```
# Load the Iris dataset
df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
```

```
import numpy as np
import pandas as pd
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
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 View recommended plots New interactive sheet

```
df.info()
```

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

```
df.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
df['species'].value_counts()
```

	count
species	
setosa	50
versicolor	50
virginica	50

dtype: int64

```
# Encode the species to numeric type
species_mapping = {'setosa': 0, 'versicolor': 1, 'virginica': 2}
df['species_encoded'] = df['species'].map(species_mapping)
```

```
# For a binary classification problem, select any two classes
X = df.iloc[:, :4][df['species_encoded'] < 2]
y = df['species_encoded'][df['species_encoded'] < 2]
```

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # Added random_state for reproducibility
```

Perceptron

https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.Perceptron.html#sklearn.linear_model.Perceptron

```
# Create a Perceptron model
perceptron = Perceptron()

# Train the model
perceptron.fit(X_train, y_train)

# Make predictions
y_pred = perceptron.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")


# Example of accessing the learned weights and bias (intercept)
print(f"Weights: {perceptron.coef_}")
print(f"Bias: {perceptron.intercept_}")
```

```
Accuracy: 1.0
Weights: [[-1.3 -4.5  6.8  3.1]]
Bias: [-1.]
```

Adaline

https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier

```
# Initialize and train the model with squared_loss (this behaves like ADALINE)
adaline = SGDClassifier(loss='squared_error', max_iter=1000, tol=1e-3, random_state=42)

adaline.fit(X_train, y_train)

# Make predictions
y_pred = adaline.predict(X_test)

# Evaluate the model
accuracy = adaline.score(X_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.4
```

Your work

- choose two species from the penguin dataset below
- repeat the steps above
- try another parameters and observe the result

Penguin dataset

https://raw.githubusercontent.com/mwaskom/seaborn-data/refs/heads/master/penguins.csv

**Finish the notebook1.ipynb, send to zwu009@citymail.cuny.edu by 5:00 pm Feb 6, 2025 along with your quiz answer. **

```
df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/refs/heads/master/penguins.csv')
```

Double-click (or enter) to edit

```
import numpy as np
import pandas as pd
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
df.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

```
df.describe()
```

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
count	342.000000	342.000000	342.000000	342.000000
mean	43.921930	17.151170	200.915205	4201.754386
std	5.459584	1.974793	14.061714	801.954536
min	32.100000	13.100000	172.000000	2700.000000
25%	39.225000	15.600000	190.000000	3550.000000
50%	44.450000	17.300000	197.000000	4050.000000
75%	48.500000	18.700000	213.000000	4750.000000
max	59.600000	21.500000	231.000000	6300.000000

```
df['species'].value_counts()
```

	count
species	
Adelie	152
Gentoo	124
Chinstrap	68

dtype: int64

```
species_mapping = {'Adelie': 0, 'Gentoo': 1, 'Chinstrap': 2}
df['species_encoded'] = df['species'].map(species_mapping)
df = df.dropna()
```

```
X = df.iloc[:, 2:6][df['species_encoded'] < 2]
y = df['species_encoded'][df['species_encoded'] < 2]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create a Perceptron model
perceptron = Perceptron()

# Train the model
perceptron.fit(X_train, y_train)

# Make predictions
y_pred = perceptron.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")


# Example of accessing the learned weights and bias (intercept)
print(f"Weights: {perceptron.coef_}")
print(f"Bias: {perceptron.intercept_}")
```

```
Accuracy: 0.37735849056603776
Weights: [[ -4229.5  -7298.2 -31017.    5325. ]]
Bias: [-265.]
```

```
# Initialize and train the model with squared_loss (this behaves like ADALINE)
adaline = SGDClassifier(loss='squared_error', max_iter=1000, tol=1e-3, random_state=42)

adaline.fit(X_train, y_train)

# Make predictions
y_pred = adaline.predict(X_test)

# Evaluate the model
accuracy = adaline.score(X_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.6226415094339622
```