Power Outages

This project uses major power outage data in the continental U.S. from January 2000 to July 2016. Here, a major power outage is defined as a power outage that impacted at least 50,000 customers or caused an unplanned firm load loss of atleast 300MW. Interesting questions to consider include:

- Where and when do major power outages tend to occur?
- What are the characteristics of major power outages with higher severity? Variables to consider include location, time, climate, land-use characteristics, electricity consumption patterns, economic characteristics, etc. What risk factors may an energy company want to look into when predicting the location and severity of its next major power outage?
- What characteristics are associated with each category of cause?
- How have characteristics of major power outages changed over time? Is there a clear trend?

Getting the Data

The data is downloadable here.

A data dictionary is available at this article under *Table 1. Variable descriptions*.

Cleaning and EDA

- Note that the data is given as an Excel file rather than a CSV. Open the data in Excel or another spreadsheet application and determine which rows and columns of the Excel spreadsheet should be ignored when loading the data in pandas.
- Clean the data.
  - The power outage start date and time is given by `OUTAGE.START.DATE` and `OUTAGE.START.TIME`. It would be preferable if these two columns were combined into one datetime column. Combine `OUTAGE.START.DATE` and `OUTAGE.START.TIME` into a new datetime column called `OUTAGE.START`. Similarly, combine `OUTAGE.RESTORATION.DATE` and `OUTAGE.RESTORATION.TIME` into a new datetime column called `OUTAGE.RESTORATION`.
- Understand the data in ways relevant to your question using univariate and bivariate analysis of the data as well as aggregations.

Hint 1: pandas can load multiple filetypes: `pd.read_csv`, `pd.read_excel`, `pd.read_html`, `pd.read_json`, *etc.*

Hint 2: `pd.to_datetime` *and* `pd.to_timedelta` *will be useful here.*

Tip: To visualize geospatial data, consider Folium or another geospatial plotting library.

Assessment of Missingness

- Assess the missingness of a column that is not missing by design.

Hypothesis Test

Find a hypothesis test to perform. You can use the questions at the top of the notebook for inspiration.

Summary of Findings

Introduction

Electrical power is essential to any institution in society. However, power outages in the electricity network are also common in people's lives. In this project, our main focus is a large power outage dataset in which each outage influenced at least 50,000 customers or caused an unexpected firm load loss of at least 300MW. Our goal is to clean data and then evaluate the of features of the dataset such as region, duration, categories of outage via exploratory data analysis.

Cleaning and EDA

Cleaning: As we downloaded the data, we found that the original columns were not matched to their corresponding entries. So, we firstly dropped irrelevant rows, reset the indices of the dataframe and sort the order of rows based on year. Then we found that there are two pair of datetime columns (`OUTAGE.START.DATE` , `OUTAGE.START.TIME` , `OUTAGE.RESTORATION.DATE` , and `OUTAGE.RESTORATION.TIME`) which we can combined into two columns called `OUTAGE.START` and `OUTAGE.RESTORATTON` . Then we check the datatype of each column where most columns are objects. Therefore, we cleaned the data by converting any possible columns into numerical value columns.

EDA: In this section, we performed univariate analysis mainly on the columns such as `OUTAGE.START` and `YEAR` . Specifically, we want to know the distribution of hours when outage starts in a day and the tendency of happened outage in each year.To plot the distribution, we firstly define a helper function to access each timestamp object's hour attribute, then we use groupby to group each hour bin. The result is that the period from 13pm to 16pm has the biggest likelihood of outage. Besides, by plotting the number of outage in each year, we find that 2011 had the highest number of outages. Since then, the number of outage decreased significantly. To perform bivariate analysis, we select a pair of columns: `U.S._STATE` and `OUTAGE.DURATION`. This is a numeric-categorical analysis in which we are trying to find out the degree of outage duration in each States. It turns out that near the Great Lake Region, Michigan and New York States have the most extreme durations.

Assessment of Missingness

Dependent Missingness

Given the statistics of missingness in each column, it showns that `CAUSE.CATEGORY.DETAIL`, `HURRICANE.NAMES`, `DEMAND.LOSS.MW`, and `CUSTOMERS.AFFECTED` have greater than 10 percent missing value

Since our goal is to assess missingness on non-trivial column, our group decide to conduct our analysis two of the columns mentioned above.

- **Null Hypothesis:** the distribution of `U.S._STATE` is the same when column `CUSTOMERS.AFFECTED` is missing and when column `CUSTOMERS.AFFECTED` is not missing.
- **Alternative Hypothesis:** the distribution of `U.S._STATE` is the different when column `CUSTOMERS.AFFECTED` is missing and when column `CUSTOMERS.AFFECTED` is not missing.

Analysis for Dependent Missingness

Given our dataframe, we first analysis the missing value of each column and observed that `CUSTOMERS.AFFECTED` has a relatively high missing proportion. We also created a map of the US where the color for each states shows how many consumer are affect. It was shown that California and Texas has the most consumer affected missing the most. We also observed that for the difference between the proportion of missing and non-missing `CUSTOMERS.AFFECTED` is very nearly about half. Therefore, we presume that there is a correlation between `CUSTOMERS.AFFECTED` and `U.S._STATE`.

Our first step is creating a visualization using a horizontal bar to observe which states has more missing values. Washington, Texas, and California are very outstood that there is a high missing values compare to the others states, yet the remaining states are unable to identity as they are relatively have the same missing value.

We will be conducting a permuation test to identify is `CUSTOMERS.AFFECTED` column is depended on the `U.S._STATE` column with 5000 simulation with a null hypothesis of the distribution of `U.S._STATE` is the same when column `CUSTOMERS.AFFECTED` is missing and when column `CUSTOMERS.AFFECTED` is not missing. The alternative hypothesis is the distribution of `U.S._STATE` is the different when column `CUSTOMERS.AFFECTED` is missing and when column `CUSTOMERS.AFFECTED` is not missing.

Given our p-value,0.0, is under the significant level of 0.05, we **reject** our null hypothesis that the distribution of U.S._STATE is the same when column `CUSTOMERS.AFFECTED` is missing and when column `CUSTOMERS.AFFECTED` is not missing. **Therefore, it supports that `CUSTOMERS.AFFECTED` column is depended on `U.S._STATE` column i.e. MAR.**

Independent Missingness

- **Null Hypothesis:** the distribution of `YEAR` is the same when column `CLIMATE.REGION` is missing and when column `CLIMATE.REGION` is not missing.
- **Alternative Hypothesis:** the distribution of `YEAR` is the different when column `CLIMATE.REGION` is missing and when column `CLIMATE.REGION` is not missing.

Analysis for Independent Missingess

In this part, we decided to perform if `CLIMATE.REGION` is independent from `YEAR`. Given our horizontal bar graph for the amount of missing and non-missing `CLIMATE.REGION` for each year, it is very obvious that in 2000 and 2006 is relatively high compare to the other years. Yet, observing the `CLIMATE.REGION` missing and non-missing per year in the horizontal graph, it showns that the difference between the missing proportion is relatively the same. Therefore, we presume that `CLIMATE.REGION` maybe independent from `YEAR`. (*`Year` is a catergorical column in this dataframe*)

We will investigate our presumptions with a permuation test to identify if `CLIMATE.REGION` column is independent from `YEAR` column with 5000 simulation. The null hypothesis is the distribution of YEAR is the same when column `CLIMATE.REGION` is missing and when column `CLIMATE.REGION` is not missing. Alternatively, the alternative hypothesis is the distribution of `YEAR` is the different when column `CLIMATE.REGION` is missing and when column `CLIMATE.REGION` is not missing.

Given our p-value, 0.129, under the significant level of 0.05, we fail to reject our null hypothesis that the distribution of `YEAR` is the same when column `CLIMATE.REGION` is missing and when column `CLIMATE.REGION` is not missing. ***Therefore, it supports that `CLIMATE.REGION` is independent from YEAR column i.e MCAR***

Hypothesis Test

Question: Is it true that outage induced by system operability disruption are more likely to have a higher amount of customer affected than outage induced non system operability disruption?

- **Null Hypothesis:** system operability disruption and non system operability disruption have the same amount of customer affected

- **Alternative Hypothesis:** system operability disruption are more likely to have a higher amount of customer affected than outage induced non system operability disruption

Since it is determined based on group distribution, we would have to perform a permutation test to observe the group distribution difference. When we did the distribution of customers with different Causes, we observed that the bell curve of both is not the same. Futhermore, the difference between both seems to be different based on the group thats in. Therefore, we decided to use the difference in mean as our test statistic. Our permutation test has conducted 5000 simulations with a significant level of 0.05, we got a p-value of 0.0262. **Therefore, we reject the null hypothesis and supports our claim that outage induced by system operability disruption are more likely to have a higher amount of customer affected than outage induced non system operability disruption**

Code

In [351…

```
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
%matplotlib inline
%config InlineBackend.figure_format = 'retina'  # Higher resolution figures
```

Cleaning and EDA

data processing and cleaning

In [398…

```
# import data and initialize dataframe
df = ( pd.read_excel('outage.xlsx')
       .drop
       (columns = ['Major power outage events in the continental U.S.','Unnamed: 1'])
     )
col_lst = list(df.iloc[4]) # get the actual column name
df = df.drop([0,1,2,3,4,5]) # drop unrelated rows
df.columns = col_lst #replace default column names with new column names
df = df.sort_values('YEAR').reset_index().drop(columns = ['index']) #sort rows based on years
df.head(10)
```

Out[398]:

	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL	CLIMATE.CATEGORY	OUTAGE.START.DATE	OUT
0	2000	NaN	North Carolina	NC	SERC	Southeast	NaN	NaN	NaN	
1	2000	3	Texas	TX	TRE	South	-1.1	cold	2000-03-18 00:00:00	
2	2000	NaN	Texas	TX	FRCC	South	NaN	NaN	NaN	
3	2000	8	Indiana	IN	ECAR	Central	-0.5	cold	2000-08-28 00:00:00	
4	2000	NaN	Alabama	AL	SERC	Southeast	NaN	NaN	NaN	
5	2000	12	Alabama	AL	SERC	Southeast	-0.8	cold	2000-12-16 00:00:00	
6	2000	8	Alabama	AL	SERC	Southeast	-0.5	cold	2000-08-10 00:00:00	
7	2000	5	Illinois	IL	SERC	Central	-0.7	cold	2000-05-18 00:00:00	
8	2000	NaN	Illinois	IL	SERC	Central	NaN	NaN	NaN	
9	2000	8	Illinois	IL	SERC	Central	-0.5	cold	2000-08-06 00:00:00	

10 rows × 55 columns

In [404…

```
def datetime_into_str(ser):
    """
    convert the Datetime object series
    into String series
    """
    return ser.astype(str)
def time_combine(date_ser,time_ser):
    """
    combine two String series into
    a new time format String series
    """
    return (date_ser.str[0:11]+time_ser).replace(['nannan'],np.NaN)
```

In [405…

```
# combine duplicate datetime cloumns into one datetime column
datetime_col = (['OUTAGE.START.DATE','OUTAGE.START.TIME',
                 'OUTAGE.RESTORATION.DATE','OUTAGE.RESTORATION.TIME'])
for i in datetime_col:
    df[i] = datetime_into_str(df[i])
df['OUTAGE.START'] = pd.to_datetime(time_combine(df[datetime_col[0]],df[datetime_col[1]]))
```

```
df['OUTAGE.RESTORATION'] = pd.to_datetime(time_combine(df[datetime_col[2]],df[datetime_col[3]]))
df = df.drop(columns = datetime_col)
```

In [406…

```
df.head()
```

Out[406]:

	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL	CLIMATE.CATEGORY	CAUSE.CATEGORY	CAUSE.
0	2000	NaN	North Carolina	NC	SERC	Southeast	NaN	NaN	severe weather	
1	2000	3	Texas	TX	TRE	South	-1.1	cold	system operability disruption	transı
2	2000	NaN	Texas	TX	FRCC	South	NaN	NaN	equipment failure	
3	2000	8	Indiana	IN	ECAR	Central	-0.5	cold	equipment failure	
4	2000	NaN	Alabama	AL	SERC	Southeast	NaN	NaN	severe weather	

5 rows × 53 columns

In [407…

```
# check data type of dataframe
df.dtypes
```

```
Out[407]: YEAR                       object
          MONTH                      object
          U.S._STATE                 object
          POSTAL.CODE                object
          NERC.REGION                object
          CLIMATE.REGION             object
          ANOMALY.LEVEL              object
          CLIMATE.CATEGORY           object
          CAUSE.CATEGORY             object
          CAUSE.CATEGORY.DETAIL      object
          HURRICANE.NAMES            object
          OUTAGE.DURATION            object
          DEMAND.LOSS.MW             object
          CUSTOMERS.AFFECTED         object
          RES.PRICE                  object
          COM.PRICE                  object
          IND.PRICE                  object
          TOTAL.PRICE                object
          RES.SALES                  object
          COM.SALES                  object
          IND.SALES                  object
          TOTAL.SALES                object
          RES.PERCEN                 object
          COM.PERCEN                 object
          IND.PERCEN                 object
          RES.CUSTOMERS              object
          COM.CUSTOMERS              object
          IND.CUSTOMERS              object
          TOTAL.CUSTOMERS            object
          RES.CUST.PCT               object
          COM.CUST.PCT               object
          IND.CUST.PCT               object
          PC.REALGSP.STATE           object
          PC.REALGSP.USA             object
          PC.REALGSP.REL             object
          PC.REALGSP.CHANGE          object
          UTIL.REALGSP               object
          TOTAL.REALGSP              object
          UTIL.CONTRI                object
          PI.UTIL.OFUSA              object
          POPULATION                 object
          POPPCT_URBAN               object
          POPPCT_UC                  object
```

```
POPDEN_URBAN                        object
POPDEN_UC                           object
POPDEN_RURAL                        object
AREAPCT_URBAN                       object
AREAPCT_UC                          object
PCT_LAND                            object
PCT_WATER_TOT                       object
PCT_WATER_INLAND                    object
OUTAGE.START                datetime64[ns]
OUTAGE.RESTORATION          datetime64[ns]
dtype: object
```

In [408…

```
# convert columns wtih appropriate data type
to_be_numeric_col = df.columns[0:df.shape[1]-2]
for i in to_be_numeric_col:
    df[i] = pd.to_numeric(df[i],errors = 'ignore')
df['MONTH'] = df['MONTH'].astype('Int16')
```

In [409…

```
# check datatype after converting
df.dtypes.head()
```

Out[409]:

```
YEAR             int64
MONTH            Int16
U.S._STATE      object
POSTAL.CODE     object
NERC.REGION     object
dtype: object
```

Exploratory Data Analysis (EDA)

In [410…

```
# Brief information of data statistics
df.describe()
```

Out[410]:

	YEAR	MONTH	ANOMALY.LEVEL	OUTAGE.DURATION	DEMAND.LOSS.MW	CUSTOMERS.AFFECTED	RES.PRICE	COM.PRICE	
count	1534.000000	1525.000000	1525.000000	1476.000000	829.000000	1.091000e+03	1512.000000	1512.000000	15
mean	2010.119296	6.234754	-0.096852	2625.398374	536.287093	1.434562e+05	11.968373	10.135053	
std	3.822306	3.254510	0.739957	5942.483307	2196.450393	2.869863e+05	3.088631	2.824150	
min	2000.000000	1.000000	-1.600000	0.000000	0.000000	0.000000e+00	5.650000	4.700000	
25%	2008.000000	4.000000	-0.500000	102.250000	3.000000	9.650000e+03	9.540000	8.017500	
50%	2011.000000	6.000000	-0.300000	701.000000	168.000000	7.013500e+04	11.460000	9.465000	
75%	2013.000000	9.000000	0.300000	2880.000000	400.000000	1.500000e+05	13.900000	11.340000	
max	2016.000000	12.000000	2.300000	108653.000000	41788.000000	3.241437e+06	34.580000	32.140000	

8 rows × 43 columns

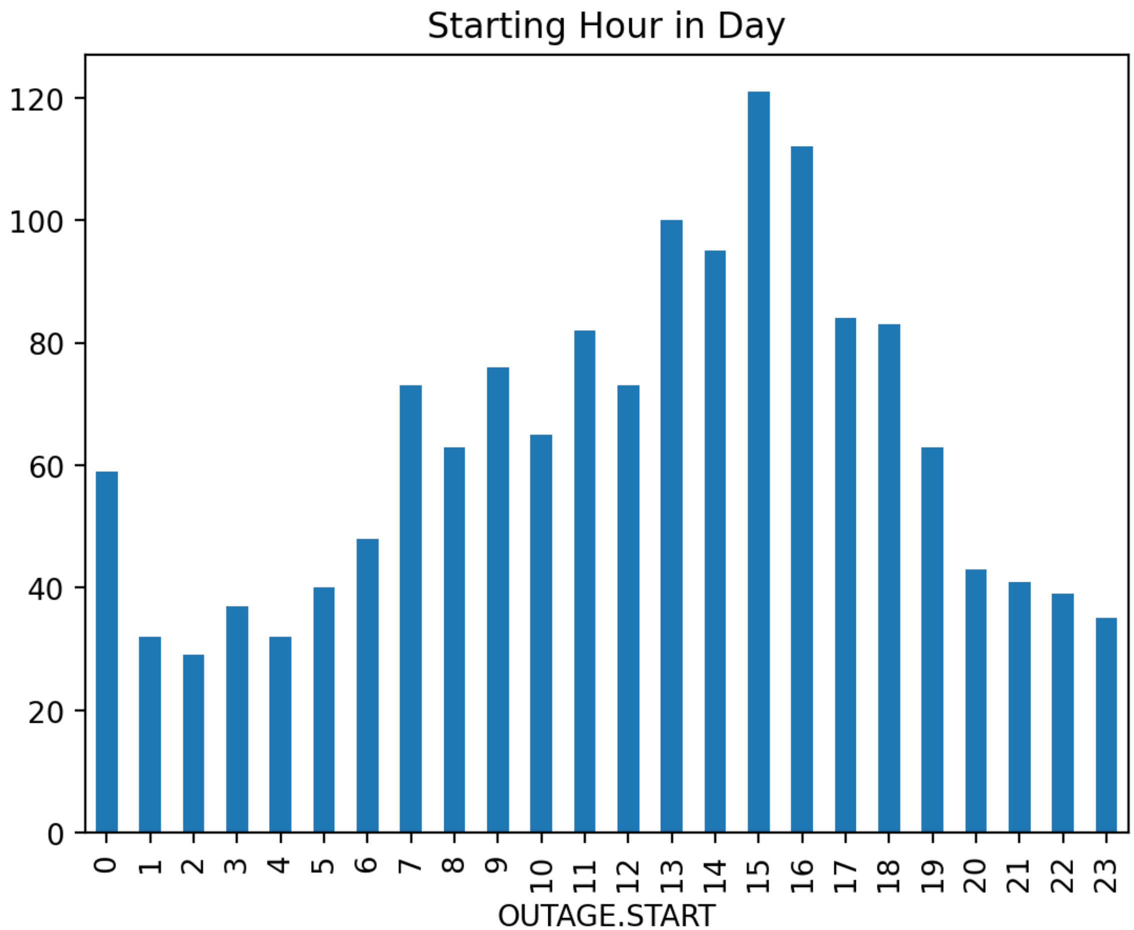
In [411...

```python
def hour_accessor(timestamp_obj):
    """
    access the data of hour from
    timestamp object
    """
    return timestamp_obj.hour
```

In [412...

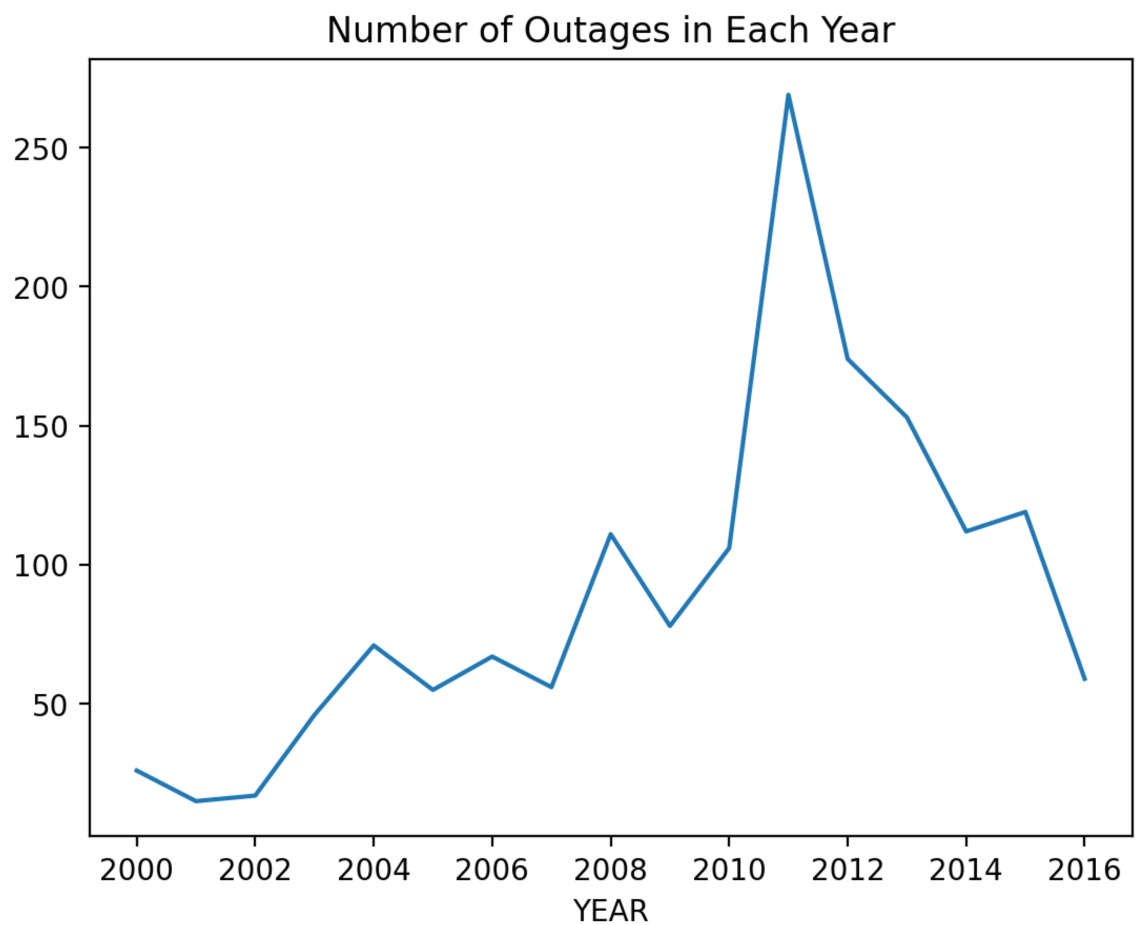
```python
# Produce the distribution of starting hour in day
outage_hours = df['OUTAGE.START'].transform(hour_accessor).to_frame().astype('Int16')
outage_hours = outage_hours.groupby('OUTAGE.START').size()
outage_hours.plot(kind = 'bar', title = 'Starting Hour in Day')
```

Out[412]: <AxesSubplot:title={'center':'Starting Hour in Day'}, xlabel='OUTAGE.START'>



In [413…

```
# Produce the tendency of outages in year
outage_in_each_year = df[['YEAR']].groupby('YEAR').size().plot(title = 'Number of Outages in Each Year')
```



In [414…
```python
import plotly.express as px
```

In [415…
```python
def change_state_abbrev(s):
    """
    convert the State full name into
    its abbreviation
    """
    for key, value in states.items():
        if s == value:
            return key
```

In [367…
```python
states = {'AK': 'Alaska','AL': 'Alabama','AR': 'Arkansas','AZ': 'Arizona','CA': 'California','CO': 'Colorado','CT': 'Connec
}
# extract the columns we want to analyze
duration_state_df = df[['OUTAGE.DURATION','U.S._STATE']]
```

```
data_ser = (duration_state_df
            .groupby('U.S._STATE')['OUTAGE.DURATION']
            .sum()) # sum up total duration in each state
dur_state_df = data_ser.to_frame().reset_index()
state_lst = list(dur_state_df['U.S._STATE'])
```

In [368…

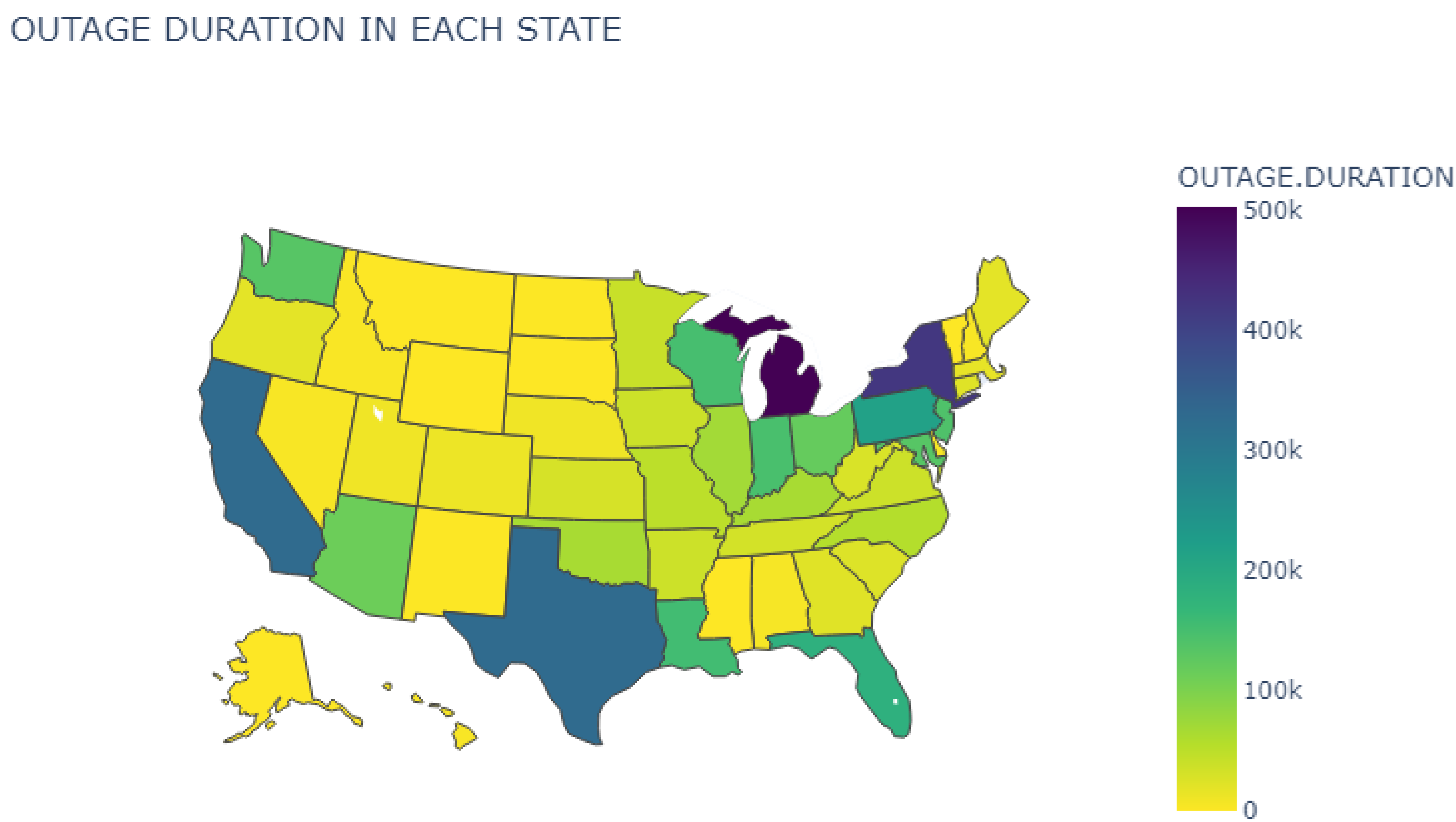
```
# replace states name with their abbreviations
for i in range(len(state_lst)):
    state_lst[i] = change_state_abbrev(state_lst[i])
dur_state_df['U.S._STATE'] = pd.Series(state_lst)
dur_state_df.head()
```

Out[368]:

	U.S._STATE	OUTAGE.DURATION
0	AL	5764.0
1	AK	0.0
2	AZ	113823.0
3	AR	37859.0
4	CA	329935.0

In [418…

```
fig = px.choropleth(dur_state_df,
                    locations='U.S._STATE',
                    locationmode="USA-states",
                    scope="usa",
                    color='OUTAGE.DURATION',
                    color_continuous_scale="Viridis_r",
                    title = 'OUTAGE DURATION IN EACH STATE'
                    )
# the code below should produce a high resolution figure
# but for pdf export purpose, we use the png file of the output instead
# fig.show()
```



Assessment of Missingness

In [372...

```
# check proportion of missingness in each column
df.isna().mean().head(15)
```

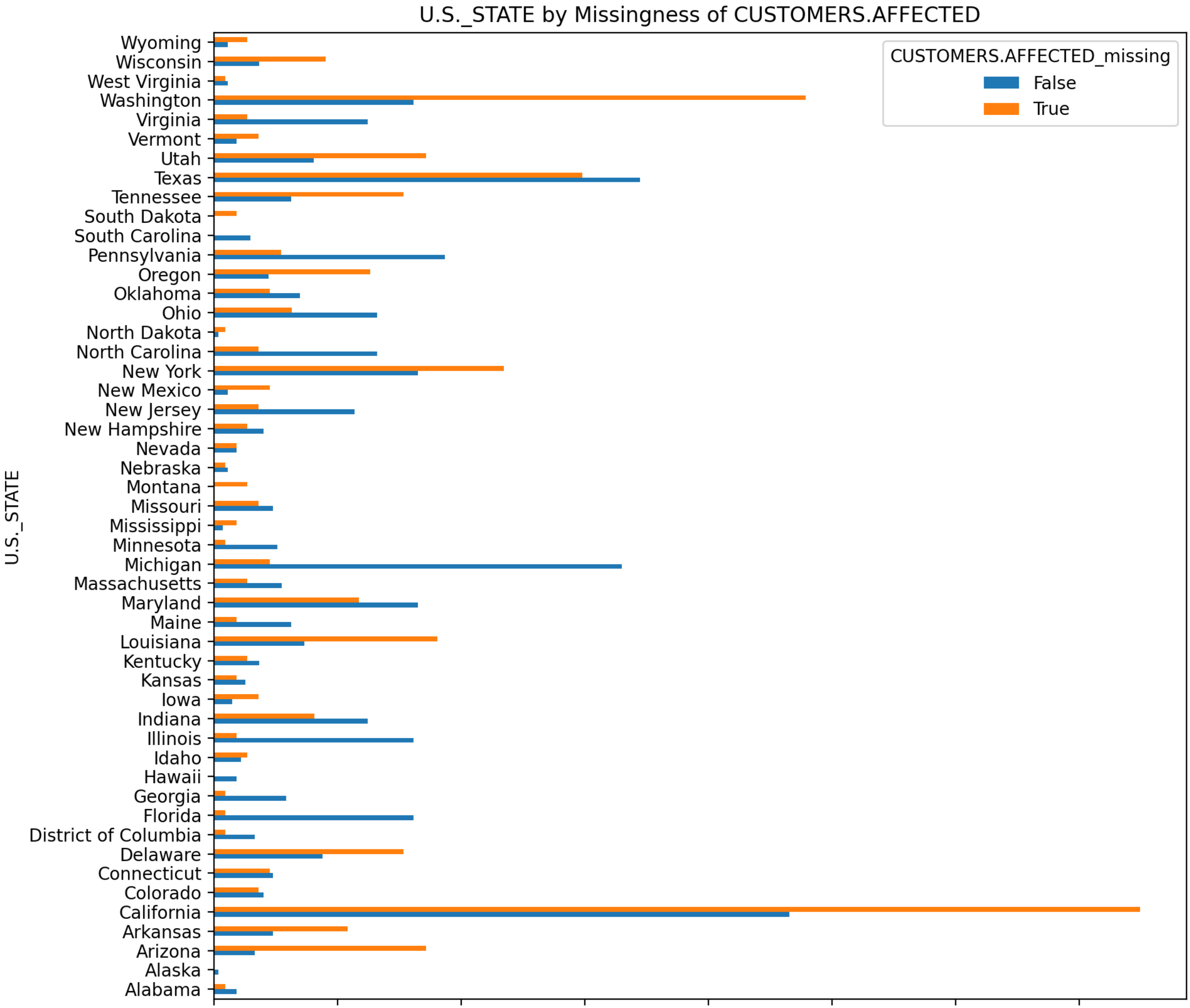
Out[372]:
```
YEAR                     0.000000
MONTH                    0.005867
U.S._STATE               0.000000
POSTAL.CODE              0.000000
NERC.REGION              0.000000
CLIMATE.REGION           0.003911
ANOMALY.LEVEL            0.005867
CLIMATE.CATEGORY         0.005867
CAUSE.CATEGORY           0.000000
CAUSE.CATEGORY.DETAIL    0.307040
HURRICANE.NAMES          0.953064
OUTAGE.DURATION          0.037810
DEMAND.LOSS.MW           0.459583
CUSTOMERS.AFFECTED       0.288787
RES.PRICE                0.014342
dtype: float64
```

In [373…
```
#find observe value and plot graph for visualization
tvds = []
shuffle = df.copy()
shuffle['CUSTOMERS.AFFECTED_missing'] = df['CUSTOMERS.AFFECTED'].isna()
shuffle = shuffle[['CUSTOMERS.AFFECTED_missing', 'U.S._STATE']]

# compute the tvd
shuffed_emp_distributions = (
    shuffle
    .pivot_table(columns='CUSTOMERS.AFFECTED_missing', index='U.S._STATE', values=None, aggfunc='size').fillna(0)
    .apply(lambda x:x/x.sum())
)

observed_tvd = np.sum(np.abs(shuffed_emp_distributions.diff(axis=1).iloc[:,-1])) / 2
shuffed_emp_distributions.plot(kind='barh', figsize=(10, 10), title='U.S._STATE by Missingness of CUSTOMERS.AFFECTED')
```

Out[373]:
```
<AxesSubplot:title={'center':'U.S._STATE by Missingness of CUSTOMERS.AFFECTED'}, ylabel='U.S._STATE'>
```


U.S._STATE by Missingness of CUSTOMERS.AFFECTED
CUSTOMERS.AFFECTED_missing
False
True
U.S._STATE
Wyoming
Wisconsin
West Virginia
Washington
Virginia
Vermont
Utah
Texas
Tennessee
South Dakota
South Carolina
Pennsylvania
Oregon
Oklahoma
Ohio
North Dakota
North Carolina
New York
New Mexico
New Jersey
New Hampshire
Nevada
Nebraska
Montana
Missouri
Mississippi
Minnesota
Michigan
Massachusetts
Maryland
Maine
Louisiana
Kentucky
Kansas
Iowa
Indiana
Illinois
Idaho
Hawaii
Georgia
Florida
District of Columbia
Delaware
Connecticut
Colorado
California
Arkansas
Arizona
Alaska
Alabama

0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175

The above is a distribution of `CUSTOMERS.AFFECTED` missing per `U.S._STATE`

Permutation Test for Dependent Missingness

- **Null Hypothesis:** the distribution of U.S._STATE is the same when column CUSTOMERS.AFFECTED is missing and when column CUSTOMERS.AFFECTED is not missing.
- **Alternative Hypothesis:** the distribution of U.S._STATE is the different when column CUSTOMERS.AFFECTED is missing and when column CUSTOMERS.AFFECTED is not missing.
- **Test-Statistic:** Total Variation Distance
- **Significant Level:** 0.05

Perform Permutation Test

In [374...
```
for i in range(5000):
    shuffled_types = (
    shuffle['U.S._STATE']
    .sample(replace=False, frac=1)
    .reset_index(drop=True)
    )

    shuffled = (
        shuffle
        .assign(**{'Shuffled Types': shuffled_types})
    )

    # compute the tvd
    shuffed_emp_distributions = (
        shuffled
        .pivot_table(columns='CUSTOMERS.AFFECTED_missing', index='Shuffled Types', values=None, aggfunc='size').fillna(0)
        .apply(lambda x:x/x.sum())
    )

    tvds.append(np.sum(np.abs(shuffed_emp_distributions.diff(axis=1).iloc[:,-1])) / 2)
```

In [391...
```
p_val = np.mean(observed_tvd <= tvds)
print (f'p_value is :{p_val}')
```

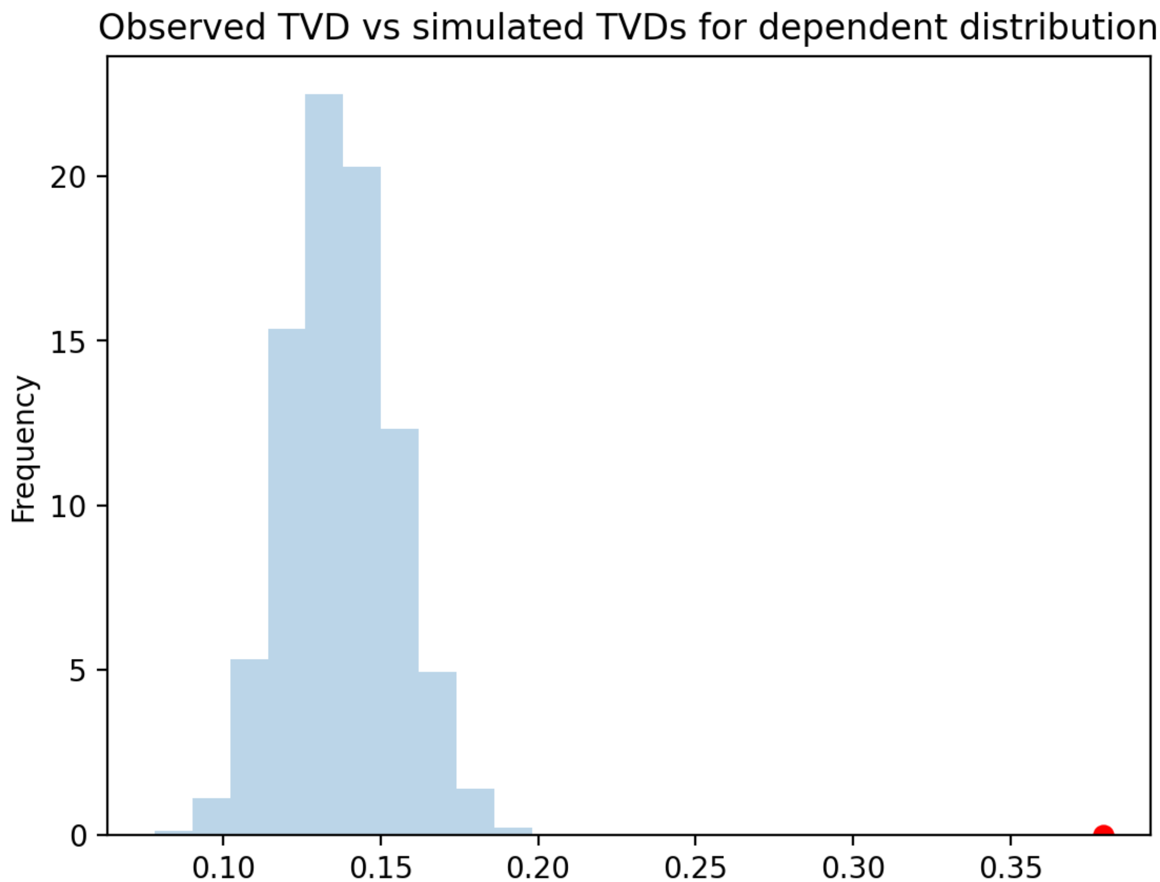
```
p_value is :0.0
```

In [376…

```
#Visualization of permutation test result
pd.Series(tvds).plot(kind='hist', density=True, alpha=0.3)
plt.scatter(observed_tvd, 0, color='red', s=40);
plt.title('Observed TVD vs simulated TVDs for dependent distribution')
```

Out[376]:

```
Text(0.5, 1.0, 'Observed TVD vs simulated TVDs for dependent distribution')
```



The p_value is 0.0, so we rejct the null hypothesis under 0.05 significance level

Permutation Test for Inependent Missingness

- **Null Hypothesis:** the distribution of `YEAR` is the same when column `CLIMATE.REGION` is missing and when column `CLIMATE.REGION` is not missing.

- **Alternative Hypothesis:** the distribution of `YEAR` is the different when column `CLIMATE.REGION` is missing and when column `CLIMATE.REGION` is not missing.
- **Test-Statistic:** Total Variation Distance
- **Significant Level:** 0.05

In [377...

```
#find observe value and plot graph for visualization
tvds_i = []
shuffle = df.copy()
shuffle['CLIMATE.REGION_missing'] = df['CLIMATE.REGION'].isna()
shuffle = shuffle[['CLIMATE.REGION_missing', 'YEAR']]

# compute the tvd
shuffed_emp_distributions = (
    shuffle
    .pivot_table(columns='CLIMATE.REGION_missing', index='YEAR', values=None, aggfunc='size').fillna(0)
    .apply(lambda x:x/x.sum())
)

observed_tvd_i = np.sum(np.abs(shuffed_emp_distributions.diff(axis=1).iloc[:,-1])) / 2
print (f'observed_tvd is :{observed_tvd_i}')
```

```
observed_tvd is :0.7030977312390925
```

Perform Permutation Test

In [386...

```
for i in range(5000):
    shuffled_types = (
    shuffle['YEAR']
    .sample(replace=False, frac=1)
    .reset_index(drop=True)
    )

    shuffled = (
        shuffle
        .assign(**{'Shuffled Types': shuffled_types})
    )

    # compute the tvd
    shuffed_emp_distributions = (
        shuffled
        .pivot_table(columns='CLIMATE.REGION_missing', index='Shuffled Types', values=None, aggfunc='size').fillna(0)
        .apply(lambda x:x/x.sum())
```

```
    )

    tvds_i.append(np.sum(np.abs(shuffed_emp_distributions.diff(axis=1).iloc[:,-1])) / 2)
```

In [389…

```
p_val = np.mean(observed_tvd_i <= tvds_i)
print (f'p_value is :{p_val}')
```

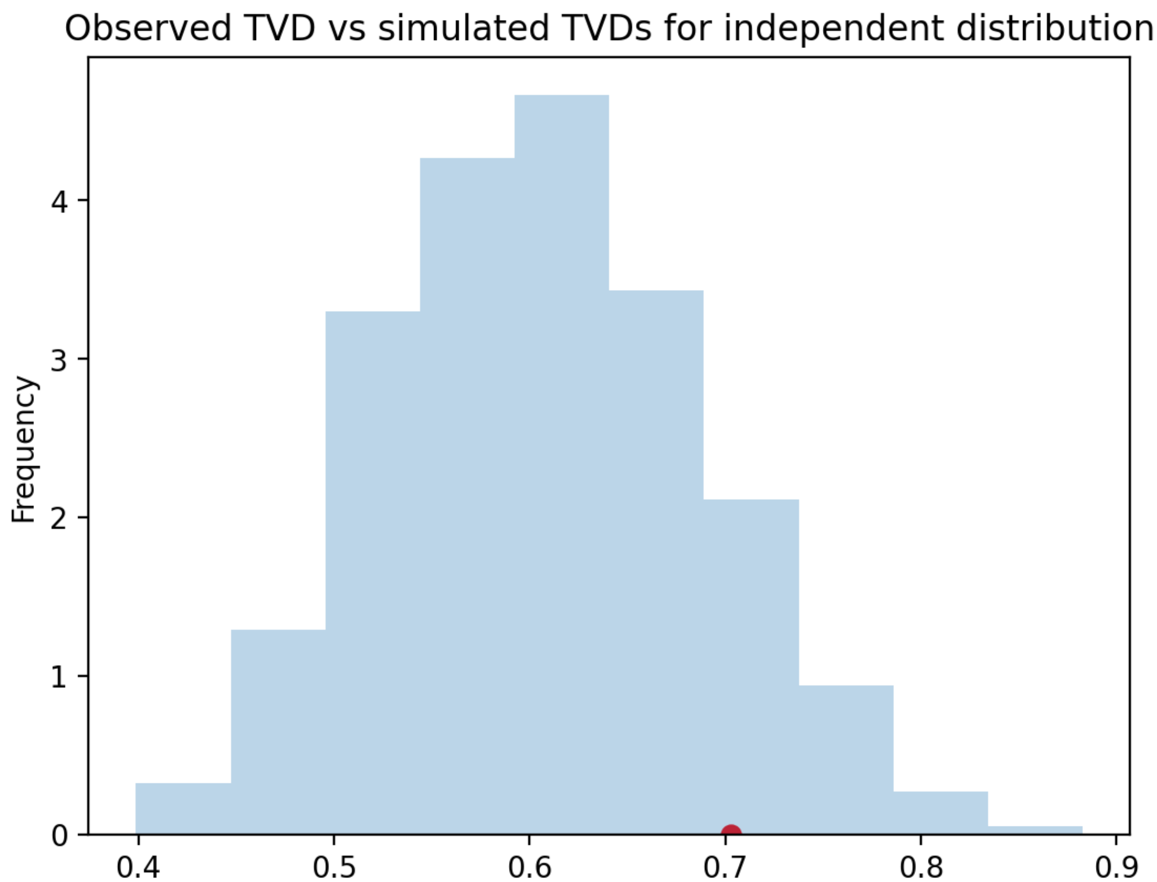
```
p_value is :0.12948
```

In [390…

```
#Visualization of permutation test result
pd.Series(tvds_i).plot(kind='hist', density=True, alpha=0.3)
plt.scatter(observed_tvd_i, 0, color='red', s=40);
plt.title('Observed TVD vs simulated TVDs for independent distribution')
```

Out[390]:

```
Text(0.5, 1.0, 'Observed TVD vs simulated TVDs for independent distribution')
```



The p_value is 0.129, so we fail to rejct the null hypothesis under 0.05 significance level

Hypothesis Test

Question: Is it true that outage induced by system operability disruption are more likely to have a higher amount of customer affected than outage induced non system operability disruption?

- **Null Hypothesis:** system operability disruption and non system operability disruption have the same amount of customer affected
- **Alternative Hypothesis:** system operability disruption are more likely to have a higher amount of customer affected than outage induced non system operability disruption
- **Test-Statistic:** Differnece in mean
- **Significant level:** 0.05

In [392…

```
#obtain only CAUSE.CATEGORY and CUSTOMERS.AFFECTED
cause_df = df[['CAUSE.CATEGORY', 'CUSTOMERS.AFFECTED']]
cause_df = cause_df.dropna()
cause_df.groupby('CAUSE.CATEGORY')['CUSTOMERS.AFFECTED'].mean()
```

Out[392]:

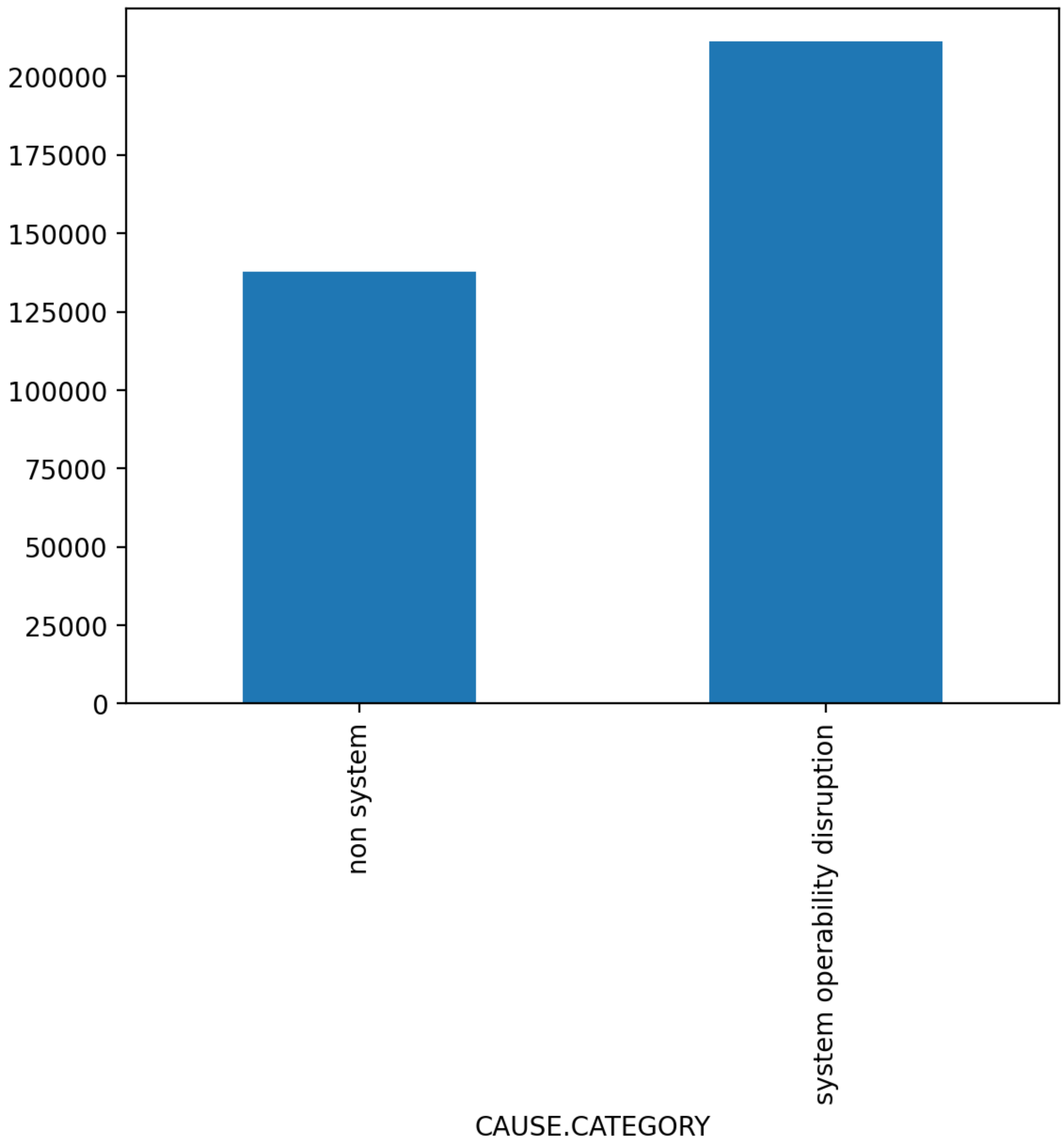
```
CAUSE.CATEGORY
equipment failure                101935.566667
fuel supply emergency                 0.142857
intentional attack                 1790.527638
islanding                          6169.088235
public appeal                      7618.761905
severe weather                   188574.801953
system operability disruption    211066.024096
Name: CUSTOMERS.AFFECTED, dtype: float64
```

In [393…

```
# Visual distribution of system operability disruption vs non system operability disruption
cause_df.loc[cause_df['CAUSE.CATEGORY'] != 'system operability disruption', 'CAUSE.CATEGORY'] = 'non system'
cause_df.groupby('CAUSE.CATEGORY')['CUSTOMERS.AFFECTED'].mean().plot(kind = 'bar')
```

Out[393]:

```
<AxesSubplot:xlabel='CAUSE.CATEGORY'>
```



In [394…

```
# Visualization of distribution of affected costomers with different Causes
cat_customer_df = df[["CAUSE.CATEGORY", "CUSTOMERS.AFFECTED"]]
cat_customer_df = cat_customer_df.dropna() #  drop Nan values
# group categories based on whether it is caused by system operation or not caused by system operation
cat_customer_df.loc[cat_customer_df['CAUSE.CATEGORY'] != 'system operability disruption', 'CAUSE.CATEGORY'] = 'non_system_c
```

```python
system_oper = cat_customer_df[cat_customer_df["CAUSE.CATEGORY"] == "system operability disruption"]
non_system_oper = cat_customer_df[cat_customer_df["CAUSE.CATEGORY"] == "non_system_caused"]
plt.figure(figsize=(7,5))
sns.distplot(system_oper["CUSTOMERS.AFFECTED"],bins = 100)
sns.distplot(non_system_oper["CUSTOMERS.AFFECTED"],bins = 10)
plt.xlim([-2000000, 3000000])
plt.ylim([0, 0.000005])
plt.legend(["caused by system operation", "non caused by system operation"])
plt.title("Distribution of costomers with Different Causes")
```

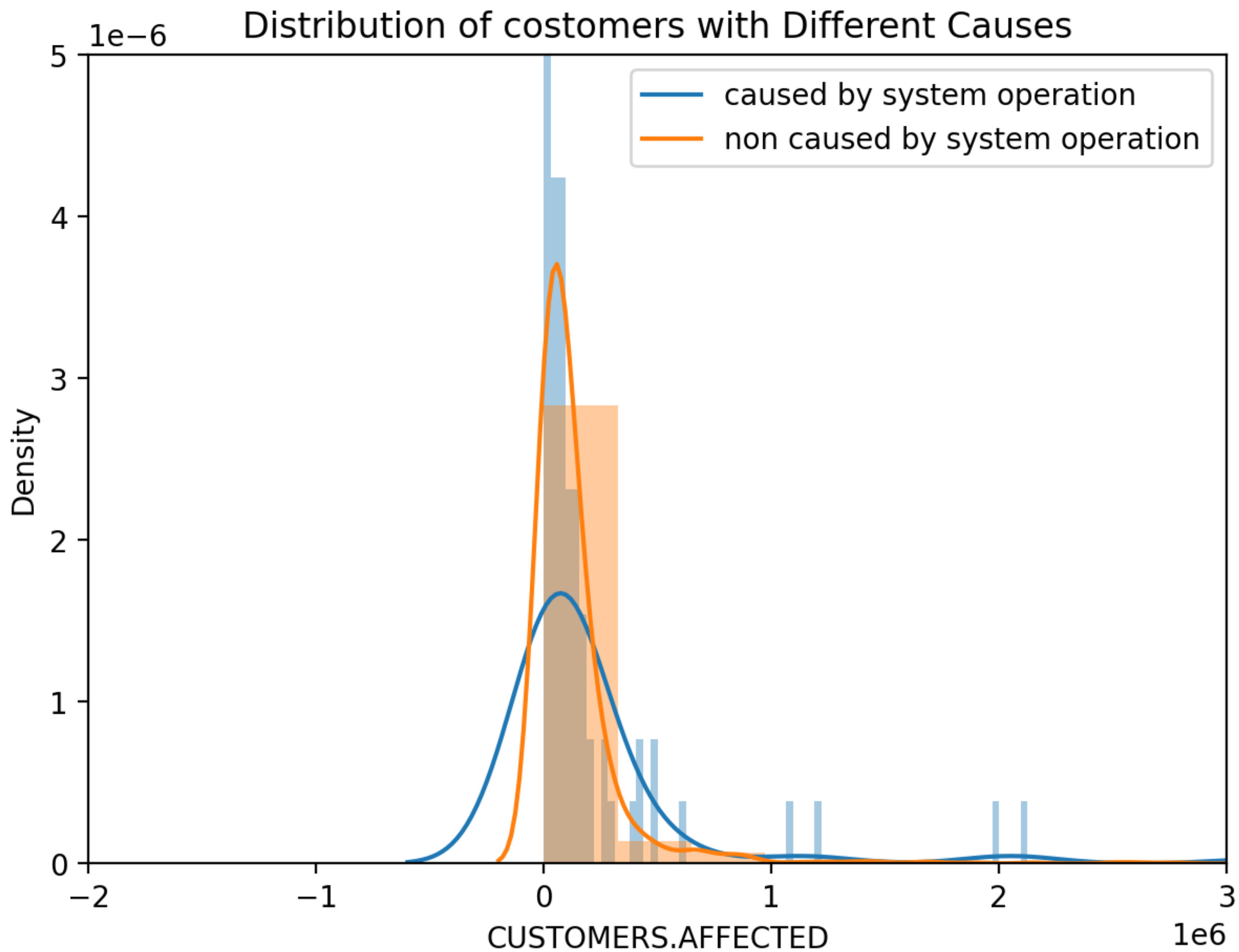
```
C:\Users\11944\anaconda3\envs\dsc80\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot
` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

C:\Users\11944\anaconda3\envs\dsc80\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot
` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```

Out[394]:
```
Text(0.5, 1.0, 'Distribution of costomers with Different Causes')
```



In [395…

```
#Get observed value
observed_diff = cause_df.groupby('CAUSE.CATEGORY')['CUSTOMERS.AFFECTED'].apply(np.mean).diff().iloc[-1]
observed_diff
```

Out[395]: 73176.8782630522

In [396…

```
#perform hypothesis tesing
diffs = []
for i in range(5000):
    shuffle = cause_df
    shuffle['CUSTOMERS.AFFECTED'] = np.random.permutation(shuffle['CUSTOMERS.AFFECTED'])
    diff = shuffle.groupby('CAUSE.CATEGORY')['CUSTOMERS.AFFECTED'].apply(np.mean).diff().iloc[-1]
    diffs.append(diff)
```

In [397…

```
p_val = np.mean(observed_diff <= diffs)
p_val
```

Out[397]:

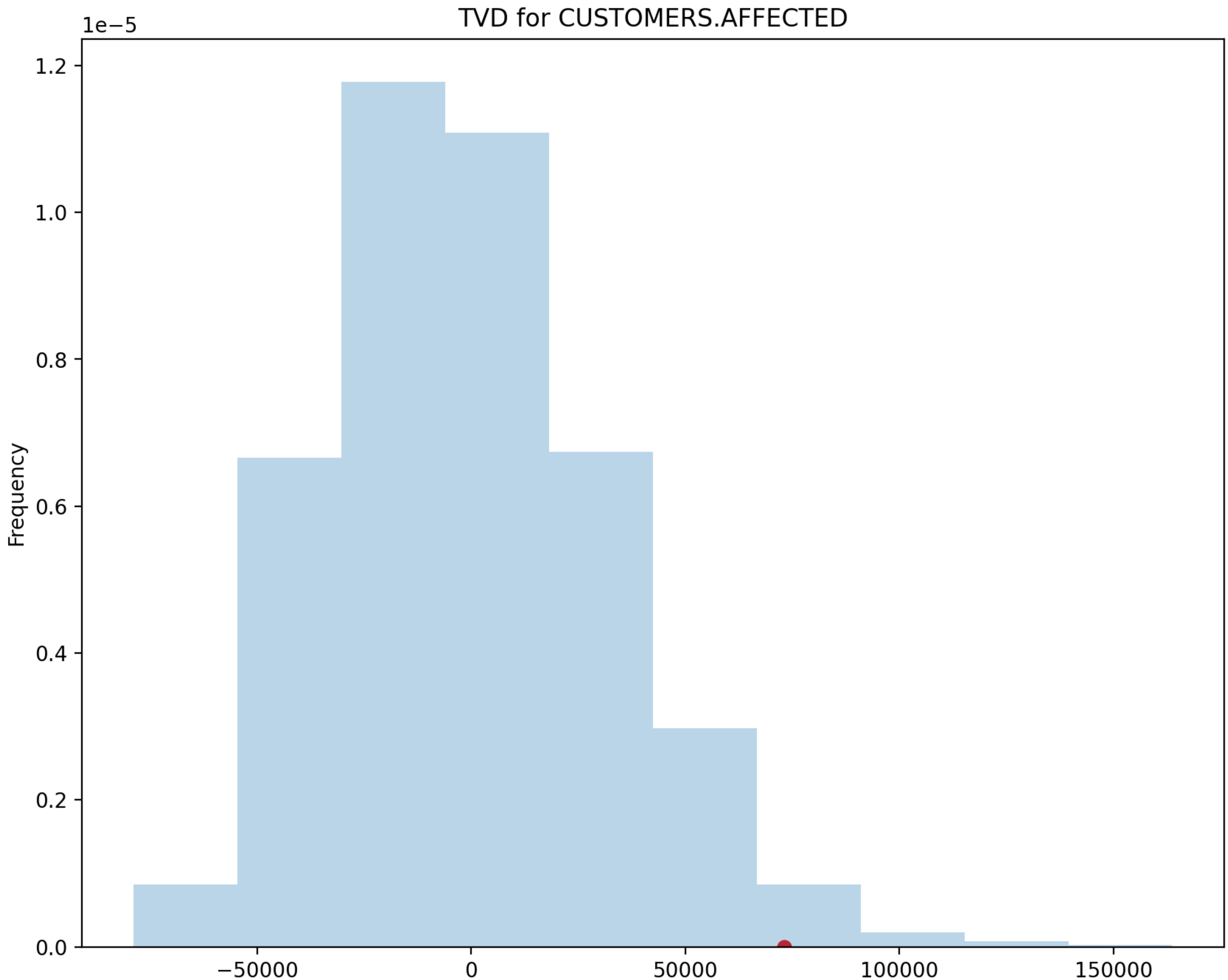
```
0.0262
```

In [342…

```
# Visualization of TVD for CUSTOMERS.AFFECTED
title = 'TVD for CUSTOMERS.AFFECTED'
plt.figure(figsize=(10,8))
plt.scatter(observed_diff, 0, color='red', s=40)
pd.Series(diffs).plot(kind='hist', density=True, title=title, alpha=0.3)
```

Out[342]:

```
<AxesSubplot:title={'center':'TVD for CUSTOMERS.AFFECTED'}, ylabel='Frequency'>
```



Given the p-value = 0.0262, which is smaller than our significant level, 0.05, so we reject the null hypothesis.