ПРАКТИЧНА РОБОТА №7

Тема: Побудова простої цифрової моделі прогнозування ризиків з використанням AI-аналітики

Мета

1. Навчитися готувати дані моніторингу умов праці для моделювання.
2. Побудувати базову модель прогнозування інциденту/підвищеного ризику (AI/ML).
3. Оцінити якість моделі та інтерпретувати результати для управлінських рішень.

Очікувані результати

Після виконання роботи студент уміє:

- сформувати **фічі** з датчиків (часові ряди → агрегати, лаги, ковзні статистики);
- навчити базову модель (Logistic Regression / Random Forest / XGBoost або детектор аномалій);
- порахувати метрики (**AUC-ROC, F1, Precision-Recall, confusion matrix**);
- побудувати прості **правила реагування** за виходом моделі;
- пояснити внесок ознак (feature importance / SHAP на базовому рівні).

Дані (можна змоделювати або взяти з попередніх ПР)

Таблиця **events.csv** (приклад колонок):

- `timestamp` — час вимірювання (хв/сек).
- `zone` — зона/лінія/цех.
- Показники сенсорів: `temp`, `hum`, `noise_db`, `vibration_rms`, `co_ppm`, `pm25`.
- Технічний стан: `maintenance_overdue` (0/1), `shift_hours` (годин від початку зміни).
- Цільова змінна: `incident_next_60m` (0/1) — чи стався інцидент у найближчі 60 хв.

Якщо немає реальних інцидентів — створіть проксі-мітку: 1, коли **≥2 параметри** перевищують поріг (напр., `co_ppm>35` або `vibration_rms>limit`) протягом 10 хв поспіль.

Хід роботи (кроки)

1) Підготовка та огляд

- Перевірте пропуски, аномалії, дублікати.
- Розбийте на **train/validation/test** із урахуванням часу (TimeSeriesSplit або розріз «перші 70% часу — train», решта — val/test).

2) Інженерія ознак

Ковзні вікна: `mean_5m`, `max_5m`, `std_15m` для основних сенсорів.

Лаги: `co_ppm_lag5`, `vibration_rms_lag5`.

Комбіновані індекси (напр., **індекс теплового навантаження** з `temp` та `hum`).

Категоріальні: one-hot для `zone`.

Нормування/масштабування за потреби (для LR).

3) Базові моделі (оберіть 1–2)

Класифікація: Logistic Regression (baseline), Random Forest або XGBoost.

Альтернатива при відсутності міток: детекція аномалій (Isolation Forest / One-Class SVM) → позначити «ризикові» інтервали, порівняти з пороговою логікою.

4) Навчання та метрики

Баланс класів (за потреби): class_weight='balanced' або **SMOTE** (обережно з time series!).

Оцініть: **AUC-ROC**, **Average Precision (PR-AUC)**, **F1** при обраному порозі, **Confusion Matrix**.

Оберіть **поріг спрацювання** за кривою Precision-Recall (компроміс між пропуском і хибними тривогами).

5) Інтерпретація

Важливість ознак (feature importance / коефіцієнти LR).

Коротко поясніть 3–5 найвпливовіших ознак і їхній фізичний сенс.

Перевірте **drift**: чи стабільні розподіли фіч у різні зміни/зони?

6) Перетворення прогнозу на дію (Actionability)

- Сформулюйте **правила реагування**:
  - якщо `P(інцидент)>0.6` → *warning* (огляд зони протягом 10 хв),
  - якщо `>0.8` → *alarm* (вентиляція ON, зниження навантаження, інспекція обладнання).
- Додайте **ескалацію** (якщо тривога >5 хв — повідомити начальника зміни).

Мінімальний приклад коду (скелет, Python/Scikit-learn)

```
import pandas as pd
import numpy as np
from sklearn.model_selection import TimeSeriesSplit
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score,
average_precision_score, f1_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# 1) Load
df = pd.read_csv("events.csv", parse_dates=["timestamp"])
df = df.sort_values("timestamp").reset_index(drop=True)

# 2) Simple feature eng (приклад)
for col in
["co_ppm","vibration_rms","noise_db","temp","hum"]:
    df[f"{col}_mean_5"] = df[col].rolling(5,
min_periods=1).mean()
    df[f"{col}_std_15"] = df[col].rolling(15,
min_periods=1).std().fillna(0)
    df[f"{col}_lag5"] = df[col].shift(5)
```

```
df = df.dropna().reset_index(drop=True)

# 3) Split по часу
target = "incident_next_60m"
features = [c for c in df.columns if c not in
["timestamp","zone",target]]
split = int(len(df)*0.7)
train, test = df.iloc[:split], df.iloc[split:]

# 4) Scale для LR (опційно)
scaler = StandardScaler()
X_train = scaler.fit_transform(train[features])
X_test  = scaler.transform(test[features])
y_train, y_test = train[target], test[target]

# 5) Baseline: Logistic Regression (balanced)
lr = LogisticRegression(max_iter=200,
class_weight="balanced")
lr.fit(X_train, y_train)
proba_lr = lr.predict_proba(X_test)[:,1]

# 6) Альтернатива: Random Forest
rf = RandomForestClassifier(n_estimators=300,
random_state=42, class_weight="balanced")
rf.fit(train[features], y_train)
proba_rf = rf.predict_proba(test[features])[:,1]

# 7) Метрики
def report(p):
    thr = 0.5
    preds = (p>=thr).astype(int)
    print("ROC-AUC:", roc_auc_score(y_test, p))
    print("PR-AUC :", average_precision_score(y_test, p))
    print("F1(0.5):", f1_score(y_test, preds))
    print("CM:\n", confusion_matrix(y_test, preds))

print("=== Logistic Regression ==="); report(proba_lr)
print("=== Random Forest ===");      report(proba_rf)
```

Що здати

1. **Ноутбук/скрипт** із кодом та коментарями.
2. **Короткий звіт (2–3 стор.):**
   - опис даних та фіч,
   - вибір моделі, метрики (ROC-AUC, PR-AUC, F1, матриця помилок),
   - інтерпретація важливих ознак,

- обраний поріг та **правила реагування**,
- обмеження моделі та подальші кроки (ініціативи з підвищення якості даних).

3. (Опційно) **1 слайд** з головними висновками для керівництва (KPI, вигода, як інтегрувати у EHS/SCADA).