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

In the power outages dataset, I will be attempting a classification problem to determine the cause of a major power outage. The target variable in this case is very simply the "CAUSE.CATEGORY" variable in the dataset, as it is able to distinguish between seven general forms of ways to cause a major power outage. And the evaluation metric that I will be looking at will be accuracy.

Baseline Model

The baseline model had 3 features:

```
- 2 Nominal ('CLIMATE.REGION', 'HURRICANE.NAMES')
- 1 Quantitative ('POPULATION')
```

and the target variable ('CAUSE.CATEGORY'). The baseline model ran on the KNeighbors Classifier had an accuracy performance of about 0.69 in the training set and about 0.60 in the testing set. I think the numbers are better than I expected given the simlicity of the varibles used, and the lack of quanititative and oridinal data. But I do think that there is plent of room for improvement.

Final Model

The final model had one more feature added and two new features engineered:

```
- 1 Quantitative added('ANOMALY.LEVEL')
- 'HURRICANE.NAMES' was one hot encoded on whether there was or
wasn't a hurricane
- 'POPULATION' was normalized
```

The final model ran on a Decision Tree Classifier with the best 'max_depth' parameter being the most optimal at 15. The model was able to perform better in both training and in testing. The training accuracy was about 0.89 and the testing was about 0.64. Both improvements from the basline model.

Fairness Evaluation

I performed a fairness evaluation on a subset of the data consisting of whether or not an outage was caused by a hurricane. So the variable 'HURRICANE.NAMES' became 'is_hurricane' and then I explored whether the model was fairer towards outages caused by hurricanes or not. I used accuracy as a parity measure because I wanted to determine whether my model has higher a higher success of correctly predicting the label for an outage caused by a hurricane.

For the permutation test with a significance level of 0.05 we had the following null and alternative hypothesis:

```
- H0: My model is fair, the accuracy for my hurricane and no
hurricane are about the same.
- H1: My model is unfair; the accuracy for hurricane is higher than
with no hurricane.
```

From the permuation test as shown below in the coding section, we must reject the null hypothesis and deternmine that there is no accuracy parity

Code

In [1]:

```
import warnings
warnings.filterwarnings('ignore')

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 figu
```

In [2]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

from sklearn import metrics
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
```

Load and filter dataset

In [3]:

```
outages = pd.read_excel('outage.xlsx', index_col=1, skiprows=5, comment='#'
outages = outages.drop(columns=['variables']).iloc[1:].reset_index(drop=Tru
outages = outages[['CAUSE.CATEGORY', 'CLIMATE.REGION', 'POPULATION',
                   'HURRICANE.NAMES'
                  ]]
outages.head()
```

Out[3]:

	CAUSE.CATEGORY	CLIMATE.REGION	POPULATION	HURRICANE.NAMES
0	severe weather	East North Central	5348119.0	NaN
1	intentional attack	East North Central	5457125.0	NaN
2	severe weather	East North Central	5310903.0	NaN
3	severe weather	East North Central	5380443.0	NaN
4	severe weather	East North Central	5489594.0	NaN

Clean columns, drop NaN, and make copy

In [4]:

```
outages['HURRICANE.NAMES'] = outages['HURRICANE.NAMES'].replace({np.NaN: 'N
```

In [5]:

```
outages.isna().sum() ## Few NaN, we can drop
```

Out[5]:

```
CAUSE.CATEGORY      0
CLIMATE.REGION      6
POPULATION          0
HURRICANE.NAMES     0
dtype: int64
```

In [6]:

```
outages = outages.dropna() ## drop NaNs
outages_final = outages.copy() ## Copy that will be used in the final model
```

Feature Engineering for baselinemodel

In [7]:

```
## One Hot region and Hurricane Names
outage_regions = pd.get_dummies(outages[['CLIMATE.REGION', 'HURRICANE.NAMES
                                prefix=['REGION', 'NAME'])
```

In [8]:

```
outages = outages.join(outage_regions).drop(['CLIMATE.REGION', 'HURRICANE.N
                                            axis=1)
outages.head()
```

Out[8]:

	CAUSE.CATEGORY	POPULATION	REGION_Central	REGION_East North Central	REGION_Northeast	REGION_N
0	severe weather	5348119.0	0	1	0	
1	intentional attack	5457125.0	0	1	0	
2	severe weather	5310903.0	0	1	0	
3	severe weather	5380443.0	0	1	0	
4	severe weather	5489594.0	0	1	0	

5 rows × 34 columns

Baseline Model

In [9]:
```
X = outages.drop('CAUSE.CATEGORY', axis=1)
y = outages['CAUSE.CATEGORY']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, r
```

In [10]:
```
## Train using KNeighbors Classifier
base_pl = Pipeline(steps=[('knn',
                           KNeighborsClassifier(n_neighbors=7))]) ## n=7 be
base_pl.fit(X_train, y_train)
print('TRAINING ACCURACY: ' + str(base_pl.score(X_train, y_train)))
```

```
TRAINING ACCURACY: 0.6913002806361085
```

In [11]:
```
print("TESTING ACCURACY: " + str(base_pl.score(X_test, y_test)))
```

```
TESTING ACCURACY: 0.6056644880174292
```

Final Model

In [12]:
```
outages_final = pd.read_excel('outage.xlsx', index_col=1, skiprows=5, comme
outages_final = outages_final.drop(columns=['variables']).iloc[1:].reset_in
outages_final = outages_final[['CAUSE.CATEGORY', 'CLIMATE.REGION', 'POPULAT
                    'ANOMALY.LEVEL','HURRICANE.NAMES'
                   ]]
outages_final['HURRICANE.NAMES'] = outages_final['HURRICANE.NAMES'].replace
outages_final = outages_final.dropna()
```

Feature Engineering and cleaning for final model

In [13]:
```
outages_final['HURRICANE.NAMES'] = outages_final['HURRICANE.NAMES'].apply(l
```

In [14]:
```
outage_reg = pd.get_dummies(outages_final['CLIMATE.REGION'], prefix='REGION
outages_final = outages_final.join(outage_reg).drop('CLIMATE.REGION', axis=
```

In [15]:
```
outages_final.head()
```

Out[15]:

	CAUSE.CATEGORY	POPULATION	ANOMALY.LEVEL	HURRICANE.NAMES	REGION_Central	REGIO
0	severe weather	5348119.0	-0.3	NO	0	
1	intentional attack	5457125.0	-0.1	NO	0	
2	severe weather	5310903.0	-1.5	NO	0	
3	severe weather	5380443.0	-0.1	NO	0	
4	severe weather	5489594.0	1.2	NO	0	

In [16]:
```
X = outages_final.drop('CAUSE.CATEGORY', axis=1)
y = outages_final['CAUSE.CATEGORY']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, r
```

In [17]:
```
ct = ColumnTransformer([
    ('POP_std', StandardScaler(), ['POPULATION']),
    ('HURR_bin', OneHotEncoder(), ['HURRICANE.NAMES'])
], remainder='passthrough')

final_pl = Pipeline(steps=[
    ('preprocess', ct),
    ('dt', DecisionTreeClassifier())
])

params = {
    'dt__max_depth': [5,10,15,20,30,40,50,100]
}

searchCV = GridSearchCV(final_pl, params, cv=10)
searchCV.fit(X_train, y_train)
print('TRAINING ACCURACY: '+ str(searchCV.score(X_train, y_train)))
```

```
TRAINING ACCURACY: 0.8890977443609023
```

In [18]:
```
print('TESTING ACCURACY: ' + str(searchCV.score(X_test, y_test)))
```

```
TESTING ACCURACY: 0.6469298245614035
```

In [19]:
```
## Best param(s) for Decision Tree
searchCV.best_params_
```

Out[19]:
```
{'dt__max_depth': 15}
```

Fairness Evaluation

In [20]:
```
preds = searchCV.predict(X_test)
```

In [21]:
```
## Confusion Matrix
np.round(metrics.confusion_matrix(y_test, preds, labels=outages_final['CAUS
```

Out[21]:
```
array([[0.37, 0.05, 0.03, 0.01, 0.01, 0.  , 0.  ],
       [0.05, 0.21, 0.01, 0.  , 0.  , 0.01, 0.  ],
       [0.02, 0.02, 0.03, 0.01, 0.01, 0.  , 0.  ],
       [0.01, 0.  , 0.01, 0.01, 0.  , 0.  , 0.  ],
       [0.01, 0.  , 0.  , 0.  , 0.02, 0.  , 0.  ],
       [0.02, 0.01, 0.  , 0.  , 0.  , 0.  , 0.01],
       [0.  , 0.01, 0.  , 0.01, 0.  , 0.  , 0.  ]])
```

Parity Measure

NULL:

H0: My model is fair, the accuracy for my two subsets are about the same.

ALT:

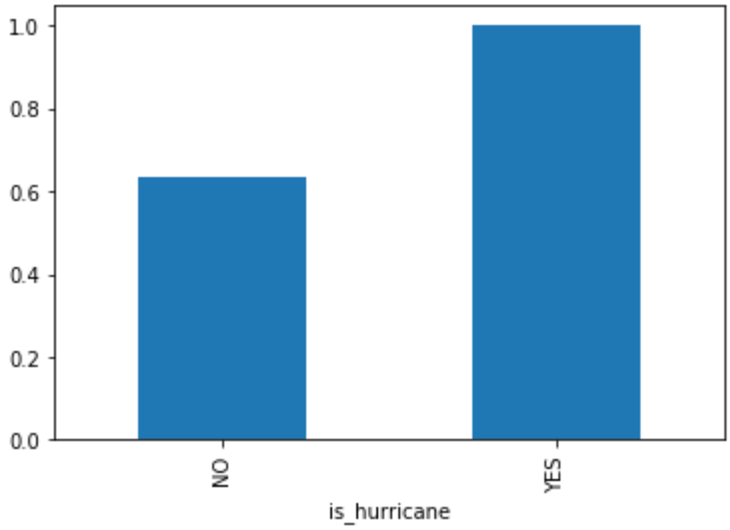
H1: My model is unfair; the accuracy for BLANK is higher than the BLANK.

In [22]:
```
results = X_test
results['predictions'] = preds
results['tag'] = y_test
results['is_hurricane'] = results['HURRICANE.NAMES']
```

VIsuallizing accuracy per hurricane subset

In [23]:
```
ac = metrics.accuracy_score(y_test, preds)
(
    results
    .groupby('is_hurricane')
    .apply(lambda x: metrics.accuracy_score(x.tag, x.predictions))
    .plot(kind='bar')
)
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x175033eee10>



Here outages that occured due to hurricane were predicted with an accuracy of 1.0 while those that occured not from a hurricane were predicted with an accuracy of 0.63.

In [24]:
```
results.groupby('is_hurricane').apply(lambda x: metrics.accuracy_score(x.ta
```

Out[24]:
```
is_hurricane
NO     0.633257
YES    1.000000
dtype: float64
```

Permutation Test

In [25]:

```
obs = results.groupby('is_hurricane').apply(lambda x: metrics.accuracy_scor

mets = []

for _ in range(1000):
    s = (
        results[['is_hurricane', 'predictions', 'tag']]
        .assign(is_hurricane=results.is_hurricane.sample(frac=1.0, replace=
        .groupby('is_hurricane')
        .apply(lambda x: metrics.accuracy_score(x.tag, x.predictions))
        .diff().abs()
        .iloc[-1]
    )
    mets.append(s)
print('OBSERVED VALUE: ' + str(obs))
```

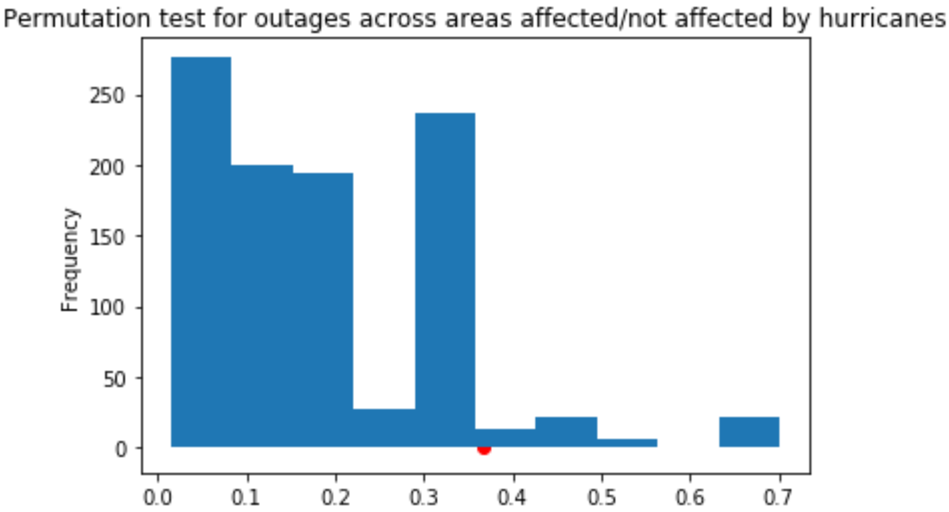
```
OBSERVED VALUE: 0.3667425968109339
```

In [26]:

```
print(pd.Series(mets>=obs).mean())
pd.Series(mets).plot(kind='hist', bins = 10,
                     title='Permutation test for outages across areas affec
plt.scatter(obs, 0.1, c='r')
```

```
0.05
```

Out[26]: `<matplotlib.collections.PathCollection at 0x17503020be0>`



Reject null hypothesis; No accuracy parity.

In [ ]:

In [ ]:

In [ ]: