svm

November 25, 2024

[1]:
```
import pandas as pd
df = pd.read_csv('/content/mushroom.csv')
```

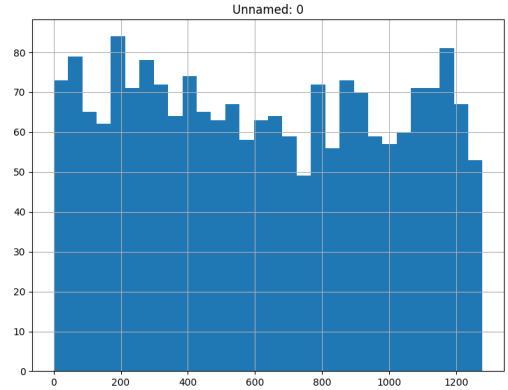
[3]:
```
print(df.head())
print(df.info())
print(df.describe())
```

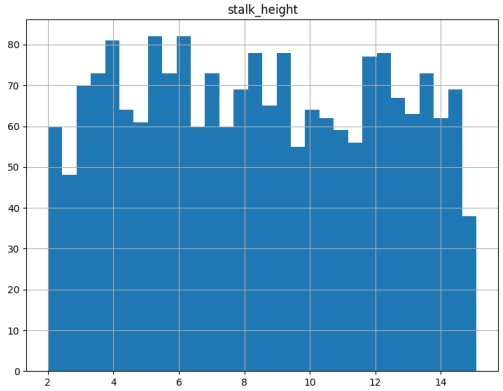
```
   Unnamed: 0 cap_shape cap_surface cap_color bruises   odor gill_attachment  \
0        1167    sunken       scaly     white      no  anise      descending
1        1037    sunken     fibrous       red      no  anise         notched
2         309      flat     grooves    purple     yes   foul      descending
3         282      bell       scaly      pink     yes  fishy         notched
4         820      flat      smooth    yellow     yes  musty            free

  gill_spacing gill_size gill_color  …  veil_type veil_color ring_number  \
0      distant     broad       pink  …    partial      brown         two
1      crowded    narrow  chocolate  …  universal      brown         two
2      crowded     broad     purple  …  universal     yellow         two
3        close     broad     orange  …    partial     yellow         two
4      crowded    narrow     orange  …  universal      white        none

   ring_type spore_print_color population  habitat      class stalk_height  \
0  sheathing         chocolate  clustered    waste  poisonous    14.276173
1  sheathing             brown   numerous    waste     edible     3.952715
2  sheathing            purple   abundant    waste  poisonous     9.054265
3   cobwebby             green  clustered  grasses  poisonous     5.226499
4       none            yellow  clustered    urban  poisonous    14.037532

  cap_diameter
0     5.054983
1    19.068319
2     7.205884
3    20.932692
4    12.545245

[5 rows x 26 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
```

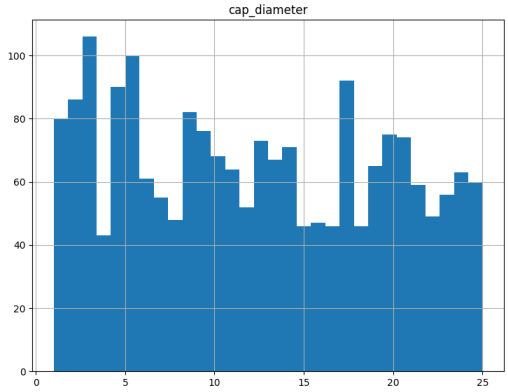
```
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                2000 non-null   int64
 1   cap_shape                 2000 non-null   object
 2   cap_surface               2000 non-null   object
 3   cap_color                 2000 non-null   object
 4   bruises                   2000 non-null   object
 5   odor                      2000 non-null   object
 6   gill_attachment           2000 non-null   object
 7   gill_spacing              2000 non-null   object
 8   gill_size                 2000 non-null   object
 9   gill_color                2000 non-null   object
 10  stalk_shape               2000 non-null   object
 11  stalk_root                2000 non-null   object
 12  stalk_surface_above_ring  2000 non-null   object
 13  stalk_surface_below_ring  2000 non-null   object
 14  stalk_color_above_ring    2000 non-null   object
 15  stalk_color_below_ring    2000 non-null   object
 16  veil_type                 2000 non-null   object
 17  veil_color                2000 non-null   object
 18  ring_number               2000 non-null   object
 19  ring_type                 2000 non-null   object
 20  spore_print_color         2000 non-null   object
 21  population                2000 non-null   object
 22  habitat                   2000 non-null   object
 23  class                     2000 non-null   object
 24  stalk_height              2000 non-null   float64
 25  cap_diameter              2000 non-null   float64
dtypes: float64(2), int64(1), object(23)
memory usage: 406.4+ KB
None
        Unnamed: 0  stalk_height  cap_diameter
count  2000.000000   2000.000000   2000.000000
mean    624.974000      8.449118     12.314345
std     375.091938      3.697217      7.048845
min       0.000000      2.000000      1.000000
25%     290.000000      5.291009      5.723521
50%     607.000000      8.318596     12.124902
75%     957.250000     11.781272     18.698605
max    1279.000000     15.095066     25.000054
```

[4]:
```
df.hist(bins=30, figsize=(20, 15))
```

[4]:
```
array([[<Axes: title={'center': 'Unnamed: 0'}>,
        <Axes: title={'center': 'stalk_height'}>],
       [<Axes: title={'center': 'cap_diameter'}>, <Axes: >]], dtype=object)
```

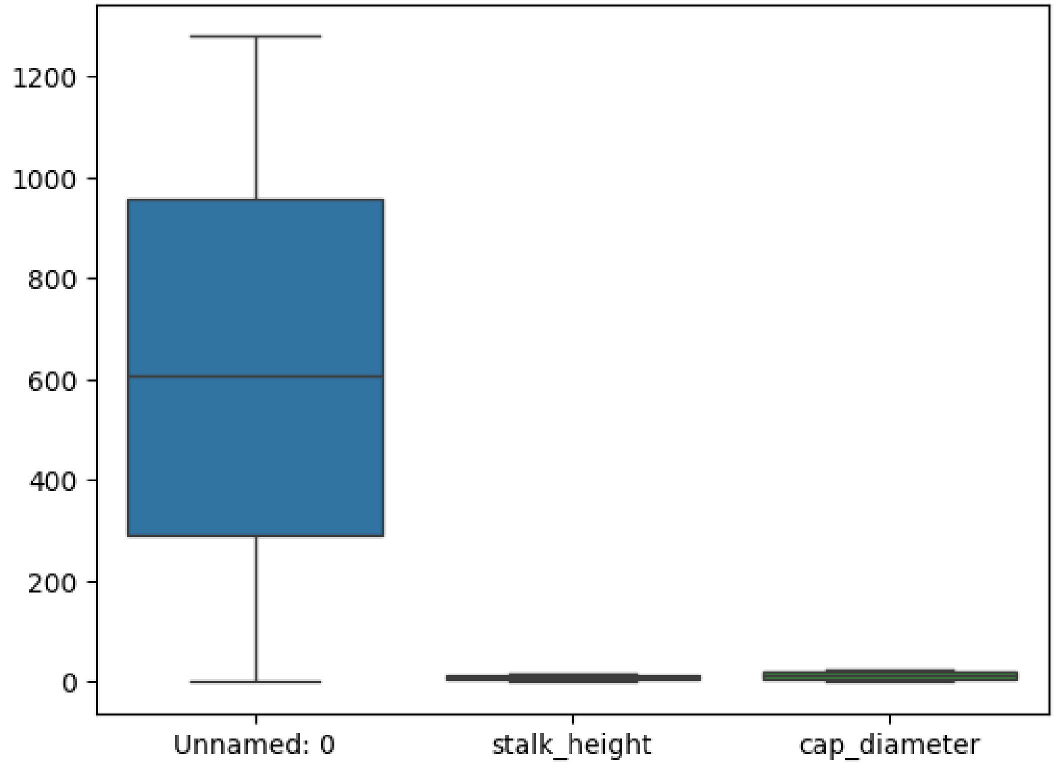






[5]:
```
import seaborn as sns
sns.boxplot(data=df)
```

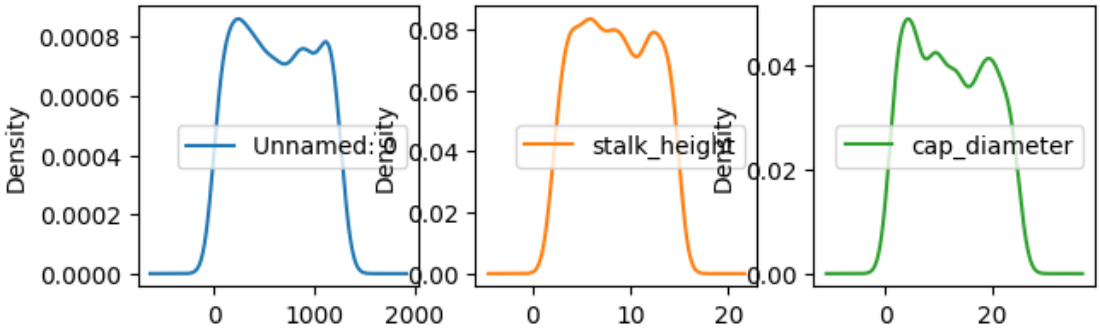
[5]: <Axes: >



[6]:
```
df.plot(kind='density', subplots=True, layout=(8, 8), sharex=False,␣
↪figsize=(20, 20))
```

[6]:
```
array([[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
```

```
 <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
 <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
 <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
 <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
 <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
 <Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
 <Axes: ylabel='Density'>, <Axes: ylabel='Density'>]], dtype=object)
```



[11]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

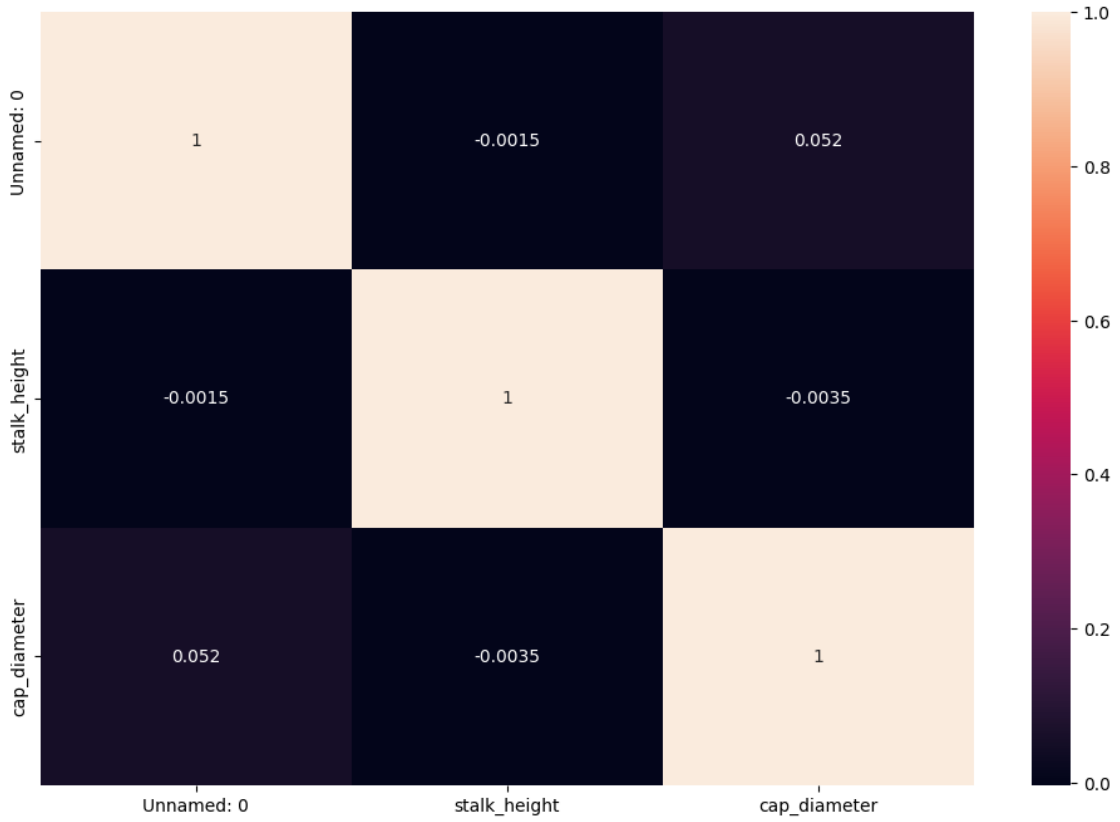
[12]:
```
print(df.isnull().sum())
```

```
Unnamed: 0                0
cap_shape              2000
cap_surface            2000
cap_color                 0
bruises                   0
odor                      0
gill_attachment           0
gill_spacing              0
gill_size                 0
gill_color                0
stalk_shape               0
stalk_root                0
stalk_surface_above_ring  0
stalk_surface_below_ring  0
```

```
stalk_color_above_ring         0
stalk_color_below_ring         0
veil_type                      0
veil_color                     0
ring_number                    0
ring_type                      0
spore_print_color              0
population                     0
habitat                        0
class                          0
stalk_height                   0
cap_diameter                   0
dtype: int64
```

[14]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np # Import numpy with the alias np

df_numeric = df.select_dtypes(include=[np.number])
corr_matrix = df_numeric.corr()
```

[15]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Load your dataset
df = pd.read_csv('/content/mushroom.csv')

# Select only numeric columns if necessary
df_numeric = df.select_dtypes(include=[np.number])

# Calculate the correlation matrix
corr_matrix = df_numeric.corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True)
plt.show()
```



[16]:
```
from sklearn.preprocessing import LabelEncoder
df = df.apply(LabelEncoder().fit_transform)
```

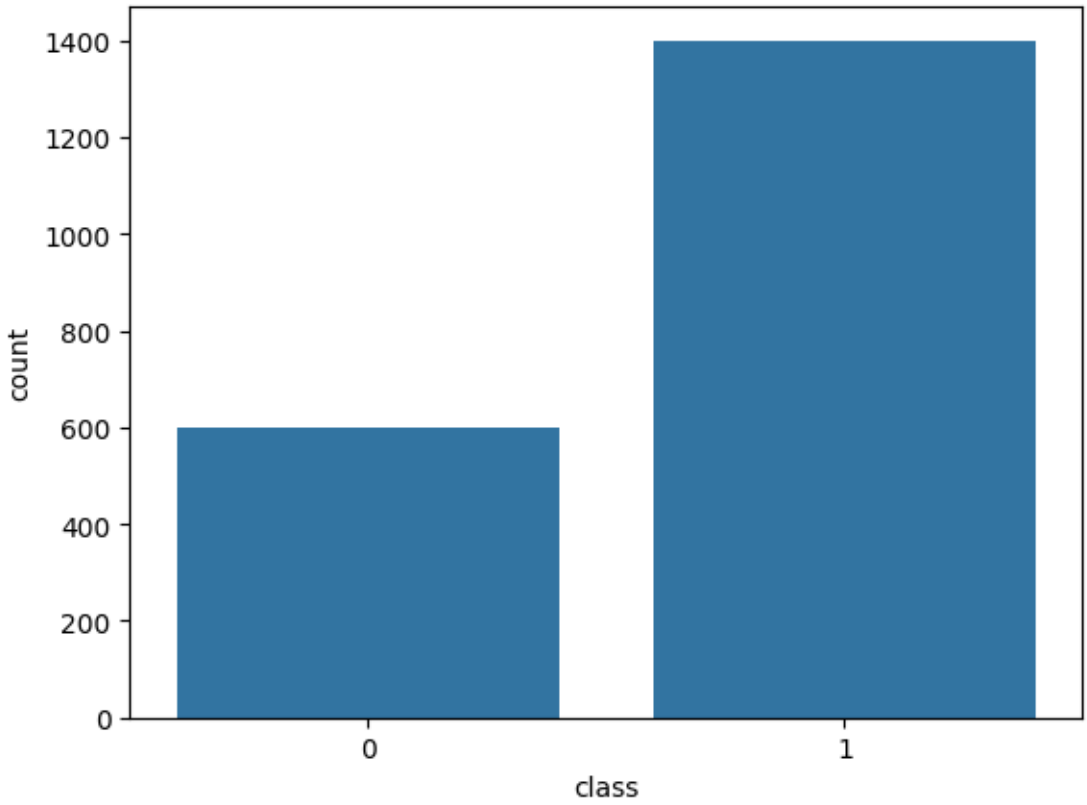
[18]:
```
from sklearn.model_selection import train_test_split

# Replace 'class' with the actual target column name if it's different
X = df.drop('class', axis=1)
y = df['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[20]:
```
sns.pairplot(df, hue='class')
```

Output hidden; open in https://colab.research.google.com to view.

[21]:
```
sns.countplot(x='class', data=df)
```

[21]:
```
<Axes: xlabel='class', ylabel='count'>
```



[22]:
```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train, y_train)
```

[22]:
```
SVC()
```

[23]:
```
model.fit(X_train, y_train)
```

[23]:
```
SVC()
```

[24]:
```
from sklearn.metrics import classification_report
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       133
           1       0.67      1.00      0.80       267

    accuracy                           0.67       400
   macro avg       0.33      0.50      0.40       400
```

```
weighted avg       0.45      0.67      0.53       400
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

[25]:
```python
# For 2D visualization, we can plot decision boundaries, for example
```

[26]:
```python
model = SVC(kernel='linear', C=1.0)
model.fit(X_train, y_train)
```

[26]:
```
SVC(kernel='linear')
```

[27]:
```python
kernels = ['linear', 'poly', 'rbf']
for kernel in kernels:
    model = SVC(kernel=kernel)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Kernel: {kernel}")
    print(classification_report(y_test, y_pred))
```

```
Kernel: linear
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       133
           1       0.67      1.00      0.80       267

    accuracy                           0.67       400
   macro avg       0.33      0.50      0.40       400
weighted avg       0.45      0.67      0.53       400
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
```

```
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Kernel: poly
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       133
           1       0.67      1.00      0.80       267

    accuracy                           0.67       400
   macro avg       0.33      0.50      0.40       400
weighted avg       0.45      0.67      0.53       400
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Kernel: rbf
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       133
           1       0.67      1.00      0.80       267

    accuracy                           0.67       400
   macro avg       0.33      0.50      0.40       400
weighted avg       0.45      0.67      0.53       400
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

[ ]: