Database Lab Assignment Task 3

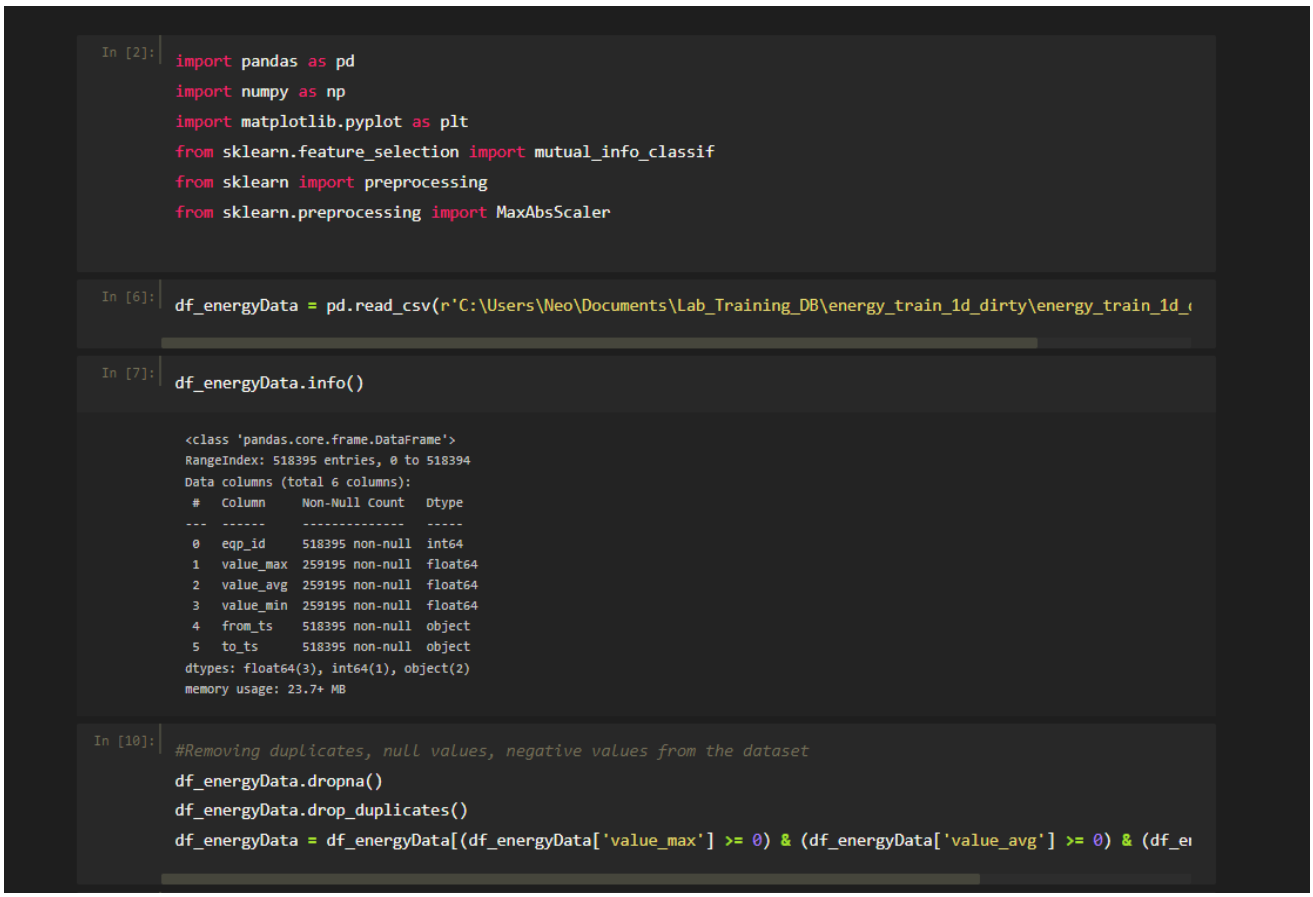
Group 2

Members: Sagnik Sarkar, Muhammad Usman, Adeel Ahmed

Step 1 - Feature Selection

The dataset is read using the pandas library. We used mutual_info_classif of the scikit learn library to identify the efficient feature among the three feature vectors. Feature vectors with higher entropy leads to good prediction accuracy. In our dataset, **value_min** has the higher entropy among the other feature vectors. A graphical representation of this is added at the end of the code.

Also, highly correlated data is identified for the regression analysis using the Correlation function between multiple variables in the given dataset.



In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_selection import mutual_info_classif
from sklearn import preprocessing
from sklearn.preprocessing import MaxAbsScaler
```

In [6]:
```
df_energyData = pd.read_csv(r'C:\Users\Neo\Documents\Lab_Training_DB\energy_train_1d_dirty\energy_train_1d_
```

In [7]:
```
df_energyData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 518395 entries, 0 to 518394
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   eqp_id     518395 non-null  int64  
 1   value_max  259195 non-null  float64
 2   value_avg  259195 non-null  float64
 3   value_min  259195 non-null  float64
 4   from_ts    518395 non-null  object 
 5   to_ts      518395 non-null  object 
dtypes: float64(3), int64(1), object(2)
memory usage: 23.7+ MB
```

In [10]:
```
#Removing duplicates, null values, negative values from the dataset
df_energyData.dropna()
df_energyData.drop_duplicates()
df_energyData = df_energyData[(df_energyData['value_max'] >= 0) & (df_energyData['value_avg'] >= 0) & (df_e
```

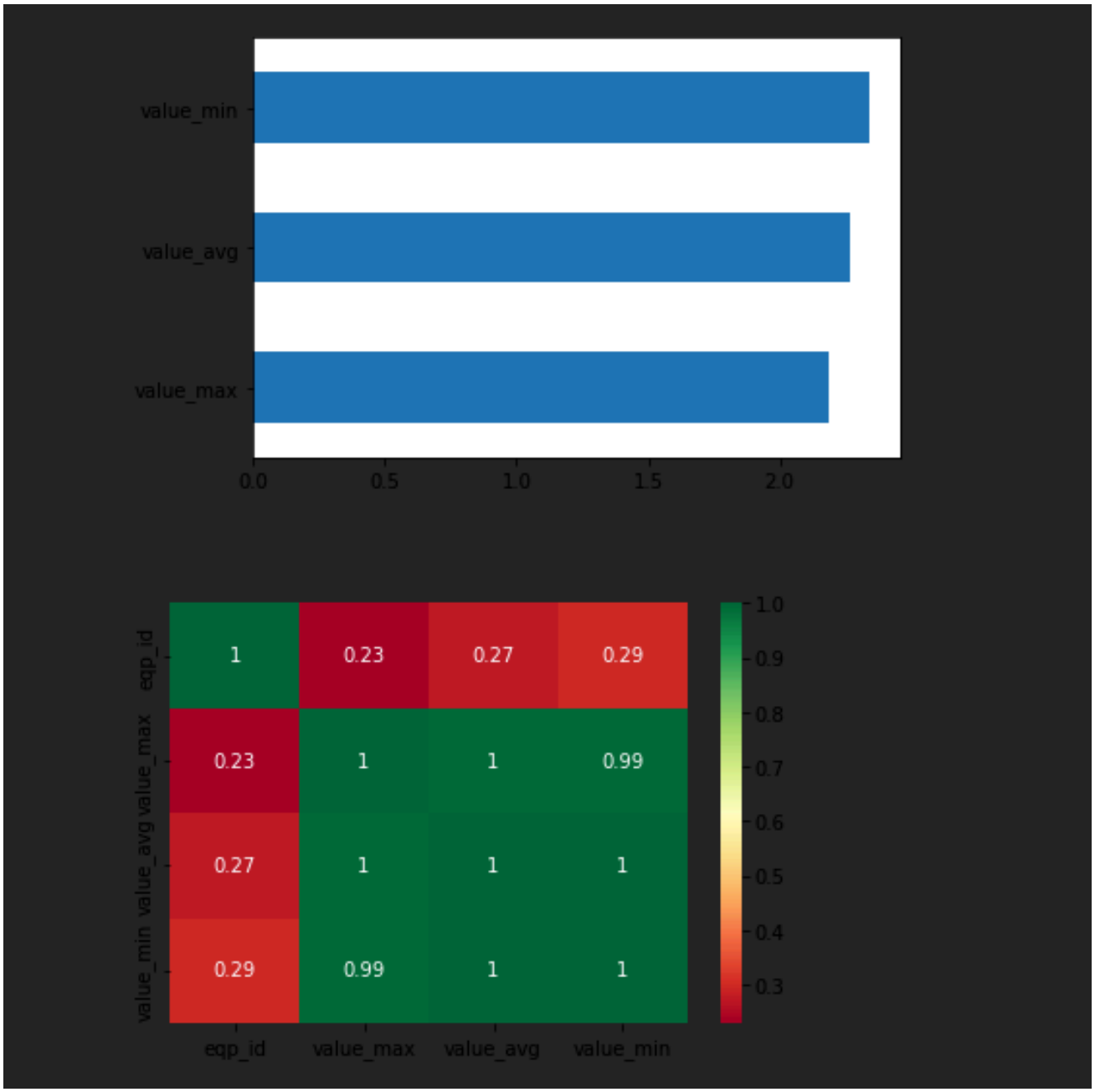
In [11]:
```
df_energyData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 259185 entries, 0 to 501194
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   eqp_id     259185 non-null  int64  
 1   value_max  259185 non-null  float64
 2   value_avg  259185 non-null  float64
 3   value_min  259185 non-null  float64
 4   from_ts    259185 non-null  object 
 5   to_ts      259185 non-null  object 
dtypes: float64(3), int64(1), object(2)
memory usage: 13.8+ MB
```

In [13]:
```
df_energyData.to_csv('newEnergyData.csv', encoding='utf-8', index=False)
df_energyDataNew = pd.read_csv('newEnergyData.csv');
```

In [15]:
```
#Feature Selection using entropy and heatmap
import seaborn as sb

X = df_energyDataNew[["value_max", "value_avg", "value_min"]]
Y = df_energyDataNew["eqp_id"]
correl = df_energyDataNew.corr()
top_feature = correl.index

importances = mutual_info_classif(X, Y)

important_feature = pd.Series(importances, X.columns)
important_feature.plot(kind = 'barh')
plt.show()

heatmap = sb.heatmap(df_energyDataNew[top_feature].corr(), annot=True, cmap = "RdYlGn")
```

Below is the entropy and heatmap results of the dataset.



Step 2 - Data Preprocessing

Data scaling is done using the min-max approach. This method will normalize the data in the range of 0-1.

```
In [16]: #Data Scaling

         scaled_abs = MaxAbsScaler()
         scaled_abs.fit(X)
         scaled_abs.max_abs_
         scaled_data = scaled_abs.transform(X)
```

Step 3 - Time Series Modelling

Time Series modelling of the data is done using the keras library and sequence length parameter is set to be 10.

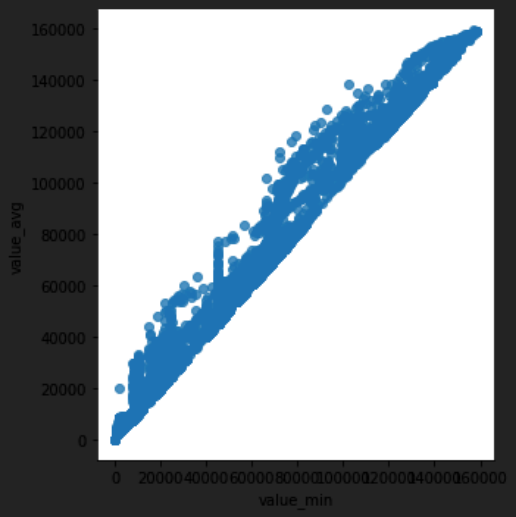
```
In [18]: #Time Series Modelling

         from tensorflow import keras

         dataset = keras.preprocessing.timeseries_dataset_from_array(scaled_data, Y, sequence_length = 10)
```

Step 4 - Model Design

Linear Regression model is used for the design. From the heatmap generated in Step 1, we can identify that **value_avg** has high positive correlation with **value_min.** X value of the graph is substituted as value_min and Y value of the graph is substituted as value_avg. Using seaborn, we can see the correlation between these two variables.

```
In [16]: import seaborn as sb

         sb.lmplot(x = 'value_min', y = 'value_avg', data = df_energyDataNew)
```

```
<seaborn.axisgrid.FacetGrid at 0x1c92a071ca0>
```



Step 5 - Model Training

To start the model training of the data, the dataset is divided into 4 parts. The sklearn provides a function called train_test_split function which does the fragmentation of the data. Random state parameter of value 10 and test size parameter of value 0.3 is passed as arguments to the function.

```
In [20]: X = df_energyDataNew[['value_min']]
         y = df_energyDataNew[['value_avg']]


         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10)
```

The LinearRegression() model function of the sklearn is used to perform the Training of the dataset. The prediction came with accuracy of 0.99 for the test data set. The first 10 values of the actual and predicted ‘y’ values are given below.

```
In [21]: from sklearn.linear_model import LinearRegression
         reg_val = LinearRegression()
         reg_val.fit(X_train, y_train)

         ▸ LinearRegression

In [22]: reg_val.score(X_test, y_test)

         0.9976173421885725

In [24]: y_pred = reg_val.predict(X_test)

         evaluate = pd.DataFrame({'Actual': y_test.values.flatten(), 'Predicted': y_pred.flatten()})
         evaluate.head(10)
```

	Actual	Predicted
0	72469.257812	72683.992765
1	18268.564453	19490.616512
2	71314.585938	71594.351243
3	71978.890625	72225.350335
4	0.000000	1515.468024
5	5482.250977	4991.366016
6	6297.559082	6113.691124
7	4579.419922	3942.725277
8	4563.474121	3500.040241
9	7163.156250	3631.810596

Step 6 - Result Visualisation

The first 20 comparisons of the actual and predicted values are plotted in the bar plot below.

```
In [25]: evaluate.head(20).plot(kind = 'bar')

         <AxesSubplot:>
```