Mercedes-Benz Greener Manufacturing Machine Learning Project

September 3, 2022

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

[2]:
```
import pandas as pd
```

[3]:
```
from sklearn.decomposition import PCA
```

[4]:
```
df_train = pd.read_csv('train.csv')
```

[5]:
```
print('Size of training set: {} rows and {} columns'
      .format(*df_train.shape))
```

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

[6]:
```
df_train.head()
```

[6]:

	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]

[7]:
```
y_train = df_train['y'].values
```

[8]:
```
cols = [c for c in df_train.columns if 'X' in c]
```

[9]:
```
print('Number of features: {}'.format(len(cols)))
```

```
Number of features: 376
```

[10]:
```
print('Feature types:')
```

```
Feature types:
```

[11]:
```
df_train[cols].dtypes.value_counts()
```

[11]:
```
int64     368
object      8
dtype: int64
```

[12]:
```
counts = [[], [], []]
```

[13]:
```
for c in cols:
    typ = df_train[c].dtype
    uniq = len(np.unique(df_train[c]))
    if uniq == 1:
        counts[0].append(c)
    elif uniq == 2 and typ == np.int64:
        counts[1].append(c)
    else:
        counts[2].append(c)
```

[14]:
```
print('Constant features: {} Binary features: {} Categorical features: {}\n'
      .format(*[len(c) for c in counts]))
```

```
Constant features: 12 Binary features: 356 Categorical features: 8
```

[15]:
```
print('Constant features:', counts[0])
```

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

[16]:
```
print('Categorical features:', counts[2])
```

```
Categorical features: ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

[17]:
```
df_test = pd.read_csv('test.csv')
```

[18]:
```
usable_columns = list(set(df_train.columns) - set(['ID', 'y']))
```

[19]:
```
y_train = df_train['y'].values
```

[20]:
```
id_test = df_test['ID'].values
```

[21]:
```
x_train = df_train[usable_columns]
```

[22]:
```
x_test = df_test[usable_columns]
```

[23]:
```
def check_missing_values(df):
```

```
  File "<ipython-input-23-4f229d394824>", line 1
    def check_missing_values(df):
                                 ^
SyntaxError: unexpected EOF while parsing
```

[25]:
```
if df.isnull().any().any():
        print("There are missing values in the dataframe")
    else:
        print("There are no missing values in the dataframe")
```

```
  File "<tokenize>", line 3
    else:
    ^
IndentationError: unindent does not match any outer indentation level
```

[26]:
```
check_missing_values(x_train)
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-26-9ff1988a14a9> in <module>
----> 1 check_missing_values(x_train)

NameError: name 'check_missing_values' is not defined
```

[30]:
```
check_missing_values(x_test)
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)
```

```
        <ipython-input-30-c755e7842d9f> in <module>
    ----> 1 check_missing_values(x_test)


        NameError: name 'check_missing_values' is not defined
```

[29]:
```
for column in usable_columns:
    cardinality = len(np.unique(x_train[column]))
    if cardinality == 1:
        x_train.drop(column, axis=1) # Column with only one
        # value is useless so we drop it
        x_test.drop(column, axis=1)
    if cardinality > 2: # Column is categorical
        mapper = lambda x: sum([ord(digit) for digit in x])
        x_train[column] = x_train[column].apply(mapper)
        x_test[column] = x_test[column].apply(mapper)
x_train.head()
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

[29]:

	X55	X238	X277	X144	X215	X323	X8	X308	X106	X165	…	X280	X342	\
0	0	0	0	1	0	0	111	0	0	0	…	0	0	
1	0	1	0	1	0	0	111	0	0	1	…	0	0	
2	0	0	0	1	0	0	120	0	0	0	…	0	0	
3	0	0	0	1	0	0	101	0	0	0	…	0	0	
4	0	0	0	1	0	0	110	0	0	0	…	0	0	

	X198	X108	X254	X159	X205	X109	X54	X256
0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0
2	0	0	0	0	1	0	1	1

```
3     0     1     0     0     1     0     1     1
4     0     1     0     0     1     0     1     1

[5 rows x 376 columns]
```

[31]:
```
print('Feature types:')
x_train[cols].dtypes.value_counts()
```

```
Feature types:
```

[31]:
```
int64    376
dtype: int64
```

[32]:
```
n_comp = 12
pca = PCA(n_components=n_comp, random_state=420)
pca2_results_train = pca.fit_transform(x_train)
pca2_results_test = pca.transform(x_test)
```

[33]:
```
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

[34]:
```
x_train, x_valid, y_train, y_valid = train_test_split(
        pca2_results_train,
        y_train, test_size=0.2,
        random_state=4242)
```

[35]:
```
d_train = xgb.DMatrix(x_train, label=y_train)
d_valid = xgb.DMatrix(x_valid, label=y_valid)
#d_test = xgb.DMatrix(x_test)
d_test = xgb.DMatrix(pca2_results_test)

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

def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)

watchlist = [(d_train, 'train'), (d_valid, 'valid')]

clf = xgb.train(params, d_train,
                1000, watchlist, early_stopping_rounds=50,
                feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

```
[18:57:10] WARNING: /workspace/src/objective/regression_obj.cu:167: reg:linear
```

```
is now deprecated in favor of reg:squarederror.
[0]     train-rmse:99.14835     valid-rmse:98.26297     train-r2:-58.35295
valid-r2:-67.63754
Multiple eval metrics have been passed: 'valid-r2' will be used for early
stopping.

Will train until valid-r2 hasn't improved in 50 rounds.
[10]    train-rmse:81.27653     valid-rmse:80.36433     train-r2:-38.88428
valid-r2:-44.91014
[20]    train-rmse:66.71610     valid-rmse:65.77334     train-r2:-25.87403
valid-r2:-29.75260
[30]    train-rmse:54.86957     valid-rmse:53.88974     train-r2:-17.17752
valid-r2:-19.64401
[40]    train-rmse:45.24491     valid-rmse:44.21971     train-r2:-11.35979
valid-r2:-12.89996
[50]    train-rmse:37.44729     valid-rmse:36.37237     train-r2:-7.46666
valid-r2:-8.40428
[60]    train-rmse:31.14749     valid-rmse:30.01874     train-r2:-4.85757
valid-r2:-5.40571
[70]    train-rmse:26.08662     valid-rmse:24.90890     train-r2:-3.10872
valid-r2:-3.41053
[80]    train-rmse:22.04639     valid-rmse:20.83274     train-r2:-1.93458
valid-r2:-2.08514
[90]    train-rmse:18.84412     valid-rmse:17.60176     train-r2:-1.14399
valid-r2:-1.20239
[100]   train-rmse:16.33254     valid-rmse:15.08443     train-r2:-0.61057
valid-r2:-0.61748
[110]   train-rmse:14.39917     valid-rmse:13.14817     train-r2:-0.25183
valid-r2:-0.22889
[120]   train-rmse:12.91792     valid-rmse:11.69199     train-r2:-0.00753
valid-r2:0.02824
[130]   train-rmse:11.80472     valid-rmse:10.61579     train-r2:0.15864
valid-r2:0.19890
[140]   train-rmse:10.97691     valid-rmse:9.85006      train-r2:0.27250
valid-r2:0.31030
[150]   train-rmse:10.37734     valid-rmse:9.31191      train-r2:0.34980
valid-r2:0.38360
[160]   train-rmse:9.93061      valid-rmse:8.95330      train-r2:0.40458
valid-r2:0.43017
[170]   train-rmse:9.59561      valid-rmse:8.71472      train-r2:0.44407
valid-r2:0.46013
[180]   train-rmse:9.34972      valid-rmse:8.55323      train-r2:0.47220
valid-r2:0.47995
[190]   train-rmse:9.16375      valid-rmse:8.44709      train-r2:0.49299
valid-r2:0.49278
[200]   train-rmse:9.01946      valid-rmse:8.38309      train-r2:0.50883
valid-r2:0.50044
[210]   train-rmse:8.91377      valid-rmse:8.34102      train-r2:0.52027
```

```
valid-r2:0.50544
[220]   train-rmse:8.83503      valid-rmse:8.31985      train-r2:0.52871
valid-r2:0.50795
[230]   train-rmse:8.77307      valid-rmse:8.30457      train-r2:0.53530
valid-r2:0.50975
[240]   train-rmse:8.72671      valid-rmse:8.29923      train-r2:0.54020
valid-r2:0.51038
[250]   train-rmse:8.68879      valid-rmse:8.29590      train-r2:0.54418
valid-r2:0.51077
[260]   train-rmse:8.65270      valid-rmse:8.29052      train-r2:0.54796
valid-r2:0.51141
[270]   train-rmse:8.62293      valid-rmse:8.28677      train-r2:0.55107
valid-r2:0.51185
[280]   train-rmse:8.59338      valid-rmse:8.28711      train-r2:0.55414
valid-r2:0.51181
[290]   train-rmse:8.56763      valid-rmse:8.29032      train-r2:0.55681
valid-r2:0.51143
[300]   train-rmse:8.53982      valid-rmse:8.28897      train-r2:0.55968
valid-r2:0.51159
[310]   train-rmse:8.51397      valid-rmse:8.28885      train-r2:0.56234
valid-r2:0.51161
[320]   train-rmse:8.49135      valid-rmse:8.28842      train-r2:0.56466
valid-r2:0.51166
Stopping. Best iteration:
[270]   train-rmse:8.62293      valid-rmse:8.28677      train-r2:0.55107
valid-r2:0.51185
```

[36]:
```
p_test = clf.predict(d_test)

sub = pd.DataFrame()
sub['ID'] = id_test
sub['y'] = p_test
sub.to_csv('xgb.csv', index=False)

sub.head()
```

[36]:

	ID	y
0	1	82.633644
1	2	97.214027
2	3	83.423355
3	4	77.185326
4	5	112.083260

[ ]: