```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


import os
for dirname, _, filenames in os.walk('./'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


# ## load data

# In[3]:


df = pd.read_csv('wind_dataset.csv')
df


# ## Exploratory Data Analysis

# In[4]:


df.shape


# In[5]:


df.info()


# ### We Find that all data good but there is problem ,The "DATE" column is worng data type

# In[6]:


round(df.describe() , 3)
```

Is df has dublicate?

In[7]:

```
df.duplicated().sum()
```

Is df has null ?

In[8]:

```
df.isnull().sum()
```

Fill null rows! , then we know to know the most repeated value in each column

In[9]:

```
colu = ["IND.1","T.MAX" ,"IND.2" ,"T.MIN" ,"T.MIN.G" ]
for i in colu:
    print(i)
    print(df[i].mode())
```

In[10]:

```
FilterInd1 = 0.0
FilterTmax = 10.0
FilterInd =0.0
FilterTMIN = 9.0
FilterTMIN_g = 5.0
df["IND.1"].fillna(FilterInd1 , inplace = True)
df["T.MAX"].fillna(FilterTmax , inplace = True)
df["IND.2"].fillna(FilterInd , inplace = True)
df["T.MIN"].fillna(FilterTMIN , inplace = True)
df["T.MIN.G"].fillna(FilterTMIN_g , inplace = True)
```

In[11]:

```
df.isnull().sum()


# ##### Done! , No (NUll) Vlaue.

# #### Fix "DATE" Column , Convert it to DateTime data type
#

# In[12]:


df['DATE'] =pd.to_datetime(df['DATE'])
df.info()


# In[13]:


df["Year"] = df['DATE'].dt.year
df["month"] = df['DATE'].dt.month
df["day"] = df['DATE'].dt.day


# ##### add 3 new columns for year and month and day

# In[14]:


df


# #### Done!

# ## EDA!

# In[15]:


df.hist(bins = 30 , ec = "white")
plt.grid(False)
plt.show()
```

Is "Month" Has a relation with "Wind"??

In[16]:

```
month = df["month"]
wind  = df['WIND']
plt.xlabel('Month')
plt.ylabel('Wind Speed')
plt.scatter(month , wind)
plt.show()
```

form this Scatter we found that: in (2,3,11,12) months the wind speed is high. , then we can estimate that it has a relation! , but we need some analysis to improve that!

correlation
#

In[17]:

```
sns.heatmap(df.corr(), cmap="crest",annot=True, fmt=".1f")
```

is tempreture has a relation with wind speed ?

In[18]:

```
plt.xlabel("T.max")
plt.ylabel("wind")
plt.scatter(df['T.MAX'],wind)
plt.show()
```

NO!

find relation between "all data" and "WIND"

In[19]:

```
for i in df.columns:
```

```
    plt.xlabel(i)
    plt.ylabel("Wind")
    plt.scatter(df[i] , df["WIND"] ,color ="c")
    plt.show()


# ### Is there an Outlier?! .....Then Drop it?!

# In[20]:


sns.boxplot(x=df['WIND'])


# In[21]:


filter1 = (df['WIND'] >25)
filter1.value_counts()


# #### there is 36 outlier value

# In[22]:


for i in np.where(df["WIND"]>=23):
    df.drop(i,inplace = True)


# ### most of outlier is fixed ! but still rain has outliers

# In[23]:


for k in np.where(df["RAIN"]>=25):
    df.drop(k,inplace =True)


# In[24]:


sns.boxplot(x=df['IND'], y=df["WIND"])
```

That is the big evidence that most of outliers is fixed

There is more outlier in data ....?

In[25]:

```
sns.boxplot(x=df['month'] ,y=df["WIND"])
```

what is the best Month to turn on the Turbine?

In[26]:

```
plt.hist(df["WIND"]) ## to get the most repeated value
```

In[27]:

```
filter2 =df[(df['WIND'] >=10)]
pd.DataFrame(filter2['month'].value_counts())
```

The best Month to Turn ON Turbine is between (11 , 3) , beacuse we found that the maximum wind speed in these months and if we don't remove the outlier the output is same

In[28]:

```
filter3 =df[(df['WIND'] >=10) & (df["RAIN"]<=10)  ]
pd.DataFrame(filter3['month'].value_counts())
```

If we add another columns , found that the best month is between(11,4)

but the best of the best is (12,1)

To conclusion this ...

In[29]:

```
import seaborn as sns
g = sns.pairplot(data=df, diag_kind="hist", dropna=True)
g.map_lower(sns.kdeplot, levels=4, color=".2")


# #### this is the correlation  between all data with each other

# In[30]:


df.drop("Year" ,axis =1 ,inplace =True)
df.drop("day" ,axis =1 ,inplace =True)
df.drop("DATE" ,axis =1 ,inplace =True)


# ## Prediction

# In[31]:


X = df.drop('WIND',axis =1)
y = df["WIND"]


# ### Linear regression

# In[32]:


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=0
,shuffle=False)


# In[33]:


from sklearn.linear_model import LinearRegression
from sklearn import metrics
reg =LinearRegression()
reg.fit(X_train,y_train)
predict = reg.predict(X_test)
print('MAE:', metrics.mean_absolute_error(y_test, predict))
```

```
print('MSE:', metrics.mean_squared_error(y_test, predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predict)))
```

```
# In[ ]:
```

```
# ### Decision Tree
```

```
# In[34]:
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
tree = DecisionTreeRegressor()
tree.fit(X_train,y_train)
predictTree = tree.predict(X_test)
print('MSE' ,metrics.mean_absolute_error(y_test, predict))
print('MSE:', metrics.mean_squared_error(y_test, predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predict)))
```

```
# ### make new prediction for new data
```

```
# ### new data for regression!
```

```
# In[35]:
```

```
x_input =pd.DataFrame(np.array([[0,10.4,0.0,7.2,1.0,-1.5,-7.5,1]]))
reg.predict(x_input)
```

```
# ### new data for Decision Tree!
```

```
# In[36]:
```

```
x_input =pd.DataFrame(np.array([[0,10.4,0.0,7.2,1.0,-1.5,-7.5,1]]))
tree.predict(x_input)
```

Make Cross Validation

In[37]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error, mean_absolute_error , make_scorer

mse = make_scorer(mean_squared_error)
mae = make_scorer(mean_absolute_error)
reg = LinearRegression()
scores = cross_val_score(reg, X, y, cv=150, scoring = mse )
```

In[38]:

```
print(f"The MEAN-squared-error:  {scores.min()}")
```

In[39]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error , mean_squared_error , make_scorer

mseTree = make_scorer(mean_squared_error)
maeTree = make_scorer(mean_absolute_error)
scoresTree = cross_val_score(tree , X, y , cv =150, scoring = mse)
```

In[40]:

```
print(f"The mean_absolute_error : {scoresTree.min()}" )
```

DONE...