Mercedes-Benz Greener Manufacturing (Machine Learning Project)

September 27, 2022

[1]:
```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from math import sqrt
```

[2]:
```
# read data into a DataFrame
df_train_data = pd.read_csv('train.csv')
df_test_data = pd.read_csv('test.csv')
```

[3]:
```
print('Train dataset: \n')
df_train_data.head()
```

```
Train dataset:
```

[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]:
```
print('Test dataset: \n')
df_test_data.head()
```

```
Test dataset:
```

[4]:
```
   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]
```

[5]:
```
df_train_data = df_train_data.drop(['ID'], axis = 1)
df_test_data = df_test_data.drop(['ID'], axis = 1)
```

[6]:
```
print('Train dataset: \n')
df_train_data.head()
```

```
Train dataset:
```

[6]:
```
        y  X0 X1  X2 X3 X4 X5 X6 X8  X10  …  X375  X376  X377  X378  X379  \
0  130.81   k  v  at  a  d  u  j  o    0  …     0     0     1     0     0
1   88.53   k  t  av  e  d  y  l  o    0  …     1     0     0     0     0
2   76.26  az  w   n  c  d  x  j  x    0  …     0     0     0     0     0
3   80.62  az  t   n  f  d  x  l  e    0  …     0     0     0     0     0
4   78.02  az  v   n  f  d  h  d  n    0  …     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 377 columns]
```

[7]:
```
print('Test dataset: \n')
df_test_data.head()
```

```
Test dataset:
```

[7]:
```
   X0 X1  X2 X3 X4 X5 X6 X8  X10  X11  …  X375  X376  X377  X378  X379  \
0  az  v   n  f  d  t  a  w    0    0  …     0     0     0     1     0
1   t  b  ai  a  d  b  g  y    0    0  …     0     0     1     0     0
2  az  v  as  f  d  a  j  j    0    0  …     0     0     0     1     0
3  az  l   n  f  d  z  l  n    0    0  …     0     0     0     1     0
4   w  s  as  c  d  y  i  m    0    0  …     1     0     0     0     0

   X380  X382  X383  X384  X385
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

[5 rows x 376 columns]
```

[8]:
```
columns_with_zero_var = df_train_data.var()[df_train_data.var()==0].index.values
columns_with_zero_var
```

[8]:
```
array(['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290',
       'X293', 'X297', 'X330', 'X347'], dtype=object)
```

[9]:
```
df_train_data = df_train_data.
 ↪drop(['X11','X93','X107','X233','X235','X268','X289','X290','X293','X297','X330','X347'],␣
 ↪axis = 1)
df_test_data = df_test_data.
 ↪drop(['X11','X93','X107','X233','X235','X268','X289','X290','X293','X297','X330','X347'],␣
 ↪axis = 1)
```

[10]:
```
df_train_data.shape
```

[10]:
```
(4209, 365)
```

[11]:
```
df_test_data.shape
```

[11]:
```
(4209, 364)
```

[12]:
```
#check for null values in train dataset
np.sum(df_train_data.isnull().sum())
```

[12]:
```
0
```

[13]:
```
#check for null values in test dataset
np.sum(df_test_data.isnull().sum())
```

[13]:
```
0
```

[14]:
```
#check for unique values in train dataset
np.sum(df_train_data.nunique().sum())
```

[14]:
```
3452
```

[15]:
```
#check for unique values in train dataset
np.sum(df_test_data.nunique().sum())
```

[15]:
```
908
```

[16]:
```
# find the columns having datatype as object
object_columns = df_train_data.describe(include=[object]).columns.values
object_columns
```

[16]:
```
array(['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8'], dtype=object)
```

[17]:
```
le = LabelEncoder()
for col in object_columns:
    le.fit(df_train_data[col].append(df_test_data[col]).values)
    df_train_data[col] = le.transform(df_train_data[col])
    df_test_data[col] = le.transform(df_test_data[col])
```

[19]:
```
# create X and y
X = df_train_data.drop(['y'], axis=1)
y = df_train_data.y
#create train and test split
from sklearn import model_selection
xtrain,xtest,ytrain,ytest = model_selection.train_test_split(X,y,test_size=0.
 ↪3,random_state=1)
```

[20]:
```
from sklearn.decomposition import PCA as sklearnPCA
pca = sklearnPCA(0.98, svd_solver='full')
sklearn_pca = pca.fit(X)
```

[21]:
```
sklearn_pca.n_components_
```

[21]:
```
12
```

[22]:
```
sklearn_pca.explained_variance_ratio_
```

[22]:
```
array([0.40868988, 0.21758508, 0.13120081, 0.10783522, 0.08165248,
       0.0140934 , 0.00660951, 0.00384659, 0.00260289, 0.00214378,
       0.00209857, 0.00180388])
```

[23]:
```
pca_xtrain = pd.DataFrame(sklearn_pca.transform(xtrain))
pca_xtest = pd.DataFrame(sklearn_pca.transform(xtest))
pca_df_test_data = pd.DataFrame(sklearn_pca.transform(df_test_data))
```

[24]:
```
import xgboost as xgb
xgb_regressor_model = xgb.XGBRegressor(objective = 'reg:squarederror',␣
 ↪learning_rate= 0.1 )
xgb_regressor_model.fit(pca_xtrain, ytrain)
```

[24]:
```
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=0.1, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method=None, validate_parameters=False, verbosity=None)
```

[25]:
```
xgb_regressor_model_predicted_y_test = xgb_regressor_model.predict(pca_xtest)
print(sqrt(mean_squared_error(ytest, xgb_regressor_model_predicted_y_test)))
```

```
8.555776331737889
```

[26]:
```
xgb_RFregressor_model = xgb.XGBRFRegressor(objective = 'reg:squarederror',␣
 ↪learning_rate= 1)
xgb_RFregressor_model.fit(pca_xtrain, ytrain)
```

[26]:
```
XGBRFRegressor(base_score=0.5, booster=None, colsample_bylevel=1,
               colsample_bytree=1, gamma=0, gpu_id=-1, importance_type='gain',
               interaction_constraints=None, max_delta_step=0, max_depth=6,
               min_child_weight=1, missing=nan, monotone_constraints=None,
               n_estimators=100, n_jobs=0, num_parallel_tree=100,
               objective='reg:squarederror', random_state=0, reg_alpha=0,
               scale_pos_weight=1, tree_method=None, validate_parameters=False,
               verbosity=None)
```

[27]:
```
xgb_RFregressor_model_predicted_y_test = xgb_RFregressor_model.
 ↪predict(pca_xtest)
print(sqrt(mean_squared_error(ytest, xgb_RFregressor_model_predicted_y_test)))
```

```
9.131954308021706
```

[ ]:
```

```