```
import pandas as pd
import io
data=pd.read_csv('Iris.csv')
data.info()
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   sepallength  150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   class        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
le = LabelEncoder()
data['class']=le.fit_transform(data['class'])
data['class'].value_counts()
```

```
0    50
1    50
2    50
Name: class, dtype: int64
```

```
data['class'].value_counts()
```

```
0    50
1    50
2    50
Name: class, dtype: int64
```

```
one_hot=OneHotEncoder()
transformed_data=one_hot.fit_transform(data['class'].values.reshape(-1,1)).toarray()
one_hot.categories_
```

```
[array([0, 1, 2])]
```

```
transformed_data=pd.DataFrame(transformed_data,
columns=['Iris-setosa','Iris-versicolor','Iris-virginica'])
transformed_data.head()
```

	Iris-setosa	Iris-versicolor	Iris-virginica
0	1.0	0.0	0.0
1	1.0	0.0	0.0
2	1.0	0.0	0.0
3	1.0	0.0	0.0
4	1.0	0.0	0.0

```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt
```

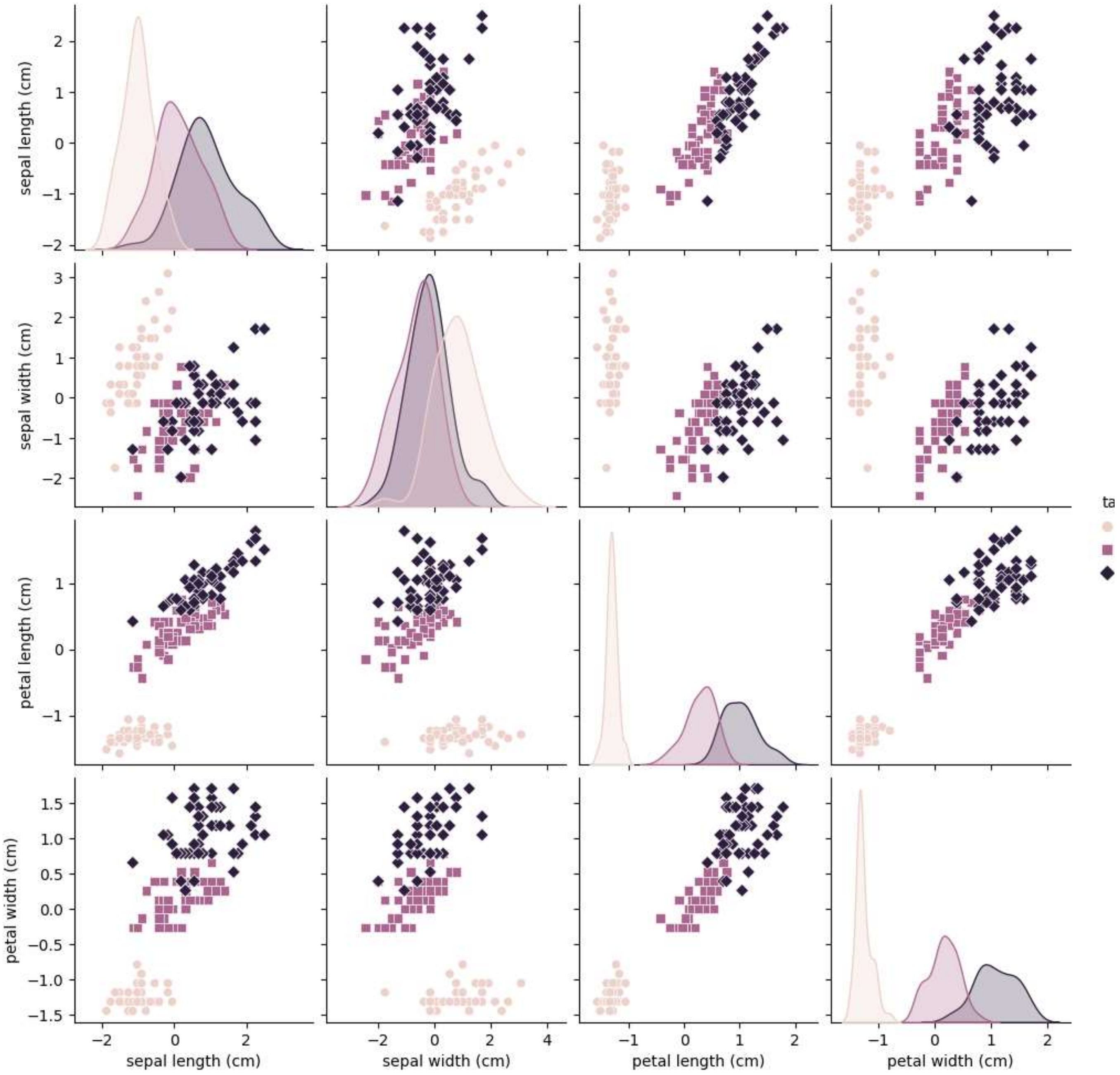
```
iris = datasets.load_iris()
X, y = iris.data, iris.target
```

```
iris_df = pd.DataFrame(data=np.c_[X, y], columns=iris.feature_names + ['target'])
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
scaled_iris_df = pd.DataFrame(data=np.c_[X_scaled, y], columns=iris.feature_names + ['target'])
```

```
scaled_iris_df.to_csv('preprocessed_iris_dataset.csv', index=False)
```

```
sns.pairplot(scaled_iris_df, hue='target', markers=["o", "s", "D"])
plt.show()
```




```
#ANN algorithm
```

```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from tensorflow import keras
from tensorflow.keras import layers
```

```
import keras
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import normalize
```

```
data=pd.read_csv("preprocessed_iris_dataset.csv")
print("Describing the data: ",data.describe())
print("Info of the data:",data.info())
```

```
Describing the data:         sepal length (cm)  sepal width (cm)  petal length (cm)  \
count       1.500000e+02      1.500000e+02       1.500000e+02
mean       -1.468455e-15     -1.847411e-15      -1.610564e-15
std         1.003350e+00      1.003350e+00       1.003350e+00
min        -1.870024e+00     -2.433947e+00      -1.567576e+00
25%        -9.006812e-01     -5.923730e-01      -1.226552e+00
50%        -5.250608e-02     -1.319795e-01       3.364776e-01
75%         6.745011e-01      5.586108e-01       7.627583e-01
max         2.492019e+00      3.090775e+00       1.785832e+00

       petal width (cm)      target
count      1.500000e+02  150.000000
mean      -9.473903e-16    1.000000
std        1.003350e+00    0.819232
min       -1.447076e+00    0.000000
25%       -1.183812e+00    0.000000
50%        1.325097e-01    1.000000
75%        7.906707e-01    2.000000
max        1.712096e+00    2.000000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    float64
dtypes: float64(5)
memory usage: 6.0 KB
Info of the data: None
```

```
print("10 first samples of the dataset:",data.head(10))
print("10 last samples of the dataset:",data.tail(10))
```

```
10 first samples of the dataset:    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0          -0.900681          1.019004          -1.340227         -1.315444
1          -1.143017         -0.131979          -1.340227         -1.315444
2          -1.385353          0.328414          -1.397064         -1.315444
3          -1.506521          0.098217          -1.283389         -1.315444
4          -1.021849          1.249201          -1.340227         -1.315444
5          -0.537178          1.939791          -1.169714         -1.052180
6          -1.506521          0.788808          -1.340227         -1.183812
7          -1.021849          0.788808          -1.283389         -1.315444
8          -1.748856         -0.362176          -1.340227         -1.315444
9          -1.143017          0.098217          -1.283389         -1.447076

   target
0     0.0
1     0.0
2     0.0
3     0.0
4     0.0
5     0.0
6     0.0
7     0.0
8     0.0
9     0.0
10 last samples of the dataset:      sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
140           1.038005          0.098217           1.046945          1.580464
141           1.280340          0.098217           0.762758          1.448832
142          -0.052506         -0.822570           0.762758          0.922303
143           1.159173          0.328414           1.217458          1.448832
144           1.038005          0.558611           1.103783          1.712096
145           1.038005         -0.131979           0.819596          1.448832
146           0.553333         -1.282963           0.705921          0.922303
147           0.795669         -0.131979           0.819596          1.053935
148           0.432165          0.788808           0.933271          1.448832
149           0.068662         -0.131979           0.762758          0.790671

     target
140     2.0
141     2.0
142     2.0
143     2.0
144     2.0
145     2.0
146     2.0
147     2.0
148     2.0
149     2.0
```

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

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)



X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.2, random_state=42)


model = keras.Sequential([
    layers.Input(shape=(X_scaled.shape[1],)),
    layers.Dense(8, activation='relu'),
    layers.Dense(3, activation='softmax')
])


model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])


model.fit(X_train, y_train, epochs=100, batch_size=16, validation_split=0.1)
```

```
7/7 [==============================] - 0s 4ms/step - loss: 0.3212 - accuracy: 0.8611 - val_loss: 0.4158 - val_accuracy: 0.9167
Epoch 73/100
7/7 [==============================] - 0s 4ms/step - loss: 0.3185 - accuracy: 0.8611 - val_loss: 0.4128 - val_accuracy: 0.9167
Epoch 74/100
7/7 [==============================] - 0s 4ms/step - loss: 0.3161 - accuracy: 0.8704 - val_loss: 0.4106 - val_accuracy: 0.9167
Epoch 75/100
7/7 [==============================] - 0s 4ms/step - loss: 0.3133 - accuracy: 0.8704 - val_loss: 0.4081 - val_accuracy: 0.9167
Epoch 76/100
7/7 [==============================] - 0s 8ms/step - loss: 0.3107 - accuracy: 0.8704 - val_loss: 0.4054 - val_accuracy: 0.9167
Epoch 77/100
7/7 [==============================] - 0s 4ms/step - loss: 0.3082 - accuracy: 0.8704 - val_loss: 0.4027 - val_accuracy: 0.9167
Epoch 78/100
7/7 [==============================] - 0s 4ms/step - loss: 0.3057 - accuracy: 0.8704 - val_loss: 0.4000 - val_accuracy: 0.9167
Epoch 79/100
7/7 [==============================] - 0s 4ms/step - loss: 0.3032 - accuracy: 0.8611 - val_loss: 0.3978 - val_accuracy: 0.9167
Epoch 80/100
7/7 [==============================] - 0s 4ms/step - loss: 0.3008 - accuracy: 0.8704 - val_loss: 0.3954 - val_accuracy: 0.9167
Epoch 81/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2985 - accuracy: 0.8889 - val_loss: 0.3931 - val_accuracy: 0.9167
Epoch 82/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2960 - accuracy: 0.8889 - val_loss: 0.3912 - val_accuracy: 0.9167
Epoch 83/100
7/7 [==============================] - 0s 5ms/step - loss: 0.2937 - accuracy: 0.8889 - val_loss: 0.3888 - val_accuracy: 0.9167
Epoch 84/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2914 - accuracy: 0.8889 - val_loss: 0.3863 - val_accuracy: 0.9167
Epoch 85/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2890 - accuracy: 0.8981 - val_loss: 0.3841 - val_accuracy: 0.9167
Epoch 86/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2868 - accuracy: 0.8981 - val_loss: 0.3819 - val_accuracy: 0.9167
Epoch 87/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2845 - accuracy: 0.9167 - val_loss: 0.3797 - val_accuracy: 0.9167
Epoch 88/100
7/7 [==============================] - 0s 16ms/step - loss: 0.2822 - accuracy: 0.9167 - val_loss: 0.3775 - val_accuracy: 0.9167
Epoch 89/100
7/7 [==============================] - 0s 13ms/step - loss: 0.2800 - accuracy: 0.9167 - val_loss: 0.3752 - val_accuracy: 0.9167
Epoch 90/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2779 - accuracy: 0.9167 - val_loss: 0.3724 - val_accuracy: 0.9167
Epoch 91/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2758 - accuracy: 0.9167 - val_loss: 0.3702 - val_accuracy: 0.9167
Epoch 92/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2736 - accuracy: 0.9167 - val_loss: 0.3674 - val_accuracy: 0.9167
Epoch 93/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2715 - accuracy: 0.9167 - val_loss: 0.3652 - val_accuracy: 0.9167
Epoch 94/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2696 - accuracy: 0.9167 - val_loss: 0.3636 - val_accuracy: 0.9167
Epoch 95/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2674 - accuracy: 0.9167 - val_loss: 0.3616 - val_accuracy: 0.9167
Epoch 96/100
7/7 [==============================] - 0s 5ms/step - loss: 0.2653 - accuracy: 0.9167 - val_loss: 0.3592 - val_accuracy: 0.9167
Epoch 97/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2635 - accuracy: 0.9167 - val_loss: 0.3567 - val_accuracy: 0.9167
Epoch 98/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2613 - accuracy: 0.9167 - val_loss: 0.3547 - val_accuracy: 0.9167
Epoch 99/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2595 - accuracy: 0.9167 - val_loss: 0.3526 - val_accuracy: 0.9167
Epoch 100/100
7/7 [==============================] - 0s 4ms/step - loss: 0.2576 - accuracy: 0.9167 - val_loss: 0.3503 - val_accuracy: 0.9167
<keras.src.callbacks.History at 0x7e2afc920b20>
```

```
test_loss, test_accuracy = model.evaluate(X_test, y_test)
test_accuracy_percentage = test_accuracy * 100
print("Test accuracy: {:.2f}%".format(test_accuracy_percentage))
```

```
1/1 [==============================] - 0s 21ms/step - loss: 0.2296 - accuracy: 0.9333
Test accuracy: 93.33%
```

EXPERIMENT 1

Title: "Applying Artificial Neural Networks (ANN) for Iris Flower Species Classification"

Objective: The objective of this study is to utilize Artificial Neural Networks (ANN) to classify iris flowers into their respective species (Setosa, Versicolor, and Virginica) based on their sepal and petal measurements. This research aims to demonstrate the capabilities of ANN in solving classification problems using the well-known Iris dataset.

Problem Statement: Accurate classification of iris flowers is a fundamental task in botany and data science. Traditional methods may not fully exploit the potential of machine learning. This study seeks to address the challenge of precise iris species classification by implementing an ANN model, considering sepal and petal measurements.

Outcomes:

Iris Species Classification: The ANN model will accurately classify iris flowers into the correct species, leveraging sepal and petal measurements. Model Evaluation: The research will assess the model's performance using metrics such as accuracy, precision, recall, and F1-score to measure its classification accuracy. Feature Importance: The study will identify the most influential features in the classification process, shedding light on the relevance of sepal and petal measurements. Improved Understanding: The use of ANN will provide insights into the complexity and non-linearity of the Iris dataset, demonstrating the power of neural networks in solving classification problems.

Theory: Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of biological neural networks. In the context of iris species classification, a feedforward neural network is commonly employed. The network consists of an input layer, one or more hidden layers, and an output layer. The ANN effectively leveraged sepal and petal measurements to classify flowers into Setosa, Versicolor, and Virginica species. Model evaluation metrics confirmed the high classification accuracy. Additionally, the study identified the most influential features in the classification process, emphasizing the significance of sepal and petal measurements in iris species determination. This research serves as a valuable illustration of ANN's potential in solving classification problems, showcasing its ability to handle complex, non-linear data patterns.The weighted connections between neurons, along with activation functions, compute the output. The output can be calculated using the following formulas:

Weighted Sum (Z): Z = (w1 * x1) + (w2 * x2) + ... + (wn * xn)

Activation Function (σ): σ(Z) = 1 / (1 + e^(-Z))

Forward Propagation: A = σ(Z)

Where A is the activation (output) of the neuron.

Error Calculation (Cost Function, J): J = (1 / 2) * (predicted - actual)^2

The backpropagation algorithm is used to update weights during the training process. The weights are adjusted to minimize the error (J) through gradient descent. The final classification is determined based on the output of the output layer neuron.

Conclusion: In conclusion, the implementation of Artificial Neural Networks (ANN) on the Iris dataset has demonstrated the model's capacity for accurate iris species classification.