{  
 "cells": [  
 {  
 "cell\_type": "markdown",  
 "id": "7daa4dc4",  
 "metadata": {},  
 "source": [  
 "### Importing Libraries and Datasets"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 1,  
 "id": "a21a251d",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "# Importing libraries\n",  
 "import numpy as np\n",  
 "import pandas as pd\n",  
 "import matplotlib.pyplot as plt\n",  
 "import seaborn as sns"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 2,  
 "id": "c03fb583",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "# Reading the dataset\n",  
 "df1 = pd.read\_csv('train.csv')\n",  
 "df2 = pd.read\_csv('test.csv')"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 3,  
 "id": "cfd2654c",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<div>\n",  
 "<style scoped>\n",  
 " .dataframe tbody tr th:only-of-type {\n",  
 " vertical-align: middle;\n",  
 " }\n",  
 "\n",  
 " .dataframe tbody tr th {\n",  
 " vertical-align: top;\n",  
 " }\n",  
 "\n",  
 " .dataframe thead th {\n",  
 " text-align: right;\n",  
 " }\n",  
 "</style>\n",  
 "<table border=\"1\" class=\"dataframe\">\n",  
 " <thead>\n",  
 " <tr style=\"text-align: right;\">\n",  
 " <th></th>\n",  
 " <th>ID</th>\n",  
 " <th>y</th>\n",  
 " <th>X0</th>\n",  
 " <th>X1</th>\n",  
 " <th>X2</th>\n",  
 " <th>X3</th>\n",  
 " <th>X4</th>\n",  
 " <th>X5</th>\n",  
 " <th>X6</th>\n",  
 " <th>X8</th>\n",  
 " <th>...</th>\n",  
 " <th>X375</th>\n",  
 " <th>X376</th>\n",  
 " <th>X377</th>\n",  
 " <th>X378</th>\n",  
 " <th>X379</th>\n",  
 " <th>X380</th>\n",  
 " <th>X382</th>\n",  
 " <th>X383</th>\n",  
 " <th>X384</th>\n",  
 " <th>X385</th>\n",  
 " </tr>\n",  
 " </thead>\n",  
 " <tbody>\n",  
 " <tr>\n",  
 " <th>0</th>\n",  
 " <td>0</td>\n",  
 " <td>130.81</td>\n",  
 " <td>k</td>\n",  
 " <td>v</td>\n",  
 " <td>at</td>\n",  
 " <td>a</td>\n",  
 " <td>d</td>\n",  
 " <td>u</td>\n",  
 " <td>j</td>\n",  
 " <td>o</td>\n",  
 " <td>...</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>1</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>1</th>\n",  
 " <td>6</td>\n",  
 " <td>88.53</td>\n",  
 " <td>k</td>\n",  
 " <td>t</td>\n",  
 " <td>av</td>\n",  
 " <td>e</td>\n",  
 " <td>d</td>\n",  
 " <td>y</td>\n",  
 " <td>l</td>\n",  
 " <td>o</td>\n",  
 " <td>...</td>\n",  
 " <td>1</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>2</th>\n",  
 " <td>7</td>\n",  
 " <td>76.26</td>\n",  
 " <td>az</td>\n",  
 " <td>w</td>\n",  
 " <td>n</td>\n",  
 " <td>c</td>\n",  
 " <td>d</td>\n",  
 " <td>x</td>\n",  
 " <td>j</td>\n",  
 " <td>x</td>\n",  
 " <td>...</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>1</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>3</th>\n",  
 " <td>9</td>\n",  
 " <td>80.62</td>\n",  
 " <td>az</td>\n",  
 " <td>t</td>\n",  
 " <td>n</td>\n",  
 " <td>f</td>\n",  
 " <td>d</td>\n",  
 " <td>x</td>\n",  
 " <td>l</td>\n",  
 " <td>e</td>\n",  
 " <td>...</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>4</th>\n",  
 " <td>13</td>\n",  
 " <td>78.02</td>\n",  
 " <td>az</td>\n",  
 " <td>v</td>\n",  
 " <td>n</td>\n",  
 " <td>f</td>\n",  
 " <td>d</td>\n",  
 " <td>h</td>\n",  
 " <td>d</td>\n",  
 " <td>n</td>\n",  
 " <td>...</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " </tbody>\n",  
 "</table>\n",  
 "<p>5 rows × 378 columns</p>\n",  
 "</div>"  
 ],  
 "text/plain": [  
 " ID y X0 X1 X2 X3 X4 X5 X6 X8 ... X375 X376 X377 X378 X379 \\\n",  
 "0 0 130.81 k v at a d u j o ... 0 0 1 0 0 \n",  
 "1 6 88.53 k t av e d y l o ... 1 0 0 0 0 \n",  
 "2 7 76.26 az w n c d x j x ... 0 0 0 0 0 \n",  
 "3 9 80.62 az t n f d x l e ... 0 0 0 0 0 \n",  
 "4 13 78.02 az v n f d h d n ... 0 0 0 0 0 \n",  
 "\n",  
 " X380 X382 X383 X384 X385 \n",  
 "0 0 0 0 0 0 \n",  
 "1 0 0 0 0 0 \n",  
 "2 0 1 0 0 0 \n",  
 "3 0 0 0 0 0 \n",  
 "4 0 0 0 0 0 \n",  
 "\n",  
 "[5 rows x 378 columns]"  
 ]  
 },  
 "execution\_count": 3,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "df1.head()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 4,  
 "id": "eac383ab",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<div>\n",  
 "<style scoped>\n",  
 " .dataframe tbody tr th:only-of-type {\n",  
 " vertical-align: middle;\n",  
 " }\n",  
 "\n",  
 " .dataframe tbody tr th {\n",  
 " vertical-align: top;\n",  
 " }\n",  
 "\n",  
 " .dataframe thead th {\n",  
 " text-align: right;\n",  
 " }\n",  
 "</style>\n",  
 "<table border=\"1\" class=\"dataframe\">\n",  
 " <thead>\n",  
 " <tr style=\"text-align: right;\">\n",  
 " <th></th>\n",  
 " <th>ID</th>\n",  
 " <th>X0</th>\n",  
 " <th>X1</th>\n",  
 " <th>X2</th>\n",  
 " <th>X3</th>\n",  
 " <th>X4</th>\n",  
 " <th>X5</th>\n",  
 " <th>X6</th>\n",  
 " <th>X8</th>\n",  
 " <th>X10</th>\n",  
 " <th>...</th>\n",  
 " <th>X375</th>\n",  
 " <th>X376</th>\n",  
 " <th>X377</th>\n",  
 " <th>X378</th>\n",  
 " <th>X379</th>\n",  
 " <th>X380</th>\n",  
 " <th>X382</th>\n",  
 " <th>X383</th>\n",  
 " <th>X384</th>\n",  
 " <th>X385</th>\n",  
 " </tr>\n",  
 " </thead>\n",  
 " <tbody>\n",  
 " <tr>\n",  
 " <th>0</th>\n",  
 " <td>1</td>\n",  
 " <td>az</td>\n",  
 " <td>v</td>\n",  
 " <td>n</td>\n",  
 " <td>f</td>\n",  
 " <td>d</td>\n",  
 " <td>t</td>\n",  
 " <td>a</td>\n",  
 " <td>w</td>\n",  
 " <td>0</td>\n",  
 " <td>...</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>1</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>1</th>\n",  
 " <td>2</td>\n",  
 " <td>t</td>\n",  
 " <td>b</td>\n",  
 " <td>ai</td>\n",  
 " <td>a</td>\n",  
 " <td>d</td>\n",  
 " <td>b</td>\n",  
 " <td>g</td>\n",  
 " <td>y</td>\n",  
 " <td>0</td>\n",  
 " <td>...</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>1</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>2</th>\n",  
 " <td>3</td>\n",  
 " <td>az</td>\n",  
 " <td>v</td>\n",  
 " <td>as</td>\n",  
 " <td>f</td>\n",  
 " <td>d</td>\n",  
 " <td>a</td>\n",  
 " <td>j</td>\n",  
 " <td>j</td>\n",  
 " <td>0</td>\n",  
 " <td>...</td>\n",  
 " <td>0</td>\n",

" <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>1</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>3</th>\n",  
 " <td>4</td>\n",  
 " <td>az</td>\n",  
 " <td>l</td>\n",  
 " <td>n</td>\n",  
 " <td>f</td>\n",  
 " <td>d</td>\n",  
 " <td>z</td>\n",  
 " <td>l</td>\n",  
 " <td>n</td>\n",  
 " <td>0</td>\n",  
 " <td>...</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>1</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>4</th>\n",  
 " <td>5</td>\n",  
 " <td>w</td>\n",  
 " <td>s</td>\n",  
 " <td>as</td>\n",  
 " <td>c</td>\n",  
 " <td>d</td>\n",  
 " <td>y</td>\n",  
 " <td>i</td>\n",  
 " <td>m</td>\n",  
 " <td>0</td>\n",  
 " <td>...</td>\n",  
 " <td>1</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " <td>0</td>\n",  
 " </tr>\n",  
 " </tbody>\n",  
 "</table>\n",  
 "<p>5 rows × 377 columns</p>\n",  
 "</div>"  
 ],  
 "text/plain": [  
 " ID X0 X1 X2 X3 X4 X5 X6 X8 X10 ... X375 X376 X377 X378 X379 X380 \\\n",  
 "0 1 az v n f d t a w 0 ... 0 0 0 1 0 0 \n",  
 "1 2 t b ai a d b g y 0 ... 0 0 1 0 0 0 \n",  
 "2 3 az v as f d a j j 0 ... 0 0 0 1 0 0 \n",  
 "3 4 az l n f d z l n 0 ... 0 0 0 1 0 0 \n",  
 "4 5 w s as c d y i m 0 ... 1 0 0 0 0 0 \n",  
 "\n",  
 " X382 X383 X384 X385 \n",  
 "0 0 0 0 0 \n",  
 "1 0 0 0 0 \n",  
 "2 0 0 0 0 \n",  
 "3 0 0 0 0 \n",  
 "4 0 0 0 0 \n",  
 "\n",  
 "[5 rows x 377 columns]"  
 ]  
 },  
 "execution\_count": 4,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "df2.head()"  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "id": "af73d8c8",  
 "metadata": {},  
 "source": [  
 "### Data Exploration"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 5,  
 "id": "5413e13e",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<div>\n",  
 "<style scoped>\n",  
 " .dataframe tbody tr th:only-of-type {\n",  
 " vertical-align: middle;\n",  
 " }\n",  
 "\n",  
 " .dataframe tbody tr th {\n",  
 " vertical-align: top;\n",  
 " }\n",  
 "\n",  
 " .dataframe thead th {\n",  
 " text-align: right;\n",  
 " }\n",  
 "</style>\n",  
 "<table border=\"1\" class=\"dataframe\">\n",  
 " <thead>\n",  
 " <tr style=\"text-align: right;\">\n",  
 " <th></th>\n",  
 " <th>ID</th>\n",  
 " <th>y</th>\n",  
 " <th>X10</th>\n",  
 " <th>X11</th>\n",  
 " <th>X12</th>\n",  
 " <th>X13</th>\n",  
 " <th>X14</th>\n",  
 " <th>X15</th>\n",  
 " <th>X16</th>\n",  
 " <th>X17</th>\n",  
 " <th>...</th>\n",  
 " <th>X375</th>\n",  
 " <th>X376</th>\n",  
 " <th>X377</th>\n",  
 " <th>X378</th>\n",  
 " <th>X379</th>\n",  
 " <th>X380</th>\n",  
 " <th>X382</th>\n",  
 " <th>X383</th>\n",  
 " <th>X384</th>\n",  
 " <th>X385</th>\n",  
 " </tr>\n",  
 " </thead>\n",  
 " <tbody>\n",  
 " <tr>\n",  
 " <th>count</th>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.0</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>mean</th>\n",  
 " <td>4205.960798</td>\n",  
 " <td>100.669318</td>\n",  
 " <td>0.013305</td>\n",  
 " <td>0.0</td>\n",  
 " <td>0.075077</td>\n",  
 " <td>0.057971</td>\n",  
 " <td>0.428130</td>\n",  
 " <td>0.000475</td>\n",  
 " <td>0.002613</td>\n",  
 " <td>0.007603</td>\n",  
 " <td>...</td>\n",  
 " <td>0.318841</td>\n",  
 " <td>0.057258</td>\n",  
 " <td>0.314802</td>\n",  
 " <td>0.020670</td>\n",  
 " <td>0.009503</td>\n",  
 " <td>0.008078</td>\n",  
 " <td>0.007603</td>\n",  
 " <td>0.001663</td>\n",  
 " <td>0.000475</td>\n",  
 " <td>0.001426</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>std</th>\n",  
 " <td>2437.608688</td>\n",  
 " <td>12.679381</td>\n",  
 " <td>0.114590</td>\n",  
 " <td>0.0</td>\n",  
 " <td>0.263547</td>\n",  
 " <td>0.233716</td>\n",  
 " <td>0.494867</td>\n",  
 " <td>0.021796</td>\n",  
 " <td>0.051061</td>\n",  
 " <td>0.086872</td>\n",  
 " <td>...</td>\n",  
 " <td>0.466082</td>\n",  
 " <td>0.232363</td>\n",  
 " <td>0.464492</td>\n",  
 " <td>0.142294</td>\n",  
 " <td>0.097033</td>\n",  
 " <td>0.089524</td>\n",  
 " <td>0.086872</td>\n",  
 " <td>0.040752</td>\n",  
 " <td>0.021796</td>\n",  
 " <td>0.037734</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>min</th>\n",  
 " <td>0.000000</td>\n",  
 " <td>72.110000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.0</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>25%</th>\n",  
 " <td>2095.000000</td>\n",  
 " <td>90.820000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.0</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>50%</th>\n",  
 " <td>4220.000000</td>\n",  
 " <td>99.150000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.0</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>75%</th>\n",  
 " <td>6314.000000</td>\n",  
 " <td>109.010000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.0</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>1.000000</td>\n",

" <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>max</th>\n",  
 " <td>8417.000000</td>\n",  
 " <td>265.320000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>0.0</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " </tr>\n",  
 " </tbody>\n",  
 "</table>\n",  
 "<p>8 rows × 370 columns</p>\n",  
 "</div>"  
 ],  
 "text/plain": [  
 " ID y X10 X11 X12 \\\n",  
 "count 4209.000000 4209.000000 4209.000000 4209.0 4209.000000 \n",  
 "mean 4205.960798 100.669318 0.013305 0.0 0.075077 \n",  
 "std 2437.608688 12.679381 0.114590 0.0 0.263547 \n",  
 "min 0.000000 72.110000 0.000000 0.0 0.000000 \n",  
 "25% 2095.000000 90.820000 0.000000 0.0 0.000000 \n",  
 "50% 4220.000000 99.150000 0.000000 0.0 0.000000 \n",  
 "75% 6314.000000 109.010000 0.000000 0.0 0.000000 \n",  
 "max 8417.000000 265.320000 1.000000 0.0 1.000000 \n",  
 "\n",  
 " X13 X14 X15 X16 X17 ... \\\n",  
 "count 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 ... \n",  
 "mean 0.057971 0.428130 0.000475 0.002613 0.007603 ... \n",  
 "std 0.233716 0.494867 0.021796 0.051061 0.086872 ... \n",  
 "min 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n",  
 "25% 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n",  
 "50% 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n",  
 "75% 0.000000 1.000000 0.000000 0.000000 0.000000 ... \n",  
 "max 1.000000 1.000000 1.000000 1.000000 1.000000 ... \n",  
 "\n",  
 " X375 X376 X377 X378 X379 \\\n",  
 "count 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 \n",  
 "mean 0.318841 0.057258 0.314802 0.020670 0.009503 \n",  
 "std 0.466082 0.232363 0.464492 0.142294 0.097033 \n",  
 "min 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "25% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "50% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "75% 1.000000 0.000000 1.000000 0.000000 0.000000 \n",  
 "max 1.000000 1.000000 1.000000 1.000000 1.000000 \n",  
 "\n",  
 " X380 X382 X383 X384 X385 \n",  
 "count 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 \n",  
 "mean 0.008078 0.007603 0.001663 0.000475 0.001426 \n",  
 "std 0.089524 0.086872 0.040752 0.021796 0.037734 \n",  
 "min 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "25% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "50% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "75% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "max 1.000000 1.000000 1.000000 1.000000 1.000000 \n",  
 "\n",  
 "[8 rows x 370 columns]"  
 ]  
 },  
 "execution\_count": 5,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "# Describe the dataset\n",  
 "df1.describe()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 6,  
 "id": "f40a5403",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<div>\n",  
 "<style scoped>\n",  
 " .dataframe tbody tr th:only-of-type {\n",  
 " vertical-align: middle;\n",  
 " }\n",  
 "\n",  
 " .dataframe tbody tr th {\n",  
 " vertical-align: top;\n",  
 " }\n",  
 "\n",  
 " .dataframe thead th {\n",  
 " text-align: right;\n",  
 " }\n",  
 "</style>\n",  
 "<table border=\"1\" class=\"dataframe\">\n",  
 " <thead>\n",  
 " <tr style=\"text-align: right;\">\n",  
 " <th></th>\n",  
 " <th>ID</th>\n",  
 " <th>X10</th>\n",  
 " <th>X11</th>\n",  
 " <th>X12</th>\n",  
 " <th>X13</th>\n",  
 " <th>X14</th>\n",  
 " <th>X15</th>\n",  
 " <th>X16</th>\n",  
 " <th>X17</th>\n",  
 " <th>X18</th>\n",  
 " <th>...</th>\n",  
 " <th>X375</th>\n",  
 " <th>X376</th>\n",  
 " <th>X377</th>\n",  
 " <th>X378</th>\n",  
 " <th>X379</th>\n",  
 " <th>X380</th>\n",  
 " <th>X382</th>\n",  
 " <th>X383</th>\n",  
 " <th>X384</th>\n",  
 " <th>X385</th>\n",  
 " </tr>\n",  
 " </thead>\n",  
 " <tbody>\n",  
 " <tr>\n",  
 " <th>count</th>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " <td>4209.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>mean</th>\n",  
 " <td>4211.039202</td>\n",  
 " <td>0.019007</td>\n",  
 " <td>0.000238</td>\n",  
 " <td>0.074364</td>\n",  
 " <td>0.061060</td>\n",  
 " <td>0.427893</td>\n",  
 " <td>0.000713</td>\n",  
 " <td>0.002613</td>\n",  
 " <td>0.008791</td>\n",  
 " <td>0.010216</td>\n",  
 " <td>...</td>\n",  
 " <td>0.325968</td>\n",  
 " <td>0.049656</td>\n",  
 " <td>0.311951</td>\n",  
 " <td>0.019244</td>\n",  
 " <td>0.011879</td>\n",  
 " <td>0.008078</td>\n",  
 " <td>0.008791</td>\n",  
 " <td>0.000475</td>\n",  
 " <td>0.000713</td>\n",  
 " <td>0.001663</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>std</th>\n",  
 " <td>2423.078926</td>\n",  
 " <td>0.136565</td>\n",  
 " <td>0.015414</td>\n",  
 " <td>0.262394</td>\n",  
 " <td>0.239468</td>\n",  
 " <td>0.494832</td>\n",  
 " <td>0.026691</td>\n",  
 " <td>0.051061</td>\n",  
 " <td>0.093357</td>\n",  
 " <td>0.100570</td>\n",  
 " <td>...</td>\n",  
 " <td>0.468791</td>\n",  
 " <td>0.217258</td>\n",  
 " <td>0.463345</td>\n",  
 " <td>0.137399</td>\n",  
 " <td>0.108356</td>\n",  
 " <td>0.089524</td>\n",  
 " <td>0.093357</td>\n",  
 " <td>0.021796</td>\n",  
 " <td>0.026691</td>\n",  
 " <td>0.040752</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>min</th>\n",  
 " <td>1.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>25%</th>\n",  
 " <td>2115.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",

" <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>50%</th>\n",  
 " <td>4202.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>75%</th>\n",  
 " <td>6310.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " <td>0.000000</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>max</th>\n",  
 " <td>8416.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>...</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " <td>1.000000</td>\n",  
 " </tr>\n",  
 " </tbody>\n",  
 "</table>\n",  
 "<p>8 rows × 369 columns</p>\n",  
 "</div>"  
 ],  
 "text/plain": [  
 " ID X10 X11 X12 X13 \\\n",  
 "count 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 \n",  
 "mean 4211.039202 0.019007 0.000238 0.074364 0.061060 \n",  
 "std 2423.078926 0.136565 0.015414 0.262394 0.239468 \n",  
 "min 1.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "25% 2115.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "50% 4202.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "75% 6310.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "max 8416.000000 1.000000 1.000000 1.000000 1.000000 \n",  
 "\n",  
 " X14 X15 X16 X17 X18 ... \\\n",  
 "count 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 ... \n",  
 "mean 0.427893 0.000713 0.002613 0.008791 0.010216 ... \n",  
 "std 0.494832 0.026691 0.051061 0.093357 0.100570 ... \n",  
 "min 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n",  
 "25% 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n",  
 "50% 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n",  
 "75% 1.000000 0.000000 0.000000 0.000000 0.000000 ... \n",  
 "max 1.000000 1.000000 1.000000 1.000000 1.000000 ... \n",  
 "\n",  
 " X375 X376 X377 X378 X379 \\\n",  
 "count 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 \n",  
 "mean 0.325968 0.049656 0.311951 0.019244 0.011879 \n",  
 "std 0.468791 0.217258 0.463345 0.137399 0.108356 \n",  
 "min 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "25% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "50% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "75% 1.000000 0.000000 1.000000 0.000000 0.000000 \n",  
 "max 1.000000 1.000000 1.000000 1.000000 1.000000 \n",  
 "\n",  
 " X380 X382 X383 X384 X385 \n",  
 "count 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 \n",  
 "mean 0.008078 0.008791 0.000475 0.000713 0.001663 \n",  
 "std 0.089524 0.093357 0.021796 0.026691 0.040752 \n",  
 "min 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "25% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "50% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "75% 0.000000 0.000000 0.000000 0.000000 0.000000 \n",  
 "max 1.000000 1.000000 1.000000 1.000000 1.000000 \n",  
 "\n",  
 "[8 rows x 369 columns]"  
 ]  
 },  
 "execution\_count": 6,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "df2.describe()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 7,  
 "id": "df93b1dc",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "(4209, 378)"  
 ]  
 },  
 "execution\_count": 7,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "df1.shape"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 8,  
 "id": "be9ca935",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "(4209, 377)"  
 ]  
 },  
 "execution\_count": 8,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "df2.shape"  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "id": "9dbc9ac0",  
 "metadata": {},  
 "source": [  
 "As stated in the question, if for any column(s), the variance is equal to zero, then you need to remove those variable(s)."  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 9,  
 "id": "bacd0765",  
 "metadata": {},  
 "outputs": [  
 {  
 "name": "stdout",  
 "output\_type": "stream",  
 "text": [  
 "Name = X11\n",  
 "Name = X93\n",  
 "Name = X107\n",  
 "Name = X233\n",  
 "Name = X235\n",  
 "Name = X268\n",  
 "Name = X289\n",  
 "Name = X290\n",  
 "Name = X293\n",  
 "Name = X297\n",  
 "Name = X330\n",  
 "Name = X347\n",  
 "No of columns which has zero variance = 12\n"  
 ]  
 },  
 {  
 "name": "stderr",  
 "output\_type": "stream",  
 "text": [  
 "C:\\Users\\lrnem\\AppData\\Local\\Temp\\ipykernel\_4380\\1335638582.py:1: FutureWarning: The default value of numeric\_only in DataFrame.std is deprecated. In a future version, it will default to False. In addition, specifying 'numeric\_only=None' is deprecated. Select only valid columns or specify the value of numeric\_only to silence this warning.\n",  
 " variance = pow(df1.drop(columns={'ID','y'}).std(),2).to\_dict()\n"  
 ]  
 }  
 ],  
 "source": [  
 "variance = pow(df1.drop(columns={'ID','y'}).std(),2).to\_dict()\n",  
 "\n",  
 "variance\_is\_zero\_count = 0\n",  
 "for key, value in variance.items():\n",  
 " if(value==0):\n",  
 " print('Name = ',key)\n",  
 " variance\_is\_zero\_count = variance\_is\_zero\_count+1\n",  
 "print('No of columns which has zero variance = ', variance\_is\_zero\_count)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 10,  
 "id": "cefa1626",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "df1 = df1.drop(columns={'X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293', 'X297', 'X330', 'X347'})"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 11,  
 "id": "3974dc1a",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "(4209, 366)"  
 ]  
 },  
 "execution\_count": 11,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "df1.shape"  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "id": "e0bafa96",  
 "metadata": {},  
 "source": [  
 "Check for null and unique values for test and train sets."  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 12,  
 "id": "c7be2a9f",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "check1 = df1.isnull().sum()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 13,  
 "id": "8a40baa6",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "False"  
 ]  
 },  
 "execution\_count": 13,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "check1.any()"  
 ]  
 },  
 {

"cell\_type": "code",  
 "execution\_count": 14,  
 "id": "4d1daa4b",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "check2 = df2.isnull().sum()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 15,  
 "id": "d20cbe92",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "False"  
 ]  
 },  
 "execution\_count": 15,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "check2.any()"  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "id": "13a9fe08",  
 "metadata": {},  
 "source": [  
 "### Label encoding"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 16,  
 "id": "8a189be5",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "from sklearn.preprocessing import LabelEncoder\n",  
 "label\_encoder = LabelEncoder()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 17,  
 "id": "33d3bf7b",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "x1 = df1.drop(columns={'y', 'ID'}, axis=1)\n",  
 "y1 = df1['y']"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 18,  
 "id": "7b15d92d",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "(4209, 364)"  
 ]  
 },  
 "execution\_count": 18,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "x1.shape"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 19,  
 "id": "76016823",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "(4209,)"  
 ]  
 },  
 "execution\_count": 19,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "y1.shape"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 20,  
 "id": "d619c16a",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<div>\n",  
 "<style scoped>\n",  
 " .dataframe tbody tr th:only-of-type {\n",  
 " vertical-align: middle;\n",  
 " }\n",  
 "\n",  
 " .dataframe tbody tr th {\n",  
 " vertical-align: top;\n",  
 " }\n",  
 "\n",  
 " .dataframe thead th {\n",  
 " text-align: right;\n",  
 " }\n",  
 "</style>\n",  
 "<table border=\"1\" class=\"dataframe\">\n",  
 " <thead>\n",  
 " <tr style=\"text-align: right;\">\n",  
 " <th></th>\n",  
 " <th>X0</th>\n",  
 " <th>X1</th>\n",  
 " <th>X2</th>\n",  
 " <th>X3</th>\n",  
 " <th>X4</th>\n",  
 " <th>X5</th>\n",  
 " <th>X6</th>\n",  
 " <th>X8</th>\n",  
 " </tr>\n",  
 " </thead>\n",  
 " <tbody>\n",  
 " <tr>\n",  
 " <th>count</th>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>unique</th>\n",  
 " <td>47</td>\n",  
 " <td>27</td>\n",  
 " <td>44</td>\n",  
 " <td>7</td>\n",  
 " <td>4</td>\n",  
 " <td>29</td>\n",  
 " <td>12</td>\n",  
 " <td>25</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>top</th>\n",  
 " <td>z</td>\n",  
 " <td>aa</td>\n",  
 " <td>as</td>\n",  
 " <td>c</td>\n",  
 " <td>d</td>\n",  
 " <td>w</td>\n",  
 " <td>g</td>\n",  
 " <td>j</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>freq</th>\n",  
 " <td>360</td>\n",  
 " <td>833</td>\n",  
 " <td>1659</td>\n",  
 " <td>1942</td>\n",  
 " <td>4205</td>\n",  
 " <td>231</td>\n",  
 " <td>1042</td>\n",  
 " <td>277</td>\n",  
 " </tr>\n",  
 " </tbody>\n",  
 "</table>\n",  
 "</div>"  
 ],  
 "text/plain": [  
 " X0 X1 X2 X3 X4 X5 X6 X8\n",  
 "count 4209 4209 4209 4209 4209 4209 4209 4209\n",  
 "unique 47 27 44 7 4 29 12 25\n",  
 "top z aa as c d w g j\n",  
 "freq 360 833 1659 1942 4205 231 1042 277"  
 ]  
 },  
 "execution\_count": 20,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "x1.describe(include=object)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 21,  
 "id": "4ab6a64a",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "x1['X0'] = label\_encoder.fit\_transform(x1.X0)\n",  
 "x1['X1'] = label\_encoder.fit\_transform(x1.X1)\n",  
 "x1['X2'] = label\_encoder.fit\_transform(x1.X2)\n",  
 "x1['X3'] = label\_encoder.fit\_transform(x1.X3)\n",  
 "x1['X4'] = label\_encoder.fit\_transform(x1.X4)\n",  
 "x1['X5'] = label\_encoder.fit\_transform(x1.X5)\n",  
 "x1['X6'] = label\_encoder.fit\_transform(x1.X6)\n",  
 "x1['X8'] = label\_encoder.fit\_transform(x1.X8)"  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "id": "fc53f530",  
 "metadata": {},  
 "source": [  
 "### Dimensionality reduction"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 22,  
 "id": "686bc708",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "from sklearn.decomposition import PCA\n",  
 "pca = PCA(n\_components=0.90)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 23,  
 "id": "5c79a368",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<style>#sk-container-id-1 {color: black;background-color: white;}#sk-container-id-1 pre{padding: 0;}#sk-container-id-1 div.sk-toggleable {background-color: white;}#sk-container-id-1 label.sk-toggleable\_\_label {cursor: pointer;display: block;width: 100%;margin-bottom: 0;padding: 0.3em;box-sizing: border-box;text-align: center;}#sk-container-id-1 label.sk-toggleable\_\_label-arrow:before {content: \"▸\";float: left;margin-right: 0.25em;color: #696969;}#sk-container-id-1 label.sk-toggleable\_\_label-arrow:hover:before {color: black;}#sk-container-id-1 div.sk-estimator:hover label.sk-toggleable\_\_label-arrow:before {color: black;}#sk-container-id-1 div.sk-toggleable\_\_content {max-height: 0;max-width: 0;overflow: hidden;text-align: left;background-color: #f0f8ff;}#sk-container-id-1 div.sk-toggleable\_\_content pre {margin: 0.2em;color: black;border-radius: 0.25em;background-color: #f0f8ff;}#sk-container-id-1 input.sk-toggleable\_\_control:checked~div.sk-toggleable\_\_content {max-height: 200px;max-width: 100%;overflow: auto;}#sk-container-id-1 input.sk-toggleable\_\_control:checked~label.sk-toggleable\_\_label-arrow:before {content: \"▾\";}#sk-container-id-1 div.sk-estimator input.sk-toggleable\_\_control:checked~label.sk-toggleable\_\_label {background-color: #d4ebff;}#sk-container-id-1 div.sk-label input.sk-toggleable\_\_control:checked~label.sk-toggleable\_\_label {background-color: #d4ebff;}#sk-container-id-1 input.sk-hidden--visually {border: 0;clip: rect(1px 1px 1px 1px);clip: rect(1px, 1px, 1px, 1px);height: 1px;margin: -1px;overflow: hidden;padding: 0;position: absolute;width: 1px;}#sk-container-id-1 div.sk-estimator {font-family: monospace;background-color: #f0f8ff;border: 1px dotted black;border-radius: 0.25em;box-sizing: border-box;margin-bottom: 0.5em;}#sk-container-id-1 div.sk-estimator:hover {background-color: #d4ebff;}#sk-container-id-1 div.sk-parallel-item::after {content: \"\";width: 100%;border-bottom: 1px solid gray;flex-grow: 1;}#sk-container-id-1 div.sk-label:hover label.sk-toggleable\_\_label {background-color: #d4ebff;}#sk-container-id-1 div.sk-serial::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: 0;}#sk-container-id-1 div.sk-serial {display: flex;flex-direction: column;align-items: center;background-color: white;padding-right: 0.2em;padding-left: 0.2em;position: relative;}#sk-container-id-1 div.sk-item {position: relative;z-index: 1;}#sk-container-id-1 div.sk-parallel {display: flex;align-items: stretch;justify-content: center;background-color: white;position: relative;}#sk-container-id-1 div.sk-item::before, #sk-container-id-1 div.sk-parallel-item::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: -1;}#sk-container-id-1 div.sk-parallel-item {display: flex;flex-direction: column;z-index: 1;position: relative;background-color: white;}#sk-container-id-1 div.sk-parallel-item:first-child::after {align-self: flex-end;width: 50%;}#sk-container-id-1 div.sk-parallel-item:last-child::after {align-self: flex-start;width: 50%;}#sk-container-id-1 div.sk-parallel-item:only-child::after {width: 0;}#sk-container-id-1 div.sk-dashed-wrapped {border: 1px dashed gray;margin: 0 0.4em 0.5em 0.4em;box-sizing: border-box;padding-bottom: 0.4em;background-color: white;}#sk-container-id-1 div.sk-label label {font-family: monospace;font-weight: bold;display: inline-block;line-height: 1.2em;}#sk-container-id-1 div.sk-label-container {text-align: center;}#sk-container-id-1 div.sk-container {/\* jupyter's `normalize.less` sets `[hidden] { display: none; }` but bootstrap.min.css set `[hidden] { display: none !important; }` so we also need the `!important` here to be able to override the default hidden behavior on the sphinx rendered scikit-learn.org. See: https://github.com/scikit-learn/scikit-learn/issues/21755 \*/display: inline-block !important;position: relative;}#sk-container-id-1 div.sk-text-repr-fallback {display: none;}</style><div id=\"sk-container-id-1\" class=\"sk-top-container\"><div class=\"sk-text-repr-fallback\"><pre>PCA(n\_components=0.9)</pre><b>In a Jupyter environment, please rerun this cell to show the

HTML representation or trust the notebook. <br />On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.</b></div><div class=\"sk-container\" hidden><div class=\"sk-item\"><div class=\"sk-estimator sk-toggleable\"><input class=\"sk-toggleable\_\_control sk-hidden--visually\" id=\"sk-estimator-id-1\" type=\"checkbox\" checked><label for=\"sk-estimator-id-1\" class=\"sk-toggleable\_\_label sk-toggleable\_\_label-arrow\">PCA</label><div class=\"sk-toggleable\_\_content\"><pre>PCA(n\_components=0.9)</pre></div></div></div></div></div>"  
 ],  
 "text/plain": [  
 "PCA(n\_components=0.9)"  
 ]  
 },  
 "execution\_count": 23,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "pca.fit(x1, y1)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 24,  
 "id": "4de5d871",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "(4209, 5)"  
 ]  
 },  
 "execution\_count": 24,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "x1\_transformed = pca.fit\_transform(x1)\n",  
 "x1\_transformed.shape"  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "id": "387fd182",  
 "metadata": {},  
 "source": [  
 "Predict your test\_df values using XGBoost"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 25,  
 "id": "9818bc4c",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "from math import sqrt\n",  
 "from sklearn.model\_selection import train\_test\_split\n",  
 "from sklearn.metrics import r2\_score, mean\_squared\_error\n",  
 "import xgboost as xgb"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 26,  
 "id": "8e5766eb",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "x\_train, x\_test, y\_train, y\_test = train\_test\_split(x1\_transformed, y1, test\_size=.1, random\_state=3)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 27,  
 "id": "2b7cd9dd",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "((3788, 5), (421, 5))"  
 ]  
 },  
 "execution\_count": 27,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "x\_train.shape, x\_test.shape"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 28,  
 "id": "a43f3bd8",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "((3788,), (421,))"  
 ]  
 },  
 "execution\_count": 28,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "y\_train.shape, y\_test.shape"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 29,  
 "id": "5097f285",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "dtrain\_reg = xgb.DMatrix(x\_train, y\_train, enable\_categorical=True)\n",  
 "dtest\_reg = xgb.DMatrix(x\_test, y\_test, enable\_categorical=True)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 30,  
 "id": "3a51ab75",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "params = {\"objective\": \"reg:squarederror\", \"tree\_method\": \"gpu\_hist\"}"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 31,  
 "id": "5610c25a",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "params = {\"objective\": \"reg:squarederror\", \"tree\_method\": \"gpu\_hist\"}\n",  
 "\n",  
 "n = 300\n",  
 "xgb1 = xgb.train(params=params, dtrain=dtrain\_reg, num\_boost\_round=n,)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 32,  
 "id": "224aa05a",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "from sklearn.metrics import mean\_squared\_error\n",  
 "\n",  
 "pred = xgb1.predict(dtest\_reg)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 33,  
 "id": "d4109cfb",  
 "metadata": {},  
 "outputs": [  
 {  
 "name": "stdout",  
 "output\_type": "stream",  
 "text": [  
 "Base model RMSE val : 10.211\n"  
 ]  
 }  
 ],  
 "source": [  
 "rmse = mean\_squared\_error(y\_test, pred, squared=False)\n",  
 "\n",  
 "print(f\"Base model RMSE val : {rmse:.3f}\")"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 34,  
 "id": "9b1b1dd1",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "params = {\"objective\": \"reg:squarederror\", \"tree\_method\": \"gpu\_hist\"}\n",  
 "n = 1000\n",  
 "results = xgb.cv(params, dtrain\_reg, num\_boost\_round=n, nfold=5, early\_stopping\_rounds=20)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 35,  
 "id": "40f6c1a6",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<div>\n",  
 "<style scoped>\n",  
 " .dataframe tbody tr th:only-of-type {\n",  
 " vertical-align: middle;\n",  
 " }\n",  
 "\n",  
 " .dataframe tbody tr th {\n",  
 " vertical-align: top;\n",  
 " }\n",  
 "\n",  
 " .dataframe thead th {\n",  
 " text-align: right;\n",  
 " }\n",  
 "</style>\n",  
 "<table border=\"1\" class=\"dataframe\">\n",  
 " <thead>\n",  
 " <tr style=\"text-align: right;\">\n",  
 " <th></th>\n",  
 " <th>train-rmse-mean</th>\n",  
 " <th>train-rmse-std</th>\n",  
 " <th>test-rmse-mean</th>\n",  
 " <th>test-rmse-std</th>\n",  
 " </tr>\n",  
 " </thead>\n",  
 " <tbody>\n",  
 " <tr>\n",  
 " <th>0</th>\n",  
 " <td>71.148586</td>\n",  
 " <td>0.139979</td>\n",  
 " <td>71.156159</td>\n",  
 " <td>0.752683</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>1</th>\n",  
 " <td>50.469313</td>\n",  
 " <td>0.122809</td>\n",  
 " <td>50.496843</td>\n",  
 " <td>0.822578</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>2</th>\n",  
 " <td>36.170590</td>\n",  
 " <td>0.123311</td>\n",  
 " <td>36.214890</td>\n",  
 " <td>0.906990</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>3</th>\n",  
 " <td>26.413531</td>\n",  
 " <td>0.136587</td>\n",  
 " <td>26.525787</td>\n",  
 " <td>0.959477</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>4</th>\n",  
 " <td>19.851765</td>\n",  
 " <td>0.161268</td>\n",  
 " <td>20.093561</td>\n",  
 " <td>1.039389</td>\n",  
 " </tr>\n",  
 " </tbody>\n",  
 "</table>\n",  
 "</div>"  
 ],  
 "text/plain": [  
 " train-rmse-mean train-rmse-std test-rmse-mean test-rmse-std\n",  
 "0 71.148586 0.139979 71.156159 0.752683\n",  
 "1 50.469313 0.122809 50.496843 0.822578\n",  
 "2 36.170590 0.123311 36.214890 0.906990\n",  
 "3 26.413531 0.136587 26.525787 0.959477\n",  
 "4 19.851765 0.161268 20.093561 1.039389"  
 ]  
 },  
 "execution\_count": 35,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "results.head()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 36,  
 "id": "189df95b",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "best\_rmse = results['test-rmse-mean'].min()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 37,  
 "id": "bdbbdad9",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "10.195234054808457"  
 ]  
 },  
 "execution\_count": 37,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "best\_rmse"  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "id": "1705cde4",  
 "metadata": {},  
 "source": [  
 "Using k-fold cross validation, RMSE comes as 10.195."  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "id": "abca32e7",  
 "metadata": {},  
 "source": [  
 "### Testing dataset"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 38,  
 "id": "0d951768",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "df2 = df2.drop(columns={'X11','X93','X107','X233','X235','X268','X289','X290','X293','X297','X330','X347'})"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 39,  
 "id": "df198ece",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "(4209, 365)"  
 ]  
 },  
 "execution\_count": 39,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "df2.shape"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 40,  
 "id": "e22c8b8c",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "check3 = df2.isnull().sum()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 41,  
 "id": "127d9155",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "False"  
 ]  
 },  
 "execution\_count": 41,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "check3.any()"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 42,  
 "id": "25432b04",  
 "metadata": {},  
 "outputs": [  
 {  
 "name": "stdout",  
 "output\_type": "stream",  
 "text": [  
 "(4209, 364)\n"  
 ]  
 }  
 ],  
 "source": [  
 "x2 = df2.drop(columns={'ID'})\n",  
 "print(x2.shape)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 43,  
 "id": "1583e6ac",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<div>\n",  
 "<style scoped>\n",  
 " .dataframe tbody tr th:only-of-type {\n",  
 " vertical-align: middle;\n",  
 " }\n",  
 "\n",  
 " .dataframe tbody tr th {\n",  
 " vertical-align: top;\n",  
 " }\n",  
 "\n",  
 " .dataframe thead th {\n",  
 " text-align: right;\n",  
 " }\n",  
 "</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",  
 " <thead>\n",  
 " <tr style=\"text-align: right;\">\n",  
 " <th></th>\n",  
 " <th>X0</th>\n",  
 " <th>X1</th>\n",  
 " <th>X2</th>\n",  
 " <th>X3</th>\n",  
 " <th>X4</th>\n",  
 " <th>X5</th>\n",  
 " <th>X6</th>\n",  
 " <th>X8</th>\n",  
 " </tr>\n",  
 " </thead>\n",  
 " <tbody>\n",  
 " <tr>\n",  
 " <th>count</th>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " <td>4209</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>unique</th>\n",  
 " <td>49</td>\n",  
 " <td>27</td>\n",  
 " <td>45</td>\n",  
 " <td>7</td>\n",  
 " <td>4</td>\n",  
 " <td>32</td>\n",  
 " <td>12</td>\n",  
 " <td>25</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>top</th>\n",  
 " <td>ak</td>\n",  
 " <td>aa</td>\n",  
 " <td>as</td>\n",  
 " <td>c</td>\n",  
 " <td>d</td>\n",  
 " <td>v</td>\n",  
 " <td>g</td>\n",  
 " <td>e</td>\n",  
 " </tr>\n",  
 " <tr>\n",  
 " <th>freq</th>\n",  
 " <td>432</td>\n",  
 " <td>826</td>\n",  
 " <td>1658</td>\n",  
 " <td>1900</td>\n",  
 " <td>4203</td>\n",  
 " <td>246</td>\n",  
 " <td>1073</td>\n",  
 " <td>274</td>\n",  
 " </tr>\n",  
 " </tbody>\n",  
 "</table>\n",  
 "</div>"  
 ],  
 "text/plain": [  
 " X0 X1 X2 X3 X4 X5 X6 X8\n",  
 "count 4209 4209 4209 4209 4209 4209 4209 4209\n",  
 "unique 49 27 45 7 4 32 12 25\n",  
 "top ak aa as c d v g e\n",  
 "freq 432 826 1658 1900 4203 246 1073 274"  
 ]  
 },  
 "execution\_count": 43,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "x2.describe(include='object')"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 44,  
 "id": "c2aab59e",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "x2['X0'] = label\_encoder.fit\_transform(x2.X0)\n",  
 "x2['X1'] = label\_encoder.fit\_transform(x2.X1)\n",  
 "x2['X2'] = label\_encoder.fit\_transform(x2.X2)\n",  
 "x2['X3'] = label\_encoder.fit\_transform(x2.X3)\n",  
 "x2['X4'] = label\_encoder.fit\_transform(x2.X4)\n",  
 "x2['X5'] = label\_encoder.fit\_transform(x2.X5)\n",  
 "x2['X6'] = label\_encoder.fit\_transform(x2.X6)\n",  
 "x2['X8'] = label\_encoder.fit\_transform(x2.X8)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 45,  
 "id": "410c934d",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/html": [  
 "<style>#sk-container-id-2 {color: black;background-color: white;}#sk-container-id-2 pre{padding: 0;}#sk-container-id-2 div.sk-toggleable {background-color: white;}#sk-container-id-2 label.sk-toggleable\_\_label {cursor: pointer;display: block;width: 100%;margin-bottom: 0;padding: 0.3em;box-sizing: border-box;text-align: center;}#sk-container-id-2 label.sk-toggleable\_\_label-arrow:before {content: \"▸\";float: left;margin-right: 0.25em;color: #696969;}#sk-container-id-2 label.sk-toggleable\_\_label-arrow:hover:before {color: black;}#sk-container-id-2 div.sk-estimator:hover label.sk-toggleable\_\_label-arrow:before {color: black;}#sk-container-id-2 div.sk-toggleable\_\_content {max-height: 0;max-width: 0;overflow: hidden;text-align: left;background-color: #f0f8ff;}#sk-container-id-2 div.sk-toggleable\_\_content pre {margin: 0.2em;color: black;border-radius: 0.25em;background-color: #f0f8ff;}#sk-container-id-2 input.sk-toggleable\_\_control:checked~div.sk-toggleable\_\_content {max-height: 200px;max-width: 100%;overflow: auto;}#sk-container-id-2 input.sk-toggleable\_\_control:checked~label.sk-toggleable\_\_label-arrow:before {content: \"▾\";}#sk-container-id-2 div.sk-estimator input.sk-toggleable\_\_control:checked~label.sk-toggleable\_\_label {background-color: #d4ebff;}#sk-container-id-2 div.sk-label input.sk-toggleable\_\_control:checked~label.sk-toggleable\_\_label {background-color: #d4ebff;}#sk-container-id-2 input.sk-hidden--visually {border: 0;clip: rect(1px 1px 1px 1px);clip: rect(1px, 1px, 1px, 1px);height: 1px;margin: -1px;overflow: hidden;padding: 0;position: absolute;width: 1px;}#sk-container-id-2 div.sk-estimator {font-family: monospace;background-color: #f0f8ff;border: 1px dotted black;border-radius: 0.25em;box-sizing: border-box;margin-bottom: 0.5em;}#sk-container-id-2 div.sk-estimator:hover {background-color: #d4ebff;}#sk-container-id-2 div.sk-parallel-item::after {content: \"\";width: 100%;border-bottom: 1px solid gray;flex-grow: 1;}#sk-container-id-2 div.sk-label:hover label.sk-toggleable\_\_label {background-color: #d4ebff;}#sk-container-id-2 div.sk-serial::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: 0;}#sk-container-id-2 div.sk-serial {display: flex;flex-direction: column;align-items: center;background-color: white;padding-right: 0.2em;padding-left: 0.2em;position: relative;}#sk-container-id-2 div.sk-item {position: relative;z-index: 1;}#sk-container-id-2 div.sk-parallel {display: flex;align-items: stretch;justify-content: center;background-color: white;position: relative;}#sk-container-id-2 div.sk-item::before, #sk-container-id-2 div.sk-parallel-item::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: -1;}#sk-container-id-2 div.sk-parallel-item {display: flex;flex-direction: column;z-index: 1;position: relative;background-color: white;}#sk-container-id-2 div.sk-parallel-item:first-child::after {align-self: flex-end;width: 50%;}#sk-container-id-2 div.sk-parallel-item:last-child::after {align-self: flex-start;width: 50%;}#sk-container-id-2 div.sk-parallel-item:only-child::after {width: 0;}#sk-container-id-2 div.sk-dashed-wrapped {border: 1px dashed gray;margin: 0 0.4em 0.5em 0.4em;box-sizing: border-box;padding-bottom: 0.4em;background-color: white;}#sk-container-id-2 div.sk-label label {font-family: monospace;font-weight: bold;display: inline-block;line-height: 1.2em;}#sk-container-id-2 div.sk-label-container {text-align: center;}#sk-container-id-2 div.sk-container {/\* jupyter's `normalize.less` sets `[hidden] { display: none; }` but bootstrap.min.css set `[hidden] { display: none !important; }` so we also need the `!important` here to be able to override the default hidden behavior on the sphinx rendered scikit-learn.org. See: https://github.com/scikit-learn/scikit-learn/issues/21755 \*/display: inline-block !important;position: relative;}#sk-container-id-2 div.sk-text-repr-fallback {display: none;}</style><div id=\"sk-container-id-2\" class=\"sk-top-container\"><div class=\"sk-text-repr-fallback\"><pre>PCA(n\_components=0.9)</pre><b>In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. <br />On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.</b></div><div class=\"sk-container\" hidden><div class=\"sk-item\"><div class=\"sk-estimator sk-toggleable\"><input class=\"sk-toggleable\_\_control sk-hidden--visually\" id=\"sk-estimator-id-2\" type=\"checkbox\" checked><label for=\"sk-estimator-id-2\" class=\"sk-toggleable\_\_label sk-toggleable\_\_label-arrow\">PCA</label><div class=\"sk-toggleable\_\_content\"><pre>PCA(n\_components=0.9)</pre></div></div></div></div></div>"  
 ],  
 "text/plain": [  
 "PCA(n\_components=0.9)"  
 ]  
 },  
 "execution\_count": 45,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "pca.fit(x2)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 46,  
 "id": "a804cd61",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "(4209, 5)"  
 ]  
 },  
 "execution\_count": 46,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "x2\_transformed = pca.fit\_transform(x2)\n",  
 "x2\_transformed.shape"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 47,  
 "id": "4525be65",  
 "metadata": {},  
 "outputs": [],  
 "source": [  
 "dtest = xgb.DMatrix(x2\_transformed)"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": 48,  
 "id": "41d203a4",  
 "metadata": {},  
 "outputs": [  
 {  
 "data": {  
 "text/plain": [  
 "array([ 73.144936, 95.4681 , 95.36935 , ..., 91.54465 , 108.6302 ,\n",  
 " 96.124054], dtype=float32)"  
 ]  
 },  
 "execution\_count": 48,  
 "metadata": {},  
 "output\_type": "execute\_result"  
 }  
 ],  
 "source": [  
 "test\_pred = xgb1.predict(dtest)\n",  
 "test\_pred"  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "execution\_count": null,  
 "id": "c7bd3734",  
 "metadata": {},  
 "outputs": [],  
 "source": []  
 }  
 ],  
 "metadata": {  
 "kernelspec": {  
 "display\_name": "Python 3 (ipykernel)",  
 "language": "python",  
 "name": "python3"  
 },  
 "language\_info": {  
 "codemirror\_mode": {  
 "name": "ipython",  
 "version": 3  
 },  
 "file\_extension": ".py",  
 "mimetype": "text/x-python",  
 "name": "python",  
 "nbconvert\_exporter": "python",  
 "pygments\_lexer": "ipython3",  
 "version": "3.11.3"  
 }  
 },  
 "nbformat": 4,  
 "nbformat\_minor": 5  
}