

Introduction to Machine Learning on IBM Cloud

IBM Developer

Upkar Lidder

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

<http://bit.ly/ibm-cloud-summit-2019>
<http://bit.ly/upkar-autoai>

Prerequisites

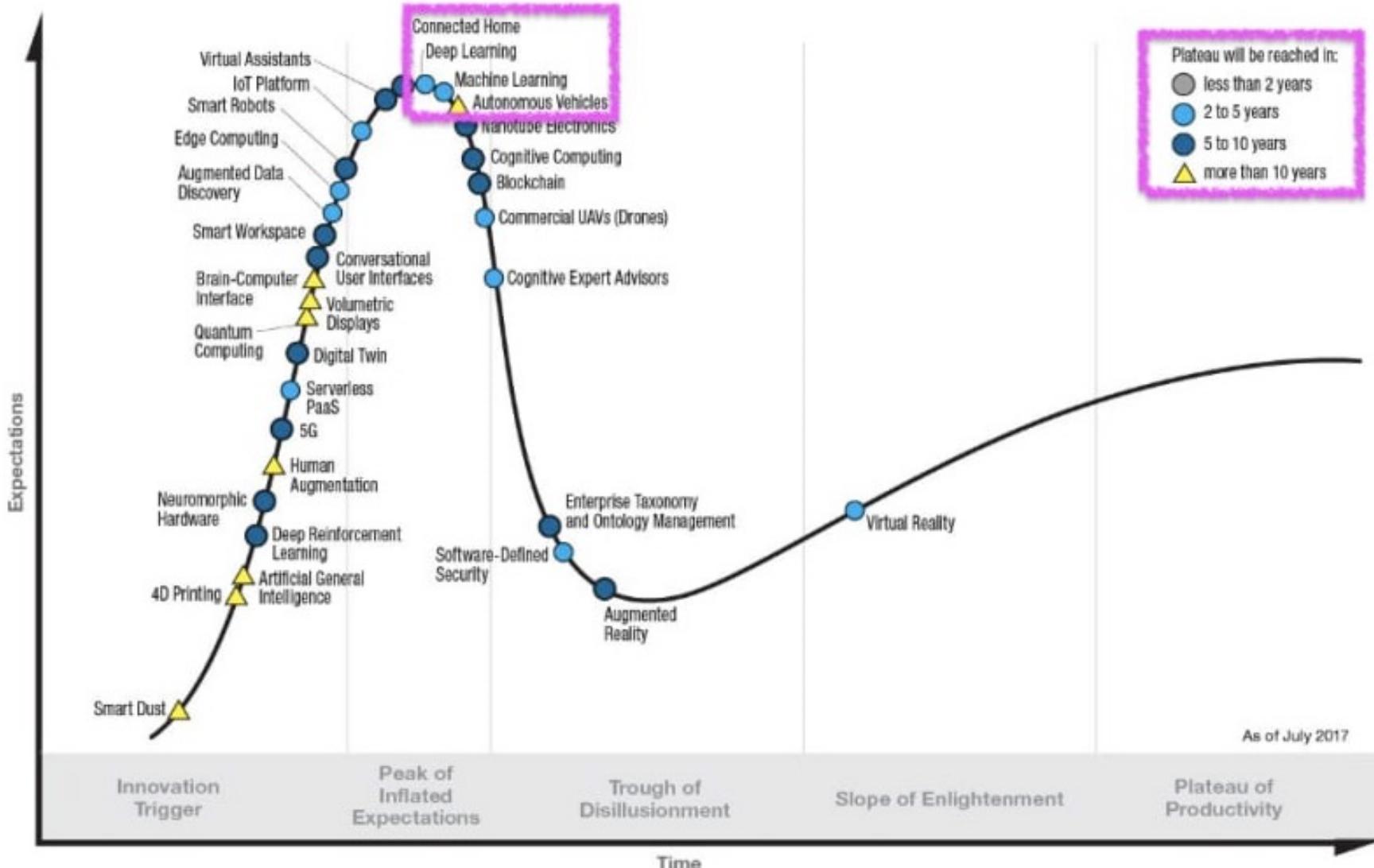
1. Create IBM Cloud Account using THIS URL

<http://bit.ly/ibm-cloud-summit-2019>

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.

ML Hype

Gartner Hype Cycle for Emerging Technologies, 2017



Source: Gartner (July 2017)

© 2017 Gartner, Inc. and/or its affiliates. All rights reserved.

ML on IBM Cloud - Cognitive Services

Natural Language Processing

Watson Assistant Lite • IBM • IAM-enabled

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

Natural Language Understanding Lite • IBM • IAM-enabled

Analyze text to extract meta-data from content such as concepts, entities, emotion, relations, sentiment and more.

Voice Agent with Watson Lite • IBM • IAM-enabled

Create a cognitive voice agent that uses Watson services to speak directly with customers using natural language over the telephone

Visual Recognition

Visual Recognition Lite • IBM • IAM-enabled

Find meaning in visual content! Analyze images for scenes, objects, faces, and other content. Choose a default model off the shelf, or create...

Discovery Lite • IBM • IAM-enabled

Add a cognitive search and content analytics engine to applications.

Personality Insights Lite • IBM • IAM-enabled

The Watson Personality Insights derives insights from transactional and social media data to identify psychological traits.

Language Translator Lite • IBM • IAM-enabled

Translate text, documents, and websites from one language to another. Create industry or region-specific translations via the service's...

Speech to Text Lite • IBM • IAM-enabled

Low-latency, streaming transcription

Text to Speech Lite • IBM • IAM-enabled

Synthesizes natural-sounding speech from text.

Tone Analyzer Lite • IBM • IAM-enabled

Tone Analyzer uses linguistic analysis to detect three types of tones from communications: emotion, social, and language. This insight can...

ML on IBM Cloud - Guided ML

To help simplify an AI lifecycle management, AutoAI automates:

- Data preparation
- Model development
- Feature engineering
- Hyper parameter optimization

The screenshot displays the AutoAI interface for managing machine learning pipelines. At the top, a message indicates "Run finished" with "4 PIPELINES GENERATED" and a duration of "115 seconds". Below this, a timeline shows the stages of pipeline generation: Read dataset, Split holdout data, Read training data, Preprocessing, Model selection, Gradient boosting regressor (highlighted with a teal dot), Hyperparameter optimization, and Feature engine. A callout box highlights the "Transformers" used in the "Gradient boosting regressor" stage, which include Preprocessing, Cube root, Univariate feature selection, Product, Univariate feature selection, and Gradient boosting regressor estimator.

Pipeline leaderboard

RANK	PIPELINE	RMSE	ESTIMATOR	ENHANCEMENTS
1	Pipeline 3	3.009	Gradient boosting regressor	HPO FE

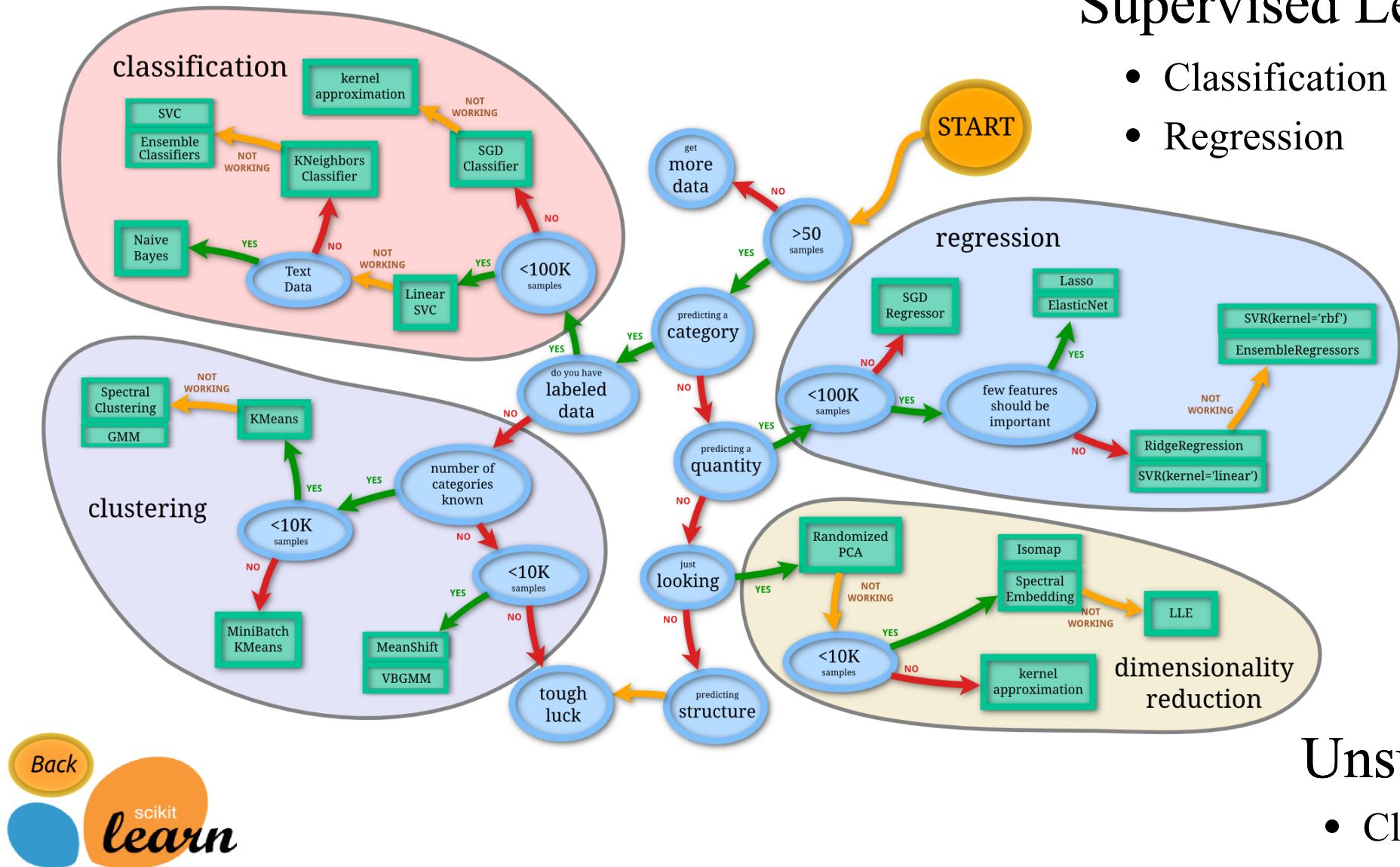
Model evaluation measures

	Training score	Holdout score
Explained Variance	0.895	0.860
MAE	2.190	2.040
MSE	9.059	8.318
MSLE	0.021	0.023
MedAE	1.619	1.428

Feature Importance

Feature	Importance
RM	0.28
NewFeature_1...	0.21
NewFeature_7...	0.11
NewFeature_1...	0.11
etc	0.07

ML Map



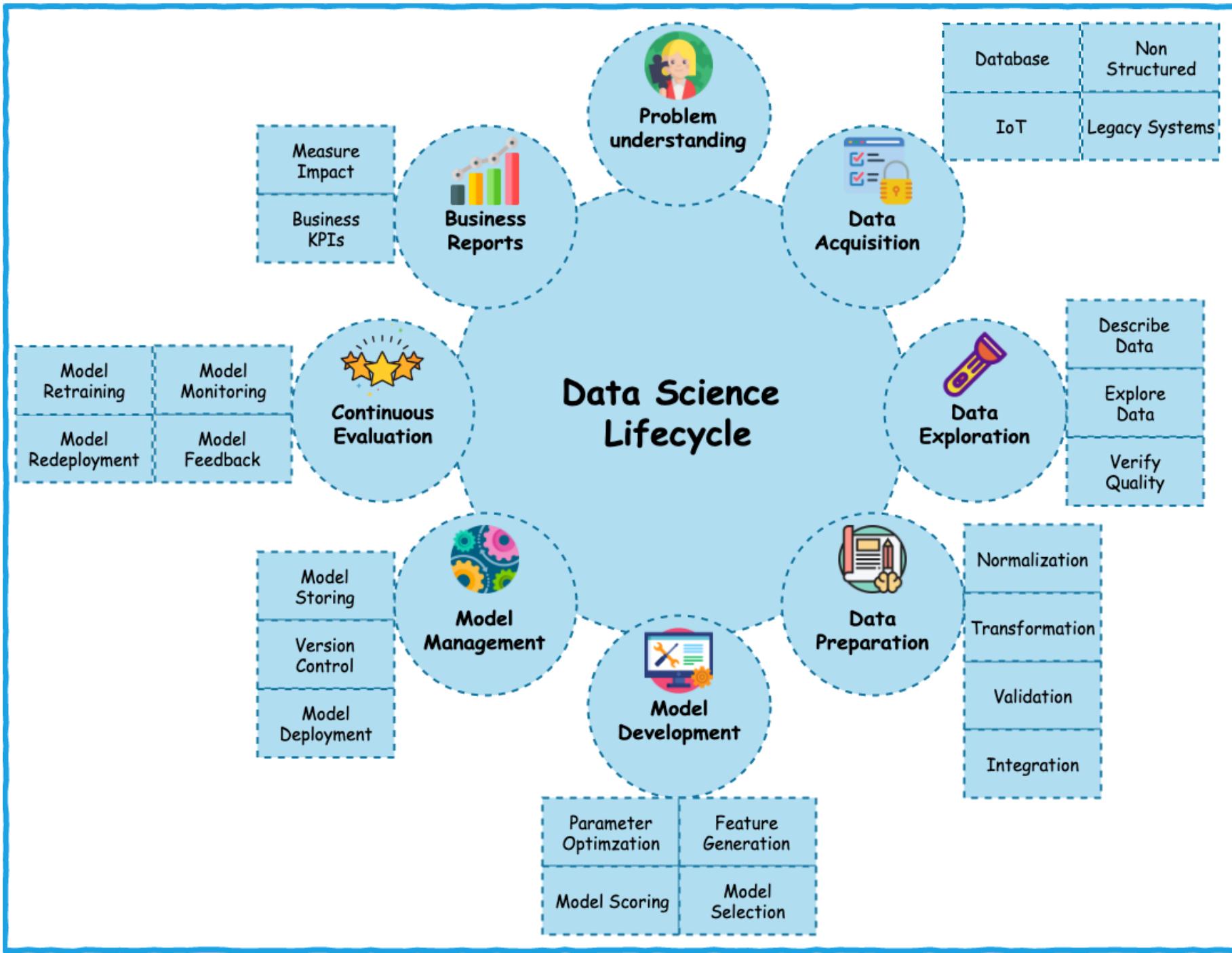
Supervised Learning

- Classification
- Regression

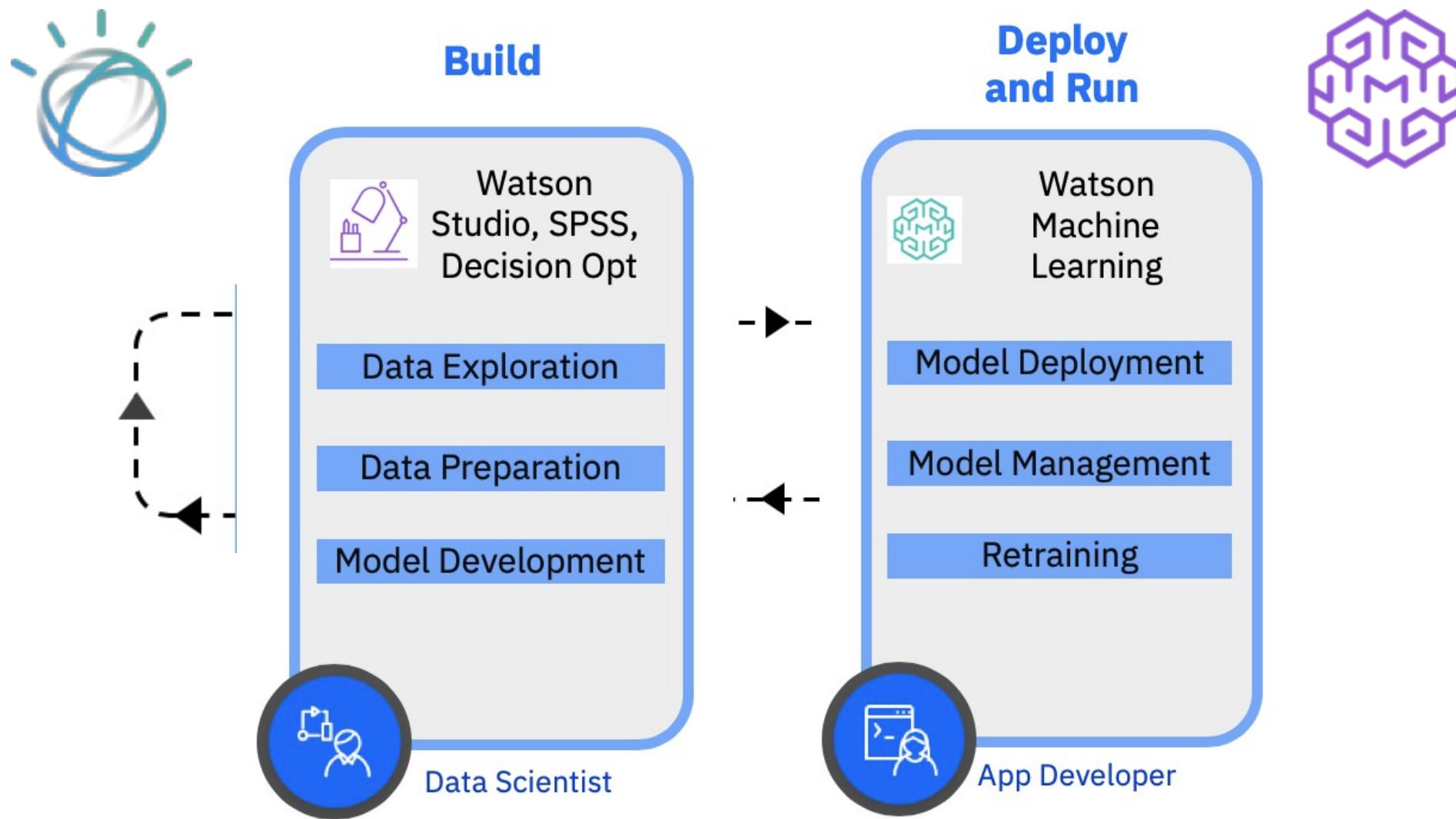
Unsupervised Learning

- Clustering
- Dimensionality Reduction

ML Lifecycle



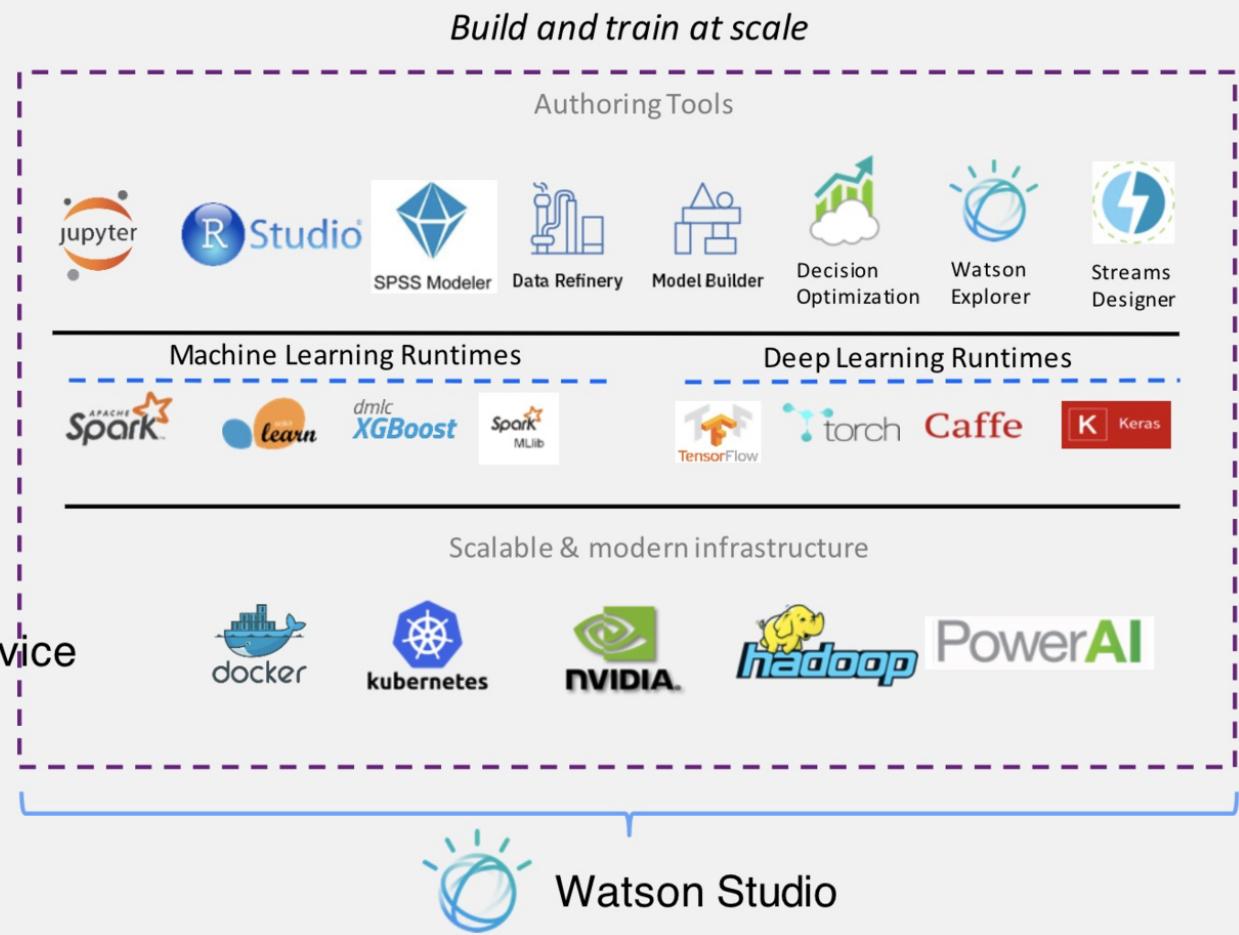
Watson Studio & Watson Machine Learning



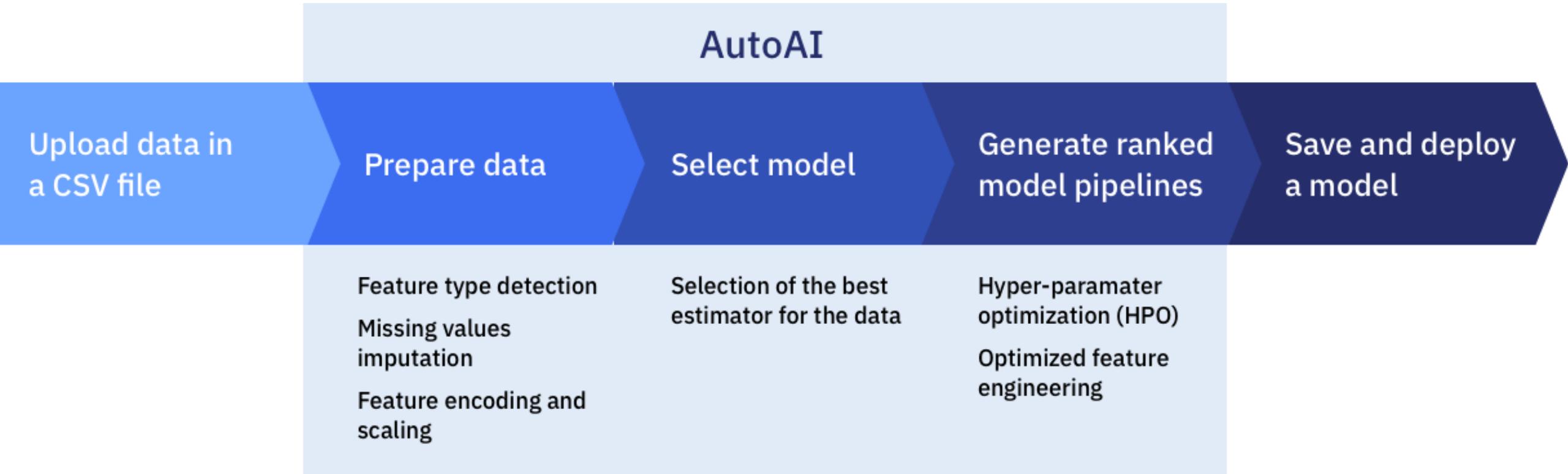
Watson Studio

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



AutoAI



IBM Watson Studio - project based development platform

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
- 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
 - Modeler flow
 - Data Refinery flow
 - Natural Language Cl...
 - Streams flow
 - Synthesized neural n...

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 [MEDV] 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.

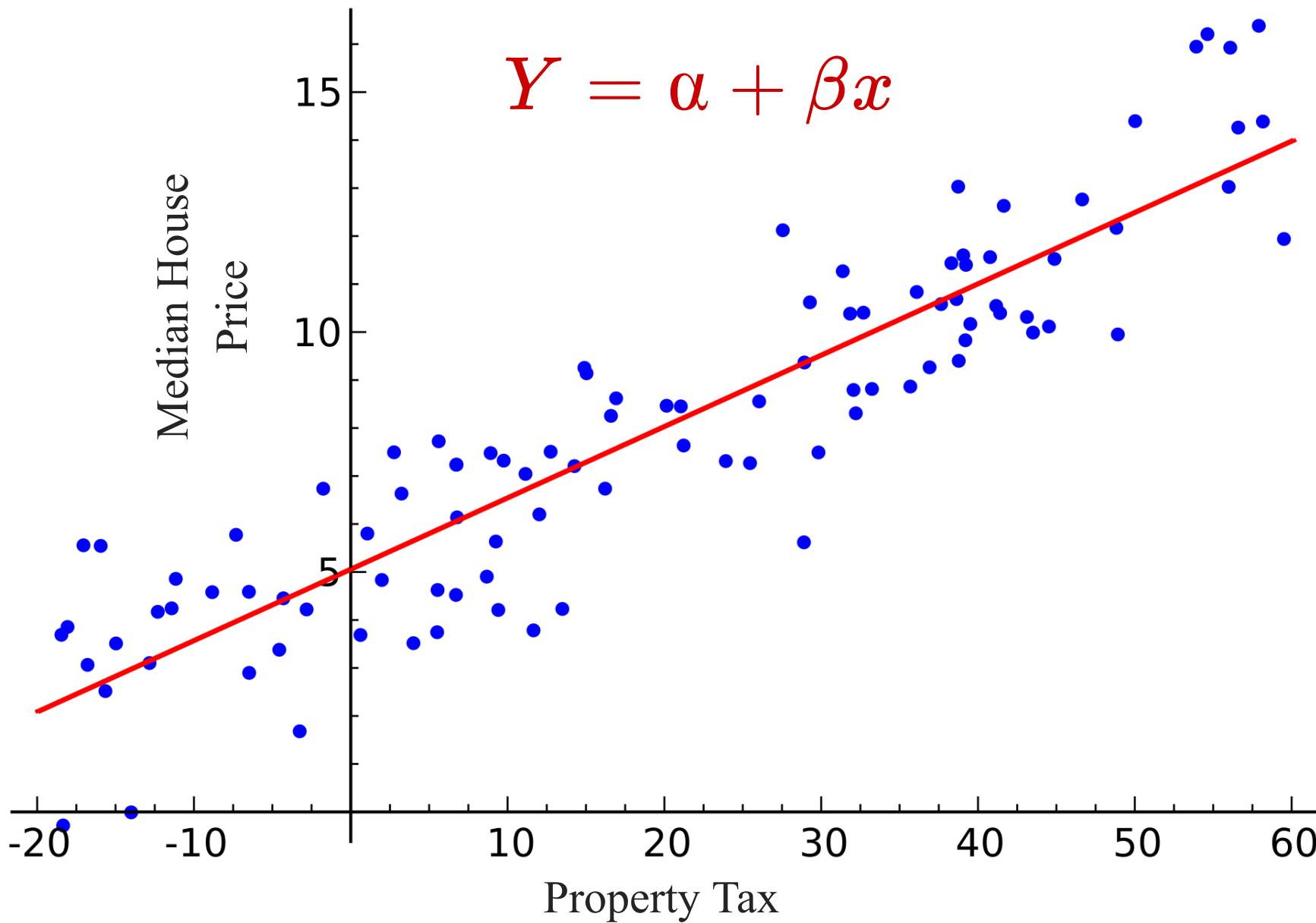
<http://archive.ics.uci.edu/ml/datasets/Housing>

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

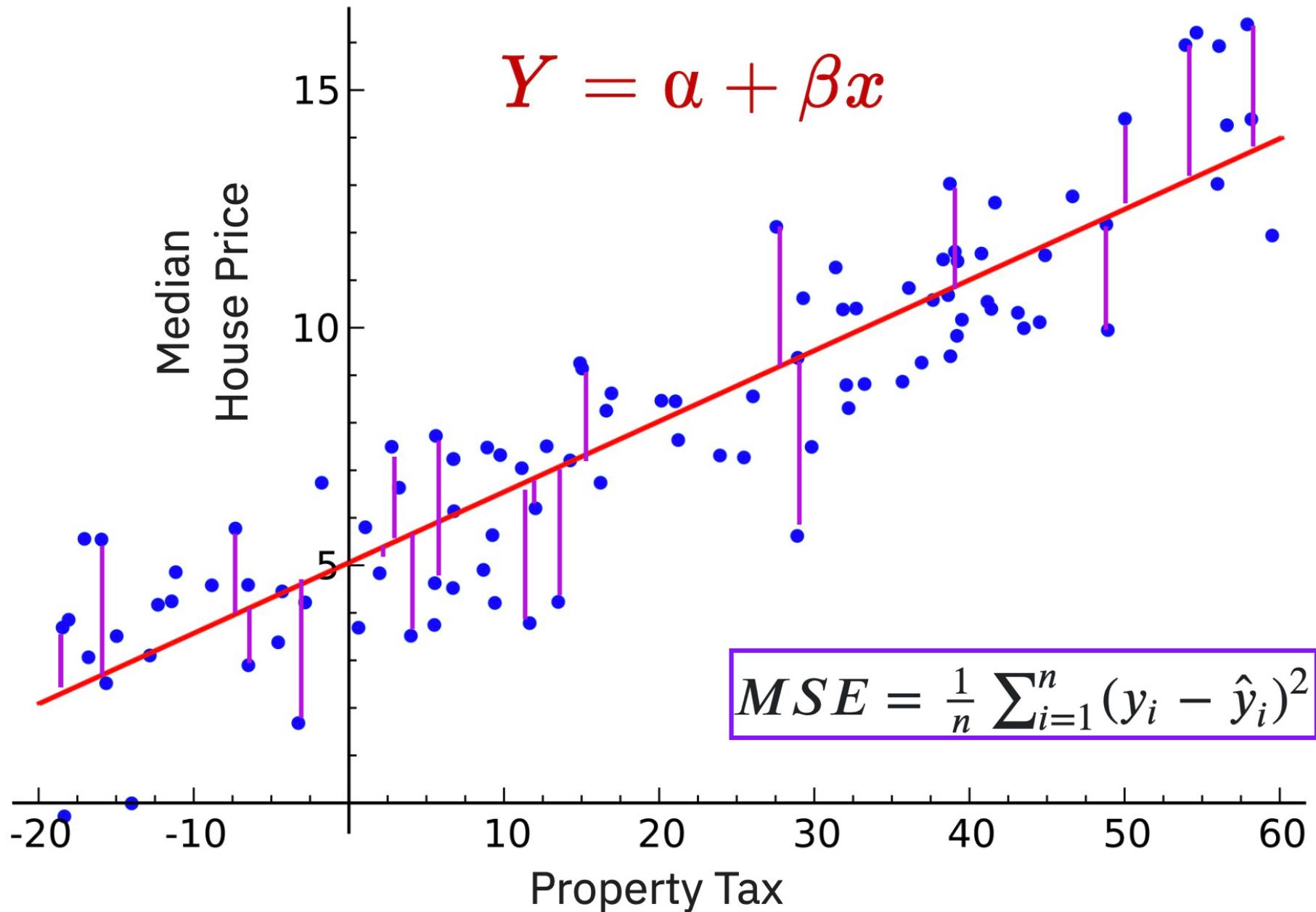
	0	1	2	3	4
CRIM	0.00632	0.02731	0.02729	0.03237	0.06905
ZN	18.00000	0.00000	0.00000	0.00000	0.00000
INDUS	2.31000	7.07000	7.07000	2.18000	2.18000
CHAS	0.00000	0.00000	0.00000	0.00000	0.00000
NOX	0.53800	0.46900	0.46900	0.45800	0.45800
RM	6.57500	6.42100	7.18500	6.99800	7.14700
AGE	65.20000	78.90000	61.10000	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

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

Linear regression - try to fit a line



Linear regression - loss function to get best fit



Steps

1. Sign up / Log into *IBM Cloud* - <http://bit.ly/ibm-cloud-summit-2019>
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. Upload csv *data* to your project.
5. Add a new *AutoAI Experiment* to your project.
6. Create a *ML Model* and *save* it to IBM Cloud.
7. Create a new *deployment* on IBM Cloud.
8. *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](#)

Step 2 - locate Watson Studio in Catalog

The screenshot shows the IBM Cloud Catalog interface. At the top, there is a navigation bar with links for IBM Cloud, Catalog (which is highlighted with a pink border), Docs, Support, Manage, and Upkar6 Lidder6's Account. Below the navigation bar, the word "Catalog" is displayed in large letters. A search bar with the placeholder "Search the catalog..." and a "Filter" button are also present.

On the left side, there is a sidebar titled "All Categories" containing a list of service categories: Compute, Containers, Networking, Storage, AI (which is highlighted with a pink border), Analytics, Databases, Developer Tools, Integration, Internet of Things, Security and Identity, Starter Kits, Web and Mobile, and Web and Application.

The main content area is titled "AI" and contains several service cards:

- Watson Assistant**: Lite • IBM • IAM-enabled. Description: Watson Assistant lets you build conversational interfaces into any application, device, or channel.
- Watson Studio**: Lite • IBM • IAM-enabled. Description: Embed AI and machine learning into your business. Create custom models using your own data. This card is highlighted with a pink border.
- Compare and Comply**: Lite • IBM • IAM-enabled. Description: Process governing documents to convert, identify, classify, and compare important elements.
- Discovery**: Lite • IBM • IAM-enabled. Description: Add a cognitive search and content analytics engine to applications.
- Knowledge Catalog**: Lite • IBM • IAM-enabled. Description: Discover, catalog, and securely share enterprise data.
- Knowledge Studio**: Lite • IBM • IAM-enabled. Description: Teach Watson the language of your domain.

A "FEEDBACK" button is located on the right side of the page. At the bottom right, there is a blue circular icon with a white speech bubble symbol.

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. Below the navigation, a sidebar for 'Watson Studio' is visible, showing its description as a 'Lite + IBM' service. The main form for creating a new instance has the following fields:

- Service name:** Watson Studio-xyz (highlighted with a pink box)
- Choose a region/location to deploy in:** Dallas (dropdown menu)
- Select a resource group:** Default
- Tags:** env:dev, version-1

On the right side of the form, there is a 'FEEDBACK' button and a blue circular icon with a white speech bubble.

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.

At the bottom, there are buttons for 'Add to estimate' and 'Create' (highlighted with a pink box). A 'Need Help?' link and a 'Contact IBM Cloud Support' button are also present.

Step 3 - already have Watson Studio? Find it in Resources

The screenshot shows the IBM Cloud Resource List interface. The left sidebar lists various service categories: Dashboard, Resource List (which is selected and highlighted with a pink border), Cloud Foundry, Kubernetes, VPC Infrastructure (marked as New), Classic Infrastructure, VMware, API Management, Apple Development, Blockchain, DevOps, Finance, Functions, Integrate, Managed Solutions, and Mobile. The main content area displays a table titled 'Resource List' with columns for Name, Group, and Location. The table includes filters for Name, Group, and Location. The data in the table is as follows:

Name	Group	Location
> Devices (0)		
> VPC Infrastructure (0)		
> Kubernetes Clusters (0)		
> Cloud Foundry Apps (0)		
> Cloud Foundry Services (0)		
Services (2)		
Watson Studio-de	Default	Dallas
pm-20-dsx	Default	Dallas
Storage (1)		
cloud-object-storage-dsx	Default	Global
> Cloud Foundry Enterprise Environments (0)		
> Functions Namespaces (0)		
> Apps (0)		

The Watson Studio resource entry ('Watson Studio-de') is highlighted with a pink border.

Step 4 - launch Watson Studio

IBM Cloud Catalog Docs Support Manage Upkar6 Lidder6's Account  

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 image shows the IBM Watson Studio interface. At the top, there is a dark header bar with the text "IBM Watson Studio" on the left, a bell icon, the account name "Upkar6 Lidder6's Account" in the middle, and a user 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 abstract shapes. It features the text "Welcome Upkar6!" in large white font, followed by "Watson Studio • Watson Knowledge Catalog". Below this, there is a section titled "Start by creating a project" with the subtext "A project is how you organize your resources to work with data and collaborate with team members". There are two call-to-action boxes: one labeled "Create a project" with the subtext "Create a project, then add the tools and assets you need.", and another labeled "Search a catalog" with the subtext "Find the assets you need in a catalog.". To the right of these boxes is a circular graphic containing a line-art illustration of a computer monitor displaying binary code, a magnifying glass over a document, and a small plant.

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.

Step 6 - pick Data Science starter

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 and AutoAI**
Analyze and model data to discover insights or generate predictions.
ASSETS
Data • Notebooks • AutoAI Experiments
[Create Project](#)

 **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

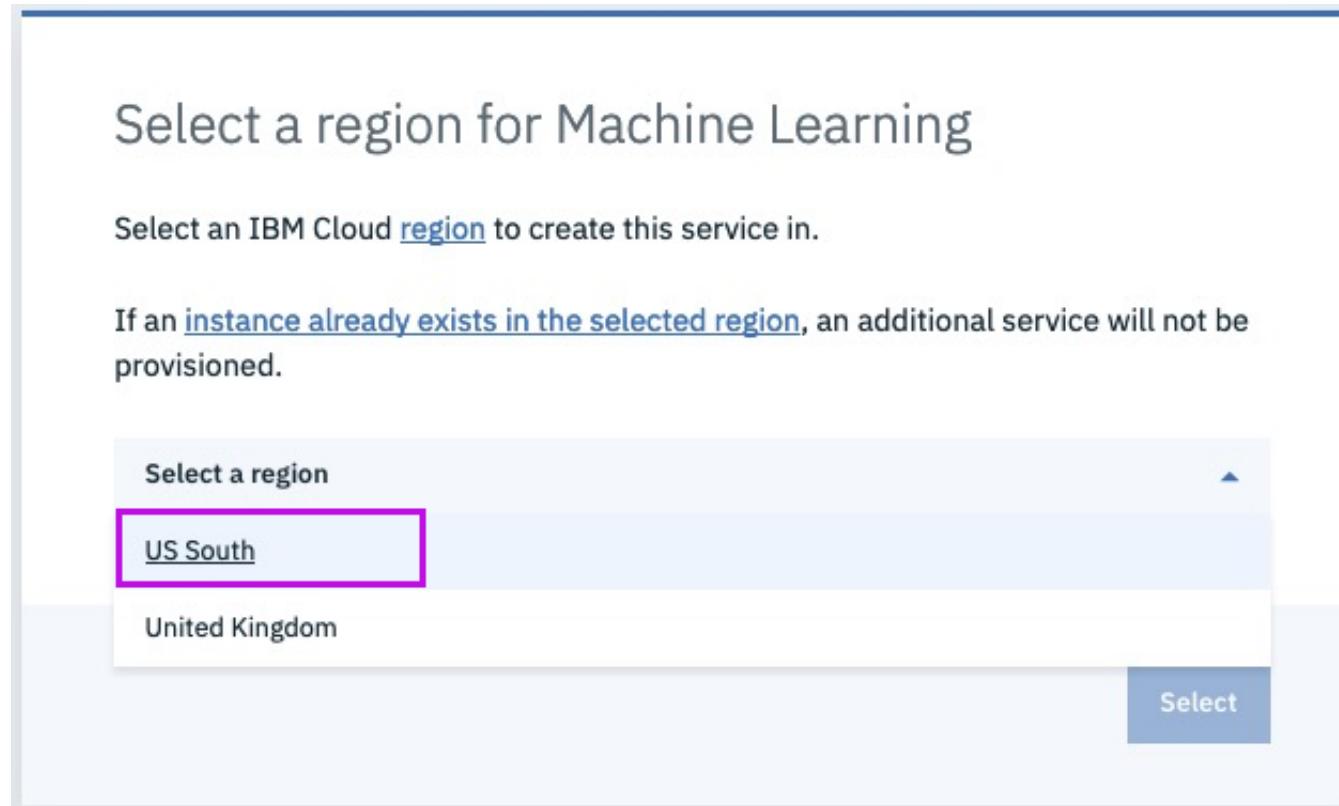
 **Modeler**
Build modeler flows to train SPSS models or design deep neural networks.
ASSETS
Data • Modeler Flow • Model • Experiment

 **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



Step 6a - pick region [US South]



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, this is your goto page!

The screenshot shows the IBM Watson Studio interface with the 'Assets' tab selected (highlighted with a pink box). The top navigation bar includes 'Upgrade', 'Upkar Lidder's Account', and a user icon. Below the bar, the breadcrumb path is 'My Projects / My Data Science Project'. The main content area displays sections for 'Data assets', 'AutoAI experiments', and 'Experiments', each with a table header and a message indicating no items are present.

IBM Watson Studio

Upgrade Upkar Lidder's Account

My Projects / My Data Science Project

Assets

Overview Environments Bookmarks Deployments Access Control Settings

Add to project

Load Files Catalog

What assets are you looking for?

Data assets

NAME	TYPE	CREATED BY	LAST MODIFIED	ACTIONS
You don't have any Data assets yet.				

New AutoAI experiment +

AutoAI experiments

NAME	STATUS	MODEL TYPE	LAST MODIFIED	ACTIONS
You don't have any AutoAI experiments yet.				

New experiment +

Experiments

NAME	LAST MODIFIED	ACTIONS
You don't have any Experiments yet.		

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

Step 9 - create a new AutoAI experiment

Create AutoAI experiment

Use this no-code approach to generate a prediction based on data you provide. AutoAI automatically finds the best algorithm for your data and use case. [Learn more](#)

Define AutoAI details

Create AutoAI experiment type

From blank From sample

Asset name *

boston-housing-autoai

Description

Description of AutoAI experiment

Associated services

Watson Machine Learning Service Instance *

pm-20-dsx

Compute configuration * ⓘ

8 vCPU and 32 GB RAM

This compute configuration consumes **20 capacity units per hour**. [Learn more](#) about capacity unit hours and Watson Machine Learning pricing plans.

Cancel Create

Step 10 - adding training data

My Projects / My Data Science Project / boston-housing-autoai

CONFIGURE AUTOAI EXPERIMENT
boston-housing-autoai |

Add training data

Drop a .csv file here or [browse](#) for a file to upload.

[Select from project](#)

Name Size Date Modified Kind
ai-ml-watson.pdf 24.5 MB May 22, 2019 at 3:47 PM PDF Document
housing-regression.ipynb 7 KB May 21, 2019 at 10:04 PM Document
boston_house_prices.csv 35 KB May 20, 2019 at 10:15 AM CSV Document

Options Cancel Open

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

Step 11 - pick target column to predict [MEDV]

My Projects / My Data Science Project / boston-housing-autoai



CONFIGURE AUTOAI EXPERIMENT boston-housing-autoai |

Add training data



Drop a .csv file here or [browse](#) for a file to upload.

[Select from project](#)

Data source

NAME	SIZE (MB)	COLUMNS	X
boston_house_prices.csv	0.035	14	X

Select column to predict

DATA SOURCE: boston_house_prices.csv

Column name	Type	
LSTAT	Decimal	
MEDV	Decimal	<input checked="" type="checkbox"/>

Selected prediction

[Edit prediction](#)

PREDICTION TYPE

Regression

OPTIMIZED METRIC

RMSE

[Save and close](#)

[Run experiment](#)



Step 11a - change model and metric if needed

My Projects / My Data Science Project / boston-housing-autoai

CONFIGURE AUTOAI EXPERIMENT
boston-housing-autoai

Add training data

Data source
NAME
boston_house_prices.csv

Edit prediction

Type
Decimal

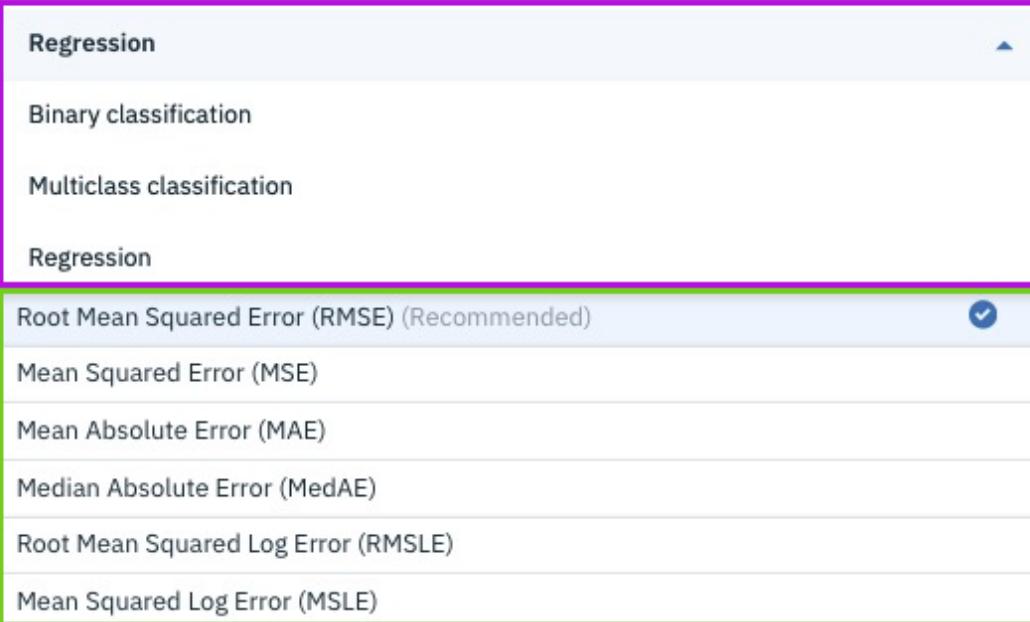
Regression

- Binary classification
- Multiclass classification
- Regression
 - Root Mean Squared Error (RMSE) (Recommended)
 - Mean Squared Error (MSE)
 - Mean Absolute Error (MAE)
 - Median Absolute Error (MedAE)
 - Root Mean Squared Log Error (RMSLE)
 - Mean Squared Log Error (MSLE)

Cancel **Apply**

Save and close Run experiment ▶

OPTIMIZED METRIC
RMSE ⓘ



Step 12 - run experiment

My Projects / My Data Science Project / boston-housing-autoai

CONFIGURE AUTOAI EXPERIMENT
boston-housing-autoai |

Add training data

Drop a .csv file here or [browse](#) for a file to upload.

Select from project

Data source

NAME	SIZE (MB)	COLUMNS	X
boston_house_prices.csv	0.035	14	X

Select column to predict

DATA SOURCE: boston_house_prices.csv

Column name	Type	
LSTAT	Decimal	
MEDV	Decimal	<input checked="" type="checkbox"/>

Selected prediction

PREDICTION TYPE: Regression Edit prediction

OPTIMIZED METRIC: RMSE

Save and close Run experiment

Step 13a - sit back and relax!

My Projects / My Data Science Project / boston-housing-autoai

boston-housing-autoai SOURCE TABLE: boston_house_prices.csv

PREDICTION COLUMN MEDV PREDICTION TYPE Regression OPTIMIZED METRIC RMSE

Submitted
The run will begin momentarily **Status**

0 seconds

Read dataset Split holdout data Read training data Preprocessing Model selection Selected estimator Hyperparameter optimization Feature engineering Hyperparameter optimization P1 P2 P3 P4

Live Report

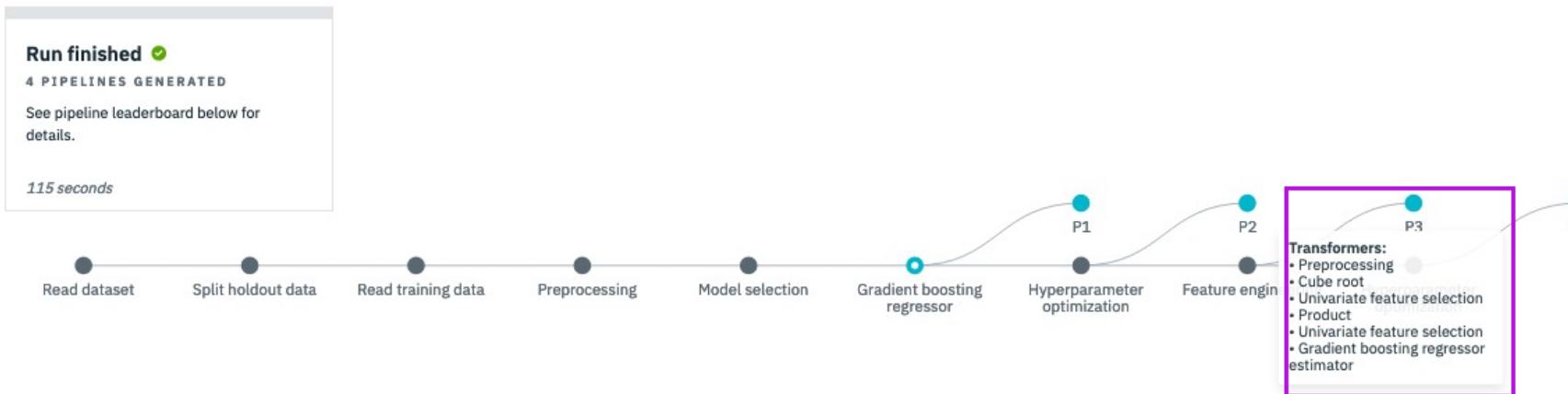
Pipeline leaderboard

Compare pipelines Ranking based on: Root Mean Squared Error (RMSE)

RANK	PIPELINE	RMSE	ESTIMATOR	ENHANCEMENTS
...

Pipelines

Step 13b - explore different models in the pipeline



Pipeline leaderboard

Compare pipelines Ranking based on: Root Mean Squared Error (RMSE) ▾

RANK	PIPELINE	RMSE	ESTIMATOR	ENHANCEMENTS	
1	Pipeline 3	3.009	Gradient boosting regressor	HPO FE	Save as model

Model evaluation measures

	Training score	Holdout score
Explained Variance	0.895	0.860
MAE	2.190	2.040
MSE	9.059	8.318
MSLE	0.021	0.023
MedAE	1.619	1.428

Feature Importance ⓘ

Feature	Importance
RM	0.28
NewFeature_1...	0.21
NewFeature_7...	0.11
NewFeature_1...	0.11
ptc	0.07

Step 14 - save the best model

Run finished ✓
4 PIPELINES GENERATED
See pipeline leaderboard below for details.

115 seconds

Read dataset Split holdout data Read training data Preprocess

Save as model

boston-housing-autoai - P3 GradientBoostingRegressorEstimator

Description (optional)

Associated project

My Data Science Project

Watson Machine Learning service

pm-20-dsx

based on: Root Mean Squared Error (RMSE)

Cancel Save

MENTS

FE Save as model

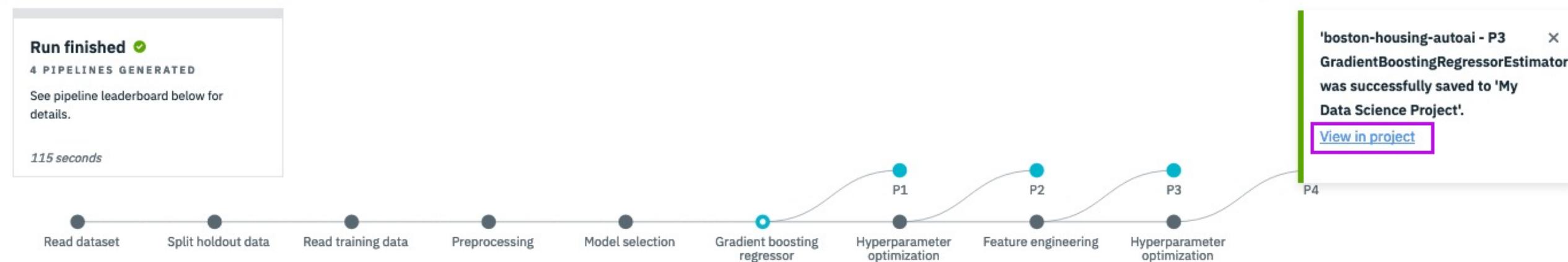
FE HPO + FE Save as model

None Save as model

HPO Save as model

RANK	Pipeline
> 1	Pipeline 3
> 2	Pipeline 4
> 3	Pipeline 1
> 4	Pipeline 2

Step 15a - view the model



Step 15b - view the model, another way

My Projects / My Data Science Project

Launch IDE Add to project

Overview Assets Environments Bookmarks Deployments Access Control Settings

What assets are you looking for?

Data assets

0 asset selected.

<input type="checkbox"/>	NAME	TYPE	CREATED BY	LAST MODIFIED	ACTIONS
<input type="checkbox"/>	CSV boston_house_prices.csv	Data Asset	Upkar5 Lidder5	27 Jun 2019, 1:27:12 pm	

AutoAI experiments

New AutoAI experiment +

NAME	STATUS	MODEL TYPE	LAST MODIFIED	ACTIONS
boston-housing-autoai	Completed	Regression	27 Jun 2019, 1:49:41 pm	

Models

Watson Machine Learning models

New Watson Machine Learning model +

NAME	STATUS	TYPE	RUNTIME	LAST MODIFIED	ACTIONS
boston-housing-autoai - P3 GradientBoostingRegressorEstimator	trained	wml-hybrid_0.1	hybrid_0.1	27 Jun 2019	

IBM Developer

@lidderupk

Step 16 - add a new deployment

The screenshot shows the IBM Watson Studio interface. On the left, there's a navigation bar with 'My Projects' / 'My Data Science Project' / 'my_machine_learning'. Below it, the 'MODEL' section for 'my_machine_learning' is shown, with tabs for 'Overview', 'Evaluation', and 'Deployments' (which is highlighted with a pink box). A message says 'Your model is not deployed.' On the right, a 'Create Deployment' dialog box is open, also highlighted with a pink box. It contains fields for 'Name' (set to 'boston-housing-regression'), 'Description' (empty), 'Deployment type' (set to 'Web service'), and buttons for 'Cancel' and 'Save' (the latter is also highlighted with a pink box).

My Projects / My Data Science Project / my_machine_learning

MODEL

my_machine_learning Delete

Overview Evaluation Deployments

NAME STATUS DE

Your model is not deployed.

Create Deployment

Define deployment details

Name

Description

Deployment type

Web service

Cancel Save

Step 17 - ensure that deployment is successful with ready status

1

This screenshot shows the 'Deployments' tab for the 'my_machine_learning' model. A single deployment named 'boston-housing-linear-regression' is listed. The status of this deployment is 'INITIALIZING'. The deployment type is 'Web Service'. There is a button labeled 'Add Deployment' with a plus sign.

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

2

This screenshot shows the same 'Deployments' tab for the 'my_machine_learning' model. The deployment named 'boston-housing-linear-regression' now has a status of 'DEPLOY_SUCCESS'. A pink arrow points from the previous 'INITIALIZING' status to the new 'DEPLOY_SUCCESS' status.

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

Step 18a - 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 18b - implementation / test the deployed model

```
1 {"input_data":{  
2     "fields": ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"],  
3     "values": [[0.00632,18,2.31,0,0.538,6.575,65.2,4.09,1,296,15.3,396.9,4.98]]  
4 }}}
```

boston-housing-regression

The screenshot shows a user interface for a machine learning model named 'boston-housing-regression'. At the top, there are three tabs: 'Overview', 'Implementation', and 'Test', with 'Test' being the active tab. Below the tabs, the title 'Enter input data' is displayed. A code snippet representing input data is pasted into a text area, which is highlighted with a magenta border. The code is identical to the one shown above. To the right of the input area, a JSON response is shown, also enclosed in a magenta border. This response contains a 'predictions' array with one object. The 'prediction' field is highlighted with a green border and contains the value '25.77754622481215'. At the bottom left, a blue 'Predict' button is visible.

```
{"input_data":{  
    "fields": ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"],  
    "values": [[0.00632,18,2.31,0,0.538,6.575,65.2,4.09,1,296,15.3,396.9,4.98]]}}
```

```
{  
    "predictions": [  
        {  
            "fields": [  
                "prediction"  
            ],  
            "values": [  
                [  
                    25.77754622481215  
                ]  
            ]  
        }  
    ]  
}
```

Predict

AutoAI - Behind the scenes

Data pre-processing

- analyze, clean and prepare raw data for ML
- automatically detects and categorizes features based on data type
- missing value imputation
- feature encoding
- feature scaling

Automated model selection

- test and rank candidate estimators
- select the best performing estimator with the ranking choice made by the user

Automated Feature Engineering

- transform raw data into combination of features that best fit the model

Hyperparameter Optimization

- refine the best performing model

AutoAI - Supported Estimators

<http://bit.ly/autoai-estimators>

Classification
AdaBoost Classifier
Bernoulli Naïve Bayes Classifier
Calibrated Classifier with Cross-Validation
Decision Tree Classifier
Extra Trees Classifier
Gaussian Naïve Bayes Classifier
Gaussian Process Classifier
Gradient Boosted Tree Classifier
Nearest Neighbor Analysis (KNN) Classifier
Label Propagation
Label Spreading
LGBM Classifier
Linear Discriminant Analysis
Linear Support Vector Classifier
Logistic Regression with Cross-Validation
Logistic Regression
MLP Classifier
Multinomial Naïve Bayes Classifier
Nearest Centroid
Nu Support Vector Classifier
Passive Aggressive Classifier
Perceptron
Quadratic Discriminant Analysis
Radius Neighbors Classifier
Random Forest Classifier
Ridge Classifier with Cross-Validation
Ridge Classifier
SGD Classifier
Support Vector Classifier
XGBoost Classifier

Regression
AdaBoost Regression
ARD Regression
Bayesian Ridge Regression
CCA
Decision Tree Regression
Extra Trees Regression
Elastic Net with Cross-Validation
Elastic Net
Gaussian Process
Gaussian Process Regression
Gradient Boosting Regression
Huber Regression
Nearest Neighbor Analysis (KNN)
Kernel Ridge
Lars with Cross-Validation
Lars
Lasso with Cross-Validation
Lasso
Lasso Lars with Cross-Validation
Lasso Lars
Lasso Lars IC
LGBM Regression
Linear Regression
Linear Support Vector Regression
MLP Regression
MultiTask Elastic Net CV
MultiTask Elastic Net
Multi Task Lasso CV
Multi Task Lasso
Nu SVR

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

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

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

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

Deploy & Manage

Manage models and versions

Automate ML

Your models always learning with a feedback loop

- 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

- 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 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

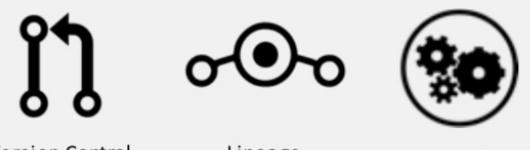
Operationalize



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

AutoAI

Name ▾	Root Mean Squared Error (RMSE) (optimized)	Explained Variance	Mean Absolute Error (MAE)	Mean Squared Error (MSE)	Mean Squared Log Error (MSLE)	Median Absolute Error (MedAE)	Root Mean Squared Log Error (RMSLE)	R ²	Created At
P1	3.261	0.876	2.222	10.657	0.022	1.607	0.148	0.875	6/27/2019, 3:29:38 PM
P2	3.261	0.876	2.222	10.657	0.022	1.607	0.148	0.875	6/27/2019, 3:29:38 PM
P3	3.017	0.894	2.137	9.122	0.021	1.600	0.144	0.894	6/27/2019, 3:30:39 PM
P4	3.017	0.894	2.137	9.122	0.021	1.600	0.144	0.894	6/27/2019, 3:30:59 PM

Notebook

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

The screenshot shows the IBM Watson Studio interface. The left sidebar has a dark header with an 'X' icon and the text 'IBM Watson Studio'. Below it are sections for 'Home', 'Projects' (with 'View All Projects' and 'My Data Science Project'), 'Services' (with 'Watson Services' highlighted by a purple border), 'Compute Services', 'Community', 'Manage', and 'Support'. The main content area has a light header with 'Services / Watson Services', a search bar, and buttons for 'Add service', 'Info', 'Copy', 'Share', 'Grid View', and 'Map View'. It shows filter options for 'RESOURCE GROUP' (All Resources), 'LOCATION' (Locations 2), and 'CLOUD FOUNDRY ORG' (upkar.ibm.watson.5@gmail.com). A section titled 'Machine Learning (1)' lists a single service with the name 'pm-20-dsx' highlighted by a purple border. The table columns are 'Name', 'Tool', and 'Actions'.

Name	Tool	Actions
pm-20-dsx		

WML - get Machine Learning service credentials

Manage /
Service credentials
Plan
Connections

Resource list /
 pm-20-wp
Resource group: Default Location: Dallas [Add Tags](#) ⋮

Service credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service. [Learn more](#)

Service credentials [New credential +](#) ⋮

Items per page **10** | 1-1 of 1 items 1 of 1 pages < 1 >

<input type="checkbox"/> KEY NAME	DATE CREATED	ACTIONS
<input type="checkbox"/> wdp-writer	MAY 21, 2019 - 02:10:45 PM	View credentials ▾ trash

```
{  
  "apikey": "████████████████████████████████████████",  
  "iam_apikey_description": "████████████████████████████████████████",  
  "iam_apikey_name": "wdp-writer",  
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",  
  "iam_serviceid_crn": "████████████████████████████████████████████████████████",  
  "instance_id": "████████████████████████████████████████████████████████",  
  "password": "████████████████████████████████████████████████████████",  
  "url": "https://us-south.ml.cloud.ibm.com",  
  "username": "████████████████████████████████████████████████████████"}  
}
```

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 ! Step1 - Sign into IBM Cloud

The screenshot shows the IBM Watson Studio interface. At the top, there's a navigation bar with 'IBM Watson Studio', a user account dropdown for 'Upkar6 Lidder6's Account', and a 'UL' button. Below the navigation bar, the current project is 'My Data Science Project', indicated by a pink box around the tab. A prominent blue button labeled '+ Add to project' is also highlighted with a pink box. 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 'Choose asset type' lists 'AVAILABLE ASSET TYPES' in a grid. One item, 'Notebook', is highlighted with a pink box. Other asset types include Data, Connection, Connected data, Dashboard, Image classification ..., Object detection mo..., Natural Language Cl..., Watson Machine Lea..., Experiment, Modeler flow, Data Refinery flow, Streams flow, and Synthesized neural n...'. On the left, there are collapsed sections for 'Data assets', 'Model assets', 'Watson assets' (with a 'NAME' field containing 'my_machine'), and 'Experiment assets' (with a 'NAME' field). A 'Close' button is located at the bottom right of the dialog.

Step2 - create a new notebook from URL

The screenshot shows the 'Add Notebook' page in IBM Watson Studio. 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 title is 'New notebook'. There are 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>

Step3 - paste your WML credentials and run each cell!

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** My Projects / testttt-upkar / try
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Help, Run (highlighted with a purple box)
- Cell Type:** Trusted | Python 3.5
- Section:** 1. Load the data
- Code Block (In [2]):**

```
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_boston
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

boston = load_boston()
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target)

# Create a new Linear Regression Model
LR_model = LinearRegression()

# Train the model
LR_model.fit(X_train, y_train)

# store actual and predicted data to draw chart
predicted = LR_model.predict(X_test)
actual = y_test

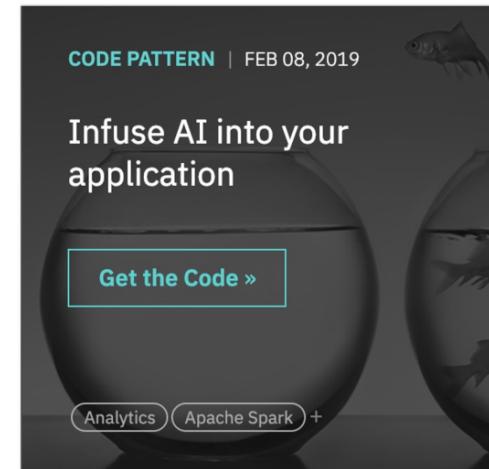
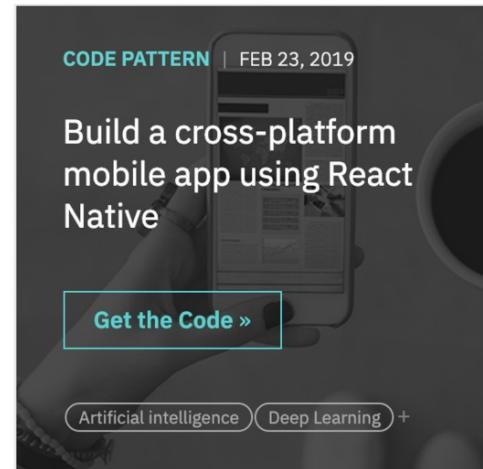
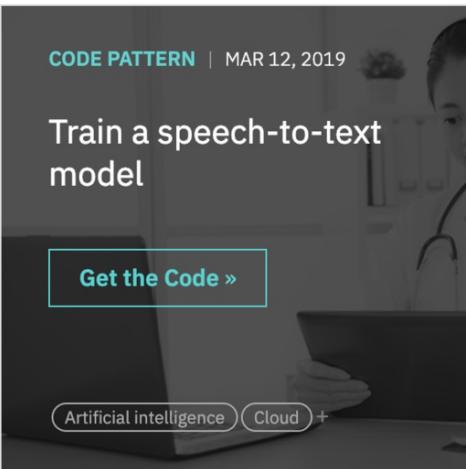
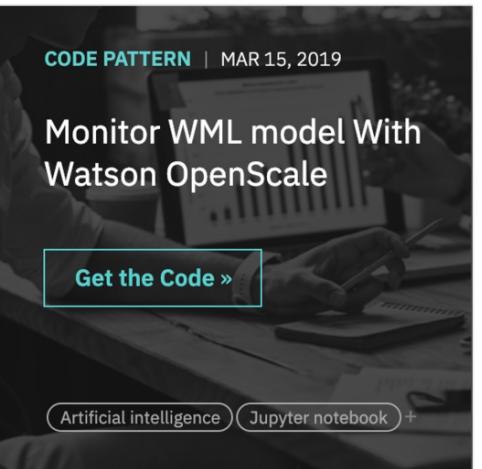
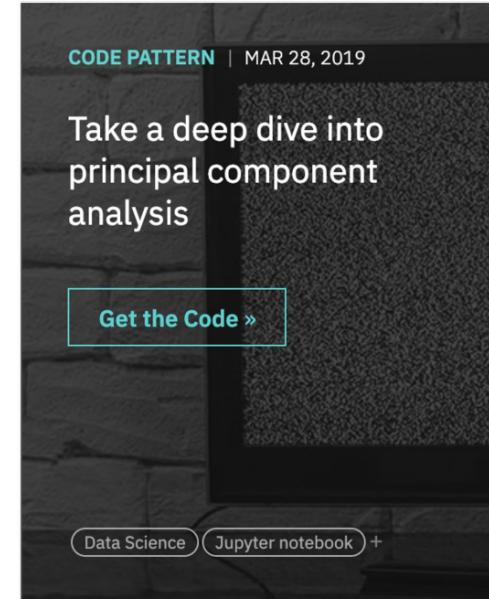
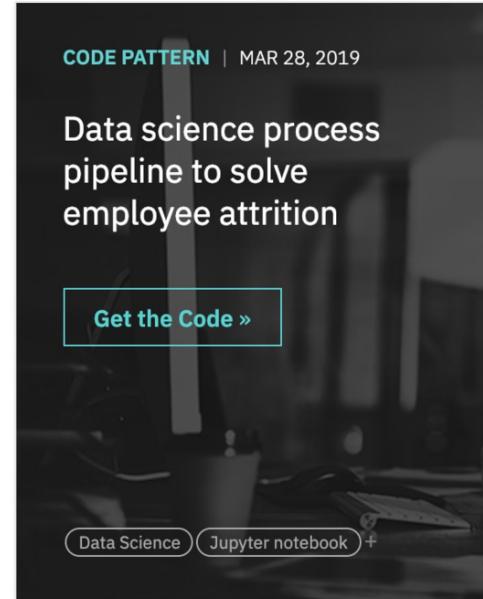
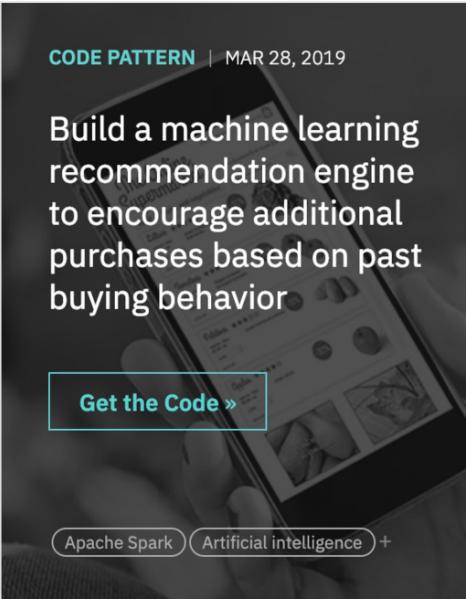
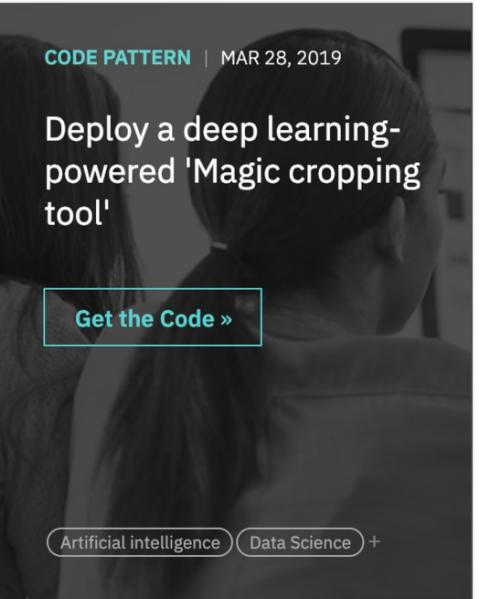
plt.figure(figsize=(4, 3))
plt.scatter(actual, predicted)
plt.plot([0, 50], [0, 50], '--k')
plt.axis('tight')
plt.xlabel('True Median Value ($1000s)')
plt.ylabel('Predicted Median Value ($1000s)')
plt.tight_layout()

# The coefficients
print('Coefficients: \n', LR_model.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(actual, predicted))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(actual, predicted))
```
- Feedback:** A blue speech bubble icon is located in the bottom right corner of the code block.

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

IBM Code Patterns

Machine Learning 



Thank you

Let's chat !

Upkar Lidder, IBM

@lidderupk

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

ulidder@us.ibm.com



@lidderupk

IBM Developer


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](#) [↗](#)

WED, AUG 21, 9:30 AM

Online Meetup: Let's explore some data using Spark, PixieDust, and more!

Let's explore some data using Spark, PixieDust, and more!

Wednesday, August 21st, 2019 | 9:30 AM - 10:30 AM PT
[Crowdcast]

Needs a location

RSVP on Crowdcast NOW! <https://www.crowdcast.io/e/8u3d6q3c> ! Please do not forget to register on Crowdcast and join us using Chrome browser via Crowdcast on the event date! Spark and PixieDust! What cool names! They also happen to be two of my favorite open source tools t...

40 attendees Manage Going

THU, AUG 22, 12:00 PM

Lunch & Learn: A Tale of Two Disciplines-How Designers & Developers Co-Create

A Tale of Two Disciplines- How Designers & Developers Co-Create

Thursday, August 22nd, 2019 | 12 PM - 2 PM PT
[Gathering - 44 Tehama St (SF MEETUP)]

44 Tehama St

Two worlds collide – Design and Development. Will they destroy each other? Or will they peacefully co-exist and thrive, creating a beautiful new world! Join IBM Developer SF's lunch and learn to discover how IBM Cloud Garage designers and developers collaborate and co-create...

66 attendees Manage Attend

WED, AUG 28, 9:30 AM

Online Meetup: Build serverless APIs using Postman and Apache OpenWhisk

Build serverless APIs using Postman and Apache OpenWhisk

Wednesday, August 28, 2019 | 9:30 AM - 10:30 AM PT
[Crowdcast]

Needs a location

RSVP on Crowdcast NOW! <https://www.crowdcast.io/e/lvaejgjs> ! Please do not forget to register on Crowdcast and join us using Chrome browser via Crowdcast on the event date! IBM and Postman are partnering up to show you how to use open source tools to create and test scalab...

36 attendees Manage Going

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.

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



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.