```
Start coding or generate with AI.
```

code done and debugged by ROKON

```
import pandas as pd

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

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

```
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
```

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

```
core_weather.apply(pd.isnull).sum()
```

```
precip         281
snow          5479
snow_depth    5355
temp_max         9
temp_min        10
dtype: int64
```

```
core_weather["snow"].value_counts()
```

```
0.0    11379
1.0        1
Name: snow, dtype: int64
```

```
core_weather["snow_depth"].value_counts()
```

```
0.0    11504
Name: snow_depth, dtype: int64
```

```
del core_weather["snow"]
```

```
del core_weather["snow_depth"]
```

```
core_weather[pd.isnull(core_weather["precip"])]
```

DATE	precip	snow_depth	temp_max	temp_min
1983-10-29	NaN	0.0	67.0	57.0
1983-10-30	NaN	0.0	70.0	63.0
1983-10-31	NaN	0.0	69.0	61.0
1983-11-12	NaN	0.0	63.0	55.0
1983-11-13	NaN	0.0	60.0	50.0
...	...	...	...	...
2013-12-15	NaN	NaN	58.0	33.0
2016-05-01	NaN	NaN	80.0	55.0
2016-05-02	NaN	NaN	68.0	53.0
2016-05-08	NaN	NaN	67.0	56.0
2017-10-28	NaN	NaN	68.0	50.0

281 rows × 4 columns

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

```
precip         NaN
snow_depth     NaN
temp_max      58.0
temp_min      33.0
Name: 2013-12-15, dtype: float64
```

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

```
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
```

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

```
core_weather.apply(pd.isnull).sum()
```

```
precip       0
temp_max     9
temp_min    10
dtype: int64
```

```
core_weather[pd.isnull(core_weather["temp_min"])]
```

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

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

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

```
core_weather = core_weather.fillna(method="ffill")
```

```
core_weather.apply(pd.isnull).sum()
```

```
precip      0
temp_max    0
temp_min    0
dtype: int64
```

```
core_weather.apply(lambda x: (x == 9999).sum())
```

```
precip      0
temp_max    0
temp_min    0
dtype: int64
```

```
core_weather.dtypes
```

```
precip      float64
temp_max    float64
temp_min    float64
dtype: object
```

```
core_weather.index
```

```
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)
```

```
core_weather.index = pd.to_datetime(core_weather.index)
```

```
core_weather.index
```

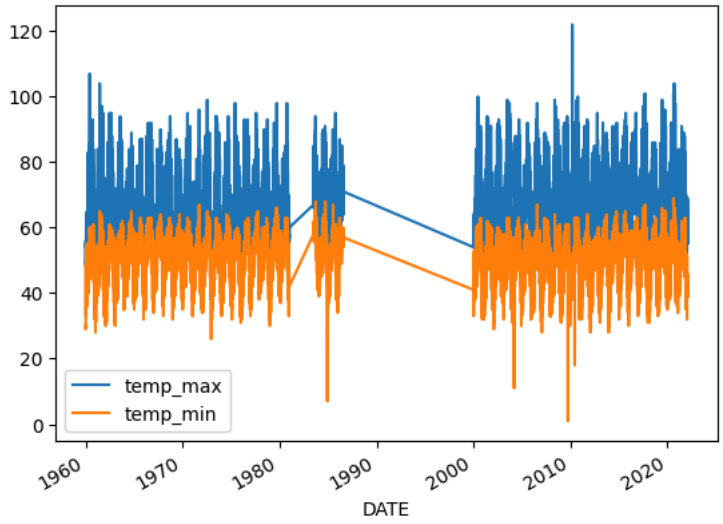
```
DatetimeIndex(['1960-01-01', '1960-01-02', '1960-01-03', '1960-01-04',
               '1960-01-05', '1960-01-06', '1960-01-07', '1960-01-08',
               '1960-01-09', '1960-01-10',
               ...
               '2022-01-19', '2022-01-20', '2022-01-21', '2022-01-22',
               '2022-01-23',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

```
core_weather.index.year
```

```
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)
```

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

```
<Axes: xlabel='DATE'>
```

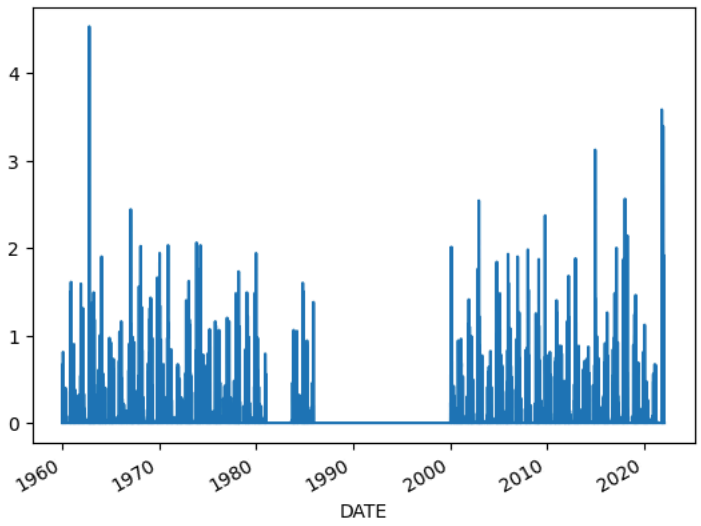



```
core_weather.index.year.value_counts().sort_index()
```

```
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
```

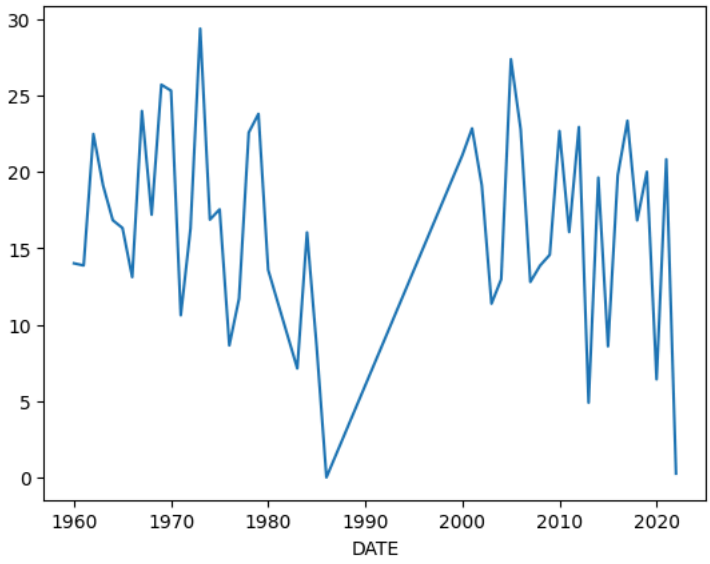
```
core_weather["precip"].plot()
```

```
<Axes: xlabel='DATE'>
```




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

```
<Axes: xlabel='DATE'>
```




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

```
core_weather
```

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

Next steps: Generate code with core_weather View recommended plots

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

```
core_weather
```

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

Next steps: Generate code with core_weather View recommended plots

```
from sklearn.linear_model import Ridge

reg = Ridge(alpha=.1)

predictors = ["precip", "temp_max", "temp_min"]

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

```
train
```

DATE	precip	temp_max	temp_min	target
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

Next steps: Generate code with train View recommended plots

```
test
```

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
...	...	...	...	...
2022-01-22	0.00	69.0	44.0	60.0
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

390 rows × 4 columns

Next steps: Generate code with `test` View recommended plots

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

```
Ridge
Ridge(alpha=0.1)
```

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

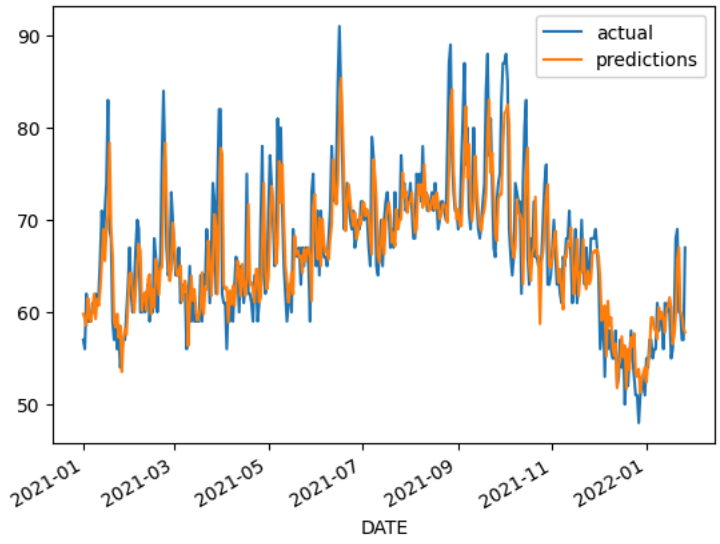
```
from sklearn.metrics import mean_squared_error

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

```
20.612222357537078
```

```
combined = pd.concat([test["target"], pd.Series(predictions, index=test.index)], axis=1)
combined.columns = ["actual", "predictions"]
```

```
combined.plot()
```

```
<Axes: xlabel='DATE'>
```




```
reg.coef_
```

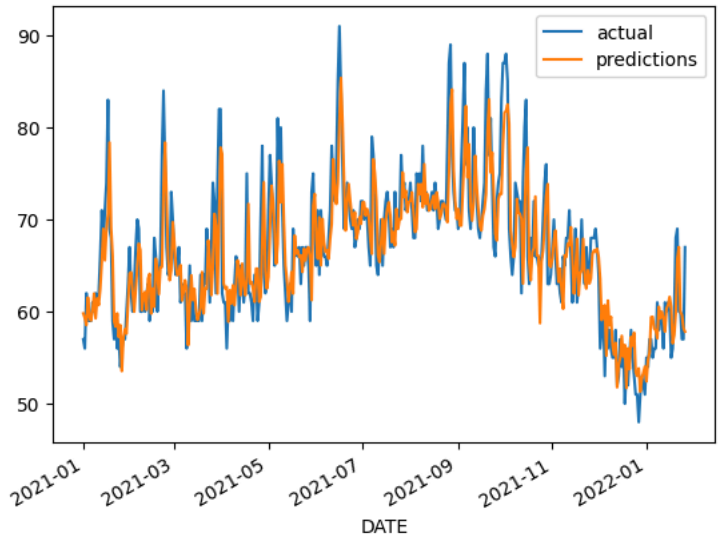
```
array([-2.20730384,  0.72113834,  0.17969047])
```

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

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

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


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


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.index)], axis=1)
    combined.columns = ["actual", "predictions"]
    return error, combined


!pip install autograd
from autograd import grad
def create_predictions(predictors, core_weather, reg):

  predictors = ["precip", "temp_max", "temp_min", "month_day_max", "max_min"]

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

```
Requirement already satisfied: autograd in /usr/local/lib/python3.10/dist-packages (1.6.2)
Requirement already satisfied: numpy>=1.12 in /usr/local/lib/python3.10/dist-packages (from autograd) (1.25.2)
Requirement already satisfied: future>=0.15.2 in /usr/local/lib/python3.10/dist-packages (from autograd) (0.18.3)
```

```
combined.plot()
```

```
<Axes: xlabel='DATE'>
```




```
core_weather["monthly_avg"] = core_weather["temp_max"].groupby(core_weather.index.month).apply(lambda x: x.expanding(1).mea
core_weather["day_of_year_avg"] = core_weather["temp_max"].groupby(core_weather.index.day_of_year).apply(lambda x: x.expand
```

```
<ipython-input-58-33e63cccf9b3>:1: FutureWarning: Not prepending group keys to the result index of transform-like apply.
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  core_weather["monthly_avg"] = core_weather["temp_max"].groupby(core_weather.index.month).apply(lambda x: x.expanding(1
<ipython-input-58-33e63cccf9b3>:2: FutureWarning: Not prepending group keys to the result index of transform-like apply.
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  core_weather["day_of_year_avg"] = core_weather["temp_max"].groupby(core_weather.index.day_of_year).apply(lambda x: x.e
```

```
reg.coef_
```

```
array([-2.20730384,  0.72113834,  0.17969047])
```

```
core_weather.corr()["target"]
```

```
precip            -0.205417
temp_max           0.821653
temp_min           0.596047
target             1.000000
month_max          0.686679
month_day_max     -0.422259
max_min            0.045235
monthly_avg        0.689826
day_of_year_avg    0.712348
Name: target, dtype: float64
```

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

```
combined.sort_values("diff", ascending=False).head(10)
```

DATE	actual	predictions	diff
2021-04-01	62.0	77.108591	15.108591
2021-05-07	81.0	67.372035	13.627965
2021-07-07	79.0	66.286763	12.713237
2021-01-17	83.0	70.620722	12.379278
2021-02-21	77.0	64.851616	12.148384
2021-03-29	74.0	61.967062	12.032938
2021-08-26	87.0	75.122490	11.877510