Project Code: Mercedes-Benz Greener Manufacturing

Source Code:

```
import pandas as pd
import numpy as np
import logging
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer

logging.basicConfig(level=logging.INFO)

# Load datasets
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")

# Ensure 'y' column exists
if 'y' not in train_df.columns:
raise ValueError("Target variable 'y' not found in training data.")

# Separate target variable
y = train_df['y']
train_df.drop(columns=['ID', 'y'], inplace=True)
test_ids = test_df['ID']
test_df.drop(columns=['ID'], inplace=True)

# Encode categorical variables
for col in train_df.select_dtypes(include=['object']).columns:
encoder = LabelEncoder()
all_values = pd.concat([train_df[col], test_df[col]], axis=0)
encoder.fit(all_values)
```

```
train_df[col] = encoder.transform(train_df[col])
test_df[col] = encoder.transform(test_df[col])

# Function to remove zero-variance columns consistently
def remove_zero_variance(df, reference_cols=None):
if reference_cols is None:
zero_variance_cols = df.var(numeric_only=True) == 0
reference_cols = df.columns[~zero_variance_cols]
return df[reference_cols], reference_cols

# Apply zero-variance column removal
train_df, reference_cols = remove_zero_variance(train_df)
test_df = test_df[reference_cols]  # Ensure test set has the same features

# Handle missing values using median imputation
imputer = SimpleImputer(strategy='median')
train_df[:] = imputer.fit_transform(train_df)
test_df[:] = imputer.transform(test_df)

# Ensure test set has the same features as training before PCA
test_df = test_df.reindex(columns=train_df.columns, fill_value=0)

# Apply PCA for dimensionality reduction
pca_threshold = 0.95  # Explained variance ratio threshold
pca = PCA(n_components=pca_threshold)
X_pca = pca.fit_transform(train_df)
test_pca = pca.transform(test_df)

# Train-validation split
X_train, X_valid, y_train, y_valid = train_test_split(X_pca, y, test_size=0.2, random_state=42)
```

```
# Log feature consistency
train_features = set(train_df.columns)
test_features = set(test_df.columns)

missing_in_test = train_features - test_features
extra_in_test = test_features - train_features

logging.info(f"Missing in test: {missing_in_test}")
logging.info(f"Extra in test: {extra_in_test}")

print("Preprocessing completed successfully.")
```

ScreenShots: