

```
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()}"
21
22 df_whole = pd.read_csv(rf"{my_path}\df.csv")
23
24
25 activation_functions = {
26     "relu": keras.layers.ReLU(),
27     "leaky_relu": keras.layers.LeakyReLU(),
28     "elu": keras.layers.ELU(),
29 }
30 optimizers = {
31     "rmsprop": keras.optimizers.legacy.RMSprop(),
32     "adam": keras.optimizers.legacy.Adam(),
33     "amsgrad": keras.optimizers.legacy.Adam(amsgrad=True),
34     "adamax": keras.optimizers.legacy.Adamax(),
35 }
36 nodes = [
37     (400,),
38     (500,),
39     (600,),
40     (200, 50),
41     (200, 100),
42     (400, 200),
43     (400, 300),
44     (500, 200),
45     (500, 300),
46     (600, 200),
47     (600, 300),
48     (200, 100, 50),
49     (400, 200, 100),
50     (400, 200, 200),
51     (400, 300, 100),
52     (400, 300, 200),
53     (500, 300, 100),
54     (500, 300, 200),
55     (600, 300, 100),
56     (600, 300, 200),
57     (200, 200, 100, 50),
58     (400, 200, 100, 50),
59     (400, 300, 200, 50),
60     (400, 300, 200, 100),
61     (400, 300, 300, 50),
62     (500, 300, 200, 50),
63     (500, 300, 200, 100),
64     (500, 400, 200, 50),
65     (500, 400, 200, 100),
66     (600, 300, 200, 50),
```

```

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

```

```

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

```

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