Introduction to Watson Studio

Power of data. Simplicity of design. Speed of innovation.

Bernie Beekman
Michael Cronk
Prithvi Rao
Zoya Yeprem

Outline

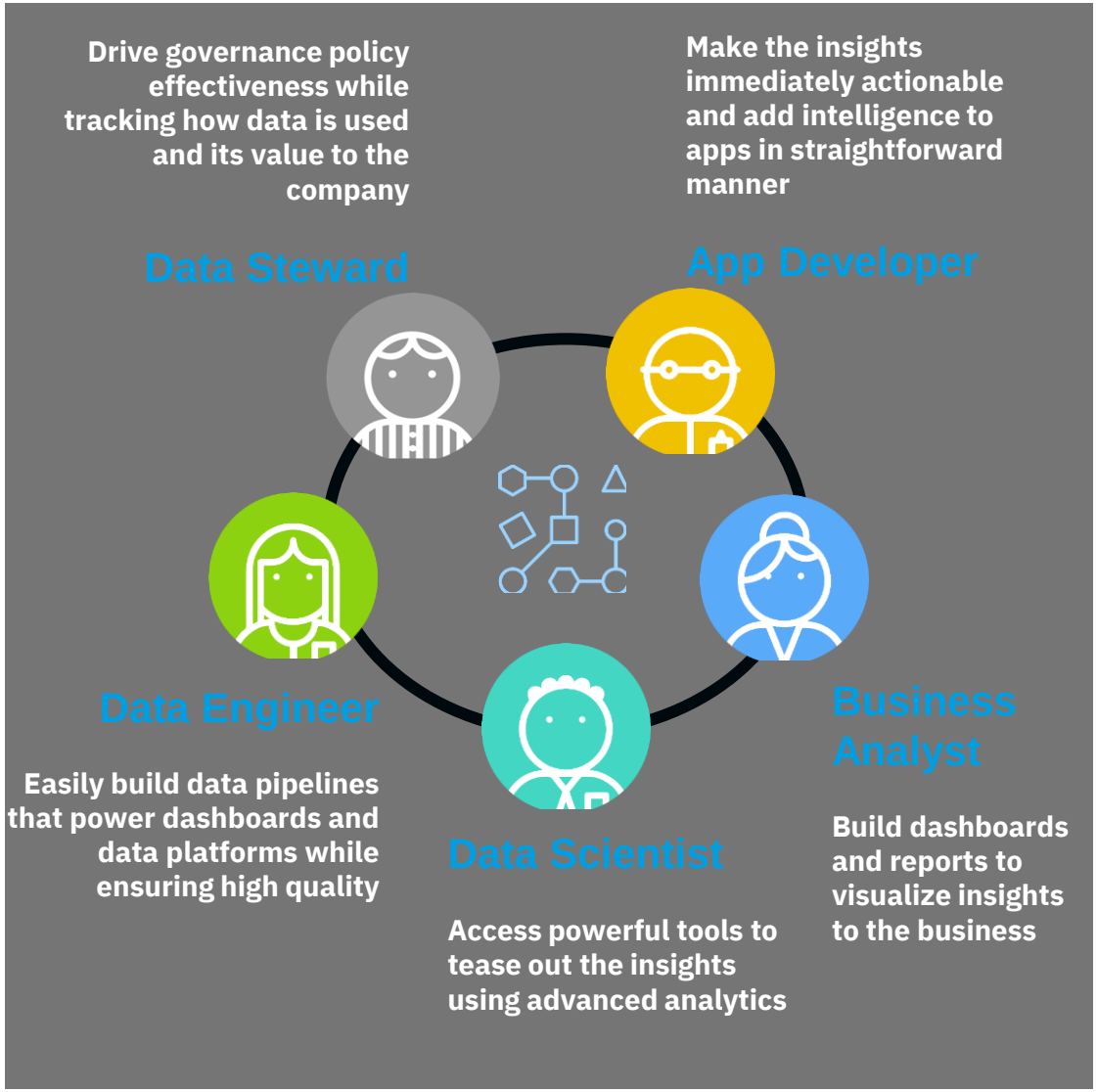
- **Watson Studio Overview** 
- **Lab Overview**

IBM takes an Enterprise Approach to Data Science

- Freedom of Choice
  - Choose programming languages, open source libraries, IBM value-add capabilities
  - Code/Click
  - Machine Learning/Deep Learning/Decision Optimization.
- Operationalize Machine Learning
  - Manage complete ML lifecycle – Build, Deploy, Manage, Scale, Monitor, Re-train
- Hybrid ML
  - Build where you want, deploy where you want
- Governance
  - Ensure that right people get access to the right data

IBM Watson Studio Platform

An integrated platform of tools, services, data, and metadata that help companies or agencies accelerate their shift to be data-driven organizations.



Watson Studio Deployment Options

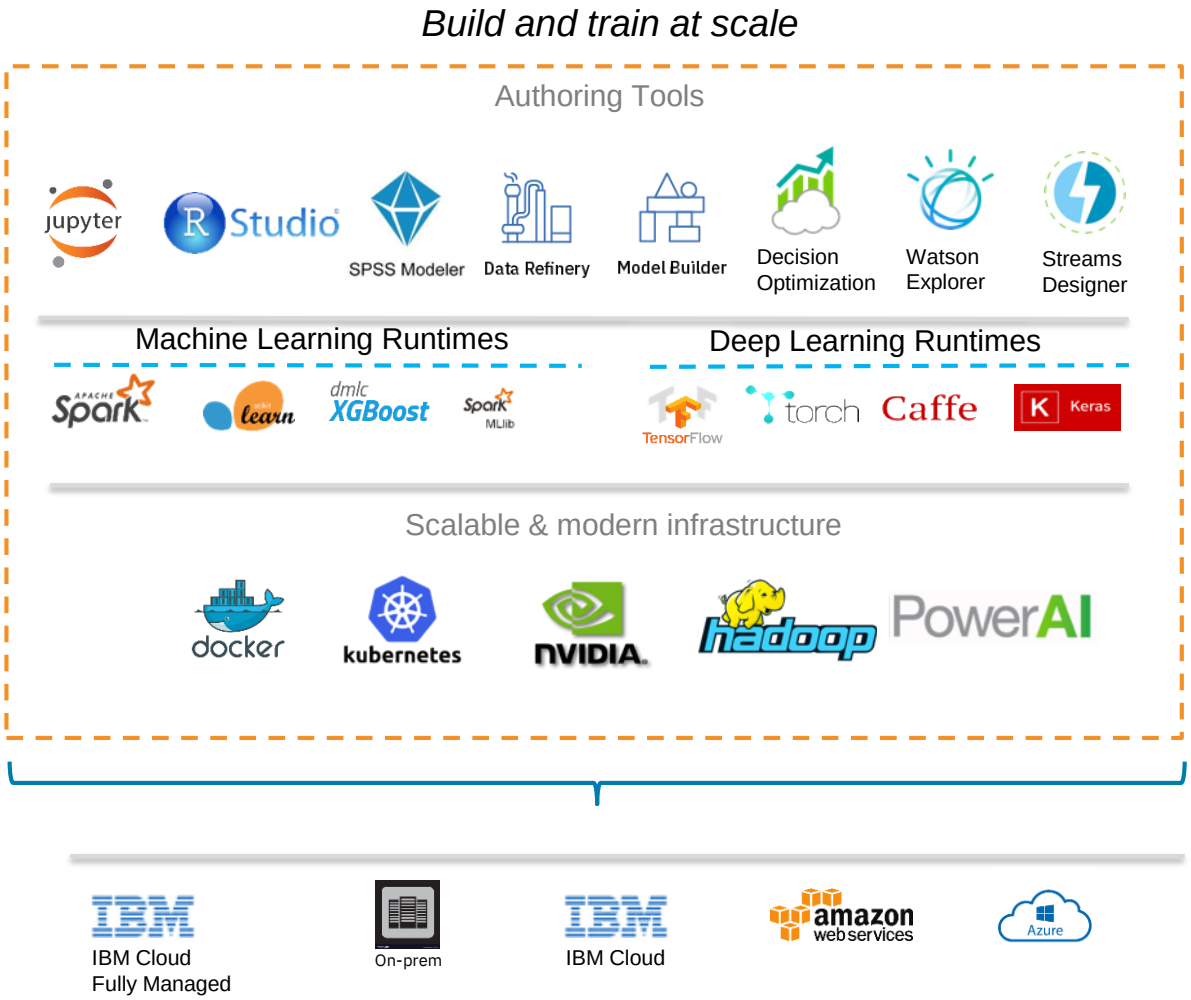
- Watson Studio on IBM Cloud
  - Managed offering provided by IBM
- Watson Studio Local
  - On-premise – Private Cloud
  - IBM Cloud, AWS, Azure
- Watson Studio Desktop
- IBM Cloud Private for Data
  - Watson Studio Local

Watson Studio Tools

- Using best of breed - Open source & IBM tools
- Code (R, Python or Scala) and no-code/visual modeling tools

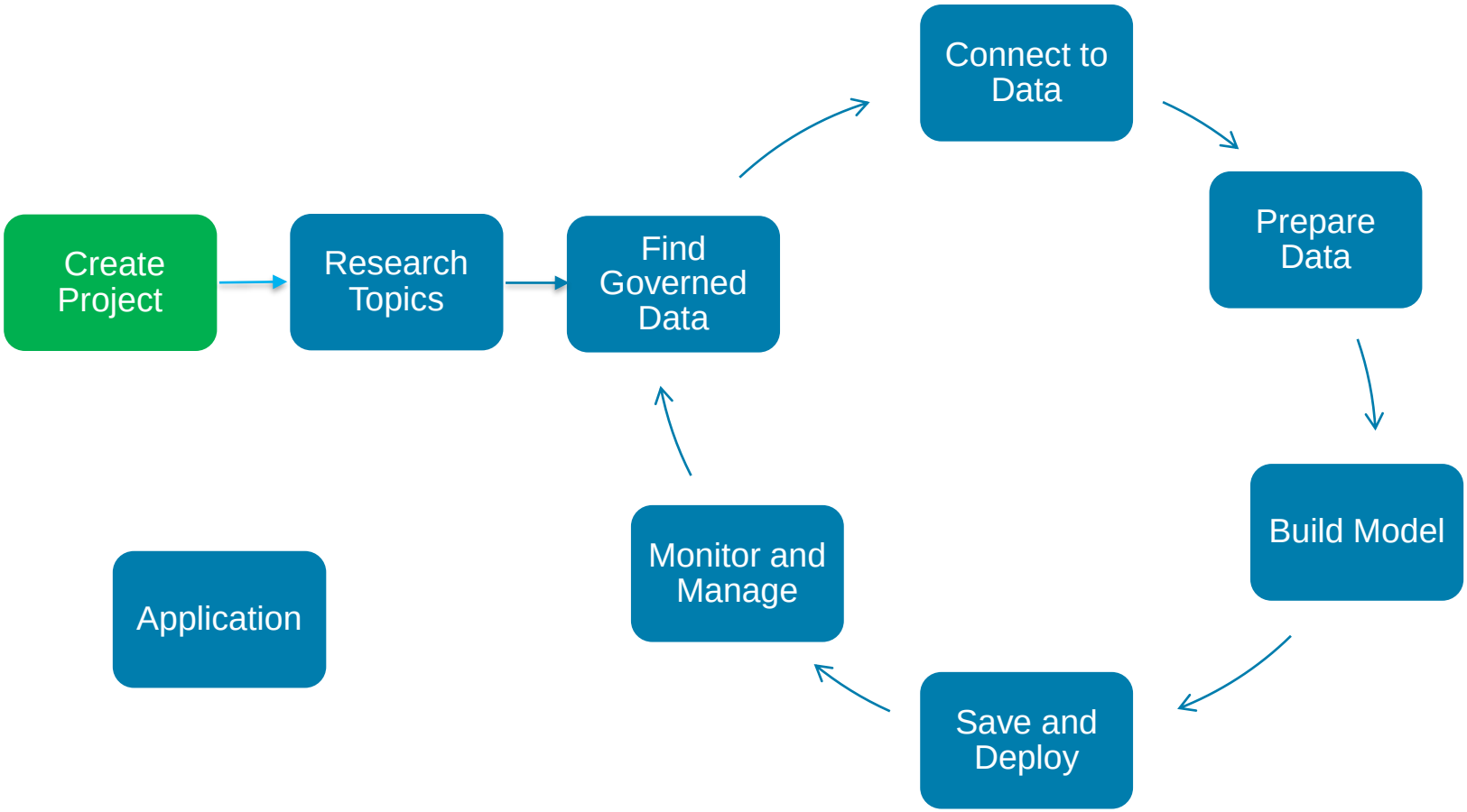
- Container-based resource management
- Elastic cpu/gpu power
- Run on x86, Power, zLinux
- Integrate with Cloudera and HDP

- Train and deploy where your data lives



Watson Studio supports the Data Science Lifecycle

Build, train, deploy, and monitor at scale ML/DL workflows to infuse AI into the enterprise to drive innovation.



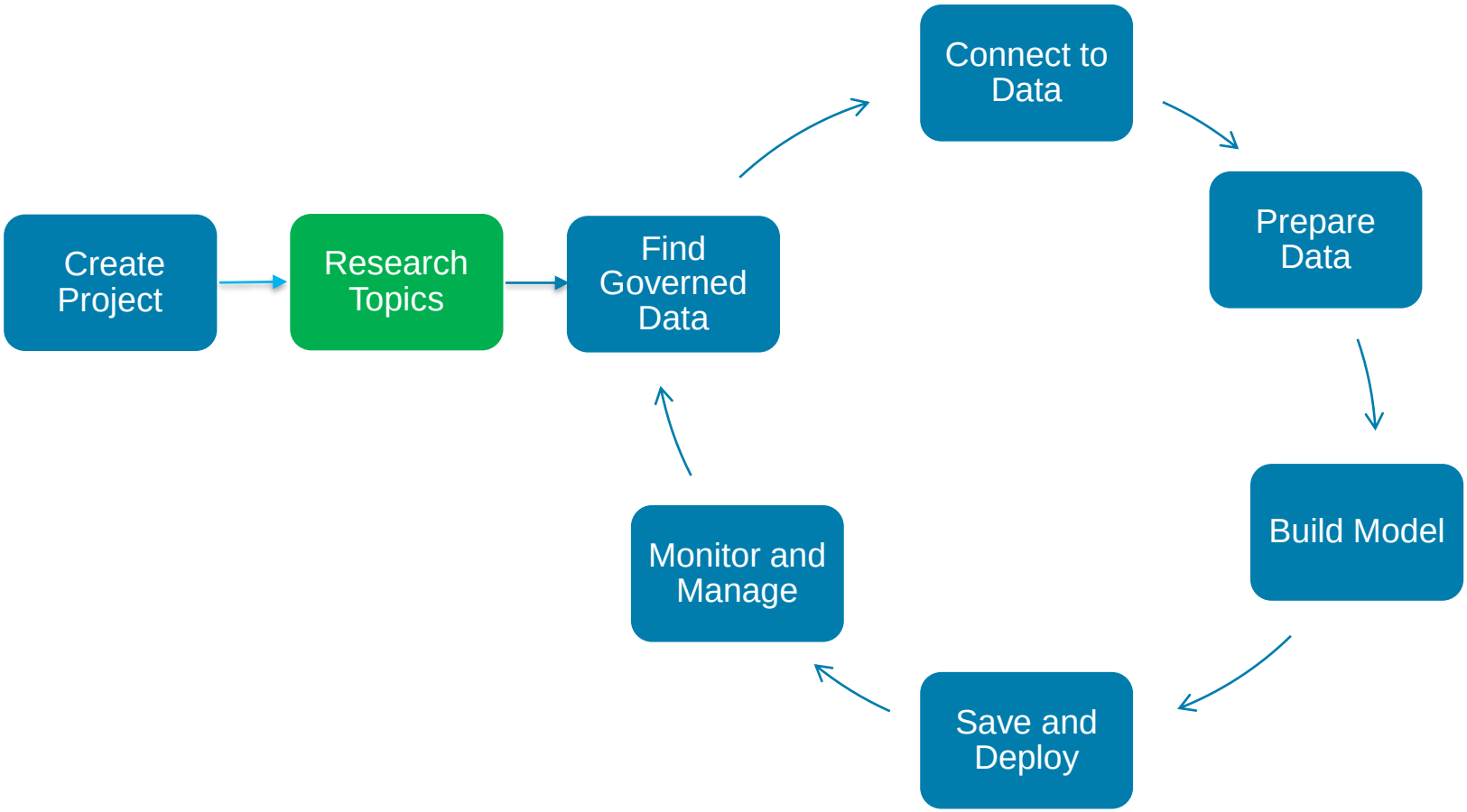
Watson Studio Project Features

Making Data Science a Team Sport

- Organizes resources to achieve a particular data analysis goal
- Support role-based collaboration (Admin, Editor, Viewer)
- Assets from all IDEs can be included in one Watson Studio project: notebooks, data sources, flows, models, etc.
- Export/Import Projects

Watson Studio supports the Data Science Lifecycle

Build, train, deploy, and monitor at scale ML/DL workflows to infuse AI into the enterprise to drive innovation.



Watson Studio Community Card Features

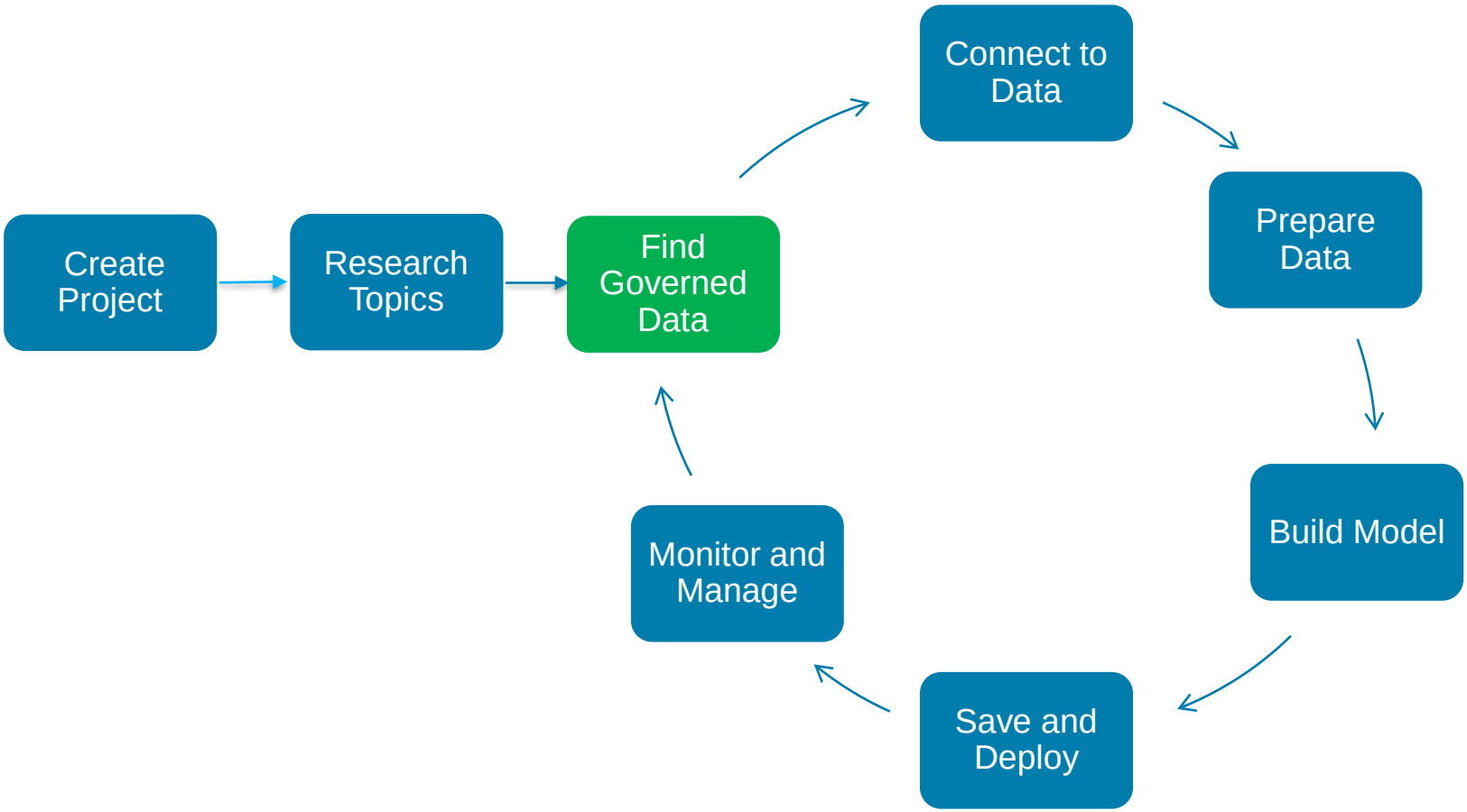
Built-in learning to get started

Research Topics

- Community Card Feature includes curated articles, tutorials, notebooks, data sets, and papers
- Bookmark in Projects
- Copy notebooks or Data Sets into projects
- Continuously updated in IBM's managed service

Watson Studio supports the Data Science Lifecycle

Build, train, deploy, and monitor at scale ML/DL workflows to infuse AI into the enterprise to drive innovation.



Watson Knowledge Catalog Features

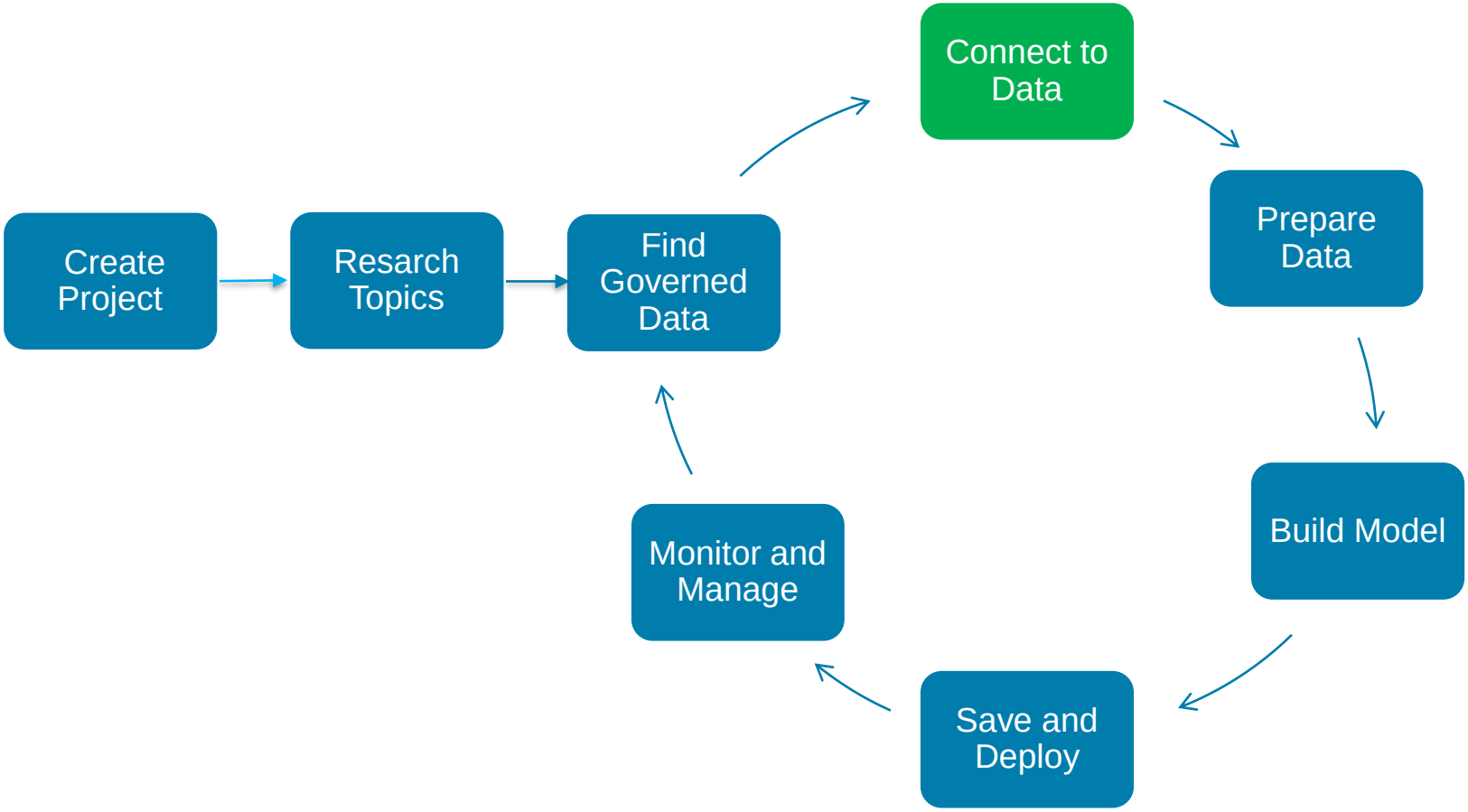
Find Governed Data

Unlock tribal knowledge and unleash knowledge workers

- **Find** data (structured, unstructured) and AI assets (e.g., ML/DL models, notebooks, Watson Data Kits) in the **Knowledge Catalog** with intelligent search and giving the right access to the right users.
- Discover assets, profiling, classification
- Policy, rule authoring
- Policy, rule enforcement
- Asset Usage Statistics

Watson Studio supports the Data Science Lifecycle

Build, train, deploy, and monitor at scale ML/DL workflows to infuse AI into the enterprise to drive innovation.



Watson Studio Connection Features

Connect to Data

- Upload files
- Connectors to Structured and Unstructured, On-prem and Cloud data sources.
- Wizard based connection definition and code generation

Connection Options

Connect to Data

New connection

IBM services

- BigInsights HDFS
- Cloud Object Storage
- Cloud Object Storage (infrastructure)
- Cloudant
- Cognos Analytics
- Compose for MySQL
- Compose for PostgreSQL
- Db2
- Db2 Big SQL
- Db2 for i
- Db2 for z/OS
- Db2 Hosted
- Db2 on Cloud
- Db2 Warehouse
- Informix
- Object Storage OpenStack Swift (infrastructure)
- PureData for Analytics
- Watson Analytics

Third-party services

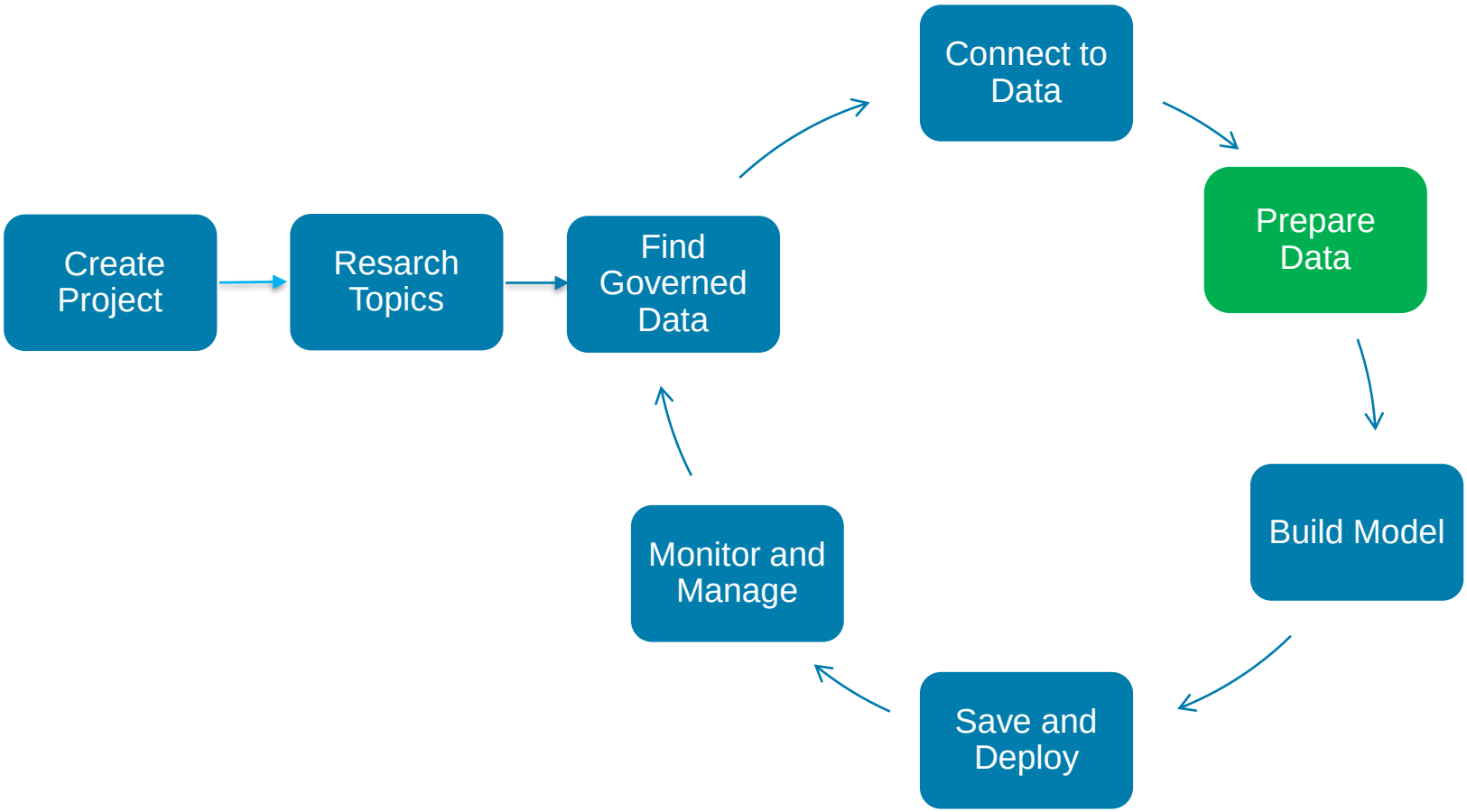
- Amazon Redshift
- Amazon S3
- Apache Hive
- Cloudera Impala
- Dropbox
- FTP
- Google BigQuery
- Google Cloud Storage
- Hortonworks HDFS
- Looker
- Microsoft Azure Data Lake Store
- Microsoft Azure SQL Database
- Microsoft SQL Server
- MySQL
- Oracle
- Pivotal Greenplum
- PostgreSQL
- Salesforce.com
- Sybase
- Sybase IQ
- Tableau
- Teradata

Notebook Screenshot

Connect to Data

Watson Studio supports the Data Science Lifecycle

Build, train, deploy, and monitor at scale ML/DL workflows to infuse AI into the enterprise to drive innovation.



Watson Studio Data Refinery Features

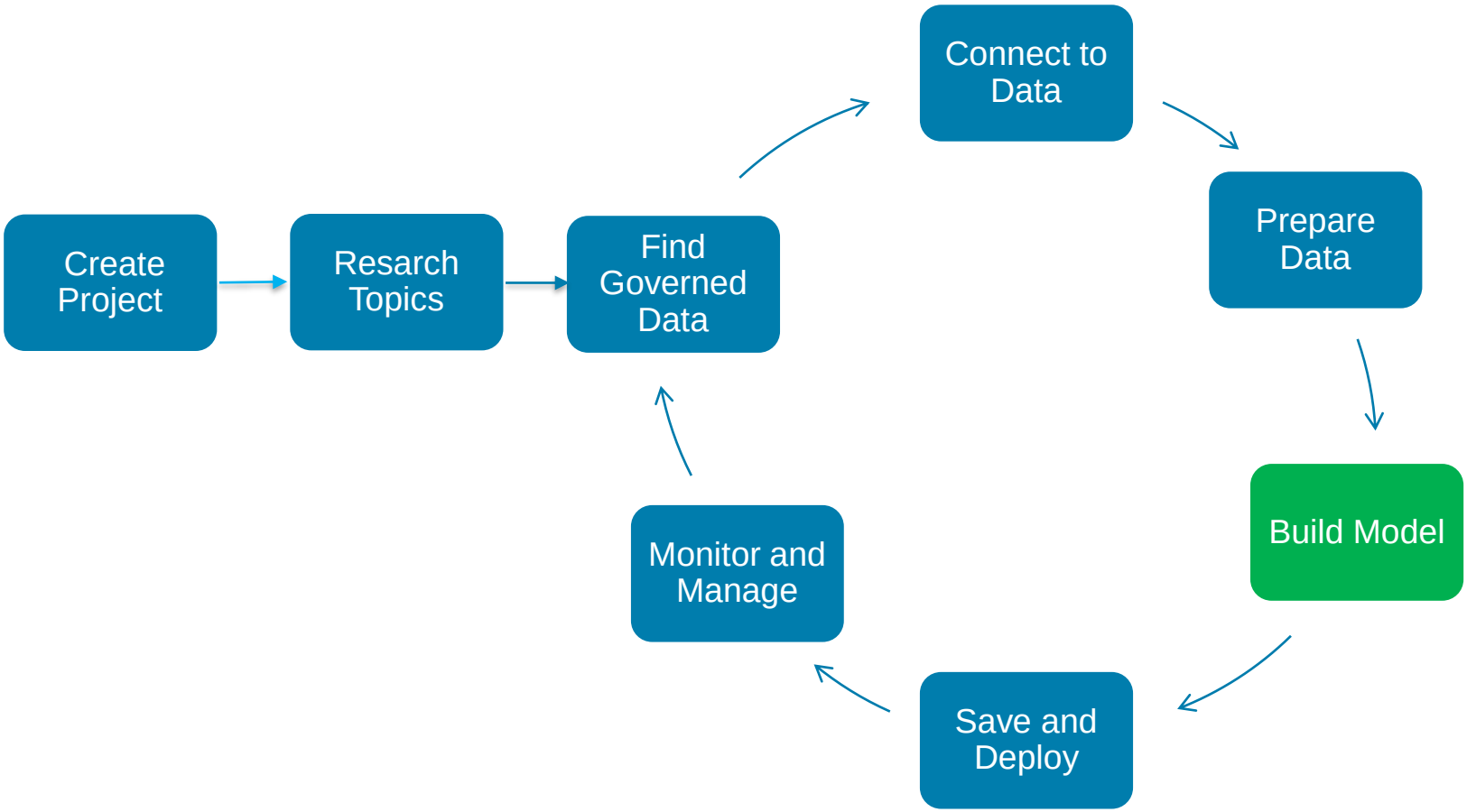
Making Data fit for use

Prepare Data

- Data Refinery tool to profile, visualize, and shape data.
- Create data preparation pipelines via point and click capability on subset of data
  - ✓ Cleanse the data: fixing or removing data that is incorrect, incomplete, improperly formatted, or duplicated
  - ✓ Shape the data: customize data by filtering, sorting, combining, or removing columns, and performing operations
- Run the pipeline on all the data
  - Manually (on demand)
  - Automated (scheduled)

Watson Studio supports the Data Science Lifecycle

Build, train, deploy, and monitor at scale ML/DL workflows to infuse AI into the enterprise to drive innovation.



Watson Studio Model Building Features

Build Model

The best of open source and IBM Watson tools to create start-of-the-art data products

Open Source Tools

- Jupyter Notebooks**
- RStudio and Shiny
- Libraries- scikit-learn**, XGBoost**, Spark, TensorFlow**, Caffe, Keras, PyTorch

IBM Tools

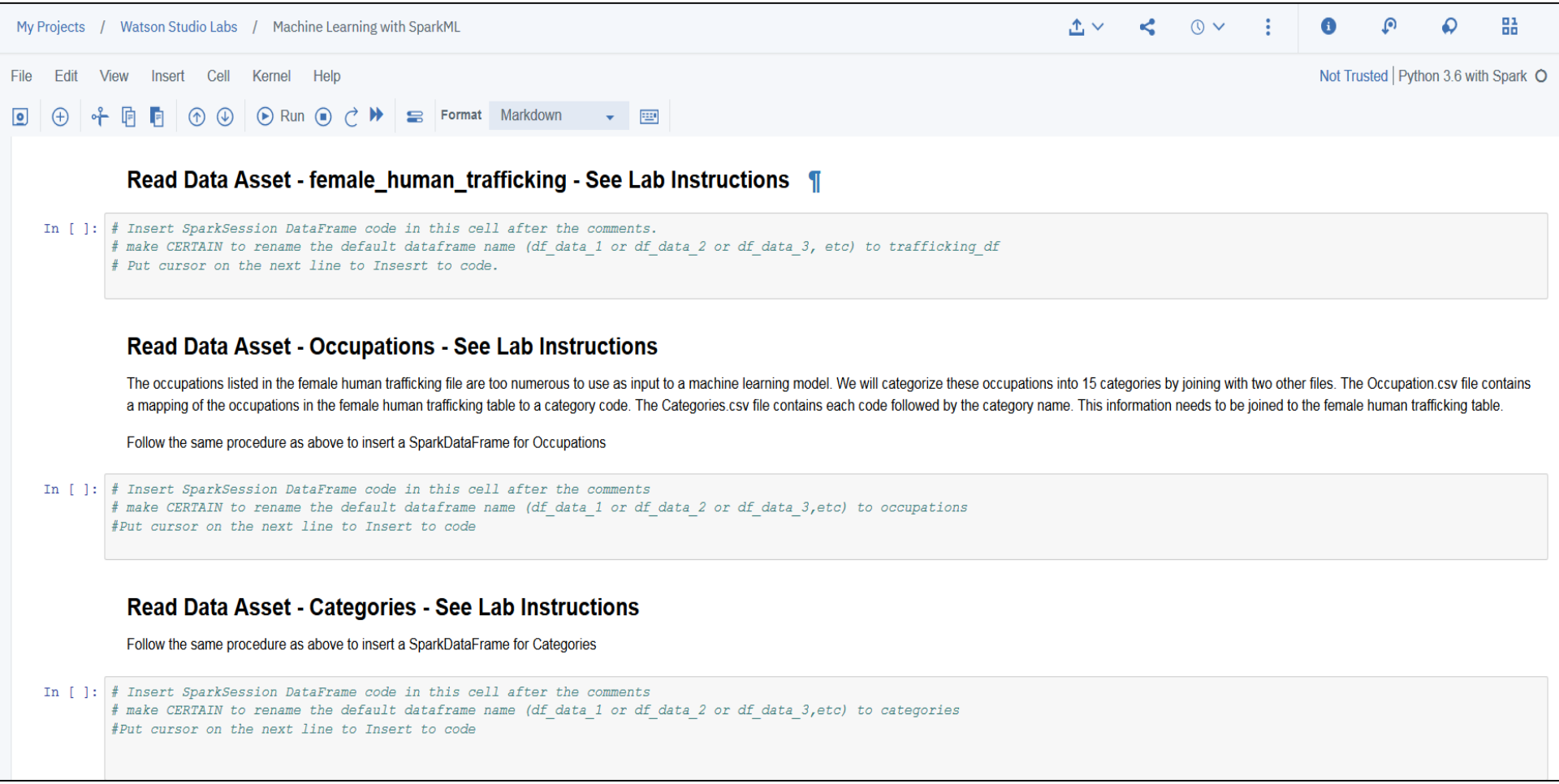
- AutoAI **
- SPSS Modeler**
- Neural Network Modeler**
- Experiment Builder**
- Natural Language Classifier Model
- Visual Recognition Model

Train at scale on **GPUs** and **distributed** compute

** in hands-on labs

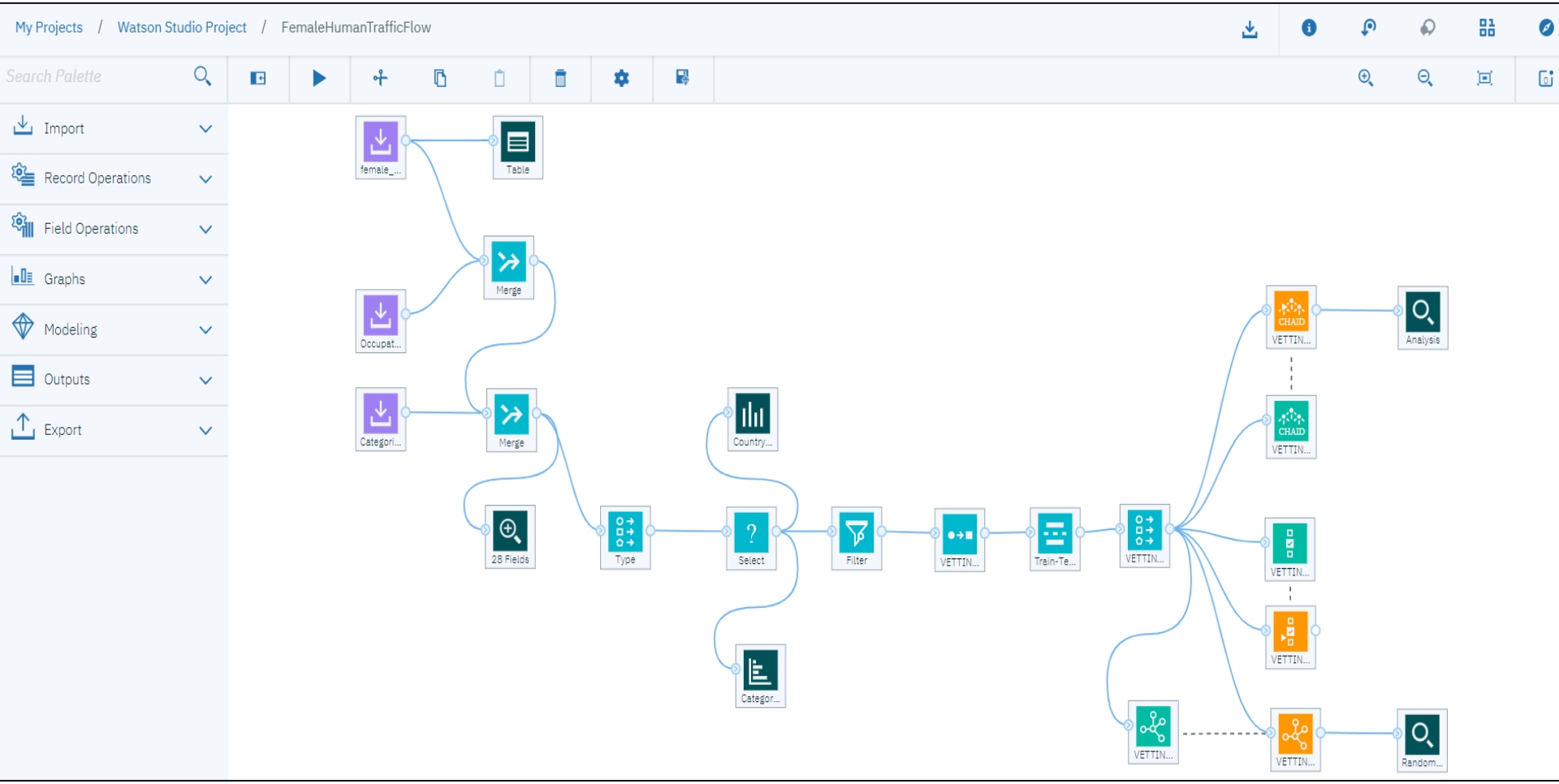
Jupyter Notebook

Build Model



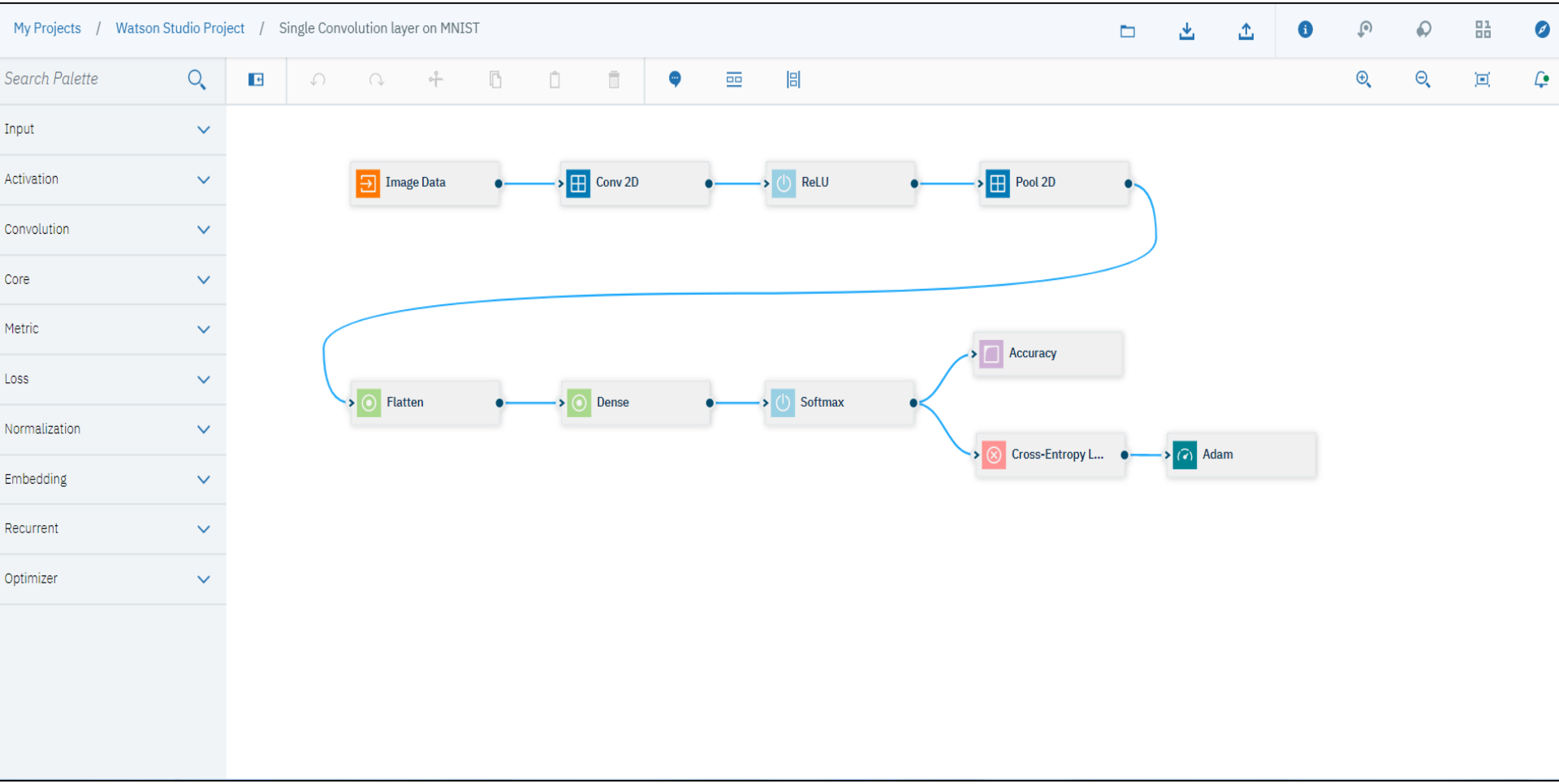
SPSS Modeler

Build Model



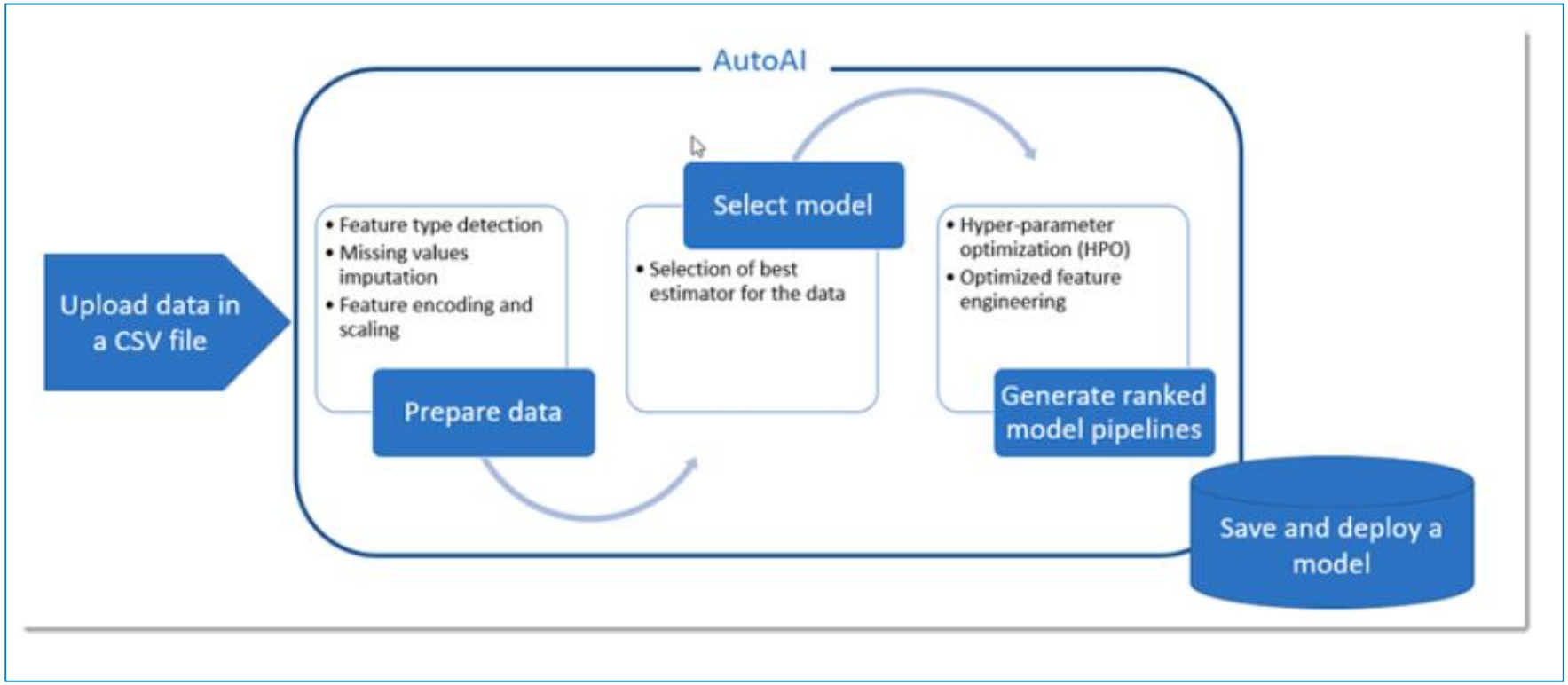
Neural Network Modeler

Build Model



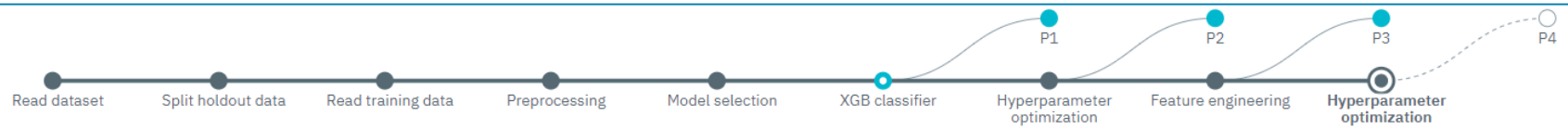
AutoAI

Build Model



AutoAI

Build Model



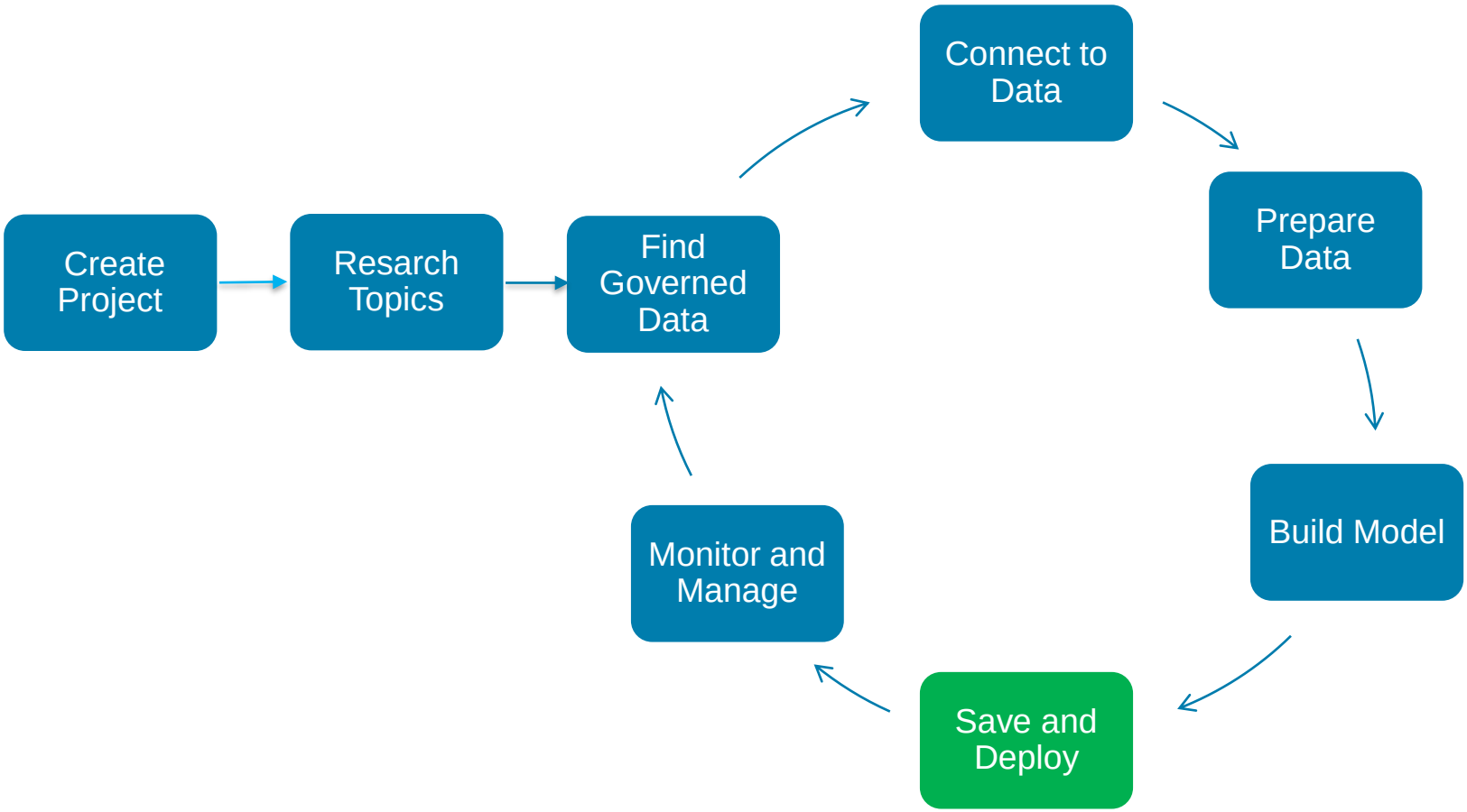
Pipeline leaderboard

Compare pipelines | Ranking based on: Accuracy

	RANK	ACCURACY	PIPELINE INFORMATION		
>	1	0.897	P3 - XGB classifier estimator Transformers (8): Preprocessing > Standard scaler > Univariate feature selection > Sine > Univariate feature selection > Tangent > ...	View details	Save as model
>	2	0.884	P1 - XGB classifier estimator Transformers (2): Preprocessing > XGB classifier estimator	View details	Save as model
>	3	0.884	P2 - XGB classifier estimator Transformers (2): Preprocessing > XGB classifier estimator	View details	Save as model

Watson Studio supports the Data Science Lifecycle

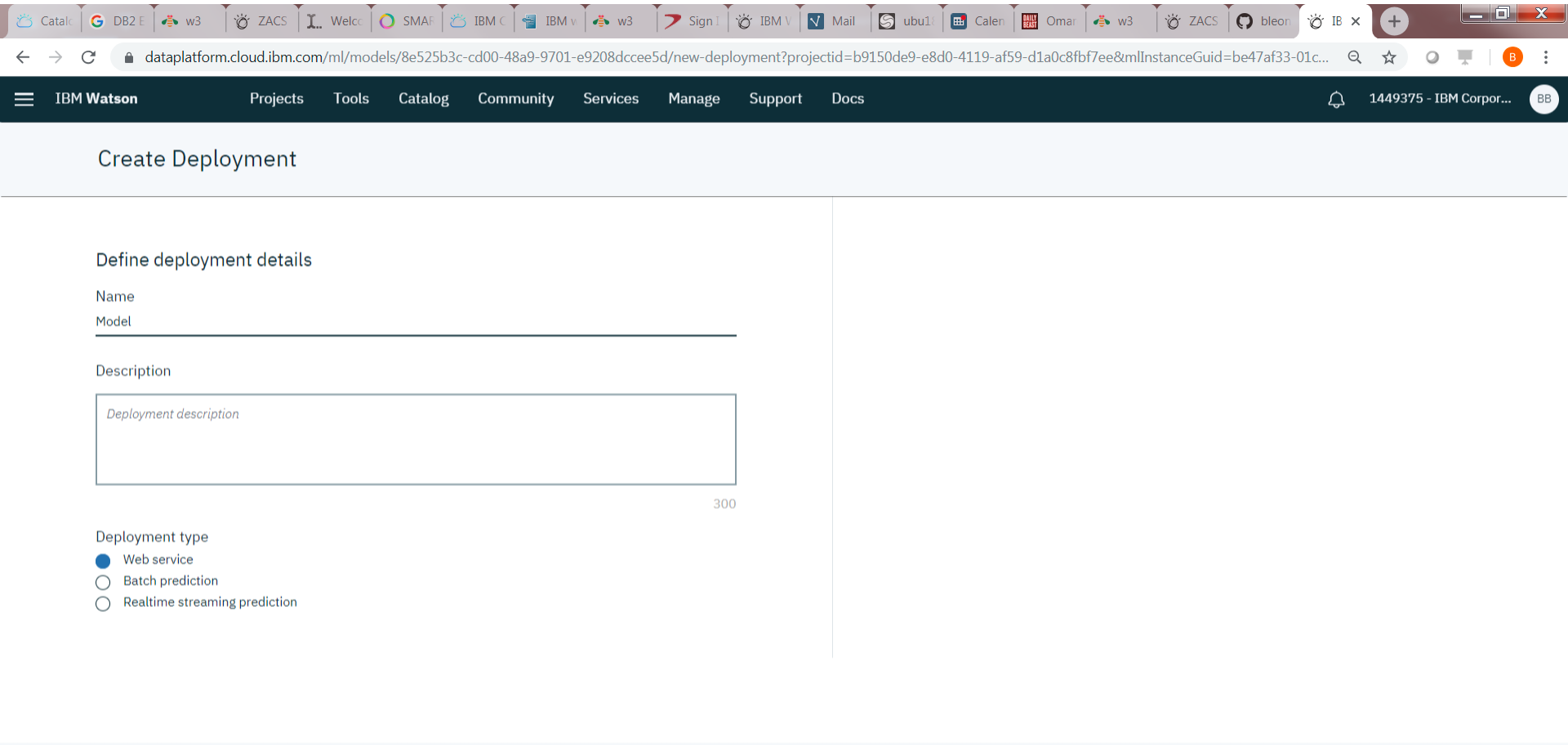
Build, train, deploy, and monitor at scale ML/DL workflows to infuse AI into the enterprise to drive innovation.



Watson Studio Save and Deploy Models

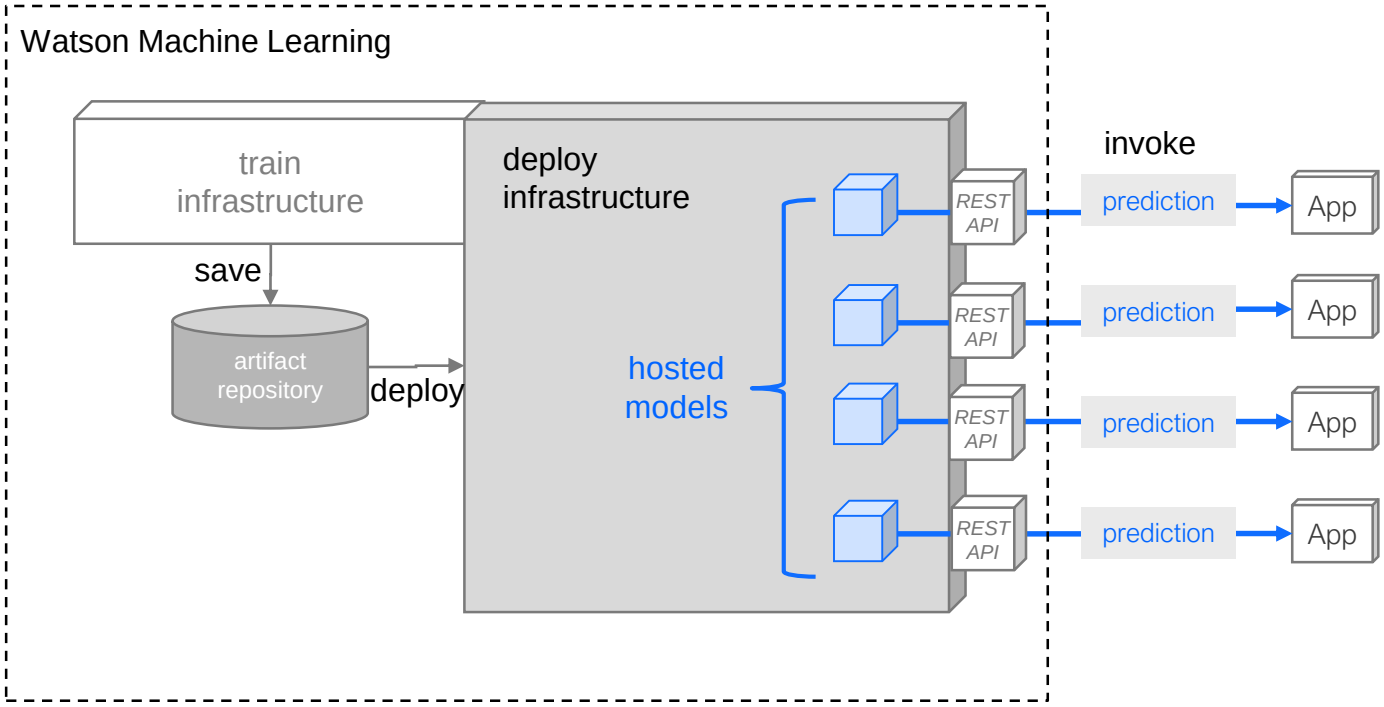
Save and Deploy Models with Watson Machine Learning

Save and Deploy



Watson Studio Save and Deploy Trained Models

Save and Deploy Models with Watson Machine Learning



Watson Studio Save and Deploy Features

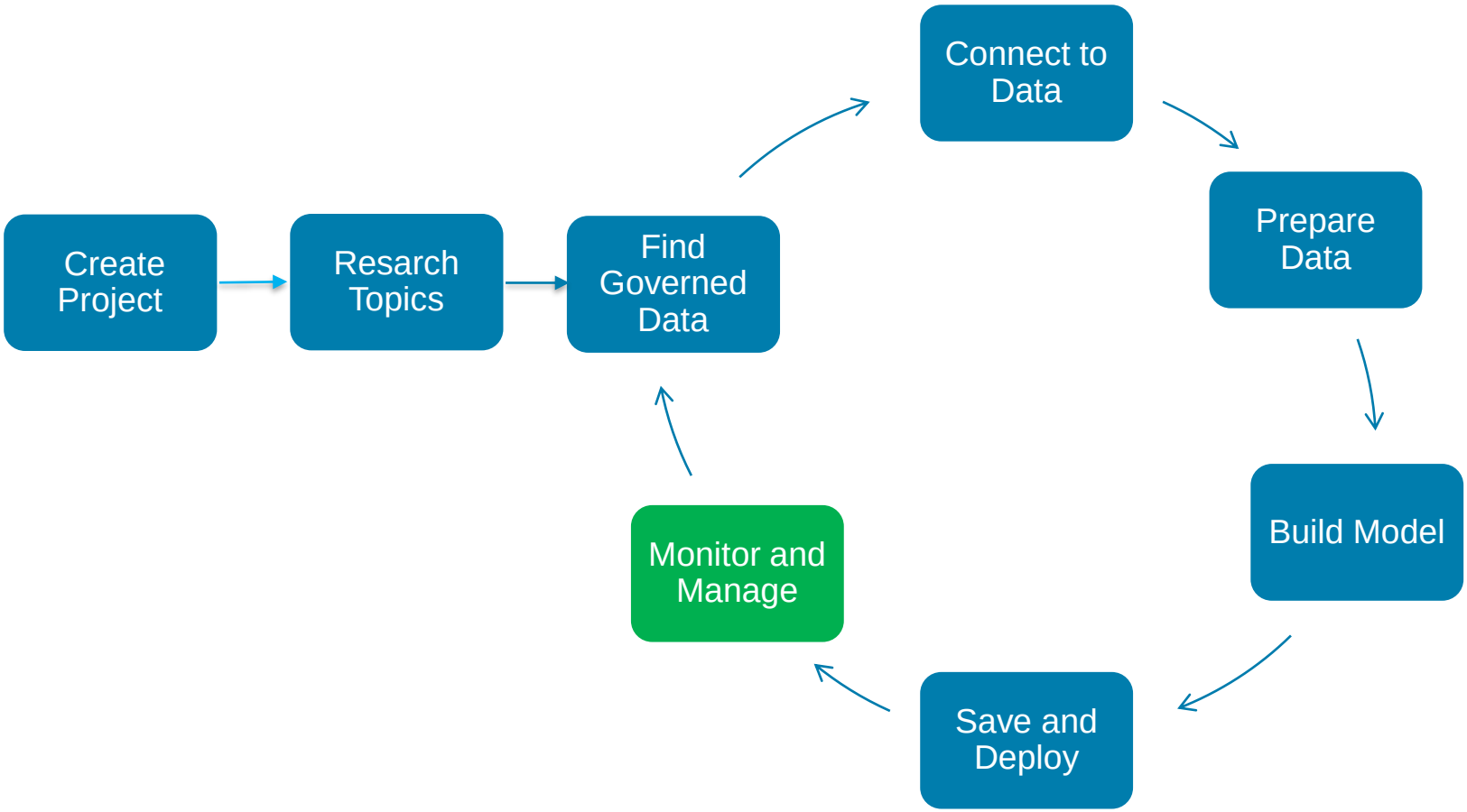
Save and Deploy

Save and Deploy Models with Watson Machine Learning

- Watson Machine Learning API to save/load models to/from repository
- Watson Machine Learning API to deploy saved models easily and have them scale automatically.
- Watson Machine Learning API to invoke deployed models

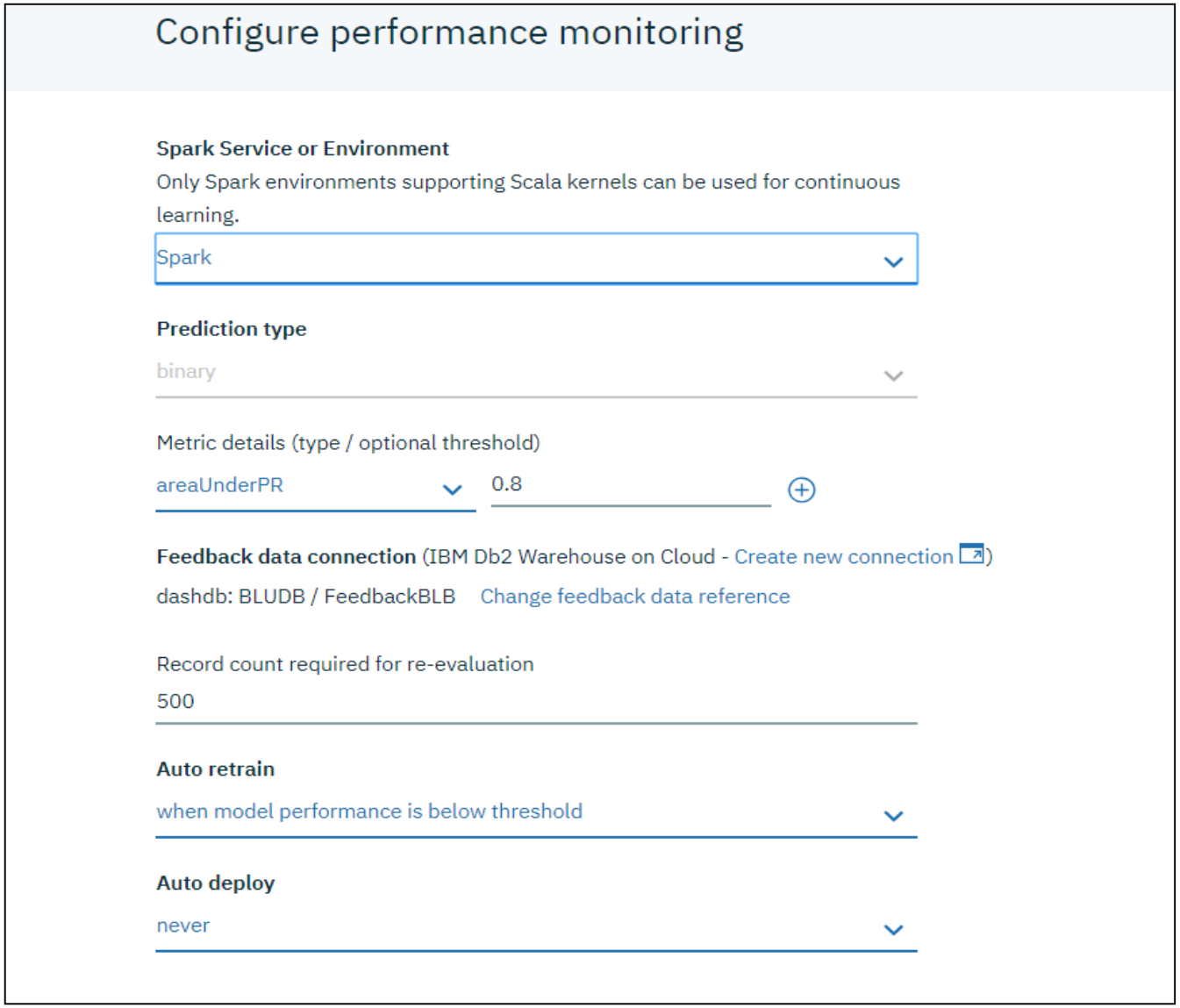
Watson Studio supports the Data Science Lifecycle

Build, train, deploy, and monitor at scale ML/DL workflows to infuse AI into the enterprise to drive innovation.



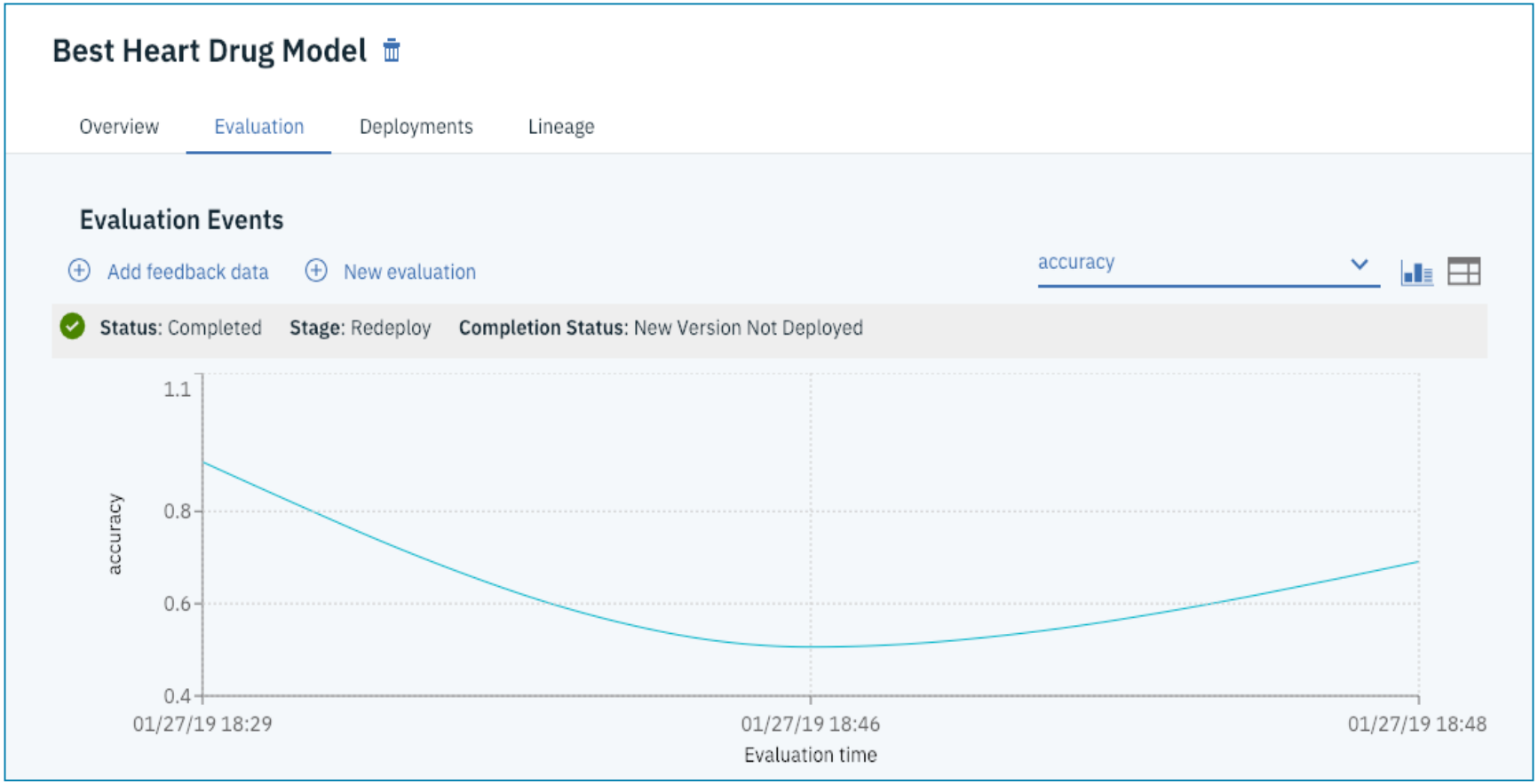
Watson Studio Monitor and Manage

Monitor and Manage



Watson Studio - Monitor and Manage

Monitor and Manage



Watson Studio Monitor and Manage Features

Monitor and Manage

- Monitor the performance of the models in production and trigger automatic retraining and redeployment of models.

Watson OpenScale

Monitor and Manage

Trust and Transparency

- Intelligently delivers bias mitigation help
- Provides traceability & auditability of AI predictions made in production applications
- Tracks AI accuracy in applications
- Explains an outcome in business terms

Automation

- Automatically detects and mitigates bias in model output, without affecting currently deployed model or outcomes
- *NeuNetS (beta) – automatically generate Neural Networks

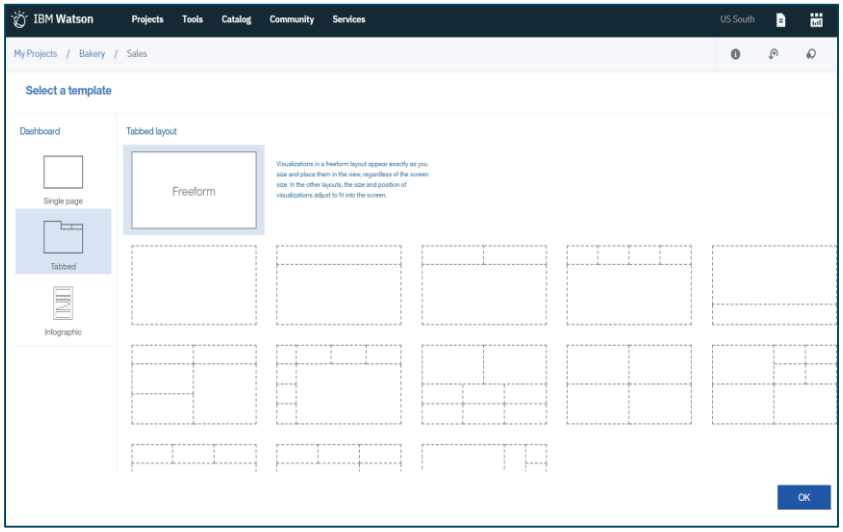
Open By Design

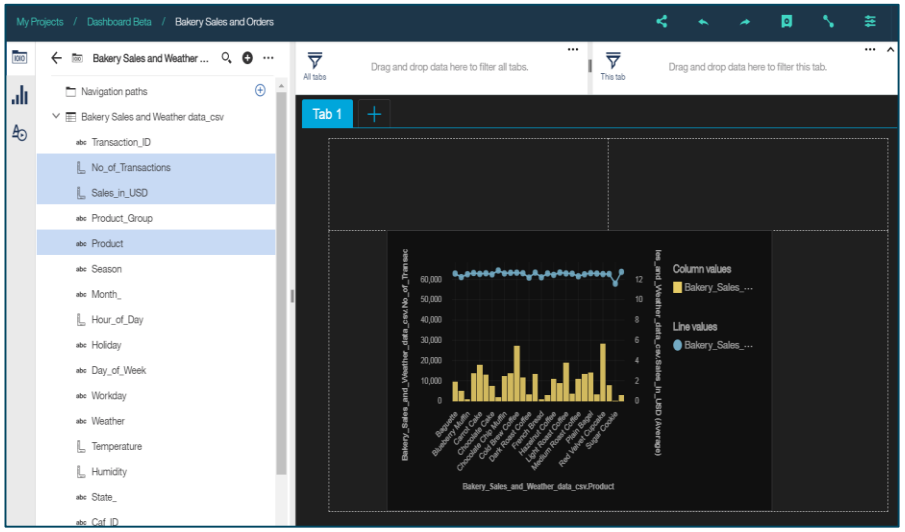
- Monitor models deployed on third party model server engines
- Deploy behind enterprise firewall or on IaaS provider

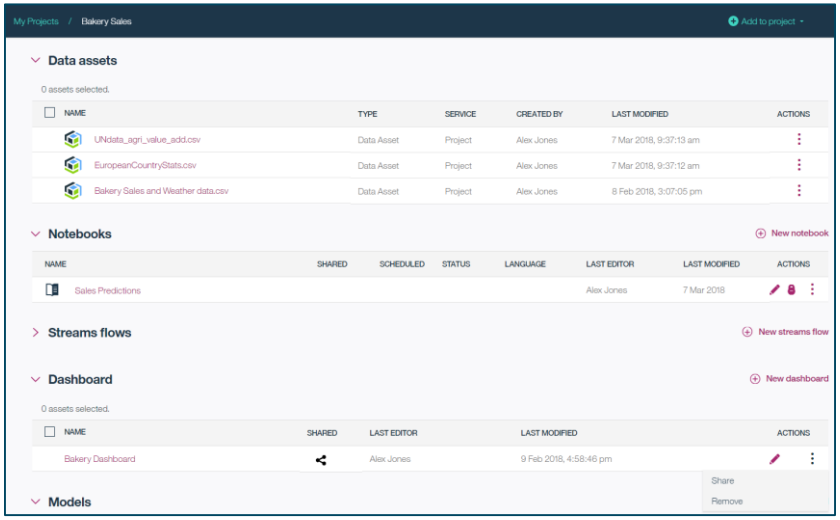
* https://arxiv.org/abs/1901.06261

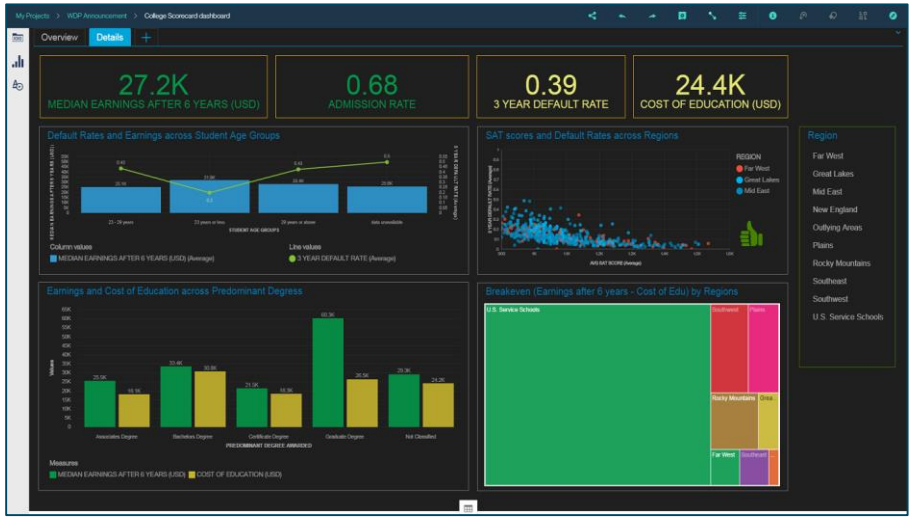
Watson Studio Dynamic Dashboards

Making insights available to all









Watson Studio Takeaways

Integrated Collaboration Environment

- Data Scientists, Subject Matter experts, Business Analysts & Developers all in one environment to accelerate innovation, collaboration and productivity
- Built-in learning to get started or go the distance with advanced tutorials

Choice of Tools for the full AI lifecycle

- Best in-breed open source and IBM tools that support the end-to-end AI lifecycle
- Choice of code or no-code tools to build and train your own ML/DL models or easily train and customize pre-trained Watson APIs

Support for all levels of expertise

- Use Watson smarts and recommendations for the best algorithms to use given your data, OR
- Use the rich capabilities and controls to fine tune your models

Multiple Deployment Options

- Watson Studio on IBM Cloud – Managed offering
- Watson Studio Local – Private Cloud, Public Cloud-(IBM, Azure, AWS)
- Watson Studio Desktop

Model lifecycle & management

- Deploy models into production then monitor them to evaluate performance.
- Capture new data for continuous learning and retrain models so they continually adapt to changing conditions.

Integrated with Knowledge Catalog

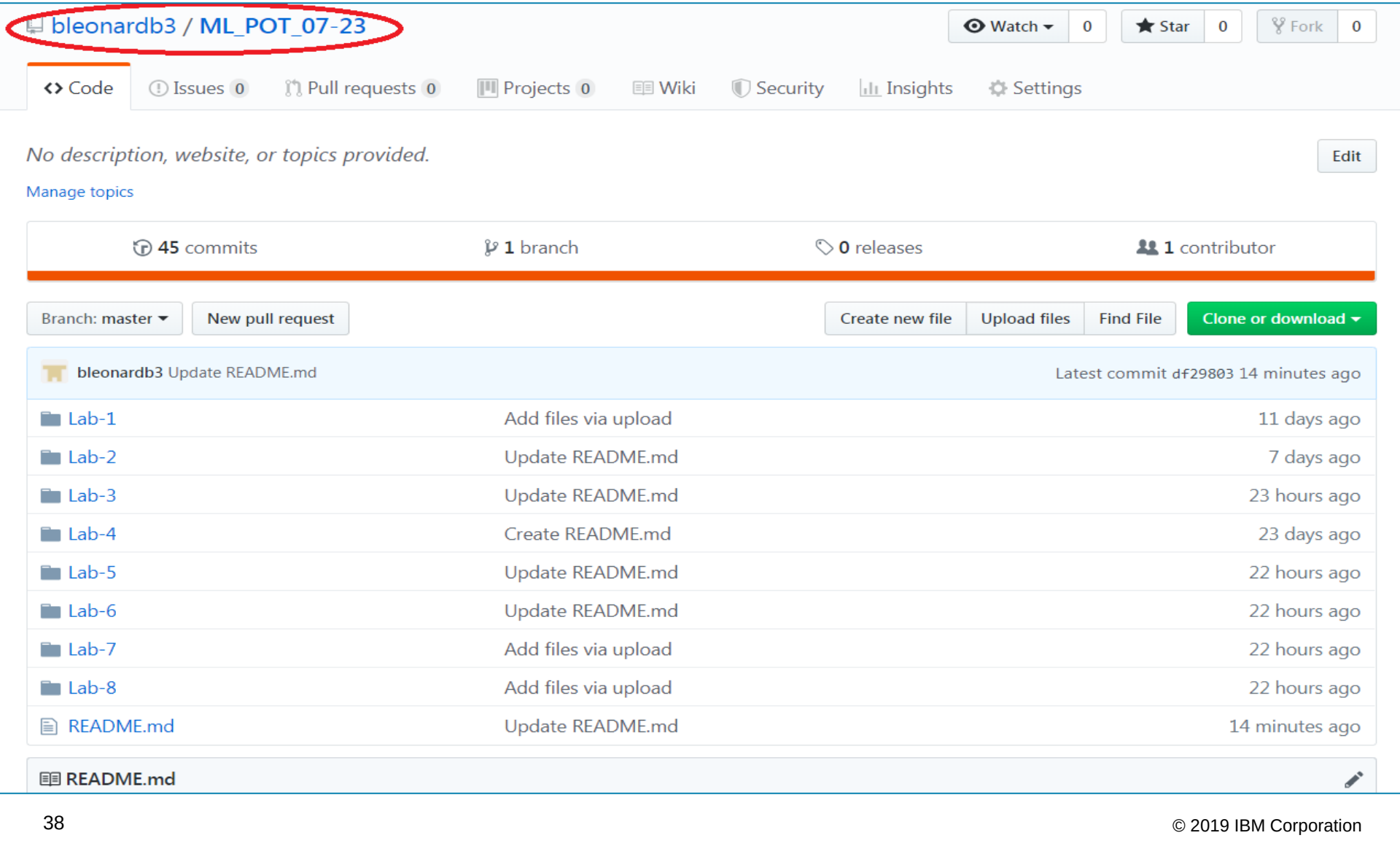
- Intelligent discovery of data and AI assets that enables reuse & improves productivity
- Seamlessly integrated for productive use with Machine Learning and Data science
- Powerful governance tools to control and protect access to data

36

Outline

- **Watson Studio Overview**
- **Lab Overview**

Github Repository



Github Repository

Readme

Hands-on Introduction to Machine Learning using Watson Studio

Description:

Work with IBM's Watson Studio in this workshop to build, train, and test machine learning/deep learning models. Participants will be led through the following nine hands-on labs. Note, the first lab is a prerequisite for the other labs. Once Lab-1 is completed, the other labs can be done in any order.

1. Lab-1 - This lab will set up the environment for the subsequent labs.
2. Lab-2 - This lab will use a Jupyter Notebook and the XGBoost library to apply machine learning to a classification problem in the healthcare profession. The Watson Machine Learning API will then be used to save and deploy the model.
3. Lab-3 - This lab will feature the Watson Studio Neural Network modeler, and Experiment Assistant to build, train, and test a Convolutional Neural Network to classify images of handwritten digits.
4. Lab-4 - This lab will feature IBM's Adversarial Robustness Toolbox (ART). ART is a library dedicated to adversarial machine learning. Its purpose is to allow rapid crafting and analysis of attacks and defense methods for machine learning models. ART provides an implementation for many state-of-the-art methods for attacking and defending classifiers.
5. Lab-5 - This lab consists of two parts. The first part will demonstrate the new and exciting AutoAI capability to build and deploy an optimized model based on the Titanic data set. The second part will deploy an application using the IBM Cloud DevOps toolchain that will invoke the deployed model to predict whether a passenger would have survived.
6. Lab-6 - This lab will feature the Watson Studio Data Refinery to demonstrate data profiling, visualization, and data preparation.
7. Lab-7 - This lab will feature the Watson Studio SPSS modeler to demonstrate visual drag and drop creation of a machine learning model.
8. Lab-8 - This lab will feature the Watson Studio Cognos Dashboard Embedded Service.

Github Repository

Lab-1 Readme

Lab-1 - Setup Environment

Introduction:

This lab will set up the Watson Studio environment for subsequent labs and introduce you to the Project features of Watson Studio. Watson Studio is an integrated platform of tools, services, data, and meta-data to help companies and agencies accelerate their shift to be data driven organizations. The platform enables data professionals such as data scientists, data engineers, business analysts, and application developers collaboratively work with data to build, train, deploy machine learning and deep learning models at scale to infuse AI into business to drive innovation. Watson Studio is designed to support the development and deployment of data and analytics assets for the enterprise.

Objectives:

Upon completing the lab, you will:

1. Create a project
2. Create an object storage instance and associate it with the project
3. Associate an existing Watson Machine Learning service instance with the project
4. Add a collaborator to the project

Step 1. Please click on the link below to download the instructions to your machine.

Instructions.

Lab Tips

- Watson Studio url: datascience.ibm.com
- Labs are in www.github.com/bleonardb3/ML_POT_07-23 repository.
- Instructions for each Lab are in the README file in the respective Lab folder.
- Cloud development enables making frequent improvements in the user interface. We reviewed the lab instructions and made screen updates so they should be pretty faithful to the user interface. Small differences may occur but shouldn't get in the way of successfully completing the labs.
- Do not use Internet Explorer as the browser. For Mac users do not use Safari.
- All of the Labs should be done in the project that you created when following the signup instructions.
- For Lab-1 make sure that you uncheck the "restrict who can be a collaborator" checkbox when creating the project.
- For Lab-2 make sure when you are creating the notebook that you switch the environment to the Python-Spark environment.

Lab-1: Set up Environment

Introduction:

This lab will set up the Watson Studio environment for subsequent labs and introduce you to the Project features of Watson Studio.

Objectives:

Upon completing this lab, you will know how to:

- Create a project
- Create an object storage instance and associate it with the project
- Add a collaborator to the project

Lab-2: Build a Heart Disease Prediction Model

Introduction:

In this lab, you will use a Jupyter Notebook to train a model using the XGBoost library to classify whether a person has heart disease or not. In addition to training a model, the notebook also explains how to persist a trained model to the IBM Watson Machine Learning repository, and deploy the model as a REST service.

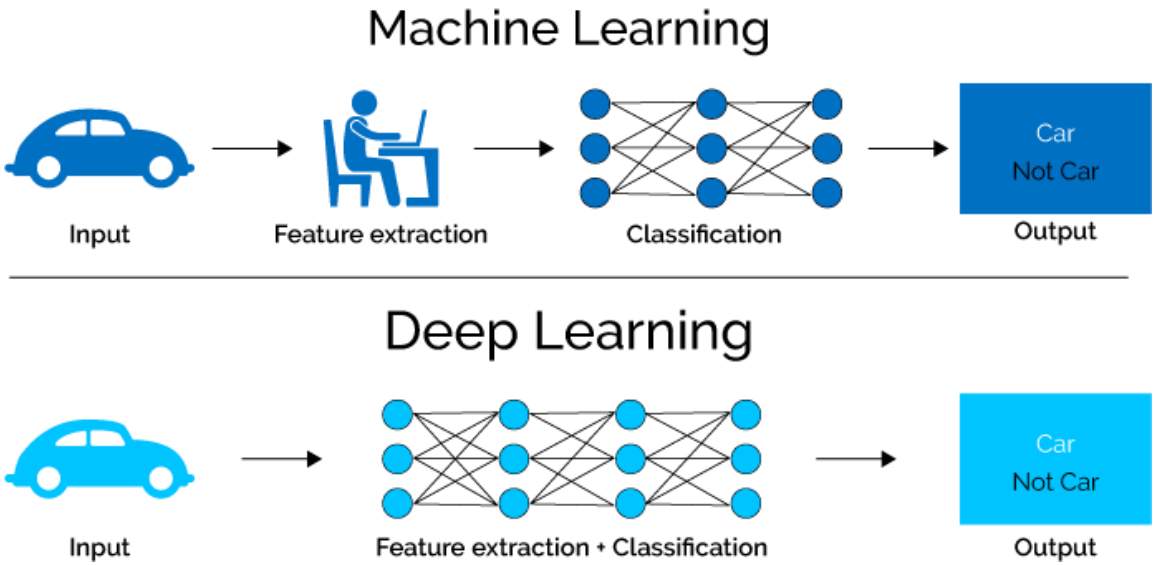
Objectives:

Upon completing the lab, you will know how to:

- Load a CSV file into Pandas DataFrame.
- Prepare data for training and evaluation.
- Create, train, and evaluate a XGBoost model.
- Visualize the importance of features that were used to train the model.
- Use cross validation to select optimal model hyperparameters based on a parameter grid
- Persist best model in Watson Machine Learning repository using Python client library.
- Deploy the model for online scoring using the Watson Machine Learning's REST APIs

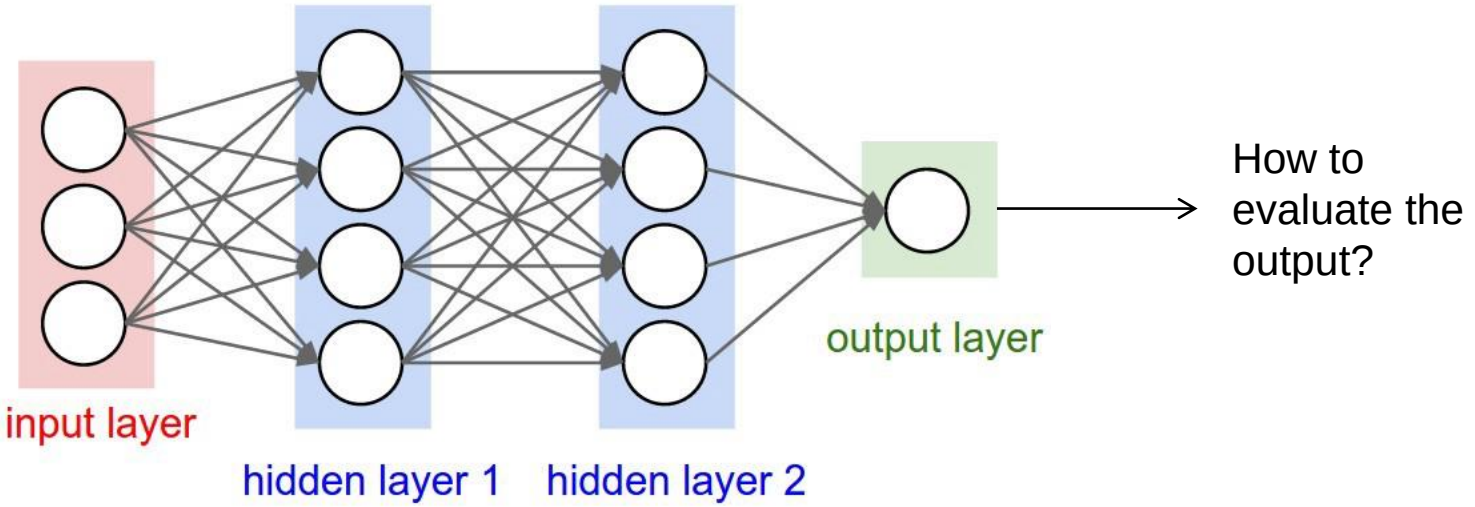
Deep Learning

- Deep Learning is a machine learning method.
- Could be supervised or unsupervised
- Originated in 1940s
- Became very popular this decade
  - Hardware Improvements/Cost – GPUs, Storage
  - Availability of Large Datasets for Training
  - Better performing algorithms.
- Especially good for human perception type task



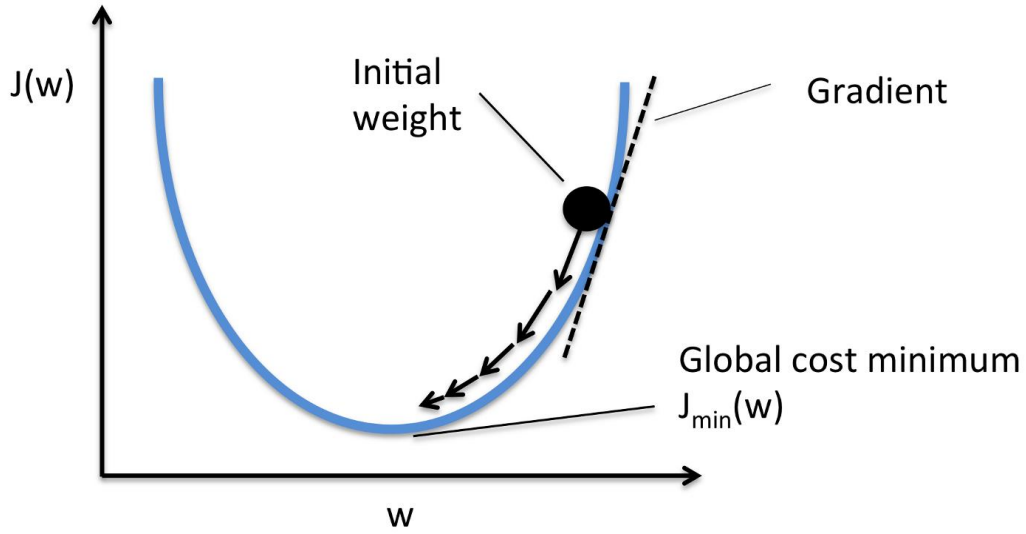
Deep Neural Networks (DNN)

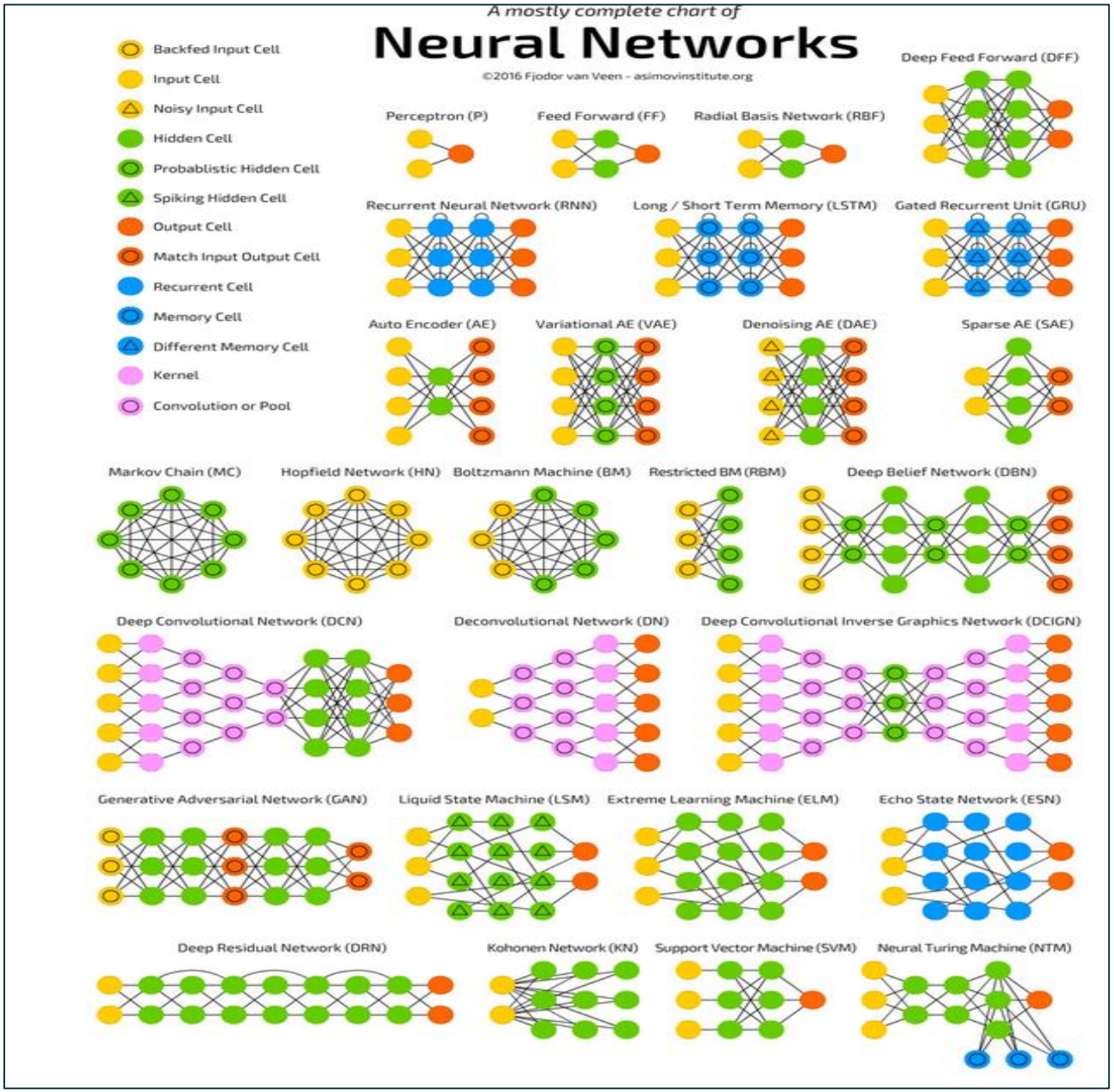
- **A Neural Network with <u>more than one</u> hidden layer**
  - The **input layer** receives input data.
  - The **hidden layers** perform mathematical computations on our inputs.
  - The **output layer** returns the output data.
- Training the AI is the hardest part of Deep Learning.
  - You need a large data set.
  - You need a large amount of computational power



Cost Function

- During training we need to know how our DNN is doing!
  - Compare it's predictions to dataset's output
  - Based on how far it is from actual value → update weights
- This function that does this is called "Cost Function"
- Ideally, we want our cost function to be zero.
  - Does not happen in real world
  - Instead use techniques like "Gradient Descent" → allows us to find the minimum of a function by iterating through dataset and updating the weights





Common Types of Deep Neural Networks

Convolutional Neural Networks

- Images recognition
- Images classifications
- Objects detections
- Recognizing faces
- Natural language processing
- ..

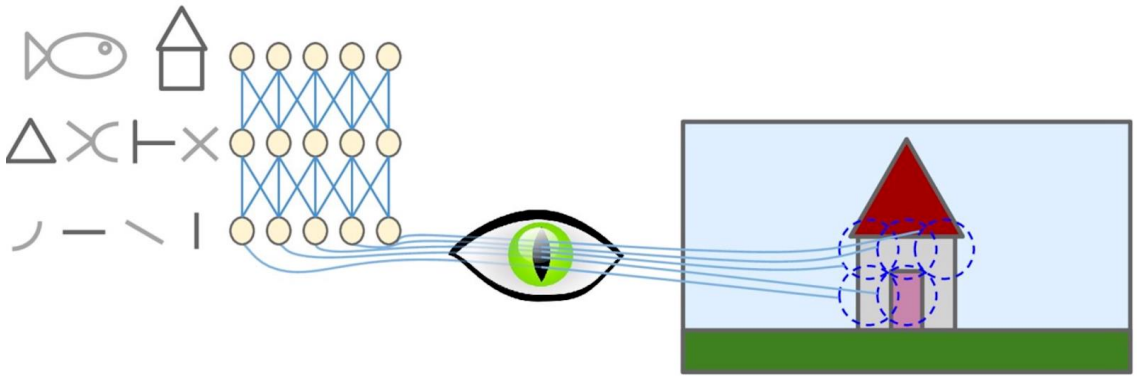
Recurrent Neural Networks

- Speech Recognition
- Handwriting Recognition
- Machine Translation
- Sequence prediction
- …

Convolutional Neural Networks (CNN)

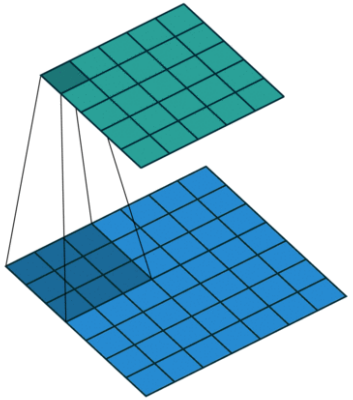
- Inspired by the architecture of the visual cortex → convolutional neural network
- Some neurons only react to horizontal lines, while others reacts to lines with different orientations
- Some neurons have larger receptive fields, so they react to more complex patterns based on output of lower-level neuron

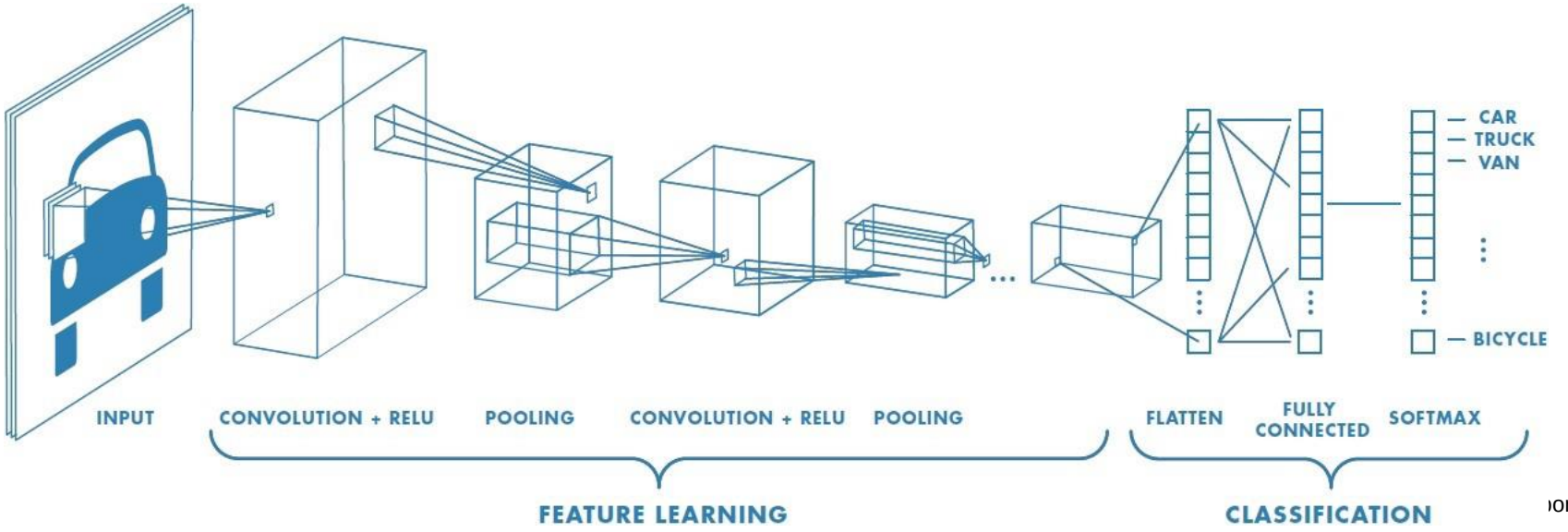
- Two building blocks: **Convolutional** layers and **Pooling** layers



Convolutional Layer

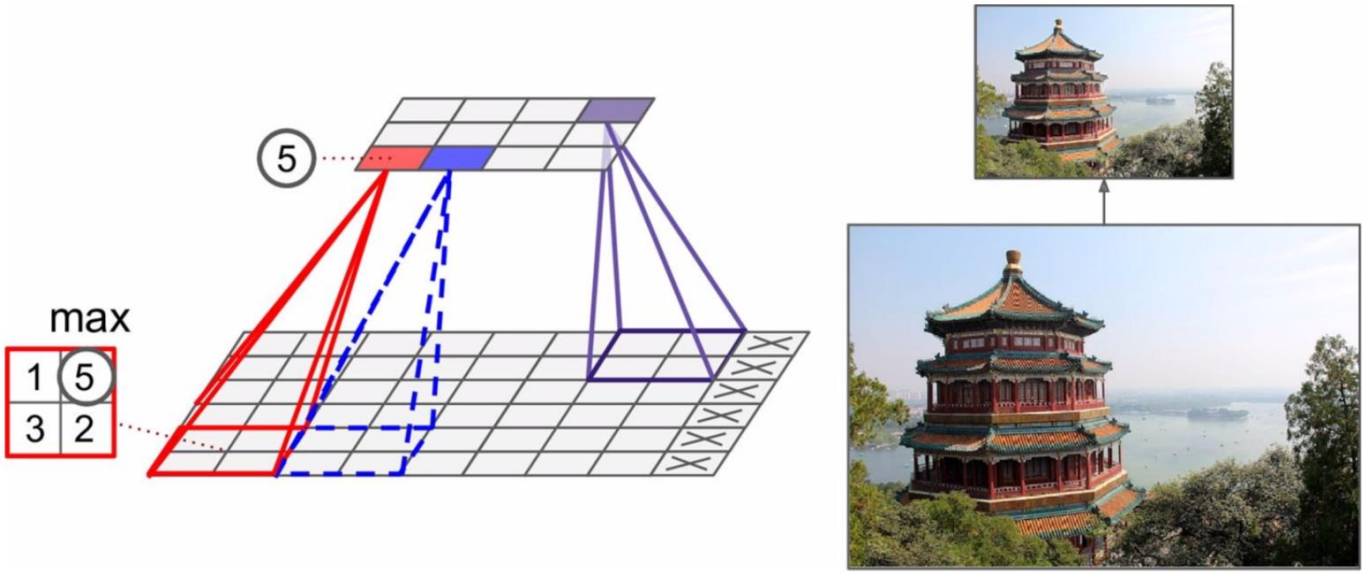
- Convolution is the first layer to extract features from an input image.
- Neurons in the first convolutional layer are not connected to every pixel in the input image, but **only to pixels in their receptive fields**
- Each neuron in the second convolutional layer is connected only to **neurons located within a small rectangle** in the first layer.
- Convolution preserves the relationship between pixels by learning image features using small squares of input data.



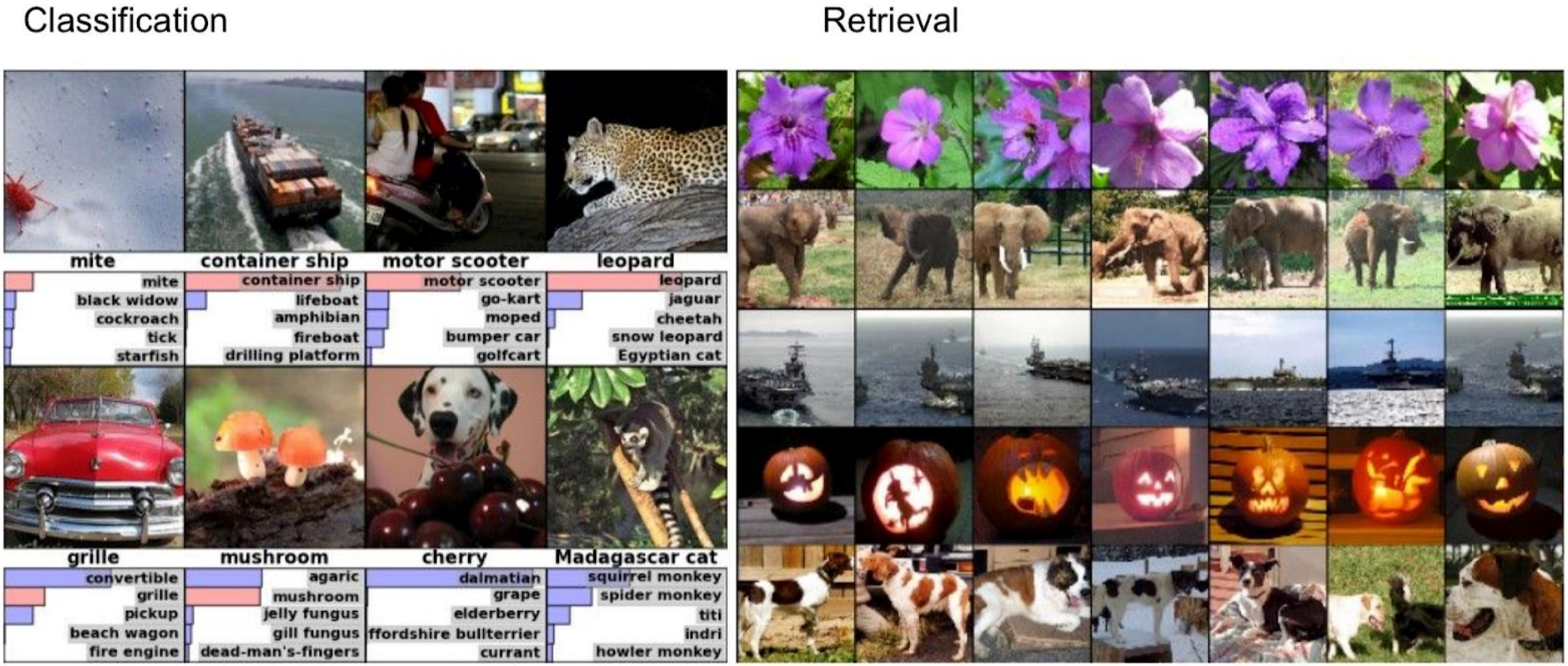


Pooling Layer

- Pooling Layer reduce the number of parameters when the images are too large.
  - Max Pooling
  - Average Pooling
  - Sum Pooling

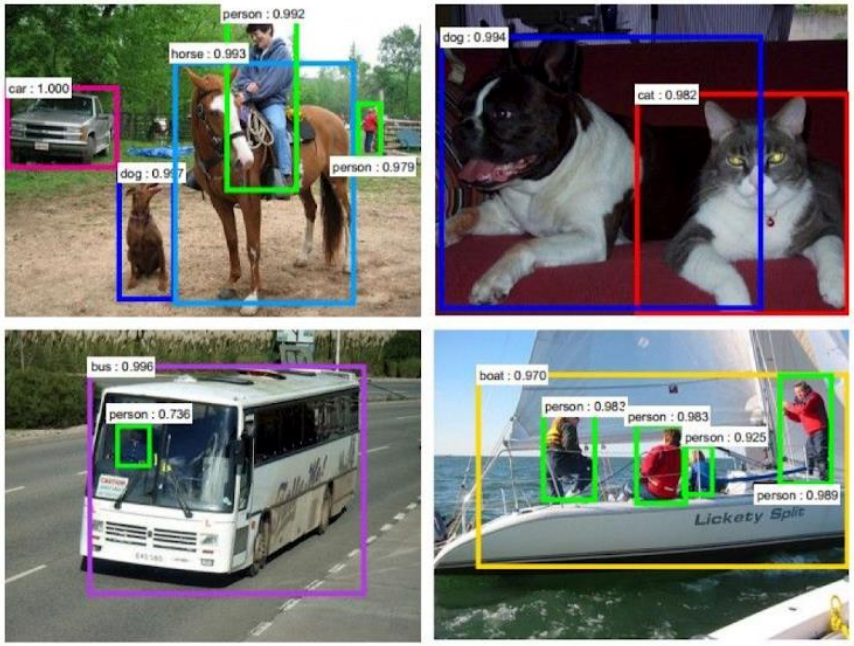


CNN Applications



CNN Applications

Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

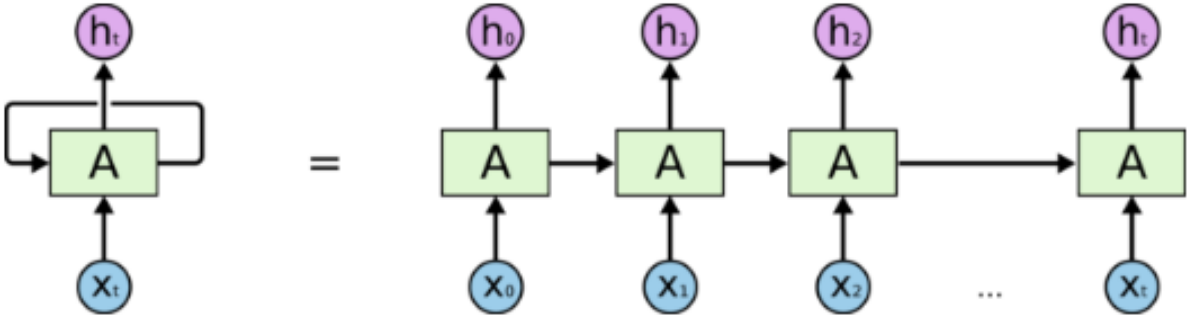
Segmentation

Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

Recurrent Neural Networks (RNN)

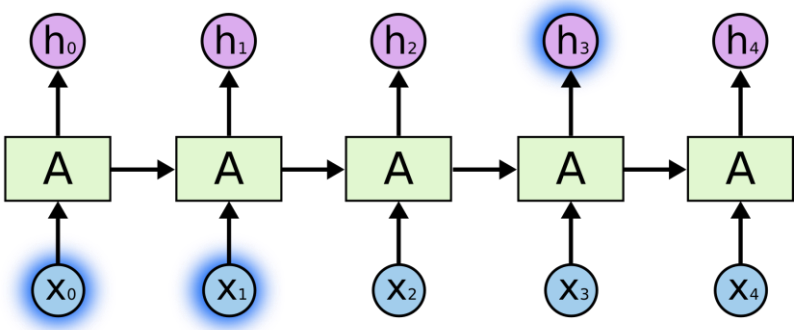
- Humans don't start their thinking from scratch every second. We rely on our memory!
- Traditional Neural Networks **CAN NOT** help → data flows forward only
- Recurrent Neural Networks address this issue.
  - They are networks with loops in them, allowing information to persist.
- RNN Applications are: Speech recognition, Language modeling, Translation



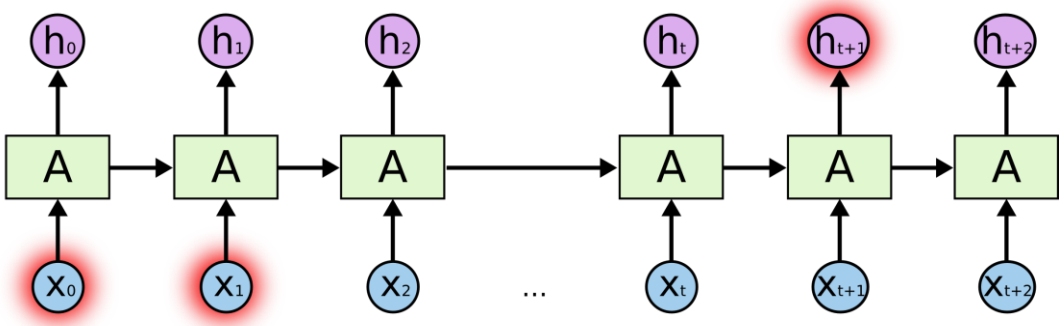
An unrolled recurrent neural network.

Long-Term Dependencies Problem

- RNNs can look at old information. But how old?
- Successfully uses recent information → Clouds are in the … [Sky]



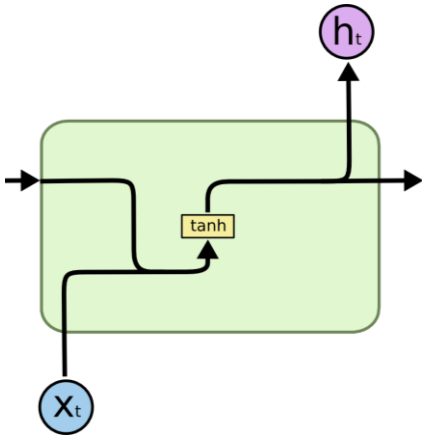
- Can NOT use older information → I grew up in France, in a small city near Paris, so I speak fluent … [??] **But why?**



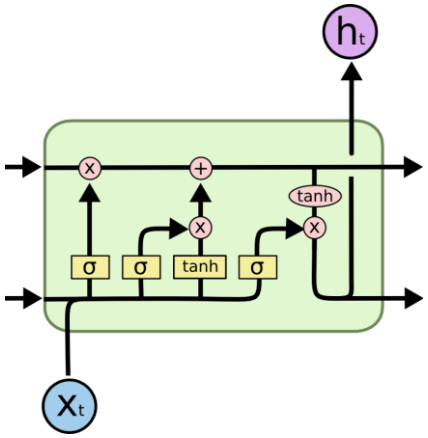
Long-Term Dependencies Problem

- **Vanishing gradients**
  - 0 < Gradient values < 1 → they leave the connection weights **unchanged**
- **Exploding gradients**
  - Calculated gradients are large values → many layers got **insanely large** weight updates, and the network becomes unstable and diverged

- **Long Short Term Memory (LSTM) fixed the gradient problem**
  - by introducing a few more gates that control access to the cell state



The repeating module in RNN contains a single layer.



The repeating module in LSTM contains four interacting layers

Lab-3: Recognizing Handwritten Digits

Introduction:

This lab will use the MNIST computer vision data set to train a convolutional neural network (CNN) model to recognize handwritten digits. The Watson Studio neural network flow editor, Watson Studio experiment builder and the Watson Machine Learning component will be used to build, train, and save the trained model.

Objectives:

Upon completing the lab, you will know how to:

- Create Cloud Object Storage buckets to contain the input and result files
- Create a neural network design from an example using the flow editor
- Use the experiment builder used to set up a training definition to train the neural network model
- Monitor the training progress and results.
- Save the trained model.
- Test the model

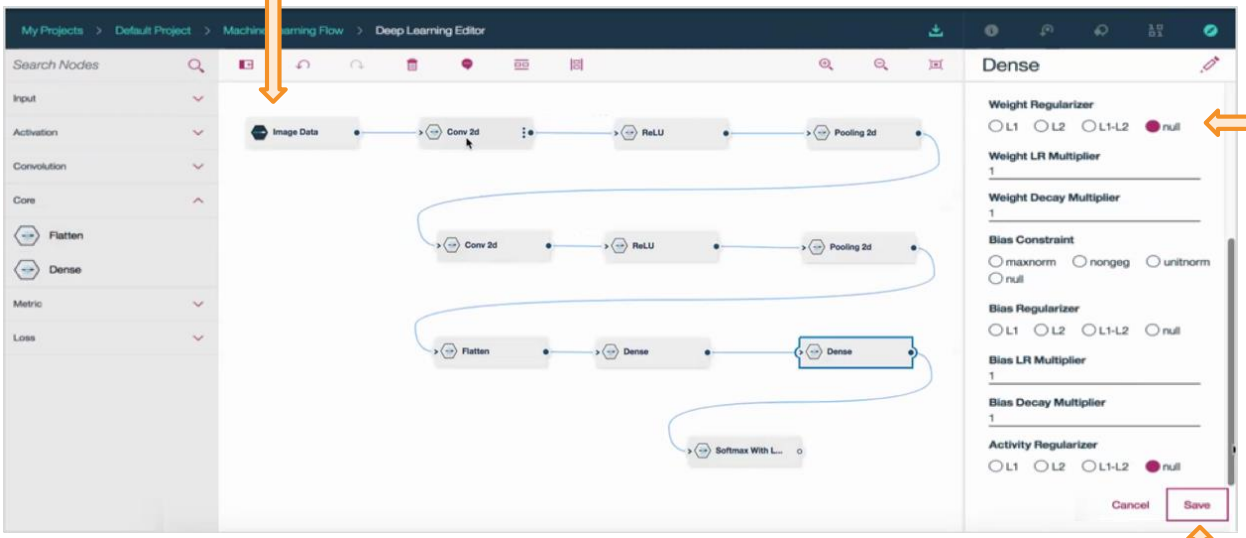
Neural Network Modeler

An intuitive drag-and-drop, no-code interface for designing neural network structures using the most popular deep learning frameworks. Quickly capture your network design then single click export for experimental optimization.

Supported Frameworks

TensorFlow™ PYTORCH Caffe Keras

Real-time validation of network flow



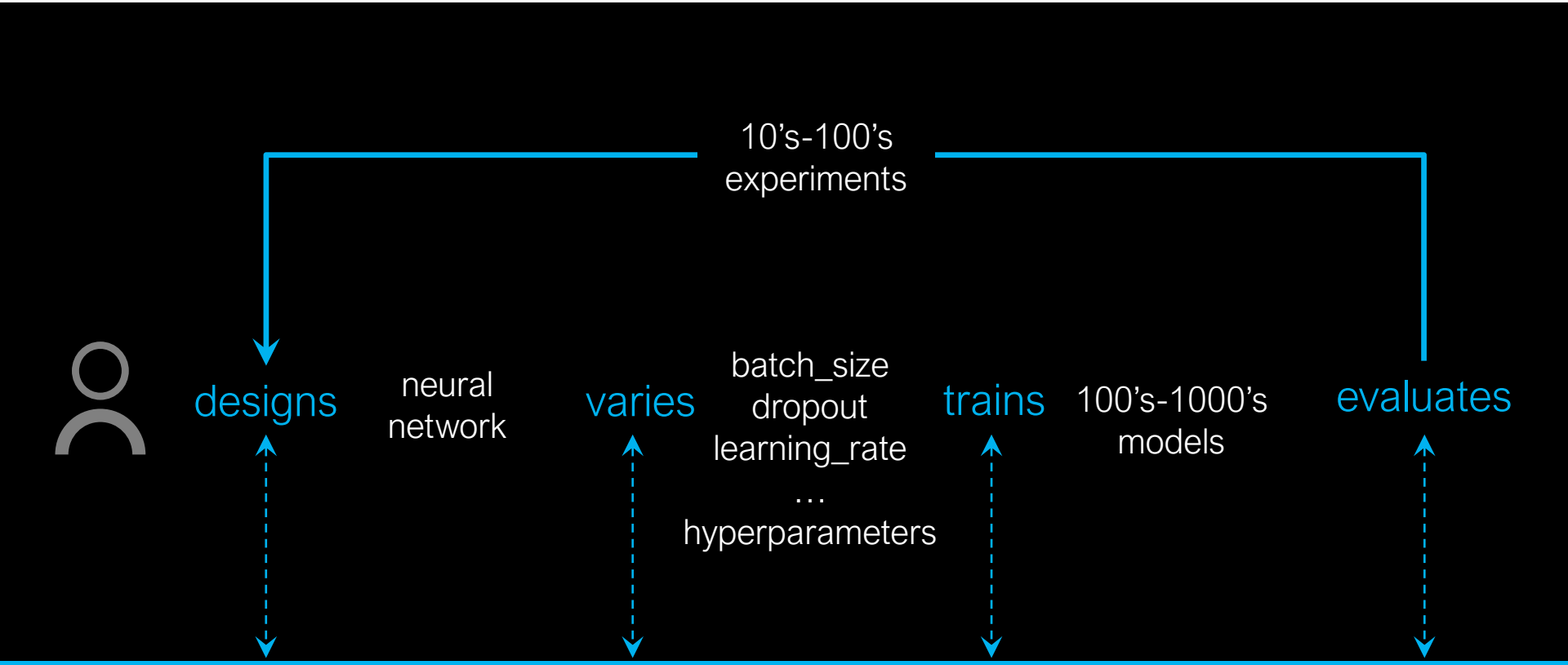
Drag-and-drop network layers

- Define layer configuration
- Choose optimizer params

- Generate CPU or GPU compatible code

- Save as popular framework code
- Export as a python notebook
- Execute as batch experiment

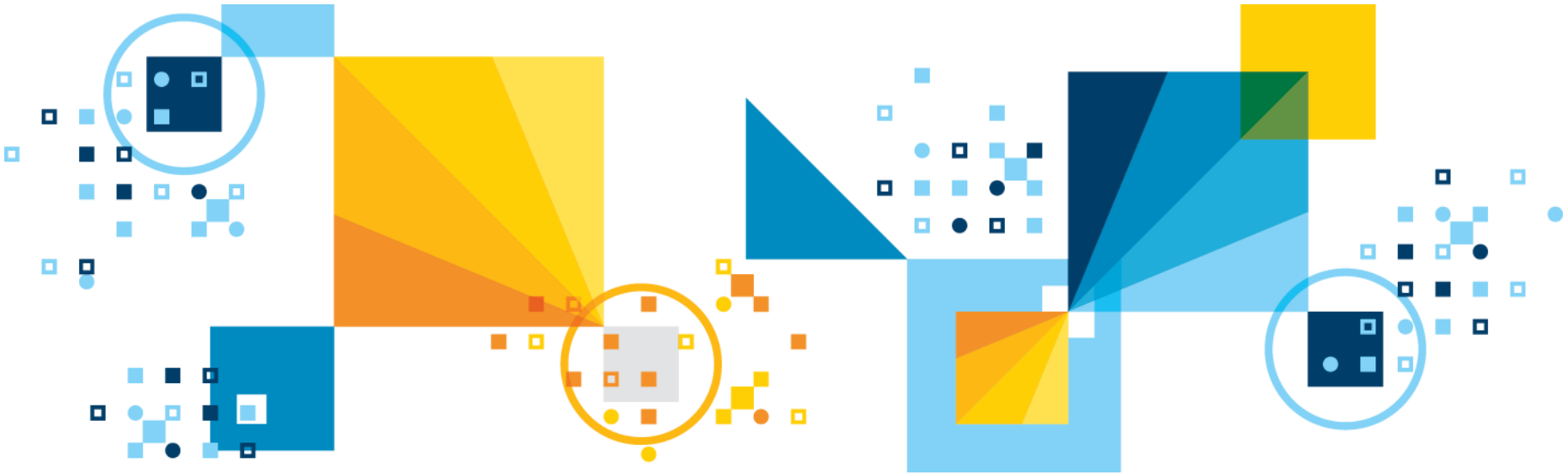
Experiment Builder



Experiment Builder
supports the end-to-end workflow

Introduction to Adversarial Robustness Toolbox

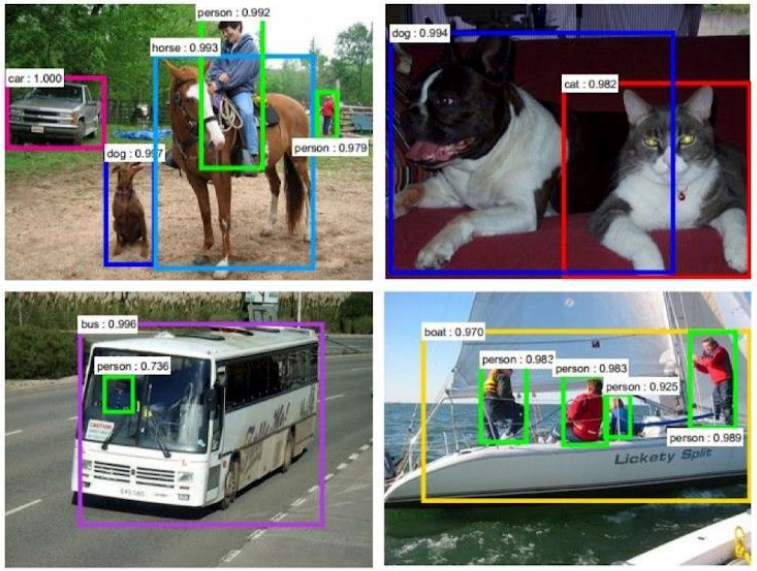
Zoya Yeprem



DNN in Visual Recognition

- **We use DNNs for lots of things:**
  - Facial recognition → iPhone FaceID
  - Text recognition → Mobile Check Deposit
  - Self driving cars → help detect signs, pedestrians, traffic lights, etc.

Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation

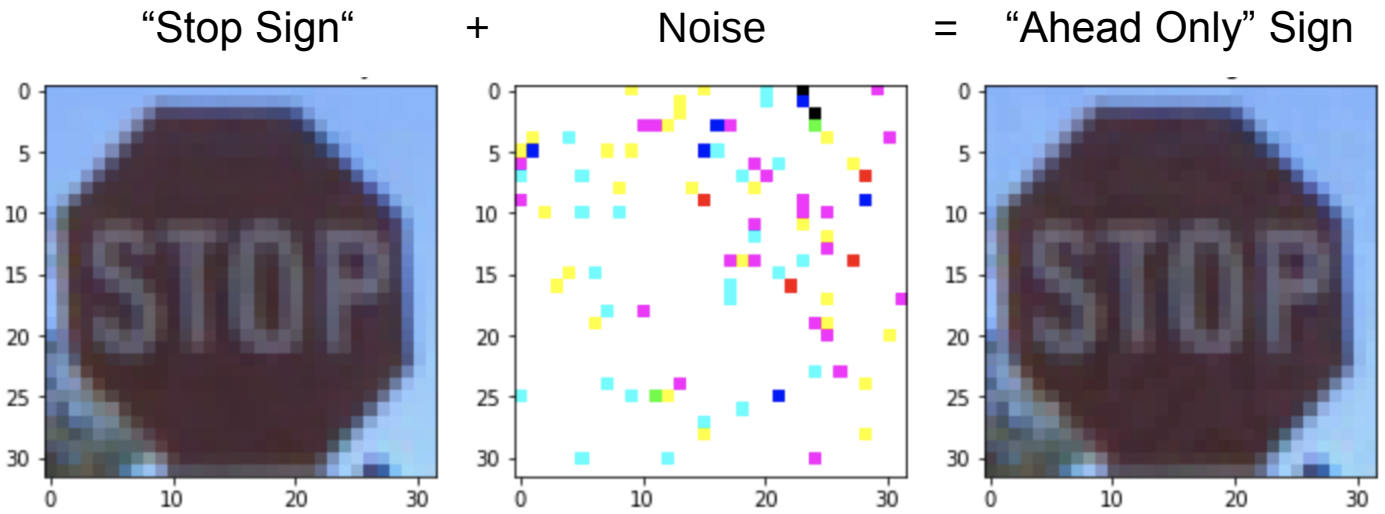
Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

What are Adversarial Images?

- Adversarial examples are inputs (say, images) which have deliberately been modified to produce a desired response by a DNN.

- Often, the target of adversarial examples is **misclassification** or a **specific incorrect prediction** which would benefit an attacker.



Adversarial Attacks

- Carefully crafted noise added to the input
- This noise gets amplified when ran through the layers of DNN
- The goal is to generate adversaries visually similar to original image

Stronger
BUT
More
Expensive

- FGSM
- Random + FGSM
- Projected Gradient Descent *
- DeepFool
- JSMA
- C&W

Threat Model

Black Box

- Attackers can only observe the outputs of a model. E.g. Attacking a model via an API
  - The adversary has knowledge of the training algorithm and hyperparameters
  - The adversary has no knowledge of the training algorithm or hyperparameters.

➢ Examples:
  - ➢ Boundary Attack
  - ➢ Substitute Blackbox Attack
  - ➢ Etc.

White Box

- attackers have complete access to the model that they want to attack.
- These are most effective attacks

➢ Examples:
  - ➢ Fast Gradient Sign Method (FGSM)
  - ➢ Random + FGSM
  - ➢ Projected Gradient Descent
  - ➢ Etc.

Why are they dangerous?

- **Can be crafted even if the attacker doesn't have exact knowledge of the architecture of the DNN**
- **Adversarial attacks can be launched in the physical world**
  - adversaries could evade face recognition systems by wearing specially designed glasses
  - defeat visual recognition systems in autonomous vehicles by sticking patches to traffic signs

Subtle Poster

Camouflage Sticker

* Pictures from paper: Kevin Eykholt, et al. "Robust Physical-World Attacks on Deep Learning Visual Classification"

Are they effective?

- **Researchers proved that these attacks are successful! [1]**

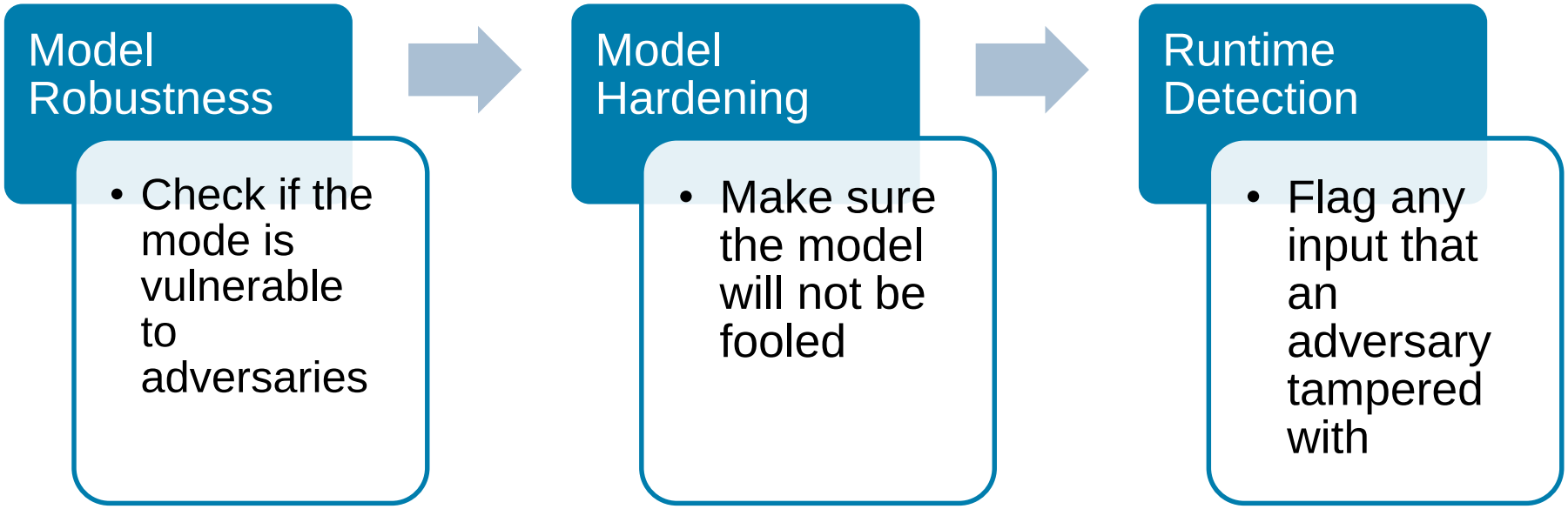
Perturbation	Attack Success	A Subset of Sampled Frames $k = 10$
Subtle poster	100%	
Camouflage abstract art	84.8%	

[1] Kevin Eykholt, et al. “Robust Physical-World Attacks on Deep Learning Visual Classification”

Adversarial Robustness Toolbox (ART)

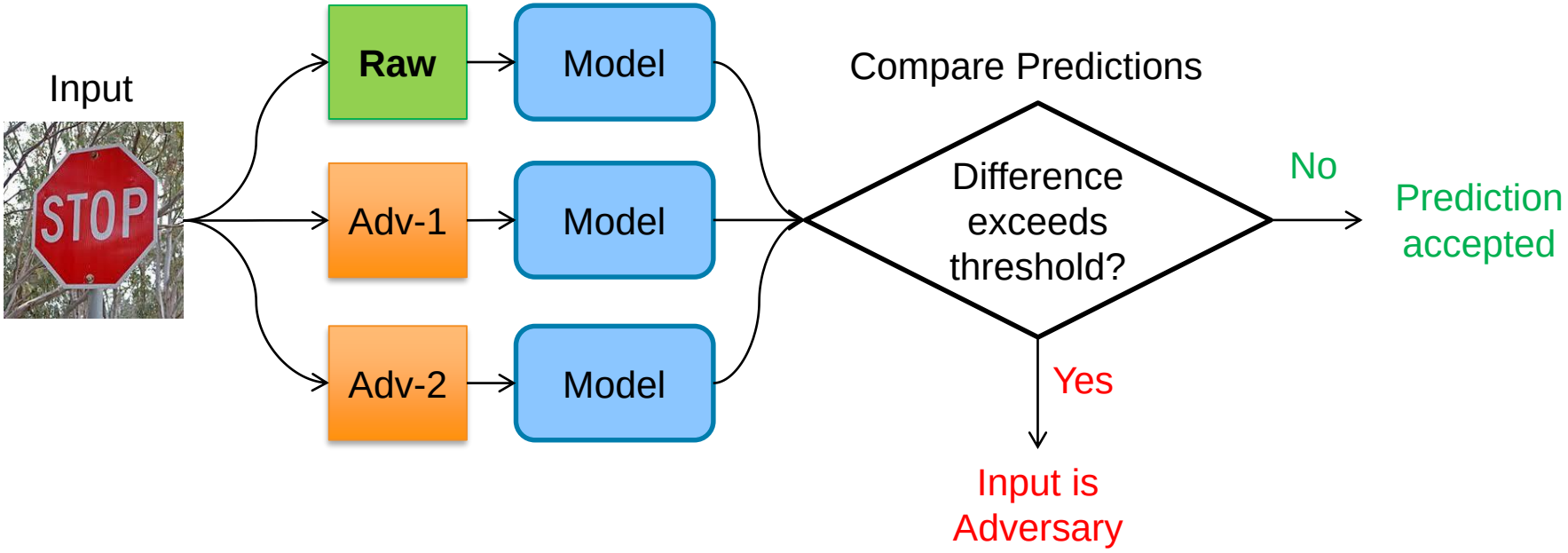
- IBM Research team in Ireland developed the toolkit to help defend DNNs against adversarial attacks
- Open-source software library
- Written in python
- Supports most deep learning frameworks : TensorFlow, Keras, PyTorch, etc.
- It creates adversarial examples AND provides methods for defending DNNs against those.

How can ART help?



Model Robustness

- Generate an adversarial image and pass it to DNN
  - Record the loss of accuracy on altered inputs
  - Measure how much the internal representations and the output of a DNN vary when small changes are applied to its inputs



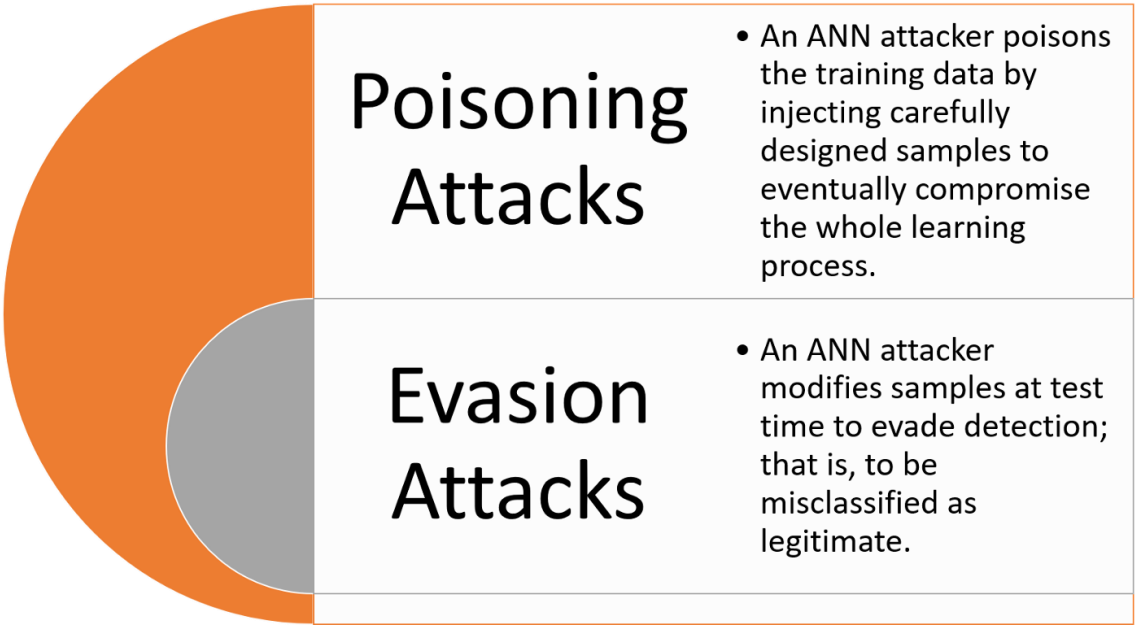
Model Hardening

- **Preprocess training data to generate adversaries**
  - Augment them to training data
  - Flag adversaries so the model can learn what is legit and what is not
  - Train the model using all samples

- **Use augmented train set to change DNN architecture to prevent adversarial signals from propagating through the internal representation layers.**

Runtime Detection

- Examining the neural activations produced by a training dataset
- Discriminate legit examples ones manipulated by an adversarial attack.
- The current version of ART focuses on two types of adversarial attacks: Evasion and Poisoning:



Conclusions

- **Adversarial attacks are real threats**
  - Self-driving cars
  - Healthcare
  - Financial institutions
  - Insurance companies
  - ...
- **It's important to**
  - Realize there are vulnerabilities
  - Have means to protect ourselves

Lab 4: ART in Action

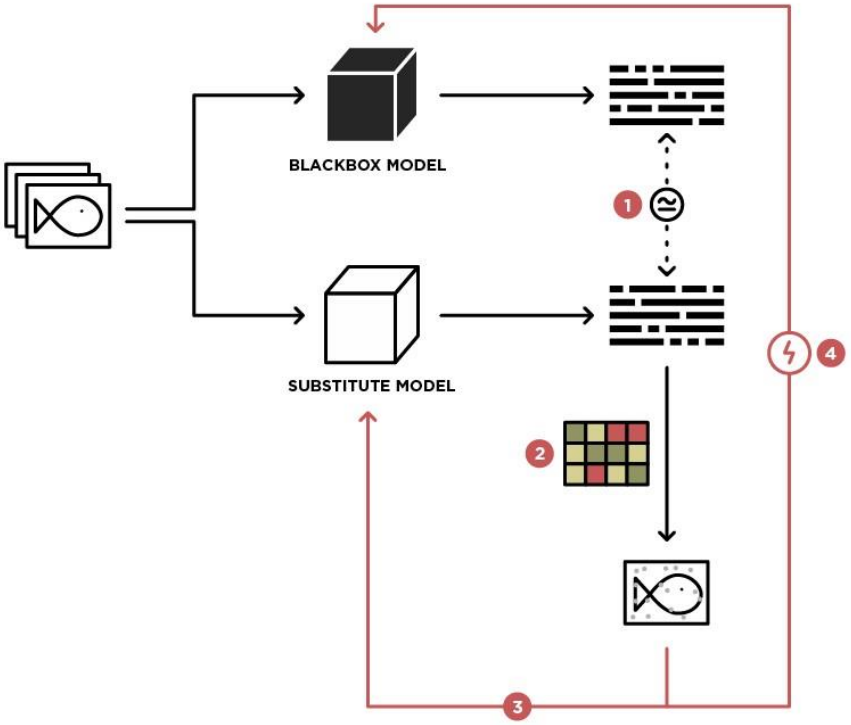
- **Create a Notebook in Watson Studio**
- **Upload the Lab-4 Notebook file using provided URL**
- **Run through cells**

- **Overview**
  - Load a Tensorflow trained model
  - Create an ART classifier object using the loaded model
  - Perform an adversarial attack
  - Perform a defense to make sure manipulated images can still be classified correctly

Bonus Slides

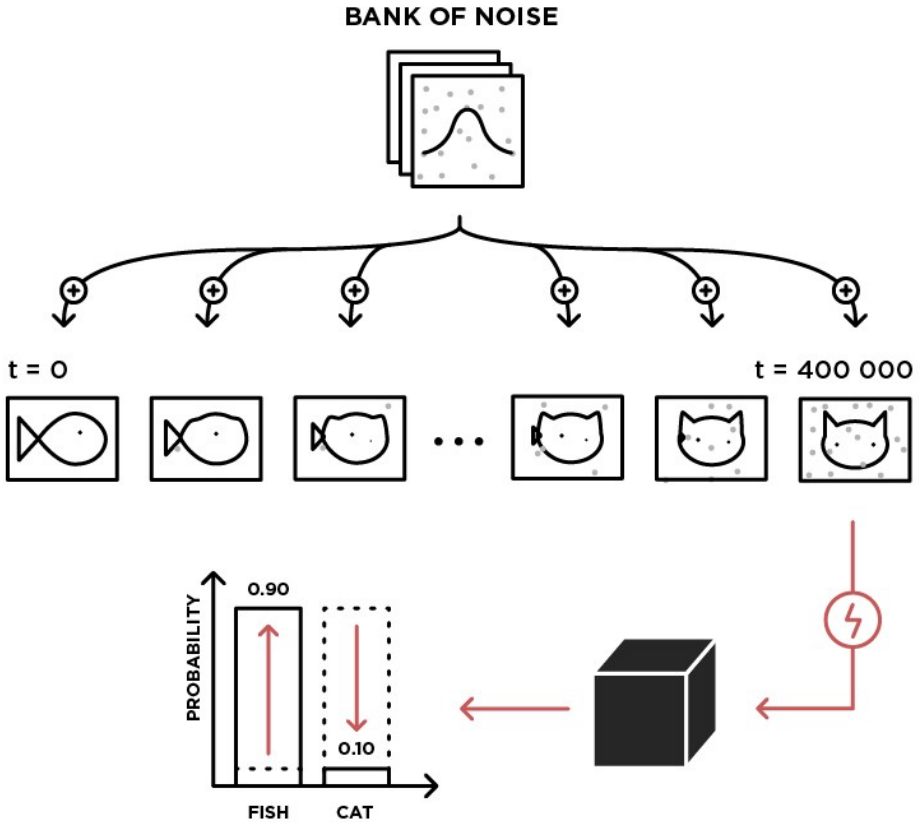
Substitute Blackbox Attack

- **Approximate the decision boundary of the Blackbox model that we want to attack**.
  - Train a **substitute model** on a synthetic dataset that is similar to the dataset that the Blackbox model is trained on.
  - The trick here is that the **label for the synthetic dataset should come from the Blackbox model's prediction**.



Boundary Attack

1. Evaluates a sequence of perturbed images through the model
   - In **non-targeted attack**, the starting image can be sampled from uniform noise.
   - In **targeted attack**, the starting image is an example from the target misclassification class.
2. The method **modifies the image iteratively** to look more like an example from another class while continuing to preserve its adversarial nature



Labs: 5,6,7,8 Titanic Data

- **Variable Descriptions:**

survival	Survival
	(0 = No; 1 = Yes)
pclass	Passenger Class
	(1 = 1st; 2 = 2nd; 3 = 3rd)
name	Name
sex	Sex
age	Age
sibsp	Number of Siblings/Spouses Aboard
parch	Number of Parents/Children Aboard
ticket	Ticket Number
fare	Passenger Fare
cabin	Cabin
embarked	Port of Embarkation
	(C = Cherbourg; Q = Queenstown; S = Southampton)

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C

Lab-5: AutoAI + DevOps

Introduction:

In this lab, you will use IBM's Watson Machine Learning GUI to train, evaluate, and deploy a Watson Machine Learning model based on the Titanic dataset. You will then deploy a web application that calls the Watson Machine Learning model.

Objectives:

Upon completing the lab, you will:

- Become familiar with the AutoAI feature of Watson Studio.
- Train/Evaluate a machine learning model
- Deploy a machine learning model.
- Deploy a Python Flask web application that we will configure to "call" the deployed machine learning model.
- Configure the application to connect to the machine learning service.
- Update the code in the application to specify the endpoint of the deployed model, and use DevOps to build and re-deploy the application.
- Run the application to demonstrate the use of the deployed machine learning model to score the survivability of a Titanic passenger.

Lab-6: Introduction to the Data Refinery

Introduction:

In this lab, you will use the Watson Studio Data Refinery to profile data, visualize data, and prepare data for modeling.

Objectives:

Upon completing the lab, you will know how to:

- Profile the data
- Visualize the data to gain a better understanding
- Prepare the data for modeling
- Run the sequence of data preparation operations on the entire data set.

Lab-7: SPSS Modeler

Introduction:

In this lab, you will use the Watson Studio SPSS Modeler capability to explore, prepare, and model trafficking data. The SPSS Modeler is a drag and drop capability to build machine learning pipelines.

Objectives:

Upon completing this lab, you will have:

- Become familiar with the Watson Studio SPSS Modeler capability
- Profiled the data set
- Explored the data set with visualizations
- Transformed the data
- Trained/Evaluated a machine learning mode.

Lab 8: Cognos Dashboard Embedded Lab

Introduction:

In this lab you will use the Cognos Dashboard Embedded Service to create a dashboard in Watson Studio.

Objectives:

Upon completing this lab, you will have:

- Created a Cognos Dashboard Embedded Service
- Associated the service with a project
- Created a dashboard based on the Titanic dataset.
- Shared the dashboard