```
In [40]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas_profiling
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
import xgboost as xgb
import seaborn as sns
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

EDA

```
In [41]:
df_train = pd.read_csv('mercedes_train.csv')

print('Size of training set: {} rows and {} columns'
      .format(*df_train.shape))
# print few rows and see how the data looks like
df_train.head()
```

```
Size of training set: 4209 rows and 378 columns
```

Out[41]:

	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 [42]:
df_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
```

```
In [43]:
df_train.shape
```

```
Out[43]: (4209, 378)
```

```
In [44]:
pfr = pandas_profiling.ProfileReport(df_train)
pfr.to_file("Descriptive_Analysis_train.html")
```

```
In [45]:
pfr
```

Pandas Profiling Report   Overview   Variables   Interactions   Correlations   Missing values   Sample

Overview

Overview   Reproduction   Warnings 159

Dataset statistics

Number of variables	378
Number of observations	4209
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	13.8 MiB
Average record size in memory	3.3 KiB

Variable types

BOOL	368
CAT	8
NUM	2

Variables

ID
Real number ($\mathbb{R}_{\geq 0}$)

Distinct count	4209

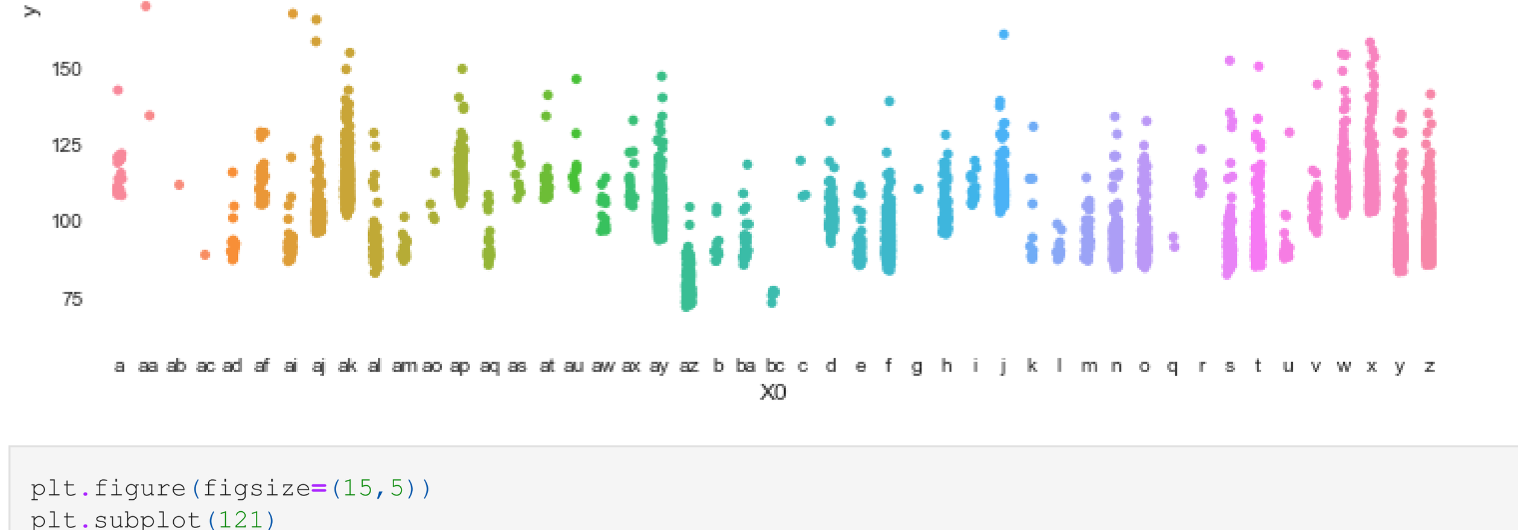
Mean	4205.96079828(
Minimum	0

Out[45]:

```
In [46]:
y_train = df_train['y'].values
y_train
```

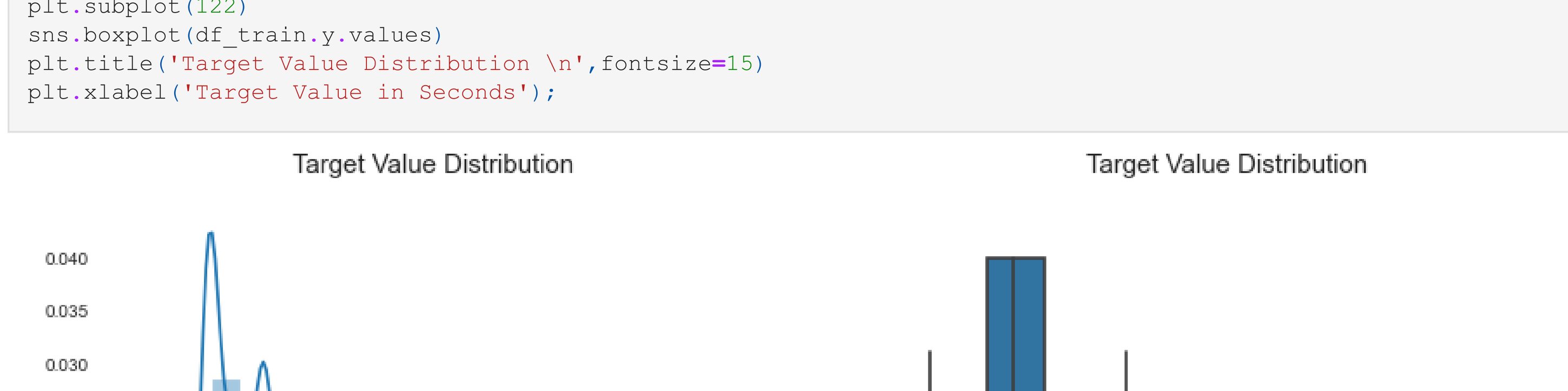
```
Out[46]: array([130.81,  88.53,  76.26, ..., 109.22,  87.48, 110.85])
```

```
In [47]:
var="X0"
colu_order=np.sort(df_train[var].unique()).tolist()
plt.figure(figsize=(12,6))
sns.stripplot(x=var,y="y",data=df_train,order=colu_order)
plt.xlabel(var,fontsize=12)
plt.ylabel("y",fontsize=12)
plt.title("Distribution of y variable with "+var, fontsize=15)
plt.show()
```




```
In [48]:
plt.figure(figsize=(15,5))
plt.subplot(121)
sns.distplot(df_train.y.values, bins=20)
plt.title('Target Value Distribution \n',fontsize=15)
plt.xlabel('Target Value in Seconds'); plt.ylabel('Occurances');

plt.subplot(122)
sns.boxplot(df_train.y.values)
plt.title('Target Value Distribution \n',fontsize=15)
plt.xlabel('Target Value in Seconds');
```




```
In [49]:
# Looking at the test datset for simiilar features
df_test = pd.read_csv('mercedes_test.csv')
df_test.head()
```

Out[49]:

	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 [50]:
df_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
```

```
In [51]:
df_test.shape
```

```
Out[51]: (4209, 377)
```

```
In [52]:
pfr1 = pandas_profiling.ProfileReport(df_test)
pfr1.to_file("Descriptive_Analysis_test.html")
```

```
In [53]:
pfr1
```

Pandas Profiling Report   Overview   Variables   Interactions   Correlations   Missing values   Sample

Overview

Overview   Reproduction   Warnings 157

Dataset statistics

Number of variables	377
Number of observations	4209
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	13.7 MiB
Average record size in memory	3.3 KiB

Variable types

BOOL	368
CAT	8
NUM	1

Variables

ID
Real number ($\mathbb{R}_{\geq 0}$)

Distinct count	4209

Mean	4211.03920171(
Minimum	1

Out[53]:

```
In [54]:
final_column = list(set(df_train.columns) - set(['ID', 'y']))

x_train = df_train[final_column]
# x_train
x_test = df_test[final_column]
# x_test
```

```
In [55]:
df_train.isnull().sum()
```

```
Out[55]:
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 378, dtype: int64
```

```
In [56]:
df_test.isnull().sum()
```

```
Out[56]:
ID      0
X0      0
X1      0
X2      0
X3      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 377, dtype: int64
```

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

```
In [57]:
for column in final_column:
    check = len(np.unique(x_train[column]))
    if check == 1:
        x_train.drop(column, axis = 1)
        x_test.drop(column, axis = 1)
    if check > 2: # Column is categorical; hence mapping to ordinal measure of value
        mapit = lambda x: sum([ord(digit) for digit in x])
        x_train[column] = x_train[column].apply(mapit)
        x_test[column] = x_test[column].apply(mapit)

x_train.head()
```

Out[57]:

	X18	X300	X163	X38	X80	X171	X368	X53	X269	X48	...	X290	X264	X306	X105	X355	X116	X243	X85	X94	X33
0	1	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	1	0	1	0	0
1	1	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
2	0	0	1	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 376 columns

Performing dimensionality reduction with principal components analysis

```
In [58]:
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)
```

Splitting the Dataset

```
In [59]:
x_train, x_valid, y_train, y_valid = train_test_split(pca_result_train, y_train, test_size = 0.2, random_state
```

```
In [60]:
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)

params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.02
params['max_depth'] = 4
```

```
In [61]:
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_
```

```
[15:56:36] WARNING: c:\ci\xgboost-split_1619728435298\work\src\objective\regression_obj.cu:170: reg:linear is n
ow deprecated in favor of reg:squarederror.
[0]	train-rmse:98.99695	train-r2:-59.49733	valid-rmse:98.88884	valid-r2:-61.82690
[10]	train-rmse:81.14410	train-r2:-39.64492	valid-rmse:81.07848	valid-r2:-41.23399
[20]	train-rmse:66.59753	train-r2:-26.37844	valid-rmse:66.55611	valid-r2:-27.45948
[30]	train-rmse:54.75785	train-r2:-17.50910	valid-rmse:54.73342	valid-r2:-18.24670
[40]	train-rmse:45.14008	train-r2:-11.57816	valid-rmse:45.13794	valid-r2:-12.08984
[50]	train-rmse:37.34669	train-r2:-7.60987	valid-rmse:37.35502	valid-r2:-7.96497
[60]	train-rmse:31.04961	train-r2:-4.95120	valid-rmse:31.08432	valid-r2:-5.20775
[70]	train-rmse:25.98398	train-r2:-3.16777	valid-rmse:26.03103	valid-r2:-3.35345
[80]	train-rmse:21.93872	train-r2:-1.97108	valid-rmse:21.99812	valid-r2:-2.10901
[90]	train-rmse:18.73534	train-r2:-1.16678	valid-rmse:18.81639	valid-r2:-1.27470
[100]	train-rmse:16.22450	train-r2:-0.62493	valid-rmse:16.33551	valid-r2:-0.71442
[110]	train-rmse:14.28563	train-r2:-0.25977	valid-rmse:14.42642	valid-r2:-0.33711
[120]	train-rmse:12.81326	train-r2:-0.01347	valid-rmse:12.98237	valid-r2:-0.08283
[130]	train-rmse:11.69547	train-r2:0.15564	valid-rmse:11.90197	valid-r2:0.08990
[140]	train-rmse:10.86897	train-r2:0.27076	valid-rmse:11.11649	valid-r2:0.20606
[150]	train-rmse:10.25092	train-r2:0.35134	valid-rmse:10.54881	valid-r2:0.28508
[160]	train-rmse:9.79518	train-r2:0.40773	valid-rmse:10.15443	valid-r2:0.33754
[170]	train-rmse:9.46583	train-r2:0.44689	valid-rmse:9.87627	valid-r2:0.37333
[180]	train-rmse:9.21180	train-r2:0.47618	valid-rmse:9.68326	valid-r2:0.39759
[190]	train-rmse:9.01729	train-r2:0.49807	valid-rmse:9.54228	valid-r2:0.41500
[200]	train-rmse:8.87619	train-r2:0.51365	valid-rmse:9.43849	valid-r2:0.42766
[210]	train-rmse:8.77925	train-r2:0.52422	valid-rmse:9.37426	valid-r2:0.43542
[220]	train-rmse:8.70048	train-r2:0.53272	valid-rmse:9.33206	valid-r2:0.44049
[230]	train-rmse:8.63702	train-r2:0.53951	valid-rmse:9.30257	valid-r2:0.44402
[240]	train-rmse:8.57832	train-r2:0.54575	valid-rmse:9.27772	valid-r2:0.44699
[250]	train-rmse:8.52980	train-r2:0.55087	valid-rmse:9.26267	valid-r2:0.44878
[260]	train-rmse:8.49648	train-r2:0.55437	valid-rmse:9.25403	valid-r2:0.44981
[270]	train-rmse:8.46473	train-r2:0.55770	valid-rmse:9.24233	valid-r2:0.45120
[280]	train-rmse:8.43379	train-r2:0.56093	valid-rmse:9.23400	valid-r2:0.45219
[290]	train-rmse:8.40086	train-r2:0.56435	valid-rmse:9.22974	valid-r2:0.45269
[300]	train-rmse:8.37398	train-r2:0.56713	valid-rmse:9.22219	valid-r2:0.45359
[310]	train-rmse:8.34338	train-r2:0.57029	valid-rmse:9.22097	valid-r2:0.45373
[320]	train-rmse:8.31196	train-r2:0.57352	valid-rmse:9.21900	valid-r2:0.45397
[330]	train-rmse:8.28336	train-r2:0.57645	valid-rmse:9.21562	valid-r2:0.45437
[340]	train-rmse:8.25519	train-r2:0.57932	valid-rmse:9.21509	valid-r2:0.45443
[350]	train-rmse:8.22063	train-r2:0.58284	valid-rmse:9.21484	valid-r2:0.45446
[360]	train-rmse:8.19075	train-r2:0.58587	valid-rmse:9.21704	valid-r2:0.45420
[370]	train-rmse:8.16890	train-r2:0.58807	valid-rmse:9.21725	valid-r2:0.45417
[380]	train-rmse:8.14266	train-r2:0.59072	valid-rmse:9.22060	valid-r2:0.45378
[390]	train-rmse:8.10716	train-r2:0.59428	valid-rmse:9.22203	valid-r2:0.45361
[400]	train-rmse:8.08520	train-r2:0.59647	valid-rmse:9.22419	valid-r2:0.45335
[403]	train-rmse:8.08132	train-r2:0.59686	valid-rmse:9.22592	valid-r2:0.45315
```

```
In [62]:
p_test = P.predict(f_test)
p_test
```

```
Out[62]: array([ 79.14215 ,  96.457306,  81.29516 , ...,  98.5201  , 107.15611 ,
        95.1247  ], dtype=float32)
```

```
In [63]:
Predicted_Data = pd.DataFrame()
Predicted_Data['y'] = p_test
Predicted_Data.head()
```

Out[63]:

	y
0	79.142151
1	96.457306
2	81.295158
3	77.608681
4	109.836014