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

Prediction problem chosen is to predict the severity of a major power outage using Regression.

- Target used: OUTAGE.DURATION
- Evaluation metric used is RMSE since we want to see if our prediction model improves if more features or enginnered features are added. I thought of using R-squared at the beginning but realised that it only increases as predictors are added to the regression model.

Baseline Model

For the baseline model I wanted to see if the outage duration could be predicted just by using GENERAL INFORMATIO such as geographic data, climate region information and the cause, without taking into consideration the REGIONAL ELECTRICITY CONSUMPTION INFORMATION, ECONOMIC CHARACTERISTICS or LAND-USE CHARACTERICS. 6 Features (all Nominal):

- MONTH
- POSTAL.CODE
- NERC.REGION
- CLIMATE.REGION
- ANOMALY.LEVEL
- CAUSE.CATEGORY

I was not expecting good results for this model and it was confirmed once I anlysed it. Out of 100 analyses comparing 50 different train and test sets, the distribution shows that only between 0 and 35% of the test_rmse where lower than the train_rmse. The r-squared scores for one example are not great either.

Final Model

For the engineered features I just changed the all the prices from cents/kilowwat-hour to cents/megawatt-hour to match the values for the energy comsumption. I also changed percentages to proportions.
I wanted to generate more complex features like combining 2 features but a this dataset already came with lots of combined features and I couldn't find a suitable one.

Didn't get to finish, I was left trying to select the features that would improve my baseline model.

Fairness Evaluation

TODO

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]:
```
# Read in the excel file and format it
outages = pd.read_excel('data/outage.xlsx', header=5, usecols='c:BE').drop(0, axis=0).reset_index(drop=True)
```

In [3]:
```
pd.set_option('display.max_columns', None)
```

In [4]:
```
outages.head()
```

Out[4]:

	YEAR	MONTH	U.S._STATE	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMA
0	2011.0	7.0	Minnesota	MN	MRO	East North Central	
1	2014.0	5.0	Minnesota	MN	MRO	East North Central	
2	2010.0	10.0	Minnesota	MN	MRO	East North Central	
3	2012.0	6.0	Minnesota	MN	MRO	East North Central	
4	2015.0	7.0	Minnesota	MN	MRO	East North Central	

In [5]:
```
def RMSE(predictions, actual):
    return np.sqrt(np.mean((predictions - actual)**2))
```

Baseline Model

In [6]:
```
# getting the columns for baseline model
base = outages[['MONTH', 'POSTAL.CODE', 'NERC.REGION', 'CLIMATE.REG
ION', 'ANOMALY.LEVEL', 'CAUSE.CATEGORY', 'OUTAGE.DURATION']]
base
```

Out[6]:

	MONTH	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL	CAUSE
0	7.0	MN	MRO	East North Central	-0.3	se
1	5.0	MN	MRO	East North Central	-0.1	inter
2	10.0	MN	MRO	East North Central	-1.5	se
3	6.0	MN	MRO	East North Central	-0.1	se
4	7.0	MN	MRO	East North Central	1.2	se
...	...	...	...	...	...	
1529	12.0	ND	MRO	West North Central	-0.9	p
1530	NaN	ND	MRO	West North Central	NaN	
1531	8.0	SD	RFC	West North Central	0.5	
1532	8.0	SD	MRO	West North Central	0.5	
1533	NaN	AK	ASCC	NaN	NaN	equi

1534 rows × 7 columns

In [7]:
```
# checking for null values
base.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1534 entries, 0 to 1533
Data columns (total 7 columns):
MONTH              1525 non-null float64
POSTAL.CODE        1534 non-null object
NERC.REGION        1534 non-null object
CLIMATE.REGION     1528 non-null object
ANOMALY.LEVEL      1525 non-null object
CAUSE.CATEGORY     1534 non-null object
OUTAGE.DURATION    1476 non-null object
dtypes: float64(1), object(6)
memory usage: 84.0+ KB
```

In [8]:
```
# remove rows where OUTAGE.DURATION is null since there are only 58
rows out of 1534
base = base.dropna(subset=['OUTAGE.DURATION'])
# fill the null climate regions with Outside (continental U.S.A)
base['CLIMATE.REGION'] = base[['CLIMATE.REGION']].fillna('Outside')
```

```
/Users/erlin/anaconda3/lib/python3.7/site-packages/ipykernel_launc
her.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  after removing the cwd from sys.path.
```

In [9]:
```
base.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1476 entries, 0 to 1532
Data columns (total 7 columns):
MONTH              1476 non-null float64
POSTAL.CODE        1476 non-null object
NERC.REGION        1476 non-null object
CLIMATE.REGION     1476 non-null object
ANOMALY.LEVEL      1476 non-null object
CAUSE.CATEGORY     1476 non-null object
OUTAGE.DURATION    1476 non-null object
dtypes: float64(1), object(6)
memory usage: 92.2+ KB
```

In [10]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [11]:
```
# pipeline to onehot encode columns and perform linear regression
pl = Pipeline([
        ('one-hot', OneHotEncoder(handle_unknown = 'ignore')),
        ('lin-reg', LinearRegression())
    ])
```

In [12]:
```
# get X and y
X = base.drop('OUTAGE.DURATION', 1)
y = base['OUTAGE.DURATION']
```

In [13]:
```
# split data into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

In [14]:
```
# fit the model and compare scores
pl.fit(X_train, y_train)
pl.score(X_train, y_train), pl.score(X_test, y_test)
```

Out[14]:
```
(0.3415442428663398, 0.07017470375519275)
```

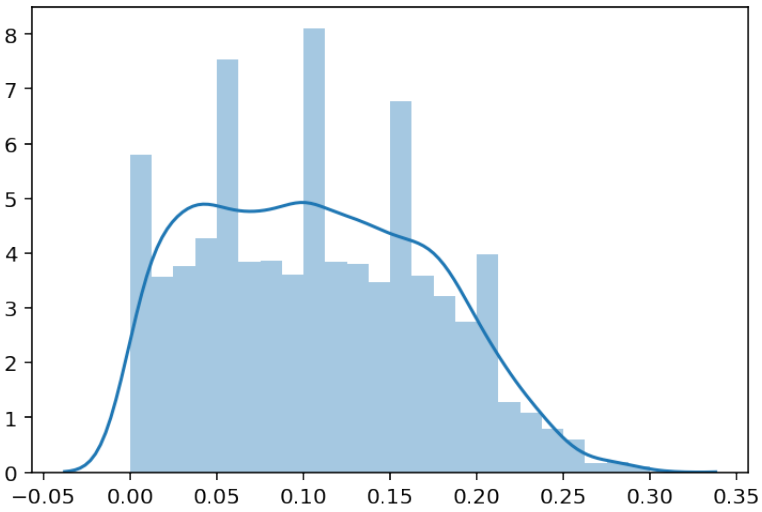
In [15]:
```
train_rmse = RMSE(pl.predict(X_train), y_train)
test_rmse = RMSE(pl.predict(X_test), y_test)
train_rmse, test_rmse
```

Out[15]:
```
(4149.407572415524, 7636.054301367026)
```

In [16]:
```
# getting couple more scores for comparison
x = []

for i in range(100):
    result = {
        'train_rmse':[],
        'test_rmse':[]
    }

    for _ in range(50):
        X_train, X_test, y_train, y_test = train_test_split(X, y)
        pl.fit(X_train, y_train)

        train_rmse = RMSE(pl.predict(X_train), y_train)
        test_rmse = RMSE(pl.predict(X_test), y_test)

        result['train_rmse'].append(train_rmse)
        result['test_rmse'].append(test_rmse)

        percent = (np.array(result['test_rmse']) < np.array(result[
'train_rmse'])).sum() / 100

        x.append(percent)
```

In [17]:
```
sns.distplot(x)
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x1a24b629d0>



Final Model

In [163]:
```
cols = outages.columns[:45]
```

In [164]:
```
drop = list(cols[36:44]) + [cols[2]] + list(cols[7:12]) + list(cols
[13:15]) + list(cols[16:18])
```

In [206]:
```
final = outages[cols].drop(drop, 1)
final
```

Out[206]:

	YEAR	MONTH	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL
0	2011.0	7.0	MN	MRO	East North Central	-0.3
1	2014.0	5.0	MN	MRO	East North Central	-0.1
2	2010.0	10.0	MN	MRO	East North Central	-1.5
3	2012.0	6.0	MN	MRO	East North Central	-0.1
4	2015.0	7.0	MN	MRO	East North Central	1.2
...	...	...	...	...	...	...
1529	2011.0	12.0	ND	MRO	West North Central	-0.9
1530	2006.0	NaN	ND	MRO	West North Central	NaN
1531	2009.0	8.0	SD	RFC	West North Central	0.5
1532	2009.0	8.0	SD	MRO	West North Central	0.5
1533	2000.0	NaN	AK	ASCC	NaN	NaN

1534 rows × 27 columns

In [207]:
```
# drop rows
final = final.dropna(subset=['OUTAGE.DURATION', 'RES.PRICE'])
# fill the null climate regions with Outside (continental U.S.A)
final['CLIMATE.REGION'] = final[['CLIMATE.REGION']].fillna('Outside
')
```

```
/Users/erlin/anaconda3/lib/python3.7/site-packages/ipykernel_launc
her.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  after removing the cwd from sys.path.
```

In [208]:
```
# make sure no null values
final.isnull().sum().all() == 0
```

Out[208]:
```
True
```

In [209]:
```
# getting column names where the numerical values are type object
columns = ['ANOMALY.LEVEL'] + list(final.columns[7:])
```

In [210]:
```
# changing the type of the previously fetched columns to float type values
for col in columns:
    final[col] =  final[col].astype(float)
```

```
/Users/erlin/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Engineered Features

In [211]:
```
final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1464 entries, 0 to 1532
Data columns (total 27 columns):
YEAR                1464 non-null float64
MONTH               1464 non-null float64
POSTAL.CODE         1464 non-null object
NERC.REGION         1464 non-null object
CLIMATE.REGION      1464 non-null object
ANOMALY.LEVEL       1464 non-null float64
CAUSE.CATEGORY      1464 non-null object
OUTAGE.DURATION     1464 non-null float64
RES.PRICE           1464 non-null float64
COM.PRICE           1464 non-null float64
IND.PRICE           1464 non-null float64
TOTAL.PRICE         1464 non-null float64
RES.SALES           1464 non-null float64
COM.SALES           1464 non-null float64
IND.SALES           1464 non-null float64
TOTAL.SALES         1464 non-null float64
RES.PERCEN          1464 non-null float64
COM.PERCEN          1464 non-null float64
IND.PERCEN          1464 non-null float64
RES.CUSTOMERS       1464 non-null float64
COM.CUSTOMERS       1464 non-null float64
IND.CUSTOMERS       1464 non-null float64
TOTAL.CUSTOMERS     1464 non-null float64
RES.CUST.PCT        1464 non-null float64
COM.CUST.PCT        1464 non-null float64
IND.CUST.PCT        1464 non-null float64
POPULATION          1464 non-null float64
dtypes: float64(23), object(4)
memory usage: 320.2+ KB
```

In [212]:
```
# changed electricity prices to cents per megawatt per hour to matc
h electricity consumption data
prices = [x for x in final.columns if 'PRICE' in x]
for price in prices:
    final[price] = final[price]*1000
```

```
/Users/erlin/anaconda3/lib/python3.7/site-packages/ipykernel_launc
her.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  after removing the cwd from sys.path.
```

In [213]:
```
# change percentages to proportions
percentages = [x for x in final.columns if ('PERCEN' in x ) or ('PCT' in x)]
for percent in percentages:
    final[percent] = final[percent]/100
```

```
/Users/erlin/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

In [214]:
```
# one hot encoder
one_hot = ['MONTH', 'POSTAL.CODE', 'NERC.REGION', 'CLIMATE.REGION', 'ANOMALY.LEVEL', 'CAUSE.CATEGORY']
one_hot_encoder = Pipeline(steps=[
    ('one-hot', OneHotEncoder(handle_unknown = 'ignore'))
])
```

In [215]:
```
preproc = ColumnTransformer(transformers=[('one-hot', one_hot_encoder, one_hot)], remainder='passthrough')
```

In [216]:
```
final.head()
```

Out[216]:

	YEAR	MONTH	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL	CA
0	2011.0	7.0	MN	MRO	East North Central	-0.3	
1	2014.0	5.0	MN	MRO	East North Central	-0.1	
2	2010.0	10.0	MN	MRO	East North Central	-1.5	
3	2012.0	6.0	MN	MRO	East North Central	-0.1	
4	2015.0	7.0	MN	MRO	East North Central	1.2	

In [ ]:

In [ ]:

In [188]:
```
sales = [x for x in final.columns if 'SALES' in x]
customers = [x for x in final.columns if 'CUSTOMERS' in x]
```

In [140]:
```
final = final.drop(prices, 1)
```

In [141]:
```
final = final.drop(sales, 1)
```

In [142]:
```
final = final.drop(customers, 1)
```

In [253]:
```
final2 = final[list(base.columns) + [x for x in final.columns if 'P
ERCEN' in x]]
```

In [254]:
```
# get X and y for final model
X = final2.drop('OUTAGE.DURATION', 1)
y = final2['OUTAGE.DURATION']
```

In [255]:
```
pl = Pipeline(steps=[('preprocessor', preproc), ('regressor', Linea
rRegression())])
```

In [261]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

In [262]:
```
X_train
```

Out[262]:

	MONTH	POSTAL.CODE	NERC.REGION	CLIMATE.REGION	ANOMALY.LEVEL	CAUSE
993	8.0	LA	SERC	South	-1.2	p
611	9.0	PA	RFC	Northeast	-0.3	se
24	1.0	TN	SERC	Central	-0.5	se
964	5.0	NY	NPCC	Northeast	0.6	
1409	12.0	OK	SPP	South	2.3	se
...	...	...	...	...	...	
1135	3.0	CA	WECC	West	0.6	inter
1523	1.0	ID	WECC	Northwest	-1.3	inter
1337	12.0	VA	SERC	Southeast	1.1	se
1430	12.0	OK	SPP	South	2.3	se
1019	5.0	LA	SERC	South	-0.7	syste

1098 rows × 9 columns

In [263]:
```
pl.fit(X_train, y_train)
pl.score(X_train, y_train), pl.score(X_test, y_test)
```

Out[263]:
```
(0.25944102771604227, 0.1675276958604276)
```

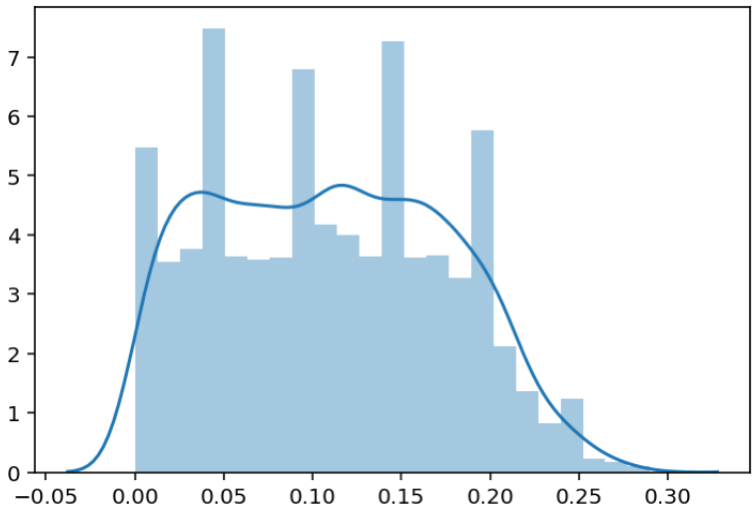
In [264]:
```
train_rmse = RMSE(pl.predict(X_train), y_train)
test_rmse = RMSE(pl.predict(X_test), y_test)
train_rmse, test_rmse
```

Out[264]:
```
(4485.867210729153, 7101.168879547022)
```

In [265]:
```
# getting couple more scores for comparison
x = []

for i in range(100):
    result = {
        'train_rmse':[],
        'test_rmse':[]
    }

    for _ in range(50):
        X_train, X_test, y_train, y_test = train_test_split(X, y)
        pl.fit(X_train, y_train)

        train_rmse = RMSE(pl.predict(X_train), y_train)
        test_rmse = RMSE(pl.predict(X_test), y_test)

        result['train_rmse'].append(train_rmse)
        result['test_rmse'].append(test_rmse)

        percent = (np.array(result['test_rmse']) < np.array(result[
'train_rmse'])).sum() / 100

        x.append(percent)
```

In [266]:
```
sns.distplot(x)
```

Out[266]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a25c58f10>
```



In [267]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [303]:
```
pl = Pipeline(steps=[('preprocessor', preproc), ('regressor', Decis
ionTreeRegressor(max_depth=3))])
```

In [304]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y)
pl.fit(X_train, y_train)
pl.score(X_train, y_train), pl.score(X_test, y_test)
```

Out[304]:
```
(0.4379802488132849, 0.21117761598429385)
```

In [305]:
```
train_rmse = RMSE(pl.predict(X_train), y_train)
test_rmse = RMSE(pl.predict(X_test), y_test)
train_rmse, test_rmse
```

Out[305]:
```
(4812.191620323374, 3785.204777208536)
```

In [199]:
```
from sklearn.model_selection import GridSearchCV
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Fairness Evaluation

In [30]:
```
# TODO
```