**Data acquisition and understanding stage of the Team Data Science Process**

**Goals**

* Produce a clean, high-quality data set whose relationship to the target variables is understood. Locate the data set in the appropriate analytics environment so you are ready to model.
* Develop a solution architecture of the data pipeline that refreshes and scores the data regularly.

**How to do it**

There are three main tasks addressed in this stage:

* **Ingest the data** into the target analytic environment.
* **Explore the data** to determine if the data quality is adequate to answer the question.
* **Set up a data pipeline** to score new or regularly refreshed data.

**Ingest the data**

Set up the process to move the data from the source locations to the target locations where you run analytics operations, like training and predictions. For technical details and options on how to move the data with various Azure data services, see [Load data into storage environments for analytics](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/ingest-data).

**Explore the data**

Before you train your models, you need to develop a sound understanding of the data. Real-world data sets are often noisy, are missing values, or have a host of other discrepancies. You can use data summarization and visualization to audit the quality of your data and provide the information you need to process the data before it's ready for modeling. This process is often iterative.

TDSP provides an automated utility, called [IDEAR](https://github.com/Azure/Azure-TDSP-Utilities/blob/master/DataScienceUtilities/DataReport-Utils), to help visualize the data and prepare data summary reports. We recommend that you start with IDEAR first to explore the data to help develop initial data understanding interactively with no coding. Then you can write custom code for data exploration and visualization. For guidance on cleaning the data, see [Tasks to prepare data for enhanced machine learning](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/prepare-data).

After you're satisfied with the quality of the cleansed data, the next step is to better understand the patterns that are inherent in the data. This data analysis helps you choose and develop an appropriate predictive model for your target. Look for evidence for how well connected the data is to the target. Then determine whether there is sufficient data to move forward with the next modeling steps. Again, this process is often iterative. You might need to find new data sources with more accurate or more relevant data to augment the data set initially identified in the previous stage.

**Set up a data pipeline**

In addition to the initial ingestion and cleaning of the data, you typically need to set up a process to score new data or refresh the data regularly as part of an ongoing learning process. Scoring may be completed with a data pipeline or workflow. The [Move data from a SQL Server instance to Azure SQL Database with Azure Data Factory](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/move-sql-azure-adf) article gives an example of how to set up a pipeline with [Azure Data Factory](https://azure.microsoft.com/services/data-factory/).

In this stage, you develop a solution architecture of the data pipeline. You develop the pipeline in parallel with the next stage of the data science project. Depending on your business needs and the constraints of your existing systems into which this solution is being integrated, the pipeline can be one of the following options:

* Batch-based
* Streaming or real time
* A hybrid

**Artifacts**

The following are the deliverables in this stage:

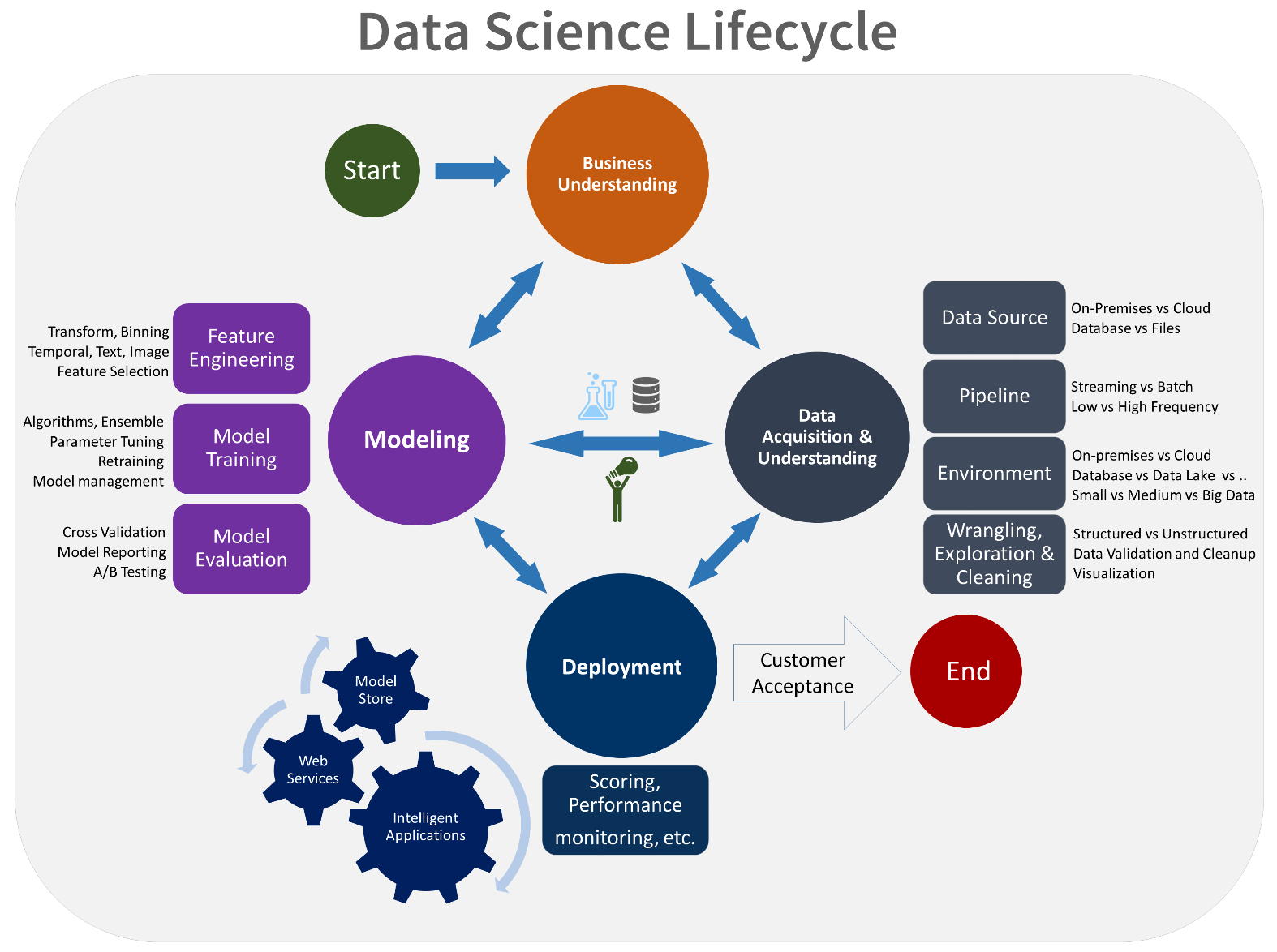
* [Data quality report](https://github.com/Azure/Azure-TDSP-ProjectTemplate/blob/master/Docs/Data_Report/DataSummaryReport.md): This report includes data summaries, the relationships between each attribute and target, variable ranking, and more. The [IDEAR](https://github.com/Azure/Azure-TDSP-Utilities/blob/master/DataScienceUtilities/DataReport-Utils) tool provided as part of TDSP can quickly generate this report on any tabular data set, such as a CSV file or a relational table.
* **Solution architecture**: The solution architecture can be a diagram or description of your data pipeline that you use to run scoring or predictions on new data after you have built a model. It also contains the pipeline to retrain your model based on new data. Store the document in the [Project](https://github.com/Azure/Azure-TDSP-ProjectTemplate/tree/master/Docs/Project) directory when you use the TDSP directory structure template.
* **Checkpoint decision**: Before you begin full-feature engineering and model building, you can reevaluate the project to determine whether the value expected is sufficient to continue pursuing it. You might, for example, be ready to proceed, need to collect more data, or abandon the project as the data does not exist to answer the question.

**Modeling stage of the Team Data Science Process lifecycle**

This article outlines the goals, tasks, and deliverables associated with the modeling stage of the Team Data Science Process (TDSP). This process provides a recommended lifecycle that you can use to structure your data-science projects. The lifecycle outlines the major stages that projects typically execute, often iteratively:

1. **Business understanding**
2. **Data acquisition and understanding**
3. **Modeling**
4. **Deployment**
5. **Customer acceptance**

Here is a visual representation of the TDSP lifecycle:



**Goals**

* Determine the optimal data features for the machine-learning model.
* Create an informative machine-learning model that predicts the target most accurately.
* Create a machine-learning model that's suitable for production.

**How to do it**

There are three main tasks addressed in this stage:

* **Feature engineering**: Create data features from the raw data to facilitate model training.
* **Model training**: Find the model that answers the question most accurately by comparing their success metrics.
* Determine if your model is **suitable for production.**

**Feature engineering**

Feature engineering involves the inclusion, aggregation, and transformation of raw variables to create the features used in the analysis. If you want insight into what is driving a model, then you need to understand how the features relate to each other and how the machine-learning algorithms are to use those features.

This step requires a creative combination of domain expertise and the insights obtained from the data exploration step. Feature engineering is a balancing act of finding and including informative variables, but at the same time trying to avoid too many unrelated variables. Informative variables improve your result; unrelated variables introduce unnecessary noise into the model. You also need to generate these features for any new data obtained during scoring. As a result, the generation of these features can only depend on data that's available at the time of scoring.

For technical guidance on feature engineering when make use of various Azure data technologies, see [Feature engineering in the data science process](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/create-features).

**Model training**

Depending on the type of question that you're trying to answer, there are many modeling algorithms available. For guidance on choosing the algorithms, see [How to choose algorithms for Microsoft Azure Machine Learning](https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-choice). Although this article uses Azure Machine Learning, the guidance it provides is useful for any machine-learning projects.

The process for model training includes the following steps:

* **Split the input data** randomly for modeling into a training data set and a test data set.
* **Build the models** by using the training data set.
* **Evaluate** the training and the test data set. Use a series of competing machine-learning algorithms along with the various associated tuning parameters (known as a *parameter sweep*) that are geared toward answering the question of interest with the current data.
* **Determine the “best” solution** to answer the question by comparing the success metrics between alternative methods.

**Note**

**Avoid leakage**: You can cause data leakage if you include data from outside the training data set that allows a model or machine-learning algorithm to make unrealistically good predictions. Leakage is a common reason why data scientists get nervous when they get predictive results that seem too good to be true. These dependencies can be hard to detect. To avoid leakage often requires iterating between building an analysis data set, creating a model, and evaluating the accuracy of the results.

We provide an [automated modeling and reporting tool](https://github.com/Azure/Azure-TDSP-Utilities/blob/master/DataScienceUtilities/Modeling) with TDSP that's able to run through multiple algorithms and parameter sweeps to produce a baseline model. It also produces a baseline modeling report that summarizes the performance of each model and parameter combination including variable importance. This process is also iterative as it can drive further feature engineering.

**Artifacts**

The artifacts produced in this stage include:

* [Feature sets](https://github.com/Azure/Azure-TDSP-ProjectTemplate/blob/master/Docs/Data_Report/Data%20Defintion.md): The features developed for the modeling are described in the **Feature sets** section of the **Data definition** report. It contains pointers to the code to generate the features and a description of how the feature was generated.
* [Model report](https://github.com/Azure/Azure-TDSP-ProjectTemplate/blob/master/Docs/Model/Model%201/Model%20Report.md): For each model that's tried, a standard, template-based report that provides details on each experiment is produced.
* **Checkpoint decision**: Evaluate whether the model performs sufficiently for production. Some key questions to ask are:
  + Does the model answer the question with sufficient confidence given the test data?
  + Should you try any alternative approaches? Should you collect additional data, do more feature engineering, or experiment with other algorithms?

**Next steps**

Here are links to each step in the lifecycle of the TDSP:

1. [Business understanding](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle-business-understanding)
2. [Data acquisition and understanding](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle-data)
3. [Modeling](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle-modeling)
4. [Deployment](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle-deployment)
5. [Customer acceptance](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle-acceptance)

We provide full end-to-end walkthroughs that demonstrate all the steps in the process for specific scenarios. The [Example walkthroughs](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/walkthroughs) article provides a list of the scenarios with links and thumbnail descriptions. The walkthroughs illustrate how to combine cloud, on-premises tools, and services into a workflow or pipeline to create an intelligent application.

For examples of how to execute steps in TDSPs that use Azure Machine Learning Studio, see [Use the TDSP with Azure Machine Learning](https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/).

**Feature engineering in data science**

In this article, you learn about feature engineering and its role in enhancing data in machine learning. Learn from illustrative examples drawn from [Azure Machine Learning Studio (classic)](https://docs.microsoft.com/en-us/azure/machine-learning/studio/what-is-ml-studio) experiments.

* **Feature engineering**: The process of creating new features from raw data to increase the predictive power of the learning algorithm. Engineered features should capture additional information that is not easily apparent in the original feature set.
* **Feature selection**: The process of selecting the key subset of features to reduce the dimensionality of the training problem.

Normally **feature engineering** is applied first to generate additional features, and then **feature selection** is done to eliminate irrelevant, redundant, or highly correlated features.

**What is feature engineering?**

Training data consists of a matrix composed of rows and columns. Each row in the matrix is an observation or record. The columns of each row are the features that describe each record. The features specified in the experimental design should characterize the patterns in the data.

Although many of the raw data fields can be used directly to train a model, it's often necessary to create additional (engineered) features for an enhanced training dataset.

Engineered features that enhance training provide information that better differentiates the patterns in the data. However, this process is something of an art. Sound and productive decisions often require domain expertise.

**Example 1: Add temporal features for a regression model**

Let's use the experiment [Demand forecasting of bikes rentals](https://gallery.azure.ai/Experiment/Regression-Demand-estimation-4) in Azure Machine Learning Studio (classic) to demonstrate how to engineer features for a regression task. The objective of this experiment is to predict the demand for bike rentals within a specific month/day/hour.

**Bike rental dataset**

The [Bike Rental UCI dataset](http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset/) is based on real data from a bike share company based in the United States. It represents the number of bike rentals within a specific hour of a day for the years 2011 and 2012. It contains 17,379 rows and 17 columns.

The raw feature set contains weather conditions (temperature/humidity/wind speed) and the type of the day (holiday/weekday). The field to predict is the count, which represents the bike rentals within a specific hour. Count ranges from 1 to 977.

**Create a feature engineering experiment**

With the goal of constructing effective features in the training data, four regression models are built using the same algorithm but with four different training datasets. The four datasets represent the same raw input data, but with an increasing number of features set. These features are grouped into four categories:

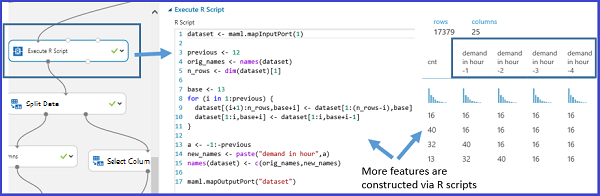
1. A = weather + holiday + weekday + weekend features for the predicted day
2. B = number of bikes that were rented in each of the previous 12 hours
3. C = number of bikes that were rented in each of the previous 12 days at the same hour
4. D = number of bikes that were rented in each of the previous 12 weeks at the same hour and the same day

Besides feature set A, which already exists in the original raw data, the other three sets of features are created through the feature engineering process. Feature set B captures recent demand for the bikes. Feature set C captures the demand for bikes at a particular hour. Feature set D captures demand for bikes at particular hour and particular day of the week. The four training datasets each includes feature set A, A+B, A+B+C, and A+B+C+D, respectively.

**Feature engineering using Studio (classic)**

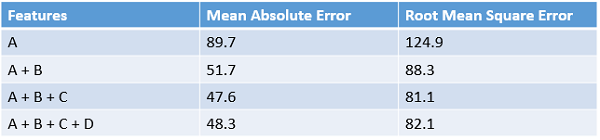
In the Studio (classic) experiment, these four training datasets are formed via four branches from the pre-processed input dataset. Except for the leftmost branch, each of these branches contains an [Execute R Script](https://msdn.microsoft.com/library/azure/30806023-392b-42e0-94d6-6b775a6e0fd5/) module, in which the derived features (feature set B, C, and D) are constructed and appended to the imported dataset.

The following figure demonstrates the R script used to create feature set B in the second left branch.



**Results**

A comparison of the performance results of the four models is summarized in the following table:



The best results are shown by features A+B+C. The error rate decreases when additional feature set are included in the training data. It verifies the presumption that the feature set B, C provide additional relevant information for the regression task. But adding the D feature does not seem to provide any additional reduction in the error rate.

**Example 2: Create features for text mining**

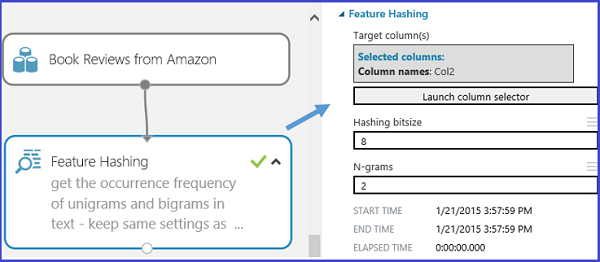
Feature engineering is widely applied in tasks related to text mining such as document classification and sentiment analysis. Since individual pieces of raw text usually serve as the input data, the feature engineering process is needed to create the features involving word/phrase frequencies.

**Feature hashing**

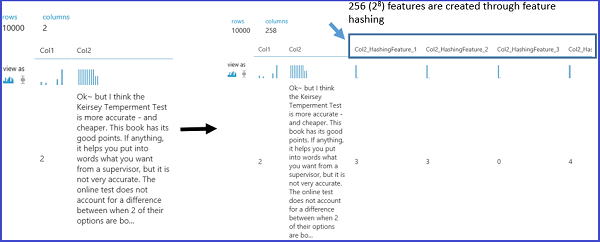
To achieve this task, a technique called [feature hashing](https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/feature-hashing) is applied to efficiently turn arbitrary text features into indices. Instead of associating each text feature (words/phrases) to a particular index, this method applies a hash function to the features and using their hash values as indices directly.

In Studio (classic), there is a [Feature Hashing](https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/feature-hashing) module that creates word/phrase features conveniently. Following figure shows an example of using this module. The input dataset contains two columns: the book rating ranging from 1 to 5, and the actual review content. The goal of this module is to retrieve a bunch of new features that show the occurrence frequency of the corresponding word(s)/phrase(s) within the particular book review. To use this module, complete the following steps:

* First, select the column that contains the input text ("Col2" in this example).
* Second, set the "Hashing bitsize" to 8, which means 2^8=256 features will be created. The word/phase in all the text will be hashed to 256 indices. The parameter "Hashing bitsize" ranges from 1 to 31. The word(s)/phrase(s) are less likely to be hashed into the same index if setting it to be a larger number.
* Third, set the parameter "N-grams" to 2. This value gets the occurrence frequency of unigrams (a feature for every single word) and bigrams (a feature for every pair of adjacent words) from the input text. The parameter "N-grams" ranges from 0 to 10, which indicates the maximum number of sequential words to be included in a feature.



The following figure shows what these new feature look like.



**Conclusion**

Engineered and selected features increase the efficiency of the training process, which attempts to extract the key information contained in the data. They also improve the power of these models to classify the input data accurately and to predict outcomes of interest more robustly.

Feature engineering and selection can also combine to make the learning more computationally tractable. It does so by enhancing and then reducing the number of features needed to calibrate or train a model. Mathematically, the selected features are a minimal set of independent variables that explain the patterns in the data and predict outcomes successfully.

It's not always necessarily to perform feature engineering or feature selection. It depends on the data, the algorithm selected, and the objective of the experiment.