```python
import pandas as pd
import numpy as np
```

```python
from sklearn import neighbors, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

```python
email= pd.read_csv("C:\\Users\\tayea\\Downloads\\Spam.csv")
```

```python
email.head(5)
```

Out[42]:

	word_make	word_address	word_all	word_3d	word_our	word_over	word_remove	word_internet	word_order	word_mail	...	char_semicolo
0	0	1	1	0	1	0	0	0	0	0	...	
1	1	1	1	0	1	1	1	1	0	1	...	
2	1	0	1	0	1	1	1	1	1	1	...	
3	0	0	0	0	1	0	1	1	1	1	...	
4	0	0	0	0	1	0	1	1	1	1	...	

5 rows × 58 columns

```python
email.shape
```

Out[43]: (4601, 58)

```python
email.isnull().any().any()
```

Out[44]: False

```python
# Define x and y
X = email.drop(["spam"], axis = 1).values
y = email["spam"].values
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=12345, stratify = y )
knn = neighbors.KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```python
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3220, 57)
(1381, 57)
(3220,)
(1381,)
```

```python
print("\nAccuracy Score: ", accuracy_score(y_test, y_pred)*100, "\n")
print("Class_Report:\n", classification_report(y_test, y_pred), end = "")
print("\nConf_Matrix: \n",confusion_matrix(y_test, y_pred))
```

```
Accuracy Score:  80.59377262853005

Class_Report:
               precision    recall  f1-score   support

           0       0.85      0.82      0.84       837
           1       0.74      0.78      0.76       544
```

```
    accuracy                           0.81      1381
   macro avg       0.80      0.80      0.80      1381
weighted avg       0.81      0.81      0.81      1381

Conf_Matrix:
 [[686 151]
 [117 427]]
```

```
# The confusion matrix shows that the model predicted 686 true positive and 427 true negative correctly .
# And has 151 false posive and 117 false negative .
# The classification report shows that the model predicts an important mail with accuracy of 85% and 74%  for sp
# Recall shows the percentage of positive predictions relative to actual positives
# The f1-scores are closer to 1 which shows that the model is good.
# The f1-score shows that the model is able to identify 76% of spam emails and 84% of good emails i.e, for every
# emails, the model will identify 74, and for every 100 good emails, the model will identify 84.
```

```
from sklearn.model_selection import GridSearchCV
knn2 = neighbors.KNeighborsClassifier()
# create a dictionary of all values to test for different n_neighbors
param_grid = {"n_neighbors":np.arange(1, 50)}

knn_grid = GridSearchCV(knn, param_grid, cv = 5)
# fit model
knn_grid.fit(X, y)
```

Out[49]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(n_neighbors=1),
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17
,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])})
```

```
# best k
knn_grid.best_params_
```

Out[50]:
```
{'n_neighbors': 1}
```

```
# Accuracy scores at different k values
i = 0
for i in range(1, 31):
    knn = neighbors.KNeighborsClassifier(n_neighbors = i)        # The best value of k 1
    knn.fit(X_train, y_train)
    pred = knn.predict(X_test)
    z = knn.score(X_test, y_test)
    i += 0
    print("when k =", i, "Accuracy score =", round(z*100, 1), end = " ,  ")
```

```
when k = 1 Accuracy_score = 80.6 ,  when k = 2 Accuracy_score = 78.1 ,  when k = 3 Accuracy_score = 79.1 ,  when
k = 4 Accuracy_score = 78.7 , when k = 5 Accuracy_score = 77.9 , when k = 6 Accuracy_score = 78.3 , when k = 7
Accuracy_score = 78.5 , when k = 8 Accuracy_score = 77.6 , when k = 9 Accuracy_score = 77.8 , when k = 10 Accu
racy_score = 77.6 , when k = 11 Accuracy_score = 77.2 , when k = 12 Accuracy_score = 77.3 , when k = 13 Accura
cy_score = 77.0 ,  when k = 14 Accuracy_score = 76.1 ,  when k = 15 Accuracy_score = 76.2 ,  when k = 16 Accuracy
_score = 76.2 , when k = 17 Accuracy_score = 77.0 , when k = 18 Accuracy_score = 76.5 , when k = 19 Accuracy_s
core = 76.2 , when k = 20 Accuracy_score = 75.6 ,  when k = 21 Accuracy_score = 76.1 ,  when k = 22 Accuracy_sco
re = 76.0 ,  when k = 23 Accuracy_score = 75.8 ,  when k = 24 Accuracy_score = 75.2 ,  when k = 25 Accuracy_score
= 75.3 ,  when k = 26 Accuracy_score = 75.0 ,  when k = 27 Accuracy_score = 74.6 ,  when k = 28 Accuracy_score =
74.8 ,  when k = 29 Accuracy_score = 75.3 ,  when k = 30 Accuracy_score = 75.2 ,
```

```
# In this particular dataset, the best value for K is 1 because it has the best scores.
# for confusion matrix and accuracy score, and has the a larger number for true positive and true negative .
# For other values of k, the accuracy scores are lower so also the number of true positive and true negative.
```

```
# Predicting the first four rows of the dataset
knn.predict(X[:4, :])
```

Out[53]:
```
array([1, 1, 1, 0], dtype=int64)
```

```
# Predicting the last four rows of the dataset
```

```
knn.predict(X[-4:, :])
```

Out[54]:
```
array([0, 0, 0, 0], dtype=int64)
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [35]:

```
import pandas as pd
import numpy as np
```

In [36]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

In [37]:

```
email= pd.read_csv("C:\\Users\\tayea\\Downloads\\Spam.csv")
```

In [38]:

```
email.tail(3)
```

Out[38]:

	word_make	word_address	word_all	word_3d	word_our	word_over	word_remove	word_internet	word_order	word_mail	...	char_semi
4598	1	0	1	0	0	0	0	0	0	0	...	
4599	1	0	0	0	1	0	0	0	0	0	...	
4600	0	0	1	0	0	0	0	0	0	0	...	

3 rows × 58 columns

In [39]:

```
email.shape
```

Out[39]:

```
(4601, 58)
```

In [40]:

```
# Define x and y
X = email.drop(["spam"], axis = 1).values
y = email["spam"].values
```

In [42]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=12345, stratify = y)
lr = LogisticRegression(max_iter = 1000, random_state = 12345)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
y_pred
```

Out[42]:

```
array([1, 1, 1, ..., 1, 0, 0], dtype=int64)
```

In [49]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3220, 57)
(1381, 57)
(3220,)
(1381,)
```

In [43]:

```
round(accuracy_score(y_test, y_pred)*100, 2)
```

Out[43]:

```
91.89
```

In [44]:

```
print("\nAccuracy Score: ", accuracy_score(y_test, y_pred)*100, "\n")
print("\nClassRep: \n", classification_report(y_test, y_pred), end = "")
print("\nConf_Matrix: \n" ,confusion_matrix(y_test, y_pred))
```

```
Accuracy Score:  91.88993482983345


ClassRep:
              precision    recall  f1-score   support
```

```
              0       0.91      0.96      0.93       837
              1       0.93      0.86      0.89       544

       accuracy                           0.92      1381
      macro avg       0.92      0.91      0.91      1381
   weighted avg       0.92      0.92      0.92      1381

Conf_Matrix:
 [[800  37]
 [ 75 469]]
```

In [ ]:

```
# The confusion matrix shows that the model predicted 803 true positive and 469 true negative correctly.
# And has 34 false posive and 75 false negative .
# The classification report shows that the model predicts an important mail with accuracy of 85% and 74%  for sp
# Recall shows the percentage of positive predictions relative to actual positives
# The f1-scores are closer to 1 which shows that the model is good.
# The f1-score shows that the model can identify 90% of spam emails and 94% of good emails i.e, for every 100 sp
# emails, the model will identify 90, and for every 100 good emails, the model will identify 94.
```

In [45]:

```
# predicting the first four rows of the dataset
lr.predict(X[:4, :])
```

Out[45]: array([1, 1, 1, 1], dtype=int64)

In [46]:

```
# predicting the last four rows of the dataset
lr.predict(X[-4:, :])
```

Out[46]: array([0, 0, 0, 0], dtype=int64)

In [ ]:

The best model to classify the data is Logistic Regression Model (LR). Based on this data, LR has a better predictive model than that of K-Nearest Neighbors (KNN). LR did better than KNN on different metrics of evaluating a model as shown in the table below.

Logistic Regression	**K-Nearest Neighbor**
• Model accuracy score is 92%	• Model Accuracy score is 81%
• It has 803 True Positive and 469 True Negative	• It has 686 True Positive and 427 True Negative
• It predicted the first four rows of the data correctly	• It predicted the first four rows of the data correctly
• It predicted the last four rows of the data correctly	• It predicted the last four rows of the data wrongly
• For every 100 spam emails, it can identify 90	• For every 100 spam emails, it can identify 76
• F1-score is closer to 1, so the model is a good fit and can be trusted.	• F1-score is not too close to 1, so the model is unreliable.

According to the information in the table above, LR model is the better model to classify the data. For example, a business enterprise who wants to reduce the risk of falling victim to fraudulent activities emanating from spam emails will choose LR model.