heart_disease_Prediction

August 8, 2023

Table of Contents

1 Motivation

Heart disease stands as a significant and prevalent global health concern, impacting millions across the world. Its grave consequences make it the foremost cause of death not only in developed nations but also in numerous developing regions. With its spectrum of symptoms and complications, encompassing factors like chest pain and others documented within our dataset, addressing heart disease promptly and effectively becomes paramount. The pivotal goals we set out to achieve involve the early prediction, diagnosis, and subsequent treatment of heart disease. Through this endeavor, we aim to empower healthcare practitioners and systems with the ability to foresee the presence of heart disease in patients. This project holds a special place in the realm of data science, holding a distinguished status within the Kaggle community as a renowned endeavor.

2 Imported Libraries

The following cells are used to bring in external Python libraries or modules that we need to use in this notebook. By leveraging Python packages like the following listed packages analysts can efficiently conduct data overviews and embark on a comprehensive data analysis journey to extract valuable knowledge from complex datasets.

2.1 Processing

PCA (e.g. Data reduction: selecting, aggregating, or summarizing the data to reduce its size or complexity.) and preprocessing (e.g. Data normalization: scaling or standardizing the data to make it comparable or compatible across different features or samples.) are both related to the data preparation phase of data science, which is the process of transforming raw data into a format that can be used for analysis, modeling. pandas: This is a Python package that provides fast and easy-to-use data structures and analysis tools, such as DataFrame and Series. scikit-learn: This is another Python package that provides various tools for data analysis (e.g. exploratory data analyses (EDA)) but also machine learning (see the next section), including PCA and other preprocessing methods.

[1]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import numpy as np
from collections import Counter
import random
import re
import string
```

2.2 Modeling

The modeling phase involves creating, training, and evaluating a machine learning model using the data that has been collected, annotated, and wrangled in the previous phases. There are many different types of classification models, such as logistic regression, decision trees, support vector machines, neural networks, and so on. scikit-learn: as in the last section mentioned it is also a Python package that provides various tools for machine learning and, such as classification, regression, clustering, feature selection, model evaluation, and model hyper-parameter optimization. Scikit-learn is also built on top of NumPy and Scipy (They are actually used for both processing and modeling in data science.), and it follows a consistent and simple interface for creating and using machine learning models. Keras: is a popular Python package that provides a high-level API for creating and using deep learning models. Deep learning is a branch of machine learning that uses multiple layers of artificial neural networks to learn from complex and high-dimensional data. Keras can work with different backends, such as TensorFlow or Theano, which are frameworks that offer low-level operations for building and running deep learning models. Keras can help with data representation, data learning, and data generation.

[3]:
```
from scipy import interp # This can be replaced newly by 'interp' from numpy.
from tqdm import tqdm
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
```

```python
from sklearn.svm import SVC
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc

import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
from keras.utils import to_categorical
```

2.3 Visualization

Data visualization is an important aspect of data analysis, as it can help to reveal patterns, trends, outliers, and relationships that might not be obvious from numerical or textual data alone. There are many tools and libraries available in Python for creating different types of plots, such as matplotlib , seaborn (It is a visualization library based on matplotlib), and plotly (express).

[4]:
```python
# Import all necessary packages
from pathlib import Path
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sbn
import seaborn.objects as so

sbn.set_style('whitegrid')
sbn.set(font_scale=1.0, font='sans-serif')
#import seaborn_image as isns
%matplotlib inline
plt.rcParams["figure.figsize"] = [8, 6]
font = {'family': 'serif',
        'color': 'darkblue',
        'weight': 'normal',
        'size': 18,
        }
kwargs = {'edgecolor': "black",  # for edge color
          'linewidth': 2,  # line width of spot
          }
from IPython.display import Markdown,display
from termcolor import colored
#import ast
```

3 Dataset Overview

A thorough data overview is the cornerstone of any successful data analysis endeavor. It empowers analysts to identify data-related challenges, understand the dataset's characteristics, and uncover critical patterns and insights.

[5]:
```
df_heart_disease = pd.read_csv('../DS/datasets/heart_disease.csv', sep=',',␣
↪encoding='utf-8-sig')
print(colored('The Head of Dataset:', attrs=['bold'], color='green'))
display(df_heart_disease.head(3))

print(colored('The Tail of Dataset:', attrs=['bold'], color='red'))
display( df_heart_disease.tail(3))

print(colored('The Dataset (short) Overview:', attrs=['bold'], color='blue'))
display(df_heart_disease.info())
df_test = df_heart_disease.head()
```

The Head of Dataset:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1

The Tail of Dataset:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

The Dataset (short) Overview:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
```

```
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
None
```

3.1 Header Explanation

The header that is above mentioned are the features or attributes of a dataset that is commonly used for heart disease prediction. This dataset is called the Heart Disease UCI Dataset and as in the above showed it contains ***303*** records of patients with 14 features each. The features are:

- **Age:** the age of the patient in years
- **Sex:** the sex of the patient (1 = male, 0 = female)
- **Cp:** the chest pain type of the patient (0 = typical angina, 1 = atypical angina, 2 = non-anginal pain, 3 = asymptomatic)
- **Trestbps:** the resting blood pressure of the patient in mm Hg
- **Chol:** the serum cholesterol of the patient in mg/dl
- **Fbs:** the fasting blood sugar of the patient ($>$ 120 mg/dl, 1 = true, 0 = false)
- **Restecg:** the resting electrocardiographic results of the patient (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy)
- **Thalach:** the maximum heart rate achieved by the patient
- **Exang:** the exercise induced angina of the patient (1 = yes, 0 = no)
- **Oldpeak:** the ST depression induced by exercise relative to rest
- **Slope:** the slope of the peak exercise ST segment (1 = up-sloping, 2 = flat, 3 = down-sloping)
- **Ca:** the number of major vessels colored by fluoroscopy (0-3)
- **Thal:** the thalassemia status of the patient (3 = normal, 6 = fixed defect, 7 = reversible defect)
- **Target:** the presence of heart disease in the patient (1 = yes, 0 = no)

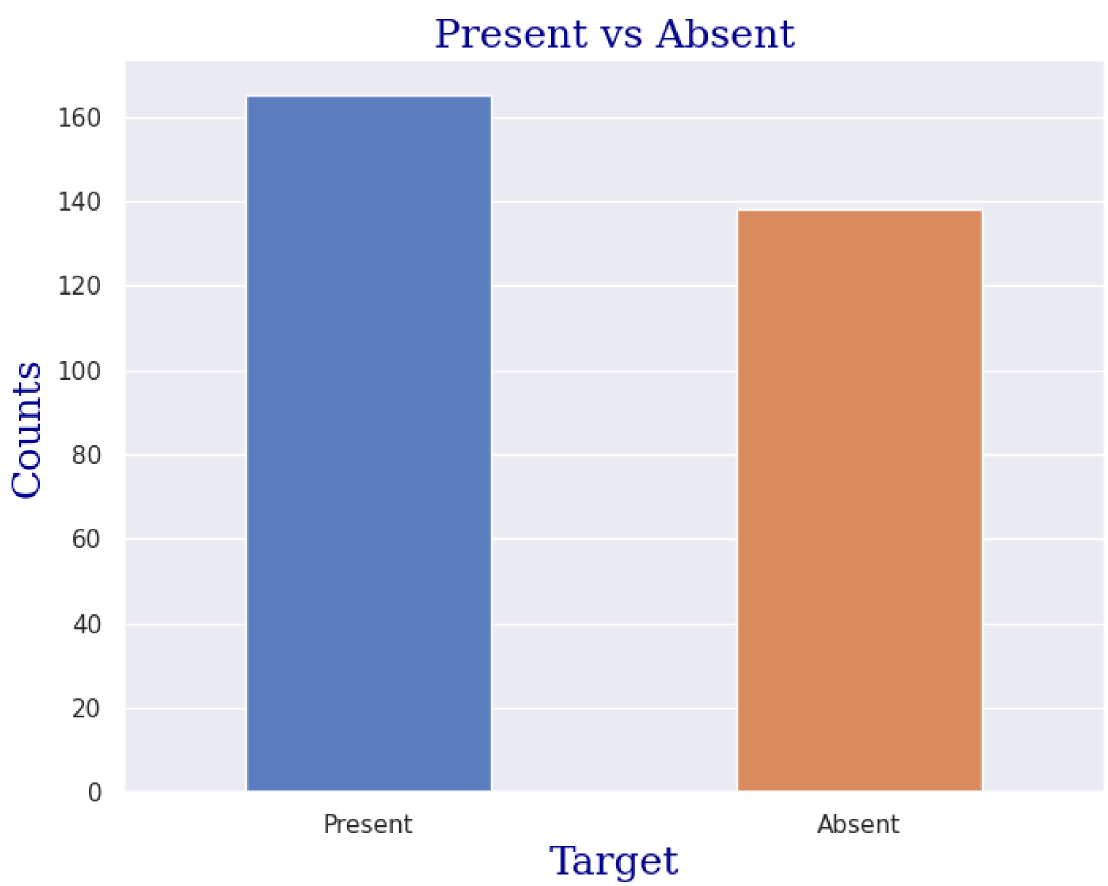
3.2 Presence vs Absence

This section refers to the comparison of the two possible outcomes, as shortly mentioned in the last section. The target variable is a crucial focus of analysis to understand the presence or absence of heart disease in the patients being studied.

[6]:
```
"""
Comparison between patients with heart disease and without that according their␣
 ↪age
present:  with heart disease
```

```
absent:  without heart disease
"""
# Firstly, we condense the dataframe so it counts the number of times each␣
 ↪target appears
present_absent = df_heart_disease['target'].value_counts().
 ↪rename_axis('target').to_frame('counts').reset_index()
present_absent['target'] = present_absent['target'].replace({1:'Present', 0:
 ↪'Absent'})
#display(present_absent)

# Plot a bar chart of the counts
fg1 = sbn.barplot(data= present_absent, x='target', y='counts',␣
 ↪palette='muted', width=0.5)
fg1.set_title('Present vs Absent', fontdict=font)
fg1.set_xlabel('Target', fontdict=font)
fg1.set_ylabel('Counts', fontdict=font)
```

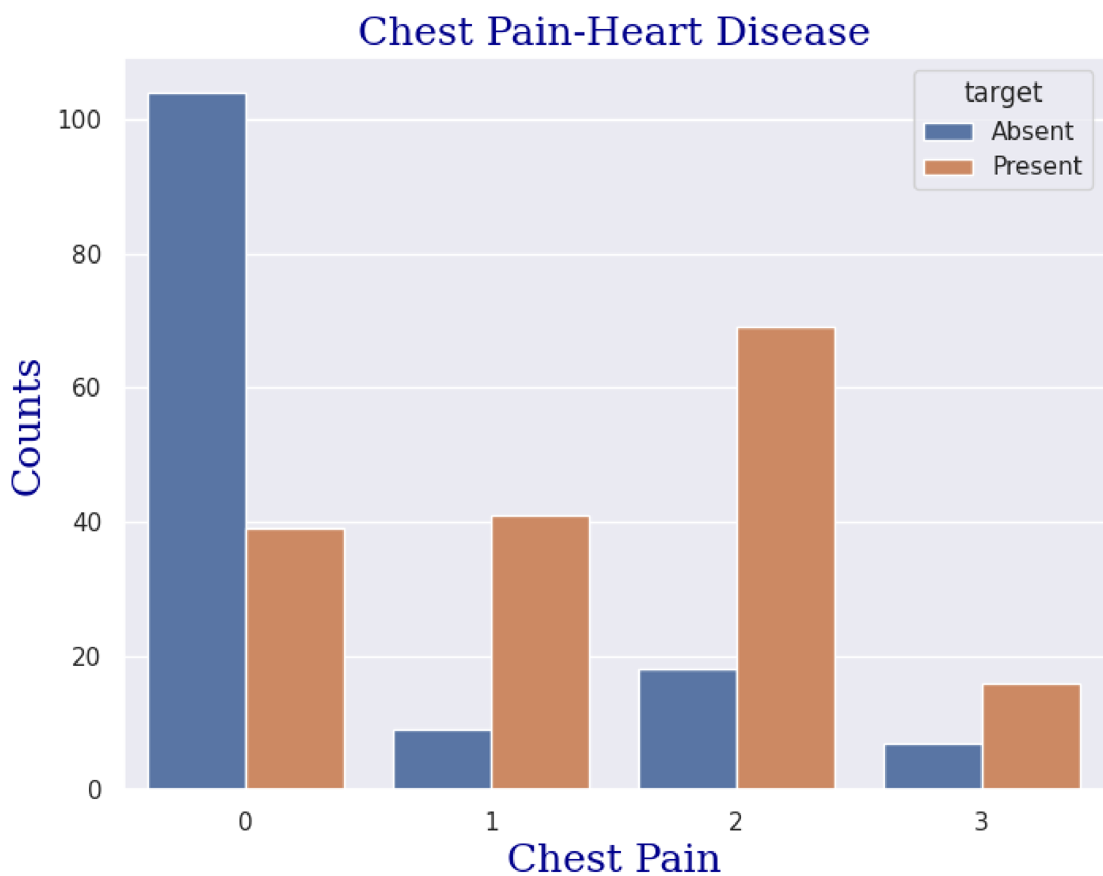
[6]: Text(0, 0.5, 'Counts')



Next, it would be useful for us to gain some insight into whether the chest pain (cp) of the patient plays an important role in the presence or absence of heart disease.

[7]:
```
present_absent_cp = df_heart_disease[['target', 'cp']].value_counts().
    ↪to_frame('counts').reset_index()
present_absent_cp['target'] = present_absent_cp['target'].replace({1:'Present',␣
    ↪0:'Absent'})
#display(present_absent_cp)
#present_absent_age.plot()
fg2 = sbn.barplot(data=present_absent_cp, x='cp', y='counts',hue='target')
fg2.set_title('Chest Pain-Heart Disease', fontdict=font)
fg2.set_xlabel('Chest Pain', fontdict=font)
fg2.set_ylabel('Counts', fontdict=font)
```

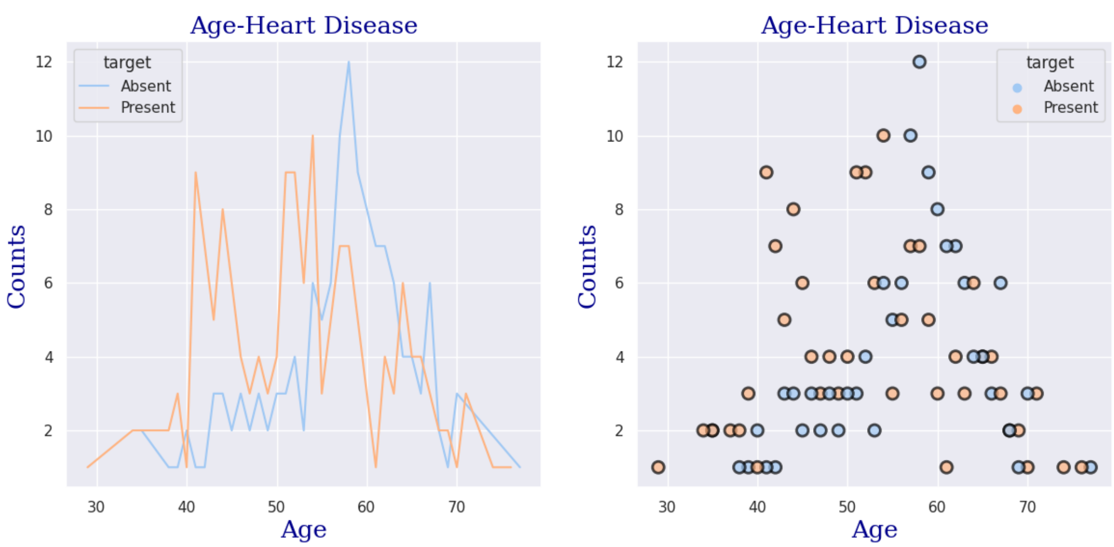
[7]:
```
Text(0, 0.5, 'Counts')
```



Only majority of patients with the first type of chest paine namely typical angina don't have any heart diseases. what

[8]:
```
present_absent_age = df_heart_disease[['target', 'age']].value_counts().
    ↪to_frame('counts').reset_index()
present_absent_age['target'] = present_absent_age['target'].replace({1:
    ↪'Present', 0:'Absent'})
#display(present_absent_age)
#present_absent_age.plot()
ax1= plt.subplots(nrows=1, ncols=2, figsize=(14,6))[1]
fg3 = sbn.lineplot(data=present_absent_age, x='age', y='counts',␣
    ↪hue='target',ax=ax1[0], palette='pastel')
sbn.move_legend(ax1[0], 'upper left')
ax1[0].set_title('Age-Heart Disease', fontdict=font)
ax1[0].set_xlabel('Age', fontdict=font)
ax1[0].set_ylabel('Counts', fontdict=font)

fg4= sbn.scatterplot(data=present_absent_age, x='age', y='counts', ax=ax1[1],␣
    ↪hue= 'target', palette='pastel',
 s=75, alpha=0.7,**kwargs)
ax1[1].set_title('Age-Heart Disease', fontdict=font)
ax1[1].set_xlabel('Age', fontdict=font)
ax1[1].set_ylabel('Counts', fontdict=font)
```

[8]:
```
Text(0, 0.5, 'Counts')
```



4 Exploratory Data Analysis

The last section demonstrated that observing an individual feature (in this case, the age of the patients) alone does not provide clear or conclusive information to accurately predict the presence or absence of heart diseases in the patients. The reason for this limitation is that the presence or absence of heart diseases is influenced by multiple factors (Multivariate Analysis (MVA), and a single feature, such as age, might not capture the full complexity of the relationship. To overcome this limitation and improve prediction accuracies, it is necessary to proceed with further steps, as we explain can in the two next sections.

4.1 Understanding Feature Relationships

To achieve this, we will in the next steps visualize the features regarding their correlations.

First we split the dataset into the feature Matrix **X** and the target array as corresponding class labels **y**. After that the feature matrix will be standardized by StandardScaler (from scikit-learn) with scales the features to have zero mean and unit variance.

[8]:
```
X = df_heart_disease.iloc[:,0:13].values
y = df_heart_disease.iloc[:,13].values

X_sts = StandardScaler().fit_transform(X)
```

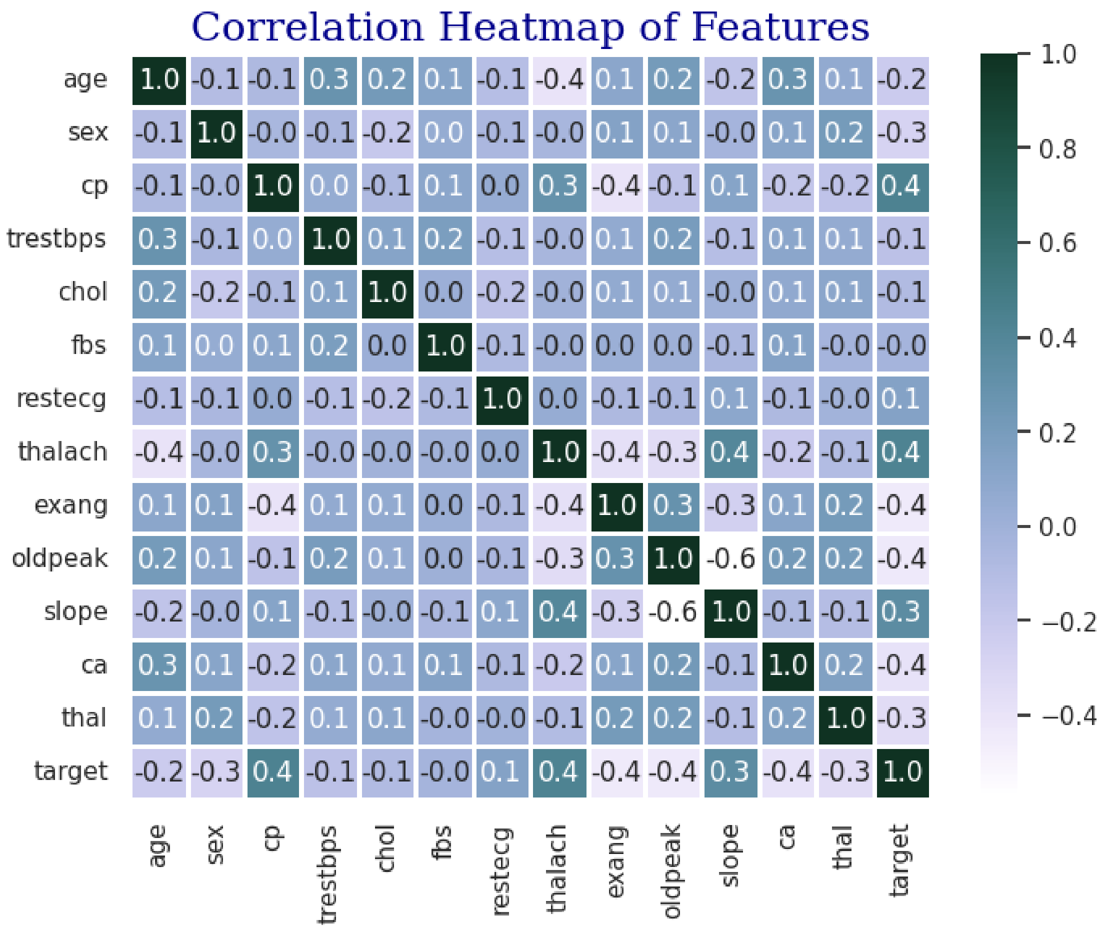
Data normalization is a preprocessing technique that scales the data to a standardized range. By normalizing the data, we ensure that all features have the same scale, which is particularly useful when dealing with features that have different units or ranges. This can improve the performance of certain machine learning algorithms that are sensitive to the scale of the features.

[9]:
```
df_norm = pd.DataFrame(X_sts, index=df_heart_disease.index,␣
 ↪columns=df_heart_disease.columns[0:13])

# Insert a non-feature target column to the dataframe
df_norm['target'] = df_heart_disease['target']
df_norm.head(10)

X = df_norm.iloc[:,0:13].values
y = df_norm.iloc[:,13].values
```

We calculate the correlation matrix for a normalized DataFrame and then create a heatmap to visually represent the correlations between its columns. The heatmap displays the correlation values between different features (columns) of the DataFrame, with the x-axis and y-axis labeled using the column names of the DataFrame.

[10]:
```
corr = df_norm.corr()
fg5 = sbn.heatmap(data=corr,xticklabels=corr.columns, yticklabels=corr.columns,␣
 ↪linewidths=.75,
 annot=True, fmt='.1f', cmap=sbn.cubehelix_palette(as_cmap=True, start=2,␣
 ↪light=1))
fg5.set_title('Correlation Heatmap of Features', fontdict=font)
```

[10]:
```
Text(0.5, 1.0, 'Correlation Heatmap of Features')
```



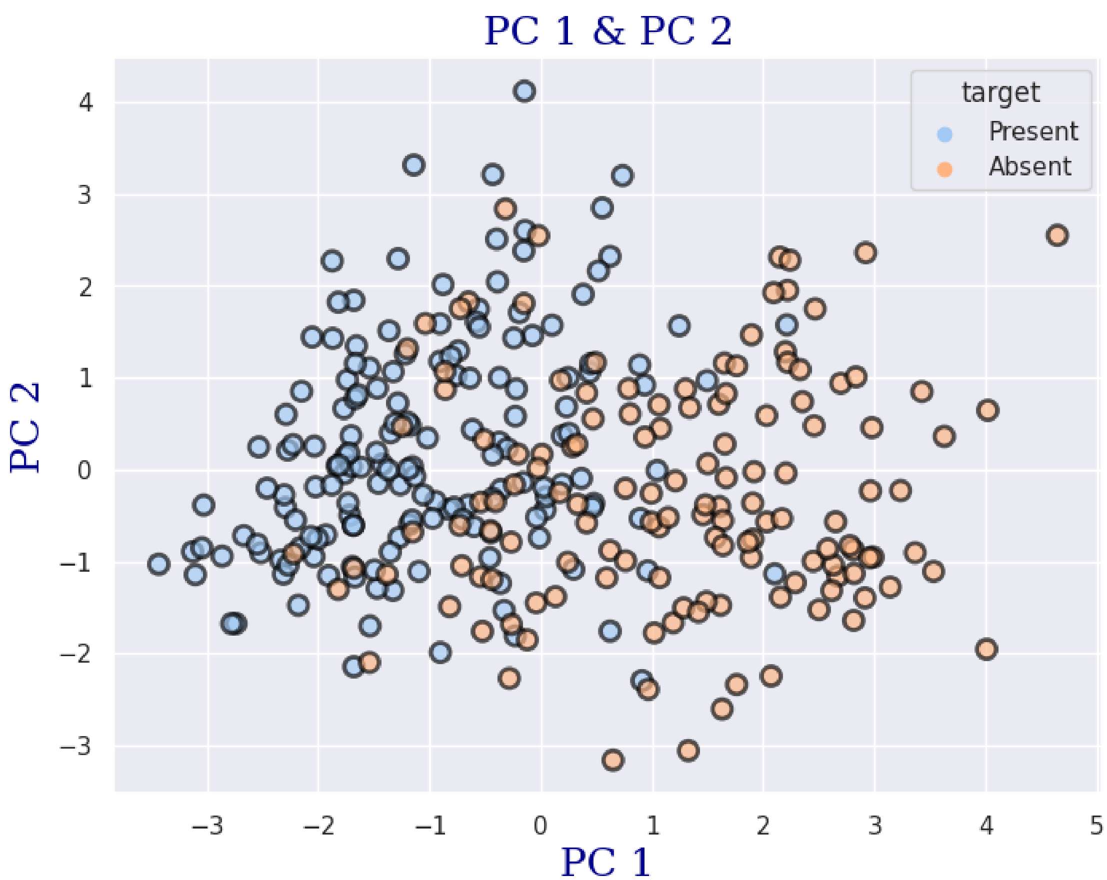
4.2 Principal Component Analysis

PCA is employed as a preprocessing step to simplify the data and facilitate its exploration and visualization. It helps identify strong patterns and relationships in the data, which can aid in understanding the underlying structure and informing subsequent analyses or modeling tasks.

[11]:
```
pca = PCA(n_components=2)
pcs = pca.fit_transform(X)
df_pc = pd.DataFrame(data = pcs, columns = ['PC 1', 'PC 2'])
#dff:dtat frame final
dff = pd.concat([df_pc, df_heart_disease[['target']]], axis = 1)
dff['target'] = dff['target'].replace({1:'Present', 0:'Absent'})
x_tar = dff[df_heart_disease['target'] == 'Present']
y_tar = dff[dff['target'] == 'Absent']
fg6 = sbn.scatterplot(data=dff, x='PC 1', y='PC 2', hue='target',
 ↪palette='pastel',
        s=75, alpha=0.65,**kwargs)
fg6.set_title('PC 1 & PC 2', fontdict=font)
```

```
fg6.set_xlabel('PC 1', fontdict=font)
fg6.set_ylabel('PC 2', fontdict=font)
```

[11]: Text(0, 0.5, 'PC 2')



5 Data-driven Model

Data-driven Model consists of many computational methods that can be divided into two main categories: clustering and classification. The emphasis is here on using data-driven approaches to develop a classification model that accurately predicts the class labels of new data based on its features and finally validated the classification. The process of data-driven classification typically includes many steps as listed in the next sections.

Data Preprocessing: This step basically involves cleaning, transforming, and preparing the dataset for modeling. It includes handling missing values, encoding categorical variables, and scaling numerical features if necessary. Notice: This has happen already during the last section.

5.1 Feature Selection

Selecting relevant features or extracting important information from the data to be used as inputs for the classification model. This step aims to reduce dimensionality and focus on the most informative features. In the following, we choose the Sequential Feature Selector (**SFS**) class with the setting defined in the sfs_config dictionary. This dictionary contains specific configurations to customize the behavior of the SFS: - forward: indicates if the Sequential Feature Selector will perform a forward feature selection process. In forward feature selection, the process starts with an empty set of features and iteratively adds one feature at a time that improves the model's performance the most. This continues until a stopping criterion is met. - floating: combines forward feature and backward selection (it is the inverse of forward feature selection start with all features and iteratively removes one feature at a time that has the least impact on the model's performance) which is basically more expensive than individual forward or backward selection, but it offers the advantage of exploring a wider range of feature combinations, potentially leading to better feature subsets for improved model performance. - scoring: This sets the scoring parameter to 'accuracies', indicating that the accuracies metric will be used to evaluate the quality of selected feature subsets. - cv: This sets the cv parameter to 5, specifying that 5-fold cross-validation will be used during the feature selection process. This means that the dataset will be divided into 5 subsets (folds), and the feature selection algorithm will run 5 times. In each run, one of the subsets will be used as the test set, and the other four subsets will be used as the training set. The purpose of cross-validation is to provide a more robust estimate of the model's performance by evaluating it on different subsets of the data.

Using a pipeline that integrates SFS provides a structured and controlled environment for feature selection, ensuring that it's performed in a consistent and reliable manner across different stages of the modeling process. It also supports proper evaluation techniques like cross-validation and facilitates easier experimentation and customization.

[13]:
```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS

# To avoid the repetition of som settings of the SFS
sfs_config = {
    'forward': True,
    'floating': False,
    'scoring': 'accuracies',
    'cv': 5
}
```

5.2 Train-Test Split

Splitting the dataset into training and testing subsets. The training set is used to train the classification model, while the testing set is used to evaluate its performance on unseen data. The splitting below separates the feature matrix X (containing input features) from the target array $\boldsymbol{y}$ (containing class labels). The data is split into training and testing sets, with 70% of the data used for training and 30% for testing as shortly mentioned above. Setting the random_state parameter ensures that the train and test sets will be the same every time you run the code, which can be useful for reproducibility. This process is a crucial step in data preparation for building and evaluating predictive models in machine learning tasks in later steps.

[15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=0)
```

5.3 Model Selection and Evaluation

In this section, we focus on the crucial steps of Model Selection and Evaluation for classification tasks. By incorporating both model selection and evaluation, we can make informed decisions about which classification algorithm best suits our specific problem and dataset. The section provides a comprehensive understanding of these critical steps in building effective classification models. To achieve this, we implement in this section an objective-oriented programming (OOP) approach, encapsulating the key functionalities into a **ModelEvaluator** class. By utilizing this ModelEvaluator class, we streamline the model selection process, assess model performance, and visualize the ROC curves and confusion matrices. The OOP approach enhances code modularity, reusability, and readability, making it easier to compare and evaluate different classification algorithms systematically. **Notice:** In K-Nearest Neighbors (KNN) the number of neighbors considered for classification should ideally be an odd number. This is particularly important in binary classification scenarios where an odd number prevents ties when votes are cast by the neighbors. When there's a tie in the voting process (e.g., an equal number of nearest neighbors from each class), using an odd number of neighbors (see the method k_opt() below) ensures that one class will have more votes than the other, thus leading to a clear majority decision.

[29]:
```
from sklearn.model_selection import train_test_split
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from tqdm import tqdm
from sklearn import model_selection
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc


class ModelEvaluator:
    results = {'acc_test': {}, 'acc_train': {}, }
    #-------------------------------------------------------------------
    # Constructor
    #-------------------------------------------------------------------
    def __init__(self, model=LinearDiscriminantAnalysis(), df=df_heart_disease,␣
 ↪n=13, cvn=5, mif=1, maf=5, head='target'):

        # Check the number of the features of dataframe greater than cross␣
 ↪validation number.
        if df.shape[1] < n:
            raise InvalidDataError('Number of features of dataframe is less␣
 ↪than the target index number')

        # Check the maximum number of the feature range of a SFS greater than␣
 ↪its minimum number.
```

```
        elif maf < mif:
            raise InvalidDataError('Maximum number of feature range of a SFS is␣
↪less than the minimum number')

        # Check the maximum number of the feature range less than or equal␣
↪cross validation number.
        elif maf > cvn:
            raise InvalidDataError('Maximum number of feature range of a SFS is␣
↪greater than the the cross-validation number')
        else:
            X_ = df.iloc[:,0:n].values
            y_ = df.iloc[:,n].values
            df_norm = pd.DataFrame(X_sts, index=df.index, columns=df.columns[0:
↪13])
            df_norm[head] = df_heart_disease[head]
            df_norm.head(10)

            self.X = df_norm.iloc[:,0:13].values
            self.y = df_norm.iloc[:,13].values
            self.X_train, self.X_test, self.y_train, self.y_test =␣
↪train_test_split(
                                self.X, self.y, test_size=0.3, random_state=0)
            self.sfs = SFS(estimator=model, k_features=(mif, maf), forward=True,
            floating=False, scoring='accuracy', cv=cvn).fit(self.X, self.y)
            self.df = df
            self.model = model

    #----------------------------------------------------------------------
    # This function aims to find the optimal number of neighbors for a
    # K-Nearest Neighbors classifier using cross-validation. It takes
    # training data X_train and corresponding labels y_train as inputs.
    # It iterates over a range of potential neighbor counts, constructs
    # KNN models for each count, evaluates them using cross-validation,
    # and stores the mean accuracy scores. Finally, it determines the neighbor
    # count with the highest accuracy and prints the result. The optimal number
    # of neighbors is returned as the output of the function.
    #----------------------------------------------------------------------
    def k_opt(self, X_train = None, y_train=None):
        if X_train is None:
            X_train = self.X_train
        if y_train is None:
            y_train = self.y_train
        Ns = [N for N in list(range(1,50)) if N % 2 == 1]
        cv_scores = []

        for k in Ns:
```

```
            knn = KNeighborsClassifier(n_neighbors = k + 1, weights='uniform',␣
↪p=2, metric='euclidean')
            kf = model_selection.KFold(n_splits=10, shuffle=True,␣
↪random_state=123)
            scores = model_selection.cross_val_score(knn, X_train, y_train,␣
↪cv=kf, scoring='accuracy')
            cv_scores.append(scores.mean()*100)
            #print("k=%d %0.2f (+/- %0.2f)" % (k_value, scores.mean()*100,␣
↪scores.std()*100))

        #opt_k = Ns[cv_scores.index(max(cv_scores))]
        opt_k = Ns[np.argmax(cv_scores)]
        #print ()
        print(( 'The optimal number of neighbors is %d with avg_acc of KNN %0.
↪1f%%.' % (opt_k, cv_scores[Ns.index(opt_k)])))
        return opt_k

    #--------------------------------------------------------------------------
    # This This method encapsulates the process of building, training,␣
↪evaluating,
    # and plotting the results of a neural network model for a given dataset.
    # The flexibility in specifying different parameters allows you to␣
↪experiment
    # with different network architectures and training settings. The boolean␣
↪input
    #'_tqdm' determines whether the model is trained using Keras' built-in␣
↪progress
    #  bar (when _tqdm is set to False) or using a loop over the epoch number␣
↪with
    #  a tqdm progress bar (when _tqdm is set to True).
    #--------------------------------------------------------------------------
    def nnw(self, hu1=64, hu2=32, lr=0.001, num_epoch=1000, _tqdm=True,␣
↪X_train=None,
     y_train=None, X_test=None, y_test=None):
        if X_train is None:
            X_train = self.X_train
        if y_train is None:
            y_train = self.y_train
        if X_test is None:
            X_test = self.X_test
        if y_test is None:
            y_test = self.y_test

        # Build model
        model = Sequential()
        model.add(Dense(hu1, input_dim=13, activation='relu'))
```

```
    model.add(Dense(hu2, activation='relu'))
    model.add(Dense(2, activation='softmax'))
    # Choose optimizer and loss function
    optimizer = SGD(learning_rate=lr, momentum=0.7)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer,␣
↪metrics=['accuracy'])

    # Convert labels to one-hot encoding
    yTrain_oneHot = to_categorical(y_train, num_classes=2)
    if _tqdm:
        # Train the model with tqdm progress bar
        progress_bar = tqdm(range(num_epoch), unit="epoch")
        loss_per_epoch = []  # Store loss values for each epoch
        for epoch in progress_bar:
            history = model.fit(X_train, yTrain_oneHot, epochs=1, verbose=0)
            loss_per_epoch.append(history.history['loss'][0])  # Append the␣
↪loss for this epoch
    else:
        # Train the model with built-in progress bar
        history = model.fit(X_train, yTrain_oneHot, epochs=num_epoch,␣
↪verbose=1)
        loss_per_epoch = history.history['loss']
    # Plot the training loss
    epochs = np.arange(1, num_epoch + 1)
    fg10 = sbn.lineplot(x= epochs, y=loss_per_epoch, label='Training',␣
↪legend=True)
    fg10.set_ylabel('Average Loss', fontdict=font)
    fg10.set_xlabel('Epochs', fontdict=font)
    fg10.set_title('Heart Disease', fontdict=font)
    fg10.legend()

    # Evaluate the model on test data
    yTest_oneHot = to_categorical(y_test, num_classes=2)
    accuracies_test = model.evaluate(X_test, yTest_oneHot, verbose=0)[1]

    # Evaluate the model on training data
    accuracies_train = model.evaluate(X_train, yTrain_oneHot, verbose=0)[1]

    print('Accuracies of the network on test data: {:.2f}%'.
↪format(accuracies_test * 100))
    print('Accuracies of the network on training data: {:.2f}%'.
↪format(accuracies_train * 100))


#--------------------------------------------------------------------
# The function valuates the model's accuracy on both the training
```

```
    # and test sets, and it allows for easy tracking and comparison of
    # different models by storing their accuracy scores in dictionaries.
    #-------------------------------------------------------------------
    def accuracies(self, model= None, display=True,
     X_train=None,X_test=None, y_train=None, y_test=None):

        if model is None:
            model = self.model
        if X_train is None:
            X_train = self.X_train
        if X_test is None:
            X_test = self.X_test
        if y_train is None:
            y_train = self.y_train
        if y_test is None:
            y_test = self.y_test

        model.fit(self.X_train, self.y_train)
        y_pred = model.predict(X_test)
        # Update results dictionaries
        acc_train = round(model.score(self.X_train, y_train) * 100, 2)
        acc_val = round(model.score(self.X_test, y_test) * 100, 2)
        if 'Pipeline' in str(model):
            method_name = str(model.steps[-1][1])[0:str(model.steps[-1][1]).
↪find('(')]
            method_name += '_pip'
        else:
            method_name = str(model)[0:str(model).find('(')]

        self.results['acc_test'][method_name] = acc_val
        self.results['acc_train'][method_name] = acc_train

        method_data = {method_name:{'y_pred': [y_pred], 'acc_val':acc_val,␣
↪'acc_train': acc_train}}

        if display:
            print('acc_train: %.2f\n'
                'acc_val: %.2f' %(acc_train, acc_val))
        else:
            return method_data

    #-------------------------------------------------------------------
    # This function encapsulates the steps needed to create and train a
    # pipeline,making it convenient work with models that involve feature
    # selection.
    #------------------------------------------------------------------
    def getPipe(self):
```

```python
        pipe = make_pipeline(
            self.sfs,
            StandardScaler(),
            self.model
        )
        pipe = pipe.fit(self.X, self.y)
        return pipe

    #--------------------------------------------------------------------
    # The following function serves as a utility to print and retrieve
    # the names and indices of the selected features obtained from the
    # Sequential Feature Selector object.
    #--------------------------------------------------------------------
    def display_features(self, sfs=None):
        if sfs is None:
            sfs = self.sfs
        #if display == 0:
            # Assuming sfs is an instance of the SequentialFeatureSelector class
        feature_names_dict = {sfs.k_feature_idx_[i]: self.df.iloc[:, sfs.
↪k_feature_idx_[i]].name
                        for i in range(len(sfs.k_feature_idx_))}
        print('Selected features: ',  feature_names_dict)
        print('\n')
        return tuple(np.array(sfs.k_feature_idx_[0:]))

    #-------------------------------------------------------------------------
    # This function provides a quick and easy way to assess the performance of
    # a classification model and is useful for evaluating its effectiveness
    # in distinguishing between different classes.
    #-------------------------------------------------------------------------
    def class_report(self):
        report = classification_report(y_true=self.y_test,
        y_pred=self.getPipe().predict(self.X_test), output_dict=True)
        df_report = pd.DataFrame(report).transpose()
        return df_report

    #-------------------------------------------------------------------
    # The val_diags function provides a comprehensive analysis of the
    # classification model's performance using cross-validated ROC curves,
    #  which helps assess the model's ability to distinguish between
    # classes and compare its performance across different folds.
    # Additionally it plots histogram of the mean TPR and TNR values.
    #-------------------------------------------------------------------
    def val_diags(self, model=None, X=None, y=None, cvn=5):
        cv = StratifiedKFold(n_splits=cvn)
        # List to store TPR, TNR, and AUC values
        tprs = []
```

```
        tnrs = []
        aucs = []

        ax2 = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))[1]
        _fprs = np.linspace(start=0, stop=1, num=300)

        if model is None:
            model = self.getPipe()
        if X is None:
            X = self.X
        if y is None:
            y = self.y
        attrs = self.display_features()

        for i, (train, test) in enumerate(cv.split(X=X[:, attrs], y=y)):
            predicts = model.fit(X[train], y[train].ravel()).
↪predict_proba(X[test])
            fpr, tpr = roc_curve(y[test].ravel(), predicts[:, 1])[0:2]

            tprs.append(np.interp(_fprs, fpr, tpr))
            #fprs.append(fpr)

            # Calculate TNR based on TPR and FPR
            tnr = 1 - fpr
            tnrs.append(np.interp(_fprs, fpr, tnr))  # Interpolate TNR values

            tprs[-1][0] = 0.0

            roc_auc = auc(fpr, tpr)
            aucs.append(roc_auc)
            fg7 = sbn.lineplot(data=pd.DataFrame({'fpr': fpr, 'tpr': tpr}),␣
↪x='fpr', y='tpr', alpha=0.3, legend='full',
                            label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc),␣
↪err_style='band', ax=ax2[0])

        fg7 = sbn.lineplot(x=[0, 1], y=[0, 1], label='Diagonal', legend='full',␣
↪lw=1.5, alpha=0.7, ax=ax2[0])

        mean_tpr = np.mean(tprs, axis=0)
        mean_tpr[-1] = 1.0
        mean_auc = auc(_fprs, mean_tpr)
        std_auc = np.std(aucs)
        fg7 = sbn.lineplot(x=_fprs, y=mean_tpr, label=r'Mean ROC (AUC = %0.2f␣
↪$\pm$ %0.2f)'
                        % (mean_auc, std_auc), lw=2.2, alpha=.7,␣
↪err_style='band', ax=ax2[0])
```

```
        # Calculate and plot Mean TNR (Specificity)
        mean_tnr = np.mean(tnrs, axis=0)
        mean_tnr[-1] = 1.0  # Set the last TNR value to 1.0 for proper plotting
        #fg7 = sbn.lineplot(x=_fprs, y=mean_tnr, label='Mean TNR␣
↪(Specificity)', lw=2.2, alpha=.7, err_style='band', ax=ax2[0])

        fg7.set_title('ROC', fontdict=font)
        fg7.set_xlabel('FPR', fontdict=font)
        fg7.set_ylabel('TRR', fontdict=font)

        fg81= sbn.histplot(mean_tpr, label='TPR',ax=ax2[1], kde=True,␣
↪legend=True, cbar=True, thresh=0, stat='frequency')
        fg81.legend()
        fg82= sbn.histplot(mean_tnr, label='TNR',ax=ax2[1],␣
↪kde=True,legend=True, cbar=True, thresh=0, stat='frequency')
        fg82.legend()
        fg82.set_ylabel('Frequency',fontdict= font)

        #fg81= sbn.lineplot(x=_fprs,y=mean_tnr, label='TP',ax=ax2[1], ␣
↪legend=True)



  ␣
↪#----------------------------------------------------------------------------------
    # The cfm(abbreviation of Confusion Matrix) function provides an␣
↪easy-to-interpret
    # visualization of the classification model's performance in terms of true␣
↪positive,
    # true negative, false positive, and false negative predictions, allowing␣
↪for quick
    # assessment of the model's effectiveness in distinguishing between classes.
  ␣
↪#----------------------------------------------------------------------------------
    def cfm(self, model=None, X_test=None, y_test=None):
        if model is None:
            model = self.getPipe()

        if X_test is None:
            X_test = self.X_test

        if y_test is None:
            y_test = self.y_test

        y_pred = model.predict(X_test)
```

```
        cm = metrics.confusion_matrix(y_pred= y_pred, y_true= y_test,␣
↪labels=[1, 0])
        fg9 = sbn.heatmap(cm, xticklabels=["1", "0"], yticklabels=["1", "0"],␣
↪annot=True, fmt='.2f',
          cmap=sbn.cubehelix_palette(as_cmap=True, start=2, light=.75, dark=.
↪25, gamma=.5,rot=-.2), linewidths=.75)
        #model_name = str(model.steps[-1][1])
        model_name = str(self.model)
        model_name = model_name[0:str(model_name).find('(')]
        fg9.set_title(model_name, fontdict=font)
        fg9.set_xlabel('Predicted label', fontdict=font)
        fg9.set_ylabel('True label', fontdict=font)
```

5.3.1 Assessing Models

In this section, we evaluate the performance of several classification algorithms commonly used for data classification tasks, such as LinearDiscriminant Analysis (LDA), Decision Trees, Random Forest, Support Vector Classification (SVC), Logistic Regression, Gradient Boosting, K-Nearest Neighbors,and Neural Networks. The evaluation is done based on the following metrics: - **Precision:** Irt represents the proportion of true positive predictions(TPP) (correctly predicted positive instances) over all positive predictions (both true positives and false positives). It measures the accuracies of positive predictions. - **Recall:** It is also known as sensitivity or true positive rate. It represents the proportion of true positive predictions over all actual positive instances in the dataset. It measures the ability of the model to identify positive instances. - **F1-score:** It is the harmonic mean of precision and recall. It provides a balanced measure of precision and recall, especially when there is an imbalance between positive and negative instances. - **Support:** It indicates the number of instances in each class in the test set, showing the distribution of data across classes. - **accuracies:** It represents the overall correctness of the model's predictions. It is the proportion of correctly predicted instances (both true positives and true negatives) over the total number of instances.

The evaluation results help us later compare the performance of different models and select the most suitable one for our specific classification problem.

LinearDiscriminant Analysis (LDA) It is a dimensionality reduction and classification technique commonly used in the field of machine learning and statistics. Its primary goal is to find a linear combination of features that best separates different classes in a dataset. It is often used for supervised classification tasks, where the classes of the data are known.

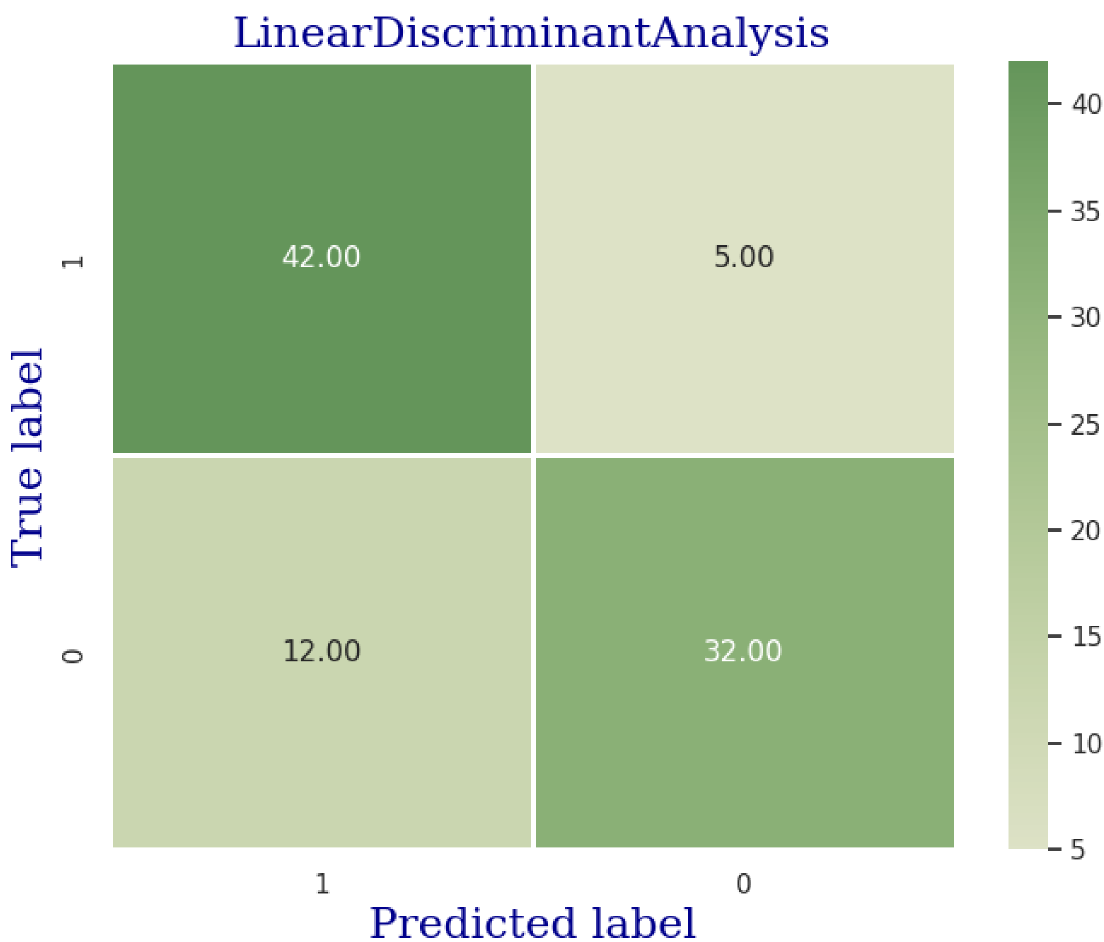
[30]:
```
evaluator = ModelEvaluator(model=LinearDiscriminantAnalysis())

evaluator.accuracies()
```

```
acc_train: 85.38
acc_val: 80.22
```

[31]:
```
evaluator.class_report()
```

	precision	recall	f1-score	support
0	0.864865	0.727273	0.790123	44.000000
1	0.777778	0.893617	0.831683	47.000000
accuracy	0.813187	0.813187	0.813187	0.813187
macro avg	0.821321	0.810445	0.810903	91.000000
weighted avg	0.819886	0.813187	0.811588	91.000000

[32]:
```
evaluator.cfm()
```

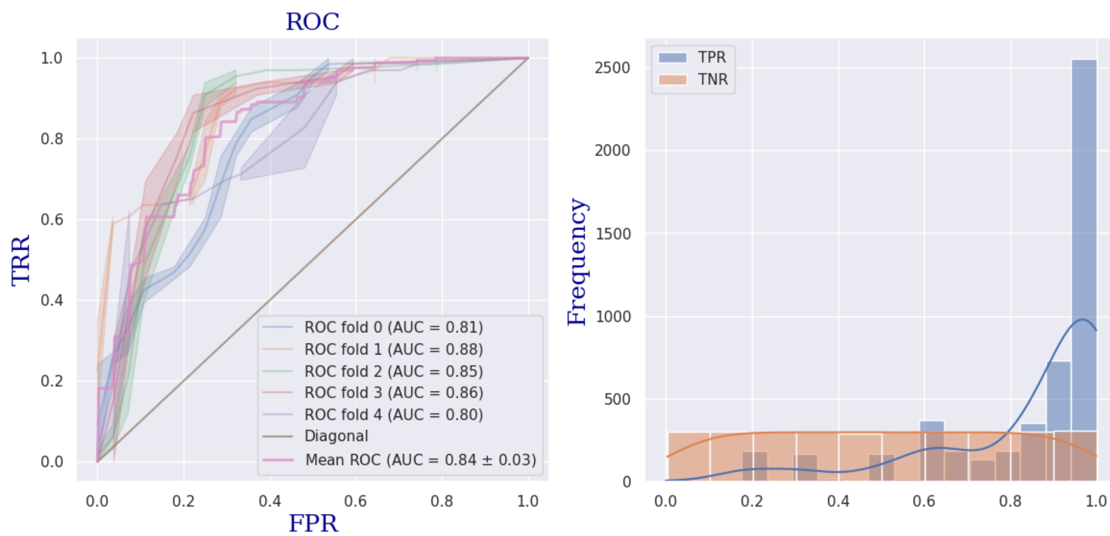


[33]:
```
evaluator.accuracies(model=evaluator.getPipe())
```

```
acc_train: 83.49
acc_val: 79.12
```

[34]:
```
evaluator.val_diags()
```

```
Selected features:  {2: 'cp', 7: 'thalach', 9: 'oldpeak', 11: 'ca', 12: 'thal'}
```



Decision Trees: Decision tree is a popular and interpretable machine learning algorithm used for both classification and regression tasks. It recursively splits the data based on the features, creating a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents a class label or regression output.

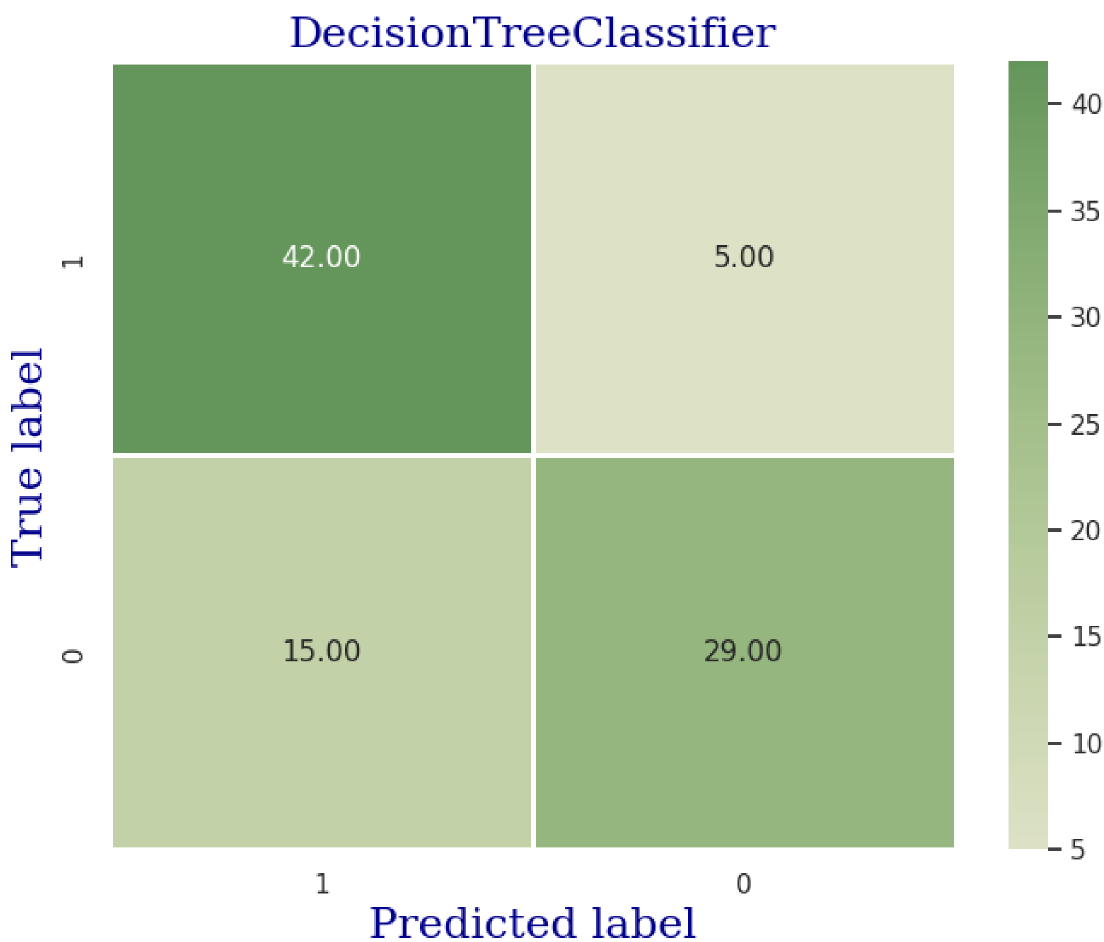
[35]:
```
evaluator = ModelEvaluator(model=tree.DecisionTreeClassifier())

evaluator.accuracies()
```

```
acc_train: 100.00
acc_val: 72.53
```

[36]:
```
evaluator.class_report()
```

	precision	recall	f1-score	support
0	0.852941	0.659091	0.743590	44.00000
1	0.736842	0.893617	0.807692	47.00000
accuracy	0.780220	0.780220	0.780220	0.78022
macro avg	0.794892	0.776354	0.775641	91.00000
weighted avg	0.792978	0.780220	0.776698	91.00000

[37]:
```
evaluator.cfm()
```

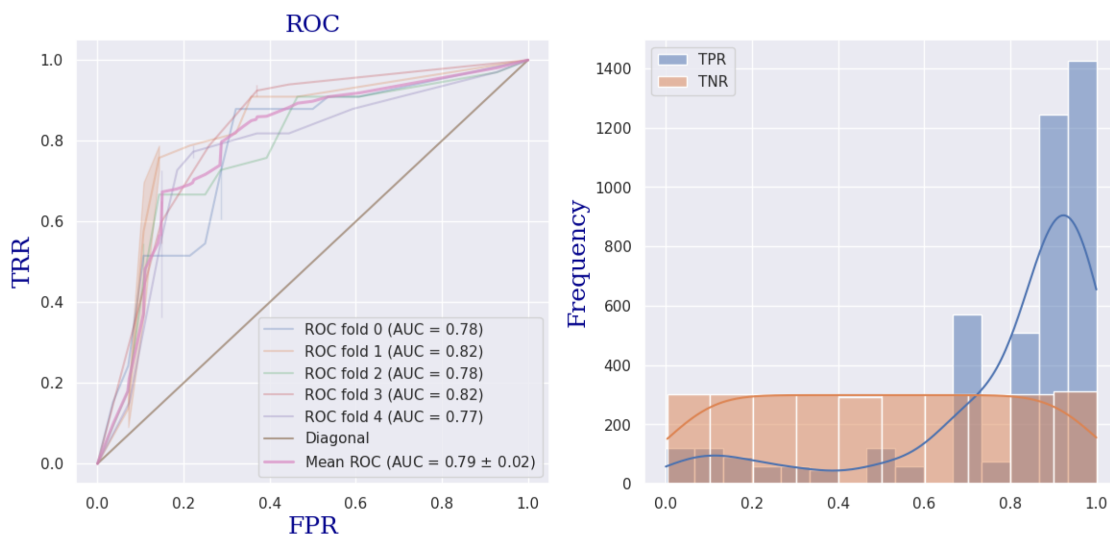


[38]:
```
evaluator.accuracies(model=evaluator.getPipe())
```

```
acc_train: 88.68
acc_val: 74.73
```

[39]:
```
evaluator.val_diags()
```

```
Selected features:  {8: 'exang', 11: 'ca', 12: 'thal'}
```



Random Forests: It is a popular machine learning algorithm used for classification tasks. It is an ensemble learning method that builds multiple decision trees during training and combines their predictions to make the final classification decision. - n_estimators: The number of decision trees to be built in the random forest. Increasing the number of estimators generally improves the model's performance, but it also increases training time and memory requirements. - random_state: The seed used by the random number generator. It is used to ensure reproducibility of results. Setting random_state=0 in the RandomForestClassifier means that the random number generator's seed is fixed to the value 0. This will ensure that the random initialization of the algorithm is the same every time you run the code, leading to consistent results. It effectively removes the randomness in the process, making the results reproducible.

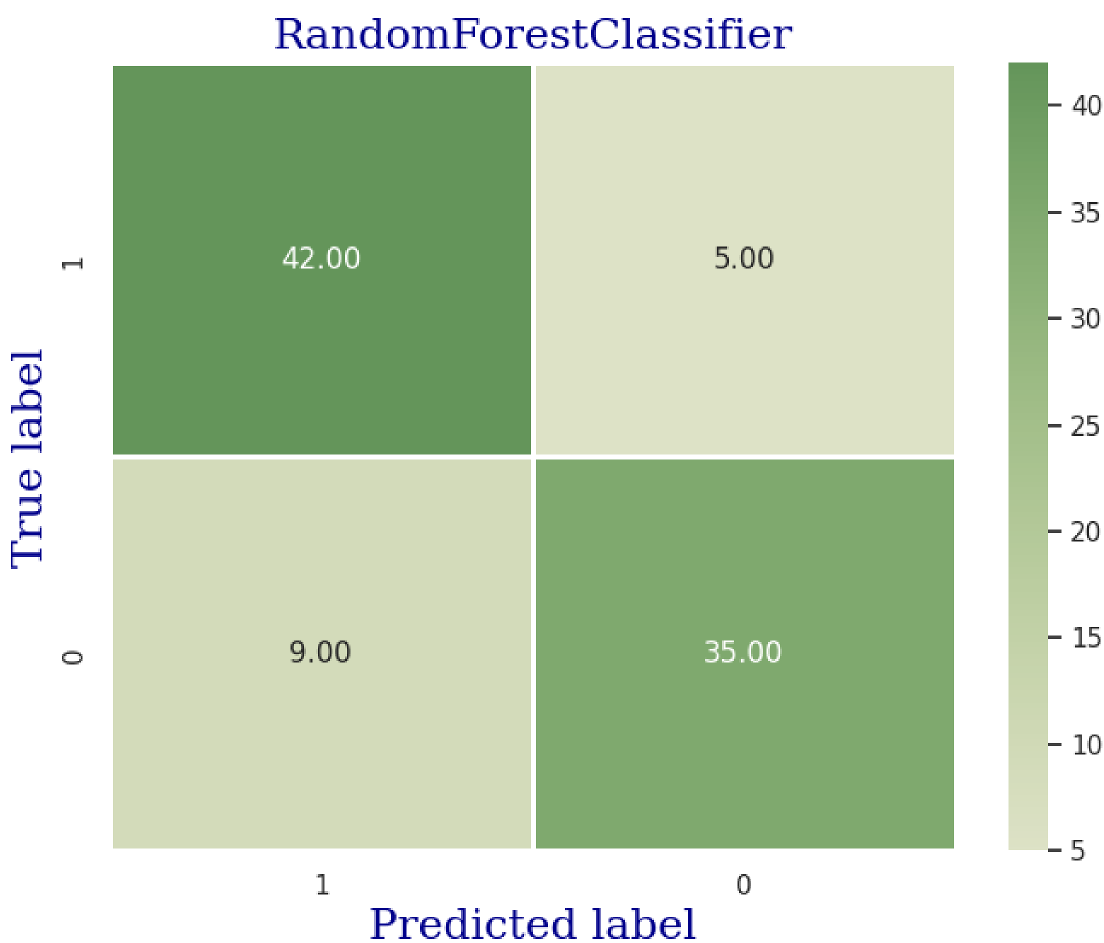
[40]:
```
evaluator = ModelEvaluator( model=RandomForestClassifier(n_estimators=50,␣
  ↪random_state = 0))
evaluator.accuracies()
```

```
acc_train: 100.00
acc_val: 84.62
```

[41]:
```
evaluator.class_report()
```

	precision	recall	f1-score	support
0	0.875000	0.795455	0.833333	44.000000
1	0.823529	0.893617	0.857143	47.000000
accuracy	0.846154	0.846154	0.846154	0.846154
macro avg	0.849265	0.844536	0.845238	91.000000
weighted avg	0.848416	0.846154	0.845631	91.000000

[42]:
```
evaluator.cfm()
```

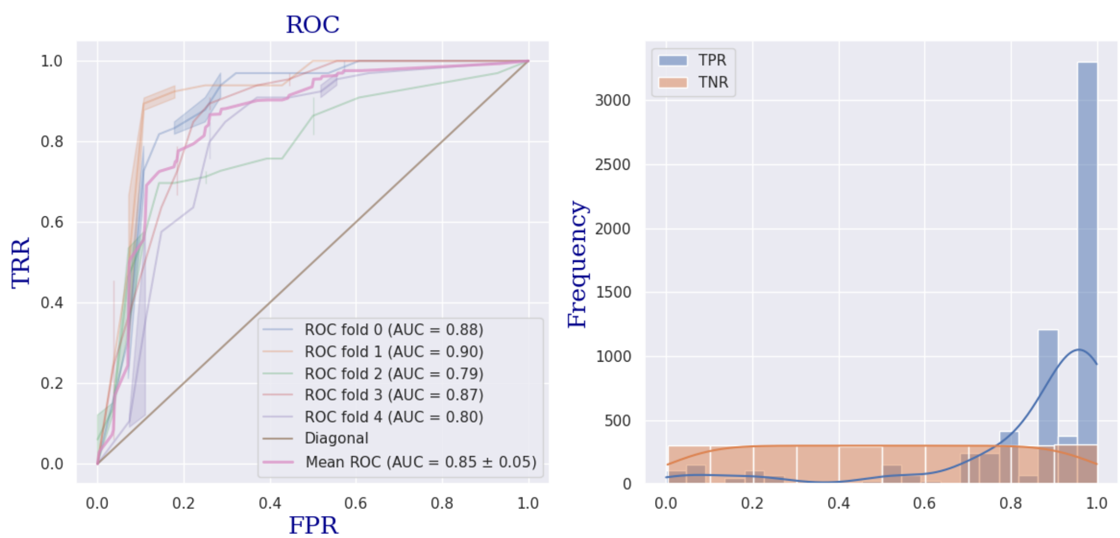


[43]:
```
evaluator.accuracies(model=evaluator.getPipe())
```

```
acc_train: 93.40
acc_val: 68.13
```

[44]:
```
evaluator.val_diags()
```

```
Selected features:  {2: 'cp', 11: 'ca', 12: 'thal'}
```



Support Vector Classification (SVC): SVC specifically refers to the implementation of Support Vector Machines (SVM) for classification tasks. It is used when dealing with labeled data and aims to find the best hyperplane that maximizes the margin between classes. - kernel: This parameter specifies the type of function used to transform the input features into a higher-dimensional space, where the classes can be better separated. In this case, 'linear' indicates that a linear kernel is being used, which means the model is assuming that the data can be separated by a straight line in the feature space. - probability: It indicates whether the SVC model should provide probability estimates for its predictions. When set to True, the model will calculate the probability scores of each class prediction. This can be helpful for tasks like ROC-AUC evaluation or when you want to assess the model's confidence in its predictions.

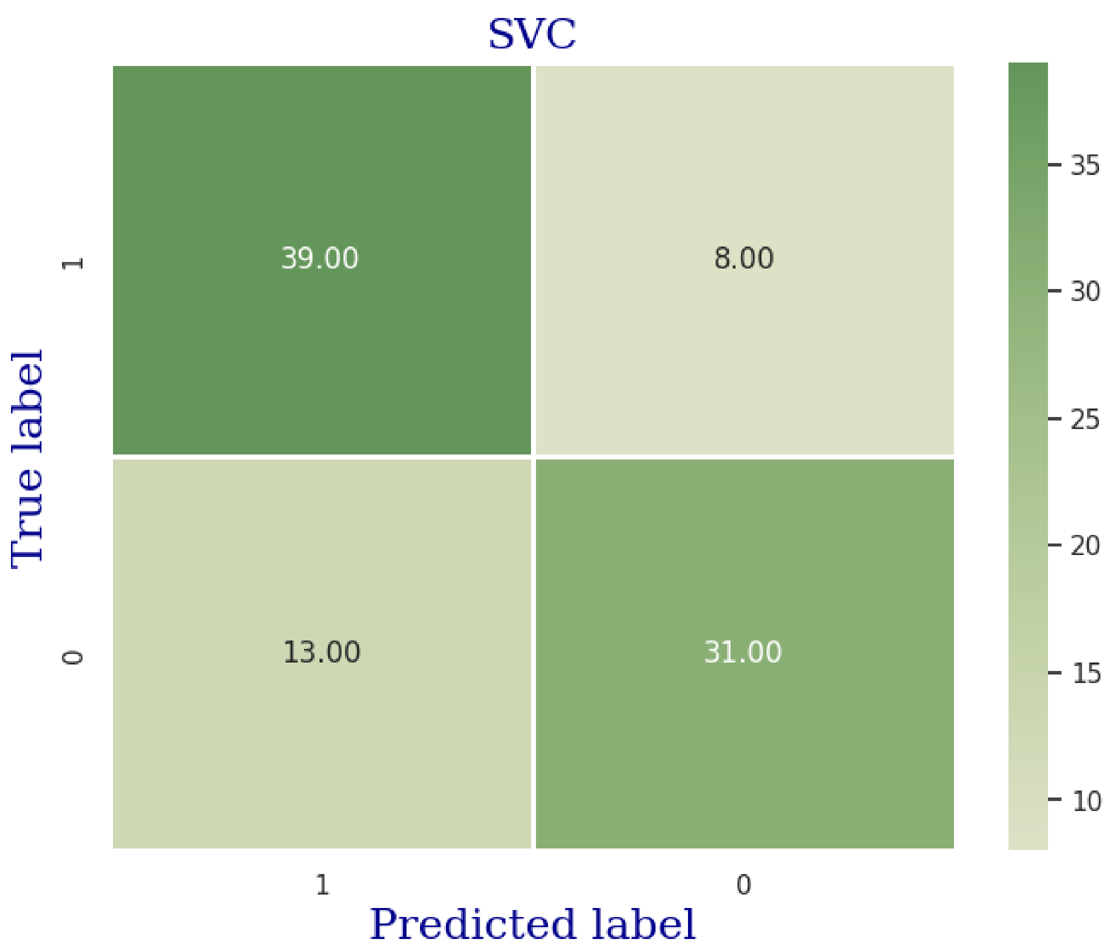
[45]:
```
evaluator = ModelEvaluator(model=SVC(kernel='linear',probability=True))
evaluator.accuracies()
```

```
acc_train: 86.79
acc_val: 80.22
```

[46]:
```
evaluator.class_report()
```

	precision	recall	f1-score	support
0	0.794872	0.704545	0.746988	44.000000
1	0.750000	0.829787	0.787879	47.000000
accuracy	0.769231	0.769231	0.769231	0.769231
macro avg	0.772436	0.767166	0.767433	91.000000
weighted avg	0.771696	0.769231	0.768107	91.000000

[47]:
```
evaluator.cfm()
```

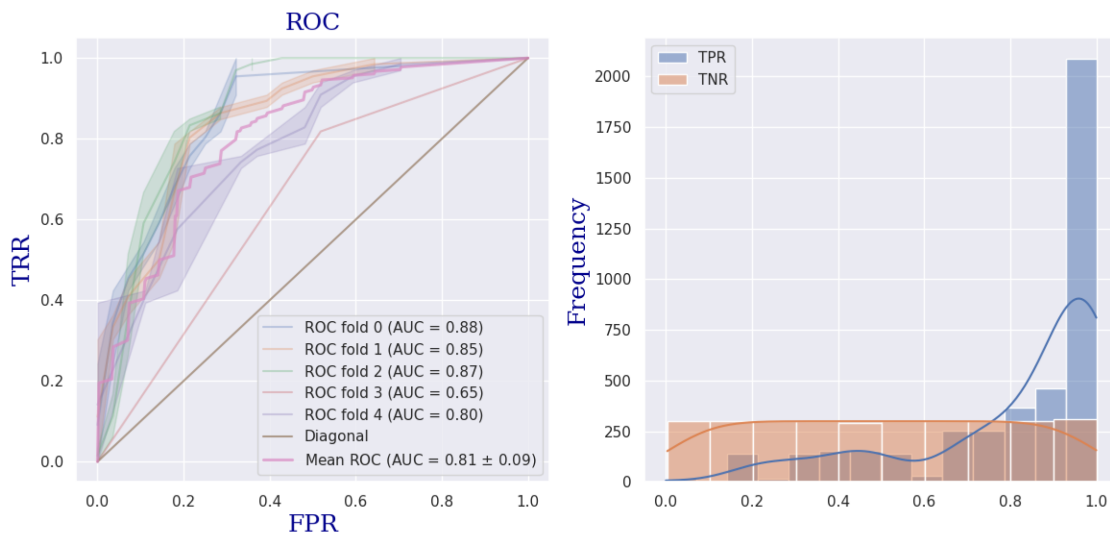


[48]:
```
evaluator.accuracies(model=evaluator.getPipe())
```

```
acc_train: 84.43
acc_val: 79.12
```

[49]:
```
evaluator.val_diags()
```

```
Selected features:  {2: 'cp', 7: 'thalach', 8: 'exang', 11: 'ca'}
```



Logistic Regression: It is a classification algorithm used for binary and multiclass classification tasks. The logistic regression model is a linear classifier that models the probability of a sample belonging to a particular class using the logistic (sigmoid) function.

[50]:
```
evaluator = ModelEvaluator(model= LogisticRegression(max_iter=1000))
evaluator.accuracies()
```

```
acc_train: 86.32
acc_val: 81.32
```

[51]:
```
evaluator.class_report()
```

	precision	recall	f1-score	support
0	0.837838	0.704545	0.765432	44.000000
1	0.759259	0.872340	0.811881	47.000000
accuracy	0.791209	0.791209	0.791209	0.791209
macro avg	0.798549	0.788443	0.788657	91.000000
weighted avg	0.797253	0.791209	0.789422	91.000000

[52]:

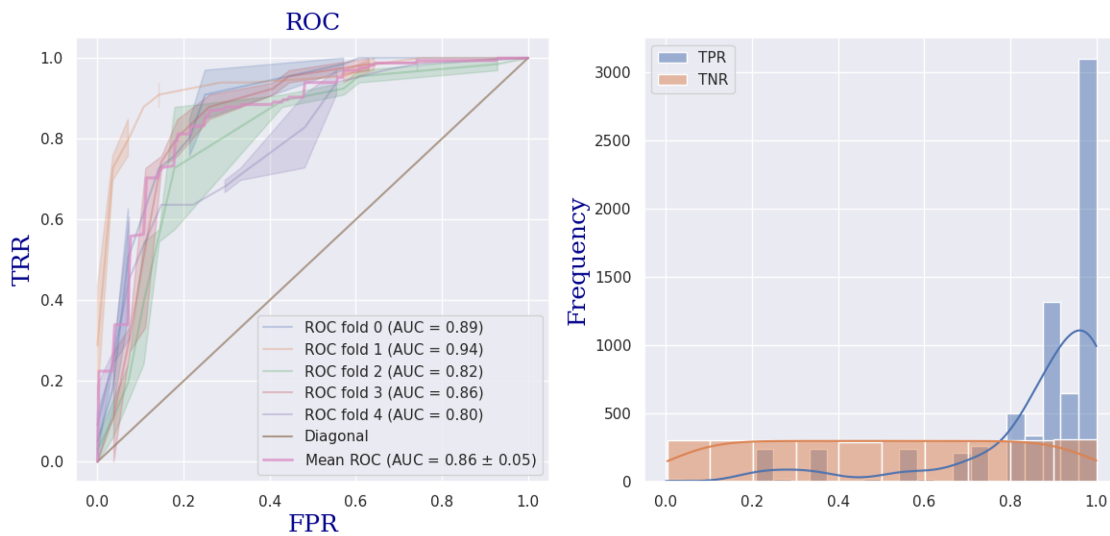
```
evaluator.cfm()
```

[53]:
```
evaluator.accuracies(model=evaluator.getPipe())
```

```
acc_train: 81.60
acc_val: 81.32
```

[54]:
```
evaluator.val_diags()
```

```
Selected features:  {1: 'sex', 2: 'cp', 8: 'exang', 9: 'oldpeak', 11: 'ca'}
```



Gradient Boosting: Gradient Boosting is a powerful machine learning technique used for both regression and classification tasks. It belongs to the family of ensemble learning methods, which combine the predictions of multiple weak learners (usually decision trees) to create a strong predictive model. - loss: This parameter defines the loss function to be optimized during the boosting process. The 'exponential' loss is suitable for classification problems and encourages the model to focus more on misclassified samples. - learning_rate: The learning rate controls the contribution of each weak learner (individual decision tree) to the ensemble. A lower learning rate requires more iterations to reach optimal performance but can help prevent over-fitting. - n_estimators: This is the number of weak learners (decision trees) that will be trained sequentially. Increasing the number of estimators can improve the model's performance, but it also increases computation time. - max_depth: This parameter sets the maximum depth of the individual decision trees. It controls the complexity of the trees. A deeper tree can capture more complex relationships in the data but can also lead to over-fitting.

[55]:
```
evaluator = ModelEvaluator(model=␣
 ↪GradientBoostingClassifier(loss='exponential', learning_rate=0.05,␣
 ↪n_estimators=200, max_depth=6))
evaluator.accuracies()
```

```
acc_train: 100.00
acc_val: 80.22
```

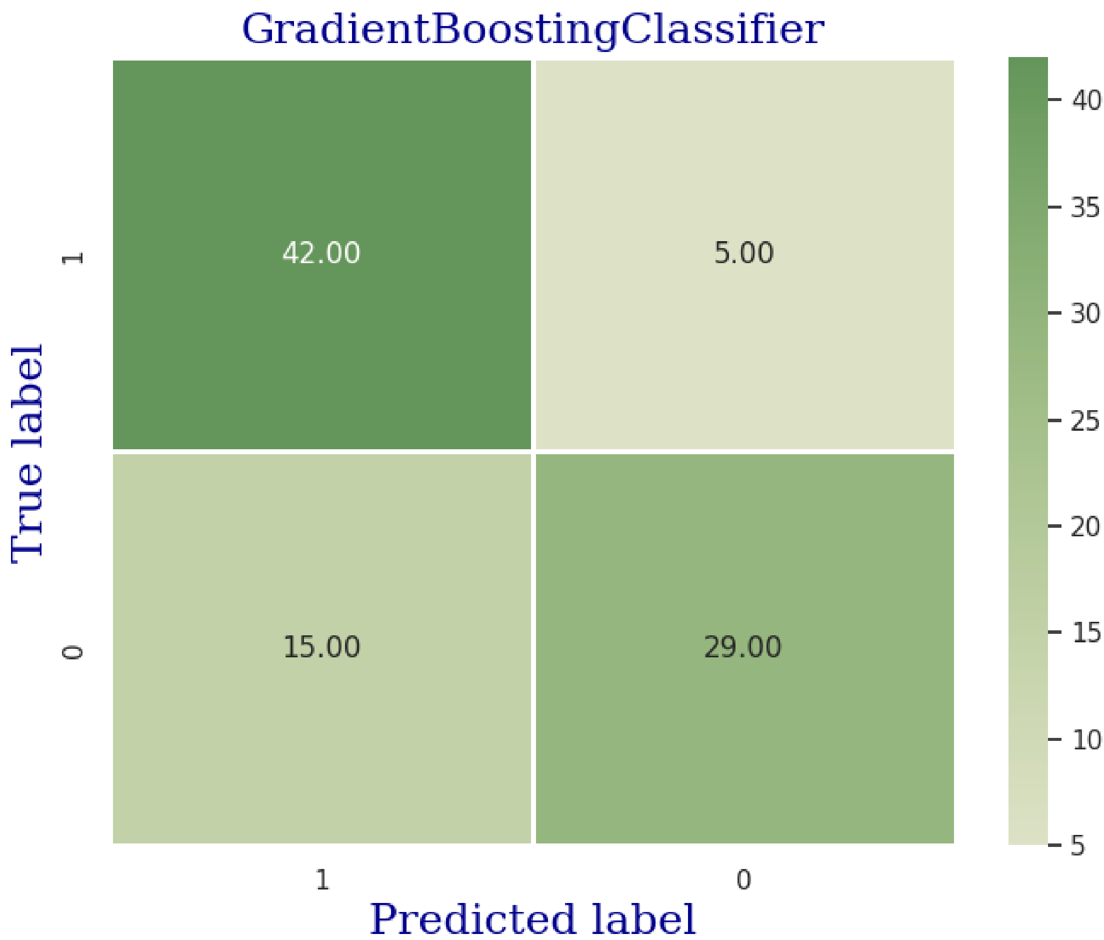
[56]:

```
evaluator.class_report()
```

	precision	recall	f1-score	support
0	0.852941	0.659091	0.743590	44.00000
1	0.736842	0.893617	0.807692	47.00000
accuracy	0.780220	0.780220	0.780220	0.78022
macro avg	0.794892	0.776354	0.775641	91.00000
weighted avg	0.792978	0.780220	0.776698	91.00000

[57]:

```
evaluator.cfm()
```

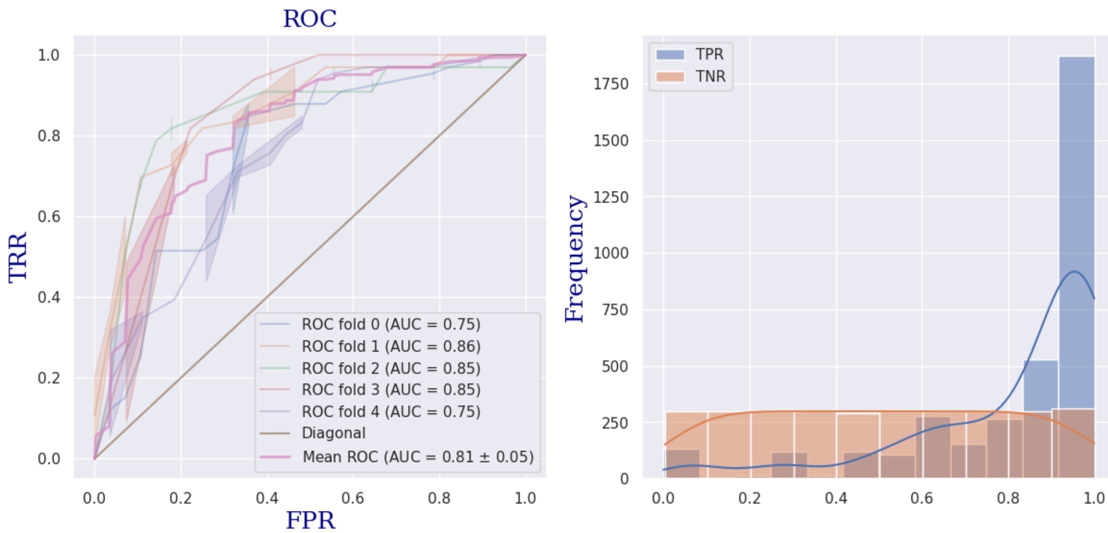


[58]:
```
evaluator.accuracies(model=evaluator.getPipe())
```

```
acc_train: 88.68
acc_val: 74.73
```

[59]:
```
evaluator.val_diags()
```

```
Selected features:  {8: 'exang', 11: 'ca', 12: 'thal'}
```



K-Nearest Neighbors (KNN): This is a class from scikit-learn's neighbors module that represents the K-Nearest Neighbors classifier. KNN is a simple and effective classification algorithm used for both binary and multi-class classification tasks. - n_neighbors: This is a parameter of the KNeighborsClassifier constructor. It specifies the number of neighbors to consider when making predictions. Each data point is classified by a majority vote of its n_neighbors nearest neighbors. The optimal number of neighbors is returned as the output of the function k_opt().

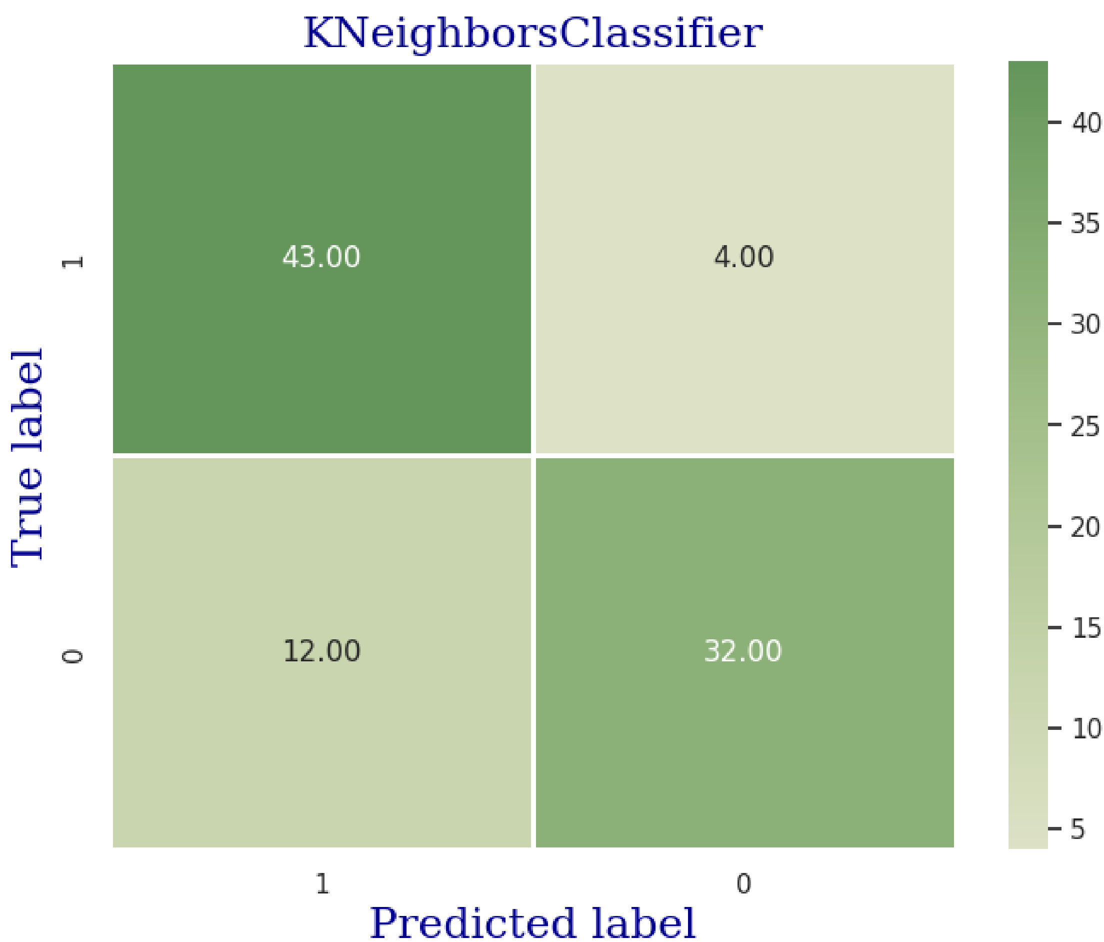
[60]:
```
evaluator = ModelEvaluator(model=KNeighborsClassifier(n_neighbors=evaluator.
  ↪k_opt()))

evaluator.accuracies()
```

```
The optimal number of neighbors is 31 with avg_acc of KNN 83.6%.
acc_train: 84.91
acc_val: 78.02
```

[61]:
```
evaluator.class_report()
```

	precision	recall	f1-score	support
0	0.888889	0.727273	0.800000	44.000000
1	0.781818	0.914894	0.843137	47.000000
accuracy	0.824176	0.824176	0.824176	0.824176
macro avg	0.835354	0.821083	0.821569	91.000000
weighted avg	0.833589	0.824176	0.822280	91.000000

[62]:
```
evaluator.cfm()
```

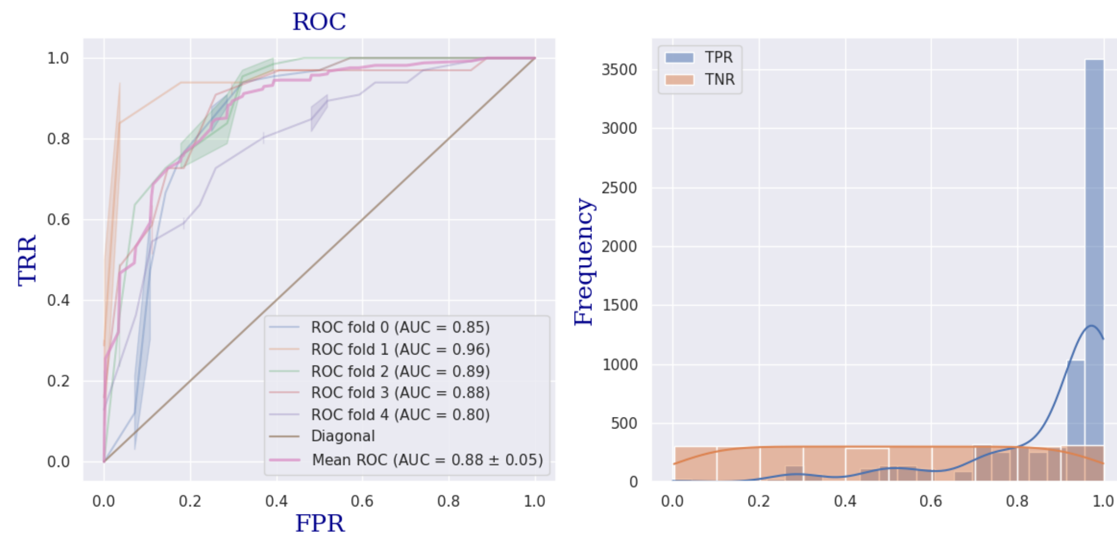


[63]:
```
evaluator.accuracies(model=evaluator.getPipe())
```

```
acc_train: 83.96
acc_val: 80.22
```

[64]:
```
evaluator.val_diags()
```

```
Selected features:  {2: 'cp', 7: 'thalach', 11: 'ca', 12: 'thal'}
```



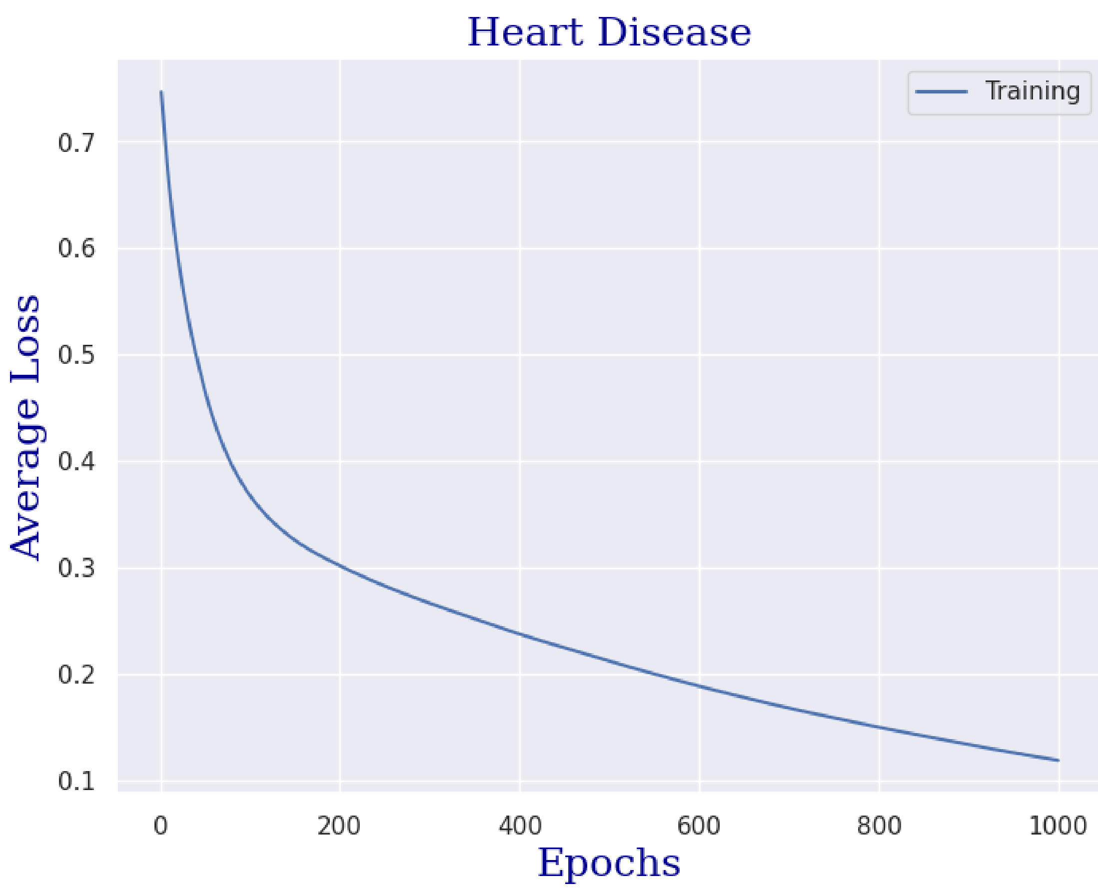
Neural Networks: Neural networks are advanced machine learning models widely used for classification tasks. They are inspired by the human brain's interconnected neurons and are highly effective in solving complex classification problems. In this context, they learn to map input data to different classes, making them invaluable tools for tasks like image recognition, sentiment analysis, disease diagnosis, and more. Neural networks' ability to automatically learn intricate patterns and hierarchies in data makes them a go-to choice when dealing with complex and non-linear classification challenges. To create an effective neural network, consider the following input parameters that influence the model's performance and convergence: - Number of Hidden Units: The capacity of a model to learn intricate patterns depends on the number of hidden units in each layer (they are here set to hu1= 64 and hu2=32 as default). More hidden units can capture complex relationships, but they also risk over-fitting if not properly regularized. Experimentation helps find the ideal balance of hidden units for your specific problem. - Number of Epochs: The epochs determine how many times the model processes the training dataset(it is here set to 1000 as default). Too few epochs may cause under-fitting, while excessive epochs may lead to over-fitting. Monitoring validation performance and stopping training at the right moment can help find the optimal balance. - Learning Rate: The learning rate controls the step size during parameter updates in training (It is here set to 0.001 as default). A higher rate speeds up convergence but might lead to overshooting. Conversely, a smaller rate may result in slow convergence. Adaptive optimizers like Adam can alleviate manual tuning of the learning rate.

These parameters collectively shape the neural network's architecture and training dynamics, guiding it towards effectively learning from data and achieving the desired classification performance.

[47]:
```
evaluator = ModelEvaluator()
```

[49]:
```
#nnw(self, hu1=64, hu2=32, lr=0.001, num_epoch=1000, _tqdm=True)
evaluator.nnw()
```

```
  0%|          | 0/1000 [00:00<?, ?epoch/s]100%|     | 1000/1000
[00:32<00:00, 30.77epoch/s]

Accuracies of the network on test data: 83.52%
Accuracies of the network on training data: 97.64%
```



[50]:
```
evaluator.nnw(_tqdm=False)
```

```
Epoch 1/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.6816 - accuracy:
0.5425
Epoch 2/1000
7/7 [==============================] - 0s 893us/step - loss: 0.6751 - accuracy:
```

```
0.5802
Epoch 3/1000
7/7 [==============================] - 0s 764us/step - loss: 0.6681 - accuracy:
0.5943
Epoch 4/1000
7/7 [==============================] - 0s 822us/step - loss: 0.6611 - accuracy:
0.6274
Epoch 5/1000
7/7 [==============================] - 0s 679us/step - loss: 0.6544 - accuracy:
0.6415
Epoch 6/1000
7/7 [==============================] - 0s 685us/step - loss: 0.6479 - accuracy:
0.6604
Epoch 7/1000
7/7 [==============================] - 0s 851us/step - loss: 0.6417 - accuracy:
0.6887
Epoch 8/1000
7/7 [==============================] - 0s 743us/step - loss: 0.6357 - accuracy:
0.6981
Epoch 9/1000
7/7 [==============================] - 0s 908us/step - loss: 0.6298 - accuracy:
0.7028
Epoch 10/1000
7/7 [==============================] - 0s 806us/step - loss: 0.6242 - accuracy:
0.7075
Epoch 11/1000
7/7 [==============================] - 0s 772us/step - loss: 0.6187 - accuracy:
0.7170
Epoch 12/1000
7/7 [==============================] - 0s 924us/step - loss: 0.6134 - accuracy:
0.7170
Epoch 13/1000
7/7 [==============================] - 0s 663us/step - loss: 0.6081 - accuracy:
0.7170
Epoch 14/1000
7/7 [==============================] - 0s 787us/step - loss: 0.6032 - accuracy:
0.7217
Epoch 15/1000
7/7 [==============================] - 0s 757us/step - loss: 0.5983 - accuracy:
0.7264
Epoch 16/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.5936 - accuracy:
0.7311
Epoch 17/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.5889 - accuracy:
0.7358
Epoch 18/1000
7/7 [==============================] - 0s 921us/step - loss: 0.5843 - accuracy:
```

```
0.7406
Epoch 19/1000
7/7 [==============================] - 0s 881us/step - loss: 0.5798 - accuracy:
0.7453
Epoch 20/1000
7/7 [==============================] - 0s 815us/step - loss: 0.5753 - accuracy:
0.7547
Epoch 21/1000
7/7 [==============================] - 0s 850us/step - loss: 0.5709 - accuracy:
0.7547
Epoch 22/1000
7/7 [==============================] - 0s 954us/step - loss: 0.5666 - accuracy:
0.7594
Epoch 23/1000
7/7 [==============================] - 0s 840us/step - loss: 0.5624 - accuracy:
0.7594
Epoch 24/1000
7/7 [==============================] - 0s 756us/step - loss: 0.5582 - accuracy:
0.7594
Epoch 25/1000
7/7 [==============================] - 0s 680us/step - loss: 0.5541 - accuracy:
0.7594
Epoch 26/1000
7/7 [==============================] - 0s 909us/step - loss: 0.5501 - accuracy:
0.7736
Epoch 27/1000
7/7 [==============================] - 0s 979us/step - loss: 0.5463 - accuracy:
0.7689
Epoch 28/1000
7/7 [==============================] - 0s 680us/step - loss: 0.5423 - accuracy:
0.7689
Epoch 29/1000
7/7 [==============================] - 0s 655us/step - loss: 0.5385 - accuracy:
0.7736
Epoch 30/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.5347 - accuracy:
0.7783
Epoch 31/1000
7/7 [==============================] - 0s 703us/step - loss: 0.5310 - accuracy:
0.7830
Epoch 32/1000
7/7 [==============================] - 0s 734us/step - loss: 0.5275 - accuracy:
0.7783
Epoch 33/1000
7/7 [==============================] - 0s 799us/step - loss: 0.5241 - accuracy:
0.7783
Epoch 34/1000
7/7 [==============================] - 0s 878us/step - loss: 0.5204 - accuracy:
```

```
0.7830
Epoch 35/1000
7/7 [==============================] - 0s 961us/step - loss: 0.5170 - accuracy:
0.7925
Epoch 36/1000
7/7 [==============================] - 0s 637us/step - loss: 0.5137 - accuracy:
0.7877
Epoch 37/1000
7/7 [==============================] - 0s 799us/step - loss: 0.5103 - accuracy:
0.7877
Epoch 38/1000
7/7 [==============================] - 0s 642us/step - loss: 0.5071 - accuracy:
0.7877
Epoch 39/1000
7/7 [==============================] - 0s 671us/step - loss: 0.5038 - accuracy:
0.7877
Epoch 40/1000
7/7 [==============================] - 0s 667us/step - loss: 0.5007 - accuracy:
0.7877
Epoch 41/1000
7/7 [==============================] - 0s 860us/step - loss: 0.4976 - accuracy:
0.7830
Epoch 42/1000
7/7 [==============================] - 0s 652us/step - loss: 0.4945 - accuracy:
0.7830
Epoch 43/1000
7/7 [==============================] - 0s 813us/step - loss: 0.4917 - accuracy:
0.7877
Epoch 44/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.4887 - accuracy:
0.7830
Epoch 45/1000
7/7 [==============================] - 0s 894us/step - loss: 0.4858 - accuracy:
0.7830
Epoch 46/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.4830 - accuracy:
0.7830
Epoch 47/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.4802 - accuracy:
0.7877
Epoch 48/1000
7/7 [==============================] - 0s 784us/step - loss: 0.4775 - accuracy:
0.7972
Epoch 49/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.4749 - accuracy:
0.7972
Epoch 50/1000
7/7 [==============================] - 0s 754us/step - loss: 0.4724 - accuracy:
```

```
0.8019
Epoch 51/1000
7/7 [==============================] - 0s 846us/step - loss: 0.4699 - accuracy:
0.8019
Epoch 52/1000
7/7 [==============================] - 0s 691us/step - loss: 0.4674 - accuracy:
0.8019
Epoch 53/1000
7/7 [==============================] - 0s 826us/step - loss: 0.4648 - accuracy:
0.8066
Epoch 54/1000
7/7 [==============================] - 0s 656us/step - loss: 0.4625 - accuracy:
0.8113
Epoch 55/1000
7/7 [==============================] - 0s 762us/step - loss: 0.4601 - accuracy:
0.8113
Epoch 56/1000
7/7 [==============================] - 0s 715us/step - loss: 0.4578 - accuracy:
0.8113
Epoch 57/1000
7/7 [==============================] - 0s 662us/step - loss: 0.4556 - accuracy:
0.8113
Epoch 58/1000
7/7 [==============================] - 0s 752us/step - loss: 0.4532 - accuracy:
0.8113
Epoch 59/1000
7/7 [==============================] - 0s 643us/step - loss: 0.4510 - accuracy:
0.8160
Epoch 60/1000
7/7 [==============================] - 0s 659us/step - loss: 0.4490 - accuracy:
0.8160
Epoch 61/1000
7/7 [==============================] - 0s 797us/step - loss: 0.4469 - accuracy:
0.8160
Epoch 62/1000
7/7 [==============================] - 0s 679us/step - loss: 0.4446 - accuracy:
0.8160
Epoch 63/1000
7/7 [==============================] - 0s 715us/step - loss: 0.4426 - accuracy:
0.8160
Epoch 64/1000
7/7 [==============================] - 0s 739us/step - loss: 0.4407 - accuracy:
0.8160
Epoch 65/1000
7/7 [==============================] - 0s 625us/step - loss: 0.4386 - accuracy:
0.8208
Epoch 66/1000
7/7 [==============================] - 0s 697us/step - loss: 0.4366 - accuracy:
```

```
0.8208
Epoch 67/1000
7/7 [==============================] - 0s 630us/step - loss: 0.4347 - accuracy:
0.8208
Epoch 68/1000
7/7 [==============================] - 0s 679us/step - loss: 0.4329 - accuracy:
0.8208
Epoch 69/1000
7/7 [==============================] - 0s 694us/step - loss: 0.4309 - accuracy:
0.8208
Epoch 70/1000
7/7 [==============================] - 0s 815us/step - loss: 0.4292 - accuracy:
0.8208
Epoch 71/1000
7/7 [==============================] - 0s 652us/step - loss: 0.4272 - accuracy:
0.8160
Epoch 72/1000
7/7 [==============================] - 0s 782us/step - loss: 0.4254 - accuracy:
0.8160
Epoch 73/1000
7/7 [==============================] - 0s 787us/step - loss: 0.4236 - accuracy:
0.8160
Epoch 74/1000
7/7 [==============================] - 0s 666us/step - loss: 0.4220 - accuracy:
0.8160
Epoch 75/1000
7/7 [==============================] - 0s 808us/step - loss: 0.4203 - accuracy:
0.8160
Epoch 76/1000
7/7 [==============================] - 0s 784us/step - loss: 0.4186 - accuracy:
0.8113
Epoch 77/1000
7/7 [==============================] - 0s 929us/step - loss: 0.4170 - accuracy:
0.8113
Epoch 78/1000
7/7 [==============================] - 0s 664us/step - loss: 0.4153 - accuracy:
0.8113
Epoch 79/1000
7/7 [==============================] - 0s 663us/step - loss: 0.4137 - accuracy:
0.8113
Epoch 80/1000
7/7 [==============================] - 0s 673us/step - loss: 0.4122 - accuracy:
0.8160
Epoch 81/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.4106 - accuracy:
0.8208
Epoch 82/1000
7/7 [==============================] - 0s 904us/step - loss: 0.4090 - accuracy:
```

```
0.8208
Epoch 83/1000
7/7 [==============================] - 0s 770us/step - loss: 0.4075 - accuracy:
0.8208
Epoch 84/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.4060 - accuracy:
0.8160
Epoch 85/1000
7/7 [==============================] - 0s 906us/step - loss: 0.4046 - accuracy:
0.8208
Epoch 86/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.4031 - accuracy:
0.8208
Epoch 87/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.4018 - accuracy:
0.8208
Epoch 88/1000
7/7 [==============================] - 0s 902us/step - loss: 0.4003 - accuracy:
0.8208
Epoch 89/1000
7/7 [==============================] - 0s 902us/step - loss: 0.3990 - accuracy:
0.8208
Epoch 90/1000
7/7 [==============================] - 0s 937us/step - loss: 0.3977 - accuracy:
0.8208
Epoch 91/1000
7/7 [==============================] - 0s 926us/step - loss: 0.3963 - accuracy:
0.8255
Epoch 92/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3951 - accuracy:
0.8255
Epoch 93/1000
7/7 [==============================] - 0s 731us/step - loss: 0.3936 - accuracy:
0.8255
Epoch 94/1000
7/7 [==============================] - 0s 763us/step - loss: 0.3923 - accuracy:
0.8255
Epoch 95/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.3910 - accuracy:
0.8255
Epoch 96/1000
7/7 [==============================] - 0s 802us/step - loss: 0.3899 - accuracy:
0.8349
Epoch 97/1000
7/7 [==============================] - 0s 837us/step - loss: 0.3886 - accuracy:
0.8349
Epoch 98/1000
7/7 [==============================] - 0s 678us/step - loss: 0.3874 - accuracy:
```

```
0.8349
Epoch 99/1000
7/7 [==============================] - 0s 720us/step - loss: 0.3861 - accuracy:
0.8349
Epoch 100/1000
7/7 [==============================] - 0s 694us/step - loss: 0.3850 - accuracy:
0.8349
Epoch 101/1000
7/7 [==============================] - 0s 702us/step - loss: 0.3838 - accuracy:
0.8349
Epoch 102/1000
7/7 [==============================] - 0s 729us/step - loss: 0.3826 - accuracy:
0.8349
Epoch 103/1000
7/7 [==============================] - 0s 609us/step - loss: 0.3815 - accuracy:
0.8349
Epoch 104/1000
7/7 [==============================] - 0s 666us/step - loss: 0.3803 - accuracy:
0.8349
Epoch 105/1000
7/7 [==============================] - 0s 723us/step - loss: 0.3792 - accuracy:
0.8349
Epoch 106/1000
7/7 [==============================] - 0s 705us/step - loss: 0.3782 - accuracy:
0.8396
Epoch 107/1000
7/7 [==============================] - 0s 687us/step - loss: 0.3770 - accuracy:
0.8396
Epoch 108/1000
7/7 [==============================] - 0s 796us/step - loss: 0.3760 - accuracy:
0.8396
Epoch 109/1000
7/7 [==============================] - 0s 611us/step - loss: 0.3748 - accuracy:
0.8396
Epoch 110/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3739 - accuracy:
0.8396
Epoch 111/1000
7/7 [==============================] - 0s 688us/step - loss: 0.3728 - accuracy:
0.8396
Epoch 112/1000
7/7 [==============================] - 0s 688us/step - loss: 0.3719 - accuracy:
0.8396
Epoch 113/1000
7/7 [==============================] - 0s 686us/step - loss: 0.3708 - accuracy:
0.8396
Epoch 114/1000
7/7 [==============================] - 0s 703us/step - loss: 0.3698 - accuracy:
```

```
0.8396
Epoch 115/1000
7/7 [==============================] - 0s 727us/step - loss: 0.3688 - accuracy:
0.8396
Epoch 116/1000
7/7 [==============================] - 0s 719us/step - loss: 0.3678 - accuracy:
0.8396
Epoch 117/1000
7/7 [==============================] - 0s 720us/step - loss: 0.3668 - accuracy:
0.8396
Epoch 118/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3658 - accuracy:
0.8396
Epoch 119/1000
7/7 [==============================] - 0s 671us/step - loss: 0.3648 - accuracy:
0.8396
Epoch 120/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3638 - accuracy:
0.8396
Epoch 121/1000
7/7 [==============================] - 0s 957us/step - loss: 0.3628 - accuracy:
0.8396
Epoch 122/1000
7/7 [==============================] - 0s 894us/step - loss: 0.3619 - accuracy:
0.8443
Epoch 123/1000
7/7 [==============================] - 0s 800us/step - loss: 0.3610 - accuracy:
0.8443
Epoch 124/1000
7/7 [==============================] - 0s 657us/step - loss: 0.3600 - accuracy:
0.8443
Epoch 125/1000
7/7 [==============================] - 0s 782us/step - loss: 0.3591 - accuracy:
0.8491
Epoch 126/1000
7/7 [==============================] - 0s 747us/step - loss: 0.3581 - accuracy:
0.8491
Epoch 127/1000
7/7 [==============================] - 0s 720us/step - loss: 0.3572 - accuracy:
0.8491
Epoch 128/1000
7/7 [==============================] - 0s 712us/step - loss: 0.3563 - accuracy:
0.8491
Epoch 129/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.3554 - accuracy:
0.8491
Epoch 130/1000
7/7 [==============================] - 0s 727us/step - loss: 0.3545 - accuracy:
```

```
0.8491
Epoch 131/1000
7/7 [==============================] - 0s 903us/step - loss: 0.3537 - accuracy:
0.8491
Epoch 132/1000
7/7 [==============================] - 0s 798us/step - loss: 0.3529 - accuracy:
0.8491
Epoch 133/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3520 - accuracy:
0.8491
Epoch 134/1000
7/7 [==============================] - 0s 741us/step - loss: 0.3512 - accuracy:
0.8538
Epoch 135/1000
7/7 [==============================] - 0s 886us/step - loss: 0.3503 - accuracy:
0.8538
Epoch 136/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3496 - accuracy:
0.8538
Epoch 137/1000
7/7 [==============================] - 0s 939us/step - loss: 0.3487 - accuracy:
0.8538
Epoch 138/1000
7/7 [==============================] - 0s 931us/step - loss: 0.3479 - accuracy:
0.8538
Epoch 139/1000
7/7 [==============================] - 0s 784us/step - loss: 0.3471 - accuracy:
0.8538
Epoch 140/1000
7/7 [==============================] - 0s 981us/step - loss: 0.3464 - accuracy:
0.8538
Epoch 141/1000
7/7 [==============================] - 0s 933us/step - loss: 0.3457 - accuracy:
0.8538
Epoch 142/1000
7/7 [==============================] - 0s 905us/step - loss: 0.3449 - accuracy:
0.8538
Epoch 143/1000
7/7 [==============================] - 0s 656us/step - loss: 0.3441 - accuracy:
0.8538
Epoch 144/1000
7/7 [==============================] - 0s 673us/step - loss: 0.3434 - accuracy:
0.8585
Epoch 145/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3425 - accuracy:
0.8585
Epoch 146/1000
7/7 [==============================] - 0s 717us/step - loss: 0.3418 - accuracy:
```

```
0.8585
Epoch 147/1000
7/7 [==============================] - 0s 800us/step - loss: 0.3411 - accuracy:
0.8585
Epoch 148/1000
7/7 [==============================] - 0s 632us/step - loss: 0.3404 - accuracy:
0.8585
Epoch 149/1000
7/7 [==============================] - 0s 708us/step - loss: 0.3397 - accuracy:
0.8585
Epoch 150/1000
7/7 [==============================] - 0s 742us/step - loss: 0.3390 - accuracy:
0.8632
Epoch 151/1000
7/7 [==============================] - 0s 775us/step - loss: 0.3383 - accuracy:
0.8632
Epoch 152/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3375 - accuracy:
0.8632
Epoch 153/1000
7/7 [==============================] - 0s 961us/step - loss: 0.3369 - accuracy:
0.8632
Epoch 154/1000
7/7 [==============================] - 0s 937us/step - loss: 0.3362 - accuracy:
0.8632
Epoch 155/1000
7/7 [==============================] - 0s 726us/step - loss: 0.3355 - accuracy:
0.8632
Epoch 156/1000
7/7 [==============================] - 0s 719us/step - loss: 0.3347 - accuracy:
0.8632
Epoch 157/1000
7/7 [==============================] - 0s 728us/step - loss: 0.3342 - accuracy:
0.8632
Epoch 158/1000
7/7 [==============================] - 0s 692us/step - loss: 0.3334 - accuracy:
0.8632
Epoch 159/1000
7/7 [==============================] - 0s 775us/step - loss: 0.3328 - accuracy:
0.8632
Epoch 160/1000
7/7 [==============================] - 0s 668us/step - loss: 0.3321 - accuracy:
0.8632
Epoch 161/1000
7/7 [==============================] - 0s 894us/step - loss: 0.3315 - accuracy:
0.8632
Epoch 162/1000
7/7 [==============================] - 0s 873us/step - loss: 0.3308 - accuracy:
```

```
0.8632
Epoch 163/1000
7/7 [==============================] - 0s 677us/step - loss: 0.3301 - accuracy:
0.8632
Epoch 164/1000
7/7 [==============================] - 0s 796us/step - loss: 0.3296 - accuracy:
0.8632
Epoch 165/1000
7/7 [==============================] - 0s 783us/step - loss: 0.3289 - accuracy:
0.8632
Epoch 166/1000
7/7 [==============================] - 0s 641us/step - loss: 0.3283 - accuracy:
0.8632
Epoch 167/1000
7/7 [==============================] - 0s 710us/step - loss: 0.3277 - accuracy:
0.8632
Epoch 168/1000
7/7 [==============================] - 0s 726us/step - loss: 0.3271 - accuracy:
0.8632
Epoch 169/1000
7/7 [==============================] - 0s 712us/step - loss: 0.3265 - accuracy:
0.8632
Epoch 170/1000
7/7 [==============================] - 0s 825us/step - loss: 0.3259 - accuracy:
0.8632
Epoch 171/1000
7/7 [==============================] - 0s 701us/step - loss: 0.3253 - accuracy:
0.8585
Epoch 172/1000
7/7 [==============================] - 0s 774us/step - loss: 0.3248 - accuracy:
0.8585
Epoch 173/1000
7/7 [==============================] - 0s 800us/step - loss: 0.3241 - accuracy:
0.8585
Epoch 174/1000
7/7 [==============================] - 0s 685us/step - loss: 0.3236 - accuracy:
0.8632
Epoch 175/1000
7/7 [==============================] - 0s 711us/step - loss: 0.3231 - accuracy:
0.8632
Epoch 176/1000
7/7 [==============================] - 0s 612us/step - loss: 0.3226 - accuracy:
0.8632
Epoch 177/1000
7/7 [==============================] - 0s 709us/step - loss: 0.3220 - accuracy:
0.8632
Epoch 178/1000
7/7 [==============================] - 0s 679us/step - loss: 0.3215 - accuracy:
```

```
0.8632
Epoch 179/1000
7/7 [==============================] - 0s 660us/step - loss: 0.3210 - accuracy:
0.8632
Epoch 180/1000
7/7 [==============================] - 0s 650us/step - loss: 0.3204 - accuracy:
0.8632
Epoch 181/1000
7/7 [==============================] - 0s 752us/step - loss: 0.3200 - accuracy:
0.8632
Epoch 182/1000
7/7 [==============================] - 0s 682us/step - loss: 0.3194 - accuracy:
0.8632
Epoch 183/1000
7/7 [==============================] - 0s 751us/step - loss: 0.3189 - accuracy:
0.8632
Epoch 184/1000
7/7 [==============================] - 0s 691us/step - loss: 0.3184 - accuracy:
0.8632
Epoch 185/1000
7/7 [==============================] - 0s 699us/step - loss: 0.3179 - accuracy:
0.8632
Epoch 186/1000
7/7 [==============================] - 0s 686us/step - loss: 0.3174 - accuracy:
0.8632
Epoch 187/1000
7/7 [==============================] - 0s 702us/step - loss: 0.3169 - accuracy:
0.8632
Epoch 188/1000
7/7 [==============================] - 0s 653us/step - loss: 0.3164 - accuracy:
0.8679
Epoch 189/1000
7/7 [==============================] - 0s 706us/step - loss: 0.3160 - accuracy:
0.8679
Epoch 190/1000
7/7 [==============================] - 0s 665us/step - loss: 0.3155 - accuracy:
0.8679
Epoch 191/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3150 - accuracy:
0.8679
Epoch 192/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3145 - accuracy:
0.8679
Epoch 193/1000
7/7 [==============================] - 0s 751us/step - loss: 0.3142 - accuracy:
0.8679
Epoch 194/1000
7/7 [==============================] - 0s 721us/step - loss: 0.3136 - accuracy:
```

```
0.8679
Epoch 195/1000
7/7 [==============================] - 0s 644us/step - loss: 0.3131 - accuracy:
0.8679
Epoch 196/1000
7/7 [==============================] - 0s 722us/step - loss: 0.3127 - accuracy:
0.8679
Epoch 197/1000
7/7 [==============================] - 0s 746us/step - loss: 0.3122 - accuracy:
0.8679
Epoch 198/1000
7/7 [==============================] - 0s 721us/step - loss: 0.3118 - accuracy:
0.8679
Epoch 199/1000
7/7 [==============================] - 0s 737us/step - loss: 0.3113 - accuracy:
0.8679
Epoch 200/1000
7/7 [==============================] - 0s 647us/step - loss: 0.3109 - accuracy:
0.8679
Epoch 201/1000
7/7 [==============================] - 0s 751us/step - loss: 0.3104 - accuracy:
0.8679
Epoch 202/1000
7/7 [==============================] - 0s 767us/step - loss: 0.3099 - accuracy:
0.8679
Epoch 203/1000
7/7 [==============================] - 0s 692us/step - loss: 0.3095 - accuracy:
0.8679
Epoch 204/1000
7/7 [==============================] - 0s 866us/step - loss: 0.3089 - accuracy:
0.8679
Epoch 205/1000
7/7 [==============================] - 0s 787us/step - loss: 0.3086 - accuracy:
0.8679
Epoch 206/1000
7/7 [==============================] - 0s 676us/step - loss: 0.3080 - accuracy:
0.8679
Epoch 207/1000
7/7 [==============================] - 0s 670us/step - loss: 0.3077 - accuracy:
0.8679
Epoch 208/1000
7/7 [==============================] - 0s 662us/step - loss: 0.3072 - accuracy:
0.8679
Epoch 209/1000
7/7 [==============================] - 0s 747us/step - loss: 0.3068 - accuracy:
0.8679
Epoch 210/1000
7/7 [==============================] - 0s 657us/step - loss: 0.3063 - accuracy:
```

```
0.8726
Epoch 211/1000
7/7 [==============================] - 0s 696us/step - loss: 0.3059 - accuracy:
0.8774
Epoch 212/1000
7/7 [==============================] - 0s 674us/step - loss: 0.3055 - accuracy:
0.8774
Epoch 213/1000
7/7 [==============================] - 0s 721us/step - loss: 0.3050 - accuracy:
0.8774
Epoch 214/1000
7/7 [==============================] - 0s 634us/step - loss: 0.3046 - accuracy:
0.8726
Epoch 215/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.3042 - accuracy:
0.8726
Epoch 216/1000
7/7 [==============================] - 0s 676us/step - loss: 0.3038 - accuracy:
0.8726
Epoch 217/1000
7/7 [==============================] - 0s 703us/step - loss: 0.3034 - accuracy:
0.8726
Epoch 218/1000
7/7 [==============================] - 0s 798us/step - loss: 0.3030 - accuracy:
0.8726
Epoch 219/1000
7/7 [==============================] - 0s 728us/step - loss: 0.3026 - accuracy:
0.8726
Epoch 220/1000
7/7 [==============================] - 0s 641us/step - loss: 0.3021 - accuracy:
0.8726
Epoch 221/1000
7/7 [==============================] - 0s 701us/step - loss: 0.3017 - accuracy:
0.8726
Epoch 222/1000
7/7 [==============================] - 0s 609us/step - loss: 0.3012 - accuracy:
0.8726
Epoch 223/1000
7/7 [==============================] - 0s 710us/step - loss: 0.3009 - accuracy:
0.8726
Epoch 224/1000
7/7 [==============================] - 0s 657us/step - loss: 0.3005 - accuracy:
0.8726
Epoch 225/1000
7/7 [==============================] - 0s 624us/step - loss: 0.3001 - accuracy:
0.8726
Epoch 226/1000
7/7 [==============================] - 0s 631us/step - loss: 0.2997 - accuracy:
```

```
0.8726
Epoch 227/1000
7/7 [==============================] - 0s 675us/step - loss: 0.2993 - accuracy:
0.8726
Epoch 228/1000
7/7 [==============================] - 0s 719us/step - loss: 0.2989 - accuracy:
0.8726
Epoch 229/1000
7/7 [==============================] - 0s 737us/step - loss: 0.2985 - accuracy:
0.8726
Epoch 230/1000
7/7 [==============================] - 0s 691us/step - loss: 0.2981 - accuracy:
0.8726
Epoch 231/1000
7/7 [==============================] - 0s 680us/step - loss: 0.2977 - accuracy:
0.8726
Epoch 232/1000
7/7 [==============================] - 0s 701us/step - loss: 0.2973 - accuracy:
0.8726
Epoch 233/1000
7/7 [==============================] - 0s 695us/step - loss: 0.2970 - accuracy:
0.8726
Epoch 234/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2966 - accuracy:
0.8726
Epoch 235/1000
7/7 [==============================] - 0s 848us/step - loss: 0.2962 - accuracy:
0.8726
Epoch 236/1000
7/7 [==============================] - 0s 711us/step - loss: 0.2958 - accuracy:
0.8726
Epoch 237/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2955 - accuracy:
0.8726
Epoch 238/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2952 - accuracy:
0.8726
Epoch 239/1000
7/7 [==============================] - 0s 864us/step - loss: 0.2947 - accuracy:
0.8726
Epoch 240/1000
7/7 [==============================] - 0s 890us/step - loss: 0.2944 - accuracy:
0.8726
Epoch 241/1000
7/7 [==============================] - 0s 726us/step - loss: 0.2940 - accuracy:
0.8726
Epoch 242/1000
7/7 [==============================] - 0s 714us/step - loss: 0.2937 - accuracy:
```

```
0.8726
Epoch 243/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2933 - accuracy:
0.8726
Epoch 244/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2930 - accuracy:
0.8726
Epoch 245/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2925 - accuracy:
0.8726
Epoch 246/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2922 - accuracy:
0.8726
Epoch 247/1000
7/7 [==============================] - 0s 943us/step - loss: 0.2918 - accuracy:
0.8726
Epoch 248/1000
7/7 [==============================] - 0s 988us/step - loss: 0.2915 - accuracy:
0.8726
Epoch 249/1000
7/7 [==============================] - 0s 679us/step - loss: 0.2912 - accuracy:
0.8726
Epoch 250/1000
7/7 [==============================] - 0s 734us/step - loss: 0.2908 - accuracy:
0.8726
Epoch 251/1000
7/7 [==============================] - 0s 715us/step - loss: 0.2905 - accuracy:
0.8726
Epoch 252/1000
7/7 [==============================] - 0s 695us/step - loss: 0.2901 - accuracy:
0.8726
Epoch 253/1000
7/7 [==============================] - 0s 877us/step - loss: 0.2897 - accuracy:
0.8774
Epoch 254/1000
7/7 [==============================] - 0s 693us/step - loss: 0.2894 - accuracy:
0.8774
Epoch 255/1000
7/7 [==============================] - 0s 843us/step - loss: 0.2890 - accuracy:
0.8774
Epoch 256/1000
7/7 [==============================] - 0s 797us/step - loss: 0.2887 - accuracy:
0.8774
Epoch 257/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2884 - accuracy:
0.8774
Epoch 258/1000
7/7 [==============================] - 0s 807us/step - loss: 0.2881 - accuracy:
```

```
0.8774
Epoch 259/1000
7/7 [==============================] - 0s 679us/step - loss: 0.2877 - accuracy:
0.8774
Epoch 260/1000
7/7 [==============================] - 0s 766us/step - loss: 0.2873 - accuracy:
0.8774
Epoch 261/1000
7/7 [==============================] - 0s 724us/step - loss: 0.2870 - accuracy:
0.8774
Epoch 262/1000
7/7 [==============================] - 0s 838us/step - loss: 0.2867 - accuracy:
0.8774
Epoch 263/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2863 - accuracy:
0.8774
Epoch 264/1000
7/7 [==============================] - 0s 910us/step - loss: 0.2860 - accuracy:
0.8774
Epoch 265/1000
7/7 [==============================] - 0s 775us/step - loss: 0.2856 - accuracy:
0.8774
Epoch 266/1000
7/7 [==============================] - 0s 644us/step - loss: 0.2854 - accuracy:
0.8821
Epoch 267/1000
7/7 [==============================] - 0s 674us/step - loss: 0.2850 - accuracy:
0.8821
Epoch 268/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2847 - accuracy:
0.8821
Epoch 269/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2844 - accuracy:
0.8821
Epoch 270/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2840 - accuracy:
0.8821
Epoch 271/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2837 - accuracy:
0.8821
Epoch 272/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2834 - accuracy:
0.8821
Epoch 273/1000
7/7 [==============================] - 0s 735us/step - loss: 0.2830 - accuracy:
0.8821
Epoch 274/1000
7/7 [==============================] - 0s 929us/step - loss: 0.2827 - accuracy:
```

```
0.8821
Epoch 275/1000
7/7 [==============================] - 0s 869us/step - loss: 0.2824 - accuracy:
0.8821
Epoch 276/1000
7/7 [==============================] - 0s 842us/step - loss: 0.2820 - accuracy:
0.8821
Epoch 277/1000
7/7 [==============================] - 0s 621us/step - loss: 0.2818 - accuracy:
0.8821
Epoch 278/1000
7/7 [==============================] - 0s 742us/step - loss: 0.2814 - accuracy:
0.8821
Epoch 279/1000
7/7 [==============================] - 0s 690us/step - loss: 0.2812 - accuracy:
0.8821
Epoch 280/1000
7/7 [==============================] - 0s 855us/step - loss: 0.2808 - accuracy:
0.8821
Epoch 281/1000
7/7 [==============================] - 0s 725us/step - loss: 0.2805 - accuracy:
0.8821
Epoch 282/1000
7/7 [==============================] - 0s 729us/step - loss: 0.2802 - accuracy:
0.8821
Epoch 283/1000
7/7 [==============================] - 0s 856us/step - loss: 0.2799 - accuracy:
0.8821
Epoch 284/1000
7/7 [==============================] - 0s 719us/step - loss: 0.2796 - accuracy:
0.8868
Epoch 285/1000
7/7 [==============================] - 0s 765us/step - loss: 0.2792 - accuracy:
0.8821
Epoch 286/1000
7/7 [==============================] - 0s 752us/step - loss: 0.2789 - accuracy:
0.8868
Epoch 287/1000
7/7 [==============================] - 0s 678us/step - loss: 0.2786 - accuracy:
0.8868
Epoch 288/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2783 - accuracy:
0.8821
Epoch 289/1000
7/7 [==============================] - 0s 896us/step - loss: 0.2780 - accuracy:
0.8868
Epoch 290/1000
7/7 [==============================] - 0s 887us/step - loss: 0.2777 - accuracy:
```

```
0.8868
Epoch 291/1000
7/7 [==============================] - 0s 912us/step - loss: 0.2774 - accuracy:
0.8868
Epoch 292/1000
7/7 [==============================] - 0s 999us/step - loss: 0.2771 - accuracy:
0.8868
Epoch 293/1000
7/7 [==============================] - 0s 892us/step - loss: 0.2768 - accuracy:
0.8868
Epoch 294/1000
7/7 [==============================] - 0s 954us/step - loss: 0.2765 - accuracy:
0.8868
Epoch 295/1000
7/7 [==============================] - 0s 979us/step - loss: 0.2762 - accuracy:
0.8868
Epoch 296/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2758 - accuracy:
0.8868
Epoch 297/1000
7/7 [==============================] - 0s 3ms/step - loss: 0.2756 - accuracy:
0.8868
Epoch 298/1000
7/7 [==============================] - 0s 959us/step - loss: 0.2753 - accuracy:
0.8868
Epoch 299/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2750 - accuracy:
0.8868
Epoch 300/1000
7/7 [==============================] - 0s 715us/step - loss: 0.2746 - accuracy:
0.8868
Epoch 301/1000
7/7 [==============================] - 0s 880us/step - loss: 0.2743 - accuracy:
0.8868
Epoch 302/1000
7/7 [==============================] - 0s 721us/step - loss: 0.2741 - accuracy:
0.8868
Epoch 303/1000
7/7 [==============================] - 0s 802us/step - loss: 0.2738 - accuracy:
0.8868
Epoch 304/1000
7/7 [==============================] - 0s 684us/step - loss: 0.2735 - accuracy:
0.8868
Epoch 305/1000
7/7 [==============================] - 0s 711us/step - loss: 0.2733 - accuracy:
0.8868
Epoch 306/1000
7/7 [==============================] - 0s 812us/step - loss: 0.2729 - accuracy:
```

```
0.8868
Epoch 307/1000
7/7 [==============================] - 0s 733us/step - loss: 0.2726 - accuracy:
0.8868
Epoch 308/1000
7/7 [==============================] - 0s 727us/step - loss: 0.2723 - accuracy:
0.8868
Epoch 309/1000
7/7 [==============================] - 0s 667us/step - loss: 0.2721 - accuracy:
0.8868
Epoch 310/1000
7/7 [==============================] - 0s 616us/step - loss: 0.2717 - accuracy:
0.8868
Epoch 311/1000
7/7 [==============================] - 0s 646us/step - loss: 0.2714 - accuracy:
0.8868
Epoch 312/1000
7/7 [==============================] - 0s 646us/step - loss: 0.2711 - accuracy:
0.8868
Epoch 313/1000
7/7 [==============================] - 0s 694us/step - loss: 0.2708 - accuracy:
0.8868
Epoch 314/1000
7/7 [==============================] - 0s 996us/step - loss: 0.2705 - accuracy:
0.8868
Epoch 315/1000
7/7 [==============================] - 0s 708us/step - loss: 0.2702 - accuracy:
0.8915
Epoch 316/1000
7/7 [==============================] - 0s 783us/step - loss: 0.2699 - accuracy:
0.8915
Epoch 317/1000
7/7 [==============================] - 0s 793us/step - loss: 0.2696 - accuracy:
0.8915
Epoch 318/1000
7/7 [==============================] - 0s 622us/step - loss: 0.2694 - accuracy:
0.8915
Epoch 319/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2690 - accuracy:
0.8915
Epoch 320/1000
7/7 [==============================] - 0s 917us/step - loss: 0.2687 - accuracy:
0.8915
Epoch 321/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2684 - accuracy:
0.8915
Epoch 322/1000
7/7 [==============================] - 0s 700us/step - loss: 0.2681 - accuracy:
```

```
0.8915
Epoch 323/1000
7/7 [==============================] - 0s 778us/step - loss: 0.2679 - accuracy:
0.8915
Epoch 324/1000
7/7 [==============================] - 0s 684us/step - loss: 0.2675 - accuracy:
0.8915
Epoch 325/1000
7/7 [==============================] - 0s 944us/step - loss: 0.2672 - accuracy:
0.8915
Epoch 326/1000
7/7 [==============================] - 0s 927us/step - loss: 0.2669 - accuracy:
0.8915
Epoch 327/1000
7/7 [==============================] - 0s 673us/step - loss: 0.2666 - accuracy:
0.8915
Epoch 328/1000
7/7 [==============================] - 0s 701us/step - loss: 0.2664 - accuracy:
0.8915
Epoch 329/1000
7/7 [==============================] - 0s 755us/step - loss: 0.2660 - accuracy:
0.8915
Epoch 330/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2658 - accuracy:
0.8915
Epoch 331/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2655 - accuracy:
0.8915
Epoch 332/1000
7/7 [==============================] - 0s 796us/step - loss: 0.2653 - accuracy:
0.8915
Epoch 333/1000
7/7 [==============================] - 0s 671us/step - loss: 0.2649 - accuracy:
0.8915
Epoch 334/1000
7/7 [==============================] - 0s 735us/step - loss: 0.2646 - accuracy:
0.8915
Epoch 335/1000
7/7 [==============================] - 0s 687us/step - loss: 0.2643 - accuracy:
0.8915
Epoch 336/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2641 - accuracy:
0.8915
Epoch 337/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2638 - accuracy:
0.8915
Epoch 338/1000
7/7 [==============================] - 0s 876us/step - loss: 0.2635 - accuracy:
```

```
0.8915
Epoch 339/1000
7/7 [==============================] - 0s 752us/step - loss: 0.2632 - accuracy:
0.8915
Epoch 340/1000
7/7 [==============================] - 0s 965us/step - loss: 0.2629 - accuracy:
0.8915
Epoch 341/1000
7/7 [==============================] - 0s 724us/step - loss: 0.2626 - accuracy:
0.8915
Epoch 342/1000
7/7 [==============================] - 0s 682us/step - loss: 0.2623 - accuracy:
0.8915
Epoch 343/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2621 - accuracy:
0.8915
Epoch 344/1000
7/7 [==============================] - 0s 638us/step - loss: 0.2618 - accuracy:
0.8915
Epoch 345/1000
7/7 [==============================] - 0s 814us/step - loss: 0.2614 - accuracy:
0.8915
Epoch 346/1000
7/7 [==============================] - 0s 745us/step - loss: 0.2612 - accuracy:
0.8915
Epoch 347/1000
7/7 [==============================] - 0s 666us/step - loss: 0.2609 - accuracy:
0.8962
Epoch 348/1000
7/7 [==============================] - 0s 882us/step - loss: 0.2606 - accuracy:
0.8962
Epoch 349/1000
7/7 [==============================] - 0s 769us/step - loss: 0.2604 - accuracy:
0.8962
Epoch 350/1000
7/7 [==============================] - 0s 906us/step - loss: 0.2601 - accuracy:
0.9009
Epoch 351/1000
7/7 [==============================] - 0s 934us/step - loss: 0.2598 - accuracy:
0.9009
Epoch 352/1000
7/7 [==============================] - 0s 973us/step - loss: 0.2596 - accuracy:
0.9009
Epoch 353/1000
7/7 [==============================] - 0s 709us/step - loss: 0.2593 - accuracy:
0.9009
Epoch 354/1000
7/7 [==============================] - 0s 643us/step - loss: 0.2590 - accuracy:
```

```
0.9009
Epoch 355/1000
7/7 [==============================] - 0s 768us/step - loss: 0.2587 - accuracy:
0.9009
Epoch 356/1000
7/7 [==============================] - 0s 716us/step - loss: 0.2584 - accuracy:
0.9009
Epoch 357/1000
7/7 [==============================] - 0s 702us/step - loss: 0.2582 - accuracy:
0.9009
Epoch 358/1000
7/7 [==============================] - 0s 730us/step - loss: 0.2579 - accuracy:
0.9057
Epoch 359/1000
7/7 [==============================] - 0s 710us/step - loss: 0.2576 - accuracy:
0.9057
Epoch 360/1000
7/7 [==============================] - 0s 759us/step - loss: 0.2573 - accuracy:
0.9057
Epoch 361/1000
7/7 [==============================] - 0s 708us/step - loss: 0.2571 - accuracy:
0.9057
Epoch 362/1000
7/7 [==============================] - 0s 915us/step - loss: 0.2568 - accuracy:
0.9057
Epoch 363/1000
7/7 [==============================] - 0s 754us/step - loss: 0.2566 - accuracy:
0.9057
Epoch 364/1000
7/7 [==============================] - 0s 712us/step - loss: 0.2563 - accuracy:
0.9057
Epoch 365/1000
7/7 [==============================] - 0s 739us/step - loss: 0.2560 - accuracy:
0.9057
Epoch 366/1000
7/7 [==============================] - 0s 666us/step - loss: 0.2558 - accuracy:
0.9057
Epoch 367/1000
7/7 [==============================] - 0s 631us/step - loss: 0.2555 - accuracy:
0.9057
Epoch 368/1000
7/7 [==============================] - 0s 846us/step - loss: 0.2552 - accuracy:
0.9009
Epoch 369/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2550 - accuracy:
0.9009
Epoch 370/1000
7/7 [==============================] - 0s 918us/step - loss: 0.2549 - accuracy:
```

```
0.9057
Epoch 371/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2544 - accuracy:
0.9057
Epoch 372/1000
7/7 [==============================] - 0s 718us/step - loss: 0.2542 - accuracy:
0.9057
Epoch 373/1000
7/7 [==============================] - 0s 818us/step - loss: 0.2541 - accuracy:
0.9057
Epoch 374/1000
7/7 [==============================] - 0s 869us/step - loss: 0.2537 - accuracy:
0.9057
Epoch 375/1000
7/7 [==============================] - 0s 775us/step - loss: 0.2535 - accuracy:
0.9009
Epoch 376/1000
7/7 [==============================] - 0s 790us/step - loss: 0.2532 - accuracy:
0.9009
Epoch 377/1000
7/7 [==============================] - 0s 680us/step - loss: 0.2528 - accuracy:
0.9009
Epoch 378/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2526 - accuracy:
0.9009
Epoch 379/1000
7/7 [==============================] - 0s 730us/step - loss: 0.2524 - accuracy:
0.9009
Epoch 380/1000
7/7 [==============================] - 0s 967us/step - loss: 0.2521 - accuracy:
0.9009
Epoch 381/1000
7/7 [==============================] - 0s 790us/step - loss: 0.2518 - accuracy:
0.9009
Epoch 382/1000
7/7 [==============================] - 0s 666us/step - loss: 0.2516 - accuracy:
0.9009
Epoch 383/1000
7/7 [==============================] - 0s 930us/step - loss: 0.2513 - accuracy:
0.9057
Epoch 384/1000
7/7 [==============================] - 0s 879us/step - loss: 0.2511 - accuracy:
0.9057
Epoch 385/1000
7/7 [==============================] - 0s 832us/step - loss: 0.2508 - accuracy:
0.9057
Epoch 386/1000
7/7 [==============================] - 0s 885us/step - loss: 0.2506 - accuracy:
```

```
0.9057
Epoch 387/1000
7/7 [==============================] - 0s 805us/step - loss: 0.2503 - accuracy:
0.9009
Epoch 388/1000
7/7 [==============================] - 0s 695us/step - loss: 0.2500 - accuracy:
0.9057
Epoch 389/1000
7/7 [==============================] - 0s 719us/step - loss: 0.2499 - accuracy:
0.9057
Epoch 390/1000
7/7 [==============================] - 0s 724us/step - loss: 0.2495 - accuracy:
0.9009
Epoch 391/1000
7/7 [==============================] - 0s 630us/step - loss: 0.2493 - accuracy:
0.9057
Epoch 392/1000
7/7 [==============================] - 0s 692us/step - loss: 0.2490 - accuracy:
0.9057
Epoch 393/1000
7/7 [==============================] - 0s 765us/step - loss: 0.2488 - accuracy:
0.9057
Epoch 394/1000
7/7 [==============================] - 0s 651us/step - loss: 0.2485 - accuracy:
0.9057
Epoch 395/1000
7/7 [==============================] - 0s 688us/step - loss: 0.2482 - accuracy:
0.9009
Epoch 396/1000
7/7 [==============================] - 0s 757us/step - loss: 0.2481 - accuracy:
0.9057
Epoch 397/1000
7/7 [==============================] - 0s 697us/step - loss: 0.2478 - accuracy:
0.9009
Epoch 398/1000
7/7 [==============================] - 0s 778us/step - loss: 0.2475 - accuracy:
0.9009
Epoch 399/1000
7/7 [==============================] - 0s 850us/step - loss: 0.2472 - accuracy:
0.9009
Epoch 400/1000
7/7 [==============================] - 0s 723us/step - loss: 0.2470 - accuracy:
0.9009
Epoch 401/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2467 - accuracy:
0.9009
Epoch 402/1000
7/7 [==============================] - 0s 938us/step - loss: 0.2465 - accuracy:
```

```
0.9009
Epoch 403/1000
7/7 [==============================] - 0s 813us/step - loss: 0.2463 - accuracy:
0.9009
Epoch 404/1000
7/7 [==============================] - 0s 969us/step - loss: 0.2460 - accuracy:
0.9009
Epoch 405/1000
7/7 [==============================] - 0s 734us/step - loss: 0.2458 - accuracy:
0.9057
Epoch 406/1000
7/7 [==============================] - 0s 639us/step - loss: 0.2454 - accuracy:
0.9057
Epoch 407/1000
7/7 [==============================] - 0s 734us/step - loss: 0.2452 - accuracy:
0.9057
Epoch 408/1000
7/7 [==============================] - 0s 853us/step - loss: 0.2449 - accuracy:
0.9057
Epoch 409/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2448 - accuracy:
0.9009
Epoch 410/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2445 - accuracy:
0.9009
Epoch 411/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2443 - accuracy:
0.9009
Epoch 412/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2440 - accuracy:
0.9009
Epoch 413/1000
7/7 [==============================] - 0s 797us/step - loss: 0.2438 - accuracy:
0.9009
Epoch 414/1000
7/7 [==============================] - 0s 708us/step - loss: 0.2435 - accuracy:
0.9009
Epoch 415/1000
7/7 [==============================] - 0s 743us/step - loss: 0.2433 - accuracy:
0.9057
Epoch 416/1000
7/7 [==============================] - 0s 813us/step - loss: 0.2430 - accuracy:
0.9009
Epoch 417/1000
7/7 [==============================] - 0s 701us/step - loss: 0.2428 - accuracy:
0.9057
Epoch 418/1000
7/7 [==============================] - 0s 751us/step - loss: 0.2425 - accuracy:
```

```
0.9057
Epoch 419/1000
7/7 [==============================] - 0s 827us/step - loss: 0.2423 - accuracy:
0.9057
Epoch 420/1000
7/7 [==============================] - 0s 710us/step - loss: 0.2421 - accuracy:
0.9057
Epoch 421/1000
7/7 [==============================] - 0s 741us/step - loss: 0.2418 - accuracy:
0.9009
Epoch 422/1000
7/7 [==============================] - 0s 649us/step - loss: 0.2415 - accuracy:
0.9057
Epoch 423/1000
7/7 [==============================] - 0s 686us/step - loss: 0.2414 - accuracy:
0.9009
Epoch 424/1000
7/7 [==============================] - 0s 785us/step - loss: 0.2411 - accuracy:
0.9057
Epoch 425/1000
7/7 [==============================] - 0s 674us/step - loss: 0.2408 - accuracy:
0.9057
Epoch 426/1000
7/7 [==============================] - 0s 718us/step - loss: 0.2406 - accuracy:
0.9057
Epoch 427/1000
7/7 [==============================] - 0s 785us/step - loss: 0.2404 - accuracy:
0.9057
Epoch 428/1000
7/7 [==============================] - 0s 635us/step - loss: 0.2401 - accuracy:
0.9057
Epoch 429/1000
7/7 [==============================] - 0s 688us/step - loss: 0.2398 - accuracy:
0.9057
Epoch 430/1000
7/7 [==============================] - 0s 713us/step - loss: 0.2396 - accuracy:
0.9057
Epoch 431/1000
7/7 [==============================] - 0s 741us/step - loss: 0.2394 - accuracy:
0.9057
Epoch 432/1000
7/7 [==============================] - 0s 681us/step - loss: 0.2391 - accuracy:
0.9057
Epoch 433/1000
7/7 [==============================] - 0s 645us/step - loss: 0.2389 - accuracy:
0.9057
Epoch 434/1000
7/7 [==============================] - 0s 728us/step - loss: 0.2387 - accuracy:
```

```
0.9104
Epoch 435/1000
7/7 [==============================] - 0s 733us/step - loss: 0.2384 - accuracy:
0.9104
Epoch 436/1000
7/7 [==============================] - 0s 624us/step - loss: 0.2383 - accuracy:
0.9057
Epoch 437/1000
7/7 [==============================] - 0s 696us/step - loss: 0.2379 - accuracy:
0.9057
Epoch 438/1000
7/7 [==============================] - 0s 725us/step - loss: 0.2376 - accuracy:
0.9104
Epoch 439/1000
7/7 [==============================] - 0s 734us/step - loss: 0.2374 - accuracy:
0.9104
Epoch 440/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2372 - accuracy:
0.9104
Epoch 441/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2370 - accuracy:
0.9104
Epoch 442/1000
7/7 [==============================] - 0s 889us/step - loss: 0.2367 - accuracy:
0.9104
Epoch 443/1000
7/7 [==============================] - 0s 858us/step - loss: 0.2364 - accuracy:
0.9104
Epoch 444/1000
7/7 [==============================] - 0s 824us/step - loss: 0.2362 - accuracy:
0.9104
Epoch 445/1000
7/7 [==============================] - 0s 747us/step - loss: 0.2361 - accuracy:
0.9104
Epoch 446/1000
7/7 [==============================] - 0s 921us/step - loss: 0.2358 - accuracy:
0.9104
Epoch 447/1000
7/7 [==============================] - 0s 844us/step - loss: 0.2355 - accuracy:
0.9104
Epoch 448/1000
7/7 [==============================] - 0s 846us/step - loss: 0.2354 - accuracy:
0.9104
Epoch 449/1000
7/7 [==============================] - 0s 821us/step - loss: 0.2350 - accuracy:
0.9104
Epoch 450/1000
7/7 [==============================] - 0s 725us/step - loss: 0.2348 - accuracy:
```

```
0.9104
Epoch 451/1000
7/7 [==============================] - 0s 665us/step - loss: 0.2346 - accuracy:
0.9104
Epoch 452/1000
7/7 [==============================] - 0s 797us/step - loss: 0.2343 - accuracy:
0.9104
Epoch 453/1000
7/7 [==============================] - 0s 751us/step - loss: 0.2341 - accuracy:
0.9104
Epoch 454/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2338 - accuracy:
0.9104
Epoch 455/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2337 - accuracy:
0.9104
Epoch 456/1000
7/7 [==============================] - 0s 879us/step - loss: 0.2335 - accuracy:
0.9104
Epoch 457/1000
7/7 [==============================] - 0s 676us/step - loss: 0.2332 - accuracy:
0.9104
Epoch 458/1000
7/7 [==============================] - 0s 855us/step - loss: 0.2329 - accuracy:
0.9151
Epoch 459/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2327 - accuracy:
0.9104
Epoch 460/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2325 - accuracy:
0.9104
Epoch 461/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2322 - accuracy:
0.9151
Epoch 462/1000
7/7 [==============================] - 0s 814us/step - loss: 0.2320 - accuracy:
0.9104
Epoch 463/1000
7/7 [==============================] - 0s 750us/step - loss: 0.2317 - accuracy:
0.9151
Epoch 464/1000
7/7 [==============================] - 0s 805us/step - loss: 0.2315 - accuracy:
0.9151
Epoch 465/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2313 - accuracy:
0.9151
Epoch 466/1000
7/7 [==============================] - 0s 693us/step - loss: 0.2310 - accuracy:
```

```
0.9151
Epoch 467/1000
7/7 [==============================] - 0s 737us/step - loss: 0.2307 - accuracy:
0.9151
Epoch 468/1000
7/7 [==============================] - 0s 836us/step - loss: 0.2306 - accuracy:
0.9151
Epoch 469/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2303 - accuracy:
0.9198
Epoch 470/1000
7/7 [==============================] - 0s 643us/step - loss: 0.2302 - accuracy:
0.9151
Epoch 471/1000
7/7 [==============================] - 0s 957us/step - loss: 0.2299 - accuracy:
0.9151
Epoch 472/1000
7/7 [==============================] - 0s 821us/step - loss: 0.2296 - accuracy:
0.9198
Epoch 473/1000
7/7 [==============================] - 0s 893us/step - loss: 0.2294 - accuracy:
0.9198
Epoch 474/1000
7/7 [==============================] - 0s 711us/step - loss: 0.2291 - accuracy:
0.9198
Epoch 475/1000
7/7 [==============================] - 0s 759us/step - loss: 0.2288 - accuracy:
0.9198
Epoch 476/1000
7/7 [==============================] - 0s 864us/step - loss: 0.2286 - accuracy:
0.9198
Epoch 477/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2284 - accuracy:
0.9198
Epoch 478/1000
7/7 [==============================] - 0s 789us/step - loss: 0.2282 - accuracy:
0.9198
Epoch 479/1000
7/7 [==============================] - 0s 804us/step - loss: 0.2280 - accuracy:
0.9198
Epoch 480/1000
7/7 [==============================] - 0s 737us/step - loss: 0.2276 - accuracy:
0.9198
Epoch 481/1000
7/7 [==============================] - 0s 899us/step - loss: 0.2274 - accuracy:
0.9198
Epoch 482/1000
7/7 [==============================] - 0s 979us/step - loss: 0.2273 - accuracy:
```

```
0.9198
Epoch 483/1000
7/7 [==============================] - 0s 691us/step - loss: 0.2270 - accuracy:
0.9198
Epoch 484/1000
7/7 [==============================] - 0s 794us/step - loss: 0.2267 - accuracy:
0.9198
Epoch 485/1000
7/7 [==============================] - 0s 655us/step - loss: 0.2265 - accuracy:
0.9198
Epoch 486/1000
7/7 [==============================] - 0s 712us/step - loss: 0.2263 - accuracy:
0.9198
Epoch 487/1000
7/7 [==============================] - 0s 727us/step - loss: 0.2260 - accuracy:
0.9198
Epoch 488/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2257 - accuracy:
0.9245
Epoch 489/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2256 - accuracy:
0.9245
Epoch 490/1000
7/7 [==============================] - 0s 838us/step - loss: 0.2253 - accuracy:
0.9245
Epoch 491/1000
7/7 [==============================] - 0s 924us/step - loss: 0.2252 - accuracy:
0.9245
Epoch 492/1000
7/7 [==============================] - 0s 698us/step - loss: 0.2248 - accuracy:
0.9245
Epoch 493/1000
7/7 [==============================] - 0s 693us/step - loss: 0.2246 - accuracy:
0.9245
Epoch 494/1000
7/7 [==============================] - 0s 756us/step - loss: 0.2244 - accuracy:
0.9245
Epoch 495/1000
7/7 [==============================] - 0s 661us/step - loss: 0.2241 - accuracy:
0.9245
Epoch 496/1000
7/7 [==============================] - 0s 693us/step - loss: 0.2239 - accuracy:
0.9245
Epoch 497/1000
7/7 [==============================] - 0s 665us/step - loss: 0.2236 - accuracy:
0.9245
Epoch 498/1000
7/7 [==============================] - 0s 650us/step - loss: 0.2235 - accuracy:
```

```
0.9245
Epoch 499/1000
7/7 [==============================] - 0s 753us/step - loss: 0.2232 - accuracy:
0.9245
Epoch 500/1000
7/7 [==============================] - 0s 654us/step - loss: 0.2230 - accuracy:
0.9245
Epoch 501/1000
7/7 [==============================] - 0s 638us/step - loss: 0.2227 - accuracy:
0.9245
Epoch 502/1000
7/7 [==============================] - 0s 702us/step - loss: 0.2224 - accuracy:
0.9245
Epoch 503/1000
7/7 [==============================] - 0s 656us/step - loss: 0.2222 - accuracy:
0.9292
Epoch 504/1000
7/7 [==============================] - 0s 663us/step - loss: 0.2220 - accuracy:
0.9340
Epoch 505/1000
7/7 [==============================] - 0s 847us/step - loss: 0.2218 - accuracy:
0.9340
Epoch 506/1000
7/7 [==============================] - 0s 793us/step - loss: 0.2216 - accuracy:
0.9340
Epoch 507/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2213 - accuracy:
0.9340
Epoch 508/1000
7/7 [==============================] - 0s 937us/step - loss: 0.2211 - accuracy:
0.9340
Epoch 509/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2208 - accuracy:
0.9340
Epoch 510/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2206 - accuracy:
0.9340
Epoch 511/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2204 - accuracy:
0.9340
Epoch 512/1000
7/7 [==============================] - 0s 858us/step - loss: 0.2201 - accuracy:
0.9340
Epoch 513/1000
7/7 [==============================] - 0s 676us/step - loss: 0.2200 - accuracy:
0.9340
Epoch 514/1000
7/7 [==============================] - 0s 899us/step - loss: 0.2198 - accuracy:
```

```
0.9340
Epoch 515/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2195 - accuracy:
0.9340
Epoch 516/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2192 - accuracy:
0.9340
Epoch 517/1000
7/7 [==============================] - 0s 832us/step - loss: 0.2190 - accuracy:
0.9340
Epoch 518/1000
7/7 [==============================] - 0s 730us/step - loss: 0.2188 - accuracy:
0.9340
Epoch 519/1000
7/7 [==============================] - 0s 676us/step - loss: 0.2186 - accuracy:
0.9340
Epoch 520/1000
7/7 [==============================] - 0s 695us/step - loss: 0.2183 - accuracy:
0.9340
Epoch 521/1000
7/7 [==============================] - 0s 631us/step - loss: 0.2181 - accuracy:
0.9340
Epoch 522/1000
7/7 [==============================] - 0s 891us/step - loss: 0.2180 - accuracy:
0.9340
Epoch 523/1000
7/7 [==============================] - 0s 757us/step - loss: 0.2176 - accuracy:
0.9340
Epoch 524/1000
7/7 [==============================] - 0s 702us/step - loss: 0.2174 - accuracy:
0.9340
Epoch 525/1000
7/7 [==============================] - 0s 616us/step - loss: 0.2172 - accuracy:
0.9340
Epoch 526/1000
7/7 [==============================] - 0s 738us/step - loss: 0.2169 - accuracy:
0.9340
Epoch 527/1000
7/7 [==============================] - 0s 662us/step - loss: 0.2167 - accuracy:
0.9340
Epoch 528/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2165 - accuracy:
0.9340
Epoch 529/1000
7/7 [==============================] - 0s 745us/step - loss: 0.2163 - accuracy:
0.9340
Epoch 530/1000
7/7 [==============================] - 0s 624us/step - loss: 0.2160 - accuracy:
```

```
0.9340
Epoch 531/1000
7/7 [==============================] - 0s 827us/step - loss: 0.2158 - accuracy:
0.9340
Epoch 532/1000
7/7 [==============================] - 0s 653us/step - loss: 0.2156 - accuracy:
0.9340
Epoch 533/1000
7/7 [==============================] - 0s 650us/step - loss: 0.2153 - accuracy:
0.9340
Epoch 534/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2151 - accuracy:
0.9340
Epoch 535/1000
7/7 [==============================] - 0s 600us/step - loss: 0.2148 - accuracy:
0.9340
Epoch 536/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2146 - accuracy:
0.9340
Epoch 537/1000
7/7 [==============================] - 0s 884us/step - loss: 0.2144 - accuracy:
0.9340
Epoch 538/1000
7/7 [==============================] - 0s 975us/step - loss: 0.2142 - accuracy:
0.9340
Epoch 539/1000
7/7 [==============================] - 0s 648us/step - loss: 0.2140 - accuracy:
0.9340
Epoch 540/1000
7/7 [==============================] - 0s 743us/step - loss: 0.2138 - accuracy:
0.9340
Epoch 541/1000
7/7 [==============================] - 0s 717us/step - loss: 0.2136 - accuracy:
0.9340
Epoch 542/1000
7/7 [==============================] - 0s 714us/step - loss: 0.2133 - accuracy:
0.9340
Epoch 543/1000
7/7 [==============================] - 0s 852us/step - loss: 0.2130 - accuracy:
0.9340
Epoch 544/1000
7/7 [==============================] - 0s 742us/step - loss: 0.2128 - accuracy:
0.9340
Epoch 545/1000
7/7 [==============================] - 0s 768us/step - loss: 0.2126 - accuracy:
0.9340
Epoch 546/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2124 - accuracy:
```

```
0.9340
Epoch 547/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2122 - accuracy:
0.9340
Epoch 548/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2119 - accuracy:
0.9340
Epoch 549/1000
7/7 [==============================] - 0s 918us/step - loss: 0.2117 - accuracy:
0.9340
Epoch 550/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2115 - accuracy:
0.9340
Epoch 551/1000
7/7 [==============================] - 0s 758us/step - loss: 0.2112 - accuracy:
0.9340
Epoch 552/1000
7/7 [==============================] - 0s 702us/step - loss: 0.2110 - accuracy:
0.9340
Epoch 553/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2108 - accuracy:
0.9340
Epoch 554/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2106 - accuracy:
0.9340
Epoch 555/1000
7/7 [==============================] - 0s 1000us/step - loss: 0.2104 - accuracy:
0.9387
Epoch 556/1000
7/7 [==============================] - 0s 687us/step - loss: 0.2101 - accuracy:
0.9387
Epoch 557/1000
7/7 [==============================] - 0s 940us/step - loss: 0.2100 - accuracy:
0.9340
Epoch 558/1000
7/7 [==============================] - 0s 721us/step - loss: 0.2097 - accuracy:
0.9387
Epoch 559/1000
7/7 [==============================] - 0s 637us/step - loss: 0.2095 - accuracy:
0.9387
Epoch 560/1000
7/7 [==============================] - 0s 704us/step - loss: 0.2093 - accuracy:
0.9387
Epoch 561/1000
7/7 [==============================] - 0s 680us/step - loss: 0.2090 - accuracy:
0.9387
Epoch 562/1000
7/7 [==============================] - 0s 952us/step - loss: 0.2088 - accuracy:
```

```
0.9387
Epoch 563/1000
7/7 [==============================] - 0s 671us/step - loss: 0.2086 - accuracy:
0.9387
Epoch 564/1000
7/7 [==============================] - 0s 690us/step - loss: 0.2084 - accuracy:
0.9387
Epoch 565/1000
7/7 [==============================] - 0s 750us/step - loss: 0.2082 - accuracy:
0.9387
Epoch 566/1000
7/7 [==============================] - 0s 959us/step - loss: 0.2079 - accuracy:
0.9387
Epoch 567/1000
7/7 [==============================] - 0s 736us/step - loss: 0.2078 - accuracy:
0.9387
Epoch 568/1000
7/7 [==============================] - 0s 724us/step - loss: 0.2075 - accuracy:
0.9387
Epoch 569/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2073 - accuracy:
0.9387
Epoch 570/1000
7/7 [==============================] - 0s 994us/step - loss: 0.2070 - accuracy:
0.9387
Epoch 571/1000
7/7 [==============================] - 0s 659us/step - loss: 0.2069 - accuracy:
0.9387
Epoch 572/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2067 - accuracy:
0.9387
Epoch 573/1000
7/7 [==============================] - 0s 722us/step - loss: 0.2065 - accuracy:
0.9387
Epoch 574/1000
7/7 [==============================] - 0s 743us/step - loss: 0.2062 - accuracy:
0.9387
Epoch 575/1000
7/7 [==============================] - 0s 821us/step - loss: 0.2060 - accuracy:
0.9387
Epoch 576/1000
7/7 [==============================] - 0s 721us/step - loss: 0.2057 - accuracy:
0.9387
Epoch 577/1000
7/7 [==============================] - 0s 858us/step - loss: 0.2055 - accuracy:
0.9387
Epoch 578/1000
7/7 [==============================] - 0s 659us/step - loss: 0.2053 - accuracy:
```

```
0.9387
Epoch 579/1000
7/7 [==============================] - 0s 653us/step - loss: 0.2051 - accuracy:
0.9387
Epoch 580/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2049 - accuracy:
0.9387
Epoch 581/1000
7/7 [==============================] - 0s 715us/step - loss: 0.2047 - accuracy:
0.9387
Epoch 582/1000
7/7 [==============================] - 0s 746us/step - loss: 0.2045 - accuracy:
0.9387
Epoch 583/1000
7/7 [==============================] - 0s 950us/step - loss: 0.2043 - accuracy:
0.9387
Epoch 584/1000
7/7 [==============================] - 0s 846us/step - loss: 0.2040 - accuracy:
0.9387
Epoch 585/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2038 - accuracy:
0.9387
Epoch 586/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.2035 - accuracy:
0.9387
Epoch 587/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2034 - accuracy:
0.9387
Epoch 588/1000
7/7 [==============================] - 0s 701us/step - loss: 0.2032 - accuracy:
0.9387
Epoch 589/1000
7/7 [==============================] - 0s 708us/step - loss: 0.2030 - accuracy:
0.9387
Epoch 590/1000
7/7 [==============================] - 0s 821us/step - loss: 0.2027 - accuracy:
0.9387
Epoch 591/1000
7/7 [==============================] - 0s 711us/step - loss: 0.2025 - accuracy:
0.9387
Epoch 592/1000
7/7 [==============================] - 0s 766us/step - loss: 0.2023 - accuracy:
0.9387
Epoch 593/1000
7/7 [==============================] - 0s 859us/step - loss: 0.2020 - accuracy:
0.9387
Epoch 594/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2019 - accuracy:
```

```
0.9387
Epoch 595/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.2017 - accuracy:
0.9387
Epoch 596/1000
7/7 [==============================] - 0s 971us/step - loss: 0.2015 - accuracy:
0.9387
Epoch 597/1000
7/7 [==============================] - 0s 742us/step - loss: 0.2013 - accuracy:
0.9387
Epoch 598/1000
7/7 [==============================] - 0s 675us/step - loss: 0.2011 - accuracy:
0.9387
Epoch 599/1000
7/7 [==============================] - 0s 728us/step - loss: 0.2008 - accuracy:
0.9387
Epoch 600/1000
7/7 [==============================] - 0s 750us/step - loss: 0.2006 - accuracy:
0.9387
Epoch 601/1000
7/7 [==============================] - 0s 741us/step - loss: 0.2004 - accuracy:
0.9387
Epoch 602/1000
7/7 [==============================] - 0s 670us/step - loss: 0.2001 - accuracy:
0.9387
Epoch 603/1000
7/7 [==============================] - 0s 745us/step - loss: 0.2000 - accuracy:
0.9387
Epoch 604/1000
7/7 [==============================] - 0s 753us/step - loss: 0.1997 - accuracy:
0.9387
Epoch 605/1000
7/7 [==============================] - 0s 867us/step - loss: 0.1996 - accuracy:
0.9387
Epoch 606/1000
7/7 [==============================] - 0s 713us/step - loss: 0.1993 - accuracy:
0.9387
Epoch 607/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1991 - accuracy:
0.9387
Epoch 608/1000
7/7 [==============================] - 0s 656us/step - loss: 0.1989 - accuracy:
0.9387
Epoch 609/1000
7/7 [==============================] - 0s 970us/step - loss: 0.1987 - accuracy:
0.9387
Epoch 610/1000
7/7 [==============================] - 0s 679us/step - loss: 0.1985 - accuracy:
```

```
0.9387
Epoch 611/1000
7/7 [==============================] - 0s 743us/step - loss: 0.1982 - accuracy:
0.9387
Epoch 612/1000
7/7 [==============================] - 0s 696us/step - loss: 0.1982 - accuracy:
0.9387
Epoch 613/1000
7/7 [==============================] - 0s 736us/step - loss: 0.1979 - accuracy:
0.9387
Epoch 614/1000
7/7 [==============================] - 0s 717us/step - loss: 0.1977 - accuracy:
0.9387
Epoch 615/1000
7/7 [==============================] - 0s 720us/step - loss: 0.1975 - accuracy:
0.9387
Epoch 616/1000
7/7 [==============================] - 0s 656us/step - loss: 0.1973 - accuracy:
0.9387
Epoch 617/1000
7/7 [==============================] - 0s 750us/step - loss: 0.1971 - accuracy:
0.9387
Epoch 618/1000
7/7 [==============================] - 0s 701us/step - loss: 0.1967 - accuracy:
0.9387
Epoch 619/1000
7/7 [==============================] - 0s 652us/step - loss: 0.1967 - accuracy:
0.9387
Epoch 620/1000
7/7 [==============================] - 0s 718us/step - loss: 0.1964 - accuracy:
0.9387
Epoch 621/1000
7/7 [==============================] - 0s 701us/step - loss: 0.1962 - accuracy:
0.9387
Epoch 622/1000
7/7 [==============================] - 0s 696us/step - loss: 0.1959 - accuracy:
0.9387
Epoch 623/1000
7/7 [==============================] - 0s 834us/step - loss: 0.1958 - accuracy:
0.9387
Epoch 624/1000
7/7 [==============================] - 0s 822us/step - loss: 0.1956 - accuracy:
0.9387
Epoch 625/1000
7/7 [==============================] - 0s 795us/step - loss: 0.1954 - accuracy:
0.9387
Epoch 626/1000
7/7 [==============================] - 0s 856us/step - loss: 0.1953 - accuracy:
```

```
0.9387
Epoch 627/1000
7/7 [==============================] - 0s 964us/step - loss: 0.1949 - accuracy:
0.9387
Epoch 628/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1947 - accuracy:
0.9387
Epoch 629/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1945 - accuracy:
0.9387
Epoch 630/1000
7/7 [==============================] - 0s 793us/step - loss: 0.1942 - accuracy:
0.9387
Epoch 631/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1941 - accuracy:
0.9387
Epoch 632/1000
7/7 [==============================] - 0s 769us/step - loss: 0.1939 - accuracy:
0.9387
Epoch 633/1000
7/7 [==============================] - 0s 751us/step - loss: 0.1936 - accuracy:
0.9387
Epoch 634/1000
7/7 [==============================] - 0s 685us/step - loss: 0.1935 - accuracy:
0.9434
Epoch 635/1000
7/7 [==============================] - 0s 848us/step - loss: 0.1932 - accuracy:
0.9434
Epoch 636/1000
7/7 [==============================] - 0s 966us/step - loss: 0.1930 - accuracy:
0.9434
Epoch 637/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1928 - accuracy:
0.9434
Epoch 638/1000
7/7 [==============================] - 0s 696us/step - loss: 0.1926 - accuracy:
0.9434
Epoch 639/1000
7/7 [==============================] - 0s 728us/step - loss: 0.1925 - accuracy:
0.9434
Epoch 640/1000
7/7 [==============================] - 0s 689us/step - loss: 0.1922 - accuracy:
0.9434
Epoch 641/1000
7/7 [==============================] - 0s 683us/step - loss: 0.1920 - accuracy:
0.9434
Epoch 642/1000
7/7 [==============================] - 0s 715us/step - loss: 0.1918 - accuracy:
```

```
0.9434
Epoch 643/1000
7/7 [==============================] - 0s 680us/step - loss: 0.1915 - accuracy:
0.9434
Epoch 644/1000
7/7 [==============================] - 0s 667us/step - loss: 0.1914 - accuracy:
0.9434
Epoch 645/1000
7/7 [==============================] - 0s 696us/step - loss: 0.1911 - accuracy:
0.9434
Epoch 646/1000
7/7 [==============================] - 0s 799us/step - loss: 0.1909 - accuracy:
0.9434
Epoch 647/1000
7/7 [==============================] - 0s 739us/step - loss: 0.1907 - accuracy:
0.9434
Epoch 648/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1905 - accuracy:
0.9434
Epoch 649/1000
7/7 [==============================] - 0s 650us/step - loss: 0.1903 - accuracy:
0.9434
Epoch 650/1000
7/7 [==============================] - 0s 846us/step - loss: 0.1902 - accuracy:
0.9481
Epoch 651/1000
7/7 [==============================] - 0s 799us/step - loss: 0.1899 - accuracy:
0.9481
Epoch 652/1000
7/7 [==============================] - 0s 716us/step - loss: 0.1898 - accuracy:
0.9434
Epoch 653/1000
7/7 [==============================] - 0s 1000us/step - loss: 0.1895 - accuracy:
0.9481
Epoch 654/1000
7/7 [==============================] - 0s 645us/step - loss: 0.1893 - accuracy:
0.9481
Epoch 655/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1891 - accuracy:
0.9481
Epoch 656/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1889 - accuracy:
0.9481
Epoch 657/1000
7/7 [==============================] - 0s 838us/step - loss: 0.1887 - accuracy:
0.9481
Epoch 658/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1885 - accuracy:
```

```
0.9481
Epoch 659/1000
7/7 [==============================] - 0s 682us/step - loss: 0.1883 - accuracy:
0.9481
Epoch 660/1000
7/7 [==============================] - 0s 804us/step - loss: 0.1880 - accuracy:
0.9481
Epoch 661/1000
7/7 [==============================] - 0s 842us/step - loss: 0.1878 - accuracy:
0.9481
Epoch 662/1000
7/7 [==============================] - 0s 676us/step - loss: 0.1876 - accuracy:
0.9481
Epoch 663/1000
7/7 [==============================] - 0s 777us/step - loss: 0.1874 - accuracy:
0.9481
Epoch 664/1000
7/7 [==============================] - 0s 756us/step - loss: 0.1873 - accuracy:
0.9481
Epoch 665/1000
7/7 [==============================] - 0s 705us/step - loss: 0.1870 - accuracy:
0.9481
Epoch 666/1000
7/7 [==============================] - 0s 709us/step - loss: 0.1868 - accuracy:
0.9481
Epoch 667/1000
7/7 [==============================] - 0s 736us/step - loss: 0.1866 - accuracy:
0.9481
Epoch 668/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1864 - accuracy:
0.9481
Epoch 669/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1862 - accuracy:
0.9481
Epoch 670/1000
7/7 [==============================] - 0s 773us/step - loss: 0.1860 - accuracy:
0.9481
Epoch 671/1000
7/7 [==============================] - 0s 848us/step - loss: 0.1857 - accuracy:
0.9481
Epoch 672/1000
7/7 [==============================] - 0s 666us/step - loss: 0.1856 - accuracy:
0.9481
Epoch 673/1000
7/7 [==============================] - 0s 748us/step - loss: 0.1853 - accuracy:
0.9481
Epoch 674/1000
7/7 [==============================] - 0s 710us/step - loss: 0.1852 - accuracy:
```

0.9481
Epoch 675/1000
7/7 [==============================] - 0s 742us/step - loss: 0.1849 - accuracy:
0.9481
Epoch 676/1000
7/7 [==============================] - 0s 825us/step - loss: 0.1848 - accuracy:
0.9481
Epoch 677/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1846 - accuracy:
0.9481
Epoch 678/1000
7/7 [==============================] - 0s 829us/step - loss: 0.1844 - accuracy:
0.9481
Epoch 679/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1841 - accuracy:
0.9481
Epoch 680/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1840 - accuracy:
0.9481
Epoch 681/1000
7/7 [==============================] - 0s 975us/step - loss: 0.1838 - accuracy:
0.9481
Epoch 682/1000
7/7 [==============================] - 0s 753us/step - loss: 0.1835 - accuracy:
0.9481
Epoch 683/1000
7/7 [==============================] - 0s 705us/step - loss: 0.1834 - accuracy:
0.9481
Epoch 684/1000
7/7 [==============================] - 0s 786us/step - loss: 0.1832 - accuracy:
0.9481
Epoch 685/1000
7/7 [==============================] - 0s 666us/step - loss: 0.1829 - accuracy:
0.9481
Epoch 686/1000
7/7 [==============================] - 0s 869us/step - loss: 0.1828 - accuracy:
0.9481
Epoch 687/1000
7/7 [==============================] - 0s 672us/step - loss: 0.1826 - accuracy:
0.9481
Epoch 688/1000
7/7 [==============================] - 0s 683us/step - loss: 0.1823 - accuracy:
0.9481
Epoch 689/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1822 - accuracy:
0.9528
Epoch 690/1000
7/7 [==============================] - 0s 723us/step - loss: 0.1820 - accuracy:

```
0.9528
Epoch 691/1000
7/7 [==============================] - 0s 741us/step - loss: 0.1817 - accuracy:
0.9528
Epoch 692/1000
7/7 [==============================] - 0s 742us/step - loss: 0.1816 - accuracy:
0.9528
Epoch 693/1000
7/7 [==============================] - 0s 635us/step - loss: 0.1814 - accuracy:
0.9528
Epoch 694/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1812 - accuracy:
0.9528
Epoch 695/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1810 - accuracy:
0.9528
Epoch 696/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1808 - accuracy:
0.9528
Epoch 697/1000
7/7 [==============================] - 0s 734us/step - loss: 0.1806 - accuracy:
0.9528
Epoch 698/1000
7/7 [==============================] - 0s 800us/step - loss: 0.1804 - accuracy:
0.9528
Epoch 699/1000
7/7 [==============================] - 0s 701us/step - loss: 0.1802 - accuracy:
0.9481
Epoch 700/1000
7/7 [==============================] - 0s 722us/step - loss: 0.1799 - accuracy:
0.9528
Epoch 701/1000
7/7 [==============================] - 0s 809us/step - loss: 0.1797 - accuracy:
0.9528
Epoch 702/1000
7/7 [==============================] - 0s 701us/step - loss: 0.1796 - accuracy:
0.9528
Epoch 703/1000
7/7 [==============================] - 0s 713us/step - loss: 0.1795 - accuracy:
0.9528
Epoch 704/1000
7/7 [==============================] - 0s 696us/step - loss: 0.1792 - accuracy:
0.9528
Epoch 705/1000
7/7 [==============================] - 0s 680us/step - loss: 0.1790 - accuracy:
0.9528
Epoch 706/1000
7/7 [==============================] - 0s 711us/step - loss: 0.1788 - accuracy:
```

```
0.9528
Epoch 707/1000
7/7 [==============================] - 0s 682us/step - loss: 0.1786 - accuracy:
0.9528
Epoch 708/1000
7/7 [==============================] - 0s 751us/step - loss: 0.1785 - accuracy:
0.9528
Epoch 709/1000
7/7 [==============================] - 0s 680us/step - loss: 0.1782 - accuracy:
0.9528
Epoch 710/1000
7/7 [==============================] - 0s 629us/step - loss: 0.1780 - accuracy:
0.9528
Epoch 711/1000
7/7 [==============================] - 0s 774us/step - loss: 0.1779 - accuracy:
0.9528
Epoch 712/1000
7/7 [==============================] - 0s 835us/step - loss: 0.1778 - accuracy:
0.9528
Epoch 713/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1775 - accuracy:
0.9528
Epoch 714/1000
7/7 [==============================] - 0s 920us/step - loss: 0.1773 - accuracy:
0.9528
Epoch 715/1000
7/7 [==============================] - 0s 654us/step - loss: 0.1771 - accuracy:
0.9528
Epoch 716/1000
7/7 [==============================] - 0s 792us/step - loss: 0.1769 - accuracy:
0.9528
Epoch 717/1000
7/7 [==============================] - 0s 662us/step - loss: 0.1767 - accuracy:
0.9528
Epoch 718/1000
7/7 [==============================] - 0s 745us/step - loss: 0.1765 - accuracy:
0.9528
Epoch 719/1000
7/7 [==============================] - 0s 749us/step - loss: 0.1763 - accuracy:
0.9528
Epoch 720/1000
7/7 [==============================] - 0s 821us/step - loss: 0.1761 - accuracy:
0.9528
Epoch 721/1000
7/7 [==============================] - 0s 846us/step - loss: 0.1759 - accuracy:
0.9528
Epoch 722/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1757 - accuracy:
```

```
0.9528
Epoch 723/1000
7/7 [==============================] - 0s 766us/step - loss: 0.1754 - accuracy:
0.9528
Epoch 724/1000
7/7 [==============================] - 0s 666us/step - loss: 0.1753 - accuracy:
0.9528
Epoch 725/1000
7/7 [==============================] - 0s 640us/step - loss: 0.1751 - accuracy:
0.9528
Epoch 726/1000
7/7 [==============================] - 0s 691us/step - loss: 0.1749 - accuracy:
0.9528
Epoch 727/1000
7/7 [==============================] - 0s 689us/step - loss: 0.1748 - accuracy:
0.9528
Epoch 728/1000
7/7 [==============================] - 0s 704us/step - loss: 0.1745 - accuracy:
0.9528
Epoch 729/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1742 - accuracy:
0.9528
Epoch 730/1000
7/7 [==============================] - 0s 929us/step - loss: 0.1740 - accuracy:
0.9528
Epoch 731/1000
7/7 [==============================] - 0s 818us/step - loss: 0.1739 - accuracy:
0.9528
Epoch 732/1000
7/7 [==============================] - 0s 757us/step - loss: 0.1736 - accuracy:
0.9528
Epoch 733/1000
7/7 [==============================] - 0s 840us/step - loss: 0.1734 - accuracy:
0.9528
Epoch 734/1000
7/7 [==============================] - 0s 669us/step - loss: 0.1733 - accuracy:
0.9528
Epoch 735/1000
7/7 [==============================] - 0s 844us/step - loss: 0.1730 - accuracy:
0.9528
Epoch 736/1000
7/7 [==============================] - 0s 672us/step - loss: 0.1728 - accuracy:
0.9528
Epoch 737/1000
7/7 [==============================] - 0s 669us/step - loss: 0.1726 - accuracy:
0.9528
Epoch 738/1000
7/7 [==============================] - 0s 836us/step - loss: 0.1724 - accuracy:
```

```
0.9528
Epoch 739/1000
7/7 [==============================] - 0s 702us/step - loss: 0.1721 - accuracy:
0.9528
Epoch 740/1000
7/7 [==============================] - 0s 710us/step - loss: 0.1719 - accuracy:
0.9528
Epoch 741/1000
7/7 [==============================] - 0s 754us/step - loss: 0.1718 - accuracy:
0.9528
Epoch 742/1000
7/7 [==============================] - 0s 684us/step - loss: 0.1717 - accuracy:
0.9528
Epoch 743/1000
7/7 [==============================] - 0s 778us/step - loss: 0.1714 - accuracy:
0.9528
Epoch 744/1000
7/7 [==============================] - 0s 691us/step - loss: 0.1711 - accuracy:
0.9528
Epoch 745/1000
7/7 [==============================] - 0s 766us/step - loss: 0.1709 - accuracy:
0.9528
Epoch 746/1000
7/7 [==============================] - 0s 760us/step - loss: 0.1708 - accuracy:
0.9528
Epoch 747/1000
7/7 [==============================] - 0s 659us/step - loss: 0.1705 - accuracy:
0.9528
Epoch 748/1000
7/7 [==============================] - 0s 731us/step - loss: 0.1703 - accuracy:
0.9528
Epoch 749/1000
7/7 [==============================] - 0s 699us/step - loss: 0.1702 - accuracy:
0.9528
Epoch 750/1000
7/7 [==============================] - 0s 685us/step - loss: 0.1699 - accuracy:
0.9528
Epoch 751/1000
7/7 [==============================] - 0s 674us/step - loss: 0.1697 - accuracy:
0.9528
Epoch 752/1000
7/7 [==============================] - 0s 874us/step - loss: 0.1695 - accuracy:
0.9528
Epoch 753/1000
7/7 [==============================] - 0s 713us/step - loss: 0.1693 - accuracy:
0.9528
Epoch 754/1000
7/7 [==============================] - 0s 743us/step - loss: 0.1692 - accuracy:
```

```
0.9528
Epoch 755/1000
7/7 [==============================] - 0s 735us/step - loss: 0.1689 - accuracy:
0.9528
Epoch 756/1000
7/7 [==============================] - 0s 777us/step - loss: 0.1688 - accuracy:
0.9528
Epoch 757/1000
7/7 [==============================] - 0s 677us/step - loss: 0.1685 - accuracy:
0.9528
Epoch 758/1000
7/7 [==============================] - 0s 688us/step - loss: 0.1683 - accuracy:
0.9528
Epoch 759/1000
7/7 [==============================] - 0s 763us/step - loss: 0.1681 - accuracy:
0.9528
Epoch 760/1000
7/7 [==============================] - 0s 984us/step - loss: 0.1679 - accuracy:
0.9528
Epoch 761/1000
7/7 [==============================] - 0s 855us/step - loss: 0.1677 - accuracy:
0.9528
Epoch 762/1000
7/7 [==============================] - 0s 713us/step - loss: 0.1675 - accuracy:
0.9528
Epoch 763/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1673 - accuracy:
0.9528
Epoch 764/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1671 - accuracy:
0.9528
Epoch 765/1000
7/7 [==============================] - 0s 928us/step - loss: 0.1669 - accuracy:
0.9528
Epoch 766/1000
7/7 [==============================] - 0s 772us/step - loss: 0.1668 - accuracy:
0.9528
Epoch 767/1000
7/7 [==============================] - 0s 800us/step - loss: 0.1665 - accuracy:
0.9528
Epoch 768/1000
7/7 [==============================] - 0s 713us/step - loss: 0.1664 - accuracy:
0.9528
Epoch 769/1000
7/7 [==============================] - 0s 851us/step - loss: 0.1661 - accuracy:
0.9528
Epoch 770/1000
7/7 [==============================] - 0s 759us/step - loss: 0.1659 - accuracy:
```

```
0.9528
Epoch 771/1000
7/7 [==============================] - 0s 736us/step - loss: 0.1657 - accuracy:
0.9528
Epoch 772/1000
7/7 [==============================] - 0s 659us/step - loss: 0.1655 - accuracy:
0.9528
Epoch 773/1000
7/7 [==============================] - 0s 777us/step - loss: 0.1652 - accuracy:
0.9528
Epoch 774/1000
7/7 [==============================] - 0s 806us/step - loss: 0.1651 - accuracy:
0.9528
Epoch 775/1000
7/7 [==============================] - 0s 833us/step - loss: 0.1649 - accuracy:
0.9528
Epoch 776/1000
7/7 [==============================] - 0s 783us/step - loss: 0.1647 - accuracy:
0.9528
Epoch 777/1000
7/7 [==============================] - 0s 696us/step - loss: 0.1645 - accuracy:
0.9528
Epoch 778/1000
7/7 [==============================] - 0s 859us/step - loss: 0.1644 - accuracy:
0.9528
Epoch 779/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1641 - accuracy:
0.9528
Epoch 780/1000
7/7 [==============================] - 0s 994us/step - loss: 0.1639 - accuracy:
0.9528
Epoch 781/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1637 - accuracy:
0.9528
Epoch 782/1000
7/7 [==============================] - 0s 916us/step - loss: 0.1635 - accuracy:
0.9528
Epoch 783/1000
7/7 [==============================] - 0s 688us/step - loss: 0.1634 - accuracy:
0.9528
Epoch 784/1000
7/7 [==============================] - 0s 648us/step - loss: 0.1631 - accuracy:
0.9528
Epoch 785/1000
7/7 [==============================] - 0s 749us/step - loss: 0.1629 - accuracy:
0.9528
Epoch 786/1000
7/7 [==============================] - 0s 731us/step - loss: 0.1627 - accuracy:
```

```
0.9528
Epoch 787/1000
7/7 [==============================] - 0s 723us/step - loss: 0.1625 - accuracy:
0.9528
Epoch 788/1000
7/7 [==============================] - 0s 807us/step - loss: 0.1623 - accuracy:
0.9528
Epoch 789/1000
7/7 [==============================] - 0s 752us/step - loss: 0.1621 - accuracy:
0.9528
Epoch 790/1000
7/7 [==============================] - 0s 823us/step - loss: 0.1619 - accuracy:
0.9528
Epoch 791/1000
7/7 [==============================] - 0s 661us/step - loss: 0.1617 - accuracy:
0.9528
Epoch 792/1000
7/7 [==============================] - 0s 709us/step - loss: 0.1616 - accuracy:
0.9528
Epoch 793/1000
7/7 [==============================] - 0s 753us/step - loss: 0.1614 - accuracy:
0.9528
Epoch 794/1000
7/7 [==============================] - 0s 643us/step - loss: 0.1612 - accuracy:
0.9528
Epoch 795/1000
7/7 [==============================] - 0s 839us/step - loss: 0.1610 - accuracy:
0.9528
Epoch 796/1000
7/7 [==============================] - 0s 997us/step - loss: 0.1608 - accuracy:
0.9528
Epoch 797/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1606 - accuracy:
0.9528
Epoch 798/1000
7/7 [==============================] - 0s 737us/step - loss: 0.1605 - accuracy:
0.9528
Epoch 799/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1602 - accuracy:
0.9575
Epoch 800/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1600 - accuracy:
0.9528
Epoch 801/1000
7/7 [==============================] - 0s 774us/step - loss: 0.1598 - accuracy:
0.9528
Epoch 802/1000
7/7 [==============================] - 0s 765us/step - loss: 0.1596 - accuracy:
```

```
0.9528
Epoch 803/1000
7/7 [==============================] - 0s 976us/step - loss: 0.1594 - accuracy:
0.9528
Epoch 804/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1592 - accuracy:
0.9528
Epoch 805/1000
7/7 [==============================] - 0s 697us/step - loss: 0.1590 - accuracy:
0.9528
Epoch 806/1000
7/7 [==============================] - 0s 704us/step - loss: 0.1588 - accuracy:
0.9575
Epoch 807/1000
7/7 [==============================] - 0s 737us/step - loss: 0.1586 - accuracy:
0.9575
Epoch 808/1000
7/7 [==============================] - 0s 674us/step - loss: 0.1584 - accuracy:
0.9575
Epoch 809/1000
7/7 [==============================] - 0s 764us/step - loss: 0.1583 - accuracy:
0.9575
Epoch 810/1000
7/7 [==============================] - 0s 733us/step - loss: 0.1580 - accuracy:
0.9575
Epoch 811/1000
7/7 [==============================] - 0s 675us/step - loss: 0.1579 - accuracy:
0.9575
Epoch 812/1000
7/7 [==============================] - 0s 712us/step - loss: 0.1577 - accuracy:
0.9575
Epoch 813/1000
7/7 [==============================] - 0s 643us/step - loss: 0.1575 - accuracy:
0.9575
Epoch 814/1000
7/7 [==============================] - 0s 633us/step - loss: 0.1573 - accuracy:
0.9575
Epoch 815/1000
7/7 [==============================] - 0s 658us/step - loss: 0.1571 - accuracy:
0.9575
Epoch 816/1000
7/7 [==============================] - 0s 693us/step - loss: 0.1570 - accuracy:
0.9623
Epoch 817/1000
7/7 [==============================] - 0s 653us/step - loss: 0.1567 - accuracy:
0.9623
Epoch 818/1000
7/7 [==============================] - 0s 725us/step - loss: 0.1566 - accuracy:
```

```
0.9623
Epoch 819/1000
7/7 [==============================] - 0s 722us/step - loss: 0.1563 - accuracy:
0.9623
Epoch 820/1000
7/7 [==============================] - 0s 688us/step - loss: 0.1561 - accuracy:
0.9623
Epoch 821/1000
7/7 [==============================] - 0s 766us/step - loss: 0.1560 - accuracy:
0.9623
Epoch 822/1000
7/7 [==============================] - 0s 688us/step - loss: 0.1557 - accuracy:
0.9623
Epoch 823/1000
7/7 [==============================] - 0s 635us/step - loss: 0.1556 - accuracy:
0.9623
Epoch 824/1000
7/7 [==============================] - 0s 726us/step - loss: 0.1554 - accuracy:
0.9623
Epoch 825/1000
7/7 [==============================] - 0s 711us/step - loss: 0.1552 - accuracy:
0.9623
Epoch 826/1000
7/7 [==============================] - 0s 828us/step - loss: 0.1550 - accuracy:
0.9623
Epoch 827/1000
7/7 [==============================] - 0s 719us/step - loss: 0.1548 - accuracy:
0.9623
Epoch 828/1000
7/7 [==============================] - 0s 687us/step - loss: 0.1547 - accuracy:
0.9623
Epoch 829/1000
7/7 [==============================] - 0s 689us/step - loss: 0.1545 - accuracy:
0.9623
Epoch 830/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1543 - accuracy:
0.9623
Epoch 831/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1541 - accuracy:
0.9623
Epoch 832/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1540 - accuracy:
0.9623
Epoch 833/1000
7/7 [==============================] - 0s 848us/step - loss: 0.1537 - accuracy:
0.9623
Epoch 834/1000
7/7 [==============================] - 0s 747us/step - loss: 0.1535 - accuracy:
```

```
0.9623
Epoch 835/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1534 - accuracy:
0.9623
Epoch 836/1000
7/7 [==============================] - 0s 949us/step - loss: 0.1532 - accuracy:
0.9623
Epoch 837/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1530 - accuracy:
0.9623
Epoch 838/1000
7/7 [==============================] - 0s 712us/step - loss: 0.1528 - accuracy:
0.9623
Epoch 839/1000
7/7 [==============================] - 0s 873us/step - loss: 0.1527 - accuracy:
0.9623
Epoch 840/1000
7/7 [==============================] - 0s 808us/step - loss: 0.1524 - accuracy:
0.9623
Epoch 841/1000
7/7 [==============================] - 0s 838us/step - loss: 0.1522 - accuracy:
0.9623
Epoch 842/1000
7/7 [==============================] - 0s 797us/step - loss: 0.1521 - accuracy:
0.9623
Epoch 843/1000
7/7 [==============================] - 0s 669us/step - loss: 0.1519 - accuracy:
0.9623
Epoch 844/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1517 - accuracy:
0.9623
Epoch 845/1000
7/7 [==============================] - 0s 986us/step - loss: 0.1515 - accuracy:
0.9623
Epoch 846/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1513 - accuracy:
0.9623
Epoch 847/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1511 - accuracy:
0.9623
Epoch 848/1000
7/7 [==============================] - 0s 863us/step - loss: 0.1509 - accuracy:
0.9623
Epoch 849/1000
7/7 [==============================] - 0s 807us/step - loss: 0.1508 - accuracy:
0.9623
Epoch 850/1000
7/7 [==============================] - 0s 746us/step - loss: 0.1505 - accuracy:
```

```
0.9623
Epoch 851/1000
7/7 [==============================] - 0s 657us/step - loss: 0.1504 - accuracy:
0.9623
Epoch 852/1000
7/7 [==============================] - 0s 925us/step - loss: 0.1502 - accuracy:
0.9623
Epoch 853/1000
7/7 [==============================] - 0s 762us/step - loss: 0.1500 - accuracy:
0.9623
Epoch 854/1000
7/7 [==============================] - 0s 729us/step - loss: 0.1499 - accuracy:
0.9623
Epoch 855/1000
7/7 [==============================] - 0s 924us/step - loss: 0.1496 - accuracy:
0.9623
Epoch 856/1000
7/7 [==============================] - 0s 938us/step - loss: 0.1495 - accuracy:
0.9623
Epoch 857/1000
7/7 [==============================] - 0s 724us/step - loss: 0.1493 - accuracy:
0.9623
Epoch 858/1000
7/7 [==============================] - 0s 741us/step - loss: 0.1491 - accuracy:
0.9623
Epoch 859/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1489 - accuracy:
0.9623
Epoch 860/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1488 - accuracy:
0.9623
Epoch 861/1000
7/7 [==============================] - 0s 955us/step - loss: 0.1485 - accuracy:
0.9623
Epoch 862/1000
7/7 [==============================] - 0s 769us/step - loss: 0.1484 - accuracy:
0.9623
Epoch 863/1000
7/7 [==============================] - 0s 802us/step - loss: 0.1482 - accuracy:
0.9623
Epoch 864/1000
7/7 [==============================] - 0s 977us/step - loss: 0.1480 - accuracy:
0.9623
Epoch 865/1000
7/7 [==============================] - 0s 701us/step - loss: 0.1479 - accuracy:
0.9623
Epoch 866/1000
7/7 [==============================] - 0s 706us/step - loss: 0.1477 - accuracy:
```

```
0.9623
Epoch 867/1000
7/7 [==============================] - 0s 755us/step - loss: 0.1476 - accuracy:
0.9623
Epoch 868/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1474 - accuracy:
0.9623
Epoch 869/1000
7/7 [==============================] - 0s 996us/step - loss: 0.1472 - accuracy:
0.9623
Epoch 870/1000
7/7 [==============================] - 0s 798us/step - loss: 0.1471 - accuracy:
0.9623
Epoch 871/1000
7/7 [==============================] - 0s 798us/step - loss: 0.1467 - accuracy:
0.9623
Epoch 872/1000
7/7 [==============================] - 0s 688us/step - loss: 0.1466 - accuracy:
0.9623
Epoch 873/1000
7/7 [==============================] - 0s 883us/step - loss: 0.1465 - accuracy:
0.9623
Epoch 874/1000
7/7 [==============================] - 0s 898us/step - loss: 0.1463 - accuracy:
0.9623
Epoch 875/1000
7/7 [==============================] - 0s 856us/step - loss: 0.1461 - accuracy:
0.9623
Epoch 876/1000
7/7 [==============================] - 0s 630us/step - loss: 0.1459 - accuracy:
0.9623
Epoch 877/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1457 - accuracy:
0.9623
Epoch 878/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1455 - accuracy:
0.9623
Epoch 879/1000
7/7 [==============================] - 0s 919us/step - loss: 0.1453 - accuracy:
0.9623
Epoch 880/1000
7/7 [==============================] - 0s 770us/step - loss: 0.1452 - accuracy:
0.9623
Epoch 881/1000
7/7 [==============================] - 0s 899us/step - loss: 0.1450 - accuracy:
0.9623
Epoch 882/1000
7/7 [==============================] - 0s 794us/step - loss: 0.1448 - accuracy:
```

```
0.9623
Epoch 883/1000
7/7 [==============================] - 0s 830us/step - loss: 0.1445 - accuracy:
0.9623
Epoch 884/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1444 - accuracy:
0.9623
Epoch 885/1000
7/7 [==============================] - 0s 732us/step - loss: 0.1443 - accuracy:
0.9623
Epoch 886/1000
7/7 [==============================] - 0s 651us/step - loss: 0.1441 - accuracy:
0.9623
Epoch 887/1000
7/7 [==============================] - 0s 913us/step - loss: 0.1439 - accuracy:
0.9623
Epoch 888/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1437 - accuracy:
0.9623
Epoch 889/1000
7/7 [==============================] - 0s 855us/step - loss: 0.1435 - accuracy:
0.9623
Epoch 890/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1433 - accuracy:
0.9623
Epoch 891/1000
7/7 [==============================] - 0s 700us/step - loss: 0.1432 - accuracy:
0.9623
Epoch 892/1000
7/7 [==============================] - 0s 897us/step - loss: 0.1430 - accuracy:
0.9623
Epoch 893/1000
7/7 [==============================] - 0s 716us/step - loss: 0.1428 - accuracy:
0.9623
Epoch 894/1000
7/7 [==============================] - 0s 799us/step - loss: 0.1426 - accuracy:
0.9623
Epoch 895/1000
7/7 [==============================] - 0s 827us/step - loss: 0.1424 - accuracy:
0.9623
Epoch 896/1000
7/7 [==============================] - 0s 811us/step - loss: 0.1422 - accuracy:
0.9623
Epoch 897/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1421 - accuracy:
0.9623
Epoch 898/1000
7/7 [==============================] - 0s 714us/step - loss: 0.1419 - accuracy:
```

```
0.9623
Epoch 899/1000
7/7 [==============================] - 0s 690us/step - loss: 0.1416 - accuracy:
0.9623
Epoch 900/1000
7/7 [==============================] - 0s 683us/step - loss: 0.1415 - accuracy:
0.9623
Epoch 901/1000
7/7 [==============================] - 0s 639us/step - loss: 0.1413 - accuracy:
0.9623
Epoch 902/1000
7/7 [==============================] - 0s 711us/step - loss: 0.1411 - accuracy:
0.9623
Epoch 903/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1409 - accuracy:
0.9623
Epoch 904/1000
7/7 [==============================] - 0s 780us/step - loss: 0.1408 - accuracy:
0.9623
Epoch 905/1000
7/7 [==============================] - 0s 694us/step - loss: 0.1405 - accuracy:
0.9623
Epoch 906/1000
7/7 [==============================] - 0s 624us/step - loss: 0.1404 - accuracy:
0.9623
Epoch 907/1000
7/7 [==============================] - 0s 714us/step - loss: 0.1402 - accuracy:
0.9623
Epoch 908/1000
7/7 [==============================] - 0s 830us/step - loss: 0.1400 - accuracy:
0.9623
Epoch 909/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1398 - accuracy:
0.9623
Epoch 910/1000
7/7 [==============================] - 0s 990us/step - loss: 0.1397 - accuracy:
0.9623
Epoch 911/1000
7/7 [==============================] - 0s 860us/step - loss: 0.1394 - accuracy:
0.9623
Epoch 912/1000
7/7 [==============================] - 0s 816us/step - loss: 0.1393 - accuracy:
0.9623
Epoch 913/1000
7/7 [==============================] - 0s 713us/step - loss: 0.1391 - accuracy:
0.9623
Epoch 914/1000
7/7 [==============================] - 0s 875us/step - loss: 0.1389 - accuracy:
```

```
0.9623
Epoch 915/1000
7/7 [==============================] - 0s 784us/step - loss: 0.1387 - accuracy:
0.9623
Epoch 916/1000
7/7 [==============================] - 0s 688us/step - loss: 0.1385 - accuracy:
0.9623
Epoch 917/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1384 - accuracy:
0.9623
Epoch 918/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1382 - accuracy:
0.9623
Epoch 919/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1380 - accuracy:
0.9623
Epoch 920/1000
7/7 [==============================] - 0s 861us/step - loss: 0.1378 - accuracy:
0.9623
Epoch 921/1000
7/7 [==============================] - 0s 714us/step - loss: 0.1376 - accuracy:
0.9623
Epoch 922/1000
7/7 [==============================] - 0s 683us/step - loss: 0.1374 - accuracy:
0.9623
Epoch 923/1000
7/7 [==============================] - 0s 952us/step - loss: 0.1374 - accuracy:
0.9623
Epoch 924/1000
7/7 [==============================] - 0s 707us/step - loss: 0.1371 - accuracy:
0.9670
Epoch 925/1000
7/7 [==============================] - 0s 723us/step - loss: 0.1369 - accuracy:
0.9670
Epoch 926/1000
7/7 [==============================] - 0s 729us/step - loss: 0.1368 - accuracy:
0.9623
Epoch 927/1000
7/7 [==============================] - 0s 650us/step - loss: 0.1366 - accuracy:
0.9623
Epoch 928/1000
7/7 [==============================] - 0s 873us/step - loss: 0.1365 - accuracy:
0.9670
Epoch 929/1000
7/7 [==============================] - 0s 707us/step - loss: 0.1362 - accuracy:
0.9623
Epoch 930/1000
7/7 [==============================] - 0s 657us/step - loss: 0.1361 - accuracy:
```

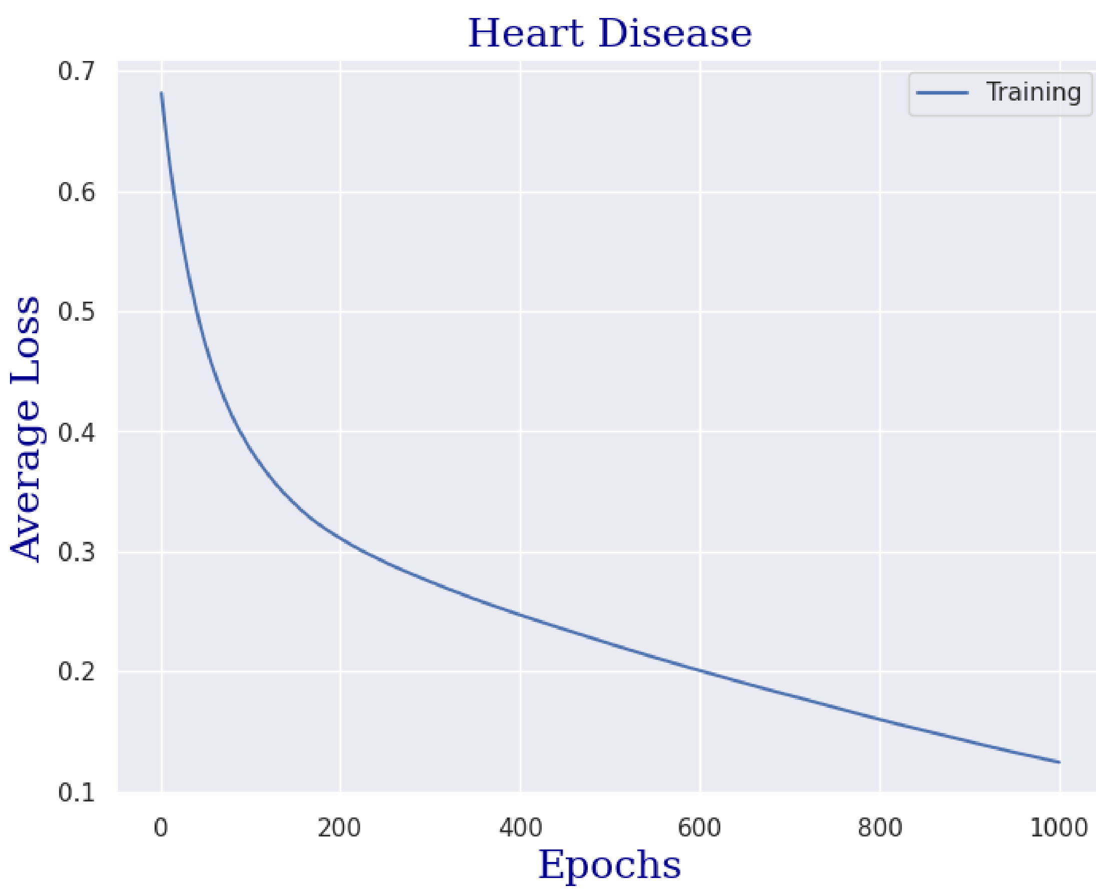
```
0.9670
Epoch 931/1000
7/7 [==============================] - 0s 735us/step - loss: 0.1359 - accuracy:
0.9623
Epoch 932/1000
7/7 [==============================] - 0s 691us/step - loss: 0.1357 - accuracy:
0.9670
Epoch 933/1000
7/7 [==============================] - 0s 690us/step - loss: 0.1355 - accuracy:
0.9670
Epoch 934/1000
7/7 [==============================] - 0s 696us/step - loss: 0.1354 - accuracy:
0.9670
Epoch 935/1000
7/7 [==============================] - 0s 718us/step - loss: 0.1352 - accuracy:
0.9670
Epoch 936/1000
7/7 [==============================] - 0s 734us/step - loss: 0.1350 - accuracy:
0.9670
Epoch 937/1000
7/7 [==============================] - 0s 828us/step - loss: 0.1348 - accuracy:
0.9670
Epoch 938/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1347 - accuracy:
0.9670
Epoch 939/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1345 - accuracy:
0.9670
Epoch 940/1000
7/7 [==============================] - 0s 843us/step - loss: 0.1343 - accuracy:
0.9670
Epoch 941/1000
7/7 [==============================] - 0s 741us/step - loss: 0.1341 - accuracy:
0.9670
Epoch 942/1000
7/7 [==============================] - 0s 705us/step - loss: 0.1339 - accuracy:
0.9670
Epoch 943/1000
7/7 [==============================] - 0s 730us/step - loss: 0.1338 - accuracy:
0.9670
Epoch 944/1000
7/7 [==============================] - 0s 845us/step - loss: 0.1336 - accuracy:
0.9670
Epoch 945/1000
7/7 [==============================] - 0s 724us/step - loss: 0.1334 - accuracy:
0.9670
Epoch 946/1000
7/7 [==============================] - 0s 819us/step - loss: 0.1332 - accuracy:
```

```
0.9670
Epoch 947/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1330 - accuracy:
0.9670
Epoch 948/1000
7/7 [==============================] - 0s 927us/step - loss: 0.1329 - accuracy:
0.9670
Epoch 949/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1328 - accuracy:
0.9670
Epoch 950/1000
7/7 [==============================] - 0s 717us/step - loss: 0.1326 - accuracy:
0.9670
Epoch 951/1000
7/7 [==============================] - 0s 862us/step - loss: 0.1324 - accuracy:
0.9670
Epoch 952/1000
7/7 [==============================] - 0s 700us/step - loss: 0.1323 - accuracy:
0.9670
Epoch 953/1000
7/7 [==============================] - 0s 761us/step - loss: 0.1321 - accuracy:
0.9670
Epoch 954/1000
7/7 [==============================] - 0s 676us/step - loss: 0.1318 - accuracy:
0.9670
Epoch 955/1000
7/7 [==============================] - 0s 710us/step - loss: 0.1317 - accuracy:
0.9670
Epoch 956/1000
7/7 [==============================] - 0s 935us/step - loss: 0.1316 - accuracy:
0.9670
Epoch 957/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1313 - accuracy:
0.9670
Epoch 958/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1312 - accuracy:
0.9670
Epoch 959/1000
7/7 [==============================] - 0s 744us/step - loss: 0.1310 - accuracy:
0.9670
Epoch 960/1000
7/7 [==============================] - 0s 680us/step - loss: 0.1308 - accuracy:
0.9670
Epoch 961/1000
7/7 [==============================] - 0s 708us/step - loss: 0.1307 - accuracy:
0.9670
Epoch 962/1000
7/7 [==============================] - 0s 660us/step - loss: 0.1305 - accuracy:
```

```
0.9670
Epoch 963/1000
7/7 [==============================] - 0s 726us/step - loss: 0.1303 - accuracy:
0.9670
Epoch 964/1000
7/7 [==============================] - 0s 764us/step - loss: 0.1301 - accuracy:
0.9670
Epoch 965/1000
7/7 [==============================] - 0s 713us/step - loss: 0.1300 - accuracy:
0.9670
Epoch 966/1000
7/7 [==============================] - 0s 976us/step - loss: 0.1299 - accuracy:
0.9670
Epoch 967/1000
7/7 [==============================] - 0s 726us/step - loss: 0.1297 - accuracy:
0.9670
Epoch 968/1000
7/7 [==============================] - 0s 759us/step - loss: 0.1295 - accuracy:
0.9670
Epoch 969/1000
7/7 [==============================] - 0s 657us/step - loss: 0.1294 - accuracy:
0.9670
Epoch 970/1000
7/7 [==============================] - 0s 655us/step - loss: 0.1293 - accuracy:
0.9670
Epoch 971/1000
7/7 [==============================] - 0s 852us/step - loss: 0.1290 - accuracy:
0.9670
Epoch 972/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1288 - accuracy:
0.9670
Epoch 973/1000
7/7 [==============================] - 0s 807us/step - loss: 0.1287 - accuracy:
0.9670
Epoch 974/1000
7/7 [==============================] - 0s 683us/step - loss: 0.1287 - accuracy:
0.9670
Epoch 975/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1283 - accuracy:
0.9670
Epoch 976/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1282 - accuracy:
0.9670
Epoch 977/1000
7/7 [==============================] - 0s 726us/step - loss: 0.1281 - accuracy:
0.9670
Epoch 978/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1278 - accuracy:
```

```
0.9670
Epoch 979/1000
7/7 [==============================] - 0s 892us/step - loss: 0.1277 - accuracy:
0.9670
Epoch 980/1000
7/7 [==============================] - 0s 748us/step - loss: 0.1275 - accuracy:
0.9670
Epoch 981/1000
7/7 [==============================] - 0s 752us/step - loss: 0.1274 - accuracy:
0.9670
Epoch 982/1000
7/7 [==============================] - 0s 692us/step - loss: 0.1272 - accuracy:
0.9670
Epoch 983/1000
7/7 [==============================] - 0s 731us/step - loss: 0.1270 - accuracy:
0.9670
Epoch 984/1000
7/7 [==============================] - 0s 728us/step - loss: 0.1268 - accuracy:
0.9670
Epoch 985/1000
7/7 [==============================] - 0s 777us/step - loss: 0.1267 - accuracy:
0.9670
Epoch 986/1000
7/7 [==============================] - 0s 1ms/step - loss: 0.1265 - accuracy:
0.9670
Epoch 987/1000
7/7 [==============================] - 0s 885us/step - loss: 0.1263 - accuracy:
0.9670
Epoch 988/1000
7/7 [==============================] - 0s 991us/step - loss: 0.1262 - accuracy:
0.9670
Epoch 989/1000
7/7 [==============================] - 0s 1000us/step - loss: 0.1261 - accuracy:
0.9670
Epoch 990/1000
7/7 [==============================] - 0s 775us/step - loss: 0.1259 - accuracy:
0.9670
Epoch 991/1000
7/7 [==============================] - 0s 618us/step - loss: 0.1258 - accuracy:
0.9670
Epoch 992/1000
7/7 [==============================] - 0s 675us/step - loss: 0.1256 - accuracy:
0.9670
Epoch 993/1000
7/7 [==============================] - 0s 705us/step - loss: 0.1254 - accuracy:
0.9670
Epoch 994/1000
7/7 [==============================] - 0s 735us/step - loss: 0.1253 - accuracy:
```

```
0.9670
Epoch 995/1000
7/7 [==============================] - 0s 746us/step - loss: 0.1251 - accuracy:
0.9670
Epoch 996/1000
7/7 [==============================] - 0s 769us/step - loss: 0.1250 - accuracy:
0.9670
Epoch 997/1000
7/7 [==============================] - 0s 707us/step - loss: 0.1248 - accuracy:
0.9670
Epoch 998/1000
7/7 [==============================] - 0s 713us/step - loss: 0.1246 - accuracy:
0.9670
Epoch 999/1000
7/7 [==============================] - 0s 685us/step - loss: 0.1244 - accuracy:
0.9670
Epoch 1000/1000
7/7 [==============================] - 0s 2ms/step - loss: 0.1243 - accuracy:
0.9670
Accuracies of the network on test data: 84.62%
Accuracies of the network on training data: 96.70%
```



5.3.2 Comparison Models

After training, testing, and obtaining the respective accuracies for the selected models, our next step involves comparing all the developed models. This comparison is crucial for evaluating the performance of each model and making well-informed decisions about which one is most suitable for our specific problem (in our case heat disease prediction). By carefully analyzing the achieved accuracies across different models, as highlighted in the previous section, we can pinpoint the model that demonstrates the highest classification performance. This informed decision-making process enables us to select the optimal model for deployment or consider further optimization if needed. This section offers a succinct overview of the accuracy scores achieved by various machine learning algorithms in the last section on the heart disease prediction. The focus here is primarily on the accuracy metrics for both the test and training datasets. The presented table showcases a sorted list of algorithms along with their corresponding test values.

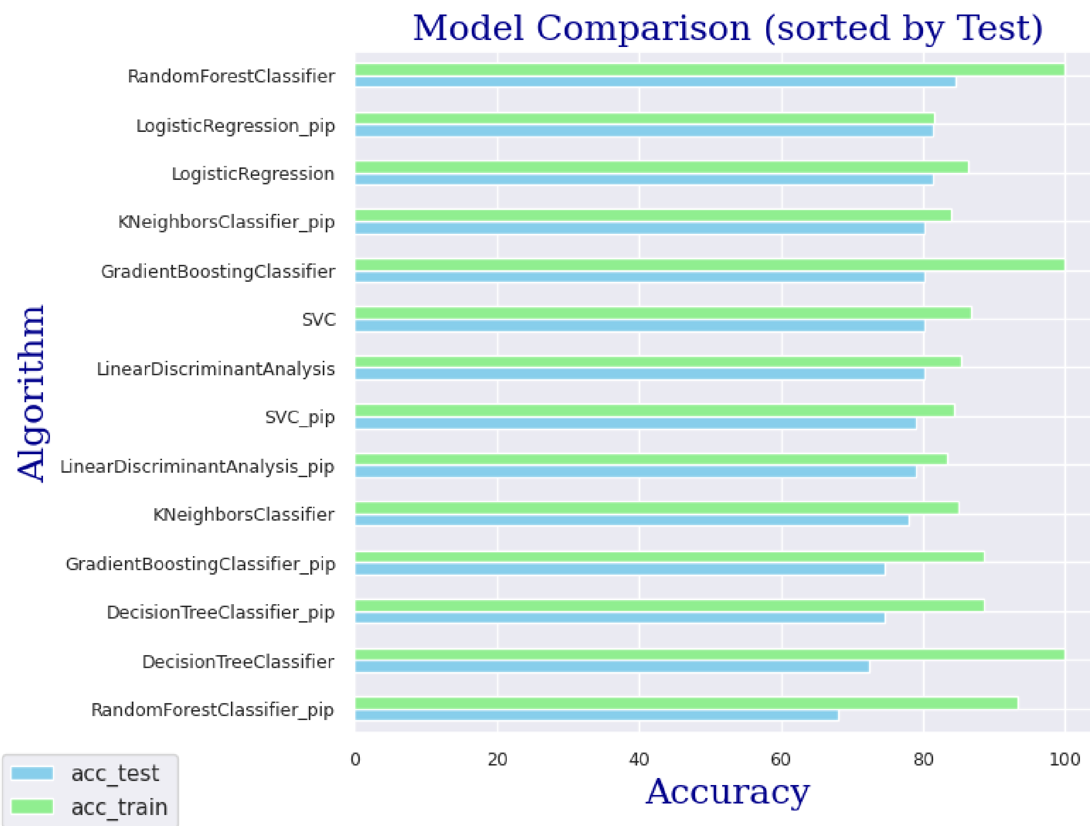
[193]:
```
results = evaluator.results
df_comparsion = pd.DataFrame.from_dict(results,
orient='index' ).transpose().sort_values('acc_test',ascending= True)
# Sorted  by Test Accuracy
df_comparsion
```

	acc_test	acc_train
RandomForestClassifier_pip	68.13	93.40
DecisionTreeClassifier	72.53	100.00
DecisionTreeClassifier_pip	74.73	88.68
GradientBoostingClassifier_pip	74.73	88.68
KNeighborsClassifier	78.02	84.91
LinearDiscriminantAnalysis_pip	79.12	83.49
SVC_pip	79.12	84.43
LinearDiscriminantAnalysis	80.22	85.38
SVC	80.22	86.79
GradientBoostingClassifier	80.22	100.00
KNeighborsClassifier_pip	80.22	83.96
LogisticRegression	81.32	86.32
LogisticRegression_pip	81.32	81.60
RandomForestClassifier	84.62	100.00

[198]:

```
fg11 = df_comparsion.plot(kind='barh', color=['skyblue', 'lightgreen'],␣
 ↪fontsize=9, legend=False)
plt.figlegend( ncol=1, frameon=True,loc= 'lower left')
#plt.figure(dpi=1000)

fg11.set_xlabel('Accuracy', fontdict=font)
fg11.set_ylabel('Algorithm', fontdict=font)
fg11.set_title('Model Comparison (sorted by Test)',  fontdict=font)
```

```
plt.tight_layout()
```



6 Conclusion

In this analysis, we embarked on a comprehensive journey through the development and evaluation of data-driven classification models for the task of heart disease prediction. Starting with the selection of relevant features and the construction of a well-balanced training and testing dataset, we delved into the heart of model selection and evaluation.

The Data-driven Model section illuminated the significance of data-driven approaches in solving classification problems. By partitioning computational methods into classification categories, we zoomed in on classification—a process aimed at predicting class labels for new data based on its features. Feature Selection emerged as a pivotal step, involving the identification of crucial features that drive accurate predictions. Our use of the Sequential Feature Selector (SFS) with a customized configuration allowed us to hone in on informative features while mitigating dimensionality issues.

The Train-Test Split step was a foundational preparation, splitting the dataset into training and testing subsets to enable robust model evaluation.

The heart of the analysis, Model Selection and Evaluation, showcased our commitment to informed decision-making. Leveraging the ModelEvaluator class, we systematically assessed multiple classification algorithms based on essential metrics like precision, recall, F1-score, support, and accuracies. This approach not only ensured a rigorous comparison but also facilitated visualization of ROC curves and confusion matrices. The utilization of an objective-oriented programming (OOP) paradigm enhanced the modularity and reusability of our code, underpinning a streamlined and comprehensive evaluation process.

In the Comparison Models section, we distilled the essence of our efforts into a concise yet revealing table of accuracy scores. This table illuminated the performance of various algorithms on both the test and training datasets, providing valuable insights into their generalization capabilities and potential overfitting tendencies.

As we conclude, it's imperative to acknowledge that while accuracy is an essential metric, the evaluation process should be multifaceted. The decision on the best model should be informed by precision, recall, F1-score, domain knowledge, and other contextual considerations.

Ultimately, this analysis serves as a strong foundation for the deployment of an effective classification model for heart disease prediction. It underscores the iterative and exploratory nature of machine learning, encouraging ongoing optimization, fine-tuning, and model enhancement. Armed with this knowledge, we're equipped to make informed decisions in real-world scenarios, contributing to the advancement of healthcare and data-driven solutions.

Back to Table of Contents