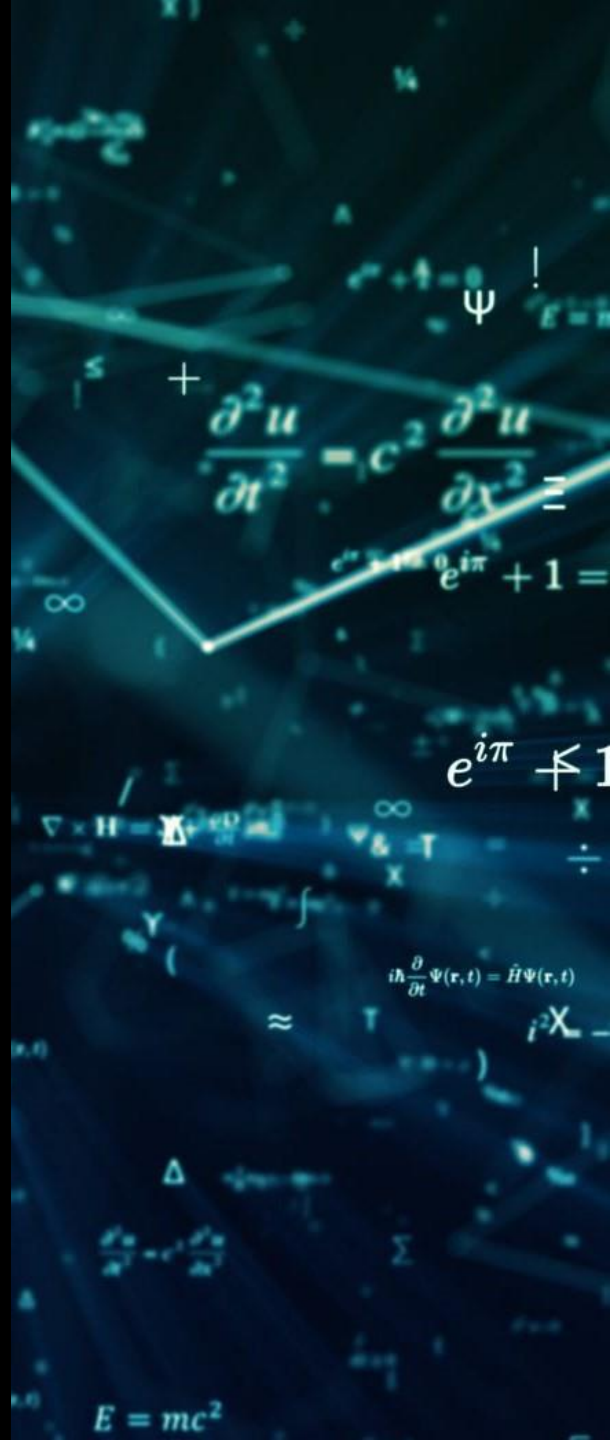
Artificial Intelligence Algorithms and Mathematics

CSCN 8000

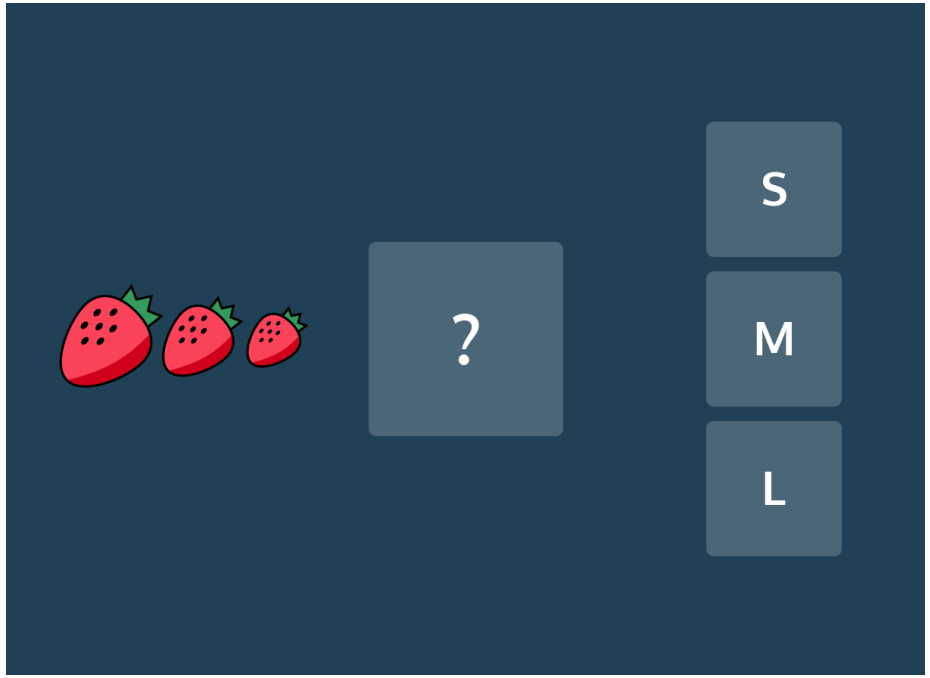


Classification

- Introduce Classification
- Classification algorithms: Logistic Regression
- Gradient Descent
- Linear vs Non-Linear methods
- Evaluation methods for classification methods

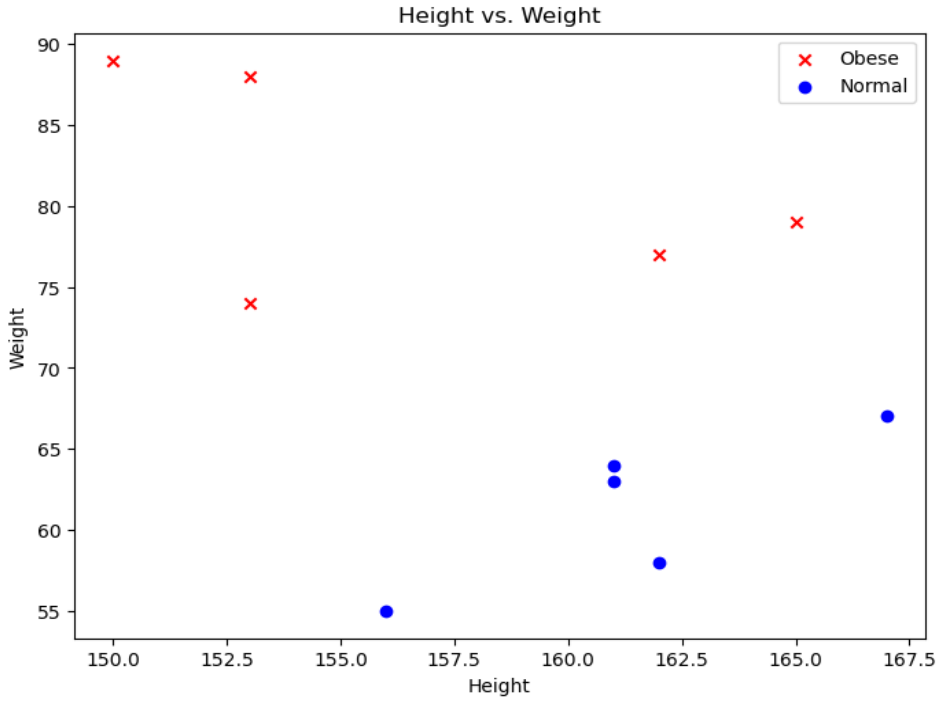
Classification Algorithms

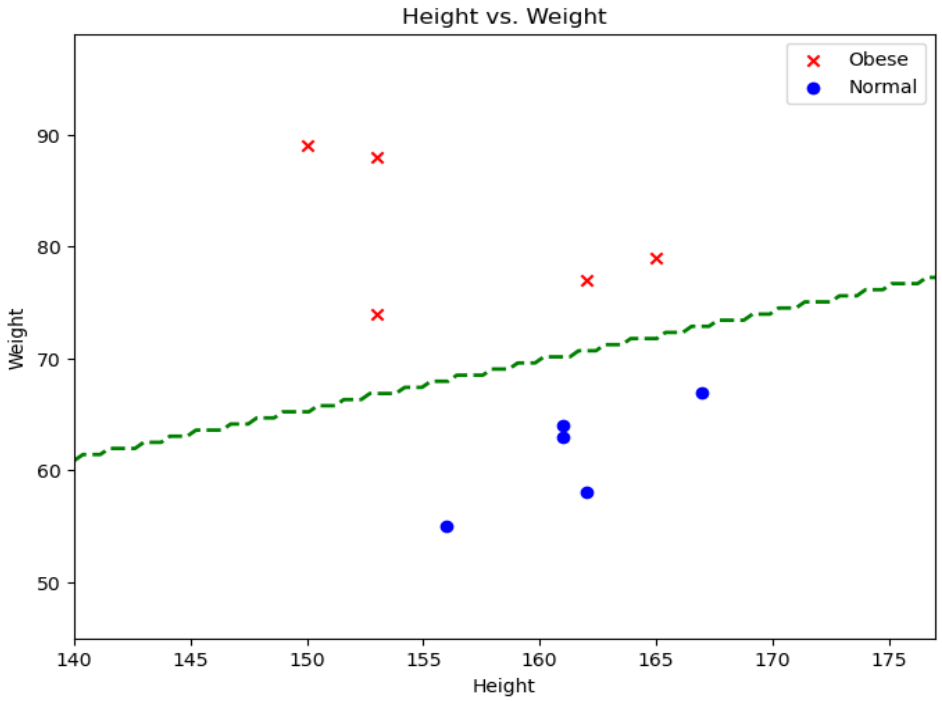
- Classification : Predicting discrete classes /categories
  - Binary Classification(between two)
    - Example : True/False, Object Identification- Cat /Not Cat
  - Multiclass classification(more than two)
    - Example : Animal Species
  - Multilabel classification(more than two)
    - Example : Colors in Image



Decision Boundary Definition

- Assume we have 10 different people with different weight and height values. 5 of those people are labeled as "Obese" and 5 are labeled as "Normal". The plot of those people is shown below. Is it possible to ***separate*** the two classes with a single line?





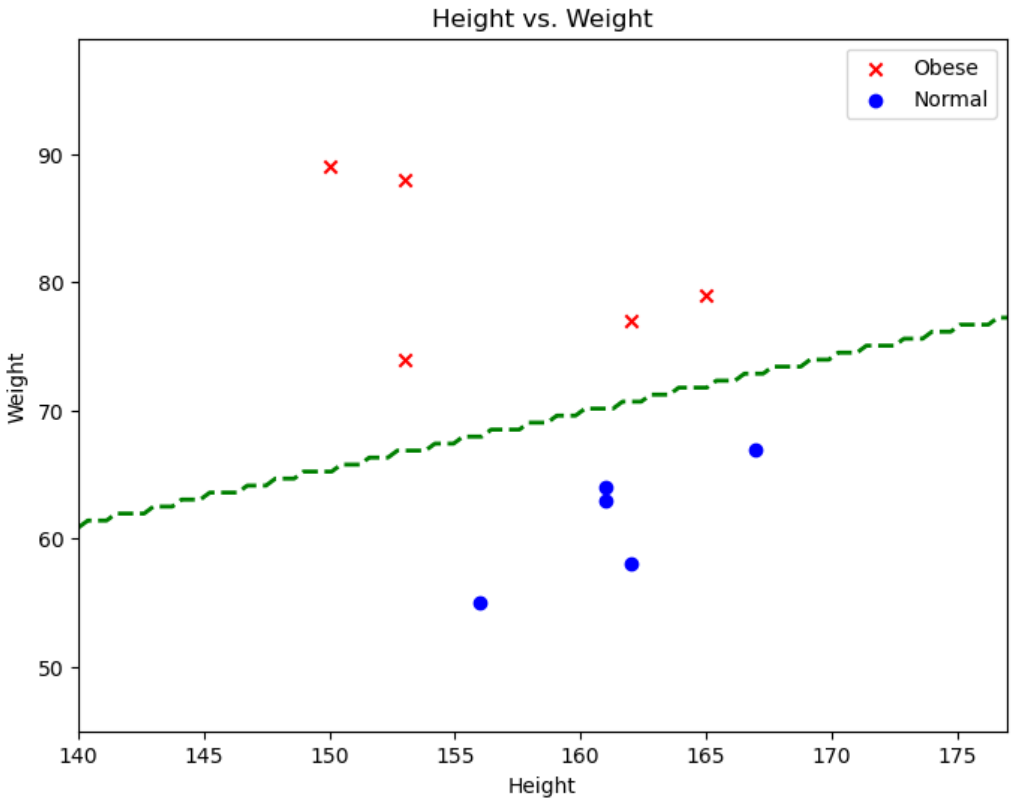
Decision Boundary Definition

- The decision boundary in a binary classification problem is a hypersurface that separates the feature space into regions corresponding to different classes.
- The decision boundary only exists as a line or hyperplane if the classes are ***linearly separable***
- The equation of the linear decision boundary is as follows:

$$\hat{y} = \vec{w}\,\vec{x} + b$$

  - $\vec{x}$ represents ***all*** input features.
  - $\hat{y}$ represents estimated class a point belongs to.
  - y represents the label of the point $\{0,1\}$

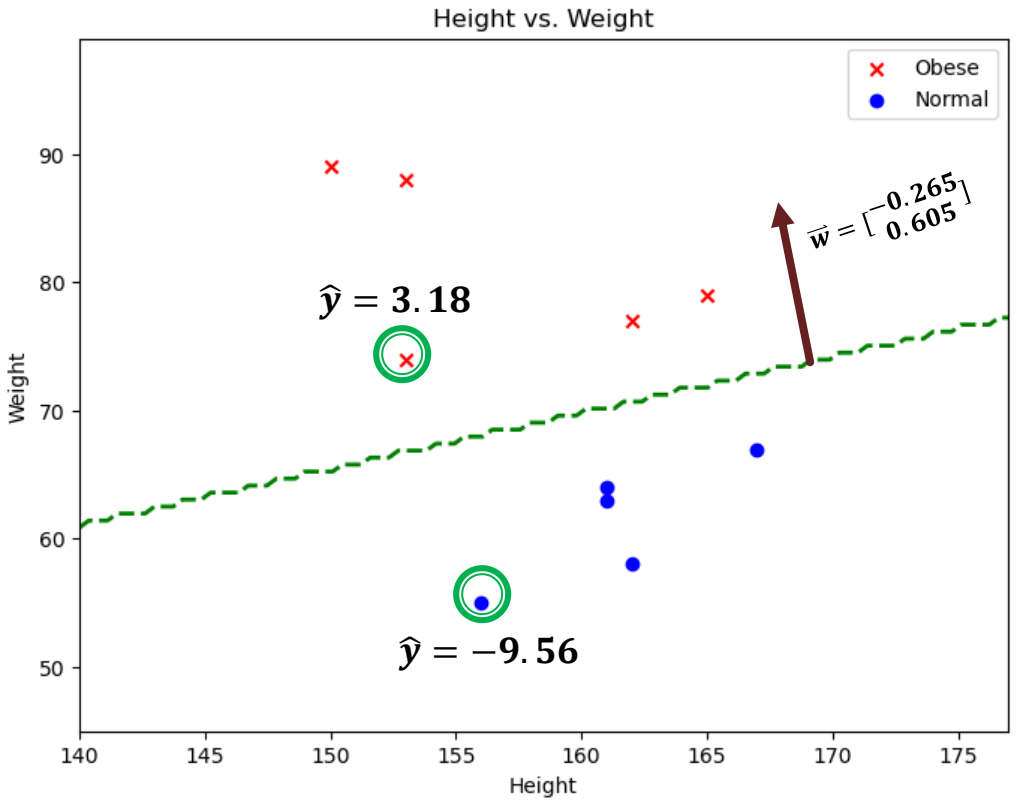
- $\vec{w}, b$ are learnable parameters to be estimated



$$\hat{y} = \vec{w}\,\vec{x} + b$$

Decision Boundary Characteristics

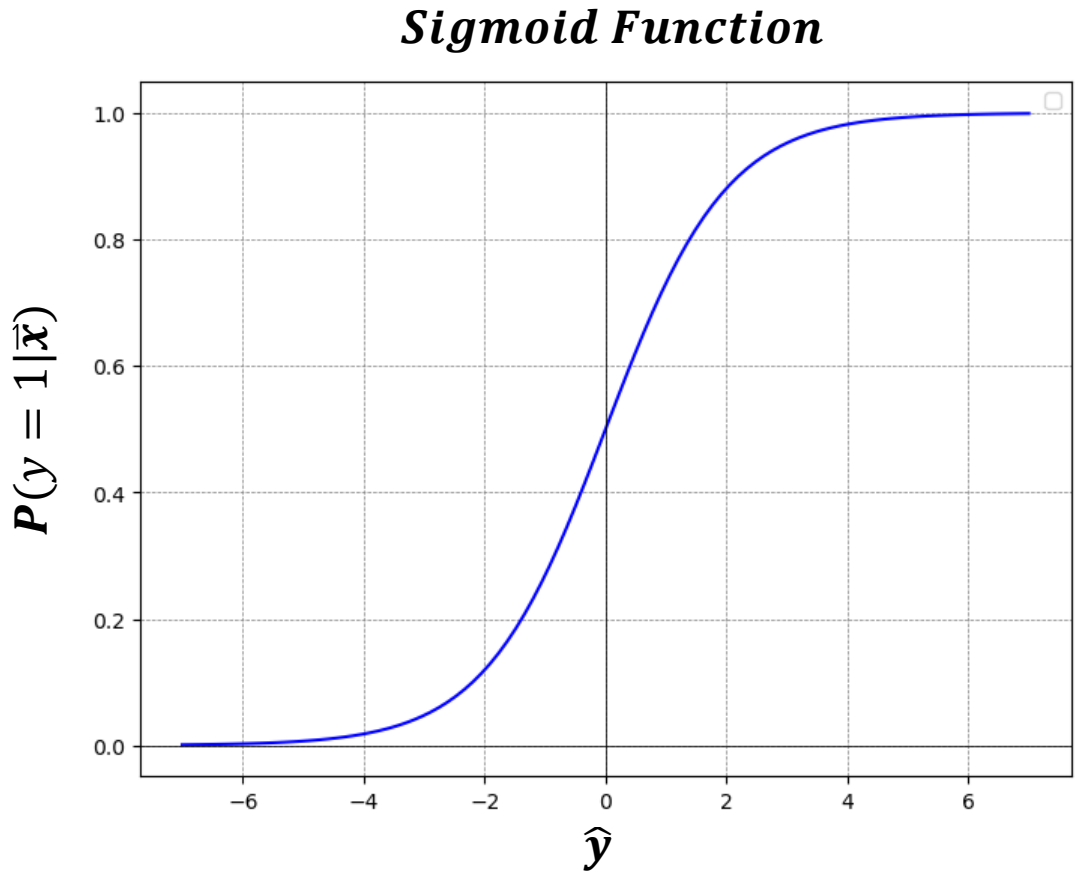
- Any point lying above the decision boundary → $\hat{y}$ is positive
- Any point lying below the decision boundary → $\hat{y}$ is negative
- Any point lying on the decision boundary → $\hat{y}$ is zero
- $\vec{w}$ is always perpendicular on the decision boundary.
- In any classification problem, our target is to find the decision boundary by estimating $\vec{w}, b$.
- Each classification algorithm has different ways (i.e. threshold) to classify the classes by matching $\hat{y}$ to y.



$$\hat{y} = w_1 * height + w_2 * weight + b$$
$$\hat{y} = -0.265 * height + 0.6054 * weight$$

Logistic Regression

- Is there a way to use the value of $\hat{y}$ to get the ***probability*** that a point belong to the positive class ($y = 1$)?
  - We need to find a function that can transform the positive and negative values of $\hat{y}$ to a probability in range $0 - 1$.



$$P(y = 1|\vec{x}) = \frac{1}{1 + e^{-\hat{y}}}$$

Logistic Regression

- Logistic regression is a classification algorithm used to predict the probability that a point belongs to a certain class $\boldsymbol{P}(y = 1|\vec{\boldsymbol{x}})$
- $\boldsymbol{P}(y = 1|\vec{\boldsymbol{x}}) = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e}^{-\hat{\boldsymbol{y}}}} = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e}^{-(\vec{\boldsymbol{w}}\,\vec{\boldsymbol{x}}+\boldsymbol{b})}}$
- In binary classification, each point $\vec{\boldsymbol{x}}$ has a label y that could be equal to 0 or 1.
- We want to represent the probability of observing the **correct** label of the point denoted as $\boldsymbol{P}(y|\vec{\boldsymbol{x}})$.

Logistic Regression

- We want to represent the probability of observing the **correct** label of the point denoted as $\boldsymbol{P}(y|\vec{\boldsymbol{x}})$.

$$\boldsymbol{P}(y|\vec{\boldsymbol{x}}) = \begin{cases} \boldsymbol{P}(y=1|\vec{\boldsymbol{x}})\ if\ \vec{\boldsymbol{x}}\ belong\ to + ve \\ \boldsymbol{1} - \boldsymbol{P}(y=1|\vec{\boldsymbol{x}})\ if\ \vec{\boldsymbol{x}}\ belong\ to - ve \end{cases}$$

- Is there a way to represent it in one equation only to handle both cases?

$$\boldsymbol{P}(y|\vec{\boldsymbol{x}}) = \boldsymbol{P}(y=1|\vec{\boldsymbol{x}})^{y} * [\boldsymbol{1} - \boldsymbol{P}(y=1|\vec{\boldsymbol{x}})]^{\boldsymbol{1}-y}$$

Logistic Regression

- In a dataset with N number of datapoints, what is the probability of observing the correct label y for ***all the points*** in the dataset?

$$\boldsymbol{P}(y|\vec{\boldsymbol{x}}) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{P}(y_i = 1|\vec{\boldsymbol{x_i}})^{y_i} * [\boldsymbol{1} - \boldsymbol{P}(y_i = 1|\vec{\boldsymbol{x_i}})]^{\boldsymbol{1}-y_i}$$

- The <u>target</u> of the logistic regression model is to ***maximize*** the probability of observing the correct label y for ***all the points*** in the dataset → maximize $[\boldsymbol{P}(y|\vec{\boldsymbol{x}})]$

Logistic Regression

- ML algorithms favor ***minimizing*** the target functions and including **summations** rather than multiplications to help in the differentiation.
  - Solution:
    - Apply ***log*** on the target function (replaces multiplication with summation)
    - Add ***–ve sign*** to the target function to be minimized rather than maximized.

$$\boldsymbol{P}(y|\vec{\boldsymbol{x}}) = \prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{P}(y_i = 1|\vec{\boldsymbol{x_i}})^{y_i} * [\boldsymbol{1} - \boldsymbol{P}(y_i = 1|\vec{\boldsymbol{x_i}})]^{\boldsymbol{1}-y_i}$$

Log and –ve sign

$$\boldsymbol{L}(\vec{\boldsymbol{w}}, \boldsymbol{b}) = -\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{log}(\boldsymbol{P}(y_i = 1|\vec{\boldsymbol{x_i}}))\boldsymbol{y_i} + (\boldsymbol{1} - \boldsymbol{y_i})\boldsymbol{log}[\boldsymbol{1} - \boldsymbol{P}(y_i = 1|\vec{\boldsymbol{x_i}})]$$

Cost function to be minimized

Logistic Regression: Loss Minimization

$$\boldsymbol{L(\vec{w}, b) = -\sum_{i=1}^{N} log(P(y_i = 1|\vec{x_i}))y_i + (1 - y_i)log[1 - P(y_i = 1|x_i)]}$$

$$\boldsymbol{P(y = 1|\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\,\vec{x} + b)}}}$$

- Get the values of $\vec{w}, b$ at which $L(\vec{w}, b)$ is minimum → Get the values of $\vec{w}, b$ at which $\frac{d(L)}{d\vec{w}} = 0$, and $\frac{d(L)}{db} = 0$.

- For logistic regression, $\frac{d(L)}{d\vec{w}} = 0$ and $\frac{d(L)}{db} = 0$ ***do not have*** closed form solutions that can be derived by making $\vec{w}$ and b the subjects of their equations.
- For that reason, we utilize iterative optimization approaches like the famous ***Gradient Descent*** algorithm.

Logistic Regression: Loss Minimization

$$L(\vec{w}, b) = -\sum_{i=1}^{N} log(P(y_i = 1|\vec{x_i}))y_i + (1 - y_i)log[1 - P(y_i = 1|\vec{x_i})]$$

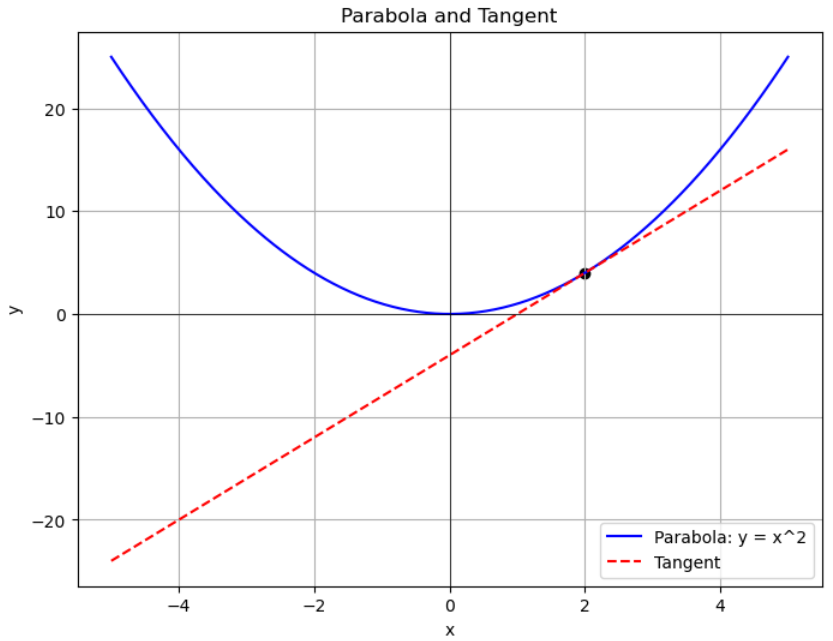
$$P(y = 1|\vec{x}) = \frac{1}{1 + e^{-(\vec{w}\,\vec{x}+b)}}$$

- We utilize iterative optimization approaches like the famous ***Gradient Descent*** algorithm to minimize $L(\vec{w}, b)$.

$$\frac{dL}{d\vec{w}} = \sum_{i=1}^{N} \left(y_i - \frac{e^{\vec{w}\,\vec{x_i}+b}}{1 + e^{\vec{w}\,\vec{x_i}+b}} \right) x_i$$

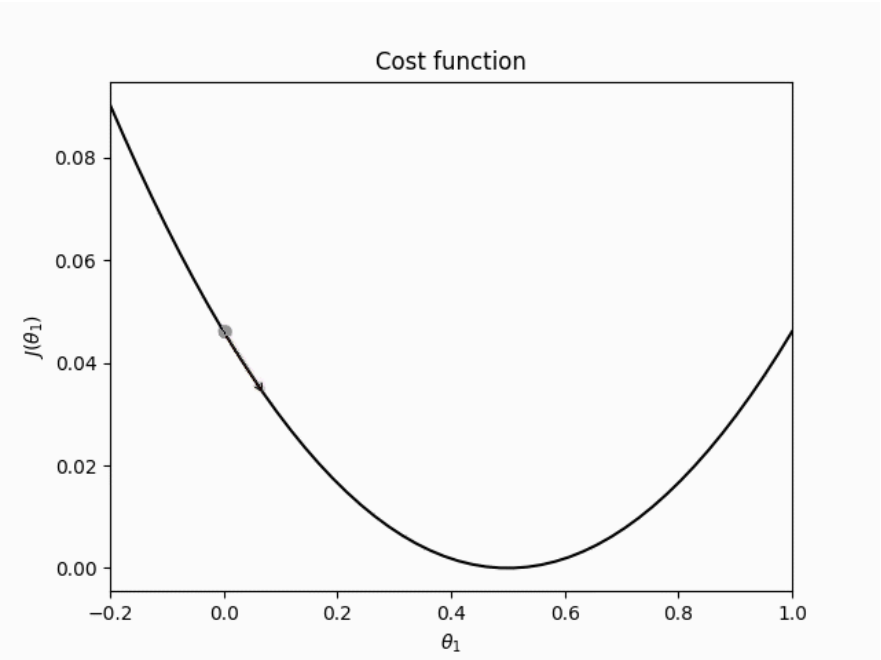
Gradient Descent

- Given a parabola in $2D$, how to get the gradient at any point along the curve?
  - Get the gradient of the ***tangent*** of the curve at the point.
  - The **tangent** to a curve at a specific point is a straight line that just "touches" the curve at that point.



Gradient Descent

- Now assume the x-axis represents the weights to be estimated $\vec{w}$ and the y-axis represents a loss (cost) function $L(\vec{w})$.
- Our target is to find the value of $\vec{w}$ at which $L(\vec{w})$ is minimum.
- To achieve this iteratively starting from some random value of $\vec{w}$, we need to keep moving in the **negative** direction of the gradient until reaching the minimum value.

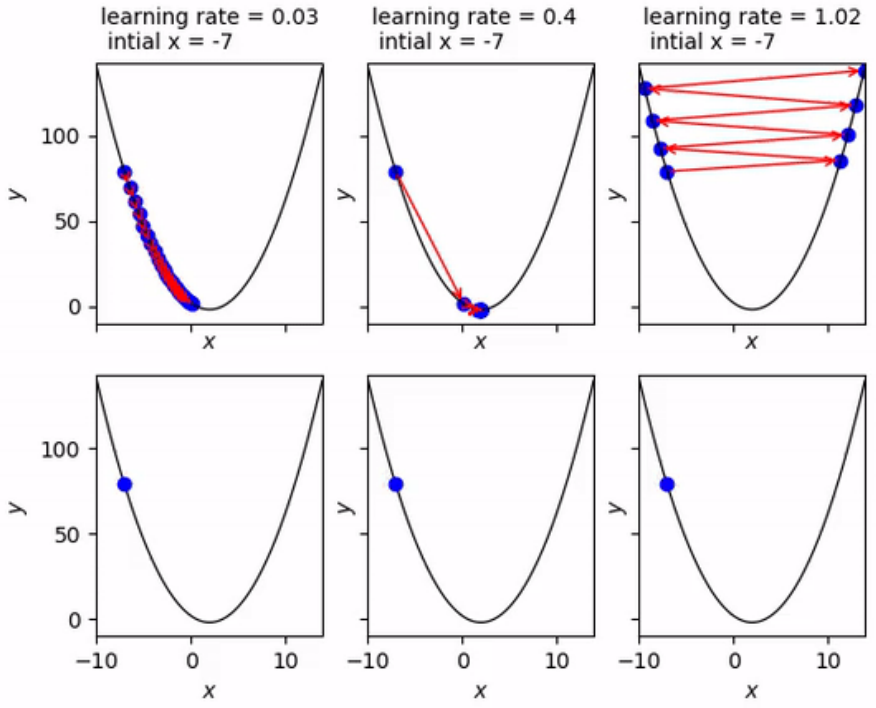


Gradient Descent

- In other words, gradient descent keeps updating the value of $\vec{w}$ in the **negative** direction of the gradient by a certain ***step size*** until reaching the minimum value such that:

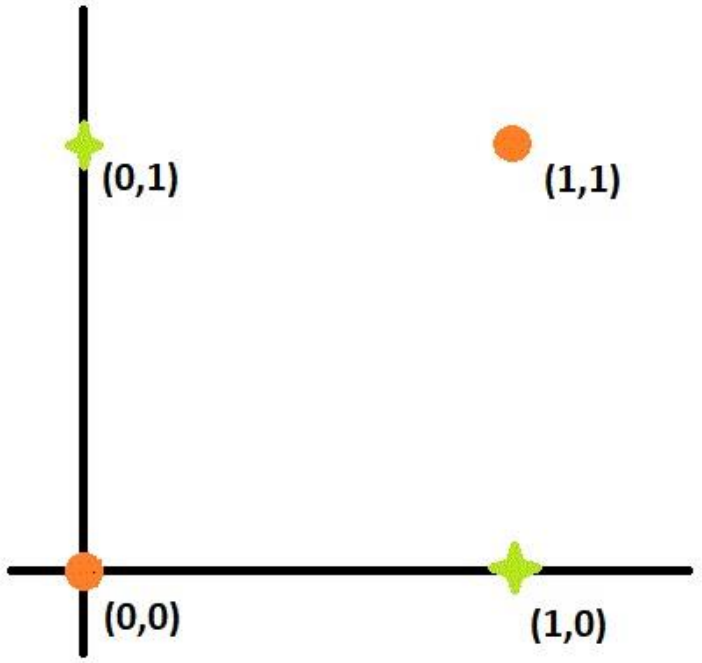
$$\vec{w}_{t+1} = \vec{w}_t - \delta \frac{dL}{d\vec{w}}$$

- The step size (δ) is a manually defined hyperparameter called the learning rate and it controls the magnitude of the change done for $\vec{w}$ in each step.
  - If we set δ too high → Might not converge to minimum
  - If we set δ too low → Might take a very long time to converge to minimum



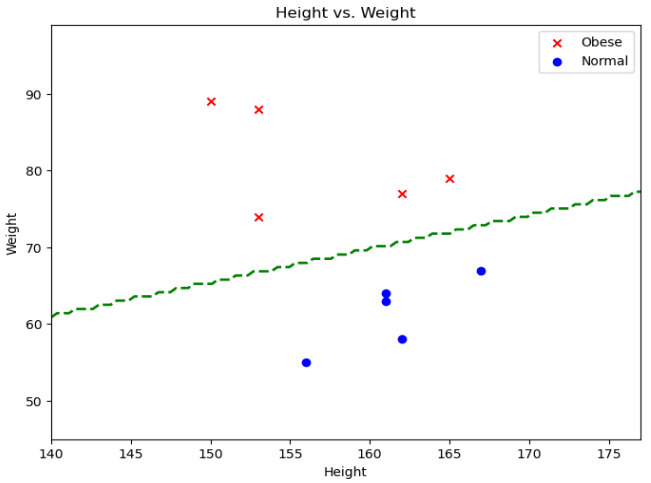
Linearly vs Non-Linearly Separable

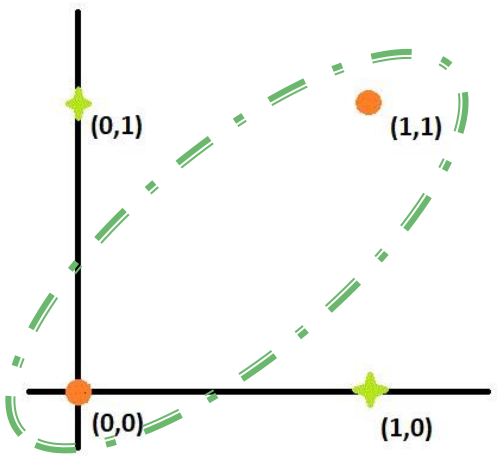
- Is it possible to have a linear decision boundary correctly separating the following classes?



Linearly vs Non-Linearly Separable

- In a **linearly separable problem**, it is possible to draw a straight line (or hyperplane in higher dimensions) that completely separates the classes or categories of data points.
  - Linearly separable problems can be effectively solved using linear models such as logistic regression.
- In a **non-linearly separable problem**, it is not possible to draw a straight line or hyperplane to separate the classes. The data points are intermingled in a way that a linear boundary cannot accurately classify them.
  - The decision boundary in a non-linearly separable problem is a complex curve or surface.
  - Non-linearly separable problems require more complex models that can learn and represent non-linear relationships.





Evaluation Metrics for Classification

- The evaluation metrics for classification problems measure how accurate are the predicted classes compared to the ground truth classes.
- They give us a *single* value that represents the classification ability of our model.
  - Accuracy
  - Precision
  - Recall
  - F1-Score
  - Specificity
  - AUC

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Confusion Matrix

- Assume we have a spam classification model that is already trained and evaluated on 100 data points. There are 50 points in the positive class (Spam) and 50 in the negative class (Not Spam). When you evaluated the model, the following output was found:
  - 40 **spam** points were predicted as **spam**
  - 10 **spam** points were predicted as **not spam**
  - 35 **not spam** points were predicted as **not spam**
  - 15 **not spam** points were predicted as **spam.**

		Actual: Positive	Actual: Negative	
Prediction	Positive	40	15	55
Prediction	Negative	10	35	45
		50	50	100

		Actual: Positive	Actual: Negative
Prediction	Positive	True Positive (TP)	False Positive (FP)
Prediction	Negative	False Negative (FN)	True Negative (TN)

Confusion Matrix

- A confusion matrix is a tabular representation used in machine learning and statistics to assess the performance of a classification model. It provides a detailed breakdown of the model's predictions compared to the actual outcomes in a classification problem.

		Actual: Positive	Actual: Negative
Prediction	**Positive**	True Positive (TP)	False Positive (FP)
Prediction	**Negative**	False Negative (FN)	True Negative (TN)

Accuracy

- Definition:
  - Accuracy is the measure of how correctly the model predicts all classes, both positive and negative. It calculates the ratio of correctly predicted instances to the total instances.
- Formulation:
  - $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

		Actual: Positive	Actual: Negative
Prediction	Positive	True Positive (TP)	False Positive (FP)
Prediction	Negative	False Negative (FN)	True Negative (TN)

Precision

- Definition:
  - Precision measures the proportion of true positive predictions out of all positive predictions made by the model.
  - In other words, from all the positives predicted by the model, how many were correct?
  - Important in search algorithms when the retrieved links need to be mostly relevant.
- Formulation:
  - $Precision = \frac{TP}{TP+FP}$

		Actual: Positive	Actual: Negative
Prediction	Positive	True Positive (TP)	False Positive (FP)
Prediction	Negative	False Negative (FN)	True Negative (TN)

Recall (Sensitivity / True Positive Rate)

- Definition:
  - Recall measures the proportion of true positive predictions out of all actual positives in the dataset.
  - In other words, from all the positives in the dataset, how many did the model predict correctly?
  - Important in medical screenings, where missing a true positive in this context can delay treatment.
- Formulation:
  - $Recall = \frac{TP}{TP+FN}$

		Actual: Positive	Actual: Negative
Prediction	Positive	True Positive (TP)	False Positive (FP)
Prediction	Negative	False Negative (FN)	True Negative (TN)

Specificity (True Negative Rate)

- Definition:
  - Specificity measures the proportion of true negatives out of all actual negatives.
  - In other words, from all the negatives in the dataset, how many did the model predict correctly?
  - Important in spam filtering, where marking a legitimate email as spam can lead to important messages being overlooked or missed.
- Formulation:
  - $Specificity = \frac{TN}{TN+FP}$

		Actual: Positive	Actual: Negative
Prediction	Positive	True Positive (TP)	False Positive (FP)
Prediction	Negative	False Negative (FN)	True Negative (TN)

Example

- Assume we have a dataset of 95 Not Spam (-ve) points and 5 Spam (+ve) points. You trained a model on this dataset and upon evaluation it predicted 99 data points to be Not Spam and 1 Spam point to be Spam.
  - $Accuracy = \frac{96}{100} = 96\%$
  - $Precision = \frac{1}{1} = 100\%$
  - $Recall = \frac{1}{5} = 20\%$
  - Is accuracy a good representative of the performance of the model?
    - In imbalanced datasets, accuracy is not a reliable measure of performance. We need another measure mixing both precision and recall reliably in one metric.

Prediction	Actual: Positive	Actual: Negative	
Positive	1	0	1
Negative	4	95	99
	5	95	100

F1-Score

- Definition:
  - F1-Score is a metric that combines precision and recall into a single value. It is the harmonic mean of precision and recall, providing a balanced measure of a model's performance.
  - Particularly useful in situations where both false positives and false negatives are critical or we have imbalanced datasets.
- Formulation:
  - $F1 - Score = \frac{2*Precision*Recall}{Precision+Recall}$

		Actual: Positive	Actual: Negative
Prediction	Positive	True Positive (TP)	False Positive (FP)
Prediction	Negative	False Negative (FN)	True Negative (TN)

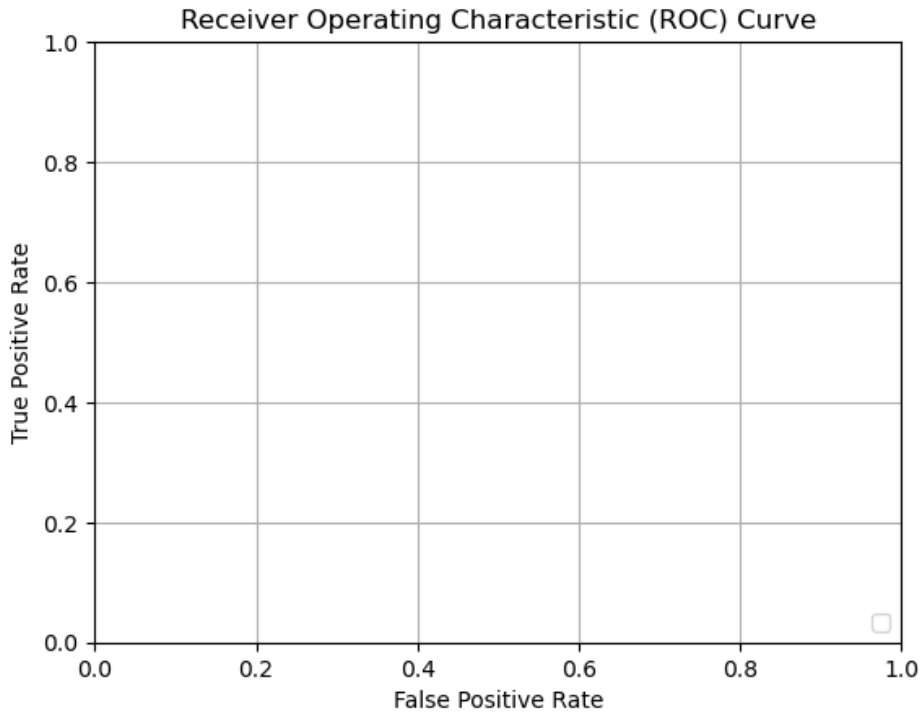
Example

- Assume we have a dataset of 95 Not Spam (-ve) points and 5 Spam (+ve) points. You trained a model on this dataset and upon evaluation it predicted 99 data points to be Not Spam and 1 Spam point to be Spam.
  - $Accuracy = \frac{96}{100} = \mathbf{96\%}$
  - $Precision = \frac{1}{1} = 100\%$
  - $Recall = \frac{1}{5} = 20\%$
  - $F1 - Score = \frac{2*20*100}{20+100} = \mathbf{33.3\%}$

		Actual		
		Positive	Negative	
Prediction	Positive	1	0	1
	Negative	4	95	99
		5	95	100

ROC Curve

- Assume we have a model that can tell if a person is sick or not. But, like any model, it's not perfect. Sometimes it says a healthy (-ve) person is sick (+ve), and sometimes it says a sick person (+ve) is healthy (-ve).
- Our model produces a probability from 0 to 1 that a person is sick.
- We can manually set a threshold on the output probability. If the probability is higher than the threshold, the person is labeled as sick.
- We want to figure out how good this model really is. We will plot a 2D graph for this purpose.
  - X-axis represents False Positive Rate (FPR) → How many of the healthy people do we label as sick.
  - Y-axis represents True Positive Rate (TPR) →How many of the sick people do we label as sick.

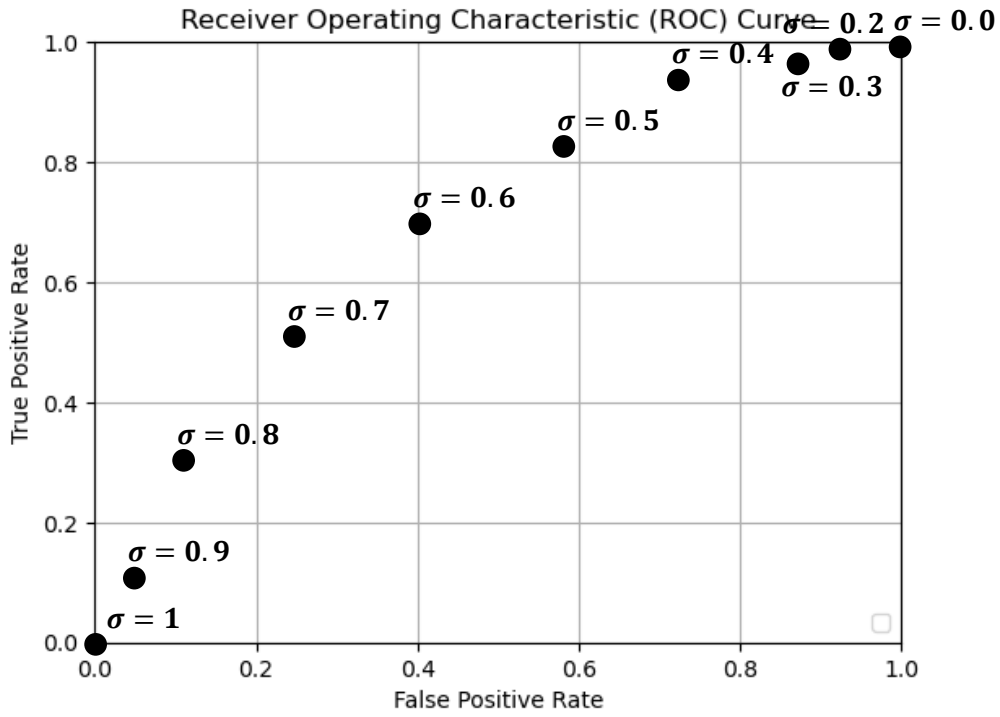


$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{TN + FP}$$

ROC Curve

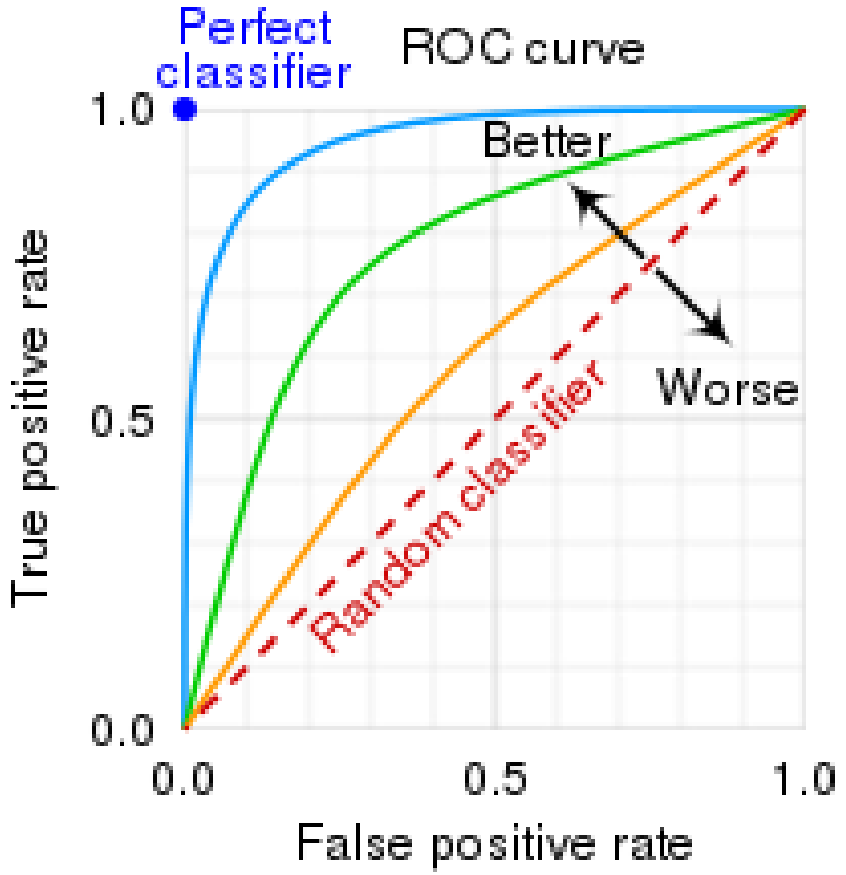
- We can manually set a threshold σ on the output probability. If the probability is higher than the threshold, the person is labeled as sick. Assuming we have a non-perfect classifier.
  - If we set $\sigma = 1 \rightarrow \mathrm{TP}R \approx 0, FPR \approx 0$
  - If we set $\sigma = 0.9 \rightarrow \mathrm{TP}R \approx 0.1, FPR \approx 0.05$
  - If we set $\sigma = 0.8 \rightarrow \mathrm{TP}R \approx 0.3, FPR \approx 0.1$
  - If we set $\sigma = 0.7 \rightarrow \mathrm{TP}R \approx 0.55, FPR \approx 0.25$
  - If we set $\sigma = 0.6 \rightarrow \mathrm{TP}R \approx 0.65, FPR \approx 0.40$

    ....
  - If we set $\sigma = 0.2 \rightarrow \mathrm{TP}R \approx 0.97, FPR \approx 0.85$
  - If we set $\sigma = 0.1 \rightarrow \mathrm{TP}R \approx 0.99, FPR \approx 0.92$
  - If we set $\sigma = 0.0 \rightarrow \mathrm{TP}R \approx 1.0, FPR \approx 1.0$



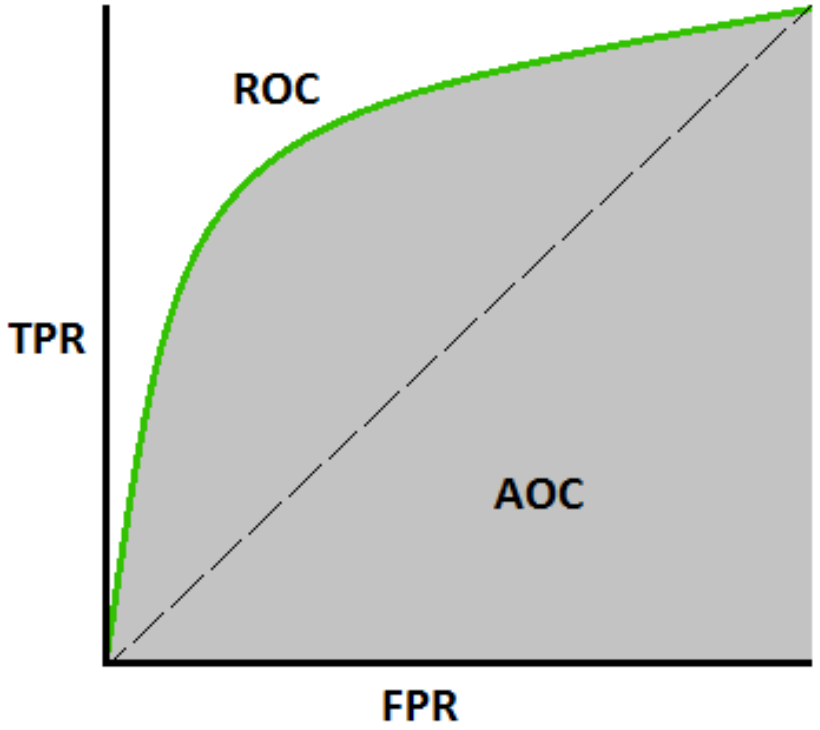
ROC Curve

- A random classifier will always have a 50% chance to label a point in the positive class → Identity Line
- A better classifier will have higher true positives compared to false positives in the higher thresholds → A more steep curve
- A worse classifier will have lower true positives compared to false positives in the higher thresholds → A less steep curve
- A perfect classifier will have 100% true positives in all thresholds, and false positives increase from 0 to 100% as threshold increases.



Area Under ROC Curve (AUC)

- Definition:
  - The AUC provides a single scalar value that summarizes the model's performance across various threshold settings.
  - Gives an understanding on how confident the model is in distinguishing between the classes.
  - A higher AUC means the model predicts the positive classes with high confidence (probability).
  - Useful when we have imbalanced datasets and care about the confidence of the model in classifying the positive class.
- Formulation:
  - $Compute\ the\ area\ under\ the\ ROC\ curve$



Activity

- Given the available diabetes.csv dataset. Train a Logistic Regression model to predict whether a person has diabetes or not.
  - Train and evaluate on the whole dataset (no need to split)
  - Evaluate the model performance by printing the classification_report
  - Plot the ROC Curve by predicting the probabilities using the predict_proba function and extracting the positive class
  - Print the AUC

References

- https://learning.oreilly.com/library/view/practical-statistics-for/9781491952955/ch06.html
- https://www.mathsisfun.com/data/standard-deviation.html

Thank you!

- Any questions?

Disclaimer

Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution

CONESTOGA

Thank You

Youssef Abdelkareem

yabdelkareem@conestogac.on.ca

**** Credits for Dr. Anasuya Bhima for assisting in the slides structure***