model.rf20k

June 13, 2021

1 Project 2

Author: Jakub Bednarz

To be quite honest, there's not much to tell; I pick a sample of 20k records from the train dataset, train a pipeline composed of:

- imputer for numerical features;
- one-hot encoder for categorical features;
- a `SelectKBest(k = 32)`;
- an untuned `RandomForestRegressor`.

and perform 5-fold CV to check if it overfits or not.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib widget
```

[2]:
```
import random

seed = 666
random.seed(seed)
np.random.seed(seed)
```

[3]:
```
X_path = 'data/SUS_project_training_data.csv'
y_path = 'data/training_targets.txt'
test_path = 'data/SUS_project_test_data.csv'

file = open(y_path, 'r', encoding='latin-1')
nrecords = sum(1 for line in file) - 1
```

[4]:
```
def any_sample(n):
    index = np.arange(nrecords)
    selected = np.sort(np.random.choice(index, n))
    skipped = np.setdiff1d(index, selected)

    _X = pd.read_csv(X_path, sep=';', encoding='latin-1',
```

```
                    skiprows=1+skipped)

    _y = pd.read_csv(y_path, sep=';', names=['y'],
                    encoding='latin-1', header=None,
                    skiprows=skipped)

    return (_X, _y.y.ravel())
```

[30]:
```
X, y = any_sample(20000)
```

[31]:
```
num_feat = X.dtypes[X.dtypes != object].index
cat_feat = X.dtypes[X.dtypes == object].index
```

[32]:
```
from sklearn.preprocessing import FunctionTransformer, OneHotEncoder
from sklearn.feature_selection import VarianceThreshold
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

class NamefulSimpleImputer(SimpleImputer):
    def __init__(self):
        super(NamefulSimpleImputer, self).__init__(
            add_indicator = True)

    def fit(self, X, y = None):
        super().fit(X, y)

        if isinstance(X, (pd.DataFrame, pd.Series)):
            self.features = list(X.columns)
        else:
            self.features = [str(x) for x in range(X.shape[1])]

        self.features = [
            *self.features,
            *(self.features[idx] + '_isna'
              for idx in self.indicator_.features_)
        ]

        return self

    def transform(self, X):
        tX = super().transform(X)

        if isinstance(X, (pd.DataFrame, pd.Series)):
            return pd.DataFrame(
                data = tX,
                columns = self.get_feature_names())
```

```
        else:
            return tX

    def get_feature_names(self):
        return self.features

prep = Pipeline([
    ('ct', ColumnTransformer([
        ('num', NamefulSimpleImputer(), num_feat),
        ('cat', OneHotEncoder(handle_unknown='ignore'), cat_feat),
    ])),
    ('nonzero_var', VarianceThreshold()),
])
```

[33]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, VarianceThreshold
from sklearn.decomposition import TruncatedSVD
from sklearn.linear_model import ElasticNetCV
from sklearn.svm import LinearSVR, SVR
from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor

# model = SVR()
model = RandomForestRegressor(n_jobs=-1)
# model = ExtraTreesRegressor(n_jobs=-1)

pipe = Pipeline([
    ('prep', prep),
    ('fsel', SelectKBest(k=32)),
#     ('scale', StandardScaler(with_mean=False)),
    ('model', model)
])
```

[34]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import r2_score
cross_val_score(pipe, X, y, scoring='r2')
```

[34]:
```
array([0.48244448, 0.51116721, 0.47507883, 0.5367601 , 0.49635531])
```

[35]:
```
pipe.fit(X, y)
test_X = pd.read_csv(test_path, sep=';', encoding='latin-1')
test_pred = pipe.predict(test_X)
test_pred[test_pred < 0] = 0
test_pred_df = pd.DataFrame(data = test_pred)
test_pred_df.to_csv('res.txt', index=False, header=None)
```

[36]:
```
X_val, y_val = any_sample(500000)
y_pred = pipe.predict(X_val)
```

```
r2_score(y_val, y_pred)
```

[36]: 0.5138471331058332