```
# pip install -U kaleido
```

```
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 8.4 MB/s eta 0:00:00
Installing collected packages: kaleido
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
lida 0.0.10 requires uvicorn, which is not installed.
Successfully installed kaleido-0.2.1
```

```
# Import libraries

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go

from sklearn.neighbors import KNeighborsClassifier
from sklearn.semi_supervised import SelfTrainingClassifier

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, balanced_accuracy_score, classification_report

import warnings
warnings.filterwarnings('ignore')
```

Load Dataset

```
data_df = pd.read_excel('data.xlsx')
print(f'Shape = {data_df.shape} \n')
data_df.head()
```

```
Shape = (2000, 6)
```

	Cancer stage	Clump thickness	No of week	Clump thickness_new	No of week_new	True cancer stage
0	1.0	10.510076	6.166544	10.269649	11.999203	1
1	1.0	11.739776	7.024066	10.494287	6.495638	1
2	1.0	7.857070	5.909366	8.516879	7.102108	1
3	1.0	10.817929	5.920890	8.979736	9.196251	1
4	1.0	10.302407	6.984937	9.553005	7.120283	1

```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Cancer stage         200 non-null    float64
 1   Clump thickness      200 non-null    float64
 2   No of week           200 non-null    float64
 3   Clump thickness_new  2000 non-null   float64
 4   No of week_new       2000 non-null   float64
 5   True cancer stage    2000 non-null   int64  
dtypes: float64(5), int64(1)
memory usage: 93.9 KB
```

```
data_df.iloc[198:202, :]
```

	Cancer stage	Clump thickness	No of week	Clump thickness_new	No of week_new	True cancer stage
198	4.0	4.594556	12.908754	10.682964	5.663148	1
199	4.0	4.076195	13.541567	11.170609	6.526154	1
200	NaN	NaN	NaN	10.475861	6.958337	1
201	NaN	NaN	NaN	11.412233	6.384493	1

```
data_df['Cancer stage'].unique()
```

```
array([ 1.,  2.,  3.,  4., nan])
```

```
data_df['True cancer stage'].unique()
```

```
array([1, 2, 3, 4])
```

```
data_df['Cancer stage'] = data_df['Cancer stage'].astype('category')
#data_df['True cancer stage'] = data_df['True cancer stage'].astype('category')
data_df.dtypes
```

```
Cancer stage          category
Clump thickness        float64
No of week             float64
Clump thickness_new    float64
No of week_new         float64
```

```
True cancer stage         int64
dtype: object
```

Show a scatter plot of ‘Clump thickness vs No of week’ and plot their class labels given in column ‘Cancer stage’ (1, 2, 3, 4) for all 200 datapoints.

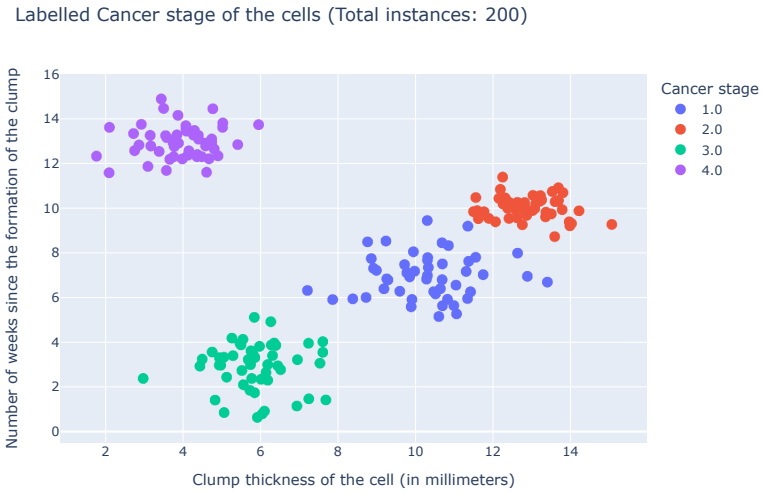
```
num_points = len(data_df['Clump thickness'].dropna())
num_points
```

```
200
```

```
title_str = f'Labelled Cancer stage of the cells (Total instances: {num_points})'
x_axis_str = 'Clump thickness of the cell (in millimeters)'
y_axis_str = 'Number of weeks since the formation of the clump'

fig = px.scatter(data_df, x = 'Clump thickness', y = 'No of week', color = 'Cancer stage',
                          width = 750,
                          title = title_str,
                          labels = {'Clump thickness' : x_axis_str, 'No of week' : y_axis_str})

fig.update_traces(marker=dict(size = 10))
fig.show()
fig.write_image('Labelled Cancer stage of the cells.png')
```



Plot the other added 2000 datapoints given in columns ‘Clump thickness_new’ and ‘No of week_new’ over the previous scatter plot without using their class label.

```
num_added_points = len(data_df['Clump thickness_new'].dropna())
num_added_points
```

```
2000
```

```
title_str = f'Non-labelled Cancer stage of the cells (New instances added: {num_added_points})'
x_axis_str = 'Clump thickness of the cell (in millimeters)'
y_axis_str = 'Number of weeks since the formation of the clump'

fig = px.scatter(data_df, x = 'Clump thickness', y = 'No of week', color = 'Cancer stage',
                          width = 750,
                          title = title_str,
                          labels = {'Clump thickness' : x_axis_str, 'No of week' : y_axis_str})

fig.update_traces(marker=dict(size = 10))


fig.add_trace(go.Scattergl(x = data_df['Clump thickness_new'], y = data_df['No of week_new'], mode = 'markers',
               marker=dict(size=2, color = 'gray'),
               name = 'Unlabelled data'))
fig.show()
fig.write_image('Non-labelled Cancer stage of the cells.png')
```

Now train the semi-supervised model for the labeled 200 datapoints and make predictions for class label of new 2000 added datapoints and plot the scatters. Use KNN for base estimator.

```
labelled_df = data_df[['Clump thickness', 'No of week', 'Cancer stage']]
labelled_df.dropna(inplace = True)
print(f'Shape = {labelled_df.shape} \n')
labelled_df.head()
```

```
Shape = (200, 3)
```

	Clump thickness	No of week	Cancer stage
0	10.510076	6.166544	1.0
1	11.739776	7.024066	1.0
2	7.857070	5.909366	1.0
3	10.817929	5.920890	1.0
4	10.302407	6.984937	1.0

```
unlabelled_df = data_df[['Clump thickness_new', 'No of week_new']]
unlabelled_df.rename(lambda col: col.split('_')[0], axis='columns', inplace = True)
print(f'Shape = {unlabelled_df.shape} \n')
unlabelled_df.head()
```

```
Shape = (2000, 2)
```

	Clump thickness	No of week
0	10.269649	11.999203
1	10.494287	6.495638
2	8.516879	7.102108
3	8.979736	9.196251
4	9.553005	7.120283

```
labelled_df_X = labelled_df.drop('Cancer stage', axis = 1)
print(f'Shape = {labelled_df_X.shape} \n')
labelled_df_X.head()
```

```
Shape = (200, 2)
```

	Clump thickness	No of week
0	10.510076	6.166544
1	11.739776	7.024066
2	7.857070	5.909366
3	10.817929	5.920890
4	10.302407	6.984937

```
labelled_df_y = labelled_df['Cancer stage']
print(f'Shape = {labelled_df_y.shape} \n')
labelled_df_y.head()
```

```
Shape = (200,)

0    1.0
1    1.0
2    1.0
3    1.0
4    1.0
Name: Cancer stage, dtype: category
Categories (4, float64): [1.0, 2.0, 3.0, 4.0]
```

```
knn_clf = KNeighborsClassifier(n_neighbors = 5)
```

```
self_training_model = SelfTrainingClassifier(knn_clf)
```

```
self_training_model.fit(labelled_df_X, labelled_df_y)
```

SelfTrainingClassifier
base_estimator: KNeighborsClassifier
KNeighborsClassifier

```
y_pred = self_training_model.predict(unlabelled_df)
```

```
y_pred
```

```
array([2., 1., 1., ..., 4., 4., 4.])
```

```
type(y_pred)
```

```
numpy.ndarray
```

```
y_pred.dtype
```

```
dtype('float64')
```

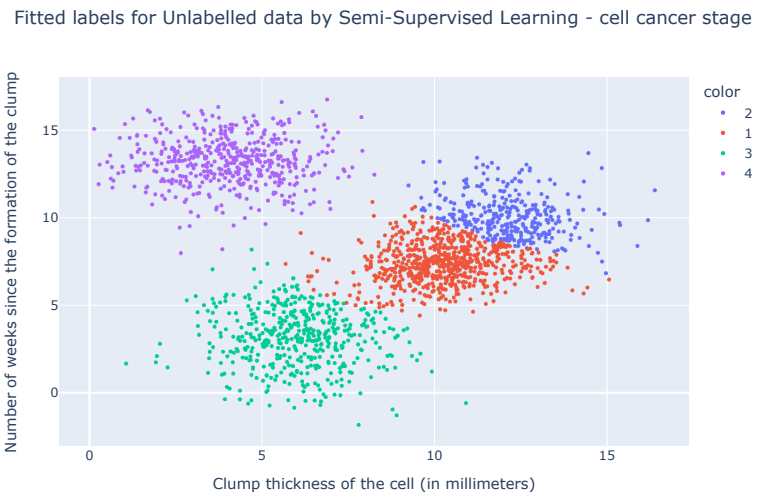
```
y_pred = y_pred.astype('int')
y_pred = pd.Series(y_pred, name = 'Cancer stage', dtype = 'category')
y_pred.head()
```

```
0    2
1    1
2    1
3    1
4    1
Name: Cancer stage, dtype: category
Categories (4, int64): [1, 2, 3, 4]
```

```
title_str = f'Fitted labels for Unlabelled data by Semi-Supervised Learning - cell cancer stage'
x_axis_str = 'Clump thickness of the cell (in millimeters)'
y_axis_str = 'Number of weeks since the formation of the clump'

fig = px.scatter(unlabelled_df, x = 'Clump thickness', y = 'No of week', color = y_pred,
           width = 750,
           title = title_str,
          labels = {'Clump thickness' : x_axis_str, 'No of week' : y_axis_str})
fig.update_traces(marker=dict(size = 4))

fig.show()
fig.write_image('Fitted labels for Unlabelled data by Semi-Supervised Learning - cell cancer stage.png')
```



Compute and print accuracy score, plot classification report and the confusion matrix.

```
true_y = data_df['True cancer stage']
print(f'Shape = {true_y.shape} \n')
true_y.head()
```

```
Shape = (2000,)

0    1
1    1
2    1
3    1
4    1
Name: True cancer stage, dtype: int64
```

```
cf_matrix = confusion_matrix(true_y, y_pred)      # Index = Actual;  Column = Predicted
```

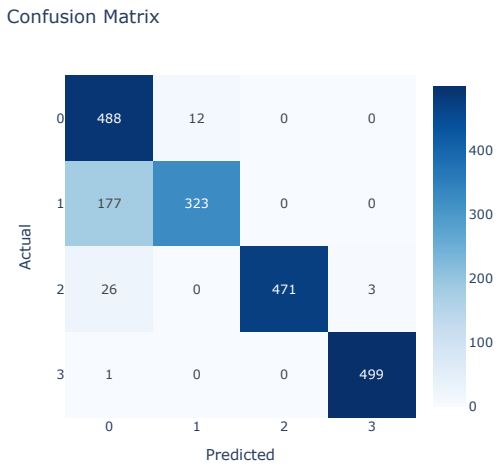
```
cf_matrix
```

```
array([[488,  12,   0,   0],
       [177, 323,   0,   0],
       [ 26,   0, 471,   3],
       [  1,   0,   0, 499]])
```

```
fig = px.imshow(cf_matrix, text_auto = True, aspect = 'auto', color_continuous_scale = 'blues', width=500, height=500,
                title = 'Confusion Matrix',
                labels = dict(x = 'Predicted', y = 'Actual'))

fig.update_layout( yaxis={'tickvals': [*range(int(min(true_y) - 1), int(max(true_y)))]
})


fig.show()
fig.write_image('Confusion Matrix.png')
```




```
# To compute accuracy score

accuracy = accuracy_score(true_y, y_pred)
accuracy
```

```
0.8905
```

```
# To compute balanced accuracy score

balanced_accuracy = balanced_accuracy_score(true_y, y_pred)
balanced_accuracy
```

```
0.8905000000000001
```

```
print(classification_report(true_y, y_pred))
```

```
                   precision    recall  f1-score   support

               1       0.71      0.98      0.82       500
               2       0.96      0.65      0.77       500
               3       1.00      0.94      0.97       500
               4       0.99      1.00      1.00       500

        accuracy                           0.89      2000
       macro avg       0.92      0.89      0.89      2000
    weighted avg       0.92      0.89      0.89      2000
```