Лабораторна робота №1

Основи штучного інтелекту

Виконав: ІПЗ-21-3 Осипчук Антон Олексійович

Github: https://github.com/AntonOsypchuk1/ai_lab/tree/main/lab1

2.1. Лістинг програми:

task1.1.py > ...

```python
import numpy as np
from sklearn import preprocessing

input_data = np.array([[5.1, -2.9, 3.3],
                       [-1.2, 7.8, -6.1],
                       [3.9, 0.4, 2.1],
                       [7.3, -9.9, -4.5]])

# Binarize data
data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
print("\nBinarized data:\n", data_binarized)

# Print mean and standard deviation
print("\nBEFORE:")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))

# Remove mean
data_scaled = preprocessing.scale(input_data)
print("\nAFTER:")
print("Mean =", data_scaled.mean(axis=0))
print("Std deviation =", data_scaled.std(axis=0))

# Min max scaling
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print("\nMin max scaled data:\n", data_scaled_minmax)

# Normalize data
data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
print("\nL1 normalized data:\n", data_normalized_l1)
print("\nL2 normalized data:\n", data_normalized_l2)
```

Результат виконання:

```
(venv) antonosypchuk@MacBook-Air-Anton lab1 % python3 task1.1.py

Binarized data:
 [[1. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]

BEFORE:
Mean = [ 3.775 -1.15  -1.3  ]
Std deviation = [3.12039661 6.36651396 4.0620192 ]

AFTER:
Mean = [1.11022302e-16 0.00000000e+00 2.77555756e-17]
Std deviation = [1. 1. 1.]

Min max scaled data:
 [[0.74117647 0.39548023 1.        ]
 [0.         1.         0.        ]
 [0.6        0.5819209  0.87234043]
 [1.         0.         0.17021277]]

L1 normalized data:
 [[ 0.45132743 -0.25663717  0.2920354 ]
 [-0.0794702   0.51655629 -0.40397351]
 [ 0.609375    0.0625      0.328125  ]
 [ 0.33640553 -0.4562212  -0.20737327]]

L2 normalized data:
 [[ 0.75765788 -0.43082507  0.49024922]
 [-0.12030718  0.78199664 -0.61156148]
 [ 0.87690281  0.08993875  0.47217844]
 [ 0.55734935 -0.75585734 -0.34357152]]
```

Висновок:

Нормалізація L1 зазвичай зводить ваги несуттєвих ознак до нуля, тоді як нормалізація L2 зберігає відносну пропорцію значень ознак.

2.1.5. Лістинг програми

task1.2.py > ...

```python
import numpy as np
from sklearn import preprocessing

Input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow', 'white']

# Create label encoder and fit the labels
encoder = preprocessing.LabelEncoder()
encoder.fit(Input_labels)

# Print the mapping
print("\nLabel mapping:")
for i, item in enumerate(encoder.classes_):
    print(item, '-->', i)

# Encode a set of labels using the encoder
test_labels = ['green', 'red', 'black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
print("Encoded values =", list(encoded_values))

# Decode a set of values using the encoder
encoded_values = [3, 0, 4, 1]
decoded_list = encoder.inverse_transform(encoded_values)
print("\nEncoded values =", encoded_values)
print("Decoded labels =", list(decoded_list))
```

Результат виконання:

```
(venv) antonosypchuk@MacBook-Air-Anton lab1 % python3 task1.2.py

Label mapping:
black --> 0
green --> 1
red --> 2
white --> 3
yellow --> 4

Labels = ['green', 'red', 'black']
Encoded values = [np.int64(1), np.int64(2), np.int64(0)]

Encoded values = [3, 0, 4, 1]
Decoded labels = [np.str_('white'), np.str_('black'), np.str_('yellow'), np.str_('green')]
```

2.2. Лістинг програми:

task2.py > ...

```python
import numpy as np
from sklearn import preprocessing

input_data = np.array([[-2.3, 3.9, -4.5], [-5.3, -4.2, -1.3], [5.2, -6.5, -1.1], [-5.2, 2.6, -2.2]])

# Binarize data
data_binarized = preprocessing.Binarizer(threshold=2.1).transform(input_data)
print("\nBinarized data:\n", data_binarized)

# Print mean and standard deviation
print("\nBEFORE:")
print("Mean =", input_data.mean(axis=0))
print("Std deviation =", input_data.std(axis=0))

# Remove mean
data_scaled = preprocessing.scale(input_data)
print("\nAFTER:")
print("Mean =", data_scaled.mean(axis=0))
print("Std deviation =", data_scaled.std(axis=0))

# Min max scaling
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0, 1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print("\nMin max scaled data:\n", data_scaled_minmax)

# Normalize data
data_normalized_l1 = preprocessing.normalize(input_data, norm='l1')
data_normalized_l2 = preprocessing.normalize(input_data, norm='l2')
print("\nL1 normalized data:\n", data_normalized_l1)
print("\nL2 normalized data:\n", data_normalized_l2)

```

Результат виконання:

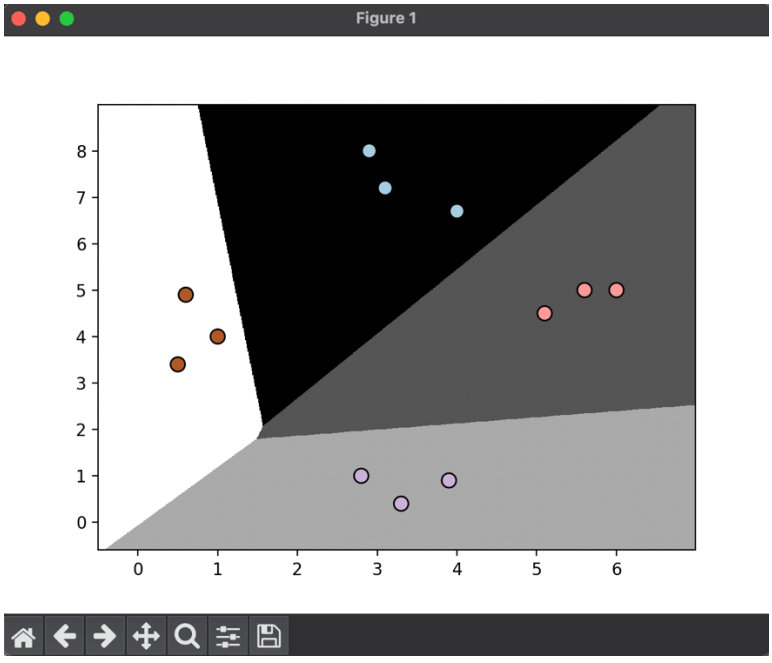
```
(venv) antonosypchuk@MacBook-Air-Anton lab1 % python3 task2.py

Binarized data:
 [[0. 1. 0.]
 [0. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]]

BEFORE:
Mean = [-1.9   -1.05  -2.275]
Std deviation = [4.27258704 4.40028408 1.3497685 ]

AFTER:
Mean = [-2.77555756e-17  5.55111512e-17  2.09901541e-16]
Std deviation = [1. 1. 1.]

Min max scaled data:
 [[0.28571429 1.         0.        ]
 [0.         0.22115385 0.94117647]
 [1.         0.         1.        ]
 [0.00952381 0.875      0.67647059]]

L1 normalized data:
 [[-0.21495327  0.36448598 -0.42056075]
 [-0.49074074 -0.38888889 -0.12037037]
 [ 0.40625    -0.5078125  -0.0859375 ]
 [-0.52        0.26       -0.22      ]]

L2 normalized data:
 [[-0.36029981  0.61094315 -0.7049344 ]
 [-0.76965323 -0.60991388 -0.18878287]
 [ 0.61931099 -0.77413873 -0.13100809]
 [-0.83653629  0.41826814 -0.3539192 ]]
```

2.3. Лістинг програми:

task3.py > ...

```python
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
from utilities import visualize_classifier

# Визначення зразка вхідних даних
X = np.array([[3.1, 7.2], [4, 6.7], [2.9, 8], [5.1, 4.5],
[6, 5], [5.6, 5], [3.3, 0.4],
[3.9, 0.9], [2.8, 1],
[0.5, 3.4], [1, 4], [0.6, 4.9]])
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])

# Створення логістичного класифікатора
classifier = linear_model.LogisticRegression(solver='liblinear',C=1)

# Тренування класифікатора
classifier.fit(X, y)

visualize_classifier(classifier, X, y)
```

Результат виконання:



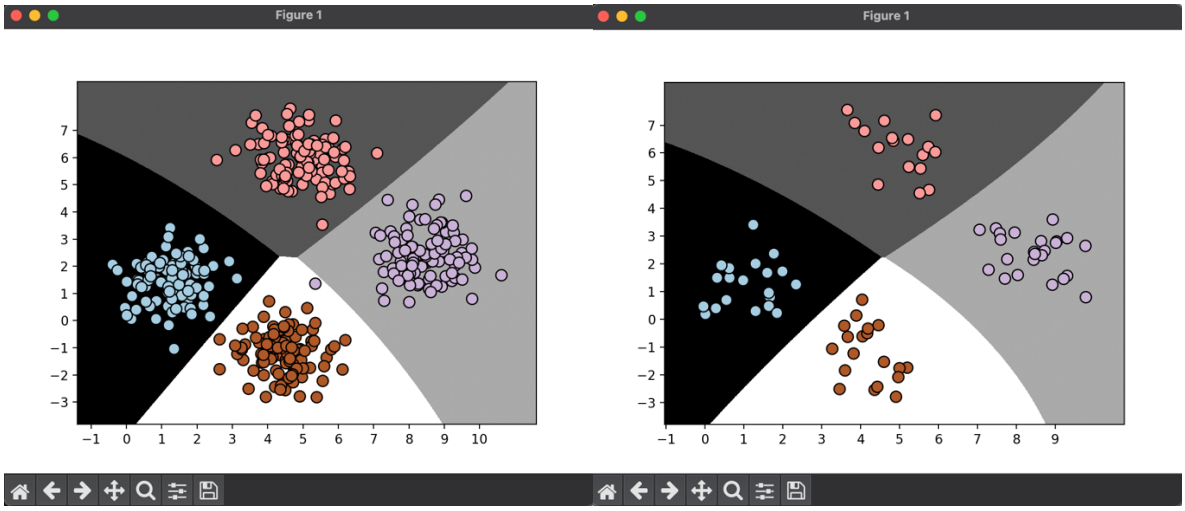
2.4. Лістинг програми:

task4.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split, cross_val_score
from utilities import visualize_classifier

# Вхідний файл, який містить дані
input_file = 'data/data_multivar_nb.txt'

# Завантаження даних із вхідного файлу
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Створення наївного байєсовського класифікатора
classifier = GaussianNB()

# Тренування класифікатора
classifier.fit(X, y)

# Прогнозування значень для тренувальних даних
y_pred = classifier.predict(X)

# Обчислення якості класифікатора
accuracy = 100.0 * (y == y_pred).sum() / X.shape[0]
print("Accuracy of Naive Bayes classifier =", round(accuracy, 2), "%")

# Візуалізація результатів роботи класифікатора
visualize_classifier(classifier, X, y)

# Розбивка даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)
classifier_new = GaussianNB()
classifier_new.fit(X_train, y_train)
y_test_pred = classifier_new.predict(X_test)
```

```python
# Обчислення якості класифікатора
accuracy = 100.0 * (y_test == y_test_pred).sum() / X_test.shape[0]
print("Accuracy of the new classifier =", round(accuracy, 2), "%")
# Візуалізація роботи класифікатора
visualize_classifier(classifier_new, X_test, y_test)

num_folds = 3

accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")

precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")

recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")

f1_values = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=num_folds)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
```

Результат виконання:




```
(venv) antonosypchuk@MacBook-Air-Anton lab1 % python3 task4.py
Accuracy of Naive Bayes classifier = 99.75 %
2024-11-25 10:49:58.532 Python[68128:4786796] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-25 10:49:58.532 Python[68128:4786796] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Accuracy of the new classifier = 100.0 %
Accuracy: 99.75%
Precision: 99.76%
Recall: 99.75%
F1: 99.75%
```

2.5. Лістинг програми:

utilities.py > ...

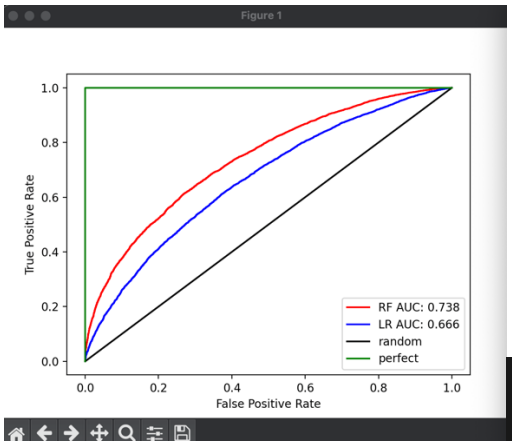
```
40
41  def find_TP(y_true, y_pred):
42      return sum((y_true == 1) & (y_pred == 1))
43
44  def find_FN(y_true, y_pred):
45      return sum((y_true == 1) & (y_pred == 0))
46
47  def find_FP(y_true, y_pred):
48      return sum((y_true == 0) & (y_pred == 1))
49
50  def find_TN(y_true, y_pred):
51      return sum((y_true == 0) & (y_pred == 0))
52
53  def find_confusion_matrix_values (y_true, y_pred):
54      TP = find_TP(y_true, y_pred)
55      FN = find_FN(y_true, y_pred)
56      FP = find_FP(y_true, y_pred)
57      TN = find_TN(y_true, y_pred)
58      return TP, FN, FP, TN
```

utilities.py > ...

```python
def osypchuk_confusion_matrix(y_true, y_pred):
    TP, FN, FP, TN = find_confusion_matrix_values (y_true, y_pred)
    return np.array([[TN, FP], [FN, TP]])

def osypchuk_accuracy_score(y_true, y_pred):
    TP, FN, FP, TN = find_confusion_matrix_values (y_true, y_pred)
    return (TP + TN) / (TP + FP + FN + TN)

def osypchuk_recall_score(y_true, y_pred):
    TP, FN, FP, TN = find_confusion_matrix_values(y_true, y_pred)
    return TP / (TP + FN)

def osypchuk_precision_score(y_true, y_pred):
    TP, FN, FP, TN = find_confusion_matrix_values(y_true, y_pred)
    return TP / (TP + FP)

def osypchuk_f1_score(y_true, y_pred):
    precision = osypchuk_precision_score(y_true, y_pred)
    recall = osypchuk_recall_score(y_true, y_pred)
    return 2 * precision * recall / (precision + recall)
```

Результат виконання:

```
(venv) antonosypchuk@MacBook-Air-Anton lab1 % python3 task5.py
TP: 5047
FN: 2832
FP: 2360
TN: 5519
Accuracy RF: 0.671
Accuracy LR: 0.616
Recall RF: 0.641
Recall LR: 0.543
Precision RF: 0.681
Precision LR: 0.636
F1 RF: 0.660
F1 LR: 0.586
scores with threshold = 0.5
Accuracy RF: 0.671
Recall RF: 0.641
Precision RF: 0.681
F1 RF: 0.660

scores with threshold = 0.25
Accuracy RF: 0.502
Recall RF: 1.000
Precision RF: 0.501
F1 RF: 0.668
2024-11-25 12:32:20.592 Python[90099:4962152] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-25 12:32:20.592 Python[90099:4962152] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```




```
AUC RF:0.738
AUC LR:0.666
```

Висновки до завдання:
Зниження порогу класифікації зазвичай призводить до того, що більше даних потрапляє до позитивного класу. Це може збільшити кількість істинно позитивних (True Positive) та хибно позитивних (False Positive) результатів, а також зменшити кількість істинно негативних (True Negative) та хибно негативних (False Negative) випадків. Якщо приріст істинно позитивних і зменшення хибно негативних значно перевищує приріст хибно позитивних і зменшення істинно негативних, загальна точність може покращитися. Проте, якщо зростання помилкових класифікацій (False Positive і False Negative) домінує, точність може знизитися.

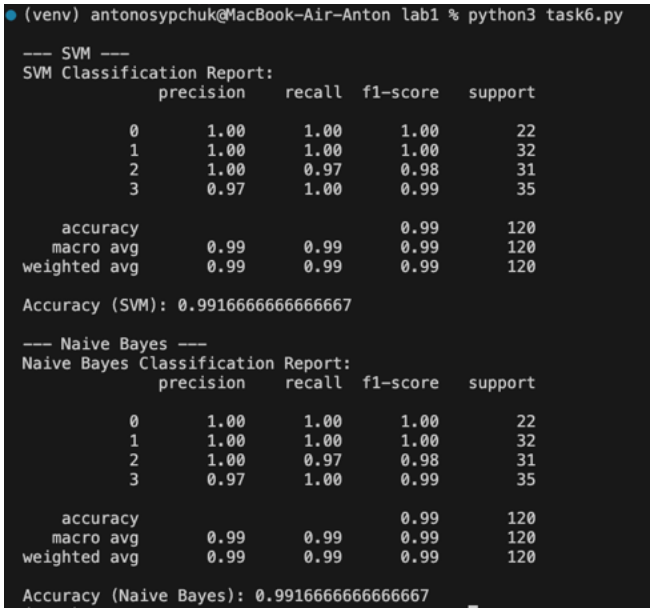
Вибір моделі залежить від того, які показники є більш важливими для задачі. Наприклад, якщо потрібно досягти високого рівня прецизійності (Precision), доцільно обрати модель з максимальним значенням цього показника, навіть якщо її повнота (Recall) або точність (Accuracy) будуть нижчими. Якщо ж завдання вимагає балансу між прецизійністю і повнотою, оптимальним вибором буде модель із найвищим показником F1-міри.

2.6. Лістинг програми:

task6.py > ...

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score

data = pd.read_csv("data/data_multivar_nb.txt")

X = data.iloc[:, :-1]
y = data.iloc[:, -1]

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

print("\n--- SVM ---")
svm_model = SVC(kernel="rbf", random_state=42)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)

print("SVM Classification Report:")
print(classification_report(y_test, y_pred_svm))
print("Accuracy (SVM):", accuracy_score(y_test, y_pred_svm))

print("\n--- Naive Bayes ---")
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_test)

print("Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_nb))
print("Accuracy (Naive Bayes):", accuracy_score(y_test, y_pred_nb))

```

Результат виконання:


(venv) antonosypchuk@MacBook-Air-Anton lab1 % python3 task6.py
--- SVM ---
SVM Classification Report:
precision recall f1-score support
0 1.00 1.00 1.00 22
1 1.00 1.00 1.00 32
2 1.00 0.97 0.98 31
3 0.97 1.00 0.99 35
accuracy 0.99 120
macro avg 0.99 0.99 0.99 120
weighted avg 0.99 0.99 0.99 120
Accuracy (SVM): 0.9916666666666667
--- Naive Bayes ---
Naive Bayes Classification Report:
precision recall f1-score support
0 1.00 1.00 1.00 22
1 1.00 1.00 1.00 32
2 1.00 0.97 0.98 31
3 0.97 1.00 0.99 35
accuracy 0.99 120
macro avg 0.99 0.99 0.99 120
weighted avg 0.99 0.99 0.99 120
Accuracy (Naive Bayes): 0.9916666666666667