Advanced Topics in Neural Networks

Fintech

Lesson 13.3

Class Objectives

By the end of the class, you will be able to:

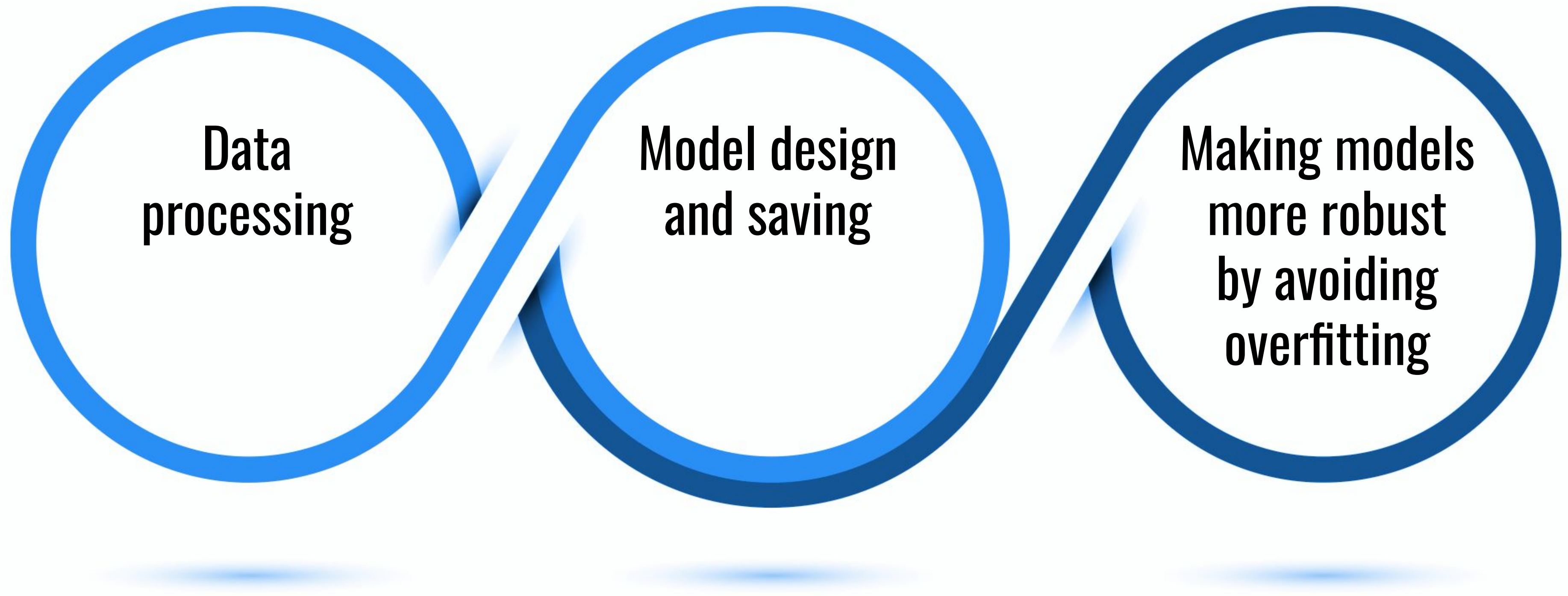
- Save and redeploy previously fit models.
- Describe the problem of overfitting and implement code that can mitigate it.
- Acquire resources for more advanced aspects of deep learning that are beyond the scope of today's class.

WELCOME

We will explore some common techniques for optimizing neural network models.

Advanced Topics in Neural Networks

These techniques span all elements of building a neural network:



The technology behind neural networks is relatively new.

Techniques for optimizing neural network performance are evolving as data scientists conduct further research.

There is no infallible approach to designing an optimal neural network, but we can apply some guidelines and best practices to find the most suitable model for a particular problem.

Overfitting

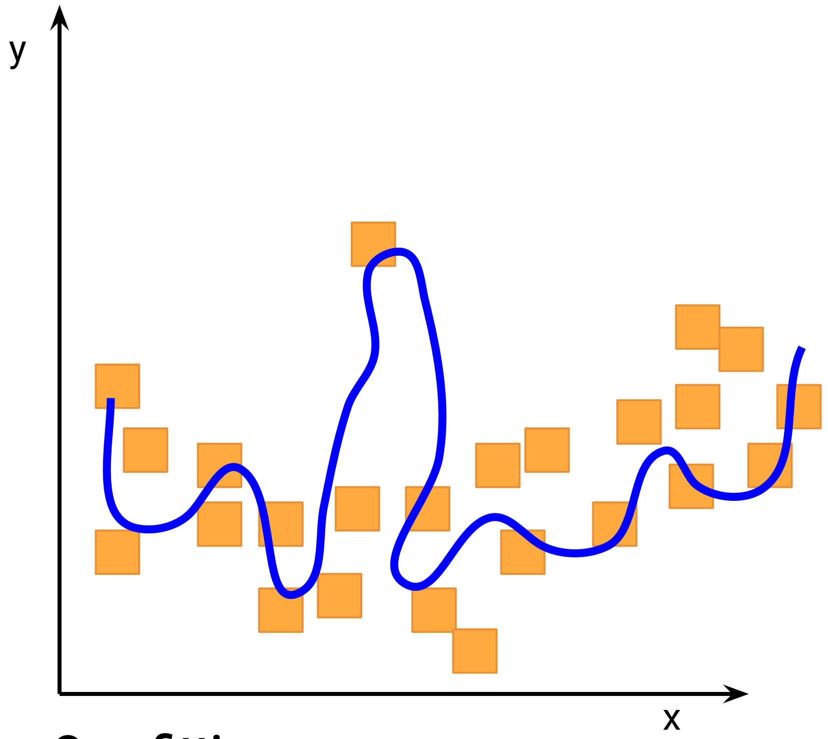
Overfitting results from attempting to model the training data too precisely.

Overfitting

Deep neural networks are beneficial because they can discover subtle, complex patterns in data that other models often can't find.

However, there's a limit to what we can learn about the data.

Random things that are unlikely to repeat— coincidences—can occur in the training data.



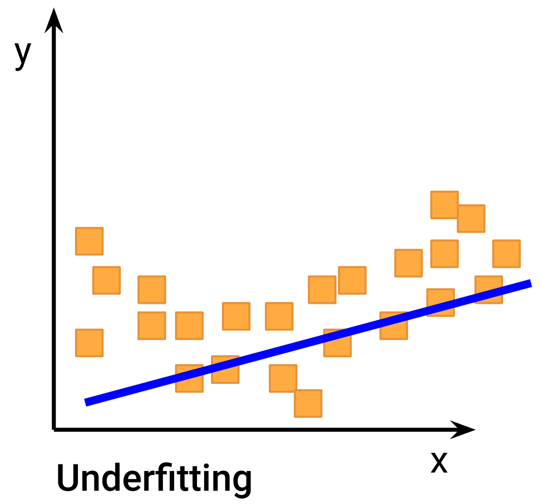
Overfitting

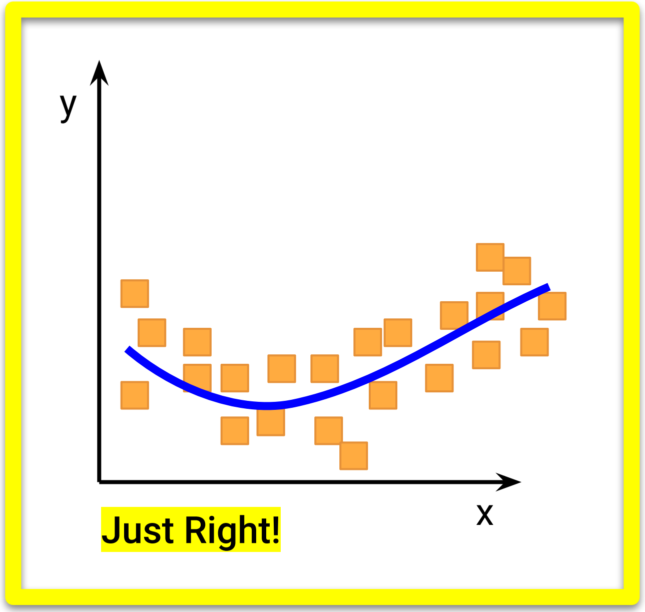
Overfitting

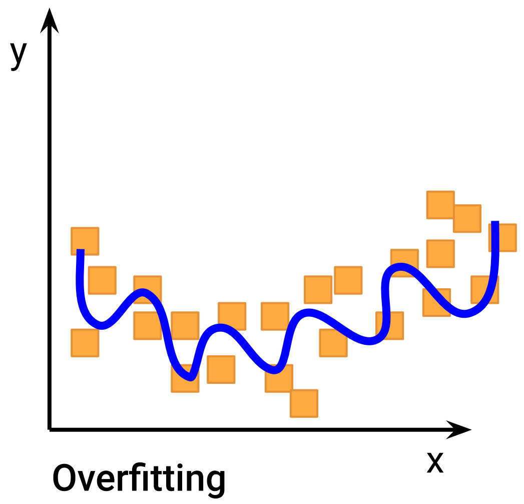
If we have a model that perfectly fits the training data, the model probably won’t perfectly fit any future data that it encounters.

Overfitting

Sometimes, a complex model that is extremely well fit on the training data will perform worse on new data than a simpler model that used the same training data.







Techniques to Reduce Overfitting

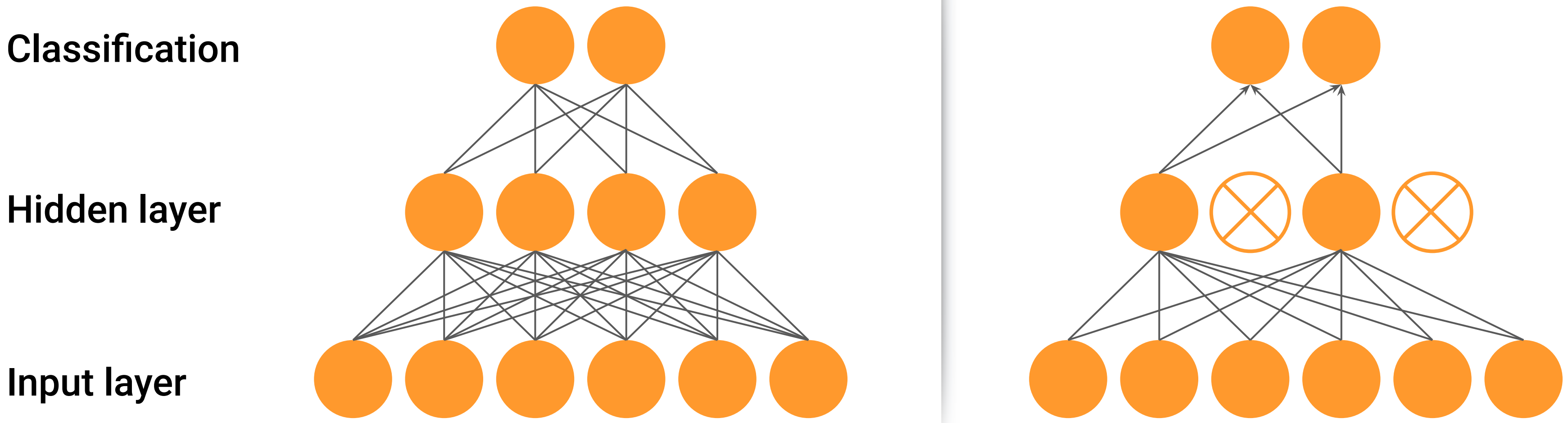
The most effective techniques to reduce overfitting in neural networks involve the following concepts:

- Dropout
- Regularization
- Use of a validation set

Techniques to Reduce Overfitting: Dropout

Dropout is when we cut connections at random in the neural network when it's training. This is similar to when we "forget on purpose." The model finds certain relationships but disregards them.

Without dropout **With dropout**



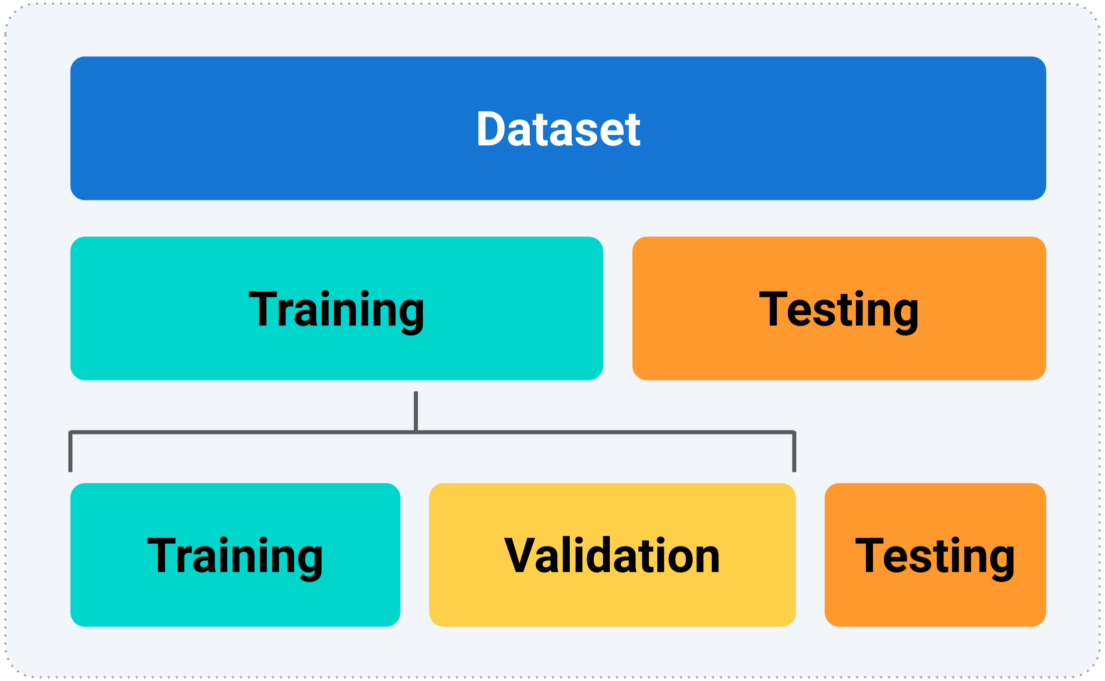
Regularization involves cutting or downplaying specific (not random) connections, depending on their importance.

The **less impact** that a connection has, the **greater chance** it will be minimized or eliminated.

Techniques to Reduce Overfitting: Use of a Validation Set

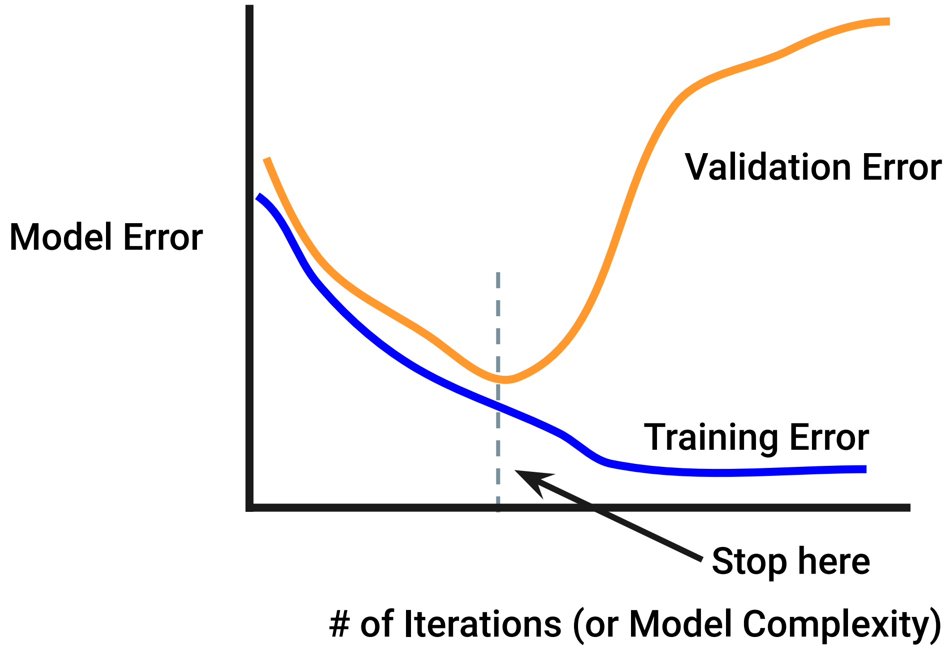
A **validation set** is an additional dataset that we do not use for training the model. It's similar to the test dataset.

By monitoring the model's performance on the validation set as we continue to train the model, we can observe any increasing risk of overfitting.



Techniques to Reduce Overfitting

By monitoring the model's performance on the validation set as we continue to train the model, we can observe any increasing risk of overfitting.



Techniques to Reduce Overfitting

Next, we'll introduce each of these three concepts in more detail and demonstrate the code in Keras.

Dropout	We add dropout as a layer, similar to other sequential layers.
Regularization	We add regularization as a parameter to any existing layers.
Validation Set	We add a `validation_split` parameter when we `fit` our model, so we can track and save the performance on this additional dataset. Once we fit the model, we can then visualize performance differences between the validation and training datasets.

Instructor Demonstration

Overfitting—Coding Demo

Questions?

Activity: Overfitting

In this activity, you will build a neural network and apply various techniques to prevent overfitting.

Suggested Time:

20 minutes

Time's Up! Let's Review.

Questions?

Additional Methods for Model Performance Evaluation—ROC and AUC

The **confusion matrix** is a method for assessing the performance of a binary classification model.

Let’s review the four components of this matrix.

Confusion Matrix: True Negatives

Negative values that are correctly classified as negative.

	Predicted: No (0)	**Predicted: Yes (1)**
Actual=No (0)	True Negatives (TN)	False Positive (FP)
Actual=Yes (1)	False Negative (FN)	True Positives (TP)

Confusion Matrix: False Negatives

Positive values that are incorrectly classified as negative.

	Predicted: No (0)	Predicted: Yes (1)
Actual=No (0)	True Negatives (TN)	False Positive (FP)
Actual=Yes (1)	False Negative (FN)	True Positives (TP)

Confusion Matrix: False Positives

Negative values that are incorrectly classified as positive.

	Predicted: No (0)	**Predicted: Yes (1)**
Actual=No (0)	True Negatives (TN)	False Positives (FP)
Actual=Yes (1)	False Negatives (FN)	True Positives (TP)

Confusion Matrix: True Positives

Positive values that are correctly classified as positive.

	Predicted: No (0)	**Predicted: Yes (1)**
Actual=No (0)	True Negatives (TN)	False Positives (FP)
Actual=Yes (1)	False Negatives (FN)	True Positives (TP)

Introducing the ROC Curve and AUC

Two techniques that use the values from the confusion matrix to check and visualize the performance of a classification model are:

ROC curve

Receiver
Operating
Characteristic

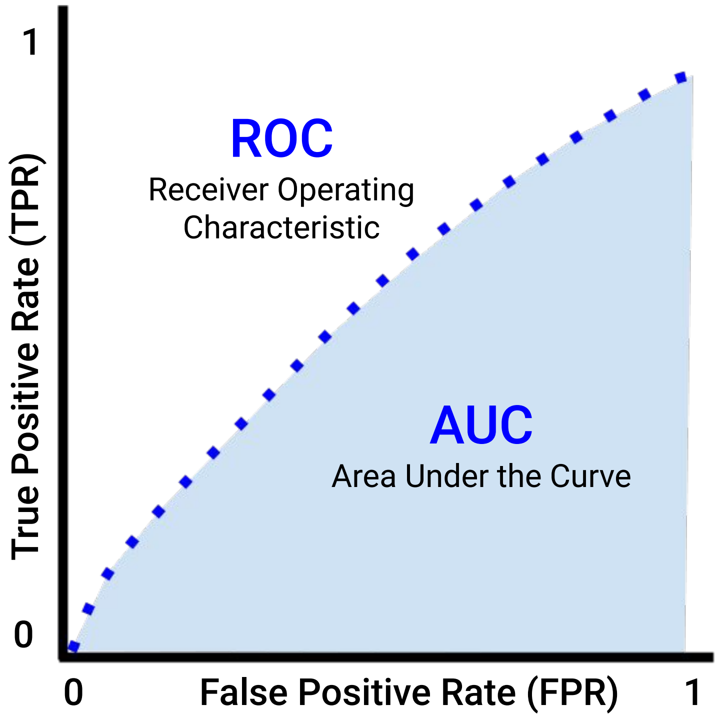
AUC

Area
Under the
Curve

Introducing the ROC Curve and AUC

The ROC curve shows the performance of a classification model as its discrimination threshold is changed—how the model decides what is a 0 and what is a 1.

By comparing ROC curves across training, test, and validation datasets, we can better understand whether our model is overfit.



Understanding ROC (Receiver Operating Characteristic)

To plot a ROC curve, we use two parameters:

True positive rate (TPR or “recall”)

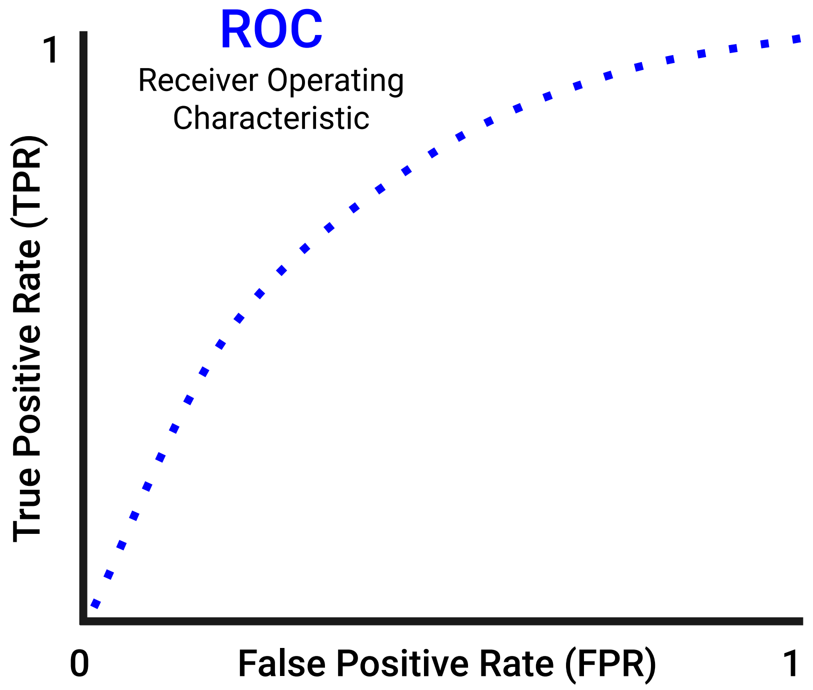
$$TPR = \frac{TP}{TP + FN}$$

False positive rate (FPR)

$$FPR = \frac{FP}{FP + TN}$$

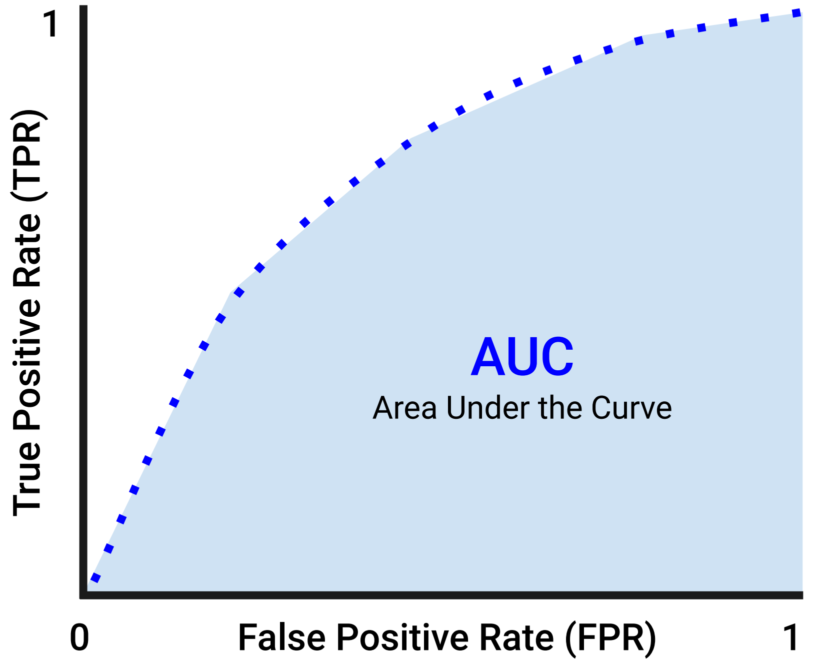
Understanding ROC (Receiver Operating Characteristic)

Every point in the ROC curve represents the TPR versus FPR at different thresholds.



Understanding AUC (Area Under the Curve)

We might find that interpreting the ROC curve is challenging. The AUC can help us by measuring the area underneath the entire ROC curve, from (0,0) to (1,1).



Understanding AUC (Area Under the Curve)

The AUC value ranges from 0 to 1.

AUC = 0.0

AUC = 1.0

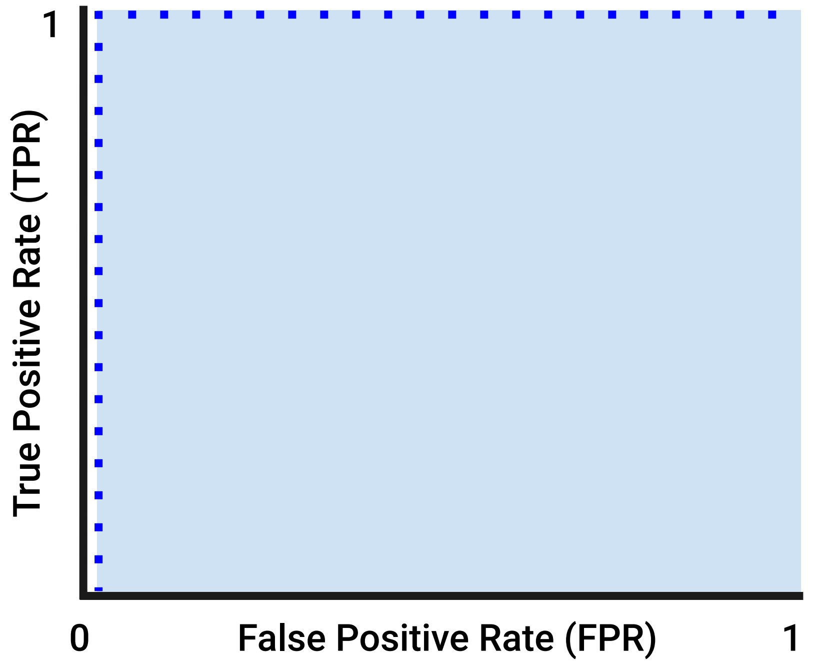
For a model with predictions that are **100% wrong**.

For a model that predicts **everything correctly**.

Understanding AUC (Area Under the Curve)

`AUC = 1` is a paradox because it means that the model is perfect, but it also is likely to be overfit.

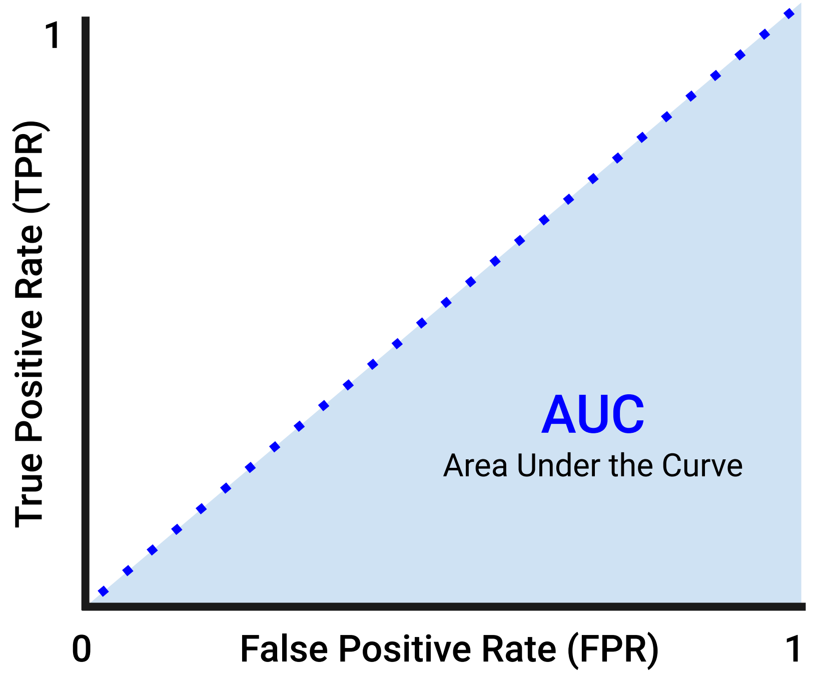
A model with this AU value correctly distinguishes between positive and negative classes.



Understanding AUC (Area Under the Curve)

`AUC = 0.50` means that the model is unable to distinguish between positive and negative classes.

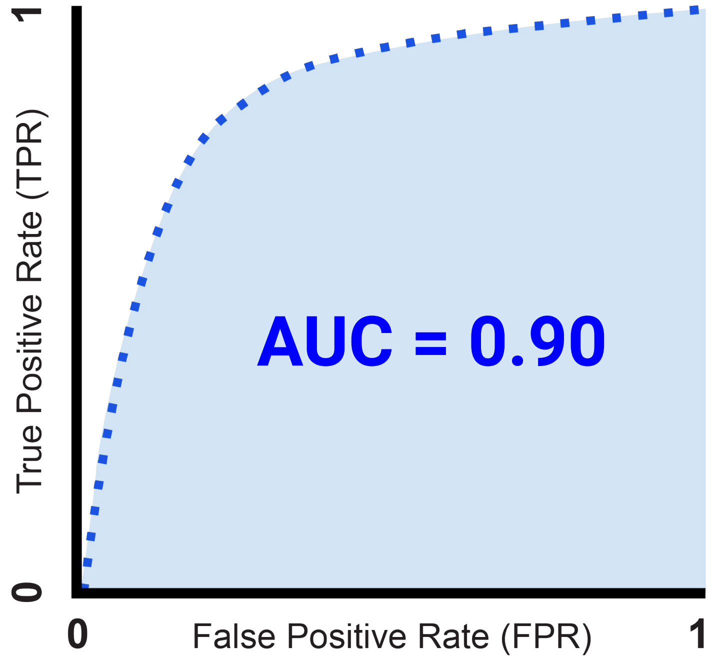
This model would produce results no different than random guessing (50% chance of either a 0 or 1).

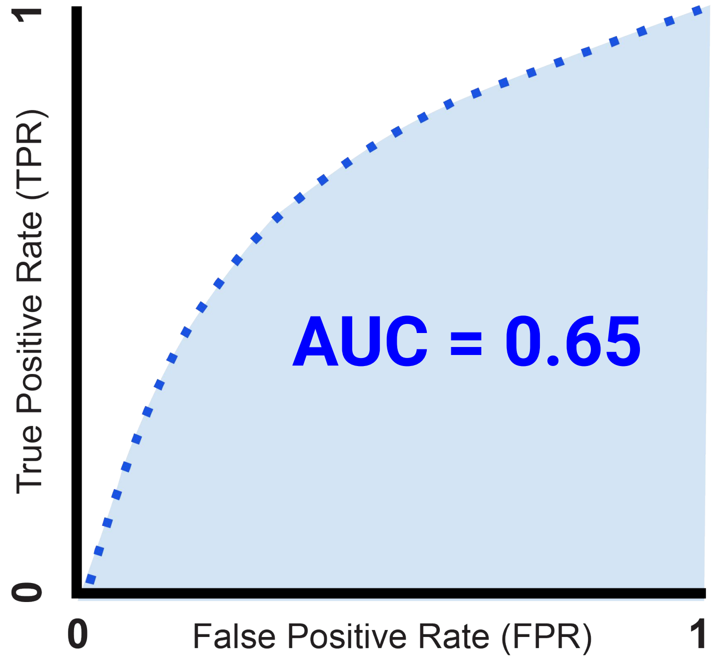


The higher the AUC, the better the model is at predicting zeros as zeros and ones as ones.

Understanding AUC (Area Under the Curve)

A model with `AUC = 0.90` may be better than a model with `AUC = 0.65`.





Time to Code

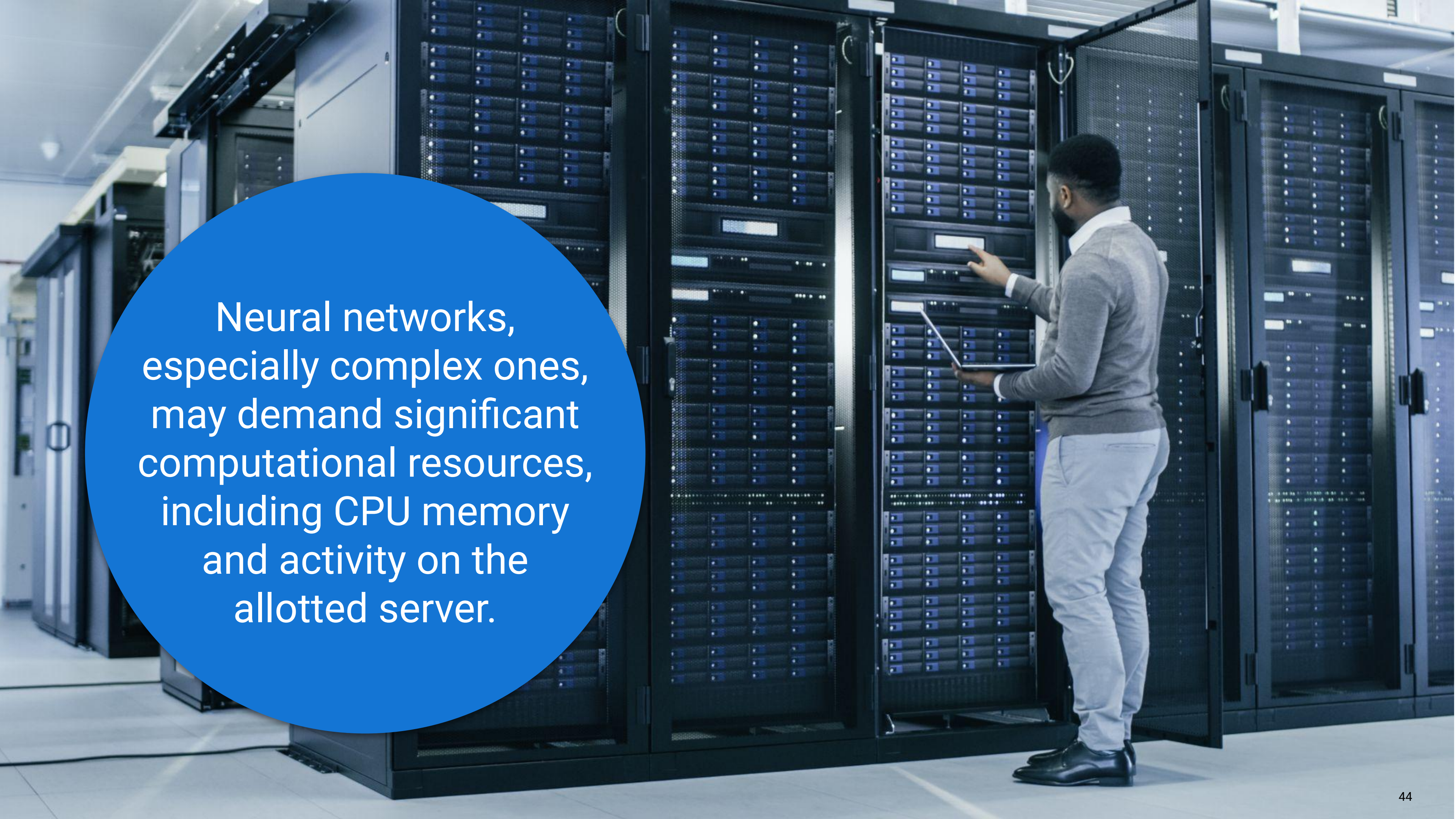
Additional Methods for Model Performance Evaluation—ROC and AUC

Suggested Time:

15 minutes

Questions?

Saving Models



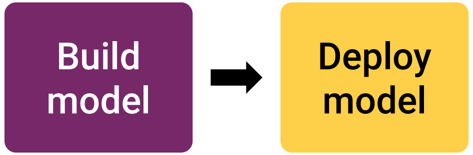
Neural networks, especially complex ones, may demand significant computational resources, including CPU memory and activity on the allotted server.

Training a complex neural network on a medium or large dataset can take hours (or even days)!

Saving Models

A Simple Modeling Approach

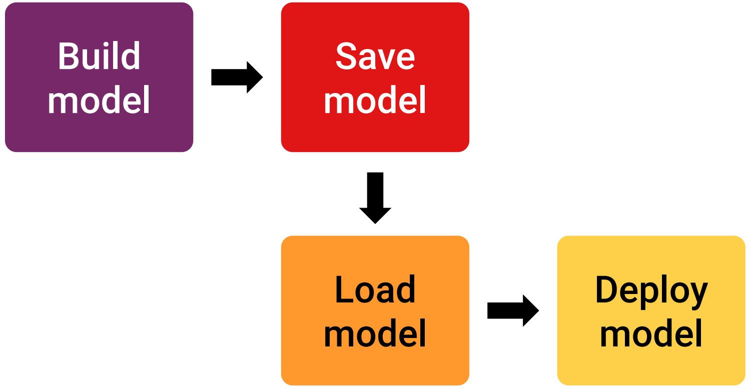
For simple modeling problems like the ones covered in this module, we can train a model in the same notebook where we analyze our data.



A Formal Modeling Approach

For more formal applications of neural networks and deep learning models, data scientists don't have enough time or resources to build and train a model each time they analyze data.

In these cases, they must store and access trained models outside of the training environment.



Saving Models

Data scientists:

- Publish trained models in scientific papers
- Deploy them in software
- Share them on GitHub
- Pass them along to colleagues

Saving Models

Sharing only the weights, parameters, input weights, and biases for each layer of the model would [illegible]stration a[illegible]confusion.

Instead, we can use the Keras `Sequential` model's `save` function to export an entire model to a Hierarchical DataFormat HDF5 file, which includes:

01 The configuration of the model layers

02 The weights associated with each layer

03 The activation functions

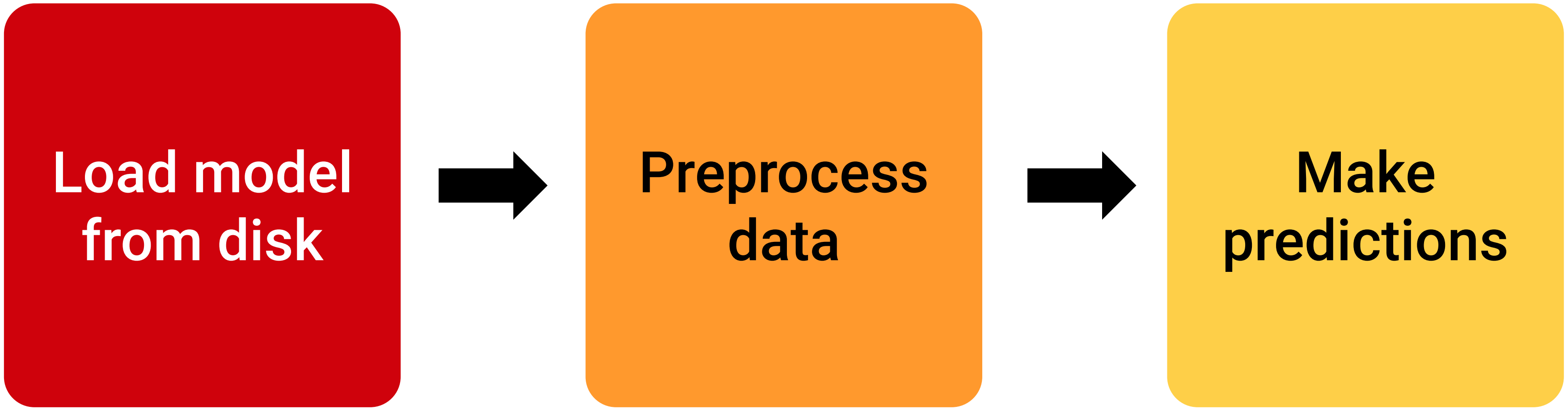
04 The optimizer

05 The set of losses and metrics

Saving Models

Once we have created the HDF5, anyone can import the same trained
their environment using the Keras load_model function.

Then, they can use the model for analysis.



Instructor Demonstration

Saving Models

Activity: Saving Models

In this activity, you will create a deep learning model from the credit score data, save it, and load it to evaluate its performance on unseen data.

Suggested Time:

20 minutes

Time’s Up! Let’s Review.

Questions?

Countdown timer

15:00

(with alarm)

Instructor Demonstration

Transfer Learning

Questions?

Activity: Transfer Learning: “Model Homes”

In this activity, you will load a pre-existing deep neural network that was trained on one housing market area and apply it to another area.

Suggested Time:

15 minutes

Time’s Up! Let’s Review.

Questions?

Recap

Recap

Congratulations on learning advanced skills. You can now:

- Utilize a number of techniques to avoid overfitting, which is a critical skill for neural networks and ML in general.
- Build and interpret ROC curves and the AUC, which are common methods in the field for determining a model's goodness of fit.
- Save DNN models either for future use or for private or public collaboration with others.
- Use previously built models and tweak them according to the specific circumstances of the data.

The End