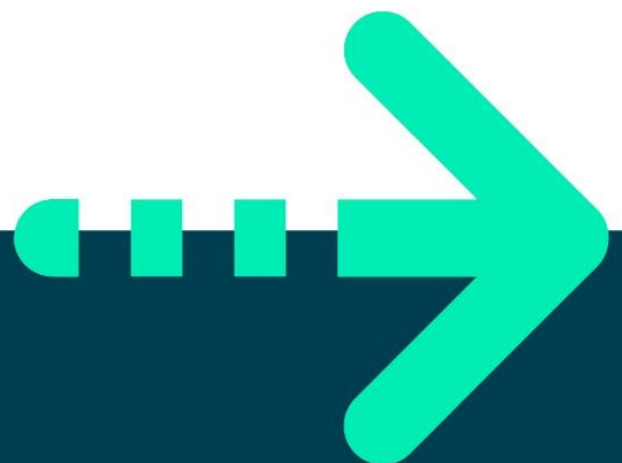




Implementing a Data Science Solution with Azure

By Sadique Mohammed – DFEDATA4





Part 1: Problem Domain Understanding

Question 1

Given the nature of the business you have been asked to describe the benefits of implementing the data science solution in the cloud compared to using on premise/hybrid data solutions.

The evaluation criteria should consider the following:

- security
- compliance
- scalability
- efficiency
- reliability
- fidelity
- flexibility
- portability

Please see table below for benefits of using a cloud data solution compared to on premise/hybrid data solution.

Cloud	On Premise/Hybrid
<ul style="list-style-type: none">• On-demand scalability• Cost efficiency• Bundled capabilities such as IAM and analytics• Security• System uptime and availability	<ul style="list-style-type: none">• Complete control over the tech stack• Local speed and performance• Governance and regulatory compliance

Part 2: Data Understanding

Question 1

Complete all the tasks below in Microsoft Azure Machine Learning Studio using the dataset identified for your project and discuss what the implications of the results for at least two of the tasks on the business questions that are candidates for your data science solution.

Task 1: Generate a descriptive statistics report for the columns in your dataset

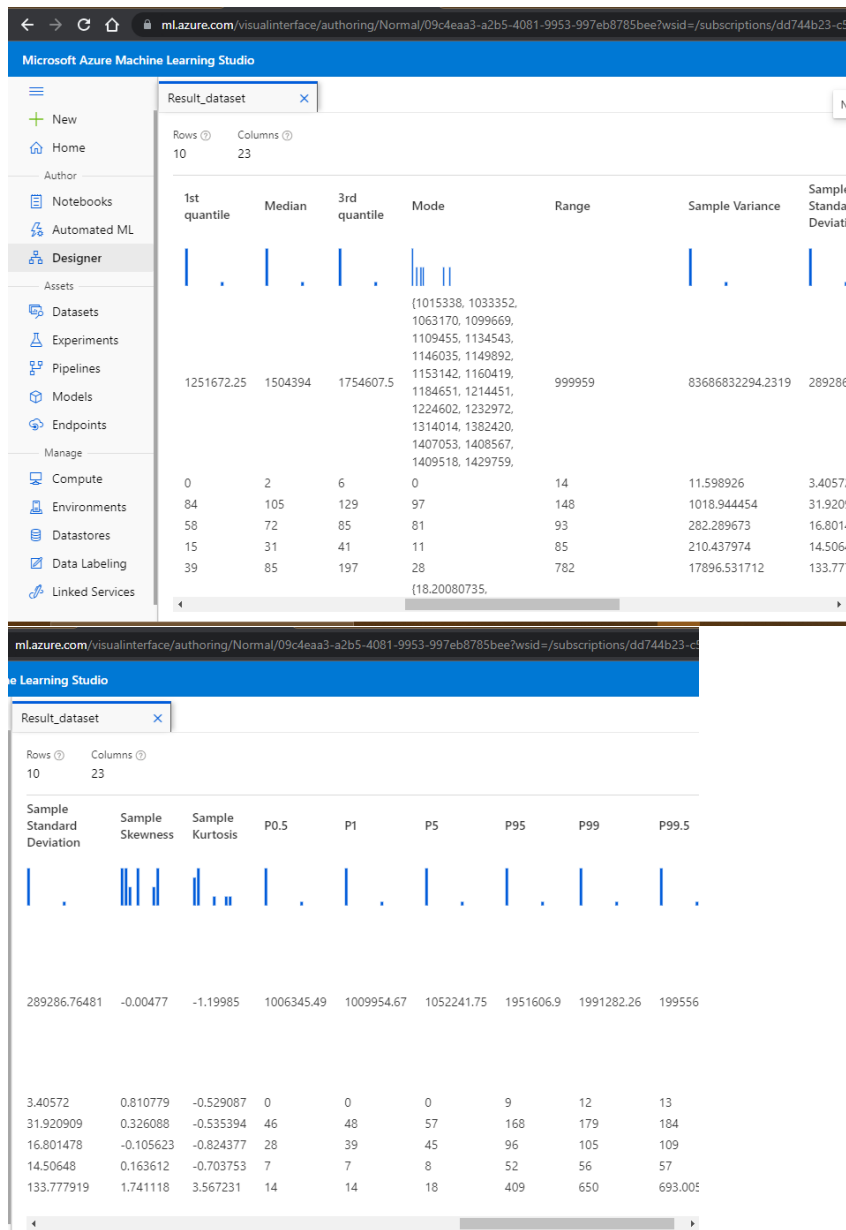
The screenshot shows the Microsoft Azure Machine Learning Studio interface. The main canvas displays a pipeline named 'Diabetes Statistics'. The pipeline starts with a 'Diabetes Dataset' node, which feeds into a 'Summarize Data' node. The 'Summarize Data' node is marked as 'Completed'. The left sidebar shows the 'Designer' tab with a search bar and a list of assets including 'Diabetes Dataset'.

The screenshot shows the 'Result_dataset' table in Microsoft Azure Machine Learning Studio. The table displays descriptive statistics for various features. The table has 10 rows and 23 columns. The features listed are PatientID, Pregnancies, PlasmaGlucose, DiastolicBloodPressure, TricepsThickness, and SerumInsulin. The statistics shown include Count, Unique Value Count, Missing Value Count, Min, Max, Mean, and Mean Deviation.

Feature	Count	Unique Value Count	Missing Value Count	Min	Max	Mean	Mean Deviation
PatientID	10000	9959	0	1000038	1999997	1502122.0827	250256.742736
Pregnancies	10000	15	0	0	14	3.2558	2.942913
PlasmaGlucose	10000	149	0	44	192	107.8502	26.074515
DiastolicBloodPressure	10000	90	0	24	117	71.2075	14.63353
TricepsThickness	10000	66	0	7	92	28.8176	12.476471
SerumInsulin	10000	620	0	14	796	139.2436	102.026558

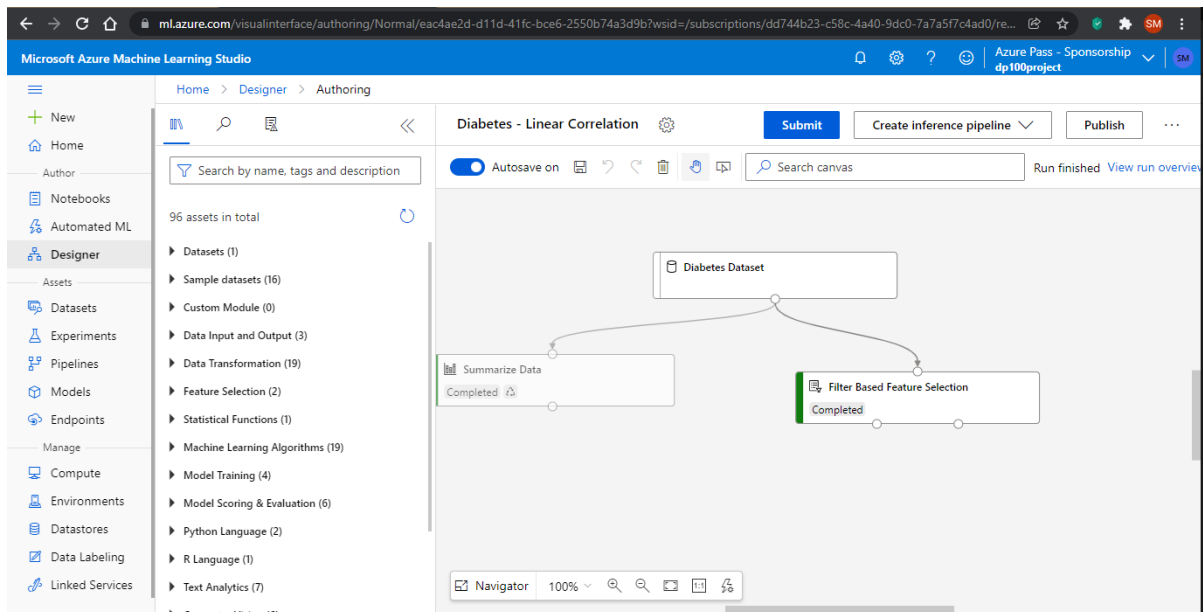
Above, you can see how I have obtained a statistical report for columns in the dataset in the Microsoft Azure Machine Learning Studio Designer.

Task 2: Calculate a single statistical measure for each column which is useful for determining central tendency, dispersion, and shape of your dataset or whatever you deem necessary.

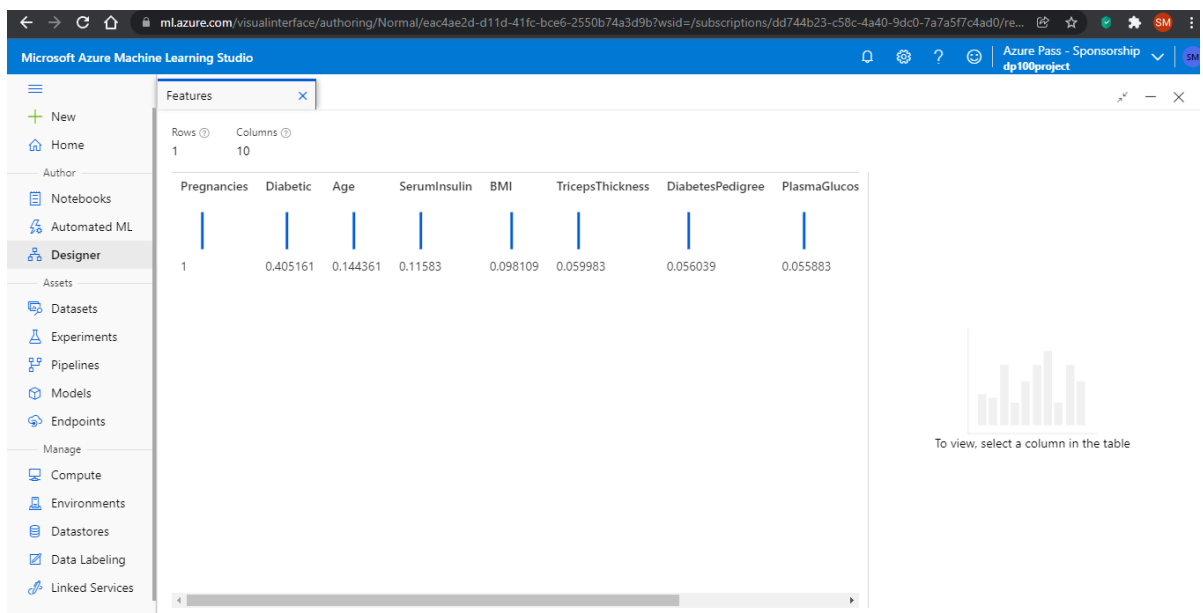


Please also see task 1 for single statistical measures for each column of dataset.

Task 3: Calculate the linear correlation between column values in your dataset



Above model is used to obtain the linear correlation between column values in dataset.



As you can see from above screenshot that Diabetic, Age, Seruminsulin and BMI are all positively correlated.

Task 4: Identify any data quality issues that may impact analysis

There are currently no data quality issues with this data sets, it is a pure dataset. There are no missing values or outliers to list. Also, there are no Nulls in this dataset.

Task 5: Conduct exploratory visual analysis of the dataset and comment on at least 2 of your key findings.





From the visual analysis above, all columns in dataset have a positive skewness. The maximum BMI value is 56.03 and minimum BMI is 18.2. Also, the mean age of the sample in this data is 30.13 yrs, which is 30 yrs.

Note: Tasks may be completed by identifying and using the relevant modules in Designer.

Part 3: Data Preparation and Transformation

Question 1:

Discuss and apply the appropriate data processing techniques to address the data quality issues you identified in your exploratory analysis of the data.

No data quality issues were identified in the exploratory analysis of the data. All values in dataset are numerical and there is not any categorical values in dataset.

Question 2:

Conduct a comparative analysis of the mathematical techniques below for data normalization and apply the relevant one to your dataset.

- **Z-SCORE**

Above shows process of obtaining the Z Score.

- **min-max**

The screenshot displays the Microsoft Azure Machine Learning Studio interface. The top navigation bar includes the 'Microsoft Azure Machine Learning Studio' title, a search bar, and user information 'Azure Pass - Sponsorship dp100project'. The left sidebar shows the 'Designer' tab selected. The main canvas shows a workflow titled 'Diabetes - Z Score, min max, LogNormal & tanh'. The workflow consists of three steps: 'Diabetes Dataset', 'Select Columns in Dataset', and 'Normalize Data'. The 'Normalize Data' step is highlighted, and its configuration panel is open on the right. The 'Parameters' tab is active, showing 'Transformation method' set to 'MinMax', 'Use 0 for constant columns when checked' set to 'True', and 'Columns to transform' listed as 'Pregnancies, PlasmaGlucose, TricepsThickness, SerumInsulin, BMI, DiabetesPedigree, Age'. The 'Exclude column names' field is set to 'Diabetic'. The 'Output settings' tab is also visible.

Above shows process of obtaining MinMax.

- log-normal

This screenshot is similar to the one above, showing the same workflow in the Microsoft Azure Machine Learning Studio. However, the 'Normalize Data' step's configuration panel is now set to 'LogNormal' for the 'Transformation method'. All other parameters, including the columns to transform and the excluded column names, remain the same as in the previous screenshot.

Above shows process of obtaining log-normal

- tanh

Microsoft Azure Machine Learning Studio

Home > Designer > Authoring

Diabetes - Z Score, min max, LogNormal & tanh

Submit Create inference pipeline Publish ...

Autosave on Search canvas Run finished View run overview

Diabetes Dataset

Select Columns in Dataset Completed

Normalize Data Completed

Normalize Data

Refresh

Parameters Outputs + logs Details Metrics Child runs Images Snapshot ...

Transformation method * Tanh

Columns to transform * Edit column

Column names: Pregnancies, PlasmaGlucose, TricepsThickness, SerumInsulin, BMI, DiabetesPedigree, Age

Exclude column names: Diabetic

Output settings >

Run settings >

Comment >

Above shows process of obtaining tanh.

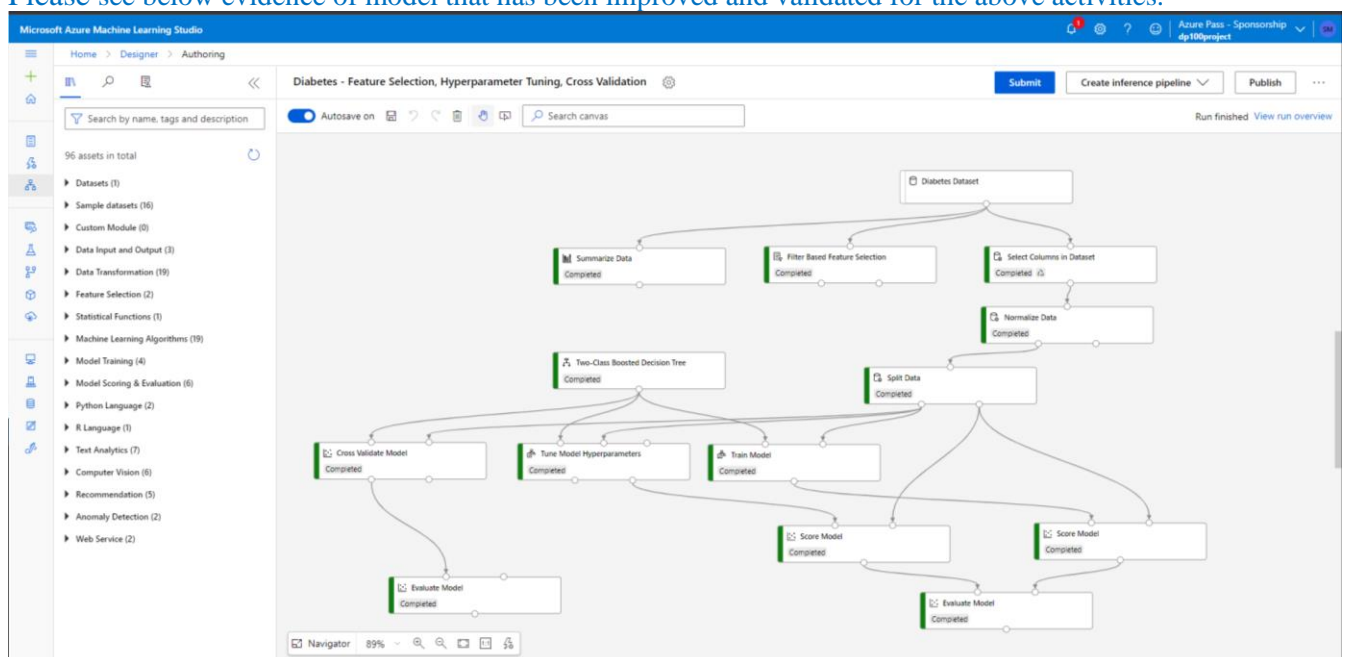
Part 4: Model Validation and Improvement

Question 1:

Discuss and evidence how your model can be improved and validated through the following activities.

- Feature Selection
- Hyperparameter Tuning
- Cross Validation

Please see below evidence of model that has been improved and validated for the above activities.



Part 5: Evaluation and Metrics

Question 1:

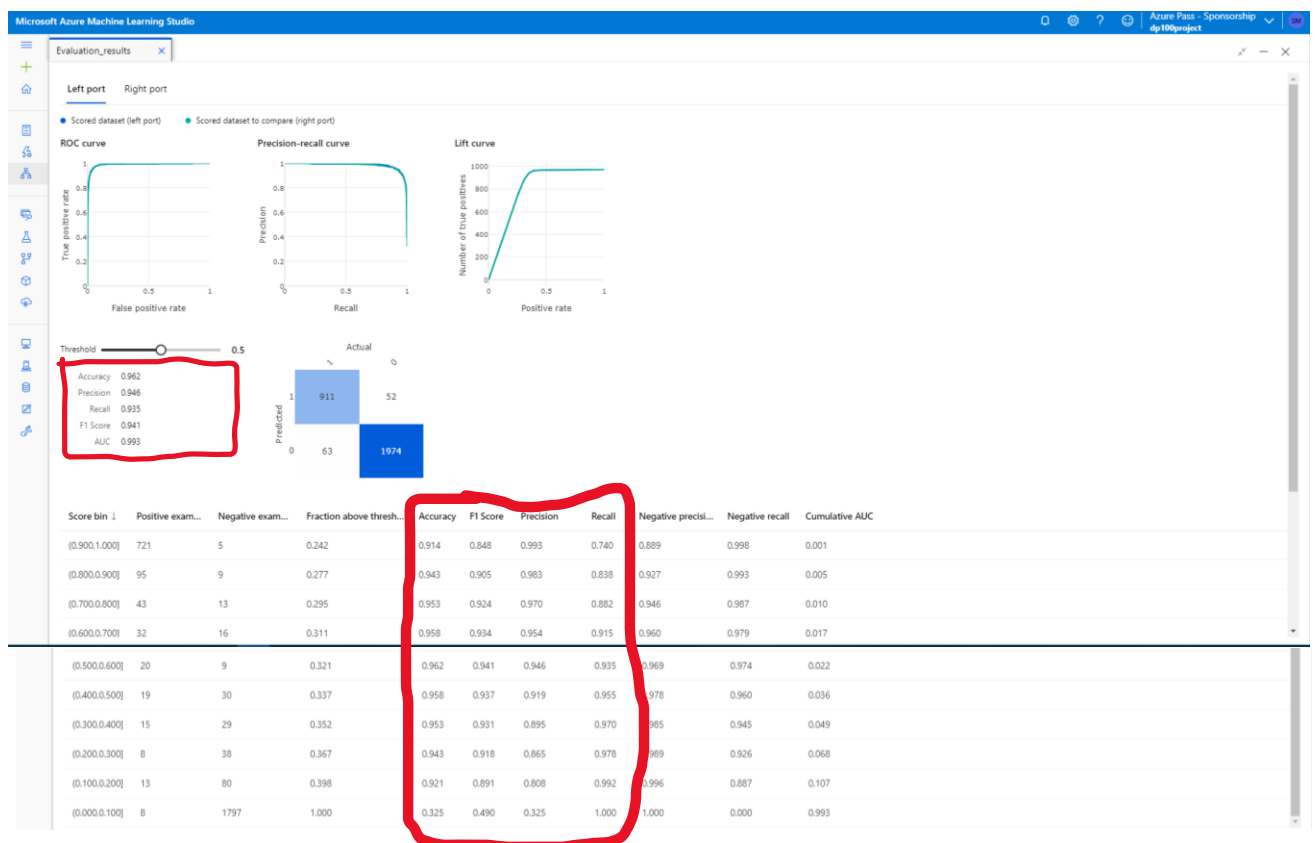
Critically evaluate the metrics for classification models below and identify which of the statistical measures are suitable for the evaluation of your data science solution. The selected metric should be used in the evaluation of your model.

Classification Metrics:

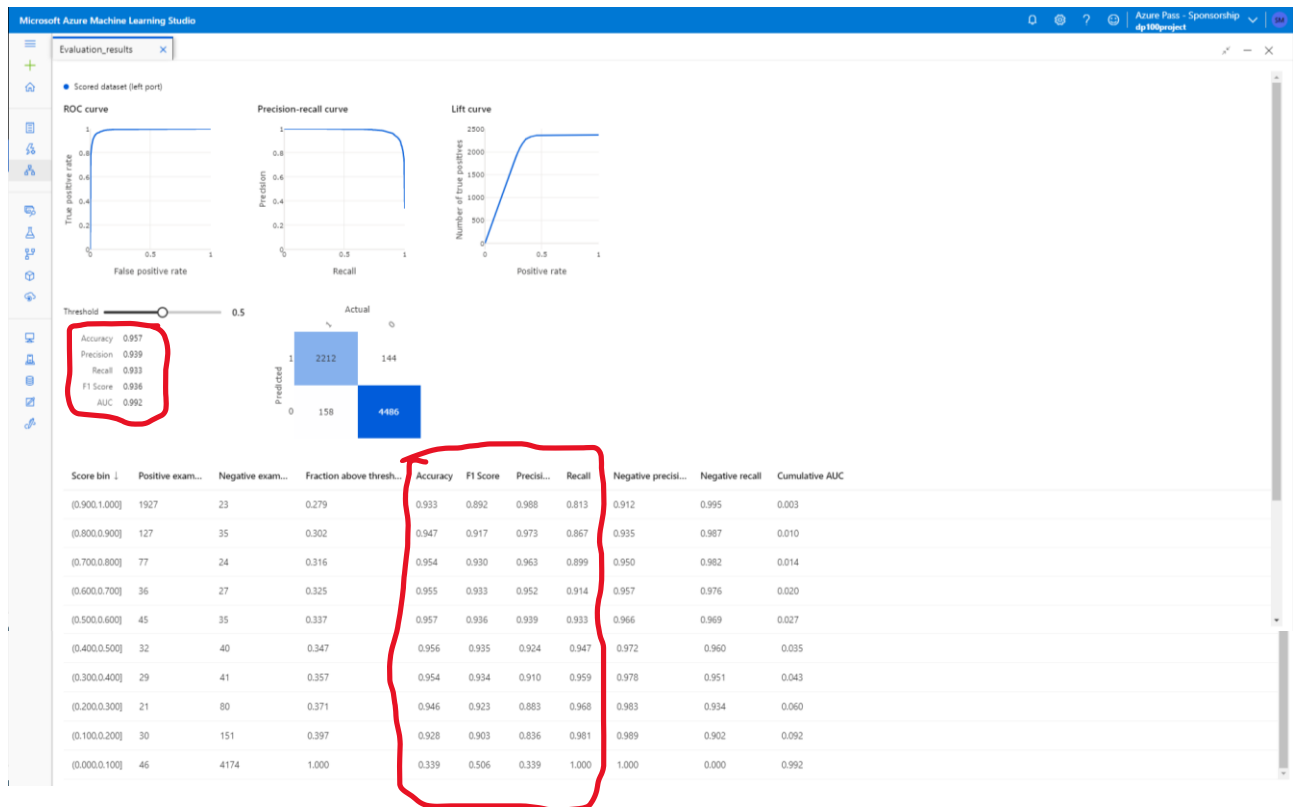
- Accuracy
- Precision
- Recall
- F1 Score

Using the designer model from Part 4, I was able to obtain the above classification metrics via the evaluate model.

Below shows a screenshot of classification metrics via the hyperparameter tuning evaluate model.



Below shows a screenshot of classification metrics via the cross validation evaluate model.

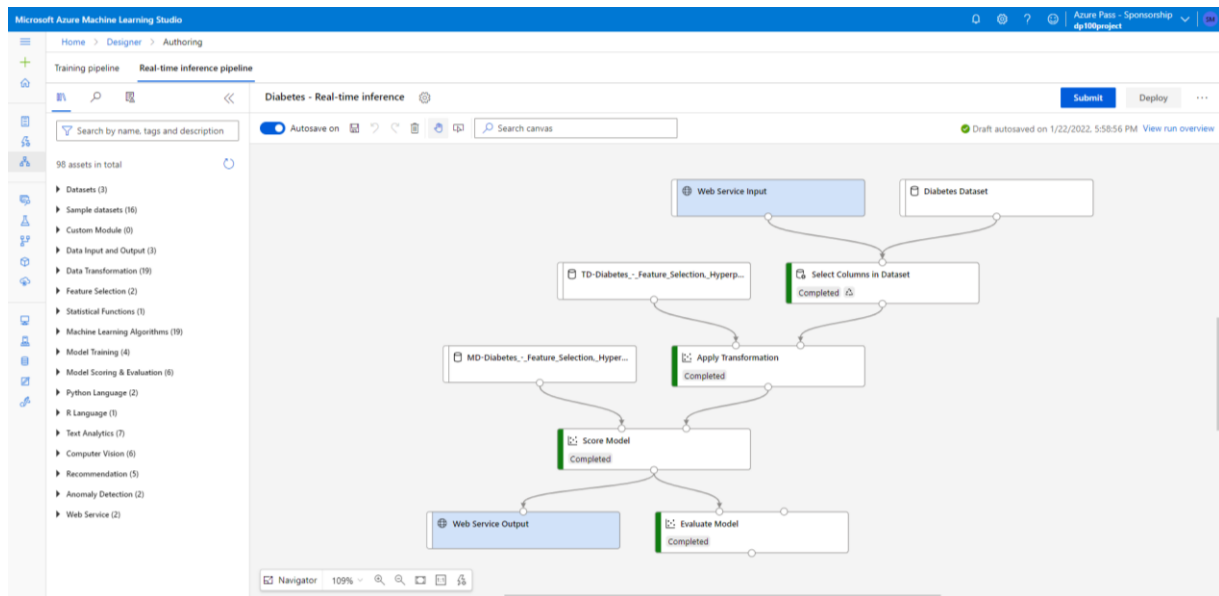


Part 6: Deployment

Question 1:

Deploy your model as a real-time inferencing service.

Below shows a model for a inference pipeline for real-time inferencing to be deployed.



Below shows deployment of Diabetes dataset as a real time inferencing service.

The screenshot shows the 'diabetesdeployment' endpoint details in Microsoft Azure Machine Learning Studio. The 'Details' tab is selected, showing the following information:

- Attributes:**
 - Service ID: diabetesdeployment
 - Description: ---
 - Deployment state: Healthy
 - Compute type: Container instance
 - Created by: Sadique Mohammed
 - Model ID: amlstudio-diabetesdeployment:1
 - Created on: 1/22/2022 6:28:36 PM
 - Last updated on: 1/22/2022 6:28:36 PM
 - Image ID: ---
 - REST endpoint: <http://0cceb6b4-5531-4c07-ba35-7617a0c5f0cf.uk.south.azurecontainer.io/score>
 - Key-based authentication enabled: true
 - Swagger URI: <http://0cceb6b4-5531-4c07-ba35-7617a0c5f0cf.uk.south.azurecontainer.io/swagger.json>
 - CPU: 0.1
- Tags:**
 - CreatedByAMLStudio: true
- Properties:**
 - LinkedPipelineDraftId: 62f4244d-e14f-4232-bc06-d2ad39c95a5d
 - LinkedPipelineRunId: 4aad36ee-3f1e-46f4-987b-2a2503204a9d
 - hasInferenceSchema: True
 - hasHttps: False

Question 2:

Write a simple test script in Jupyter Notebook to consume the service.

Please see below simple test script in Jupyter Notebook to consume the service.

The screenshot shows the Microsoft Azure Machine Learning Studio interface. On the left is a sidebar with navigation options like Home, Notebooks, Automated ML, Designer, Assets, Datasets, Experiments, Pipelines, Models, Endpoints, and Manage. The main area displays a Jupyter Notebook titled 'Diabetes-Test.ipynb'. The notebook contains a Python script that uses the 'urllib.request' module to send a POST request to a web service. The script defines a function 'allowSelfSignedHttps' to bypass certificate verification, constructs a JSON body with patient data, and sends the request. The output of the notebook shows the response from the web service, which includes predicted values for various diabetes-related metrics.

```

1 import urllib.request
2 import json
3 import os
4 import ssl
5
6 def allowSelfSignedHttps(allowed):
7     # bypass the server certificate verification on client side
8     if allowed and not os.environ.get('PYTHONHTTPSVERIFY', '') and getattr(ssl, '_create_unverified_context', None):
9         ssl._create_default_https_context = ssl._create_unverified_context
10
11 allowSelfSignedHttps(True) # this line is needed if you use self-signed certificate in your scoring service.
12
13 # Request data goes here
14 data = {
15     "Inputs": {
16         "WebServiceInput0":
17             [
18                 {
19                     "PatientID": "1354778",
20                     "Pregnancies": "5",
21                     "PlasmaGlucose": "171",
22                     "DiastolicBloodPressure": "100",
23                     "TricepsThickness": "34",
24                     "SerumInsulin": "23",
25                     "BMI": "43.5007259",
26                     "DiabetesPedigree": "1.213191354",
27                     "Age": "60",
28                     "Diabetic": "0",
29                 },
30             ],
31     },
32     "GlobalParameters": {
33     }
34 }
35
36 body = str.encode(json.dumps(data))
37
38 url = 'http://fc6eb84-5531-4c07-ba39-7617bdc5f0cf.ukouth.azurecontainer.io/score'
39 api_key = '86a1m5cV0f37uht8BMSVahp0X2ak' # Replace this with the API key for the web service
40 headers = {'Content-Type': 'application/json', 'Authorization': ('Bearer ' + api_key)}
41
42 req = urllib.request.Request(url, body, headers)
43
44 try:
45     response = urllib.request.urlopen(req)
46     result = response.read()
47     print(result)
48 except urllib.error.HTTPError as error:
49     print("The request failed with status code: " + str(error.code))
50
51 # Print the headers - they include the request ID and the timestamp, which are useful for debugging the failure
52 print(error.info())
53 print(json.loads(error.read().decode("utf8", 'ignore'))))
54
55 [2] ✓ <1 sec
56
57 b'{"Results": [{"WebServiceOutput0": [{"Pregnancies": 0.5121640657327122, "PlasmaGlucose": 1.978419756576528, "DiastolicBloodPressure": 1.7137741832356608, "TricepsThickness": 0.3572658974660305, "SerumInsulin": -0.0689730948078677, "BMI": 1.218161555749926, "DiabetesPedigree": 2.129401551452751, "Age": 2.467146668360032, "Diabetic": 0.0, "Scored Labels": 1.0, "Scored Probabilities": 0.9737454698779997}]}]}'

```

Python Script in Jupyter Notebook

```

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
data = {
    "Inputs": {
        "WebServiceInput0":
            [
                {
                    'PatientID': "1354778",
                    'Pregnancies': "5",
                    'PlasmaGlucose': "171",
                    'DiastolicBloodPressure': "100",
                    'TricepsThickness': "34",
                    'SerumInsulin': "23",
                    'BMI': "43.50972593",
                    'DiabetesPedigree': "1.213191354",
                    'Age': "60",
                    'Diabetic': "0",
                },
            ],
    },
    "GlobalParameters": {
    }
}

body = str.encode(json.dumps(data))

url = 'http://0cceb6b4-5531-4c07-ba35-
7617a0c5f0cf.uksouth.azurecontainer.io/score'
api_key = '0kAlmBsCYoF37wnhtRWNdSxAhuPdX2ak' # Replace this with the API key for t
he web service
headers = {'Content-
Type': 'application/json', 'Authorization': ('Bearer ' + api_key)}

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 request ID and the timestamp, which are
    useful for debugging the failure
    print(error.info())
    print(json.loads(error.read().decode("utf8", 'ignore'))))

```

