AGENDA

• A Bit of Background
• Game 😊
• Prototyping
• Summary & QA
MOTIVATION

Architectural drivers

<<precedes>>

Architectural design

<<precedes>>

Architectural documentation

<<precedes>>

Architectural evaluation

Clouds

Machine Learning

IoT

Mobile

Big Data

Blockchain
ADD 3.0

ML Design Concepts:
- Problem
- Algorithm Family
- Algorithm

Carnegie Mellon University
Software Engineering Institute
SMART DECISIONS GAME

First presented at SATURN 2015
A fun, lightweight way to introduce architectural design and ADD
Available at:
http://smartdecisionsgame.com/

softserve
Carnegie Mellon University
Software Engineering Institute
SHORT QUIZ 😊

What’s the name of this company in AI field?

10x increase over the past 2 years!
AI PROGRESS SINCE 1950s
MYTHS AND FICTION ABOUT ARTIFICIAL BEINGS

R.U.R. (Karel Čapek)  1921

Golem (Bible)  
~1000 BC

Sumerian Anunnaki creating the first man  
~2300 BC
THE CURRENT STATE OF AI

Number of neurons:

- 5K
- 1M
- 16M
- 71M
- 760M
- 22B
- 86B

Carnegie Mellon University
Software Engineering Institute

[DICTIONARY STATEMENT A] Approved for public release and unlimited distribution.
GAME CHALLENGE OVERVIEW

Business Use Case

Banner A

RED PILL
RESTART YOUR LIFE AT 10-YEARS-OLD WITH ALL THE KNOWLEDGE YOU HAVE NOW

Banner B

BLUE PILL
FAST FORWARD TO AGE 50 WITH $10 MILLION IN YOUR BANK ACCOUNT

Which ad will the user choose?
GAME CHALLENGE OVERVIEW

Marketecture Diagram

\[ \text{CTR} = \frac{\text{Number of click-throughs}}{\text{Number of impressions}} \times 100(\%) \]

- Web Servers
  - Hundreds of servers
  - Massive logs from multiple sources
- Online user
  - Predict CTR and show appropriate ad
- CTR Prediction System
  - Train
  - Ingest
- Web Logs
WHY DO WE NEED MACHINE LEARNING?
WHY NOT JUST CODING?!

Most of the today’s AI problems:

• Deal with an infinite problem space – think about how many words are there in the English language

• Poorly defined – we still do not know how our brain solves problems

Therefore, traditional rule-based hand-coding for such problems suffers a 'complexity collapse' and is not feasible
MACHINE LEARNING APPROACH

Instead of writing a program by hand, we use a set of examples to train the algorithm.

Developer writes code

Algorithm “writes code”
ML BUILDING BLOCKS

1. Training Data
2. Testing Data
3. New Data
4. Model
5. Machine Learning Algorithm
6. Predictions
TYPES OF LEARNING
SUPERVISED LEARNING

- **Input examples** and corresponding **ground truth outputs** are provided.
- The goal is to learn general rules that map a new example to the predicted output.
SUPERVISED LEARNING

Example: Given a set of house features along with corresponding house prices, predict a price for a new house based on its features (e.g. size, location, etc.)
UNSUPERVISED LEARNING

• Only **input examples** are provided
• No explicit information about ground truth
• The algorithm tries to discover the internal structure of the data based on some prior knowledge about desired outcome
UNSUPERVISED LEARNING

Example: Given a set of customer transactions discover what would be the best way to group them into clusters based on customer similarity
THE CURRENT STATE OF AI

SUPERVISED LEARNING

UNSUPERVISED LEARNING

Carnegie Mellon University
Software Engineering Institute
ITERATION 1:
What type of learning best fits a given use case?

Select from: supervised or unsupervised
ITERATION 1:
Supervised or Unsupervised Learning?

**Historical Dataset**

<table>
<thead>
<tr>
<th>RED PILL</th>
<th>BLUE PILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banner A</td>
<td>Banner B</td>
</tr>
</tbody>
</table>

**Logs:**
- User X Features | Banner A | Click 0
- User Y Features | Banner B | Click 0
- User X Features | Banner B | Click 1
- ...

**Train**
- Input: Logs

**ML Algorithm**

**Predict**
- Input: User Features
- Output: Most preferred banner to show
MACHINE LEARNING CARDS
MACHINE LEARNING CARDS

ITERATION 2: PROBLEM TYPE

ITERATION 3a: ALGORITHM FAMILY

ITERATION 3b: ML ALGORITHM
PROBLEM TYPES
CLASSIFICATION

Key Highlights:

- Identifies which category an object belongs to
- Supervised learning problem

Examples:

- Detect fraudulent transactions (one-class)
- Categorize emails by spam or not spam (binary)
- Categorize articles based on their topic (multi-class)
- Detect objects on the image (multi-label)
REGRESSION

Key Highlights:

• Predict a continuous value associated with an object
• Supervised learning problem

Examples:

• Predict stock prices from market data
• Score a credit application based on historical data
• Estimate demand for a given product
CLUSTERING

Key Highlights:

• Group similar objects into clusters
• Unsupervised learning problem

Examples:

• Discover audiences to target on social networks
• Group checking data based on GEO-proximity
• Detect common topics in corporate knowledge base
ANOMALY DETECTION

Key Highlights:

- Identify observations that do not conform to an expected pattern
- Addresses both supervised and unsupervised learning

Examples:

- Identify fraudulent transactions or abnormal customer behavior
- In manufacturing, detect physical parts that are likely to fail in the near future
ITERATION 2:
What type of problem best fits a given use case?

Select problem card from: classification, regression, clustering or anomaly detection
ITERATION 2:
What type of problem?

**Historical Dataset**

**Logs:**
- User X Features | Banner A | Click 0
- User Y Features | Banner B | Click 0
- User X Features | Banner B | Click 1
- ...

**Input:** Logs

**Train**

**Input:** User Features

**Output:** Most preferred banner to show

**Predict**

**ML Algorithm**

Banner A  Banner B

RED PILL  BLUE PILL

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
FAMILIES AND ALGORITHMS
CLASSIFICATION FAMILIES

Artificial Neural Network
- Algorithm Family
- Description: ANN is a computational model based on the structure and functions of biological neural networks. It can be used for both classification and regression unsupervised learning. The model works well for data and complex non-linear relationships but can be computationally expensive.
- Characteristics:
  - Big Data — performance positively correlated with data volume, but at the expense of computational resources.
  - Small Data — the less data available, the more training data required.
  - Overall, other algorithms will outperform neural networks.
- Artificial Neural Network
- Support Vector Machines
- Algorithm Family
- Description: SVMs are supervised learning algorithms which can be used for both classification or regression problems. Given labeled training data, the algorithms learn a hyperplane which maximizes the margin between the classes.
- Characteristics:
  - Big Data — computation and memory-intensive while training, data sampling can be used as a workaround.
  - Small Data — good accuracy even for small # of observations.
  - Imbalanced Data — compensates imbalance through class weights, works fine out of the box for moderately imbalanced data.
  - Ease of Use — moderate number of parameters.
- Implementations: Linear SVM, Non-Linear SVM, One-Class SVM.

Generalized Linear Models
- Algorithm Family
- Description: A class of linear models with a non-linear transformation of the dependent variable is linearly related to independent variables. Generalized Least Squares is often used as a way of unifying various statistical models.
- Characteristics:
  - Big Data — regression models are generally considered more efficient for big data compared to classification models.
  - Small Data — sufficient generalization even for very small datasets.
  - Imbalanced Data — can be handled by stratified bootstrap technique.
  - Ease of Use — models tuning is user-friendly.
- Implementations: Classification/Regression Decision Trees, Random Forest, Isolation Forest.

Decision Tree Learning
- Algorithm Family
- Description: Decision Tree Learning uses a decision tree structure to go from observations about an item to conclusions about the item's target value. It is one of the most interpretable families of machine learning algorithms. This approach can also be used for both classification or regression problems.
- Characteristics:
  - Big Data — interpretation is getting worse on large datasets.
  - Small Data — sufficient generalization even for very small datasets, but can lead to overfitting.
  - Imbalanced Data — can be handled by stratified bootstrap technique.
  - Results Interpretation — represented by a set of decision rules.
  - Ease of Use — models tuning is user-friendly.
- Implementations: Classification/Regression Decision Trees, Random Forest, Isolation Forest.

Instance-Based Learning (sometimes called instance learning) is a family of learning algorithms that, from an example, try to learn how to perform explicit generalization, compares new instances with instances seen in training and stores in a form that can be used for both classification or regression problems.
- Characteristics:
  - Big Data — good accuracy, even for a small number of instances.
  - Small Data — more robust with data that are imbalanced and sparse.
  - Imbalanced data — can be handled by stratified bootstrap technique.
  - Ease of Use — models tuning is user-friendly.
- Implementations: Classification/Regression Decision Trees, Random Forest, Isolation Forest.
# Classification Algorithms

## Linear SVM
**Algorithm:** Support Vector Machines
**Description:** A SVM algorithm with a linear kernel.

**Characteristics:**
- Accurate — sufficient for most tasks, but there is limited support for multi-class problems.
- Training Speed — linear on the number of support vectors.
- Predictive Speed — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).

## Non-Linear SVM
**Algorithm:** Support Vector Machines
**Description:** A SVM algorithm with a non-linear kernel and RBF activation.

**Characteristics:**
- **Accuracy** — sufficient accuracy for most tasks, but there is limited support for multi-class problems.
- **Training Speed** — linear on the number of support vectors.
- **Predictive Speed** — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).

## Bayesian Naive Classifier
**Algorithm:** Generative Linear Models
**Description:** A single probability classifier based on Bayes’ Theorem with an assumption of strong independence between features.

**Characteristics:**
- **Accuracy** — sufficient accuracy for most tasks, but there is limited support for multi-class problems.
- **Training Speed** — linear on the number of support vectors.
- **Predictive Speed** — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).

## Logistic Regression
**Algorithm:** Generative Linear Models
**Description:** A linear regression classifier with a binary outcome.

**Characteristics:**
- **Accuracy** — sufficient accuracy for most tasks, but there is limited support for multi-class problems.
- **Training Speed** — linear on the number of support vectors.
- **Predictive Speed** — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).

## Classification/Regression Trees
**Algorithm:** Decision Tree Learning
**Description:** A class of algorithms that consists of either one decision tree with the non-linear behavior or two decision trees with a different structure that are combined to form a regression tree.

**Characteristics:**
- **Accuracy** — sufficient accuracy for most tasks, but there is limited support for multi-class problems.
- **Training Speed** — linear on the number of support vectors.
- **Predictive Speed** — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).

## K-Nearest Neighbors (KNN)
**Algorithm:** Instance-Based Learning
**Description:** A non-parametric supervised learning method used for classification and regression, a type of lazy learning, as the results of the training phase are not used for classification and regression.

**Characteristics:**
- **Accuracy** — sufficient accuracy for most tasks, but there is limited support for multi-class problems.
- **Training Speed** — linear on the number of support vectors.
- **Predictive Speed** — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).

## Radius Neighbors
**Algorithm:** Instance-Based Learning
**Description:** A variant of the KNN algorithm, where the nearest neighbors are determined by a radius hyperparameter.

**Characteristics:**
- **Accuracy** — sufficient accuracy for most tasks, but there is limited support for multi-class problems.
- **Training Speed** — linear on the number of support vectors.
- **Predictive Speed** — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).

## Random Forest
**Algorithm:** Decision Tree Learning
**Description:** An ensemble learning method that constructs a model using multiple decision trees, each of which is trained on a random subset of the training data.

**Characteristics:**
- **Accuracy** — sufficient accuracy for most tasks, but there is limited support for multi-class problems.
- **Training Speed** — linear on the number of support vectors.
- **Predictive Speed** — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).

## Multilayer Perceptron
**Algorithm:** Artificial Neural Networks
**Description:** A class of algorithms that consists of at least one hidden layer with a non-linear behavior.

**Characteristics:**
- **Accuracy** — sufficient accuracy for most tasks, but there is limited support for multi-class problems.
- **Training Speed** — linear on the number of support vectors.
- **Predictive Speed** — slow to near-infinite for large-scale problems.

**Tips:**
- Uses all training data.
- Margins maximization.

**Implementations:** R, Python (SVMLIB).
DECISION DRIVERS
FAMILY DRIVERS

- **Big Data** – scalability and ability to leverage from new data
- **Small Data** – ability to learn from a few examples
- **Imbalanced Data** – ability to distinguish rare events
- **Results Interpretation** – human-friendly results
- **Online Learning** – ability to continuously train from new data
- **Ease of Use** – number of parameters to manually tune

**Decision Tree Learning**

**Description:** Decision Tree Learning uses a decision tree structure to go from observations about an item to conclusions about the item’s target value. It is one of the most interpretable families of machine learning algorithms. This approach can be used for both classification or regression problems.

**Characteristics:**

- ★★ Big Data — interpretability is getting worse on large datasets
- ★★ Small Data — sufficient generalization even for very small dataset, but can lead to overfitting
- ★★ Imbalanced Data — can be handled by stratified bootstrap technique
- ★★★ Results Interpretation — represented by a set of decision rules
- ★★★ Online Learning — can be trained sequentially
- ★★★ Ease of Use — models tuning is user-friendly

**Algorithms:** Classification/Regression Decision Tree, Random Forest, Isolation Forest.

---

Carnegie Mellon University
Software Engineering Institute

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
ALGORITHM DRIVERS

- **Accuracy** – ability to solve complex problems
- **Training Speed** – training runtime performance
- **Prediction Speed** – production runtime performance
- **Overfitting Resistance** – ability to generalize to new data
- **Probabilistic Interpretation** – return results as probabilities

**K-Nearest Neighbors (KNN)**

**Description:** A non-parametric supervised learning method used for classification and regression; a type of lazy learning, where the function is only approximated locally and all computation is deferred until classification.

**Characteristics:**
- ★★★ Accuracy — sufficient accuracy for most tasks, but there is a tradeoff between accuracy vs avoiding overfitting
- ★★★ Training Speed — training time is high on large datasets
- ★ Prediction Speed — full training set processing is required
- ★★ Overfitting Resistance — with an increase of k nearest training objects, the probability of overfitting decreases
- ★★★ Probabilistic Interpretation — naturally determined by the inference process

**Tips:**
- ✓ One of the simplest machine learning algorithms
- ✓ Good choice for low dimensional space

**Implementations:** R, Python (scikit-learn).
ITERATION 3:
Select a family and an algorithm card that would best fit a given use case

Family Key Drivers: Big Data, Imbalanced Data, Ease of Use
Algorithm Key Drivers: Accuracy, Training and Prediction Speed
DESIGN PROCESS

- Architectural drivers
- Architectural design
- Architectural documentation
- Architectural evaluation
DESIGN PROCESS

- Architectural drivers
- Architectural design
- Architectural documentation
- Architectural evaluation

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
PROTOTYPING AND EVALUATION SESSION
PROTOTYPING FOR EVALUATION

Initial Architecture Analysis
- Assess the existing system and identify gaps that pose risks for new requirements.
- Perform architecture analysis, update the existing architecture, and select candidate technologies to replace or augment existing technologies.
- Is this a proof-of-concept system design?
  - NO
  - Perform architecture analysis, and select a reference architecture and candidate technologies.
  - YES

Vertical Evolutionary Prototype
- Go to the throwaway-prototype chart for each technology that poses a risk.
- Integrate technologies and test critical functionality and quality attribute scenarios.
- Are there known issues discovered with any technology?
  - NO
  - Integrate technologies and test critical functionality and quality attribute scenarios.
  - YES

Throwaway Prototype
- Is the selected technology now on the market?
  - NO
  - Does the prototype satisfy the specific scenario?
    - NO
    - Recommit the technology selection, and go to the throwaway-prototype chart.
    - YES
    - Are there more configurations, data models, or resources to try?
      - NO
      - NO
      - YES
  - YES
  - Change the candidate technology
  - YES
  - YES
  - NO
  - NO
  - YES

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
## RESULTS SUMMARY

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Training Time</th>
<th>Prediction Time</th>
<th>Tuning Time</th>
<th>Initial Accuracy</th>
<th>Final Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>2.61</td>
<td>0.47</td>
<td>94.44</td>
<td>81.61%</td>
<td>83.05%</td>
</tr>
<tr>
<td>KNeighbors</td>
<td>0.41</td>
<td>44.29</td>
<td>84.27</td>
<td>80.57%</td>
<td>83.05%</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.12</td>
<td>0.05</td>
<td>45.94</td>
<td>82.93%</td>
<td>82.93%</td>
</tr>
<tr>
<td>MLP</td>
<td>0.80</td>
<td>0.08</td>
<td>164.04</td>
<td>66.25%</td>
<td>82.90%</td>
</tr>
<tr>
<td>SVM</td>
<td>177.78</td>
<td>54.87</td>
<td>973.73</td>
<td>82.83%</td>
<td>82.83%</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>5.93</td>
<td>0.04</td>
<td>82.91</td>
<td>82.69%</td>
<td>82.69%</td>
</tr>
<tr>
<td>Decision Trees</td>
<td>0.03</td>
<td>0.005</td>
<td>52.97</td>
<td>73.16%</td>
<td>82.36%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.02</td>
<td>0.01</td>
<td>0</td>
<td>78.46%</td>
<td>78.46%</td>
</tr>
</tbody>
</table>
KEY TAKEAWAYS

• Machine Learning solution design is an iterative process
• ADD principles help make ML design decisions in a systematic way
• ML Cards aim to select candidate algorithms from a wide variety of alternatives
• Prototyping is necessary to validate design decisions
QUESTIONS?
WE’VE GOT THE ANSWERS.