

✖ Importing Libraries

```
import pandas as pd
from sklearn.datasets import load_breast_cancer, load_iris, load_wine
from sklearn.preprocessing import StandardScaler
```

✖ Loading Datasets

```
bc_data = load_breast_cancer()
i_data = load_iris()
w_data = load_wine()
```

```
bcd = pd.DataFrame(bc_data.data, columns=bc_data.feature_names)
bcd["target"] = bc_data.target
idf = pd.DataFrame(i_data.data, columns=i_data.feature_names)
idf["target"] = i_data.target
wdf = pd.DataFrame(w_data.data, columns=w_data.feature_names)
wdf["target"] = w_data.target
```

✖ Breast Cancer Data

bcd

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | ... | wor textu |
|-----|----------------|-----------------|-------------------|--------------|--------------------|---------------------|-------------------|---------------------------|------------------|------------------------------|-----|--------------|
| 0 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.30010 | 0.14710 | 0.2419 | 0.07871 | ... | 17. |
| 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.08690 | 0.07017 | 0.1812 | 0.05667 | ... | 23. |
| 2 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.19740 | 0.12790 | 0.2069 | 0.05999 | ... | 25. |
| 3 | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.24140 | 0.10520 | 0.2597 | 0.09744 | ... | 26. |
| 4 | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.19800 | 0.10430 | 0.1809 | 0.05883 | ... | 16. |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 564 | 21.56 | 22.39 | 142.00 | 1479.0 | 0.11100 | 0.11590 | 0.24390 | 0.13890 | 0.1726 | 0.05623 | ... | 26. |
| 565 | 20.13 | 28.25 | 131.20 | 1261.0 | 0.09780 | 0.10340 | 0.14400 | 0.09791 | 0.1752 | 0.05533 | ... | 38. |
| 566 | 16.60 | 28.08 | 108.30 | 858.1 | 0.08455 | 0.10230 | 0.09251 | 0.05302 | 0.1590 | 0.05648 | ... | 34. |
| 567 | 20.60 | 29.33 | 140.10 | 1265.0 | 0.11780 | 0.27700 | 0.35140 | 0.15200 | 0.2397 | 0.07016 | ... | 39. |
| 568 | 7.76 | 24.54 | 47.92 | 181.0 | 0.05263 | 0.04362 | 0.00000 | 0.00000 | 0.1587 | 0.05884 | ... | 30. |

569 rows × 31 columns

```
print(bcd.info())
bcd.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   mean radius                           569 non-null    float64
1   mean texture                           569 non-null    float64
2   mean perimeter                         569 non-null    float64
3   mean area                             569 non-null    float64
4   mean smoothness                       569 non-null    float64
5   mean compactness                      569 non-null    float64
6   mean concavity                        569 non-null    float64
7   mean concave points                   569 non-null    float64
8   mean symmetry                         569 non-null    float64
9   mean fractal dimension                 569 non-null    float64
10  radius error                           569 non-null    float64
11  texture error                          569 non-null    float64
12  perimeter error                       569 non-null    float64
13  area error                            569 non-null    float64
14  smoothness error                      569 non-null    float64
15  compactness error                     569 non-null    float64
16  concavity error                       569 non-null    float64
17  concave points error                  569 non-null    float64
18  symmetry error                        569 non-null    float64
19  fractal dimension error               569 non-null    float64
20  worst radius                          569 non-null    float64
21  worst texture                         569 non-null    float64
22  worst perimeter                       569 non-null    float64
23  worst area                            569 non-null    float64
24  worst smoothness                      569 non-null    float64
25  worst compactness                     569 non-null    float64
26  worst concavity                       569 non-null    float64
27  worst concave points                  569 non-null    float64
28  worst symmetry                        569 non-null    float64
29  worst fractal dimension                569 non-null    float64
30  target                               569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
None
```

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension |
|-------|-------------|--------------|----------------|-------------|-----------------|------------------|----------------|---------------------|---------------|------------------------|
| count | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 |
| mean | 14.127292 | 19.289649 | 91.969033 | 654.889104 | 0.096360 | 0.104341 | 0.088799 | 0.048919 | 0.181162 | 0.0627 |
| std | 3.524049 | 4.301036 | 24.298981 | 351.914129 | 0.014064 | 0.052813 | 0.079720 | 0.038803 | 0.027414 | 0.0070 |
| min | 6.981000 | 9.710000 | 43.790000 | 143.500000 | 0.052630 | 0.019380 | 0.000000 | 0.000000 | 0.106000 | 0.0499 |
| 25% | 11.700000 | 16.170000 | 75.170000 | 420.300000 | 0.086370 | 0.064920 | 0.029560 | 0.020310 | 0.161900 | 0.0577 |
| 50% | 13.370000 | 18.840000 | 86.240000 | 551.100000 | 0.095870 | 0.092630 | 0.061540 | 0.033500 | 0.179200 | 0.0615 |
| 75% | 15.780000 | 21.800000 | 104.100000 | 782.700000 | 0.105300 | 0.130400 | 0.130700 | 0.074000 | 0.195700 | 0.0661 |
| max | 28.110000 | 39.280000 | 188.500000 | 2501.000000 | 0.163400 | 0.345400 | 0.426800 | 0.201200 | 0.304000 | 0.0974 |

8 rows × 31 columns

▼ Iris Dataset

idf

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|-----|-------------------|------------------|-------------------|------------------|--------|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | 0 |
| ... | ... | ... | ... | ... | ... |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | 2 |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | 2 |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | 2 |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | 2 |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | 2 |

150 rows × 5 columns

Next steps:

[Generate code with idf](#)[New interactive sheet](#)

```
print(idf.info())
idf.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
#   Column              Non-Null Count  Dtype
---  -
0   sepal length (cm)    150 non-null   float64
1   sepal width (cm)     150 non-null   float64
2   petal length (cm)    150 non-null   float64
3   petal width (cm)     150 non-null   float64
4   target               150 non-null   int64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
None
```

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|-------|-------------------|------------------|-------------------|------------------|------------|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 5.843333 | 3.057333 | 3.758000 | 1.199333 | 1.000000 |
| std | 0.828066 | 0.435866 | 1.765298 | 0.762238 | 0.819232 |
| min | 4.300000 | 2.000000 | 1.000000 | 0.100000 | 0.000000 |
| 25% | 5.100000 | 2.800000 | 1.600000 | 0.300000 | 0.000000 |
| 50% | 5.800000 | 3.000000 | 4.350000 | 1.300000 | 1.000000 |
| 75% | 6.400000 | 3.300000 | 5.100000 | 1.800000 | 2.000000 |
| max | 7.900000 | 4.400000 | 6.900000 | 2.500000 | 2.000000 |

Wine Dataset

wdf

| | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols | flavanoids | nonflavanoid_phenols | proan |
|---|---------|------------|------|-------------------|-----------|---------------|------------|----------------------|-------|
| 0 | 14.23 | 1.71 | 2.43 | | 15.6 | 127.0 | 2.80 | 3.06 | 0.28 |
| 1 | 13.20 | 1.78 | 2.14 | | 11.2 | 100.0 | 2.65 | 2.76 | 0.26 |

```

print(wdf.info())
wdf.describe()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
#   Column              ...    Non-Null Count  Dtype  ...
---  ---
0    alcohol             178 non-null    float64  ...
1    malic_acid           178 non-null    float64  ...
2    ash                  178 non-null    float64  ...
3    alcalinity_of_ash    178 non-null    float64  ...
4    magnesium            178 non-null    float64  ...
5    total_phenols        178 non-null    float64  ...
6    flavanoids           178 non-null    float64  ...
7    nonflavanoid_phenols 178 non-null    float64  ...
8    proanthocyanins      178 non-null    float64  ...
9    color_intensity      178 non-null    float64  ...
10   hue                  178 non-null    float64  ...
11   ...                  ...            ...      ...
12   ...                  ...            ...      ...
13   target              178 non-null    int64    ...
dtypes: float64(13), int64(1)
memory usage: 19.6 KB
None

```

| | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols | flavanoids | nonflavanoid_phenols | proan |
|-------|------------|------------|------------|-------------------|------------|---------------|------------|----------------------|------------|
| count | 178.000000 | 178.000000 | 178.000000 | | 178.000000 | 178.000000 | 178.000000 | 178.000000 | 178.000000 |
| mean | 13.000618 | 2.336348 | 2.366517 | | 19.494944 | 99.741573 | 2.295112 | 2.029270 | 0.3618 |
| std | 0.811827 | 1.117146 | 0.274344 | | 3.339564 | 14.282484 | 0.625851 | 0.998859 | 0.1244 |
| min | 11.030000 | 0.740000 | 1.360000 | | 10.600000 | 70.000000 | 0.980000 | 0.340000 | 0.1300 |
| 25% | 12.362500 | 1.602500 | 2.210000 | | 17.200000 | 88.000000 | 1.742500 | 1.205000 | 0.2700 |
| 50% | 13.050000 | 1.865000 | 2.360000 | | 19.500000 | 98.000000 | 2.355000 | 2.135000 | 0.3400 |
| 75% | 13.677500 | 3.082500 | 2.557500 | | 21.500000 | 107.000000 | 2.800000 | 2.875000 | 0.4375 |
| max | 14.830000 | 5.800000 | 3.230000 | | 30.000000 | 162.000000 | 3.880000 | 5.080000 | 0.6600 |

Standardizing Datasets

```
scaler = StandardScaler()
```

```
bcdf_stand = pd.DataFrame(scaler.fit_transform(bcdf), columns=bcdf.columns, index=bcdf.index)
bcdf_stand["target"] = bcdf["target"].values
```

```
idf_stand = pd.DataFrame(scaler.fit_transform(idf), columns=idf.columns, index=idf.index)
idf_stand["target"] = idf["target"].values
```

```
wdf_stand = pd.DataFrame(scaler.fit_transform(wdf), columns=wdf.columns, index=wdf.index)
wdf_stand["target"] = wdf["target"].values
```

Splitting data based on features and targets

```
bc_features = bcdf.iloc[:, :-1]
bc_target = bcdf.iloc[:, -1]
idf_features = idf.iloc[:, :-1]
idf_target = idf.iloc[:, -1]
w_features = wdf.iloc[:, :-1]
w_target = wdf.iloc[:, -1]
```

Importing libraries to train data

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn import svm
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

Splitting Dataset for training and testing

```
bc_X_train, bc_X_test, bc_y_train, bc_y_test = train_test_split(bc_features, bc_target, test_size=0.2, random_state=0)
i_X_train, i_X_test, i_y_train, i_y_test = train_test_split(i_features, i_target, test_size=0.2, random_state=0)
w_X_train, w_X_test, w_y_train, w_y_test = train_test_split(w_features, w_target, test_size=0.2, random_state=0)
```

Fitting and measuring models

```
kernels = ["linear", "poly", "rbf"]

bc_report = {
    "linear": None,
    "poly": None,
    "rbf": None,
    "accuracy": []
}

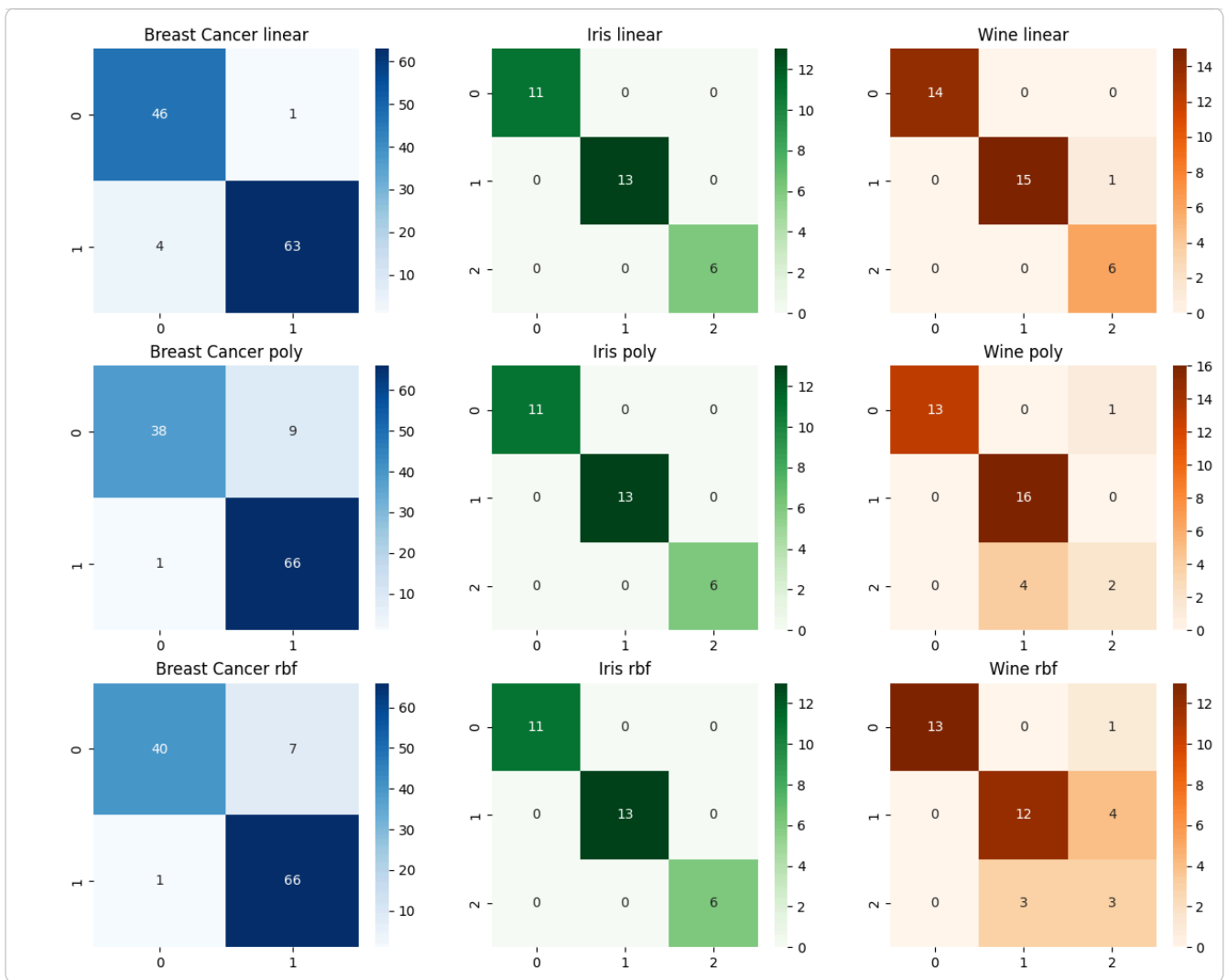
i_report = {
    "linear": None,
    "poly": None,
    "rbf": None,
    "accuracy": []
}

w_report = {
    "linear": None,
    "poly": None,
    "rbf": None,
    "accuracy": []
}

i = 1;
plt.figure(figsize=(15,12))
for kernel in kernels:
    bc_model = svm.SVC(kernel=kernel, gamma="scale")
    bc_model.fit(bc_X_train, bc_y_train)
    bc_report[kernel] = classification_report(bc_y_test, bc_model.predict(bc_X_test), output_dict=True, target_names =
    bc_pred = bc_model.predict(bc_X_test)
    bc_report["accuracy"].append(accuracy_score(bc_y_test, bc_pred))
    plt.subplot(3,3,i)
    i+=1
    sns.heatmap(confusion_matrix(bc_y_test, bc_pred), annot=True, cmap=plt.cm.Blues)
    plt.title(f"Breast Cancer {kernel}")

    i_model = svm.SVC(kernel=kernel, gamma="scale")
    i_model.fit(i_X_train, i_y_train)
    i_report[kernel] = classification_report(i_y_test, i_model.predict(i_X_test), output_dict=True, target_names = i_d
    i_pred = i_model.predict(i_X_test)
    i_report["accuracy"].append(accuracy_score(i_y_test, i_pred))
    plt.subplot(3,3,i)
    i+=1
    sns.heatmap(confusion_matrix(i_y_test, i_pred), annot=True, cmap=plt.cm.Greens)
    plt.title(f"Iris {kernel}")

    w_model = svm.SVC(kernel=kernel, gamma="scale")
    w_model.fit(w_X_train, w_y_train)
    w_report[kernel] = classification_report(w_y_test, w_model.predict(w_X_test), output_dict=True, target_names = w_d
    w_pred = w_model.predict(w_X_test)
    w_report["accuracy"].append(accuracy_score(w_y_test, w_pred))
    plt.subplot(3,3,i)
    i+=1
    sns.heatmap(confusion_matrix(w_y_test, w_pred), annot=True, cmap=plt.cm.Oranges)
    plt.title(f"Wine {kernel}")
plt.show()
```



Breast Cancer Model Report

```
for i, accu in enumerate(bc_report["accuracy"]):
    print(f"Accuracy {kernels[i]}: {(accu)*100:.2f}%")
```

Accuracy linear: 95.61%
 Accuracy poly: 91.23%
 Accuracy rbf: 92.98%

Linear Kernel

```
pd.DataFrame(bc_report["linear"])
```

| | malignant | benign | accuracy | macro avg | weighted avg |
|------------------|-----------|-----------|----------|------------|--------------|
| precision | 0.920000 | 0.984375 | 0.95614 | 0.952187 | 0.957834 |
| recall | 0.978723 | 0.940299 | 0.95614 | 0.959511 | 0.956140 |
| f1-score | 0.948454 | 0.961832 | 0.95614 | 0.955143 | 0.956316 |
| support | 47.000000 | 67.000000 | 0.95614 | 114.000000 | 114.000000 |


Polynomial Kernel

```
pd.DataFrame(bc_report["poly"])
```

| | malignant | benign | accuracy | macro avg | weighted avg |
|------------------|-----------|-----------|----------|------------|--------------|
| precision | 0.974359 | 0.880000 | 0.912281 | 0.927179 | 0.918902 |
| recall | 0.808511 | 0.985075 | 0.912281 | 0.896793 | 0.912281 |
| f1-score | 0.883721 | 0.929577 | 0.912281 | 0.906649 | 0.910672 |
| support | 47.000000 | 67.000000 | 0.912281 | 114.000000 | 114.000000 |

RBF Kernel

```
pd.DataFrame(bc_report["rbf"])
```

| | malignant | benign | accuracy | macro avg | weighted avg |  |
|------------------|-----------|-----------|----------|------------|--------------|---|
| precision | 0.975610 | 0.904110 | 0.929825 | 0.939860 | 0.933588 | |
| recall | 0.851064 | 0.985075 | 0.929825 | 0.918069 | 0.929825 | |
| f1-score | 0.909091 | 0.942857 | 0.929825 | 0.925974 | 0.928936 | |
| support | 47.000000 | 67.000000 | 0.929825 | 114.000000 | 114.000000 | |


Iris Model Report

```
for i,accu in enumerate(i_report["accuracy"]):
    print(f"Accuracy {kernels[i]}: {(accu)*100:.2f}%")
```

Accuracy linear: 100.00%
 Accuracy poly: 100.00%
 Accuracy rbf: 100.00%


Linear Kernel

```
pd.DataFrame(i_report["linear"])
```

| | setosa | versicolor | virginica | accuracy | macro avg | weighted avg |  |
|------------------|--------|------------|-----------|----------|-----------|--------------|---|
| precision | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| recall | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| f1-score | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| support | 11.0 | 13.0 | 6.0 | 1.0 | 30.0 | 30.0 | |


Polynomial Kernel

```
pd.DataFrame(i_report["poly"])
```

| | setosa | versicolor | virginica | accuracy | macro avg | weighted avg |  |
|------------------|--------|------------|-----------|----------|-----------|--------------|---|
| precision | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| recall | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| f1-score | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| support | 11.0 | 13.0 | 6.0 | 1.0 | 30.0 | 30.0 | |

RBF Kernel

```
pd.DataFrame(i_report["rbf"])
```

| | setosa | versicolor | virginica | accuracy | macro avg | weighted avg |  |
|------------------|--------|------------|-----------|----------|-----------|--------------|---|
| precision | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| recall | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| f1-score | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | |
| support | 11.0 | 13.0 | 6.0 | 1.0 | 30.0 | 30.0 | |


Wine Model Report

```
for i,accu in enumerate(w_report["accuracy"]):
    print(f"Accuracy {kernels[i]}: {(accu)*100:.2f}%")
```

Accuracy linear: 97.22%
 Accuracy poly: 86.11%
 Accuracy rbf: 77.78%

Linear Kernel

```
pd.DataFrame(w_report["linear"])
```

| | class_0 | class_1 | class_2 | accuracy | macro avg | weighted avg |  |
|------------------|---------|-----------|----------|----------|-----------|--------------|---|
| precision | 1.0 | 1.000000 | 0.857143 | 0.972222 | 0.952381 | 0.976190 | |
| recall | 1.0 | 0.937500 | 1.000000 | 0.972222 | 0.979167 | 0.972222 | |
| f1-score | 1.0 | 0.967742 | 0.923077 | 0.972222 | 0.963606 | 0.972843 | |
| support | 14.0 | 16.000000 | 6.000000 | 0.972222 | 36.000000 | 36.000000 | |

Polynomial Kernel

```
pd.DataFrame(w_report["poly"])
```