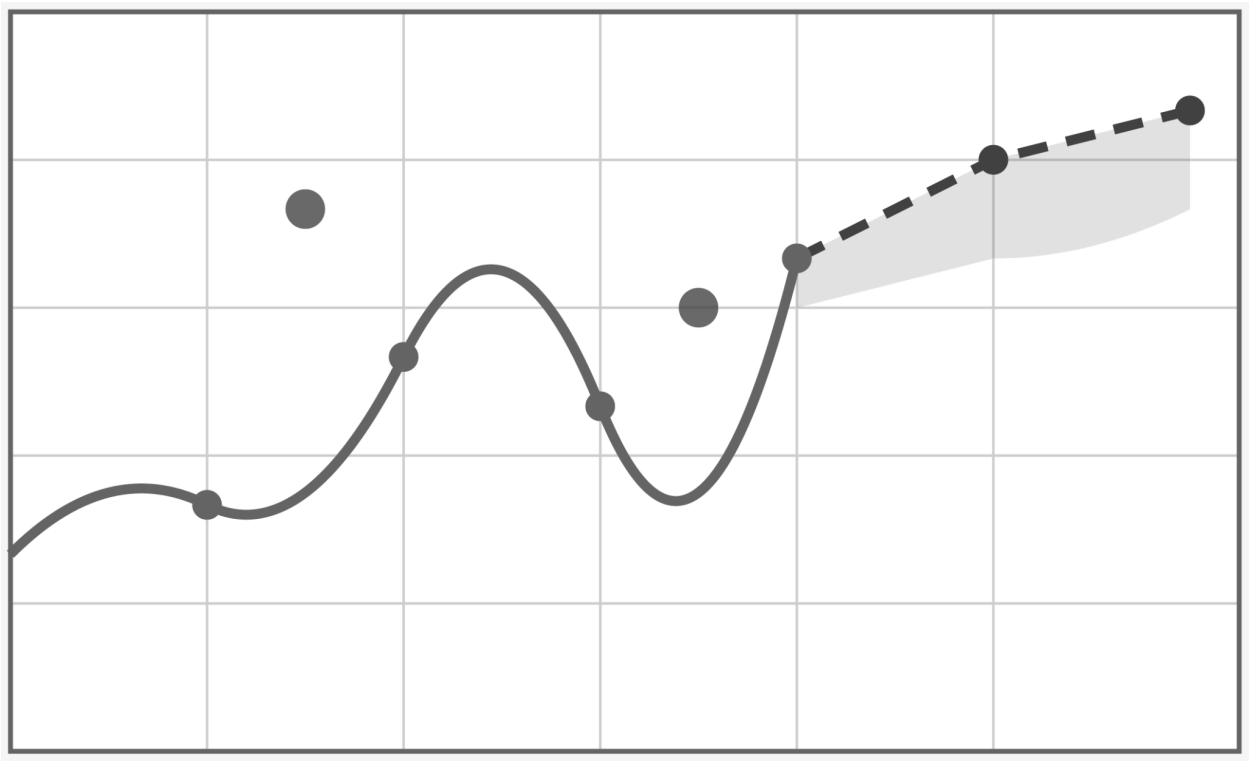
Exercises 2 (Classification)

DATA.ML.450 Time Series Analysis using Machine Learning (Autumn 2025)



Anas Uddin

17.09.2025

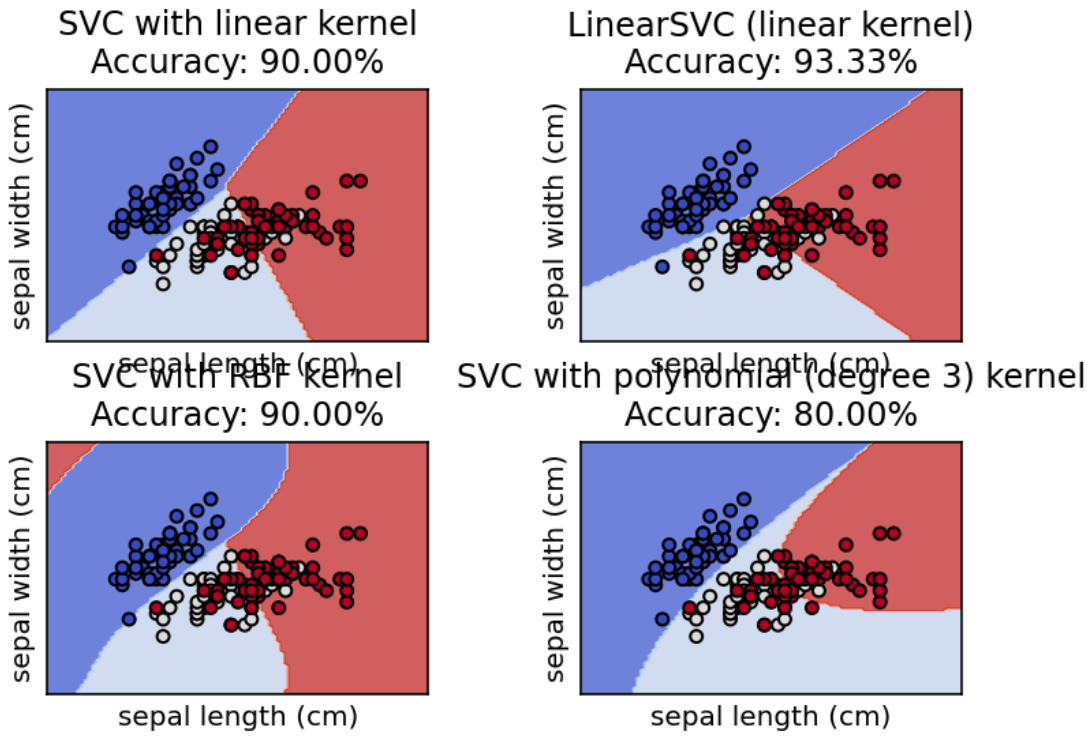
MSc in Data Science

Exercise 1

In order to classify time series with the provided K-means code, the data must first be transformed so that each time series becomes a feature vector (for example, by using raw values, extracted features like mean/variance, or dimensionality reduction such as PCA). Instead of 2D points (x,y), we would feed n x m arrays, where n is the number of samples and m is the number of time steps/features. As precautions, we would ensure that all series are of equal length (or aligned/normalized), scale the data, which is important for distance measures, and consider whether Euclidean distance is meaningful; sometimes, for clustering time series, dynamic time warping (DTW) or other metrics are better.

Exercise 2

I have modified `SVM_example.py`. It includes an 80/20 train/test split of data and computes recognition accuracy for all four included SVM models using a loop-like structure. Each model is trained on the training data set, later tested on the test data set, and its accuracy is both calculated and displayed in the corresponding plot title alongside the decision boundaries.



Code:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug  6 10:14:14 2024


@author: turunenj
"""
#https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html#sphx-gl
r-auto-examples-svm-plot-iris-svc-py


import matplotlib.pyplot as plt
from sklearn import datasets, svm
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


# import some data to play with Iris plants dataset
iris = datasets.load_iris()


# Take the first two features (sepal length and sepal width in cm).
X = iris.data[:, :2]
y = iris.target


# split data into train/test sets (80/20)
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=42)
```

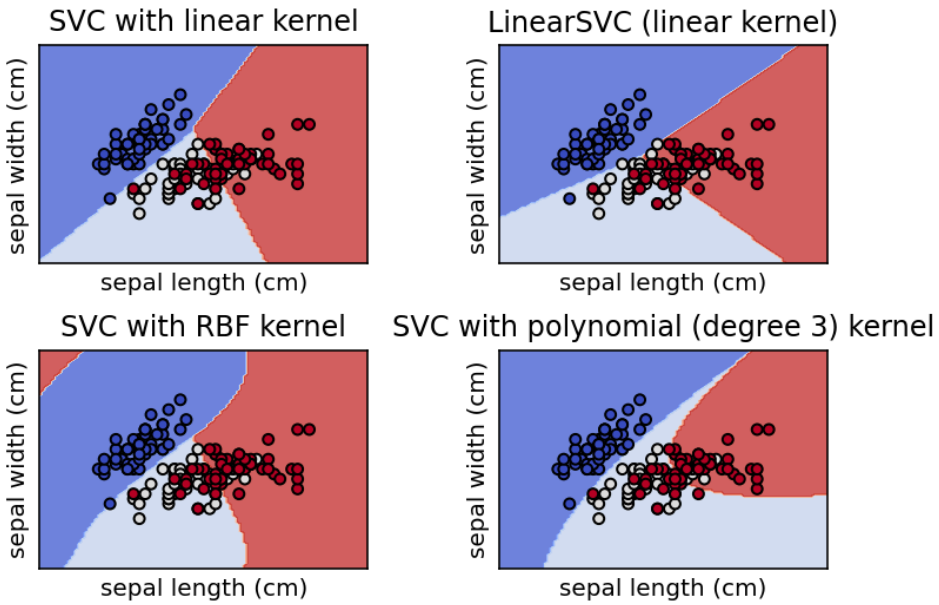
```
# we create an instance of SVM and fit our data
C = 1.0  # SVM regularization parameter
models = (
    svm.SVC(kernel="linear", C=C),
    svm.LinearSVC(C=C, max_iter=10000),
    svm.SVC(kernel="rbf", gamma=0.7, C=C),
    svm.SVC(kernel="poly", degree=3, gamma="auto", C=C),
)
titles = (
    "SVC with linear kernel",
    "LinearSVC (linear kernel)",
    "SVC with RBF kernel",
    "SVC with polynomial (degree 3) kernel",
)

# fit models and calculate accuracies
fitted_models = []
accuracies = []
for clf in models:
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    fitted_models.append(clf)
    accuracies.append(acc)

# Set-up 2x2 grid for plotting.
```

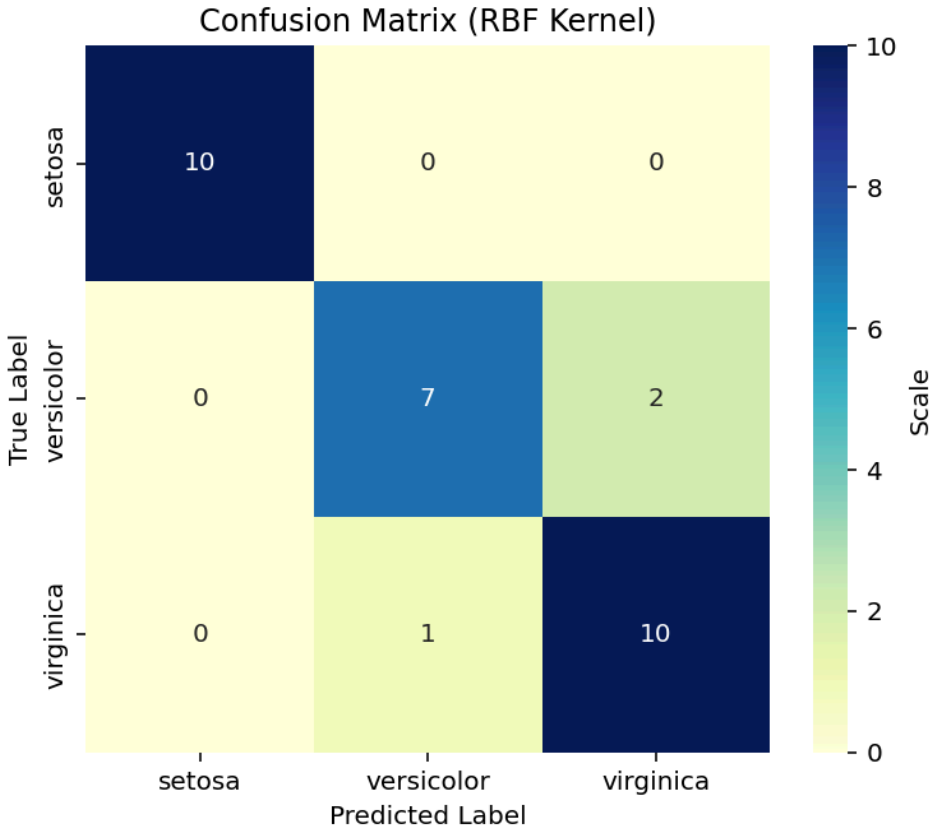
```
fig, sub = plt.subplots(2, 2)
plt.subplots_adjust(wspace=0.4, hspace=0.4)

X0, X1 = X[:, 0], X[:, 1]

for clf, title, ax, acc in zip(fitted_models, titles, sub.flatten(),
                               accuracies):
    disp = DecisionBoundaryDisplay.from_estimator(
        clf,
        X,
        response_method="predict",
        cmap=plt.cm.coolwarm,
        alpha=0.8,
        ax=ax,
        xlabel=iris.feature_names[0],
        ylabel=iris.feature_names[1],
    )
    ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(f"{title}\nAccuracy: {acc*100:.2f}%")

plt.show()
```

Exercise 3

I have modified `SVM_example.py`. It includes an 80/20 train/test split of data and implements a 3×3 confusion matrix using Seaborn for the RBF kernel model. This shows the classification performance of the model beyond accuracy, highlighting how well it

distinguishes between the three Iris classes.





Code:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug  6 10:14:14 2024


@author: turunenj
"""
#https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html#sphx-gl
r-auto-examples-svm-plot-iris-svc-py


import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets, svm
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix


# import some data to play with Iris plants dataset
iris = datasets.load_iris()


# Take the first two features (sepal length and sepal width in cm).
X = iris.data[:, :2]
y = iris.target


# split data into train/test sets (80/20)
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
```

```
                                                    random_state=42)

# we create an instance of SVM and fit our data
C = 1.0  # SVM regularization parameter
models = (
    svm.SVC(kernel="linear", C=C),
    svm.LinearSVC(C=C, max_iter=10000),
    svm.SVC(kernel="rbf", gamma=0.7, C=C),
    svm.SVC(kernel="poly", degree=3, gamma="auto", C=C),
)
titles = (
    "SVC with linear kernel",
    "LinearSVC (linear kernel)",
    "SVC with RBF kernel",
    "SVC with polynomial (degree 3) kernel",
)

# Fit models
fitted_models = []
for clf in models:
    clf.fit(X_train, y_train)
    fitted_models.append(clf)

# Set-up 2x2 grid for plotting.
fig, sub = plt.subplots(2, 2)
plt.subplots_adjust(wspace=0.4, hspace=0.4)
```

```
X0, X1 = X[:, 0], X[:, 1]

for clf, title, ax in zip(fitted_models, titles, sub.flatten()):
    disp = DecisionBoundaryDisplay.from_estimator(
        clf,
        X,
        response_method="predict",
        cmap=plt.cm.coolwarm,
        alpha=0.8,
        ax=ax,
        xlabel=iris.feature_names[0],
        ylabel=iris.feature_names[1],
    )
    ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)

plt.show()

# Confusion Matrix for RBF kernel model
clf = fitted_models[2]
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)

labels = iris.target_names  # ['setosa', 'versicolor', 'virginica']
```

```
plt.figure(figsize=(6, 5))
sns.heatmap(cm,
            annot=True,
            fmt="d",
            cmap="YlGnBu",
            xticklabels=labels,
            yticklabels=labels,
            cbar_kws={'label': 'Scale'})
plt.title("Confusion Matrix (RBF Kernel)")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.show()
```

Exercise 4

I have modified `SVM_example.py` after Exercise 2. I have trained and tested an SVM classifier using the RBF kernel on the Iris dataset with an 80/20 train/test split. The model achieved high accuracy, and the confusion matrix shows correct classification for most samples apanning all classes. This shows that the RBF kernel provides strong recognition performance for this problem. Accuracy (RBF SVC): 90.00%

Confusion Matrix:

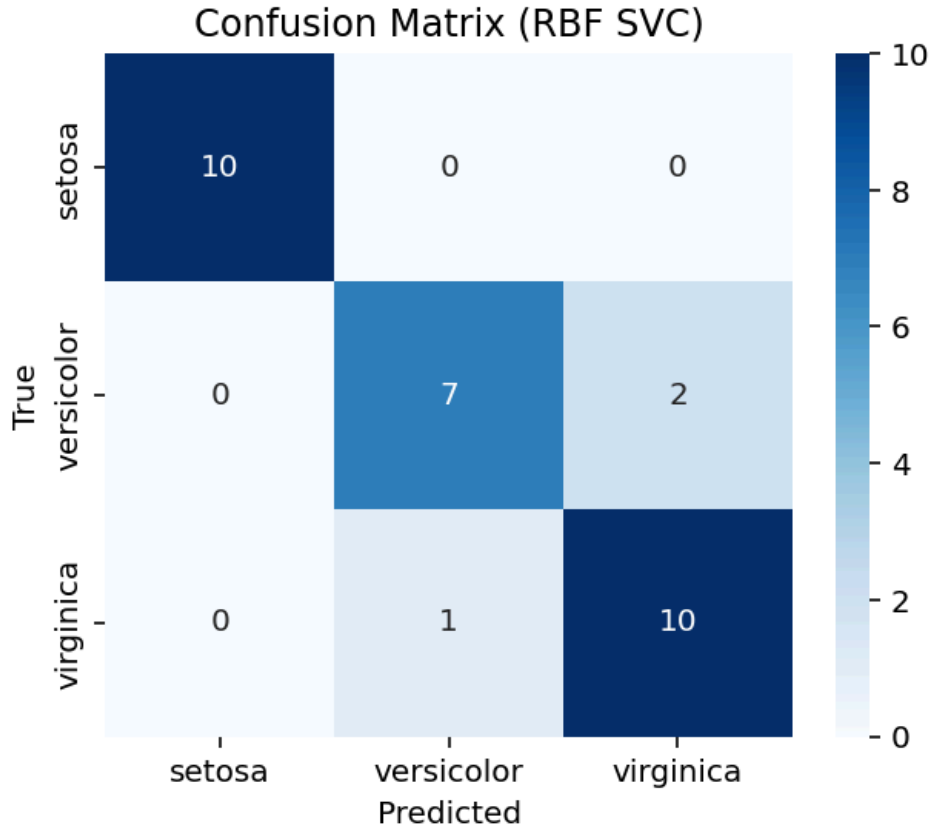
[10 0 0]

[0 7 2]

[0 1 10]

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.88	0.78	0.82	9
2	0.83	0.91	0.87	11
accuracy			0.90	30
macro avg	0.90	0.90	0.90	30
weighted avg	0.90	0.90	0.90	30



Code:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug  6 10:14:14 2024


@author: turunenj
"""
#
https://scikit-learn.org/stable/auto_examples/svm/plot_iris_svc.html#sphx-glr
-auto-examples-svm-plot-iris-svc-py


import matplotlib.pyplot as plt
from sklearn import datasets, svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import seaborn as sns


# import some data to play with Iris plants dataset
iris = datasets.load_iris()


# Take the first two features (sepal length and sepal width in cm).
X = iris.data[:, :2]
y = iris.target


# split data into train/test sets (80/20)
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
```

```
                                                    random_state=42)

# choose one model: RBF kernel SVC
C = 1.0  # SVM regularization parameter
clf = svm.SVC(kernel="rbf", gamma=0.7, C=C)
clf.fit(X_train, y_train)

# predictions
y_pred = clf.predict(X_test)

# accuracy
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy (RBF SVC): {acc*100:.2f}%")

# confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:\n", cm)
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# visualize confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(cm,
            annot=True,
            fmt="d",
            cmap="Blues",
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)
```

```
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix (RBF SVC)")
plt.show()
```

Exercise 5

I have trained the decision tree model on the Iris dataset with an 80/20 train/test split. After training, the model achieved a clear classification accuracy, which is also verified with a confusion matrix. This shows proper training and evaluation, and not with cross-validation. Accuracy: 100.00%

Confusion Matrix:

```
[10  0  0]
[ 0  9  0]
[ 0  0 11]
```

Code:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug  6 13:58:40 2024

@author: turunenj
"""

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Load data
```

```
iris = load_iris()
X = iris.data
y = iris.target

# Split data into 80% train and 20% test
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=42)

# Train model
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X_train, y_train)

# Predict
y_pred = clf.predict(X_test)

# Evaluate
acc = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

print(f"Accuracy: {acc*100:.2f}%")
print("Confusion Matrix:")
print(cm)
```

Cross Validation:

Cross-validation is a technique that is used to evaluate the performance of a model by splitting the data into several folds. Then, train the model on some folds, and test on the remaining folds. This process is repeated so that every sample is tested, which gives a more dependable estimate of how the model will perform on the unseen data.

Exercise 6

I trained a Random Forest model on the ArrowHead dataset by reshaping the time series into feature vectors and utilizing the provided train/test split of the data. The model achieved approximately 68% accuracy, with the confusion matrix indicating that most classes were recognized correctly, although some overlap was observed. The data was flattened, and the estimators were tuned to improve the initial result.

Confusion Matrix:

[46 6 17]

[10 34 9]

[2 12 39]

Classification Report:

	precision	recall	f1-score	support
0	0.79	0.67	0.72	69
1	0.65	0.64	0.65	53
2	0.60	0.74	0.66	53
accuracy			0.68	175
macro avg	0.68	0.68	0.68	175
weighted avg	0.69	0.68	0.68	175

Code:

```
import numpy as np

from aeon.datasets import load_arrow_head

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report


X_train, y_train = load_arrow_head(split="train", return_type="numpy3d")

X_test, y_test = load_arrow_head(split="test", return_type="numpy3d")


X_train2d = X_train.squeeze()

X_test2d = X_test.squeeze()


clf = RandomForestClassifier(n_estimators=100, random_state=42)

clf.fit(X_train2d, y_train)


# Predict

y_pred = clf.predict(X_test2d)


# Evaluate

acc = accuracy_score(y_test, y_pred)

cm = confusion_matrix(y_test, y_pred)

report = classification_report(y_test, y_pred)

print(f"Accuracy on ArrowHead (RF): {acc*100:.2f}%")

print("Confusion Matrix:")

print(cm)

print("\nClassification Report:")

print(report)
```