

```

1 import csv
2 import os
3
4 import numpy as np
5 import pandas as pd
6 from keras.callbacks import EarlyStopping
7 from keras.layers import Dense, Dropout
8 from sklearn.metrics import (
9     explained_variance_score,
10     mean_absolute_error,
11     mean_absolute_percentage_error,
12     mean_squared_error,
13 )
14 from sklearn.model_selection import train_test_split
15 from sklearn.preprocessing import MinMaxScaler
16 from tensorflow import keras
17 from tqdm import tqdm, trange
18
19
20 my_path = rf"{os.getcwd()}\CIAN_diplom"
21
22 df_whole = pd.read_csv(rf"{my_path}\df.csv")
23
24
25 districts = sorted(list(set(df_whole["district"])))
26 activation_functions = {
27     "relu": keras.layers.ReLU(),
28     "leaky_relu": keras.layers.LeakyReLU(),
29     "elu": keras.layers.ELU(),
30 }
31 optimizers = {
32     "rmsprop": keras.optimizers.legacy.RMSprop(),
33     "adam": keras.optimizers.legacy.Adam(),
34     "amsgrad": keras.optimizers.legacy.Adam(amsgrad=True),
35     "adamax": keras.optimizers.legacy.Adamax(),
36 }
37 nodes = [
38     (400,),
39     (500,),
40     (600,),
41     (200, 50),
42     (200, 100),
43     (400, 200),
44     (400, 300),
45     (500, 200),
46     (500, 300),
47     (600, 200),
48     (600, 300),
49     (200, 100, 50),
50     (400, 200, 100),
51     (400, 200, 200),
52     (400, 300, 100),
53     (400, 300, 200),
54     (500, 300, 200),
55     (500, 300, 100),
56     (600, 300, 200),
57     (600, 300, 100),
58     (200, 200, 100, 50),
59     (400, 200, 100, 50),
60     (400, 300, 200, 50),
61     (400, 300, 200, 100),
62     (400, 300, 300, 50),
63     (500, 300, 200, 50),
64     (500, 300, 200, 100),
65     (500, 400, 200, 50),
66     (500, 400, 200, 100),

```

```

67     (600, 300, 200, 50),
68     (600, 300, 200, 100),
69     (700, 500, 300, 200),
70     (700, 500, 200, 100),
71     (700, 600, 300, 200),
72     (800, 600, 300, 200),
73     (900, 600, 300, 200),
74 ]
75
76 with open(
77     rf"{my_path}\grid_search_districts.csv", "a", encoding="utf-8", newline=""
78 ) as ouf:
79     writer = csv.writer(ouf, delimiter="$", quotechar="|", quoting=csv.QUOTE_MINIMAL)
80     writer.writerow(
81         (
82             "district",
83             "nodes",
84             "function",
85             "optimizer",
86             "cv_number",
87             "MSE",
88             "RMSE",
89             "MAE",
90             "MAPE",
91             "var_score",
92         )
93     )
94
95
96 for distr in tqdm(districts, desc="District"):
97     df = df_whole[df_whole["district"] == distr]
98     df = pd.get_dummies(df, drop_first=True)
99
100     X = df.drop("full_price", axis=1)
101     y = df["full_price"]
102     X_train, X_test, y_train, y_test = train_test_split(
103         X, y, test_size=0.3, random_state=42
104     )
105
106     scaler = MinMaxScaler()
107     X_train = scaler.fit_transform(X_train)
108     X_test = scaler.transform(X_test)
109
110     for node in tqdm(nodes, desc="Nodes", leave=False):
111         for f_name, func in tqdm(
112             activation_functions.items(), desc="Functions", leave=False
113         ):
114             for opt_name, opt in tqdm(
115                 optimizers.items(), desc="Optimizers", leave=False
116             ):
117                 for cv_counter in trange(3, desc="CV", leave=False):
118                     if len(node) == 1:
119                         model = keras.models.Sequential(
120                             [
121                                 Dense(
122                                     node[0],
123                                     activation=func,
124                                     kernel_initializer="he_uniform",
125                                     input_dim=X_train.shape[1],
126                                 ),
127                                 Dense(1, activation="linear"),
128                             ]
129                         )
130
131                     elif len(node) == 2:
132                         model = keras.models.Sequential(
133                             [

```

```

134         Dense(
135             node[0],
136             activation=func,
137             kernel_initializer="he_uniform",
138             input_dim=X_train.shape[1],
139         ),
140         Dense(
141             node[1],
142             activation=func,
143             kernel_initializer="he_uniform",
144         ),
145         Dense(1, activation="linear"),
146     ]
147 )
148
149 elif len(node) == 3:
150     model = keras.models.Sequential(
151         [
152             Dense(
153                 node[0],
154                 activation=func,
155                 kernel_initializer="he_uniform",
156                 input_dim=X_train.shape[1],
157             ),
158             Dense(
159                 node[1],
160                 activation=func,
161                 kernel_initializer="he_uniform",
162             ),
163             Dropout(0.3, seed=123),
164             Dense(
165                 node[2],
166                 activation=func,
167                 kernel_initializer="he_uniform",
168             ),
169             Dense(1, activation="linear"),
170         ]
171     )
172
173 elif len(node) == 4:
174     model = keras.models.Sequential(
175         [
176             Dense(
177                 node[0],
178                 activation=func,
179                 kernel_initializer="he_uniform",
180                 input_dim=X_train.shape[1],
181             ),
182             Dense(
183                 node[1],
184                 activation=func,
185                 kernel_initializer="he_uniform",
186             ),
187             Dropout(0.3, seed=123),
188             Dense(
189                 node[2],
190                 activation=func,
191                 kernel_initializer="he_uniform",
192             ),
193             Dropout(0.3, seed=123),
194             Dense(
195                 node[3],
196                 activation=func,
197                 kernel_initializer="he_uniform",
198             ),
199             Dense(1, activation="linear"),
200         ]

```

```
201         )
202
203     model.compile(
204         loss="mse",
205         optimizer=opt,
206     )
207
208     early_stop = EarlyStopping(
209         monitor="val_loss", mode="min", verbose=0, patience=10
210     )
211
212     model.fit(
213         X_train,
214         y_train.values,
215         epochs=500,
216         batch_size=128,
217         validation_data=(X_test, y_test.values),
218         callbacks=[early_stop],
219         verbose=0,
220     )
221
222     predictions = model.predict(X_test, verbose=0)
223
224     del model
225
226     mae = mean_absolute_error(y_test, predictions)
227     mse = mean_squared_error(y_test, predictions)
228     rmse = np.sqrt(mean_squared_error(y_test, predictions))
229     mape = mean_absolute_percentage_error(y_test, predictions)
230     var_score = explained_variance_score(y_test, predictions)
231
232     data = (
233         distr,
234         node,
235         f_name,
236         opt_name,
237         cv_counter,
238         mse,
239         rmse,
240         mae,
241         mape,
242         var_score,
243     )
244
245     with open(
246         rf"{my_path}\grid_search_districts.csv",
247         "a",
248         encoding="utf-8",
249         newline="",
250     ) as ouf:
251         writer = csv.writer(
252             ouf,
253             delimiter="$",
254             quotechar="|",
255             quoting=csv.QUOTE_MINIMAL,
256         )
257         writer.writerow(data)
```