Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем і технологій

Лабораторна робота №3

Прикладні задачі машинного навчання

“Класифікація, регресія і кластеризація з використанням бібліотеки scikit-learn”

Виконав
студент групи ІК – 33:
Вересоцький А. Ю.

Перевірив:
асистент кафедри ІСТ
Нестерук А.О.

Київ 2025

Завдання:

1. Потрібно завантажити метеорологічні дані в 1895-2024 роках з CSV-файлу в DataFrame. Після цього дані треба буде відформатувати для використання.

Завантажую середні січневі температури в Лексінгтон штат Кентуккі з 1895 по 2024 рік через часові ряди NOAA «Climate at a Glance»:

```
lex = pd.read_csv('data.csv')
```
✓ [67] < 10 ms

```
lex.head()
```
✓ [68] < 10 ms

5 rows 5 rows × 2 cols

	# Lexington	Kentucky January Average Temperature
0	# Units: Degrees Fahrenheit	NaN
1	# Missing: -99	NaN
2	Date	Value
3	189501	28.8
4	189601	35.6

Форматування:

Видаляємо непотрібні рядки, оновлюємо індекси та змінюємо назви колонок:

```
lex.drop(index=[0, 1, 2], inplace=True)
```
✓ [69] < 10 ms

```
lex.reset_index(drop=True, inplace=True)
```
✓ [70] < 10 ms

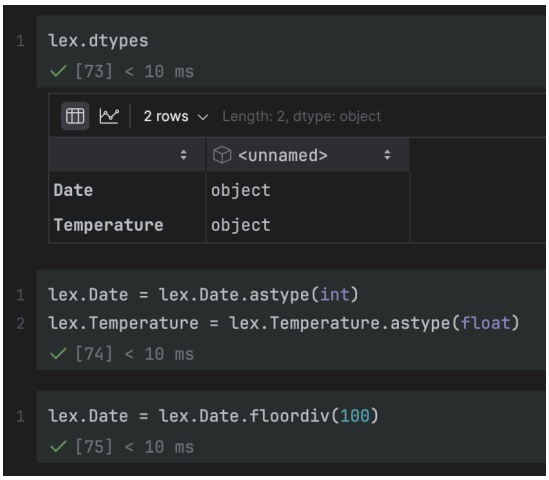
```
lex.columns = ['Date', 'Temperature']
```
✓ [71] < 10 ms

```
lex.head()
```
✓ [72] < 10 ms

5 rows 5 rows × 2 cols

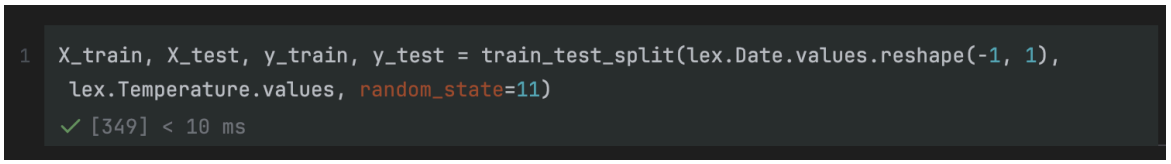
	Date	Temperature
0	189501	28.8
1	189601	35.6
2	189701	30.3
3	189801	39
4	189901	35.1

Перевіряємо типи даних та змінюємо їх за необхідності:

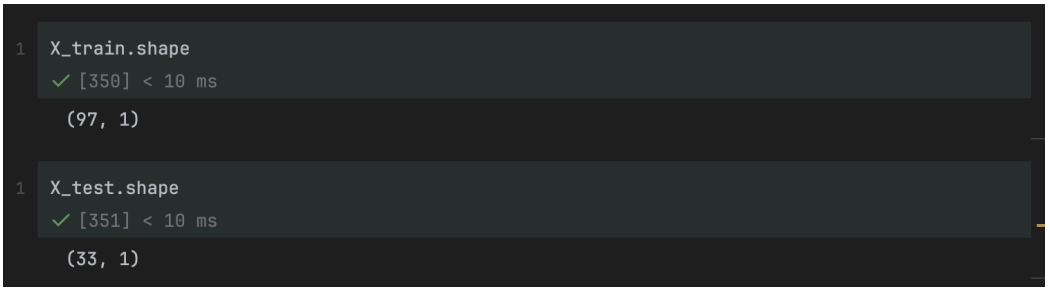


2. Розбиття даних для навчання і тестування.

Дані розбиваємо на навчальний і тестовий набори. Ключовий аргумент random_state використовується для забезпечення відтворюваності результатів:

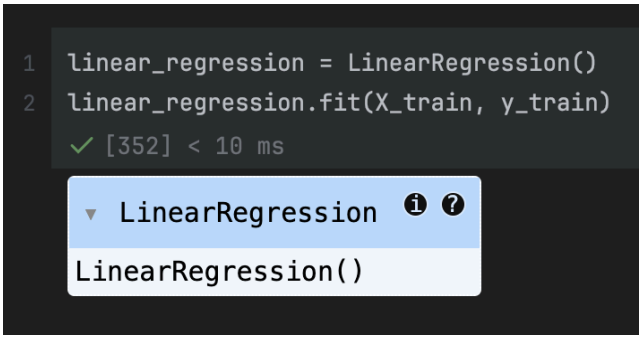


Для перевірки пропорції навчальних тестових даних (75% до 25%) задамо розміри X_train і X_test:

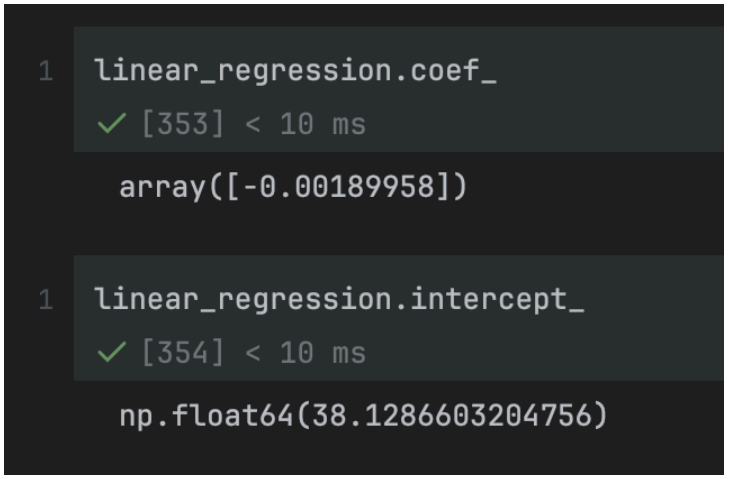


3. Навчання моделі.

Скористаємося оцінювачем LinearRegression:



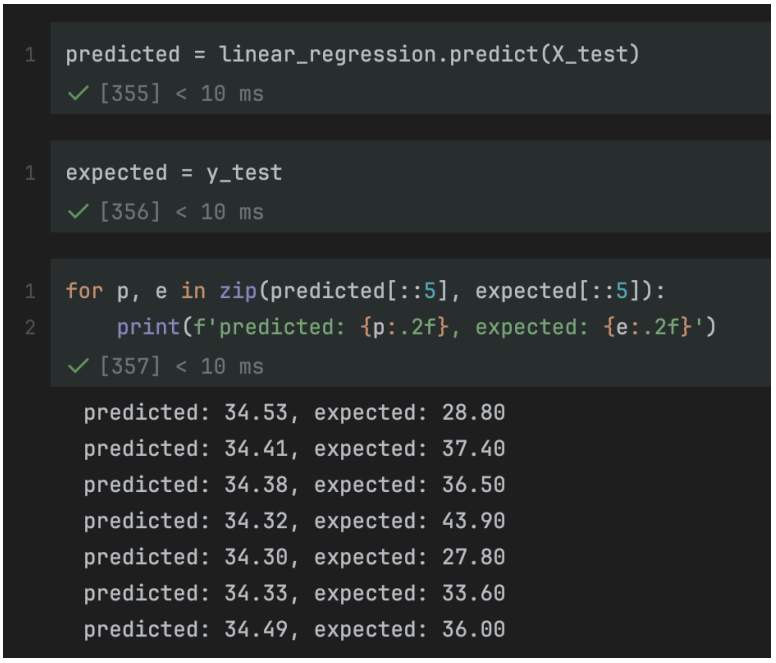
Кут нахилу зберігається в атрибуті coeff_ оцінювача, а точка перетину -в атрибуті intercept_:



Пізніше ці значення будуть використані для виведення регресійної прямої і прогнозування для конкретних дат.

4. Тестування моделі.

Протестуємо модель за даними з X_test і перевіримо прогнози по набору даних, виводячи прогнозовані і очікувані значення для кожного п'ятого елементу:



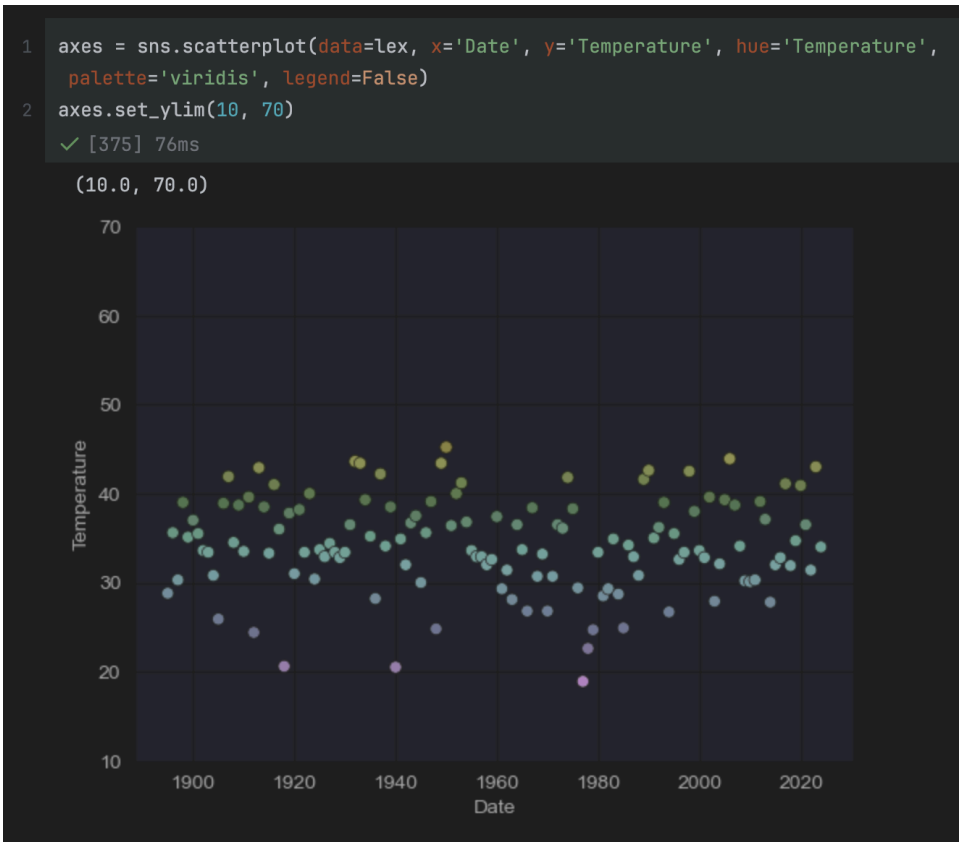
5. Прогнозування майбутніх температур і оцінка минулих температур.

Скористаємося отриманими значеннями кута нахилу і точки перетину для прогнозування середньої температури в січні 2019 року, а також оцінки середньої температури в січні 1890 року.

```
predict = (lambda x: linear_regression.coef_ * x + linear_regression.intercept_)
✓ [358] < 10 ms

predict(2019)
✓ [359] < 10 ms
array([34.29341246])

predict(1890)
✓ [360] < 10 ms
array([34.53845801])
```

6. Візуалізація набору даних з регресійній прямий.

Тепер побудуємо діаграму розкиду даних за допомогою функції scatterplot бібліотеки Seaborn і функції plot бібліотеки Matplotlib. Для виведення точок даних скористаємося методом scatterplot з колекцією DataFrame з ім'ям lex та змінимо масштаб осі y, щоб при виведенні регресійної прямої лінійність відносини була більш очевидною:



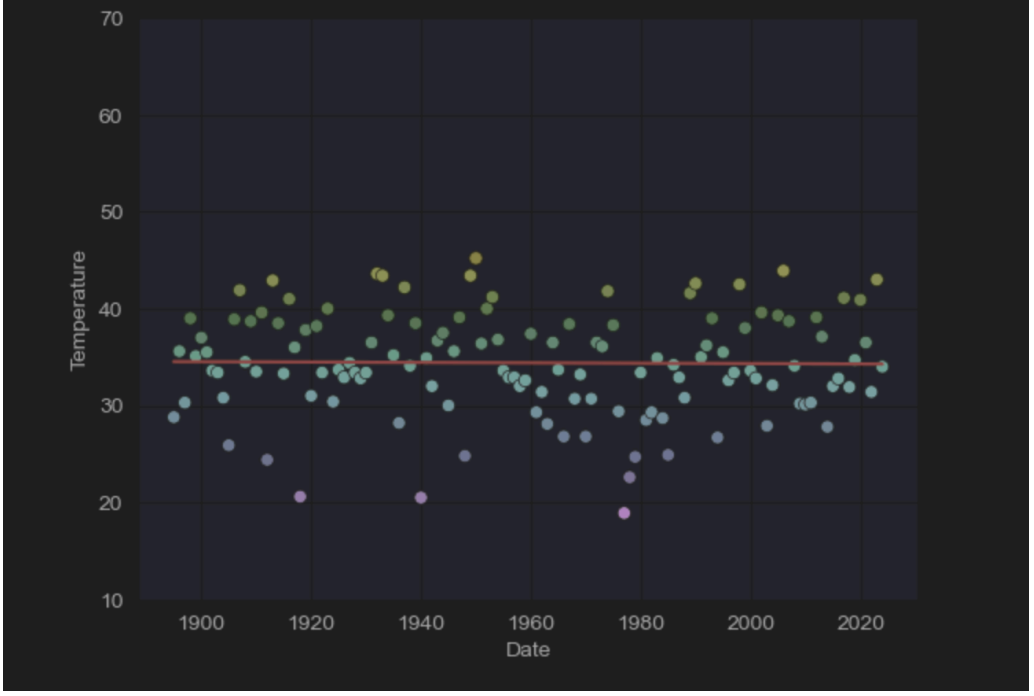
Перейдемо до висновку регресійної прямої. Почнемо зі створення масиву, що містить мінімальні і максимальні значення дати з lex.Date. Вони стануть координатами x початкової і кінцевої точок регресійної прямої.

В результаті передачі predict масиву x буде отримано масив відповідних прогнозованих значень, які будуть використовуватися в якості координат y.

Нарешті, функція plot бібліотеки Matplotlib малює лінію по масивам x та y, що зображує координати x та y точок відповідно:

Отримана діаграма розкиду даних з регресійній прямий зображена на наступній діаграмі.

```
axes = sns.scatterplot(data=lex, x='Date', y='Temperature', hue='Temperature',
 palette='viridis', legend=False)
axes.set_ylim(10, 70)

x = np.array([min(lex.Date.values), max(lex.Date.values)])
y = predict(x)
line = plt.plot(x, y, color='red')
plt.show()
✓ [376] 58ms
```



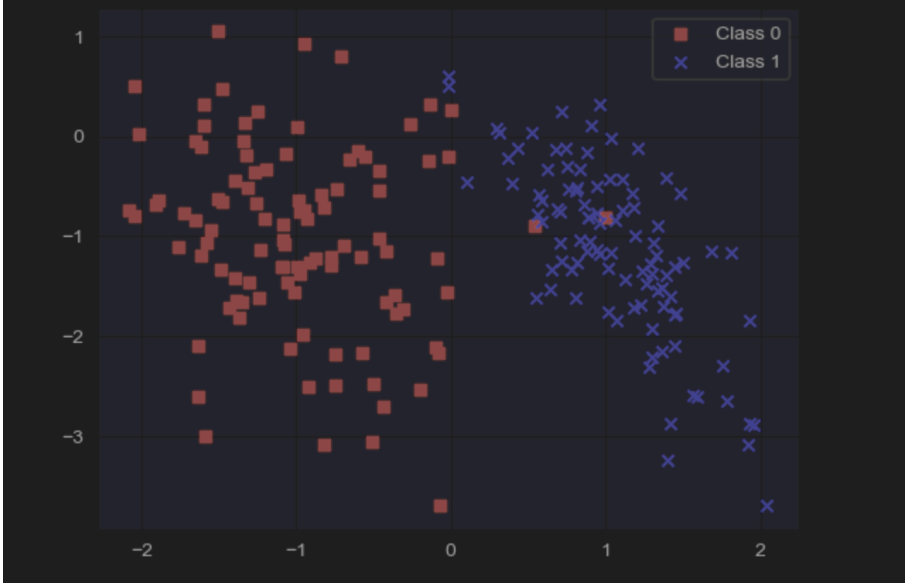
7. SVC класифікація.

енеруйте набір даних

та класифікуйте його використавши класифікатор SVC

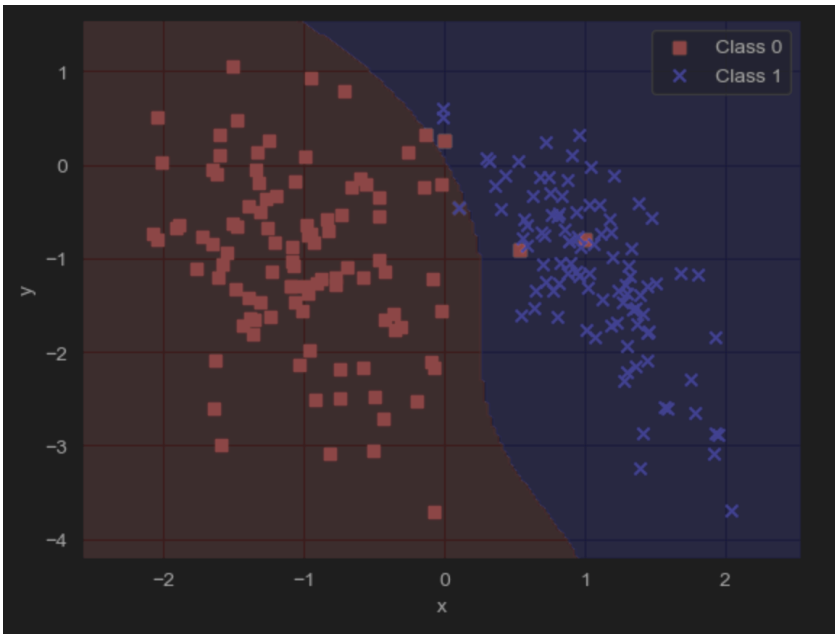
Генеруємо набір даних за допомогою вбудованої у бібліотеку scikit-learn функцію **make_classification** та візуалізуємо ці дані**:**

```python
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=200, n_features=2, n_classes=2,
 n_clusters_per_class=1, n_redundant=0, random_state=5)

plt.scatter(X[y == 0, 0], X[y == 0, 1], c='r', marker='s', label='Class 0')
plt.scatter(X[y == 1, 0], X[y == 1, 1], c='b', marker='x', label='Class 1')
plt.legend()
plt.show()

✓ [364] 41ms
```



Використовуємо Support Vector Machine (SVM) для класифікації наших двовимірних даних та візуалізуємо рішення моделі:

```python
svm = SVC(C=10).fit(X, y)

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5

xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 200))

Z = svm.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.2, cmap='bwr_r')
plt.scatter(X[y == 0, 0], X[y == 0, 1], c='r', marker='s', label='Class 0')
plt.scatter(X[y == 1, 0], X[y == 1, 1], c='b', marker='x', label='Class 1')

plt.xlabel('x')
plt.ylabel('y')
plt.xlim([x_min, x_max])
plt.ylim([y_min, y_max])
plt.legend()
plt.show()
```



8. Детальне порівняння класифікаційних оцінювачів KNeighborsClassifier, SVC та GaussianNB для вбудованого в scikit-learn набору даних breast cancer.

Спочатку завантажуємо набір даних breast cancer та розбиваємо дані на навчальний та тестовий набори:

```
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
 stratify=cancer.target, random_state=42)
```

Проводимо масштабування даних та навчання моделей:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

models = {
    "KNN (k=5)": KNeighborsClassifier(n_neighbors=5),
    "SVC (RBF Kernel)": SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42),
    "GaussianNB": GaussianNB()
}

results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, target_names=cancer.target_names,
     output_dict=True)

    results[name] = {
        "accuracy": acc,
        "precision": report["macro avg"]["precision"],
        "recall": report["macro avg"]["recall"],
        "f1-score": report["macro avg"]["f1-score"],
        "confusion_matrix": confusion_matrix(y_test, y_pred)
    }
```

Виводимо результати та візуалізуємо матрицю помилок:

```python
for name, metrics in results.items():
    print(f"\n==== {name} ====")
    print(f"Accuracy: {metrics['accuracy']:.4f}")
    print(f"Precision: {metrics['precision']:.4f}")
    print(f"Recall: {metrics['recall']:.4f}")
    print(f"F1-score: {metrics['f1-score']:.4f}")

fig, axes = plt.subplots(1, 3, figsize=(15, 4))

for ax, (name, metrics) in zip(axes, results.items()):
    sns.heatmap(metrics["confusion_matrix"], annot=True, fmt="d", cmap="Blues", ax=ax,
     xticklabels=cancer.target_names, yticklabels=cancer.target_names)
    ax.set_title(name)
    ax.set_xlabel("Predicted Label")
    ax.set_ylabel("True Label")

plt.tight_layout()
plt.show()
✓ [403] 214ms
```

```
==== KNN (k=5) ====
Accuracy: 0.9790
Precision: 0.9839
Recall: 0.9717
F1-score: 0.9772

==== SVC (RBF Kernel) ====
Accuracy: 0.9790
Precision: 0.9759
Recall: 0.9795
F1-score: 0.9776

==== GaussianNB ====
Accuracy: 0.9371
Precision: 0.9377
```

Висновки

KNN та SVC демонструють дуже схожі результати, майже ідеальні.

- KNN виграє за Precision, тому він робить менше False Positives (не буде марно турбувати пацієнтів).
- SVC виграє за Recall, тому він рідше пропускає реальні випадки хвороби (краще для медичних діагнозів).
- F1-score у них практично однаковий, тому можна вибрати будь-який.

GaussianNB значно поступається обом іншим алгоритмам, особливо за Recall (він пропускає більше хворих пацієнтів). Це може бути через те, що наївний байєсівський класифікатор припускає, що всі ознаки незалежні, що не завжди відповідає реальним даним.