MD Jisan Uddin Chowdhury

Matriculation No: 26100

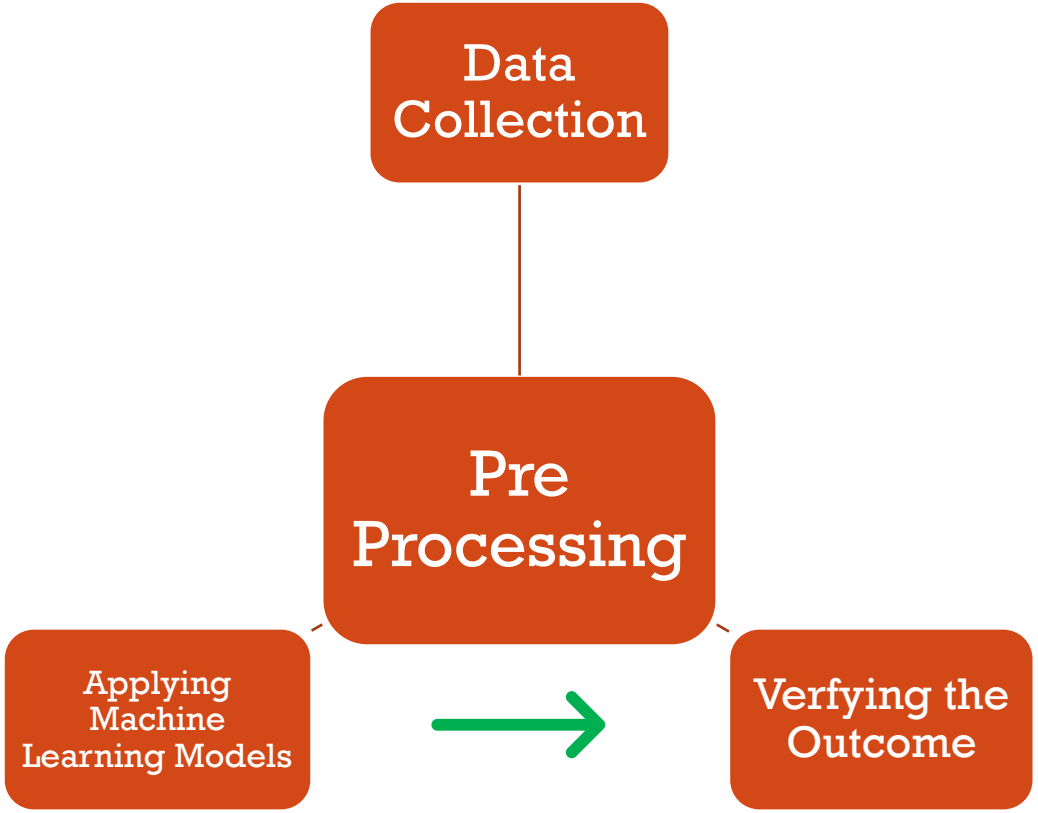
Communication & Information Engineering

SPAM EMAIL CLASSIFICATION USING NLP

- **Overview Of the Project:**

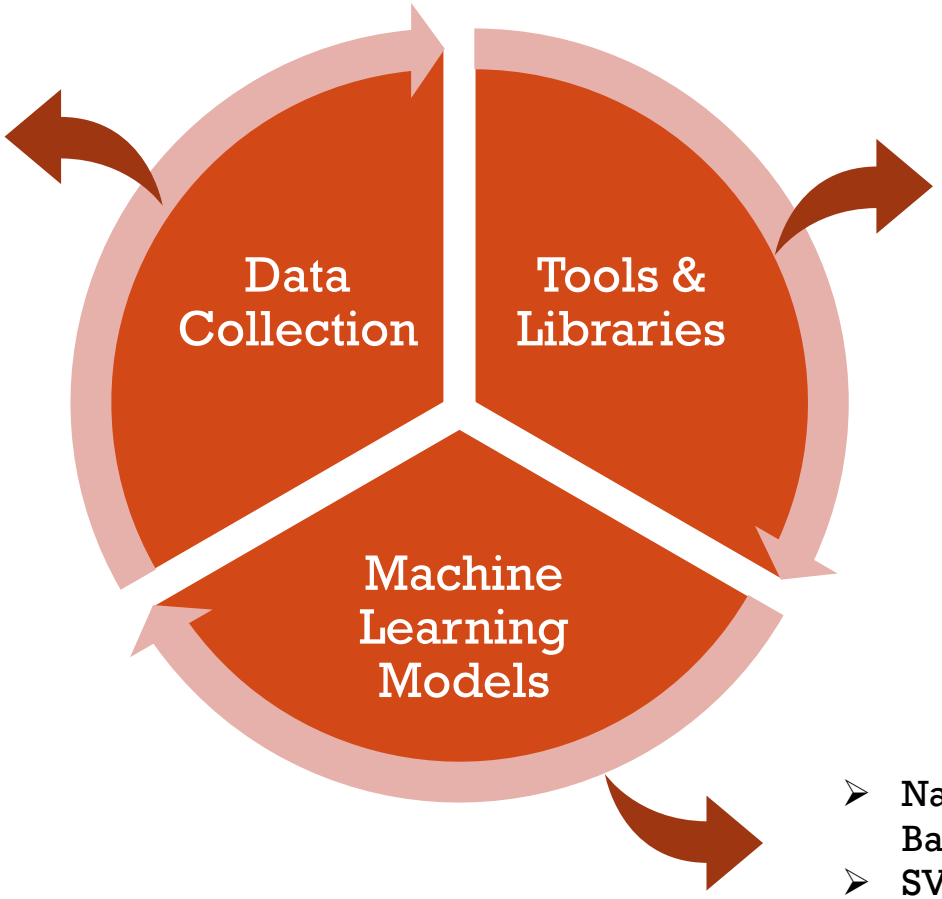
  This presentation illustrates how we can classify spam and non spam emails using some of the machine learning models.

AGENDA:


Data Collection
Pre Processing
Applying Machine Learning Models
Verfying the Outcome

WORKFLOW

- Kaggle
- StackOverflow
- Google Public Dataset

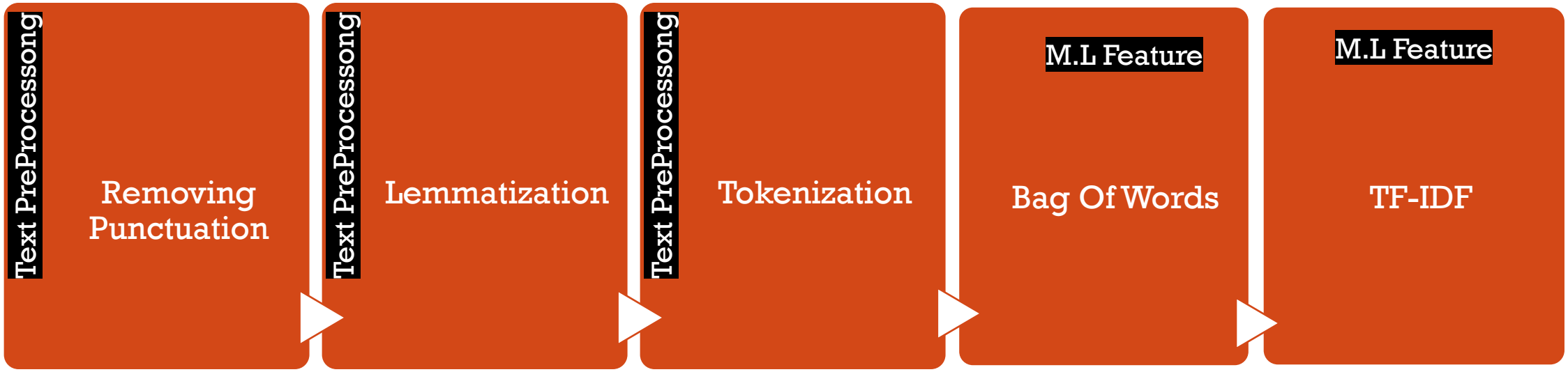


- Jupyter Notebook
- Matplotlib
- Seaborn
- Numpy
- Pandas
- Sklearn
- Wordcloud

- Naive Bayes
- SVM

Confusion Matrix

FEATURES


Text PreProcessong
Removing Punctuation
Text PreProcessong
Lemmatization
Text PreProcessong
Tokenization
M.L Feature
Bag Of Words
M.L Feature
TF-IDF

IMPLEMENTATION

```
In [4]: dt.head(10)
```

Out[4]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1
5	Subject: great nnews hello , welcome to medzo...	1
6	Subject: here ' s a hot play in motion homela...	1
7	Subject: save your money buy getting this thin...	1
8	Subject: undeliverable : home based business f...	1
9	Subject: save your money buy getting this thin...	1

Before ↑↓ After

```
In [52]: dt.head(10)
```

Out[52]:

	text	spam	length	tokenized	filtered_text
0	Subject: naturally irresistible your corporate...	1	1484	[Subject, :, naturally, irresistible, your, co...	naturally irresistible corporate identity lt r...
1	Subject: the stock trading gunslinger fanny i...	1	598	[Subject, :, the, stock, trading, gunslinger, ...	stock trading gunslinger fanny merrill muzo co...
2	Subject: unbelievable new homes made easy im ...	1	448	[Subject, :, unbelievable, new, homes, made, e...	unbelievable new homes made easy im wanting sh...
3	Subject: 4 color printing special request add...	1	500	[Subject, :, 4, color, printing, special, requ...	4 color printing special request additional in...
4	Subject: do not have money , get software cds ...	1	235	[Subject, :, do, not, have, money, ,, get, sof...	money get software cds software compatibility ...
5	Subject: great nnews hello , welcome to medzo...	1	478	[Subject, :, great, nnews, hello, ,, welcome, ...	great nnews hello welcome medzonline sh ground...
6	Subject: here ' s a hot play in motion homela...	1	9340	[Subject, :, here, ', s, a, hot, play, in, mot...	hot play motion homeland security investments...
7	Subject: save your money buy getting this thin...	1	446	[Subject, :, save, your, money, buy, getting, ...	save money buy getting thing tried cialls yet ...
8	Subject: undeliverable : home based business f...	1	507	[Subject, :, undeliverable, :, home, based, bu...	undeliverable home based business grownups mes...
9	Subject: save your money buy getting this thin...	1	446	[Subject, :, save, your, money, buy, getting, ...	save money buy getting thing tried cialls yet ...

Bag of Words

```
In [29]: bag = CountVectorizer()
         count = bag.fit_transform(dt['filtered_text'].values)

In [30]: print(count.shape)

         (5728, 37158)

In [53]: print(count)

         15
```

TF-IDF

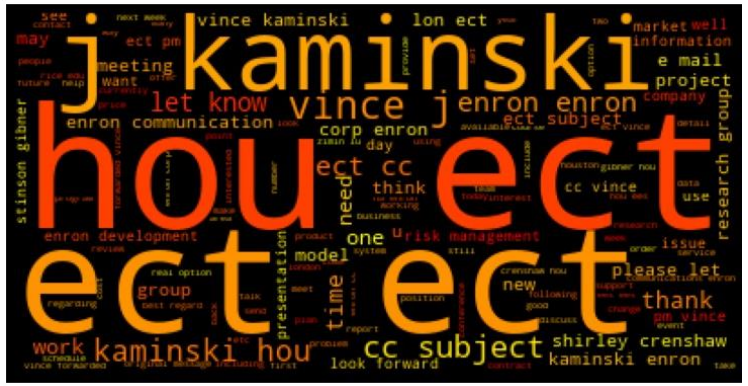
```
In [31]: tfidf_vectorizer = TfidfTransformer().fit(count)
         tfidf = tfidf_vectorizer.transform(count)

In [32]: classifier = MultinomialNB()
         targets = dt['spam'].values
         classifier.fit(count, targets)

Out[32]: MultinomialNB()

In [33]: print(tfidf)

           (0, 36494)    0.06909969862066065
           (0, 36361)    0.06758050177560837
           (0, 36359)    0.0596882190240676
           (0, 35954)    0.13195884687352108
           (0, 34965)    0.052342045903568585
           (0, 34731)    0.0938569785598016
           (0, 33491)    0.06922230324784197
           (0, 32967)    0.09491736727758139
           (0, 32775)    0.06074580071568714
```

```
In [34]: examples = ['Free Offer!! Buy now',"Million dollar Lottery","Please send the attachments"]
         example_counts = bag.transform(examples)
         example_tfidf = tfidf_vectorizer.transform(example_counts)
         predictions_tfidf = classifier.predict(example_tfidf)

         print(predictions_tfidf)

         ['SPAM' 'SPAM' 'NON-SPAM']
```

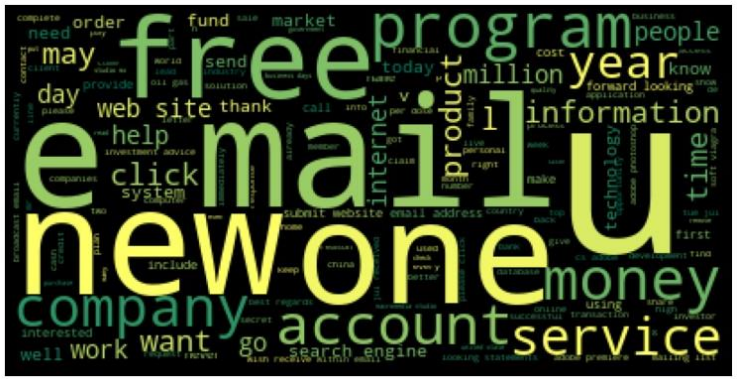
WORDCLOUD

- Graphical visualization of word frequency that appears more in a source text
- Quick and Informative

Non Spam



Spam



MODELLING

```
In [35]: matrix = np.matrix(np.c_[list_alpha, score_train, score_test, recall_test, precision_test])
         models = pd.DataFrame(data = matrix, columns =
                    ['Alpha', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
         models.head(n=10)
```

Out[35]:

	Alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.00001	0.999739	0.981491	0.941176	0.987234
1	0.11001	0.998958	0.989952	0.987830	0.974000
2	0.22001	0.998958	0.988895	0.987830	0.970120
3	0.33001	0.998436	0.988366	0.987830	0.968191
4	0.44001	0.998176	0.988366	0.987830	0.968191
5	0.55001	0.997654	0.988366	0.987830	0.968191
6	0.66001	0.997394	0.988366	0.987830	0.968191
7	0.77001	0.997133	0.987837	0.987830	0.966270
8	0.88001	0.996873	0.988366	0.987830	0.968191
9	0.99001	0.996612	0.988366	0.987830	0.968191

Naive Bayes Model

```
In [36]: best_index = models['Test Precision'].idxmax()
         models.iloc[best_index, :]
```

Out[36]:
```
Alpha              19.030010
Train Accuracy      0.897055
Test Accuracy       0.870439
Test Recall         0.503043
Test Precision      1.000000
Name: 173, dtype: float64
```

```
In [37]: models[models['Test Precision']==1].head(n=5)
```

Out[37]:

	Alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
173	19.03001	0.897055	0.870439	0.503043	1.0
174	19.14001	0.896794	0.869910	0.501014	1.0
175	19.25001	0.896534	0.868324	0.494929	1.0
176	19.36001	0.896013	0.867266	0.490872	1.0
177	19.47001	0.896013	0.867266	0.490872	1.0

```
In [38]: best_index = models[models['Test Precision']==1]['Test Accuracy'].idxmax()
         bayes = naive_bayes.MultinomialNB(alpha=list_alpha[best_index])
         bayes.fit(x_train, y_train)
         models.iloc[best_index, :]
```

Out[38]:
```
Alpha              19.030010
Train Accuracy      0.897055
Test Accuracy       0.870439
Test Recall         0.503043
Test Precision      1.000000
Name: 173, dtype: float64
```

Out[41]:

	C	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	500.0	1.0	0.978847	0.93712	0.980892
1	600.0	1.0	0.978847	0.93712	0.980892
2	700.0	1.0	0.978847	0.93712	0.980892
3	800.0	1.0	0.978847	0.93712	0.980892
4	900.0	1.0	0.978847	0.93712	0.980892

```
In [42]: best_index = models['Test Precision'].idxmax()
         models.iloc[best_index, :]
```

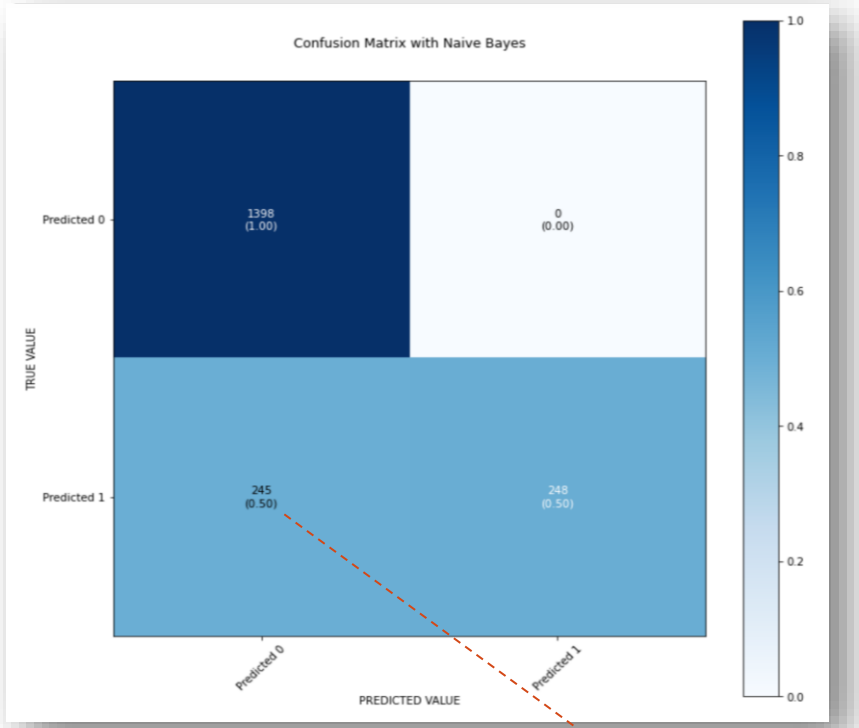
Out[42]:
```
C                 500.000000
Train Accuracy      1.000000
Test Accuracy       0.978847
Test Recall         0.937120
Test Precision      0.980892
Name: 0, dtype: float64
```

SVM

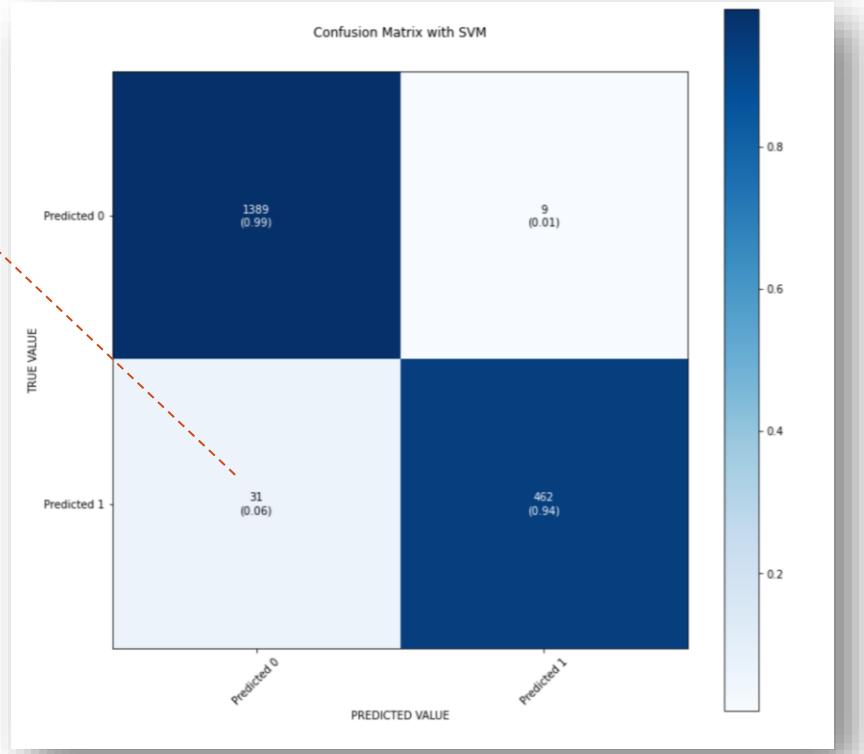
```
In [44]: best_index = models['Test Accuracy'].idxmax()
         svc = svm.SVC(C=list_C[best_index])
         svc.fit(x_train, y_train)
         models.iloc[best_index, :]
```

Out[44]:
```
C                 500.000000
Train Accuracy      1.000000
Test Accuracy       0.978847
Test Recall         0.937120
Test Precision      0.980892
Name: 0, dtype: float64
```

OUTPUT VERIFICATION



Misclassified only 31 spam with SVM model



Misclassified 245 spam as non spam email with Naive Bayes Model

FOR HEARING ME.