LESSON 4: Supervised & Unsupervised Learning

Section 1: Lesson Overview

This lesson covers two of Machine Learning's fundamental approaches: supervised and unsupervised learning.

First, we'll cover **supervised learning**. Specifically, we'll learn:

- More about classification and regression, two of the most representative supervised learning tasks
- Some of the major algorithms involved in supervised learning, as well as how to evaluate and compare their performance
- How to use automated ML to automate the training and selection of classifiers and regressors, and how to use the Designer in Azure Machine Learning Studio to create automated Machine Learning experiments

Next, the lesson will focus on **unsupervised learning**, including:

- Its most representative learning task, clustering
- How unsupervised learning can address challenges like lack of labeled data, the curse of dimensionality, overfitting, feature engineering, and outliers
- An introduction to representation learning
- How to train your first clustering model in Azure Machine Learning Studio

Section 3: Supervised Learning - Classification

In a **classification problem**, the **outputs are categorical or discrete**.

Within this broad definition, there are several main approaches, which differ based on how many classes or categories are used, and whether each output can belong to only one class or multiple classes.

Some of the most **common types of classification problems** include:

- **Classification on tabular data**: The data is available in the form of rows and columns, potentially originating from a wide variety of data sources.
- **Classification on image or sound data**: The training data consists of images or sounds whose categories are already known.
- **Classification on text data**: The training data consists of texts whose categories are already known.

ML requires **numerical data**. This means that with images, sound, and text, several steps need to be performed during the preparation phase to transform the data into numerical vectors that can be accepted by the classification algorithms.

Examples of classification problems include:

- Computer vision
- Speech recognition
- Biometric identification
- Document classification
- Sentiment analysis
- Credit scoring in banking
- Anomaly detection

Categories of Algorithms

Three **main categories of classification problems**:

- **Two class/Binary classification**: the algorithm predict the output between 2 categories.
- **Multi-Class Single-Label classification**: there are more than 2 output categories and each output belongs to a single category
- **Multi-Class Multi-Label classification**: there are more than 2 output categories and each output can belong to one or more categories

Specific algorithms are used for each problem category.

1) Algorithms used for **Two class/Binary classification**:

Algorithm	Characteristics
Two-Class Support Vector Machine	Under 100 features, linear model
Two-Class Averaged Perceptron	Fast training times, linear model
Two-Class Decision Forest	Accurate, fast training times
Two-Class Logistic Regression	Fast training times, linear model
Two-Class Boosted Decision Tree	Accurate, fast training, large memory footprint
Two-Class Neural Network	Accurate, long training times

2) Algorithms used for **Multi-class classification**:

Algorithm	Characteristics
Multi-Class Logistic Regression	Fast training times, linear model
Multi-Class Neural Network	Accurate, long training times
Multi-Class Decision Forest	Accurate, fast training times
Multi-Class Boosted Decision Tree	Non-parametric, fast training times, and scalable
One-vs-All Multiclass	Depends on the underlying two-class classifier

The **One-vs-All Multiclass algorithm** is used for multi-class single-label problems. It trains a **set of binary classifiers** where each classifier is trained to predict one of the output classes. The class with the highest probability or a score across all the binary classifiers will be the predicted class.

Section 4: Lab (Compare the performance of various 2-class classifers)

In this lab, we will be compare the performance of two binary classifiers: Two-Class Boosted Decision Tree and Two-Class Logistic Regression for predicting customer churn. The goal is to run an expensive marketing campaign for high risk customers; thus, the precision metric is going to be key in evaluating performance of these two algorithms. We will do all of this from the Azure Machine Learning designer without writing a single

line of code. You can learn more about the evaluation metrics for the classification algorithm used in this lab looking here (https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/evaluate-model#bkmk_classification)

Section 7: Multi-Class Algorithms

Different types:

- **Multi-Class Logistic Regression**. The 2 key parameters for configuring this algorithm are:
  - **Optimization Tolerance** (it controls when to stop the iterations. If the improvement between iterations is < specified threshold, the algorithm stops and returns a current model)
  - **Regularisation Weight** (regularization is a method for preventing overfitting by penalising models with extreme coefficient values; the weight controls how much to penalise the models at each iteration)

- **Multi-Class Neural Network**. One typical example includes the input layer, one hidden and the output layer. The relationship between input and output is learned by training the neural network on the input data. The 3 key parameters for configuring this algorithm are:
  - **Number of Hidden Nodes**
  - **Learning Rate** (which controls the size of the step taken at each iteration before correction
  - **Number of Learning Iterations** (the max number of times the algorithm should process the training cases)

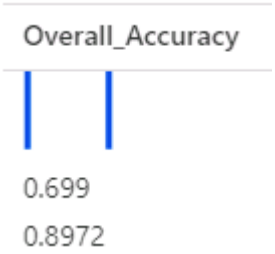
- **Multi-Class Decision Forest**. It is an ensemble of decision trees (the algorithm builds multiple decision trees and the votes the most popular output class). The 5 key parameters for configuring this algorithm are:
  - **Resampling Method** (it controls the method used to create individual trees)
  - **Number of decision trees** (it specifies the max number that can be created in the ensemble)
  - **Max Depth of the decision trees**
  - **Number of Random Splits per Node** (i.e. the number of splits to use when building each node of the tree)
  - **Min Number of Samples per Leaf Node** (it controls the min N of cases required to create any terminal node in the tree)

Section 8: Lab (Multi-Class Classifiers Performance)

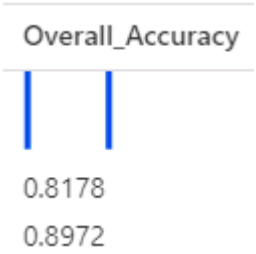
In this lab, we will be compare the performance of two different multiclass classification approaches: **Two-Class Support Vector Machine used with One-vs-All Multiclass module** vs **Multiclass Decision Forest**. We will apply the two approaches for the **letter recognition problem** and compare their performance. We will do all of this from the Azure Machine Learning designer without writing a single line of code. You can learn more about the One-vs-All Multiclass module used in this lab by looking here (https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/one-vs-all-multiclass)

Conclusions

- The Two-Class Support Vector Machine algorithm is extended for multiclass classification problem by using the One-vs-All Multiclass module.
- The native multiclass algorithm Multiclass Decision Forest outperforms the Two-Class Support Vector Machine across all key performance metrics (e.g. for Overall accuracy: 0.8972 vs 0.699)



- One recommendation for next steps is to increase the Number of iterations parameter for the Two-Class Support Vector Machine module to an higher value like 100 and observe its impact on the performance metrics --> the performance of the Two-Class Support Vector Machine has increased now to 0.8178:

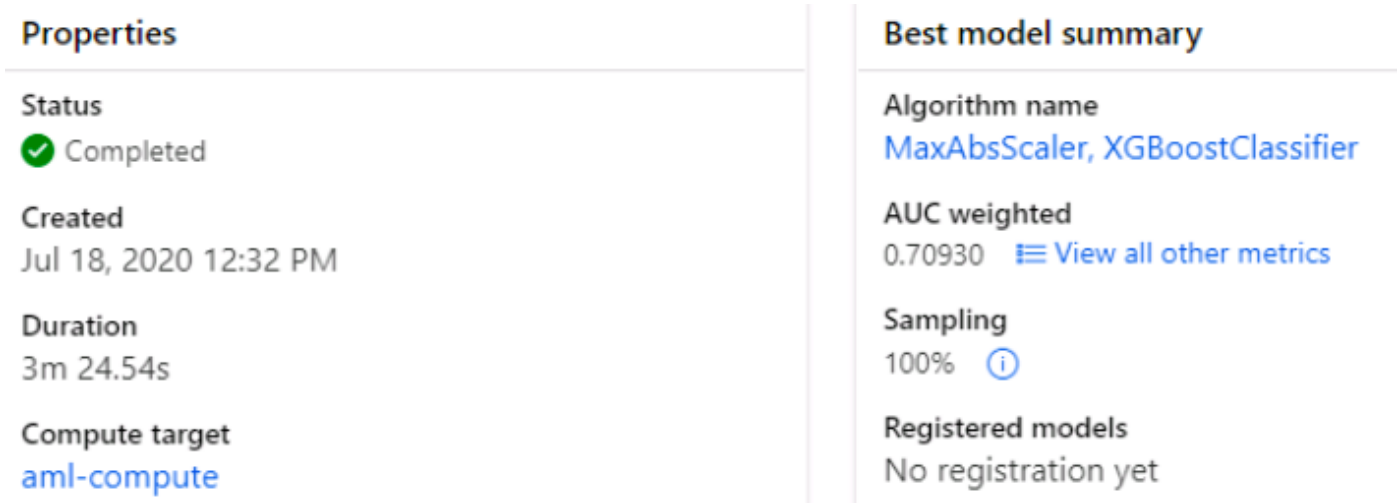


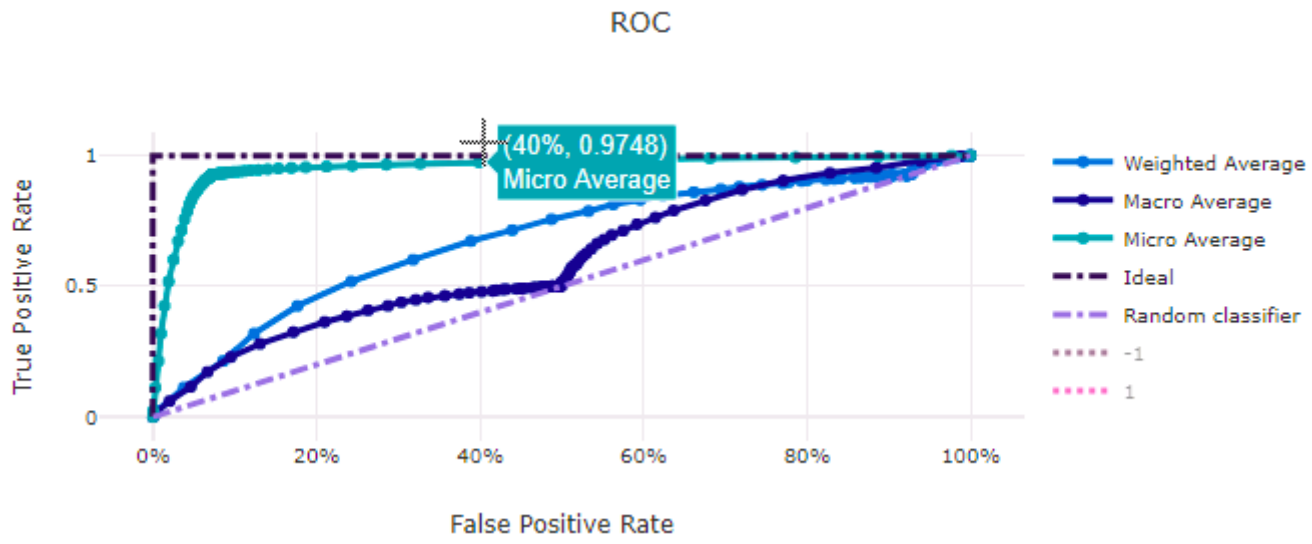
Section 11: Lab (Training a Classifier using Automated ML)

Automated machine learning picks an algorithm and hyperparameters for you and generates a model ready for deployment. There are several options that you can use to configure automated machine learning experiments.

In this lab, we will use Automated Machine Learning to find the **best performing binary classification model for predicting customer churn**.

Learn to view and understand the charts and metrics for your automated machine learning run by selecting this link (https://docs.microsoft.com/en-us/azure/machine-learning/how-to-understand-automated-ml#classification)





Section 14: Supervised Learning - Regression

In a **regression** problem, the **output** is **numerical or continuous**.

Common **types of regression problems** include:

- **Regression on tabular data**: The data is available in the form of rows and columns, potentially originating from a wide variety of data sources.
- **Regression on image or sound data**: Training data consists of images/sounds whose numerical scores are already known. Several steps need to be performed during the preparation phase to transform images/sounds into numerical vectors accepted by the algorithms.
- **Regression on text data**: Training data consists of texts whose numerical scores are already known. Several steps need to be performed during the preparation phase to transform text into numerical vectors accepted by the algorithms.

Examples of regression problems:

- Prediction of Housing prices
- Prediction of Customer churn
- Prediction of Customer lifetime value
- Forecasting (time series)
- Anomaly detection

Categories of Algorithms:

Common machine learning algorithms for regression problems include:

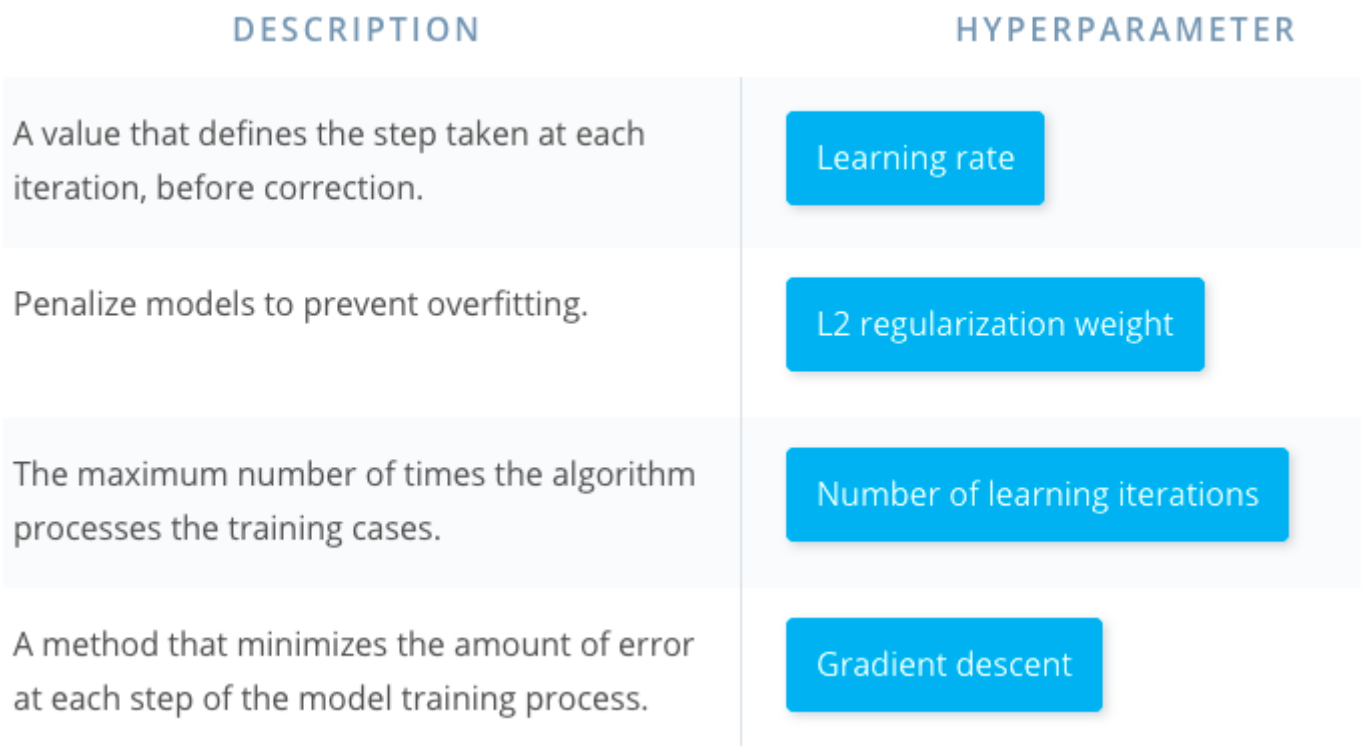
- **Linear Regression**: Fast training, **linear model**. This algorithms attempts to establish a linear relationship between one or more independent vars and a numeric outcome. There are 2 popular approaches to measure the error and fit the regression line:
  - **Ordinary least squares method** (it computes error as a sum of the squares of distances from the actual value to the predicted line and fits the model by minimising the squared error. It assumes a strong linear relationship between the inputs and the dependent var)
  - **Gradient descent** (the approach is to minimise the amount of error at each step of the model training process)

- **Decision Forest Regression**: Accurate, fast training times.

- It is an **ensemble learning** method for regression.
- The algorithm builds **multiple decision trees** and each tree in a decision forest outputs a **distribution as a prediction**.
- An **aggregation** is performed over the endemble of trees to find a distribution a distribution closest to the combined distribution for all trees in the model.
- The algorithm supports some of the same hyperparameters discussed for the multi-class decision forest algorithm.

- **Neural Net Regression**: Accurate, long training times.
  - It is a **supervised learning method** (thus requiring a tag dataset with a label column to identify the output)
  - The **label column** must be a **numerical data type**.
  - A default architecture is a **fully connected neural net** with **1 input layer + 1 hidden layer + 1 output layer**.
  - The algorithm supports the same hyperparameters discussed for the multi-class neural network algorithm.

Note, exercise 2 should have these answers as correct:

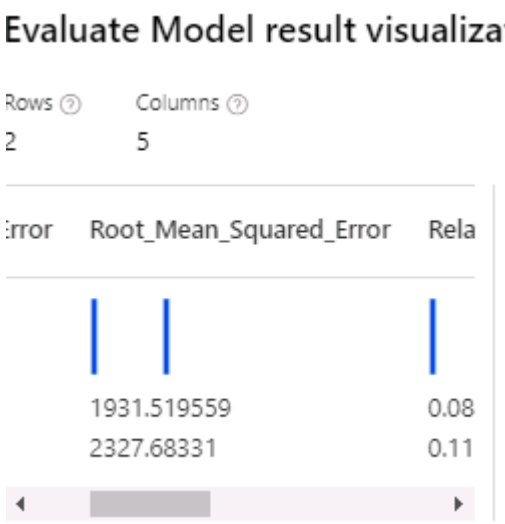


Section 15: Lab (Regressor Performance)

In this lab, we will be **compare the performance of two regression algorithms**: **Boosted Decision Tree Regression** and **Neural Net Regression** for predicting automobile prices. We will do all of this from the Azure Machine Learning designer.

You can learn more about the evaluation metrics for the regression algorithm used in this lab by selecting this link (https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/evaluate-model#bkmk_regression)

Select the regression performance metric Root_Mean_Squared_Error and compare performance of the two algorithms: Boosted Decision Tree Regression (1931.519559) and Neural Net Regression (2327.68331). Note that smaller value for Root_Mean_Squared_Error implies better performance:



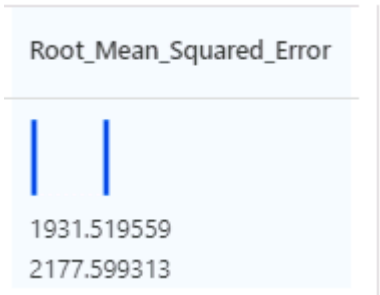
Based on the performance metric, **Root_Mean_Squared_Error**, it shows that the Boosted Decision Tree Regression algorithm outperforms the Neural Net Regression algorithm. One recommendation for next steps is to tune the hyperparameters for the Neural Net Regression module to see if we can improve its performance. [MS Azure Neural Network Regression (https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/neural-network-regression)](https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/neural-network-regression)

Initial hyperparameters for Neural net:

- Number of hidden nodes = 1000
- Learning Rate = 0.0001
- Number of learning iterations = 10,000
- Random number seed = 139

After changing:

- Number of learning iterations = 20,000 --> the RMSE has slightly decreased (to 2177.6) for the neural net (i.e. slight improvement)



Section 18: Automate the Training of a Regressor

In a conventional ML process there a few challenges that can be encountered:

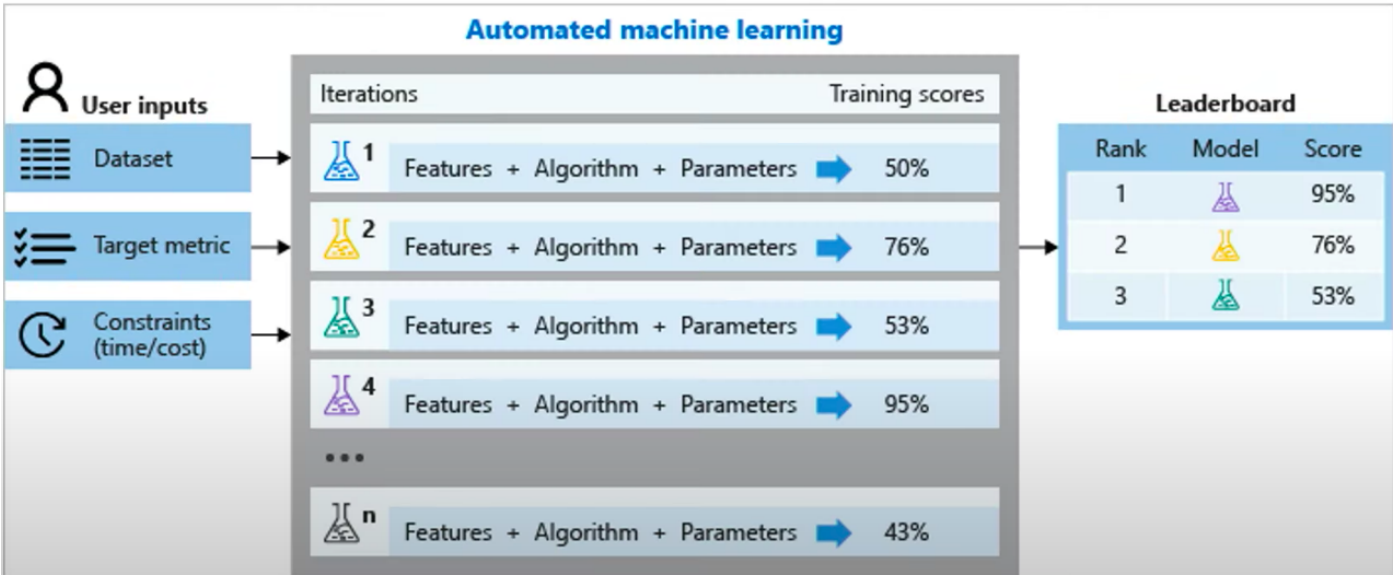
- Features selection
- Choice of an algorithm suitable for the task
- Hyperparameters' tuning for the selected algorithm
- Selection of appropriate evaluation metrics to measure the performance of the trained model This entire process is pretty iterative.

The idea behind **automated ML** is to enable the automated exploration of the combinations needed to successfully produce a trained model. It intelligently tests multiple combinations of algorithms and hyperparamters in parallel and returns the best one. Automated ML gives users the option to automatically

scale and normalize input features. It also gives users the ability to enable additional featurization, such as missing values imputation, encoding, and transforms.

This allows to build ML models with high-scale efficiency and productivity all while sustaining model quality. the resulting models can be either deployed into production or further refined and customised.

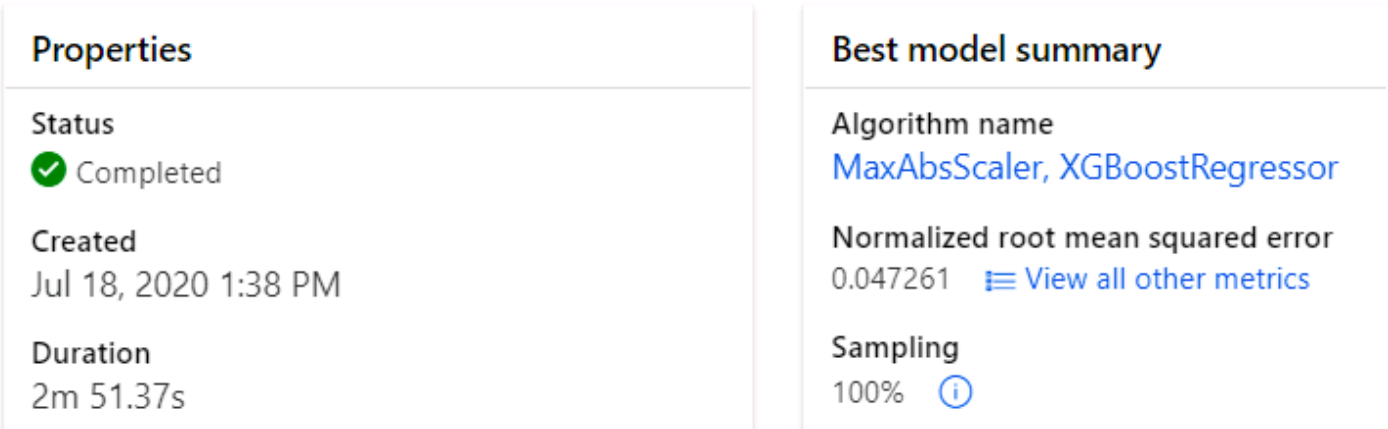
Azure ML Studio makes it easy to set up an Auto ML experiment and it is a no code experience.
1) The user needs to provide as inputs the **training dataset**, the **primary evaluation metrics** and specify the **exit criteria** (**time-bound** or **performance-based**).
2) The experiment is run and the **Auto ML engine will explore the combination of features, algorithms and hyperparameters and score each training pipeline on the primary metric**. You can compare the performance of various algorithms as AutoML continues its exploration.
3) When the experiment concludes, **Auto ML identifies the best performing model**. You can review the primary metric and a comprehensive set of other performance metrics and charts to further assess the model performance

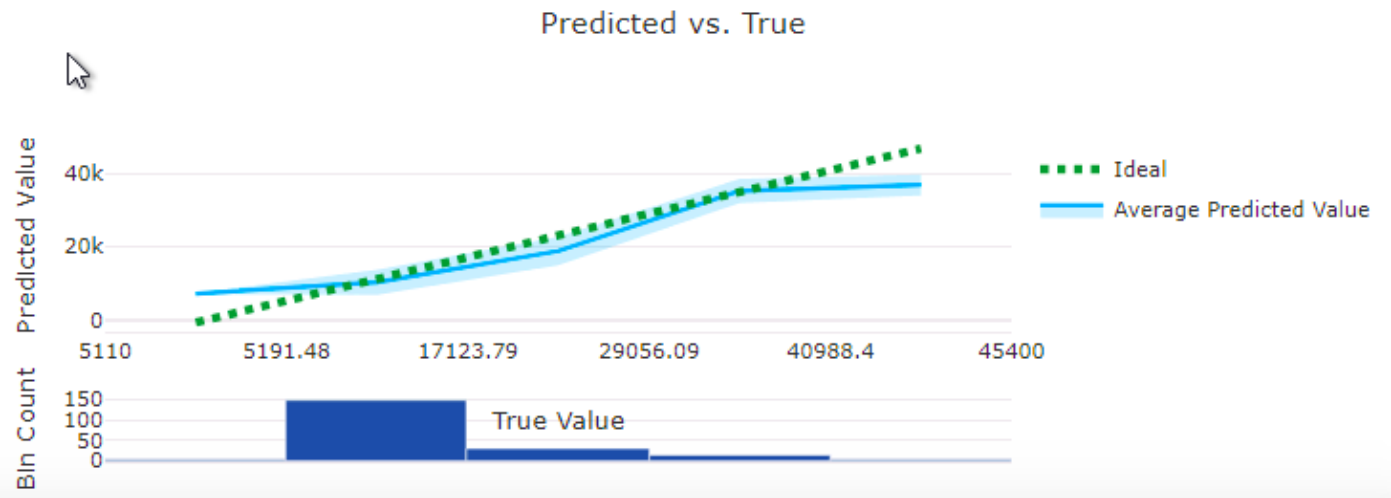


The default behavior in Auto ML to deal with missing values for categorical features is to impute the most frequent value.

Section 19: Lab (Train a Regressor using Auto ML)

In this lab, we will use Automated Machine Learning to **find the best performing regression model for predicting automobile prices**. You can learn more about how Automated Machine Learning offers preprocessing and data guardrails automatically by selecting this [link (https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-auto-features#automatic-featurization)](https://docs.microsoft.com/en-us/azure/machine-learning/how-to-configure-auto-features#automatic-featurization)





Section 21: Unsupervised Learning

All of the algorithms we have looked at so far are examples of supervised learning, in which the training data is labeled. But the cost of obtaining labeled data can be high. So now let's turn our attention to the second main type of machine learning explored in this lesson: unsupervised learning.

In **unsupervised learning**, algorithms learn from **unlabeled data** by looking for **hidden structures in the data**. The training process aims to identify common aspects between entities and then use thier presence/absence to produce various results.

Obtaining unlabeled data is comparatively **inexpensive** (the cost of labelled data is high and it is often non-scalable) and unsupervised learning can be used to uncover very useful information in such data. For example, we can use clustering algorithms to:

- **discover implicit grouping** within the data
- use association algorithms to **discover hidden rules** that are governing the data (e.g., people who buy product A also tend to buy product B)
- **dimensionality reduction**
- **feature extraction**
- **anomaly detection**

The types of unsupervised learning algorithms include **clustering**, **feature learning**, **neural networks**, **PCA**, **matrix factorisation** and **anomaly detection**.

Types of Unsupervised Learning Approaches

3 main types:

- **Clustering**: it organises entities from the input data into a finite number of subsets or clusters (e.g. clustering and tagging similar documents on the basis of their content)
- **Feature Learning (Representation Learning)**: it transforms sets of inputs into other inputs that are potentially more useful in solving a given problem (used especially for discovering better features in data with multiple categorical features with high cardinality)
- **Anomaly Detection**: it identifies 2 major groups of entities, normal and abnormal (anomalies) 9used for example in spam filtering or credit card fraud detection)

Examples of unsupervised learning algorithms: K-Means clustering, PCA, Autoencoders

Section 22: Semi-Supervised Learning

Sometimes fully labeled data cannot be obtained, or is too expensive—but at the same time, it may be possible to get **partially labeled data**. This is where semi-supervised learning is useful.

Semi-supervised learning combines the supervised and unsupervised approaches; typically it involves having **small amounts of labeled data and large amounts of unlabeled data**. This approach uses the advantages provided by the labeled part of the training data to make the unlabeled part useful.

Three approaches:

- **Self-training**: the model is trained with the labeled data and then used to make predictions for the unlabeled data. It ends with a fully-labeled dataset that can be used with a supervised learning approach.
- **Multi-view training**: multiple models are trained on different views of the data that include various feature selection, parts of training data or various model architectures.
- **Self-ensemble training**: similar to multi-view training, except you use a single base model and different hyperparameter settings

Section 24: Clustering

It is an unsupervised model approach. As the name suggests, **clustering** is the problem of organizing entities from the input data into a **finite number of subsets or clusters**. The goal is to

- **maximize intra-cluster similarity**
- **maximise inter-cluster differences**.

Clustering Algorithms

Several applications:

- **Personalisation and target marketing** (grouping of customers based on similar characteristics to develop target campaigns)
- **Document Classification** (cluster/tag similar documents based on their content, for example to use the tag improve document research or create digests/summaries)
- **Fraud detection** (to isolate new cases based on proximity with historical clusters that represent a fraudulent behaviour)
- **Medical Imaging** (e.g. to differentiate between different types of tissues in a 3D image)
- **City planning** (to help in identifying groups of houses according to their house type, value and geographical location)

Clustering algorithms can be broadly classfied into **4 subtypes**:

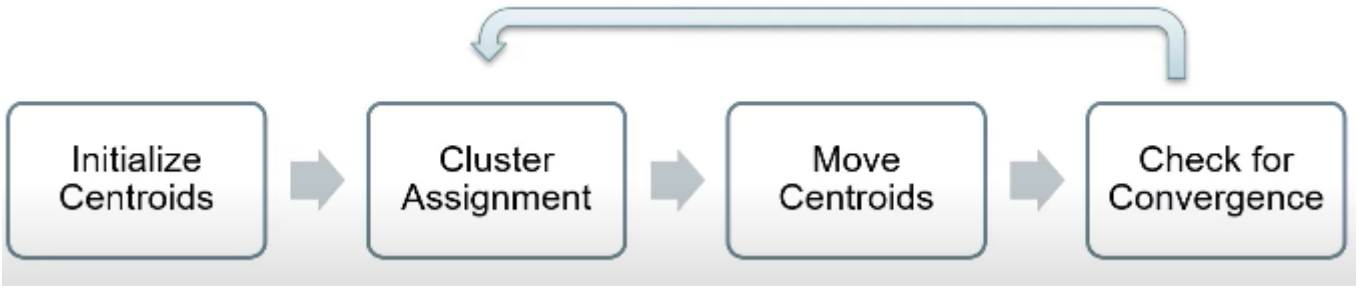
- **Centroid-based clustering** (it organises the data into clusters based on the distance of members from the centroid of the clusters; e.g. **K-means**)
- **Density-based clustering** (the algorithm clusters members that are closely packed together; this approach can learn clusters of arbitrary shapes)
- **Distribution-based clustering** (its underlying assumption is that the data has an inherent distribution type, e.g. normal distribution; the algorithm clusters depending on the probability of a member belonging to a particular distribution)
- **Hierarchical clustering** (the algorithm builds a tree of clusters; it si best suited for a hierarchical data, such as taxonomies)

K-Means Clustering

- **Centroid-based** unsupervised clustering algorithm
- It create up to a **target (K) numbers of clusters**, grouping similar members.
- Its objective is to **minimise intra-clusters distances** (i.e. the squared error of the distance between the members of the cluster and its centre).

Algorithm details:
1) **Initialise centroids**: Random initialisation of the K cluster centroids. Each centroid conceptually represents a cluster. 2) **Cluster assignment**: Based on the current centroid location, it will assign each member to a cluster, on the basis of the Euclidean distance of each member from the centroids. Members are assigned to the clusters represented by the closest centroid.
3) **Move centroids**: K-means will compute the new cluster centroids based on current cluster memebership. As aresult, the centroid locations are likely to change.
4) **Check for convergence**: different types of convergence criteria:

- Check how much did the centroid location change as a result of new cluster membership? If the total change in centroid location is < given tolerance, the algorithm will assume convergence and stop.
- On the basis of a fixed number of iterations.
  5) If the convergence criteri is not met, it goes back to step (2) and repeats.



There are many **configuration parameters**; among the most important ones:

- **N of centroids** (i.e. the N of clusters you want your algorithm to begin with; note that the model is not guaranteed to produce exactly this number of clusters as the algorithm starts with this N and iterates to find the optimal configuration)
- **Initialisation approach** to select the initial centroids (options are **first n random** or **K-means++** algorithm)
- **Distance metric**. Default is **Euclidean**.
- **Normalise features**: it uses the **min/max normaliser** to scale the numeric data point into [0,1].
- **Assign label mode**: it can be used only if your dataset has already a label column. You can optionally use the label values to guide the selection of the clusters. Another use of the label col is to fill missing values (If any row is missing a label, its value is imputed by using other features). You can also ask the K-mean module to fix the label col values by replacing them with the predicted label values using the label of the point that is closest to the current centroid.
- **N of iterations**: i.e. the N of times the algorithm should iterate over the training data before it finalises the selection of centroids.

Section 24: Lab (Train a Simple Clustering Algorithm)

In this lab, we will be using the Weather Dataset that has weather data for 66 different airports in the USA from April to October 2013. We will **cluster the dataset into 5 distinct clusters** based on key weather metrics, such as visibility, temperature, dew point, wind speed etc. The goal is to **group airports with similar weather conditions**. You can learn more about the K-Means Clustering algorithm used in this lab by selecting this link (https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/k-means-clustering#configure-the-k-means-clustering-module)

From the results you can observe that each row (input) in the dataset is assigned to one of the 5 clusters: 0, 1, 2, 3, or 4. You can also see for each input, how far that input was from the various centroids. The cluster assignment is made based on the shortest distance between the input and cluster centroids. From the bar graph you can see the frequency distribution of all the inputs across the 5 clusters.

Assignments	DistancesToClusterCenter no.0	DistancesToClusterCenter no.1	DistancesToClusterCen no.2
1	1.043761	0.248115	0.485734
4	0.624923	0.928306	0.595271
2	0.67998	0.406787	0.138439
1	0.868134	0.31572	0.320039
2	0.828748	0.447536	0.17172