|  |
| --- |
| **Project : Mercedes-Benz Greener Manufacturing** |

**By: Prashanth Teja Jamallamudi**

**DESCRIPTION:**

Reduce the time a Mercedes-Benz spends on the test bench.

Problem Statement Scenario:  
Since the first automobile, the Benz Patent Motor Car in 1886, Mercedes-Benz has stood for important automotive innovations. These include the passenger safety cell with a crumple zone, the airbag, and intelligent assistance systems. Mercedes-Benz applies for nearly 2000 patents per year, making the brand the European leader among premium carmakers. Mercedes-Benz is the leader in the premium car industry. With a huge selection of features and options, customers can choose the customized Mercedes-Benz of their dreams.

To ensure the safety and reliability of every unique car configuration before they hit the road, the company’s engineers have developed a robust testing system. As one of the world’s biggest manufacturers of premium cars, safety and efficiency are paramount on Mercedes-Benz’s production lines. However, optimizing the speed of their testing system for many possible feature combinations is complex and time-consuming without a powerful algorithmic approach.

You are required to reduce the time that cars spend on the test bench. Others will work with a dataset representing different permutations of features in a Mercedes-Benz car to predict the time it takes to pass testing. Optimal algorithms will contribute to faster testing, resulting in lower carbon dioxide emissions without reducing Mercedes-Benz’s standards.

Following actions should be performed:

* If for any column(s), the variance is equal to zero, then you need to remove those variable(s).
* Check for null and unique values for test and train sets.
* Apply label encoder.
* Perform dimensionality reduction.
* Predict your test\_df values using XGBoost.

[{"metadata":{"trusted":true},"cell\_type":"code","source":"import pandas as pd\nimport numpy as np\nimport matplotlib.pyplot as plt\nfrom matplotlib import style\nimport seaborn as sns\n%matplotlib inline","execution\_count":1352,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import warnings\nwarnings.filterwarnings('ignore')","execution\_count":1353,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train =pd.read\_csv(\"D:\\\\Simplilearn\\\\ML\\\\Benz\\\\train.csv\")","execution\_count":1354,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train.head()","execution\_count":1355,"outputs":[{"output\_type":"execute\_result","execution\_count":1355,"data":{"text/plain":" ID y X0 X1 X2 X3 X4 X5 X6 X8 ... X375 X376 X377 X378 X379 \\\n0 0 130.81 k v at a d u j o ... 0 0 1 0 0 \n1 6 88.53 k t av e d y l o ... 1 0 0 0 0 \n2 7 76.26 az w n c d x j x ... 0 0 0 0 0 \n3 9 80.62 az t n f d x l e ... 0 0 0 0 0 \n4 13 78.02 az v n f d h d n ... 0 0 0 0 0 \n\n X380 X382 X383 X384 X385 \n0 0 0 0 0 0 \n1 0 0 0 0 0 \n2 0 1 0 0 0 \n3 0 0 0 0 0 \n4 0 0 0 0 0 \n\n[5 rows x 378 columns]","text/html":"

\n\n \n \n ID\n y\n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n 0\n 0\n 130.81\n k\n v\n at\n a\n d\n u\n j\n o\n ...\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 1\n 6\n 88.53\n k\n t\n av\n e\n d\n y\n l\n o\n ...\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 2\n 7\n 76.26\n az\n w\n n\n c\n d\n x\n j\n x\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n \n \n 3\n 9\n 80.62\n az\n t\n n\n f\n d\n x\n l\n e\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 4\n 13\n 78.02\n az\n v\n n\n f\n d\n h\n d\n n\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n\n

5 rows × 378 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train.dtypes.value\_counts()","execution\_count":1356,"outputs":[{"output\_type":"execute\_result","execution\_count":1356,"data":{"text/plain":"int64 369\nobject 8\nfloat64 1\ndtype: int64"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train.head()","execution\_count":1359,"outputs":[{"output\_type":"execute\_result","execution\_count":1359,"data":{"text/plain":" ID y X0 X1 X2 X3 X4 X5 X6 X8 ... X375 X376 X377 X378 X379 \\\n0 0 130.81 k v at a d u j o ... 0 0 1 0 0 \n1 6 88.53 k t av e d y l o ... 1 0 0 0 0 \n2 7 76.26 az w n c d x j x ... 0 0 0 0 0 \n3 9 80.62 az t n f d x l e ... 0 0 0 0 0 \n4 13 78.02 az v n f d h d n ... 0 0 0 0 0 \n\n X380 X382 X383 X384 X385 \n0 0 0 0 0 0 \n1 0 0 0 0 0 \n2 0 1 0 0 0 \n3 0 0 0 0 0 \n4 0 0 0 0 0 \n\n[5 rows x 378 columns]","text/html":"

\n\n \n \n ID\n y\n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n 0\n 0\n 130.81\n k\n v\n at\n a\n d\n u\n j\n o\n ...\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 1\n 6\n 88.53\n k\n t\n av\n e\n d\n y\n l\n o\n ...\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 2\n 7\n 76.26\n az\n w\n n\n c\n d\n x\n j\n x\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n \n \n 3\n 9\n 80.62\n az\n t\n n\n f\n d\n x\n l\n e\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 4\n 13\n 78.02\n az\n v\n n\n f\n d\n h\n d\n n\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n\n

5 rows × 378 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"#benz\_train.ID.shape","execution\_count":1360,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"#seperating catageory and numerical data into different data frames","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_cat = benz\_train.select\_dtypes(exclude=[\"float64\",\"int64\"])\nbenz\_train\_cat.head()","execution\_count":1361,"outputs":[{"output\_type":"execute\_result","execution\_count":1361,"data":{"text/plain":" X0 X1 X2 X3 X4 X5 X6 X8\n0 k v at a d u j o\n1 k t av e d y l o\n2 az w n c d x j x\n3 az t n f d x l e\n4 az v n f d h d n","text/html":"

\n\n \n \n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n k\n v\n at\n a\n d\n u\n j\n o\n \n \n 1\n k\n t\n av\n e\n d\n y\n l\n o\n \n \n 2\n az\n w\n n\n c\n d\n x\n j\n x\n \n \n 3\n az\n t\n n\n f\n d\n x\n l\n e\n \n \n 4\n az\n v\n n\n f\n d\n h\n d\n n\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int = benz\_train.select\_dtypes(include=[\"float64\",\"int64\"])\nbenz\_train\_int.head()","execution\_count":1362,"outputs":[{"output\_type":"execute\_result","execution\_count":1362,"data":{"text/plain":" ID y X10 X11 X12 X13 X14 X15 X16 X17 ... X375 X376 X377 \\\n0 0 130.81 0 0 0 1 0 0 0 0 ... 0 0 1 \n1 6 88.53 0 0 0 0 0 0 0 0 ... 1 0 0 \n2 7 76.26 0 0 0 0 0 0 0 1 ... 0 0 0 \n3 9 80.62 0 0 0 0 0 0 0 0 ... 0 0 0 \n4 13 78.02 0 0 0 0 0 0 0 0 ... 0 0 0 \n\n X378 X379 X380 X382 X383 X384 X385 \n0 0 0 0 0 0 0 0 \n1 0 0 0 0 0 0 0 \n2 0 0 0 1 0 0 0 \n3 0 0 0 0 0 0 0 \n4 0 0 0 0 0 0 0 \n\n[5 rows x 370 columns]","text/html":"

\n\n \n \n ID\n y\n X10\n X11\n X12\n X13\n X14\n X15\n X16\n X17\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n 0\n 0\n 130.81\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 1\n 6\n 88.53\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 2\n 7\n 76.26\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n \n \n 3\n 9\n 80.62\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 4\n 13\n 78.02\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n\n

5 rows × 370 columns

\n

"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# If for any column(s), the variance is equal to zero, then you need to remove those variable(s)."},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int\_var = pd.DataFrame(benz\_train\_int.var(axis=0),columns=['Variance'])","execution\_count":1363,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int\_var.head()","execution\_count":1364,"outputs":[{"output\_type":"execute\_result","execution\_count":1364,"data":{"text/plain":" Variance\nID 5.941936e+06\ny 1.607667e+02\nX10 1.313092e-02\nX11 0.000000e+00\nX12 6.945713e-02","text/html":"

\n\n \n \n Variance\n \n \n \n \n ID\n 5.941936e+06\n \n \n y\n 1.607667e+02\n \n \n X10\n 1.313092e-02\n \n \n X11\n 0.000000e+00\n \n \n X12\n 6.945713e-02\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"#Dropping the columns with variance is == 0\nbenz\_train\_int\_var\_dropped = benz\_train\_int.drop(columns=benz\_train\_int\_var[benz\_train\_int\_var.Variance==0].index)","execution\_count":1365,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int\_var[benz\_train\_int\_var.Variance==0] # 12 columns with variance ==0","execution\_count":1366,"outputs":[{"output\_type":"execute\_result","execution\_count":1366,"data":{"text/plain":" Variance\nX11 0.0\nX93 0.0\nX107 0.0\nX233 0.0\nX235 0.0\nX268 0.0\nX289 0.0\nX290 0.0\nX293 0.0\nX297 0.0\nX330 0.0\nX347 0.0","text/html":"

\n\n \n \n Variance\n \n \n \n \n X11\n 0.0\n \n \n X93\n 0.0\n \n \n X107\n 0.0\n \n \n X233\n 0.0\n \n \n X235\n 0.0\n \n \n X268\n 0.0\n \n \n X289\n 0.0\n \n \n X290\n 0.0\n \n \n X293\n 0.0\n \n \n X297\n 0.0\n \n \n X330\n 0.0\n \n \n X347\n 0.0\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int\_var\_dropped.head()","execution\_count":1367,"outputs":[{"output\_type":"execute\_result","execution\_count":1367,"data":{"text/plain":" ID y X10 X12 X13 X14 X15 X16 X17 X18 ... X375 X376 X377 \\\n0 0 130.81 0 0 1 0 0 0 0 1 ... 0 0 1 \n1 6 88.53 0 0 0 0 0 0 0 1 ... 1 0 0 \n2 7 76.26 0 0 0 0 0 0 1 0 ... 0 0 0 \n3 9 80.62 0 0 0 0 0 0 0 0 ... 0 0 0 \n4 13 78.02 0 0 0 0 0 0 0 0 ... 0 0 0 \n\n X378 X379 X380 X382 X383 X384 X385 \n0 0 0 0 0 0 0 0 \n1 0 0 0 0 0 0 0 \n2 0 0 0 1 0 0 0 \n3 0 0 0 0 0 0 0 \n4 0 0 0 0 0 0 0 \n\n[5 rows x 358 columns]","text/html":"

\n\n \n \n ID\n y\n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n 0\n 0\n 130.81\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 1\n ...\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 1\n 6\n 88.53\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n ...\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 2\n 7\n 76.26\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n \n \n 3\n 9\n 80.62\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 4\n 13\n 78.02\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n\n

5 rows × 358 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int\_var\_dropped.describe() #just to check the distribution of the Y variable","execution\_count":1368,"outputs":[{"output\_type":"execute\_result","execution\_count":1368,"data":{"text/plain":" ID y X10 X12 X13 \\\ncount 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 \nmean 4205.960798 100.669318 0.013305 0.075077 0.057971 \nstd 2437.608688 12.679381 0.114590 0.263547 0.233716 \nmin 0.000000 72.110000 0.000000 0.000000 0.000000 \n25% 2095.000000 90.820000 0.000000 0.000000 0.000000 \n50% 4220.000000 99.150000 0.000000 0.000000 0.000000 \n75% 6314.000000 109.010000 0.000000 0.000000 0.000000 \nmax 8417.000000 265.320000 1.000000 1.000000 1.000000 \n\n X14 X15 X16 X17 X18 ... \\\ncount 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 ... \nmean 0.428130 0.000475 0.002613 0.007603 0.007840 ... \nstd 0.494867 0.021796 0.051061 0.086872 0.088208 ... \nmin 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n25% 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n50% 0.000000 0.000000 0.000000 0.000000 0.000000 ... \n75% 1.000000 0.000000 0.000000 0.000000 0.000000 ... \nmax 1.000000 1.000000 1.000000 1.000000 1.000000 ... \n\n X375 X376 X377 X378 X379 \\\ncount 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 \nmean 0.318841 0.057258 0.314802 0.020670 0.009503 \nstd 0.466082 0.232363 0.464492 0.142294 0.097033 \nmin 0.000000 0.000000 0.000000 0.000000 0.000000 \n25% 0.000000 0.000000 0.000000 0.000000 0.000000 \n50% 0.000000 0.000000 0.000000 0.000000 0.000000 \n75% 1.000000 0.000000 1.000000 0.000000 0.000000 \nmax 1.000000 1.000000 1.000000 1.000000 1.000000 \n\n X380 X382 X383 X384 X385 \ncount 4209.000000 4209.000000 4209.000000 4209.000000 4209.000000 \nmean 0.008078 0.007603 0.001663 0.000475 0.001426 \nstd 0.089524 0.086872 0.040752 0.021796 0.037734 \nmin 0.000000 0.000000 0.000000 0.000000 0.000000 \n25% 0.000000 0.000000 0.000000 0.000000 0.000000 \n50% 0.000000 0.000000 0.000000 0.000000 0.000000 \n75% 0.000000 0.000000 0.000000 0.000000 0.000000 \nmax 1.000000 1.000000 1.000000 1.000000 1.000000 \n\n[8 rows x 358 columns]","text/html":"

\n\n \n \n ID\n y\n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n count\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n ...\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n 4209.000000\n \n \n mean\n 4205.960798\n 100.669318\n 0.013305\n 0.075077\n 0.057971\n 0.428130\n 0.000475\n 0.002613\n 0.007603\n 0.007840\n ...\n 0.318841\n 0.057258\n 0.314802\n 0.020670\n 0.009503\n 0.008078\n 0.007603\n 0.001663\n 0.000475\n 0.001426\n \n \n std\n 2437.608688\n 12.679381\n 0.114590\n 0.263547\n 0.233716\n 0.494867\n 0.021796\n 0.051061\n 0.086872\n 0.088208\n ...\n 0.466082\n 0.232363\n 0.464492\n 0.142294\n 0.097033\n 0.089524\n 0.086872\n 0.040752\n 0.021796\n 0.037734\n \n \n min\n 0.000000\n 72.110000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n ...\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n \n \n 25%\n 2095.000000\n 90.820000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n ...\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n \n \n 50%\n 4220.000000\n 99.150000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n ...\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n \n \n 75%\n 6314.000000\n 109.010000\n 0.000000\n 0.000000\n 0.000000\n 1.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n ...\n 1.000000\n 0.000000\n 1.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n 0.000000\n \n \n max\n 8417.000000\n 265.320000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n ...\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n 1.000000\n \n \n\n

8 rows × 358 columns

\n

"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# Checking whether the target variable Y is normally distributed, Outlier treatment"},{"metadata":{"trusted":true},"cell\_type":"code","source":"sns.distplot(benz\_train\_int\_var\_dropped[\"y\"])","execution\_count":1369,"outputs":[{"output\_type":"execute\_result","execution\_count":1369,"data":{"text/plain":""},"metadata":{}},{"output\_type":"display\_data","data":{"text/plain":"","image/png":"\n"},"metadata":{"needs\_background":"light"}}]},{"metadata":{},"cell\_type":"markdown","source":"# yeah!! the data is normally distributed since mean(100.669318) and median(99.150000) are closer"},{"metadata":{"trusted":true},"cell\_type":"code","source":"sns.boxplot(benz\_train\_int\_var\_dropped[\"y\"]) #check if the y variable has any outliers","execution\_count":1370,"outputs":[{"output\_type":"execute\_result","execution\_count":1370,"data":{"text/plain":""},"metadata":{}},{"output\_type":"display\_data","data":{"text/plain":"","image/png":"\n"},"metadata":{"needs\_background":"light"}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(benz\_train\_int\_var\_dropped[\"y\"].quantile(0.50))","execution\_count":1371,"outputs":[{"output\_type":"stream","text":"99.15\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(benz\_train\_int\_var\_dropped[\"y\"].quantile(0.95)) ","execution\_count":1372,"outputs":[{"output\_type":"stream","text":"120.806\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int\_var\_dropped[\"y\"] = np.where(benz\_train\_int\_var\_dropped[\"y\"] > 120.80600000000001, 99.15, benz\_train\_int\_var\_dropped[\"y\"])","execution\_count":1373,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"train\_y = pd.DataFrame(benz\_train\_int\_var\_dropped[\"y\"]) #seperating Y variable ","execution\_count":1374,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"train\_y.head()","execution\_count":1375,"outputs":[{"output\_type":"execute\_result","execution\_count":1375,"data":{"text/plain":" y\n0 99.15\n1 88.53\n2 76.26\n3 80.62\n4 78.02","text/html":"

\n\n \n \n y\n \n \n \n \n 0\n 99.15\n \n \n 1\n 88.53\n \n \n 2\n 76.26\n \n \n 3\n 80.62\n \n \n 4\n 78.02\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int\_var\_dropped\_y = benz\_train\_int\_var\_dropped.drop([\"y\"], axis=1)","execution\_count":1376,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_int\_var\_dropped\_y.head()","execution\_count":1377,"outputs":[{"output\_type":"execute\_result","execution\_count":1377,"data":{"text/plain":" ID X10 X12 X13 X14 X15 X16 X17 X18 X19 ... X375 X376 X377 \\\n0 0 0 0 1 0 0 0 0 1 0 ... 0 0 1 \n1 6 0 0 0 0 0 0 0 1 0 ... 1 0 0 \n2 7 0 0 0 0 0 0 1 0 0 ... 0 0 0 \n3 9 0 0 0 0 0 0 0 0 0 ... 0 0 0 \n4 13 0 0 0 0 0 0 0 0 0 ... 0 0 0 \n\n X378 X379 X380 X382 X383 X384 X385 \n0 0 0 0 0 0 0 0 \n1 0 0 0 0 0 0 0 \n2 0 0 0 1 0 0 0 \n3 0 0 0 0 0 0 0 \n4 0 0 0 0 0 0 0 \n\n[5 rows x 357 columns]","text/html":"

\n\n \n \n ID\n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n X19\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 1\n 0\n ...\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 1\n 6\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n ...\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 2\n 7\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n \n \n 3\n 9\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 4\n 13\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n\n

5 rows × 357 columns

\n

"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# Apply label encoder."},{"metadata":{"trusted":true},"cell\_type":"code","source":"#Lable encoder for catagorical data\nfrom sklearn import preprocessing\nlabel\_encoder = preprocessing.LabelEncoder()","execution\_count":1382,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_cat.head()","execution\_count":1383,"outputs":[{"output\_type":"execute\_result","execution\_count":1383,"data":{"text/plain":" X0 X1 X2 X3 X4 X5 X6 X8\n0 k v at a d u j o\n1 k t av e d y l o\n2 az w n c d x j x\n3 az t n f d x l e\n4 az v n f d h d n","text/html":"

\n\n \n \n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n k\n v\n at\n a\n d\n u\n j\n o\n \n \n 1\n k\n t\n av\n e\n d\n y\n l\n o\n \n \n 2\n az\n w\n n\n c\n d\n x\n j\n x\n \n \n 3\n az\n t\n n\n f\n d\n x\n l\n e\n \n \n 4\n az\n v\n n\n f\n d\n h\n d\n n\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_cat\_LE =benz\_train\_cat.apply(LabelEncoder().fit\_transform) #Applying lable encoder on catageorical data","execution\_count":1384,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_cat\_LE.head()","execution\_count":1385,"outputs":[{"output\_type":"execute\_result","execution\_count":1385,"data":{"text/plain":" X0 X1 X2 X3 X4 X5 X6 X8\n0 32 23 17 0 3 24 9 14\n1 32 21 19 4 3 28 11 14\n2 20 24 34 2 3 27 9 23\n3 20 21 34 5 3 27 11 4\n4 20 23 34 5 3 12 3 13","text/html":"

\n\n \n \n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n 32\n 23\n 17\n 0\n 3\n 24\n 9\n 14\n \n \n 1\n 32\n 21\n 19\n 4\n 3\n 28\n 11\n 14\n \n \n 2\n 20\n 24\n 34\n 2\n 3\n 27\n 9\n 23\n \n \n 3\n 20\n 21\n 34\n 5\n 3\n 27\n 11\n 4\n \n \n 4\n 20\n 23\n 34\n 5\n 3\n 12\n 3\n 13\n \n \n\n

"},"metadata":{}}]},{"metadata":{},"cell\_type":"raw","source":"Now the data is clean after Scaling the numerical data,Label encoding for catageorical data, null & unique value treatment -\ncombining catagorical and numerical data before going for dimensioanl reduction"},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_clean = pd.concat([benz\_train\_int\_var\_dropped\_y,benz\_train\_cat\_LE], axis=1,)","execution\_count":1386,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_clean= benz\_train\_clean.drop([\"ID\"], axis=1) #dropping ID column it is not required for scaling or modeling","execution\_count":1387,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_clean.head()","execution\_count":1388,"outputs":[{"output\_type":"execute\_result","execution\_count":1388,"data":{"text/plain":" X10 X12 X13 X14 X15 X16 X17 X18 X19 X20 ... X384 X385 X0 X1 \\\n0 0 0 1 0 0 0 0 1 0 0 ... 0 0 32 23 \n1 0 0 0 0 0 0 0 1 0 0 ... 0 0 32 21 \n2 0 0 0 0 0 0 1 0 0 0 ... 0 0 20 24 \n3 0 0 0 0 0 0 0 0 0 0 ... 0 0 20 21 \n4 0 0 0 0 0 0 0 0 0 0 ... 0 0 20 23 \n\n X2 X3 X4 X5 X6 X8 \n0 17 0 3 24 9 14 \n1 19 4 3 28 11 14 \n2 34 2 3 27 9 23 \n3 34 5 3 27 11 4 \n4 34 5 3 12 3 13 \n\n[5 rows x 364 columns]","text/html":"

\n\n \n \n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n X19\n X20\n ...\n X384\n X385\n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n ...\n 0\n 0\n 32\n 23\n 17\n 0\n 3\n 24\n 9\n 14\n \n \n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n ...\n 0\n 0\n 32\n 21\n 19\n 4\n 3\n 28\n 11\n 14\n \n \n 2\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n ...\n 0\n 0\n 20\n 24\n 34\n 2\n 3\n 27\n 9\n 23\n \n \n 3\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 20\n 21\n 34\n 5\n 3\n 27\n 11\n 4\n \n \n 4\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 20\n 23\n 34\n 5\n 3\n 12\n 3\n 13\n \n \n\n

5 rows × 364 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_clean.shape#checking the shape befor going to scale","execution\_count":1389,"outputs":[{"output\_type":"execute\_result","execution\_count":1389,"data":{"text/plain":"(4209, 364)"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_clean.dtypes.value\_counts()","execution\_count":1472,"outputs":[{"output\_type":"execute\_result","execution\_count":1472,"data":{"text/plain":"int64 356\nint32 8\ndtype: int64"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# Scaling the numerical data of train data"},{"metadata":{"trusted":true},"cell\_type":"code","source":"#lets scale numerical values\nfrom sklearn.preprocessing import MinMaxScaler\nmn=MinMaxScaler()","execution\_count":1391,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_clean\_mn = mn.fit\_transform(benz\_train\_clean)","execution\_count":1392,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_clean\_mn\_df =pd.DataFrame(benz\_train\_clean\_mn, columns=benz\_train\_clean.columns, index=benz\_train\_clean.index)","execution\_count":1393,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_train\_clean\_mn\_df.head()","execution\_count":1394,"outputs":[{"output\_type":"execute\_result","execution\_count":1394,"data":{"text/plain":" X10 X12 X13 X14 X15 X16 X17 X18 X19 X20 ... X384 X385 \\\n0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 ... 0.0 0.0 \n1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 ... 0.0 0.0 \n2 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 ... 0.0 0.0 \n3 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 \n4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 \n\n X0 X1 X2 X3 X4 X5 X6 X8 \n0 0.695652 0.884615 0.395349 0.000000 1.0 0.857143 0.818182 0.583333 \n1 0.695652 0.807692 0.441860 0.666667 1.0 1.000000 1.000000 0.583333 \n2 0.434783 0.923077 0.790698 0.333333 1.0 0.964286 0.818182 0.958333 \n3 0.434783 0.807692 0.790698 0.833333 1.0 0.964286 1.000000 0.166667 \n4 0.434783 0.884615 0.790698 0.833333 1.0 0.428571 0.272727 0.541667 \n\n[5 rows x 364 columns]","text/html":"

\n\n \n \n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n X19\n X20\n ...\n X384\n X385\n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n 0.0\n 0.0\n 1.0\n 0.0\n 0.0\n 0.0\n 0.0\n 1.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.695652\n 0.884615\n 0.395349\n 0.000000\n 1.0\n 0.857143\n 0.818182\n 0.583333\n \n \n 1\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 1.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.695652\n 0.807692\n 0.441860\n 0.666667\n 1.0\n 1.000000\n 1.000000\n 0.583333\n \n \n 2\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 1.0\n 0.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.434783\n 0.923077\n 0.790698\n 0.333333\n 1.0\n 0.964286\n 0.818182\n 0.958333\n \n \n 3\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.434783\n 0.807692\n 0.790698\n 0.833333\n 1.0\n 0.964286\n 1.000000\n 0.166667\n \n \n 4\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.434783\n 0.884615\n 0.790698\n 0.833333\n 1.0\n 0.428571\n 0.272727\n 0.541667\n \n \n\n

5 rows × 364 columns

\n

"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# Start cleaning the test data"},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test =pd.read\_csv(\"D:\\\\Simplilearn\\\\ML\\\\Benz\\\\test.csv\")","execution\_count":1395,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test.head()","execution\_count":1396,"outputs":[{"output\_type":"execute\_result","execution\_count":1396,"data":{"text/plain":" ID X0 X1 X2 X3 X4 X5 X6 X8 X10 ... X375 X376 X377 X378 X379 X380 \\\n0 1 az v n f d t a w 0 ... 0 0 0 1 0 0 \n1 2 t b ai a d b g y 0 ... 0 0 1 0 0 0 \n2 3 az v as f d a j j 0 ... 0 0 0 1 0 0 \n3 4 az l n f d z l n 0 ... 0 0 0 1 0 0 \n4 5 w s as c d y i m 0 ... 1 0 0 0 0 0 \n\n X382 X383 X384 X385 \n0 0 0 0 0 \n1 0 0 0 0 \n2 0 0 0 0 \n3 0 0 0 0 \n4 0 0 0 0 \n\n[5 rows x 377 columns]","text/html":"

\n\n \n \n ID\n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n X10\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n 0\n 1\n az\n v\n n\n f\n d\n t\n a\n w\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 1\n 2\n t\n b\n ai\n a\n d\n b\n g\n y\n 0\n ...\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 2\n 3\n az\n v\n as\n f\n d\n a\n j\n j\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 3\n 4\n az\n l\n n\n f\n d\n z\n l\n n\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 4\n 5\n w\n s\n as\n c\n d\n y\n i\n m\n 0\n ...\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n\n

5 rows × 377 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test.dtypes.value\_counts()","execution\_count":1397,"outputs":[{"output\_type":"execute\_result","execution\_count":1397,"data":{"text/plain":"int64 369\nobject 8\ndtype: int64"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test.isnull().any().sum() #no null values in test data too ","execution\_count":1398,"outputs":[{"output\_type":"execute\_result","execution\_count":1398,"data":{"text/plain":"0"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"#likewise train data seperate the cat and num data\nbenz\_test\_cat = benz\_test.select\_dtypes(exclude=[\"float64\",\"int64\"])\nbenz\_test\_cat.head()","execution\_count":1399,"outputs":[{"output\_type":"execute\_result","execution\_count":1399,"data":{"text/plain":" X0 X1 X2 X3 X4 X5 X6 X8\n0 az v n f d t a w\n1 t b ai a d b g y\n2 az v as f d a j j\n3 az l n f d z l n\n4 w s as c d y i m","text/html":"

\n\n \n \n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n az\n v\n n\n f\n d\n t\n a\n w\n \n \n 1\n t\n b\n ai\n a\n d\n b\n g\n y\n \n \n 2\n az\n v\n as\n f\n d\n a\n j\n j\n \n \n 3\n az\n l\n n\n f\n d\n z\n l\n n\n \n \n 4\n w\n s\n as\n c\n d\n y\n i\n m\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_int = benz\_test.select\_dtypes(include=[\"float64\",\"int64\"])\nbenz\_test\_int.head()","execution\_count":1400,"outputs":[{"output\_type":"execute\_result","execution\_count":1400,"data":{"text/plain":" ID X10 X11 X12 X13 X14 X15 X16 X17 X18 ... X375 X376 X377 \\\n0 1 0 0 0 0 0 0 0 0 0 ... 0 0 0 \n1 2 0 0 0 0 0 0 0 0 0 ... 0 0 1 \n2 3 0 0 0 0 1 0 0 0 0 ... 0 0 0 \n3 4 0 0 0 0 0 0 0 0 0 ... 0 0 0 \n4 5 0 0 0 0 1 0 0 0 0 ... 1 0 0 \n\n X378 X379 X380 X382 X383 X384 X385 \n0 1 0 0 0 0 0 0 \n1 0 0 0 0 0 0 0 \n2 1 0 0 0 0 0 0 \n3 1 0 0 0 0 0 0 \n4 0 0 0 0 0 0 0 \n\n[5 rows x 369 columns]","text/html":"

\n\n \n \n ID\n X10\n X11\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 1\n 2\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 2\n 3\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 3\n 4\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 4\n 5\n 0\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n ...\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n\n

5 rows × 369 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"# In test dataset, remove the same features of train dataset having 0 variance.\nbenz\_test\_int\_var= benz\_test\_int.drop(columns=['X11', 'X93', 'X107','X233', 'X235', 'X268', 'X289', 'X290', 'X293','X297','X330','X347'])","execution\_count":1401,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"#here on test data we will not perform Dropping the columns with variance is == 0, because it is test data\n#So we will be drooping the same columns which had 0 variance in train data","execution\_count":1402,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_int\_var.head()","execution\_count":1403,"outputs":[{"output\_type":"execute\_result","execution\_count":1403,"data":{"text/plain":" ID X10 X12 X13 X14 X15 X16 X17 X18 X19 ... X375 X376 X377 \\\n0 1 0 0 0 0 0 0 0 0 0 ... 0 0 0 \n1 2 0 0 0 0 0 0 0 0 1 ... 0 0 1 \n2 3 0 0 0 1 0 0 0 0 0 ... 0 0 0 \n3 4 0 0 0 0 0 0 0 0 0 ... 0 0 0 \n4 5 0 0 0 1 0 0 0 0 0 ... 1 0 0 \n\n X378 X379 X380 X382 X383 X384 X385 \n0 1 0 0 0 0 0 0 \n1 0 0 0 0 0 0 0 \n2 1 0 0 0 0 0 0 \n3 1 0 0 0 0 0 0 \n4 0 0 0 0 0 0 0 \n\n[5 rows x 357 columns]","text/html":"

\n\n \n \n ID\n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n X19\n ...\n X375\n X376\n X377\n X378\n X379\n X380\n X382\n X383\n X384\n X385\n \n \n \n \n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 1\n 2\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n ...\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 2\n 3\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 3\n 4\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n 4\n 5\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n ...\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n \n \n\n

5 rows × 357 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"#Label encoder for cat data\nbenz\_test\_cat.head()","execution\_count":1404,"outputs":[{"output\_type":"execute\_result","execution\_count":1404,"data":{"text/plain":" X0 X1 X2 X3 X4 X5 X6 X8\n0 az v n f d t a w\n1 t b ai a d b g y\n2 az v as f d a j j\n3 az l n f d z l n\n4 w s as c d y i m","text/html":"

\n\n \n \n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n az\n v\n n\n f\n d\n t\n a\n w\n \n \n 1\n t\n b\n ai\n a\n d\n b\n g\n y\n \n \n 2\n az\n v\n as\n f\n d\n a\n j\n j\n \n \n 3\n az\n l\n n\n f\n d\n z\n l\n n\n \n \n 4\n w\n s\n as\n c\n d\n y\n i\n m\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_cat\_LE = benz\_test\_cat.apply(LabelEncoder().fit\_transform)","execution\_count":1405,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_cat\_LE.head()","execution\_count":1406,"outputs":[{"output\_type":"execute\_result","execution\_count":1406,"data":{"text/plain":" X0 X1 X2 X3 X4 X5 X6 X8\n0 21 23 34 5 3 26 0 22\n1 42 3 8 0 3 9 6 24\n2 21 23 17 5 3 0 9 9\n3 21 13 34 5 3 31 11 13\n4 45 20 17 2 3 30 8 12","text/html":"

\n\n \n \n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n 21\n 23\n 34\n 5\n 3\n 26\n 0\n 22\n \n \n 1\n 42\n 3\n 8\n 0\n 3\n 9\n 6\n 24\n \n \n 2\n 21\n 23\n 17\n 5\n 3\n 0\n 9\n 9\n \n \n 3\n 21\n 13\n 34\n 5\n 3\n 31\n 11\n 13\n \n \n 4\n 45\n 20\n 17\n 2\n 3\n 30\n 8\n 12\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean = pd.concat([benz\_test\_int\_var,benz\_test\_cat\_LE], axis=1,)","execution\_count":1407,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean.head()","execution\_count":1408,"outputs":[{"output\_type":"execute\_result","execution\_count":1408,"data":{"text/plain":" ID X10 X12 X13 X14 X15 X16 X17 X18 X19 ... X384 X385 X0 X1 \\\n0 1 0 0 0 0 0 0 0 0 0 ... 0 0 21 23 \n1 2 0 0 0 0 0 0 0 0 1 ... 0 0 42 3 \n2 3 0 0 0 1 0 0 0 0 0 ... 0 0 21 23 \n3 4 0 0 0 0 0 0 0 0 0 ... 0 0 21 13 \n4 5 0 0 0 1 0 0 0 0 0 ... 0 0 45 20 \n\n X2 X3 X4 X5 X6 X8 \n0 34 5 3 26 0 22 \n1 8 0 3 9 6 24 \n2 17 5 3 0 9 9 \n3 34 5 3 31 11 13 \n4 17 2 3 30 8 12 \n\n[5 rows x 365 columns]","text/html":"

\n\n \n \n ID\n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n X19\n ...\n X384\n X385\n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 21\n 23\n 34\n 5\n 3\n 26\n 0\n 22\n \n \n 1\n 2\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n ...\n 0\n 0\n 42\n 3\n 8\n 0\n 3\n 9\n 6\n 24\n \n \n 2\n 3\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 21\n 23\n 17\n 5\n 3\n 0\n 9\n 9\n \n \n 3\n 4\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 21\n 13\n 34\n 5\n 3\n 31\n 11\n 13\n \n \n 4\n 5\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 45\n 20\n 17\n 2\n 3\n 30\n 8\n 12\n \n \n\n

5 rows × 365 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_ID = benz\_test\_clean[\"ID\"] #drop ID from test data as well and assign it to new variable","execution\_count":1409,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean = benz\_test\_clean.drop([\"ID\"], axis=1)","execution\_count":1410,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean.dtypes.value\_counts()","execution\_count":1411,"outputs":[{"output\_type":"execute\_result","execution\_count":1411,"data":{"text/plain":"int64 356\nint32 8\ndtype: int64"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean.head()","execution\_count":1412,"outputs":[{"output\_type":"execute\_result","execution\_count":1412,"data":{"text/plain":" X10 X12 X13 X14 X15 X16 X17 X18 X19 X20 ... X384 X385 X0 X1 \\\n0 0 0 0 0 0 0 0 0 0 0 ... 0 0 21 23 \n1 0 0 0 0 0 0 0 0 1 0 ... 0 0 42 3 \n2 0 0 0 1 0 0 0 0 0 0 ... 0 0 21 23 \n3 0 0 0 0 0 0 0 0 0 0 ... 0 0 21 13 \n4 0 0 0 1 0 0 0 0 0 0 ... 0 0 45 20 \n\n X2 X3 X4 X5 X6 X8 \n0 34 5 3 26 0 22 \n1 8 0 3 9 6 24 \n2 17 5 3 0 9 9 \n3 34 5 3 31 11 13 \n4 17 2 3 30 8 12 \n\n[5 rows x 364 columns]","text/html":"

\n\n \n \n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n X19\n X20\n ...\n X384\n X385\n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 21\n 23\n 34\n 5\n 3\n 26\n 0\n 22\n \n \n 1\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 1\n 0\n ...\n 0\n 0\n 42\n 3\n 8\n 0\n 3\n 9\n 6\n 24\n \n \n 2\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 21\n 23\n 17\n 5\n 3\n 0\n 9\n 9\n \n \n 3\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 21\n 13\n 34\n 5\n 3\n 31\n 11\n 13\n \n \n 4\n 0\n 0\n 0\n 1\n 0\n 0\n 0\n 0\n 0\n 0\n ...\n 0\n 0\n 45\n 20\n 17\n 2\n 3\n 30\n 8\n 12\n \n \n\n

5 rows × 364 columns

\n

"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# Scaling the Test data: Min Max scaler"},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean\_mn = mn.fit\_transform(benz\_test\_clean) #scaling using MinMax scaler on test data","execution\_count":1413,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean\_mn\_df =pd.DataFrame(benz\_test\_clean\_mn, columns=benz\_test\_clean.columns, index=benz\_test\_clean.index)","execution\_count":1414,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean\_mn\_df.head()","execution\_count":1415,"outputs":[{"output\_type":"execute\_result","execution\_count":1415,"data":{"text/plain":" X10 X12 X13 X14 X15 X16 X17 X18 X19 X20 ... X384 X385 X0 \\\n0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.4375 \n1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 ... 0.0 0.0 0.8750 \n2 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.4375 \n3 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.4375 \n4 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.9375 \n\n X1 X2 X3 X4 X5 X6 X8 \n0 0.884615 0.772727 0.833333 1.0 0.838710 0.000000 0.916667 \n1 0.115385 0.181818 0.000000 1.0 0.290323 0.545455 1.000000 \n2 0.884615 0.386364 0.833333 1.0 0.000000 0.818182 0.375000 \n3 0.500000 0.772727 0.833333 1.0 1.000000 1.000000 0.541667 \n4 0.769231 0.386364 0.333333 1.0 0.967742 0.727273 0.500000 \n\n[5 rows x 364 columns]","text/html":"

\n\n \n \n X10\n X12\n X13\n X14\n X15\n X16\n X17\n X18\n X19\n X20\n ...\n X384\n X385\n X0\n X1\n X2\n X3\n X4\n X5\n X6\n X8\n \n \n \n \n 0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.4375\n 0.884615\n 0.772727\n 0.833333\n 1.0\n 0.838710\n 0.000000\n 0.916667\n \n \n 1\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 1.0\n 0.0\n ...\n 0.0\n 0.0\n 0.8750\n 0.115385\n 0.181818\n 0.000000\n 1.0\n 0.290323\n 0.545455\n 1.000000\n \n \n 2\n 0.0\n 0.0\n 0.0\n 1.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.4375\n 0.884615\n 0.386364\n 0.833333\n 1.0\n 0.000000\n 0.818182\n 0.375000\n \n \n 3\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.4375\n 0.500000\n 0.772727\n 0.833333\n 1.0\n 1.000000\n 1.000000\n 0.541667\n \n \n 4\n 0.0\n 0.0\n 0.0\n 1.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n 0.0\n ...\n 0.0\n 0.0\n 0.9375\n 0.769231\n 0.386364\n 0.333333\n 1.0\n 0.967742\n 0.727273\n 0.500000\n \n \n\n

5 rows × 364 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_clean\_mn\_df.shape","execution\_count":1416,"outputs":[{"output\_type":"execute\_result","execution\_count":1416,"data":{"text/plain":"(4209, 364)"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# PCA- Dimensional Reduction using PCA"},{"metadata":{"trusted":true},"cell\_type":"code","source":"from sklearn.decomposition import PCA\npca = PCA()\npca\_benz\_train\_clean\_mn\_df = pca.fit\_transform(benz\_train\_clean\_mn\_df)","execution\_count":1417,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"pca\_transform\_test\_90 = pca\_90.transform(benz\_test\_clean\_mn\_df)","execution\_count":1418,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"explained\_variance = pca.explained\_variance\_ratio\_\n#print(explained\_variance)","execution\_count":1419,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"explained\_variance.sum()","execution\_count":1420,"outputs":[{"output\_type":"execute\_result","execution\_count":1420,"data":{"text/plain":"1.0"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"per\_var = pd.DataFrame(np.round(pca.explained\_variance\_ratio\_ \* 100, decimals=3))\nlabels = ['PC' + str(x) for x in range(1, len(per\_var)+1)]","execution\_count":1421,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"per\_var.sum()","execution\_count":1422,"outputs":[{"output\_type":"execute\_result","execution\_count":1422,"data":{"text/plain":"0 100.001\ndtype: float64"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"per\_var.head(10)","execution\_count":1423,"outputs":[{"output\_type":"execute\_result","execution\_count":1423,"data":{"text/plain":" 0\n0 12.972\n1 8.752\n2 8.519\n3 6.748\n4 5.638\n5 4.696\n6 3.763\n7 3.240\n8 2.779\n9 2.475","text/html":"

\n\n \n \n 0\n \n \n \n \n 0\n 12.972\n \n \n 1\n 8.752\n \n \n 2\n 8.519\n \n \n 3\n 6.748\n \n \n 4\n 5.638\n \n \n 5\n 4.696\n \n \n 6\n 3.763\n \n \n 7\n 3.240\n \n \n 8\n 2.779\n \n \n 9\n 2.475\n \n \n\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"per\_var.head(50).sum() #to get 90% accuracy we need to consider first 50 columns- later we can change it accordingly","execution\_count":1424,"outputs":[{"output\_type":"execute\_result","execution\_count":1424,"data":{"text/plain":"0 90.116\ndtype: float64"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"pca\_90 = PCA(0.9) #passing the parameters that 90% accuracy needed\npca\_fit\_90 = pca\_90.fit\_transform(benz\_train\_clean\_mn\_df)","execution\_count":1425,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"per\_var\_90 = np.round(pca\_90.explained\_variance\_ratio\_ \* 100, decimals=2)","execution\_count":1426,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"np.cumsum(per\_var\_90)\n#out of 364 columns we are considering 50 columns","execution\_count":1427,"outputs":[{"output\_type":"execute\_result","execution\_count":1427,"data":{"text/plain":"array([12.97, 21.72, 30.24, 36.99, 42.63, 47.33, 51.09, 54.33, 57.11,\n 59.58, 61.89, 63.84, 65.54, 67.17, 68.69, 70.15, 71.53, 72.74,\n 73.77, 74.78, 75.74, 76.63, 77.51, 78.34, 79.13, 79.89, 80.61,\n 81.26, 81.9 , 82.47, 83. , 83.51, 83.99, 84.45, 84.89, 85.31,\n 85.73, 86.13, 86.53, 86.91, 87.27, 87.61, 87.95, 88.29, 88.62,\n 88.93, 89.24, 89.54, 89.83, 90.11])"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"print(len(pca\_fit\_90[0]))","execution\_count":1428,"outputs":[{"output\_type":"stream","text":"50\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"labels = ['PC' + str(x) for x in range(1, len(per\_var\_90)+1)]","execution\_count":1429,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"import matplotlib.pyplot as plt\nplt.figure(figsize=(25,10))\nplt.bar(x=range(1, len(per\_var\_90)+1),height= per\_var\_90,tick\_label=labels)\nplt.ylabel(\"Percentage of Variance Explained\")\nplt.xlabel(\"Principal Components\")\nplt.title('Scree Plot')\nplt.show()","execution\_count":1430,"outputs":[{"output\_type":"display\_data","data":{"text/plain":"","image/png":"\n"},"metadata":{"needs\_background":"light"}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"pca\_90\_df = pd.DataFrame(pca\_fit\_90,columns=labels )","execution\_count":1431,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"pca\_90\_df.head()","execution\_count":1432,"outputs":[{"output\_type":"execute\_result","execution\_count":1432,"data":{"text/plain":" PC1 PC2 PC3 PC4 PC5 PC6 PC7 \\\n0 0.682272 2.217390 1.233625 0.885738 1.401423 0.054226 0.654869 \n1 -0.279051 1.164201 -0.764263 -0.660639 0.237862 0.066811 1.237312 \n2 -1.018083 2.979512 0.558557 2.540751 -0.926713 3.282629 -0.940275 \n3 -0.658559 2.545045 -0.425408 2.997377 -1.681631 3.134973 0.074150 \n4 -0.652313 2.370739 -0.583703 3.194208 -1.999394 3.167654 -0.143355 \n\n PC8 PC9 PC10 ... PC41 PC42 PC43 PC44 \\\n0 -0.937416 0.192658 -0.678833 ... -0.301784 0.082351 0.376096 -0.123603 \n1 -0.530352 -0.108645 0.499650 ... -0.090422 0.155252 0.025554 0.148211 \n2 0.557085 -0.926063 -0.108851 ... 0.392666 0.341743 0.352159 -0.072961 \n3 0.083995 -1.072810 0.243721 ... -0.630611 -0.219829 -0.070023 -0.619896 \n4 0.229284 -1.754722 -0.355889 ... -0.003493 -0.434347 0.188307 -0.517897 \n\n PC45 PC46 PC47 PC48 PC49 PC50 \n0 1.037195 0.282469 -0.741828 -0.503630 -1.131286 -0.650869 \n1 0.105227 -0.089335 -0.013777 0.042125 -0.106445 0.008773 \n2 -0.213758 0.100651 -0.554772 -0.251824 -0.499742 -0.178818 \n3 -0.260746 -0.028494 0.051949 0.077641 0.366246 -0.427390 \n4 0.024977 -0.098782 0.116541 0.084835 0.035156 -0.312985 \n\n[5 rows x 50 columns]","text/html":"

\n\n \n \n PC1\n PC2\n PC3\n PC4\n PC5\n PC6\n PC7\n PC8\n PC9\n PC10\n ...\n PC41\n PC42\n PC43\n PC44\n PC45\n PC46\n PC47\n PC48\n PC49\n PC50\n \n \n \n \n 0\n 0.682272\n 2.217390\n 1.233625\n 0.885738\n 1.401423\n 0.054226\n 0.654869\n -0.937416\n 0.192658\n -0.678833\n ...\n -0.301784\n 0.082351\n 0.376096\n -0.123603\n 1.037195\n 0.282469\n -0.741828\n -0.503630\n -1.131286\n -0.650869\n \n \n 1\n -0.279051\n 1.164201\n -0.764263\n -0.660639\n 0.237862\n 0.066811\n 1.237312\n -0.530352\n -0.108645\n 0.499650\n ...\n -0.090422\n 0.155252\n 0.025554\n 0.148211\n 0.105227\n -0.089335\n -0.013777\n 0.042125\n -0.106445\n 0.008773\n \n \n 2\n -1.018083\n 2.979512\n 0.558557\n 2.540751\n -0.926713\n 3.282629\n -0.940275\n 0.557085\n -0.926063\n -0.108851\n ...\n 0.392666\n 0.341743\n 0.352159\n -0.072961\n -0.213758\n 0.100651\n -0.554772\n -0.251824\n -0.499742\n -0.178818\n \n \n 3\n -0.658559\n 2.545045\n -0.425408\n 2.997377\n -1.681631\n 3.134973\n 0.074150\n 0.083995\n -1.072810\n 0.243721\n ...\n -0.630611\n -0.219829\n -0.070023\n -0.619896\n -0.260746\n -0.028494\n 0.051949\n 0.077641\n 0.366246\n -0.427390\n \n \n 4\n -0.652313\n 2.370739\n -0.583703\n 3.194208\n -1.999394\n 3.167654\n -0.143355\n 0.229284\n -1.754722\n -0.355889\n ...\n -0.003493\n -0.434347\n 0.188307\n -0.517897\n 0.024977\n -0.098782\n 0.116541\n 0.084835\n 0.035156\n -0.312985\n \n \n\n

5 rows × 50 columns

\n

"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"pca\_90\_df.shape","execution\_count":1433,"outputs":[{"output\_type":"execute\_result","execution\_count":1433,"data":{"text/plain":"(4209, 50)"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"!pip install xgboost #installing Xgboost","execution\_count":1434,"outputs":[{"output\_type":"stream","text":"Requirement already satisfied: xgboost in c:\\programdata\\anaconda3\\lib\\site-packages (1.4.2)\nRequirement already satisfied: scipy in c:\\programdata\\anaconda3\\lib\\site-packages (from xgboost) (1.6.2)\nRequirement already satisfied: numpy in c:\\programdata\\anaconda3\\lib\\site-packages (from xgboost) (1.20.1)\n","name":"stdout"}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"# import the required libraries\nimport xgboost as xgb\nfrom sklearn.model\_selection import cross\_val\_score,cross\_val\_predict\nfrom xgboost import XGBRegressor\nfrom sklearn.model\_selection import RandomizedSearchCV","execution\_count":1435,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"XGBRegressor()","execution\_count":1437,"outputs":[{"output\_type":"execute\_result","execution\_count":1437,"data":{"text/plain":"XGBRegressor(base\_score=None, booster=None, colsample\_bylevel=None,\n colsample\_bynode=None, colsample\_bytree=None, gamma=None,\n gpu\_id=None, importance\_type='gain', interaction\_constraints=None,\n learning\_rate=None, max\_delta\_step=None, max\_depth=None,\n min\_child\_weight=None, missing=nan, monotone\_constraints=None,\n n\_estimators=100, n\_jobs=None, num\_parallel\_tree=None,\n random\_state=None, reg\_alpha=None, reg\_lambda=None,\n scale\_pos\_weight=None, subsample=None, tree\_method=None,\n validate\_parameters=None, verbosity=None)"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"#from skitlearn site we gathered the range where the best parameters lies in\nparams={ 'learning\_rate' : [0.01,0.05,0.1,1] ,\n 'max\_depth' : [2,3,5,10],\n 'min\_child\_weight': [ 0, 1, 3],\n 'n\_estimators' : [100,150,200,500],\n 'gamma' : [1e-2,1e-3,0,0.1,0.01,0.5,1],\n 'colsample\_bytree': [0.1,0.5,0.7,1],\n 'subsample' : [0.2,0.3,0.5,1],\n 'reg\_lambda' : [0,1,10],\n 'reg\_alpha' : [1e-5,1e-3,1e-1,1,1e1]}","execution\_count":1438,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"xgb\_reg = xgb.XGBRegressor()","execution\_count":1439,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"Random\_Search=RandomizedSearchCV(xgb\_reg,params,cv=10, scoring='r2', return\_train\_score=True, n\_jobs=-1,verbose=1) ","execution\_count":1440,"outputs":[]},{"metadata":{},"cell\_type":"raw","source":"(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,\n colsample\_bynode=1, colsample\_bytree=0.5, gamma=0.01, gpu\_id=-1,\n importance\_type='gain', interaction\_constraints='',\n learning\_rate=0.1, max\_delta\_step=0, max\_depth=2,\n min\_child\_weight=3, missing=None, monotone\_constraints='()',\n n\_estimators=500, n\_jobs=2, num\_parallel\_tree=1, random\_state=0,\n reg\_alpha=0.1, reg\_lambda=10, scale\_pos\_weight=1, subsample=0.5,\n tree\_method='exact', validate\_parameters=1, verbosity=None)"},{"metadata":{"trusted":true},"cell\_type":"code","source":"Random\_Search.fit(pca\_fit\_90,train\_y)","execution\_count":1441,"outputs":[{"output\_type":"stream","text":"Fitting 10 folds for each of 10 candidates, totalling 100 fits\n","name":"stdout"},{"output\_type":"execute\_result","execution\_count":1441,"data":{"text/plain":"RandomizedSearchCV(cv=10,\n estimator=XGBRegressor(base\_score=None, booster=None,\n colsample\_bylevel=None,\n colsample\_bynode=None,\n colsample\_bytree=None, gamma=None,\n gpu\_id=None, importance\_type='gain',\n interaction\_constraints=None,\n learning\_rate=None,\n max\_delta\_step=None, max\_depth=None,\n min\_child\_weight=None, missing=nan,\n monotone\_constraints=None,\n n\_estimators=100,...\n n\_jobs=-1,\n param\_distributions={'colsample\_bytree': [0.1, 0.5, 0.7, 1],\n 'gamma': [0.01, 0.001, 0, 0.1, 0.01,\n 0.5, 1],\n 'learning\_rate': [0.01, 0.05, 0.1, 1],\n 'max\_depth': [2, 3, 5, 10],\n 'min\_child\_weight': [0, 1, 3],\n 'n\_estimators': [100, 150, 200, 500],\n 'reg\_alpha': [1e-05, 0.001, 0.1, 1,\n 10.0],\n 'reg\_lambda': [0, 1, 10],\n 'subsample': [0.2, 0.3, 0.5, 1]},\n return\_train\_score=True, scoring='r2', verbose=1)"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"Random\_Search.best\_params\_ #best parameters from random search ","execution\_count":1442,"outputs":[{"output\_type":"execute\_result","execution\_count":1442,"data":{"text/plain":"{'subsample': 0.5,\n 'reg\_lambda': 0,\n 'reg\_alpha': 10.0,\n 'n\_estimators': 200,\n 'min\_child\_weight': 1,\n 'max\_depth': 2,\n 'learning\_rate': 0.1,\n 'gamma': 0.5,\n 'colsample\_bytree': 1}"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"Random\_Search.best\_estimator\_ #best estimators from random search ","execution\_count":1443,"outputs":[{"output\_type":"execute\_result","execution\_count":1443,"data":{"text/plain":"XGBRegressor(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,\n colsample\_bynode=1, colsample\_bytree=1, gamma=0.5, gpu\_id=-1,\n importance\_type='gain', interaction\_constraints='',\n learning\_rate=0.1, max\_delta\_step=0, max\_depth=2,\n min\_child\_weight=1, missing=nan, monotone\_constraints='()',\n n\_estimators=200, n\_jobs=8, num\_parallel\_tree=1, random\_state=0,\n reg\_alpha=10.0, reg\_lambda=0, scale\_pos\_weight=1, subsample=0.5,\n tree\_method='exact', validate\_parameters=1, verbosity=None)"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"# Instantiate the XGBoost classifier with the best estimators and parameters\nxgb\_reg=XGBRegressor(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,\n colsample\_bynode=1, colsample\_bytree=0.5, gamma=0.01, gpu\_id=-1,\n importance\_type='gain', interaction\_constraints='',\n learning\_rate=0.1, max\_delta\_step=0, max\_depth=2,\n min\_child\_weight=3, monotone\_constraints='()',\n n\_estimators=500, n\_jobs=2, num\_parallel\_tree=1, random\_state=0,\n reg\_alpha=0.1, reg\_lambda=10, scale\_pos\_weight=1, subsample=0.5,\n tree\_method='exact', validate\_parameters=1, verbosity=None)","execution\_count":1461,"outputs":[]},{"metadata":{},"cell\_type":"raw","source":"r2\_Score = cross\_val\_score(xgb\_reg,pca\_fit\_90,train\_y,scoring='r2',cv=10)"},{"metadata":{"trusted":true},"cell\_type":"code","source":"r2\_Score","execution\_count":1462,"outputs":[{"output\_type":"execute\_result","execution\_count":1462,"data":{"text/plain":"array([0.59189747, 0.5571714 , 0.46061637, 0.60394419, 0.61082587,\n 0.56740745, 0.41728968, 0.58178231, 0.59871083, 0.60498835])"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"round(r2\_Score.mean(),2) #r-square is between 50-100 % so it is okay to move on with the model","execution\_count":1463,"outputs":[{"output\_type":"execute\_result","execution\_count":1463,"data":{"text/plain":"0.56"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"This means the model explains 56% variability of the target variable (y)(time) around its mean.","execution\_count":null,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"xgb\_reg.fit(pca\_fit\_90,train\_y)","execution\_count":1464,"outputs":[{"output\_type":"execute\_result","execution\_count":1464,"data":{"text/plain":"XGBRegressor(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,\n colsample\_bynode=1, colsample\_bytree=0.5, gamma=0.01, gpu\_id=-1,\n importance\_type='gain', interaction\_constraints='',\n learning\_rate=0.1, max\_delta\_step=0, max\_depth=2,\n min\_child\_weight=3, missing=nan, monotone\_constraints='()',\n n\_estimators=500, n\_jobs=2, num\_parallel\_tree=1, random\_state=0,\n reg\_alpha=0.1, reg\_lambda=10, scale\_pos\_weight=1, subsample=0.5,\n tree\_method='exact', validate\_parameters=1, verbosity=None)"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# Passing the test data for predection"},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_pred=xgb\_reg.predict(pca\_transform\_test\_90)","execution\_count":1466,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"benz\_test\_pred","execution\_count":1467,"outputs":[{"output\_type":"execute\_result","execution\_count":1467,"data":{"text/plain":"array([ 78.87767, 93.90811, 78.36242, ..., 92.9734 , 106.48379,\n 90.44 ], dtype=float32)"},"metadata":{}}]},{"metadata":{"trusted":true},"cell\_type":"code","source":"# print the predicted value (time) in the form of table\ndf\_test\_pred = pd.DataFrame({'ID': benz\_test\_ID, 'y': benz\_test\_pred})","execution\_count":1468,"outputs":[]},{"metadata":{"trusted":true},"cell\_type":"code","source":"# Print the first 10 predicted values\ndf\_test\_pred.head(10)","execution\_count":1470,"outputs":[{"output\_type":"execute\_result","execution\_count":1470,"data":{"text/plain":" ID y\n0 1 78.877670\n1 2 93.908112\n2 3 78.362419\n3 4 78.691818\n4 5 109.884758\n5 8 92.630020\n6 10 109.781242\n7 11 96.902756\n8 12 110.645546\n9 14 91.780350","text/html":"

\n\n \n \n ID\n y\n \n \n \n \n 0\n 1\n 78.877670\n \n \n 1\n 2\n 93.908112\n \n \n 2\n 3\n 78.362419\n \n \n 3\n 4\n 78.691818\n \n \n 4\n 5\n 109.884758\n \n \n 5\n 8\n 92.630020\n \n \n 6\n 10\n 109.781242\n \n \n 7\n 11\n 96.902756\n \n \n 8\n 12\n 110.645546\n \n \n 9\n 14\n 91.780350\n \n \n\n

"},"metadata":{}}]},{"metadata":{},"cell\_type":"markdown","source":"# Thank you"},{"metadata":{"trusted":true},"cell\_type":"code","source":"","execution\_count":null,"outputs":[]}]