ADVANCED DATA SCIENCE

Assignment 2

Machine Learning with Energy Datasets

Prof. Sri Krishnamurthy

Team Members:

Sayali Borse

Rishi Rajani

Komal Ambekar

Content:

Summary:

As interest in IOT and sensors pick up steam, companies are trying to build algorithms and systems to understand consumer behavior to help them make better decisions. One such application is energy modeling. Though, most consumers are aware of their aggregate consumption of energy, few are aware of how and where energy is consumed. With increasing sensors in equipment, it is becoming easier to find out which equipment/instruments consume the most power. AdaptiveAlgo Systems Inc. works on solutions to build algorithms and platforms to address energy modeling challenges. The company is putting together a solution for energy modeling and is interested in understanding consumer energy usage and the attributes that contribute to appliance energy usage. The data scientists there came across a recent paper and dataset and are interested in building various machine learning models that could contribute to understanding energy usage by appliances and the attributes that contribute to aggregate energy usage. With the knowledge of energy consumed by various equipment, seasonality and attributes like temperature and humidity, a machine learning model could be used to predict aggregate energy use.

Research Paper Summary:

We were given three research papers for review and analysis. The study is related to appliances and energy consumption and prediction of energy consumption. Each document had to offer different features. The three papers were:

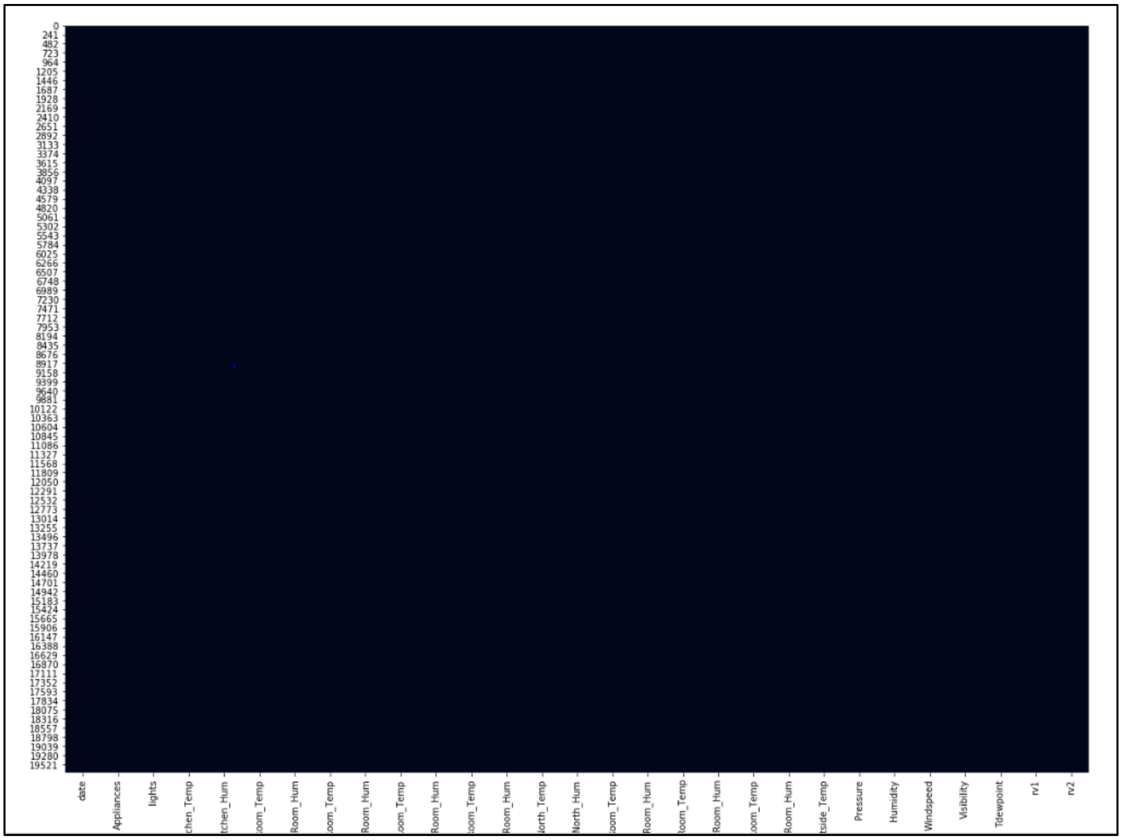
1. Data driven prediction models of energy use of appliances in a low-energy house.
2. A review of artificial intelligence-based building energy use prediction: Contrasting the capabilities of single and ensemble prediction models
3. Prediction of appliances energy use in smart homes

Exploratory Data Analysis:

It is an approach to analyze data sets to summarize their main characteristics, often with visual methods. EDA is for seeing what the data can tell us beyond the formal modeling. It is typically the first step of analysis.

To check all null values in the dataset:

```
In [3]: plt.subplots(figsize=(20,15))
        sns.heatmap(data.isnull(), cbar = False)
```

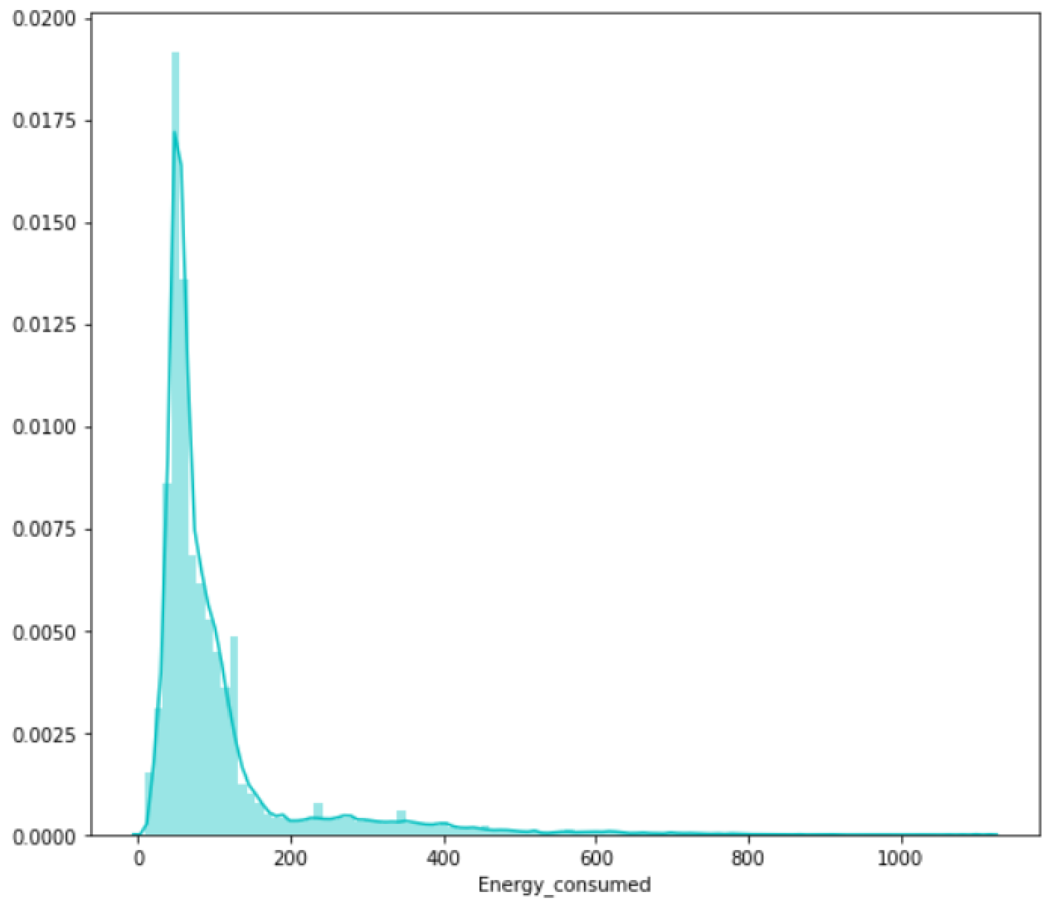


➔ Energy Consumed:

In [21]:
```python
print(data['Energy_consumed'].describe())
plt.figure(figsize=(9, 8))
sns.distplot(data['Energy_consumed'], color='c', bins=100)
```

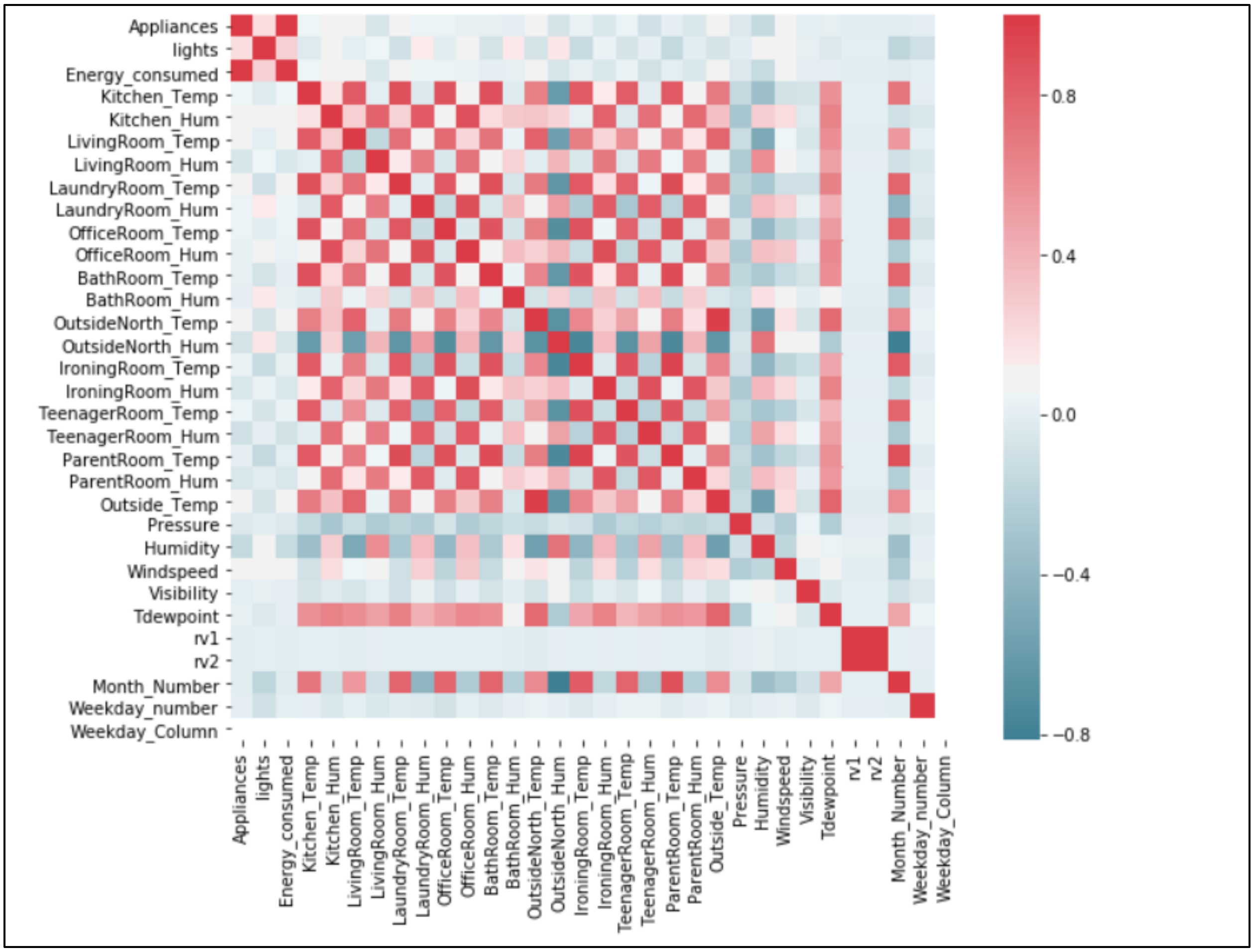
```
count    19735.000000
mean       101.496833
std        104.380829
min         10.000000
25%         50.000000
50%         60.000000
75%        100.000000
max       1110.000000
Name: Energy_consumed, dtype: float64
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x20dcdfd7438>



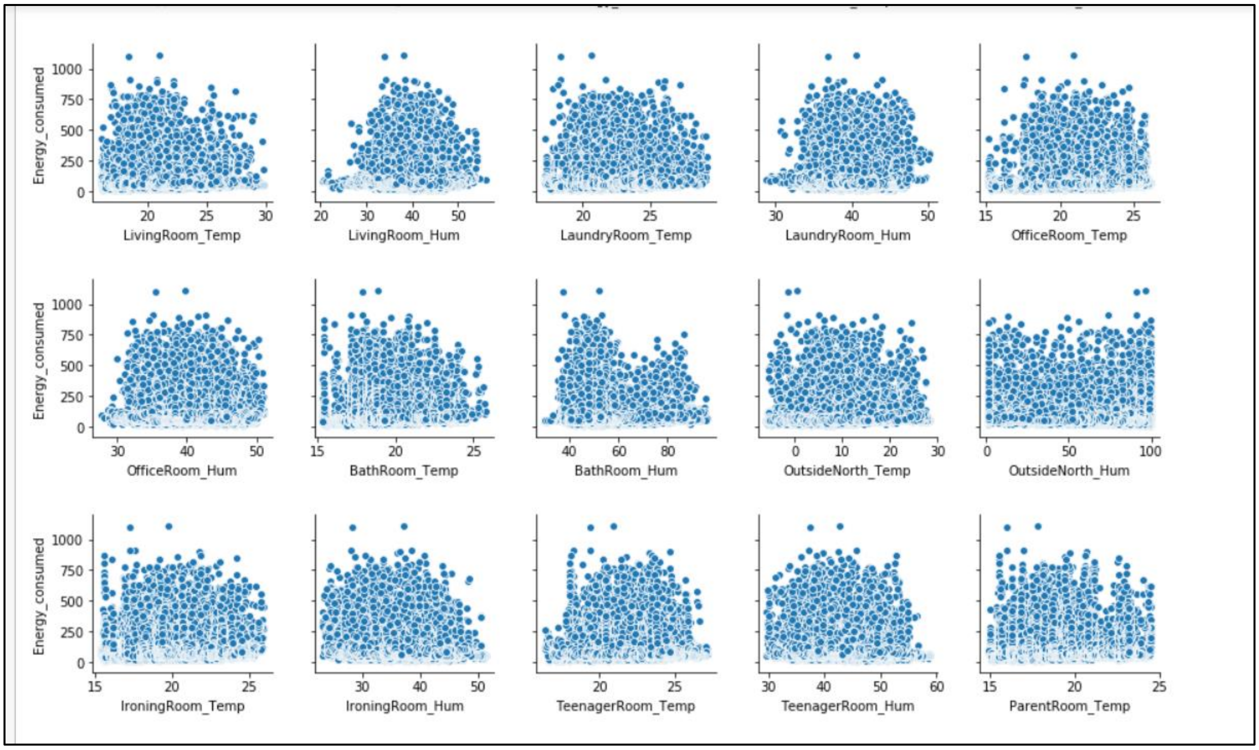
→ Correlation Matrix

In [16]:
```
f, ax = plt.subplots(figsize=(10, 8))
corr = data.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
plt.savefig("correlation.png")
```



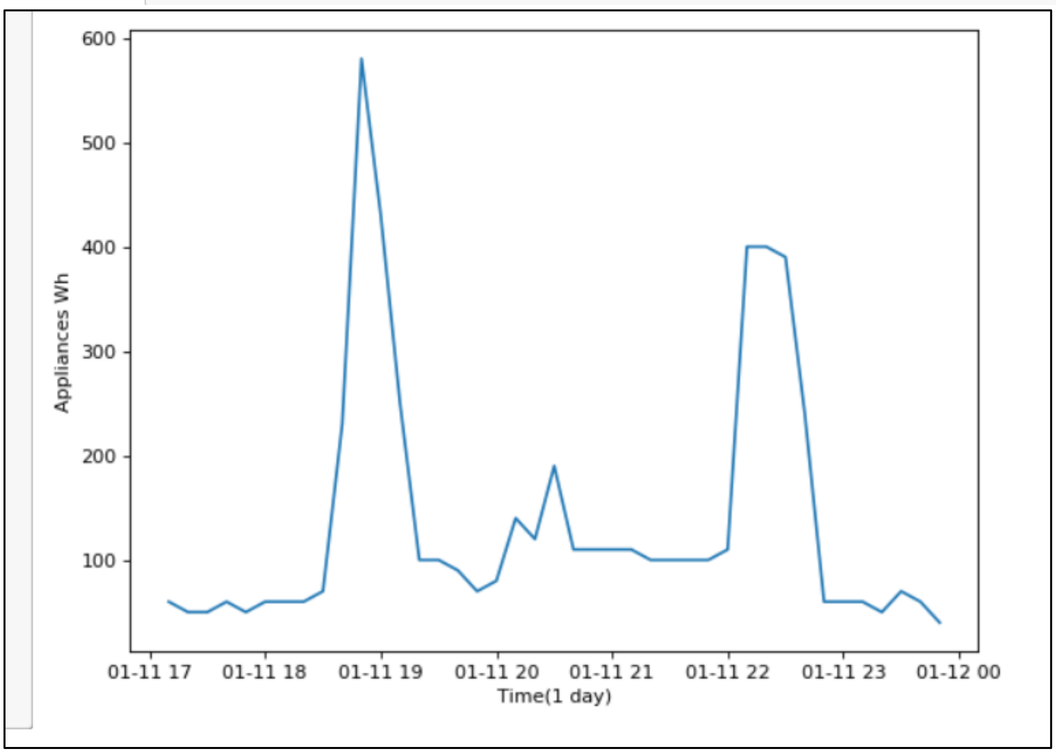
➔ Scatter plot wrt Energy Consumed

In [18]:
```
# Scatterplot w.r.t Energy_consumed
for i in range(0, len(df_num.columns), 5):
    sns.pairplot(data=df_num,
                 x_vars=df_num.columns[i:i+5],
                 y_vars=['Energy_consumed'])
```



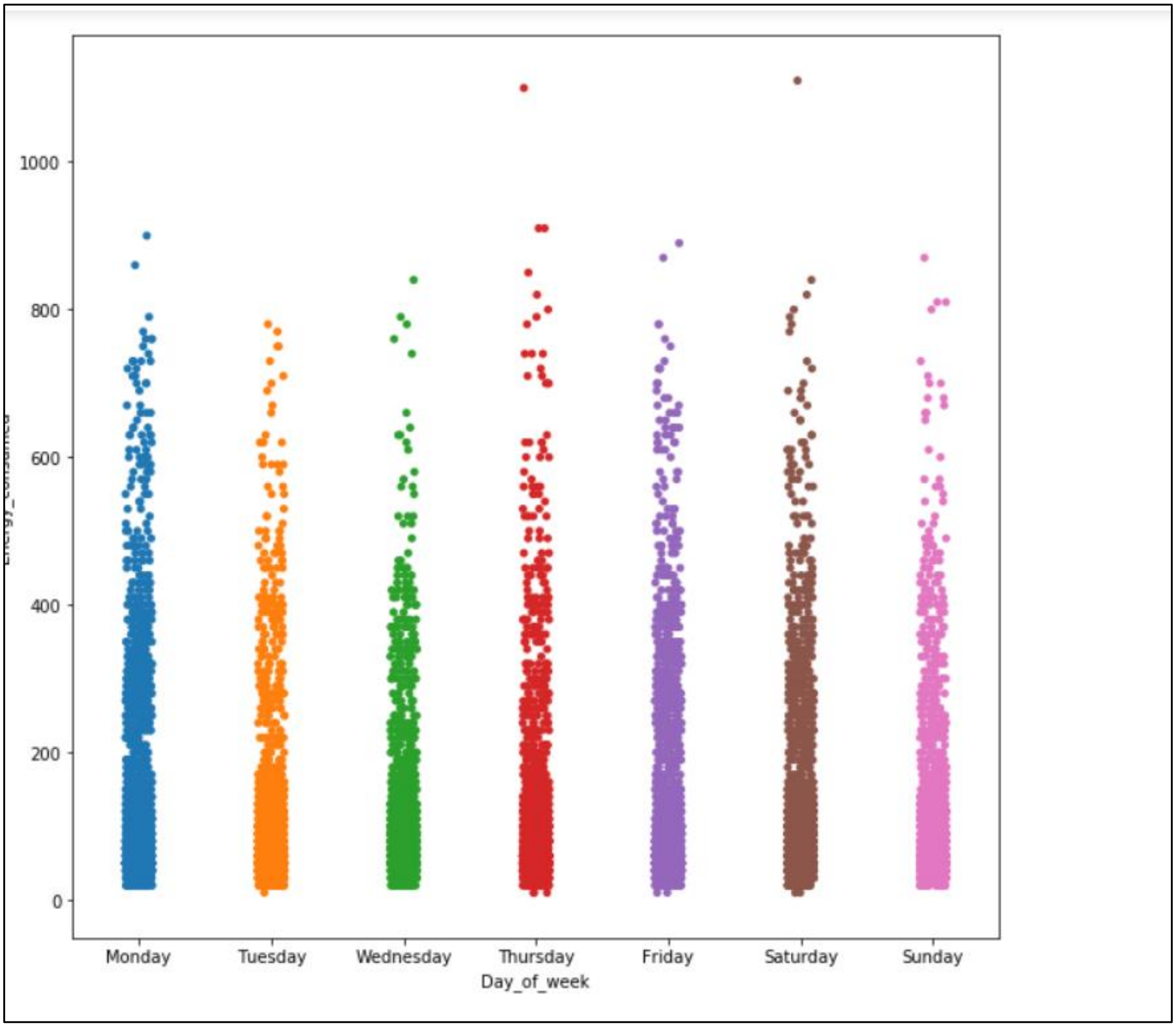
➔ Daily Energy Consumption

```
In [19]: fig=plt.figure(figsize=(8,6), dpi= 80, facecolor='w', edgecolor='k')
         ax=fig.add_subplot(111)
         ax.plot(data.date[1:42],data.Appliances[1:42])
         ax.set_xlabel('Time(1 day)')
         ax.set_ylabel('Appliances Wh')
         plt.savefig("appliance_daily.png")
```



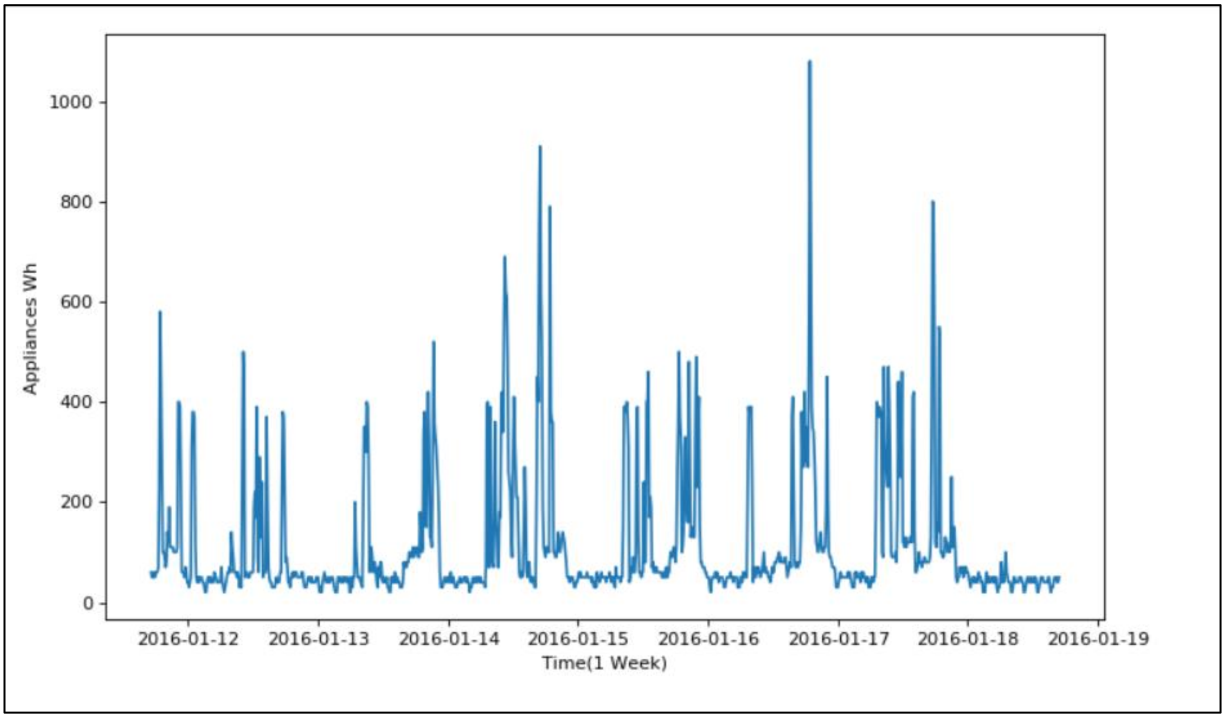
➔ Weekday wise energy consumption

In [21]:
```
# Strip plot to check which day there was more usage of energy
f, ax = plt.subplots(figsize=(10,10))
ax = sns.stripplot(x="Day_of_week", y="Energy_consumed" , data= data)
```



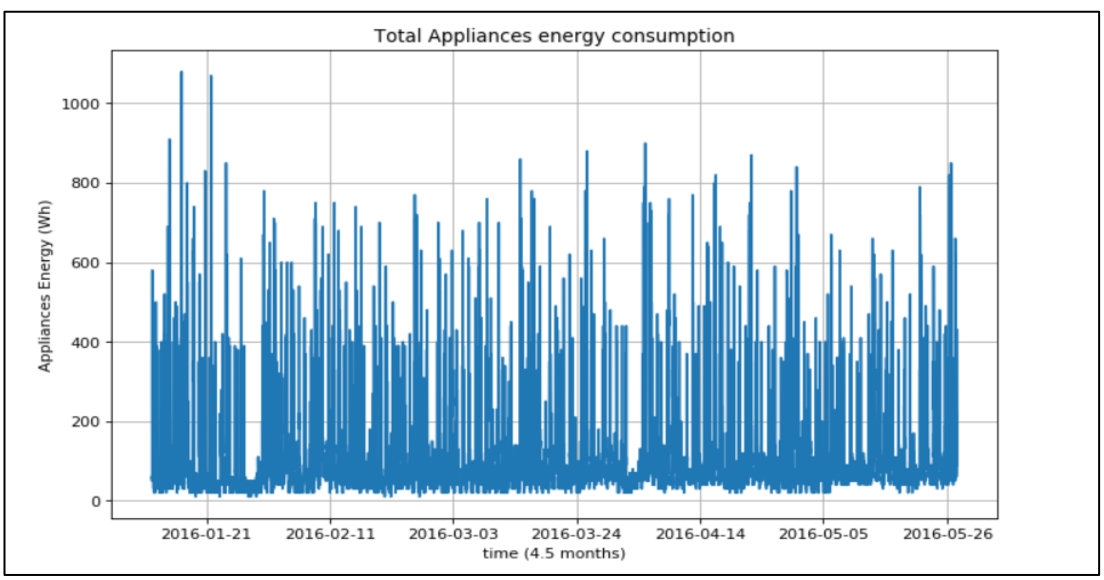
➔ Weekwise Energy Consumption

In [22]:
```
fig=plt.figure(figsize=(10,6), dpi= 80, facecolor='w', edgecolor='k')
ax=fig.add_subplot(111)
ax.plot(data.date[1:1008], data.Appliances[1:1008])
ax.set_xlabel('Time(1 Week)')
ax.set_ylabel('Appliances Wh')
plt.savefig("appliance_1week.png")
```



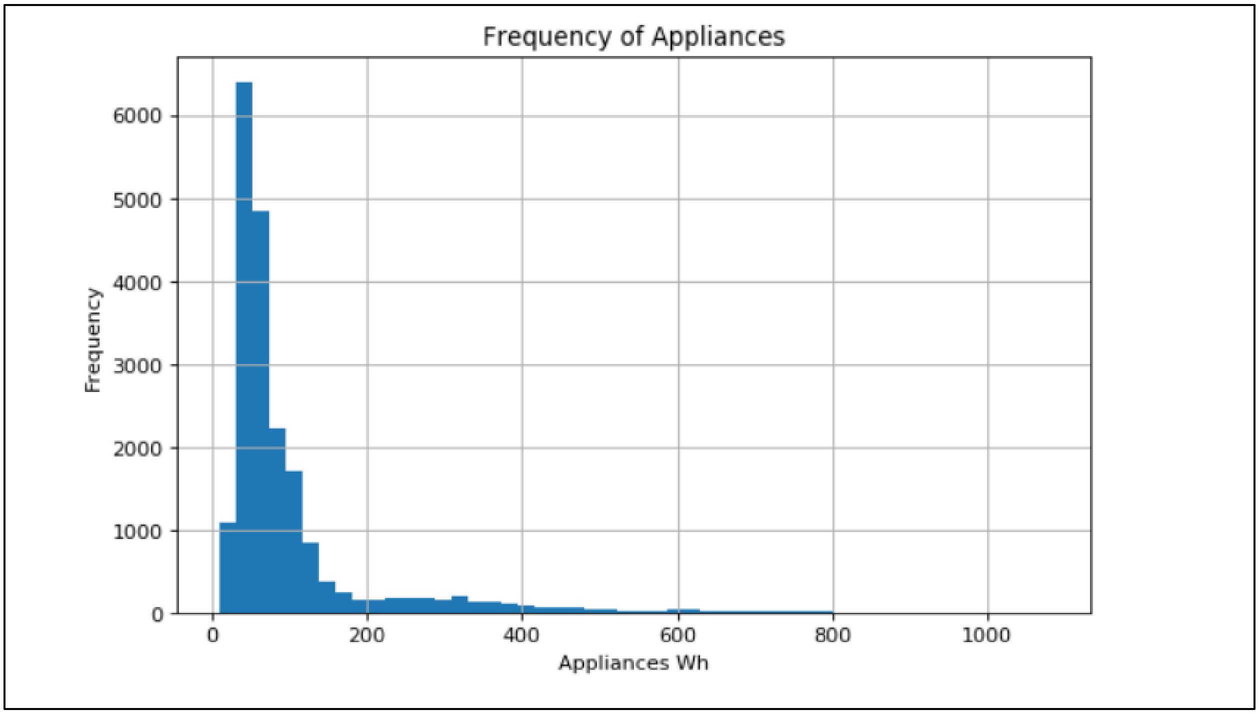
➔ Total Appliances energy consumption

In [23]:
```
%matplotlib inline
fig=plt.figure(figsize=(10,6), dpi= 80, facecolor='w', edgecolor='k')
ax=fig.add_subplot(111)
ax.plot(data['date'],data['Appliances'])
ax.set(xlabel='time (4.5 months)', ylabel='Appliances Energy (Wh)',
       title='Total Appliances energy consumption')
ax.grid()
fig.savefig("appliance_graph.png")
plt.show()
```



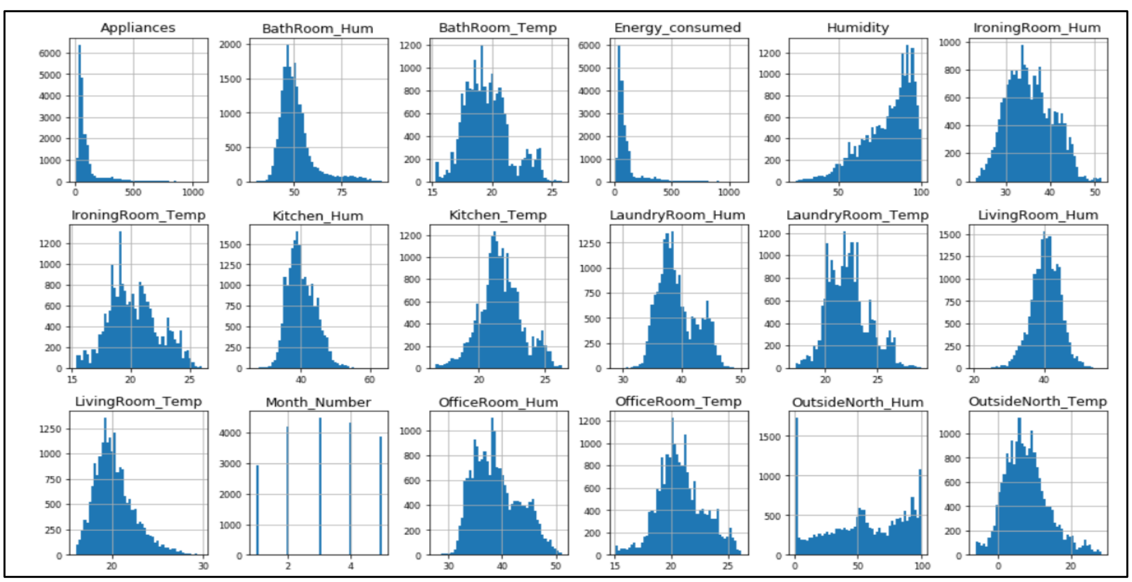
➔ Frequency of Appliances

In [24]:
```python
plt.figure(figsize=(8,5), dpi= 80, facecolor='w', edgecolor='k')
data['Appliances'].hist(bins=50)
plt.xlabel("Appliances Wh")
plt.title("Frequency of Appliances")
plt.ylabel("Frequency")
fig.savefig("frequency_application.png")
```



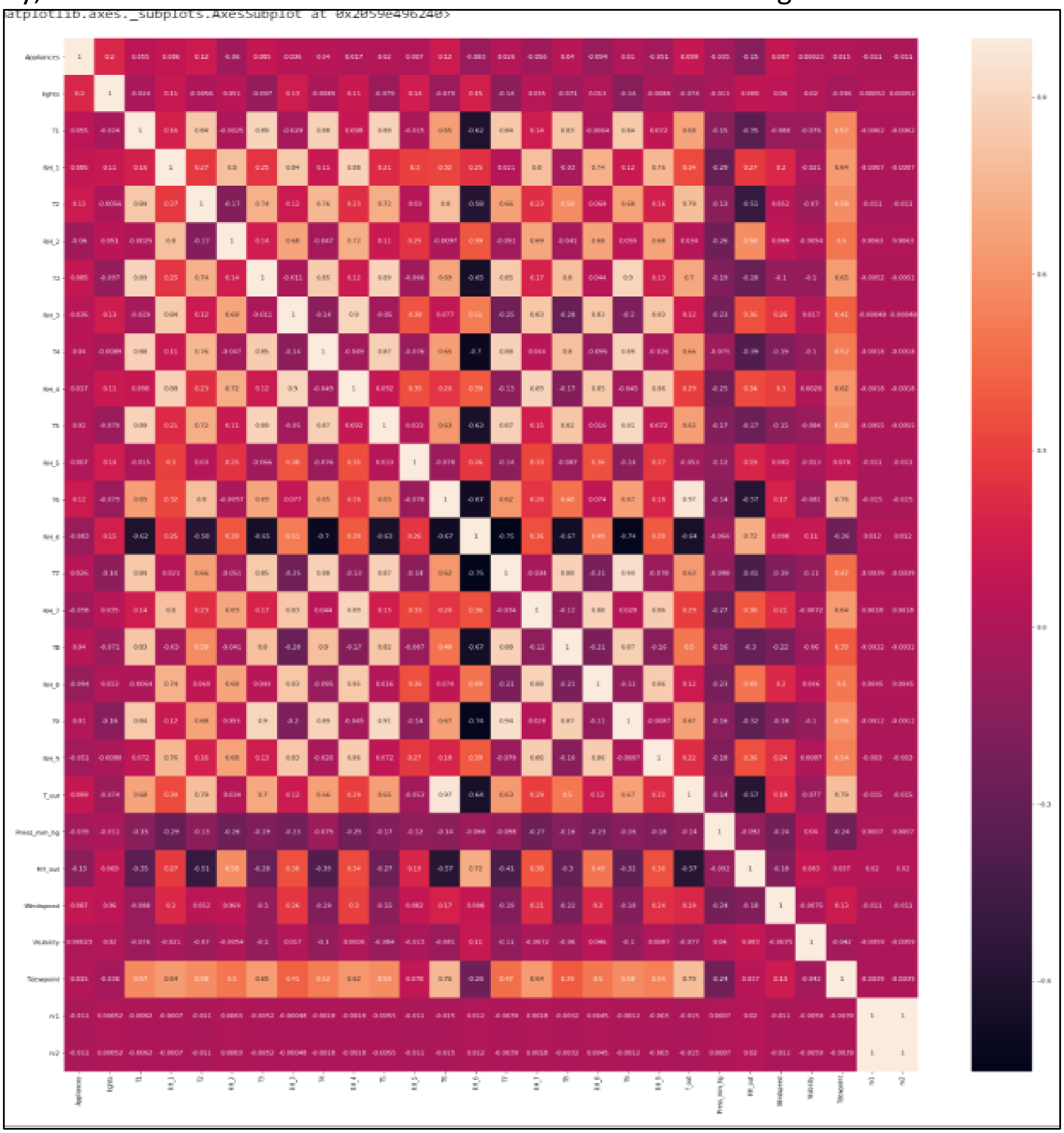
➔ Plotting Histogram of Numerical Features

In [25]:
```python
# Plotting histogram of numerical features
df_num.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8); # avoid having the matpl
```



Feature Engineering:

The dataset is numeric as we don't have any categorical features in it.
We also do not have any missing values
While studying the data we inferred that the total amount of energy consumed was derived by energy consumed by appliances and the light. The problem statement required us to predict the energy consumed basis the attributes that contribute to the energy consumption.
We also noticed the date time object is not helping us and hence decided to further divide it into attributes like time date, day of the month, week of the month, month number etc.
We also found the correlations between all variables and noticed that rv1 and rv2 were highly correlated and hence dropped rv2.
Lastly, we also renamed our columns for a better understanding.



Predictive Models:

In this section we have explored numerous machine learning models which can be used in our study to help getting best results in prediction.

We implemented the predictive models - Linear Regression, Random Forest and Neural Networks on our data sets and derived the accuracy score for all of them. We divided the fragment in 66% and 33% basis the sample training and testing data given in the problem statement.

We noticed that the accuracy score for random forest was the best among the lot where as neural networks was bad. Kindly find the below table indicating the matrix.

- Find the analysis of the models below:

Name of Model	Train / Test	Score	MAE	RMSE	R2	MAPE
Linear Regression	Train	0.1643	55.08	95.64	0.16	121.12
	Test	0.1411	54.05	95.35	0.14	120.06
Random Forest	Train	0.9420	11.96	25.18	0.94	103.20
	Test	0.5763	31.51	67.38	0.5763	108.53
Neural Network	Train	0.077	60.87	100.54	0.077	128.52
	Test	0.074	59.28	100.54	0.074	128.15

Later in the report you will also come across Extra Tree Regressor which we selected basis a technique we came across while performing feature selection.

Feature Selection

- Feature Selection Using Boruta:
  1. Feature selection deals with analyzing the feature and ranking them with different performance metrics and using them in the further model to get the best results.
  2. We have explored several feature selection techniques like:
     a) Boruta Package

```
            Tentative:      0
            Rejected:       5


            BorutaPy finished running.

            Iteration:      73 / 100
            Confirmed:      23
            Tentative:      0
            Rejected:       5

   Out[7]: array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
                   True,  True,  True,  True,  True,  True,  True,  True,  True,
                   True,  True,  True,  True, False,  True, False, False, False,
                  False])
```

Boruta on LR

```
In [ ]:

In [33]: regressor = LinearRegression()

In [34]: regressor.fit(x_train,y_train)
Out[34]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [36]: feature_selector = BorutaPy(regressor,n_estimators='auto', verbose=2, random_state=np.random)

In [37]: feature_selector.fit(x,y)
         feature_selector.support_
```

b) Forward and Backward Selection

```r
library(ISLR)
library(leaps)
library(tidyverse)
library(caret)

regfit.fwd=regsubsets(Energy_consumed~.,data=Revised_Data,nvmax=34,method = "forward")
B=summary(regfit.fwd)
names(B)
B
B$rss
B$adjr2
coef(regfit.fwd,34)
```

```
5:1    (Top Level) ÷
```

Console   Terminal ×

```
~/
[34] 178961467
> B$adjr2
 [1] 0.008530899 0.064087735 0.071460264 0.107707350 0.126127613 0.130815327 0.139459285 0.143274497 0.145882644
[10] 0.148989680 0.153854274 0.155543501 0.156781699 0.158416937 0.159558171 0.160724593 0.161863679 0.162729408
[19] 0.163855516 0.164795799 0.165267444 0.165581271 0.165966961 0.166163887 0.166266695 0.166298762 0.166350146
[28] 0.166374523 0.166395536 0.166379963 0.166342320 0.166302324 0.166260896 0.166219332
> coef(regfit.bwd,34)
         (Intercept)                date         Kitchen_Temp          Kitchen_Hum      LivingRoom_Temp
        8.914967e+05        -3.645524e-06        -4.562899e+00         1.557008e+01        -2.196521e+01
      LivingRoom_Hum     LaundryRoom_Temp      LaundryRoom_Hum      OfficeRoom_Temp       OfficeRoom_Hum
       -1.456406e+01         2.650072e+01         4.914107e+00         9.542611e+00         1.887518e+00
       BathRoom_Temp         BathRoom_Hum    OutsideNorth_Temp     OutsideNorth_Hum     IroningRoom_Temp
       -1.619373e-01         1.144701e-01         7.734786e+00         9.892616e-02        -4.612267e-01
      IroningRoom_Hum    TeenagerRoom_Temp     TeenagerRoom_Hum      ParentRoom_Temp       ParentRoom_Hum
       -1.780882e+00         9.968903e+00        -5.319341e+00        -1.696526e+01        -1.355740e+00
        Outside_Temp             Pressure             Humidity            Windspeed           Visibility
       -8.148430e+00         1.440746e-01         8.274840e-02         2.011159e+00         2.176638e-01
           Tdewpoint                  rv1     Day_of_weekMonday  Day_of_weekSaturday    Day_of_weekSunday
        2.131540e+00        -3.746970e-02        -3.542478e-01         4.165787e+00        -9.014661e+00
 Day_of_weekThursday   Day_of_weekTuesday Day_of_weekWednesday         Month_Number                 Time
       -1.695218e+01        -2.101708e+01        -1.496621e+01        -3.430104e+00         4.012106e-04
```

Source on Save Run

```r
library(ISLR)
library(leaps)
library(tidyverse)
library(caret)

regfit.bwd=regsubsets(Energy_consumed~.,data=Revised_Data,nvmax=34,method = "backward")
B=summary(regfit.bwd)
names(B)
B
B$rss
B$adjr2
coef(regfit.bwd,34)
```

12:20 (Top Level)

Console Terminal

~/

```
[34] 178961467
> B$adjr2
 [1] 0.008530899 0.064087735 0.071460264 0.107707350 0.126127613 0.130815327 0.139459285 0.143274497 0.145882644
[10] 0.148989680 0.153854274 0.155543501 0.156781699 0.158416937 0.159558171 0.160724593 0.161863679 0.162729408
[19] 0.163855516 0.164795799 0.165267444 0.165581271 0.165966961 0.166163887 0.166266695 0.166298762 0.166350146
[28] 0.166374523 0.166395536 0.166379963 0.166342320 0.166302324 0.166260896 0.166219332
> coef(regfit.bwd,34)
        (Intercept)                date        Kitchen_Temp         Kitchen_Hum     LivingRoom_Temp
       8.914967e+05       -3.645524e-06       -4.562899e+00        1.557008e+01       -2.196521e+01
     LivingRoom_Hum    LaundryRoom_Temp     LaundryRoom_Hum     OfficeRoom_Temp      OfficeRoom_Hum
      -1.456406e+01        2.650072e+01        4.914107e+00        9.542611e+00        1.887518e+00
      BathRoom_Temp        BathRoom_Hum   OutsideNorth_Temp    OutsideNorth_Hum    IroningRoom_Temp
      -1.619373e-01        1.144701e-01        7.734786e+00        9.892616e-02       -4.612267e-01
    IroningRoom_Hum   TeenagerRoom_Temp    TeenagerRoom_Hum     ParentRoom_Temp      ParentRoom_Hum
      -1.780882e+00        9.968903e+00       -5.319341e+00       -1.696526e+01       -1.355740e+00
       Outside_Temp            Pressure            Humidity           Windspeed          Visibility
      -8.148430e+00        1.440746e-01        8.274840e-02        2.011159e+00        2.176638e-01
          Tdewpoint                 rv1   Day_of_weekMonday Day_of_weekSaturday   Day_of_weekSunday
       2.131540e+00       -3.746970e-02       -3.542478e-01        4.165787e+00       -9.014661e+00
Day_of_weekThursday  Day_of_weekTuesday Day_of_weekWednesday       Month_Number                Time
      -1.695218e+01       -2.101708e+01       -1.496621e+01       -3.430104e+00        4.012106e-04
```

c) Tsfresh

```
In [3]:
df = pd.read_csv("C:/Users/Komal/Desktop/Revised_Data.csv")
x = df[['date','Energy_consumed']]
x = pd.Series(data=df['Energy_consumed'].values, index=df['date'])
df = pd.DataFrame(x)
df.reset_index(inplace=True)
df.columns = ["time", "value"]
df["kind"] = "a"
df["id"] = 1

df_shift, y = make_forecasting_frame(x, kind="price", max_timeshift=10, rolling_direction=1)
X = extract_features(df_shift, column_id="id", column_sort="time", column_value="value", impute_function=impute,
                     show_warnings=False)
```

```
Feature Extraction: 100%|████████████████████████████████████████| 20/20 [12:15<00:00, 20.58s/it]
WARNING:tsfresh.utilities.dataframe_functions:The columns ['value__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"interc
ept"'
 'value__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"rvalue"'
 'value__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"slope"'
 'value__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_"stderr"'
 'value__agg_linear_trend__f_agg_"max"__chunk_len_50__attr_"intercept"'
 'value__agg_linear_trend__f_agg_"max"__chunk_len_50__attr_"rvalue"'
 'value__agg_linear_trend__f_agg_"max"__chunk_len_50__attr_"slope"'
 'value__agg_linear_trend__f_agg_"max"__chunk_len_50__attr_"stderr"'
 'value__agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"intercept"'
 'value__agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"rvalue"'
 'value__agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"slope"'
 'value__agg_linear_trend__f_agg_"mean"__chunk_len_10__attr_"stderr"'
 'value__agg_linear_trend__f_agg_"mean"__chunk_len_50__attr_"intercept"'
 'value__agg_linear_trend__f_agg_"mean"__chunk_len_50__attr_"rvalue"'
 'value__agg_linear_trend__f_agg_"mean"__chunk_len_50__attr_"slope"'
 'value__agg_linear_trend__f_agg_"mean"__chunk_len_50__attr_"stderr"'
 'value__agg_linear_trend__f_agg_"min"__chunk_len_10__attr_"intercept"'
```

```
In [4]:
impute(X)
features_filtered = select_features(X, y)
```

```
WARNING:tsfresh.feature_selection.relevance:Infered classification as machine learning task
```

d) TPOT

```
In [ ]:

In [ ]:

In [11]: tpot = TPOTRegressor(generations=5, population_size=50, verbosity=2)
tpot.fit(x_train, y_train)
print(tpot.score(x_test, y_test))
```

```
C:\Users\HP\Anaconda\lib\importlib\_bootstrap.py:205: ImportWarning: can't resolve package from __spec__ or __package__, falli
ng back on __name__ and __path__
  return f(*args, **kwds)
```

```
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.
Generation 1 - Current best internal CV score: -5866.56965676732
Generation 2 - Current best internal CV score: -5866.56965676732
Generation 3 - Current best internal CV score: -5733.67471080355
Generation 4 - Current best internal CV score: -5318.705441811093
Generation 5 - Current best internal CV score: -5095.624828524819

Best pipeline: ExtraTreesRegressor(input_matrix, bootstrap=False, max_features=0.8, min_samples_leaf=2, min_samples_split=4, n
_estimators=100)
-3980.333783408918
```

3. Considering the results obtained in this 5 techniques some of them were not suitable for the data-set and hence we chose the best one to refine our model and get best prediction accuracy through it.

Model Validation & Selection

1) We have used different model validation techniques to choose the best model for our dataset to predict values:
   a) Cross Validation
   b) Grid Search
   c) Bias Variance Trade-off
   d) Regularization
2) After training the models, we have calculated RMSE and R2. The best models are the ones that provide the lower RMSE and highest R2 values.

Final Pipeline

We have created a pipeline to automate the entire model from data ingestion to final model prediction.