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 :

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

```
# Create an ML algorithm that can accurately predict the time a car
will spend on the test bench
# based on the vehicle configuration

# Agenda
# 1. If for any column(s), the variance is equal to zero, then you
need to remove those variable(s)
# 2. Check for null and unique values for test and train sets
# 3. Apply label encoder for categorical variables
# 4. Perform dimensaionlity reduction with PCA
# 5. Predict the test_df values using xgboost
```

```python
# import required libraries
import pandas as pd
import numpy as np
```

```python
# load train dataset
train = pd.read_csv("train.csv")
# first few rows of train dataset
train.head()
```

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

   X380  X382  X383  X384  X385
0     0     0     0     0     0
1     0     0     0     0     0
2     0     1     0     0     0
3     0     0     0     0     0
4     0     0     0     0     0

[5 rows x 378 columns]
```

```python
# size of train dataset
print("Size of train dataset: {}".format(train.shape))
```

```
Size of train dataset: (4209, 378)
```

```python
# train dataset info
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 378 entries, ID to X385
dtypes: float64(1), int64(369), object(8)
memory usage: 12.1+ MB
```

```python
# Get y_train by separating y column as this is for prediction output
y_train = train["y"].values
y_train
```

```
array([130.81,  88.53,  76.26, ..., 109.22,  87.48, 110.85])
```

```
# loading test dataset
test = pd.read_csv("test.csv")
test.head()
```

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

   X382  X383  X384  X385
0     0     0     0     0
1     0     0     0     0
2     0     0     0     0
3     0     0     0     0
4     0     0     0     0

[5 rows x 377 columns]
```

```
# size of test dataset
print("Size of test dataset: {}".format(test.shape))
```

```
Size of test dataset: (4209, 377)
```

```
# info of test dataset
print(test.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Columns: 377 entries, ID to X385
dtypes: int64(369), object(8)
memory usage: 12.1+ MB
None
```

```
train.columns
```

```
Index(['ID', 'y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383',
'X384',
       'X385'],
      dtype='object', length=378)
```

```
# Creating the final dataset
# Removing unwanted columns ID, y from dataset
```

```
column = list(set(train.columns) - set(['ID', 'y']))
X_train = train[column]
X_test = test[column]
print (X_train.shape)
print (X_test.shape)
```

```
(4209, 376)
(4209, 376)
```

Check for NULL and unique value for test and train data sets

```
# check for NULL value
def IsNULL (df):
    if df.isnull().any().any():
        print ("YES")
    else:
        print ("NO")

IsNULL(X_train)
IsNULL(X_test)
```

```
NO
NO
```

```
## Exploratory Data Analysis (EDA)
# Integer Columns Analysis
unique_value_dict = {}
for col in X_train.columns:
    if col not in ["ID", "y", "X0", "X1", "X2", "X3", "X4", "X5",
"X6", "X8"]:
        unique_value = str(np.sort(X_train[col].unique()).tolist())
        t_list = unique_value_dict.get(unique_value, [])
        t_list.append(col)
        unique_value_dict[unique_value] = t_list[:]
for unique_val, columns in unique_value_dict.items():
    print("Columns containing the unique values: {} Columns {}:
".format(unique_val, columns))

print("-----------------------------------------------------------")
```

```
Columns containing the unique values: [0, 1] Columns ['X248', 'X272',
'X156', 'X204', 'X184', 'X240', 'X320', 'X221', 'X309', 'X165', 'X31',
'X367', 'X258', 'X236', 'X143', 'X364', 'X291', 'X328', 'X222', 'X88',
'X206', 'X161', 'X249', 'X343', 'X212', 'X215', 'X105', 'X85', 'X171',
'X119', 'X69', 'X102', 'X363', 'X207', 'X225', 'X90', 'X150', 'X344',
'X166', 'X47', 'X261', 'X44', 'X81', 'X368', 'X46', 'X211', 'X182',
'X37', 'X359', 'X378', 'X326', 'X369', 'X155', 'X253', 'X241', 'X29',
'X95', 'X71', 'X199', 'X179', 'X255', 'X76', 'X130', 'X73', 'X147',
'X299', 'X287', 'X109', 'X163', 'X126', 'X101', 'X218', 'X327',
'X349', 'X41', 'X128', 'X229', 'X82', 'X357', 'X177', 'X383', 'X21',
'X242', 'X323', 'X180', 'X66', 'X190', 'X60', 'X280', 'X42', 'X194',
```

```
'X115', 'X247', 'X315', 'X277', 'X244', 'X324', 'X238', 'X198',
'X304', 'X264', 'X350', 'X168', 'X87', 'X99', 'X113', 'X278', 'X275',
'X321', 'X282', 'X28', 'X281', 'X84', 'X185', 'X231', 'X239', 'X274',
'X216', 'X237', 'X256', 'X57', 'X62', 'X123', 'X27', 'X75', 'X354',
'X152', 'X217', 'X136', 'X232', 'X341', 'X65', 'X370', 'X195', 'X284',
'X124', 'X54', 'X224', 'X311', 'X338', 'X167', 'X56', 'X18', 'X329',
'X270', 'X58', 'X154', 'X246', 'X306', 'X322', 'X53', 'X223', 'X251',
'X13', 'X302', 'X50', 'X337', 'X189', 'X96', 'X48', 'X157', 'X148',
'X260', 'X263', 'X139', 'X210', 'X245', 'X205', 'X382', 'X17', 'X336',
'X61', 'X385', 'X305', 'X365', 'X376', 'X214', 'X226', 'X286', 'X131',
'X234', 'X301', 'X52', 'X64', 'X144', 'X183', 'X15', 'X111', 'X312',
'X178', 'X271', 'X308', 'X292', 'X20', 'X30', 'X160', 'X145', 'X339',
'X377', 'X116', 'X209', 'X74', 'X219', 'X283', 'X36', 'X252', 'X33',
'X334', 'X14', 'X98', 'X151', 'X142', 'X325', 'X94', 'X125', 'X340',
'X127', 'X366', 'X259', 'X91', 'X345', 'X67', 'X208', 'X296', 'X257',
'X40', 'X80', 'X375', 'X19', 'X358', 'X295', 'X314', 'X348', 'X70',
'X267', 'X86', 'X243', 'X254', 'X12', 'X39', 'X220', 'X68', 'X279',
'X100', 'X356', 'X32', 'X313', 'X49', 'X197', 'X176', 'X135', 'X262',
'X266', 'X310', 'X78', 'X110', 'X92', 'X361', 'X164', 'X203', 'X129',
'X298', 'X187', 'X269', 'X23', 'X355', 'X146', 'X181', 'X120', 'X59',
'X250', 'X104', 'X132', 'X103', 'X285', 'X374', 'X138', 'X112',
'X307', 'X159', 'X288', 'X379', 'X79', 'X158', 'X192', 'X346', 'X55',
'X118', 'X26', 'X371', 'X200', 'X342', 'X230', 'X384', 'X316', 'X175',
'X353', 'X294', 'X140', 'X228', 'X51', 'X174', 'X77', 'X114', 'X351',
'X106', 'X117', 'X16', 'X173', 'X317', 'X201', 'X35', 'X45', 'X335',
'X108', 'X319', 'X276', 'X331', 'X372', 'X89', 'X332', 'X24', 'X196',
'X153', 'X63', 'X318', 'X333', 'X34', 'X43', 'X169', 'X360', 'X191',
'X227', 'X172', 'X38', 'X141', 'X380', 'X134', 'X162', 'X97', 'X265',
'X170', 'X137', 'X22', 'X186', 'X10', 'X213', 'X373', 'X83', 'X133',
'X202', 'X362', 'X273', 'X122', 'X352', 'X300']:
-----------------------------------------------------------
Columns containing the unique values: [0] Columns ['X293', 'X290',
'X233', 'X93', 'X107', 'X235', 'X297', 'X11', 'X289', 'X347', 'X330',
'X268']:
-----------------------------------------------------------
```

Remove columns with variance zero

```python
# Remove columns with a variance of 0
for colmn in column:
    colmn_len = len(np.unique(X_train[colmn]))
    if colmn_len == 1:
        X_train = X_train.drop(colmn, axis = 1)
        X_test = X_test.drop(colmn, axis = 1)

X_train.head()
```

```
   X248  X272  X156  X204  X184  X240  X320  X221  X309  X165  ...
X213  \
0     0     0     1     1     1     0     0     0     0     0  ...
0
```

```
1     0     0     1     0     0     0     0     0     0     1  ...
0
2     0     1     0     0     0     0     0     0     0     0  ...
0
3     0     1     0     0     0     0     0     0     0     0  ...
0
4     0     1     0     0     0     0     0     0     0     0  ...
0

   X373  X83  X133  X202  X362  X273  X122  X352 X300
0     0    0     0     0     0     1     0     0    0
1     0    0     0     0     0     1     0     0    0
2     0    0     0     0     0     1     0     0    0
3     0    0     0     0     0     1     0     0    0
4     0    0     0     0     0     1     0     0    0

[5 rows x 364 columns]
```

Apply label encoder

```
# Label encoding the Categorical columns
from sklearn import preprocessing
for col in ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']:
    label_encoder = preprocessing.LabelEncoder()
    label_encoder.fit(list(X_train[col].values))
    X_train[col] = label_encoder.transform(list(X_train[col].values))
```

Perform dimensionality reduction

```
# performing dimentionality reduction with PCA
from sklearn.decomposition import PCA
n_comp = 12
pca = PCA(n_components = n_comp, random_state = 42)
pca_result_train = pca.fit_transform(X_train)
#pca_result_test = pca.transform(X_test)
print(pca_result_train)
#print(pca_result_test)
```

```
[[  0.6147646   -0.13300945  15.62446002 ...   1.73751747   0.28952955
    0.35790984]
 [  0.56540665   1.56033294  17.9095812  ...  -0.13654979   0.76262443
   -0.36508512]
 [ 16.20171258  12.29284626  17.6335395  ...  -0.48524615  -1.03728745
    3.90819297]
 ...
 [ 29.00466039  14.86090532  -7.75333217 ...  -1.09559585   1.40194745
   -0.35839137]
 [ 22.97242171   1.68482437  -9.03124768 ...   0.254992     1.27428371
   -1.10552034]
 [-17.28304831  -9.95198181  -3.71935977 ...   0.28690991   0.43211075
   -0.7158175 ]]
```

Predict your test_df values using XGBoost.

```
# ML Modeling with XGboost
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

# Splitting the data by 80/20
x_train, x_valid, y_train, y_valid =
train_test_split(pca_result_train, y_train, test_size = 0.2,
random_state = 42)

# Building the final feature set
f_train = xgb.DMatrix(x_train, label = y_train)
f_valid = xgb.DMatrix(x_valid, label = y_valid)

#f_test = xgb.DMatrix(x_test)
#f_test = xgb.DMatrix(pca_result_test)

# Setting the parameters for XGB
params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.02  ## eta means learning rate
params['max_depth'] = 4

# Create function to Predict the score

def scorer(m, w):
    labels = w.get_label()
    return 'r2', r2_score(labels, m)

final_set = [(f_train, 'train'), (f_valid, 'valid')]

P = xgb.train(params, f_train, 1000, final_set,
early_stopping_rounds=50, feval=scorer, maximize=True,
verbose_eval=10)
```

```
[22:19:51] WARNING: /workspace/src/objective/regression_obj.cu:167:
reg:linear is now deprecated in favor of reg:squarederror.
[0]	train-rmse:98.99704	valid-rmse:98.88675	train-r2:-59.49743
	valid-r2:-61.82424
Multiple eval metrics have been passed: 'valid-r2' will be used for
early stopping.

Will train until valid-r2 hasn't improved in 50 rounds.
[10]	train-rmse:81.14532	valid-rmse:81.05431	train-r2:-39.64615
	valid-r2:-41.20883
[20]	train-rmse:66.60017	valid-rmse:66.52771	train-r2:-26.38061
	valid-r2:-27.43520
[30]	train-rmse:54.76085	valid-rmse:54.72092	train-r2:-17.51112
	valid-r2:-18.23791
```

```
[40]	train-rmse:45.14306	valid-rmse:45.11907	train-r2:-11.57983	valid-r2:-12.07891
[50]	train-rmse:37.35343	valid-rmse:37.35661	train-r2:-7.61298	valid-r2:-7.96573
[60]	train-rmse:31.07077	valid-rmse:31.08922	train-r2:-4.95932	valid-r2:-5.20970
[70]	train-rmse:26.02810	valid-rmse:26.04551	train-r2:-3.18194	valid-r2:-3.35830
[80]	train-rmse:22.00455	valid-rmse:22.02654	train-r2:-1.98894	valid-r2:-2.11705
[90]	train-rmse:18.81812	valid-rmse:18.84555	train-r2:-1.18597	valid-r2:-1.28175
[100]	train-rmse:16.32131	valid-rmse:16.36671	train-r2:-0.64438	valid-r2:-0.72097
[110]	train-rmse:14.38446	valid-rmse:14.44833	train-r2:-0.27726	valid-r2:-0.34118
[120]	train-rmse:12.89840	valid-rmse:12.98699	train-r2:-0.02699	valid-r2:-0.08360
[130]	train-rmse:11.78597	valid-rmse:11.90356	train-r2:0.14252	valid-r2:0.08966
[140]	train-rmse:10.95228	valid-rmse:11.09884	train-r2:0.25954	valid-r2:0.20858
[150]	train-rmse:10.33580	valid-rmse:10.52667	train-r2:0.34055	valid-r2:0.28808
[160]	train-rmse:9.87566	valid-rmse:10.10633	train-r2:0.39796	valid-r2:0.34380
[170]	train-rmse:9.54199	valid-rmse:9.81491	train-r2:0.43796	valid-r2:0.38110
[180]	train-rmse:9.29620	valid-rmse:9.61114	train-r2:0.46654	valid-r2:0.40653
[190]	train-rmse:9.11453	valid-rmse:9.47596	train-r2:0.48718	valid-r2:0.42310
[200]	train-rmse:8.97785	valid-rmse:9.37098	train-r2:0.50245	valid-r2:0.43582
[210]	train-rmse:8.85440	valid-rmse:9.30220	train-r2:0.51604	valid-r2:0.44407
[220]	train-rmse:8.74807	valid-rmse:9.24707	train-r2:0.52759	valid-r2:0.45064
[230]	train-rmse:8.66516	valid-rmse:9.20904	train-r2:0.53650	valid-r2:0.45515
[240]	train-rmse:8.59832	valid-rmse:9.18151	train-r2:0.54363	valid-r2:0.45840
[250]	train-rmse:8.53190	valid-rmse:9.15776	train-r2:0.55065	valid-r2:0.46120
[260]	train-rmse:8.47094	valid-rmse:9.14297	train-r2:0.55705	valid-r2:0.46294
[270]	train-rmse:8.41722	valid-rmse:9.13281	train-r2:0.56265	valid-r2:0.46413
[280]	train-rmse:8.37159	valid-rmse:9.12389	train-r2:0.56738	valid-r2:0.46518
```

```
[290]	train-rmse:8.32141	valid-rmse:9.11911	train-r2:0.57255
	valid-r2:0.46574
[300]	train-rmse:8.27764	valid-rmse:9.11133	train-r2:0.57703
	valid-r2:0.46665
[310]	train-rmse:8.23678	valid-rmse:9.10996	train-r2:0.58120
	valid-r2:0.46681
[320]	train-rmse:8.19585	valid-rmse:9.10537	train-r2:0.58535
	valid-r2:0.46735
[330]	train-rmse:8.16416	valid-rmse:9.10452	train-r2:0.58855
	valid-r2:0.46744
[340]	train-rmse:8.13607	valid-rmse:9.10184	train-r2:0.59138
	valid-r2:0.46776
[350]	train-rmse:8.10561	valid-rmse:9.09983	train-r2:0.59443
	valid-r2:0.46799
[360]	train-rmse:8.07291	valid-rmse:9.09400	train-r2:0.59770
	valid-r2:0.46867
[370]	train-rmse:8.04752	valid-rmse:9.09156	train-r2:0.60023
	valid-r2:0.46896
[380]	train-rmse:8.01868	valid-rmse:9.09196	train-r2:0.60308
	valid-r2:0.46891
[390]	train-rmse:7.99414	valid-rmse:9.09013	train-r2:0.60551
	valid-r2:0.46913
[400]	train-rmse:7.95877	valid-rmse:9.09192	train-r2:0.60899
	valid-r2:0.46892
[410]	train-rmse:7.93433	valid-rmse:9.09270	train-r2:0.61139
	valid-r2:0.46883
[420]	train-rmse:7.91197	valid-rmse:9.09264	train-r2:0.61358
	valid-r2:0.46883
[430]	train-rmse:7.88631	valid-rmse:9.08801	train-r2:0.61608
	valid-r2:0.46937
[440]	train-rmse:7.86229	valid-rmse:9.08692	train-r2:0.61842
	valid-r2:0.46950
[450]	train-rmse:7.84281	valid-rmse:9.08864	train-r2:0.62030
	valid-r2:0.46930
[460]	train-rmse:7.81202	valid-rmse:9.08876	train-r2:0.62328
	valid-r2:0.46929
[470]	train-rmse:7.79307	valid-rmse:9.09091	train-r2:0.62511
	valid-r2:0.46903
[480]	train-rmse:7.77274	valid-rmse:9.08829	train-r2:0.62706
	valid-r2:0.46934
[490]	train-rmse:7.75118	valid-rmse:9.08790	train-r2:0.62912
	valid-r2:0.46939
Stopping. Best iteration:
[440]	train-rmse:7.86229	valid-rmse:9.08692	train-r2:0.61842
	valid-r2:0.46950
```

```
# Predicting on test set
#p_test = P.predict(f_test)
```

```
p_test = P.predict(f_valid)
p_test
```

```
array([ 91.85558 ,  97.50537 , 102.70698 ,  79.37516 , 111.320755,
       102.339066,  92.144485, 102.70827 , 103.44181 , 114.60359 ,
        76.99464 ,  96.385056,  97.14054 , 103.14673 ,  96.2921  ,
        95.68799 , 109.41416 ,  96.91658 ,  95.25844 , 115.90761 ,
       114.28063 ,  97.4943  ,  95.48847 , 101.07218 ,  93.341835,
       110.99494 ,  95.407455,  77.939285,  93.5904  ,  94.38755 ,
        94.84346 , 102.11375 ,  97.09706 , 108.986946,  98.45447 ,
       113.27697 , 112.7254  ,  99.055725,  92.73723 ,  98.637   ,
       114.42097 , 101.52786 , 120.29243 , 108.7854  ,  96.14613 ,
       102.09786 ,  91.4691  , 104.28223 , 108.97863 , 104.38734 ,
        94.79384 ,  99.40892 , 103.92318 , 106.98751 ,  99.94386 ,
       101.344666,  98.6014  , 111.78628 ,  95.936844,  97.43675 ,
       109.23045 ,  76.58494 ,  95.13359 ,  95.561714,  77.84109 ,
        98.68572 ,  94.57013 , 100.45195 , 104.38734 ,  99.61501 ,
        93.91869 ,  95.01966 ,  98.889496, 105.98683 ,  95.92729 ,
       108.371414,  96.32246 , 109.20667 ,  95.42426 ,  98.5875  ,
       109.22089 , 117.89089 , 106.72527 , 111.09502 , 108.92924 ,
        98.19809 ,  97.401146, 103.04292 , 100.33582 , 104.10499 ,
        94.87488 , 102.02375 ,  95.49326 , 105.21229 , 109.70898 ,
        94.0044  , 100.922226, 110.85982 , 102.60057 ,  94.29006 ,
        78.14933 , 102.64957 ,  98.69211 ,  91.8857  , 100.82127 ,
       101.836586,  96.4268  ,  98.53219 , 107.0281  ,  95.53085 ,
       102.64957 ,  95.75728 , 102.46631 , 102.37315 , 100.10598 ,
        77.15717 ,  99.64988 ,  94.189186, 103.29069 , 112.501114,
       106.302666, 115.103584,  95.80905 , 110.39445 , 100.302635,
        97.17231 , 101.559685, 115.09893 ,  94.98706 , 100.90238 ,
       108.098755, 101.62986 ,  76.52447 ,  98.52138 , 110.73741 ,
       110.39218 , 102.73494 , 101.22333 , 110.85009 ,  95.39151 ,
        82.93532 , 104.8198  , 101.90289 , 116.488106,  97.16458 ,
        92.66883 , 107.72949 , 107.69694 , 111.468704,  97.39219 ,
       110.38511 ,  96.27474 ,  96.64371 , 109.514946, 117.521774,
       114.38039 ,  97.45502 ,  97.397354, 101.54086 , 107.12312 ,
        98.695625,  96.28677 ,  93.707794,  91.74429 , 107.21378 ,
        95.82833 ,  97.214714, 109.08186 , 121.81138 , 102.52998 ,
        76.57415 , 101.0845  ,  92.2674  , 100.71167 ,  97.69591 ,
       116.7312  ,  95.84302 ,  93.76442 , 102.427956, 105.527016,
       105.7015  , 101.5866  , 112.26593 ,  93.87154 , 110.80137 ,
        94.39677 , 105.71712 , 102.63193 , 100.384155, 102.32334 ,
       110.16298 , 115.34633 ,  93.634766, 107.613846,  91.80659 ,
        99.13333 , 103.58002 ,  96.068436, 102.628975, 110.16155 ,
        94.958664, 106.061424, 105.202705,  96.837166,  92.906265,
       107.70457 , 102.532936, 105.17211 ,  98.79227 ,  96.225555,
        96.570335,  91.8857  ,  95.42773 ,  95.92729 ,  93.29413 ,
        94.64124 ,  94.10279 ,  94.81019 , 105.236755, 102.70303 ,
       111.0121  ,  92.95905 , 102.460945,  97.12505 , 113.33663 ,
        96.23158 ,  93.96427 , 112.2637  , 106.56167 ,  98.613686,
        93.3675  ,  90.7346  ,  98.42906 ,  96.85121 , 106.82195 ,
```

```
101.53748 ,  93.8514  , 107.700615,  97.27749 ,  92.56199 ,
 79.66898 , 103.41554 , 111.98739 , 100.602356,  93.87208 ,
 93.936905,  94.2041  , 104.14163 ,  95.594894,  97.4229  ,
101.12293 , 108.86345 ,  96.10279 ,  96.97214 , 100.926384,
 94.6983  ,  95.568146, 105.61702 , 117.59253 , 107.36138 ,
 99.07186 , 104.22207 , 109.09889 ,  95.6651  , 106.98821 ,
111.77274 ,  95.02549 ,  88.61241 , 114.55637 ,  95.45047 ,
102.19536 ,  92.8272  , 108.77307 , 107.24565 , 106.67428 ,
104.4831  , 106.04442 ,  93.58357 ,  97.200226,  96.916115,
 99.99289 ,  96.52078 , 101.02474 ,  97.155205,  77.14423 ,
 95.87607 ,  94.40219 , 103.96056 ,  96.22355 , 111.295555,
 91.04291 , 103.24901 ,  93.30562 , 112.638176,  96.112236,
107.6754  ,  83.02968 ,  94.22389 , 106.853325, 110.08473 ,
110.368866, 106.18687 ,  98.79227 ,  99.320984, 104.82877 ,
 97.80305 , 102.63509 , 102.39297 ,  93.87208 ,  94.89134 ,
 97.346275,  94.7206  ,  97.11068 , 109.39842 , 101.56625 ,
 96.465324,  95.17433 , 100.05555 ,  95.89367 ,  97.31106 ,
117.193756,  94.54051 , 111.13186 ,  94.60285 ,  81.55996 ,
106.72527 ,  94.95556 , 107.00605 , 117.72414 , 107.38704 ,
118.08141 ,  95.07006 , 110.28834 , 106.058914,  97.60654 ,
 96.63401 , 107.56994 ,  97.508156, 112.07478 , 104.5048  ,
108.03882 , 102.50717 ,  95.990585,  94.10842 , 100.98409 ,
102.63509 , 111.46459 ,  94.09171 , 109.7578  , 110.83727 ,
 94.21215 , 106.974785,  78.582405, 110.48115 ,  95.27944 ,
109.48794 ,  96.05805 , 107.34061 , 110.085884,  95.81803 ,
 95.48724 , 110.709656, 107.86801 ,  98.1037  ,  96.97214 ,
 92.27094 , 106.962685,  93.548386, 103.149536, 103.99972 ,
 94.35577 ,  99.59681 , 101.905594,  95.49229 , 111.501396,
117.91235 ,  92.61647 ,  97.77454 , 103.45969 ,  96.03507 ,
106.0349  ,  94.76763 , 102.018326, 111.99961 , 106.170395,
 95.757774,  95.76722 , 100.22616 , 109.866684,  99.37948 ,
103.20925 , 117.79436 , 103.14673 , 107.66808 ,  91.345566,
101.40821 , 101.87944 , 103.364   ,  93.30625 ,  99.12608 ,
 76.537506, 106.04565 , 113.78939 , 109.22427 ,  95.66489 ,
110.19445 , 110.3489  , 105.28847 ,  77.935616, 102.78185 ,
 97.21264 ,  95.40284 , 101.07593 , 110.11104 ,  98.265656,
109.02044 , 102.63509 , 100.74599 , 108.90718 , 106.01121 ,
104.311584, 104.29131 ,  96.52813 , 109.24212 , 109.70096 ,
111.00059 , 117.223595, 103.32866 ,  94.58152 , 109.06191 ,
103.9572  , 101.72384 ,  96.54322 , 112.29871 , 109.537506,
101.25517 , 102.836044, 102.70303 , 108.3217  ,  94.34375 ,
105.80636 ,  96.61419 ,  95.04985 , 108.02088 ,  96.532875,
 93.760956,  99.14684 , 103.38526 , 109.20667 ,  94.375336,
 98.0144  ,  93.641624,  94.87989 ,  92.27227 , 104.75042 ,
 98.4744  , 104.82089 ,  99.02231 ,  96.380615,  97.487595,
 95.83749 , 102.26186 ,  99.470825, 100.29973 , 110.16588 ,
117.41115 ,  94.03323 , 108.59853 , 105.77899 ,  77.03346 ,
102.63509 ,  98.94946 , 104.89299 ,  96.229095, 110.18274 ,
105.8386  ,  91.37952 ,  96.7148  , 110.981766,  93.952576,
109.20667 , 110.12664 ,  79.711685, 100.994354, 101.21623 ,
```

```
 94.56766 , 111.761314,  98.19577 ,  94.10279 ,  99.729454,
101.29741 , 107.30902 ,  99.04891 , 109.71976 ,  97.33095 ,
 78.54539 ,  79.78823 ,  92.42212 ,  94.267624,  99.32133 ,
 95.18132 ,  95.81287 ,  98.36931 , 103.1726  , 110.81212 ,
107.456085,  98.520134,  96.368034, 116.18915 ,  99.54396 ,
 96.38409 , 100.502754,  99.25509 , 102.231186, 102.37315 ,
 78.18116 , 110.3489  ,  95.190926,  93.07379 ,  77.20119 ,
 95.85703 ,  96.652145,  94.30946 , 117.48577 , 110.98813 ,
109.20667 ,  96.35307 ,  95.67835 ,  92.741905, 104.57382 ,
104.01274 ,  96.4258  ,  96.34631 ,  96.04879 ,  98.79243 ,
117.51672 , 102.30724 ,  96.56204 ,  94.62459 ,  98.519516,
 77.664536, 104.60583 ,  95.01087 , 115.05749 ,  96.4351  ,
109.73364 , 113.06056 ,  94.361404, 108.48341 , 113.343956,
101.07895 ,  94.95062 ,  93.68007 ,  96.03451 , 105.66482 ,
 91.06434 , 102.46136 , 106.71189 , 104.20508 , 105.62261 ,
114.64228 , 101.2715  , 110.871704, 109.26564 , 109.29758 ,
 99.045456,  99.939354, 101.516304, 101.49074 , 109.36143 ,
 94.343956,  91.984825, 106.38472 ,  96.412315,  93.84593 ,
100.082924, 117.89089 , 110.19769 ,  97.38657 , 114.51076 ,
 95.29584 , 109.47056 , 108.274445, 109.057846, 101.422325,
104.64702 , 104.57019 ,  98.2198  , 105.40497 , 100.16604 ,
101.4192  , 108.97925 ,  79.02938 , 100.11315 , 100.00532 ,
110.44855 ,  95.90619 ,  77.560074,  96.53722 ,  94.87623 ,
102.93176 ,  95.78184 ,  96.3307  ,  94.128296,  95.87109 ,
 94.13697 , 101.72114 ,  77.02893 , 101.21021 ,  92.13636 ,
 95.479706,  80.76953 , 101.53239 , 107.50124 , 109.69633 ,
109.38389 ,  95.183304,  96.30451 , 105.35168 , 100.635765,
109.21143 ,  93.94617 ,  78.08442 , 111.54554 ,  98.30948 ,
 98.434654, 107.51979 , 108.10825 , 108.33175 ,  96.37342 ,
101.68723 ,  94.07123 ,  96.258125,  99.94866 , 115.34821 ,
101.90131 , 110.83012 ,  96.86146 , 111.82233 , 100.887085,
102.02109 ,  81.88424 , 110.58971 ,  96.578   , 110.40388 ,
114.8453  , 111.69764 ,  95.37132 , 102.17221 , 101.38562 ,
109.34327 , 111.353165,  99.59981 ,  92.27821 ,  77.823395,
111.58834 , 111.98739 , 103.98452 ,  95.51238 , 110.65374 ,
 94.708755, 102.60997 ,  97.23124 , 103.98121 ,  94.10279 ,
107.182655, 138.93637 , 112.93068 , 109.50405 , 106.68418 ,
102.5479  ,  96.634026,  81.79001 ,  95.20647 ,  92.012024,
 78.03501 , 103.77766 , 107.579735, 103.413864, 100.842735,
 99.33579 ,  92.82497 , 102.437126, 106.54306 , 103.12184 ,
 95.33187 ,  97.79969 , 107.59132 ,  96.92974 ,  95.56709 ,
101.32017 ,  95.551346,  94.64498 ,  77.48836 ,  94.34595 ,
110.65319 , 103.80441 , 107.36305 , 108.603745, 101.77321 ,
 94.002205,  96.679695,  98.81825 , 101.4286  ,  95.35539 ,
116.459694,  93.70944 ,  97.39219 , 100.117744,  97.9186  ,
109.98073 , 103.96337 , 108.807686, 103.56997 , 118.400925,
 93.44059 ,  93.42728 ,  95.31778 ,  91.73611 , 106.31323 ,
112.173965, 100.858795,  97.08692 ,  94.64932 ,  81.46661 ,
 93.53678 , 109.081505,  95.43176 , 105.79155 , 106.49624 ,
 99.74437 ,  98.14773 ,  98.82344 ,  78.06412 ,  93.50896 ,
```

```
        95.69548 ,  96.911385, 102.27005 ,  99.46616 , 109.60735 ,
       105.23254 ,  94.86082 ,  95.280106,  99.10615 , 109.70096 ,
        94.267624, 105.497665, 112.84933 ,  97.22007 ,  76.62243 ,
        95.08418 ,  95.71439 , 101.59704 , 103.94387 ,  97.664345,
        94.80686 ,  95.84566 ,  96.970955, 101.44937 , 102.56442 ,
        95.94973 , 103.519966,  79.44671 ,  94.11146 ,  99.52569 ,
       107.253174, 103.45493 ,  93.64485 , 114.77355 , 102.63509 ,
       104.47719 ,  97.69116 , 117.59253 ,  95.29654 , 109.13109 ,
       104.57356 ,  97.73969 ,  97.30285 ,  95.84369 ,  97.44502 ,
        92.83761 ,  99.95926 ,  95.895256, 106.86136 , 105.2415  ,
        99.65676 , 109.149025,  77.558464,  96.564354,  97.13024 ,
       104.012886, 103.04258 ,  93.738014, 111.04924 ,  93.48747 ,
       100.32133 ,  94.586136, 100.126755,  96.84464 , 111.32277 ,
        95.10032 ,  93.1551  ,  98.41308 , 101.935455, 112.13678 ,
        92.696754,  94.42393 ,  95.154045,  94.98658 ,  95.41863 ,
       112.58325 , 103.30814 ,  97.776825,  95.19152 , 105.61461 ,
        98.0916  , 113.66568 ,  91.90936 ,  97.99352 , 102.845764,
       108.65494 , 102.5479  ,  98.394875,  98.51077 ,  79.042336,
        95.13753 ,  97.56195 ,  77.20848 ,  95.419754,  95.68616 ,
        93.046555, 103.48766 ,  93.6168  , 107.9781  , 107.58211 ,
        99.02246 , 103.31566 , 104.72804 ,  92.929924,  97.6136  ,
        93.94617 , 106.00153 ], dtype=float32)
```

```
Predicted_Data = pd.DataFrame()
Predicted_Data['y'] = p_test
Predicted_Data.head()
```

```
            y
0   91.855583
1   97.505371
2  102.706978
3   79.375160
4  111.320755
```