ДОДАТОК А

Програмна реалізація алгоритмів

Реалізація функціоналу ініціалізації таблиці:

```
     def readKoVal(tp):
          ColumnList                =                ["tagName",
"tagDoc","tagVal","tagSavedVal"]
          ColumnList            =            [u"Ідентифікатор",
u"Опис",u"Значення",u"Повний шлях", "decimal"]
          fVal = float(0.0)
          Data  =  system.dataset.toDataSet(ColumnList,[["","",
fVal, "", fVal]])            #create new dataset
          Data = system.dataset.clearDataset(Data)
          tag_list  =  system.tag.browse(path  =  tp  ,filter
={}).getResults()
          tag_conf = system.tag.getConfiguration(tp,True)

          for ntag in tag_list:
               tag_name = ntag['name']
               tag_path = ntag['fullPath']
               i=tag_conf[0]['tags'].index(next(x   for   x   in
tag_conf[0]['tags'] if x['name']==ntag['name']))
               tag_doc   =
tag_conf[0]['tags'][i]['documentation']
               tag_val =  ntag['value'].getValue()
               tag_format                                    =
tag_conf[0]['tags'][i]['formatString']
               decimal = str(getDelta(tag_format))
               row = [ tag_name,  tag_doc, tag_val, tag_path,
decimal]
               #row = [ tag_name,  tag_doc, tag_val, tag_path]
               Data = system.dataset.addRow(Data,0,row)

          return Data
```

Реалізація функціоналу імпорту:

```
    def excelToDataSet(fileName, hasHeaders = False, sheetNum
= 0, firstRow = None, lastRow = None, firstCol = None, lastCol
= None):
                import
org.apache.poi.ss.usermodel.WorkbookFactory as WorkbookFactory
                import  org.apache.poi.ss.usermodel.DateUtil  as
DateUtil
                from java.io import FileInputStream
                from java.util import Date
                """
                   Function to create a dataset from an Excel
spreadsheet. It will try to automatically detect the boundaries
of the data,
                   but helper parameters are available:
                   params:
                          fileName    -The path to the Excel
spreadsheet. (required)
                          hasHeaders-If true, uses the first row
of the spreadsheet as column names.
                          sheetNum  -select the sheet to process.
defaults to the first sheet.
                          firstRow  -select first row to process.
                          lastRow   -select last row to process.
                          firstCol    -select first column to
process
                          lastCol   -select last column toprocess
                """

                fileStream = FileInputStream(fileName)

                wb = WorkbookFactory.create(fileStream)

                sheet = wb.getSheetAt(sheetNum)
```

```
                if firstRow is None:
                     firstRow = sheet.getFirstRowNum()
                if lastRow is None:
                     lastRow = sheet.getLastRowNum()
                data = []
                for i in range(firstRow , lastRow + 1):
                     row = sheet.getRow(i)
                     #print str(i).zfill(3), list(row)
                     if i == firstRow:
                          if firstCol is None:
                               firstCol = row.getFirstCellNum()
                          if lastCol is None:
                               lastCol  = row.getLastCellNum()
                          else:
                               # if lastCol is specified add 1 to
it.
                               lastCol += 1
                          if hasHeaders:
                               headers                          =
list(row)[firstCol:lastCol]
                          else:
                               headers = ['Col'+str(i)  for  i  in
range(firstCol, lastCol)]
                     rowOut = []
                     for j in range(firstCol, lastCol):
                          if i == firstRow and hasHeaders:
                               pass
                          else:
                               cell = row.getCell(j)
                               if cell != None:
                                    cellType                    =
cell.getCellType().toString()
                                    if cellType == 'NUMERIC':
                                         if
DateUtil.isCellDateFormatted(cell):
```

```
                                        value            =
cell.dateCellValue
                                    else:
                                        value            =
cell.getNumericCellValue()
                                        if     value     ==
float(value):
                                            value        =
float(value)
                                elif cellType == 'STRING':
                                        value                =
cell.getStringCellValue()
                                elif cellType == 'BOOLEAN':
                                        value                =
cell.getBooleanCellValue()
                                elif cellType == 'BLANK':
                                        value = None
                                elif cellType == 'FORMULA':

     formulatype=str(cell.getCachedFormulaResultType())
                                    if    formulatype    ==
'NUMERIC':
                                        if
DateUtil.isCellDateFormatted(cell):
                                            value        =
cell.dateCellValue
                                        else:
                                            value        =
cell.getNumericCellValue()
                                            if   value   ==
float(value):
                                                value    =
float(value)
                                    elif   formulatype   ==
'STRING':
```

```
                                        value            =
cell.getStringCellValue()
                                    elif   formulatype   ==
'BOOLEAN':
                                        value            =
cell.getBooleanCellValue()
                                    elif   formulatype   ==
'BLANK':
                                        value = None
                                else:
                                    value = None
                                rowOut.append(value)
                        if len(rowOut) > 0:
                            data.append(rowOut)

                fileStream.close()

                Data = system.dataset.toDataSet(headers, data)

                return Data
```

Реалізація алгоритмів навчання:

```
    import pandas as pd
    import numpy as np
    from sklearn.preprocessing import StandardScaler
    # Завантаження датасету
    data = pd.read_csv("train_FD001.txt", sep=" ", header=None)
    data.dropna(axis=1, how='all', inplace=True)
    data.columns = ['engine_id', 'cycle'] + [f'op_setting_{i}'
for i in range(3)] + [f'sensor_{i}' for i in range(1, 22)]
    # Додавання RUL
    rul                                                    =
data.groupby('engine_id')['cycle'].max().reset_index()
    rul.columns = ['engine_id', 'max_cycle']
```

```
    data = data.merge(rul, on='engine_id')
    data['RUL'] = data['max_cycle']-data['cycle']
    # Random Forest Regressor
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import mean_absolute_error,
mean_squared_error
    # Вибір ознак
    features = [f'sensor_{i}' for i in range(1, 22)]
    X = data[features]
    y = data['RUL']
    # Нормалізація
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    # Розділення
    X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=0.2, random_state=42)
    # Навчання
    rf = RandomForestRegressor(n_estimators=100, max_depth=10,
random_state=42)
    rf.fit(X_train, y_train)
    # Оцінка
    y_pred = rf.predict(X_test)
    print("MAE:", mean_absolute_error(y_test, y_pred))
    print("RMSE:", np.sqrt(mean_squared_error(y_test,
y_pred)))
    # LSTM-модель
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import LSTM, Dense, Dropout
    from tensorflow.keras.preprocessing.sequence import
TimeseriesGenerator
    # Підготовка часових вікон
    window_size = 30
    sensor_data = data[features].values
    target = data['RUL'].values
    # Нормалізація
```

```
    sensor_data = scaler.fit_transform(sensor_data)
    # Створення генератора послідовностей
    generator = TimeseriesGenerator(sensor_data, target,
length=window_size, batch_size=64)
    # Побудова моделі
    model = Sequential([
        LSTM(64, return_sequences=True,
input_shape=(window_size, len(features))),
        Dropout(0.2),
        LSTM(32),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')
    model.fit(generator, epochs=10)
    # Прогноз (демонстраційний приклад)
    predictions = model.predict(generator)
    # Візуалізація результатів
    import matplotlib.pyplot as plt
    plt.plot(y_test[:100].values, label='True RUL')
    plt.plot(y_pred[:100], label='RF Predicted RUL')
    plt.title("Порівняння фактичного та передбаченого RUL
(Random Forest)")
    plt.xlabel("Спостереження")
    plt.ylabel("RUL")
    plt.legend()
    plt.grid()
    plt.show()
    # Крос-валідація з TimeSeriesSplit
    from sklearn.model_selection import TimeSeriesSplit
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.metrics import mean_absolute_error,
mean_squared_error
    import numpy as np
    # Ознаки й ціль
    features = [f'sensor_{i}' for i in range(1, 22)]
    X = data[features].values
```

```
    y = data['RUL'].values
    # Нормалізація
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    # Ініціалізація TimeSeriesSplit
    tscv = TimeSeriesSplit(n_splits=5)
    mae_list, rmse_list = [], []
    for fold, (train_index, test_index) in
enumerate(tscv.split(X_scaled)):
        X_train, X_test = X_scaled[train_index],
X_scaled[test_index]
        y_train, y_test = y[train_index], y[test_index]

        model = RandomForestRegressor(n_estimators=100,
max_depth=10, random_state=42)
        model.fit(X_train, y_train)

        y_pred = model.predict(X_test)

        mae = mean_absolute_error(y_test, y_pred)
        rmse = np.sqrt(mean_squared_error(y_test, y_pred))

        mae_list.append(mae)
        rmse_list.append(rmse)

        print(f"Fold {fold + 1}: MAE = {mae:.2f}, RMSE =
{rmse:.2f}")
    print("\n=== Середні результати ===")
    print(f"MAE: {np.mean(mae_list):.2f} ±
{np.std(mae_list):.2f}")
    print(f"RMSE: {np.mean(rmse_list):.2f} ±
{np.std(rmse_list):.2f}")

    # Grid Search з TimeSeriesSplit для Random Forest
    from sklearn.model_selection import GridSearchCV,
TimeSeriesSplit
```

```
from sklearn.ensemble import RandomForestRegressor
# Параметри для пошуку
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5],
}
# TimeSeriesSplit
tscv = TimeSeriesSplit(n_splits=3)
# Модель
rf = RandomForestRegressor(random_state=42)

# Grid Search з крос-валідацією
grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=tscv,
    scoring='neg_mean_absolute_error',
    n_jobs=-1,
    verbose=2
)
grid_search.fit(X_scaled, y)
print("Найкращі параметри:", grid_search.best_params_)
print("MAE:", -grid_search.best_score_)

# TimeSeriesSplit для LSTM вручну
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.preprocessing.sequence import
TimeseriesGenerator
from sklearn.metrics import mean_absolute_error
import numpy as np
n_splits = 3
window_size = 30
mae_scores = []
# Генерація розбиттів
```

```
    tscv = TimeSeriesSplit(n_splits=n_splits)
    for fold, (train_idx, test_idx) in
enumerate(tscv.split(sensor_data)):
        X_train, y_train = sensor_data[train_idx],
target[train_idx]
        X_test, y_test = sensor_data[test_idx],
target[test_idx]
        train_gen = TimeseriesGenerator(X_train, y_train,
length=window_size, batch_size=64)
        test_gen = TimeseriesGenerator(X_test, y_test,
length=window_size, batch_size=64)

        model = Sequential([
            LSTM(64, return_sequences=True,
input_shape=(window_size, X_train.shape[1])),
            Dropout(0.2),
            LSTM(32),
            Dense(1)
        ])
        model.compile(optimizer='adam', loss='mae')
        model.fit(train_gen, epochs=5, verbose=0)
        y_pred = model.predict(test_gen)
        y_true = y_test[window_size:]
        mae = mean_absolute_error(y_true, y_pred)
        mae_scores.append(mae)
        print(f"Fold {fold + 1}: MAE = {mae:.2f}")
    print("\nСередній MAE:", np.mean(mae_scores))

    # Візуалізація помилок по фолдах
    import matplotlib.pyplot as plt
    plt.figure(figsize=(6, 4))
    plt.plot(range(1, n_splits + 1), mae_scores, marker='o',
label='LSTM MAE')
    plt.title("MAE по фолдах (LSTM, TimeSeriesSplit)")
    plt.xlabel("Фолд")
    plt.ylabel("MAE")
```

```
plt.grid(True)
plt.legend()
plt.show()

# ml_maintenance_report.py
import numpy as np
import matplotlib.pyplot as plt
from fpdf import FPDF
from PIL import Image
# Дані: MAE по фолдах
mae_list_rf = [20.1, 18.7, 19.5, 21.3, 20.0]
mae_list_lstm = [15.2, 14.8, 13.9, 16.0, 15.1]
# Обчислення середніх значень
mean_rf = np.mean(mae_list_rf)
mean_lstm = np.mean(mae_list_lstm)
# Графік 1: Random Forest
plt.figure(figsize=(6, 4))
plt.plot(range(1,    len(mae_list_rf)+1),    mae_list_rf,
marker='s', linestyle='-', color='green')
plt.axhline(mean_rf,    color='gray',    linestyle='--',
label=f'Середнє: {mean_rf:.2f}')
plt.title("MAE по фолдах – Random Forest")
plt.xlabel("Фолд")
plt.ylabel("MAE")
plt.xticks(range(1, len(mae_list_rf)+1))
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig("rf_mae_folds.png", dpi=300)
plt.close()
# Графік 2: LSTM
plt.figure(figsize=(6, 4))
plt.plot(range(1,  len(mae_list_lstm)+1),  mae_list_lstm,
marker='o', linestyle='-', color='blue')
plt.axhline(mean_lstm,   color='gray',   linestyle='--',
label=f'Середнє: {mean_lstm:.2f}')
```

```
    plt.title("MAE по фолдах – LSTM")
    plt.xlabel("Фолд")
    plt.ylabel("MAE")
    plt.xticks(range(1, len(mae_list_lstm)+1))
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.savefig("lstm_mae_folds.png", dpi=300)
    plt.close()
    # Графік 3: Порівняння моделей
    plt.figure(figsize=(7, 4))
    plt.plot(mae_list_rf, marker='s', label='Random Forest',
color='green')
    plt.plot(mae_list_lstm, marker='o', label='LSTM',
color='blue')
    plt.title("Порівняння MAE по фолдах: RF vs LSTM")
    plt.xlabel("Фолд")
    plt.ylabel("MAE")
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.savefig("comparison_mae_folds.png", dpi=300)
    plt.close()
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	95 с.
	Інші документи	
2.	Презентаційні матеріали	15 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Гордієнко Є.Д.			Дослідження методів машинного навчання для технічного обслуговування в Індустрії 4.0
Перевірив		Терзіян В.Я.			
Н.контр.		Малєєва І.А.			
Затв.		Золотухін О.В.			

Шифр групи	Код напр./спец.
СШІм-23-1	122
ХНУРЕ кафедра ШІ	