```
import pandas as pd
```

```
df = pd.read_csv('/content/emails_1.csv')
```

```
df.head()
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Pre
0	Email 1	0	0	1	0	0	0	2	0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	Email 2	8	13	24	6	6	2	102	1	27	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	
2	Email 3	0	0	1	0	0	0	8	0	0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	Email 4	0	5	22	0	5	1	51	2	10	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	Email 5	7	6	17	1	5	2	57	0	9	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	

5 rows × 3002 columns

```
#input data and op sepeatae
#inputs others col are imp

x = df.drop(['Email No.','Prediction'], axis = 1)

#Output data
y = df['Prediction']
```

```
x.shape
```

```
(518, 3000)
```

```
y.shape
```

```
(518,)
```

```
x.dtypes
```

	0
the	int64
to	int64
ect	int64
and	int64
for	int64
...	...
infrastructure	float64
military	float64
allowing	float64
ff	float64
dry	float64

3000 rows × 1 columns

dtype: object

```
df.shape
```

```
(518, 3002)
```

Start coding or generate with AI.

```
set(x.dtypes)
```

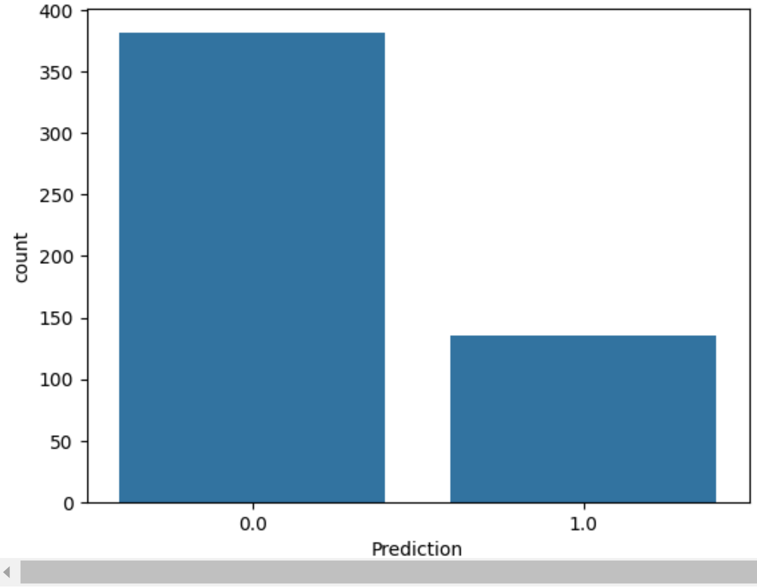
```
{dtype('int64'), dtype('float64')}
```

```
#count of spam-1 and notspm- taking the counts

import seaborn as sns

sns.countplot(x = y)
```

```
<Axes: xlabel='Prediction', ylabel='count'>
```



```
y.value_counts()
```

```
              count
Prediction
   0.0          382
   1.0          135

dtype: int64
```

```
#features scaling-
#after scaling all the features get gathered into 1 range only

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)
```

```
x_scaled
```

```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.03809524, 0.11711712, 0.25274725, ..., 0.        , 0.00877193,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.17142857, 0.26126126, 0.07692308, ..., 0.        , 0.00877193,
        0.        ],
       [0.00952381, 0.02702703, 0.01098901, ..., 0.        , 0.00877193,
        0.        ],
       [0.07142857, 0.06306306, 0.        , ...,        nan,        nan,
               nan]])
```

```
#data separate for cross validations

from sklearn.model_selection import train_test_split
#75 train data 25 test data

x_train,x_test,y_train,y_test = train_test_split(
    x_scaled,y,random_state=0,test_size=0.25
)
```

```
x_scaled.shape
```

```
(518, 3000)
```

```
x_train.shape
```

```
(388, 3000)
```

```
x_test.shape
```

```
(130, 3000)
```

```
#apply the ml algos
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
# Create the Object
```

```
knn = KNeighborsClassifier(n_neighbors=5)
```

```
#train the algos
```

```
knn.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-22-0954a299d5eb> in <cell line: 3>()
      1 #train the algos
      2 
----> 3 knn.fit(x_train,y_train)

                        7 frames

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _assert_all_finite_element_wise(X, xp, allow_nan, msg_dtype,
estimator_name, input_name)
    170                 "#estimators-that-handle-nan-values"
    171             )
--> 172         raise ValueError(msg_err)
    173 
    174 

ValueError: Input X contains NaN.
KNeighborsClassifier does not accept missing values encoded as NaN natively. For supervised learning, you might want to consider
sklearn.ensemble.HistGradientBoostingClassifier and Regressor which accept missing values encoded as NaNs natively. Alternatively,
it is possible to preprocess the data, for instance by using an imputer transformer in a pipeline or drop samples with missing
values. See https://scikit-learn.org/stable/modules/impute.html You can find a list of all estimators that handle NaN values at the
following page: https://scikit-learn.org/stable/modules/impute.html#estimators-that-handle-nan-values
```

Next steps: Explain error

```
#predict on test data
```

```
y_pred = knn.predict(x_test)
```

```
y_pred
```

```
# to evaluate the performance of both the models
#import evaluation metrics
```

```
from sklearn.metrics import ConfusionMatrixDisplay,accuracy_score
```

```
from sklearn.metrics import classification_report
```

```
#Original and the predicted data
```

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

```
y_test.value_counts()
```

```
accuracy_score(y_test,y_pred)
```

```
print(classification_report(y_test,y_pred))
```