Microsoft

Principles of Machine Learning

Lab 4 - Tree-Based Models

Overview

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

- Using ADA-boosted tree models for classification.
- Using decision forest 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.

Classification with Ada-Boosted Tree Models

Ada-boosted tree models are a powerful ensemble machine learning model. Ada-boosted tree models can be used for classification or regression. In this lab, you will perform classification of the diabetes patients using a two-class boosted tree model.

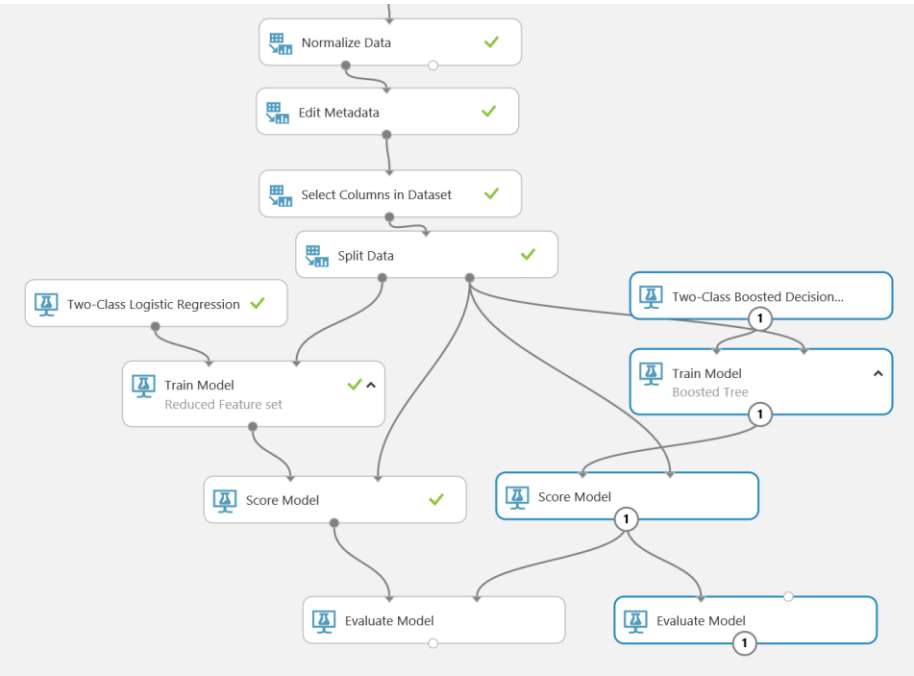
Note: In this lab you will build on the diabetes classification experiment you optimized in Lab 3. If you did not complete this lab, or if you have subsequently modified the experiment you created, you can copy clean starting experiments to your Azure ML workspace from the Cortana Intelligence Gallery using the links for your preferred programming language below:

- **R**: https://aka.ms/edx-dat203.2x-lab4-class-r
- **Python**: https://aka.ms/edx-dat203.2x-lab4-class-py

Create a Boosted Decision Tree Model

1. In Azure ML Studio, open your **Optimized Diabetes Classification** experiment (or the corresponding starting experiment in the Cortana Intelligence Gallery as listed above), and save it as **Boosted Classification**.
2. Add a **Two Class Boosted Decision Tree** module to the experiment.

3. Copy the **Train Model**, **Score Model**, and **Evaluate Model** modules, and paste the copies into the experiment under the **Two Class Boosted Decision Tree** module:
4. Edit the comment of the new **Train Model** module, and change it to *Boosted Tree*.
5. Connect the output of the **Two Class Boosted Decision Tree** module to the **Untrained model** (left) input of the new *Boosted Tree* **Train Model** module. Then connect the left output of the **Split Data** module to the **Dataset** (right) input of the new *Boosted Tree* **Train Model** module.
6. Connect the output of the new *Boosted Tree* **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.
7. Connect the output of the new **Score Model** module to the **Scored dataset to compare** (right) input of the original **Evaluate Model** module (the left input of which the **Scored Model** module for the original linear regression model is already connected).
8. 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 original **Evaluate Model** module and examine the ROC curve. The **Scored dataset** (Blue) curve represents the original Linear Regression model, and the **Scored dataset to compare** (Red) curve represents the two-class boosted decision tree 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** and **AUC** model performance metrics, which indicate the accuracy and area under the curve of the original linear regression model.
4. Visualize the output of the new **Evaluate Model** module and examine the **Accuracy** and **AUC** model performance metrics, which indicate the accuracy and area under the curve of the new

two-class boosted decision tree model. Compare this with the same metrics for the linear regression model – the model with the higher metrics is performing more accurately.

Decision Forest Regression

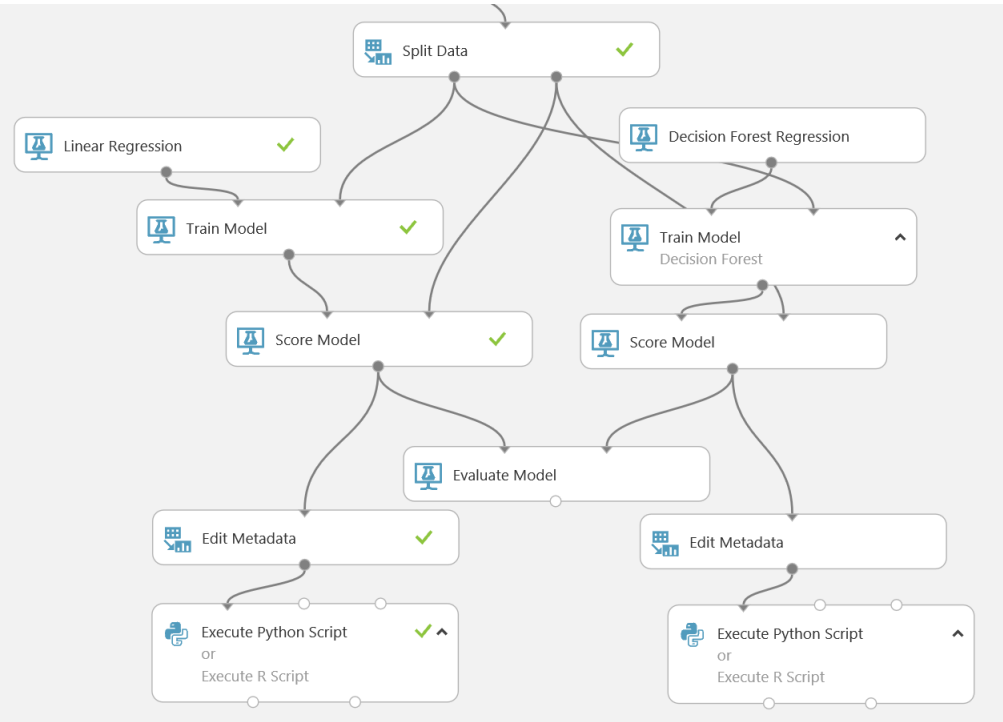
In Lab 2 you worked with a linear regression model to forecast bike demand by hour. The results of that exercise, showed that bike demand was consistently under estimated using a linear model. Decision forest regression is a non-linear tree-based regression machine learning method. In this lab, you will perform some feature engineering, with either R or Python, and use a decision forest regression model to predict bike demand.

Note: In this lab you will build on the bike rental linear regression experiment you created in Lab 2. If you did not complete this lab, or if you have subsequently modified the experiment you created, you can copy clean starting experiments to your Azure ML workspace from the Cortana Intelligence Gallery using the links for your preferred programming language below:

- **R**: https://aka.ms/edx-dat203.2x-lab4-regr-r
- **Python**: https://aka.ms/edx-dat203.2x-lab4-regr-py

Create a Decision Forest Regression Model

1. In Azure ML Studio, open your **Bike Regression** experiment (or the corresponding starting experiment in the Cortana Intelligence Gallery as listed above), and save it as **Decision Forest Regression**.
2. Add a **Decision Forest Regression** module to the experiment, and review its default properties.
3. Copy the following modules from the bottom of the experiment, and paste the copies into the experiment under the **Decision Forest Regression** module:
    - Train Model
    - Score Model
    - Edit Metadata
    - The final Execute R/Python Script
4. Edit the comment of the new **Train Model** module, and set it to *Decision Forest*.
5. Connect the output of the **Decision Forest Regression** module to the **Untrained model** (left) input of the new *Decision Forest* **Train Model** module. Then connect the left output of the **Split Data** module to the **Dataset** (right) input of the new *Decision Forest* **Train Model** module.
6. Connect the output of the new *Decision Forest* **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.
7. Connect the output of the new **Score Model** module to the **Scored dataset to compare** (right) input of the original **Evaluate Model** module (the left input of which the **Scored Model** module for the original linear regression classification model is already connected).
8. Edit the copied **Edit Metadata** module to change the selected column to **Score Label Mean** (which is the name of the predicted column returned by the Decision Forest module)
9. 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 and examine the metrics for both models. The module with the lowest values for **Mean Absolute Error**, **Root Mean Squared Error**, **Relative Absolute Error**, and **Relative Squared Error**; and the highest value for **Coefficient of Determination** is performing best.
3. Visualize the **device** (right) output of the final **Execute R/Python Script** module and review the residuals plots. Compare these to the residual plots produced by the linear regression model and use them to determine which model predicts bike rental counts most accurately.

Summary

In this experiment you have done the following:

- Created and evaluated a two-class classifier for the diabetes dataset using an Ada-boosted tree machine learning model.
- Created and evaluated a decision forest regression model for predicting bike demand.