Machine Learning

| Logistic Regression - Implementation

Prof. Dr. Mostafa Elhosseini

https://youtube.com/ElhosseiniAcademy

Agenda

- Logistic Regression
- Log Loss
- Implementation
- Iris Dataset
- Pima Indians Diabetes Dataset

Model Equation

- Logistic Regression model computes a **weighted sum** of the input features (plus a bias term), but instead of outputting the result directly like the Linear Regression model does, it **outputs the logistic** of this result

- $\sigma(x) = \frac{1}{1+e^{-x}}$

- $\hat{y} = \sigma(b + \sum_{i=1}^{n} \theta_i x_i)$
- $\hat{y} = \sigma(b + \sum_{i=1}^{n} w_i x_i)$

- $\hat{y} = h_\theta(x) = \sigma(\theta^T . X)$
  - $\hat{y}$: probability that an instance x is belongs to a class
  - θ : model parameters that determine boundary decision
  - X: features
  - σ: logistic (sigmoid – Logit) S-shaped function

Log Loss

- log loss = $-y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$

- We use the term **log loss** when we refer to the **log loss of a point** or of a **whole dataset**.
- The log loss of a dataset is the sum of the log losses at every point.

- The objective of training is to set the parameter vector θ so that the model estimates high probabilities for positive instances ($y = 1$) and low probabilities for negative instances ($y = 0$).

Training and cost function

- The cost function over the whole training set

$$L = -\frac{1}{M}\sum_{i=1}^{M}\left[y^{(i)}\ln(\hat{y}^{(i)}) + (1-y^{(i)})\ln(1-\hat{y}^{(i)})\right]$$

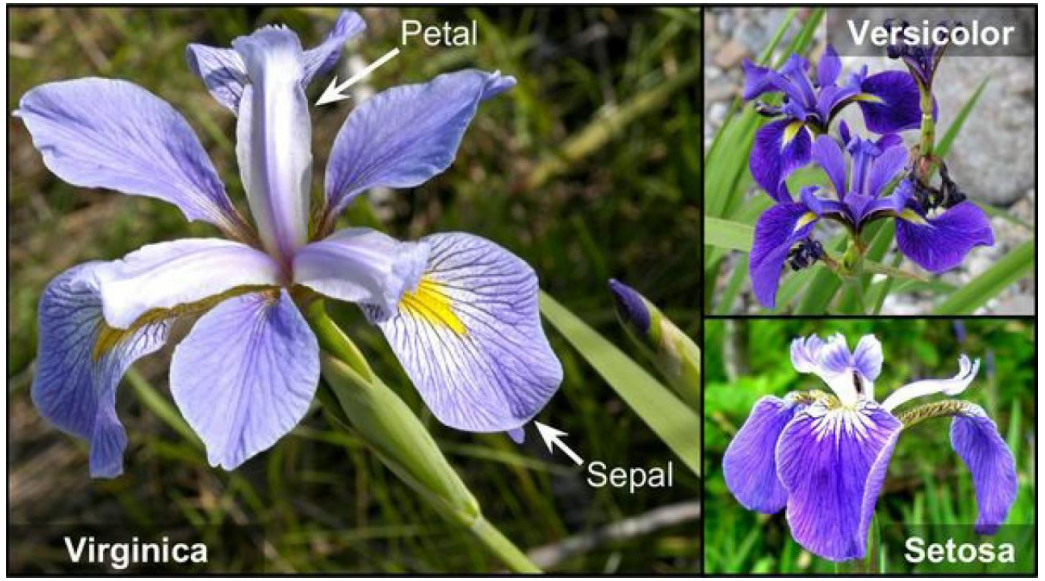
- To update the weights $\boldsymbol{w}$ to minimize the log loss, we adjust $\boldsymbol{w}$ in the opposite direction of the gradient of the loss:
- $\boldsymbol{w} \longleftarrow \boldsymbol{w} + \eta(y - \hat{y})\boldsymbol{x}$

IRIS dataset

- The Iris dataset is one of the most well-known datasets in the field of machine learning and statistics, often used as a benchmark for testing machine learning algorithms. It was introduced by the British statistician and biologist Ronald Fisher in 1936 as an example of discriminant analysis.

IRIS dataset

- Dataset that contains the **sepal** (الكأس) and **petal** (التويج) length and width of 150 iris flowers of three different species: Iris-**Setosa**, Iris-**Versicolor**, and Iris-**Virginica**
  - 4 numeric attributes
  - 50 instance per each class
  - One class is linearly separable from the other two



Python session

- Loading Dataset

```
from sklearn.datasets import load_iris

iris = load_iris(as_frame=True)
list(iris)
```

```
['data',
 'target',
 'frame',
 'target_names',
 'DESCR',
 'feature_names',
 'filename',
 'data_module']
```

Python session

- Dataset Description

```
print(iris.DESCR)
```

```
Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
```

```
:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988
```

Python session

- Summary Statistics

```
print(iris.DESCR)
```

```
============== ==== ==== ======= ===== ====================
                Min  Max   Mean    SD   Class Correlation
============== ==== ==== ======= ===== ====================
sepal length:   4.3  7.9   5.84   0.83    0.7826
sepal width:    2.0  4.4   3.05   0.43   -0.4194
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
============== ==== ==== ======= ===== ====================
```

Python session

- Quick Peek:

```
iris.data.head(3)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2

- the instances are not shuffled

```
iris.target.head(3)
```

```
0    0
1    0
2    0
Name: target, dtype: int64
```

Python session

- Features' Names:

```
iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

- Label values

```
iris.target
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int64
```

```
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

Python session

- Filter petal width and label.

```
X = iris.data[["petal width (cm)"]].values
y = iris.target_names[iris.target] == 'virginica'
```

```
X_y = np.column_stack((X, y))
```

```
array([[0.2, 0. ],
       [0.2, 0. ],
       [0.2, 0. ],
       [0.2, 0. ],
       [0.2, 0. ],
       [0.4, 0. ],
       [0.3, 0. ],
       [0.2, 0. ],
       [0.2, 0. ],
       [0.1, 0. ],
       [0.2, 0. ],
```

- Split Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

Python session

- Fitting Model

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
```

- Model’s estimated probabilities

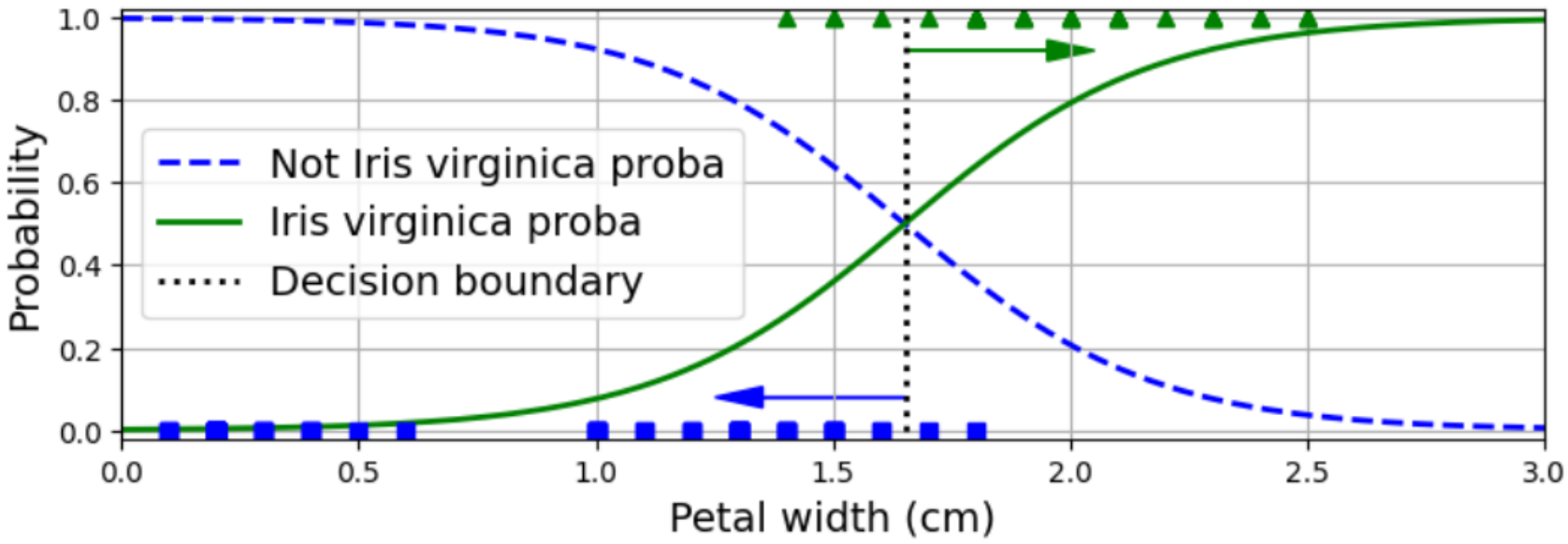
```
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
```

```
decision_boundary = X_new[y_proba[:, 1] >= 0.5][0, 0]
```

```
plt.plot(X_new, y_proba[:, 0], "b--", linewidth=2,
         label="Not Iris virginica proba")
plt.plot(X_new, y_proba[:, 1], "g-", linewidth=2, label="Iris virginica proba")
plt.plot([decision_boundary, decision_boundary], [0, 1], "k:", linewidth=2,
         label="Decision boundary")
```

Python session

- look at the model's estimated probabilities for flowers with petal widths varying from 0 to 3 cm

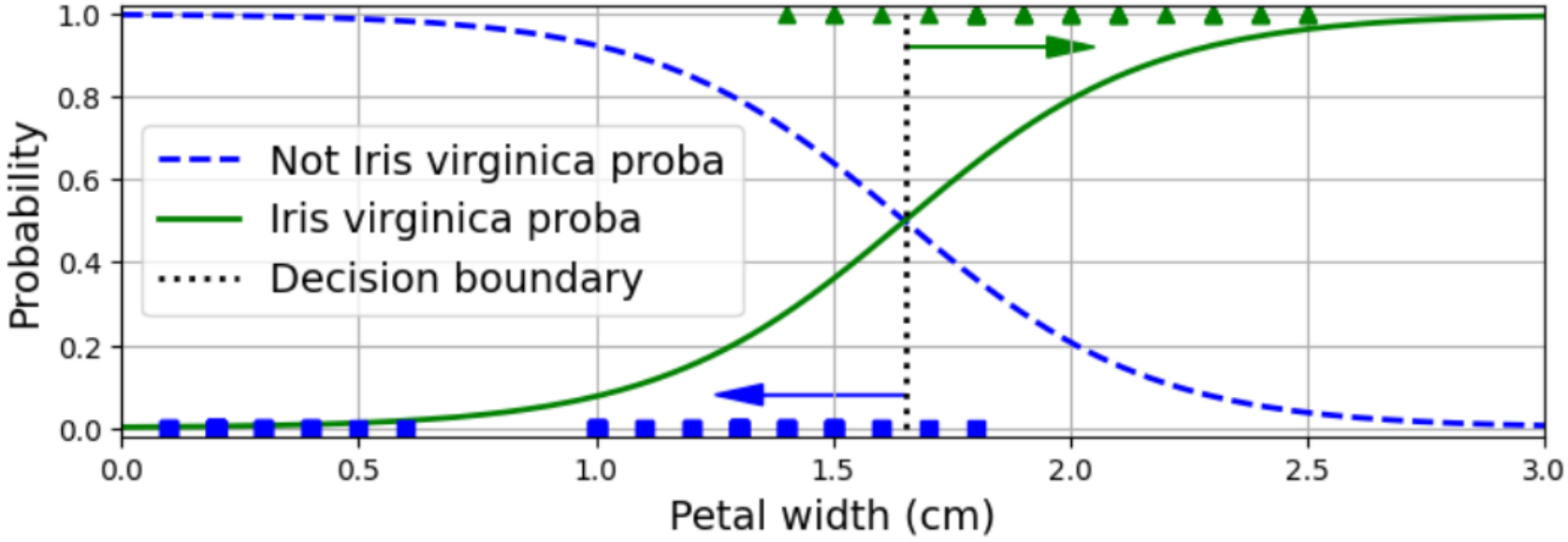



```
decision_boundary
```

```
1.6516516516516517
```

Python session

- The petal width of Iris-Virginica flowers (represented by triangles) ranges from 1.4 cm to 2.5 cm,
- while the other iris flowers (represented by squares) generally have a smaller petal width, ranging from 0.1 cm to 1.8 cm

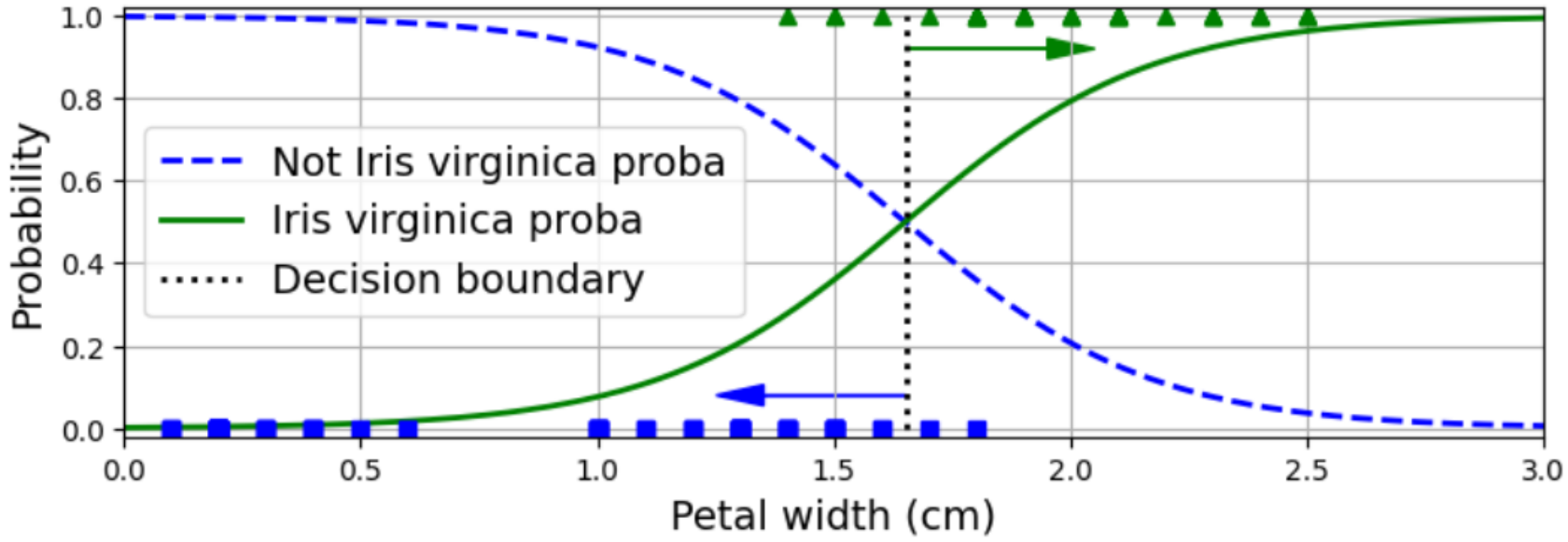



```
decision_boundary
```

```
1.6516516516516517
```

Python session

- There is a bit of overlap. Above about 2 cm the classifier is highly confident that the flower is an Iris-Virginica (it outputs a high probability to that class), while below 1 cm it is highly confident that it is not an Iris-Virginica (high probability for the “Not Iris-Virginica” class)

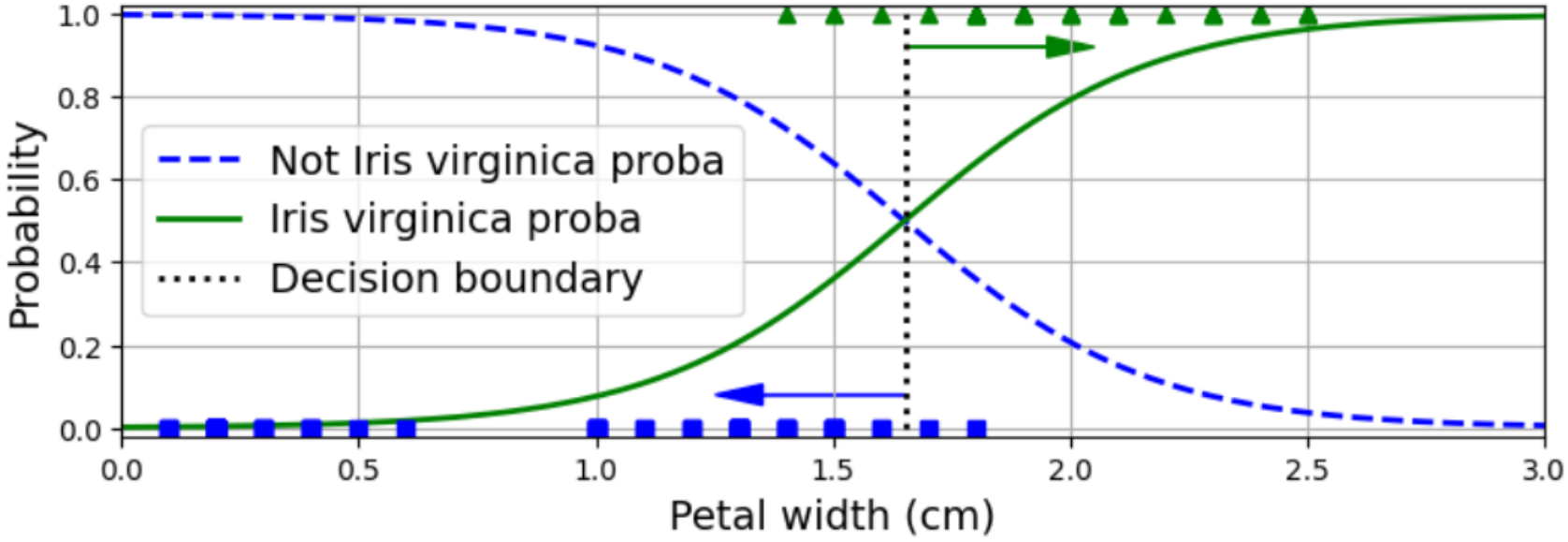



```
decision_boundary
```

```
1.6516516516516517
```

Python session

- if you ask it to predict the class (using the **predict()** method rather than the **predict_proba()**), it will return whichever class is the **most likely**.
- if the petal width is higher than 1.6 cm, the classifier will predict that the flower is an Iris-Virginica, or else it will predict that it is not (even if it is not very confident):

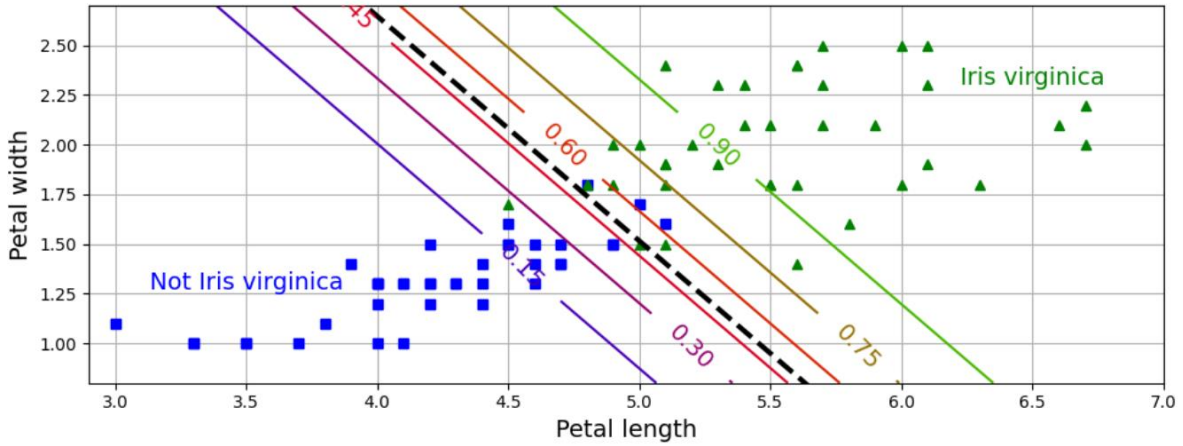



```
decision_boundary
```

```
1.6516516516516517
```

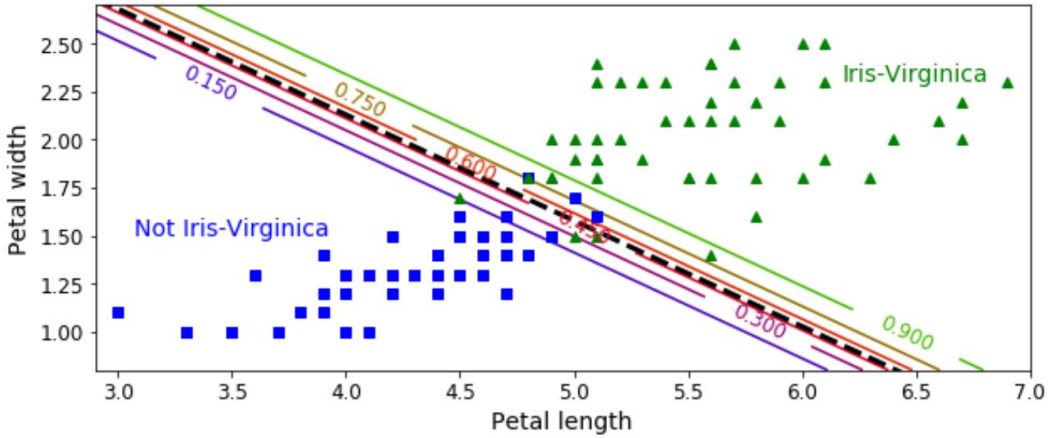
Python session

- Logistic Regression classifier can estimate the probability that a new flower is an Iris Virginica based on **petal width and length**.
  - The **dashed line** represents the points where the model estimates a **50% probability**: this is the model's **decision boundary // linear boundary**
  - Each parallel line represents the points where the model outputs a specific probability, from 15% (bottom left) to 90% (top right).



Python session

- All the flowers beyond the top-right line have an over 90% chance of being Iris-Virginica according to the model
- Logistic Regression models can be regularized using ℓ1 or ℓ2 penalties. Scitkit-Learn actually adds an ℓ2 penalty by default
- The hyperparameter controlling the regularization strength of a Scikit-Learn Logistic Regression model is not alpha (as in other linear models), but its inverse: **C**. The higher the value of C, the less the model is regularized.



Pima Indians Diabetes Database

- Predict the onset “ بداية “of diabetes based on diagnostic measures
- This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.
- The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.
  - Several constraints were placed on the selection of these instances
  - In particular, all patients here are females at least 21 years old of Pima Indian heritage

Pima Indians Diabetes Database

- The datasets consists of several medical predictor variables and one target variable.
- Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Pima Indians Diabetes Database

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver="liblinear", random_state=42)
log_reg.fit(X_train, y_train)
```

```
y_pred = log_reg.predict(X_test)
```

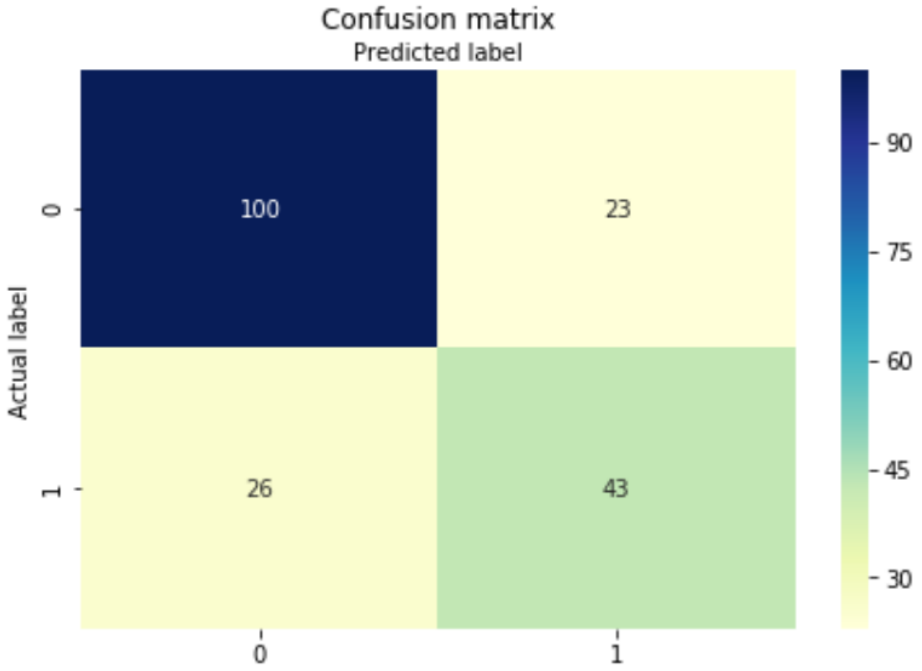
```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test,y_pred)
cnf_matrix
```

```
array([[100,  23],
       [ 26,  43]], dtype=int64)
```

Pima Indians Diabetes Database

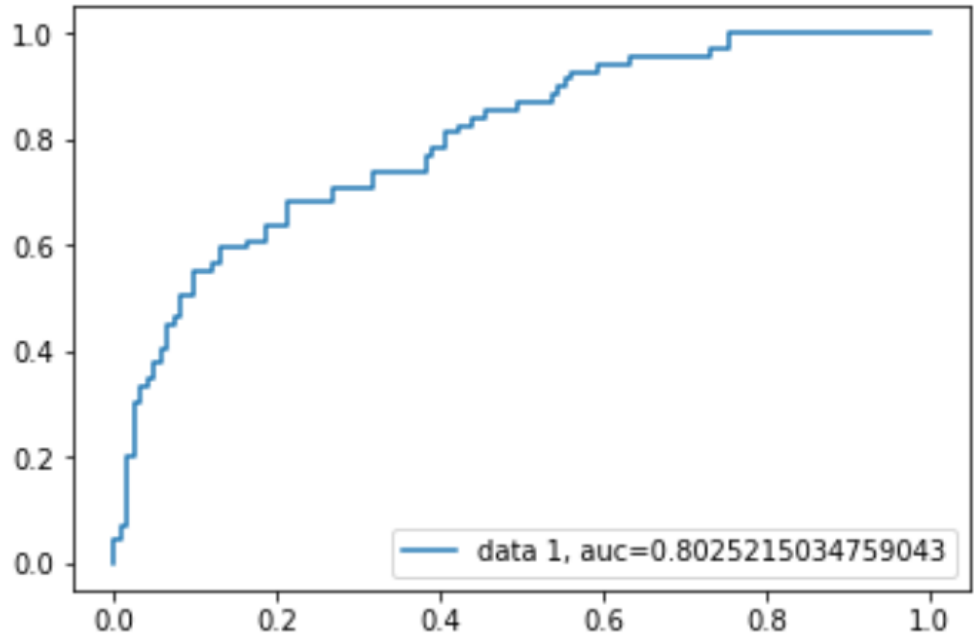
```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.7447916666666666
Precision: 0.6515151515151515
Recall: 0.6231884057971014
```



Pima Indians Diabetes Database

```
y_pred_proba = log_reg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```



Pima Indians Diabetes Database

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.Text(0.5,257.44,'Predicted label')
```

Softmax Regression

- Multinomial (multiclass) logistic regression
- Generalization of logistic regression that can handle multiple classes, as opposed to just binary classification.
- Given an instance represented by a feature vector x, **softmax** regression outputs the **probabilities that** x **belongs** to each of several classes.
- The model calculates these probabilities by applying the softmax function to the linear scores of each class calculated from x.
- Next Lecture