What are the important characteristics of major power outages in the United States?

Name(s): Ripudh Mylapur, Chia Lee

Website Link: https://ripudhm.github.io/power-outages-analysis/

Code

In [185…

```
import pandas as pd
import numpy as np
import os

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

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import QuantileTransformer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import metrics
```

Framing the Problem

In [31]:

```
import openpyxl
df = pd.read_excel(r"outage.xlsx", index_col = 1, header = 5)
df = df.drop(np.nan)
df = df.drop('variables', axis = 1)

def join_times(col_date, col_time):

    time = df[[col_date, col_time]]
    time = time.assign(date=pd.to_datetime(time[col_date]))
    time['date'] = time['date'].astype(str)
    time = time.assign(datetime=time['date'] + ' ' + time[col_time].astype(str))
    time = time.replace('NaT nan', np.nan)
    time = time.assign(fin=pd.to_datetime(time['datetime']))
    return time['fin']

out_start = join_times('OUTAGE.START.DATE', 'OUTAGE.START.TIME')
rest = join_times('OUTAGE.RESTORATION.DATE', 'OUTAGE.RESTORATION.TIME')
df = df.assign(out_start = out_start)
df = df.assign(rest_start = rest)
df = df.rename(columns = {'out_start': 'OUTAGE.START', 'rest_start': 'OUTAGE.RESTORATI
```

In [32]:
```
df.columns
```

Out[32]:
```
Index(['YEAR', 'MONTH', 'U.S._STATE', 'POSTAL.CODE', 'NERC.REGION',
       'CLIMATE.REGION', 'ANOMALY.LEVEL', 'CLIMATE.CATEGORY',
       'OUTAGE.START.DATE', 'OUTAGE.START.TIME', 'OUTAGE.RESTORATION.DATE',
       'OUTAGE.RESTORATION.TIME', 'CAUSE.CATEGORY', 'CAUSE.CATEGORY.DETAIL',
       'HURRICANE.NAMES', 'OUTAGE.DURATION', 'DEMAND.LOSS.MW',
       'CUSTOMERS.AFFECTED', 'RES.PRICE', 'COM.PRICE', 'IND.PRICE',
       'TOTAL.PRICE', 'RES.SALES', 'COM.SALES', 'IND.SALES', 'TOTAL.SALES',
       'RES.PERCEN', 'COM.PERCEN', 'IND.PERCEN', 'RES.CUSTOMERS',
       'COM.CUSTOMERS', 'IND.CUSTOMERS', 'TOTAL.CUSTOMERS', 'RES.CUST.PCT',
       'COM.CUST.PCT', 'IND.CUST.PCT', 'PC.REALGSP.STATE', 'PC.REALGSP.USA',
       'PC.REALGSP.REL', 'PC.REALGSP.CHANGE', 'UTIL.REALGSP', 'TOTAL.REALGSP',
       'UTIL.CONTRI', 'PI.UTIL.OFUSA', 'POPULATION', 'POPPCT_URBAN',
       'POPPCT_UC', 'POPDEN_URBAN', 'POPDEN_UC', 'POPDEN_RURAL',
       'AREAPCT_URBAN', 'AREAPCT_UC', 'PCT_LAND', 'PCT_WATER_TOT',
       'PCT_WATER_INLAND', 'OUTAGE.START', 'OUTAGE.RESTORATION'],
      dtype='object')
```

Baseline Model

In [159…
```
unique_regions = df['CLIMATE.REGION'].value_counts()
for reg in unique_regions.index:
    df[reg] = df['CLIMATE.REGION'].apply(lambda x: 1 if x == reg else 0)
```

In [160…
```
df['CAUSE.CATEGORY']
```

Out[160]:
```
OBS
1.0              severe weather
2.0          intentional attack
3.0              severe weather
4.0              severe weather
5.0              severe weather
                     ...
1530.0            public appeal
1531.0    fuel supply emergency
1532.0                islanding
1533.0                islanding
1534.0        equipment failure
Name: CAUSE.CATEGORY, Length: 1534, dtype: object
```

In [161…
```
df1 = df[['OUTAGE.DURATION', 'CUSTOMERS.AFFECTED','ANOMALY.LEVEL', 'MONTH','NERC.REGIC
```

In [162…
```
df1 = df1.dropna()
df1
```

Out[162]:

OBS	OUTAGE.DURATION	CUSTOMERS.AFFECTED	ANOMALY.LEVEL	MONTH	NERC.REGION	CAUSE
1.0	3060	70000.0	-0.3	7.0	MRO	sev
3.0	3000	70000.0	-1.5	10.0	MRO	sev
4.0	2550	68200.0	-0.1	6.0	MRO	sev
5.0	1740	250000.0	1.2	7.0	MRO	sev
6.0	1860	60000.0	-1.4	11.0	MRO	sev
...	...	...	...	...	...	
1523.0	95	35000.0	0.3	6.0	WECC	systen
1524.0	360	0.0	-1.3	1.0	WECC	inten
1525.0	1548	0.0	-0.1	6.0	WECC	p
1527.0	0	0.0	1.6	3.0	WECC	inten
1530.0	720	34500.0	-0.9	12.0	MRO	p

1056 rows × 8 columns

In [163…
```
model = LinearRegression()
y = df1[['OUTAGE.DURATION']]
X = df1[['CUSTOMERS.AFFECTED']]
model.fit(X = X, y = y)
```

Out[163]:
```
LinearRegression()
```

In [164…
```
preproc = ColumnTransformer(
    transformers=[
    ('ohe', OneHotEncoder(), ['MONTH','NERC.REGION','CAUSE.CATEGORY', 'CLIMATE.CATEGOR
#     ('standardise', StandardScaler(), ['CUSTOMERS.AFFECTED', 'ANOMALY.LEVEL'])
    ],
    remainder='passthrough' # Specify what to do with all other columns ('total_bill'
)
```

In [165…
```
pl_base = Pipeline([
    ('preproc', preproc),
    ('lin-reg', LinearRegression())
    ])
```

In [166…
```
pl_base.fit(df1.drop(['OUTAGE.DURATION'], axis=1), df1['OUTAGE.DURATION'])
```

Out[166]:
```
Pipeline(steps=[('preproc',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe', OneHotEncoder(),
                                                  ['MONTH', 'NERC.REGION',
                                                   'CAUSE.CATEGORY',
                                                   'CLIMATE.CATEGORY'])])),
                ('lin-reg', LinearRegression())])
```

In [167…
```
pl_base.score(df1.drop(['OUTAGE.DURATION'], axis=1), df1['OUTAGE.DURATION'])
```

Out[167]:
```
0.06895656721070198
```

In [168…
```
def rmse(actual, pred):
    return np.sqrt(np.mean((actual - pred) ** 2))
```

In [169…
```
all_preds = model.predict(X)
rmse(df1['OUTAGE.DURATION'], all_preds.reshape(-1))
```

Out[169]:
```
4282.432892427123
```

In [170…
```
all_preds.reshape(-1)
```

Out[170]:
```
array([2546.39007736, 2546.39007736, 2539.12560128, ..., 2263.88267403,
       2263.88267403, 2403.11846567])
```

Final Model

In [364…
```
X_train, X_test, y_train, y_test = train_test_split(df1.drop(['OUTAGE.DURATION'], axis
```

In [365…
```
preproc = ColumnTransformer(
    transformers=[
    ('ohe', OneHotEncoder(handle_unknown='ignore'), ['MONTH','NERC.REGION','CAUSE.CATE
    ('standardise', StandardScaler(), ['CUSTOMERS.AFFECTED']),
    ('quant', QuantileTransformer(n_quantiles=10), ['POPULATION'])
#     ('cvector', TfidfVectorizer(), ['CAUSE.CATEGORY.DETAIL'])
    ],
    remainder='passthrough' # Specify what to do with all other columns ('total_bill'
)
```

In [366…
```
pl1 = Pipeline([
    ('preproc', preproc),
    ('lin-reg', LinearRegression())
    ])
```

In [367…
```
pl1.fit(X_train, y_train)
```

Out[367]:
```
Pipeline(steps=[('preproc',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe',
                                                  OneHotEncoder(handle_unknown='ignor
e'),
                                                  ['MONTH', 'NERC.REGION',
                                                   'CAUSE.CATEGORY',
                                                   'CLIMATE.CATEGORY']),
                                                 ('standardise',
                                                  StandardScaler(),
                                                  ['CUSTOMERS.AFFECTED']),
                                                 ('quant',
                                                  QuantileTransformer(n_quantiles=1
0),
                                                  ['POPULATION']),])),
                ('lin-reg', LinearRegression())])
```

In [368…
```
pl1.score(X_train, y_train)
```

Out[368]:
```
0.25251869364755575
```

In [369…
```
pl1.score(X_test, y_test)
```

Out[369]:
```
0.36519748973673094
```

In [370…
```
polyreg = Pipeline([
    ('preproc', preproc),
    ('poly', PolynomialFeatures(1)),
    ('lin-reg', LinearRegression())
    ]
)
```

In [371…
```
polyreg.fit(X_train, y_train)
```

Out[371]:
```
Pipeline(steps=[('preproc',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ohe',
                                                  OneHotEncoder(handle_unknown='ignor
e'),
                                                  ['MONTH', 'NERC.REGION',
                                                   'CAUSE.CATEGORY',
                                                   'CLIMATE.CATEGORY']),
                                                 ('standardise',
                                                  StandardScaler(),
                                                  ['CUSTOMERS.AFFECTED']),
                                                 ('quant',
                                                  QuantileTransformer(n_quantiles=1
0),
                                                  ['POPULATION'])])),
                ('poly', PolynomialFeatures(degree=1)),
                ('lin-reg', LinearRegression())])
```

In [372…
```
polyreg.score(X_train, y_train)
```

Out[372]:
```
0.25251869364755486
```

In [373…
```
polyreg.score(X_test, y_test)
```

Out[373]:
```
0.3651974873113507
```

In [374…
```
def polynomialreg(degree=2):
    return Pipeline([
    ('preproc', preproc),
    ('poly', PolynomialFeatures(degree)),
    ('lin-reg', LinearRegression())
    ]
)
```

In [375…
```
hyperparameters = {'poly__degree': [1,2,3]}
```

In [376…
```
searcher = GridSearchCV(polyreg, hyperparameters, cv=5)
```

In [377…
```
searcher.fit(X_train, y_train)
```

Out[377]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('preproc',
                                        ColumnTransformer(remainder='passthrough',
                                                          transformers=[('ohe',
                                                                         OneHotEncode
r(handle_unknown='ignore'),
                                                                         ['MONTH',
                                                                          'NERC.REGIO
N',
                                                                          'CAUSE.CATE
GORY',
                                                                          'CLIMATE.CA
TEGORY']),
                                                                        ('standardis
e',
                                                                         StandardScal
er(),
                                                                         ['CUSTOMERS.
AFFECTED']),
                                                                        ('quant',
                                                                         QuantileTran
sformer(n_quantiles=10),
                                                                         ['POPULATIO
N'])])),
                                       ('poly', PolynomialFeatures(degree=1)),
                                       ('lin-reg', LinearRegression())]),
             param_grid={'poly__degree': [1, 2, 3]})
```

In [378…
```
searcher.best_params_
```

Out[378]:
```
{'poly__degree': 1}
```

Fairness Analysis

In [379…
```
df2 = df[['OUTAGE.DURATION', 'CUSTOMERS.AFFECTED','ANOMALY.LEVEL', 'MONTH','NERC.REGIO
df2['is_Cali'] = df2['U.S._STATE'].apply(lambda x: 1 if x=='California' else 0)
df2= df2.dropna()
df2['is_Cali'].sum()
```

```
C:\Users\ripud\AppData\Local\Temp\ipykernel_17148\243506506.py:2: SettingWithCopyWarn
ing:


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: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[379]:
```
121
```

In [380…
```
df2
```

Out[380]:

OBS	OUTAGE.DURATION	CUSTOMERS.AFFECTED	ANOMALY.LEVEL	MONTH	NERC.REGION	CAUSE
1.0	3060	70000.0	-0.3	7.0	MRO	sev
3.0	3000	70000.0	-1.5	10.0	MRO	sev
4.0	2550	68200.0	-0.1	6.0	MRO	sev
5.0	1740	250000.0	1.2	7.0	MRO	sev
6.0	1860	60000.0	-1.4	11.0	MRO	sev
...	...	...	...	...	...	
1523.0	95	35000.0	0.3	6.0	WECC	systen
1524.0	360	0.0	-1.3	1.0	WECC	inten
1525.0	1548	0.0	-0.1	6.0	WECC	p
1527.0	0	0.0	1.6	3.0	WECC	inten
1530.0	720	34500.0	-0.9	12.0	MRO	p

1056 rows × 10 columns

In [381…
```
y_pred = pl1.predict(df2)
```

In [382…
```
df2['prediction'] = y_pred
```

In [383…
```
df2
```

Out[383]:

OBS	OUTAGE.DURATION	CUSTOMERS.AFFECTED	ANOMALY.LEVEL	MONTH	NERC.REGION	CAUSE
1.0	3060	70000.0	-0.3	7.0	MRO	sev
3.0	3000	70000.0	-1.5	10.0	MRO	sev
4.0	2550	68200.0	-0.1	6.0	MRO	sev
5.0	1740	250000.0	1.2	7.0	MRO	sev
6.0	1860	60000.0	-1.4	11.0	MRO	sev
...	...	...	...	...	...	
1523.0	95	35000.0	0.3	6.0	WECC	systen
1524.0	360	0.0	-1.3	1.0	WECC	inten
1525.0	1548	0.0	-0.1	6.0	WECC	p
1527.0	0	0.0	1.6	3.0	WECC	inten
1530.0	720	34500.0	-0.9	12.0	MRO	p

1056 rows × 11 columns

In [384…
```
def rmse(actual, pred):
    return np.sqrt(np.mean((actual - pred) ** 2))
```

In [385…
```
df2['act-pred^2'] = (df2['OUTAGE.DURATION'] - df2['prediction']) ** 2
df2
```

Out[385]:

OBS	OUTAGE.DURATION	CUSTOMERS.AFFECTED	ANOMALY.LEVEL	MONTH	NERC.REGION	CAUSE
1.0	3060	70000.0	-0.3	7.0	MRO	sev
3.0	3000	70000.0	-1.5	10.0	MRO	sev
4.0	2550	68200.0	-0.1	6.0	MRO	sev
5.0	1740	250000.0	1.2	7.0	MRO	sev
6.0	1860	60000.0	-1.4	11.0	MRO	sev
...	...	...	...	...	...	
1523.0	95	35000.0	0.3	6.0	WECC	systen
1524.0	360	0.0	-1.3	1.0	WECC	inten
1525.0	1548	0.0	-0.1	6.0	WECC	p
1527.0	0	0.0	1.6	3.0	WECC	inten
1530.0	720	34500.0	-0.9	12.0	MRO	p

1056 rows × 12 columns

In [386…
```
obs = np.sqrt(df2.groupby('is_Cali')['act-pred^2'].mean()).diff().iloc[-1]
obs
```

Out[386]:
```
1235.1517809923307
```

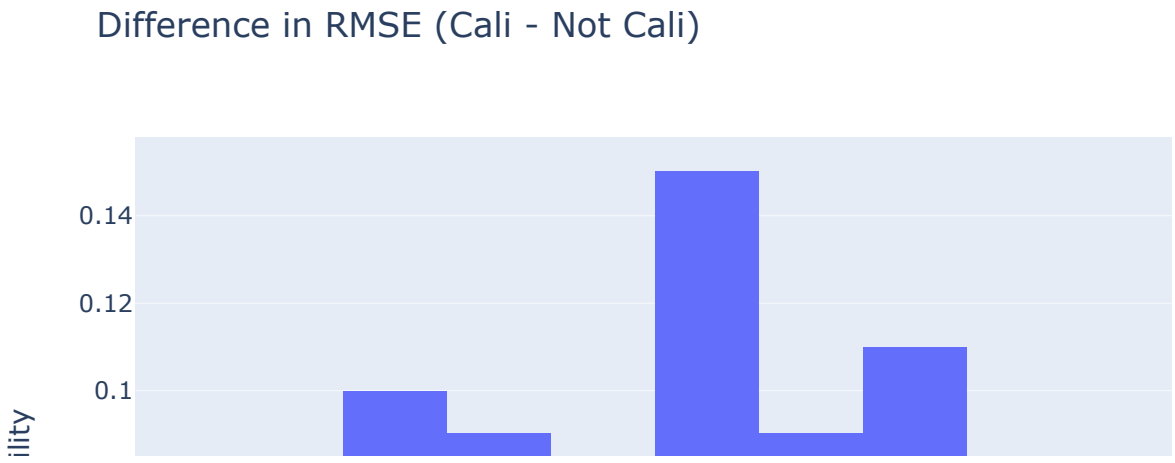
In [387…
```
diff_in_rmse = []
for _ in range(100):
    s = (
        np.sqrt(df2[['is_Cali', 'prediction', 'OUTAGE.DURATION', 'act-pred^2']]
        .assign(is_Cali=df2.is_Cali.sample(frac=1.0, replace=False).reset_index(drop=T
        .groupby('is_Cali')['act-pred^2']
        .mean())
        .diff()
        .iloc[-1]
    )

    diff_in_rmse.append(s)
```

In [388…
```
np_diff_rmse = np.array(diff_in_rmse)
(obs < np_diff_rmse).sum() / len(diff_in_rmse)
```

Out[388]:
```
0.09
```

In [390…
```
fig = pd.Series(diff_in_rmse).plot(kind='hist', histnorm='probability', nbins=20,
                           title='Difference in RMSE (Cali - Not Cali)')
fig.add_vline(x=obs, line_color='red')
fig.write_html('diff_rmse.html', include_plotlyjs='cdn')
fig
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: