Power Outages

Name(s): Karsin Dass & Cole Doyle

Website Link: https://keemarice.github.io/PowerOutages/

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler, LabelEncoder, FunctionTran
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import plotly.graph_objects as go
from sklearn.tree import DecisionTreeRegressor
import plotly.express as px
from sklearn.linear_model import Ridge
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
import mpld3

pd.options.plotting.backend = 'plotly'

# from lec_utils import * # Feel free to uncomment and use this. It'll make
```

Introduction

In [2]:
```
outageFull = pd.read_csv('outage.csv', usecols = list(range(2, 56)), header
print(outageFull.head())
print(outageFull.columns)
#print nmber of columns
print(outageFull.columns.size)
```

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

  ANOMALY.LEVEL CLIMATE.CATEGORY                 OUTAGE.START.DATE  \
0       numeric              NaN  Day of the week, Month Day, Year   
1          -0.3           normal               Friday, July 1, 2011   
2          -0.1           normal               Sunday, May 11, 2014   
3          -1.5             cold         Tuesday, October 26, 2010   
4          -0.1           normal            Tuesday, June 19, 2012   

              OUTAGE.START.TIME  ... POPULATION POPPCT_URBAN POPPCT_UC  \
0  Hour:Minute:Second (AM / PM)  ...        NaN            %         %   
1                    5:00:00 PM  ...  5348119.0        73.27     15.28   
2                    6:38:00 PM  ...  5457125.0        73.27     15.28   
3                    8:00:00 PM  ...  5310903.0        73.27     15.28   
4                    4:30:00 AM  ...  5380443.0        73.27     15.28   

              POPDEN_URBAN                POPDEN_UC             POPDEN_RURAL  
\
0  persons per square mile  persons per square mile  persons per square mile   
1                     2279                   1700.5                     18.2   
2                     2279                   1700.5                     18.2   
3                     2279                   1700.5                     18.2   
4                     2279                   1700.5                     18.2   

  AREAPCT_URBAN  AREAPCT_UC     PCT_LAND PCT_WATER_TOT  
0             %           %            %             %  
1          2.14         0.6  91.59266587   8.407334131  
2          2.14         0.6  91.59266587   8.407334131  
3          2.14         0.6  91.59266587   8.407334131  
4          2.14         0.6  91.59266587   8.407334131  

[5 rows x 54 columns]
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.REALGS
P',
       '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'],
      dtype='object')
54
```

Data Cleaning and Exploratory Data Analysis

Cleaning

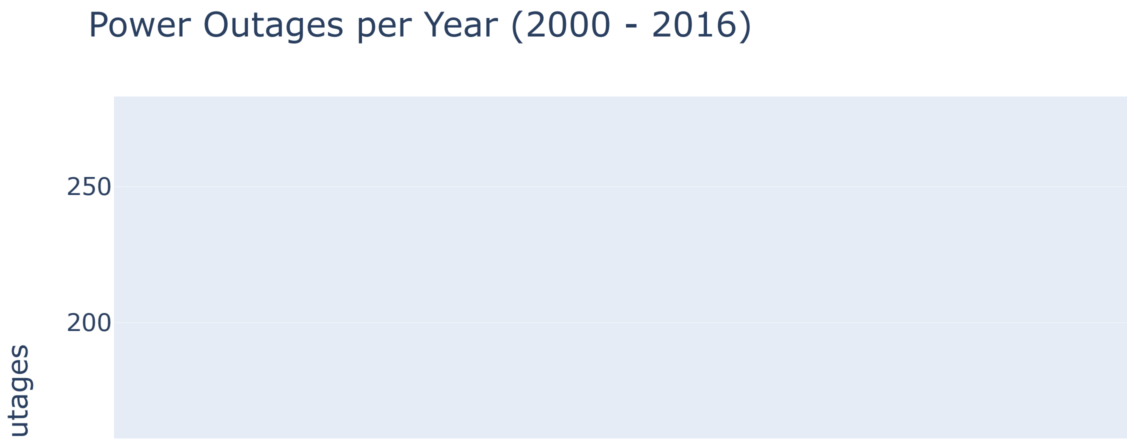
In [3]:
```
outageClean = outageFull[["YEAR", "U.S._STATE", "POSTAL.CODE", "NERC.REGION"
outageClean = outageClean.iloc[1:]
outageClean[['YEAR', 'OUTAGE.DURATION']].dropna()
outageClean['YEAR'] = pd.to_numeric(outageClean['YEAR'])
outageClean['OUTAGE.DURATION'] = pd.to_numeric(outageClean['OUTAGE.DURATION'
outageClean['DEMAND.LOSS.MW'] = pd.to_numeric(outageClean['DEMAND.LOSS.MW'])
outageClean['CUSTOMERS.AFFECTED'] = pd.to_numeric(outageClean['CUSTOMERS.AFF

outageClean.head()
```

Out[3]:

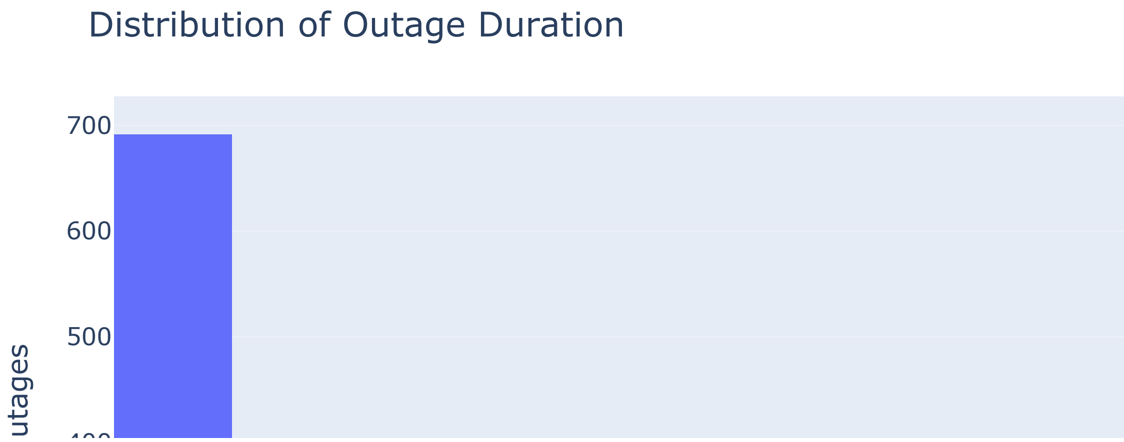
	YEAR	U.S._STATE	POSTAL.CODE	NERC.REGION	CAUSE.CATEGORY	OUTAGE.DUI
1	2011.0	Minnesota	MN	MRO	severe weather	
2	2014.0	Minnesota	MN	MRO	intentional attack	
3	2010.0	Minnesota	MN	MRO	severe weather	
4	2012.0	Minnesota	MN	MRO	severe weather	
5	2015.0	Minnesota	MN	MRO	severe weather	

Univariate Analysis

In [4]:
```
fig = px.histogram(outageClean, x = 'YEAR', nbins = 17)
fig.update_layout(title = "Power Outages per Year (2000 - 2016)", xaxis_titl
fig.update_layout(
   xaxis = dict(
      tickmode = 'linear',
      tick0 = 2000,
      dtick = 1
   )
)
fig.show()
```

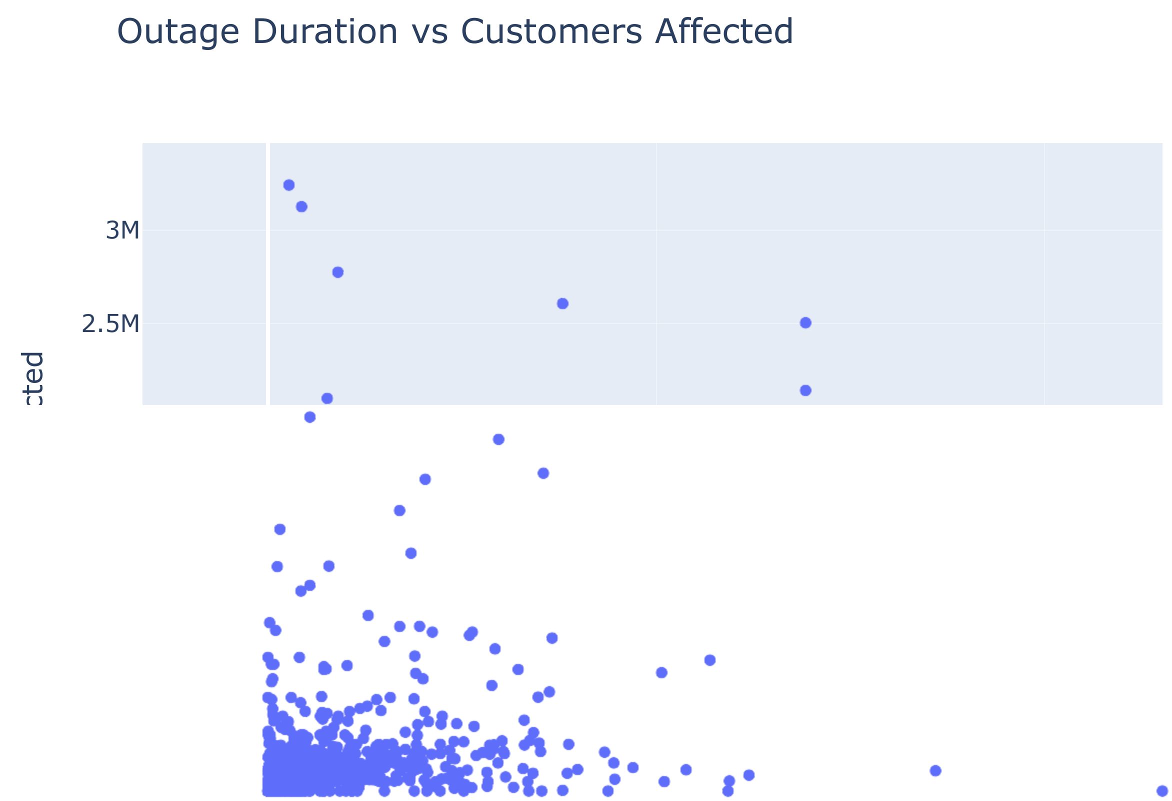


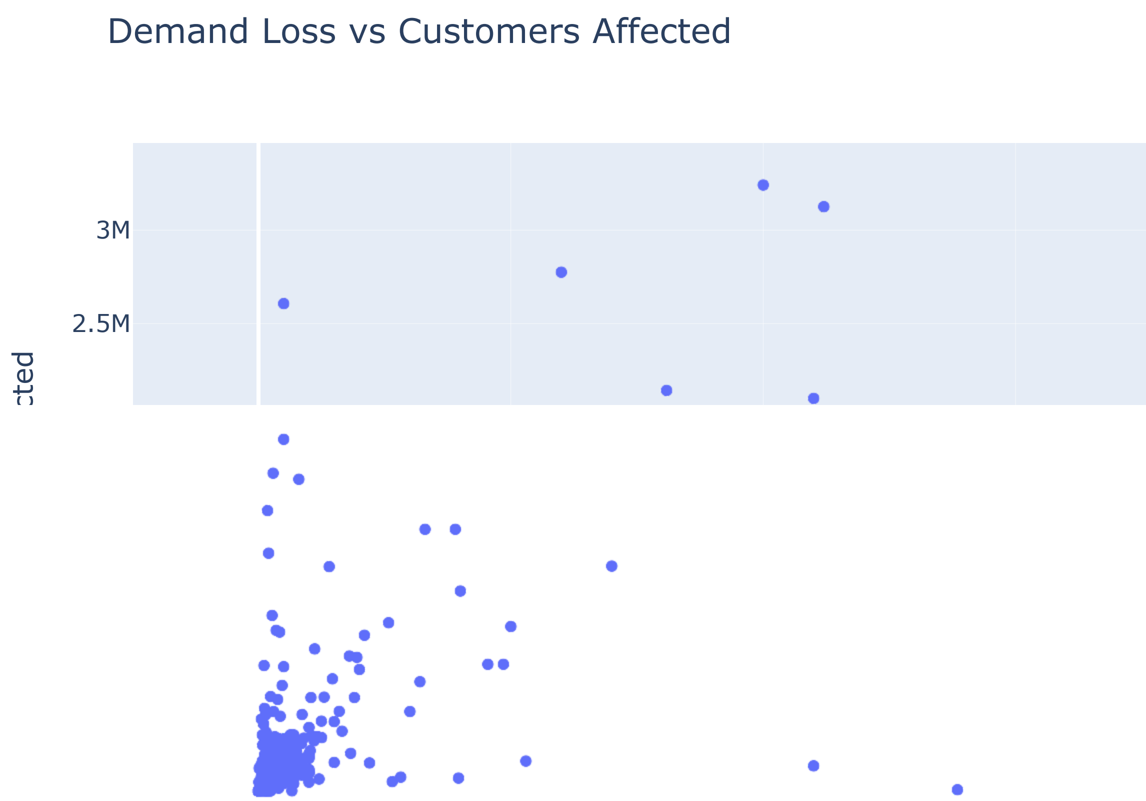

```
In [5]: small = outageClean[outageClean["OUTAGE.DURATION"] < 10000]

        fig = px.histogram(x = small['OUTAGE.DURATION'])
        fig.update_layout(title = "Distribution of Outage Duration", xaxis_title = "
        fig.show()
```



Bivariate Analysis

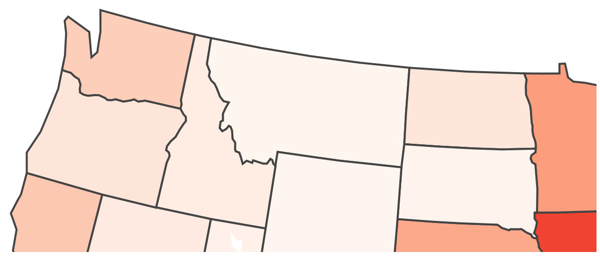
In [6]:
```
fig = px.scatter(outageClean, x='OUTAGE.DURATION', y='CUSTOMERS.AFFECTED', t
fig.update_layout(xaxis_title='Outage Duration (minutes)', yaxis_title='Cust
fig.show()

fig = px.scatter(outageClean, x='DEMAND.LOSS.MW', y='CUSTOMERS.AFFECTED', ti
fig.update_layout(xaxis_title='Demand Loss (MW)', yaxis_title='Customers Aff
fig.show()
```


Outage Duration vs Customers Affected
3M
2.5M
cted

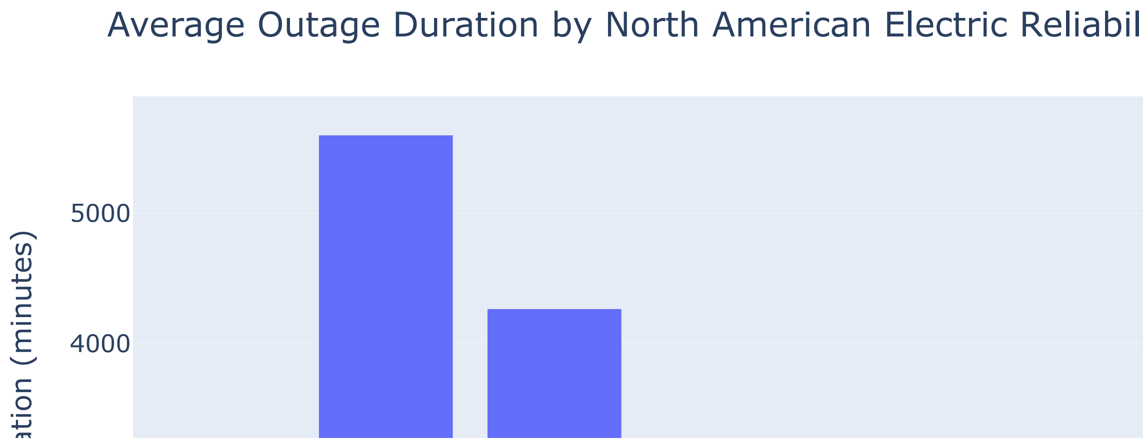


In [7]:
```
durationstate = outageClean.groupby("POSTAL.CODE")["OUTAGE.DURATION"].mean()

fig = px.choropleth(
    durationstate,
    locations="POSTAL.CODE",
    locationmode="USA-states",
    color="OUTAGE.DURATION",  # depper red = more customers affected
    scope="usa",
    title="Average Outage Duration by State",
    color_continuous_scale="Reds",
    labels={"OUTAGE.DURATION": "Average Outage Duration (minutes)"},
)
fig.show()
```

Average Outage Duration by State



In [8]:
```
durationregion = outageClean.groupby("NERC.REGION")["OUTAGE.DURATION"].mean(

fig = px.bar(durationregion, x = 'NERC.REGION', y = 'OUTAGE.DURATION')
fig.update_layout(title = "Average Outage Duration by North American Electri

fig.show()
```



Interesting Aggreates

In [9]:
```
outageClean.groupby('YEAR')['OUTAGE.DURATION'].mean().reset_index()
```

Out[9]:

	YEAR	OUTAGE.DURATION
0	2000.0	2843.076923
1	2001.0	1272.071429
2	2002.0	4751.000000
3	2003.0	4652.434783
4	2004.0	4368.788732
5	2005.0	5288.944444
6	2006.0	3329.530303
7	2007.0	2336.666667
8	2008.0	4184.018182
9	2009.0	3660.519481
10	2010.0	2937.528302
11	2011.0	1801.605948
12	2012.0	1877.976879
13	2013.0	1369.164474
14	2014.0	3107.355769
15	2015.0	935.811321
16	2016.0	2225.553191

In [10]:

```
#group by day of the week
outageClean['DATE'] = pd.to_datetime(outageClean['YEAR'], format='%Y')
outageClean['DAY'] = outageClean['DATE'].dt.day_name()
outageClean['DAY'] = pd.Categorical(outageClean['DAY'], categories=['Monday'
print(outageClean.groupby('DAY')['OUTAGE.DURATION'].mean().reset_index())
```

```
         DAY  OUTAGE.DURATION
0     Monday      2117.485294
1    Tuesday      2662.568841
2  Wednesday      3581.180000
3   Thursday      2721.417323
4     Friday      2718.816993
5   Saturday      2402.366071
6     Sunday      2278.824268
```

```
/var/folders/p_/yyb_zpwd6fj2v05f12wq70pc0000gn/T/ipykernel_91858/3888886289.
py:5: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
```

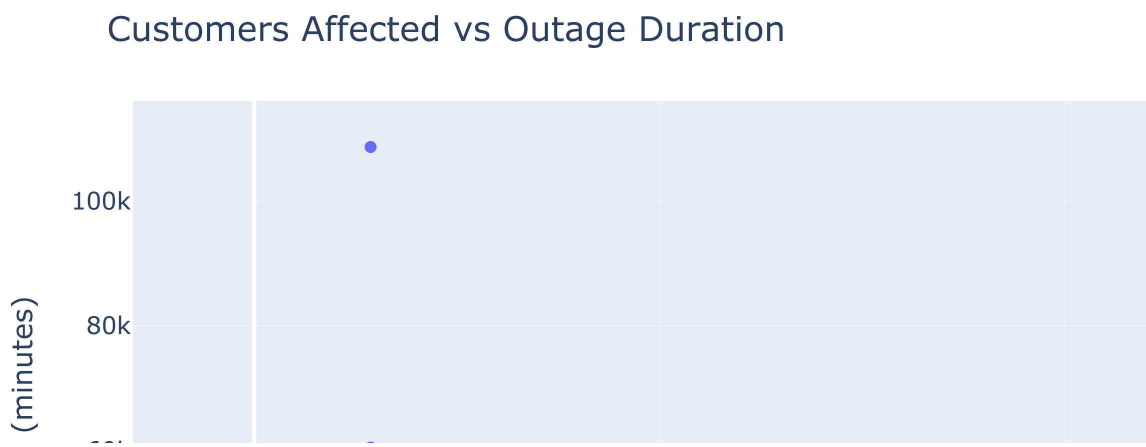
Baseline Model

In [11]:
```
numeric_columns = outageClean.select_dtypes(include=['number'])

# Compute the correlation matrix
correlation_matrix = numeric_columns.corr()
correlated_features = correlation_matrix['OUTAGE.DURATION'].drop('OUTAGE.DUR

print("Features most positively correlated with OUTAGE.DURATION:")
print(correlated_features[correlated_features > 0].head())

print("\nFeatures most negatively correlated with OUTAGE.DURATION:")
print(correlated_features[correlated_features < 0].head())
```

```
Features most positively correlated with OUTAGE.DURATION:
CUSTOMERS.AFFECTED    0.261916
DEMAND.LOSS.MW        0.026798
Name: OUTAGE.DURATION, dtype: float64

Features most negatively correlated with OUTAGE.DURATION:
YEAR   -0.144047
Name: OUTAGE.DURATION, dtype: float64
```

In [12]:
```
#linear regression
from sklearn.metrics import mean_squared_error, root_mean_squared_error

X = outageClean[['CUSTOMERS.AFFECTED']]
X = X.fillna(X.mean())
y = outageClean['OUTAGE.DURATION']
y = y.fillna(y.mean())
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
#print rmse
print(root_mean_squared_error(y_test, y_pred))

fig = px.scatter(x = X_test['CUSTOMERS.AFFECTED'], y = y_test)
fig.add_scatter(x = X_test['CUSTOMERS.AFFECTED'], y = y_pred, mode = 'lines'
fig.update_layout(title = "Customers Affected vs Outage Duration", xaxis_tit
fig.show()

#put into html
fig.write_html("assets/simpleLinReg.html")
```

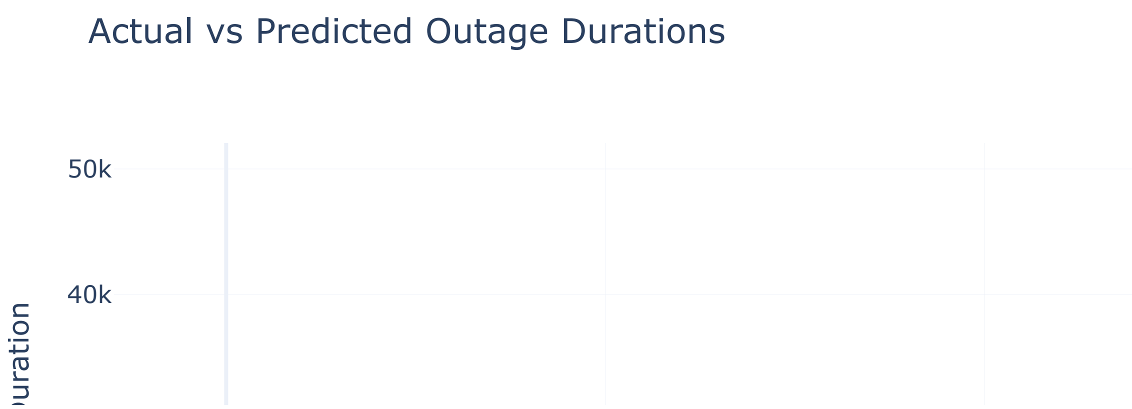
```
8350.875280739045
```



In [13]:
```
#lets do multiple linear regression, with every column except for the target

X = outageClean[['CAUSE.CATEGORY', 'NERC.REGION', 'U.S._STATE', 'YEAR', 'CUS
y = outageClean['OUTAGE.DURATION']

X.loc[:, 'YEAR'] = X['YEAR'].fillna(0).astype(int)
X.loc[:, 'CAUSE.CATEGORY'] = X['CAUSE.CATEGORY'].fillna("Missing")
X.loc[:, 'NERC.REGION'] = X['NERC.REGION'].fillna("Missing")
X.loc[:, 'U.S._STATE'] = X['U.S._STATE'].fillna("Missing")
X.loc[:, 'CUSTOMERS.AFFECTED'] = X['CUSTOMERS.AFFECTED'].fillna(X['CUSTOMERS
X.loc[:, 'DEMAND.LOSS.MW'] = X['DEMAND.LOSS.MW'].fillna(X['DEMAND.LOSS.MW'].
y = y.fillna(y.mean())

# Encode categorical columns
label_encoder_cause = LabelEncoder()
label_encoder_nerc = LabelEncoder()
label_encoder_state = LabelEncoder()

X.loc[:, 'U.S._STATE'] = label_encoder_state.fit_transform(X['U.S._STATE'])
X.loc[:, 'CAUSE.CATEGORY'] = label_encoder_cause.fit_transform(X['CAUSE.CATE
X.loc[:, 'NERC.REGION'] = label_encoder_nerc.fit_transform(X['NERC.REGION'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
model = LinearRegression()
```

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
rmse = root_mean_squared_error(y_test, y_pred)
print(f"Root Mean Squared Error: {rmse}")

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=y_test,
    y=y_pred,
    mode='markers',
    name='Predicted vs Actual',
    marker=dict(size=6, opacity=0.7)
))

fig.add_trace(go.Scatter(
    x=[y_test.min(), y_test.max()],
    y=[y_test.min(), y_test.max()],
    mode='lines',
    name='Perfect Prediction Line',
    line=dict(color='red', dash='dash')
))
fig.update_layout(
    title='Actual vs Predicted Outage Durations',
    xaxis_title='Actual Outage Duration',
    yaxis_title='Predicted Outage Duration',
    legend=dict(title="Legend"),
    template='plotly_white'
)
fig.show()
fig.write_html("assets/multLinReg.html")

print("HTML file with Plotly chart saved: assets/multLinReg.html")
```

```
Root Mean Squared Error: 4933.434321277098
```




```
HTML file with Plotly chart saved: assets/multLinReg.html
```

In [ ]:

```

```

In [14]:

```
# Prepare the data
X = outageClean[['CAUSE.CATEGORY', 'NERC.REGION', 'U.S._STATE', 'YEAR', 'CUS
y = outageClean['OUTAGE.DURATION']  # Target variable

# Handle missing values
X.loc[:, 'YEAR'] = X['YEAR'].fillna(0).astype(int)
X.loc[:, 'CAUSE.CATEGORY'] = X['CAUSE.CATEGORY'].fillna("Missing")
X.loc[:, 'NERC.REGION'] = X['NERC.REGION'].fillna("Missing")
X.loc[:, 'U.S._STATE'] = X['U.S._STATE'].fillna("Missing")
X.loc[:, 'CUSTOMERS.AFFECTED'] = X['CUSTOMERS.AFFECTED'].fillna(X['CUSTOMERS
X.loc[:, 'DEMAND.LOSS.MW'] = X['DEMAND.LOSS.MW'].fillna(X['DEMAND.LOSS.MW'].
y = y.fillna(y.mean())

# Label encoding for categorical features
label_encoder_state = LabelEncoder()
label_encoder_cause = LabelEncoder()
label_encoder_nerc = LabelEncoder()

X.loc[:, 'U.S._STATE'] = label_encoder_state.fit_transform(X['U.S._STATE'])
X.loc[:, 'CAUSE.CATEGORY'] = label_encoder_cause.fit_transform(X['CAUSE.CATE
```

```
X.loc[:, 'NERC.REGION'] = label_encoder_nerc.fit_transform(X['NERC.REGION'])

# Preprocessing pipeline (without Year_Demand_Interaction)
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('scaler', StandardScaler())
        ]), ['YEAR', 'CUSTOMERS.AFFECTED', 'DEMAND.LOSS.MW']),
        ('cat', Pipeline(steps=[
            ('log', FunctionTransformer(np.log1p, validate=True)),
            ('scaler', StandardScaler())
        ]), ['CAUSE.CATEGORY', 'NERC.REGION', 'U.S._STATE'])
    ]
)

# Define the pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('lasso', Lasso())
])

# Hyperparameter tuning
param_grid = {
    'lasso__alpha': [0.001, 0.01, 0.1, 1, 10, 100]
}

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

# Perform GridSearchCV
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squ
grid_search.fit(X_train, y_train)

# Evaluate the model
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
rmse = root_mean_squared_error(y_test, y_pred)

# Print results
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Best Alpha: {grid_search.best_params_['lasso__alpha']}")

# Extract feature importance
lasso_model = best_model.named_steps['lasso']
feature_names = ['YEAR', 'CUSTOMERS.AFFECTED', 'DEMAND.LOSS.MW', 'CAUSE.CATE
coefficients = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': lasso_model.coef_
}).sort_values(by='Coefficient', key=abs, ascending=False)

# Display the most influential features
print("Most influential features:")
print(coefficients.head(10))

# Print number of features
```

```
print(f"Number of features: {len(feature_names)}")
print(f"Number of non-zero coefficients: {len(coefficients[coefficients['Coe
```

```
Root Mean Squared Error (RMSE): 4955.1104505520325
Best Alpha: 100
Most influential features:
              Feature  Coefficient
1  CUSTOMERS.AFFECTED   736.246987
4         NERC.REGION  -496.593846
0                YEAR  -331.945961
5          U.S._STATE   276.073361
2      DEMAND.LOSS.MW   -42.942617
3      CAUSE.CATEGORY     0.000000
Number of features: 6
Number of non-zero coefficients: 5
```

In [15]:

```
X = outageClean[['CAUSE.CATEGORY', 'NERC.REGION', 'U.S._STATE', 'YEAR', 'CUS
y = outageClean['OUTAGE.DURATION']

X.loc[:, 'YEAR'] = X['YEAR'].fillna(0).astype(int)
X.loc[:, 'CAUSE.CATEGORY'] = X['CAUSE.CATEGORY'].fillna("Missing")
X.loc[:, 'NERC.REGION'] = X['NERC.REGION'].fillna("Missing")
X.loc[:, 'U.S._STATE'] = X['U.S._STATE'].fillna("Missing")
X.loc[:, 'CUSTOMERS.AFFECTED'] = X['CUSTOMERS.AFFECTED'].fillna(X['CUSTOMERS
X.loc[:, 'DEMAND.LOSS.MW'] = X['DEMAND.LOSS.MW'].fillna(X['DEMAND.LOSS.MW'].
y = y.fillna(y.mean())

label_encoder_state = LabelEncoder()
label_encoder_cause = LabelEncoder()
label_encoder_nerc = LabelEncoder()

X.loc[:, 'U.S._STATE'] = label_encoder_state.fit_transform(X['U.S._STATE'])
X.loc[:, 'CAUSE.CATEGORY'] = label_encoder_cause.fit_transform(X['CAUSE.CATE
X.loc[:, 'NERC.REGION'] = label_encoder_nerc.fit_transform(X['NERC.REGION'])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('scaler', StandardScaler())
        ]), ['YEAR', 'CUSTOMERS.AFFECTED', 'DEMAND.LOSS.MW']),
        ('cat', Pipeline(steps=[
            ('log', FunctionTransformer(np.log1p, validate=True)),
            ('scaler', StandardScaler())
        ]), ['CAUSE.CATEGORY', 'NERC.REGION', 'U.S._STATE'])
    ]
)

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('ridge', Ridge())
])

#hyperparameter tuning
param_grid = {
    'ridge__alpha': [0.001, 0.01, 0.1, 1, 10, 100]
}
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squ
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
rmse = root_mean_squared_error(y_test, y_pred)

# Print bonanaza
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Best Alpha: {grid_search.best_params_['ridge__alpha']}")
ridge_model = best_model.named_steps['ridge']
feature_names = ['YEAR', 'CUSTOMERS.AFFECTED', 'DEMAND.LOSS.MW', 'CAUSE.CATE
coefficients = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': ridge_model.coef_
}).sort_values(by='Coefficient', key=abs, ascending=False)

print("Most influential features:")
print(coefficients.head(10))
print(f"Number of features: {len(feature_names)}")
print(f"Number of non-zero coefficients: {len(coefficients[coefficients['Coe
```

```
Root Mean Squared Error (RMSE): 4948.611435073662
Best Alpha: 100
Most influential features:
              Feature  Coefficient
1  CUSTOMERS.AFFECTED   789.303757
4         NERC.REGION  -535.201748
0                YEAR  -392.811508
5          U.S._STATE   352.306317
2      DEMAND.LOSS.MW  -150.418881
3      CAUSE.CATEGORY     2.314334
Number of features: 6
Number of non-zero coefficients: 6
```

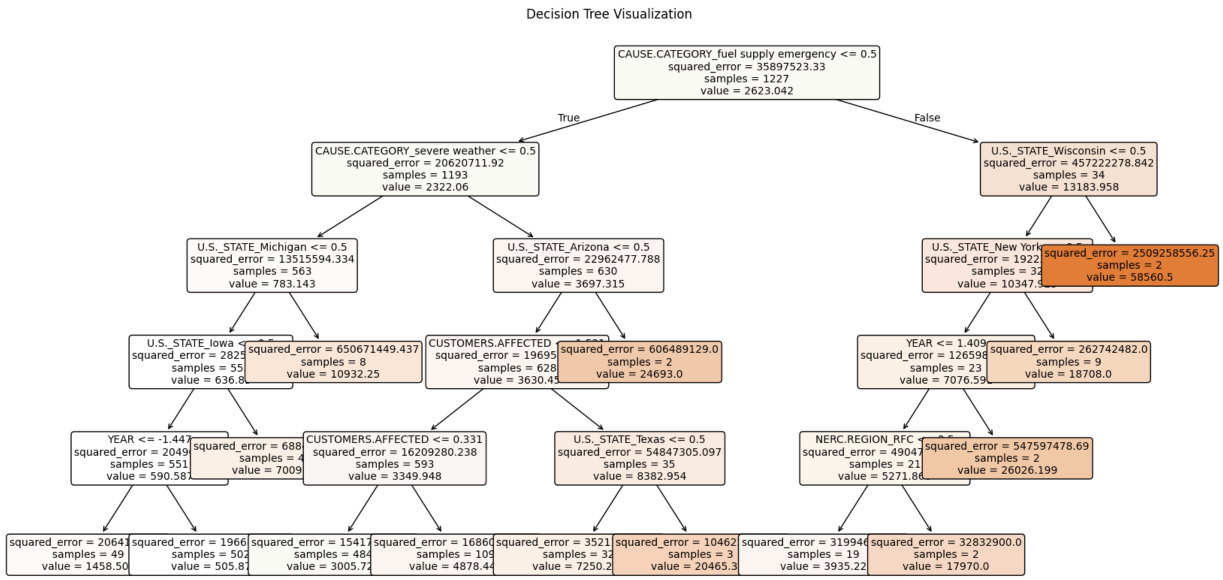
In [ ]:

In [16]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder, FunctionTra
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
import pandas as pd
import numpy as np


X = outageClean[['CAUSE.CATEGORY', 'NERC.REGION', 'U.S._STATE', 'YEAR', 'CUS
y = outageClean['OUTAGE.DURATION'].fillna(outageClean['OUTAGE.DURATION'].mea
#from error code
X.loc[:, 'YEAR'] = X['YEAR'].fillna(0).astype(int)
X.loc[:, 'CAUSE.CATEGORY'] = X['CAUSE.CATEGORY'].fillna("Missing")
X.loc[:, 'NERC.REGION'] = X['NERC.REGION'].fillna("Missing")
X.loc[:, 'U.S._STATE'] = X['U.S._STATE'].fillna("Missing")
X.loc[:, 'CUSTOMERS.AFFECTED'] = X['CUSTOMERS.AFFECTED'].fillna(0)
X.loc[:, 'DEMAND.LOSS.MW'] = X['DEMAND.LOSS.MW'].fillna(0)
```

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('scaler', StandardScaler())
        ]), ['YEAR', 'CUSTOMERS.AFFECTED', 'DEMAND.LOSS.MW']),
        ('cat', OneHotEncoder(handle_unknown='ignore'), ['CAUSE.CATEGORY', '
    ]
)

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', DecisionTreeRegressor(random_state=42))
])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
param_grid = {
    'regressor__max_depth': [3, 5, 10, None],
    'regressor__min_samples_split': [2, 5, 10],
    'regressor__min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squ
grid_search.fit(X_train, y_train)
final_model = grid_search.best_estimator_
y_pred = final_model.predict(X_test)
from sklearn.metrics import root_mean_squared_error
rmse = root_mean_squared_error(y_test, y_pred)
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Best Parameters: {grid_search.best_params_}")


from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
final_tree = final_model.named_steps['regressor']
preprocessor = final_model.named_steps['preprocessor']
categorical_features = preprocessor.transformers_[1][1].get_feature_names_ou
numeric_features = ['YEAR', 'CUSTOMERS.AFFECTED', 'DEMAND.LOSS.MW']
feature_names = list(numeric_features) + list(categorical_features)
plt.figure(figsize=(20, 10))
plot_tree(
    final_tree,
    feature_names=feature_names,
    filled=True,
    rounded=True,
    fontsize=10
)
plt.title("Decision Tree Visualization")
plt.show()
```

```
Root Mean Squared Error (RMSE): 6913.610292022421
Best Parameters: {'regressor__max_depth': 5, 'regressor__min_samples_leaf':
2, 'regressor__min_samples_split': 10}
```




```python
In [17]: def plot_sorted_longest_outages_by_year(data):

    # drop rows where YEAR is NaN
    data = data.dropna(subset=["YEAR"])
    outage_summary = (
        data.groupby(["POSTAL.CODE", "YEAR"])
        .agg({"OUTAGE.DURATION": "max"})
        .reset_index()
    )

    outage_summary = outage_summary.sort_values(by="YEAR")
    fig = px.choropleth(
        outage_summary,
        locations="POSTAL.CODE",
        locationmode="USA-states",
        color="OUTAGE.DURATION",
        scope="usa",
        animation_frame="YEAR",  # slider for year
        title="Longest Power Outages in the US by State (Yearly)",
        color_continuous_scale="Reds",
        labels={"OUTAGE.DURATION": "Outage Duration (minutes)", "YEAR": "Yea
    )

    return fig
plot_sorted_longest_outages_by_year(outageClean).show()
```

Longest Power Outages in the US by State (Yearly)