ДОДАТОК А

Вихідний код програмної реалізації

```
import shap

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from lightgbm import LGBMRegressor

from sklearn.metrics import mean_squared_error,
mean_absolute_percentage_error, r2_score

from sklearn.preprocessing import OrdinalEncoder

from sklearn.base import clone

from tqdm import tqdm

import warnings

warnings.filterwarnings("ignore")



sns.set(style="whitegrid")

plt.rcParams["figure.figsize"] = (12, 6)



ЗАВАНТАЖЕННЯ ДАНИХ ТА ПОПЕРЕДНЯ ОБРОБКА

path =
'/content/drive/MyDrive/Diploma2026/FMCG_2022_2024.csv'
```

```
    df = pd.read_csv(path)

    df_eda = df.copy()

    # Видалення рядків з від'ємними (аномальними) значеннями

    df = df[(df[["units_sold", "stock_available",
"delivered_qty", "delivery_days"]] >= 0).all(axis=1)].copy()

    df = df.sort_values(["sku",
"date"]).reset_index(drop=True)

    ГЕНЕРАЦІЯ ОЗНАК (FEATURE ENGINEERING)

    # Часові лаги для цільової змінної

    for lag in [1-6]:

        df[f"lag_{lag}"] =
df.groupby("sku")["units_sold"].shift(lag)



    # Імпульс (відношення лагів)

    df["momentum_7_1"] = df["lag_1"] / (df["lag_7"] + 1e-3)

    df["momentum_14_7"] = df["lag_7"] / (df["lag_14"] + 1e-3)



    # Ковзні статистики (Rolling statistics)

    df["rolling_mean_7"] =
df.groupby("sku")["units_sold"].transform(lambda x:
x.shift(1).rolling(7).mean())

    df["rolling_std_7"]  =
df.groupby("sku")["units_sold"].transform(lambda x:
x.shift(1).rolling(7).std())
```

```
    df["rolling_max_7"]  =
df.groupby("sku")["units_sold"].transform(lambda x:
x.shift(1).rolling(7).max())

    df["rolling_min_7"]  =
df.groupby("sku")["units_sold"].transform(lambda x:
x.shift(1).rolling(7).min())



    # Локальний тренд за останні 7 днів

    def rolling_trend(x):

        return x.shift(1).rolling(7).apply(lambda z:
np.polyfit(range(len(z)), z, 1) if len(z.dropna()) == 7 else
np.nan)



    df["trend_7"] =
df.groupby("sku")["units_sold"].transform(rolling_trend)



    # Лаги для ціни, акцій, доставки та залишків

    df["price_lag_1"] =
df.groupby("sku")["price_unit"].shift(1)

    df["promo_lag_1"] =
df.groupby("sku")["promotion_flag"].shift(1)

    df["promo_effect"] = df["promotion_flag"] * df["lag_1"]

    df["delivery_lag_7"] =
df.groupby("sku")["delivery_days"].shift(7)

    df["stock_lag_1"] =
df.groupby("sku")["stock_available"].shift(1)
```

```
    # Глобальні статистики на рівні SKU

    sku_stats = df.groupby("sku")["units_sold"].agg(["mean",
"min", "max", "std"]).reset_index()

    sku_stats.columns = ["sku", "sku_mean", "sku_min",
"sku_max", "sku_std"]

    df = df.merge(sku_stats, on="sku", how="left")



    # Календарні ознаки

    df["date"] = pd.to_datetime(df["date"])

    df["is_month_start"] =
df["date"].dt.is_month_start.astype(int)

    df["is_month_end"]   =
df["date"].dt.is_month_end.astype(int)

    df["dayofweek"]      = df["date"].dt.dayofweek

    df["month"]          = df["date"].dt.month

    df["year"]           = df["date"].dt.year

    df["is_weekend"]     = df["dayofweek"].isin([7,
8]).astype(int)

    df["day"] = df["date"].dt.day

    df["week_of_month"] = df["day"] // 7 + 1



    # Ковзні статистики для категорій та цін
```

```
    df["category_rolling_mean_7"] =
df.groupby("category")["units_sold"].transform(lambda x:
x.shift(1).rolling(7).mean())

    df["category_rolling_std_7"]  =
df.groupby("category")["units_sold"].transform(lambda x:
x.shift(1).rolling(7).std())

    df["price_rolling_mean_7"] =
df.groupby("sku")["price_unit"].transform(lambda x:
x.shift(1).rolling(7).mean())

    df["promo_rolling_7"]      =
df.groupby("sku")["promotion_flag"].transform(lambda x:
x.shift(1).rolling(7).sum())



    # Індикатори відсутності товару (Out-of-Stock)

    df["stockout_flag"] = ((df["stock_available"] == 0) &
(df["delivered_qty"] > 0)).astype(int)

    df["rolling_stockouts_7"] =
df.groupby("sku")["stockout_flag"].transform(lambda x:
x.shift(1).rolling(7).sum())



    # Час з моменту останньої акції чи доставки

    def time_since_event(series):

        return series[::-1].cumsum()[::-1] * series



    df["time_since_promo"] =
df.groupby("sku")["promotion_flag"].transform(lambda x:
time_since_event(x == 0))
```

```
    df["time_since_delivery"] =
df.groupby("sku")["delivery_days"].transform(lambda x:
time_since_event(x == 0))


    # Тригонометричні сезонні сигнали

    df["month_sin"] = np.sin(2 * np.pi * df["month"] / 12)

    df["month_cos"] = np.cos(2 * np.pi * df["month"] / 12)

    df["dow_sin"]   = np.sin(2 * np.pi * df["dayofweek"] / 7)

    df["dow_cos"]   = np.cos(2 * np.pi * df["dayofweek"] / 7)


    # Ознаки взаємодії

    df["promo_dow"] = df["promotion_flag"] * df["dayofweek"]

    df["price_x_stock"] = df["price_unit"] *
df["stock_available"]


    # Кодування категоріальних змінних та видалення NaN

    enc = OrdinalEncoder(handle_unknown='use_encoded_value',
unknown_value=-1)

    df[["sku_cat", "segment_cat", "category_cat"]] =
enc.fit_transform(df[["sku", "segment", "category"]])

    df = df.dropna()


    # Формування фінального простору ознак
```

```
    exclude_cols = ["date", "sku", "brand", "segment",
"channel", "region", "pack_type", "delivered_qty",
"units_sold", "category"]

    features = [col for col in df.columns if col not in
exclude_cols and not np.issubdtype(df[col].dtype,
np.datetime64)]

    categorical_features = ["sku_cat", "segment_cat",
"category_cat", "dayofweek", "month", "is_weekend",
"promotion_flag", "is_month_start", "is_month_end"]

    target = "units_sold"



    # Розбиття на тренувальну та тестову вибірки (останні 42
дні)

    cutoff_date = df["date"].max() - pd.Timedelta(days=42)

    train = df[df["date"] <= cutoff_date].copy()

    test  = df[df["date"] > cutoff_date].copy()



    НАВЧАННЯ МОДЕЛЕЙ LIGHTGBM ДЛЯ КОЖНОЇ КАТЕГОРІЇ

    category_models = {}

    category_preds = []

    category_metrics = {}



    def safe_mape(y_true, y_pred):

        mask = y_true != 0
```

```
        return mean_absolute_percentage_error(y_true[mask],
y_pred[mask])


    for cat in tqdm(df["category"].unique(), desc="Training
per category"):
        df_cat = df[df["category"] == cat].copy()
        cutoff = df_cat["date"].max() - pd.Timedelta(days=42)
        train_c = df_cat[df_cat["date"] <= cutoff]
        test_c  = df_cat[df_cat["date"] > cutoff]

        if len(train_c) < 30 or len(test_c) < 7:
            continue

        # Базова модель для визначення важливості ознак
(Feature Ranking)
        base_model = LGBMRegressor(n_estimators=200,
learning_rate=0.05, max_depth=5, random_state=42, verbose=-1)
        base_model.fit(train_c[features], train_c[target])
        fi = pd.Series(base_model.feature_importances_,
index=features).sort_values(ascending=False)

        best_score = float("inf")
        best_features = features
```

```
        # Жадібний відбір ознак (Greedy Forward Selection)
        for k in range(10, len(features) + 1, 5):
            top_k = fi.head(k).index.tolist()
            model_k = clone(base_model).fit(train_c[top_k],
train_c[target])
            preds_k = np.clip(model_k.predict(test_c[top_k]),
0, None)
            rmse_k =
np.sqrt(mean_squared_error(test_c[target], preds_k))
            if rmse_k < best_score:
                best_score = rmse_k
                best_features = top_k

        # Навчання фінальної моделі на відібраних ознаках
        model = LGBMRegressor(
            n_estimators=500, learning_rate=0.03, max_depth=7,
            subsample=0.8, colsample_bytree=0.8,
random_state=42, verbose=-1
        )
        model.fit(
            train_c[best_features], train_c[target],
```

```
            categorical_feature=[f for f in
categorical_features if f in best_features]

        )



        test_c = test_c.copy()

        test_c["prediction"] =
np.clip(model.predict(test_c[best_features]), 0, None)

        category_preds.append(test_c)

        category_models[cat] = {"model": model, "features":
best_features}



        # Розрахунок метрик якості

        rmse = np.sqrt(mean_squared_error(test_c[target],
test_c["prediction"]))

        mape = safe_mape(test_c[target], test_c["prediction"])

        smape = np.mean(2 * np.abs(test_c[target] -
test_c["prediction"]) /

                        (np.abs(test_c[target]) +
np.abs(test_c["prediction"]) + 1e-8))

        r2 = r2_score(test_c[target], test_c["prediction"])



        category_metrics[cat] = {"RMSE": rmse, "MAPE": mape,
"SMAPE": smape, "R2": r2, "N_features": len(best_features)}
```

```
    АГРЕГАЦІЯ РЕЗУЛЬТАТІВ ТА SHAP-ІНТЕРПРЕТАЦІЯ

    df_all = pd.concat(category_preds, axis=0)


    # Побудова графіків тижневих трендів: Факт проти Прогнозу

    df_all["week"] = df_all["date"] -
pd.to_timedelta(df_all["date"].dt.dayofweek, unit="d")

    agg = df_all.groupby(["category", "week"])[["units_sold",
"prediction"]].sum().reset_index()


    for cat in agg["category"].unique():

        df_cat = agg[agg["category"] == cat]

        plt.figure(figsize=(10, 4))

        plt.plot(df_cat["week"], df_cat["units_sold"],
label="Actual", marker="o")

        plt.plot(df_cat["week"], df_cat["prediction"],
label="Predicted", marker="x")

        plt.title(f"{cat} - Weekly Prediction vs Actual")

        plt.xticks(rotation=45)

        plt.grid(True)

        plt.legend()

        plt.tight_layout()

        plt.show()
```

```
    # SHAP-аналіз для категорії ReadyMeal

    model = category_models["ReadyMeal"]["model"]

    used_features = category_models["ReadyMeal"]["features"]

    X_sample = df[df["category"] ==
"ReadyMeal"][used_features].sample(500, random_state=42)



    explainer = shap.Explainer(model)

    shap_values = explainer(X_sample)



    # Візуалізація важливості ознак

    shap.plots.bar(shap_values)

    shap.plots.beeswarm(shap_values)
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	69 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Нарожний М.С.			Застосування методів машинного навчання для прогнозування попиту товарів	Шифр групи	Код напр./спец.
Перевірив		Кудрявцева М.С.				ІТШІпз-24-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Чала Л.Е.					