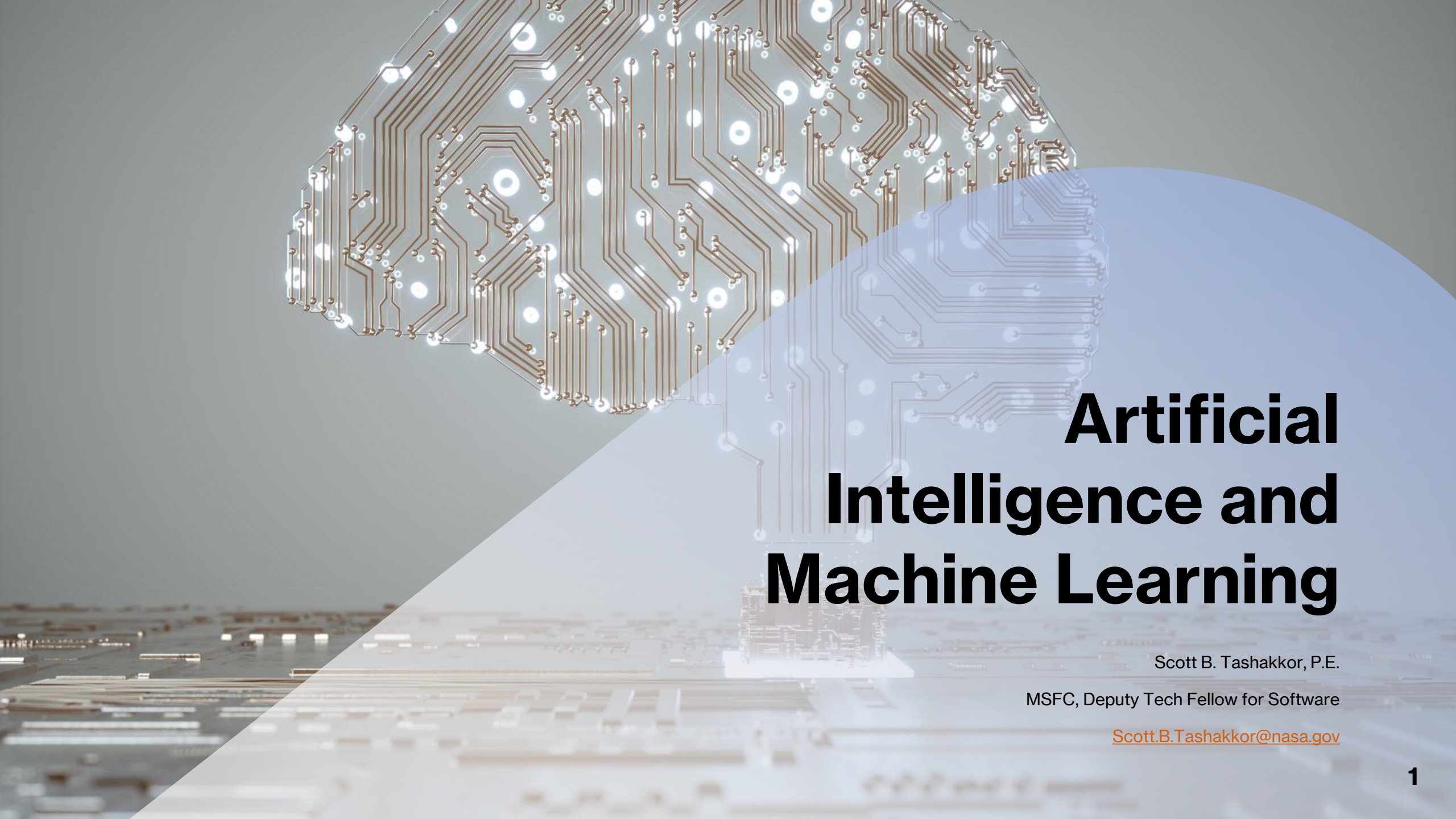


Artificial Intelligence and Machine Learning

Scott B. Tashakkor, P.E.

MSFC, Deputy Tech Fellow for Software

Scott.B.Tashakkor@nasa.gov

Agenda

- About Me
- NASA Guidance/Policy
- IT, capabilities and definitions
- Usage of AI/ML
- Summary

About Me

- NASA
  - NESC Deputy Tech Fellow for Software
  - Work at MSFC in the Embedded Software Development
  - MSFC's Lead for AI/ML efforts
    - Interfaces with Agency DT office

- Academic
  - Professional Engineer License (Thermal and Fluid Systems)
  - Aerospace Engineer and Computer Engineer degrees
  - Computational Fluid Dynamics/Numerical Methods

What is AI/ML?

- Artificial Intelligence
  - Definition, no NASA Standard, varies internally and externally
  - Computers capability to emulate functions that are associated with thinking
    - We are not at the level in which systems are “thinking” (ie synthesizing concepts)
      - No Isaac Asimov three laws, Skynet, Hal 9000…
- Machine Learning
  - Data is presented to a computer program that the machine builds an algorithm without being explicitly programmed.

AI/ML vs Autonomy

- AI/ML is Autonomy, but not all Autonomy is AI/ML
- Both concepts have been around for decades

- What is the difference? (simplified)
  - Autonomy
    - System gathers inputs, analyzes, and outputs a response
    - Can be explicitly programmed responses (Expert system)
    - Does not have to be software
      - Thermostat
    - Examples (NASA has been doing this for a long time):
      - SLS launch vehicle (short time to effect)
      - Mars Landers (delayed Communications)
  - AI/ML
    - Uses algorithms to analyze data statistically to build model
    - Is implicitly programmed (ie weights are generated for outcomes)
      - Computer cannot determine causation or independence of correlation

Pet Rock – Test works?

When to Use AI/ML

- ML is **NOT** a solution for every type of problem
- If rules, computations, or predetermined steps can be explicitly programmed, *do that*
  - Simplifies Verification/Validation
  - More deterministic
  - Order of approximation can be determined
- If rules cannot be coded explicitly
  - Complex rules
  - Rules overlap/finely tuned
- Information cannot be scaled
  - While some data can be analyzed by people, large data sets are better for ML algorithms
- Large Data sets
  - Amount of data
  - Image analysis
  - Ability to sufficiently train and test the model dataset

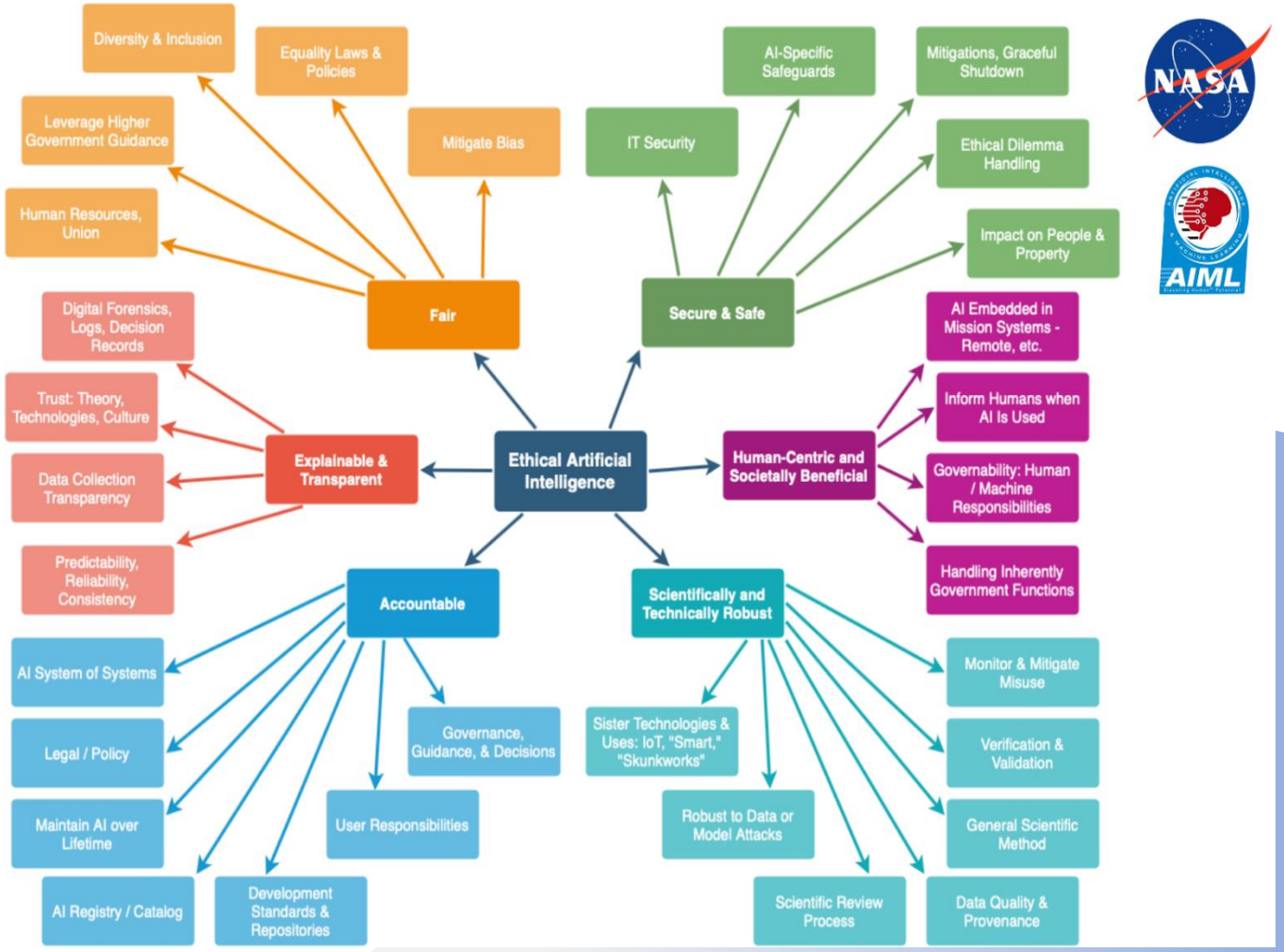
NASA Policy

- NASA Software Engineering Requirements (NPR 7150.2D)
  - Applies to all software, regardless of development methodology/strategy
  - Covers AI software in requirements definition, project, test, validation, … all phases
- NASA Software Assurance and Software Safety (NASA-STD-8739.8B)
- NASA Scientific Integrity (NPD 1920.1)
- NASA Management of NASA Scientific and Technical Information (NPD 2200.1D)
- NASA Standard for Models and Simulations (NASA-STD-7009A)

- Federal Orders/Direction
- NASA Responsible AI Plan
- OIG Reports: NASA's Management of Its Artificial Intelligence Capabilities
- NASA AI/ML Ethics Memo

AI/ML Ethics

- Fair
- Explainable and Transparent
- Accountable
- Secure and Safe
- Human-Centric and Societally Beneficial
- Scientifically and Technically Robust



NASA/TM-20210012886 NASA Framework for the Ethical Use of Artificial Intelligence (AI)

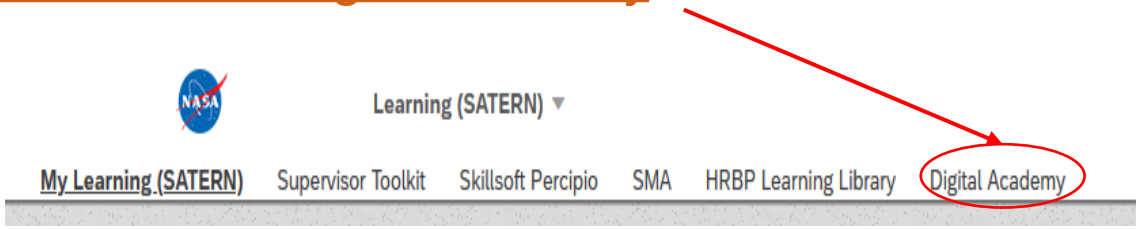
IT and usage

- Security Levels and sensitivity of data
  - NASA OCIO has been working with providers to address security approvals
- Government cloud and FedRamp approved quickens pace
- Areas with open data (scientists) are able to use most modern features

Service Name	Cloud Platform	AIML Ready	Public Access	External Partners Access	Security Category	Sensitivity	PIV Only	Availability
MACS/Google	Google Cloud Platform	Yes	Planned	Conditional (when ZeroTrust implemented)	Moderate	SBU/PII; ITAR Planned	No	Now
LaRC Data Science (sub environment to MACS/Google)	Google Cloud Platform	Yes	No	Coming soon	Moderate	SBU/PII not to include ITAR/EAR	No	Now
APPDAT	Google Cloud Platform & AWS	Yes	Yes	TBD	Moderate	SBU/PII	No	Now
MACS/AWS	AWS	Yes	Storage Buckets Only	Yes	Moderate to High	ITAR or higher	Yes	Now
Mission Cloud Platform	AWS	Yes	Yes	Yes	Moderate (High coming 2022)	ITAR/EAR/ SBU/PII/PHI	No	Now
Science Managed Cloud Environment	AWS	Yes	Yes	Yes	Low	None	No	Now
HQ ITCD AWS Managed Cloud Environment	AWS	Yes	Yes	TBD	Moderate	ITAR	No	Now
ARC NAS AIML Platform	AWS	Coming FY22	No	TBD	Moderate	Public	No	FY21
Azure Service	Azure	Yes	No	No	TBD	TBD	Yes	Now
MACS/Azure	Azure	Yes	Storage Buckets Only	Yes	Moderate to High	ITAR or higher	Yes	Now
HQ ITCD Azure Managed Cloud Environment	Azure	Yes	Yes	TBD	Moderate	ITAR	No	Now (Sandboxes); Full Production 4/2021

Agency Initiatives/learning

- https://nasa.sharepoint.com/sites/AIML
- MS Teams: Agency Wide AIML Collaboration
- NASA SATERN Digital Academy



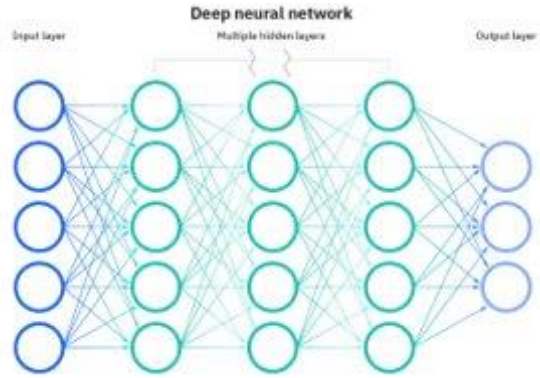
- NASA AI-usecase Inventory
  - Over 400 examples

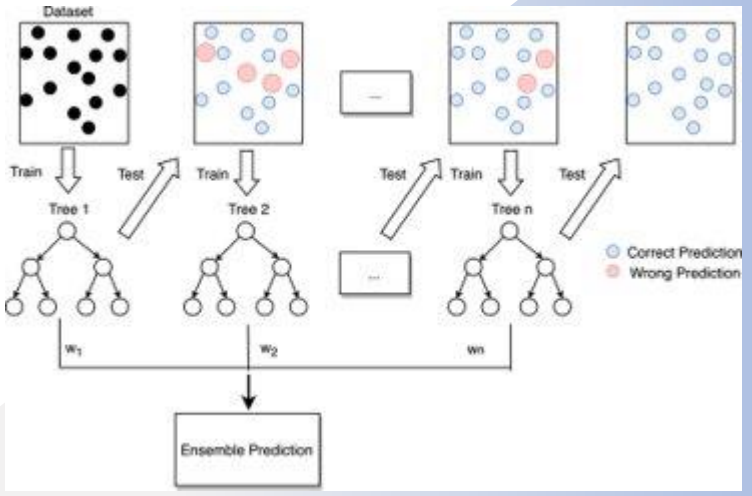
Types of ML

- Supervised Learning
  - Labeled data is presented to the system, the computer determines rules that maps inputs to outputs
- Unsupervised Learning
  - Unlabeled data is presented to the system, the computer determines the rules from the inputs only, correlations between the data only
- Reinforcement Learning
  - A reward function is created, and this is optimized to obtain the maximum reward
    - Game theory, control theory, genetic algorithms, statistics..
- Mixtures/Other

Models of ML (sample)

- Artificial Neural Networks
  - Linked layers of weights
- Decision Trees
  - Binary decisions on classifications
- Regression analysis
  - Linear regression (statistics)
- Bayesian networks
  - Probabilistic graphs
- Genetic algorithms
  - Mutations with “better” methods surviving
- Ensemble
  - Multiple combined methods





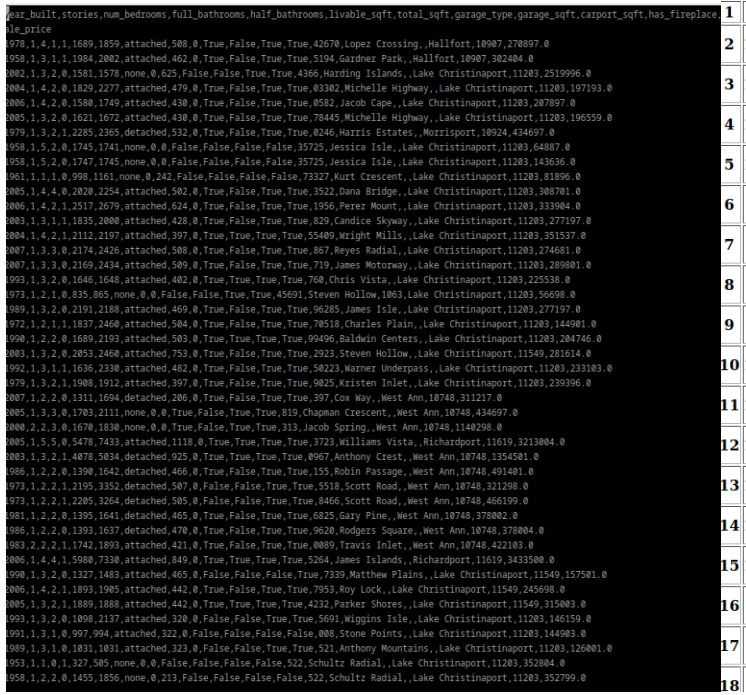
Where is AI/ML being used from Software TDT/State of Discipline?

- All centers doing some degree of AI/ML – GSFC, Ames, JPL leading -- being used as a useful tool where it best fits
- Early career employees gravitating toward use
- Agency CoP and center-level AI/ML CoP meetings held regularly and well attended
- Science (earth, space, aeronautics) and research applications readily utilizing
  - Big data, data science, data reduction, pattern recognition, image processing
  - Onboard data reduction in presence of limited comm bandwidth common
- No safety-critical AI/ML in spaceflight yet, but reliability curve may approach classical software failure levels
- Challenge in assurance and recognizing difference between probabilistic and deterministic software behavior
- General concern over high autonomy demand beyond LEO without necessary rad-hard computational hardware
- Over 400 AI projects of varying quality and usage documented (Still growing)

Example: Predict Cost of a House

- Use data to predict the cost of a house based on different parameters
- Understand and interrogate the data
- Optimize model
- Predict

Examine Data



```
year_built,stories,num_bedrooms,full_bathrooms,half_bathrooms,livable_sqft,total_sqft,garage_type,garage_sqft,carport_sqft,has_fireplace,
sale_price
1978,1,4,1,1,1689,1859,attached,508,0,True,False,True,True,42670,Lopez Crossing,,Hallfort,10907,270897.0
1958,1,3,1,1,1984,2002,attached,462,0,True,False,True,True,5194,Gardner Park,,Hallfort,10907,302404.0
2002,1,3,2,0,1581,1578,none,0,625,False,False,True,True,4366,Harding Islands,,Lake Christinaport,11203,2519996.0
2004,1,4,2,0,1829,2277,attached,479,0,True,False,True,True,03302,Michelle Highway,,Lake Christinaport,11203,197193.0
2006,1,4,2,0,1580,1749,attached,430,0,True,False,True,True,0582,Jacob Cape,,Lake Christinaport,11203,207897.0
2005,1,3,2,0,1621,1672,attached,430,0,True,False,True,True,78445,Michelle Highway,,Lake Christinaport,11203,196559.0
1979,1,3,2,1,2285,2365,detached,532,0,True,False,True,True,0246,Harris Estates,,Morrisport,10924,434697.0
1958,1,5,2,0,1745,1741,none,0,0,False,False,False,False,35725,Jessica Isle,,Lake Christinaport,11203,64887.0
1958,1,5,2,0,1747,1745,none,0,0,False,False,False,False,35725,Jessica Isle,,Lake Christinaport,11203,143636.0
1961,1,1,1,0,998,1161,none,0,242,False,False,False,False,73327,Kurt Crescent,,Lake Christinaport,11203,81896.0
2005,1,4,4,0,2020,2254,attached,502,0,True,False,True,True,3522,Dana Bridge,,Lake Christinaport,11203,308701.0
2006,1,4,2,1,2517,2679,attached,624,0,True,False,True,True,1956,Perez Mount,,Lake Christinaport,11203,333904.0
2003,1,3,1,1,1835,2000,attached,428,0,True,False,True,True,829,Candice Skyway,,Lake Christinaport,11203,277197.0
2004,1,4,2,1,2112,2197,attached,397,0,True,True,True,True,55409,Wright Mills,,Lake Christinaport,11203,351537.0
2007,1,3,3,0,2174,2426,attached,508,0,True,False,True,True,867,Reyes Radial,,Lake Christinaport,11203,274681.0
2007,1,3,3,0,2169,2434,attached,509,0,True,False,True,True,719,James Motorway,,Lake Christinaport,11203,289801.0
1993,1,3,2,0,1646,1648,attached,402,0,True,True,True,True,760,Chris Vista,,Lake Christinaport,11203,225538.0
1973,1,2,1,0,835,865,none,0,0,False,False,True,True,45691,Steven Hollow,1063,Lake Christinaport,11203,56698.0
1989,1,3,2,0,2191,2188,attached,469,0,True,False,True,True,96285,James Isle,,Lake Christinaport,11203,277197.0
1972,1,2,1,1,1837,2460,attached,504,0,True,False,True,True,70518,Charles Plain,,Lake Christinaport,11203,144901.0
1990,1,2,2,0,1689,2193,attached,503,0,True,True,True,True,99496,Baldwin Centers,,Lake Christinaport,11203,204746.0
2003,1,3,2,0,2053,2460,attached,753,0,True,False,True,True,2923,Steven Hollow,,Lake Christinaport,11549,281614.0
1992,1,3,1,1,1636,2330,attached,482,0,True,False,True,True,50223,Warner Underpass,,Lake Christinaport,11203,233103.0
1979,1,3,2,1,1908,1912,attached,397,0,True,False,True,True,9025,Kristen Inlet,,Lake Christinaport,11203,239396.0
2007,1,2,2,0,1311,1694,detached,206,0,True,False,True,True,397,Cox Way,,West Ann,10748,311217.0
2005,1,3,3,0,1703,2111,none,0,0,True,False,True,True,819,Chapman Crescent,,West Ann,10748,434697.0
2000,2,2,3,0,1670,1830,none,0,0,True,False,True,True,313,Jacob Spring,,West Ann,10748,1140298.0
2005,1,5,5,0,5478,7433,attached,1118,0,True,True,True,True,3723,Williams Vista,,Richardport,11619,3213004.0
2003,1,3,2,1,4078,5034,detached,925,0,True,True,True,True,0967,Anthony Crest,,West Ann,10748,1354501.0
1986,1,2,2,0,1390,1642,detached,466,0,True,False,True,True,155,Robin Passage,,West Ann,10748,491401.0
1973,1,2,2,1,2195,3352,detached,507,0,False,False,True,True,5518,Scott Road,,West Ann,10748,321298.0
1973,1,2,2,1,2205,3264,detached,505,0,False,False,True,True,8466,Scott Road,,West Ann,10748,466199.0
1981,1,2,2,0,1395,1641,detached,465,0,True,False,True,True,6825,Gary Pine,,West Ann,10748,378002.0
1986,1,2,2,0,1393,1637,detached,470,0,True,False,True,True,9620,Rodgers Square,,West Ann,10748,378004.0
1983,2,2,2,1,1742,1893,attached,421,0,True,False,True,True,0089,Travis Inlet,,West Ann,10748,422103.0
2006,1,4,4,1,5980,7330,attached,849,0,True,True,True,True,5264,James Islands,,Richardport,11619,3433500.0
1990,1,3,2,0,1327,1483,attached,465,0,False,False,False,True,7339,Matthew Plains,,Lake Christinaport,11549,157501.0
2006,1,4,2,1,1893,1905,attached,442,0,True,False,True,True,7953,Roy Lock,,Lake Christinaport,11549,245698.0
2005,1,3,2,1,1889,1888,attached,442,0,True,True,True,True,4232,Parker Shores,,Lake Christinaport,11549,315003.0
1993,1,3,2,0,1098,2137,attached,320,0,False,False,True,True,5691,Wiggins Isle,,Lake Christinaport,11203,146159.0
1991,1,3,1,0,997,994,attached,322,0,False,False,False,False,008,Stone Points,,Lake Christinaport,11203,144903.0
1989,1,3,1,0,1031,1031,attached,323,0,False,False,True,True,521,Anthony Mountains,,Lake Christinaport,11203,126001.0
1953,1,1,0,1,327,505,none,0,0,False,False,False,False,522,Schultz Radial,,Lake Christinaport,11203,352804.0
1958,1,2,2,0,1455,1856,none,0,213,False,False,False,False,522,Schultz Radial,,Lake Christinaport,11203,352799.0
```

	year_built	stories	num_bedrooms	full_bathrooms	half_bathrooms	livable_sqft	total_sqft	garage_type	garage_sqft	carport_sqft	has_fireplace	has_pool	has_central_heating	has_central_cooling	house_number	str
0	1978	1	4	1	1	1689	1859	attached	508	0	True	False	True	True	42670	Lop Cro
1	1958	1	3	1	1	1984	2002	attached	462	0	True	False	True	True	5194	Gar
2	2002	1	3	2	0	1581	1578	none	0	625	False	False	True	True	4366	Har Isla
3	2004	1	4	2	0	1829	2277	attached	479	0	True	False	True	True	3302	Mic Hig
4	2006	1	4	2	0	1580	1749	attached	430	0	True	False	True	True	582	Jac
5	2005	1	3	2	0	1621	1672	attached	430	0	True	False	True	True	78445	Mic Hig
6	1979	1	3	2	1	2285	2365	detached	532	0	True	False	True	True	246	Har Est
7	1958	1	5	2	0	1745	1741	none	0	0	False	False	False	False	35725	Jes
8	1958	1	5	2	0	1747	1745	none	0	0	False	False	False	False	35725	Jes
9	1961	1	1	1	0	998	1161	none	0	242	False	False	False	False	73327	Kur Cre
10	2005	1	4	4	0	2020	2254	attached	502	0	True	False	True	True	3522	Dan
11	2006	1	4	2	1	2517	2679	attached	624	0	True	False	True	True	1956	Per
12	2003	1	3	1	1	1835	2000	attached	428	0	True	False	True	True	829	Car Sky
13	2004	1	4	2	1	2112	2197	attached	397	0	True	True	True	True	55409	Wri
14	2007	1	3	3	0	2174	2426	attached	508	0	True	False	True	True	867	Rey
15	2007	1	3	3	0	2169	2434	attached	509	0	True	False	True	True	719	Jam Mot
16	1993	1	3	2	0	1646	1648	attached	402	0	True	True	True	True	760	Chr
17	1973	1	2	1	0	835	865	none	0	0	False	False	True	True	45691	Ste Hol
18	1989	1	3	2	0	2191	2188	attached	469	0	True	False	True	True	96285	Jam

Initial Training

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import ensemble
from sklearn.metrics import mean_absolute_error
import joblib

# Load the data set
df = pd.read_csv("ml_house_data_set.csv")

# Remove the fields from the data set that we don't want to include in our model
del df['house_number']
del df['unit_number']
del df['street_name']
del df['zip_code']

# Replace categorical data with one-hot encoded data
features_df = pd.get_dummies(df, columns=['garage_type', 'city'])

# Remove the sale price from the feature data
del features_df['sale_price']

# Create the X and y arrays
X = features_df.to_numpy()
y = df['sale_price'].to_numpy()

# Split the data set in a training set (70%) and a test set (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)


#purposely not best params
# Fit regression model
model = ensemble.GradientBoostingRegressor(
    n_estimators=750,
    learning_rate=0.05,
    max_depth=6,
    min_samples_leaf=9,
    max_features=0.2,
    loss='huber',
    random_state=0
)
model.fit(X_train, y_train)

# Save the trained model to a file so we can use it in other programs
joblib.dump(model, 'trained_house_classifier_model_poor.pkl')
```

- Load the Libraries
- Load the data set

- Pre-processing of Data

- Split the data into training set and test set
  - Usually 20-30% test set
    - Want to make these representative

- Setup with some base parameters for the model
  - This is similar to tuning constants it
    - Need to understand model and data

  - Train and output the model (for future)
    - This data set was small so short amount of time

Determining Feature Importance

```python
import numpy as np
import joblib

# These are the feature labels from our data set
feature_labels = np.array(['year_built', 'stories', 'num_bedrooms', 'full_bathrooms', 'half_bathroo
, 'garage_type_attached', 'garage_type_detached', 'garage_type_none', 'city_Amystad', 'city_Brownpo
', 'city_East Justin', 'city_East Lucas', 'city_Fosterberg', 'city_Hallfort', 'city_Jeffreyhaven',
iusborough', 'city_Lake Jack', 'city_Lake Jennifer', 'city_Leahview', 'city_Lewishaven', 'city_Mart
, 'city_Port Daniel', 'city_Port Jonathanborough', 'city_Richardport', 'city_Rickytown', 'city_Scot
_West Gerald', 'city_West Gregoryview', 'city_West Lydia', 'city_West Terrence'])

# Load the trained model created with train_model.py
model = joblib.load('trained_house_classifier_model_poor.pkl')

# Create a numpy array based on the model's feature importances
importance = model.feature_importances_

# Sort the feature labels based on the feature importance rankings from the model
feauture_indexes_by_importance = importance.argsort()

# Print each feature label, from most important to least important (reverse order)
for index in feauture_indexes_by_importance:
    print("{} - {:.2f}%".format(feature_labels[index], (importance[index] * 100.0)))
~
~
~
```

- Load the model and see how the weights from each input data affects the prediction

- Notice certain parameters at the bottom are highest... this is relative to the correlation

```
city_Wendybury - 0.03%
city_Brownport - 0.03%
city_Port Adamtown - 0.03%
city_Port Daniel - 0.04%
city_Davidfort - 0.07%
city_West Lydia - 0.07%
city_Clarkberg - 0.09%
city_Port Jonathanborough - 0.09%
garage_type_detached - 0.10%
city_Morrisport - 0.13%
city_Jenniferberg - 0.13%
city_West Gerald - 0.16%
has_central_heating - 0.17%
city_East Amychester - 0.18%
city_Lewishaven - 0.18%
city_Richardport - 0.18%
city_Lake Carolyn - 0.23%
city_North Erinville - 0.24%
city_East Lucas - 0.26%
city_Lake Dariusborough - 0.30%
city_West Gregoryview - 0.36%
city_West Ann - 0.38%
has_central_cooling - 0.42%
half_bathrooms - 0.45%
city_South Anthony - 0.49%
city_Hallfort - 0.54%
city_Justinport - 0.57%
stories - 1.09%
city_Scottberg - 1.17%
city_Chadstad - 1.30%
city_Lake Christinaport - 1.40%
garage_type_attached - 1.45%
num_bedrooms - 1.46%
city_Port Andrealand - 1.66%
city_Lake Jack - 1.83%
has_fireplace - 1.97%
city_Jeffreyhaven - 3.23%
carport_sqft - 3.38%
year_built - 3.75%
has_pool - 4.19%
city_Coletown - 4.63%
full_bathrooms - 5.20%
garage_type_none - 5.22%
garage_sqft - 7.38%
livable_sqft - 18.80%
total_sqft - 24.79%
```

How do we determine Parameter settings?

```python
# Load the data set
df = pandas.read_csv("ml_house_data_set.csv")

# Remove the fields from the data set that we don't want to include in our model
del df['house_number']
del df['unit_number']
del df['street_name']
del df['zip_code']

# Replace categorical data with one-hot encoded data
features_df = pandas.get_dummies(df, columns=['garage_type', 'city'])
del features_df['sale_price']

X = features_df.to_numpy()
y = df['sale_price'].to_numpy()

# Split the data set in a training set (70%) and a test set (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Create the model
model = ensemble.GradientBoostingRegressor()

# Parameters we want to try
param_grid = {
    'n_estimators': [500, 1000, 3000],
    'max_depth': [4, 6],
    'min_samples_leaf': [3, 5, 9, 17],
    'learning_rate': [0.1, 0.05, 0.02, 0.01],
    'max_features': [1.0, 0.3, 0.1],
    'loss': ['ls', 'lad', 'huber']
}

# Define the grid search we want to run. Run it with four cpus in parallel.
gs_cv = GridSearchCV(model, param_grid, n_jobs=4)

# Run the grid search - on only the training data!
gs_cv.fit(X_train, y_train)

# Print the parameters that gave us the best result!
print(gs_cv.best_params_)

# After running a .....long..... time, the output will be something like
# {'loss': 'huber', 'learning_rate': 0.1, 'min_samples_leaf': 9, 'n_estimators': 3000, 'max_features': 0.1, 'max_

# That is the combination that worked best.
```

```
{'learning_rate': 0.05, 'loss': 'huber', 'max_depth': 6, 'max_features': 0.1, 'min_samples_leaf': 9, 'n_estimators': 3000}
Training Set Mean Absolute Error: 45402.1348
Test Set Mean Absolute Error: 58163.4539
```

- This is tuning of the parameters to get a better estimate
- This method effectively runs through a “grid” of the parameters and trains multiple models
- This is computationally expensive
- While this can make a “Better” model for this data set, this does not mean that model will work for future

Train with our new parameter settings

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import ensemble
from sklearn.metrics import mean_absolute_error
import joblib

# Load the data set
df = pd.read_csv("ml_house_data_set.csv")

# Remove the fields from the data set that we don't want to include in our model
del df['house_number']
del df['unit_number']
del df['street_name']
del df['zip_code']

# Replace categorical data with one-hot encoded data
features_df = pd.get_dummies(df, columns=['garage_type', 'city'])

# Remove the sale price from the feature data
del features_df['sale_price']

# Create the X and y arrays
X = features_df.to_numpy()
y = df['sale_price'].to_numpy()

# Split the data set in a training set (70%) and a test set (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Fit regression model
model = ensemble.GradientBoostingRegressor(
    n_estimators=3000,
    learning_rate=0.05,
    max_depth=6,
    min_samples_leaf=9,
    max_features=0.1,
    loss='huber',
    random_state=0
)
model.fit(X_train, y_train)

# Save the trained model to a file so we can use it in other programs
joblib.dump(model, 'trained_house_classifier_model_best.pkl')
```

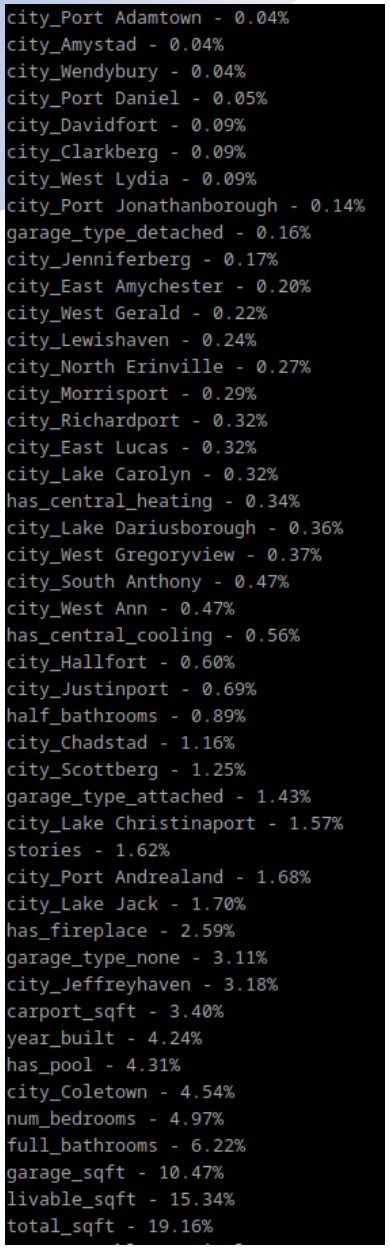
- Everything else is the same

- Modified input parameters here

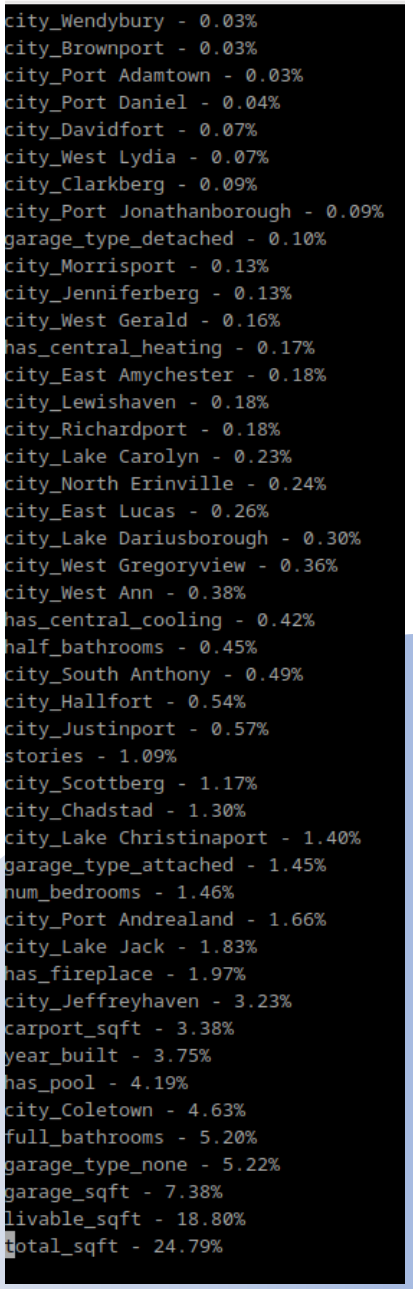
Updated Feature Importance

- Most parameters are still in the same order
  - Unless they were close:
    - Garage_type_attached – 1.43%
      - Prev 1.45%
    - City_Lake Christinaport – 1.57%
      - Prev 1.40%

- Some large changes:
  - Garage_type_none less important
    - 3.11% from 5.22%



Updated



Original

Predictions

```
    0,      # Justinport
    0,      # Lake Carolyn
    0,      # Lake Christinaport
    0,      # Lake Dariusborough
    0,      # Lake Jack
    0,      # Lake Jennifer
    0,      # Leahview
    0,      # Lewishaven
    0,      # Martinezfort
    0,      # Morrisport
    0,      # New Michele
    0,      # New Robinton
    0,      # North Erinville
    0,      # Port Adamtown
    0,      # Port Andrealand
    0,      # Port Daniel
    0,      # Port Jonathanborough
    0,      # Richardport
    0,      # Rickytown
    0,      # Scottberg
    0,      # South Anthony
    0,      # South Stevenfurt
    0,      # Toddshire
    0,      # Wendybury
    0,      # West Ann
    0,      # West Brittanyview
    0,      # West Gerald
    0,      # West Gregoryview
    0,      # West Lydia
    0       # West Terrence
]

# scikit-learn assumes you want to predict the values for lots of houses at once, so it expects an array.
# We just want to look at a single house, so it will be the only item in our array.
homes_to_value = [
    house_to_value
]

# Run the model and make a prediction for each house in the homes_to_value array
predicted_home_values = model.predict(homes_to_value)

# Since we are only predicting the price of one house, just look at the first prediction returned
predicted_value = predicted_home_values[0]

print("This house has an estimated value of ${:,.2f}".format(predicted_value))
```

- Set up the information
- Then call the model prediction

- With best Parameters:
  - $583,089.70
- With original Parameters:
  - $543,105.06

- Optimization Example took over 6 hours to run
  - Training was only ~20 seconds
    - 10,860 times as long
    - For approximately 7% difference in this case

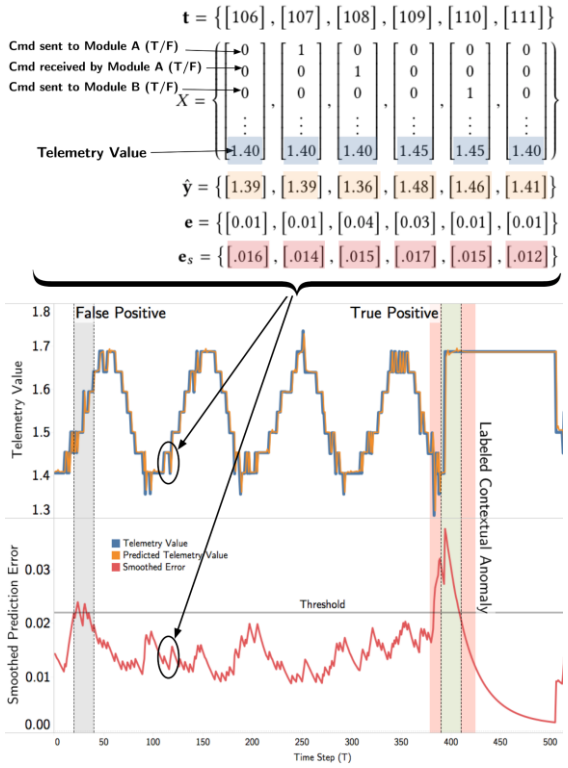
Thoughts (1 of 2)

- If parameters are changed based on data set (training) how are these verified?
  - Turbulence models had constants that could be changed, those would invalidate calculations with previous values
- ML is used for “big data”
  - Single flights and small amounts of data do not provide accurate representation
  - These are not physics/equation models
  - Interpolation inside of the ML data set is more accurate than extrapolating outside
- Data has to already be generated
  - ML models are not creating new data, they are finding patterns/correlations in presented data
  - If no valid data presented previously model can “predict” incorrect results
- Verification/Validation
  - Models are statistical and non-deterministic with less insight into the outputs

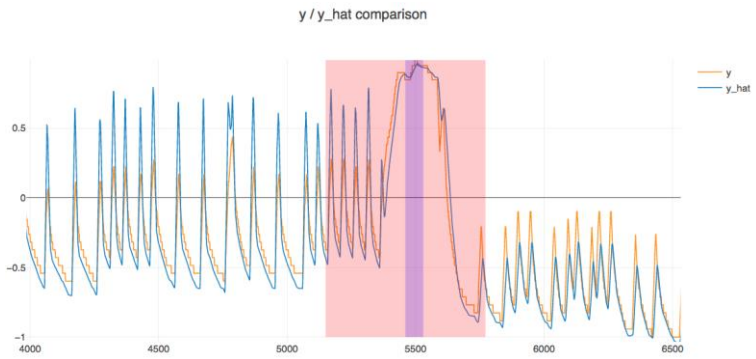
Thoughts (2 of 2)

- Software Assurance
  - Without insight (explainable) assurance is unknown
  - How to define the quality of the model and results for the usage
- Safety Criticality
- Challenges
  - Ensuring AI is defined and used within defined context and it's strengths and weaknesses are understood
  - Ensure awareness of probabilistic/non-deterministic nature
  - Develop assurance/verification strategies for AI – (i.e. need to re-verify and re-validate periodically after learning)

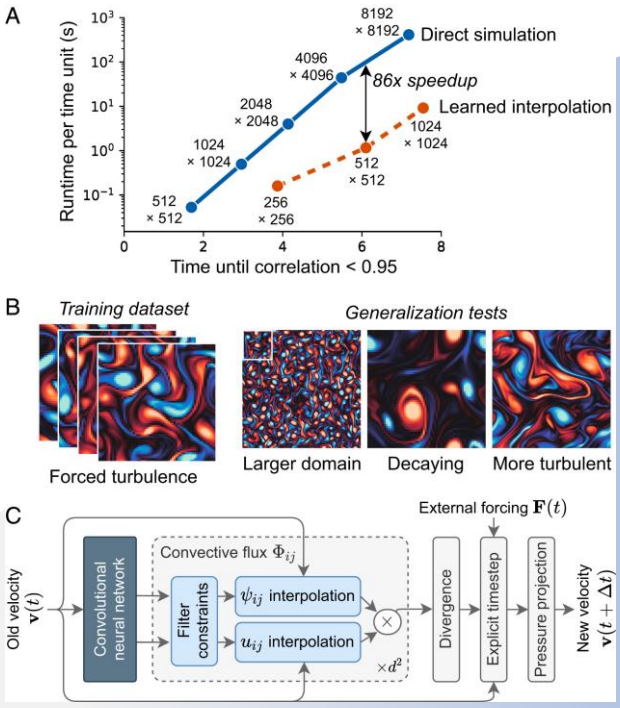
Other applications/examples



- https://github.com/khundman/telemanom



https://www.pnas.org/doi/10.1073/pnas.2101784118



Summary

- Autonomy does not have to be AI/ML
  - Understand when to use the correct tool for the correct problem
- NASA policies are still required to be followed
  - The policies do not explicitly state AI/ML but the processes still apply
- Ethical uses of AI/ML can add complexity to the problem
- IT restrictions on the data
- There are different types of AI/ML and each has strengths and weakness
- Implementation of AI/ML has multiple “knobs” that can be change
  - Data is the key

Questions?