Microsoft

Principles of Machine Learning

Lab 5 - Optimization-Based Machine Learning Models

Overview

In this lab you will explore the use of optimization-based machine learning models. Optimization-based models are powerful and widely used in machine learning. Specifically, in this lab you will investigated:

- Neural network models for classification.
- Support vector machine models for regression.

What You'll Need

To complete this lab, you will need the following:

- An Azure ML account
- A web browser and Internet connection
- The lab files for this lab

Note: To set up the required environment for the lab, follow the instructions in the Setup Guide for this course.

In this lab you will build on the classification experiment you created in Lab 4. If you did not complete lab 4, or if you have subsequently modified the experiment you created, you can copy a clean starting experiment to your Azure ML workspace from the Cortana Intelligence Gallery using the link for your preferred programming language below:

- **R**: https://aka.ms/edx-dat203.2x-lab5-class-r
- **Python**: https://aka.ms/edx-dat203.2x-lab5-class-py

Classification with Neural Network Models

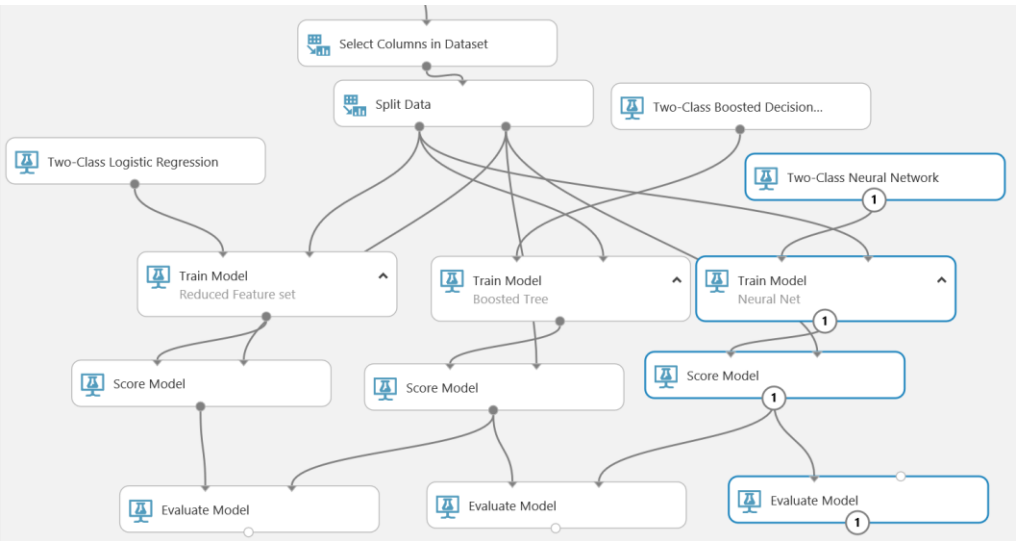
Neural networks are a widely used class of machine learning models. Neural network models can be used for classification or regression. In this lab, you will perform classification of the diabetes patients using a two-class neural network model.

In this exercise you will compare the performance of the neural network classifier to the Ada-boosted classifier you created in the previous lab.

Create a Neural Network Model

1. In Azure ML Studio, open your **Boosted Classification** experiment (or the corresponding starting experiment in the Cortana Intelligence Gallery as listed above), and save it as **Optimization-Based Classification**.

2. Add a **Two Class Neural Network** module to the experiment, and place it to the right of the existing modules.
3. Configure the **Two Class Neural Network** module as follows:
    - **Create trainer mode**: Parameter Range
    - **Hidden layer specification**: Fully-connected case
    - **Number of hidden nodes:** 100
    - **Use Range Builder (2):** Unchecked
    - **Learning rate**: 0.01, 0.02, 0.04
    - **Number of iterations**: 20, 40, 80, 160
    - **The initial learning weights diameter**: 0.1
    - **The momentum**: 0.01
    - **The type of normalizer**: Do not normalize
    - **Shuffle examples**: Checked
    - **Random number seed**: 123
    - **Allow unknown categorical levels**: Checked
4. Copy the **Train Model**, **Score Model**, and **Evaluate Model** modules that are currently used for the Boosted Tree model, and paste the copies into the experiment under the **Two Class Neural Network** module.
5. Edit the comment of the new **Train Model** module, and change it to *Neural Net*.
6. Connect the output of the **Two Class Neural Network** module to the **Untrained model** (left) input of the new *Neural Net* **Train Model** module. Then connect the left output of the **Split Data** module to the **Dataset** (right) input of the new *Neural Net* **Train Model** module.
7. Connect the output of the new *Neural Net* **Train Model** module to the **Trained Model** (left) input of the new **Score Model** module. Then connect the right output of the **Split Data** module to the **Dataset** (right) input of the new **Score Model** module.
8. Connect the output of the new **Score Model** module to the **Scored dataset to compare** (right) input of the existing **Evaluate Model** module to the left input of which the **Scored Model** module for the *Boosted Tree* model is already connected.
9. Connect the output of the new **Score Model** module to the **Scored dataset** (left) input of the new **Evaluate Model** module. Then ensure that the bottom portion of your experiment looks like this:



Compare Model Performance

1. Save and run the experiment.
2. When your experiment has finished running, visualize the output of the **Evaluate Model** module that is connected to both the *Boosted Tree* and *Neural Net* models, and examine the ROC curve. The **Scored dataset** (Blue) curve represents the *Boosted Tree* model, and the **Scored dataset to compare** (Red) curve represents the *Neural Net* model. The higher and further to the left the curve, the better the performance of the model.
3. Scroll down further in the visualization and examine the **Accuracy**, **Recall**, and **AUC** model performance metrics, which indicate the accuracy and area under the curve of the *Boosted Tree* model.
4. Visualize the output of the new **Evaluate Model** module that is connected to only the *Neural Net* model, and examine the **Accuracy**, **Recall**, and **AUC** model performance metrics, which indicate the accuracy and area under the curve of the new two-class neural network model. Compare this with the same metrics for the boosted tree model – the model with the higher metrics is performing more accurately. In particular; the lower the **Recall** metric, the higher the number of false negatives – which in this scenario represent an undesirable situation where patients are likely to be readmitted to the hospital unnecessarily.

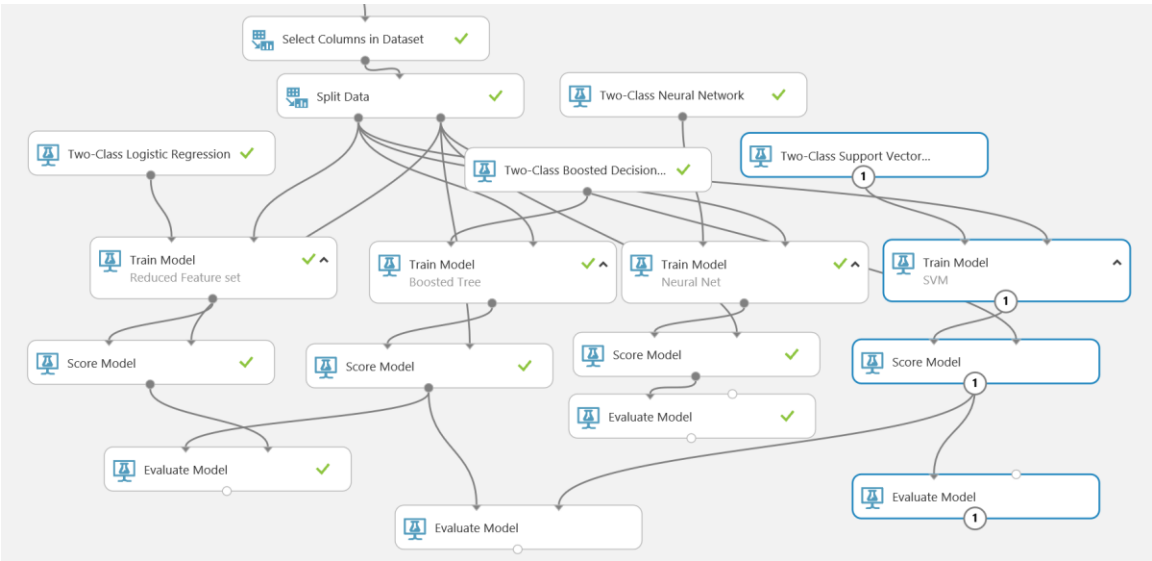
Support Vector Machine Classification

In the previous exercise you compared the performance of a neural network classifier to an Ada-boosted classifier. In this exercise, you will apply a support vector machine classifier to the diabetes patient dataset and compare the performance to the Ada-boosted decision tree classifier.

Create a Support Vector Machine Model

1. In Azure ML Studio, open your **Boosted Classification** experiment (or the corresponding starting experiment in the Cortana Intelligence Gallery as listed above), and save it as **Optimization-Based Classification**.
2. Add a **Two Class Support Vector Machine** module to the experiment, and place it to the right of the existing modules.
3. Configure the **Two Class Support Vector Machine** module as follows:
    - **Create trainer mode**: Parameter Range
    - **Number of iterations**: Use 1, 10, 100
    - **Lambda**: 0.00001, 0.0001, 0.001, 0.1
    - **Normalize features**: Unchecked
    - **Project to the unit-sphere**: Unchecked
    - **Random number seed**: 123
    - **Allow unknown categorical levels**: Checked
4. Copy the **Train Model**, **Score Model**, and **Evaluate Model** modules that are currently used for the *Neural Net* model, and paste the copies into the experiment under the **Two Class Support Vector Machine** module.
5. Edit the comment of the new **Train Model** module, and change it to *SVM*.
6. Connect the output of the **Two Class Support Vector Machine** module to the **Untrained model** (left) input of the new *SVM* **Train Model** module. Then connect the left output of the **Split Data** module to the **Dataset** (right) input of the new *SVM* **Train Model** module.

7. Connect the output of the new *SVM* **Train Model** module to the **Trained Model** (left) input of the new **Score Model** module. Then connect the right output of the **Split Data** module to the **Dataset** (right) input of the new **Score Model** module.
8. Connect the output of the new **Score Model** module to the **Scored dataset to compare** (right) input of the existing **Evaluate Model** module to the left input of which the **Scored Model** module for the *Boosted Tree* model is already connected – this will replace the connection from the *Neural Net* model.
9. Connect the output of the new **Score Model** module to the **Scored dataset** (left) input of the new **Evaluate Model** module. Then ensure that the bottom portion of your experiment looks like this:



Compare Model Performance

1. Save and run the experiment.
2. When your experiment has finished running, visualize the output of the **Evaluate Model** module that is connected to both the *Boosted Tree* and *SVM* models, and examine the ROC curve. The **Scored dataset** (Blue) curve represents the *Boosted Tree* model, and the **Scored dataset to compare** (Red) curve represents the *SVM* model. The higher and further to the left the curve, the better the performance of the model.
3. Scroll down further in the visualization of the down and examine the **Accuracy**, **Recall**, and **AUC** model performance metrics, which indicate the accuracy and area under the curve of the *Boosted Tree* model.
4. Visualize the output of the new **Evaluate Model** module that is connected to only the *SVM* model, and examine the **Accuracy**, **Recall**, and **AUC** model performance metrics, which indicate the accuracy and area under the curve of the new two-class neural network model. Compare this with the same metrics for the boosted tree model – the model with the higher metrics is performing more accurately. In particular; the lower the **Recall** metric, the higher the number of false negatives – which in this scenario represent an undesirable situation where patients are likely to be readmitted to the hospital unnecessarily.

Summary

In this experiment you have created and evaluated classifiers using two widely used optimization-based machine learning models:

- The neural network classifier.
- The support vector machine classifier.

Note: In this lab, you should have been able to determine the classification model type that worked best for the features and labels in the diabetes classification dataset. However, when you approach any other dataset there is no reason to believe that any particular machine learning model will have the best performance. Testing and comparing multiple machine learning models on a given problem is usually the best approach.

The performance achieved with any particular machine learning model can change after performing feature engineering. After performing a feature engineering step, it is usually a good idea to test and compare several machine learning models.