Microsoft

Microsoft Cloud Workshop

Machine Learning

Step 1: Review the customer case study

Outcome

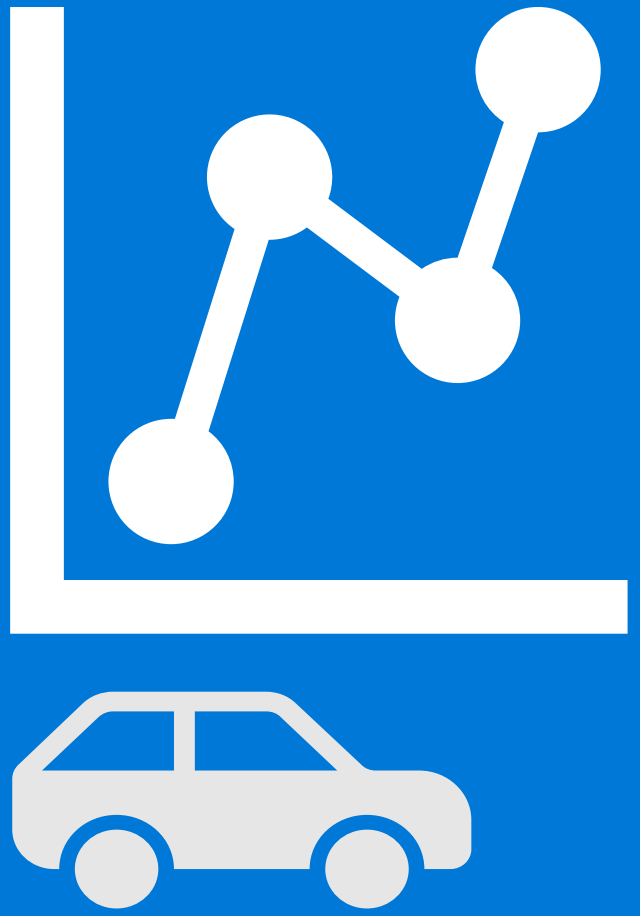
Analyze your customer needs.

Timeframe

15 minutes

Customer situation

- Trey Research Inc. delivers innovative solutions for manufacturers.
- They specialize in identifying and solving problems for manufacturers that can run the range from automating away mundane but time-intensive processes to delivering cutting edge approaches that provide new opportunities for their manufacturing clients.
- Trey Research is looking to provide the next generation experience for connected car manufacturers by enabling them to utilize AI to decide when to pro-actively reach out to the customer thru alerts delivered directly to the car's in-dash information and entertainment head unit.
- For their PoC, they would like to focus on two maintenance related scenarios.



Customer situation – requirements

- Trey Research recently instituted new regulations defining what parts are compliant or out of compliance. Rather than rely on their technicians to assess compliance, they would like to automatically assess the compliance based on component notes already entered by authorized technicians.
- Trey Research would like to predict the likelihood of battery failure based on the time series-based telemetry data that the car provides. The data contains details about how the battery performs when the vehicle is started, how it is charging while running, and how well it is holding its charge, among other factors. If they detect a battery failure is imminent within the next 30 days, they would like to send an alert. There are, however, concerns about the quality of the battery telemetry data, so Trey Research would like to be sure that, before being fed into the Machine Learning process, data is properly cleansed and prepared.

Customer situation – requirements (continued)

- Trey Research wants to understand how they might use machine learning or deep learning in both scenarios and standardize the platform that would support the data processing, model management and inferencing aspects of each.
- They are also interested to learn what new capabilities Azure provides that might help them to integrate with their existing investments in MLflow for managing machine learning experiments. Furthermore, they would also like to understand how Azure might help them to document and explain the models that are created to non-data scientists or might accelerate their time to creating production ready, performant models.
- They would like to be able to easily create dashboards that summarize the alerts generated so they can observe the solution in operation.

Customer needs

- We would like to minimize the number of distinct technologies we need to apply across the data science process, from data collection, exploratory data analysis, data preparation, model training, model management and model deployment. We already have existing investments in MLflow. How can Azure help us in this?

- Our board is concerned about the liability of not being able to justify and explain the function of the models we create. We recognize many models are not easily explained in layman's terms, but how can we go about documenting how our models make predictions generally or how why they made specific predictions for particular input?

Customer needs - continued

- Understand how they should create the NLP based model for compliance and the battery life-span forecasting model.

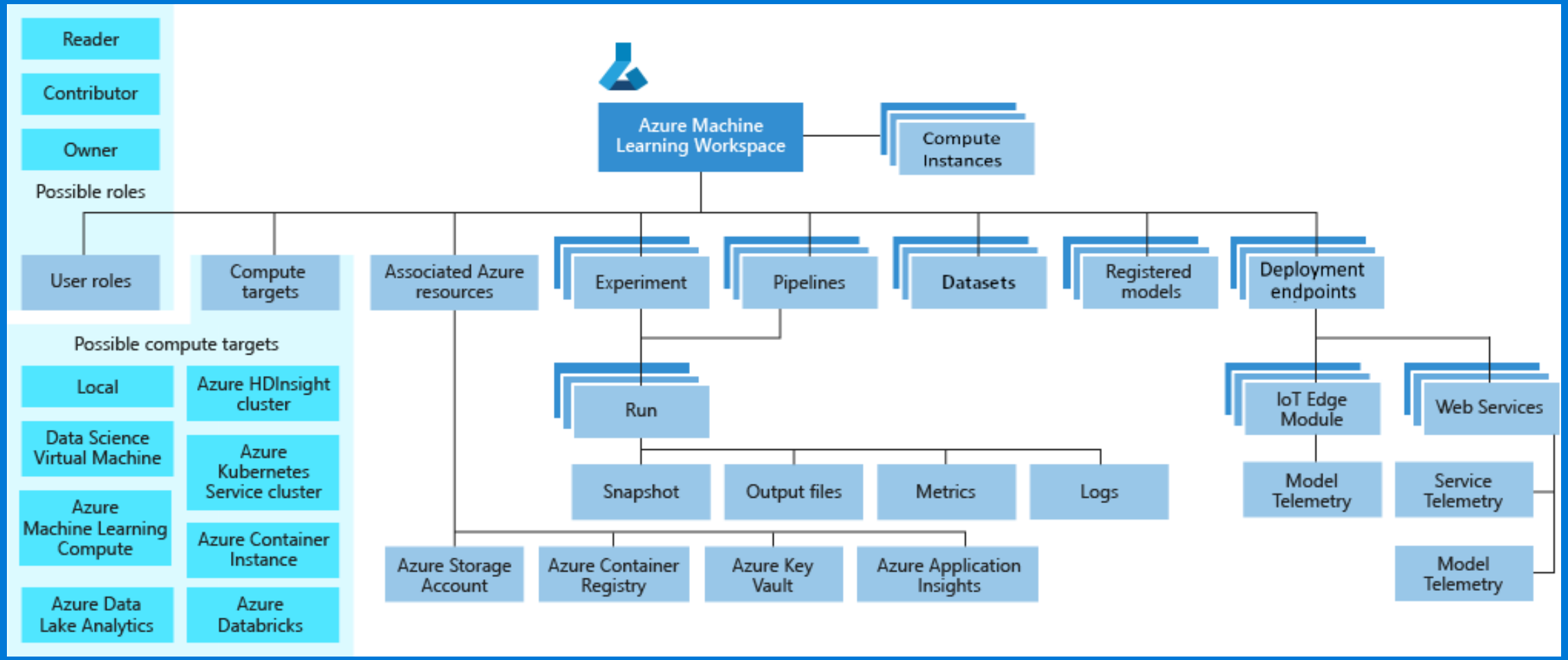
- Know the data pipeline they need to build in Azure, from ingesting telemetry, to storing both the compliance text and battery telemetry, to visualizing the result.

Customer objections

- Should we use machine learning or deep learning approaches?
- How should we choose between Keras and PyTorch for performing deep learning?
- We have heard Azure Machine Learning supports automated machine learning; can we use automated machine learning to create models using deep learning? Can we really expect a non-data scientist to create performant models using these tools?
- Some of our team has worked with Azure Databricks, and they are confused by the overlap with Azure Machine Learning. How should we be thinking about when to use which?
- We have heard a lot about how complicated and opaque trained deep learning models are. How is it even possible to attempt to explain them?

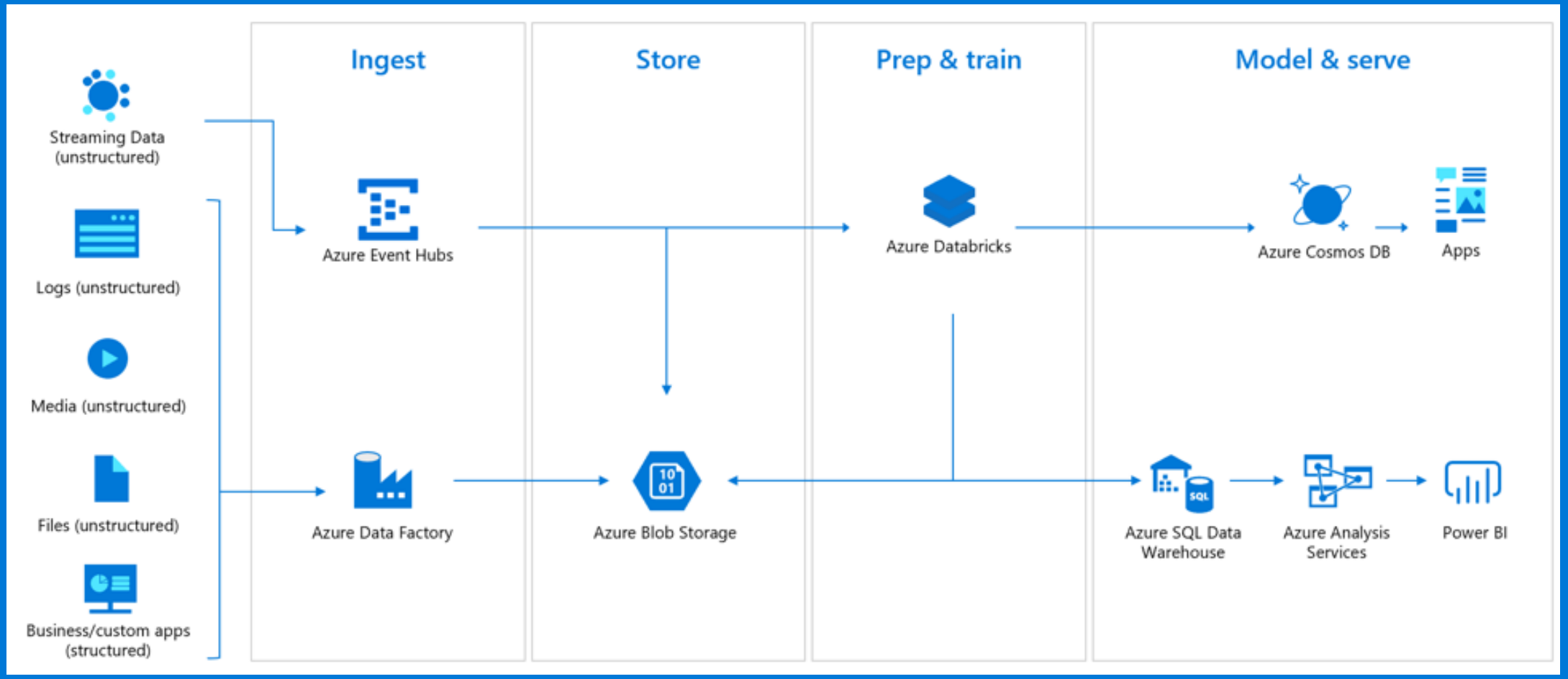
Common scenarios

- Azure Machine Learning workspace taxonomy



Common scenarios - continued

Real-time analytics



Step 2: Design the solution

Outcome

Design a solution and prepare to present the solution to the target customer audience in a 15-minute chalk-talk format.

Timeframe

60 minutes

Business needs (10 minutes)	• Respond to questions outlined in your guide and list the answers on a flipchart
Design (35 minutes)	• Design a solution for as many of the stated requirements as time allows. Show the solution on a flipchart
Prepare (15 minutes)	• Identify any customer needs that are not addressed with the proposed solution • Identify the benefits of your solution • Determine how you will respond to the customer's objections • Prepare for a 15-minute presentation to the customer

Step 3: Present the solution

Outcome

Present a solution to the target customer in a 15-minute chalk-talk format.

Timeframe

30 minutes (15 minutes for each team to present and receive feedback)

Directions

- Pair with another table.
- One table is the Microsoft team and the other table is the customer.
- The Microsoft team presents their proposed solution to the customer.
- The customer asks one of the objections from the list of objections in the case study.
- The Microsoft team responds to the objection.
- The customer team gives feedback to the Microsoft team.

Wrap-up

Outcome

Identify the preferred solution for the case study.
Identify solutions designed by other teams.

Timeframe

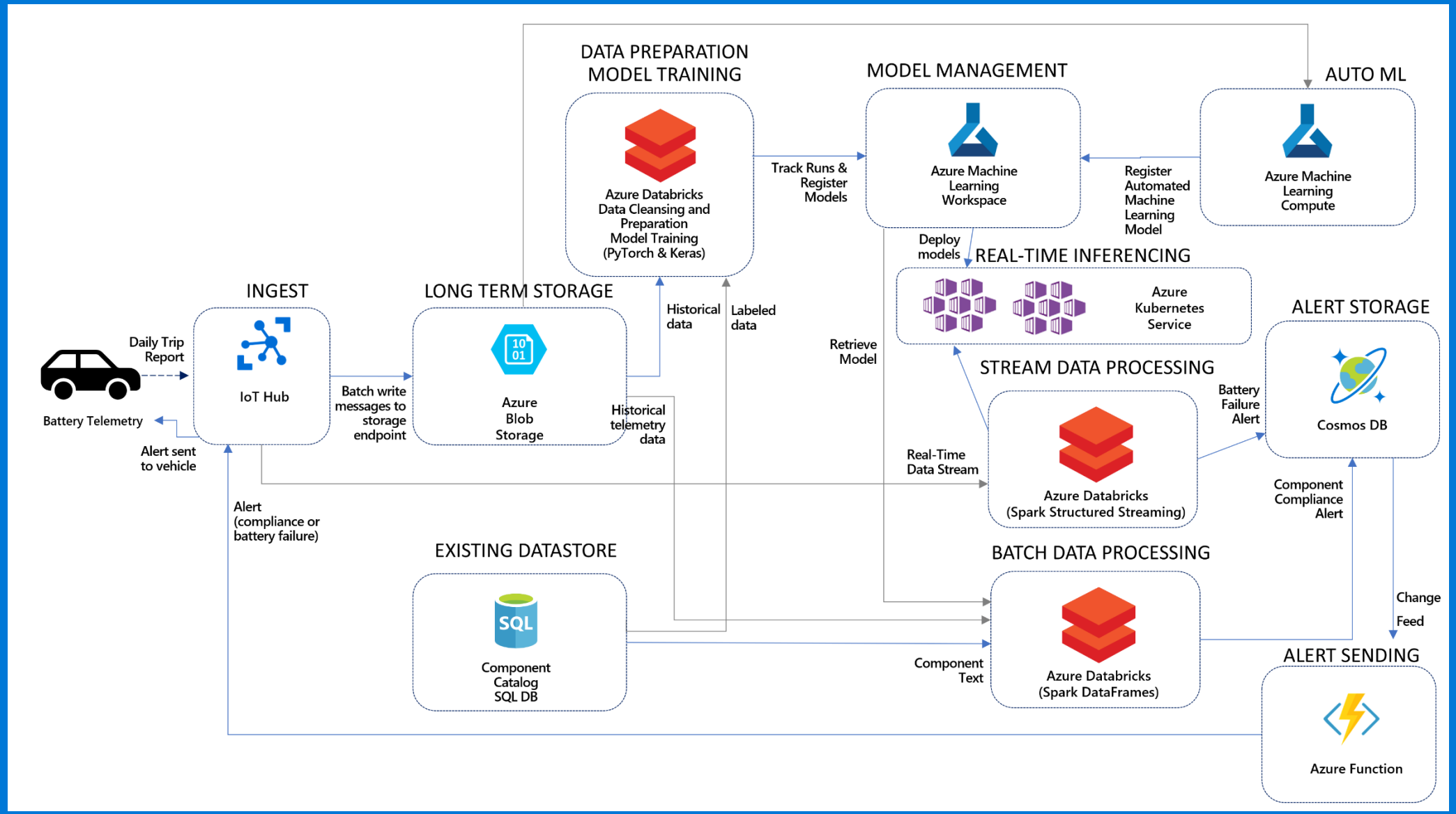
15 minutes

Preferred target audience

Francine Fischer, CIO of Trey Research

The primary audience is the business decision makers and technology decision makers. From the case study scenario, this would include the Director of Analytics. Usually we talk to the infrastructure managers, who report to the chief information officers (CIOs), or to application sponsors (like a vice president [VP] line of business [LOB], or chief marketing officer [CMO]), or to those that represent the business unit IT or developers that report to application sponsors.

Preferred solution



Preferred solution – classify component descriptions text data

What is the general pipeline for approaching the training of text analytic models such as this? What are the general steps you need to take to prepare the text data for performing tasks like classification?

- The core task in natural language processing (NLP) text pipelines is data preparation to express the textual data as numeric vectors by using word embeddings.
- The general pipeline begins by pre-processing or normalizing the text. This step typically includes tasks such as breaking the text into sentence and word tokens, standardizing the spelling of words, and removing overly common words (called stop words).
- The output of this phase is typically a multi-dimensional array consisting of an array of documents, each having an array of sentences, with each sentence having its own array of words. The next step is feature extraction, which creates a numeric representation of the textual documents.
- During feature extraction, a "vocabulary" of unique words is identified, and each word becomes a column in the output. Each row represents a document. The value in each cell is typically a measure of the relative importance of that word in the document, where if a word from the vocabulary does not appear, then that cell has a zero value in that column. This approach enables machine learning algorithms, which operate against arrays of numbers, to also operate against text.
- Deep learning algorithms operate on tensors, which are also vectors (or arrays of numbers), so this approach is also valid for preparing text for use with a deep learning algorithm.

Preferred solution – classify component descriptions text data (continued)

What data would they need to train the model?

Trey would need labeled examples consisting of the component text description and the label (0 for non-compliant, 1 for compliant).

How would Trey create a deep learning model to classify the component descriptions? Give an example of the architecture of the neural network.

Trey would author a notebook that loads the labeled component description data and preprocesses the data to vectorize the text corpus. Next, it builds a fully connected deep learning network. The first layer in the network is the embedding layer that uses a pretrained word embeddings, such as the GloVe: Global Vectors for Word Representation, to convert the vectorized text corpus to its corresponding word vectors. The embedding layer feeds into a Long Short-Term Memory (LSTM) recurrent neural network and the output from the LSTM is passed to a final layer consisting of a single neuron with a Sigmoid activation function. The output of the final layer will predict if the component description is compliant or not compliant.

Describe how Trey would use the model in the context of batch scoring the component text for compliance? What services and frameworks would you suggest?

Because the combination of components and vehicles yields a potentially large number of data points they will want to score, Trey should consider using Azure Databricks and Spark to perform the scaling since it would support scoring the large dataset.

Preferred solution – forecasting battery failure

At a high level, describe the steps to apply a forecasting model to predict battery failure pending within the next 30 days.

In a forecasting scenario, time-stamped data is provided, and the goal is to be able to forecast a value a certain number of periods in the future by looking at historical data. In Trey's case, they could use their existing daily battery telemetry to train a model that forecast the battery cycles used in any given day, for 30 days out. They would sum the battery cycles used from the historical data plus the forecast data and if the result exceeds the expected battery life they could predict this as a pending failure.

Describe how Trey would approach the data cleansing and preparation process

First, data needs to be analyzed using a combination of numerical and visual methods (e.g., display the time series and attempt to identify any anomalies visually). Next, we validate the preliminary hypothesis related to data issues using proper metrics. Once we complete the identification and validation of the problems, we eliminate each one of them until the data is clean.

Preferred solution – forecasting battery failure (continued)

With regards to the model, what options does Trey have for creating the model against the time series data? Should they create a regression model, a forecasting model or a classifier? Why?

When forecasting against time-series data Trey should pick a forecasting model so as to take advantage of the temporal signals in the data. While a regression could work in this scenario, it is not common practice to use a regression when time series data is available.

Can this model be built using machine learning or does it require deep learning?

This model can be built using forecasting techniques from either machine learning or deep learning.

Preferred solution – forecasting battery failure (continued)

If you were to suggest a deep learning model for forecasting against the time-series data, what architecture of neural network would you consider using first?

Generally, when forecasting against time-series data the predictions desired should be informed by all the historical data but place more emphasis on more recent data. Recurrent Neural Networks (RNN's) provide such capability, and that is why they are frequently the first neural network architecture attempted with time-series data.

Describe how Trey would use the model in the context of scoring the streaming telemetry? What services and frameworks would you suggest?

Trey should use the model within notebooks running in Azure Databricks, and apply the model using Spark Streaming to forecast results on micro-batches of battery telemetry as they arrive.

Preferred solution – automated machine learning

For which scenario could Trey apply automated machine learning? Why?

While automated machine learning could be applied to both scenarios, the significant additional data preparation required for text classification means that automated machine learning could not be applied outright. However, for the time-series battery lifecycle data they could directly use the automated machine learning capabilities of Azure Machine Learning to create a forecast model.

How could they use automated machine learning with Azure Databricks?

Trey has two options for leveraging automated machine learning:

- They could train the model with the user experience that is available from the Azure Machine Learning workspace in the Azure Machine Learning studio, and then register the trained model with the workspace.
- They could train the model using automated machine learning via the Azure Machine Learning Python SDK within a Databricks notebooks, and then register the model that results.

With the registered model available, any notebooks they would build that use the model for scoring would retrieve the model from the Azure Machine Learning registry using the Azure Machine Learning Python SDK.

Preferred solution – model management

For both models, how should Trey track the performance of each of their training runs?

There are two approaches to manage the life cycle of the machine learning experiments. In the first approach, within their notebooks that perform model training, Trey should be using the Azure Machine Learning Python SDK to create an experiment for session. Within this experiment they can create a run each time they train the model and capture logs, the training time taken and any performance metrics that result from the model evaluation. In the second approach, they can continue to leverage their existing investments in MLflow and simply connect the MLflow experiments with Azure Machine Learning. By doing so it will enable Trey to automatically track and log experiment metrics in artifacts in the Azure Machine Learning workspace. Thus, Trey can also take advantage of the Azure Machine Learning workspace that provides a centralized, secure, and scalable location to store training metrics and models.

Can they quickly view this experiment data somewhere? Are there programmatic options to querying this data?

Once the data is for an experiment is collected in the Azure Machine Learning workspace, Trey can view it from the Experiments tab of the Azure Machine Learning studio. From here they can drill into each run and explore the run details. Using the Azure Machine Learning Python SDK from a notebook, Trey can retrieve all of the experiments and run details and evaluate them programmatically (for example to pick the best run according to some custom logic).

Preferred solution – model management (continued)

How would they manage versioning of each the models they have created and associate these models with the results of evaluating their performance?

When they complete a training run for a model they would like to keep, Trey can use the Azure Machine Learning Python SDK to register the model in its registry. Each time they register a model with the same name, Azure Machine Learning will automatically version the model and add the new model to the version history. They can register models using the run, and by doing so will automatically associate the model with run and the performance metrics it contains.

In your solution, how will Trey retrieve previous versions of models and use them for further evaluation or scoring?

They can retrieve previous versions of a model by going to the Azure Machine Learning workspace, Models tab in the Azure Machine Learning studio or by requesting the model from a notebook using the Azure Machine Learning Python SDK.

Preferred solution – model explainability and reproducibility

How would you suggest Trey programmatically create explanations of their models? What is the process? Explain how it works to improve the interpretability of the model generally as well as for the results of specific predictions.

The Azure Machine Learning SDK includes interpretability features that accept a trained model along with training or test data and can return insights on why the model is making its decisions by providing analysis like which features have the greatest impact on the model's predictions.

This approach can be performed globally (against the entire data set provided) or against a single sample from the data set. The former enables Trey to understand generally what is influencing the models predictions while the latter enables Trey to explain what influenced a particular prediction. The SDK also provides visualizations of the explanations that result that can be use within a notebook.

Is the approach you suggest limited to working against machine learning models only (e.g., it does not work against deep learning models)?

The approach taken under the covers by Azure Machine Learning Python SDK is effectively black box testing of a model- it takes an input sample, uses the model to make the prediction and evaluates the outcome using a variety of techniques (called explainers). As such it is agnostic to whether the model is machine learning based or deep learning based.

Preferred solution – model explainability and reproducibility (continued)

Does your approach support explaining models created with automated machine learning?

Yes, in fact the Azure Machine Learning Python SDK provides additional functionality specifically for enabling model interpretability on models created using Azure Machine Learning automated machine learning.

How do machine learning pipelines help improve the reproducibility of model training and scoring? How could your solution leverage pipelines?

Machine learning pipelines encapsulate the steps taken to from input training data, to trained model, as well as to go from input data to scored result. Pipelines package these steps into reusable objects. Azure Machine Learning and the SDK support the creation, registration and execution of pipelines. The use of pipelines does increase the reproducibility of being able to re-create a model or re-execute an inference in a way that helps to guarantee fidelity across all executions. Trey could author Azure Machine Learning pipelines in notebooks using the Azure Machine Learning Python SDK.

Preferred solution – enabling visualization

During model training and evaluation, what services and libraries would you recommend that Trey utilize for visualizing the data and performance results?

During modeling, Trey should utilize the visualization capabilities of Azure Databricks notebooks, as well as open source visualizations libraries like matplotlib to explore and understand the data and performance results. By doing so within the modeling notebook, they enable rapid iteration and understanding of their data and model without having to switch into a different environment for visualization.

Would the approach you suggest extend to visualizing the streaming battery data?

Trey could continue to use Azure Notebooks to visualize the results of the Spark Structured Streaming queries.

What should Trey utilize for providing easy to customize visualizations to business analysts and stakeholders? Is it the same or different tool you suggested they use during modeling?

While Trey could create simplified notebooks and even dashboards using Azure Databricks notebooks, its is more likely that business analysts would be more comfortable using Power BI, which is why it is important to ensure the scored results are made available in a service datastore compatible with Power BI like Cosmos DB.

Preferred objections handling (1, 2)

1. *Should we use machine learning or deep learning approaches?*

For Trey's two scenarios, they could actually use either approach. They would likely want to try both approaches and determine which yields the best performance in terms of model training time, inferencing time, and inferencing performance (e.g., accuracy).

2. *How should we choose between Keras and PyTorch for performing deep learning?*

This is the subject of much discussion in the community, however the guidance for selecting between Keras and PyTorch boils down to: Keras may be easier to start with and easier to build production grade models, while PyTorch has a steeper initial learning, as it is lower level than Keras, but it offers greater flexibility, faster inferencing and improved debuggability in the balance.

For a comprehensive comparison, see https://deepsense.ai/keras-or-pytorch/.

Preferred objections handling (3)

3. We have heard Azure Machine Learning supports automated machine learning, can we use automated machine learning to create models using deep learning? Can we really expect a non-data scientist to create performant models using these tools?

Automated machine learning in Azure Machine Learning helps to simplify and expedite the process of producing a performant model. It does this by trying many combinations of best practice data preparation (automated pre-processing and featurization), algorithm selection and algorithm parameters (hyper-parameter tuning) while asking the user only for some relatively simple configuration information (such as the type of prediction problem, the input training data set, the feature to predict and the compute resources on which to experiment) to perform the job.

Azure Machine Learning provides access to the automated machine learning capabilities via a Python SDK and via visual interface in the Azure Machine Learning studio. The latter user experience can simplify the setup enough such that a non-data scientist who has an understanding of the fundamentals of training a model can use it to create a model.

Preferred objections handling (4)

4. Some of our team has worked with Azure Databricks, and they are confused by the overlap with Azure Machine Learning. How should we be thinking about when to use which?

Consider using both. The best way to think about the relationship between Azure Databricks and Azure Machine Learning is that Azure Databricks provides the tools for data engineers and data scientists to author their data and machine learning pipelines as well as the compute that powers these, and Azure Machine Learning provides the platform that formalizes the modeling process by capturing data about training runs, versioning pipelines and models and assisting with the deployment of models as web services.

Preferred objections handling (5)

5. We have heard a lot about how complicated and opaque trained deep learning models are. How is it even possible to attempt to explain them?

The approach taken under the covers by Azure Machine Learning Python SDK is effectively black box testing of a model- it takes an input sample, uses the model to make the prediction and evaluates the outcome using a variety of techniques (called explainers). As such it is agnostic to whether the model is machine learning based or deep learning based. An example of such an explainer is the Mimic Explainer (also called a Global Surrogate). The Mimic Explainer is based on the idea of training global surrogate models (https://christophm.github.io/interpretable-ml-book/global.html) to mimic blackbox models. A global surrogate model is an intrinsically interpretable model that is trained to approximate the predictions of any black box model as accurately as possible. Data scientists can interpret the surrogate model to draw conclusions about the black box model. You can use one of the following interpretable models as your surrogate model: LightGBM (LGBMExplainableModel), Linear Regression (LinearExplainableModel), Stochastic Gradient Descent explainable model (SGDExplainableModel), and Decision Tree (DecisionTreeExplainableModel).

Using the classes and methods in the SDK, you can:

Explain model prediction by generating feature importance values for the entire model and/or individual datapoints.

Achieve model interpretability on real-world datasets at scale, during training and inference.

Use an interactive visualization dashboard to discover patterns in data and explanations at training time

The azureml-interpret SDK module uses the interpretability techniques developed in Interpret-Community, an open source python package for training interpretable models and helping to explain blackbox AI systems.

For a detailed introduction, see Model interpretability in Azure Machine Learning (https://docs.microsoft.com/en-us/azure/machine-learning/how-to-machine-learning-interpretability).

Customer quote

"We are excited by how Azure enables us with a comprehensive platform for performing machine learning and deep learning without complicating big data analytics."

- Francine Fischer, CIO of Trey Research

Microsoft