
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.

```
data.iloc[:, :2].head(10)
```

	M1	M2
0	I'm going to learn how to bake a chocolate cak...	Baking a chocolate cake is delicious! What's y...
1	I'm planning a weekend getaway by the lake.	Weekend getaways by the lake are so relaxing! ...
2	I'm going to a rock music concert.	Rock concerts are energetic! Which rock band a...
3	I'm planning a day at the beach with my family...	Beach days with family are fun! What beach act...
4	I'm going to start a new workout routine.	Starting a new workout routine is great for yo...
5	I'm planning a barbecue in my backyard.	Barbecues are tasty! What's your special barbe...
6	I'm going to learn how to salsa dance.	Learning to dance salsa is exciting! Have you ...
7	I'm planning a visit to an art gallery.	Art galleries are inspiring! Do you have a fav...
8	I'm going to a basketball game.	Basketball games are thrilling! Which teams ar...
9	I'm planning a day at the amusement park.	Amusement parks are so much fun! What's your f...

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.

In [130]:
```
data.rename(columns={'Message 1': 'M1',
                     'Message 2': 'M2',
                     'M1 Send Time': 'SendTime',
                     'M2 Receive Time': 'RecTime',
                     'Number of Words (M1)': 'WC-M1',
                     'Number of Punctuation (M1)': 'PC-M1',
                     'Presence of Emojis (M1)':'EC-M1',
                     'Presence of Images (M1)':'IC-M1',
                     'Presence of Videos (M1)':'VC-M1',
                     'Presence of Links (M1)':'LC-M1',
                     'Number of Words (M2)':'WC-M2',
                     'Number of Punctuation (M2)':'PC-M2',
                     'Presence of Emojis (M2)':'EC-M2',
                     'Presence of Images (M2)':'IC-M2',
                     'Presence of Videos (M2)':'VC-M2',
                     'Presence of Links (M2)':'LC-M2'
                    }, inplace=True)
```

- **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.

In [192]:
```
df['EC-M1'] = df['EC-M1'].replace({'Yes':1,'No':0})
df['IC-M1'] = df['IC-M1'].replace({'Yes':1,'No':0})
df['VC-M1'] = df['VC-M1'].replace({'Yes':1,'No':0})
df['LC-M1'] = df['LC-M1'].replace({'Yes':1,'No':0})
df['EC-M2'] = df['EC-M1'].replace({'Yes':1,'No':0})
df['IC-M2'] = df['IC-M1'].replace({'Yes':1,'No':0})
df['VC-M2'] = df['VC-M1'].replace({'Yes':1,'No':0})
df['LC-M2'] = df['LC-M1'].replace({'Yes':1,'No':0})
df['Response'] = df['Response'].replace({'Yes':1,'No':0})
df.head(3)
```

Out[192]:

	M1	M2	Response	SendTime	RecTime	WC-M1	PC-M1	EC-M1	IC-M1	VC-M1	LC-M1	WC-M2	PC-M2	EC-M2	IC-M2	VC-M2	LC-M2
0	I'm going to learn how to bake a chocolate cak...	Baking a chocolate cake is delicious! What's y...	0	11/8/2023 21:45	11/8/2023 21:50	8	3	1	0	0	0	9	3	1	0	0	0
1	I'm planning a weekend getaway by the lake.	Weekend getaways by the lake are so relaxing! ...	0	11/8/2023 21:50	11/8/2023 21:55	8	3	1	0	0	0	8	3	1	0	0	0
2	I'm going to a rock music concert.	Rock concerts are energetic! Which rock band a...	0	11/8/2023 21:55	11/8/2023 22:00	6	3	1	0	0	0	9	3	1	0	0	0

- **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.

In [152]:
```
df.corr()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_3244\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[152]:

	Response	WC-M1	PC-M1	EC-M1	IC-M1	VC-M1	LC-M1	WC-M2	PC-M2	EC-M2	IC-M2	VC-M2	LC-M2
Response	1.000000	0.277410	-0.335809	-0.290275	0.150170	0.150170	NaN	0.626026	-0.964206	-0.290275	0.150170	0.150170	NaN
WC-M1	0.277410	1.000000	-0.332529	-0.464071	0.168566	0.091012	NaN	0.114512	-0.304480	-0.464071	0.168566	0.091012	NaN
PC-M1	-0.335809	-0.332529	1.000000	0.698721	0.043542	0.043542	NaN	-0.207488	0.323534	0.698721	0.043542	0.043542	NaN
EC-M1	-0.290275	-0.464071	0.698721	1.000000	-0.307528	-0.307528	NaN	-0.216464	0.300922	1.000000	-0.307528	-0.307528	NaN
IC-M1	0.150170	0.168566	0.043542	-0.307528	1.000000	-0.018605	NaN	0.155851	-0.144795	-0.307528	1.000000	-0.018605	NaN
VC-M1	0.150170	0.091012	0.043542	-0.307528	-0.018605	1.000000	NaN	0.079323	-0.144795	-0.307528	-0.018605	1.000000	NaN
LC-M1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
WC-M2	0.626026	0.114512	-0.207488	-0.216464	0.155851	0.079323	NaN	1.000000	-0.583934	-0.216464	0.155851	0.079323	NaN
PC-M2	-0.964206	-0.304480	0.323534	0.300922	-0.144795	-0.144795	NaN	-0.583934	1.000000	0.300922	-0.144795	-0.144795	NaN
EC-M2	-0.290275	-0.464071	0.698721	1.000000	-0.307528	-0.307528	NaN	-0.216464	0.300922	1.000000	-0.307528	-0.307528	NaN
IC-M2	0.150170	0.168566	0.043542	-0.307528	1.000000	-0.018605	NaN	0.155851	-0.144795	-0.307528	1.000000	-0.018605	NaN
VC-M2	0.150170	0.091012	0.043542	-0.307528	-0.018605	1.000000	NaN	0.079323	-0.144795	-0.307528	-0.018605	1.000000	NaN
LC-M2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

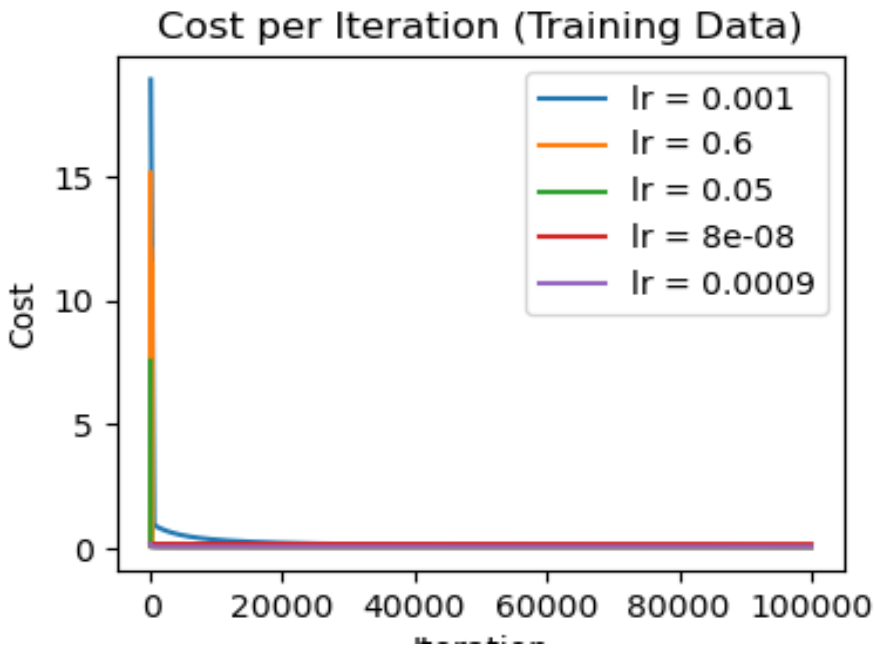
As we can see in the above table, most of the features are contributing exactly same, so to avoid the data redundancy we drop all those columns which are contributing same in prediction. After dropping the unnecessary columns, we are left with only 6 features, which will be use to train the Binary classification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

A	B	C	D
Learning Rate	Final Cost	Final w	Final b
0.001	0.085	[0.42, -1.00, 1.00, 1.27, -4.95, -0.95]	0.58
0.6	0.047	[-0.62, -2.10, 1.92, 1.29, -11.09, -0.04]	23.33
0.05	0.054	[0.69, -1.91, 2.27, 1.91, -9.61, 0.31]	4.2
8.00E-08	0.17	[0.69, -1.91, 2.27, 1.92, -9.61, 0.31]	0.95
0.0009	0.056	[0.91, -1.85, 2.31, 2.06, -9.56, 0.35]	1.03

And the loss graph is given below



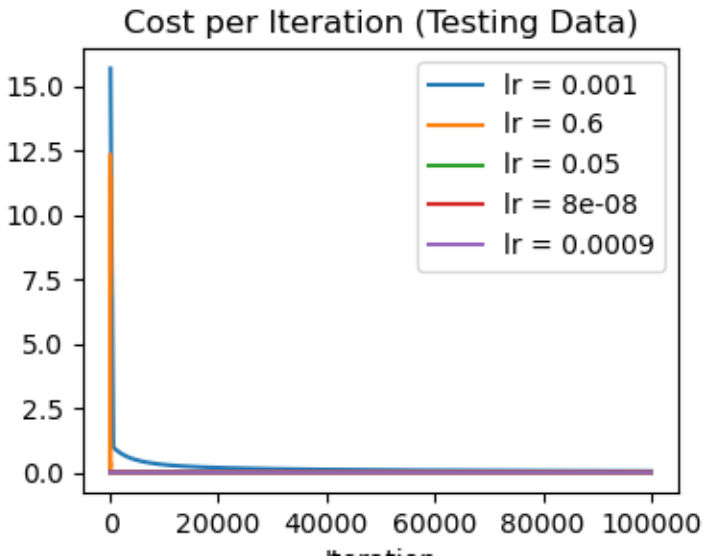
Modal accuracy on Training data is.

R-squared (Training Data): 93.0 %

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

A	B	C	D
Learning Rate	Final Cost	Final w	Final b
0.001	0.049	[0.97, -0.96, 1.17, 0.89, -5.00, -0.79]	0.58
0.6	0	[4.68, -0.40, 3.64, 1.87, -21.55, 1.68]	1.55
0.05	0	[4.71, -0.30, 3.68, 1.90, -21.66, 1.72]	0.99
8.00E-08	0	[4.71, -0.30, 3.68, 1.90, -21.66, 1.72]	0.95
0.0009	0	[4.71, -0.30, 3.68, 1.90, -21.66, 1.72]	0.95

And the loss graph is given below



Modal accuracy on testing data is.

R-squared (Testing Data): 100.0%