Projet MLOps – Automation

Objectifs

Ce notebook illustre l'exécution automatisée du pipeline d'inférence : chargement d'un nouveau jeu de données, application du même prétraitement que celui utilisé lors de l'entraînement, génération des prédictions avec le modèle final, puis export des résultats pour un usage opérationnel. L'objectif est de rendre le processus reproductible : il suffit de changer le fichier d'entrée et/ou le modèle sauvegardé sans modifier la logique du code. Cette étape constitue une base pour une automatisation future via un script Python planifié (cron / tâche planifiée) ou un job dédié.

Bibliothèques

Entrée [2]:
```
import pandas as pd
import numpy as np
import joblib
from pathlib import Path
```

Paramètres

Entrée [3]:
```
DATA_NEW_PATH = Path("../data/prediction/nouvelle_data.csv")
MODEL_PATH = Path("../outputs/jobs/rf_optimized.joblib")
OUT_PATH      = Path("../data/prediction/prediction_automated/newpredictions.csv")

print("✅ Chemins prêts")
print("DATA:", DATA_NEW_PATH)
print("MODEL:", MODEL_PATH)
print("OUT:", OUT_PATH)
```

```
✅ Chemins prêts
DATA: ..\data\prediction\nouvelle_data.csv
MODEL: ..\outputs\jobs\rf_optimized.joblib
OUT: ..\data\prediction\prediction_automated\newpredictions.csv
```

Chargement des données

Entrée [4]:
```
df_new_raw = pd.read_csv(DATA_NEW_PATH, sep=";")

print("✅ Data chargée :", df_new_raw.shape)
display(df_new_raw.head())
print("Colonnes :", df_new_raw.columns.tolist())
```

```
✅ Data chargée : (9681, 14)
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain
0	41	Never-worked	156105	Prof-school	5	Married-spouse-absent	Exec-managerial	Husband	Amer-Indian-Eskimo	Female	4397
1	67	Self-emp-not-inc	350376	Prof-school	7	Widowed	Transport-moving	Other-relative	Black	Male	0
2	54	Local-gov	274063	Prof-school	14	Never-married	Handlers-cleaners	Wife	Black	Male	0
3	31	State-gov	78158	9th	9	Divorced	Exec-managerial	Husband	Black	Female	4378
4	32	Self-emp-inc	286245	Some-college	6	Married-spouse-absent	Priv-house-serv	Wife	Black	Male	0

```
Colonnes : ['age', 'workclass', 'fnlwgt', 'education', 'educational-num', 'marital-status', 'occupation', 'relationship', 'race', 'gender', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country']
```

Nettoyage

Entrée [5]:
```
import numpy as np
import pandas as pd

def clean_data(df_input: pd.DataFrame) -> pd.DataFrame:
    df_cleaned = df_input.copy()

    # remplace les "-" par "_" (ex: capital-gain -> capital_gain)
    df_cleaned.columns = df_cleaned.columns.str.replace("-", "_", regex=False)

    # 2) remplacer "?" par "Non renseigné" (sur colonnes concernées)
    for c in ["occupation", "workclass", "native_country"]:
        if c in df_cleaned.columns:
            df_cleaned[c] = df_cleaned[c].astype(str).str.replace("?", "Non renseigné", re

    # 3) âge en classes
    if "age" in df_cleaned.columns and pd.api.types.is_numeric_dtype(df_cleaned["age"]):
        bins = [0, 18, 30, 50, 65, float("inf")]
        labels = ["Moins de 18", "Entre 18 et 30 ans", "Entre 31 et 50 ans", "Entre 51 et
        df_cleaned["age"] = pd.cut(df_cleaned["age"], bins=bins, labels=labels, right=Fals

    # 4) capital_gain / capital_loss (après renommage)
    if "capital_gain" in df_cleaned.columns and pd.api.types.is_numeric_dtype(df_cleaned["
        df_cleaned["capital_gain"] = np.where(df_cleaned["capital_gain"] > 0, "Gain de cap

    if "capital_loss" in df_cleaned.columns and pd.api.types.is_numeric_dtype(df_cleaned["
        df_cleaned["capital_loss"] = np.where(df_cleaned["capital_loss"] > 0, "Perte de ca

    # 5) hours_per_week
    if "hours_per_week" in df_cleaned.columns and pd.api.types.is_numeric_dtype(df_cleaned
        df_cleaned["hours_per_week"] = pd.cut(
            df_cleaned["hours_per_week"],
            bins=[0, 40, 50, float("inf")],
            labels=["under_employed", "normally_employed", "over_employed"],
            right=False
        )

    # 6) native_country (USA / Not USA)
    if "native_country" in df_cleaned.columns:
        df_cleaned["native_country"] = np.where(
            df_cleaned["native_country"] == "United-States", "USA", "Not USA"
        )

    # 7) relationship – regroupement raisonné
    if "relationship" in df_cleaned.columns:
        df_cleaned["relationship"] = df_cleaned["relationship"].replace({
            "Husband": "Married",
            "Wife": "Married",
            "Own-child": "Own-child",
            "Not-in-family": "Others",
            "Other-relative": "Others",
            "Unmarried": "Unmarried"
        })

    # 8) race (White / Black / Other)
    if "race" in df_cleaned.columns:
        df_cleaned["race"] = np.where(
            df_cleaned["race"] == "White", "White",
            np.where(df_cleaned["race"] == "Black", "Black", "Other")
        )

    # 9) occupation mapping
    if "occupation" in df_cleaned.columns:
        occupation_mapping = {
            "Tech-support": "White Collar",
            "Craft-repair": "Blue Collar",
            "Other-service": "Other",
            "Sales": "Sales",
            "Exec-managerial": "White Collar",
            "Prof-specialty": "White Collar",
```

```
            "Handlers-cleaners": "Blue Collar",
            "Machine-op-inspct": "Blue Collar",
            "Adm-clerical": "White Collar",
            "Farming-fishing": "Agriculture",
            "Transport-moving": "Blue Collar",
            "Priv-house-serv": "Blue Collar",
            "Protective-serv": "Protective Services",
            "Armed-Forces": "Protective Services",
            "Non renseigné": "Other"
        }
        df_cleaned["occupation"] = df_cleaned["occupation"].replace(occupation_mapping)

    # 10) workclass regroupement
    if "workclass" in df_cleaned.columns:
        df_cleaned["workclass"] = df_cleaned["workclass"].replace({
            "Local-gov": "Gov",
            "State-gov": "Gov",
            "Federal-gov": "Gov",
            "Self-emp-not-inc": "Self_emp",
            "Self-emp-inc": "Self_emp",
            "Without-pay": "Other",
            "Never-worked": "Other",
            "Non renseigné": "Non renseigné"
        })

    # 11) education regroupement
    if "education" in df_cleaned.columns:
        df_cleaned["education"] = df_cleaned["education"].replace({
            "Preschool": "Primary", "1st-4th": "Primary", "5th-6th": "Primary",
            "7th-8th": "Middle_Low", "9th": "Middle_Low", "10th": "Middle_Low",
            "11th": "Middle_Low", "12th": "Middle_Low",
            "HS-grad": "HighSchool_SomeCollege", "Some-college": "HighSchool_SomeCollege",
            "Assoc-voc": "HighSchool_SomeCollege", "Assoc-acdm": "HighSchool_SomeCollege",
            "Bachelors": "Higher", "Masters": "Higher", "Prof-school": "Higher", "Doctorat
        })

    # Regroupement de marital_status (modalités courtes et en anglais)
    if "marital_status" in df_cleaned.columns:
        df_cleaned["marital_status"] = df_cleaned["marital_status"].replace({
        "Married-civ-spouse": "Married",
        "Married-AF-spouse": "Married",
        "Never-married": "Never_married",
        "Divorced": "Divorced",
        "Separated": "Separated",
        "Widowed": "Widowed",
        "Married-spouse-absent": "Separated"
        })


    #variable gender (2 modalités unique), educational_num (pas d'infos sur les modalités)

    return df_cleaned


# Application
df_new_clean = clean_data(df_new_raw)

print("✅ Data nettoyée :", df_new_clean.shape)
display(df_new_clean.head())
print("Colonnes nettoyées :", df_new_clean.columns.tolist())
```

```
✅ Data nettoyée : (9681, 14)
```

	age	workclass	fnlwgt	education	educational_num	marital_status	occupation	relationship
0	Entre 31 et 50 ans	Other	156105	Higher	5	Separated	White Collar	Married
1	Plus de 65 ans	Self_emp	350376	Higher	7	Widowed	Blue Collar	Others
2	Entre 51 et 65 ans	Gov	274063	Higher	14	Never_married	Blue Collar	Married
3	Entre 31 et 50 ans	Gov	78158	Middle_Low	9	Divorced	White Collar	Married
4	Entre 31 et 50 ans	Self_emp	286245	HighSchool_SomeCollege	6	Separated	Blue Collar	Married

```
Colonnes nettoyées : ['age', 'workclass', 'fnlwgt', 'education', 'educational_num', 'mari
tal_status', 'occupation', 'relationship', 'race', 'gender', 'capital_gain', 'capital_los
s', 'hours_per_week', 'native_country']
```

Entrée [6]:
```
# sélection des variables utilisées par le modèle
FEATURES_MODEL = ["education", "age", "hours_per_week", "occupation", "capital_gain", "mar

X_new = df_new_clean[FEATURES_MODEL].copy()
print("✅ X_new prêt :", X_new.shape)
display(X_new.head())
```

```
✅ X_new prêt : (9681, 6)
```

	education	age	hours_per_week	occupation	capital_gain	marital_status
0	Higher	Entre 31 et 50 ans	over_employed	White Collar	Gain de capital	Separated
1	Higher	Plus de 65 ans	under_employed	Blue Collar	Pas de gain de capital	Widowed
2	Higher	Entre 51 et 65 ans	over_employed	Blue Collar	Pas de gain de capital	Never_married
3	Middle_Low	Entre 31 et 50 ans	under_employed	White Collar	Gain de capital	Divorced
4	HighSchool_SomeCollege	Entre 31 et 50 ans	under_employed	Blue Collar	Pas de gain de capital	Separated

Chargement du modèle sauvegardé

Entrée [7]:
```
rf_model = joblib.load(MODEL_PATH)
print("✅ Modèle chargé :", MODEL_PATH)
```

```
✅ Modèle chargé : ..\outputs\jobs\rf_optimized.joblib
```

Prédictions

Entrée [8]:
```
y_pred_new = rf_model.predict(X_new)
y_proba_new = rf_model.predict_proba(X_new)[:, 1]  # proba de la classe ">50K"
print("✅ Prédictions générées")
```

```
✅ Prédictions générées
```

Exportations des résultats

Entrée [9]:
```
df_predictions = df_new_clean.copy()
df_predictions["income_predicted"] = y_pred_new
df_predictions["proba_>50K"] = y_proba_new

# Création auto du dossier de sortie
OUT_PATH.parent.mkdir(parents=True, exist_ok=True)

df_predictions.to_csv(OUT_PATH, index=False, sep=";")
print("✅ Export terminé :", OUT_PATH)

display(df_predictions.head())
```

```
✅ Export terminé : ..\data\prediction\prediction_automated\newpredictions.csv
```

	age	workclass	fnlwgt	education	educational_num	marital_status	occupation	relationship
0	Entre 31 et 50 ans	Other	156105	Higher	5	Separated	White Collar	Married
1	Plus de 65 ans	Self_emp	350376	Higher	7	Widowed	Blue Collar	Others
2	Entre 51 et 65 ans	Gov	274063	Higher	14	Never_married	Blue Collar	Married
3	Entre 31 et 50 ans	Gov	78158	Middle_Low	9	Divorced	White Collar	Married
4	Entre 31 et 50 ans	Self_emp	286245	HighSchool_SomeCollege	6	Separated	Blue Collar	Married

Entrée [10]:
```
# utils/io.py
from pathlib import Path
import matplotlib.pyplot as plt

BASE_OUTPUT = Path("../outputs")
FIGURES = BASE_OUTPUT / "figures"
TABLES = BASE_OUTPUT / "tables"

FIGURES.mkdir(parents=True, exist_ok=True)
TABLES.mkdir(parents=True, exist_ok=True)

def save_figure(name):
    plt.savefig(FIGURES / name, dpi=300, bbox_inches="tight")
    plt.close()

def save_table(df, name):
    df.to_csv(TABLES / name, index=False)
```

Entrée [11]:
```
from utils.io import save_figure, save_table
save_figure("roc_rf_optimized.png")
save_table(model_results, "model_comparison.csv")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[11], line 1
----> 1 from utils.io import save_figure, save_table
      2 save_figure("roc_rf_optimized.png")
      3 save_table(model_results, "model_comparison.csv")

ModuleNotFoundError: No module named 'utils'
```

Entrée [ ]:
```
```