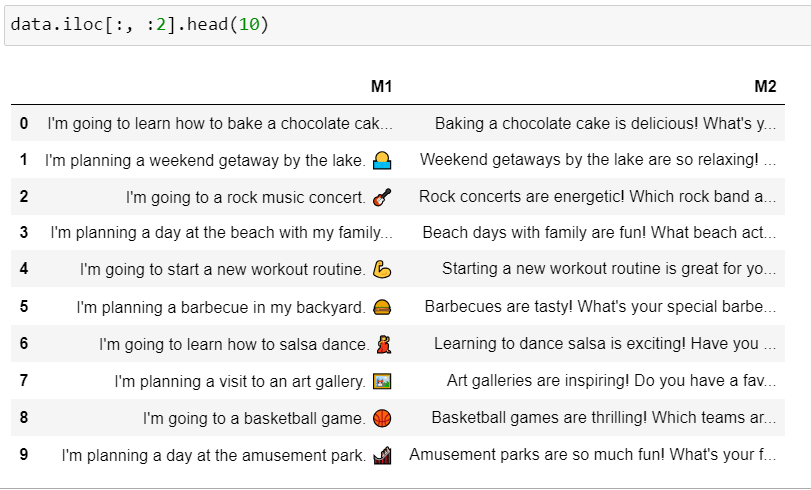
**Muhammad Taha *Artificial Intelligence* Assignment #1**

**Data Collection:**

The dataset includes 214 examples of chats btw two users are generated using different AI tools, to make it sure that that these examples are meaningful, so that we can extract features out of it.



**Feature Extraction:**

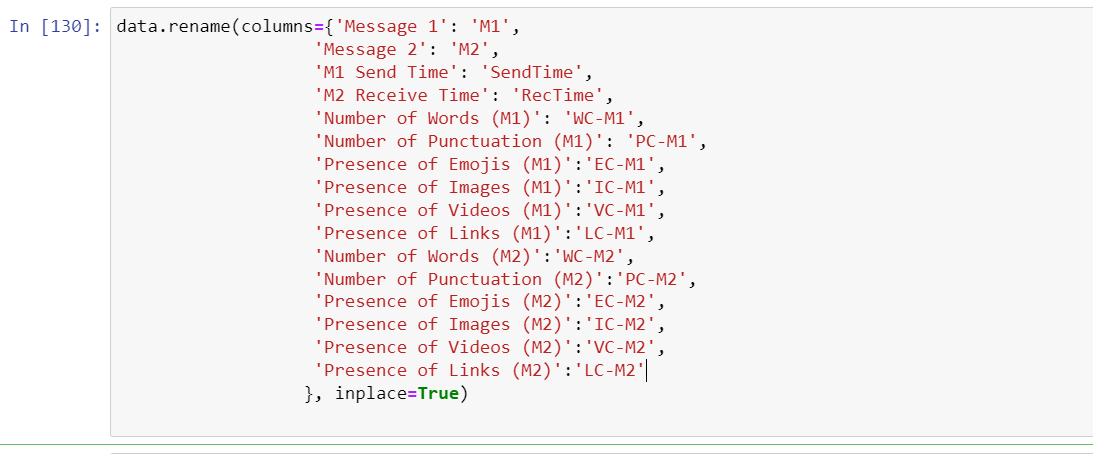
Whenever two users chat, there are different mode in which they express their feelings, thus their chat may include emojis, files, images, videos, punctuations no of words etc. To get a good prediction on “Is the received message is response of 1 message or not” I have extracted following features from the dataset.

*Message 1', 'Message 2', 'Response', 'M1 Send Time', 'M2 Receive Time', 'Number of Words (M1)', 'Number of Punctuation (M1)', 'Presence of Emojis (M1)', 'Presence of Images (M1)', 'Presence of Videos (M1)', 'Presence of Links (M1)', 'Number of Words (M2)', 'Number of Punctuation (M2)', 'Presence of Emojis (M2)', 'Presence of Images (M2)', 'Presence of Videos (M2)', 'Presence of Links (M2)'*

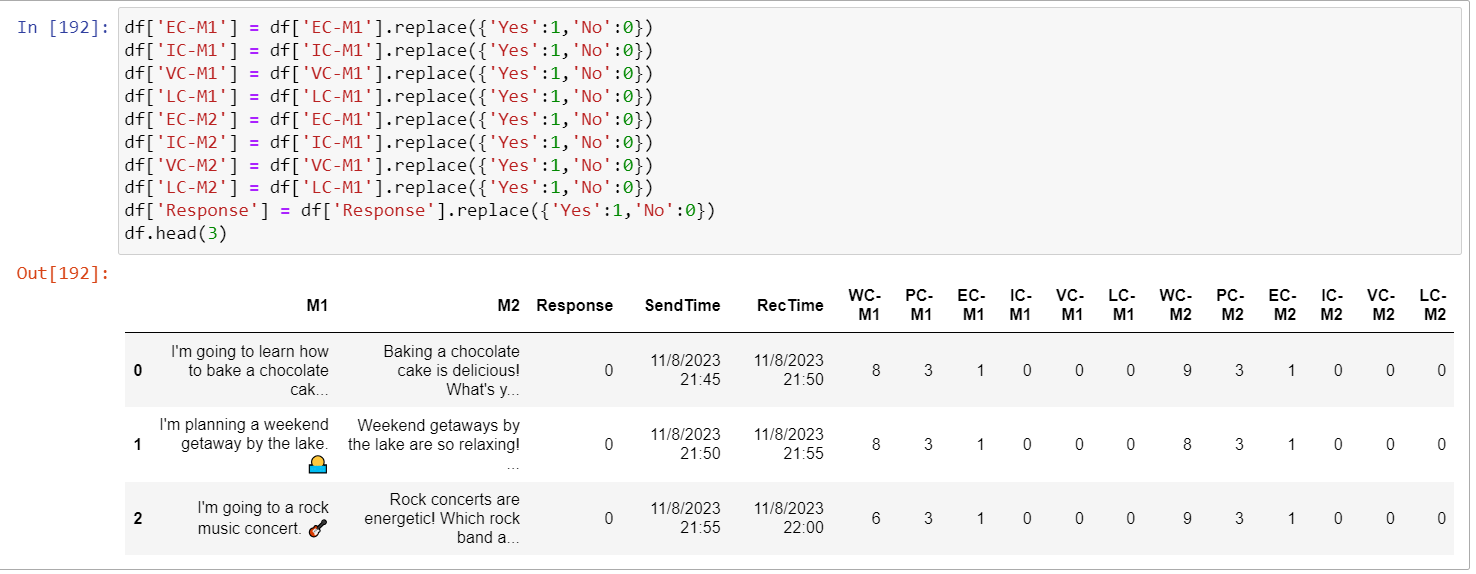
**Exploratory Data Analysis:**

* **Column renaming.**

The extracted features above have a very detailed names initially for the understanding of what kind of features we have selected.  
but make it very clear to user in Jupyter notebook’s interface. The column names have been abbreviated.



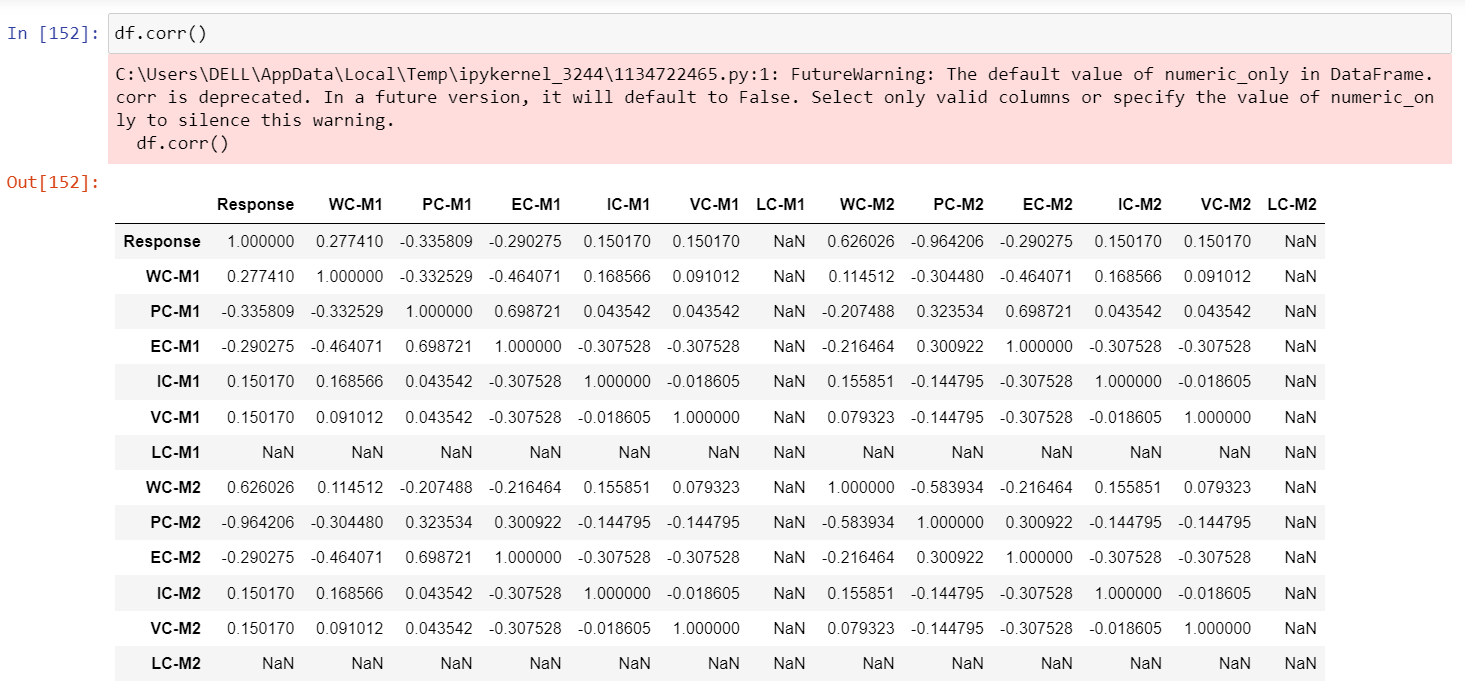
* **Data Type Conversion**

****Since there were multiple features with string values, to train the model we need numerical values, to do so, converted those columns values to numerical values.

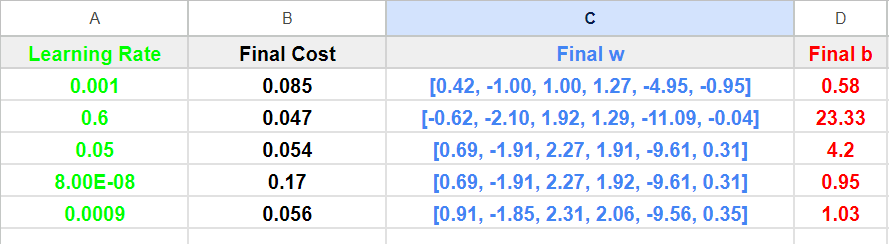
* **Response Time Calculation:**

From the two columns in a dataset where, time of sender and response were recorded.  
we calculated the time and converted it into minutes.

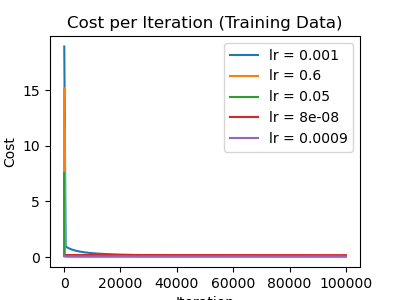
df1['RespTime'] = (df1['RecTime'] - df1['SendTime']).dt.total\_seconds() / 60

* **Correlation checking**
  + df.corr() has been used to check each columns contribution in prediction.
  + ****Moreover, some other EDA has been performed and plot some graphs to understand the data. the visualization is included in the code section.

As we can see in the above table, most of the features are contributing exactly same, so to avoid the data redundancy we drop all those columns which are contributing same in prediction. After dropping the unnecessary columns, we are left with only 6 features, which will be use to train the Binary classification model.

* **Model Evaluation**
  + Split the data into training and testing set with the ratio of 20%
  + Convert the Dataframe into NumPy values
  + Calculate the sigmoid, cost and gradient descent.
  + We used different learning rates to check get the minimum loss
  + Weight vector has been randomly initialized.
  + Bias term has also been randomly initialized
  + w = np.random.randn(x\_train.shape[1])
  + b = np.random.randn()
  + lr = [0.001, 0.6, 0.05, 0.00000008, 0.0009]
  + iterations = 100000
* **Training Data Evaluation with loss graph and Accuracy**

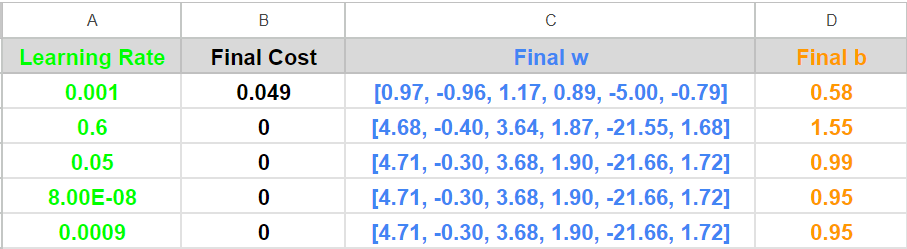
And the loss graph is given below



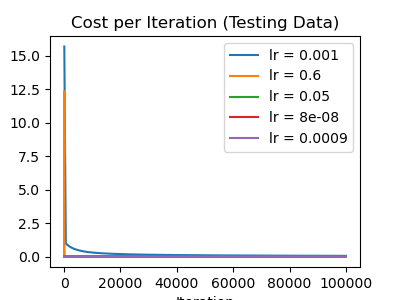
**Modal accuracy on Training data is.**

**R-squared (Training Data): 93.0 %**

* **Testing Data Evaluation with loss graph and Accuracy.**

****

**And the loss graph is given below**



**Modal accuracy on testing data is.**

**R-squared (Testing Data): 100.0%**