Weather Forecasting using Machine Learning

COEN 240: Machine Learning

Submitted by:

Arpita Verma (W1632653)
Vishakha Kulkarni (W1632897)

Table of Contents

Abstract 3

Introduction 3

History 4

Why Weather Forecasting is required 4

Weather Phenomenon 5

Machine Learning Modeling 6

Data Collection 6

Data Cleaning/Preparation/Preprocessing 8

Data Processing 8

Exploratory Data Analysis 9

Data Split 9

Model Selection 10

Model Training and Modifications 11

Feature Engineering and Model Evaluation 11

Neural Network Model 13

Training Model 13

Model Evaluation 14

Conclusion 15

References 16

Abstract

Weather is one of the most important environmental restrictions we face in multitudinous aspects of our life on this earth. To cover ourselves against anomalous situations, it must be suitable to prognosticate climate conditions analogous as temperature, downfall, humidity, and so on. The purpose of Weather prediction is to give information that people, and associations can use to reduce losses and enhance societal benefits, including protection of life and property, public health, and safety. And this was our motive to apply Weather Prediction using Machine Learning's Linear Regression Model and Deep Neural Network to predict the maximum temperature of coming day in San Jose municipality by exercising the large set of data openly available from the NOAA data center from the time 1988 to 2022. We used a series of feature engineering techniques, model evaluations at each step and analyzed the results. As this was a time series historical data, a regressor was suitable for the modelling and thus we used Linear Regression and Deep neural Network Tensor flow Regressor. The results from each model were evaluated to understand the difference in prediction.

Introduction

Climate conditions are constantly changing each across the world. Precision projections are critical in moment's terrain. From husbandry to sedulity, from trip to quotidian trip, we calculate heavily on the weather that day, in all we do. Given that the entire world is witnessing ongoing environmental change and its implications, it's critical to precisely predict to give simple and smooth everyday operations, as well as quick and indefectible severity.

General gyration models are used to read the climate. multitudinous supposition- predicated analyses of circumstances have supplied us with the prediction models, analogous as numerical prediction, climatic variability evaluation, surveillance radar, warming projections, and so on.

Scientists are still trying to overcome the limits of computer models in order to meliorate cast delicacy rates using new technologies that give intelligence to machines. To make the system more intelligent, we've created a disquisition platform that includes Artificial Neural Networks, Machine knowledge, and Rule-predicated ways, all of which give ample provocation for studying these circumstances and cast.

The wisdom of machine knowledge can be simplified by using prediction as an illustration. This enables access to a large set of data by establishing APIs for piercing available data from meteorological institutes and other stations. Weather data is a commodity that utmost people are familiar with in their quotidian lives, but it's also critical for energy systems, flood tide drift auguring, and other operations. There are several good physical-predicated meteorological models available, which makes comparing the quality of machine knowledge models simple. Weather report services for a large region or municipality can readily give large-scale information

History

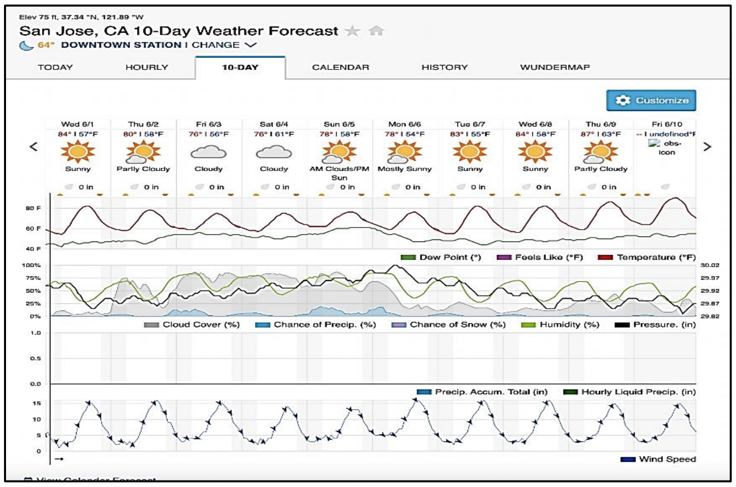
Observed patterns of events, constantly known as pattern recognition, were used in ancient styles. It was discovered, for illustration, that if the evening was particularly sanguine, the following day was generally sunny. This knowledge was passed down through the times, performing in downfall lore. still, not all of these prognostications are accurate, and multitudinous of them have been proved to be unreliable after being vanquished to rigorous statistical testing.

Advances in the understanding of atmospheric medicines did not lead to the establishment of modern numerical calculations until the twentieth century. After discovering notes and derivations he worked on as an ambulance automobilist during World War I, English physicist Lewis Fry Richardson published" Weather Prediction by Numerical Process" in 1922. Small terms in the prognostic fluid dynamics equations driving air flux could be ignored, and a finite differencing system in time and space could be designed to allow numerical prophecy results to be attained, he explained in the paper.

On August 1, 1861, The Times published the 1st weather forecast followed by the first maps subsequently that time. The first marine climate was broadcast over the airwaves by the Met Office in 1911.

During the nineteenth and twentieth centuries, more data were getting available for observation-predicated weather. Also, these data were reused and made available for constructing maps or insertion into computer models for this. Presently, radiosondes are launched every 12 hours from hundreds of ground stations all over the world.

Why Weather Forecasting is required


Elev 75 ft, 37.34 °N, 121.89 °W
San Jose, CA 10-Day Weather Forecast
64° DOWNTOWN STATION I CHANGE
TODAY
HOURLY
10-DAY
CALENDAR
HISTORY
WUNDERMAP
Customize
Wed 6/1 84° I 57°F Sunny 0 in
Thu 6/2 80° I 58°F Partly Cloudy 0 in
Fri 6/3 76° I 56°F Cloudy 0 in
Sat 6/4 76° I 61°F Cloudy 0 in
Sun 6/5 78° I 58°F AM Clouds/PM Sun 0 in
Mon 6/6 78° I 54°F Mostly Sunny 0 in
Tue 6/7 83° I 55°F Sunny 0 in
Wed 6/8 84° I 58°F Sunny 0 in
Thu 6/9 87° I 63°F Partly Cloudy 0 in
Fri 6/10 -- I undefined°F obs-icon
Dew Point (°)
Feels Like (°F)
Temperature (°F)
Cloud Cover (%)
Chance of Precip. (%)
Chance of Snow (%)
Humidity (%)
Pressure. (in)
Precip. Accum. Total (in)
Hourly Liquid Precip. (in)
Wind Speed

There are several reasons why this is truly important. It's a product of wisdom that impacts the lives of multitudinous people. The advantages of better data gathering, and sharing are substantial. With a cost-benefit rate of 126, the World Bank estimates that awaited advances in models might deliver roughly$ 5 billion in periodic benefits on top of the being$ 160 billion in periodic benefits from climate prediction. The following is a list of various reasons why this is important

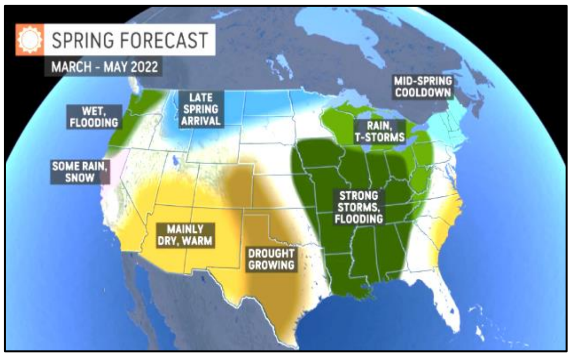
- The weather prediction can help us decide what type of clothes to put on(warm, cold, windy, stormy)
- Helps businesses and people plan for power product and how important power to use(i.e. power companies, where to set thermostat)
- Helps people prepare if they need to take spare gear to prepare (i.e. marquee, raincoat, sunscreen)
- Helps people plan out-of-door exertion (i.e. to see if rain/ storms/ cold waves will impact out- of-door event)
- Helps curious people to know what can be anticipated (i.e. a snow on the way, severe storms)
- Helps businesses plan for transportation hazards that can affect from this (i.e. fog, snow, ice, storms, murk as it relates to driving and flying for illustration)
- Helps people with health- related issues to plan the day (i.e. aversions, asthma, heat stress) Helps businesses and people plan for the severe downfall and other hazards(lightning, hail, tornadoes, hurricanes, ice storms)
- Helps farmers and gardeners plan for crop irrigation and protection (irrigation scheduling, snap protection)

Weather Phenomenon

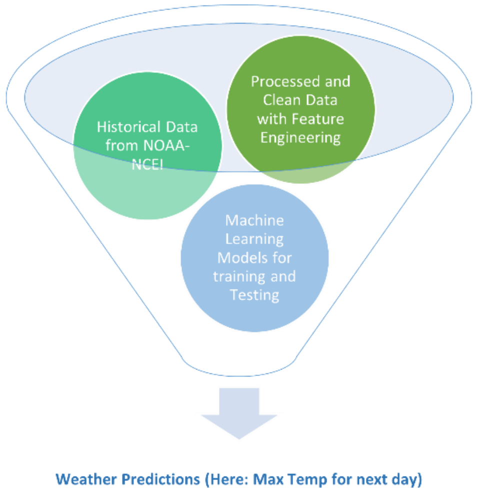
Identification and prediction of the below extreme weather phenomena:

For weather forecast, the major phenomena usually analyzed are:

- Storms
- Hail Swath
- Lightning
- Precipitation
- Tornado
- Flash Flood
- Damaging Wind
- Wind Speed
- Maximum and Minimum temperature



Machine Learning Modeling


Historical Data from NOAA-NCEI
Processed and Clean Data with Feature Engineering
Machine Learning Models for training and Testing
Weather Predictions (Here: Max Temp for next day)

Our model collects historical weather data that includes a variety of significant parameters that influence weather change, such as temperature, including maximum and lowest temperatures, precipitation, storms, floods, windspeed.

In our proposed model, the acquired dataset is divided into parts that are useful to the machine learning model and parts that aren't. After that, the dataset goes through data preprocessing, which involves passing the data through a procedure that replaces missing and error values in the dataset with mean values or the most often occurring value in that field. Another option is to ignore certain values and replace them with EAN before continuing with the rest of the tasks.

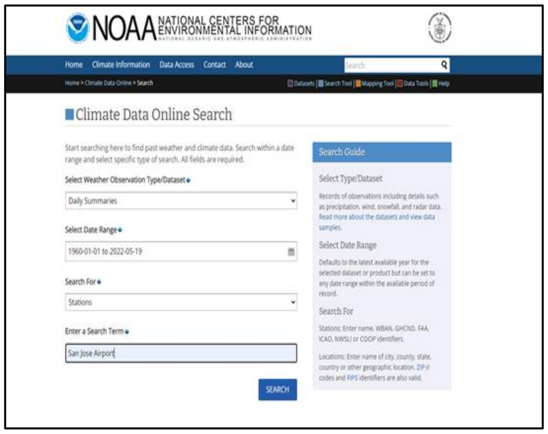
In data preprocessing, we did the initial analysis of the data to understand the type of information through Exploratory Data Analysis (EDA). To prepare the data for the analysis, we cleaned the data and then applied Feature Engineering to pick and add attributes that could be used for the training. We divided the dataset into training and testing sets.

Following the data preprocessing, the cleaned dataset is divided into two parts: the training set and the test set. The training set is used to teach the machine learning model how to compute the results, while the testing set is used to locate the results, compare the real and calculated values, and utilize the error value as a benchmark to teach the machine learning model even more.

After this, we selected Machine Learning Model to find out the maximum temperature of the next day using the Linear Regression model initially. Later we used the DNN regressor to get accurate predictions.

Data Collection

We collected real weather data for the city of San Jose from *ncei.noaa.gov* (screenshot as shown here). It returns the weather data with many parameters like Precipitation, Snowfall, Maximum and Minimum Temperature, Wind speed, Average cloudiness etc.


NOAA NATIONAL CENTERS FOR ENVIRONMENTAL INFORMATION
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
Home
Climate Information
Data Access
Contact
About
Search
Home > Climate Data Online > Search
Datasets | Search Tool | Mapping Tool | Data Tools | Help
SEARCH

Climate Data Online Search

Start searching here to find past weather and climate data. Search within a date range and select specific type of search. All fields are required.

Select Weather Observation Type/Dataset: Daily Summaries

Select Date Range: 1960-01-01 to 2022-05-19

Search For: Stations

Enter a Search Term: San Jose Airport

Search Guide

Select Type/Dataset

Records of observations including details such as precipitation, wind, snowfall, and radar data. Read more about the datasets and view data samples.

Select Date Range

Defaults to the latest available year for the selected dataset or product but can be set to any date range within the available period of record.

Search For

Stations: Enter name, WBAN, GHCND, FAA, ICAO, NWSLI or COOP identifiers.

Locations: Enter name of city, county, state, country or other geographic location. ZIP # codes and FIPS identifiers are also valid.

STATION	NAME	DATE	AWND	FMTM	PGTM	PRCP	SNOW	SNWD	TAVG	TMAX	TMIN	TSUN	WDF2	WDF5	WSF2	WSF5
USW00023293	SAN JOSE, CA US	7/4/1998	8.5	1740	1739	0			68	79	56		320	320	16.1	17.9

Among these, as we plan to predict maximum temperature of the next day based on the historical data, we selected some of the core values.

The documentation provided with the dataset mentioned Precipitation, Snowfall and temperature being core value for weather prediction and thus we selected those. Also, as we analyzed the given data, we found that Windspeed could also be a factor in determining the temperature.

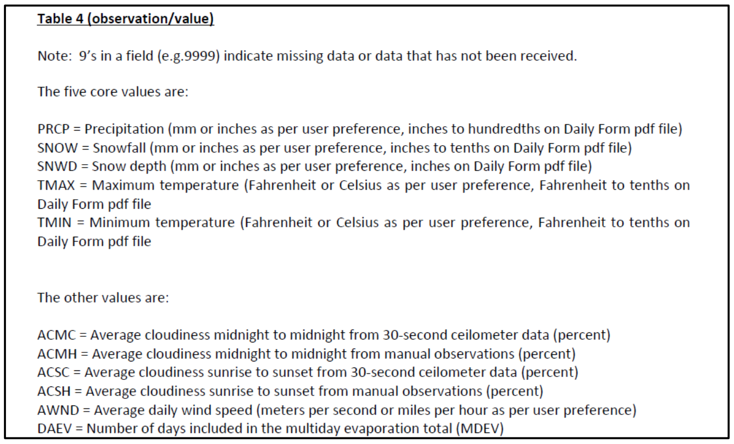


Table 4 (observation/value)

Note: 9's in a field (e.g.9999) indicate missing data or data that has not been received.

The five core values are:

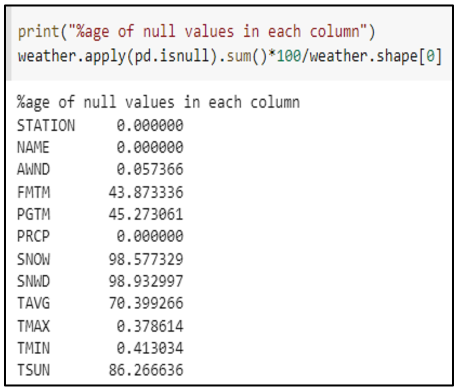
PRCP = Precipitation (mm or inches as per user preference, inches to hundredths on Daily Form pdf file)
SNOW = Snowfall (mm or inches as per user preference, inches to tenths on Daily Form pdf file)
SNWD = Snow depth (mm or inches as per user preference, inches on Daily Form pdf file)
TMAX = Maximum temperature (Fahrenheit or Celsius as per user preference, Fahrenheit to tenths on Daily Form pdf file
TMIN = Minimum temperature (Fahrenheit or Celsius as per user preference, Fahrenheit to tenths on Daily Form pdf file

The other values are:

ACMC = Average cloudiness midnight to midnight from 30-second ceilometer data (percent)
ACMH = Average cloudiness midnight to midnight from manual observations (percent)
ACSC = Average cloudiness sunrise to sunset from 30-second ceilometer data (percent)
ACSH = Average cloudiness sunrise to sunset from manual observations (percent)
AWND = Average daily wind speed (meters per second or miles per hour as per user preference)
DAEV = Number of days included in the multiday evaporation total (MDEV)

From the available data, we tried to first find the factors or attributes that had proper information available. We checked the percentage of Null values for each feature given and we saw that we didn't have enough values for Snowfall or Snow Depth or Gust time of wind etc.

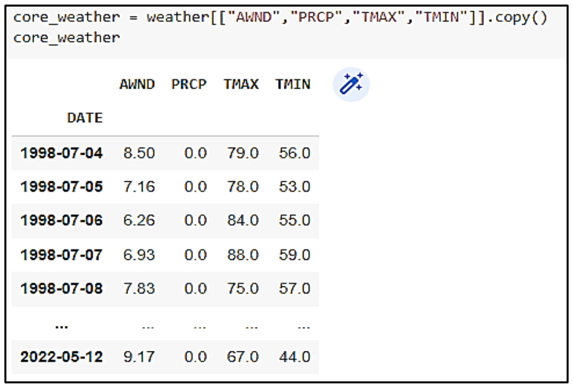
As our city for analysis is San Jose, and it rarely snows here, there was no data available for Snow for San Jose.



```
print("%age of null values in each column")
weather.apply(pd.isnull).sum()*100/weather.shape[0]
```

```
%age of null values in each column
STATION     0.000000
NAME        0.000000
AWND        0.057366
FMTM       43.873336
PGTM       45.273061
PRCP        0.000000
SNOW       98.577329
SNWD       98.932997
TAVG       70.399266
TMAX        0.378614
TMIN        0.413034
TSUN       86.266636
```

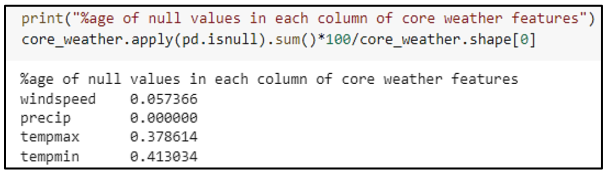
Thus, our Core Weather features are Precipitation, Wind Speed, and Maximum and Minimum Temperature.



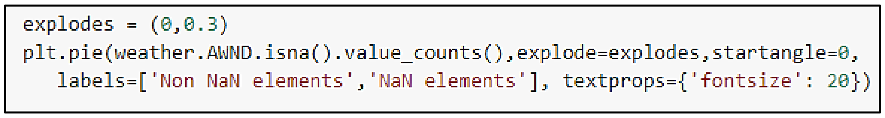
```
core_weather = weather[["AWND","PRCP","TMAX","TMIN"]].copy()
core_weather
```

	AWND	PRCP	TMAX	TMIN
DATE				
1998-07-04	8.50	0.0	79.0	56.0
1998-07-05	7.16	0.0	78.0	53.0
1998-07-06	6.26	0.0	84.0	55.0
1998-07-07	6.93	0.0	88.0	59.0
1998-07-08	7.83	0.0	75.0	57.0
...	...	...	...	...
2022-05-12	9.17	0.0	67.0	44.0

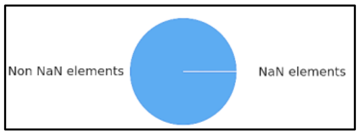
Data Cleaning/Preparation/Preprocessing

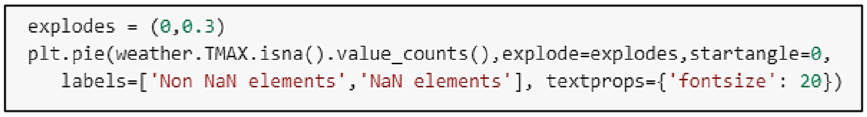


```
print("%age of null values in each column of core weather features")
core_weather.apply(pd.isnull).sum()*100/core_weather.shape[0]

%age of null values in each column of core weather features
windspeed    0.057366
precip       0.000000
tempmax      0.378614
tempmin      0.413034
```

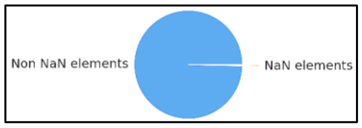


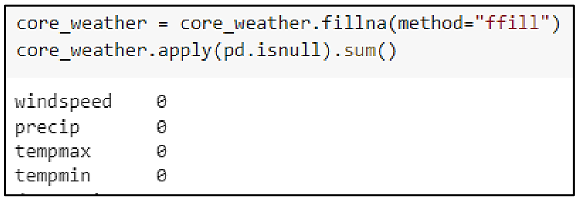
```
explodes = (0,0.3)
plt.pie(weather.AWND.isna().value_counts(),explode=explodes,startangle=0,
    labels=['Non NaN elements','NaN elements'], textprops={'fontsize': 20})
```





```
explodes = (0,0.3)
plt.pie(weather.TMAX.isna().value_counts(),explode=explodes,startangle=0,
    labels=['Non NaN elements','NaN elements'], textprops={'fontsize': 20})
```

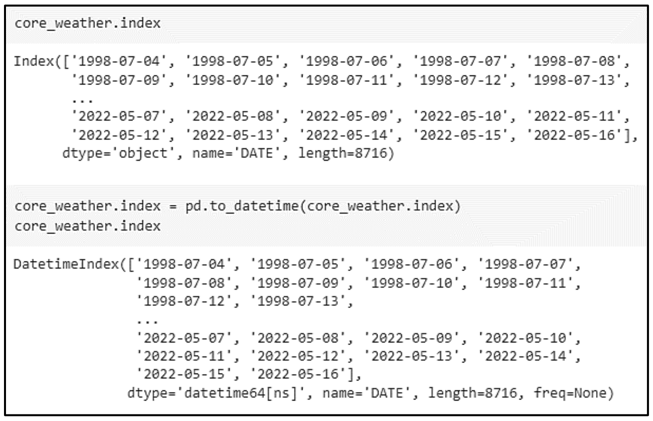




```
core_weather = core_weather.fillna(method="ffill")
core_weather.apply(pd.isnull).sum()

windspeed    0
precip       0
tempmax      0
tempmin      0
```

Data Processing

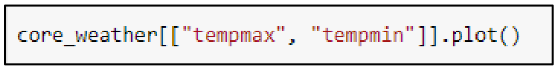
The date was fixed as the index for this dataset while reading the file. We converted the object type of the date in a datetime format, so that the data can be accessed easily using this index.



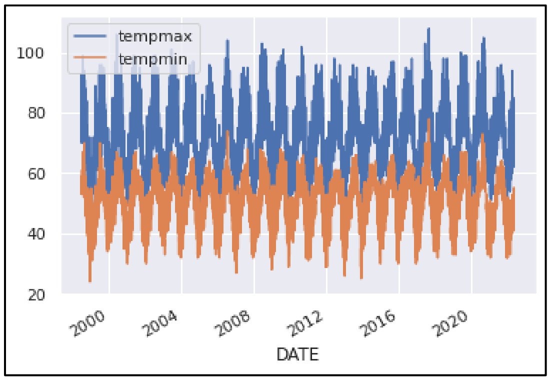
```
core_weather.index

Index(['1998-07-04', '1998-07-05', '1998-07-06', '1998-07-07', '1998-07-08',
       '1998-07-09', '1998-07-10', '1998-07-11', '1998-07-12', '1998-07-13',
       ...
       '2022-05-07', '2022-05-08', '2022-05-09', '2022-05-10', '2022-05-11',
       '2022-05-12', '2022-05-13', '2022-05-14', '2022-05-15', '2022-05-16'],
      dtype='object', name='DATE', length=8716)

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

DatetimeIndex(['1998-07-04', '1998-07-05', '1998-07-06', '1998-07-07',
               '1998-07-08', '1998-07-09', '1998-07-10', '1998-07-11',
               '1998-07-12', '1998-07-13',
               ...
               '2022-05-07', '2022-05-08', '2022-05-09', '2022-05-10',
               '2022-05-11', '2022-05-12', '2022-05-13', '2022-05-14',
               '2022-05-15', '2022-05-16'],
              dtype='datetime64[ns]', name='DATE', length=8716, freq=None)
```

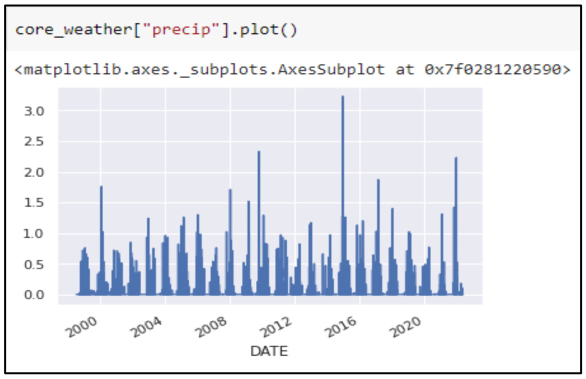
Exploratory Data Analysis

Some of the EDA tools like plot were used to get the visual idea about the data.



```
core_weather[["tempmax", "tempmin"]].plot()
```

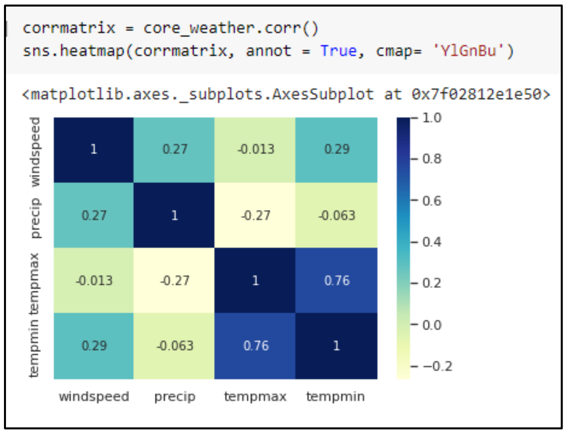




```
core_weather["precip"].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0281220590>
```

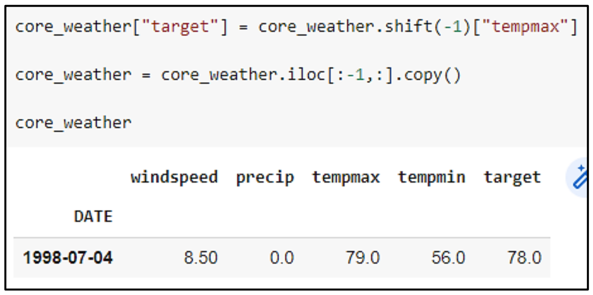
Using Heatmap, we also tried to find the correlation among the core_weather features. We found obvious correlations between the temperature and the precipitation – showing a negative correlation. Thus, when the precipitation is high, the maximum temperature of that day decreases.



```
corrmatrix = core_weather.corr()
sns.heatmap(corrmatrix, annot = True, cmap= 'YlGnBu')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02812e1e50>
```

<u>**Target**</u>: Our problem statement here is to predict the maximum temperature for the next day based on the historical values of the core weather features. Thus, we add another column in the dataset – Target. It is the maximum temperature of the next day throughout the dataset.

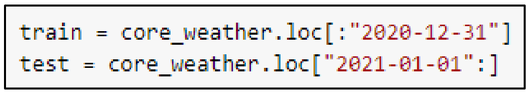


```
core_weather["target"] = core_weather.shift(-1)["tempmax"]

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

core_weather
```

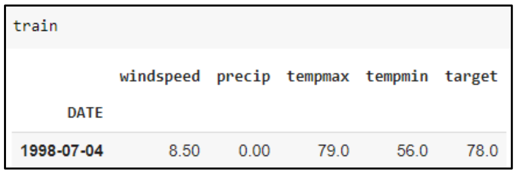
	windspeed	precip	tempmax	tempmin	target
DATE					
1998-07-04	8.50	0.0	79.0	56.0	78.0

Data Split

We are trying to predict the future value (next day Maximum Temperature here) based on the historical value. Thus, we fixed the training data from the initial day to the year 2020 Dec i.e. 90% of the total dataset and the training data is almost the current values for the year 2021 and 2022 to predict the next day maximum temperature for 2022.

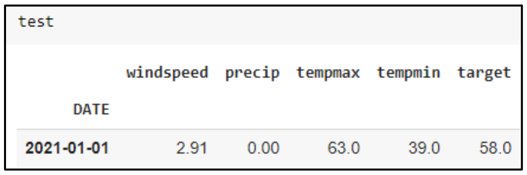


```
train = core_weather.loc[:"2020-12-31"]
test = core_weather.loc["2021-01-01":]
```



train

DATE	windspeed	precip	tempmax	tempmin	target
1998-07-04	8.50	0.00	79.0	56.0	78.0

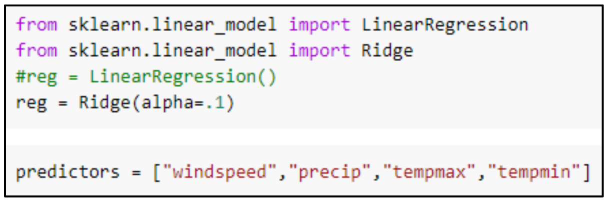


test

DATE	windspeed	precip	tempmax	tempmin	target
2021-01-01	2.91	0.00	63.0	39.0	58.0

Model Selection

As we have a time series historical data, we selected the Linear Regression model with Ridge regression adjustment to avoid Overfitting.



```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
#reg = LinearRegression()
reg = Ridge(alpha=.1)

predictors = ["windspeed","precip","tempmax","tempmin"]
```

Linear Regression: The general formula for the linear regression cab be written as

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_{(p-n)} x_{(p-n)} + E$$

where,

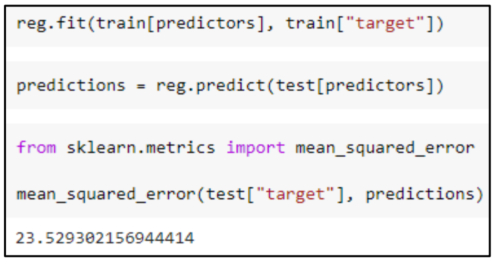
$\hat{y}$ is the predicted value (Target)

x_j are the features or attributes used to analyze the prediction (predictors)

β_j are the regression coefficients

E is the error associated with the difference between the target and actual

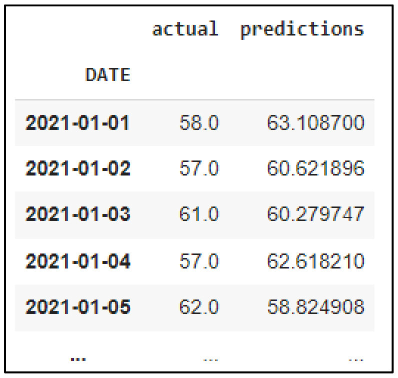
Model Training and Modifications



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

predictions = reg.predict(test[predictors])

from sklearn.metrics import mean_squared_error

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

```
23.529302156944414
```

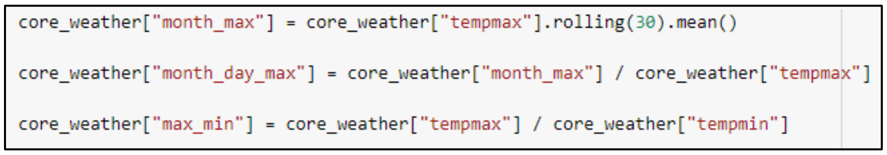
We see here that the mean square error is very high (23.52) considering the predictors as the core weather features defined before.



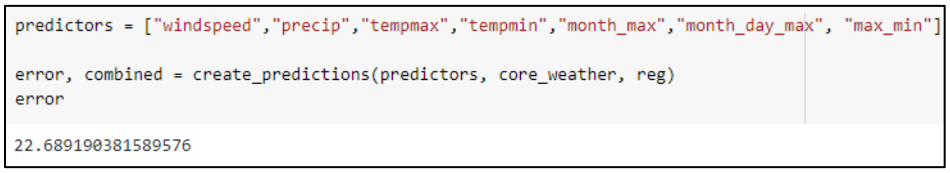
	actual	predictions
DATE		
2021-01-01	58.0	63.108700
2021-01-02	57.0	60.621896
2021-01-03	61.0	60.279747
2021-01-04	57.0	62.618210
2021-01-05	62.0	58.824908
...	...	...

Feature Engineering and Model Evaluation

To improve the predictions, we now try to add some features derived from the core weather values, like monthly maximum temperature, max-min temperature etc.



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

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

core_weather["max_min"] = core_weather["tempmax"] / core_weather["tempmin"]
```

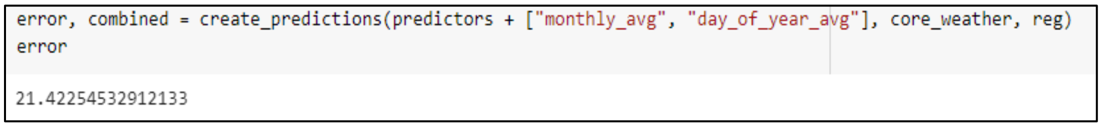


```
predictors = ["windspeed","precip","tempmax","tempmin","month_max","month_day_max", "max_min"]

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

```
22.689190381589576
```

The error reduced a little by adding the above features.

We also added a few more features like day of the year etc.



```
error, combined = create_predictions(predictors + ["monthly_avg", "day_of_year_avg"], core_weather, reg)
error
```

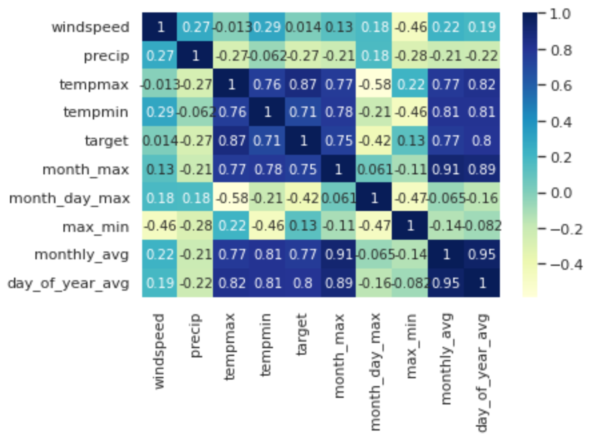
```
21.42254532912133
```

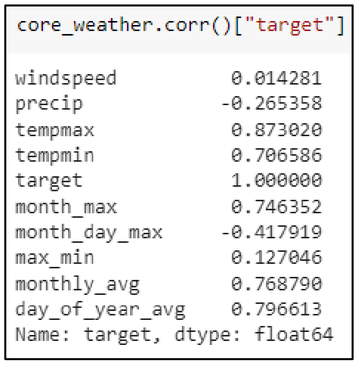
We see that the error is reducing but still needs a lot of improvement.

Also, we check the correlation between the predictors and the Target and among the predictors, that can also lead to more error or not so good predictions.

```
corrmatrix = core_weather.corr()
sns.heatmap(corrmatrix, annot = True, cmap= 'YlGnBu')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f027b8e7710>
```





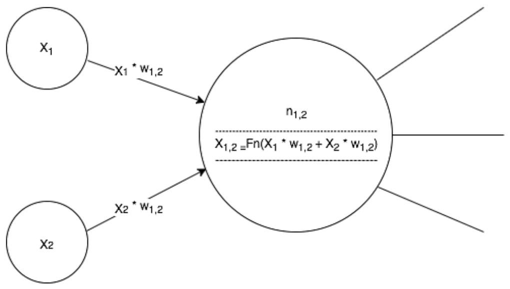
```
core_weather.corr()["target"]

windspeed          0.014281
precip            -0.265358
tempmax            0.873020
tempmin            0.706586
target             1.000000
month_max          0.746352
month_day_max     -0.417919
max_min            0.127046
monthly_avg        0.768790
day_of_year_avg    0.796613
Name: target, dtype: float64
```

Neural Network Model

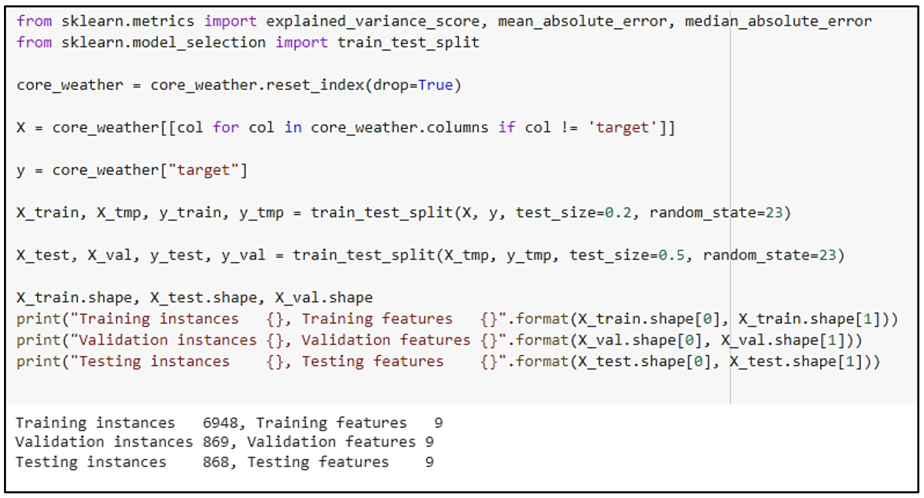
As we understood that this prediction cannot be fully accurate with a linear model, we planned to use Neural Networks.

The type of predictions being made are numerical real values, which means we are dealing with regression prediction algorithms. And thus, we used the DNN Regressor model with the Tensor flow Estimator.

Our neural network regressor is iterating over the training data feeding in feature values, calculating the cost function (using SSE) and making adjustments to the weights in a way that minimizes the cost function.



From the core weather data available as above, the dataset was split into training, validation, and test data.



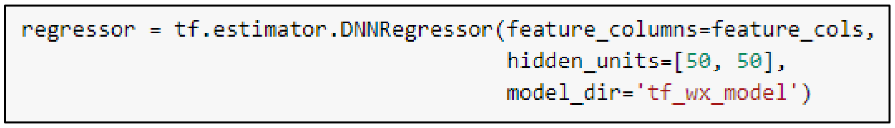
```
from sklearn.metrics import explained_variance_score, mean_absolute_error, median_absolute_error
from sklearn.model_selection import train_test_split

core_weather = core_weather.reset_index(drop=True)

X = core_weather[[col for col in core_weather.columns if col != 'target']]

y = core_weather["target"]

X_train, X_tmp, y_train, y_tmp = train_test_split(X, y, test_size=0.2, random_state=23)

X_test, X_val, y_test, y_val = train_test_split(X_tmp, y_tmp, test_size=0.5, random_state=23)

X_train.shape, X_test.shape, X_val.shape
print("Training instances   {}, Training features   {}".format(X_train.shape[0], X_train.shape[1]))
print("Validation instances {}, Validation features {}".format(X_val.shape[0], X_val.shape[1]))
print("Testing instances    {}, Testing features    {}".format(X_test.shape[0], X_test.shape[1]))
```

```
Training instances   6948, Training features   9
Validation instances 869, Validation features 9
Testing instances    868, Testing features    9
```

Training Model

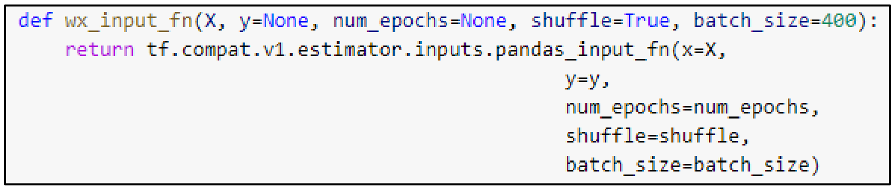
We used Tensor Flow Estimator for Deep Neural Network Regressor.

Here we have instantiated DNNRegressor class and storing it in regressor.



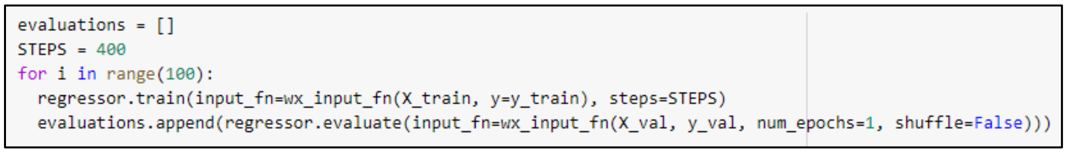
```
regressor = tf.estimator.DNNRegressor(feature_columns=feature_cols,
                                      hidden_units=[50, 50],
                                      model_dir='tf_wx_model')
```

Then we define an input function wx_input_fn() to be used whenever the inputs are to be fed to the neural network.



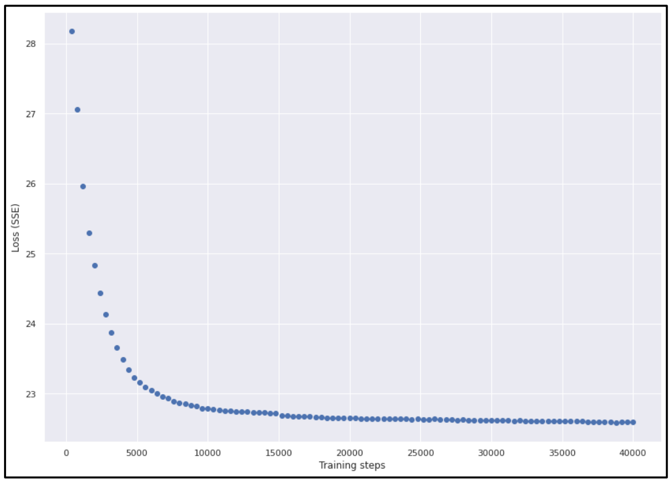
```python
def wx_input_fn(X, y=None, num_epochs=None, shuffle=True, batch_size=400):
    return tf.compat.v1.estimator.inputs.pandas_input_fn(x=X,
                                                         y=y,
                                                         num_epochs=num_epochs,
                                                         shuffle=shuffle,
                                                         batch_size=batch_size)
```

We now the train the neural network 100 times and 400 instances fed every time, thus batch size being 400 instances. The validation inputs are used here for evaluations.

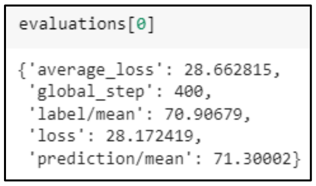


```python
evaluations = []
STEPS = 400
for i in range(100):
  regressor.train(input_fn=wx_input_fn(X_train, y=y_train), steps=STEPS)
  evaluations.append(regressor.evaluate(input_fn=wx_input_fn(X_val, y_val, num_epochs=1, shuffle=False)))
```

Model Evaluation



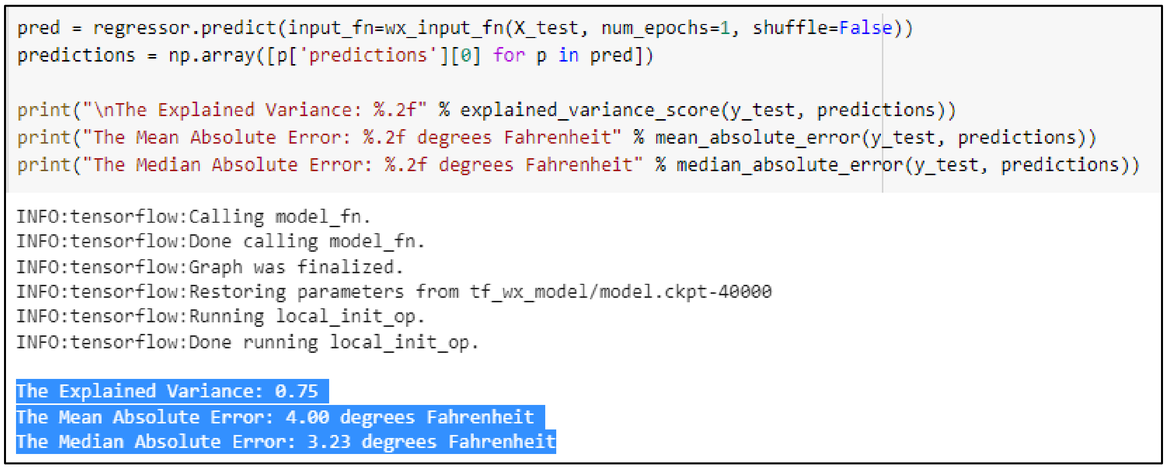
After the training and validation process, we see that the Loss function is decreasing and is almost there for several epochs. Thus, we can conclude that it has reached its minimum value and to avoid overfitting, we stop.



```
evaluations[0]
```

```
{'average_loss': 28.662815,
 'global_step': 400,
 'label/mean': 70.90679,
 'loss': 28.172419,
 'prediction/mean': 71.30002}
```

We now use the trained model to predict the values from the test instances and then find the mean absolute error. We see a huge difference in the error than the linear regression.



```python
pred = regressor.predict(input_fn=wx_input_fn(X_test, num_epochs=1, shuffle=False))
predictions = np.array([p['predictions'][0] for p in pred])

print("\nThe Explained Variance: %.2f" % explained_variance_score(y_test, predictions))
print("The Mean Absolute Error: %.2f degrees Fahrenheit" % mean_absolute_error(y_test, predictions))
print("The Median Absolute Error: %.2f degrees Fahrenheit" % median_absolute_error(y_test, predictions))
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from tf_wx_model/model.ckpt-40000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.

The Explained Variance: 0.75
The Mean Absolute Error: 4.00 degrees Fahrenheit
The Median Absolute Error: 3.23 degrees Fahrenheit
```

We see that the error or the difference in the predicted and the actual value has reduced to 4 degrees Fahrenheit.

Conclusion

By using Linear regression algorithm to predict weather.

- By using predictors : "windspeed","precip","tempmax","tempmin"
  - Mean Squared Error = 23.52
- By using predictors : "windspeed","precip","tempmax","tempmin", "month_max”, "month_day_max", "max_min"
  - Mean Squared Error = 22.68
- By using predictors : "windspeed", "precip", "tempmax", "tempmin", "month_max", "month_day_max", "max_min”, "monthly_avg", "day_of_year_avg"
  - Mean Squared Error = 21.42

By using DNN regressor (Neural Networks) from Tensor Flow to predict weather:

The Explained Variance: 0.75
The Mean Absolute Error: 4.01 degrees Fahrenheit
The Median Absolute Error: 3.25 degrees Fahrenheit

We are getting a better-predicted value of **maximum Temperature** using the DNN regressor. The difference between actual and predicted values got reduced by 80%.

References

1. https://www.youtube.com/watch?v=km95-NMT6lU&ab_channel=Dataquest
2. https://www.ncei.noaa.gov/cdo-web/search
3. https://towardsdatascience.com/weather-forecasting-with-machine-learning-using-python-55e90c346647
4. https://www.accuweather.com/en/weather-forecasts/accuweather-2022-us-spring-forecast/1132504
5. https://earthobservatory.nasa.gov/features/WxForecasting/wx2.php
6. https://doi.org/10.48550/arXiv.2008.10789
7. Singh, Shashank & Faraz, Ahmed & Nagrami, & Pillai, Aditya. (2020). WEATHER PREDICTION BY USING MACHINE LEARNING.
8. https://en.wikipedia.org/wiki/Weather_forecasting
9. https://aqua.nasa.gov/sites/default/files/references/Wx_Forecasting.pdf
10. https://givingcompass.org/article/why-we-need-better-weather-forecasts?gclid=Cj0KCQjwnNyUBhCZARIsAI9AYlG2yg6qAHyxeK_Oly6b2k-d6pYc9mJOol9x1S4G9mJWPWdNG8Oh8YAaAiynEALw_wcB
11. https://www.wunderground.com/forecast/us/ca/san-jose