|  |  |
| --- | --- |
| **Date** | **14 November 2022** |
| **Team ID** | **PNT2022TMID45948** |
| **Project Name** | **Predicting the energy output of wind turbine based on weather condition** |
| **Maximum Marks** | **4 Marks** |

1666 lines (1666 sloc) 66.6 KB

In [1]:

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.linear\_model **import** Lasso

**from** sklearn.linear\_model **import** Ridge

**from** sklearn.metrics **import** mean\_squared\_error , r2\_score

**import** joblib

**%**matplotlib inline

In [2]:

data **=** pd**.**read\_csv('https://raw.githubusercontent.com/IBM-EPBL/IBM-Project-27 data**.**rename(columns **=** {'LV ActivePower (kW)':'ActivePower(kW)',

"Wind Speed (m/s)":"WindSpeed(m/s)",

"Wind Direction (°)":"WindDirection","Theoretical\_Powe inplace **= True**)

data**.**head()

Out[2]:

In [3]:

data**.**shape

00:00

00:10

|  |  |  |  |
| --- | --- | --- | --- |
| **Date/Time ActivePower(kW)** | **WindSpeed(m/s)** | **TheoreticalPowerCurve(KWh)** | **WindDirect** |
| **0** 01 01 2018 380.047791 | 5.311336 | 416.328908 | 259.994 |
| **1** 01 01 2018 453.769196 | 5.672167 | 519.917511 | 268.641 |
| **2** 01 01 2018 306.376587 | 5.216037 | 390.900016 | 272.564 |
| **3** 01 01 2018 419.645905 | 5.659674 | 516.127569 | 271.258 |
| **4** 01 01 2018 380.650696 | 5.577941 | 491.702972 | 265.674 |

00:20

00:30

00:40



Out[3]: (50530, 5)

In [4]:

data**.**describe()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Out[4]: | **ActivePower(kW)** | **WindSpeed(m/s)** | **TheoreticalPowerCurve(KWh)** | **WindDirection** |
|  | **count** 50530.000000 | 50530.000000 | 50530.000000 | 50530.000000 |
|  | **mean** 1307.684332 | 7.557952 | 1492.175463 | 123.687559 |
|  | **std** 1312.459242 | 4.227166 | 1368.018238 | 93.443736 |
|  | **min** -2.471405 | 0.000000 | 0.000000 | 0.000000 |
|  | **25%** 50.677890 | 4.201395 | 161.328167 | 49.315437 |
|  | **50%** 825.838074 | 7.104594 | 1063.776283 | 73.712978 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **75%** | 2482.507568 | 10.300020 | 2964.972462 | 201.696720 |
| **max** | 3618.732910 | 25.206011 | 3600.000000 | 359.997589 |

In [5]:

data**.**info()

RangeIndex: 50530 entries, 0 to 50529 Data columns (total 5 columns):

# Column Non-Null Count Dtype

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 Date/Time | 50530 | non-null |  | object |
| 1 ActivePower(kW) | 50530 | non-null |  | float64 |
| 2 WindSpeed(m/s) | 50530 | non-null |  | float64 |
| 3 TheoreticalPowerCurve(KWh) | 50530 | non-null |  | float64 |
| 4 WindDirection | 50530 | non-null |  | float64 |

dtypes: float64(4), object(1) memory usage: 1.9+ MB

In [6]:

data**.**isnull()**.**any()

Out[6]:

In [7]:

Date/Time False

ActivePower(kW) False

WindSpeed(m/s) False TheoreticalPowerCurve(KWh) False WindDirection False dtype: bool

# Data Preprocessing

data['Date/Time'] **=** pd**.**to\_datetime(data['Date/Time'],format**=**'%d %m %Y %H:%M') data['year'] **=** data['Date/Time']**.**dt**.**year

data['month'] **=** data['Date/Time']**.**dt**.**month data['day'] **=** data['Date/Time']**.**dt**.**day

data['Hour'] **=** data['Date/Time']**.**dt**.**hour

data['minute'] **=** data['Date/Time']**.**dt**.**minute data**.**head()

Out[7]:

|  |  |  |  |
| --- | --- | --- | --- |
| **Date/Time ActivePower(kW)** | **WindSpeed(m/s)** | **TheoreticalPowerCurve(KWh)** | **WindDirect** |
| 2018-01-  **0** 01 380.047791  00:00:00 | 5.311336 | 416.328908 | 259.994 |
| 2018-01-  **1** 01 453.769196 | 5.672167 | 519.917511 | 268.641 |
| 00:10:00 |  |  |  |
| 2018-01-  **2** 01 306.376587 | 5.216037 | 390.900016 | 272.564 |
| 00:20:00 |  |  |  |
| 2018-01-  **3** 01 419.645905 | 5.659674 | 516.127569 | 271.258 |
| 00:30:00 |  |  |  |
| 2018-01-  **4** 01 380 650696 | 5 577941 | 491 702972 | 265 674 |

00:40:00

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In [8]: | data["Date/Time"] **=** pd**.**to\_datetime(data["Date/Time"], format **=** "%d %m %Y %H:% | | | | | |
|  | data |  |  |  |  |  |
| Out[8]: |  | **Date/Time** | **ActivePower(kW)** | **WindSpeed(m/s)** | **TheoreticalPowerCurve(KWh)** | **WindD** |
|  |  | 2018-01- |  |  |  |  |
|  | **0** | 01  00:00:00 | 380.047791 | 5.311336 | 416.328908 | 25 |
|  |  | 2018-01- |  |  |  |  |
|  | **1** | 01  00:10:00 | 453.769196 | 5.672167 | 519.917511 | 26 |
|  |  | 2018-01- |  |  |  |  |
|  | **2** | 01  00:20:00 | 306.376587 | 5.216037 | 390.900016 | 27 |
|  |  | 2018-01- |  |  |  |  |
|  | **3** | 01  00:30:00 | 419.645905 | 5.659674 | 516.127569 | 27 |
|  |  | 2018-01- |  |  |  |  |
|  | **4** | 01  00:40:00 | 380.650696 | 5.577941 | 491.702972 | 26 |
|  | **...** | ... | ... | ... | ... |  |
|  |  | 2018-12- |  |  |  |  |
|  | **50525** | 31  23:10:00 | 2963.980957 | 11.404030 | 3397.190793 | 8 |
|  |  | 2018-12- |  |  |  |  |
|  | **50526** | 31  23:20:00 | 1684.353027 | 7.332648 | 1173.055771 | 8 |
|  |  | 2018-12- |  |  |  |  |
|  | **50527** | 31  23:30:00 | 2201.106934 | 8.435358 | 1788.284755 | 8 |
|  |  | 2018-12- |  |  |  |  |
|  | **50528** | 31  23:40:00 | 2515.694092 | 9.421366 | 2418.382503 | 8 |
|  |  | 2018-12- |  |  |  |  |
|  | **50529** | 31  23:50:00 | 2820.466064 | 9.979332 | 2779.184096 | 8 |

50530 rows × 10 columns

# Splitting the dataset

In [9]:

X**=**data[['WindSpeed(m/s)','WindDirection']] X**.**head()

|  |  |  |  |
| --- | --- | --- | --- |
| Out[9]: |  | **WindSpeed(m/s)** | **WindDirection** |
|  | **0** | 5.311336 | 259.994904 |
|  | **1** | 5.672167 | 268.641113 |
|  | **2** | 5.216037 | 272.564789 |
|  | **3** | 5.659674 | 271.258087 |
|  | **4** | 5.577941 | 265.674286 |

In [10]:

y **=** data['ActivePower(kW)'] y**.**head()

|  |  |  |
| --- | --- | --- |
| Out[10]: | 0 | 380.047791 |
|  | 1 | 453.769196 |
|  | 2 | 306.376587 |
|  | 3 | 419.645905 |
|  | 4 | 380.650696 |

Name: ActivePower(kW), dtype: float64

In [11]:

X\_train, X\_test,y\_train, y\_test **=** train\_test\_split(X,y ,

random\_state**=**6, test\_size**=**0.25)

# Importing the regression Models

In [12]:

**from** sklearn.tree **import** DecisionTreeRegressor

**from** sklearn.svm **import** SVR

**from** sklearn.linear\_model **import** LinearRegression **from** sklearn.ensemble **import** RandomForestRegressor **from** xgboost **import** XGBRegressor

**from** sklearn.metrics **import** accuracy\_score,r2\_score,mean\_squared\_error xgr**=**XGBRegressor()

rf**=**RandomForestRegressor()

lr**=**LinearRegression()

dt**=**DecisionTreeRegressor() sm**=**SVR()

# Fitting the models with the dataset

In [13]:

model\_xg**=**xgr**.**fit(X\_train,y\_train) y\_xg**=**model\_xg**.**predict(X\_test)

model\_rf**=**rf**.**fit(X\_train,y\_train)

y\_rf**=**model\_rf**.**predict(X\_test)

model\_lr**=**lr**.**fit(X\_train,y\_train) y\_lr**=**model\_lr**.**predict(X\_test)

model\_dt**=**dt**.**fit(X\_train,y\_train)

y\_dt**=**model\_dt**.**predict(X\_test)

model\_sm**=**sm**.**fit(X\_train,y\_train) y\_sm**=**model\_sm**.**predict(X\_test)

[06:47:03] WARNING: /workspace/src/objective/regression\_obj.cu:152: reg:linea r is now deprecated in favor of reg:squarederror.

# Checking the metrics

In [14]:

print('R2-xgb',r2\_score(y\_test,y\_xg))

print('RMSE-xgb',np**.**sqrt(mean\_squared\_error(y\_test,y\_xg)))

print('R2-rf',r2\_score(y\_test,y\_rf))

print('RMSE-rf',np**.**sqrt(mean\_squared\_error(y\_test,y\_rf)))

print('R2-lr',r2\_score(y\_test,y\_lr))

print('RMSE-lr',np**.**sqrt(mean\_squared\_error(y\_test,y\_lr)))

print('R2-dt',r2\_score(y\_test,y\_dt))

print('RMSE-dt',np**.**sqrt(mean\_squared\_error(y\_test,y\_dt)))

print('R2-svm',r2\_score(y\_test,y\_sm))

print('RMSE-svm',np**.**sqrt(mean\_squared\_error(y\_test,y\_sm)))

R2-xgb 0.9222746826171284

RMSE-xgb 364.85477293970644

R2-rf 0.9097702879938478

RMSE-rf 393.10952377367164

R2-lr 0.8368251429450982

RMSE-lr 528.6465476346768

R2-dt 0.8388459591904157

RMSE-dt 525.3628747175155

R2-svm 0.005368134807760105

RMSE-svm 1305.1786596858901

# Hyperparameter Tuning

In [18]:

params**=**{

"colsample\_bylevel" : [ 0.3, 0.4, 0.5 , 0.7,0.9 ]

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| "learning\_rate" : | [0.05, 0.01,0.03,0.1, 0.15, 0.2] , | | | | | |  |
| "n\_estimators" : | [50, 100, 150, 200, 500, 800,1000,1500] | | | | | | , |
| "max\_depth" : | [ 3, 4, 5, 6, 8, 10, 12, 15,20,25], | | | | | |  |
| "min\_child\_weight" : | [ 1, 3, 5, 7 ,10,15,20,25], | | | | | |  |
| "gamma" : | [ 0.0, 0.1, 0.2 , 0.3, 0.4 ], | | | | | |  |
| "subsample" : | [ 0.1, 0.2 , 0.3, 0.4,0.6,0.8,1 ], | | | | | |  |
| "reg\_lambda" : | [ | 0.0, | 0.1, | 0.2 | , | 0.3, 0.4 ,0.6,0.8,1], | |
| "reg\_alpha" : | [ | 0.0, | 0.1, | 0.2 | , | 0.3, 0.4 ], | |
| "colsample\_bytree" : | [ | 0.3, | 0.4, | 0.5 | , | 0.7,0.9 ], | |

In [19]:

**from** sklearn.model\_selection **import** RandomizedSearchCV, GridSearchCV

In [20]:

**def** timer(start\_time**=None**):

**if not** start\_time:

start\_time **=** datetime**.**now()

**return** start\_time



**elif** start\_time:

thour, temp\_sec **=** divmod((datetime**.**now() **-** start\_time)**.**total\_seconds( tmin, tsec **=** divmod(temp\_sec, 60)

print('\n Time taken: %i hours %i minutes and %s seconds.' **%** (thour,

In [22]:

random\_search**=**RandomizedSearchCV(xgr,param\_distributions**=**params,n\_iter**=**10,n\_j

**from** datetime **import** datetime

start\_time **=** timer(**None**) *# timing starts from this point for "start\_time" var*

random\_search**.**fit(X\_train,y\_train)

timer(start\_time) *# timing ends here for "start\_time" variable*

Fitting 5 folds for each of 10 candidates, totalling 50 fits

[07:12:25] WARNING: /workspace/src/objective/regression\_obj.cu:152: reg:linea r is now deprecated in favor of reg:squarederror.

Time taken: 0 hours 12 minutes and 58.3 seconds.

In [23]:

random\_search**.**best\_estimator\_

*#XGBRegressor(colsample\_bylevel=0.4, colsample\_bytree=0.3, gamma=0.1, # learning\_rate=0.01, max\_depth=6, min\_child\_weight=25,*

*# n\_estimators=1500, reg\_alpha=0.1, reg\_lambda=0.8, subsample=0.6*

Out[23]:

In [24]:

XGBRegressor(colsample\_bylevel=0.4, colsample\_bytree=0.3, gamma=0.1, learning\_rate=0.01, max\_depth=6, min\_child\_weight=25,

n\_estimators=1500, reg\_alpha=0.1, reg\_lambda=0.8, subsample=0.6)

xg**=**XGBRegressor(colsample\_bylevel**=**0.4, colsample\_bytree**=**0.3, gamma**=**0.1, learning\_rate**=**0.01, max\_depth**=**6, min\_child\_weight**=**25,

n\_estimators**=**1500, reg\_alpha**=**0.1, reg\_lambda**=**0.8, subsample**=**0.6) x**=**xgr**.**fit(X\_train,y\_train)

y1**=**x**.**predict(X\_test)

r2\_score(y\_test,y1)

Out[24]:

[07:14:27] WARNING: /workspace/src/objective/regression\_obj.cu:152: reg:linea r is now deprecated in favor of reg:squarederror.

0.9222746826171284