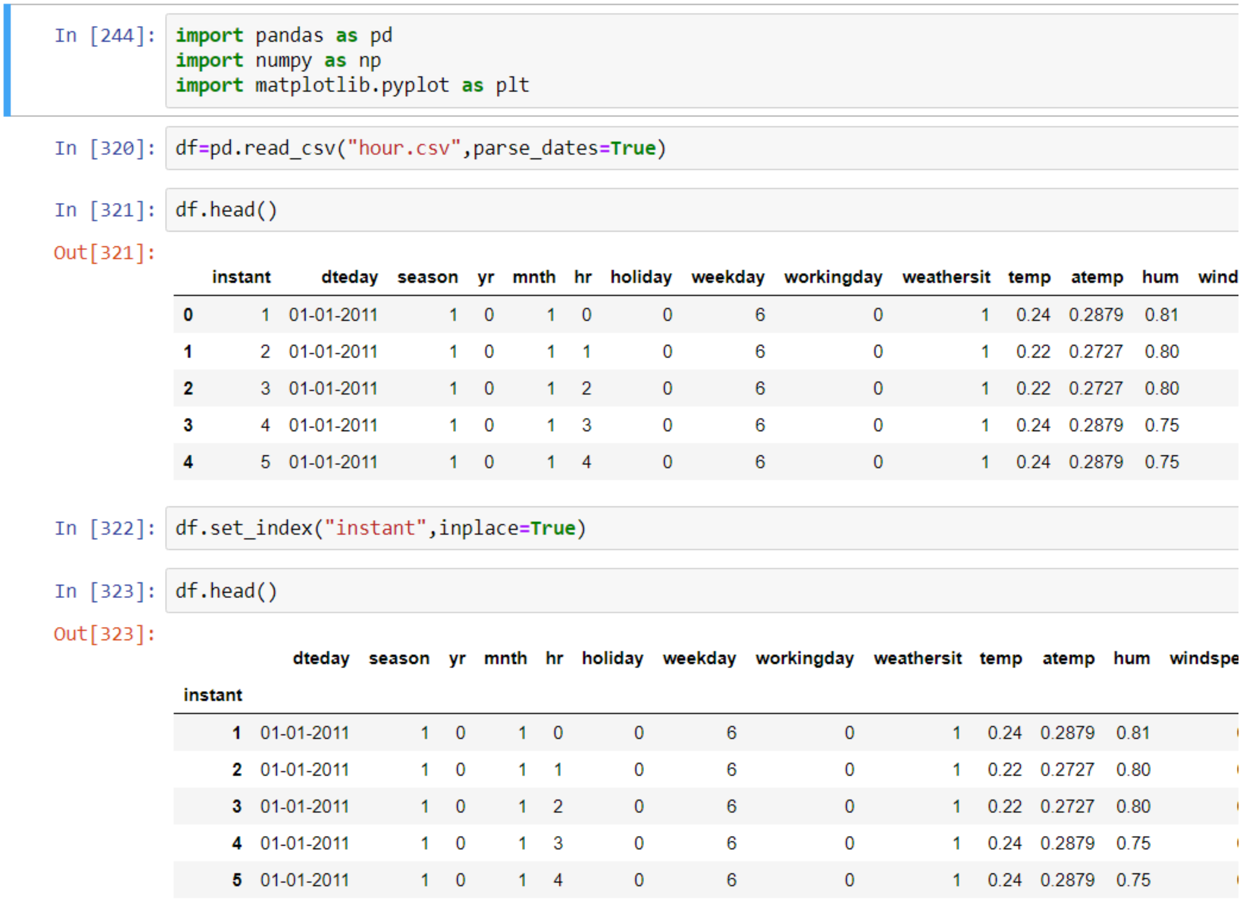
Respected team , I am Gurdeep Singh Bhatia from Haryana , Yamunanagar
by following the instructions , I have successfully completed my project , and following is my project report ,
First the git of my project is :
https://github.com/gurdeep-singh-bhatia/ML-projects/blob/master/Predicting%20The%20Count%20Of%20Bikes%20Rented.ipynb

Now I want to explain my code :

First of all I import the respective data set in my Jupyter notebook using pandas lib. then read it , and set the index of the data frame



```
In [244]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
```

```
In [320]: df=pd.read_csv("hour.csv",parse_dates=True)
```

```
In [321]: df.head()
```

Out[321]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	wind
0	1	01-01-2011	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	
1	2	01-01-2011	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	
2	3	01-01-2011	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	
3	4	01-01-2011	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	
4	5	01-01-2011	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	

```
In [322]: df.set_index("instant",inplace=True)
```

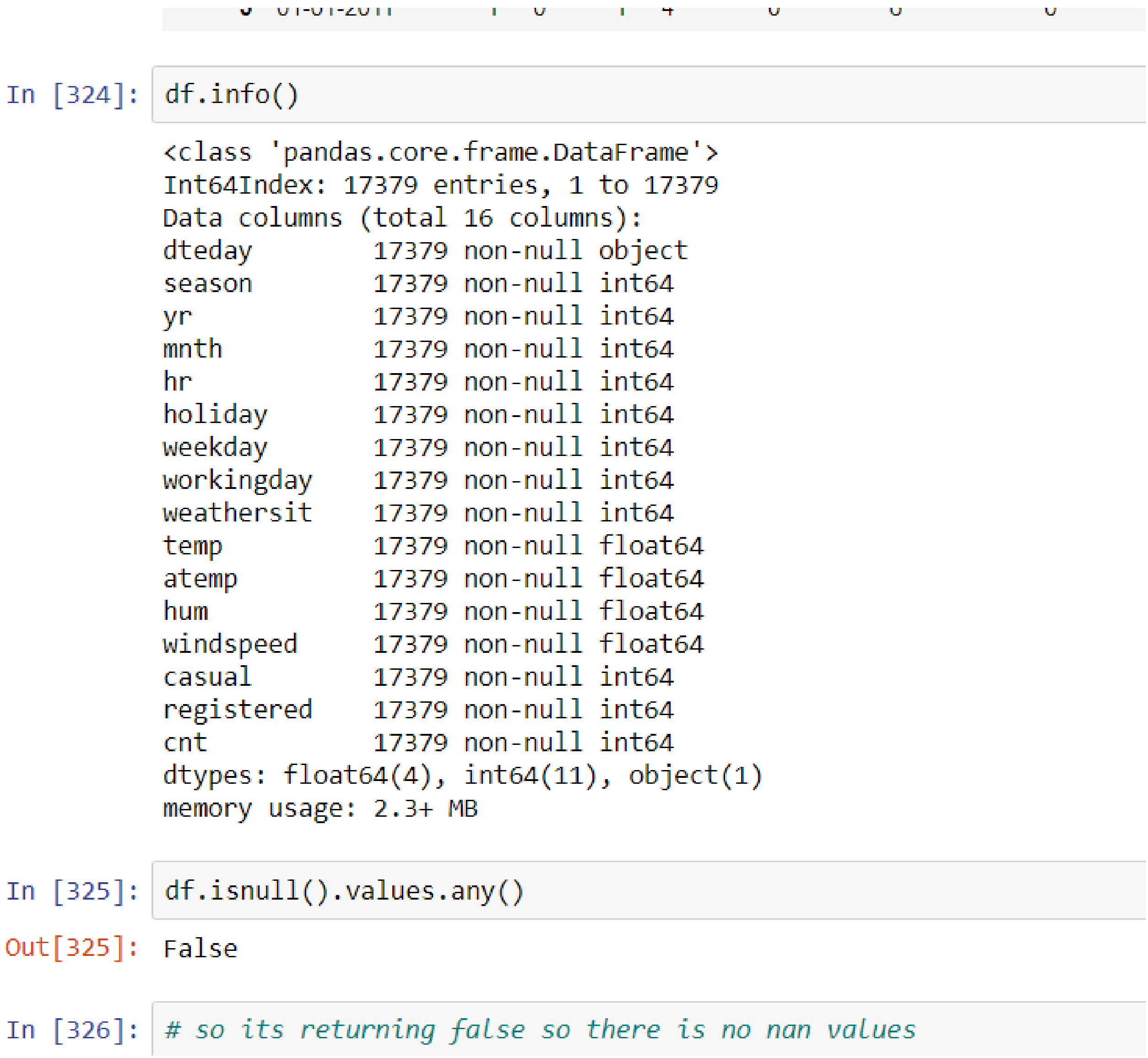
```
In [323]: df.head()
```

Out[323]:

	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspe
instant													
1	01-01-2011	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	
2	01-01-2011	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	
3	01-01-2011	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	
4	01-01-2011	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	
5	01-01-2011	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	

Then I check whether the data set or data frame has any missing (nan) values or not , because its very important to solve the missing value problem as it can impact our ml model

MISSING DATA



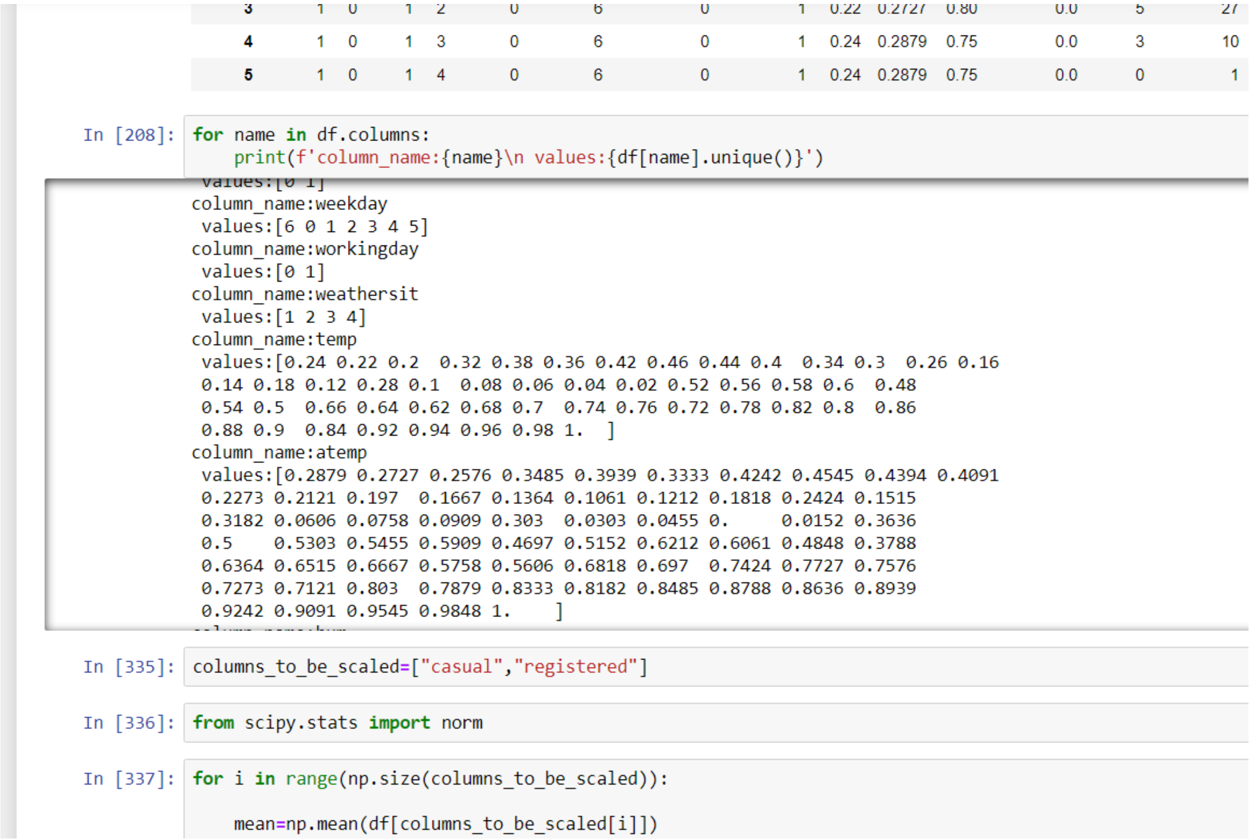
```
   0  01-01-2011       1  0      1   4        0         0           0
```

```
In [324]: df.info()

          <class 'pandas.core.frame.DataFrame'>
          Int64Index: 17379 entries, 1 to 17379
          Data columns (total 16 columns):
          dteday        17379 non-null object
          season        17379 non-null int64
          yr            17379 non-null int64
          mnth          17379 non-null int64
          hr            17379 non-null int64
          holiday       17379 non-null int64
          weekday       17379 non-null int64
          workingday    17379 non-null int64
          weathersit    17379 non-null int64
          temp          17379 non-null float64
          atemp         17379 non-null float64
          hum           17379 non-null float64
          windspeed     17379 non-null float64
          casual        17379 non-null int64
          registered    17379 non-null int64
          cnt           17379 non-null int64
          dtypes: float64(4), int64(11), object(1)
          memory usage: 2.3+ MB

In [325]: df.isnull().values.any()

Out[325]: False

In [326]: # so its returning false so there is no nan values
```

as we can see there is no missing value .

Then I check that which column needs the scaling(feature scaling)

SCALING



3	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27
4	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10
5	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1

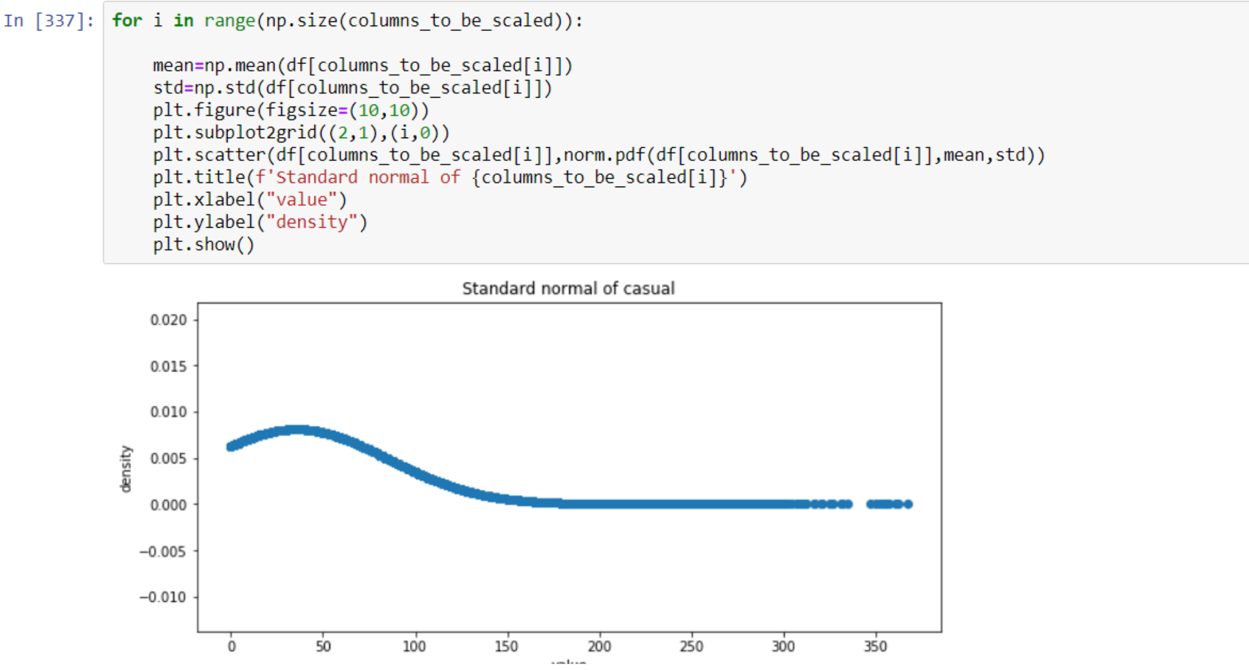
In [208]:
```
for name in df.columns:
    print(f'column_name:{name}\n values:{df[name].unique()}')
```

```
 values:[0 1]
column_name:weekday
 values:[6 0 1 2 3 4 5]
column_name:workingday
 values:[0 1]
column_name:weathersit
 values:[1 2 3 4]
column_name:temp
 values:[0.24 0.22 0.2  0.32 0.38 0.36 0.42 0.46 0.44 0.4  0.34 0.3  0.26 0.16
 0.14 0.18 0.12 0.28 0.1  0.08 0.06 0.04 0.02 0.52 0.56 0.58 0.6  0.48
 0.54 0.5  0.66 0.64 0.62 0.68 0.7  0.74 0.76 0.72 0.78 0.82 0.8  0.86
 0.88 0.9  0.84 0.92 0.94 0.96 0.98 1.  ]
column_name:atemp
 values:[0.2879 0.2727 0.2576 0.3485 0.3939 0.3333 0.4242 0.4545 0.4394 0.4091
 0.2273 0.2121 0.197  0.1667 0.1364 0.1061 0.1212 0.1818 0.2424 0.1515
 0.3182 0.0606 0.0758 0.0909 0.303  0.0303 0.0455 0.     0.0152 0.3636
 0.5    0.5303 0.5455 0.5909 0.4697 0.5152 0.6212 0.6061 0.4848 0.3788
 0.6364 0.6515 0.6667 0.5758 0.5606 0.6818 0.697  0.7424 0.7727 0.7576
 0.7273 0.7121 0.803  0.7879 0.8333 0.8182 0.8485 0.8788 0.8636 0.8939
 0.9242 0.9091 0.9545 0.9848 1.    ]
column_name:hum
```

In [335]:
```
columns_to_be_scaled=["casual","registered"]
```

In [336]:
```
from scipy.stats import norm
```

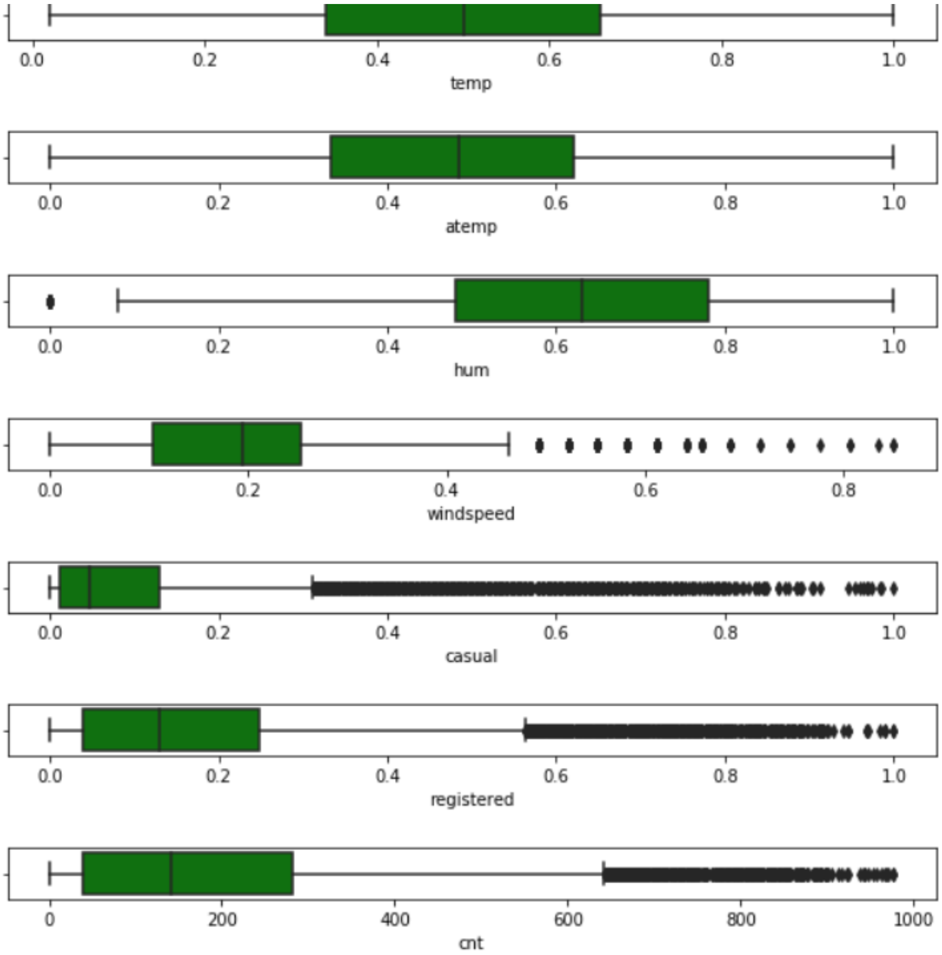
In [337]:
```
for i in range(np.size(columns_to_be_scaled)):

    mean=np.mean(df[columns_to_be_scaled[i]])
```



In [337]:
```
for i in range(np.size(columns_to_be_scaled)):

    mean=np.mean(df[columns_to_be_scaled[i]])
    std=np.std(df[columns_to_be_scaled[i]])
    plt.figure(figsize=(10,10))
    plt.subplot2grid((2,1),(i,0))
    plt.scatter(df[columns_to_be_scaled[i]],norm.pdf(df[columns_to_be_scaled[i]],mean,std))
    plt.title(f'Standard normal of {columns_to_be_scaled[i]}')
    plt.xlabel("value")
    plt.ylabel("density")
    plt.show()
```

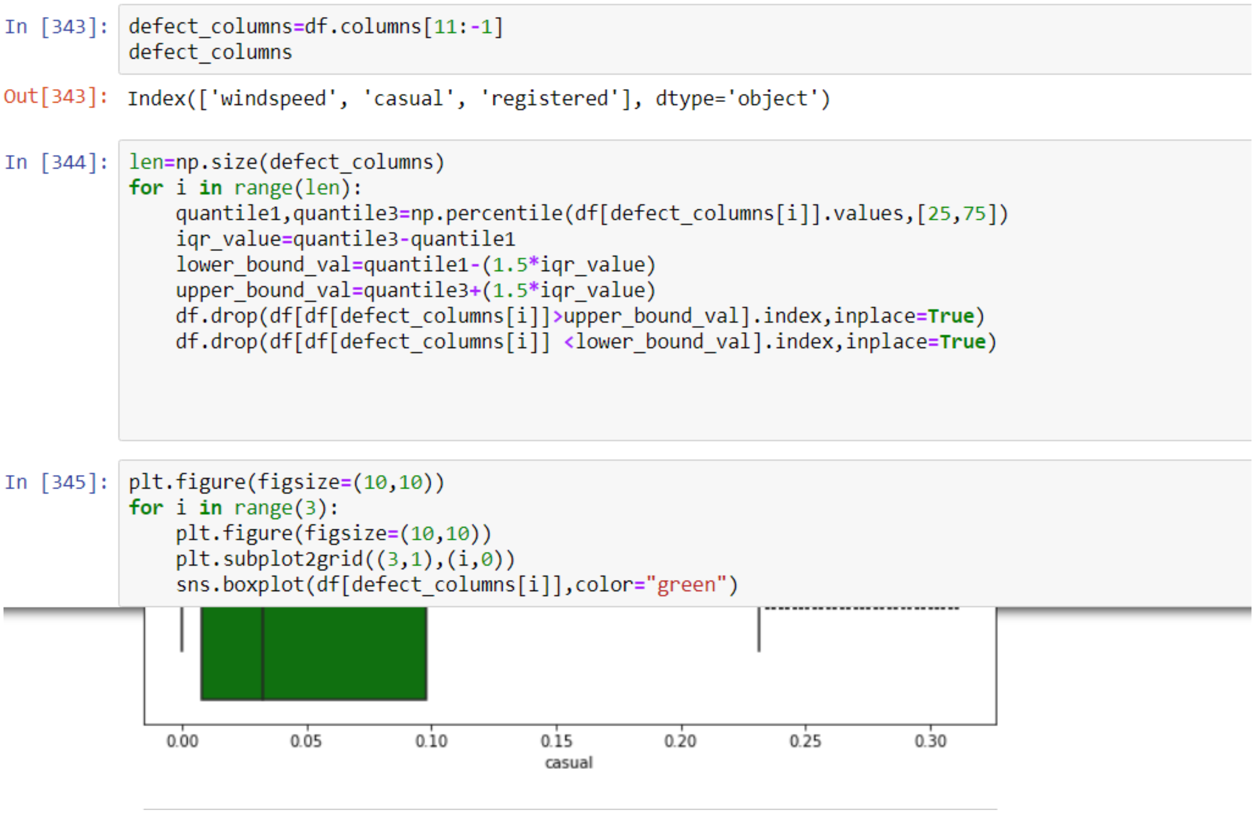
Now to check which scaling will be best , I do some analysis , because as we know that if a column is following Gaussian distribution , then applying standardization on that column will increase our model performance otherwise we will apply min max scaling ,Now at the last pic , we can see that the data is not in normal/Gaussian form so we will apply min max scaling

After this I apply box plot to check the outliers



Now as we can see that there are lots of outlier , so lets remove these all , by applying IQR METHOD

Removing outlier by using INTER QUARTILE RANGE



```
In [343]: defect_columns=df.columns[11:-1]
          defect_columns
```

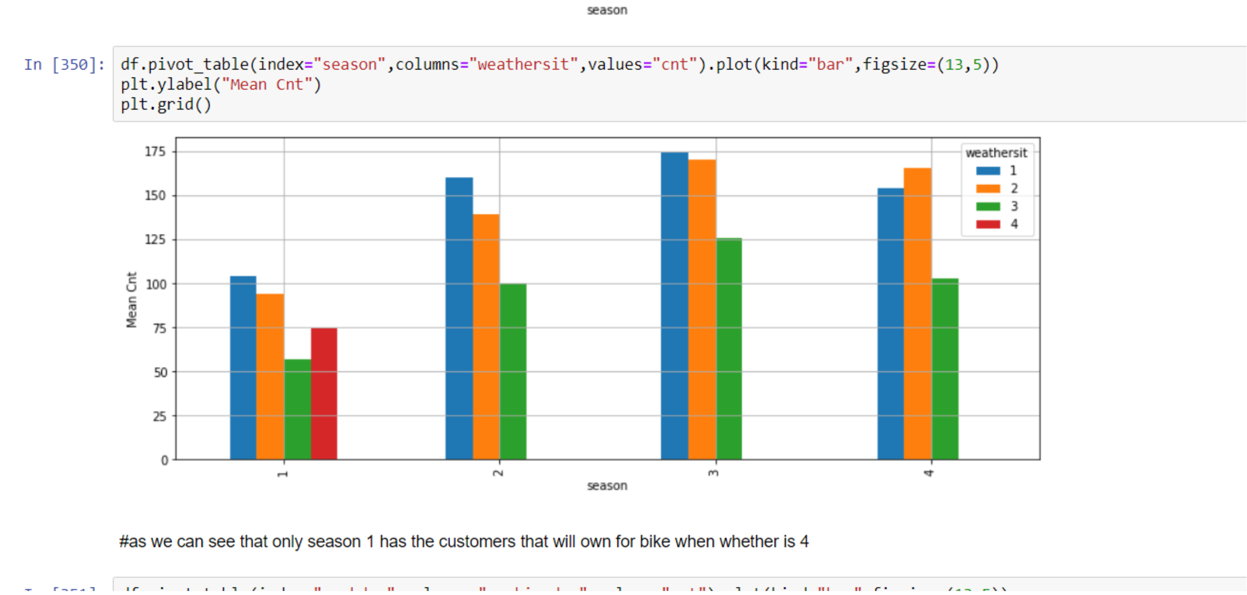
```
Out[343]: Index(['windspeed', 'casual', 'registered'], dtype='object')
```

```
In [344]: len=np.size(defect_columns)
          for i in range(len):
              quantile1,quantile3=np.percentile(df[defect_columns[i]].values,[25,75])
              iqr_value=quantile3-quantile1
              lower_bound_val=quantile1-(1.5*iqr_value)
              upper_bound_val=quantile3+(1.5*iqr_value)
              df.drop(df[df[defect_columns[i]]>upper_bound_val].index,inplace=True)
              df.drop(df[df[defect_columns[i]] <lower_bound_val].index,inplace=True)
```

```
In [345]: plt.figure(figsize=(10,10))
          for i in range(3):
              plt.figure(figsize=(10,10))
              plt.subplot2grid((3,1),(i,0))
              sns.boxplot(df[defect_columns[i]],color="green")
```

After that perform eda to get more insights of the data as shown below :

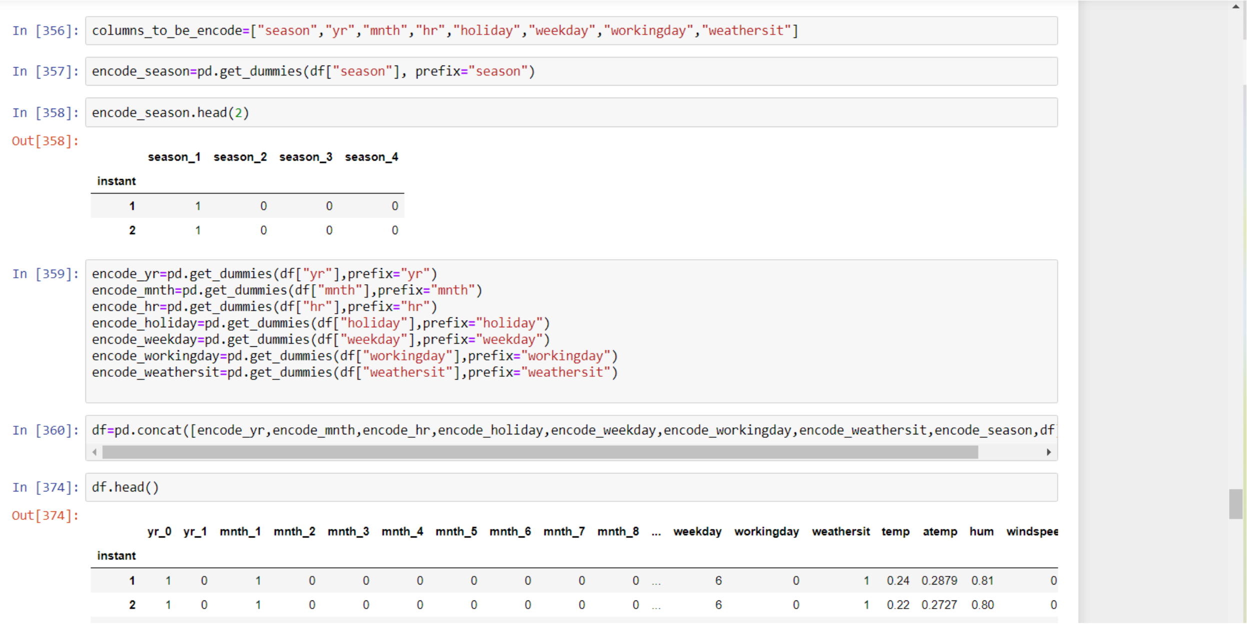
EDA



```
In [350]: df.pivot_table(index="season",columns="weathersit",values="cnt").plot(kind="bar",figsize=(13,5))
          plt.ylabel("Mean Cnt")
          plt.grid()
```

#as we can see that only season 1 has the customers that will own for bike whem whether is 4

NOW DOING ONE HOT ENCODING AFTER EDA :



```
In [356]: columns_to_be_encode=["season","yr","mnth","hr","holiday","weekday","workingday","weathersit"]

In [357]: encode_season=pd.get_dummies(df["season"], prefix="season")

In [358]: encode_season.head(2)
```

Out[358]:

	season_1	season_2	season_3	season_4
instant				
1	1	0	0	0
2	1	0	0	0

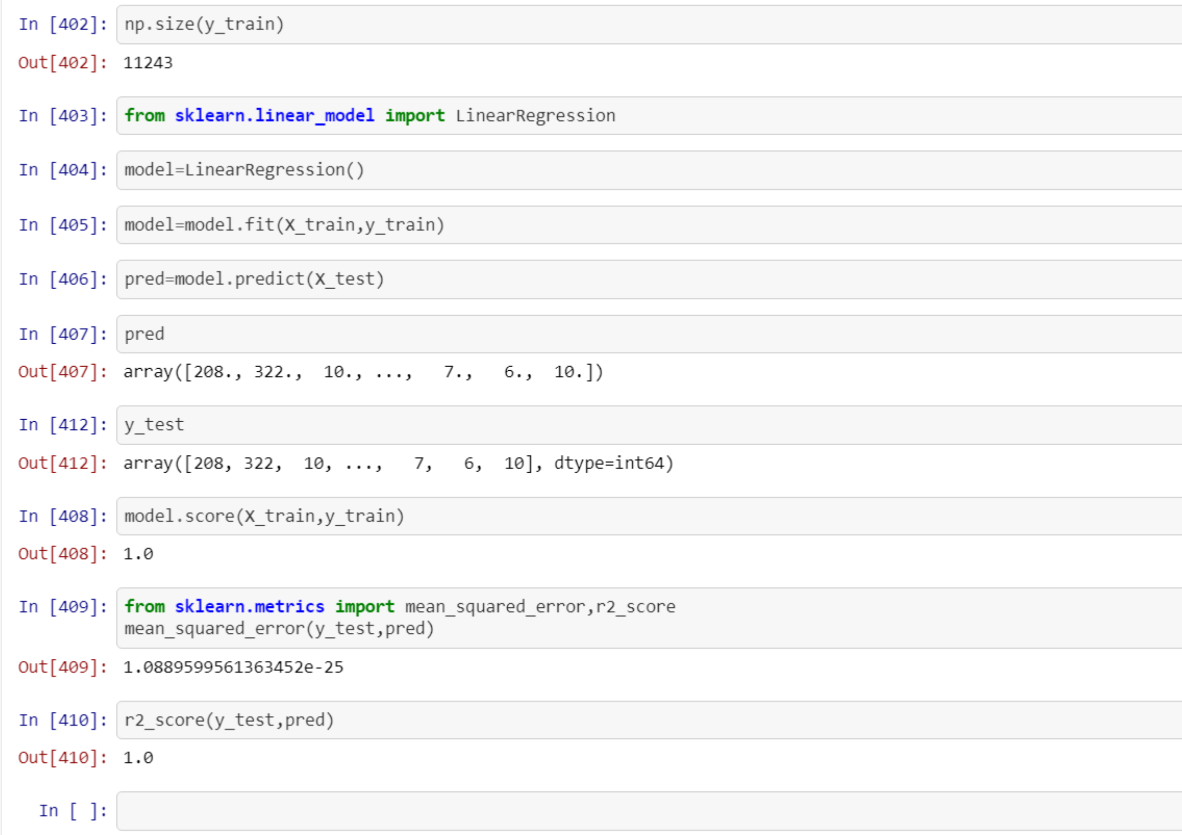
```
In [359]: encode_yr=pd.get_dummies(df["yr"],prefix="yr")
          encode_mnth=pd.get_dummies(df["mnth"],prefix="mnth")
          encode_hr=pd.get_dummies(df["hr"],prefix="hr")
          encode_holiday=pd.get_dummies(df["holiday"],prefix="holiday")
          encode_weekday=pd.get_dummies(df["weekday"],prefix="weekday")
          encode_workingday=pd.get_dummies(df["workingday"],prefix="workingday")
          encode_weathersit=pd.get_dummies(df["weathersit"],prefix="weathersit")

In [360]: df=pd.concat([encode_yr,encode_mnth,encode_hr,encode_holiday,encode_weekday,encode_workingday,encode_weathersit,encode_season,df

In [374]: df.head()
```

Out[374]:

	yr_0	yr_1	mnth_1	mnth_2	mnth_3	mnth_4	mnth_5	mnth_6	mnth_7	mnth_8	...	weekday	workingday	weathersit	temp	atemp	hum	windspee
instant																		
1	1	0	1	0	0	0	0	0	0	0	...	6	0	1	0.24	0.2879	0.81	0
2	1	0	1	0	0	0	0	0	0	0	...	6	0	1	0.22	0.2727	0.80	0

Now all features are set , so now finally made linear regression model , and this is done as shown below , and we can see that my model is giving r2_score of 1 on test data set

MODELLING BUILDING:



```
In [402]: np.size(y_train)

Out[402]: 11243

In [403]: from sklearn.linear_model import LinearRegression

In [404]: model=LinearRegression()

In [405]: model=model.fit(X_train,y_train)

In [406]: pred=model.predict(X_test)

In [407]: pred

Out[407]: array([208., 322.,  10., ...,   7.,   6.,  10.])

In [412]: y_test

Out[412]: array([208, 322,  10, ...,   7,   6,  10], dtype=int64)

In [408]: model.score(X_train,y_train)

Out[408]: 1.0

In [409]: from sklearn.metrics import mean_squared_error,r2_score
          mean_squared_error(y_test,pred)

Out[409]: 1.0889599561363452e-25

In [410]: r2_score(y_test,pred)

Out[410]: 1.0

  In [ ]:
```