ASSIGNMENT 2: REPORT

GitHub Repository : https://github.com/Rloanuald/DATA0006_assign_2.git

INTRODUCTION

In this report, there are two datasets used for our modelling. The approach and algorithms are based on **Supervised Machine Learning (predictive modelling)**.

We used data mining techniques:

- Data cleaning preprocessing,
- Predictive modelling (Classification for categorical class label & Regression for numerical class label),
- Decision Tree,
- kFold,
- KNN,
- Chi square test,
- SelectKBest,
- PCA and
- Train/Test method.

The various stages of modelling included:
a) Data Collection and Extraction
b) Data Transformation
c) Data Analytics and Machine Learning Algorithms

The Python libraries like Pandas, Numpy, Matplotlib, Seaborn, Sklearn, Scipy were used.

DATA IMPLEMENTATION

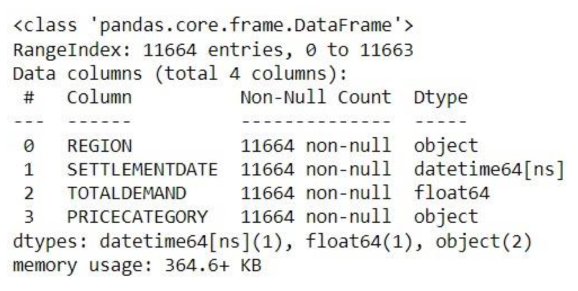
Data source

In this report, we sourced the two raw datasets, Price-Demand and Weather
For the **Price-Demand dataset** (dependent dataset), the dataset contained a series of measurements every 30 minutes, including four columns and none null values as depicted in Table 1 and Figure1.

Columns	Example
REGION	VIC1
SETTLEMENTDATE	1/01/2021 0:30
TOTALDEMAND	4179.21
PRICECATEGORY	LOW

[1]*Source: https://www.energy.gov.au/data/electricity-generation)*

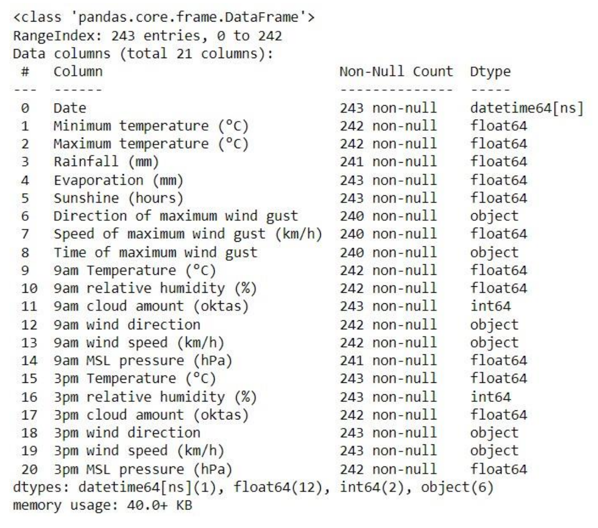
Table 1



```
<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  datetime64[ns]
 2   TOTALDEMAND     11664 non-null  float64
 3   PRICECATEGORY   11664 non-null  object
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 364.6+ KB
```

Figure 1: Price-demand data variables

Weather dataset (independent dataset) containing 21 columns and some null values as shown in Figure 2.



```
<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    datetime64[ns]
 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: datetime64[ns](1), float64(12), int64(2), object(6)
memory usage: 40.0+ KB
```

Figure 2: Weather data variables

Data exploration

Figure 3 shows the distribution of variables. From the figure below it is evident that all the temperature variables follow a normal distribution except the maximum temperature which is left skewed. 9am and 3pm cloud amount (oktas) consists of numerous zero values. The rainfall variable also contains numerous zero values. Thus, data visualisation helps in making the model more accurate for prediction.

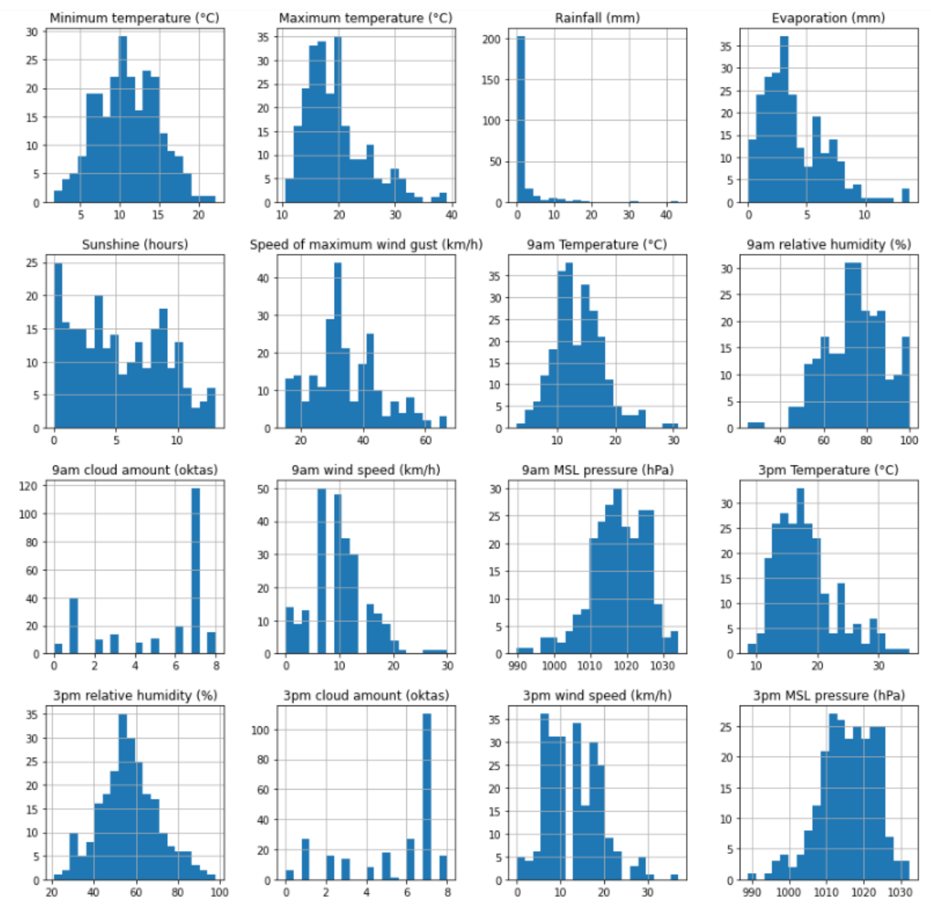


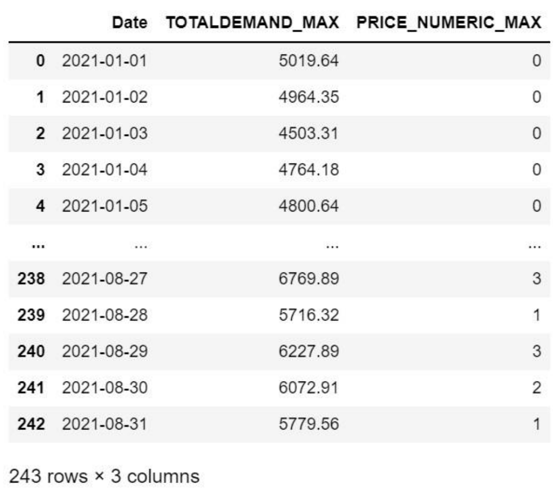
Figure 3: The nature of the distribution

According to the data provider, the daily time period starts on the first day at 0:30 upto the following day at 0:00. For example, the first day "1st January 0:30 - 2nd January 0:00" is considered the previous day as 1st January. Therefore, each day is grouped by 48 rows of 30-minute time slots each in the dataset. As a result, the maximum daily total demand and the maximum daily price category were extracted.
In the Price-Demand dataset, the categories were replaced with integers/ numeric values as shown in Table 2 for data analysis later.

LOW	0
MEDIUM	1
HIGH	2
EXTREME	3

Table 2

Extracted the maximum demand and maximum price from the Price-Demand dataset, as shown below in Figure 4.

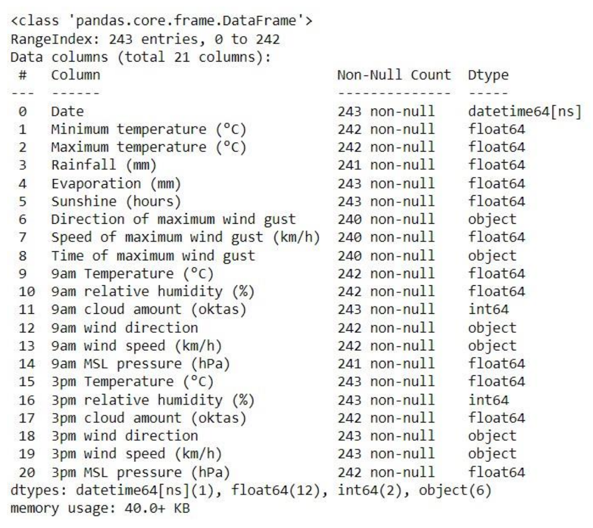


	Date	TOTALDEMAND_MAX	PRICE_NUMERIC_MAX
0	2021-01-01	5019.64	0
1	2021-01-02	4964.35	0
2	2021-01-03	4503.31	0
3	2021-01-04	4764.18	0
4	2021-01-05	4800.64	0
...	...	...	...
238	2021-08-27	6769.89	3
239	2021-08-28	5716.32	1
240	2021-08-29	6227.89	3
241	2021-08-30	6072.91	2
242	2021-08-31	5779.56	1

243 rows × 3 columns

Figure 4: Maximum total demand and maximum price category

After extracting necessary data from the two datasets, 'max_price_demand' and 'weather', they are merged together into one dataframe named as 'all_dataset' comprising 21 columns and 243 rows.



```
<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    datetime64[ns]
 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: datetime64[ns](1), float64(12), int64(2), object(6)
memory usage: 40.0+ KB
```

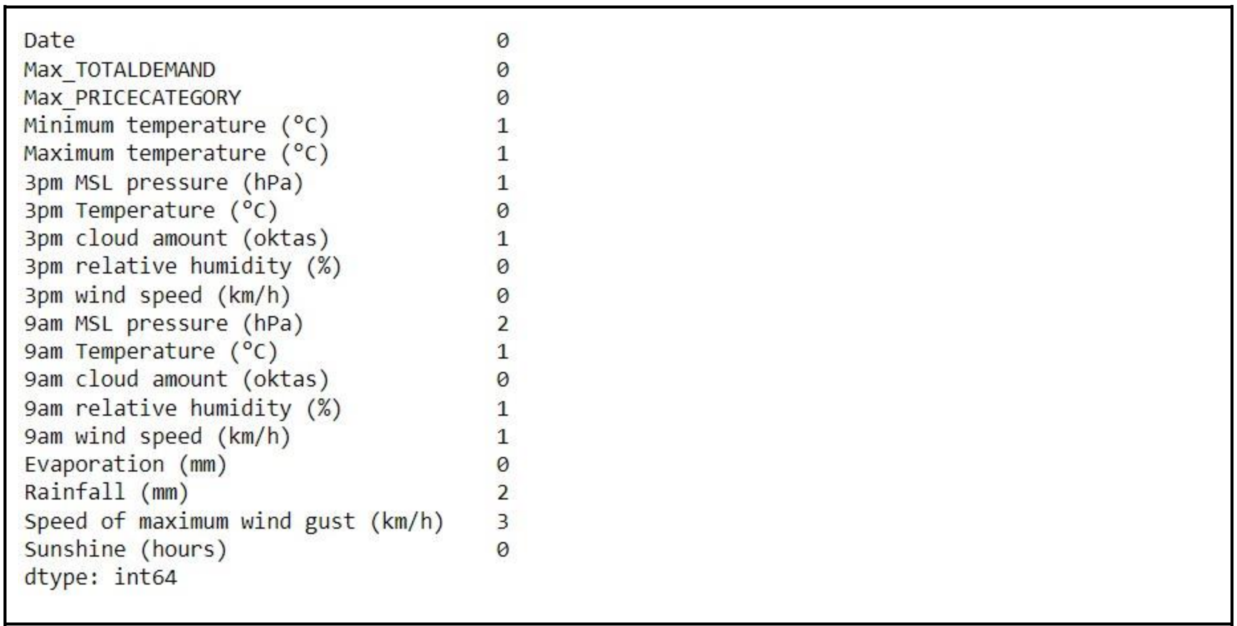
Figure 5: Data variables

Data Preprocessing

We used python libraries (pandas, numpy, matplotlib, seaborn, datetime, sklearn, scipy) for importing various functions which allows us to clean and explore data.

Firstly, we have chosen to load the two raw datasets with the 'parse date' method because it can convert the string "1/01/2021 0:30" and parse it to a standardised date format '2020–01–01 00:30:00' in date type 'datetime64'. So the date will not be object type and require us to convert the date format manually. This is very important for merging the two datasets by the same date format and data type at ease.

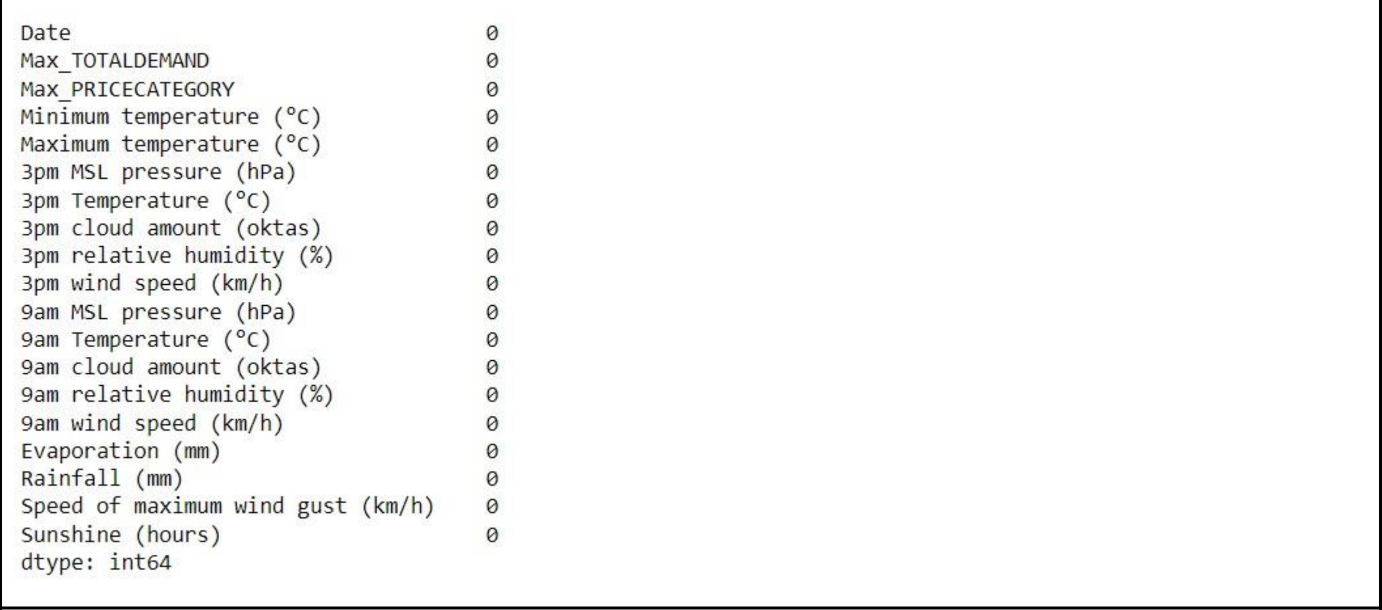
Since the raw dataset consists of null values as depicted in Figure 6-1, we use the .isnull() method to check how many rows contain null values. We chose to find null instead of deleting the rows because we ensure there is no data reduction.



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

Figure 6-1: Null values

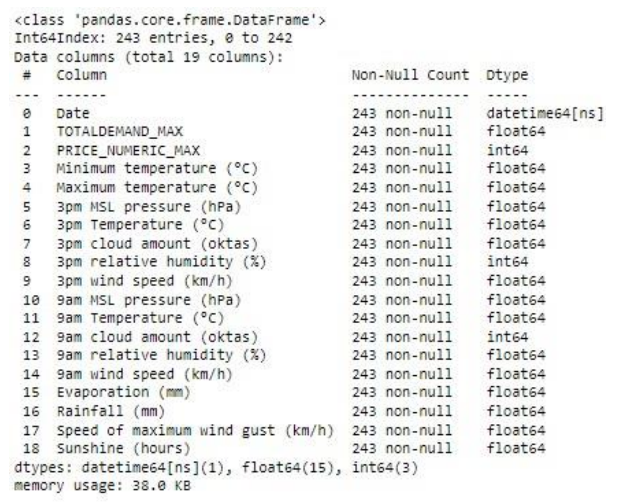
Then, we use the .fillna() method to fill the null values with the mean value by applying the .mean() method. We choose the mean method over median (middle value) and mode (most repeated value) because it includes all data in the calculations. For the 9am and 3pm wind speed columns, all data cells containing 'Calm' are replaced with numeric value 0 which represents calm wind according to Beaufort scale, so that the column can be converted to numeric type. A summary of non-null values are shown below in Figure 6-2.



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

Figure 6-2: Cleaned null values

After data cleaning, the datatype of the columns are converted from object to numeric type. A summary of the clean dataset is shown below in Figure 7.



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 243 entries, 0 to 242
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Date                            243 non-null    datetime64[ns]
 1   TOTALDEMAND_MAX                 243 non-null    float64
 2   PRICE_NUMERIC_MAX               243 non-null    int64
 3   Minimum temperature (°C)        243 non-null    float64
 4   Maximum temperature (°C)        243 non-null    float64
 5   3pm MSL pressure (hPa)          243 non-null    float64
 6   3pm Temperature (°C)            243 non-null    float64
 7   3pm cloud amount (oktas)        243 non-null    float64
 8   3pm relative humidity (%)       243 non-null    int64
 9   3pm wind speed (km/h)           243 non-null    float64
 10  9am MSL pressure (hPa)          243 non-null    float64
 11  9am Temperature (°C)            243 non-null    float64
 12  9am cloud amount (oktas)        243 non-null    int64
 13  9am relative humidity (%)       243 non-null    float64
 14  9am wind speed (km/h)           243 non-null    float64
 15  Evaporation (mm)                243 non-null    float64
 16  Rainfall (mm)                   243 non-null    float64
 17  Speed of maximum wind gust (km/h)  243 non-null    float64
 18  Sunshine (hours)                243 non-null    float64
dtypes: datetime64[ns](1), float64(15), int64(3)
memory usage: 38.0 KB
```

Figure 7: Complete dataset info

MODEL 1 – PREDICTION OF MAXIMUM DAILY ENERGY USAGE

Feature Selection using Pearson's Correlation

To get more understanding of the data, we plotted a heatmap showing the correlation between all the features and the target variables. This is helpful as we can ascertain the correlation between all the features and the target features, which will help us in the feature selection. Few of the features we have in our data are redundant as they are highly correlated to each other. For example, Maximum Temperature vs 3pm Temperature has a correlation of 0.97 as per the heat map (Figure 8) and the correlation values below (Figure 9). Features that are strongly correlated with each other will compromise the effectiveness of the model.

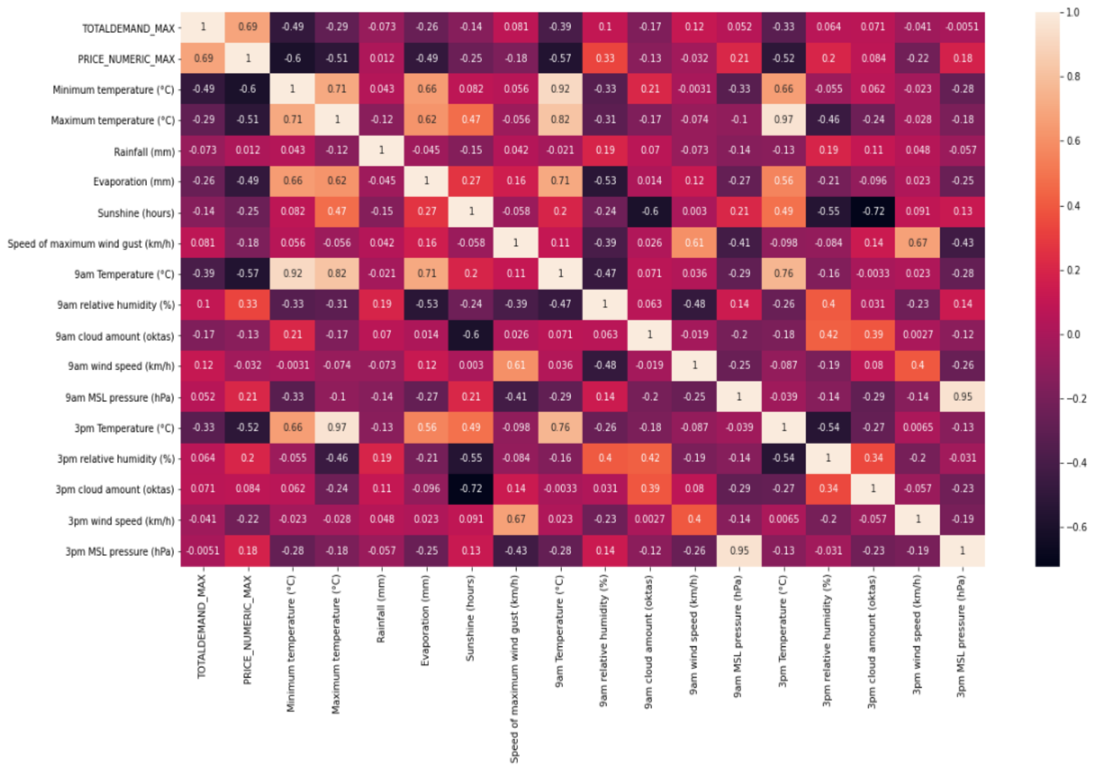
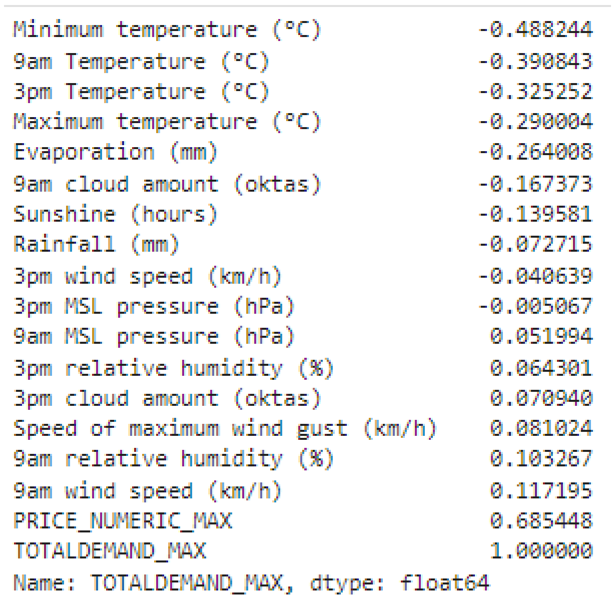


Figure 8: Heatmap

We also calculated the correlation between the features and the Maximum Total Demand as follows (Figure 9):



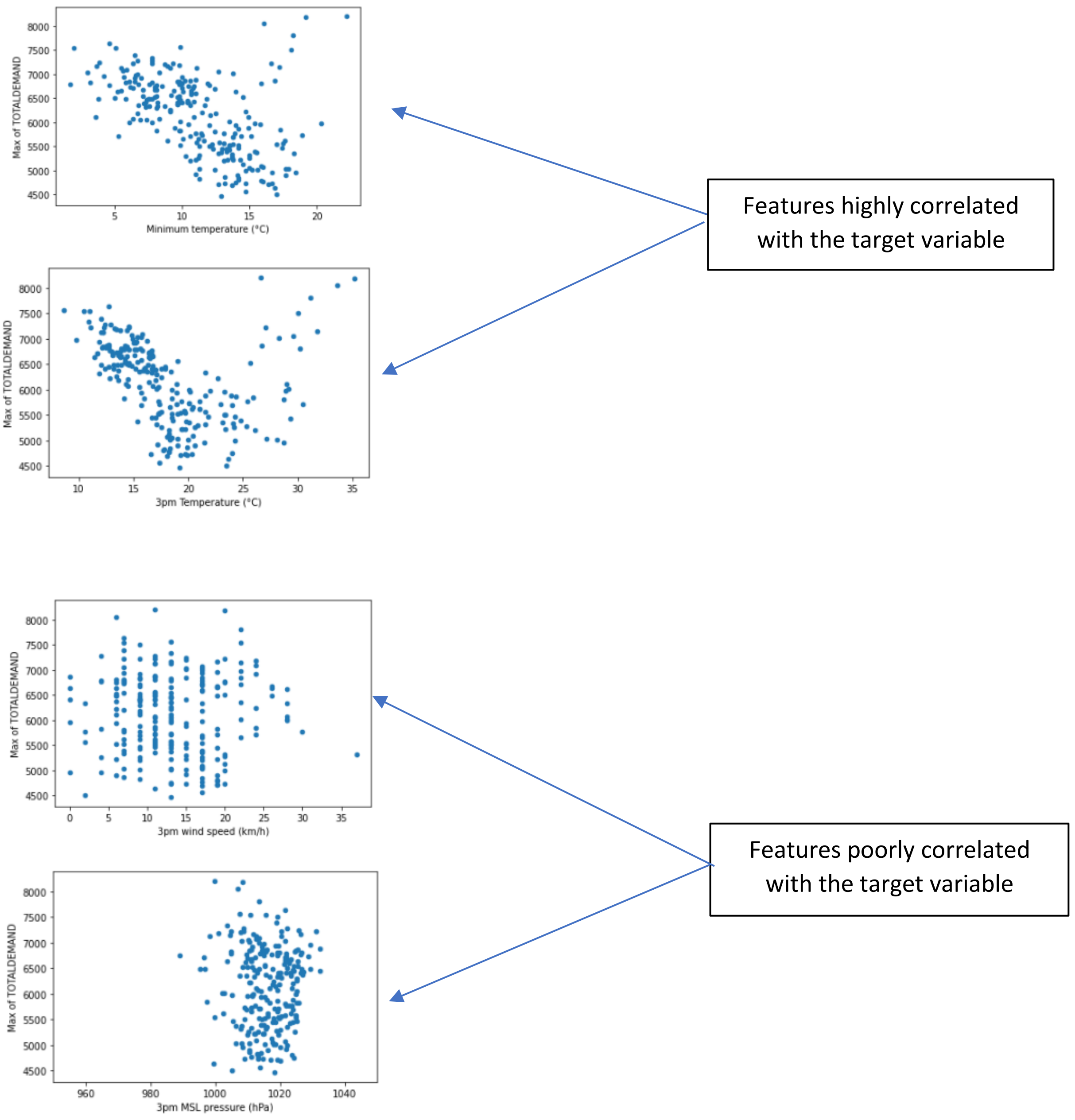
```
Minimum temperature (°C)            -0.488244
9am Temperature (°C)                -0.390843
3pm Temperature (°C)                -0.325252
Maximum temperature (°C)            -0.290004
Evaporation (mm)                    -0.264008
9am cloud amount (oktas)            -0.167373
Sunshine (hours)                    -0.139581
Rainfall (mm)                       -0.072715
3pm wind speed (km/h)               -0.040639
3pm MSL pressure (hPa)              -0.005067
9am MSL pressure (hPa)               0.051994
3pm relative humidity (%)            0.064301
3pm cloud amount (oktas)             0.070940
Speed of maximum wind gust (km/h)    0.081024
9am relative humidity (%)            0.103267
9am wind speed (km/h)                0.117195
PRICE_NUMERIC_MAX                    0.685448
TOTALDEMAND_MAX                      1.000000
Name: TOTALDEMAND_MAX, dtype: float64
```

Figure 9:F Correlation between Features and Maximum Total Demand

There are high negative correlations between the features and Maximum Total Demand (target variable) with the top four temperature features being the most negative. Intuitively, these correlations make sense since the low or high temperature during the day will impact the amount of energy consumption.

Feature Selection using f_regression

We have used the f_regression feature selection to calculate the interaction between the features and the target variable and to rank them according to the significance of the regression parameter.



Few of the features we have in our data are redundant as they are highly correlated to each other. For example, Maximum Temperature vs 3pm Temperature has a correlation of 0.97 as per the heat map (Figure 8) and the table (Figure 9) above. Features that are strongly correlated with each other will compromise the effectiveness of the model.

Thus, based on the analysis above we have used ‘Practical wrapper approach – Decremental’ and choosen the below features:

```
Minimum temperature (°C)
Evaporation (mm)
3pm Temperature (°C)
Speed of maximum wind gust (km/h)
Sunshine (hours)
3pm relative humidity (%)
3pm cloud amount (oktas)
```

Model Selection

Regression Analysis is best suited to this problem as the target values of ‘Maximum Daily Energy Usage’ are on a continuous scale and not discrete. Multiple Linear Regression will determine the numerical relationship between the target variable and the other variables.

Train/Test Data

The model is trained to predict the known output and later tested on the test data so that we can generalize it to other non-trained data. Test data is used to test the prediction ability (accuracy) of the model. Training data (X_train, y_train) is used to fit the regression model. Maximum Daily Energy Usage is predicted from the independent variables in the model below.

```python
# Choose the features (input)
X = dataset[['Minimum temperature (°C)', 'Evaporation (mm)', '3pm Temperature (°C)', 'Sunshine (hours)', 'Speed of maximum wind gust (km/h)', '3pm relative humidity (%)']]

# What we have to predict (output)
y = dataset['TOTALDEMAND_MAX']

# Splitting the data into training set and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, random_state = 1)

# Generate the regression model (import .linear_model to use .LinearRegression() func)
lm = linear_model.LinearRegression()

# Create/ Produce the model: Fit training data into model
model = lm.fit(X_train, y_train)
display(model)

# Predict coefficient & intercept from linear regression model(lm)
print(f'Coefficients :{lm.coef_}')
print(f'Intercept :{lm.intercept_}')
```

Finally, r2 - the proportion of the variance explained by the predictors is given by:

```python
r2_test = lm.score(X_test, y_test) #on testing data
```

The result indicates that the predictors account for only 35.1% of the variance explained in the Maximum Daily Energy Usage.

EFFECTIVENESS: The linear regression model doesn’t seem to be much effective since the predictors account for only 35.1% of the variance explained.

The r2 score, also called the coefficient of determination, comes out to be 35.1%.

This statistic indicates the proportion of variance in the dependent variables (Minimum temperature, Evaporation, 3 pm Temperature, Sunshine, Speed of maximum wind gust, 3 pm relative humidity) that can be explained by the independent variable (TOTALDEMAND_MAX) collectively.

Thus, an r2 score of 0.35, can be considered a **weak or low effect size.**

Thus, the effectiveness of the model is a little below average. In the real-world scenario, it doesn't seem to be a decent model due to the following **limitations**:

- Low r2 score
- Insufficient data to predict future energy usage
- Maximum energy depends on other features as well other than weather data.
- Most of the features are with respect to 9 am & 3 pm timelines which restrict the spread of the data
- Most of the features are not highly correlated with the target variable. The maximum Pearson r correlation existing is -0.49.

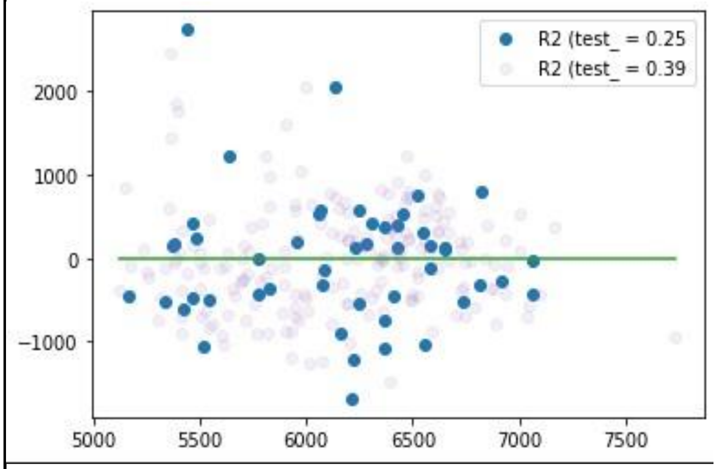
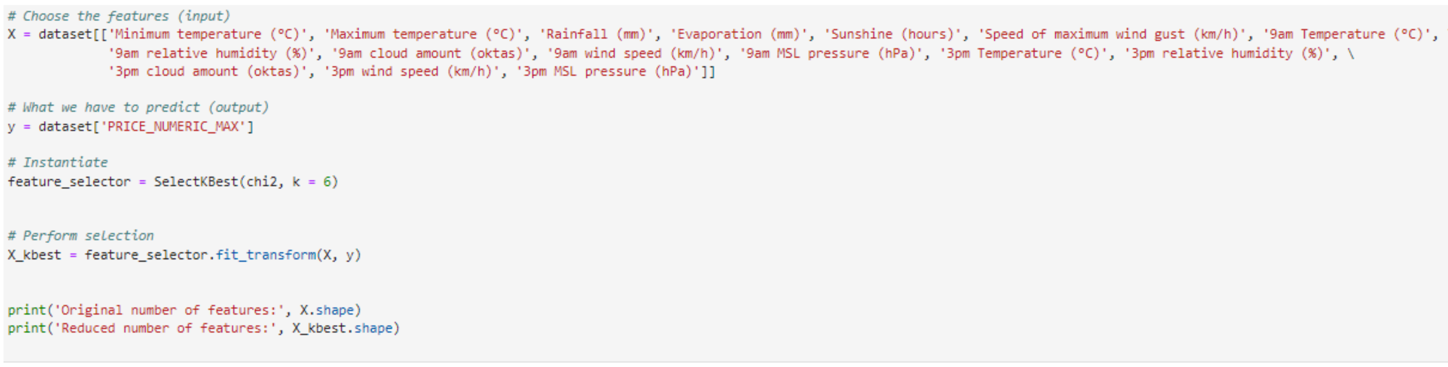


Figure 10 Linear regression graph of Maximum Total Demand in energy

MODEL 2: PREDICTION OF MAXIMUM DAILY PRICE CATEGORY

Feature Selection using Chi Squared Test with k-Fold Cross Validation

Chi-square test (CHI) sorts features based on the class and filters out the top features which depend on the class label. This test is used for categorical features in a dataset. After testing on a number of features we concluded that 6 features are optimal for our model as per the Chi test result.



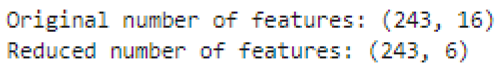
```
# Choose the features (input)
X = dataset[['Minimum temperature (°C)', 'Maximum temperature (°C)', 'Rainfall (mm)', 'Evaporation (mm)', 'Sunshine (hours)', 'Speed of maximum wind gust (km/h)', '9am Temperature (°C)',
            '9am relative humidity (%)', '9am cloud amount (oktas)', '9am wind speed (km/h)', '9am MSL pressure (hPa)', '3pm Temperature (°C)', '3pm relative humidity (%)', \
            '3pm cloud amount (oktas)', '3pm wind speed (km/h)', '3pm MSL pressure (hPa)']]

# What we have to predict (output)
y = dataset['PRICE_NUMERIC_MAX']

# Instantiate
feature_selector = SelectKBest(chi2, k = 6)


# Perform selection
X_kbest = feature_selector.fit_transform(X, y)


print('Original number of features:', X.shape)
print('Reduced number of features:', X_kbest.shape)
```



```
Original number of features: (243, 16)
Reduced number of features: (243, 6)
```

Tested the model using the selected 6 features and used the K-Fold cross validation to evaluate the model. This approach is used so the split comes out to be random and the model is not overfitting.

The below table 3 shows the score with different ‘k’ values:

k-Fold	Knn Accuracy Score	Decision Tree Accuracy Score
5	50.60%	42.40%
10	52.30%	37%
15	53.10%	42.40%
20	53%	36.20%
22	51.8	42.4

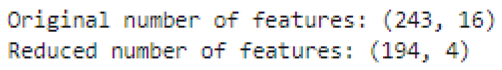
Table 3

OBSERVATION: k = 15 gets the highest accuracy score and a decreasing trend after that in the accuracy. So, on this basis we conclude that the most appropriate value for k = 15.

Feature Selection using PCA Test with k-Fold Cross Validation

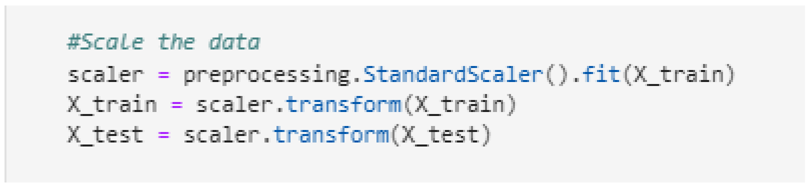
Second method used to perform the feature selection is PCA. This approach is used for dimensionality reduction that allows the reduction in number of variables. We have used this approach in conjunction with k-fold Cross Validation to evaluate our model.

Through the PCA features selection approach we have reduced the 16 features to 4 features:



```
Original number of features: (243, 16)
Reduced number of features: (194, 4)
```

Before applying PCA we have also scaled the data. This is important to avoid the principal components being biased towards the features with high variance which may lead to false results.



```
#Scale the data
scaler = preprocessing.StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Again applied the k-Fold Cross Validation to evaluate the model.

Below are the scores with different no of k-Folds:

k-Fold	Knn Accuracy Score	Decision Tree Accuracy Score
5	55.50%	51.80%
10	55.30%	50%
15	55.23%	52.10%
20	55.20%	51.30%
22	55.19%	51.60%

Table 4

OBSERVATION: k = 5 gets the highest accuracy score and a decreasing trend after that in the accuracy. Thus, we can conclude that the most appropriate value for k = 5.

Model Selection

Based on the above results between 2 feature selection approaches and comparisons with KNN and Decision Tree Model, below is the table showing the accuracy results:

	CHI SQUARED TEST		PCA	
k-Fold	Knn Accuracy Score	Decision Tree Accuracy Score	Knn Accuracy Score	Decision Tree Accuracy Score
5	50.60%	42.40%	55.50%	51.80%
10	52.30%	37%	55.30%	50%
15	53.10%	42.40%	55.23%	52.10%
20	52.70%	36.20%	55.20%	51.30%
22	51.80%	42.40%	55.19%	51.60%

Table 5

Highest Accuracy between all models

EFFECTIVENESS:

We conducted the KNN and Decision Tree classification model to predict maximum price category and out of the two, we believe that KNN is a better and effective model because of better accuracy.

On the basis of above table 5, it is evident that the KNN Model with PCA feature selection is the best fit model for this problem. The accuracy score comes out to be maximum, viz., 55.5% for k = 5 fold.

Thus, the model is moderately effective due to the following reasons:

The accuracy score is above average.

SIGNIFICANCE OF RESULT:

As shown in Figure (E. data visualisation Figure 11) below, the price category is higher in the later half of the 8-month time period which signifies that "Lower the temperature, higher the price". This implies, that during the winter season, the price is seen to be higher. Thus, the later part of the time period doesn't seem to be a good time for businesses to plan for energy-intensive operations due to high prices.

LIMITATIONS:

- A categorical target variable is provided and not numerical, which means there could be information loss in the process of binning.
- Insufficient data, i.e., 8-month period

Data visualisation

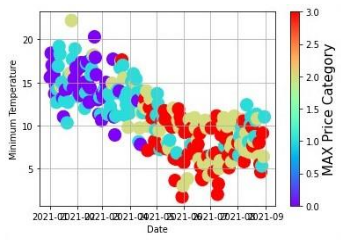
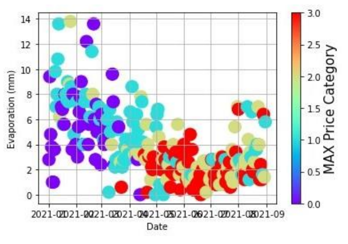
	
The graph shows the price category was higher in the later period due to the drop in minimum temperature.	The graph shows the price category was higher in the later period due to the drop in evaporation.

Figure 11

We are using root mean squared error to calculate the efficiency of our model.

LIMITATION

One of the limitations in the data provided is per day data and timestamp data which led to information loss while cleaning and wrangling so that we could merge the datasets. There is a loss of insights in different time measurements after merging the two sets of datasets.

IMPROVEMENT FOR FUTURE

- The price category is higher in the latter half of the 8-month time period which signifies that the “Lower the temperature, higher the price”. This implies, that during the winter season, the price is seen to be higher. Thus, the later part of the time period doesn’t seem to be a good time for businesses to plan for energy-intensive operations due to high prices. Businesses should plan energy for business operations in the initial or former part of the year.
- This flaw can be improved by having a timely basis of 30 minutes in the weather dataset. So, the datasets can be merged without data loss.
- Perhaps, a more accurate model can predict the change in weather in the morning, afternoon, evening and night from time to time with the energy consumption and price change than a daily basis model.
- More data can be provided instead of just 8 months
- From the price and energy demand models for the Victoria region, we perform predictable energy patterns throughout the first eight months of 2021. We can see the max price category is largely distributed more in the second half of the 8-month period in the red colour cluster. Perhaps, the high price category (red clusters) is because of the lockdown period (June - Aug) of the Victoria region. There were more heaters being used at home during the winter period. The company can consider increasing the price for the first half year during the summer.