PS18

January 26, 2024

1 Car test-time prediction

1.1 Loading MB dataset

[1]:

```
import pandas as pd
data = pd.read_csv('mercedes_test.csv')
```

1.2 Data pre-processing

[2]:

```
# Choose categorical data columns
cf = data.select_dtypes(include=['object']).columns
# To change it into "categorical" data type
data[cf]=data[cf].astype('category')
# One hot encoding
data = pd.get_dummies(data)
# Obtain X from data (excluding 'ID' and 'y')
X_df = data.drop(['ID','y'],axis=1)
# Obtain y from data
y_df = data['y']

# Convert y_df into binary labels
import numpy as np
TF_vector= (y_df<np.median(y_df))
y_df=TF_vector.astype(float)

# Conver data frame into numpy array
X,y = X_df.values, y_df.values

# Split into train and test datasets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,stratify=y)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3788, 563)
(421, 563)
```

```
(3788,)
(421,)
```

1.3 DNN: Hyparameter search via cross validation

[3]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
```

[4]:

```
pip install scikeras
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikeras in
c:\users\chsuh\appdata\roaming\python\python39\site-packages (0.12.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem<0.32,>=0.23.1 in
c:\users\chsuh\appdata\roaming\python\python39\site-packages (from scikeras)
(0.31.0)
Requirement already satisfied: scikit-learn>=1.0.0 in
c:\programdata\anaconda3\lib\site-packages (from scikeras) (1.0.2)
Requirement already satisfied: packaging>=0.21 in
c:\programdata\anaconda3\lib\site-packages (from scikeras) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
c:\programdata\anaconda3\lib\site-packages (from packaging>=0.21->scikeras)
(3.0.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=1.0.0->scikeras)
(2.2.0)
Requirement already satisfied: joblib>=0.11 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=1.0.0->scikeras)
(1.1.0)
Requirement already satisfied: scipy>=1.1.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=1.0.0->scikeras)
(1.7.3)
Requirement already satisfied: numpy>=1.14.6 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=1.0.0->scikeras)
(1.21.5)
Note: you may need to restart the kernel to use updated packages.
```

[29]:

```
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
#from scikeras.wrappers import KerasClassifier
```

[30]:

```
from sklearn.model_selection import RandomizedSearchCV
```

[31]:

```
def build_model(n_layer=2,lambda_=0,lr=1e-3):
    model = Sequential()
    for i in range(n_layer-1):
        model.add(Dense(20,activation='relu',
                kernel_regularizer=l2(lambda_),bias_regularizer=l2(lambda_)))

    model.add(Dense(1, activation='sigmoid',
                kernel_regularizer=l2(lambda_),bias_regularizer=l2(lambda_)))
    optimizer = Adam(learning_rate=lr)
    model.compile(optimizer=optimizer,
                loss='binary_crossentropy',
                metrics=['acc'])
    return model
```

[33]:

```
# return a scikit-learn-like Keras model
model = KerasClassifier(build_model)
n_layer = [2,5,10]
lambda_ = [1e-3,1e-2,1e-1,1,10]
grid = {'n_layer':n_layer,'lambda_':lambda_}
#grid = dict(n_layer=n_layer,lambda_=lambda_)
cv = RandomizedSearchCV(model,grid,n_iter=15,cv=5)
```

```
C:\Users\chsuh\AppData\Local\Temp\ipykernel_40480\933115641.py:2:
DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras
(https://github.com/adriangb/scikeras) instead. See
https://www.adriangb.com/scikeras/stable/migration.html for help migrating.
  model = KerasClassifier(build_model)
```

[34]:

```
cv.fit(X_train,y_train,epochs=10,verbose=0)
```

```
24/24 [==============================] - 0s 2ms/step - loss: 0.3389 - acc:
0.8892
24/24 [==============================] - 0s 2ms/step - loss: 0.3370 - acc:
0.8839
24/24 [==============================] - 0s 2ms/step - loss: 0.3245 - acc:
0.8945
24/24 [==============================] - 0s 2ms/step - loss: 0.3613 - acc:
0.8785
24/24 [==============================] - 0s 2ms/step - loss: 0.3397 - acc:
0.8798
24/24 [==============================] - 0s 2ms/step - loss: 0.3501 - acc:
0.8852
24/24 [==============================] - 0s 3ms/step - loss: 0.3732 - acc:
0.8734
24/24 [==============================] - 0s 3ms/step - loss: 0.3379 - acc:
0.8931
24/24 [==============================] - 0s 3ms/step - loss: 0.3861 - acc:
0.8679
24/24 [==============================] - 0s 4ms/step - loss: 0.3816 - acc:
0.8600
24/24 [==============================] - 0s 4ms/step - loss: 0.3663 - acc:
0.8839
```

```
24/24 [==============================] - 0s 3ms/step - loss: 0.3653 - acc:
0.8826
24/24 [==============================] - 0s 4ms/step - loss: 0.3870 - acc:
0.8549
24/24 [==============================] - 0s 3ms/step - loss: 0.4283 - acc:
0.8520
24/24 [==============================] - 0s 3ms/step - loss: 0.3688 - acc:
0.8705
24/24 [==============================] - 0s 2ms/step - loss: 0.4009 - acc:
0.8892
24/24 [==============================] - 0s 2ms/step - loss: 0.3849 - acc:
0.8839
24/24 [==============================] - 0s 2ms/step - loss: 0.3724 - acc:
0.8958
24/24 [==============================] - 0s 2ms/step - loss: 0.3989 - acc:
0.8732
24/24 [==============================] - 0s 2ms/step - loss: 0.3845 - acc:
0.8824
24/24 [==============================] - 0s 2ms/step - loss: 0.4406 - acc:
0.8879
24/24 [==============================] - 0s 2ms/step - loss: 0.4421 - acc:
0.8839
24/24 [==============================] - 0s 4ms/step - loss: 0.4204 - acc:
0.8931
24/24 [==============================] - 0s 2ms/step - loss: 0.4531 - acc:
0.8758
24/24 [==============================] - 0s 3ms/step - loss: 0.4443 - acc:
0.8811
24/24 [==============================] - 0s 3ms/step - loss: 0.5438 - acc:
0.8813
24/24 [==============================] - 0s 3ms/step - loss: 0.5614 - acc:
0.8707
24/24 [==============================] - 0s 2ms/step - loss: 0.5168 - acc:
0.8839
24/24 [==============================] - 0s 4ms/step - loss: 0.5588 - acc:
0.8732
24/24 [==============================] - 0s 3ms/step - loss: 0.4985 - acc:
0.8851
24/24 [==============================] - 0s 2ms/step - loss: 0.6304 - acc:
0.8298
24/24 [==============================] - 0s 2ms/step - loss: 0.6185 - acc:
0.8747
24/24 [==============================] - 0s 2ms/step - loss: 0.6115 - acc:
0.8945
24/24 [==============================] - 0s 2ms/step - loss: 0.6236 - acc:
0.8639
24/24 [==============================] - 0s 2ms/step - loss: 0.6186 - acc:
0.8771
```

```
24/24 [==============================] - 0s 3ms/step - loss: 0.6933 - acc:
0.4947
24/24 [==============================] - 0s 3ms/step - loss: 0.6933 - acc:
0.4921
24/24 [==============================] - 0s 3ms/step - loss: 0.6935 - acc:
0.4842
24/24 [==============================] - 0s 2ms/step - loss: 0.6933 - acc:
0.4875
24/24 [==============================] - 0s 2ms/step - loss: 0.6935 - acc:
0.4835
24/24 [==============================] - 0s 3ms/step - loss: 0.6933 - acc:
0.4947
24/24 [==============================] - 0s 3ms/step - loss: 0.6934 - acc:
0.4921
24/24 [==============================] - 0s 4ms/step - loss: 0.6935 - acc:
0.4842
24/24 [==============================] - 0s 3ms/step - loss: 0.6936 - acc:
0.4875
24/24 [==============================] - 0s 2ms/step - loss: 0.6936 - acc:
0.4835
24/24 [==============================] - 0s 2ms/step - loss: 0.6945 - acc:
0.8061
24/24 [==============================] - 0s 3ms/step - loss: 0.6936 - acc:
0.6860
24/24 [==============================] - 0s 2ms/step - loss: 0.6934 - acc:
0.4842
24/24 [==============================] - 0s 2ms/step - loss: 0.6944 - acc:
0.8032
24/24 [==============================] - 0s 2ms/step - loss: 0.6947 - acc:
0.5443
24/24 [==============================] - 0s 2ms/step - loss: 0.6943 - acc:
0.5053
24/24 [==============================] - 0s 2ms/step - loss: 0.6947 - acc:
0.4921
24/24 [==============================] - 0s 2ms/step - loss: 0.6949 - acc:
0.4842
24/24 [==============================] - 0s 2ms/step - loss: 0.6946 - acc:
0.4875
24/24 [==============================] - 0s 2ms/step - loss: 0.6951 - acc:
0.4835
24/24 [==============================] - 0s 3ms/step - loss: 0.6954 - acc:
0.4947
24/24 [==============================] - 0s 3ms/step - loss: 0.6961 - acc:
0.5079
24/24 [==============================] - 0s 3ms/step - loss: 0.6956 - acc:
0.4842
24/24 [==============================] - 0s 3ms/step - loss: 0.6965 - acc:
0.4875
```

```
24/24 [==============================] - 0s 4ms/step - loss: 0.6963 - acc:
0.4835
24/24 [==============================] - 0s 2ms/step - loss: 0.7020 - acc:
0.4947
24/24 [==============================] - 0s 2ms/step - loss: 0.7081 - acc:
0.4921
24/24 [==============================] - 0s 3ms/step - loss: 0.7123 - acc:
0.4842
24/24 [==============================] - 0s 2ms/step - loss: 0.7020 - acc:
0.4875
24/24 [==============================] - 0s 2ms/step - loss: 0.7133 - acc:
0.4835
24/24 [==============================] - 0s 2ms/step - loss: 0.7031 - acc:
0.5053
24/24 [==============================] - 0s 3ms/step - loss: 0.7068 - acc:
0.4921
24/24 [==============================] - 0s 3ms/step - loss: 0.7113 - acc:
0.5158
24/24 [==============================] - 0s 3ms/step - loss: 0.7052 - acc:
0.4875
24/24 [==============================] - 0s 3ms/step - loss: 0.7180 - acc:
0.5165
24/24 [==============================] - 0s 3ms/step - loss: 0.7057 - acc:
0.5053
24/24 [==============================] - 0s 5ms/step - loss: 0.7164 - acc:
0.4921
24/24 [==============================] - 0s 3ms/step - loss: 0.7152 - acc:
0.4842
24/24 [==============================] - 0s 5ms/step - loss: 0.7179 - acc:
0.5125
24/24 [==============================] - 0s 3ms/step - loss: 0.7116 - acc:
0.4835
```

[34]:

```
RandomizedSearchCV(cv=5,
                   estimator=<keras.wrappers.scikit_learn.KerasClassifier object
at 0x000002330A162D60>,
                   n_iter=15,
                   param_distributions={'lambda_': [0.001, 0.01, 0.1, 1, 10],
                                        'n_layer': [2, 5, 10]})
```

[35]:

```
cv.cv_results_ # logs results
```

[35]:

```
{'mean_fit_time': array([3.06601644, 3.82905293, 6.10004716, 2.88781676,
4.27630439,
        6.03118043, 2.79550567, 3.62947946, 5.44901824, 2.92013431,
        3.9192627 , 5.30468607, 2.76934686, 3.89925404, 5.59904408]),
 'std_fit_time': array([0.23373162, 0.17218339, 0.31065451, 0.27062337,
0.35877938,
        0.28334342, 0.14102986, 0.12940562, 0.21499381, 0.38372727,
        0.19477592, 0.10804774, 0.16849958, 0.28835562, 0.16325863]),
 'mean_score_time': array([0.1961771 , 0.30822968, 0.35282798, 0.19484248,
0.29811249,
        0.38234458, 0.1984549 , 0.26301684, 0.32303238, 0.20584655,
        0.24384947, 0.38469467, 0.20960279, 0.29249825, 0.3631844 ]),
 'std_score_time': array([0.0133205 , 0.07441279, 0.07247743, 0.00705605,
0.05835765,
        0.05312108, 0.00787894, 0.01601813, 0.01323504, 0.01020617,
        0.01294958, 0.09252542, 0.01193084, 0.08635409, 0.04688477]),
 'param_n_layer': masked_array(data=[2, 5, 10, 2, 5, 10, 2, 5, 10, 2, 5, 10, 2,
5, 10],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_lambda_': masked_array(data=[0.001, 0.001, 0.001, 0.01, 0.01, 0.01, 0.1,
0.1, 0.1,
                    1, 1, 1, 10, 10, 10],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'n_layer': 2, 'lambda_': 0.001},
  {'n_layer': 5, 'lambda_': 0.001},
  {'n_layer': 10, 'lambda_': 0.001},
  {'n_layer': 2, 'lambda_': 0.01},
  {'n_layer': 5, 'lambda_': 0.01},
  {'n_layer': 10, 'lambda_': 0.01},
  {'n_layer': 2, 'lambda_': 0.1},
  {'n_layer': 5, 'lambda_': 0.1},
  {'n_layer': 10, 'lambda_': 0.1},
  {'n_layer': 2, 'lambda_': 1},
  {'n_layer': 5, 'lambda_': 1},
  {'n_layer': 10, 'lambda_': 1},
  {'n_layer': 2, 'lambda_': 10},
  {'n_layer': 5, 'lambda_': 10},
  {'n_layer': 10, 'lambda_': 10}],
 'split0_test_score': array([0.88918203, 0.88522428, 0.88390499, 0.88918203,
0.8878628 ,
        0.88126647, 0.82981533, 0.49472296, 0.49472296, 0.8060686 ,
        0.50527704, 0.49472296, 0.49472296, 0.50527704, 0.50527704]),
 'split1_test_score': array([0.88390499, 0.87335092, 0.88258576, 0.88390499,
0.88390499,
        0.8707124 , 0.87467021, 0.49208444, 0.49208444, 0.68601584,
        0.49208444, 0.50791556, 0.49208444, 0.49208444, 0.49208444]),
 'split2_test_score': array([0.89445913, 0.89313984, 0.85488129, 0.89577836,
0.89313984,
        0.88390499, 0.89445913, 0.48416886, 0.48416886, 0.48416886,
        0.48416886, 0.48416886, 0.48416886, 0.51583111, 0.48416886]),
 'split3_test_score': array([0.87846762, 0.8678996 , 0.85204756, 0.87318361,
0.87582564,
        0.87318361, 0.8639366 , 0.48745045, 0.48745045, 0.80317038,
        0.48745045, 0.48745045, 0.48745045, 0.48745045, 0.51254952]),
 'split4_test_score': array([0.87978864, 0.85997361, 0.87054163, 0.88243067,
0.88110965,
        0.88507265, 0.8771466 , 0.48348746, 0.48348746, 0.54425365,
        0.48348746, 0.48348746, 0.48348746, 0.51651257, 0.48348746]),
 'mean_test_score': array([0.88516048, 0.87591765, 0.86879225, 0.88489593,
0.88436859,
        0.87882802, 0.86800557, 0.48838283, 0.48838283, 0.66473547,
        0.49049365, 0.49154906, 0.48838283, 0.50343112, 0.49551346]),
 'std_test_score': array([0.00596431, 0.01190216, 0.01338369, 0.00749714,
0.00588153,
        0.00580413, 0.02146074, 0.00439297, 0.00439297, 0.13169078,
        0.00799287, 0.00910155, 0.00439297, 0.01193667, 0.01158134]),
 'rank_test_score': array([ 1,  5,  6,  2,  3,  4,  7, 13, 13,  8, 12, 11, 13,
9, 10])}
```

1.4 Store logs into csv file

[36]:

```
# Store logs into csv file
import pandas as pd
df_DNN=pd.DataFrame.from_dict(cv.cv_results_,orient='columns')
# Select columns to be stored
columns = ['params','mean_test_score','std_test_score','rank_test_score']
df_DNN = df_DNN[columns]
df_DNN.to_csv("logs_DNN.csv")
```

1.5 Save the best model

[37]:

```
best_model_DNN=cv.best_estimator_
best_model_DNN.model.save('best_model_DNN')
```

```
INFO:tensorflow:Assets written to: best_model_DNN\assets
```

1.6 Load the best model

[38]:

```
from tensorflow.keras.models import load_model
loaded_model = load_model('best_model_DNN')
loaded_model.evaluate(X_test, y_test)
```

```
14/14 [==============================] - 0s 3ms/step - loss: 0.4166 - acc:
0.8480
```

```
[38]: [0.41655808687210083, 0.8479809761047363]
```

```
[ ]:
```