```
pip install ISLP
```

```
Requirement already satisfied: ISLP in /usr/local/lib/python3.10/dist-packages (0.3.22)
Requirement already satisfied: numpy<1.25,>=1.7.1 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.24.4)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.11.4)
Requirement already satisfied: pandas<=1.9,>=0.20 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.5.3)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from ISLP) (4.9.4)
Requirement already satisfied: scikit-learn>=1.2 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.2.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.4.0)
Requirement already satisfied: statsmodels>=0.13 in /usr/local/lib/python3.10/dist-packages (from ISLP) (0.14.1)
Requirement already satisfied: lifelines in /usr/local/lib/python3.10/dist-packages (from ISLP) (0.28.0)
Requirement already satisfied: pygam in /usr/local/lib/python3.10/dist-packages (from ISLP) (0.9.0)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from ISLP) (2.2.1+cu121)
Requirement already satisfied: pytorch-lightning in /usr/local/lib/python3.10/dist-packages (from ISLP) (2.2.2)
Requirement already satisfied: torchmetrics in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.3.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas<=1.9,>=0.20->ISLP) (2.8
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<=1.9,>=0.20->ISLP) (2023.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.2->ISLP) (3.4.0)
Requirement already satisfied: patsy>=0.5.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13->ISLP) (0.5.6)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13->ISLP) (24.0)
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP) (3.7.1)
Requirement already satisfied: autograd>=1.5 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP) (1.6.2)
Requirement already satisfied: autograd-gamma>=0.3 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP) (0.5.0)
Requirement already satisfied: formulaic>=0.2.2 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP) (1.0.1)
Requirement already satisfied: progressbar2<5.0.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pygam->ISLP) (4.2.0)
Requirement already satisfied: tqdm>=4.57.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (4.66.2)
Requirement already satisfied: PyYAML>=5.4 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (6.0.1)
Requirement already satisfied: fsspec[http]>=2022.5.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (2023
Requirement already satisfied: typing-extensions>=4.4.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (4
Requirement already satisfied: lightning-utilities>=0.8.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.13.4)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.1.3)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.1
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.1
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (8.9.2.26)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.3.1)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (11.0.2.54)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (10.3.2.1
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (11.4.5
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.0
Requirement already satisfied: nvidia-nccl-cu12==2.19.3 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (2.19.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.105)
Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (2.2.0)
Requirement already satisfied: nvidia-nvjitlink-cu12 in /usr/local/lib/python3.10/dist-packages (from nvidia-cusolver-cu12==11.4.5.10
Requirement already satisfied: future>=0.15.2 in /usr/local/lib/python3.10/dist-packages (from autograd>=1.5->lifelines->ISLP) (0.18
Requirement already satisfied: interface-meta>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from formulaic>=0.2.2->lifelines->IS
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.10/dist-packages (from formulaic>=0.2.2->lifelines->ISLP) (1.14.1
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2022.5.0->pytorch-lightning->I
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2022.5.0->py
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from lightning-utilities>=0.8.0->pytorch-lightr
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (0.12
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (9.4
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (3
```

```
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import pandas as pd
from matplotlib.pyplot import subplots
import statsmodels.api as sm
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
                         summarize,
                         poly)
from ISLP import confusion_table
from sklearn.metrics import accuracy_score
from ISLP.models import sklearn_sm
from sklearn.model_selection import \
     (cross_validate,
      KFold,
      ShuffleSplit)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
```

```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

```
#mount to google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Source of Data: SREEKAR G V - Kaggle

https://www.kaggle.com/datasets/sreekargv/bike-rentals?resource=download

```
bike_df = pd.read_csv('/content/archive (2).zip')
```

Data Dictionary (Taken from Kaggle): datetime: year/month/day/time(military)

season: 1: spring, 2: summer, 3: fall, 4: winter

holiday: whether day is a holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)

workingday: if day is neither weekend nor holiday is 1, otherwise is 0.

weather: 1: Clear, Few clouds, partly cloudy, partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

temp: temperature in Celsius

atemp: feeling temperature in Celsius

humidity: humidity

windspeed: wind speed

casual: count of casual users

registered: count of registered users

count: count of total rental bikes including both casual and registered

```
bike_df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

Data Cleaning:

No extra cleaning is necessary for the dataset. I looked through the dataset, there are no N/A, repetitions, etc.

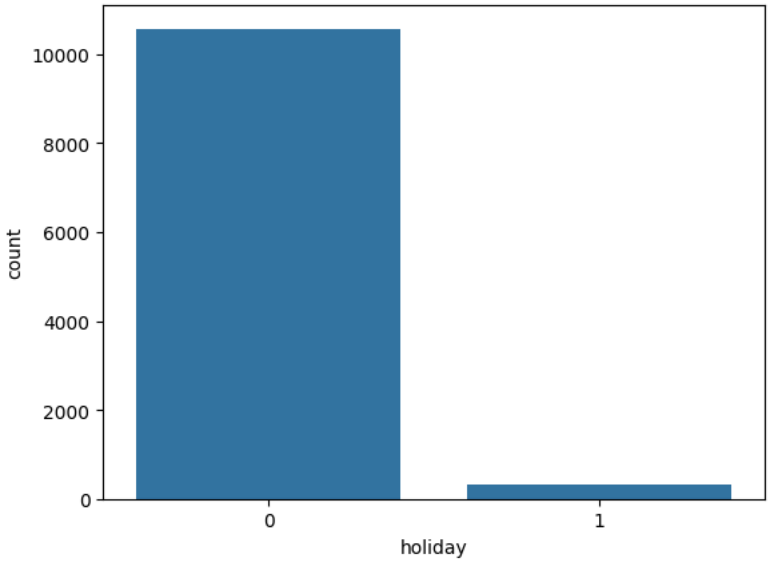
Data Encoding/Scaling:

No extra scaling or encoding was needed for this dataset. Each qualitative variable is already encoded with dummy variables. Based on the models I am building, I do not need to scale my data.

Data Visualizations:

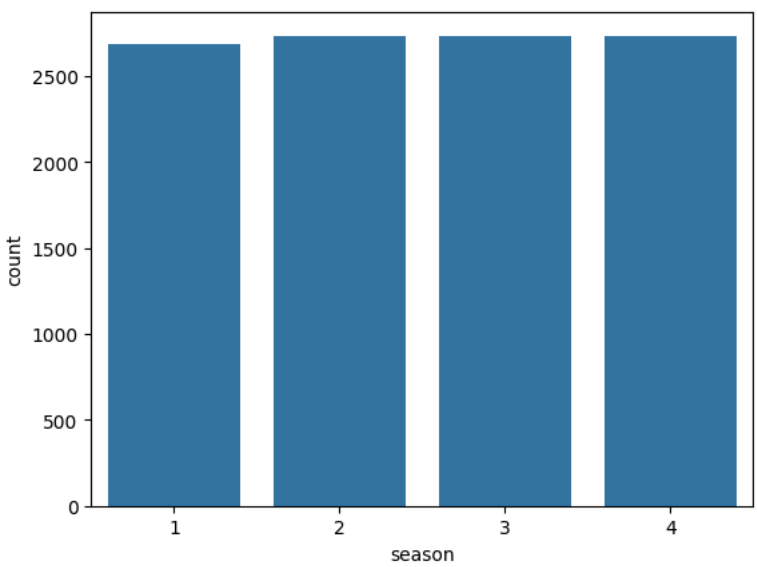
```
#Visualization One
sns.countplot(data= bike_df, x= 'holiday')
```

```
<Axes: xlabel='holiday', ylabel='count'>
```




```
#Visualization Two
sns.countplot(data= bike_df, x= 'season')
```

```
<Axes: xlabel='season', ylabel='count'>
```

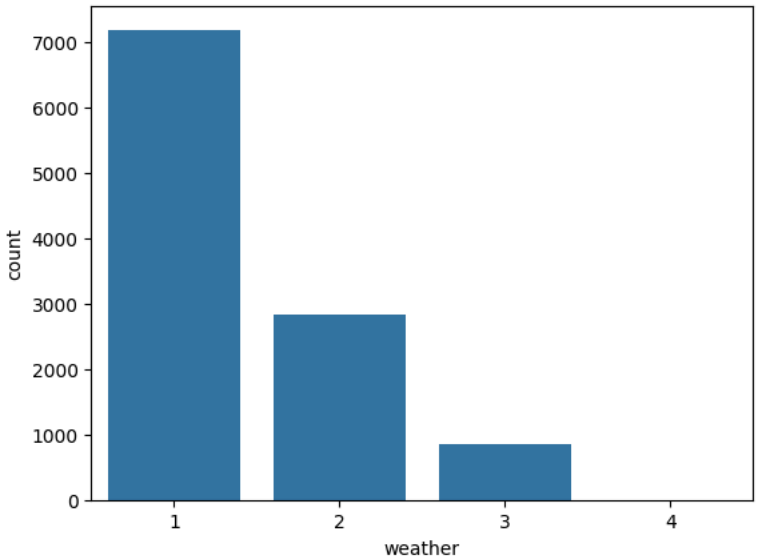


Prediction 1

Question: I am testing to see which of the columns are good predictors for weather = 1 (clear) using a Logistic Regression Model.

```
#Distribution of Weather (target)
sns.countplot(data=bike_df, x= 'weather')
```

```
<Axes: xlabel='weather', ylabel='count'>
```




```
bike_df['weather'].count()
```

```
10886
```

```
not_weather = bike_df.columns.drop(['weather', 'datetime'])
design = MS(not_weather)
X = design.fit_transform(bike_df)
y = bike_df.weather == 1
glm = sm.GLM(y,
             X,
             family=sm.families.Binomial())
results = glm.fit()
summarize(results)
```

	coef	std err	z	P>|z|
intercept	4.4193	0.145	30.564	0.000
season	0.0956	0.021	4.497	0.000
holiday	-0.1894	0.134	-1.409	0.159
workingday	-0.2533	0.056	-4.484	0.000
temp	0.0015	0.017	0.087	0.930
atemp	0.0154	0.016	0.976	0.329
humidity	-0.0565	0.002	-36.950	0.000
windspeed	-0.0340	0.003	-11.165	0.000
casual	-0.0021	0.000	-4.216	0.000
registered	0.0010	0.000	3.145	0.002
count	0.0011	0.000	5.276	0.000

Interpretation of Summary:

holiday, temp, and atemp are not significant because their p-values are greater than 0.05. Meaning, holiday has 0.159 probability that there is no relationship with weather=1 and temp is 0.930 and atemp is 0.329. Every other variable does have a relationship with weather=1 because the p-values are less than 0.05.

```
clear_array = np.zeros(len(bike_df.weather))
idx_clear = np.where(bike_df.weather == 1)[0]
clear_array[idx_clear] = 1
```

```
clear_array
```

```
array([1., 1., 1., ..., 1., 1., 1.])
```

```
#7192 of the datapoints are 1 out of the 10886 total
sum(clear_array)
```

```
7192.0
```

```
#total length of the clear_array
len(clear_array)
```

```
10886
```

```
probs = results.predict()
probs[:10]

labels = np.array([0]*10886)
labels[probs>0.5] = 1

confusion_table(labels, clear_array)
```

Truth	0.0	1.0
Predicted		
0.0	1646	825
1.0	2048	6367

```
#Accuracy score
accuracy_score(labels, clear_array)
```

```
0.736083042439831
```

Interpretation of Confusion Matrix:

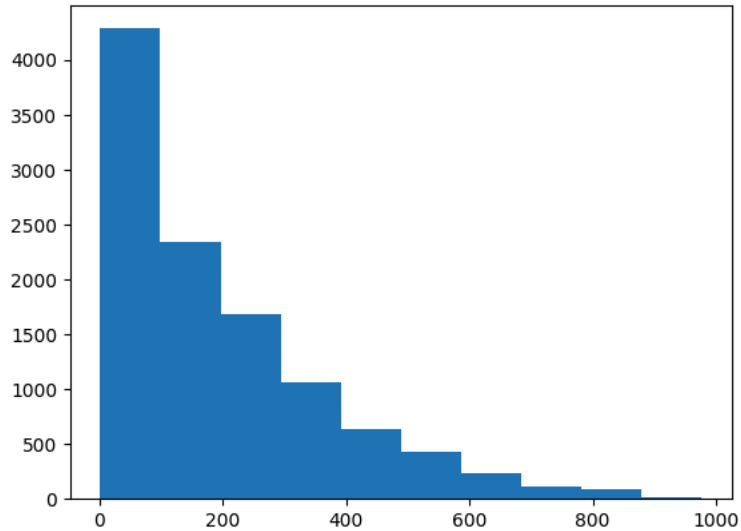
The biggest mistake my model is making is when it is predicting 1 when it is really 0. I think it is making this mistake because the threshold might be too strong or not strong enough. It makes this mistake 2048 times. My model performs well when it guesses 1 and it actually is 1 it does this correct 6367 times.

Prediction 2

Question: I am testing to see which columns are the best predictors for the target variable count using a Linear Regression Model.

```
#Distribution of Counts (target)
plt.hist(bike_df['count'])
```

```
(array([4284., 2337., 1686., 1067.,  633.,  426.,  233.,  116.,   85.,
          19.]),
 array([  1. ,  98.6, 196.2, 293.8, 391.4, 489. , 586.6, 684.2, 781.8,
        879.4, 977. ]),
 <BarContainer object of 10 artists>)
```




```
design_interaction = MS(['season', 'holiday', 'workingday', 'temp', 'atemp', 'humidity', 'windspeed'])
design_interaction = design_interaction.fit(bike_df)
X = design_interaction.transform(bike_df)

#predicting count
y = bike_df['count']
#initializing model named "model"
count_model = sm.OLS(y, X)
#fit the model
count_results = count_model.fit()
#print coeff and statistics
summarize(count_results)
```

	coef	std err	t	P>|t|
intercept	138.2457	8.644	15.993	0.000
season	22.5740	1.425	15.846	0.000
holiday	-9.1419	9.272	-0.986	0.324
workingday	-0.8567	3.313	-0.259	0.796
temp	1.9117	1.142	1.674	0.094
atemp	5.5670	1.050	5.300	0.000
humidity	-2.9660	0.084	-35.508	0.000
windspeed	0.8425	0.198	4.256	0.000

Interpretation of Summary:

The predictors that are not significant are temp, workingday, and holiday because their p-values are greater than 0.05. Season, atemp, humidity, and windspeed are all significant since their p-values are less than 0.05.

```
#R^2 Score
count_results.rsquared
```

```
0.2605631584232564
```

Interpretation of R^2:

The model does not perform well, it only accurately predicts count 26% of the time.

Cross Validation and Tuning:

Cross Validation

```
length_model = sklearn_sm(sm.OLS,
                      MS(['season', 'holiday', 'workingday', 'temp', 'atemp', 'humidity', 'windspeed']))
```

```
cv_results = cross_validate(length_model, X, y, cv = 5)
```

```
cv_results
```

```
{'fit_time': array([0.03634477, 0.03599215, 0.03635883, 0.03784013, 0.03284431]),
 'score_time': array([0.01467156, 0.01499867, 0.01279449, 0.01325083, 0.0149107 ]),
 'test_score': array([13620.25618454, 25083.00407027, 16311.82469564, 40572.62515677,
        45559.50059793])}
```

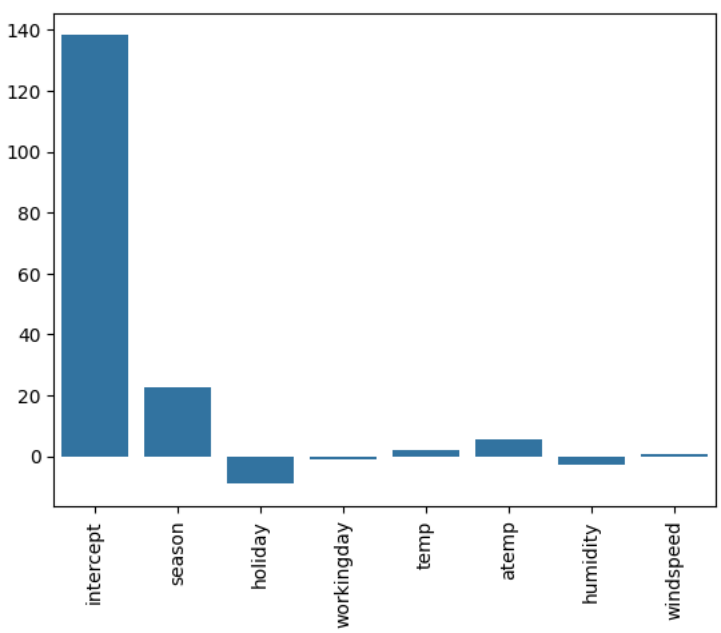
```
cv_results['test_score'].mean()
```

```
28229.44214102945
```

I used a 5-fold cross validation because my dataset is large so LOOCV would be too expensive with both time and computer storage.

```
#Feature importance of Prediction 2
count_coefs= count_results.params
sns.barplot(count_coefs)
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'intercept'),
  Text(1, 0, 'season'),
  Text(2, 0, 'holiday'),
  Text(3, 0, 'workingday'),
  Text(4, 0, 'temp'),
  Text(5, 0, 'atemp'),
  Text(6, 0, 'humidity'),
  Text(7, 0, 'windspeed')])
```



Tuning

```
alphas = 10**np.linspace(10,-2,100)*0.5
```

```
alphas
```

```
array([5.00000000e+09, 3.78231664e+09, 2.86118383e+09, 2.16438064e+09,
       1.63727458e+09, 1.23853818e+09, 9.36908711e+08, 7.08737081e+08,
       5.36133611e+08, 4.05565415e+08, 3.06795364e+08, 2.32079442e+08,
       1.75559587e+08, 1.32804389e+08, 1.00461650e+08, 7.59955541e+07,
       5.74878498e+07, 4.34874501e+07, 3.28966612e+07, 2.48851178e+07,
       1.88246790e+07, 1.42401793e+07, 1.07721735e+07, 8.14875417e+06,
       6.16423370e+06, 4.66301673e+06, 3.52740116e+06, 2.66834962e+06,
       2.01850863e+06, 1.52692775e+06, 1.15506485e+06, 8.73764200e+05,
       6.60970574e+05, 5.00000000e+05, 3.78231664e+05, 2.86118383e+05,
       2.16438064e+05, 1.63727458e+05, 1.23853818e+05, 9.36908711e+04,
       7.08737081e+04, 5.36133611e+04, 4.05565415e+04, 3.06795364e+04,
       2.32079442e+04, 1.75559587e+04, 1.32804389e+04, 1.00461650e+04,
       7.59955541e+03, 5.74878498e+03, 4.34874501e+03, 3.28966612e+03,
       2.48851178e+03, 1.88246790e+03, 1.42401793e+03, 1.07721735e+03,
       8.14875417e+02, 6.16423370e+02, 4.66301673e+02, 3.52740116e+02,
       2.66834962e+02, 2.01850863e+02, 1.52692775e+02, 1.15506485e+02,
       8.73764200e+01, 6.60970574e+01, 5.00000000e+01, 3.78231664e+01,
       2.86118383e+01, 2.16438064e+01, 1.63727458e+01, 1.23853818e+01,
       9.36908711e+00, 7.08737081e+00, 5.36133611e+00, 4.05565415e+00,
       3.06795364e+00, 2.32079442e+00, 1.75559587e+00, 1.32804389e+00,
       1.00461650e+00, 7.59955541e-01, 5.74878498e-01, 4.34874501e-01,
       3.28966612e-01, 2.48851178e-01, 1.88246790e-01, 1.42401793e-01,
       1.07721735e-01, 8.14875417e-02, 6.16423370e-02, 4.66301673e-02,
       3.52740116e-02, 2.66834962e-02, 2.01850863e-02, 1.52692775e-02,
       1.15506485e-02, 8.73764200e-03, 6.60970574e-03, 5.00000000e-03])
```

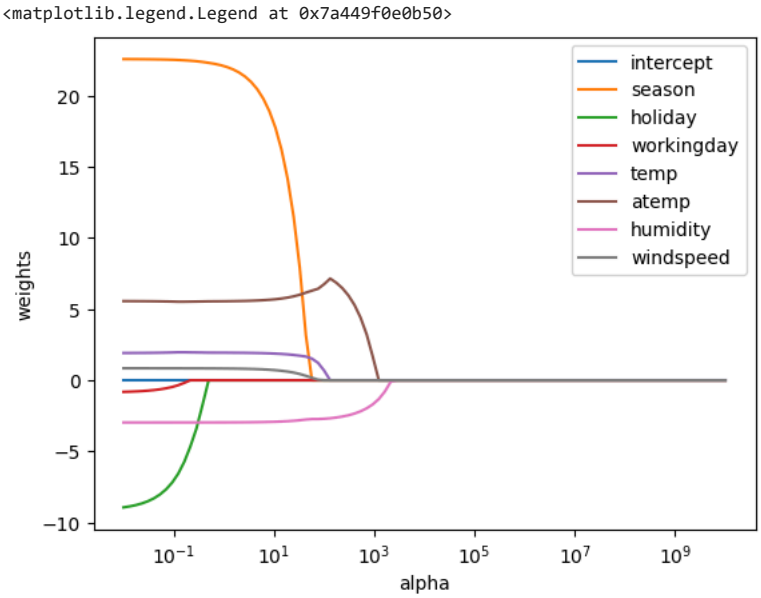
Feature Selection Method: Lasso

```
lasso = Lasso(max_iter = 10000)
coefs = []
MSEs = []

for a in alphas:
  lasso.set_params(alpha=a)
  lasso.fit(X, y)
  coefs.append(lasso.coef_)
  pred = lasso.predict(X)
  MSEs.append(mean_squared_error(clear_array, pred))

ax = plt.gca()
ax.plot(alphas*2, coefs, label = X.columns)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7a449f0e0b50>
```



Tuning Parameter Method: Lasso MSE

```
#plotting the coef as a function of tuning param
plt.plot(alphas, MSEs)
```

```
plt.xscale('log')
```