Classification Assignment

November 4, 2022

Data Set Information:

Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Prediction task is to determine whether a person makes over 50K a year.

Attribute Information:

Listing of attributes:

50K, <=50K.

age: continuous. workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked. fnlwgt: continuous. education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool. education-num: continuous. marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces. relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. sex: Female, Male. capital-gain: continuous. capital-loss: continuous. hours-per-week: continuous. native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

[1]:
```
import pandas as pd
```

[16]:
```
df=pd.read_csv("adult2.txt",delimiter=",")
```

[17]:
```
df.head()
```

[17]:

	age	workclass	fnlwgt	education	education-num	\
0	39	State-gov	77516	Bachelors	13	
1	50	Self-emp-not-inc	83311	Bachelors	13	
2	38	Private	215646	HS-grad	9	
3	53	Private	234721	11th	7	

```
4   28           Private  338409   Bachelors            13

        marital-status          occupation    relationship    race      sex  \
0        Never-married        Adm-clerical   Not-in-family   White     Male
1   Married-civ-spouse     Exec-managerial         Husband   White     Male
2             Divorced   Handlers-cleaners   Not-in-family   White     Male
3   Married-civ-spouse   Handlers-cleaners         Husband   Black     Male
4   Married-civ-spouse      Prof-specialty            Wife   Black   Female

   capital-gain  capital-loss  hours-per-week  native-country  target
0          2174             0              40   United-States   <=50K
1             0             0              13   United-States   <=50K
2             0             0              40   United-States   <=50K
3             0             0              40   United-States   <=50K
4             0             0              40            Cuba   <=50K
```

[18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64
 11  capital-loss    32561 non-null  int64
 12  hours-per-week  32561 non-null  int64
 13  native-country  32561 non-null  object
 14  target          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[21]:
```
df.columns
```

[21]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'target'],
```

```
       dtype='object')
```

[22]:
```
df['workclass'].unique()
```

[22]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

[23]:
```
df['education'].unique()
```

[23]:
```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

[24]:
```
df['marital-status'].unique()
```

[24]:
```
array([' Never-married', ' Married-civ-spouse', ' Divorced',
       ' Married-spouse-absent', ' Separated', ' Married-AF-spouse',
       ' Widowed'], dtype=object)
```

[25]:
```
df['occupation'].unique()
```

[25]:
```
array([' Adm-clerical', ' Exec-managerial', ' Handlers-cleaners',
       ' Prof-specialty', ' Other-service', ' Sales', ' Craft-repair',
       ' Transport-moving', ' Farming-fishing', ' Machine-op-inspct',
       ' Tech-support', ' ?', ' Protective-serv', ' Armed-Forces',
       ' Priv-house-serv'], dtype=object)
```

[26]:
```
df['relationship'].unique()
```

[26]:
```
array([' Not-in-family', ' Husband', ' Wife', ' Own-child', ' Unmarried',
       ' Other-relative'], dtype=object)
```

[27]:
```
df['sex'].unique()
```

[27]:
```
array([' Male', ' Female'], dtype=object)
```

[28]:
```
df['race'].unique()
```

[28]:
```
array([' White', ' Black', ' Asian-Pac-Islander', ' Amer-Indian-Eskimo',
       ' Other'], dtype=object)
```

[29]:
```
df['native-country'].unique()
```

[29]:
```
array([' United-States', ' Cuba', ' Jamaica', ' India', ' ?', ' Mexico',
       ' South', ' Puerto-Rico', ' Honduras', ' England', ' Canada',
```

```
       ' Germany', ' Iran', ' Philippines', ' Italy', ' Poland',
       ' Columbia', ' Cambodia', ' Thailand', ' Ecuador', ' Laos',
       ' Taiwan', ' Haiti', ' Portugal', ' Dominican-Republic',
       ' El-Salvador', ' France', ' Guatemala', ' China', ' Japan',
       ' Yugoslavia', ' Peru', ' Outlying-US(Guam-USVI-etc)', ' Scotland',
       ' Trinadad&Tobago', ' Greece', ' Nicaragua', ' Vietnam', ' Hong',
       ' Ireland', ' Hungary', ' Holand-Netherlands'], dtype=object)
```

[30]:
```
df['target'].unique()
```

[30]:
```
array([' <=50K', ' >50K'], dtype=object)
```

[19]:
```
from sklearn import preprocessing
```

[20]:
```
le = preprocessing.LabelEncoder()
```

[32]:
```
df['workclass']= le.fit_transform(df['workclass'])
df['education']= le.fit_transform(df['education'])
df['marital-status']= le.fit_transform(df['marital-status'])
df['occupation']= le.fit_transform(df['occupation'])
df['relationship']= le.fit_transform(df['relationship'])
df['sex']= le.fit_transform(df['sex'])
df['race']= le.fit_transform(df['race'])
df['native-country']= le.fit_transform(df['native-country'])
df['target']= le.fit_transform(df['target'])




df.head()
```

[32]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	target
0	39	7	77516	9	13	4	1	1	4	1	2174	0	40	39	0
1	50	6	83311	9	13	2	4	0	4	1	0	0			
2	38	4	215646	11	9	0	6	1	4	1	0	0			
3	53	4	234721	1	7	2	6	0	2	1	0	0			
4	28	4	338409	9	13	2	10	5	2	0	0	0			

```
1              13              39       0
2              40              39       0
3              40              39       0
4              40               5       0
```

[33]:
```
df['target'].unique()
```

[33]:
```
array([0, 1])
```

[36]:
```
df.describe().T
```

[36]:

	count	mean	std	min	25%	\
age	32561.0	38.581647	13.640433	17.0	28.0	
workclass	32561.0	3.868892	1.455960	0.0	4.0	
fnlwgt	32561.0	189778.366512	105549.977697	12285.0	117827.0	
education	32561.0	10.298210	3.870264	0.0	9.0	
education-num	32561.0	10.080679	2.572720	1.0	9.0	
marital-status	32561.0	2.611836	1.506222	0.0	2.0	
occupation	32561.0	6.572740	4.228857	0.0	3.0	
relationship	32561.0	1.446362	1.606771	0.0	0.0	
race	32561.0	3.665858	0.848806	0.0	4.0	
sex	32561.0	0.669205	0.470506	0.0	0.0	
capital-gain	32561.0	1077.648844	7385.292085	0.0	0.0	
capital-loss	32561.0	87.303830	402.960219	0.0	0.0	
hours-per-week	32561.0	40.437456	12.347429	1.0	40.0	
native-country	32561.0	36.718866	7.823782	0.0	39.0	
target	32561.0	0.240810	0.427581	0.0	0.0	

	50%	75%	max
age	37.0	48.0	90.0
workclass	4.0	4.0	8.0
fnlwgt	178356.0	237051.0	1484705.0
education	11.0	12.0	15.0
education-num	10.0	12.0	16.0
marital-status	2.0	4.0	6.0
occupation	7.0	10.0	14.0
relationship	1.0	3.0	5.0
race	4.0	4.0	4.0
sex	1.0	1.0	1.0
capital-gain	0.0	0.0	99999.0
capital-loss	0.0	0.0	4356.0
hours-per-week	40.0	45.0	99.0
native-country	39.0	39.0	41.0
target	0.0	0.0	1.0

[37]:
```
df.cov()
```

[37]:

	age	workclass	fnlwgt	education
age	186.061400	0.075217	-1.103507e+05	-0.554754
workclass	0.075217	2.119819	-2.559577e+03	0.132496
fnlwgt	-110350.685300	-2559.577407	1.114080e+10	-11497.242194
education	-0.554754	0.132496	-1.149724e+04	14.978943
education-num	1.281849	0.195099	-1.172953e+04	3.576124
marital-status	-5.471025	-0.141955	4.475830e+03	-0.223892
occupation	-1.208296	1.569380	7.126630e+02	-0.347952
relationship	-5.779473	-0.211624	1.514673e+03	-0.067634
race	0.332504	0.061473	-1.907459e+03	0.046423
sex	0.570114	0.065751	1.333823e+03	-0.049815
capital-gain	7824.818537	363.818056	3.366625e+05	858.802295
capital-loss	317.560742	7.166771	-4.360303e+05	26.116794
hours-per-week	11.580130	2.498173	-2.446043e+04	2.652712
native-country	-0.122838	-0.087597	-4.291333e+04	1.946646
target	1.364997	0.032126	-4.270567e+02	0.131257

	education-num	marital-status	occupation	relationship
age	1.281849	-5.471025	-1.208296	-5.779473
workclass	0.195099	-0.141955	1.569380	-0.211624
fnlwgt	-11729.527298	4475.830232	712.662958	1514.673191
education	3.576124	-0.223892	-0.347952	-0.067634
education-num	6.618890	-0.268559	1.193471	-0.389207
marital-status	-0.268559	2.268704	-0.061491	0.448820
occupation	1.193471	-0.061491	17.883230	-0.513735
relationship	-0.389207	0.448820	-0.513735	2.581713
race	0.069527	-0.086954	0.024276	-0.158279
sex	0.014865	-0.091643	0.159766	-0.440333
capital-gain	2330.007877	-482.704062	796.540747	-687.299579
capital-loss	82.856445	-20.749935	30.651071	-39.535622
hours-per-week	4.705338	-3.543274	4.197263	-4.939527
native-country	1.023327	-0.280690	-0.414992	-0.069226
target	0.368685	-0.128360	0.136460	-0.172387

	race	sex	capital-gain	capital-loss
age	0.332504	0.570114	7.824819e+03	317.560742
workclass	0.061473	0.065751	3.638181e+02	7.166771
fnlwgt	-1907.458702	1333.822718	3.366625e+05	-436030.333167
education	0.046423	-0.049815	8.588023e+02	26.116794
education-num	0.069527	0.014865	2.330008e+03	82.856445
marital-status	-0.086954	-0.091643	-4.827041e+02	-20.749935
occupation	0.024276	0.159766	7.965407e+02	30.651071
relationship	-0.158279	-0.440333	-6.872996e+02	-39.535622
race	0.720471	0.034827	6.986316e+01	6.464118
sex	0.034827	0.221376	1.684584e+02	8.639360
capital-gain	69.863158	168.458406	5.454254e+07	-94085.760688
capital-loss	6.464118	8.639360	-9.408576e+04	162376.937814

```
hours-per-week     0.439236     1.332182  7.150032e+03     269.953755
native-country     0.915456    -0.029886 -1.145330e+02       1.319510
target             0.026075     0.043451  7.052309e+02      25.935432

                hours-per-week  native-country      target
age                  11.580130       -0.122838    1.364997
workclass             2.498173       -0.087597    0.032126
fnlwgt            -24460.426185   -42913.330957 -427.056721
education             2.652712        1.946646    0.131257
education-num         4.705338        1.023327    0.368685
marital-status       -3.543274       -0.280690   -0.128360
occupation            4.197263       -0.414992    0.136460
relationship         -4.939527       -0.069226   -0.172387
race                  0.439236        0.915456    0.026075
sex                   1.332182       -0.029886    0.043451
capital-gain       7150.032029     -114.533031  705.230910
capital-loss        269.953755        1.319510   25.935432
hours-per-week      152.458995       -0.258032    1.212651
native-country       -0.258032       61.211563    0.052991
target                1.212651        0.052991    0.182826
```

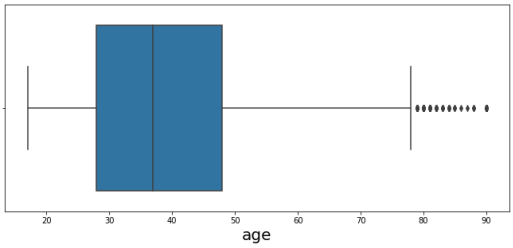
[38]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

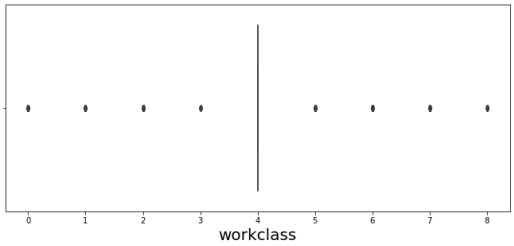
[43]:
```
plt.figure(figsize=(25,45),facecolor='white')
n=1
for col in df:
    if n<15:
        ax=plt.subplot(8,2,n)
        sns.boxplot(df[col])
        plt.xlabel(col,fontsize=20)
    n+=1
plt.show()
```

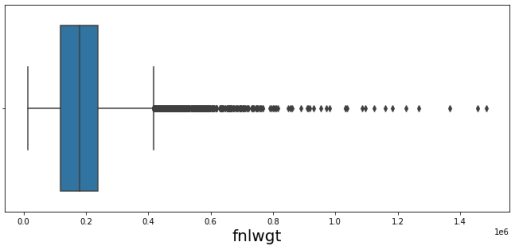
```
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
```

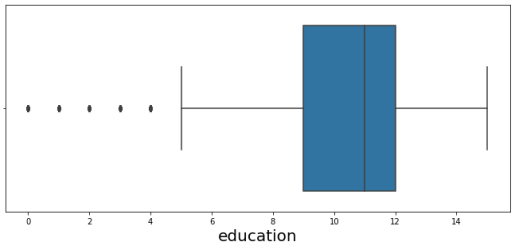
```
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
```

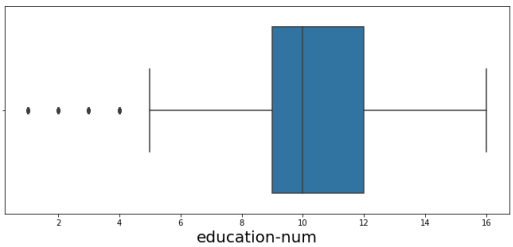
```
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
```

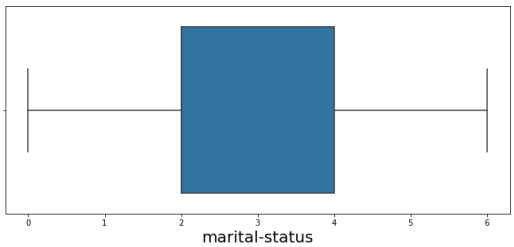

20 30 40 50 60 70 80 90
age

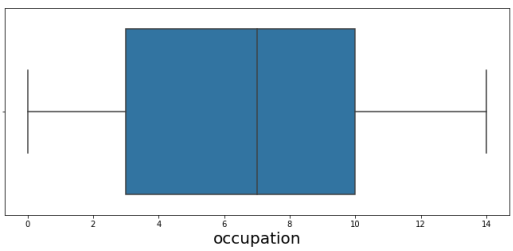

0 1 2 3 4 5 6 7 8
workclass

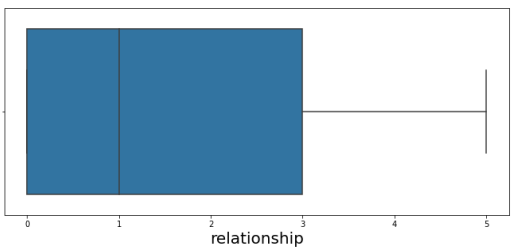

0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4
1e6
fnlwgt

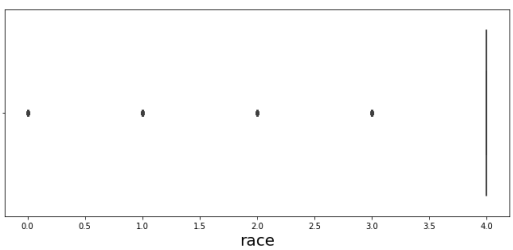

0 2 4 6 8 10 12 14
education

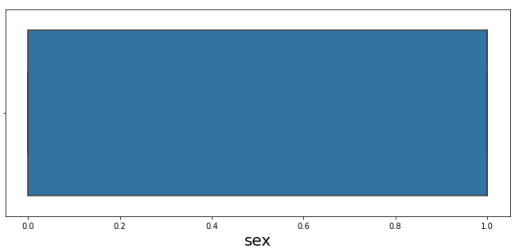

2 4 6 8 10 12 14 16
education-num

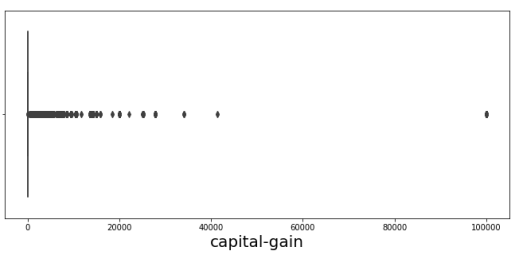

0 1 2 3 4 5 6
marital-status

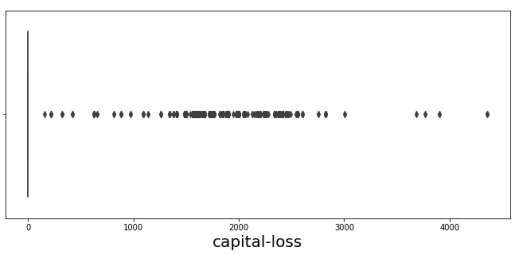

0 2 4 6 8 10 12 14
occupation

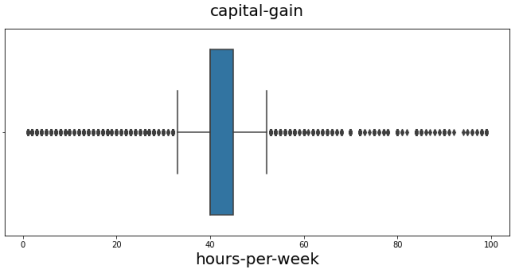

0 1 2 3 4 5
relationship

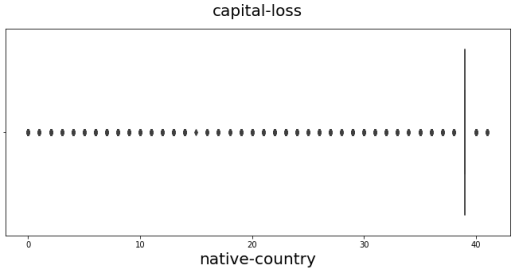

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
race


0.0 0.2 0.4 0.6 0.8 1.0
sex


0 20000 40000 60000 80000 100000
capital-gain


0 1000 2000 3000 4000
capital-loss


0 20 40 60 80 100
hours-per-week


0 10 20 30 40
native-country

[ ]:
```
age,fnlwgt,edu-num,hours-per week
```

[45]:
```
df2=[]
df2=df[['age','fnlwgt','education-num','hours-per-week']]
df2.head()
```

[45]:

	age	fnlwgt	education-num	hours-per-week
0	39	77516	13	40
1	50	83311	13	13
2	38	215646	9	40
3	53	234721	7	40
4	28	338409	13	40

[49]:
```
def outliers_imputation(df2,col):
    IQR=df2[col].quantile(0.75)-df2[col].quantile(0.25)
    lower_fence=df2[col].quantile(0.25)-(IQR*1.5)
    upper_fence=df2[col].quantile(0.75)+(IQR*1.5)
    df2.loc[df2[col]<=lower_fence,col]=lower_fence
    df2.loc[df2[col]>=upper_fence,col]=upper_fence
```

[50]:
```
col=df2.columns
```

[51]:
```
for j in col:
    outliers_imputation(df2,j)
```

```
/home/arijit/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py:1732:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
/home/arijit/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py:723:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value, self.name)
/home/arijit/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py:1817:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
```

[52]:
```
df.drop(['age','fnlwgt','education-num','hours-per-week'], axis=1)
```

[52]:

	workclass	education	marital-status	occupation	relationship	race	\
0	7	9	4	1	1	4	
1	6	9	2	4	0	4	
2	4	11	0	6	1	4	
3	4	1	2	6	0	2	
4	4	9	2	10	5	2	
…	…	…	…	…	…	…	
32556	4	7	2	13	5	4	
32557	4	11	2	7	0	4	
32558	4	11	6	1	4	4	
32559	4	11	4	1	3	4	
32560	5	11	2	4	5	4	

	sex	capital-gain	capital-loss	native-country	target
0	1	2174	0	39	0
1	1	0	0	39	0
2	1	0	0	39	0
3	1	0	0	39	0
4	0	0	0	5	0
…	…	…	…	…	…
32556	0	0	0	39	0
32557	1	0	0	39	1
32558	0	0	0	39	0
32559	1	0	0	39	0
32560	0	15024	0	39	1

[32561 rows x 11 columns]

[54]:
```
df_final = pd.concat([df, df2], axis=1)
df_final
```

[54]:

	age	workclass	fnlwgt	education	education-num	marital-status	\
0	39	7	77516	9	13	4	
1	50	6	83311	9	13	2	
2	38	4	215646	11	9	0	
3	53	4	234721	1	7	2	
4	28	4	338409	9	13	2	
…	…	…	…	…	…	…	
32556	27	4	257302	7	12	2	
32557	40	4	154374	11	9	2	
32558	58	4	151910	11	9	6	
32559	22	4	201490	11	9	4	

```
32560   52          5  287927         11              9               2

       occupation  relationship  race  sex  capital-gain  capital-loss  \
0               1             1     4    1          2174             0
1               4             0     4    1             0             0
2               6             1     4    1             0             0
3               6             0     2    1             0             0
4              10             5     2    0             0             0
…             …            …    …  …           …            …
32556          13             5     4    0             0             0
32557           7             0     4    1             0             0
32558           1             4     4    0             0             0
32559           1             3     4    1             0             0
32560           4             5     4    0         15024             0

       hours-per-week  native-country  target  age  fnlwgt  education-num  \
0                  40              39       0   39   77516           13.0
1                  13              39       0   50   83311           13.0
2                  40              39       0   38  215646            9.0
3                  40              39       0   53  234721            7.0
4                  40               5       0   28  338409           13.0
…                 …             …      …  …      …            …
32556              38              39       0   27  257302           12.0
32557              40              39       1   40  154374            9.0
32558              40              39       0   58  151910            9.0
32559              20              39       0   22  201490            9.0
32560              40              39       1   52  287927            9.0

       hours-per-week
0                40.0
1                32.5
2                40.0
3                40.0
4                40.0
…               …
32556            38.0
32557            40.0
32558            40.0
32559            32.5
32560            40.0

[32561 rows x 19 columns]
```

[55]:
```
import pymongo
import dns
```

[56]:
```
client = pymongo.MongoClient("mongodb+srv://bob:parabola@bob.wi8vkjq.mongodb.
↪net/?retryWrites=true&w=majority")
```

[57]:
```
database = client['Classification']
collection = database['census']
```

[58]:
```
data_dict = df_final.to_dict("records")
collection.insert_many(data_dict)
```

```
/home/arijit/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
UserWarning: DataFrame columns are not unique, some columns will be omitted.
  """Entry point for launching an IPython kernel.
```

[58]:
```
<pymongo.results.InsertManyResult at 0x7f5aae8d6d20>
```

[59]:
```
# Reading from Mongo Db
data_db = pd.DataFrame(list(collection.find()))
```

[61]:
```
data_db.columns
```

[61]:
```
Index(['_id', 'age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'target'],
      dtype='object')
```

[63]:
```
X_final=data_db[['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country']]
X_final
```

[63]:

	age	workclass	fnlwgt	education	education-num	marital-status	\
0	39	7	77516	9	13.0	4	
1	50	6	83311	9	13.0	2	
2	38	4	215646	11	9.0	0	
3	53	4	234721	1	7.0	2	
4	28	4	338409	9	13.0	2	
…	…	…	…	…	…	…	
32556	27	4	257302	7	12.0	2	
32557	40	4	154374	11	9.0	2	
32558	58	4	151910	11	9.0	6	
32559	22	4	201490	11	9.0	4	
32560	52	5	287927	11	9.0	2	

	occupation	relationship	race	sex	capital-gain	capital-loss	\
0	1	1	4	1	2174	0	
1	4	0	4	1	0	0	

```
2               6             1     4    1             0             0
3               6             0     2    1             0             0
4              10             5     2    0             0             0
…             …           …   …   …          …            …
32556          13             5     4    0             0             0
32557           7             0     4    1             0             0
32558           1             4     4    0             0             0
32559           1             3     4    1             0             0
32560           4             5     4    0         15024             0

       hours-per-week  native-country
0                40.0              39
1                32.5              39
2                40.0              39
3                40.0              39
4                40.0               5
…                …                 …
32556            38.0              39
32557            40.0              39
32558            40.0              39
32559            32.5              39
32560            40.0              39

[32561 rows x 14 columns]
```

[65]:
```
y=data_db['target']
y
```

[65]:
```
0        0
1        0
2        0
3        0
4        0
        ..
32556    0
32557    1
32558    0
32559    0
32560    1
Name: target, Length: 32561, dtype: int64
```

[99]:
```
# Creating Model

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

```
from sklearn.metrics import␣
 ↪accuracy_score,classification_report,confusion_matrix
```

[67]:
```
scaler=StandardScaler()
```

[68]:
```
X_train, X_test, y_train,y_test = train_test_split(X_final,y␣
 ↪,random_state=42,train_size=0.25)
```

[69]:
```
X_train_tf=scaler.fit_transform(X_train)
```

[70]:
```
X_test_tf=scaler.transform(X_test)
```

Logistic Regression

[74]:
```
from sklearn.linear_model import LogisticRegression
log_classifier=LogisticRegression()
log_classifier.fit(X_train, y_train)
ytrain_pred = log_classifier.predict_proba(X_train)
print('Logistic train roc-auc: {}'.format(roc_auc_score(y_train, ytrain_pred[:
 ↪,1])))
ytest_pred = log_classifier.predict_proba(X_test)
print('Logistic test roc-auc: {}'.format(roc_auc_score(y_test, ytest_pred[:
 ↪,1])))
```

```
Logistic train roc-auc: 0.7339925307453228
Logistic test roc-auc: 0.7276256917380297
```

```
/home/arijit/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

selecting the best threshold for maximum accuracy

[77]:
```
pred=[]
for model in [log_classifier]:
    pred.append(pd.Series(model.predict_proba(X_test)[:,1]))
final_prediction=pd.concat(pred,axis=1).mean(axis=1)
print('Ensemble test roc-auc: {}'.
 ↪format(roc_auc_score(y_test,final_prediction)))
```

```
Ensemble test roc-auc: 0.7276256917380297
```

[78]:
```
pd.concat(pred,axis=1)
```

[78]:
```
              0
0      0.147175
1      0.235967
2      0.158981
3      0.157747
4      0.233049
…          …
24416  0.145238
24417  0.114512
24418  0.391119
24419  0.262646
24420  0.101170

[24421 rows x 1 columns]
```

[79]:
```
final_prediction
```

[79]:
```
0        0.147175
1        0.235967
2        0.158981
3        0.157747
4        0.233049
           …
24416    0.145238
24417    0.114512
24418    0.391119
24419    0.262646
24420    0.101170
Length: 24421, dtype: float64
```

[80]:
```
#### Calculate the ROC Curve


fpr, tpr, thresholds = roc_curve(y_test, final_prediction)
thresholds
```

[80]:
```
array([2.        , 1.        , 1.        , …, 0.06244032, 0.0621363 ,
       0.05963384])
```

[82]:
```
import numpy as np
from sklearn.metrics import accuracy_score
accuracy_ls = []
for thres in thresholds:
    y_pred = np.where(final_prediction>thres,1,0)
    accuracy_ls.append(accuracy_score(y_test, y_pred, normalize=True))
```

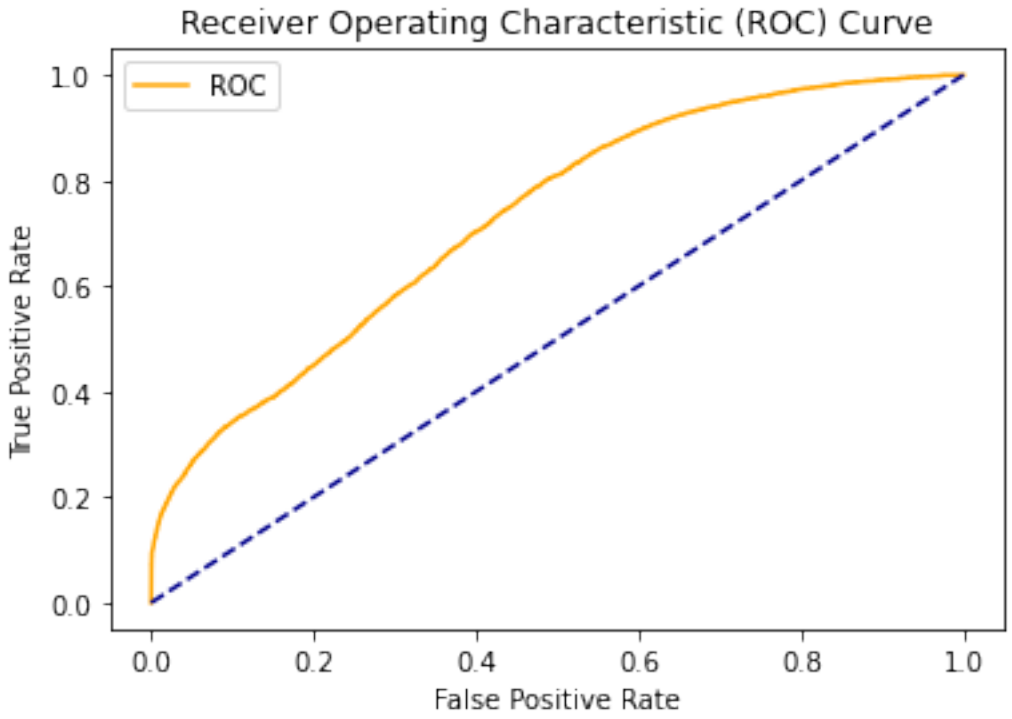
```
accuracy_ls = pd.concat([pd.Series(thresholds), pd.Series(accuracy_ls)],
                        axis=1)
accuracy_ls.columns = ['thresholds', 'accuracy']
accuracy_ls.sort_values(by='accuracy', ascending=False, inplace=True)
accuracy_ls.head()
```

[82]:

	thresholds	accuracy
560	0.605190	0.791163
563	0.604760	0.791122
564	0.604614	0.791122
561	0.604948	0.791122
502	0.622077	0.791081

[83]:
```
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```

[84]:
```
plot_roc_curve(fpr,tpr)
```



Accuarcy Score

[94]:
```
y_pred=log_classifier.predict(X_test)
score=accuracy_score(y_pred,y_test)
score
```

[94]:
```
0.7857581589615494
```

Classification Report

[96]:
```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.80      0.87     21982
           1       0.26      0.63      0.37      2439

    accuracy                           0.79     24421
   macro avg       0.61      0.72      0.62     24421
weighted avg       0.88      0.79      0.82     24421
```

[102]:
```
conf_mat=confusion_matrix(y_pred,y_test)
```

[103]:
```
conf_mat
```

[103]:
```
array([[17653,  4329],
       [  903,  1536]])
```

[ ]:
```

```

Regression_Assignment

November 4, 2022

Individual household electric power consumption Data Set

Data Set Information:

This archive contains 2075259 measurements gathered in a house located in Sceaux (7km of Paris, France) between December 2006 and November 2010 (47 months). Notes: 1.(global_active_power*1000/60 - sub_metering_1 - sub_metering_2 - sub_metering_3) represents the active energy consumed every minute (in watt hour) in the household by electrical equipment not measured in sub-meterings 1, 2 and 3. 2.The dataset contains some missing values in the measurements (nearly 1,25% of the rows). All calendar timestamps are present in the dataset but for some timestamps, the measurement values are missing: a missing value is represented by the absence of value between two consecutive semi-colon attribute separators. For instance, the dataset shows missing values on April 28, 2007.

Attribute Information:

1.date: Date in format dd/mm/yyyy 2.time: time in format hh:mm:ss 3.global_active_power: household global minute-averaged active power (in kilowatt) 4.global_reactive_power: household global minute-averaged reactive power (in kilowatt) 5.voltage: minute-averaged voltage (in volt) 6.global_intensity: household global minute-averaged current intensity (in ampere) 7.sub_metering_1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered). 8.sub_metering_2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light. 9.sub_metering_3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

[6]:
```
import pandas as pd
```

[7]:
```
df=pd.read_csv("household_power_consumption.txt",delimiter=";")
```

```
/home/arijit/anaconda3/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3331: DtypeWarning: Columns
(2,3,4,5,6,7) have mixed types.Specify dtype option on import or set
low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

[8]:
```
df.head()
```

[8]:
```
         Date      Time Global_active_power Global_reactive_power  Voltage  \
0  16/12/2006  17:24:00               4.216                 0.418  234.840
1  16/12/2006  17:25:00               5.360                 0.436  233.630
2  16/12/2006  17:26:00               5.374                 0.498  233.290
3  16/12/2006  17:27:00               5.388                 0.502  233.740
4  16/12/2006  17:28:00               3.666                 0.528  235.680

  Global_intensity Sub_metering_1 Sub_metering_2  Sub_metering_3
0           18.400          0.000          1.000            17.0
1           23.000          0.000          1.000            16.0
2           23.000          0.000          2.000            17.0
3           23.000          0.000          1.000            17.0
4           15.800          0.000          1.000            17.0
```

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075259 entries, 0 to 2075258
Data columns (total 9 columns):
 #   Column                 Dtype
---  ------                 -----
 0   Date                   object
 1   Time                   object
 2   Global_active_power    object
 3   Global_reactive_power  object
 4   Voltage                object
 5   Global_intensity       object
 6   Sub_metering_1         object
 7   Sub_metering_2         object
 8   Sub_metering_3         float64
dtypes: float64(1), object(8)
memory usage: 142.5+ MB
```

[10]:
```
df1=df.dropna(axis = 0)
```

[11]:
```
df1.isnull().sum()
```

[11]:
```
Date                   0
Time                   0
Global_active_power    0
Global_reactive_power  0
Voltage                0
Global_intensity       0
Sub_metering_1         0
Sub_metering_2         0
Sub_metering_3         0
dtype: int64
```

[12]:
```
df3=df1.sample(50000)
```

[13]:
```
df3.head()
```

[13]:

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
1302220	8/6/2009	01:04:00	0.340	0.102	242.360	1.600	0.000	1.000	1.0
665232	22/3/2008	16:36:00	1.728	0.000	241.120	7.000	1.000	0.000	18.0
2032843	28/10/2010	10:07:00	1.55	0.218	245.54	6.2	0.0	2.0	19.0
1058014	20/12/2008	10:58:00	1.756	0.000	241.270	7.200	0.000	0.000	18.0
20769	31/12/2006	03:33:00	0.220	0.000	246.490	1.000	0.000	0.000	0.0

[14]:
```
df3['Sub_metering_1'] = df3['Sub_metering_1'].astype('float')
```

[15]:
```
df3['Sub_metering_2'] = df3['Sub_metering_2'].astype('float')
```

[16]:
```
df3['Sub_metering_3'] = df3['Sub_metering_3'].astype('float')
```

[17]:
```
df3['Total_metering'] =␣
↪df3['Sub_metering_1']+df3['Sub_metering_2']+df3['Sub_metering_3']
```

[18]:
```
df3.head()
```

[18]:

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2
1302220	8/6/2009	01:04:00	0.340	0.102	242.360	1.600	0.0	1.0
665232	22/3/2008	16:36:00	1.728	0.000	241.120	7.000	1.0	0.0
2032843	28/10/2010	10:07:00	1.55	0.218	245.54	6.2	0.0	2.0
1058014	20/12/2008	10:58:00	1.756	0.000				
20769	31/12/2006	03:33:00	0.220	0.000				

```
1058014  241.270            7.200              0.0              0.0
20769    246.490            1.000              0.0              0.0

         Sub_metering_3  Total_metering
1302220             1.0             2.0
665232             18.0            19.0
2032843            19.0            21.0
1058014            18.0            18.0
20769               0.0             0.0
```

[19]:
```
df_final=df3.
 ↪drop(columns=['Date','Time','Sub_metering_1','Sub_metering_2','Sub_metering_3'])
df_final.head()
```

[19]:
```
         Global_active_power Global_reactive_power  Voltage Global_intensity  \
1302220                0.340                 0.102  242.360             1.600
665232                 1.728                 0.000  241.120             7.000
2032843                 1.55                 0.218   245.54               6.2
1058014                1.756                 0.000  241.270             7.200
20769                  0.220                 0.000  246.490             1.000

         Total_metering
1302220             2.0
665232             19.0
2032843            21.0
1058014            18.0
20769               0.0
```

Creating a copy of the original Dataframe

[20]:
```
data=df_final.copy()
data.head()
```

[20]:
```
         Global_active_power Global_reactive_power  Voltage Global_intensity  \
1302220                0.340                 0.102  242.360             1.600
665232                 1.728                 0.000  241.120             7.000
2032843                 1.55                 0.218   245.54               6.2
1058014                1.756                 0.000  241.270             7.200
20769                  0.220                 0.000  246.490             1.000

         Total_metering
1302220             2.0
665232             19.0
2032843            21.0
1058014            18.0
20769               0.0
```

Statistical Analysis

[21]:
```
data['Global_active_power'] = data['Global_active_power'].astype('float')
data['Global_reactive_power'] = data['Global_reactive_power'].astype('float')
data['Voltage'] = data['Voltage'].astype('float')
data['Global_intensity'] = data['Global_intensity'].astype('float')
```

[22]:
```
data.describe().T
```

[22]:

	count	mean	std	min	25%	\
Global_active_power	50000.0	1.085569	1.055215	0.078	0.306	
Global_reactive_power	50000.0	0.123293	0.112357	0.000	0.048	
Voltage	50000.0	240.856064	3.219926	224.940	239.020	
Global_intensity	50000.0	4.601900	4.433737	0.200	1.400	
Total_metering	50000.0	8.866540	12.926165	0.000	0.000	

	50%	75%	max
Global_active_power	0.592	1.522	9.732
Global_reactive_power	0.100	0.194	1.034
Voltage	241.040	242.890	253.530
Global_intensity	2.600	6.400	43.000
Total_metering	1.000	18.000	127.000

[23]:
```
data.cov()
```

[23]:

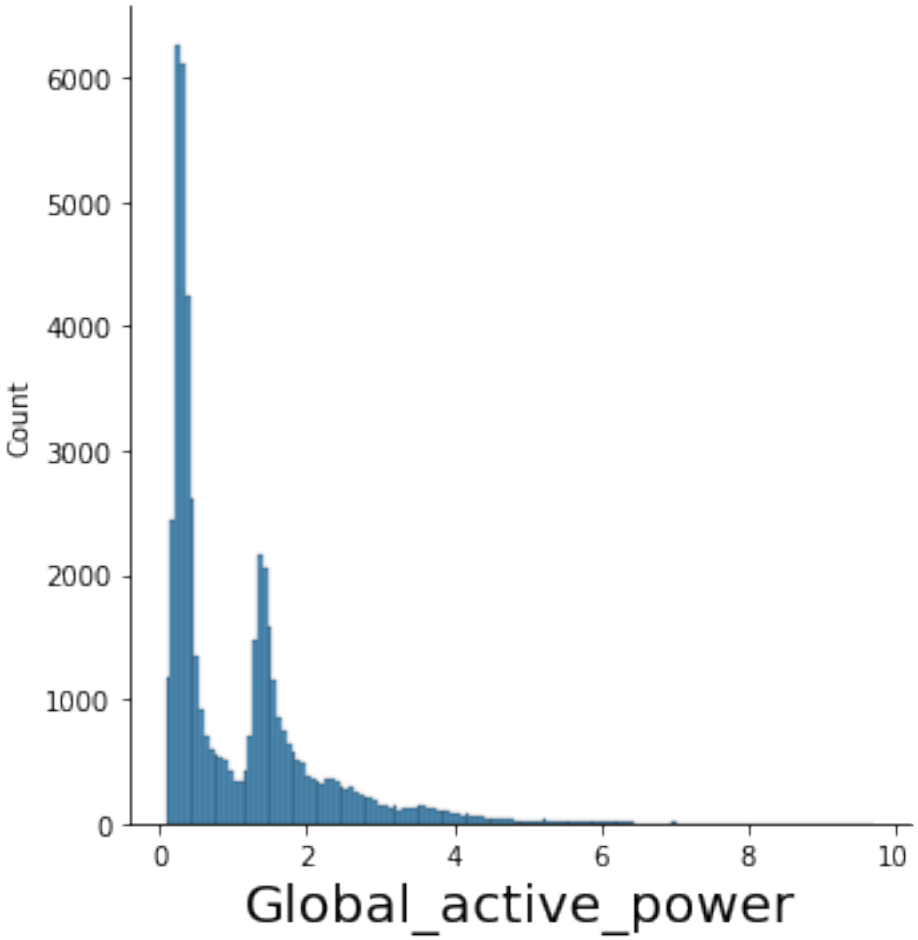
	Global_active_power	Global_reactive_power	Voltage	\
Global_active_power	1.113478	0.028964	-1.345832	
Global_reactive_power	0.028964	0.012624	-0.040899	
Voltage	-1.345832	-0.040899	10.367921	
Global_intensity	4.673418	0.131270	-5.817872	
Total_metering	11.611630	0.261175	-14.376093	

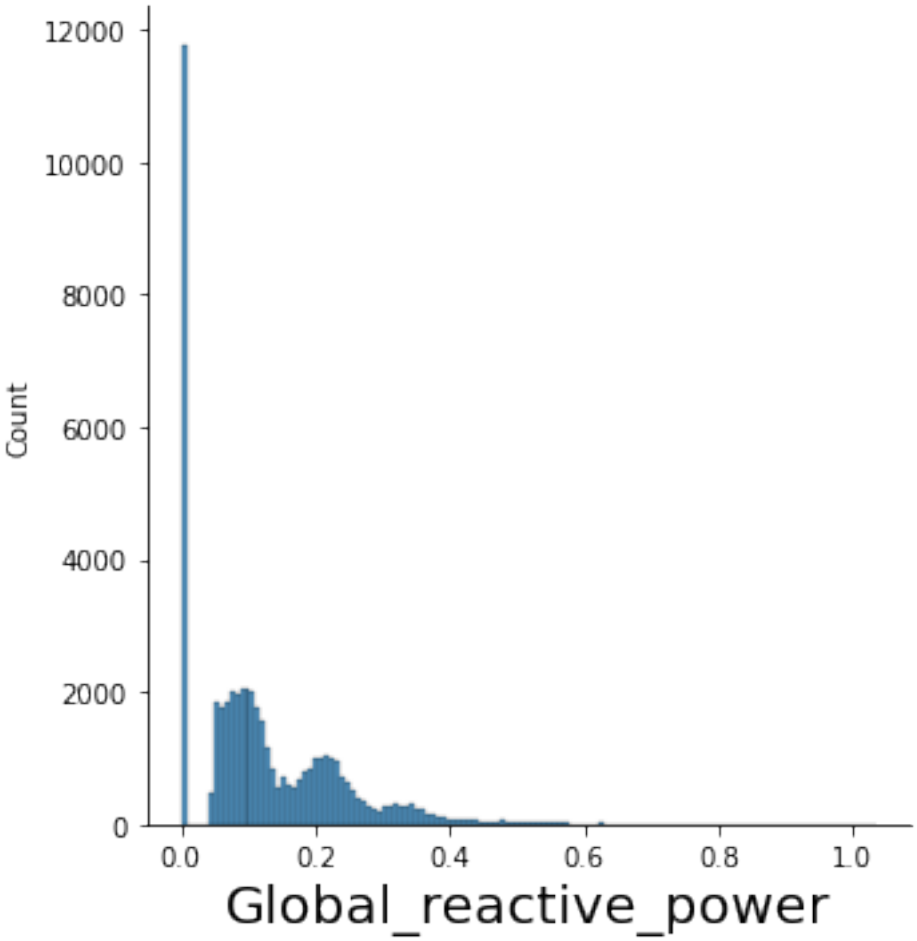
	Global_intensity	Total_metering
Global_active_power	4.673418	11.611630
Global_reactive_power	0.131270	0.261175
Voltage	-5.817872	-14.376093
Global_intensity	19.658027	48.634698
Total_metering	48.634698	167.085750

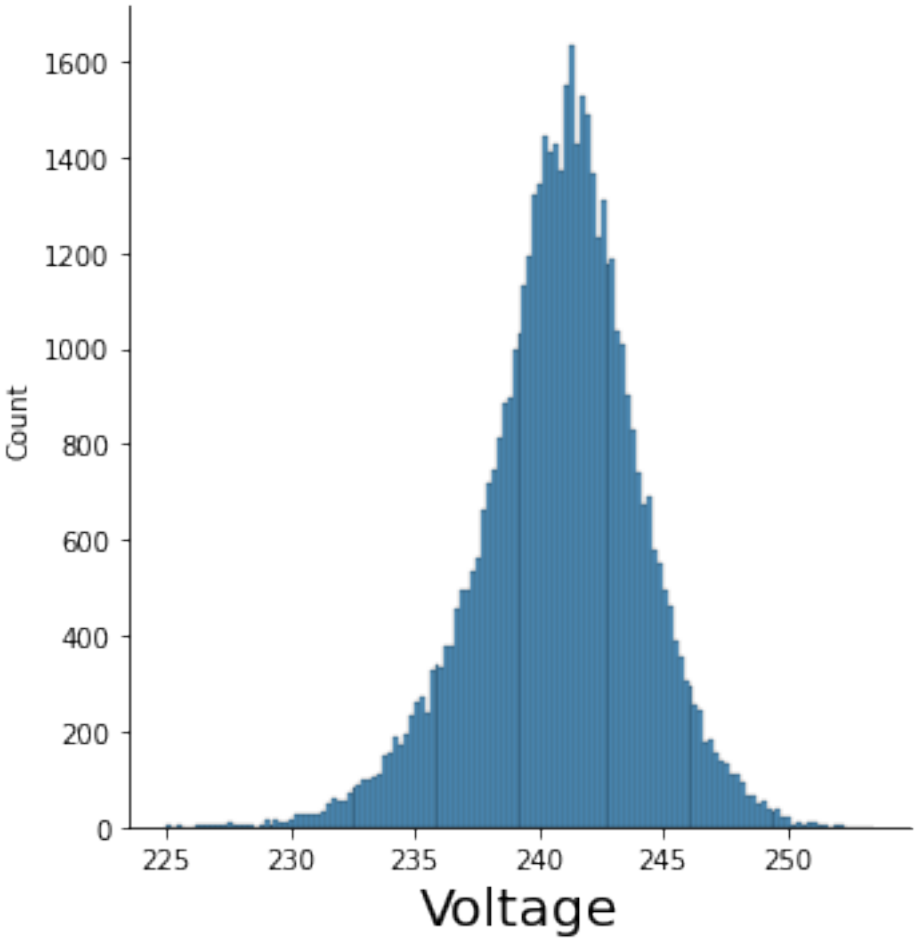
[24]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

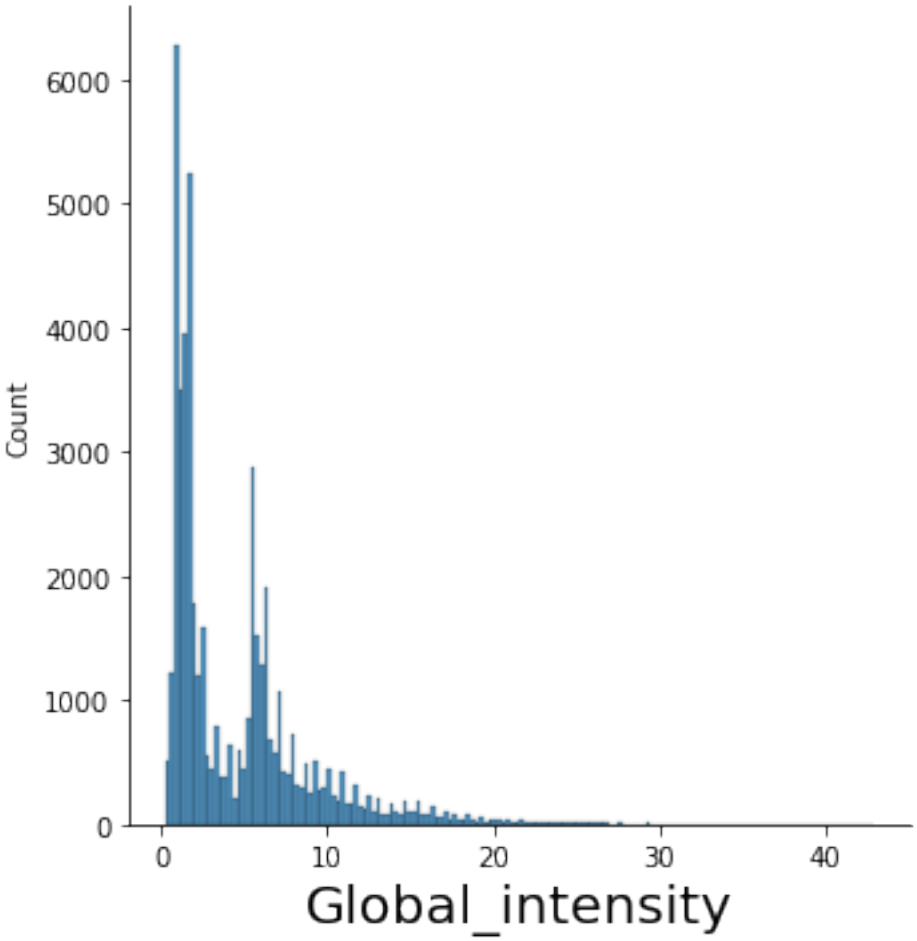
[25]:
```
plt.figure(figsize=(15,35),facecolor='red')

for col in data:
    sns.displot(data[col])
    plt.xlabel(col,fontsize=20)
plt.show()
```

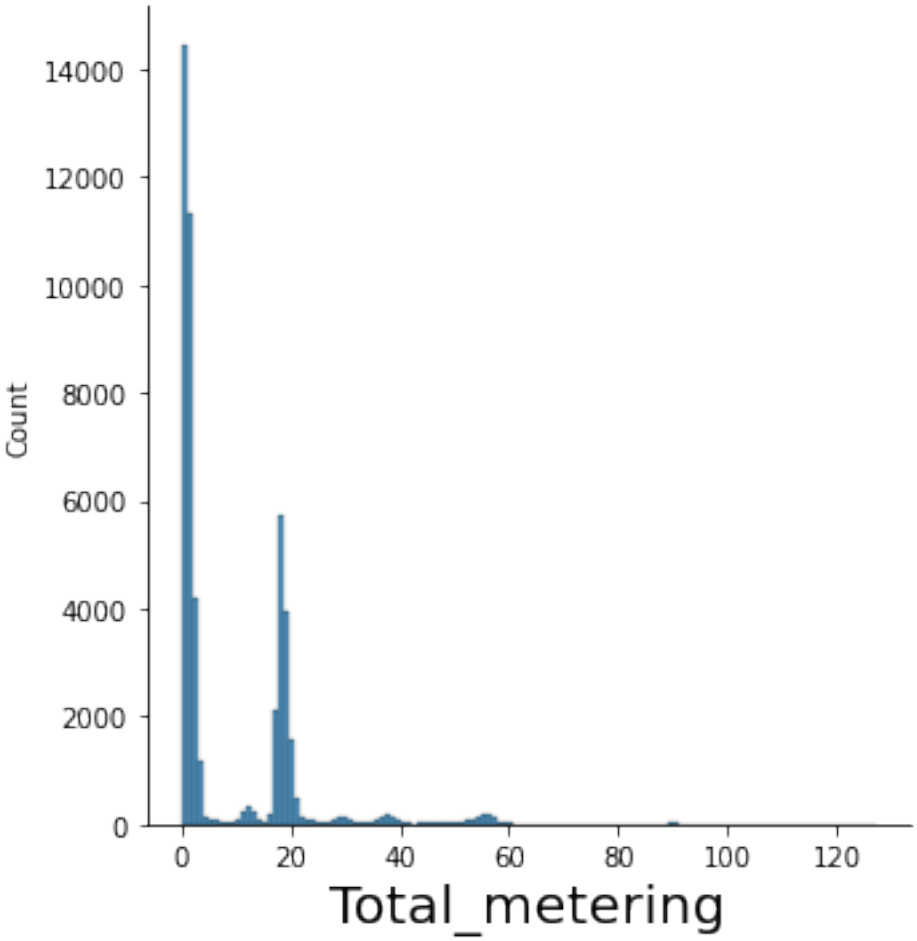
```
<Figure size 1080x2520 with 0 Axes>
```




12000
10000
8000
6000
4000
2000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Global_reactive_power


1600
1400
1200
1000
800
600
400
200
0
Count
225
230
235
240
245
250
Voltage

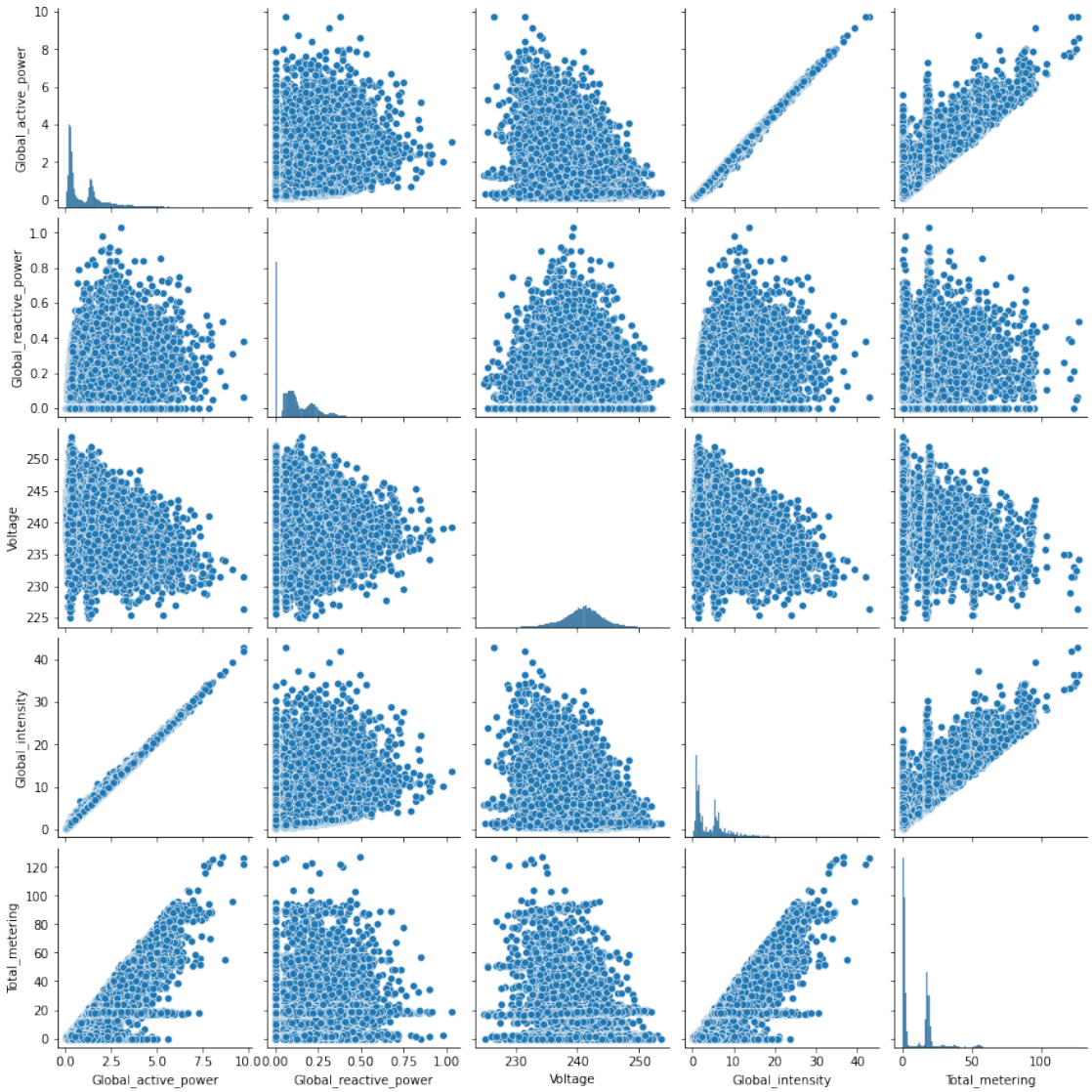

6000
5000
4000
3000
2000
1000
0
Count
0
10
20
30
40
Global_intensity



Multivariate Analysis

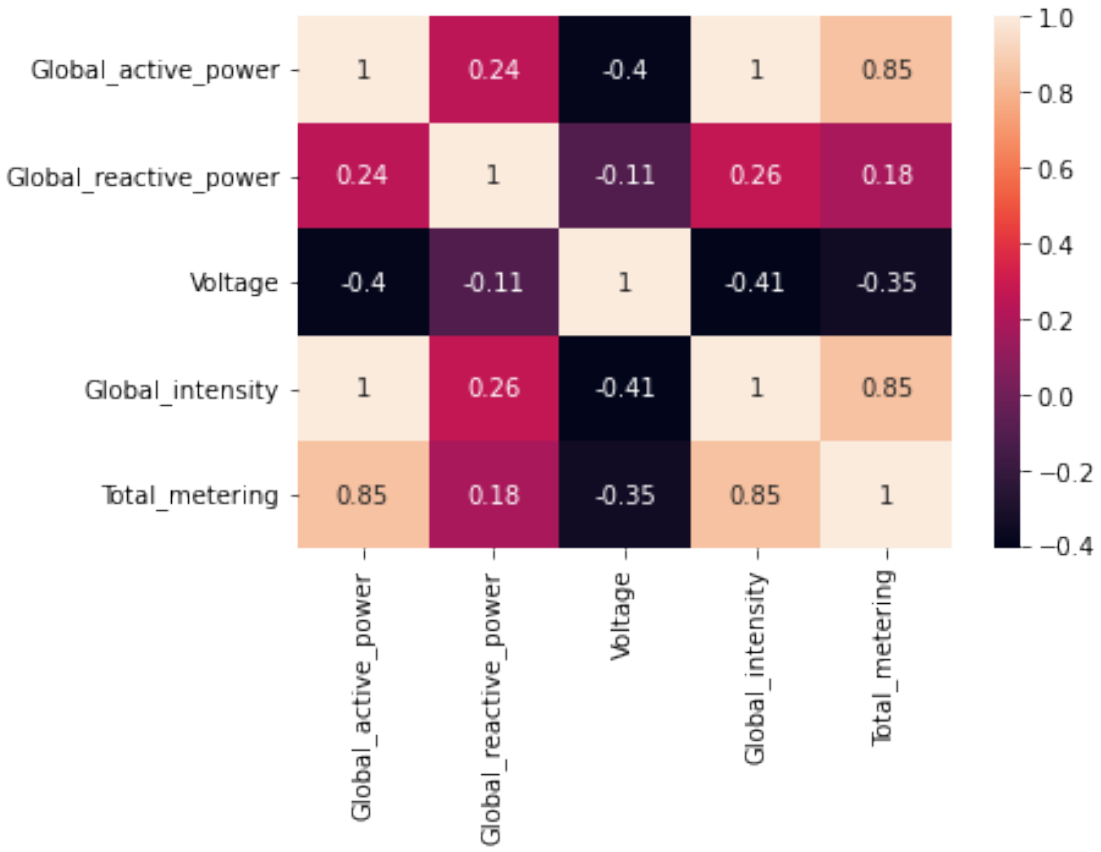
[26]:
```
sns.pairplot(data)
```

[26]:
```
<seaborn.axisgrid.PairGrid at 0x7fe297bee910>
```



[27]:
```
sns.heatmap(data.corr(),annot=True)
```

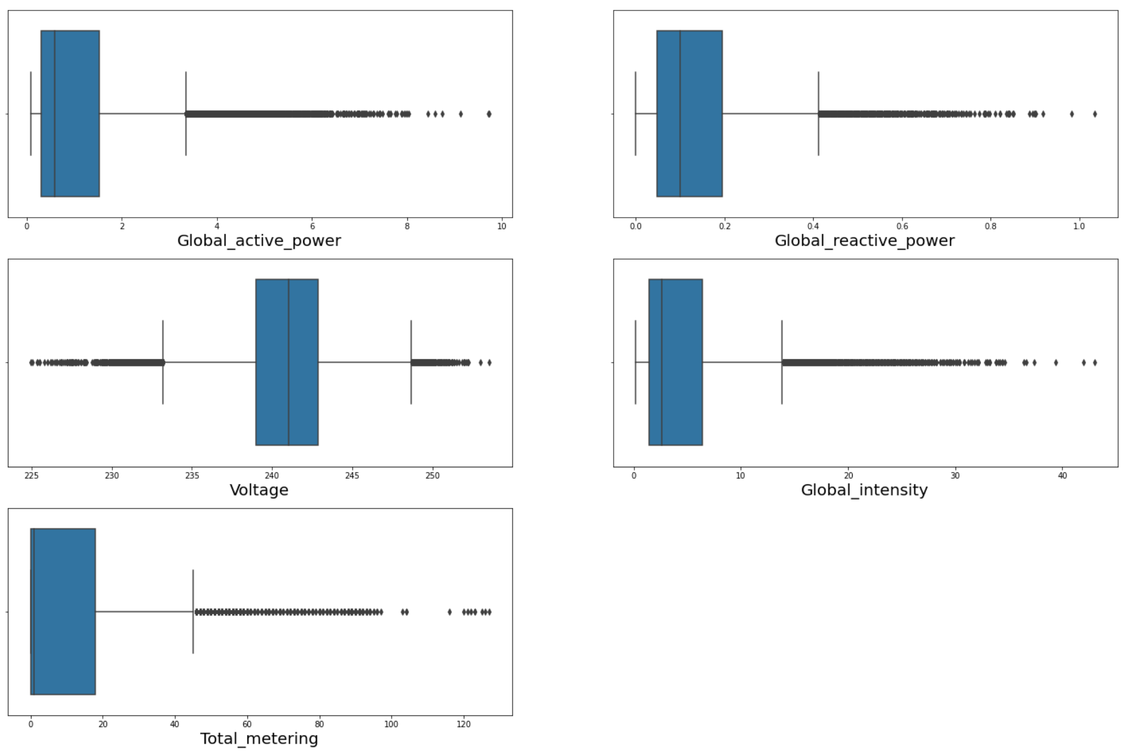
[27]: <AxesSubplot:>



[28]:
```
plt.figure(figsize=(25,45),facecolor='white')
n=1
for col in data:
    if n<10:
        ax=plt.subplot(8,2,n)
        sns.boxplot(data[col])
        plt.xlabel(col,fontsize=20)
    n+=1
plt.show()
```

```
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
```

```
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
```



[29]:
```
data_final=data[['Global_active_power','Global_reactive_power','Voltage','Global_intensity']]
data_final
```

[29]:
```
         Global_active_power  Global_reactive_power  Voltage  Global_intensity
1302220                0.340                  0.102   242.36               1.6
665232                 1.728                  0.000   241.12               7.0
2032843                1.550                  0.218   245.54               6.2
1058014                1.756                  0.000   241.27               7.2
20769                  0.220                  0.000   246.49               1.0
…                      …                      …       …                    …
1464627                3.372                  0.214   237.51              14.2
1710070                3.038                  0.314   239.91              13.0
236194                 0.394                  0.194   236.92               1.8
1570302                0.346                  0.166   245.41               1.6
2066830                0.510                  0.000   242.87               2.2

[50000 rows x 4 columns]
```

[30]:
```
def outliers_imputation(data_final,col):
    IQR=data_final[col].quantile(0.75)-data_final[col].quantile(0.25)
    lower_fence=data_final[col].quantile(0.25)-(IQR*1.5)
    upper_fence=data_final[col].quantile(0.75)+(IQR*1.5)
    data_final.loc[data_final[col]<=lower_fence,col]=lower_fence
    data_final.loc[data_final[col]>=upper_fence,col]=upper_fence
```

[31]:
```
col=data_final.columns
```

[32]:
```
for j in col:
    outliers_imputation(data_final,j)
```

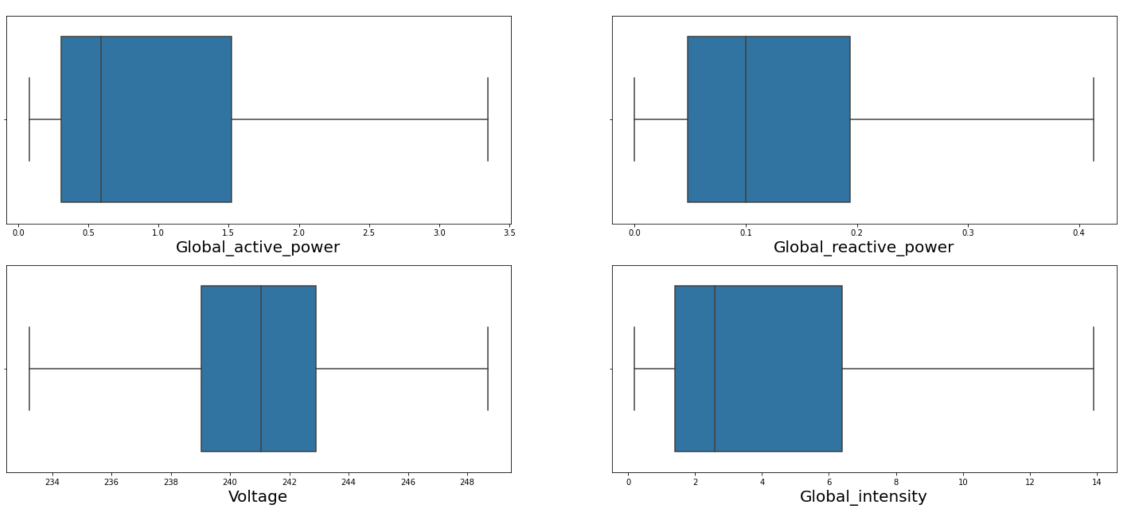
```
/home/arijit/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py:1732:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
/home/arijit/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py:723:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value, self.name)
```

Checking after Imputation

[33]:
```
plt.figure(figsize=(25,45),facecolor='white')
n=1
for col in data_final:
```

```
    if n<10:
        ax=plt.subplot(8,2,n)
        sns.boxplot(data_final[col])
        plt.xlabel(col,fontsize=20)
    n+=1
plt.show()
```

```
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
/home/arijit/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43:
FutureWarning: 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 an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
```



[37]:
```
X=data_final
X
```

[37]:

	Global_active_power	Global_reactive_power	Voltage	Global_intensity
1302220	0.340	0.102	242.36	1.6
665232	1.728	0.000	241.12	7.0
2032843	1.550	0.218	245.54	6.2
1058014	1.756	0.000	241.27	7.2
20769	0.220	0.000	246.49	1.0
…	…	…	…	…
1464627	3.346	0.214	237.51	13.9
1710070	3.038	0.314	239.91	13.0
236194	0.394	0.194	236.92	1.8
1570302	0.346	0.166	245.41	1.6
2066830	0.510	0.000	242.87	2.2

[50000 rows x 4 columns]

[36]:
```
y=data['Total_metering']
y
```

[36]:
```
1302220     2.0
665232     19.0
2032843    21.0
1058014    18.0
20769       0.0
           …
1464627    30.0
1710070    19.0
236194      0.0
1570302     2.0
2066830     1.0
Name: Total_metering, Length: 50000, dtype: float64
```

[38]:
```
pip install "pymongo[srv]"
```

```
Requirement already satisfied: pymongo[srv] in
/home/arijit/anaconda3/lib/python3.7/site-packages (3.12.3)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0; extra == "srv" in
/home/arijit/anaconda3/lib/python3.7/site-packages (from pymongo[srv]) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[39]:
```
pip install dnspython==1.16.0
```

```
Requirement already satisfied: dnspython==1.16.0 in
/home/arijit/anaconda3/lib/python3.7/site-packages (1.16.0)
```

```
Note: you may need to restart the kernel to use updated packages.
```

[40]:
```
# Uploading the data in MongoDB database
import pymongo
import dns
```

[59]:
```
client = pymongo.MongoClient("mongodb+srv://bob:parabola@bob.wi8vkjq.mongodb.
↪net/?retryWrites=true&w=majority")
```

[60]:
```
database = client['power_consumption']
collection = database['household_power_data']
```

[61]:
```
data_dict = X.to_dict("records")
collection.insert_many(data_dict)
```

[61]:
```
<pymongo.results.InsertManyResult at 0x7fe213b3dbe0>
```

[64]:
```
# Reading from Mongo Db
data_db = pd.DataFrame(list(collection.find()))
```

[65]:
```
data_db
```

[65]:

	_id	Global_active_power	Global_reactive_power	Voltage	Global_intensity
0	63637576c77d7402da66266c	0.340	0.102	242.36	1.6
1	63637576c77d7402da66266d	1.728	0.000	241.12	7.0
2	63637576c77d7402da66266e	1.550	0.218	245.54	6.2
3	63637576c77d7402da66266f	1.756	0.000	241.27	7.2
4	63637576c77d7402da662670	0.220	0.000	246.49	1.0
…	…	…	…	…	…
49995	63637578c77d7402da66e9b7	3.346	0.214	237.51	13.9
49996	63637578c77d7402da66e9b8	3.038	0.314	239.91	13.0
49997	63637578c77d7402da66e9b9	0.394	0.194	236.92	1.8
49998	63637578c77d7402da66e9ba	0.346	0.166	245.41	1.6
49999	63637578c77d7402da66e9bb	0.510	0.000	242.87	2.2

```
[50000 rows x 5 columns]
```

[75]:
```
X_final=data_db[['Global_active_power','Global_reactive_power','Voltage','Global_intensity']]
X_final
```

[75]:

	Global_active_power	Global_reactive_power	Voltage	Global_intensity
0	0.340	0.102	242.36	1.6
1	1.728	0.000	241.12	7.0
2	1.550	0.218	245.54	6.2
3	1.756	0.000	241.27	7.2
4	0.220	0.000	246.49	1.0
…	…	…	…	…
49995	3.346	0.214	237.51	13.9
49996	3.038	0.314	239.91	13.0
49997	0.394	0.194	236.92	1.8
49998	0.346	0.166	245.41	1.6
49999	0.510	0.000	242.87	2.2

```
[50000 rows x 4 columns]
```

[69]:
```
y
```

[69]:
```
1302220     2.0
665232     19.0
2032843    21.0
1058014    18.0
20769       0.0
           …
1464627    30.0
1710070    19.0
236194      0.0
1570302     2.0
2066830     1.0
Name: Total_metering, Length: 50000, dtype: float64
```

[72]:
```
# Creating Model

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

[73]:
```
scaler=StandardScaler()
```

[76]:
```
X_train, X_test, y_train,y_test = train_test_split(X_final,y␣
↪,random_state=42,train_size=0.25)
```

[77]:
```
X_train_tf=scaler.fit_transform(X_train)
```

[78]:
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.svm import SVR
```

[80]:
```
model1=LinearRegression()
model2=Ridge()
model3=Lasso()
model4=ElasticNet()
model5=SVR()
model1.fit(X_train_tf,y_train)
model2.fit(X_train_tf,y_train)
model3.fit(X_train_tf,y_train)
model4.fit(X_train_tf,y_train)
model5.fit(X_train_tf,y_train)
```

[80]:
```
SVR()
```

[81]:
```
model1.score(X_train_tf,y_train)
model2.score(X_train_tf,y_train)
model3.score(X_train_tf,y_train)
model4.score(X_train_tf,y_train)
model5.score(X_train_tf,y_train)
```

[81]:
```
0.6932961477211593
```

[82]:
```
X_test_tf=scaler.transform(X_test)
```

[83]:
```
y_test
```

[83]:
```
1546376     1.0
351341      0.0
1698122     1.0
255342      0.0
1772448     4.0
            …
1947105    31.0
252019      0.0
179411     17.0
621867      0.0
843532      0.0
Name: Total_metering, Length: 37500, dtype: float64
```

[85]:
```
y_predict1=model1.predict(X_test_tf)
y_predict2=model2.predict(X_test_tf)
y_predict3=model3.predict(X_test_tf)
y_predict4=model4.predict(X_test_tf)
y_predict5=model5.predict(X_test_tf)
```

[87]:
```
from sklearn.metrics import r2_score
score1=r2_score(y_test,y_predict1)
score2=r2_score(y_test,y_predict2)
score3=r2_score(y_test,y_predict3)
score4=r2_score(y_test,y_predict4)
score5=r2_score(y_test,y_predict5)
adj_r1=1 - (1-score1)*(X_final.shape[0]-1)/(X_final.shape[0]-X_final.shape[1]-1)
adj_r2=1 - (1-score2)*(X_final.shape[0]-1)/(X_final.shape[0]-X_final.shape[1]-1)
adj_r3=1 - (1-score3)*(X_final.shape[0]-1)/(X_final.shape[0]-X_final.shape[1]-1)
adj_r4=1 - (1-score4)*(X_final.shape[0]-1)/(X_final.shape[0]-X_final.shape[1]-1)
adj_r5=1 - (1-score5)*(X_final.shape[0]-1)/(X_final.shape[0]-X_final.shape[1]-1)
```

[93]:
```
models={'model':
 ↪['LinearRegression','Ridge','Lasso','ElasticNet','SVR'],'R_squared':
 ↪[score1,score2,score3,score4,score5]
       ,'Adjusted_R_squared':[adj_r1,adj_r2,adj_r3,adj_r4,adj_r5]}
```

[94]:
```
results = pd.DataFrame(models)
results
```

[94]:

	model	R_squared	Adjusted_R_squared
0	LinearRegression	0.660538	0.660511
1	Ridge	0.660602	0.660575
2	Lasso	0.651229	0.651201
3	ElasticNet	0.622063	0.622033
4	SVR	0.687512	0.687487

Hyperparameter Tunning for SVR

[98]:
```
params = { 'kernel' : ['linear','poly','sigmoid','rbf']}
from sklearn.model_selection import GridSearchCV
grid=GridSearchCV(estimator=model5,param_grid=params,cv=10,n_jobs=-1)
grid.fit(X_train_tf,y_train)
```

[98]:
```
GridSearchCV(cv=10, estimator=SVR(), n_jobs=-1,
             param_grid={'kernel': ['linear', 'poly', 'sigmoid', 'rbf']})
```

[102]:
```
grid.best_score_
grid.best_params_
```

[102]:
```
{'kernel': 'rbf'}
```