In [260…
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
```

In [90]:
```
train_data=pd.read_csv("D:\\AI-SL\\ML Project\\Mercedes Benz\\train.csv")
test_data=pd.read_csv("D:\\AI-SL\\ML Project\\Mercedes Benz\\test.csv")
```

In [91]:
```
train_data.head()
```

Out[91]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X38
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	

5 rows × 378 columns

In [92]:
```
train_data.tail()
```

Out[92]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380
4204	8405	107.39	ak	s	as	c	d	aa	d	q	...	1	0	0	0	0	(
4205	8406	108.77	j	o	t	d	d	aa	h	h	...	0	1	0	0	0	(
4206	8412	109.22	ak	v	r	a	d	aa	g	e	...	0	0	1	0	0	(
4207	8415	87.48	al	r	e	f	d	aa	l	u	...	0	0	0	0	0	(
4208	8417	110.85	z	r	ae	c	d	aa	g	w	...	1	0	0	0	0	(

5 rows × 378 columns

In [93]:
```
train_data.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
```

In [103…
```
train_data.describe()
```

Out[103]:

	ID	y	X10	X12	X13	X14	X15
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	4205.960798	100.669318	0.013305	0.075077	0.057971	0.428130	0.000475
std	2437.608688	12.679381	0.114590	0.263547	0.233716	0.494867	0.021796
min	0.000000	72.110000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2095.000000	90.820000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	4220.000000	99.150000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	6314.000000	109.010000	0.000000	0.000000	0.000000	1.000000	0.000000
max	8417.000000	265.320000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 358 columns

In [105…

```
test_data.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
```

In [106…

```
test_data.describe()
```

Out[106]:

	ID	X10	X11	X12	X13	X14	X15
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	4211.039202	0.019007	0.000238	0.074364	0.061060	0.427893	0.000713
std	2423.078926	0.136565	0.015414	0.262394	0.239468	0.494832	0.026691
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2115.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	4202.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	6310.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
max	8416.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 369 columns

Check for Zero Variance columns

In [95]:

```
numeric_columns=train_data.select_dtypes(include=np.number) #take columns that has
variance_per_column=np.var(numeric_columns,axis=0)
print(variance_per_column)
```

```
ID      5.940524e+06
y       1.607285e+02
X10     1.312780e-02
X11     0.000000e+00
X12     6.944063e-02
            ...
X380    8.012675e-03
X382    7.544954e-03
X383    1.660337e-03
X384    4.749465e-04
X385    1.423485e-03
Length: 370, dtype: float64
```

In [96]:
```
zero_var_col=numeric_columns.columns[variance_per_column==0]
zero_var_col
```

Out[96]:
```
Index(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293',
       'X297', 'X330', 'X347'],
      dtype='object')
```

In [97]:
```
train_data[zero_var_col]
```

Out[97]:

	X11	X93	X107	X233	X235	X268	X289	X290	X293	X297	X330	X347
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
4204	0	0	0	0	0	0	0	0	0	0	0	0
4205	0	0	0	0	0	0	0	0	0	0	0	0
4206	0	0	0	0	0	0	0	0	0	0	0	0
4207	0	0	0	0	0	0	0	0	0	0	0	0
4208	0	0	0	0	0	0	0	0	0	0	0	0

4209 rows × 12 columns

In [108…
```
train_data=train_data.drop(columns=zero_var_col)
#print(train_data.columns)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[108], line 1
----> 1 train_data=train_data.drop(columns=zero_var_col)

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:311, in deprecate_no
nkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:4954, in DataFrame.drop(se
lf, labels, axis, index, columns, level, inplace, errors)
   4806 @deprecate_nonkeyword_arguments(version=None, allowed_args=["self", "label
s"])
   4807 def drop(
   4808     self,
   (...)
   4815     errors: str = "raise",
   4816 ):
   4817     """
   4818     Drop specified labels from rows or columns.
   4819
   (...)
   4952             weight  1.0     0.8
   4953     """
-> 4954     return super().drop(
   4955         labels=labels,
   4956         axis=axis,
   4957         index=index,
   4958         columns=columns,
   4959         level=level,
   4960         inplace=inplace,
   4961         errors=errors,
   4962     )

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:4267, in NDFrame.drop(se
lf, labels, axis, index, columns, level, inplace, errors)
   4265 for axis, labels in axes.items():
   4266     if labels is not None:
-> 4267         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4269 if inplace:
   4270     self._update_inplace(obj)

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:4311, in NDFrame._drop_a
xis(self, labels, axis, level, errors, consolidate, only_slice)
   4309         new_axis = axis.drop(labels, level=level, errors=errors)
   4310     else:
-> 4311         new_axis = axis.drop(labels, errors=errors)
   4312     indexer = axis.get_indexer(new_axis)
   4314 # Case for non-unique axis
   4315 else:

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:6644, in Index.drop
(self, labels, errors)
   6642 if mask.any():
   6643     if errors != "ignore":
-> 6644         raise KeyError(f"{list(labels[mask])} not found in axis")
   6645     indexer = indexer[~mask]
   6646 return self.delete(indexer)
```

```
KeyError: "['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293',
'X297', 'X330', 'X347'] not found in axis"
```

In [110…

```
test_data_num_col=test_data.select_dtypes(include=np.number)
test_var_percol=np.var(test_data_num_col)
test_var_percol
```

Out[110]:

```
ID      5.869917e+06
X10     1.864563e-02
X11     2.375297e-04
X12     6.883438e-02
X13     5.733136e-02
            ...
X380    8.012675e-03
X382    8.713410e-03
X383    4.749465e-04
X384    7.122504e-04
X385    1.660337e-03
Length: 369, dtype: float64
```

In [118…

```
test_data_zero_var_col=test_data_num_col.columns[test_var_percol==0]
test_data_zero_var_col
```

Out[118]:

```
Index(['X257', 'X258', 'X295', 'X296', 'X369'], dtype='object')
```

In [120…

```
test_data[test_data_zero_var_col]
```

Out[120]:

	X257	X258	X295	X296	X369
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
...	...	...	...	...	...
4204	0	0	0	0	0
4205	0	0	0	0	0
4206	0	0	0	0	0
4207	0	0	0	0	0
4208	0	0	0	0	0

4209 rows × 5 columns

In [122…

```
test_data=test_data.drop(columns=[test_data_zero_var_col])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[122], line 1
----> 1 test_data=test_data.drop(columns=[test_data_zero_var_col])

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:311, in deprecate_no
nkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:4954, in DataFrame.drop(se
lf, labels, axis, index, columns, level, inplace, errors)
   4806 @deprecate_nonkeyword_arguments(version=None, allowed_args=["self", "label
s"])
   4807 def drop(
   4808     self,
   (...)
   4815     errors: str = "raise",
   4816 ):
   4817     """
   4818     Drop specified labels from rows or columns.
   4819 
   (...)
   4952             weight  1.0     0.8
   4953     """
-> 4954     return super().drop(
   4955         labels=labels,
   4956         axis=axis,
   4957         index=index,
   4958         columns=columns,
   4959         level=level,
   4960         inplace=inplace,
   4961         errors=errors,
   4962     )

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:4267, in NDFrame.drop(se
lf, labels, axis, index, columns, level, inplace, errors)
   4265 for axis, labels in axes.items():
   4266     if labels is not None:
-> 4267         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4269 if inplace:
   4270     self._update_inplace(obj)

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:4311, in NDFrame._drop_a
xis(self, labels, axis, level, errors, consolidate, only_slice)
   4309         new_axis = axis.drop(labels, level=level, errors=errors)
   4310     else:
-> 4311         new_axis = axis.drop(labels, errors=errors)
   4312     indexer = axis.get_indexer(new_axis)
   4314 # Case for non-unique axis
   4315 else:

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:6644, in Index.drop
(self, labels, errors)
   6642 if mask.any():
   6643     if errors != "ignore":
-> 6644         raise KeyError(f"{list(labels[mask])} not found in axis")
   6645     indexer = indexer[~mask]
   6646 return self.delete(indexer)
```

```
KeyError: "[('X257', 'X258', 'X295', 'X296', 'X369')] not found in axis"
```

In [123…

```
test_data.columns
```

Out[123]:

```
Index(['ID', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8', 'X10',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=377)
```

Check for Null and unique values

In [124…

```
train_data.isna().sum()
```

Out[124]:

```
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 366, dtype: int64
```

In [125…

```
test_data.isna().sum()
```

Out[125]:

```
ID      0
X0      0
X1      0
X2      0
X3      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 377, dtype: int64
```

In [126…

```
features_with_na= [features for features in train_data.columns if train_data[featu
print(features_with_na)
```

```
[]
```

In [129…

```
test_features_with_na=[features for features in test_data.columns if test_data[fea
print(test_features_with_na)
```

```
[]
```

In [101…

```
train_data.columns
```

Out[101]:

```
Index(['ID', 'y', 'X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8',
       ...
       'X375', 'X376', 'X377', 'X378', 'X379', 'X380', 'X382', 'X383', 'X384',
       'X385'],
      dtype='object', length=366)
```

In [130…

```
train_data.nunique()
```

Out[130]:
```
ID      4209
y       2545
X0        47
X1        27
X2        44
        ... 
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 366, dtype: int64
```

In [131…
```
test_data.nunique()
```

Out[131]:
```
ID      4209
X0        49
X1        27
X2        45
X3         7
        ... 
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 377, dtype: int64
```

Label encoding for category columns

In [140…
```
train_cat_cols=train_data.select_dtypes(include=object) #to extract columns with s
train_cat_cols
```

Out[140]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	k	v	at	a	d	u	j	o
1	k	t	av	e	d	y	l	o
2	az	w	n	c	d	x	j	x
3	az	t	n	f	d	x	l	e
4	az	v	n	f	d	h	d	n
...	...	...	...	...	...	...	...	...
4204	ak	s	as	c	d	aa	d	q
4205	j	o	t	d	d	aa	h	h
4206	ak	v	r	a	d	aa	g	e
4207	al	r	e	f	d	aa	l	u
4208	z	r	ae	c	d	aa	g	w

4209 rows × 8 columns

In [160…
```
for col in train_cat_cols.columns:
    print("col:{} and unique values :{}".format(col,train_cat_cols[col].unique()))
```

```
col:X0 and unique values :['k' 'az' 't' 'al' 'o' 'w' 'j' 'h' 's' 'n' 'ay' 'f' 'x'
'y' 'aj' 'ak' 'am'
 'z' 'q' 'at' 'ap' 'v' 'af' 'a' 'e' 'ai' 'd' 'aq' 'c' 'aa' 'ba' 'as' 'i'
 'r' 'b' 'ax' 'bc' 'u' 'ad' 'au' 'm' 'l' 'aw' 'ao' 'ac' 'g' 'ab']
col:X1 and unique values :['v' 't' 'w' 'b' 'r' 'l' 's' 'aa' 'c' 'a' 'e' 'h' 'z'
'j' 'o' 'u' 'p' 'n'
 'i' 'y' 'd' 'f' 'm' 'k' 'g' 'q' 'ab']
col:X2 and unique values :['at' 'av' 'n' 'e' 'as' 'aq' 'r' 'ai' 'ak' 'm' 'a' 'k'
'ae' 's' 'f' 'd'
 'ag' 'ay' 'ac' 'ap' 'g' 'i' 'aw' 'y' 'b' 'ao' 'al' 'h' 'x' 'au' 't' 'an'
 'z' 'ah' 'p' 'am' 'j' 'q' 'af' 'l' 'aa' 'c' 'o' 'ar']
col:X3 and unique values :['a' 'e' 'c' 'f' 'd' 'b' 'g']
col:X4 and unique values :['d' 'b' 'c' 'a']
col:X5 and unique values :['u' 'y' 'x' 'h' 'g' 'f' 'j' 'i' 'd' 'c' 'af' 'ag' 'ab'
'ac' 'ad' 'ae'
 'ah' 'l' 'k' 'n' 'm' 'p' 'q' 's' 'r' 'v' 'w' 'o' 'aa']
col:X6 and unique values :['j' 'l' 'd' 'h' 'i' 'a' 'g' 'c' 'k' 'e' 'f' 'b']
col:X8 and unique values :['o' 'x' 'e' 'n' 's' 'a' 'h' 'p' 'm' 'k' 'd' 'i' 'v' 'j'
'b' 'q' 'w' 'g'
 'y' 'l' 'f' 'u' 'r' 't' 'c']
```

In [153…

```python
train_cat_cols.columns
```

Out[153]:

```
Index(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype='object')
```

In [183…

```python
label_enc=preprocessing.LabelEncoder()
for col in train_cat_cols.columns:
    train_cat_cols[col]=label_enc.fit_transform(train_cat_cols[col])
    print("col:{} & unique values:{}".format(col, train_cat_cols[col].unique()))
```

```
col:X0 & unique values:[32 20 40  9 36 43 31 29 39 35 19 27 44 45  7  8 10 46 37 1
5 12 42  5  0
 26  6 25 13 24  1 22 14 30 38 21 18 23 41  4 16 34 33 17 11  3 28  2]
col:X1 & unique values:[23 21 24  3 19 13 20  1  4  0  6  9 26 11 16 22 17 15 10 2
5  5  7 14 12
  8 18  2]
col:X2 & unique values:[17 19 34 25 16 14 38  7  8 33  0 31  3 39 26 24  5 21  2 1
3 27 29 20 42
 22 12  9 28 41 18 40 11 43  6 36 10 30 37  4 32  1 23 35 15]
col:X3 & unique values:[0 4 2 5 3 1 6]
col:X4 & unique values:[3 1 2 0]
col:X5 & unique values:[24 28 27 12 11 10 14 13  9  8  5  6  1  2  3  4  7 16 15 1
8 17 20 21 23
 22 25 26 19  0]
col:X6 & unique values:[ 9 11  3  7  8  0  6  2 10  4  5  1]
col:X8 & unique values:[14 23  4 13 18  0  7 15 12 10  3  8 21  9  1 16 22  6 24 1
1  5 20 17 19
  2]
```

In [175…

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[175], line 1
----> 1 train_data.select_dtype(include=np.number)

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:5575, in NDFrame.__getat
tr__(self, name)
   5568 if (
   5569     name not in self._internal_names_set
   5570     and name not in self._metadata
   5571     and name not in self._accessors
   5572     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5573 ):
   5574     return self[name]
-> 5575 return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'select_dtype'
```

In [180…

```
train_data_upd=pd.concat([numeric_columns,train_cat_cols],axis=1)
train_data_upd
```

Out[180]:

	ID	y	X10	X11	X12	X13	X14	X15	X16	X17	...	X384	X385	X0	X1	X2	X3
0	0	130.81	0	0	0	1	0	0	0	0	...	0	0	32	23	17	0
1	6	88.53	0	0	0	0	0	0	0	0	...	0	0	32	21	19	4
2	7	76.26	0	0	0	0	0	0	0	1	...	0	0	20	24	34	2
3	9	80.62	0	0	0	0	0	0	0	0	...	0	0	20	21	34	5
4	13	78.02	0	0	0	0	0	0	0	0	...	0	0	20	23	34	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8405	107.39	0	0	0	0	1	0	0	0	...	0	0	8	20	16	2
4205	8406	108.77	0	0	0	0	0	0	0	0	...	0	0	31	16	40	3
4206	8412	109.22	0	0	1	1	0	0	0	0	...	0	0	8	23	38	0
4207	8415	87.48	0	0	0	0	1	0	0	0	...	0	0	9	19	25	5
4208	8417	110.85	0	0	0	0	0	0	0	0	...	0	0	46	19	3	2

4209 rows × 378 columns

In [182…

```
test_cat_cols=test_data.select_dtypes(include=object)
test_cat_cols
```

Out[182]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	az	v	n	f	d	t	a	w
1	t	b	ai	a	d	b	g	y
2	az	v	as	f	d	a	j	j
3	az	l	n	f	d	z	l	n
4	w	s	as	c	d	y	i	m
...	...	...	...	...	...	...	...	...
4204	aj	h	as	f	d	aa	j	e
4205	t	aa	ai	d	d	aa	j	y
4206	y	v	as	f	d	aa	d	w
4207	ak	v	as	a	d	aa	c	q
4208	t	aa	ai	c	d	aa	g	r

4209 rows × 8 columns

In [189…

```python
for col in test_cat_cols.columns:
    print(f'col : {col},unique values : {test_data[col].unique()}')
```

```
col : X0,unique values : ['az' 't' 'w' 'y' 'x' 'f' 'ap' 'o' 'ay' 'al' 'h' 'z' 'aj'
'd' 'v' 'ak'
 'ba' 'n' 'j' 's' 'af' 'ax' 'at' 'aq' 'av' 'm' 'k' 'a' 'e' 'ai' 'i' 'ag'
 'b' 'am' 'aw' 'as' 'r' 'ao' 'u' 'l' 'c' 'ad' 'au' 'bc' 'g' 'an' 'ae' 'p'
 'bb']
col : X1,unique values : ['v' 'b' 'l' 's' 'aa' 'r' 'a' 'i' 'p' 'c' 'o' 'm' 'z' 'e'
'h' 'w' 'g' 'k'
 'y' 't' 'u' 'd' 'j' 'q' 'n' 'f' 'ab']
col : X2,unique values : ['n' 'ai' 'as' 'ae' 's' 'b' 'e' 'ak' 'm' 'a' 'aq' 'ag'
'r' 'k' 'aj' 'ay'
 'ao' 'an' 'ac' 'af' 'ax' 'h' 'i' 'f' 'ap' 'p' 'au' 't' 'z' 'y' 'aw' 'd'
 'at' 'g' 'am' 'j' 'x' 'ab' 'w' 'q' 'ah' 'ad' 'al' 'av' 'u']
col : X3,unique values : ['f' 'a' 'c' 'e' 'd' 'g' 'b']
col : X4,unique values : ['d' 'b' 'a' 'c']
col : X5,unique values : ['t' 'b' 'a' 'z' 'y' 'x' 'h' 'g' 'f' 'j' 'i' 'd' 'c' 'af'
'ag' 'ab' 'ac'
 'ad' 'ae' 'ah' 'l' 'k' 'n' 'm' 'p' 'q' 's' 'r' 'v' 'w' 'o' 'aa']
col : X6,unique values : ['a' 'g' 'j' 'l' 'i' 'd' 'f' 'h' 'c' 'k' 'e' 'b']
col : X8,unique values : ['w' 'y' 'j' 'n' 'm' 's' 'a' 'v' 'r' 'o' 't' 'h' 'c' 'k'
'p' 'u' 'd' 'g'
 'b' 'q' 'e' 'l' 'f' 'i' 'x']
```

In [192…

```python
for col in test_cat_cols.columns:
    test_cat_cols[col]=label_enc.fit_transform(test_cat_cols[col])
    print(f"cols: {col}, unique values:{test_cat_cols[col]}")
```

```
cols: X0, unique values:0       21
1       42
2       21
3       21
4       45
        ..
4204     6
4205    42
4206    47
4207     7
4208    42
Name: X0, Length: 4209, dtype: int64
cols: X1, unique values:0       23
1        3
2       23
3       13
4       20
        ..
4204     9
4205     1
4206    23
4207    23
4208     1
Name: X1, Length: 4209, dtype: int32
cols: X2, unique values:0       34
1        8
2       17
3       34
4       17
        ..
4204    17
4205     8
4206    17
4207    17
4208     8
Name: X2, Length: 4209, dtype: int32
cols: X3, unique values:0       5
1       0
2       5
3       5
4       2
       ..
4204    5
4205    3
4206    5
4207    0
4208    2
Name: X3, Length: 4209, dtype: int32
cols: X4, unique values:0       3
1       3
2       3
3       3
4       3
       ..
4204    3
4205    3
4206    3
4207    3
4208    3
Name: X4, Length: 4209, dtype: int32
cols: X5, unique values:0       26
1        9
2        0
3       31
```

```
4       30
        ..
4204     1
4205     1
4206     1
4207     1
4208     1
Name: X5, Length: 4209, dtype: int32
cols: X6, unique values:0        0
1        6
2        9
3       11
4        8
        ..
4204     9
4205     9
4206     3
4207     2
4208     6
Name: X6, Length: 4209, dtype: int32
cols: X8, unique values:0       22
1       24
2        9
3       13
4       12
        ..
4204     4
4205    24
4206    22
4207    16
4208    17
Name: X8, Length: 4209, dtype: int32
```

In [195…

```
test_data_upd=pd.concat([test_data_num_col,test_cat_cols],axis=1)
test_data_upd
```

Out[195]:

	ID	X10	X11	X12	X13	X14	X15	X16	X17	X18	...	X384	X385	X0	X1	X2	X3
0	1	0	0	0	0	0	0	0	0	0	...	0	0	21	23	34	5
1	2	0	0	0	0	0	0	0	0	0	...	0	0	42	3	8	0
2	3	0	0	0	0	1	0	0	0	0	...	0	0	21	23	17	5
3	4	0	0	0	0	0	0	0	0	0	...	0	0	21	13	34	5
4	5	0	0	0	0	1	0	0	0	0	...	0	0	45	20	17	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8410	0	0	0	0	1	0	0	0	0	...	0	0	6	9	17	5
4205	8411	0	0	0	0	0	0	0	0	0	...	0	0	42	1	8	3
4206	8413	0	0	0	0	1	0	0	0	0	...	0	0	47	23	17	5
4207	8414	0	0	0	1	1	0	0	0	0	...	0	0	7	23	17	0
4208	8416	0	0	0	0	0	0	0	0	0	...	0	0	42	1	8	2

4209 rows × 377 columns

Dimensionality reduction using PCA

In [196…
```
train_data_upd.shape
```
Out[196]:
```
(4209, 378)
```

In [198…
```
y=train_data_upd['y']
y
```
Out[198]:
```
0       130.81
1        88.53
2        76.26
3        80.62
4        78.02
         ...  
4204    107.39
4205    108.77
4206    109.22
4207     87.48
4208    110.85
Name: y, Length: 4209, dtype: float64
```

In [199…
```
train_data_upd=train_data_upd.drop(columns=['y'])
```

In [200…
```
train_data_upd.shape
```
Out[200]:
```
(4209, 377)
```

In [202…
```
#Scaling the input and target
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
train_data_scaled=scaler.fit_transform(train_data_upd)
train_data_scaled
```
Out[202]:
```
array([[-1.72565045, -0.11612161,  0.        , ...,  1.29211729,
         0.75178725,  0.33944483],
       [-1.72318873, -0.11612161,  0.        , ...,  1.77697445,
         1.43751106,  0.33944483],
       [-1.72277844, -0.11612161,  0.        , ...,  1.65576016,
         0.75178725,  1.61838949],
       ...,
       [ 1.72568262, -0.11612161,  0.        , ..., -1.61702573,
        -0.27679847, -1.08160479],
       [ 1.72691348, -0.11612161,  0.        , ..., -1.61702573,
         1.43751106,  1.1920746 ],
       [ 1.72773405, -0.11612161,  0.        , ..., -1.61702573,
        -0.27679847,  1.47628453]])
```

In [229…
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(train_data_scaled,y,test_size=0.2,r
```

In [230…
```
train_data_scaled.shape
```
Out[230]:
```
(4209, 377)
```

In [231…
```
import xgboost as xgb
from sklearn.metrics import mean_squared_error
from xgboost import XGBRegressor
```

In [232…
```
model=XGBRegressor(objective='reg:squarederror',n_estimators=50,learning_rate=0.1,
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
```

In [233…
```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```
```
Mean Squared Error: 62.124620867989364
```

In [239…
```
from sklearn.decomposition import PCA
pca=PCA(0.95) #retain 95% of useful features
X_train_pca=pca.fit_transform(train_data_scaled)
print(X_train_pca.shape)
```
```
(4209, 149)
```

In [240…
```
print(X_train_pca.shape,y.shape)
```
```
(4209, 149) (4209,)
```

In [241…
```
pca.explained_variance_ratio_
```

Out[241]:
```
array([0.06873845, 0.05672831, 0.04525105, 0.03417386, 0.03255383,
       0.03154186, 0.02854713, 0.02118177, 0.01968633, 0.01778935,
       0.0163563 , 0.015601  , 0.0145906 , 0.01445648, 0.01344956,
       0.01292573, 0.01241382, 0.01171394, 0.01119126, 0.01074961,
       0.00989891, 0.0096776 , 0.00940046, 0.00908605, 0.00872347,
       0.0084076 , 0.00792762, 0.00761389, 0.00734903, 0.00718305,
       0.00691227, 0.00675052, 0.00655057, 0.00646544, 0.00621348,
       0.00600246, 0.0058665 , 0.00574454, 0.00562534, 0.00555771,
       0.00550145, 0.00538603, 0.00532449, 0.00523216, 0.00511352,
       0.00501857, 0.00497724, 0.00477276, 0.0046579 , 0.00459137,
       0.00446221, 0.0043733 , 0.00431693, 0.00429122, 0.00422545,
       0.0041891 , 0.00413148, 0.00405572, 0.0040222 , 0.00388352,
       0.00386855, 0.00380218, 0.00374184, 0.00365935, 0.00359751,
       0.00357123, 0.0035294 , 0.00346016, 0.00341059, 0.00335091,
       0.00332836, 0.0032594 , 0.00323873, 0.0032048 , 0.00316934,
       0.00315804, 0.0031486 , 0.00308903, 0.00306594, 0.00303922,
       0.00299867, 0.00298425, 0.00295864, 0.00292366, 0.0029006 ,
       0.00289135, 0.00286429, 0.00284373, 0.0028264 , 0.00280433,
       0.0027932 , 0.00276794, 0.00274409, 0.00273399, 0.00271654,
       0.00270406, 0.0026484 , 0.00264044, 0.00261697, 0.0025998 ,
       0.00258923, 0.00255473, 0.00253179, 0.00251264, 0.00250014,
       0.00248148, 0.00243858, 0.00241888, 0.00240045, 0.00237785,
       0.00234644, 0.00230577, 0.00230055, 0.00227058, 0.00225174,
       0.00222925, 0.0022086 , 0.0021946 , 0.00214567, 0.00213139,
       0.00211387, 0.00209153, 0.00205648, 0.00203631, 0.00202058,
       0.00198862, 0.0019337 , 0.00191698, 0.00191371, 0.00188131,
       0.001847  , 0.00181313, 0.00178889, 0.00178173, 0.00175136,
       0.00171302, 0.00170264, 0.00167895, 0.0016536 , 0.00161437,
       0.00160919, 0.00157355, 0.00154212, 0.00153118, 0.001496  ,
       0.00149006, 0.00147679, 0.0014261 , 0.00140735])
```

In [242…
```
X_train_pcaa,X_test_pca,y_train,y_test=train_test_split(X_train_pca,y,test_size=0.
```

In [243…
```
model=XGBRegressor(objective='reg:squarederror',n_estimators=50,learning_rate=0.1,
model.fit(X_train_pcaa,y_train)
y_pred_pca=model.predict(X_test_pca)
```

In [244…
```
mse = mean_squared_error(y_test, y_pred_pca)
print(f'Mean Squared Error: {mse}')
```
```
Mean Squared Error: 76.65334766897016
```

In [245…
```
#pca with n_components=10
pca_1=PCA(n_components=10)
X_train_pca1=pca_1.fit_transform(train_data_scaled)
X_train_pca1.shape
```

```
Out[245]: (4209, 10)
```

```
In [246…  X_train_pca1,X_test_pca1,y_train,y_test=train_test_split(X_train_pca1,y,test_size=
```

```
In [248…  model=XGBRegressor(objective='reg:squarederror',n_estimators=50,learning_rate=0.1,
          model.fit(X_train_pca1,y_train)
          y_pred_pca1=model.predict(X_test_pca1)
```

```
In [249…  mse_pca1 = mean_squared_error(y_test, y_pred_pca1)
          print(f'Mean Squared Error: {mse_pca1}')
```

```
Mean Squared Error: 91.79506809351535
```

Accuracy achieved upon PCA = 91.795

```
In [250…  model.save_model('my_xgboost_model.model')
```

```
In [254…  test_data_scaled=scaler.fit_transform(test_data_upd)
          test_data_scaled.shape
```

```
Out[254]: (4209, 377)
```

```
In [255…  X_test_data_pca1=pca_1.fit_transform(test_data_scaled)
          X_test_data_pca1.shape
```

```
Out[255]: (4209, 10)
```

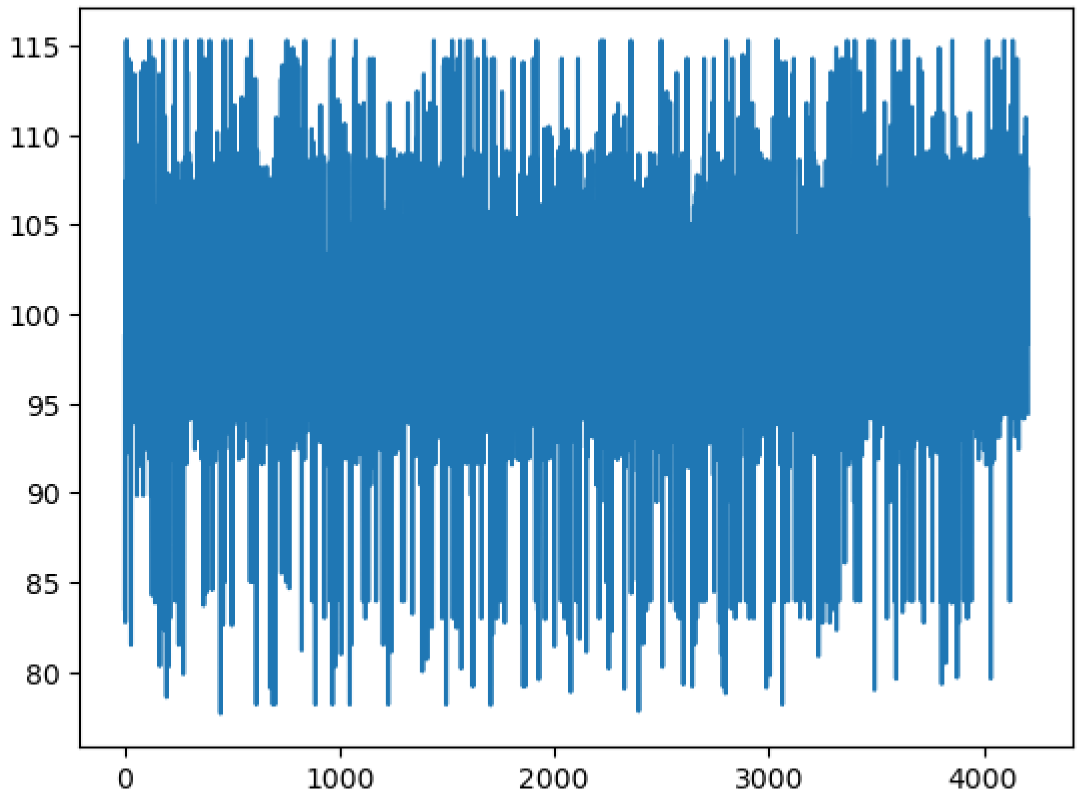
```
In [256…  y_pred_test=model.predict(X_test_data_pca1)
```

```
In [257…  y_pred_test #testing time of Mercedes Benz on the test bench
```

```
Out[257]: array([ 83.54843 ,  98.88009 ,  82.77069 , ...,  98.371254, 105.347786,
                  94.44501 ], dtype=float32)
```

```
In [264…  plt.plot(y_pred_test)
```

```
Out[264]: [<matplotlib.lines.Line2D at 0x1ec45acaeb0>]
```



```
In [ ]:
```