
H2O.ai

Missing Data Handling

```
nacnts = dict(zip(census_data.col_names, census_data.nacnt()))
print(dict((k, int(v)) for (k, v) in nacnts.items() if v > 0))
```

```
{'workclass': 1836, 'occupation': 1843, 'native-country': 583}
```

```
codes = census_data["native-country"].asnumeric()
levels = census_data["native-country"].levels()[0]
levels.append("Unknown")

census_data["native-country-clean"] = h2o.H2OFrame.ifelse(codes != None, codes, len(levels))
census_data["native-country-clean"] = census_data["native-country-clean"].asfactor()
census_data["native-country-clean"] = census_data["native-country-clean"].set_levels(levels)

print((census_data["native-country-clean"] == "Unknown").table())
```

native-country-clean	Count
0	31978
1	583

Summary & Aggregation

```
h2o_frame[x].table(dense = TRUE)
```

Arguments

h2o_frame	An H2OFrame object with at least one column
x	Column name
dense	A logical for dense representation, which lists only non-zero counts, 1 combination per row. Set to FALSE to expand counts across all combinations.

Value: Returns a tabulated H2OFrame Object

Missing Data Handling

```
1 nacnts = dict(zip(census_data.col_names, census_data.nacnt()))
2 print(dict((k, int(v)) for (k, v) in nacnts.items() if v > 0))
```

```
{'workclass': 1836, 'occupation': 1843, 'native-country': 583}
```

```
1 codes = census_data["native-country"].asnumeric()
2 levels = census_data["native-country"].levels()[0]
3 levels.append("Unknown")
4
5 census_data["native-country-clean"] = h2o.H2OFrame.ifelse(codes != None, codes, len(levels))
6 census_data["native-country-clean"] = census_data["native-country-clean"].asfactor()
7 census_data["native-country-clean"] = census_data["native-country-clean"].set_levels(levels)
8
9 print((census_data["native-country-clean"] == "Unknown").table())
```

native-country-clean	Count
0	31978
1	583