In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import linear_model,ensemble, tree, model_selection
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
import datetime as dt
import ml_insights as mli
%matplotlib inline
```

In [2]:
```
sns.set_style('whitegrid')
```

Dataset Description

- date time year-month-day hour:minute:second
- TotalConsmp (AC+TV+LED+Peripherals), energy use in Wh (Target)
- R1, Temperature in Room 1, in Celsius
- H_1, Humidity Room 1, in %
- R2, Temperature in Room 2, in Celsius
- H_2, Humidity in Room 2, in %
- R3, Temperature in Room 3, in Celsius
- H_3, Humidity in Room 3, in %
- R4, Temperature Room 4, in Celsius
- H_4, Humidity in Room 4, in %
- R5, Temperature in Room 5, in Celsius
- H_5, Humidity in Room 5, in %
- R6, Temperature Room 6, in Celsius
- H_6, Humidity in Room 6, in %
- R7, Temperature in Room 7, in Celsius
- H_7, Humidity in Room 7, in %
- R8, Temperature in Room 8, in Celsius
- H_8, Humidity in Room 8, in %
- R9, Temperature in Room 9, in Celsius
- H_9, Humidity in Room 9, in %
- To, Temperature outside, in Celsius
- Pressure outside, in mm Hg
- RH_out, Humidity outside, in %
- Windspeed, in m/s
- Visibility, in km

In [3]:
```
energy_data = pd.read_csv('data.csv')
```

In [4]:
```
energy_data.head()
```

Out[4]:

	date	TotalConsmp	R1	H_1	R2	H_2	R3	H_3	R4	H_
0	1/11/2016 17:00	90	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	45.5666
1	1/11/2016 17:10	90	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	45.9925
2	1/11/2016 17:20	80	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	45.8900
3	1/11/2016 17:30	90	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	45.7233
4	1/11/2016 17:40	100	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	45.5300

5 rows × 25 columns

In [5]:
```
energy_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19735 entries, 0 to 19734
Data columns (total 25 columns):
date           19735 non-null object
TotalConsmp    19735 non-null int64
R1             19735 non-null float64
H_1            19735 non-null float64
R2             19735 non-null float64
H_2            19735 non-null float64
R3             19735 non-null float64
H_3            19735 non-null float64
R4             19735 non-null float64
H_4            19735 non-null float64
R5             19735 non-null float64
H_5            19735 non-null float64
R6             19735 non-null float64
H_6            19735 non-null float64
R7             19735 non-null float64
H_7            19735 non-null float64
R8             19735 non-null float64
H_8            19735 non-null float64
R9             19735 non-null float64
H_9            19735 non-null float64
TempOutSide    19735 non-null float64
Press_mm_hg    19735 non-null float64
H_OutSide      19735 non-null float64
Windspeed      19735 non-null float64
Visibility     19735 non-null float64
dtypes: float64(23), int64(1), object(1)
memory usage: 3.8+ MB
```

In [6]:
```
energy_data.describe()
```

Out[6]:

	TotalConsmp	R1	H_1	R2	H_2	R3	
count	19735.000000	19735.000000	19735.000000	19735.000000	19735.000000	19735.000000	19735.(
mean	101.496833	21.686571	40.259739	20.341219	40.420420	22.267611	39.:
std	104.380829	1.606066	3.979299	2.192974	4.069813	2.006111	3.:
min	10.000000	16.790000	27.023333	16.100000	20.463333	17.200000	28.:
25%	50.000000	20.760000	37.333333	18.790000	37.900000	20.790000	36.!
50%	60.000000	21.600000	39.656667	20.000000	40.500000	22.100000	38.:
75%	100.000000	22.600000	43.066667	21.500000	43.260000	23.290000	41.:
max	1110.000000	26.260000	63.360000	29.856667	56.026667	29.236000	50.

8 rows × 24 columns

Adding Features

In [7]:
```
energy_data['datetime'] = pd.to_datetime(energy_data.date)
```

In [8]:
```
energy_data['date_only'] = energy_data.datetime.dt.date
```

In [9]:
```
energy_data['weekday'] = energy_data.datetime.dt.weekday
energy_data['month'] = energy_data.datetime.dt.month
energy_data['week'] = energy_data.datetime.dt.week
```

In [10]:
```
energy_data['hour'] = energy_data.datetime.dt.hour
```

In [11]:
```
#Assumptions for day and night- sunrise at 6 am and sunset at 6 pm
energy_data['is_day'] = energy_data.apply(lambda x: 1 if x.hour >= 6 anc
```

In [12]:
```
#sanity check
energy_data.hour.loc[energy_data.is_day == 0].unique()
```

Out[12]: array([18, 19, 20, 21, 22, 23, 0, 1, 2, 3, 4, 5])

In [13]:
```
def time_of_day(x):
    '''A function to flag different times of the day'''

    if x.hour > 3 and x.hour < 12:
        return 'Morning'
    elif x.hour >= 12 and x.hour < 16:
        return 'Afternoon'
    elif x.hour >= 16 and x.hour < 20:
        return 'Evening'
    else:
        return 'Night'
```

In [14]:
```
energy_data['time_of_day'] = energy_data.apply(lambda x: time_of_day(x),
```

In [15]:
```
# Sanity check
energy_data.hour.loc[energy_data.time_of_day == 'Night'].unique()
```

Out[15]:
```
array([20, 21, 22, 23,  0,  1,  2,  3])
```

In [16]:
```
energy_data['is_weekend'] = energy_data.apply(lambda x: 1 if x.weekday >
```

In [17]:
```
# Sanity Check
energy_data.weekday.loc[energy_data.is_weekend==1].unique()
```

Out[17]:
```
array([5, 6])
```

Mean room temperature and humidity - average of temperatures and humidities across all rooms

In [18]:
```
energy_data['mean_room_temp'] = energy_data[['R1','R2','R3','R4','R5','R
```

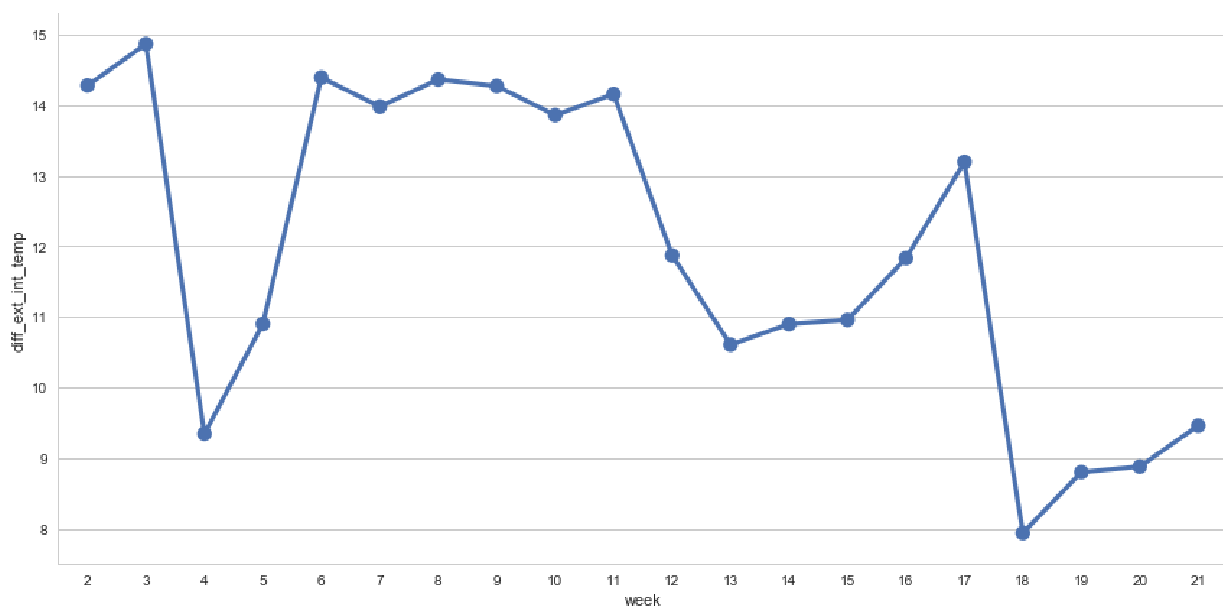
In [19]:
```
energy_data['average_room_humidity'] = energy_data[['H_1','H_2','H_3','H
```

In [20]:
```
energy_data['diff_ext_int_temp'] = energy_data.mean_room_temp - energy_d
```

Plotting the difference between external and internal temperatures across weeks to check to for trends

In [21]:
```
sns.factorplot(x='week',y='diff_ext_int_temp',data=energy_data,ci=None,s
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x10f18ce80>
```



Heating Degree Day

If the day's average temperature is lower than 18 C (65 F), then that day is tagged as a heating degree day. It means that the building/room has to be heated to be habitable

In [22]:
```
# Creating the Heating Degree Day (HDD)
energy_data['TempOutside_F'] = (energy_data['TempOutSide'] *1.8) + 32

daily_min_max = energy_data.groupby('date_only')['TempOutside_F'].agg([

daily_min_max['average'] = 0.5 * (daily_min_max['min'] + daily_min_max[

daily_min_max.columns = ['date_only','min_temp_F','max_temp_F','average_

daily_min_max['is_HDD'] = daily_min_max.apply(lambda x: 1 if x.average_t
```

In [23]:
```
daily_min_max.loc[daily_min_max.is_HDD==0]
```

Out[23]:

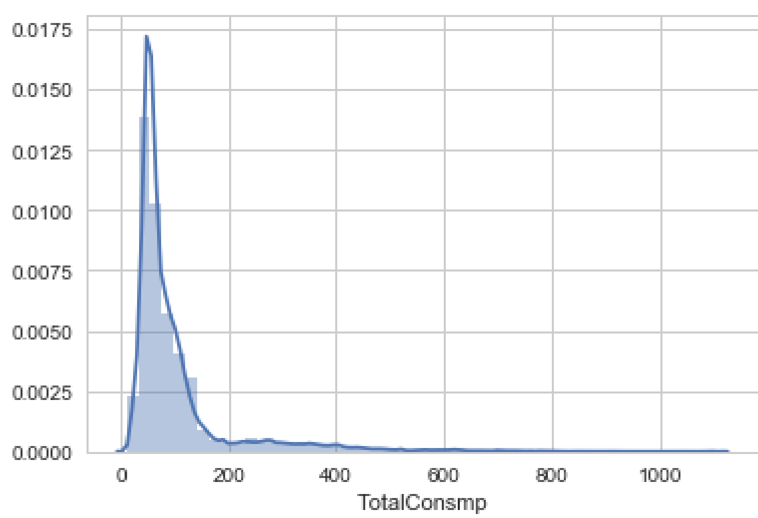
	date_only	min_temp_F	max_temp_F	average_temp_F	is_HDD
117	2016-05-07	52.34	77.72	65.03	0
118	2016-05-08	53.60	78.98	66.29	0
121	2016-05-11	57.74	74.66	66.20	0

In [24]:
```
energy_data = energy_data.merge(daily_min_max,on='date_only',how='left')
```

Checking the distributions of the different features

In [25]:
```
sns.distplot(energy_data.TotalConsmp)
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x112671ba8>



It seems like majority of the energy readings are below 200 Wh. Let us see the time of the day during which the higher energy reading are observed

In [26]:

```
energy_data.loc[energy_data.TotalConsmp > 800]
```

Out[26]:

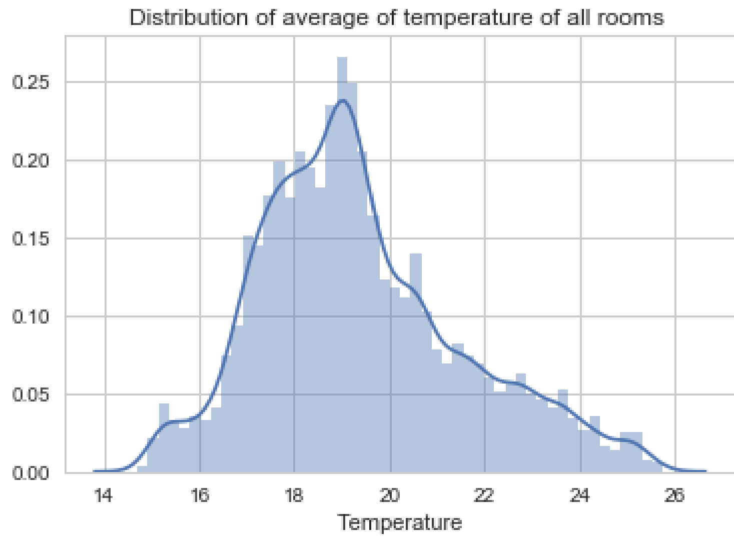
	date	TotalConsmp	R1	H_1	R2	H_2	R3	H_3
432	1/14/2016 17:00	910	21.463333	41.693333	20.856667	38.363333	21.666667	43.930000
731	1/16/2016 18:50	1110	21.930000	42.766667	21.040000	38.080000	20.700000	40.633333
867	1/17/2016 17:30	810	21.500000	36.760000	20.100000	36.077273	20.790000	37.260000
1307	1/20/2016 18:50	840	17.600000	38.090000	16.960000	38.030000	17.790000	37.390000
1451	1/21/2016 18:50	1100	19.600000	34.300000	18.426667	33.963333	18.390000	36.930000
1452	1/21/2016 19:00	910	19.730000	37.863333	18.566667	34.090000	18.390000	36.863333
1821	1/24/2016 8:30	870	17.600000	44.166667	16.926667	43.230000	18.230000	43.290000
1823	1/24/2016 8:50	810	17.790000	45.490000	17.133333	43.566667	18.596667	45.693333
9031	3/14/2016 10:10	860	20.000000	32.730000	19.326667	33.266667	20.033333	34.696667
10668	3/25/2016 19:00	890	23.200000	40.530000	20.856667	41.030000	25.633333	38.260000
12088	4/4/2016 15:40	900	23.000000	43.166667	22.200000	40.426667	26.100000	38.930000
13821	4/16/2016 16:30	820	22.500000	47.866667	21.230000	45.656667	25.033333	39.790000
14693	4/22/2016 17:50	870	22.890000	37.590000	22.267500	35.740000	27.163333	38.066667
15798	4/30/2016 10:00	840	21.390000	37.700000	20.133333	38.466667	22.700000	36.590000
19541	5/26/2016 9:50	820	23.600000	44.260000	27.408571	35.771429	24.890000	40.193333
19582	5/26/2016 16:40	850	24.356667	43.626667	25.390000	36.795714	26.000000	37.333333

16 rows × 42 columns

From the above table, we can see that the high energy consumptions are occuring mostly during the evenings. Some readings are measured even during the morning. It could be due to the usage of high wattage devices like hair dryers

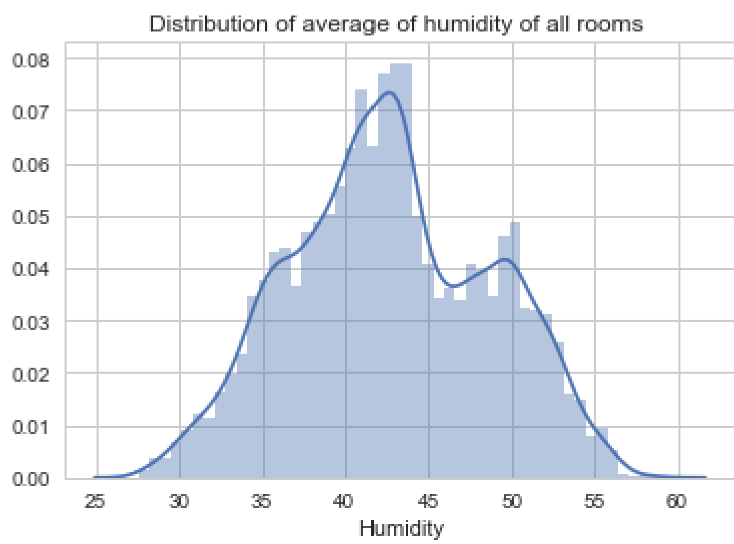
In [27]:
```
sns.distplot(energy_data.mean_room_temp)
plt.title('Distribution of average of temperature of all rooms ')
plt.xlabel('Temperature')
```

Out[27]: <matplotlib.text.Text at 0x1114ba160>



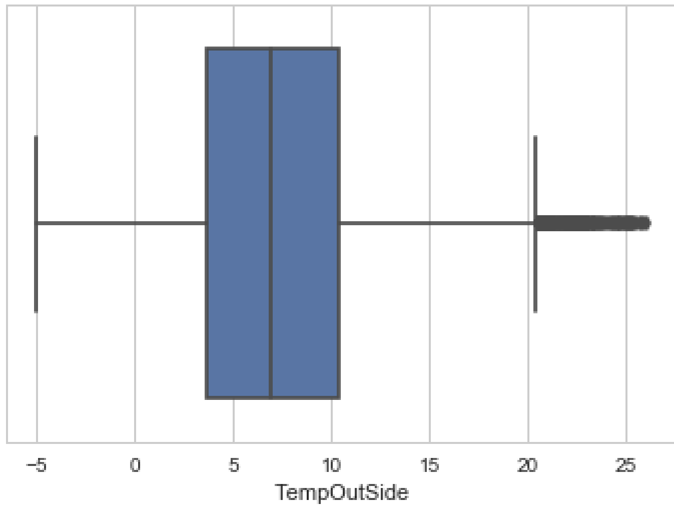
In [28]:
```
sns.distplot(energy_data.average_room_humidity)
plt.title('Distribution of average of humidity of all rooms ')
plt.xlabel('Humidity')
```

Out[28]: <matplotlib.text.Text at 0x110f539e8>



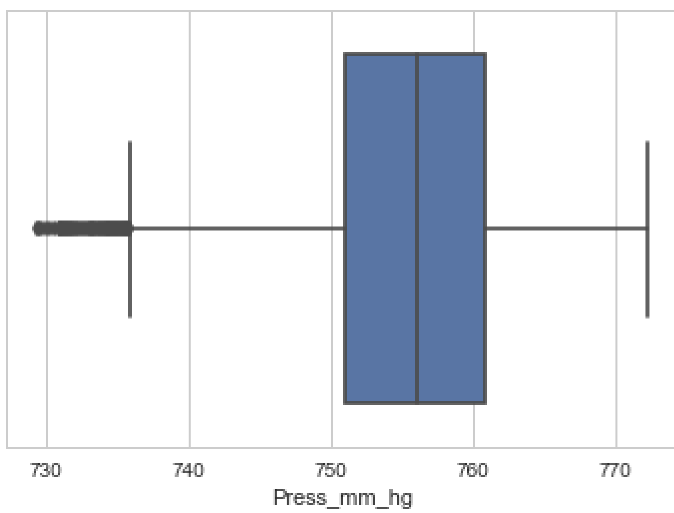
In [29]:
```
1 sns.boxplot(energy_data.TempOutSide)
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x10f991f98>

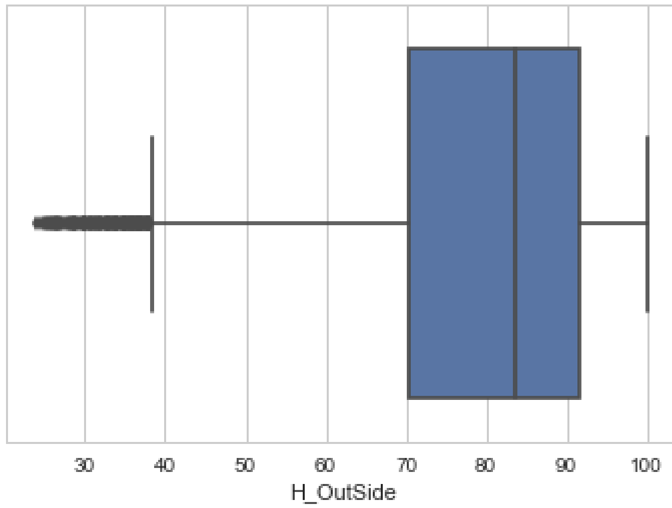


In [30]:
```
1 sns.boxplot(energy_data.Press_mm_hg)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x10f4baba8>

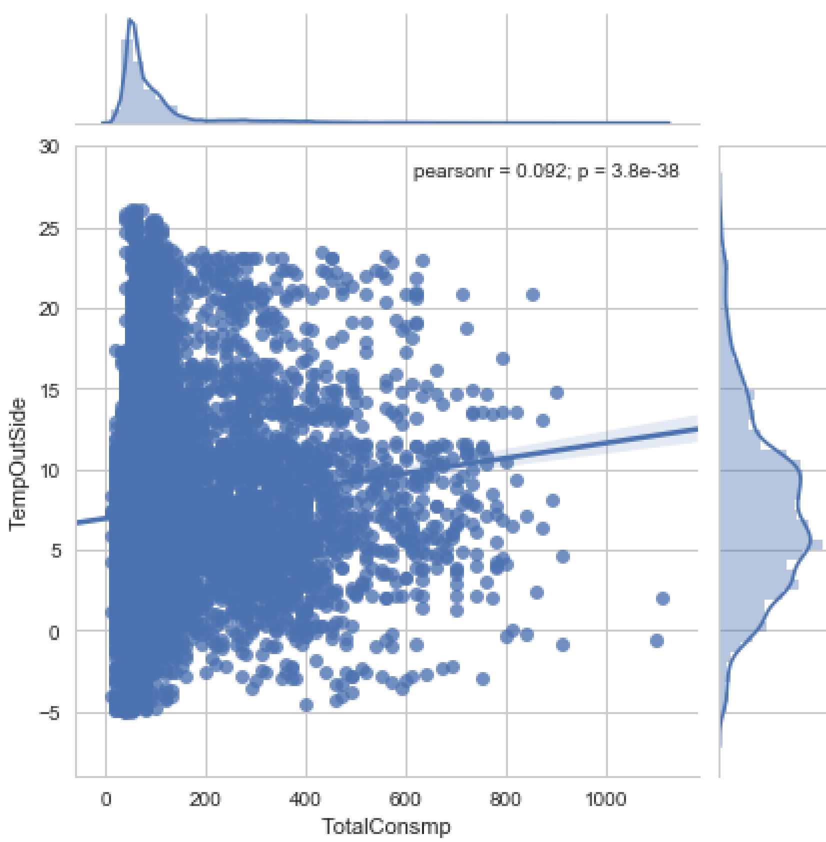


In [31]:
```
sns.boxplot(energy_data.H_OutSide)
```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x110b2a390>



Checking for correlation between the features and Total Consumption

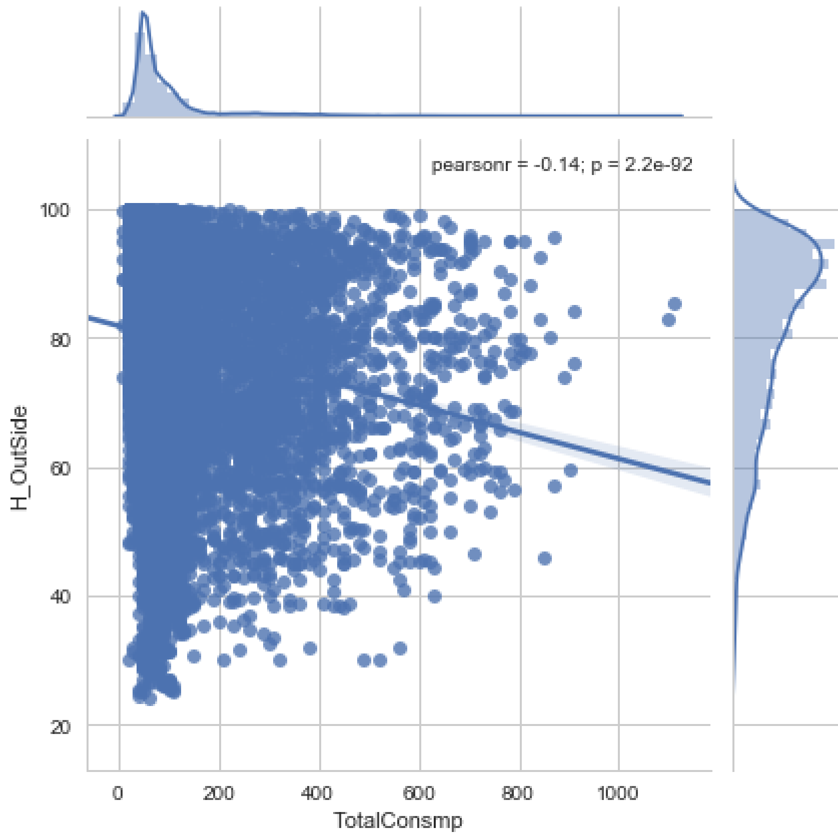
In [32]:
```
# External Temperature vs Total Consumption
sns.jointplot(energy_data.TotalConsmp, energy_data.TempOutSide,kind='re
```
Out[32]: <seaborn.axisgrid.JointGrid at 0x1107acb70>



In [33]:

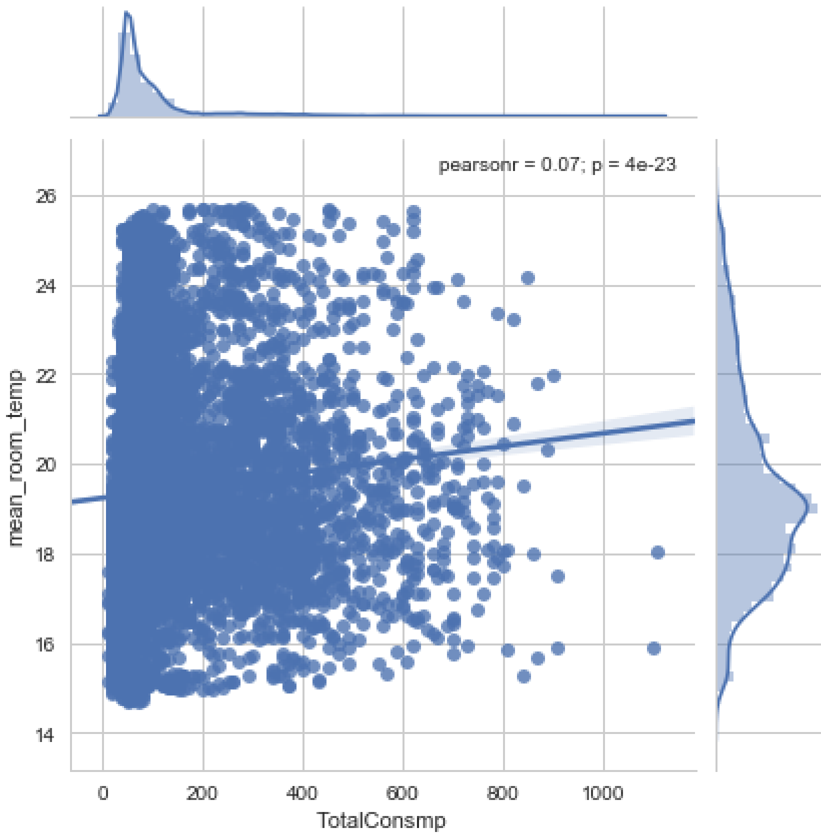
```
# External Humidity vs Total Consumption
sns.jointplot(energy_data.TotalConsmp, energy_data.H_OutSide, kind='reg
```

Out[33]: <seaborn.axisgrid.JointGrid at 0x1124355c0>



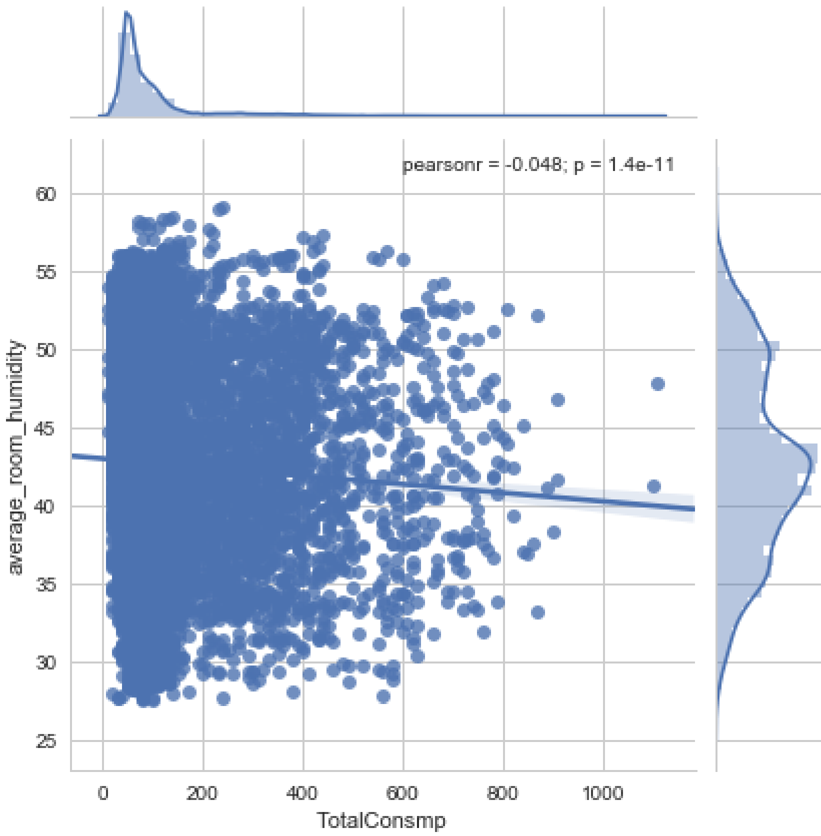
In [34]:
```
# Mean Room Temperature vs Total Consumption
sns.jointplot(energy_data.TotalConsmp, energy_data.mean_room_temp, kind=
```

Out[34]: <seaborn.axisgrid.JointGrid at 0x1105d9048>



In [35]:
```
# Average Room Humidity vs Total Consumption
sns.jointplot(energy_data.TotalConsmp, energy_data.average_room_humidity
```

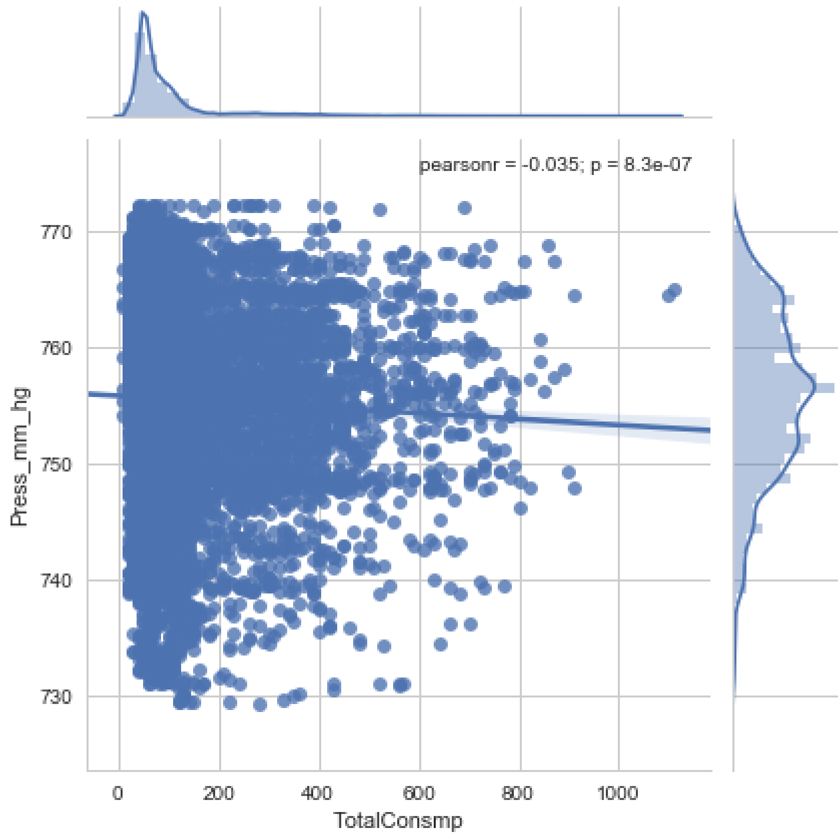
Out[35]: <seaborn.axisgrid.JointGrid at 0x113232400>



In [36]:

```
# Pressure vs Total Consumption
sns.jointplot(energy_data.TotalConsmp, energy_data.Press_mm_hg, kind='re
```

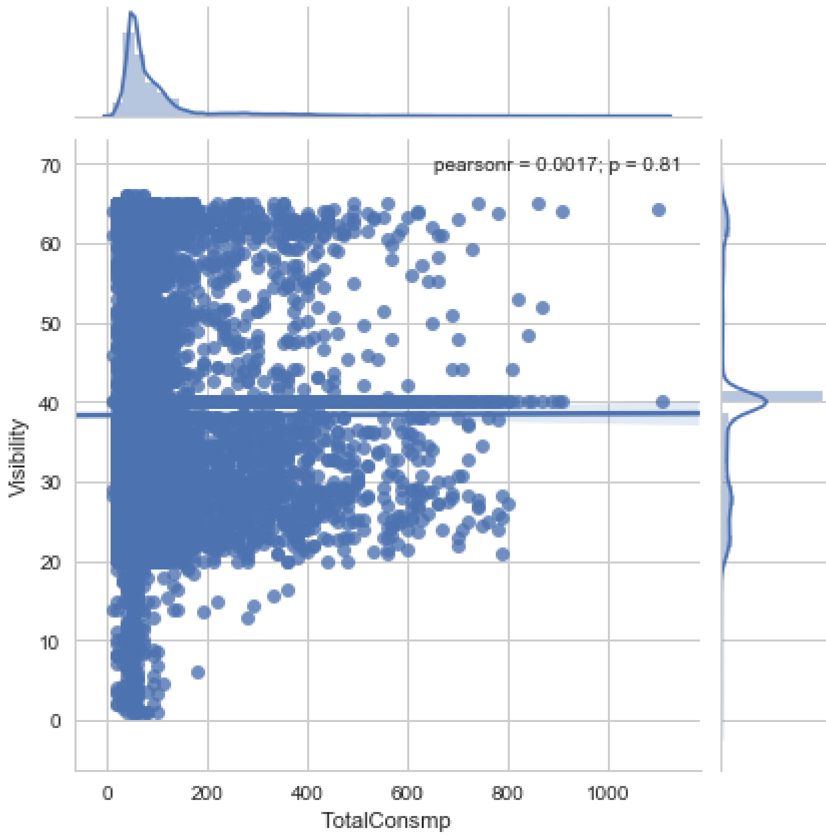
Out[36]: <seaborn.axisgrid.JointGrid at 0x11354b4e0>



In [37]:

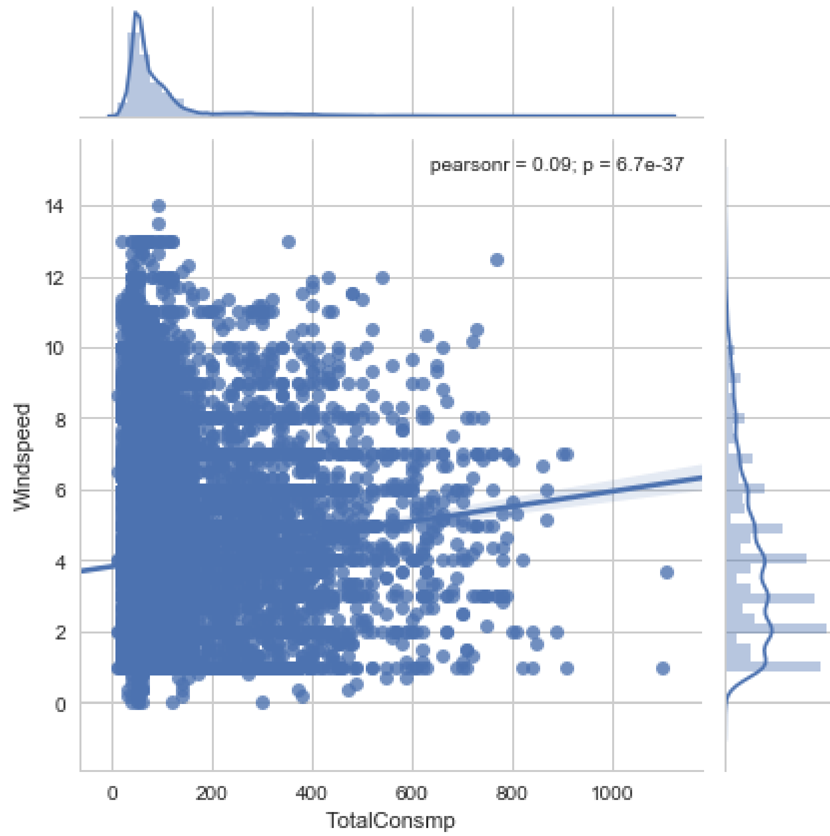
```
# Visibility vs Total Consumption
sns.jointplot(energy_data.TotalConsmp, energy_data.Visibility, kind='reg
```

Out[37]: <seaborn.axisgrid.JointGrid at 0x113f1a0b8>



In [38]:
```
1 # Windspeed vs Total Consumption
2 sns.jointplot(energy_data.TotalConsmp, energy_data.Windspeed, kind='reg
```

Out[38]: <seaborn.axisgrid.JointGrid at 0x1127b3e10>



From the above plots , we can see that Total Consumption has slight positive relationship with-

- External Temperature
- Mean Room Temperature
- Windspeed

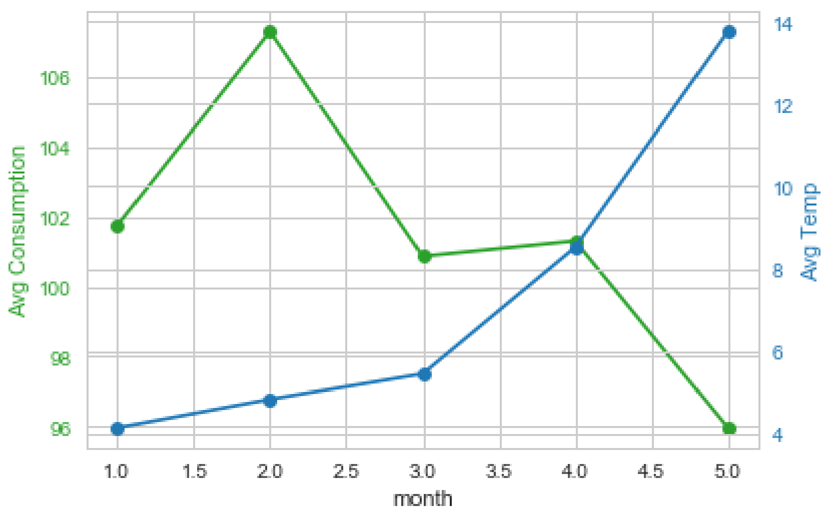
Overall Trends

Checking monthly trends

In [39]:
```
1 overall_month_data = energy_data.groupby('month')['TotalConsmp','TempOut
```

In [40]:

```
# plotting monthly average total consumption vs monthly average externa

fig, ax1 = plt.subplots()

color = 'tab:green'
ax1.set_xlabel('month')
ax1.set_ylabel('Avg Consumption', color=color)
ax1.plot(overall_month_data.month, overall_month_data.TotalConsmp, colo
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()

color = 'tab:blue'
ax2.set_ylabel('Avg Temp', color=color)
ax2.plot(overall_month_data.month, overall_month_data.TempOutSide, colo
ax2.tick_params(axis='y', labelcolor=color)

# plt.savefig('tempvscons.png',dpi=300,bbox_inches='tight')
```

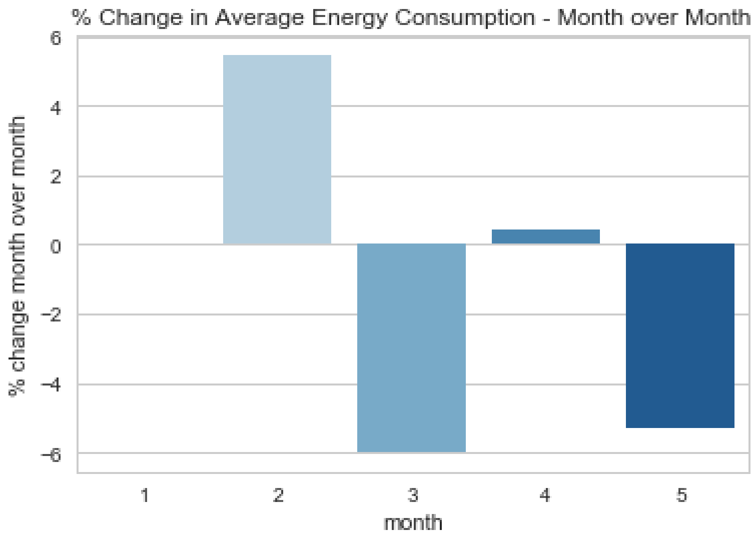


In [41]:

```
monthly_consump = energy_data.groupby('month')['TotalConsmp'].agg(['mea

monthly_consump.columns = ['month','avg_consmp']

monthly_consump['%_change_avg'] = monthly_consump.avg_consmp.pct_change(
```

In [42]:
```
sns.barplot(x='month',y='%_change_avg',data=monthly_consump,palette='Blu
plt.title('% Change in Average Energy Consumption - Month over Month')
plt.ylabel('% change month over month')
```

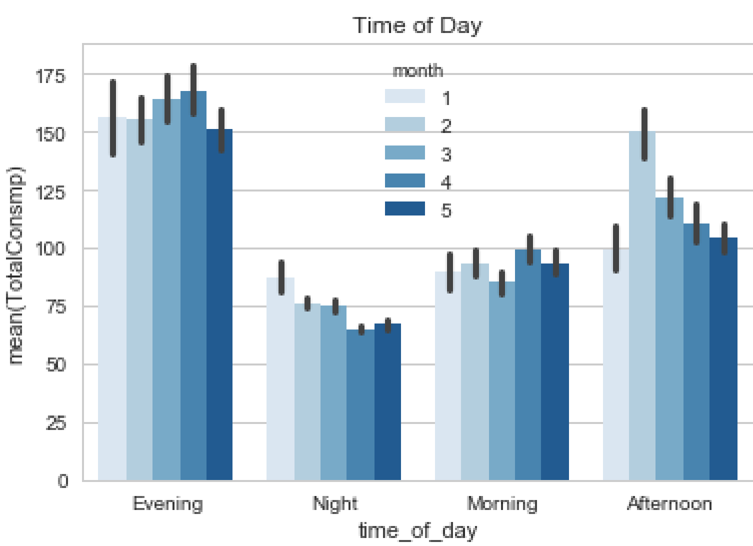
Out[42]: <matplotlib.text.Text at 0x114b120b8>



Here we see that at the turn of the season, there are dips in the monthly average energy consumption. Thus, increasing external temperature reduces energy consumption

In [43]:
```
sns.barplot(x='time_of_day',y='TotalConsmp',hue='month',data=energy_data
plt.title('Time of Day')
```

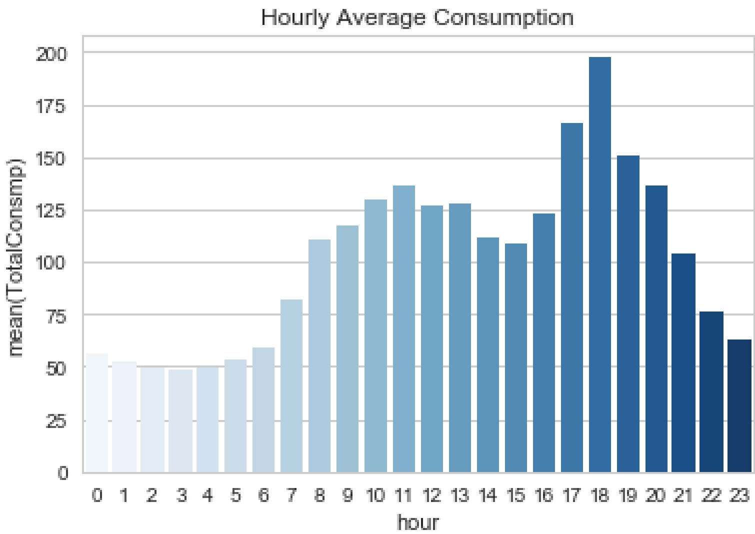
Out[43]: <matplotlib.text.Text at 0x11497df28>



Evenings register higher energy consumptions. This could be due to the fact that guests mostly return to their rooms during this time

In [44]:
```
sns.barplot(x='hour',y='TotalConsmp',data=energy_data,\
                  ci=None,estimator=np.mean,palette='Blues')
plt.title('Hourly Average Consumption')
```

Out[44]: <matplotlib.text.Text at 0x114ec6630>

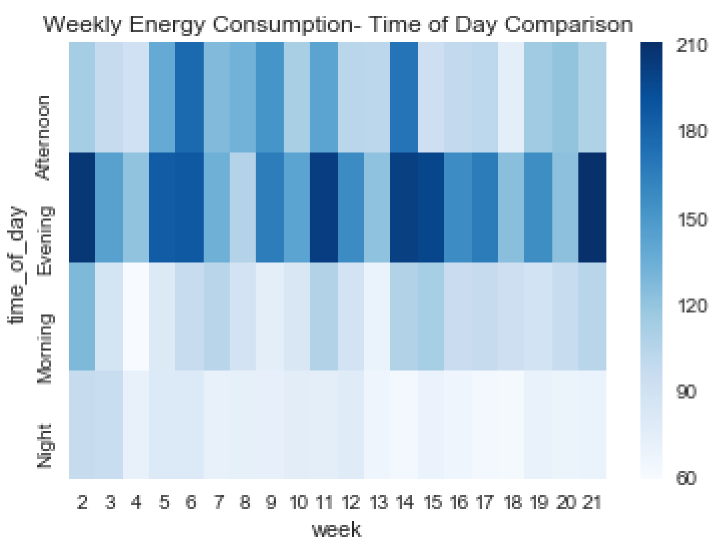


Peak consumption is at 5 and 6 PM

In [45]:
```
# Checking weekly time of day consumptions
weekly_table = pd.pivot_table(energy_data,index='time_of_day',values='To
```

In [46]:
```
sns.heatmap(weekly_table,cmap='Blues')
plt.title('Weekly Energy Consumption- Time of Day Comparison')
```

Out[46]: <matplotlib.text.Text at 0x11501c898>



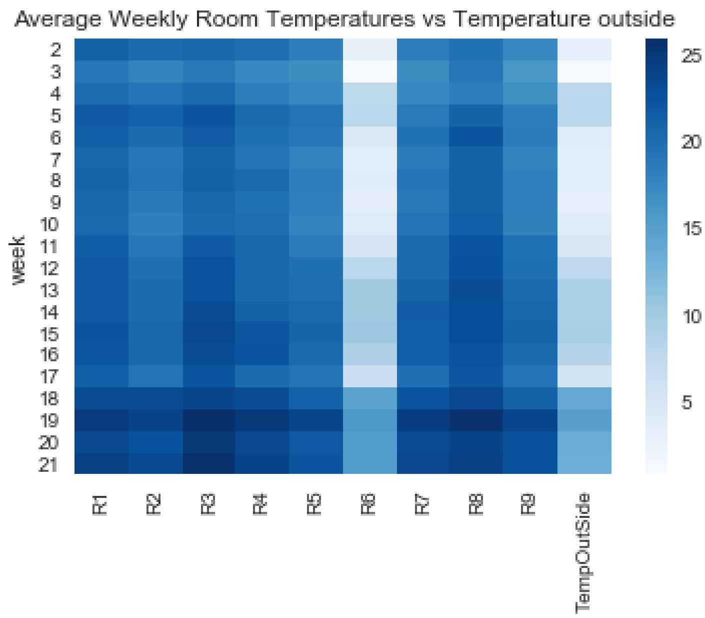
Checking for trends in Room Temperatures and Humidities

In [47]:
```
room_temp_weekly = pd.pivot_table(energy_data,index='week',values=['R1',
                                                                    ,'Ter
```

In [48]:
```
humidity_weekly = pd.pivot_table(energy_data,index='week',values=['H_1',
                                                                   ,'H_8
```

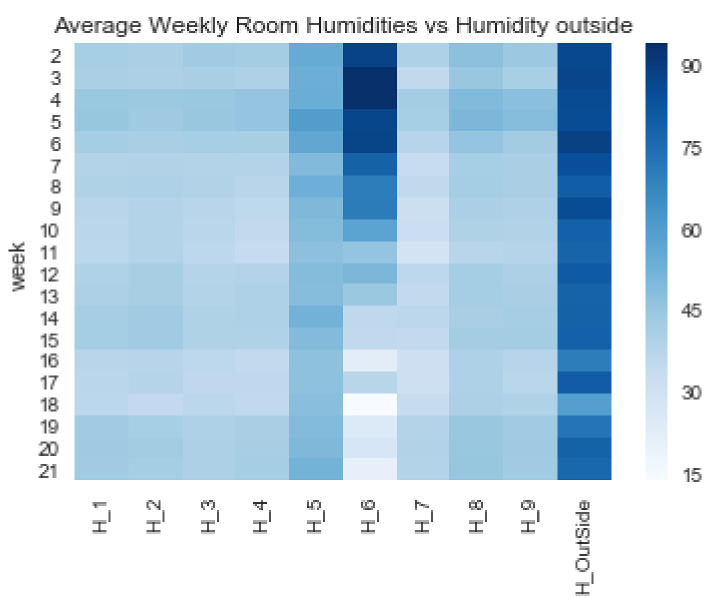
In [49]:
```
sns.heatmap(room_temp_weekly,cmap='Blues')
plt.title('Average Weekly Room Temperatures vs Temperature outside')
```

Out[49]: <matplotlib.text.Text at 0x1152186a0>



In [50]:
```
sns.heatmap(humidity_weekly,cmap='Blues')
plt.title('Average Weekly Room Humidities vs Humidity outside')
```

Out[50]: <matplotlib.text.Text at 0x1156cd518>



From the above graphs, that Room 6 appears to be unoccupied

Modeling

In [51]:
```
X = energy_data[['mean_room_temp','average_room_humidity','Windspeed',
                 'is_day','is_HDD','TempOutSide', 'Press_mm_hg', 'H_OutS
y = energy_data.TotalConsmp
```

Doing train test split of 80:20 of the dataset

In [52]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2(
```

Checking the perfomance of multiple regression models using mean squared error and r2 score to determine the best model

In [53]:
```
models = {}
parameters = {}

models['linear_model'] = linear_model.LinearRegression()
models['ridge_model'] = linear_model.Ridge()
models['lasso_model'] = linear_model.Lasso(alpha=.5)
models['robust_regression'] = linear_model.SGDRegressor(loss='huber',n_i
models['eps_insensitive'] = linear_model.SGDRegressor(loss='epsilon_inse


models['cart'] = tree.DecisionTreeRegressor(max_depth=7)
models['extratrees'] = tree.ExtraTreeRegressor(max_depth=7)
models['randomForest'] = ensemble.RandomForestRegressor()
models['adaboostedTrees'] = ensemble.AdaBoostRegressor()
models['gradboostedTrees'] = ensemble.GradientBoostingRegressor()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2(

for name,model in models.items():
#     scores = model_selection.cross_val_score(model, X, y, scoring='neg
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    print('Model: '+name)
    print("RMSE: " + str(mean_squared_error(y_test,y_pred)))
    print('R-squared: '+str(r2_score(y_test,y_pred)))
    print()
```

```
/Users/chinnu/anaconda/lib/python3.6/site-packages/scipy/linalg/basic.py:
1018: RuntimeWarning: internal gelsd driver lwork query error, required i
work dimension not returned. This is likely the result of LAPACK bug 003
8, fixed in LAPACK 3.2.2 (released July 21, 2010). Falling back to 'gels
s' driver.
  warnings.warn(mesg, RuntimeWarning)

Model: linear_model
RMSE: 9533.51055309
R-squared: 0.0849657937493

Model: ridge_model
RMSE: 9533.41839595
R-squared: 0.0849746390675

Model: lasso_model
RMSE: 9529.91746133
R-squared: 0.0853106616602

Model: robust_regression
RMSE: 10390.6171182
R-squared: 0.00269999867527

Model: eps_insensitive
RMSE: 10235.6221166
R-squared: 0.0175765467686

Model: cart
RMSE: 8492.12136572
R-squared: 0.184919186904
```

```
Model: extratrees
RMSE: 8689.9048486
R-squared: 0.165935765083

Model: randomForest
RMSE: 4474.10818343
R-squared: 0.570571406251

Model: adaboostedTrees
RMSE: 23489.471461
R-squared: -1.25453884525

Model: gradboostedTrees
RMSE: 7867.88270025
R-squared: 0.244834128896
```

Based on the mean squared error and the r2 score, the Random Forest Regressor performs best on the test set

Using the linear regression model to understand how the different variables affect the energy consumption (positive or negative effect)

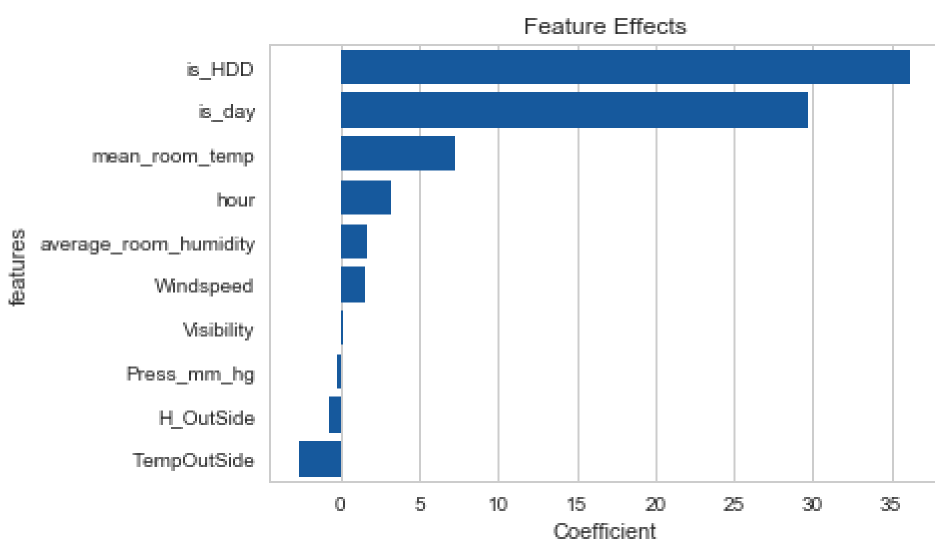
In [54]:

```
lr = linear_model.LinearRegression()
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
```

In [55]:

```
feature_coeff = pd.DataFrame(sorted(list(zip(X.columns,lr.coef_)),key=la

feature_coeff.columns = ['features','coefficient']
sns.barplot(x= 'coefficient',y='features',data=feature_coeff,color='#005
plt.title('Feature Effects')
plt.xlabel('Coefficient')
```

Out[55]: <matplotlib.text.Text at 0x114ea5e80>



Here we can see that-

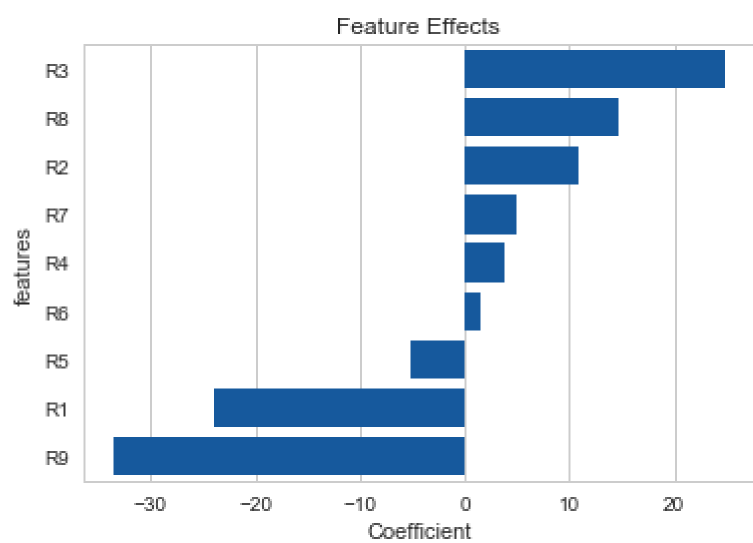
- If a day is flagged as a heating degree day (HDD), the energy consumption is more
- The greater the room temperatures, greater the energy consumption
- Greater the external temperature, lower the energy consumption

Now let us check how the each room is affecting the energy consumption

In [56]:
```
X = energy_data[['R1', 'R2',  'R3',  'R4', 'R5',  'R6',  'R7',  'R8', 'R
y = energy_data.TotalConsmp

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2(

lr = linear_model.LinearRegression()
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
```

In [57]:
```
feature_coeff = pd.DataFrame(sorted(list(zip(X.columns,lr.coef_)),key=la

feature_coeff.columns = ['features','coefficient']

sns.barplot(x= 'coefficient',y='features',data=feature_coeff,color='#005
plt.title('Feature Effects')
plt.xlabel('Coefficient')
```

Out[57]: `<matplotlib.text.Text at 0x1167ba4e0>`



From the above we can conclude that-

- Rooms 3 and 8 consume more energy than all the rooms
- Rooms 1 and 9 uses the lesser energy
- Room 6 has little impact on the energy consumption (confirming that it is empty for the duration of the analysis)

Performing Cross Validation on the Random Forest Model to check if the model is overfitting on the dataset

In [58]:

```
X = energy_data[['mean_room_temp','average_room_humidity','Windspeed',
                 'is_day','is_HDD','TempOutSide', 'Press_mm_hg', 'H_OutS
y = energy_data.TotalConsmp
```

In [59]:

```
kf = KFold(n_splits=10,shuffle=True)
MSE = []
for train_index, test_index in kf.split(X):

    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    final_model = ensemble.RandomForestRegressor()

    final_model.fit(X_train,y_train)
    y_pred = final_model.predict(X_test)
    MSE.append(np.sqrt(np.mean((y_pred - y_test)**2)))

cross_val = model_selection.cross_val_score(final_model, X, y, cv=kf,n_
```

In [60]:

```
cross_val
```

Out[60]:

```
array([ 0.61338541,  0.56044623,  0.54039766,  0.58249879,  0.57754078,
        0.58128434,  0.54942106,  0.58662208,  0.57477558,  0.62905436])
```

In [61]:

```
np.mean(cross_val)
```

Out[61]:

```
0.57954262861035377
```

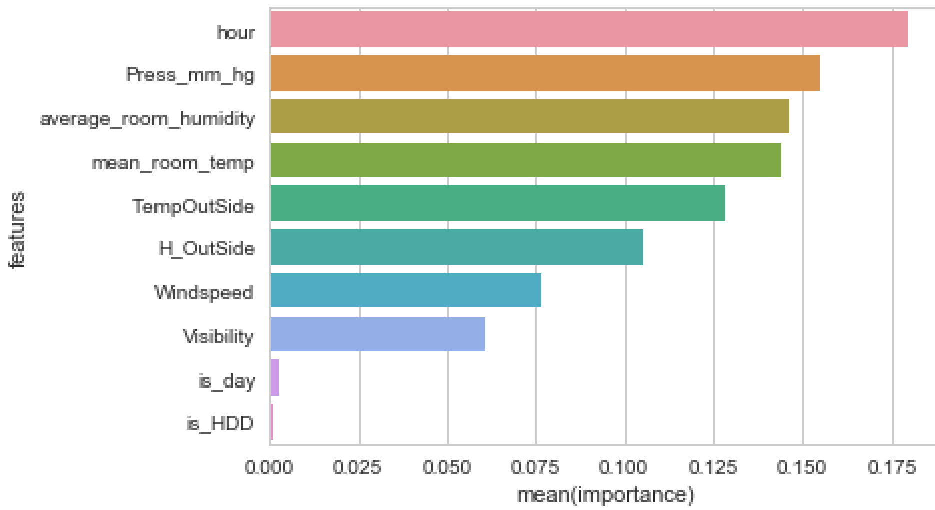
Feature importances- Random Forest Regressor

In [62]:

```
feature_importances = pd.DataFrame(sorted(list(zip(X.columns,final_model
                                           key=lambda x: x[1], reverse=Tr
feature_importances.columns = ['features','importance']
```

In [63]:
```
1 sns.barplot(x='importance',y='features',data=feature_importances)
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x11683aac8>



In [ ]:
```
1
```