import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.decomposition import PCA

from xgboost import XGBRegressor

from sklearn.metrics import mean\_squared\_error

train\_df = pd.read\_csv('train.csv')

test\_df = pd.read\_csv('test.csv')

#I removed Columns with Zero variance (no differnce)

numeric\_columns\_train = train\_df.select\_dtypes(include=[np.number])

zero\_variance\_cols\_train = numeric\_columns\_train.columns[numeric\_columns\_train.var() == 0]

train\_df = train\_df.drop(zero\_variance\_cols\_train, axis=1)

numeric\_columns\_test = test\_df.select\_dtypes(include=[np.number])

zero\_variance\_cols\_test = numeric\_columns\_test.columns[numeric\_columns\_test.var() == 0]

test\_df = test\_df.drop(zero\_variance\_cols\_test, axis=1)

# Checked null values in the training data

print("Null values in training set:")

print(train\_df.isnull().sum())

# Checking unique values in the training data

print("\nUnique values in training set:")

print(train\_df.nunique())

# Checked for null values in the test data

print("\nNull values in test set:")

print(test\_df.isnull().sum())

# Checked for unique values in the test data

print("\nUnique values in test set:")

print(test\_df.nunique())

#label encoding

label\_encoder = LabelEncoder()

for col in train\_df.select\_dtypes(include=['object']).columns:

train\_df[col] = label\_encoder.fit\_transform(train\_df[col])

for col in test\_df.select\_dtypes(include=['object']).columns:

test\_df[col] = label\_encoder.fit\_transform(test\_df[col])

common\_columns = list(set(train\_df.columns).intersection(test\_df.columns))

X\_test = test\_df[common\_columns]

for col in train\_df.select\_dtypes(include=['object']).columns:

train\_df[col] = label\_encoder.fit\_transform(train\_df[col])

test\_df[col] = label\_encoder.transform(test\_df[col])

#dimensionality reduction

pca = PCA(n\_components=0.95)

X\_train = train\_df.drop('y', axis=1)

X\_test = test\_df

X\_train\_pca = pca.fit\_transform(X\_train)

missing\_columns = set(X\_train.columns) - set(X\_test.columns)

for col in missing\_columns:

X\_test[col] = 0

X\_test = X\_test[X\_train.columns]

X\_test\_pca = pca.transform(X\_test)

#predicted values using XGBoost

model = XGBRegressor()

model.fit(X\_train\_pca, train\_df['y'])

test\_predictions = model.predict(X\_test\_pca)

#finding dataset

print(test\_predictions)

import matplotlib.pyplot as plt

# histogram

plt.figure(figsize=(10, 6))

plt.hist(test\_predictions, bins=30, color='blue', alpha=0.7)

plt.xlabel('Predicted Values')

plt.ylabel('Frequency')

plt.title('Histogram - Test Predictions')

plt.grid(True)

plt.show()