Urban cycling

Sebastian Schwarz

Abstract

In the context of environmentally friendly transportation in cities, this project investigates the extent to which the daily use of bicycles can be predicted. The dataset for this project comes from the "Capital Bikeshare" program, which is based in Washington, D.C.. A regression analysis is performed, where the independent variables include weather data and calendar data. The dependent variable is the number of daily bicycle rentals, which are further divided into registered and casual users. Weather has a strong influence on the number of rentals, with perceived temperature showing the largest effect. Registered and casual users differ in particular for the variable "workingday", with the number of rentals increasing for registered users on a „workingday“, and the opposite effect being observed for casual users.

Motivation

In light of the current climate crisis, bicycles are gaining in importance as a low-emission means of transportation. Increased use of bicycles can also improve air quality in large cities, as well as help relieve traffic congestion. Compared to the conventional use of bicycles, bike sharing systems offer the advantage of data collection regarding travel time, departure and arrival locations. This data can be used to predict how many people will rent a bike on a certain day or at a certain time of day. These insights may be valuable for a stronger integration of bicycles into traffic.

Dataset

The dataset for this project comes from the "Capital Bikeshare" program, which is based in Washington, D.C.. Over a two-year period (2011 - 2012), the dataset contains the number of daily as well as hourly bike rentals. These are further broken down into rentals by registered and casual users. In addition, the dataset provides a variety of attributes, both in terms of weather data, and in terms of various calendar information, such as season, vacation, month. In total, the dataset contains 17389 rows and 16 columns.

Source: https://archive.ics.uci.edu/ml/machine-learning-databases/00275/

Data Preparation and Cleaning

- Some of the variables had already been transformed and had to be restored using the appropriate formula.
- Certain categorical variables were excluded from the analysis because they had a very high number of levels (in fact, taking these variables into account resulted in overfitting).
- Some variables were very highly correlated, so selection was also made (for example, pearson corr > 0.99 in the case of temperature and perceived temperature).
- All numerical variables were normalized and a one-hot encoding was used for all non-binary categorical variables.
- The data set did not contain any missing values.

Research Questions

How can the daily use of bicycles in cities be predicted?

- Can the number of daily bicycle rentals in Washington be predicted using weather data and calendar information?
- Which of these features are most important for predicting bike rentals in Washington D.C.?
- How are casual users different from registered users?

Methods

I am trying to predict the number of bike rentals, which is a numeric variable. Therefore, I use a regression analysis.

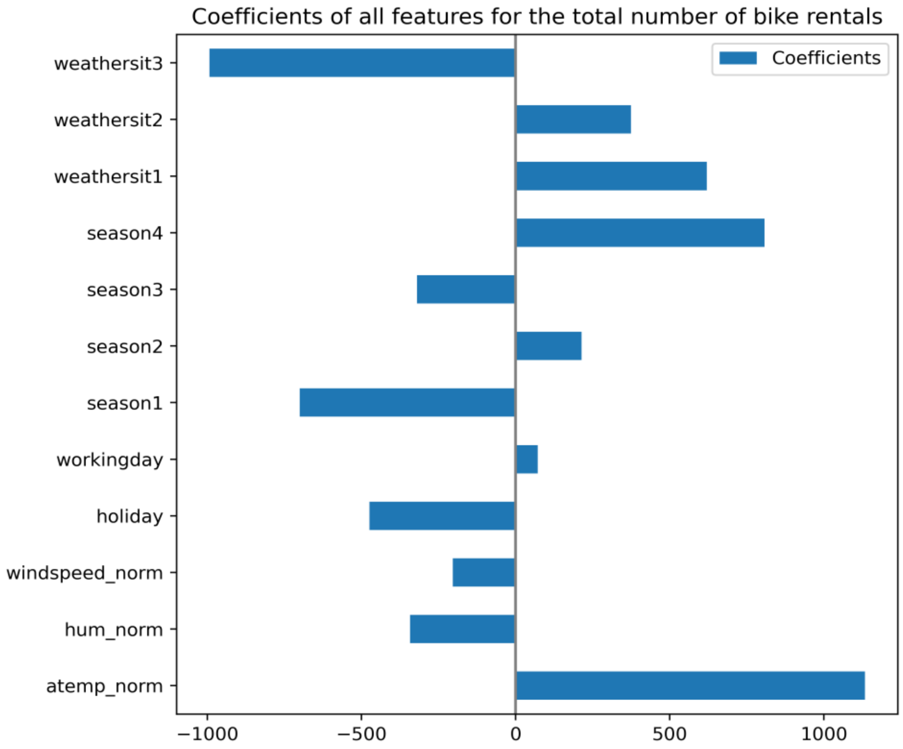
Independent variables: atemp (feeling temperature in C°), humidity, windspeed, holiday, workingday (neither weekend nor holiday), season (1: winter ... 4: fall), weathersituation (Classification of the weather in 4 levels. 1: Clear, Few clouds, 2: Mist + Cloudy, 3: Light Snow, Light Rain, 4: Heavy Rain, Snow, see link above for more details)

Dependent variables: total bike rentals, bike rentals of registered users, bike rentals of casual users

The Data set was divided into a training and a test set (25%).

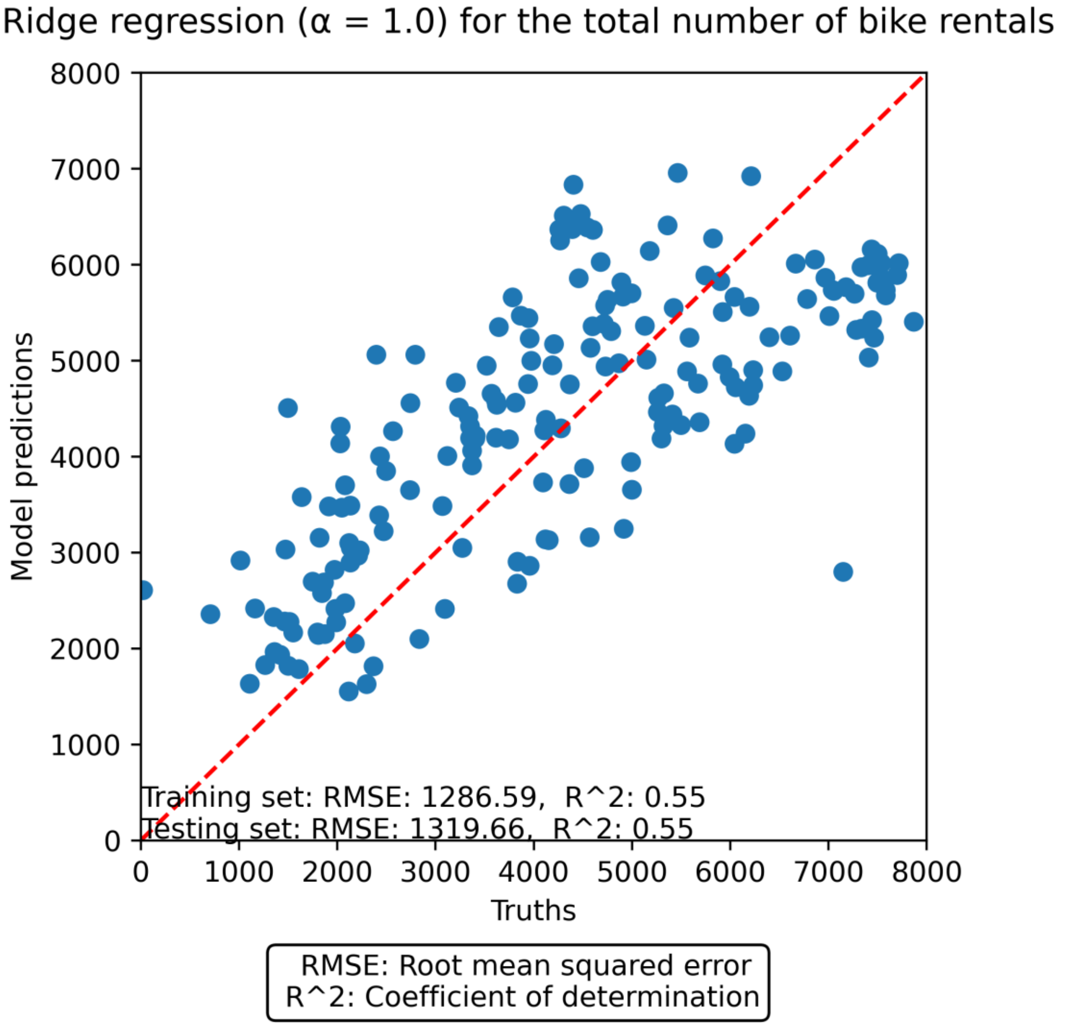
Total number of bike rentals

- From this plot we can inspect the most important features according to the absolute values of the coefficients.
- atemp seems to be the most important feature, where an increase in one unit (one std) leads to an increase of more than 1000 bike rentals.
- weathersit3 has the second highest coefficient. Here weather like Rain or Snow leads to a significant decrease of bike rentals.
- Season1 (winter) reduces and season 4 (fall) increases the number of bike rentals.
- Interestingely a holiday leads to a decrease of total bike rentals.



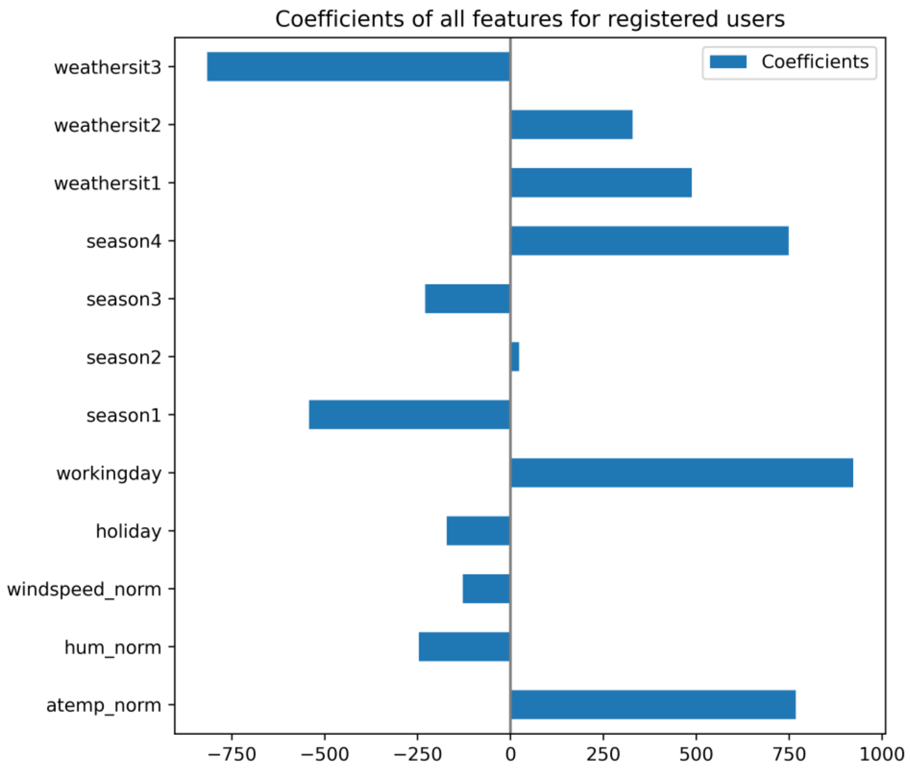
Note: weathersit4 was not present in the data set

- In this scatter plot the true values are plottet against the predictions of the test set.
- Overall a linear model seems to be apropriate to predict the number of bike rentals.
- However the RMSE value is quite high and the true value is usually more than 1000 counts above or below the prediction.
- The model seems to generalize well, as R^2 does not increase for the test set.



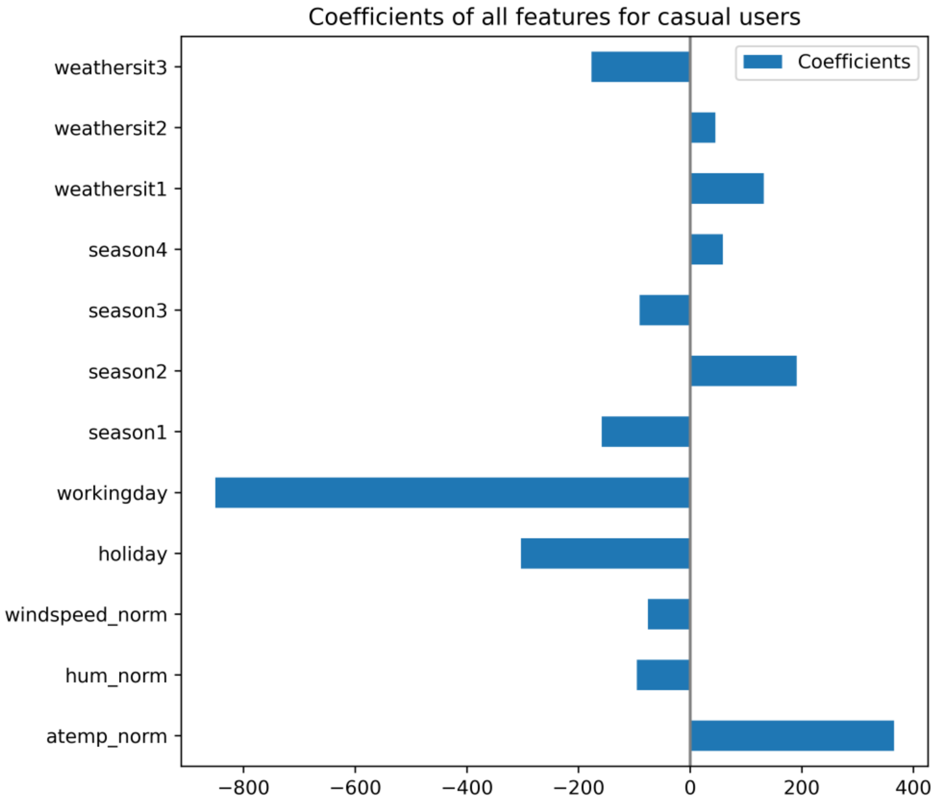
Registered and casual bike rentals

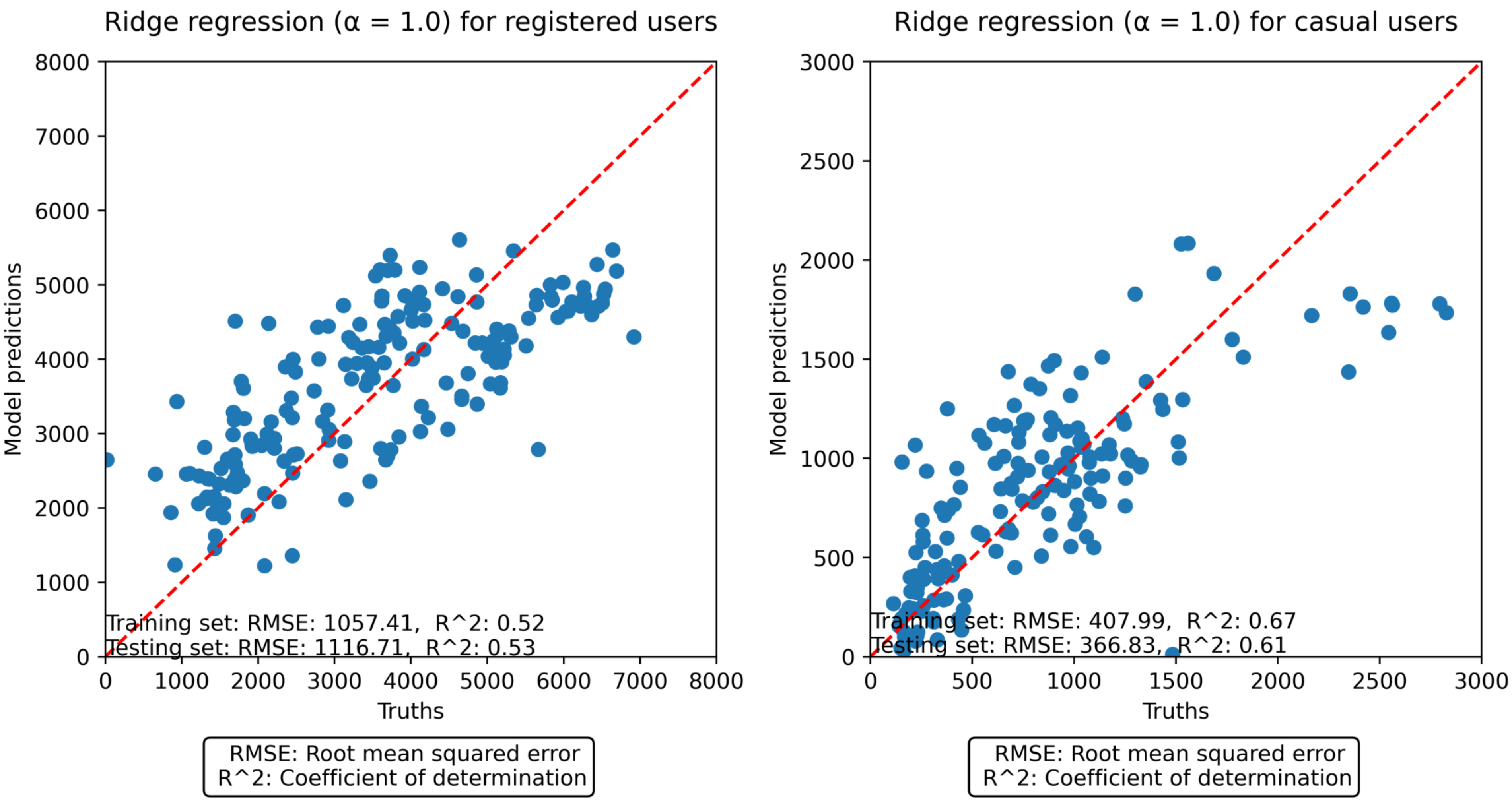
- For registered users, the pattern of the coefficients looks similar to that for the total number of users.
- However in contrast to the total number of bike rentals "workingday" has a large positive coefficient (922.52).



- For casual users, the coefficient for "workingday" is the largest. On working days, there are on average -850.84 fewer rentals than on non-working days.

- Weather (weathersit., season, atemp, etc.) is significant as for total number of users, but the coefficients are much smaller compared to "workingday".




Ridge regression (α = 1.0) for registered users
Model predictions
Truths
0
1000
2000
3000
4000
5000
6000
7000
8000
Training set: RMSE: 1057.41, R^2: 0.52
Testing set: RMSE: 1116.71, R^2: 0.53
RMSE: Root mean squared error
R^2: Coefficient of determination
Ridge regression (α = 1.0) for casual users
Model predictions
Truths
0
500
1000
1500
2000
2500
3000
Training set: RMSE: 407.99, R^2: 0.67
Testing set: RMSE: 366.83, R^2: 0.61
RMSE: Root mean squared error
R^2: Coefficient of determination

- For casual users, the coefficient of determination for the test set is higher than for registered users. However, for a high number of rentals, the model does not seem to make good predictions.

- Compared to the total number as well as casual users, the coefficient of determination is lowest for registered users in the test set. Nevertheless, overall a linear model seems to be suitable to predict the number of bicycle rentals.

Limitations

- The data set contains data from only one bike sharing provider. In addition, the data was collected in Washington, D.C. Generalization to other cities should be done with caution.

- The number of features is rather small and limited in content. Demographic (age, gender) or geographic (terrain, bike trails, etc.) features would also be a good idea.

Conclusions

- As expected, the weather plays an important role. On average, the number of users increases with rising temperatures. However, a decrease can be expected again for very high temperatures. For all user groups, the number of rentals decreases especially in winter and increases in fall and spring. For the "weathersituation" feature, a similar pattern emerges for all user groups, with the number of users increasing for "wethaersituation" 1-2 and decreasing sharply for "weathersituation" 3.

- The comparison of casual and registered users is particularly interesting. The number of casual users decreases sharply on a working day. For registered users, the opposite effect can be observed. A possible interpretation is that registered users use the bicycle for everyday trips, such as going to work or shopping. Casual users, on the other hand, may use the bicycle more for leisure activities for which there is no time during a normal working day.

- All in all, it is possible to predict the number of rentals with a linear model and the given features. It seems to be useful to analyze user groups separately, as this can improve the predictions. The data can be used for a stronger integration of bicycles into traffic, as it provides information about the potential but also the limitations of bicycle usage. In addition to a temporal perspective, a geographic perspective would also be important to identify areas with an increased need for bike lanes. The performance of the models can certainly be improved by using additional features.

Acknowledgements

The dataset for this project comes from the "Capital Bikeshare" program, which is based in Washington, D.C.. I have not received any feedback regarding my project yet.

References

https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset#

https://scikit-learn.org/stable/

This Project contains modified code from the scikit online documentation.

Bike_Sharing_Notebook

March 24, 2022

[44]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
import seaborn as sns
from sklearn.metrics import mean_squared_error
from math import sqrt
```

[45]:
```
df = pd.read_csv('./Bike-Sharing-Dataset/day.csv', sep=',')
```

[46]:
```
# rescaling taransformed features to get a feel for real values. Also␣
↪normalization can be later performed by Standard Scalar
```

[47]:
```
df['temp_org'] = df['temp']*(39-(-8)) + (-8)
df['atemp_org'] = df['atemp']*(50-(-16)) + (-16)
df['windspeed_org'] = df['windspeed']*100
df['hum_org'] = df['hum']*67
```

[48]:
```
# Scale the numerical features using Standard Scaler
```

[49]:
```
data_num = df[['atemp_org', 'temp_org', 'hum_org', 'windspeed_org']].copy()
```

[50]:
```
X_num = StandardScaler().fit_transform(data_num)
```

[51]:
```
df['atemp_norm'] = X_num[:,0]
df['temp_norm'] = X_num[:,1]
df['hum_norm'] = X_num[:,2]
df['windspeed_norm'] = X_num[:,3]
```

[52]:
```
df.head()
```

[52]:
```
   instant      dteday  season  yr  mnth  holiday  weekday  workingday  \
0        1  2011-01-01       1   0     1        0        6           0
```

```
1          2  2011-01-02        1   0     1         0         0            0
2          3  2011-01-03        1   0     1         0         1            1
3          4  2011-01-04        1   0     1         0         2            1
4          5  2011-01-05        1   0     1         0         3            1

   weathersit      temp  …  registered   cnt  temp_org  atemp_org  \
0           2  0.344167  …         654   985  8.175849   7.999250
1           2  0.363478  …         670   801  9.083466   7.346774
2           1  0.196364  …        1229  1349  1.229108  -3.499270
3           1  0.200000  …        1454  1562  1.400000  -1.999948
4           1  0.226957  …        1518  1600  2.666979  -0.868180

   windspeed_org    hum_org  atemp_norm  temp_norm  hum_norm  windspeed_norm
0        16.0446  53.990811   -0.679946  -0.826662  1.250171       -0.387892
1        24.8539  46.637829   -0.740652  -0.721095  0.479113        0.749602
2        24.8309  29.297291   -1.749767  -1.634657 -1.339274        0.746632
3        16.0296  39.559145   -1.610270  -1.614780 -0.263182       -0.389829
4        18.6900  29.276119   -1.504971  -1.467414 -1.341494       -0.046307

[5 rows x 24 columns]
```

[53]:
```python
df.corr()
```

[53]:
```
                 instant    season        yr      mnth   holiday   weekday  \
instant         1.000000  0.412224  0.866025  0.496702  0.016145 -0.000016
season          0.412224  1.000000 -0.001844  0.831440 -0.010537 -0.003080
yr              0.866025 -0.001844  1.000000 -0.001792  0.007954 -0.005461
mnth            0.496702  0.831440 -0.001792  1.000000  0.019191  0.009509
holiday         0.016145 -0.010537  0.007954  0.019191  1.000000 -0.101960
weekday        -0.000016 -0.003080 -0.005461  0.009509 -0.101960  1.000000
workingday     -0.004337  0.012485 -0.002013 -0.005901 -0.253023  0.035790
weathersit     -0.021477  0.019211 -0.048727  0.043528 -0.034627  0.031087
temp            0.150580  0.334315  0.047604  0.220205 -0.028556 -0.000170
atemp           0.152638  0.342876  0.046106  0.227459 -0.032507 -0.007537
hum             0.016375  0.205445 -0.110651  0.222204 -0.015937 -0.052232
windspeed      -0.112620 -0.229046 -0.011817 -0.207502  0.006292  0.014282
casual          0.275255  0.210399  0.248546  0.123006  0.054274  0.059923
registered      0.659623  0.411623  0.594248  0.293488 -0.108745  0.057367
cnt             0.628830  0.406100  0.566710  0.279977 -0.068348  0.067443
temp_org        0.150580  0.334315  0.047604  0.220205 -0.028556 -0.000170
atemp_org       0.152638  0.342876  0.046106  0.227459 -0.032507 -0.007537
windspeed_org  -0.112620 -0.229046 -0.011817 -0.207502  0.006292  0.014282
hum_org         0.016375  0.205445 -0.110651  0.222204 -0.015937 -0.052232
atemp_norm      0.152638  0.342876  0.046106  0.227459 -0.032507 -0.007537
temp_norm       0.150580  0.334315  0.047604  0.220205 -0.028556 -0.000170
hum_norm        0.016375  0.205445 -0.110651  0.222204 -0.015937 -0.052232
windspeed_norm -0.112620 -0.229046 -0.011817 -0.207502  0.006292  0.014282
```

	workingday	weathersit	temp	atemp	…	registered	\
instant	-0.004337	-0.021477	0.150580	0.152638	…	0.659623	
season	0.012485	0.019211	0.334315	0.342876	…	0.411623	
yr	-0.002013	-0.048727	0.047604	0.046106	…	0.594248	
mnth	-0.005901	0.043528	0.220205	0.227459	…	0.293488	
holiday	-0.253023	-0.034627	-0.028556	-0.032507	…	-0.108745	
weekday	0.035790	0.031087	-0.000170	-0.007537	…	0.057367	
workingday	1.000000	0.061200	0.052660	0.052182	…	0.303907	
weathersit	0.061200	1.000000	-0.120602	-0.121583	…	-0.260388	
temp	0.052660	-0.120602	1.000000	0.991702	…	0.540012	
atemp	0.052182	-0.121583	0.991702	1.000000	…	0.544192	
hum	0.024327	0.591045	0.126963	0.139988	…	-0.091089	
windspeed	-0.018796	0.039511	-0.157944	-0.183643	…	-0.217449	
casual	-0.518044	-0.247353	0.543285	0.543864	…	0.395282	
registered	0.303907	-0.260388	0.540012	0.544192	…	1.000000	
cnt	0.061156	-0.297391	0.627494	0.631066	…	0.945517	
temp_org	0.052660	-0.120602	1.000000	0.991702	…	0.540012	
atemp_org	0.052182	-0.121583	0.991702	1.000000	…	0.544192	
windspeed_org	-0.018796	0.039511	-0.157944	-0.183643	…	-0.217449	
hum_org	0.024327	0.591045	0.126963	0.139988	…	-0.091089	
atemp_norm	0.052182	-0.121583	0.991702	1.000000	…	0.544192	
temp_norm	0.052660	-0.120602	1.000000	0.991702	…	0.540012	
hum_norm	0.024327	0.591045	0.126963	0.139988	…	-0.091089	
windspeed_norm	-0.018796	0.039511	-0.157944	-0.183643	…	-0.217449	

	cnt	temp_org	atemp_org	windspeed_org	hum_org	\
instant	0.628830	0.150580	0.152638	-0.112620	0.016375	
season	0.406100	0.334315	0.342876	-0.229046	0.205445	
yr	0.566710	0.047604	0.046106	-0.011817	-0.110651	
mnth	0.279977	0.220205	0.227459	-0.207502	0.222204	
holiday	-0.068348	-0.028556	-0.032507	0.006292	-0.015937	
weekday	0.067443	-0.000170	-0.007537	0.014282	-0.052232	
workingday	0.061156	0.052660	0.052182	-0.018796	0.024327	
weathersit	-0.297391	-0.120602	-0.121583	0.039511	0.591045	
temp	0.627494	1.000000	0.991702	-0.157944	0.126963	
atemp	0.631066	0.991702	1.000000	-0.183643	0.139988	
hum	-0.100659	0.126963	0.139988	-0.248489	1.000000	
windspeed	-0.234545	-0.157944	-0.183643	1.000000	-0.248489	
casual	0.672804	0.543285	0.543864	-0.167613	-0.077008	
registered	0.945517	0.540012	0.544192	-0.217449	-0.091089	
cnt	1.000000	0.627494	0.631066	-0.234545	-0.100659	
temp_org	0.627494	1.000000	0.991702	-0.157944	0.126963	
atemp_org	0.631066	0.991702	1.000000	-0.183643	0.139988	
windspeed_org	-0.234545	-0.157944	-0.183643	1.000000	-0.248489	
hum_org	-0.100659	0.126963	0.139988	-0.248489	1.000000	
atemp_norm	0.631066	0.991702	1.000000	-0.183643	0.139988	

```
temp_norm       0.627494  1.000000   0.991702      -0.157944  0.126963
hum_norm       -0.100659  0.126963   0.139988      -0.248489  1.000000
windspeed_norm -0.234545 -0.157944  -0.183643       1.000000 -0.248489

                atemp_norm  temp_norm  hum_norm  windspeed_norm
instant           0.152638   0.150580  0.016375       -0.112620
season            0.342876   0.334315  0.205445       -0.229046
yr                0.046106   0.047604 -0.110651       -0.011817
mnth              0.227459   0.220205  0.222204       -0.207502
holiday          -0.032507  -0.028556 -0.015937        0.006292
weekday          -0.007537  -0.000170 -0.052232        0.014282
workingday        0.052182   0.052660  0.024327       -0.018796
weathersit       -0.121583  -0.120602  0.591045        0.039511
temp              0.991702   1.000000  0.126963       -0.157944
atemp             1.000000   0.991702  0.139988       -0.183643
hum               0.139988   0.126963  1.000000       -0.248489
windspeed        -0.183643  -0.157944 -0.248489        1.000000
casual            0.543864   0.543285 -0.077008       -0.167613
registered        0.544192   0.540012 -0.091089       -0.217449
cnt               0.631066   0.627494 -0.100659       -0.234545
temp_org          0.991702   1.000000  0.126963       -0.157944
atemp_org         1.000000   0.991702  0.139988       -0.183643
windspeed_org    -0.183643  -0.157944 -0.248489        1.000000
hum_org           0.139988   0.126963  1.000000       -0.248489
atemp_norm        1.000000   0.991702  0.139988       -0.183643
temp_norm         0.991702   1.000000  0.126963       -0.157944
hum_norm          0.139988   0.126963  1.000000       -0.248489
windspeed_norm   -0.183643  -0.157944 -0.248489        1.000000

[23 rows x 23 columns]
```

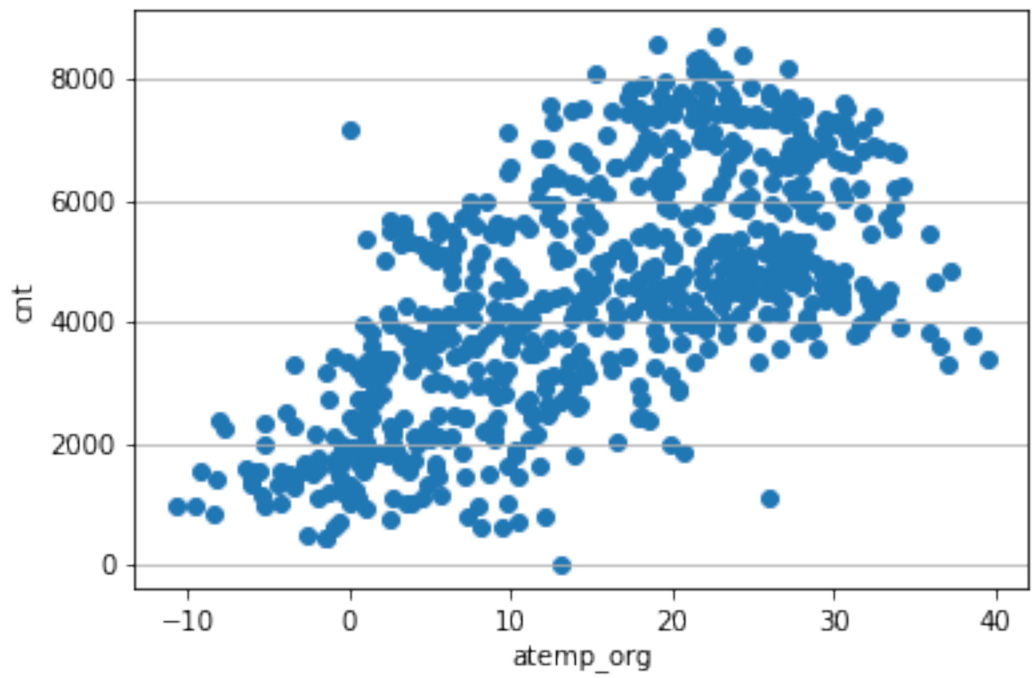
[54]:
```
df.isna().sum()
```

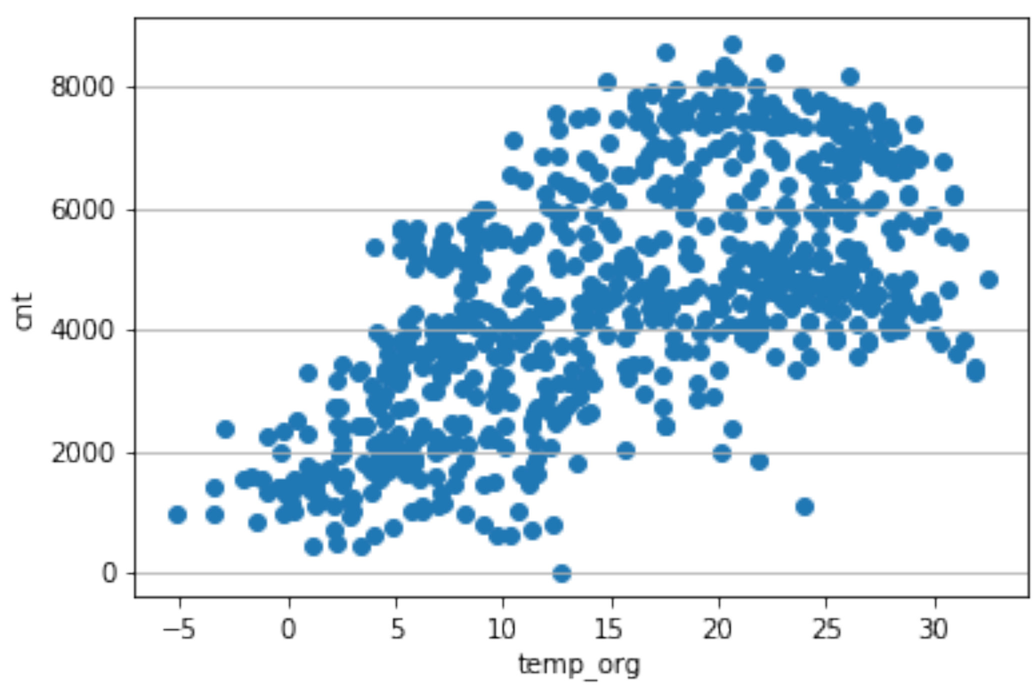
[54]:
```
instant           0
dteday            0
season            0
yr                0
mnth              0
holiday           0
weekday           0
workingday        0
weathersit        0
temp              0
atemp             0
hum               0
windspeed         0
casual            0
```

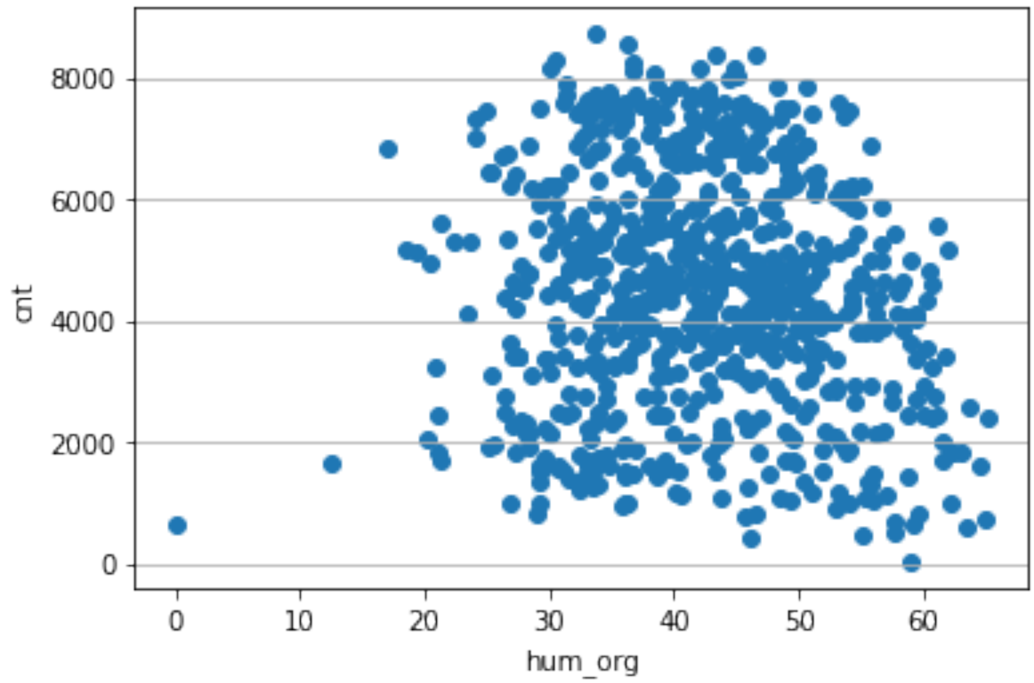
```
registered        0
cnt               0
temp_org          0
atemp_org         0
windspeed_org     0
hum_org           0
atemp_norm        0
temp_norm         0
hum_norm          0
windspeed_norm    0
dtype: int64
```

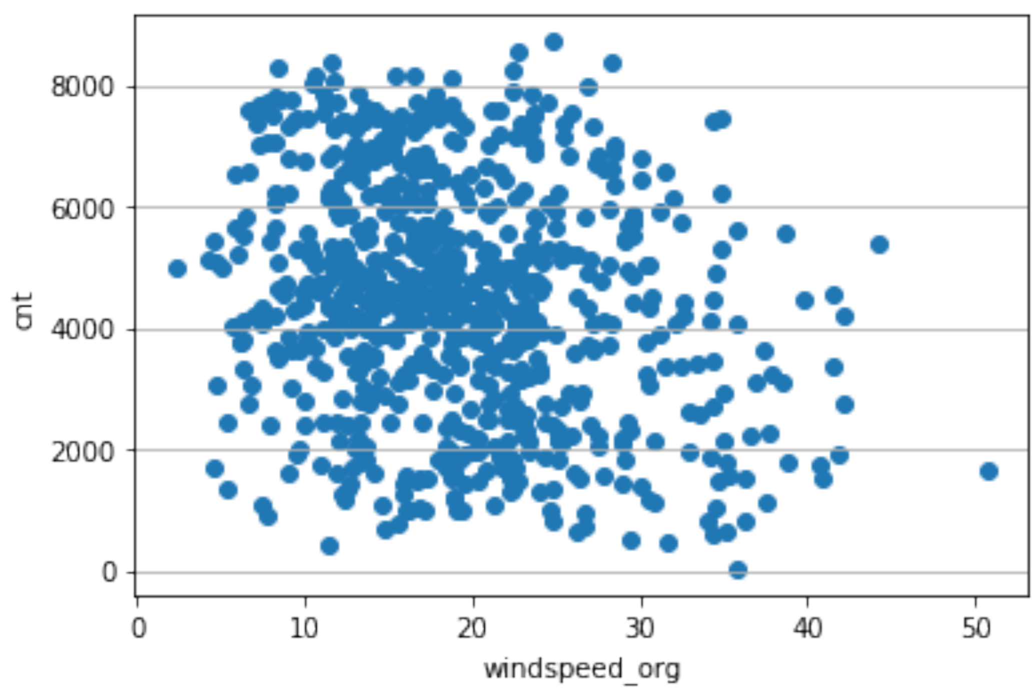
[55]:

```
%matplotlib inline


features = ['atemp_org', 'temp_org', 'hum_org', 'windspeed_org']

for f in features:
    fig, axis = plt.subplots()
    # Grid lines, Xticks, Xlabel, Ylabel

    axis.yaxis.grid(True)
    X = df[f]
    Y = df['cnt']
    axis.set_xlabel(f,fontsize=10)
    axis.set_ylabel('cnt',fontsize=10)
    axis.scatter(X, Y)
    plt.show()
```


8000
6000
4000
2000
0
cnt
−10
0
10
20
30
40
atemp_org


8000
6000
4000
2000
0
cnt
−5
0
5
10
15
20
25
30
temp_org





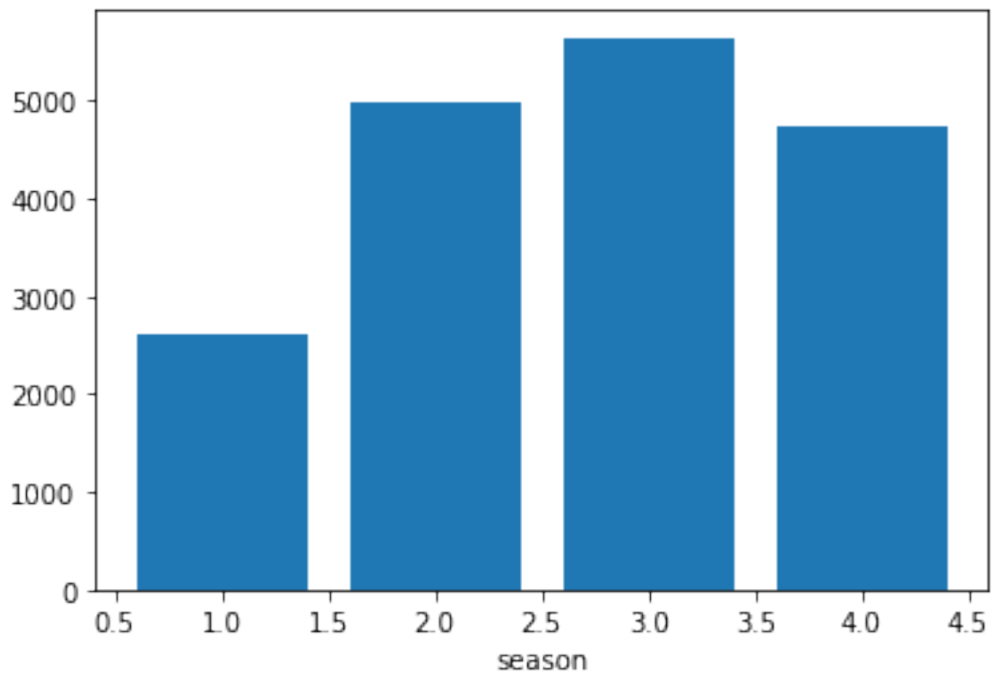
[56]:
```
categorical_features = ['season', 'mnth', 'holiday', 'weekday',
        'workingday', 'weathersit']
```

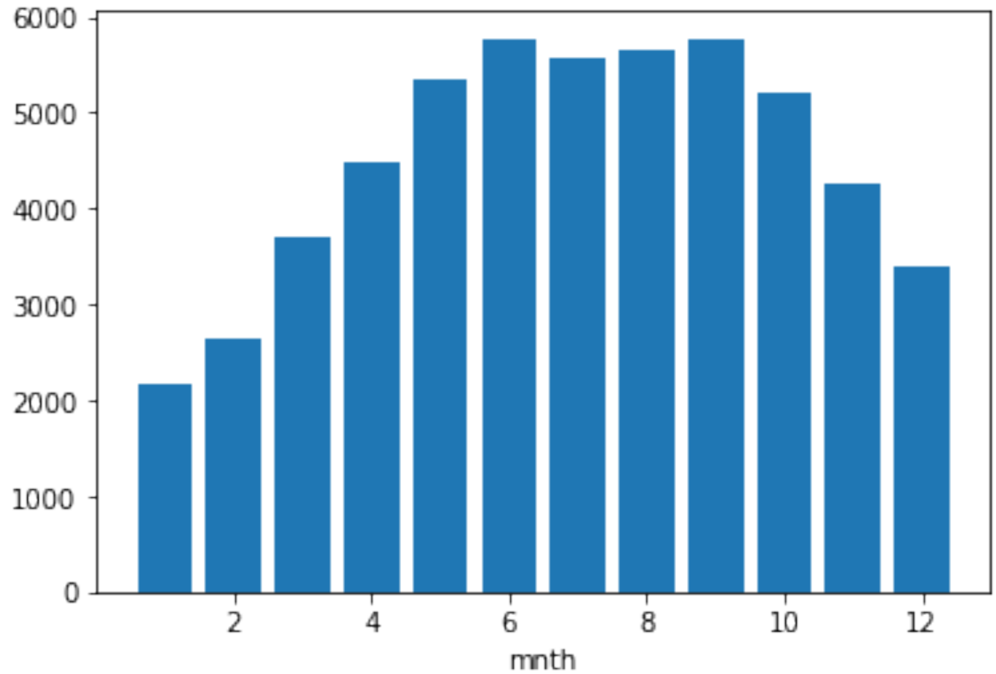
[57]:
```
mean_data = df[['cnt', 'season']].groupby('season').mean()
mean_data
```

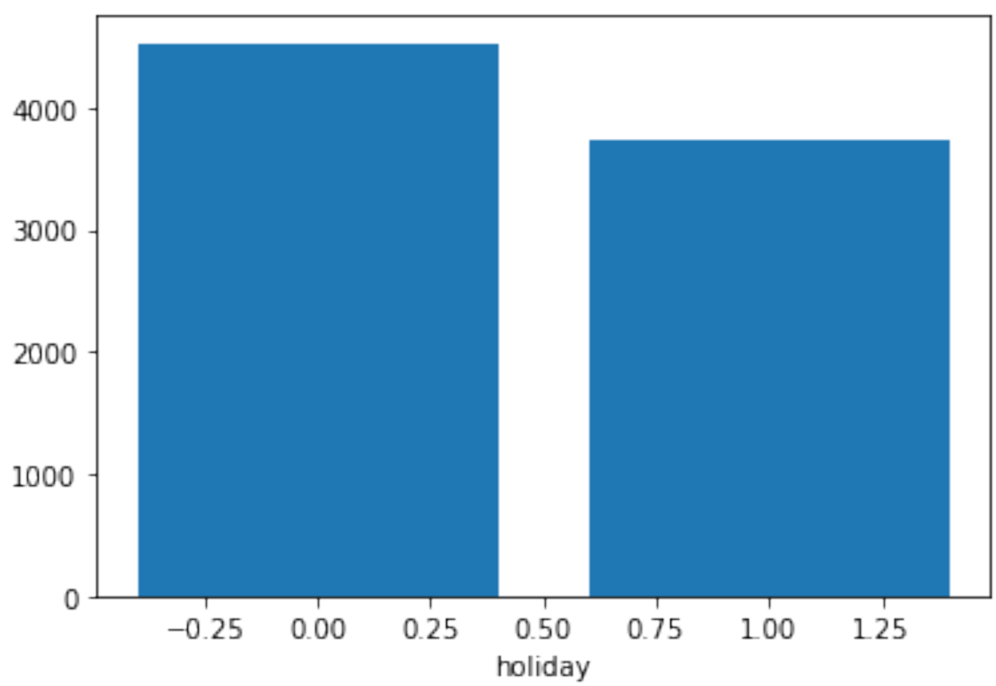
[57]:

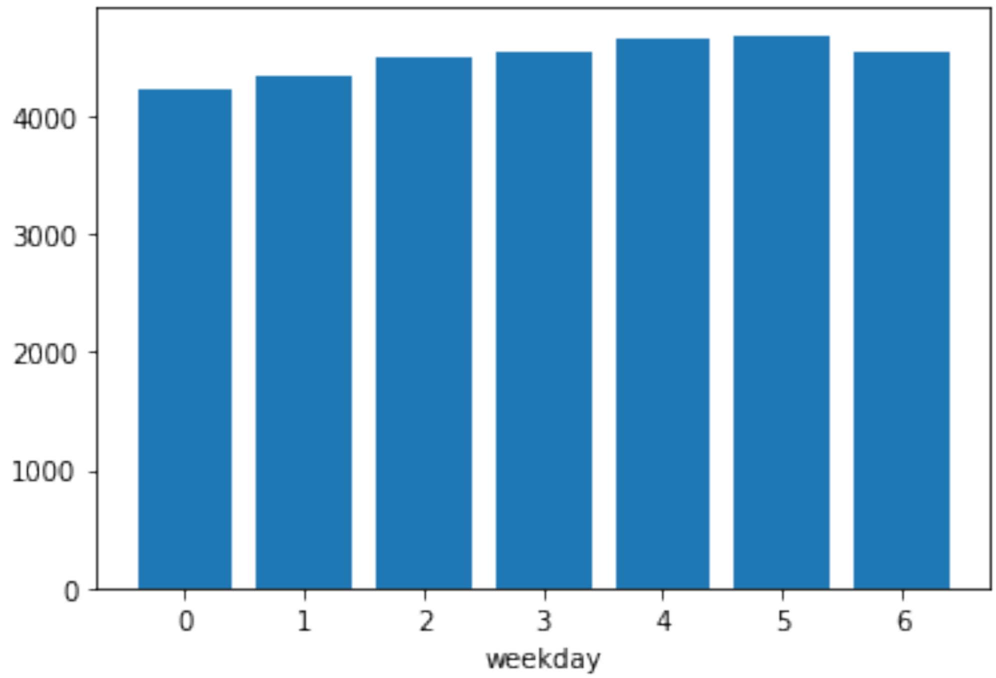
season	cnt
1	2604.132597
2	4992.331522
3	5644.303191
4	4728.162921

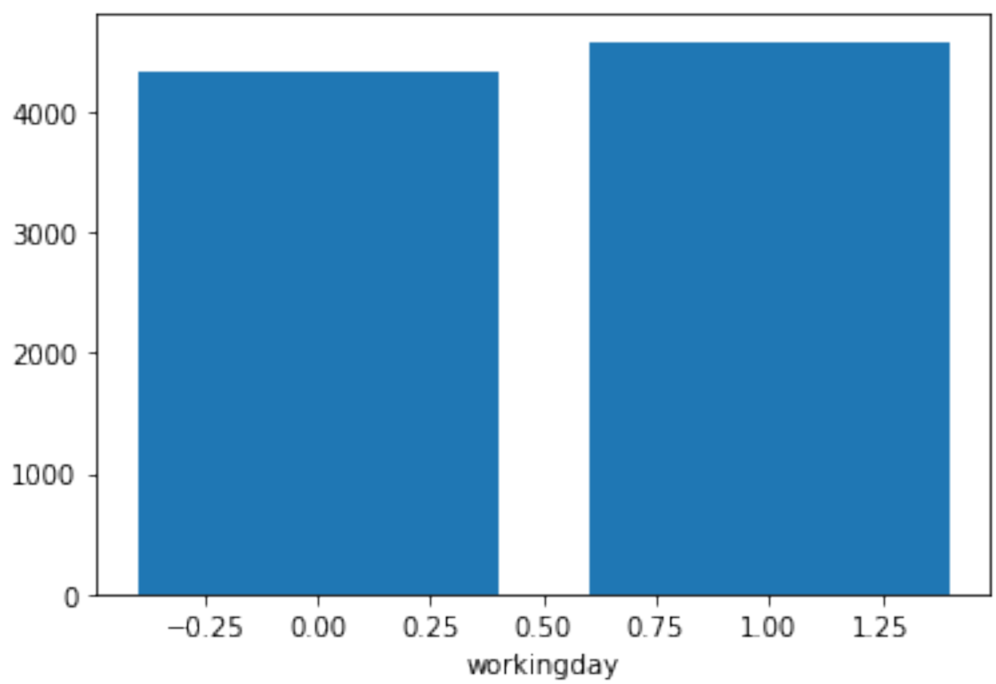
[58]:
```
for f in categorical_features:
    mean_data = df[['cnt', f]].groupby(f).mean()
    plt.bar(mean_data.index, mean_data['cnt'])
    plt.xlabel(f)
    plt.show()
```

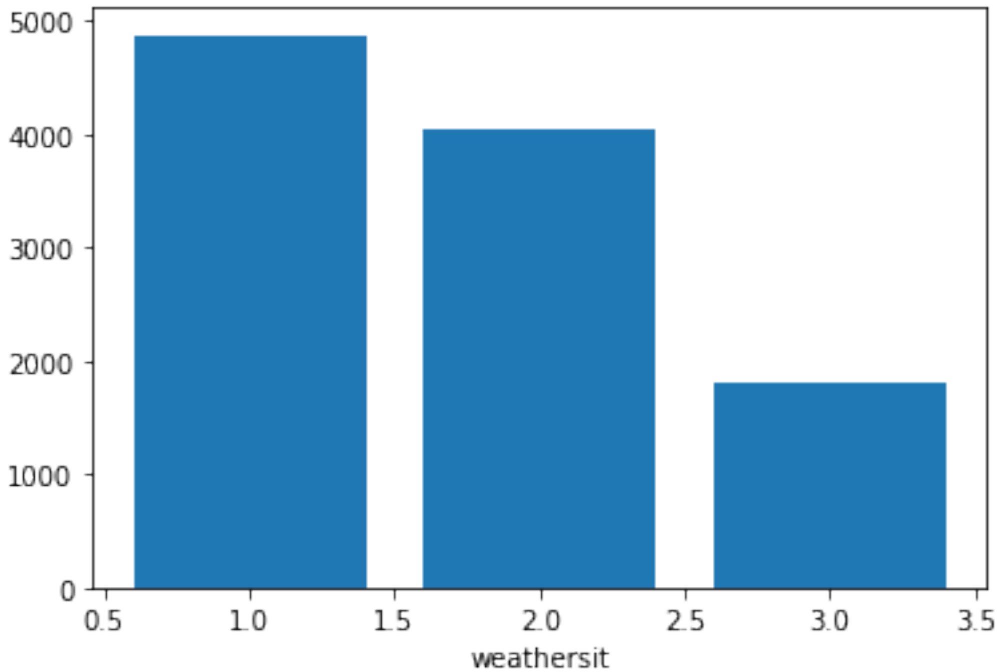



6000
5000
4000
3000
2000
1000
0
2
4
6
8
10
12
mnth


4000
3000
2000
1000
0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
holiday

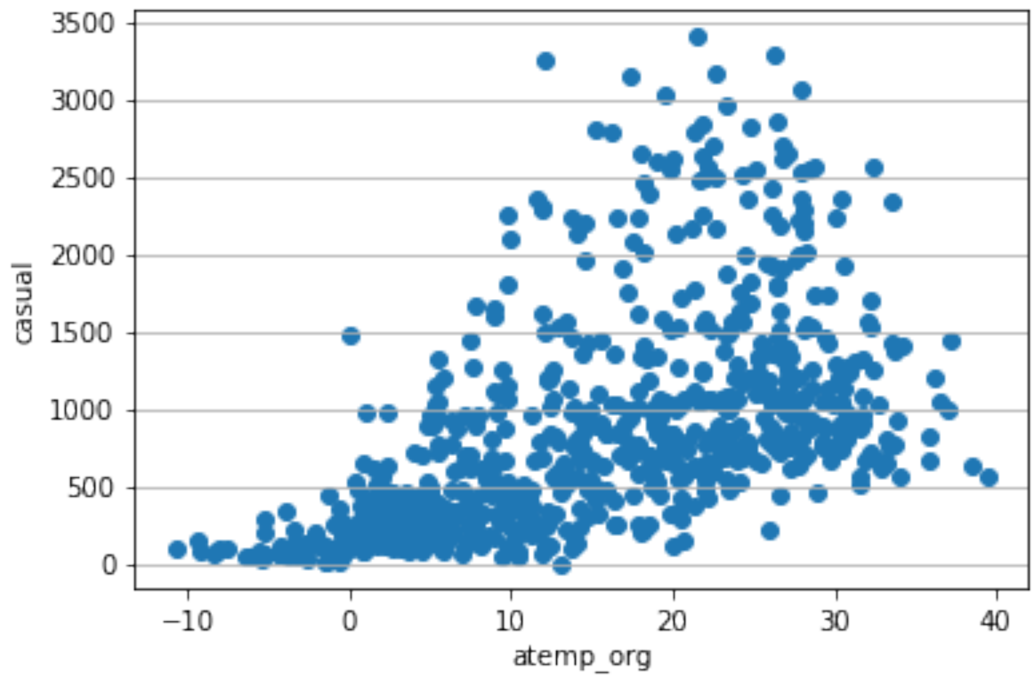

0
1000
2000
3000
4000
0
1
2
3
4
5
6
weekday

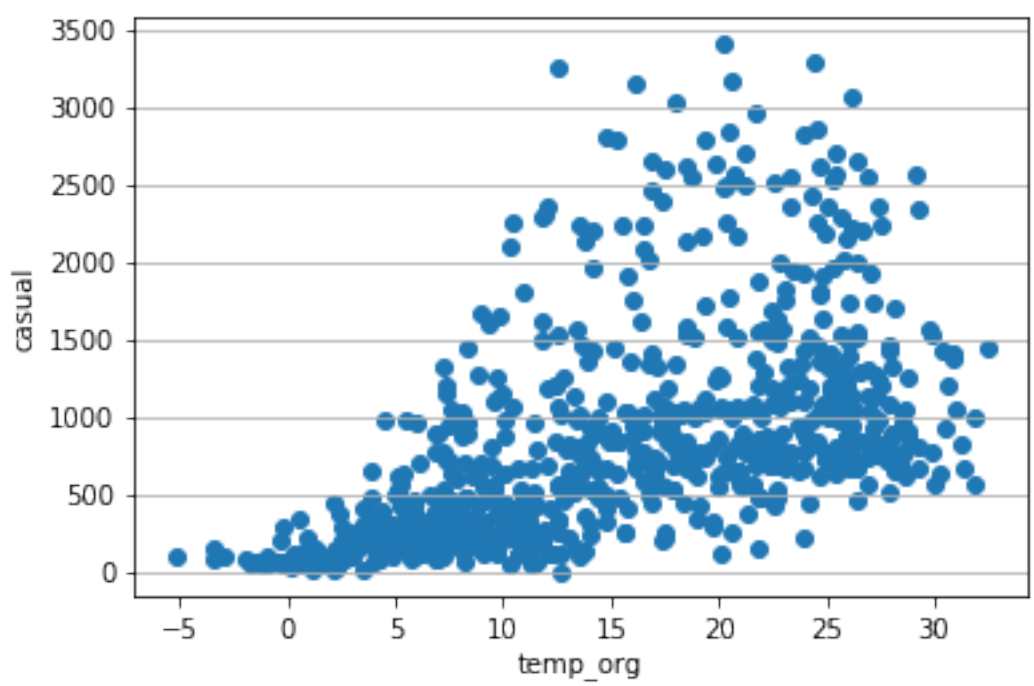

0
1000
2000
3000
4000
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
workingday

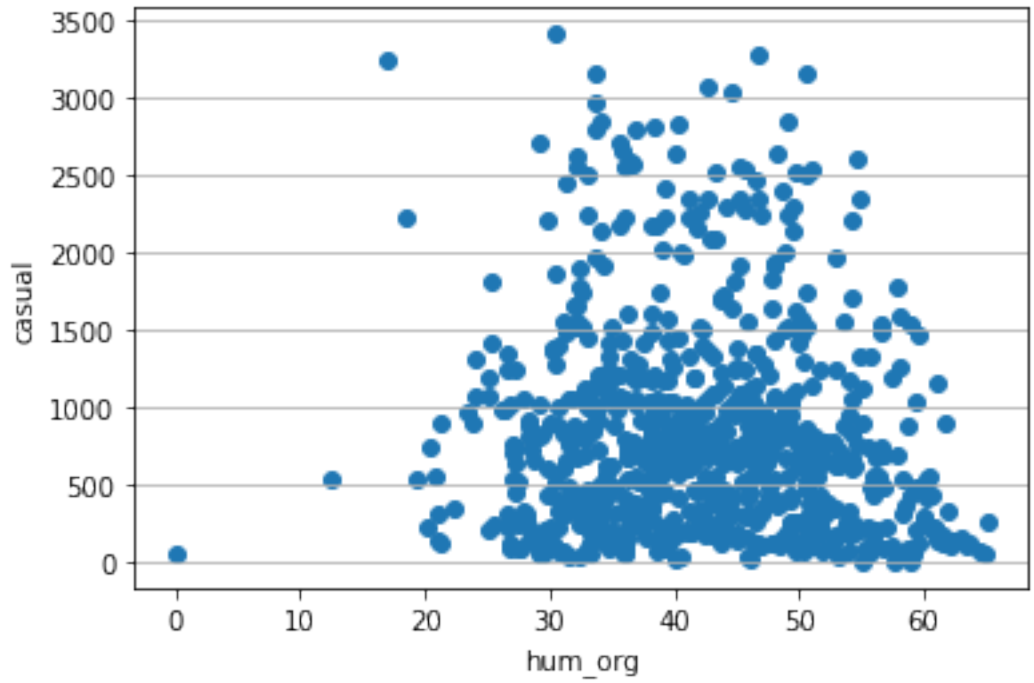


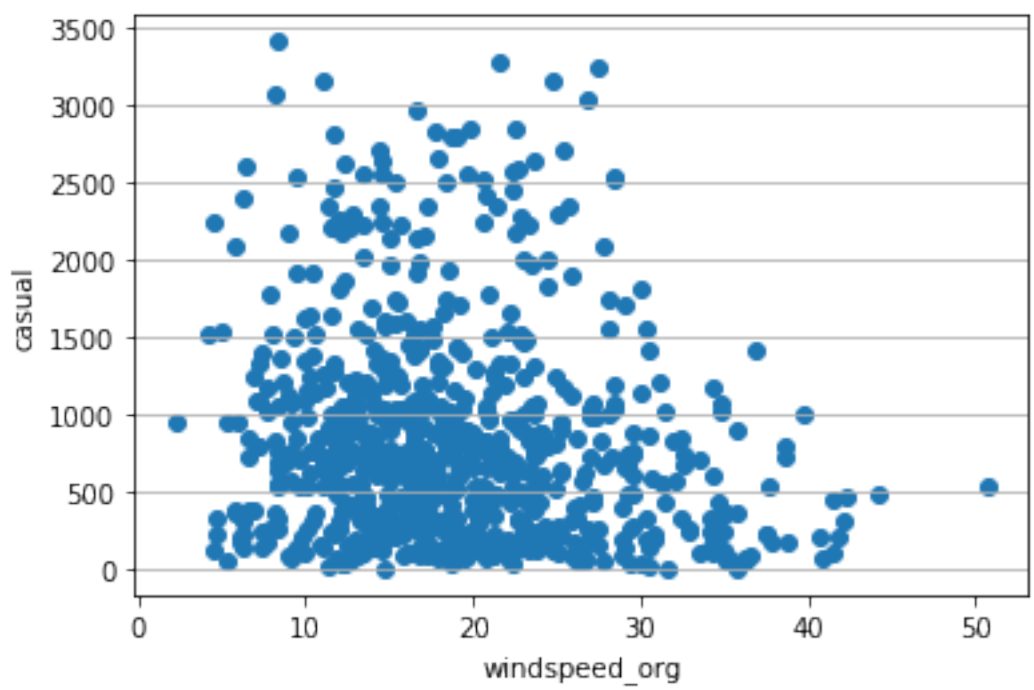
[59]:
```
# create the same plots, separately for casual and registered counts
```

[60]:
```
%matplotlib inline


features = ['atemp_org', 'temp_org', 'hum_org', 'windspeed_org']

for f in features:
    fig, axis = plt.subplots()
    # Grid lines, Xticks, Xlabel, Ylabel

    axis.yaxis.grid(True)
    X = df[f]
    Y = df['casual']
    axis.set_xlabel(f,fontsize=10)
    axis.set_ylabel('casual',fontsize=10)
    axis.scatter(X, Y)
    plt.show()
```


3500
3000
2500
2000
1500
1000
500
0
casual
−10
0
10
20
30
40
atemp_org

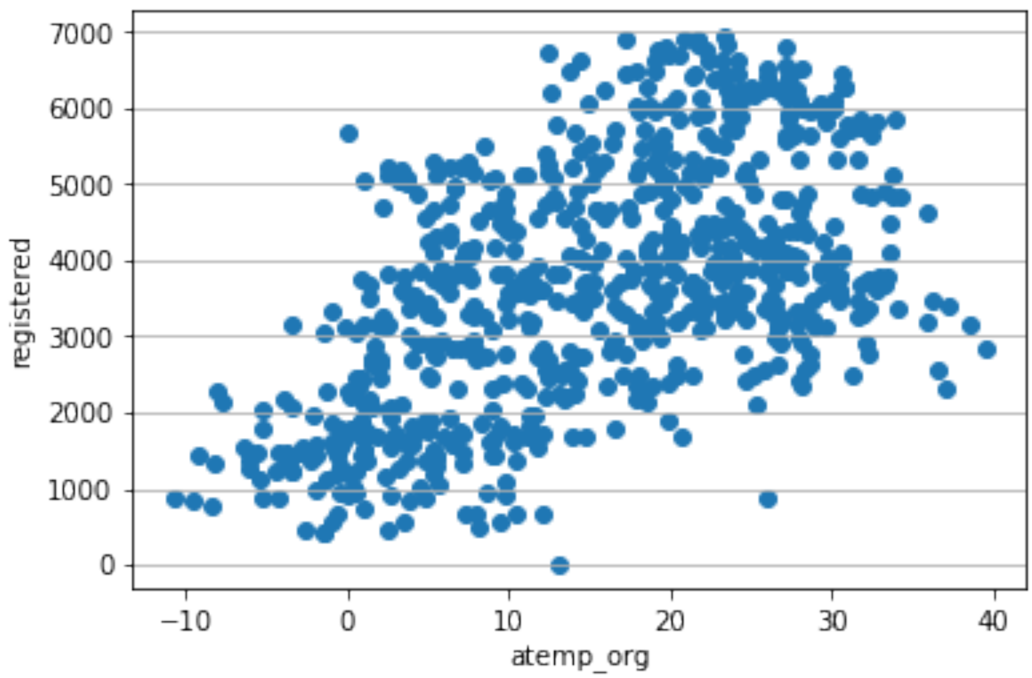

3500
3000
2500
2000
1500
1000
500
0
casual
−5
0
5
10
15
20
25
30
temp_org

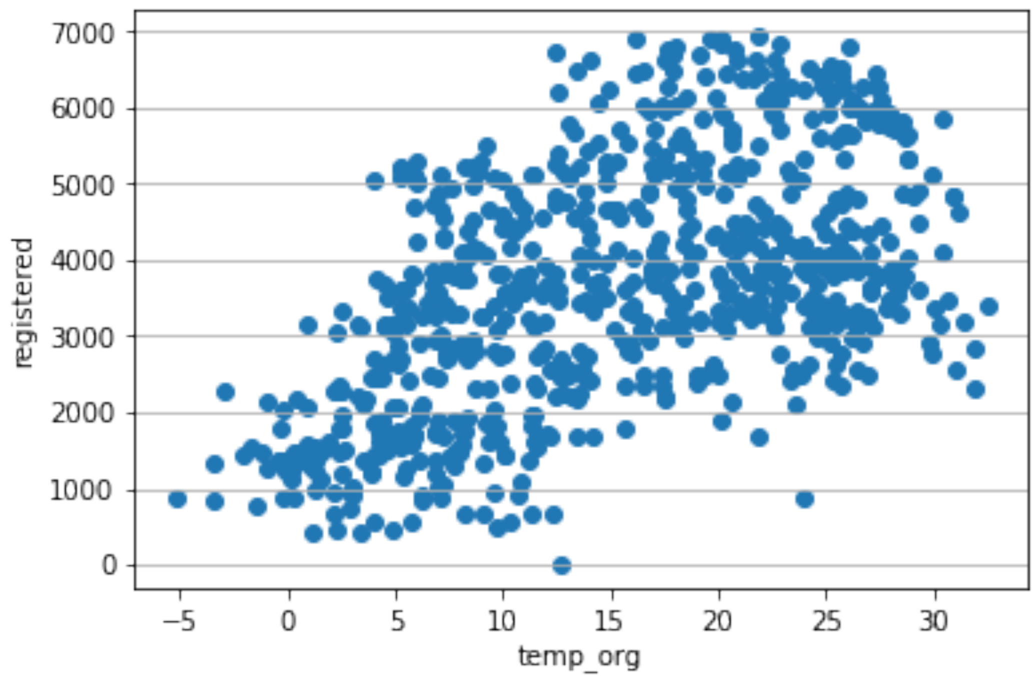


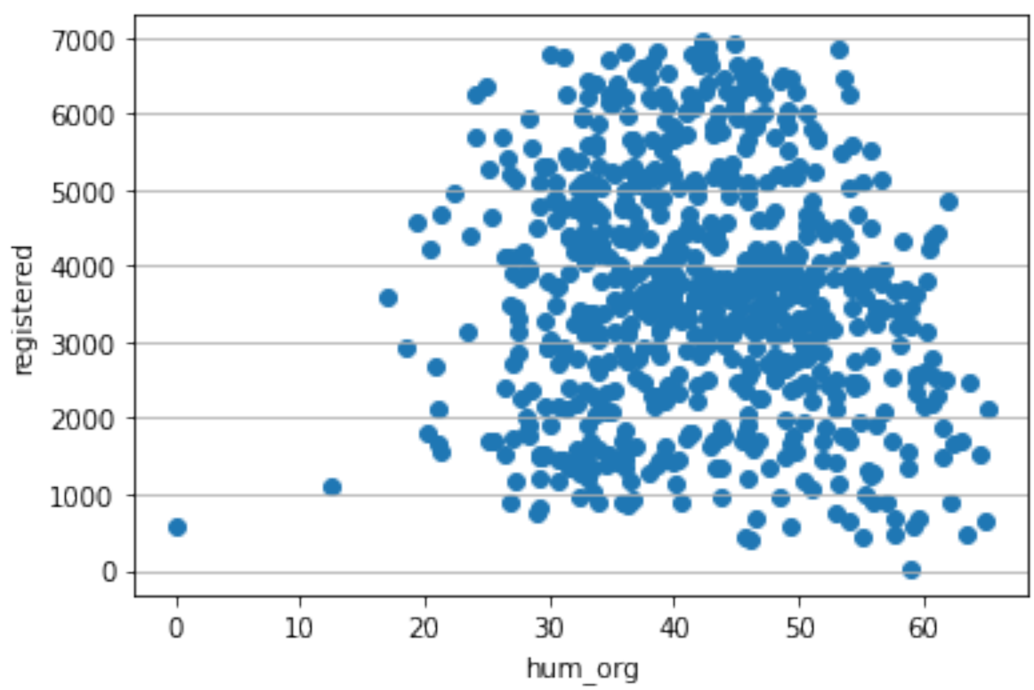


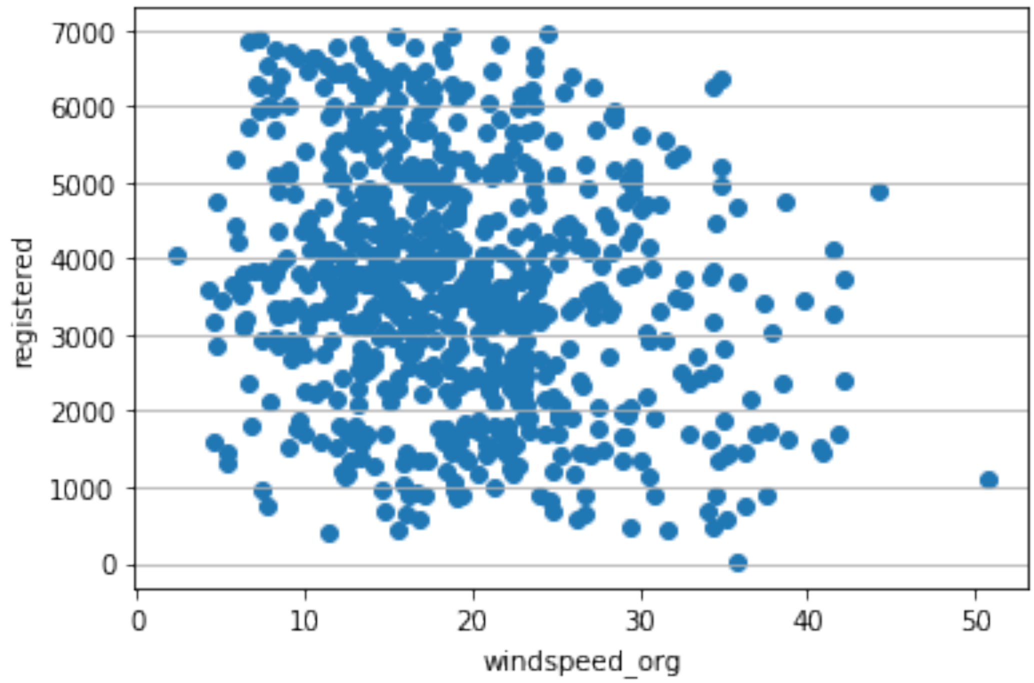
[61]:
```
%matplotlib inline
```

```
features = ['atemp_org', 'temp_org', 'hum_org', 'windspeed_org']

for f in features:
    fig, axis = plt.subplots()
    # Grid lines, Xticks, Xlabel, Ylabel

    axis.yaxis.grid(True)
    X = df[f]
    Y = df['registered']
    axis.set_xlabel(f,fontsize=10)
    axis.set_ylabel('registered',fontsize=10)
    axis.scatter(X, Y)
    plt.show()
```

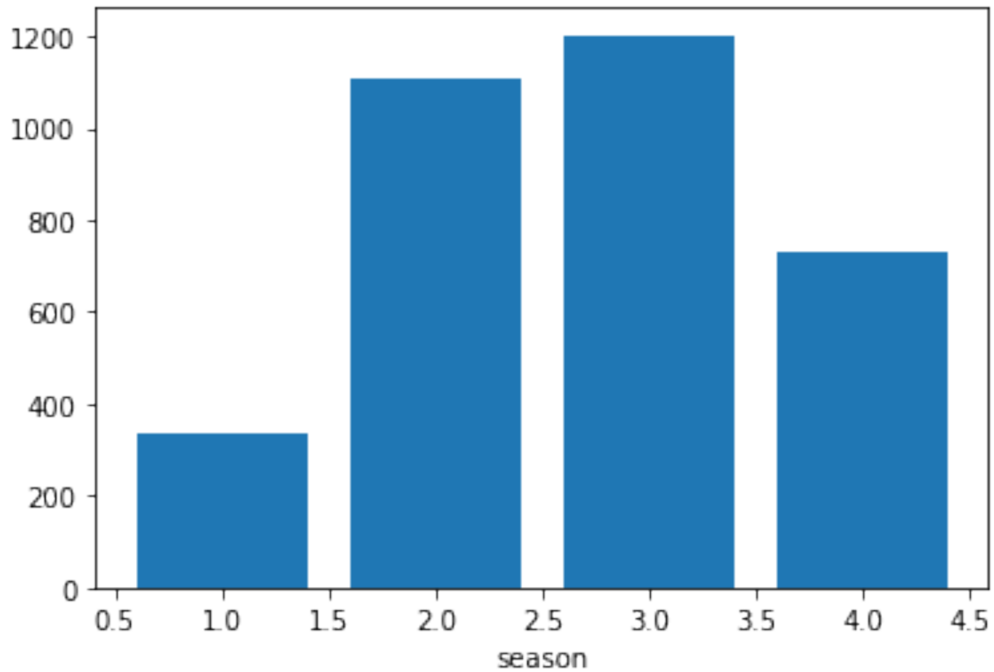



registered
temp_org
7000
6000
5000
4000
3000
2000
1000
0
−5
0
5
10
15
20
25
30

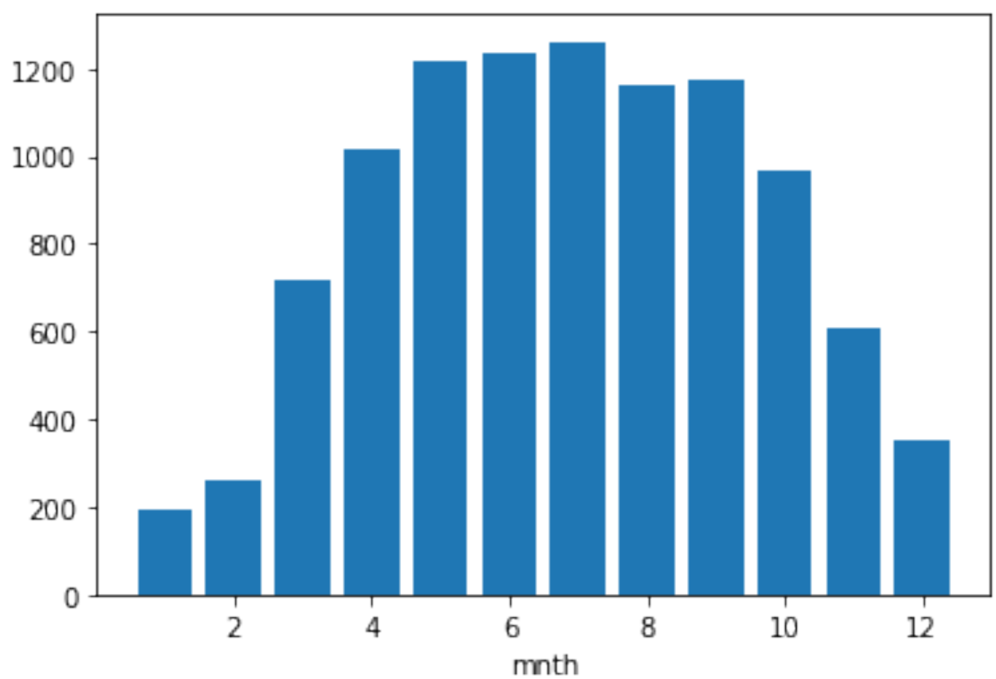

registered
hum_org
7000
6000
5000
4000
3000
2000
1000
0
0
10
20
30
40
50
60



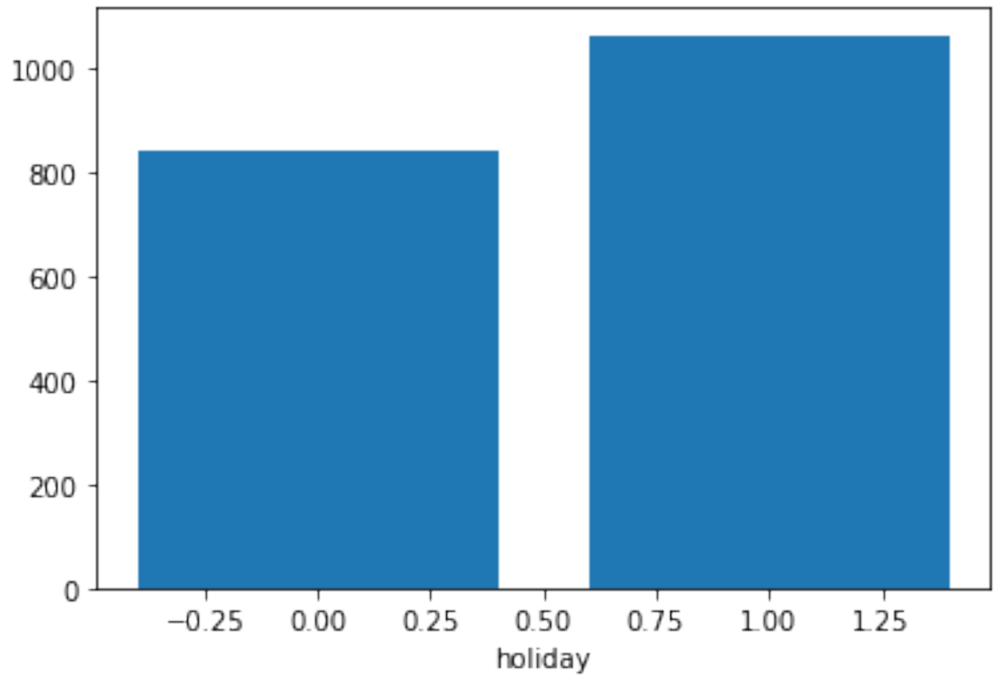
[62]:
```
for f in categorical_features:
    mean_data = df[['casual', f]].groupby(f).mean()
    plt.bar(mean_data.index, mean_data['casual'])
    plt.xlabel(f)
    plt.show()
```

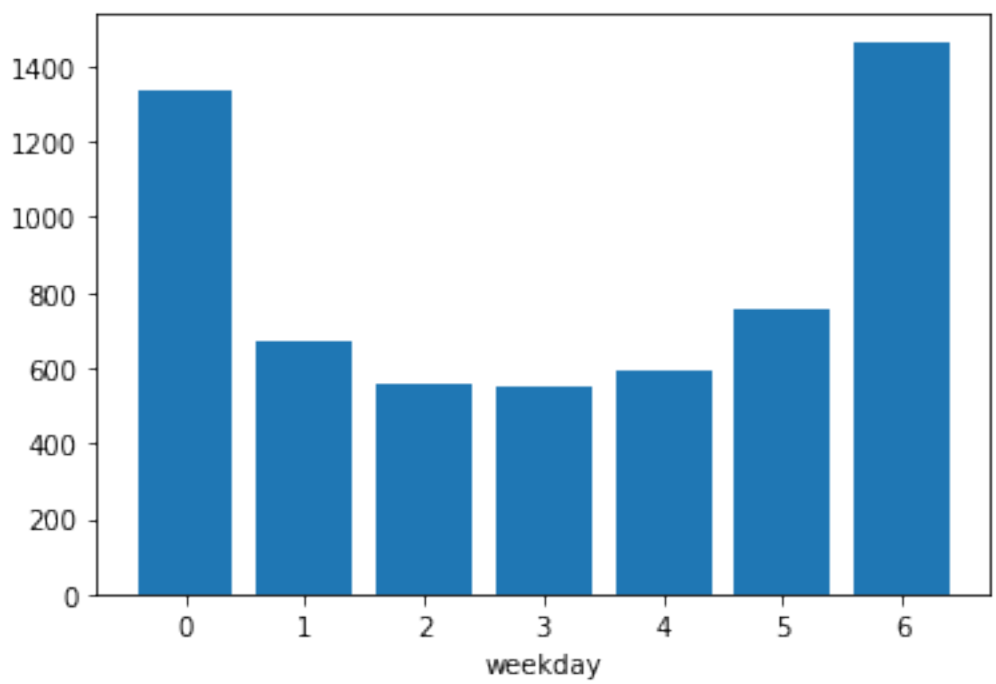


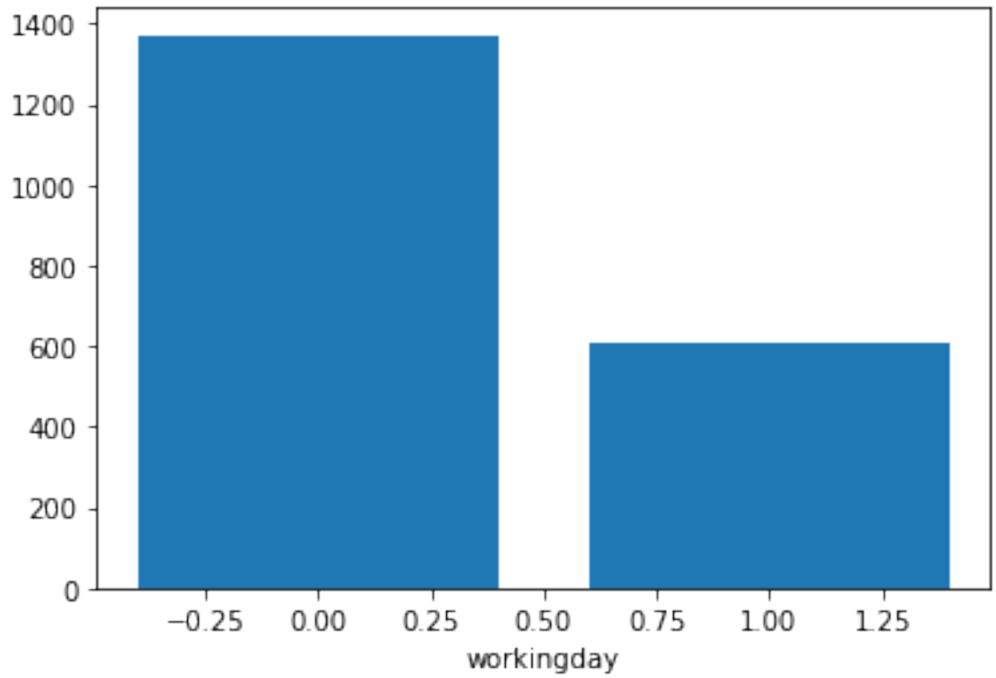
1200
1000
800
600
400
200
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
season

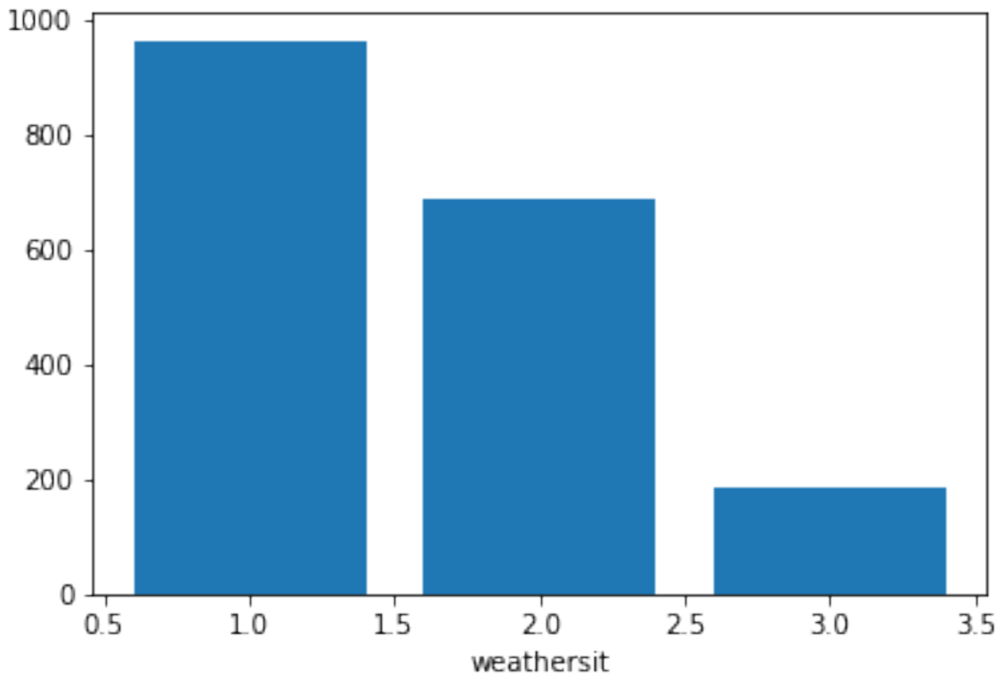


1200
1000
800
600
400
200
0
2
4
6
8
10
12
mnth


1000
800
600
400
200
0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
holiday

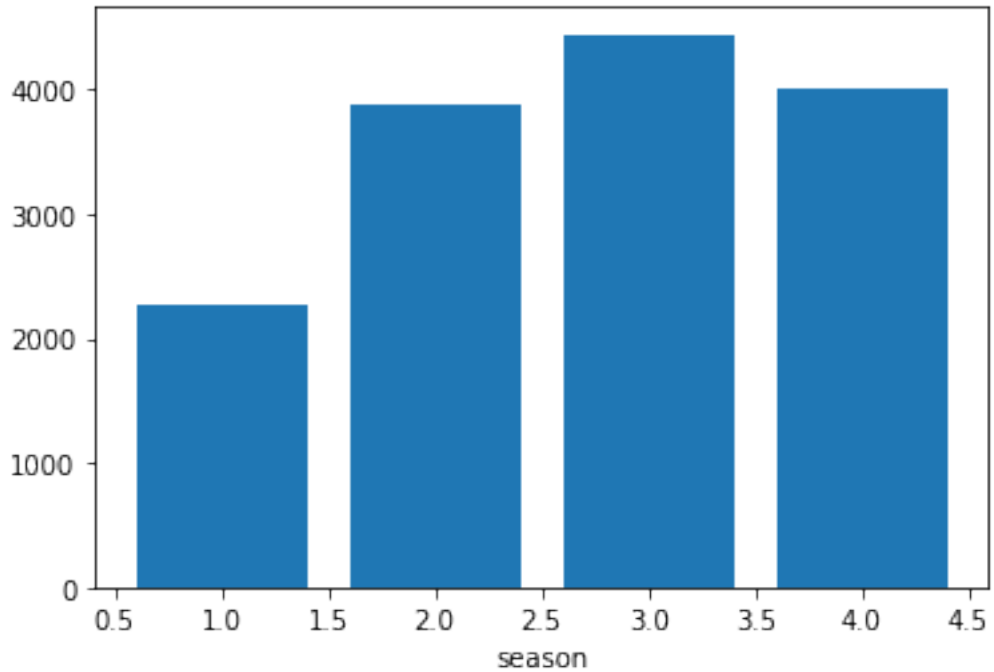

1400
1200
1000
800
600
400
200
0
0
1
2
3
4
5
6
weekday

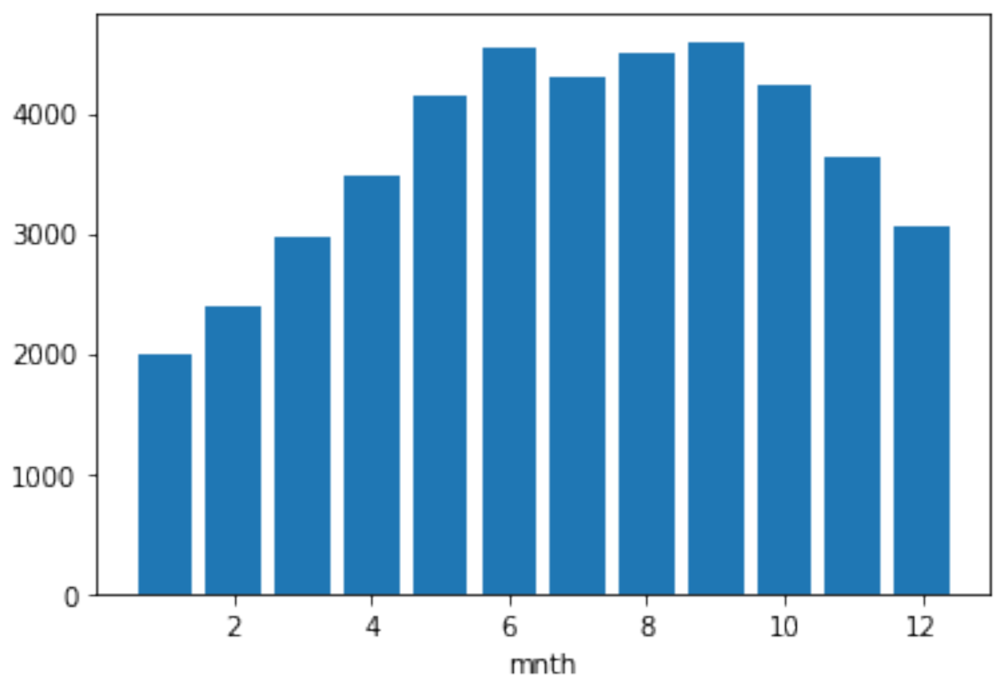


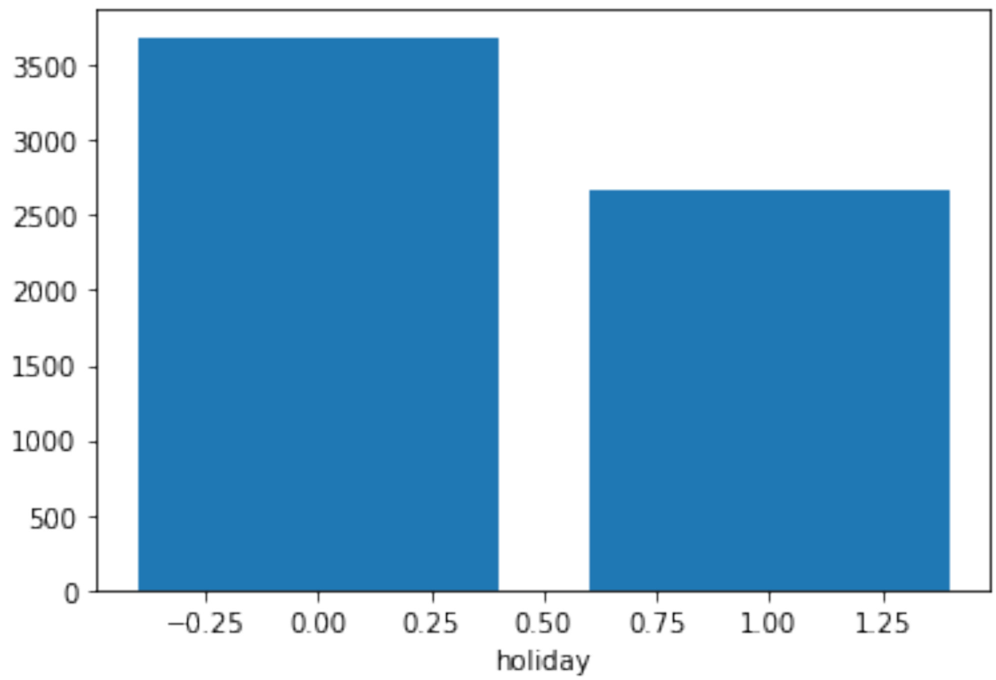


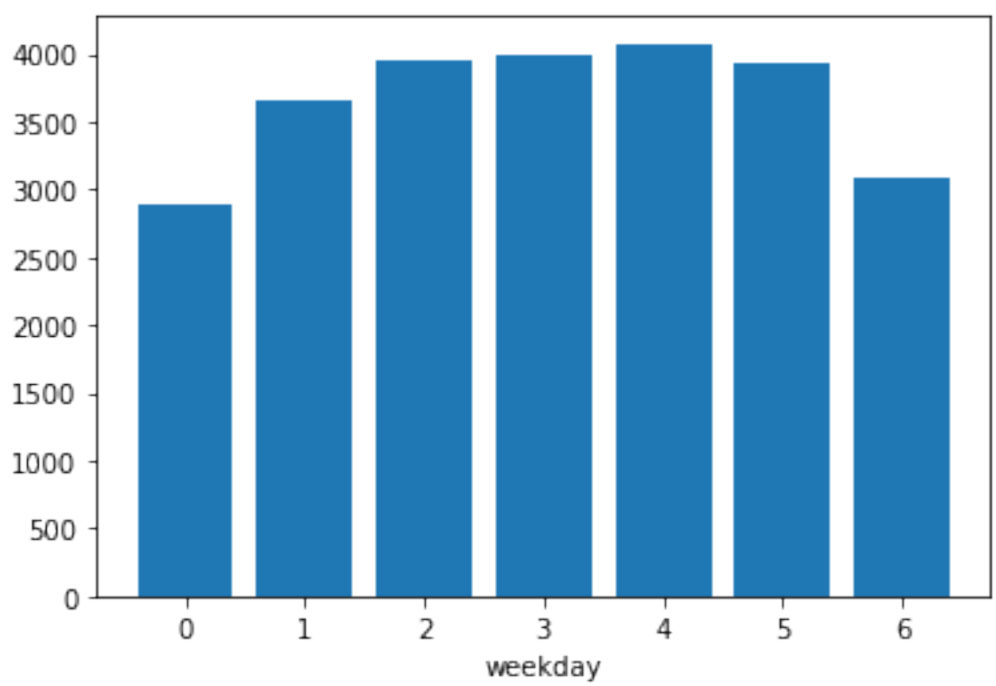
[63]:
```
for f in categorical_features:
    mean_data = df[['registered', f]].groupby(f).mean()
```

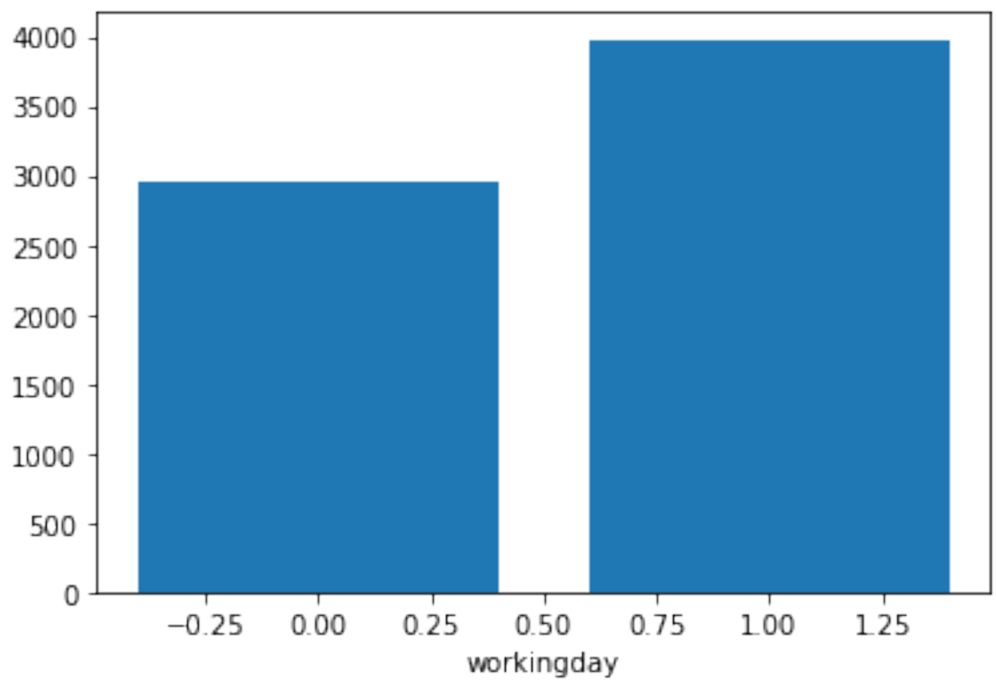
```
    plt.bar(mean_data.index, mean_data['registered'])
    plt.xlabel(f)
    plt.show()
```

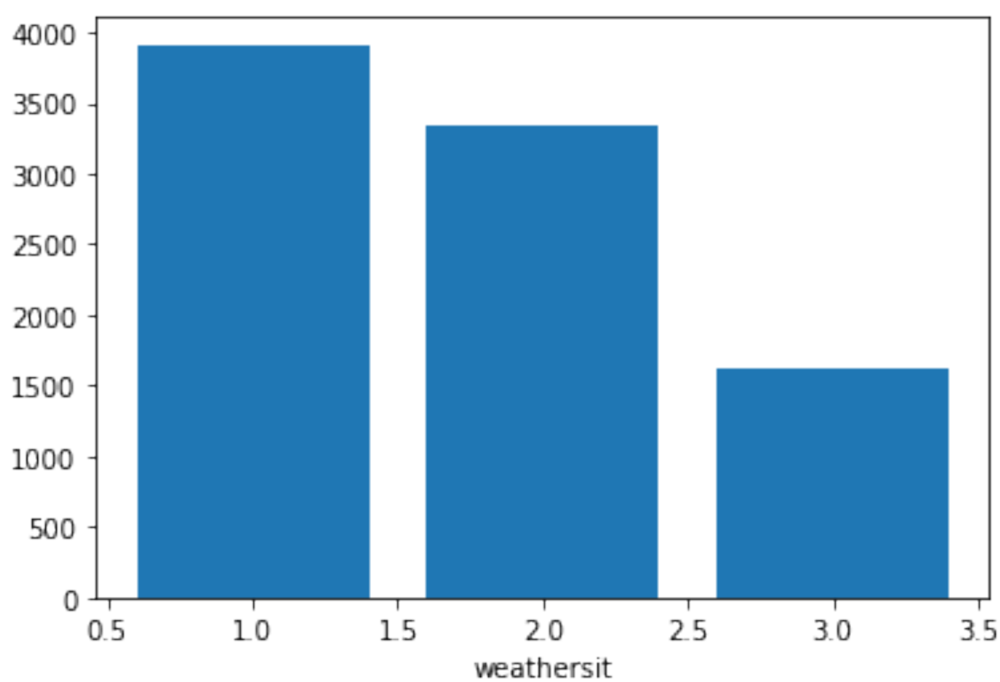





3500
3000
2500
2000
1500
1000
500
0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
holiday


4000
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
weekday





[ ]:

```

```

Discard temp, as it is highly correlated with atemp and mnth and weekday, as these have a lot of levels

[64]:
```
#features = ['atemp_norm', 'temp_norm', 'hum_norm', 'windspeed_norm', 'season',␣
↪'mnth', 'holiday', 'weekday',
#       'workingday', 'weathersit']
features = ['atemp_norm', 'hum_norm', 'windspeed_norm', 'holiday',
       'workingday']
categorical_columns = ['season','weathersit']


X = df[features]
```

[65]:
```
from sklearn.preprocessing import OneHotEncoder



categorical_data = df[categorical_columns]
```

[66]:
```
#create a OneHotEncoder object, and fit it to all of X

enc = OneHotEncoder()

enc.fit(categorical_data)

onehotlabels = enc.transform(categorical_data).toarray()
onehotlabels.shape

cat_onehot_df = pd.DataFrame(onehotlabels, columns = ['season1', 'season2',␣
↪'season3', 'season4', 'weathersit1', 'weathersit2', 'weathersit3'])
```

[67]:
```
cat_onehot_df.head()
```

[67]:

	season1	season2	season3	season4	weathersit1	weathersit2	weathersit3
0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
1	1.0	0.0	0.0	0.0	0.0	1.0	0.0
2	1.0	0.0	0.0	0.0	1.0	0.0	0.0
3	1.0	0.0	0.0	0.0	1.0	0.0	0.0
4	1.0	0.0	0.0	0.0	1.0	0.0	0.0

[68]:
```
X = pd.concat([X,cat_onehot_df], axis = 1, join = 'inner')
X.head()
```

[68]:

	atemp_norm	hum_norm	windspeed_norm	holiday	workingday	season1 \
0	-0.679946	1.250171	-0.387892	0	0	1.0
1	-0.740652	0.479113	0.749602	0	0	1.0
2	-1.749767	-1.339274	0.746632	0	1	1.0
3	-1.610270	-0.263182	-0.389829	0	1	1.0

```
4   -1.504971 -1.341494       -0.046307        0           1       1.0

   season2  season3  season4  weathersit1  weathersit2  weathersit3
0      0.0      0.0      0.0          0.0          1.0          0.0
1      0.0      0.0      0.0          0.0          1.0          0.0
2      0.0      0.0      0.0          1.0          0.0          0.0
3      0.0      0.0      0.0          1.0          0.0          0.0
4      0.0      0.0      0.0          1.0          0.0          0.0
```

[69]:
```
y_t = df['cnt']
y_r = df['registered']
y_c = df['casual']
```

[70]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y_t, random_state=42)
```

[71]:
```
X_train
```

[71]:
```
     atemp_norm  hum_norm  windspeed_norm  holiday  workingday  season1  \
688   -0.606283 -0.032045        0.575646        0           1      0.0
649   -0.265195 -1.156180       -0.114794        0           1      0.0
637    0.343519 -0.597036        0.479280        0           0      0.0
525    1.068550 -1.340609       -0.596543        0           0      0.0
367   -2.137425 -1.311332        2.262059        0           1      1.0
..          …         …             …       …          …        …
71    -0.578834 -0.706119        1.034514        0           0      1.0
106   -0.175978 -1.042010        1.459229        0           0      0.0
270    0.619000  1.551700       -0.540478        0           1      0.0
435   -0.704232 -1.060460        0.414499        0           0      1.0
102   -0.350452  1.343854        0.776434        0           1      0.0

     season2  season3  season4  weathersit1  weathersit2  weathersit3
688      0.0      0.0      1.0          0.0          1.0          0.0
649      0.0      0.0      1.0          1.0          0.0          0.0
637      0.0      0.0      1.0          1.0          0.0          0.0
525      1.0      0.0      0.0          1.0          0.0          0.0
367      0.0      0.0      0.0          1.0          0.0          0.0
..       …        …        …            …            …            …
71       0.0      0.0      0.0          1.0          0.0          0.0
106      1.0      0.0      0.0          1.0          0.0          0.0
270      0.0      0.0      1.0          0.0          1.0          0.0
435      0.0      0.0      0.0          1.0          0.0          0.0
102      1.0      0.0      0.0          0.0          1.0          0.0

[548 rows x 12 columns]
```

[72]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   instant         731 non-null    int64
 1   dteday          731 non-null    object
 2   season          731 non-null    int64
 3   yr              731 non-null    int64
 4   mnth            731 non-null    int64
 5   holiday         731 non-null    int64
 6   weekday         731 non-null    int64
 7   workingday      731 non-null    int64
 8   weathersit      731 non-null    int64
 9   temp            731 non-null    float64
 10  atemp           731 non-null    float64
 11  hum             731 non-null    float64
 12  windspeed       731 non-null    float64
 13  casual          731 non-null    int64
 14  registered      731 non-null    int64
 15  cnt             731 non-null    int64
 16  temp_org        731 non-null    float64
 17  atemp_org       731 non-null    float64
 18  windspeed_org   731 non-null    float64
 19  hum_org         731 non-null    float64
 20  atemp_norm      731 non-null    float64
 21  temp_norm       731 non-null    float64
 22  hum_norm        731 non-null    float64
 23  windspeed_norm  731 non-null    float64
dtypes: float64(12), int64(11), object(1)
memory usage: 137.2+ KB
```

[73]:
```
from sklearn.linear_model import RidgeCV
from sklearn.metrics import r2_score

# Here cross validation is used to determine the best value for alpha

model_rigde = RidgeCV(alphas=np.logspace(-10, 10, 21))

model_rigde.fit(X_train, y_train)
```

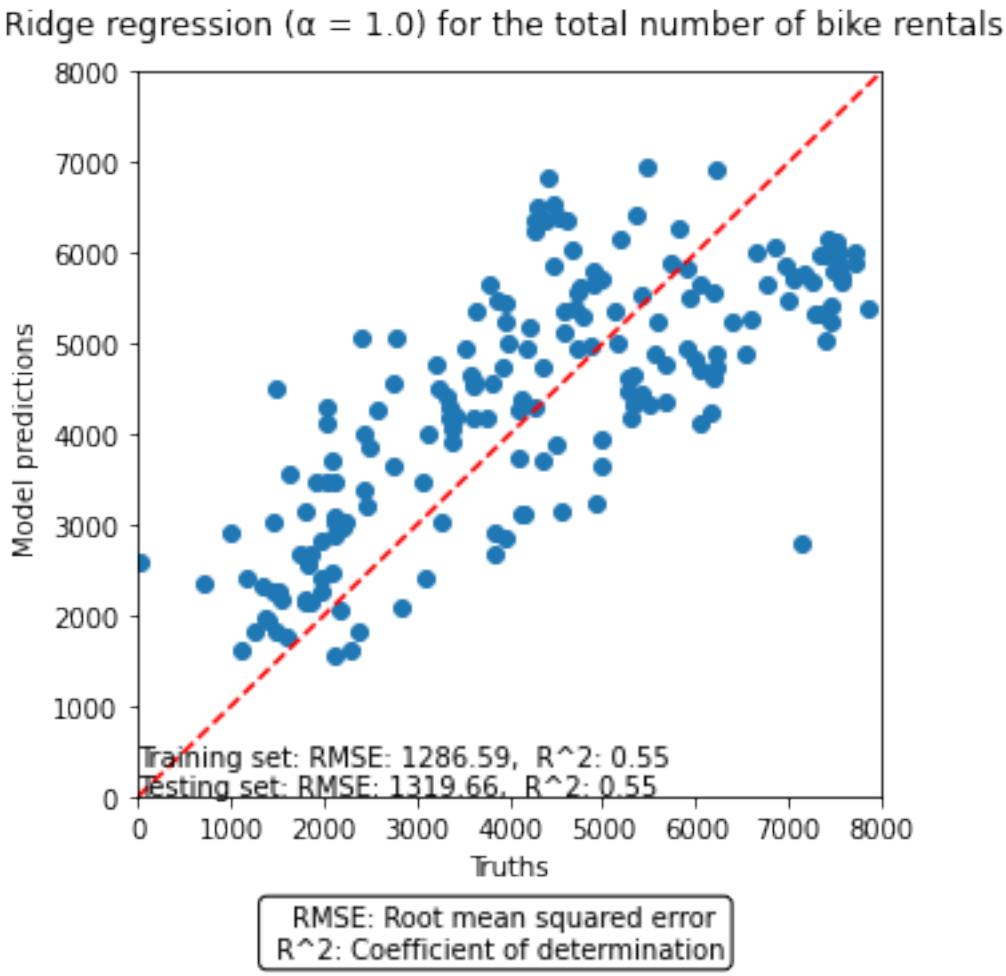
[73]:
```
RidgeCV(alphas=array([1.e-10, 1.e-09, 1.e-08, 1.e-07, 1.e-06, 1.e-05, 1.e-04,
1.e-03,
       1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03, 1.e+04, 1.e+05,
       1.e+06, 1.e+07, 1.e+08, 1.e+09, 1.e+10]))
```

Check which value of α has been selected.

[74]:
```
model_rigde.alpha_
```

[74]: 1.0

[75]:
```
y_pred = model_rigde.predict(X_train)

mae = sqrt(mean_squared_error(y_true = y_train, y_pred = y_pred))
c_det = r2_score(y_train, y_pred)
string_score = f"Training set: RMSE: {mae:.2f}" + ", " + " R^2: "  + "{:.2f}".
↪format(c_det)
y_pred = model_rigde.predict(X_test)
mae = sqrt(mean_squared_error(y_true = y_test, y_pred = y_pred))
c_det = r2_score(y_test, y_pred)
string_score += f"\nTesting set: RMSE: {mae:.2f}" + ", " + " R^2: "  + "{:.2f}".
↪format(c_det)
fig, ax = plt.subplots(figsize=(5, 5))
plt.scatter(y_test, y_pred)
ax.plot([0, 1], [0, 1], transform=ax.transAxes, ls="--", c="red")
ax.annotate('RMSE: Root mean squared error \n R^2: Coefficient of␣
↪determination', xy=(220, -40), xycoords='axes points',
            size=10, ha='right', va='top',
            bbox=dict(boxstyle='round', fc='w'))
plt.text(3, 20, string_score)
plt.title("Ridge regression (\u03B1 = 1.0) for the total number of bike rentals␣
↪", x=0.5, y=1.03)
plt.ylabel("Model predictions")
plt.xlabel("Truths")
plt.xlim([0, 8000])
_ = plt.ylim([0, 8000])
plt.savefig('Number of total bike rentals for the regression model', dpi = 500,␣
↪bbox_inches='tight')
```



[76]:

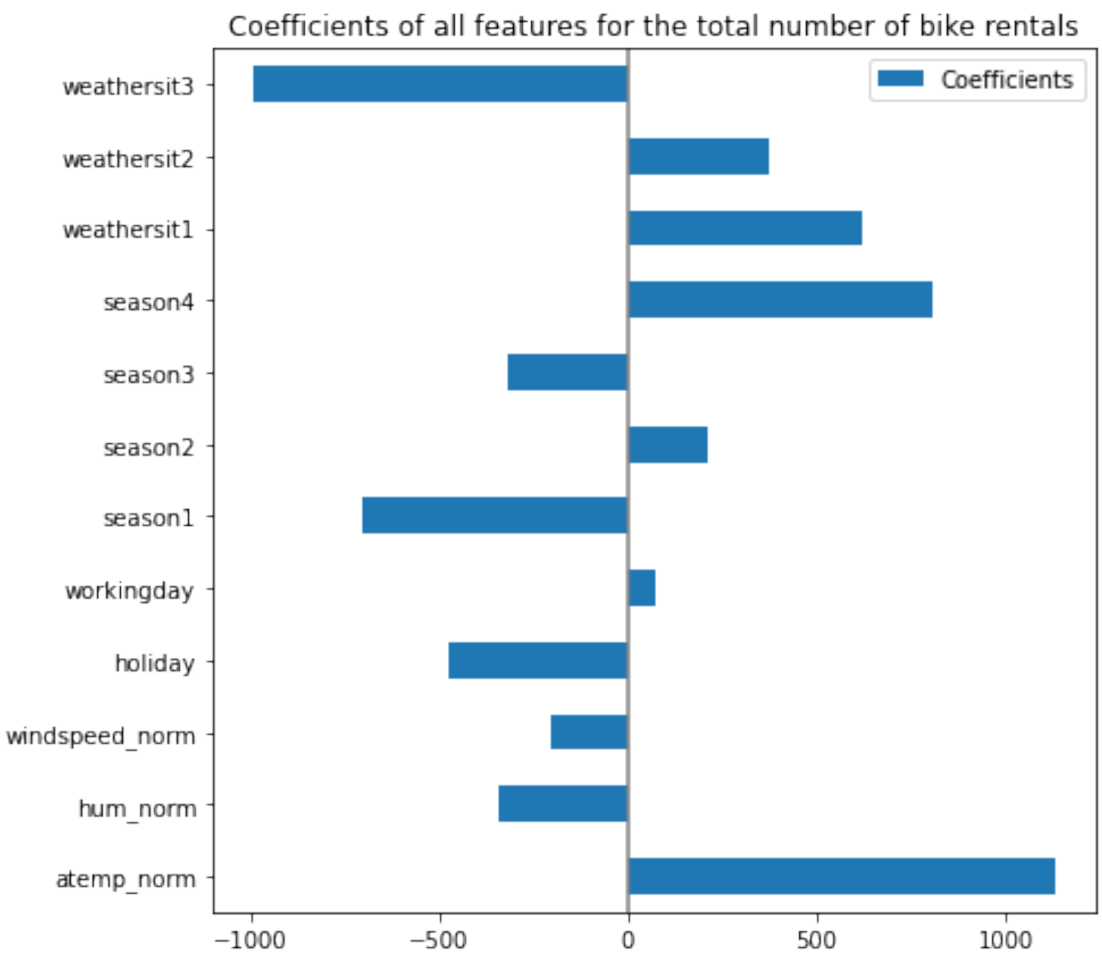
```
feature_names = X.columns
coefs = pd.DataFrame(
    model_rigde.coef_,
    columns=["Coefficients"],
    index=feature_names,
)
coefs
```

[76]:

	Coefficients
atemp_norm	1133.361311
hum_norm	-342.650815
windspeed_norm	-203.825740
holiday	-474.742179
workingday	71.683843
season1	-700.682607
season2	214.039214
season3	-320.396090

```
season4           807.039483
weathersit1       619.956495
weathersit2       373.721751
weathersit3      -993.678247
```

[77]:
```
coefs.plot(kind="barh", figsize=(9, 7))
plt.title("Coefficients of all features for the total number of bike rentals")
plt.axvline(x=0, color=".5")
plt.subplots_adjust(left=0.3)
plt.savefig(' Coefficients Number of total bike rentals ', dpi = 500)
```



[ ]:
```

```

[78]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y_r, random_state=42)
```

[79]:
```
from sklearn.linear_model import RidgeCV
from sklearn.metrics import r2_score
```

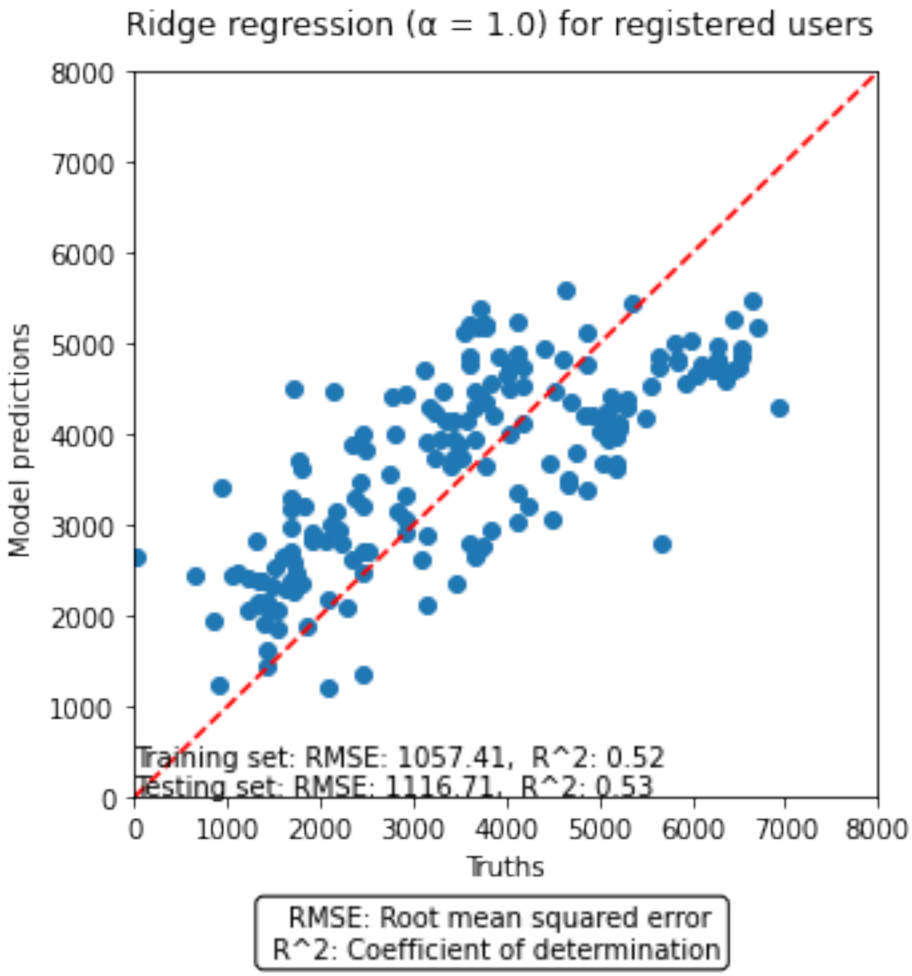
```python
model_rigde = RidgeCV(alphas=np.logspace(-10, 10, 21))

model_rigde.fit(X_train, y_train)
```

[79]:

```
RidgeCV(alphas=array([1.e-10, 1.e-09, 1.e-08, 1.e-07, 1.e-06, 1.e-05, 1.e-04,
1.e-03,
       1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03, 1.e+04, 1.e+05,
       1.e+06, 1.e+07, 1.e+08, 1.e+09, 1.e+10]))
```

[80]:

```python
y_pred = model_rigde.predict(X_train)

mae = sqrt(mean_squared_error(y_true = y_train, y_pred = y_pred))
c_det = r2_score(y_train, y_pred)
string_score = f"Training set: RMSE: {mae:.2f}" + ", " + " R^2: "  + "{:.2f}".
 ↪format(c_det)
y_pred = model_rigde.predict(X_test)
mae = sqrt(mean_squared_error(y_true = y_test, y_pred = y_pred))
c_det = r2_score(y_test, y_pred)
string_score += f"\nTesting set: RMSE: {mae:.2f}" + ", " + " R^2: "  + "{:.2f}".
 ↪format(c_det)
fig, ax = plt.subplots(figsize=(5, 5))
plt.scatter(y_test, y_pred)
ax.plot([0, 1], [0, 1], transform=ax.transAxes, ls="--", c="red")
ax.annotate('RMSE: Root mean squared error \n R^2: Coefficient of␣
 ↪determination', xy=(220, -40), xycoords='axes points',
            size=10, ha='right', va='top',
            bbox=dict(boxstyle='round', fc='w'))
plt.text(3, 20, string_score)
plt.title("Ridge regression (\u03B1 = 1.0) for registered users ", x=0.5, y=1.
 ↪03)
plt.ylabel("Model predictions")
plt.xlabel("Truths")
plt.xlim([0, 8000])
_ = plt.ylim([0, 8000])
plt.savefig('Ridge regression for registered users', dpi = 500,␣
 ↪bbox_inches='tight')
```



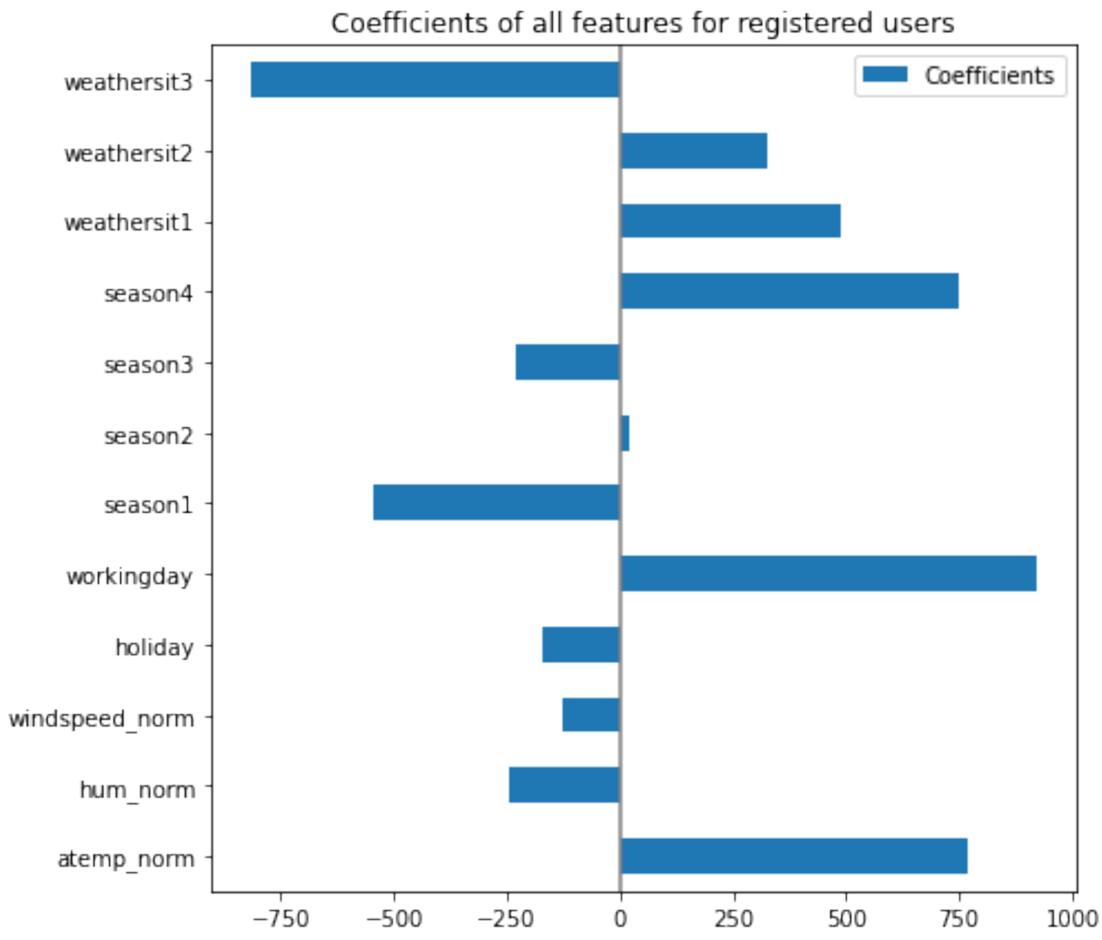
[81]:
```
coefs = pd.DataFrame(
    model_rigde.coef_,
    columns=["Coefficients"],
    index=feature_names,
)
coefs
```

[81]:

	Coefficients
atemp_norm	768.057721
hum_norm	-246.869372
windspeed_norm	-128.234429
holiday	-171.295155
workingday	922.525027
season1	-542.315422
season2	23.329909
season3	-229.796058
season4	748.781571

```
weathersit1       487.979402
weathersit2       328.579428
weathersit3      -816.558830
```

[82]:
```
coefs.plot(kind="barh", figsize=(9, 7))
plt.title("Coefficients of all features for registered users")
plt.axvline(x=0, color=".5")
plt.subplots_adjust(left=0.3)
plt.savefig(' Coefficients of all features for registered users ', dpi = 500)
```



[83]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y_c, random_state=42)
```

[84]:
```
from sklearn.linear_model import RidgeCV
from sklearn.metrics import r2_score


model_rigde = RidgeCV(alphas=np.logspace(-10, 10, 21))
```

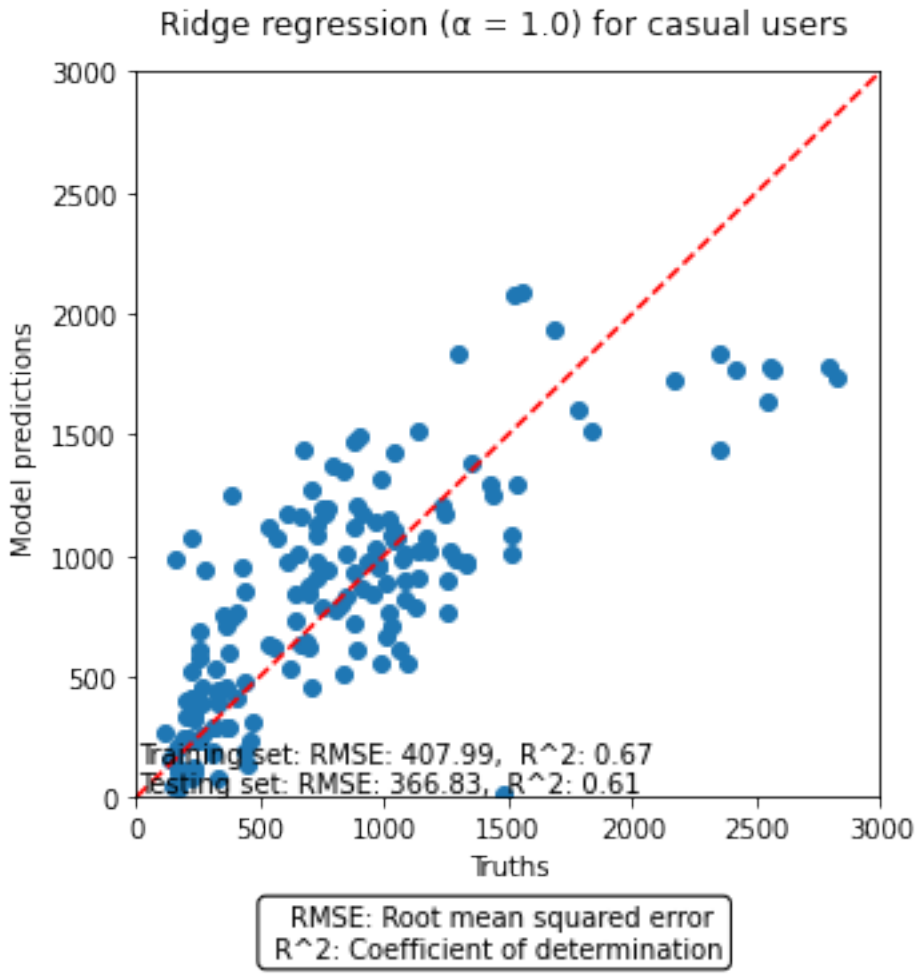
```
model_rigde.fit(X_train, y_train)
```

[84]:

```
RidgeCV(alphas=array([1.e-10, 1.e-09, 1.e-08, 1.e-07, 1.e-06, 1.e-05, 1.e-04,
1.e-03,
       1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03, 1.e+04, 1.e+05,
       1.e+06, 1.e+07, 1.e+08, 1.e+09, 1.e+10]))
```

[85]:

```
y_pred = model_rigde.predict(X_train)

mae = sqrt(mean_squared_error(y_true = y_train, y_pred = y_pred))
c_det = r2_score(y_train, y_pred)
string_score = f"Training set: RMSE: {mae:.2f}" + ", " + " R^2: "  + "{:.2f}".
 ↪format(c_det)
y_pred = model_rigde.predict(X_test)
mae = sqrt(mean_squared_error(y_true = y_test, y_pred = y_pred))
c_det = r2_score(y_test, y_pred)
string_score += f"\nTesting set: RMSE: {mae:.2f}" + ", " + " R^2: "  + "{:.2f}".
 ↪format(c_det)
fig, ax = plt.subplots(figsize=(5, 5))
plt.scatter(y_test, y_pred)
ax.plot([0, 1], [0, 1], transform=ax.transAxes, ls="--", c="red")
ax.annotate('RMSE: Root mean squared error \n R^2: Coefficient of␣
 ↪determination', xy=(220, -40), xycoords='axes points',
            size=10, ha='right', va='top',
            bbox=dict(boxstyle='round', fc='w'))
plt.text(3, 20, string_score)
plt.title("Ridge regression (\u03B1 = 1.0) for casual users ", x=0.5, y=1.03)
plt.ylabel("Model predictions")
plt.xlabel("Truths")
plt.xlim([0, 3000])
_ = plt.ylim([0, 3000])
plt.savefig('Ridge regression for casual users', dpi = 500, bbox_inches='tight')
```



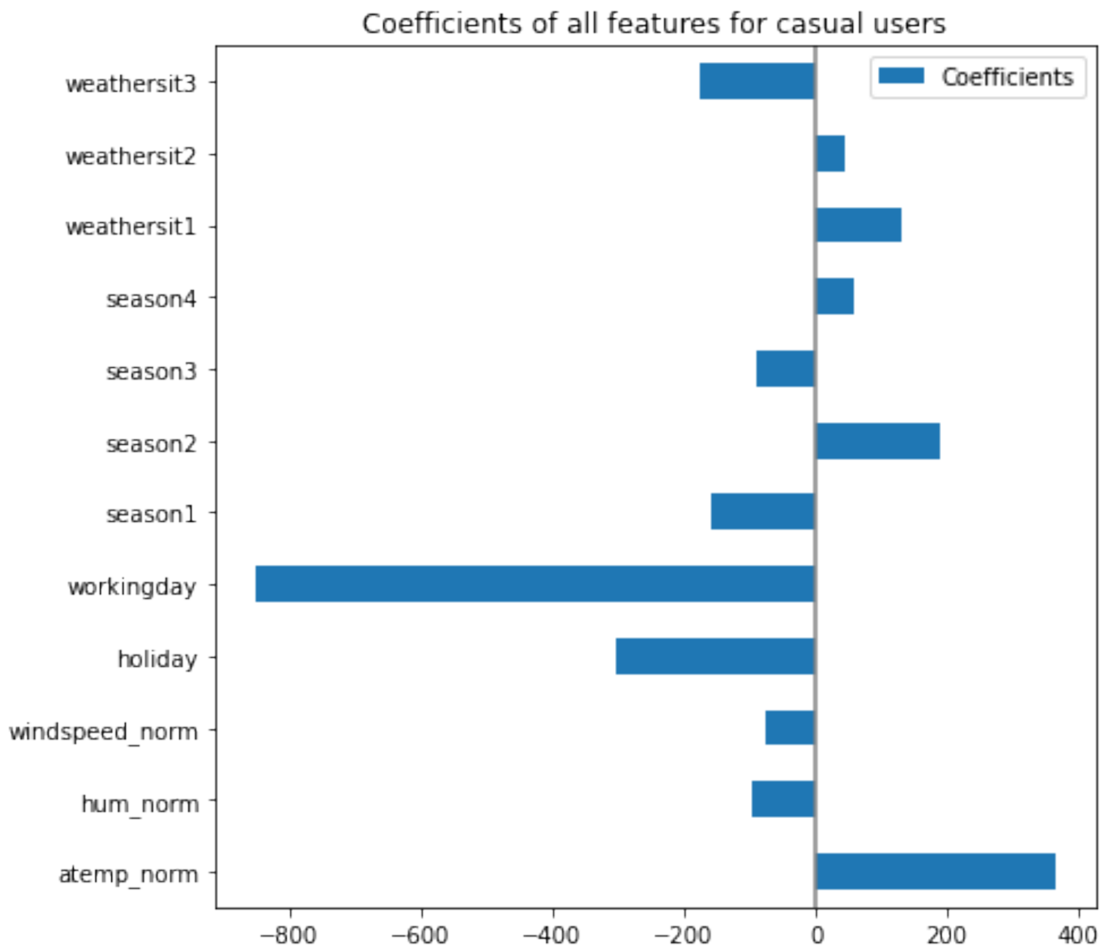
[86]:
```
coefs = pd.DataFrame(
    model_rigde.coef_,
    columns=["Coefficients"],
    index=feature_names,
)
coefs
```

[86]:

	Coefficients
atemp_norm	365.303590
hum_norm	-95.781444
windspeed_norm	-75.591311
holiday	-303.447024
workingday	-850.841185
season1	-158.367184
season2	190.709304
season3	-90.600032
season4	58.257912

```
weathersit1       131.977094
weathersit2        45.142323
weathersit3      -177.119417
```

[87]:
```
coefs.plot(kind="barh", figsize=(9, 7))
plt.title("Coefficients of all features for casual users")
plt.axvline(x=0, color=".5")
plt.subplots_adjust(left=0.3)
plt.savefig(' Coefficients of all features for casual users ', dpi = 500)
```



[ ]: