Internet Of Things Final Project

Name:	**Dinesh Adhithya**
Registration No./Roll No.:	18097
Institute/University Name:	IISER Bhopal
Program/Stream:	EECS

1 Introduction

We need to import necessary modules . import the data set and then convert to the datatype of timestamp column to datetime data type so that it can be used to future processing and filtering tasks.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv("weather_data_2sites.csv")

sum=15 # last 3 digits of roll no. sum
data["timestamp"]=pd.to_datetime(data["timestamp"])
```

Listing 1: Importing Data set

The necessary columns such as temperaturesite1 and humiditysite1 were imported and 5 dates from 15th to 19th of march were selected for plotting.

```
data_plot=data[np.logical_and(data["timestamp"]<"2018-03-20 00:00:00", "2018-03-15
    00:00:00"<=data["timestamp"])][["timestamp","temperature_site1","humidity_site1"
    ]]
```

Listing 2: Filtering Data set

2 Plotting: Visualization of Data

Preprocessing of data to make it suitable for plotting:

```
time=[]
date=[]
temp=[]
hum=[]
for i in range(data_plot.shape[0]):
  td=str(data_plot.iloc[i]["timestamp"]).split()
  date.append(td[0])
  time.append(td[1])
  temp.append(data_plot.iloc[i]["temperature_site1"])
  hum.append(data_plot.iloc[i]["humidity_site1"])

data_new=pd.DataFrame({"TIME":time,"DATE":date,"TEMPERATURE":temp,"HUMIDITY":hum})

dates=np.unique(data_new["DATE"])
```

Listing 3: Preprocessing Data set

2.1 Task1

Line plots for temperature data showing 5 lines in each plot corresponding to 5 days of data. Time on the X-axis should show day hour. Plot legend must show the day number (i.e., month day) used for the plotting.

```python

plt.figure(figsize=(50,10))
for i in range(len(dates)):
  temp=data_new[data_new["DATE"]==dates[i]]
  plt.plot(temp["TIME"],temp["TEMPERATURE"],label=dates[i])
plt.title("Temperature vs Time")
plt.xlabel("Time")
plt.ylabel("Temperature")
plt.legend()
```

Listing 4: Task 1 plots.

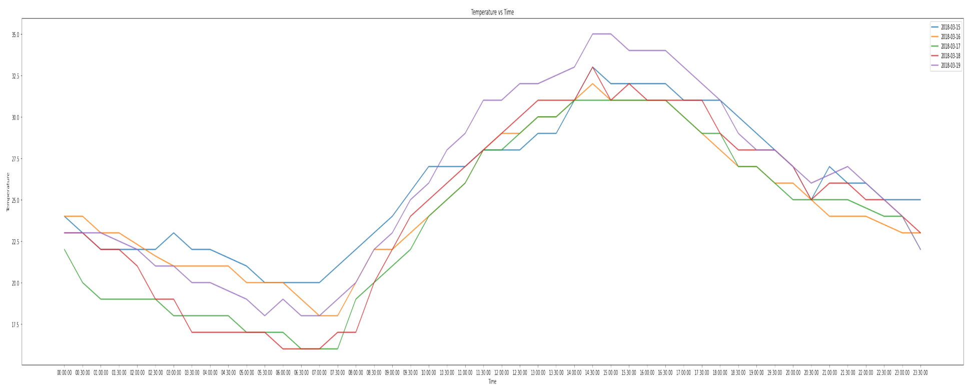


Figure 1: Plot showing Temperature vs Time plot for 5 days in march with a a legend of each day shown with each line plot corresponding to date in different color.

With temperature falling during the night , eventually peaking and again falling down over the course of the day.

2.2 Task2

Line plots for humidity data showing 5 lines in each plot corresponding to 5 days of data. Time on the X-axis should show day hour. Plot legend must show the day number (i.e., month day) used for the plotting.

```python

plt.figure(figsize=(50,10))
for i in range(len(dates)):
  temp=data_new[data_new["DATE"]==dates[i]]
  plt.plot(temp["TIME"],temp["HUMIDITY"],label=dates[i])
plt.title("Humidity vs Time")
plt.xlabel("Time")
plt.ylabel("Humidity")
plt.legend()
```

Listing 5: Task 2 plots.

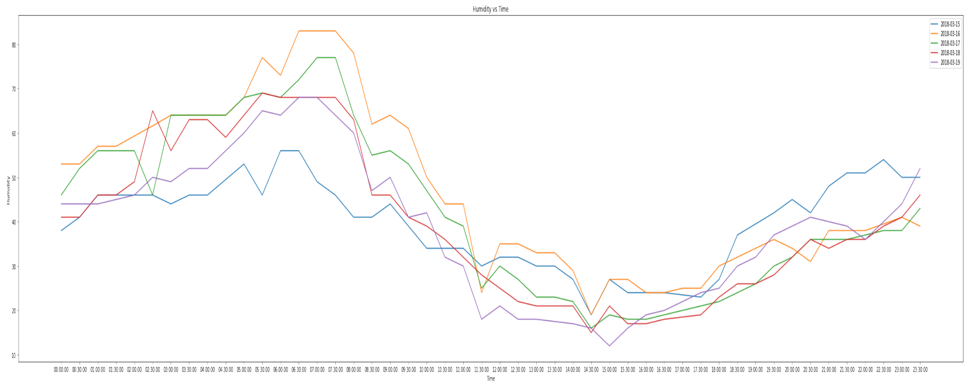


Figure 2: Plot showing Humidity vs Time plot for 5 days in march with a legend of each day shown with each line plot corresponding to date in different color.

With humidity rising during the night , eventually peaking and again falling down over the course of the day and rising again as it reaches night.

2.3 Task3

Box plots on temperature data such that each box corresponds to one day's data. X axis labels should reflect the days used for the plotting.

```python

plt.figure(figsize=(50,10))
temp=[]
for i in range(len(dates)):
  temp.append(data_new[data_new["DATE"]==dates[i]]["TEMPERATURE"])
plt.boxplot(temp)
plt.title("Temperature vs Time")
plt.xlabel("Time")
plt.ylabel("Temperature")
plt.xticks([1, 2, 3,4,5],dates)
plt.legend()
plt.show()
```

Listing 6: Task 3 plots.

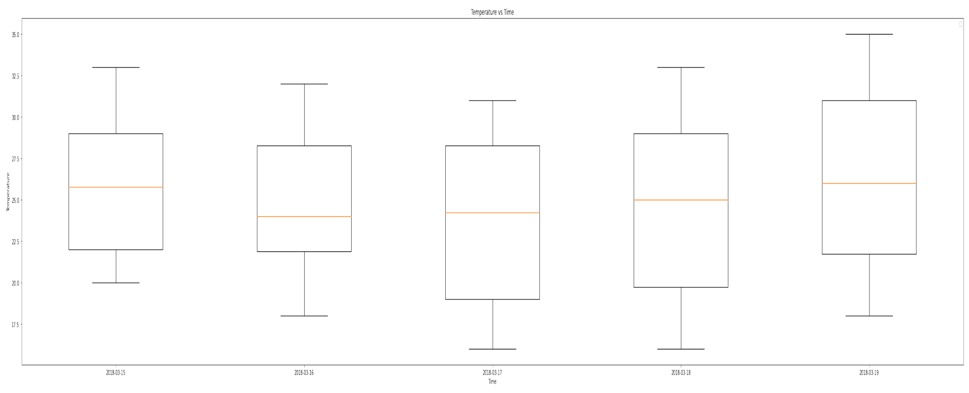


Figure 3: Boxplot showing Temperature for 5 days in march whose median is depicted with a orange line and first the box encompassing values within .25 up and down of median. With the small horizontal lines indicating end points.

2.4 Task4

Box plots on humidity data such that each box corresponds to one day's data. X axis labels should reflect the days used for the plotting.

```python

plt.figure(figsize=(50,10))
temp=[]
for i in range(len(dates)):
  temp.append(data_new[data_new["DATE"]==dates[i]]["HUMIDITY"])

plt.boxplot(temp)
plt.title("Humidity vs Time")
plt.xlabel("Time")
plt.ylabel("Humidity")
plt.xticks([1, 2, 3,4,5],dates)
plt.legend()
plt.show()
```

Listing 7: Task 4 plots.

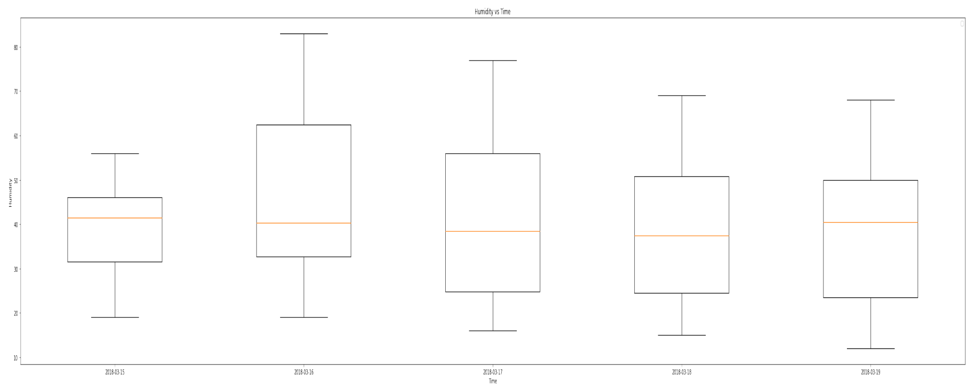


Figure 4: Boxplot showing Humidity for 5 days in march whose median is depicted with a orange line and first the box encompassing values within .25 up and down of median. With the small horizontal lines indicating end points

2.5 Task5

Line grid plot for any 3 days of data where labels on the plot are temperature day[x], humidity day[y]. Axis labels must show which days data you have used for plotting.

```python
df=pd.DataFrame()
for i in range(len(dates)):
  temp=data_new[data_new["DATE"]==dates[i]]
  df["Temperature day "+str(i+1)]=list(temp["TEMPERATURE"])
  df["Humility day "+str(i+1)]=list(temp["HUMIDITY"])


sns.pairplot(data=df)
```

Listing 8: Task 5 plots.

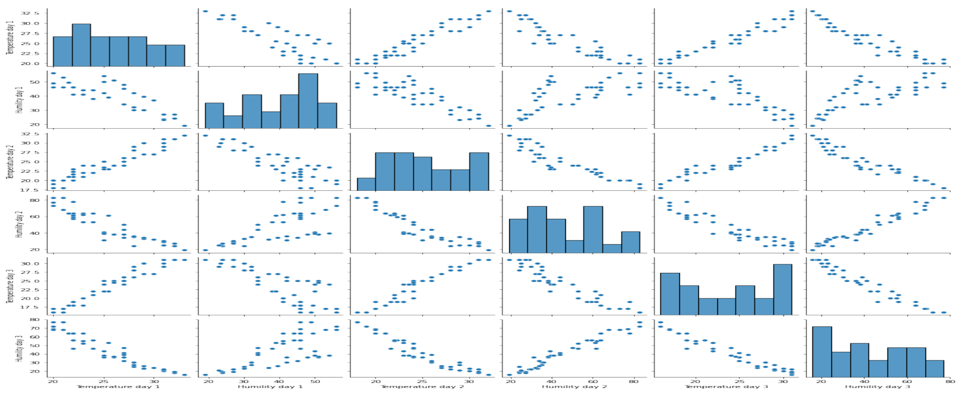


Figure 5: Grid plot showing Humidity and Temperature for 3 days in march

Along the diagonals we see histogram for various columns in the data frame and other elements we see scatter plot of 2 variables decided by the corresponding row and column name.

3 Analysis

Use Weatherdata2sites data set to create a new data set, Synweather, consisting of 5 variables, i.e, temperature, humidity, day-minutes, day-of-the-week, previous-temperature. Such that the dimensions of the Synweather will be 5779 x 5. Use the Sysweather data set and perform the following tasks

We preprocess the data and then make synweather data frame , latter necessary machine learning modules for further tasks.The day of the week has been encoded by a number as machine learning can be performed while it is in text form.

```python
Syn_weather=pd.DataFrame()
DATA=data.drop(["Unnamed: 0"],axis=1)
time=[]
for i in range(5780):
  time.append(DATA["timestamp"].iloc[i].weekday())
minute=[]
for i in range(5780):
  minute.append((DATA["timestamp"].iloc[i].minute)+DATA["timestamp"].iloc[i].hour*60)
temp_previous=np.array(DATA.iloc[:5779][["temperature_site1"]])
Syn_weather[["temperature_site","humidity_site"]]=DATA.iloc[1:][["temperature_site1",
    "humidity_site1"]]
Syn_weather["previous_temperature"]=temp_previous
Syn_weather["week_day"]=time[1:]
Syn_weather["minutes_of_day"]=minute[1:]
```

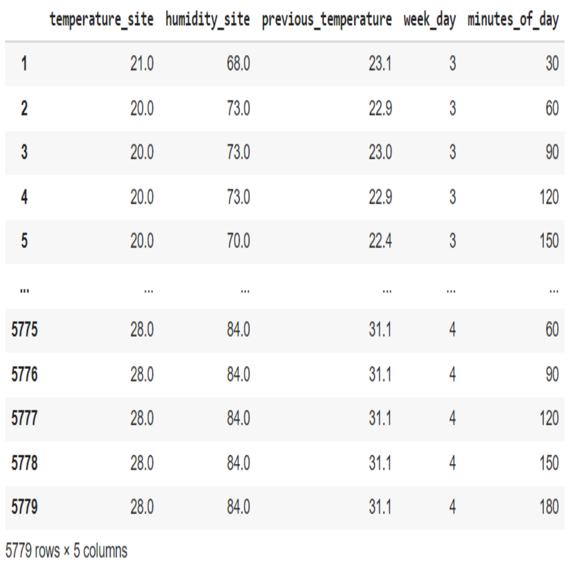
Listing 9: Processing given data set to make synweather data frame.

```python
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error , r2_score
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

Listing 10: Importing modules for machine learning tasks.

3.1 Task1

A regression model, M1 named as lrmodel, predicting temperature using variables humidity, day-minutes, day-of-the-week, previous-temperature. Use 2/3 of data without replacement for training and rest for 1/3 for testing.



	temperature_site	humidity_site	previous_temperature	week_day	minutes_of_day
1	21.0	68.0	23.1	3	30
2	20.0	73.0	22.9	3	60
3	20.0	73.0	23.0	3	90
4	20.0	73.0	22.9	3	120
5	20.0	70.0	22.4	3	150
...	...	...	...	...	...
5775	28.0	84.0	31.1	4	60
5776	28.0	84.0	31.1	4	90
5777	28.0	84.0	31.1	4	120
5778	28.0	84.0	31.1	4	150
5779	28.0	84.0	31.1	4	180

5779 rows × 5 columns

Figure 6: Processed Data Frame Synweather ready for analysis

```
x_train,x_test,y_train,y_test=train_test_split(Syn_weather[['humidity_site', '
    previous_temperature','week_day','minutes_of_day']],Syn_weather['temperature_site
    '],train_size=0.66)

lr_model=LinearRegression().fit(np.array(x_train),np.array(y_train))
mean_squared_error(lr_model.predict(np.array(x_test)),np.array(y_test))**0.5
```

Listing 11: Task 1 plots.

The root mean squared error was found out to be 0.5190507334047213

3.2 Task2

A regression model, M2, predicting temperature using variables humidity, day-minutes, day-of-the-week, previous-temperature. 10 fold Cross-validation was used as shown below.

```
X=np.array(Syn_weather[['humidity_site', 'previous_temperature', 'week_day','
    minutes_of_day']])
Y=np.array(Syn_weather['temperature_site'])

reg_mod2 = LinearRegression()
cv = KFold(n_splits=10, random_state=1, shuffle=False)
scores = cross_val_score(reg_mod2, X, Y,scoring="neg_root_mean_squared_error" ,cv=cv,
     n_jobs=-1)

scores*-1

np.mean(scores*-1)
```

Listing 12: KFold splitting of data and checking their performance.

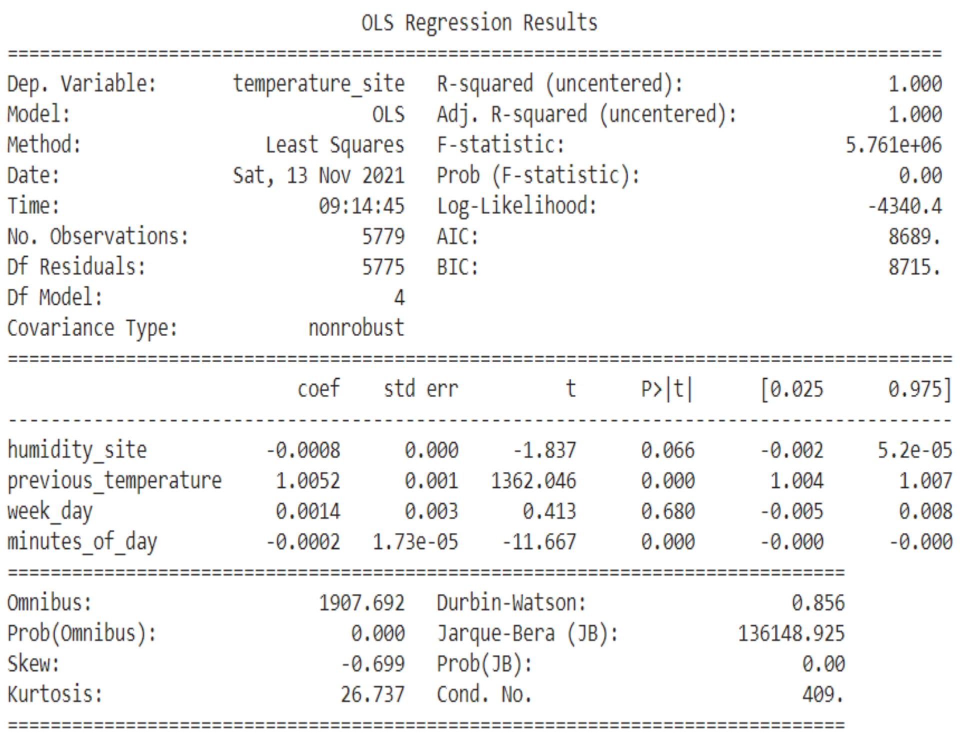
[0.59095253, 0.48902806, 0.49232317, 0.52696451, 0.45733666, 0.48618446, 0.62988745, 0.45959622, 0.39534381, 0.54279907] was the RMSE for each fold and their mean value was 0.5070415935013529.

3.3 Task 3

Using step-wise regression we find the top-2 predictors in M1.

```
import statsmodels.api as sm
results = sm.OLS(Syn_weather["temperature_site"], Syn_weather[['humidity_site', '
    previous_temperature', 'week_day','minutes_of_day']]).fit()
print(results.summary())
```

Listing 13: Performing stepwise regression on dataset.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:      temperature_site   R-squared (uncentered):          1.000
Model:                           OLS   Adj. R-squared (uncentered):     1.000
Method:                Least Squares   F-statistic:                 5.761e+06
Date:               Sat, 13 Nov 2021   Prob (F-statistic):               0.00
Time:                       09:14:45   Log-Likelihood:                -4340.4
No. Observations:               5779   AIC:                              8689.
Df Residuals:                   5775   BIC:                              8715.
Df Model:                          4
Covariance Type:           nonrobust
==============================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
humidity_site         -0.0008      0.000     -1.837      0.066      -0.002    5.2e-05
previous_temperature   1.0052      0.001   1362.046      0.000       1.004      1.007
week_day               0.0014      0.003      0.413      0.680      -0.005      0.008
minutes_of_day        -0.0002   1.73e-05    -11.667      0.000      -0.000     -0.000
==============================================================================
Omnibus:                    1907.692   Durbin-Watson:                   0.856
Prob(Omnibus):                 0.000   Jarque-Bera (JB):           136148.925
Skew:                         -0.699   Prob(JB):                         0.00
Kurtosis:                     26.737   Cond. No.                         409.
==============================================================================
```

Figure 7: Step-wise regression for temperature prediction summary.

From the p values we can see that previous temperature and minutes of the week are most important parameters.The humidity also some importance , but the day of the week is the least important parameter from step wise regression results.

3.4 Task 4

correlation coefficient of all variables in Synweather were computed below and plotted as heat map.

```
sns.heatmap(Syn_weather.corr(),annot=True)
plt.show()
```

Listing 14: Code for Plotting heatmap of correlation values between model parameters.

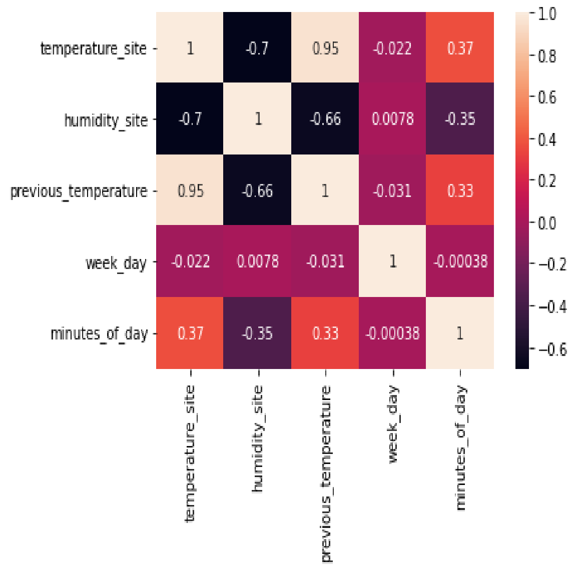


Figure 8: Correlation heatmap among various model paramters and temperature.

Humidity shows negative correlation with temperature increase humidity decreases in that region. Whereas previous temperature and minutes of the day show positive correlation , that means with the increase of these quantities the temperature increases. Week day barely shows any correlation as it's almost zero , meaning it shows no correlation with temperature.

3.5 Task 5

Plot actual vs predicted values for both M1 and M2 aree shown below.

```python
plt.scatter(y_test,lr_model.predict(x_test))
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("M1 regression model")
plt.show()
```

Listing 15: Task 2 Codes.

the scatter plot being mostly on the y=x curve shows that the model predicts very well.

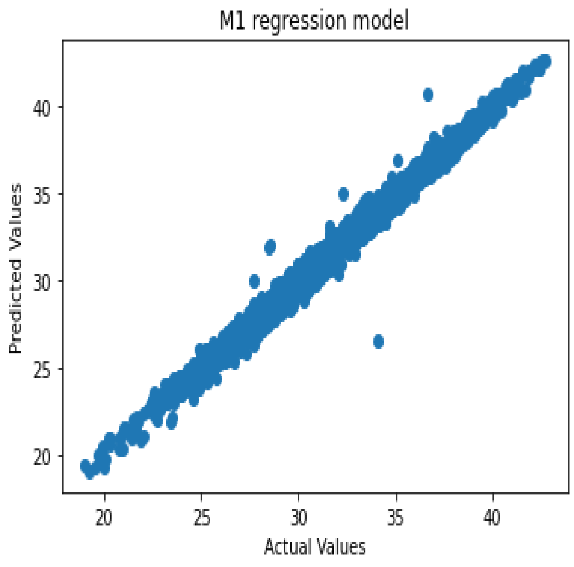


Figure 9: Processed Data Frame Synweather ready for analysis

```python
i=1
plt.figure(figsize=(20,10))
plt.suptitle("M2 Regression Actual vs Predicted values")
features=['humidity_site', 'previous_temperature', 'week_day',
       'minutes_of_day']
for train_index, test_index in cv.split(X):
  X_train = Syn_weather.iloc[train_index].loc[:, features]
  X_test = Syn_weather.iloc[test_index][features]
  y_train = Syn_weather.iloc[train_index].loc[:,'temperature_site']
  y_test = Syn_weather.iloc[test_index]['temperature_site']


  M2=LinearRegression()
  M2.fit(X_train,y_train)

  plt.subplot(2,5,i)

  plt.scatter(y_test,M2.predict(X_test))
  #plt.plot([0,0],[45,45])
  plt.xlabel("Actual Values")
  plt.ylabel("Predicted Values")
  plt.title("KFold split no."+str(i))

```

```
    plt.plot([20,40],[20,40],color="orange")

    i+=1
plt.show()
```

Listing 16: KFold scatter plots for 10 splits.

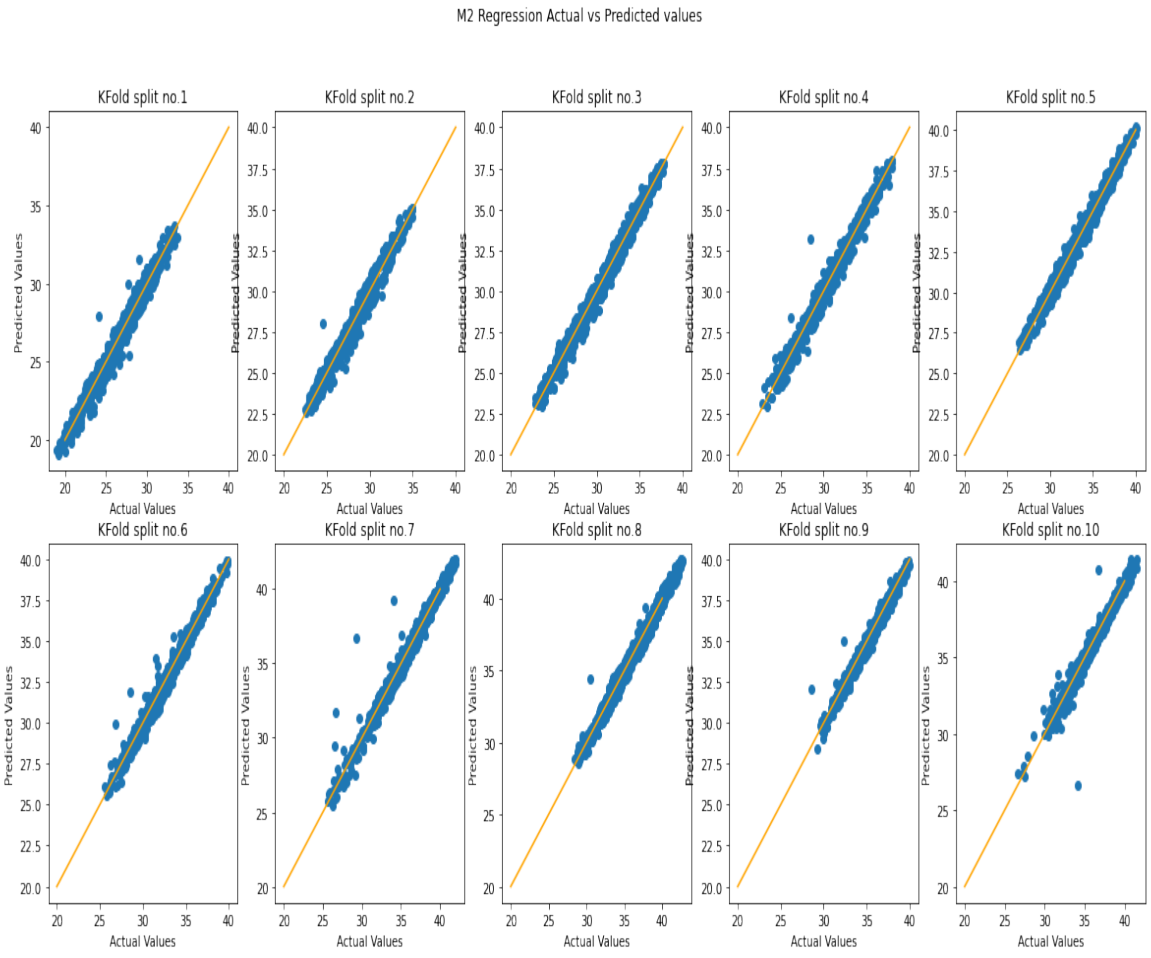


Figure 10: M2 regression KFold plots

From the plots we can see that while performing Kfold analysis of the data set , there is bias towards values that occur often in the model Hence the edge case points aren't well predicted and we see them as outliers. Otherwise our model performs very well , with most points along x=y curve.

This plots clearly show that the model performs well to predict the next temperature with the given parameters as input.