

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій
Кафедра системного проектування

Звіт
Про виконання лабораторної роботи №4
З курсу «Системи машинного навчання»
Класифікаційні моделі

Виконала:
Студентка групи ФЕС-32
Філь Дарина

Перевірив:
Доцент Колич І.І.

Львів 2024

Мета: Навчитися будувати та оцінювати класифікаційні моделі.

Інструменти: Python, Scikit-learn, Matplotlib, Seaborn.

Теоретичні відомості

Логістична регресія

Логістична регресія - це статистичний метод для бінарної класифікації, який оцінює ймовірність того, що заданий вхід належить до певного класу.

Формула логістичної регресії

Логістична регресія використовує логістичну функцію (сигмоїдну), щоб перетворити лінійну комбінацію незалежних змінних на ймовірності:

$$P(y = 1|x) = \frac{1}{1 + e^{-(b+w_1x_1+w_2x_2+\dots+w_nx_n)}}$$

де:

- (y) — залежна змінна;
- (b) — вільний член (intercept);
- (w_1, \dots, w_n) — коефіцієнти регресії для кожної незалежної змінної;
- (x_1, \dots, x_n) — незалежні змінні;

Переваги логістичної регресії

- Легко інтерпретована: результати можна інтерпретувати як ймовірності;
- Обчислювально ефективна: може бути швидко навчена навіть на великих наборах даних;
- Відмінно працює на лінійно роздільних даних: добре підходить, якщо класи можуть бути розділені лінійно;

Недоліки логістичної регресії

- Лінійність: не в змозі моделювати складні нелінійні взаємозв'язки;
- Чутливість до шуму: може бути чутливою до випадкових шумів у даних;

Дерева рішень

Дерева рішень - це ієрархічна модель класифікації, яка використовує правила ухвалення рішень, побудовані на основі ознак даних, щоб передбачити результат.

Побудова дерева рішень

Дерева рішень використовують рекурсивне розбиття, щоб створити модель у вигляді дерева. На кожному вузлі дерева вибирається ознака, за якою дані будуть розбиті на дві або більше підмножини.

Переваги дерев рішень

- Легко інтерпретовані: моделі у вигляді дерева можуть бути легко візуалізовані та інтерпретовані;
- Не потребують масштабування даних: дерева рішень управління з ознаками, які не потребують нормалізації або стандартизації;
- Працюють з категоріальними змінними: можуть легко працювати з категоріальними даними без потреби в додаткових перетвореннях;

Недоліки дерев рішень

- Перенавчання: дерева рішень часто схильні до перенавчання, особливо коли вони є глибокими;
- Чутливість до змін у даних: незначні зміни в даних можуть призвести до побудови зовсім іншого дерева;

Оцінка моделей

Точність (Accuracy)

Визначення: Точність (Accuracy) є часткою правильно передбачених випадків серед усіх випадків.

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

де:

- TP (True Positive) — кількість правильно передбачених позитивних випадків;
- TN (True Negative) — кількість правильно передбачених негативних випадків;
- FP (False Positive) — кількість неправильно передбачених позитивних випадків;
- FN (False Negative) — кількість неправильно передбачених негативних випадків;

Переваги

- Легкість інтерпретації;

Недоліки

- Не підходить для незбалансованих наборів даних;

Влучність (Precision)

Визначення: Частка правильно передбачених позитивних випадків серед усіх випадків, які модель передбачила як позитивні.

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

Переваги

- Важлива, коли помилкові позитивні передбачення мають високі витрати;

Недоліки

- Не враховує помилкові негативні передбачення;

Повнота (Recall)

Визначення: Частка правильно передбачених позитивних випадків серед усіх фактичних позитивних випадків.

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

Переваги

- Важлива, коли помилкові негативні передбачення мають високі витрати;

Недоліки

- Не враховує помилкові позитивні передбачення;

F1-score

Визначення: Гармонічне середнє між Precision та Recall.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Переваги

- Комбінує Precision та Recall, забезпечуючи баланс між ними;

Недоліки

- Не завжди легко інтерпретувати;

Матриця невідповідностей (Confusion Matrix)

Визначення: Матриця невідповідностей є таблицею, яка дозволяє візуалізувати продуктивність алгоритму класифікації. Вона відображає кількість правильних та неправильних передбачень, розподілених за кожним класом.

CONFUSION MATRIX		PREDICTED LABELS	
		POSITIVE	NEGATIVE
TRUE LABELS	POSITIVE	TP	FN
	NEGATIVE	FP	TN

Переваги

- Забезпечує повний огляд продуктивності моделі;
- Допомогає виявити патерни помилок;

Недоліки

- Не дає безпосереднього уявлення про баланс між Precision та Recall;

Набір даних Iris

Набір даних Iris є класичним набором даних для задач класифікації. Він містить інформацію про довжину та ширину чашолистків і пелюсток для трьох видів ірисів (Iris setosa, Iris versicolor, Iris virginica).

Опис набору даних

- **Кількість зразків:** 150
- **Кількість ознак:** 4
- **Кількість класів:** 3 (три різні види ірисів)

Ознаки

1. **Sepal length (cm):** довжина чашолистка
2. **Sepal width (cm):** ширина чашолистка
3. **Petal length (cm):** довжина пелюстки
4. **Petal width (cm):** ширина пелюстки

Класи

Цільова змінна (клас) вказує на вид ірису та має три можливі значення:

- 0: Iris setosa
- 1: Iris versicolor

- 2: Iris virginica

Хід роботи

ЗАВДАННЯ

1. Підготовка даних

- 1.1. Використайте набір даних Iris.
- 1.2. Розділіть дані на тренувальний (80%) та тестовий (20%) набори.

2. Логістична регресія

- 2.1. Навчіть модель логістичної регресії на тренувальних даних.
- 2.2. Виконайте прогнозування на тестових даних.
- 2.3. Оцініть результати за допомогою метрик якості моделі: точність (Accuracy), Precision, Recall, F1-score.
- 2.4. Візуалізуйте матрицю невідповідностей.

3. Деревя рішень

- 3.1. Навчіть модель дерева рішень на тренувальних даних.
- 3.2. Виконайте прогнозування на тестових даних.
- 3.3. Оцініть результати за допомогою метрик якості моделі: точність (Accuracy), Precision, Recall, F1-score.
- 3.4. Візуалізуйте матрицю невідповідностей.

4. Повторити навчання та оцінку для Fashion-MNIST набору даних

- 4.1. Рекомендовано використовувати PyTorch backend для Keras.
- 4.2. Порівняти зміни в моделях при використанні іншого набору

5. Оформити звіт

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import numpy as np
from sklearn.model_selection import cross_val_score
import seaborn as sns
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.tree import DecisionTreeClassifier
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.optim as optim

iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

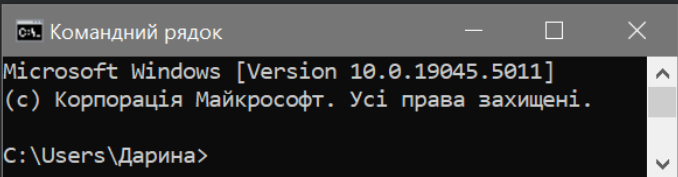


Рис. 1 Завантажування бібліотек та поділ даних

```
log_reg = LogisticRegression(max_iter=200)
log_reg.fit(X_train, y_train)
y_pred_logreg = log_reg.predict(X_test)

accuracy = accuracy_score(y_test, y_pred_logreg)
precision = precision_score(y_test, y_pred_logreg, average='macro')
recall = recall_score(y_test, y_pred_logreg, average='macro')
f1 = f1_score(y_test, y_pred_logreg, average='macro')

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")

ConfusionMatrixDisplay.from_estimator(log_reg, X_test, y_test, display_labels=iris.target_names)
plt.show()
```

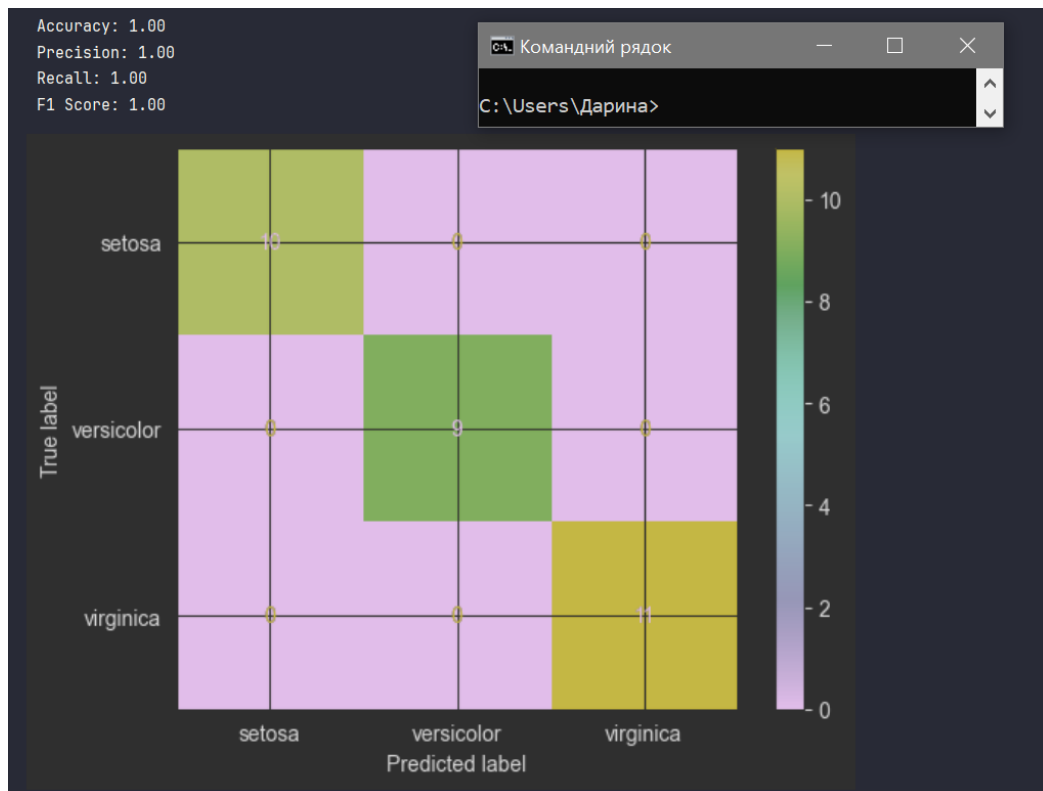
Executed at 2024.10.27 13:54:47 in 64ms

Add Code Cell Add Markdown Cell

Командний рядок
Microsoft Windows [Version 10.0.19045.5011]
(c) Корпорація Майкрософт. Усі права захищені.
C:\Users\Дарина>

Executed at 2024.10.27 13:54:48 in 440ms

Рис. 2 Тренування моделі логістичної регресії



```
Logistic Regression Cross-Val Accuracy: 0.9733333333333334
Decision Tree Cross-Val Accuracy: 0.9600000000000002
Training Accuracy (LogReg): 0.975
Test Accuracy (LogReg): 1.0
Training Accuracy (Tree): 1.0
Test Accuracy (Tree): 1.0
```

Рис. 3-4 Результати тренування моделі логістичної регресії та її оцінка показують, що модель, натренована на датасеті Iris, має фактично ідеальну точність і F1-score. Це може свідчити про можливе перенасичення даними. Для перевірки цього було проведено крос-валідацію, яка також показала фактично ідеальну точність, що

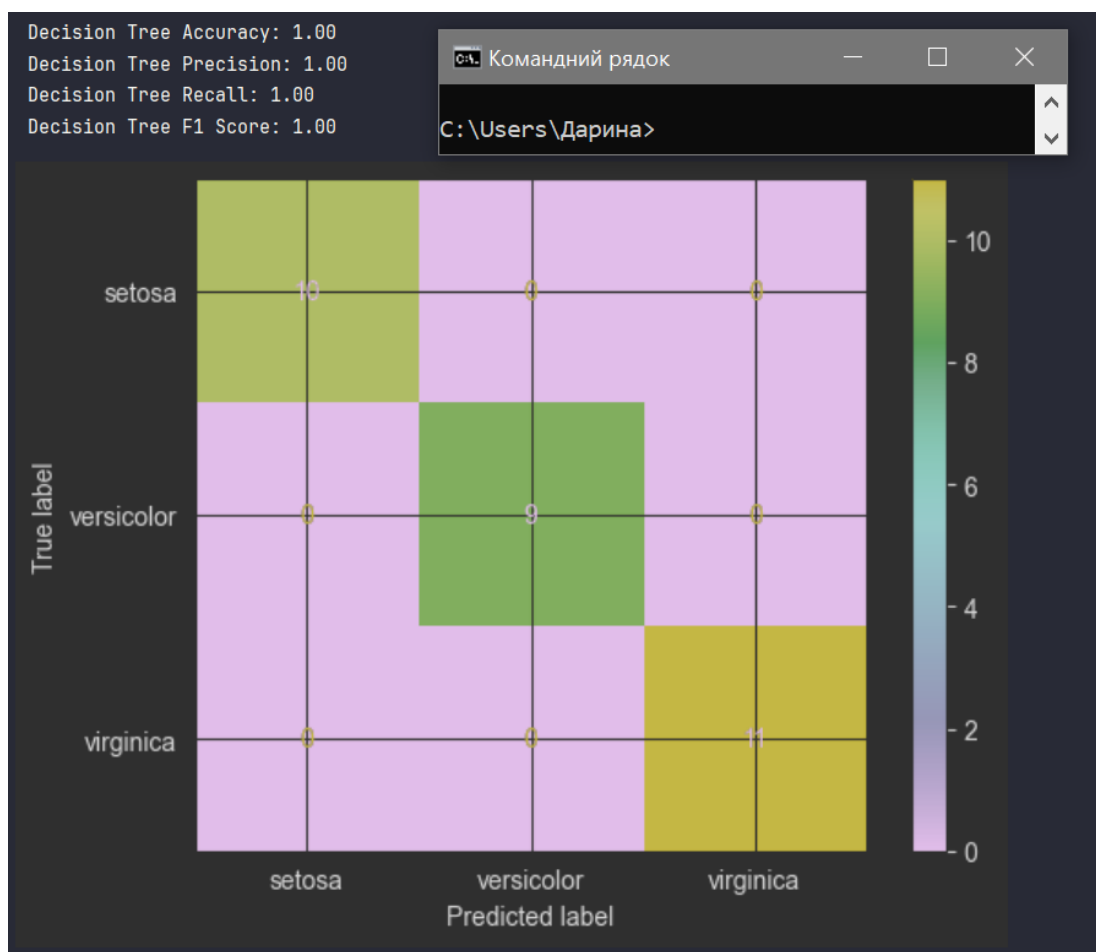
свідчить про коректність результатів тренування моделі та відсутність перенасичення. Висока точність пояснюється тим, що датасет Iris є досить маленьким у порівнянні з іншими, тому модель легше класифікує дані

```
tree_clf = DecisionTreeClassifier()
tree_clf.fit(X_train, y_train)
y_pred_tree = tree_clf.predict(X_test)
Executed at 2024.10.27 13:54:48 in 14ms

accuracy_tree = accuracy_score(y_test, y_pred_tree)
precision_tree = precision_score(y_test, y_pred_tree, average='macro')
recall_tree = recall_score(y_test, y_pred_tree, average='macro')
f1_tree = f1_score(y_test, y_pred_tree, average='macro')

print(f"Decision Tree Accuracy: {accuracy_tree:.2f}")
print(f"Decision Tree Precision: {precision_tree:.2f}")
print(f"Decision Tree Recall: {recall_tree:.2f}")
print(f"Decision Tree F1 Score: {f1_tree:.2f}")
ConfusionMatrixDisplay.from_estimator(tree_clf, X_test, y_test, display_labels=iris.target_names)
plt.show()
Executed at 2024.10.27 13:54:48 in 399ms
```

Рис. 5 Тренування моделі дерева рішень




```
Logistic Regression Cross-Val Accuracy: 0.9733333333333334
Decision Tree Cross-Val Accuracy: 0.9600000000000002
Training Accuracy (LogReg): 0.975
Test Accuracy (LogReg): 1.0
Training Accuracy (Tree): 1.0
Test Accuracy (Tree): 1.0
```

Рис. 6-7 Схожа ситуація спостерігається і з моделлю дерева рішень, де точність тренування є ідеальною. Це також можна пояснити малим розміром датасету, що підтверджується результатами крос-валідації

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

train_dataset = torchvision.datasets.FashionMNIST(root='./data', train=True, download=True, transform=transform)
test_dataset = torchvision.datasets.FashionMNIST(root='./data', train=False, download=True, transform=transform)

train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=64, shuffle=False)
```

Рис. 8 Назва класів у дата-сеті FashionMNIST та поділ його на тренувальний та тестувальний

```
X_train = train_dataset.data.numpy().reshape(-1, 28*28)
y_train = train_dataset.targets.numpy()

X_test = test_dataset.data.numpy().reshape(-1, 28*28)
y_test = test_dataset.targets.numpy()

X_train = X_train / 255.0
X_test = X_test / 255.0
```

Executed at 2024.10.27 13:55:09 in 1s 691ms

Рис. 9 Трансформація зображень в дата-сеті на вектора та зведення значень до типу від 0 до 1

```

log_reg_fashion = LogisticRegression(max_iter=10000, verbose=1)
log_reg_fashion.fit(X_train, y_train)

y_pred_logreg_fashion = log_reg_fashion.predict(X_test)

accuracy_logreg_fashion = accuracy_score(y_test, y_pred_logreg_fashion)
precision_logreg_fashion = precision_score(y_test, y_pred_logreg_fashion, average='macro')
recall_logreg_fashion = recall_score(y_test, y_pred_logreg_fashion, average='macro')
f1_logreg_fashion = f1_score(y_test, y_pred_logreg_fashion, average='macro')

print(f"Logistic Regression on Fashion-MNIST\nAccuracy: {accuracy_logreg_fashion:.2f}")
print(f"Precision: {precision_logreg_fashion:.2f}")
print(f"Recall: {recall_logreg_fashion:.2f}")
print(f"F1 Score: {f1_logreg_fashion:.2f}")

disp = ConfusionMatrixDisplay.from_estimator(log_reg_fashion, X_test, y_test, display_labels=class_names)
plt.xticks(rotation='vertical')
plt.show()

```

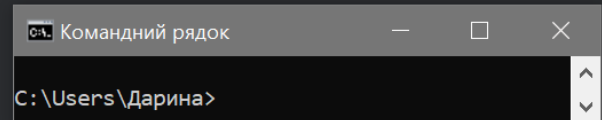


Рис. 10 Тренування моделі логістичної регресії для дата-сету FashionMNIST

```

Logistic Regression on Fashion-MNIST
Accuracy: 0.84
Precision: 0.84
Recall: 0.84
F1 Score: 0.84

```

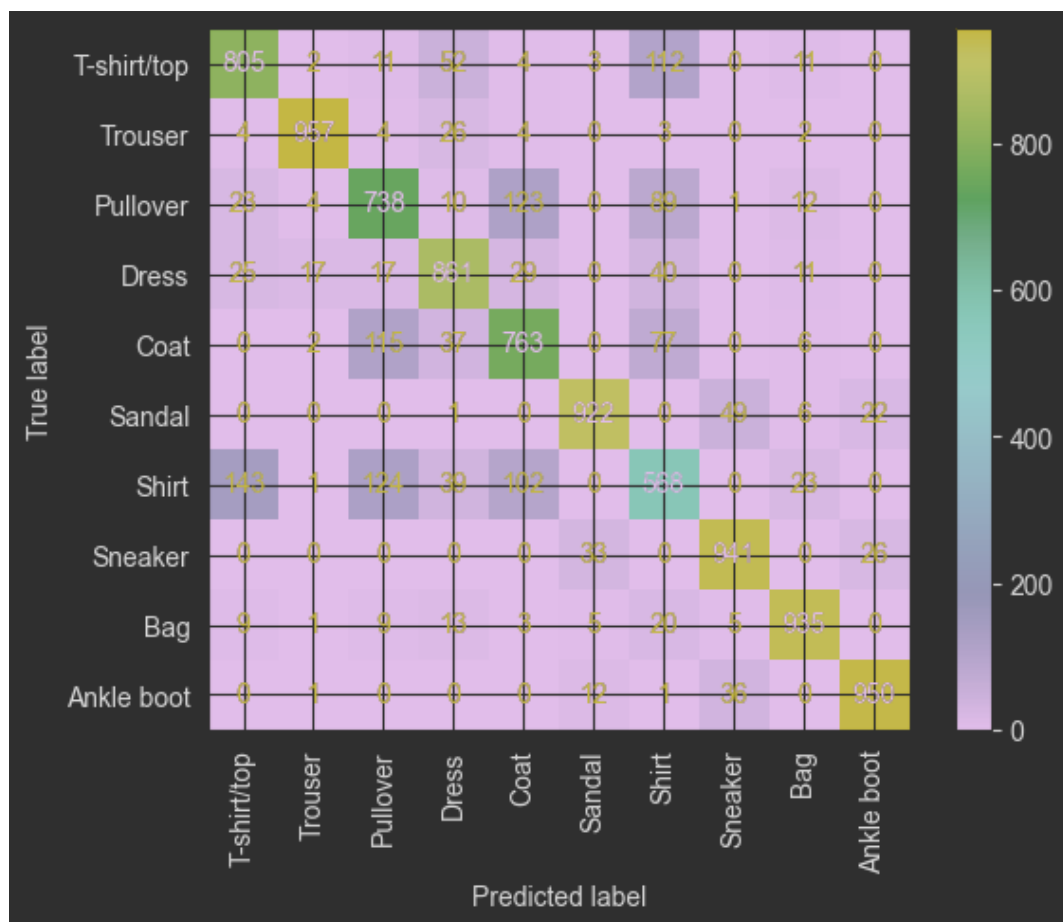


Рис. 11-12 Оцінка тренування моделі, можемо бачити, що результат вже далекі від ідеалу, цей результат можна назвати задовільнім

```

tree_clf_fashion = DecisionTreeClassifier()
tree_clf_fashion.fit(X_train, y_train)

y_pred_tree_fashion = tree_clf_fashion.predict(X_test)

accuracy_tree_fashion = accuracy_score(y_test, y_pred_tree_fashion)
precision_tree_fashion = precision_score(y_test, y_pred_tree_fashion, average='macro')
recall_tree_fashion = recall_score(y_test, y_pred_tree_fashion, average='macro')
f1_tree_fashion = f1_score(y_test, y_pred_tree_fashion, average='macro')

print(f"Decision Tree on Fashion-MNIST\nAccuracy: {accuracy_tree_fashion:.2f}")
print(f"Precision: {precision_tree_fashion:.2f}")
print(f"Recall: {recall_tree_fashion:.2f}")
print(f"F1 Score: {f1_tree_fashion:.2f}")

disp = ConfusionMatrixDisplay.from_estimator(tree_clf_fashion, X_test, y_test, display_labels=class_names)
plt.xticks(rotation='vertical')
plt.show()

```

Рис. 13 Тренування моделі дерева рішень для дата-сету FashionMNIST

```

Decision Tree on Fashion-MNIST
Accuracy: 0.79
Precision: 0.79
Recall: 0.79
F1 Score: 0.79

```

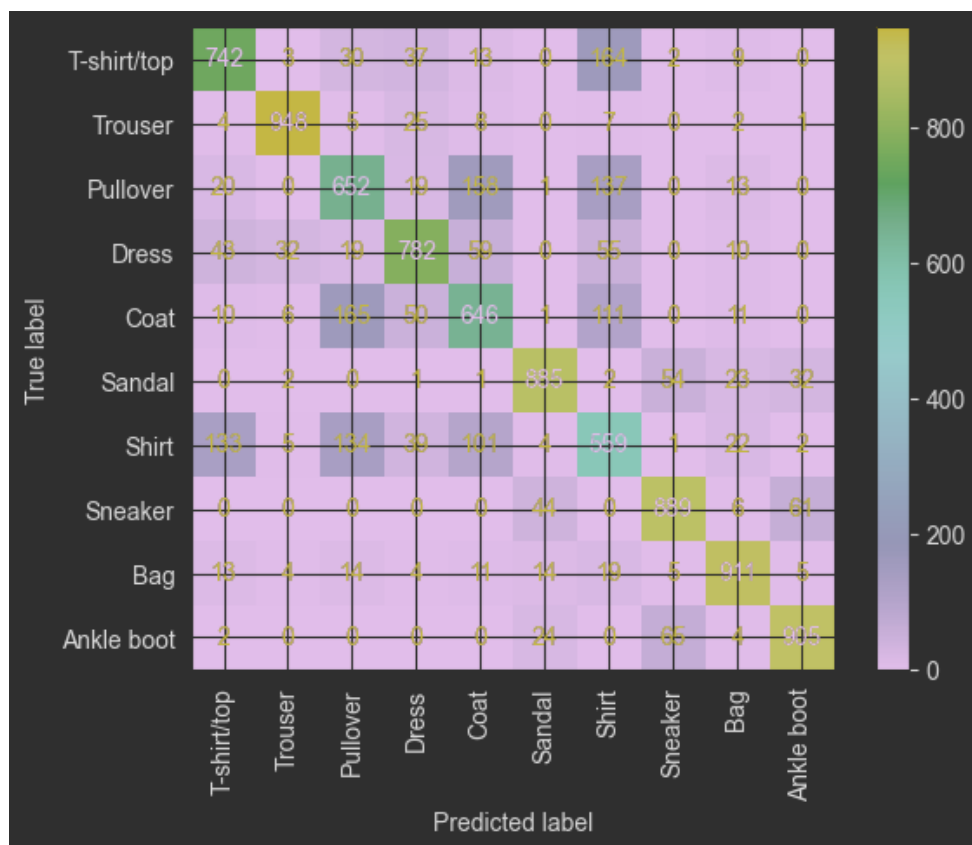


Рис. 14-15 Оцінка результатів класифікації дата-сету FashionMNIST на основі моделі дерева рішень, бачимо що є орієнтовна похибка в 0.21, що є задовільним результатом, але знову ж таки далеким від ідеалу.

Висновок: У цій лабораторній роботі я ознайомилася з методами класифікації, зокрема з логістичною регресією та деревами рішень. Я також мала змогу вивчити метрики оцінки моделей класифікації та матрицю невідповідностей, яка показує, які значення були класифіковані правильно, а які — ні, а також яким чином відбулася класифікація.