

LAB MANUAL

INDEX

S.NO	DATE	NAME OF THE EXPERIMENT	PAGE NO.	STAFF SIGNATURE
1.	19.12.22	CREATE A LIST IN R		
2.	26.12.22	CREATE A DATAFRAME IN R		
3.	2.1.23	CREATE PROJECT IN WATSON STUDIO		
4.	9.1.23	DATA REFINERY-PREPARING THE DATA		
5.	23.1.23	VISUALIZATION DATA IN WATSON STUDIO		
6.	30.1.23	NATURAL LANGUAGE PROCESSING USING WATSON STUDIO		
7.	13.2.23	LINNER REGRESSION IN R		
8.	20.2.23	DECISION TREE IN R PROGRAMMING		

1. CREATE A LIST IN R

Aim :

To create the program for List by using R programming

Procedure :

Step 1 : Open R studio – New- New script and R programming work area will get created with console and environment .

Step 2 : Create list by declaring variable in vector

Step 3: By using list () , data will get created inside List .

Step 4 : Select the element from the list by index value .

Step 5 : To find the length use Len() and more list function to atchive List operations.

Step 6: To merge two list we use merge.List

Step 7: Save the program

Step 8 : Select the entire code and click run .

Program:

```
# Creating list  
a<-1:5  
b<-c("A","b","c","d","e")  
c<-3+2i  
l<-list(a,b,c)  
l  
l1<-list("red", 1, c(2,3,4),5.5)  
l1  
#selecting Elements from list  
l[[2]]  
l[[2]][4]  
l[[c(1,2)]]  
l<-list(f=a,s=b,t=c)  
l  
class(l)  
lapply(l, length)  
mode(l)  
str(l)  
list1 <- list(1,2,3)  
list2 <- list("S","u","n")  
merged.list<- c(list1,list2)  
print(merged.list)  
list1
```

Output:

```
Console Terminal × Jobs ×
R 4.1.3 · ~/ ↗
> a<-1:5
> b<-c("A","b","c","d","e")
> c<-3+2i
> l<-list(a,b,c)
> l
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "A" "b" "c" "d" "e"

[[3]]
[1] 3+2i

> l1<-list("red", 1, c(2,3,4),5.5)
> l1
[[1]]
[1] "red"

[[2]]
[1] 1

[[3]]
[1] 2 3 4

[[4]]
[1] 5.5
```

```
Console Terminal × Jobs ×
R 4.1.3 · ~/ ↗
> l[[2]]
[1] "A" "b" "c" "d" "e"
> l[[2]][4]
[1] "d"
> l[[c(1,2)]]
[1] 2
> l<-list(f=a,s=b,t=c)
> l
$f
[1] 1 2 3 4 5

$s
[1] "A" "b" "c" "d" "e"

$t
[1] 3+2i

> l$t
[1] 3+2i
> class(l)
[1] "list"
> lapply(l, length)
$f
[1] 5

$s
[1] 5
```

```
Console Terminal x Jobs x
R 4.1.3 · ~/ ↗
$ S. CII [1.0] A u c u ...
$ t: cplx 3+2i
> list1 <- list(1,2,3)
> list2 <- list("s","u","n")
> merged.list <- c(list1,list2)
> print(merged.list)
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] "s"

[[5]]
[1] "u"

[[6]]
[1] "n"

> list1
[[1]]
[1] 1
```

Result :

Thus, the above program was successfully executed.

2.CREATE A DATAFRAME IN R

Aim :

To create the program for Data frame by using R programming

Procedure :

Step 1 : Open R studio – New- New script and R programming work area will get created with console and environment .

Step 2 : Create vector by c(), pass the value of same datatype

Step 3: By using data. frame() , can convert the List to Dataframe with rows and column .

Step 4 : To find the mid value use the index [] value that comes in column .

Step 5 : rbind() , we can add the row of the dataframe

Step 6: Adding and sorting operations can also be done

Step 7: Save the program

Step 8 : Select the entire code and click run .

Program code:

```
#Creating dataframe  
name<-c("ram","sam","tom")  
age<-c(18,17,16)  
mark<-c(430,450,356)  
df<-data.frame(name,age,mark)  
df  
  
#naming  
names(df)<-c("Names","Age","CGPA")  
df  
df<-data.frame(Name=name,Age=age,CGPA=mark)  
df  
  
#Selection  
str(df)  
df[3,2]  
df[3]  
df["Age"]  
df[c(1,3),c("Age","CGPA")]  
df$Name  
df[[2]]  
  
# ADDING values  
  
#adding column  
dep<-c("BCA","CS","IT")  
df$dep<-dep  
df  
df[["dep"]]<-dep
```

```
df
#adding Row
zam<-data.frame(Name="zam",Age=19,CGPA=415,dep="MBA")
rbind(df,zam)
df
zam
#sorting
sort(df$Age)
ranks<-order(df$Age)
ranks
df$Age
df[ranks,]
```

Output:

```
Console Terminal × Jobs ×
R 4.1.3 · ~/ ↗
> #Creating dataframe
> name<-c("ram","sam","tom")
> age<-c(18,17,16)
> mark<-c(430,450,356)
> df<-data.frame(name,age,mark)
> df
  name age mark
1 ram 18 430
2 sam 17 450
3 tom 16 356
> #naming
> names(df)<-c("Names", "Age", "CGPA")
> df
  Names Age CGPA
1 ram 18 430
2 sam 17 450
3 tom 16 356
> df<-data.frame(Name=name, Age=age, CGPA=mark)
> df
  Name Age CGPA
1 ram 18 430
2 sam 17 450
3 tom 16 356
```

```
Console Terminal × Jobs ×
R 4.1.3 · ~/ ↗
> #Selection
> str(df)
'data.frame': 3 obs. of 3 variables:
 $ Name: chr "ram" "sam" "tom"
 $ Age : num 18 17 16
 $ CGPA: num 430 450 356
> df[3,2]
[1] 16
> df[3]
  CGPA
1 430
2 450
3 356
> df["Age"]
  Age
1 18
2 17
3 16
> df[c(1,3),c("Age", "CGPA")]
  Age CGPA
1 18 430
3 16 356
> df$name
[1] "ram" "sam" "tom"
> df[[2]]
[1] 18 17 16
```

```
Console Terminal x Jobs x
R 4.1.3 · ~/ ↗
> # ADDING values
> #adding column
> dep<-c("BCA","cs","IT")
> df$dep<-dep
> df
  Name Age CGPA dep
1 ram 18 430 BCA
2 sam 17 450 CS
3 tom 16 356 IT
> df[["dep"]]<-dep
> df
  Name Age CGPA dep
1 ram 18 430 BCA
2 sam 17 450 CS
3 tom 16 356 IT
> #adding Row
> zam<-data.frame(Name="zam",Age=19,CGPA=415,dep="MBA")
> rbind(df,zam)
  Name Age CGPA dep
1 ram 18 430 BCA
2 sam 17 450 CS
3 tom 16 356 IT
4 zam 19 415 MBA
> df
  Name Age CGPA dep
1 ram 18 430 BCA
2 sam 17 450 CS
3 tom 16 356 IT
> zam
  Name Age CGPA dep
1 zam 19 415 MBA
> #sorting
> sort(df$Age)
[1] 16 17 18
> ranks<-order(df$Age)
> ranks
[1] 3 2 1
> df$Age
[1] 18 17 16
> df[ranks,]
  Name Age CGPA dep
3 tom 16 356 IT
2 sam 17 450 CS
1 ram 18 430 BCA
> |
```

Result :

Thus, the above program was successfully executed

3.CREATE PROJECT IN WATSON STUDIO

Procedure:

Step 1: Open IBM Watson studio and Login using your account. The project and the catalog must be created by members of the same IBM Cloud account.

Step 2: Click **New project** on the home page or on your **Projects** page.

Step 3: Choose whether to create an empty project or to create a project based on an exported project file or a sample project.

Step 4: On the New project screen, add a name and optional description for the project.

Step 5: After giving name , create the storage by clicking Define storage .

Step 6: Click Lite plan as it is free

Step 7 : click create at the bottom of the page

Step 8: Finally , Your project was created .

Output screen short:

The screenshot shows the IBM Watson Studio creation interface on the IBM Cloud platform. The top navigation bar includes 'IBM Cloud', a search bar, 'Catalog', 'Manage', and 'Deepak D's Account'. The main area is titled 'Watson Studio' with a sub-section 'Develop sophisticated machine learning models using Notebooks and code-free tools to infuse AI throughout your business.' Below this, there are two tabs: 'Create' (selected) and 'About'. A sidebar on the left provides details: Type: Service, Provider: IBM, Last updated: 07/06/2022, Category: AI / Machine Learning, Compliance: HIPAA Enabled, IAM-enabled, and Location: Frankfurt, London. The main form asks to 'Select a location' with a dropdown set to 'London (eu-gb)'. It also asks to 'Select a pricing plan' with a note that prices do not include tax and are for United States. A table shows the 'Lite' plan with 1 authorized user, 10 capacity unit-hours monthly limit, and environment requirements. To the right, a 'Summary' panel shows the project details: Watson Studio, Location: London, Plan: Lite, Service name: Watson Studio-wm, and Resource group: Default. It also includes a checkbox for accepting license agreements and buttons for 'Create' and 'Add to estimate'.

IBM Cloud Search resources and products... Catalog Manage Deepak D's Account

Resource list / Watson Studio-wm Active Add tags

Details Actions...

Manage Plan

Watson Studio in Cloud Pak for Data

Watson Studio is one of the core services in Cloud Pak for Data as a Service. Build, deploy and manage AI models, and optimize decisions on IBM Cloud Pak for Data.

Launch in IBM Cloud Pak for Data

IBM Watson Studio in Cloud Pak for Data
IBM Cloud Pak for Data Unifying platform
IBM Cloud Base cloud infrastructure

IBM Watson Studio is part of IBM Cloud Pak for Data and serves as the data science capability of the data fabric architecture.

Helpful links

Documentation Learn about tools, features, and how to use Watson Studio in Cloud Pak for Data

Learning path Start a step-by-step tutorial to get up and running with Watson Studio

Videos Watch videos to learn about Watson Studio in Cloud Pak for Data

Welcome, Deepak!

Take a tutorial Step through implementing a Data fabric use case in a sample project.

Work Create prepare mode

Build and manage ML models with Watson Studio

Watson Studio is a service that you use to build, deploy, and manage AI models and to optimize decisions. Work within a project to build models. Customize how you work by choosing from notebooks, graphical canvases, and no-code tools.

Get started

Provision Watson Studio Create an instance of Watson Studio from the service catalog.

Provision Watson Machine Learning Create an instance of Watson Machine Learning from the service catalog.

Deployments +

No deployment spaces After you create spaces, you'll see them here.

New deployment space +

Welcome, Deepak!

Take a tutorial
Step through implementing a Data fabric use case in a sample project.

Work with data
Create a project for your team to prepare data, find insights, or build models.

Learn what's new
Stay current with new features, enhancements, and other changes.

Quick start

- Build customer profiles** with IBM Match 360 with Watson
- Catalog and govern data** with Watson Knowledge Catalog
- Build and manage ML models** with Watson Studio

Create new project

Projects

EX-1 Today at 06:26 PM

New in gallery

Notifications

No notifications
You will see your most recent notifications here.

Deployments

No deployment spaces
After you create spaces, you'll see them here.

New deployment space

[← Back](#)

Create a project

Choose whether to create an empty project or to preload your project with data and analytical assets. Add collaborators and data, and then choose the right tools to accomplish your goals. Add services as necessary.

Create an empty project

Add the data you want to prepare, analyze, or model. Choose tools based on how you want to work: write code, create a flow on a graphical canvas, or automatically build models.

USE TO

Prepare and visualize data
Analyze data in notebooks
Train models

Create a project from a sample or file

Get started fast by loading existing assets. Choose a project file from your system, or choose a curated sample project.

<https://eu-qb.dataplatform.cloud.ibm.com/projects/create-project?context=cpdaas>

USE TO

Learn by example
Build on existing work
Run tutorials

New project

Define details

Name

Description

Choose project options

Restrict who can be a collaborator 

Mark as sensitive 

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

Define storage

① Select storage service 

Add
Add an object storage instance, and then return to this page and click Refresh.
Refresh.

② Refresh

Cancel
Create

 IBM Watson Studio  Search in your workspaces    Deepak D's Account  London 

[Services catalog](#) /

 **Cloud Object Storage**

Author: IBM • Date of last update: Jul 6, 2022 • [Docs](#) • [API Docs](#)

[Create](#) [About](#)

	COS on Satellite 96TB	COS on Satellite 96TB
Lite	Lite plan instance is free to use for Storage capacity up to 25 GB per month. Lite plan instance is used for trial, and can be easily upgraded to Standard plan for unlimited scalability and full functionality. None Lite plan services are deleted after 30 days of inactivity.	Free 
Standard	Standard plan is our most popular Pay-as-You-Go pricing plan. There is no minimum fee. This plan meets the requirements of most of the enterprise workloads.	See pricing details

[Create](#) [View terms](#)

The screenshot shows the IBM Watson Studio interface. At the top, there's a navigation bar with 'IBM Watson Studio' on the left, a search bar, and account information for 'Deepak D's Account' and 'London'. Below the navigation bar, the title 'Projects / lab3' is displayed. The main content area has tabs for 'Overview', 'Assets', 'Jobs', and 'Manage', with 'Overview' being the active tab. The 'Assets' section contains a message: 'Assets that you create with tools show here. See data assets on the Assets page.' It features a small icon of a diamond and bars. A blue link 'View all' is present. The 'Resource usage' section shows '0 CUH' for the current month. The 'Project history' section shows a recent event: 'You created project lab3 Today at 06:38 PM'. The 'Readme' section is empty, with a placeholder 'Type project notes, reminders, or instructions'.

Result :

Thus, the above experiment was successfully executed

4.DATA REFINERY-PREPARING THE DATA

Procedure:

Step 1: Open IBM Watson studio and Login using your account. The project and the catalog must be created by members of the same IBM Cloud account.

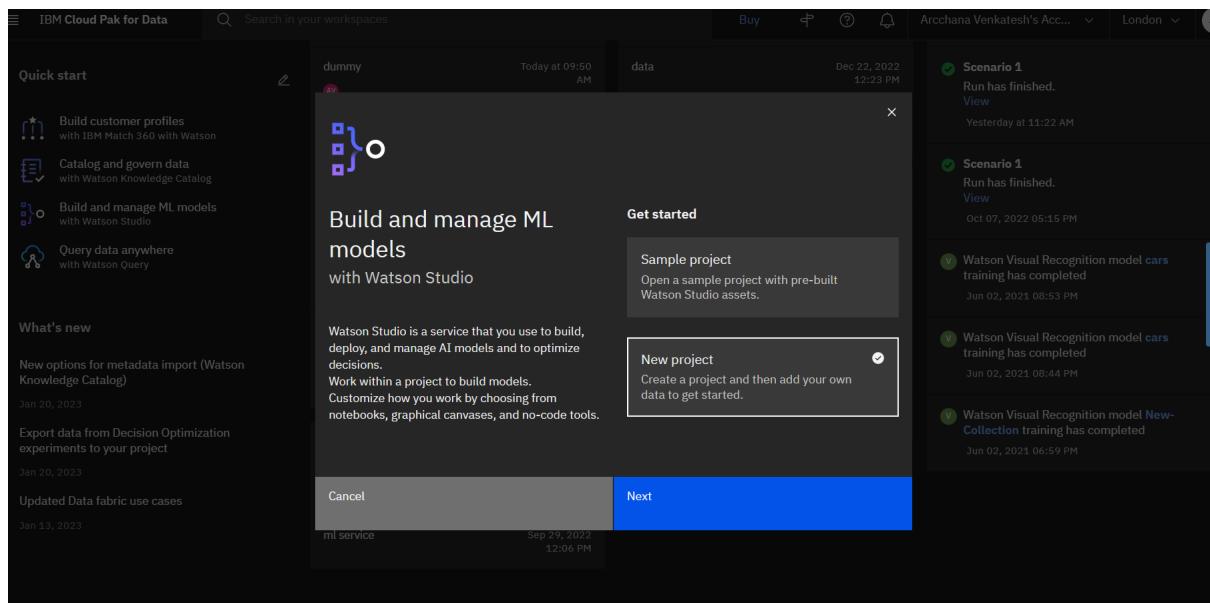
Step 1: click Watson studio h8 .

The screenshot shows the IBM Cloud dashboard with the search bar set to "watson studio". The left sidebar has a "Dashboard" section with a "Build" button. The main area shows "Resource Results" and "Catalog Results" sections, both listing "Watson Studio" as a service. A central panel displays a "Recent AI search in your environment" card and three recommended actions: "Build a virtual machine" (3 min), "Getting started with Watson Discovery" (7 min), and "Getting started with Watson Assistant" (2 min). At the bottom, there are news cards about IBM Data Fabric Solutions, Hyper Protect DBaaS, and PostgreSQL 10 end-of-life. The URL in the address bar is <https://cloud.ibm.com/resources/watson-studio>.

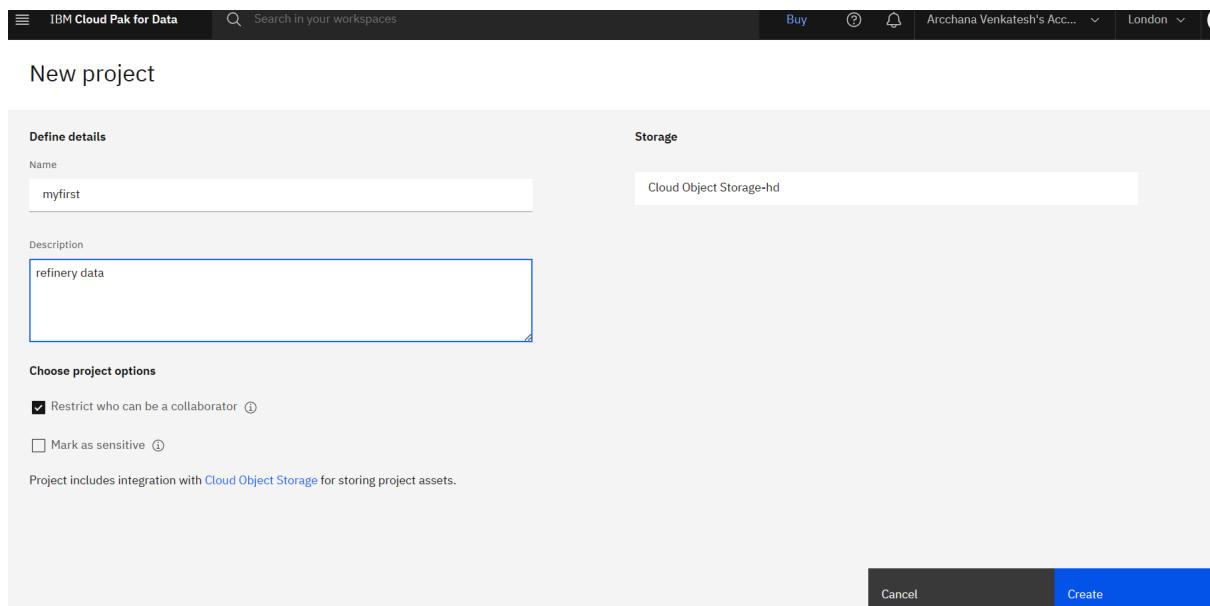
Step 3: click launch in IBM cloud park data

The screenshot shows the service details page for "Watson Studio-h8". The top navigation includes "Catalog", "Manage", and "Arcchana Venkatesh's A...". The main content area features a "Watson Studio in Cloud Pak for Data" section with a "Launch in IBM Cloud Pak for Data" button. Below this, there's a diagram showing "IBM Watson Studio in Cloud Pak for Data" as a component of "IBM Cloud Pak for Data Unifying platform" running on "IBM Cloud Base cloud infrastructure". A note states that Watson Studio is part of Cloud Pak for Data. The "Helpful links" section includes "Documentation", "Learning path", and "Videos". The URL in the address bar is <https://cloud.ibm.com/services/watson-studio-h8>.

Step 4: create new project and give next



Step 5: Give project name and description and storage for your project and click create



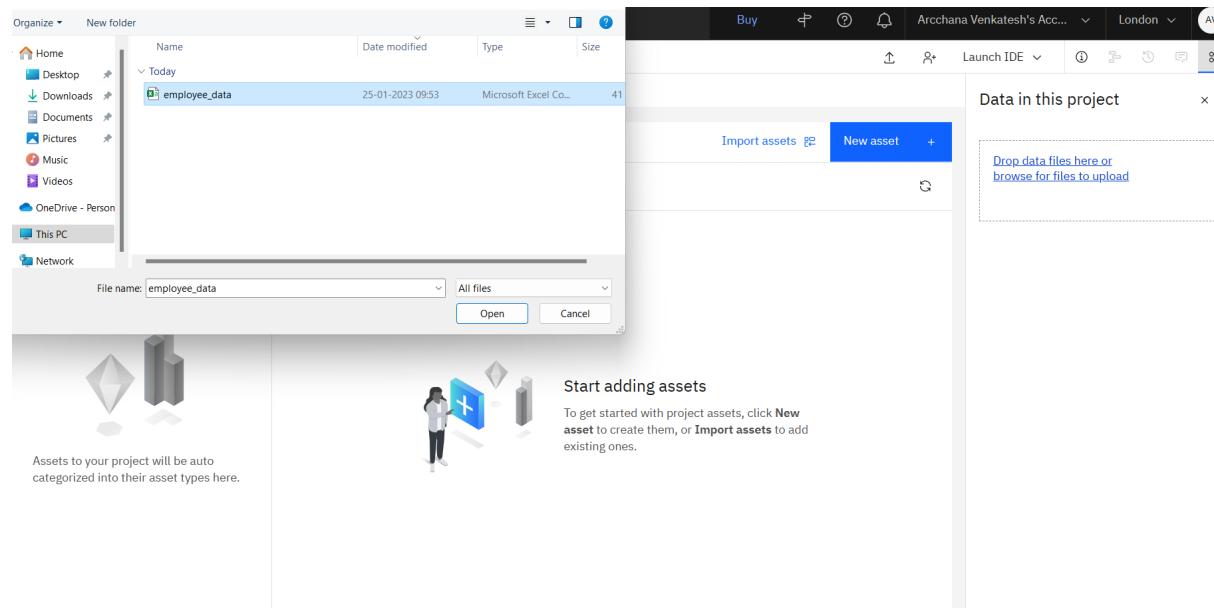
Step 6: Project created successfully .

The screenshot shows the IBM Cloud Pak for Data interface. At the top, there's a navigation bar with 'IBM Cloud Pak for Data', a search bar, and user information for 'Arcchana Venkatesh's Acc...'. Below the navigation bar, the main area is titled 'Projects / myfirst'. There are three tabs: 'Overview' (which is selected), 'Assets', and 'Jobs'. Under 'Overview', there are three sections: 'Assets' (with a note about creating assets with tools), 'Resource usage' (showing 0 CUH for the month), and 'Project history' (listing the creation of the project by the user). A 'Readme' section is also present.

Step 7: Click assets and inset data set

The screenshot shows the 'Assets' tab selected in the IBM Cloud Pak for Data interface. On the left, there's a sidebar with a search bar ('Find assets'), a note about auto-categorizing assets into asset types, and a section for 'Asset types' with a bar chart icon. The main area is titled 'All assets' and contains a message 'Start adding assets' with instructions to click 'New asset' or 'Import assets'. To the right, there's a 'Data in this project' section with a placeholder for dropping or browsing files.

Step 8: import dataset -Employee_data with the help of link



Step 9: Data set was imported successfully and click new asset .

A screenshot of the IBM Cloud Pak for Data interface. The top navigation bar includes 'Buy', 'Arcchana Venkatesh's Acc...', 'London', and a user icon. The main area is titled 'Projects / myfirst'. The 'Assets' tab is selected. On the left, there's a sidebar with '1 assets' and 'All assets' highlighted. The main content area shows a table titled 'All assets' with one item: 'employee_data.csv' (Last modified: Now, Modified by you). A 'Data in this project' sidebar on the right has a 'Drop data files here or browse for files to upload' section.

Step 10: After clicking New asset click Data Refinery and give select .

The screenshot shows two consecutive steps in the IBM Cloud Pak for Data interface.

Step 1: New asset dialog

The 'New asset' dialog is open, showing a search bar and a sidebar with tool types: All types, Data access tools, Automated builders, Graphical builders, and Code editors. A 'Graphical builders' section is visible, featuring icons for Dashboard editor, Data Refinery, Decision Optimization, and Masking flow. The 'Data Refinery' option is highlighted with a blue border.

Step 2: Select data from project dialog

The 'Select data from project' dialog is open, showing a search bar and a sidebar with project and connection details. The main area displays a list of data assets, with 'employee_data.csv' selected and highlighted with a blue border. On the right, the 'Selected assets' panel shows the selected asset: employee_data.csv, with details like Asset name, Asset type (Data asset), Size (40 KB), Last modified (2023/01/25 10:43:35), and Created on (2023/01/25 10:43:35). The 'Select' button at the bottom is highlighted in blue.

Step 11: Refinery process will start working .

The screenshot shows the IBM Cloud Pak for Data interface. The top navigation bar includes 'IBM Cloud Pak for Data', a search bar, and user information for 'Arcchana Venkatesh's Account' and 'London'. The main area displays a 'Refine data' workflow titled 'employee_data.csv / Refine data'. The workflow contains one step: '1. Convert column type', which is described as automatically converting one or more columns to inferred data types. The data preview shows 14 rows of employee information with columns: number, first_name, last_name, gender, and birth_date. The right side of the screen shows the 'About this asset' panel with details like Name: 'employee_data.csv_flow', Description: 'Data Refinery flow', Asset details: Steps: 1, and Associated assets: Source: 'employee_data.csv' and Target: 'employee_data_csv_shaped'. The bottom status bar indicates 'Full data set: 1000 rows, 8 columns'.

Step 12: Click on a new step and aggriate and select the column and operation and create a new column name and give Apply .

The screenshot shows the 'Aggregate' step configuration in the IBM Cloud Pak for Data interface. The 'AGGREGATION 1' section is selected, showing an aggregation for the 'annual_salary' column with the operator 'Minimum'. A new column name 'new' is being typed into the 'Column' input field. The 'Apply' button is highlighted in blue at the bottom. The right side of the screen shows the 'About this asset' panel with details like Name: 'employee_data.csv_flow', Description: 'Data Refinery flow', Asset details: Steps: 1, and Associated assets: Source: 'employee_data.csv' and Target: 'employee_data'. The bottom status bar indicates 'Viewing: 1000 rows, 8 columns' and 'Full data set: 1000 rows, 8 columns'.

Step 13: After Aggregating new data was refined successfully .

The screenshot shows the Data Refinery interface with the following details:

- Project Path:** Projects / myfirst / employee_data.csv / Refine data
- Steps (2):**
 - Data source:** employee_data.csv
 - 1. Convert column type:** Converts 'annual_salary' from String to Integer. Description: "Automatically converted one or more columns to inferred data types. Strings that are converted to decimal use a dot (.) for the decimal symbol." Status: "Auto-generated".
 - 2. Aggregate:** Aggregates the values of 'annual_salary' into 'new'. Description: "Aggregated the values of annual_salary into new". Status: "Just added".
- Data View:** Shows a single row with the value '55003' under the 'new' column.
- Configure:** Buttons for 'Configure' and 'Viewing: 1 rows, 1 columns'.
- Asset Details:**
 - Name:** employee_data.csv_flow
 - Description:** What is the purpose of this Data Refinery flow?
 - Asset details:** Steps: 2
 - Associated assets:** Source: employee_data.csv, Target: employee_data_csv_shaped
 - Last modified:** Not yet saved
 - Created on:** Not yet saved
- Statistics:** Full data set: 1000 rows, 8 columns

Result :

Thus, the above experiment was successfully executed

5. Visualization data in Watson studio

Procedure:

Step 1: Open IBM Watson studio and Login using your account. The project and the catalog must be created by members of the same IBM Cloud account.

Step 1: click Watson studio h8 .

The screenshot shows the IBM Cloud dashboard with the search bar set to "watson studio". The left sidebar has a "Dashboard" section expanded, showing various services like Watson Studio, NeuralSeek, Converistics, Knowledge Studio, and Watson Assistant. The "Watson Studio" service is highlighted with a blue border. The main content area displays "Resource Results" and "Catalog Results" sections, both showing the "Watson Studio" service. A central callout box provides instructions for "Introducing AI search in your environment" by "Adding Watson Knowledge Studio custom models into Watson". Below this, there are three cards: "Build a virtual machine" (recommended, 3 min), "Getting started with Watson Discovery" (recommended, 7 min), and "Getting started with Watson Discovery" (recommended, 2 min). At the bottom, there are sections for "News", "Recent support cases", "Planned maintenance", and "IBM Cloud status".

Step 3: click launch in IBM cloud park data

Step 4: create new project and give next

Step 5: Give project name and description and storage for your project and click create

IBM Cloud Pak for Data Search in your workspaces Buy ? 🔔 Arcchana Venkatesh's Acc... London

New project

Define details

Name: myfirst

Description: refinery data

Storage

Cloud Object Storage-hd

Choose project options

Restrict who can be a collaborator ⓘ

Mark as sensitive ⓘ

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

Cancel
Create

Step 6: Project created successfully .

IBM Cloud Pak for Data Search in your workspaces Buy ? 🔔 Arcchana Venkatesh's Acc... London

Projects / myfirst

Overview	Assets	Jobs	Manage
Assets Assets that you create with tools show here. See data assets on the Assets page.  View all	Resource usage For this month in this project 0 CUH	Project history  You created project myfirst Today at 10:37 AM	
Readme <small>Type project notes, reminders, or instructions</small>			

Step 7: Click assets and inset data set

The screenshot shows the 'Assets' tab in the IBM Cloud Pak for Data interface. On the left, there's a sidebar with 'Asset types' and a note about auto-categorization. The main area shows a message 'Start adding assets' with instructions to click 'New asset' or 'Import assets'. A prominent blue 'New asset' button is located at the top right of the main content area.

Step 8: import dataset -Employee_data with the help of link

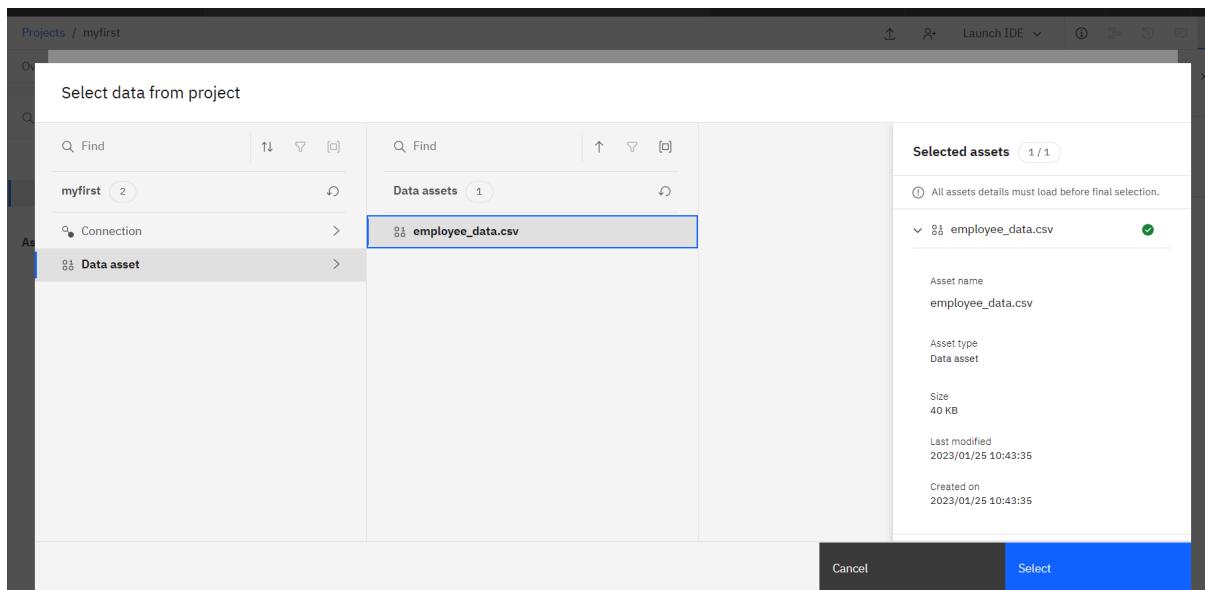
The screenshot shows a file selection dialog box overlaid on the IBM Cloud Pak for Data interface. The dialog box displays a list of files, with 'employee_data' selected. The file path is shown as 'D:\employee_data'. Below the dialog, there's a note about auto-categorization and a 'Start adding assets' section with instructions to click 'New asset' or 'Import assets'.

Step 9: Data set was imported successfully and click new asset .

The screenshot shows the IBM Cloud Pak for Data interface. The top navigation bar includes 'Buy', 'Help', 'Archana Venkatesh's Acc...', 'London', and a user icon. The main header says 'IBM Cloud Pak for Data' and 'Projects / myfirst'. Below it, tabs for 'Overview', 'Assets' (which is selected), 'Jobs', and 'Manage' are visible. A search bar says 'Search in your workspaces'. On the left, a sidebar shows '1 assets' and 'All assets'. The main content area displays 'All assets' with a table showing one item: 'Name: employee_data.csv', 'Type: CSV', 'Last modified: Now', and 'Modified by you'. To the right, a panel titled 'Data in this project' has a section for 'Drop data files here or browse for files to upload'. At the bottom, there are pagination controls.

Step 10: After clicking New asset click Data Refinery and give select .

The screenshot shows the 'New asset' dialog box. The title is 'New asset' and it says 'Select a tool based on what type of asset you want and how you want to work.' On the left, a sidebar lists 'Tool type' categories: 'All types' (selected), 'Data access tools', 'Automated builders', 'Graphical builders', and 'Code editors'. A search bar at the top right says 'Find tools by name or description'. Below, under 'Graphical builders', four options are shown: 'Dashboard editor', 'Data Refinery' (highlighted with a blue border), 'Decision Optimization', and 'Masking flow'. Each option has a small icon and a brief description. At the bottom left, there is a toggle switch for 'Show descriptions'.



Step 11: Refinery process will start working .

	number	first_name	last_name	gender	birth_date
1	483	Lenord	Kihn	M	1994-07-01
2	478	Palma	Beahan	F	1972-05-06
3	348	Hebert	Muller	M	1990-09-04
4	757	Virginia	Ullrich	M	1991-09-23
5	937	Roby	Hudson	M	1997-01-15
6	264	Jaeda	Effertz	--	1994-03-21
7	165	Jalissa	Bogisich	F	1974-01-07
8	211	Lexie	Robel	M	1997-09-05
9	510	Billy	Reilly	M	1977-04-30
10	940	Marlana	Moen	F	1979-05-05
11	175	Aja	Lynch	M	1986-08-21
12	772	Jabez	McLaughlin	M	1971-01-12
13	950	Adrienne	Stokes	F	1982-07-28
14	906	Florene	O'Connell	M	1979-04-03

Step 12: Click on a new step and do filtering ,select column name as first name and select the operation ends with and select text and give “e” and click Apply .

Projects / myfirst / employee_data.csv / Refine data

All Operations / Filter

Filter the rows by keeping the rows with the matching values for the selected columns and removing the other rows. You can create multiple filter conditions in a single operation with the AND/OR selections.

CONDITIONS (1)

CONDITION 1

Column: first_name, Operator: Ends with, Value: e

Do not enclose the value in quotation marks. If the value contains quotation marks, escape them with a slash character.

Add condition +

Cancel Apply

Data Profile Visualizations

first_name String

Lenord
Palma
Hebert
Virginia
Roby
Jaeda
Jalissa
Lexie
Billy
Marlana
Aja
Jabez
Adrienne
Florene

Configure Viewing: 1000 rows, 8 columns Full data set: 1000 rows, 8 columns

About this asset

Name: employee_data.csv_flow
Data Refinery flow

Description: What is the purpose of this Data Refinery flow?

Asset details: Steps: 0

Associated assets: Source: employee_data.csv, Target: employee_data_csv_sh

Last modified: Not yet saved
Created on: Not yet saved

Step 13: first name was end up with “e” successfully and click visualization .

Projects / myfirst / employee_data.csv / Refine data

Steps (1) x

Use a code template to add a step

Data Profile Visualizations

number String first_name String last_name String gender String birth_date String

	number	first_name	last_name	gender	birth_date
1	211	Lexie	Robel	M	1997-09-05
2	950	Adrienne	Stokes	F	1982-07-28
3	906	Florene	O'Connell	M	1979-04-03
4	194	Ace	Upton	--	1987-08-25
5	171	Hope	Zboncak	F	1989-07-18
6	442	Kyrie	Sanford	M	1992-04-15
7	829	Shade	Runte	M	1991-11-08
8	900	Kate	Blick	F	1975-05-10
9	725	Lance	Ratke	M	1979-08-03
10	429	Ernie	Harber	M	1997-10-15
11	38	Duke	Krajcik	M	1977-09-29
12	840	Vassie	Turcotte	F	1999-05-22
13	138	Nannette	Wintheiser	--	1985-10-05
14	845	Shellie	Zboncak	M	1988-08-27

New step +

Configure Viewing: 214 rows, 8 columns Full data set: 1000 rows, 8 columns

About this asset

Name: employee_data.csv_flow
Data Refinery flow

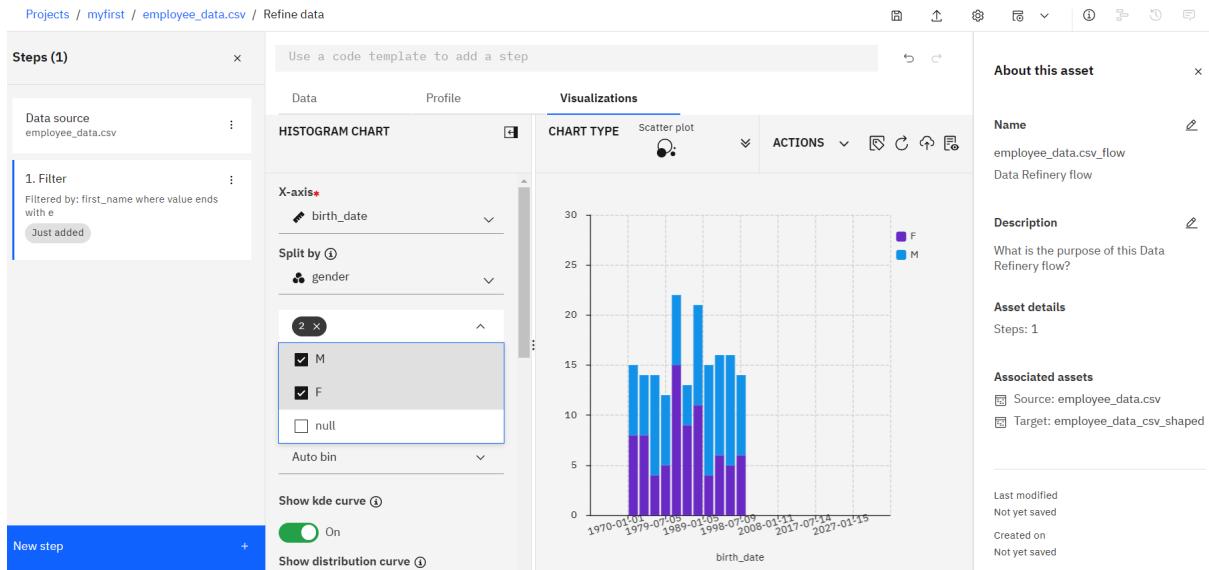
Description: What is the purpose of this Data Refinery flow?

Asset details: Steps: 1

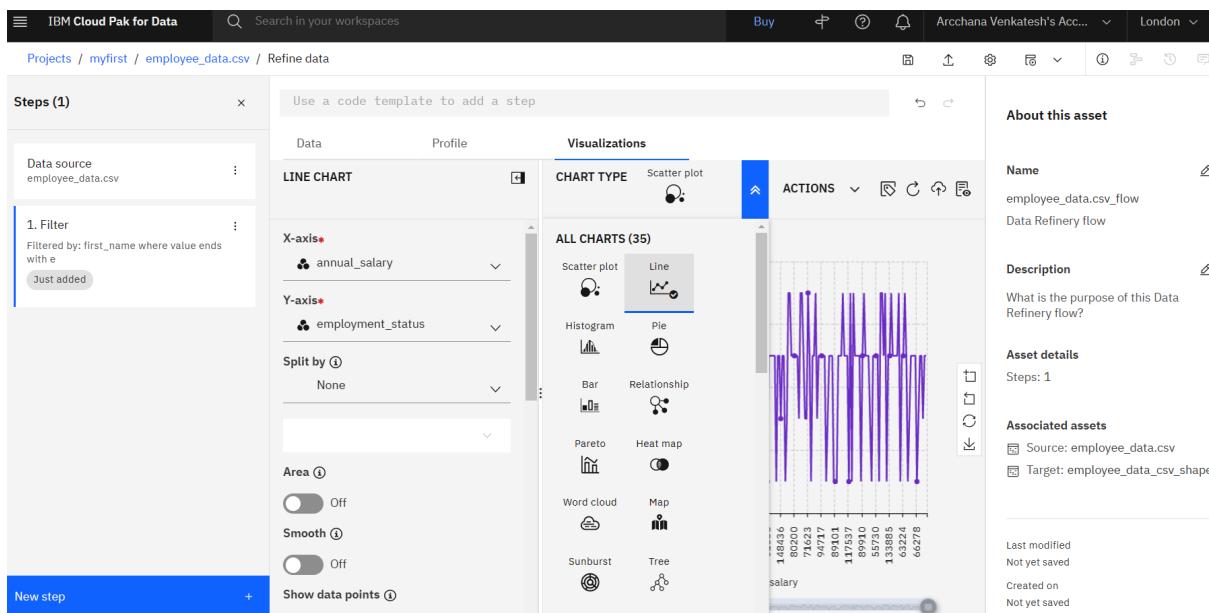
Associated assets: Source: employee_data.csv, Target: employee_data_csv_shaped

Last modified: Not yet saved
Created on: Not yet saved

Step 14: Give x axis and split by and visualization will exist successfully .



Step 14: Select the chart as per requirement and do visualization in watson studio .



Result :

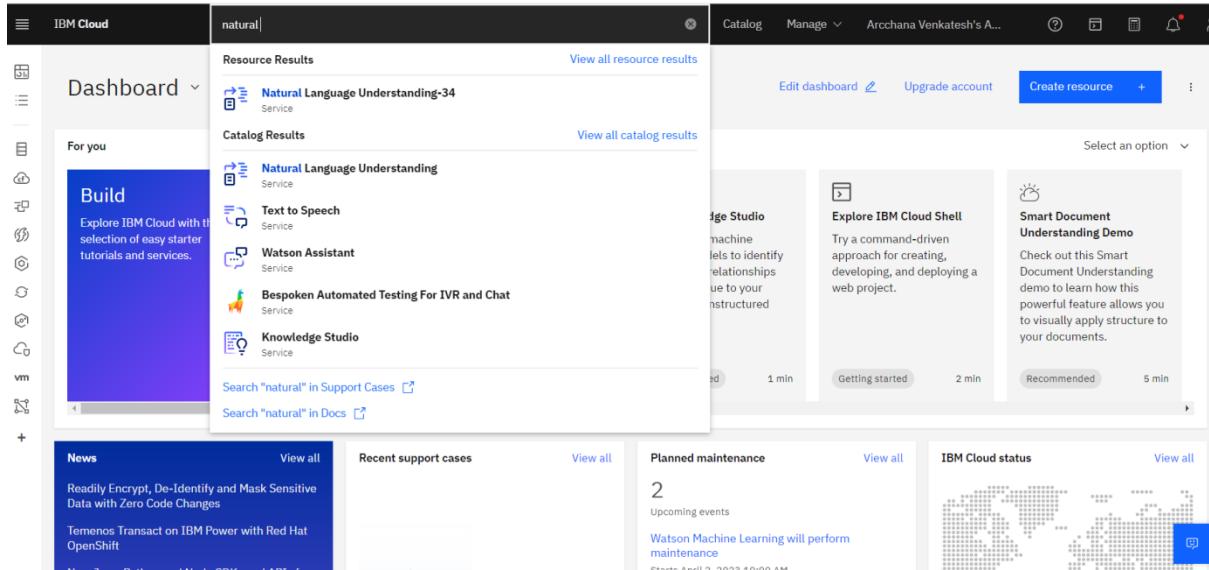
Thus, the above experiment was successfully executed

6.Natural Language Processing using Watson studio.

Procedure:

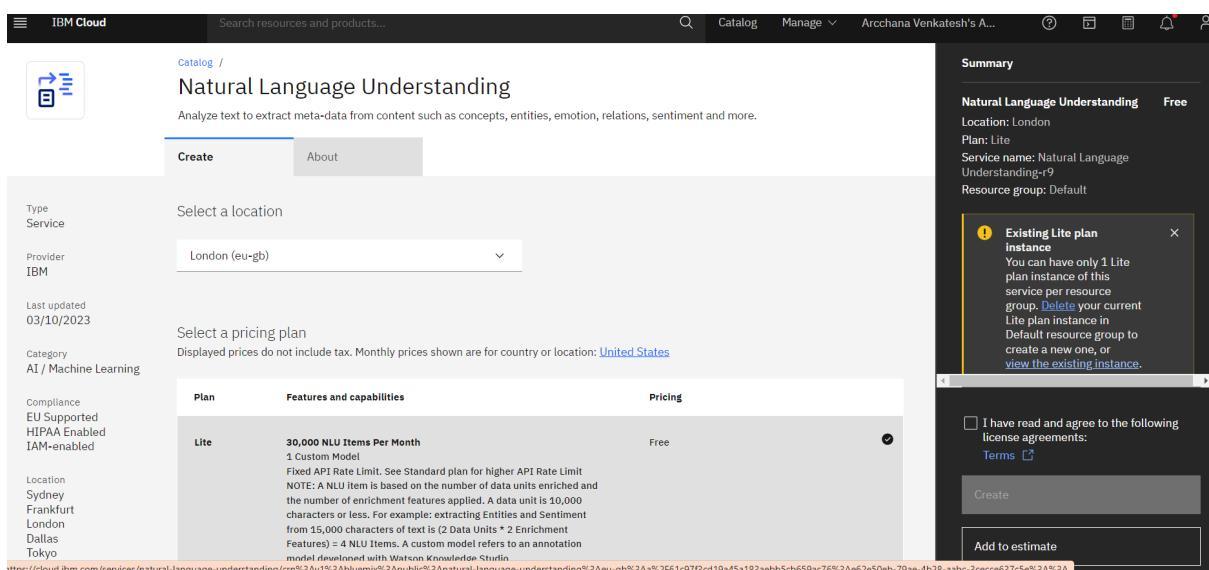
Step 1: Open IBM Watson studio and Login using your account. The project and the catalog must be created by members of the same IBM Cloud account.

Step 2: click Natural language understanding



The screenshot shows the IBM Cloud dashboard with a search bar at the top containing "natural". Under the "Catalog Results" section, the "Natural Language Understanding" service is listed. To the right, there are cards for "Watson Studio", "Explore IBM Cloud Shell", and "Smart Document Understanding Demo". Below the catalog, there are sections for "News", "Recent support cases", "Planned maintenance", and "IBM Cloud status".

Step 3: click the link for view the existing instance link .



The screenshot shows the "Natural Language Understanding" service catalog page. The "Create" tab is selected. A modal window is open, prompting the user to create a new "Existing Lite plan Instance". The modal includes a note about the limit of one instance per resource group, a checkbox for accepting license agreements, and a "Create" button. The "Summary" section on the right provides details about the service, including its location (London), plan (Lite), and resource group (Default).

Step 4: Click on to API reference .

Natural Language Understanding-34 Active Add tags

Manage

- Getting started
- Service credentials
- Plan
- Connections

Start by viewing the tutorial

Getting started tutorial API reference

Credentials

API key: Download Show credentials

URL: View all credentials in the Service credentials tab.

Plan

Lite Upgrade

Step 5: In the right hand side under python the code will already exist , Make use of the code in Python note book inside Watson studio

IBM Cloud API Docs / Natural Language Understanding

Introduction

Last updated: 2023-03-10

Analyze various features of text content at scale. Provide text, raw HTML, or a public URL and IBM Watson Natural Language Understanding will give you results for the features you request. The service cleans HTML content before analysis by default, so the results can ignore most advertisements and other unwanted content.

You can create [custom models](#) with Watson Knowledge Studio to detect custom entities and relations in Natural Language Understanding.

Deprecation Note: IBM is sunsetting Watson Natural Language Understanding Custom Sentiment (BETA). From **June 3, 2023** onward, you will no longer be able to use the Custom Sentiment feature.

To ensure we continue providing our clients with robust and powerful text classification capabilities, IBM recently announced the general availability of a new [single-label text classification capability](#). This new feature includes extended language support and training data customizations suited for building a custom sentiment classifier.

If you would like more information or further guidance, please contact IBM Cloud Support.

Curl Java Node Python .NET

The code examples on this tab use the client library that is provided for Python.

Installation

```
pip install --upgrade "ibm-watson>=6.1.0"
```

GitHub

<https://github.com/watson-developer-cloud/python-sdk>

IBM Cloud Products Solutions Pricing Docs Support Explore more

Natural Language Understanding

Overview Introduction Text analytics features Categories Classifications Concepts Emotion Entities Keywords Metadata Relations Semantic Roles Sentiment Syntax

Categories

Returns a hierarchical [taxonomy](#) of the content. For example, a news website may return categories like `/international/news` or `/arts_and_entertainment`. The top three categories are returned by default.

Categories request options

Request options	Description
<code>categories.explanation</code>	Set this option to <code>true</code> to return explanations for each categorization. This option is available only for English <code>categories</code> . Default: <code>false</code>
<code>categories.model</code>	(Beta) Enter a custom model ID to override the standard categories model for all categories analysis operations in the request. This option is available only for English <code>categories</code> .
<code>categories.limit</code>	Maximum number of categories to return. Default: <code>3</code>

Curl Java Node Python .NET

Example Categories feature request

```
import json
from ibm_watson import NaturalLanguageUnderstandingV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
from ibm_watson.natural_language_understanding_v1 import Features, CategoriesOptions

authenticator = IAMAuthenticator('apikey')
natural_language_understanding = NaturalLanguageUnderstandingV1(
    version='2022-04-07',
    authenticator=authenticator
)

natural_language_understanding.set_service_url('url')

response = natural_language_understanding.analyze(
    url='www.ibm.com',
    features=Features(categories=CategoriesOptions(limit=3)).get_result()

print(json.dumps(response, indent=2))
```

```
import json

from ibm_watson import NaturalLanguageUnderstandingV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
from ibm_watson.natural_language_understanding_v1 \
    import Features, CategoriesOptions

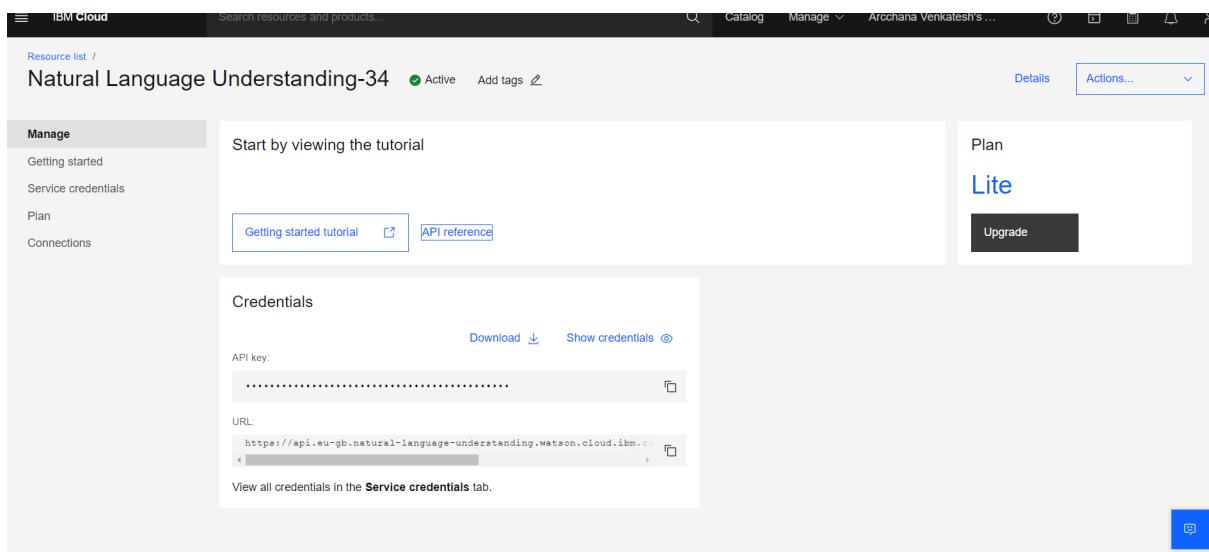
authenticator = IAMAuthenticator('{apikey}')
natural_language_understanding = NaturalLanguageUnderstandingV1(
    version='2022-04-07',
    authenticator=authenticator
)

natural_language_understanding.set_service_url('{url}')

response = natural_language_understanding.analyze(
    url='www.ibm.com',
    features=Features(categories=CategoriesOptions(limit=3))).get_result()

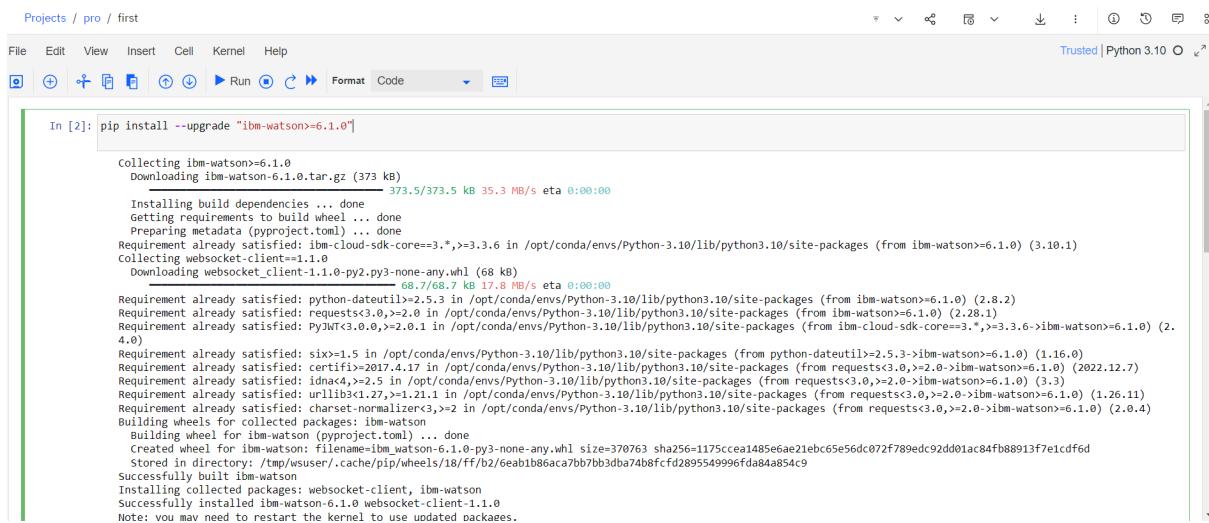
print(json.dumps(response, indent=2))
```

Step 6: Click the URL and paste in the code



The screenshot shows the IBM Cloud service details page for 'Natural Language Understanding-34'. The 'Manage' tab is selected, showing options like 'Getting started', 'Service credentials', 'Plan', and 'Connections'. The 'Plan' section indicates a 'Lite' plan with an 'Upgrade' button. The 'Credentials' section displays an API key and a URL. The URL is highlighted with a red box: `https://api.eu-gb.natural-language-understanding.watson.cloud.ibm.com`.

Step 7: After pip install and write the code in note book and run



```
In [2]: pip install --upgrade "ibm-watson>=6.1.0"

Collecting ibm-watson>=6.1.0
  Downloading ibm-watson-6.1.0.tar.gz (373.5/373.5 kB 35.3 MB/s eta 0:00:00)
    Installing build dependencies ... done
    Getting requirements to build wheel ... done
    Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: ibm-cloud-sdk-core==3.*,>=3.3.6 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson>=6.1.0) (3.10.1)
Collecting websocket-client==1.1.0
  Downloading websocket_client-1.1.0-py2.py3-none-any.whl (68 kB)
    Requirement already satisfied: python-dateutil>=2.5.3 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from websocket-client==1.1.0->=1.1.0)
    Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from requests<3.0,>=2.0 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson>=6.1.0) (2.28.1))
    Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from requests<3.0,>=2.0->ibm-watson>=6.1.0) (3.3)
    Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from requests<3.0,>=2.0->ibm-watson>=6.1.0) (1.26.11)
    Requirement already satisfied: charset-normalizer<3,>=2 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from requests<3.0,>=2.0->ibm-watson>=6.1.0) (2.0.4)
Building wheels for collected packages: ibm-watson
  Building wheel for ibm-watson (pyproject.toml) ... done
  Created wheel for ibm-watson: filename=ibm_watson-6.1.0-py3-none-any.whl size=370763 sha256=1175cce1a485e6ae21ebc65e56dc072f789edc92dd01ac84fb88913f7e1cdf6d
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/18/ff/b2/6eab1b86aca7bb7b3dba74b8fcfd2895549996fda84a854c9
Successfully built ibm-watson
Installing collected packages: websocket-client, ibm-watson
Successfully installed ibm-watson-6.1.0 websocket-client-1.1.0
Note: you may need to restart the kernel to use updated packages.
```

```
In [4]: import json
from ibm_watson import NaturalLanguageUnderstandingV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
from ibm_watson.natural_language_understanding_v1 \
    import Features, CategoriesOptions

authenticator = IAMAuthenticator('ggQ901RE0wcc4s0lI86tgUFHN5jq1XPkRnCYce0d3js')
natural_language_understanding = NaturalLanguageUnderstandingV1(
    version='2022-04-07',
    authenticator=authenticator
)
natural_language_understanding.set_service_url('https://api.eu-gb.natural-language-understanding.watson.cloud.ibm.com/instances/e62e50eb-79ae-4b28-aabc-3cecce637c5e')

response = natural_language_understanding.analyze(
    url='www.ibm.com',
    features=Features(categories=CategoriesOptions(limit=3)).get_result()

print(json.dumps(response, indent=2))
```

Output :

```
{ "usage": { "text_units": 1, "text_characters": 1138, "features": 1 }, "retrieved_url": "https://www.ibm.com/uk-en", "language": "en", "categories": [ { "score": 0.952923, "label": "/business and finance/industries/technology industry" }, { "score": 0.936069, "label": "/technology & computing/computing" }, { "score": 0.917225, "label": "/education/online education" } ] }
```

Result :

Thus, the above experiment was successfully executed

7.Linner Regression in R

Procedure:

Step:1 Create the predictor and response variable.

Step:2 Give the chart file a name

Step:3 Plot the chart and save the chart

Step:4 Predict the values of testing data.

Step 5: save the code by dev.off() and run the code Linner regression plot will get conformed in console

Program code:

```
# Create the predictor and response variable.
```

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
```

```
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
```

```
relation <- lm(y~x)
```

```
# Give the chart file a name.
```

```
png(file = "linearregression.png")
```

```
# Plot the chart.
```

```
plot(y,x,col = "blue",main = "Height & Weight Regression",
abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in
cm")
```

```
# Save the file.
```

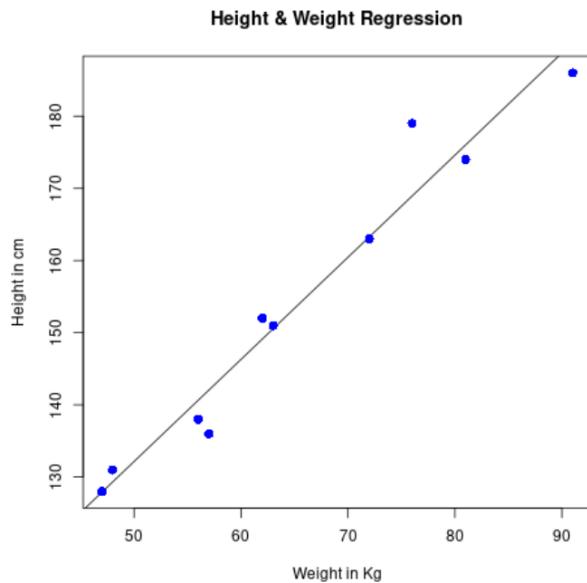
```
dev.off()
```

```

1 # Create the predictor and response variable.
2 x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
3 y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
4 relation <- lm(y~x)
5
6 # Give the chart file a name.
7 png(file = "linearregression.png")
8
9 # Plot the chart.
10 plot(y,x,col = "blue",main = "Height & Weight Regression",
11 abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")
12
13 # Save the file.
14 dev.off()
15

```

Output:



Result :

Thus, the above experiment was successfully execute.

8.Decision tree in R programming

Procedure:

Step:1 install the needed library – datasets, caTools, Party, Dplyr , mgridtr.

Step:2 Split the dataset into training and testing data.

Step:3 Create a model for decision tree algorithm using R programming language.

Step:4 Predict the values of testing data.

Step 5: find the accuracy of the decision tree model.

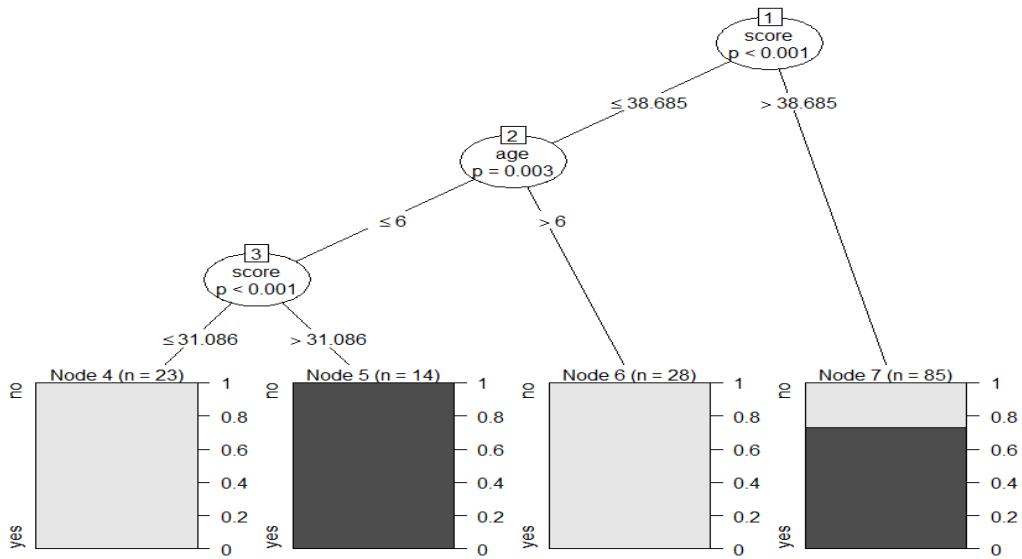
Step 6: Save the code and run the code Decision tree will get conformed in console.

Program:

```
library(datasets)
library(caTools)
library(party)
library(dplyr)
library(magrittr)
data("readingSkills")
head(readingSkills)
```

```
> data("readingskills")
> head(readingskills)
  nativeSpeaker age shoesize    score
1      yes     5  24.83189 32.29385
2      yes     6  25.95238 36.63105
3      no     11  30.42170 49.60593
4      yes     7  28.66450 40.28456
5      yes    11  31.88207 55.46085
6      yes    10  30.07843 52.83124
```

```
sample_data = sample.split(readingSkills, SplitRatio = 0.8)
train_data <- subset(readingSkills, sample_data == TRUE)
test_data <- subset(readingSkills, sample_data == FALSE)
model<- ctree(nativeSpeaker ~ ., train_data)
plot(model)
```



```

predict_model<-predict(ctree_, test_data)
m_at <- table(test_data$nativeSpeaker, predict_model)
m_at
  
```

```

> predict_model<-predict(ctree_,test_data)
> predict_model<-predict(ctree_,test_data)
> m_at <- table(test_data$nativeSpeaker,predict_model)
> m_at
  predict_model
    no yes
  no 13 13
  yes 0 24
  
```

```

ac_Test <- sum(diag(table_mat)) / sum(table_mat)
print(paste('Accuracy for test is found to be', ac_Test))
  
```

```

> ac_Test <- sum(diag(table_mat)) / sum(table_mat)
> print(paste('Accuracy for test is found to be', ac_Test))
[1] "Accuracy for test is found to be 0.74"
> 
  
```

Result :

Thus, the above experiment was successfully executed

