In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

In [2]:

```
#Fetch data from file
my_data = open(r"C:\\Users\\user\\python_example\\HomeC.csv")
df = pd.read_csv(my_data)
df
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_9040\188017517.py:3: DtypeWarnin
g: Columns (0,27) have mixed types. Specify dtype option on import or set lo
w_memory=False.
  df = pd.read_csv(my_data)
```

Out[2]:

| | time | use [kW] | gen [kW] | House overall [kW] | Dishwasher [kW] | Furnace 1 [kW] | Furnace 2 [kW] | Home office [kW] | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1451624400 | 0.932833 | 0.003483 | 0.932833 | 0.000033 | 0.020700 | 0.061917 | 0.442633 | 0. |
| **1** | 1451624401 | 0.934333 | 0.003467 | 0.934333 | 0.000000 | 0.020717 | 0.063817 | 0.444067 | 0. |
| **2** | 1451624402 | 0.931817 | 0.003467 | 0.931817 | 0.000017 | 0.020700 | 0.062317 | 0.446067 | 0. |
| **3** | 1451624403 | 1.022050 | 0.003483 | 1.022050 | 0.000017 | 0.106900 | 0.068517 | 0.446583 | 0. |
| **4** | 1451624404 | 1.139400 | 0.003467 | 1.139400 | 0.000133 | 0.236933 | 0.063983 | 0.446533 | 0. |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **503906** | 1452128306 | 1.599333 | 0.003233 | 1.599333 | 0.000050 | 0.104017 | 0.625033 | 0.041750 | 0. |
| **503907** | 1452128307 | 1.924267 | 0.003217 | 1.924267 | 0.000033 | 0.422383 | 0.637733 | 0.042033 | 0. |
| **503908** | 1452128308 | 1.978200 | 0.003217 | 1.978200 | 0.000050 | 0.495667 | 0.620367 | 0.042100 | 0. |
| **503909** | 1452128309 | 1.990950 | 0.003233 | 1.990950 | 0.000050 | 0.494700 | 0.634133 | 0.042100 | 0. |
| **503910** | \ | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

503911 rows × 32 columns

In [3]:
```
df.head(10)
```

Out[3]:

| | time | use [kW] | gen [kW] | House overall [kW] | Dishwasher [kW] | Furnace 1 [kW] | Furnace 2 [kW] | Home office [kW] | Fridg [kW |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1451624400 | 0.932833 | 0.003483 | 0.932833 | 0.000033 | 0.020700 | 0.061917 | 0.442633 | 0.12415 |
| **1** | 1451624401 | 0.934333 | 0.003467 | 0.934333 | 0.000000 | 0.020717 | 0.063817 | 0.444067 | 0.12400 |
| **2** | 1451624402 | 0.931817 | 0.003467 | 0.931817 | 0.000017 | 0.020700 | 0.062317 | 0.446067 | 0.12353 |
| **3** | 1451624403 | 1.022050 | 0.003483 | 1.022050 | 0.000017 | 0.106900 | 0.068517 | 0.446583 | 0.12313 |
| **4** | 1451624404 | 1.139400 | 0.003467 | 1.139400 | 0.000133 | 0.236933 | 0.063983 | 0.446533 | 0.12285 |
| **5** | 1451624405 | 1.391867 | 0.003433 | 1.391867 | 0.000283 | 0.503250 | 0.063667 | 0.447033 | 0.12230 |
| **6** | 1451624406 | 1.366217 | 0.003450 | 1.366217 | 0.000283 | 0.499400 | 0.063717 | 0.443267 | 0.12205 |
| **7** | 1451624407 | 1.431900 | 0.003417 | 1.431900 | 0.000250 | 0.477867 | 0.178633 | 0.444283 | 0.12180 |
| **8** | 1451624408 | 1.627300 | 0.003417 | 1.627300 | 0.000183 | 0.447650 | 0.365700 | 0.441467 | 0.12161 |
| **9** | 1451624409 | 1.735383 | 0.003417 | 1.735383 | 0.000017 | 0.171550 | 0.682500 | 0.438733 | 0.12163 |

10 rows × 32 columns

In [4]:
```
df.tail(10)
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **503904** | 1452128304 | 1.608867 | 0.003217 | 1.608867 | 0.000033 | 0.114300 | 0.623283 | 0.041817 |
| **503905** | 1452128305 | 1.601233 | 0.003183 | 1.601233 | 0.000050 | 0.085267 | 0.642417 | 0.041783 |
| **503906** | 1452128306 | 1.599333 | 0.003233 | 1.599333 | 0.000050 | 0.104017 | 0.625033 | 0.041750 |
| **503907** | 1452128307 | 1.924267 | 0.003217 | 1.924267 | 0.000033 | 0.422383 | 0.637733 | 0.042033 |
| **503908** | 1452128308 | 1.978200 | 0.003217 | 1.978200 | 0.000050 | 0.495667 | 0.620367 | 0.042100 |
| **503909** | 1452128309 | 1.990950 | 0.003233 | 1.990950 | 0.000050 | 0.494700 | 0.634133 | 0.042100 |
| **503910** | \ | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

10 rows × 32 columns

In [5]:

```
# this will describe all the numeric values on the dataset
df.describe()
```

Out[5]:

| | use [kW] | gen [kW] | House overall [kW] | Dishwasher [kW] | Furnace 1 [kW] | Furnace [k |
|---|---|---|---|---|---|---|
| **count** | 503910.000000 | 503910.000000 | 503910.000000 | 503910.000000 | 503910.000000 | 503910.0000 |
| **mean** | 0.858962 | 0.076229 | 0.858962 | 0.031368 | 0.099210 | 0.1367 |
| **std** | 1.058207 | 0.128428 | 1.058207 | 0.190951 | 0.169059 | 0.1786 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000017 | 0.0000 |
| **25%** | 0.367667 | 0.003367 | 0.367667 | 0.000000 | 0.020233 | 0.0644 |
| **50%** | 0.562333 | 0.004283 | 0.562333 | 0.000017 | 0.020617 | 0.0666 |
| **75%** | 0.970250 | 0.083917 | 0.970250 | 0.000233 | 0.068733 | 0.0806 |
| **max** | 14.714567 | 0.613883 | 14.714567 | 1.401767 | 1.934083 | 0.7949 |

8 rows × 28 columns

In [6]:

```
#Check for missing values on the dataset.if zero corresponds with that particu
df.isnull().sum()
```

Out[6]:

```
time                    0
use [kW]                1
gen [kW]                1
House overall [kW]      1
Dishwasher [kW]         1
Furnace 1 [kW]          1
Furnace 2 [kW]          1
Home office [kW]        1
Fridge [kW]             1
Wine cellar [kW]        1
Garage door [kW]        1
Kitchen 12 [kW]         1
Kitchen 14 [kW]         1
Kitchen 38 [kW]         1
Barn [kW]               1
Well [kW]               1
Microwave [kW]          1
Living room [kW]        1
Solar [kW]              1
temperature             1
icon                    1
humidity                1
visibility              1
summary                 1
apparentTemperature     1
pressure                1
windSpeed               1
cloudCover              1
windBearing             1
precipIntensity         1
dewPoint                1
precipProbability       1
dtype: int64
```

In [7]:

```
#This shows you the data types of the values assigned to particular varible
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 503911 entries, 0 to 503910
Data columns (total 32 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   time                 503911 non-null  object 
 1   use [kW]             503910 non-null  float64
 2   gen [kW]             503910 non-null  float64
 3   House overall [kW]   503910 non-null  float64
 4   Dishwasher [kW]      503910 non-null  float64
 5   Furnace 1 [kW]       503910 non-null  float64
 6   Furnace 2 [kW]       503910 non-null  float64
 7   Home office [kW]     503910 non-null  float64
 8   Fridge [kW]          503910 non-null  float64
 9   Wine cellar [kW]     503910 non-null  float64
 10  Garage door [kW]     503910 non-null  float64
 11  Kitchen 12 [kW]      503910 non-null  float64
 12  Kitchen 14 [kW]      503910 non-null  float64
 13  Kitchen 38 [kW]      503910 non-null  float64
 14  Barn [kW]            503910 non-null  float64
 15  Well [kW]            503910 non-null  float64
 16  Microwave [kW]       503910 non-null  float64
 17  Living room [kW]     503910 non-null  float64
 18  Solar [kW]           503910 non-null  float64
 19  temperature          503910 non-null  float64
 20  icon                 503910 non-null  object 
 21  humidity             503910 non-null  float64
 22  visibility           503910 non-null  float64
 23  summary              503910 non-null  object 
 24  apparentTemperature  503910 non-null  float64
 25  pressure             503910 non-null  float64
 26  windSpeed            503910 non-null  float64
 27  cloudCover           503910 non-null  object 
 28  windBearing          503910 non-null  float64
 29  precipIntensity      503910 non-null  float64
 30  dewPoint             503910 non-null  float64
 31  precipProbability    503910 non-null  float64
dtypes: float64(28), object(4)
memory usage: 123.0+ MB
```

In [8]:

```
#drop row with Nan in stipulated column
df = df.dropna(subset=["House overall [kW]","temperature","humidity",
                       "apparentTemperature",
                       "pressure","Fridge [kW]","Microwave [kW]",
                       "Dishwasher [kW]",
                       "Garage door [kW]",
                      "Living room [kW]"])
df
```

Out[8]:

| | time | use [kW] | gen [kW] | House overall [kW] | Dishwasher [kW] | Furnace 1 [kW] | Furnace 2 [kW] | Home office [kW] | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1451624400 | 0.932833 | 0.003483 | 0.932833 | 0.000033 | 0.020700 | 0.061917 | 0.442633 | 0. |
| **1** | 1451624401 | 0.934333 | 0.003467 | 0.934333 | 0.000000 | 0.020717 | 0.063817 | 0.444067 | 0. |
| **2** | 1451624402 | 0.931817 | 0.003467 | 0.931817 | 0.000017 | 0.020700 | 0.062317 | 0.446067 | 0. |
| **3** | 1451624403 | 1.022050 | 0.003483 | 1.022050 | 0.000017 | 0.106900 | 0.068517 | 0.446583 | 0. |
| **4** | 1451624404 | 1.139400 | 0.003467 | 1.139400 | 0.000133 | 0.236933 | 0.063983 | 0.446533 | 0. |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **503905** | 1452128305 | 1.601233 | 0.003183 | 1.601233 | 0.000050 | 0.085267 | 0.642417 | 0.041783 | 0. |
| **503906** | 1452128306 | 1.599333 | 0.003233 | 1.599333 | 0.000050 | 0.104017 | 0.625033 | 0.041750 | 0. |
| **503907** | 1452128307 | 1.924267 | 0.003217 | 1.924267 | 0.000033 | 0.422383 | 0.637733 | 0.042033 | 0. |
| **503908** | 1452128308 | 1.978200 | 0.003217 | 1.978200 | 0.000050 | 0.495667 | 0.620367 | 0.042100 | 0. |
| **503909** | 1452128309 | 1.990950 | 0.003233 | 1.990950 | 0.000050 | 0.494700 | 0.634133 | 0.042100 | 0. |

503910 rows × 32 columns

In [9]:

```
#Split data into features for training
X = df[["temperature","humidity","apparentTemperature","pressure",
        "Fridge [kW]","Microwave [kW]","Dishwasher [kW]",
        "Garage door [kW]","Living room [kW]"]]

Y = df[["House overall [kW]"]]
```

In [10]:

```
#Train and Test splitting
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
                                                     random_state=42)

#Drop Nan rows values
X_train = X_train.dropna()
Y_train = Y_train.loc[X_train.index]

X_test = X_test.dropna()
Y_test = Y_test.loc[X_test.index]

#scale the features using standardscaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#MLP
MLP = MLPRegressor(hidden_layer_sizes=(64, 32), activation='relu', max_iter=20
                   learning_rate_init=0.001, random_state=42)
#Training of Model
MLP.fit(X_train, Y_train)
```

```
C:\Users\user\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer
_perceptron.py:1625: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_samples, ), fo
r example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[10]:

```
▾                                  MLPRegressor

MLPRegressor(hidden_layer_sizes=(64, 32), max_iter=2000, random_state=42)
```

In [11]:

```
#first prediction
X_prediction = pd.DataFrame({
    "temperature" :[36.17],
    "humidity" :[0.20],
    "apparentTemperature":[29.22],
    "pressure" :[1016.90],
    "Fridge [kW]" :[0.123540],
    "Microwave [kW]" : [0.004065],
    "Dishwasher [kW]" :[0.000020],
    "Garage door [kW]" :[0.013081],
    "Living room [kW]" :[0.00162]


})
predict = MLP.predict(X_prediction)
predict
```

```
C:\Users\user\anaconda3\Lib\site-packages\sklearn\base.py:457: UserWarning:
X has feature names, but MLPRegressor was fitted without feature names
  warnings.warn(
```

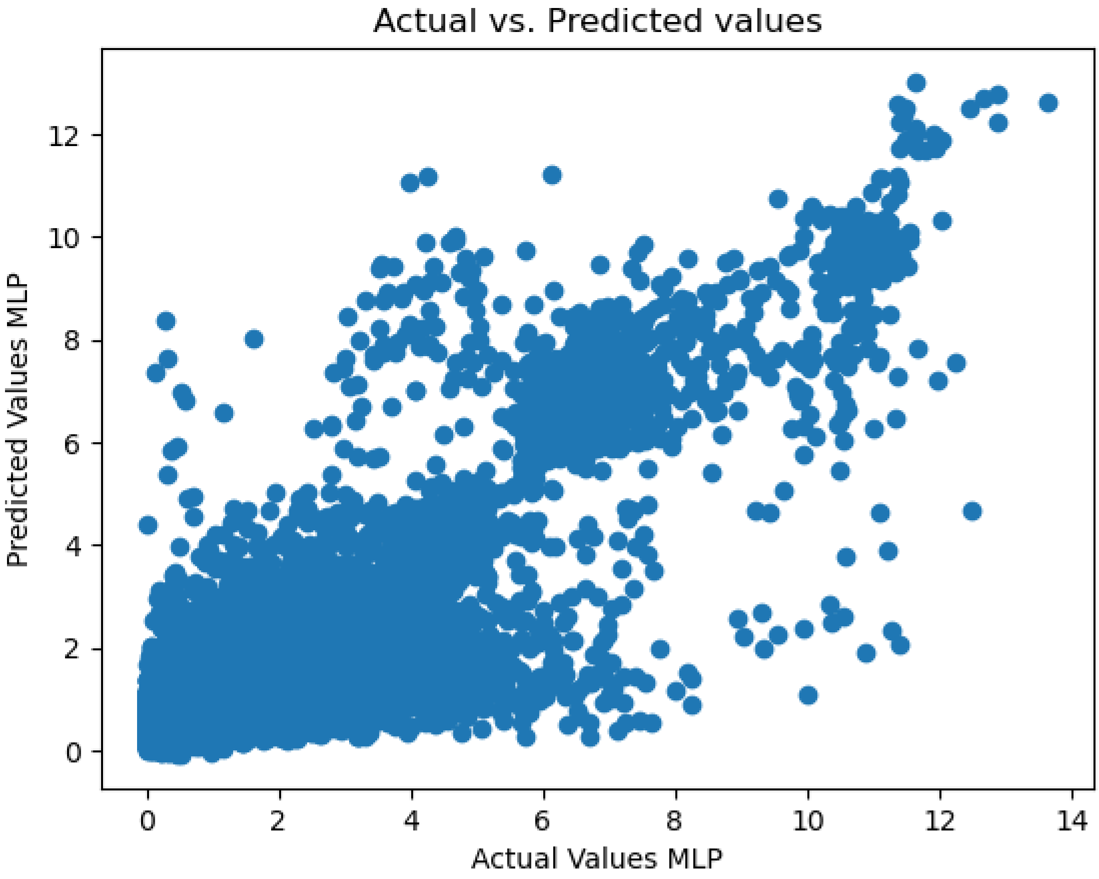
Out[11]:

```
array([373.249988])
```

In [12]:
```
# make predictions using the MLP regressor
Y_pred = MLP.predict(X_test)

#calculate Mean absolute error (MAE)
MAE = mean_absolute_error(Y_test,Y_pred)
print("Mean Absolute Error (MLP):",MAE)
```

```
Mean Absolute Error (MLP): 0.2845635740535989
```

In [13]:
```
# visualize the results using a scatter plot
ya = Y_test
yp = MLP.predict(X_test)

plt.scatter(ya,yp)
plt.xlabel("Actual Values MLP")
plt.ylabel("Predicted Values MLP")
plt.title("Actual vs. Predicted values")
plt.show()
```



In [14]:

```
#training of random forst model
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state=0)
regressor.fit(X_train, Y_train)
```

```
C:\Users\user\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

Out[14]:

```
▾                RandomForestRegressor

RandomForestRegressor(n_estimators=10, random_state=0)
```

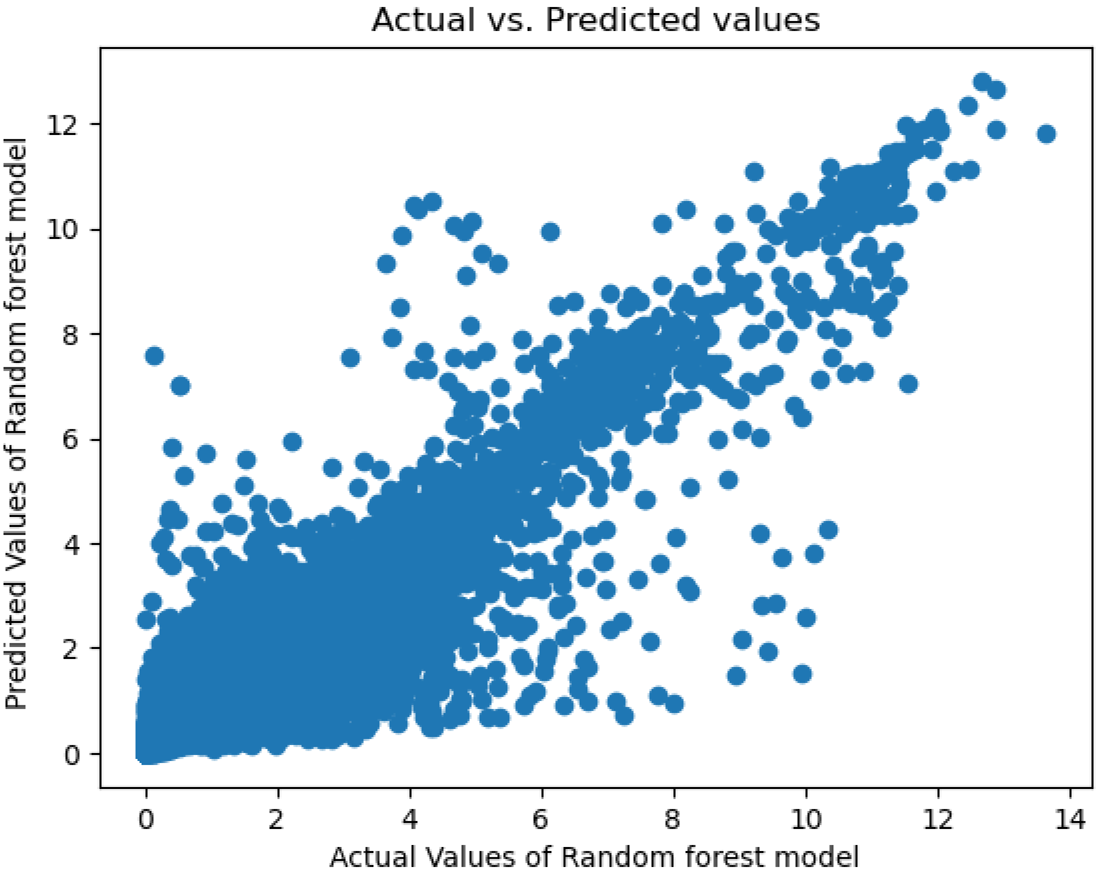
In [15]:

```
# make predictions using the Random forrest regressor
Y_pred_RF = regressor.predict(X_test)

#calculate Mean absolute error (MAE)
MAE = mean_absolute_error(Y_test,Y_pred_RF)
print("Mean Absolute Error (RF):",MAE)
```

```
Mean Absolute Error (RF): 0.1366152315021199
```

In [16]:

```
# visualize the results using a scatter plot
ya_RF = Y_test
yp_RF = regressor.predict(X_test)

plt.scatter(ya_RF,yp_RF)
plt.xlabel("Actual Values of Random forest model")
plt.ylabel("Predicted Values of Random forest model ")
plt.title("Actual vs. Predicted values")
plt.show()
```



In [ ]: