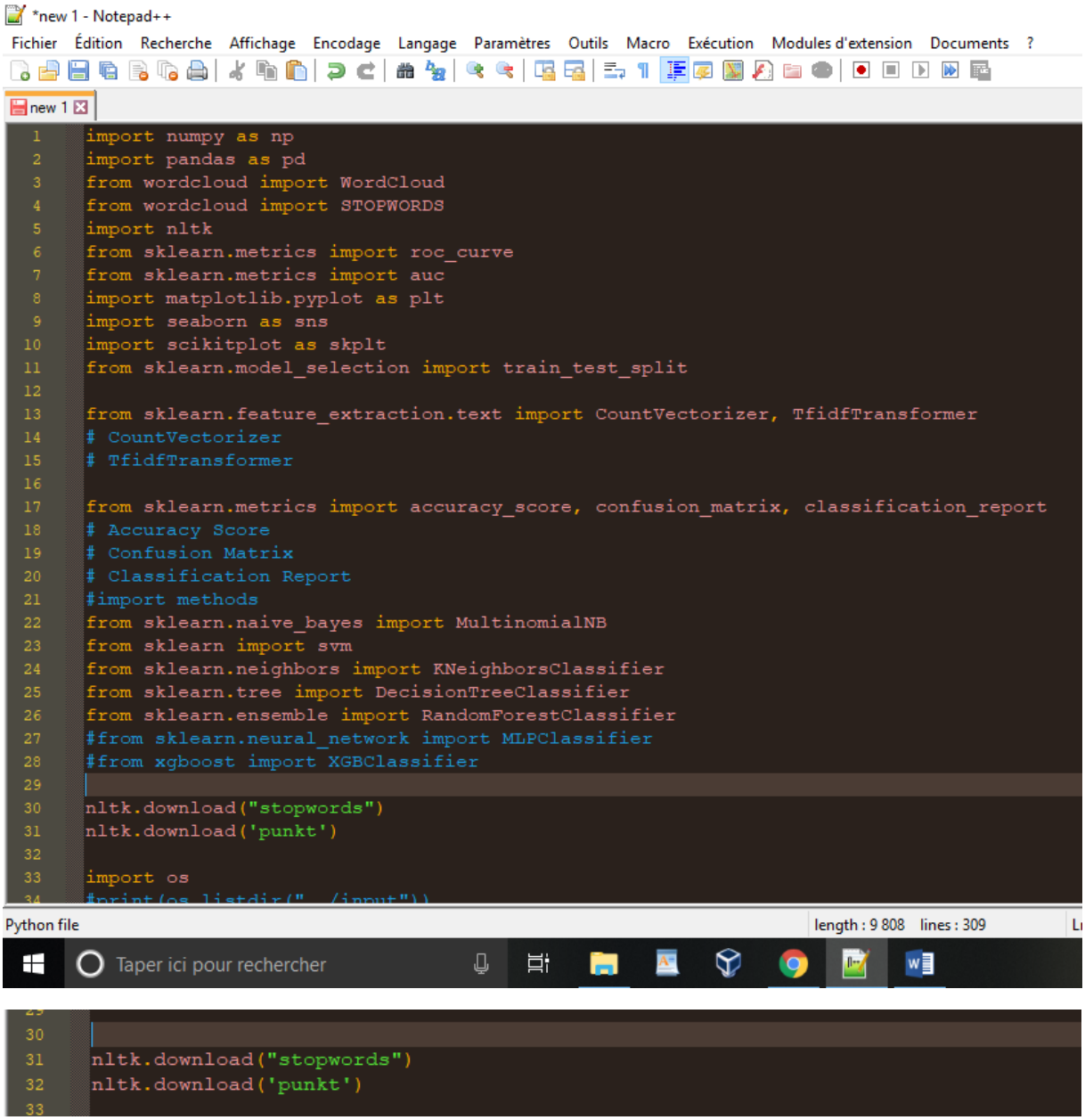
Spam Project of Machine Learning:

Members:

- ***Edna Batana Diane***
- ***Loukrimi Souad***
- ***Zemmouri Kenza***



```
import os
#print(os.listdir("../input"))
```

Read Data :

```
# ###### Read Data:
df = pd.read_csv("/Users/elizabethlorelei/Downloads/spam.csv", encoding = 'latin-1')
```

Show Data :

```


# ###### Show Data :
df.head(11)
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
5	spam	FreeMsg Hey there darling it's been 3 week's n...	NaN	NaN	NaN
6	ham	Even my brother is not like to speak with me. ...	NaN	NaN	NaN
7	ham	As per your request 'Melle Melle (Oru Minnamin...	NaN	NaN	NaN
8	spam	WINNER!! As a valued network customer you have...	NaN	NaN	NaN
9	spam	Had your mobile 11 months or more? U R entitle...	NaN	NaN	NaN
10	ham	I'm gonna be home soon and i don't want to tal...	NaN	NaN	NaN

```

df = df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'])
df.columns = ['Label', 'Message']
df.head()
```

	Label	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Splitting the labels and the data separately:

```
77
78  # ##### Splitting the labels and the data separately :
79  df_labels = df['Label']
80  df_labels.head(11)
81
```

```
0      ham
1      ham
2     spam
3      ham
4      ham
5     spam
6      ham
7      ham
8     spam
9     spam
10     ham
Name: Label, dtype: object
```

Data Visualization :

- To check the most used word in Ham sms and Spam SMS
- To visualize the percentage of Ham and Spam SMS

new 1

```
104  # Stopwords (Шумовые слова)
105  stopwords = STOPWORDS
106  stopwords = list(stopwords)
107  STOPWORDS = nltk.corpus.stopwords.words('english')
108  stopwords = stopwords + STOPWORDS
109
110  ham_dataset = df[df.Label == 'ham']
111  spam_dataset = df[df.Label == 'spam']
112  ham_words = ' '
113  spam_words = ' '
114
115  for words in ham_dataset.Message:
116      txt = words.lower()
117      tokens = nltk.word_tokenize(txt)
118      for word in tokens:
119          ham_words = ham_words + word + " "
120  for words in spam_dataset.Message:
121      txt = words.lower()
122      tokens = nltk.word_tokenize(txt)
123      for word in tokens:
124          spam_words = spam_words + word + " "
125
126  def gen_wordcloud(wordcloud):
127      plt.figure(figsize = (10,8))
128      plt.imshow(wordcloud)
129      plt.tight_layout(pad=0)
130      plt.axis('off')
131      plt.show()
132  print("\n")
133  print("\t\t\t\t HAM WORDS")
134  wordcloud = WordCloud(background_color = 'white', width = 500, height = 500, stopwords = stopwords,
135                        max_words = 500, max_font_size = 50, random_state = 42).generate(ham_words)
136  gen_wordcloud(wordcloud)
137  print("\t\t\t\t SPAM WORDS")
138  wordcloud = WordCloud(background_color = 'white', width = 500, height = 500, stopwords = stopwords,
139                        max_words = 500, max_font_size = 50, random_state = 42).generate(spam_words)
140  gen_wordcloud(wordcloud)
141
```

Python file | length : 9 802 lines : 308 | Ln : 126 C

HAM WORDS

SPAM WORDS

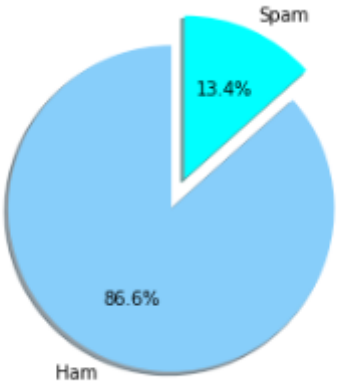
Plotting ham and spam data % in pie chart:

```
# ##### Plotting ham and spam data % in pie chart :


count_Class = pd.value_counts(df.Label, sort = True)

# Data to Plot
labels = 'Ham', 'Spam'
sizes = [count_Class[0], count_Class[1]]
colors = ['lightskyblue', 'aqua']
explode = (0.1, 0.1)

# Plot
plt.pie(sizes, explode = explode, labels = labels, colors = colors,
        autopct = '%1.1f%%', shadow = True, startangle = 90)
plt.axis('equal')
plt.show()
```



Splitting the Test and Train Data:

```

# ###### Splitting the Test and Train Data:

train_set, test_set, train_label, test_label = train_test_split(df, df_labels, test_size = 0.33, random_state = 42)
print(train_set.shape)
print(test_set.shape)
print("\nThe Trainset consists of {} records and {} features".format(train_set.shape[0],train_set.shape[1]))
print("\nThe Testset consists of {} records and {} features".format(test_set.shape[0],train_set.shape[1]))


```

```
(3733, 2)
(1839, 2)

The Trainset consists of 3733 records and 2 features

The Testset consists of 1839 records and 2 features
```

Extracting N-grams from the Text Data:

```
176
177  # ##### Extracting N-grams from the Text Data:
178
179  countvect = CountVectorizer(ngram_range = (2,2), )
180  x_counts = countvect.fit(train_set.Message)
181
182  # preparing for training set
183  x_train_df = countvect.transform(train_set.Message)
184
185  # preparing for test set
186  x_test_df = countvect.transform(test_set.Message)
187
```

Data Model:

The Algorithms used below are:

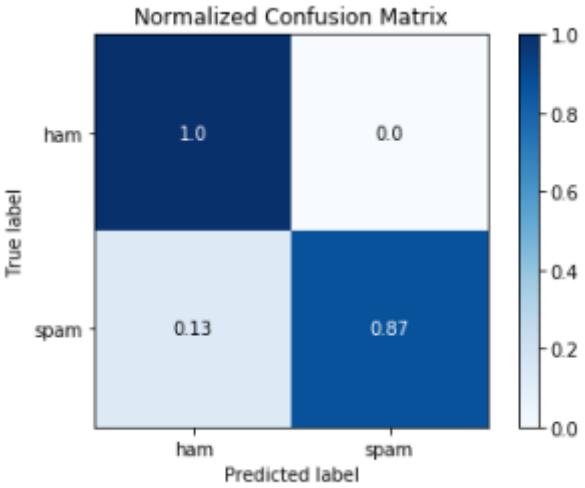
- Naive Bayes
- K-Nearest
- Decision Tree
- Support Vector Machine
- Random Forest

Naive Bayes classifier :

```
191  # ######  Naive Bayes classifier :
192
193
194  clf = MultinomialNB()
195  clf.fit(x_train_df,train_set.Label)
196  predicted_values_NB = clf.predict(x_test_df)
197  predictions = dict()
198  accuracy = accuracy_score(test_set.Label, predicted_values_NB)
199  predictions['Naive Bayes'] = accuracy * 100
200  confusionmatrix = confusion_matrix(test_set.Label, predicted_values_NB)
201  print("The accuracy of Naive Bayes clasifier is {}%".format(accuracy * 100))
202  print("\n", confusionmatrix)
203  skplt.metrics.plot_confusion_matrix(test_set.Label, predicted_values_NB, normalize = True)
204  plt.show()
205
```

```
The accuracy of Naive Bayes clasifier is 97.87928221859707%

 [[1581    6]
 [33      219]]
```



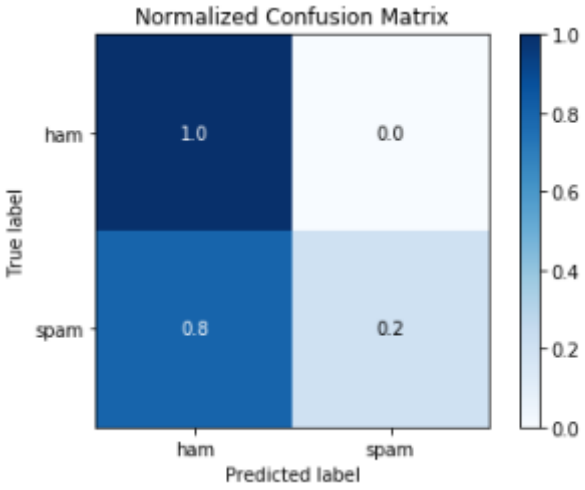
K-Nearest Neighbors algorithm :

```
209 #KNN = KNeighborsClassifier(metric = 'euclidean')
210 KNN = KNeighborsClassifier()
211 KNN.fit(x_train_df, train_set.Label)
212 predicted_values_KNN = KNN.predict(x_test_df)
213 print(predicted_values_KNN)
214 accuracy_KNN = accuracy_score(test_set.Label, predicted_values_KNN)
215 predictions['K-Nearest Neighbors algorithm'] = accuracy_KNN * 100
216 print("\nThe accuracy of K-Nearest Neighbors algorithm is {}%".format(accuracy_KNN * 100))
217 confusion_matrix_KNN = confusion_matrix(test_set.Label, predicted_values_KNN)
218 print("\n", confusion_matrix_KNN)
219 skplt.metrics.plot_confusion_matrix(test_set.Label, predicted_values_KNN, normalize = True)
220 plt.show()
221
```

```
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']

The accuracy of K-Nearest Neighbors algorithm is 89.07014681892332%

 [[1587    0]
 [201    51]]
```



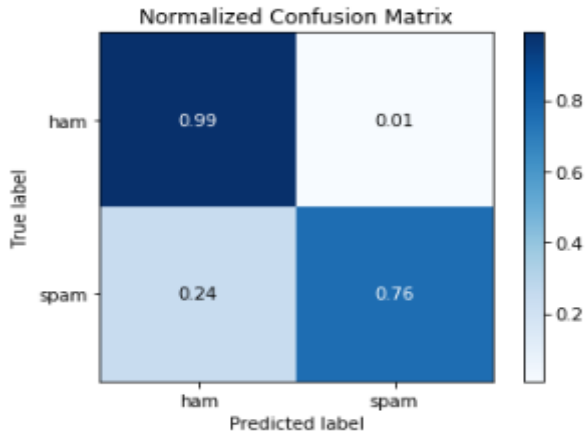
Decision Tree learning :

```
221
222  # #### Decision Tree learning :
223
224  DT = DecisionTreeClassifier()
225  DT.fit(x_train_df, train_set.Label)
226  predicted_values_DT = DT.predict(x_test_df)
227  print(predicted_values_DT)
228  accuracy_DT = accuracy_score(test_set.Label, predicted_values_DT)
229  predictions['Decision Tree learning'] = accuracy_DT * 100
230  print("\nThe accuracy of Decision Tree learning is {}%".format(accuracy_DT * 100))
231  confusion_matrix_DT = confusion_matrix(test_set.Label, predicted_values_DT)
232  print("\n", confusion_matrix_DT)
233  skplt.metrics.plot_confusion_matrix(test_set.Label, predicted_values_DT, normalize = True)
234  plt.show()
235
236
```

['ham' 'ham' 'spam' ... 'ham' 'ham' 'spam'] The accuracy of Decision Tree learning is 96.08482871125612%

[[1576 11]

[61 191]]



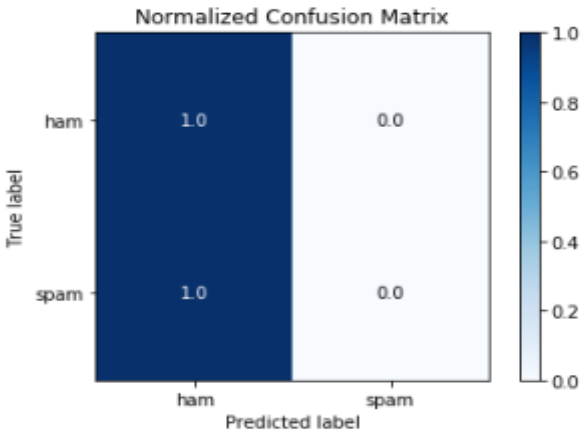
Support Vector Machine (SVM) :

```
238
239  # ###### Support Vector Machine (SVM) :
240
241  SVM = svm.SVC()
242  SVM.fit(x_train_df, train_set.Label)
243  predicted_values_SVM = SVM.predict(x_test_df)
244  print(predicted_values_SVM)
245  accuracy_SVM = accuracy_score(test_set.Label, predicted_values_SVM)
246  predictions['Support Vector Machine (SVM)'] = accuracy_SVM * 100
247  print("\nThe accuracy of Support Vector Machine (SVM) is {}%".format(accuracy_SVM * 100))
248  confusion_matrix_SVM = confusion_matrix(test_set.Label, predicted_values_SVM)
249  print("\n", confusion_matrix_SVM)
250  skplt.metrics.plot_confusion_matrix(test_set.Label, predicted_values_SVM, normalize = True)
251  plt.show()
252
```

```
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']

The accuracy of Support Vector Machine (SVM) is 86.2969004893964%

 [[1587    0]
 [252      0]]
```

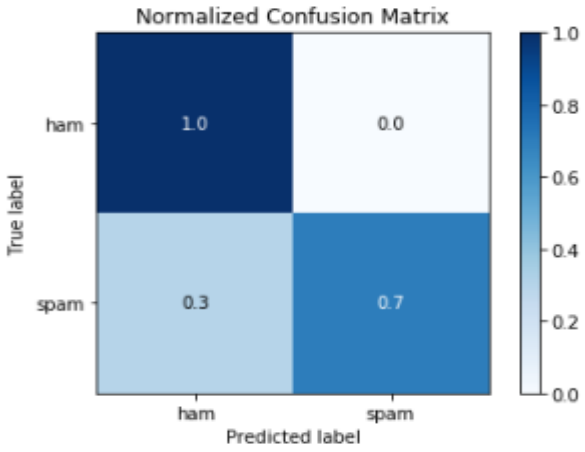


Random Forest:

```
254
255  # ###### Random Forest :
256
257  RF = RandomForestClassifier(n_estimators = 100, oob_score = True, random_state = 123456)
258  # n_estimators - количество деревьев в лесе
259  # oob_score - использовать ли образцы вне примеров для оценки точности обобщения
260  RF.fit(x_train_df, train_set.Label)
261  predicted_values_RF = RF.predict(x_test_df)
262  print(predicted_values_RF)
263  accuracy_RF = accuracy_score(test_set.Label, predicted_values_RF)
264  predictions['Random Forest'] = accuracy_RF * 100
265  print("\nThe accuracy of Random Forest is {}%".format(accuracy_RF * 100))
266  confusion_matrix_RF = confusion_matrix(test_set.Label, predicted_values_RF)
267  print("\n", confusion_matrix_RF)
268  skplt.metrics.plot_confusion_matrix(test_set.Label, predicted_values_RF, normalize = True)
269  plt.show()
270
```

```
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham'] The accuracy of Random Forest
is 95.86731919521479%
```

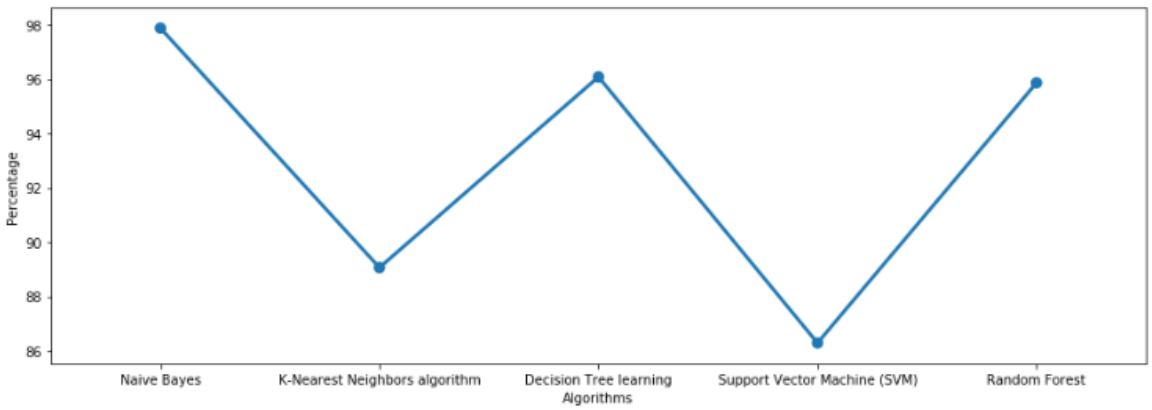
```
[[1587  0]
[76  176]]
```



Méthodes Comparison :

```
271
272  # ##### Method Comparison:
273
274  fig, (ax1) = plt.subplots(ncols = 1, sharey = True,figsize = (15,5))
275  df = pd.DataFrame(list(predictions.items()),columns = ['Algorithms','Percentage'])
276  display(df)
277  sns.pointplot(x = "Algorithms", y = "Percentage", data = df,ax = ax1);
278
```

	Algorithms	Percentage
0	Naive Bayes	97.879282
1	K-Nearest Neighbors algorithm	89.070147
2	Decision Tree learning	96.084829
3	Support Vector Machine (SVM)	86.296900
4	Random Forest	95.867319



ROC Accuracy :

```
#pr, tpr, thresholds = roc_curve(testset.v1,predicted_values_XGB, pos_label=2)
test_prediction = test_set.Label.tolist()
predicted_values = predicted_values_NB.tolist()
test_prediction = [1 if pred=="spam" else 0 for pred in test_prediction]
predicted_values = [1 if pred=="spam" else 0 for pred in predicted_values]
fpr, tpr, thresholds = roc_curve(test_prediction,predicted_values)
roc_auc = auc(fpr, tpr)
print("The ROC Accuracy is {}".format(roc_auc))
```

The ROC Accuracy is 0.9326334503555676

```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b',
label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```