```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
import numpy as np
#Load the Iris dataset
iris=datasets.load_iris()
x=iris.data[:,:2] #Consider only the first two features for simplicity
y=(iris.target!=0)*10 #Convert labels to binary (1 if not Iris-setosa,0 otherwise)
#split the dataset into training and testing sets
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
#Standardize features
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
#Initialize the Perceptron model
perceptron=Perceptron()
#Train the Perceptron model
perceptron.fit(x_train,y_train)
#Make predictions on the test set
y_pred=perceptron.predict(x_test)
#Evaluate accuracy
accuracy=accuracy_score(y_test,y_pred*100)
print("Accuracy:{accuracy*100.2f}%")
# Plot decision boundary
plt.figure(figsize=(10,6))
plt.scatter(x[:,0],x[:,1],c=y,cmap=plt.cm.Paired,edgecolors='black',marker='o')
plt.xlabel('Sepal Length (standardized)')
plt.ylabel('Sepal Width (standardized)')
plt.title('Perceptron Decision Boundary')
x_min,x_max=x[:,0].min()-1,x[:,0].max()+1
y_min,y_max=x[:,1].min()-1,x[:,1].max()+1
xx,yy=np.meshgrid(np.arange(x_min,x_max,0.02),np.arange(y_min,y_max,0.02))
z=perceptron.predict(np.c_[xx.ravel(),yy.ravel()])
z=z.reshape(xx.shape)
plt.contourf(xx,yy,z,alpha=0.3,cmap=plt.cm.Paired)
plt.show()
```

```
Accuracy:{accuracy*100.2f}%
```