Assignment-2_Group-9_Report

Group 9 Members:

Jialu He, Manqi Wang, Xiu Zhou, Yaping Zhang

Introduction

The Provisional State of the Global Climate 2022 from the United Nations, report that the consequences of global warming are now inevitable. However, the trend can be managed the management and control of current greenhouse gas emissions.

Greenhouse gases are generated when fossil fuels are burnt to produce energy. Studies have shown that energy consumption are higher than usual when the weather is warmer. In order to reduce greenhouse gasses, the energy companies must be supported to rationalize future energy usage and pricing. It is particularly important to estimate future energy prices and demand based on the weather forecast to determine the impact it will have on consumers.

In this project, two forecasting models are built;

- The first model estimates future electricity consumption in Melbourne using historical weather data between January and August 2021.
- The second model predicts the maximum daily price category based on energy price and demand figures for the state of Victoria for each half hour period between January and August 2021.

Consumers are more likely to use more air conditioners in warmer weather and more heaters in colder periods. This information therefore correlates that the energy consumption is directly proportional to the condition of the weather.

This project not only provides a reliable tool to help energy companies forecast for future energy usage, as well as help businesses schedule energy-intensive activities, but also provide valuable information to the government where they would be able to put into effect impactful and efficient energy and environmental policies.

Data Wrangling and Aggregation

Energy Price and Demand Analysis and Data Preparation

As a part of this report, 2 data sets are provided;

- `weather_data;` and
- `price_demand_data;`

`price_demand_data` contains features need to be predicted and `weather_data` contains features used to predict. So the first step is to review the contents and format of the data provided.

Price and Energy Demand Dataset

For `price_demand_data`, we used `pd.read_csv()` to import data into Jupiter, and have a first look at the data with `head().`

	REGION	SETTLEMENTDATE	TOTALDEMAND	PRICECATEGORY
0	VIC1	1/01/2021 0:30	4179.21	LOW
1	VIC1	1/01/2021 1:00	4047.76	LOW
2	VIC1	1/01/2021 1:30	3934.70	LOW
3	VIC1	1/01/2021 2:00	3766.45	LOW
4	VIC1	1/01/2021 2:30	3590.37	LOW

Once the information is readable, we used .`info()` to obtain the metadata about this dataset (e.g. data type, length). All of the features have the same length 11664.

```
# Check the data information
price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11664 entries, 0 to 11663
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   REGION          11664 non-null  object 
 1   SETTLEMENTDATE  11664 non-null  object 
 2   TOTALDEMAND     11664 non-null  float64
 3   PRICECATEGORY   11664 non-null  object 
dtypes: float64(1), object(3)
memory usage: 364.6+ KB
```

As the data could contain missing null values;`.isna().sum()` is used to detect them. We can see there is no missing values from this data set.

```
## Detect missing values
price.isna().sum()
```

```
REGION            0
SETTLEMENTDATE    0
TOTALDEMAND       0
PRICECATEGORY     0
dtype: int64
```

We then divided date and time from [SETTLEMENTDATE] into different column.

```
# Divide Date & Time into different column
price['Date'] = pd.to_datetime(price['SETTLEMENTDATE'], format='%d/%m/%Y %H:%M').dt.date
price['Time'] = pd.to_datetime(price['SETTLEMENTDATE'], format='%d/%m/%Y %H:%M').dt.time
display(price.head())
```

	REGION	SETTLEMENTDATE	TOTALDEMAND	PRICECATEGORY	Date	Time
0	VIC1	1/01/2021 0:30	4179.21	LOW	2021-01-01	00:30:00
1	VIC1	1/01/2021 1:00	4047.76	LOW	2021-01-01	01:00:00
2	VIC1	1/01/2021 1:30	3934.70	LOW	2021-01-01	01:30:00
3	VIC1	1/01/2021 2:00	3766.45	LOW	2021-01-01	02:00:00
4	VIC1	1/01/2021 2:30	3590.37	LOW	2021-01-01	02:30:00

Model 1 is to predict daily total usage, therefore the daily total demand is calculated based on summing every day usage using `.groupby('Date')['TOTALDEMAND'].sum()`.

```
# Calculate daily total demand
sum_price = price.groupby('Date')['TOTALDEMAND'].sum()
sum_price = sum_price.reset_index()
display(sum_price.head())
```

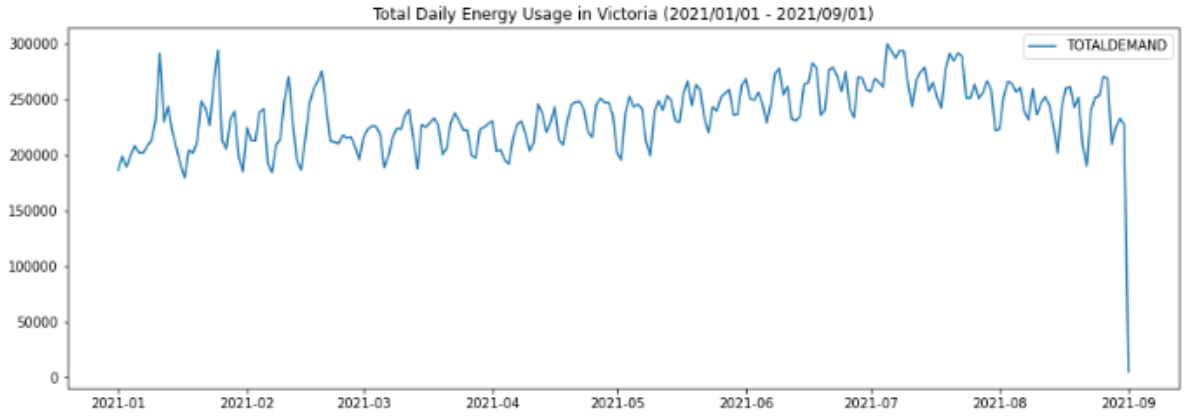
	Date	TOTALDEMAND
0	2021-01-01	185853.37
1	2021-01-02	197990.13
2	2021-01-03	188742.96
3	2021-01-04	199281.07
4	2021-01-05	207680.91

Once the daily total was obtained, `set_index()` is used to set extracted date as an index to allow for easier indexing for steps further in this report.

```
# Set data as index
sum_price.set_index('Date',inplace = True)
display(sum_price.head())
```

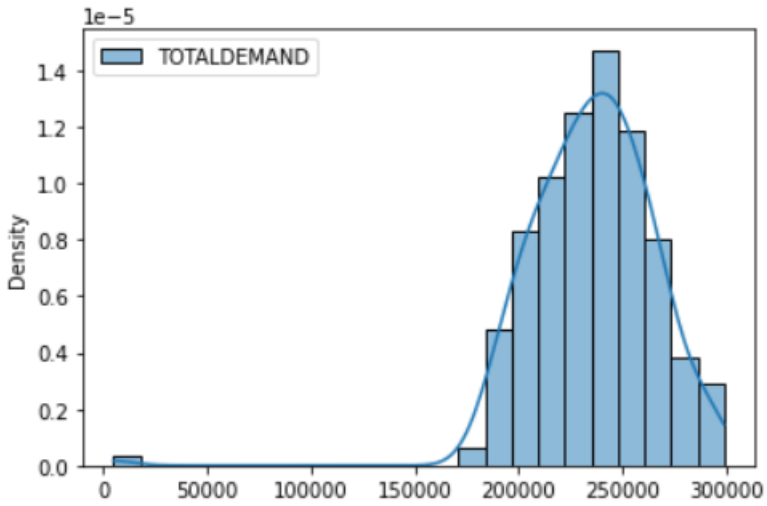
	TOTALDEMAND
Date	
2021-01-01	185853.37
2021-01-02	197990.13
2021-01-03	188742.96
2021-01-04	199281.07
2021-01-05	207680.91

This plot below is the total daily energy demand in Melbourne.

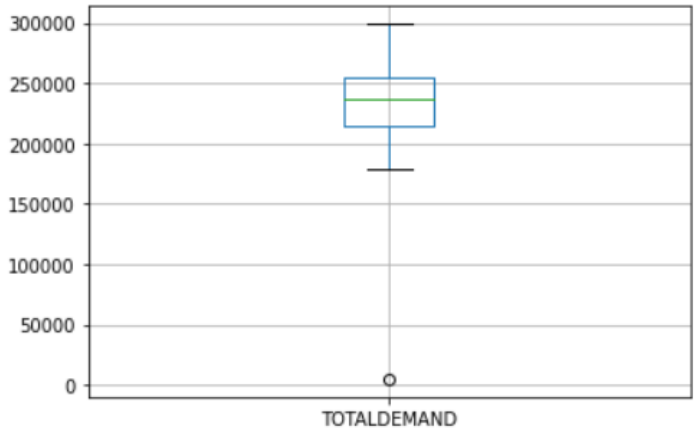



```
sns.histplot(sum_price, kde=True, stat="density")
```

```
<AxesSubplot:ylabel='Density'>
```




```
sum_price.boxplot();
```



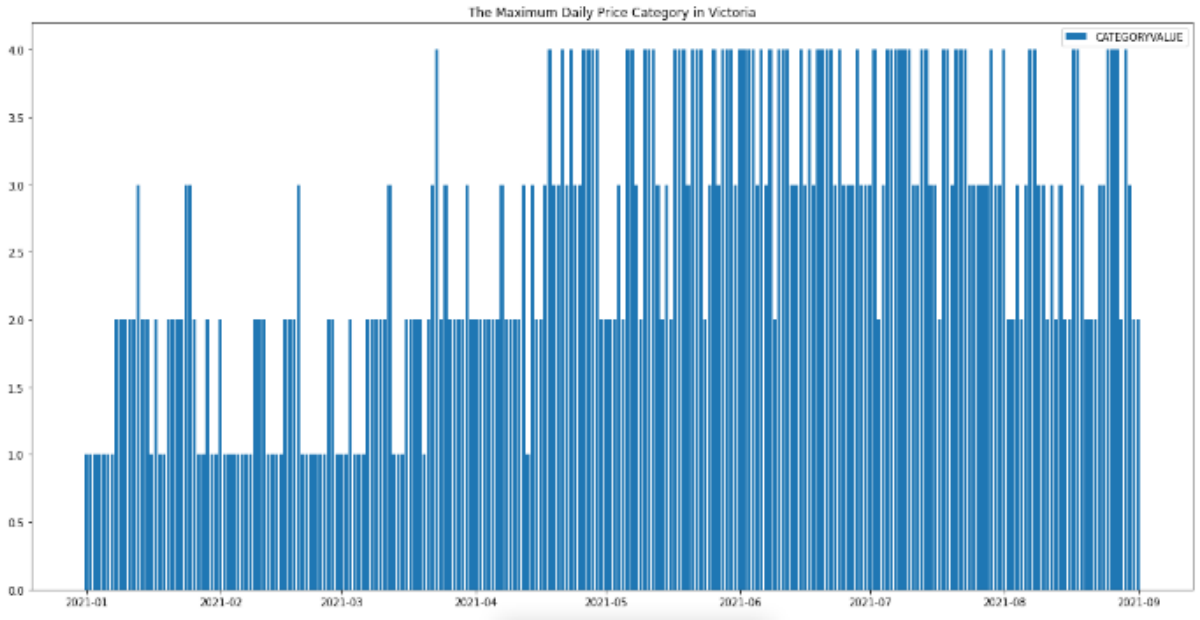
Model 2 is used to predict maximum daily price category, but the category value within the data is a string format. For the data to be manipulated we used `replace ()` to assign 1-Low, 2-Medium, 3-High, 4-Extreme.

`groupby('Date')['CATEGORYVALUE']max()` is then used to identify the maximum price category value for each day.

```
Date
2021-01-01    1
2021-01-02    1
2021-01-03    1
2021-01-04    1
2021-01-05    1
2021-01-06    1
2021-01-07    1
2021-01-08    2
2021-01-09    2
2021-01-10    2
Name: CATEGORYVALUE, dtype: int64
```

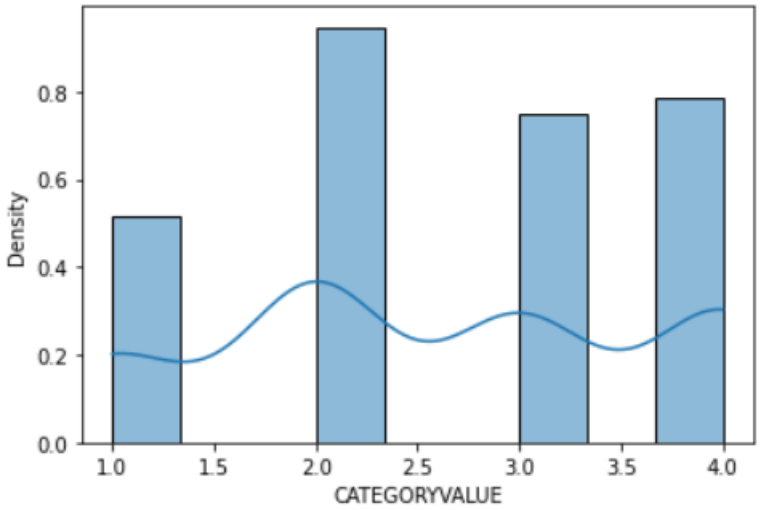
Below is the plot of the category values.

```
fig = plt.figure(figsize=(20,10))

plt.bar(category_price.index, category_price.values, label = "CATEGORYVALUE"

plt.title("The Maximum Daily Price Category in Victoria")
plt.legend()
plt.show()
```



```
sns.histplot(category_price, kde=True, stat="density")
```

```
<AxesSubplot:xlabel='CATEGORYVALUE', ylabel='Density'>
```



Weather Dataset

For `weather_data,` data wrangling is done in the same method as those completed for `price_demand_data`

`pd.read_csv(), head(), info()` are used to import data and below is the output of the information in a readable format.

```
# Load in the weather data
weather = pd.read_csv('weather_data.csv')
display(weather.head())
```

	Date	Minimum temperature (°C)	Maximum temperature (°C)	Rainfall (mm)	Evaporation (mm)	Sunshine (hours)	Direction of maximum wind gust	Speed of maximum wind gust (km/h)	Time of maximum wind gust	9am Temperature (°C)	...	9am cloud amount (oktas)	9am wind direction	9am wind speed (km/h)	9am MSL pressure (hPa)	3pm Temperature (°C)	3pm relative humidity (%)	3pm cloud amount (oktas)
0	1/01/2021	15.6	29.9	0.0	2.8	9.3	NNE	31.0	13:14	19.2	...	6	N	2	1018.8	28.1	43	5.0
1	2/01/2021	18.4	29.0	0.0	9.4	1.3	NNW	30.0	8:22	23.3	...	7	NNW	17	1013.3	28.7	38	7.0
2	3/01/2021	17.0	26.2	12.6	4.8	7.1	WSW	33.0	17:55	18.3	...	8	WSW	4	1007.7	23.5	59	4.0
3	4/01/2021	16.0	18.6	2.6	3.8	0.0	SSE	41.0	16:03	16.2	...	8	SSE	11	1010.0	18.2	82	8.0
4	5/01/2021	15.9	19.1	11.2	1.0	0.0	SSE	35.0	11:02	17.2	...	8	SSE	13	1012.5	18.2	82	8.0

5 rows × 21 columns

By visually checking the data, it can be observed that some features data types are objects and some have missing values.

```
# Check data information
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243 entries, 0 to 242
Data columns (total 21 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Date                             243 non-null    object
 1   Minimum temperature (°C)         242 non-null    float64
 2   Maximum temperature (°C)         242 non-null    float64
 3   Rainfall (mm)                    241 non-null    float64
 4   Evaporation (mm)                 243 non-null    float64
 5   Sunshine (hours)                 243 non-null    float64
 6   Direction of maximum wind gust   240 non-null    object
 7   Speed of maximum wind gust (km/h)  240 non-null  float64
 8   Time of maximum wind gust        240 non-null    object
 9   9am Temperature (°C)             242 non-null    float64
 10  9am relative humidity (%)        242 non-null    float64
 11  9am cloud amount (oktas)         243 non-null    int64
 12  9am wind direction               242 non-null    object
 13  9am wind speed (km/h)            242 non-null    object
 14  9am MSL pressure (hPa)           241 non-null    float64
 15  3pm Temperature (°C)             243 non-null    float64
 16  3pm relative humidity (%)        243 non-null    int64
 17  3pm cloud amount (oktas)         242 non-null    float64
 18  3pm wind direction               243 non-null    object
 19  3pm wind speed (km/h)            243 non-null    object
 20  3pm MSL pressure (hPa)           242 non-null    float64
dtypes: float64(12), int64(2), object(7)
memory usage: 40.0+ KB
```

```
# Detect missing values
weather.isna().sum()
```

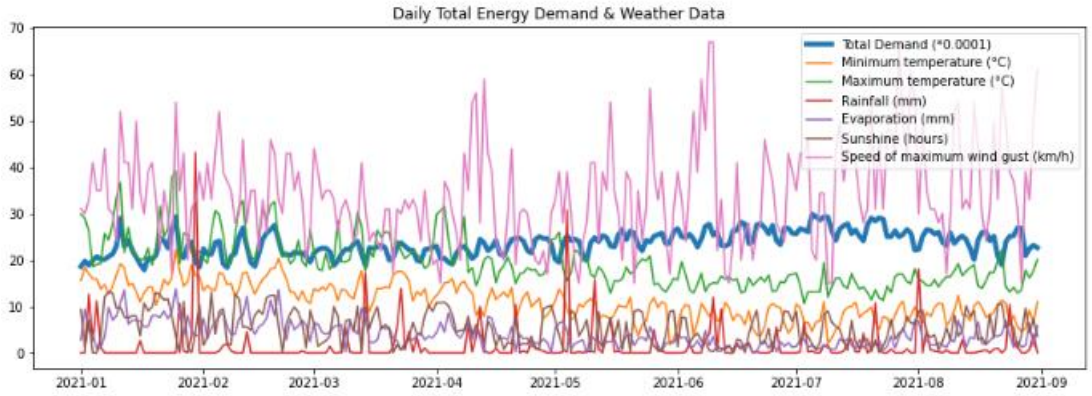
```
Minimum temperature (°C)              1
Maximum temperature (°C)              1
Rainfall (mm)                         2
Evaporation (mm)                      0
Sunshine (hours)                      0
Direction of maximum wind gust        3
Speed of maximum wind gust (km/h)     3
Time of maximum wind gust             3
9am Temperature (°C)                  1
9am relative humidity (%)             1
9am cloud amount (oktas)              0
9am wind direction                    1
9am wind speed (km/h)                 1
9am MSL pressure (hPa)                2
3pm Temperature (°C)                  0
3pm relative humidity (%)             0
3pm cloud amount (oktas)              1
3pm wind direction                    0
3pm wind speed (km/h)                 0
3pm MSL pressure (hPa)                1
dtype: int64
```

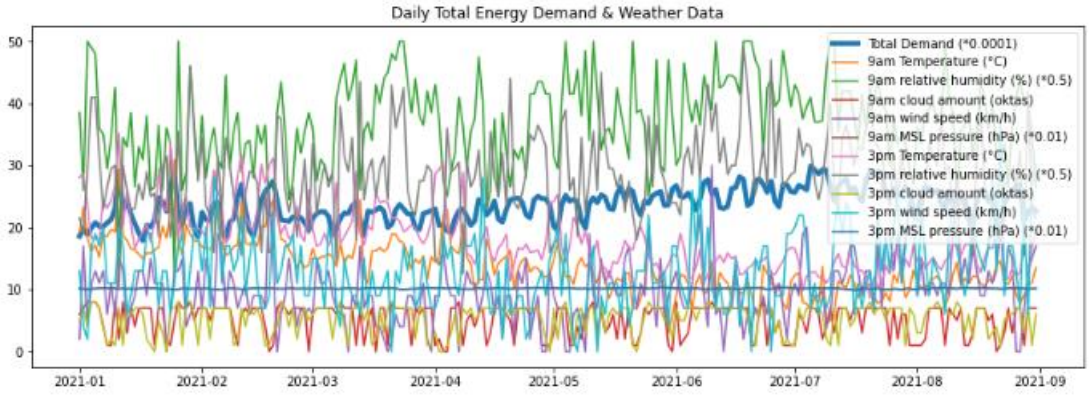
It can be seen that [9am wind speed] and [3 am wind speed] are object, which we convert to numeric using `pd.to_numeric(),` and also we replaced 'Calm' to 0 based on its definition.

Some missing values are found by using `isna().sum(),` so the missing values were replaced by mean of each column.

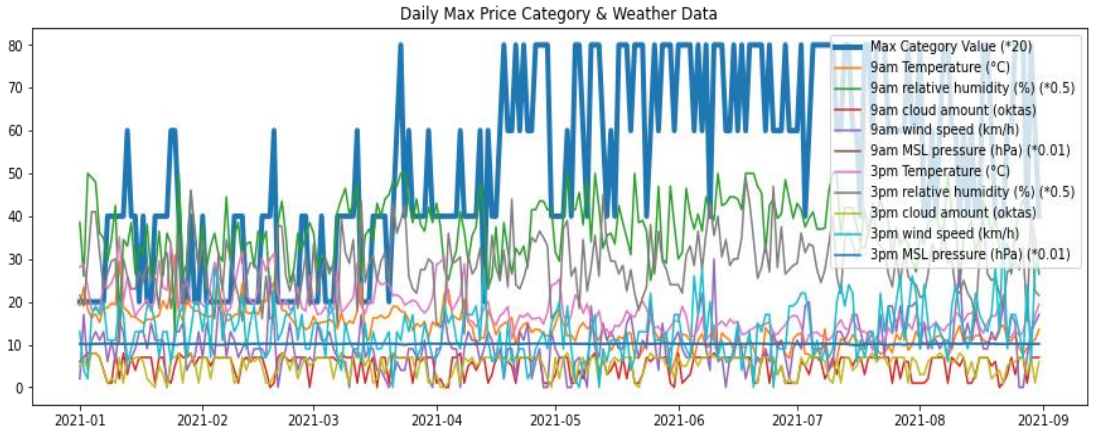
Plotted below is the information extracted from `weather_data` for Melbourne between January and August 2021.

Model 1:





Model 2:



Modelling & Evaluation

Before modelling, Granger's Causality Test is used to select relevant features. The purpose of the test is to identify the relationship between each feature. The null hypothesis is a classification where the values of TOTALDEMAND (x) that does not affect other features (y) is classified as null values (or non-relatable values). If the tested p-value is less than the significance level (in this instance 0.05 is used). Then, we can filter the null values and only

withhold the values where other features (x) that have a direct relationship to cause TOTALDEMAND / CATEGORYVALUE (y).

Model 1:

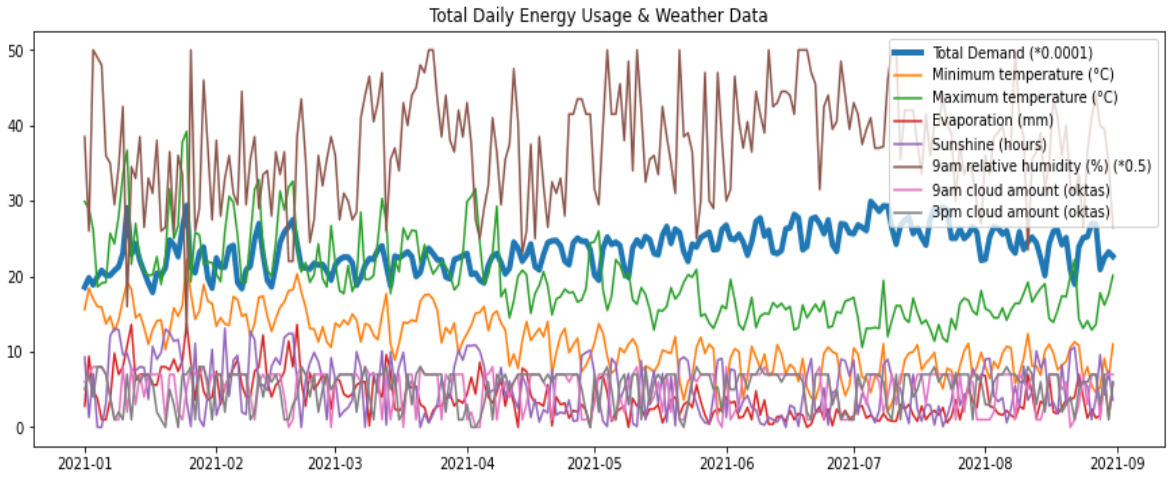
	TOTALDEMAND_x	Minimum temperature (°C)_x	Maximum temperature (°C)_x	Rainfall (mm)_x	Evaporation (mm)_x	Sunshine (hours)_x	Speed of maximum wind gust (km/h)_x	9am Temperature (°C)_x	9am relative humidity (%)_x	9am cloud amount (oktas)_x	9am wind speed (km/h)_x	9am MSL pressure (hPa)_x	3pm Temperature (°C)_x
TOTALDEMAND_y	1.0000	0.0000	0.0000	0.3398	0.0010	0.0012	0.1785	0.0000	0.0023	0.0000	0.2043	0.0039	0.0000

Model 2:

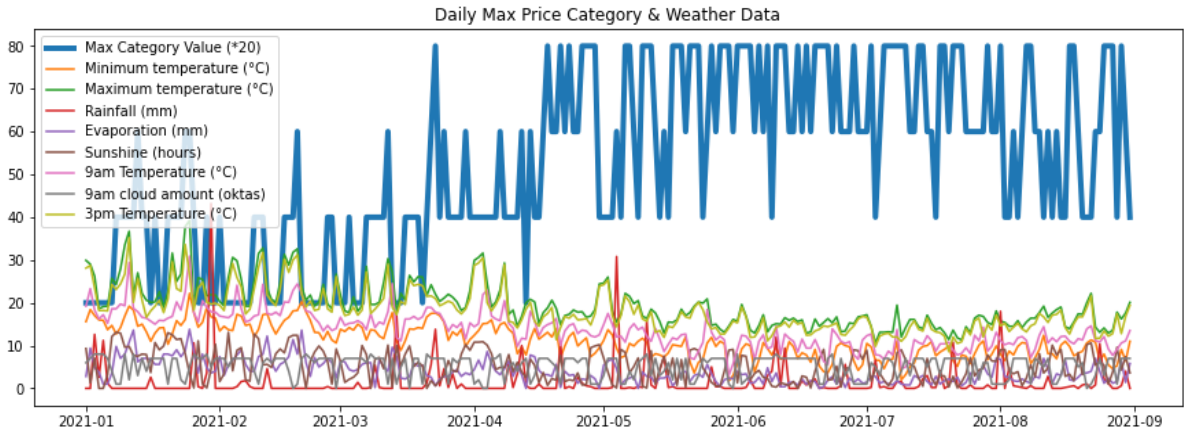
	CATEGORYVALUE_x	Minimum temperature (°C)_x	Maximum temperature (°C)_x	Rainfall (mm)_x	Evaporation (mm)_x	Sunshine (hours)_x	Speed of maximum wind gust (km/h)_x	9am Temperature (°C)_x	9am relative humidity (%)_x	9am cloud amount (oktas)_x	9am wind speed (km/h)_x	9am MSL pressure (hPa)_x
CATEGORYVALUE_y	1.0000	0.0000	0.0000	0.0188	0.0000	0.0196	0.1377	0.0000	0.7285	0.0313	0.1173	0.3413

Below are the filtered data/plots (null values removed):

Model 1:



Model 2:

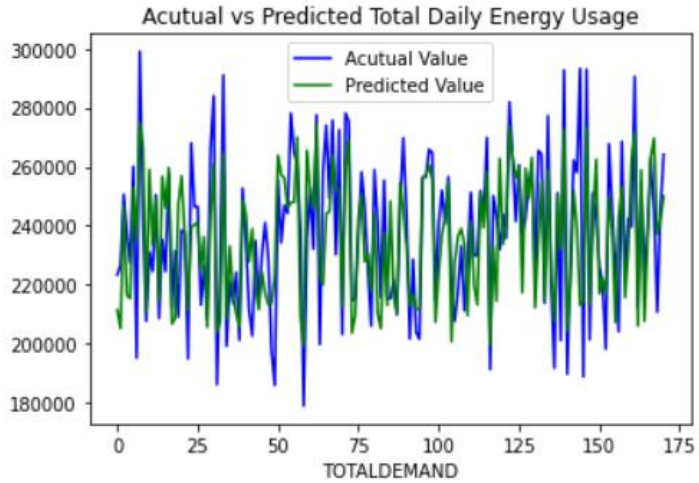


Random Forest

Based on two models which are used to predict continuous number and category respectively, we decide to use Random Forest (RF). RF has good performance when using only default parameters and it has both a Regressor and a Classifier to do it. RF is a combination of many different decision trees based on bootstrap samples. Where a dataset contains 'x' samples, the RF algorithm is used, it will select 'x' times from the sample and each time it selects one sample, that sample will be put back in the dataset. Therefore, the real train data may have duplicated samples on each node, and the algorithm will choose a subset of features and find the best solution based on only one feature. The feature selection is based on lowest entropy value or MSE number for Classifier or Regressor. The subset of features is mutually exclusive to each other on each node, so every node of the tree can make a decision based on different subsets of features. Finally, a soft voting algorithm is used for Classifier, where the final result is the most frequent result among results of all different trees and mean value of all results for Regressor.

Random Forest Regressor

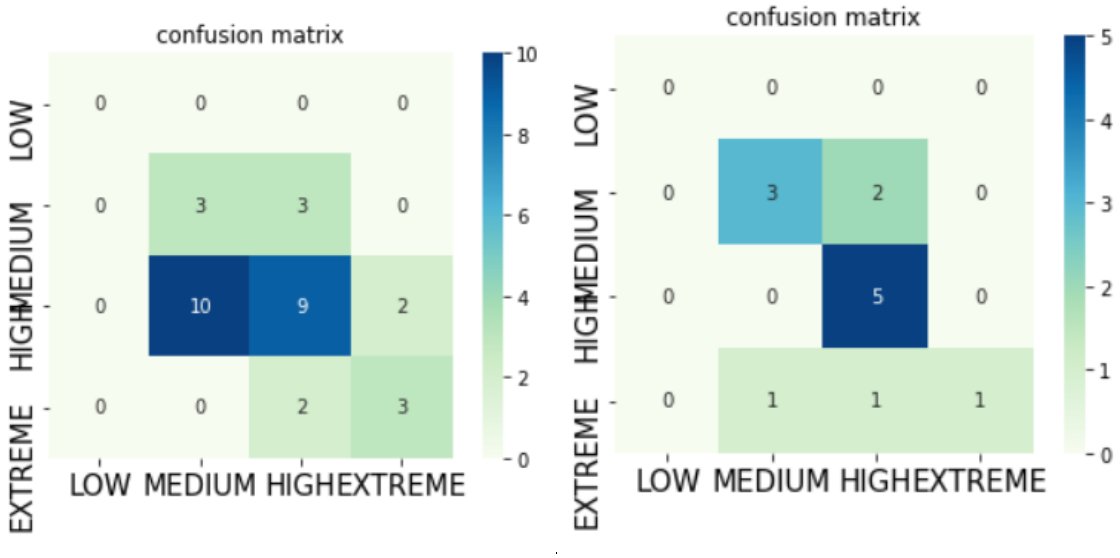
For Model 1, we spilt dataset into 7:3 of train:test on `train_test_split (test_size=0.7)` and substantialize `RandomForestRegressor (n_estimators=200)` where `n_estimators` is the number of decision trees in Forest, which is bigger, the result is more accurate but more resource-consuming. Considering this dataset is not large, 200 is a feasible value. Then use `fit()` and `predict()` to train and predict. Finally, we used `metrics.r2_score()` to calculate R2 value which represents the fraction of variance of the actual value of the response variable captured by the regression model to measure the performance. Generally, it reflects how much the predicted regression line matches the real data line. It is better if the value is closer to 1.



Random Forest Classifier

For Model 2, we split data into 7:2:1 of train:validation:test. Train set is used to train the model, validation is used to find the best model based on parameters tuning and a test set is used to test the model. `Score()` is used to test the accuracy, which validation is 0.64 and test is 0.57. Finally, we chose estimators as 50.

Confusion Matrix is one of the evaluation methods. Figures below represent the test and validation set, repressively. From the figures, it can be seen that the model has more accurate predictions on MEDIUM and HIGH. Most predictions are around True-Positive values.



Meanwhile `classification_report()` is used to check precision, recall and f1-score, which are showed below. From the figures, it can be seen three main metrics are aligned with confusion matrix results.

```
res = classification_report(test_y.values, pred_y)
res.split('\n')
```

```
['              precision    recall  f1-score   support',
 '',
 '           1       0.23      0.50      0.32         6',
 '           2       0.64      0.39      0.49        23',
 '           3       0.33      0.38      0.35         8',
 '           4       0.67      0.71      0.69        14',
 '',
 '    accuracy                           0.49        51',
 '   macro avg       0.47      0.50      0.46        51',
 'weighted avg       0.55      0.49      0.50        51',
 '']
```

```
res = classification_report(val_y.values, pred_y)
res.split('\n')
```

```
['              precision    recall  f1-score   support',
 '',
 '           1       0.75      0.60      0.67         5',
 '           2       0.56      1.00      0.71         5',
 '           3       0.50      0.20      0.29         5',
 '           4       0.71      0.71      0.71         7',
 '',
 '    accuracy                           0.64        22',
 '   macro avg       0.63      0.63      0.60        22',
 'weighted avg       0.64      0.64      0.61        22',
 '']
```

Insights

The Granger causality test is used to test the correlation between variables and targets. The threshold is set to 0.05. When it is greater than 0.05, we believe that the impact of the variable is too small to the target and discard it. The variables that are extracted are different depending on the tasks.

For the prediction of total daily energy consumption, the variables are;

- Minimum temperature,
- Maximum temperature,
- Evaporation,
- Sunshine,
- 9am relative humidity,
- 9am cloud amount,
- 3pm cloud amount,

(When considering the daily total demand, the set of 9am & 3pm Temperatures are highly similar to the set of Minimum & Maximum temperatures as well as being within the range between Minimum & Maximum. Therefore 9am & 3pm Temperatures were not selected in Model 1)

But when predicting daily highest price category,

- Minimum temperature,
- Maximum temperature,
- Rainfall,
- Evaporation,
- Sunshine,
- 9am Temperature,
- 9am cloud amount,
- 3pm Temperature

are the variables that are selected as their p-values are all less than 0.05, which means that these variables in the dataset are influencing the total demand of energy.

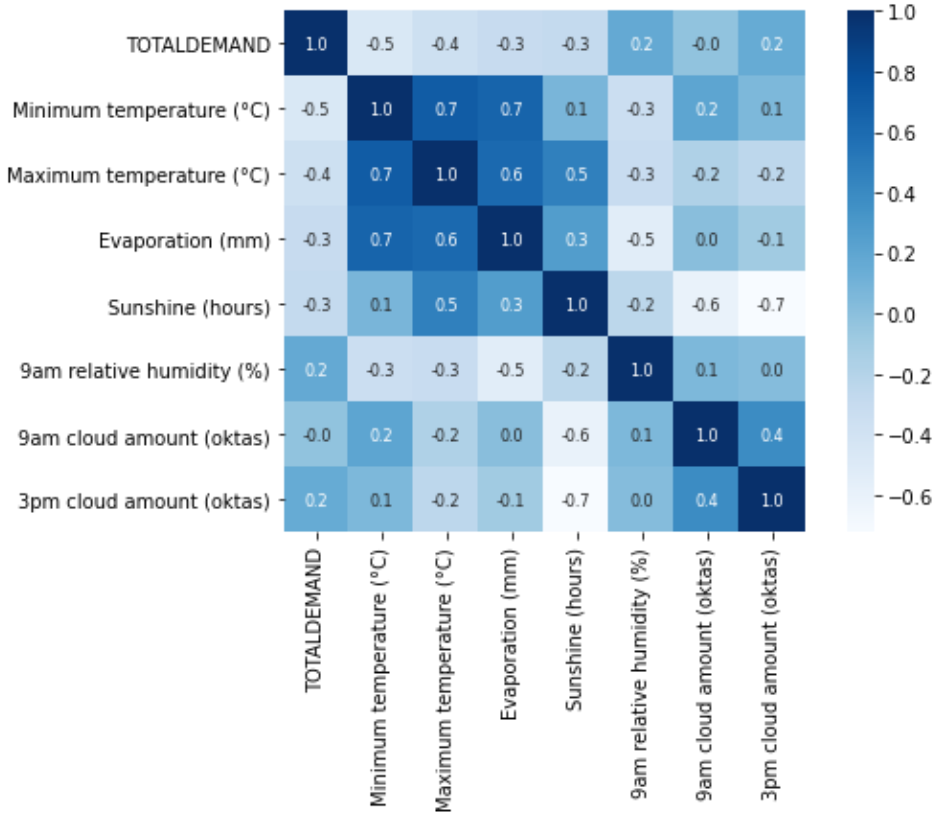
However, the P-values for the following categories;

- Rainfall (mm)_x (in Model 1),
- Speed of maximum wind gust (km/h)_x,
- 9am wind speed (km/h)_x,
- 9am MSL pressure (hPa)_x,
- 3pm relative humidity (%)_x,
- 3pm wind speed (km/h)_x and
- 3pm MSL pressure (hPa)_x

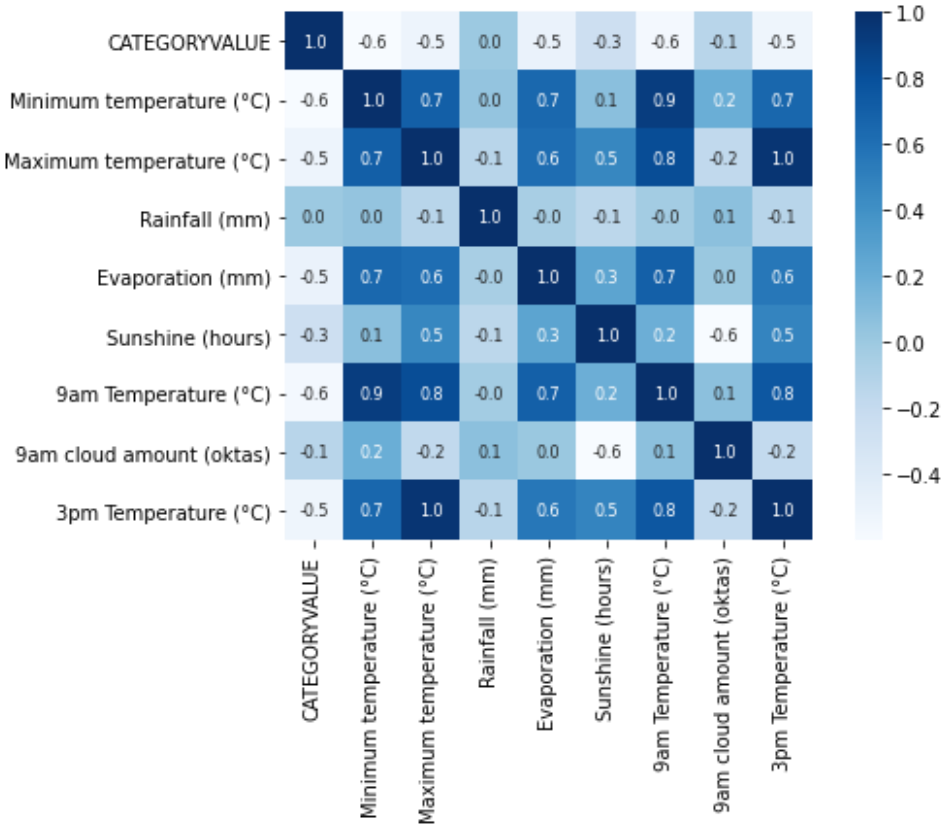
are greater than 0.05, which means that these variables in these dataset are not influencing the total daily energy consumption.

Below are the heatmaps of correlations between selected features

Model 1:



Model 2:



From the correlation heatmaps above, we can see that the Minimum temperature & Maximum temperature have the first and second strongest negative correlations to both Total demand and Category value. Similarly, 9am Temperature & 3pm Temperature strongly negative correlate to Category value. Evaporation & Sunshine also have significant negative correlations to both Total demand and Category value. Generally, temperatures are most valuable inputs to predict energy usage/price.

Significant and Valuable

The results detailed in this report are significant and valuable as the energy consumption and price of the day can now be predicted using the weather forecast data.

The regressor's Predicted vs. Actual value plot and classification's report show that both of our models have high accuracies. The models can be used to help companies rationalize energy usage and prices for its long- and short-term business planning.

For general public, being able to know the demand and price of energy in time helps them to make appliance usage decisions. In terms of protecting our environment, we hope to provide relevant data to the government and enterprises so that precise and reliable environmental policies can be introduced to help slow down the progress of global warming.

Limitations and Future Directions

Our result is limited due to the amount of data that is used is too small. Because of this, we do not want to use an overly complex neural network model. Machine learning can outperform neural networks when insufficient data provided. The models can be improved by importing large amount of relevant data.

Github repository link:

https://github.com/Joanne-zhou/Assignment-2_Group-9