CS 514 - Applied Artificial Intelligence
Project -5 Machine Learning
Great Energy Predictor

Username: Snow White

Problem Specification: To build an accurate machine learning model to predict the Energy Consumption based on the historic usage rates and weather conditions which could support better market incentives and enable lower cost financing.

Abstract :

Assessing the value of energy efficiency improvements can be challenging as there's no way to truly know how much energy a building would have used without the improvements. The best we can do is to build counterfactual models. Once a building is overhauled the new (lower) energy consumption is compared against modeled values for the original building to calculate the savings from the retrofit. More accurate models could support better market incentives and enable lower cost financing.

As given in the Kaggle competition, the goal of the project is to build the counterfactual models across four energy types based on historic usage rates and observed weather.

Data sets :

The dataset includes three years of hourly meter readings from over one thousand buildings at several different sites around the world.

train.csv - training set
building_meta.csv - building data
weather_train.csv - weather training set
test.csv - test set
weather_test.csv - weather test set

Libraries and Techniques used:

The project is built in python on jupyter notebook and it uses Pandas, Numpy, Scikit learn, Math, Matplotlib, Seaborn, Category_encoders.

Data Collection :

The training data is collected from train.csv, building_meta.csv and weather_train.csv. The test data is collected from test.csv, building_meta.csv and weather_test.csv.

Using Pandas, the data is read from the csv files into pandas dataframes.

Data Preparation and Exploratory Data Analysis:

In this step, the data is cleaned, exploratory data analysis is performed and data is prepared for building the models.

Merge the datasets to form the training set:

To form the training data set, the building_meta.csv and train.csv are merged on 'building_id' and then this merged set is merged with weather_train.csv on 'site_id' and 'timestamp'.

The final merged dataset 'data_merged' is considered as the training set used for building the model.

Add new features:

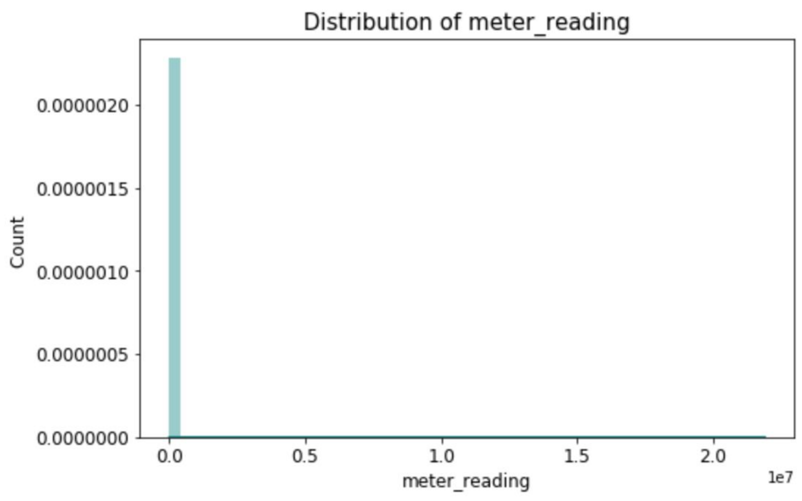
The 'timestamp' feature has characters, so it is split into four features 'hour', 'day', 'weekday' and 'month'. The original 'timestamp' feature is dropped.

Convert numerical values of categorical feature to categorical values:

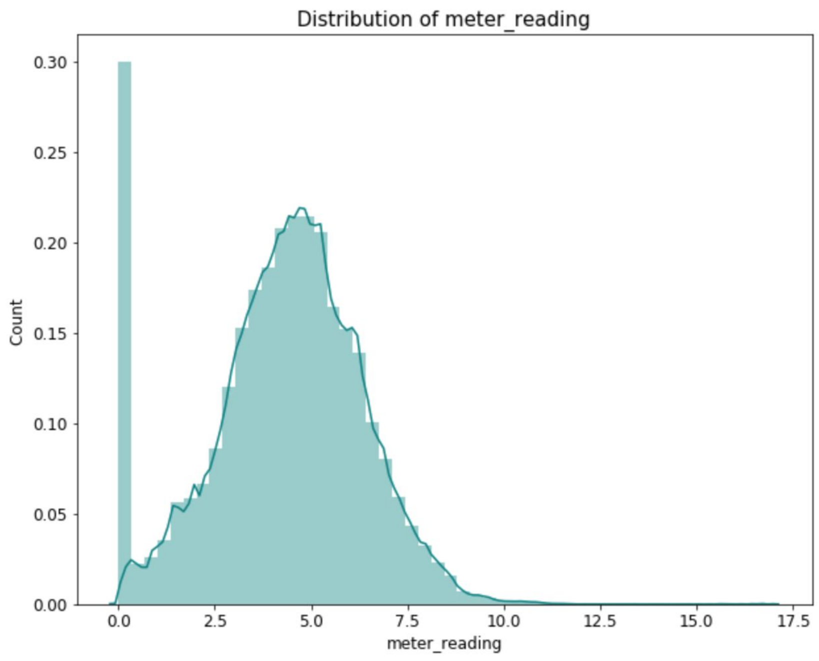
As 'meter' is a categorical feature which corresponds to the meter type, the numerical values of 'meter' are replaced with the categorical values using replace {0:"Electricity",1:"ChilledWater",2:"Steam",3:"HotWater"}

Data Analysis:

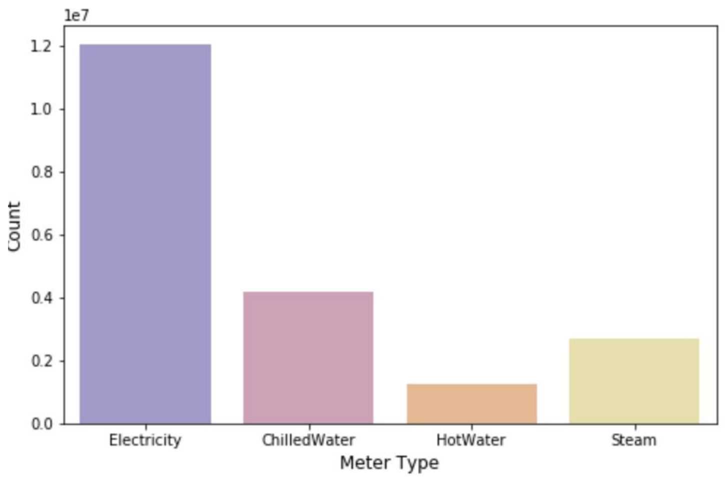
The target variable is explored to see the distribution of the values.



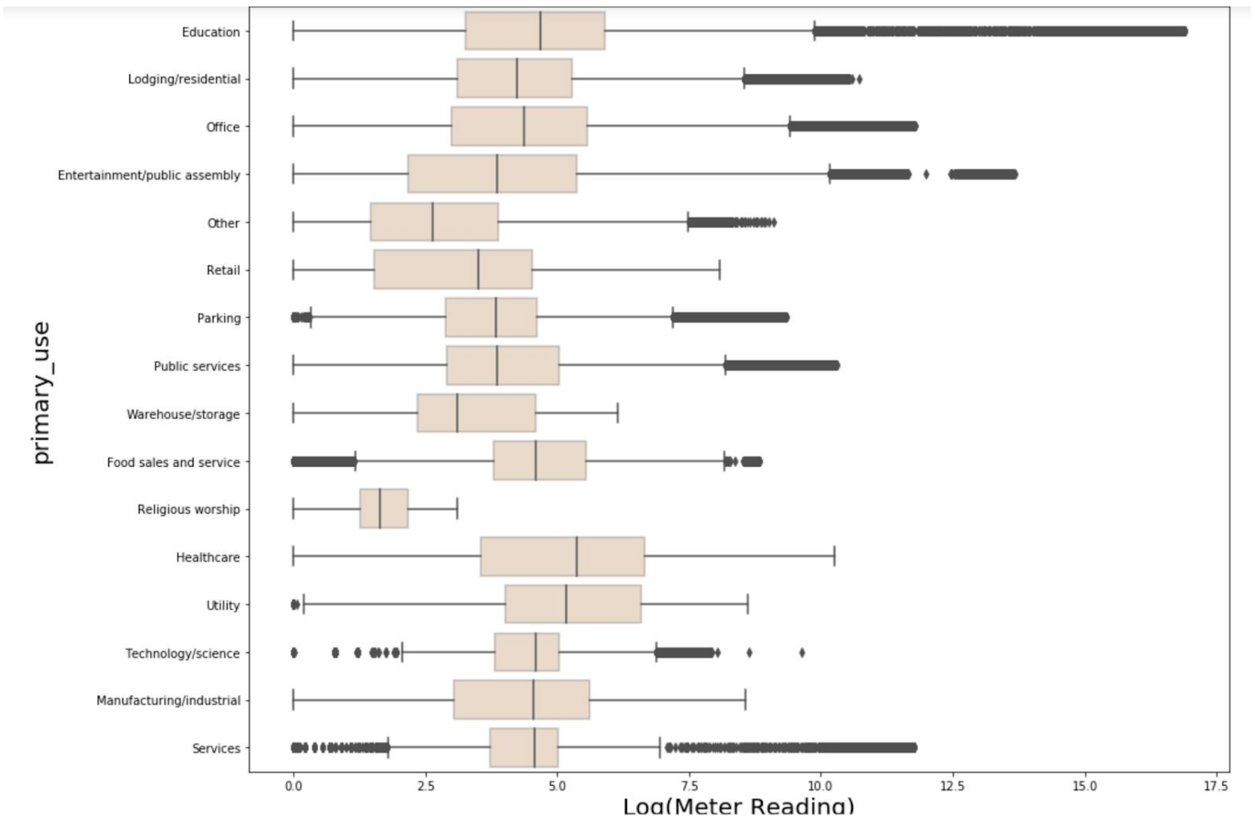
Since the target variable is positive skewed, log transformation is applied on the meter_reading which changes the distribution as below.



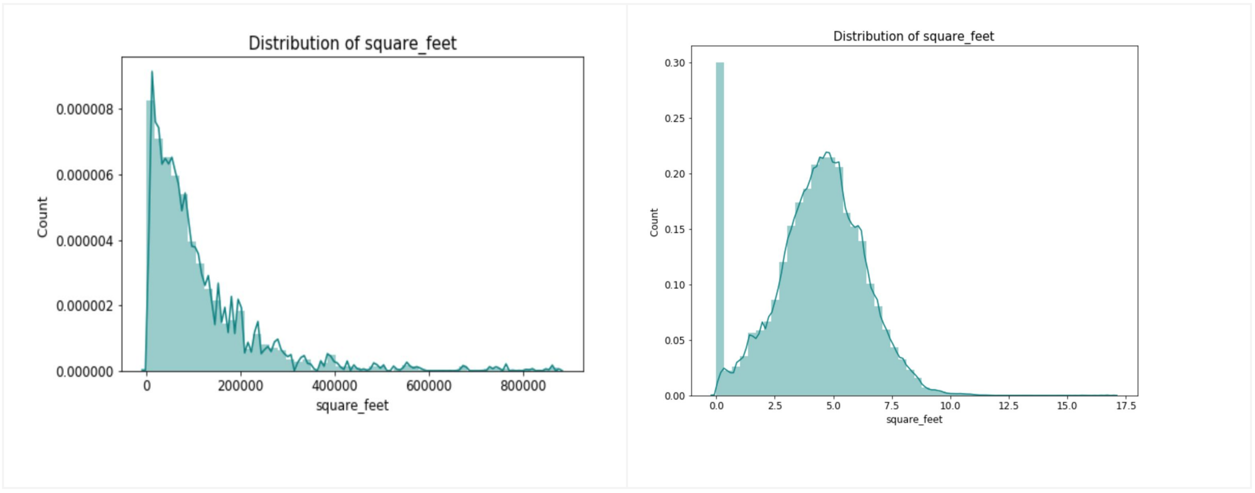
When we check the proportion of the meter types in the training set, we can see that Electricity is the most frequent meter type that is measured.



The readings with respect to primary_use are as below, which shows that Healthcare and Utility have higher readings compared to others.



As the distribution of square_feet is also positive skewed, log transformation is applied on the square_feet.



The other numerical variables seem to be normally distributed, so no transformation is performed on them.

Convert categorical attributes to numerical attributes:

The ‘primary_use’ is a categorical feature, which is converted to numerical feature using map() which maps {'Education':1, 'Lodging/residential':2, 'Entertainment/public assembly':3, 'Public services':4, 'Office':5,' Technology/science':6, 'Utility':7, 'Parking':8, 'Other':9, 'Healthcare':10, 'Manufacturing/industrial':11}.

Get Dummies:

The categorical feature is dropped and is replaced by four dummy features one each for Electricity, ChilledWater, HotWater, Steam. The dummy features are created using pandas.get_dummies(feature).

Categorical Encoders:

Categorical Encoders are used to encode the categorical features.
The categorical features in the given dataset are 'ChilledWater', 'Electricity', 'HotWater', 'Steam', ’site_id’, ‘building_id’, ‘primary_use’, ‘hour’, ‘weekday’, ‘wind_direction’.

Handle Missing Values:

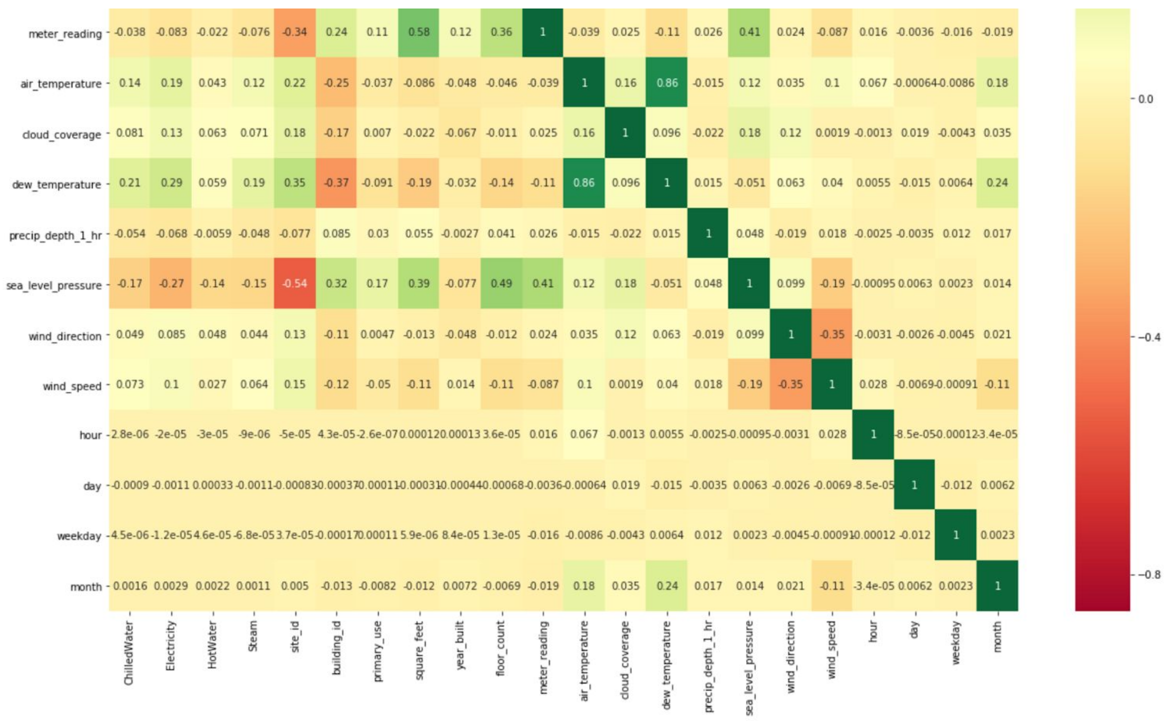
To find the missing values (NAN values) in each column of the dataframe, dataframe.isnull().sum() is used. The missing values are filled with zero.

Drop Observations:

The observations with zero values for square_feet, year_built, floor_count and primary_use are dropped.

Build Correlation Matrix:

The correlation matrix is built using dataframe.corr(). The matrix is displayed using heatmap. The correlation matrix displays the pair wise correlation coefficients between the variables. The features with high correlation coefficients with ‘meter_reading’ can be determined as important variables since our target variable is meter_reading and we want to build a model to predict the ‘meter_reading’.



Data Modelling using Regression models:

Partition the data:

Once the data is prepared, the data is partitioned into training and validation set. The training set is used to train the models and validation set is used to evaluate the models.

Build model:

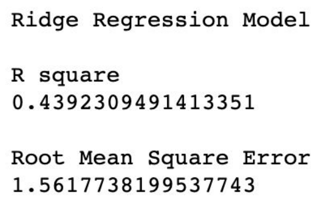
After the partitioning of data, Five different regression models are built and evaluated on the training data set.

The models are evaluated using Root Mean Squared Logarithmic Error (RMSE) and R Square (Coefficient Of Determination).

1. Linear Regression :

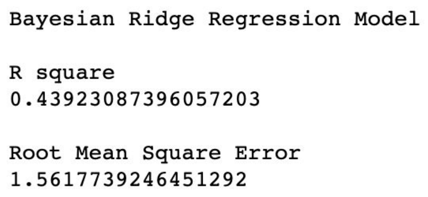
```
Linear Regression Model

R square
0.43923074997777584

Root Mean Square Error
1.561774097294691
```

2. Ridge Regression:



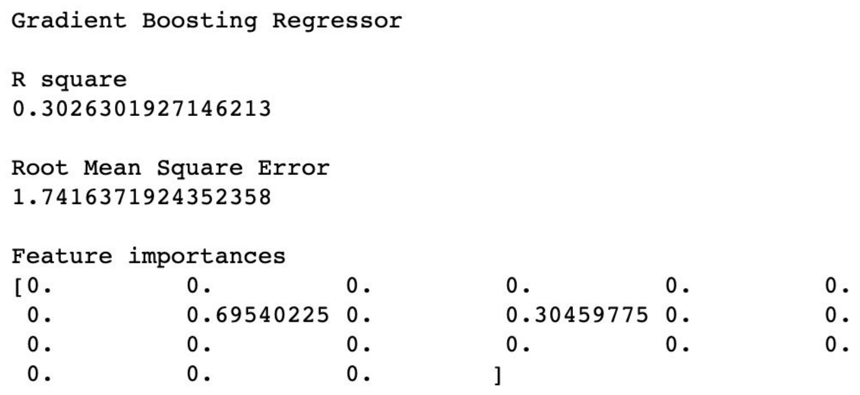
```
Ridge Regression Model

R square
0.4392309491413351

Root Mean Square Error
1.5617738199537743
```

3. Bayesian Ridge Regression:



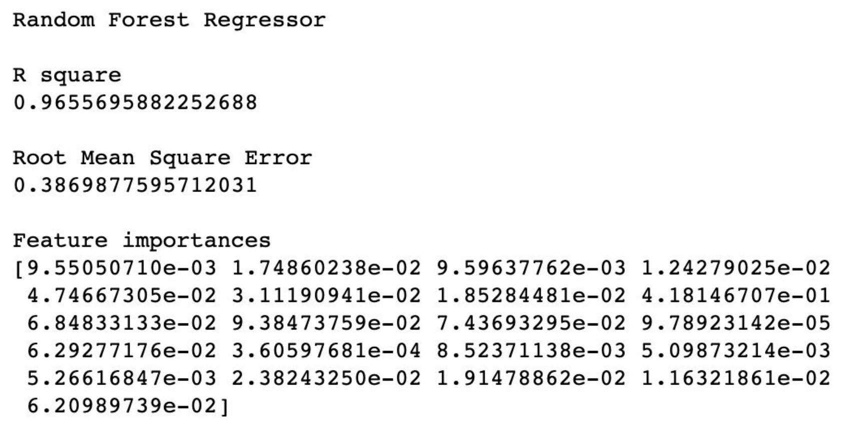
```
Bayesian Ridge Regression Model

R square
0.43923087396057203

Root Mean Square Error
1.5617739246451292
```

4. Gradient Boost Regressor:



```
Gradient Boosting Regressor

R square
0.3026301927146213

Root Mean Square Error
1.7416371924352358

Feature importances
[0.         0.         0.         0.         0.         0.
 0.         0.69540225 0.         0.30459775 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
```

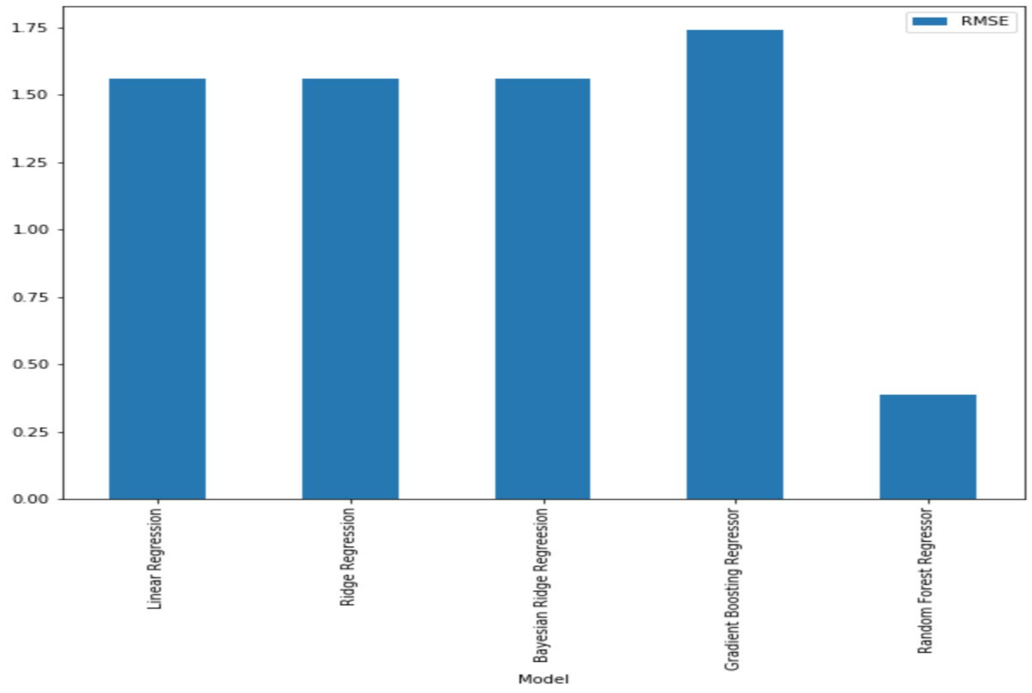
5. Random Forest Regressor:

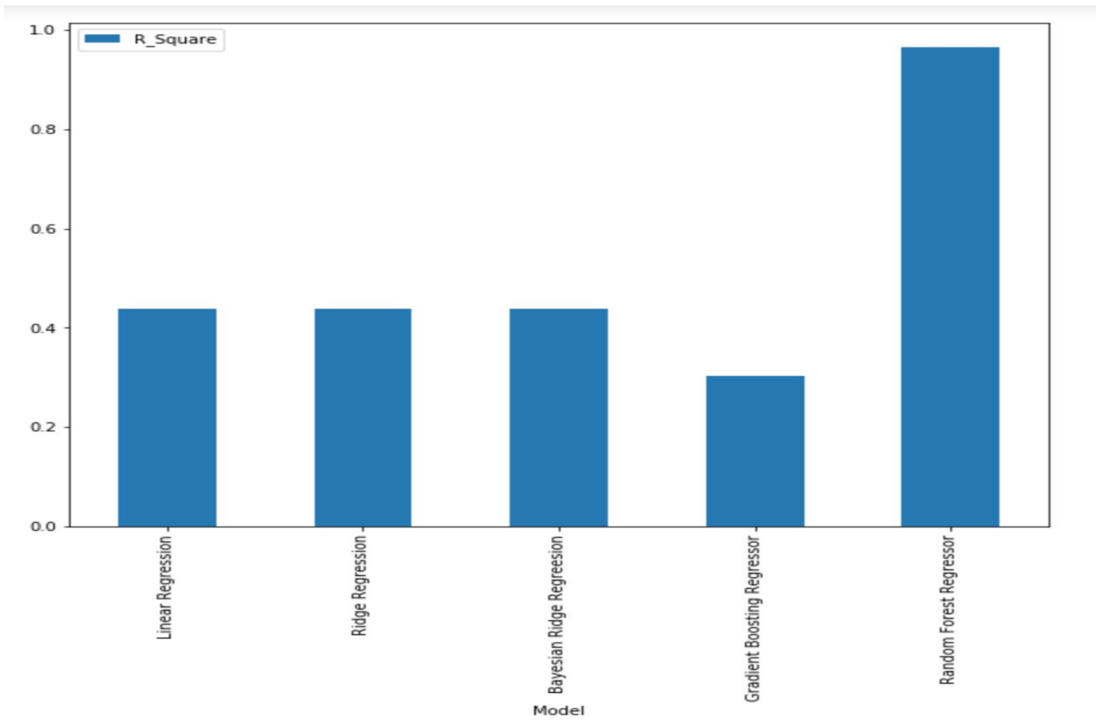


```
Random Forest Regressor

R square
0.9655695882252688

Root Mean Square Error
0.3869877595712031

Feature importances
[9.55050710e-03 1.74860238e-02 9.59637762e-03 1.24279025e-02
 4.74667305e-02 3.11190941e-02 1.85284481e-02 4.18146707e-01
 6.84833133e-02 9.38473759e-02 7.43693295e-02 9.78923142e-05
 6.29277176e-02 3.60597681e-04 8.52371138e-03 5.09873214e-03
 5.26616847e-03 2.38243250e-02 1.91478862e-02 1.16321861e-02
 6.20989739e-02]
```

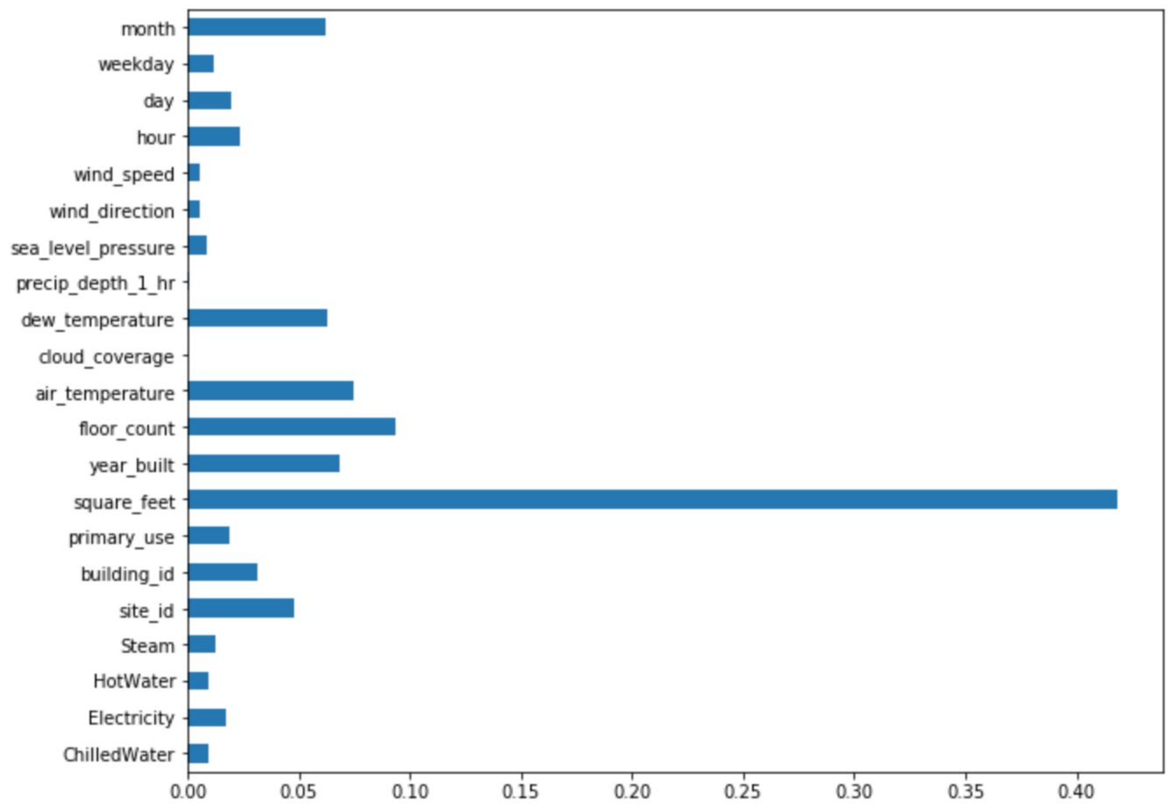
Best model:

Out of the models tried, Random Forest Regressor gave the best performance on the validation set with a Root Mean Squared Logarithmic Error of 0.38 and RSquare of 0.96.
And, the Random Forest Regressor model is used for predicting the test set.





Feature Importance given by Random Forest Regressor:



Prediction on Test Set:

Data Preparation:

All the data preparation steps are performed on the test set as well.

1. Building_meta.csv, test.csv and weather_test.csv are merged.
2. New features hour', 'day', 'weekday' and 'month' are added and 'timestamp' is dropped.
3. Dummy features are created for 'meter' and 'meter' feature is dropped.
4. Categorical values are encoded.
5. Missing values are replaced with averages.
6. Rowids column is copied to a dataframe which is later used to create the submission file.

Predict using best model

Using the best model, which is Random Forest Regressor, the target variable that is meter_reading is predicted.

Output

The output dataframe is generated with rowids and meter_reading values. The generated output is copied onto “Project5_submission_results.csv” file.

Best Results:

As mentioned above, the best model is Random Forest Regressor with Root Mean Squared Logarithmic Error of 0.38 and RSquare of 0.98.

Instructions to run:

1. Download Project5_SnowWhite.zip file.
2. Extract the contents of the file. It will have **Project5_GreatEnergyPredictor.ipynb, Project5_GreatEnergyPredictor_code.pdf** and **Project5_GreatEnergyPredictor_report.pdf.**
3. Download the csv files : **train.csv, building_meta.csv, weather_train.csv, test.csv, weather_test.csv, sample_submission.csv** from Kaggle. Place these files in the same folder as Project5_GreatEnergyPredictor.ipynb.
4. Open Jupyter Notebook.
5. Open Project5_GreatEnergyPredictor.ipynb file in Jupyter Notebook. (make sure train.csv, building_meta.csv, weather_train.csv, test.csv, weather_test.csv, sample_submission.csv are in the same folder as Project5_GreatEnergyPredictor.ipynb for running the contents in Project5_GreatEnergyPredictor.ipynb)
6. Install all required libraries and run the steps in the Jupyter Notebook to see the outputs.
7. After running all the steps, check “**Project5_submission_results.csv**” for the predicted meter readings on the test set.

Note : **Project5_GreatEnergyPredictor_code.pdf** is the jupyter notebook which is saved as pdf with the code and results generated for each step.