# Using Machine Learning to Improve Cost Data Normalization

Prepared for the 2023 ICEAA Workshop

Bryan Mariscal
SSC/AC FMCR

# Presentation Goals / Outline

## Presentation Goals

- Provide a concrete end-to-end example of using machine learning (ML) to improve an existing process (cost data normalization)
- Share…
  - Considerations
  - Development process & techniques
  - Lessons learned
  - Future initiatives

## Outline

- Background
  - Unmanned Space Vehicle Cost Model (USCM) & Normalization Process
  - Mapping Cost
  - ML Application to Normalization
- ML Considerations
- ML Model Development
- Summary & Way Forward

# Unmanned Space Vehicle Cost Model (USCM)

## USCM components

Robust **data collection** and **normalization** process

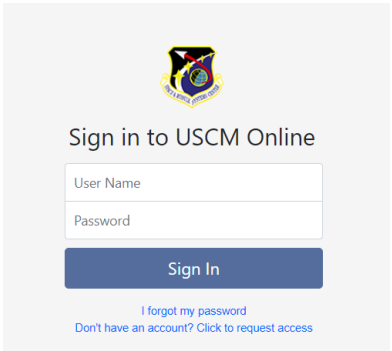
**Satellite Database**

**CERs**

Standard | Custom

*Independent Research*

**Access**




## About

The **Unmanned Space Vehicle Cost Model (USCM)** is a Space Systems Command (SSC) product used to enable the estimation of unmanned, earth-orbiting satellites. First published in November 1969, it has evolved significantly since then.

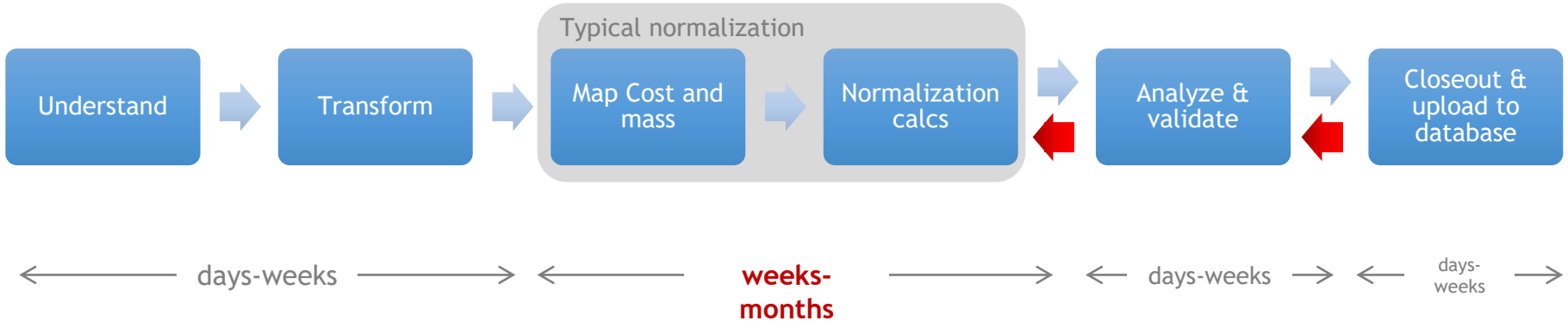
## Key Facts

- Contains over 120 normalized datasets
- Suite of products
- End-of-program costs
- Standard CERs developed at given points in time / Database updated quarterly

## Use Cases

- Source selection
- Establishing program budgets
- Crosschecks
- Education
- Glean insights regarding SSC satellite costs

# USCM Normalization Process




## Normalization is a time intensive effort

- Bottlenecks
  - Mapping cost and mass
- Challenges
  - (Potentially) lots of data to sift through
  - Analysts can (and often) encounter “gray” areas in data where multiple work breakdown structure (WBS) mappings are plausible

## Can we improve?

- Explore utilization of machine learning (ML) techniques
  - Improve consistency of mappings and decrease normalization turn-around times
  - Analyst validation still expected, but will give analysts a “running start” by providing a draft cost mapping
  - Effort focused on improving mapping cost process (for now)

# USCM Normalization – Mapping Cost

**Raw Cost Data**

Feature 1, Feature 2 … $ Charge — Record 1

⋮

Feature 1, Feature 2 … $ Charge — Record n

**Documentation and other metadata**

- Hardware design documentation
- Program WBS Dictionary
- …

Analyst Assessment

| USCM Standard WBS # | Descriptor |
|---|---|
| 1.1.1 – System of Systems Systems Engineering | Requirements Analysis |
| 1.2.1.1 – Space Vehicle Systems Engineering | Risk management |
| 1.2.2.2 – Space Vehicle Bus Structures and Mechanisms Subsystem | Systems Analysis |
| … | Primary Structures |
| 1.2.10 – Mission Operations | Mechanisms |
| | Tanks |
| | Engines |
| | … |
| | Training |

- Inputs:
  - Raw Cost data (matrix with m rows and n columns)
  - Documentation and other metadata
- Outputs:
  - Mapped Dollars (USCM WBS # and descriptor for each row in raw cost data matrix)

- Challenges:
  - (Possibly) lots of data to sift through
  - Analysts can (and often) encounter “gray” areas in data where multiple WBS mappings are plausible

# ML Application to Normalization

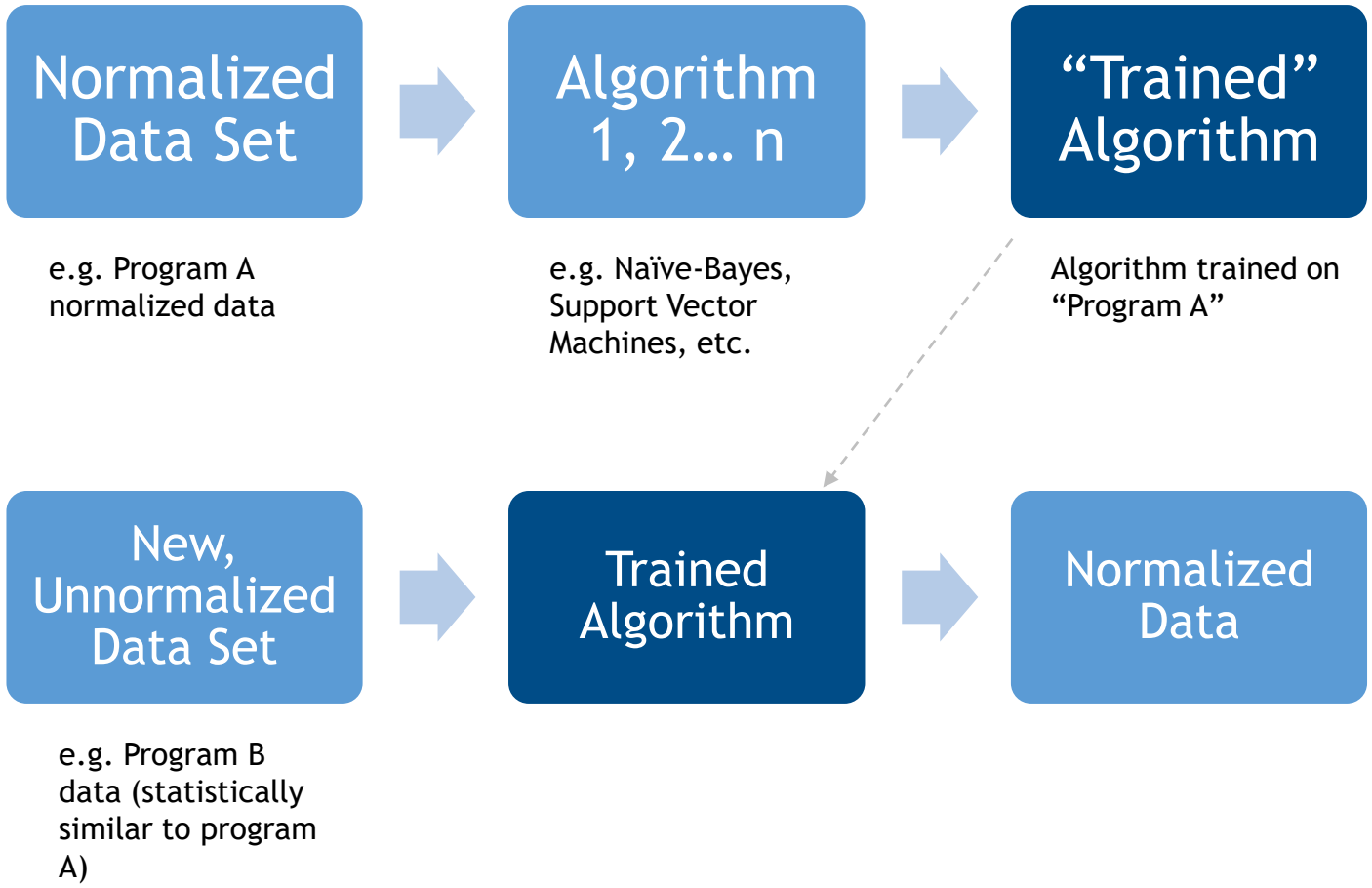
## USCM cost normalization is a text classification problem

(more generally, supervised ML)

Supervised ML

- Definition: Give data to an algorithm where we know the right answer, and have the algorithm learn how to predict future occurrences
  - Requires labeled data (input-output pairs)
- Model learns input to output mapping based on example input-output pairs (i.e. training)
  - Need to choose and/or engineer features that correlate with output
- Want future data to be statistically similar to training data for adequate performance
  - Model will suggest a mapping that is consistent with mapped training data

Example

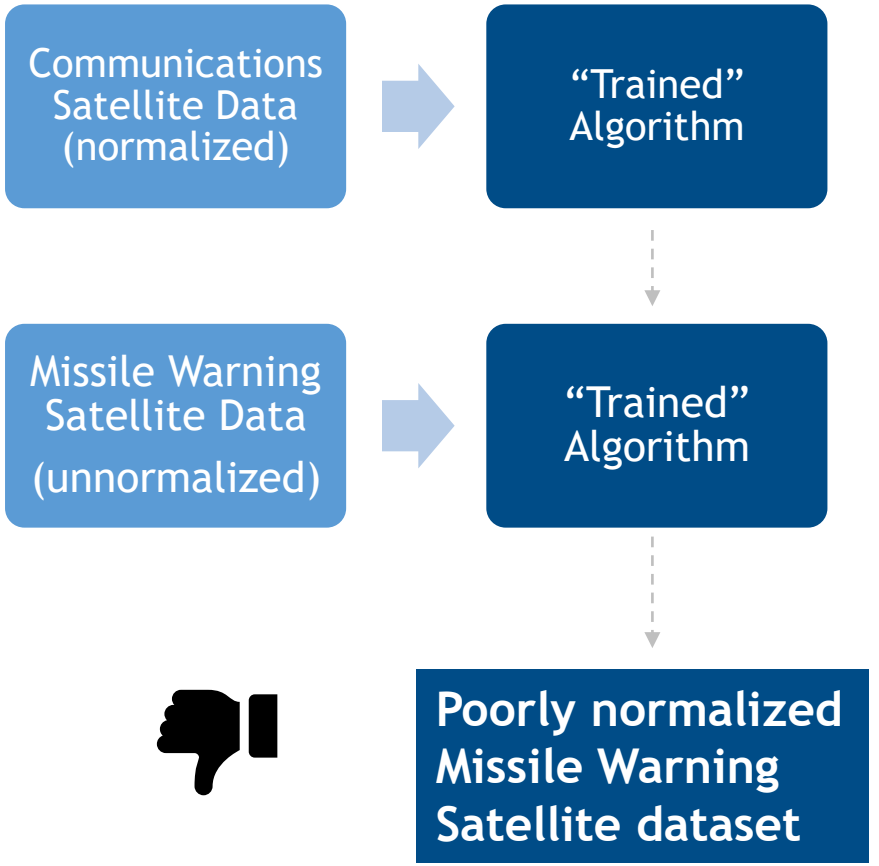


# ML Modeling Considerations

Training an algorithm on historical data could yield poor results in production if future data is significantly different from training data

- Training data should be representative of data you expect to receive in production
- Need to carefully choose which programs/datasets are utilized to train an algorithm
- Grabbing any past data to train your algorithm and expecting it to generalize well to your dataset of interest may lead to disappointment

**Hypothetical Scenario to Avoid**



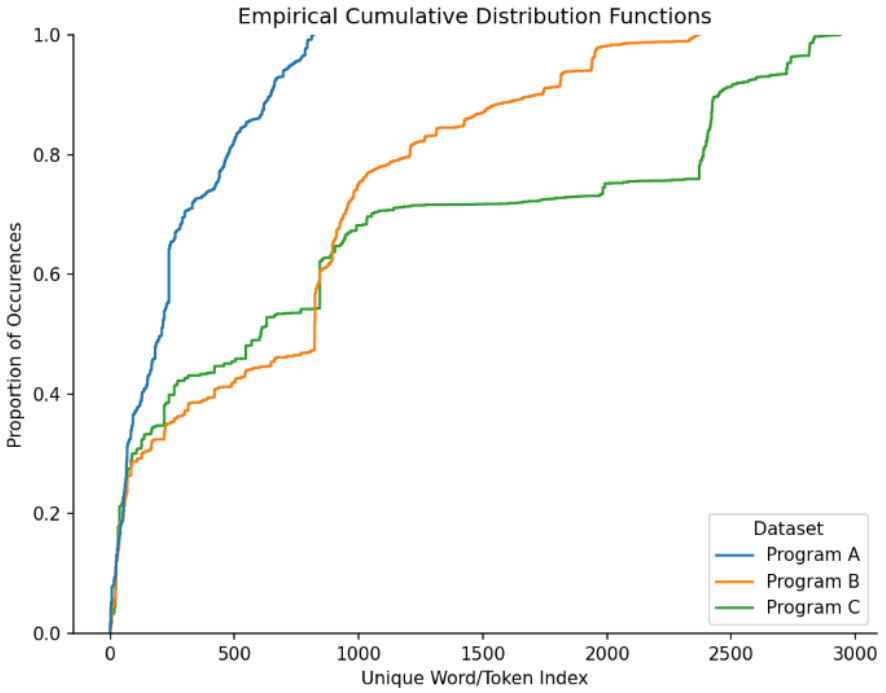
# ML Modeling Considerations (cont.)

USCM cost data structure & contents vary by…

- Contractor
- Program
- Over time (accounting systems and programs can evolve)

Cost data often contains repeated records (i.e. repeated descriptions but with different charge amounts)

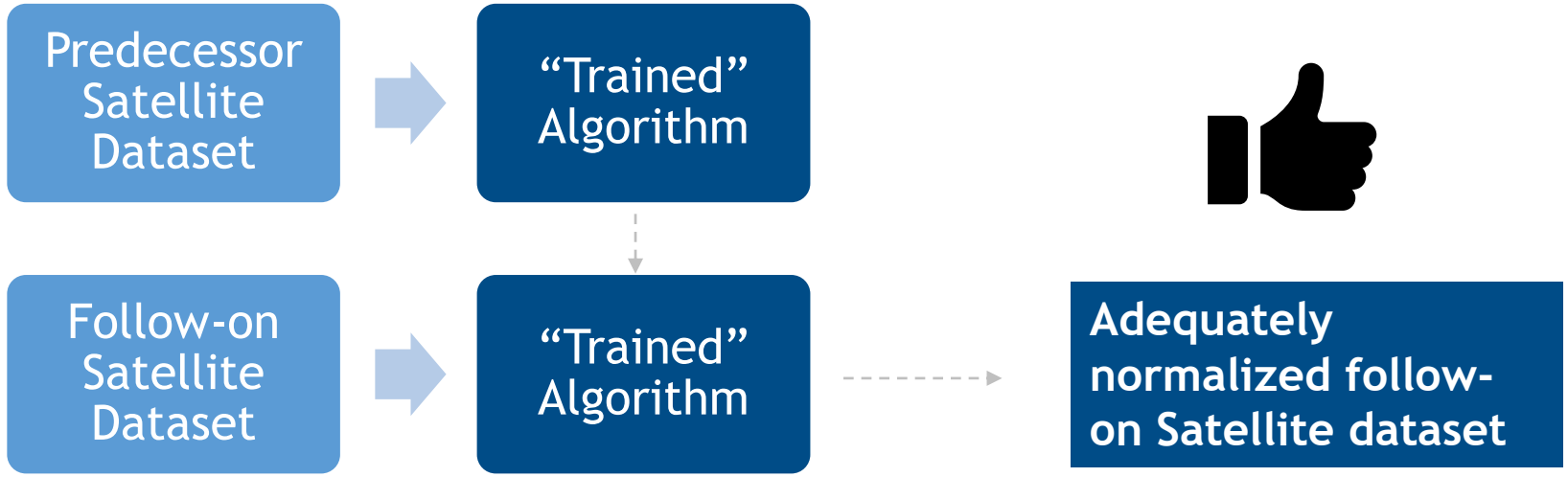
- Important consideration when setting up a model validation strategy




The vocabulary is more similar between Programs B and C relative to A and B or A and C (~75% overlap in words/tokens vs. ~45-55%).

# ML Modeling Considerations (cont.)

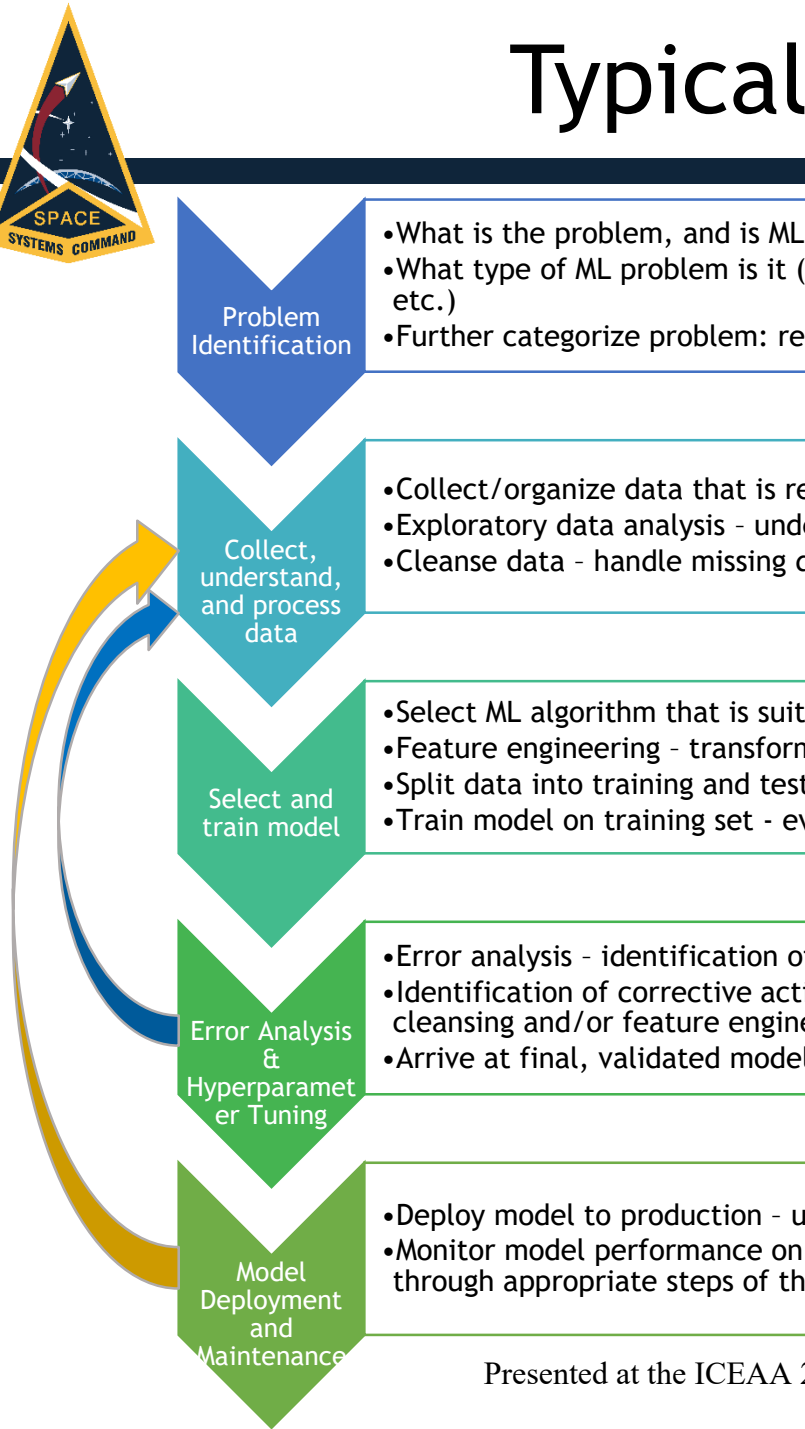
**Scenario with (perceived) Highest Probability of Success**




Based off the considerations, decided to pursue scenario with (perceived) highest probability of success: train on a predecessor program and apply to follow-on program

- Similar nomenclature in cost data
- Facilitates adequate generalization of trained model to production data

# Typical ML Model Development Workflow




Plan of attack: start simple and iterate. It is generally a good idea to start with simple models. This will allow for more rapid: 1) setup of pre/post-processing steps, 2) setup of validation strategy, and 3) identification of areas of deficiency.

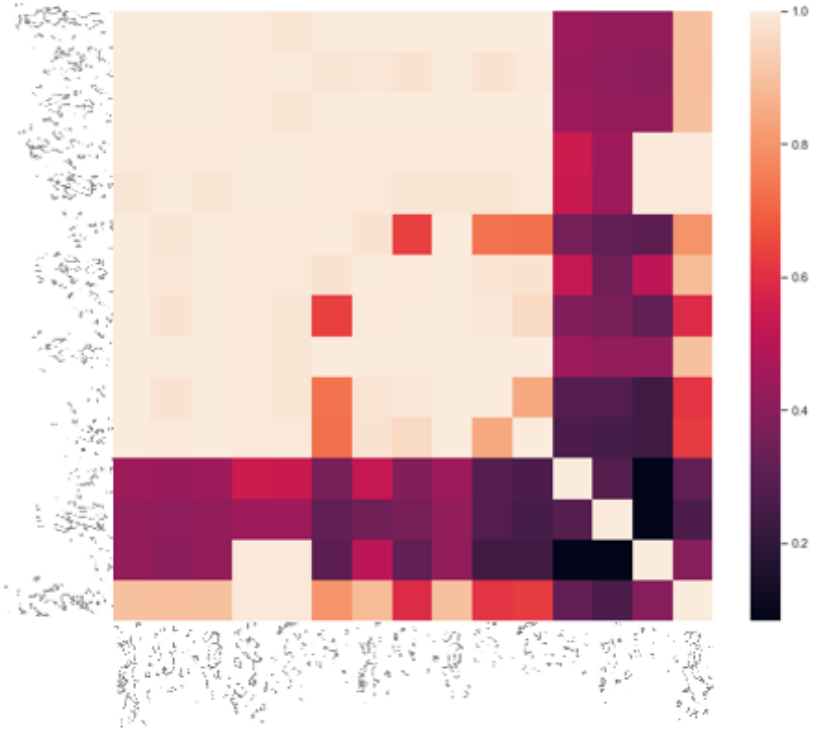
The following slides will demonstrate how this workflow & plan of attack was applied to develop the ML model.

# ML Model Development

Problem Identification

Collect, understand, and process data

- USCM cost normalization is a **supervised text classification** problem
- Gather data:
  - Normalized predecessor program
  - Unnormalized follow-on program
- Exploratory data analysis (EDA) performed on gathered data
  - Identification of features that will best serve as predictors for the WBS #
  - Confirm similarity of data between predecessor and follow-on data




Heatmap showing Cramer's V for features and WBS labels from normalized data. Cramer's V is a statistic from 0 to 1 that measure strength of association between two categorical variables (i.e. does knowing the value of one help us determine the other). 1 = perfect association, 0 = no association. Can use this as an indicator of correlation.

https://www.spss-tutorials.com/cramers-v-what-and-why/

# ML Model Development (cont.)

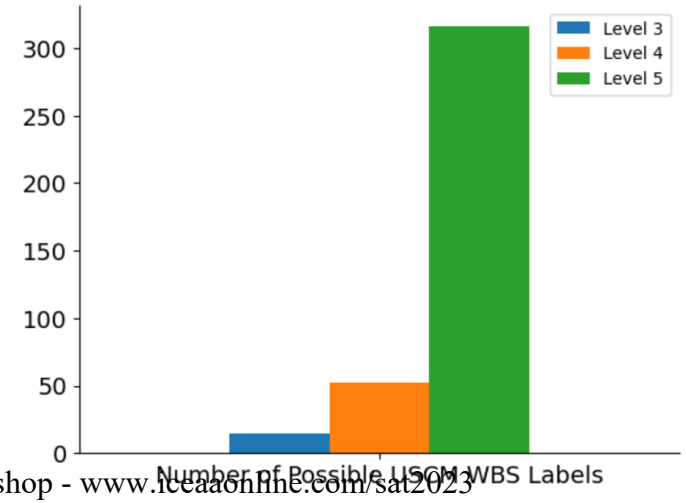
Collect, understand, and process data

## Clean the data

- Lower case all words, remove stop words and random characters that do not add predictive signal
- Misspelled words identified and corrected
- (Initially) Aggregate WBS labels to level 3
  - Significantly reduce the classification space
  - Simplifies error analysis for initial model iterations

```
'manag.' : 'management',
'engrg.' : 'engineering',
'ops' : 'operations',
'sppt' : 'support',
'mat' : 'material',
'devel.': 'development',
'sys' : 'system',
'equip.' : 'equipment',
'oper.' : 'operations',
'anom.': 'anomaly',
'integ.' : 'integration',
```

*Data shown is notional.*

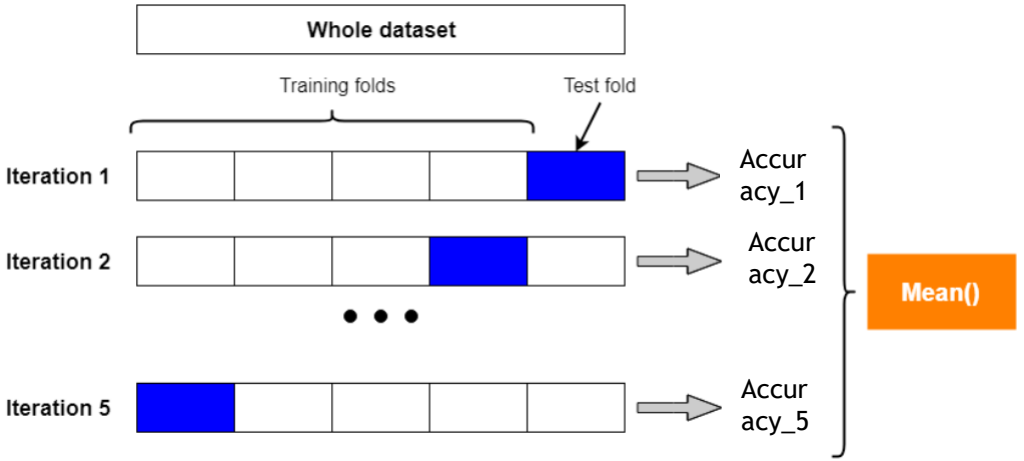


# ML Model Development (cont.)

Select and train model

- Model validation strategy:
  - Purpose:
    - Assess how well the model generalizes to "unseen" data (data not in the training set)
  - Use stratified group k-fold cross-validation (CV)
    - Similar to k-fold CV, but additional constraints are applied (see next chart)
      - Necessary due to presence of repeated rows and imbalanced classes
    - 5-folds → 80% of data assigned to training set, 20% to test set for each iteration (5 total)
      - Each data point is in testing set once → avoid overly-optimistic evaluations in scenario where "easy" data points happen to be in test set

Image from https://towardsdatascience.com/k-fold-cross-validation-explained-in-plain-english-659e33c0bc0



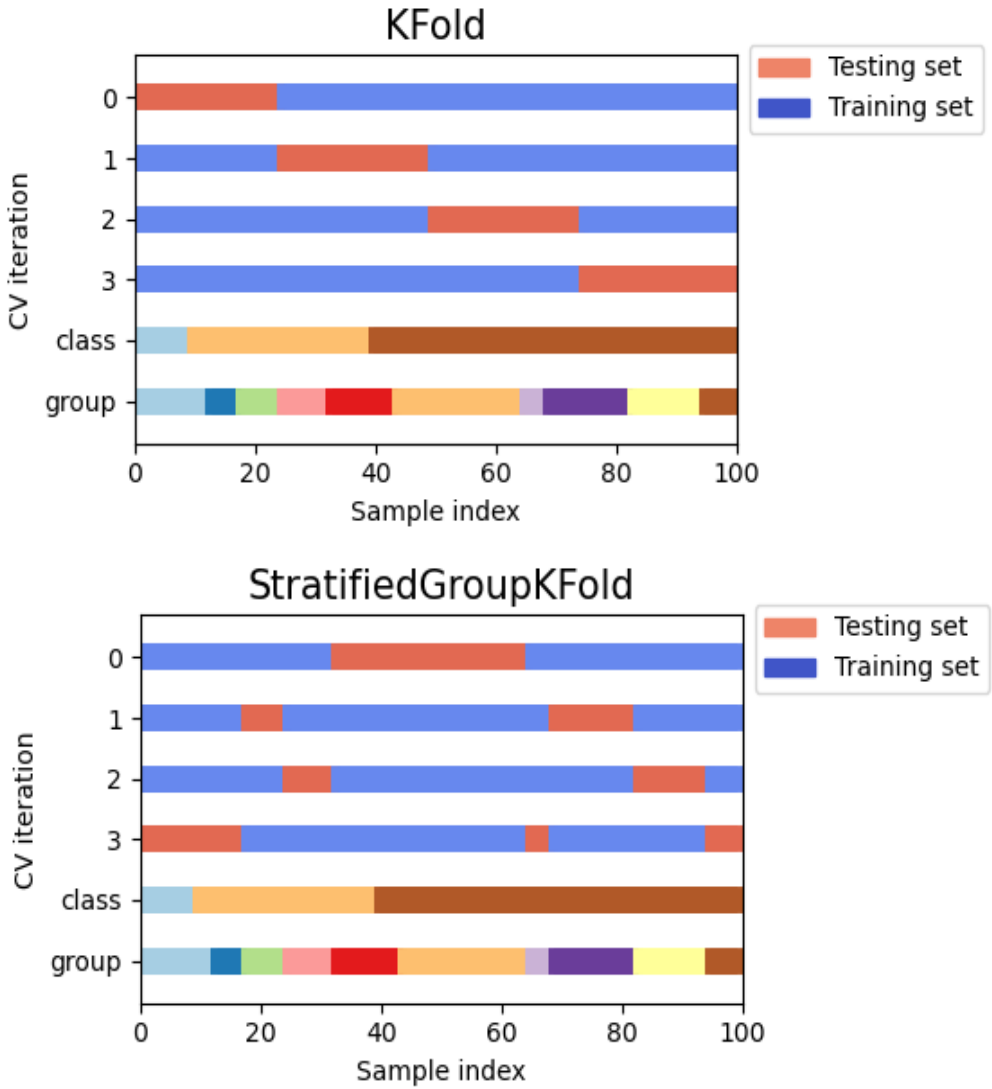

In k-fold cross validation, the dataset is split into k folds (or subsets): k-1 of those folds are used to train the model (training set), and the k'th fold is used to evaluate the model (test set). This process is repeated k times, and in each iteration, a different fold is used as the test set, ensuring each data point is in the test fold once. The resulting accuracies (k accuracies) are then averaged to get a summary accuracy score.

# ML Model Development (cont.)

Select and train model

## Relative to K-Fold, stratified group k-fold applies the following constraints

- Ensure that no group will appear in both test and train set for a given iteration
  - Each set of repeated accounts constitutes a group
  - Cost data has repeated accounts, so this constraint is crucial to avoid train-test set leakage
- Attempts to preserve the percentage of samples for each class across training and test sets, subject to non-overlapping groups constraint
  - Necessary since dataset has imbalanced classes (i.e. split of classes is not uniform) – avoids scenario of training on a class that does not appear in test set




Images from https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html#sphx-glr-auto-examples-model-selection-plot-cv-indices-py

# ML Model Development (cont.)


## Several iterations of selecting modeling techniques followed by error analysis were conducted

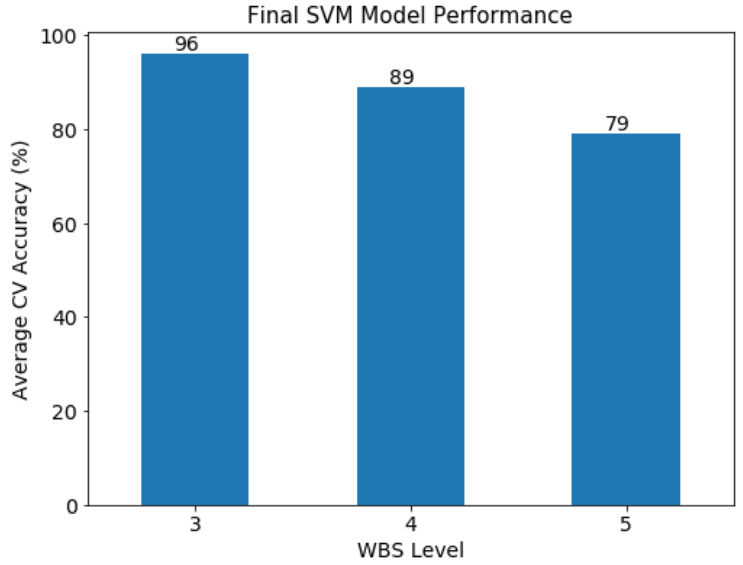
| Iteration # | Algorithm/Feature Engineering/Data Cleansing Technique | Performance (Average Accuracy, WBS Level 3) |
|---|---|---|
| 1 | Naïve-Bayes/Bag-of-Words(BOW) | 50% |
| 2 | Naïve-Bayes/BOW with additional feature | 65% |
| 3 | Support Vector Machine (SVM)/Term Frequency-Inverse Document Frequency (TF-IDF) w/ additional feature | 72% |
| … | … | … |
| n | SVM/TF-IDF with improved data cleansing, additional features, and tuned regularization coefficient | 96% |

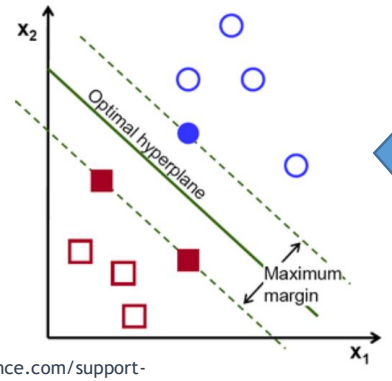
# ML Model Development (cont.)

Error Analysis & Hyperparameter Tuning

## Several iterations later, arrived at "final" model

- Algorithm
  - Support Vector Machine (SVM) classifier
- Feature Engineering
  - Term Frequency-Inverse Document Frequency (TF-IDF) vectors
  - Utilize 6 most correlated features to WBS #
- Improved data cleansing
  - Correct misspelled words and expand abbreviations in raw data
- Hyperparameter tuning
  - CV results showed overfitting → Tuned regularization parameter to help alleviate




Notional depiction of SVM model in 2-dimensions

Image from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

- tf = term frequency, the relative frequency of a term t within a document d
- idf = inverse document frequency, a measure of how common a term t is across all documents (D)

Interpretation: higher TF-IDF score → important term (algorithm will place more weight on these terms), lower TF-IDF score → less important term (algorithm places less weight on these terms)

# ML Model Development (cont.)

Model Deployment and Maintenance

- "Final" model used on follow-on data
  - 82% accuracy at WBS Level 5 classification
- Insights gained:
  - Better data cleaning (acronyms/abbreviation cleanup) and additional features had the largest impact on performance (>10%)
  - Change from NB/bag-of-words to SVM/TF-IDF had a modest impact (~5-10%)
  - Regularization parameter tuning had a minor impact (~<5%)
  - Use of n-grams had insignificant affect on results (<1%)
- Potential improvements:
  - Address cause of misclassifications and improve/retrain model for future use
  - Experiment with / determine impacts of:
    - More advanced algorithms
    - Improved feature engineering (e.g. word embeddings)

# Summary & Future Initiatives

- ML solution successfully developed and applied to the USCM cost-to-WBS mapping problem
  - Benefits
    - Reduction in time and manual effort spent on cost data normalization
      - ~2-4 weeks to develop initial mapping manually. ~1 day using ML approach.
    - Improved consistency in WBS mapping
      - Algorithm will suggest mappings that are consistent with historical mappings found in training set
  - Limitations
    - Currently only expected to perform well on follow-on program data
    - Room for improvement in modeling
- Training data should be representative of production data to foster a ML model that generalizes well
- Future Initiatives
  - Investigate methods to generalize solution to other programs
  - Investigate use of more advanced ML algorithms

# Acknowledgements & References


## Acknowledgements

This work was funded by the Space Systems Command, contract 47QFPA22F006. The authors thank Ms. Adriana Contreras and Mr. Raj Palejwala for their support of this effort.


## References and Resources


- https://www.uscmonline.com/loginpage.aspx
- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html#sphx-glr-auto-examples-model-selection-plot-cv-indices-py
- https://towardsdatascience.com/k-fold-cross-validation-explained-in-plain-english-659e33c0bc0
- https://www.spss-tutorials.com/cramers-v-what-and-why/
- https://en.wikipedia.org/wiki/Tf%E2%80%93idf

# Abstract

Cost data normalization is time consuming, but remains critical to building a defensible cost model. SSC's Unmanned Space Vehicle Cost Model (USCM) database contains over 120 normalized datasets that span nearly a 50-year history. Developing machine learning models from the older data that generalize well to newer data would be difficult due to data obsolescence. However, we have identified an example use case with promising initial results that will be examined in detail.