LOGISTIC REGRESSION

MADALINA, SHRUTHI, HANNAH & REBECCA

AGENDA

1. WHAT IS LOGISTIC REGRESSION
2. ASSUMPTIONS
3. SIGMOID FUNCTION
4. CUT OFF VALUE
5. CONFUSION MATRIX EVALUATION
6. PYTHON EXAMPLES

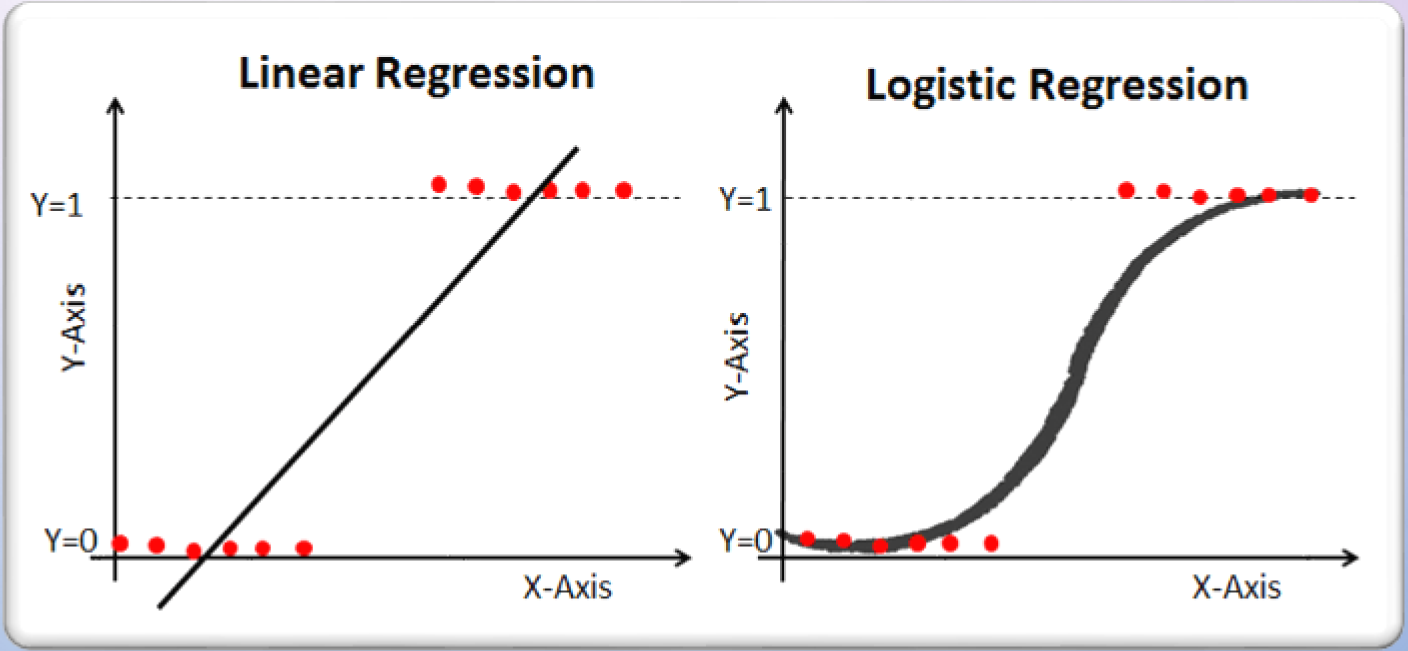
WHAT IS LOGISTIC REGRESSION

LOGISTIC REGRESSION IS A **SUPERVISED MACHINE LEARNING ALGORITHM** USED TO PREDICT CATEGORICAL OUTCOMES ON THE BASIS OF SEVERAL CONTINUOUS OR CATEGORICAL EVENTS.

- **SIGMUID** "S' SHAPED LOGISTIC FUNCTION

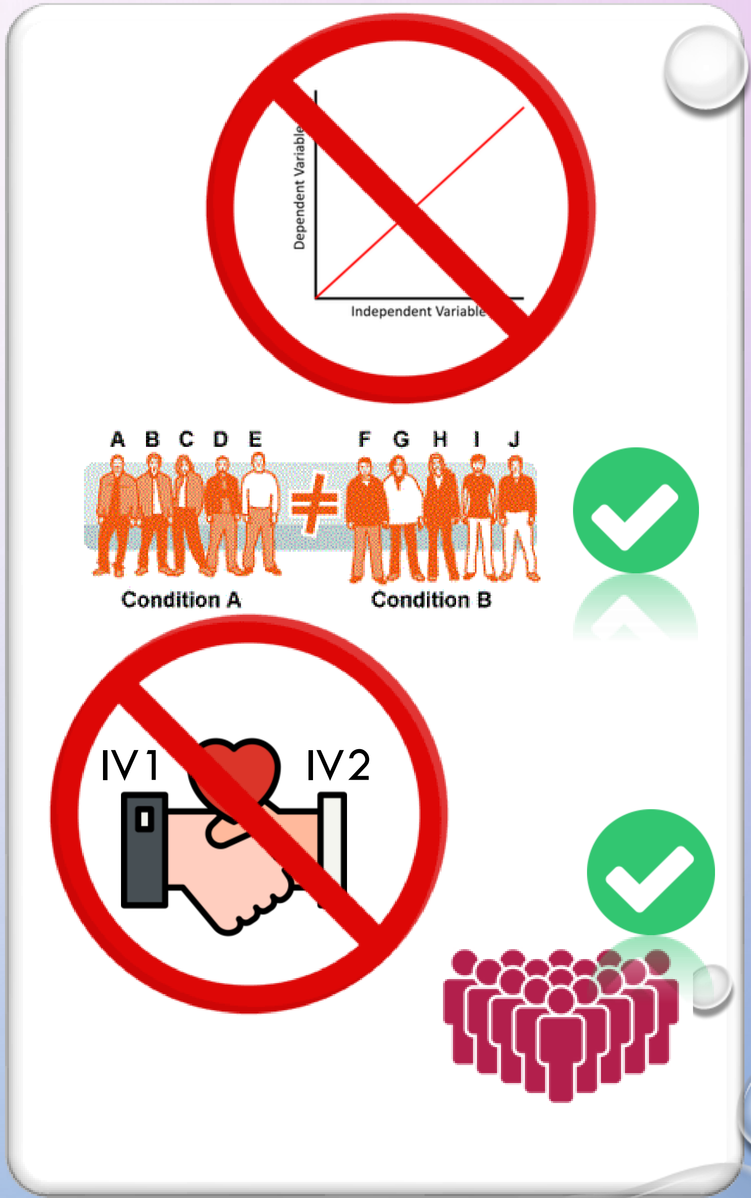
- THE OUTCOME IS **CATEGORICAL - BINARY** (E.G., **YES/NO, 0/1**).

- THE GOAL IS TO ESTIMATE THE **PROBABILITY OF A CLASS**



ASSUMPTIONS

- **NO LINEAR RELATIONSHIP** BETWEEN VARIABLES
- ERROR TERMS (RESIDUALS) **DON'T NEED TO BE NORMALLY DISTRIBUTED**
- BINARY LOGISTIC REGRESSION REQUIRES DV TO BE BINARY
- ORDINAL LOGISTIC REGRESSION REQUIRES DV TO BE ORDINAL
- **OBSERVATIONS SHOULD BE INDEPENDENT**
    - NO REPEATED MEASURES OR MATCHED DATA
- LITTLE TO **NO MULTICOLLINEARITY**
    - IVS CANNOT BE TOO STRONGLY RELATED WITH EACH OTHER
- **IVS ARE LINEARLY RELATED TO LOG ODDS**
- USUALLY REQUIRES **LARGE SAMPLE SIZE**



SIGMOID (LOGISTIC) FUNCTION

(BEHIND THE SCENES)

THE **SIGMOID FUNCTION** IS A MATHEMATICAL FUNCTION THAT MAPS ANY REAL-VALUED NUMBER, X TO A VALUE **BETWEEN 0** AND **1.**

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

THE FUNCTION TURNS A LINEAR COMBINATION OF INPUTS (WHICH CAN BE ANY NUMBER FROM −∞ TO +∞) INTO SOMETHING **BOUNDED BETWEEN 0 AND 1**, PERFECT FOR PROBABILITY MODELLING.

- **X** - ANY **REAL NUMBER** (POSITIVE/NEGATIVE)
- **e** – **EULER'S NUMBER** (~2.718)
- **Σ(X) -** GIVES A NUMBER **BETWEEN 0 AND 1**

SIGMOID (LOGISTIC) FUNCTION

(BEHIND THE SCENES)

LOGISTIC REGRESSION MODEL LEARNS A **LINEAR EQUATION** , AND FINDS THE COEFFICIENTS B0, B1, …, BN.

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

THEN IT APPLIES THE **SIGMOID FUNCTION Σ(Z)** TO TURN THE RESULT INTO A PROBABILITY.

IT COMPARES PREDICTIONS TO ACTUAL LABELS AND COMPUTES A LOSS FUNCTION TO MINIMISE TOTAL LOSS.

AND IT USES **MAXIMUM LIKELIHOOD ESTIMATOR** TO IMPROVE COEFFICIENTS.

SO, THE NEXT TIME YOU CALL **MODEL.FIT(X,Y)**

```
model.fit(X, y)
```

- **YOU'RE ASKING SCIKIT-LEARN LOGISTIC MODEL TO LEARN FROM YOUR DATA**
- YOU'VE JUST TRAINED A MACHINE LEARNING MODEL !

CUTOFF VALUE

- IN LOGISTIC REGRESSION, THE CUTOFF VALUE (ALSO CALLED THE DECISION THRESHOLD) IS THE PROBABILITY THRESHOLD USED TO CONVERT THE PREDICTED PROBABILITIES INTO BINARY CLASS LABELS

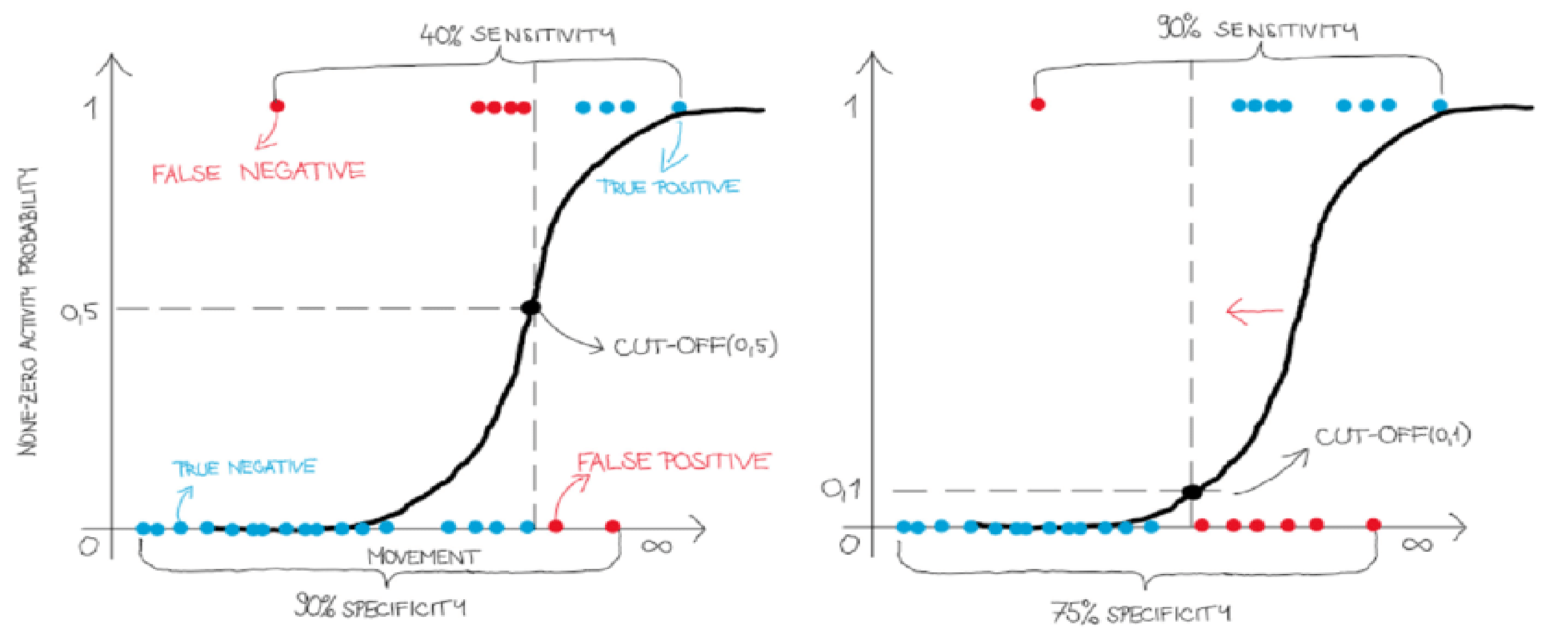
- DEFAULT CUTOFF VALUE IS 0.5, BUT THIS CAN BE ADJUSTED
- IF THE **PREDICTED PROBABILITY ≥ 0.5**, THEN PREDICT **POSITIVE CLASS**
- IF THE **PREDICTED PROBABILITY < 0.5**, THEN PREDICT **NEGATIVE CLASS**

- WHEN THE PREDICTED CLASS MATCHES THE ACTUAL CLASS, THESE ARE CALLED **TRUE POSITIVES OR TRUE NEGATIVES (TP/TN)**
- OTHERWISE, THEY ARE **FALSE POSITIVES OR FALSE NEGATIVES (FP/FN)**
- ONCE THESE VALUES ARE COUNTED, THEY CAN BE REPRESENTED IN A CONFUSION MATRIX
- CALCULATIONS USING THESE VALUES CAN EVALUATE THE MODEL'S PERFORMANCE
- DEFAULT VALUE IS ADJUSTED ACCORDING TO THE COST OF THE **TYPE OF ERROR** (EITHER FP OR FN)

FOR EXAMPLE:

- IN FRAUD DETECTION, YOU MAY SET THE CUTOFF TO 0.3 TO CATCH MORE FRAUD (REDUCE FALSE NEGATIVES)
- IN EMAIL SPAM DETECTION, YOU MIGHT RAISE IT TO 0.7 TO AVOID MARKING REAL EMAILS AS SPAM (REDUCE FALSE POSITIVES)

- **SENSITIVITY (RECALL / TRUE POSITIVE RATE)**
- DEFINITION: THE PROPORTION OF ACTUAL POSITIVES THAT ARE CORRECTLY IDENTIFIED.
- SENSITIVITY=TP/(TP+FN)
- **SPECIFICITY (TRUE NEGATIVE RATE)**
- DEFINITION: THE PROPORTION OF ACTUAL NEGATIVES THAT ARE CORRECTLY IDENTIFIED.
- SPECIFICITY= TN/(TN+FP)



CONFUSION MATRIX

TOTAL =200	**Predicted(no)**	**Predicted(Yes)**	
Actual (no)	70 (True Negative)	10 (False Positive)	80
Actual (yes)	20 (False Negative)	100 (True Positive)	120
	90	110	

- **True Positive** (TP)
- **True Negative** (TN)
- **False Positive** (FP) → Type 1 error
- **False Negative** (FN) → Type 2 error

Accuracy rate (TP + TN)/TOTAL ----- Here accuracy rate of model is (100+70)/200 = 0.85

Misclassification rate (FP + FN)/TOTAL ----- Here misclassification rate of model is (10+20)/200 = 0.15

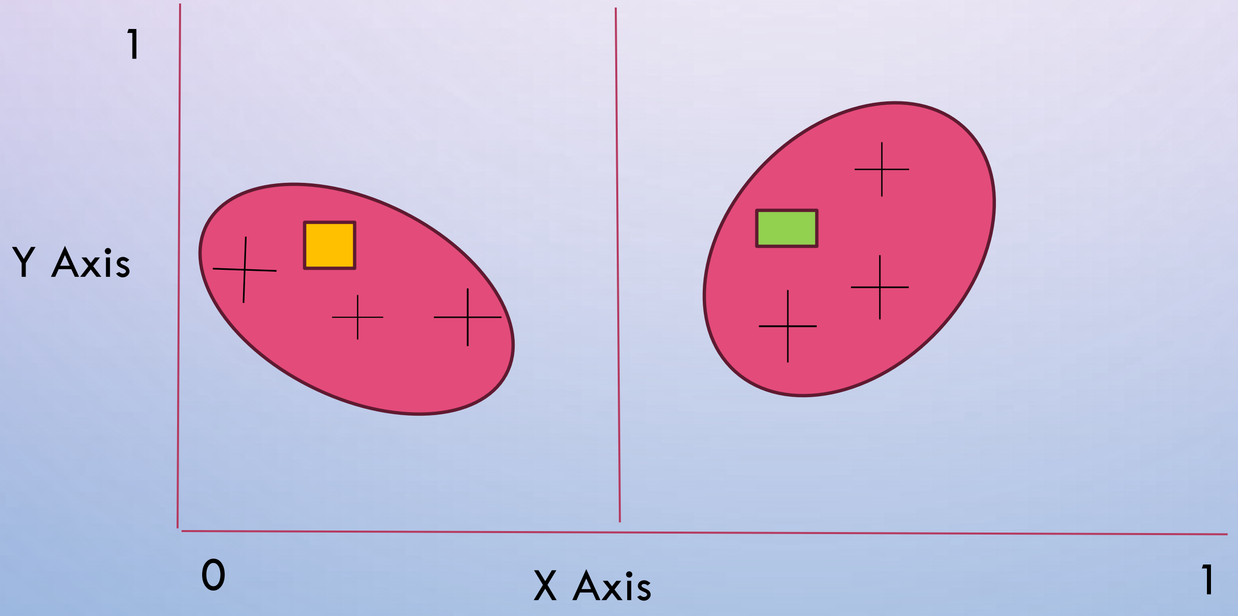
EXAMPLES + USAGE

APPLE OR ORANGE?

Examples	Description/Usage
Disease Diagnosis	Predicting if someone is diagnosed for a disease.
Spam Detection	Detecting if an email is spam or not
Fraud Detection	Detecting if a transaction is fraudulent
Credit Scoring	If someone is a high risk for a credit score
Classifying Animals	Detecting if an animal is a cat or a dog
Classifying Fruits	Classifying an apple from an orange
Click-through Rate	If a user will click on an ad or not
Churn Prediction	If a customer will leave a service or not

LOGISTIC REGRESSION IN PYTHON

Graph to show the classification such as categorising apples and oranges. It has to be binary classification, meaning that it has to be separated into two categories. If it is more than two categories it should probably be clustered, maybe using K-means clustering.

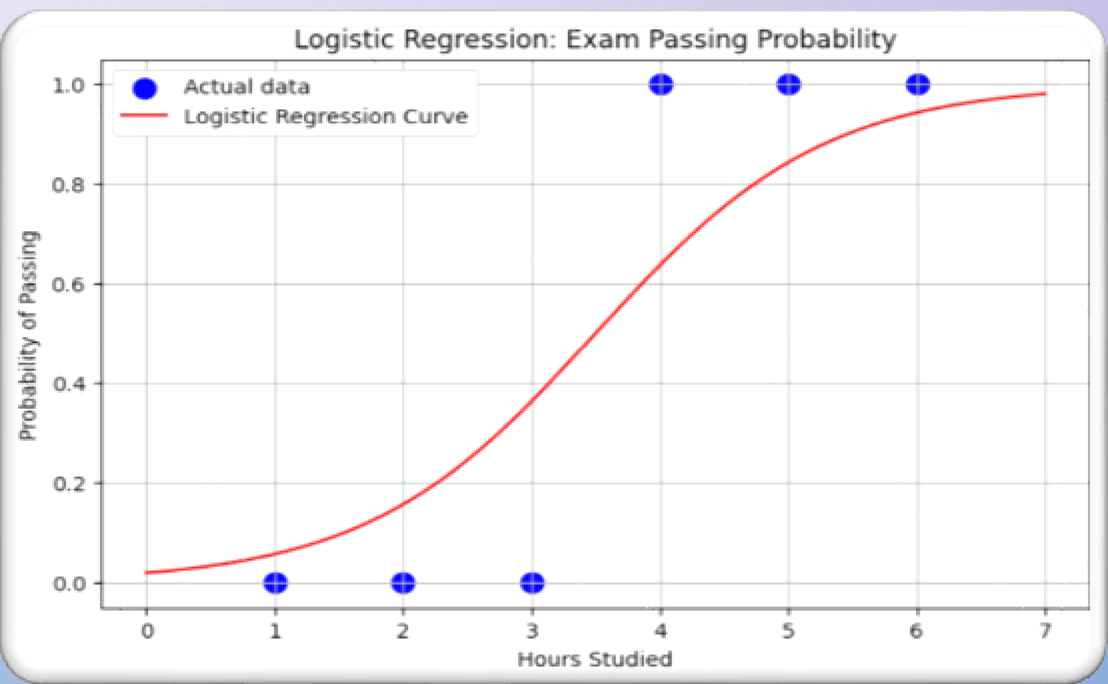


EXAM PASSING PROBABILITY - LOGISTIC REGRESSION

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

```python
# 1. Sample dataset
data = {'hours_studied': [1,2,3,4,5,6],
        'passed': ['No','No','No','Yes','Yes','Yes']}

df = pd.DataFrame(data)
df

# Convert Y, DEPENDENT target/ output/ variable to categorical variable i.e binary Yes/No
df['passed'] = df['passed'].map({'No':0,'Yes':1})

# Train logistic regression model
X = df[['hours_studied']]
y = df['passed']

# LogisticRegression() - creates a logistic regression model object from scikit learn
model = LogisticRegression()
model.fit(X,y)

# Make predictions over a smooth range
x_vals = np.linspace(0, 7, 100).reshape(-1, 1)
y_probs = model.predict_proba(x_vals)[:, 1]
```

```python
# PLOT THE FIGURE
plt.figure(figsize=(8, 5))
plt.scatter(df['hours_studied'], df['passed'], color='blue', s=100, label='Actual data')
plt.plot(x_vals, y_probs, color='red', label='Logistic Regression Curve')
plt.xlabel('Hours Studied')
plt.ylabel('Probability of Passing')
plt.title('Logistic Regression: Exam Passing Probability')
plt.legend()
plt.grid(True)
plt.show()
```



FLOWER SPECIES LOGISTIC REGRESSION IN PYTHON

Importing data

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder

iris= sns.load_dataset("iris")
iris_binary= iris[iris["species"]!= "virginica"]

x= iris_binary[["petal_length", "petal_width"]]
y= iris_binary["species"]
le= LabelEncoder()
y_encoded= le.fit_transform(y)
model= LogisticRegression(max_iter= 200)
```

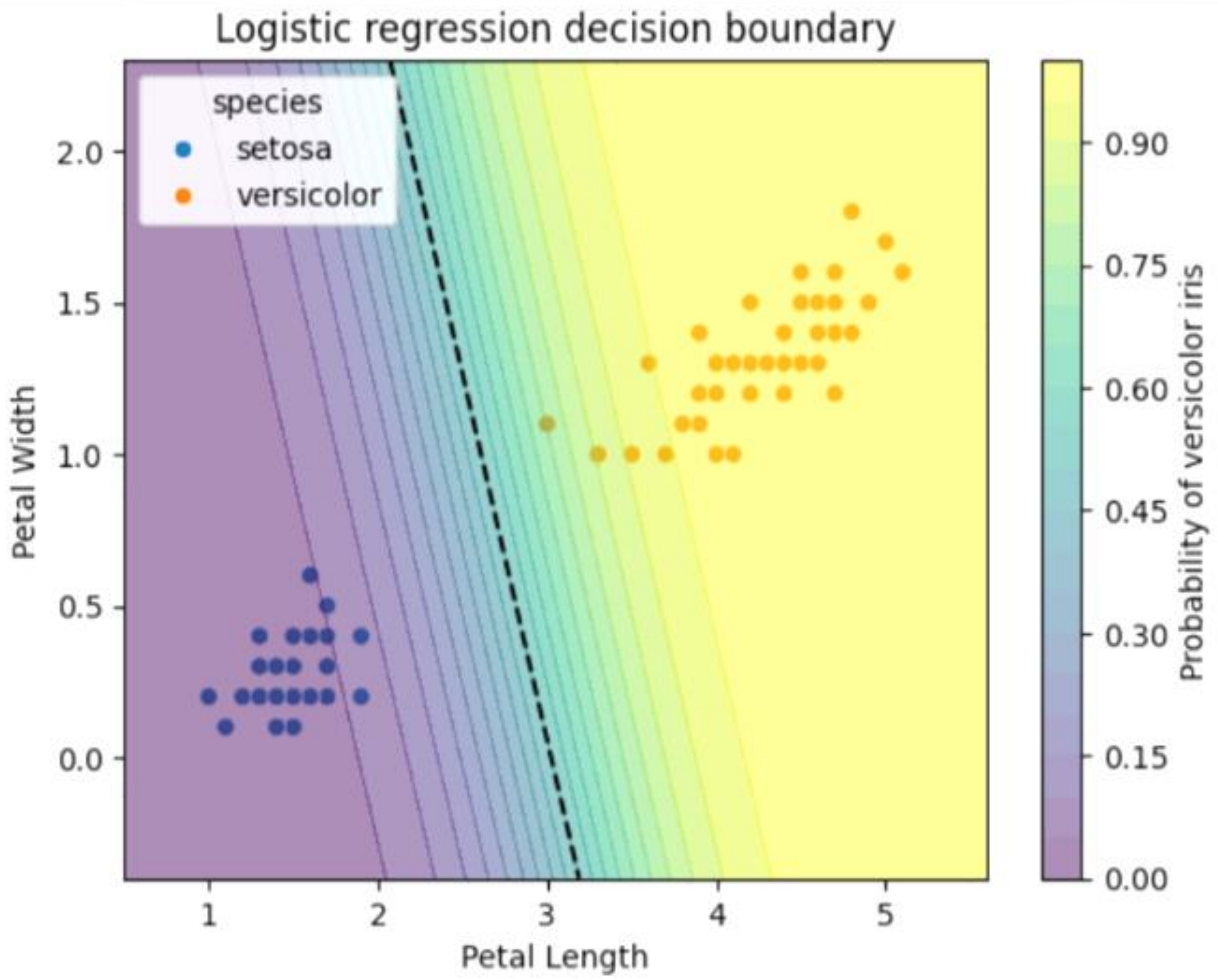
```
sns.scatterplot(data= iris_binary, x="petal_length", y="petal_width",
#Cutoff values
x_min, x_max = x["petal_length"].min() - 0.5, x["petal_length"].max()
y_min, y_max = x["petal_width"].min() - 0.5, x["petal_width"].max() +
x_cutoff, y_cutoff = np.meshgrid(np.linspace(x_min, x_max, 200),
                                 np.linspace(y_min, y_max, 200))
#predictions
grid = np.c_[x_cutoff.ravel(), y_cutoff.ravel()]
predic = model.predict(grid).reshape(x_cutoff.shape)

#probability contours
probs = model.predict_proba(grid)[:, 1].reshape(x_cutoff.shape)
contour = plt.contourf(x_cutoff, y_cutoff, probs, levels = 20, a
contours = plt.colorbar(contour)
contours.set_label("Probability of versicolor iris")
plt.contour(x_cutoff, y_cutoff, probs, levels=[0.5], colors='black',

#graph
plt.title("Logistic regression decision boundary")
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
```

Adding cutoff values

Making prediction model

Creating graph

Adding probability contours for 0.5 cutoff


Logistic regression decision boundary
species
setosa
versicolor
Petal Width
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
Petal Length
Probability of versicolor iris
0.90
0.75
0.60
0.45
0.30
0.15
0.00

THANK YOU FOR LISTENING