Power Outages

- **See the main project notebook for instructions to be sure you satisfy the rubric!**
- See Project 03 for information on the dataset.
- A few example prediction questions to pursue are listed below. However, don't limit yourself to them!
  - Predict the severity (number of customers, duration, or demand loss) of a major power outage.
  - Predict the cause of a major power outage.
  - Predict the number and/or severity of major power outages in the year 2020.
  - Predict the electricity consumption of an area.

Be careful to justify what information you would know at the "time of prediction" and train your model using only those features.

Summary of Findings

Introduction

We are attempting to predict the climate category (cold, neutral, warm) in which an outage occurs based on data regarding the outage itself and the location in which it occured. This is a classification problem with 3 possible classifications. The evaluation metric of our model will be its accuracy in predicting the climate category, objective is to reach a 70% accuracy in our model.

Baseline Model

Features:

- Quantitative: 3

`AREAPCT_URBAN`, `OUTAGE.DURATION`, `CUSTOMERS.AFFECTED`

- Nominal: 4

`U.S._STATE`, `CLIMATE.REGION`, `CAUSE.CATEGORY`, `MONTH`

Performance:

- Accuracy: 0.65
- Accuracy is the best metric for our purposes as we only care about our models correctness in prediction

Final Model

Added Features:

1. Added an engineered region feature according to the State in which the outage occured in order to provide more generalizable classifications of weather conditions
2. Added Population and Population Density of Urban Areas as these can coorrelate to certain type of cities, namely warmer cities leading to better predictions of weather categories
3. Binarize customers affected based on the median amount due to its high standard deviation.

Features:

- Quantitative: 5

`AREAPCT_URBAN`, `POPDEN_URBAN`, `POPULATION`, `OUTAGE.DURATION`, `CUSTOMERS.AFFECTED`

- Nominal: 4

`U.S._STATE`, `CLIMATE.REGION`, `CAUSE.CATEGORY`, `MONTH`

Model Type:

- `RandomForestClassifier()`

Parameters:

- `{criterion= 'entropy', max_depth=260, max_features='auto'}`

Performance:

- Accuracy: 0.71

Fairness Evaluation

Our fairness evaluation concerns the region in which an outage occurs to ensure that all unique parts of the country are equally represented in our classification in terms of its accuracy

- Null Hypothesis: The classifier's accuracy is the same across regions
- Alternative Hypothesis: The classifier's accuracy is different across regions

Test statistic: Absolute Difference in accuracy between regions
Significance level: 0.01
Outcome: 0.56 (Fail to reject the null)

Code

In [1]:
```
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
```

In [2]:
```
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import Binarizer
import warnings
warnings.simplefilter('ignore')
```

Data Set

In [3]:
```
# importing the excel file
data = pd.read_excel('outage.xlsx')
data = data[4:].reset_index(drop=True)

#renaming the column names to match the excel sheet
data.columns = data.iloc[0]

# dropping unneccesary columns
data = data.drop([0]).drop([1]).drop(columns = ['variables', 'OBS', 'YEAR']).reset_index(drop=True)

# DataFrame
data.head()
```

Out[3]:

	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL	CLIMATE.CATEGORY	OUTAGE.START.DATE	OUTAGE.!
0	7	Minnesota	MN	MRO	East North Central	-0.3	normal	2011-07-01 00:00:00	
1	5	Minnesota	MN	MRO	East North Central	-0.1	normal	2014-05-11 00:00:00	
2	10	Minnesota	MN	MRO	East North Central	-1.5	cold	2010-10-26 00:00:00	
3	6	Minnesota	MN	MRO	East North Central	-0.1	normal	2012-06-19 00:00:00	
4	7	Minnesota	MN	MRO	East North Central	1.2	warm	2015-07-18 00:00:00	

5 rows × 54 columns

Baseline Model

With the `OUTAGES` dataset, the features that were most prevalent in determining the `CLIMATE.CATEGORY` were: `AREAPCT_URBAN`, `POPULATION`, `OUTAGE.DURATION`, `CUSTOMERS.AFFECTED`, `NERC.REGION`, `CLIMATE.REGION`, `CAUSE.CATEGORY`, `MONTH`, `U.S._STATE`.

In [4]:
```
data = data.dropna(how='any', subset=['CLIMATE.CATEGORY'])
data[['POPULATION', 'AREAPCT_URBAN', 'TOTAL.PRICE', 'CUSTOMERS.AFFECTED']] = data[['POPULATION',
                                                                                  'AREAPCT_URBAN', 'TOTAL.PRICE',
                                                                                  'CUSTOMERS.AFFECTED']].fillna(0)
data[['NERC.REGION', 'CLIMATE.REGION', 'CAUSE.CATEGORY']] = data[['NERC.REGION',
                                                                  'CLIMATE.REGION',
                                                                  'CAUSE.CATEGORY']].fillna('missing')
data['OUTAGE.DURATION'] = data['OUTAGE.DURATION'].fillna(0)
```

In [5]:
```
preproc = ColumnTransformer(
        transformers=[
            ('as is', FunctionTransformer(), ['AREAPCT_URBAN', 'OUTAGE.DURATION',
                                              'CUSTOMERS.AFFECTED']),
            ('ohe', OneHotEncoder(handle_unknown='ignore'), ['U.S._STATE',
                                                             'CLIMATE.REGION', 'CAUSE.CATEGORY',
                                                             'MONTH'])
        ]
    )

base_pl = Pipeline([
        ('preprocessing', preproc),
        ('f', RandomForestClassifier())
    ])
```

In [6]:
```
features = data.drop('CLIMATE.CATEGORY', axis=1)
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    data['CLIMATE.CATEGORY'],
                                                    test_size=0.3)
base_pl.fit(X_train, y_train)
base_pl.score(X_test, y_test)
```

Out[6]:
```
0.6572052401746725
```

Final Model

In [7]:
```
# importing the excel file
data = pd.read_excel('outage.xlsx')
data = data[4:].reset_index(drop=True)

#renaming the column names to match the excel sheet
data.columns = data.iloc[0]

# dropping unneccesary columns
data = data.drop([0]).drop([1]).drop(columns = ['variables', 'OBS', 'YEAR']).reset_index(drop=True)

# DataFrame
data.head()

data = data.dropna(how='any', subset=['CLIMATE.CATEGORY'])
data[['POPULATION', 'AREAPCT_URBAN', 'TOTAL.PRICE', 'CUSTOMERS.AFFECTED']] = data[['POPULATION',
                                                                                  'AREAPCT_URBAN',
                                                                                  'TOTAL.PRICE',
                                                                                  'CUSTOMERS.AFFECTED']].fillna(0)
data[['NERC.REGION', 'CLIMATE.REGION', 'CAUSE.CATEGORY']] = data[['NERC.REGION', 'CLIMATE.REGION',
                                                                  'CAUSE.CATEGORY']].fillna('missing')
data['OUTAGE.DURATION'] = data['OUTAGE.DURATION'].fillna(0)
```

With the states, we found the correlating region that they are in. This was created in order to generalized classification conditions.

In [8]:
```
states_to_regions = {
    'Washington': 'West', 'Oregon': 'West', 'California': 'West', 'Nevada': 'West',
    'Idaho': 'West', 'Montana': 'West', 'Wyoming': 'West', 'Utah': 'West',
    'Colorado': 'West', 'Alaska': 'West', 'Hawaii': 'West', 'Maine': 'Northeast',
    'Vermont': 'Northeast', 'New York': 'Northeast', 'New Hampshire': 'Northeast',
    'Massachusetts': 'Northeast', 'Rhode Island': 'Northeast', 'Connecticut': 'Northeast',
```

```
    'New Jersey': 'Northeast', 'Pennsylvania': 'Northeast', 'North Dakota': 'Midwest',
    'South Dakota': 'Midwest', 'Nebraska': 'Midwest', 'Kansas': 'Midwest',
    'Minnesota': 'Midwest', 'Iowa': 'Midwest', 'Missouri': 'Midwest', 'Wisconsin': 'Midwest',
    'Illinois': 'Midwest', 'Michigan': 'Midwest', 'Indiana': 'Midwest', 'Ohio': 'Midwest',
    'West Virginia': 'South', 'District of Columbia': 'South', 'Maryland': 'South',
    'Virginia': 'South', 'Kentucky': 'South', 'Tennessee': 'South', 'North Carolina': 'South',
    'Mississippi': 'South', 'Arkansas': 'South', 'Louisiana': 'South', 'Alabama': 'South',
    'Georgia': 'South', 'South Carolina': 'South', 'Florida': 'South', 'Delaware': 'South',
    'Arizona': 'Southwest', 'New Mexico': 'Southwest', 'Oklahoma': 'Southwest',
    'Texas': 'Southwest'}
```

Changed `U.S._STATE` from state names into regions.

In [9]:
```
data = data.dropna(how='any', subset=['CLIMATE.CATEGORY'])
data['U.S._STATE'] = data['U.S._STATE'].apply(lambda s: states_to_regions[s])
features = data.drop('CLIMATE.CATEGORY', axis=1)
```

In [10]:
```
data.head()
```

Out[10]:

	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL	CLIMATE.CATEGORY	OUTAGE.START.DATE	OUTAGE.!
0	7	Midwest	MN	MRO	East North Central	-0.3	normal	2011-07-01 00:00:00	
1	5	Midwest	MN	MRO	East North Central	-0.1	normal	2014-05-11 00:00:00	
2	10	Midwest	MN	MRO	East North Central	-1.5	cold	2010-10-26 00:00:00	
3	6	Midwest	MN	MRO	East North Central	-0.1	normal	2012-06-19 00:00:00	
4	7	Midwest	MN	MRO	East North Central	1.2	warm	2015-07-18 00:00:00	

5 rows × 54 columns

Binarized `CUSTOMERS.AFFECTED` since the spread was high. We created a threadhold of `7000` because we found that it was the median number of customers affected.

We also made a `FunctionTransformer` to log all the `OUTAGE.DURATION` values because the range was large for different purposes and the `DURATION` feature was not a significant feature to weigh.

In [11]:
```
preproc2 = ColumnTransformer(
        transformers=[
            ('as is', FunctionTransformer(), ['AREAPCT_URBAN', 'POPDEN_URBAN',
                                              'POPULATION']),
            ('log scale', FunctionTransformer(lambda ser: (ser + 0.1).apply(np.log)), ['OUTAGE.DURATION']),
            ('bin', Binarizer(threshold=7000), ['CUSTOMERS.AFFECTED']),
            ('ohe', OneHotEncoder(handle_unknown='ignore'), ['CLIMATE.REGION',
                                                               'CAUSE.CATEGORY', 'MONTH', 'U.S._STATE'])
        ]
    )

pl = Pipeline([
        ('preprocessing', preproc2),
        ('f', RandomForestClassifier())
    ])
```

We ran a `GridSearch` in order to find the best hyperparameters for `RandomForestClassifier`

In [12]:
```
hyperparameters = {
    'f__max_depth' : np.arange(200, 400, 20),
    'f__criterion' :['gini', 'entropy']
}
```

In [13]:
```
searcher = GridSearchCV(pl, param_grid = hyperparameters, cv=5)
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    data['CLIMATE.CATEGORY'],
                                                    test_size=0.3)
searcher.fit(X_train, y_train)
```

Out[14]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('preprocessing',
                                        ColumnTransformer(transformers=[('as '
                                                                         'is',
                                                                         FunctionTransformer(),
                                                                         ['AREAPCT_URBAN',
                                                                          'POPDEN_URBAN',
                                                                          'POPULATION']),
                                                                        ('log '
                                                                         'scale',
                                                                         FunctionTransformer(func=<function <lambda> at
0x0000025156775A60>),
                                                                         ['OUTAGE.DURATION']),
                                                                        ('bin',
                                                                         Binarizer(threshold=7000),
                                                                         ['CUSTOMERS.AFFECTED']),
                                                                        ('ohe',
                                                                         OneHotEncoder(handle_unknown='ignore'),
                                                                         ['CLIMATE.REGION',
                                                                          'CAUSE.CATEGORY',
                                                                          'MONTH',
                                                                          'U.S._STATE'])])),
                                       ('f', RandomForestClassifier())]),
             param_grid={'f__criterion': ['gini', 'entropy'],
                         'f__max_depth': array([200, 220, 240, 260, 280, 300, 320, 340, 360, 380])})
```

In [15]:
```
searcher.best_params_
```

Out[15]:
```
{'f__criterion': 'entropy', 'f__max_depth': 260}
```

The final pipeline with the best hyperparameters.

In [25]:
```
final_pl = Pipeline([
        ('preprocessing', preproc2),
        ('f', RandomForestClassifier(criterion= 'entropy', max_depth=260, max_features='auto'))
    ])
```

In [26]:
```
X_train, X_test, y_train, y_test = train_test_split(features,
                                                     data['CLIMATE.CATEGORY'],
                                                     test_size=0.3)
final_pl.fit(X_train, y_train)
```

Out[26]:
```
Pipeline(steps=[('preprocessing',
                 ColumnTransformer(transformers=[('as is',
                                                  FunctionTransformer(),
                                                  ['AREAPCT_URBAN',
                                                   'POPDEN_URBAN',
                                                   'POPULATION']),
                                                 ('log scale',
                                                  FunctionTransformer(func=<function <lambda> at 0x0000025156775A60>),
                                                  ['OUTAGE.DURATION']),
                                                 ('bin',
                                                  Binarizer(threshold=7000),
                                                  ['CUSTOMERS.AFFECTED']),
                                                 ('ohe',
                                                  OneHotEncoder(handle_unknown='ignore'),
                                                  ['CLIMATE.REGION',
                                                   'CAUSE.CATEGORY', 'MONTH',
                                                   'U.S._STATE'])])),
                ('f',
                 RandomForestClassifier(criterion='entropy', max_depth=260))])
```

In [27]:
```
final_pl.score(X_test, y_test)
```

Out[27]:
```
0.7096069868995634
```

Fairness Evaluation

In [19]:
```
import sklearn.metrics as metrics
```

In [20]:
```
results = X_test
results['prediction'] = final_pl.predict(X_test)
results['tag'] = y_test
```

In [21]:
```
results
```

Out[21]:

	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL	OUTAGE.START.DATE	OUTAGE.START.TIME	OUTA
789	6	Northeast	NJ	RFC	Northeast	-0.5	2008-06-09 00:00:00	14:52:00	
63	3	Midwest	WI	MRO	East North Central	-0.4	2014-03-04 00:00:00	09:06:00	
813	1	South	SC	SERC	Southeast	-0.5	2014-01-07 00:00:00	18:00:00	
661	12	West	UT	WECC	Southwest	-0.3	2013-12-06 00:00:00	08:47:00	
1107	5	West	CA	WECC	West	-0.2	2007-05-14 00:00:00	11:15:00	
...	...	...	...	...	...	...	...	...	
571	7	South	MD	RFC	Northeast	-0.9	2010-07-25 00:00:00	15:20:00	
1245	10	West	CA	WECC	West	-1.1	2007-10-22 00:00:00	14:05:00	
1433	1	Northeast	MA	NPCC	Northeast	-1.3	2011-01-12 00:00:00	06:00:00	
413	7	West	WA	WECC	Northwest	1.2	2015-07-21 00:00:00	12:47:00	
906	1	South	DE	RFC	Northeast	-1.3	2011-01-27 00:00:00	09:30:00	

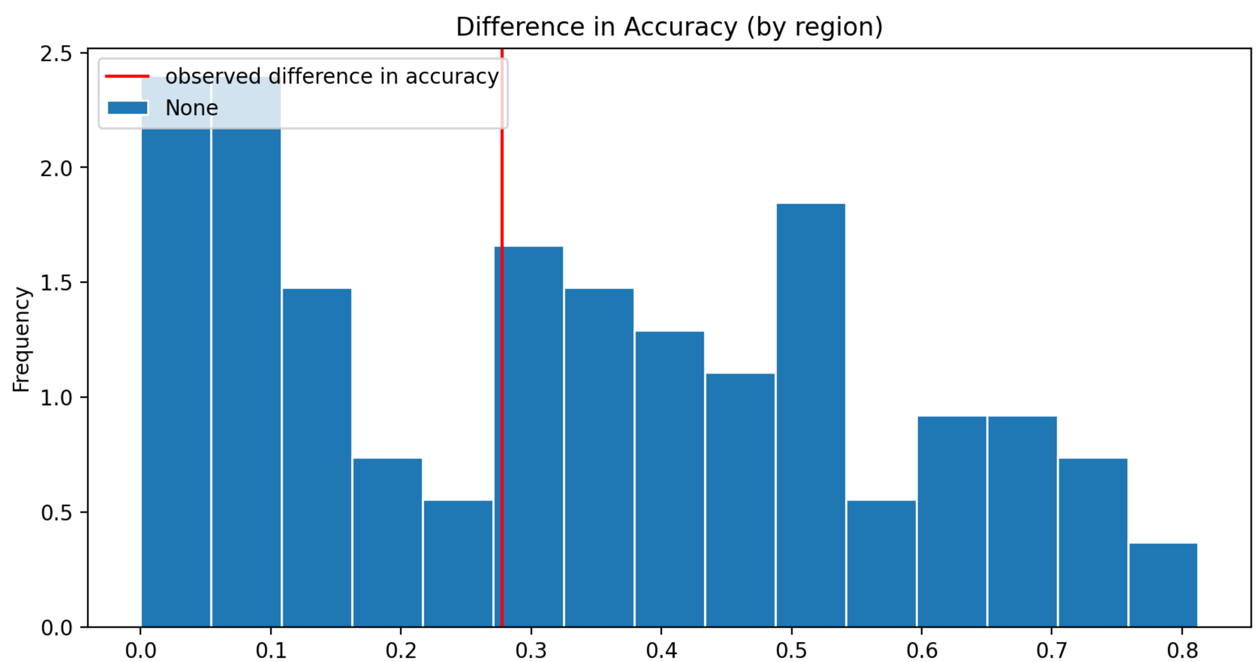
458 rows × 55 columns

Ran a permutation test based on accuracy.

In [22]:
```
obs = results.groupby('CLIMATE.REGION').apply(lambda x: metrics.accuracy_score(x['tag'],
                                                                               x['prediction'])).diff().iloc[-1]

diff_in_acc = []
for _ in range(100):
    df = results[['CLIMATE.REGION', 'prediction', 'tag']]
    df['CLIMATE.REGION'] = results['CLIMATE.REGION'].sample(frac=1.0,
                                                            replace=False).reset_index(drop=True)
    out = df.groupby('CLIMATE.REGION').apply(lambda x: metrics.accuracy_score(x['tag'], x['prediction'])).diff().iloc[-1
    diff_in_acc.append(abs(out))
```

In [23]:
```
plt.figure(figsize=(10, 5))
pd.Series(diff_in_acc).plot(kind='hist', ec='w', density=True, bins=15, title='Difference in Accuracy (by region)')
plt.axvline(x=abs(obs), color='red', label='observed difference in accuracy')
plt.legend(loc='upper left');
```



In [24]:
```
p_val = (np.array(diff_in_acc) >= abs(obs)).mean()
p_val
```

Out[24]: 0.58

```
In [ ]:
```