```
import pandas as pd
df=pd.read_csv('iris.csv')
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Next steps: Generate code with df   New interactive sheet

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

X = df.iloc[:, :-1].values
y = LabelEncoder().fit_transform(df['species'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

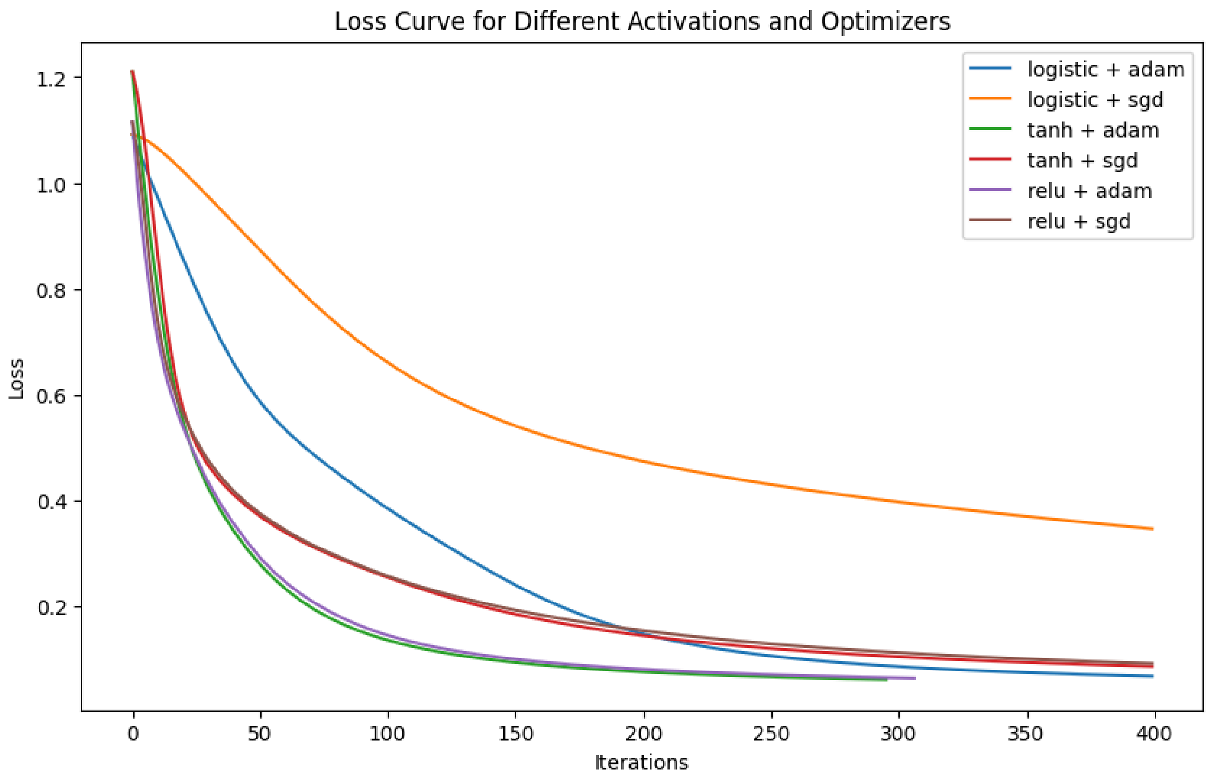
```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

activations = ['logistic', 'tanh', 'relu']
optimizers = ['adam', 'sgd']
results = {}

for act in activations:
    for opt in optimizers:
        mlp = MLPClassifier(hidden_layer_sizes=(5,), activation=act, solver=opt, max_iter=400, learning_rate_init=0.01, rando
        mlp.fit(X_train, y_train)
        y_pred = mlp.predict(X_test)
        acc = accuracy_score(y_test, y_pred)
        results[(act, opt)] = (mlp.loss_curve_, acc, mlp.coefs_)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic O
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic O
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic O
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic O
  warnings.warn(
```

```
for key, value in results.items():
    print(f"Activation: {key[0]}, Optimizer: {key[1]}, Accuracy: {value[1]:.4f}")
```

```
Activation: logistic, Optimizer: adam, Accuracy: 1.0000
Activation: logistic, Optimizer: sgd, Accuracy: 0.9333
Activation: tanh, Optimizer: adam, Accuracy: 1.0000
Activation: tanh, Optimizer: sgd, Accuracy: 1.0000
Activation: relu, Optimizer: adam, Accuracy: 1.0000
Activation: relu, Optimizer: sgd, Accuracy: 1.0000
```

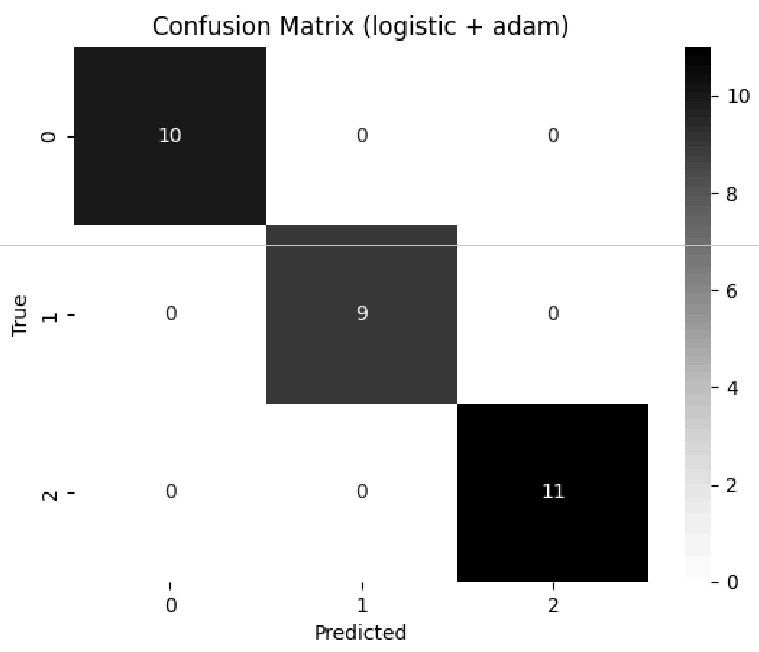
```
plt.figure(figsize=(10,6))
for key, value in results.items():
    plt.plot(value[0], label=f'{key[0]} + {key[1]}')
plt.title('Loss Curve for Different Activations and Optimizers')
plt.xlabel('Iterations')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



```
best_key = max(results, key=lambda k: results[k][1])
best_act, best_opt = best_key
print("Best Model:", best_act, "+", best_opt)

best_model = MLPClassifier(hidden_layer_sizes=(5,), activation=best_act, solver=best_opt, max_iter=400, learning_rate_init=0.
best_model.fit(X_train, y_train)
y_pred_best = best_model.predict(X_test)
print(classification_report(y_test, y_pred_best))
```

```
Best Model: logistic + adam
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

/usr/local/lib/python3.12/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic O
  warnings.warn(
```

```
import seaborn as sns
cm = confusion_matrix(y_test, y_pred_best)
sns.heatmap(cm, annot=True, fmt='d', cmap='Greys')
plt.title(f'Confusion Matrix ({best_act} + {best_opt})')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



Confusion Matrix (logistic + adam)
True
Predicted
0
1
2
10
9
11
8
6
4
2