```
#importing libraries
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style('darkgrid')
sns.set_palette('Set2')

import os
print("Module Loaded")
```

```
Module Loaded
```

```
#reading data
data=pd.read_csv('/content/DailyDelhiClimateTrain.csv')
data.head()
```

	date	meantemp	humidity	wind_speed	meanpressure
0	2013-01-01	10.000000	84.500000	0.000000	1015.666667
1	2013-01-02	7.400000	92.000000	2.980000	1017.800000
2	2013-01-03	7.166667	87.000000	4.633333	1018.666667
3	2013-01-04	8.666667	71.333333	1.233333	1017.166667
4	2013-01-05	6.000000	86.833333	3.700000	1016.500000

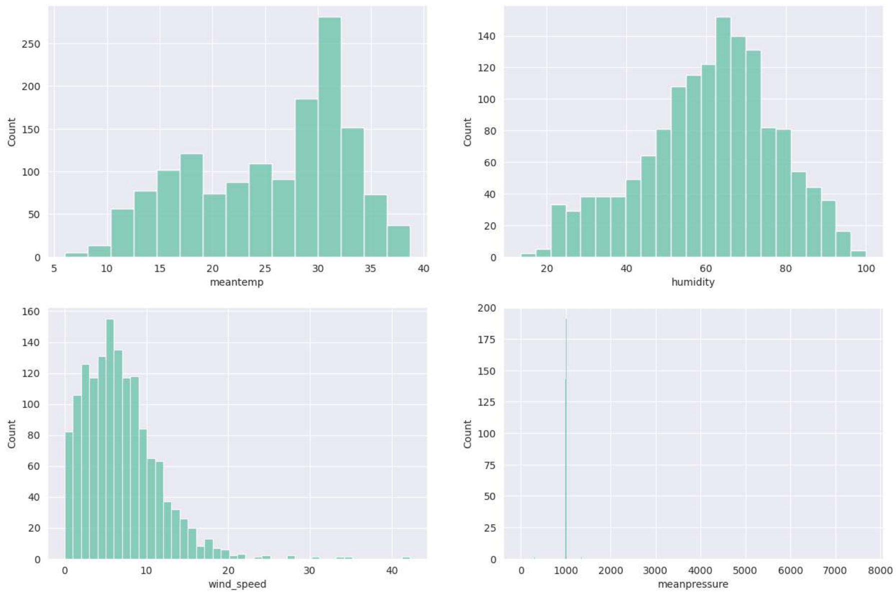
```
data.describe()
```

	meantemp	humidity	wind_speed	meanpressure
count	1462.000000	1462.000000	1462.000000	1462.000000
mean	25.495521	60.771702	6.802209	1011.104548
std	7.348103	16.769652	4.561602	180.231668
min	6.000000	13.428571	0.000000	-3.041667
25%	18.857143	50.375000	3.475000	1001.580357
50%	27.714286	62.625000	6.221667	1008.563492
75%	31.305804	72.218750	9.238235	1014.944901
max	38.714286	100.000000	42.220000	7679.333333

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1462 entries, 0 to 1461
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   date          1462 non-null   object
 1   meantemp      1462 non-null   float64
 2   humidity      1462 non-null   float64
 3   wind_speed    1462 non-null   float64
 4   meanpressure  1462 non-null   float64
dtypes: float64(4), object(1)
memory usage: 57.2+ KB
```

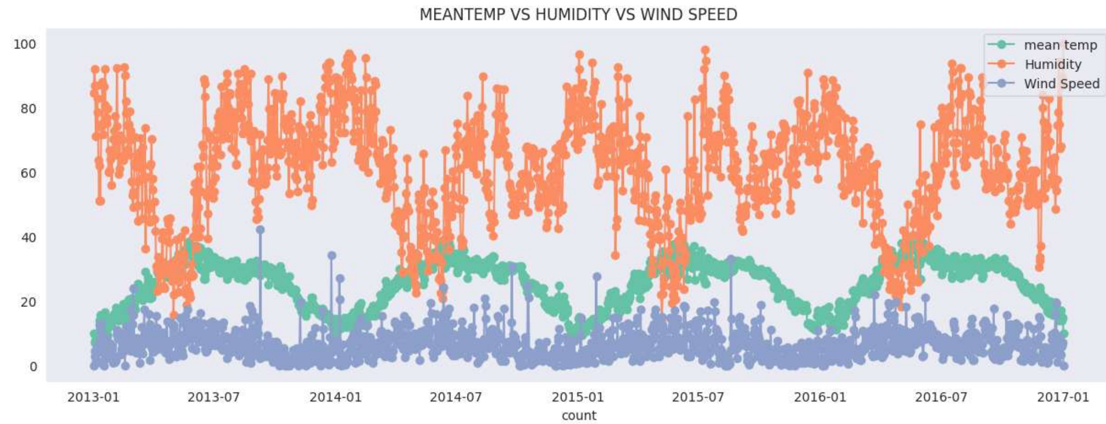
```
#ploting figures for different columns
plt.figure(figsize=(15,10))
i=1
for col in data.iloc[:,1:5]:
    plt.subplot(2,2,i)
    sns.histplot(data[col])
    i=i+1
```




```
#setting index values equal to value of column 'date'
data.index=pd.to_datetime(data.date)
data.head()
```

date	date	meantemp	humidity	wind_speed	meanpressure
2013-01-01	2013-01-01	10.000000	84.500000	0.000000	1015.666667
2013-01-02	2013-01-02	7.400000	92.000000	2.980000	1017.800000
2013-01-03	2013-01-03	7.166667	87.000000	4.633333	1018.666667
2013-01-04	2013-01-04	8.666667	71.333333	1.233333	1017.166667
2013-01-05	2013-01-05	6.000000	86.833333	3.700000	1016.500000

```
#plotting graph to show realtion between temperature, humidity and wind speed
plt.figure(figsize=(15,5))
plt.plot(data['meantemp'],marker="o",label="mean temp")
plt.plot(data['humidity'],marker="o",label="Humidity")
plt.plot(data['wind_speed'],marker="o",label="Wind Speed")
plt.title("MEANTEMP VS HUMIDITY VS WIND SPEED")
plt.xlabel("count")
plt.grid()
plt.legend()
plt.show()
```



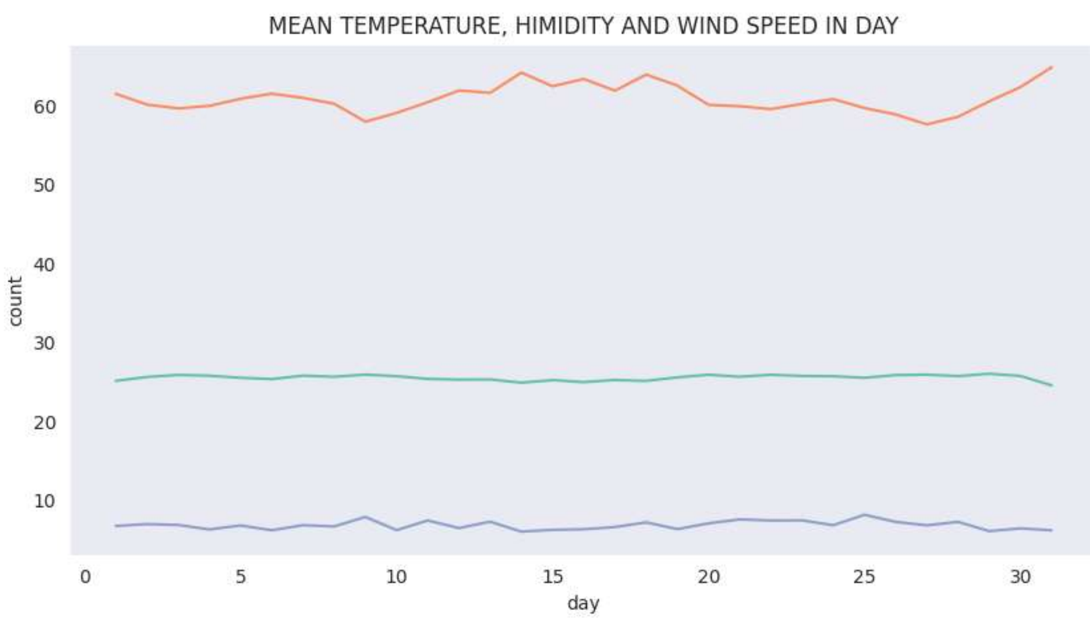

```
#creating new columns(day,hour, month) in reaaltion with column date
data['date'] = pd.to_datetime(data['date'])
data['day'] = data['date'].dt.day
data['hour'] = data['date'].dt.hour
data['month'] = data['date'].dt.month
data.head()
```

	date	meantemp	humidity	wind_speed	meanpressure	day	hour	month
date								
2013-01-01	2013-01-01	10.000000	84.500000	0.000000	1015.666667	1	0	1
2013-01-02	2013-01-02	7.400000	92.000000	2.980000	1017.800000	2	0	1
2013-01-03	2013-01-03	7.166667	87.000000	4.633333	1018.666667	3	0	1
2013-01-04	2013-01-04	8.666667	71.333333	1.233333	1017.166667	4	0	1
2013-01-05	2013-01-05	6.000000	86.833333	3.700000	1016.500000	5	0	1

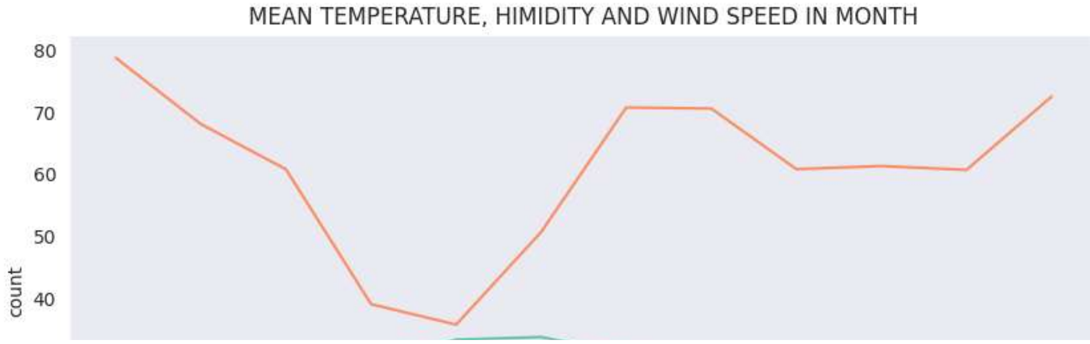
```
data.groupby(by="day").mean()
```

	meantemp	humidity	wind_speed	meanpressure	hour	month
day						
1	25.072710	61.433882	6.691956	1008.890207	0.0	6.387755
2	25.570504	60.079116	6.927360	994.497353	0.0	6.500000
3	25.827786	59.588013	6.815051	1008.446359	0.0	6.500000
4	25.727867	59.914409	6.253566	1007.884475	0.0	6.500000
5	25.463945	60.817339	6.756849	1008.004427	0.0	6.500000
6	25.301865	61.454747	6.149787	1008.110992	0.0	6.500000
7	25.735067	60.925444	6.792539	1008.246721	0.0	6.500000
8	25.594226	60.183539	6.628434	1008.337854	0.0	6.500000
9	25.873889	57.910425	7.839333	1006.665426	0.0	6.500000

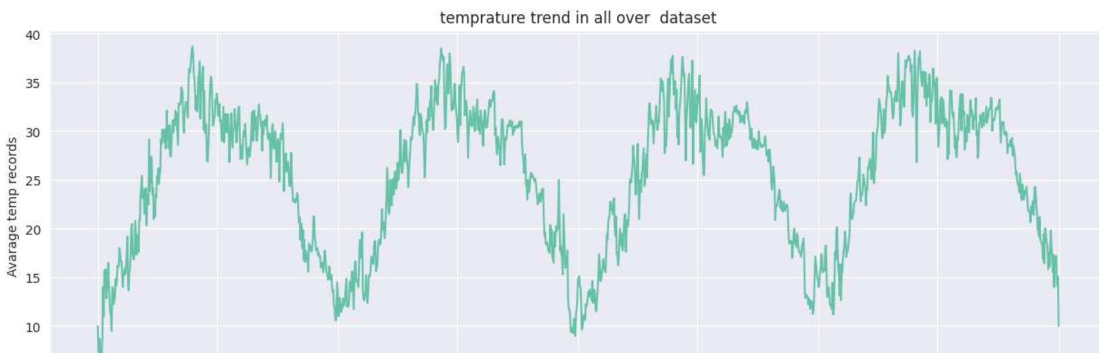
```
#making a lineplot of mean of temperature,humidity and wind speed of data groupby day
plt.figure(figsize=(10,5))
plt.title("MEAN TEMPERATURE, HIMIDITY AND WIND SPEED IN DAY")
sns.lineplot(data.groupby(by="day")['meantemp'].mean())
sns.lineplot(data.groupby(by="day")['humidity'].mean())
sns.lineplot(data.groupby(by="day")['wind_speed'].mean())
plt.ylabel('count')
plt.grid()
plt.show()
```



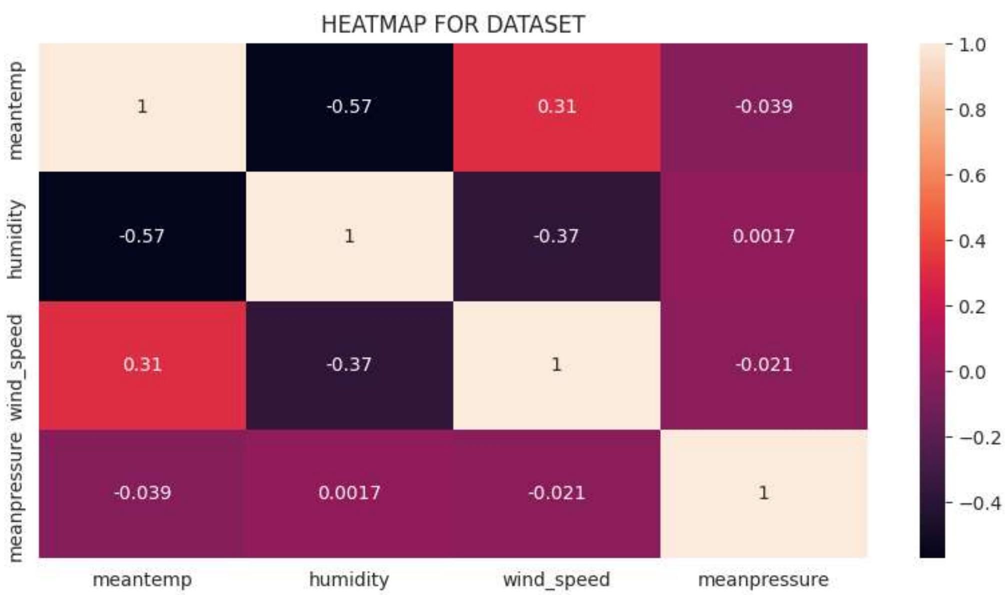

```
#ploting a lineplot of mean temperature, humidity and eind speed of data group by month
plt.figure(figsize=(10,5))
plt.title("MEAN TEMPERATURE, HIMIDITY AND WIND SPEED IN MONTH")
sns.lineplot(data.groupby(by="month")['meantemp'].mean())
sns.lineplot(data.groupby(by="month")['humidity'].mean())
sns.lineplot(data.groupby(by="month")['wind_speed'].mean())
plt.ylabel('count')
plt.grid()
plt.show()
```




```
#showing temperature trend in different months all over dataset
tempdata=data[['meantemp']]
tempdata.index=pd.to_datetime(data.date)
plt.figure(figsize=(15,5))
plt.xlabel("Time Frame")
plt.ylabel("Avarage temp records")
plt.title("temprature trend in all over  dataset")
plt.plot(tempdata)
plt.show()
```




```
plt.figure(figsize=(10,5))
plt.title("HEATMAP FOR DATASET")
sns.heatmap(data.iloc[:,1:5].corr(),annot=True);
plt.show();
```

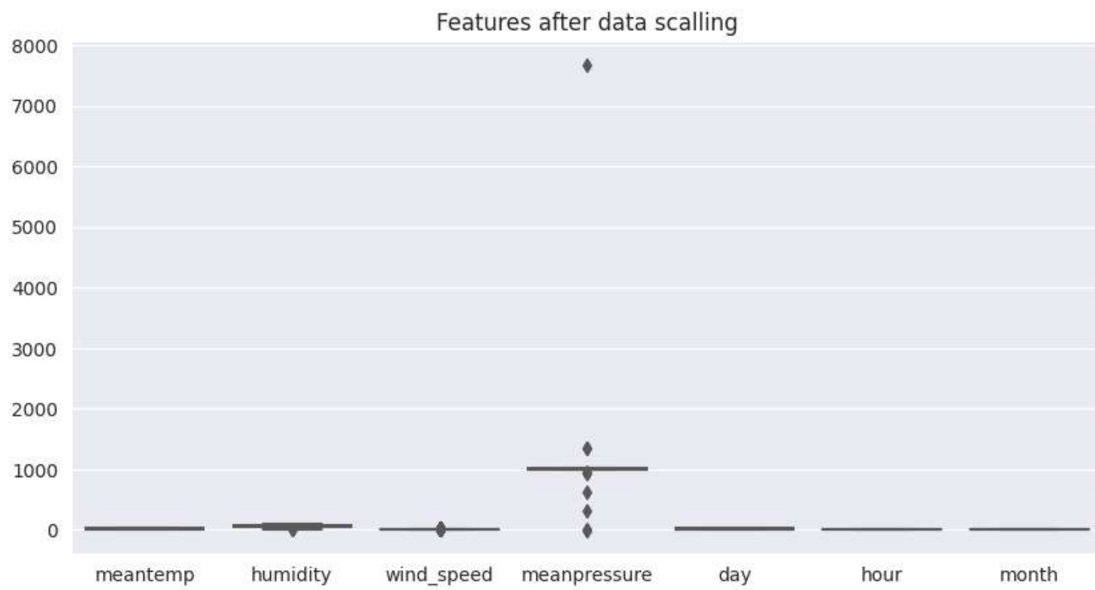



```
x=data.drop(['meantemp','date'],axis=1)
y=data['meantemp']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
#feature scaling of datasets
from sklearn.preprocessing import  StandardScaler
scaler=StandardScaler()
```

```
x_train=pd.DataFrame(scaler.fit_transform(x_train),columns=x_train.columns)
x_test=pd.DataFrame(scaler.fit_transform(x_test),columns=x_test.columns)
x_test.head()
```

	humidity	wind_speed	meanpressure	day	hour	month
0	1.054984	-1.331483	-0.019473	0.631268	0.0	1.635681
1	-0.946817	-0.912609	-0.029285	-0.066784	0.0	1.345554
2	-1.834889	1.079874	-0.055187	0.398584	0.0	-0.395207
3	1.530216	-0.580259	-0.027995	-1.695572	0.0	-0.975461
4	-0.466385	2.441216	-0.054795	1.329320	0.0	-0.685334

```
plt.figure(figsize=(10,5))
plt.title("Features after data scalling")
sns.boxplot(data)
plt.show()
```




```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

#linear regression is used here to prediction
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
lr=LinearRegression()
lr.fit(x_train,y_train)
print('score: ',lr.score(x_test,y_test))
pred=lr.predict(x_test)
print("mean absolute error:",mean_absolute_error(y_test,pred))
print("r2 score: ",r2_score(y_test,pred))
```

```
score:  0.33834827653407495
mean absolute error: 4.819502009335955
r2 score:  0.33834827653407495
```

```
# using kneighbors regression since it gives us more score
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor()
knn.fit(x_train,y_train)
print('score:',knn.score(x_test,y_test))
```

```
score: 0.8840271162237024
```

```
from sklearn.ensemble import RandomForestRegressor
rr=RandomForestRegressor()
rr.fit(x_train,y_train)
print("score:",rr.score(x_test,y_test))
```

```
score: 0.9220413139648098
```

```
import tensorflow  #tensorflow for deep learning
from tensorflow import keras # for making neurons network
from keras import Sequential
from keras.layers import Dense# for adding layes
model=Sequential()
#relu activation function is used
model.add(Dense(18,activation="relu",input_dim=x_train.shape[1]))
model.add(Dense(9,activation="relu"))
model.add(Dense(4,activation="relu"))
model.add(Dense(1,activation="linear"))# for regression problem output activation is linear
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 18)                126

 dense_1 (Dense)             (None, 9)                 171

 dense_2 (Dense)             (None, 4)                 40

 dense_3 (Dense)             (None, 1)                 5

=================================================================
Total params: 342
Trainable params: 342
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss="mean_squared_error",optimizer="Adam",metrics=['accuracy'])
history=model.fit(x_train,y_train,epochs=100,validation_split=0.2)
```

```
Epoch 1/100
26/26 [==============================] - 2s 13ms/step - loss: 23668.0938 - accuracy: 0.0000e+00 - val_loss: 6420.3135 - val_accuracy:
Epoch 2/100
26/26 [==============================] - 0s 4ms/step - loss: 1973.2231 - accuracy: 0.0000e+00 - val_loss: 57.2835 - val_accuracy: 0.0
Epoch 3/100
26/26 [==============================] - 0s 5ms/step - loss: 168.2191 - accuracy: 0.0000e+00 - val_loss: 59.5602 - val_accuracy: 0.00
Epoch 4/100
26/26 [==============================] - 0s 4ms/step - loss: 130.5052 - accuracy: 0.0000e+00 - val_loss: 60.1788 - val_accuracy: 0.00
Epoch 5/100
26/26 [==============================] - 0s 4ms/step - loss: 115.3021 - accuracy: 0.0000e+00 - val_loss: 55.7183 - val_accuracy: 0.00
Epoch 6/100
26/26 [==============================] - 0s 4ms/step - loss: 119.6406 - accuracy: 0.0000e+00 - val_loss: 57.8150 - val_accuracy: 0.00
Epoch 7/100
26/26 [==============================] - 0s 4ms/step - loss: 119.0507 - accuracy: 0.0000e+00 - val_loss: 59.0607 - val_accuracy: 0.00
Epoch 8/100
26/26 [==============================] - 0s 4ms/step - loss: 112.1202 - accuracy: 0.0000e+00 - val_loss: 63.6156 - val_accuracy: 0.00
Epoch 9/100
26/26 [==============================] - 0s 4ms/step - loss: 110.6992 - accuracy: 0.0000e+00 - val_loss: 54.2500 - val_accuracy: 0.00
Epoch 10/100
26/26 [==============================] - 0s 5ms/step - loss: 113.2010 - accuracy: 0.0000e+00 - val_loss: 56.7223 - val_accuracy: 0.00
Epoch 11/100
26/26 [==============================] - 0s 4ms/step - loss: 112.7476 - accuracy: 0.0000e+00 - val_loss: 62.0758 - val_accuracy: 0.00
Epoch 12/100
26/26 [==============================] - 0s 5ms/step - loss: 113.2405 - accuracy: 0.0000e+00 - val_loss: 63.9634 - val_accuracy: 0.00
Epoch 13/100
26/26 [==============================] - 0s 4ms/step - loss: 109.4575 - accuracy: 0.0000e+00 - val_loss: 53.8424 - val_accuracy: 0.00
Epoch 14/100
26/26 [==============================] - 0s 4ms/step - loss: 111.2756 - accuracy: 0.0000e+00 - val_loss: 64.1716 - val_accuracy: 0.00
Epoch 15/100
26/26 [==============================] - 0s 4ms/step - loss: 116.8366 - accuracy: 0.0000e+00 - val_loss: 53.3634 - val_accuracy: 0.00
Epoch 16/100
26/26 [==============================] - 0s 3ms/step - loss: 115.1342 - accuracy: 0.0000e+00 - val_loss: 76.1934 - val_accuracy: 0.00
Epoch 17/100
26/26 [==============================] - 0s 4ms/step - loss: 119.6554 - accuracy: 0.0000e+00 - val_loss: 54.6963 - val_accuracy: 0.00
Epoch 18/100
26/26 [==============================] - 0s 4ms/step - loss: 108.4904 - accuracy: 0.0000e+00 - val_loss: 55.5999 - val_accuracy: 0.00
Epoch 19/100
26/26 [==============================] - 0s 5ms/step - loss: 109.3023 - accuracy: 0.0000e+00 - val_loss: 53.7656 - val_accuracy: 0.00
Epoch 20/100
26/26 [==============================] - 0s 4ms/step - loss: 107.5772 - accuracy: 0.0000e+00 - val_loss: 73.8421 - val_accuracy: 0.00
Epoch 21/100
26/26 [==============================] - 0s 6ms/step - loss: 109.7241 - accuracy: 0.0000e+00 - val_loss: 55.2927 - val_accuracy: 0.00
Epoch 22/100
26/26 [==============================] - 0s 4ms/step - loss: 117.6778 - accuracy: 0.0000e+00 - val_loss: 54.0281 - val_accuracy: 0.00
Epoch 23/100
26/26 [==============================] - 0s 4ms/step - loss: 107.4337 - accuracy: 0.0000e+00 - val_loss: 67.9536 - val_accuracy: 0.00
```

```
Epoch 24/100
26/26 [==============================] - 0s 4ms/step - loss: 110.3257 - accuracy: 0.0000e+00 - val_loss: 73.0165 - val_accuracy: 0.00
Epoch 25/100
26/26 [==============================] - 0s 4ms/step - loss: 101.5673 - accuracy: 0.0000e+00 - val_loss: 53.4765 - val_accuracy: 0.00
Epoch 26/100
26/26 [==============================] - 0s 4ms/step - loss: 109.9839 - accuracy: 0.0000e+00 - val_loss: 58.7043 - val_accuracy: 0.00
Epoch 27/100
26/26 [==============================] - 0s 4ms/step - loss: 97.8430 - accuracy: 0.0000e+00 - val_loss: 54.7570 - val_accuracy: 0.000
Epoch 28/100
26/26 [==============================] - 0s 4ms/step - loss: 96.7450 - accuracy: 0.0000e+00 - val_loss: 64.3545 - val_accuracy: 0.000
Epoch 29/100
```

```
pred=model.predict(x_test)
print("prediction",pred)
```

```
14/14 [==============================] - 0s 5ms/step
prediction [[19.703722 ]
 [18.310467 ]
 [25.195047 ]
 [23.110754 ]
 [20.403517 ]
 [23.065954 ]
 [22.590633 ]
 [22.74784  ]
 [23.15567  ]
 [24.026508 ]
 [12.961147 ]
 [21.169054 ]
 [21.103872 ]
 [19.40076  ]
 [22.669052 ]
 [21.418228 ]
 [24.359783 ]
 [24.460003 ]
 [19.86724  ]
 [22.402952 ]
 [21.007275 ]
 [21.958344 ]
 [20.695421 ]
 [19.58328  ]
 [22.720018 ]
 [20.712967 ]
 [27.432983 ]
 [23.056883 ]
 [25.299078 ]
 [24.214247 ]
 [23.482222 ]
 [22.28459  ]
 [17.962723 ]
 [21.493301 ]
 [22.707104 ]
 [22.551487 ]
 [20.457592 ]
 [24.658422 ]
 [26.028872 ]
 [23.500477 ]
 [20.154633 ]
 [19.69206  ]
 [21.66318  ]
 [23.722866 ]
 [22.43694  ]
 [21.920631 ]
 [20.940424 ]
 [19.865768 ]
 [22.837929 ]
 [22.333408 ]
 [21.872631 ]
 [22.680029 ]
 [21.633186 ]
 [21.056473 ]
 [21.819698 ]
 [21.242508 ]
 [18.53904  ]
```

```
print("score:",r2_score(y_test,pred))
```

```
score: -0.06747339689724208
```

```
plt.figure(figsize=(10,5))
i=1
history_list=["loss","accuracy"]
```

```
history_list1=["val_loss","val_accuracy"]

for option,val_option in zip(history_list,history_list1):
    plt.subplot(1,2,i)
    sns.lineplot(history.history[option],label=option)
    sns.lineplot(history.history[val_option],label=val_option)
    plt.xlabel("number of Epoch")
    plt.title(f"{option} and {val_option} ")
    i=i+1
```