Download data

- Go to https://www.ncdc.noaa.gov/cdo-web/search (https://www.ncdc.noaa.gov/cdo-web/search)
- Enter the years you want data for (I recommend starting with 1970), and search for the closest airport to you
  -
- Click add to cart on the airport you want
  - If there is no airport near you, try your city or country name instead
  -
- Go to the cart at https://www.ncdc.noaa.gov/cdo-web/cart (https://www.ncdc.noaa.gov/cdo-web/cart)
- Select the csv format and click continue
  -
- Select all of the checkboxes for data types
  -
- Enter your email and click continue
  -
- You'll get an email with a link to download the data
  -
- Make sure to take a look at the data documentation (https://www1.ncdc.noaa.gov/pub/data/cdo/documentation/GHCND_documentation.pdf) as well

In [1]:
```
import pandas as pd

weather = pd.read_csv("local_weather.csv", index_col="DATE")
```

In [2]:
```
weather
```

Out[2]:

DATE	STATION	NAME	ACMH	ACSH	AWND	DAPR	FMTM	FRGT	MDPR	PGTI
1960-01-01	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1960-01-02	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1960-01-03	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1960-01-04	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1960-01-05	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	.
2022-01-24	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	4.47	NaN	NaN	NaN	NaN	Na
2022-01-25	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	4.70	NaN	NaN	NaN	NaN	Na
2022-01-26	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	2.68	NaN	NaN	NaN	NaN	Na
2022-01-27	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	3.13	NaN	NaN	NaN	NaN	1526.
2022-01-28	USW00023230	OAKLAND INTERNATIONAL AIRPORT, CA US	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

16859 rows × 35 columns

In [506]:
```
weather.apply(pd.isnull).sum()/weather.shape[0]
```

Out[506]:
```
STATION    0.000000
NAME       0.000000
ACMH       0.653360
ACSH       0.653360
AWND       0.522451
DAPR       0.999525
FMTM       0.870099
FRGT       0.999881
MDPR       0.999525
PGTM       0.495106
PRCP       0.016668
SNOW       0.324990
SNWD       0.317634
TAVG       0.879174
TMAX       0.000534
TMIN       0.000593
TSUN       0.931728
WDF1       0.653360
WDF2       0.522392
WDF5       0.527552
WDFG       0.746901
WSF1       0.653360
WSF2       0.522332
WSF5       0.527552
WSFG       0.746901
WT01       0.779939
WT02       0.980248
WT03       0.992941
WT04       0.999763
WT05       0.998339
WT07       0.999881
WT08       0.810368
WT09       0.999881
WT16       0.884038
WT18       0.999822
dtype: float64
```

In [507]:
```
core_weather = weather[["PRCP", "SNOW", "SNWD", "TMAX", "TMIN"]].copy()
core_weather.columns = ["precip", "snow", "snow_depth", "temp_max", "temp_min"]
```

In [508]:
```
core_weather.apply(pd.isnull).sum()
```

Out[508]:
```
precip         281
snow          5479
snow_depth    5355
temp_max         9
temp_min        10
dtype: int64
```

In [509]:
```
core_weather["snow"].value_counts()
```

Out[509]:
```
0.0    11379
1.0        1
Name: snow, dtype: int64
```

In [510]:
```
core_weather["snow_depth"].value_counts()
```

Out[510]:
```
0.0    11504
Name: snow_depth, dtype: int64
```

In [511]:
```
del core_weather["snow"]
```

In [512]:
```
del core_weather["snow_depth"]
```

In [513]:
```
core_weather[pd.isnull(core_weather["precip"])]
```

Out[513]:

	precip	temp_max	temp_min
DATE			
1983-10-29	NaN	67.0	57.0
1983-10-30	NaN	70.0	63.0
1983-10-31	NaN	69.0	61.0
1983-11-12	NaN	63.0	55.0
1983-11-13	NaN	60.0	50.0
...	...	...	...
2013-12-15	NaN	58.0	33.0
2016-05-01	NaN	80.0	55.0
2016-05-02	NaN	68.0	53.0
2016-05-08	NaN	67.0	56.0
2017-10-28	NaN	68.0	50.0

281 rows × 3 columns

In [514]:
```
core_weather.loc["2013-12-15",:]
```

Out[514]:
```
precip       NaN
temp_max    58.0
temp_min    33.0
Name: 2013-12-15, dtype: float64
```

In [515]:
```
core_weather["precip"].value_counts() / core_weather.shape[0]
```

Out[515]:
```
0.00    0.810487
0.01    0.025980
0.02    0.011804
0.03    0.007236
0.04    0.006050
          ...
1.29    0.000059
1.73    0.000059
1.05    0.000059
1.38    0.000059
1.02    0.000059
Name: precip, Length: 176, dtype: float64
```

In [516]:
```
core_weather["precip"] = core_weather["precip"].fillna(0)
```

In [517]:
```
core_weather.apply(pd.isnull).sum()
```

Out[517]:
```
precip       0
temp_max     9
temp_min    10
dtype: int64
```

In [518]:
```
core_weather[pd.isnull(core_weather["temp_min"])]
```

Out[518]:

DATE	precip	temp_max	temp_min
2004-11-20	0.0	NaN	NaN
2011-12-21	0.0	61.0	NaN
2011-12-22	0.0	62.0	NaN
2011-12-23	0.0	56.0	NaN
2011-12-24	0.0	55.0	NaN
2011-12-25	0.0	54.0	NaN
2013-06-16	0.0	NaN	NaN
2020-08-29	0.0	NaN	NaN
2020-09-08	0.0	NaN	NaN
2020-09-09	0.0	NaN	NaN

In [519]:
```
core_weather.loc["2011-12-18":"2011-12-28"]
```

Out[519]:

DATE	precip	temp_max	temp_min
2011-12-18	0.0	52.0	33.0
2011-12-19	0.0	55.0	35.0
2011-12-20	0.0	61.0	35.0
2011-12-21	0.0	61.0	NaN
2011-12-22	0.0	62.0	NaN
2011-12-23	0.0	56.0	NaN
2011-12-24	0.0	55.0	NaN
2011-12-25	0.0	54.0	NaN
2011-12-26	0.0	50.0	32.0
2011-12-27	0.0	56.0	39.0
2011-12-28	0.0	57.0	38.0

In [520]:
```
core_weather = core_weather.fillna(method="ffill")
```

In [521]:
```
core_weather.apply(pd.isnull).sum()
```

Out[521]:
```
precip      0
temp_max    0
temp_min    0
dtype: int64
```

In [522]:
```
# Check for missing value defined in data documentation
core_weather.apply(lambda x: (x == 9999).sum())
```

Out[522]:
```
precip      0
temp_max    0
temp_min    0
dtype: int64
```

In [523]:
```
core_weather.dtypes
```

Out[523]:
```
precip      float64
temp_max    float64
temp_min    float64
dtype: object
```

In [524]:
```
core_weather.index
```

Out[524]:
```
Index(['1960-01-01', '1960-01-02', '1960-01-03', '1960-01-04', '1960-01-05',
       '1960-01-06', '1960-01-07', '1960-01-08', '1960-01-09', '1960-01-10',
       ...
       '2022-01-19', '2022-01-20', '2022-01-21', '2022-01-22', '2022-01-23',
       '2022-01-24', '2022-01-25', '2022-01-26', '2022-01-27', '2022-01-28'],
      dtype='object', name='DATE', length=16859)
```

In [525]:
```
core_weather.index = pd.to_datetime(core_weather.index)
```

In [526]:
```
core_weather.index
```

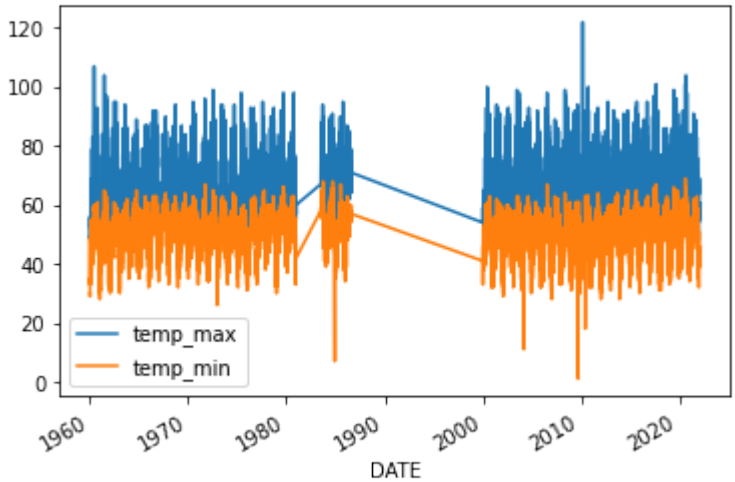
Out[526]:
```
DatetimeIndex(['1960-01-01', '1960-01-02', '1960-01-03', '1960-01-04',
               '1960-01-05', '1960-01-06', '1960-01-07', '1960-01-08',
               '1960-01-09', '1960-01-10',
               ...
               '2022-01-19', '2022-01-20', '2022-01-21', '2022-01-22',
               '2022-01-23', '2022-01-24', '2022-01-25', '2022-01-26',
               '2022-01-27', '2022-01-28'],
              dtype='datetime64[ns]', name='DATE', length=16859, freq=None)
```

In [527]:
```
core_weather.index.year
```

Out[527]:
```
Int64Index([1960, 1960, 1960, 1960, 1960, 1960, 1960, 1960, 1960, 1960,
            ...
            2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022],
           dtype='int64', name='DATE', length=16859)
```

In [528]:
```
core_weather[["temp_max", "temp_min"]].plot()
```

Out[528]:
```
<AxesSubplot:xlabel='DATE'>
```

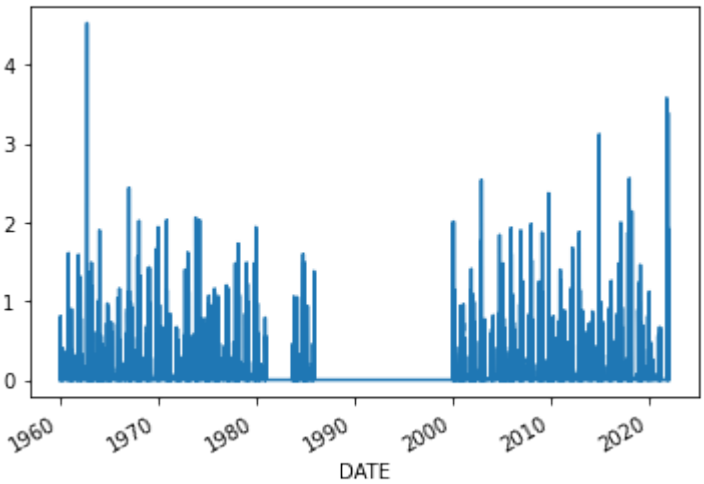


In [529]:
```
core_weather.index.year.value_counts().sort_index()
```

Out[529]:
```
1960    366
1961    365
1962    365
1963    365
1964    366
1965    365
1966    365
1967    365
1968    366
1969    365
1970    365
1971    365
1972    366
1973    365
1974    365
1975    365
1976    366
1977    365
1978    365
1979    365
1980    366
1983    184
1984    366
1985    365
1986    212
2000    365
2001    365
2002    365
2003    365
2004    366
2005    365
2006    365
2007    365
2008    366
2009    365
2010    365
2011    365
2012    365
2013    365
2014    365
2015    365
2016    366
2017    365
2018    365
2019    365
2020    366
2021    364
2022     28
Name: DATE, dtype: int64
```

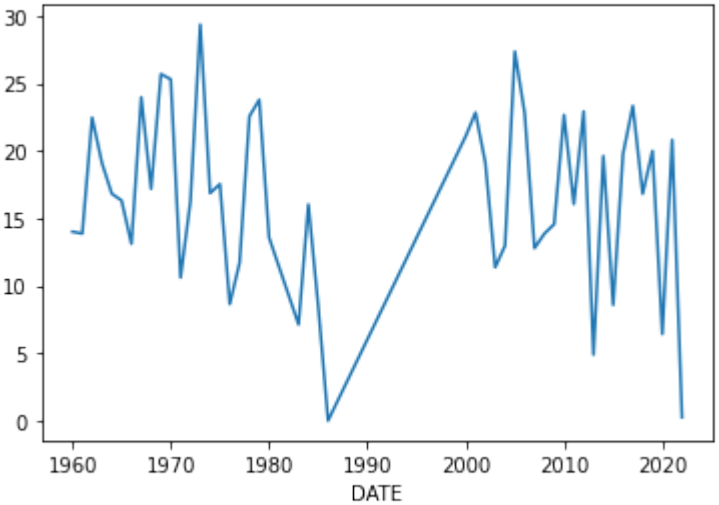
In [530]:
```
core_weather["precip"].plot()
```

Out[530]: <AxesSubplot:xlabel='DATE'>



In [531]:
```
core_weather.groupby(core_weather.index.year).apply(lambda x: x["precip"].sum()
```

Out[531]: <AxesSubplot:xlabel='DATE'>



In [532]:
```
core_weather["target"] = core_weather.shift(-1)["temp_max"]
```

In [533]:
```
core_weather
```

Out[533]:

DATE	precip	temp_max	temp_min	target
1960-01-01	0.0	49.0	30.0	49.0
1960-01-02	0.0	49.0	29.0	54.0
1960-01-03	0.0	54.0	35.0	54.0
1960-01-04	0.0	54.0	36.0	55.0
1960-01-05	0.0	55.0	33.0	53.0
...	...	...	...	...
2022-01-24	0.0	60.0	39.0	57.0
2022-01-25	0.0	57.0	43.0	57.0
2022-01-26	0.0	57.0	41.0	67.0
2022-01-27	0.0	67.0	39.0	64.0
2022-01-28	0.0	64.0	39.0	NaN

16859 rows × 4 columns

In [534]:
```
core_weather = core_weather.iloc[:-1,:].copy()
```

In [535]:
```
core_weather
```

Out[535]:

DATE	precip	temp_max	temp_min	target
1960-01-01	0.0	49.0	30.0	49.0
1960-01-02	0.0	49.0	29.0	54.0
1960-01-03	0.0	54.0	35.0	54.0
1960-01-04	0.0	54.0	36.0	55.0
1960-01-05	0.0	55.0	33.0	53.0
...	...	...	...	...
2022-01-23	0.0	60.0	41.0	60.0
2022-01-24	0.0	60.0	39.0	57.0
2022-01-25	0.0	57.0	43.0	57.0
2022-01-26	0.0	57.0	41.0	67.0
2022-01-27	0.0	67.0	39.0	64.0

16858 rows × 4 columns

In [536]:
```
from sklearn.linear_model import Ridge

reg = Ridge(alpha=.1)
```

In [537]:
```
predictors = ["precip", "temp_max", "temp_min"]
```

In [538]:
```
train = core_weather.loc[:"2020-12-31"]
test = core_weather.loc["2021-01-01":]
```

In [539]:
```
train
```

Out[539]:

	precip	temp_max	temp_min	target
DATE				
1960-01-01	0.00	49.0	30.0	49.0
1960-01-02	0.00	49.0	29.0	54.0
1960-01-03	0.00	54.0	35.0	54.0
1960-01-04	0.00	54.0	36.0	55.0
1960-01-05	0.00	55.0	33.0	53.0
...	...	...	...	...
2020-12-27	0.00	63.0	44.0	61.0
2020-12-28	0.10	61.0	42.0	60.0
2020-12-29	0.00	60.0	39.0	56.0
2020-12-30	0.07	56.0	36.0	62.0
2020-12-31	0.06	62.0	44.0	60.0

16467 rows × 4 columns

In [540]:
```
test
```

Out[540]:

DATE	precip	temp_max	temp_min	target
2021-01-01	0.00	60.0	40.0	57.0
2021-01-02	0.14	57.0	51.0	56.0
2021-01-03	0.00	56.0	49.0	62.0
2021-01-04	0.36	62.0	46.0	59.0
2021-01-05	0.00	59.0	42.0	59.0
...	...	...	...	...
2022-01-23	0.00	60.0	41.0	60.0
2022-01-24	0.00	60.0	39.0	57.0
2022-01-25	0.00	57.0	43.0	57.0
2022-01-26	0.00	57.0	41.0	67.0
2022-01-27	0.00	67.0	39.0	64.0

391 rows × 4 columns

In [541]:
```
reg.fit(train[predictors], train["target"])
```

Out[541]:
```
Ridge(alpha=0.1)
```

In [542]:
```
predictions = reg.predict(test[predictors])
```

In [543]:
```
from sklearn.metrics import mean_squared_error

mean_squared_error(test["target"], predictions)
```

Out[543]:
```
20.560668548118763
```

In [544]:
```
combined = pd.concat([test["target"], pd.Series(predictions, index=test.index)]
combined.columns = ["actual", "predictions"]
```

In [545]:
```
combined
```

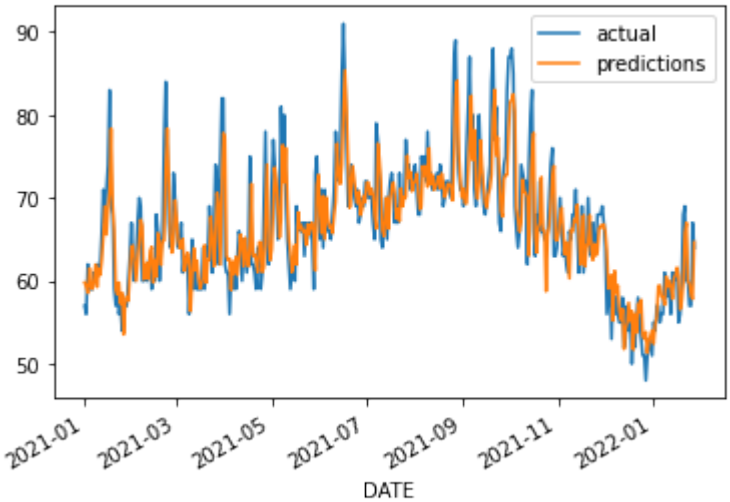
Out[545]:

DATE	actual	predictions
2021-01-01	57.0	59.806024
2021-01-02	56.0	59.310181
2021-01-03	62.0	58.538685
2021-01-04	59.0	61.531814
2021-01-05	59.0	59.444266
...	...	...
2022-01-23	60.0	59.985714
2022-01-24	57.0	59.626333
2022-01-25	57.0	58.181680
2022-01-26	67.0	57.822299
2022-01-27	64.0	64.674302

391 rows × 2 columns

In [546]:
```
combined.plot()
```

Out[546]: <AxesSubplot:xlabel='DATE'>



In [547]:
```
reg.coef_
```

Out[547]:
```
array([-2.20730384,  0.72113834,  0.17969047])
```

In [548]:
```
core_weather["month_max"] = core_weather["temp_max"].rolling(30).mean()

core_weather["month_day_max"] = core_weather["month_max"] / core_weather["temp_

core_weather["max_min"] = core_weather["temp_max"] / core_weather["temp_min"]
```

In [549]:
```
core_weather = core_weather.iloc[30:,:].copy()
```

In [550]:
```
def create_predictions(predictors, core_weather, reg):
    train = core_weather.loc[:"2020-12-31"]
    test = core_weather.loc["2021-01-01":]

    reg.fit(train[predictors], train["target"])
    predictions = reg.predict(test[predictors])

    error = mean_squared_error(test["target"], predictions)

    combined = pd.concat([test["target"], pd.Series(predictions, index=test.inc
    combined.columns = ["actual", "predictions"]
    return error, combined
```

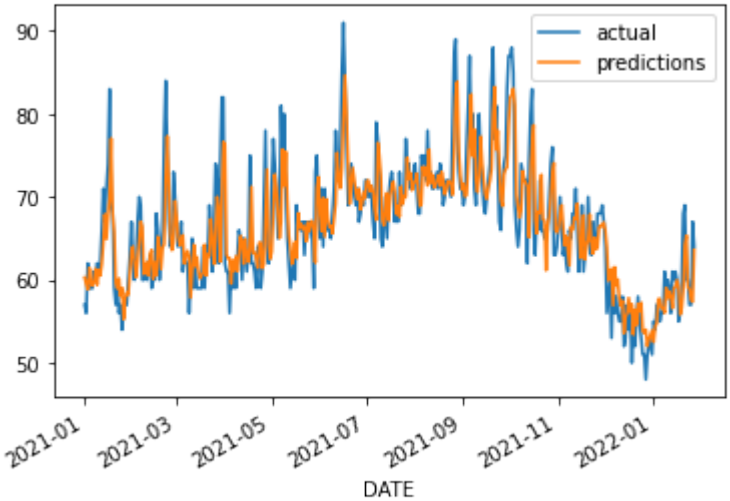
In [551]:
```
predictors = ["precip", "temp_max", "temp_min", "month_day_max", "max_min"]

error, combined = create_predictions(predictors, core_weather, reg)
error
```

Out[551]:
```
20.170663808991087
```

In [552]:
```
combined.plot()
```

Out[552]:
```
<AxesSubplot:xlabel='DATE'>
```



In [553]:
```
core_weather["monthly_avg"] = core_weather["temp_max"].groupby(core_weather.inc
core_weather["day_of_year_avg"] = core_weather["temp_max"].groupby(core_weather
```

In [554]:
```
error, combined = create_predictions(predictors + ["monthly_avg", "day_of_year_
error
```

Out[554]:
```
19.375850526432696
```

In [555]:
```
reg.coef_
```

Out[555]:
```
array([-1.07706522,  0.69350145,  0.04696919,  4.78060588,  0.07003167,
        0.16384976,  0.08581002])
```

In [556]:
```
core_weather.corr()["target"]
```

Out[556]:
```
precip            -0.205413
temp_max           0.821650
temp_min           0.596016
target             1.000000
month_max          0.686842
month_day_max     -0.421537
max_min            0.045228
monthly_avg        0.689805
day_of_year_avg    0.712334
Name: target, dtype: float64
```

In [557]:
```
combined["diff"] = (combined["actual"] - combined["predictions"]).abs()
```

In [558]:
```
combined.sort_values("diff", ascending=False).head(10)
```

Out[558]:

DATE	actual	predictions	diff
2021-01-17	83.0	68.433744	14.566256
2021-04-01	62.0	75.713379	13.713379
2021-05-07	81.0	67.678091	13.321909
2021-02-21	77.0	64.141065	12.858935
2021-10-16	66.0	78.707594	12.707594
2021-02-22	84.0	71.354231	12.645769
2021-03-30	82.0	69.994973	12.005027
2021-07-07	79.0	67.323738	11.676262
2021-03-29	74.0	62.502014	11.497986
2021-10-04	69.0	80.384267	11.384267

```
# Next steps

* Predict weather for entire next week versus a single day
* Try data from multiple close-by weather stations over a region
* Use more of the predictors in the data file
* Create more predictors (ratios, daily averages, etc)
* Try different algorithms
```

```
* Setup backtesting to make predictions for all years (versus just 2021)
```

In [ ]:

```

```