In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
df = pd.read_csv(r'C:\Users\sutharsan\Downloads\iNeubytes airpollution T2\PRSA_data_2010.1.1-2014.12.31.csv')
```

In [3]:
```python
df.head(5)
```

Out[3]:

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
0	1	2010	1	1	0	NaN	-21	-11.0	1021.0	NW	1.79	0	0
1	2	2010	1	1	1	NaN	-21	-12.0	1020.0	NW	4.92	0	0
2	3	2010	1	1	2	NaN	-21	-11.0	1019.0	NW	6.71	0	0
3	4	2010	1	1	3	NaN	-21	-14.0	1019.0	NW	9.84	0	0
4	5	2010	1	1	4	NaN	-20	-12.0	1018.0	NW	12.97	0	0

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43824 entries, 0 to 43823
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   No      43824 non-null  int64
 1   year    43824 non-null  int64
 2   month   43824 non-null  int64
 3   day     43824 non-null  int64
 4   hour    43824 non-null  int64
 5   pm2.5   41757 non-null  float64
 6   DEWP    43824 non-null  int64
 7   TEMP    43824 non-null  float64
 8   PRES    43824 non-null  float64
 9   cbwd    43824 non-null  object
 10  Iws     43824 non-null  float64
 11  Is      43824 non-null  int64
 12  Ir      43824 non-null  int64
dtypes: float64(4), int64(8), object(1)
memory usage: 4.2+ MB
```

In [5]:
```python
df.shape  # rows and columns
```

Out[5]:
```
(43824, 13)
```

In [6]:
```python
df.describe().transpose()
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
No	43824.0	21912.500000	12651.043435	1.00	10956.75	21912.50	32868.25	43824.0
year	43824.0	2012.000000	1.413842	2010.00	2011.00	2012.00	2013.00	2014.0
month	43824.0	6.523549	3.448572	1.00	4.00	7.00	10.00	12.0
day	43824.0	15.727820	8.799425	1.00	8.00	16.00	23.00	31.0
hour	43824.0	11.500000	6.922266	0.00	5.75	11.50	17.25	23.0
pm2.5	41757.0	98.613215	92.050387	0.00	29.00	72.00	137.00	994.0
DEWP	43824.0	1.817246	14.433440	-40.00	-10.00	2.00	15.00	28.0
TEMP	43824.0	12.448521	12.198613	-19.00	2.00	14.00	23.00	42.0
PRES	43824.0	1016.447654	10.268698	991.00	1008.00	1016.00	1025.00	1046.0
Iws	43824.0	23.889140	50.010635	0.45	1.79	5.37	21.91	585.6
Is	43824.0	0.052734	0.760375	0.00	0.00	0.00	0.00	27.0
Ir	43824.0	0.194916	1.415867	0.00	0.00	0.00	0.00	36.0

In [7]:
```python
# check missing values
df.isnull().sum()   # in target we have null values.
```

Out[7]:
```
No          0
year        0
month       0
day         0
hour        0
pm2.5    2067
DEWP        0
TEMP        0
PRES        0
cbwd        0
Iws         0
Is          0
Ir          0
dtype: int64
```

In [8]:
```python
# to deal with null value we can dropna,ffill, bfill,3m.
df.dropna(axis=0,inplace=True)
```

In [9]:
```python
df.isnull().sum()   # drop the missing rows-->axis=0
```

Out[9]:
```
No       0
year     0
month    0
day      0
hour     0
pm2.5    0
DEWP     0
TEMP     0
PRES     0
cbwd     0
Iws      0
Is       0
Ir       0
dtype: int64
```

In [10]:
```python
# categorical variable
df.dtypes=='object'
```

Out[10]:
```
No       False
year     False
month    False
day      False
hour     False
pm2.5    False
DEWP     False
TEMP     False
PRES     False
cbwd      True
Iws      False
Is       False
Ir       False
dtype: bool
```

In [11]:
```python
df = df[df['DEWP'] > 0]
df = df[df['TEMP'] > 0]
```

In [12]:
```python
df.shape
```

Out[12]:
```
(21758, 13)
```

In [13]:
```python
df['cbwd'].nunique()  # 4 unique value
```

Out[13]:
```
4
```

In [14]:
```python
df1 = pd.get_dummies(df['cbwd'])
```

In [15]:
```python
df = pd.concat([df,df1],axis=1)
```

In [16]:
```python
df
```

Out[16]:

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir	NE	NW	SE	cv
1315	1316	2010	2	24	19	368.0	2	6.0	1006.0	cv	0.89	0	0	0	0	0	1
1316	1317	2010	2	24	20	309.0	3	7.0	1007.0	SE	1.79	0	0	0	0	1	0
1317	1318	2010	2	24	21	284.0	3	6.0	1008.0	SE	3.58	0	0	0	0	1	0
1318	1319	2010	2	24	22	244.0	2	6.0	1009.0	SE	6.71	0	0	0	0	1	0
1319	1320	2010	2	24	23	218.0	3	5.0	1010.0	cv	0.89	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
43059	43060	2014	11	30	3	194.0	3	4.0	1021.0	NW	3.58	0	0	0	1	0	0
43060	43061	2014	11	30	4	227.0	2	4.0	1021.0	NW	6.71	0	0	0	1	0	0
43061	43062	2014	11	30	5	242.0	2	3.0	1021.0	NW	9.84	0	0	0	1	0	0
43062	43063	2014	11	30	6	235.0	2	3.0	1021.0	NW	11.63	0	0	0	1	0	0
43063	43064	2014	11	30	7	168.0	2	4.0	1022.0	NW	14.76	0	0	0	1	0	0

21758 rows × 17 columns

In [17]:
```python
df.drop('cbwd',axis=1,inplace=True)
```

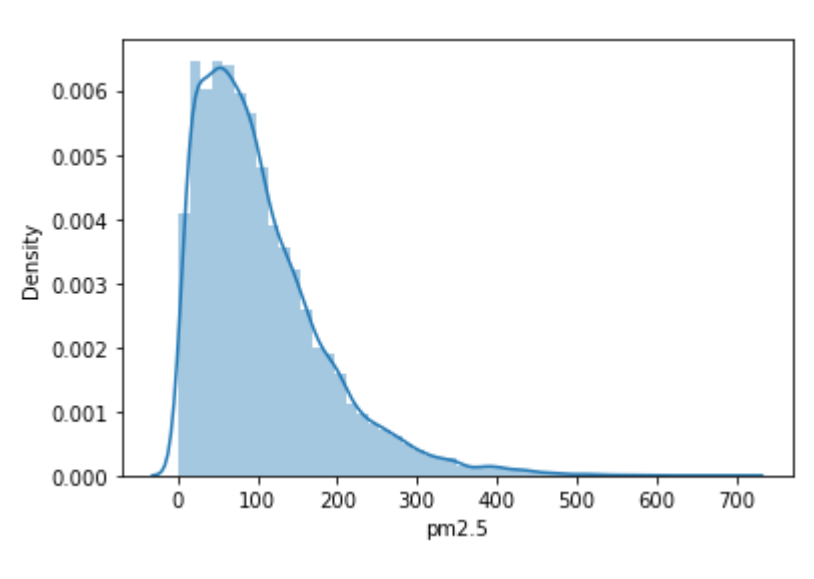
In [18]:
```python
df.head(2)   # cbwd is removed and got dummies.
```

Out[18]:

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	Iws	Is	Ir	NE	NW	SE	cv
1315	1316	2010	2	24	19	368.0	2	6.0	1006.0	0.89	0	0	0	0	0	1
1316	1317	2010	2	24	20	309.0	3	7.0	1007.0	1.79	0	0	0	0	1	0

In [19]:
```python
# plt.figure(figsize=(18,10))
sns.distplot(df['pm2.5'],bins=50)
```

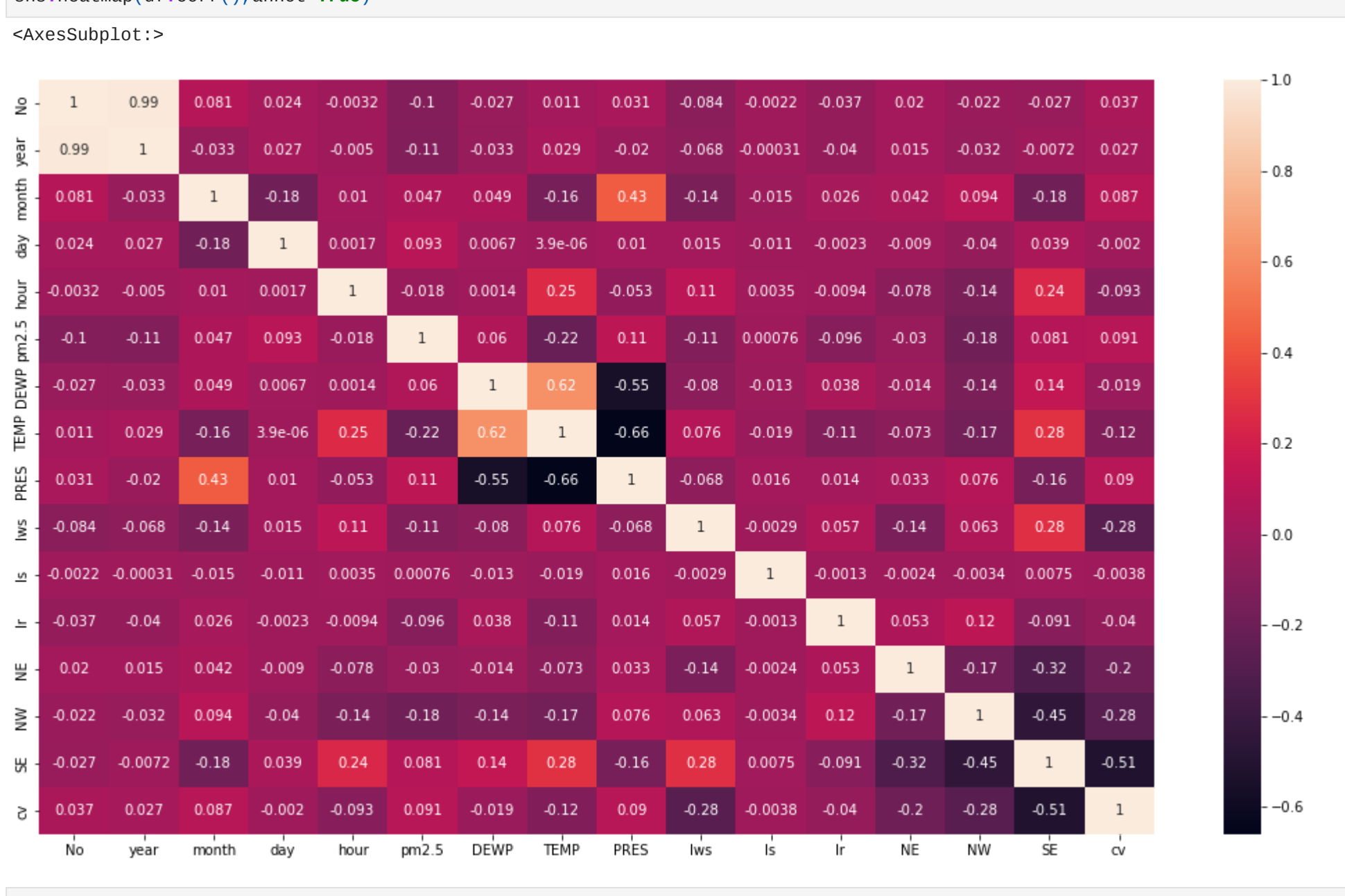
Out[19]:
```
<AxesSubplot:xlabel='pm2.5', ylabel='Density'>
```



In [20]:
```python
# df['PM2.5'] = np.log(df['pm2.5'])
```

In [21]:
```python
# visual
plt.figure(figsize=(18,10))
sns.heatmap(df.corr(),annot=True)
```

Out[21]:
```
<AxesSubplot:>
```

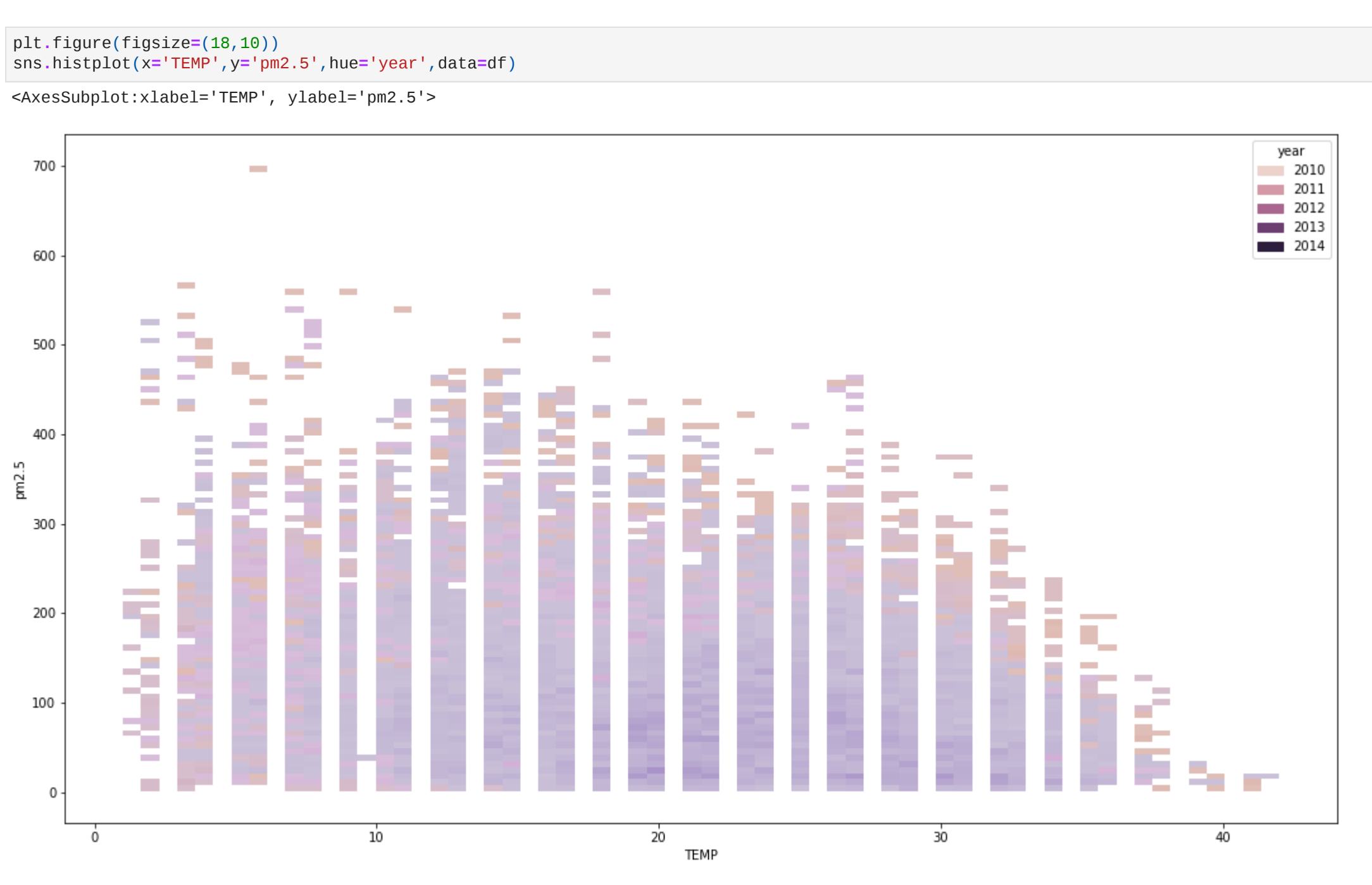


In [22]:
```python
df['year'].unique()
```

Out[22]:
```
array([2010, 2011, 2012, 2013, 2014], dtype=int64)
```

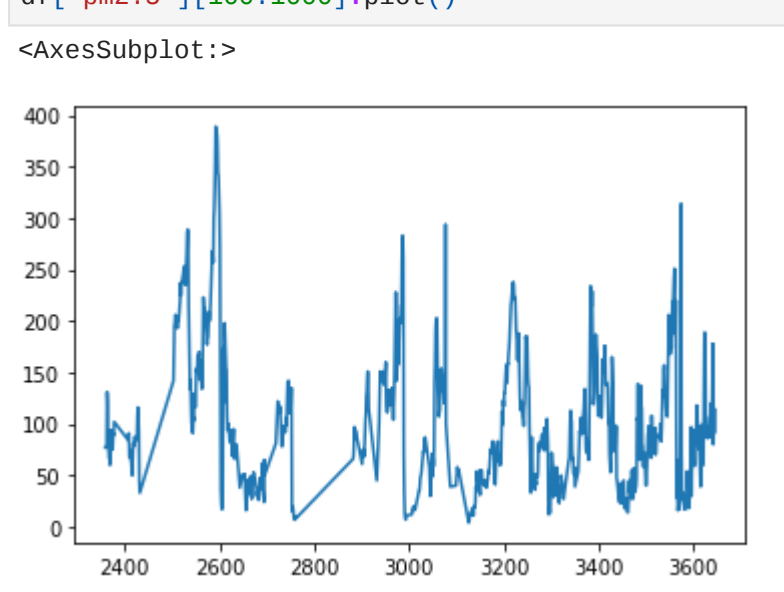
In [23]:
```python
plt.figure(figsize=(18,10))
sns.histplot(x='TEMP',y='pm2.5',hue='year',data=df)
```

Out[23]:
```
<AxesSubplot:xlabel='TEMP', ylabel='pm2.5'>
```



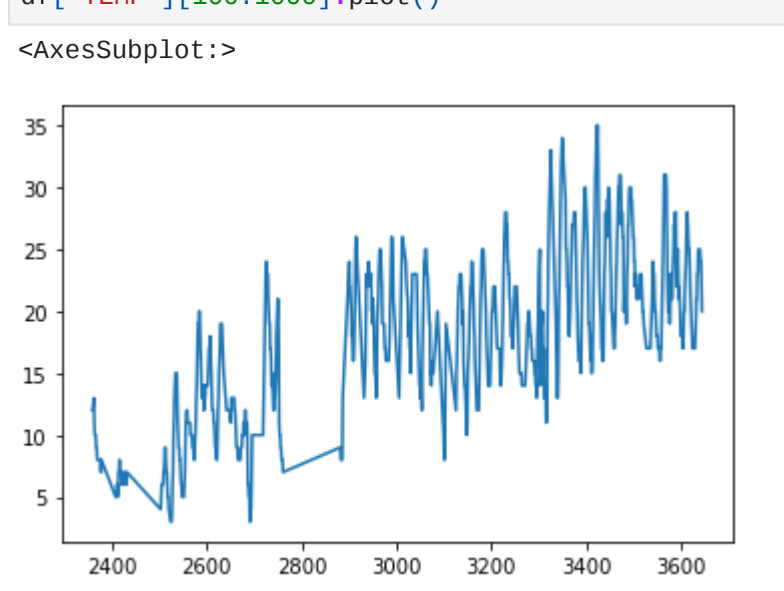
In [24]:
```python
df['pm2.5'][100:1000].plot()
```

Out[24]:
```
<AxesSubplot:>
```



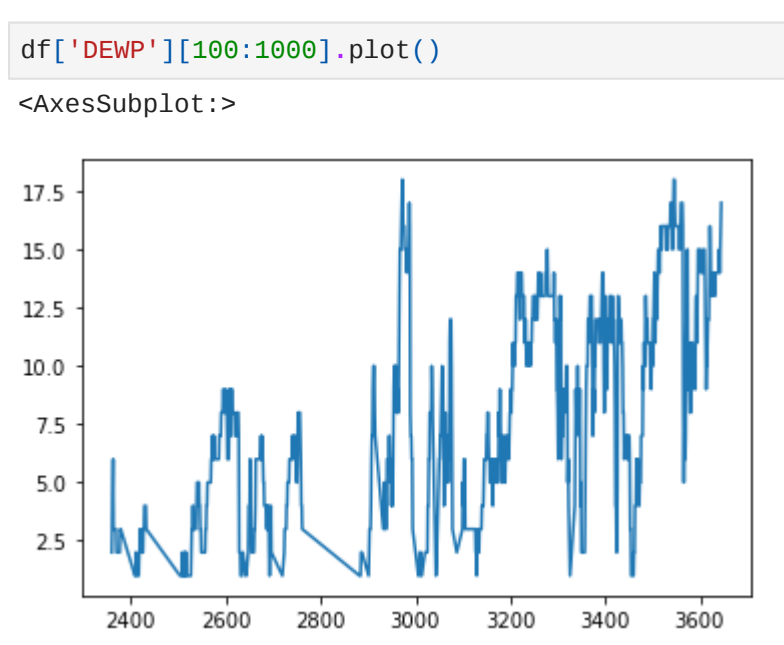
In [25]:
```python
df['TEMP'][100:1000].plot()
```

Out[25]:
```
<AxesSubplot:>
```



In [26]:
```python
df['DEWP'][100:1000].plot()
```

Out[26]:
```
<AxesSubplot:>
```



In [27]:
```python
df.columns
```

Out[27]:
```
Index(['No', 'year', 'month', 'day', 'hour', 'pm2.5', 'DEWP', 'TEMP', 'PRES',
       'Iws', 'Is', 'Ir', 'NE', 'NW', 'SE', 'cv'],
      dtype='object')
```

In [28]:
```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [29]:
```python
df1 = df[['DEWP','TEMP', 'PRES','Iws', 'Is', 'Ir', 'NE', 'NW', 'SE', 'cv']]
import statsmodels as sm
data = sm.tools.add_constant(df1)
series = pd.Series([variance_inflation_factor(df1.values,i) for i in range(df1.shape[1])],index=df1.columns)
series
```

Out[29]:
```
DEWP        1.817447
TEMP        2.324700
PRES        1.896035
Iws         1.174493
Is          1.000595
Ir          1.057735
NE       4187.394102
NW       7581.569333
SE      17215.065842
cv       9298.934220
dtype: float64
```

In [30]:
```python
df2 = df[['DEWP','TEMP', 'PRES','Iws', 'Is', 'Ir']]
import statsmodels as sm
data = sm.tools.add_constant(df2)
series = pd.Series([variance_inflation_factor(df2.values,i) for i in range(df2.shape[1])],index=df2.columns)
series
```

Out[30]:
```
DEWP     9.080935
TEMP    16.565522
PRES     9.901542
Iws      1.418709
Is       1.000427
Ir       1.077952
dtype: float64
```

In [31]:
```python
y = df['pm2.5']
x = df[['DEWP','TEMP', 'PRES','Iws', 'Is', 'Ir', 'NE', 'NW', 'SE', 'cv']]
```

In [32]:
```python
y.shape,x.shape
```

Out[32]:
```
((21758,), (21758, 10))
```

In [58]:
```python
# model implementation - splitting, scaling and model building

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,random_state=42)
```

In [59]:
```python
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
```

In [60]:
```python
x_train = ss.fit_transform(x_train)
x_test = ss.transform(x_test)
print(x_train.shape)
print(x_test.shape)
```

```
(15230, 10)
(6528, 10)
```

In [61]:
```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,mean_absolute_error,explained_variance_score
mse = mean_squared_error
mae = mean_absolute_error
```

In [62]:
```python
RF = RandomForestRegressor()
RF.fit(x_train,y_train)
RF_score = RF.score(x_train,y_train)
RF_pred = RF.predict(x_test)
print('RF Regressor',RF_score)     # 88 % on training
```

```
RF Regressor 0.8817636924606274
```

In [63]:
```python
print('RF on testing',explained_variance_score(RF_pred,y_test))  # -60 % on testing like overfit so hyperparameter tuning
```

```
RF on testing -0.6691269415004197
```

In [64]:
```python
RF_mae = mae(y_test,RF_pred)
RF_mse = mse(y_test,RF_pred)
RF_rmse = np.sqrt(RF_mse)
```

In [65]:
```python
print('Random Forest Regressor')
RF_mae,RF_mse,RF_rmse
```

```
Random Forest Regressor
```

Out[65]:
```
(47.636537060136554, 4570.119042404703, 67.60265558692723)
```

Overfitting - so cross validation

In [88]:
```python
from sklearn.model_selection import RandomizedSearchCV
param_grid = {#'n_estimators': [100, 200, 300],
              'max_depth': [None, 5, 10],
              'max_features': ['sqrt', 'log2']
             }

rf = RandomizedSearchCV(RF, param_distributions=param_grid,cv=10,n_iter=10)
rf.fit(x_train,y_train)
print('best param',rf.best_params_)
print('score',rf.best_score_)
```

```
best param {'max_features': 'log2', 'max_depth': 10}
score 0.2963333787569263
```

In [89]:
```python
RF_model = rf.best_estimator_
test_score_rf = RF_model.score(x_test,y_test)
print('Random Forest score',test_score_rf)            # 30 % on testing
```

```
Random Forest score 0.29152297938390026
```

Linear Regression

In [68]:
```python
from sklearn.decomposition import PCA
pca=PCA()
pca.fit(x_train)
print(pca.components_)
print(pca.explained_variance_)
```

```
[[-4.62783922e-01 -5.47218003e-01  4.91659383e-01 -1.33807426e-01
  -3.46944695e-18  6.95181227e-02  8.86300327e-02  2.00110377e-01
  -3.74495208e-01  1.83769545e-01]
 [-3.11890287e-01 -1.61116933e-01  1.96384141e-01  5.05873782e-01
   0.00000000e+00  4.53692058e-02 -1.81280346e-01  1.46512706e-01
   4.71243226e-01 -5.51211112e-01]
 [ 1.81403797e-01  9.70616353e-02 -2.37547957e-01  8.25353058e-02
  -2.22044605e-16  4.11800781e-01  1.68911923e-01  6.74907988e-01
  -3.81667476e-01 -3.08461694e-01]
 [-1.42548847e-02 -4.73646799e-02  6.47250021e-02 -1.23901835e-01
   4.16333634e-17  5.50895543e-02  8.46335180e-01 -3.34408374e-01
   6.23551222e-02 -3.78123229e-01]
 [ 7.82528293e-02 -9.27383064e-02  4.02046963e-02  1.67073438e-01
   3.33066907e-16  8.68542064e-01 -7.37822380e-02 -3.48629109e-01
   1.11520256e-01  2.48913623e-01]
 [-1.31804890e-01  8.17454430e-02 -1.54676591e-01  7.92859554e-01
   0.00000000e+00 -1.96305850e-01  2.57713778e-01 -1.19204457e-01
  -3.31183704e-01  3.07488710e-01]
 [-7.11840114e-01  2.68488954e-02 -6.57483030e-01 -2.09109156e-01
   1.11022302e-16  1.07738302e-01 -1.75340377e-02 -5.07297861e-02
   4.46939508e-02  8.13457324e-03]
 [-3.53882140e-01  8.04317004e-01  4.48620957e-01 -6.12952115e-02
  -1.11022302e-16  1.26432901e-01  1.93276361e-02  3.79676757e-02
  -6.56352927e-02  2.67460384e-02]
 [ 1.14062185e-16 -8.47323084e-16  3.56555229e-16  3.61793537e-16
  -1.03974380e-04 -1.64040265e-16  3.76731293e-01  4.81033361e-01
   5.99749931e-01  5.16701508e-01]
 [ 6.01952919e-18 -1.14979795e-16 -1.14125537e-16  9.06440140e-17
  -9.99999995e-01  1.51272569e-16 -3.91704029e-05 -5.00151459e-05
  -6.23586278e-05 -5.37237194e-05]]
[2.45432469e+00 1.62625214e+00 1.33726217e+00 1.16499753e+00
 9.49404022e-01 7.42438330e-01 4.25624683e-01 3.00287417e-01
 5.13288289e-30 3.78521440e-35]
```

In [69]:
```python
x_train = pca.transform(x_train)
x_train.shape
```

Out[69]:
```
(15230, 10)
```

In [70]:
```python
x_test = pca.transform(x_test)
x_test.shape
```

Out[70]:
```
(6528, 10)
```

In [71]:
```python
from sklearn.linear_model import LinearRegression,Ridge
```

In [72]:
```python
lr = LinearRegression()
lr.fit(x_train,y_train)
train_score = lr.score(x_train,y_train)
lr_pred = lr.predict(x_test)
print('Linear Regression Training score',train_score)
```

```
Linear Regression Training score 0.1847777624955791
```

In [73]:
```python
lr_mae = mae(y_test,lr_pred)
```

In [74]:
```python
print('Linear Regression testing score',explained_variance_score(y_test,lr_pred))
```

```
Linear Regression testing score 0.1782127811876898
```

In [75]:
```python
lr_mse = mse(y_test,lr_pred)
lr_rmse = np.sqrt(lr_mse)
```

In [76]:
```python
lr_mae,lr_mse,lr_rmse
```

Out[76]:
```
(54.68663472657801, 5486.9756743680655, 74.07412283900543)
```

RIDGE REGRESSION

In [77]:
```python
# ridge regression
r = Ridge()
r.fit(x_train,y_train)
r_score = r.score(x_train,y_train)
r_pred = r.predict(x_test)
print('RIDGE training score',r_score)
```

```
RIDGE training score 0.18477775767132398
```

In [78]:
```python
print('RF on testing',explained_variance_score(y_test,r_pred))
```

```
RF on testing 0.1782111099108883
```

In [79]:
```python
r_mae = mae(y_test,r_pred)
r_mse = mse(y_test,r_pred)
r_rmse = np.sqrt(r_mse)
```

In [80]:
```python
print('Ridge metrics')
r_mae,r_mse,r_rmse
```

```
Ridge metrics
```

Out[80]:
```
(54.68661616501064, 5486.986978053476, 74.07419913879242)
```

In [81]:
```python
from sklearn.model_selection import RandomizedSearchCV
param_grid = {'alpha':[0.001,0.01,0.1,1,10],
             'solver':['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga', 'lbfgs']
             }
ridge_tune = RandomizedSearchCV(r,param_grid,cv=5)
ridge_tune.fit(x_train,y_train)
print('best param',ridge_tune.best_params_)
print('score',ridge_tune.best_score_)
```

```
best param {'solver': 'sag', 'alpha': 0.01}
score 0.18351749906408105
```

In [82]:
```python
ridge_model = ridge_tune.best_estimator_
test_score = ridge_model.score(x_test,y_test)
print('ridge test score',test_score)
```

```
ridge test score 0.17778910564257666
```

In [ ]:

In [ ]:

In [ ]: