Mercedes-Benz

June 20, 2023

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
train=pd.read_csv('train.csv')
```

[3]:
```
train.head()
```

[3]:

	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]

[4]:
```
pd.options.display.float_format = '{:,.4f}'.format
```

[5]:
```
variance = train.var()
variance = variance.reset_index()
variance.columns = ["id","values"]
variance= variance.sort_values("values",ascending=1)
variance
```

[5]:

	id	values
275	X289	0.0000
315	X330	0.0000
254	X268	0.0000
332	X347	0.0000
97	X107	0.0000

```
..    …             …
347  X362         0.2496
322  X337         0.2498
116  X127         0.2500
1       y       160.7667
0      ID 5,941,936.1180

[370 rows x 2 columns]
```

[6]:
```
# We will remove the varialbes with variance 0 and low variance (less than 0.2)
# We will also remove id since it has a huge variance
var = variance.loc[variance["values"] < 0.2,"id"]
data1 = train.drop(var,axis=1)
data1.drop("ID",axis=1,inplace=True)
data1.head()
```

[6]:
```
         y  X0 X1  X2 X3 X4 X5 X6 X8  X14  …  X329  X334  X337  X350  X351  \
0 130.8100   k  v  at  a  d  u  j  o    0  …     1     1     0     0     0
1  88.5300   k  t  av  e  d  y  l  o    0  …     1     0     1     0     0
2  76.2600  az  w   n  c  d  x  j  x    0  …     0     1     0     1     0
3  80.6200  az  t   n  f  d  x  l  e    0  …     0     0     0     1     0
4  78.0200  az  v   n  f  d  h  d  n    0  …     0     1     0     1     0

   X355  X358  X362  X375  X377
0     0     0     0     0     1
1     0     0     0     1     0
2     0     1     0     0     0
3     0     1     0     0     0
4     0     1     0     0     0

[5 rows x 55 columns]
```

[7]:
```
# Missing values
data1[data1.isnull().any(axis=1)]
# We don't have any missing values in the data
```

[7]:
```
Empty DataFrame
Columns: [y, X0, X1, X2, X3, X4, X5, X6, X8, X14, X27, X46, X51, X58, X64, X85,
X100, X115, X118, X119, X127, X132, X137, X156, X157, X163, X171, X178, X186,
X187, X191, X194, X218, X220, X223, X224, X246, X250, X251, X261, X273, X311,
X313, X314, X324, X329, X334, X337, X350, X351, X355, X358, X362, X375, X377]
Index: []

[0 rows x 55 columns]
```

[8]:
```
c = data1.corr().abs()
s = c.unstack()
```

```
s = pd.DataFrame(s)
s.reset_index(inplace = True)
s.head()
```

[8]:
```
  level_0 level_1      0
0       y       y 1.0000
1       y     X14 0.1936
2       y     X27 0.0535
3       y     X46 0.1360
4       y     X51 0.2300
```

[9]:
```
s["flag"] = np.where(s["level_0"] == s["level_1"],"same","not same")
s.columns.values[2] = "corr"
s.head()
```

[9]:
```
  level_0 level_1   corr      flag
0       y       y 1.0000      same
1       y     X14 0.1936  not same
2       y     X27 0.0535  not same
3       y     X46 0.1360  not same
4       y     X51 0.2300  not same
```

[10]:
```
# Remove the variables with correlation more than .9
#s.loc[s["flag"] != "same",]
name = s.loc[(s["corr"] > .9) & (s["flag"] != "same") ,"level_1"]
final_name = name.unique()
```

[11]:
```
data2 = data1.drop(final_name,axis=1)
data2.head()
```

[11]:
```
         y  X0 X1  X2 X3 X4 X5 X6 X8  X27  …  X223  X224  X273  X313  X329  \
0 130.8100   k  v  at  a  d  u  j  o    0  …     0     0     1     0     1
1  88.5300   k  t  av  e  d  y  l  o    1  …     0     0     1     0     1
2  76.2600  az  w   n  c  d  x  j  x    1  …     1     1     1     0     0
3  80.6200  az  t   n  f  d  x  l  e    1  …     1     0     1     0     0
4  78.0200  az  v   n  f  d  h  d  n    1  …     1     0     1     0     0

   X350  X351  X355  X375  X377
0     0     0     0     0     1
1     0     0     0     1     0
2     1     0     0     0     0
3     1     0     0     0     0
4     1     0     0     0     0

[5 rows x 33 columns]
```

[12]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4209 entries, 0 to 4208
Data columns (total 33 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   y       4209 non-null   float64
 1   X0      4209 non-null   object
 2   X1      4209 non-null   object
 3   X2      4209 non-null   object
 4   X3      4209 non-null   object
 5   X4      4209 non-null   object
 6   X5      4209 non-null   object
 7   X6      4209 non-null   object
 8   X8      4209 non-null   object
 9   X27     4209 non-null   int64
 10  X46     4209 non-null   int64
 11  X51     4209 non-null   int64
 12  X64     4209 non-null   int64
 13  X85     4209 non-null   int64
 14  X100    4209 non-null   int64
 15  X115    4209 non-null   int64
 16  X127    4209 non-null   int64
 17  X132    4209 non-null   int64
 18  X163    4209 non-null   int64
 19  X171    4209 non-null   int64
 20  X191    4209 non-null   int64
 21  X218    4209 non-null   int64
 22  X220    4209 non-null   int64
 23  X223    4209 non-null   int64
 24  X224    4209 non-null   int64
 25  X273    4209 non-null   int64
 26  X313    4209 non-null   int64
 27  X329    4209 non-null   int64
 28  X350    4209 non-null   int64
 29  X351    4209 non-null   int64
 30  X355    4209 non-null   int64
 31  X375    4209 non-null   int64
 32  X377    4209 non-null   int64
dtypes: float64(1), int64(24), object(8)
memory usage: 1.1+ MB
```

[13]:
```
char = data2.select_dtypes(exclude='number')
num = data2.select_dtypes(include='number')
```

[14]:
```
char1 = pd.get_dummies(char.astype(str),drop_first=True)
char1.head()
```

[14]:
```
   X0_aa  X0_ab  X0_ac  X0_ad  X0_af  X0_ai  X0_aj  X0_ak  X0_al  X0_am  … \
0      0      0      0      0      0      0      0      0      0      0  …
1      0      0      0      0      0      0      0      0      0      0  …
2      0      0      0      0      0      0      0      0      0      0  …
3      0      0      0      0      0      0      0      0      0      0  …
4      0      0      0      0      0      0      0      0      0      0  …

   X8_p  X8_q  X8_r  X8_s  X8_t  X8_u  X8_v  X8_w  X8_x  X8_y
0     0     0     0     0     0     0     0     0     0     0
1     0     0     0     0     0     0     0     0     0     0
2     0     0     0     0     0     0     0     0     1     0
3     0     0     0     0     0     0     0     0     0     0
4     0     0     0     0     0     0     0     0     0     0

[5 rows x 187 columns]
```

[15]:
```
data3 = pd.concat([char1,num],axis=1)
data3.head()
```

[15]:
```
   X0_aa  X0_ab  X0_ac  X0_ad  X0_af  X0_ai  X0_aj  X0_ak  X0_al  X0_am  … \
0      0      0      0      0      0      0      0      0      0      0  …
1      0      0      0      0      0      0      0      0      0      0  …
2      0      0      0      0      0      0      0      0      0      0  …
3      0      0      0      0      0      0      0      0      0      0  …
4      0      0      0      0      0      0      0      0      0      0  …

   X223  X224  X273  X313  X329  X350  X351  X355  X375  X377
0     0     0     1     0     1     0     0     0     0     1
1     0     0     1     0     1     0     0     0     1     0
2     1     1     1     0     0     1     0     0     0     0
3     1     0     1     0     0     1     0     0     0     0
4     1     0     1     0     0     1     0     0     0     0

[5 rows x 212 columns]
```

APPLY XGBOOST MODEL

[16]:
```
#segregating ind and dep var
X = data3.drop("y",axis=1)
y = data3.loc[:,"y"]
```

[18]:
```
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y,test_size = 0.3,␣
 ↪random_state = 123)
```

[ ]:
```
import xgboost as xg
```

```
xgb_r = xg.XGBRegressor(objective ='reg:squarederror',n_estimators = 10, seed =␣
↪123)
# Fitting the model
xgb_r.fit(train_X, train_y)
```

[ ]:
```
XGBRegressor(base_score=0.5, booster=None, colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints=None,
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints=None,
             n_estimators=10, n_jobs=0, num_parallel_tree=1, random_state=123,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=123,
             subsample=1, tree_method=None, validate_parameters=False,
             verbosity=None)
```

[20]:
```
# Predict the model
y_pred = xgb_r.predict(test_X)
```

[21]:
```
d = pd.DataFrame()
d["test_y"] = test_y
d["y_pred"] = y_pred
d["mp"] = abs((d["test_y"]- d["y_pred"])/d["test_y"])
(d.mp.mean())*100
```

[21]:
```
4.782253811935617
```

[ ]:
```

```