LOGISTIC
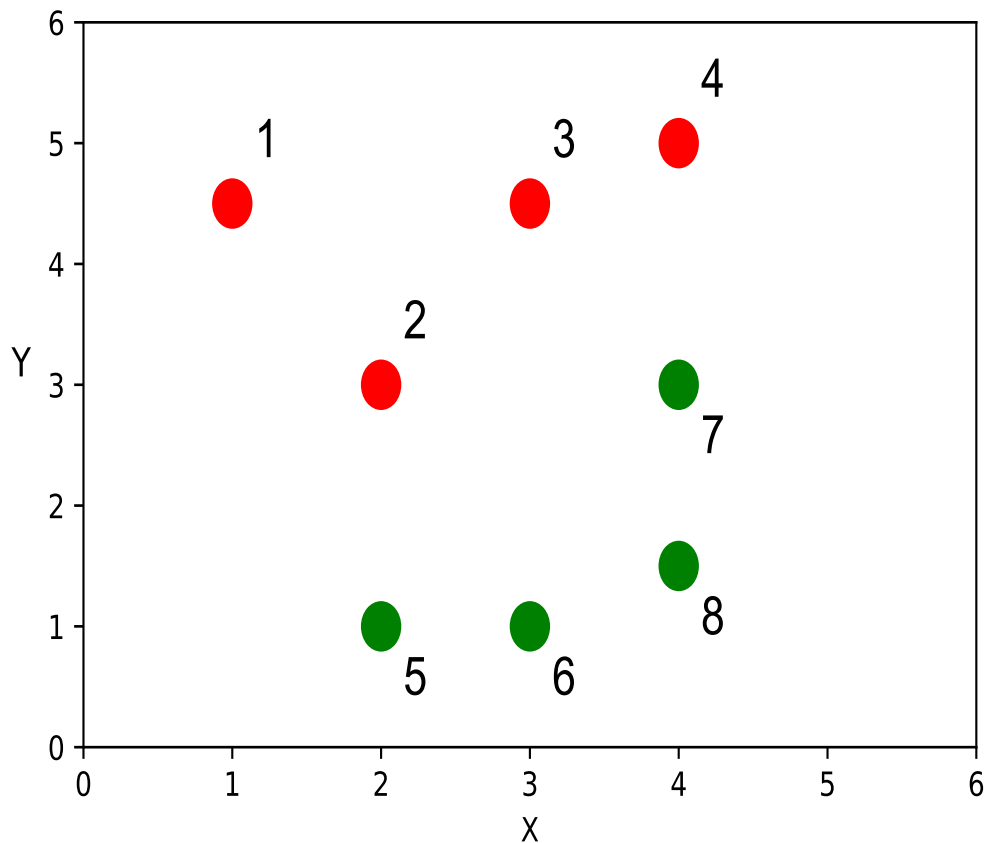
REGRESSION

Overview

- want to separate classes
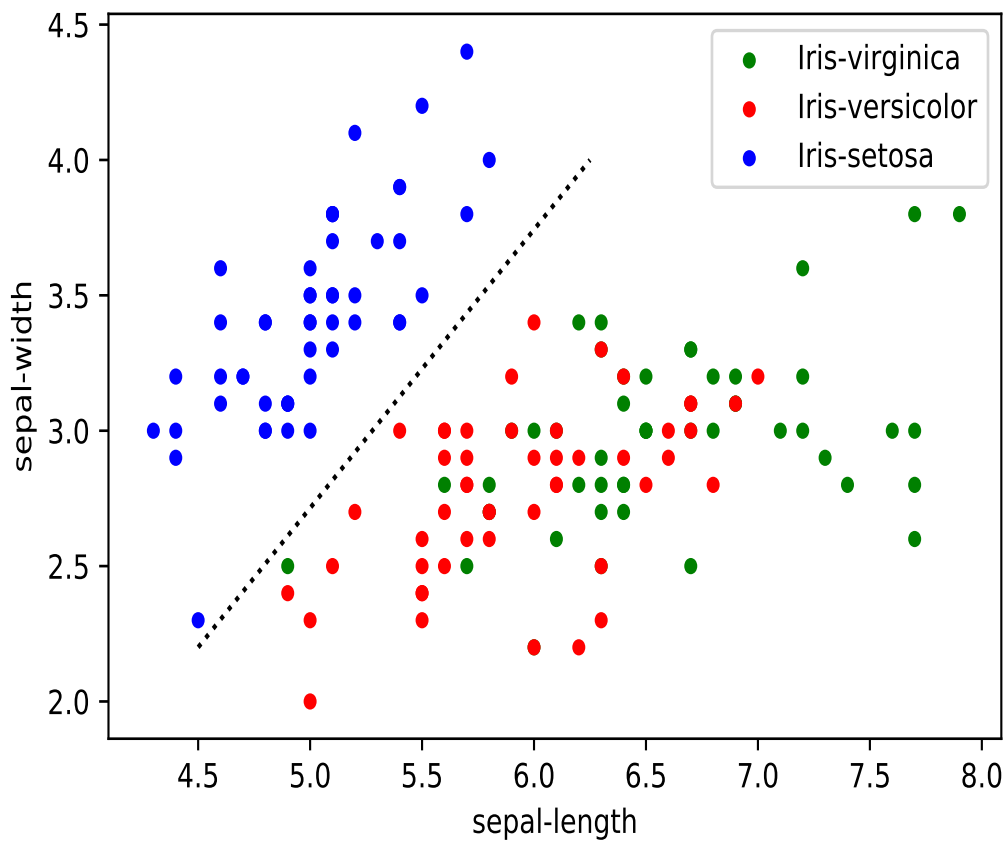
- smooth decision function

Binary Classification

- training set S with labels $\{-1, +1\}$
- find a classifier H

$$H : X \mapsto \{-1, +1\}$$

- low generalization error
- linear classification (based on logistic regression)

Linear Separability

- draw a hyperplane
- difficult in many cases

A Numerical Dataset

| object x_i	Height (H)	Weight (W)	Foot (F)	Label (L)
x_1	5.00	100	6	green
x_2	5.50	150	8	green
x_3	5.33	130	7	green
x_4	5.75	150	9	green
x_5	6.00	180	13	red
x_6	5.92	190	11	red
x_7	5.58	170	12	red
x_8	5.92	165	10	red

Code for the Dataset

```python
import pandas as pd
data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
         'Label': ['green','green','green','green',
                   'red','red','red','red'],
         'Height': [5, 5.5, 5.33, 5.75,
                    6.00, 5.92,  5.58, 5.92],
         'Weight': [100, 150, 130, 150,
                    180, 190, 170, 165],
         'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                    'Foot', 'Label'] )
```
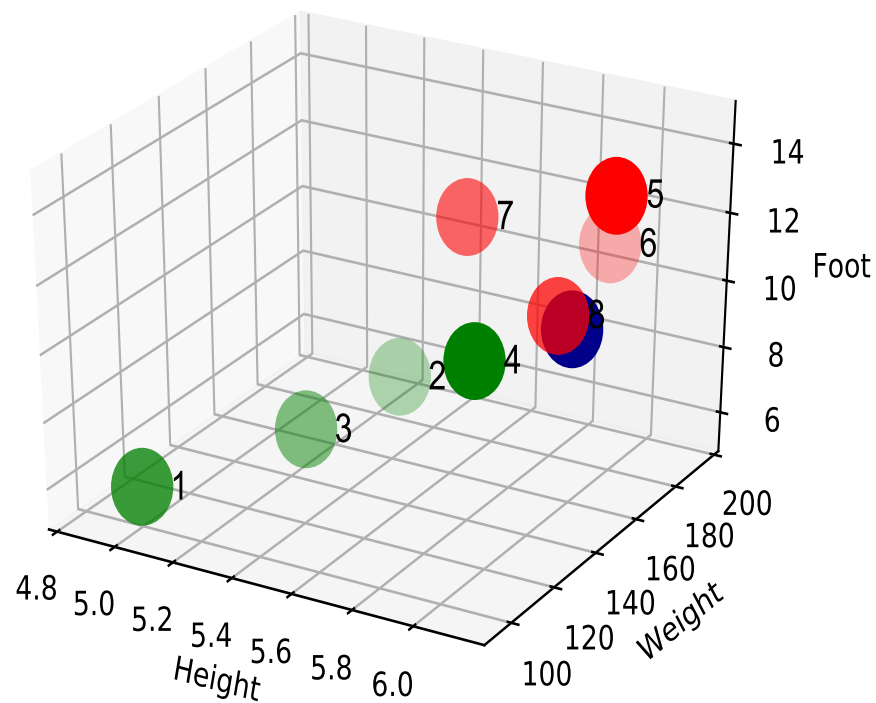
```
ipdb> data
   id Height Weight Foot Label
0  1  5.00     100     6  green
1  2  5.50     150     8  green
2  3  5.33     130     7  green
3  4  5.75     150     9  green
4  5  6.00     180    13    red
5  6  5.92     190    11    red
6  7  5.58     170    12    red
7  8  5.92     165    10    red
```
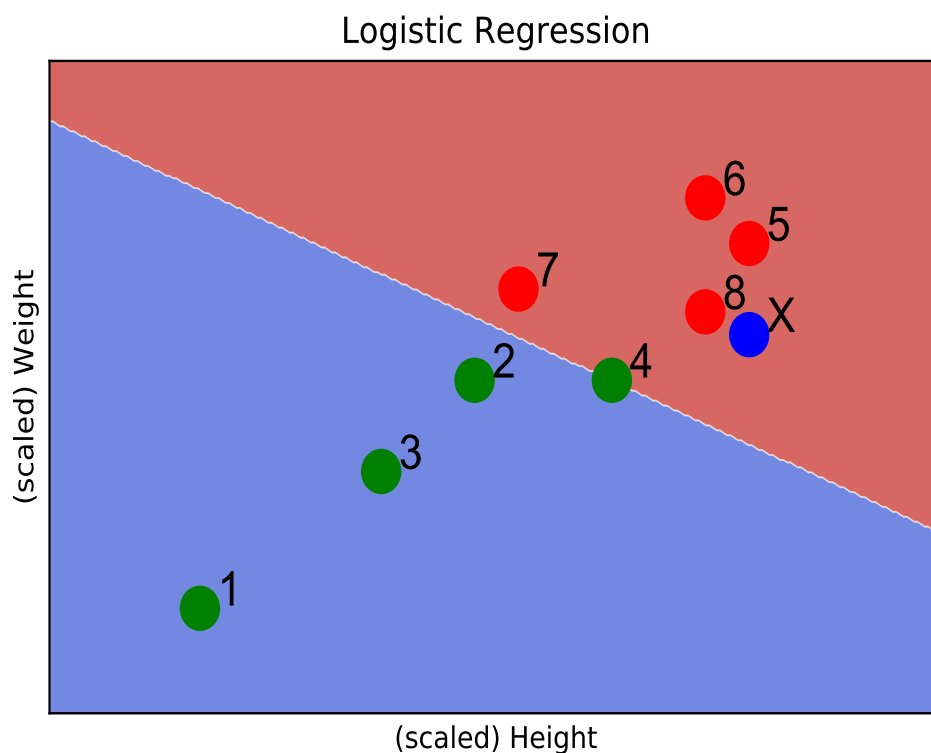
A Dataset Illustration
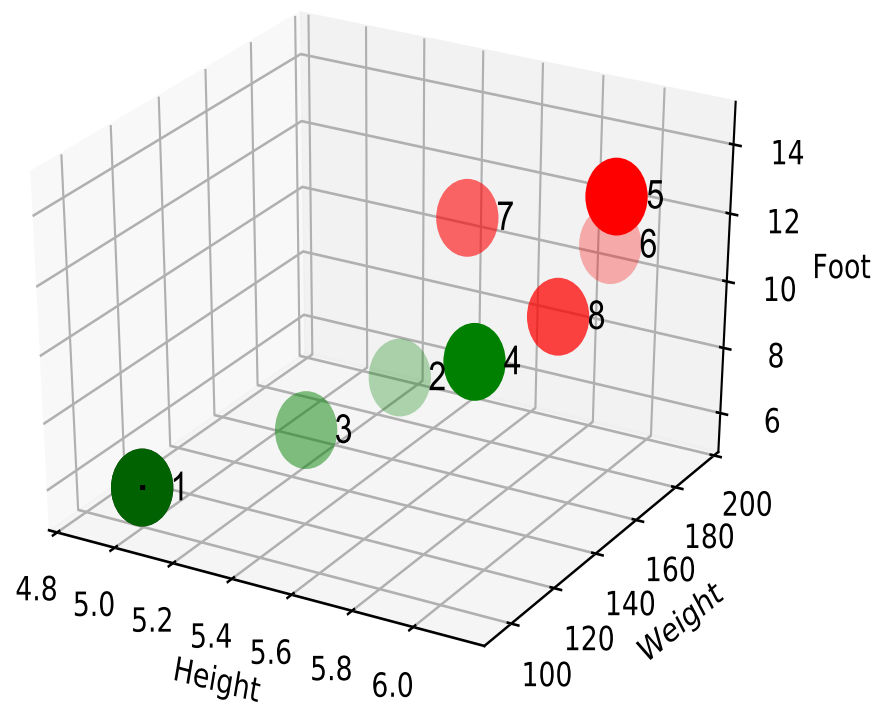
A New Instance

$$(\text{H=6, W=160, F=10}) \mapsto ?$$

Logistic Regression (original dataset)

Logistic Regression

- predict(x^*)=red
- accuracy $= 100\%$

Code: Log. Regression

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                        'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75, 6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150, 180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight', 'Foot', 'Label'] )

X = data[['Height', 'Weight']].values
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
Y = data['Label'].values

log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X,Y)

new_x = scaler.transform(np.asmatrix([6, 160]))
predicted = log_reg_classifier.predict(new_x)
accuracy = log_reg_classifier.score(X, Y)
```
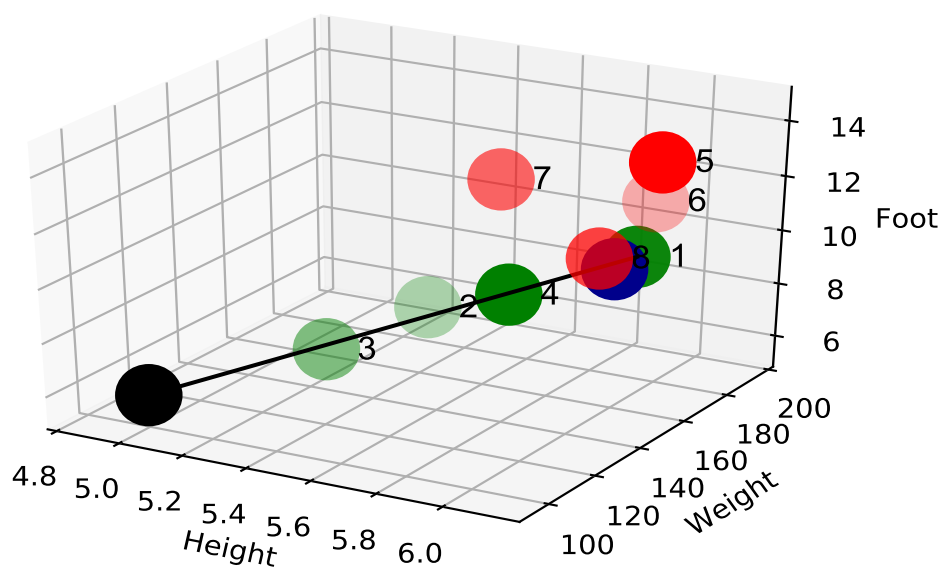
```
ipdb> predicted[0]
red
ipdb> accuracy
0.875
```

F/W/H Change

id	Height	Weight	Foot	Label
1	$5 \mapsto 6$	$100 \mapsto 170$	$6 \mapsto 10$	green

$$(\text{H}=6,\ \text{W}=160,\ \text{F}=10) \mapsto\ ?$$

Logistic Regression (modified dataset)
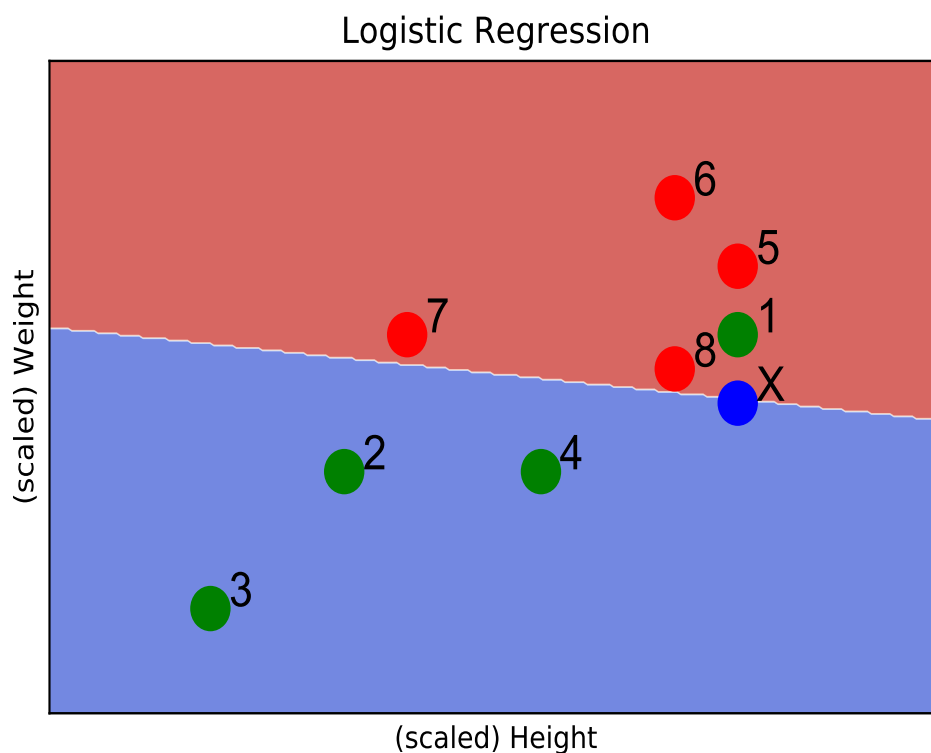
- predict(x^*)=green
- accuracy = 87.5%

Code: Log. Regression (modified dataset)

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, LabelEncoder

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                        'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75, 6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150, 180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight', 'Foot', 'Label'] )

data['Height'].iloc[0] = 6;
data['Weight'].iloc[0] = 170;
data['Foot'].iloc[0] = 10
X = data[['Height', 'Weight']].values
scaler = StandardScaler().fit(X)

X = scaler.transform(X)
Y = data['Label'].values
log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X,Y)
new_x = scaler.transform(np.asmatrix([6, 160]))
predicted = log_reg_classifier.predict(new_x)
accuracy = log_reg_classifier.score(X, Y)
```

ipdb> predicted[0]

green

ipdb> accuracy

0.875

Categorical Dataset

Day	Weather	Temperature	Wind	Play
1	sunny	hot	low	**no**
2	rainy	mild	high	**yes**
3	sunny	cold	low	**yes**
4	rainy	cold	high	**no**
5	sunny	cold	high	**yes**
6	overcast	mild	low	**yes**
7	sunny	hot	low	**yes**
8	overcast	hot	high	**yes**
9	rainy	hot	high	**no**
10	rainy	mild	low	**yes**

- $x^* = (\text{sunny, cold, low}) \mapsto ?$
- need numeric values for attributes

Change to Dummy Variables

Day	Weather			Temp.			Wind	
	overcast	rainy	sunny	cold	hot	mild	high	low
1	0	0	1	0	1	0	0	1
2	0	1	0	0	0	1	1	0
3	0	0	1	1	0	0	0	1
4	0	1	0	1	0	0	1	0
5	0	0	1	1	0	0	1	0
6	1	0	0	0	0	1	0	1
7	0	0	1	0	1	0	0	1
8	1	0	0	0	1	0	1	0
9	0	1	0	0	1	0	1	0
10	0	1	0	0	0	1	0	1

Python Code

```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
data = pd.DataFrame(
        {'Day':          [1,2,3,4,5,6,7,8,9,10],
         'Weather':      ['sunny','rainy','sunny','rainy',
                           'sunny','overcast','sunny','overcast',
                           'rainy','rainy'],
         'Temperature': ['hot', 'mild', 'cold','cold','cold',
                           'mild','hot','hot', 'hot','mild'],
         'Wind':         ['low','high','low','high','high',
                           'low','low', 'high','high','low'],
         'Play':         ['no', 'yes','yes','no','yes',
                           'yes','yes','yes','no','yes']},
         columns = ['Day','Weather','Temperature','Wind','Play'])
input_data = data[['Weather', 'Temperature', 'Wind']]
dummies = [pd.get_dummies(data[c]) for c in input_data.columns]
binary_data = pd.concat(dummies, axis=1)
X = binary_data[0:10].values
le = LabelEncoder()
Y = le.fit_transform(data['Play'].values)
log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X,Y)

# sunny -> (0,0,1), cold-> (0,1,0), low -> (0,1)
new_instance = np.asmatrix([0,0,1,1,0,0,0,1])
prediction = log_reg_classifier.predict(new_instance)
accuracy = log_reg_classifier.score(X, Y)
```
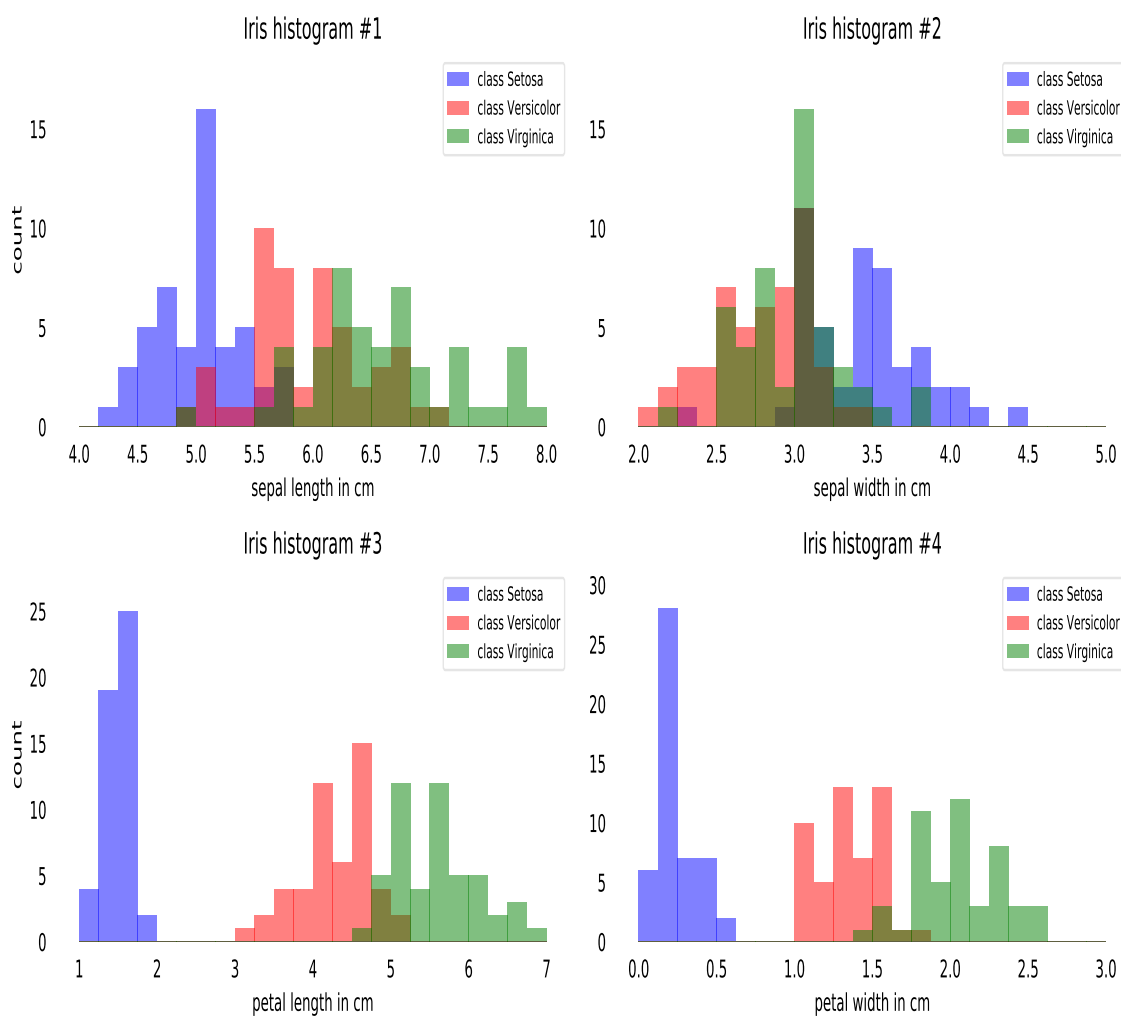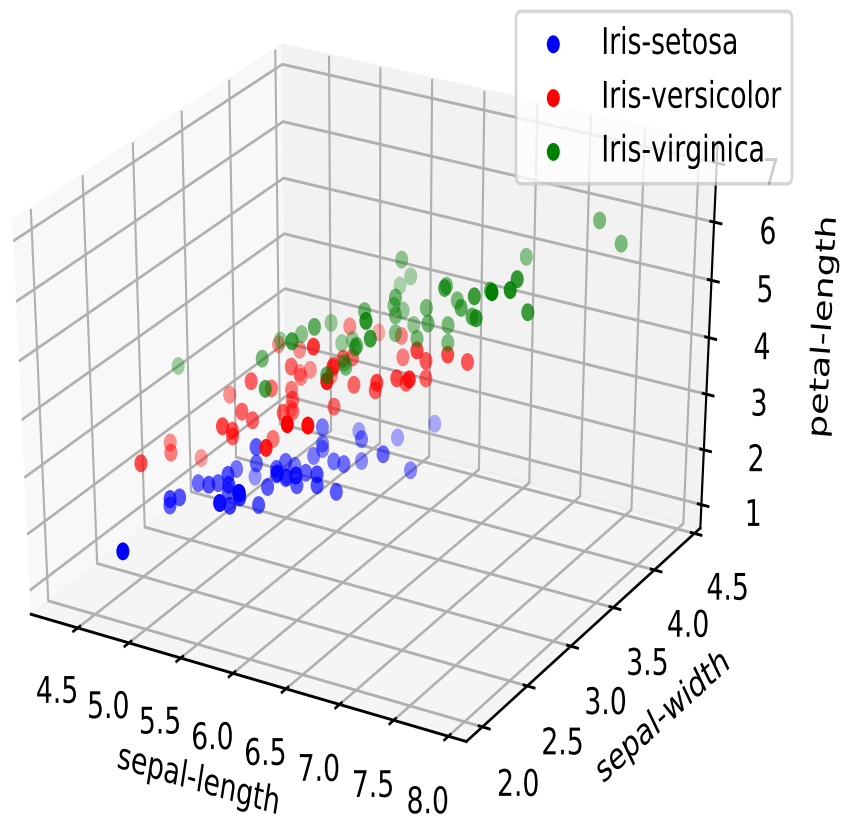
ipdb> prediction[0]

1

ipdb> accuracy

0.8

Iris Histograms

Iris Dataset:

Iris: Python Code

```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

url = r'https://archive.ics.uci.edu/ml/'  + \
         r'machine-learning-databases/iris/iris.data'

data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                      'petal-length', 'petal-width', 'Class'])

features = ['sepal-length', 'sepal-width']
class_labels = ['Iris-setosa', 'Iris-versicolor']

X = data[features].values

le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train,X_test,Y_train,Y_test = train_test_split(X, Y,
                          test_size=0.5, random_state=3)
log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X_train,Y_train)

prediction = log_reg_classifier.predict(X_test)
accuracy = np.mean(prediction == Y_test)


ipdb> accuracy
1.0
```
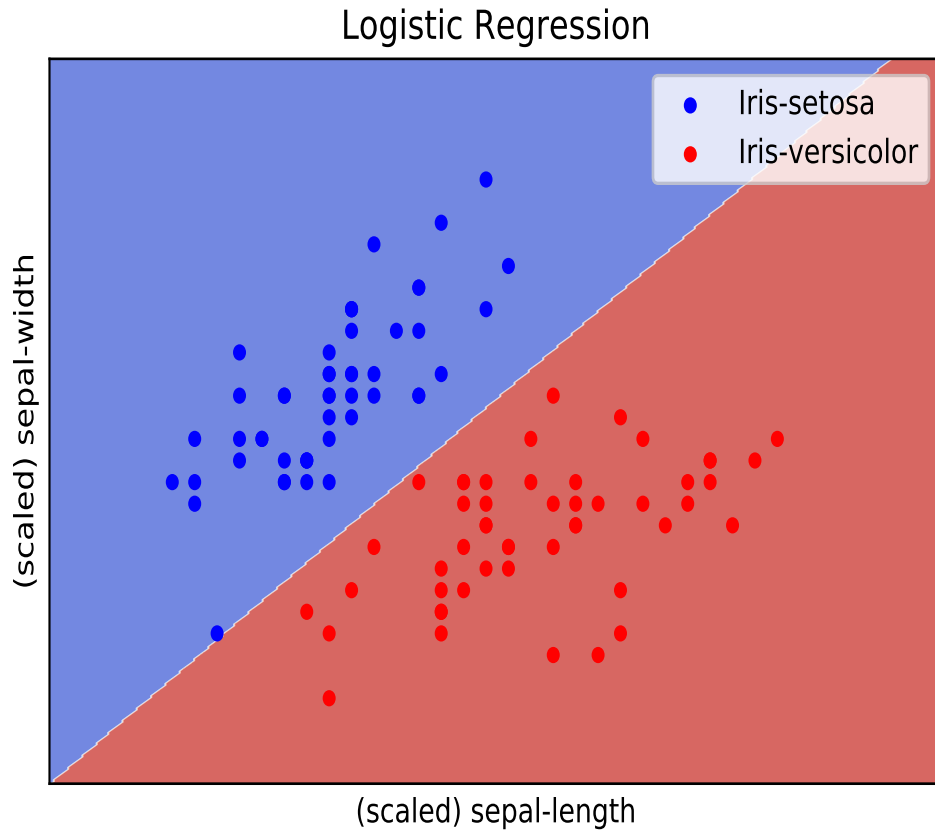
Iris: Logistic Regression

- accuracy = 100%

- easy to separate

Iris: Python Code

```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

url = r'https://archive.ics.uci.edu/ml/'  + \
          r'machine-learning-databases/iris/iris.data'

data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                      'petal-length', 'petal-width', 'Class'])

features = ['sepal-length', 'sepal-width']
class_labels = ['Iris-versicolor', 'Iris-virginica']

X = data[features].values

le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train,X_test,Y_train,Y_test = train_test_split(X, Y,
                        test_size=0.5, random_state=3)
log_reg_classifier = LogisticRegression()
log_reg_classifier.fit(X_train,Y_train)

prediction = log_reg_classifier.predict(X_test)
accuracy = np.mean(prediction == Y_test)


ipdb> accuracy
0.68
```
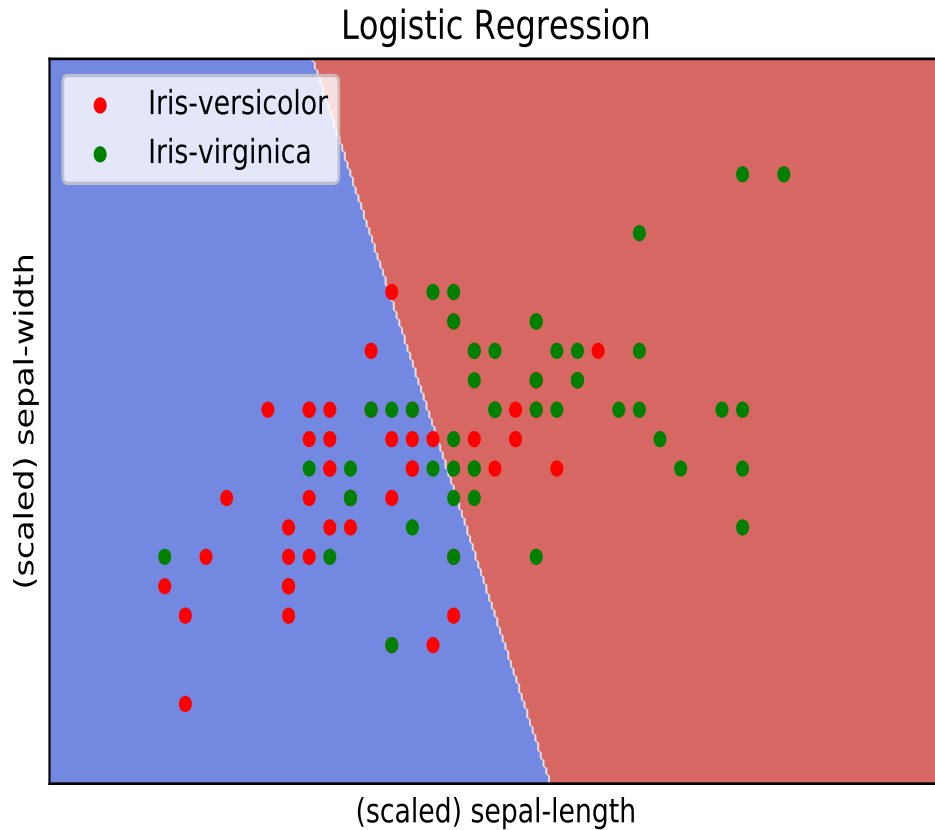
Iris: Logistic Regression

- accuracy = 68%

- difficult to separate