**MINI PROJECT 1**

**Data Preprocessing and Feature Engineering**

**1. Data Collection:**

* **Choose a Dataset:**
  + Visit a reliable dataset repository such as Kaggle, UCI Machine Learning Repository, or another trusted source.
  + Choose a dataset that contains a mix of numerical and categorical features. Ensure that it is suitable for a machine learning task (classification, regression, etc.).

**2. Data Inspection:**

* **Overview:**
  + Load the dataset and provide an overview including:
    - The number of samples (rows) and features (columns).
    - The target variable (if applicable).
  + **Initial Observations:**
    - Look for any obvious issues such as missing values, imbalanced classes, or unusual distributions.
    - Mention any challenges you anticipate during preprocessing.

**3. Data Preprocessing:**

* **Data Cleaning:**
  + **Missing Values:**
    - Identify missing values in the dataset.
    - Choose an appropriate method to handle missing data (e.g., imputation, deletion, or using models to predict missing values).
    - Explain why you chose this method.
* **Feature Scaling:**
  + **Normalization:**
    - Apply feature scaling to the numerical features using techniques such as Standardization (z-score normalization) or Min-Max scaling.
    - Explain the rationale behind choosing the scaling method.
* **Handling Categorical Data:**
  + **Encoding:**
    - Encode categorical variables using techniques such as One-Hot Encoding or Label Encoding.
    - Explain your choice of encoding method and how it helps the model to interpret categorical features.

**4. Feature Engineering:**

* **Create New Features:**
  + Apply at least two feature engineering techniques. Examples include:
    - Polynomial features, interaction terms, binning continuous variables, or domain-specific features.
    - Explain the logic behind each technique and how it improves the dataset's ability to represent the underlying patterns.

**5. Handling Imbalanced Data:**

* **Addressing Imbalance:**
  + If applicable, address class imbalance in the target variable using techniques like oversampling, undersampling, or SMOTE (Synthetic Minority Over-sampling Technique).
  + Describe the method used and justify why it was chosen.

**6. Data Transformation:**

* **Save Preprocessed Data:**
  + After completing the preprocessing and feature engineering steps, save the cleaned and transformed dataset as a CSV file.
  + Include a link or attachment to this CSV file in your submission.

**7. Analysis:**

* **Visualizations and Summary Statistics:**
  + Provide visualizations (e.g., histograms, box plots, correlation matrices) to illustrate the effects of your preprocessing and feature engineering steps.
  + Include summary statistics before and after preprocessing.
  + Discuss how these steps have improved the dataset’s readiness for machine learning.

**8. Conclusion:**

* **Summarize Key Takeaways:**
  + Reflect on the importance of each step in the preprocessing and feature engineering process.
  + Highlight how these techniques contribute to building effective machine learning models.

**9. Submission Guidelines:**

* Code

import pandas as pd

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler

from sklearn.impute import SimpleImputer

from imblearn.over\_sampling import SMOTE

# Load the dataset

data = pd.read\_csv("adult.data", header=None)

columns = ["age", "workclass", "fnlwgt", "education", "education-num", "marital-status", "occupation", "relationship", "race", "sex", "capital-gain", "capital-loss", "hours-per-week", "native-country", "income"]

data.columns = columns

# Handle missing values

imputer = SimpleImputer(strategy="most\_frequent")

data["occupation"] = imputer.fit\_transform(data["occupation"].values.reshape(-1, 1))

data["native-country"] = imputer.fit\_transform(data["native-country"].values.reshape(-1, 1))

# Feature scaling

scaler = StandardScaler()

data[["age", "fnlwgt", "education-num", "capital-gain", "capital-loss", "hours-per-week"]] = scaler.fit\_transform(data[["age", "fnlwgt", "education-num", "capital-gain", "capital-loss", "hours-per-week"]])

# Encode categorical features

le = LabelEncoder()

data["workclass"] = le.fit\_transform(data["workclass"])

data["marital-status"] = le.fit\_transform(data["marital-status"])

data["relationship"] = le.fit\_transform(data["relationship"])

data["race"] = le.fit\_transform(data["race"])

data["sex"] = le.fit\_transform(data["sex"])

# One-hot encode remaining categorical features

ohe = OneHotEncoder(sparse=False)

encoded\_features = ohe.fit\_transform(data[["education", "occupation", "native-country"]])

data = data.drop(["education", "occupation", "native-country"], axis=1)

data = pd.concat([data, pd.DataFrame(encoded\_features, columns=ohe.get\_feature\_names\_out())], axis=1)

# Feature engineering

data["age\_bin"] = pd.cut(data["age"], bins=[0, 18, 30, 45, 60, 100], labels=["0-18", "19-30", "31-45", "46-60", "61+"])

data["education\_occupation"] = data["education"] + "\_" + data["occupation"]

# Handle class imbalance

X = data.drop("income", axis=1)

y = data["income"]

smote = SMOTE(random\_state=42)

X\_resampled, y\_resampled = smote.fit\_resample(X, y)

# Save the preprocessed dataset

X\_resampled.to\_csv("preprocessed\_adult\_income.csv", index=False)