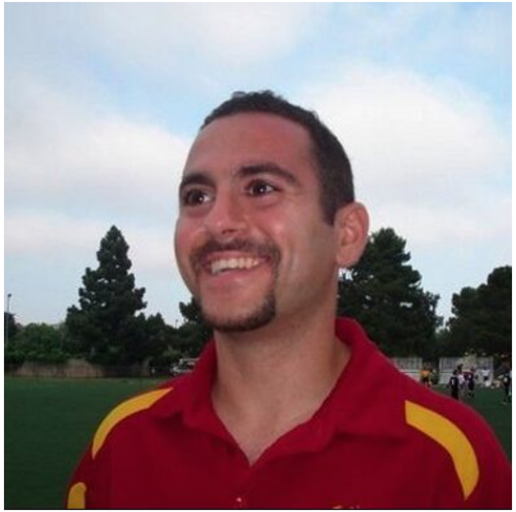
Deploy ML Models into Production

Armen Donigian

Who am I?

- Computer Science Undergrad degree @UCLA
- Computer Science Grad degree @USC
- 15+ years experience as Software & Data Engineer
- Computer Science Instructor
- Mentor @Udacity Deep Learning Nanodegree
- Real-time wagering algorithms @GamePlayerNetwork
- Differential GPS corrections @Jet Propulsion Laboratory, landing sequence for Mars Curiosity
- Most recently Director of Data Science Engineering @ZestFinance, worked with Baidu, JD.com, Prestige Financial, Synchrony & Ford Financial
- Productionalized over two dozen Machine Learning Models

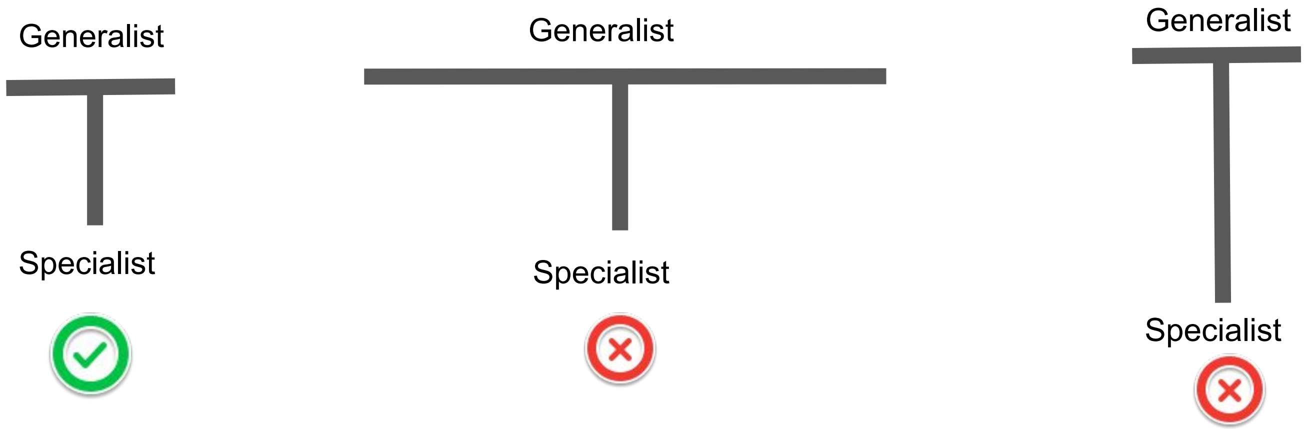


Goals, Breadth vs Depth...

Goal: Provide context of the *requirements*, *tools* & *methodologies* involved with deploying a machine learning model into production.

Slides will provide you with *breadth*.

Notebooks will provide you with *depth* (i.e. implementation details).

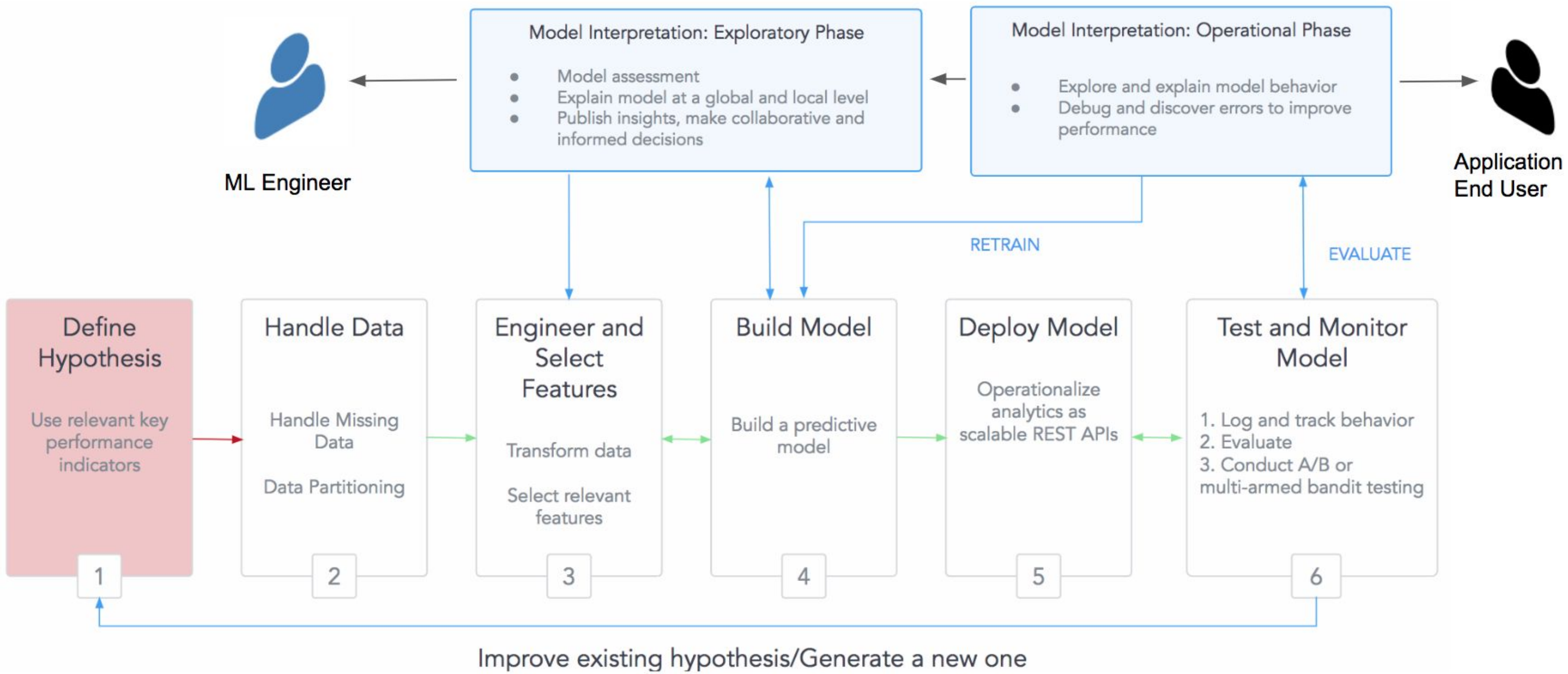


Lesson Roadmap

- **Overview of Data Science Workflow (20 mins)**
  a. Identify Business Objectives & Stakeholders
  b. Acquire Data
  c. Build Model
  d. Evaluate Model
  e. Communicate Results
- **Version Control ML vs SW Projects (95 mins)**
  a. Directory structure
  b. Data Versioning
  c. Model Build Governance
  d. Environment & Package management
  e. Exploratory Data & Business Analysis Results
  f. ML Interpretability Results
- **Cloud Model Hosting Hardware Cost Estimation**
  a. Amazon AWS Case Study
  b. Google Case Study
- **Pitfalls during Model Development**
  a. Choose input signals carefully
  b. Data to remove
- **Offline vs Online Model Training**
  a. XGBoost case study
  b. Vowpal-wabbit case study
- **Offline vs Online Model Inference**
  a. XGBoost hosted via clipper
  b. XGBoost on batch data
- **Monitoring ML model in production**
- **Production ML Model Best Practices**

Machine Learning Deployment Pipeline

Data Science Workflow



[Reference]

Overview of Data Science Workflow

1. Define Business Objective & Stakeholders	Identify goals & state hypothesize
	Define criteria for what success looks like, state your *assumptions*
	How much time & money do we have to work with?
	Identify Business Sponsor
	Define functional & non-functional requirements...How often are predictions needed? How long do we expect predictions to take? Where will predictions be made (on-cloud, on-prem)?
	Define which machine learning evaluation metric will be used to quantify quality of predictions

2. Acquire Data	Create a set of questions for identifying correct data set
	Identify data sources, window of time, data formats (CSV, XML, JSON etc) , data dictionary, features & target
	What existing transformations have been made to the data?
	Determine which tools/frameworks (Spark, Scikit-Learn) will be used to retrieve & work with data?
	Determine which database(s) the data is stored in.

Overview of Data Science Workflow Continued...

3. Build Model	Pre-processing: How will we handle missing value(s)? How will we handle missing type(s)? Outlier(s)? Class imbalance?
	Exploratory Data Analysis: Observe correlations, descriptive & inferential statistics. EDA is often the goal of many data science projects.
	Feature Scaling, Normalization, Engineering
	Check for Data Leakages, Knowledge Leaks
	Feature Selection, Hyper-parameter Tuning
	Learning Estimator Selection, Learner Ensembling
	Create a pipeline to run model train in an automated way

4. Evaluate Model	How good of a fit (quantitatively) is the chosen learners with respect to the chosen evaluation metric (Log-Loss, AUC, Accuracy, Precision, Recall etc)?
	Define criteria for what success looks like
	How well does model predictions qualitatively solve our business objective?
	Model Interpretability

Overview of Data Science Workflow Continued...

5. Communicate Results	Determine net benefit value for correct predictions
	Determine net cost value for incorrect predictions
	Perform analysis of revenue, cost & benefit with respect to financials
	State assumptions (i.e. conversion rate)
	Why should we trust this machine learning model?
	How do you understand this machine learning model works?
	Create a pipeline to run analysis in an automated way
	Determine the best format to present results to business stakeholders

How Version Control differs for ML vs SW projects

How Version Control Differs for ML Projects

Requirements:

Data Science project is a cross disciplinary function

- Data Engineers
- ML Engineers
- Business Analysts
- Software Engineers to put model into production
- DevOps personnel to maintain production operations
- Non-technical stakeholders interested in main takeaways

A) Directory structure is as important as code quality, at times even more important

B) Data Versioning

C) Environment & Package management

D) Business Analysis Results

E) ML Interpretability Results

A) Directory Structure

Requirements:

Data Science is a team sport; thus, we need a way to communicate the diverse set of artifacts.

Directory structure should clearly help organize, communicate and make it easier to find what you're looking for

Reduce human error and bugs by conforming to conventions (secrets out of version control)

Facilitate reproducible & repeatable executions of the data science pipeline

Find previous execution results (ex1: log of train, validation & test loss for each execution, ex2: feature engineering vs inference timing results)

Solution(s):

- mlflow
- Cookiecutter Data Science
- Pachyderm
- Modeldb (paper, docs, implementation)
- Manually configured project structure

A) Directory Structure Example

Project Organization

```
├── LICENSE            <- Assert your rights for ownership & conditions for use, extensions & re-distribution
├── README.md          <- The top-level README for developers using this project
├── data
│   ├── external       <- Data from third party sources
│   ├── interim        <- Intermediate data that has been transformed
│   ├── processed      <- Processed data sets for modeling
│   └── raw            <- The original, immutable data dump
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator initials, and a short "-" delimited description, e.g.
│                         "1.0-jqp-initial-data-exploration"
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials
│
├── reports            <- Generated analysis as documents, slide decks, spreadsheets, HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting.
│
├── requirements.txt   <- Optional: The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.py           <- makes project pip installable (pip install -e .) so src can be imported
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
```

B) Data Versioning

Requirements:

Data is immutable (no overwrite)

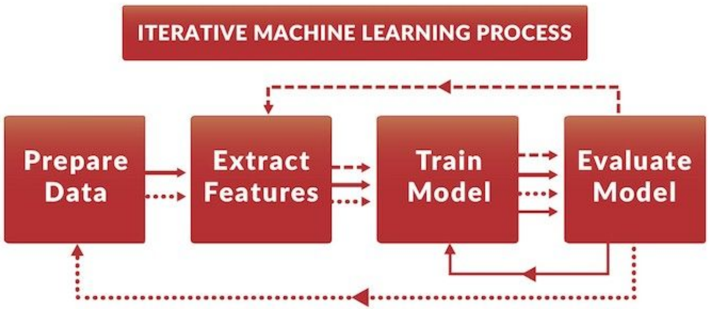
Save intermediate data artifacts

Reproducible & Repeatable (inputs & outputs need to be tracked)

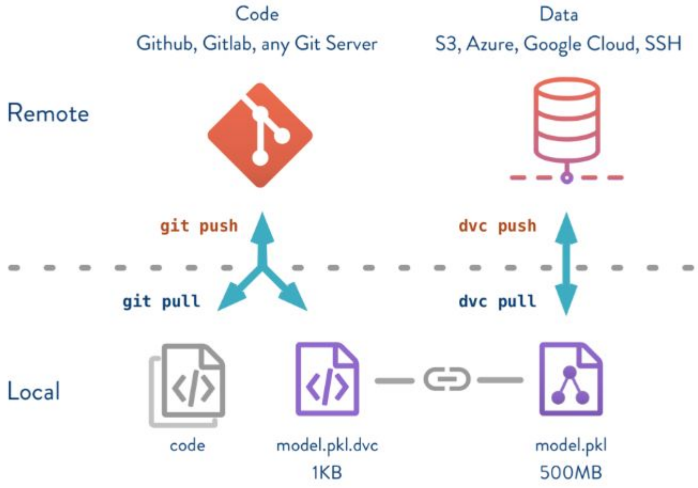
Problem: Data files are typically larger than what most version control tools allow

Solution(s):

- pachyderm
- AWS S3
- Git Large File Storage
- git-annex
- dat
- Network File Server (Ceph, FreeNAS, ZFS)
- dvc



Reference

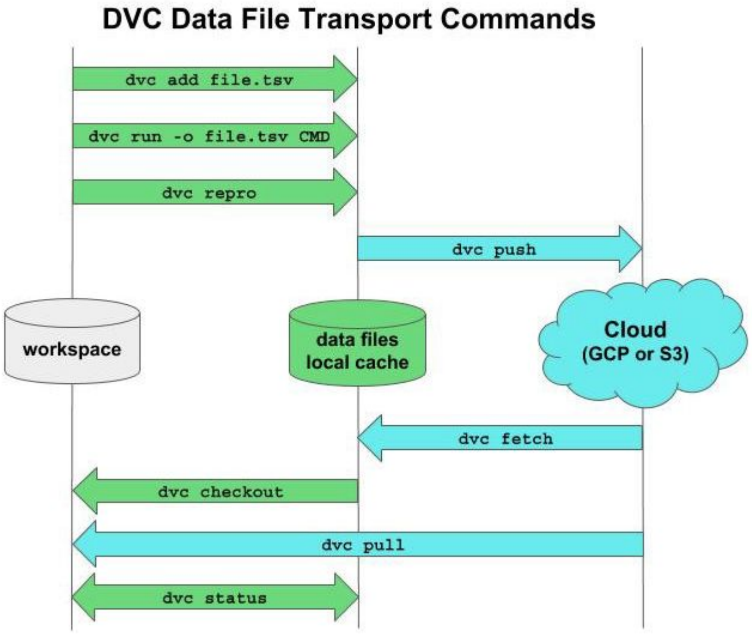


Reference

B) Data Versioning Example

DVC allows storing and versioning source data files, ML models, intermediate results with Git, without checking the file contents into Git.

```
In [1]: !dvc add ../data/external/allyears2k.csv

        Saving '../data/external/allyears2k.csv' to cache '.././dvc/cache'.
        Saving information to '../data/external/allyears2k.csv.dvc'.
```



[Reference]

```
!yes | dvc run -d ../src/models/Static_Model_Pitfalls_of_Model_Development.py -d ../data/external/allyears2k.csv \
               -o ../data/processed/ \
               python ../src/models/Static_Model_Pitfalls_of_Model_Development.py

Running command:
        python ../src/models/Static_Model_Pitfalls_of_Model_Development.py
numpy: 1.14.3
pandas: 0.23.0
sklearn: 0.19.1
xgboost: 0.72

Label Encode Target into Integers...

Get Training Data...
Original shape: (43978, 31)
After columns dropped shape: (43978, 13)

Naive One-Hot-Encode for features: ['UniqueCarrier', 'Dest', 'Origin']

Total number of features before encoding: 13

Total number of features after encoding: 286

Label Encode Target into Integers...

Get Training Data...
Original shape: (43978, 31)
After columns dropped shape: (43978, 13)

Naive One-Hot-Encode for features: ['UniqueCarrier', 'Dest', 'Origin']

Total number of features before encoding: 13

Total number of features after encoding: 286
[23:56:50] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 50 extra nodes, 0 pruned nodes, max_depth=5
[0]     train-error:0.348395
[23:56:50] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 48 extra nodes, 0 pruned nodes, max_depth=5
[1]     train-error:0.345861

Feature Importances...

Feature Importances {'DayofMonth': 4, 'Year': 7, 'CRSArrTime': 9, 'Distance': 4, 'Cancelled': 6, 'UniqueCarrier__HP':
3, 'CRSElapsedTime': 9, 'CRSDepTime': 2, 'Dest__BWI': 1, 'UniqueCarrier__WN': 1, 'Origin__MDW': 1, 'Origin__JAX': 1,
'Origin__HNL': 1}
Accuracy: 64.87%

Confusion Matrix...
[[3442 2826]
 [1809 5117]]
```

C) Model Build Governance

Requirements:

A way to track the activities during model build process

Packaging format for reproducible runs on any platform

Record and query experiments: code, data, config, and results

Facilitate reproducibility & repeatability of model build process

Solution(s):

- mlflow (github, tutorial, docs)

C) Model Build Governance

```python
# Logging test scores
mlflow.log_param("accuracy", accuracy)
mlflow.log_param("precision", precision)
mlflow.log_param("AUC", auc)
mlflow.log_param("TP", matrix[0][0])
mlflow.log_param("FP", matrix[0][1])
mlflow.log_param("FN", matrix[1][0])
mlflow.log_param("TN", matrix[1][1])
for metric, value in additional_data.items():
    print("Logging {} {}".format(metric, value))
    if metric == "importances" or metric=="params":
        for feat, importance in value.items():
            if metric == "importances":
                prefix = "imp_"
            elif metric=="params":
                prefix = "params_"
            mlflow.log_param(prefix + feat, importance)
    mlflow.log_param(metric, value)
run_id = mlflow.tracking.active_run().info.run_uuid
print("Run with id %s finished" % run_id)
```

mlflow GitHub Docs

Run b1bf57f140a34b2ea97cd3a4669f4a85

Experiment Name: Default Start Time: 2018-08-29 01:29:42 Source: ipykernel_launcher.py User: arm

▾ Parameters

Name	Value
AUC	0.7157366365886004
FN	2009
FP	3182
TN	5564
TP	3758
accuracy	0.6423206780128161
imp_CRSArrTime	32
imp_CRSDepTime	18
imp_CRSElapsedTime	15
imp_DayOfWeek	13
imp_DayofMonth	18
imp_Distance	20
imp_Year	34
importances	{'DayofMonth': 18, 'CRSArrTime': 32, 'CRSElapsedTime': 15, 'DayOfWeek': 13, 'Year': 34, 'CRSDepTime': 18, 'Distance': 20}
params	{'objective': 'binary:logistic', 'booster': 'gbtree', 'eval_metric': 'auc', 'eta': 0.01, 'nround': 1000, 'tree_method': 'exact', 'max_depth': 5, 'subsample': 0.5, 'min_child_weight': 1, 'silent': 1, 'seed': 42}

```
Usage: mlflow [OPTIONS] COMMAND [ARGS]...

Options:
  --version  Show the version and exit.
  --help     Show this message and exit.

Commands:
  azureml      Serve models on Azure ML.
  download     Download the artifact at the specified DBFS...
  experiments  Manage experiments.
  pyfunc       Serve Python models locally.
  run          Run an MLflow project from the given URI.
  sagemaker    Serve models on SageMaker.
  server       Run the MLflow tracking server.
  sklearn      Serve scikit-learn models.
  ui           Launch the MLflow tracking UI.
```

127.0.0.1:5000/#/

mlflow GitHub Docs

Experiments

Default

Default

Experiment ID: 0 Artifact Location: /Users/arm/code/ML_Models_to_Production/notebooks/mlruns/0

Search Runs: metrics.rmse < 1 and params.model = "tree" Search

Filter Params: alpha, lr Filter Metrics: rmse, r2 Clear

25 matching runs Compare Selected Download CSV

Date ▾	User	Source	Version	AUC	FN	FP	TN	TP	accuracy	imp_CRSArrTime	imp_CRSDepTime	imp_CRSElapsedTime	imp_DayOfWeek	imp_DayofMonth
2018-08-29 01:29:42	arm	ipykernel_launcher.py		0.7157366365886004	2009	3182	5564	3758	0.6423206780128161	32	18	15	13	18

D) Environment & Package Management

Requirements:

A way for others to reproduce results

Reduce time to reproduce other people’s work

Facilitate reproducibility & repeatability of other people's work

Solution(s):

- Conda (env & package management, Python & R)
- Packrat (project specific env & package manager for R)

D) Environment & Package Management Example

We'll be using conda to manage OS system libraries & python dependencies.

To create a new environment from an existing conda manifest file...

```
conda env create -f=environment.yml
```

To update a new environment from an existing conda manifest file...

```
conda env update -f=environment.yml
```

To export dependencies from your current environment...

```
conda env export > environment.yml
```

After you've created or update an environment, you should source it…

```
source activate py36_oreilly_ml_prod_course
```

When you're working, you can deactivate the current environment…

```
source deactivate
```

deploy_ml_to_production_toolkit / **environment.yml**

```yaml
name: py36_oreilly_ml_prod_course
channels:
  - defaults
dependencies:
  - alabaster=0.7.10=py36h174008c_0
  - anaconda=5.2.0=py36_3
  - anaconda-client=1.6.14=py36_0
  - anaconda-project=0.8.2=py36h9ee5d53_0
  - appnope=0.1.0=py36hf537a9a_0
  - appscript=1.0.1=py36h9e71e49_1
  - asn1crypto=0.24.0=py36_0
  - astroid=1.6.3=py36_0
  - astropy=3.0.2=py36h917ab60_1
  - attrs=18.1.0=py36_0
  - babel=2.5.3=py36_0
  - backcall=0.1.0=py36_0
  - backports=1.0=py36ha3c1827_1
  - backports.shutil_get_terminal_size=1.0.0=py36hd7a2ee4_2
  - beautifulsoup4=4.6.0=py36h72d3c9f_1
  - bitarray=0.8.1=py36h1de35cc_1
  - bkcharts=0.2=py36h073222e_0
  - blas=1.0=mkl
  - blaze=0.11.3=py36h02e7a37_0
  - bleach=2.1.3=py36_0
  - blosc=1.14.3=hd9629dc_0
  - bokeh=0.12.16=py36_0
  - boto=2.48.0=py36hdbc59ac_1
  - bottleneck=1.2.1=py36hbd380ad_0
  - bzip2=1.0.6=h1de35cc_5
  - ca-certificates=2018.03.07=0
  - certifi=2018.4.16=py36_0
  - cffi=1.11.5=py36h342bebf_0
  - chardet=3.0.4=py36h96c241c_1
  - click=6.7=py36hec950be_0
  - cloudpickle=0.5.3=py36_0
  - clyent=1.2.2=py36hae3ad88_0
  - colorama=0.3.9=py36hd29a30c_0
  - contextlib2=0.5.5=py36hd66e5e7_0
```

E) Exploratory Data & Business Analysis Results

Requirements:

Data Science projects in industry have specific business objectives (increase profits, reduce costs, increase cross sell etc)

Business Analysis methodology to quantify value of deploying a new predictive model into production (ie: How do you know your model is successful?)

What assumptions are behind your analysis? Which conditions make your assumptions invalid?

Solutions:

- Business Analysis methodology is specific to the business model. Choose the right model evaluation metric & quantify in terms of profits & loss...
  - Metric Selection for Classification ML models
    - Binary Models (Accuracy, LogLoss, Confusion Matrix, Gain & Lift chart, KS, Precision, Recall, AUC, ROC plot, more details)
      - Which type of errors to reduce (Type 1 or Type 2, more details)
    - Multiclass Models (F1, more details)
  - Metric Selection for Regression ML models (RMSE, MSE, MAE, r2, more details)

E) Exploratory Data & Business Analysis (Part 1)

Summary Statistics

Dataset info

Number of variables	32
Number of observations	43978
Total Missing (%)	19.4%
Total size in memory	10.7 MiB
Average record size in memory	256.0 B

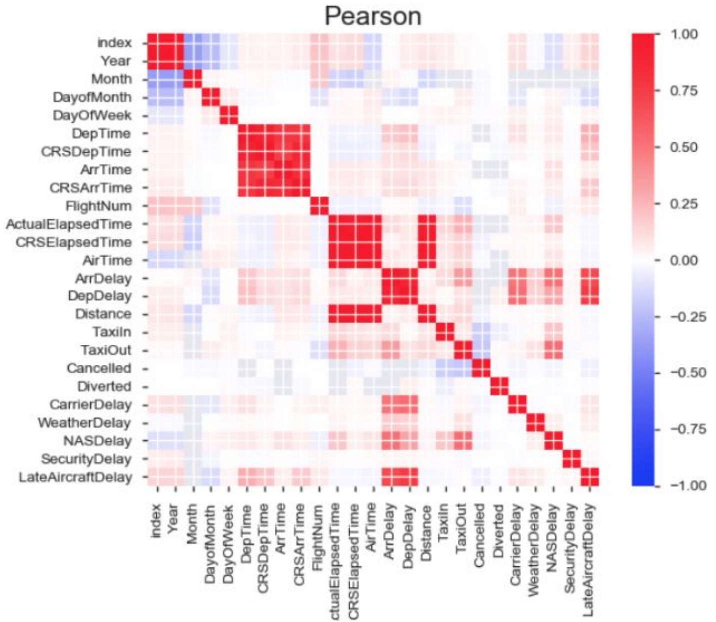
Variables types

Numeric	17
Categorical	7
Boolean	3
Date	0
Text (Unique)	0
Rejected	5
Unsupported	0

Warnings

- `ActualElapsedTime` has 1195 / 2.7% missing values Missing
- `AirTime` is highly correlated with `CRSElapsedTime` (ρ = 0.98769) Rejected
- `ArrDelay` has 1514 / 3.4% zeros Zeros
- `ArrDelay` has 1195 / 2.7% missing values Missing
- `ArrTime` has 1195 / 2.7% missing values Missing
- `CRSArrTime` has 569 / 1.3% zeros Zeros
- `CRSDepTime` is highly correlated with `DepTime` (ρ = 0.91498) Rejected
- `CRSElapsedTime` is highly correlated with `ActualElapsedTime` (ρ = 0.98409) Rejected
- `CancellationCode` has 43757 / 99.5% missing values Missing
- `CarrierDelay` has 7344 / 16.7% zeros Zeros
- `CarrierDelay` has 35045 / 79.7% missing values Missing
- `DepDelay` has 6393 / 14.5% zeros Zeros
- `DepDelay` has 1086 / 2.5% missing values Missing
- `DepTime` has 1086 / 2.5% missing values Missing
- `Dest` has a high cardinality: 134 distinct values Warning
- `Distance` is highly correlated with `AirTime` (ρ = 0.97816) Rejected
- `LateAircraftDelay` has 7140 / 16.2% zeros Zeros
- `LateAircraftDelay` has 35045 / 79.7% missing values Missing
- `NASDelay` has 7388 / 16.8% zeros Zeros
- `NASDelay` has 35045 / 79.7% missing values Missing
- `Origin` has a high cardinality: 132 distinct values Warning
- `SecurityDelay` has 8914 / 20.3% zeros Zeros
- `SecurityDelay` is highly skewed (γ1 = 26.096) Skewed
- `SecurityDelay` has 35045 / 79.7% missing values Missing
- `TailNum` has 16024 / 36.4% missing values Missing
- `TailNum` has a high cardinality: 3501 distinct values Warning
- `TaxiIn` has 623 / 1.4% zeros Zeros
- `TaxiIn` has 16026 / 36.4% missing values Missing
- `TaxiOut` has 557 / 1.3% zeros Zeros
- `TaxiOut` has 16024 / 36.4% missing values Missing
- `WeatherDelay` has 8840 / 20.1% zeros Zeros
- `WeatherDelay` is highly skewed (γ1 = 26.234) Skewed
- `WeatherDelay` has 35045 / 79.7% missing values Missing
- `Year` is highly correlated with `index` (ρ = 0.99897) Rejected

Correlations



Target Analysis

IsDepDelayed
Categorical

Distinct count	2
Unique (%)	0.0%
Missing (%)	0.0%
Missing (n)	0

Toggle details

Value	Count	Frequency (%)
YES	23091	52.5%
NO	20887	47.5%

E) Exploratory Data & Business Analysis (Part 2)

	Actual	
Predicted	**Flight Not Delayed**	**Flight Delayed**
Flight Not Delayed	**Correct**	**False Positive (Type 1)** Cost: $500, customer sentiment & loyalty
Flight Delayed	**False Negative (Type 2)** Cost: $100, customer inconvenience	**Correct**

Let's refer to the test set confusion matrix which resulted from the model development process covered earlier.

```
Confusion Matrix...
[[3768 3178]
 [2009 5558]]

 True Positive (TP, correct prediction): 3,768.
 Assume flight on-time results in $100 of revenue per customer.

 True Negative (TN, correct prediction): 5,558
 Assume flight which has been delayed results in $100 of revenue per customer.

 False Positive (FP, incorrect prediction): 3,178
 Assume the inconvenience of a delayed flight when the customer was notified it will be on-time is $500 of cost
 per customer.

 False Negative (FN, incorrect prediction): 2,009
 Assume the inconvenience of a flight on-time when the customer was notified it will be delayed is $100 of cost
 per customer.
```

```
Business Impact = Revenue * TP + Revenue * TN - Cost * FP - Cost * FN
```

```
business_impact = 700 * 3768 + 700 * 5558 - 3178 * 500 - 2009 * 100
business_impact
```

```
4738300
```

F) ML Interpretability

Requirements:

How do we trust a machine learning model?

For regulated industries, it's required to comply with established regulations?

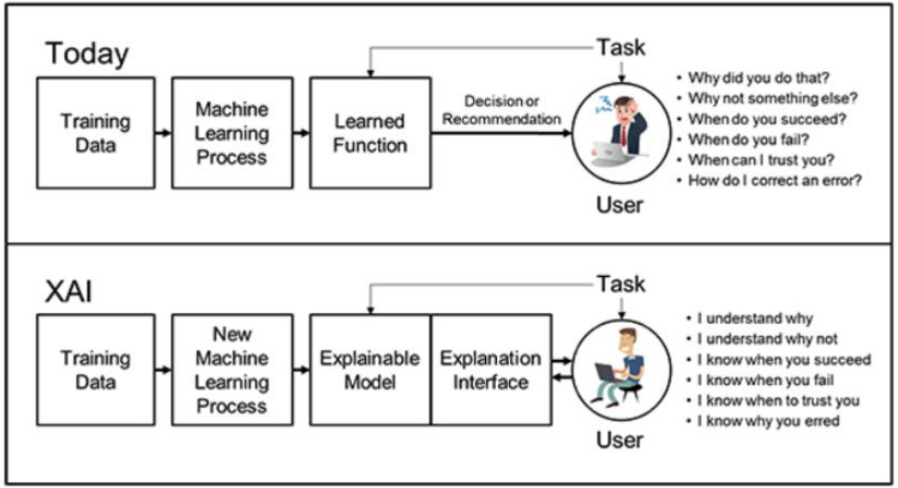
Model approval from legal or compliance stakeholders

Solutions:

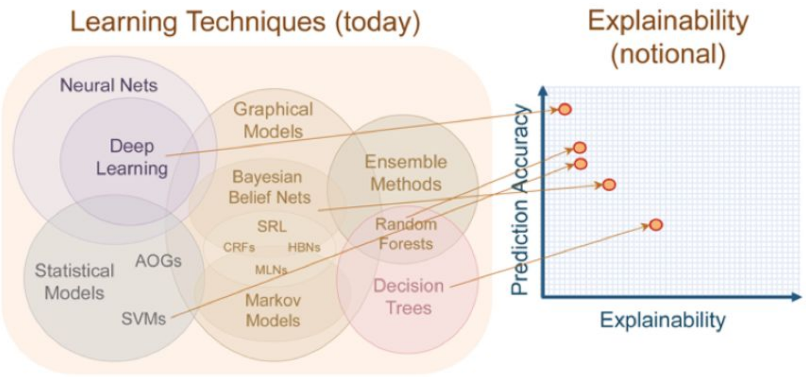
No one size fits all solution, but some recent advances to help explainability...

- SHAP: SHapley Additive exPlanations (paper, implementation)
- DeepLIFT (aka DeepExplainer in SHAP Repo): Deep Learning Important FeaTures (paper, implementation)
- H2O ML Interpretability (paper, implementation)
- A Guide for Making Black Box Models Explainable (docs, implementation)

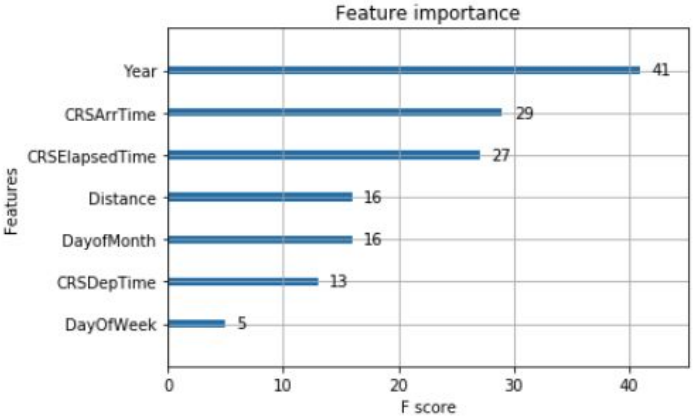
F) ML Interpretability & Explainability Example

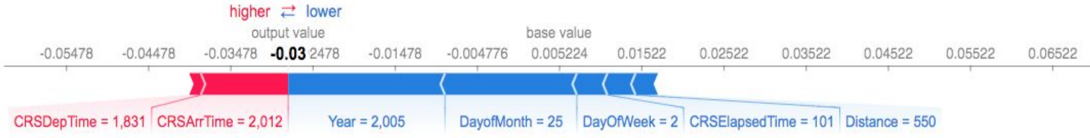


Reference



Reference





Cloud Model Hosting Cost Estimation Tools

Production ML Model Best Practices

- Begin with a starter seed project template
  - Helps enforce best practices
- Create baseline model (simple > complex)
  - Useful for comparison against a future more complex model
- Understand & validate data pipeline
  - bias in, bias out
- Instrument input feature monitoring
  - Best done async in parallel to invocation of predict
- Leverage reproducible & repeatable practices covered earlier to treat configs data & output results as code
  - Increase the likelihood of your work to be adopted
- Provide interpretability & explainability of model outputs to build trust
- Provide projected business impact of putting model into production

Highly recommend reading [Rules of ML] (from Google)

Hardware Cost Estimation

Requirements:

On-premise or On-Cloud?

Average Transactions per second to support?

Number of Transactions during peak utilization?

Load balancer, number of servers (fixed, elastic), monitoring server, object storage & logging store

System availability expectations (SLOs & SLIs come in handy)

Cloud Pricing Calculators: Google Cloud Calculator, Amazon ML Pricing, Microsoft Azure Calculator

Cloud Vendor	Batch Fees	Real-Time Prediction Fees
Amazon AWS (details)	$0.10 per 1,000 predictions, rounded up to the next 1,000	$0.0001 per prediction, rounded up to the nearest penny
Microsoft Azure (details)	$0.50 per 1,000 transactions	$0.50 per 1,000 transactions
Google Cloud Platform (details)	$0.09262 per node hour	$0.056 per node hour

Google Cloud Platform Case Study

[Visualize GCP Billing using BigQuery and Data Studio]

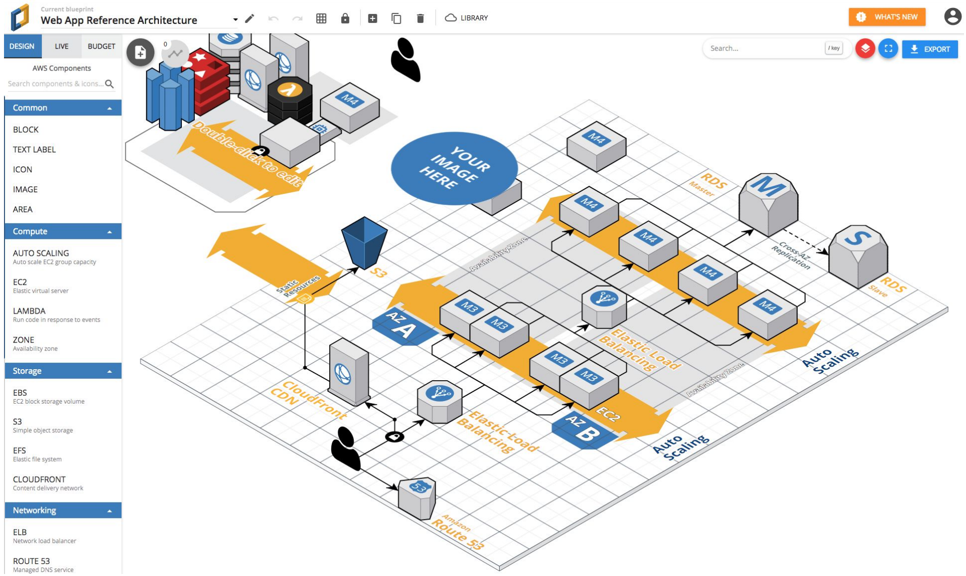
1. Follow procedure in [Export Billing Data to BigQuery]
2. Make a copy of [Billing Report Demo]
3. [Create & Manage Labels] to slice/dice your billing reports
    a. Labels can be based on team (team:data_science)
    b. Labels can be based on component (component: flight_model)
    c. Labels can be based on environment (environment:prod)

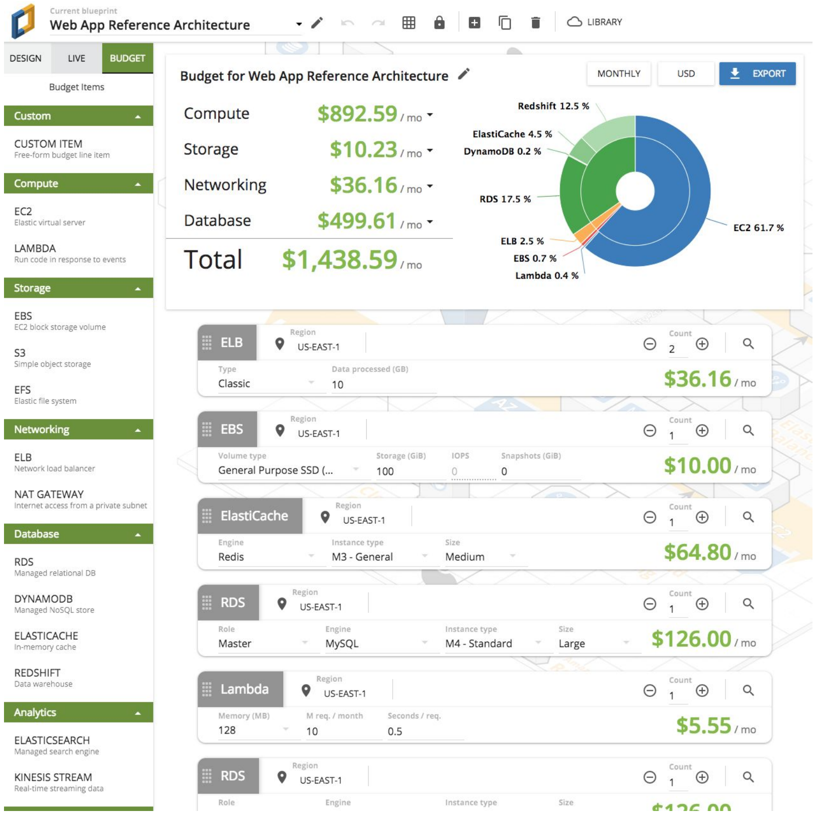
Amazon AWS Case Study

Cloudcraft allows you to...

1. Create an architecture diagram of various Amazon AWS building blocks & provides a budget dashboard.
2. Scan your AWS account, create architecture diagrams & populate a budget dashboard

You may also find Amazon Cost Explorer useful.





Pitfalls During Model Development

Pitfalls during Model Development: Choose Input Signals Carefully

Checklist Item	Things to keep in mind	What can you do
Is input signal reliable?	How does your model behave if input signal values are not available?	Instrument an indicator which specifies whether input signal is available or not.
Does the input signal change over time?	What happens when it does? How often?	Version it.
Is input signal necessary?	Does the cost of acquiring signal justify the value it gives to the model? How much effort does it take to maintain or transform this input signal during scoring?	Determine importance of input signal via model interpretability.
Input signal correlation	Is input signal causal or just correlated?	A common strategy on Kaggle is to create a new feature with random values, to see if this feature is correlated with the target.
Feedback Loops	Which of input signals may be impacted by models output?	Suppose one of your third party providers provides an input signal which is an output of their model.
Semantic Changes to Input Signals	If you rely on an external data source for an input signal, what would you do if how it was populated changed over time?	Flight carrier_name (i.e. Southwest) stays relatively constant, but carrier_id may change say from 1 to 10.
Avoid rarely used discrete feature values	Good feature values should appear atleast more than 5% or so times in a data set.	FlightNum is a bad feature because each value would be used only once, so the model couldn't learn anything from it:
Out of Vocabulary	Production data might contain new categorical input values which weren't included in training set.	Suppose a new carrier (not in training set) appears in production during scoring.
Is input signal available in production during time of prediction?	What is available during scoring? Is any data overwritten after scoring?	ArrDelay is only available after the flight takes place; thus not available during time of prediction.
Anomoiles (Outliers, Missing Data)	Real world data often contains outliers & missing data. What's our plan to deal with this?	Mean, median or a model can be used to impute missing values.

Pitfalls during Model Development: Data to Remove

Type of Issue	Things to keep in mind	Example
Bad labels	Investigate how target was created?	Think about scenarios when where the evaluation of IsDepDelayed changes over time.
Bad feature values	What to do when feature values are outside the feature range specified in the data dictionary?	Suppose AirTime is 10 mins due to a bug in the upstream process.
Duplicate examples	Does the real world use case produce duplicate values?	Suppose during data acquisition, several duplicate instances of flights from NY were provided for you as a starting point.
Omitted values	How will you handle missing data?	In several kaggle comptitions, it's often useful to create a new binary feature which can represent whether the value is omitted or not.
Leaky Features	Data or Knowledge Leakage	Most obvious form of leakage is when a variable in training dataset is derived from target.
Feature which are illegal to use	In many regulated industries (finance, health, transportation), you're limited by which input data you can train a model with.	In finance, if an organization denies an applicant credit, the organization must provide reason code(s) to the applicant.
Class Target Imbalance	Class target imbalance when one class in the target appears < 15% compared to the other target in the dataset.	Suppose you work for an e-commerce company with fraud rate of 2%. 98% of transactions are not fraud.
Variance Analysis	Some features don't change values.	All houses in Los Angeles are located in United States.

Airline On-Time Performance Data Dictionary

Column	Description	Type	Questions/Comments
Year	year of the flight	Integer	
Month	month of the flight	Integer	
DayofMonth	day of the month (1 to 31)	Integer	
DayOfWeek	day of the week	Integer	
DepTime	actual departure time	Float	Is this available 24 hours prior to departure (i.e. time of prediction)?
CRSDepTime	scheduled departure time	Integer	Is this available 24 hours prior to departure (i.e. time of prediction)?
ArrTime	actual arrival time	Float	Is this info available during time of prediction?
CRSArrTime	scheduled arrival time	Integer	Is this info available during time of prediction? How likely is it to change?
UniqueCarrier	carrier ID	String	Why would this matter?
FlightNum	flight number	Integer	How are flight numbers assigned?
TailNum	plane's tail number	String	How are tail numbers assigned & why would that matter? What happens if this plane is decomissioned?
ActualElapsedTime	actual elapsed time of the flight, in minutes	Float	Is this info available during time of prediction? What happens if we include this variable in the model?
CRSElapsedTime	scheduled elapsed time of the flight, in minutes	Float	Is this info available during time of prediction? How likely is it to change?
AirTime	airborne time for the flight, in minutes	Float	Is this info available during time of prediction?
ArrDelay	arrival delay, in minutes	Float	Is this info available during time of prediction?
DepDelay	departure delay, in minutes	Float	Is this info available during time of prediction?
Origin	originating airport	String	How likely is this to change?
Dest	destination airport	String	How likely is this to change?
Distance	flight distance	Float	How likely is this to change?
TaxiIn	taxi time from wheels down to arrival at the gate, in minutes	Float	Is this info available during time of prediction?
TaxiOut	taxi time from departure from the gate to wheels up, in minutes	Float	Is this info available during time of prediction?
Cancelled	cancellation status (stored as logical).	Integer	Should we bother predicting whether flight is delayed or not for a cancelled flight?
CancellationCode	cancellation code, if applicable	String	Should we bother predicting whether flight is delayed or not for a cancelled flight?
Diverted	diversion status	Integer	Is this info available during time of prediction?
CarrierDelay	delay, in minutes, attributable to the carrier	Float	
WeatherDelay	delay, in minutes, attributable to weather factors	Float	Weather predictions are available 24 hour in advance. Will you still include this variable if the model is expected run 48 hours instead of 24 hours in advance? How about if model expected to run 4 hours instead of 24 hours in advance?
NASDelay	delay, in minutes, attributable to the National Aviation System	Float	How far in advance do we know about national aviation delays? Consult domain expert.
SecurityDelay	delay, in minutes, attributable to security factors	Float	How far in advance do we know about security delays? Consult domain expert.
LateAircraftDelay	delay, in minutes, attributable to late-arriving aircraft	Float	How far in advance do we know about security delays? Consult domain expert.
IsArrDelayed	represents whether flight arrival was delayed or not	String	How was this generated? How is delayed define (in terms of mins)? Should you trust this?
IsDepDelayed	represents whether flight departure was delayed or not	String	How was this generated? How is delayed define (in terms of mins)? Should you trust this?

Offline vs Online Model Training

Offline vs. Online Model Training

A **Static Model** is…

+ Straightforward to build & evaluate
  + Iterate on train set until "it's good", use test set to assess performance, then evaluate on validation set
- Training model done offline (i.e. can take hours/days/weeks or months)
- If distribution of input changes & model hasn't adopted…whacky results
  - Requires monitoring during inference time
  - Model can easily grow stale

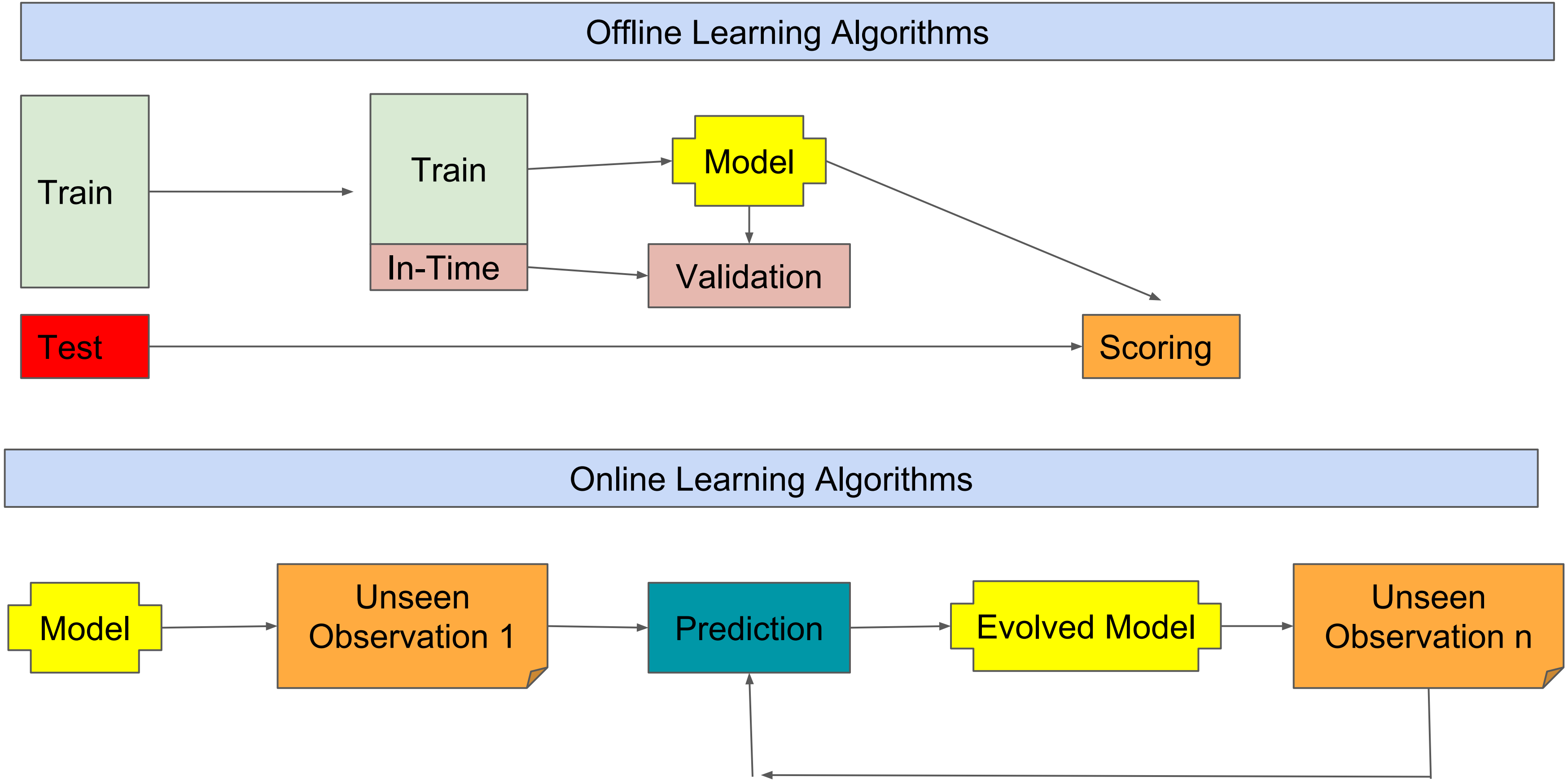
Best when **distribution** of **input data doesn't change** over time (i.e. images of dog breeds)

A **Dynamic Model** is…

+ Training model done online (i.e. continuously, soon as data arrives during inference)
+ Use progressive validation rather than batch train & test
+ Needs model update/rollback capability
+ Adapts to changes in input data, avoids staleness
- Needs monitoring of model outputs

Best when **distribution of input data likely to change** over time (i.e. contains trends, seasonalities)

Offline vs. Online Model Training



Dynamic Model Training: vowpal-wabbit

Vowpal-wabbit (vw) is a fast out-of-core learning system sponsored by Microsoft Research and (previously) Yahoo! Research.

1) **Input Format** for the learning algorithm is flexible

   [Label] [Importance] [Tag]|Namespace Features |Namespace Features … |Namespace Features

   ```
   1 1.0 |MetricFeatures:3.28 height:1.5 length:2.0 |Says stripes |OtherFeatures NumberOfLegs:4.0 HasStripes
   ```

2) **Speed**

   The learning algorithm is really fast (in-memory XGBoost is about an order of magnitude slower than vw)

3) **Scalability**

   The memory footprint of the program is bounded independent of data. Training set is not loaded into main memory before learning starts. The size of the set of features is bounded independent of the amount of training data using the hashing trick.

4) **Feature Pairing**

   Subsets of features can be internally paired so that the algorithm is linear in the cross-product of the subsets. This is useful for ranking problems.

Dynamic Model Training

csv2vw allows you to create a vowpal-wabbit formatted file from a CSV file.

Refer to example usage in help documentation.

If a target isn't numeric, it will be assumed to be a multi-class label and be converted to an integer [1..k]. In our example, a YES will be represented by a 1, and NO will be represented by 2 by default.

For Binary Logistic Regression, vowpal-wabbit requires all targets to be either 1 or -1.

So we must replace every occurrence of 2 by -1.

```
Examples:
    csv2vw -h -- -1 iris.csv
        Use 1st line as header, last column as label

    csv2vw 2 data.tsv
        Use 1..k as column/feature names, use 3rd column
        as the label column (base index is 0) - no header
        assumed in input
```

```
!csv2vw -h -- -1 ../data/processed/train.csv > ../data/processed/train.vw
```

```
!head ../data/processed/train.vw
```

```
1 1|f :17168 Year:1995 Month:1 DayofMonth:12 DayOfWeek:4 DepTime:1124.0 CRSDepTime:1111 ArrTime:1224.0 CRSArrTime:122
4 UniqueCarrier=US FlightNum:2451 TailNum=N920VJ ActualElapsedTime:60.0 CRSElapsedTime:73.0 AirTime:50.0 ArrDelay:0.0
DepDelay:13.0 Origin=RDU Dest=PHL Distance:336.0 TaxiIn:4.0 TaxiOut:6.0 Cancelled:0 CancellationCode= Diverted:0 Carr
ierDelay= WeatherDelay= NASDelay= SecurityDelay= LateAircraftDelay= IsArrDelayed=NO
1 2|f :42622 Year:2008 Month:1 DayofMonth:3 DayOfWeek:4 DepTime:1730.0 CRSDepTime:1620 ArrTime:2028.0 CRSArrTime:1920
UniqueCarrier=WN FlightNum:1484 TailNum=N242WN ActualElapsedTime:298.0 CRSElapsedTime:300.0 AirTime:276.0 ArrDelay:6
8.0 DepDelay:70.0 Origin=MCO Dest=PHX Distance:1848.0 TaxiIn:8.0 TaxiOut:14.0 Cancelled:0 CancellationCode= Diverted:
0 CarrierDelay:68.0 WeatherDelay:0.0 NASDelay:0.0 SecurityDelay:0.0 LateAircraftDelay:0.0 IsArrDelayed=YES
2 3|f :33789 Year:2003 Month:1 DayofMonth:6 DayOfWeek:1 DepTime:1227.0 CRSDepTime:1230 ArrTime:1736.0 CRSArrTime:1749
UniqueCarrier=UA FlightNum:1084 TailNum=N526UA ActualElapsedTime:189.0 CRSElapsedTime:199.0 AirTime:168.0 ArrDelay:-1
3.0 DepDelay:-3.0 Origin=DEN Dest=MCO Distance:1545.0 TaxiIn:4.0 TaxiOut:17.0 Cancelled:0 CancellationCode= Diverted:
0 CarrierDelay= WeatherDelay= NASDelay= SecurityDelay= LateAircraftDelay= IsArrDelayed=NO
```

```
!sed 's/^2 /-1 /g' ../data/processed/train.vw > ../data/processed/train_transformed.vw
```

```
!head ../data/processed/train_transformed.vw
```

```
1 1|f :17168 Year:1995 Month:1 DayofMonth:12 DayOfWeek:
4 UniqueCarrier=US FlightNum:2451 TailNum=N920VJ Actual
DepDelay:13.0 Origin=RDU Dest=PHL Distance:336.0 TaxiIr
ierDelay= WeatherDelay= NASDelay= SecurityDelay= LateAi
1 2|f :42622 Year:2008 Month:1 DayofMonth:3 DayOfWeek:4
UniqueCarrier=WN FlightNum:1484 TailNum=N242WN ActualEl
8.0 DepDelay:70.0 Origin=MCO Dest=PHX Distance:1848.0 T
0 CarrierDelay:68.0 WeatherDelay:0.0 NASDelay:0.0 Secur
-1 3|f :33789 Year:2003 Month:1 DayofMonth:6 DayOfWeek:
9 UniqueCarrier=UA FlightNum:1084 TailNum=N526UA Actual
-13.0 DepDelay:-3.0 Origin=DEN Dest=MCO Distance:1545.(
d:0 CarrierDelay= WeatherDelay= NASDelay= SecurityDelay
```

Dynamic Model Training

vowpal-wabbit

Resources:

Link to Command Line Arguments documentation

“--data” represents train data set

“--binary” represents loss as binary classification with -1,1 targets

“--loss_function” specifies loss function to be used.

“--readable_model” output human readable model.

“--kill_cache” don’t reuse cache, create new one

“--predictions” file to output predictions

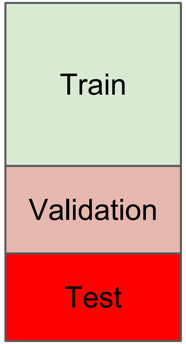
In [47]:
```
!vw \
  --data=../data/processed/train_transformed.vw \
  --binary \
  --loss_function=logistic \
  --readable_model=model.vw \
  --kill_cache \
  --predictions=predictions_train
```

```
predictions = predictions_train
Num weight bits = 18
learning rate = 0.5
initial_t = 0
power_t = 0.5
using no cache
Reading datafile = ../data/processed/train_transformed.vw
num sources = 1
average  since         example        example  current  current  current
loss     last          counter         weight    label  predict features
1.000000 1.000000            1            1.0   1.0000  -1.0000       29
0.500000 0.000000            2            2.0   1.0000   1.0000       26
0.500000 0.500000            4            4.0   1.0000   1.0000       28
0.375000 0.250000            8            8.0  -1.0000   1.0000       29
0.312500 0.250000           16           16.0   1.0000   1.0000       30
0.468750 0.625000           32           32.0  -1.0000   1.0000       25
0.406250 0.343750           64           64.0  -1.0000  -1.0000       29
0.375000 0.343750          128          128.0  -1.0000  -1.0000       30
0.312500 0.250000          256          256.0   1.0000   1.0000       30
0.285156 0.257812          512          512.0   1.0000   1.0000       27
0.269531 0.253906         1024         1024.0  -1.0000  -1.0000       29
0.269043 0.268555         2048         2048.0  -1.0000  -1.0000       30
0.250488 0.231934         4096         4096.0  -1.0000   1.0000       29
0.239746 0.229004         8192         8192.0  -1.0000  -1.0000       29
0.229614 0.219482        16384        16384.0   1.0000   1.0000       30

finished run
number of examples = 29465
weighted example sum = 29465.000000
weighted label sum = 1667.000000
average loss = 0.215646
best constant = 0.113272
best constant's loss = 0.691546
total feature number = 851319
```

Offline vs Online Model Inference

Offline Model Inference

Offline inference, meaning that you make all possible predictions in a batch, using a MapReduce or something similar. You then write the predictions to an SSTable or Bigtable, and then feed these to a cache/lookup table.



- Most often, split data into train, validation & test sets
  - **Train (60%)**: set of examples used for learning; fit parameters of classifier or regressor & perform cross validation
  - **Validation (20%)**: set of examples used to tune params of classifier or regressor & perform cross validation
  - **Test (20%)**: set of examples used only to assess performance of classifier or regressor. Cardinal sin if you tune the model after observing results on test set
  - Remember not to skip Test phase, this is a preview of how the model will perform in the wild (i.e. out of time) examples
- You can often leverage code & cross validation strategy from model build steps
- Model query (single row, multiple rows) via data loaded in memory (RAM) from file/database

```
X_train.fillna(missing, inplace=True)
X_valid.fillna(missing, inplace=True)
test_transformed.fillna(missing, inplace=True)
dtrain = xgb.DMatrix(X_train, label=y_train)
dvalid = xgb.DMatrix(X_valid, label=y_valid)

watchlist = [(dtrain, 'train'), (dvalid, 'eval')]
params = {
    "objective": "binary:logistic",
    "booster" : "gbtree",
    "eval_metric": "auc",
    "eta": .01,
    'nround': 1000,
    "tree_method": 'exact',
    "max_depth": 5,
    "subsample": 0.5,
    'min_child_weight': 1,
    "silent": 1,
    "seed": random_state }
print("XGBoost Params:                    " + json.dumps(params))
num_boost_round = 5
early_stopping_rounds = 20

gbm = xgb.train(params,
                dtrain,
                num_boost_round,
                evals=watchlist,
                early_stopping_rounds=early_stopping_rounds,
                verbose_eval=True)
# explain the model's predictions using SHAP values
# (same syntax works for LightGBM, CatBoost, and scikit-learn models)
explainer = shap.TreeExplainer(gbm)
shap_values = explainer.shap_values(X_train)

print("\nFeature Importances...")
importances = gbm.get_fscore()

validation_set_predictions = gbm.predict(xgb.DMatrix(X_valid), ntree_limit=gbm.best_iteration+1)
log_metrics(y_valid, validation_set_predictions, "Validation", {"params":params, "importances": importances})
```

```
def production_predict(model, unseen_data, train_features_encoded):
    unseen_data = filter_training_data(unseen_data, raw_features_not_to_use)
    unseen_data = categorical_encode(unseen_data, train_features_encoded, raw_features_to_encode)

    predictions = model.predict(xgb.DMatrix(unseen_data))
    return predictions

predictions = production_predict(model, test, train_features_encoded)
predictions
```

```
Get Training Data...
Original shape: (14513, 31)
After columns dropped shape: (14513, 11)

Total number of features before encoding: 11

Total number of features after encoding: 282

array([0.50005496, 0.5119886 , 0.5092711 , ..., 0.51288676, 0.5088346 ,
       0.49939823], dtype=float32)
```

Online Model Inference

Online inference, meaning that you predict on demand (single or multi-rows), using a server.

- Clipper cluster creation
- App creation & model deployment
- Model query (single row, multiple rows) via Python requests & curl
- Model versioning update
- Model versioning rollback + Model replication

Useful Resources:

“What’s your ML Test Score” paper from Google

Clipper Model Deployers support for:

- PySpark Models
- PyTorch Models
- Tensorflow Models
- MXNet Models
- XGBoost Models
- Pure Python Functions

Query Model (single row)

```
import requests, json, numpy as np
print("Model predict for a single instance via Python requests POST request...")
headers = {"Content-type": "application/json"}
requests.post("http://localhost:1337/xgboost-airlines/predict", headers=headers,
              data=json.dumps({"input": get_test_point(0)})).json()
```

```
Model predict for a single instance via Python requests POST request...

{'query_id': 5, 'output': 0.9041093, 'default': False}
```

Model Replication

Machine learning models can be computationally expensive. A single instance of the model hosting machine may not meet the throughput requirements of a serving workload. In order to increase the prediction throughput you can add additional replicas...

```
clipper_conn.set_num_replicas('xgboost-model', num_replicas=10, version='1')
```

```
18-08-22:00:33:13 INFO     [docker_container_manager.py:353] [default-cluster] Found 1 replicas for xgboost-model:1.
Adding 9
```

Model Monitoring in Production

Monitoring ML model in production

Requirements:

Input data distribution shift

Model score distribution

When model needs to be re-trained?

Feedback loops (adversarial attacks)?

Solution:

- Clipper (paper, implementation, presentation)
- Amazon SageMaker & CloudWatch monitoring model performance (tutorial)

Monitoring ML Model in Production

Features of clipper:

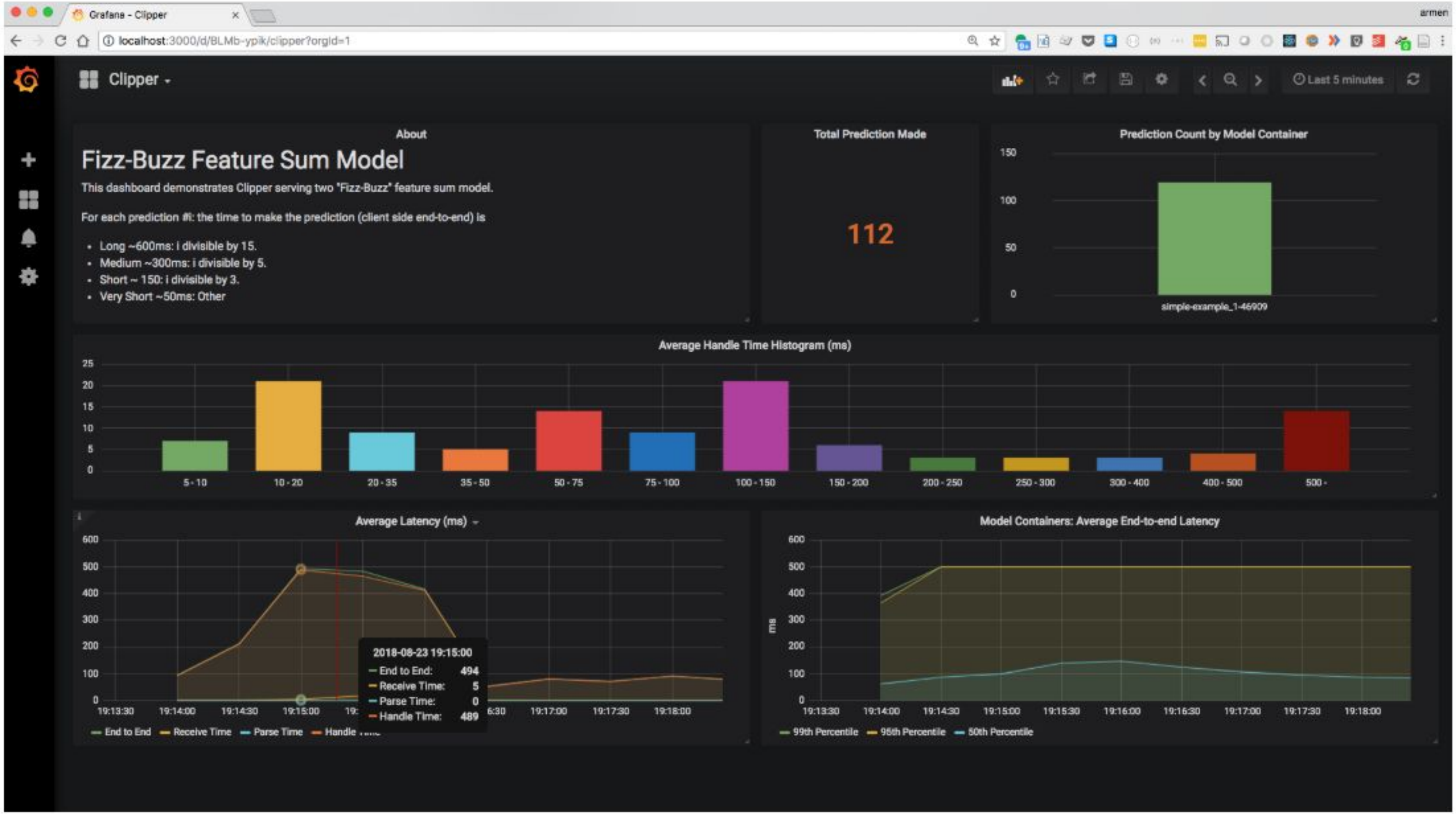
- clipper uses grafana as the metric tracking system
- clipper provides latency & throughput metrics by default
- You can find demo here

User Defined Metrics API

```
import clipper_admin.metric as metric

metric.add(metric_name, metric_type, metric_description,
optional_histogram_bucket)

metric.report(metric_name, metric_data)
```

```python
def predict_spam(inp):
    metrics.add_metric('custom_vectorization_time_ms', 'Histogram',
                       'Time it takes to use tfidf transform',
                       [0.1, 0.5, 0.8, 1.0, 1.2])
    metrics.add_metric('custom_lr_time_ms', 'Histogram',
                       'Time it takes to use logistic regression',
                       [0.03, 0.05, 0.06, 0.1])
    metrics.add_metric('custom_choice_probability', 'Histogram',
                       'The logistic regressor probability output',
                       [0.5, 0.7, 0.9, 1.0])
    metrics.add_metric('custom_spam_option_counter', 'Counter',
                       'The number of spam classified')
    metrics.add_metric('custom_ham_option_counter', 'Counter',
                       'The number of ham classfied')
    metrics.add_metric('custom_char_count', 'Histogram',
                       'The number of characters',
                       [10, 50, 100, 300, 500, 800, 1200, 2000])
    metrics.add_metric('custom_word_count', 'Histogram', 'The number of words',
                       [10, 50, 100, 150, 200])

    string = inp[0]
    metrics.report_metric('custom_char_count', len(string))
    metrics.report_metric('custom_word_count', len(string.split()))
```

Monitoring ML model in production



Backup Slides