Federated Learning on Solar Radiation Prediction

*Prediction of Solar Radiation on given weather data

These datasets are meteorological data from the HI-SEAS weather station from four months (September through December 2016)

- Dataset consists different features like
- Solar radiation [W/m^2]
- Temperature [F]
- Atmospheric pressure [Hg]
- Humidity [%]
- Wind speed [miles/h]
- Wind direction [degrees]

Our objective is to derive an ML model to forecast the solar radiation as a function of available features.

Dataset:

https://www.kaggle.com/code/enricobaldasso/prediction-of-solar-radiation-data/input

*Importing Libraries

```
import re         #for searching string in a given large text
import numpy as np #for scientific computing
import pandas as pd #to perform computations on series of data(dataframe)

import matplotlib.pyplot as plt #for visulaization of data
import seaborn as sns

from datetime import datetime
from dateutil.tz import *

from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression,Ridge
from sklearn.svm import SVR
import xgboost as xgb

import tensorflow as tf
```

Federated Learning on Solar Radiation Prediction

```python
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.models import Sequential
from collections import Counter

from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

import warnings
warnings.filterwarnings("ignore")
```

***Loading Dataset**

```python
df=pd.read_csv('SolarPrediction.csv')
df.head()
```

```
     UNIXTime                    Data      Time  Radiation  Temperature  \
0  1475229326  9/29/2016 12:00:00 AM  23:55:26       1.21           48
1  1475229023  9/29/2016 12:00:00 AM  23:50:23       1.21           48
2  1475228726  9/29/2016 12:00:00 AM  23:45:26       1.23           48
3  1475228421  9/29/2016 12:00:00 AM  23:40:21       1.21           48
4  1475228124  9/29/2016 12:00:00 AM  23:35:24       1.17           48

   Pressure  Humidity  WindDirection(Degrees)  Speed TimeSunRise TimeSunSet
0     30.46        59                  177.39   5.62    06:13:00   18:13:00
1     30.46        58                  176.78   3.37    06:13:00   18:13:00
2     30.46        57                  158.75   3.37    06:13:00   18:13:00
3     30.46        60                  137.71   3.37    06:13:00   18:13:00
4     30.46        62                  104.95   5.62    06:13:00   18:13:00
```

```python
#checking each feature datatype
# to correct datatype of columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32686 entries, 0 to 32685
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   UNIXTime                32686 non-null  int64
 1   Data                    32686 non-null  object
 2   Time                    32686 non-null  object
 3   Radiation               32686 non-null  float64
 4   Temperature             32686 non-null  int64
 5   Pressure                32686 non-null  float64
 6   Humidity                32686 non-null  int64
 7   WindDirection(Degrees)  32686 non-null  float64
 8   Speed                   32686 non-null  float64
 9   TimeSunRise             32686 non-null  object
 10  TimeSunSet              32686 non-null  object
dtypes: float64(4), int64(3), object(4)
memory usage: 2.7+ MB
```

Federated Learning on Solar Radiation Prediction

```
#rename columns
df=df.rename(columns={'Data':'Date'})
df.head()
```

```
     UNIXTime                   Date      Time  Radiation  Temperature  \
0  1475229326  9/29/2016 12:00:00 AM  23:55:26       1.21           48
1  1475229023  9/29/2016 12:00:00 AM  23:50:23       1.21           48
2  1475228726  9/29/2016 12:00:00 AM  23:45:26       1.23           48
3  1475228421  9/29/2016 12:00:00 AM  23:40:21       1.21           48
4  1475228124  9/29/2016 12:00:00 AM  23:35:24       1.17           48

   Pressure  Humidity  WindDirection(Degrees)  Speed TimeSunRise TimeSunSet
0     30.46        59                  177.39   5.62    06:13:00   18:13:00
1     30.46        58                  176.78   3.37    06:13:00   18:13:00
2     30.46        57                  158.75   3.37    06:13:00   18:13:00
3     30.46        60                  137.71   3.37    06:13:00   18:13:00
4     30.46        62                  104.95   5.62    06:13:00   18:13:00
```

```
#find dimension and shape of dataframe
print(df.ndim)
print(df.shape)
print(df.size)
```

```
2
(32686, 11)
359546
```

*Data Preprocessing

In this section,we make sure that data is free from duplicates, missing values ,outliers.--->Data Cleaning *Data Transformation-->converting data into desired form.* Data Normalization -->Scaling numerical data to a standard range.

```
#checking for missing values
#missing values can be handled by imputation 1.mean,median,mode strategies
2.forward,backward 3.model prediction
#from skleran.impute import SimpleImputer
df.isnull().sum()
```

```
UNIXTime                  0
Date                      0
Time                      0
Radiation                 0
Temperature               0
Pressure                  0
Humidity                  0
WindDirection(Degrees)    0
Speed                     0
TimeSunRise               0
TimeSunSet                0
dtype: int64
```

Federated Learning on Solar Radiation Prediction

```
#checking for duplicates of data
# how handle duplicates 1.remove duplicates 2.keep first,last occurance 3.mark
duplicates by adding new boolean column
df.duplicated().sum()
```

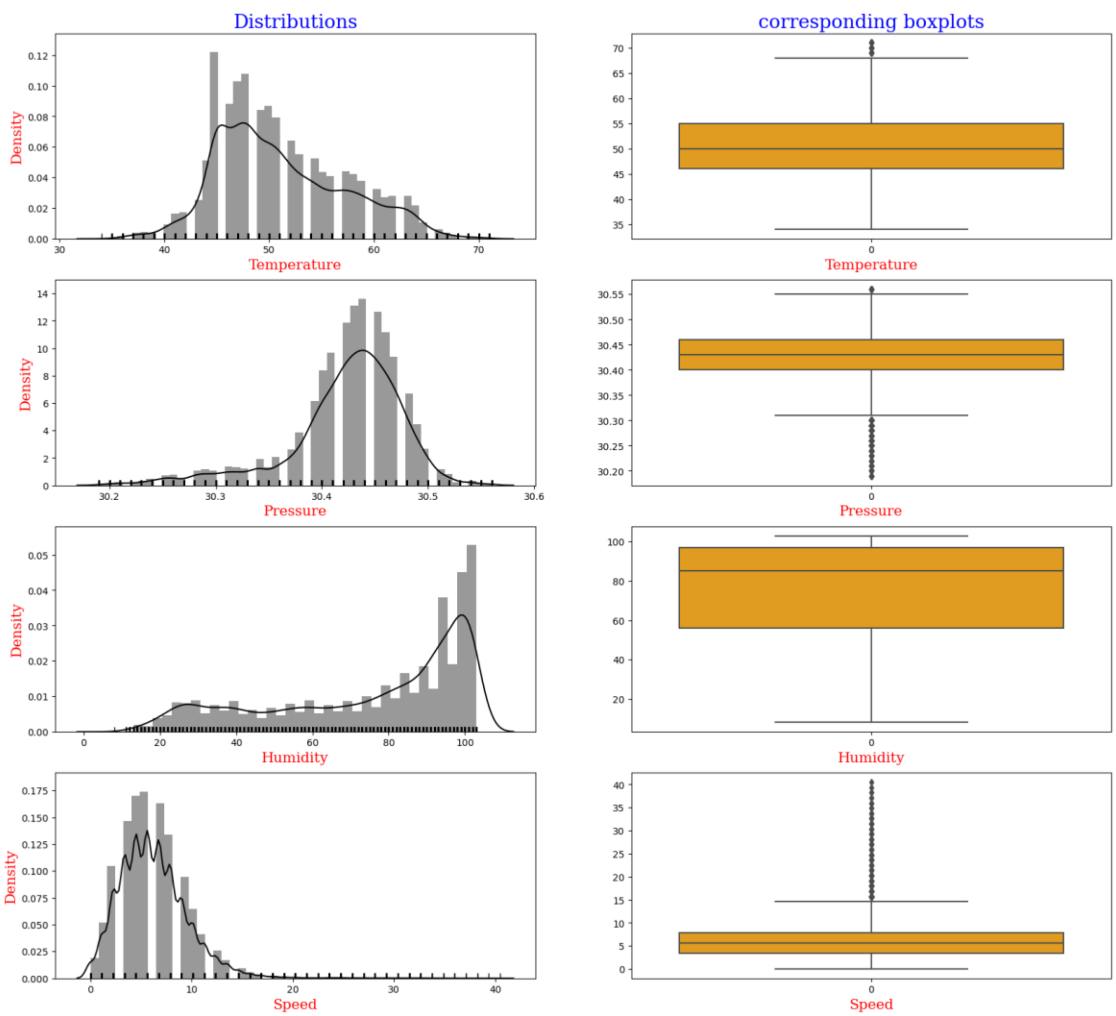
```
0
```

```
df.columns
```

```
Index(['UNIXTime', 'Date', 'Time', 'Radiation', 'Temperature', 'Pressure',
       'Humidity', 'WindDirection(Degrees)', 'Speed', 'TimeSunRise',
       'TimeSunSet'],
      dtype='object')
```

```
#checking for outliers
#outliers can be found by using boxplot
plt.figure(figsize=(20,18))

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.subplot(4,2,1)
plt.title("Distributions",fontdict=font1)
p=plt.xlabel("Temperature",fontdict=font2)
p.set_color("red")
p=plt.ylabel("Density",fontdict=font2)
p.set_color("red")
sns.distplot(df['Temperature'],rug=True,color='black')
plt.subplot(4,2,2)
plt.title("corresponding boxplots",fontdict=font1)
p=plt.xlabel("Temperature",fontdict=font2)
p.set_color("red")
sns.boxplot(df['Temperature'],color='orange')

plt.subplot(4,2,3)
p=plt.xlabel("Pressure",fontdict=font2)
p.set_color("red")
p=plt.ylabel("Density",fontdict=font2)
p.set_color("red")
sns.distplot(df['Pressure'],rug=True,color='black')
plt.subplot(4,2,4)
p=plt.xlabel("Pressure",fontdict=font2)
p.set_color("red")
sns.boxplot(df['Pressure'],color='orange')

plt.subplot(4,2,5)
p=plt.xlabel("Humidity",fontdict=font2)
p.set_color("red")
p=plt.ylabel("Density",fontdict=font2)
p.set_color("red")
```

```
sns.distplot(df['Humidity'],rug=True,color='black')
plt.subplot(4,2,6)
p=plt.xlabel("Humidity",fontdict=font2)
p.set_color("red")
sns.boxplot(df['Humidity'],color='orange')

plt.subplot(4,2,7)
p=plt.xlabel("Speed",fontdict=font2)
p.set_color("red")
p=plt.ylabel("Density",fontdict=font2)
p.set_color("red")
sns.distplot(df['Speed'],rug=True,color='black')
plt.subplot(4,2,8)
p=plt.xlabel("Speed",fontdict=font2)
p.set_color("red")
sns.boxplot(df['Speed'],color='orange')

plt.show()
```



Federated Learning on Solar Radiation Prediction

* for handling outliers we use different methods based on the distribution of data

- For Normally Distributed Data: Visual Inspection: Start by visually inspecting the distribution using histograms or Q-Q plots to check for symmetry and normality. Z-Score Method: Use the z-score method to identify outliers. Typically, values beyond ±3 standard deviations from the mean are considered outliers.

- For Skewed Data: Percentile-based Methods: Use percentile-based methods like interquartile range (IQR) to identify outliers. Values below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR are considered outliers

- For Heavy-tailed Distributions: Trimming: Trim the extreme values from the dataset if they are clearly outliers and do not represent genuine data points.

```
# knowing statistical info about data
df.describe()

            UNIXTime     Radiation   Temperature      Pressure      Humidity  \
count   3.268600e+04  32686.000000  32686.000000  32686.000000  32686.000000
mean    1.478047e+09    207.124697     51.103255     30.422879     75.016307
std     3.005037e+06    315.916387      6.201157      0.054673     25.990219
min     1.472724e+09      1.110000     34.000000     30.190000      8.000000
25%     1.475546e+09      1.230000     46.000000     30.400000     56.000000
50%     1.478026e+09      2.660000     50.000000     30.430000     85.000000
75%     1.480480e+09    354.235000     55.000000     30.460000     97.000000
max     1.483265e+09   1601.260000     71.000000     30.560000    103.000000

       WindDirection(Degrees)         Speed
count            32686.000000  32686.000000
mean               143.489821      6.243869
std                 83.167500      3.490474
min                  0.090000      0.000000
25%                 82.227500      3.370000
50%                147.700000      5.620000
75%                179.310000      7.870000
max                359.950000     40.500000
```

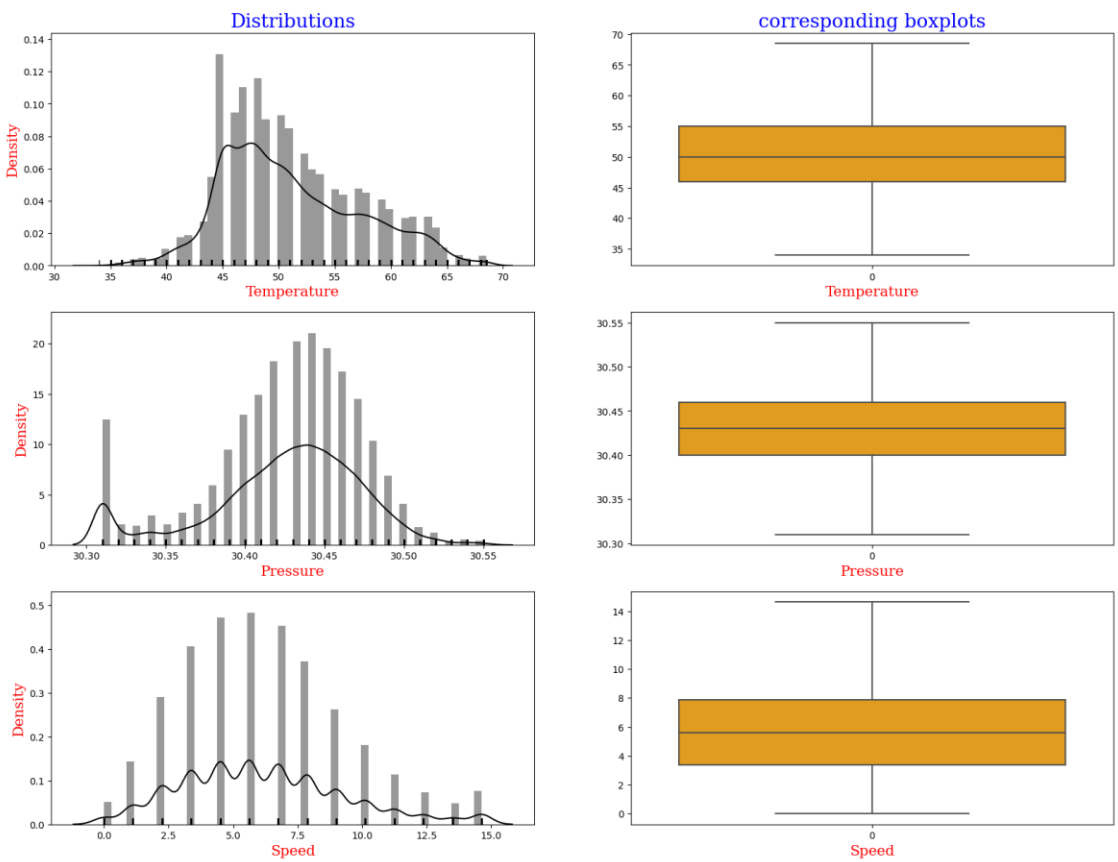
```
# finding outliers of temperature and filling with min and max
percentile25 = df["Temperature"].quantile(0.25)
percentile75 = df["Temperature"].quantile(0.75)

IQR = percentile75 - percentile25

min = percentile25 - 1.5*IQR
max = percentile75 + 1.5*IQR


df["Temperature"] = np.where(df["Temperature"]>max,
         max,
         np.where(df["Temperature"]<min,min, df["Temperature"]))
```

Federated Learning on Solar Radiation Prediction

```
# finding outliers of pressure and filling with min and max
percentile25 = df["Pressure"].quantile(0.25)
percentile75 = df["Pressure"].quantile(0.75)

IQR = percentile75 - percentile25

min = percentile25 - 1.5*IQR
max = percentile75 + 1.5*IQR


df["Pressure"] = np.where(df["Pressure"]>max,
         max,
         np.where(df["Pressure"]<min,min, df["Pressure"]))

# finding outliers of speed and filling with min and max
percentile25 = df["Speed"].quantile(0.25)
percentile75 = df["Speed"].quantile(0.75)

IQR = percentile75 - percentile25

min = percentile25 - 1.5*IQR
max = percentile75 + 1.5*IQR


df["Speed"] = np.where(df["Speed"]>max,
         max,
         np.where(df["Speed"]<min,min, df["Speed"]))

#after filling outliers with max and min
plt.figure(figsize=(20,15))

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.subplot(3,2,1)
plt.title("Distributions",fontdict=font1)
p=plt.xlabel("Temperature",fontdict=font2)
p.set_color("red")
p=plt.ylabel("Density",fontdict=font2)
p.set_color("red")
sns.distplot(df['Temperature'],rug=True,color='black')
plt.subplot(3,2,2)
plt.title("corresponding boxplots",fontdict=font1)
p=plt.xlabel("Temperature",fontdict=font2)
p.set_color("red")
sns.boxplot(df['Temperature'],color='orange')

plt.subplot(3,2,3)
p=plt.xlabel("Pressure",fontdict=font2)
```

Federated Learning on Solar Radiation Prediction

```
p.set_color("red")
p=plt.ylabel("Density",fontdict=font2)
p.set_color("red")
sns.distplot(df['Pressure'],rug=True,color='black')
plt.subplot(3,2,4)
p=plt.xlabel("Pressure",fontdict=font2)
p.set_color("red")
sns.boxplot(df['Pressure'],color='orange')


plt.subplot(3,2,5)
p=plt.xlabel("Speed",fontdict=font2)
p.set_color("red")
p=plt.ylabel("Density",fontdict=font2)
p.set_color("red")
sns.distplot(df['Speed'],rug=True,color='black')
plt.subplot(3,2,6)
p=plt.xlabel("Speed",fontdict=font2)
p.set_color("red")
sns.boxplot(df['Speed'],color='orange')

plt.show()
```




```
df.head()
```

```
     UNIXTime                   Date      Time  Radiation  Temperature  \
0  1475229326  9/29/2016 12:00:00 AM  23:55:26       1.21         48.0
1  1475229023  9/29/2016 12:00:00 AM  23:50:23       1.21         48.0
2  1475228726  9/29/2016 12:00:00 AM  23:45:26       1.23         48.0
3  1475228421  9/29/2016 12:00:00 AM  23:40:21       1.21         48.0
4  1475228124  9/29/2016 12:00:00 AM  23:35:24       1.17         48.0

   Pressure  Humidity  WindDirection(Degrees)  Speed TimeSunRise TimeSunSet
0     30.46        59                  177.39   5.62    06:13:00   18:13:00
1     30.46        58                  176.78   3.37    06:13:00   18:13:00
2     30.46        57                  158.75   3.37    06:13:00   18:13:00
3     30.46        60                  137.71   3.37    06:13:00   18:13:00
4     30.46        62                  104.95   5.62    06:13:00   18:13:00
```

```python
#extracting date from date feature
df['Date']=df['Date'].apply(lambda x: x.split()[0])
df.head()
```

```
     UNIXTime       Date      Time  Radiation  Temperature  Pressure  \
0  1475229326  9/29/2016  23:55:26       1.21         48.0     30.46
1  1475229023  9/29/2016  23:50:23       1.21         48.0     30.46
2  1475228726  9/29/2016  23:45:26       1.23         48.0     30.46
3  1475228421  9/29/2016  23:40:21       1.21         48.0     30.46
4  1475228124  9/29/2016  23:35:24       1.17         48.0     30.46

   Humidity  WindDirection(Degrees)  Speed TimeSunRise TimeSunSet
0        59                  177.39   5.62    06:13:00   18:13:00
1        58                  176.78   3.37    06:13:00   18:13:00
2        57                  158.75   3.37    06:13:00   18:13:00
3        60                  137.71   3.37    06:13:00   18:13:00
4        62                  104.95   5.62    06:13:00   18:13:00
```

```python
#extracting the day,month,hour,minute,sec from date and time features
df['Month']=pd.to_datetime(df['Date']).dt.month
df['Day']=pd.to_datetime(df['Date']).dt.day
df['Hour']=pd.to_datetime(df['Time']).dt.hour
df['Minute']=pd.to_datetime(df['Time']).dt.minute
df['Second']=pd.to_datetime(df['Date']).dt.second
df.head()
```

```
     UNIXTime       Date      Time  Radiation  Temperature  Pressure  \
0  1475229326  9/29/2016  23:55:26       1.21         48.0     30.46
1  1475229023  9/29/2016  23:50:23       1.21         48.0     30.46
2  1475228726  9/29/2016  23:45:26       1.23         48.0     30.46
3  1475228421  9/29/2016  23:40:21       1.21         48.0     30.46
4  1475228124  9/29/2016  23:35:24       1.17         48.0     30.46

   Humidity  WindDirection(Degrees)  Speed TimeSunRise TimeSunSet  Month  Day  \
0        59                  177.39   5.62    06:13:00   18:13:00      9   29
1        58                  176.78   3.37    06:13:00   18:13:00      9   29
```

```
2        57                    158.75   3.37    06:13:00   18:13:00      9   29
3        60                    137.71   3.37    06:13:00   18:13:00      9   29
4        62                    104.95   5.62    06:13:00   18:13:00      9   29

   Hour  Minute  Second
0    23      55       0
1    23      50       0
2    23      45       0
3    23      40       0
4    23      35       0
```

```
# extract the sunrise and sunset information using regular expression
df['risehour'] = df['TimeSunRise'].apply(lambda x : re.search(r'^\d+',
x).group(0)).astype(int)
df['riseminuter'] = df['TimeSunRise'].apply(lambda x : re.search(r'(?<=\:)\d+(?=\:)',
x).group(0)).astype(int)

df['sethour'] = df['TimeSunSet'].apply(lambda x : re.search(r'^\d+',
x).group(0)).astype(int)
df['setminute'] = df['TimeSunSet'].apply(lambda x : re.search(r'(?<=\:)\d+(?=\:)',
x).group(0)).astype(int)
df.head()
```

```
     UNIXTime       Date      Time  Radiation  Temperature  Pressure  \
0  1475229326  9/29/2016  23:55:26       1.21         48.0     30.46
1  1475229023  9/29/2016  23:50:23       1.21         48.0     30.46
2  1475228726  9/29/2016  23:45:26       1.23         48.0     30.46
3  1475228421  9/29/2016  23:40:21       1.21         48.0     30.46
4  1475228124  9/29/2016  23:35:24       1.17         48.0     30.46

   Humidity  WindDirection(Degrees)  Speed TimeSunRise TimeSunSet  Month  Day  \
0        59                  177.39   5.62    06:13:00   18:13:00      9   29
1        58                  176.78   3.37    06:13:00   18:13:00      9   29
2        57                  158.75   3.37    06:13:00   18:13:00      9   29
3        60                  137.71   3.37    06:13:00   18:13:00      9   29
4        62                  104.95   5.62    06:13:00   18:13:00      9   29

   Hour  Minute  Second  risehour  riseminuter  sethour  setminute
0    23      55       0         6           13       18         13
1    23      50       0         6           13       18         13
2    23      45       0         6           13       18         13
3    23      40       0         6           13       18         13
4    23      35       0         6           13       18         13
```

```
#drop the features which are not required after extracting desire information
df.drop(columns=['UNIXTime','Date','Time','TimeSunRise','TimeSunSet'],axis=1,inplace=
True)
df.head()
```

Federated Learning on Solar Radiation Prediction

```
   Radiation  Temperature  Pressure  Humidity  WindDirection(Degrees)  Speed  \
0       1.21         48.0     30.46        59                  177.39   5.62
1       1.21         48.0     30.46        58                  176.78   3.37
2       1.23         48.0     30.46        57                  158.75   3.37
3       1.21         48.0     30.46        60                  137.71   3.37
4       1.17         48.0     30.46        62                  104.95   5.62

   Month  Day  Hour  Minute  Second  risehour  riseminuter  sethour  setminute
0      9   29    23      55       0         6           13       18         13
1      9   29    23      50       0         6           13       18         13
2      9   29    23      45       0         6           13       18         13
3      9   29    23      40       0         6           13       18         13
4      9   29    23      35       0         6           13       18         13
```

```python
#splitting the data into features and labels
X=df.drop('Radiation',axis=1)
Y=df['Radiation']
print(X.shape)
print(Y.shape)
```

```
(32686, 14)
(32686,)
```

***Feature Engineering**

- feature selection using correlation coefficient

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

- r = correlation coefficient
- x_i = values of the x-variable in a sample
- $\bar{x}$ = mean of the values of the x-variable
- y_i = values of the y-variable in a sample
- $\bar{y}$ = mean of the values of the y-variable

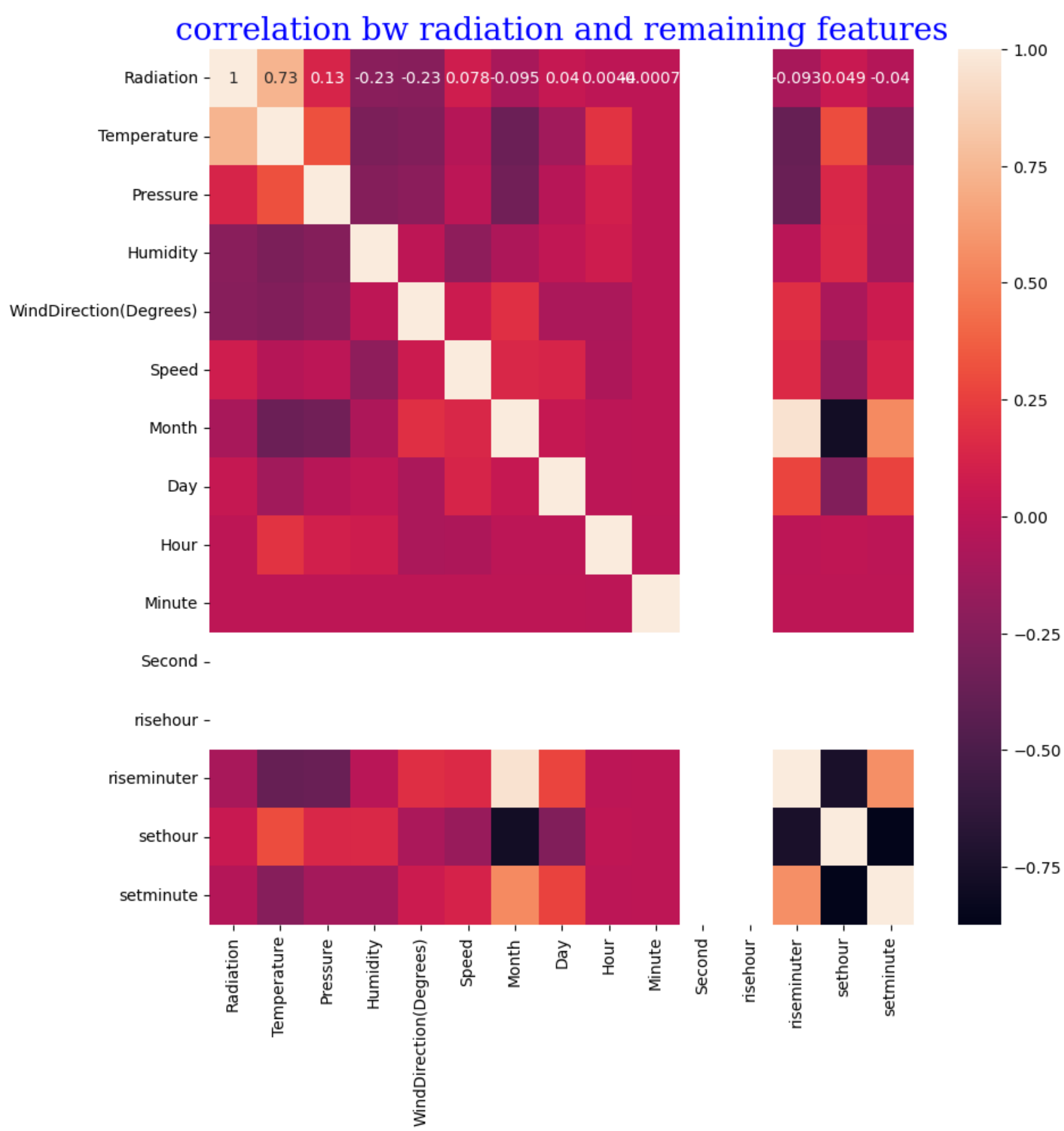
```python
#correaltion matrix
corr_mat=df.corr()
corr_mat
```

```
                        Radiation  Temperature  Pressure  Humidity  \
Radiation                1.000000     0.734831  0.127280 -0.226171
Temperature              0.734831     1.000000  0.314977 -0.284613
Pressure                 0.127280     0.314977  1.000000 -0.250540
Humidity                -0.226171    -0.284613 -0.250540  1.000000
WindDirection(Degrees)  -0.230324    -0.259700 -0.216271 -0.001833
Speed                    0.078178    -0.037583 -0.008457 -0.207280
Month                   -0.095450    -0.354449 -0.325061 -0.068854
Day                      0.039978    -0.124220 -0.024957  0.014637
Hour                     0.004398     0.197740  0.098771  0.077899
Minute                  -0.000730    -0.001922  0.001803  0.000499
Second                        NaN          NaN       NaN       NaN
```

```
risehour                         NaN          NaN       NaN       NaN
riseminuter                -0.092850    -0.381032 -0.363418 -0.023955
sethour                     0.048719     0.300844  0.138642  0.145143
setminute                  -0.039816    -0.242880 -0.108588 -0.119526

                        WindDirection(Degrees)     Speed     Month       Day  \
Radiation                            -0.230324  0.078178 -0.095450  0.039978
Temperature                          -0.259700 -0.037583 -0.354449 -0.124220
Pressure                             -0.216271 -0.008457 -0.325061 -0.024957
Humidity                             -0.001833 -0.207280 -0.068854  0.014637
WindDirection(Degrees)                1.000000  0.064424  0.181485 -0.082354
Speed                                 0.064424  1.000000  0.138759  0.126346
Month                                 0.181485  0.138759  1.000000  0.038027
Day                                  -0.082354  0.126346  0.038027  1.000000
Hour                                 -0.077969 -0.074721 -0.005396 -0.008010
Minute                               -0.000602  0.000107  0.000168 -0.000196
Second                                     NaN       NaN       NaN       NaN
risehour                                   NaN       NaN       NaN       NaN
riseminuter                           0.176929  0.154357  0.952472  0.274522
sethour                              -0.078540 -0.159356 -0.784783 -0.263575
setminute                             0.070030  0.119862  0.541883  0.265662

                            Hour    Minute  Second  risehour  riseminuter  \
Radiation               0.004398 -0.000730     NaN       NaN    -0.092850
Temperature             0.197740 -0.001922     NaN       NaN    -0.381032
Pressure                0.098771  0.001803     NaN       NaN    -0.363418
Humidity                0.077899  0.000499     NaN       NaN    -0.023955
WindDirection(Degrees) -0.077969 -0.000602     NaN       NaN     0.176929
Speed                  -0.074721  0.000107     NaN       NaN     0.154357
Month                  -0.005396  0.000168     NaN       NaN     0.952472
Day                    -0.008010 -0.000196     NaN       NaN     0.274522
Hour                    1.000000 -0.004052     NaN       NaN    -0.006772
Minute                 -0.004052  1.000000     NaN       NaN    -0.000158
Second                       NaN       NaN     NaN       NaN          NaN
risehour                     NaN       NaN     NaN       NaN          NaN
riseminuter            -0.006772 -0.000158     NaN       NaN     1.000000
sethour                 0.008629  0.001052     NaN       NaN    -0.742329
setminute              -0.007056 -0.002215     NaN       NaN     0.562851

                         sethour  setminute
Radiation               0.048719  -0.039816
Temperature             0.300844  -0.242880
Pressure                0.138642  -0.108588
Humidity                0.145143  -0.119526
WindDirection(Degrees) -0.078540   0.070030
Speed                  -0.159356   0.119862
Month                  -0.784783   0.541883
Day                    -0.263575   0.265662
Hour                    0.008629  -0.007056
```

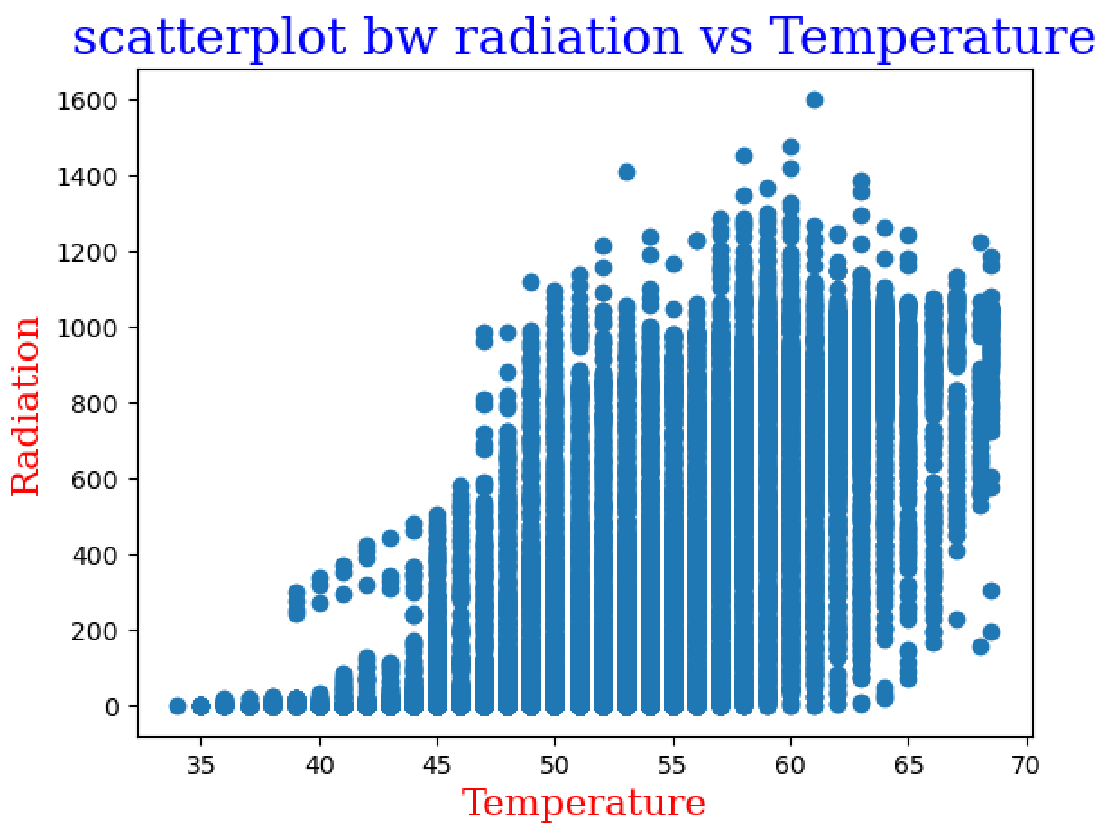
```
Minute                   0.001052  -0.002215
Second                        NaN        NaN
risehour                      NaN        NaN
riseminuter             -0.742329   0.562851
sethour                  1.000000  -0.873471
setminute               -0.873471   1.000000
```

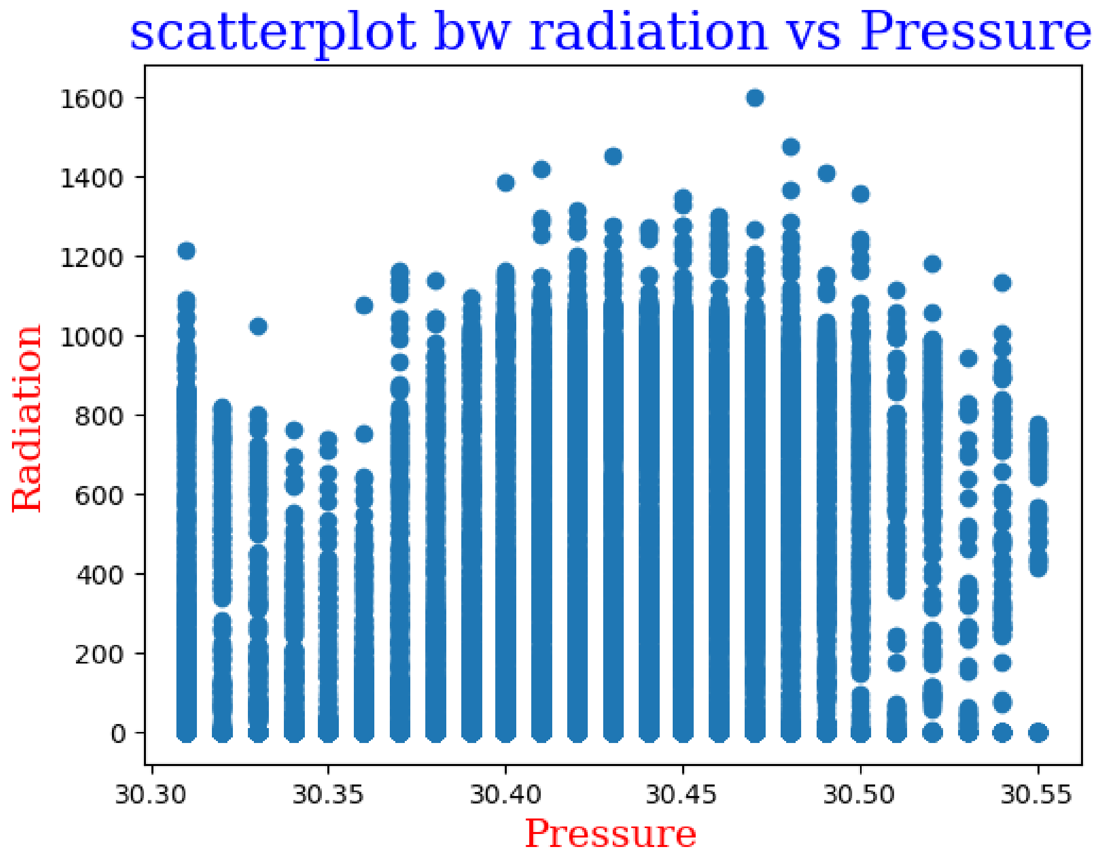
```
#heatmap of correlation matrix
plt.figure(figsize=(10,10))
plt.title("correlation bw radiation and remaining features",fontdict=font1)
sns.heatmap(df.corr(),annot=True)
plt.show()
```

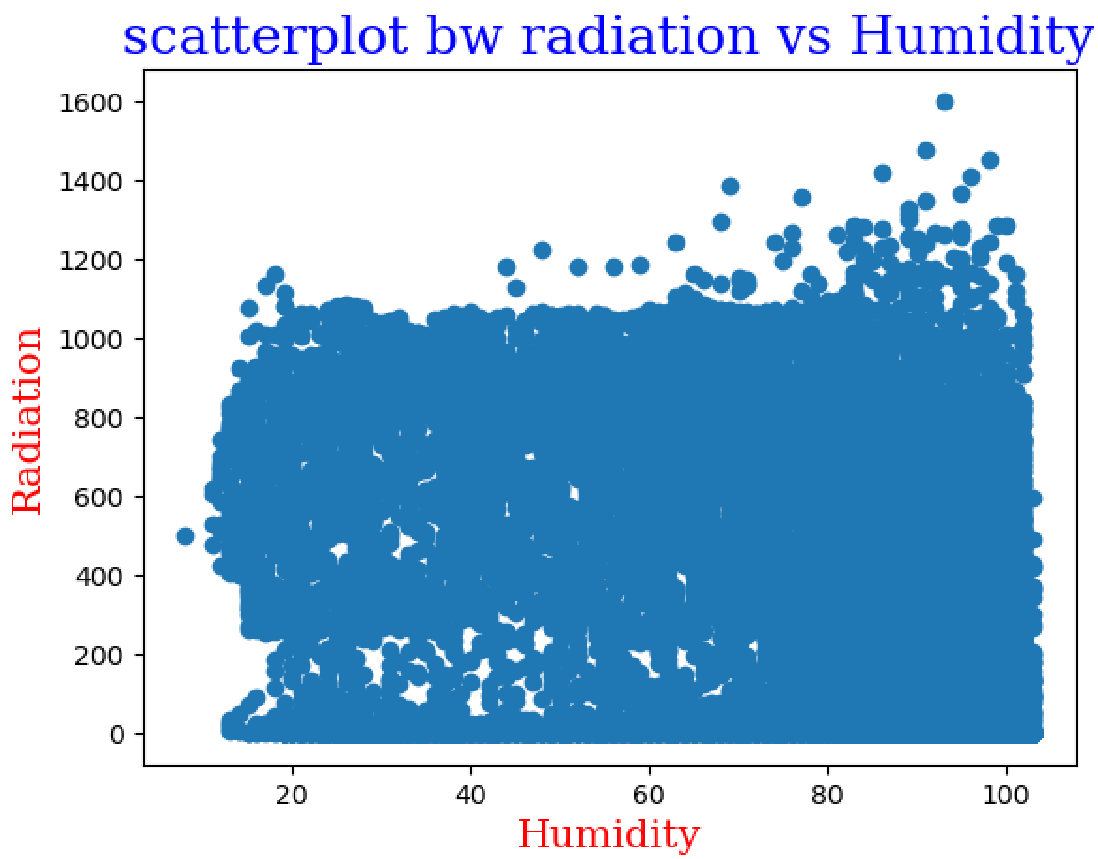


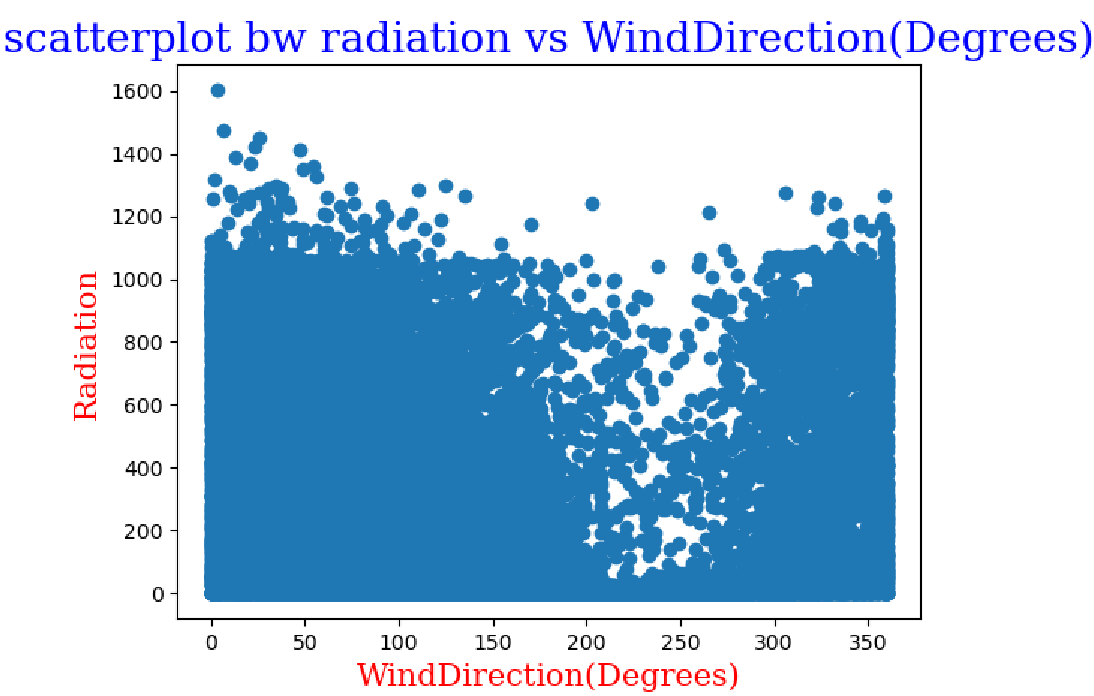

```
#visual realtion bw radiation and other features
for i in df.columns:
```

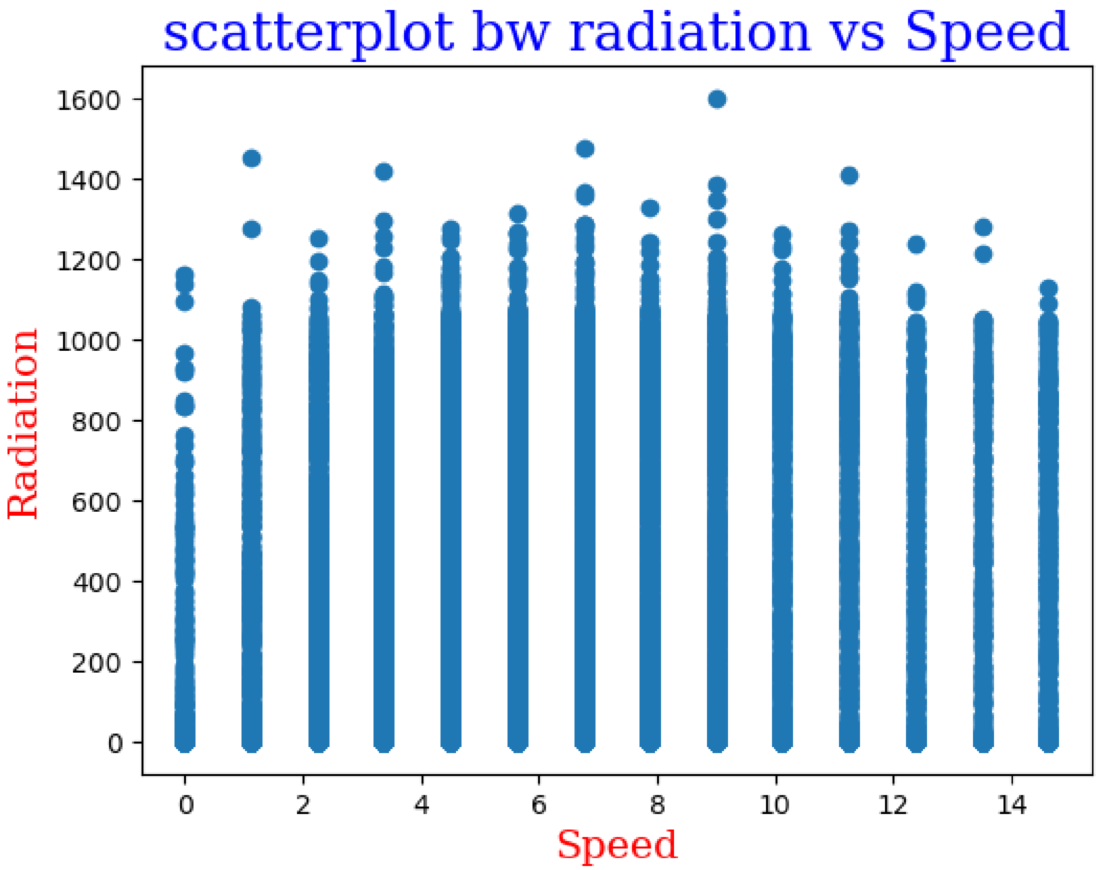
```
if i!='Radiation':
    plt.title('scatterplot bw radiation vs '+i,fontdict=font1)
    p=plt.xlabel(i,fontdict=font2)
    p.set_color("red")
    p=plt.ylabel("Radiation",fontdict=font2)
    p.set_color("red")

    plt.scatter(x=df[i],y=df['Radiation'])

    plt.show()
```

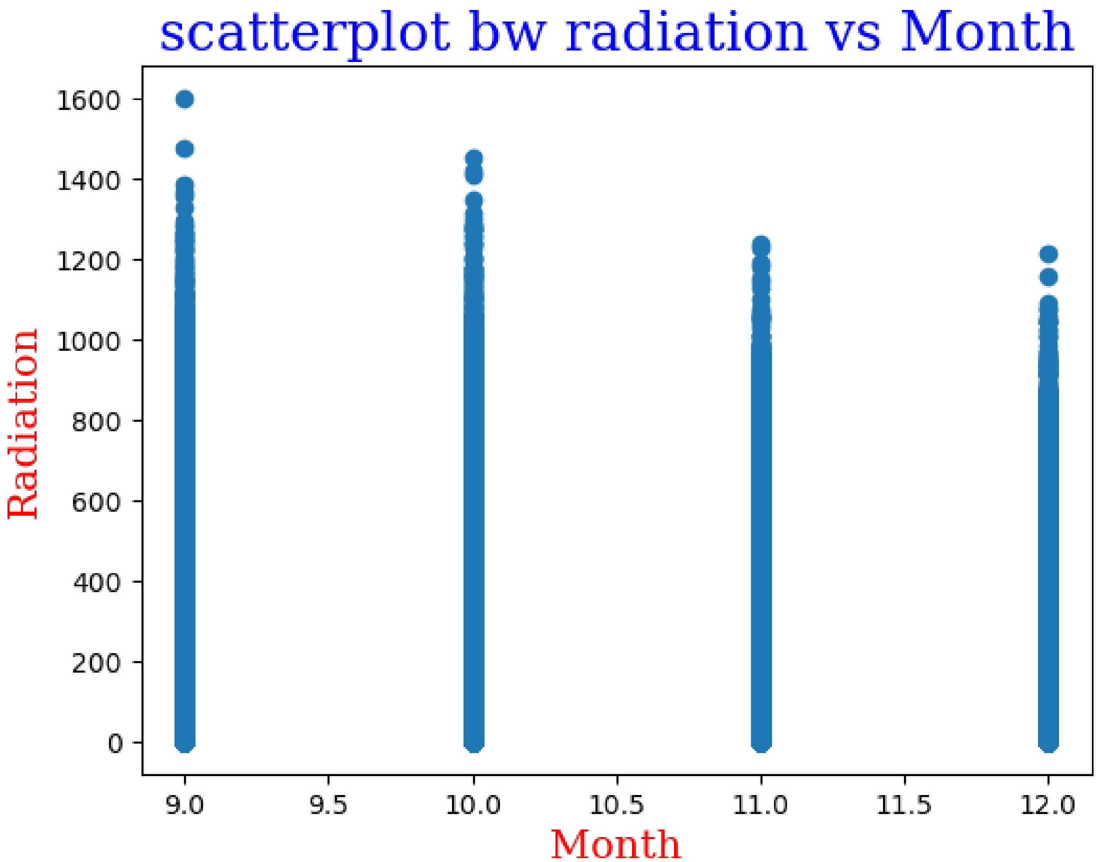


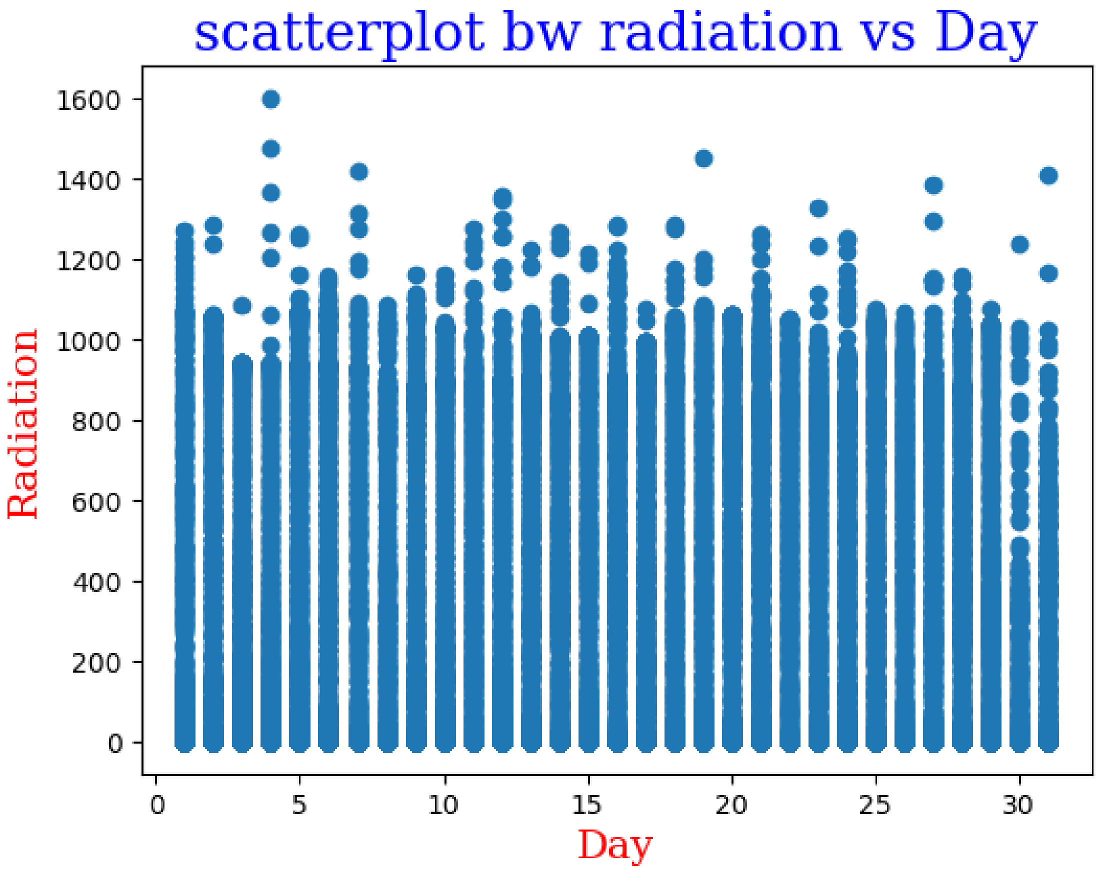

scatterplot bw radiation vs Pressure
Radiation
Pressure
1600
1400
1200
1000
800
600
400
200
0
30.30
30.35
30.40
30.45
30.50
30.55

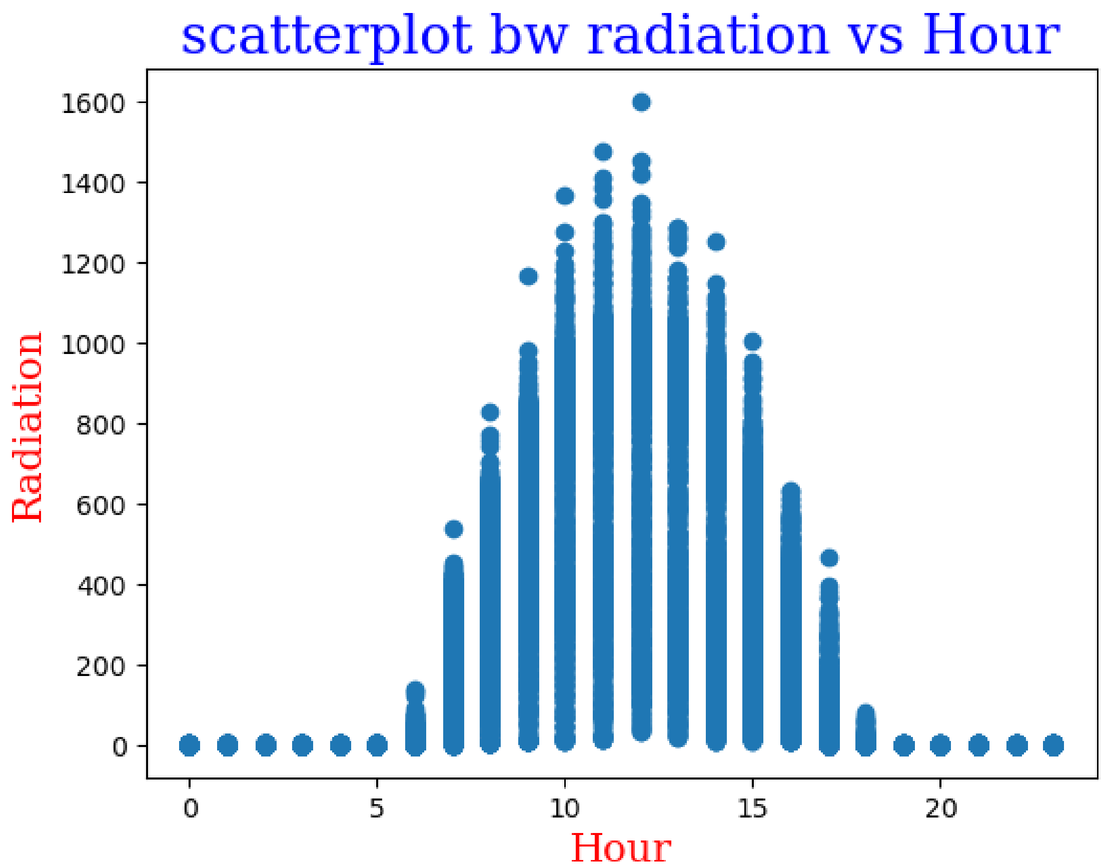

scatterplot bw radiation vs Humidity
Radiation
1600
1400
1200
1000
800
600
400
200
0
20
40
60
80
100
Humidity

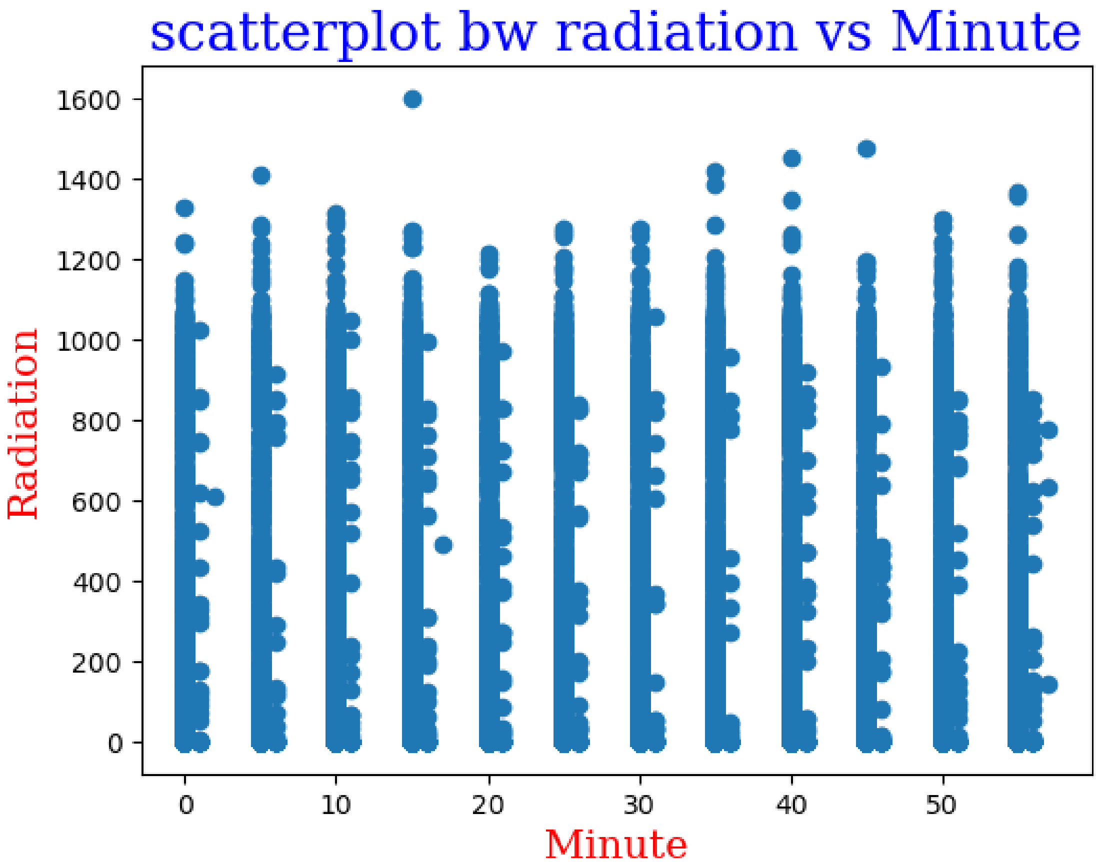

scatterplot bw radiation vs WindDirection(Degrees)
Radiation
1600
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300
350
WindDirection(Degrees)

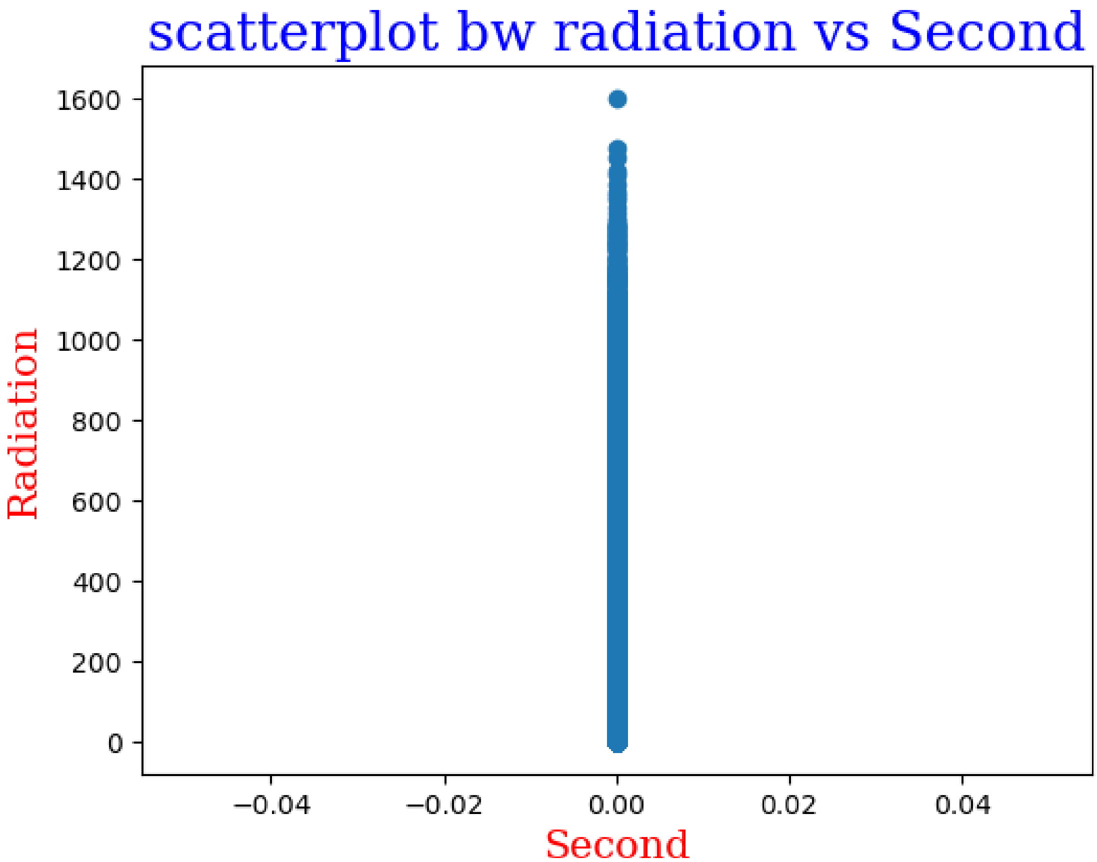

scatterplot bw radiation vs Speed
Radiation
1600
1400
1200
1000
800
600
400
200
0
0
2
4
6
8
10
12
14
Speed

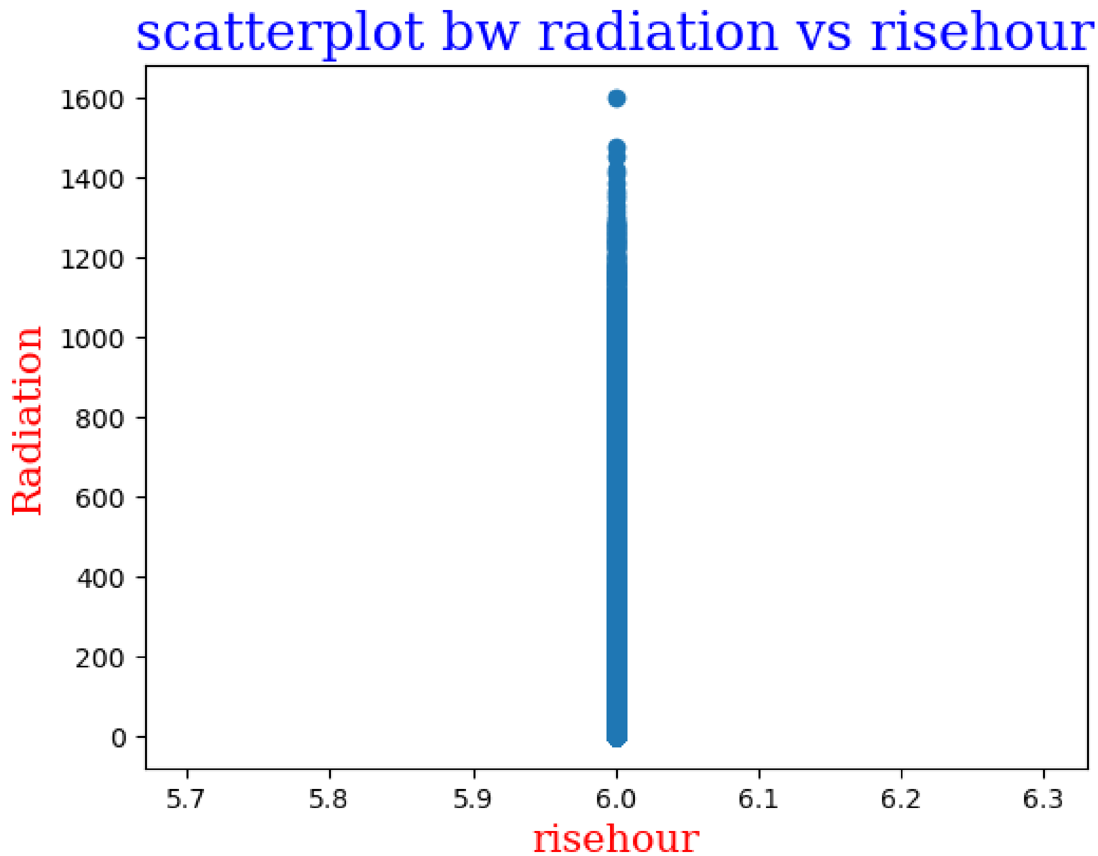

scatterplot bw radiation vs Month
Radiation
Month
1600
1400
1200
1000
800
600
400
200
0
9.0
9.5
10.0
10.5
11.0
11.5
12.0

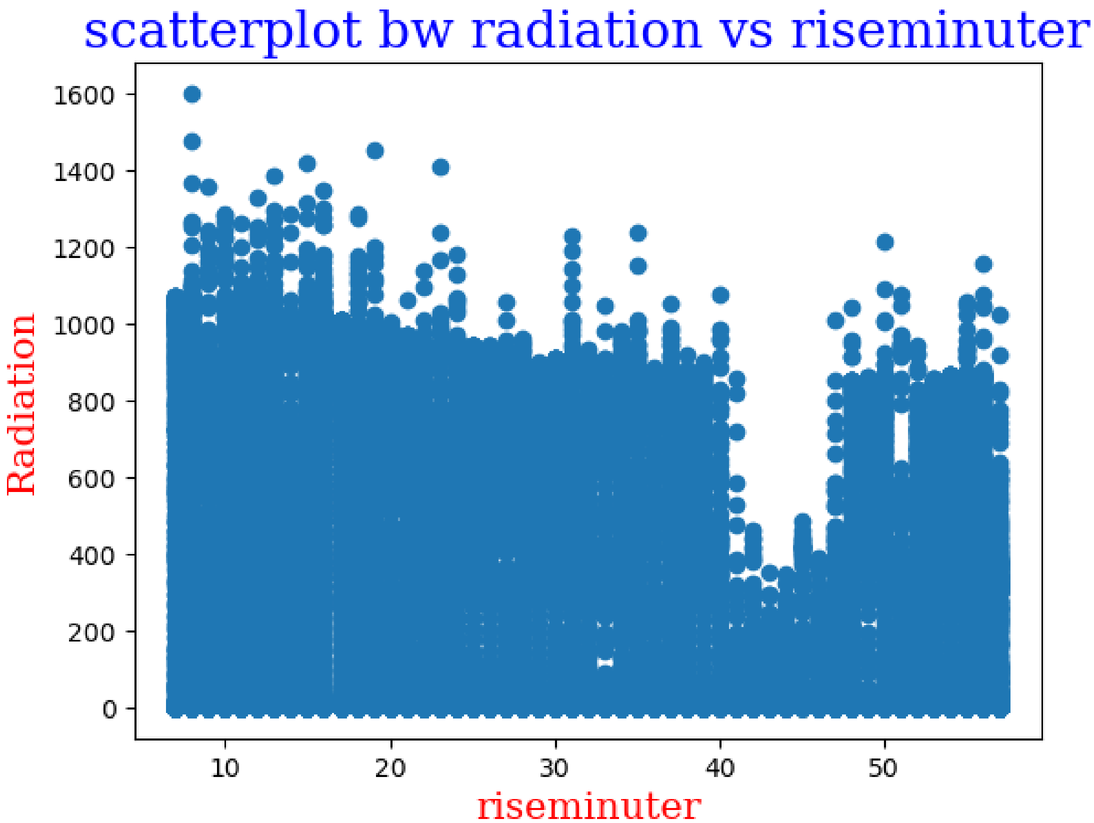

scatterplot bw radiation vs Day
Radiation
1600
1400
1200
1000
800
600
400
200
0
0
5
10
15
20
25
30
Day

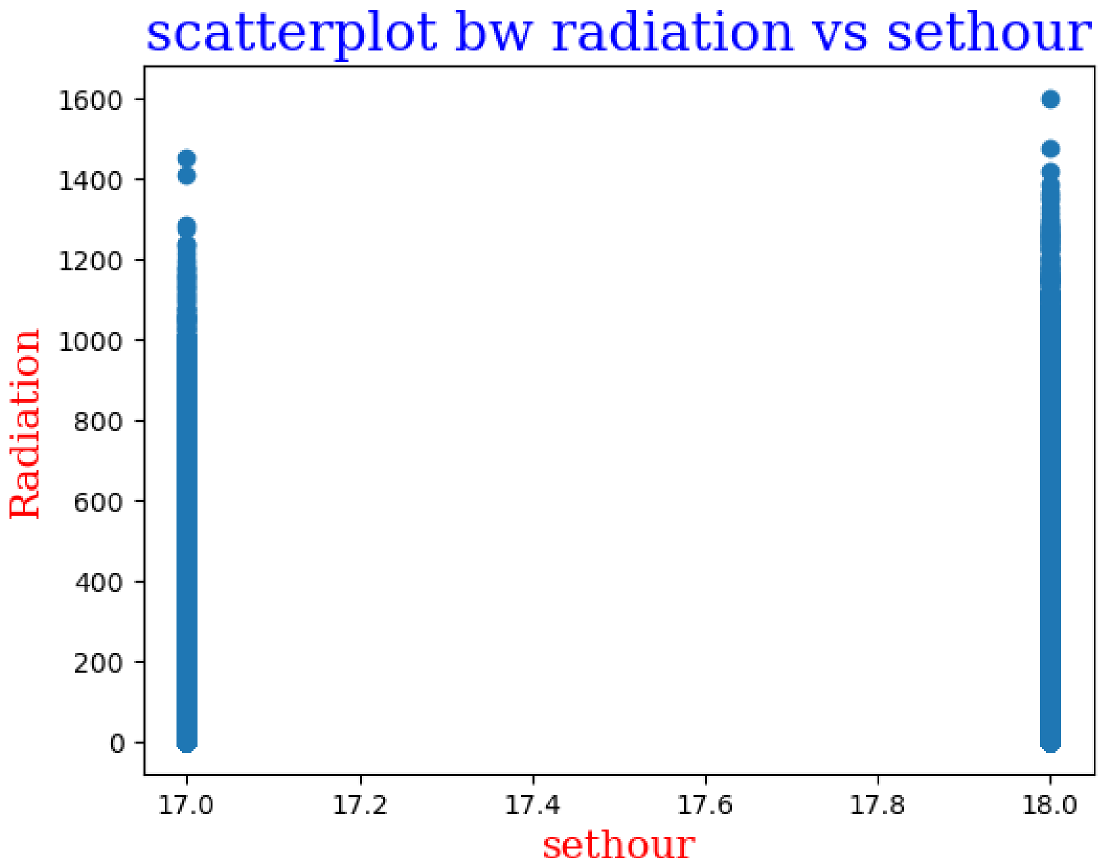

scatterplot bw radiation vs Hour
Radiation
1600
1400
1200
1000
800
600
400
200
0
0
5
10
15
20
Hour

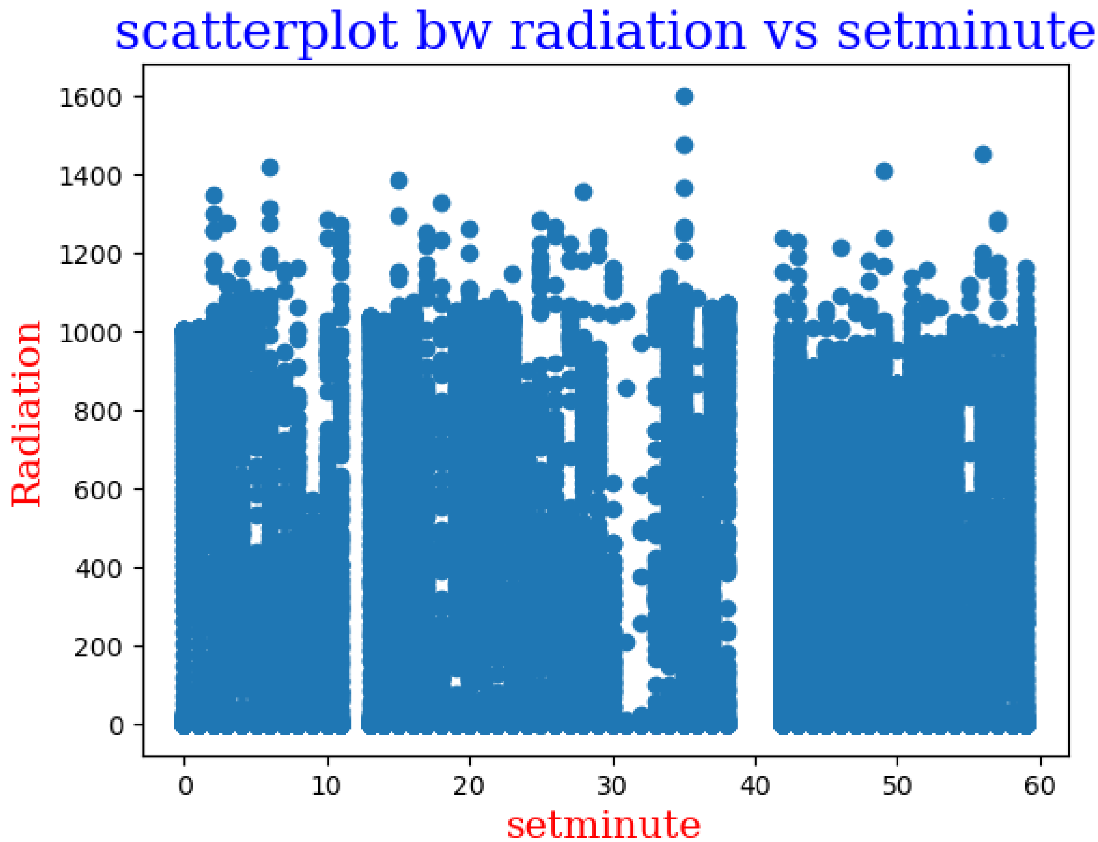

scatterplot bw radiation vs Minute
Radiation
Minute
0
200
400
600
800
1000
1200
1400
1600
0
10
20
30
40
50


scatterplot bw radiation vs Second
Radiation
Second
1600
1400
1200
1000
800
600
400
200
0
−0.04
−0.02
0.00
0.02
0.04


scatterplot bw radiation vs risehour
Radiation
1600
1400
1200
1000
800
600
400
200
0
5.7
5.8
5.9
6.0
6.1
6.2
6.3
risehour


scatterplot bw radiation vs riseminuter
Radiation
riseminuter
0
200
400
600
800
1000
1200
1400
1600
10
20
30
40
50


scatterplot bw radiation vs sethour
Radiation
sethour
1600
1400
1200
1000
800
600
400
200
0
17.0
17.2
17.4
17.6
17.8
18.0




```python
#transformation of data
minmax_scaler = MinMaxScaler()
for i in X.columns:
    X[i]=minmax_scaler.fit_transform(np.array(X[i]).reshape(-1,1))

X.tail()
```

```
       Temperature  Pressure  Humidity  WindDirection(Degrees)     Speed  \
32681     0.289855  0.500000  0.989474                0.403851  0.461696
32682     0.289855  0.458333  0.989474                0.327044  0.461696
32683     0.289855  0.458333  0.989474                0.403212  0.615595
32684     0.289855  0.458333  0.978947                0.456011  0.538304
32685     0.289855  0.500000  0.978947                0.232035  0.230506

       Month  Day  Hour    Minute  Second  risehour  riseminuter  sethour  \
32681    1.0  0.0   0.0  0.350877     0.0       0.0         0.68      0.0
32682    1.0  0.0   0.0  0.263158     0.0       0.0         0.68      0.0
32683    1.0  0.0   0.0  0.175439     0.0       0.0         0.68      0.0
32684    1.0  0.0   0.0  0.087719     0.0       0.0         0.68      0.0
32685    1.0  0.0   0.0  0.000000     0.0       0.0         0.68      0.0

       setminute
32681   0.711864
```

Federated Learning on Solar Radiation Prediction

```
32682    0.711864
32683    0.711864
32684    0.711864
32685    0.711864
```

```
#spliting the data into train and test data
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size=0.2, random_state=42)

print(xtrain.shape,xtest.shape)
```

```
(26148, 14) (6538, 14)
```

```
data_x=xtrain.iloc[0:20]
data_y=ytrain.iloc[0:20]

#divide data into 4 clients

c1_x=xtrain.iloc[0:7000]
c1_y=ytrain.iloc[0:7000]
c2_x=xtrain.iloc[7000:14000]
c2_y=ytrain.iloc[7000:14000]
c3_x=xtrain.iloc[14000:21000]
c3_y=ytrain.iloc[14000:21000]
c4_x=xtrain.iloc[21000:]
c4_y=ytrain.iloc[21000:]

print(c1_x.shape,c1_y.shape)
print(c2_x.shape,c2_y.shape)
print(c3_x.shape,c3_y.shape)
print(c4_x.shape,c4_y.shape)
```

```
(7000, 14) (7000,)
(7000, 14) (7000,)
(7000, 14) (7000,)
(5148, 14) (5148,)
```

* federated learning using linear regression

```
#model training & evaluation
model_server=LinearRegression()
model_server.fit(data_x,data_y)
print(model_server.intercept_)

model_c1=LinearRegression(fit_intercept=False)
model_c2=LinearRegression(fit_intercept=False)
model_c3=LinearRegression(fit_intercept=False)
model_c4=LinearRegression(fit_intercept=False)

rounds=5
for i in range(rounds):
    print('')
```

```
    print('round-{}'.format(i+1))
    #client-1 model
    model_c1.intercept_=model_server.intercept_
    model_c1.coef_=model_server.coef_
    model_c1.fit(c1_x,c1_y)

    #clinet-2 model
    model_c2.intercept_=model_server.intercept_
    model_c2.coef_=model_server.coef_
    model_c2.fit(c2_x,c2_y)

    #client-3 model
    model_c3.intercept_=model_server.intercept_
    model_c3.coef_=model_server.coef_
    model_c3.fit(c3_x,c3_y)

    #client-4 model
    model_c4.intercept_=model_server.intercept_
    model_c4.coef_=model_server.coef_
    model_c4.fit(c4_x,c4_y)

    #averaging the parameters
    stacked_arrays = np.vstack((model_c1.coef_, model_c2.coef_,
model_c3.coef_,model_c4.coef_,model_server.coef_))

    # Calculate the average along the specified axis (axis=0 for columns)
    server_coef = np.mean(stacked_arrays, axis=0)
    print('*******************************************')
    print("Average coef values of server model:", server_coef)

server_intercept=(model_c1.intercept_+model_c2.intercept_+model_c3.intercept_+model_c
4.intercept_+model_server.intercept_)/5
    print("Average intercept values of server model:", (server_intercept))

    #update parameters of server_model

    model_server.intercept_=server_intercept
    model_server.coef_=server_coef

    ypred=model_server.predict(xtest)

    #evaluation parameters
    rmse =np.sqrt(mean_squared_error(ytest, ypred))
    r2 = r2_score(ytest, ypred)

    print('\n')
    print("Testing performance")
```

```
    print("RMSE: {:.2f}".format(rmse))
    print("R2: {:.2f}".format(r2))
```

```
-1145.2910754888383

round-1
******************************************
Average coef values of server model: [ 1.40958575e+03 -9.58697741e+01  9.46433154e+01
-2.47263723e+02
  3.04692739e+01 -2.37197484e+02 -1.21154078e+01 -1.66822495e+02
 -2.85683638e+01  7.95807864e-14  7.10542736e-14  3.61456839e+02
 -1.54281205e+02 -1.02825615e+02]
Average intercept values of server model: -229.05821509776766


Testing performance
RMSE: 199.39
R2: 0.60

round-2
******************************************
Average coef values of server model: [ 1.51050776e+03 -1.22377168e+02  4.84828954e+01
-1.16691070e+02
  7.99803177e+01 -6.49384251e+02 -1.19283680e+02 -1.66780532e+02
 -7.96554306e+00  4.54747351e-15 -5.68434189e-15  7.42440710e+02
 -3.21988446e+02 -2.76071589e+02]
Average intercept values of server model: -45.81164301955353


Testing performance
RMSE: 193.87
R2: 0.62

round-3
******************************************
Average coef values of server model: [ 1.53069216e+03 -1.27678647e+02  3.92508115e+01
-9.05765397e+01
  8.98825264e+01 -7.31821605e+02 -1.40717335e+02 -1.66772140e+02
 -3.84497891e+00 -1.04591891e-14 -2.10320650e-14  8.18637484e+02
 -3.55529895e+02 -3.10720784e+02]
Average intercept values of server model: -9.162328603910705


Testing performance
RMSE: 193.72
R2: 0.62
```

```
round-4
******************************************
Average coef values of server model: [ 1.53472904e+03 -1.28738943e+02  3.74043947e+01
-8.53536336e+01
  9.18629682e+01 -7.48309076e+02 -1.45004066e+02 -1.66770461e+02
 -3.02086608e+00 -1.34605216e-14 -2.41016096e-14  8.33876839e+02
 -3.62238184e+02 -3.17650623e+02]
Average intercept values of server model: -1.832465720782141


Testing performance
RMSE: 193.73
R2: 0.62

round-5
******************************************
Average coef values of server model: [ 1.53553642e+03 -1.28951002e+02  3.70351113e+01
-8.43090523e+01
  9.22590565e+01 -7.51606570e+02 -1.45861412e+02 -1.66770125e+02
 -2.85604351e+00 -1.40607881e-14 -2.47155185e-14  8.36924709e+02
 -3.63579842e+02 -3.19036591e+02]
Average intercept values of server model: -0.3664931441564282


Testing performance
RMSE: 193.74
R2: 0.62
```

```
ypred=model_server.predict(xtest)

#evaluation parameters
rmse =np.sqrt(mean_squared_error(ytest, ypred))
r2 = r2_score(ytest, ypred)
mae = mean_absolute_error(ytest,ypred)

print('\n')
print("Testing performance")

print("RMSE: {:.2f}".format(rmse))
print("R2: {:.2f}".format(r2))
print("mae: {:.2f}".format(mae))
```

```
Testing performance
RMSE: 193.74
R2: 0.62
mae: 146.72
```

Federated Learning on Solar Radiation Prediction

* without federated learning

```
#model training
model=LinearRegression()
model.fit(xtrain,ytrain)
print(model.coef_)
print(model.intercept_)
```

```
[ 1.53979292e+03 -1.22238800e+02  4.63469849e+01 -8.11679485e+01
  9.82044284e+01 -3.45413930e+02 -2.34509749e+01 -1.67319337e+02
 -2.09907173e+00  0.00000000e+00 -3.33066907e-16  4.54215069e+02
 -2.03131857e+02 -1.53913289e+02]
-285.8875076092297
```

```
#model testing
ypred=model.predict(xtest)
```

```
#evaluation parameters
rmse =np.sqrt(mean_squared_error(ytest, ypred))
r2 = r2_score(ytest, ypred)
mae = mean_absolute_error(ytest,ypred)

print("Testing performance")

print("RMSE: {:.2f}".format(rmse))
print("R2: {:.2f}".format(r2))
print("mae: {:.2f}".format(mae))
```

```
Testing performance
RMSE: 193.44
R2: 0.62
mae: 146.50
```

Federated learning Using MultiLayer Perceptron

```
def create_model():
    model = None
    model = Sequential()

    model.add(Dense(128, activation='relu', input_dim=14))
    model.add(Dropout(0.33))

    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.33))

    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.33))

    model.add(Dense(1, activation='linear'))
```

```
    return model


#model training & evaluation
model_server=create_model()
model_server.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))

model_c1=create_model()
model_c2=create_model()
model_c3=create_model()
model_c4=create_model()

rounds=20
for i in range(rounds):
    print('')
    print('round-{}'.format(i+1))
    #client-1 model
    model_c1.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))
    model_c1.set_weights(model_server.get_weights())
    model_c1.fit(c1_x,c1_y,validation_split=0.1, epochs=1, batch_size=32)

    print('*********************************')
    #clinet-2 model
    model_c2.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))
    model_c2.set_weights(model_server.get_weights())
    model_c2.fit(c2_x,c2_y,validation_split=0.1, epochs=1, batch_size=32)

    print('***********************************')
    #client-3 model
    model_c3.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))
    model_c3.set_weights(model_server.get_weights())
    model_c3.fit(c3_x,c3_y,validation_split=0.1, epochs=1, batch_size=32)

    print('***********************************')
    #client-4 model
    model_c4.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))
    model_c4.set_weights(model_server.get_weights())
    model_c4.fit(c4_x,c4_y,validation_split=0.1, epochs=1, batch_size=32)

    #averaging the parameters


avg_weights=[(model_c1.get_weights()[i]+model_c2.get_weights()[i]+model_c3.get_weight
s()[i]+model_c4.get_weights()[i]+model_server.get_weights()[i])/5 for i in
```

```
range(len(model_server.get_weights()))]

    model_server.set_weights(avg_weights)

    mae = mean_absolute_error(ytest,model_server.predict(xtest))
    print(f'mean absolute error: {mae}')
```

```
round-1
197/197 ━━━━━━━━━━━━━━━━━━━━3s 4ms/step - loss: 200.9048 - mse: 136525.9375 -
val_loss: 209.0922 - val_mse: 136608.6094
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━3s 2ms/step - loss: 206.0894 - mse: 140739.3750 -
val_loss: 224.8019 - val_mse: 150307.0469
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 211.6204 - mse: 149247.3281 -
val_loss: 189.5787 - val_mse: 118873.7578
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 207.5963 - mse: 143960.6250 -
val_loss: 211.0222 - val_mse: 140122.3906
205/205 ━━━━━━━━━━━━━━━━━━━━0s 1ms/step
mean absolute error: 200.40483560411926

round-2
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 193.1158 - mse: 124663.3828 -
val_loss: 166.0232 - val_mse: 68501.2266
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 205.7316 - mse: 138996.5312 -
val_loss: 175.8246 - val_mse: 82505.6094
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 201.8798 - mse: 134200.8906 -
val_loss: 153.0273 - val_mse: 77970.0078
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 205.5784 - mse: 138877.8125 -
val_loss: 188.1009 - val_mse: 108062.7266
205/205 ━━━━━━━━━━━━━━━━━━━━0s 833us/step
mean absolute error: 177.53996769882846

round-3
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 167.7762 - mse: 87140.4297 -
val_loss: 125.2884 - val_mse: 46413.1016
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 161.2383 - mse: 82965.0000 -
val_loss: 130.1884 - val_mse: 49857.9609
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 173.8496 - mse: 91029.0234 -
val_loss: 111.3008 - val_mse: 40278.0156
***********************************
```

```
145/145 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 174.9710 - mse: 92615.4297 -
val_loss: 133.4687 - val_mse: 55818.9336
205/205 ━━━━━━━━━━━━━━━━━━━━0s 805us/step
mean absolute error: 134.67154794605037

round-4
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 131.5976 - mse: 53639.3750 -
val_loss: 111.3175 - val_mse: 39574.2930
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 133.7327 - mse: 57122.2188 -
val_loss: 115.2894 - val_mse: 39815.1836
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 131.7661 - mse: 54663.2227 -
val_loss: 96.0243 - val_mse: 28228.7734
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 137.0204 - mse: 56304.0547 -
val_loss: 110.1122 - val_mse: 41102.2734
205/205 ━━━━━━━━━━━━━━━━━━━━0s 844us/step
mean absolute error: 111.68445482742717

round-5
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 118.2147 - mse: 44835.9648 -
val_loss: 102.4152 - val_mse: 34855.9883
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 118.0465 - mse: 47213.0039 -
val_loss: 102.5502 - val_mse: 33631.8320
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━3s 2ms/step - loss: 118.3943 - mse: 45701.6523 -
val_loss: 87.3875 - val_mse: 25968.2715
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━2s 4ms/step - loss: 119.9025 - mse: 47538.6523 -
val_loss: 101.8534 - val_mse: 36002.0312
205/205 ━━━━━━━━━━━━━━━━━━━━0s 817us/step
mean absolute error: 100.59438890915357

round-6
197/197 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 107.4667 - mse: 38850.9219 -
val_loss: 91.7846 - val_mse: 28077.1797
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━3s 4ms/step - loss: 105.4186 - mse: 38218.4688 -
val_loss: 96.2232 - val_mse: 30461.7891
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 108.8218 - mse: 40308.6484 -
val_loss: 81.9803 - val_mse: 24153.4980
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 115.6142 - mse: 43718.9219 -
val_loss: 91.2156 - val_mse: 28643.1719
205/205 ━━━━━━━━━━━━━━━━━━━━0s 883us/step
mean absolute error: 92.25374029418167
```

Federated Learning on Solar Radiation Prediction

```
round-7
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 100.3767 - mse: 35369.5000 -
val_loss: 85.9984 - val_mse: 23795.8535
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 100.6651 - mse: 35151.2266 -
val_loss: 97.7520 - val_mse: 32993.1797
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 101.8137 - mse: 36010.0234 -
val_loss: 75.1879 - val_mse: 20771.3613
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 107.1042 - mse: 37744.2812 -
val_loss: 84.2929 - val_mse: 23961.7070
205/205 ━━━━━━━━━━━━━━━━━━━━0s 861us/step
mean absolute error: 85.65982506626715

round-8
197/197 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 93.8510 - mse: 31955.5801 -
val_loss: 81.4964 - val_mse: 22498.3809
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 93.7150 - mse: 31830.2754 -
val_loss: 86.8676 - val_mse: 25771.4941
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 99.3833 - mse: 33347.2812 -
val_loss: 73.8934 - val_mse: 20775.6758
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 94.1546 - mse: 32034.3281 -
val_loss: 80.7916 - val_mse: 20978.3203
205/205 ━━━━━━━━━━━━━━━━━━━━0s 1ms/step
mean absolute error: 80.75668998620459

round-9
197/197 ━━━━━━━━━━━━━━━━━━━━3s 3ms/step - loss: 89.2915 - mse: 29862.0137 -
val_loss: 80.6704 - val_mse: 22601.8359
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 93.2114 - mse: 30925.1406 -
val_loss: 82.5277 - val_mse: 22394.1934
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━2s 4ms/step - loss: 93.8271 - mse: 31397.1289 -
val_loss: 72.4711 - val_mse: 20201.0527
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 8ms/step - loss: 92.5356 - mse: 30636.1094 -
val_loss: 79.2461 - val_mse: 22204.5312
205/205 ━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 78.0527554146315

round-10
197/197 ━━━━━━━━━━━━━━━━━━━━5s 6ms/step - loss: 83.8716 - mse: 26467.2344 -
val_loss: 74.6025 - val_mse: 18811.3516
```

```
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 6ms/step - loss: 87.7970 - mse: 28296.1328 -
val_loss: 79.2129 - val_mse: 20999.1055
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━4s 4ms/step - loss: 90.3667 - mse: 29001.6094 -
val_loss: 69.3493 - val_mse: 18718.8438
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 9ms/step - loss: 89.8697 - mse: 28744.7422 -
val_loss: 77.3529 - val_mse: 21439.0156
205/205 ━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 74.65601496342454

round-11
197/197 ━━━━━━━━━━━━━━━━━━━━7s 13ms/step - loss: 88.7163 - mse: 29052.4043 -
val_loss: 72.3964 - val_mse: 18210.3066
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━9s 16ms/step - loss: 85.2988 - mse: 26050.7598 -
val_loss: 77.8555 - val_mse: 20666.6777
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━7s 7ms/step - loss: 89.6634 - mse: 29603.3926 -
val_loss: 66.9394 - val_mse: 17440.5586
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━7s 14ms/step - loss: 91.0631 - mse: 30036.4629 -
val_loss: 77.1756 - val_mse: 21594.3848
205/205 ━━━━━━━━━━━━━━━━━━━━1s 4ms/step
mean absolute error: 72.13476359934135

round-12
197/197 ━━━━━━━━━━━━━━━━━━━━10s 6ms/step - loss: 84.7592 - mse: 27112.5176 -
val_loss: 82.8562 - val_mse: 24133.1855
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 82.1973 - mse: 26041.2422 -
val_loss: 75.8934 - val_mse: 19379.1348
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 89.7582 - mse: 28549.9980 -
val_loss: 67.8563 - val_mse: 18566.8574
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 91.6280 - mse: 30826.8477 -
val_loss: 73.2507 - val_mse: 19185.4629
205/205 ━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 72.06679822699672

round-13
197/197 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 83.7112 - mse: 26664.6094 -
val_loss: 70.4985 - val_mse: 17268.6523
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 85.8669 - mse: 27105.2715 -
val_loss: 76.5680 - val_mse: 20238.1094
***********************************
```

```
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 84.4624 - mse: 27408.5977 -
val_loss: 70.6761 - val_mse: 19614.8223
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━5s 8ms/step - loss: 86.7172 - mse: 27934.4922 -
val_loss: 70.1440 - val_mse: 17645.0957
205/205 ━━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 70.43618192613945

round-14
197/197 ━━━━━━━━━━━━━━━━━━━━━5s 5ms/step - loss: 82.5608 - mse: 25813.1387 -
val_loss: 68.7721 - val_mse: 16183.0576
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━6s 11ms/step - loss: 81.2297 - mse: 25255.8945 -
val_loss: 77.0573 - val_mse: 20678.8359
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 86.2848 - mse: 27147.3965 -
val_loss: 67.2649 - val_mse: 17913.1133
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━3s 5ms/step - loss: 87.4811 - mse: 27886.2559 -
val_loss: 69.6195 - val_mse: 17697.6836
205/205 ━━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 69.23357723842814

round-15
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 82.4325 - mse: 26039.7246 -
val_loss: 72.1929 - val_mse: 18072.1309
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 84.3492 - mse: 26722.7383 -
val_loss: 78.4618 - val_mse: 21928.8594
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 88.4608 - mse: 28456.9980 -
val_loss: 64.8702 - val_mse: 16622.6875
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━3s 5ms/step - loss: 82.9506 - mse: 25086.6914 -
val_loss: 72.8318 - val_mse: 19079.5703
205/205 ━━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 69.15213053911096

round-16
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 81.2611 - mse: 25100.7539 -
val_loss: 68.3108 - val_mse: 15433.6152
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━3s 5ms/step - loss: 83.9423 - mse: 26195.8926 -
val_loss: 75.1787 - val_mse: 19878.1094
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 84.7350 - mse: 26670.3438 -
val_loss: 66.9168 - val_mse: 18284.1562
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 82.3146 - mse: 26734.1230 -
```

```
val_loss: 74.6226 - val_mse: 20308.7148
205/205 ━━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 68.46796918973021

round-17
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 78.8232 - mse: 24276.0859 -
val_loss: 67.2111 - val_mse: 15475.2930
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 79.0833 - mse: 24477.2148 -
val_loss: 73.4085 - val_mse: 17973.5488
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 83.1587 - mse: 26450.3789 -
val_loss: 64.1276 - val_mse: 16466.1074
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 80.8243 - mse: 24050.3359 -
val_loss: 69.4968 - val_mse: 17607.9941
205/205 ━━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 66.7237194253728

round-18
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 82.7439 - mse: 26384.1094 -
val_loss: 71.4253 - val_mse: 17375.2227
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 78.7571 - mse: 23564.3027 -
val_loss: 75.1038 - val_mse: 19873.0020
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 82.2802 - mse: 25712.4785 -
val_loss: 62.7195 - val_mse: 16106.6416
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 83.3680 - mse: 26162.7422 -
val_loss: 70.7192 - val_mse: 17899.1895
205/205 ━━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 66.93038947523533

round-19
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 79.6779 - mse: 24114.6680 -
val_loss: 65.4622 - val_mse: 14807.0195
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 78.4816 - mse: 24317.9863 -
val_loss: 72.1457 - val_mse: 18217.3145
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 80.7039 - mse: 24963.2012 -
val_loss: 64.1389 - val_mse: 16275.1182
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 83.2614 - mse: 26036.5059 -
val_loss: 70.0419 - val_mse: 17355.7793
205/205 ━━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 65.24333144410373
```

Federated Learning on Solar Radiation Prediction

```
round-20
197/197 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 80.4273 - mse: 25360.5547 -
val_loss: 70.1131 - val_mse: 16789.3398
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 77.2632 - mse: 22931.9004 -
val_loss: 70.7607 - val_mse: 17218.8906
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 82.4332 - mse: 25265.4238 -
val_loss: 65.7184 - val_mse: 17277.0098
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 81.0173 - mse: 24503.5293 -
val_loss: 66.9613 - val_mse: 16457.2402
205/205 ━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 65.75033116335385
```

```
mae = mean_absolute_error(ytest,model_server.predict(xtest))
print(f'mean absolute error: {mae}')
```

```
205/205 ━━━━━━━━━━━━━━━━━━━━0s 2ms/step
mean absolute error: 65.75033116335385
```

without federated learning

```
model = None
model = Sequential()

model.add(Dense(128, activation='relu', input_dim=14))
model.add(Dropout(0.33))

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.33))

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.33))

model.add(Dense(1, activation='linear'))

model.compile(metrics=['mse'], loss='mae', optimizer=Adam(learning_rate=0.001))
print(model.summary())
```

```
Model: "sequential_29"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━
━━┓
┃ Layer (type)                            ┃ Output Shape                 ┃         Param
#  ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━
━━┩
│ dense_62 (Dense)                        │ (None, 128)                  │
1,920 │
├────────────────────────────────────────┼──────────────────────────────┼───────────────
```

```
──┤
│ dropout_33 (Dropout)                  │ (None, 128)                  │
0 │
├───────────────────────────────────────┼──────────────────────────────┼───────────────
──┤
│ dense_63 (Dense)                      │ (None, 64)                   │
8,256 │
├───────────────────────────────────────┼──────────────────────────────┼───────────────
──┤
│ dropout_34 (Dropout)                  │ (None, 64)                   │
0 │
├───────────────────────────────────────┼──────────────────────────────┼───────────────
──┤
│ dense_64 (Dense)                      │ (None, 32)                   │
2,080 │
├───────────────────────────────────────┼──────────────────────────────┼───────────────
──┤
│ dropout_35 (Dropout)                  │ (None, 32)                   │
0 │
├───────────────────────────────────────┼──────────────────────────────┼───────────────
──┤
│ dense_65 (Dense)                      │ (None, 1)                    │
33 │
└───────────────────────────────────────┴──────────────────────────────┴───────────────
──┘

 Total params: 12,289 (48.00 KB)

 Trainable params: 12,289 (48.00 KB)

 Non-trainable params: 0 (0.00 B)

None
```

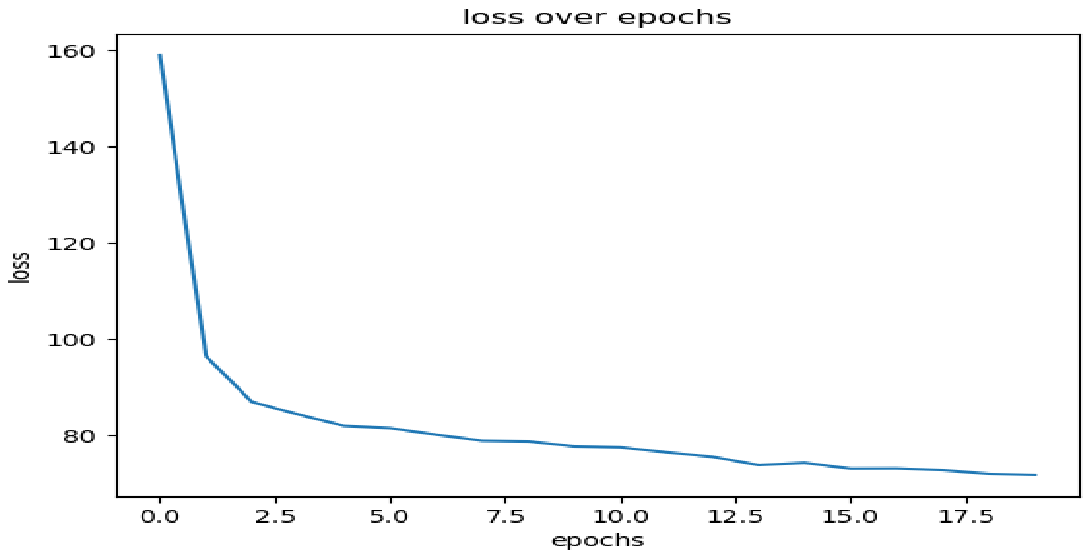
```
history = model.fit(xtrain, ytrain, validation_split=0.1, epochs=20, batch_size=32)
```

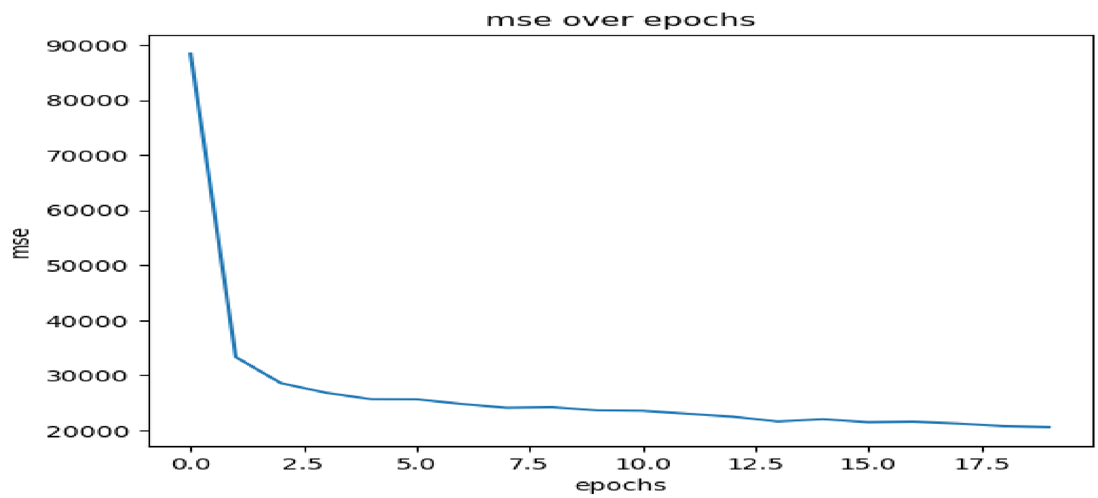
```
Epoch 1/20
736/736 ━━━━━━━━━━━━━━━━━━━━6s 4ms/step - loss: 190.6389 - mse: 123219.3047 -
val_loss: 105.3094 - val_mse: 37469.4766
Epoch 2/20
736/736 ━━━━━━━━━━━━━━━━━━━━3s 4ms/step - loss: 100.7433 - mse: 36108.3516 -
val_loss: 82.1599 - val_mse: 24055.6445
Epoch 3/20
736/736 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 88.2746 - mse: 29381.3398 -
val_loss: 77.9851 - val_mse: 22345.6465
Epoch 4/20
736/736 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 84.2945 - mse: 26790.3398 -
val_loss: 72.8098 - val_mse: 19216.8633
Epoch 5/20
736/736 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 83.6006 - mse: 26539.6172 -
val_loss: 76.2510 - val_mse: 22209.7754
```

```
Epoch 6/20
736/736 ━━━━━━━━━━━━━━━━━━━━3s 4ms/step - loss: 83.3967 - mse: 27009.0859 -
val_loss: 69.3737 - val_mse: 17559.5254
Epoch 7/20
736/736 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 79.5685 - mse: 24627.1641 -
val_loss: 68.4454 - val_mse: 17488.5039
Epoch 8/20
736/736 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 79.5342 - mse: 24548.6133 -
val_loss: 66.2488 - val_mse: 16435.6035
Epoch 9/20
736/736 ━━━━━━━━━━━━━━━━━━━━3s 3ms/step - loss: 78.5831 - mse: 24085.8164 -
val_loss: 65.6828 - val_mse: 16828.9258
Epoch 10/20
736/736 ━━━━━━━━━━━━━━━━━━━━2s 3ms/step - loss: 77.5231 - mse: 23938.4512 -
val_loss: 64.2911 - val_mse: 15771.8076
Epoch 11/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 77.7460 - mse: 23420.3535 -
val_loss: 66.7774 - val_mse: 17313.7363
Epoch 12/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 76.2985 - mse: 22938.1406 -
val_loss: 62.5187 - val_mse: 15705.9824
Epoch 13/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 74.1837 - mse: 22170.0000 -
val_loss: 66.5449 - val_mse: 17240.8262
Epoch 14/20
736/736 ━━━━━━━━━━━━━━━━━━━━2s 2ms/step - loss: 74.4020 - mse: 21776.6953 -
val_loss: 63.7792 - val_mse: 15675.7646
Epoch 15/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 75.3618 - mse: 22456.7500 -
val_loss: 63.2864 - val_mse: 15618.5303
Epoch 16/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 71.8913 - mse: 20707.0605 -
val_loss: 61.3848 - val_mse: 14785.3848
Epoch 17/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 71.3504 - mse: 21132.4180 -
val_loss: 57.0063 - val_mse: 13382.3145
Epoch 18/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 73.2032 - mse: 21140.2656 -
val_loss: 58.6813 - val_mse: 13638.2451
Epoch 19/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 72.7626 - mse: 21102.1543 -
val_loss: 60.0144 - val_mse: 14325.8633
Epoch 20/20
736/736 ━━━━━━━━━━━━━━━━━━━━1s 2ms/step - loss: 71.6579 - mse: 20849.5391 -
val_loss: 61.7661 - val_mse: 14871.8535
```

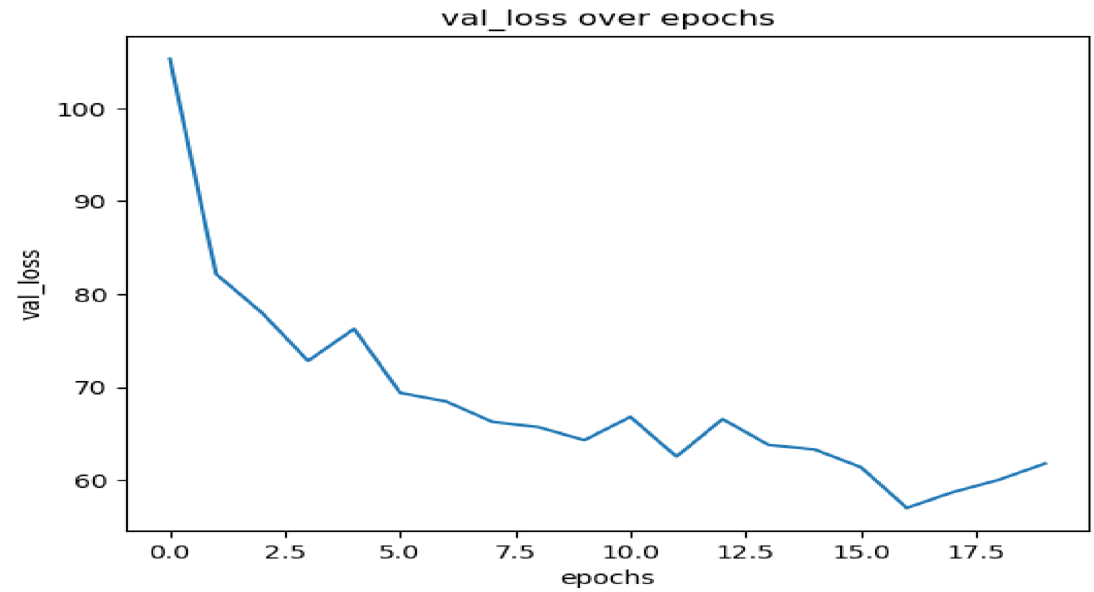
```python
fit = history.history
for i in fit:
    plt.plot(fit[i])
    plt.title(i + ' over epochs')
```

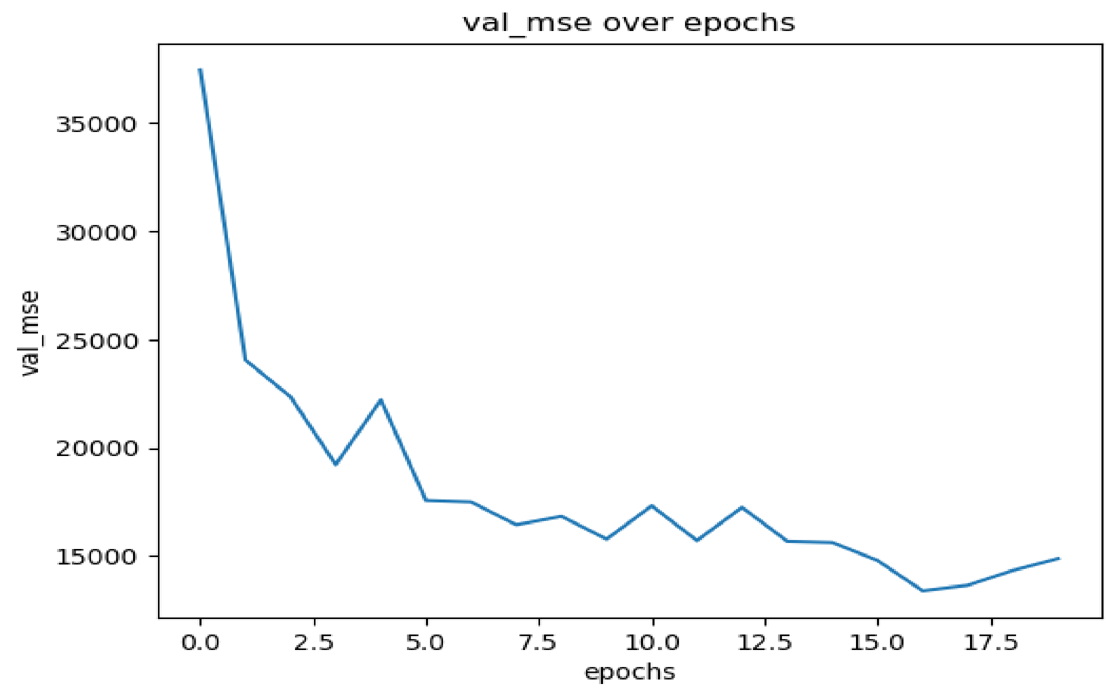
```
plt.ylabel(i)
plt.xlabel('epochs')
plt.show()
```





Federated Learning on Solar Radiation Prediction






```
mean_absolute_error(ytest, model.predict(xtest))

205/205 ━━━━━━━━━━━━━━━━━━━━0s 1ms/step

58.33900969015126
```

Federated learning using Ridge Regressor

```
#model training & evaluation
model_server=Ridge(alpha=0.1)
model_server.fit(data_x,data_y)
```

Federated Learning on Solar Radiation Prediction

```
print(model_server.intercept_)

model_c1=Ridge(alpha=0.1)
model_c2=Ridge(alpha=0.1)
model_c3=Ridge(alpha=0.1)
model_c4=Ridge(alpha=0.1)

rounds=5
for i in range(rounds):
    print('')
    print('round-{}'.format(i+1))
    #client-1 model
    model_c1.intercept_=model_server.intercept_
    model_c1.coef_=model_server.coef_
    model_c1.fit(c1_x,c1_y)

    #clinet-2 model
    model_c2.intercept_=model_server.intercept_
    model_c2.coef_=model_server.coef_
    model_c2.fit(c2_x,c2_y)

    #client-3 model
    model_c3.intercept_=model_server.intercept_
    model_c3.coef_=model_server.coef_
    model_c3.fit(c3_x,c3_y)

    #client-4 model
    model_c4.intercept_=model_server.intercept_
    model_c4.coef_=model_server.coef_
    model_c4.fit(c4_x,c4_y)

    #averaging the parameters
    stacked_arrays = np.vstack((model_c1.coef_, model_c2.coef_,
model_c3.coef_,model_c4.coef_,model_server.coef_))
    # Calculate the average along the specified axis (axis=0 for columns)
    server_coef = np.mean(stacked_arrays, axis=0)

server_intercept=(model_c1.intercept_+model_c2.intercept_+model_c3.intercept_+model_c
4.intercept_+model_server.intercept_)/5
    print('*****************************************')
    print("Average coef values of server model:", server_coef)
    print("Average intercept values of server model:", (server_intercept))

    #update parameters of server_model

    model_server.intercept_=server_intercept
    model_server.coef_=server_coef

    ypred=model_server.predict(xtest)
```

Federated Learning on Solar Radiation Prediction

```
    #evaluation parameters
    rmse =np.sqrt(mean_squared_error(ytest, ypred))
    r2 = r2_score(ytest, ypred)

    print('\n')
    print("Testing performance")

    print("RMSE: {:.2f}".format(rmse))
    print("R2: {:.2f}".format(r2))
```

```
15.097673430802814

round-1
******************************************
Average coef values of server model: [1370.57941212  -82.60898539   84.43679592 -
193.02907039   10.95450608
 -232.54613168  -21.35805463 -149.57396399  -19.51325568    0.
    0.          361.119728   -140.84259531 -107.27794544]
Average intercept values of server model: -236.70306532454782


Testing performance
RMSE: 198.54
R2: 0.60

round-2
******************************************
Average coef values of server model: [1505.69659233 -114.27387848   53.919519   -
103.21710589   79.99425813
 -305.93358809  -17.47848648 -163.2943551    -5.2151603     0.
    0.          419.27536632 -184.3138304  -137.85179391]
Average intercept values of server model: -287.06321307561797


Testing performance
RMSE: 193.62
R2: 0.62

round-3
******************************************
Average coef values of server model: [1532.72002837 -120.6068571    47.81606362  -
85.254713     93.80220853
 -320.61107937  -16.70257286 -166.03843332   -2.35554122    0.
    0.          430.90649398 -193.00807742 -143.9665636 ]
Average intercept values of server model: -297.135242625832
```

```
Testing performance
RMSE: 193.45
R2: 0.62

round-4
*****************************************
Average coef values of server model: [1538.12471558 -121.87345282   46.59537254  -
81.66223442   96.56379862
 -323.54657763  -16.54739013 -166.58724896   -1.78361741    0.
    0.          433.23271951 -194.74692682 -145.18951754]
Average intercept values of server model: -299.1496485358748


Testing performance
RMSE: 193.44
R2: 0.62

round-5
*****************************************
Average coef values of server model: [1539.20565302 -122.12677197   46.35123433  -
80.9437387    97.11611663
 -324.13367728  -16.51635359 -166.69701209   -1.66923265    0.
    0.          433.69796462 -195.0946967  -145.43410833]
Average intercept values of server model: -299.5525297178834


Testing performance
RMSE: 193.45
R2: 0.62
```

```
ypred=model_server.predict(xtest)

#evaluation parameters
rmse =np.sqrt(mean_squared_error(ytest, ypred))
r2 = r2_score(ytest, ypred)
mae = mean_absolute_error(ytest,ypred)

print('\n')
print("Testing performance")

print("RMSE: {:.2f}".format(rmse))
print("R2: {:.2f}".format(r2))
print("mae: {:.2f}".format(mae))
```

```
Testing performance
RMSE: 193.45
```

```
R2: 0.62
mae: 146.50
```

without federated learning

```
#model training
model=Ridge(alpha=0.1)
model.fit(xtrain,ytrain)
print(model.coef_)
print(model.intercept_)
```

```
[1539.46224508 -122.20711507   46.38998569  -81.18860454   98.23282163
 -336.59574379  -20.79263869 -167.28228575   -2.10200582    0.
    0.          445.68508995 -199.83031209 -150.57793504]
-291.3252597794941
```

```
#model testing
ypred=model.predict(xtest)

#evaluation parameters
rmse =np.sqrt(mean_squared_error(ytest, ypred))
r2 = r2_score(ytest, ypred)
mae = mean_absolute_error(ytest,ypred)

print("Testing performance")

print("RMSE: {:.2f}".format(rmse))
print("R2: {:.2f}".format(r2))
print("mae: {:.2f}".format(mae))
```

```
Testing performance
RMSE: 193.44
R2: 0.62
mae: 146.50
```

Federated Learning using RNN

```
def create_model(input_shape):
    model = tf.keras.Sequential([
        tf.keras.layers.SimpleRNN(64, input_shape=input_shape),
        tf.keras.layers.Dense(1)
    ])
    return model

input_shape = (10, 1)

#model training & evaluation
model_server=create_model(input_shape)
model_server.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))

model_c1=create_model(input_shape)
```

```
model_c2=create_model(input_shape)
model_c3=create_model(input_shape)
model_c4=create_model(input_shape)

#train_dataset = create_dataset(train_data, WINDOW_SIZE).batch(BATCH_SIZE)
#val_dataset = create_dataset(val_data, WINDOW_SIZE).batch(BATCH_SIZE)

rounds=20
for i in range(rounds):
    print('')
    print('round-{}'.format(i+1))
    #client-1 model
    model_c1.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))
    model_c1.set_weights(model_server.get_weights())

    model_c1.fit(c1_x,c1_y,validation_split=0.1, epochs=1, batch_size=32)

    print('**********************************')
    #clinet-2 model
    model_c2.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))
    model_c2.set_weights(model_server.get_weights())
    model_c2.fit(c2_x,c2_y,validation_split=0.1, epochs=1, batch_size=32)

    print('************************************')
    #client-3 model
    model_c3.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))
    model_c3.set_weights(model_server.get_weights())
    model_c3.fit(c3_x,c3_y,validation_split=0.1, epochs=1, batch_size=32)

    print('************************************')
    #client-4 model
    model_c4.compile(metrics=['mse'], loss='mae',
optimizer=Adam(learning_rate=0.001))
    model_c4.set_weights(model_server.get_weights())
    model_c4.fit(c4_x,c4_y,validation_split=0.1, epochs=1, batch_size=32)

    #averaging the parameters


avg_weights=[(model_c1.get_weights()[i]+model_c2.get_weights()[i]+model_c3.get_weight
s()[i]+model_c4.get_weights()[i]+model_server.get_weights()[i])/5 for i in
range(len(model_server.get_weights()))]

    model_server.set_weights(avg_weights)
```

```python
    mae = mean_absolute_error(ytest,model_server.predict(xtest))
    print(f'mean absolute error: {mae}')
```

```
round-1
197/197 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 198.9961 - mse: 134301.5938 - val_loss: 211.9183 - val_mse: 143623.0625
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 207.0446 - mse: 144008.6875 - val_loss: 227.1699 - val_mse: 155532.6875
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 207.2077 - mse: 143191.3594 - val_loss: 191.9134 - val_mse: 126908.5391
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 219.5626 - mse: 155212.9844 - val_loss: 211.0187 - val_mse: 142148.1562
205/205 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
mean absolute error: 200.86843470853748

round-2
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 194.6102 - mse: 131241.9531 - val_loss: 209.5619 - val_mse: 141475.0938
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 194.5598 - mse: 129852.3438 - val_loss: 223.7124 - val_mse: 151085.5938
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 204.0407 - mse: 138237.8906 - val_loss: 188.8907 - val_mse: 123650.5469
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 209.1131 - mse: 141785.5938 - val_loss: 207.8045 - val_mse: 138676.7188
205/205 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
mean absolute error: 198.0135531790762

round-3
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 206.8800 - mse: 140276.3438 - val_loss: 207.9254 - val_mse: 139829.2344
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 202.9610 - mse: 135289.3125 - val_loss: 221.0057 - val_mse: 148268.3594
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 203.2859 - mse: 137368.7031 - val_loss: 186.9907 - val_mse: 120648.9844
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 201.1902 - mse: 137454.9844 - val_loss: 205.0765 - val_mse: 135424.3750
205/205 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
mean absolute error: 195.5655428580171
```

```
round-4
197/197 ━━━━━━━━━━━━━━━━━━━━4s 8ms/step - loss: 187.7156 - mse: 121725.5234 -
val_loss: 205.4653 - val_mse: 135181.7812
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 200.7611 - mse: 134418.3750 -
val_loss: 222.3219 - val_mse: 144508.4219
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━4s 8ms/step - loss: 203.0480 - mse: 135610.6719 -
val_loss: 185.3230 - val_mse: 119903.7031
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 9ms/step - loss: 206.8904 - mse: 142472.2344 -
val_loss: 203.5565 - val_mse: 132163.9844
205/205 ━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 193.82291897339974

round-5
197/197 ━━━━━━━━━━━━━━━━━━━━5s 8ms/step - loss: 193.1654 - mse: 128574.8672 -
val_loss: 203.4036 - val_mse: 132710.4375
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 189.0858 - mse: 122658.6484 -
val_loss: 217.1902 - val_mse: 142480.8906
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 8ms/step - loss: 203.3484 - mse: 135165.0781 -
val_loss: 182.5062 - val_mse: 116044.7578
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 9ms/step - loss: 200.2258 - mse: 133921.9688 -
val_loss: 201.3929 - val_mse: 129857.1641
205/205 ━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 192.19286694295732

round-6
197/197 ━━━━━━━━━━━━━━━━━━━━4s 8ms/step - loss: 190.0846 - mse: 125220.7031 -
val_loss: 200.9153 - val_mse: 131343.3906
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 188.9658 - mse: 124149.5781 -
val_loss: 214.8518 - val_mse: 140711.9375
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━6s 10ms/step - loss: 194.0970 - mse: 127858.4844 -
val_loss: 181.3022 - val_mse: 114187.5000
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 202.2410 - mse: 135411.2344 -
val_loss: 198.6187 - val_mse: 128959.9062
205/205 ━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 189.63298739041346

round-7
197/197 ━━━━━━━━━━━━━━━━━━━━5s 8ms/step - loss: 190.4332 - mse: 124977.6328 -
val_loss: 207.8416 - val_mse: 127823.5859
*********************************
```

```
197/197 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - loss: 190.9418 - mse: 124924.6094 -
val_loss: 213.6406 - val_mse: 139745.9375
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 198.9374 - mse: 131678.0312 -
val_loss: 180.0951 - val_mse: 113532.8594
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 193.7701 - mse: 127913.8672 -
val_loss: 196.9430 - val_mse: 125919.8984
205/205 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
mean absolute error: 188.37716666057193

round-8
197/197 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - loss: 187.2531 - mse: 121286.1797 -
val_loss: 196.8640 - val_mse: 126727.2109
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - loss: 188.2975 - mse: 121951.4844 -
val_loss: 211.6209 - val_mse: 136004.2344
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 195.7352 - mse: 127075.3906 -
val_loss: 177.3530 - val_mse: 110791.2266
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 197.5573 - mse: 129815.5078 -
val_loss: 196.1411 - val_mse: 122964.1797
205/205 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
mean absolute error: 186.82492496897078

round-9
197/197 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - loss: 183.8313 - mse: 116075.5078 -
val_loss: 195.8220 - val_mse: 124308.1562
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 185.1258 - mse: 118232.7734 -
val_loss: 209.0715 - val_mse: 134035.0625
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - loss: 200.1709 - mse: 131183.7188 -
val_loss: 174.9381 - val_mse: 108099.5859
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━ 5s 12ms/step - loss: 192.5552 - mse: 125981.6016 -
val_loss: 193.7582 - val_mse: 123052.7891
205/205 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
mean absolute error: 184.64896912783576

round-10
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 182.1345 - mse: 114174.2734 -
val_loss: 194.2906 - val_mse: 122394.8828
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - loss: 189.0209 - mse: 121449.2422 -
val_loss: 207.0425 - val_mse: 131335.2969
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - loss: 199.0601 - mse: 129433.5000 -
```

```
val_loss: 174.2624 - val_mse: 106070.2422
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━4s 10ms/step - loss: 197.5841 - mse: 126611.9922 - val_loss: 191.2265 - val_mse: 119542.2734
205/205 ━━━━━━━━━━━━━━━━━━━━━1s 4ms/step
mean absolute error: 183.10173161082622

round-11
197/197 ━━━━━━━━━━━━━━━━━━━━━5s 8ms/step - loss: 183.0441 - mse: 115578.7031 - val_loss: 191.7824 - val_mse: 121065.1172
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 8ms/step - loss: 188.4434 - mse: 120371.1641 - val_loss: 206.9195 - val_mse: 132073.1250
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 186.0895 - mse: 117468.2891 - val_loss: 172.7735 - val_mse: 104180.8906
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━4s 10ms/step - loss: 200.2372 - mse: 130151.8047 - val_loss: 189.4798 - val_mse: 117577.1797
205/205 ━━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 181.23415168176479

round-12
197/197 ━━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 179.4753 - mse: 110925.0625 - val_loss: 189.7012 - val_mse: 119020.7188
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 178.0103 - mse: 110536.0625 - val_loss: 202.8212 - val_mse: 127165.0156
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━5s 8ms/step - loss: 189.4712 - mse: 120819.7656 - val_loss: 173.3941 - val_mse: 106108.4609
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━4s 10ms/step - loss: 190.4860 - mse: 121953.8516 - val_loss: 188.8707 - val_mse: 115776.3906
205/205 ━━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 179.58284886212186

round-13
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 8ms/step - loss: 180.2385 - mse: 112018.5391 - val_loss: 188.4563 - val_mse: 115301.2578
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━4s 8ms/step - loss: 176.8313 - mse: 111153.8906 - val_loss: 200.6484 - val_mse: 124827.1406
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━━6s 10ms/step - loss: 188.9022 - mse: 119842.6953 - val_loss: 169.3185 - val_mse: 101530.3516
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━━5s 11ms/step - loss: 182.5607 - mse: 114073.2812 - val_loss: 186.0687 - val_mse: 112995.0312
```

```
205/205 ━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 177.55475977654922

round-14
197/197 ━━━━━━━━━━━━━━━━━━━━5s 10ms/step - loss: 180.7644 - mse: 112120.7266 -
val_loss: 185.1907 - val_mse: 113334.1797
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 178.3021 - mse: 111203.2812 -
val_loss: 202.6559 - val_mse: 126525.4141
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 179.0110 - mse: 110892.8125 -
val_loss: 167.2024 - val_mse: 99448.6250
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 10ms/step - loss: 191.6509 - mse: 119330.2656 -
val_loss: 185.6849 - val_mse: 114378.0938
205/205 ━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 176.70518038796956

round-15
197/197 ━━━━━━━━━━━━━━━━━━━━5s 10ms/step - loss: 173.1214 - mse: 105648.3125 -
val_loss: 184.2363 - val_mse: 112757.4609
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 179.7957 - mse: 111120.6484 -
val_loss: 198.0630 - val_mse: 119667.1094
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 183.6134 - mse: 113452.9688 -
val_loss: 166.8187 - val_mse: 98643.8203
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 10ms/step - loss: 181.6959 - mse: 112129.5469 -
val_loss: 185.1658 - val_mse: 108289.5703
205/205 ━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 174.1781631075348

round-16
197/197 ━━━━━━━━━━━━━━━━━━━━5s 10ms/step - loss: 169.8174 - mse: 101240.5234 -
val_loss: 183.3872 - val_mse: 111101.2266
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 173.9238 - mse: 105824.4141 -
val_loss: 195.6796 - val_mse: 119443.7344
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━6s 9ms/step - loss: 176.3634 - mse: 108432.2734 -
val_loss: 166.7121 - val_mse: 98624.6172
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━5s 10ms/step - loss: 185.3051 - mse: 114292.9062 -
val_loss: 179.8132 - val_mse: 107855.3672
205/205 ━━━━━━━━━━━━━━━━━━━━1s 4ms/step
mean absolute error: 172.73471805539467

round-17
```

```
197/197 ━━━━━━━━━━━━━━━━━━━━5s 10ms/step - loss: 172.1390 - mse: 101929.1562 -
val_loss: 179.9615 - val_mse: 108396.6172
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━6s 11ms/step - loss: 177.5094 - mse: 107509.8828 -
val_loss: 193.3047 - val_mse: 116565.7812
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 8ms/step - loss: 179.4089 - mse: 111083.5156 -
val_loss: 162.3445 - val_mse: 94529.1641
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 11ms/step - loss: 181.6313 - mse: 111221.0234 -
val_loss: 179.3335 - val_mse: 107714.4297
205/205 ━━━━━━━━━━━━━━━━━━━━1s 4ms/step
mean absolute error: 170.918409233331

round-18
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 172.6595 - mse: 103213.0391 -
val_loss: 179.5574 - val_mse: 108034.1484
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 170.9400 - mse: 101588.2500 -
val_loss: 193.1767 - val_mse: 115394.5938
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 8ms/step - loss: 181.4666 - mse: 111542.3125 -
val_loss: 161.1421 - val_mse: 91770.3203
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━5s 11ms/step - loss: 183.2261 - mse: 112185.1406 -
val_loss: 176.6903 - val_mse: 104162.5391
205/205 ━━━━━━━━━━━━━━━━━━━━1s 4ms/step
mean absolute error: 169.1172456017763

round-19
197/197 ━━━━━━━━━━━━━━━━━━━━5s 10ms/step - loss: 171.5367 - mse: 99159.3281 -
val_loss: 176.9853 - val_mse: 104851.4766
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 172.6839 - mse: 103312.7969 -
val_loss: 189.6765 - val_mse: 112423.2344
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 172.8570 - mse: 104217.7188 -
val_loss: 160.3317 - val_mse: 91844.5000
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━4s 10ms/step - loss: 181.0727 - mse: 109268.4297 -
val_loss: 174.4079 - val_mse: 101671.2812
205/205 ━━━━━━━━━━━━━━━━━━━━1s 4ms/step
mean absolute error: 167.3294116448663

round-20
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 170.9731 - mse: 101438.3672 -
val_loss: 174.6565 - val_mse: 101551.2266
*********************************
197/197 ━━━━━━━━━━━━━━━━━━━━6s 10ms/step - loss: 170.0791 - mse: 99777.3203 -
```

```
val_loss: 188.7623 - val_mse: 110268.5703
***********************************
197/197 ━━━━━━━━━━━━━━━━━━━━5s 9ms/step - loss: 175.2561 - mse: 104800.6719 -
val_loss: 157.4140 - val_mse: 88853.1094
***********************************
145/145 ━━━━━━━━━━━━━━━━━━━━5s 11ms/step - loss: 176.6441 - mse: 104872.6406 -
val_loss: 172.1677 - val_mse: 99801.7500
205/205 ━━━━━━━━━━━━━━━━━━━━1s 4ms/step
mean absolute error: 165.62572284290644
```

```
mae = mean_absolute_error(ytest,model_server.predict(xtest))
print(f'mean absolute error: {mae}')
```

```
205/205 ━━━━━━━━━━━━━━━━━━━━1s 3ms/step
mean absolute error: 165.62572284290644
```

without federated learning

```
model=create_model(input_shape)
model.compile(metrics=['mse'], loss='mae', optimizer=Adam(learning_rate=0.001))
model.fit(xtrain,ytrain,validation_split=0.1, epochs=20, batch_size=32)
```

```
Epoch 1/20
736/736 ━━━━━━━━━━━━━━━━━━━━8s 7ms/step - loss: 200.0496 - mse: 135148.5469 -
val_loss: 206.1861 - val_mse: 135081.4844
Epoch 2/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 194.3742 - mse: 125674.7109 -
val_loss: 200.4387 - val_mse: 121345.0312
Epoch 3/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 185.6413 - mse: 117063.5938 -
val_loss: 191.8227 - val_mse: 119027.8516
Epoch 4/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 176.6500 - mse: 104634.8125 -
val_loss: 183.5354 - val_mse: 109115.6328
Epoch 5/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 172.0884 - mse: 99623.5547 -
val_loss: 193.9531 - val_mse: 120066.3828
Epoch 6/20
736/736 ━━━━━━━━━━━━━━━━━━━━5s 6ms/step - loss: 170.2103 - mse: 99905.4688 -
val_loss: 173.2588 - val_mse: 97589.8516
Epoch 7/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 168.9972 - mse: 88949.0859 -
val_loss: 169.9329 - val_mse: 91880.8203
Epoch 8/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 157.8313 - mse: 83516.6406 -
val_loss: 162.6765 - val_mse: 83573.6328
Epoch 9/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 152.3613 - mse: 78047.7812 -
val_loss: 155.7840 - val_mse: 80383.3594
Epoch 10/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 142.3953 - mse: 69390.5078 -
```

```
val_loss: 146.0643 - val_mse: 70289.3438
Epoch 11/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 136.9454 - mse: 63943.3516 -
val_loss: 142.9894 - val_mse: 67345.3281
Epoch 12/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 131.3447 - mse: 59684.5781 -
val_loss: 140.5509 - val_mse: 64119.5000
Epoch 13/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 128.6766 - mse: 56884.4883 -
val_loss: 136.8489 - val_mse: 61594.6602
Epoch 14/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 127.1648 - mse: 54735.8164 -
val_loss: 130.2390 - val_mse: 56198.0039
Epoch 15/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 121.4602 - mse: 50890.1250 -
val_loss: 127.0576 - val_mse: 51860.9570
Epoch 16/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 119.2822 - mse: 49487.2070 -
val_loss: 129.8987 - val_mse: 50149.4688
Epoch 17/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 114.3145 - mse: 45902.6016 -
val_loss: 119.1315 - val_mse: 44869.6328
Epoch 18/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 112.5773 - mse: 44415.4883 -
val_loss: 122.8419 - val_mse: 49896.7500
Epoch 19/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 6ms/step - loss: 110.7061 - mse: 43165.5273 -
val_loss: 113.5357 - val_mse: 43552.7031
Epoch 20/20
736/736 ━━━━━━━━━━━━━━━━━━━━4s 5ms/step - loss: 109.2288 - mse: 41376.8320 -
val_loss: 118.7419 - val_mse: 45115.1680
```

```
<keras.src.callbacks.history.History at 0x20519ce6c90>
```

```python
mae = mean_absolute_error(ytest,model.predict(xtest))
print(f'mean absolute error: {mae}')
```

```
205/205 ━━━━━━━━━━━━━━━━━━━━1s 4ms/step
mean absolute error: 110.38326333556199
```

Federated Learning on Solar Radiation Prediction

Results:

Model Performances on Solar Radiation Prediction:

Type of Model	With Federated Learning	Without Federated Learning
Linear Regression	***RMSE: 193.74 R2: 0.62 MAE: 146.72***	***RMSE: 193.44 R2: 0.62 MAE: 146.50***
Multi-Layer Perceptron	***Mean Absolute Error (MAE): 65.75033116335385***	***Mean Absolute Error (MAE): 58.33900969015126***
Ridge Regression	***RMSE: 193.45 R2: 0.62 MAE: 146.50***	***RMSE: 193.44 R2: 0.62 MAE: 146.50***
Recurrent Neural Network	***Mean Absolute Error (MAE): 165.62572284290644***	***Mean Absolute Error (MAE): 110.38326333556199***

GitHub Link: https://github.com/balaji-reddy-helper/Federated_Learning