DATA PREPROCESSING

Preprocessing User Data

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
df = pd.read_csv('cleaned_dataset.csv')
df.head()
# Identify relevant columns
numerical_columns = ['Reward Amount', 'Skill Points Earned']
categorical_columns = ['Opportunity Category', 'Status Description']

# Task 1: Handling Outliers and Anomalies
# Apply outlier detection techniques (you can customize this based on your data)
def handle_outliers_zscore(series, threshold=3):
    z_scores = (series - series.mean()) / series.std()
    return series[abs(z_scores) < threshold]

# Apply Z-score outlier detection
for column in numerical_columns:
    df[column] = handle_outliers_zscore(df[column])
# Task 2: Normalize or Scale Relevant Features
numerical_features = numerical_columns
categorical_features = categorical_columns

numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),  # Impute missing values with mean (you can ch
oose a different strategy)
    ('scaler', MinMaxScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
])
# Combine transformers using ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

df_transformed = preprocessor.fit_transform(df)
# Task 4: Feature Engineering
# Convert 'Skill Points Earned' to numeric type
df['Skill Points Earned'] = pd.to_numeric(df['Skill Points Earned'], errors='coerce')

# Convert 'Opportunity Start Date' to Unix timestamp (in seconds) and then to int64
df['Opportunity Start Date'] = pd.to_datetime(df['Opportunity Start Date'], errors='coerce').ast
ype('int64') // 10**9
```

```
# Convert 'Skills Earned' to numeric type
df['Skills Earned'] = pd.to_numeric(df['Skills Earned'], errors='coerce')

# Create a new feature: completion rate for each opportunity
df['Completion Rate'] = df['Skills Earned'] / df['Opportunity Start Date'].astype('float')  # Assu
ming 'Skills Earned' is now in numeric format
# Task 5: Data Transformation
# Perform one-hot encoding for categorical variables
df_transformed = preprocessor.fit_transform(df)
```

Preprocessing Opportunity Signup and Completion Data

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

df = pd.read_csv('cleaned_dataset.csv')
df.head()
# Identify relevant columns
numerical_columns = ['Reward Amount', 'Skill Points Earned']
categorical_columns = ['Opportunity Category', 'Status Description']

# Task 1: Handling Outliers and Anomalies
# Apply outlier detection techniques (you can customize this based on your data)
def handle_outliers_zscore(series, threshold=3):
    z_scores = (series - series.mean()) / series.std()
    return series[abs(z_scores) < threshold]

# Apply Z-score outlier detection
for column in numerical_columns:
    df[column] = handle_outliers_zscore(df[column])
# Task 2: Normalize or Scale Relevant Features
numerical_features = numerical_columns
categorical_features = categorical_columns

numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),  # Impute missing values with mean (you can ch
oose a different strategy)
    ('scaler', MinMaxScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
])
# Combine transformers using ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

df_transformed = preprocessor.fit_transform(df)
# Task 4: Feature Engineering
# Convert 'Skill Points Earned' to numeric type
df['Skill Points Earned'] = pd.to_numeric(df['Skill Points Earned'], errors='coerce')

# Convert 'Opportunity Start Date' to Unix timestamp (in seconds) and then to int64
df['Opportunity Start Date'] = pd.to_datetime(df['Opportunity Start Date'], errors='coerce').ast
ype('int64') // 10**9

# Convert 'Skills Earned' to numeric type
df['Skills Earned'] = pd.to_numeric(df['Skills Earned'], errors='coerce')

# Create a new feature: completion rate for each opportunity
df['Completion Rate'] = df['Skills Earned'] / df['Opportunity Start Date'].astype('float')  # Assu
ming 'Skills Earned' is now in numeric format
# Task 5: Data Transformation
# Perform one-hot encoding for categorical variables
df_transformed = preprocessor.fit_transform(df)
```