Predict Patient diabetes using Microsoft Azure ML Work Bench

Prerequisites

- Valid Azure Subscription
- Azure Machine Learning Workbench

Steps Involved

- Download and Understanding DataSet
- Build Model

Objective

The objective of the dataset is to diagnostically predict whether a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. All patients here are females at least 21 years old of Pima Indian heritage.

The Tool

Azure Machine Learning services (preview) aka Azure ML workbench are an integrated, end-to-end data science and advanced analytics solution. It helps **professional data scientists** prepare data, develop experiments, and deploy models at cloud scale.

Download and Understanding Dataset

The datasets consist of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on

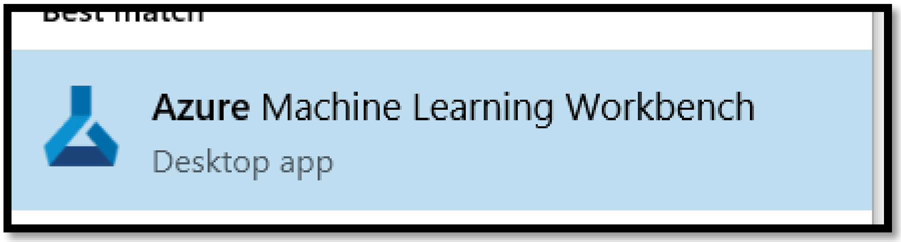
// Please download dataset from https://aka.ms/aidevdaysds2

Column Name	Description	Data Type
Pregnancies	Number of times pregnant	Numeric
Glucose	Plasma glucose concentration a 2 hour in an oral glucose tolerance test	Numeric
BloodPressure	Diastolic blood pressure (mm Hg)	Numeric
SkinThickness	Triceps skin fold thickness (mm)	Numeric
Insulin	2-Hour serum insulin (mu U/ml)	Numeric
BMI	Body mass index (weight in kg/(height in m)^2)	Numeric
DiabetesPedigreeFunction	Diabetes pedigree function	Numeric
Age	Age (years)	Numeric
Outcome	Class variable (0 or 1)	Numeric

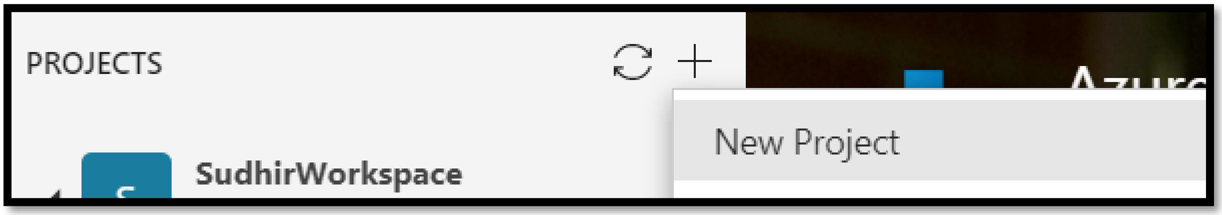
Note:- Datasource Reference: - https://www.kaggle.com/uciml/pima-indians-diabetes-database/data

Evaluating Data Fields

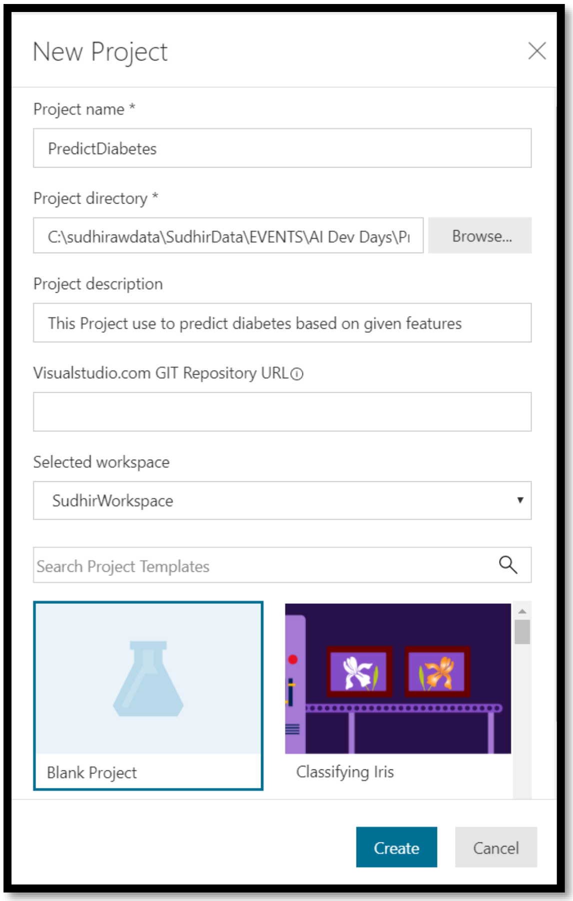
- Open “**Azure Machine Learning Workbench**”



- Click **+** and **New Project** to add new project



- Provide **Project Name**, **Project directory**, **Project Description**, **Selected Workspace** and select **Blank Project**. Click **Create**



New Project

Project name *

PredictDiabetes

Project directory *

C:\sudhirawdata\SudhirData\EVENTS\AI Dev Days\Pı

Project description

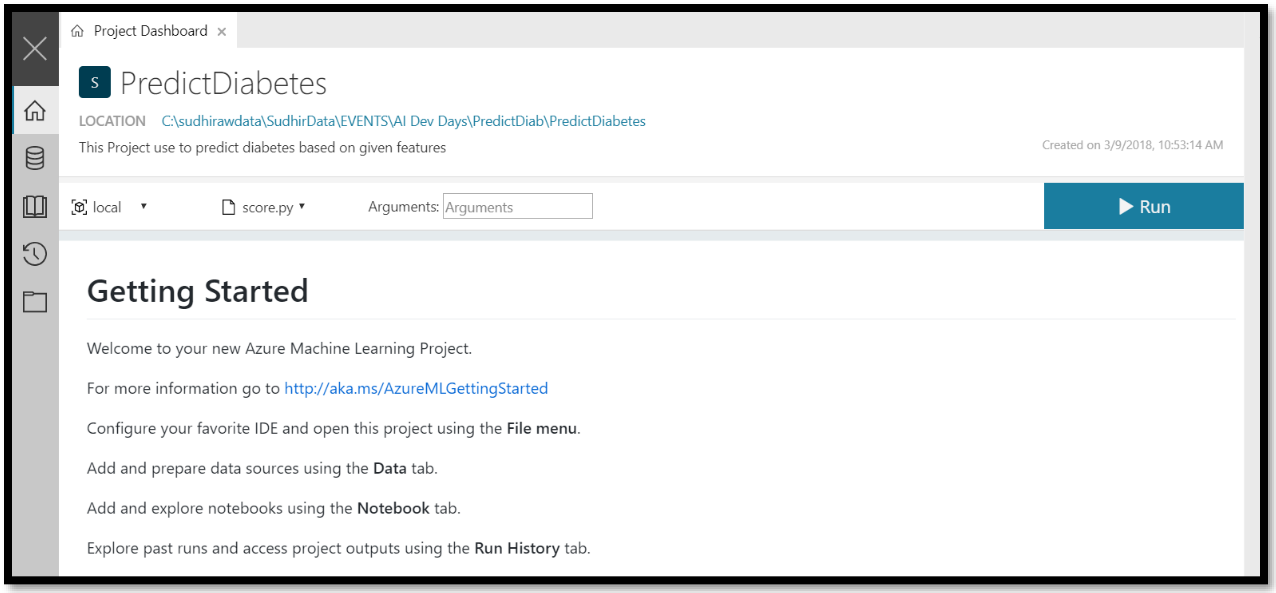
This Project use to predict diabetes based on given features

Visualstudio.com GIT Repository URL

Selected workspace

SudhirWorkspace

- ➢ Once created it will look like below



Project Dashboard

PredictDiabetes

LOCATION C:\sudhirawdata\SudhirData\EVENTS\AI Dev Days\PredictDiab\PredictDiabetes

This Project use to predict diabetes based on given features

Created on 3/9/2018, 10:53:14 AM

local score.py Arguments: Arguments

Getting Started

Welcome to your new Azure Machine Learning Project.

For more information go to http://aka.ms/AzureMLGettingStarted

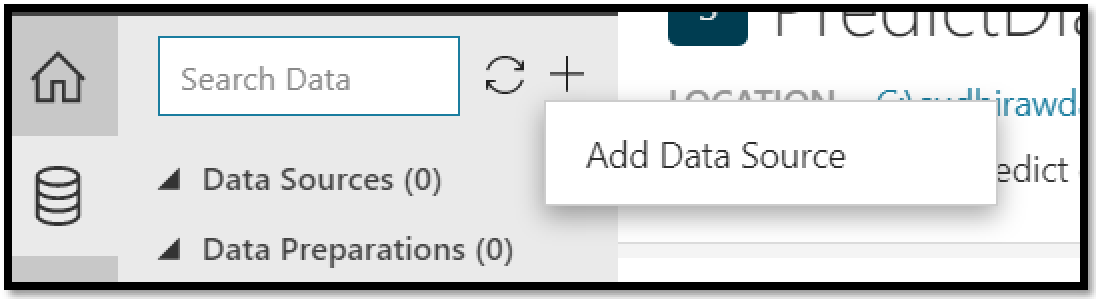
Configure your favorite IDE and open this project using the **File menu**.

Add and prepare data sources using the **Data** tab.

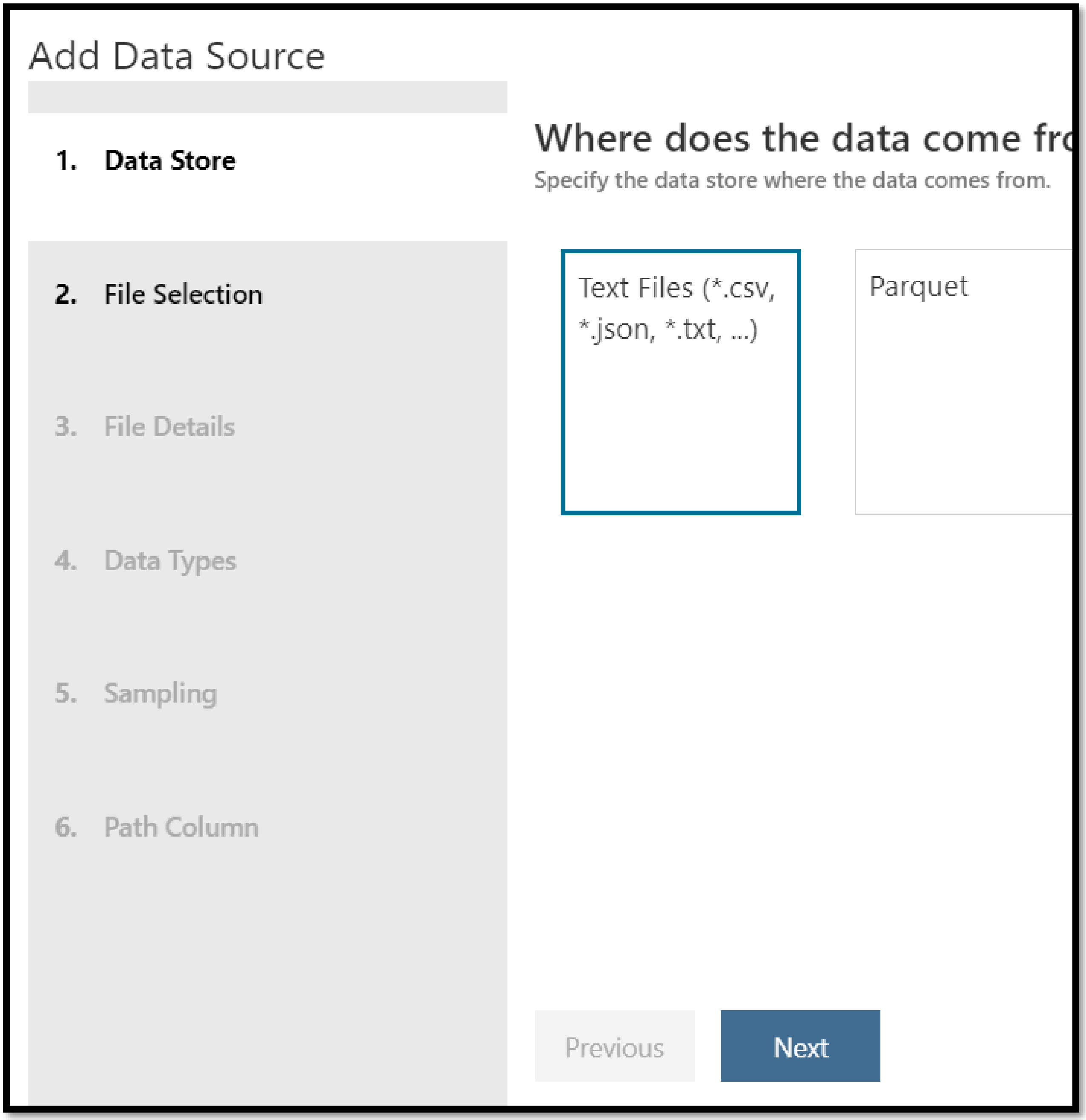
Add and explore notebooks using the **Notebook** tab.

Explore past runs and access project outputs using the **Run History** tab.

- ➢ Click **Data** (next icon after home) and Click **+** and **Add Data Source**



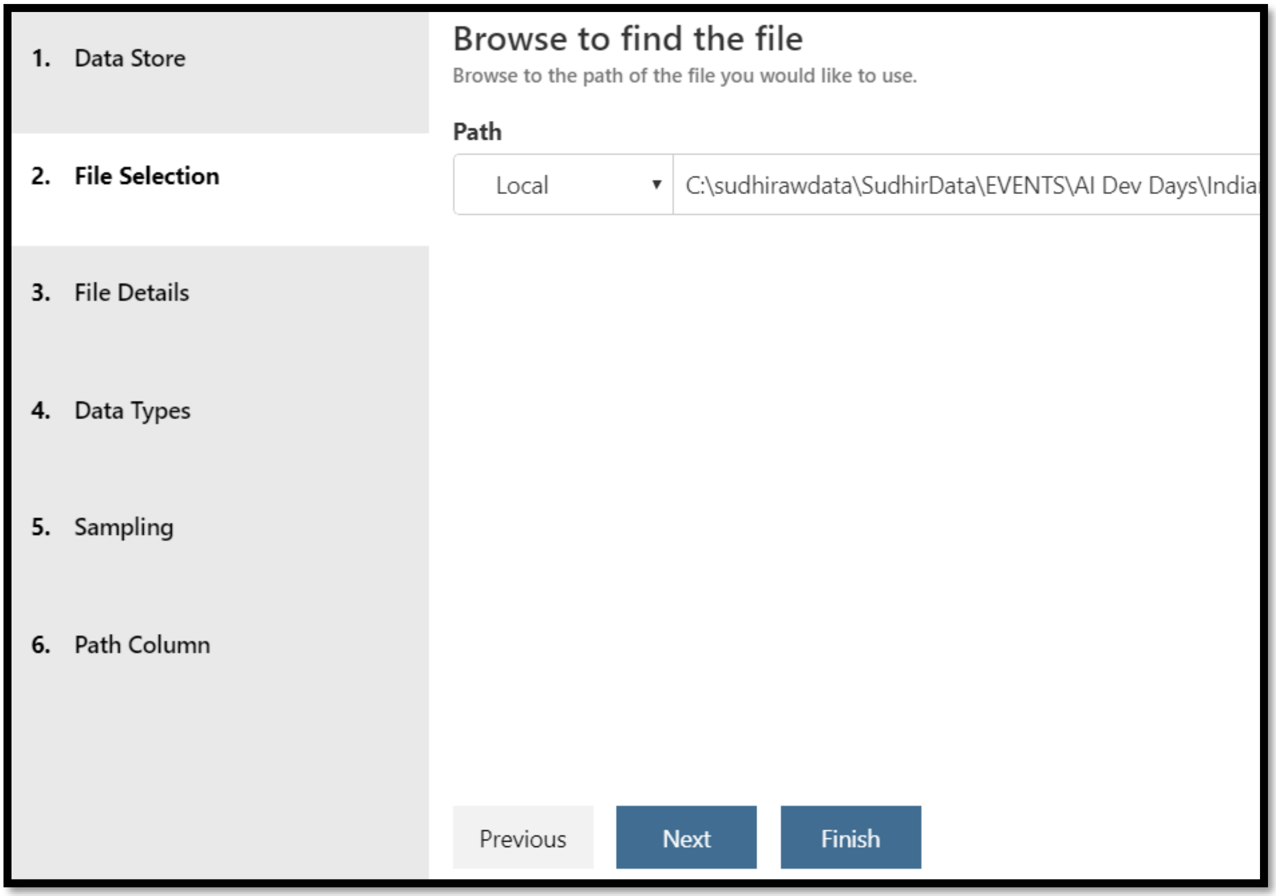
- Select **Text Files** and **Next**



Add Data Source

1. Data Store
2. File Selection
3. File Details
4. Data Types
5. Sampling
6. Path Column
Where does the data come fro
Specify the data store where the data comes from.
Text Files (*.csv, *.json, *.txt, ...)
Parquet
Previous
Next

- Select the file which downloaded in earlier step, Click **Next**



1. Data Store
2. **File Selection**
3. File Details
4. Data Types
5. Sampling
6. Path Column

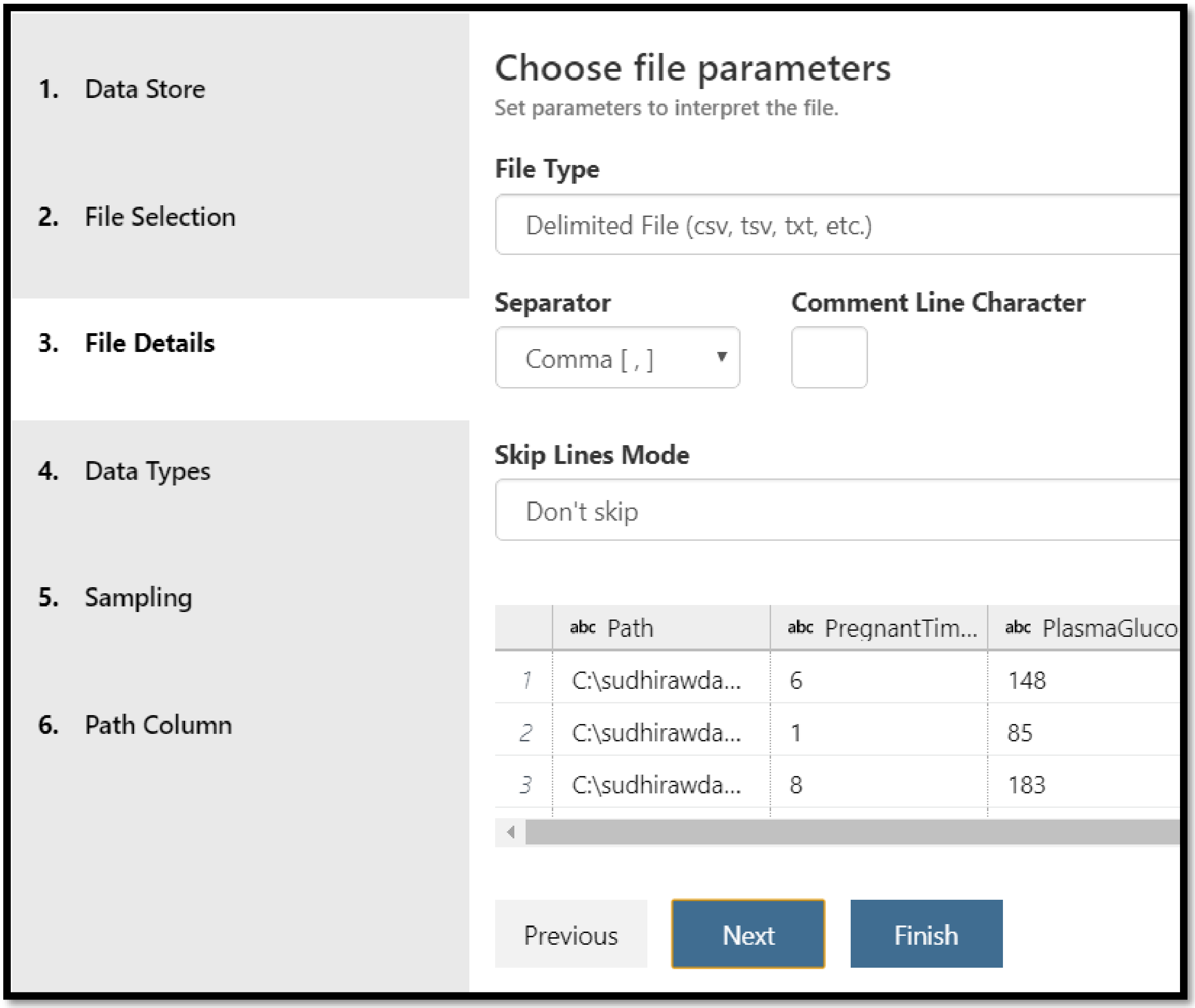
Browse to find the file

Browse to the path of the file you would like to use.

Path

Local ▾ C:\sudhirawdata\SudhirData\EVENTS\AI Dev Days\India

➢ On step number **3,** leave it default and click **Next**



Choose file parameters

Set parameters to interpret the file.

File Type: Delimited File (csv, tsv, txt, etc.)

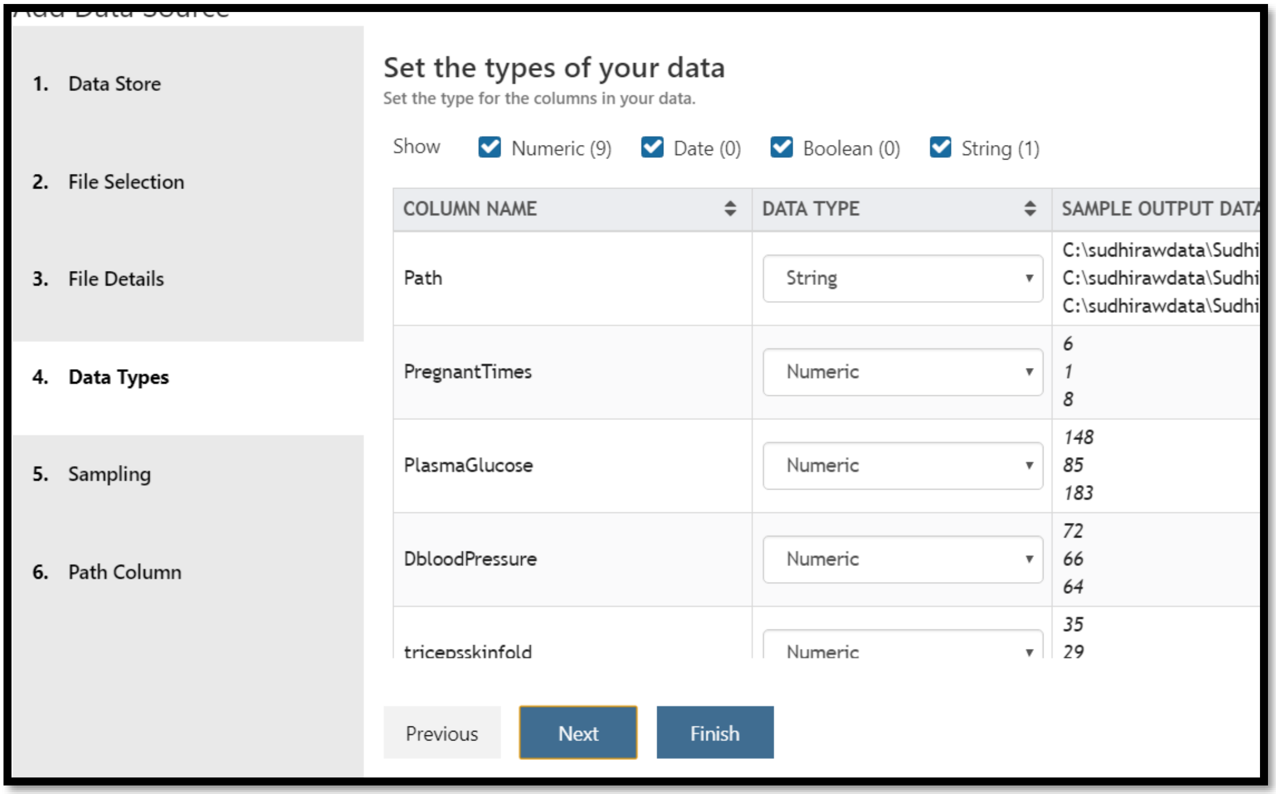
Separator: Comma [,]

Comment Line Character:

Skip Lines Mode: Don't skip

	abc Path	abc PregnantTim...	abc PlasmaGluco
1	C:\sudhirawda...	6	148
2	C:\sudhirawda...	1	85
3	C:\sudhirawda...	8	183

- Check the datatype on step number **4**. All should be numeric except first column (**Path**). Click **Next**



Add Data Source

1. Data Store
2. File Selection
3. File Details
4. **Data Types**
5. Sampling
6. Path Column

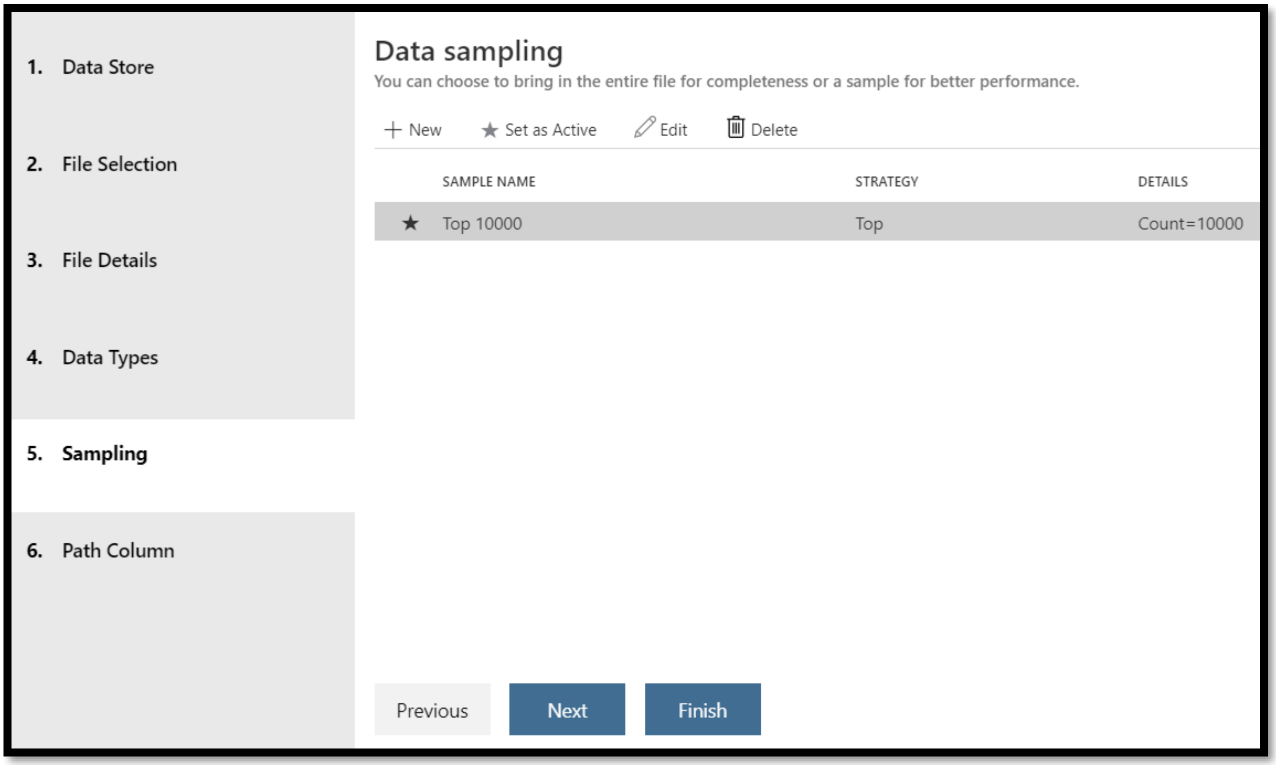
Set the types of your data

Set the type for the columns in your data.

Show ☑ Numeric (9) ☑ Date (0) ☑ Boolean (0) ☑ String (1)

COLUMN NAME	DATA TYPE	SAMPLE OUTPUT DATA
Path	String	C:\sudhirawdata\Sudhi C:\sudhirawdata\Sudhi C:\sudhirawdata\Sudhi
PregnantTimes	Numeric	*6* *1* *8*
PlasmaGlucose	Numeric	*148* *85* *183*
DbloodPressure	Numeric	*72* *66* *64*
tricepsskinfold	Numeric	*35* *29*

- Click **Next** on step number **5**



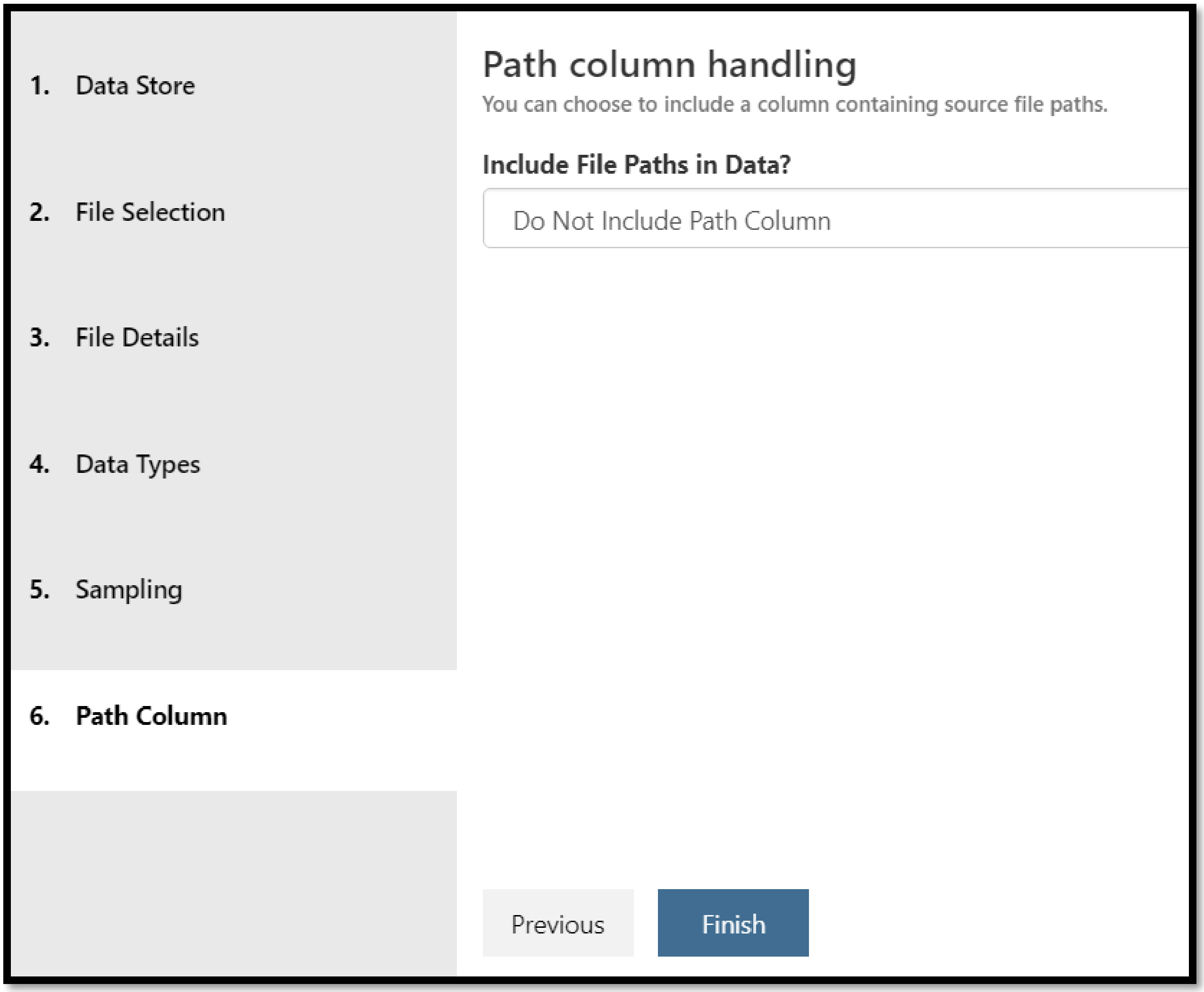
1. Data Store
2. File Selection
3. File Details
4. Data Types
5. **Sampling**
6. Path Column

Data sampling

You can choose to bring in the entire file for completeness or a sample for better performance.

	SAMPLE NAME	STRATEGY	DETAILS
★	Top 10000	Top	Count=10000

- On step number **6** (**Path Column**), Select “**Do Not Include Path Column**” and click **Finish**



1. Data Store
2. File Selection
3. File Details
4. Data Types
5. Sampling
6. **Path Column**

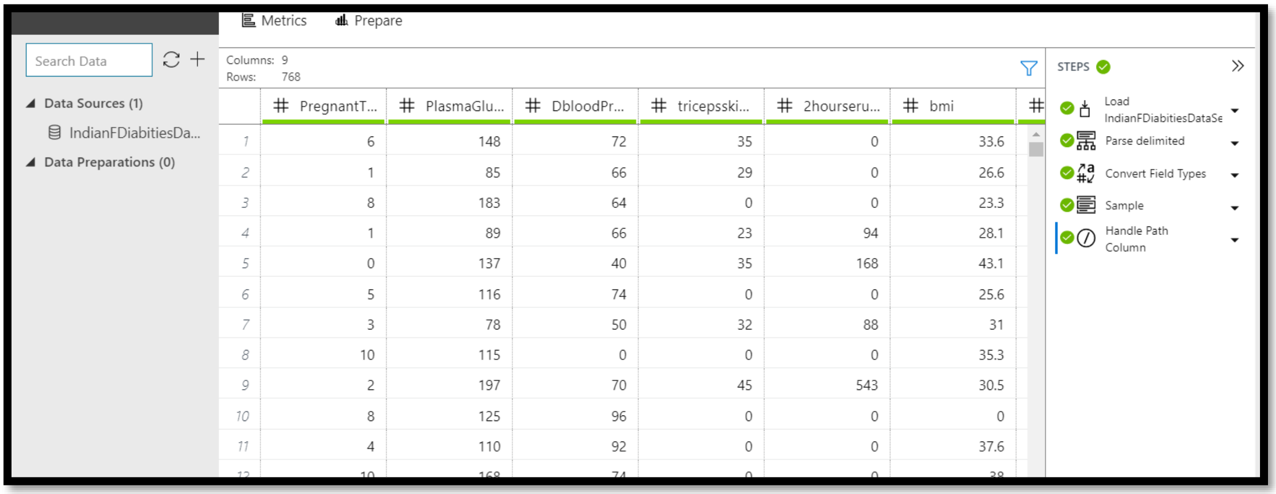
Path column handling

You can choose to include a column containing source file paths.

Include File Paths in Data?

Do Not Include Path Column

- Once setup, screen will look like



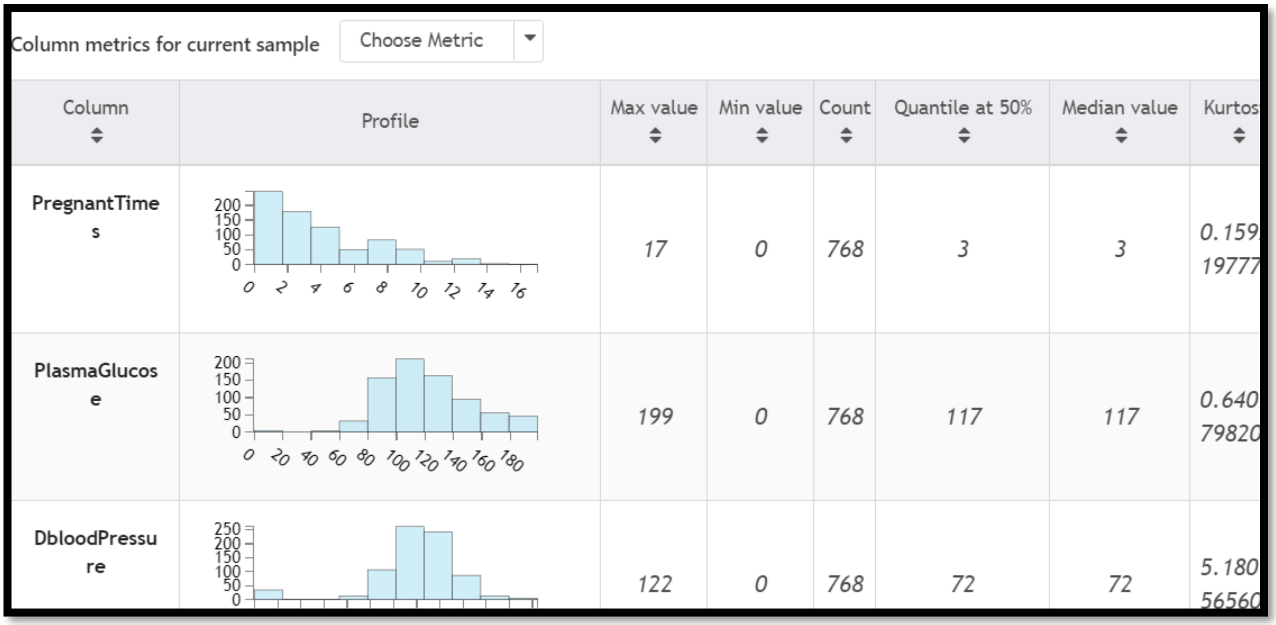
Columns: 9
Rows: 768

	# PregnantT...	# PlasmaGlu...	# DbloodPr...	# tricepsski...	# 2hourseru...	# bmi
1	6	148	72	35	0	33.6
2	1	85	66	29	0	26.6
3	8	183	64	0	0	23.3
4	1	89	66	23	94	28.1
5	0	137	40	35	168	43.1
6	5	116	74	0	0	25.6
7	3	78	50	32	88	31
8	10	115	0	0	0	35.3
9	2	197	70	45	543	30.5
10	8	125	96	0	0	0
11	4	110	92	0	0	37.6
12	10	168	74	0	0	38

STEPS ✓

- ✓ Load IndianFDiabitiesDataSe
- ✓ Parse delimited
- ✓ Convert Field Types
- ✓ Sample
- ✓ Handle Path Column

- Click on **Matrix.** Data Profiling will help you identify problems in your data such as invalid dates, missing values etc.

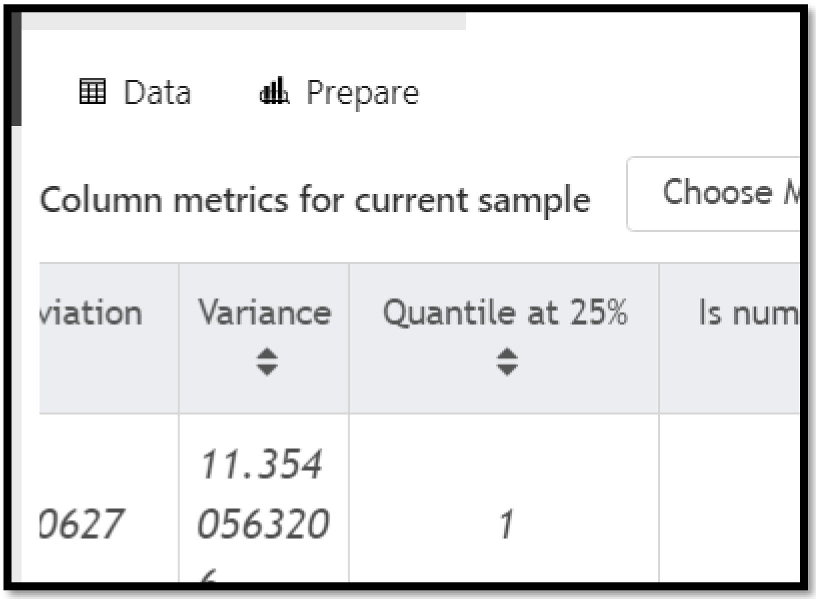


Column metrics for current sample

Column	Profile	Max value	Min value	Count	Quantile at 50%	Median value	Kurtos
PregnantTimes		17	0	768	3	3	0.159 19777
PlasmaGlucose		199	0	768	117	117	0.640 79820
DbloodPressure		122	0	768	72	72	5.180 56560

*<u>**Note**</u>: - Since data is clean we are not going to do any transformation

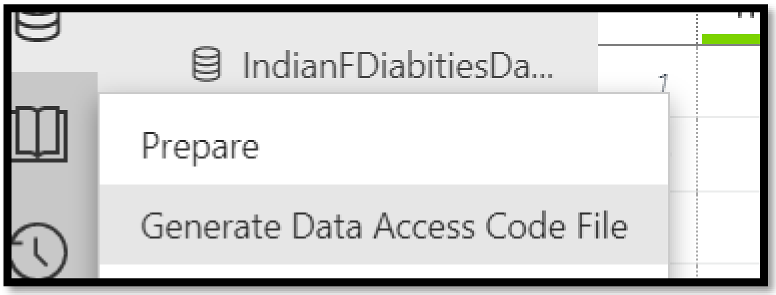
- Click **Data**



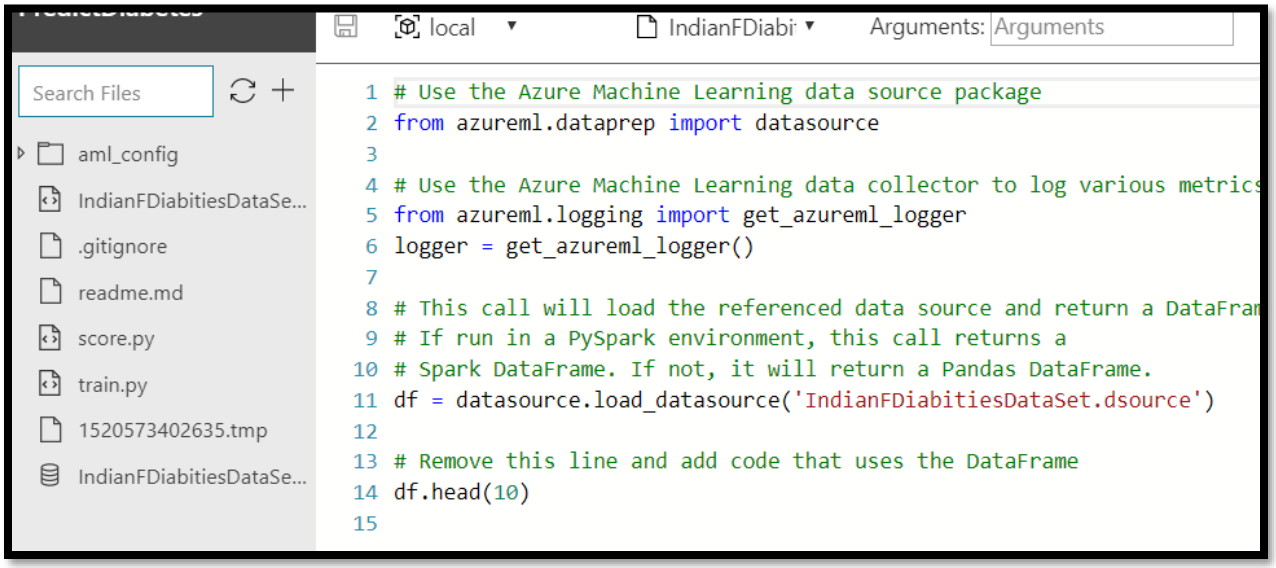
Column metrics for current sample

viation	Variance	Quantile at 25%	Is num
0627	11.354 056320	1	

- Right click on **Data Source** and click **Generate Data Access Code File**



- After generating data access code file, the screen will look like



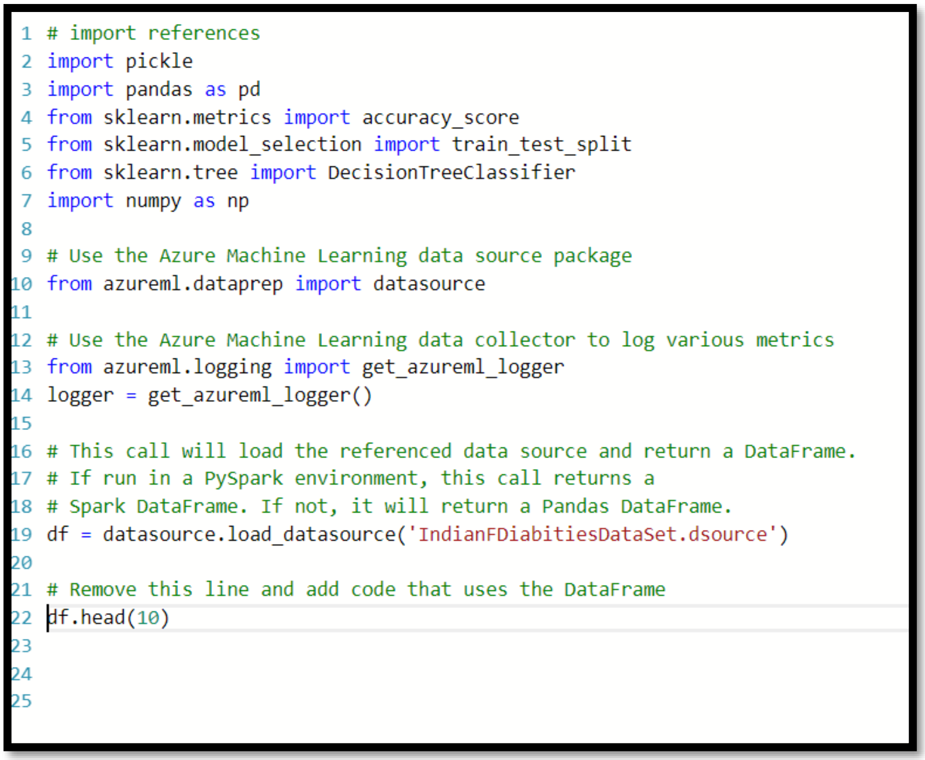
```
# Use the Azure Machine Learning data source package
from azureml.dataprep import datasource

# Use the Azure Machine Learning data collector to log various metrics
from azureml.logging import get_azureml_logger
logger = get_azureml_logger()

# This call will load the referenced data source and return a DataFran
# If run in a PySpark environment, this call returns a
# Spark DataFrame. If not, it will return a Pandas DataFrame.
df = datasource.load_datasource('IndianFDiabitiesDataSet.dsource')

# Remove this line and add code that uses the DataFrame
df.head(10)

```

Building Machine Learning Model using scikitlearn

- Copy below code and paste it in editor

```
# import references
import pickle
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import numpy as np

# Use the Azure Machine Learning data source package
from azureml.dataprep import datasource
```

- The screen will look like below after adding above code

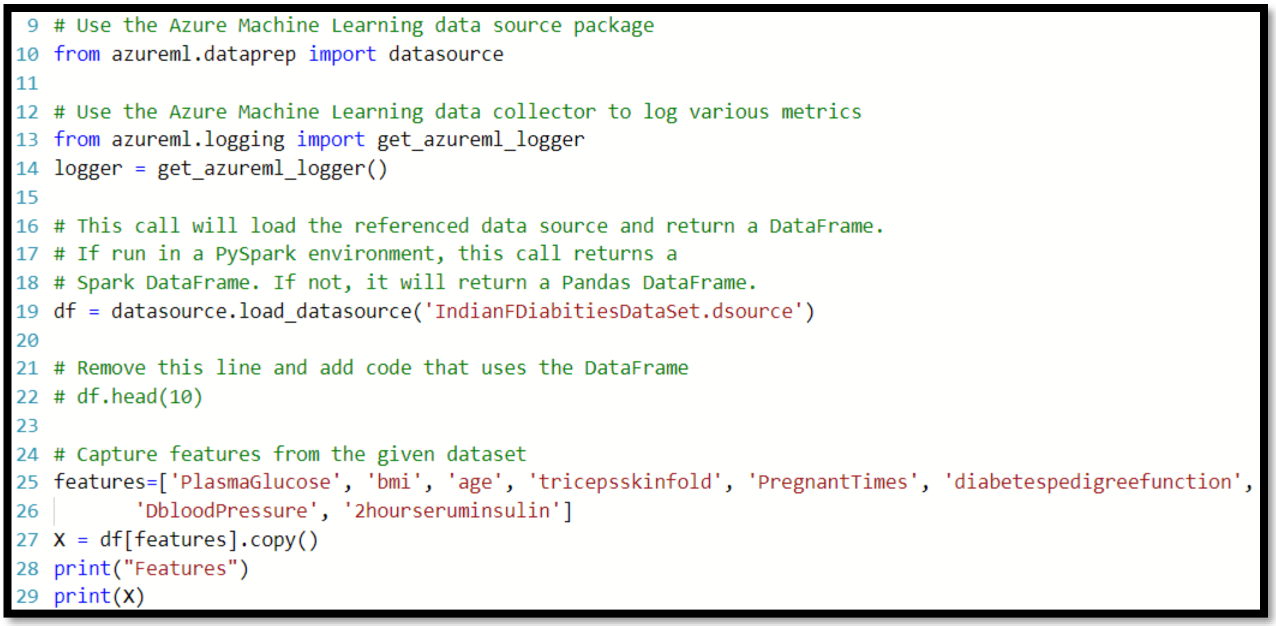


```
# import references
import pickle
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import numpy as np

# Use the Azure Machine Learning data source package
from azureml.dataprep import datasource

# Use the Azure Machine Learning data collector to log various metrics
from azureml.logging import get_azureml_logger
logger = get_azureml_logger()

# This call will load the referenced data source and return a DataFrame.
# If run in a PySpark environment, this call returns a
# Spark DataFrame. If not, it will return a Pandas DataFrame.
df = datasource.load_datasource('IndianFDiabitiesDataSet.dsource')

# Remove this line and add code that uses the DataFrame
df.head(10)
```

- Comment code **# df.head(10)** and copy below code and paste it to the editor. In this piece of code we are defining columns which algorithm used to define patterns

```
# Capture features from the given dataset
features=['PlasmaGlucose', 'bmi', 'age', 'tricepsskinfold', 'PregnantTimes',
'diabetespedigreefunction',
    'DbloodPressure', '2hourseruminsulin']
X = df[features].copy()
print("Features")
print(X)
```

- After changes

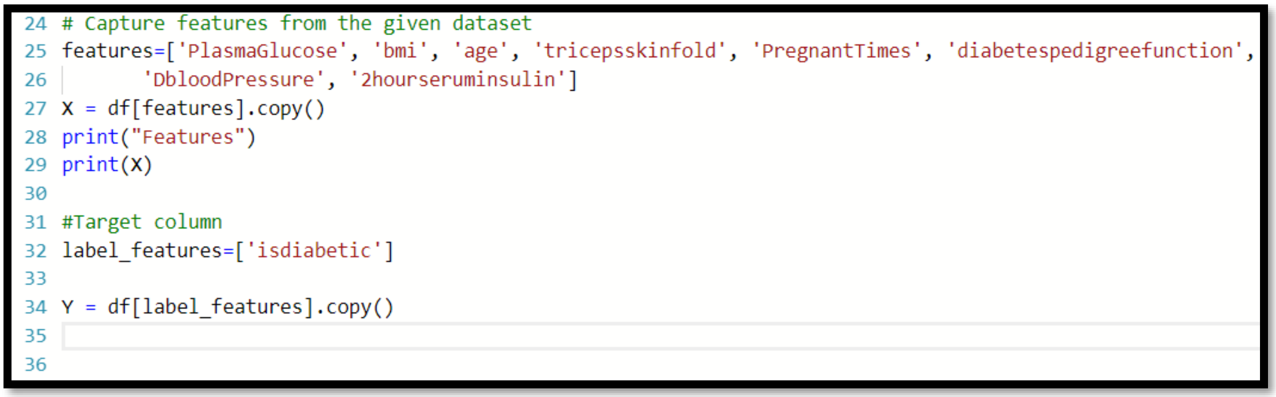


```
# Use the Azure Machine Learning data source package
from azureml.dataprep import datasource

# Use the Azure Machine Learning data collector to log various metrics
from azureml.logging import get_azureml_logger
logger = get_azureml_logger()

# This call will load the referenced data source and return a DataFrame.
# If run in a PySpark environment, this call returns a
# Spark DataFrame. If not, it will return a Pandas DataFrame.
df = datasource.load_datasource('IndianFDiabitiesDataSet.dsource')

# Remove this line and add code that uses the DataFrame
# df.head(10)

# Capture features from the given dataset
features=['PlasmaGlucose', 'bmi', 'age', 'tricepsskinfold', 'PregnantTimes', 'diabetespedigreefunction',
       'DbloodPressure', '2hourseruminsulin']
X = df[features].copy()
print("Features")
print(X)
```

- Copy below code and paste it in editor. In this piece of code, we are defining the target column

```
#Target column
label_features=['isdiabetic']

Y = df[label_features].copy()
```

- Below is the screenshot after pasting the code

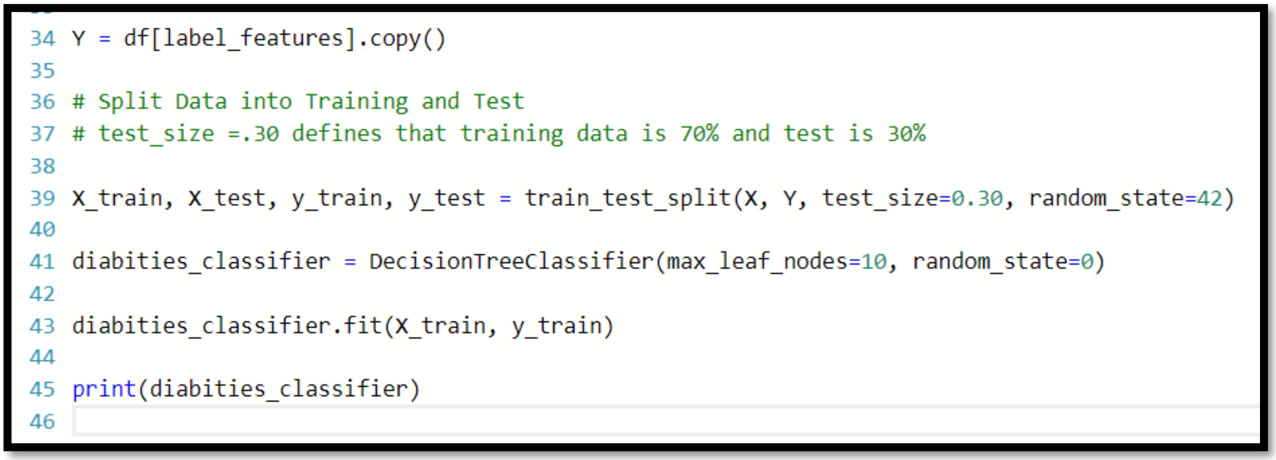


```
# Capture features from the given dataset
features=['PlasmaGlucose', 'bmi', 'age', 'tricepsskinfold', 'PregnantTimes', 'diabetespedigreefunction',
       'DbloodPressure', '2hourseruminsulin']
X = df[features].copy()
print("Features")
print(X)

#Target column
label_features=['isdiabetic']

Y = df[label_features].copy()


```

- Copy below code and paste it in editor. In this example we are splitting the dataset to avoid any overfitting and underfitting scenario. Next we are defining the algorithm (DecisionTreeClassifier) to solve this problem and used fit method to build a decision tree classifier from the training set.

```
# Split Data into Training and Test
# test_size =.30 defines that training data is 70% and test is 30%

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=42)

diabities_classifier = DecisionTreeClassifier(max_leaf_nodes=10, random_state=0)

diabities_classifier.fit(X_train, y_train)

print(diabities_classifier)
```

- Below is the screenshot after implementing above code

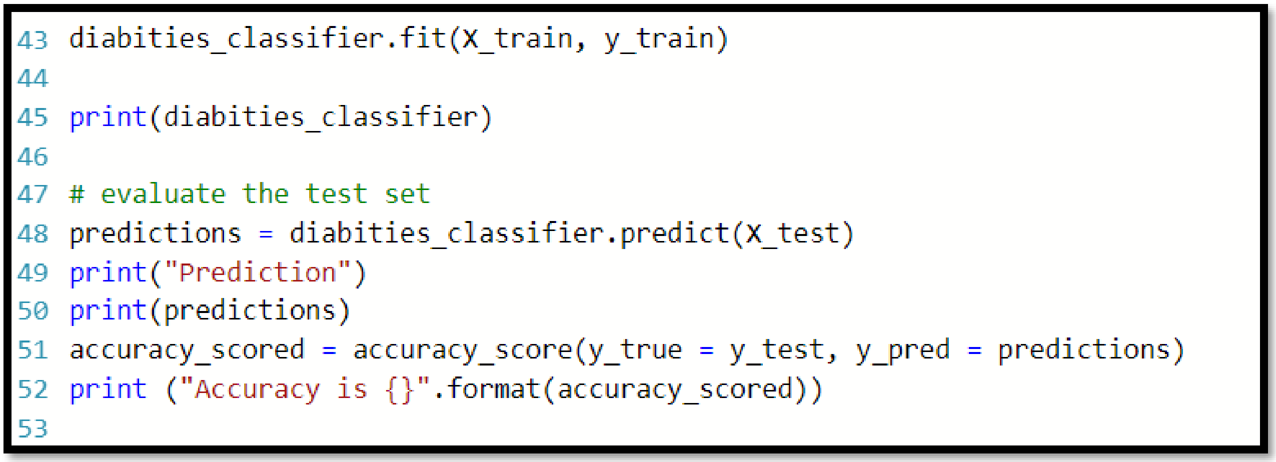


```
Y = df[label_features].copy()

# Split Data into Training and Test
# test_size =.30 defines that training data is 70% and test is 30%

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=42)

diabities_classifier = DecisionTreeClassifier(max_leaf_nodes=10, random_state=0)

diabities_classifier.fit(X_train, y_train)

print(diabities_classifier)

```

- Copy below code. Once Model is trained. Next step is to test it against the test dataset

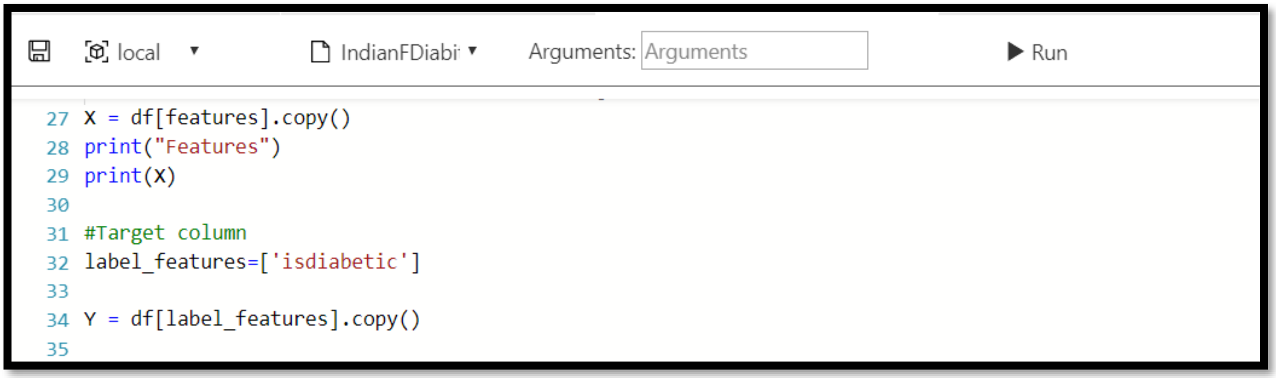
```
# evaluate the test set

predictions = diabities_classifier.predict(X_test)

print("Prediction")

print(predictions)

accuracy_scored = accuracy_score(y_true = y_test, y_pred = predictions)

print ("Accuracy is {}".format(accuracy_scored))
```

- Below is the screenshot after implementing above code



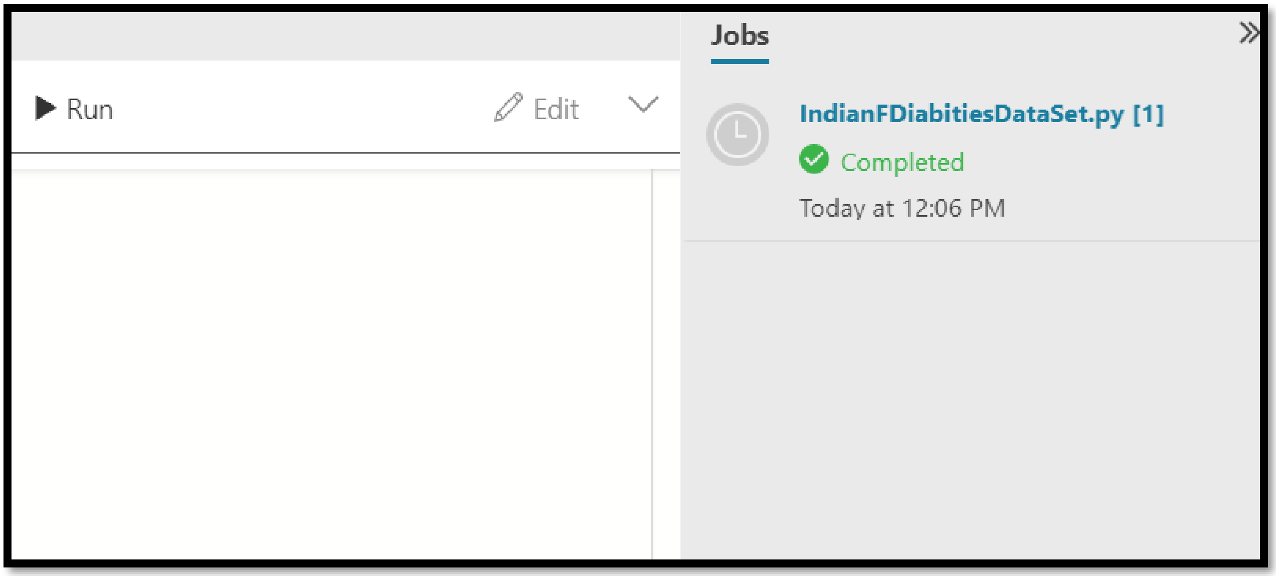
```
diabities_classifier.fit(X_train, y_train)

print(diabities_classifier)

# evaluate the test set
predictions = diabities_classifier.predict(X_test)
print("Prediction")
print(predictions)
accuracy_scored = accuracy_score(y_true = y_test, y_pred = predictions)
print ("Accuracy is {}".format(accuracy_scored))

```

- Let’s save and run the code to test the accuracy of the model. Click “b” to save the file and click **Run**

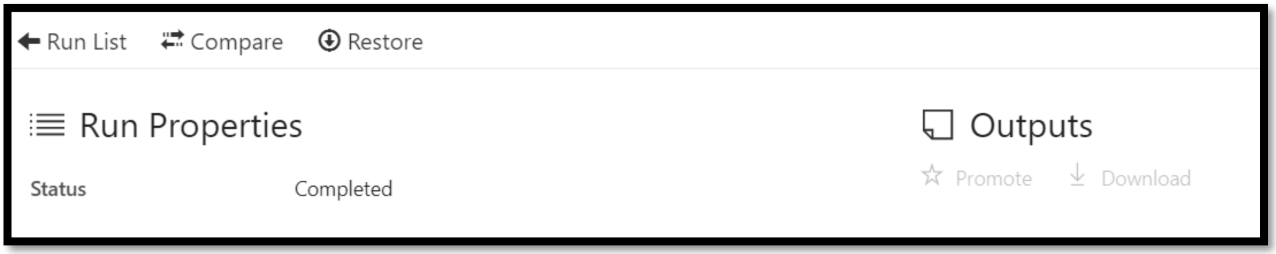


```
X = df[features].copy()
print("Features")
print(X)

#Target column
label_features=['isdiabetic']

Y = df[label_features].copy()

```

- On the right side of the screen it will show the progress of the job. Wait till job gets completed

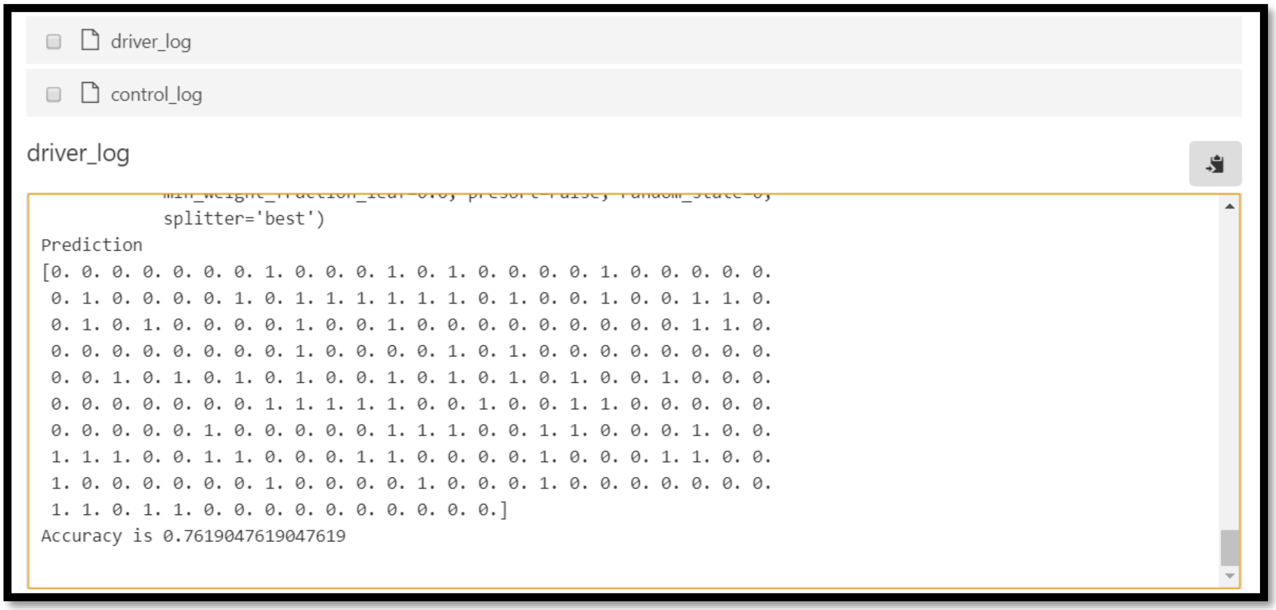


- Click on the completed Job



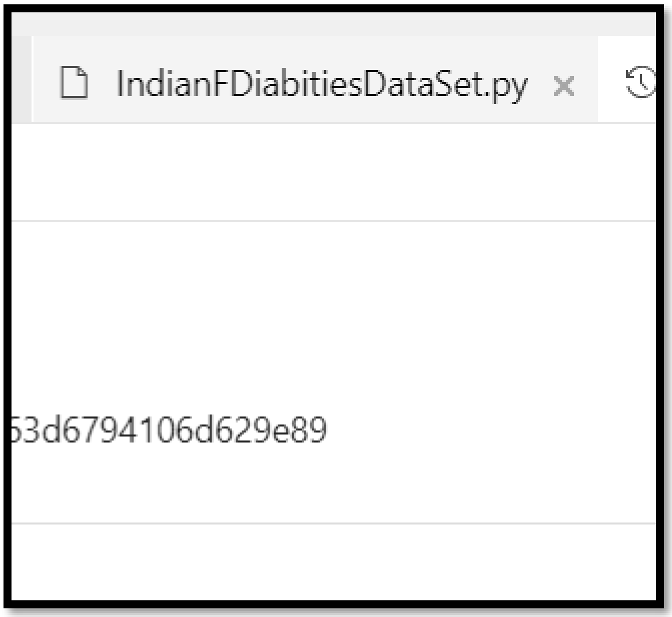
Status Completed

- Scroll down and click on **driver_log**



```
                min_weight_fraction_leaf=0.0, presort=False, random_state=0,
                splitter='best')
Prediction
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 1. 0. 0. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0.
 0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 0. 0.
 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0.
 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 1. 1. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Accuracy is 0.7619047619047619
```

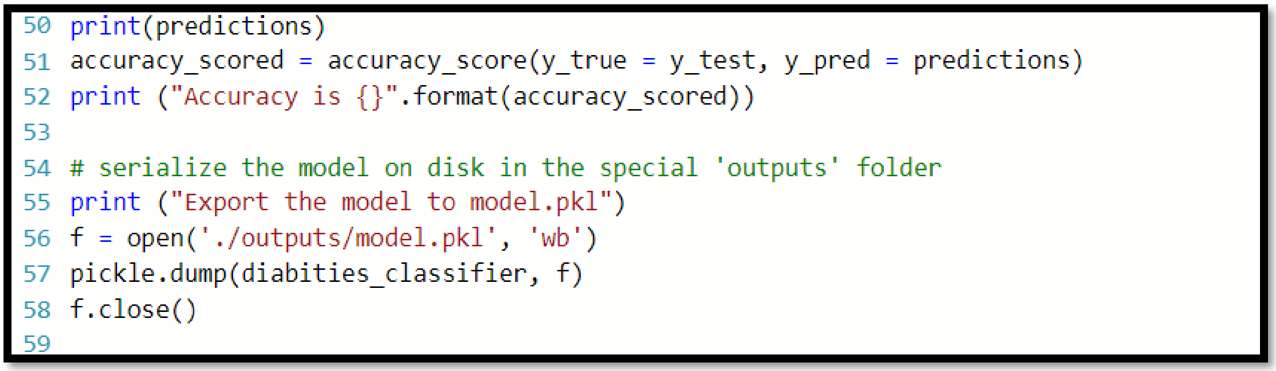
- It will show the accuracy around 76. You can try to add more random data and tweaking algorithm parameter to test for better accuracy. For now we are good with 76.
- Click on python file to build model file.



- Copy below code and paste it. Below code will serialize the model by inserting pickle file in output folder.

```
# serialize the model on disk in the special 'outputs' folder
print ("Export the model to model.pkl")
f = open('./outputs/model.pkl', 'wb')
pickle.dump(diabities_classifier, f)
f.close()
```

- Below is the screenshot after implementing above code.

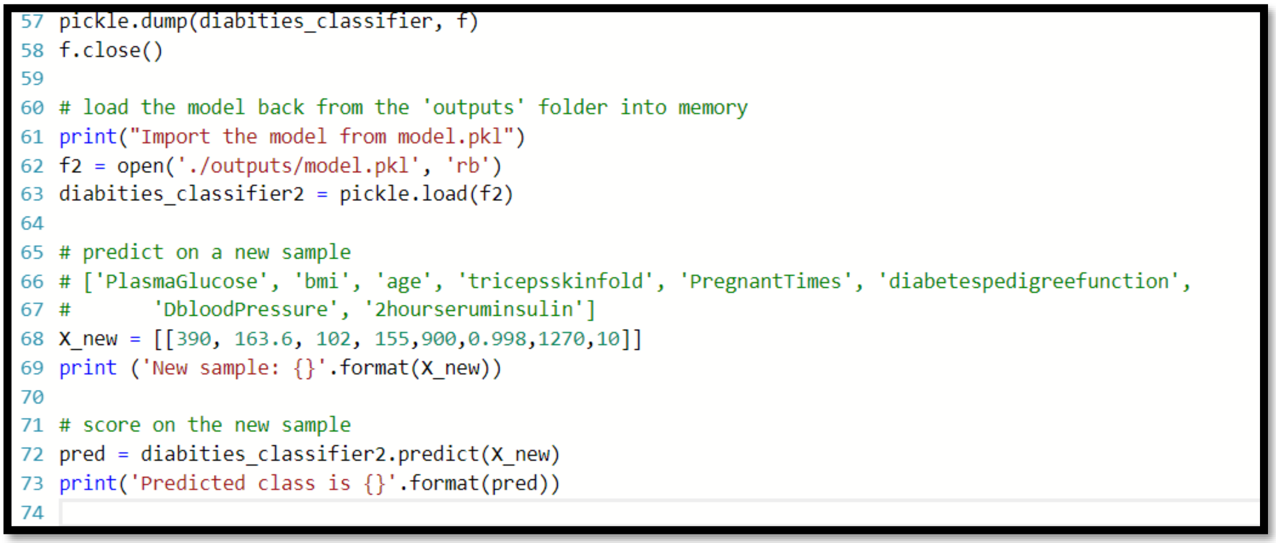


```
print(predictions)
accuracy_scored = accuracy_score(y_true = y_test, y_pred = predictions)
print ("Accuracy is {}".format(accuracy_scored))

# serialize the model on disk in the special 'outputs' folder
print ("Export the model to model.pkl")
f = open('./outputs/model.pkl', 'wb')
pickle.dump(diabities_classifier, f)
f.close()

```

- Copy below code and past it in editor. In below code, we are loading pickle file in pass sample data to predict the outcome

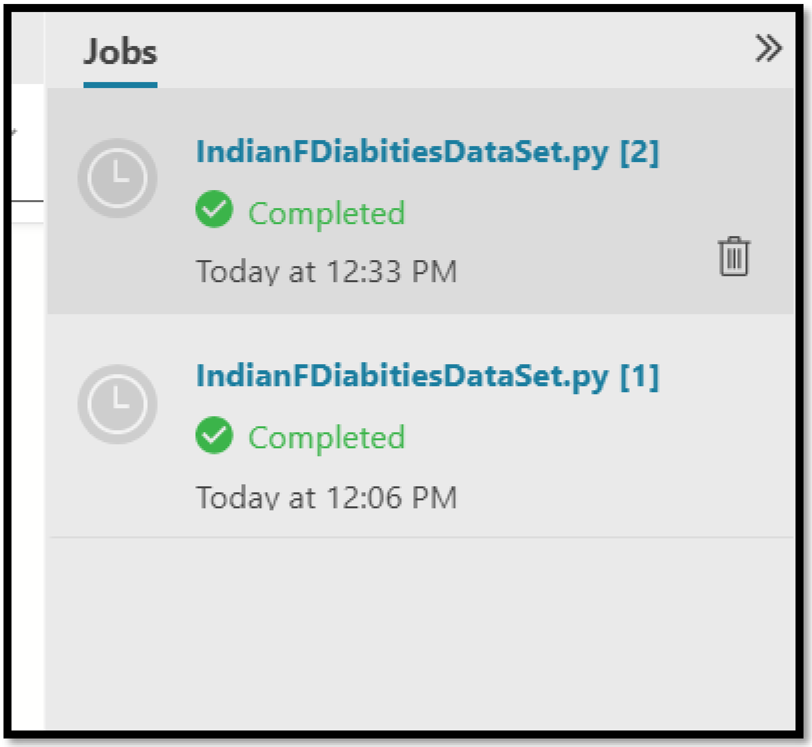
```
# load the model back from the 'outputs' folder into memory

print("Import the model from model.pkl")

f2 = open('./outputs/model.pkl', 'rb')

diabities_classifier2 = pickle.load(f2)


# predict on a new sample

# ['PlasmaGlucose', 'bmi', 'age', 'tricepsskinfold', 'PregnantTimes', 'diabetespedigreefunction',

#      'DbloodPressure', '2hourseruminsulin']

X_new = [[390, 163.6, 102, 155,900,0.998,1270,10]]

print ('New sample: {}'.format(X_new))

# score on the new sample

pred = diabities_classifier2.predict(X_new)

print('Predicted class is {}'.format(pred))
```

- Below is the screenshot after pasting above code



```
pickle.dump(diabities_classifier, f)
f.close()

# load the model back from the 'outputs' folder into memory
print("Import the model from model.pkl")
f2 = open('./outputs/model.pkl', 'rb')
diabities_classifier2 = pickle.load(f2)

# predict on a new sample
# ['PlasmaGlucose', 'bmi', 'age', 'tricepsskinfold', 'PregnantTimes', 'diabetespedigreefunction',
#       'DbloodPressure', '2hourseruminsulin']
X_new = [[390, 163.6, 102, 155,900,0.998,1270,10]]
print ('New sample: {}'.format(X_new))

# score on the new sample
pred = diabities_classifier2.predict(X_new)
print('Predicted class is {}'.format(pred))

```

- Click **Run**

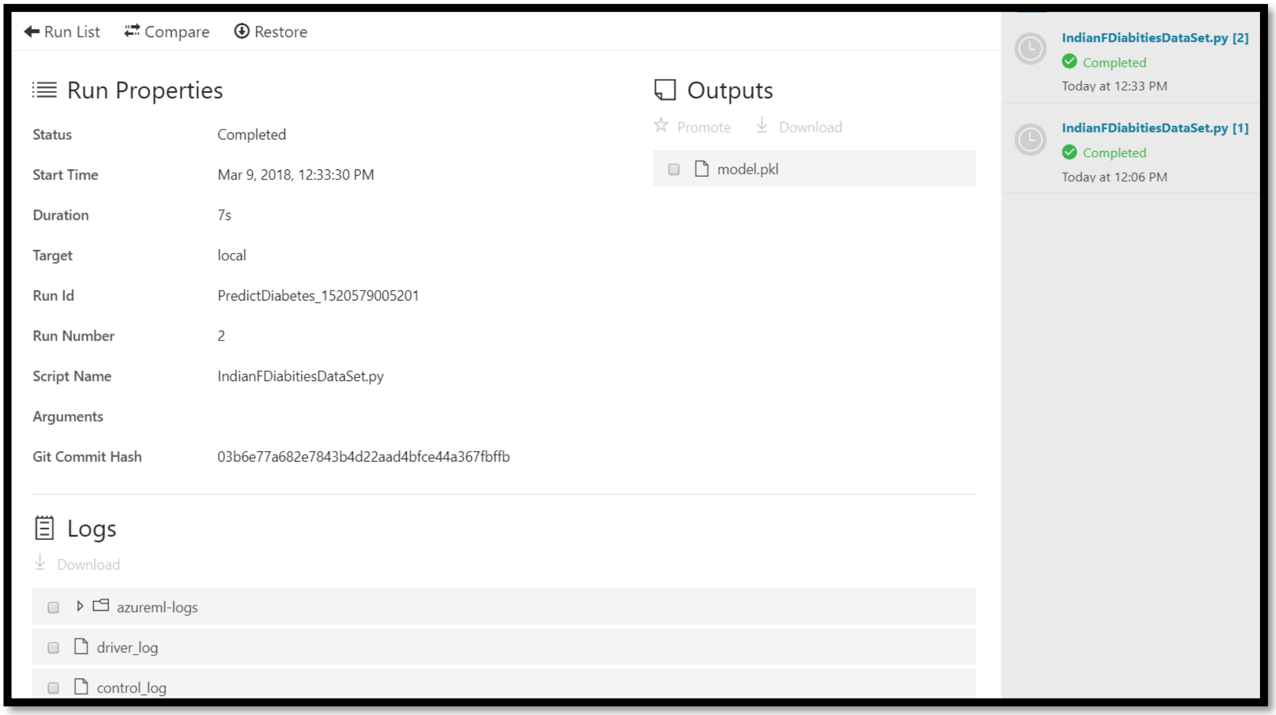


Jobs

IndianFDiabitiesDataSet.py [2]
✓ Completed
Today at 12:33 PM

IndianFDiabitiesDataSet.py [1]
✓ Completed
Today at 12:06 PM

- ➢ Click on latest job execution



Run Properties

Status	Completed
Start Time	Mar 9, 2018, 12:33:30 PM
Duration	7s
Target	local
Run Id	PredictDiabetes_1520579005201
Run Number	2
Script Name	IndianFDiabitiesDataSet.py
Arguments	
Git Commit Hash	03b6e77a682e7843b4d22aad4bfce44a367fbffb

Outputs

□ model.pkl

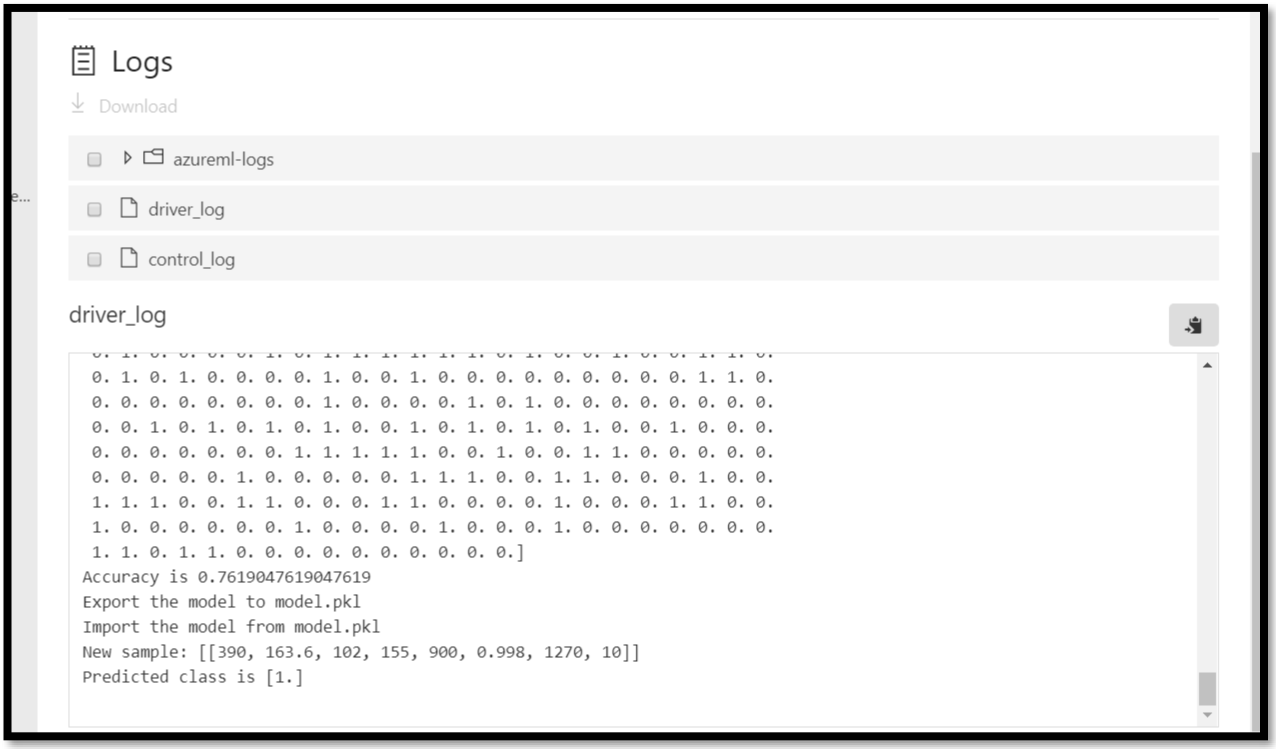
Logs

□ azureml-logs
□ driver_log
□ control_log

IndianFDiabitiesDataSet.py [2]
✓ Completed
Today at 12:33 PM

IndianFDiabitiesDataSet.py [1]
✓ Completed
Today at 12:06 PM

- Click on **driver_log.** In below screenshot you will notice exporting and importing of model file and test it against new sample data



```
 0. 1. 0. 0. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0.
 0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 0. 0.
 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0.
 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 1. 1. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Accuracy is 0.7619047619047619
Export the model to model.pkl
Import the model from model.pkl
New sample: [[390, 163.6, 102, 155, 900, 0.998, 1270, 10]]
Predicted class is [1.]
```

- Next Step is to operationalize it which you already did in previous HOL. If time permits you can try it again here. You can also look out advance data preparation to understand various data wrangling techniques.

Summary

Well done! In this hands-on lab we solved a problem to predict whether a person going to be a diabetic or not. This will help to take proactive measure and cure patient. You can deploy this model and make it available as a web service so that any application from anywhere can access it. We also learn the capabilities of Azure Machine Learning and how to work with it. You can download data from UCI and Kaggle to solve other problems. Let's move onto the next step.