Machine_Learning_Model

Data Science in Fabric - Bamidele

Data Import

This loads the necessary dataset into the notebook

In [1]:
```
# Load open dataset from Azure - (The storage access info for open dataset diabetes
blob_account_name = "azureopendatastorage"
blob_container_name = "mlsamples"
blob_relative_path = "diabetes"
blob_sas_token = r"" # This will be blank since container is Anonymous access

# Setting Spark config to access  blob storage
wasbs_path = f"wasbs://%s@%s.blob.core.windows.net/%s" % (blob_container_name, blob
spark.conf.set("fs.azure.sas.%s.%s.blob.core.windows.net" % (blob_container_name, b
print("Remote blob path: " + wasbs_path)

# Spark read parquet and save the data as diabetesdata
diabetesdata = spark.read.parquet(wasbs_path)
```

```
StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 4, Finished, Available)
Remote blob path: wasbs://mlsamples@azureopendatastorage.blob.core.windows.net/diabe
tes
```

In [2]:
```
# Display the dataset
display(diabetesdata)
```

```
StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 5, Finished, Available)
SynapseWidget(Synapse.DataFrame, 0e70d94c-89b5-4c1f-bb0b-b1576c4a9418)
```

In [3]:
```
# Graph an output using the default Chart option
display(diabetesdata)
# Using Boxplot to view BMI column
```

```
StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 6, Finished, Available)
SynapseWidget(Synapse.DataFrame, f29c7c4d-3fea-494f-b931-d136343b8749)
```

Data Preparation

-This can be done using the data wrangler to generate code or writing code in notebook

In [4]:
```
# Convert the data into a Pandas dataframe using the toPandas() function

diabetesdata = diabetesdata.toPandas()
diabetesdata.head()
```

```
StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 7, Finished, Available)
```

Out[4]:

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
0	59	2	32.1	101.0	157	93.2	38.0	4.0	4.8598	87	151
1	48	1	21.6	87.0	183	103.2	70.0	3.0	3.8918	69	75
2	72	2	30.5	93.0	156	93.6	41.0	4.0	4.6728	85	141
3	24	1	25.3	84.0	198	131.4	40.0	5.0	4.8903	89	206
4	50	1	23.0	101.0	192	125.4	52.0	4.0	4.2905	80	135

In [5]:
```
# Code generated by Data Wrangler for pandas DataFrame

def clean_data(diabetesdata):
    # Created column 'Risk' from formula
    diabetesdata['Risk'] = (diabetesdata['Y'] > 211.5).astype(int)
    return diabetesdata

diabetesdata_clean = clean_data(diabetesdata.copy())
diabetesdata_clean.head()
```

StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 8, Finished, Available)

Out[5]:

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y	Risk
0	59	2	32.1	101.0	157	93.2	38.0	4.0	4.8598	87	151	0
1	48	1	21.6	87.0	183	103.2	70.0	3.0	3.8918	69	75	0
2	72	2	30.5	93.0	156	93.6	41.0	4.0	4.6728	85	141	0
3	24	1	25.3	84.0	198	131.4	40.0	5.0	4.8903	89	206	0
4	50	1	23.0	101.0	192	125.4	52.0	4.0	4.2905	80	135	0

Confirm the code produced from data wrangler

In [6]:
```
# description of the risk column added by data wrangler with MAX as 1 and MIN as 0
diabetesdata_clean.describe()
```

StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 9, Finished, Available)

Out[6]:

	AGE	SEX	BMI	BP	S1	S2	S3
count	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000
mean	48.518100	1.468326	26.375792	94.647014	189.140271	115.439140	49.788462
std	13.109028	0.499561	4.418122	13.831283	34.608052	30.413081	12.934202
min	19.000000	1.000000	18.000000	62.000000	97.000000	41.600000	22.000000
25%	38.250000	1.000000	23.200000	84.000000	164.250000	96.050000	40.250000
50%	50.000000	1.000000	25.700000	93.000000	186.000000	113.000000	48.000000
75%	59.000000	2.000000	29.275000	105.000000	209.750000	134.500000	57.750000
max	79.000000	2.000000	42.200000	133.000000	301.000000	242.400000	99.000000

Train machine learning models - REGRESSION

In [7]:
```
# Regression Model

from sklearn.model_selection import train_test_split

X, y = diabetesdata_clean[['AGE','SEX','BMI','BP','S1','S2','S3','S4','S5','S6']].v

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_st
```

StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 10, Finished, Available)

In [8]:
```
# import MLFlow

import mlflow
experiment_name = "diabetes-regression"
mlflow.set_experiment(experiment_name)
```

StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 11, Finished, Available)

```
2023-12-03:11:36:06,46 WARNING  [synapse_mlflow_utils.py:244] To save or load Apache
Spark model files, please attach a Lakehouse.
```

Out[8]:
```
<Experiment: artifact_location='', creation_time=1701600334627, experiment_id='af6
d6495-21c0-472e-ba69-412f72af6a72', last_update_time=None, lifecycle_stage='activ
e', name='diabetes-regression', tags={}>
```

In [9]:
```
from sklearn.linear_model import LinearRegression

with mlflow.start_run():
   mlflow.autolog()

   model = LinearRegression()
   model.fit(X_train, y_train)
```

StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 12, Finished, Available)

```
2023/12/03 11:36:09 INFO mlflow.tracking.fluent: Autologging successfully enabled fo
r sklearn.
2023-12-03:11:36:09,662 WARNING  [tracking_store.py:153] log_inputs not supported
2023/12/03 11:36:13 WARNING mlflow.utils.autologging_utils: MLflow autologging encou
ntered a warning: "/home/trusted-service-user/cluster-env/trident_env/lib/python3.1
0/site-packages/_distutils_hack/__init__.py:33: UserWarning: Setuptools is replacing
distutils."
```

The code above trains a regression model using Linear Regression. Parameters, metrics, and artifacts, are automatically logged with MLflow.

Train a classification model

In [10]:
```
from sklearn.model_selection import train_test_split

X, y = diabetesdata_clean[['AGE','SEX','BMI','BP','S1','S2','S3','S4','S5','S6']].v

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_st
```

StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 13, Finished, Available)

In [11]:
```
import mlflow
experiment_name = "diabetes-classification"
mlflow.set_experiment(experiment_name)
```

StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 14, Finished, Available)

Out[11]:
```
<Experiment: artifact_location='', creation_time=1701600571599, experiment_id='191
70325-cd41-4135-9aa5-1bf0505c1c66', last_update_time=None, lifecycle_stage='activ
e', name='diabetes-classification', tags={}>
```

In [12]:
```
from sklearn.linear_model import LogisticRegression

with mlflow.start_run():
    mlflow.sklearn.autolog()

    model = LogisticRegression(C=1/0.1, solver="liblinear").fit(X_train, y_train)
```

StatementMeta(, d12dacfa-ceaf-4264-ab1b-93ec404d33f4, 15, Finished, Available)

```
2023-12-03:11:36:21,925 WARNING  [tracking_store.py:153] log_inputs not supported
```

The above code trains a classification model using Logistic Regression. Parameters, metrics, and artifacts, are automatically logged with MLflow.