Порівнюючи точність класифікації методів з цієї практичної роботи та з попередньої:

Random forest – 89%

Naive Bayes – 91%

KNN – 93%

SVM – 93%

Можемо сказати, що метод Random forest виявився найменш точним. Що до цього алгоритму, можемо додати що перебір кожного розбиття значень ознак являється обчислювально затратним завданням і має обчислювальну складність порядку O (2^q), Де q - кількість прийнятих ознакою значень, то на

практиці частіше використовується варіант з перебором деякої довільної частини розбиття.

Naive Bayes дозволяє розв’язувати задачі класифікації, де необхідно приймати рішення про приналежність об'єкту до одного з виділених класів, аналізуючи одночасно значення декількох окремих ознак цього об'єкту. Наївна байесівська класифікація використовує теорему Байеса, щоб визначити, наскільки ймовірно, що елемент є членом категорії. Наївний байесовский класифікатор сортує елементи за категоріями на основі того, яка ймовірність є найбільшою.