Project 1 ML

February 27, 2023

[1]:
```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
```

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

[3]:
```
print('Size of training set: {} rows and {} columns'
      .format(*df_train.shape))
```

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

[4]:
```
df_train.head()
```

[4]:

	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]

[5]:
```
y_train = df_train['y'].values
```

[6]:
```
cols = [c for c in df_train.columns if 'X' in c]
print('Number of features: {}'.format(len(cols)))
```

```
Number of features: 376
```

[7]:
```
df_train[cols].dtypes.value_counts()
```

[7]:
```
int64     368
object      8
dtype: int64
```

[8]:
```
counts = [[], [], []]
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)

print('Constant features: {} Binary features: {} Categorical features: {}\n'
      .format(*[len(c) for c in counts]))
print('Constant features:', counts[0])
print('Categorical features:', counts[2])
```

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

Constant features: ['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289',
'X290', 'X293', 'X297', 'X330', 'X347']
Categorical features: ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

[9]:
```
df_test = pd.read_csv('test.csv')
```

[10]:
```
usable_columns = list(set(df_train.columns) - set(['ID', 'y']))
y_train = df_train['y'].values
id_test = df_test['ID'].values
```

[11]:
```
x_train = df_train[usable_columns]
x_test = df_test[usable_columns]
```

[14]:
```
def check_missing_values(df):
    if df.isnull().any().any():
        print("There are missing values in the dataframe")
    else:
        print("There are no missing values in the dataframe")
check_missing_values(x_train)
check_missing_values(x_test)
```

```
There are no missing values in the dataframe
There are no missing values in the dataframe
```

[15]:
```
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.
```

[15]:

	X385	X212	X297	X96	X362	X288	X349	X322	X226	X346	…	X159	X313	\
0	0	0	0	0	0	0	0	0	0	0	…	0	0	
1	0	0	0	1	0	0	0	0	0	0	…	0	0	
2	0	0	0	1	0	0	0	0	0	0	…	0	0	
3	0	0	0	1	0	0	0	0	0	0	…	0	0	
4	0	0	0	1	0	0	0	0	0	0	…	0	0	

	X47	X59	X211	X29	X200	X222	X2	X148
0	0	0	0	0	0	0	213	0
1	0	0	0	0	0	0	215	0
2	0	0	0	1	0	0	110	1
3	0	0	0	1	0	0	110	1
4	0	0	0	1	0	0	110	1

[5 rows x 376 columns]

[16]:
```
print('Feature types:')
x_train[cols].dtypes.value_counts()
```

```
Feature types:
```

[16]:
```
int64    376
dtype: int64
```

[17]:
```
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)
```

[19]:
```
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

x_train, x_valid, y_train, y_valid = train_test_split(
        pca2_results_train,
        y_train, test_size=0.2,
        random_state=4242)
```

[20]:
```
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)
```

[21]:
```
params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.02
params['max_depth'] = 4
```

[22]:
```
def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)
```

[23]:
```
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)
```

```
[03:07:34] 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:54.86914     valid-rmse:53.89102     train-r2:-17.17724
valid-r2:-19.64500
[40]    train-rmse:45.24563     valid-rmse:44.22213     train-r2:-11.36018
valid-r2:-12.90149
[50]    train-rmse:37.44741     valid-rmse:36.37738     train-r2:-7.46671
valid-r2:-8.40687
[60]    train-rmse:31.15105     valid-rmse:30.01744     train-r2:-4.85891
valid-r2:-5.40515
[70]    train-rmse:26.08768     valid-rmse:24.90811     train-r2:-3.10906
valid-r2:-3.41026
[80]    train-rmse:22.04897     valid-rmse:20.82466     train-r2:-1.93527
valid-r2:-2.08275
[90]    train-rmse:18.84762     valid-rmse:17.59923     train-r2:-1.14479
valid-r2:-1.20176
[100]   train-rmse:16.33632     valid-rmse:15.08798     train-r2:-0.61131
valid-r2:-0.61824
[110]   train-rmse:14.40530     valid-rmse:13.15865     train-r2:-0.25290
valid-r2:-0.23085
[120]   train-rmse:12.93736     valid-rmse:11.70636     train-r2:-0.01056
valid-r2:0.02585
[130]   train-rmse:11.82038     valid-rmse:10.63217     train-r2:0.15640
valid-r2:0.19643
[140]   train-rmse:10.98619     valid-rmse:9.86382      train-r2:0.27127
valid-r2:0.30837
[150]   train-rmse:10.37995     valid-rmse:9.33719      train-r2:0.34948
valid-r2:0.38025
[160]   train-rmse:9.92892      valid-rmse:8.97131      train-r2:0.40478
valid-r2:0.42787
[170]   train-rmse:9.60261      valid-rmse:8.72923      train-r2:0.44326
valid-r2:0.45833
[180]   train-rmse:9.35843      valid-rmse:8.57137      train-r2:0.47122
valid-r2:0.47774
[190]   train-rmse:9.17119      valid-rmse:8.46826      train-r2:0.49217
valid-r2:0.49023
[200]   train-rmse:9.02690      valid-rmse:8.40121      train-r2:0.50802
valid-r2:0.49828
[210]   train-rmse:8.92291      valid-rmse:8.36370      train-r2:0.51929
valid-r2:0.50274
[220]   train-rmse:8.83920      valid-rmse:8.33808      train-r2:0.52827
valid-r2:0.50579
[230]   train-rmse:8.77610      valid-rmse:8.32875      train-r2:0.53498
valid-r2:0.50689
[240]   train-rmse:8.72743      valid-rmse:8.32010      train-r2:0.54012
valid-r2:0.50792
[250]   train-rmse:8.68406      valid-rmse:8.31107      train-r2:0.54468
```

```
valid-r2:0.50898
[260]   train-rmse:8.64603      valid-rmse:8.30592      train-r2:0.54866
valid-r2:0.50959
[270]   train-rmse:8.60566      valid-rmse:8.30602      train-r2:0.55286
valid-r2:0.50958
[280]   train-rmse:8.57266      valid-rmse:8.30284      train-r2:0.55629
valid-r2:0.50996
[290]   train-rmse:8.54801      valid-rmse:8.30553      train-r2:0.55883
valid-r2:0.50964
[300]   train-rmse:8.52263      valid-rmse:8.30467      train-r2:0.56145
valid-r2:0.50974
[310]   train-rmse:8.49918      valid-rmse:8.30303      train-r2:0.56386
valid-r2:0.50993
[320]   train-rmse:8.47071      valid-rmse:8.30450      train-r2:0.56678
valid-r2:0.50976
[330]   train-rmse:8.44156      valid-rmse:8.29897      train-r2:0.56975
valid-r2:0.51041
[340]   train-rmse:8.41497      valid-rmse:8.29856      train-r2:0.57246
valid-r2:0.51046
[350]   train-rmse:8.39309      valid-rmse:8.29449      train-r2:0.57468
valid-r2:0.51094
[360]   train-rmse:8.36732      valid-rmse:8.29281      train-r2:0.57729
valid-r2:0.51114
[370]   train-rmse:8.34113      valid-rmse:8.29552      train-r2:0.57993
valid-r2:0.51082
[380]   train-rmse:8.31828      valid-rmse:8.29337      train-r2:0.58223
valid-r2:0.51107
[390]   train-rmse:8.29450      valid-rmse:8.29060      train-r2:0.58461
valid-r2:0.51140
[400]   train-rmse:8.27138      valid-rmse:8.29096      train-r2:0.58693
valid-r2:0.51136
[410]   train-rmse:8.23775      valid-rmse:8.29364      train-r2:0.59028
valid-r2:0.51104
[420]   train-rmse:8.21518      valid-rmse:8.29204      train-r2:0.59252
valid-r2:0.51123
[430]   train-rmse:8.19153      valid-rmse:8.29294      train-r2:0.59486
valid-r2:0.51112
[440]   train-rmse:8.17095      valid-rmse:8.29089      train-r2:0.59690
valid-r2:0.51136
Stopping. Best iteration:
[393]   train-rmse:8.28942      valid-rmse:8.28982      train-r2:0.58512
valid-r2:0.51149
```

[24]:
```
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()
```

[24]:

	ID	y
0	1	82.533493
1	2	97.487167
2	3	83.476692
3	4	77.462120
4	5	112.954796

[ ]: