Cosmoscope: Martian Atoms

Summer of Innovation 2023

Team: mystic

July, 2023

Abstract

In this report, I focus on the Sample Analysis at Mars (SAM) instrument, a crucial part of NASA's Curiosity rover mission. By studying how molecules are produced and destroyed by various processes in the Martian environment, SAM helps us understand Mars' potential habitability and its historical conditions. I aim to employ machine learning models to discover specific families of these organic chemicals in geological samples from Mars exploration missions, thus enhancing our understanding of the planet's past habitability.

1. Introduction

The data collected by **SAM** was segregated into two parts:

1. **Training:** It consists of **809** observation runs of mass spectroscopy. Each of these mass spectroscopy runs is associated with labels.
2. **Submission:** It consists of **312** observation runs of mass spectroscopy. The labels for each of these have to be predicted and submitted.

1.1. Experimental Run

- Each of the experimental runs, consists of 3 columns **time**, **mass** and **intensity**
- It consists of intensity measurements of different masses at different times with varied relative intensities
- It can be observed that a given molecule will not give high intensities for all masses. That is most of the intensity measurements are **0**

1.2. Labels

- Each molecule can be classified into **nine** labels as shown in **Table 1**
- It is observed that each molecule can belong to **more than one class**

aromatic
hydrocarbon
carboxylic_acid
nitrogen_bearing_compound
chlorine_bearing_compound
sulfur_bearing_compound
alcohol
other_oxygen_bearing_compound
mineral

Table 1: Molecular Classification Labels

2. Insights

To proceed with the classification task the following observations were made.

1. Since the labels are given, this problem comes under **Supervised Learning**
2. Since a single data point can belong to multiple classes, this is a **Multi-Label Classification** problem

Before delving into the details, I have made a few assumptions regarding the data set itself:

1. The prediction of classes will not be affected much if the mass measurements vary by around **0.1** units
2. Any new experimental data will not contain masses exceeding the maximum mass of the entire data set (training + validation)

3. Pre-Processing

This step mostly involved theoretical challenges with respect to **Feature Extraction**. Here each data point is an entire file. I couldn't use it directly due to the following inconsistencies among experimental runs:

1. Length of the experimental run
2. No uniform intensity measurements
3. Range of masses

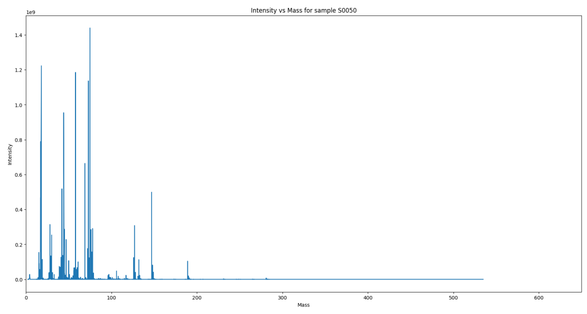


Figure 1: Mass vs Intensity plot for Sample S0050
‘

Therefore, feature extraction was needed to extract uniform features across all experimental runs. In classical Physics and Chemistry, the most important feature scientists look for in Mass Spectroscopy plots is **Peak Intensity Detection**. This technique plots the **relative** intensities for all the masses, and based on which masses hit peak intensities, reasonable conclusions are made. Therefore the first step was to plot the Mass Spectroscopy Intensity Plot by **Sorting** the experimental run by **Mass**. The plot is shown in **Fig**1.

From the graph, we can see that most of the masses have no intensities. Few masses have peak intensities. The masses for which there are peak intensities and the relative intensity values **are our features** for this problem statement.

To detect peaks, the initial approach was to use standard peak detection algorithms to fetch the Top 200 masses which had the highest peaks. These can then be used as features. But this had a few issues:

1. Standard peak detection algorithms were mostly working for a small range of masses
2. It found most of the peaks in concentrated areas while ignoring other areas
3. The number of masses was huge, and this led to isolated random peaks

To tackle this issue, I used a different approach for feature extraction. The method is outlined below:

1. First, the maximum mass across the entire data set (training + validation) is calculated
2. Now, for each data point (experimental run, file, etc.), the plot is divided into bins of size **0.1**.
3. The intensity value for each bin is the maximum intensity among all the masses in that bin

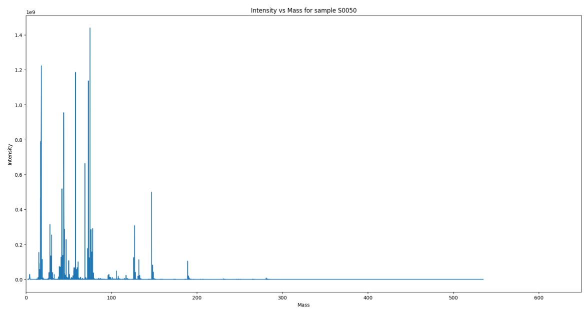


Figure 2: Continuous Mass vs Intensity plot for Sample S0050
‘

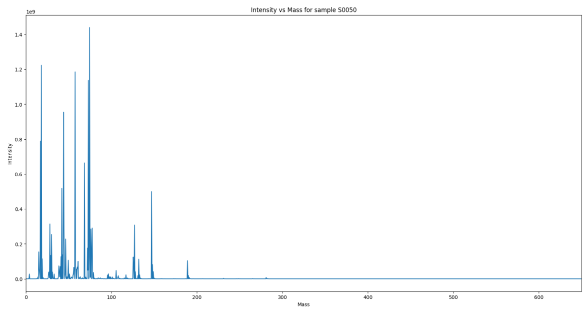


Figure 3: Descretized Mass vs Intensity plot for Sample S0050
‘

4. If an experimental run does not have masses reaching **650** units, the intensity values are padded to **0**

Since the minimum is **0** and the maximum mass obtained was **650**, and considering bin size to be **0.1**, each file could be mapped to a data set containing **6501** masses (features) and corresponding normalized intensities. Finally, a discretized version of the training data is obtained with **312** rows and **6501** columns, which will be uniform across all experimental runs. The same was performed for validation data as well.

The actual **Feature Extraction** is done by an Artificial Neural Network with **6501** inputs. The plots in **Fig**2 and **Fig**3 show that there is not much difference between the continuous and discretized version of the data that is passed on to the ANN.

4. Approach

I used an Artificial Neural Network (Architecture is shown in the next section) for feature extraction and prediction. The training data is split into **Training** and **Validation** data. Validation data is further split into **Validation** and **Evaluation** data. The following specifications were applied to the model:

Type	Percentage
Training	85%
Validation	7.5%
Evaluation	7.5%

Table 2: Training Data Split

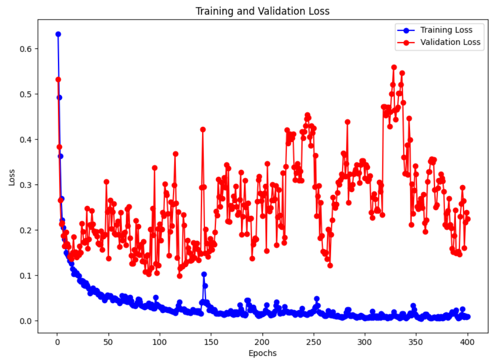


Figure 4: Loss During Training Process

1. Input shape **6501** and output shape **9**
2. Last layer has **Sigmoid** activation function
3. Custom loss function used is **multilabel_aggregated_log_loss**
4. Number of epochs for training is **400**
5. During training, the best model checkpoint is stored based on validation loss

I have split the data according to **Table2**. The training process took about **20 minutes**. The evaluation of the model is shown in later sections.

On training the model, the following plots were obtained. For **Loss Fig4** and **Accuracy Fig5**

5. Model Architecture

The model is a multi-layer neural network for multi-label classification using TensorFlow's Keras API

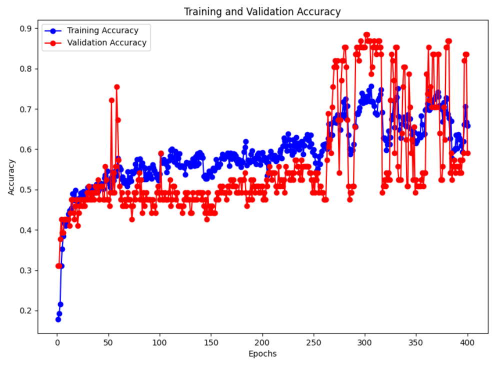
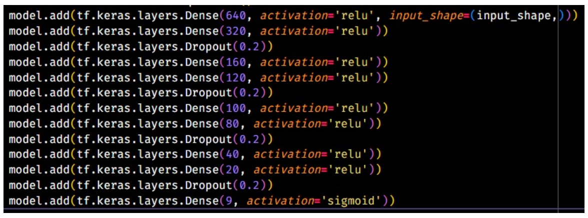
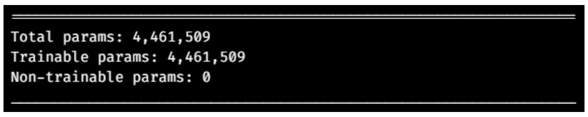


Figure 5: Accuracy During Training Process



```
model.add(tf.keras.layers.Dense(640, activation='relu', input_shape=(input_shape,)))
model.add(tf.keras.layers.Dense(320, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(160, activation='relu'))
model.add(tf.keras.layers.Dense(120, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(100, activation='relu'))
model.add(tf.keras.layers.Dense(80, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(40, activation='relu'))
model.add(tf.keras.layers.Dense(20, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(9, activation='sigmoid'))
```

Figure 6: Tensorflow Model Architecture
‘



```
Total params: 4,461,509
Trainable params: 4,461,509
Non-trainable params: 0
```

Figure 7: Total Parameter Count
‘

with the following architecture **Fig6**:

- **Input Layer:** Unspecified number of neurons depending on the input shape.
- **Dense Layers:** 640, 320, 160, 120, 100, 80, 40, and 20 neurons, all using ReLU activation.
- **Dropout Layers:** Applied after every dense layer with a dropout rate 0.2.
- **Output Layer:** 9 neurons using a sigmoid activation function for multi-label classification.

The model uses the Adam optimizer and is trained with a custom loss function called **multilabel_aggregated_log_loss**. Early stopping is implemented based on validation accuracy with patience of 10 epochs. The total number of parameters is shown in **Fig7**

6. Model Evalution

The model was evaluated using both the **validation** as well as **Evaluation** data that were split from the original training data.

6.1. Validation Data

There are a few metrics shown in the figures. The metrics chosen are:

1. Confusion Matrix in **Fig8**
2. Precision, Recall, F1-score in **Fig9**
3. Log Loss and Accuracy in **Fig10**

6.2. Evaluation Data

There are a few metrics shown in the figures. The metrics chosen are:

1. Confusion Matrix in **Fig11**
2. Precision, Recall, F1-score in **Fig12**
3. Log Loss and Accuracy in **Fig13**

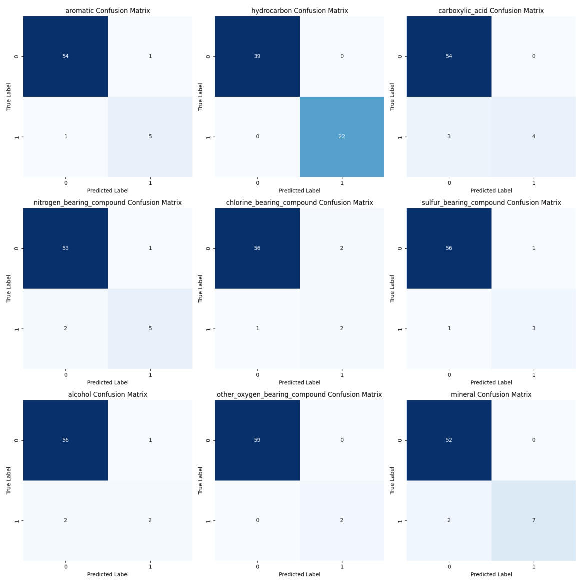
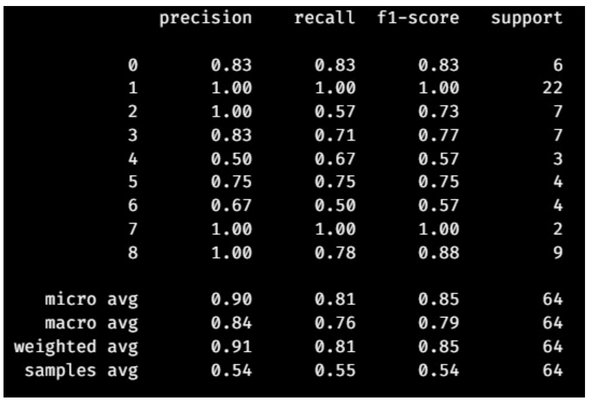


Figure 8: Confusion Matrix For Validation

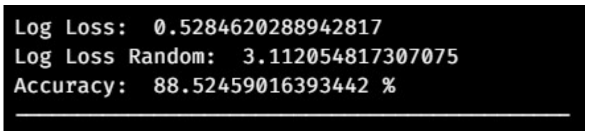


```
              precision    recall  f1-score   support

           0       0.83      0.83      0.83         6
           1       1.00      1.00      1.00        22
           2       1.00      0.57      0.73         7
           3       0.83      0.71      0.77         7
           4       0.50      0.67      0.57         3
           5       0.75      0.75      0.75         4
           6       0.67      0.50      0.57         4
           7       1.00      1.00      1.00         2
           8       1.00      0.78      0.88         9

   micro avg       0.90      0.81      0.85        64
   macro avg       0.84      0.76      0.79        64
weighted avg       0.91      0.81      0.85        64
 samples avg       0.54      0.55      0.54        64
```

Figure 9: Metrics for Validation



```
Log Loss:  0.5284620288942817
Log Loss Random:  3.112054817307075
Accuracy:  88.52459016393442 %
```

Figure 10: Validation Loss and Accuracy

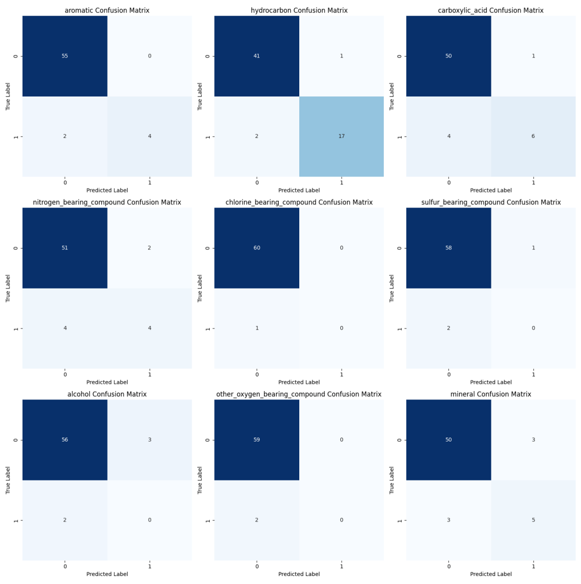
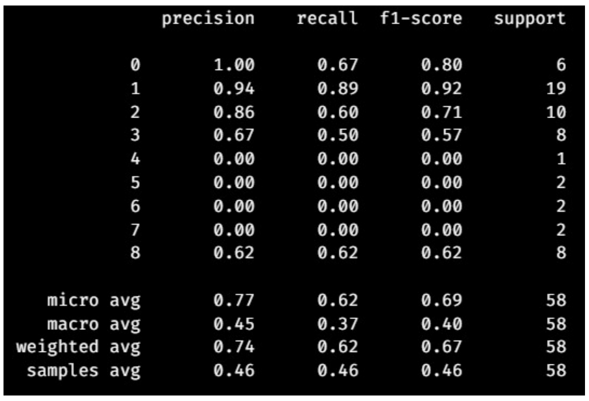


Figure 11: Confusion Matrix For Evaluation

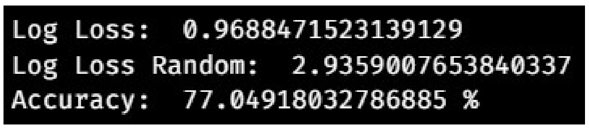


```
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         6
           1       0.94      0.89      0.92        19
           2       0.86      0.60      0.71        10
           3       0.67      0.50      0.57         8
           4       0.00      0.00      0.00         1
           5       0.00      0.00      0.00         2
           6       0.00      0.00      0.00         2
           7       0.00      0.00      0.00         2
           8       0.62      0.62      0.62         8

   micro avg       0.77      0.62      0.69        58
   macro avg       0.45      0.37      0.40        58
weighted avg       0.74      0.62      0.67        58
 samples avg       0.46      0.46      0.46        58
```

Figure 12: Metrics for Evaluation



```
Log Loss:  0.9688471523139129
Log Loss Random:  2.9359007653840337
Accuracy:  77.04918032786885 %
```

Figure 13: Evaluation Loss and Accuracy

7. Conclusion

The following improvements can be made to the model:

1. Better algorithm for peak detection
2. Using a smaller model than bulky Neural Network
3. Predictions for generalized masses instead of constrained

In conclusion, this was a very challenging and fun event, with lots of new technologies used. It was a very nice learning experience. The model, discretized data set, and the Jupyter Notebook can be found in GitHub