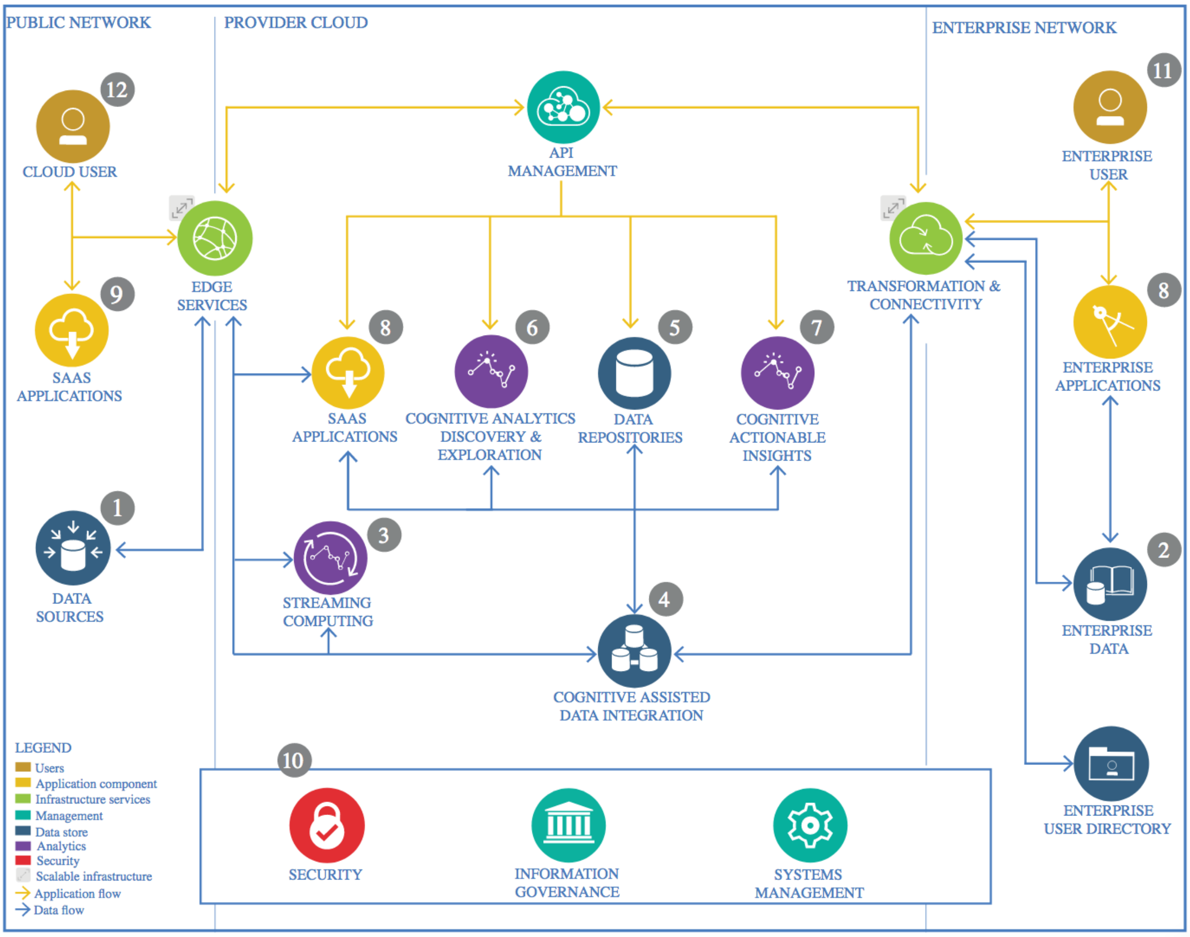
Architecture Design Document – Soil Moisture Prediction

- Sivasakthi Jayashankar
- May 2019

1 Architectural Components Overview

Overview of different stages provided by IBM is used as a guideline



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

1.1 Objective –

Predict soil moisture using

1.2 Data Source

1.2.1 Technology Choice

Data was obtained from the Library of United States Department of Agriculture

- Dataset includes hourly hydro-meteorological variables including soil moisture, air temperature and relative humidity from 11 sites in Reynolds Creek in southwestern Idaho

Two Datasets were retrieved.

1. Weather Dataset
2. Soil Moisture Dataset

```python
weatherurl="https://data.nal.usda.gov/system/files/weather_data_jdt2b.csv"
s=requests.get(weatherurl).content
weatherData=pd.read_csv(io.StringIO(s.decode('utf-8')))
```

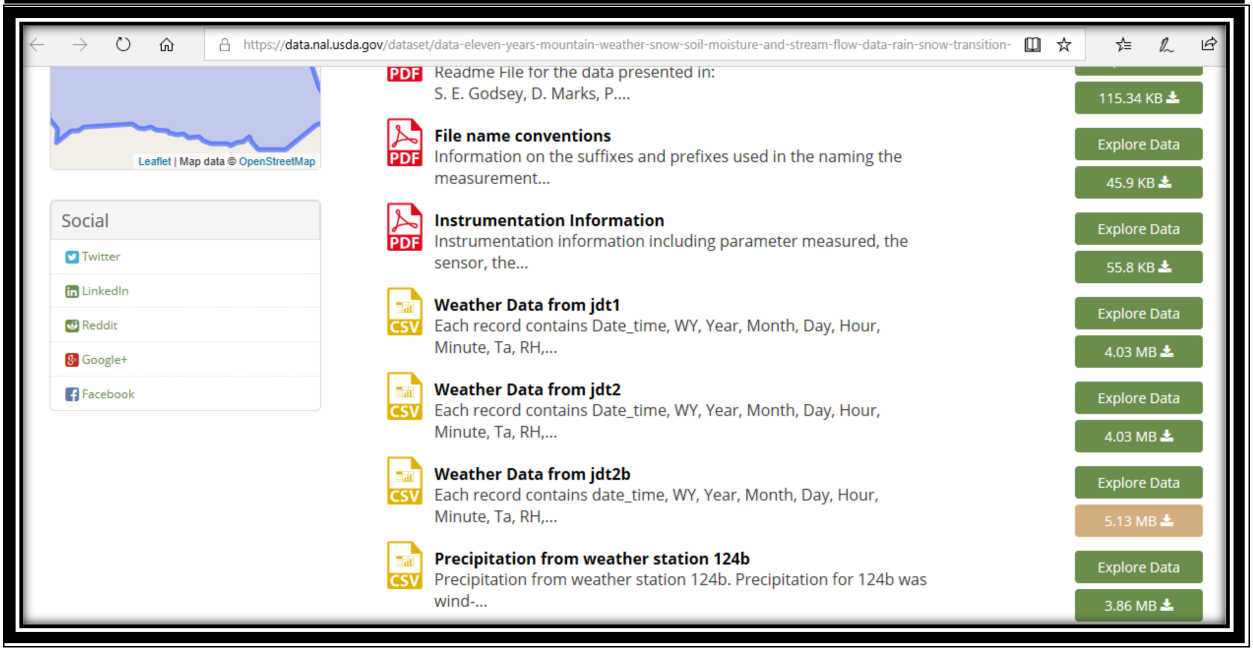


Figure 1 – Download Weather Dataset from Source

```python
soilmoistureurl="https://data.nal.usda.gov/system/files/rc.tg_.dc_.jd-jdt2b_stm_0.csv"
s=requests.get(soilmoistureurl).content
soilmoistureData=pd.read_csv(io.StringIO(s.decode('utf-8')))
```

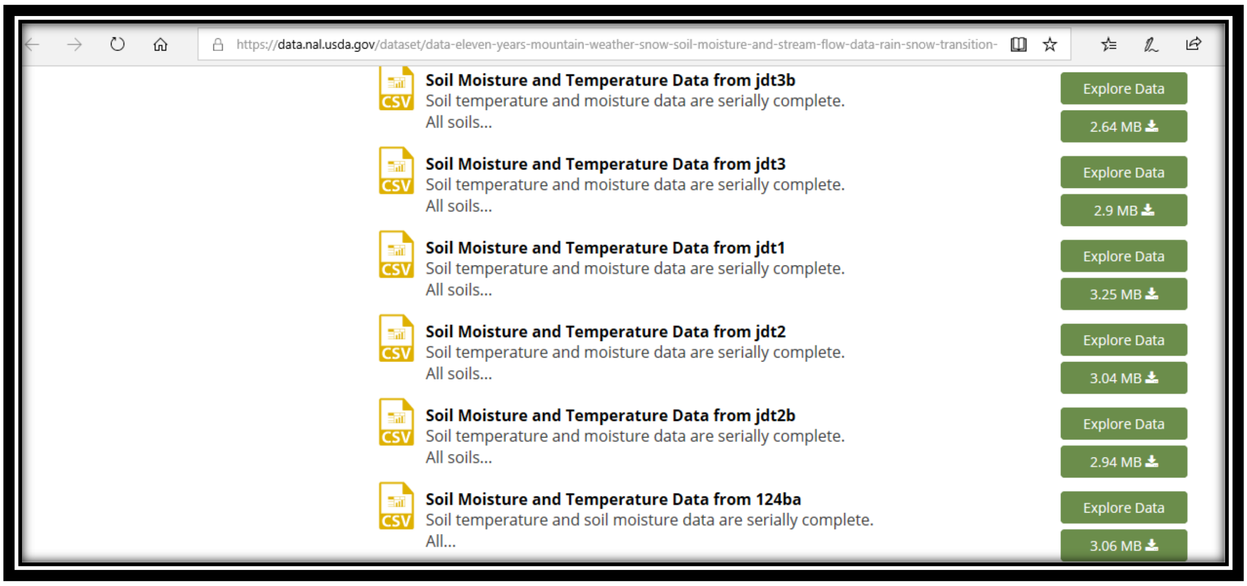


Figure 2 –Download Soil Moisture Dataset from source

1.2.2 Justification

Dataset was downloaded as CSV from the USDA repository

1.3 Enterprise Data

1.3.1 Technology Choice

N/A

1.3.2 Justification

Data obtained is from public data source

1.4 Streaming analytics

1.4.1 Technology Choice

Real-time data processing is not applicable for the current application

1.4.2 Justification

Use case is to analyze finite data set.

1.5 Data Integration

1.5.1 Technology Choice

Apache Spark, Python, Jupyter Notebook

1. Weather Data

```
weatherData.head()
```

	Date_time	WY	Year	Month	Day	Hour	Minute	T_a	RH	e_a	T_d	w_s	w_d
0	11/5/2005 0:00	2006	2005	11	5	0	0	-9999.0	-9999.0	-9999	-9999.0	-9999.0	-9999.0
1	11/5/2005 1:00	2006	2005	11	5	1	0	-9999.0	-9999.0	-9999	-9999.0	-9999.0	-9999.0
2	11/5/2005 2:00	2006	2005	11	5	2	0	-9999.0	-9999.0	-9999	-9999.0	-9999.0	-9999.0
3	11/5/2005 3:00	2006	2005	11	5	3	0	-9999.0	-9999.0	-9999	-9999.0	-9999.0	-9999.0
4	11/5/2005 4:00	2006	2005	11	5	4	0	-9999.0	-9999.0	-9999	-9999.0	-9999.0	-9999.0

Figure 3 – Glimpse of Weather Dataset

Data Dictionary of Weather Dataset

Feature Name	Feature Description	Feature Name	Feature Description
Date_time	Date followed by Time	Hour	Hour of Day
WY	Water Year	T_a	Air Temperature(Degrees)
Year	Calendar Year	RH	Relative Humidity
Month	Month of Year	e_a	Water Vapor Pressure
Day	Day of Month	w_s	Wind Speed
T_d	Dew Point Temperature(Degrees)	w_d	Wind Direction

2. Soil Moisture Data

```
soilmoistureData.head()
```

	Date_time	WY	Year	Month	Day	Hour	Minute	T_g_5	T_g_20	T_g_35	T_g_50	T_g_75	s_m_5	s_m_20	s_m_35	s_m_50	s_m_75
0	10/1/2010 0:00	2011	2010	10	1	0	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
1	10/1/2010 1:00	2011	2010	10	1	1	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
2	10/1/2010 2:00	2011	2010	10	1	2	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
3	10/1/2010 3:00	2011	2010	10	1	3	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0
4	10/1/2010 4:00	2011	2010	10	1	4	0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0	-9999.0

Figure 4 – Glimpse of Soil Moisture Data

Data Dictionary of Soil Moisture Dataset

Feature Name	Feature Description	Feature Name	Feature Description
Date_time	Date followed by Time	WY	Water Year
Year	Calendar Year	T_g_50	Soil temperature at 50 cm Depth
Month	Month of Year	T_g_75	Soil temperature at 75 cm Depth

Day	Day of Month	s_m_5	Soil moisture at 5 cm Depth
Hour	Hour of Day	s_m_20	Soil moisture at 20 cm Depth
T_g_5	Soil temperature at 5 cm Depth	s_m_35	Soil moisture at 35 cm Depth
T_g_20	Soil temperature at 20 cm Depth	s_m_50	Soil moisture at 50 cm Depth
T_g_35	Soil temperature at 35 cm Depth	s_m_75	Soil moisture at 75 cm Depth

3. Data Cleansing

Data source documentation indicates if there was no data available, it is denoted by 9999 .

Remove all rows where variables are equal to 9999

```
#Data source documentation indicates null values are denoted by 9999
# Lets remove those
weatherClean = weatherData[(weatherData[['T_a','RH','e_a','T_d','w_s','w_d']] != -9999.000000).all(axis=1)]
```

```
#Data source documentation indicates null values are denoted by 9999
# # Lets remove those
soilMoistureClean = soilmoistureData[(soilmoistureData[['T_g_5','T_g_20','T_g_35','T_g_50','T_g_75','s_m_5','s_m_20','s_m_35','s_
```

Figure 5 – Data Cleansing – remove rows with values equal to -9999

Merge on Weather and Soil Moisture Datasets

```
#Merge Data with Date being the common column
soilWeatherData = soilMoistureClean.merge(weatherClean,how="inner", on="Date_time")
```

```
soilWeatherData.head()
```

	Date_time	WY	Year_x	Month_x	Day_x	Hour_x	Minute_x	T_g_5	T_g_20	T_g_35	...	Month_y	Day_y	Hour_y	Minute_y	T_a	RH	e_a	T_d	w_s	v
0	3/4/2011 11:00	2011	2011	3	4	11	0	0.3	1.6	2.1	...	3	4	11	0	2.8	0.61	456	-3.5	3.1	16
1	3/4/2011 12:00	2011	2011	3	4	12	0	1.1	1.6	2.1	...	3	4	12	0	3.4	0.55	429	-4.2	3.2	19
2	3/4/2011 13:00	2011	2011	3	4	13	0	2.4	1.6	2.1	...	3	4	13	0	3.9	0.52	420	-4.5	3.1	13
3	3/4/2011 14:00	2011	2011	3	4	14	0	3.3	1.7	2.0	...	3	4	14	0	3.6	0.56	443	-3.8	4.0	24
4	3/4/2011 15:00	2011	2011	3	4	15	0	4.1	1.9	2.0	...	3	4	15	0	4.8	0.54	465	-3.3	2.7	13

5 rows × 29 columns

Figure 6 – Merge Weather and Soil Datasets

Drop columns not required and are duplicates from weather and soil datasets

```
#Drop columns not needed and are duplicate
soilWeatherData.dtypes
```

```
Date_time     object
WY             int64
Year_x         int64
Month_x        int64
Day_x          int64
Hour_x         int64
Minute_x       int64
T_g_5        float64
T_g_20       float64
T_g_35       float64
T_g_50       float64
T_g_75       float64
s_m_5        float64
s_m_20       float64
s_m_35       float64
s_m_50       float64
s_m_75       float64
 WY            int64
Year_y         int64
Month_y        int64
Day_y          int64
Hour_y         int64
Minute_y       int64
T_a          float64
RH           float64
e_a            int64
T_d          float64
w_s          float64
w_d          float64
dtype: object
```

```
# Date, Time & Hour is not required

soilWeatherConcise = soilWeatherData.filter(['T_g_5','T_g_20','T_g_35','T_g_50','T_g_75','s_m_5','s_m_20','s_m_35','s_m_50','s_m_
```

Figure 7 – Drop Obvious Features not required for prediction

Check on values and understand the data attributes

```
soilWeatherConcise.describe()
```

	T_g_5	T_g_20	T_g_35	T_g_50	T_g_75	s_m_5	s_m_20	s_m_35	s_m_50	s_m_75	
count	27093.000000	27093.000000	27093.000000	27093.000000	27093.000000	27093.000000	27093.000000	27093.000000	27093.000000	27093.000000	27093.00
mean	12.653412	12.525926	12.328476	12.058214	11.765740	0.134306	0.225555	0.210865	0.217434	0.204170	9.38
std	10.120863	8.716143	8.141291	7.606521	6.742225	0.064648	0.072174	0.069939	0.069741	0.071845	9.80
min	-2.000000	0.200000	0.600000	1.300000	2.200000	0.038000	0.132000	0.120000	0.126000	0.121000	-16.10
25%	2.900000	3.900000	4.200000	4.300000	5.000000	0.073000	0.163000	0.151000	0.160000	0.144000	1.90
50%	11.500000	12.000000	11.900000	11.600000	11.200000	0.121000	0.184000	0.175000	0.180000	0.162000	8.10
75%	21.100000	21.100000	20.500000	19.500000	18.000000	0.193000	0.301000	0.292000	0.292000	0.291000	17.40
max	38.200000	29.200000	26.900000	25.100000	23.500000	0.313000	0.382000	0.344000	0.352000	0.336000	36.20

Figure 8 – Understand the values , variance and deviation

1.5.2 Justification

In this stage, data is cleansed, transformed,

1.6 Data Repository

1.6.1 Technology Choice

Object storage / File system

1.6.2 Justification

Object storage allows to store unlimited amount of Data. Also suited for archival when size of data becomes an issue

1.7 Discovery and Exploration

1.7.1 Technology Choice

Apache Spark ,Jupyter, Python 3.6, Matplotlib, Seaborn libraries suffice our needs for data discovery and exploration.

Visualization helps discover the relationship between the attributes.

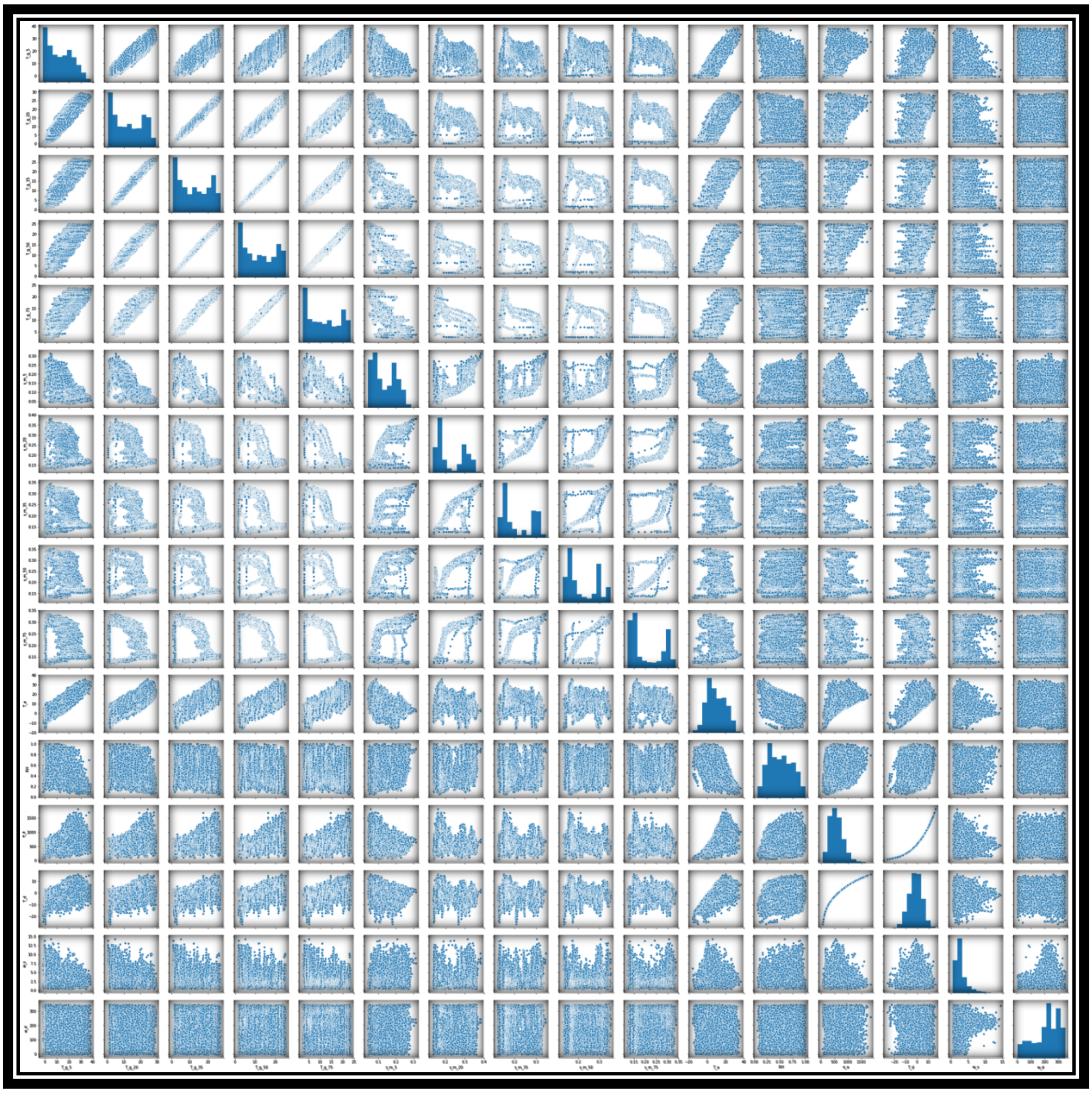


Figure 9 – Feature Relationship study

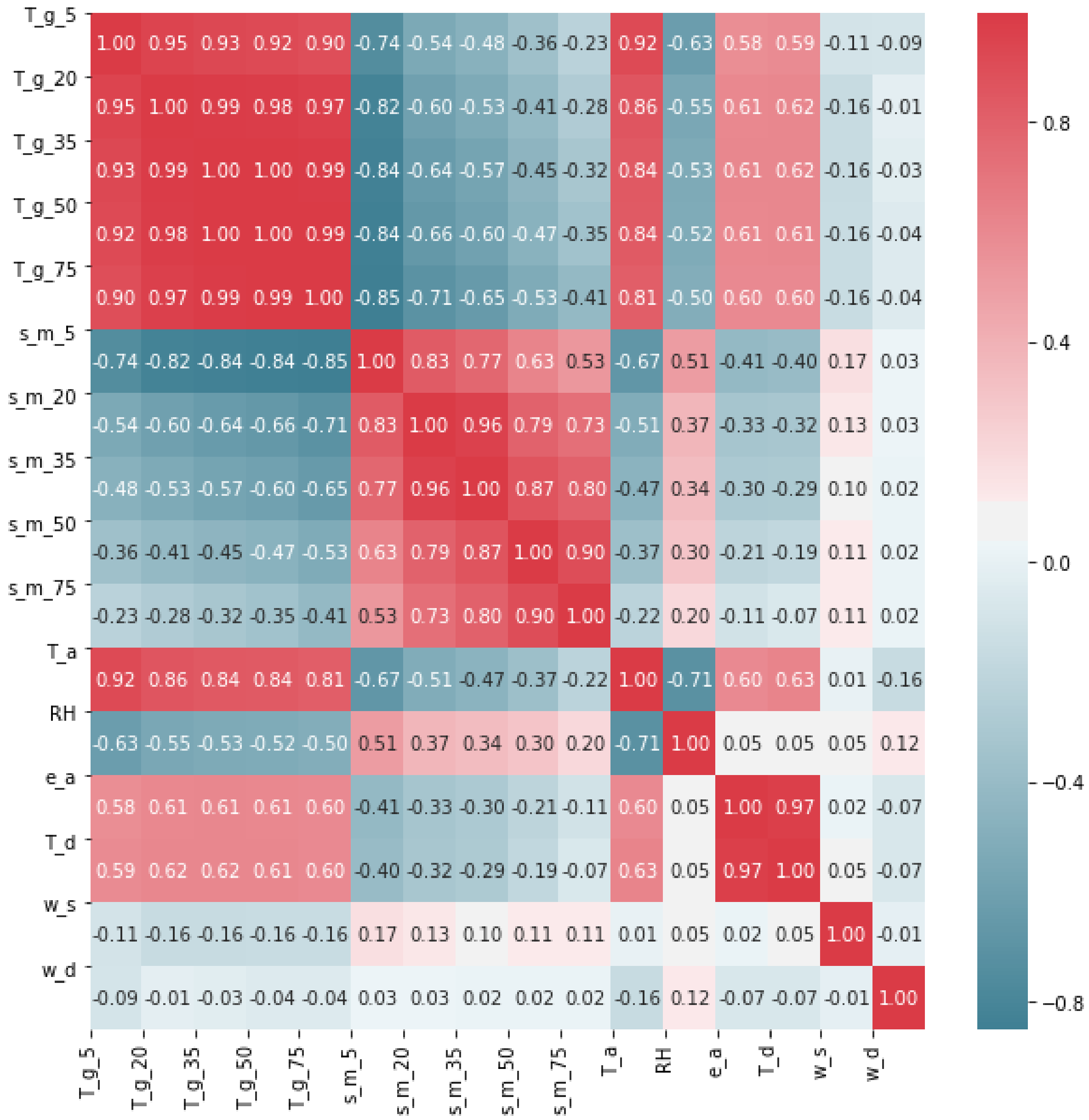


Figure 10 – Feature relation study - Correlation

1.7.2 Justification

Jupyter, Python, Pandas, Seaborn & Matplotlib libraries are all open source and supported in all platforms.

1.8 Actionable Insights

1.8.1 Technology Choice

Support Vector Regression and Neural Network are the two ML models evaluated.

Support Vector Regression - Python scikit-learn library is used to build SVR model. There are different parameters in SVR. This project will compare the results between different kernels, C values, epsilon values.
Neural Networks - Sequential Keras Model with multiple hidden layers applied using Adam optimizer and relu activation.

1.8.2 Justification

Exploration and Visualization of data shows data is not linear. SVR help transform to a higher dimensional data so the points can be linear separable. The options of kernels in SVR are linear, poly and RBF. Choosing the right kernel and the epsilon is very important. R square metric score can be dramatically affect with different choices of kernel and epsilon.
For sequential NN model, trying different number of layers and the number of neurons in each layer helps the model to extract and combine higher order features that are part of the data.

Metrics –
For Regression model, most appropriate to use Root Mean Square Error , Mean Absolute Error and R2

1.9 Applications / Data Products

1.9.1 Technology Choice

Guideline is to use Node-RED

1.9.2 Justification

Node-RED is a great tool for process flow visualization and more consumer friendly.

1.10 Security, Information Governance and Systems Management

1.10.1 Technology Choice

IBM Identity Access Management is a known choice for security and information governance.

1.10.2 Justification

Identity and Access Management (IAM) integration allows for granular access control at the bucket-level using role-based policies.