In [1]:
```
import pandas as pd, numpy as np
import matplotlib.pyplot as plt, seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [2]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving autumn.csv to autumn (3).csv
```

In [3]:
```
df = pd.read_csv(r'autumn.csv')
print(df)
#sns.countplot(df['Power output'])
#plt.title('Power output')
#plt.show()
#date; last two denotes month, rest first digits date
#time is in format hhmm
```

```
       Date  Time  humidity  windspeed  Temperature  solar Irradiance  \
0       103   700      66.1       -0.4         22.0              37.2
1       103   701      66.1       -0.4         22.0              37.2
2       103   702      66.1       -0.4         22.0              37.2
3       103   703      66.2       -0.4         22.0              36.8
4       103   704      66.2       -0.4         22.0              36.8
...     ...   ...       ...        ...          ...               ...
48676  3105  1656      56.1        1.0         19.2               5.7
48677  3105  1657      56.1        1.0         19.2               5.7
48678  3105  1658      56.1        1.0         19.2               5.7
48679  3105  1659      56.1        1.0         19.2               2.8
48680  3105  1700      56.1        1.0         19.2               2.8

       Power output
0             14.15
1             14.00
2             13.80
3             13.75
4             13.55
...             ...
48676          0.29
48677          0.00
48678          0.00
48679          0.00
48680          0.00

[48681 rows x 7 columns]
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48681 entries, 0 to 48680
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Date              48681 non-null  int64  
 1   Time              48681 non-null  int64  
 2   humidity          48681 non-null  float64
 3   windspeed         48681 non-null  float64
 4   Temperature       48681 non-null  float64
 5   solar Irradiance  48681 non-null  float64
 6   Power output      48681 non-null  float64
dtypes: float64(5), int64(2)
memory usage: 2.6 MB
```

In [5]:
```
attributes = df.keys()
attributes[6]
#class attribute is power output
```

Out[5]:
```
'Power output'
```

In [6]:
```
df.duplicated().sum()
#duplicated data
```

Out[6]:
```
0
```

In [7]:
```
MS_df = df.isna().sum()
MS_Values = MS_df.values
MS_Keys = MS_df.keys()
for i in range(len(MS_Values)):
    V = MS_Values[i]
    if V>0:
        Keys_Missing = MS_Keys[i]
        print(f'      {Keys_Missing}                      {V}')
```

In [8]:
```
df["humidity"] = df["humidity"].astype(int)
df["windspeed"] = df["windspeed"].astype(int)
df["Temperature"] = df["Temperature"].astype(int)
df["Power output"] = df["Power output"].astype(int)
df["solar Irradiance"] = df["solar Irradiance"].astype(int)
```

In [9]:
```
df_f =df.copy()
```

In [10]:
```
df.isnull()
#checking null valuess
```

Out[10]:

	Date	Time	humidity	windspeed	Temperature	solar Irradiance	Power output
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
48676	False	False	False	False	False	False	False
48677	False	False	False	False	False	False	False
48678	False	False	False	False	False	False	False
48679	False	False	False	False	False	False	False
48680	False	False	False	False	False	False	False

48681 rows × 7 columns

In [11]:
```
df.isnull().sum()
#checkig total null values
```

Out[11]:
```
Date                0
Time                0
humidity            0
windspeed           0
Temperature         0
solar Irradiance    0
Power output        0
dtype: int64
```

In [12]:
```
df.duplicated()
#checking duplicted values
```

Out[12]:
```
0        False
1        False
2        False
3        False
4        False
         ...
48676    False
48677    False
48678    False
48679    False
48680    False
Length: 48681, dtype: bool
```

In [13]:
```
df.drop_duplicates()
#dropping duplicated values
```

Out[13]:

	Date	Time	humidity	windspeed	Temperature	solar Irradiance	Power output
0	103	700	66	0	22	37	14
1	103	701	66	0	22	37	14
2	103	702	66	0	22	37	13
3	103	703	66	0	22	36	13
4	103	704	66	0	22	36	13
...	...	...	...	...	...	...	...
48676	3105	1656	56	1	19	5	0
48677	3105	1657	56	1	19	5	0
48678	3105	1658	56	1	19	5	0
48679	3105	1659	56	1	19	2	0
48680	3105	1700	56	1	19	2	0

48681 rows × 7 columns

In [14]:
```
df.describe()
#found humidity negative, inconsistent data, remove the inconsistency later
```

Out[14]:

	Date	Time	humidity	windspeed	Temperature	solar Irradiance	Power
count	48681.000000	48681.000000	48681.000000	48681.000000	48681.000000	48681.000000	48681
mean	1648.419753	1180.366057	55.734393	0.781455	21.008258	362.897989	144
std	921.784712	288.294790	13.037593	1.061617	3.946256	266.362149	100
min	103.000000	700.000000	-25.000000	-13.000000	0.000000	0.000000	0
25%	705.000000	930.000000	47.000000	0.000000	18.000000	140.000000	58
50%	1803.000000	1200.000000	54.000000	0.000000	21.000000	297.000000	123
75%	2405.000000	1430.000000	65.000000	1.000000	24.000000	563.000000	229
max	3105.000000	1700.000000	83.000000	9.000000	33.000000	1284.000000	430

In [15]:
```
df.humidity.unique()
```

Out[15]:
```
array([ 66,  65,  64,  63,  62,  67,  59,  60,  61,  68,  69,  70,  71,
        72,  73,  74,  75,  76,  77,  78,  80,  79,  81,  57,  56,  55,
        54,  52,  53,  51,  50,  49,  48,  58,  47,  46,  45,  44,  43,
        42,  41,  39,  40,  38,  37,  36,  35, -25,  32,  34,  30,  31,
        29,  28,  26,  27,  25,  24,  23,  22,  21,  82,  33,  83,   0])
```

In [16]:
```
df.windspeed.unique()
```

Out[16]:
```
array([  0,   1,   2,   3,   5,   4,   6, -13,   7,   8,   9])
```

In [17]:
```
df.Temperature.unique()
```

Out[17]:
```
array([22, 23, 24, 25, 21, 19, 20, 26, 27, 18, 28, 29, 30, 31, 32, 33, 17,
       16, 15, 14, 12, 13,  0, 11, 10,  9,  8])
```

In [18]:
```
df.drop(df[df['humidity'] < 21].index,inplace = True)
df.drop(df[df['Temperature'] < 10].index,inplace=True)
df.drop(df[df['windspeed'] < 1].index,inplace=True)
```

In [19]:
```
df.drop_duplicates()
```

Out[19]:

	Date	Time	humidity	windspeed	Temperature	solar Irradiance	Power output
310	103	1210	61	1	24	416	170
311	103	1211	61	1	24	416	167
312	103	1212	61	1	24	416	167
313	103	1213	61	1	24	416	167
707	203	846	81	2	20	498	314
...	...	...	...	...	...	...	...
48676	3105	1656	56	1	19	5	0
48677	3105	1657	56	1	19	5	0
48678	3105	1658	56	1	19	5	0
48679	3105	1659	56	1	19	2	0
48680	3105	1700	56	1	19	2	0

23463 rows × 7 columns

In [20]:
```
df.Temperature.unique()
```

Out[20]:
```
array([24, 20, 19, 21, 22, 25, 26, 23, 27, 18, 28, 29, 31, 30, 17, 16, 15,
       14, 13, 12, 11, 10])
```

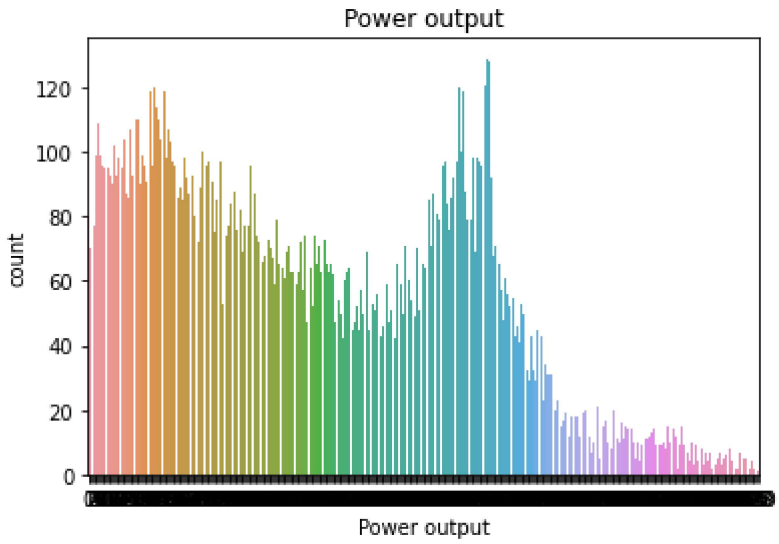
In [21]:
```
df1=df.drop('Date',axis=1)
df2=df1.drop('Time',axis=1)
```

In [22]:
```
df2.info()
```

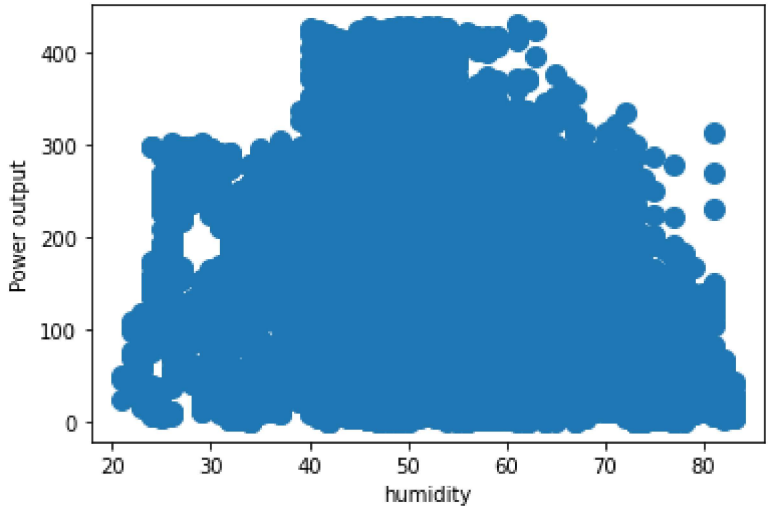
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23463 entries, 310 to 48680
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   humidity          23463 non-null  int64
 1   windspeed         23463 non-null  int64
 2   Temperature       23463 non-null  int64
 3   solar Irradiance  23463 non-null  int64
 4   Power output      23463 non-null  int64
dtypes: int64(5)
memory usage: 1.1 MB
```

In [23]:
```
sns.countplot(df2['Power output'])
plt.title('Power output')
plt.show()
```

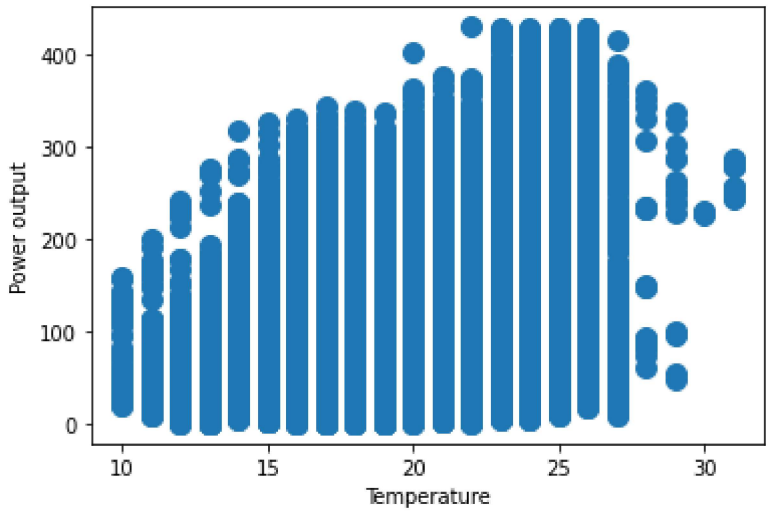
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



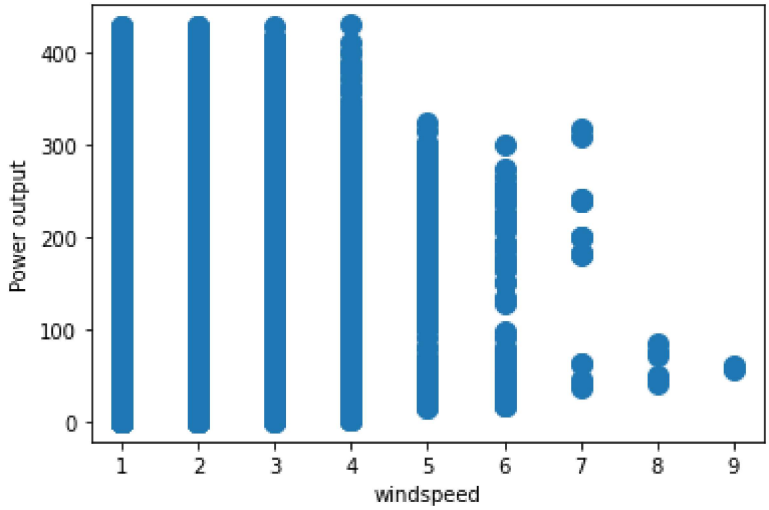
In [24]:
```
df2.plot.scatter(x = 'humidity', y = 'Power output', s = 100);
```



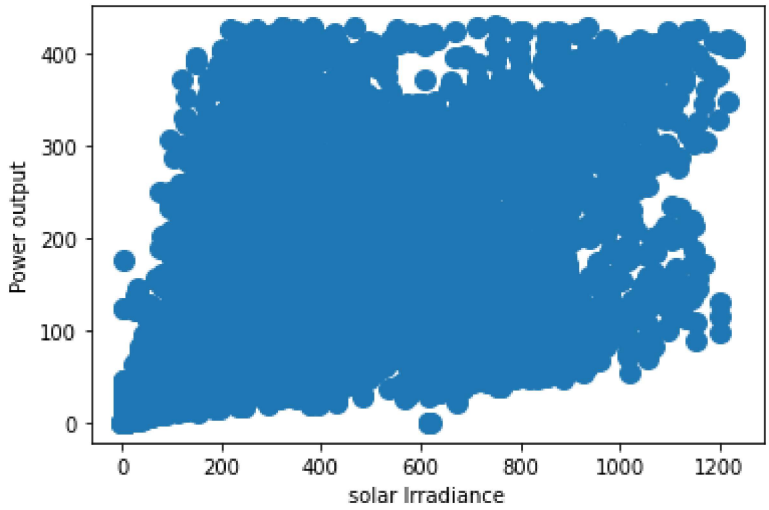
In [25]:
```
df2.plot.scatter(x = 'Temperature', y = 'Power output', s = 100);
```



In [26]:
```
df2.plot.scatter(x = 'windspeed', y = 'Power output', s = 100);
```



In [27]:
```
df2.plot.scatter(x = 'solar Irradiance', y = 'Power output', s = 100);
```



In [28]:
```
X = df2.drop('Power output',axis=1)
# Putting response variable to y
y = df2['Power output']
```

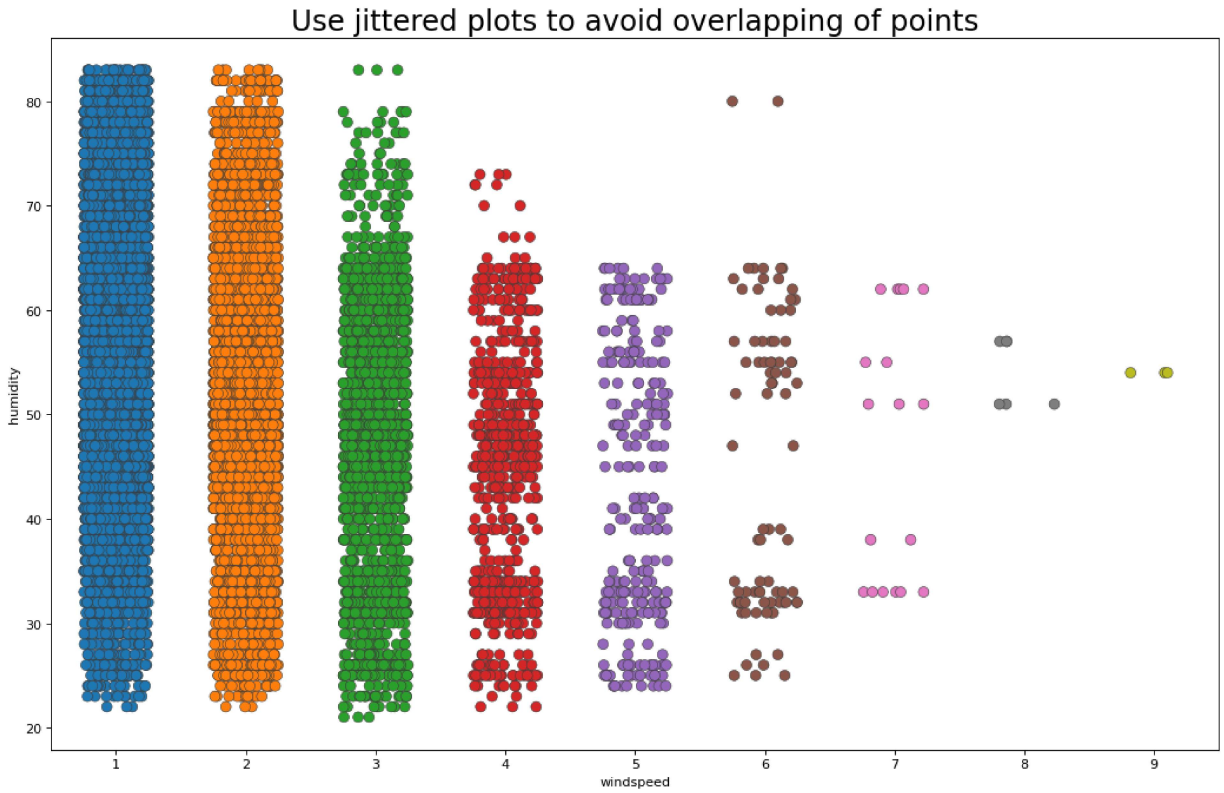
In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.99, randor
X_train.shape, X_test.shape
```

Out[29]:
```
((23228, 4), (235, 4))
```

In [30]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

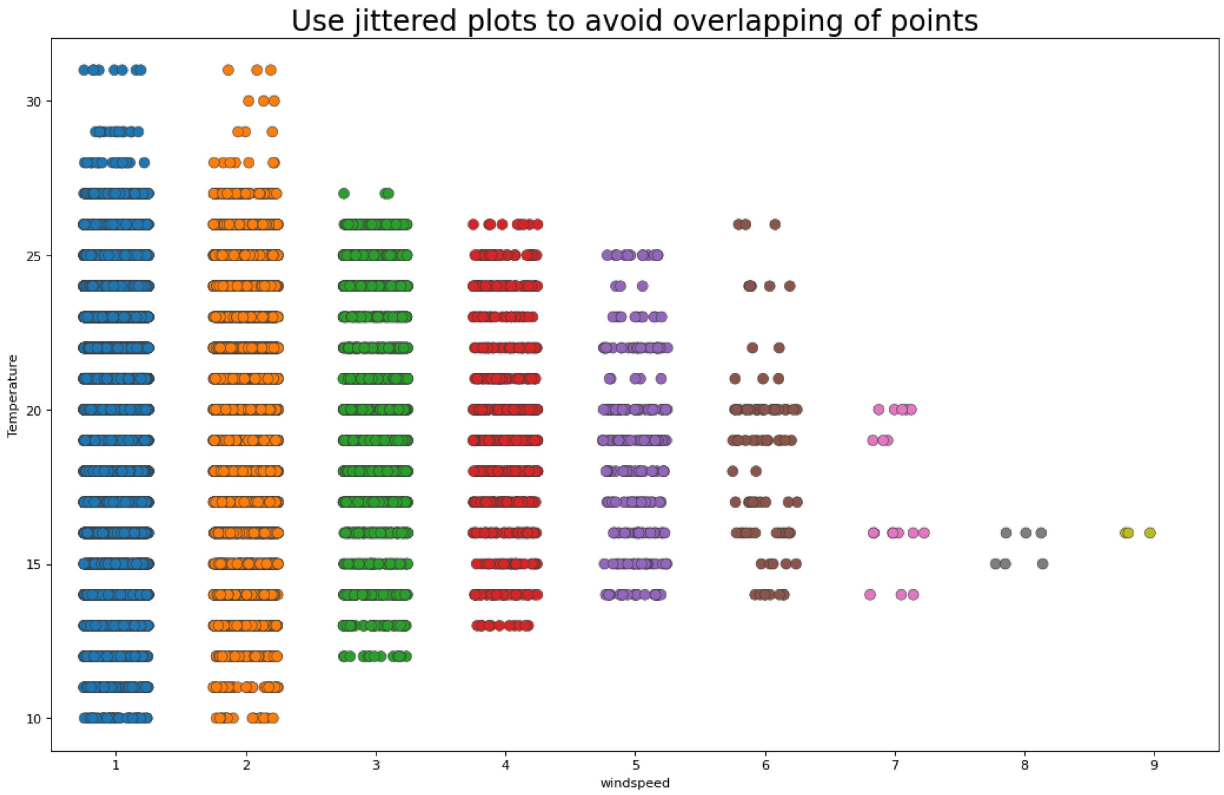
In [31]:
```
fig, ax = plt.subplots(figsize=(16,10), dpi= 80)
sns.stripplot(df2.windspeed, df2.humidity, jitter=0.25, size=8, ax=ax, linewidth=

# Decorations
plt.title('Use jittered plots to avoid overlapping of points', fontsize=22)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



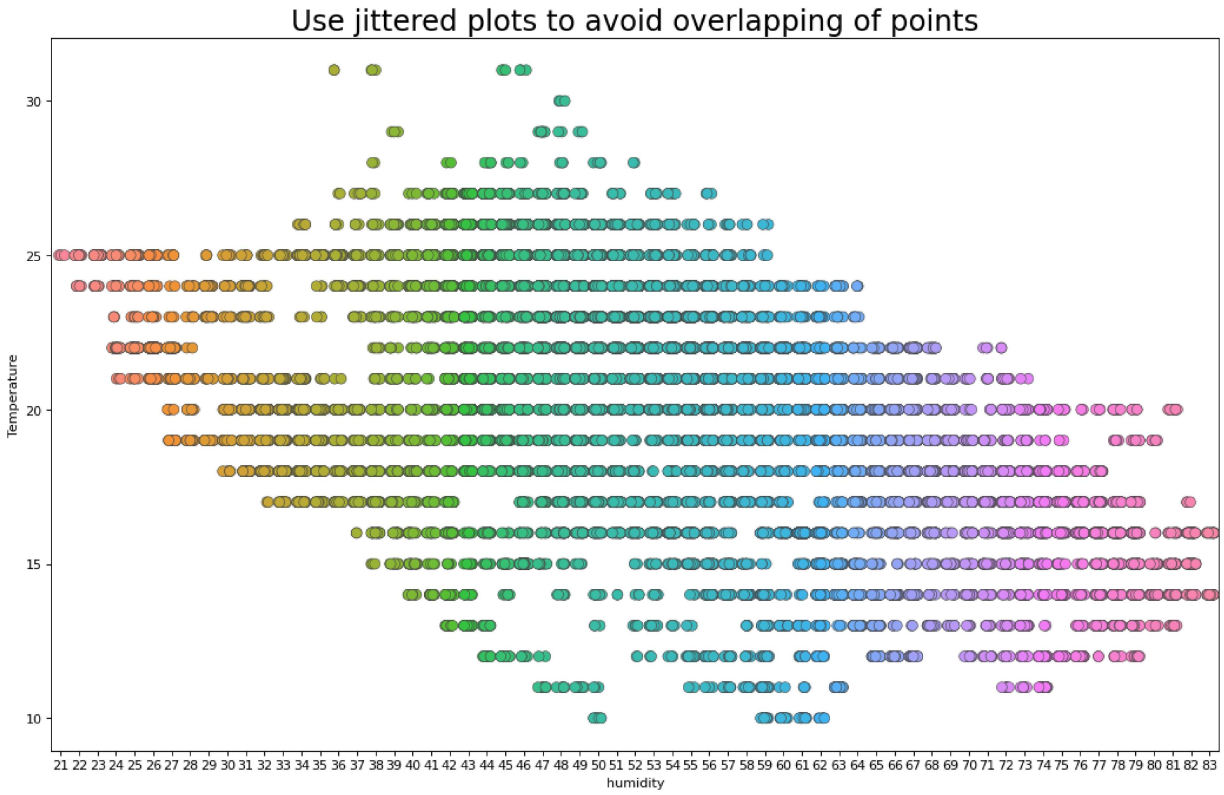
In [32]:
```
fig, ax = plt.subplots(figsize=(16,10), dpi= 80)
sns.stripplot(df2.windspeed, df2.Temperature, jitter=0.25, size=8, ax=ax, linewic

# Decorations
plt.title('Use jittered plots to avoid overlapping of points', fontsize=22)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



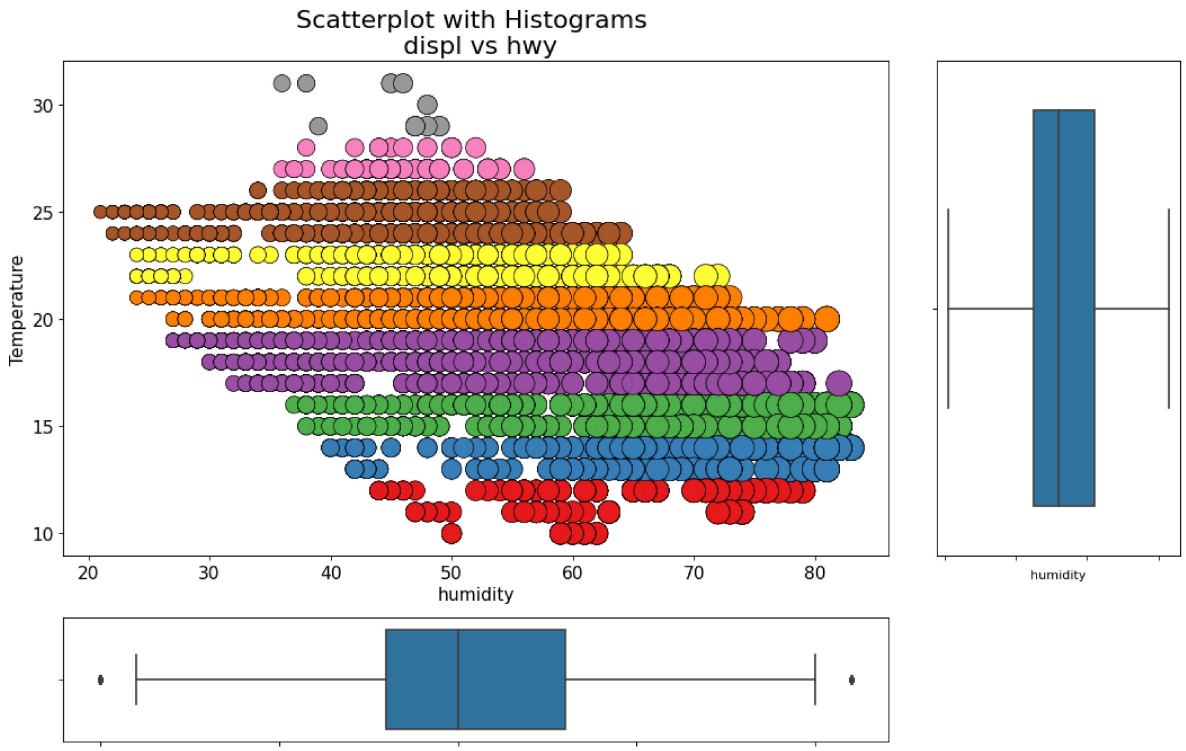
In [33]:
```
fig, ax = plt.subplots(figsize=(16,10), dpi= 80)
sns.stripplot(df2.humidity, df2.Temperature, jitter=0.25, size=8, ax=ax, linewidt

# Decorations
plt.title('Use jittered plots to avoid overlapping of points', fontsize=22)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



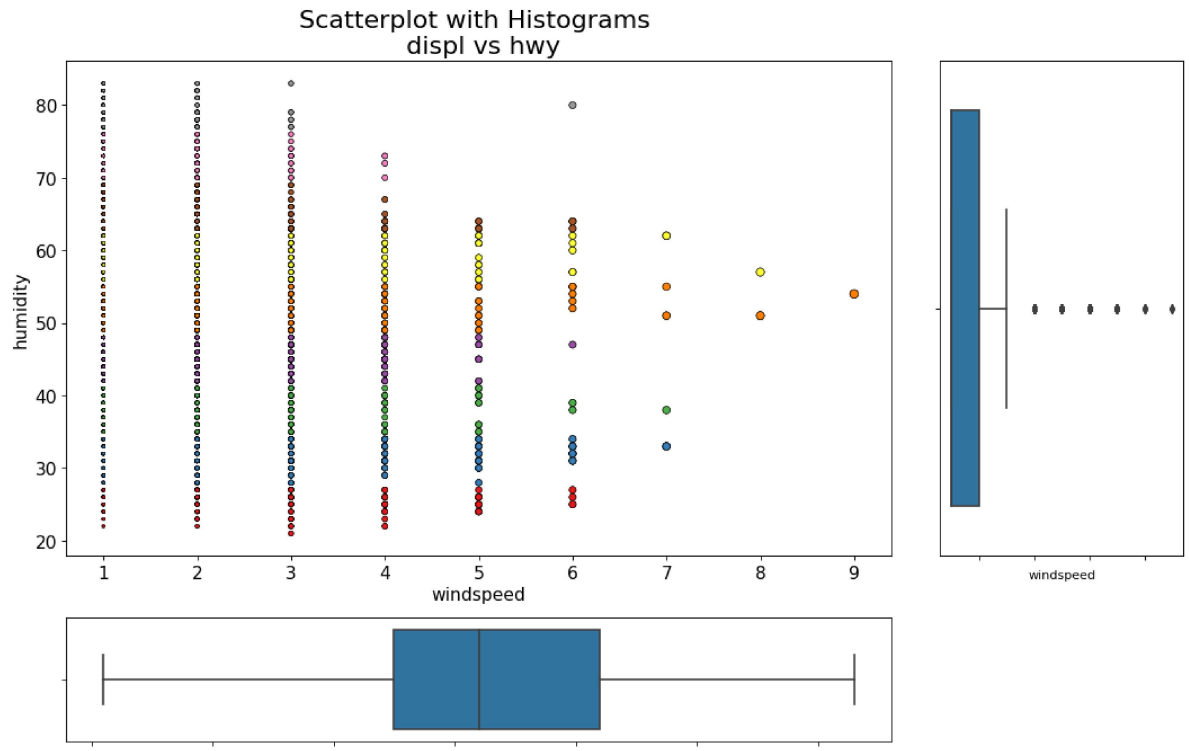
In [34]:
```
fig = plt.figure(figsize=(16, 10), dpi= 80)
grid = plt.GridSpec(4, 4, hspace=0.5, wspace=0.2)

# Define the axes
ax_main = fig.add_subplot(grid[:-1, :-1])
ax_right = fig.add_subplot(grid[:-1, -1], xticklabels=[], yticklabels=[])
ax_bottom = fig.add_subplot(grid[-1, 0:-1], xticklabels=[], yticklabels=[])

# Scatterplot on main ax
ax_main.scatter('humidity', 'Temperature', s=df2.humidity*5, c=df2.Temperature.as

# Add a graph in each part
sns.boxplot(df2.humidity, ax=ax_right, orient="v")
sns.boxplot(df2.Temperature, ax=ax_bottom, orient="h")

# Decorations ------------------
# Remove x axis name for the boxplot
ax_bottom.set(xlabel='')
ax_right.set(ylabel='')

# Main Title, Xlabel and YLabel
ax_main.set(title='Scatterplot with Histograms \n displ vs hwy', xlabel='humidity

# Set font size of different components
ax_main.title.set_fontsize(20)
for item in ([ax_main.xaxis.label, ax_main.yaxis.label] + ax_main.get_xticklabels
    item.set_fontsize(14)

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_core.py:1326: UserWarning: Vert
ical orientation ignored with only `x` specified.
  warnings.warn(single_var_warning.format("Vertical", "x"))
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```


Scatterplot with Histograms
displ vs hwy
Temperature
humidity
30
25
20
15
10
20
30
40
50
60
70
80
humidity

In [35]:
```
fig = plt.figure(figsize=(16, 10), dpi= 80)
grid = plt.GridSpec(4, 4, hspace=0.5, wspace=0.2)

# Define the axes
ax_main = fig.add_subplot(grid[:-1, :-1])
ax_right = fig.add_subplot(grid[:-1, -1], xticklabels=[], yticklabels=[])
ax_bottom = fig.add_subplot(grid[-1, 0:-1], xticklabels=[], yticklabels=[])

# Scatterplot on main ax
ax_main.scatter('windspeed', 'humidity', s=df2.windspeed*5, c=df2.humidity.astype

# Add a graph in each part
sns.boxplot(df2.windspeed, ax=ax_right, orient="v")
sns.boxplot(df2.humidity, ax=ax_bottom, orient="h")

# Decorations ------------------
# Remove x axis name for the boxplot
ax_bottom.set(xlabel='')
ax_right.set(ylabel='')

# Main Title, Xlabel and YLabel
ax_main.set(title='Scatterplot with Histograms \n displ vs hwy', xlabel='windspee

# Set font size of different components
ax_main.title.set_fontsize(20)
for item in ([ax_main.xaxis.label, ax_main.yaxis.label] + ax_main.get_xticklabels
    item.set_fontsize(14)

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_core.py:1326: UserWarning: Vert
ical orientation ignored with only `x` specified.
  warnings.warn(single_var_warning.format("Vertical", "x"))
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



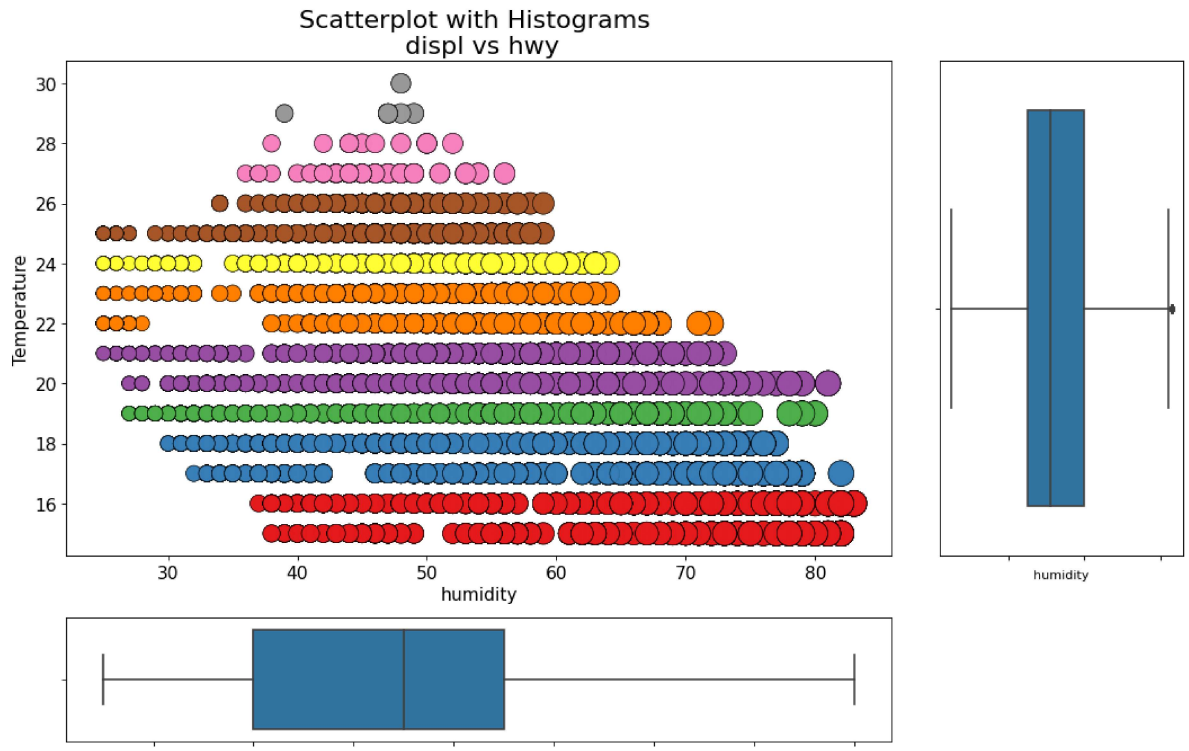
In [36]:
```
df2.drop(df2[df2['Temperature'] < 15].index,inplace = True)
```

In [37]:
```
df2.drop(df2[df2['Temperature'] > 30].index,inplace = True)
```

In [38]:
```
df2.drop(df2[df2['humidity'] < 25].index,inplace = True)
```

In [39]:
```
fig = plt.figure(figsize=(16, 10), dpi= 80)
grid = plt.GridSpec(4, 4, hspace=0.5, wspace=0.2)

# Define the axes
ax_main = fig.add_subplot(grid[:-1, :-1])
ax_right = fig.add_subplot(grid[:-1, -1], xticklabels=[], yticklabels=[])
ax_bottom = fig.add_subplot(grid[-1, 0:-1], xticklabels=[], yticklabels=[])

# Scatterplot on main ax
ax_main.scatter('humidity', 'Temperature', s=df2.humidity*5, c=df2.Temperature.as

# Add a graph in each part
sns.boxplot(df2.humidity, ax=ax_right, orient="v")
sns.boxplot(df2.Temperature, ax=ax_bottom, orient="h")

# Decorations ------------------
# Remove x axis name for the boxplot
ax_bottom.set(xlabel='')
ax_right.set(ylabel='')

# Main Title, Xlabel and YLabel
ax_main.set(title='Scatterplot with Histograms \n displ vs hwy', xlabel='humidity

# Set font size of different components
ax_main.title.set_fontsize(20)
for item in ([ax_main.xaxis.label, ax_main.yaxis.label] + ax_main.get_xticklabels
    item.set_fontsize(14)

plt.show()
```

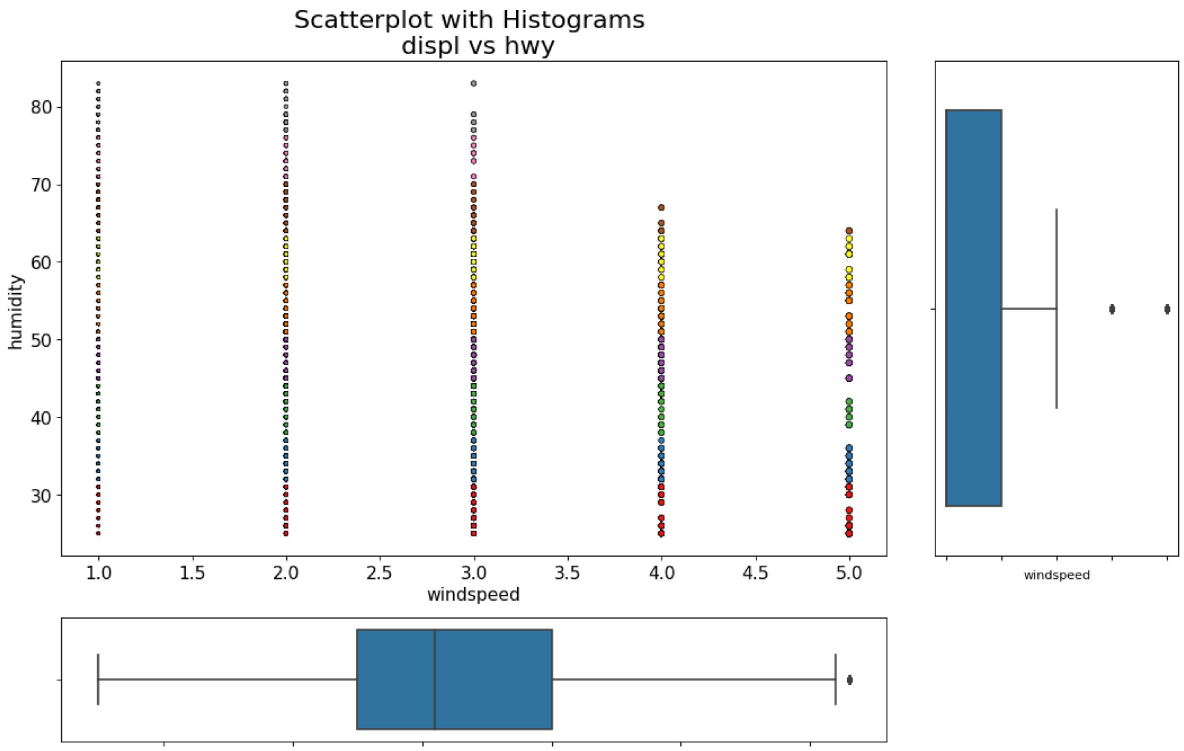
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_core.py:1326: UserWarning: Vert
ical orientation ignored with only `x` specified.
  warnings.warn(single_var_warning.format("Vertical", "x"))
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



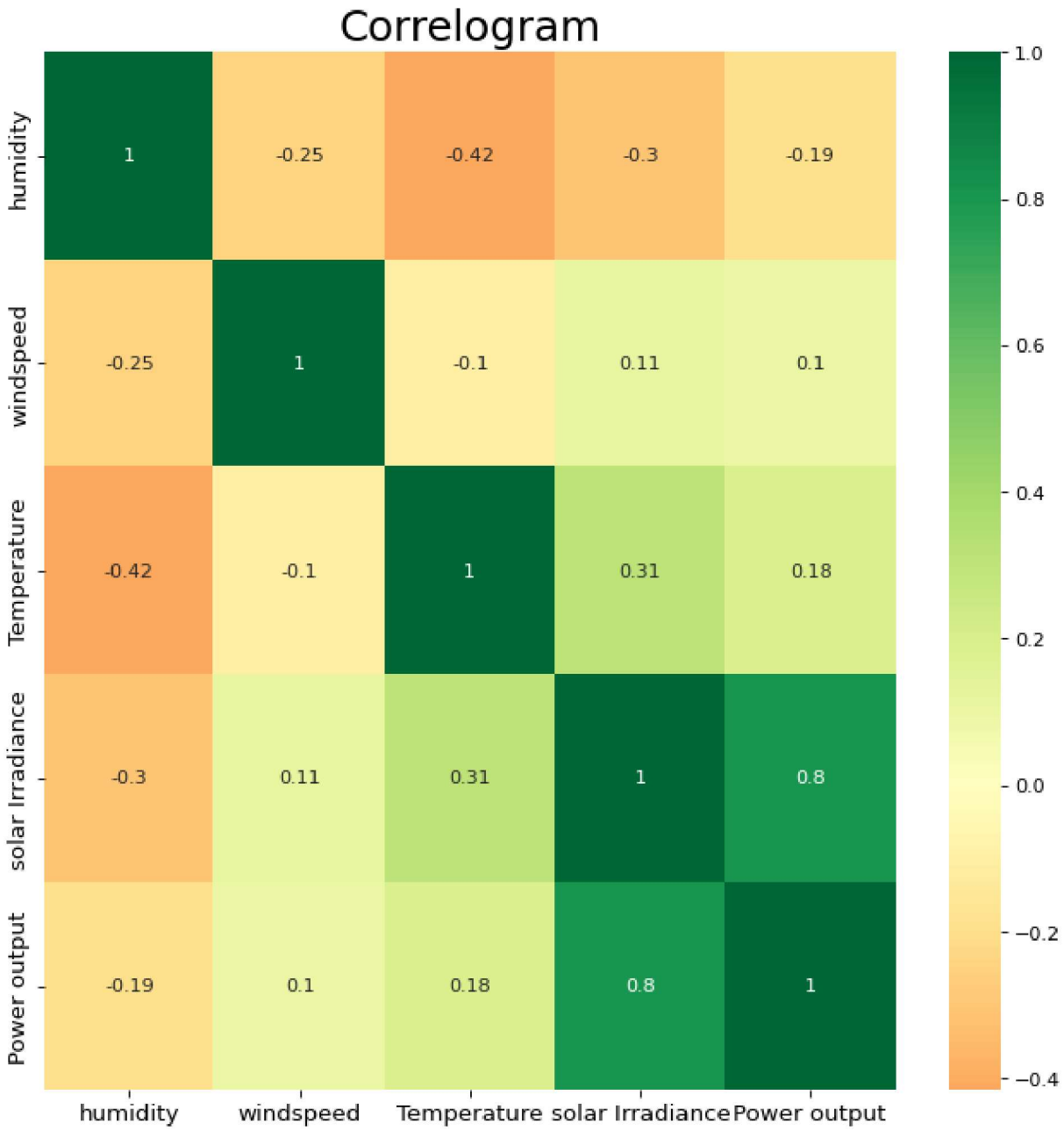
In [40]:
```
df2.drop(df2[df2['windspeed'] > 5].index,inplace = True)
```

In [41]:
```
fig = plt.figure(figsize=(16, 10), dpi= 80)
grid = plt.GridSpec(4, 4, hspace=0.5, wspace=0.2)

# Define the axes
ax_main = fig.add_subplot(grid[:-1, :-1])
ax_right = fig.add_subplot(grid[:-1, -1], xticklabels=[], yticklabels=[])
ax_bottom = fig.add_subplot(grid[-1, 0:-1], xticklabels=[], yticklabels=[])

# Scatterplot on main ax
ax_main.scatter('windspeed', 'humidity', s=df2.windspeed*5, c=df2.humidity.astype

# Add a graph in each part
sns.boxplot(df2.windspeed, ax=ax_right, orient="v")
sns.boxplot(df2.humidity, ax=ax_bottom, orient="h")

# Decorations ------------------
# Remove x axis name for the boxplot
ax_bottom.set(xlabel='')
ax_right.set(ylabel='')

# Main Title, Xlabel and YLabel
ax_main.set(title='Scatterplot with Histograms \n displ vs hwy', xlabel='windspee

# Set font size of different components
ax_main.title.set_fontsize(20)
for item in ([ax_main.xaxis.label, ax_main.yaxis.label] + ax_main.get_xticklabels
    item.set_fontsize(14)

plt.show()
```

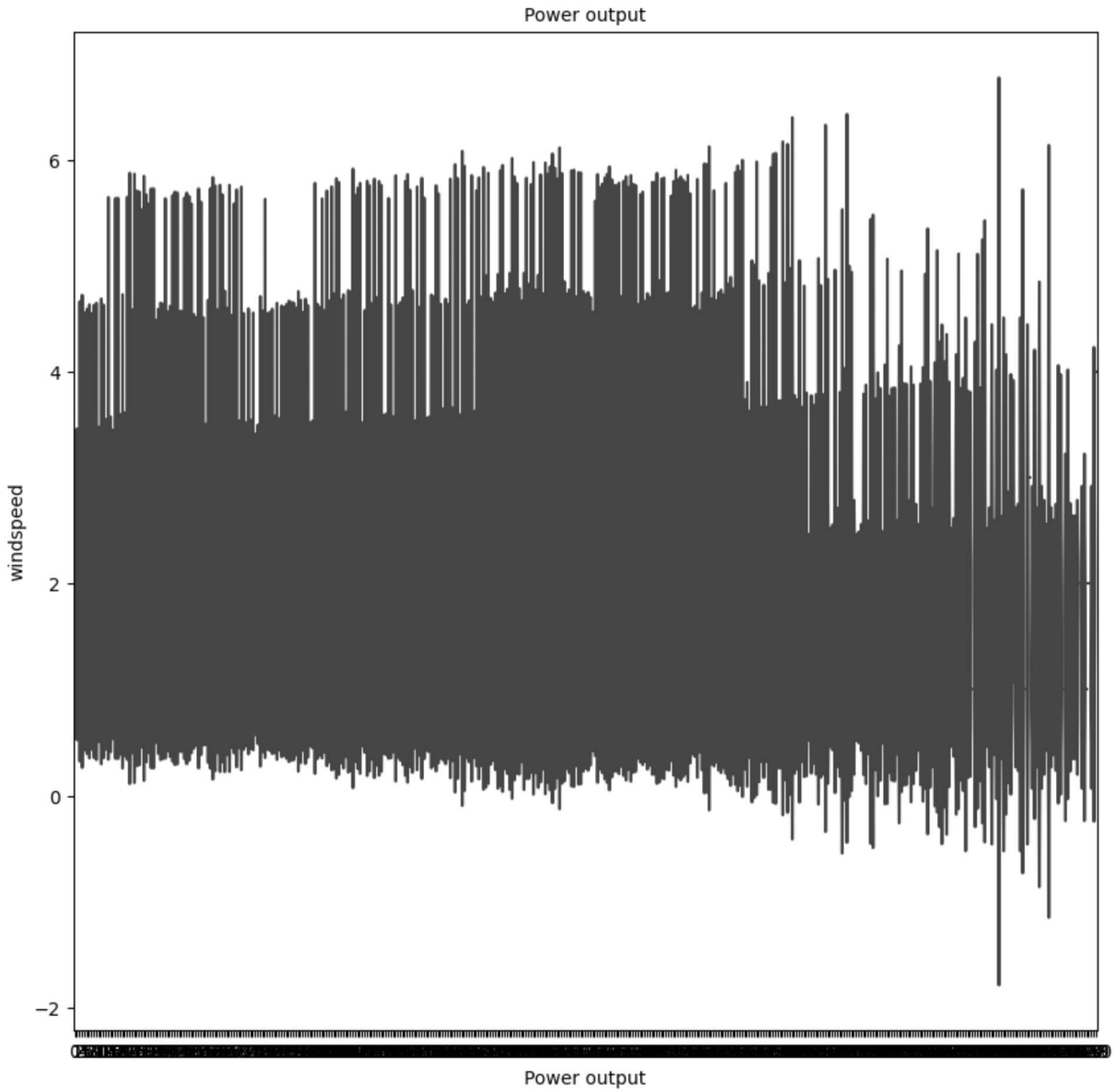
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_core.py:1326: UserWarning: Vert
ical orientation ignored with only `x` specified.
  warnings.warn(single_var_warning.format("Vertical", "x"))
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```


Scatterplot with Histograms
displ vs hwy
humidity
windspeed
windspeed

In [42]:
```
plt.figure(figsize=(10,10), dpi= 80)
sns.heatmap(df2.corr(), xticklabels=df2.corr().columns, yticklabels=df2.corr().co

# Decorations
plt.title('Correlogram', fontsize=22)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



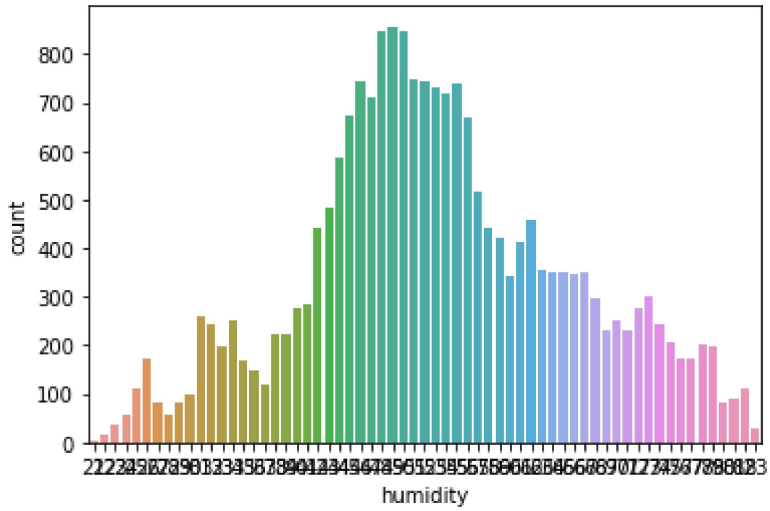
In [43]:
```
plt.figure(figsize=(10,10), dpi=100)
sns.violinplot(x='Power output', y='windspeed', data=df2, scale='width', inner='c

# Decoration
plt.title('Power output', fontsize=10)
plt.show()
```

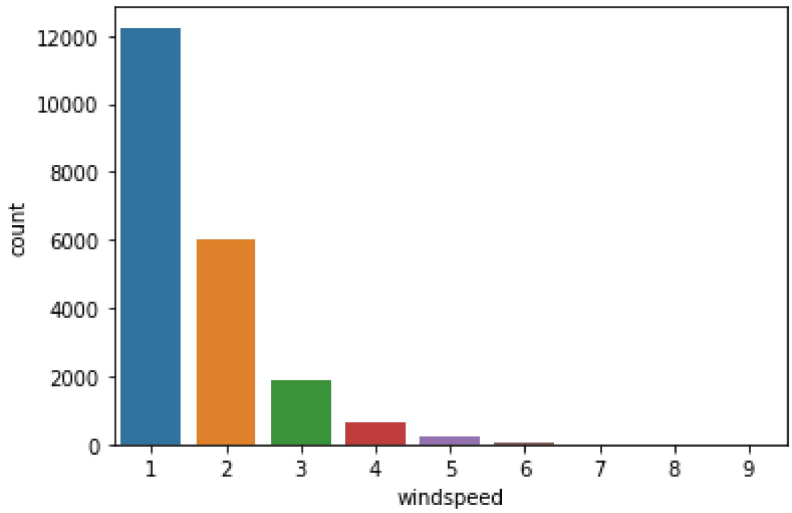


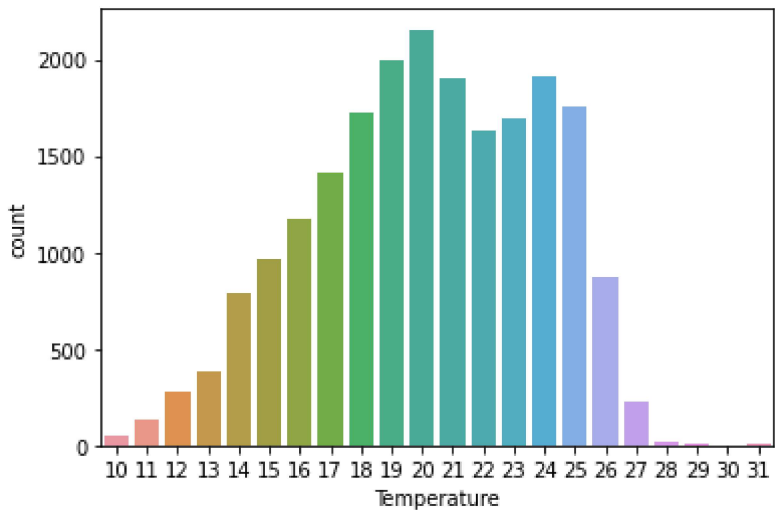
In [52]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.9, random_
X_train.shape, X_test.shape
```

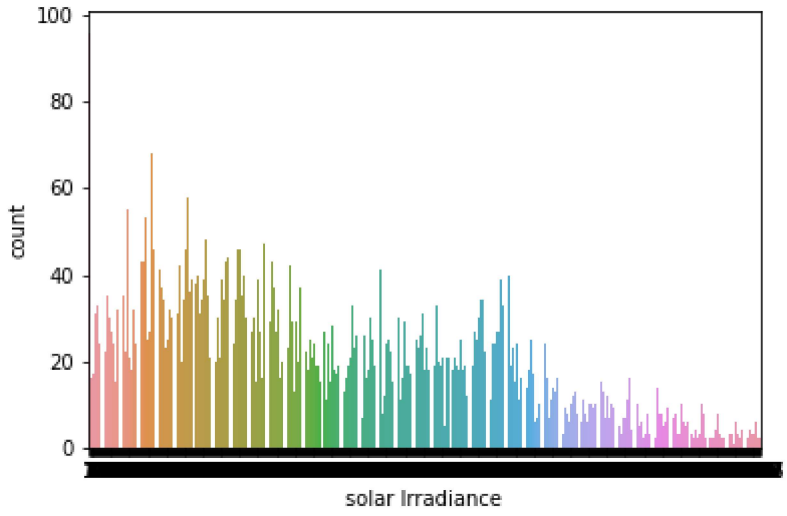
Out[52]:
```
((21116, 4), (2347, 4))
```

In [53]:
```
for i, predictor in enumerate(X_train):
    plt.figure(i)
    sns.countplot(data=X_train, x=predictor)
```









In [54]:
```
X = df2.drop('Power output',axis=1)
# Putting response variable to y
y = df2['Power output']
```

In [55]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.4, random_
X_train.shape, X_test.shape
```

Out[55]:
```
((8558, 4), (12838, 4))
```

In [56]:
```
classifier_rf = RandomForestClassifier(random_state=100, n_jobs=-1, max_depth=5,r
```

In [57]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21396 entries, 310 to 48680
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   humidity          21396 non-null  int64
 1   windspeed         21396 non-null  int64
 2   Temperature       21396 non-null  int64
 3   solar Irradiance  21396 non-null  int64
 4   Power output      21396 non-null  int64
dtypes: int64(5)
memory usage: 1.5 MB
```

In [58]:
```
%%time
classifier_rf.fit(X_train, y_train)
```

```
CPU times: user 2.24 s, sys: 96.6 ms, total: 2.34 s
Wall time: 1.83 s
```

Out[58]:
```
RandomForestClassifier(max_depth=5, n_jobs=-1, oob_score=True, random_state=10
0)
```

In [59]:
```
classifier_rf.oob_score_
```

Out[59]:
```
0.037275064267352186
```

In [60]:
```
rf = RandomForestClassifier(random_state=100, n_jobs=-1)
```

In [61]:
```
params = {
    'max_depth': [2,3,5,10,20],
    'min_samples_leaf': [5,10,20,50,100,200],
    'n_estimators': [10,25,30,50,100,200]
}
```

In [62]:
```
grid_search = GridSearchCV(estimator=rf,param_grid=params,cv = 4,n_jobs=-1, verbo
```

In [63]:
```
grid_search.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 180 candidates, totalling 720 fits

/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: U
serWarning: The least populated class in y has only 1 members, which is less th
an n_splits=4.
  UserWarning,
/usr/local/lib/python3.7/dist-packages/joblib/externals/loky/process_executor.p
y:705: UserWarning: A worker stopped while some jobs were given to the executo
r. This can be caused by a too short worker timeout or by a memory leak.
  "timeout or by a memory leak.", UserWarning
```

Out[63]:
```
GridSearchCV(cv=4,
             estimator=RandomForestClassifier(n_jobs=-1, random_state=100),
             n_jobs=-1,
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy', verbose=1)
```

In [64]:
```
grid_search.best_score_
```

Out[64]:
```
0.0781739763973907
```

In [65]:
```
rf_best = grid_search.best_estimator_
rf_best
```

Out[65]:
```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_jobs=-1,
                       random_state=100)
```

In [66]:
```
rf_best.feature_importances_
```

Out[66]:
```
array([0.27593903, 0.08417366, 0.15317147, 0.48671583])
```

In [67]:
```
imp_df = pd.DataFrame({
    "Varname": X_train.columns,
    "Imp": rf_best.feature_importances_
})
```

In [68]:
```
imp_df.sort_values(by="Imp", ascending=False)
```

Out[68]:

	Varname	Imp
3	solar Irradiance	0.486716
0	humidity	0.275939
2	Temperature	0.153171
1	windspeed	0.084174

In [69]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.95, randor
X_train.shape, X_test.shape
```

Out[69]:
```
((20326, 4), (1070, 4))
```

In [70]:
```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=2000, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

In [71]:
```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pr
```

```
Mean Absolute Error: 17.17955615264725
Mean Squared Error: 1363.913218498463
Root Mean Squared Error: 36.9311957361045
```

In [ ]: