DATA COLLECTION

Import libraries

In [135]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\solar-measurements_palau-airai_ppa_qc_y1_v01.csv")
data
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (14)
have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[3]:

	time	dhi_pyr	ghi_pyr_1	ghi_pyr_2	air_temperature	relative_humidity	barometric_pressure	precipitation	wind_speed	win
0	2020/02/01 00:01	0.0	0.0	0.0	24.31900	88.270	1006.328	0.0	0.341667	
1	2020/02/01 00:02	0.0	0.0	0.0	24.37833	88.476	1006.323	0.0	0.000000	
2	2020/02/01 00:03	0.0	0.0	0.0	24.37933	88.144	1006.300	0.0	0.000000	
3	2020/02/01 00:04	0.0	0.0	0.0	24.38700	87.848	1006.285	0.0	0.047500	
4	2020/02/01 00:05	0.0	0.0	0.0	24.39200	87.812	1006.266	0.0	0.095000	
...	...	...	...	...	...	...	...	...	...	
527035	2021/01/31 23:56	0.0	0.0	0.0	27.45466	80.193	1005.332	0.0	2.750000	
527036	2021/01/31 23:57	0.0	0.0	0.0	27.45867	80.449	1005.303	0.0	3.050000	
527037	2021/01/31 23:58	0.0	0.0	0.0	27.46733	80.478	1005.290	0.0	2.700000	
527038	2021/01/31 23:59	0.0	0.0	0.0	27.46933	80.574	1005.263	0.0	3.900000	
527039	2021/02/01 00:00	0.0	0.0	0.0	27.49917	80.574	1005.273	0.0	2.312500	

527040 rows × 15 columns

In [4]:
```
data.head(10)
```

Out[4]:

	time	dhi_pyr	ghi_pyr_1	ghi_pyr_2	air_temperature	relative_humidity	barometric_pressure	precipitation	wind_speed	wind_fr
0	2020/02/01 00:01	0.0	0.0	0.0	24.31900	88.270	1006.328	0.0	0.341667	
1	2020/02/01 00:02	0.0	0.0	0.0	24.37833	88.476	1006.323	0.0	0.000000	
2	2020/02/01 00:03	0.0	0.0	0.0	24.37933	88.144	1006.300	0.0	0.000000	
3	2020/02/01 00:04	0.0	0.0	0.0	24.38700	87.848	1006.285	0.0	0.047500	
4	2020/02/01 00:05	0.0	0.0	0.0	24.39200	87.812	1006.266	0.0	0.095000	
5	2020/02/01 00:06	0.0	0.0	0.0	24.41233	88.012	1006.251	0.0	0.174167	
6	2020/02/01 00:07	0.0	0.0	0.0	24.41400	88.148	1006.278	0.0	0.269167	
7	2020/02/01 00:08	0.0	0.0	0.0	24.37183	88.632	1006.242	0.0	0.000000	
8	2020/02/01 00:09	0.0	0.0	0.0	24.32883	88.324	1006.239	0.0	0.000000	
9	2020/02/01 00:10	0.0	0.0	0.0	24.31567	88.378	1006.227	0.0	0.195833	

In [5]:
```
data.tail(10)
```

Out[5]:

_temperature	relative_humidity	barometric_pressure	precipitation	wind_speed	wind_from_direction	gti_clean	gti_soil	gti_monthly	sens
27.51517	79.418	1005.338	0.0	3.0000	60.18936	0.0	0.0	0.0	
27.50400	79.560	1005.342	0.0	2.3250	53.10275	0.0	0.0	0.0	
27.46533	79.812	1005.328	0.0	2.4500	62.07859	0.0	0.0	0.0	
27.44666	79.982	1005.364	0.0	2.5875	63.04394	0.0	0.0	0.0	
27.44000	80.158	1005.339	0.0	2.6000	67.50400	0.0	0.0	0.0	
27.45466	80.193	1005.332	0.0	2.7500	68.45277	0.0	0.0	0.0	
27.45867	80.449	1005.303	0.0	3.0500	65.66559	0.0	0.0	0.0	
27.46733	80.478	1005.290	0.0	2.7000	65.63803	0.0	0.0	0.0	
27.46933	80.574	1005.263	0.0	3.9000	64.82665	0.0	0.0	0.0	
27.49917	80.574	1005.273	0.0	2.3125	64.97405	0.0	0.0	0.0	

DATA CLEANING AND PRE_PROCESSING

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 527040 entries, 0 to 527039
Data columns (total 15 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   time                 527040 non-null  object 
 1   dhi_pyr              527023 non-null  float64
 2   ghi_pyr_1            525725 non-null  float64
 3   ghi_pyr_2            525725 non-null  float64
 4   air_temperature      527010 non-null  float64
 5   relative_humidity    527019 non-null  float64
 6   barometric_pressure  527023 non-null  float64
 7   precipitation        527023 non-null  float64
 8   wind_speed           527023 non-null  float64
 9   wind_from_direction  527023 non-null  float64
 10  gti_clean            527023 non-null  float64
 11  gti_soil             527023 non-null  float64
 12  gti_monthly          527023 non-null  float64
 13  sensor_cleaning      527023 non-null  float64
 14  comments             1328 non-null    object 
dtypes: float64(13), object(2)
memory usage: 60.3+ MB
```

To display summary of statistics

In [7]:
```
data.describe()
```

Out[7]:

	dhi_pyr	ghi_pyr_1	ghi_pyr_2	air_temperature	relative_humidity	barometric_pressure	precipitation	wind_spe
count	527023.000000	525725.000000	525725.000000	527010.000000	527019.000000	527023.000000	527023.000000	527023.000(
mean	93.190033	208.434584	207.754793	27.195682	87.336787	1004.324044	0.007698	1.7268
std	131.142826	313.937158	313.537540	2.041439	9.323507	1.582703	0.067473	1.5347
min	0.000000	0.000000	0.000000	21.764330	48.429000	999.215900	0.000000	0.000(
25%	0.000000	0.000000	0.000000	25.465000	79.980000	1003.227000	0.000000	0.1266
50%	2.637488	2.495833	2.064672	26.901330	89.066000	1004.304000	0.000000	1.550(
75%	163.268200	337.135200	335.112900	28.956500	95.914000	1005.399000	0.000000	2.825(
max	789.664900	1622.539000	1623.983000	33.167500	99.998000	1009.690000	2.600000	12.4375

In [8]:
```
data.columns
```

Out[8]:
```
Index(['time', 'dhi_pyr', 'ghi_pyr_1', 'ghi_pyr_2', 'air_temperature',
       'relative_humidity', 'barometric_pressure', 'precipitation',
       'wind_speed', 'wind_from_direction', 'gti_clean', 'gti_soil',
       'gti_monthly', 'sensor_cleaning', 'comments'],
      dtype='object')
```

In [9]:
```
np.shape(data)
```

Out[9]:
```
(527040, 15)
```

In [10]:
```
np.size(data)
```

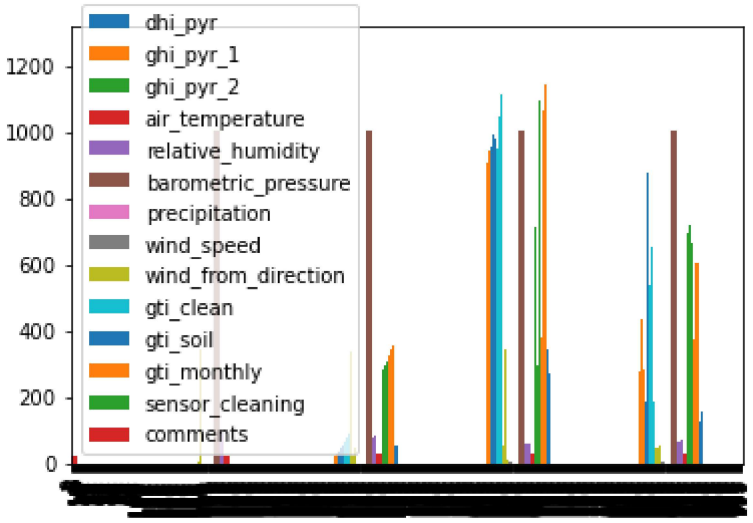
Out[10]:
```
7905600
```

In [69]:
```
dd=data.head(1000)
```

EDA and visualization

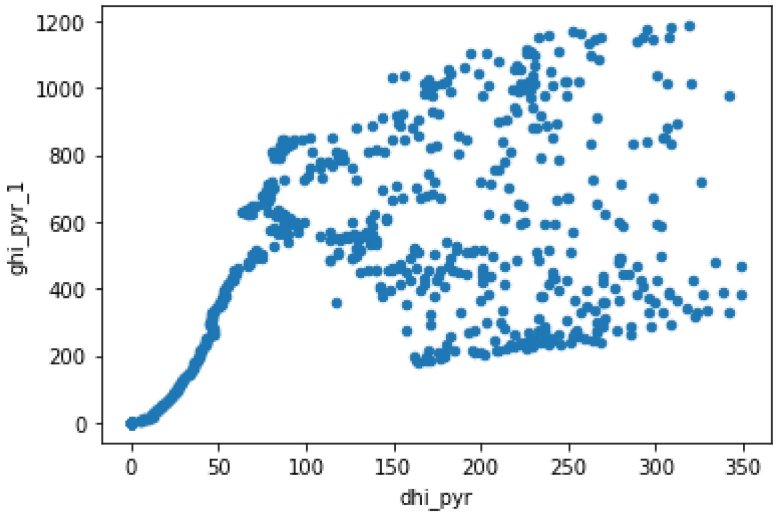
In [71]:
```
dd.plot.bar()
```

Out[71]: <AxesSubplot:>

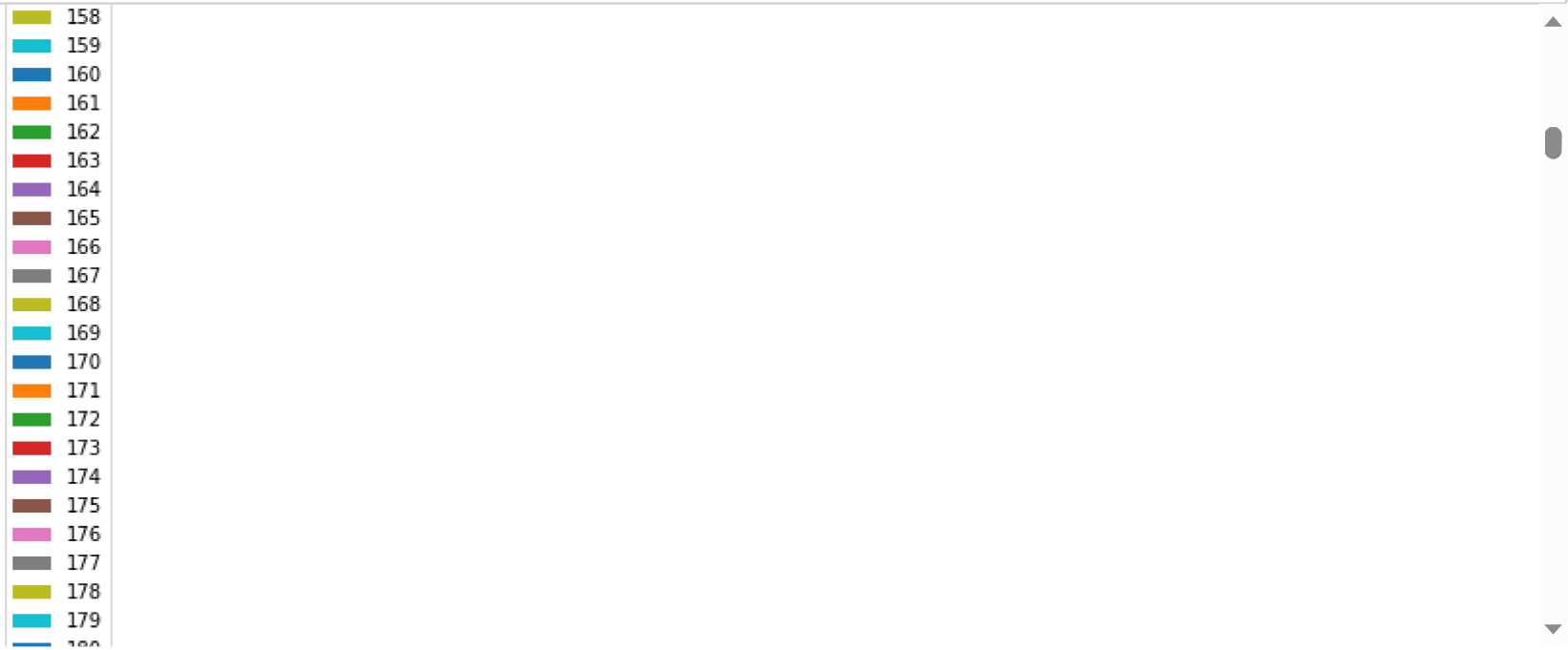


In [73]:
```
dd.plot.scatter(x='dhi_pyr',y='ghi_pyr_1')
```

Out[73]: <AxesSubplot:xlabel='dhi_pyr', ylabel='ghi_pyr_1'>

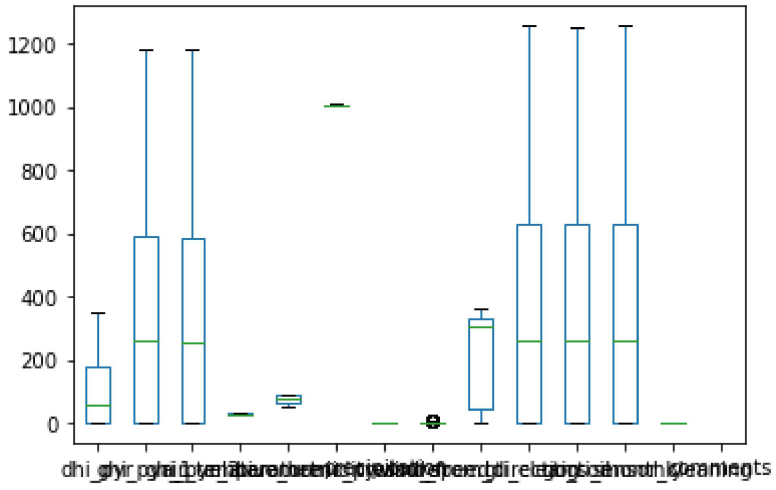


In [74]:
```
dd.plot.pie(y='air_temperature')
```



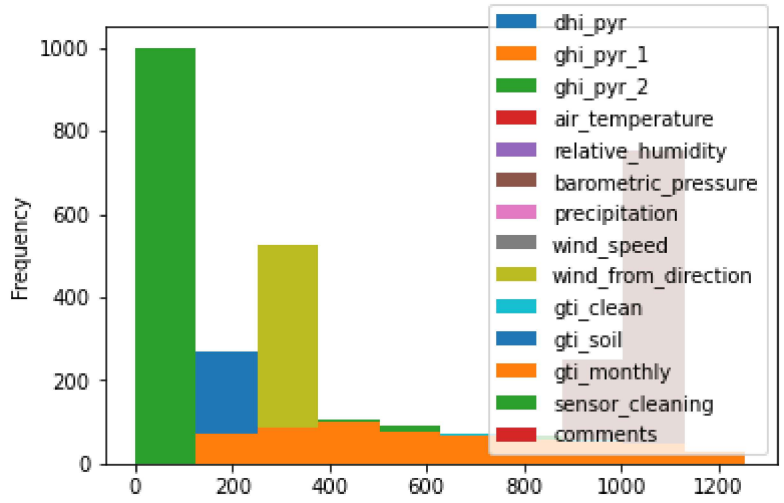
In [35]:
```
dd.plot.box()
```

Out[35]: <AxesSubplot:>



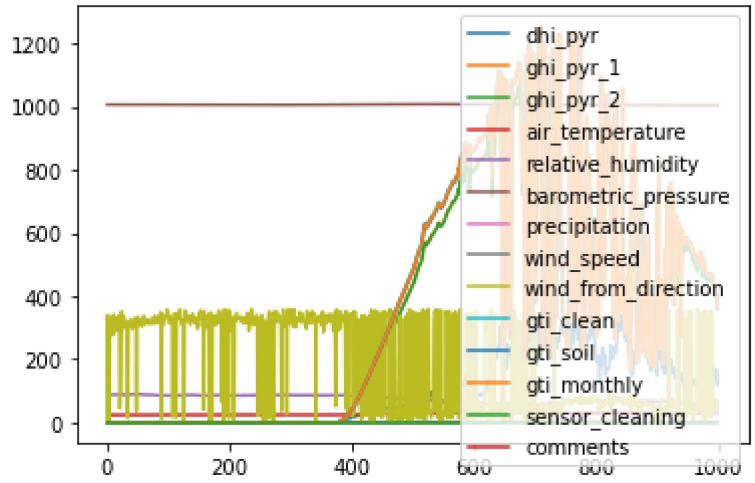
In [20]:
```
dd.plot.hist()
```

Out[20]: <AxesSubplot:ylabel='Frequency'>



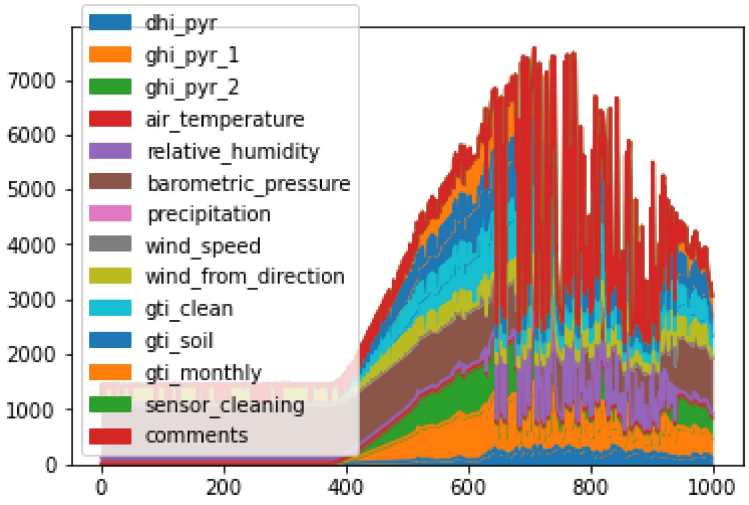
In [21]:
```
dd.plot.line()
```

Out[21]: <AxesSubplot:>



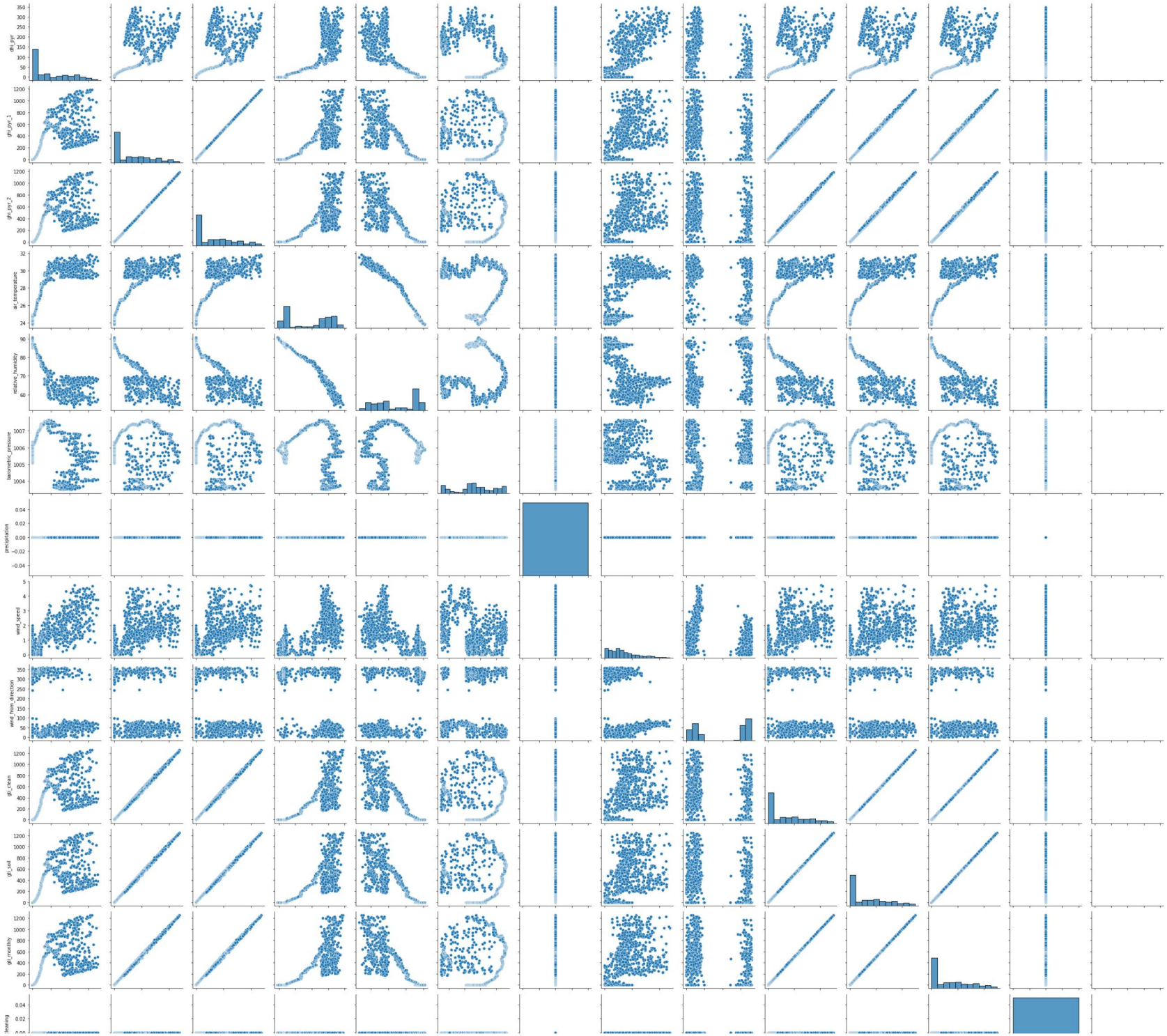
In [22]:
```
dd.plot.area()
```

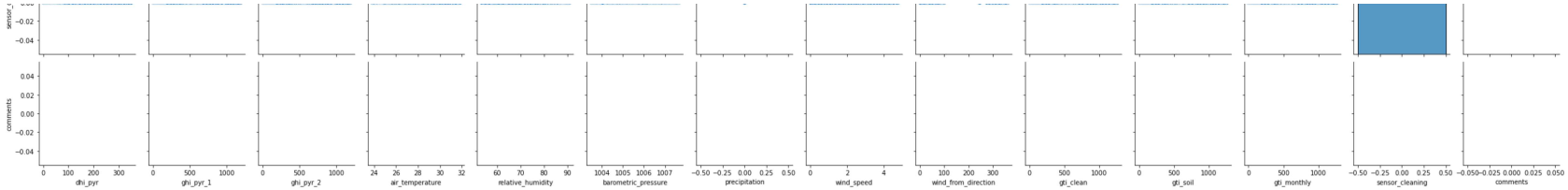
Out[22]: <AxesSubplot:>



In [23]:
```
sns.pairplot(dd)
```

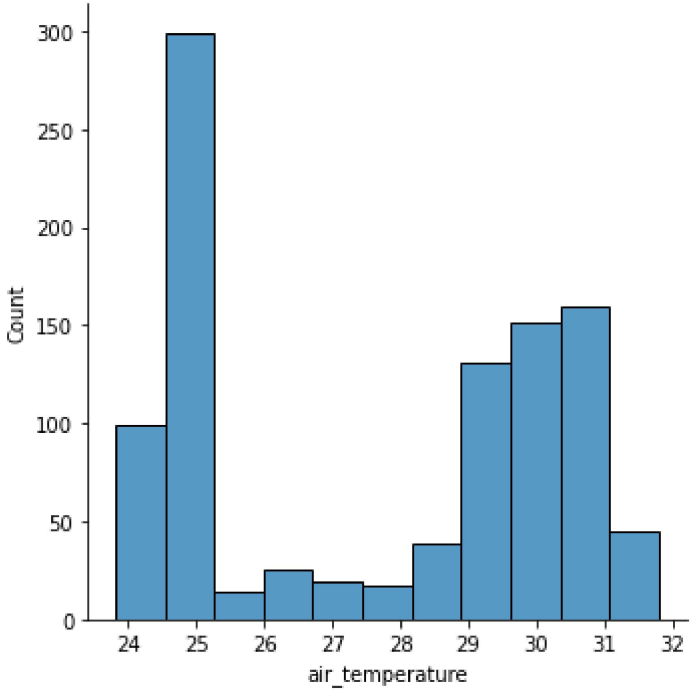
Out[23]:
```
<seaborn.axisgrid.PairGrid at 0x182b715fbb0>
```


poa_pyr
ghi_pyr_1
ghi_pyr_2
air_temperature
relative_humidity
barometric_pressure
precipitation
wind_speed
wind_from_direction
gti_clean
gti_soil
gti_monthly
cleaning



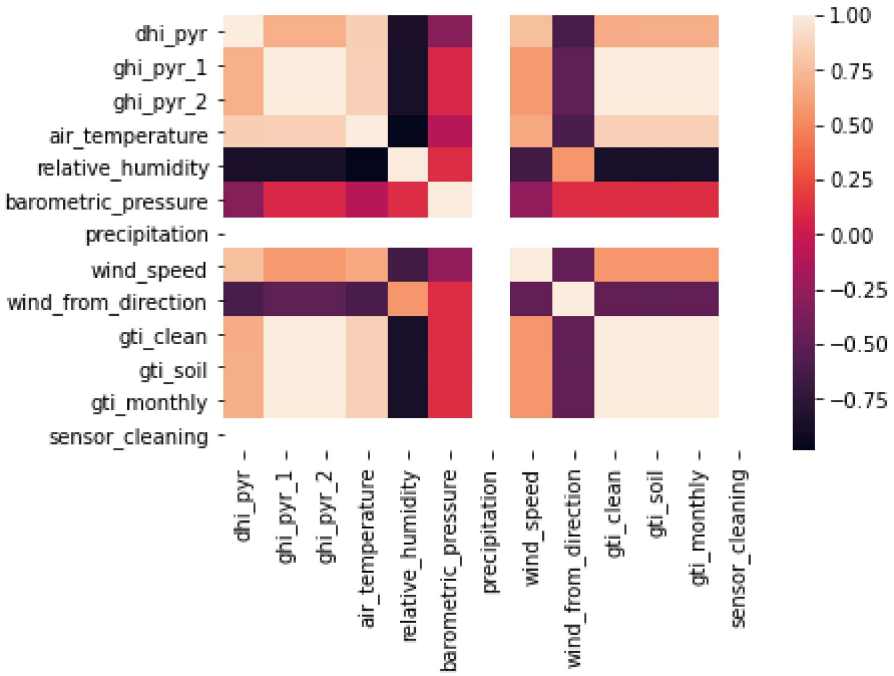
In [24]:
```
sns.displot(dd['air_temperature'])
```

Out[24]: <seaborn.axisgrid.FacetGrid at 0x182bd3db340>



In [25]:
```
sns.heatmap(dd.corr())
```

Out[25]: <AxesSubplot:>



TO TRAIN THE MODEL _MODEL BUILDING

In [84]:
```
x=dd[['air_temperature',
       'relative_humidity', 'barometric_pressure',
       'wind_speed']]
y=dd['wind_from_direction']
```

To split my dataset into training data and test data

In [85]:
```
from sklearn .model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [86]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[86]:
```
LinearRegression()
```

In [87]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_test,y_test)
```

Out[87]:
```
LinearRegression()
```

In [88]:
```
print(lr.intercept_)
```

```
-5939.835911360598
```

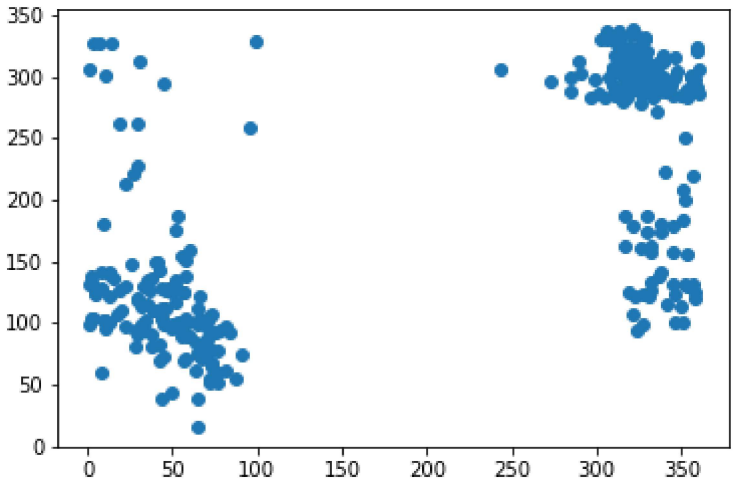
In [89]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[89]:

	Co-efficient
air_temperature	-109.782496
relative_humidity	-17.920616
barometric_pressure	10.453557
wind_speed	-4.907249

In [90]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[90]:
```
<matplotlib.collections.PathCollection at 0x182dfba6250>
```



In [91]:
```
print(lr.score(x_test,y_test))
```

```
0.47305341842940163
```

In [92]:
```
lr.score(x_test,y_test)
```

Out[92]:
```
0.47305341842940163
```

In [93]:
```
lr.score(x_train,y_train)
```

Out[93]:
```
0.3102617861732466
```

In [94]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [95]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[95]:
```
Ridge(alpha=10)
```

In [96]:
```
dr.score(x_test,y_test)
```

Out[96]:
```
0.4377531903485882
```

In [97]:
```
dr.score(x_train,y_train)
```

Out[97]:
```
0.348479433341745
```

In [98]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[98]:
```
Lasso(alpha=10)
```

In [99]:
```
la.score(x_test,y_test)
```

Out[99]:
```
0.3961252146108115
```

In [100]:
```
la.score(x_train,y_train)
```

Out[100]:
```
0.32734929367923815
```

ElasticNet

In [101]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[101]:
```
ElasticNet()
```

In [102]:
```
print(en.coef_)
```

```
[-13.0267167    2.90141913   4.76941939 -15.29040885]
```

In [103]:
```
print(en.intercept_)
```

```
-4437.297832567254
```

In [104]:
```
prediction=en.predict(x_test)
```

In [105]:
```
print(en.score(x_test,y_test))
```

```
0.40711535845420666
```

In [136]:
```
dd.fillna
```

Out[136]:

ıir_temperature	relative_humidity	barometric_pressure	precipitation	wind_speed	wind_from_direction	gti_clean	gti_soil	gti_monthly	s
24.31900	88.270	1006.328	0.0	0.341667	7.285217	0.0000	0.0000	0.0000	
24.37833	88.476	1006.323	0.0	0.000000	340.589800	0.0000	0.0000	0.0000	
24.37933	88.144	1006.300	0.0	0.000000	13.601040	0.0000	0.0000	0.0000	
24.38700	87.848	1006.285	0.0	0.047500	13.410120	0.0000	0.0000	0.0000	
24.39200	87.812	1006.266	0.0	0.095000	3.582747	0.0000	0.0000	0.0000	
...	...	...	...	...	...	...	...	...	
30.89783	60.035	1003.998	0.0	1.346667	37.132930	409.8599	415.8232	414.5171	
30.74333	61.347	1004.006	0.0	1.537500	44.638590	405.7869	411.8906	410.5445	
30.51017	60.611	1004.005	0.0	2.475000	72.471370	396.3362	402.3746	401.0027	
30.16133	61.802	1004.011	0.0	2.875000	49.169500	378.9769	385.1050	383.7575	
29.97217	62.386	1004.021	0.0	1.800000	39.462180	360.5901	366.6233	365.2733	

Logistic Regression

In [215]:
```
ssd=dd[['air_temperature',
       'relative_humidity', 'barometric_pressure',
       'wind_speed','wind_from_direction']]
ssd
```

Out[215]:

	air_temperature	relative_humidity	barometric_pressure	wind_speed	wind_from_direction
0	24.31900	88.270	1006.328	0.341667	7.285217
1	24.37833	88.476	1006.323	0.000000	340.589800
2	24.37933	88.144	1006.300	0.000000	13.601040
3	24.38700	87.848	1006.285	0.047500	13.410120
4	24.39200	87.812	1006.266	0.095000	3.582747
...	...	...	...	...	...
995	30.89783	60.035	1003.998	1.346667	37.132930
996	30.74333	61.347	1004.006	1.537500	44.638590
997	30.51017	60.611	1004.005	2.475000	72.471370
998	30.16133	61.802	1004.011	2.875000	49.169500
999	29.97217	62.386	1004.021	1.800000	39.462180

1000 rows × 5 columns

In [224]:
```
from sklearn.linear_model import LogisticRegression
```

In [238]:
```
x=ssd[['air_temperature',
       'relative_humidity', 'barometric_pressure',
       'wind_speed']]
y=ssd['wind_from_direction']
```

In [239]:
```
x
```

Out[239]:

	air_temperature	relative_humidity	barometric_pressure	wind_speed
0	24.31900	88.270	1006.328	0.341667
1	24.37833	88.476	1006.323	0.000000
2	24.37933	88.144	1006.300	0.000000
3	24.38700	87.848	1006.285	0.047500
4	24.39200	87.812	1006.266	0.095000
...	...	...	...	...
995	30.89783	60.035	1003.998	1.346667
996	30.74333	61.347	1004.006	1.537500
997	30.51017	60.611	1004.005	2.475000
998	30.16133	61.802	1004.011	2.875000
999	29.97217	62.386	1004.021	1.800000

1000 rows × 4 columns

In [240]:
```
y
```

Out[240]:
```
0        7.285217
1      340.589800
2       13.601040
3       13.410120
4        3.582747
           ...
995     37.132930
996     44.638590
997     72.471370
998     49.169500
999     39.462180
Name: wind_from_direction, Length: 1000, dtype: float64
```

In [241]:
```
# feature_matrix = dd[['barometric_pressure']]
# target_vector=dd['wind_from_direction']

feature_matrix = x
target_vector=y
```

In [242]:
```
feature_matrix.shape
```

Out[242]:
```
(1000, 4)
```

In [243]:
```
target_vector.shape
```

Out[243]:
```
(1000,)
```

In [244]:
```
from sklearn.preprocessing import StandardScaler
```

In [245]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

Evaluation metrics

In [145]:
```
from sklearn import metrics
```

In [146]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 134.49723050596643
```

In [124]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 12117.16659682255
```

In [125]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root mean Absolytre Error: 110.07800232935983
```

RandomForestClassifier()

In [247]:
```
ged=ssd[['air_temperature','relative_humidity', 'barometric_pressure','wind_speed','wind_from_direction']]
```

In [248]:
```
d=ged.fillna(900)
```

In [249]:
```
dg=d.head(1000)
```

In [250]:
```
x=dg[['air_temperature','relative_humidity', 'barometric_pressure','wind_speed']]
y=dg['wind_from_direction']
```

In [251]:
```
print(len(x))
print(len(y))
```

```
1000
1000
```