Mobile Price Prediction

A PROJECT REPORT

Submitted by

Vaibhav Pramod Kumar Singh

in partial fulfillment for the award ofthe degree of

RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ART'S, SCIENCE& COMMERCE (AUTONOMOUS), GHATKOPAR W

April-2023

RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ART'S, SCIENCE & COMMERCE (AUTONOMOUS), GHATKOPAR W

(Affiliated to University of Mumbai)

Certificate

This is to certify that the Project entitled ***Mobile Price Prediction*** *is bonafide work of* ***Vaibhav Pramod Kumar Singh*** *bearing Seat No* ***735*** *submitted in partial fulfillment of the requirements for the award of Degree****Master of Science*** *in* ***Data Science & Artificial Intelligence.***

Signature of Internal Guide **Signature of Co-Ordinator**

College Seal and Date **Signature Examiner**

1. ABSTRACT

The purpose of this project was to build a model that could accurately predict the prices of mobile phones based on various features. To achieve this, I used Azure AutoML to train a model and UiPath for web scraping to collect data from an online mobile store. The scraped data was preprocessed to remove any missing or invalid values and prepare it for use in training the model.

The preprocessed data was then fed into Azure AutoML, which automatically selected the best algorithm and hyperparameters for the data. The trained model was then deployed using Azure Web Services and Postman.

The model was evaluated using various metrics, including mean squared error and R-squared. The most important features for predicting mobile prices were found to be battery power, RAM, internal memory, and screen size.

This project showcases the application of various techniques in data science, including web scraping, data preprocessing, model training, and model deployment. Web scraping was carried out using UiPath, a popular tool for data extraction, while Azure AutoML was used for model training, a technique that automates the selection of the best algorithm and hyperparameters for the data. Azure Web Services was used for deploying the trained model, which allowed for easy access and use of the model.

The results of this project demonstrate the effectiveness of using Azure AutoML and UiPath for building models for predicting mobile phone prices. By using the model, consumers can make informed decisions about which mobile phone to purchase, and retailers can adjust their pricing strategy based on predicted demand. Furthermore, this project highlights the importance of understanding the most relevant features for predicting mobile phone prices.

In conclusion, this project shows the potential of using web scraping, machine learning, and model deployment in building models for predicting mobile phone prices. The use of Azure AutoML, UiPath, and Azure Web Services provides an effective framework for building and deploying machine learning models, which can be used to improve decision-making and business strategies in various industries.s.

2. ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my project guide "**Mr. Madhav Mishra" and "Mr. Mujtaba Shaikh"** for his monitoring andconstant encouragement throughout the course of this project work, without his encouragement and guidance this project would not have materialized.

I also extend my sincere appreciation to Director of RJ college "Dr. (Mrs.) Usha Mukundan" and to the principle of RJ college "Dr. Himanshu Dawda" and all other staff members of the department.

3. DECLARATION

I hereby declare that the project entitled, “Mobile Prediction” done at R.J.COLLEGE, Ghatkopar (W),Mumbai has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, No one has submitted to any other university. The Project is done in partial fulfillment of the requirements for the award of degree of MASTERS OF SCIENCE (Data Science and Artificial Intelligence) to be submitted as major project as part of our curriculum.

Vaibhav Singh

CONTENTS

4.INTRODUCTION

As someone interested in the field of machine learning, I undertook a project to predict mobile phone prices using Azure AutoML, UiPath, and other tools. The mobile phone industry is constantly evolving, with new models being released every year with different features and price points. This can make it difficult for consumers to make informed decisions about which mobile phone to purchase, and for retailers to set appropriate prices based on consumer demand. Machine learning algorithms can be used to address this challenge by predicting mobile phone prices based on relevant features.

To begin the project, I used UiPath to scrape data from an online mobile store. The data includes various features such as battery power, RAM, internal memory, and screen size, along with their corresponding prices. I then preprocessed the data to remove any missing or invalid values and prepare it for use in training a machine learning model.

I used Azure AutoML to train a model on the preprocessed data. Azure AutoML automates the selection of the best algorithm and hyperparameters for the data, making it a powerful tool for machine learning. I evaluated the performance of the model using metrics such as mean squared error and R-squared.

Once the model was trained and evaluated, I deployed it using Azure Web Services and Postman. This allowed me to easily access and use the model to predict mobile phone prices based on their features. I also identified the most important features for predicting mobile prices based on the performance of the model.

The results of this project can be useful for both consumers and retailers in the mobile phone industry. Consumers can use the model to make informed decisions about which mobile phone to purchase based on predicted prices, while retailers can use the model to adjust their pricing strategy based on predicted demand. Additionally, this project showcases the use of various techniques in data science, including web scraping, machine learning, and model deployment.

In conclusion, as someone interested in machine learning, the mobile price prediction project was a useful application of machine learning in the mobile phone industry. By using Azure AutoML, UiPath, and other tools, I was able to predict mobile phone prices based on relevant features and improve decision-making for both consumers and retailers..

5. PROBLEM STATEMENT

As a data science enthusiast, I undertook a project to predict mobile phone prices using Azure AutoML and UiPath. The mobile phone industry is constantly evolving, with new models being released every year with different features and price points. This can make it difficult for consumers to make informed decisions about which mobile phone to purchase, and for retailers to set appropriate prices based on consumer demand. Machine learning algorithms can be used to address this challenge by predicting mobile phone prices based on relevant features.

To build a machine learning model for price prediction, a large amount of data and expertise in data science are required. Additionally, collecting and preprocessing the data can be time-consuming and challenging, especially if the data is obtained through web scraping. To address these challenges, I used UiPath to efficiently scrape data from an online mobile store. The data includes various features such as battery power, RAM, internal memory, and screen size, along with their corresponding prices.

I used Azure AutoML to train a model on the preprocessed data. Azure AutoML automates the selection of the best algorithm and hyperparameters for the data, making it a powerful tool for machine learning. I evaluated the performance of the model using metrics such as mean squared error and R-squared. Once the model was trained and evaluated, I deployed it using Azure Web Services and Postman, allowing for easy access and use for both consumers and retailers.

The aim of this project is to build a machine learning model that accurately predicts mobile phone prices based on relevant features. The success of this project will provide valuable insights into the mobile phone industry, including identifying the most important features for predicting prices and improving decision-making for both consumers and retailers. The results of this project can be useful for consumers to make informed decisions about which mobile phone to purchase based on predicted prices, while retailers can use the model to adjust their pricing strategy based on predicted demand.

In conclusion, this project showcases the use of various techniques in data science, including web scraping, machine learning, and model deployment, to predict mobile phone prices. The success of this project will have practical applications in the mobile phone industry and demonstrate the power of machine learning in addressing real-world challenges.

6. DATASET DESCRIPTION

The dataset, available in the form of a CSV file, contains information about mobile phones sold on Flipkart, an Indian e-commerce platform. The dataset has 3114 rows and 8 columns, with each row representing a single mobile phone and each column representing a specific attribute or feature of the phone. The columns in the dataset are:

Brand: The brand name of the mobile phone.
Model Name: The model name or number of the mobile phone.
Color: The color of the mobile phone.
RAM (in GB): The amount of RAM in the mobile phone.
Storage (in GB): The storage capacity of the mobile phone..
Price: The selling price of the mobile phone on Flipkart.
Rating: The customer rating of the mobile phone on Flipkart.

The dataset contains both categorical and numerical variables, making it suitable for machine learning models that can handle both types of data. The dataset is preprocessed and cleaned, with no missing or null values in any of the columns.

The dataset is a CSV file containing information on mobile phones sold on Flipkart, an Indian e-commerce platform. The dataset consists of 3114 rows and 8 columns, with each row representing a mobile phone and each column representing a specific attribute or feature of the phone.

The dataset has been preprocessed and cleaned, with no missing or null values in any of the columns. It was used in the mobile price prediction project to train and evaluate a machine learning model for predicting mobile phone prices based on relevant features.

Overall, the dataset provides a valuable resource for developing and evaluating machine learning models related to the mobile phone industry, and can help identify important features for predicting prices and improving decision-making for both consumers and retailers.

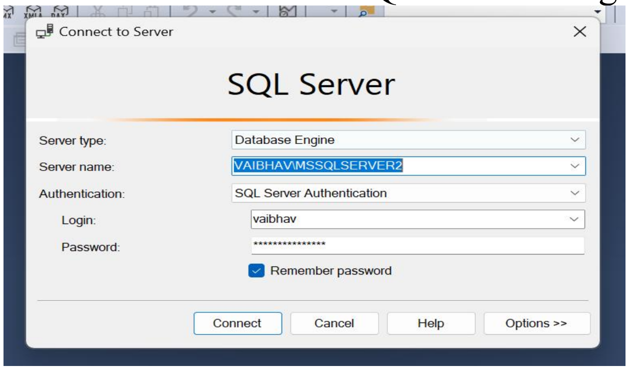
7. METHODOLOGY

- Data Collection: In this project, data was collected from Flipkart's website using UIpath. The data included various features of mobile phones such as brand, model name, color, RAM, storage, display size, camera resolution, battery capacity, processor, operating system, price, and rating. The data was saved in a CSV file for further analysis.
- Data Preprocessing: The collected data was preprocessed to clean and transform it into a format suitable for machine learning models. This included handling missing or incorrect values, encoding categorical variables using one-hot encoding, and scaling numerical variables using the StandardScaler function from the Scikit-learn library. The preprocessed data was then split into training and validation sets.
- Model Selection: Azure AutoML was used to select an appropriate machine learning algorithm for the problem at hand. AutoML is an automated machine learning service provided by Azure that can quickly build, train, and deploy machine learning models without the need for extensive coding or data science expertise. In this project, we selected a regression algorithm to predict mobile phone prices based on the available features.
- Model Training: The selected machine learning algorithm was trained using Azure AutoML. The training process involved selecting appropriate hyperparameters such as the maximum number of iterations, the regularization parameter, and the learning rate. AutoML then searched for the best combination of hyperparameters using a combination of techniques such as grid search and random search. The trained model was then saved for further evaluation.
- Model Evaluation: The performance of the trained model was evaluated on a test dataset. Various metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were used to evaluate the model's accuracy. The best-performing model was then selected for deployment.
- Model Deployment: The selected model was deployed using Azure Web Service. The deployed model was integrated with a Flask web application that allowed users to input the features of a mobile phone and receive a predicted price. The web application was hosted on Azure App Service and could be accessed through a web browser or API.
- Monitoring and Updating: The performance of the deployed model was monitored over time using Azure Application Insights. Any necessary updates or improvements to the model were made based on feedback from users or changes in the data.

8. AUTOML STEPS

Step 1:- Import data from Microsoft Excel into Microsoft SQL Server

1. Connect to the local SQL Server using Microsoft SQL Server Management Studio



Connect to Server

SQL Server

Server type: Database Engine

Server name: VAIBHAV\MSSQLSERVER2

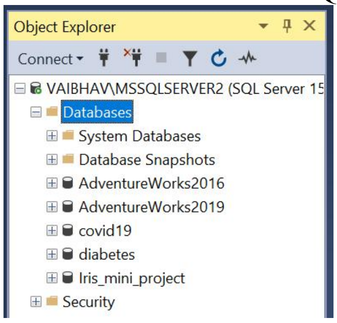
Authentication: SQL Server Authentication

Login: vaibhav

Password: ***************

☑ Remember password

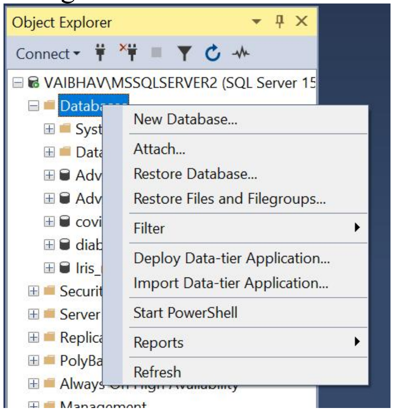
2. Look for the available SQL Databases in the system.



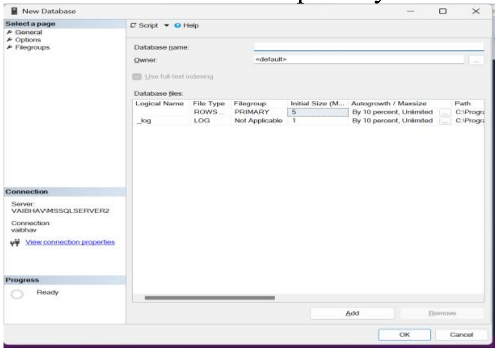
Object Explorer

VAIBHAV\MSSQLSERVER2 (SQL Server 15
- Databases
  - System Databases
  - Database Snapshots
  - AdventureWorks2016
  - AdventureWorks2019
  - covid19
  - diabetes
  - Iris_mini_project
- Security

3. Right Click on the data base to create a new SQL Database for the file to be uploaded.



4. Enter the Name for the New Database which is needed to be created and keep rest of the things as default because in the further step the system will create the necessary details from the source file itself.



New Database

Select a page: General, Options, Filegroups

Database name:

Owner: <default>

Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (M...	Autogrowth / Maxsize	Path
	ROWS ...	PRIMARY	5	By 10 percent, Unlimited	C:\Progr
_log	LOG	Not Applicable	1	By 10 percent, Unlimited	C:\Progr

Connection

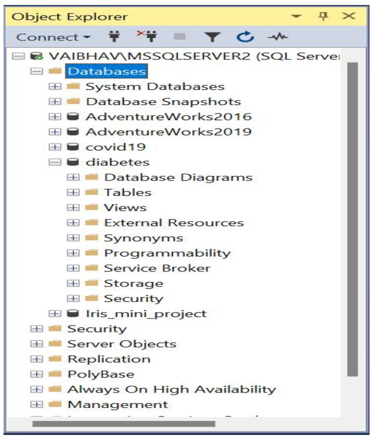
Server: VAIBHAV\MSSQLSERVER2

Connection: vaibhav

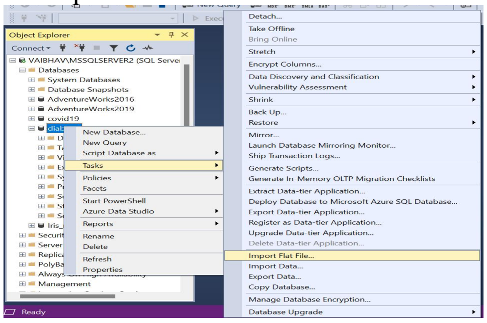
View connection properties

Progress: Ready

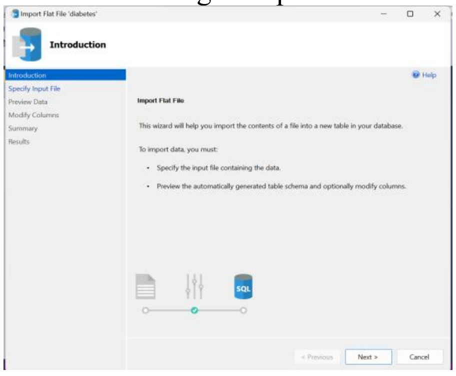
5. Once the Database is created refresh the Server and look for the newly created Database and expand it for further use.



6. After expanding the Database, right click on the database name and then locate the task option, click on the task option and look for the format of the file to be imported as for this I have used a flat file.



7. After Selecting the option from the above step a new Import Wizard will appear and click next to proceed.



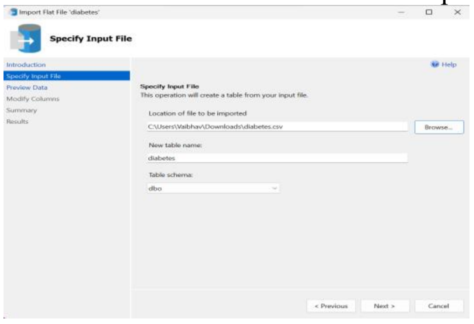
Import Flat File

This wizard will help you import the contents of a file into a new table in your database.

To import data, you must:

- Specify the input file containing the data.
- Preview the automatically generated table schema and optionally modify columns.

8. In this specify the path of the file which is needed to be imported and also mention the name of the table with the table schema for further use purpose in migration of Data and click next.



Specify Input File
This operation will create a table from your input file.

Location of file to be imported
C:\Users\Vaibhav\Downloads\diabetes.csv

New table name:
diabetes

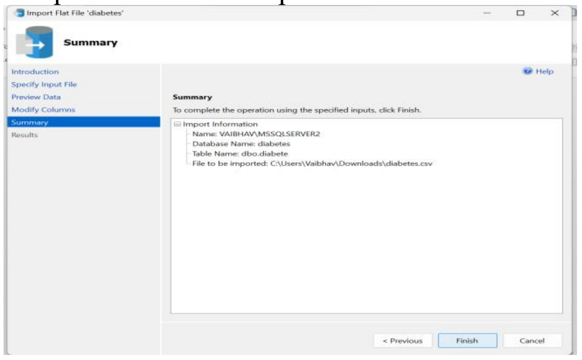
Table schema:
dbo

9. Once the above step is done a Preview of Data will open, Preview the data which is needed to be imported

and click next.

10. Define the necessary Keys and look for the data types are they correctly allocated if not then updated it in the wizard and click next.

11. Once the above steps are done correctly summary will be generated and click on the finish button to complete the task of import of file from local to SQL Database.



Summary

To complete the operation using the specified inputs, click Finish.

Import Information
- Name: VAIBHAV\MSSQLSERVER2
- Database Name: diabetes
- Table Name: dbo.diabete
- File to be imported: C:\Users\Vaibhav\Downloads\diabetes.csv

12. Once the task is finished refresh the Server and expand the Tables Option in the Database and right click on the table name which was created with schema in the import wizard step and then click on the Select Statement to check and use the data in the further process.

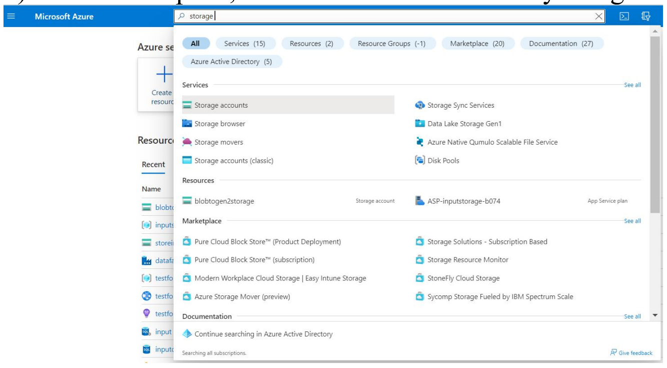
Step 2:- Copy data from a SQL Server database to Azure Blob storage

General Steps:-
1. Create a data factory.
2. Create a self-hosted integration runtime.
3. Create SQL Server and Azure Storage linked services.
4. Create SQL Server and Azure Blob datasets.
5. Create a pipeline with a copy activity to move the data.
6. Start a pipeline run.
7. Monitor the pipeline run.

Get the storage account name and account key

You use the name and key of your storage account in this. To get the name and key of your storage account, take the following steps:

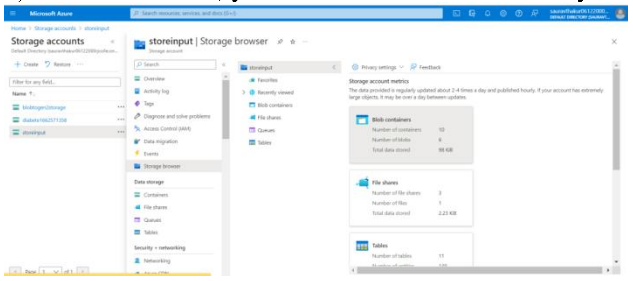
1) Sign in to the Azure portal with your Azure user name and password.
2) In the left pane, select All services. Filter by using the Storage keyword, and then select Storage accounts.



3) In the list of storage accounts, filter for your storage account if needed. Then select your storage account.
4) In the Storage account window, select Access keys.
5) In the Storage account name and key1 boxes, copy the values, and then paste them into Notepad or another editor for later use

Create the container

1) In this section, you create a blob container in your Blob storage.



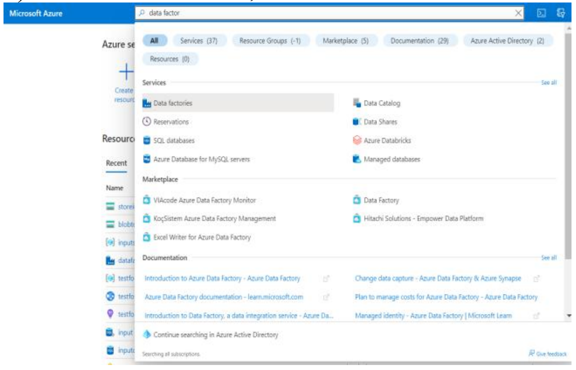
2) In the Storage account window, go to Overview, and then select Containers.
3) Select Blobs option
4) In the Containers window, select + Container to create a new one.
5) In the New container window, under Name, enter NAME_FOR_THE_CONTAINER. Then select Create.
6) In the list of containers, select container you just created.

Keep the container window open. You use it to verify the output at the end of the below step. Data Factory automatically creates the output folder in this container, so you don't need to create one.

Create a data factory

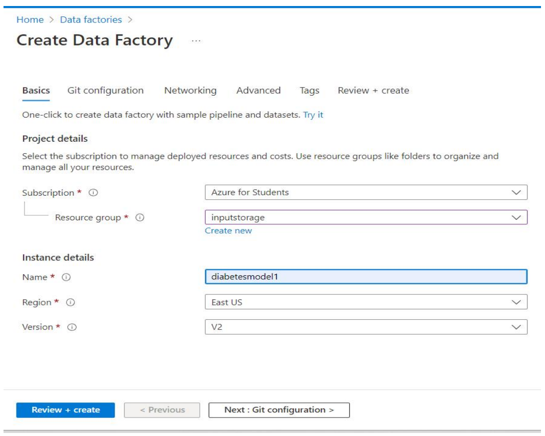
In this step, you create a data factory and start the Data Factory UI to create a pipeline in the data factory.

1) On the left menu, select Create a resource > Integration > Data Factory:



Microsoft Azure
data factor
All
Services (37)
Resource Groups (-1)
Marketplace (5)
Documentation (29)
Azure Active Directory (2)
Resources (0)
Services
Data factories
Reservations
SQL databases
Azure Database for MySQL servers
Data Catalog
Data Shares
Azure Databricks
Managed databases
Marketplace
VIAcode Azure Data Factory Monitor
KoçSistem Azure Data Factory Management
Excel Writer for Azure Data Factory
Data Factory
Hitachi Solutions - Empower Data Platform
Documentation
Introduction to Azure Data Factory - Azure Data Factory
Azure Data Factory documentation - learn.microsoft.com
Introduction to Data Factory, a data integration service - Azure Da...
Change data capture - Azure Data Factory & Azure Synapse
Plan to manage costs for Azure Data Factory - Azure Data Factory
Managed identity - Azure Data Factory | Microsoft Learn
Continue searching in Azure Active Directory
Searching all subscriptions.

2) Data Factory selection in the &quot;New&quot; pane
3) On the New data factory page, under Name, enter a unique name.



Home > Data factories >

Create Data Factory

Basics | Git configuration | Networking | Advanced | Tags | Review + create

One-click to create data factory with sample pipeline and datasets. Try it

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure for Students
Resource group *	inputstorage
	Create new

Instance details

Name *	diabetesmodel1
Region *	East US
Version *	V2

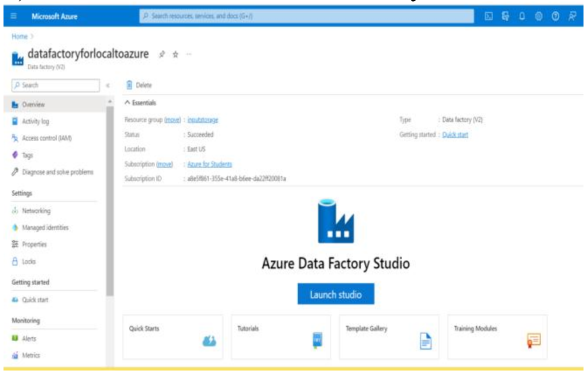
Review + create | < Previous | Next : Git configuration >

4) Select the Azure subscription in which you want to create the data factory.
5) For Resource Group, take one of the following steps:
   a) Select Use existing, and select an existing resource group from the drop-down list.
   b) Select Create new, and enter the name of a resource group.
6) Under Version, select V2.
7) Under Location, select the location for the data factory. Only locations that are supported are displayed in

the drop-down list. The data stores and computes used by Data Factory can be in other regions.

8) Select Create.

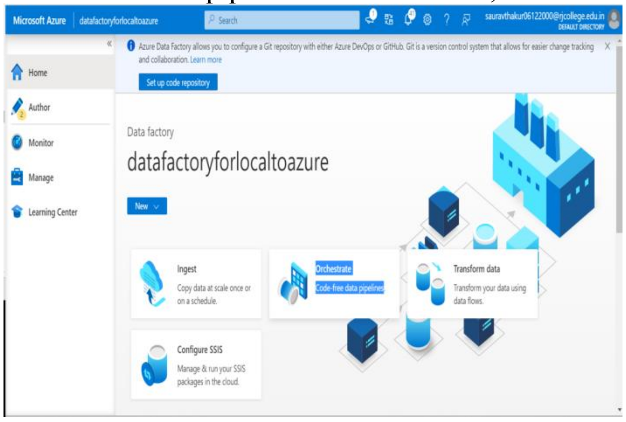
9) After the creation is finished, you see the Data Factory page as shown in the image:



10) Select Open on the Open Azure Data Factory Studio tile to launch the Data Factory UI in a separate tab.

Create a pipeline

1) On the Azure Data Factory home page, select Orchestrate. A pipeline is automatically created for you. You see the pipeline in the tree view, and its editor opens.

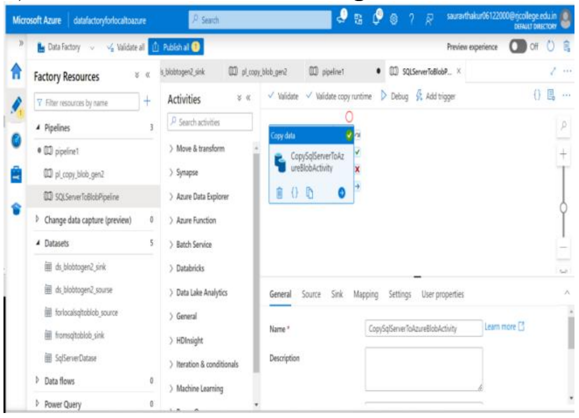

Microsoft Azure
datafactoryforlocaltoazure
Home
Author
Monitor
Manage
Learning Center
Set up code repository
Data factory
datafactoryforlocaltoazure
New
Ingest
Copy data at scale once or on a schedule.
Orchestrate
Code-free data pipelines
Transform data
Transform your data using data flows.
Configure SSIS
Manage & run your SSIS packages in the cloud.

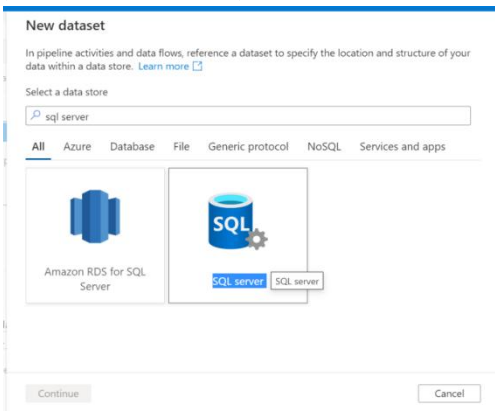
2) In the General panel under Properties, specify SQLServerToBlobPipeline for Name. Then collapse the panel by clicking the Properties icon in the top-right corner.

3) In the Activities tool box, expand Move & Transform. Drag and drop the Copy activity to the pipeline design surface. Set the name of the activity to CopySqlServerToAzureBlobActivity.

4) In the Properties window, go to the Source tab, and select + New.

5) In the New Dataset dialog box, search for SQL Server. Select SQL Server, and then select Continue.

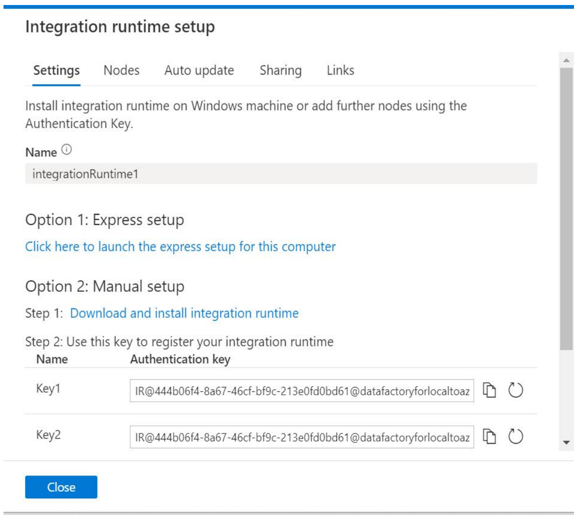



New dataset
In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. Learn more
Select a data store
sql server
All
Azure
Database
File
Generic protocol
NoSQL
Services and apps
Amazon RDS for SQL Server
SQL server
Continue
Cancel

6) In the Set Properties dialog box, under Name, enter SqlServerDataset. Under Linked service, select + New. You create a connection to the source data store (SQL Server database) in this step.

7) In the New Linked Service dialog box, add Name as SqlServerLinkedService. Under Connect via integration runtime, select +New. In this section, you create a self-hosted integration runtime and associate it with an on-premises machine with the SQL Server database. The self-hosted integration

runtime is the component that copies data from the SQL Server database on your machine to Blob storage.
8) In the Integration Runtime Setup dialog box, select Self-Hosted, and then select Continue.
9) Under name, enter TutorialIntegrationRuntime. Then select Create.
10) For Settings, select Click here to launch the express setup for this computer. This action installs the integration runtime on your machine and registers it with Data Factory. Alternatively, you can use the manual setup option to download the installation file, run it, and use the key to register the integration runtime.


Integration runtime setup

Settings | Nodes | Auto update | Sharing | Links

Install integration runtime on Windows machine or add further nodes using the Authentication Key.

Name: integrationRuntime1

Option 1: Express setup
Click here to launch the express setup for this computer

Option 2: Manual setup
Step 1: Download and install integration runtime
Step 2: Use this key to register your integration runtime

Name	Authentication key
Key1	IR@444b06f4-8a67-46cf-bf9c-213e0fd0bd61@datafactoryforlocaltoaz
Key2	IR@444b06f4-8a67-46cf-bf9c-213e0fd0bd61@datafactoryforlocaltoaz

Close

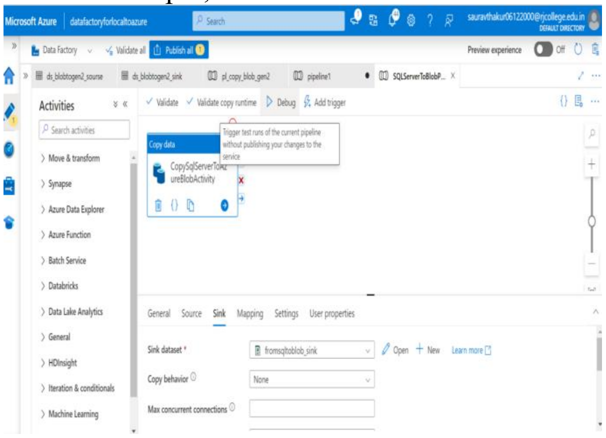
11. In the Integration Runtime (Self-hosted) Express Setup window, select Close when the process is finished.
12. In the New linked service (SQL Server) dialog box, confirm that TutorialIntegrationRuntime is selected under Connect via integration runtime. Then, take the following steps:
a. Under Name, enter SqlServerLinkedService.
b. Under Server name, enter the name of your SQL Server instance.
c. Under Database name, enter the name of the database with the emp table.
d. Under Authentication type, select the appropriate authentication type that Data Factory should use to connect to your SQL Server database.
e. Under User name and Password, enter the user name and password. Use mydomain\myuser as user name if needed.
f. Select Test connection. This step is to confirm that Data Factory can connect to your SQL Server database by using the self-hosted integration runtime you created.
g. To save the linked service, select Create.
13) After the linked service is created, you're back to the Set properties page for the SqlServerDataset. Take the following steps:
a. In Linked service, confirm that you see SqlServerLinkedService.
b. Under Table name, select [dbo].[table_name].
c. Select OK.
14) Go to the tab with SQLServerToBlobPipeline, or select SQLServerToBlobPipeline in the tree view.
15) Go to the Sink tab at the bottom of the Properties window, and select + New.
16) In the New Dataset dialog box, select Azure Blob Storage. Then select Continue.
17) In Select Format dialog box, choose the format type of your data. Then select Continue.
18) In the Set Properties dialog box, enter AzureBlobDataset for Name. Next to the Linked service text box, select + New.
19) In the New Linked Service (Azure Blob Storage) dialog box, enter AzureStorageLinkedService as name, select your storage account from the Storage account name list. Test connection, and then select Create to deploy the linked service.
20) After the linked service is created, you're back to the Set properties page. Select OK.
21) Open the sink dataset. On the Connection tab, take the following steps:
a. In Linked service, confirm that AzureStorageLinkedService is selected.
b. In File path, enter ‘container_created’ /fromonprem for the Container/ Directory part. If the output folder doesn't exist in the ‘container_created’ container, Data Factory automatically creates the output folder.

c. For the File part, select Add dynamic content. dynamic expression for resolving file name

d. Add @CONCAT(pipeline().RunId, '.txt'), and then select Finish. This action will rename the file with PipelineRunID.txt.

22) Go to the tab with the pipeline opened, or select the pipeline in the tree view. In Sink Dataset, confirm that AzureBlobDataset is selected.

23) To validate the pipeline settings, select Validate on the toolbar for the pipeline. To close the Pipe validation output, select the >> icon. validate pipeline



Sink dataset * fromsqltoblob_sink

Copy behavior None

Max concurrent connections

24) To publish entities you created to Data Factory, select Publish all.

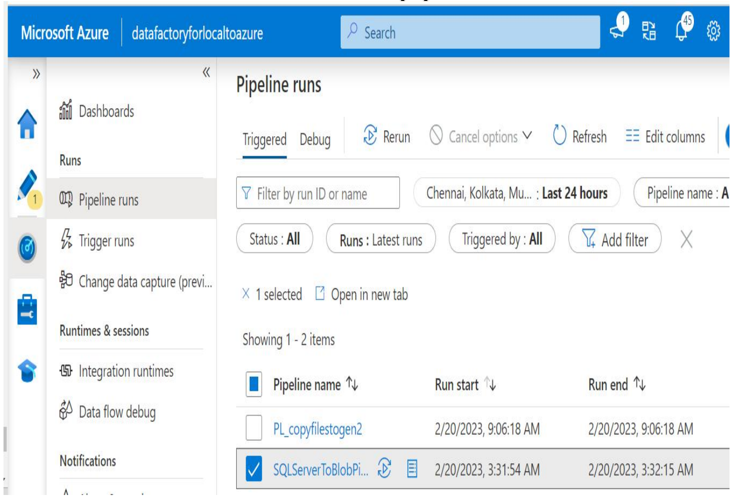
25) Wait until you see the Publishing completed pop-up. To check the status of publishing, select the Show notifications link on the top of the window. To close the notification window, select Close.

Trigger a pipeline run

Select Add Trigger on the toolbar for the pipeline, and then select Trigger Now.

Monitor the pipeline run

1) Go to the Monitor tab. You see the pipeline that you manually triggered in the previous step.
2) To view activity runs associated with the pipeline run, select the SQLServerToBlobPipeline link under PIPELINE NAME. Monitor pipeline runs



Pipeline runs

Showing 1 - 2 items

Pipeline name	Run start	Run end
PL_copyfilestogen2	2/20/2023, 9:06:18 AM	2/20/2023, 9:06:18 AM
SQLServerToBlobPi...	2/20/2023, 3:31:54 AM	2/20/2023, 3:32:15 AM

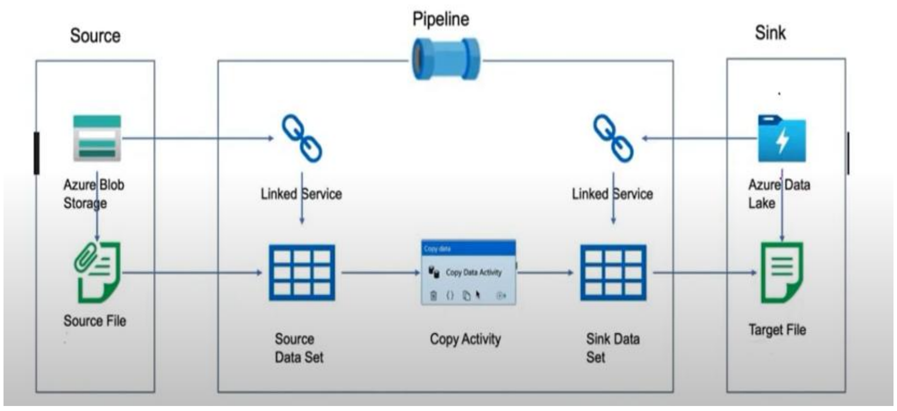
3) On the Activity runs page, select the Details (eyeglasses image) link to see details about the copy operation. To go back to the Pipeline Runs view, select All pipeline runs at the top.

Step 3:- Copy data from a Azure Blob storage to azure data lake gen2 storage in Azure Data Factory

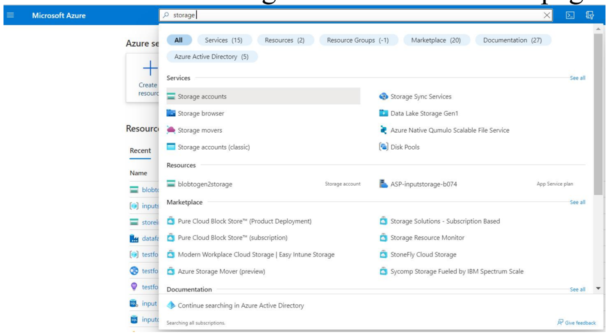
General Steps:-

1) Create a data factory.

2) Create a self-hosted integration runtime.
3) Create Azure Data Lake Gen2 Storage and Azure Storage linked services.
4) Create Azure Blob datasets(source) and Azure Data Lake Gen2(sink)
5) Create a pipeline with a copy activity to move the data.
6) Start a pipeline run.
7) Monitor the pipeline run.



Note:- ** the steps upto to the Create Pipeline is same as above. Kindly refer that.
Just one new task to be done apasrt from above as creating a Azure Data Lake Gen2 Storage.
Steps:
1. Search for the storage account in the home page and click on the storage account.



Microsoft Azure

storage

All | Services (15) | Resources (2) | Resource Groups (-1) | Marketplace (20) | Documentation (27) | Azure Active Directory (5)

Services — See all
- Storage accounts
- Storage Sync Services
- Storage browser
- Data Lake Storage Gen1
- Storage movers
- Azure Native Qumulo Scalable File Service
- Storage accounts (classic)
- Disk Pools

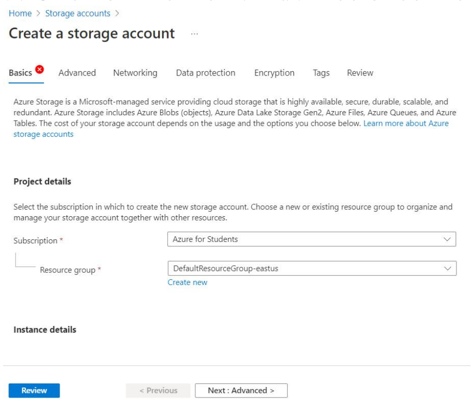
Resources
- blobtogen2storage — Storage account
- ASP-inputstorage-b074 — App Service plan

Marketplace — See all
- Pure Cloud Block Store™ (Product Deployment)
- Storage Solutions - Subscription Based
- Pure Cloud Block Store™ (subscription)
- Storage Resource Monitor
- Modern Workplace Cloud Storage | Easy Intune Storage
- StoneFly Cloud Storage
- Azure Storage Mover (preview)
- Sycomp Storage Fueled by IBM Spectrum Scale

Documentation — See all
- Continue searching in Azure Active Directory

Searching all subscriptions.
Give feedback

2. Once the Storage account open, look for create option
3. Clcik on the create option and fill the required details for the creation of the account.
4. Provide a Unique name to the storage account so that it can be easily recognizable.
5. Click on the Next: Advance once the details are filled.



Home > Storage accounts >

Create a storage account

Basics | Advanced | Networking | Data protection | Encryption | Tags | Review

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. Learn more about Azure storage accounts

Project details

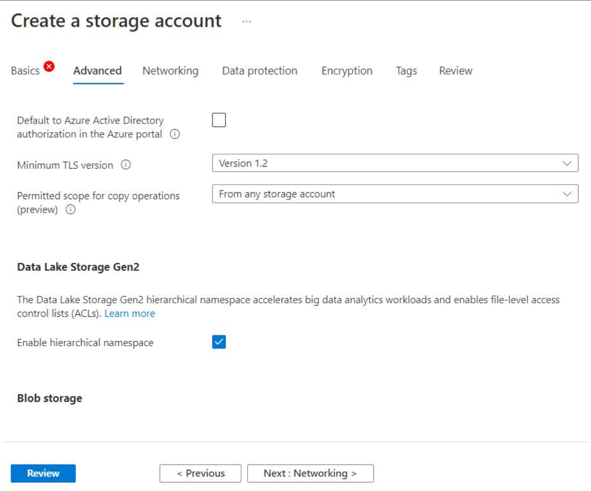
Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription * : Azure for Students
Resource group * : DefaultResourceGroup-eastus
Create new

Instance details

Review | < Previous | Next : Advanced >

6. In the Advance tab scroll down and look for Data Lake Storage Gen2 and enable the same to create the storage account.



Default to Azure Active Directory authorization in the Azure portal	☐
Minimum TLS version	Version 1.2
Permitted scope for copy operations (preview)	From any storage account

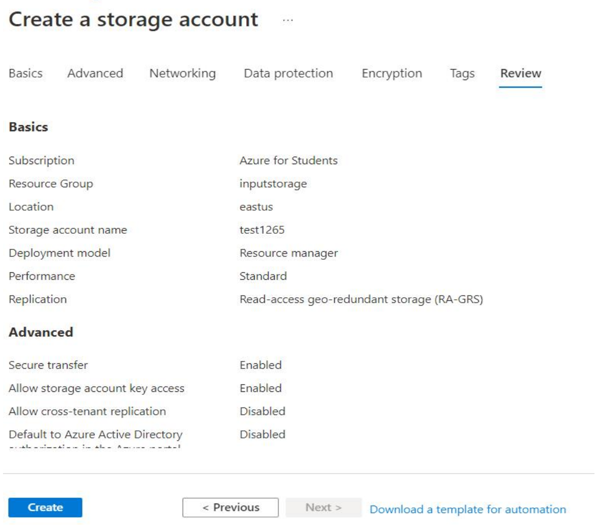
Data Lake Storage Gen2

The Data Lake Storage Gen2 hierarchical namespace accelerates big data analytics workloads and enables file-level access control lists (ACLs). Learn more

Enable hierarchical namespace	✓

Blob storage

7. After this click on review and create Azure Data Lake Gen2 Storage account.



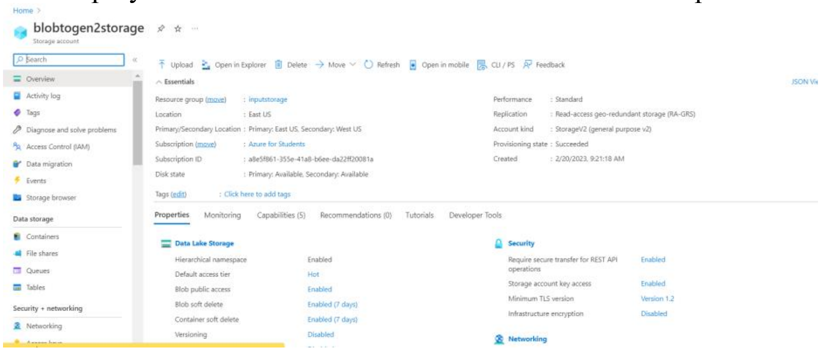
Basics

Subscription	Azure for Students
Resource Group	inputstorage
Location	eastus
Storage account name	test1265
Deployment model	Resource manager
Performance	Standard
Replication	Read-access geo-redundant storage (RA-GRS)

Advanced

Secure transfer	Enabled
Allow storage account key access	Enabled
Allow cross-tenant replication	Disabled
Default to Azure Active Directory	Disabled

8. Once the process is completed wait for the deployment to be completed and then open the storage account after deployment and click on the container tab in left side panel of the storage account.



Resource group (move)	: inputstorage
Location	: East US
Primary/Secondary Location	: Primary: East US, Secondary: West US
Subscription (move)	: Azure for Students
Subscription ID	: a8e5f861-355e-41a8-b6ee-da22ff20081a
Disk state	: Primary: Available, Secondary: Available
Performance	: Standard
Replication	: Read-access geo-redundant storage (RA-GRS)
Account kind	: StorageV2 (general purpose v2)
Provisioning state	: Succeeded
Created	: 2/20/2023, 9:21:18 AM
Tags (edit)	: Click here to add tags

Data Lake Storage

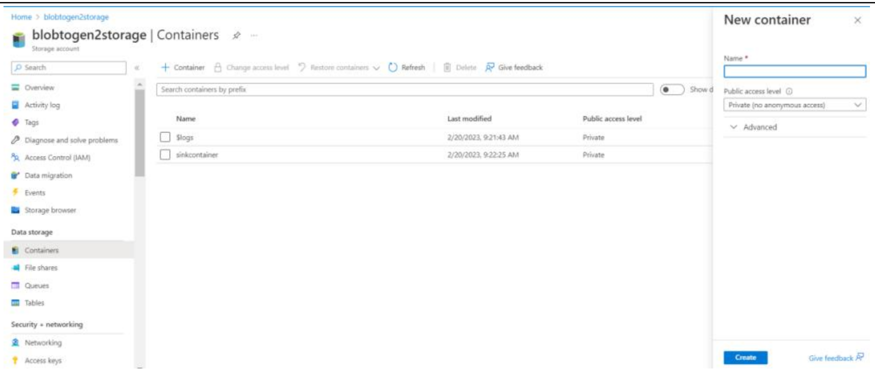
Hierarchical namespace	Enabled
Default access tier	Hot
Blob public access	Enabled
Blob soft delete	Enabled (7 days)
Container soft delete	Enabled (7 days)
Versioning	Disabled

Security

Require secure transfer for REST API operations	Enabled
Storage account key access	Enabled
Minimum TLS version	Version 1.2
Infrastructure encryption	Disabled

Networking

9. Once the container tab open look for the header and click on the + container option and create a new container for the file location where the file is needed to be store.


Home > blobtogen2storage

blobtogen2storage | Containers

Storage account

	Name	Last modified	Public access level
☐	$logs	2/20/2023, 9:21:43 AM	Private
☐	sinkcontainer	2/20/2023, 9:22:25 AM	Private

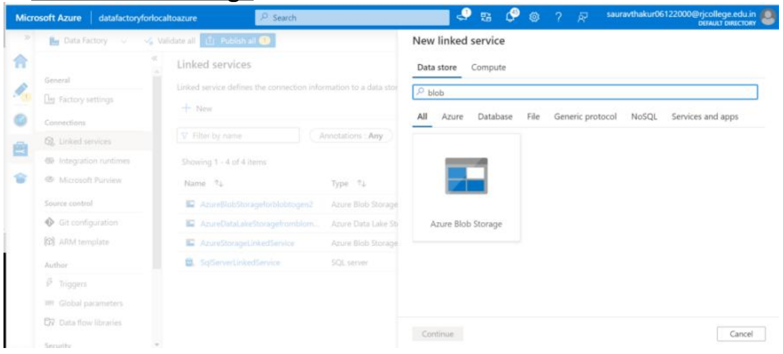
New container

Name *

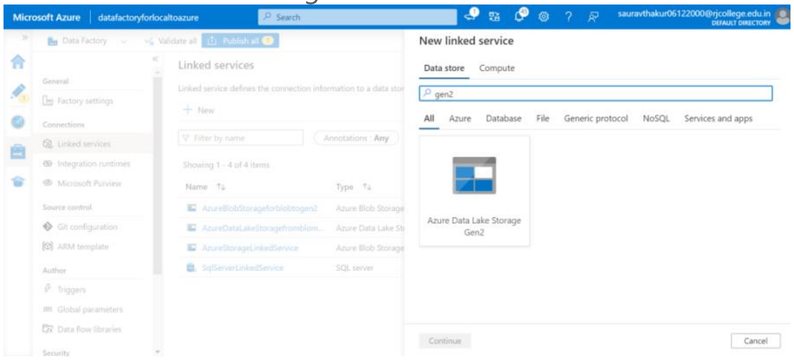
Public access level: Private (no anonymous access)

Create a Linked Services:

1. Azure Blob Storage



2. Azure Data Lake Storage Gen2

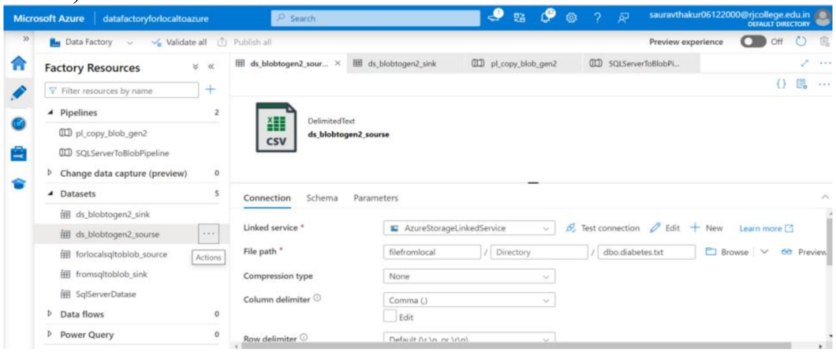


Create Factory Resources:

1. Datasets
2. Pipeline

1. Datasets

    a) Source Dataset.


DelimitedText

ds_blobtogen2_sourse

Linked service *: AzureStorageLinkedService

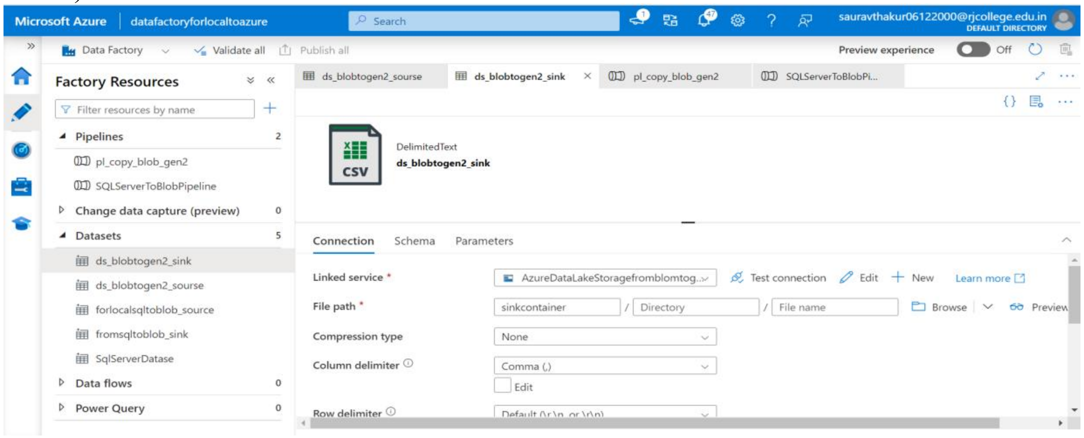
File path *: filefromlocal / Directory / dbo.diabetes.txt

Compression type: None

Column delimiter: Comma (,)

Row delimiter: Default (\r,\n, or \r\n)

b) Sink Dataset

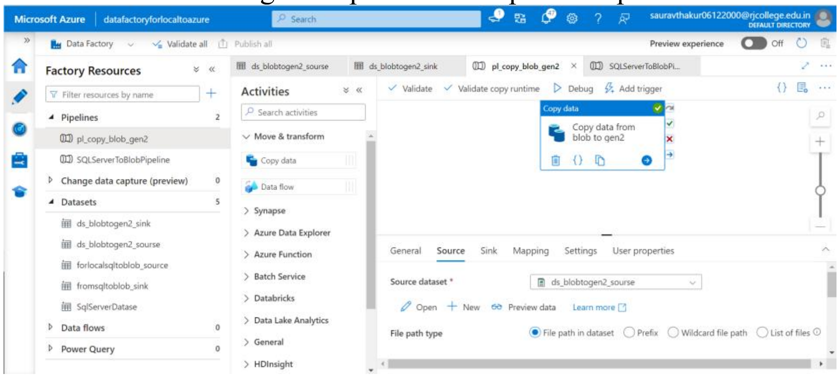


DelimitedText
ds_blobtogen2_sink

Connection | Schema | Parameters

Linked service * AzureDataLakeStoragefromblomtog...
File path * sinkcontainer / Directory / File name
Compression type None
Column delimiter Comma (,)
Row delimiter Default (\r\n, or \r\n)

Pipeline:-
1. Create a new pipeline
2. Search for the copy option in the activities section.
3. Drag and drop the copy option
4. Provide a name to the Pipeline
5. Use the source and sink data set in defining the pipeline properties.
6. Check the connection after using the datasets to avoid any error.
7. Validate and Debug the Pipeline to complete the process.



Copy data
Copy data from blob to gen2

General | Source | Sink | Mapping | Settings | User properties

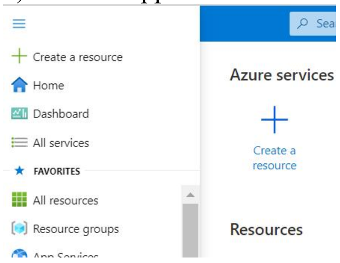
Source dataset * ds_blobtogen2_sourse
File path type ● File path in dataset ○ Prefix ○ Wildcard file path ○ List of files

** note for detailed steps look for the above step in copying data from sql to blob storage and replace the properties of SQL with Blob and Blob with Azure Data Lake Gen2 Storage.

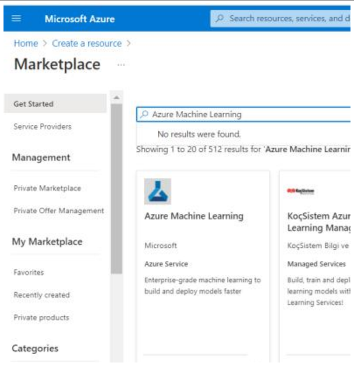
<u>Step 4:- Creating a Automl model.</u>

<u>Create a workspace</u>

1) Sign in to the Azure portal by using the credentials for your Azure subscription.
2) In the upper-left corner of the Azure portal, select the three bars, then + Create a resource.



3) Use the search bar to find Azure Machine Learning.
4) Select Azure Machine Learning.

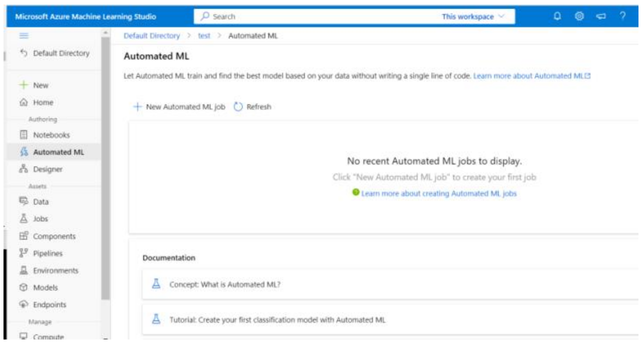


No results were found.
Showing 1 to 20 of 512 results for 'Azure Machine Learnir
Enterprise-grade machine learning to build and deploy models faster
Build, train and depl learning models wit Learning Services!

5) In the Machine Learning pane, select Create to begin.
6) Provide the necessary information to configure your new workspace:
7) After you're finished configuring the workspace, select Review + Create.
8) Select Create to create the workspace. When the process is finished, a deployment success message appears.
9) To view the new workspace, select Go to resource.
10) From the portal view of your workspace, select Launch studio to go to the Azure Machine Learning studio.

Sign in to the studio

1) Sign in to Azure Machine Learning studio.
2) Select your subscription and the workspace you created.
3) Select Get started.
4) In the left pane, select Automated ML under the Author section.

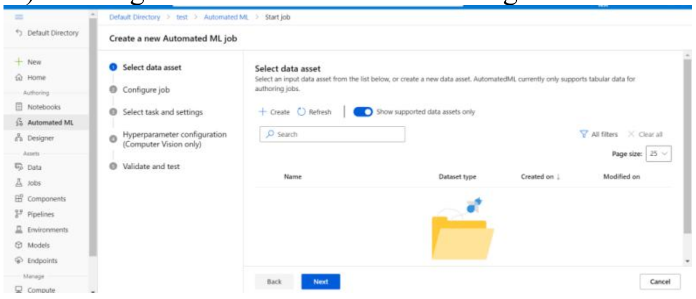


Let Automated ML train and find the best model based on your data without writing a single line of code. Learn more about Automated ML
No recent Automated ML jobs to display.
Click "New Automated ML job" to create your first job
Learn more about creating Automated ML jobs
Concept: What is Automated ML?
Tutorial: Create your first classification model with Automated ML

5) Select +New automated ML job.

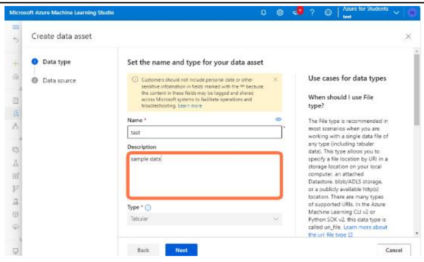
Steps for the Automated ML job.

1)Selecting a dataset for the model training.



Select data asset
Select an input data asset from the list below, or create a new data asset. AutomatedML currently only supports tabular data for authoring jobs.

2)Providing a name for the dataset.


Create data asset

Data type
Data source

Set the name and type for your data asset

Customers should not include personal data or other sensitive information in fields marked with the because the content in these fields may be logged and shared across Microsoft systems to facilitate operations and troubleshooting. Learn more

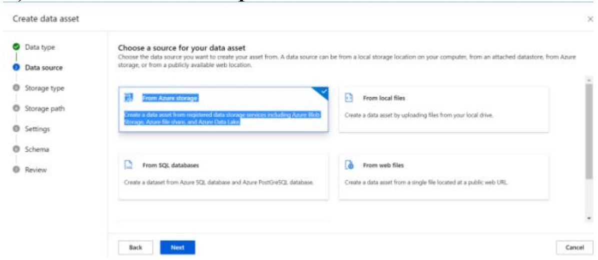
Name *: test
Description: sample data
Type *: Tabular

Use cases for data types

When should I use File type?

The File type is recommended in most scenarios when you are working with a single data file of any type (including tabular data). This type allows you to specify a file location by URI in a storage location on your local computer, an attached Datastore, blob/ADLS storage, or a publicly available http(s) location. There are many types of supported URIs. In the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called uri_file. Learn more about the uri_file type

3)Provind the path where we have copied the data set in the previous step


Create data asset

Data type
Data source
Storage type
Storage path
Settings
Schema
Review

Choose a source for your data asset

Choose the data source you want to create your asset from. A data source can be from a local storage location on your computer, from an attached datastore, from Azure storage, or from a publicly available web location.

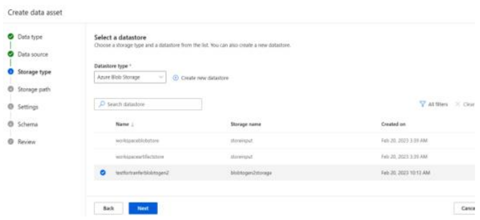
From Azure storage: Create a data asset from registered data storage services including Azure Blob Storage, Azure file share, and Azure Data Lake.

From local files: Create a data asset by uploading files from your local drive.

From SQL databases: Create a dataset from Azure SQL database and Azure PostGreSQL database.

From web files: Create a data asset from a single file located at a public web URL.

4)Selecting the storage account.


Create data asset

Data type
Data source
Storage type
Storage path
Settings
Schema
Review

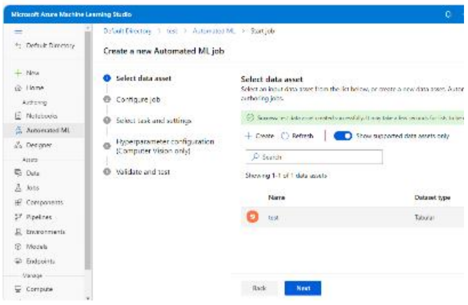
Select a datastore

Choose a storage type and a datastore from the list. You can also create a new datastore.

Datastore type *: Azure Blob Storage

Name ↓	Storage name	Created on
workspaceblobstore	storeinput	Feb 20, 2023 3:39 AM
workspaceartifactstore	storeinput	Feb 20, 2023 3:39 AM
testfortranferblobtogen2	blobtogen2storage	Feb 20, 2023 10:13 AM

5)Selecting the Data set file.


Create a new Automated ML job

Select data asset
Configure job
Select task and settings
Hyperparameter configuration (Computer Vision only)
Validate and test

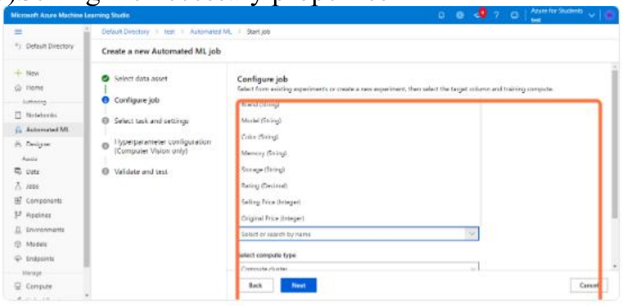
Select data asset

Select an input data asset from the list below, or create a new data asset. Auto authoring jobs.

Showing 1-1 of 1 data assets

Name	Dataset type
test	Tabular

6)Setting the necessary properties


Create a new Automated ML job

Select data asset
Configure job
Select task and settings
Hyperparameter configuration (Computer Vision only)
Validate and test

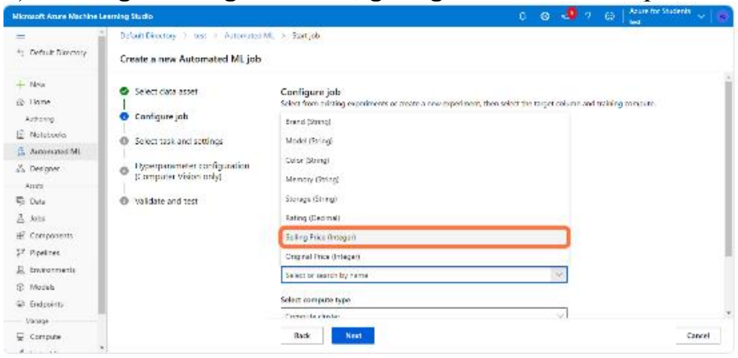
Configure job

Select from existing experiments or create a new experiment, then select the target column and training compute.

Brand (String)
Model (String)
Color (String)
Memory (String)
Storage (String)
Rating (Decimal)
Selling Price (Integer)
Original Price (Integer)

Select compute type: Compute cluster

7)Defining the Target column, giving a name to the experiment, and selecting the compute cluster.


Create a new Automated ML job

Select data asset
Configure job
Select task and settings
Hyperparameter configuration (Computer Vision only)
Validate and test

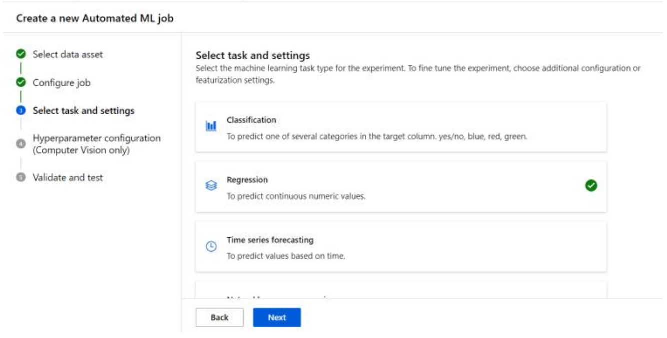
Configure job

Select from existing experiments or create a new experiment, then select the target column and training compute.

Brand (String)
Model (String)
Color (String)
Memory (String)
Storage (String)
Rating (Decimal)
Selling Price (Integer)
Original Price (Integer)

Select compute type: Compute cluster

8)Seleting the task for the AutoML of the Azure Machine learning studio


Create a new Automated ML job

- Select data asset
- Configure job
- Select task and settings
- Hyperparameter configuration (Computer Vision only)
- Validate and test

Select task and settings
Select the machine learning task type for the experiment. To fine tune the experiment, choose additional configuration or featurization settings.

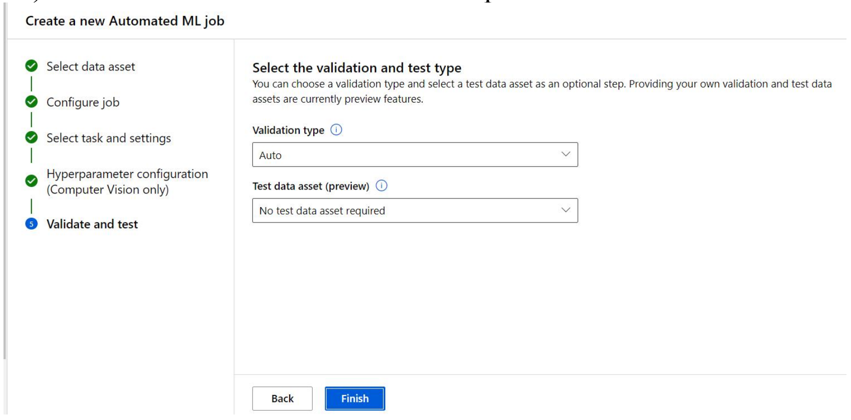
Classification
To predict one of several categories in the target column. yes/no, blue, red, green.

Regression ✓
To predict continuous numeric values.

Time series forecasting
To predict values based on time.

Back Next

12)Select the Validation and if available provide the test dataset


Create a new Automated ML job

- Select data asset
- Configure job
- Select task and settings
- Hyperparameter configuration (Computer Vision only)
- Validate and test

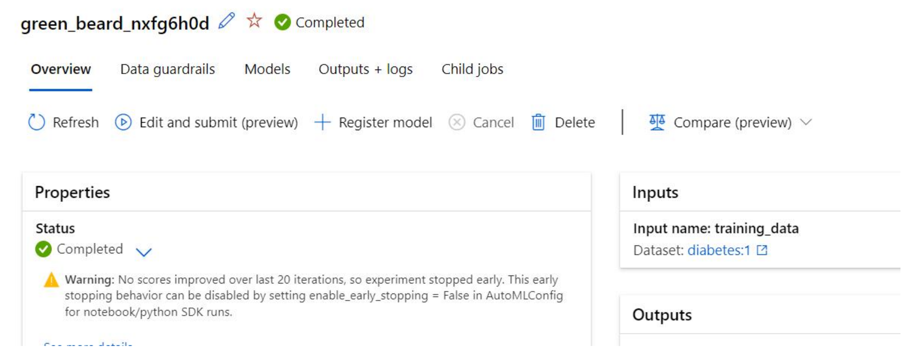
Select the validation and test type
You can choose a validation type and select a test data asset as an optional step. Providing your own validation and test data assets are currently preview features.

Validation type: Auto

Test data asset (preview): No test data asset required

Back Finish

13) Once the experiment gets completed we get a Completed notification(The average time required for a model to gets trained is 45-50 minutes)


green_beard_nxfg6h0d ☆ Completed

Overview | Data guardrails | Models | Outputs + logs | Child jobs

Refresh | Edit and submit (preview) | Register model | Cancel | Delete | Compare (preview)

Properties

Status
Completed

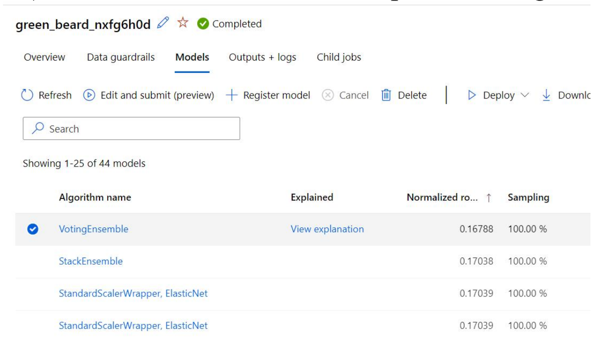
Warning: No scores improved over last 20 iterations, so experiment stopped early. This early stopping behavior can be disabled by setting enable_early_stopping = False in AutoMLConfig for notebook/python SDK runs.

Inputs

Input name: training_data
Dataset: diabetes:1

Outputs

14) Go in the model tab in the experiment to get the best model for the experiment.


green_beard_nxfg6h0d ☆ Completed

Overview | Data guardrails | Models | Outputs + logs | Child jobs

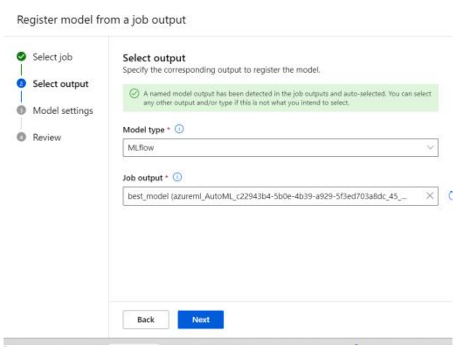
Refresh | Edit and submit (preview) | Register model | Cancel | Delete | Deploy | Downlc

Search

Showing 1-25 of 44 models

Algorithm name	Explained	Normalized ro... ↑	Sampling
VotingEnsemble	View explanation	0.16788	100.00 %
StackEnsemble		0.17038	100.00 %
StandardScalerWrapper, ElasticNet		0.17039	100.00 %
StandardScalerWrapper, ElasticNet		0.17039	100.00 %

Register the best model which you get after completion of the azure automl experiment.


Register model from a job output

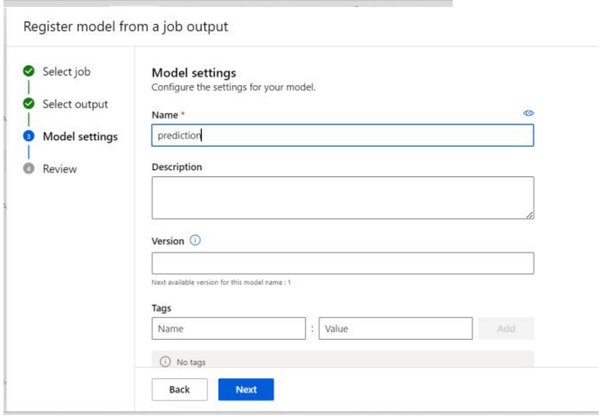
Select output

Specify the corresponding output to register the model.

A named model output has been detected in the job outputs and auto-selected. You can select any other output and/or type if this is not what you intend to select.

Model type *: MLflow

Job output *: best_model (azureml_AutoML_c22943b4-5b0e-4b39-a929-5f3ed703a8dc_45_...


Register model from a job output

Model settings

Configure the settings for your model.

Name *: prediction

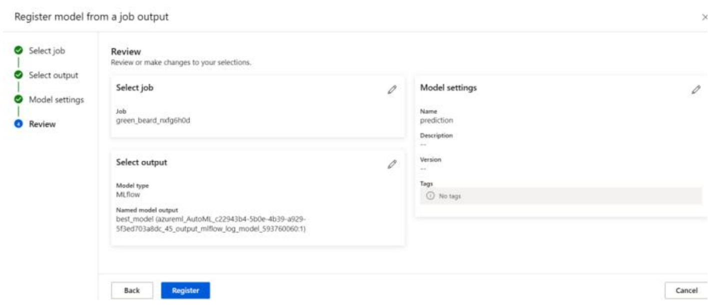
Description:

Version:

Next available version for this model name : 1

Tags: Name : Value

No tags


Register model from a job output

Review

Review or make changes to your selections.

Select job

Job: green_beard_nxfg6h0d

Select output

Model type: MLflow

Named model output: best_model (azureml_AutoML_c22943b4-5b0e-4b39-a929-5f3ed703a8dc_45_output_mlflow_log_model_593760060:1)

Model settings

Name: prediction

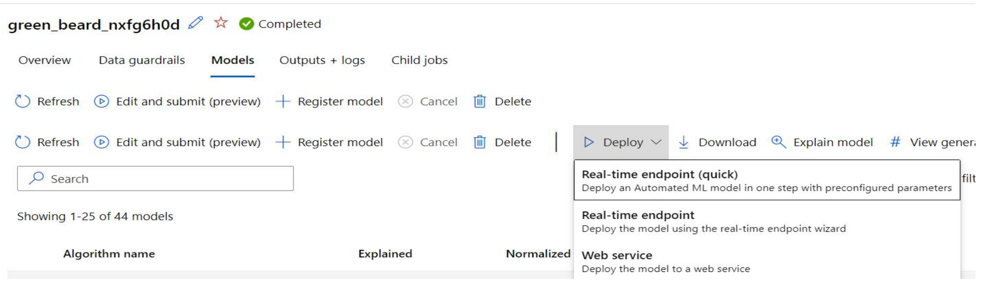
Description: --

Version: --

Tags: No tags

Model Deployment

1. Once the best the model is registerd select the model and the deployment can be done using this model.


green_beard_nxfg6h0d Completed

Showing 1-25 of 44 models

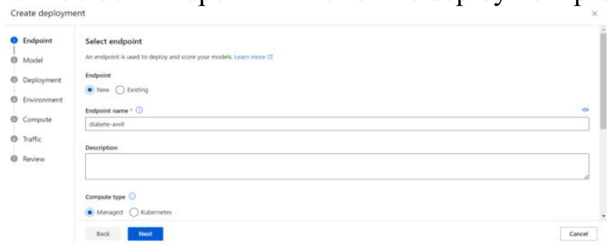
Algorithm name | Explained | Normalized

Real-time endpoint (quick)
Deploy an Automated ML model in one step with preconfigured parameters

Real-time endpoint
Deploy the model using the real-time endpoint wizard

Web service
Deploy the model to a web service

2. Provide a Endpoint Name for the deployment process and click next.


Create deployment

Select endpoint

An endpoint is used to deploy and score your models. Learn more

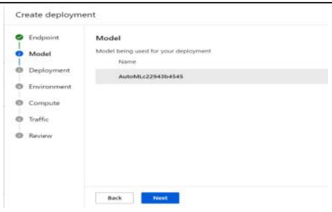
Endpoint: [✓ New] Existing

Endpoint name *: diabete-avvll

Description:

Compute type: [✓ Managed] Kubernetes

3. The selected and registed model name will appear click next.


Create deployment

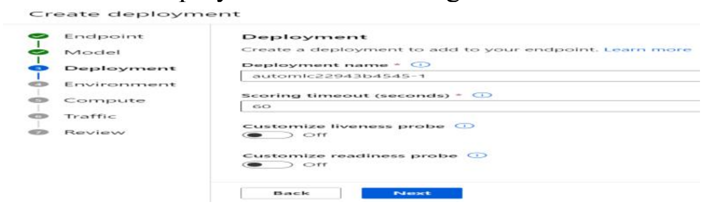
Model

Model being used for your deployment

Name

AutoMLc22943b4545

4. Provide a deployment Name scoring time and click next.


Create deployment

Deployment

Create a deployment to add to your endpoint. Learn more

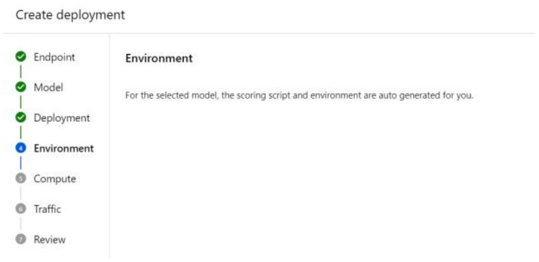
Deployment name * automlc22943b4545-1

Scoring timeout (seconds) * 60

Customize liveness probe Off

Customize readiness probe Off

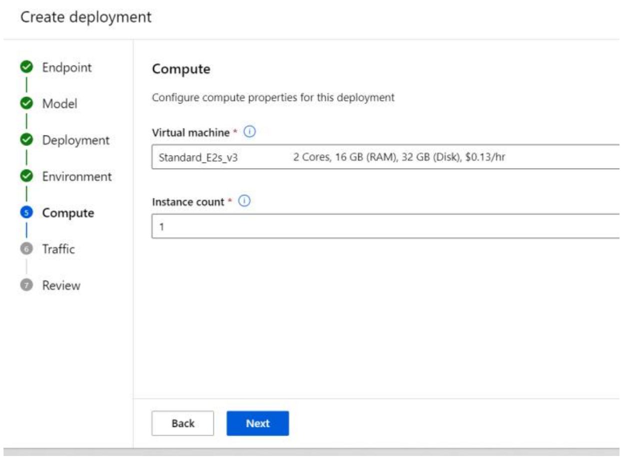
5. The environment will auto generated or else the environment can be generated using the conda env file which can be downloaded from the artifact of the best model registered and click next.


Create deployment

Environment

For the selected model, the scoring script and environment are auto generated for you.

6. Select the Virtual Machine plan as per the need and provide the number of instance count needed and click next.


Create deployment

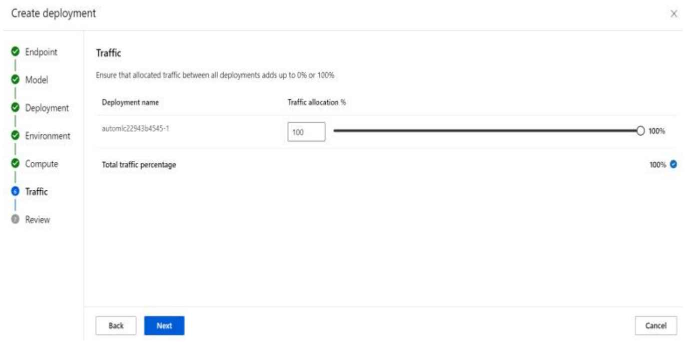
Compute

Configure compute properties for this deployment

Virtual machine * Standard_E2s_v3 2 Cores, 16 GB (RAM), 32 GB (Disk), $0.13/hr

Instance count * 1

7. Allocated the amount of traffic it must be minimum 25 % and click next.

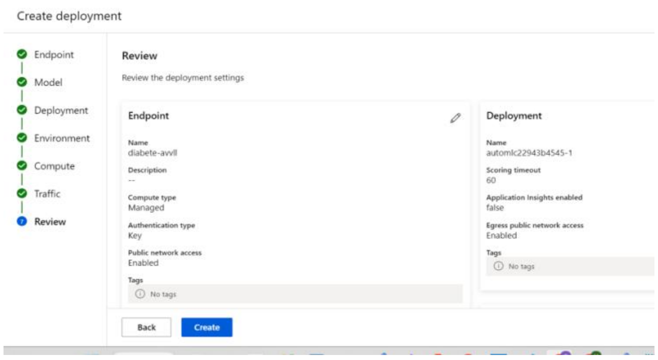

Create deployment

Traffic

Ensure that allocated traffic between all deployments adds up to 0% or 100%

Deployment name	Traffic allocation %
automlc22943b4545-1	100 100%
Total traffic percentage	100%

8. Review all the details provided and click create.


Create deployment

Review

Review the deployment settings

Endpoint

Name
diabete-avvll

Description
--

Compute type
Managed

Authentication type
Key

Public network access
Enabled

Tags
No tags

Deployment

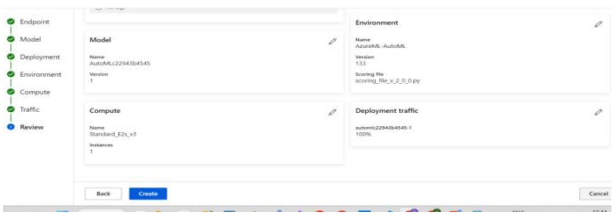
Name
automlc22943b4545-1

Scoring timeout
60

Application Insights enabled
false

Egress public network access
Enabled

Tags
No tags


Model

Name
AutoMLc22943b4545

Version
1

Environment

Name
AzureML-AutoML

Version
133

Scoring file
scoring_file_v_2_0_0.py

Compute

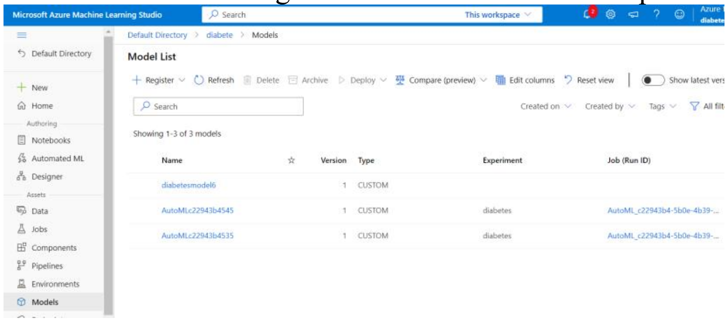
Name
Standard_E2s_v3

Instances
1

Deployment traffic

automlc22943b4545-1
100%

9. Once the model is registered it can be viewd and explore in the Models option.


Microsoft Azure Machine Learning Studio

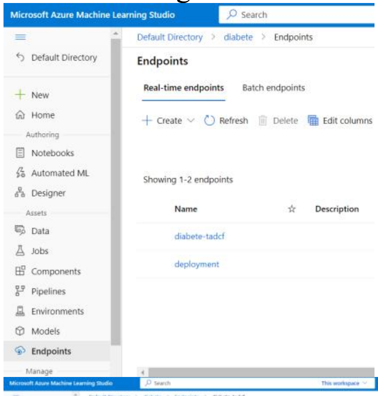
Default Directory > diabete > Models

Model List

Showing 1-3 of 3 models

Name	Version	Type	Experiment	Job (Run ID)
diabetesmodel6	1	CUSTOM		
AutoMLc22943b4545	1	CUSTOM	diabetes	AutoML_c22943b4-5b0e-4b39-...
AutoMLc22943b4535	1	CUSTOM	diabetes	AutoML_c22943b4-5b0e-4b39-...

10. Once the model is deployed successfully an end will be created and this end point can be used to deploy the model using web services.


Microsoft Azure Machine Learning Studio

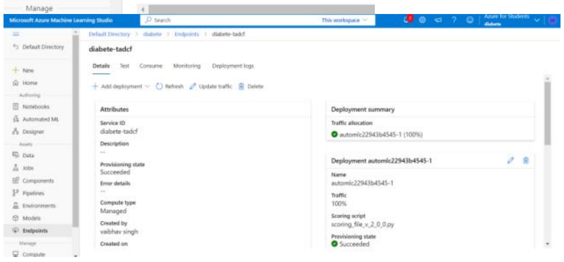
Default Directory > diabete > Endpoints

Endpoints

Real-time endpoints | Batch endpoints

Showing 1-2 endpoints

Name	Description
diabete-tadcf	
deployment	


Microsoft Azure Machine Learning Studio

Default Directory > diabete > Endpoints > diabete-tadcf

diabete-tadcf

Attributes

Service ID
diabete-tadcf

Description
--

Provisioning state
Succeeded

Error details
--

Compute type
Managed

Created by
vaibhav singh

Created on

Deployment summary

Traffic allocation
automlc22943b4545-1 (100%)

Deployment automlc22943b4545-1

Name
automlc22943b4545-1

Traffic
100%

Scoring script
scoring_file_v_2_0_0.py

Provisioning state
Succeeded

Sample Consume code:-

```
import urllib.request
import json
import os
import ssl

def allowSelfSignedHttps(allowed):
    # bypass the server certificate verification on client side
    if allowed and not os.environ.get('PYTHONHTTPSVERIFY', '') and getattr(ssl,
'_create_unverified_context', None):
        ssl._create_default_https_context = ssl._create_unverified_context

allowSelfSignedHttps(True) # this line is needed if you use self-signed certificate in your scoring service.
# Request data goes here
# The example below assumes JSON formatting which may be updated
# depending on the format your endpoint expects.
# More information can be found here:
# https://docs.microsoft.com/azure/machine-learning/how-to-deploy-advanced-entry-script
data = {
  "Inputs": {
    "data": [
      {
        "Brand": 0,
        "Model": 0,
        "Color": 0,
        "Memory": 0.0,
        "Storage": 0.0,
        "Rating": 0,
      }
    ]
  },
  "GlobalParameters": 0.0
}
body = str.encode(json.dumps(data))
url = 'https://prediction-tadcf.eastus.inference.ml.azure.com/score'
# Replace this with the primary/secondary key or AMLToken for the endpoint
api_key = ''
if not api_key:
    raise Exception("A key should be provided to invoke the endpoint")
# The azureml-model-deployment header will force the request to go to a specific deployment.
# Remove this header to have the request observe the endpoint traffic rules
headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key), 'azureml-model-deployment':
'automlc22943b4545-1' }
req = urllib.request.Request(url, body, headers)
try:
    response = urllib.request.urlopen(req)
    result = response.read()
    print(result)
except urllib.error.HTTPError as error:
    print("The request failed with status code: " + str(error.code))
    # Print the headers - they include the requert ID and the timestamp, which are useful for debugging the
failure
    print(error.info())
   print(error.read().decode("utf8", 'ignore'))
```

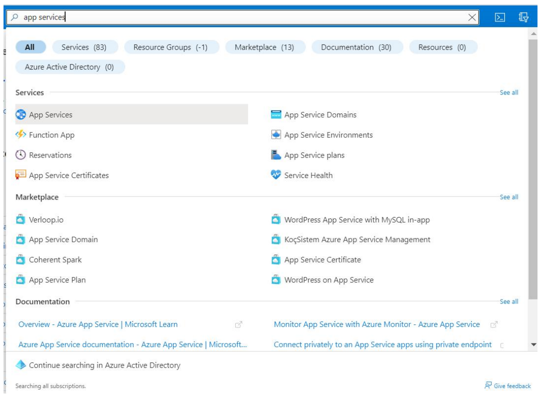
# Step 5:- a) Create a web app in Azure

Sign in to the Azure portal and follow these steps to create your Azure App Service resources.

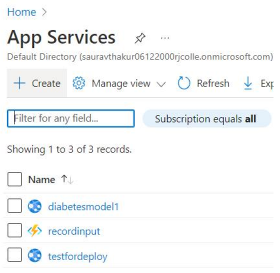
In the Azure portal:

Enter app services in the search bar at the top of the Azure portal.

Select the item labeled App Services under the under Services heading on the menu that appears below the search bar.


app services
All
Services (83)
Resource Groups (-1)
Marketplace (13)
Documentation (30)
Resources (0)
Azure Active Directory (0)
Services
See all
App Services
Function App
Reservations
App Service Certificates
App Service Domains
App Service Environments
App Service plans
Service Health
Marketplace
See all
Verloop.io
App Service Domain
Coherent Spark
App Service Plan
WordPress App Service with MySQL in-app
KoçSistem Azure App Service Management
App Service Certificate
WordPress on App Service
Documentation
See all
Overview - Azure App Service | Microsoft Learn
Azure App Service documentation - Azure App Service | Microsoft...
Monitor App Service with Azure Monitor - Azure App Service
Connect privately to an App Service apps using private endpoint
Continue searching in Azure Active Directory
Searching all subscriptions.
Give feedback

On the **App Services** page, select **+ Create**


Home >

App Services

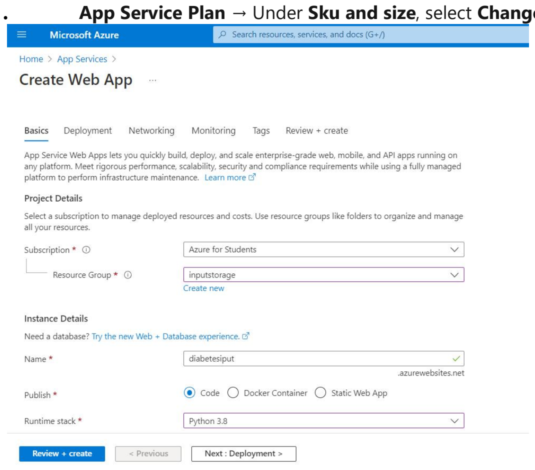
Default Directory (sauravthakur06122000rjcolle.onmicrosoft.com)

Showing 1 to 3 of 3 records.

Name
diabetesmodel1
recordinput
testfordeploy

On the **Create Web App** page, fill out the form as follows.

1. **Resource Group** → Select **Create new** and use a name of msdocs-python-webapp-quickstart.
2. **Name** → msdocs-python-webapp-quickstart-XYZ where XYZ is any three random characters. This name must be unique across Azure.
3. **Runtime stack** → **Python 3.8**.
4. **Region** → Any Azure region near you.
5. **App Service Plan** → Under **Sku and size**, select **Change size** to select a different App Service plan.



Create Web App

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. Learn more

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Azure for Students

Resource Group * : inputstorage

Create new

Instance Details

Need a database? Try the new Web + Database experience.

Name * : diabetesiput .azurewebsites.net

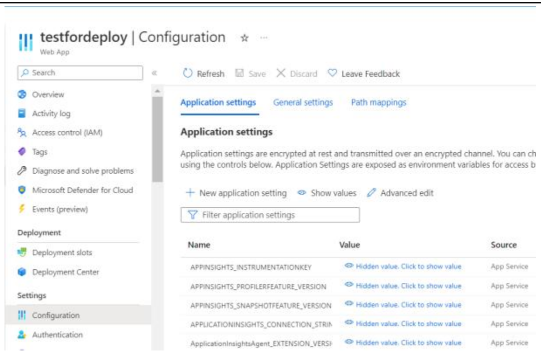
Publish * : [✓ Code] Docker Container Static Web App

Runtime stack * : Python 3.8

Step 5 :- b) Deploy your application code to Azure

On the page for the web app in the Azure portal:

1. Select **Configuration** under the **Settings** header in the left toolbar to bring up the Application settings.
2. Under **Application settings**, select **New application setting**.


testfordeploy | Configuration

Web App

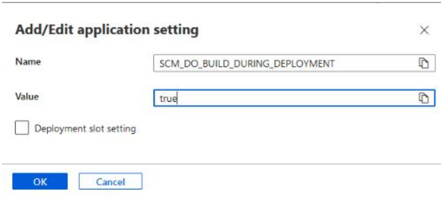
Application settings

Application settings are encrypted at rest and transmitted over an encrypted channel. You can ch
using the controls below. Application Settings are exposed as environment variables for access b

Name	Value	Source
APPINSIGHTS_INSTRUMENTATIONKEY	Hidden value. Click to show value	App Service
APPINSIGHTS_PROFILERFEATURE_VERSION	Hidden value. Click to show value	App Service
APPINSIGHTS_SNAPSHOTFEATURE_VERSION	Hidden value. Click to show value	App Service
APPLICATIONINSIGHTS_CONNECTION_STRIN	Hidden value. Click to show value	App Service
ApplicationInsightsAgent_EXTENSION_VERSI	Hidden value. Click to show value	App Service

Using the dialog, enter a new setting with:

Name → *SCM_DO_BUILD_DURING_DEPLOYMENT*
Value → *true*


Add/Edit application setting

Name: SCM_DO_BUILD_DURING_DEPLOYMENT

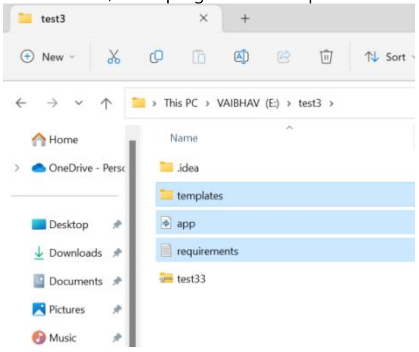
Value: true

Deployment slot setting

Select the **Save** to save your settings.

Step 5:- c) I)Create a ZIP file of your application

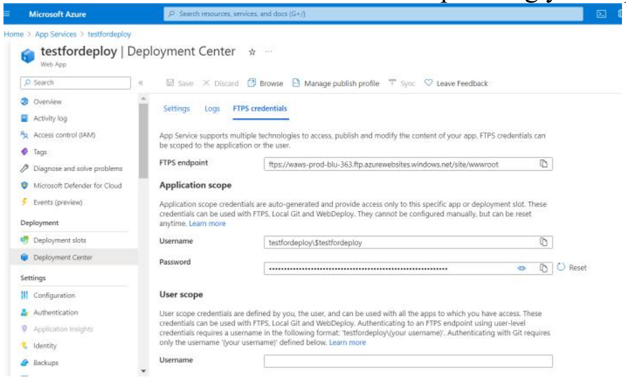
On Windows, use a program like 7-Zip to create a ZIP file needed to deploy the application.



Step 5:- c) II)Upload the ZIP file to Azure

To use Postman to upload your ZIP file to Azure, you will need the deployment username and password for your App Service. These credentials can be obtained from the Azure portal.

1. On the page for the web app, select Deployment center from the menu on the left side of the page.
2. Select the FTPS credentials tab.
3. The Username and Password are shown under the Application scope heading. For zip file deployments, only use the part of the username after the \ character that starts with a $, for example $testfordeploy. These credentials will be needed when uploading your zip file with Postman.


testfordeploy | Deployment Center

Web App

App Service supports multiple technologies to access, publish and modify the content of your app. FTPS credentials can be scoped to the application or the user.

FTPS endpoint: ftps://waws-prod-blu-363.ftp.azurewebsites.windows.net/site/wwwroot

Application scope

Application scope credentials are auto-generated and provide access only to this specific app or deployment slot. These credentials can be used with FTPS, Local Git and WebDeploy. They cannot be configured manually, but can be reset anytime. Learn more

Username: testfordeploy\$testfordeploy

Password: ••

User scope

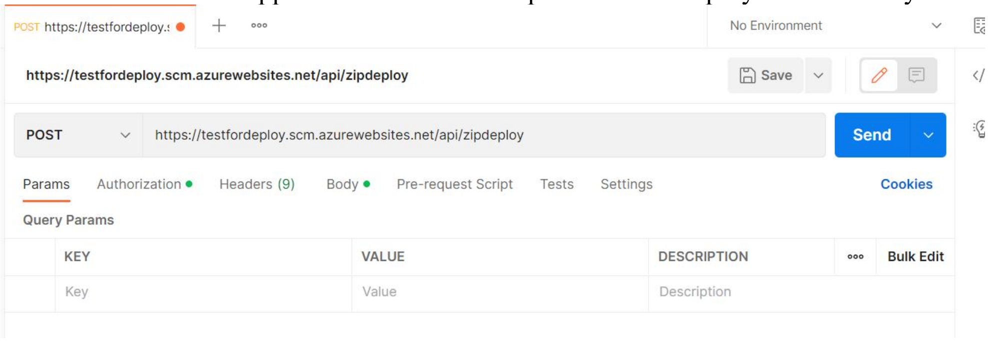
User scope credentials are defined by you, the user, and can be used with all the apps to which you have access. These credentials can be used with FTPS, Local Git and WebDeploy. Authenticating to an FTPS endpoint using user-level credentials requires a username in the following format: 'testfordeploy\(your username)'. Authenticating with Git requires only the username '(your username)' defined below. Learn more

Username:

Step 5:- c) III) Postman

In Postman, upload your file using the following steps.
Click on the plus (+) sign icon to create a new request.

a) Select POST for the request type.
b) Enter the url https://\<app-name\>.scm.azurewebsites.net/api/zipdeploy where <app-name> is the name of the web app. This URL is the endpoint used to deploy a ZIP file to your Azure service.


POST https://testfordeploy.scm.azurewebsites.net/api/zipdeploy

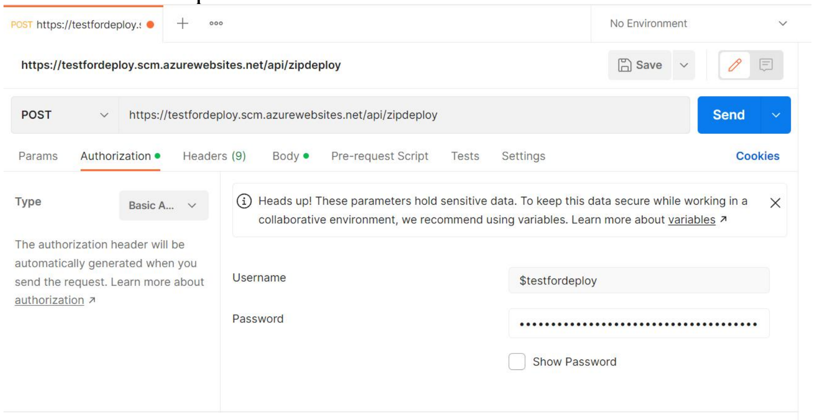
Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings | Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

In the Authorization tab:

a) Set the Type to Basic.
b) Enter the deployment username and password obtained from the Azure portal above. Be sure to only use the portion of the username after the \ character that starts with a $.


POST https://testfordeploy.scm.azurewebsites.net/api/zipdeploy

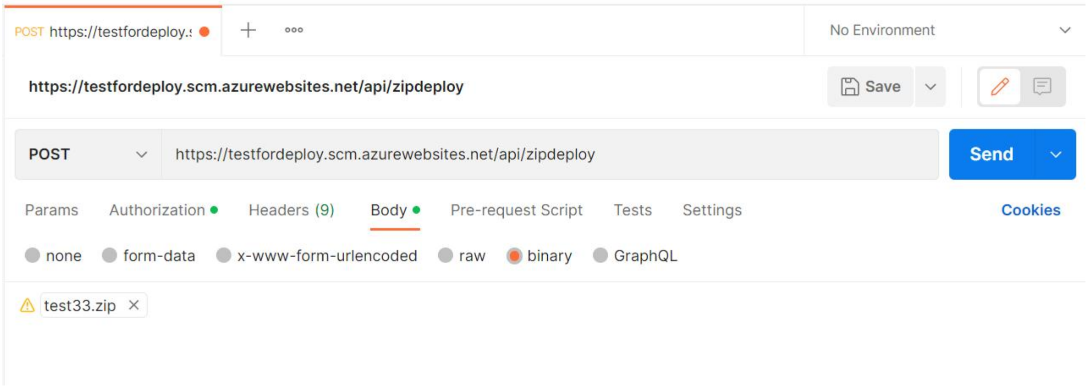
Type: Basic A...

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Username: $testfordeploy
Password: ••
Show Password

In the Body tab:
1. Select binary as the content type.
2. Use the Select File button to select your zip file.


POST https://testfordeploy.scm.azurewebsites.net/api/zipdeploy

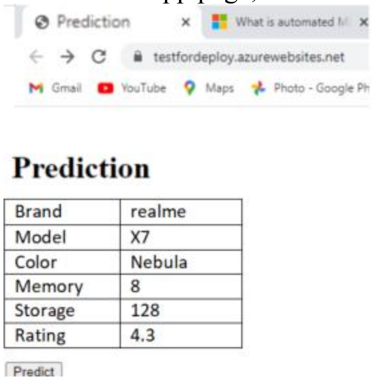
Body: none | form-data | x-www-form-urlencoded | raw | [✓ binary] | GraphQL

test33.zip

The filename of the file to be uploaded will be shown in the Body section.
Select the Send button to upload your zip file to Azure.
Depending on your network bandwidth, files usually take between 10 and 30 seconds to upload to Azure

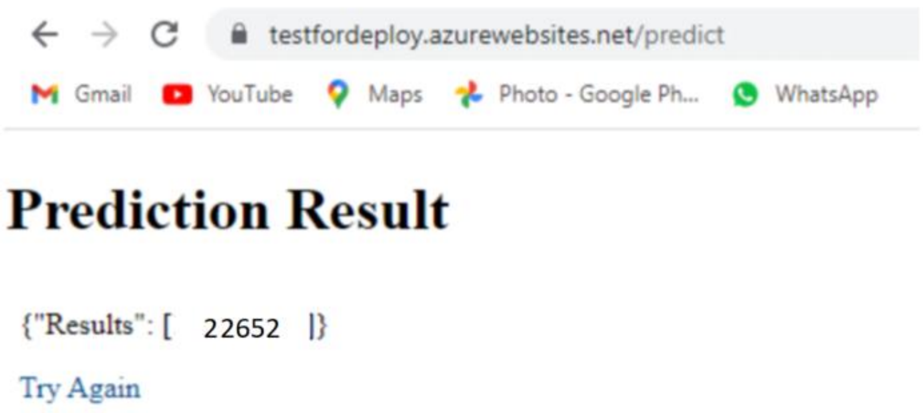
Step 5:- d) Browse to the app

Browse to the deployed application in your web browser at the URL http://<app-name>.azurewebsites.net. If you see a default app page, wait a minute and refresh the browser.



Prediction

Brand	realme
Model	X7
Color	Nebula
Memory	8
Storage	128
Rating	4.3



Prediction Result

```
{"Results": [   22652   ]}
```

app.py Code:-

```
from flask import Flask, render_template, request, jsonify
import json
import urllib.request

app = Flask(__name__)
url = "https://prediction-tadcf.eastus.inference.ml.azure.com/score"
api_key = '5l3xsfu0X0Hy4cww3J8rColjXRu0eKta'
@app.route('/')
def home():
   return render_template('index.html')
@app.route('/predict', methods=['POST','GET'])
def predict():
       data = { "Inputs" :
               {
           "data": [
           {
              "Brand": request.form['Brand'],
              "Model": request.form['Model'"],
              "Color": request.form['Color'],
              "Memory": request.form['Memory'],
              "Storage": request.form['Storage'],
              "Rating": request.form['Rating'],

           }
           ]
       }
   }
```

```
    headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key),'azureml-model-deployment': 'automlc22943b4545-1'}
    body = str.encode(json.dumps(data))
    req = urllib.request.Request(url, body, headers)

    try:
        response = urllib.request.urlopen(req)
        result = response.read().decode()
        return render_template('predict.html', prediction=result)
    except:
        error_msg = response.json().get('message', 'Prediction failed')
        print(f"Prediction failed: {error_msg}")
        return render_template('error.html', message='Prediction failed')

if __name__ == "__main__":
    app.run(debug=True)
```

Predict.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="html">
    <title> Prediction Result</title>
  </head>
  <body>
    <h1> Prediction Result</h1>
    <p>{{ prediction }}</p>
    <p><a href="/">Try Again</a></p>
  </body>
</html>
```

Error.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Error </title>
</head>
<body>
    <h1>Error Page</h1>
    <p>{{ error }}</p>
</body>
</html>
```

9.CONCLUSION

In conclusion, this project aimed to develop a machine learning model for predicting the prices of mobile phones based on their various features. The project utilized Azure AutoML, UIpath, Azure Web Service, and Postman to collect data, preprocess it, select an appropriate machine learning algorithm, train and evaluate the model, deploy it, and monitor its performance.

The results of the project showed that the machine learning model developed using Azure AutoML had good accuracy in predicting the prices of mobile phones. The model was able to effectively utilize the various features of the mobile phones to accurately predict their prices, making it a valuable tool for users interested in buying or selling mobile phones.

The project also demonstrated the effectiveness of Azure AutoML in automating the machine learning model selection and training process, making it accessible to users without extensive coding or data science expertise. The use of Azure Web Service and Postman also allowed for easy deployment and integration of the model with web applications or APIs.

Overall, this project highlights the importance of careful data preprocessing, model selection and evaluation, and effective deployment and monitoring strategies for developing accurate and reliable machine learning models. The developed model has practical applications in the mobile phone market and could be extended to other similar industries for price prediction.

.

10.FUTURE SCOPE

There are several areas of future work that could be explored to improve upon the current project:

- Incorporating more data sources: The current project utilized data from a single source (Flipkart), and incorporating data from other sources such as Amazon or Best Buy could lead to a more comprehensive dataset and potentially more accurate predictions.
- Exploring more feature engineering techniques: While the current project utilized basic feature engineering techniques such as one-hot encoding and standardization, more advanced techniques such as feature selection or dimensionality reduction could be explored to improve the model's performance.
- Investigating other machine learning algorithms: While the current project utilized the best performing algorithm from Azure AutoML, other algorithms such as decision trees, support vector machines, or neural networks could be investigated to see if they could improve the model's performance.
- Incorporating time-series data: As mobile phone prices can change rapidly, incorporating time-series data into the model could improve its accuracy in predicting prices over time.
- Developing a user-friendly interface: While the current project utilized Postman for deployment, developing a more user-friendly interface for users to input mobile phone features and receive predicted prices could increase its practicality and usability.

11.REFERENCES

1. For data scraping with UiPath: "UiPath Web Automation - Scraping Data" (https://www.youtube.com/watch?v=rrc-5B5jBp8)
2. For training machine learning models with Azure AutoML: "Getting started with automated machine learning in Azure Machine Learning" (https://docs.microsoft.com/en-us/azure/machine-learning/how-to-auto-train-models)
3. For hosting web services on Azure: "Create and deploy a REST service to Azure" (https://docs.microsoft.com/en-us/azure/app-service/app-service-web-tutorial-rest-api)
4. For deploying trained models with Postman: "Deploying a Machine Learning Model as a REST API using Flask and Postman" (https://towardsdatascience.com/deploying-a-machine-learning-model-as-a-rest-api-using-flask-and-postman-4b9dd9b62e9)
5. For evaluating machine learning models: "A Comprehensive Guide to Machine Learning Model Evaluation Metrics" (https://towardsdatascience.com/a-comprehensive-guide-to-machine-learning-model-evaluation-metrics-8cefbae4c12e)
6. For feature engineering: "The Ultimate Guide to Feature Engineering" (https://towardsdatascience.com/the-ultimate-guide-to-feature-engineering-357b537d3c5c)
7. For model selection: "A Beginner's Guide to Machine Learning Model Selection" (https://towardsdatascience.com/a-beginners-guide-to-machine-learning-model-selection-2c966c70a49f)
8. For model interpretation: "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable" (https://towardsdatascience.com/interpretable-machine-learning-a-guide-for-making-black-box-models-explainable-8742c4f3dd1f)
9. Gaurav, K., & Kumawat, N. (2019). Mobile Price Prediction using Machine Learning. International Journal of Innovative Technology and Exploring Engineering, 8(7), 255-258.
10. Zhang, S., Gao, S., Xu, Y., Wu, Q., & Li, X. (2020). Mobile phone price prediction based on improved XGBoost algorithm. Multimedia Tools and Applications, 79(3), 2443-2460.
11. Liu, J., Li, L., Guo, J., & Li, Y. (2020). An improved gradient boosting machine algorithm for mobile phone price prediction. International Journal of Machine Learning and Cybernetics, 11(1), 115-127.
12. Kaur, P., & Saini, G. K. (2021). Mobile price prediction using machine learning techniques. International Journal of Engineering and Advanced Technology, 10(1), 3789-3795.