```
In [1]: import os
        os.getcwd()

Out[1]: 'C:\\Users\\218882'
```

```
In [2]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
In [3]: train_df = pd.read_csv("C:\\Users\\218882\\Mercedes-Benz_Train_Data.csv")
        train_df.head()
```

Out[3]:

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

5 rows × 378 columns

```
In [4]: test_df = pd.read_csv("C:\\Users\\218882\\Mercedes-Benz_Test_Data.csv")
        test_df.head()
```

Out[4]:

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

5 rows × 377 columns

```
In [5]: train_df.shape

Out[5]: (4209, 378)
```

```
In [6]: test_df.shape

Out[6]: (4209, 377)
```

In [7]:
```
train_df.describe()
```

Out[7]:

	ID	y	X10	X11	X12	X13	X14	X15	X16	X17	...	X375	X376	X377	X378	X379	X380
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	...	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.428130	0.000475	0.002613	0.007603	...	0.318841	0.057258	0.314802	0.020670	0.009503	0.008078
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.494867	0.021796	0.051061	0.086872	...	0.466082	0.232363	0.464492	0.142294	0.097033	0.089524
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	...	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 370 columns

In [8]:
```
train_df = train_df.drop(["ID","y"] ,axis=1)
train_df
```

Out[8]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0	0	0	0	0
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0	0	0	0	0
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0	1	0	0	0
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0	0	0	0	0
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	ak	s	as	c	d	aa	d	q	0	0	...	1	0	0	0	0	0	0	0	0	0
4205	j	o	t	d	d	aa	h	h	0	0	...	0	1	0	0	0	0	0	0	0	0
4206	ak	v	r	a	d	aa	g	e	0	0	...	0	0	1	0	0	0	0	0	0	0
4207	al	r	e	f	d	aa	l	u	0	0	...	0	0	0	0	0	0	0	0	0	0
4208	z	r	ae	c	d	aa	g	w	0	0	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 376 columns

In [9]:
```
## TASK 1 ##

# If for any column(s), the variance is equal to zero, then you need to remove those variable(s).
```

In [10]:
```
train_df.var()
```

```
C:\Users\218882\AppData\Local\Temp\ipykernel_2788\57518514.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a
future version this will raise TypeError.  Select only valid columns before calling the reduction.
  train_df.var()
```

Out[10]:
```
X10     0.013131
X11     0.000000
X12     0.069457
X13     0.054623
X14     0.244893
          ...
X380    0.008015
X382    0.007547
X383    0.001661
X384    0.000475
X385    0.001424
Length: 368, dtype: float64
```

In [11]:
```
train_df.var() == 0
```

```
C:\Users\218882\AppData\Local\Temp\ipykernel_2788\2393790271.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in
a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  train_df.var() == 0
```

Out[11]:
```
X10     False
X11      True
X12     False
X13     False
X14     False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 368, dtype: bool
```

In [13]:
```
zeros = []
for x, y in train_df.any().items():
    if y == False:
        zeros.append(x)
```

In [14]:
```
zeros
```

Out[14]:
```
['X11',
 'X93',
 'X107',
 'X233',
 'X235',
 'X268',
 'X289',
 'X290',
 'X293',
 'X297',
 'X330',
 'X347']
```

In [15]:
```
train_df = train_df.drop(zeros, axis=1)
```

In [16]:
```
train_df
```

Out[16]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0	0	0	0	0
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0	0	0	0	0
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0	1	0	0	0
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0	0	0	0	0
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	ak	s	as	c	d	aa	d	q	0	0	...	1	0	0	0	0	0	0	0	0	0
4205	j	o	t	d	d	aa	h	h	0	0	...	0	1	0	0	0	0	0	0	0	0
4206	ak	v	r	a	d	aa	g	e	0	1	...	0	0	1	0	0	0	0	0	0	0
4207	al	r	e	f	d	aa	l	u	0	0	...	0	0	0	0	0	0	0	0	0	0
4208	z	r	ae	c	d	aa	g	w	0	0	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 364 columns

In [17]:
```
train_df.shape
```

Out[17]:
```
(4209, 364)
```

In [18]:
```
## TASK 2 ##

# Check the null and unique values for test and train sets.
```

In [19]:
```
train_df.isnull().sum().values
```

Out[19]:
```
array([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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [20]:
```
train_df.isnull().any()
```

Out[20]:
```
X0      False
X1      False
X2      False
X3      False
X4      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 364, dtype: bool
```

In [22]:
```
test_df.isnull().sum().values
```

Out[22]:
```
array([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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 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, 0, 0], dtype=int64)
```

In [23]:
```
test_df.isnull().any()
```

Out[23]:
```
ID      False
X0      False
X1      False
X2      False
X3      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 377, dtype: bool
```

In [24]:
```
train_df.nunique()
```

Out[24]:
```
X0      47
X1      27
X2      44
X3       7
X4       4
        ..
X380     2
X382     2
X383     2
X384     2
X385     2
Length: 364, dtype: int64
```

In [25]:
```
test_df.nunique()
```

Out[25]:
```
ID      4209
X0        49
X1        27
X2        45
X3         7
        ...
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 377, dtype: int64
```

In [26]:
```
## TASK 3 ##

# Apply Label Encoder.
```

In [27]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [29]:
```
train_df_feature = train_df
train_df_target = train_df
print(train_df_feature.shape)
print(train_df_target.shape)
```

```
(4209, 364)
(4209, 364)
```

In [30]:
```
train_df_feature.describe(include='object')
```

Out[30]:

	X0	X1	X2	X3	X4	X5	X6	X8
count	4209	4209	4209	4209	4209	4209	4209	4209
unique	47	27	44	7	4	29	12	25
top	z	aa	as	c	d	w	g	j
freq	360	833	1659	1942	4205	231	1042	277

In [31]:
```
train_df_feature['X0'] = le.fit_transform(train_df_feature.X0)
train_df_feature['X1'] = le.fit_transform(train_df_feature.X1)
train_df_feature['X2'] = le.fit_transform(train_df_feature.X2)
train_df_feature['X3'] = le.fit_transform(train_df_feature.X3)
train_df_feature['X4'] = le.fit_transform(train_df_feature.X4)
train_df_feature['X5'] = le.fit_transform(train_df_feature.X5)
train_df_feature['X6'] = le.fit_transform(train_df_feature.X6)
train_df_feature['X8'] = le.fit_transform(train_df_feature.X8)
```

In [32]:
```
## TASK 4 ##

# Perform dimensionality reduction.
```

In [33]:
```
print(train_df_feature.shape)
print(train_df_target.shape)
```

```
(4209, 364)
(4209, 364)
```

In [34]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.95)
```

In [35]:
```
pca.fit(train_df_feature, train_df_target)
```

Out[35]:
```
PCA(n_components=0.95)
```

In [36]:
```
train_df_feature_trans = pca.fit_transform(train_df_feature)
print(train_df_feature_trans.shape)
```

```
(4209, 6)
```

In [37]:
```
## TASK 5 ##

# Predict your test_df values using XGBoost.
```

In [38]:
```
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt
```

In [41]:
```
x_train,x_test,y_train,y_test = train_test_split(train_df_feature_trans, train_df_target, test_size=.3, random_state=7)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(2946, 6)
(2946, 364)
(1263, 6)
(1263, 364)
```

In [42]:
```
xgb_reg = xgb.XGBRegressor(objective = 'reg:linear', colsample_bytree = 0.3, learning_rate = 0.4, max_depth = 10, alpha = 6, n_estimators = 20)
model = xgb_reg.fit(x_train, y_train)
print('RMSE = ',sqrt(mean_squared_error(model.predict(x_test), y_test)))
```

```
[13:30:45] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/src/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squarederror.
RMSE =  0.5112402443904084
```

In [43]:
```
# Cross validation using XGBoost.
```

In [44]:
```
dmatrix_train = xgb.DMatrix(data=train_df_feature_trans, label=train_df_target)

params = {'objective':'reg:linear', 'colsample_bytree': 0.3, 'learning_rate': 0.3, 'max_depth': 5, 'alpha': 10}

model_cv = xgb.cv(dtrain=dmatrix_train, params=params, nfold=3, num_boost_round=50, early_stopping_rounds=10, metrics="rmse", as_pandas=True, seed=7)
model_cv.tail(5)
```

```
[13:39:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/src/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squarederror.
[13:39:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/src/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squarederror.
[13:39:03] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/src/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squarederror.
```

Out[44]:

	train-rmse-mean	train-rmse-std	test-rmse-mean	test-rmse-std
45	0.307012	0.018648	0.378617	0.037242
46	0.300640	0.020960	0.371604	0.039927
47	0.299699	0.020927	0.371078	0.039867
48	0.295647	0.016663	0.366244	0.034182
49	0.293847	0.017055	0.364615	0.034807

In [45]:
```
# Prediction on test data set using XGBoost.
```

In [46]:
```
test_df
```

Out[46]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8410	aj	h	as	f	d	aa	j	e	0	...	0	0	0	0	0	0	0	0	0	0
4205	8411	t	aa	ai	d	d	aa	j	y	0	...	0	1	0	0	0	0	0	0	0	0
4206	8413	y	v	as	f	d	aa	d	w	0	...	0	0	0	0	0	0	0	0	0	0
4207	8414	ak	v	as	a	d	aa	c	q	0	...	0	0	1	0	0	0	0	0	0	0
4208	8416	t	aa	ai	c	d	aa	g	r	0	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 377 columns

```
In [49]: test_df.isnull().sum().any()

Out[49]: False
```

```
In [51]: test_df_feature = test_df.drop(["ID"] ,axis=1)
         print(test_df_feature.shape)

         (4209, 376)
```

```
In [52]: test_df_feature.describe(include='object')

Out[52]:
```

	X0	X1	X2	X3	X4	X5	X6	X8
count	4209	4209	4209	4209	4209	4209	4209	4209
unique	49	27	45	7	4	32	12	25
top	ak	aa	as	c	d	v	g	e
freq	432	826	1658	1900	4203	246	1073	274

```
In [53]: test_df_feature['X0'] = le.fit_transform(test_df_feature.X0)
         test_df_feature['X1'] = le.fit_transform(test_df_feature.X1)
         test_df_feature['X2'] = le.fit_transform(test_df_feature.X2)
         test_df_feature['X3'] = le.fit_transform(test_df_feature.X3)
         test_df_feature['X4'] = le.fit_transform(test_df_feature.X4)
         test_df_feature['X5'] = le.fit_transform(test_df_feature.X5)
         test_df_feature['X6'] = le.fit_transform(test_df_feature.X6)
         test_df_feature['X8'] = le.fit_transform(test_df_feature.X8)
```

```
In [54]: pca.fit(test_df_feature)

Out[54]: PCA(n_components=0.95)
```

```
In [55]: test_df_feature_trans = pca.fit_transform(test_df_feature)
         print(test_df_feature_trans.shape)

         (4209, 6)
```

```
In [56]: test_pred = model.predict(test_df_feature_trans)
         test_pred

Out[56]: array([[ 1.7273520e+01,  1.9237247e+01,  3.1263712e+01, ...,
                  4.5788325e-03,  2.2881597e-03,  1.6800487e-02],
                [ 3.4823170e+01,  5.2492523e+00,  1.0608750e+01, ...,
                  4.1679339e-03,  2.2881597e-03,  6.8906322e-03],
                [ 2.2416216e+01,  1.8002998e+01,  1.8460388e+01, ...,
                  3.8033123e-03,  2.2881597e-03,  2.3315079e-03],
                ...,
                [ 4.5763992e+01,  1.8715649e+01,  1.6286839e+01, ...,
                  4.1679339e-03,  2.2881597e-03,  6.8906322e-03],
                [ 2.5774267e+01,  1.5437351e+01,  1.5841972e+01, ...,
                  3.8033123e-03,  2.2881597e-03,  2.3315079e-03],
                [ 3.2438564e+01, -1.0689995e+00,  9.0051785e+00, ...,
                  5.0759236e-03,  2.2881597e-03,  2.3315079e-03]], dtype=float32)
```

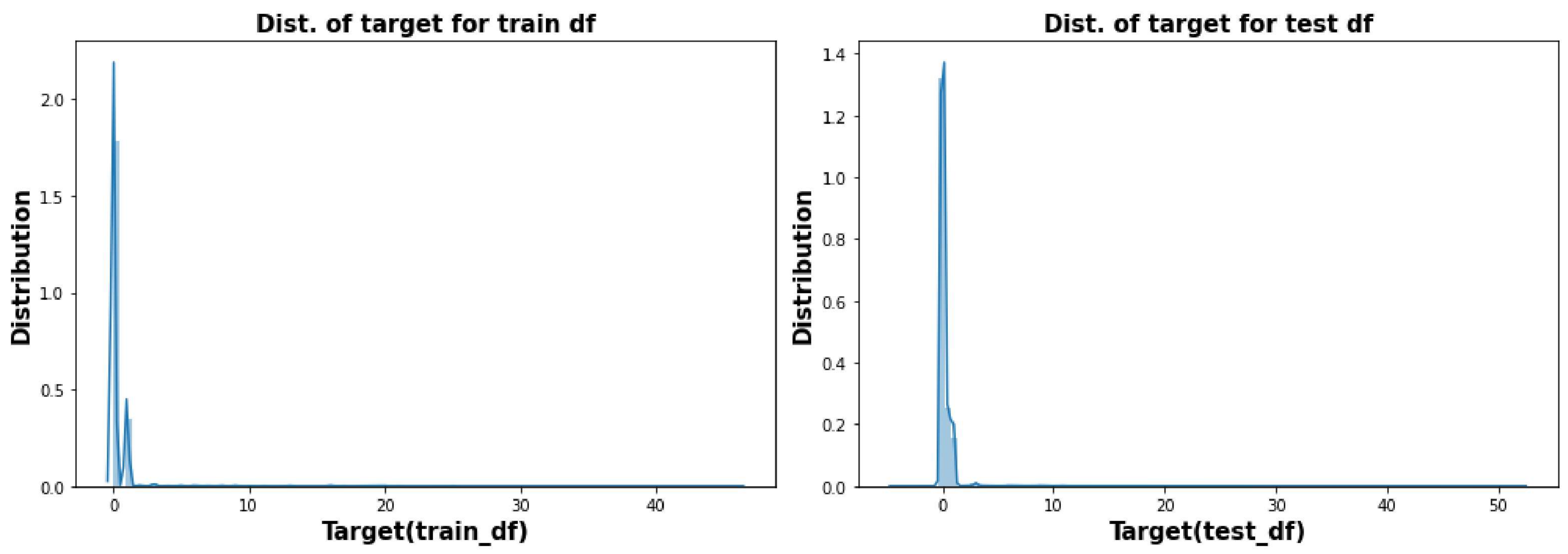
In [58]:
```
fig, ax = plt.subplots(1,2, figsize=(14,5))

train_plot = sns.distplot(train_df_target[train_df_target<200], bins=100, kde=True, ax=ax[0])
train_plot.set_xlabel('Target(train_df)', weight='bold', size=15)
train_plot.set_ylabel('Distribution', weight='bold', size=15)
train_plot.set_title('Dist. of target for train df', weight='bold', size=15)

test_plot = sns.distplot(test_pred[test_pred<200], bins=100, kde=True, ax=ax[1])
test_plot.set_xlabel('Target(test_df)', weight='bold', size=15)
test_plot.set_ylabel('Distribution', weight='bold', size=15)
test_plot.set_title('Dist. of target for test df', weight='bold', size=15)

plt.tight_layout()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [ ]: