template

December 10, 2024

[1]:
```
%load_ext autoreload
%autoreload 2
```

[ ]:
```

```

1 Interesting Title

Name(s): Quy-Dzu Do

Website Link: https://chess.com

[2]:
```
import pandas as pd
import numpy as np
import os

import plotly.express as px
pd.options.plotting.backend = 'plotly'

from dsc80_utils import * # Feel free to uncomment and use this.

LOCAL_DIR = os.getcwd()
```

[3]:
```
from final_proj import *
```

1.1 Step 1: Introduction

I chose to do the Outage Dataset as I felt the data to be more relevant to me as I hav lived in California for most of my life have gone through many outages due to High Winds, fires and other reasons. The League data does not interest me since I have no experience with the game so I have no expertise in the field and any conclusions I couls draw would most likely lack context needed to properly extrapolate the data. As for the recepies, I would not be opposed to working with it but I find the subject of outages to be more compelling than data on cooking and nutritional facts.

For the outages dataset, I am most interested in looking at the relation between frequency and duration of outages with the corresponding population affected. I expect there to be a difference as most companies would be more concerned with population centers and centers of commerce getting an outage than a rural population but I would like to see how disproporionate is the result.

1.2 Step 2: Data Cleaning and Exploratory Data Analysis

We shall begin by pulling the data into a Pandas Dataframe skipping the first 5 rows and the seventh row as they do not contain data and we set the "OBV" Column to the index as it IDs the rows of data. We will also convert the "OUTAGE.START.TIME" and "OUTAGE.RESTORATION.TIME" to timedelta objects, and "OUTAGE.START.DATE" and "OUTAGE.RESTORATION.DATE" to timestamp objects so they can be combined into a single date-time value stores as a Pandas Timestamp in a new dataframe with "OUTAGE.START.DATE" and "OUTAGE.RESTORATION.DATE" containing the new objects and the Time columns dropped.

[4]:
```
data_set = os.path.join(LOCAL_DIR, "outage.xlsx")
data = pd.read_excel(data_set, skiprows=lambda x: x in [0,1,2,3,4,6],
                     dtype={"OUTAGE.START.TIME": str,
                            "OUTAGE.RESTORATION.TIME": str})
data = data.drop("variables", axis=1).set_index("OBS")
data_time= time_to_datetime(data)
data_time
```

[4]:
```
       YEAR  MONTH    U.S._STATE POSTAL.CODE  … AREAPCT_UC PCT_LAND  \
OBS                                           …
1      2011    7.0     Minnesota          MN  …       0.60    91.59
2      2014    5.0     Minnesota          MN  …       0.60    91.59
3      2010   10.0     Minnesota          MN  …       0.60    91.59
…     …    …           …          …  …        …       …
1532   2009    8.0  South Dakota          SD  …       0.15    98.31
1533   2009    8.0  South Dakota          SD  …       0.15    98.31
1534   2000    NaN        Alaska          AK  …       0.02    85.76

       PCT_WATER_TOT PCT_WATER_INLAND
OBS
1               8.41             5.48
2               8.41             5.48
3               8.41             5.48
…               …                …
1532            1.69             1.69
1533            1.69             1.69
1534           14.24             2.90

[1534 rows x 55 columns]
```

[5]:
```
data_time["OUTAGE.DURATION"]\
    .hist(bins=100, title="Outage Duration Distribution")
```

[6]:
```
data_time.loc[data_time["U.S._STATE"] == "California", "OUTAGE.DURATION"]\
    .hist(bins=100)
```

[7]:
```
data_time.groupby("YEAR")["OUTAGE.DURATION"].agg("mean")\
    .plot.line(title="Average Outage Duration by Year")
```

[8]:
```
# Create the scatter plot
fig = px.scatter(data_time, x='OUTAGE.DURATION',
                 y='CUSTOMERS.AFFECTED',
                 title="Customers Affected vs. Outage Duration",
                 labels={'OUTAGE.DURATION': 'Outage Duration (minutes)',
                         'CUSTOMERS.AFFECTED': 'Customers Affected'})

# Show the plot
fig.show()
```

[9]:
```
# Create the scatter plot
fig = px.scatter(data_time, x='TOTAL.CUSTOMERS',
                 y='CUSTOMERS.AFFECTED',
                 title="Customers Affected vs. Total Customers",
                 labels={'TOTAL.CUSTOMERS': 'Number of Customers in the State',
                         'CUSTOMERS.AFFECTED': 'Customers Affected'})

# Show the plot
fig.show()
```

[10]:
```
table = pd.pivot_table(data_time, values=['OUTAGE.DURATION',
                                          "CUSTOMERS.AFFECTED",
                                          "DEMAND.LOSS.MW"],
                       index=["U.S._STATE"],
                      aggfunc="mean")
table
```

[10]:

U.S._STATE	CUSTOMERS.AFFECTED	DEMAND.LOSS.MW	OUTAGE.DURATION
Alabama	94328.80	291.50	1152.80
Alaska	14273.00	35.00	NaN
Arizona	64402.67	1245.70	4552.92
…	…	…	…
West Virginia	179794.33	362.00	6979.00
Wisconsin	45876.00	161.00	7904.11
Wyoming	11833.33	26.75	33.33

[50 rows x 3 columns]

1.3 Step 3: Assessment of Missingness

[11]:
```
missing_data = data_time.copy()
missing_data["MISSING_LABEL"] = (missing_data["OUTAGE.DURATION"].isna()).
 ↪astype(str)
missing_data
```

[11]:
```
       YEAR  MONTH    U.S._STATE POSTAL.CODE  … PCT_LAND PCT_WATER_TOT  \
OBS                                           …
1      2011    7.0     Minnesota          MN  …    91.59          8.41
2      2014    5.0     Minnesota          MN  …    91.59          8.41
3      2010   10.0     Minnesota          MN  …    91.59          8.41
…       …      …           …          …   …      …           …
1532   2009    8.0  South Dakota          SD  …    98.31          1.69
1533   2009    8.0  South Dakota          SD  …    98.31          1.69
1534   2000    NaN        Alaska          AK  …    85.76         14.24

      PCT_WATER_INLAND MISSING_LABEL
OBS
1                 5.48         False
2                 5.48         False
3                 5.48         False
…                 …            …
1532              1.69         False
1533              1.69         False
1534              2.90          True

[1534 rows x 56 columns]
```

[12]:
```
stats, obs = permutation_test(missing_data, 'MONTH', 'MISSING_LABEL', tvd)
np.mean(stats >= obs)
```

[12]:
```
np.float64(0.127)
```

[13]:
```
fig = px.histogram(stats)
fig.add_vline(x=obs, line_width=3, line_dash="dash", line_color="red")
fig.show()
```

[14]:
```
stats, obs = permutation_test(missing_data, 'NERC.REGION', 'MISSING_LABEL', tvd)
np.mean(stats >= obs)
```

[14]:
```
np.float64(0.0)
```

[15]:
```
obs
```

[15]:
```
np.float64(0.3153910849453322)
```

[16]:
```
fig = px.histogram(stats)
fig.add_vline(x=obs, line_width=3, line_dash="dash", line_color="red")
fig.show()
```

[17]:
```
missing_data
```

[17]:
```
       YEAR  MONTH    U.S._STATE POSTAL.CODE  … PCT_LAND PCT_WATER_TOT  \
OBS                                           …
```

```
1      2011    7.0      Minnesota          MN  …    91.59          8.41
2      2014    5.0      Minnesota          MN  …    91.59          8.41
3      2010   10.0      Minnesota          MN  …    91.59          8.41
…       …       …           …          …   …      …             …
1532   2009    8.0   South Dakota          SD  …    98.31          1.69
1533   2009    8.0   South Dakota          SD  …    98.31          1.69
1534   2000    NaN         Alaska          AK  …    85.76         14.24

       PCT_WATER_INLAND MISSING_LABEL
OBS
1                  5.48         False
2                  5.48         False
3                  5.48         False
…                   …             …
1532               1.69         False
1533               1.69         False
1534               2.90          True

[1534 rows x 56 columns]
```

1.4 Step 4: Hypothesis Testing

[18]:
```
permutation_data = data_time.copy()
permutation_data["Is_California"] = (permutation_data["U.S._STATE"] ==␣
 ↪"California").astype(str)
```

[19]:
```
diff_medians(permutation_data, "OUTAGE.DURATION", "Is_California")
```

[19]:
```
np.float64(-581.5)
```

[20]:
```
n = 1000
medians_diff = []
observed_diff = diff_medians(permutation_data, "OUTAGE.DURATION",␣
 ↪"Is_California")
for _ in range(n):
    permutation_data["shuffled_labels"] = permutation_data["Is_California"].
 ↪sample(frac=1, replace=False).reset_index(drop=True)
    medians_diff.append(diff_medians(permutation_data, "OUTAGE.DURATION",␣
 ↪"shuffled_labels"))

np.mean([diff <= observed_diff for diff in medians_diff])
```

[20]:
```
np.float64(0.0)
```

1.5 Step 5: Framing a Prediction Problem

[24]:
```
# TODO
```

1.6 Step 6: Baseline Model

[25]:
```
# TODO
```

1.7 Step 7: Final Model

[26]:
```
# TODO
```

1.8 Step 8: Fairness Analysis

[27]:
```
# TODO
```