In [1]:
```
import numpy as np
import pandas as pd


import os
for dirname, _, filenames in os.walk(r'C:\Users\LENOVO\Downloads\time siries analysis'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting.zip
C:\Users\LENOVO\Downloads\time siries analysis\time_siries_analysis_project_work (1).py
C:\Users\LENOVO\Downloads\time siries analysis\time_siries_analysis_project_work.ipynb
C:\Users\LENOVO\Downloads\time siries analysis\time_siries_analysis_project_work.py
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\benchmark.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\test.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\train.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf1.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf2.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf3.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf4.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf5.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf6.csv
C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf7.csv
```

In [6]:
```
import pandas as pd
import numpy as np
import os

df1 = pd.read_csv(r'C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\train.csv')

df1.dtypes

df1['date'] = pd.to_datetime(df1['date'], format='%Y%m%d%H')
df1['date'].min()
```

Out[6]:
```
Timestamp('2009-07-01 00:00:00')
```

In [5]:
```
df1['date'].max()
```

Out[5]:
```
Timestamp('2012-06-26 12:00:00')
```

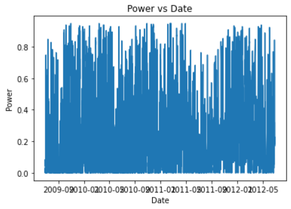
In [7]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18757 entries, 0 to 18756
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   date    18757 non-null  datetime64[ns]
 1   wp1     18757 non-null  float64
 2   wp2     18757 non-null  float64
 3   wp3     18757 non-null  float64
 4   wp4     18757 non-null  float64
 5   wp5     18757 non-null  float64
 6   wp6     18757 non-null  float64
 7   wp7     18757 non-null  float64
dtypes: datetime64[ns](1), float64(7)
memory usage: 1.1 MB
```

In [11]:
```
import matplotlib.pyplot as plt

K = df1[['date', 'wp1']]

# Plot the data
plt.plot(K['date'], K['wp1'])
plt.xlabel('Date')
plt.ylabel('Power')
plt.title('Power vs Date')
plt.show()
```



In [10]:
```
def data_processing(df_wf1, y_col):

    df_wf1['date'] = pd.to_datetime(df_wf1['date'], format='%Y%m%d%H')
    df_wf1['hors_delta'] = pd.to_timedelta(df_wf1.hors, unit='hours')
    df_wf1['forecast_date'] = df_wf1['date']+ df_wf1['hors_delta']
    df_wf1.drop(columns= 'hors_delta', inplace=True)
    df_wf1a = df_wf1.merge(df1[['date',y_col]], how='left', left_on='forecast_date', right_on='date')
    df_wf1a.rename(columns={y_col:'power'}, inplace=True)
    df_wf1a.rename(columns={'date_x':'date'}, inplace=True)
    df_wf1a.drop(columns='date_y', inplace=True)

    df_wf1b = df_wf1a.groupby('forecast_date').agg({'power': 'mean',
                                                    'hors':'count'}).reset_index()
    df_wf1b.dropna(subset=['power'], inplace=True)
    df_wf1b = df_wf1b[df_wf1b.hors==4]
    df_wf1b.drop(columns='hors', inplace=True)

    df_wf1a['u_avg'] = df_wf1a.groupby('forecast_date').u.transform(lambda x: x.rolling(2, min_periods=1).mean())
    df_wf1a['v_avg'] = df_wf1a.groupby('forecast_date').v.transform(lambda x: x.rolling(2, min_periods=1).mean())
    df_wf1a['ws_avg'] = df_wf1a.groupby('forecast_date').ws.transform(lambda x: x.rolling(2, min_periods=1).mean())
    df_wf1a['wd_avg'] = df_wf1a.groupby('forecast_date').wd.transform(lambda x: x.rolling(2, min_periods=1).mean())

    df_wf1c = df_wf1a.groupby('forecast_date').last().reset_index()
    df_wf1c.columns

    df_wf1c['year'] = df_wf1c.forecast_date.dt.year
    df_wf1c['month'] = df_wf1c.forecast_date.dt.month
    df_wf1c['day'] = df_wf1c.forecast_date.dt.day
    df_wf1c['hour'] = df_wf1c.forecast_date.dt.hour
    df_wf1c['dayofweek'] = df_wf1c.forecast_date.dt.dayofweek
    df_wf1c['weekday'] = np.where(df_wf1c.forecast_date.dt.dayofweek>4,0,1)

    df_wf1c['wd_radian'] = df_wf1c['wd']*((22/7)/180)

    df_wf1c.columns

    df_wf1d = df_wf1b.merge(df_wf1c[['forecast_date','u_avg', 'v_avg', 'ws_avg', 'wd_avg', 'year', 'month', 'day',
                                     'hour', 'dayofweek', 'weekday', 'wd_radian']],
                            on='forecast_date', how='left')

    return df_wf1d


df_wf1 = pd.read_csv(r'C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf1.csv')
farm1 = data_processing(df_wf1,'wp1')
farm1['farm_no'] = 'farm1'

df_wf2 = pd.read_csv(r'C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf2.csv')
farm2 = data_processing(df_wf2,'wp2')
farm2['farm_no'] = 'farm2'

df_wf3 = pd.read_csv(r'C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf3.csv')
farm3 = data_processing(df_wf3,'wp3')
farm3['farm_no'] = 'farm3'

df_wf4 = pd.read_csv(r'C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf4.csv')
farm4 = data_processing(df_wf4,'wp4')
farm4['farm_no'] = 'farm4'

df_wf5 = pd.read_csv(r'C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf5.csv')
farm5 = data_processing(df_wf5,'wp5')
farm5['farm_no'] = 'farm5'

df_wf6 = pd.read_csv(r'C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf6.csv')
farm6 = data_processing(df_wf6,'wp6')
farm6['farm_no'] = 'farm6'

df_wf7 = pd.read_csv(r'C:\Users\LENOVO\Downloads\time siries analysis\GEF2012-wind-forecasting\windforecasts_wf7.csv')
farm7 = data_processing(df_wf7,'wp7')
farm7['farm_no'] = 'farm7'

df = pd.concat([farm1, farm2, farm3, farm4, farm5, farm6, farm7])

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['farm_no']= label_encoder.fit_transform(df['farm_no'])
df['farm_no'].unique()

y = df[['power']]
X = df.drop(columns=['power','forecast_date'])

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=13,
                                                    stratify=X.farm_no)


from xgboost import XGBRegressor

xgb_r = XGBRegressor(alpha=0.02, learning_rate=0.1, max_depth=15,
                     min_child_weight=3, n_estimators=100,
                     subsample=0.8, tree_method='hist')

xgb_r.fit(X_train, y_train)

pred = xgb_r.predict(X_test)

from sklearn.metrics import mean_squared_error, r2_score
rmse = np.sqrt(mean_squared_error(y_test, pred))
r2 = r2_score(y_test, pred)
print(r2)
```

```
0.8549756085102491
```

In [14]:
```
print(pred)
```

```
[0.06473721 0.16594554 0.33033013 ... 0.02591081 0.2899041  0.7101369 ]
```

In [ ]:
```

```