```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.utils import resample
from sklearn import metrics
from tqdm.notebook import tqdm
from tqdm.gui import tqdm
from tqdm import tqdm_notebook
%matplotlib inline
warnings.filterwarnings("ignore")

df = pd.read_csv("emails.csv")

df.head()
```

```
  Email No.  the  to  ect  and  for  of    a  you  hou  ...  connevey
jay  \
0   Email 1    0   0    1    0    0   0    2    0    0  ...         0
0
1   Email 2    8  13   24    6    6   2  102    1   27  ...         0
0
2   Email 3    0   0    1    0    0   0    8    0    0  ...         0
0
3   Email 4    0   5   22    0    5   1   51    2   10  ...         0
0
4   Email 5    7   6   17    1    5   2   57    0    9  ...         0
0

   valued  lay  infrastructure  military  allowing  ff  dry
Prediction
0       0    0               0         0         0   0    0
0
1       0    0               0         0         0   1    0
0
2       0    0               0         0         0   0    0
0
3       0    0               0         0         0   0    0
0
4       0    0               0         0         0   1    0
0

[5 rows x 3002 columns]
```

```
df.shape
```

```
(5172, 3002)
```

```
df.describe().T
```

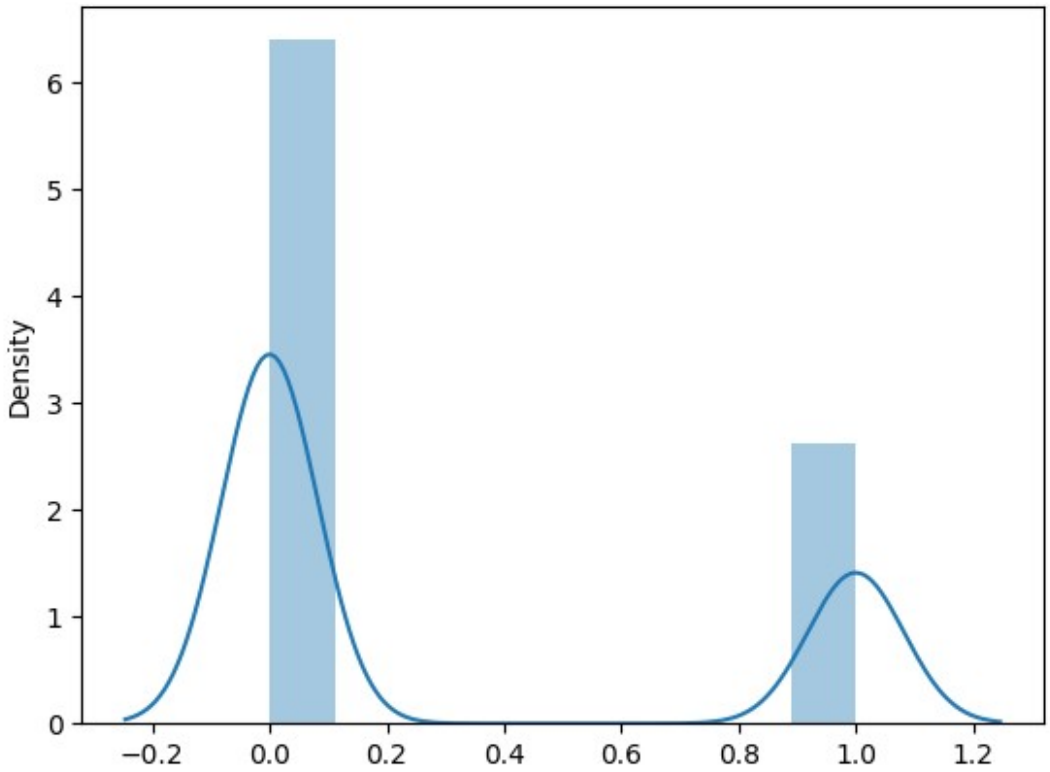
```
             count      mean         std  min  25%  50%  75%    max
the         5172.0  6.640565   11.745009  0.0  0.0  3.0  8.0  210.0
to          5172.0  6.188128    9.534576  0.0  1.0  3.0  7.0  132.0
ect         5172.0  5.143852   14.101142  1.0  1.0  1.0  4.0  344.0
and         5172.0  3.075599    6.045970  0.0  0.0  1.0  3.0   89.0
for         5172.0  3.124710    4.680522  0.0  1.0  2.0  4.0   47.0
...            ...       ...         ...  ...  ...  ...  ...    ...
military    5172.0  0.006574    0.138908  0.0  0.0  0.0  0.0    4.0
allowing    5172.0  0.004060    0.072145  0.0  0.0  0.0  0.0    3.0
ff          5172.0  0.914733    2.780203  0.0  0.0  0.0  1.0  114.0
dry         5172.0  0.006961    0.098086  0.0  0.0  0.0  0.0    4.0
Prediction  5172.0  0.290023    0.453817  0.0  0.0  0.0  1.0    1.0

[3001 rows x 8 columns]
```

```
df = df.drop("Email No.", axis=1)
```

```
df.isna().sum()
```

```
the           0
to            0
ect           0
and           0
for           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3001, dtype: int64
```

```
sns.distplot(x=df["Prediction"])
plt.show()
```

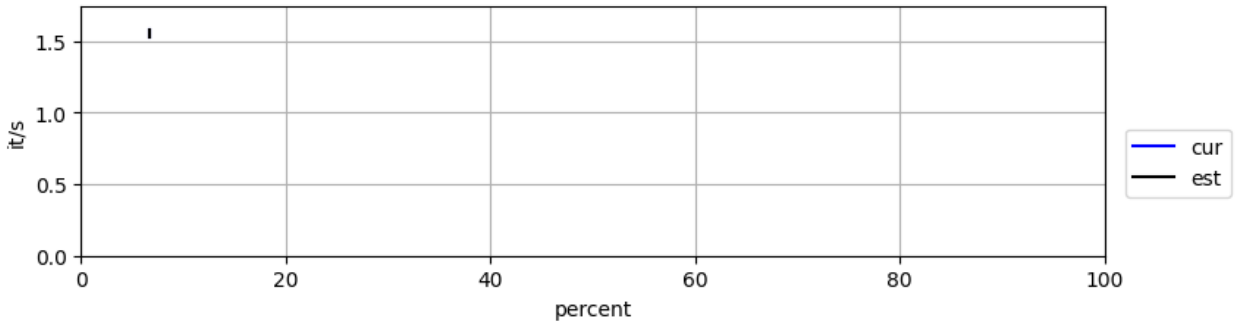



```
x = df.drop("Prediction", axis=1)
y = df[["Prediction"]]

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2)

k_values = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
accuracy_values = []

for k in tqdm(k_values):  # Use k directly from k_values
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    accuracy = metrics.accuracy_score(y_test, y_pred)
    accuracy_values.append(accuracy)
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
File c:\Program Files\Python312\Lib\site-packages\tqdm\std.py:1191, in
tqdm.__iter__(self)
   1190 if dt >= mininterval and cur_t >= min_start_t:
-> 1191     self.update(n - last_print_n)
   1192     last_print_n = self.last_print_n
```

```
File c:\Program Files\Python312\Lib\site-packages\tqdm\std.py:1242, in
tqdm.update(self, n)
   1241     self._ema_dt(dt)
-> 1242 self.refresh(lock_args=self.lock_args)
   1243 if self.dynamic_miniters:
   1244     # If no `miniters` was specified, adjust automatically to
the
   1245     # maximum iteration rate seen so far between two prints.
   1246     # e.g.: After running `tqdm.update(5)`, subsequent
   1247     # calls to `tqdm.update()` will only cause an update after
   1248     # at least 5 more iterations.

File c:\Program Files\Python312\Lib\site-packages\tqdm\std.py:1347, in
tqdm.refresh(self, nolock, lock_args)
   1346         self._lock.acquire()
-> 1347 self.display()
   1348 if not nolock:

File c:\Program Files\Python312\Lib\site-packages\tqdm\gui.py:156, in
tqdm_gui.display(self, *_, **__)
    155     poly_lims = self.hspan.get_xy()
--> 156 poly_lims[0, 1] = ymin
    157 poly_lims[1, 1] = ymax

TypeError: 'tuple' object does not support item assignment

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call
last)
Cell In[12], line 1
----> 1 for k in tqdm(k_values):  # Use k directly from k_values
      2     model = KNeighborsClassifier(n_neighbors=k)
      3     model.fit(x_train, y_train)

File c:\Program Files\Python312\Lib\site-packages\tqdm\std.py:1196, in
tqdm.__iter__(self)
   1194 finally:
   1195     self.n = n
-> 1196     self.close()

File c:\Program Files\Python312\Lib\site-packages\tqdm\gui.py:103, in
tqdm_gui.close(self)
    101     self.plt.ioff()
    102 if self.leave:
--> 103     self.display()
    104 else:
    105     self.plt.close(self.fig)

File c:\Program Files\Python312\Lib\site-packages\tqdm\gui.py:156, in
```

```
tqdm_gui.display(self, *_, **__)
    154     self.hspan = self.plt.axhspan(0, 0.001, xmin=0, xmax=0,
color='g')
    155     poly_lims = self.hspan.get_xy()
--> 156 poly_lims[0, 1] = ymin
    157 poly_lims[1, 1] = ymax
    158 poly_lims[2] = [n / total, ymax]

TypeError: 'tuple' object does not support item assignment
```




```
px.line(x=k_values, y=accuracy_values)

---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[43], line 1
----> 1 px.line(x=k_values, y=accuracy_values)

File c:\Program Files\Python312\Lib\site-packages\plotly\express\
_chart_types.py:264, in line(data_frame, x, y, line_group, color,
line_dash, symbol, hover_name, hover_data, custom_data, text,
facet_row, facet_col, facet_col_wrap, facet_row_spacing,
facet_col_spacing, error_x, error_x_minus, error_y, error_y_minus,
animation_frame, animation_group, category_orders, labels,
orientation, color_discrete_sequence, color_discrete_map,
line_dash_sequence, line_dash_map, symbol_sequence, symbol_map,
markers, log_x, log_y, range_x, range_y, line_shape, render_mode,
title, template, width, height)
    216 def line(
    217     data_frame=None,
    218     x=None,
   (...)
    258     height=None,
    259 ) -> go.Figure:
    260     """
    261     In a 2D line plot, each row of `data_frame` is represented
as vertex of
```

```
    262     a polyline mark in 2D space.
    263     """
--> 264     return make_figure(args=locals(), constructor=go.Scatter)

File c:\Program Files\Python312\Lib\site-packages\plotly\express\
_core.py:2117, in make_figure(args, constructor, trace_patch,
layout_patch)
   2114 layout_patch = layout_patch or {}
   2115 apply_default_cascade(args)
-> 2117 args = build_dataframe(args, constructor)
   2118 if constructor in [go.Treemap, go.Sunburst, go.Icicle] and
args["path"] is not None:
   2119     args = process_dataframe_hierarchy(args)

File c:\Program Files\Python312\Lib\site-packages\plotly\express\
_core.py:1513, in build_dataframe(args, constructor)
   1510     args["color"] = None
   1511 # now that things have been prepped, we do the systematic
rewriting of `args`
-> 1513 df_output, wide_id_vars = process_args_into_dataframe(
   1514     args, wide_mode, var_name, value_name
   1515 )
   1517 # now that `df_output` exists and `args` contains only
references, we complete
   1518 # the special-case and wide-mode handling by further rewriting
args and/or mutating
   1519 # df_output
   1521 count_name = _escape_col_name(df_output, "count", [var_name,
value_name])

File c:\Program Files\Python312\Lib\site-packages\plotly\express\
_core.py:1274, in process_args_into_dataframe(args, wide_mode,
var_name, value_name)
   1271         col_name = _check_name_not_reserved(field,
reserved_names)
   1273     if length and len(argument) != length:
-> 1274         raise ValueError(
   1275             "All arguments should have the same length. "
   1276             "The length of argument `%s` is %d, whereas the "
   1277             "length of  previously-processed arguments %s is
%d"
   1278             % (field, len(argument),
str(list(df_output.keys())), length)
   1279         )
   1280     df_output[str(col_name)] = to_unindexed_series(argument,
str(col_name))
   1282 # Finally, update argument with column name now that column
exists

ValueError: All arguments should have the same length. The length of
```

```
argument `y` is 4, whereas the length of  previously-processed
arguments ['x'] is 15
```

```
optimal_k =-1
optimal_accuracy =-1
for i in list(zip(k_values, accuracy_values)):
    if i[1] > optimal_accuracy:
        optimal_k = i[0]
        optimal_accuracy = i[1]
```