MercedesBenz_Project

July 26, 2022

[1]:
```
# 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
```

[2]:
```
# Importing the required libraries
# Loading the train/test data
# The lowercase alphabets are categorical variables
import numpy as np
import pandas as pd

train = pd.read_csv('train.csv')
train.head()
```

[2]:
```
   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]
```

[3]:
```python
print('Size of training set')
print(train.shape)
```

```
Size of training set
(4209, 378)
```

[4]:
```python
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
```

[5]:
```python
# Separating y column as this is for prediction output
y_train = train['y'].values
y_train
```

[5]:
```
array([130.81,  88.53,  76.26, …, 109.22,  87.48, 110.85])
```

[6]:
```python
# A lot of columns that have an X
# Let's check for the same
# 376 features with X
colums_x = [c for c in train.columns if 'X' in c]

# info about colums_x
print(len(colums_x))
print(train[colums_x].dtypes.value_counts())
```

```
376
int64     368
object      8
dtype: int64
```

[7]:
```python
# Looking at the test dataset for simiilar features
test = pd.read_csv('test.csv')
test.head()
```

[7]:
```
   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]
```

[8]:

```
print('Size of training set')
test.shape
```

```
Size of training set
```

[8]:

```
(4209, 377)
```

[9]:

```
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
```

[10]:

```
# Creating the final dataset
# Removing unwanted columns (ID); y has been removed earlier
final_column = list(set(train.columns) - set(['ID', 'y']))

x_train = train[final_column]
# x_train
x_test = test[final_column]
# x_test
```

[11]:

```
# Searching for null values
# Creating a function for the same
def detect(df):
    if df.isnull().any().any():
        print("Yes")
    else:
        print("No")

detect(x_train)
detect(x_test)

# Observation : There are no missing values.
```

```
No
No
```

[12]:
```python
## 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 : ",unique_val)
    print(columns)
    print("-----------------------------------------------------------")
```

```
Columns containing the unique values :  [0, 1]
['X247', 'X51', 'X274', 'X270', 'X219', 'X148', 'X116', 'X236', 'X186', 'X23',
'X33', 'X35', 'X382', 'X47', 'X106', 'X134', 'X18', 'X70', 'X171', 'X359',
'X80', 'X374', 'X29', 'X288', 'X253', 'X131', 'X273', 'X178', 'X189', 'X168',
'X91', 'X32', 'X252', 'X28', 'X105', 'X136', 'X282', 'X311', 'X125', 'X153',
'X64', 'X95', 'X205', 'X320', 'X292', 'X266', 'X38', 'X368', 'X90', 'X159',
'X351', 'X169', 'X225', 'X96', 'X98', 'X27', 'X109', 'X142', 'X94', 'X151',
'X334', 'X37', 'X342', 'X209', 'X162', 'X196', 'X369', 'X13', 'X129', 'X57',
'X185', 'X104', 'X256', 'X267', 'X315', 'X81', 'X319', 'X158', 'X302', 'X316',
'X226', 'X190', 'X241', 'X211', 'X231', 'X69', 'X141', 'X77', 'X140', 'X56',
'X363', 'X146', 'X321', 'X357', 'X276', 'X76', 'X114', 'X175', 'X326', 'X337',
'X191', 'X379', 'X88', 'X305', 'X103', 'X350', 'X294', 'X31', 'X370', 'X366',
'X281', 'X317', 'X345', 'X145', 'X39', 'X237', 'X318', 'X135', 'X338', 'X258',
'X71', 'X376', 'X97', 'X265', 'X223', 'X310', 'X234', 'X179', 'X214', 'X248',
'X41', 'X216', 'X372', 'X122', 'X312', 'X177', 'X62', 'X155', 'X110', 'X257',
'X238', 'X333', 'X172', 'X242', 'X201', 'X124', 'X383', 'X215', 'X17', 'X157',
'X164', 'X255', 'X115', 'X61', 'X298', 'X206', 'X24', 'X324', 'X355', 'X296',
'X46', 'X68', 'X112', 'X239', 'X356', 'X182', 'X240', 'X181', 'X15', 'X137',
'X250', 'X26', 'X43', 'X275', 'X380', 'X360', 'X120', 'X307', 'X84', 'X354',
'X246', 'X143', 'X74', 'X378', 'X224', 'X54', 'X144', 'X21', 'X130', 'X284',
'X244', 'X367', 'X117', 'X108', 'X385', 'X204', 'X65', 'X87', 'X279', 'X286',
'X52', 'X111', 'X308', 'X50', 'X287', 'X42', 'X243', 'X127', 'X180', 'X227',
'X304', 'X249', 'X352', 'X60', 'X79', 'X85', 'X329', 'X183', 'X309', 'X228',
'X332', 'X278', 'X207', 'X49', 'X328', 'X99', 'X384', 'X220', 'X322', 'X377',
'X161', 'X341', 'X371', 'X53', 'X362', 'X335', 'X339', 'X82', 'X198', 'X327',
'X331', 'X259', 'X16', 'X277', 'X353', 'X89', 'X138', 'X14', 'X199', 'X269',
'X222', 'X343', 'X150', 'X365', 'X344', 'X314', 'X194', 'X260', 'X261', 'X192',
'X73', 'X163', 'X102', 'X36', 'X167', 'X285', 'X200', 'X48', 'X325', 'X221',
'X86', 'X20', 'X92', 'X280', 'X336', 'X165', 'X34', 'X75', 'X212', 'X154',
'X361', 'X119', 'X10', 'X218', 'X101', 'X262', 'X128', 'X195', 'X78', 'X203',
'X58', 'X22', 'X174', 'X210', 'X349', 'X272', 'X66', 'X295', 'X291', 'X263',
'X358', 'X139', 'X63', 'X113', 'X300', 'X348', 'X364', 'X301', 'X373', 'X132',
'X173', 'X208', 'X299', 'X123', 'X232', 'X170', 'X184', 'X271', 'X147', 'X67',
```

```
'X166', 'X323', 'X44', 'X126', 'X213', 'X176', 'X346', 'X156', 'X264', 'X59',
'X202', 'X217', 'X251', 'X40', 'X197', 'X230', 'X83', 'X313', 'X30', 'X152',
'X245', 'X19', 'X375', 'X133', 'X187', 'X100', 'X118', 'X340', 'X45', 'X229',
'X55', 'X12', 'X254', 'X283', 'X306', 'X160']
------------------------------------------------------------
Columns containing the unique values :  [0]
['X235', 'X290', 'X233', 'X107', 'X268', 'X289', 'X293', 'X93', 'X347', 'X11',
'X330', 'X297']
------------------------------------------------------------
```

[13]:
```
# Removal of columns with a variance of 0
# means columns that have only one unique value 0.

for column in final_column:
    check = len(np.unique(x_train[column]))
    if check == 1:
        x_train.drop(column, axis = 1, inplace=True)
        x_test.drop(column, axis = 1, inplace=True)

x_train.head()
```

```
/usr/local/lib/python3.7/site-packages/pandas/core/frame.py:4174:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```

[13]:

	X247	X51	X274	X270	X219	X148	X116	X236	X4	X186	…	X340	X45	\
0	0	0	0	0	0	0	1	0	d	0	…	0	0	
1	0	1	0	0	0	0	0	0	d	0	…	0	0	
2	0	1	1	0	1	1	0	0	d	0	…	0	0	
3	0	0	0	0	0	1	0	0	d	0	…	0	0	
4	0	1	0	0	0	1	0	0	d	0	…	0	0	

	X229	X55	X12	X254	X283	X1	X306	X160
0	0	0	0	0	0	v	1	0
1	1	0	0	0	0	t	0	0
2	0	0	0	0	0	w	0	0
3	1	0	0	0	0	t	0	0
4	1	0	0	0	0	v	0	0

[5 rows x 364 columns]

[14]:
```
## Label encoding the Categorical columns
from sklearn import preprocessing
for f in ["X0", "X1", "X2", "X3", "X4", "X5", "X6", "X8"]:
```

```
        lbl = preprocessing.LabelEncoder()
        lbl.fit(list(x_train[f].values))
        x_train[f] = lbl.transform(list(x_train[f].values))
        #x_test[f] = lbl.transform(list(x_test[f].values))   ## as values in␣
↪test dataset differs from train set
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

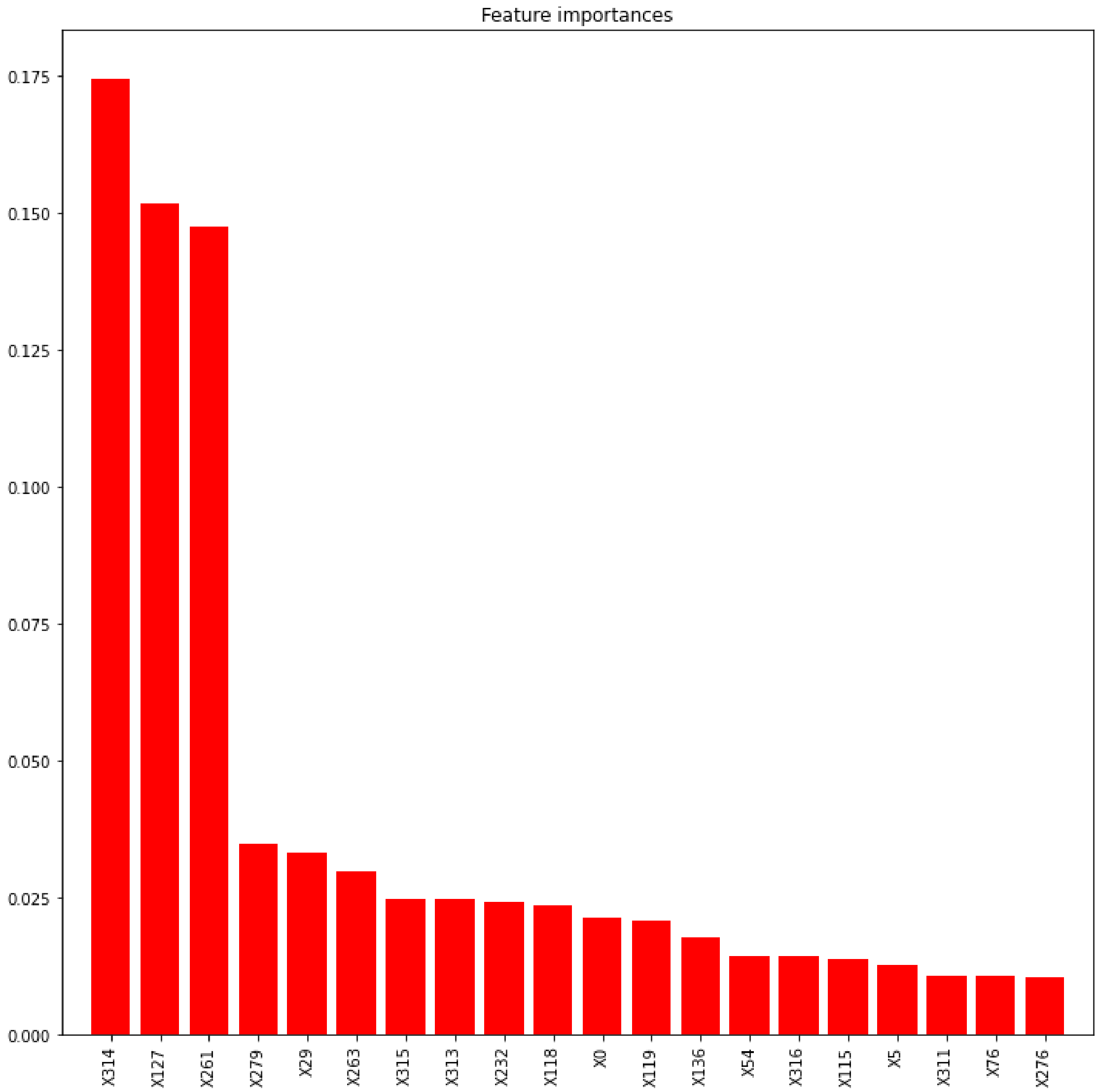
[15]:

```
## Let us build a Random Forest model and check the important variables.

from sklearn import ensemble
model = ensemble.RandomForestRegressor(n_estimators=200,
                                       max_depth=10, min_samples_leaf=4,
                                       max_features=0.2, n_jobs=-1,
                                       random_state=0)
model.fit(x_train, y_train)
feat_names = x_train.columns.values

## plot the importances ##
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0)
indices = np.argsort(importances)[::-1][:20]

import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
plt.title("Feature importances")
plt.bar(range(len(indices)), importances[indices], color="r", align="center")
plt.xticks(range(len(indices)), feat_names[indices], rotation='vertical')
plt.xlim([-1, len(indices)])
plt.show()
```



[16]:
```
# Performing dimensionality reduction with principal components analysis
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)
```

[17]:
```
# 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)
```

[18]:
```
# Building the final feature set
f_train = xgb.DMatrix(x_train, label = y_train)
f_valid = xgb.DMatrix(x_valid, label = y_valid)
```

[19]:
```
# Setting the parameters for XGB
params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.02  ## eta means learning rate
params['max_depth'] = 4
```

[20]:
```
# Predicting the score
# Creating a function for the same

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)
```

```
[15:15:22] 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:45.14307     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:26.02783     valid-rmse:26.04540     train-r2:-3.18185
valid-r2:-3.35826
[80]    train-rmse:22.00439     valid-rmse:22.02672     train-r2:-1.98890
valid-r2:-2.11710
[90]    train-rmse:18.81535     valid-rmse:18.85042     train-r2:-1.18533
valid-r2:-1.28293
[100]   train-rmse:16.31674     valid-rmse:16.36231     train-r2:-0.64346
valid-r2:-0.72005
[110]   train-rmse:14.38474     valid-rmse:14.45036     train-r2:-0.27731
valid-r2:-0.34156
[120]   train-rmse:12.90109     valid-rmse:12.99663     train-r2:-0.02741
valid-r2:-0.08521
[130]   train-rmse:11.79115     valid-rmse:11.91737     train-r2:0.14177
valid-r2:0.08754
[140]   train-rmse:10.96575     valid-rmse:11.12483     train-r2:0.25772
valid-r2:0.20487
[150]   train-rmse:10.35605     valid-rmse:10.55144     train-r2:0.33797
valid-r2:0.28472
[160]   train-rmse:9.89863      valid-rmse:10.13802     train-r2:0.39516
valid-r2:0.33968
[170]   train-rmse:9.56053      valid-rmse:9.84641      train-r2:0.43577
valid-r2:0.37712
[180]   train-rmse:9.30283      valid-rmse:9.63360      train-r2:0.46578
valid-r2:0.40375
[190]   train-rmse:9.11631      valid-rmse:9.48747      train-r2:0.48698
valid-r2:0.42170
[200]   train-rmse:8.96881      valid-rmse:9.38373      train-r2:0.50345
valid-r2:0.43428
[210]   train-rmse:8.86698      valid-rmse:9.31308      train-r2:0.51466
valid-r2:0.44277
[220]   train-rmse:8.76993      valid-rmse:9.25871      train-r2:0.52523
valid-r2:0.44925
[230]   train-rmse:8.69552      valid-rmse:9.21908      train-r2:0.53325
valid-r2:0.45396
[240]   train-rmse:8.62931      valid-rmse:9.19054      train-r2:0.54033
valid-r2:0.45733
[250]   train-rmse:8.55097      valid-rmse:9.16471      train-r2:0.54864
valid-r2:0.46038
[260]   train-rmse:8.48003      valid-rmse:9.14429      train-r2:0.55610
valid-r2:0.46278
[270]   train-rmse:8.41509      valid-rmse:9.13383      train-r2:0.56287
valid-r2:0.46401
[280]   train-rmse:8.36279      valid-rmse:9.12279      train-r2:0.56829
valid-r2:0.46531
[290]   train-rmse:8.30989      valid-rmse:9.11852      train-r2:0.57373
valid-r2:0.46581
[300]   train-rmse:8.26866      valid-rmse:9.11413      train-r2:0.57795
valid-r2:0.46632
```

```
[310]	train-rmse:8.23677	valid-rmse:9.11099	train-r2:0.58120
valid-r2:0.46669
[320]	train-rmse:8.20386	valid-rmse:9.10859	train-r2:0.58454
valid-r2:0.46697
[330]	train-rmse:8.17460	valid-rmse:9.10561	train-r2:0.58750
valid-r2:0.46732
[340]	train-rmse:8.14394	valid-rmse:9.10432	train-r2:0.59059
valid-r2:0.46747
[350]	train-rmse:8.11652	valid-rmse:9.10443	train-r2:0.59334
valid-r2:0.46746
[360]	train-rmse:8.08833	valid-rmse:9.10303	train-r2:0.59616
valid-r2:0.46762
[370]	train-rmse:8.05940	valid-rmse:9.10149	train-r2:0.59904
valid-r2:0.46780
[380]	train-rmse:8.03252	valid-rmse:9.10307	train-r2:0.60171
valid-r2:0.46761
[390]	train-rmse:8.00597	valid-rmse:9.10491	train-r2:0.60434
valid-r2:0.46740
[400]	train-rmse:7.97740	valid-rmse:9.10658	train-r2:0.60716
valid-r2:0.46720
[410]	train-rmse:7.94743	valid-rmse:9.10668	train-r2:0.61011
valid-r2:0.46719
Stopping. Best iteration:
[366]	train-rmse:8.07108	valid-rmse:9.10038	train-r2:0.59788
valid-r2:0.46793
```

[21]:
```
# Predicting on test set
p_test = P.predict(f_valid)
p_test
```

[21]:
```
array([ 92.60499 ,  96.806854, 102.80264 ,  79.457   , 111.14049 ,
       101.756035,  92.88736 , 102.632   , 102.81461 , 114.01289 ,
        77.04445 ,  96.07492 ,  96.87875 , 103.35972 ,  96.36682 ,
        95.68255 , 109.60038 ,  97.138   ,  95.19734 , 115.63238 ,
       112.34259 ,  98.16956 ,  96.117035, 101.5805  ,  93.74107 ,
       111.3088  ,  96.555145,  78.199066,  93.47996 ,  94.52608 ,
        94.96783 , 102.384865,  97.06934 , 109.00841 ,  98.18227 ,
       113.868645, 113.02013 ,  99.33244 ,  92.98425 ,  99.22035 ,
       112.89446 , 101.96907 , 118.01389 , 108.41425 ,  96.2537  ,
       102.12306 ,  91.533966, 103.979225, 109.84129 , 104.8943  ,
        94.53346 ,  98.731186, 103.45173 , 107.13904 , 100.23186 ,
       101.31478 ,  98.88241 , 111.5741  ,  95.82149 ,  97.51418 ,
       109.04537 ,  76.7217  ,  95.34523 ,  95.91618 ,  77.54216 ,
        98.24471 ,  95.23625 ,  99.64998 , 104.8943  ,  99.94928 ,
        94.26675 ,  94.351074,  99.41977 , 105.952446,  96.03121 ,
       108.86716 ,  96.47004 , 109.09719 ,  95.72579 ,  99.09363 ,
```

```
108.72874 , 117.66448 , 106.68439 , 110.78265 , 108.83856 ,
 98.39304 ,  97.13509 , 103.70876 ,  98.49357 , 104.89489 ,
 94.92148 , 102.433586,  95.284744, 105.138535, 109.74104 ,
 94.812805, 100.75868 , 111.16399 , 103.0544  ,  94.497765,
 78.5258  , 102.45562 ,  99.230965,  92.57083 , 101.124054,
101.91825 ,  96.316055,  98.90676 , 107.06945 ,  96.074394,
102.45562 ,  95.63763 , 102.51549 , 101.66769 ,  99.65106 ,
 77.28205 ,  99.273796,  94.42956 , 102.007095, 112.44842 ,
105.776665, 114.93352 ,  96.00163 , 110.14417 ,  99.79223 ,
 96.56924 , 100.93587 , 114.52954 ,  95.10923 , 101.00649 ,
107.68371 , 102.12306 ,  76.61551 ,  98.64229 , 111.281006,
109.60007 , 101.88947 , 101.57508 , 110.759575,  95.25144 ,
 82.639145, 104.33289 , 101.9817  , 116.68952 ,  97.58357 ,
 92.279816, 107.85746 , 107.66956 , 110.9162  ,  96.62174 ,
110.18566 ,  96.83927 ,  96.917404, 109.692894, 116.56731 ,
113.62168 ,  97.665665,  97.37928 , 101.61786 , 107.0307  ,
 97.47725 ,  96.110756,  94.184166,  92.455284, 107.142975,
 95.826996,  96.83472 , 109.056984, 118.817314, 102.72384 ,
 76.56015 , 100.98559 ,  92.45786 , 100.42722 ,  97.85391 ,
116.10634 ,  95.507935,  93.90122 , 102.80341 , 105.80255 ,
105.746185, 101.86515 , 112.50821 ,  93.800095, 110.90007 ,
 94.73847 , 106.01695 , 102.63345 , 100.028755, 102.216515,
108.76231 , 114.413284,  94.14479 , 107.39243 ,  91.971825,
 99.65048 , 103.64251 ,  96.04788 , 103.05668 , 109.92132 ,
 94.73317 , 106.17988 , 105.25002 ,  96.62253 ,  94.135315,
107.21151 , 102.728615, 105.09033 ,  98.52006 ,  96.17434 ,
 96.182625,  92.57083 ,  95.79801 ,  96.03121 ,  93.173935,
 94.941795,  94.3206  ,  94.79936 , 105.69819 , 102.91301 ,
111.1001  ,  93.240685, 102.095665,  97.29785 , 112.67492 ,
 96.26714 ,  94.25467 , 111.174065, 106.44342 ,  99.227806,
 93.53982 ,  91.19891 ,  98.85083 ,  96.54218 , 106.33364 ,
101.00411 ,  94.12351 , 107.78963 ,  97.720764,  93.274826,
 79.58499 , 103.12054 , 110.43233 , 100.438805,  94.42433 ,
 93.875694,  93.13608 , 103.13305 ,  95.14386 ,  96.952034,
101.1275  , 108.764114,  96.46538 ,  97.4495  , 100.90388 ,
 95.02394 ,  95.99068 , 104.92516 , 117.43134 , 107.55676 ,
 98.011536, 104.36592 , 109.618576,  95.64919 , 106.78103 ,
111.78899 ,  95.20972 ,  88.81448 , 114.266815,  95.770226,
101.97147 ,  92.95514 , 108.24426 , 107.576515, 107.13346 ,
103.36699 , 105.441696,  93.961975,  96.81793 ,  97.09693 ,
 99.38764 ,  96.81865 , 100.38637 ,  97.77793 ,  77.16998 ,
 95.8964  ,  94.62029 , 103.5899  ,  96.17194 , 111.1578  ,
 91.40179 , 103.33011 ,  93.56026 , 112.5711  ,  96.151505,
107.523125,  83.05226 ,  94.77419 , 107.08587 , 108.76528 ,
109.848595, 106.26511 ,  98.52006 ,  99.1994  , 104.918015,
 97.881134, 101.918594, 102.710464,  94.42433 ,  94.54003 ,
 97.6235  ,  94.61863 ,  97.12267 , 108.73693 , 101.805855,
```

```
 97.30874 ,  95.033806, 100.09829 ,  95.522255,  97.49251 ,
117.13099 ,  95.06454 , 110.863106,  94.64607 ,  81.398155,
106.68439 ,  94.99    , 107.16246 , 116.76684 , 107.31885 ,
117.75343 ,  94.783295, 109.83969 , 105.79001 ,  98.02176 ,
 96.75551 , 107.085075,  97.45835 , 111.66608 , 104.91228 ,
106.97887 , 102.60011 ,  96.08345 ,  93.56558 , 100.3004  ,
101.918594, 111.182594,  94.33023 , 108.92563 , 110.388954,
 94.26794 , 106.94248 ,  78.52514 , 110.40091 ,  95.190765,
109.46803 ,  96.32045 , 106.115715, 109.989   ,  95.62205 ,
 95.70654 , 110.859215, 107.60158 ,  98.03645 ,  97.4495  ,
 92.58775 , 105.70897 ,  93.7164  , 102.61456 , 103.99519 ,
 94.49149 ,  99.6547  , 101.93422 ,  95.12822 , 111.51089 ,
117.5905  ,  92.609184,  97.65683 , 103.29891 ,  96.29402 ,
106.24778 ,  94.8827  , 102.47209 , 111.50615 , 106.86135 ,
 96.4871  ,  96.18855 , 100.052284, 109.901535,  99.17396 ,
102.26955 , 117.59058 , 103.35972 , 106.80081 ,  91.72305 ,
101.4267  , 101.735985, 102.81364 ,  93.32291 ,  98.747505,
 76.7217  , 105.418015, 114.85297 , 109.23579 ,  95.796455,
109.74503 , 110.37343 , 105.1484  ,  78.18046 , 102.45483 ,
 97.36134 ,  95.59508 , 100.88766 , 109.84912 ,  98.237976,
108.69852 , 101.918594, 101.1375  , 108.751755, 105.76151 ,
104.20084 , 104.54192 ,  96.943596, 106.844505, 109.373825,
110.35304 , 117.0164  , 102.48308 ,  94.47586 , 110.274086,
103.97436 , 100.31314 ,  96.85753 , 111.79011 , 110.0536  ,
101.28586 , 102.84757 , 102.91301 , 108.65894 ,  94.528824,
105.344604,  97.09442 ,  95.62986 , 108.37576 ,  96.72114 ,
 93.67147 , 100.3934  , 102.81518 , 109.09719 ,  94.19344 ,
 98.40252 ,  94.0642  ,  95.57784 ,  91.89911 , 104.33391 ,
 98.67646 , 105.38797 ,  98.67716 ,  96.5153  ,  97.3941  ,
 95.663506, 102.63668 ,  99.29541 ,  99.12364 , 110.743416,
117.19045 ,  94.08307 , 108.11833 , 105.795784,  77.32129 ,
101.918594,  98.90704 , 105.541985,  97.09436 , 109.53955 ,
105.57941 ,  91.64876 ,  96.58269 , 111.323074,  94.65462 ,
109.09719 , 110.03216 ,  79.77929 , 101.44446 , 101.5275  ,
 94.59146 , 111.49257 ,  98.31125 ,  94.3206  ,  99.94442 ,
100.89789 , 109.64721 ,  99.23686 , 109.839096,  97.57621 ,
 77.96703 ,  79.77477 ,  92.65756 ,  94.14132 ,  99.20931 ,
 94.97722 ,  96.0842  ,  98.73771 , 102.53959 , 110.519005,
109.277306,  98.948006,  96.27841 , 115.43721 ,  99.65145 ,
 96.31542 , 100.748116,  97.641716, 101.92147 , 101.66769 ,
 78.35126 , 110.37343 ,  95.88282 ,  93.48639 ,  77.491   ,
 97.13072 ,  97.47415 ,  94.42645 , 117.5593  , 110.93521 ,
109.09719 ,  96.973976,  96.06538 ,  92.85539 , 104.52106 ,
104.15107 ,  96.5688  ,  96.30485 ,  95.83074 ,  98.75451 ,
117.681564, 102.580505,  97.66188 ,  94.65129 ,  98.33422 ,
 77.88335 , 104.42898 ,  95.247826, 114.644325,  96.7462  ,
110.007774, 112.564865,  94.65097 , 108.92167 , 114.41849 ,
```

```
101.03241 ,  94.86228 ,  94.105865,  96.16394 , 105.93023 ,
 91.252975, 102.237854, 106.320694, 103.27051 , 104.57418 ,
114.4089  , 100.58683 , 110.38667 , 109.24049 , 109.17287 ,
 99.489006, 100.46873 , 102.09853 , 101.82798 , 109.50259 ,
 94.659775,  92.042   , 106.70591 ,  96.14943 ,  93.82716 ,
 99.97439 , 117.68501 , 110.28962 ,  97.40453 , 114.02887 ,
 95.60103 , 109.552574, 108.8436  , 109.62044 , 101.48364 ,
104.585526, 104.60771 ,  98.70513 , 104.94389 ,  99.74634 ,
101.09647 , 109.03579 ,  79.18411 ,  99.94783 , 100.44582 ,
110.8622  ,  95.9044  ,  77.70651 ,  95.845276,  95.248474,
102.12246 ,  95.45547 ,  95.75336 ,  93.96617 ,  94.8679  ,
 94.23587 , 101.59536 ,  77.16653 , 101.51986 ,  92.24279 ,
 95.96856 ,  80.27081 ,  99.67621 , 107.433136, 109.71676 ,
108.93496 ,  95.06748 ,  96.93555 , 105.93957 , 100.67643 ,
109.43031 ,  94.170074,  78.24217 , 113.147964,  98.52272 ,
 97.67914 , 107.47106 , 107.72649 , 108.61602 ,  96.83306 ,
101.02607 ,  94.04835 ,  96.39166 , 100.54679 , 115.41624 ,
102.17683 , 110.98115 ,  97.77275 , 111.901535, 101.01382 ,
101.91378 ,  81.706154, 110.731285,  96.33363 , 110.369896,
113.17615 , 109.8532  ,  95.54546 , 102.63106 , 101.48854 ,
109.34651 , 110.98189 ,  99.80167 ,  93.128105,  77.58102 ,
109.71873 , 110.43233 , 103.27682 ,  95.9801  , 110.420525,
 95.5172  , 102.44727 ,  97.603806, 103.791046,  94.3206  ,
107.1778  , 138.02776 , 112.79421 , 109.86206 , 106.14948 ,
102.17059 ,  97.15741 ,  81.83285 ,  94.307434,  91.9639  ,
 78.158226, 103.68047 , 107.23172 , 103.58331 , 101.12069 ,
 99.58548 ,  93.04956 , 102.07898 , 106.38889 , 103.41811 ,
 94.89418 ,  97.54977 , 107.62267 ,  97.3356  ,  95.70692 ,
100.868454,  95.58384 ,  94.69769 ,  77.76693 ,  94.6152  ,
110.43732 , 103.56313 , 107.4855  , 108.05296 , 100.40644 ,
 94.18149 ,  97.00315 ,  98.636116, 101.500534,  95.29169 ,
116.53644 ,  93.96853 ,  96.62174 , 100.56817 ,  98.15233 ,
110.02552 , 103.74763 , 107.48445 , 103.583786, 118.17058 ,
 94.19589 ,  93.6197  ,  95.361885,  92.11123 , 105.84988 ,
113.31071 , 101.00748 ,  96.75222 ,  94.769585,  81.36063 ,
 94.25873 , 109.22102 ,  95.60379 , 105.92466 , 106.201485,
 99.93879 ,  98.57171 ,  98.15788 ,  78.06684 ,  93.620705,
 95.96218 ,  96.33714 , 102.83824 ,  98.79868 , 109.472595,
105.61224 ,  94.92382 ,  95.02763 ,  98.994514, 109.373825,
 94.14132 , 105.49076 , 112.51299 ,  97.07691 ,  76.851456,
 95.00128 ,  95.37197 , 100.77815 , 103.709404,  97.49661 ,
 94.69524 ,  95.63061 ,  96.890686, 101.604515, 102.5089  ,
 96.06702 , 102.801414,  79.59672 ,  94.04835 ,  99.080345,
106.89011 , 103.47778 ,  93.556885, 114.548225, 101.918594,
104.17367 ,  97.759605, 117.45187 ,  95.01565 , 109.318016,
105.645386,  97.222176,  96.58233 ,  95.56753 ,  97.194244,
 92.77588 ,  99.64506 ,  95.83796 , 106.19046 , 105.62774 ,
```

```
       100.88026 , 108.54349 ,  77.38707 ,  95.99978 ,  98.029175,
       104.235016, 103.11292 ,  93.61895 , 111.110695,  93.46543 ,
        99.4976  ,  94.60362 , 100.42038 ,  97.410126, 111.22598 ,
        95.14624 ,  93.41916 ,  98.61688 , 102.28138 , 111.39841 ,
        92.55669 ,  94.909676,  94.7862  ,  95.0441  ,  95.50709 ,
       112.80915 , 103.15393 ,  97.13871 ,  94.982346, 105.65416 ,
        98.115486, 113.096146,  92.26009 ,  97.66299 , 103.06418 ,
       108.22845 , 102.17059 ,  98.12046 ,  98.56349 ,  79.02904 ,
        95.557365,  98.29806 ,  77.39744 ,  95.555855,  96.14678 ,
        93.70941 , 103.535835,  93.5425  , 105.43336 , 107.15543 ,
        99.605156, 103.30476 , 104.33832 ,  93.509766,  97.24388 ,
        94.125854, 106.76455 ], dtype=float32)
```

[22]:
```python
Predicted_Data = pd.DataFrame()
Predicted_Data['y'] = p_test
Predicted_Data.head()
```

[22]:

	y
0	92.604988
1	96.806854
2	102.802643
3	79.457001
4	111.140488

[ ]:
```python

```