1. **Explain what problem you are going to solve using this dataset. Provide a brief overview of your problem statement.**

   When diabetes is detected early, people can prevent progression and is less costly. If Diabetes goes undiagnosed or diagnosed too late, it can damage your heart, blood vessels , eyes, kidneys, and nerves. In 2018, 34.2 million Americans had diabetes. Of that 34.2 million people, 7.3 million were undiagnosed (https://www.diabetes.org/). To prevent undiagnosed Diabetes or late detection I am going to create an Early-Stage Diabetes classification model using this dataset.

   The dataset will be checked for missing values (depending on how many of that specific feature is missing different approaches can be made – taking the average, removing those records and/or using the median value). After cleaning is done then preprocessing will take place. Here I will process the dataset to be a numerical and standardize the dataset if required based on the numerical values. Once that is completed then I will do feature selection to select the best features for the model. Using those selected features, I will train and test the model and use Accuracy as the success metric. The goal is to accurately predict whether someone has diabetes based on their answers to a survey of their symptoms/characteristics.

2. **Explain your dataset. Explore your dataset and provide at least 5 meaningful charts/graphs with explanation.**

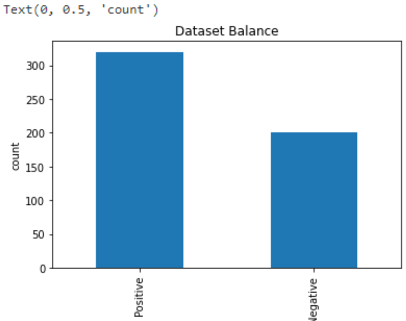
   This dataset consists of survey data of people between the ages 16-90, has 17 features (columns) and 520 records (rows). Only one column out of the whole dataset is not binary (Age column). The dataset consists of symptoms/characteristics that surgery respondents either responded with a ‘yes’ or a ‘no’ along with their age, gender and whether they do have diabetes or not.

   The first graph that was created when exploring the dataset was a bar graph to show the balance of the dataset. By just looking at this graph one can notice that the dataset is unbalanced, there is a significant amount of more positive labelled cases than negative.

```
#checking the balance of the dataset
df['class'].value_counts().plot(kind='bar')
plt.title('Dataset Balance')
plt.ylabel('count')
```

[4] ✓ <1 sec

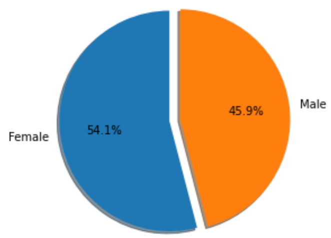
```
Text(0, 0.5, 'count')
```



   The second chart is a pie chart showing the number of people that have diabetes split by gender. When looking at the chart one can notice that there are more female positive cases than men. When extracting information like this from a chart its important to note the percentage of the number of males and females in the study to begin with. The total number of women in the study is 192 and 173 of them being positive meaning 90% of the women in the study have diabetes whereas for the men 147 out of 328 were positive for diabetes (45%).

```python
# Pie chart for positive cases by gender
gendersplit = pd.crosstab(df['Gender'], df['class'])
gendersplit['total'] = gendersplit['Negative'] +gendersplit['Positive']
gendersplit['perc'] = ((gendersplit['Positive']/gendersplit['total'])*100).round(2)

labels = 'Female', 'Male'
sizes = gendersplit['Positive']
explode = (0, 0.1)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

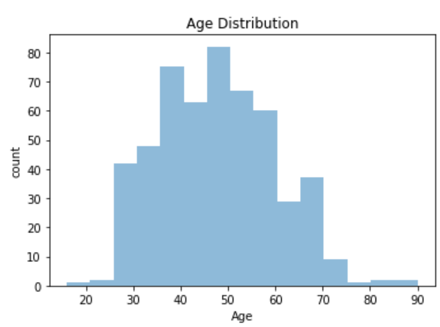
[5] ✓ <1 sec



The distribution graph and boxplot use the same data but are presented in different charts/graphs to show the different ways the same data can be presented. These graphs show the distribution of the survey takers age. You can see from the graph that the median age is just under 50 (the median in the boxplot is the line in the middle inside the box). From the boxplot you can also see the minimum, maximum, first quartile, third quartile and outliers.

```python
#distribution plot of age for positive cases
data = df['Age']
plt.hist(data, bins = 15, alpha=0.5, label = "class")
plt.title('Age Distribution')
plt.ylabel('count')
plt.xlabel('Age')
plt.subplots_adjust(hspace=0.5)
```

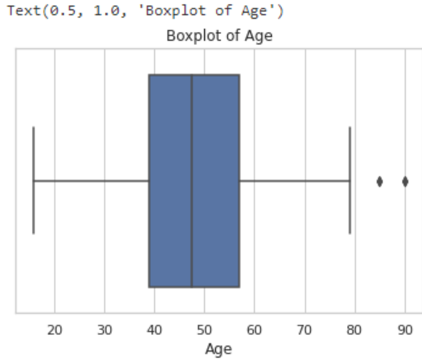
[6] ✓ <1 sec




```python
#boxplot of positive cases by age
sns.set_theme(style="whitegrid")
ax = sns.boxplot(x=data)
plt.title('Boxplot of Age')
```

[7] ✓ <1 sec

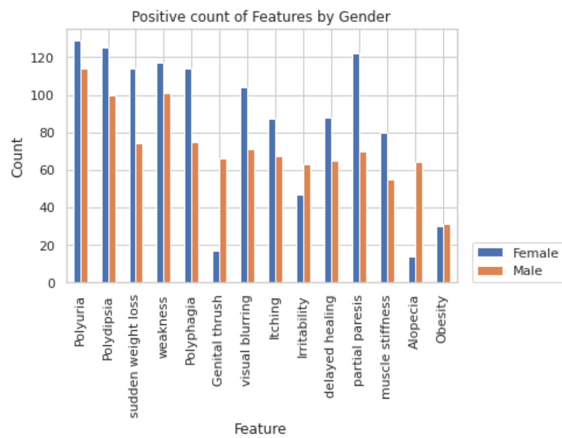
```
Text(0.5, 1.0, 'Boxplot of Age')
```



The bar graph below shows the comparison between the positive counts for diabetes between men and women by the features. This is a good way to see which features are more prominent in positive cases of women or men. For example, from the chart you can see that genital thrush and alopecia is much more common for diabetes in men in this dataset while partial paresis and sudden weight loss is more common with women.

```
gender_features.plot(kind = 'bar')
plt.title('Positive count of Features by Gender')
plt.ylabel('Count')
plt.legend(loc=(1.04,0))
plt.xlabel('Feature')
plt.subplots_adjust(hspace=0.5)
```

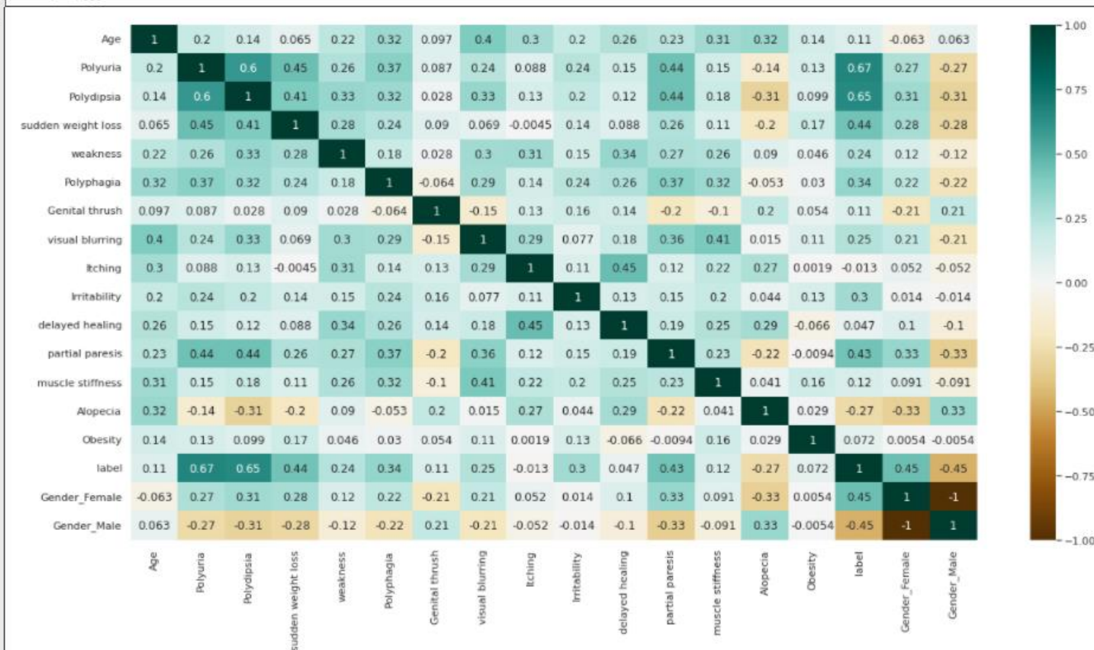
[10] ✓ <1 sec



The final chart is a correlation heatmap. The heatmap visualizes the correlation between the different variables. From the heatmap one can see that the two main features affecting the label (target variable) are Polyuria and Polydipsia. The relationship between all the other variables can also be observed from the correlation heatmap.

```
# Finding the relations between the variables.
plt.figure(figsize=(20,10))
cor = dataset.corr()
sns.heatmap(cor,cmap='BrBG',annot=True)
cor
```

[12] ✓ <1 sec



3. **Do data cleaning/pre-processing as required and explain what you have done for your dataset and why?**

For data cleaning/pre-processing a few steps were taken. I started with checking for null values then encoding the dataset so that all string values (categorical values) would be numerical values. From there I did feature selection to select the best features for my model.

I first checked for null values to make sure that all records were complete. I found that there were no Null values so no further action was required to replace any values. Most of the columns as mentioned in the previous question were binary columns with strings "yes/no." I converted all these columns to have 1/0 instead and used One Hot Encoding with the binary strategy for the gender column. The binary strategy essentially means that each category value was converted into a new column and assigned 1 or 0 based on that column encoded value. At this point all the columns are now numerical with all except the age column being binary columns. In this case, a lot of the data is binary, in these types of dataframes typically there is no need for scaling the data since most of the values are either 1 or 0.

Feature selection is used to minimize the number of input variables one needs for a model to predict the target variable. There are many different techniques for feature selection. The feature selection technique that was used in this model is Mutual Information with select K Best. Mutual information measures the dependencies between the variables. When the variables are independent from each other than the score is 0 and higher values mean higher dependency between the variables. The Select K best feature selection will then select the 10 best features based on the mutual information score.

[3]
```
#checking for missing values
df.isnull().sum()
```
✓ <1 sec

```
Age                   0
Gender                0
Polyuria              0
Polydipsia            0
sudden weight loss    0
weakness              0
Polyphagia            0
Genital thrush        0
visual blurring       0
Itching               0
Irritability          0
delayed healing       0
partial paresis       0
muscle stiffness      0
Alopecia              0
Obesity               0
class                 0
dtype: int64
```

[8]
```
#encoding data to be 1/0 binary
encode_df = df.replace(["Yes", 'No'],[1,0])
encode_df['class'] = encode_df['class'].apply(lambda x: 1 if x == 'Positive' else 0)
```
✓ <1 sec

[9]
```
#oneHotEncoding gender column
catecols = ['Gender']
for col in catecols:
    dfeat = pd.get_dummies(encode_df[col], prefix=col, drop_first=False)

    #adding the new feature to the dataframe
    encode_df = pd.concat([encode_df, dfeat], axis=1)
```
✓ <1 sec

[16]
```
#Feature selection techniques
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif

# define feature selection
fs = SelectKBest(score_func = mutual_info_classif,k=10)
#apply feature selection
x = fs.fit_transform(x, y.ravel())
print(x.shape)
```
✓ 1 sec

```
(520, 10)
```

4. Implement 2 machine learning models, explain which algorithms you have selected and why. Compare them and show success metrics (Accuracy/RMSE/Confusion Matrix) as per your problem. Explain results.

The two machine learning algorithms that were chosen for this classification model is Support Vector Machine (SVM) and Decision Tree Classifier. SVM was chosen since this algorithm is known for performing well on datasets with binary classification and outliers having minimal impact. Decision Tree Classifier was also chosen due to popularity with binary classification and its ability to quickly classify unknown records. As patient records are very different from each other (every person has different health records) this feature was very important when choosing a model.

Comparison of the two models were done using accuracy and confusion matrix which can be seen in below screenshots. The confusion matrix shows True Positives (predicted positive and its true), True Negative (predicted negative and its true), False Positive (predicted positive and its false), and False Negative (predicted negative and its false SVM had an accuracy of 0.929 and had TP of 47, FP of 7, FN of 4 and TN of 98. Decision Tree had an accuracy of 0.942 and had TP of 51, FP of 3, FN of 6 and TN of 96. Therefore, the better model between these two is the Decision Tree Classifier.

```
def SVC(X_train, X_test, y_train, y_test):

    # Classification algorithm, predictions, and Success metrics.
    model = LinearSVC()
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    acc = accuracy_score(y_test, pred)
    cm = confusion_matrix(y_test, pred)

    print('\nAccuracy of Support Vector Machine:', acc)
    print('\nConfusion Matrix of Support Vector Machine:\n', cm)

    return model, pred, acc, cm
```

[24] ✓ <1 sec

```
SVC = SVC(X_train, X_test, y_train, y_test)
```

[25] ✓ <1 sec

```
Accuracy of Support Vector Machine: 0.9294871794871795

Confusion Matrix of Support Vector Machine:
 [[47  7]
 [ 4 98]]
```

```
def DT(X_train, X_test, y_train, y_test):

    # Classification algorithm, predictions, and Success metrics.
    model = DecisionTreeClassifier(random_state=1)
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    acc = accuracy_score(y_test, pred)
    cm = confusion_matrix(y_test, pred)

    print('\nAccuracy of Decision Tree:', acc)
    print('\nConfusion Matrix of Decision Tree:\n', cm)

    return model, pred, acc, cm
```

[26] ✓ <1 sec

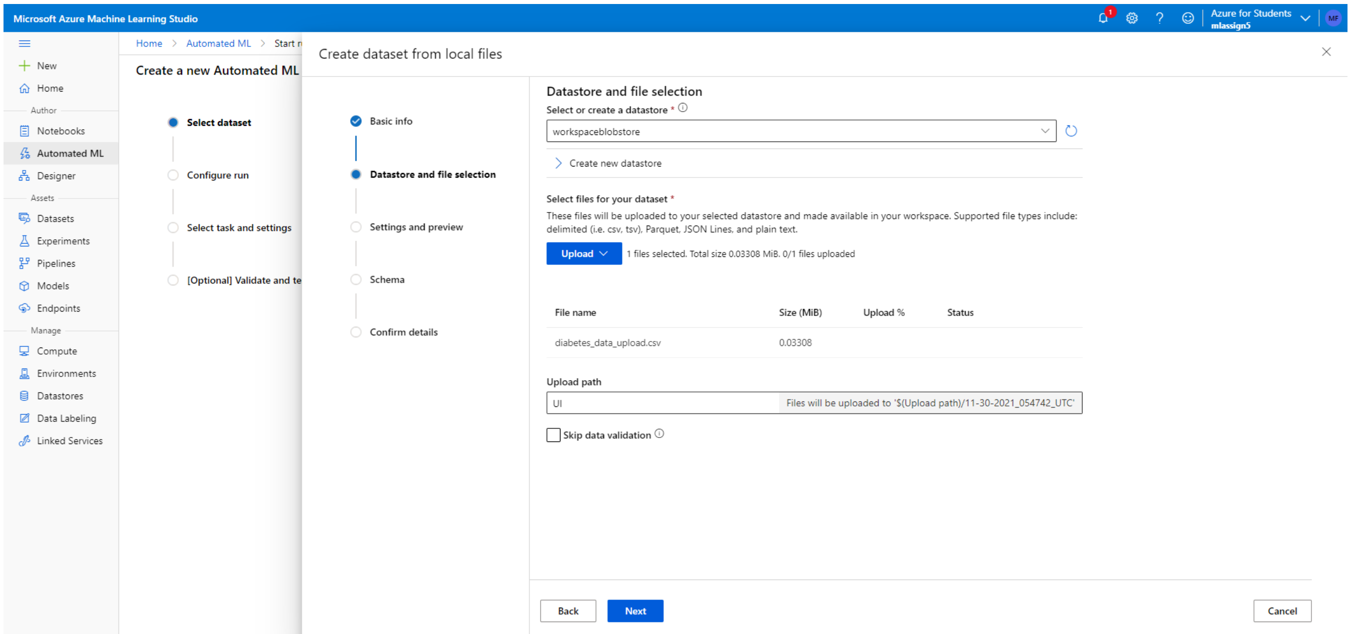
```
DecTree = DT(X_train, X_test, y_train, y_test)
```

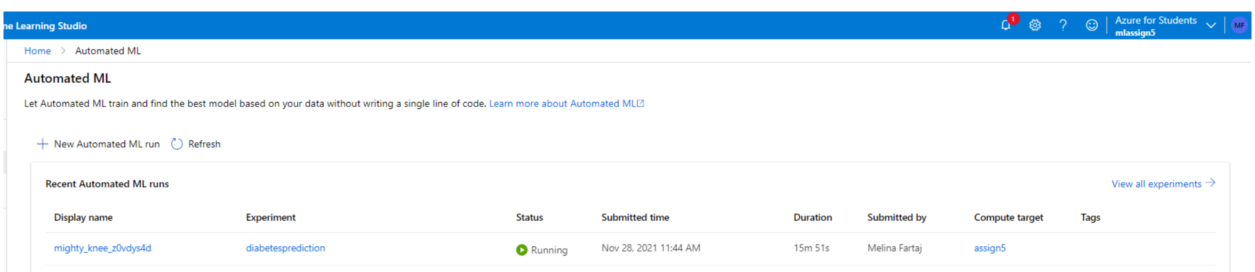
[27] ✓ <1 sec

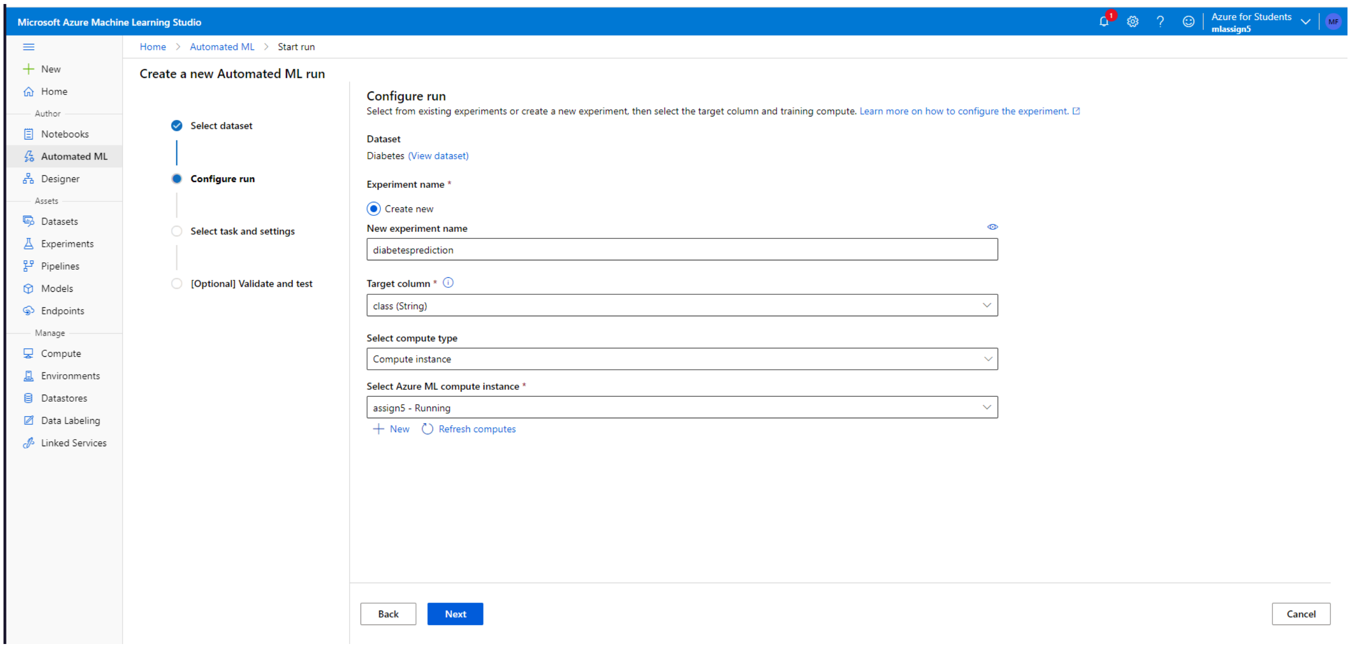
```
Accuracy of Decision Tree: 0.9423076923076923

Confusion Matrix of Decision Tree:
 [[51  3]
 [ 6 96]]
```

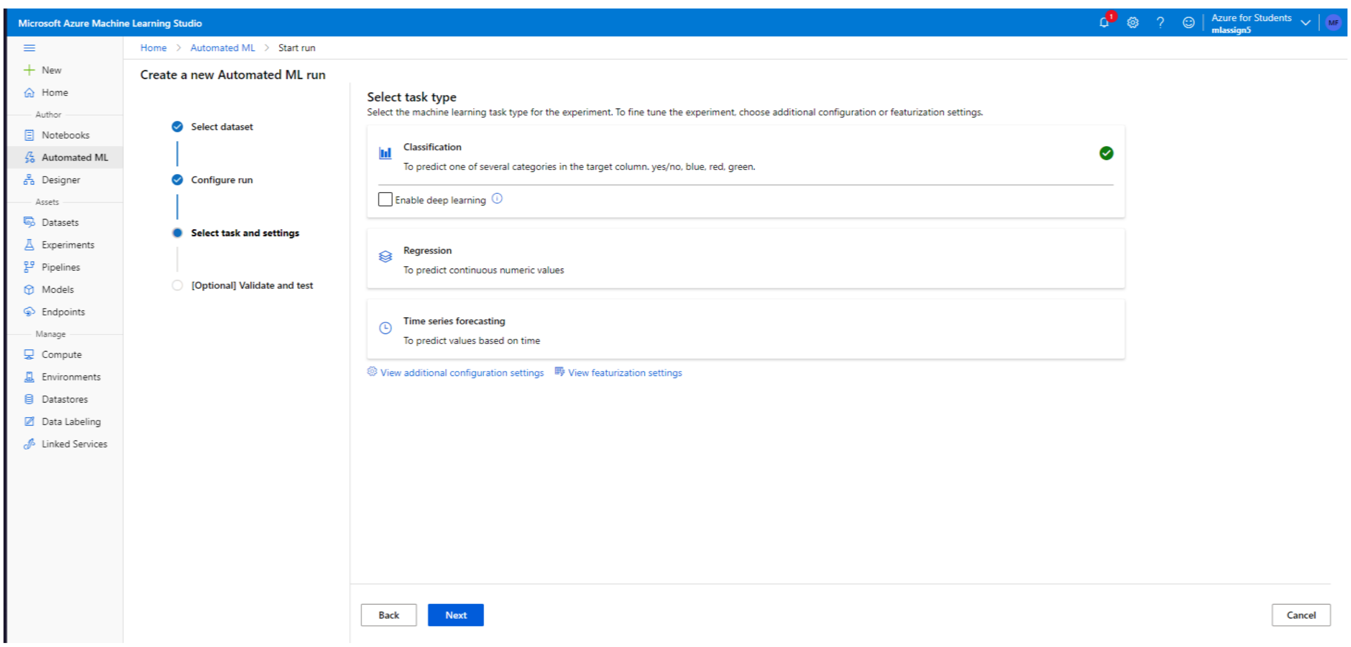
5. Use Automated ML for your data set. Explain best model results.

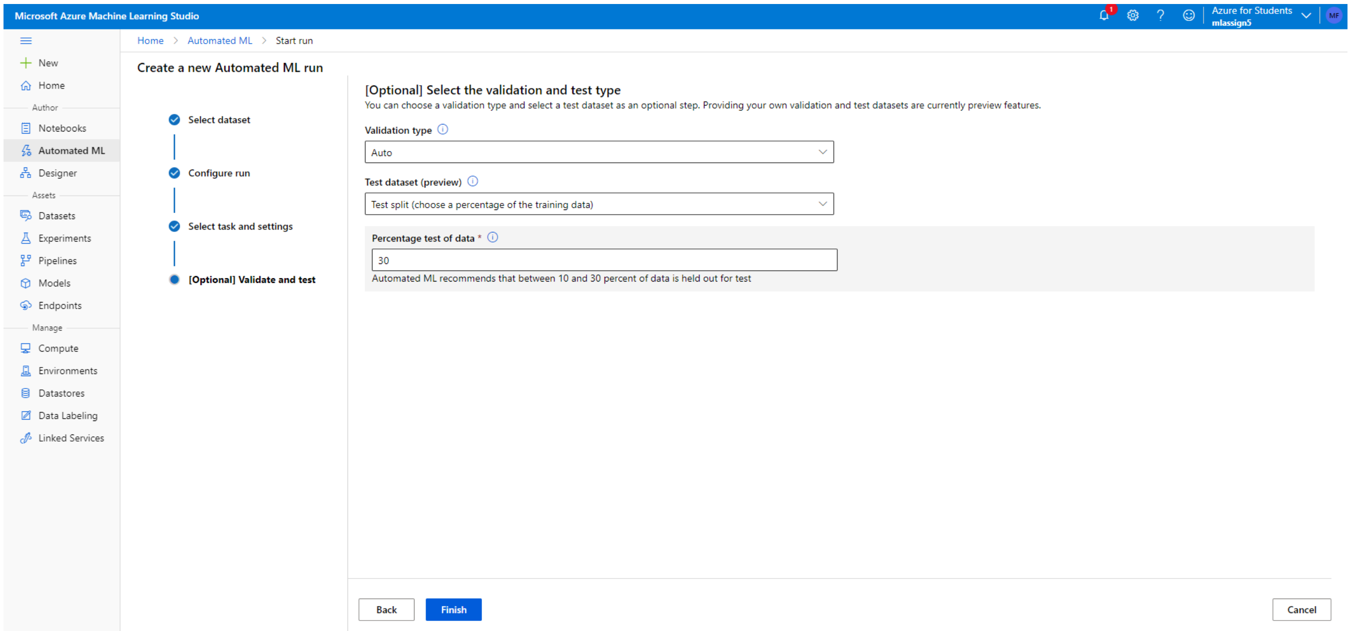
Below you can find the screenshots for the process of created an automated ML for my dataset. The process consisted of uploading my dataset, creating a cluster instance, selecting a classification algorithm, splitting the data and then running the model. In the screenshots you can see the different models that were run as well as their accuracy. The best model was found to be ‘VotingEnsemble’ which had an accuracy of 0.967. The top 4 features for this model were Polyuria, Polydipsia, Gender and Itching. Other success metrics can be noted in the screenshots below as well.

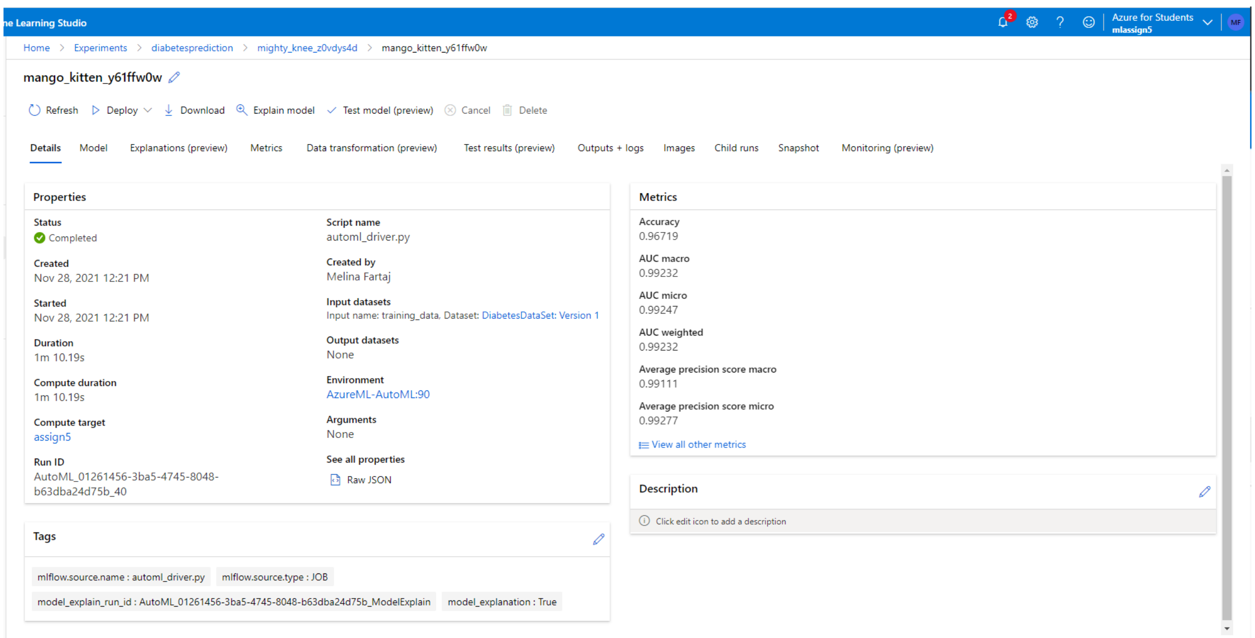


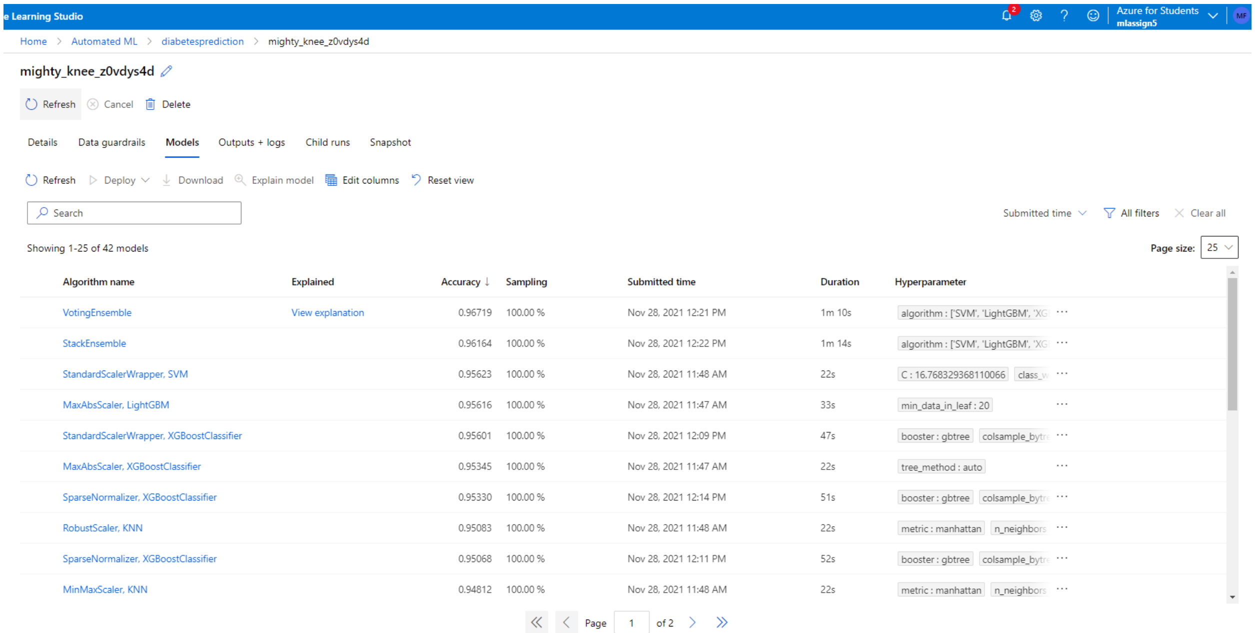


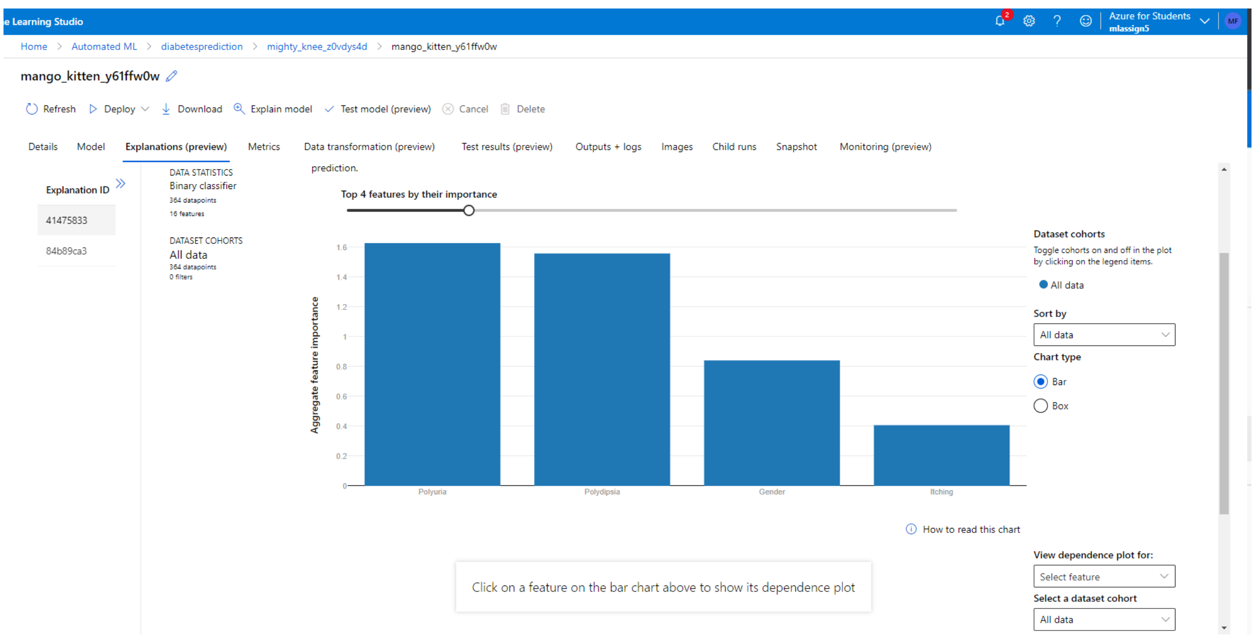

Microsoft Azure Machine Learning Studio
Azure for Students
mlassign5
Home
Automated ML
Start run
New
Home
Author
Notebooks
Automated ML
Designer
Assets
Datasets
Experiments
Pipelines
Models
Endpoints
Manage
Compute
Environments
Datastores
Data Labeling
Linked Services
Create a new Automated ML run
Select dataset
Configure run
Select task and settings
[Optional] Validate and test
Configure run
Select from existing experiments or create a new experiment, then select the target column and training compute. Learn more on how to configure the experiment.
Dataset
Diabetes (View dataset)
Experiment name *
Create new
New experiment name
diabetesprediction
Target column *
class (String)
Select compute type
Compute instance
Select Azure ML compute instance *
assign5 - Running
New
Refresh computes
Back
Next
Cancel

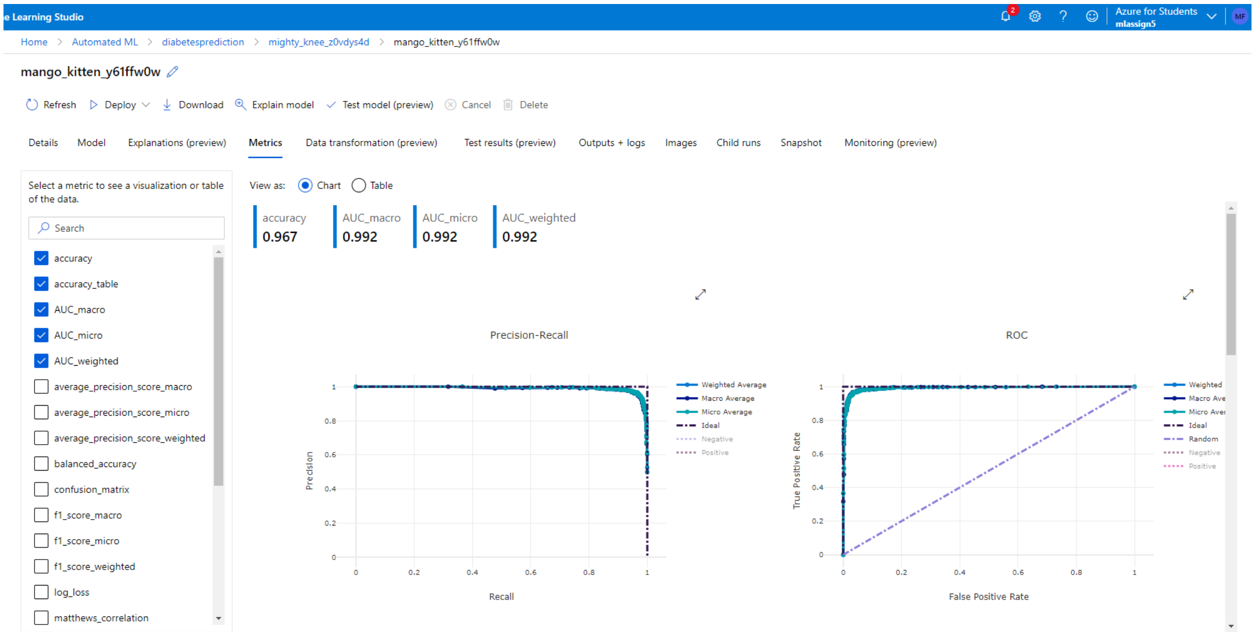

Microsoft Azure Machine Learning Studio
Azure for Students
mlassign5
Home
Automated ML
Start run
New
Home
Author
Notebooks
Automated ML
Designer
Assets
Datasets
Experiments
Pipelines
Models
Endpoints
Manage
Compute
Environments
Datastores
Data Labeling
Linked Services
Create a new Automated ML run
Select dataset
Configure run
Select task and settings
[Optional] Validate and test
Select task type
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.
Enable deep learning
Regression
To predict continuous numeric values
Time series forecasting
To predict values based on time
View additional configuration settings
View featurization settings
Back
Next
Cancel


Microsoft Azure Machine Learning Studio
Azure for Students
mlassign5
New
Home
Author
Notebooks
Automated ML
Designer
Assets
Datasets
Experiments
Pipelines
Models
Endpoints
Manage
Compute
Environments
Datastores
Data Labeling
Linked Services
Home > Automated ML > Start run
Create a new Automated ML run
Select dataset
Configure run
Select task and settings
[Optional] Validate and test
[Optional] Select the validation and test type
You can choose a validation type and select a test dataset as an optional step. Providing your own validation and test datasets are currently preview features.
Validation type
Auto
Test dataset (preview)
Test split (choose a percentage of the training data)
Percentage test of data *
30
Automated ML recommends that between 10 and 30 percent of data is held out for test
Back
Finish
Cancel


ne Learning Studio
Azure for Students
mlassign5
Home > Experiments > diabetesprediction > mighty_knee_z0vdys4d > mango_kitten_y61ffw0w
mango_kitten_y61ffw0w
Refresh
Deploy
Download
Explain model
Test model (preview)
Cancel
Delete
Details
Model
Explanations (preview)
Metrics
Data transformation (preview)
Test results (preview)
Outputs + logs
Images
Child runs
Snapshot
Monitoring (preview)
Properties
Status
Completed
Created
Nov 28, 2021 12:21 PM
Started
Nov 28, 2021 12:21 PM
Duration
1m 10.19s
Compute duration
1m 10.19s
Compute target
assign5
Run ID
AutoML_01261456-3ba5-4745-8048-b63dba24d75b_40
Script name
automl_driver.py
Created by
Melina Fartaj
Input datasets
Input name: training_data. Dataset: DiabetesDataSet: Version 1
Output datasets
None
Environment
AzureML-AutoML:90
Arguments
None
See all properties
Raw JSON
Metrics
Accuracy
0.96719
AUC macro
0.99232
AUC micro
0.99247
AUC weighted
0.99232
Average precision score macro
0.99111
Average precision score micro
0.99277
View all other metrics
Description
Click edit icon to add a description
Tags
mlflow.source.name : automl_driver.py
mlflow.source.type : JOB
model_explain_run_id : AutoML_01261456-3ba5-4745-8048-b63dba24d75b_ModelExplain
model_explanation : True


e Learning Studio
Azure for Students
mlassign5
Home > Automated ML > diabetesprediction > mighty_knee_z0vdys4d
mighty_knee_z0vdys4d
Refresh
Cancel
Delete
Details
Data guardrails
Models
Outputs + logs
Child runs
Snapshot
Refresh
Deploy
Download
Explain model
Edit columns
Reset view
Search
Submitted time
All filters
Clear all
Showing 1-25 of 42 models
Page size: 25
Algorithm name | Explained | Accuracy | Sampling | Submitted time | Duration | Hyperparameter
VotingEnsemble | View explanation | 0.96719 | 100.00 % | Nov 28, 2021 12:21 PM | 1m 10s | algorithm : ['SVM', 'LightGBM', 'XG
StackEnsemble | | 0.96164 | 100.00 % | Nov 28, 2021 12:22 PM | 1m 14s | algorithm : ['SVM', 'LightGBM', 'XG
StandardScalerWrapper, SVM | | 0.95623 | 100.00 % | Nov 28, 2021 11:48 AM | 22s | C : 16.768329368110066
MaxAbsScaler, LightGBM | | 0.95616 | 100.00 % | Nov 28, 2021 11:47 AM | 33s | min_data_in_leaf : 20
StandardScalerWrapper, XGBoostClassifier | | 0.95601 | 100.00 % | Nov 28, 2021 12:09 PM | 47s | booster : gbtree
MaxAbsScaler, XGBoostClassifier | | 0.95345 | 100.00 % | Nov 28, 2021 11:47 AM | 22s | tree_method : auto
SparseNormalizer, XGBoostClassifier | | 0.95330 | 100.00 % | Nov 28, 2021 12:14 PM | 51s | booster : gbtree
RobustScaler, KNN | | 0.95083 | 100.00 % | Nov 28, 2021 11:48 AM | 22s | metric : manhattan
SparseNormalizer, XGBoostClassifier | | 0.95068 | 100.00 % | Nov 28, 2021 12:11 PM | 52s | booster : gbtree
MinMaxScaler, KNN | | 0.94812 | 100.00 % | Nov 28, 2021 11:48 AM | 22s | metric : manhattan
Page 1 of 2


e Learning Studio
Azure for Students
mlassign5
Home > Automated ML > diabetesprediction > mighty_knee_z0vdys4d > mango_kitten_y61ffw0w
mango_kitten_y61ffw0w
Refresh
Deploy
Download
Explain model
Test model (preview)
Cancel
Delete
Details
Model
Explanations (preview)
Metrics
Data transformation (preview)
Test results (preview)
Outputs + logs
Images
Child runs
Snapshot
Monitoring (preview)
Explanation ID
41475833
84b89ca3
DATA STATISTICS
Binary classifier
364 datapoints
16 features
DATASET COHORTS
All data
364 datapoints
0 filters
prediction.
Top 4 features by their importance
Aggregate feature importance
Polyuria
Polydipsia
Gender
Itching
Dataset cohorts
Toggle cohorts on and off in the plot by clicking on the legend items.
All data
Sort by
All data
Chart type
Bar
Box
How to read this chart
View dependence plot for:
Select feature
Select a dataset cohort
All data
Click on a feature on the bar chart above to show its dependence plot



Learning Studio
Azure for Students
mlassign5
Home > Automated ML > diabetesprediction > mighty_knee_z0vdys4d > mango_kitten_y61ffw0w
mango_kitten_y61ffw0w
Refresh
Deploy
Download
Explain model
Test model (preview)
Cancel
Delete
Details
Model
Explanations (preview)
Metrics
Data transformation (preview)
Test results (preview)
Outputs + logs
Images
Child runs
Snapshot
Monitoring (preview)
Select a metric to see a visualization or table of the data.
Search
accuracy
accuracy_table
AUC_macro
AUC_micro
AUC_weighted
average_precision_score_macro
average_precision_score_micro
average_precision_score_weighted
balanced_accuracy
confusion_matrix
f1_score_macro
f1_score_micro
f1_score_weighted
log_loss
matthews_correlation
View as: Chart Table
accuracy 0.967
AUC_macro 0.992
AUC_micro 0.992
AUC_weighted 0.992
Precision-Recall
Precision
Recall
Weighted Average
Macro Average
Micro Average
Ideal
Negative
Positive
ROC
True Positive Rate
False Positive Rate
Random

2
Azure for Students
mlassign5
MF
Ensemble details
Select an ensemble algorithm to see the ensemble weights and hyperparameters.
SparseNormalizer, XGBoostClas...
StandardScalerWrapper, XGBoo...
RobustScaler, KNN
StandardScalerWrapper, SVM
MaxAbsScaler, XGBoostClassifier
MaxAbsScaler, LightGBM
Ensemble weight: 0.16666666666666666
Data transformation:
{
"class_name": "SparseNormalizer",
"module": "automl.client.core.common.model_wrappers",
"param_args": [],
"param_kwargs": {
"norm": "l2"
},
"prepared_kwargs": {},
"spec_class": "preproc"
}
Training algorithm:
{
"class_name": "XGBoostClassifier",
"module": "automl.client.core.common.model_wrappers",
"param_args": [],
"param_kwargs": {
"booster": "gbtree",
"colsample_bytree": 0.7,
"eta": 0.1,
"max_depth": 4,
"max_leaves": 0,
"n_estimators": 100,
"objective": "reg:logistic",
"reg_alpha": 0.