

## data-processing.py

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.preprocessing import LabelEncoder
4 from sklearn.model_selection import train_test_split
5 from shutil import rmtree
6 from os.path import exists, isdir
7 from os import makedirs
8
9 DATA_NAME = "original_data.csv"
10 OUT_DATA_PREFIX = "/data"
11 PNN_OUTPUT_DIR = "./out-pnn"
12 PNN_REFERENCE_OUTPUT_DIR = PNN_OUTPUT_DIR + OUT_DATA_PREFIX
13 TRAIN_OUTPUT_DIR = "./out-train"
14 TRAIN_REFERENCE_OUTPUT_DIR = TRAIN_OUTPUT_DIR + OUT_DATA_PREFIX
15 TEST_OUTPUT_DIR = "./out-test"
16 TEST_REFERENCE_OUTPUT_DIR = TEST_OUTPUT_DIR + OUT_DATA_PREFIX
17
18 PNN_REFERENCE_COUNT_BY_CLASS = {
19     "smurf": 0.6,
20     "neptune": 0.6,
21     "normal": 0.6,
22     "back": 1.0,
23     "satan": 1.0,
24     "ipsweep": 1.0,
25     "portsweep": 1.0,
26     "warezclient": 1.0,
27     "teardrop": 1.0,
28     "pod": 1.0,
29     "nmap": 1.0,
30     "guess_passwd": 1.0,
31     "buffer_overflow": 1.0,
32     "land": 1.0,
33     "warezmaster": 1.0,
34     "imap": 1.0,
35     "rootkit": 1.0,
36     "loadmodule": 1.0,
37     "ftp_write": 1.0,
38     "multihop": 1.0,
39     "phf": 1.0,
40     "perl": 1.0,
41     "spy": 1.0
42 }
43 TEST_SIZE = 0.25
44 REAL_DATA_COUNT_PER_CLASS = 8000
45 RANDOM_STATE = 1450
46
47 # Read data
48 data = pd.read_csv(DATA_NAME)
49
50 # Show column info
51 print(">>> Original data >>>")
52 for column in data.columns:
53     unique_values_count = data[column].nunique()
```

```
54     unique_values = data[column].unique()
55     print(f"Column: {column}")
56     print(f"Unique Values Count: {unique_values_count}")
57     print(f"Unique Values: {unique_values}")
58     print()
59 print(">>> Column types >>>\n", data.dtypes, "\n")
60
61 # Show unique classes and their count
62 print(">>> Class | Count >>>")
63 print(data["label"].value_counts(), "\n")
64
65 # Drop lnum_outbound_cmds and is_host_login columns since values are same for all rows
66 data = data.drop(columns = ["lnum_outbound_cmds", "is_host_login"])
67
68 # Convert protocol_type, service, flag columns to numeric
69 nonnumeric_columns = ["protocol_type", "service", "flag"]
70 label_encoder = LabelEncoder()
71 for column in nonnumeric_columns:
72     data[column] = label_encoder.fit_transform(data[column])
73
74 # normalize data
75 for column_name in data.columns:
76     if column_name != 'label':
77         column = data[column_name]
78         min_val = column.min()
79         max_val = column.max()
80         data[column_name] = (column - min_val) / (max_val - min_val)
81
82 # Show column info
83 print(">>> Data: removed useless columns, all numeric >>>")
84 for column in data.columns:
85     unique_values_count = data[column].nunique()
86     unique_values = data[column].unique()
87     print(f"Column: {column}")
88     print(f"Unique Values Count: {unique_values_count}")
89     print(f"Unique Values: {unique_values}")
90     print()
91 print(">>> Column types >>>\n", data.dtypes, "\n")
92
93 # Show unique classes and their count
94 print(">>> Class | Count >>>")
95 print(data["label"].value_counts(), "\n")
96
97 # Delete old run data and recreate out directory tree
98 if exists(PNN_OUTPUT_DIR):
99     rmtree(PNN_OUTPUT_DIR)
100 makedirs(PNN_REFERENCE_OUTPUT_DIR)
101 if exists(TRAIN_OUTPUT_DIR):
102     rmtree(TRAIN_OUTPUT_DIR)
103 makedirs(TRAIN_REFERENCE_OUTPUT_DIR)
104 if exists(TEST_OUTPUT_DIR):
105     rmtree(TEST_OUTPUT_DIR)
106 makedirs(TEST_REFERENCE_OUTPUT_DIR)
107
108 # data export
109 property_count = len(data.columns.tolist()) - 1
```

```

110 data.insert(0, "property_count", property_count)
111 rows_by_class = dict(tuple(data.groupby("label")))
112 count_by_class = { "pnn": 0, "train": 0, "test": 0 }
113 for k,k_data in rows_by_class.items():
114     k_data.drop(columns = ["label"], inplace = True)
115     if k_data.shape[0] > REAL_DATA_COUNT_PER_CLASS:
116         # select real count from k_data
117         k_data_real = k_data.sample(n = REAL_DATA_COUNT_PER_CLASS, random_state =
RANDOM_STATE)
118
119         # split
120         k_data_train, k_data_test = train_test_split(k_data_real, test_size = TEST_SIZE,
random_state = RANDOM_STATE)
121
122         # take pnn data from train data
123         k_data_pnn = k_data_train.sample(frac = PNN_REFERENCE_COUNT_BY_CLASS[k], random_state
= RANDOM_STATE)
124
125         # save data
126         count_by_class["pnn"] += k_data_pnn.shape[0]
127         k_data_pnn.to_csv(PNN_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data_pnn.shape[0])
+ ".csv",
128                             header = False, index = True, lineterminator = ',')
129         count_by_class["train"] += k_data_train.shape[0]
130         k_data_train.to_csv(TRAIN_REFERENCE_OUTPUT_DIR + "/" + k + "-" +
str(k_data_train.shape[0]) + ".csv",
131                             header = False, index = True, lineterminator = ',')
132         count_by_class["test"] += k_data_test.shape[0]
133         k_data_test.to_csv(TEST_REFERENCE_OUTPUT_DIR + "/" + k + "-" +
str(k_data_test.shape[0]) + ".csv",
134                             header = False, index = True, lineterminator = ',')
135     else:
136         # save whole data as pnn, train, test
137         count_by_class["pnn"] += k_data.shape[0]
138         k_data.to_csv(PNN_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data.shape[0]) + "
.csv",
139                             header = False, index = True, lineterminator = ',')
140         count_by_class["train"] += k_data.shape[0]
141         k_data.to_csv(TRAIN_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data.shape[0]) + "
.csv",
142                             header = False, index = True, lineterminator = ',')
143         count_by_class["test"] += k_data.shape[0]
144         k_data.to_csv(TEST_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data.shape[0]) + "
.csv",
145                             header = False, index = True, lineterminator = ',')
146
147 # save metadata
148 with open(PNN_OUTPUT_DIR + "/info.csv", "w", newline = "\n") as info:
149     info.write(",".join([str(property_count),
150                             str(count_by_class["pnn"]),
151                             str(len(data["label"].unique()))]))
152 with open(TRAIN_OUTPUT_DIR + "/info.csv", "w", newline = "\n") as info:
153     info.write(",".join([str(property_count),
154                             str(count_by_class["train"]),
155                             str(len(data["label"].unique()))]))
156 with open(TEST_OUTPUT_DIR + "/info.csv", "w", newline = "\n") as info:
157     info.write(",".join([str(property_count),
158                             str(count_by_class["test"]),

```

159 |

```
str(len(data["label"].unique()))))
```