CLASSIFICATION (CONCEPTS – PART 2)

Machine Learning for Financial Data

February 2021

Contents

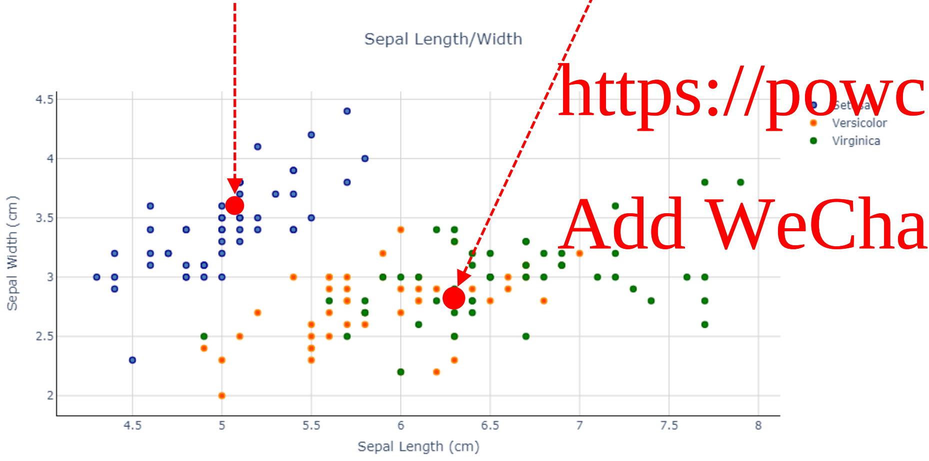
- Naive Bayes Classifier
- Support Vector Machine (SVM)
- Random Forest
- Logistic Regression

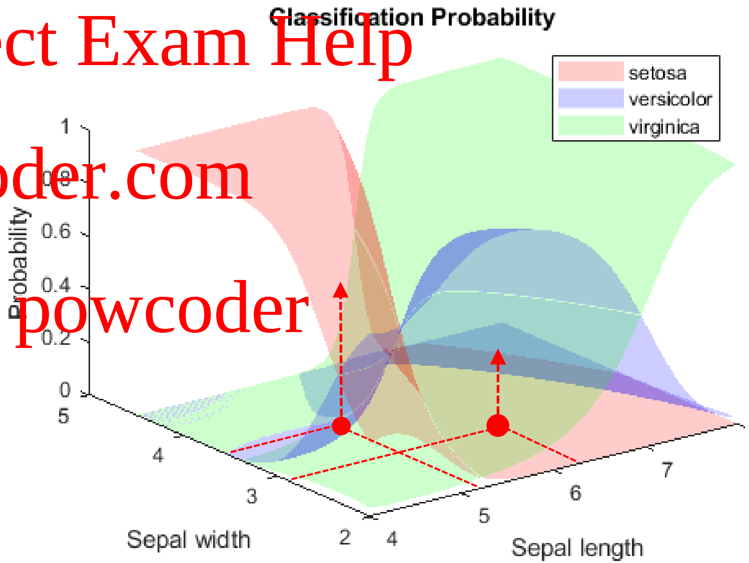
Naive Bayes Classifier

Naïve Bayes classifier relies on the probability function of pedal & sepal measures to species over the sample space

prediction is obvious

prediction is not so obvious



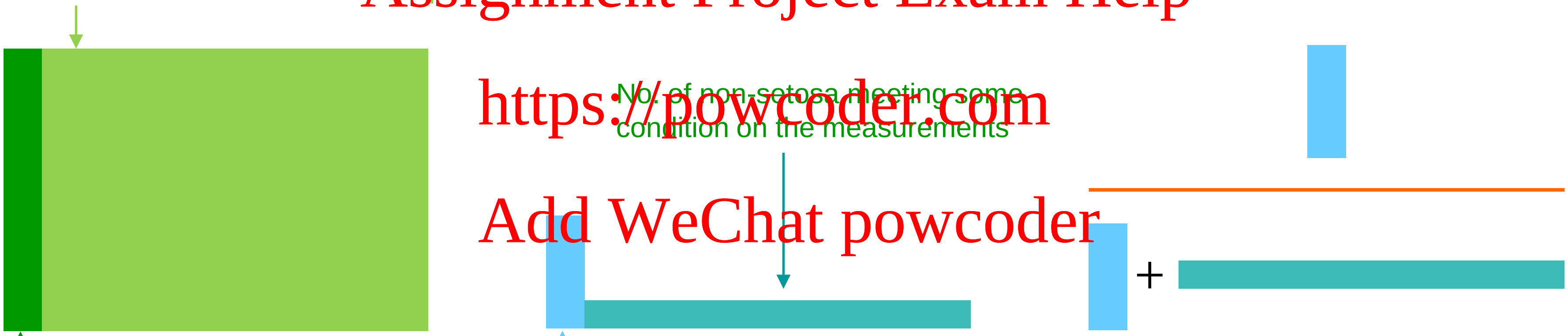


Bayes' Theorem is about updating the belief based on evidence

ALL POSSIBILITIES

ALL POSSIBILITIES FITTING THE EVIDENCE

PROBABILITY OF A HYPOTHESIS GIVEN THE EVIDENCE IS TRUE



No. of non-setosa in the sample

No. of setosa in the sample

No. of non-setosa meeting some condition on the measurements

No. of setosa meeting some condition on the measurements

Probability of a sample being a setosa given some condition on the measurements

Bayes' Theorem (1)

$P(H)$	Probability of a hypothesis being true (before any evidence)
$P(E\|H)$	Probability of seeing the evidence if the hypothesis is true
$P(E)$	Probability of seeing the evidence
$P(H\|E)$	Probability a hypothesis being true given seeing the evidence

$$P(H|E) \cdot P(E) = P(E|H) \cdot P(H) = P(H \cap E) = P(E \cap H)$$

Bayes' Theorem (2)

Background Proposition	B	$\neg B$ (not B)	Total
A	$P(B\|A) \cdot P(A)$ $= P(A\|B) \cdot P(B)$	$P(\neg B\|A) \cdot P(A)$ $= P(A\|\neg B) \cdot P(\neg B)$	$P(A)$
$\neg A$ (not A)	$P(B\|\neg A) \cdot P(\neg A)$ $= P(\neg A\|B) \cdot P(B)$	$P(\neg B\|\neg A) \cdot P(\neg A)$ $= P(\neg A\|\neg B) \cdot P(\neg B)$	$P(\neg A) = 1 - P(A)$
Total	$P(B)$	$P(\neg B) = 1 - P(B)$	1

Bayes' Theorem (3)

No. of setosa = 10

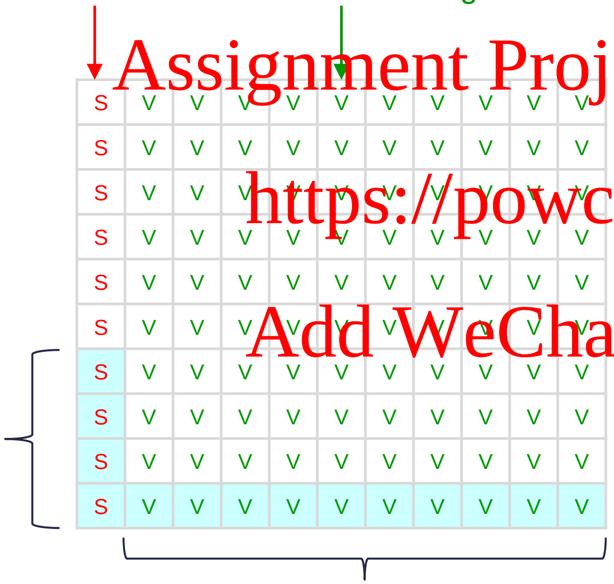
No. of versicolor & virginica = 100

PRIOR:

$P(S) = 1/11$

LIKELIHOOD:

$P(M|S) = 0.4$



S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V
S	V	V	V	V	V	V	V	V	V	V

$P\ (M|\neg S) = 0.1$

POSTERIOR: $P(S|M)$

$$= \frac{\begin{smallmatrix}s\\s\\s\\s\end{smallmatrix}}{\begin{smallmatrix}s\\s\\s\\s\end{smallmatrix} + v\ v\ v\ v\ v\ v\ v\ v\ v\ v}$$

$$= \frac{(\#Iris) \cdot P(S) \cdot P(M|S)}{(\#Iris) \cdot P(S) \cdot P(M|S) + (\#Iris) \cdot P(\neg S) \cdot P(M|\neg S)}$$

$$= \frac{P(S) \cdot P(M|S)}{P(S) \cdot P(M|S) + P(\neg S) \cdot P(M|\neg S)}$$

$$= \frac{0.0909 \cdot 0.4}{0.0909 \cdot 0.4 + 0.9091 \cdot 0.1} \quad = \frac{0.0364}{0.0364 + 0.0909}$$

$$= 0.2857$$

Gaussian Naive Bayes classifier relies on probability of each feature value within a class and the class probability

- A Naive Bayes classifier is a probabilistic ML model that is used for classification
- The crux of the classifier is based on the Bayes' theorem

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

- The theorem provides the probability of A happening, given that B has occurred
  - B is the evidence and A is the hypothesis
  - Features are assumed to be independent; hence, it is called naïve

Iris classification is based the maximum probability value of the 3 species classes given 4 sepal & petal measurements

- Question: which species has the highest probability given 4 measurements
- The hypothesis (y) is the Iris being one of the three species

- The evidence (x_1, x_2, x_3, x_4) is the 4 sepal and petal measurements

$$P(y|x_1, x_2, x_3, x_4) = \frac{P(x_1|y)\cdot P(x_2|y)\cdot P(x_3|y)\cdot P(x_4|y)\cdot P(y)}{P(x_1)\cdot P(x_2)\cdot P(x_3)\cdot P(x_4)}$$

- Given that the denominator is a constant, the probability of an Iris being a particular species (y) given the 4 measurements (x_i) can be expressed as

$$P(y|x_1, x_2, x_3, x_4) \propto P(y) \prod_{i=1}^{4} P(x_i|y)$$

- The initial estimation of $P(y)$ is simply the proportion of y among the samples
- The species with the largest probability will be taken as the prediction

Python: Fitting a Naive Bayes Model to Make Prediction

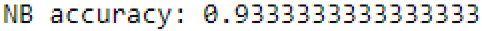
```
# load relevant modules
from sklearn.naive_bayes import GaussianNB

# instantiate a Naive Bayes classifier
# https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
nb = GaussianNB()

# fit/train the classifier to the training dataset
model = nb.fit(X_train, y_train)

# predict the targets for the test features
test_t = model.predict(X_test)

# calculate the accuracy score for the predicted targets using the known targets
print("NB accuracy:", accuracy_score(y_test, test_t))
```



```
NB accuracy: 0.9333333333333333
```

Naïve Bayes Classifier in a Nutshell

	Property	Description
1	Feature Data Types	Categorical or numerical.
2	Target Data Types	Categorical (with probability).
3	Key Principles	Uses the Bayes' theorem of conditional probabilities. For each feature, it calculates the probability for a class depending on the value of the feature.
4	Hyperparameters	None
5	Data Assumptions	Assume features are independent. Numerical features are assumed to be normally distributed.
6	Performance	Low computation cost. Fast and accurate. Efficient on large datasets.
7	Accuracy	When assumption of independence holds, outperform even highly sophisticated classification methods. Also perform well in multi-class prediction hence mostly used in text classification, e.g. spam filtering, sentiment analysis. Classifier combination technique like ensembling, bagging and boosting would not help its performance since their purpose is to reduce variance but Naive Bayes has no variance to minimize.
8	Explainability	How much each feature contributes to a class prediction is Interpretable in the form of conditional probability.

Support Vector Machine (SVM)

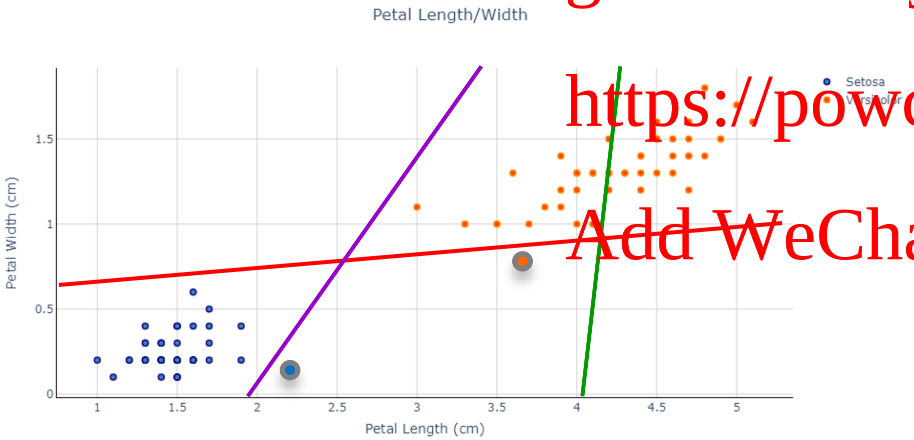
Support Vector Machine

A SVM is a powerful and versatile ML model, capable of performing linear or non-linear classification, regression, and even outlier detection. It is one of the more complex but accurate family of models making it one of most popular models in ML despite being a black box technique. SVMs are particularly well suited for classification of complex and small- or medium-size datasets.

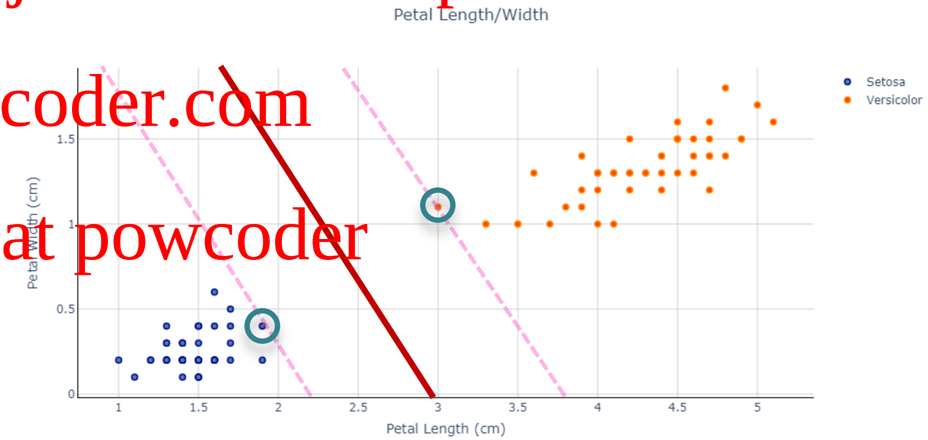
Linear SVM Classification

An SVM classifier tries to fit the widest possible street between the data points – large margin classification

- Using the Iris dataset, the scatterplot showing petal length vs petal width can clearly be separated easily with a straight line – linearly separable

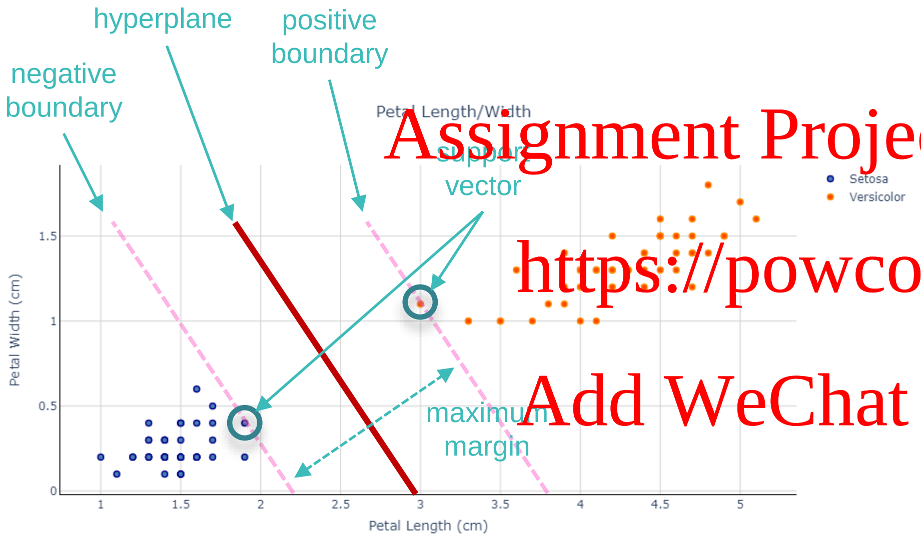


3 possible linear classifiers: green is bad, the other two too close to the data points & may not perform well on new data



an SVM classifier: the line not only separates the two classes but also stays as far away from the closest training data points as possible

Hard margin classification may not generalize well



negative boundary

hyperplane

positive boundary

support vector

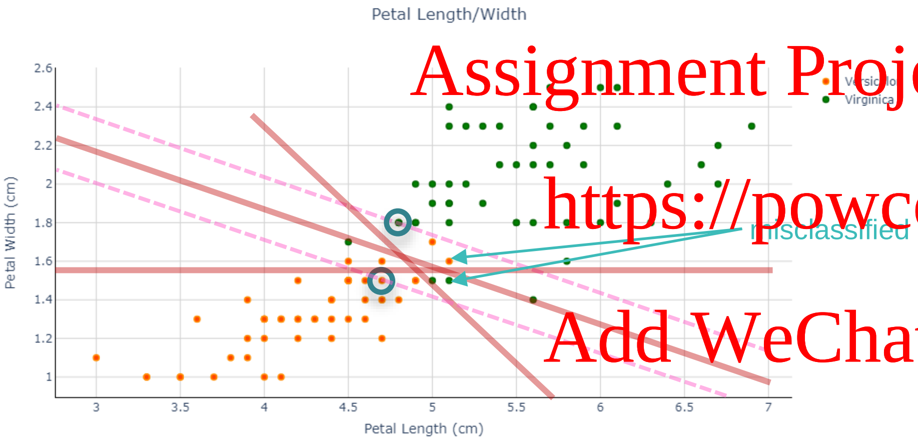
maximum margin

HARD MARGIN / CONSTRAINT

no data point is allowed to appear in the street implying that misclassification is not allowed

- Strictly imposing that all instances must be off the street is called hard margin classification
- Only works if the data is linearly separable
- Sensitive to outliers
  - Sometime, it is impossible to find a hard margin that will generalize well

Soft margin classification trades margin violations for better generalization

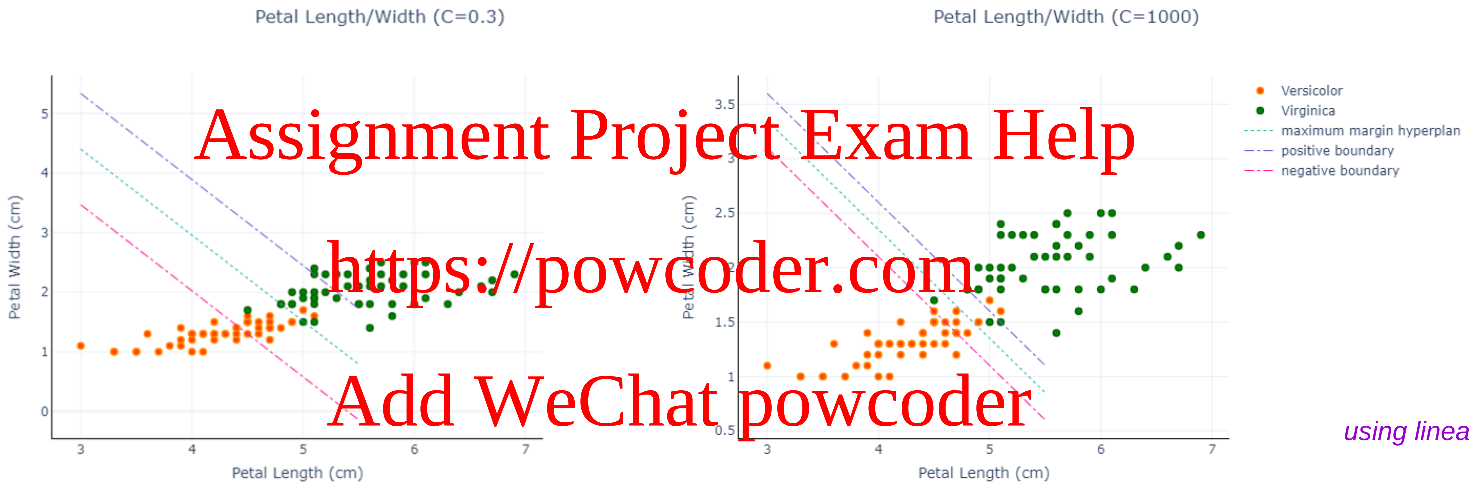


SOFT MARGIN / CONSTRAINT

data point is allowed to appear in the street implying that misclassification is allowed

- To avoid the issues with hard margin classification, a more flexible soft margin classification is introduced
- The objective is to find a good balance between keeping the street as large as possible and limiting the margin violations
- Samples may end up in the middle of the street or even on the wrong side, allowing misclassification

The C hyperparameter is used to control error by specifying a mis-classification penalty

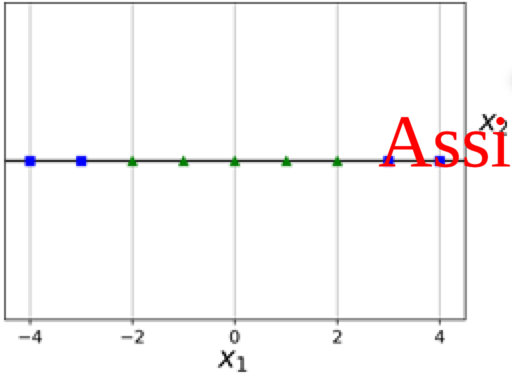


using linear kernel

- C is a hyperparameter for SVM
  - Setting it to a low value, we might end up having a lot of margin violations but will probably generalize better
  - Setting it to a high value, we might get less margin violations but the model may not generalize well
- Reducing C can regularize the model to avoid overfitting

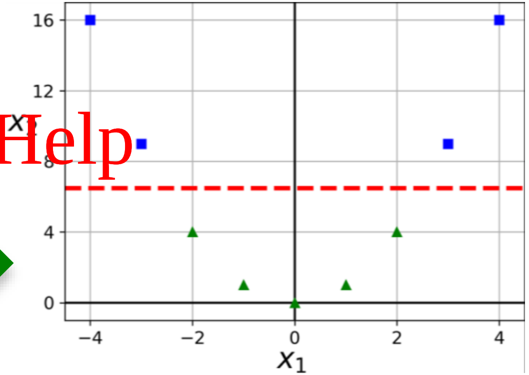
Nonlinear SVM Classification

Features can be added to make a dataset linearly separable



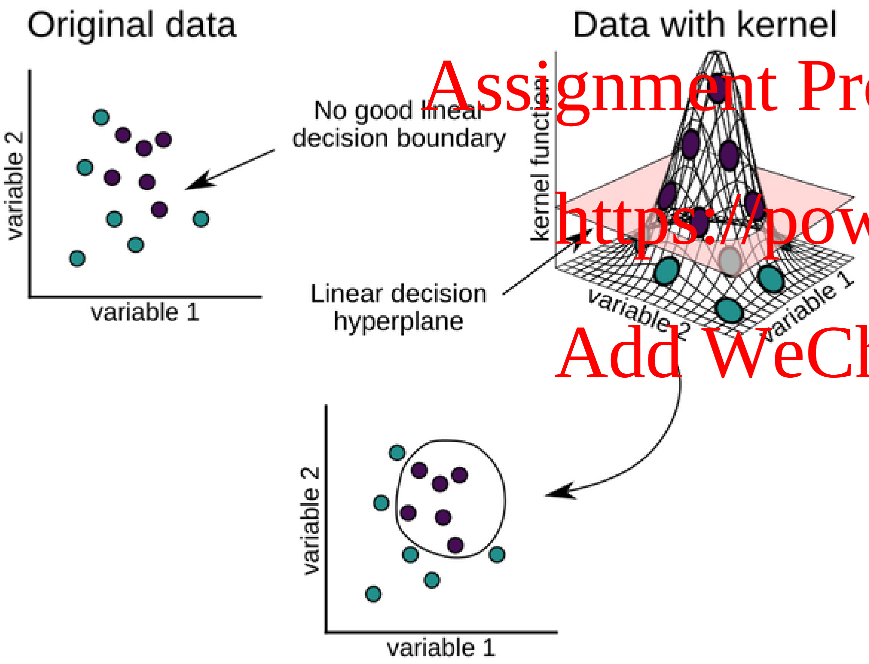
data points not **linearly separable**

add feature x_2, which is the square of x_1 ($x_2 = x_1^2$) to make the data points linearly separable



- Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly separable
- One approach to handling nonlinear datasets is to add more features, such as polynomial features, in some cases this can result in a linearly separable dataset

A kernel function "adds" features by using a similarity function over a landmark and each existing data point

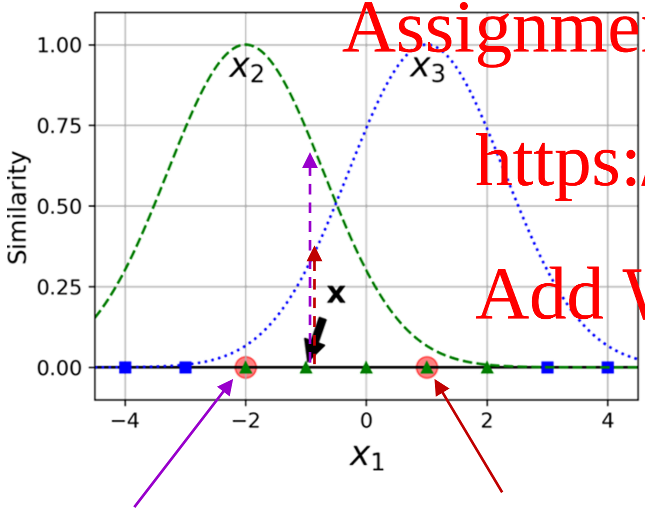


- Adding polynomial features significantly increases the complexity of ML algorithms (SVM & others), which hurts model performance
- When using SVM, **kernel functions** can be applied to get the same result as if many polynomial features were added to the model, even with very high-degree polynomials, without actually having to add them and therefore avoiding the combinatorial explosion of features

The Radial Basis Function (RBF) introduces a new feature having values between 0 and 1

x_2 is a new feature obtained by applying $\emptyset_\gamma(x, l_1)$ over the existing data points

x_3 is a new feature obtained by applying $\emptyset_\gamma(x, l_2)$ over the existing data points



landmark l_1

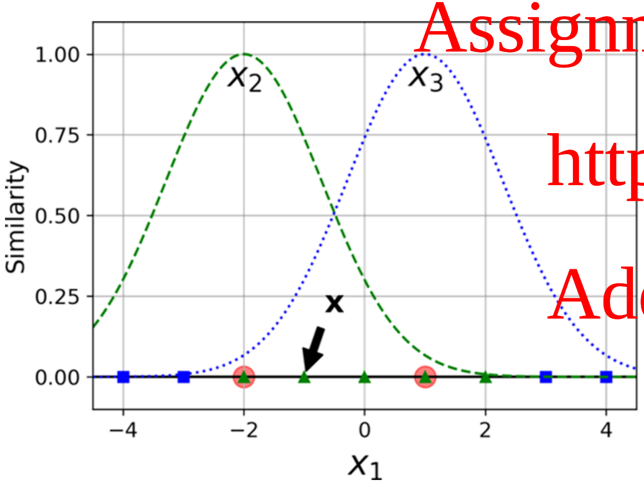
landmark l_2

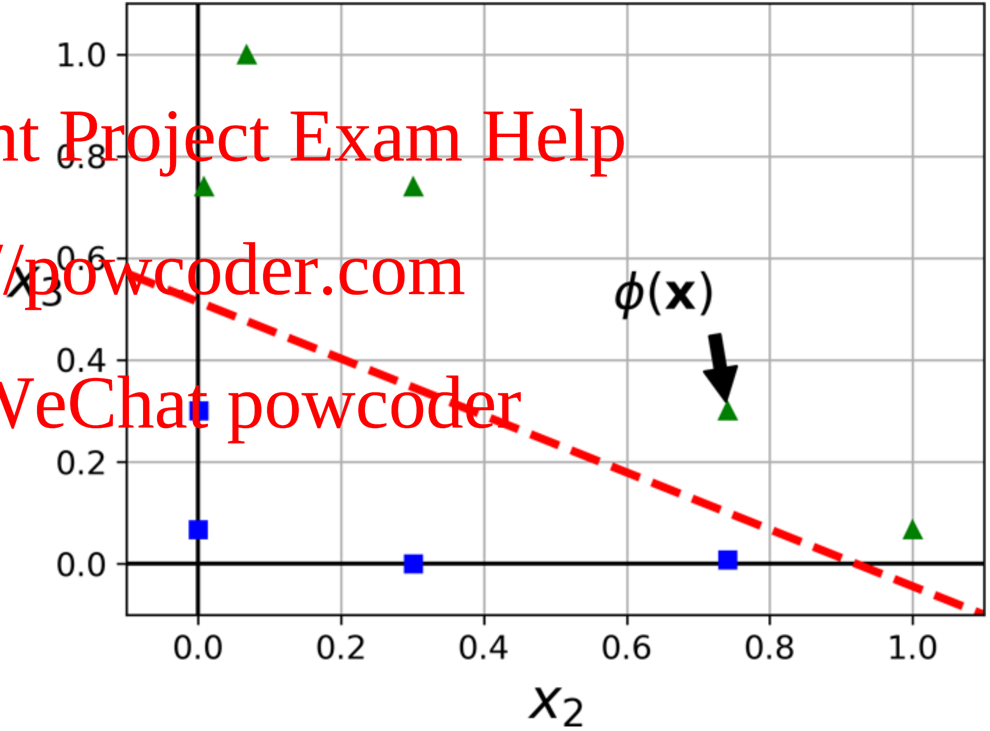
input x_1 has a 1D feature space

$$\emptyset_\gamma(x, l) = \exp(-\gamma\|x - l\|^2)\, where\ \gamma = \frac{1}{2\sigma^2}$$

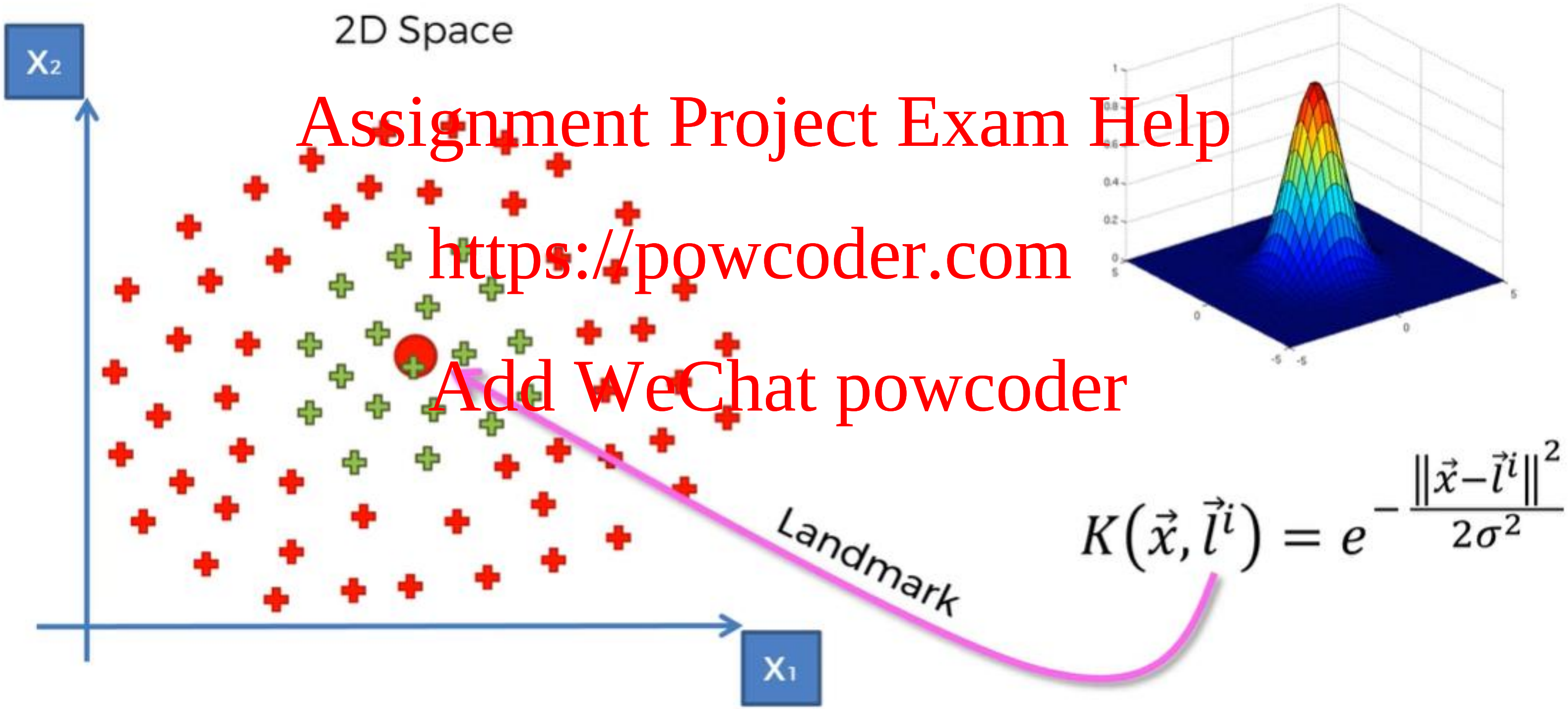
- The RBF is a bell-shaped function measuring the similarity between a landmark point (i.e. l) and any existing data point (e.g. x)
  - $\emptyset_\gamma(x, l) = 0$ indicates the data point x is far from the landmark point l
  - $\emptyset_\gamma(x, l) = 1$ indicates the data point x is at the landmark point l
- γ is a hyperparameter and can be seen as the inverse of the radius of influence of data points selected by the model as support vectors
  - It can be perceived as deciding how much curvature we want in a decision boundary (i.e. high γ means more curvature)

The transformed dataset, dropping the original feature, is linearly separable



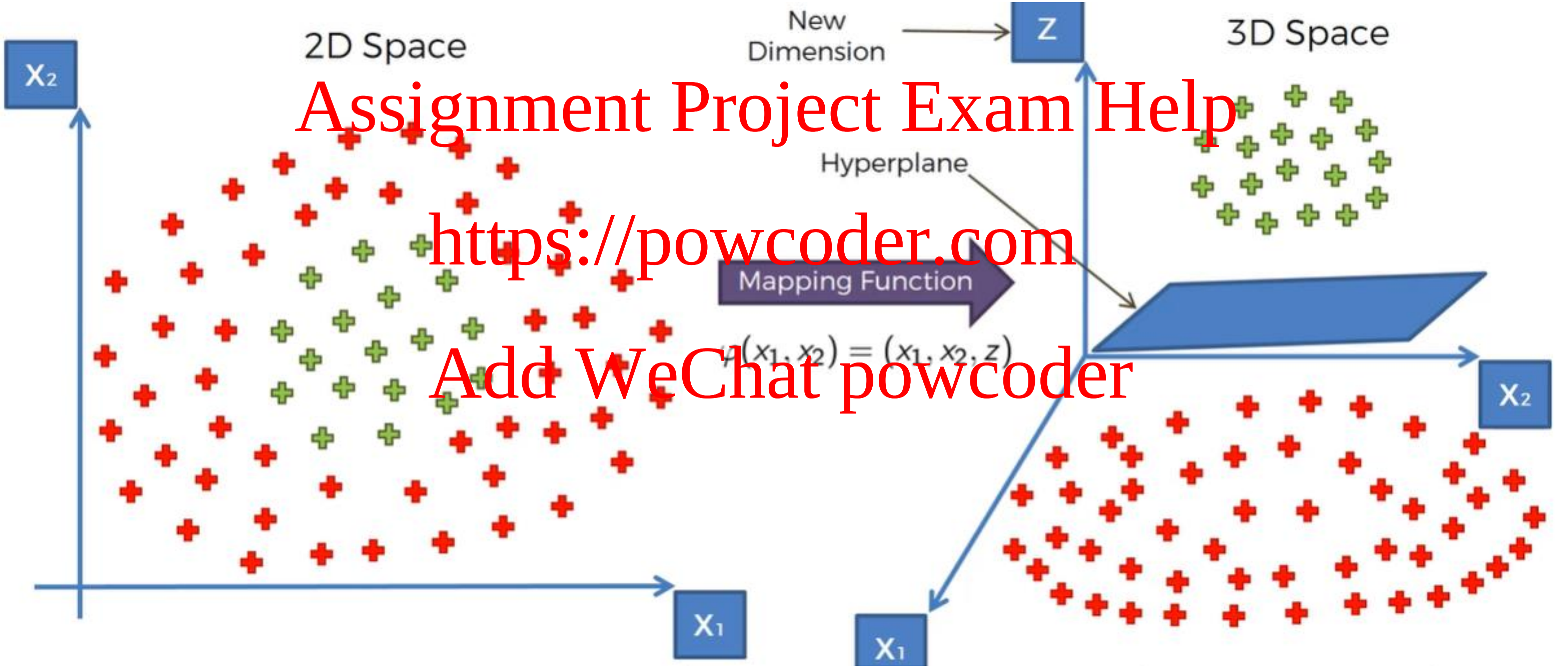


Setting the centroid of the data points as the landmark and then uplifting the data points around the landmark

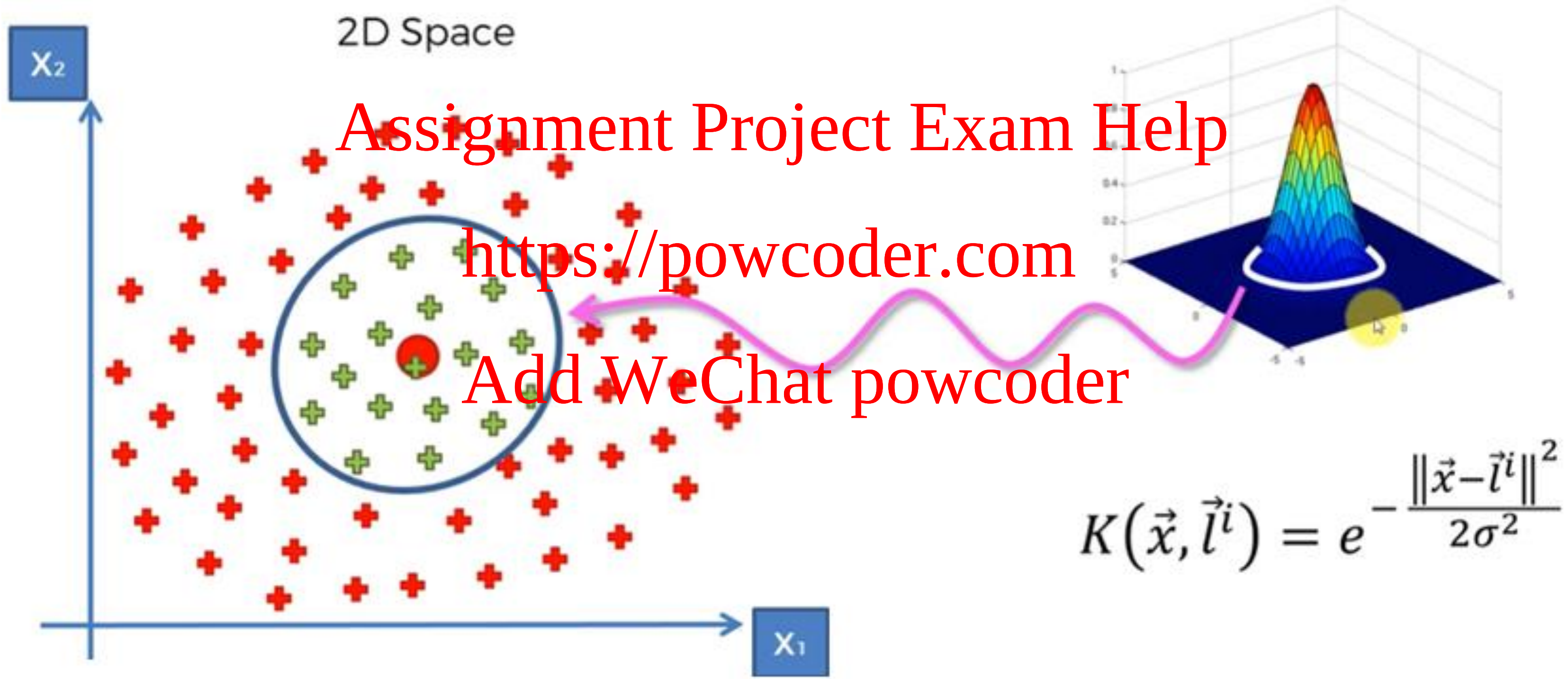


$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

The hyperplane is chosen in the 3D space

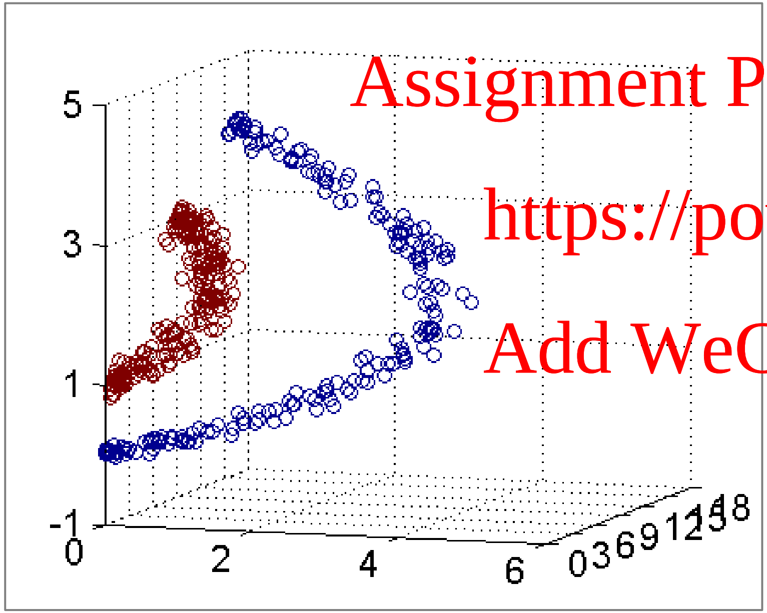


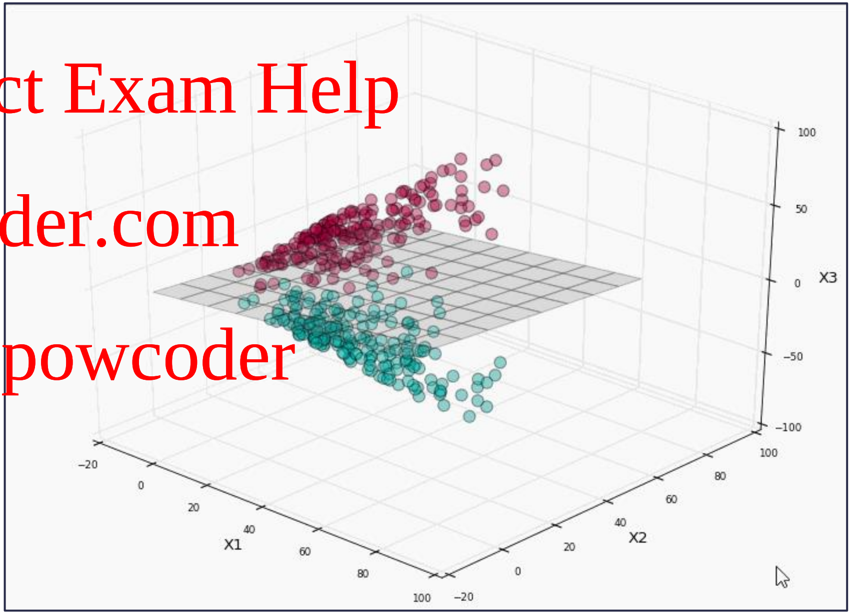
The hyperplane therefore provides a decision boundary for the original dataset



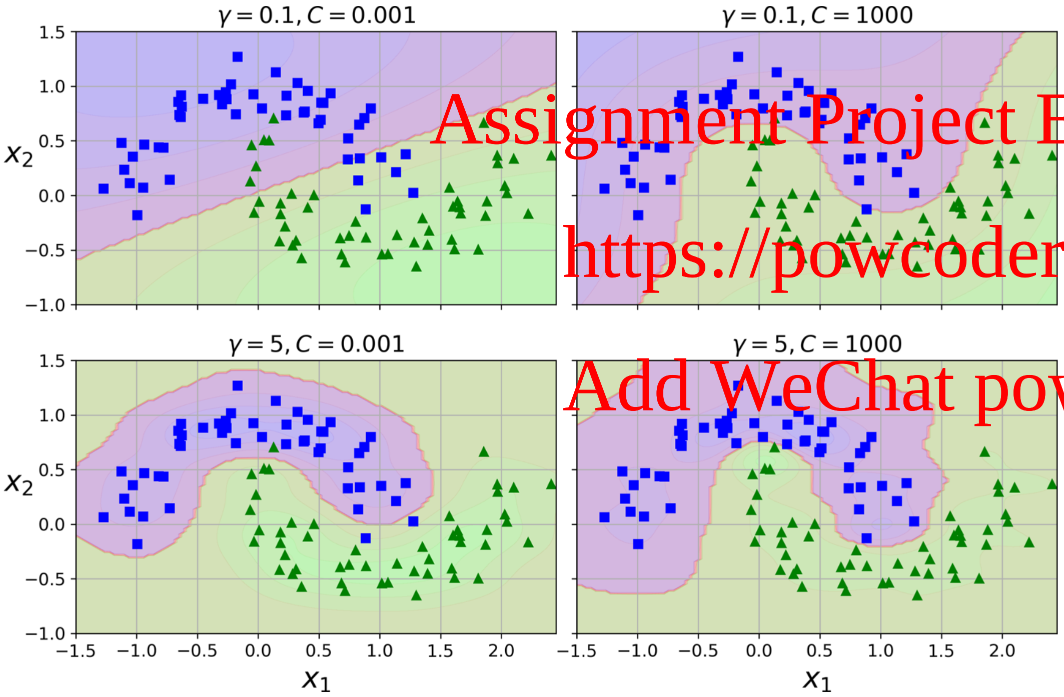
$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^{2}}{2\sigma^{2}}}$$

Transforming the training dataset into a linear separable dataset is the objective of the kernel trick





When a model is overfitting/underfitting, γ should be reduced/increased



- Increasing gamma makes the bell-shaped curve narrower
  - Each sample's range of influence is smaller
  - The decision boundary ends up being more irregular, wiggling around individual samples
- A small gamma value makes the bell-shaped curve wider
  - Samples have a larger range of influence, and the decision boundary ends up smoother
- So γ acts like a regularization hyperparameter
  - When overfitting, it should be reduced
  - When underfitting, it should be increased

With so many kernel functions to choose from, how can you decide which one to use?

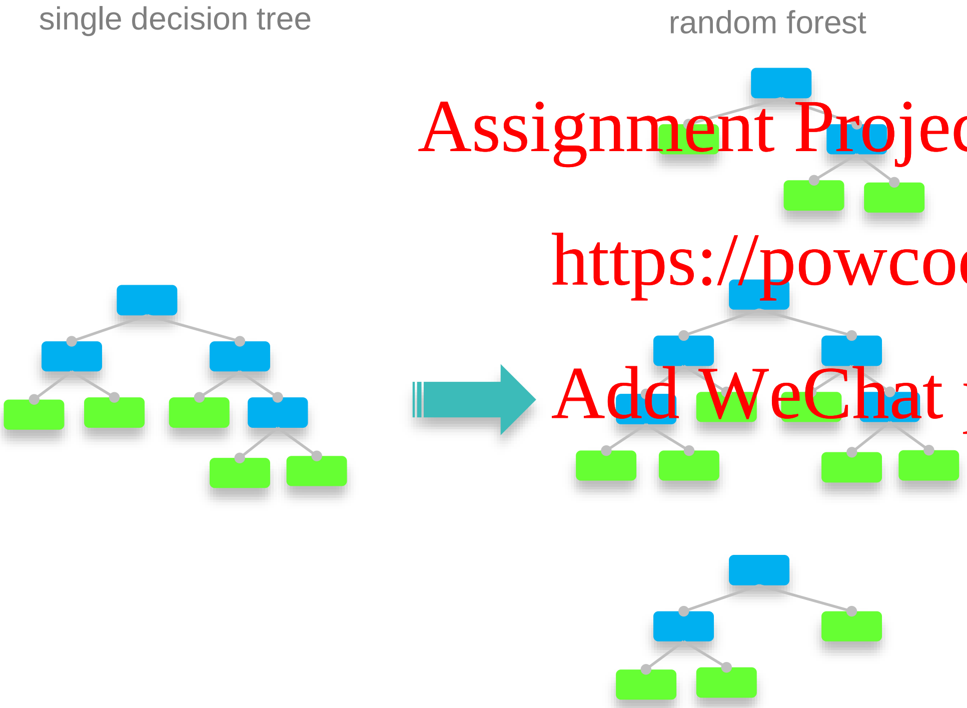
- As a rule of thumb, you should always try the linear kernel first
  - LinearSVC is much faster than SVC(kernel="linear") especially if the training set is very large or if it has plenty of features
- If the training set is not too large, you should also try the RBF kernel - it works well in most cases
- Then if you have spare time and computing power, you can experiment with a few other kernels, using cross-validation and grid search
- You would want to experiment like that especially if there are kernels specialized for your training set's data structure

Support Vector Machine (SVM) in a Nutshell

	Property	Description
1	Feature Data Types	Requires feature scaling.
2	Target Data Types	Categorical or numerical.
3	Key Principles	Find the maximum separation between classes while minimizing the classification error. Using kernel tricks to turn data into linearly separable data.
4	Hyperparameters	With linear and non-linear kernel functions. The C hyperparameter specifying the penalty of mis-classification is needed. The gamma hyperparameter specifying the degree of curvature of the decision boundary is not always needed. With the RBF kernel, both gamma and C are needed.
5	Data Assumptions	No data distributional requirement.
6	Performance	Fairly robust against overfitting, especially in higher dimensional space. Handles non-linear relationships quite well. Can be inefficient to train as well as memory-intensive to run and tune. Does not perform well with large datasets.
7	Accuracy	SVM is known as the most accurate and robust machine learning algorithms.
8	Explainability	Support vectors provide some information about how the classification decision is determined.

Random Forest

Decision trees work great with the data used to create them but not flexible when it comes to classifying new samples

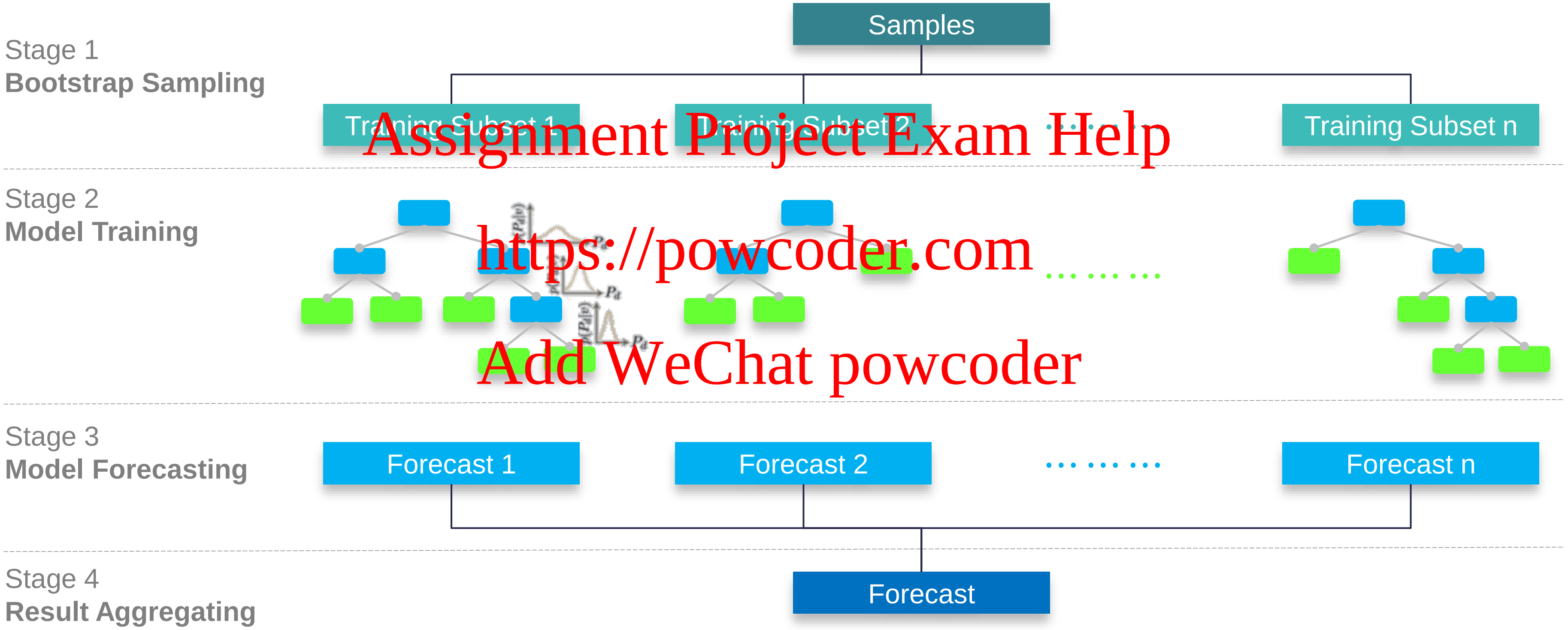


- Decision trees are easy to build, easy to use, and easy to interpret
- Inaccuracy prevents them from being the ideal tool for predictive learning
- They work great with the data used to create them
- However, they are not flexible when it comes to classifying new samples

Random Forest

A random forest is comprised of multiple decision trees. It is said that the more trees it has, the more robust a forest is. A random forest creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests combine the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy



Stage 1
Bootstrap Sampling

Samples

Training Subset 1

Training Subset 2

……

Training Subset n

Stage 2
Model Training

……

Stage 3
Model Forecasting

Forecast 1

Forecast 2

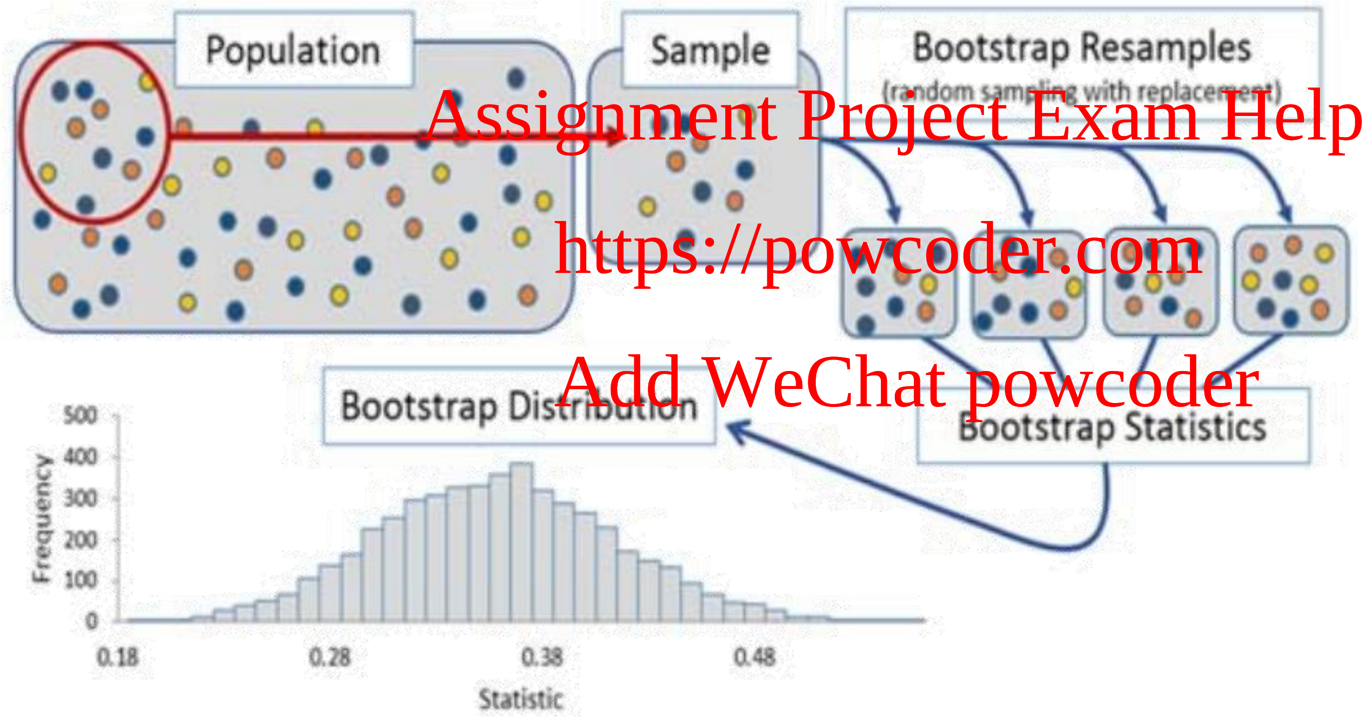
……

Forecast n

Stage 4
Result Aggregating

Forecast

Bootstrapping is a resampling technique used to estimate population statistics by sampling a dataset with replacement



The basic idea of bootstrapping is that inference about a population from sample data can be modelled by resampling the sample data and performing inference about a sample from resampled data

Data subset is created by randomly selecting samples from the sample dataset – bootstrapping with replacement

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

original sample dataset

- A bootstrapped data subset is created by randomly selecting samples from the original sample dataset
- The bootstrapped data subset is of the same size as the original dataset
- The important detail is that it is allowed to pick the same sample more than once

Data subset is created by randomly selecting samples from the sample dataset – bootstrapping with replacement

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

original sample dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes

Data subset is created by randomly selecting samples from the sample dataset – bootstrapping with replacement

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

original sample dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No

Data subset is created by randomly selecting samples from the sample dataset – bootstrapping with replacement

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

original sample dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes

Data subset is created by randomly selecting samples from the sample dataset – bootstrapping with replacement

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

original sample dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

The 4th selected sample is the same as the 3rd one - sampling with replacement is at work here

A decision tree is constructed using a randomly selected subset of the features at each step

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

bootstrapped data subset

randomly selected two features as candidates for the root node of the new decision tree

assuming **Good Blood Circulation** does the best job separating the samples

Good Blood Circulation

root node of the new decision tree

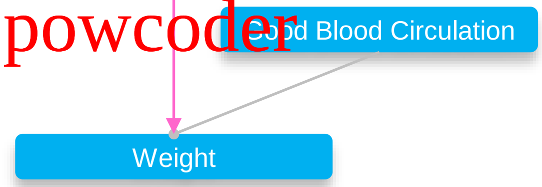
The candidate feature with the best separating power is selected as the decision feature

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

bootstrapped data subset

randomly selected two features as candidates for the next internal node

assuming **Weight** does the best job separating the samples

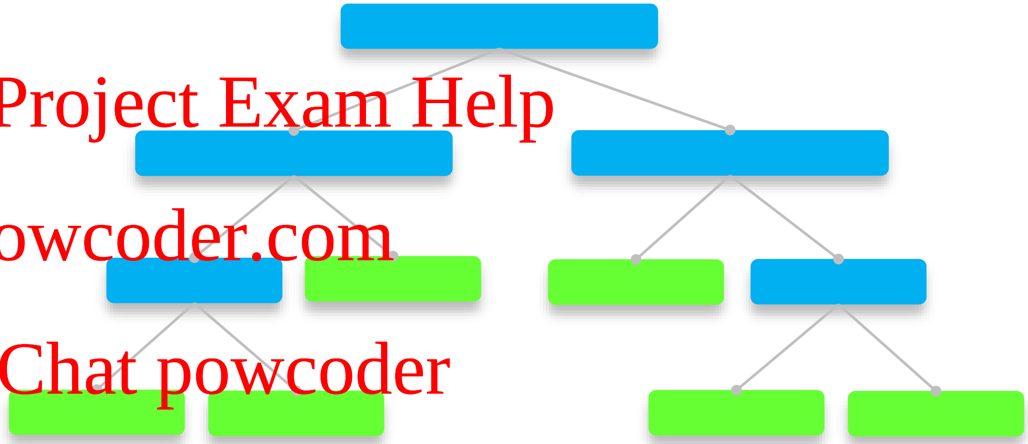


the new decision tree with the root node and one internal node

A decision tree is built as usual but only considering a randomly selected subset of features at each step

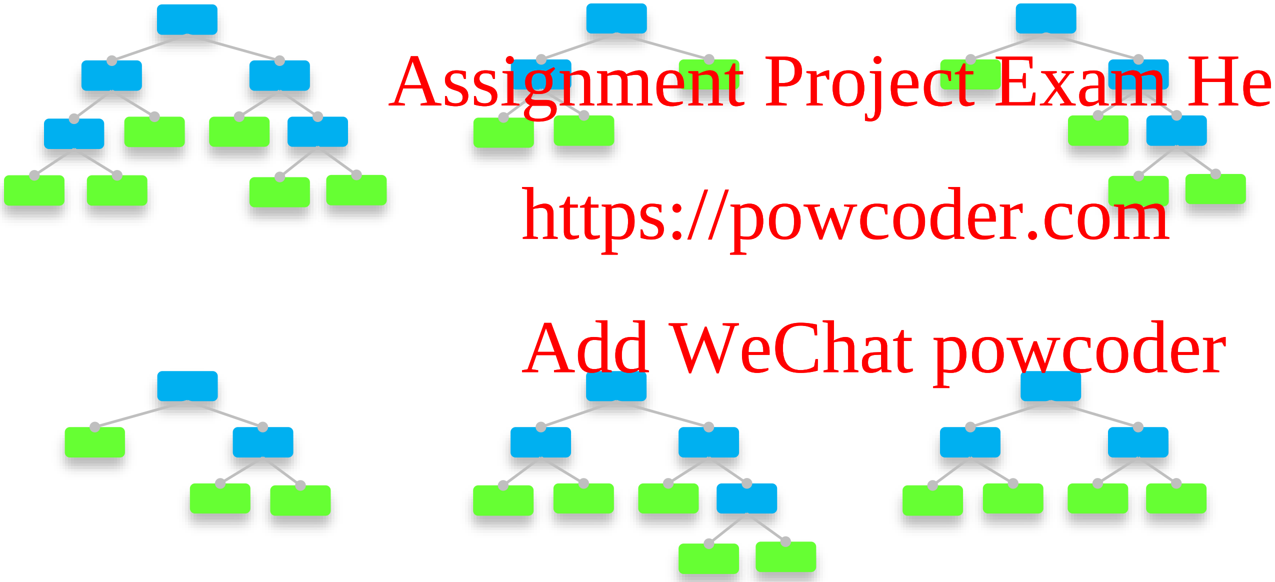
Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

bootstrapped data subset



the new decision tree

Repeatedly make a new bootstrapped dataset and build a tree considering a subset of features at each step

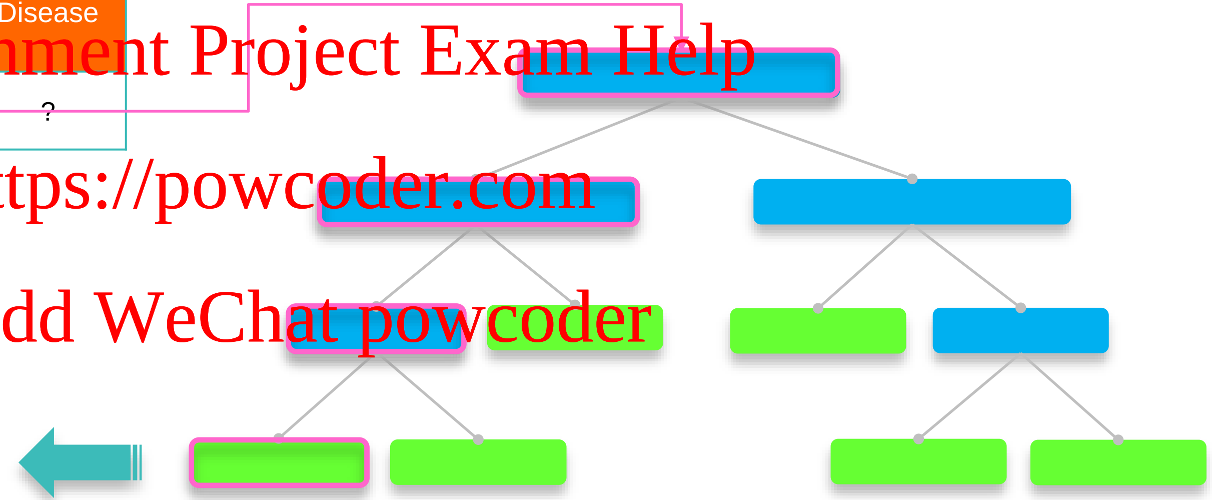


- After building hundreds of decision trees, it results in a wide variety of trees
- The variety is the fundamental element that makes random forests more effective than individual decision trees

New data will be run through the decision trees one by one and the result of each decision tree is recorded

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	?

a new data



run the data down the 1st tree

the 1st tree says YES

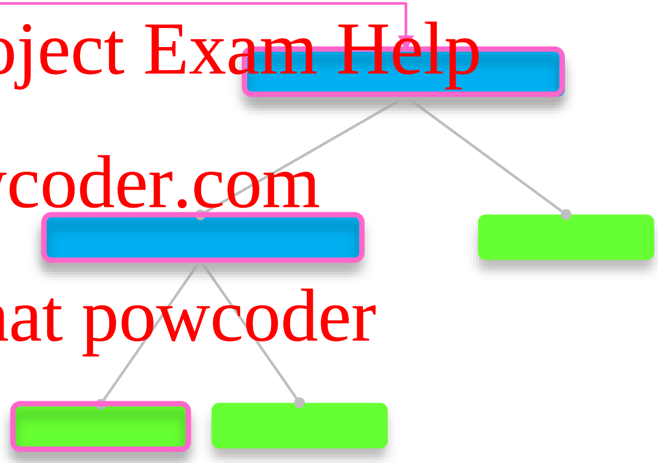
Heart Disease **YES**	Heart Disease **No**
1	0

Each decision tree result is tracked against the prediction classes

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	?

a new data

run the data down the 2nd tree



the 2nd tree says YES

Heart Disease **YES**	Heart Disease **No**
2	0

The prediction outcome is determined by the votes of all decision trees in the forest

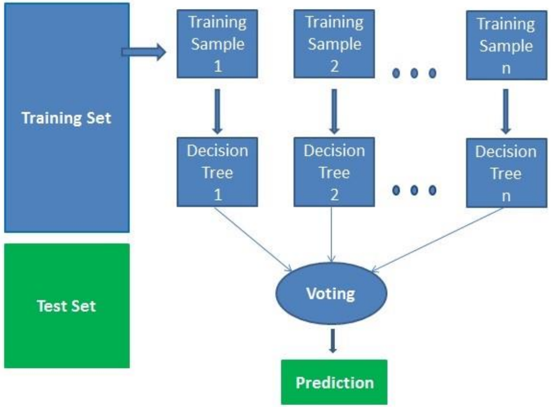
Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	**YES**

a new data

Heart Disease **YES**	Heart Disease **No**
5	1

- In this case, "YES" received the most votes, so the conclusion is that the patient does have heart disease

Ensemble Method



- Random forest is technically an ensemble method based on the divide-and-conquer approach
- Each decision tree in the forest is generated based on a random sample from the training dataset selected using information gain, gain ratio, and Gini index for each feature
- In a classification problem, each tree votes and the most popular class is chosen as the final result
- In the case of regression, the average of all the tree outputs is considered as the final result
- It is simpler and more powerful compared to the other non-linear classification algorithms

Bagging uses the same algorithm for every predictor but using different random subsets of the training dataset

- Bagging / Bootstrap aggregating uses the same algorithm for each predictor but using different random subsets of the training dataset to allow for a more generalised result

- Subsets can be created with or without replacement
  - With replacement, some samples may be present & repeated in more than one subset
  - Without replacement, all samples in each subset are unique with no repeated sample

- Once all the predictors are trained, the ensemble can make a prediction for a new instance by aggregating the predicted values of all trained predictors
- Although each individual predictor has a higher bias than if it were trained on the original dataset, the aggregation allows the reduction of both bias & variance

Typically, about 1/3 of the original data does not end up in the bootstrapped dataset – the **Out-of-Bag** dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

original sample dataset

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

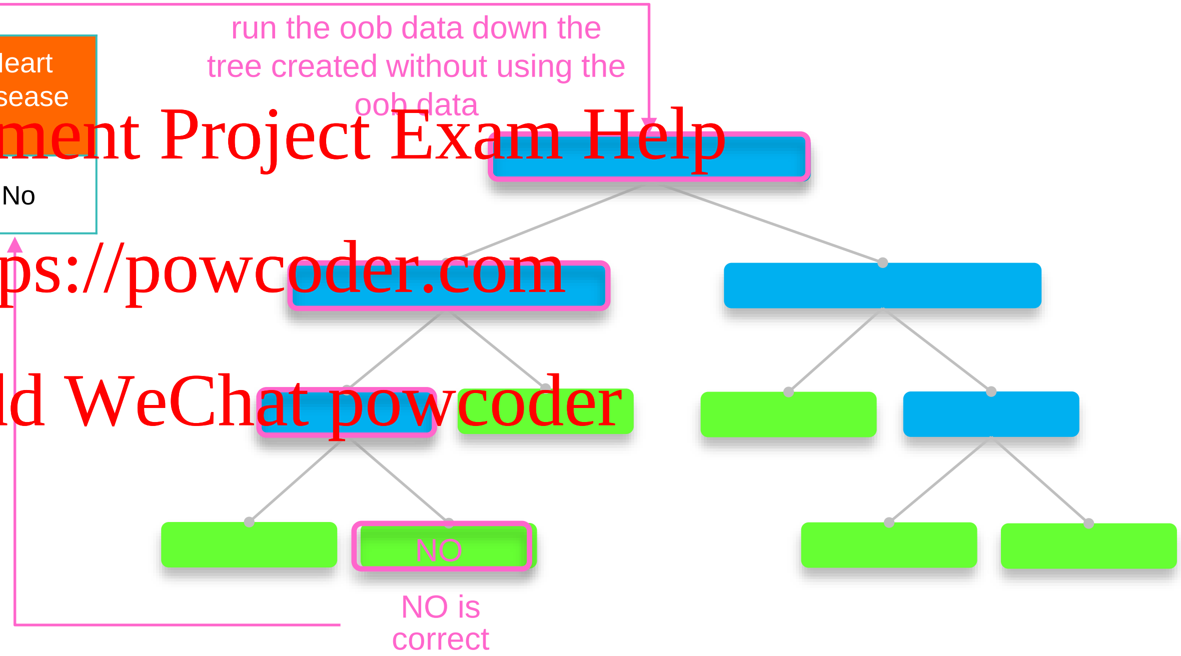
bootstrapped dataset

this sample is not included in the bootstrapped dataset so will be considered as a sample in the **Out-Of-Bag** dataset

The OOB dataset was not used to create this decision tree so it can be run through the decision tree for validation

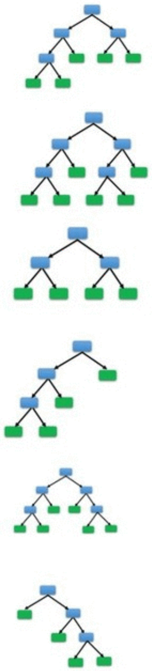
Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

out-of-bag dataset


run the oob data down the tree created without using the oob data

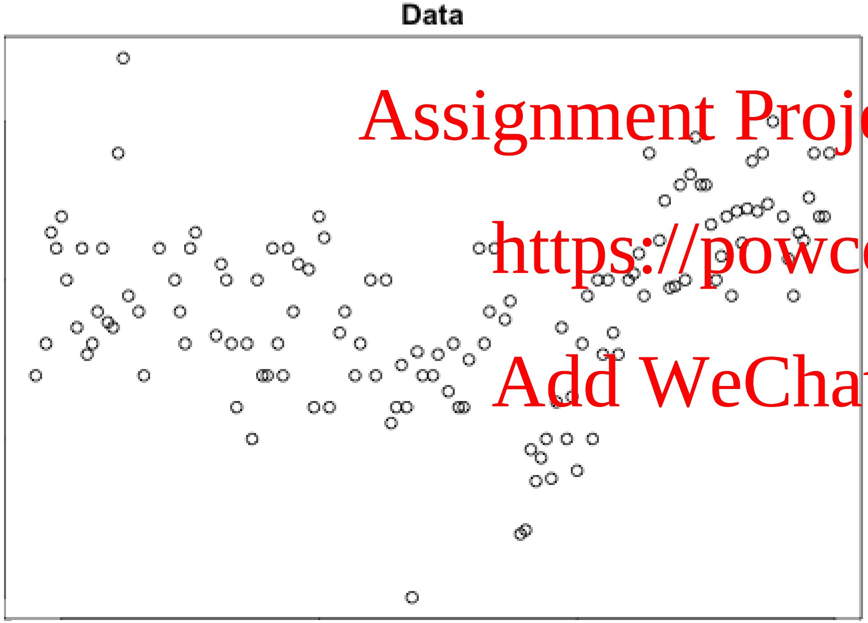
NO is correct

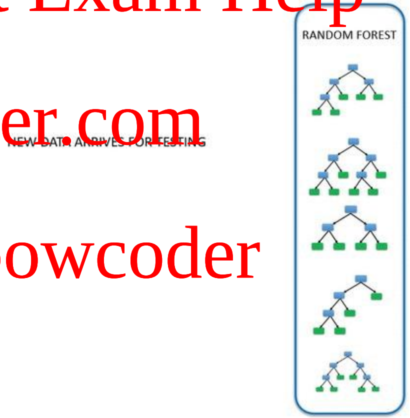
Continuing running this out-of-bag sample through all of the other trees that were built without it & aggregate the results



Random Forest in Action!!!

Accuracy of the model can be determined by running the out-of-bag dataset against all applicable decision trees





MISTAKES	CORRECT PREDICTIONS
0	0

The proportion of Out-Of-Bag samples that are incorrectly classified is the **Out-Of-Bag Error**

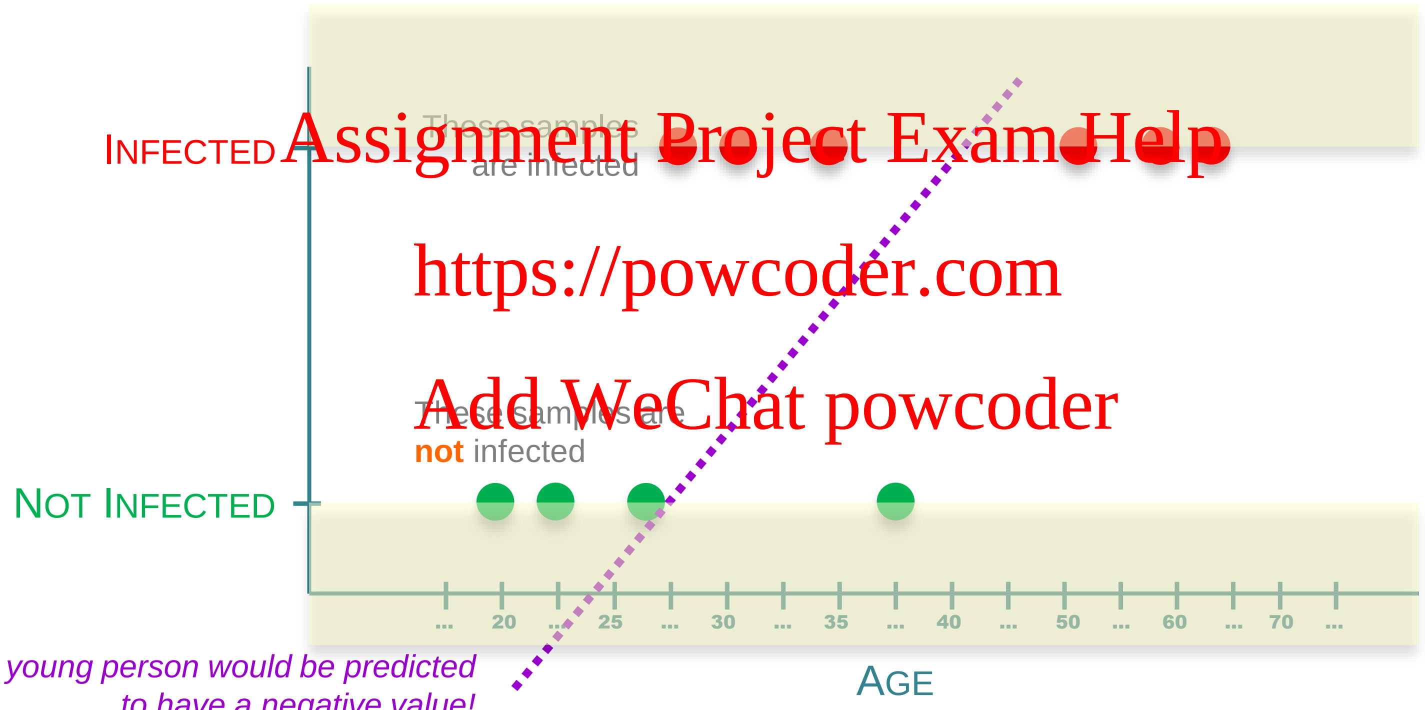
Random Forest Models in a Nutshell

	Property	Description
1	Feature Data Types	Numerical.
2	Target Data Types	Categorical or Numerical.
3	Key Principles	Extremely flexible & easy to use. Can be used for both classification & regression problems. Can handle missing values in training and prediction by replacing imputing continuous features with median values and categorical values using the proximity weighted average of missing values.
4	Hyperparameters	No of trees in the forest. Quality function for internal node split. Minimum number of samples required to split an internal node. Minimum number of samples required to be a leaf node. Maximum number of leaf nodes. Maximum depth of the tree.
5	Data Assumptions	Data scaling is expected.
6	Performance	Overfitting does not occur because of the use of the average of predictions and hence cancels out the biases. Slow in generating predictions due to the number of decision trees involved.
7	Accuracy	Considered as a very accurate and robust method because of the number of decision trees taking part in the prediction. Simpler and more powerful than other non-linear classification algorithms.
8	Explainability	Relative feature contribution to the prediction. Less interpretable than simple decision tree.

Logistic Regression

Applying Logistic Regression

Linear regression does not always predict a value that falls within the expected range



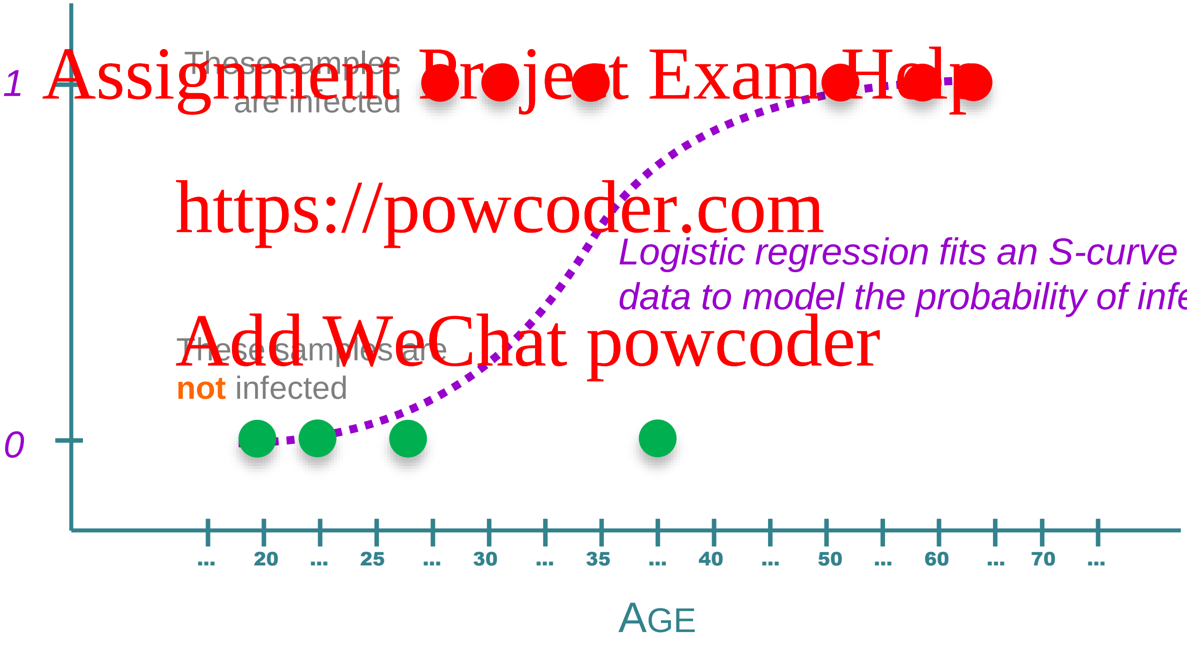
large variability in the outcome at all ages

a young person would be predicted to have a negative value!

Logistic Regression

Unlike linear regression, logistic regression does not try to predict the value of a numeric variable given a set of inputs. Instead, the output of logistic regression is the probability of a given input point belonging to a specific class. The output of logistic regression always lies in [0,1].

The sample contains people of different ages and each person is either infected or not infected



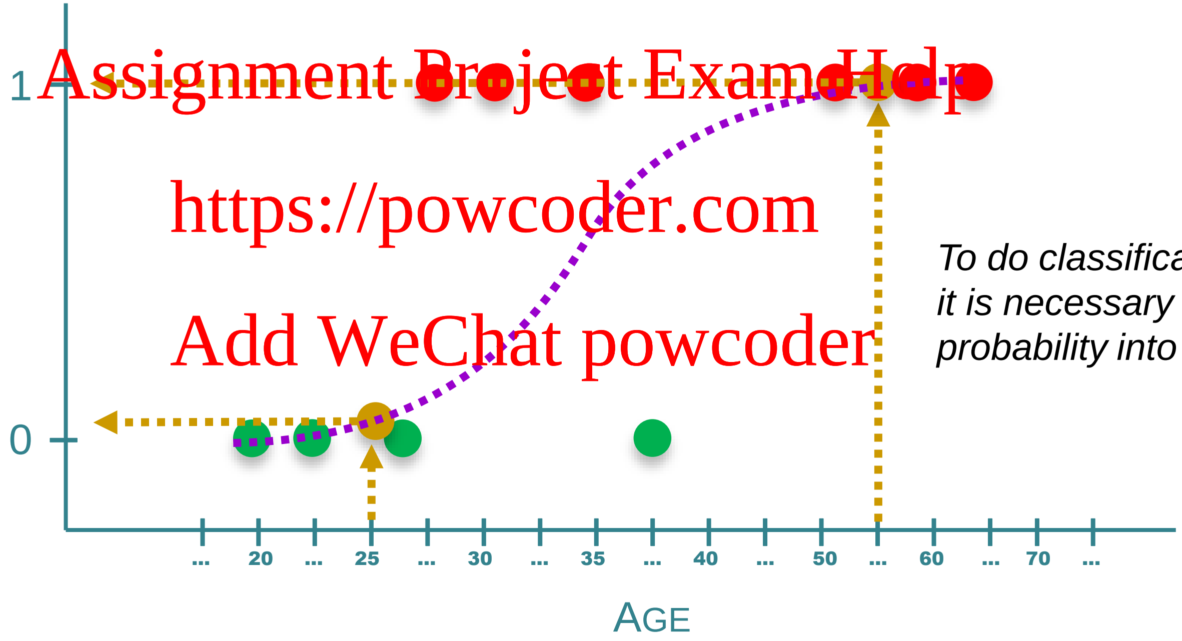
These samples are infected

These samples are **not** infected

Logistic regression fits an S-curve to the data to model the probability of infection

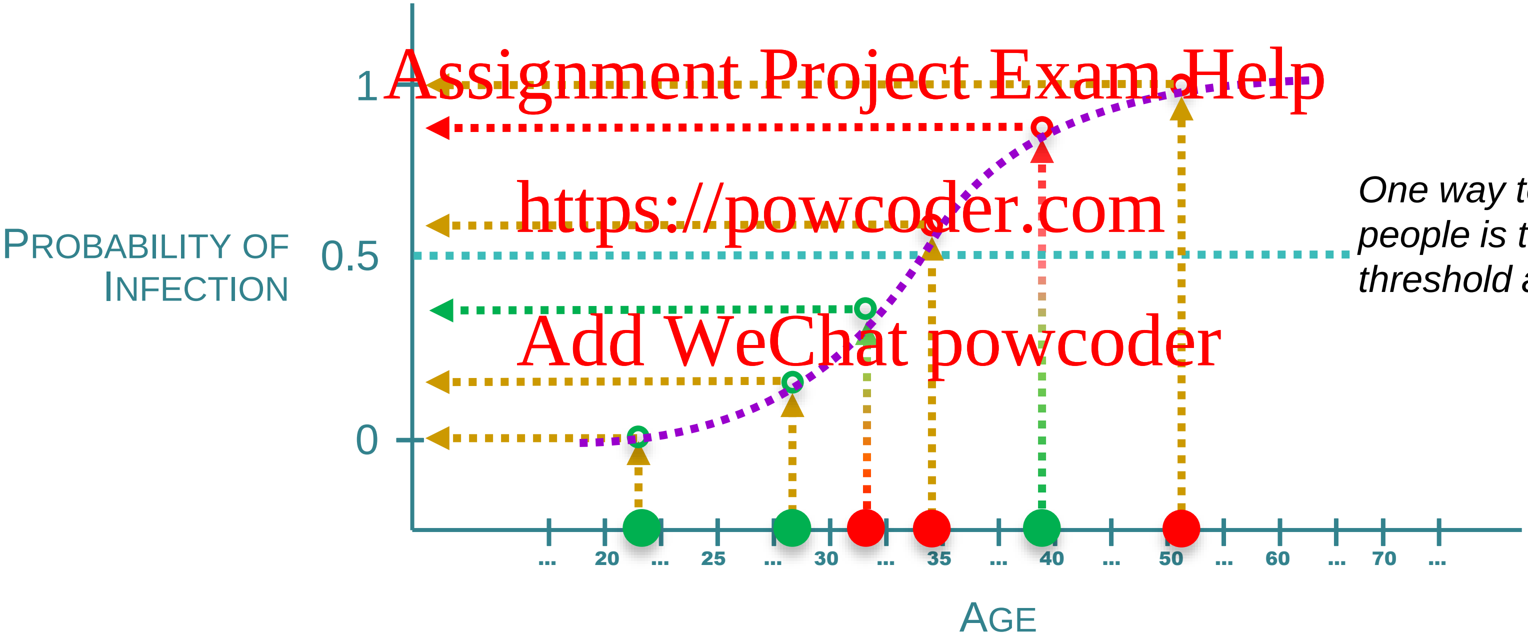
The logistic regression predicts the probability of a person being infected based on the person's age

When doing logistic regression, the y-axis is converted to the probability that a person is infected



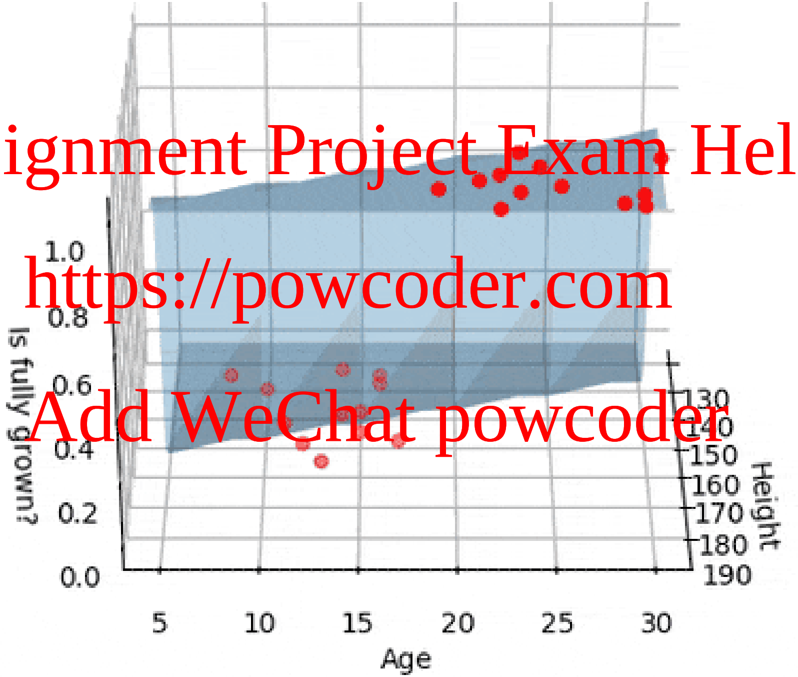
To do classification, it is necessary to turn probability into classification

People with a probability greater than the threshold will be classified as infected; otherwise, not infected

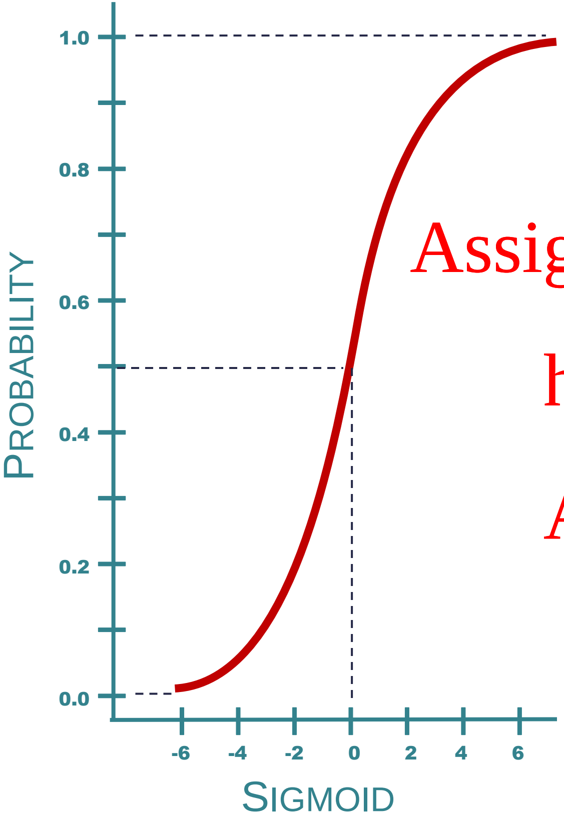


One way to classify people is to set a threshold at ***0.5***

Logistic regression is generalised to predict using multiple variables



Logistic Regression S-Curve



The logistic function belongs to a class of functions called the sigmoid function

$$\text{Probaility} = \sigma(z) = \frac{e^{z}}{1+e^{z}} = \frac{1}{1+e^{-z}}$$

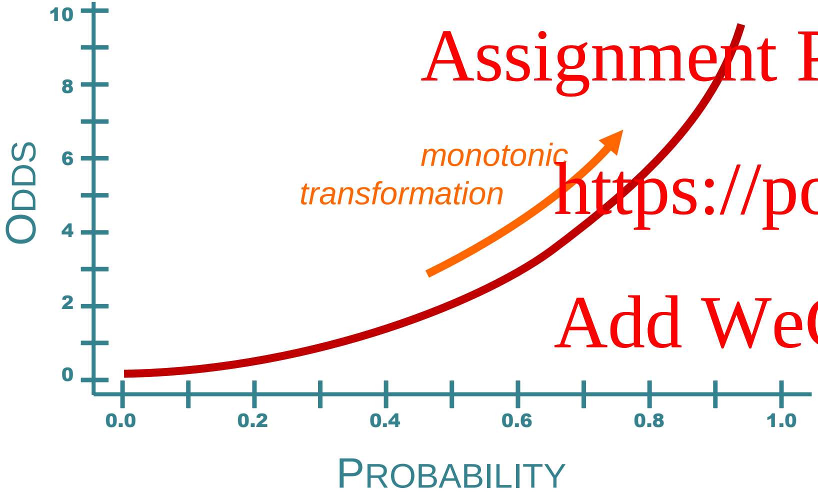
$$\text{where } z = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

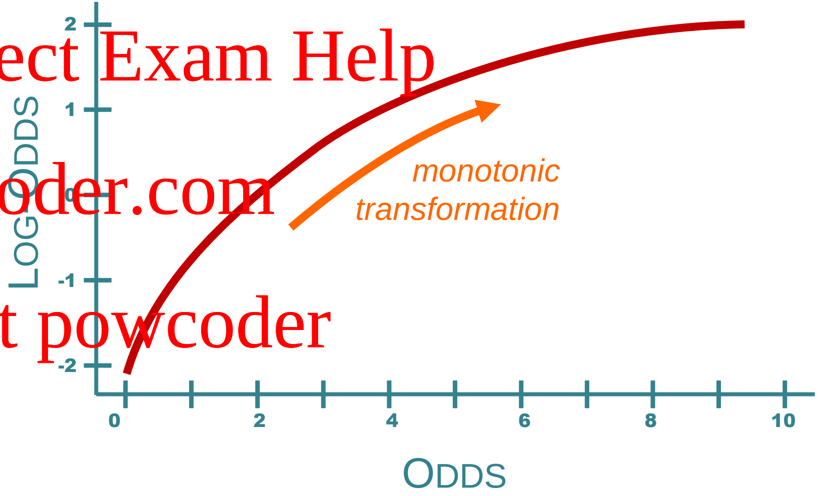
- $\sigma(z)$ is close to 1 when z is big
- $\sigma(z)$ is close to 0 when z is small
- The change in $\sigma(z)$ per unit change in z becomes progressively smaller as $\sigma(z)$ gets close to 0 and 1

Transformations make likelihood measure symmetrical (easy to interpret), more succinct & with unrestricted range

$$\text{Odds}(p) = \frac{\text{chances of something happening}}{\text{chances of something not happening}}$$

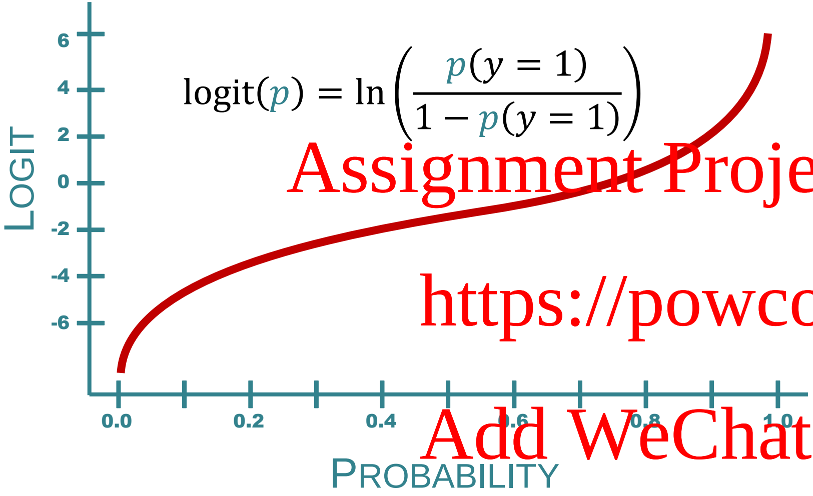
$$\text{Log}-\text{Odds}\big(\text{Odd}(p)\big) = \ln\big(\text{Odds}(p)\big)$$



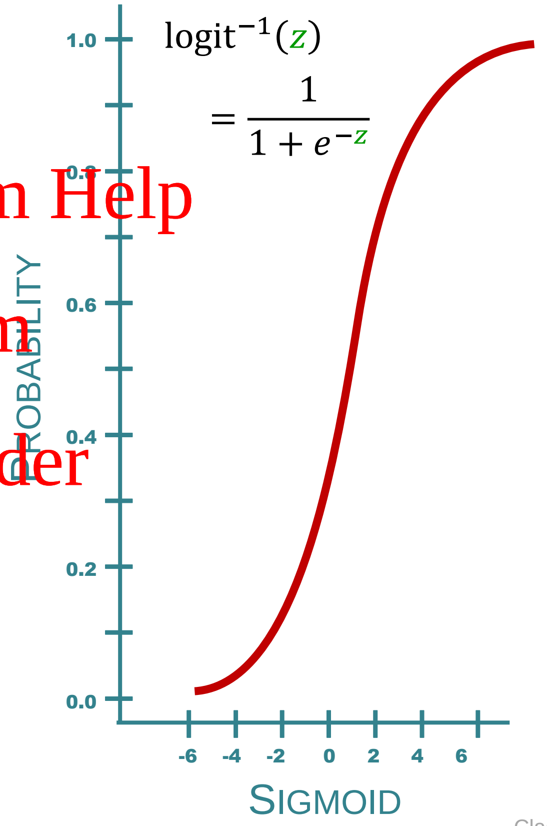


- A change in a feature by one unit changes the odds by a factor of e^{β_i} (i.e. e to a constant power that equals to the coefficient of that feature)

The logistic sigmoid function can be obtained by taking the inverse of the logit function



$$\text{logit}(p) = \ln\left(\frac{p(y=1)}{1-p(y=1)}\right)$$



$$\text{logit}^{-1}(z) = \frac{1}{1+e^{-z}}$$

- Flipping the axes, the logit curve becomes the sigmoid curve
- The sigmoid function is the inverse of the logit function

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

Logistic regression can be perceived as regressing against the log of the odds that the class is 1

chances of something not happening

chances of something happening

$$\boxed{p(y = 0|z)} = 1 - \boxed{p(y = 1|z)}$$

$$= 1 - \frac{1}{1 + e^{-z}}$$

$$= \frac{e^{-z}}{1 + e^{-z}}$$

$$odds = \frac{chances\ of\ something\ happening}{chances\ of\ something\ not\ happening}$$

$$\boxed{\ln\left(\boxed{\frac{p(y = 1|z)}{p(y = 0|z)}}\right)} = \ln\left(\frac{\frac{1}{1 + e^{-z}}}{\frac{e^{-z}}{1 + e^{-z}}}\right)$$

$$= \ln(e^{z}) = z$$

the logit transformation is central to logistic regression

Finding the Best S-Curve

Likelihood measures the goodness of fit of a model to a sample of data for given values of the unknown parameters

- Likelihood is formed from the joint probability distribution of the sample data, but viewed and used as a function of the unknown parameters only, thus treating the independent variables as fixed at the observed values
- The likelihood function describes a hypersurface whose peak, if it exists, represents the combination of model parameter values that maximize the probability of drawing the sample obtained

$$\left.\begin{aligned}\text{Likelihood} &= p(data|parameters) = p(y|z) \\ &= \prod_{i=1}^{N} p(y_i = 1|z)^{y_i} \cdot p(y_i = 0|z)^{1-y_i}\end{aligned}\right\}$$

best fit means
maximum likelihood

Performing gradient descent on the negative log-likelihood will get us the optimal β values that minimizes the total loss

$\text{Negative Log}-\text{Likelihood}$

$$= -\ln\left(\prod_{i=1}^{N} p(y_i = 1|z)^{y_i} \cdot p(y_i = 0|z)^{1-y_i}\right)$$

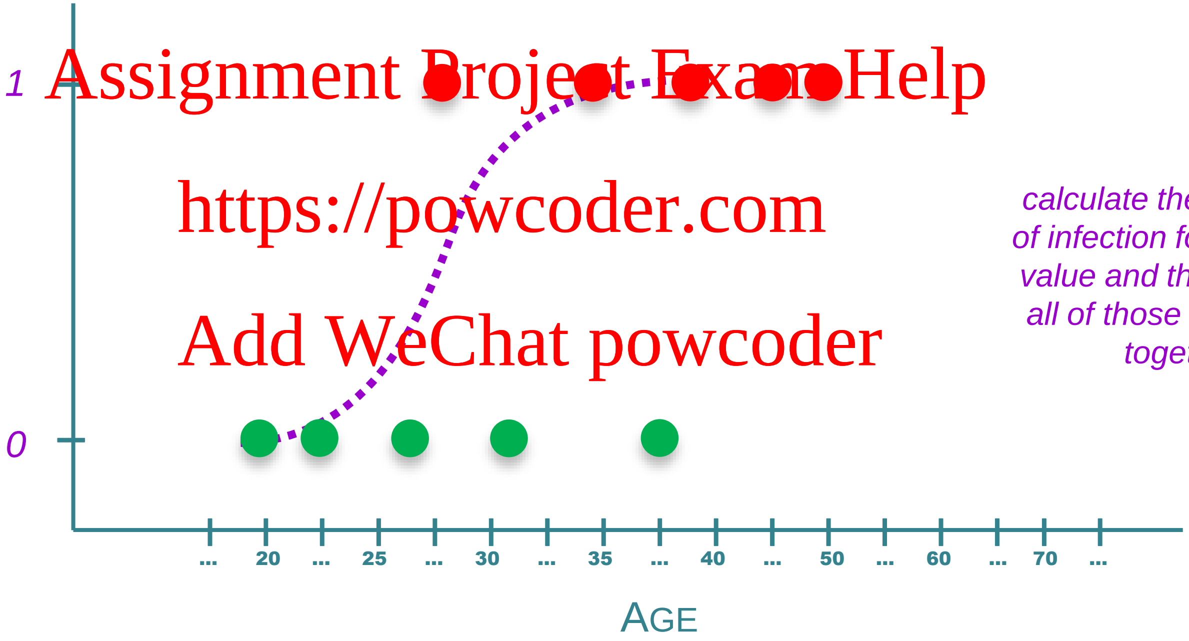
$$= -\sum_{i=1}^{N} y_i \cdot \ln\big(p(y_i = 1|z)\big) + (1 - y_i) \cdot \ln\big(p(y_i = 0|z)\big)$$

$$= -\sum_{i=1}^{N} y_i \cdot \ln\left(\frac{1}{1 + e^{-z}}\right) + (1 - y_i) \cdot \ln\left(\frac{e^{-z}}{1 + e^{-z}}\right)$$

$$= -\sum_{i=1}^{N} -z - \ln(1 + e^{-z}) + y_i \cdot z$$

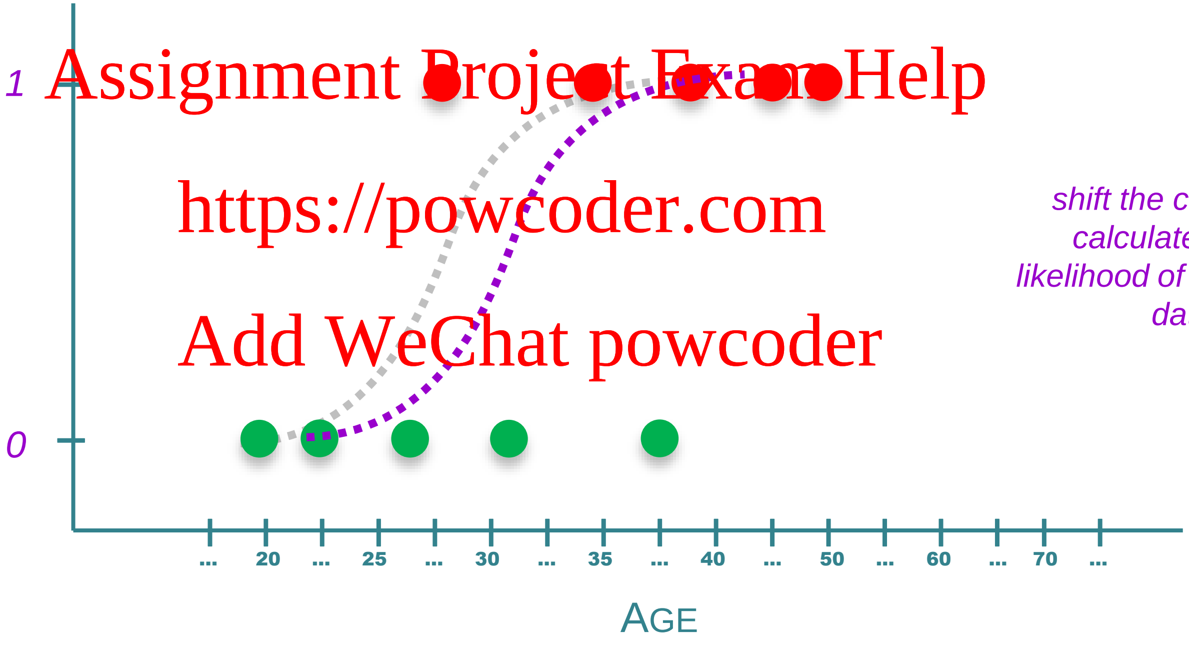
For computational convenience, the maximization of likelihood is usually done by minimizing the negative of the natural logarithm of the likelihood, known as the log-likelihood function

Logistic regression uses maximum likelihood to obtain the curve that fits the sample data best



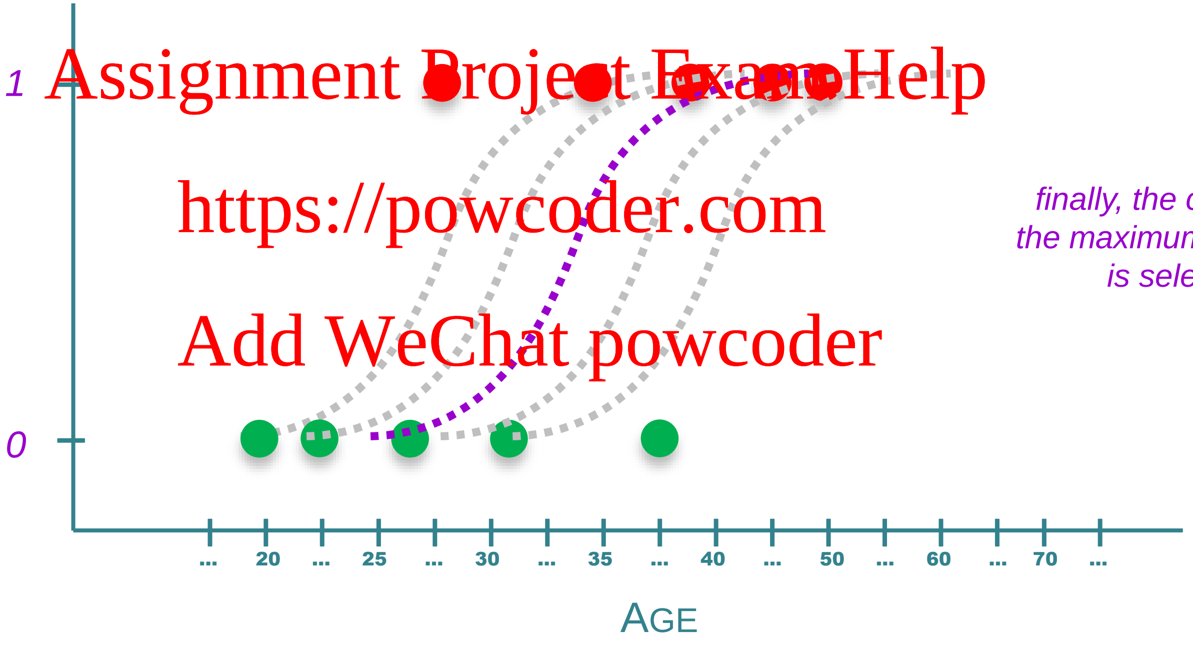
calculate the likelihood of infection for each age value and then multiply all of those likelihoods together

Logistic regression uses maximum likelihood to obtain the curve that fits the sample data best

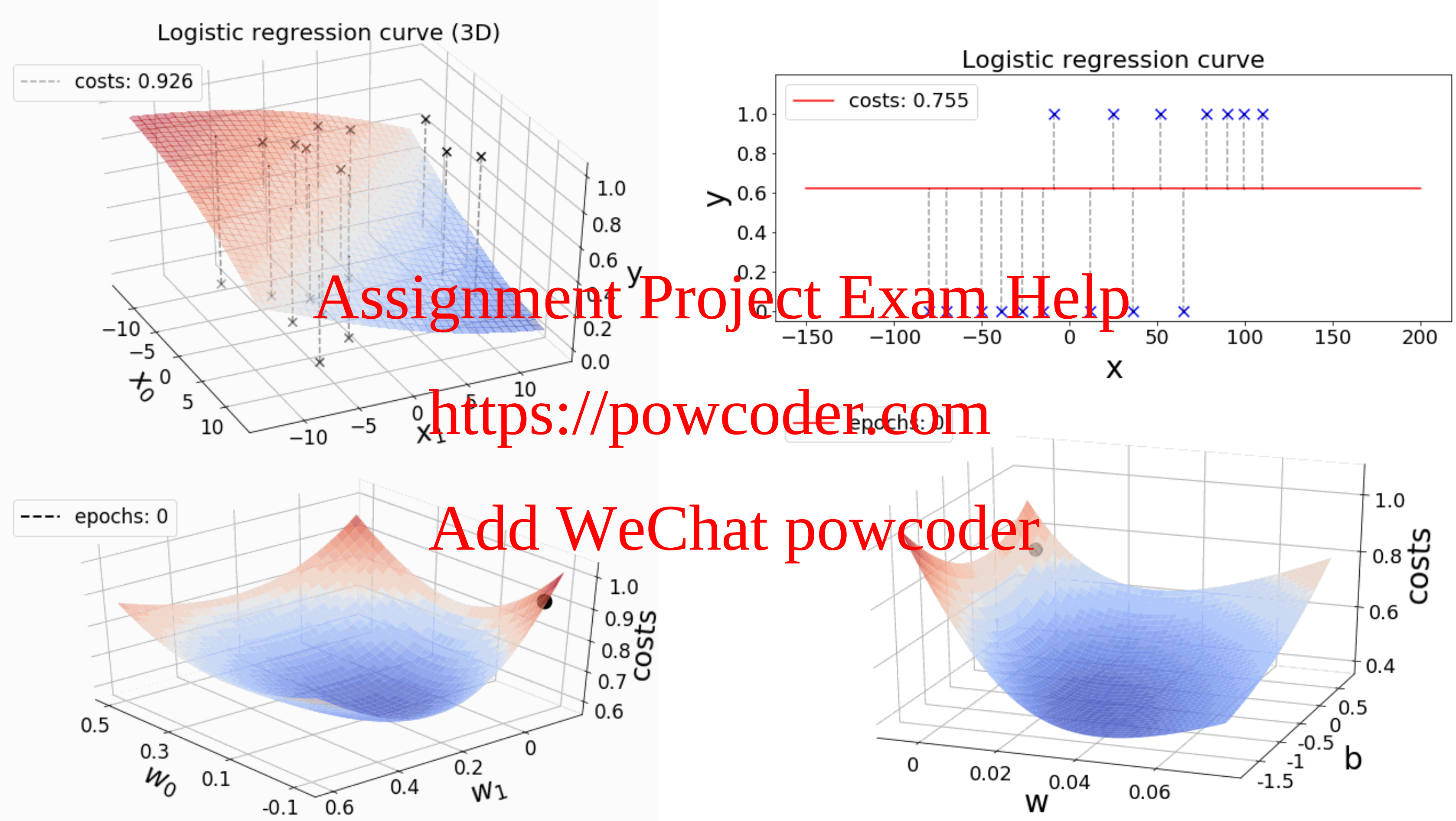


shift the curve and calculate a new likelihood of the sample data

Logistic regression uses maximum likelihood to obtain the curve that fits the sample data best



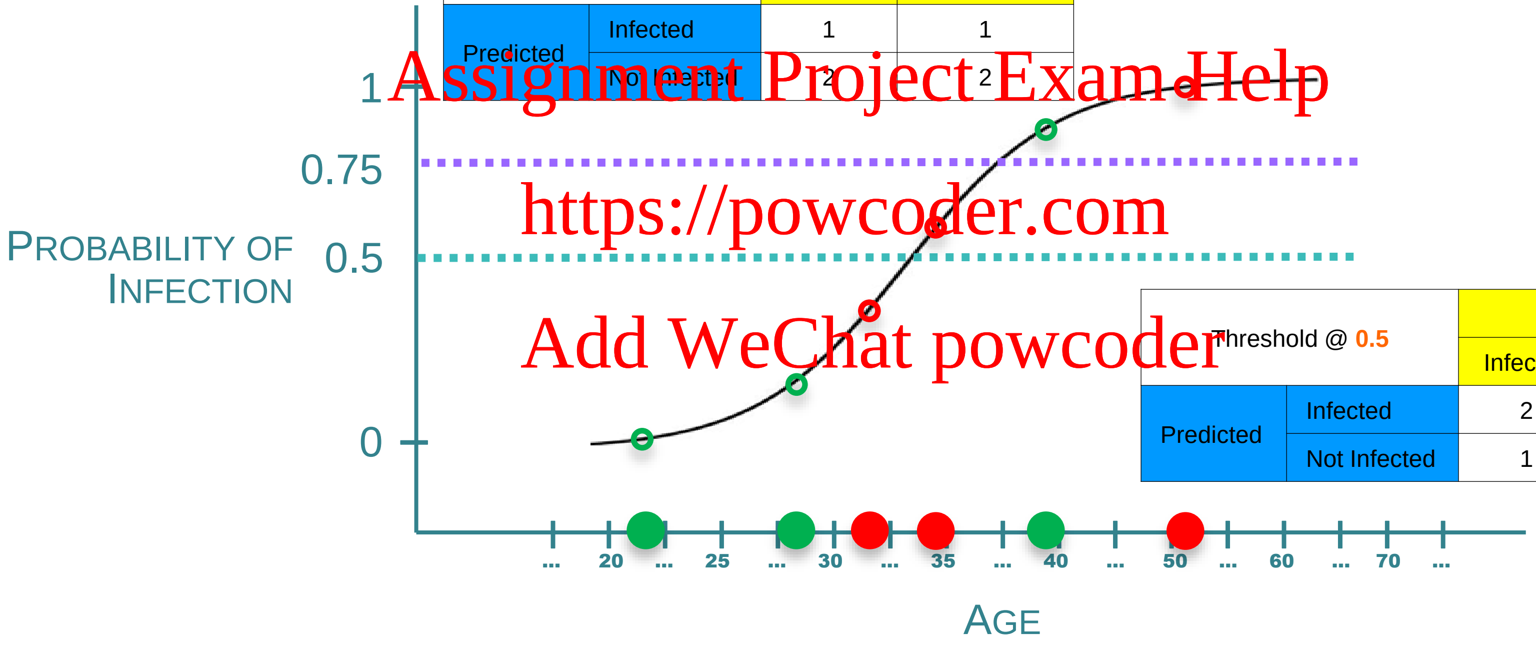
finally, the curve with the maximum likelihood is selected


Logistic regression curve (3D)
costs: 0.926
y
1.0
0.8
0.6
0.4
0.2
0.0
x0
−10
−5
0
5
10
x1
−10
−5
0
5
10
Logistic regression curve
costs: 0.755
y
1.0
0.8
0.6
0.4
0.2
0
x
−150
−100
−50
0
50
100
150
200
epochs: 0
costs
1.0
0.9
0.8
0.7
0.6
w0
0.5
0.3
0.1
-0.1
w1
0.6
0.4
0.2
0
epochs: 0
costs
1.0
0.8
0.6
0.4
w
0
0.02
0.04
0.06
b
0.5
0
-0.5
-1
-1.5

Receiver Operating Characteristic (ROC) Curve

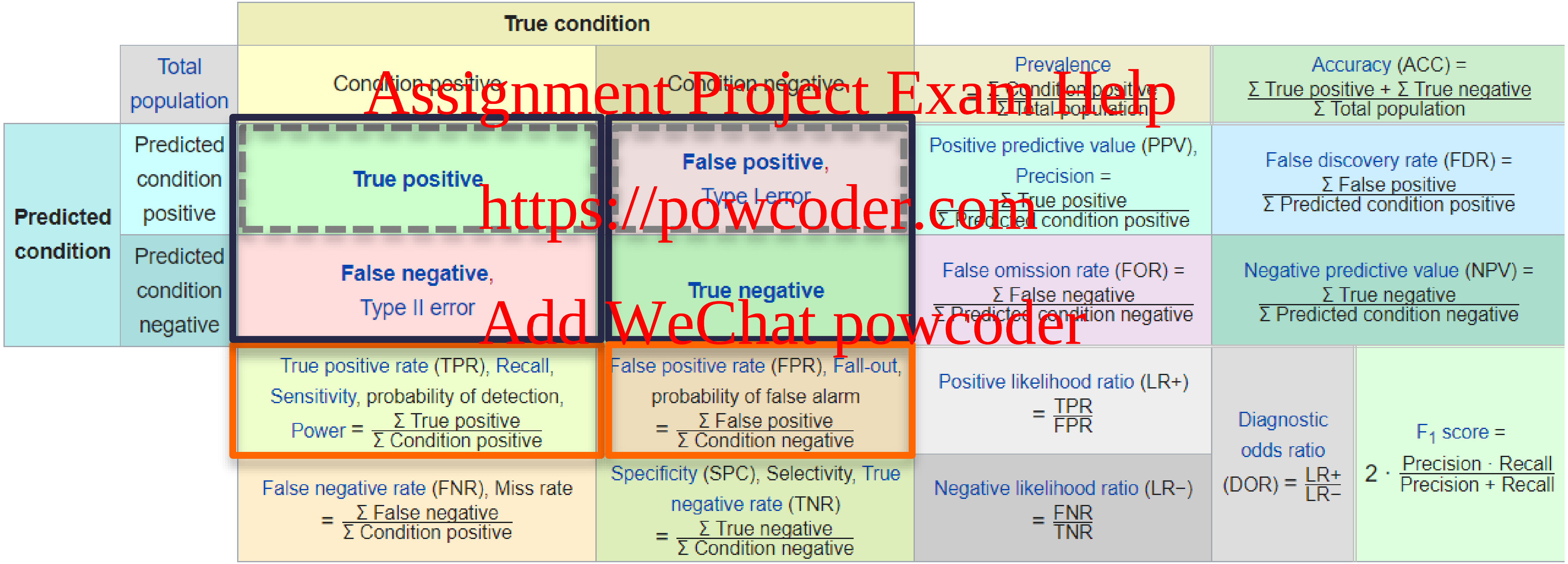
The classification will change as the threshold value changes giving a different confusion matrix each time

Threshold @ 0.75		Actual	
		Infected	Not Infected
Predicted	Infected	1	1
	Not Infected	2	2



Threshold @ 0.5		Actual	
		Infected	Not Infected
Predicted	Infected	2	1
	Not Infected	1	2

A confusion matrix can be characterised by the True Positive Rate and False Positive Rate



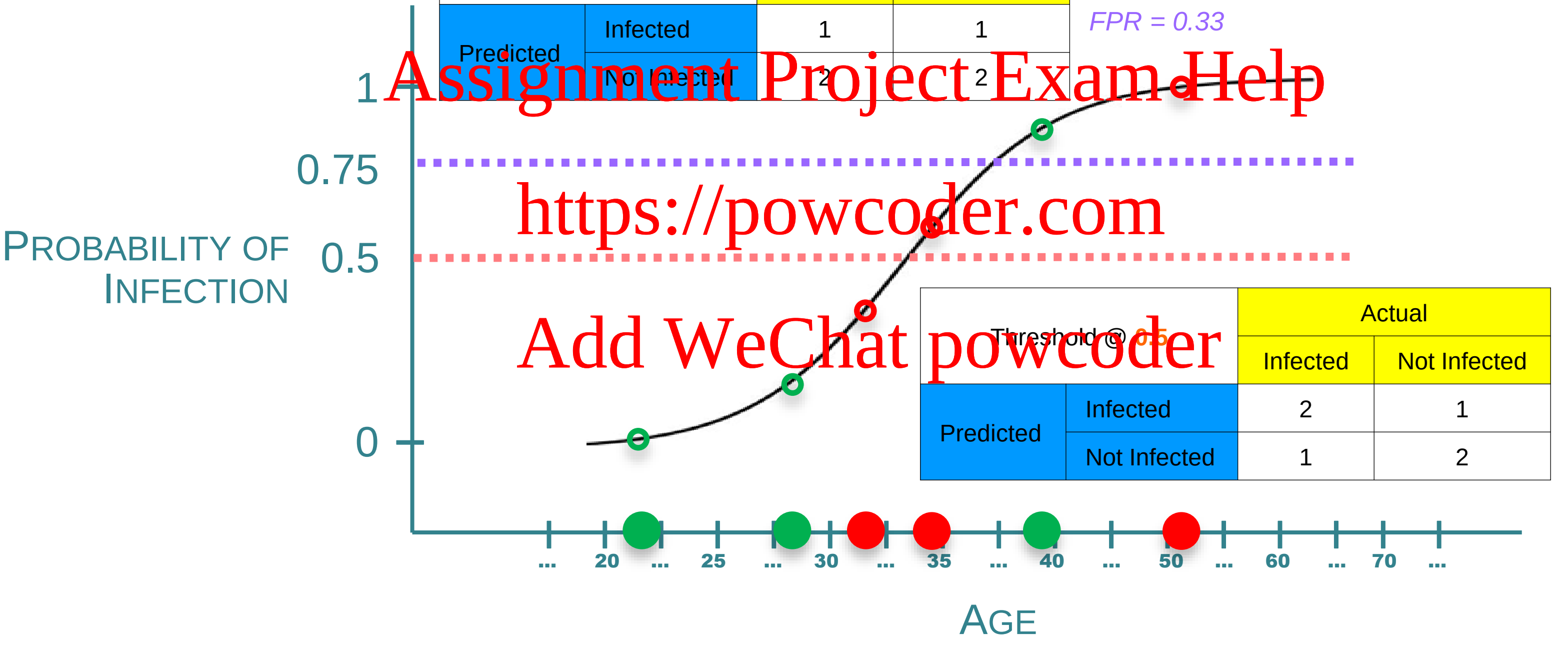
		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

Source: https://en.wikipedia.org/wiki/Confusion_matrix

Therefore, changing the threshold will generate possibly infinite number of TPR and FPR pairs

Threshold @ 0.75		Actual	
		Infected	Not Infected
Predicted	Infected	1	1
	Not Infected	2	2

TPR = 0.33
FPR = 0.33



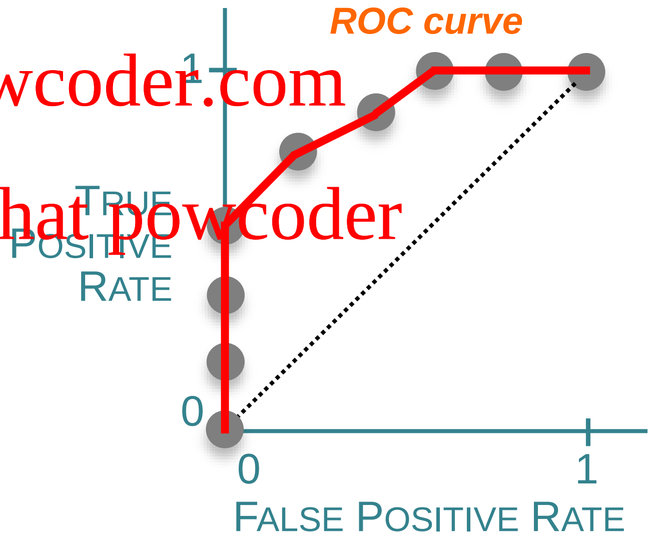
Threshold @ 0.5		Actual	
		Infected	Not Infected
Predicted	Infected	2	1
	Not Infected	1	2

TPR = 0.67
FPR = 0.33

What is the Receiver Operating Characteristic (ROC) curve?

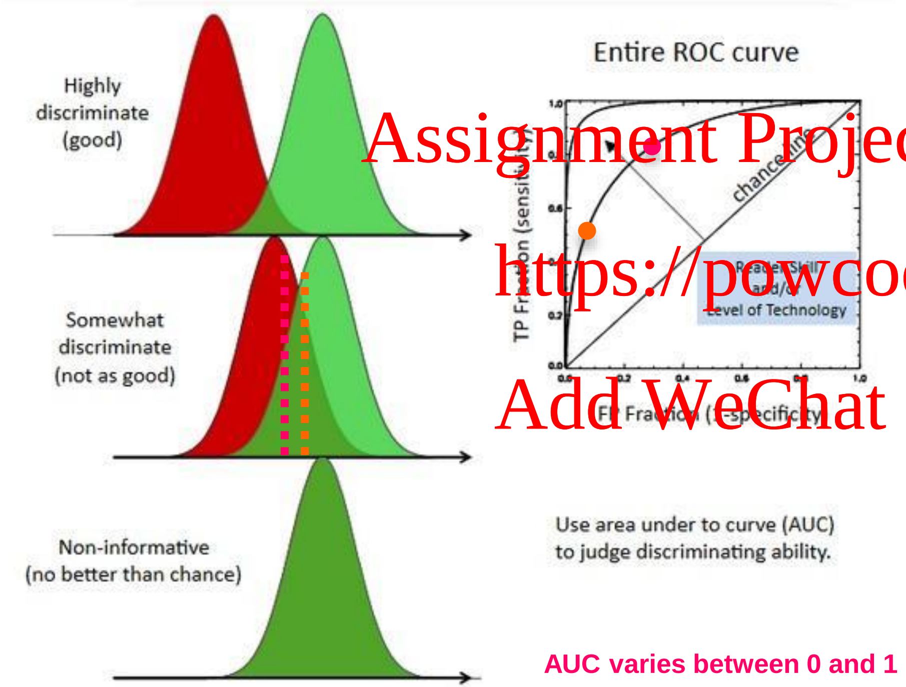
- An ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier as its discrimination threshold is varied

So instead of being overwhelmed with confusion matrices, the ROC curve provides a simple way to summarize all of the information



Classification Metric: AUC (Area Under Curve)

A balanced measure of precision and sensitivity

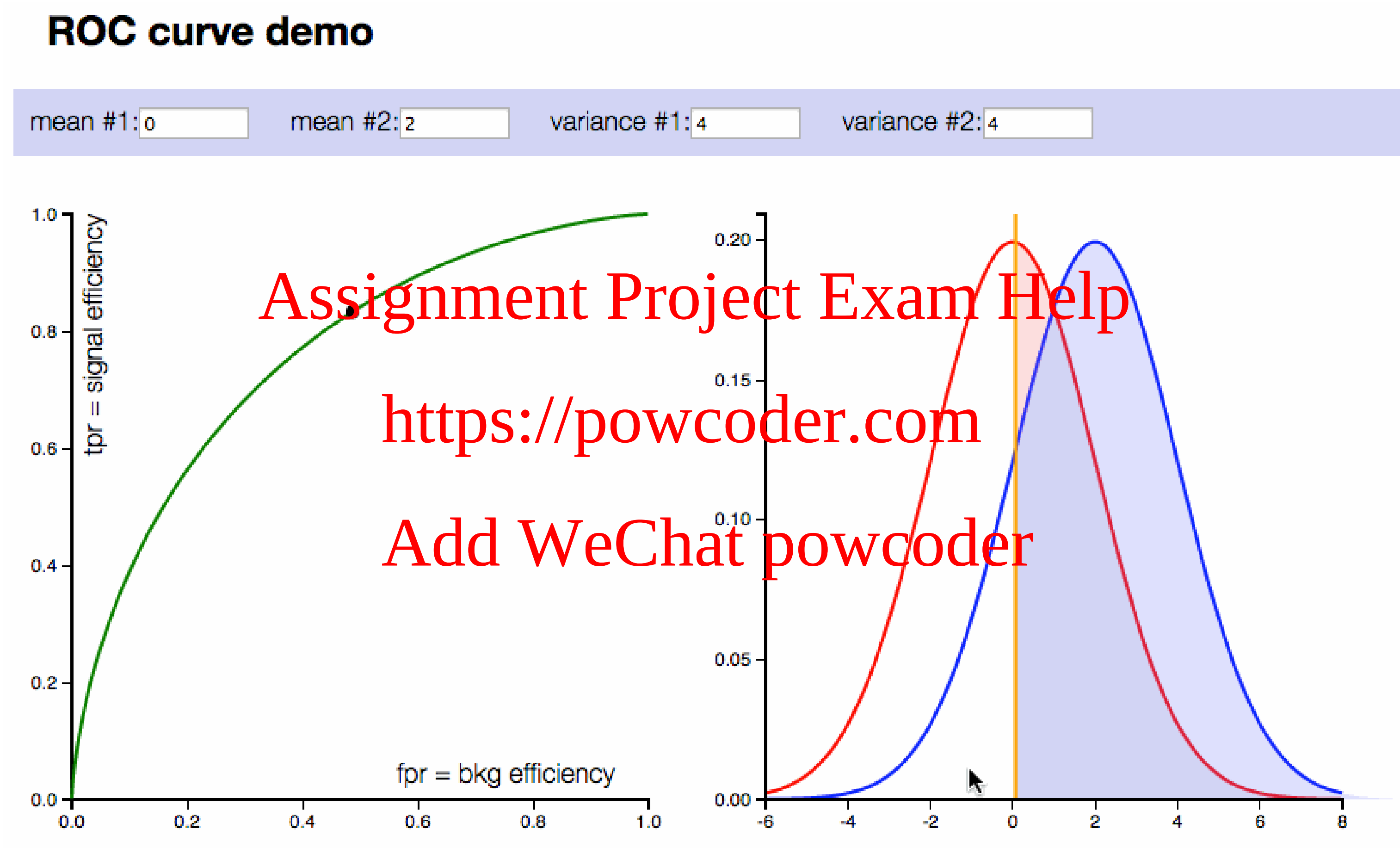


Use area under to curve (AUC) to judge discriminating ability.

AUC varies between 0 and 1

- The ROC curve can be used to compare model predictive power based on TPR and FPR
- Decision will be based on how much area is under the curve
- The ideal curve fills in 100% and will be able to tell negative from positive results 100% of the time
- The ROC curve at the bottom does a worse job than chance, mixing up the negatives and positives

http://arogozhnikov.github.io/2015/10/05/roc-curve.html

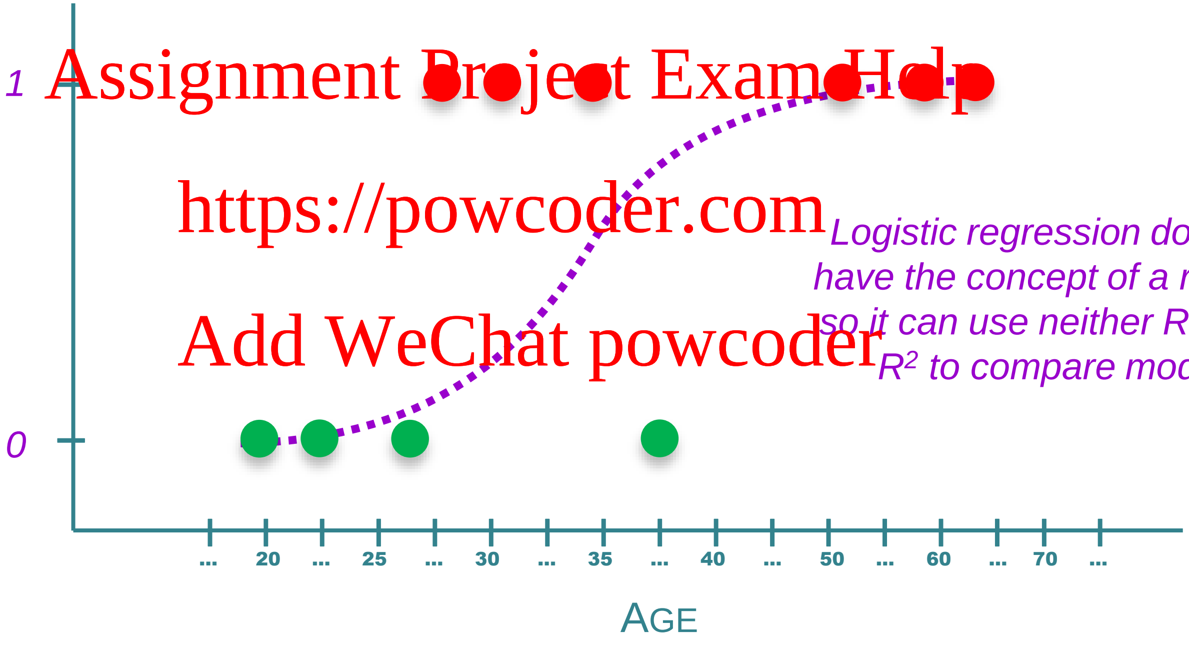


ROC curve demo

mean #1: 0 mean #2: 2 variance #1: 4 variance #2: 4

The Log Loss Function

There is no census on how to calculate R^2 for logistic regression – there are more than 10 different ways to do it

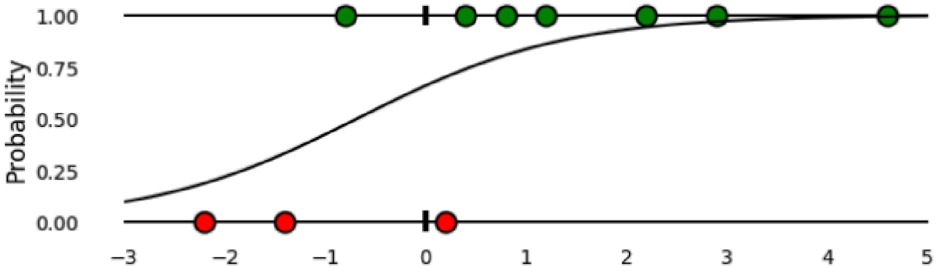


Logistic regression does not have the concept of a residual so it can use neither RSS nor R^2 to compare models

The Log Loss function represents the price paid for inaccuracy of predictions in classification problems

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot \log(1 - p(y_i))$$

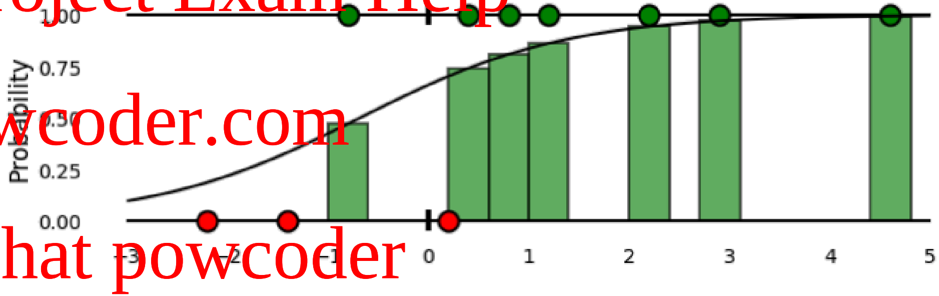
- For each row i in a dataset with N rows
  - y is the outcome (dependent variable) which can be either 0 or 1
  - p is the predicted probability outcome by applying the logistic regression function
- The objective is to minimize the total log loss over the whole dataset by adjusting the estimates in the logistic regression equation
- If y is 1, log loss is minimized with high value of p
- If y is 0, log loss is minimized with low value of p

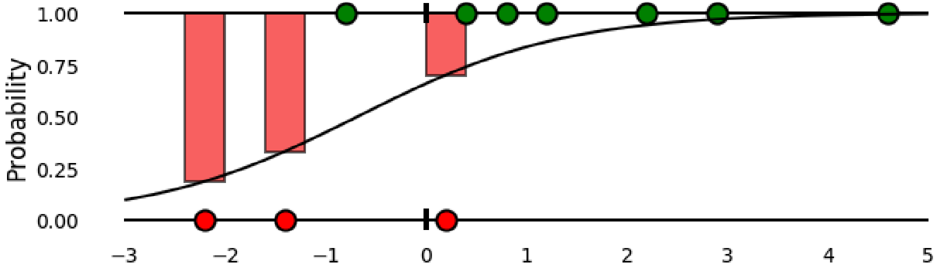


Fitting a logistic regression to predict the probability of a point being green for any given value of x, which can take on either negative or positive value

For all the points belonging to the positive class (green), what are the predicted probabilities given by the classifier?

The green bars represent the probability of a given point being green

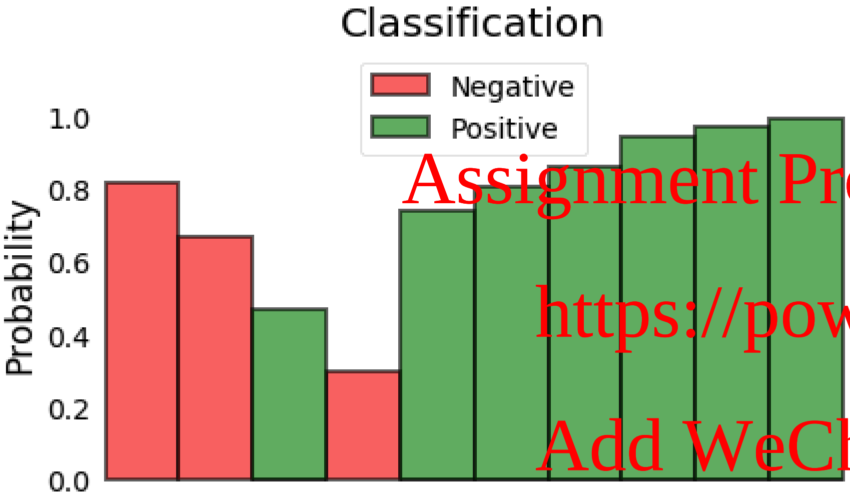


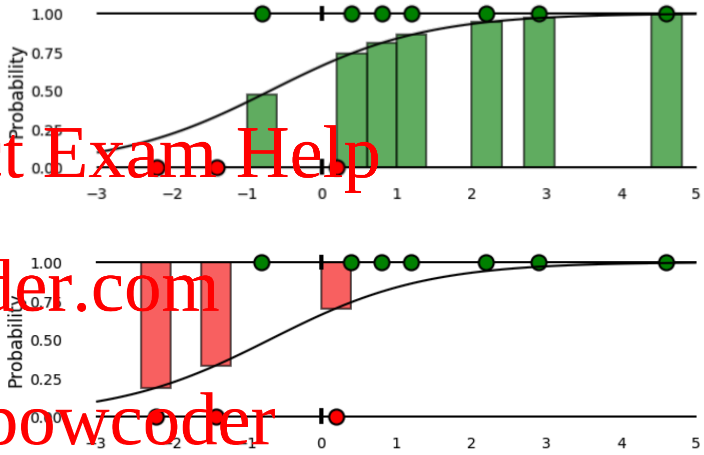


What is the probability of a given point being red?

The red bars above the curve represent the probability of the negative class

The loss function aims to penalize bad predictions



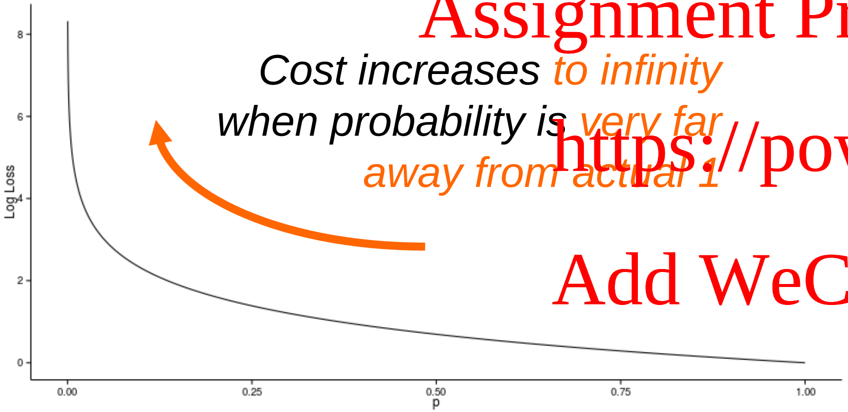


- If the probability associated with the true class is 1.0, we need its loss to be 0
- Conversely, if that probability is low, say, 0.01, we need its loss to be **HUGE**

- Taking the negative log of the probability suits well enough for this purpose
  - the log of values between 0.0 and 1.0 is negative
  - taking the negative log provides a positive value for the loss

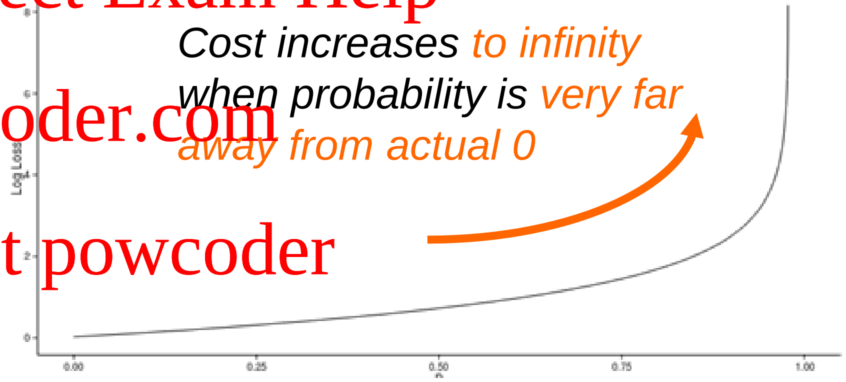
The Log Loss function penalizes heavily the predictions that are confident but wrong

$y = 1$



Cost increases to infinity when probability is very far away from actual 1

$y = 0$



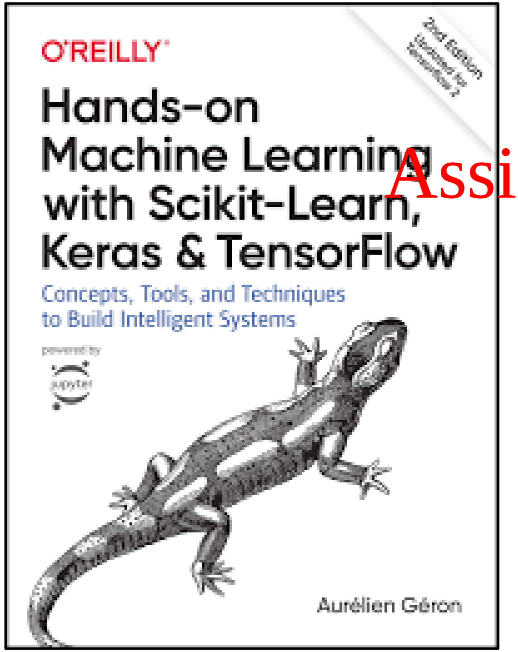
Cost increases to infinity when probability is very far away from actual 0

Logistic Regression Models in a Nutshell

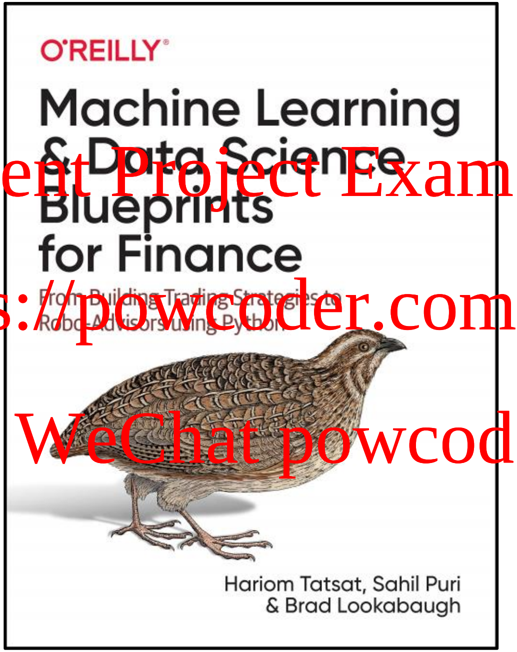
	Property	Description
1	Feature Data Types	Any data type. Encoding is expected for categorical features.
2	Target Data Types	Binary.
3	Key Principles	Predicts the probabilities of an event occurring (probability=1) given certain values of input variables x. The output is a value between 0 and 1. A threshold probability determines to which class the output belongs.
4	Hyperparameters	None.
5	Data Assumptions	Does not require scaling of features.
6	Performance	Regularization is applied by default. Can handle both dense and sparse input. Not able to handle a large number pf categorical features. Vulnerable to overfitting. Cannot solve the non-linear problems.
7	Accuracy	Restrictive expressiveness (e.g. interactions must be added manually) and other models may have better predictive performance.
8	Explainability	Provides probability associated with the classification. Interpretation is more difficult because the interpretation of the weights is multiplicative and not additive.

References

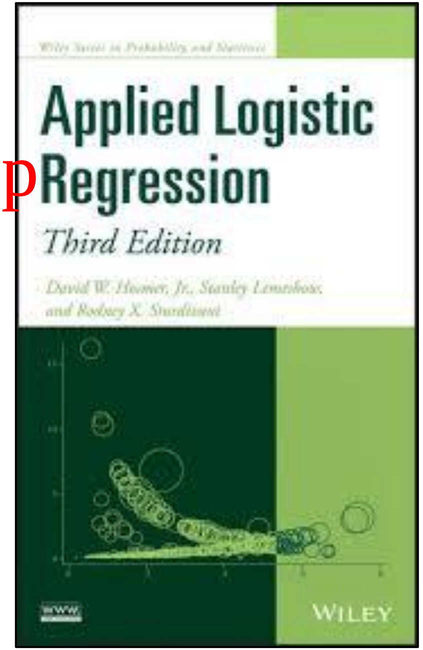
References


O'REILLY
2nd Edition
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
powered by jupyter
Aurélien Géron

“Hands-On Machine Learning with Scikit-Learn and TensorFlow”, Aurelien Geron, O'Reilly Media, Inc., 2017


O'REILLY
Machine Learning & Data Science Blueprints for Finance
From Building Trading Strategies to Robo-Advisors Using Python
Hariom Tatsat, Sahil Puri & Brad Lookabaugh

“Machine Learning & Data Science Blueprints for Finance”, Hariom Tatsat, Sahil Puri, and Brad Lookabaugh, O'Reilly Media, Inc., 2020


Wiley Series in Probability and Statistics
Applied Logistic Regression
Third Edition
David W. Hosmer, Jr., Stanley Lemeshow, and Rodney X. Sturdivant
WILEY

“Applied Logistic Regression”, David W. Hosmer Jr., Wiley, 2013

References

- "Chapter 14 Support Vector Machine" in "Hands-On Machine Learning with R" (https://bradleyboehmke.github.io/HOML/svm.html)
- "The Gaussian RBF Kernel in Nonlinear SVM", Suvigya Saxena, 2020 (https://medium.com/@suvigya2001/the-gaussian-rbf-kernel-in-non-linear-svm-2fb1c822aae0)
- "C and Gamma in SVM, A Man Kumar", 2018 (https://medium.com/@myselfaman12345/c-and-gamma-in-svm-e6cee48626be)
- "Using Random Forests in Python with Scikit-Learn", Fergus Boyles, 2017 (https://www.blopig.com/blog/2017/07/using-random-forests-in-python-with-scikit-learn/)
- "Ensemble Learning: 5 Main Approaches", Diogo Menezes Borges (https://www.kdnuggets.com/2019/01/ensemble-learning-5-main-approaches.html)
- "Logistic Regression: A Concise Technical Overview", Reena Shaw, Kdnuggets (https://www.kdnuggets.com/2018/02/logistic-regression-concise-technical-overview.html)

THANK YOU