

Introduction to Machine Learning on IBM Watson Studio

Upkar Lidder

Developer Advocate, IBM

IBM Developer

- > ulidder@us.ibm.com
- > @lidderupk
- > upkar.dev

<http://bit.ly/waston-ml-sign>

Prerequisites

1. Create IBM Cloud Account using THIS URL

<http://bit.ly/waston-ml-sign>

2. Check your email and activate your account. Once activated, log back into your IBM Cloud account using the link above.

3. If you already have an account, use the above URL to sign into your IBM Cloud account.

Call for Code 2019

COMPETITION DETAILS

Commit to the Cause Push for Change

Over 100,000 developers from 156 nations competed in the Call for Code 2018 Global Challenge.

They built over 2,500 applications with IBM Cloud technology to reduce the impact of natural disasters.

This year, it's your turn.

[Join the Call for Code Community >](#)

[See complete Call for Code Challenge details >](#)

[Learn about IBM Code and Response >](#)



Call for Code Prizes in 2019

The winner of the second annual Call for Code Challenge that [builds the best solution for natural disaster preparedness and response](#) receives:

- The Call for Code Global Award presented at the Call for Code Award Event.
- A **\$200,000 USD cash prize**.
- Open source project support from [The Linux Foundation](#).
- Meetings with mentors and potential investors.
- Solution implementation support through [Code and Response](#).

Call for Code 2019 - Get started quickly

<https://developer.ibm.com/callforcode/>

Explore technologies that can be used to build world-changing solutions, then get your hands on the code

Data Science
Understand, analyze, and predict health and nutrition needs to improve services with data science.

Artificial Intelligence
Use AI and bots to improve real-time communications with natural language processing.

Blockchain
Build secure, resilient, traceable, and transparent supply networks with blockchain.

Traffic & Weather
Improve logistics based on traffic and weather activity to reduce the number of people affected.

- Background
- Videos
- Research
- Solutions
- Data
- Blogs/Articles
- Code patterns

Explore types of disasters and how we can lessen their effects through technology

Earthquakes and Tsunamis

Hurricanes and Typhoons

Wildfires

Volcanoes

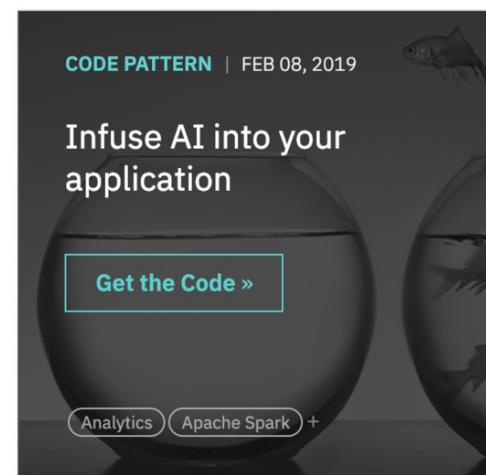
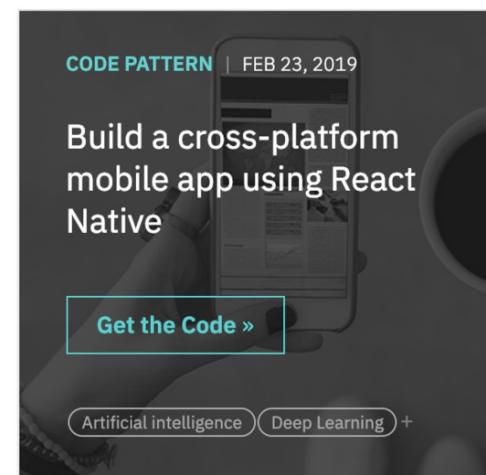
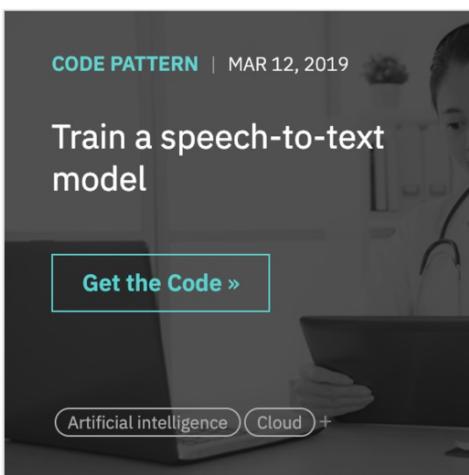
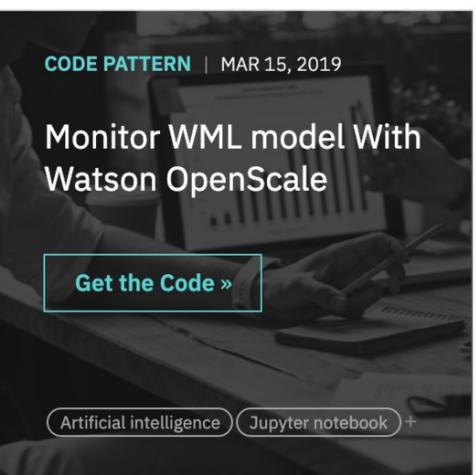
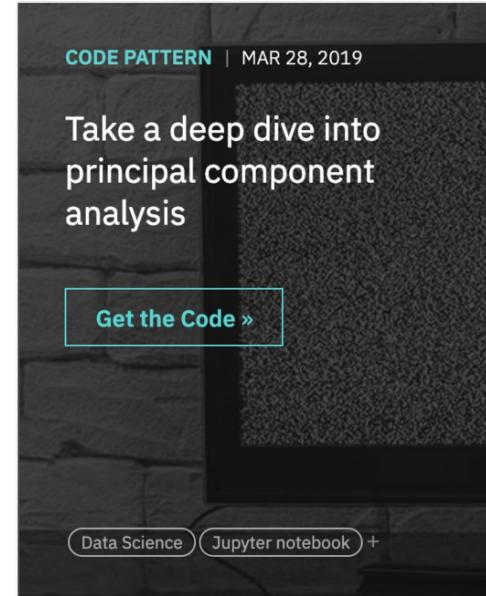
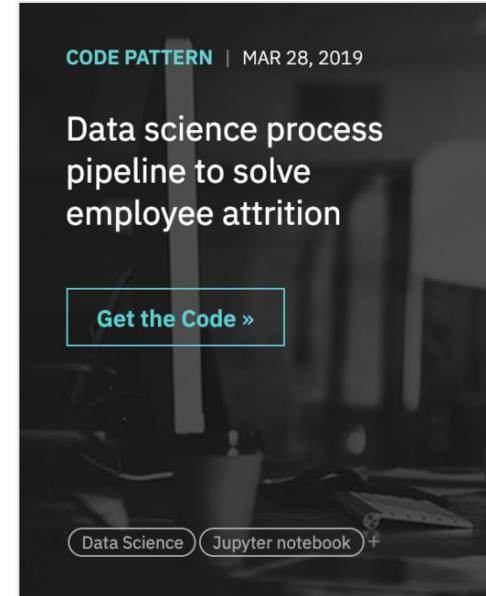
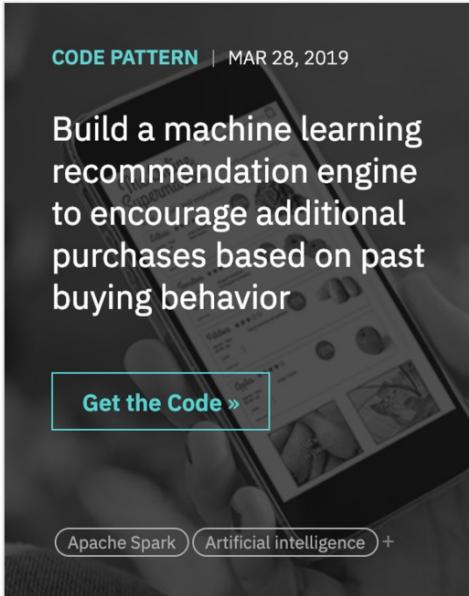
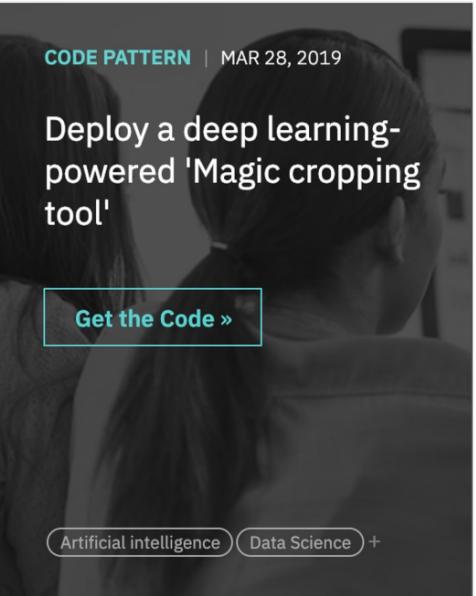
Floods and Storm Surges

Extreme Weather

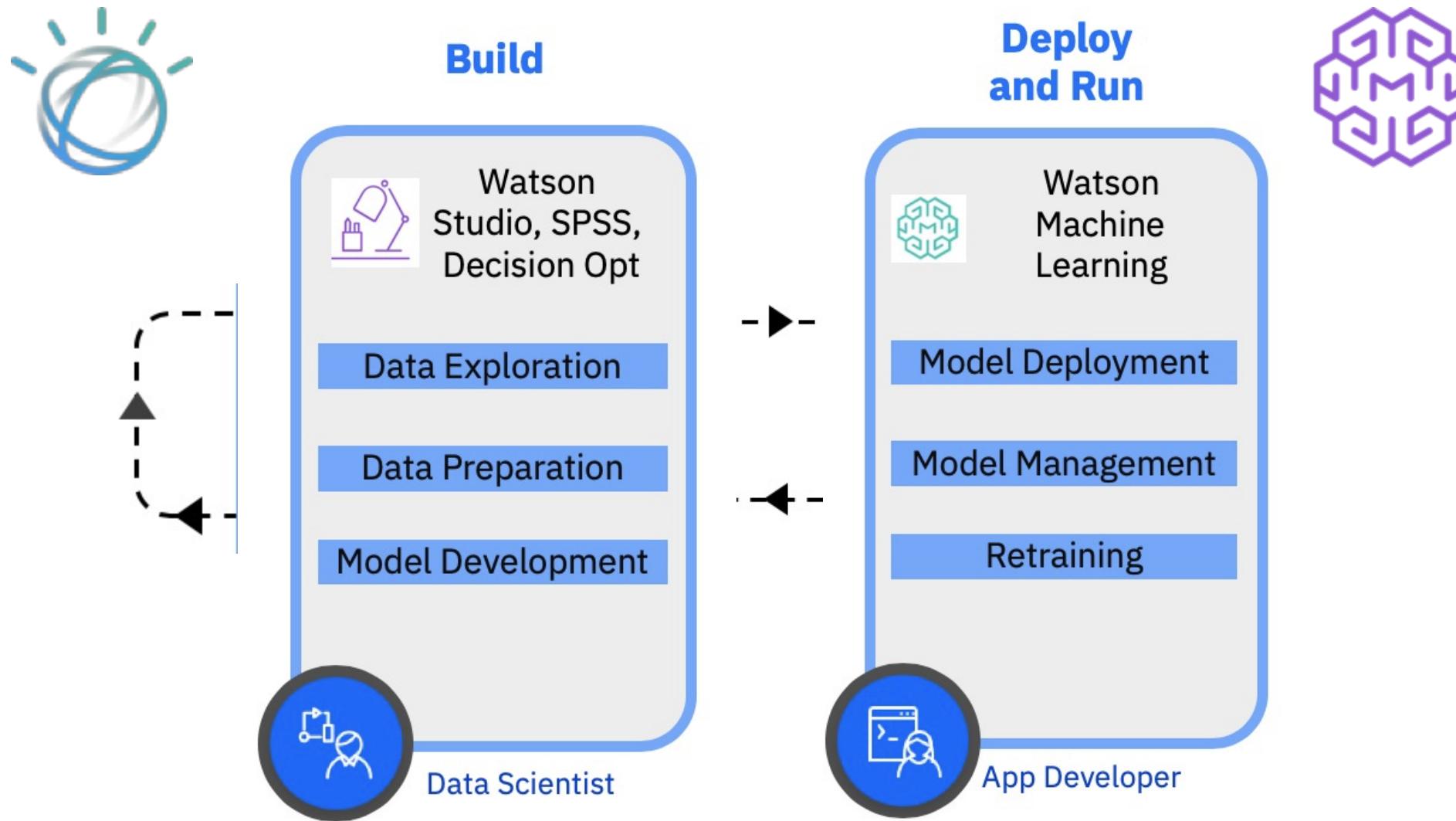
IBM Developer [@lidderupk](#)

IBM Code Patterns

Machine Learning 



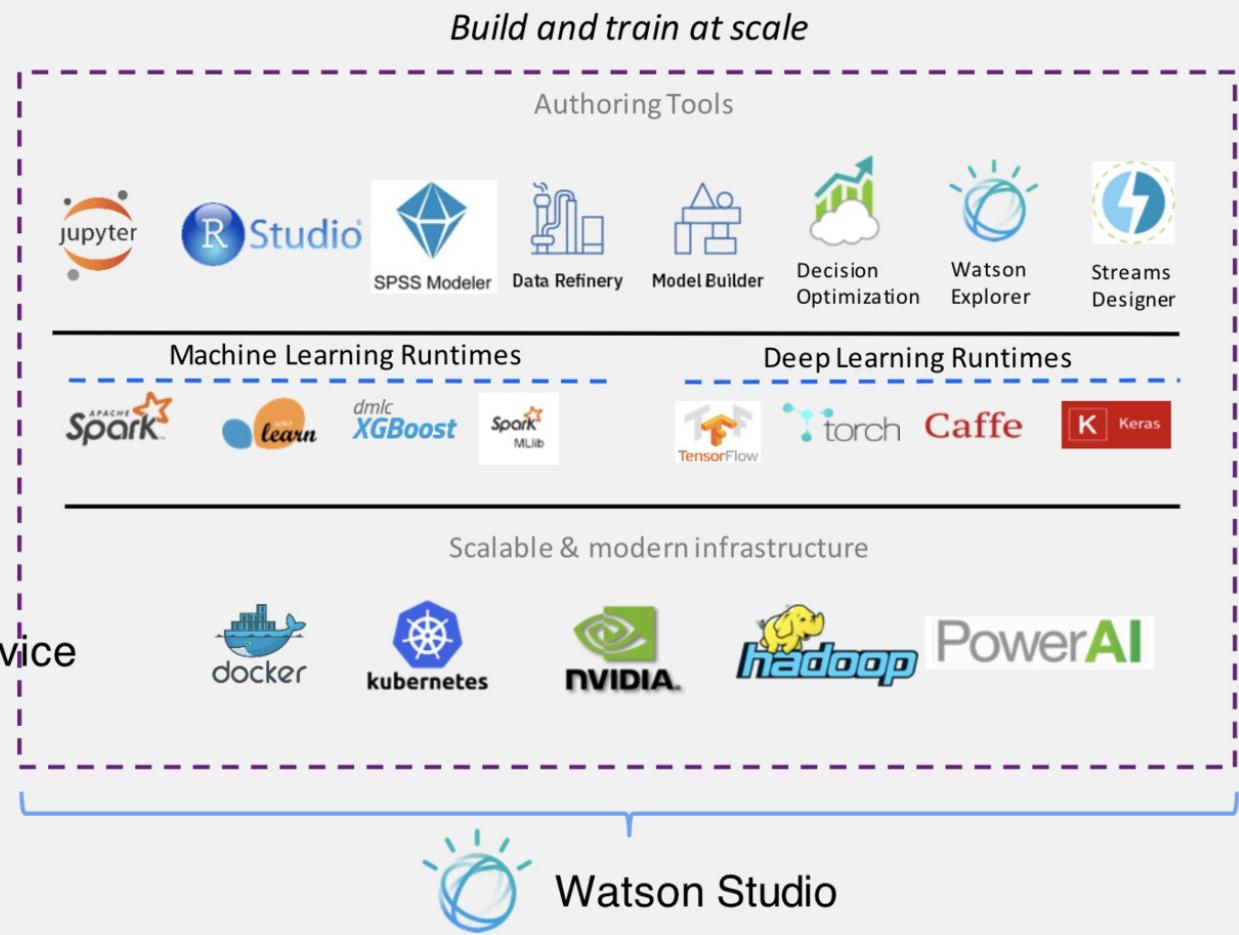
Watson Studio & Watson Machine Learning



Watson Studio

IBM Watson Studio

- Create, collaborate, govern and integrate
- Using best of breed - Open source & IBM tools
- Code (R, Python or Scala) and no-code/visual modeling tools
- Most popular open source frameworks
- IBM best-in-class frameworks
- Workflow driven data science
- Container-based resource management
- Elastic pay as you go cpu/gpu power
- Run on x86, Power, zLinux
- Integrate with Cloudera and HDP using Hadoop Integration service
- Train and deploy where your data lives
 - As a Fully managed Service
 - In your Data center
 - On AWS, Azure, IBM Soft layer



IBM Watson Studio

The screenshot shows the IBM Watson Studio interface with various numbered callouts highlighting specific features:

- 1: My Projects / My Data Science Project
- 2: Add to project
- 3: RStudio
- 4: Choose asset type dialog
- 5: Drop files here or browse for files to upload
- 6: Catalog
- 7: Launch IDE
- 8: Overview
- 9: Assets
- 10: Environments
- 11: Bookmarks
- 12: Deployments
- 13: Access Control
- 14: Settings
- 15: Search bar: What assets are you looking for?
- 16: Data assets section
- 17: Available asset types:
 - Data
 - Connection
 - Connected data
 - Notebook
 - Dashboard
 - Watson Machine Lea...
 - Experiment
 - Image classification ...
 - Object detection mo...
 - Natural Language Cl...
 - Modeler flow
 - Data Refinery flow
 - Streams flow
 - Synthesized neural n...
- 18: Close button

Workshop - Goals

Successfully **Create**, **Store** and
Deploy a Linear Regression Model
on IBM Cloud using Watson Studio
and Watson Machine Learning
Services.

Question - predict median house price for Boston area

Boston House Prices dataset

=====

Notes

Data Set Characteristics:

:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical predictive

:Median Value (attribute 14) is usually the target

:Attribute Information (in order):

- CRIM per capita crime rate by town
- ZN proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS proportion of non-retail business acres per town
- CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX nitric oxides concentration (parts per 10 million)
- RM average number of rooms per dwelling
- AGE proportion of owner-occupied units built prior to 1940
- DIS weighted distances to five Boston employment centres
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per \$10,000
- PTRATIO pupil-teacher ratio by town
- B $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
- LSTAT % lower status of the population
- MEDV Median value of owner-occupied homes in \$1000's

:Missing Attribute Values: None

:Creator: Harrison, D. and Rubinfeld, D.L.

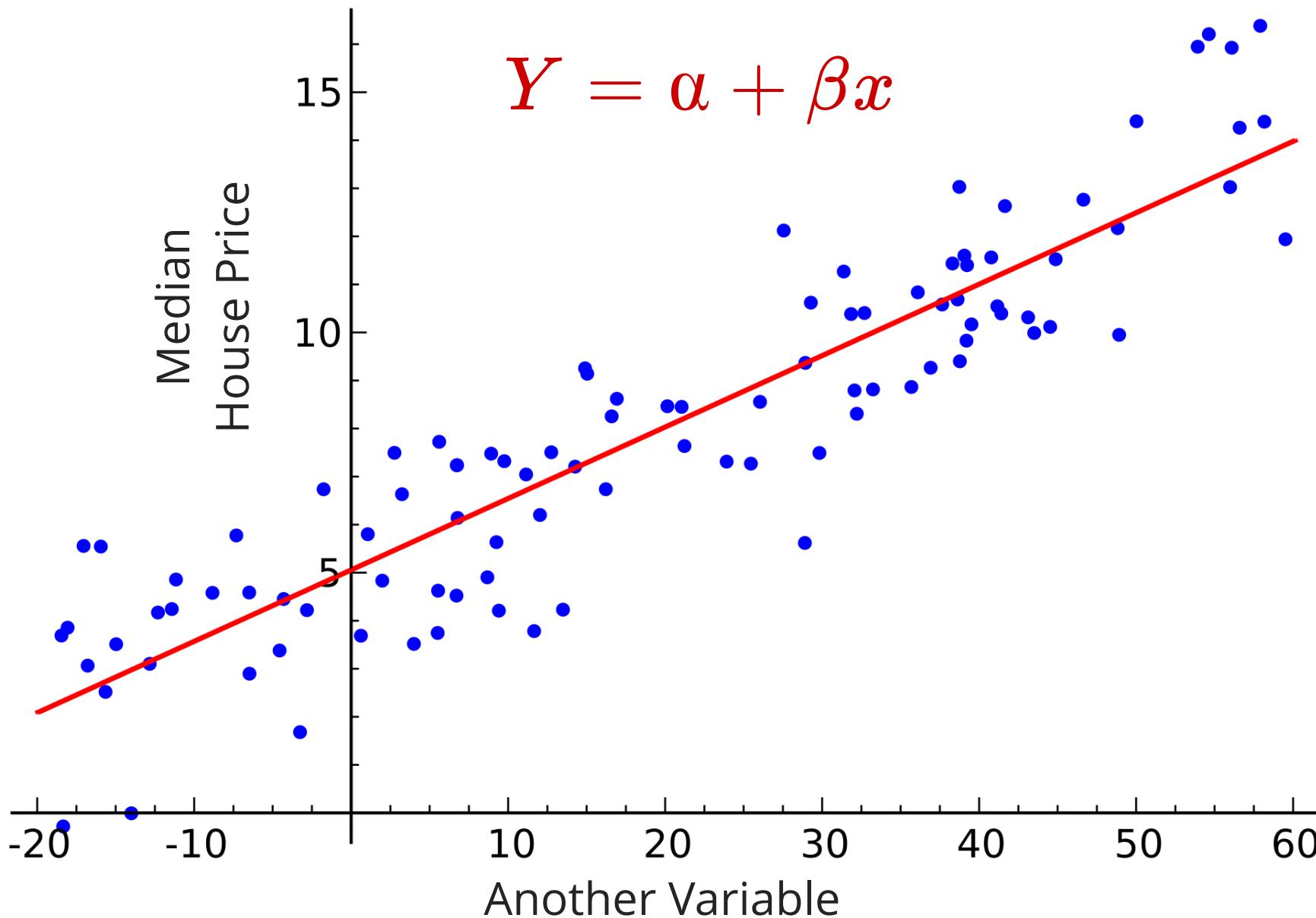
This is a copy of UCI ML housing dataset.

<http://archive.ics.uci.edu/ml/datasets/Housing>

	0	1	2	3	4
CRIM	0.00632	0.02731	0.02729	0.03237	0.06905
ZN	18.00000	0.00000	0.00000	0.00000	0.00000
INDUS	2.31000	7.07000	7.07000	2.18000	2.18000
CHAS	0.00000	0.00000	0.00000	0.00000	0.00000
NOX	0.53800	0.46900	0.46900	0.45800	0.45800
RM	6.57500	6.42100	7.18500	6.99800	7.14700
AGE	65.20000	78.90000	61.10000	45.80000	54.20000
DIS	4.09000	4.96710	4.96710	6.06220	6.06220
RAD	1.00000	2.00000	2.00000	3.00000	3.00000
TAX	296.00000	242.00000	242.00000	222.00000	222.00000
PTRATIO	15.30000	17.80000	17.80000	18.70000	18.70000
B	396.90000	396.90000	392.83000	394.63000	396.90000
LSTAT	4.98000	9.14000	4.03000	2.94000	5.33000
MEDV	24.00000	21.60000	34.70000	33.40000	36.20000

This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

Linear regression - Simple



Steps

1. Sign up / Log into *IBM Cloud* - <http://bit.ly/waston-ml-sign>
2. Create *Watson Studio Service*.
3. Sign into Watson Studio and create a new *Data Science Project*. It also creates a *Cloud Object Store* for you.
4. Associate a *Machine Learning Service* with your project.
5. Upload csv *data* to your project.
6. Add a new *Machine Learning Model* to your project.
7. Create a *Linear Regression Model* and *save* it to IBM Cloud.
8. Create a new *deployment* on IBM Cloud.
9. *Test* your model !

Step 1 - sign up/ log into IBM Cloud

Already have an IBM Cloud account? [Log in](#)

Create an IBM Cloud Account

Email* →

First Name*

Last Name*

Country or Region*
United States

Password*

IBM may use my contact data to keep me informed of products, services and offerings:
 by email. by telephone.

You can withdraw your marketing consent at any time by sending an email to netsupp@us.ibm.com. Also you may unsubscribe from receiving marketing emails by clicking the unsubscribe link in each such email.

More information on our processing can be found in the [IBM Privacy Statement](#). By submitting this form, I acknowledge that I have read and understand the IBM Privacy Statement.

[Create Account](#)

<http://bit.ly/waston-ml-sign>

Step 2 - locate Watson Studio in Catalog

IBM Cloud Catalog Docs Support Manage Upkar6 Lidder6's Account

Catalog

Search the catalog... Filter

All Categories

- Compute
- Containers
- Networking
- Storage
- AI**
- Analytics
- Databases
- Developer Tools
- Integration
- Internet of Things
- Security and Identity
- Starter Kits
- Web and Mobile
- Web and Application

AI

Watson Assistant
Lite • IBM • IAM-enabled

Watson Assistant lets you build conversational interfaces into any application, device, or channel.

Watson Studio
Lite • IBM • IAM-enabled

Embed AI and machine learning into your business. Create custom models using your own data.

Compare and Comply
Lite • IBM • IAM-enabled

Process governing documents to convert, identify, classify, and compare important elements

Discovery
Lite • IBM • IAM-enabled

Add a cognitive search and content analytics engine to applications.

Knowledge Catalog
Lite • IBM • IAM-enabled

Discover, catalog, and securely share enterprise data.

Knowledge Studio
Lite • IBM • IAM-enabled

Teach Watson the language of your domain.

FEEDBACK



Step 3 - create Watson Studio instance

The screenshot shows the IBM Cloud interface for creating a new Watson Studio instance. The top navigation bar includes 'IBM Cloud', a search bar, 'Catalog', 'Docs', 'Support', 'Manage', and a user account section. The main content area displays the 'Watson Studio' service details. On the left, there's a brief description of Watson Studio, links to 'View Docs' and 'Terms', and metadata like 'AUTHOR: IBM', 'PUBLISHED: 05/08/2019', and 'TYPE: Service'. The right side contains configuration fields: 'Service name' (set to 'Watson Studio-xyz'), 'Choose a region/location to deploy in' (set to 'Dallas'), 'Select a resource group' (set to 'Default'), and 'Tags' (with an example 'env:dev, version-1'). Below these are 'Features' sections with bullet points: 'Use what you know, learn what you don't' (describing the ability to start from tutorials or scratch) and 'Power on demand' (describing enterprise-scale features). At the bottom, there are 'Need Help?' links, 'Add to estimate' and 'Create' buttons, and a 'Feedback' button.

Watson Studio democratizes machine learning and deep learning to accelerate infusion of AI in your business to drive innovation. Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

[View Docs](#) [Terms](#)

AUTHOR IBM
PUBLISHED 05/08/2019
TYPE Service

Service name:
Watson Studio-xyz

Choose a region/location to deploy in:
Dallas

Select a resource group: ⓘ
Default

Tags: ⓘ
Examples: env:dev, version-1

Features

- Use what you know, learn what you don't
Start from a tutorial, start from a sample, or start from scratch. Tap into the power of the best of open source (RStudio, Jupyter Notebooks) and Watson services for flexible model creation. Use Python, R, or Scala. Stop downloading and configuring analysis environments and start
- Power on demand
Enterprise-scale features on demand. From data exploration and preparation, to enterprise-scale performance. Manage your data, your analytical assets, and your projects in a secured cloud environment.

Need Help?
[Contact IBM Cloud Support](#)

Add to estimate **Create**

FEEDBACK

Step 4 - launch Watson Studio

The screenshot shows the IBM Cloud interface for launching Watson Studio. The top navigation bar includes 'IBM Cloud' (with a three-line menu icon), a search bar, 'Catalog', 'Docs', 'Support', 'Manage' (with a dropdown arrow), 'Upkar6 Lidder6's Account', and user icons. A sidebar on the left has 'Manage' selected (highlighted in blue) and 'Plan' as an option. The main content area displays a resource named 'Watson Studio-xyz' under 'Resource list /'. It shows 'Resource group: Default' and 'Location: Dallas' with a link to 'Add Tags'. Below this is a circular icon containing a desk lamp and a notepad. The text 'Watson Studio' is prominently displayed, followed by 'Welcome to Watson Studio. Let's get started!' and a large 'Get Started' button. At the bottom, there are sections for 'Documentation' (describing getting started and how-to's) and 'Community' (describing tutorials, articles, notebooks, and datasets). A small navigation arrow is at the bottom left.

IBM Cloud

Resource list /

Watson Studio-xyz

Resource group: Default Location: Dallas Add Tags

Watson Studio

Welcome to Watson Studio. Let's get started!

Get Started

Documentation

From getting started to how to's – see what's available.

Community

Check out our tutorials, articles, along with sample notebooks and data sets you can use to get going.

Step 5 - create a new project

The screenshot shows the IBM Watson Studio interface. At the top, there's a dark header bar with the text "IBM Watson Studio" on the left, a bell icon, the user account "Upkar6 Lidder6's Account" in the middle, and a profile icon on the right. Below the header is a light-colored navigation bar with a "Get started" button on the right. The main content area has a blue background with white text. It says "Welcome Upkar6!" and "Watson Studio • Watson Knowledge Catalog". A large call-to-action button labeled "Start by creating a project" is prominent. Below it, a text box explains that a project is for organizing resources and collaborating with team members. Two buttons are shown: "Create a project" (highlighted with a pink border) and "Search a catalog". To the right of the main content is a circular graphic featuring a computer monitor with a magnifying glass over binary code, a smartphone, and a small plant, all set against a blue background with white geometric shapes.

IBM Watson Studio

Upkar6 Lidder6's Account

Get started

Welcome Upkar6!

Watson Studio • Watson Knowledge Catalog

Start by creating a project

A project is how you organize your resources to work with data and collaborate with team members

Create a project

Create a project, then add the tools and assets you need.

Search a catalog

Find the assets you need in a catalog.

@lidderupk

Step 6 - pick Data Science starter

IBM Watson Studio Upkar6 Liddler6's Account UL

← Back

Create a project

Choose the project starter for your work. Required services with Lite plans are provisioned automatically. You can add other assets and services later.

Standard
Work with any type of asset. Add services for analytical assets as you need them.

Import project
Import a project from a file.

Data Science
Analyze data to discover insights and share your findings with others.

ASSETS
Data • Notebooks

Visual Recognition
Tag and classify visual content using the Watson Visual Recognition service.

ASSETS
Data • Visual recognition model

Deep Learning
Build neural networks and deploy deep learning models.

ASSETS
Data • Modeler flow • Model • Experiment

Modeler
Build modeler flows to train SPSS models or design deep neural networks.

ASSETS
Data • Modeler Flow • Model • Experiment

Business Analytics
Create visual dashboards from your data to gain insights faster.

ASSETS
Data • Dashboard

Data Engineering
Combine, cleanse, analyze, and shape data using Data Refinery.

ASSETS
Data • Data Refinery flow

Streams Flow
Ingest and analyze streaming data using the Streaming Analytics service.

ASSETS
Data • Streams flow

Step 7 - give the project a name and assign COS

New project

Define project details

Name

My Data Science Project

Description

Project to explore Boston housing data and create some machine learning models

Choose project options

Restrict who can be a collaborator i

Project will include integration with [Cloud Object Storage](#) for storing project assets.

Additional tools and services can be added in Project Settings after project creation.

Storage

cloud-object-storage-dsx

[Cancel](#) [Create](#)

Step 8 - open asset tab

The screenshot shows the IBM Watson Studio interface. At the top, there's a dark header bar with the title "IBM Watson Studio". Below it, a navigation bar includes "My Projects / My Data Science Project", "Launch IDE", "Add to project", and several icons. The main menu has tabs for "Overview", "Assets" (which is highlighted with a pink box), "Environments", "Bookmarks", "Deployments", "Access Control", and "Settings". To the right of the main content area, there's a sidebar with tabs for "Load" (highlighted with a pink box), "Files", and "Catalog". A large central panel displays a search bar with the placeholder "What assets are you looking for?", a section titled "Data assets" with a "Data assets" icon, and a message stating "You don't have any data assets yet". To the right of this panel is a "Drop files here or [browse](#) for files to upload" area.

IBM Watson Studio

My Projects / My Data Science Project

Launch IDE Add to project

Overview Assets Environments Bookmarks Deployments Access Control Settings

Load Files Catalog

What assets are you looking for?

Data assets

You don't have any data assets yet

Use the [Load page](#) to upload files. To add data from a connection, click [Add to project](#) and then choose [Connected Data](#).

Drop files here or [browse](#) for files to upload.

Step 9 - drag and drop data file into Load Assets

The screenshot shows the IBM Watson Studio interface. The top navigation bar includes 'IBM Watson Studio', 'Upkar6 Liddler6's Account', and tabs for 'Overview', 'Assets', 'Environments', 'Bookmarks', 'Deployments', 'Access Control', 'Settings', 'Load', 'Files', and 'Catalog'. The 'Load' tab is active. On the left, under 'Data assets', there is a message: 'You don't have any data assets yet' with a note to use the 'Load page' to upload files. A pink arrow points from this area to the 'Data assets' section on the right. The right side shows a search bar and a table with one asset listed:

<input type="checkbox"/>	NAME	TYPE	CREATED BY	LAST MODIFIED	ACTIONS
<input type="checkbox"/>	boston_house_prices.csv	Data Asset	Upkar6 Liddler6	21 May 2019, 1:55:20 pm	

<http://bit.ly/boston-house-csv>

Step 10 - add Machine Learning model to the project

The screenshot shows the IBM Watson Studio interface. At the top, there's a navigation bar with 'IBM Watson Studio', a bell icon, 'Upkar6 Lidder6's Account', and a user profile icon. Below the navigation bar, the main area shows 'My Projects / My Data Science Project'. A prominent blue button labeled 'Add to project' is highlighted with a pink border. A modal window titled 'Choose asset type' is open in the center. On the left of the modal, there's a sidebar with 'Overview', 'What's new', and a 'Data' section indicating '0 asset s'. The main content of the modal is titled 'AVAILABLE ASSET TYPES' and lists several options: 'Connection', 'Connected data', 'Notebook', 'Image classification ...', 'Object detection mo...', 'Natural Language Cl...', 'Experiment', 'Modeler flow', 'Watson Machine Le...', 'Data Refinery flow', 'Streams flow', and 'Synthesized neural n...'. The 'Watson Machine Le...' option is also highlighted with a pink border. In the bottom right corner of the modal, there's a 'Close' button.

Step 11 - associate a ML instance

The screenshot shows the 'New model' creation interface in IBM Watson Studio. The top navigation bar includes the 'IBM Watson Studio' logo, a bell icon, 'Upkar6 Lidder6's Account', and a user profile icon labeled 'UL'. The main form is titled 'Define model details'.

Name: my_machine_learning

Description: Model description

Machine Learning Service: No Machine Learning service instances associated with your project.

[Associate a Machine Learning service instance](#) with your project on the project settings page, then click the reload button below to refresh the instances available for association with your new model builder instance.

Select model type: Model builder From file From sample

Select runtime: (options not visible)

Buttons: Cancel, Create

Step 12a - create ML instance if you don't have one

The screenshot shows the IBM Watson Studio interface. At the top, there's a dark header bar with the "IBM Watson Studio" logo, a bell icon for notifications, the user account "Upkar6 Lidder6's Account", and a profile icon. Below the header, the main content area has a title "Machine Learning". Underneath it, there are two buttons: "Existing" and "New", with "New" being highlighted and having a pink border. The main content area contains several sections: "Machine Learning" (describing the service as a full-service IBM Cloud offering), "Features" (with subsections for "Machine Learning features" and "Wide choice of interfaces"), and "Integration with Watson Studio" (describing how to create and train machine learning models). The "Machine Learning features" section includes a detailed description of the service's capabilities.

Machine Learning

Existing New

Machine Learning

IBM Watson Machine Learning is a full-service IBM Cloud offering that makes it easy for developers and data scientists to work together to integrate predictive capabilities with their applications. The Machine Learning service is a set of REST APIs that you can call from any programming language to develop applications that make smarter decisions, solve tough problems, and improve user outcomes.

Features

Machine Learning features

Take advantage of machine learning models management (continuous learning system) and deployment (online, batch, streaming). Select any of widely supported machine learning frameworks: TensorFlow, Keras, Caffe, PyTorch, Spark MLLib, scikit learn, xgboost and SPSS.

Wide choice of interfaces

Use the command line interface and Python client to manage your artifacts. Extend your application with artificial intelligence through the Watson Machine Learning REST API.

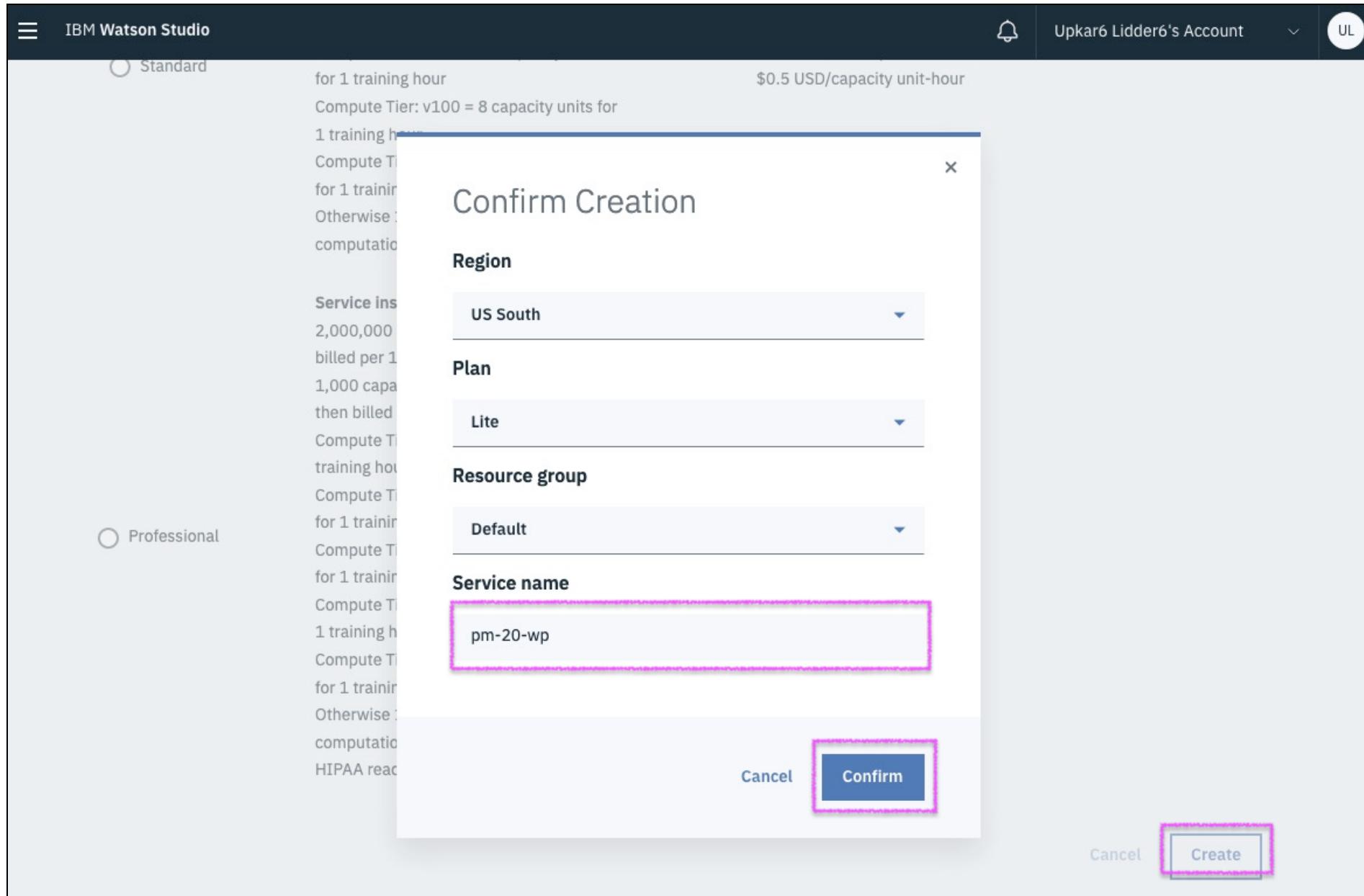
Integration with Watson Studio

Create and train machine learning models with the best tools and the latest expertise in a social environment built by and for data scientists.

Step 12b - create a new lite ML instance

Pricing Plan: Monthly Process shown above reflect the: United States		
PLAN	FEATURES	PRICING
 Lite	<p>Service instance (5 models per instance)</p> <p>5,000 predictions</p> <p>50 capacity unit-hours:</p> <p>Compute Tier: k80 = 2 capacity units for 1 training hour</p> <p>Compute Tier: k80x2 = 4 capacity units for 1 training hour</p> <p>Compute Tier: k80x4 = 8 capacity units for 1 training hour</p> <p>Otherwise 1 capacity unit for 1 computation hour</p> <p>Max 8 k80 GPUs (Deep Learning Training)</p>	Free
<p>The lite plan instance of the IBM Watson Machine Learning service provides you with a maximum of 5 deployed models, 5,000 predictions per month, and 50 capacity unit-hours per month during which model can be trained, evaluated, and deployed to be available to accept prediction events, with a minimum of 1 minute per training job.</p>		

Step 12c - create a new lite ML instance



Step 12d - reload the ML service section

The screenshot shows the 'New model' dialog in IBM Watson Studio. At the top, there's a header with the IBM Watson Studio logo, a bell icon, the account name 'Upkar6 Lidder6's Account', and a user icon. Below the header, the title 'New model' is displayed. The main area is titled 'Define model details' and contains fields for 'Name' (set to 'my_machine_learning') and 'Description' (with placeholder text 'Model description'). Below this, the 'Machine Learning Service' section indicates 'No Machine Learning service instances associated with your project.' It provides instructions to 'Associate a Machine Learning service instance' via project settings and a 'Reload' button, which is highlighted with a pink box. Under 'Select model type', the 'Model builder' option is selected. The 'Select runtime' section is partially visible at the bottom. At the very bottom right are 'Cancel' and 'Create' buttons.

New model

Define model details

Name

my_machine_learning

Description

Model description

Machine Learning Service

No Machine Learning service instances associated with your project.

[Associate a Machine Learning service instance](#) with your project on the project settings page, then click the reload button below to refresh the instances available for association with your new model builder instance.

Reload

Select model type

Model builder From file From sample

Select runtime

Cancel Create

Step 13 - pick new service and create a model manually

The screenshot shows the 'New model' creation interface in IBM Watson Studio. At the top, there's a header bar with the 'IBM Watson Studio' logo, a bell icon, 'Upkar6 Lidder6's Account', and a user profile icon. Below the header, the page title is 'New model'. A dropdown menu for 'Machine Learning Service' is open, showing 'pm-20-wp' with a pink border around it. The next section is 'Select model type', where 'Model builder' is selected (indicated by a pink border). There are also options 'From file' and 'From sample'. The 'Select runtime' section follows, with a note that only Spark environments supporting Scala kernels can be used. A dropdown menu for 'Default Spark Scala 2.11' is shown with a pink border. Below this, a note states: 'The selected runtime uses one driver with 1 vCPU and 4 GB RAM, and 2 executors each with 1 vCPU and 4 GB RAM. This runtime consumes 1.5 capacity units per hour.' A warning message in yellow says: '⚠ Your Spark runtime will be automatically stopped when you save your model, or after 3 hours of inactivity. To avoid consuming extra capacity unit hours delete your model builder instance or [stop your runtime](#) when you are finished with it.' A link to 'Learn more about capacity unit hours and Watson Studio pricing plans.' is provided. At the bottom, there are two options: 'Automatic' (Prepare my data and create a model automatically) and 'Manual' (Let me prepare my data and select which models to train), with 'Manual' highlighted by a pink border. Finally, there are 'Cancel' and 'Create' buttons at the bottom right, with 'Create' also highlighted by a pink border.

New model

Machine Learning Service

pm-20-wp

Select model type

Model builder From file From sample

Select runtime

Only Spark environments supporting Scala kernels can be used for model builder creation.

Default Spark Scala 2.11

The selected runtime uses one driver with 1 vCPU and 4 GB RAM, and 2 executors each with 1 vCPU and 4 GB RAM. This runtime consumes 1.5 capacity units per hour.

⚠ Your Spark runtime will be automatically stopped when you save your model, or after 3 hours of inactivity.
To avoid consuming extra capacity unit hours delete your model builder instance or [stop your runtime](#) when you are finished with it.

[Learn more about capacity unit hours and Watson Studio pricing plans.](#)

Automatic
Prepare my data and create a model automatically

Manual
Let me prepare my data and select which models to train

Cancel Create

Step 14 - pick the Boston data file as *data asset*

The screenshot shows the IBM Watson Studio interface. The top navigation bar includes 'IBM Watson Studio', a bell icon, 'Upkar6 Liddler6's Account', and a user icon labeled 'UL'. Below the navigation is a breadcrumb path: 'My Projects / My Data Science Project / my_machine_learning'. To the right of the path are several icons: a person, a gear, a refresh, a magnifying glass, a grid, and a gear with a pencil.

The main content area has a left sidebar with 'Select Data' (highlighted in blue), 'Train', and 'Evaluate'. The main panel title is 'Select data asset'. It contains a message: 'The model builder currently supports CSV files and IBM Db2 Warehouse on Cloud data assets.' and a 'Add Data Assets' button with a plus sign. Below this is a search bar with the placeholder 'What asset are you looking for?'. A table lists data assets:

Name	Type	Service
boston_house_prices.csv	Data Asset	Project

At the bottom are 'Close' and 'Next' buttons, with 'Next' highlighted by a pink box.

Step 15 - pick target, input features and model type

The screenshot shows the IBM Watson Studio interface in the 'Train' mode. The top navigation bar includes 'IBM Watson Studio', a user account dropdown for 'Upkar6 Lidder6's Account', and a 'UL' button. The left sidebar has tabs for 'Select Data', 'Train' (which is selected), and 'Evaluate'. The main area is titled 'Select a technique'.

Column value to predict (Label Col): MEDV (Decimal) (highlighted with a pink box)

Feature columns: All (default)

Configured estimators: Add Estimators +

Model Techniques:

- Binary Classification:** Classify new data into defined categories based on existing data. Choose if your label column contains two distinct categories.
- Multiclass Classification:** Classify new data into defined categories based on existing data. Choose if your label column contains a discrete number of categories.
- Regression:** Predict values from a continuous set of values. Choose if your label column contains a large number of values. (highlighted with a pink box)

Validation Split: Train: 60 % Test: 20 % Holdout: 20 %

Buttons at the bottom include 'Close', 'Previous', and 'Next'.

Step 16 - add estimator(s)

The screenshot shows the IBM Watson Studio interface with the title "Step 16 - add estimator(s)". The main window displays a "Select estimator(s)" dialog. On the left, there are navigation tabs: "Select" (highlighted), "Train", and "Evaluate". A search bar at the top right contains the placeholder text "What type of estimator are you looking for?". Below the search bar, five estimator options are listed in cards:

- Linear Regression**: Models the linear relationship between a scalar-dependent variable y and one or more explanatory variables (or independent...)
- Decision Tree Regressor**: Maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in...)
- Random Forest Regressor**: Constructs multiple decision trees to produce the mean prediction of each decision tree. It supports both continuous and c...
- Gradient Boosted Tree Regressor**
- Isotonic Regression**: Models the isotonic relationship of a

A pink rectangular box highlights the "Linear Regression" card. In the top right corner of the dialog, there is a button labeled "Add Estimators". At the bottom right of the dialog, there are buttons for "Cancel", "Add", "Close", "Previous", and "Next".

Step 17 - change hyperparameters (optional)

My Projects / My Data Science Project / my_machine_learning

Select Data Train Evaluate

Select a technique

You cannot change label column, feature columns, model type, or validation split after adding an estimator.
You must first delete all estimators in order to make changes to these attributes.

Column value to predict (Label Col)
MEDV (Decimal)

Feature columns
All (default)

Configured estimators

Linear Regression Not Yet Trained

Binary Classification
Classify new data into defined categories based on existing data. Choose if your label column contains two distinct categories.

Multiclass Classification
Classify new data into defined categories based on existing data. Choose if your label column contains a discrete number of categories.

Regression
Predict values from a continuous set of values. Choose if your label column contains a large number of values.

Validation Split
Train: 60 % Test: 20 % Holdout: 20 %

Close **Previous** **Next**

Step 18 - save/store the trained model on IBM Cloud

The screenshot shows the IBM Watson Studio interface for a Data Science Project named "my_machine_learning". The "Evaluate" tab is currently selected. A table displays the results of a Linear Regression model, which has been trained and evaluated. The table includes columns for Estimator Type, Status, Root Mean Squared Error, Mean Squared Error, R2, Explained Variance, Mean Absolute Error, Last Evaluation, and Actions. The "Actions" column contains a "Save" button, which is highlighted with a pink box.

Estimator Type	Status	Root Mean Squared Error	Mean Squared Error	R2	Explained Variance	Mean Absolute Error	Last Evaluation	Actions
LinearRegression	Trained & Evaluated	4.73187	22.39055	0.72876	75.14468	3.43486	21 May 2019, 2:29 PM	<button>Save</button>

Step 19 - add a new deployment

My Projects / My Data Science Project / my_machine_learning

MODEL

my_machine_learning

Overview Evaluation Deployments Lineage

NAME STATUS

Your model is not deployed.

Define deployment details

Name

Description

Add Deployment

Deployment type

Web service Batch prediction Realtime streaming prediction

Cancel Save

IBM Developer @lidderupk

Step 20 - ensure that deployment is successful

1

This screenshot shows the 'Deployments' tab for a model named 'my_machine_learning'. A single deployment entry is listed:

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
boston-housing-linear-regression	INITIALIZING	Web Service	

The 'NAME' and 'STATUS' columns are highlighted with pink boxes, and a pink arrow points from the 'INITIALIZING' status to the second screenshot below.

2

This screenshot shows the same 'Deployments' tab for the same model. The deployment status has changed to 'DEPLOY_SUCCESS'.

NAME	STATUS	DEPLOYMENT TYPE	ACTIONS
boston-housing-linear-regression	DEPLOY_SUCCESS	Web Service	

The 'NAME' and 'STATUS' columns are highlighted with pink boxes, and a pink arrow points from the 'DEPLOY_SUCCESS' status to the final step in the sequence.

Step 21a - implementation / test the deployed model

boston-housing-linear-regression

Overview Implementation Test

Implementation

[View API Specification](#)

Scoring End-point https://us-south.ml.cloud.ibm.com/v3/wml_instances/9c828a1e-3061-4be0-8425-60d890887250/deployments/eaeb3435-fa94-4c7f-bd94-732f627a2cf7/online

Authorization: Bearer <token> See code snippets below for information on how to retrieve the WML Authorization Token to be passed with scoring requests.

Content-type: application/json Required if the request body is sent in JSON format.

Code Snippets

cURL Java JavaScript Python Scala

```
import urllib3, requests, json

# retrieve your wml_service_credentials_username, wml_service_credentials_password, and wml_service_credentials_url from the
# Service credentials associated with your IBM Cloud Watson Machine Learning Service instance

wml_credentials={
    "url": wml_service_credentials_url,
    "username": wml_service_credentials_username,
    "password": wml_service_credentials_password
}
```

Step 21b - implementation / test the deployed model

```
1 {
2   "fields": [ "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT" ],
3   "values": [
4     [0.00632, 18, 2.31, 0, 0.538, 6.575, 66, 4.09, 1, 296, 15.3, 396.9, 4.99]
5   ]
6 }
```

Overview Implementation **Test**

Enter input data

CRIM
0.00632

ZN
18

INDUS
2.31

CHAS

Predict

0.7604992319356272,
0.3330389777682168,
0,
4.55334418055325,
9.114173857821418,
2.3200124382718625,
1.8740219118675838,
0.11380329238192755,
1.7244220002108668,
7.376180771626507,
4.110920792065207,
0.6920487903272683
],
]
}
30.03193787294093
]
}

boston-housing-linear-regression

Overview Implementation **Test**

Enter input data

{
 "fields": ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD",
 "TAX", "PTRATIO", "B", "LSTAT"],
 "values": [
 [0.00632, 18, 2.31, 0, 0.538, 6.575, 66, 4.09, 1, 296, 15.3, 396.9, 4.99]
]
}

Predict

0.7604992319356272,
0.3330389777682168,
0,
4.55334418055325,
9.114173857821418,
2.3200124382718625,
1.8740219118675838,
0.11380329238192755,
1.7244220002108668,
7.376180771626507,
4.110920792065207,
0.6920487903272683
],
]
}
30.03193787294093
]
}

Watson Studio - Binary Classification

Estimator	Description
Logistic regression	Analyzes a data set in which there are one or more independent variables that determine one of two outcomes. Only binary logistic regression is supported.
Decision tree classifier	Maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It supports both binary and multiclass labels, as well as both continuous and categorical features.
Random forest classifier	Constructs multiple decision trees to produce the label that is a mode of each decision tree. It supports both binary and multiclass labels, as well as both continuous and categorical features.
Gradient boosted tree classifier	Produces a classification prediction model in the form of an ensemble of decision trees. It only supports binary labels, as well as both continuous and categorical features.

Watson Studio - Multiclass Classification

Estimator	Description
Decision tree classifier	Maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It supports both binary and multiclass labels, as well as both continuous and categorical features.
Random forest classifier	Constructs multiple decision trees to produce the label that is a mode of each decision tree. It supports both binary and multiclass labels, as well as both continuous and categorical features.
Naive Bayes	Classifies features based on Bayes' theorem, which assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Watson Studio - Regression

Estimator	Description
Linear regression	Models the linear relationship between a scalar-dependent variable y and one or more explanatory variables (or independent variables) x .
Decision tree regressor	Maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It supports both continuous and categorical features.
Random forest regressor	Constructs multiple decision trees to produce the mean prediction of each decision tree. It supports both continuous and categorical features.
Gradient boosted tree regressor	Produces a regression prediction model in the form of an ensemble of decision trees. It supports both continuous and categorical features.
Isotonic regression	Models the isotonic relationship of a sequence of observations by fitting a free-form line to the observations under the following constraints: the fitted free-form line must be non-decreasing everywhere, and it must lie as close to the observations as possible.

Watson Machine Learning

WML - Supported Frameworks as of 06.21.19

TensorFlow

- Version 1.5
- Version 1.11 in an Anaconda 5.2.0 environment

Spark MLLib

Spark 2.1

Caffe

Version 1.0

Predictive Model Markup Language (PMML)

Version 3.0 to 4.3

XGBoost

- XGBoost 0.6a2 and 0.71 in an Anaconda 4.2.x environment with scikit-learn 0.17 and Python 3.5
- XGBoost 0.80 in an Anaconda 5.0.1 environment with scikit-learn 0.19 and Python 3.5

Keras

- Version 2.1.3 with Tensorflow version 1.5
- Version 2.2.4 with TensorFlow version 1.11 in an Anaconda 5.2.0 environment

IBM SPSS Modeler

- IBM SPSS Modeler 17.1
- IBM SPSS Modeler 18.0

scikit-learn

- scikit-learn 0.17 on Anaconda 4.2.x for Python 3.5 Runtime
- scikit-learn 0.19 on Anaconda 5.0.0 for Python 3.5 Runtime

PyTorch

Versions: 0.3, 0.4.1, 1.0

IBM Watson Machine Learning

Embed Machine Learning and Deep Learning in your Business

Push analytics to Data

ML close to data for faster and secure insights

- Connect to remote Hadoop and Spark clusters to train ML models at scale
- Move analytics to the data, access data from Hadoop and combine it with RDBMS to expand data access and optimize performance.
- Simplify big data analysis for analysts and business users.
- Mix and match deployment options

Deploy & Manage

Manage models and versions

- Save models on a central repository with version control built in
- Portable models – deploy in the cloud, on devices or on premise
- Import models trained somewhere else and deploy in WML
- Transfer models to connected devices (e.g. Core ML, Tensorflow Lite)
- Deployment flexibility: Shiny apps, scripts, decision optimization and WEX models

Automate ML

Your models always learning with a feedback loop

- Automate the retraining of models with feedback loop capabilities
- Automate the deployment of models in products
- Automate the Hyperparameter Optimization and Feature Engineering to rapidly train models
- A/B testing of models and performance Monitoring

Fully Managed on IBM Cloud

Deploy it on Private or Public Cloud

Embed ML in your business

Operationalize



Deployment methods



Management & Monitoring



Watson Machine Learning

IBM Watson Machine Learning Client

Welcome to watson-machine-learning-client's documentation!

watson-machine-learning-client is a python library that allows to work with Watson Machine Learning service on [IBM Cloud](#). Test and deploy your models as APIs for application development, share with colleagues using this python library.

Contents

- [Welcome to watson-machine-learning-client's documentation!](#)
 - [Installation](#)
 - [Requirements](#)
 - [Supported machine learning frameworks](#)
 - [Sample notebooks](#)
 - [API](#)
 - [service_instance](#)
 - [repository](#)
 - [deployments](#)
 - [training](#)
 - [experiments](#)
 - [learning_system](#)
 - [runtimes](#)
- [Indices and tables](#)

WML - create scikit-learn linear regression model

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn.datasets import load_boston
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 boston = load_boston()
8 X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target)
9
10 # Create a new Linear Regression Model
11 LR_model = LinearRegression()
12
13 # Train the model
14 LR_model.fit(X_train, y_train)
15
16 # store actual and predicted data to draw chart
17 predicted = LR_model.predict(X_test)
18 actual = y_test
19
20
21 # The coefficients
22 print('Coefficients: \n', LR_model.coef_)
23 # The mean squared error
24 print("Mean squared error: %.2f"
25      % mean_squared_error(actual, predicted))
26 # Explained variance score: 1 is perfect prediction
27 print('Variance score: %.2f' % r2_score(actual, predicted))
```

Output



Coefficients:

[-1.05001380e-01	3.29973196e-02	3.09915520e-02	2.74529793e+00
-2.00482943e+01	4.60880933e+00	-1.36509230e-03	-1.27392294e+00
2.69375701e-01	-1.12477385e-02	-9.48125356e-01	8.76288422e-03
-4.24918821e-01]			

Mean squared error: 33.77

Variance score: 0.67

WML - create scikit-learn linear regression model

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn.datasets import load_boston
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 boston = load_boston()
8 X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target)
9
10 # Create a new Linear Regression Model
11 LR_model = LinearRegression()
12
13 # Train the model
14 LR_model.fit(X_train, y_train)
15
16 # store actual and predicted data to draw chart
17 predicted = LR_model.predict(X_test)
18 actual = y_test
19
20
21 # The coefficients
22 print('Coefficients: \n', LR_model.coef_)
23 # The mean squared error
24 print("Mean squared error: %.2f"
25      % mean_squared_error(actual, predicted))
26 # Explained variance score: 1 is perfect prediction
27 print('Variance score: %.2f' % r2_score(actual, predicted))
```

Output



```
Coefficients:
[-1.05001380e-01   3.29973196e-02   3.09915520e-02   2.74529793e+00
 -2.00482943e+01   4.60880933e+00   -1.36509230e-03  -1.27392294e+00
  2.69375701e-01  -1.12477385e-02  -9.48125356e-01   8.76288422e-03
 -4.24918821e-01]
Mean squared error: 33.77
Variance score: 0.67
```

WML - create scikit-learn linear regression model

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn.datasets import load_boston
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 boston = load_boston()
8 X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target)
9
10 # Create a new Linear Regression Model
11 LR_model = LinearRegression()
12
13 # Train the model
14 LR_model.fit(X_train, y_train)
15
16 # store actual and predicted data to draw chart
17 predicted = LR_model.predict(X_test)
18 actual = y_test
19
20
21 # The coefficients
22 print('Coefficients: \n', LR_model.coef_)
23 # The mean squared error
24 print("Mean squared error: %.2f"
25      % mean_squared_error(actual, predicted))
26 # Explained variance score: 1 is perfect prediction
27 print('Variance score: %.2f' % r2_score(actual, predicted))
```

Output



```
Coefficients:
[-1.05001380e-01   3.29973196e-02   3.09915520e-02   2.74529793e+00
 -2.00482943e+01   4.60880933e+00   -1.36509230e-03  -1.27392294e+00
  2.69375701e-01  -1.12477385e-02  -9.48125356e-01   8.76288422e-03
 -4.24918821e-01]
Mean squared error: 33.77
Variance score: 0.67
```

WML - create scikit-learn linear regression model

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn.datasets import load_boston
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 boston = load_boston()
8 X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target)
9
10 # Create a new Linear Regression Model
11 LR_model = LinearRegression()
12
13 # Train the model
14 LR_model.fit(X_train, y_train)
15
16 # store actual and predicted data to draw chart
17 predicted = LR_model.predict(X_test)
18 actual = y_test
19
20
21 # The coefficients
22 print('Coefficients: \n', LR_model.coef_)
23 # The mean squared error
24 print("Mean squared error: %.2f"
25      % mean_squared_error(actual, predicted))
26 # Explained variance score: 1 is perfect prediction
27 print('Variance score: %.2f' % r2_score(actual, predicted))
```

Output



```
Coefficients:
[ -1.05001380e-01   3.29973196e-02   3.09915520e-02   2.74529793e+00
 -2.00482943e+01   4.60880933e+00  -1.36509230e-03  -1.27392294e+00
  2.69375701e-01  -1.12477385e-02  -9.48125356e-01   8.76288422e-03
 -4.24918821e-01]
Mean squared error: 33.77
Variance score: 0.67
```

WML - create scikit-learn linear regression model

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn.datasets import load_boston
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 boston = load_boston()
8 X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target)
9
10 # Create a new Linear Regression Model
11 LR_model = LinearRegression()
12
13 # Train the model
14 LR_model.fit(X_train, y_train)
15
16 # store actual and predicted data to draw chart
17 predicted = LR_model.predict(X_test)
18 actual = y_test
19
20
21 # The coefficients
22 print('Coefficients: \n', LR_model.coef_)
23 # The mean squared error
24 print("Mean squared error: %.2f"
25      % mean_squared_error(actual, predicted))
26 # Explained variance score: 1 is perfect prediction
27 print('Variance score: %.2f' % r2_score(actual, predicted))
```

Output



Coefficients:

-1.05001380e-01	3.29973196e-02	3.09915520e-02	2.74529793e+00
-2.00482943e+01	4.60880933e+00	-1.36509230e-03	-1.27392294e+00
2.69375701e-01	-1.12477385e-02	-9.48125356e-01	8.76288422e-03
-4.24918821e-01]			

Mean squared error: 33.77

Variance score: 0.67

WML - create scikit-learn linear regression model

```
1 from sklearn.linear_model import LinearRegression
2 from sklearn.datasets import load_boston
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import mean_squared_error, r2_score
6
7 boston = load_boston()
8 X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target)
9
10 # Create a new Linear Regression Model
11 LR_model = LinearRegression()
12
13 # Train the model
14 LR_model.fit(X_train, y_train)
15
16 # store actual and predicted data to draw chart
17 predicted = LR_model.predict(X_test)
18 actual = y_test
19
20
21 # The coefficients
22 print('Coefficients: \n', LR_model.coef_)
23 # The mean squared error
24 print("Mean squared error: %.2f"
25      % mean_squared_error(actual, predicted))
26 # Explained variance score: 1 is perfect prediction
27 print('Variance score: %.2f' % r2_score(actual, predicted))
```

Output



```
Coefficients:
[-1.05001380e-01   3.29973196e-02   3.09915520e-02   2.74529793e+00
 -2.00482943e+01   4.60880933e+00   -1.36509230e-03  -1.27392294e+00
  2.69375701e-01  -1.12477385e-02  -9.48125356e-01   8.76288422e-03
 -4.24918821e-01]
Mean squared error: 33.77
Variance score: 0.67
```

WML - evaluation metrics

Version	a38f373b-44cd-4a1e-9cb5-87343a69da12
Phase	setup
meanSquaredError	19.775
rootMeanSquaredError	4.447
r2	0.751
meanAbsoluteError	3.223
explainedVariance	74.316

Coefficients:

```
[ -1.05001380e-01   3.29973196e-02   3.09915520e-02   2.74529793e+00
 -2.00482943e+01   4.60880933e+00   -1.36509230e-03  -1.27392294e+00
  2.69375701e-01  -1.12477385e-02   -9.48125356e-01   8.76288422e-03
 -4.24918821e-01]
```

Mean squared error: 33.77

Variance score: 0.67

WML - get Machine Learning service credentials

WML - save scikit-learn linear regression model

```
1 # we will use WML to work with IBM Machine Learning Service
2 from watson_machine_learning_client import WatsonMachineLearningAPIClient
3
4 # Grab your credentials from the Watson Service section in Watson Studio or IBM Cloud Dashboard
5 wml_credentials = {
6 }
7
8 # Instantiate WatsonMachineLearningAPIClient
9 from watson_machine_learning_client import WatsonMachineLearningAPIClient
10 client = WatsonMachineLearningAPIClient( wml_credentials )
11
12 # store the model
13 published_model = client.repository.store_model(model=LR_model,
14                                                 meta_props={'name':'upkar-housing-linear-reg'},
15                                                 training_data=X_train, training_target=y_train)
```

WML - save scikit-learn linear regression model

```
1 # we will use WML to work with IBM Machine Learning Service
2 from watson_machine_learning_client import WatsonMachineLearningAPIClient
3
4 # Grab your credentials from the Watson Service section in Watson Studio or IBM Cloud Dashboard
5 wml_credentials = {
6 }
7
8 # Instantiate WatsonMachineLearningAPIClient
9 from watson_machine_learning_client import WatsonMachineLearningAPIClient
10 client = WatsonMachineLearningAPIClient( wml_credentials )
11
12 # store the model
13 published_model = client.repository.store_model(model=LR_model,
14                                                 meta_props={'name':'upkar-housing-linear-reg'},
15                                                 training_data=X_train, training_target=y_train)
```

WML - save scikit-learn linear regression model

```
1 # we will use WML to work with IBM Machine Learning Service
2 from watson_machine_learning_client import WatsonMachineLearningAPIClient
3
4 # Grab your credentials from the Watson Service section in Watson Studio or IBM Cloud Dashboard
5 wml_credentials = {
6 }
7
8 # Instantiate WatsonMachineLearningAPIClient
9 from watson_machine_learning_client import WatsonMachineLearningAPIClient
10 client = WatsonMachineLearningAPIClient( wml_credentials )
11
12 # store the model
13 published_model = client.repository.store_model(model=LR_model,
14                                                 meta_props={'name':'upkar-housing-linear-reg'},
15                                                 training_data=X_train, training_target=y_train)
```

WML - save scikit-learn linear regression model

```
1 # we will use WML to work with IBM Machine Learning Service
2 from watson_machine_learning_client import WatsonMachineLearningAPIClient
3
4 # Grab your credentials from the Watson Service section in Watson Studio or IBM Cloud Dashboard
5 wml_credentials = {
6 }
7
8 # Instantiate WatsonMachineLearningAPIClient
9 from watson_machine_learning_client import WatsonMachineLearningAPIClient
10 client = WatsonMachineLearningAPIClient( wml_credentials )
11
12 # store the model
13 published_model = client.repository.store_model(model=LR_model,
14                                                 meta_props={'name':'upkar-housing-linear-reg'},
15                                                 training_data=X_train, training_target=y_train)
```

WML - deploy scikit-learn linear regression model

```
1 import json
2
3 # grab the model from IBM Cloud
4 published_model_uid = client.repository.get_model_uid(published_model)
5
6 # create a new deployment for the model
7 model_deployed = client.deployments.create(published_model_uid, "Deployment of scikit model")
8
9 #get the scoring endpoint
10 scoring_endpoint = client.deployments.get_scoring_url(model_deployed)
11 print(scoring_endpoint)
12
13 #use the scoring endpoint to predict house median price some test data
14 scoring_payload = {"values": [list(X_test[0]), list(X_test[1])]}
15 predictions = client.deployments.score(scoring_endpoint, scoring_payload)
16 print(json.dumps(predictions, indent=2))
```

WML - deploy scikit-learn linear regression model

```
1 import json
2
3 # grab the model from IBM Cloud
4 published_model_uid = client.repository.get_model_uid(published_model)
5
6 # create a new deployment for the model
7 model_deployed = client.deployments.create(published_model_uid, "Deployment of scikit model")
8
9 #get the scoring endpoint
10 scoring_endpoint = client.deployments.get_scoring_url(model_deployed)
11 print(scoring_endpoint)
12
13 #use the scoring endpoint to predict house median price some test data
14 scoring_payload = {"values": [list(X_test[0]), list(X_test[1])]}
15 predictions = client.deployments.score(scoring_endpoint, scoring_payload)
16 print(json.dumps(predictions, indent=2))
```

WML - deploy scikit-learn linear regression model

```
1 import json
2
3 # grab the model from IBM Cloud
4 published_model_uid = client.repository.get_model_uid(published_model)
5
6 # create a new deployment for the model
7 model_deployed = client.deployments.create(published_model_uid, "Deployment of scikit model")
8
9 #get the scoring endpoint
10 scoring_endpoint = client.deployments.get_scoring_url(model_deployed)
11 print(scoring_endpoint)
12
13 #use the scoring endpoint to predict house median price some test data
14 scoring_payload = {"values": [list(X_test[0]), list(X_test[1])]}
15 predictions = client.deployments.score(scoring_endpoint, scoring_payload)
16 print(json.dumps(predictions, indent=2))
```

WML - deploy scikit-learn linear regression model

```
1 import json
2
3 # grab the model from IBM Cloud
4 published_model_uid = client.repository.get_model_uid(published_model)
5
6 # create a new deployment for the model
7 model_deployed = client.deployments.create(published_model_uid, "Deployment of scikit model")
8
9 #get the scoring endpoint
10 scoring_endpoint = client.deployments.get_scoring_url(model_deployed)
11 print(scoring_endpoint)
12
13 #use the scoring endpoint to predict house median price some test data
14 scoring_payload = {"values": [list(X_test[0]), list(X_test[1])]}
15 predictions = client.deployments.score(scoring_endpoint, scoring_payload)
16 print(json.dumps(predictions, indent=2))
```

WML - deploy scikit-learn linear regression model

```
#####
Synchronous deployment creation for uid: [REDACTED] started
#####
INITIALIZING
DEPLOY_SUCCESS

-----
Successfully finished deployment creation, deployment_uid=[REDACTED]
-----

https://us-south.ml.cloud.ibm.com/v3/wml\_instances/\[REDACTED\]/online
{
  "values": [
    [
      10.750284346772386
    ],
    [
      17.59893318342104
    ]
  ],
  "fields": [
    "prediction"
  ]
}
```

WML - try it out on your own !

The screenshot shows the IBM Watson Studio interface. At the top, there's a navigation bar with 'IBM Watson Studio', 'Upkar6 Lidder6's Account', and a user icon. Below the navigation bar, the main header says 'My Projects / My Data Science Project'. A blue box highlights the 'Add to project' button in the top right corner of the header. The main content area has tabs for 'Overview', 'Assets' (which is selected), 'Environments', 'Bookmarks', 'Deployments', 'Access Control', and 'Settings'. A search bar says 'What assets are you looking for?'. Below it, a section titled 'Data assets' shows '0 asset selected'. A sidebar on the left lists 'Modeler flows', 'Watson Assistant', and 'Experiments'. On the right, there's a 'Load' tab, 'Files' tab, and a 'Catalog' tab. A large central panel is titled 'Choose asset type' and contains a grid of 'AVAILABLE ASSET TYPES'. The 'Notebook' option is highlighted with a blue box. Other options include 'Data', 'Connection', 'Connected data', 'Dashboard', 'Image classification model', 'Object detection model', 'Natural Language Classifier', 'Watson Machine Learning', 'Experiment', 'Modeler flow', 'Data Refinery flow', 'Streams flow', and 'Synthesized neural network'. A dashed box on the right says 'Drop files here or [browse](#) load.' At the bottom right of the central panel is a 'Close' button.

<http://bit.ly/waston-ml-sign>

WML - create a new notebook from URL

The screenshot shows the IBM Watson Studio interface for creating a new notebook. The top navigation bar includes 'IBM Watson Studio', 'Upkar6 Lidder6's Account', and a user icon. The breadcrumb path is 'My Projects / My Data Science Project / Add Notebook'. The main form is titled 'New notebook' and has three tabs: 'Blank', 'From file', and 'From URL', with 'From URL' selected. The 'Name*' field contains 'boston-housing' with 36 characters remaining. The 'Notebook URL*' field contains 'https://github.com/lidderupk/watson-studio-ml/blob/master/ass'. The 'Select runtime*' dropdown is set to 'Default Python 3.5 Free (1 vCPU and 4 GB RAM)'. A note below states: 'The selected runtime has 1 vCPU and 4 GB RAM and is free. Learn more about capacity unit hours and Watson Studio pricing plans.' At the bottom are 'Cancel' and 'Create Notebook' buttons.

New notebook

Blank From file **From URL**

Name*

boston-housing 36 Characters Remaining

Description

Type your Description here

Notebook URL*

https://github.com/lidderupk/watson-studio-ml/blob/master/ass

Select runtime* Includes notebook environments [\(i\)](#)

Default Python 3.5 Free (1 vCPU and 4 GB RAM)

The selected runtime has 1 vCPU and 4 GB RAM and is free.
Learn more about capacity unit hours and Watson Studio pricing plans.

Cancel **Create Notebook**

Grab the FULL URL from : <http://bit.ly/boston-house-notebook>

Thank you

Let's chat !

Upkar Lidder, IBM

@lidderupk

<https://github.com/lidderupk/>

ulidder@us.ibm.com

#BehindtheCode



It's about time
developers got
some attention.




Change photo

Build Smart

↳



Part of **IBM Developer** – 29 groups [?](#)

IBM Developer SF Bay Area

📍 San Francisco, CA

👤 7,164 members · Public group [?](#)

👤 Organized by Angie K and 6 others

IBM Developer

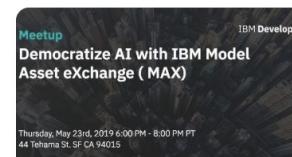


Share: [f](#) [t](#) [in](#) [↗](#)

THU, MAY 23, 6:00 PM

Meetup: Democratize AI with IBM Model Asset eXchange (MAX)

📍 44 Tehama St



Have you ever wanted to classify images, recognize faces or places in images, process natural language or text, or create recommendations based on time-series data in one of your applications? Join IBM Developer's meetup and learn how! With deep-learning (machine learnin...



61 attendees

Manage [▼](#)



Going

WED, MAY 29, 9:30 AM

Online Meetup: The what, the why, and the how of Knative on Kubernetes

📍 Needs a location



RSVP on Crowdcast NOW! <https://www.crowdcast.io/e/the-what-the-why-and-the> ! Please do not forget to register on Crowdcast and join us using Chrome browser via Crowdcast on the event date! Wouldn't it be nice if as developers we could just focus on our code? That is the...



42 attendees

Manage [▼](#)

Attend

IBM Partners

Enabling Independent Software Vendors (ISVs)
and tech companies for growth

Target audience

- ISVs and tech companies building and selling cloud solutions
- New to IBM Cloud
- Startups who aspire to build and sell their own solutions

Offers to help you get started



Build with up to \$12,000 of free IBM Cloud™ credits (\$1,000 per month for 12 months)

Integrate your solutions with leading-edge IBM Cloud technologies to deliver more innovation and value to your clients. Access more than **130 unparalleled services** including Watson™, Analytics and Security.



Build with 10TB of IBM Cloud Object Storage at no charge

Build data capability into your offering. IBM Cloud Object Storage is designed for high durability, resiliency and security.



Build with IBM Watson Assistant with a 1-year free trial

Receive access to 100K API calls per month plus 10 workspaces. Build and deploy chatbots quickly and efficiently with IBM Watson Assistant's advanced capabilities and seamless interface.



Build with IBM Cloud Kubernetes Service with a 1-year free trial

Containerize your solution with 1TB of block storage. Ship all your applications in one agile, well-defined structure with IBM Cloud Kubernetes Service.



Build with IBM Blockchain with a 6-month free trial

Build a network with up to 3 organizations to prototype. Build a secure business transaction network for your clients using blockchain and smart contracts.



Finished building and testing? Go-to-market with IBM

Access Provider Workbench, attend an orientation session and join the premier network of over 400 partners who are already listing their solutions on the IBM Marketplace.



Is your business a Startup? Build with up to \$120,000 in IBM Cloud credits

If your business revenue in the last 12 months is less than \$1M and you've been in business for fewer than five years, then you may qualify for Startup with IBM.

Get started

Experience IBM's countless partner benefits. Start building and selling with IBM today.

Learn more and access offers at
ibm.com/partners/start