```
# Bike Sharing System -Part I:  Exploratory Analysis

# import the necessary packages

import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
import csv
import os
import pandas as pd
import shutil, glob
```

```
# After downloading the train.csv and test.csv files, we are reading the two files in
DataFrames

path='F:/dataprojects'
```

```
df = pd.read_csv('F:/dataprojects/train.csv')
dftest=pd.read_csv('F:/dataprojects/test.csv')
```

```
# Exploratory analysis of the train DataFrame
```

```
# In my case, there where some problems with identifying the 'count' column of the Dat
aFrame, so we renamed it
#Rename the last column of the DataFrame
df.rename(columns={'count': 'totalcustomers'}, inplace=True)
list(df.columns.values)
```

```
['datetime',
 'season',
 'holiday',
 'workingday',
 'weather',
 'temp',
 'atemp',
 'humidity',
 'windspeed',
 'casual',
 'registered',
 'totalcustomers']
```

```
#feature engineering
# We want to analyze the bike sharing distribution on hours, days, months, years.
# so let's regain these informations from the 'datetime' column
df['datetime']=pd.to_datetime(df['datetime'])
print(df['datetime'].dtype)
df['year'] = df['datetime'].dt.year
df['month'] = df['datetime'].dt.month
df['day'] = df['datetime'].dt.day
df['hour']=df['datetime'].dt.hour
cols = df.columns.tolist()
cols=cols[-4:]+cols[:-4]
df=df[cols]
df=df.drop('datetime', axis=1)
list(df.columns.values)
```

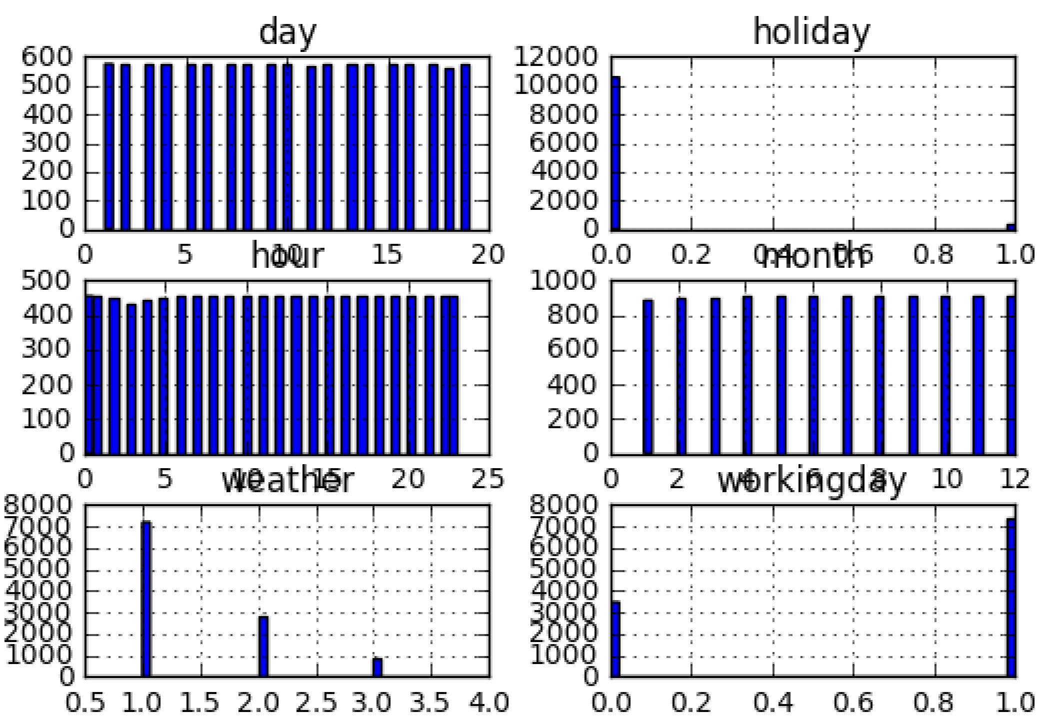
```
datetime64[ns]





['year',
 'month',
 'day',
 'hour',
 'season',
 'holiday',
 'workingday',
 'weather',
 'temp',
 'atemp',
 'humidity',
 'windspeed',
 'casual',
 'registered',
 'totalcustomers']
```

```
# check if there are any missing values in the DataFrame: in this case there are no mi
ssing values in our database
# the isnull() detects missing values in the specified DataFrame df
df.isnull().any()
# there are 10886 entries in our dataframe
len(df.weather)
```

```
10886
```

```
# Analyze the data in order to bring a better understanding of the data we are looking at.
# The shape and dispersion of the data output can help significantly in noticing the e volution of one variable.
# Visualisation tools are extremely valuable here
# For each of the predictor variables, we first see the histogram :
# (season, workingday, weather, temp, atemp, humidity, windspeed, casual, registered, count).
# We notice that most of our variables do not follow a normal distribution
df1=df[['month', 'day', 'hour', 'holiday','weather','workingday']]
df2=df[['temp', 'atemp','humidity', 'windspeed']]
df3=df[['casual', 'registered', 'totalcustomers']]
```

```
df1.hist(layout=(3,2), bins='rice')
plt.show()
# nb of bins has to be manually chosen, to reflect the nb of elements in the bin
```



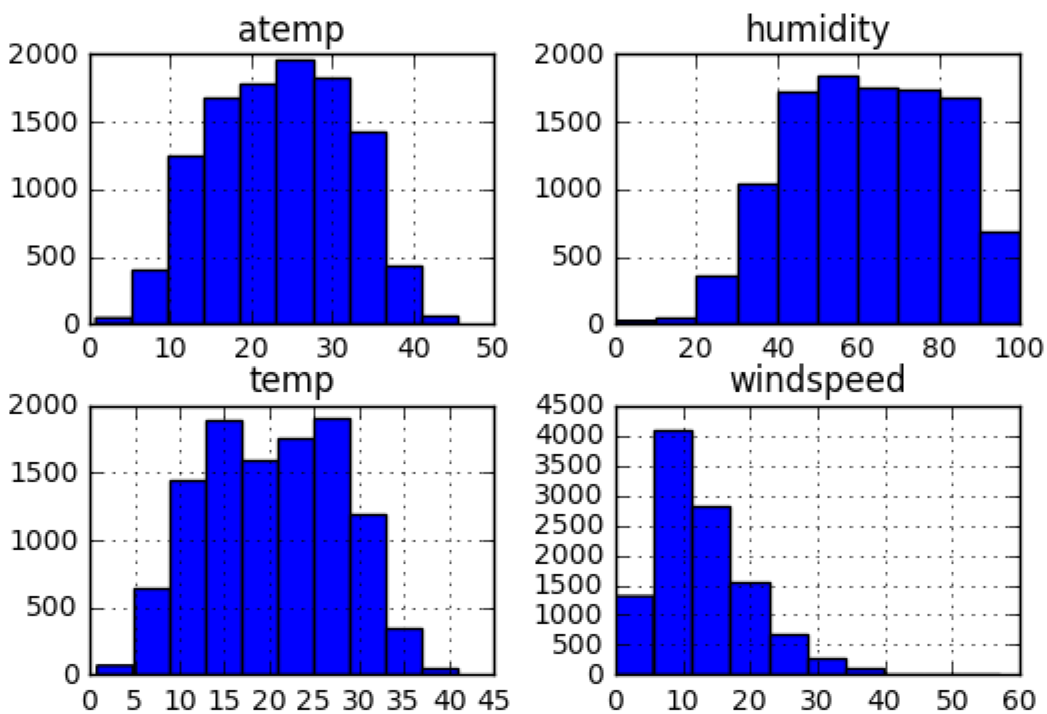
png

```
print len(df[df['day']==1])
print len(df[df['weather']==1])
```

```
575
7192
```

```
# Histograms above correspond to variables of nominal type data.
# Nominal refers to data that is categorical, for example substracting one month from
another has no meaning.
# workingday predictor: if day is neither weekend nor holiday is 1, otherwise is 0.
# weather predictor: + weathersit :
#- 1: Clear, Few clouds, Partly cloudy, Partly cloudy
#- 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
#- 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattere
d clouds
#- 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
# temperature predictor: Normalized temperature in Celsius.
# The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hou
rly scale)
# hour predictor
# The most values seems to be collected in the first and last hour of the day
```

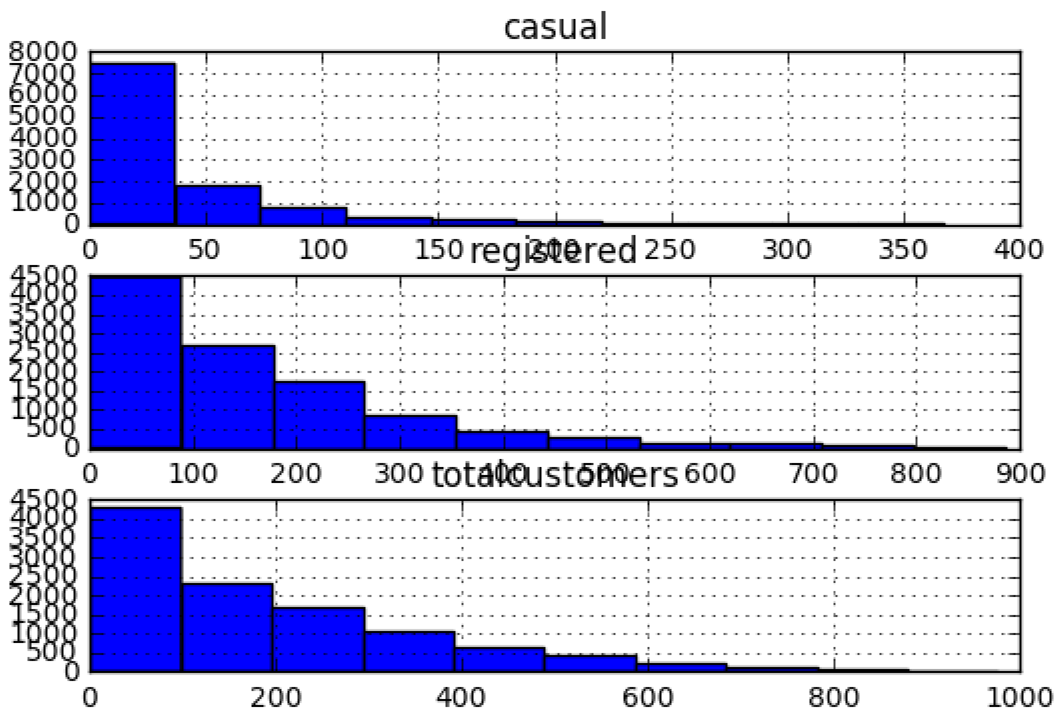
```
df2.hist(layout=(2,2))
plt.show()
```



png

```
# The histograms above are example of interval data, these type of data have scale.
# The variables do not follow a normal distribution either.
```

```
df3.hist(layout=(3,1))
plt.show()
```



png

```
# The diagrams are positively skewed
```

```
# Descriptive statistic elements
# let's evaluate the mean and the median values of the customers for each season.
# We check for noticeable differences between the mean and the median.
# As the mean is more susceptible to outliers, it would conceivably be distorted great
ly in the presence of a large number of outliers or large outliers values.
# For variables depicting the total number of clients, casual or registered show diffe
rences between the
# mean value and the median. This shows the possible existence of outliers.
```

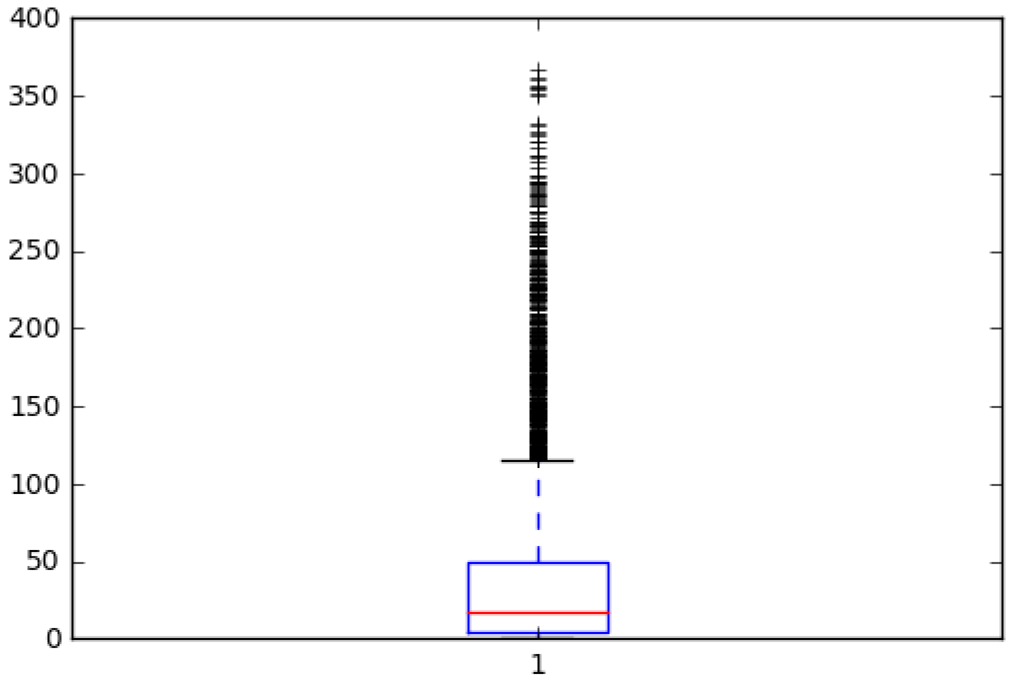
```
df["casual"].mean()
```

```
36.02195480433584
```

```
df["casual"].median()
```

```
17.0
```

```
# casual customers boxplot
plt.boxplot(df.casual)
plt.show()
data=df.casual
median = np.median(data)
upper_quartile = np.percentile(data, 75)
lower_quartile = np.percentile(data, 25)
iqr = upper_quartile - lower_quartile
upper_whisker =data[data<=upper_quartile+1.5*iqr].max()
lower_whisker = data[data>=lower_quartile-1.5*iqr].min()

# the Boxplot does show the presence of many outliers
```



png

```
df["totalcustomers"].mean()
```

```
191.57413191254824
```

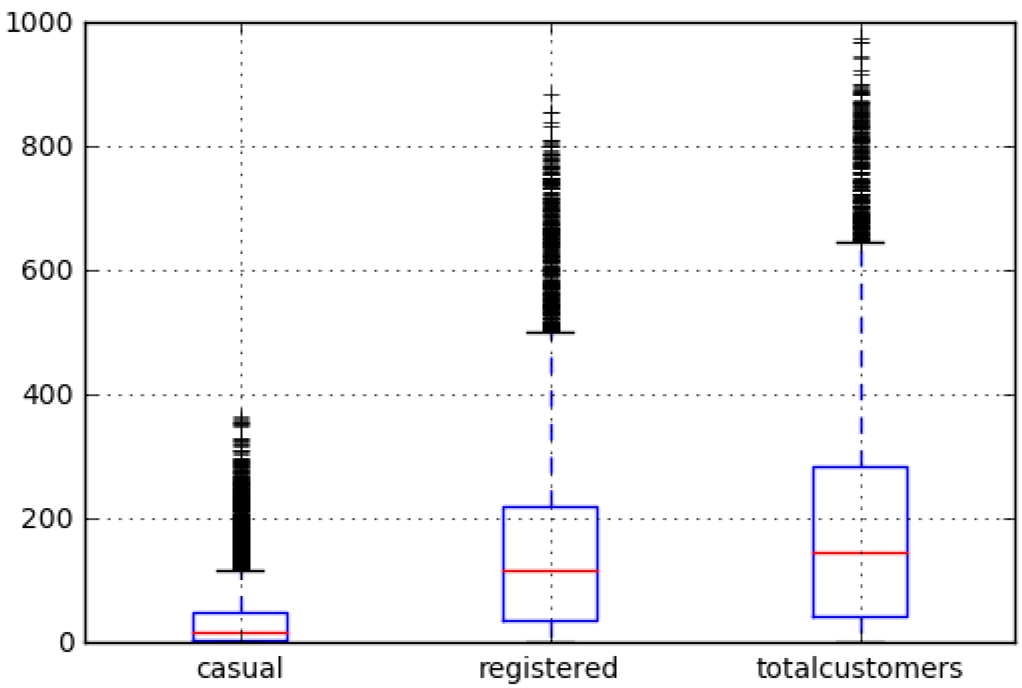
```
df["totalcustomers"].median()
```

```
145.0
```

```
#  for the variables like season, holiday, workingday, a median or mean value makes no
sense.
```

```
# boxes: the main body of the boxplot showing the quartiles and the medianâs confidenc
e intervals if enabled.
# medians: horizonal lines at the median of each box.
# whiskers: the vertical lines extending to the most extreme, n-outlier data points.
# caps: the horizontal lines at the ends of the whiskers.
# fliers: points representing data that extend beyone the whiskers (outliers).
```

```
df3.boxplot()
plt.show()
```



png

```
# we can then extract all the information in the boxplots that you are interested in,
e.g. median, upper_quartile, iqr, etc.
# I have wrote and example for the 'totalcustomers' variable
data=df.totalcustomers
median = np.median(data)
upper_quartile = np.percentile(data, 75)
lower_quartile = np.percentile(data, 25)
iqr = upper_quartile - lower_quartile
upper_whisker =data[data<=upper_quartile+1.5*iqr].max()
lower_whisker = data[data>=lower_quartile-1.5*iqr].min()
print upper_whisker
print iqr
```

```
647
242.0
```

```
# For the humidity variable there seems to be no significant difference between the me
an and the median values.
# Same situation for the windspeed and for the temperature, which means that most prob
ably outliers are not present
```

```
df["humidity"].mean()
```

```
61.88645967297446
```

```
df["humidity"].median()
```

```
62.0
```

```
df["windspeed"].mean()
```

```
12.799395406945093
```

```
df["windspeed"].median()
```

```
12.998
```

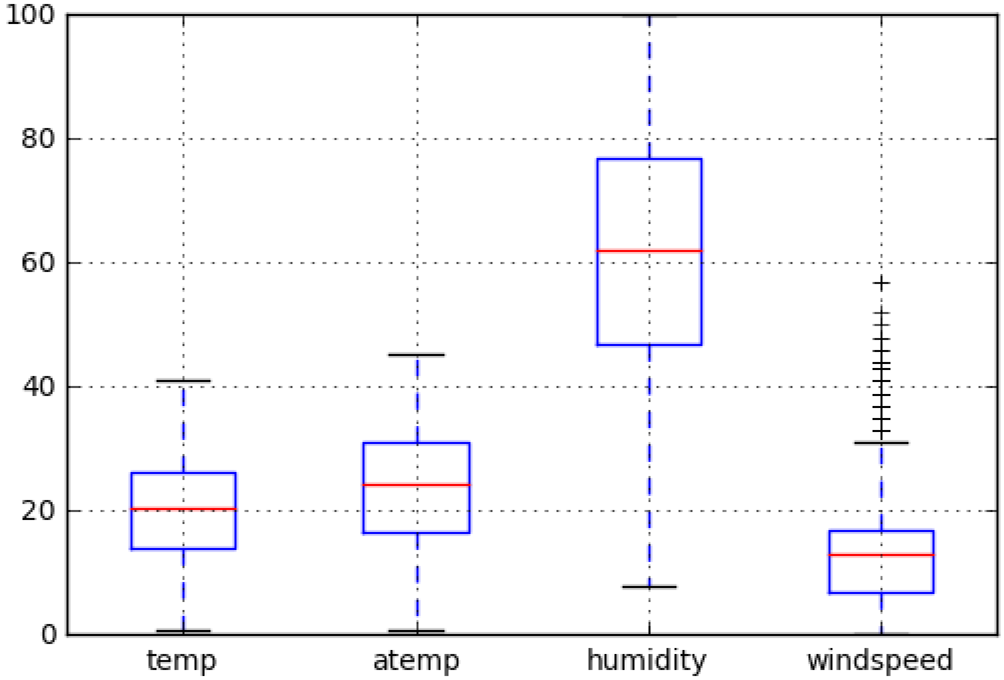
```
df["temp"].mean()
```

```
20.230859819952173
```

```
df["temp"].median()
```

```
20.5
```

```
df2.boxplot()
plt.show()
# for the variable windspeed a small number of outliers is present, the influence on t
he mean value is not significantly large
```



png

```
# We will continue to investigate these attributes in the nex paragraph about normalit
y
# The Shapiro-Wills normality test
```

```
# Let us start by checking the correlation between our variables
# Analyze the level of correlation between the DataFrame variables
```

```
df.corr(method='pearson', min_periods=1)
# The correlation matrix show some existing relation between the nb of casual and tota
l customers,
# between the registered and total nb of customers,
# however, weather characteristic does not seems to have a significant impact on the n
b of customers
```

```
<tr style="text-align: right;">
  <th></th>
  <th>year</th>
  <th>month</th>
  <th>day</th>
  <th>hour</th>
  <th>season</th>
  <th>holiday</th>
  <th>workingday</th>
  <th>weather</th>
  <th>temp</th>
  <th>atemp</th>
  <th>humidity</th>
  <th>windspeed</th>
  <th>casual</th>
  <th>registered</th>
  <th>totalcustomers</th>
</tr>
```

```
<tr>
  <th>year</th>
  <td>1.000000</td>
  <td>-0.004932</td>
  <td>0.001800</td>
  <td>-0.004234</td>
  <td>-0.004797</td>
  <td>0.012021</td>
  <td>-0.002482</td>
  <td>-0.012548</td>
  <td>0.061226</td>
  <td>0.058540</td>
  <td>-0.078606</td>
  <td>-0.015221</td>
  <td>0.145241</td>
  <td>0.264265</td>
  <td>0.260403</td>
</tr>
<tr>
  <th>month</th>
  <td>-0.004932</td>
  <td>1.000000</td>
  <td>0.001974</td>
  <td>-0.006818</td>
  <td>0.971524</td>
  <td>0.001731</td>
  <td>-0.003394</td>
  <td>0.012144</td>
  <td>0.257589</td>
  <td>0.264173</td>
  <td>0.204537</td>
  <td>-0.150192</td>
  <td>0.092722</td>
  <td>0.169451</td>
  <td>0.166862</td>
</tr>
<tr>
  <th>day</th>
  <td>0.001800</td>
  <td>0.001974</td>
  <td>1.000000</td>
  <td>0.001132</td>
  <td>0.001729</td>
  <td>-0.015877</td>
  <td>0.009829</td>
  <td>-0.007890</td>
  <td>0.015551</td>
  <td>0.011866</td>
```

```
    <td>-0.011335</td>
    <td>0.036157</td>
    <td>0.014109</td>
    <td>0.019111</td>
    <td>0.019826</td>
  </tr>
  <tr>
    <th>hour</th>
    <td>-0.004234</td>
    <td>-0.006818</td>
    <td>0.001132</td>
    <td>1.000000</td>
    <td>-0.006546</td>
    <td>-0.000354</td>
    <td>0.002780</td>
    <td>-0.022740</td>
    <td>0.145430</td>
    <td>0.140343</td>
    <td>-0.278011</td>
    <td>0.146631</td>
    <td>0.302045</td>
    <td>0.380540</td>
    <td>0.400601</td>
  </tr>
  <tr>
    <th>season</th>
    <td>-0.004797</td>
    <td>0.971524</td>
    <td>0.001729</td>
    <td>-0.006546</td>
    <td>1.000000</td>
    <td>0.029368</td>
    <td>-0.008126</td>
    <td>0.008879</td>
    <td>0.258689</td>
    <td>0.264744</td>
    <td>0.190610</td>
    <td>-0.147121</td>
    <td>0.096758</td>
    <td>0.164011</td>
    <td>0.163439</td>
  </tr>
  <tr>
    <th>holiday</th>
    <td>0.012021</td>
    <td>0.001731</td>
    <td>-0.015877</td>
    <td>-0.000354</td>
    <td>0.029368</td>
```

```
    <td>1.000000</td>
    <td>-0.250491</td>
    <td>-0.007074</td>
    <td>0.000295</td>
    <td>-0.005215</td>
    <td>0.001929</td>
    <td>0.008409</td>
    <td>0.043799</td>
    <td>-0.020956</td>
    <td>-0.005393</td>
  </tr>
  <tr>
    <th>workingday</th>
    <td>-0.002482</td>
    <td>-0.003394</td>
    <td>0.009829</td>
    <td>0.002780</td>
    <td>-0.008126</td>
    <td>-0.250491</td>
    <td>1.000000</td>
    <td>0.033772</td>
    <td>0.029966</td>
    <td>0.024660</td>
    <td>-0.010880</td>
    <td>0.013373</td>
    <td>-0.319111</td>
    <td>0.119460</td>
    <td>0.011594</td>
  </tr>
  <tr>
    <th>weather</th>
    <td>-0.012548</td>
    <td>0.012144</td>
    <td>-0.007890</td>
    <td>-0.022740</td>
    <td>0.008879</td>
    <td>-0.007074</td>
    <td>0.033772</td>
    <td>1.000000</td>
    <td>-0.055035</td>
    <td>-0.055376</td>
    <td>0.406244</td>
    <td>0.007261</td>
    <td>-0.135918</td>
    <td>-0.109340</td>
    <td>-0.128655</td>
  </tr>
  <tr>
    <th>temp</th>
```

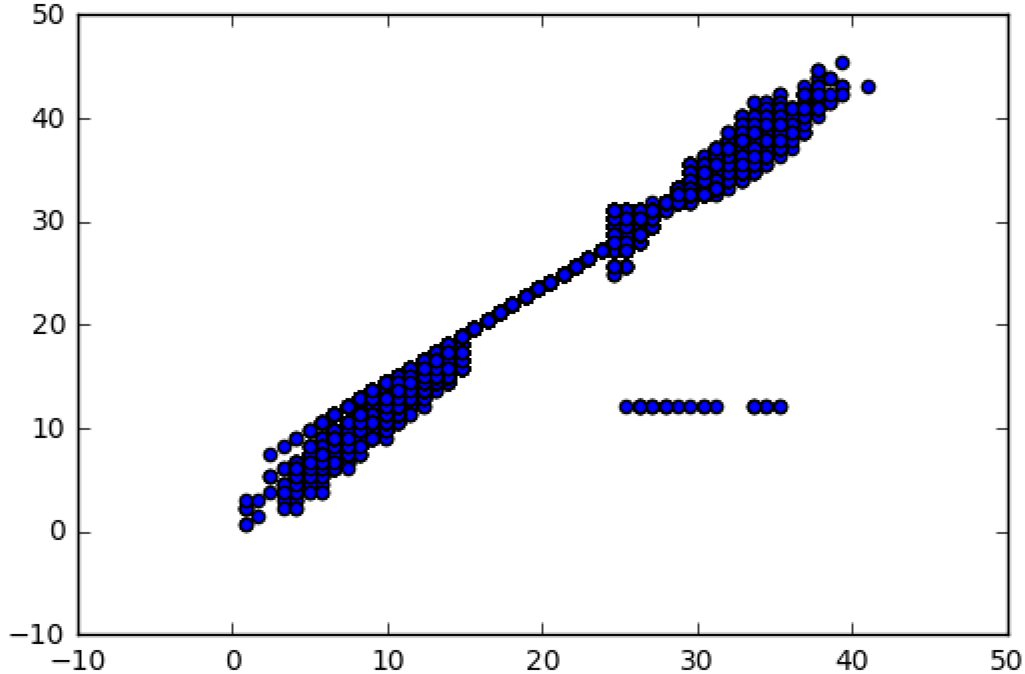
```
    <td>0.061226</td>
    <td>0.257589</td>
    <td>0.015551</td>
    <td>0.145430</td>
    <td>0.258689</td>
    <td>0.000295</td>
    <td>0.029966</td>
    <td>-0.055035</td>
    <td>1.000000</td>
    <td>0.984948</td>
    <td>-0.064949</td>
    <td>-0.017852</td>
    <td>0.467097</td>
    <td>0.318571</td>
    <td>0.394454</td>
  </tr>
  <tr>
    <th>atemp</th>
    <td>0.058540</td>
    <td>0.264173</td>
    <td>0.011866</td>
    <td>0.140343</td>
    <td>0.264744</td>
    <td>-0.005215</td>
    <td>0.024660</td>
    <td>-0.055376</td>
    <td>0.984948</td>
    <td>1.000000</td>
    <td>-0.043536</td>
    <td>-0.057473</td>
    <td>0.462067</td>
    <td>0.314635</td>
    <td>0.389784</td>
  </tr>
  <tr>
    <th>humidity</th>
    <td>-0.078606</td>
    <td>0.204537</td>
    <td>-0.011335</td>
    <td>-0.278011</td>
    <td>0.190610</td>
    <td>0.001929</td>
    <td>-0.010880</td>
    <td>0.406244</td>
    <td>-0.064949</td>
    <td>-0.043536</td>
    <td>1.000000</td>
    <td>-0.318607</td>
    <td>-0.348187</td>
```

```
    <td>-0.265458</td>
    <td>-0.317371</td>
  </tr>
  <tr>
    <th>windspeed</th>
    <td>-0.015221</td>
    <td>-0.150192</td>
    <td>0.036157</td>
    <td>0.146631</td>
    <td>-0.147121</td>
    <td>0.008409</td>
    <td>0.013373</td>
    <td>0.007261</td>
    <td>-0.017852</td>
    <td>-0.057473</td>
    <td>-0.318607</td>
    <td>1.000000</td>
    <td>0.092276</td>
    <td>0.091052</td>
    <td>0.101369</td>
  </tr>
  <tr>
    <th>casual</th>
    <td>0.145241</td>
    <td>0.092722</td>
    <td>0.014109</td>
    <td>0.302045</td>
    <td>0.096758</td>
    <td>0.043799</td>
    <td>-0.319111</td>
    <td>-0.135918</td>
    <td>0.467097</td>
    <td>0.462067</td>
    <td>-0.348187</td>
    <td>0.092276</td>
    <td>1.000000</td>
    <td>0.497250</td>
    <td>0.690414</td>
  </tr>
  <tr>
    <th>registered</th>
    <td>0.264265</td>
    <td>0.169451</td>
    <td>0.019111</td>
    <td>0.380540</td>
    <td>0.164011</td>
    <td>-0.020956</td>
    <td>0.119460</td>
    <td>-0.109340</td>
```

```
  <td>0.318571</td>
  <td>0.314635</td>
  <td>-0.265458</td>
  <td>0.091052</td>
  <td>0.497250</td>
  <td>1.000000</td>
  <td>0.970948</td>
</tr>
<tr>
  <th>totalcustomers</th>
  <td>0.260403</td>
  <td>0.166862</td>
  <td>0.019826</td>
  <td>0.400601</td>
  <td>0.163439</td>
  <td>-0.005393</td>
  <td>0.011594</td>
  <td>-0.128655</td>
  <td>0.394454</td>
  <td>0.389784</td>
  <td>-0.317371</td>
  <td>0.101369</td>
  <td>0.690414</td>
  <td>0.970948</td>
  <td>1.000000</td>
</tr>
```

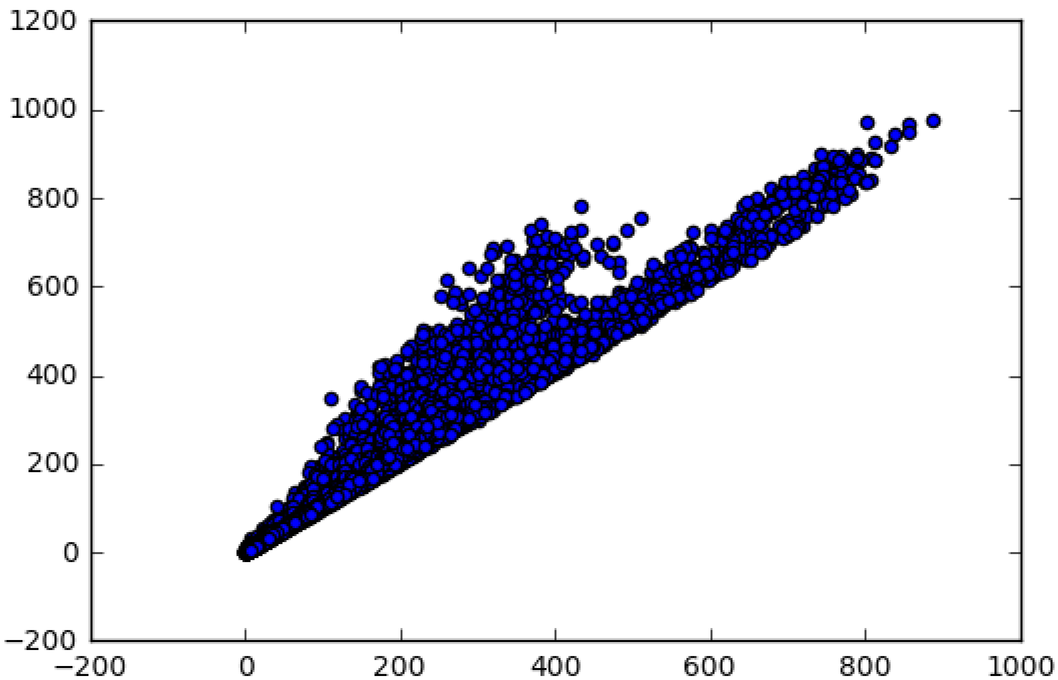
```
# The correlation coefficient between the temp and atemp variables is close to 1, i.e.
0.984948. The same situation for registered and total customer number.
# In order to avoid a multicollinearity situation, we will eliminate the atemp variabl
e from our dataframe
# season and month very high correlation 0.97
# the correlation is moderate also between the casual and total customers 0.69
# This corelation can also be seen in the figures below
```

```
plt.scatter(df.temp, df.atemp)
plt.show()
```



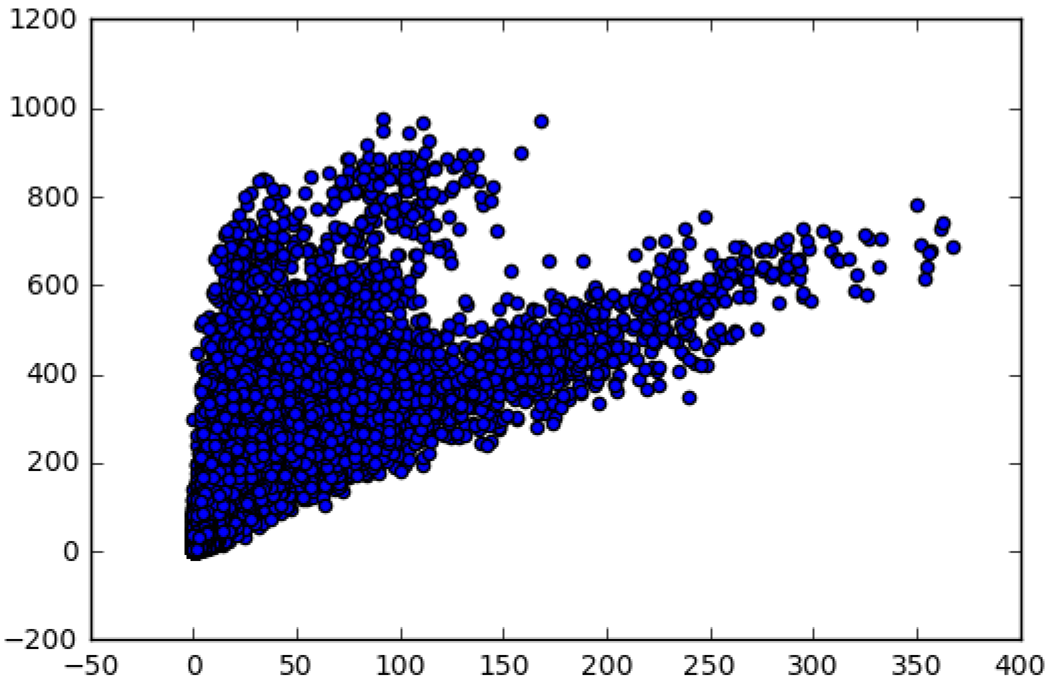
png

```
plt.scatter(df.registered, df.totalcustomers)
plt.show()
```



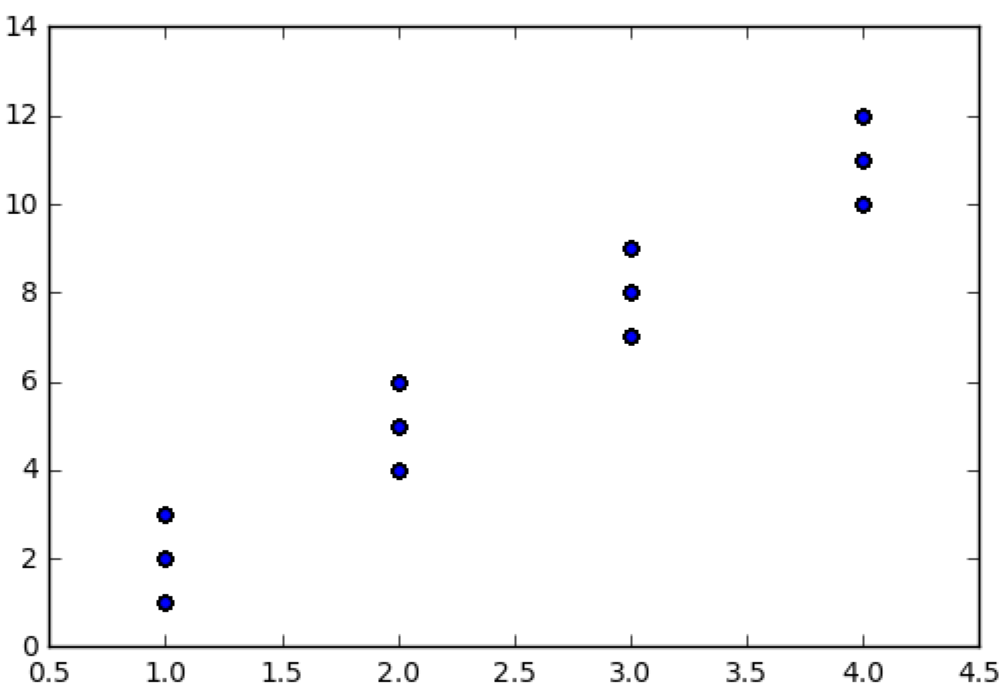
png

```
plt.scatter(df.casual, df.totalcustomers)
plt.show()
# moderate correlation
```



png

```
plt.scatter(df.season, df.month)
plt.show()
```



png

```
# Heatmap to see the correlation matrix on a different level

# import seaborn as sns
# sns.heatmap(df[['year', 'month', 'day','hour', 'holiday','workingday', 'weather','te
mp','humidity','windspeed','casual', 'totalcustomers']].corr(), annot=True)
# plt.show()
# There is an issue with matplotlib boxplot fliers not showing up when seaborn is impo
rted,
# even when fliers are explicitly enabled. In these conditions this heatmap, showing i
n a nice way the correlation coefficients is here commented.
# you can used, but pay attention that in the boxplot histogram, you will not be able
to see the fliers.
```

```
from scipy import stats
```

```
# Perform the Shapiro-Wilk test for normality.
# Some of our variable have a distribution that is could be normal.
# The Shapiro-Wills test is in fact a test for the existence of a normal distributuio
n in data: it tests the null hypothesis that data are normal.
# If the p-value is greater than the chosen alpha level, then the null hypothesis tha
t the data came from a normally distributed population cannot be rejected (e.g., for a
n alpha level of 0.05, a data set with a p-value of 0.02 rejects the null hypothesis t
hat the data are from a normally distributed population).[2] However, since the test i
s biased by sample size,[3] the test may be statistically significant from a normal di
stribution in any large samples. Thus a QâQ plot is required for verification in addit
ion to the test.
# The Shapiro-Wilk test tests the null hypothesis that the data was drawn from a norma
l distribution: if the p-value is less than the chosen alpha level (0.05 here),
# then the null hypothesis is rejected and there is evidence that the data tested are
not from a normally distributed population.
# The test show that the data are actually not normally distributed. The p-values are
extremely small.
newdf=df.drop('atemp',axis=1).drop('season',axis=1)
stats.shapiro(newdf.humidity)
```

```
C:\Users\ss_cr\Anaconda1\lib\site-packages\scipy\stats\morestats.py:1326: UserWarning:
p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")





(0.9822689294815063, 1.245496990918048e-34)
```

```
# because of this message, we also back our normality test with a qqplot
```

```
stats.shapiro(newdf.temp)
```

```
(0.9804092645645142, 4.47221826500091e-36)
```

```
stats.shapiro(newdf.windspeed)
```

```
(0.9587375521659851, 0.0)
```

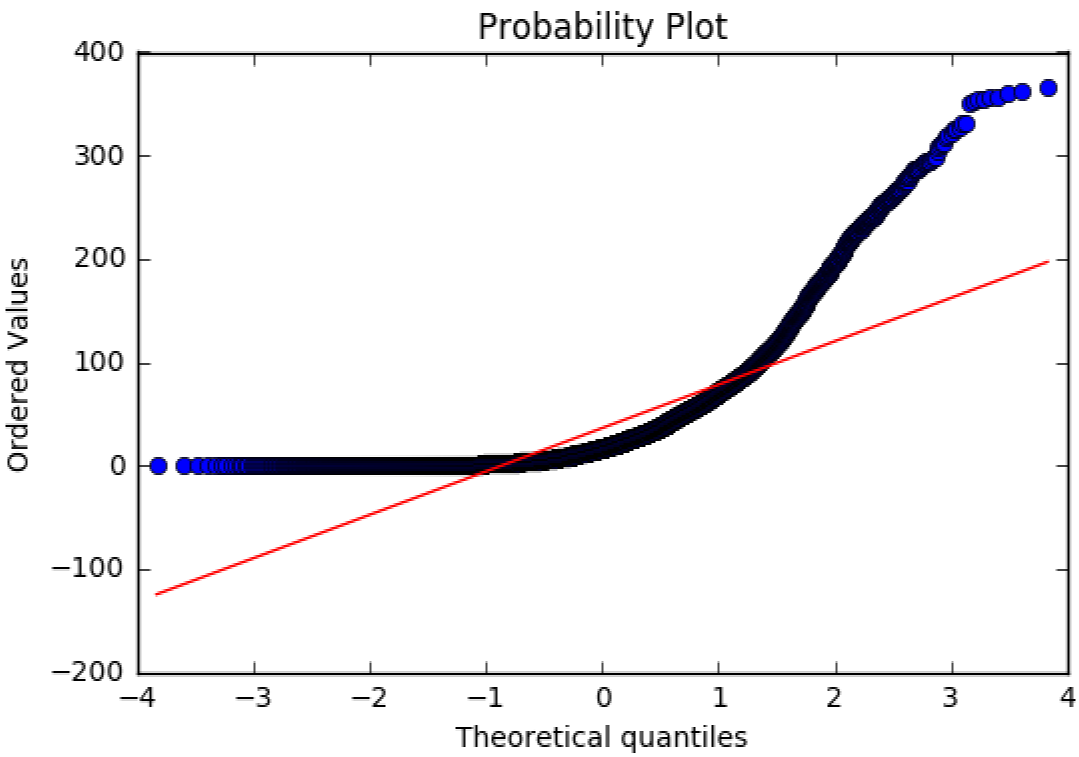
```
stats.shapiro(newdf.totalcustomers)
```

```
(0.8783667087554932, 0.0)
```

```
stats.shapiro(newdf.casual)
```

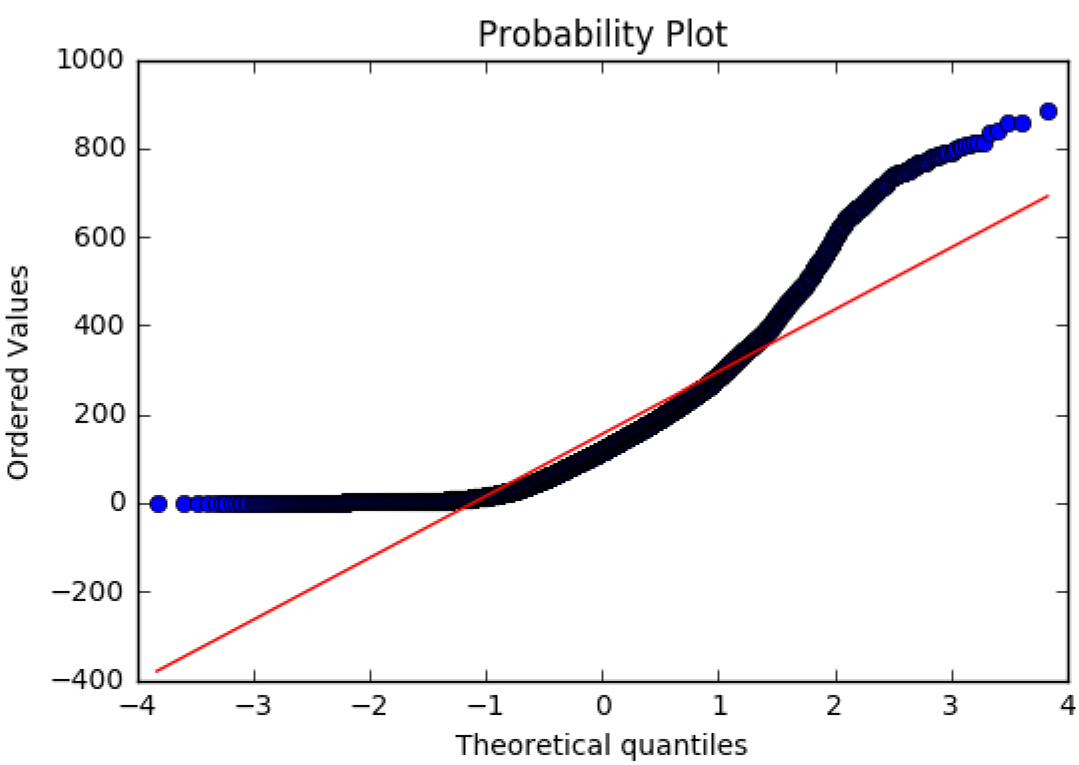
```
(0.7056357264518738, 0.0)
```

```
# the Shapiro Wills test is biased by sample size, our data based has more than 5000 e
ntries
# the test may be statistically significant from a normal distribution in any large sa
mples.
# Thus a QâQ plot is required for verification in addition to the test.
# test the normal distribution case with quantile - quantile plot with scipy
# Th thick blue line represents the distribution of the actual variable from the data
set
# and the straight red line is a mapping of what the normal distribution would look li
ke
import pylab
stats.probplot(newdf.casual, dist="norm", plot=pylab)
pylab.show()
# clearly not normally distributed
```



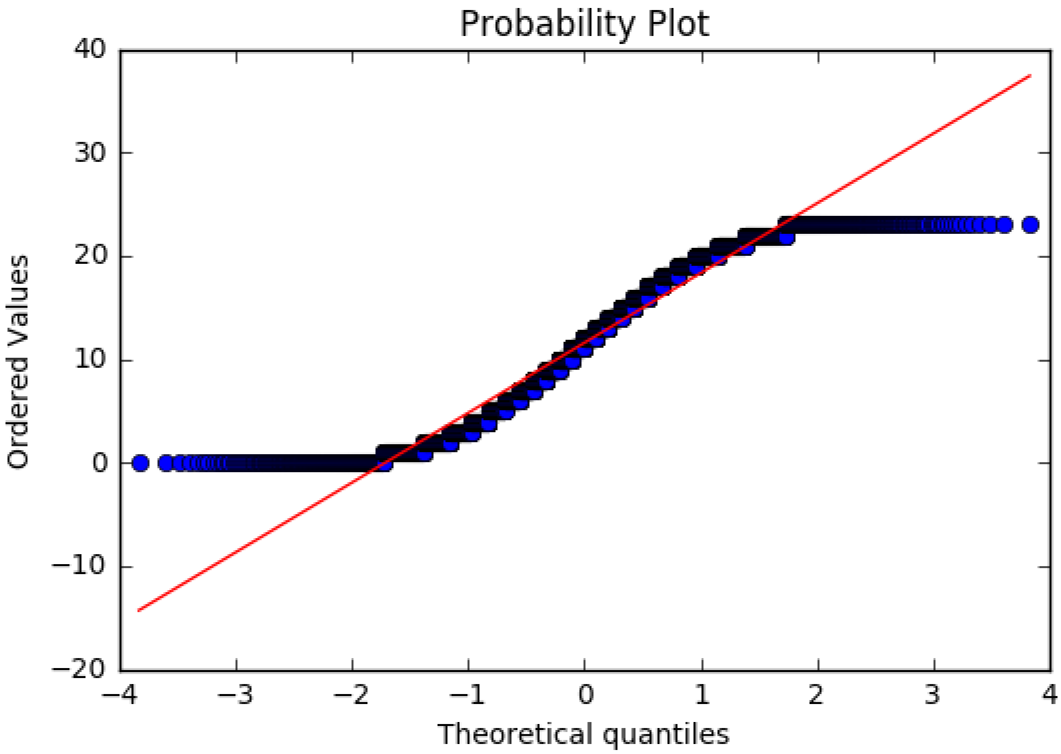
png

```
stats.probplot(newdf.registered, dist="norm", plot=pylab)
pylab.show()
```



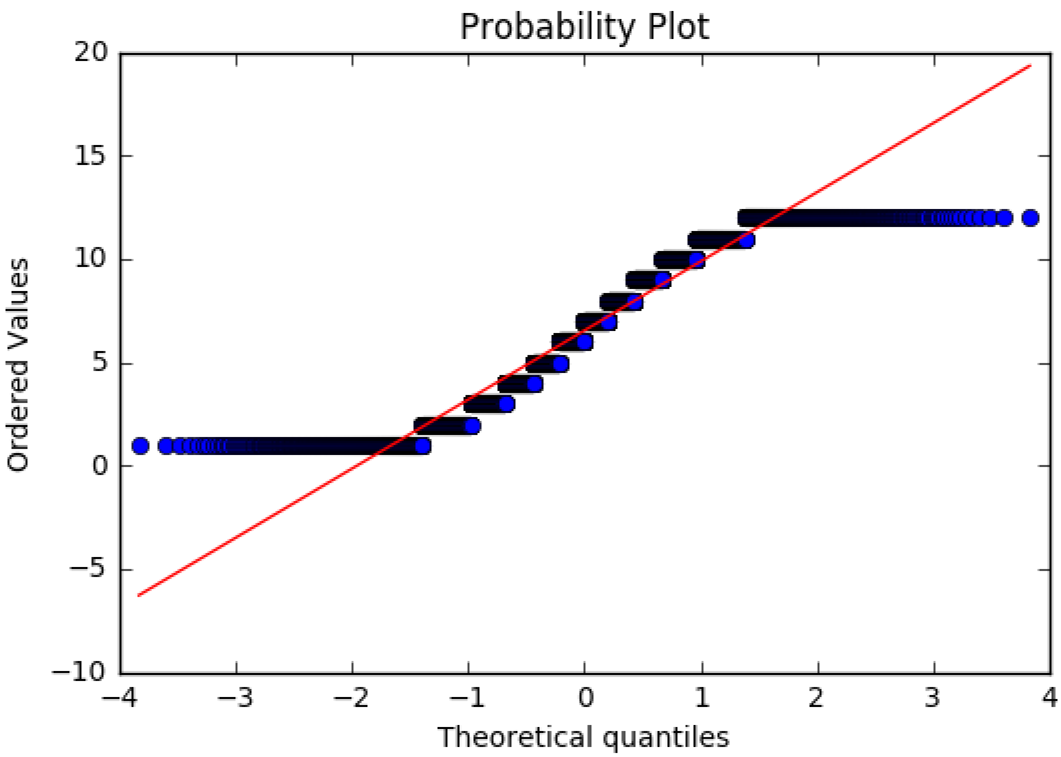
png

```
stats.probplot(newdf.hour, dist="norm", plot=pylab)
pylab.show()
```



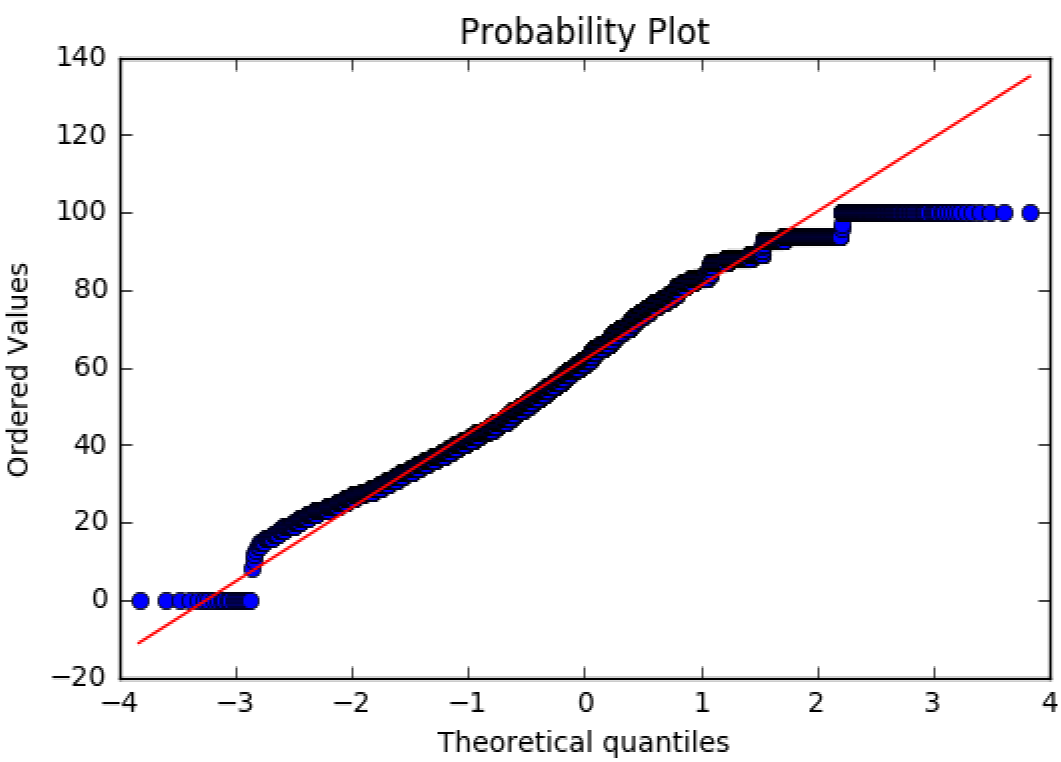
png

```
stats.probplot(newdf.month, dist="norm", plot=pylab)
pylab.show()
```



png

```
stats.probplot(newdf.humidity, dist="norm", plot=pylab)
pylab.show()
```

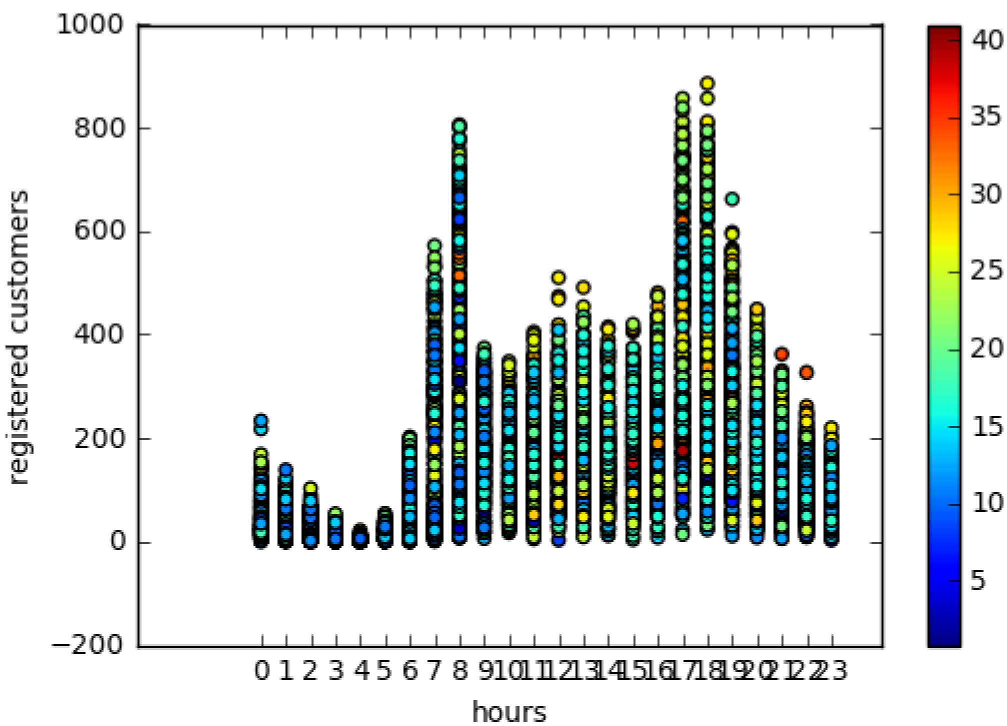


png

```
# It looks like the tested variables are not normally distributed
```

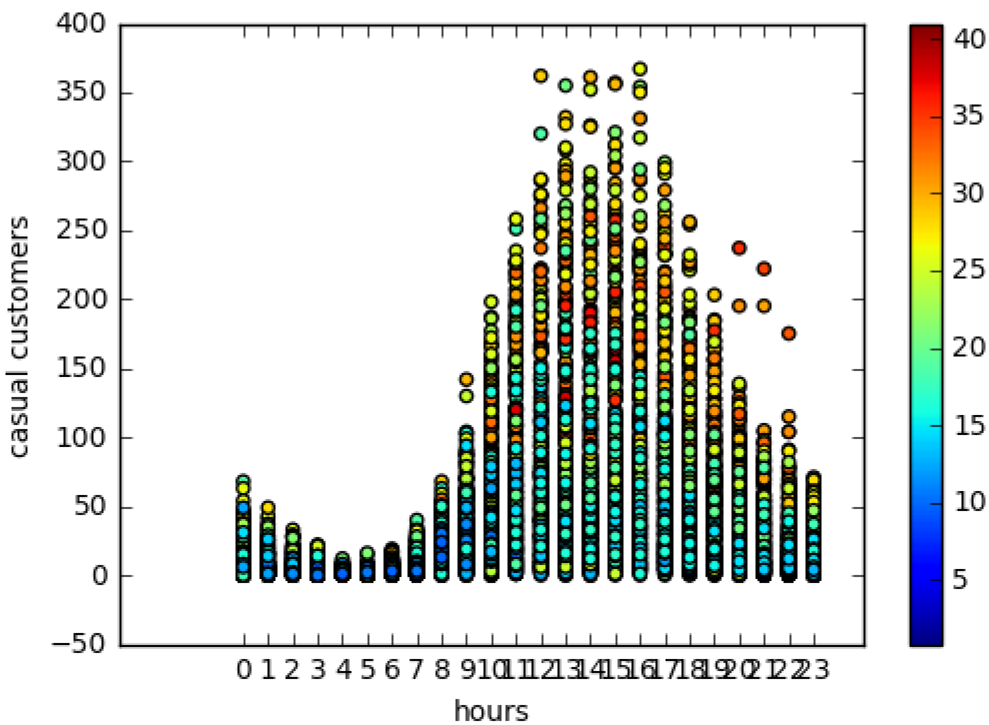
```
# Let's use some visualisation tools to explore our data
```

```
plt.scatter(newdf.hour, newdf.registered, c=df.temp,cmap='jet')
plt.colorbar()
ticks=np.arange(0,24,1)
labels = range(ticks.size)
plt.xticks(ticks,labels)
plt.xlabel('hours')
plt.ylabel('registered customers')
plt.show()
# It looks like the largest amount of registered customers is between 7 and 8, when pe
ople probably go to work
# and then again between 17.00 and 18.00, when people leave work.
# This diagram clearly makes a lot of sense.
```



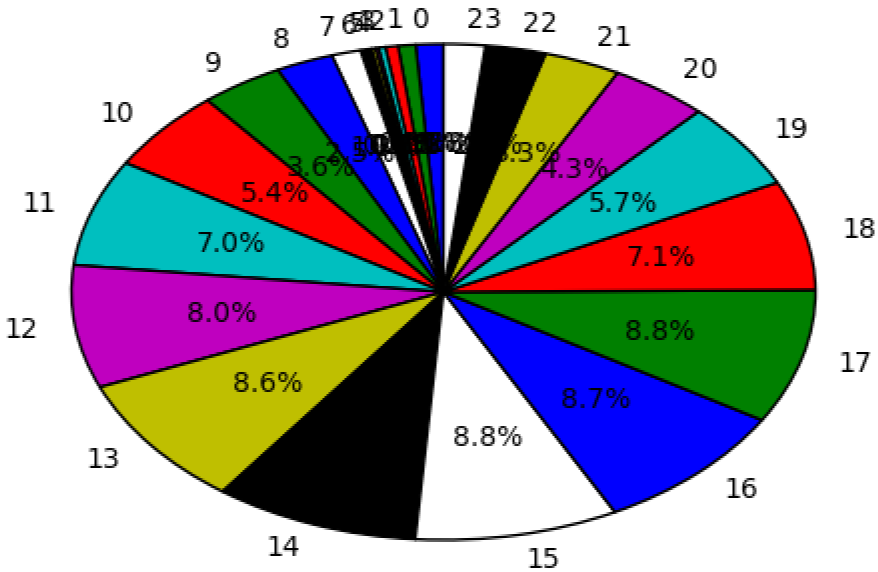
png

```
plt.scatter(newdf.hour, newdf.casual, c=df.temp, cmap='jet')
plt.colorbar()
ticks=np.arange(0,24,1)
labels = range(ticks.size)
plt.xticks(ticks,labels)
plt.xlabel('hours')
plt.ylabel('casual customers')
plt.show()
# It looks like the nb of casual customers (most of them probably turists) starts to i
ncrease around 9.00 a.m. and then decreases agin starting with 22.00.
# The number of casul customers is clearly larger for optimal wheater between 20 and 3
0 degrees.
```



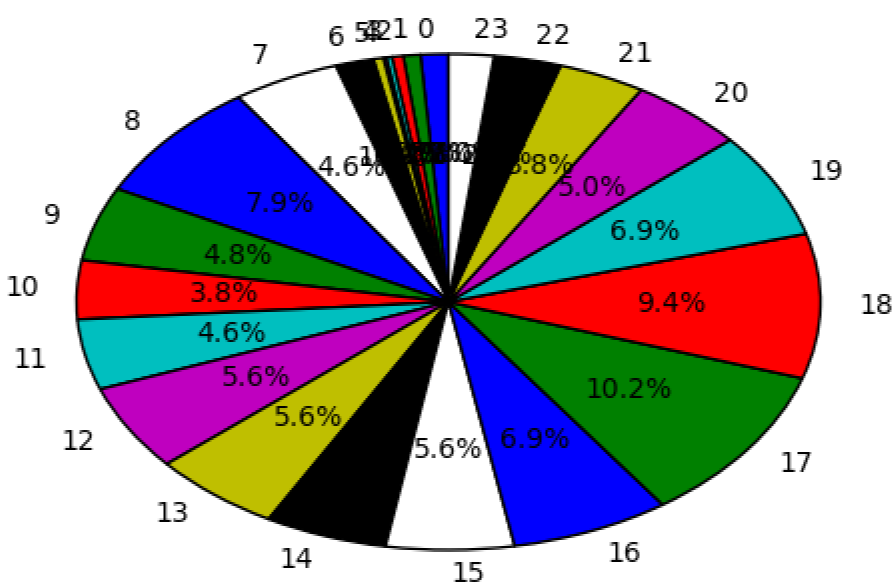
png

```
labels_hour=df.hour.unique()
values_hour=df.groupby('hour')['casual'].sum()

plt.pie(values_hour, labels=labels_hour,
                autopct='%1.1f%%', shadow=False, startangle=90)
plt.show()
# Same conclusion as above: the number of customers increase after 11a.m. and decreas
e after 6p.m.
```



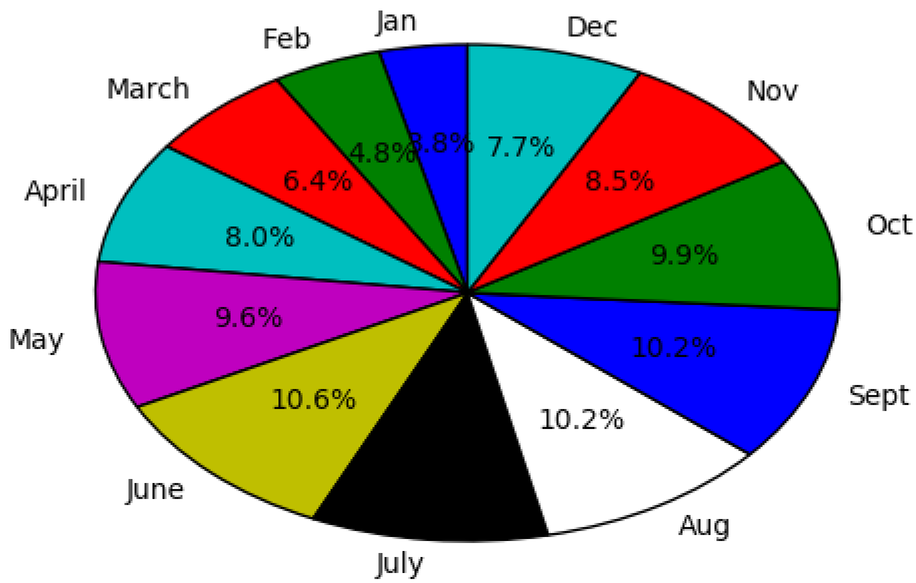
png

```
labels_hour=df.hour.unique()
values_hour=df.groupby('hour')['totalcustomers'].sum()

plt.pie(values_hour, labels=labels_hour,
                autopct='%1.1f%%', shadow=False, startangle=90)
plt.show()
# We regain the conclusion from figures above: totalcustomers/registered number is lar
ger around 8 a.m and 5-6 p.m.
```



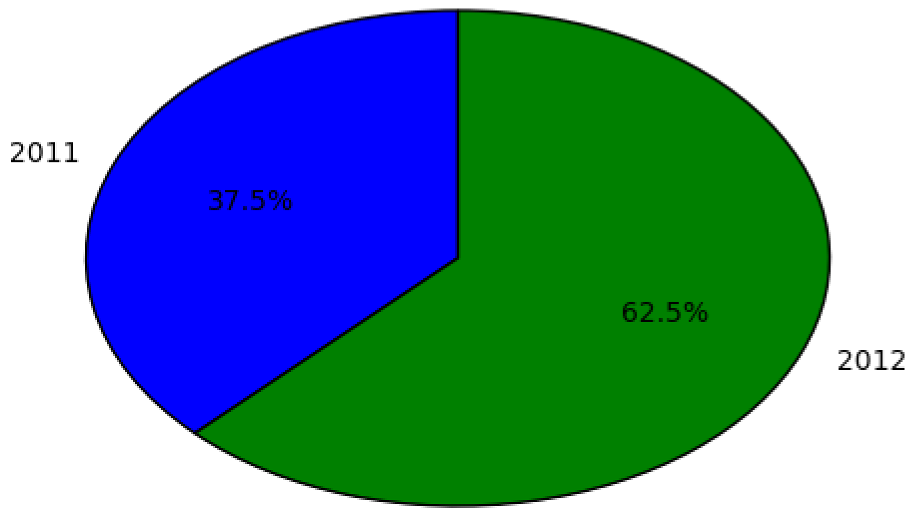
png

```
labels_month = ['Jan','Feb','March','April', 'May', 'June', 'July', 'Aug', 'Sept', 'Oc
t', 'Nov', 'Dec']
values_month = df.groupby('month')['totalcustomers'].sum()
plt.pie(values_month, labels=labels_month,
                autopct='%1.1f%%', shadow=False, startangle=90)
plt.show()
# The number of totalcustomers is larger between May and October
```



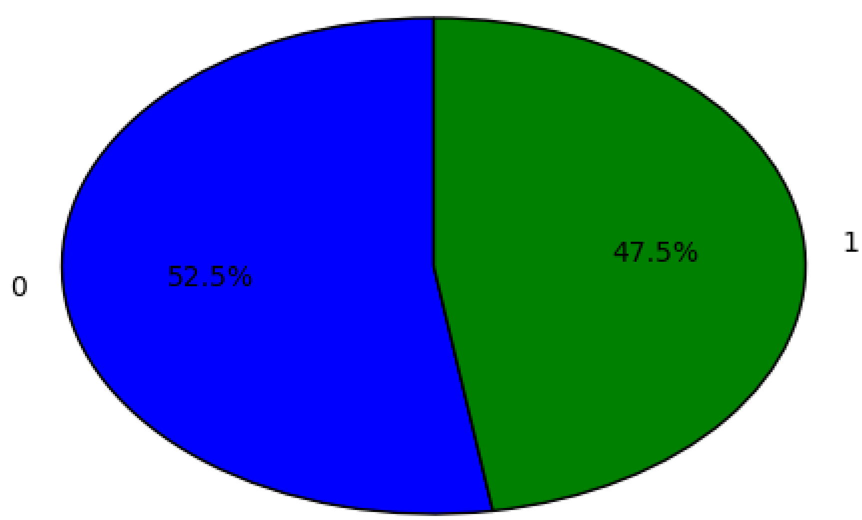
png

```
labels_year=df.year.unique()
values_year=df.groupby('year')['totalcustomers'].sum()

plt.pie(values_year, labels=labels_year,
                autopct='%1.1f%%', shadow=False, startangle=90)
plt.show()
# This shows that the number of bike demands is much larger for 2012
```



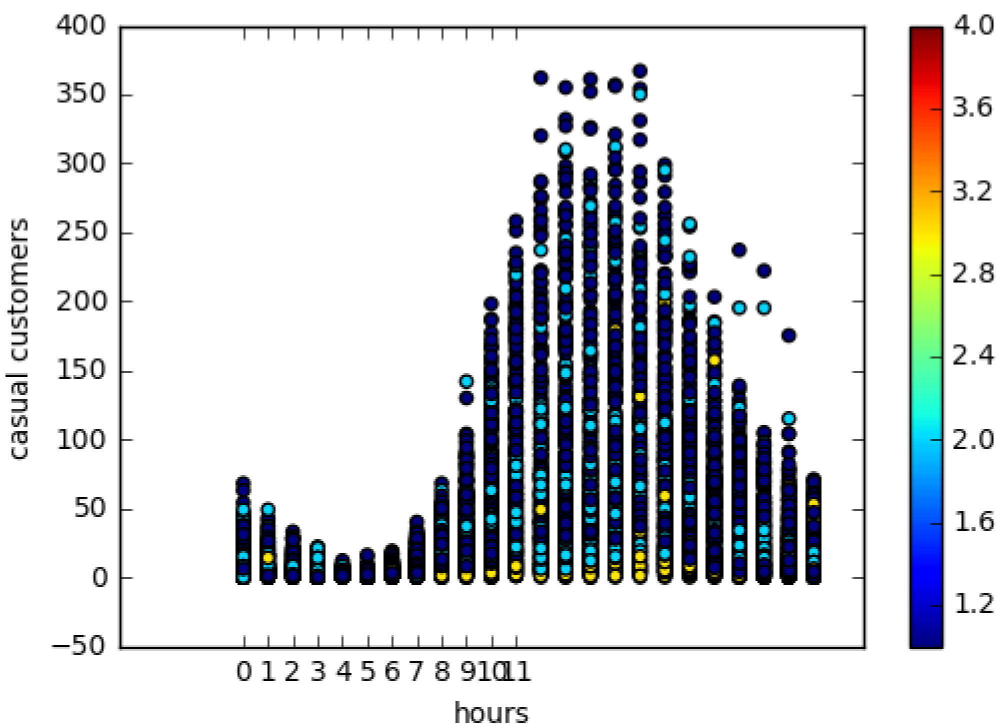
png

```
labels_day=df.workingday.unique()
values_day=df.groupby('workingday')['casual'].sum()
plt.pie(values_day, labels=labels_day,
                autopct='%1.1f%%', shadow=False, startangle=90)
plt.show()
# slight increase in the number of casual bikers over the weekends
```



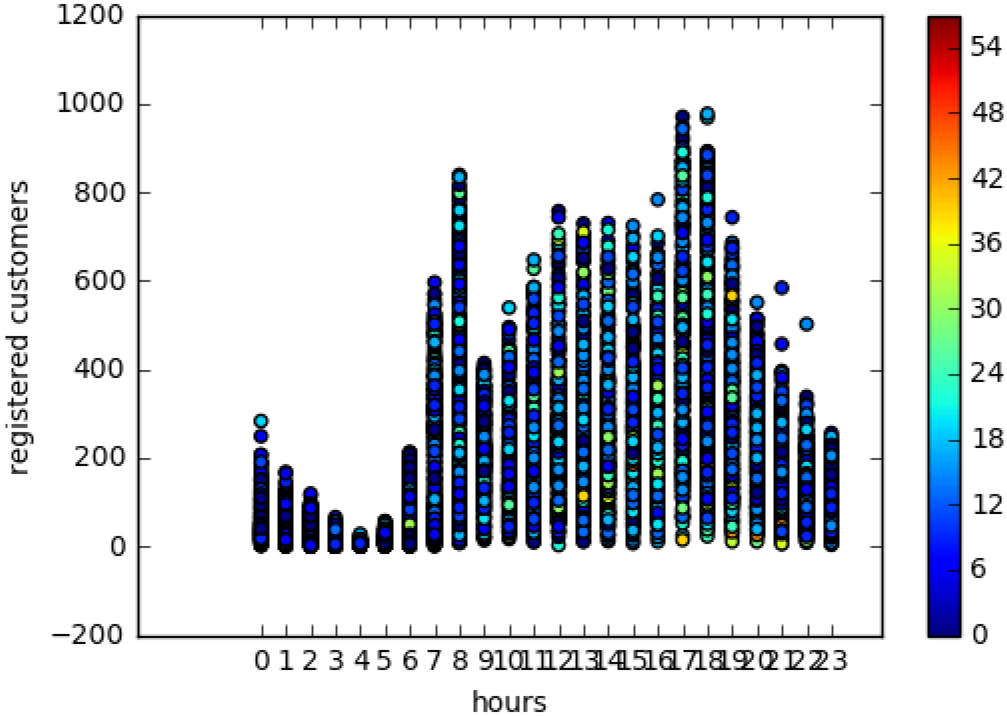
png

```
plt.scatter(newdf.hour, newdf.casual, c=df.weather)
plt.colorbar()
ticks=np.arange(0,12,1)
labels = range(ticks.size)
plt.xticks(ticks,labels)
plt.xlabel('hours')
plt.ylabel('casual customers')
plt.show()
# casual customers number has the same hourly distribution in terms of wheater as in t
erms of temperature
```



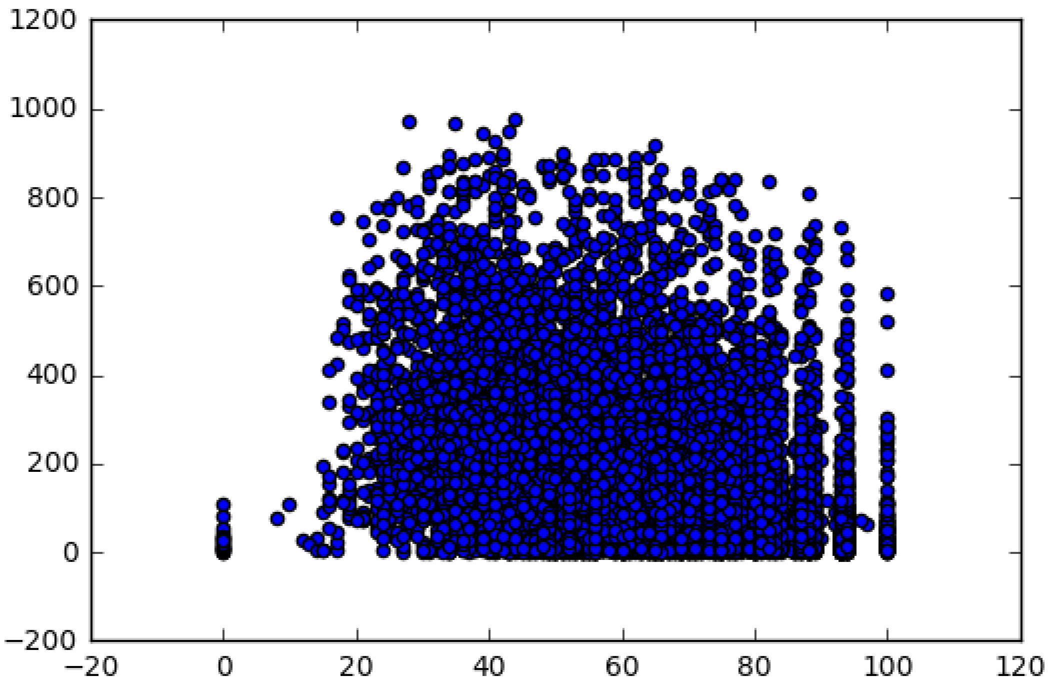
png

```
plt.scatter(newdf.hour, newdf.totalcustomers, c=df.windspeed, cmap='jet')
plt.colorbar()
ticks=np.arange(0,24,1)
labels = range(ticks.size)
plt.xticks(ticks,labels)
plt.xlabel('hours')
plt.ylabel('registered customers')
plt.show()
# the larger numb of customers use the bike for wind temperature up to 18km/h
```



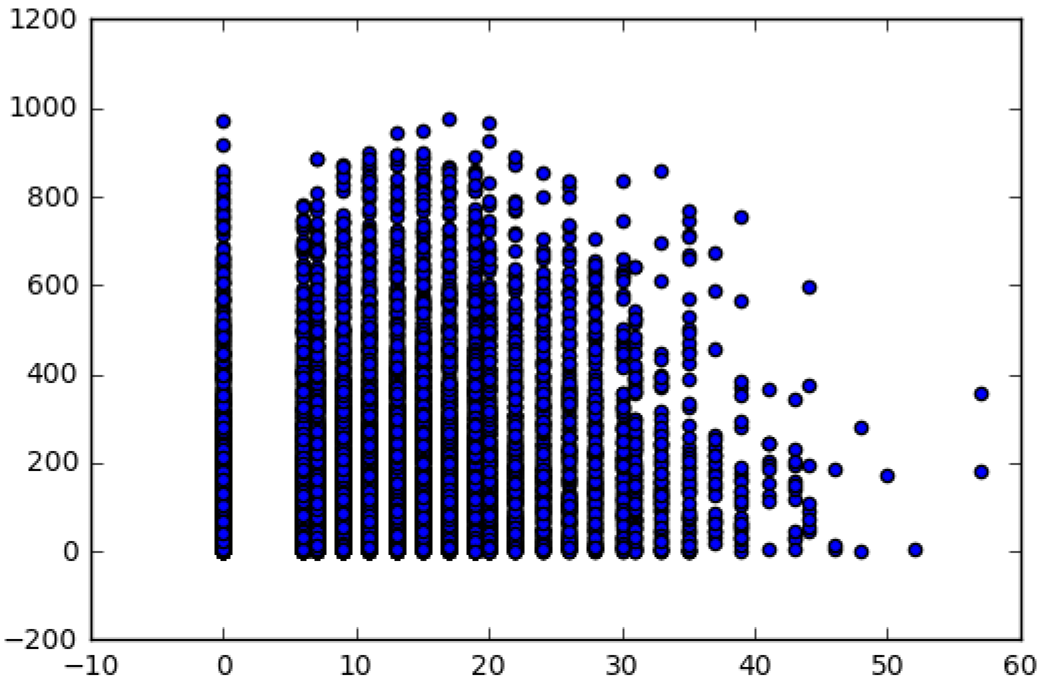
png

```
plt.scatter(newdf.humidity, newdf.totalcustomers)
plt.show()
```



png

```
plt.scatter(newdf.windspeed, newdf.totalcustomers)
plt.show()
# the wind speed has a certain influence on the nb of customers
```



png

```
#Bike sharing system - Part II  - Building the model ...
```

```

```

```
# Building the regression model. First we split the dataFrame into a training and test
case (60%, 40%).
# test_size=0.4 inside the function indicates the percentage of the data that should b
e held over for testing.
```

```
from sklearn.cross_validation import train_test_split
training, testing= train_test_split( newdf, test_size=0.4, random_state=1 )
print len(training)
print len(testing)
```

```
6531
4355


C:\Users\ss_cr\Anaconda1\lib\site-packages\sklearn\cross_validation.py:44: Deprecation
Warning: This module was deprecated in version 0.18 in favor of the model_selection mo
dule into which all the refactored classes and functions are moved. Also note that th
e interface of the new CV iterators are different from that of this module. This modul
e will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

```
# ...
```