Automated Machine Learning in Biomedicine: AutoMLPipe-BC

Ryan J. Urbanowicz, PhD @docurbs

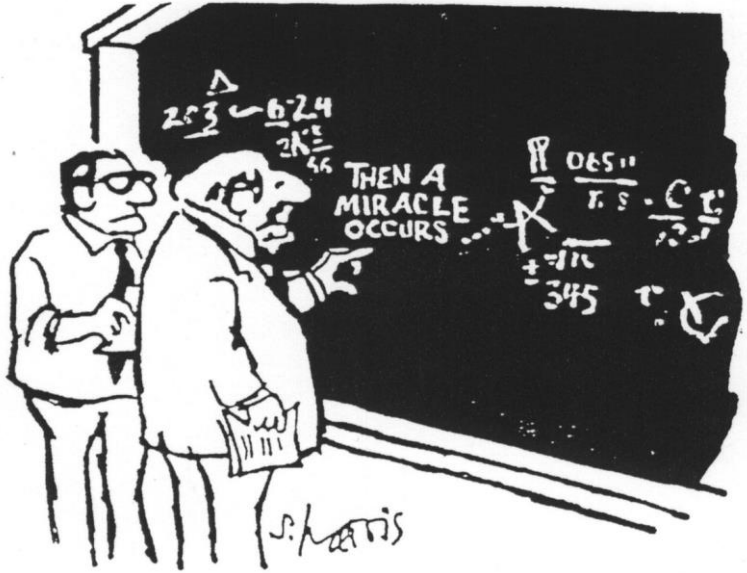
Research Scientist II (Assistant Professor Title Pending)
Computational Biomedicine Department, Cedars-Sinai Medical Center

Cedars Sinai

cedars-sinai.org

Outline

- Machine Learning (ML) Basics
- Biomedical Data: Goals and Challenges
- Elements of a ML Analysis Pipeline
  - Data Preparation
  - ML Modeling
  - Evaluation
  - Post-Analysis
- AutoML: Making Things Easier
- AutoMLPipe-BC Demonstration



I think you should be a little more specific, here in Step 2

https://twitter.com/irispenagos/status/565379514310885379

Machine Learning (ML) Basics

ML Essential Terminology

Instance: an example or single occurrence of something

- a.k.a. individual, sample, subject, or patient

Feature: a measurable piece of data describing some aspect of an instance

- a.k.a. variable, independent variable, or attribute

Outcome: an event or metric that can be observed or measured

- a.k.a. endpoint, dependent variable, class/label (in classification)

Model: a representation or simulation of reality (based on assumptions)

- ML: an expression of an algorithm that represents patterns and can be used to make predictions using data

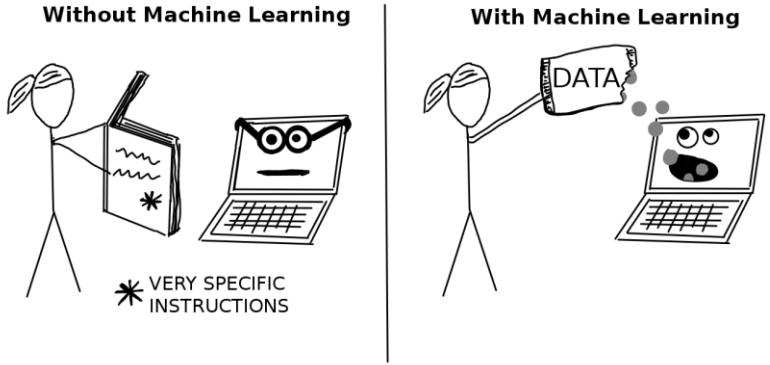
Algorithm: a set of instructions (calculations or problem-solving operations) for completing a task

What is Machine Learning (ML)?

A subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" with data, without being explicitly programmed

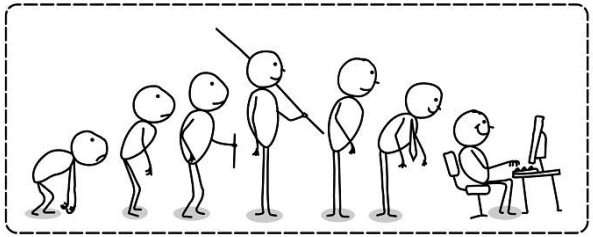
Samuel Arthur – 1959 – ML in Checkers

Learn: progressively improve performance on a specific task



Why Machine Learning?

- **In order to make decisions…**
  - Automated
  - Optimize performance beyond human capabilities
  - With greater speed
  - Without human bias (assuming that bias is absent from the data)
  - Better adapting to new data

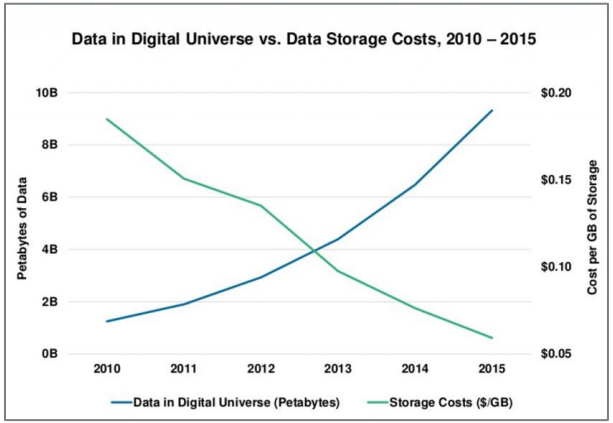


- **Particularly well suited to tasks involving:**
  - Big data – very large numbers of features and/or instances
  - Complex data – underlying patterns involve many features with subtle, interacting, and heterogeneous relationships

https://www.dreamstime.com/stock-images-computer-user-evolution-image20168724?fbclid=IwAR2aYTbwBR2l7rwDmIwz-9dCBzcDqNztEtTxqLIWEApR-Av-zL5QVjUDIAY

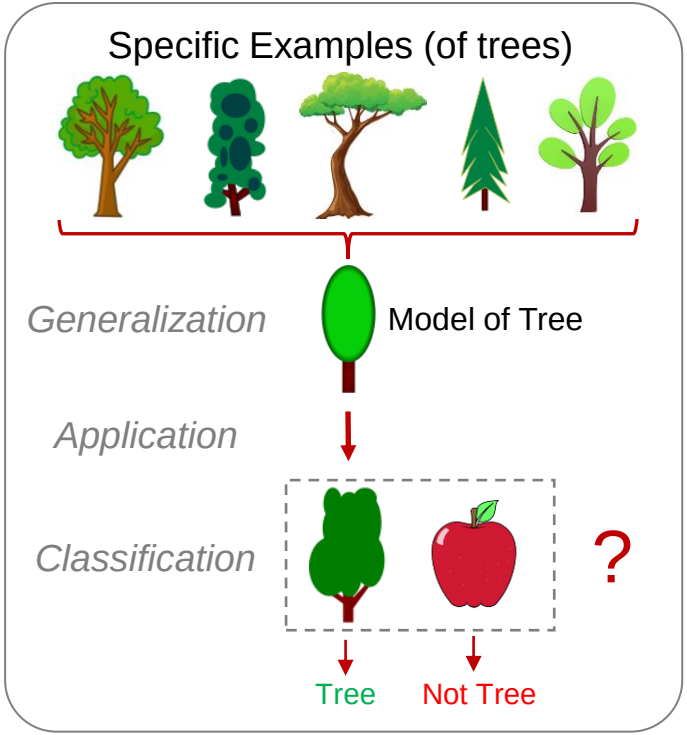
Why now?

- Flood of available data
- Increasing computational power and availability (e.g. cloud computing)
- Ongoing progress in available algorithms and theory
- Increased support and resources through industry



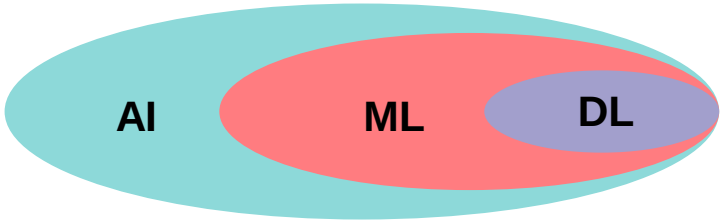
Big Picture Goals of ML

- **Discover patterns / associations within data**
- **Represent those patterns as useful generalizations**
  - e.g. Models
  - Generalization Process:
    - Specific examples simplified to one ‘best fit’ idea
- **Apply those generalizations to decision making**
  - e.g. Classification / Prediction



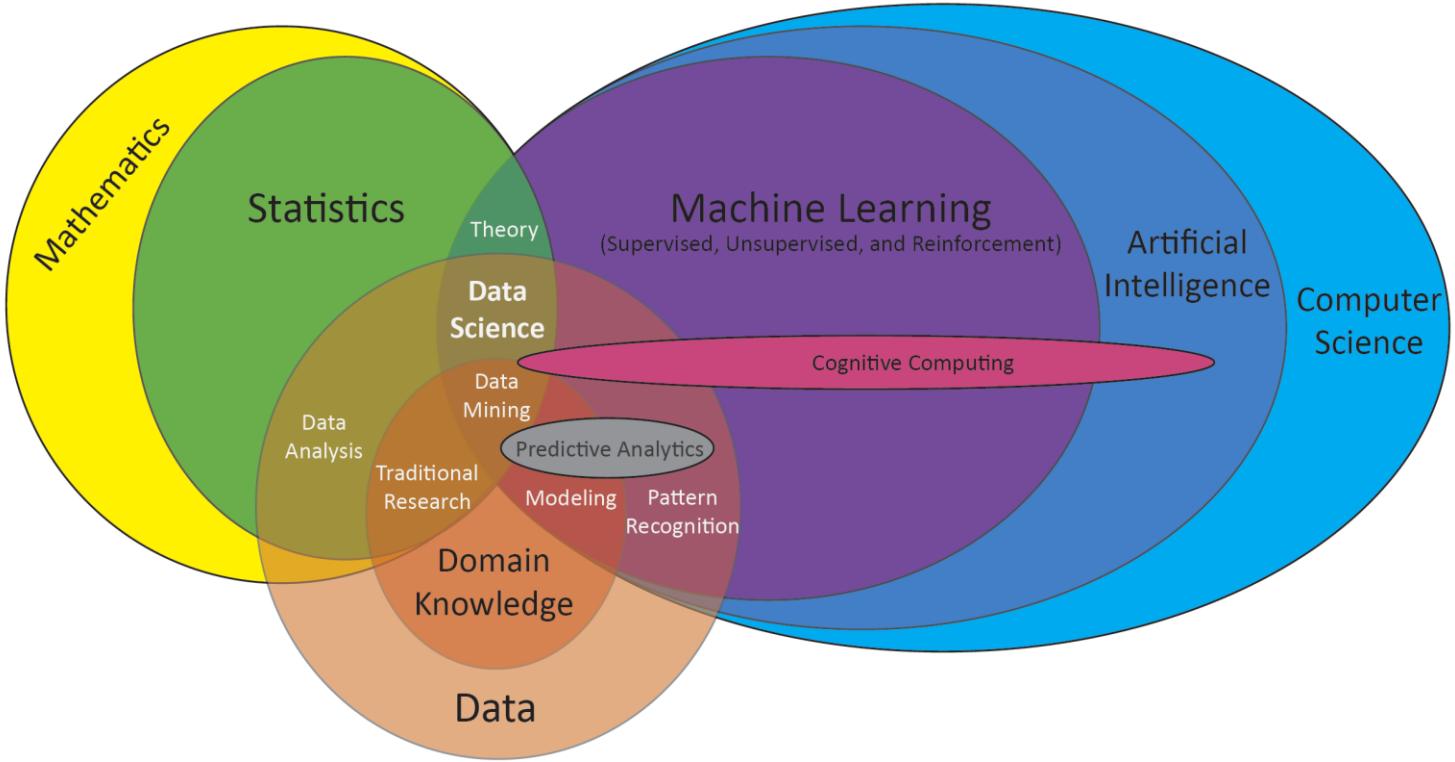
Addressing Possible ML Misconceptions

- **ML ≠ artificial intelligence (AI) ≠ deep learning (DL)**

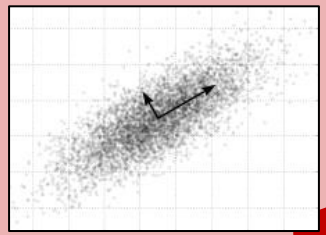


- **ML not designed to demonstrate causality**
  - At best, found associations are candidates for causality
- **Getting more data (i.e. features and instances) does not ensure a better model trained**
  - Increased data quality and making correct modeling assumptions be of greater value
  - > features can make the ML modeling more difficult (i.e. a larger search space)
  - > instances can increase data heterogeneity requiring more complex models to detect
- **ML can not be applied to solve any problem**

Fields and Terms Related to Machine Learning



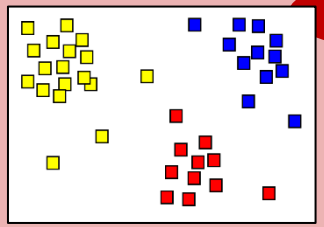
ML Methods Categorized by Learning Style



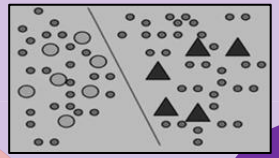
Dimensionality Reduction

Unsupervised Learning

Unlabeled Data



Clustering



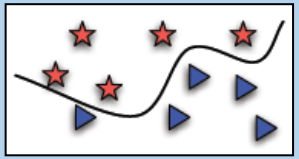
Partially Labeled Data

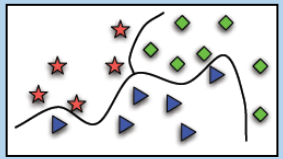
Semi-Supervised Learning

Machine Learning

Reinforcement Learning

No Data but Environment to Interact With (Multi-Step Problems)



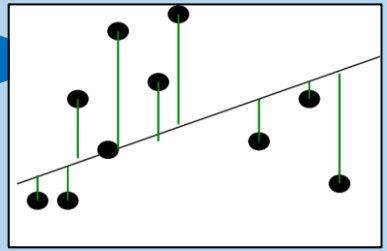


Classification

Supervised Learning

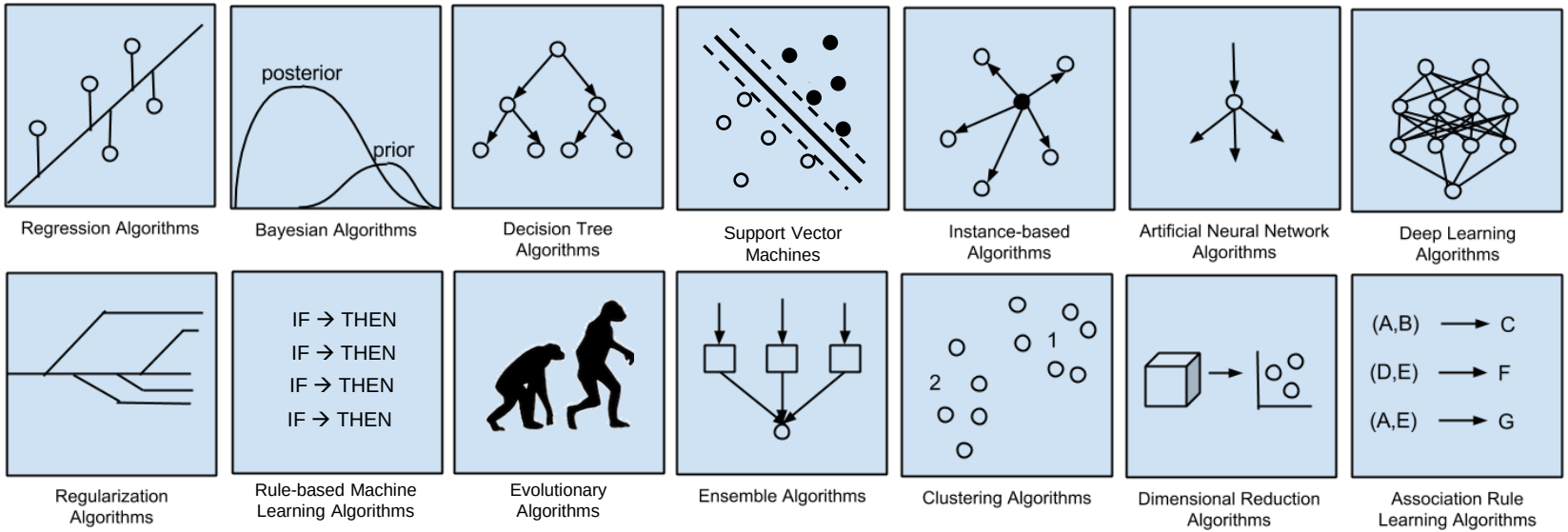
Labeled Data

Regression



Machine Learning: A General Term for a Variety of Algorithms

- Differ by knowledge representation, search strategy, learning strategy, and assumptions
- Each algorithm has different implementation possibilities



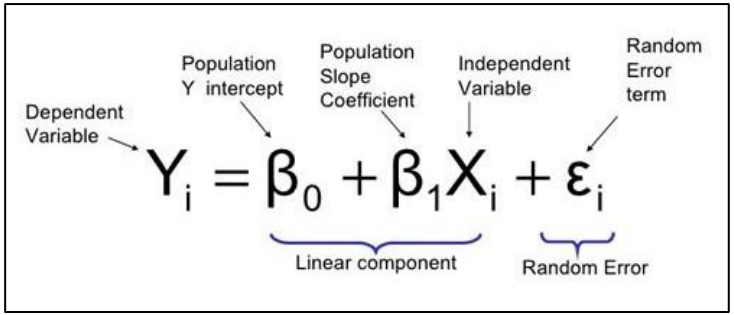
Regression Algorithms | Bayesian Algorithms | Decision Tree Algorithms | Support Vector Machines | Instance-based Algorithms | Artificial Neural Network Algorithms | Deep Learning Algorithms

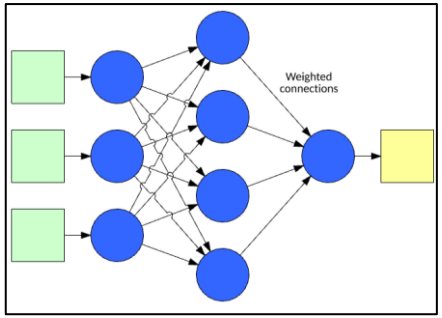
Regularization Algorithms | Rule-based Machine Learning Algorithms | Evolutionary Algorithms | Ensemble Algorithms | Clustering Algorithms | Dimensional Reduction Algorithms | Association Rule Learning Algorithms

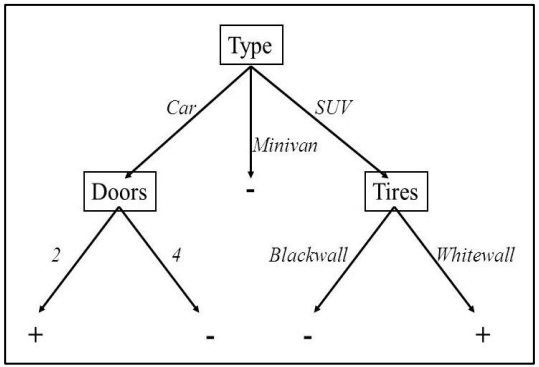
Cedars Sinai

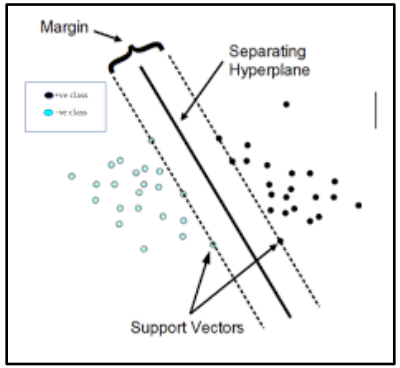
Note: This is not a comprehensive list of ML Algorithm types.
Adapted from: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

One of the Biggest ML Algorithm Differences: Model Representation



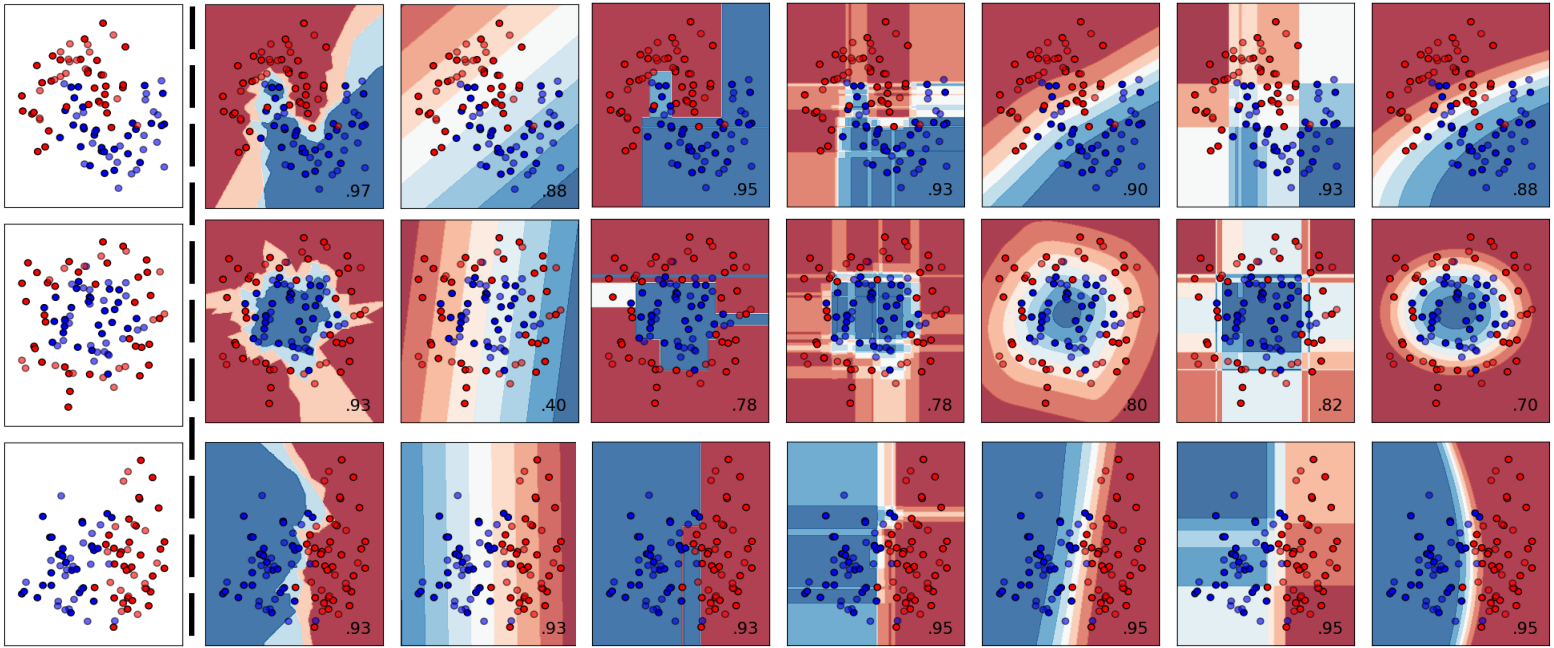






- R1: **IF** the animal has hair **THEN** it is a mammal
- R2: **IF** the animal gives milk **THEN** it is a mammal
- R3: **IF** the animal has feathers **THEN** it is a bird
- R4: **IF** the animal flies the animal lays eggs **THEN** it is a bird
- R5: **IF** the animal is a mammal the animal eats meat **THEN** it is a carnivore

Classification (Decision) Boundaries Using Various ML Algorithms



Adapted from: https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Biomedical Data: Goals and Challenges

Raw Biomedical Data

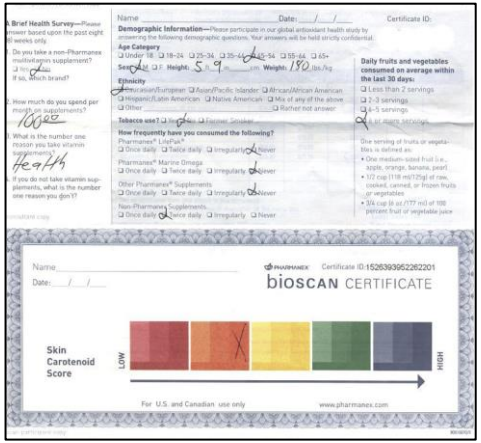
```
Exam Date: 04/05/2012   10:27  ORD #90003 Accession #9672187
History Number: 1636284
Age: 65Y  Sex: M Race: W
Requester: ANDREW COSGAREA


RESULT:
Bilateral knees, history of right knee pain. Four views right and 3 views
left. Bilateral 3 compartment arthritis. Small osteophytes femoral notch
on flexion view bilaterally. Bilateral patellofemoral arthritis left
greater than right.
Moderate to marked Prepatellar soft tissue swelling right knee,
suggesting bursitis. No evidence of right joint effusion.
```

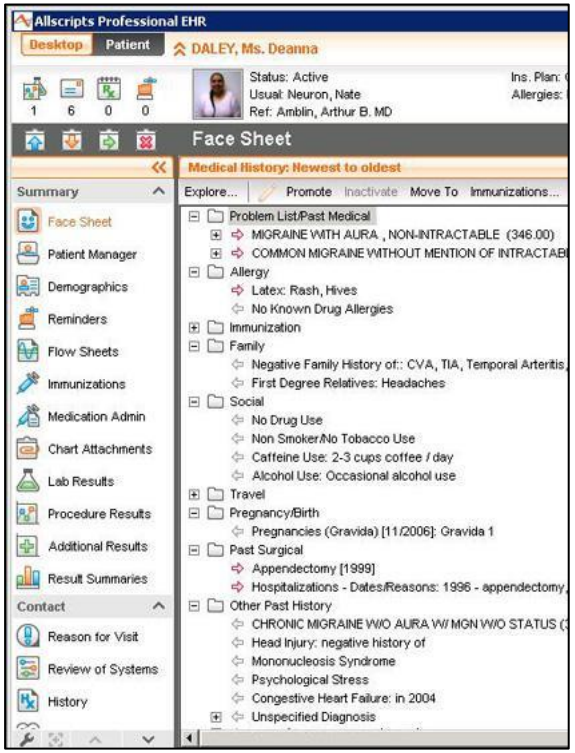
COMPLETE BLOOD COUNT (CBC) WITHOUT

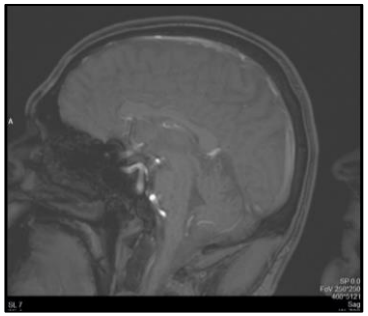
Component	Value
White Blood Cell Count	6040
Red Blood Cell Count	4.72
Hemoglobin	13.9
Hematocrit	40.0
Mean Corpuscular Volume	84.7
Mean Corpus HgB	29.4
Mean Corpus HgB Conc	34.8
RBC Distribution Width	12.7
Platelet Count	254
Mean Platelet Volume	10.1
Nucleated RBC Number	0

Narrative

```
Johns Hopkins Medical Labs
Meyer B-171
600 North Wolfe Street
Baltimore, MD. 21287
```

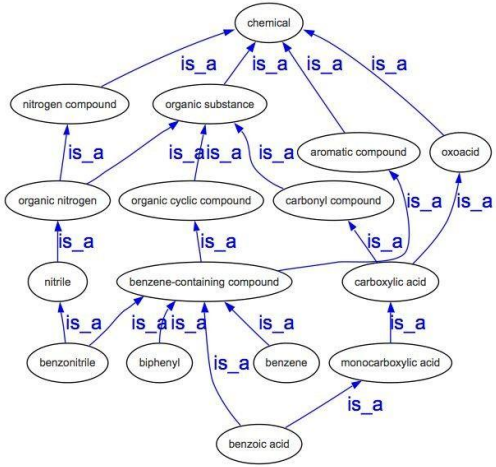


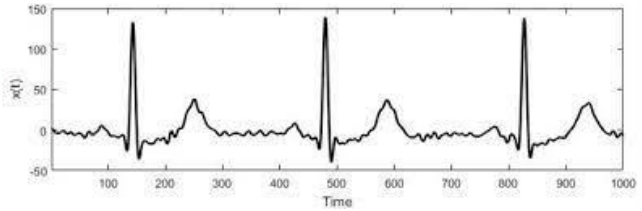


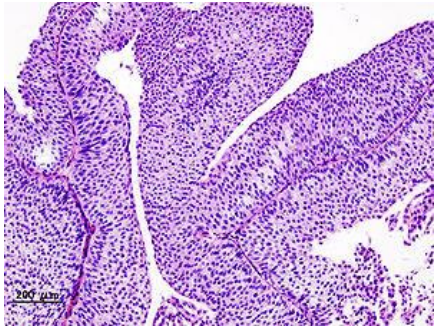


Common Data Types

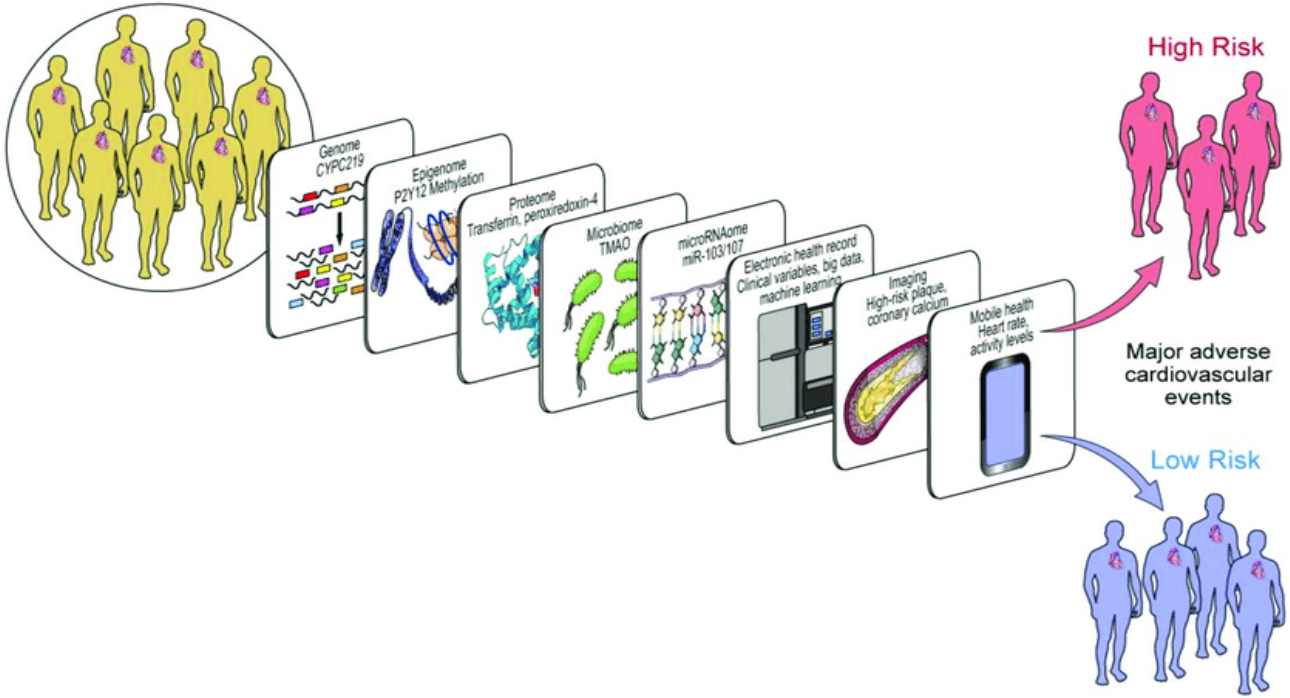
Date collected	Plot	Species	Sex	Weight
1/9/78	1	DM	M	40
1/9/78	1	DM	F	36
1/9/78	1	DS	F	135
1/20/78	1	DM	F	39
1/20/78	2	DM	M	43
1/20/78	2	DS	F	144
3/13/78	2	DM	F	51
3/13/78	2	DM	F	44
3/13/78	2	DS	F	146





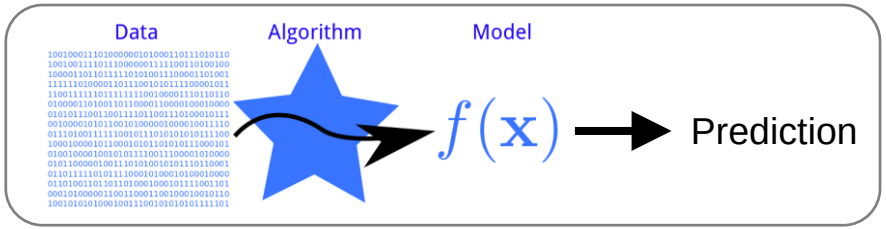


Biomedical Big Data



Goals: ML Analysis with Biomedical/Clinical Data

- **Identify risk factors/biomarkers of health outcomes**
  - Etiology
  - Diagnosis, Outcome Prediction
  - Personalized Medicine
    - Preventative Medicine
    - Drug/Treatment Selection
- **Develop prediction models for deployment**
- **Improve clinical efficiency**
  - Speed and accuracy of diagnosis/treatment
  - Cost

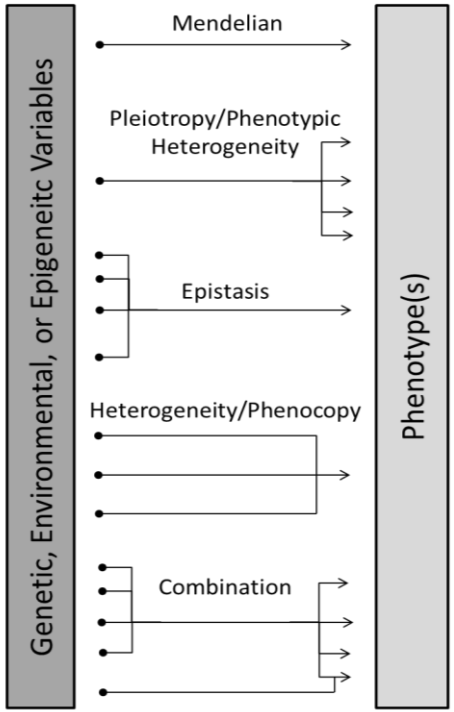


https://medium.datadriveninvestor.com/difference-between-an-machine-learning-algorithm-and-model-14879f4aec7b

Common Challenges in Biomedical Data Analysis

- **Cleaning / Data Processing**
  - Missing data values
  - Combining variable types
  - Feature engineering
- **Analysis**
  - Imbalanced outcome (classification)
  - Covariate adjustment
- **Computing**
  - Scale of data (# features, # instances)
- **Signal Detection**
  - Noise
  - Missing predictive variables
  - Rare variation
  - Complex multivariate associations
- **Reliability**
  - Interpretability / Explainability
  - Reproducibility
  - Identify and account for sources of bias

Complex Patterns of Association (From Genetics Perspective)

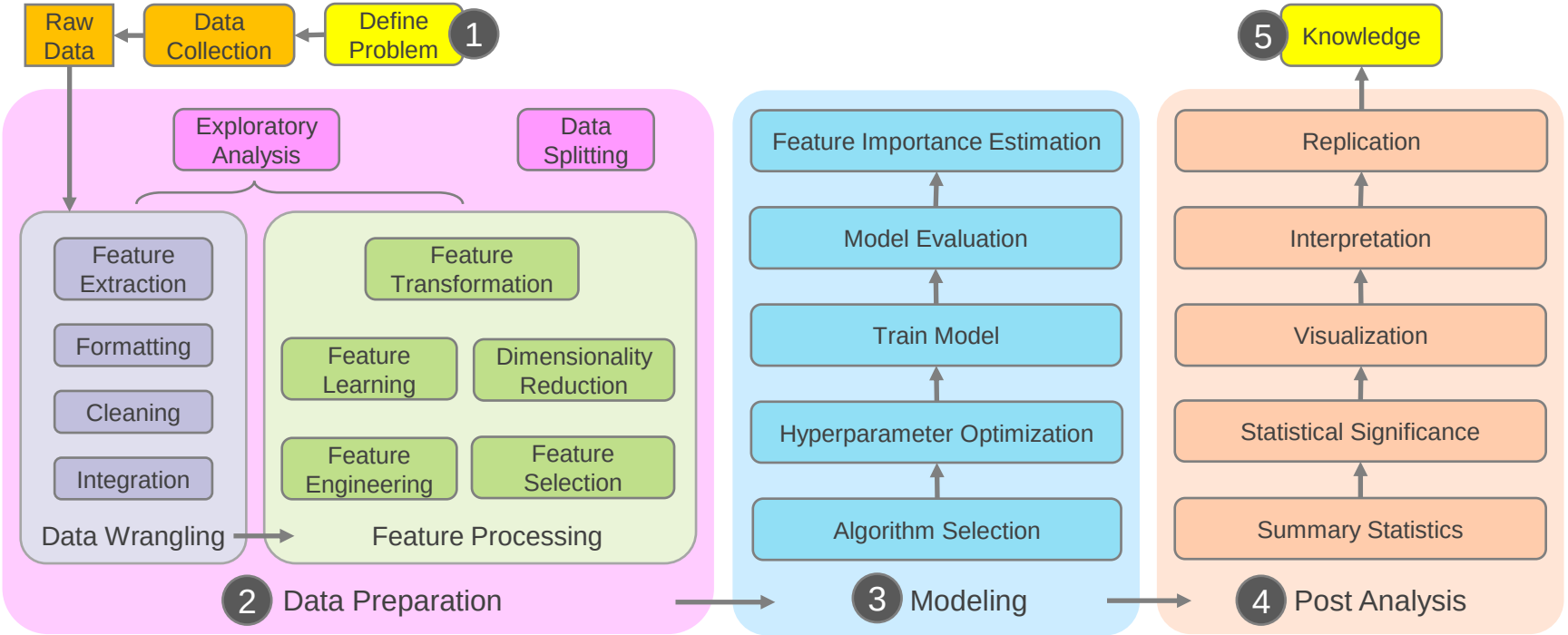


- **Univariate Association:** Association between single gene and phenotype
- **Pleiotropy:** effect of a single gene on two or more seemingly unrelated phenotypes/traits
- **Epistasis (multivariate interaction):** a gene that controls or masks the phenotypic expression of a different gene
- **Heterogeneous Associations:**
  - **Phenocopy:** variation in phenotype caused by environmental conditions similar to a phenotype determined by genetic factors
  - **Genetic Heterogeneity:** production of same or similar phenotypes through different genetic mechanisms

Figure by Ryan Urbanowicz

Elements of an ML Analysis Pipeline

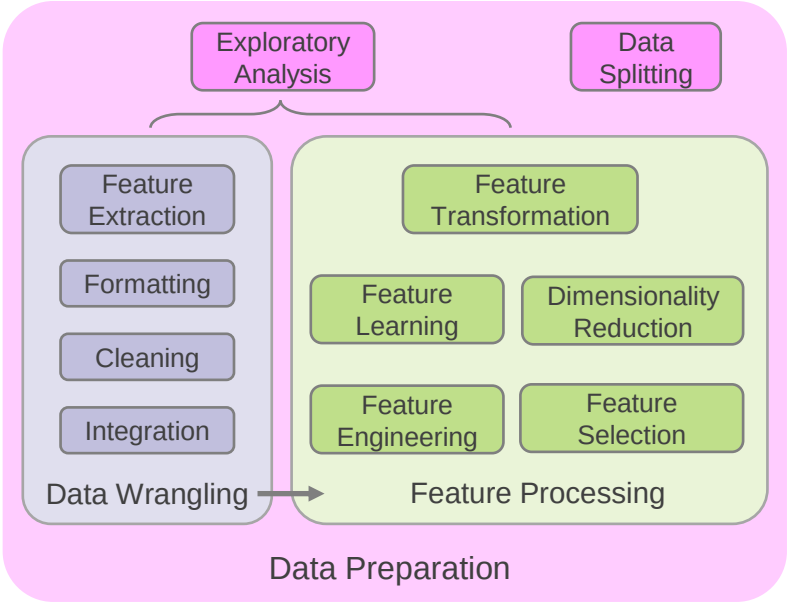
Elements of Data Mining



Data Preparation

Purpose:

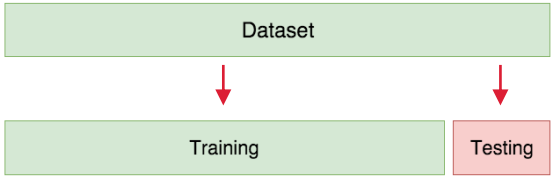
- Get oriented with available data
- Getting data into a format ML can work with
- Identify and remove, errors and bias
- Provide ML with the smallest set of relevant, or maximally informative features possible



Data Preparation: Data Splitting

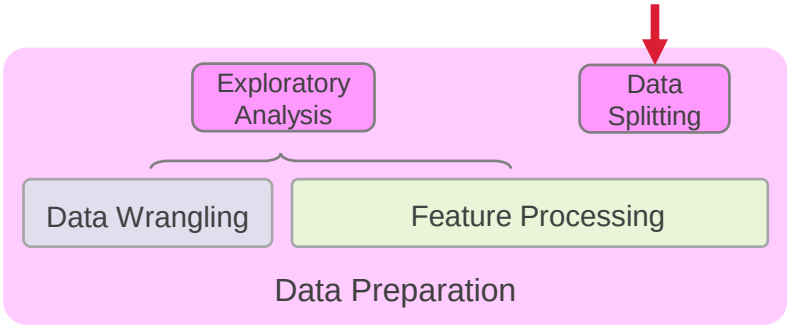
Partitioning available data into at least two parts:

1. Training Data: instances used to train a predictive model
2. Testing Data: instances used to evaluate model performance (generalization to unseen data)



When? Before any step where there is potential to learn from instances that will become testing data

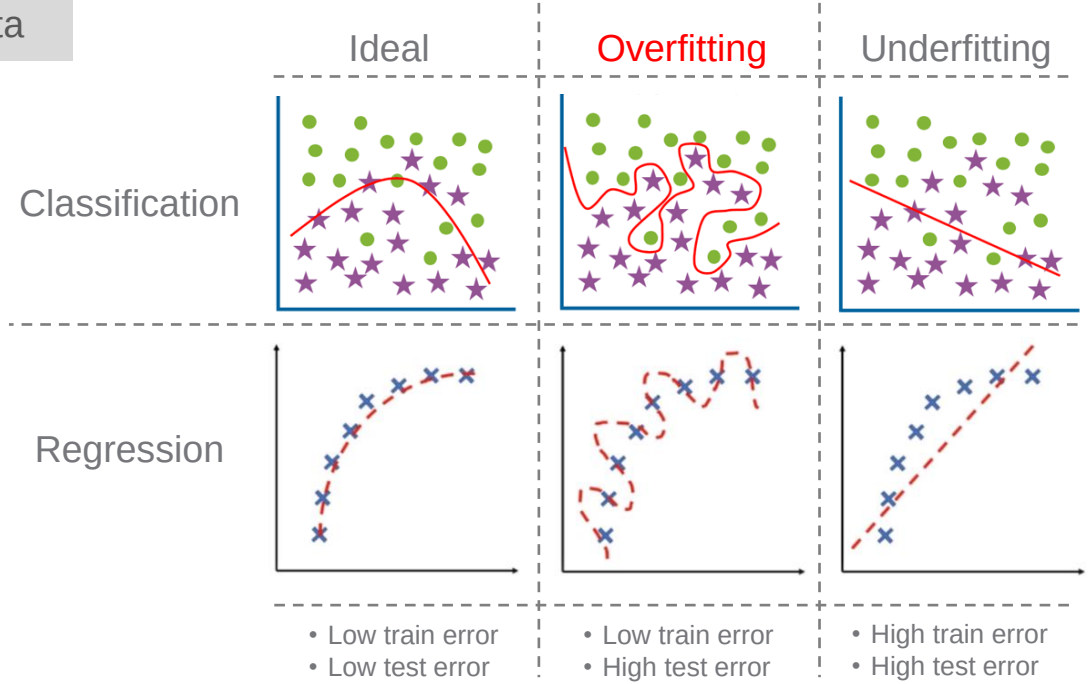
- Modeling
- Feature processing (-) feature engineering
- Imputation



https://machinelearningmastery.com/difference-test-validation-datasets/

ML Pitfall: Overfitting

Fitting a model exactly to the training data



Adapted from: https://medium.com/geekculture/investigating-underfitting-and-overfitting-70382835e45c
https://twitter.com/jason_mayes/status/1296599149748551680

Data Preparation: Data Splitting → k-fold Cross Validation

Resampling procedure used to evaluate ML models on a limited data sample

Why?

- Addresses risk of sample bias in data splitting

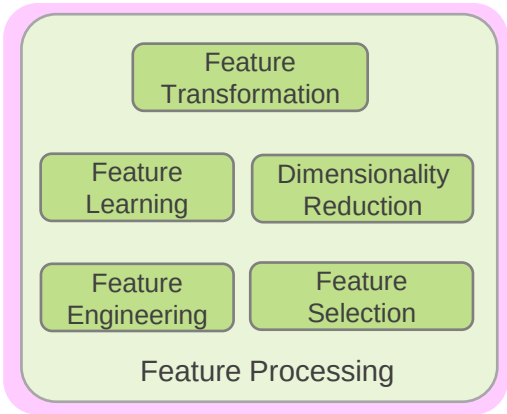
How?

1. Shuffle data randomly
2. Split data into k 'folds'
3. For each fold:
   - Take current fold as test data
   - Take all other folds as training data
   - Fit a model on the training data and evaluate on test
4. Summarize algorithm skill on data via k-fold average

Data Preparation: Feature Processing

Improving the data representation so that ML modeling can achieve higher performance

- **Feature Engineering:** Using domain knowledge to build new features from existing ones
- **Feature Learning:** Automatically use data to build new features from existing ones
- **Feature Transformation:** Change the scale or distribution of features
- **Dimensionality Reduction:** Create compact projections of features
- **Feature Selection:** Remove irrelevant or highly correlated features



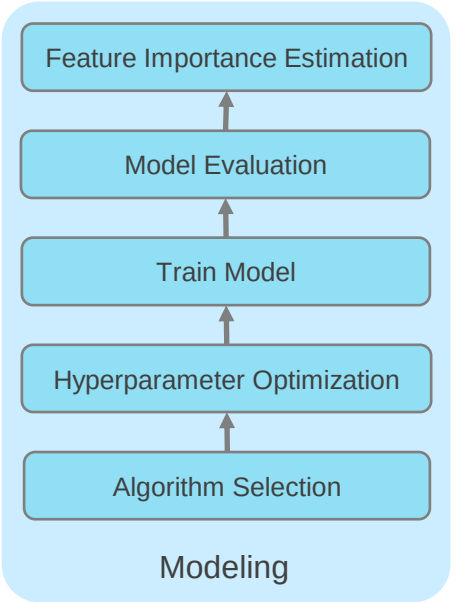
ML Modeling

The process of training an ML model with an ML training algorithm

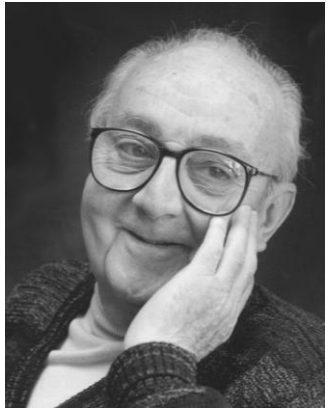
- The ML model = the artifact created by the algorithm training process

Can/should involve:

- Selection of Algorithm(s)
- Setting and/or optimizing algorithm hyperparameter settings
- Training a model with selected hyperparameters (learning from data)
- Evaluating performance of trained model using hold-out data (test data)
- Estimate the importance of features in the model (part of interpretation)



“All models are wrong, but some models are useful” – George Box



A model is a simplification of reality →

Simplifications are based on assumptions (model bias) →

Assumptions fail in certain situations →

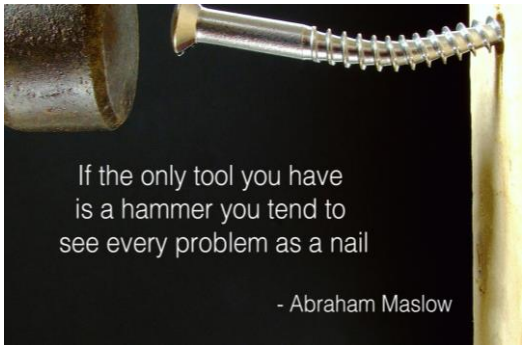
A model will never represent the exact real behavior

- However, a ‘close-enough’ model can be useful to describe the behavior in practice
- Each algorithm (and the models it generates) makes unique assumptions
  - With a uniquely biased view of the problem

The “No Free Lunch” (NFL) Theorem

“Any two optimization algorithms are equivalent when their performance is averaged across all possible problems”

- No single algorithm can perform optimally across all problems
- Doesn’t mean: Algorithm choice doesn’t matter
- Does mean: You shouldn’t always (only) use the same algorithm for every problem
- Also applies to hyperparameter selection within a given algorithm
  - Don’t always use the same/default settings



https://twitter.com/sterlingrockuk/status/499145957150580737

How to Deal with “No Free Lunch”?

1. **For any algorithm, test and optimize hyperparameter settings**
2. **Try to chose the most appropriate single algorithm:**
   - Choice based on many factors: data type, computational resources, assumptions…
   - Easiest, quickest, but riskiest
3. **Apply, compare, and pick the best performing of multiple algorithms:**
   - Pick algorithms with different strengths and weaknesses to get a diversity of perspectives
   - 3 or more algorithms suggested (or as many as you have the time/resources to apply)
4. **Utilize “Wisdom of the Crowd” with Ensemble Algorithms or Meta-Algorithms:**
   - Combining multiple models in the prediction process
   - Least interpretable

ML Modeling: Supervised Machine Learning Algorithms

(Designed for or that can be adapted to supervised ML modeling)

Regression

- Logistic
- Linear
- Polynomial
- Stepwise
- Cox
- Ordinal
- Poisson
- Quantile
- Ordinary Least Squares
- Partial Least Squares
- Multiple Linear
- Generalized Linear Model
- Locally Estimated Scatterplot Smoothing
- Multivariate Adaptive Regression Splines (MARS)

Discriminant Analysis

Modeling + Dimensionality Reduction

- Linear
- Symbolic
- Mixture
- Quadratic
- Regularized
- Flexible
- Partial Least Squares

Bayesian

- Naïve Bayes (NB)
- Gaussian (GNB)
- Multinomial (MNB)
- Bayesian [Belief] Network (BBN)
- Averaged One-Dependence Estimators (AODE)

Regularized Regression

Modeling + Feature Selection

- Least Absolute Shrinkage and Selection Operator (LASSO)
- Elastic Net Regression
- Least Angle Regression (LARS)
- Ridge Regression

Neural Networks

- Perceptron
- Multilayer Perceptron
- Hopfield Network
- Radial Basis Function Network

Decision Tree

- Classification and Regression Trees (CART)
- Iterative Dichotomizer 3 (ID3)
- Chi Squared Automatic Interaction Detection (CHAID)
- Conditional Decision Trees
- C4.5
- C5
- M5

Ensemble Learning

- Random Forest (RF)
- Bagging Decision Forest (DF)
- Gradient Boosting Tree (GBT)
- CatBoost
- AdaBoost
- Extreme Gradient Boosting (XGB)
- Light Gradient Boosting (LGB)
- Decision Jungle (DJ)

Deep Learning

- Deep Multilayer Perceptron
- Recursive/Recurrent Neural Network (RNN)
- Convolutional Neural Network (CNN)
- Deep Belief Networks (DBN)

Instance-Based

- k-Nearest Neighbors (k-NN)
- Locally Weighted Learning (LWL)
- Learning Vector Quantization (LVQ)

Support Vector Machines (SVM)

- Linear SVM
- Kernel SVM

Rule-Based

- Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
- Learning Classifier System (LCS)
- Association Rule Mining
- One Rule (OneR)
- Cubist
- Apriori

Other Algorithms

- Genetic Algorithm
- Genetic Programming

Cedars Sinai

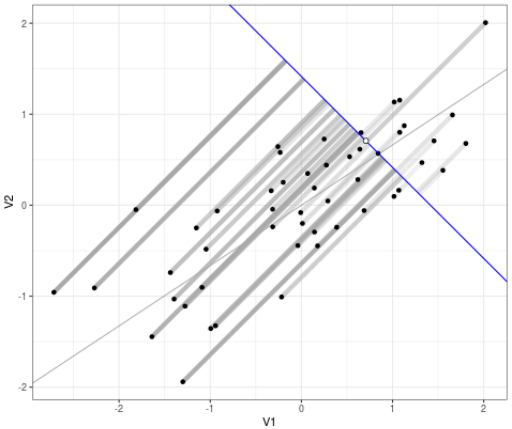
Non-exhaustive, and fuzzily organized listing

Adapted from: https://www.kaggle.com/getting-started/153090 https://www.theinsaneapp.com/2021/02/types-of-machine-learning-algorithms.html

What is Regression?

Two potential meanings:

1. **Statistics:** modeling method that attempts to determine the strength and character of the relationship between one dependent variable and a series of other variables
   - e.g. Linear regression: minimize the sum of the squared residuals
2. **Machine learning:** modeling to predict a continuous-valued outcome
   - i.e. classification vs. regression



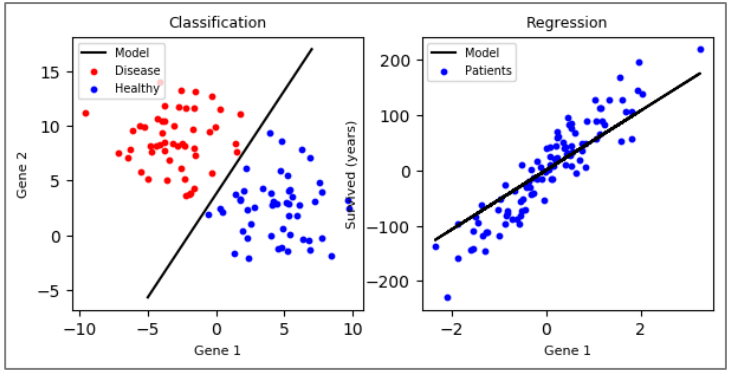
https://www.r-bloggers.com/2020/12/least-squares-as-springs-the-shiny-app/

ML Modeling: Algorithm Selection

Pick an algorithm/implementation that is applicable (not all algorithms are)

Learning Goal and Data Format:

- Supervised – labeled data
- Semi-Supervised - partially-labeled data
- Unsupervised - unlabeled data
- Reinforcement - interacting with environment

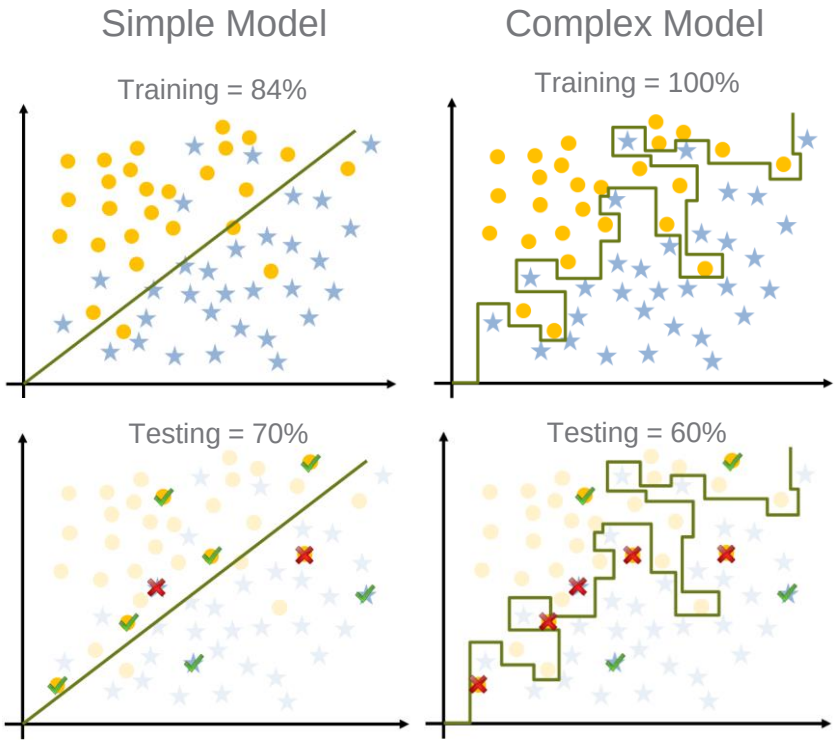


Outcome type: binary class or multi-class (classification), continuous-valued (regression)

Feature types: categorical, numerical, a mix of both

ML Modeling: Algorithm Selection – Start Simple

- If there is a strong, simple relationship among variables, most methods will find it
- Generally, start with simpler (and faster) algorithms if you know nothing about the problem
  - Linear/Logistic Regression
  - Decision Tree
  - Naïve Bayes
- Consider the strengths and weaknesses of algorithms
  - Assumptions made, computational requirements, data size, interpretability, ease of use

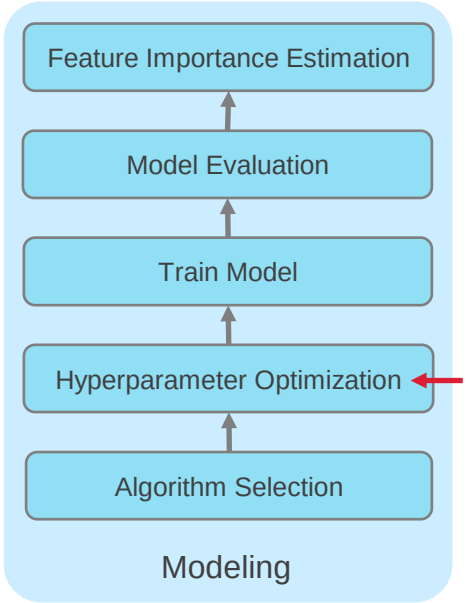


ML Modeling: Hyperparameters

Hyperparameters

- An algorithm configuration external to the model
- Value cannot be estimated from the data
- Specified by practitioner
- Can be set by using heuristics
- Often tuned to modeling problem

- Most algorithms have default settings for hyperparameters
- Default hyperparameters are a starting point only
  - Do not rely on them!



ML Modeling: Hyperparameter Optimization

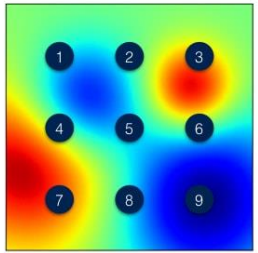
How do we go about optimizing algorithm hyperparameters?

Hyperparameter Sweep:

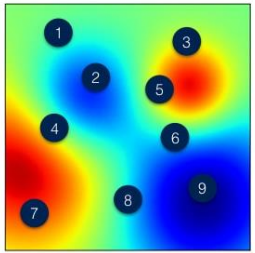
Search for the best hyperparameter configuration of an algorithm on a given dataset

- Strategies:
  - Manual search (attempt to apply domain knowledge)
  - Grid Search
  - Random Search
  - Smart Search
    - Bayesian Optimization (e.g. Optuna)
    - Evolutionary Optimization

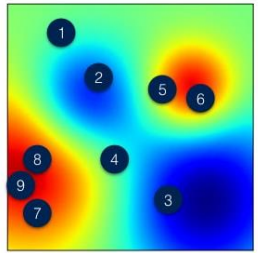
OPTUNA



Grid Search



Random Search



Smart Search

https://blog.ml.cmu.edu/2018/12/12/massively-parallel-hyperparameter-optimization/
https://optuna.org/

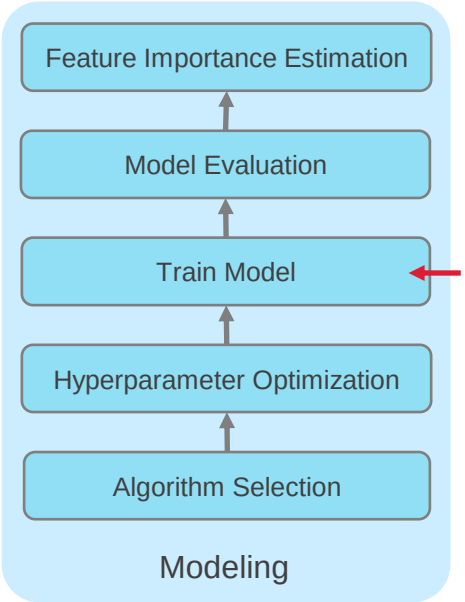
Modeling: Training

Train the ‘final’ model using the ‘optimized’ hyperparameter configuration

Decision Tree Example:

- CV Validation Accuracy = 0.6818
- **Final Model**
  - Training Accuracy: 0.818
  - Testing Accuracy: 0.757

- Training accuracy says little!
  - Generally, ignore it

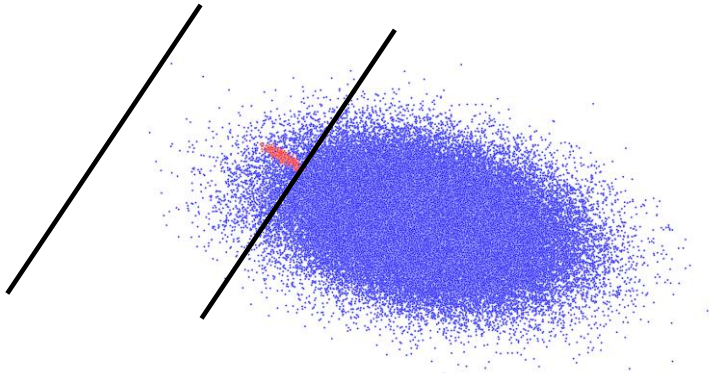


https://www.analyticsvidhya.com/blog/2020/02/underfitting-overfitting-best-fitting-machine-learning/

Modeling: Evaluation

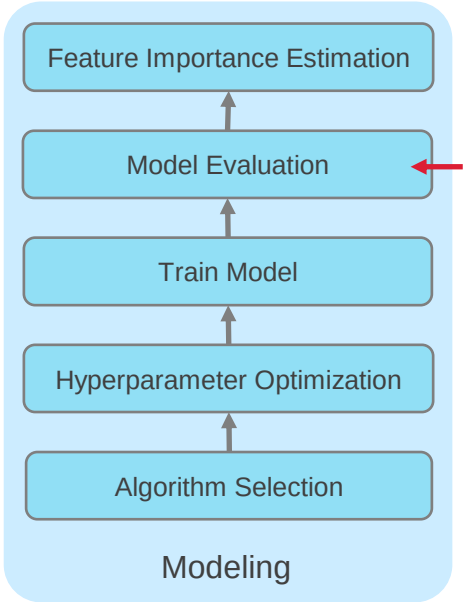
Using different evaluation metrics to understand an ML model's performance

- Many available, which to use?
- **Consider:**
  - Using standard accuracy as metric for an imbalanced dataset



Accuracy = 0.92 Accuracy = 0.95

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

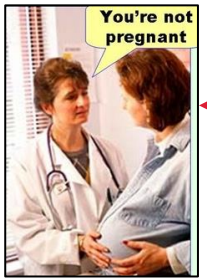


https://towardsdatascience.com/weighted-logistic-regression-for-imbalanced-dataset-9a5cd88e68b

Modeling: Evaluation – Confusion Matrix for Classification

All classification metrics are calculated based on a confusion matrix

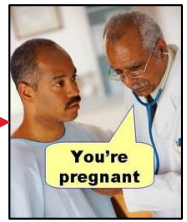
TP	True Positive
TN	True Negative
FN	False Negative
FP	False Positive



True Class

Predicted Class	Positive	Negative
Positive	TP	False Alarm FP Type I Error
Negative	Missed It FN Type II Error	TN

Binary Classification



True Class

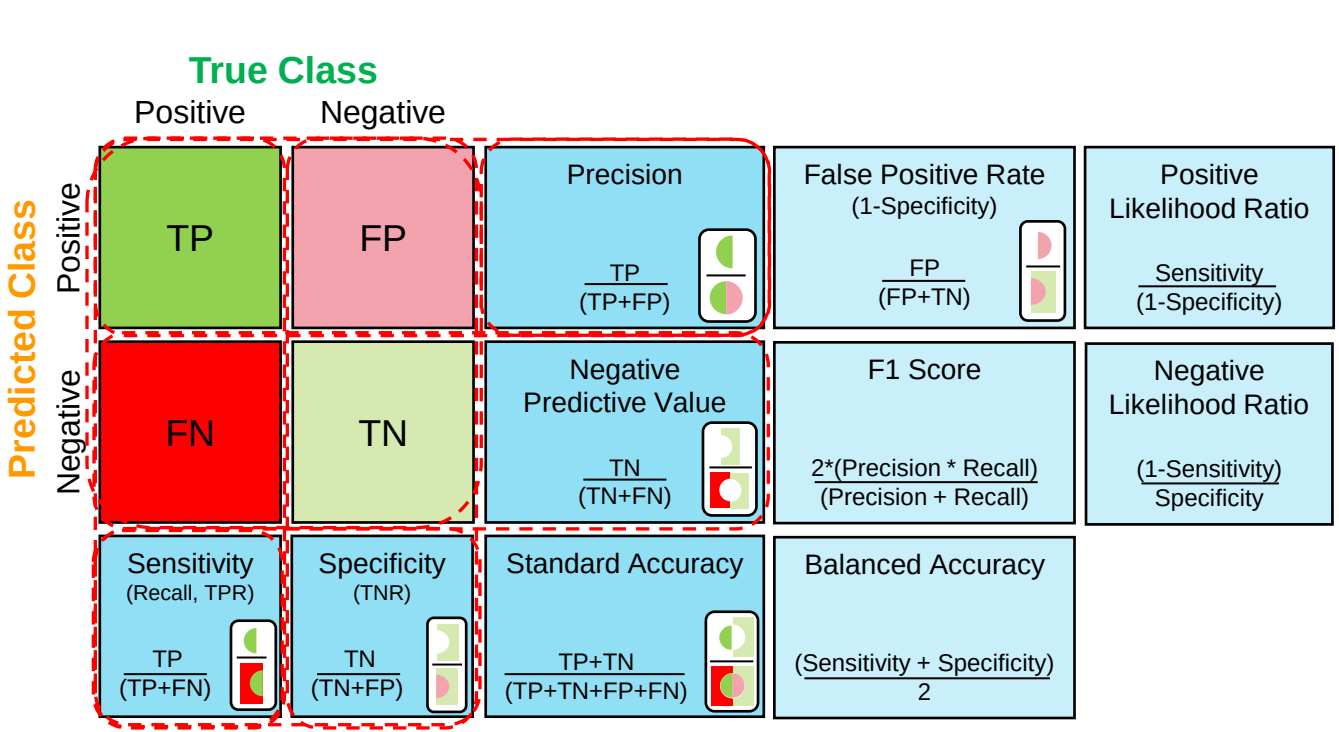
Predicted Class	$C_0 \dots C_{k-1}$	C_k	$C_{k+1} \dots C_n$
$C_{k+1} \dots C_n$	TN	FN	TN
C_k	FP	TP	FP
$C_0 \dots C_{k-1}$	TN	FN	TN

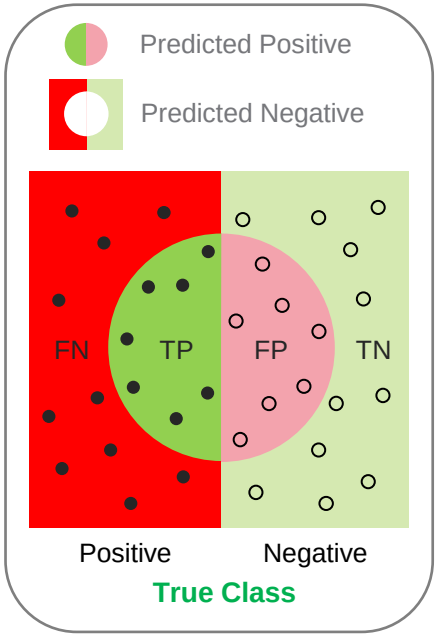
Multi-Class Classification*

*For multi-class problems, calculate average of target metric over each of k classes

Adapted From: https://www.researchgate.net/figure/Confusion-matrix-for-multi-class-classification-The-confusion-matrix-of-a_fig7_314116591

Modeling: Evaluation – Classification Metrics





Modeling: Evaluation – Classification Metrics - ROC

Receiver Operating Characteristic (ROC):

- A graph showing the performance of classification at all classification thresholds

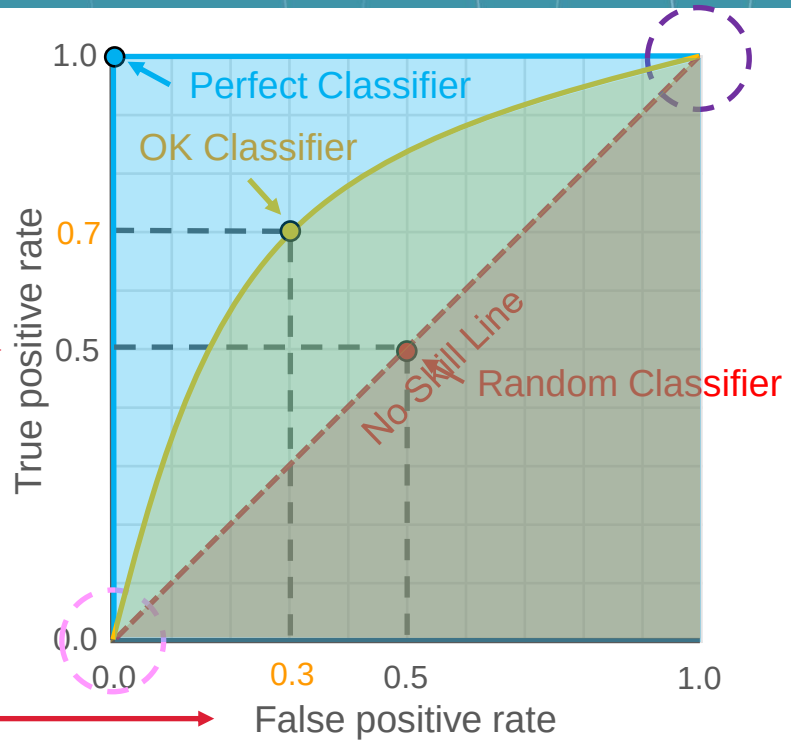
- Start by looking at typical threshold of 0.5
- No skill line – No better than random
- **Decision Threshold Extremes:**
  - Model - All predicted positive
  - Model - All predicted negative
- Area under the curve (AUC) used to summarize ROC
- High class imbalance:
  - AUROC overly optimistic

Sensitivity
(Recall, TPR)

$$\frac{TP}{(TP+FN)}$$

False Positive Rate
(1-Specificity)

$$\frac{FP}{(FP+TN)}$$



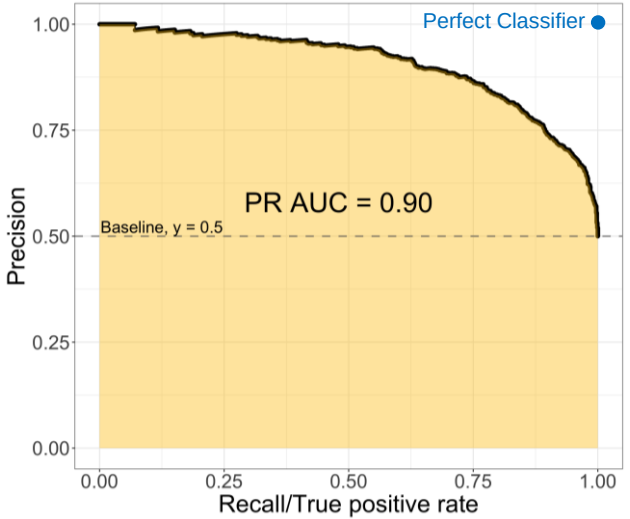
Modeling: Evaluation – Classification Metrics - PRC

Precision-Recall Curve (PRC):

- A graph showing tradeoff of precision and recall at all classification thresholds
  - Use when positive class (P) is of more interest than negative class (N) or when there is significant class imbalance
  - Again, class prediction probabilities needed
  - "No skill" or (base) line is determined by class balance: (P / (P+N))
    - 0.5 for balanced data
    - Higher when P>N, lower when N>P
  - PR AUC used to summarize PRC

Precision

$$\frac{TP}{(TP+FP)}$$



Sensitivity
(Recall, TPR)

$$\frac{TP}{(TP+FN)}$$

Modeling: Evaluation – Regression Metrics

Three common regression metrics:

- **Mean Squared Error (MSE):**
  - Important loss function for algorithm fit
  - Magnifies impact of larger errors
  - Units are squared which can be confusing when reported
- **Root Mean Squared Error (RMSE):**
  - Units preserved
- **Mean Absolute Error (MAE):**
  - Units preserved
  - Does not weigh larger errors more

Note: Accuracy (maximize) vs. Error (minimize)

Difference between predicted and real values = $\hat{r}_{ui} - r_{ui}$

$$MSE = \frac{\sum_{i=1}^{n}(\hat{r}_{ui} - r_{ui})^2}{n}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{r}_{ui} - r_{ui})^2}{n}}$$

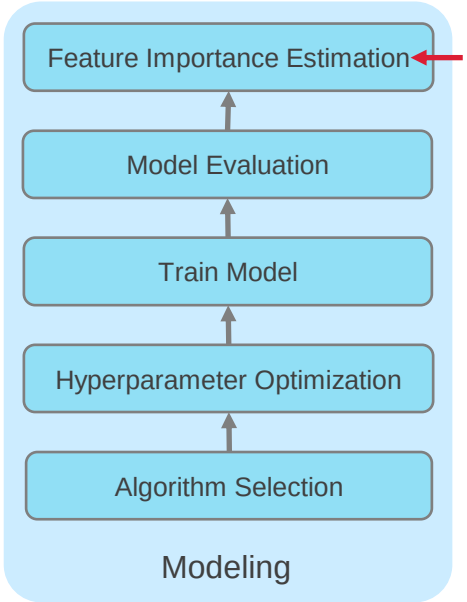
$$MAE = \frac{\sum_{i=1}^{n}|\hat{r}_{ui} - r_{ui}|}{n}$$

https://machinelearningmastery.com/regression-metrics-for-machine-learning/#:~:text=Regression%20refers%20to%20predictive%20modeling,involves%20predicting%20a%20class%20label.

Modeling: Feature Importance (FI) Estimation

Measuring/estimating the contribution of individual features to the performance of a supervised learning model

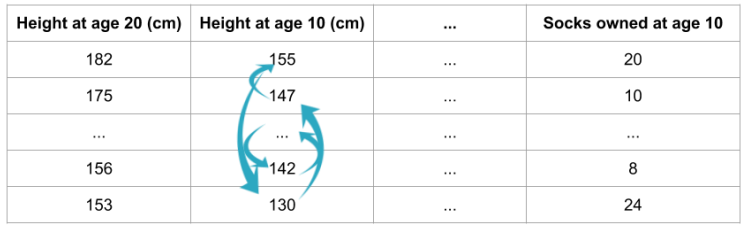
- Why?
  - Better understand the data and model
  - Facilitate another round of feature selection and re-modeling
- Some (not all) algorithms offer built-in strategy for feature importance estimation
  - These are algorithm-specific estimates (don't capture the same thing)
- Alternatively, permutation-based approaches can be applied in an algorithm-independent manner



Modeling: Feature Importance Estimation - Permutation Estimates

Permutation Feature Importance: A technique to calculate relative feature importance independent of the model/algorithm used

- Compare model performance (on original training or testing data) to average performance on ‘p’ permutations of (train or test) data with given feature’s values shuffled
- Scikit-learn Permutation Importance
  - `results = permutation_importance(model, X, y, scoring='accuracy’)`
  - `importance = results.importances_mean`



Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24

https://www.kaggle.com/code/dansbecker/permutation-importance/tutorial
https://machinelearningmastery.com/calculate-feature-importance-with-python/

Post-Analysis

Summary Statistics:

- Average evaluation metrics, feature importance scores, and runtime

Statistical Significance:

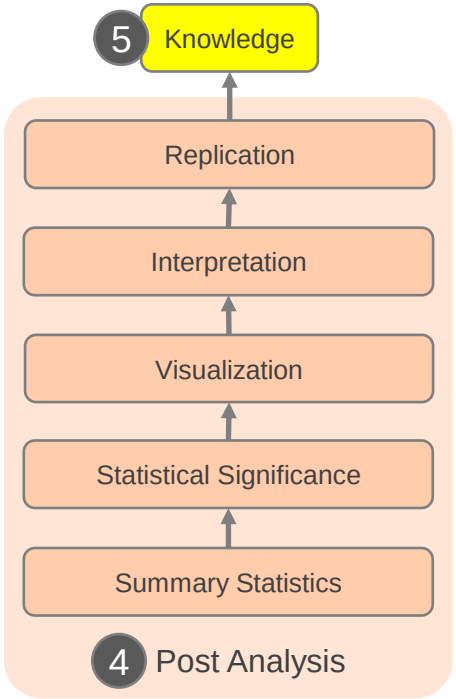
- Compare algorithm/dataset performance differences with parametric or non-parametric significance tests

Visualization and Interpretation:

- Summarize results, understand relevant features and model predictions

Replication (Gold standard of science):

- Repeat ML analysis in same data? Replication data? Applied data?



AutoML: Making Things Easier

Why Automate ML?

Effectively infinite ML paths/pipelines…

- How should the data be cleaned and/or transformed?
- What features should be engineered?
- What features should be selected?
- What model/method should we use?
- What algorithm run parameters will yield optimal performance?

Answers may be different for every new task/problem…

- There is no single correct way to assemble an ML analysis pipeline

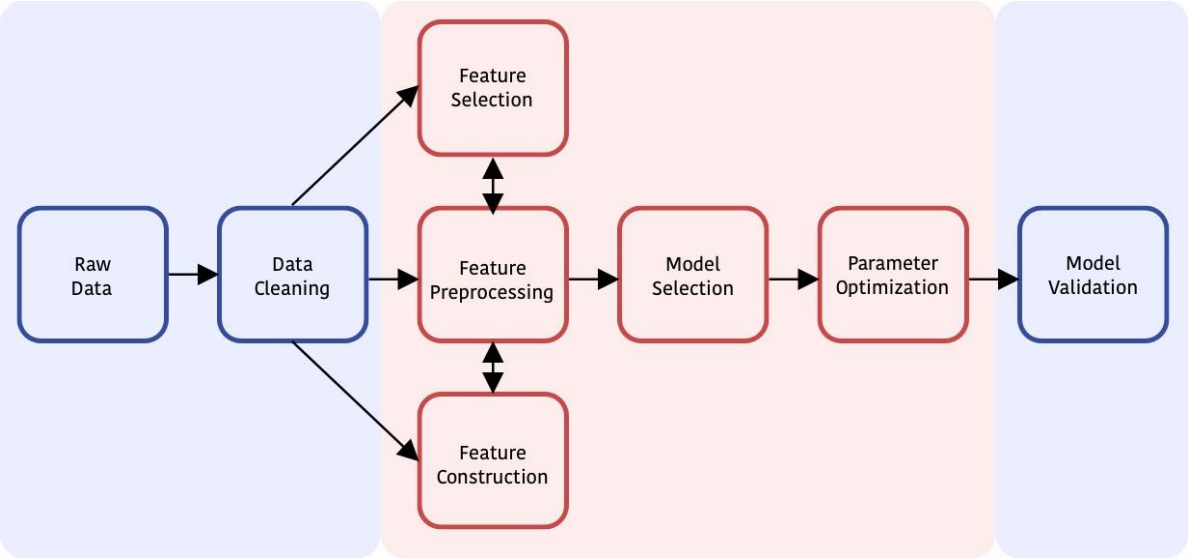
AutoML: Automated Machine Learning

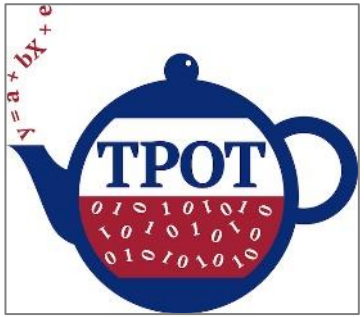
Process of automating the elements of applying ML to real-world problems

- Automate any number of the elements involved (potentially every one, to some degree)
- Primary aim is to allow non-experts to make use of ML models and techniques
- Has the potential to produce solutions more quickly and easily that are:
  - Simpler and with better performance
- Can facilitate a more rigorous exploration of ML model optimization
- Most commonly automated components:
  - Hyperparameter Optimization
  - Model Selection
  - Feature Extraction
  - Feature Processing: feature selection, feature learning, feature transformation

AutoML: TPOT - A Genetic Programming-Based AutoML

Automates the process of identifying the optimal combination of ML analysis pipeline elements below (in orange)





https://github.com/epistasislab/tpot/

Olson, Randal S., and Jason H. Moore. "TPOT: A tree-based pipeline optimization tool for automating machine learning." Workshop on automatic machine learning. PMLR, 2016.

AutoML: PennAI - Putting Previous Analysis Experience to Work

- Human specified and AI recommended ML experiments
- All results for every dataset organized in graph database
- Seek to improve AI recommendation system over time

https://github.com/EpistasisLab/pennai

La Cava, William, et al. "Evaluating recommender systems for AI-driven biomedical informatics." Bioinformatics 37.2 (2021): 250-256.

AutoML: Other Popular Options

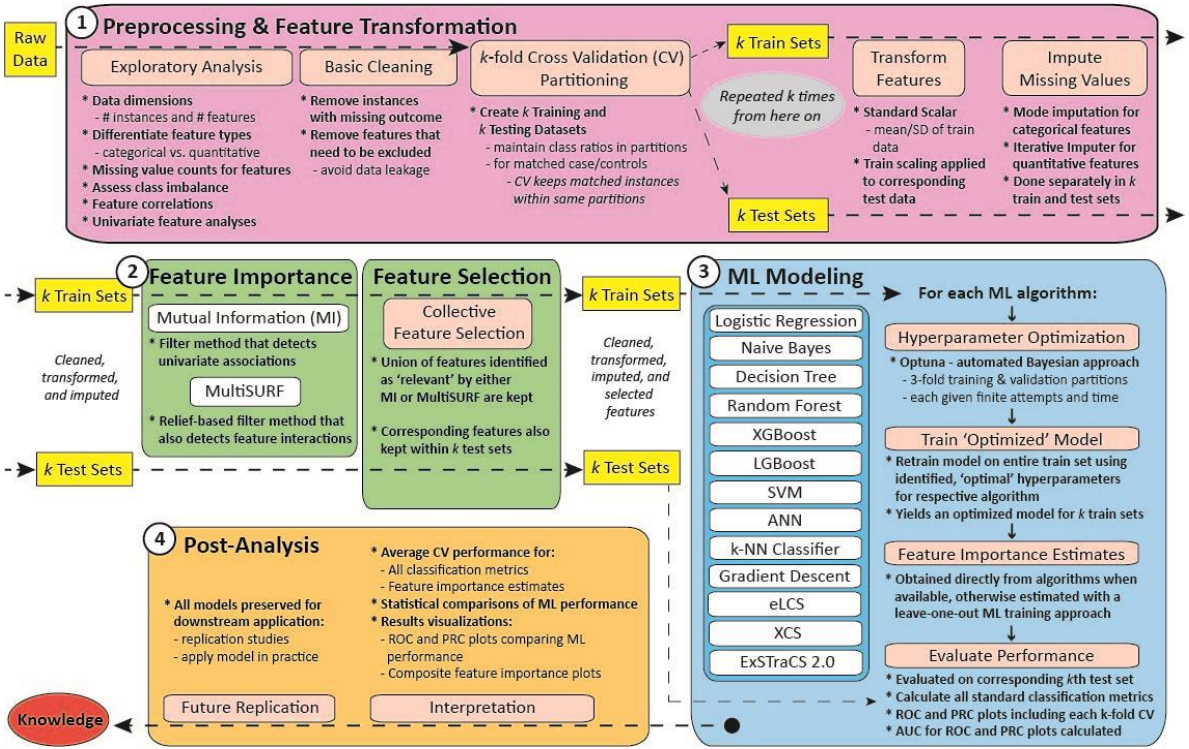
- Auto-sklearn: (python) github.com/automl/auto-sklearn
- MLBox: (python) github.com/AxeldeRomblay/MLBox
- Auto-Weka: (java) github.com/automl/autoweka
- H20.ai: (java w/ python, scala, R, web GUI) github.com/h2oai/h2o-3
- Devol: (python) github.com/joeddav/devol
- Auto-Keras: (python) github.com/keras-team/autokeras
- TransmorgrifAI: (scala) github.com/salesforce/TransmogrifAI

https://www.alibabacloud.com/blog/6-top-automl-frameworks-for-machine-learning-applications-may-2019_595317

AutoML: Limitations and Risks

- Current AutoML methods typically focus on finding and returning a single ‘best’ model
  - Ensembles may be most effective for a specific problem
  - Lost opportunity to learn from the perspectives of multiple ML algorithms
- Not all elements of ML analysis can (yet) be reliably/effectively automated
  - e.g. data cleaning, feature engineering, bias identification, interpretation
  - Best ML analysis still employs domain knowledge and human experts in the loop
- Can be extremely computationally expensive
- We still have a lot to learn and discover with regards to ML methods
  - Any AutoML tool is limited by the algorithms and heuristics it employs
  - Can be the ultimate black box

AutoML: AutoMLPipe-BC – Overview & Demonstration



Robert Zhang

Allan Pack

Shannon Lynch

Using AutoML: An ‘AutoMLPipe-BC’ Demonstration

AutoMLPipe-BC: Overview

Automated Machine Learning Analysis Pipeline (for Binary Classification)

Description:

- 'A' – 'Z' pipeline for ML predictive modeling in a rigorous, transparent manner
  - Automates: Exploratory analysis, data processing, feature processing, modeling, analysis summary, significance analysis, figure generation, and PDF report generation
- Adopts 'best practices' in ML analysis
- Offers user flexibility/customizability/expandability
- **Designed to:**
  - Deal with common data challenges
  - Scale to bigger data analyses
  - Detect and interpret complex associations

AutoMLPipe-BC: Target Uses

Given a binary classification dataset/problem:

- Compare performance of different ML algorithms on given data
- Compare ML performance across multiple target datasets (e.g.):
  - Utilizing different feature subsets
  - Adjusting (or not) for covariates
- Easily apply trained models to future data (e.g. replication data, or as clinical decision support)
- Serve as baseline of comparison to evaluate other AutoML tools that output a ‘best’ model
- Educational resource for those looking to design their own custom ML analysis pipeline

AutoMLPipe-BC: Logistics

Basics:

- Coded in Python 3 and makes use of established python packages:
  - e.g. scikit-learn, pandas, numpy, scipy, matplotlib, optuna, etc.

Ways to run it:

- For those with basic knowledge of Python and Unix:
  - **Run from command line (as 6 primary and 5 optional scripts/phases)**:
    - Parallelable, fastest option for larger analyses
- Those with little to no code experience:
  - **Some assembly required:** As a 'Jupyter Notebook' (on your local computer)
    - Need to install Anaconda, and required packages in your compute environment
  - **Simplest:** As a 'Google Colab' notebook (via webpage and google cloud)
    - No installations or coding knowledge required, but slowest (smaller analyses only)

AutoMLPipe-BC: Overview of Pipeline Elements and Phases

Phase 1 Exploratory Analysis

Phase 2 Data Preprocessing

Phase 3 Feature Importance

Phase 4 Feature Selection

Phase 5 Model

Phase 6: Stats

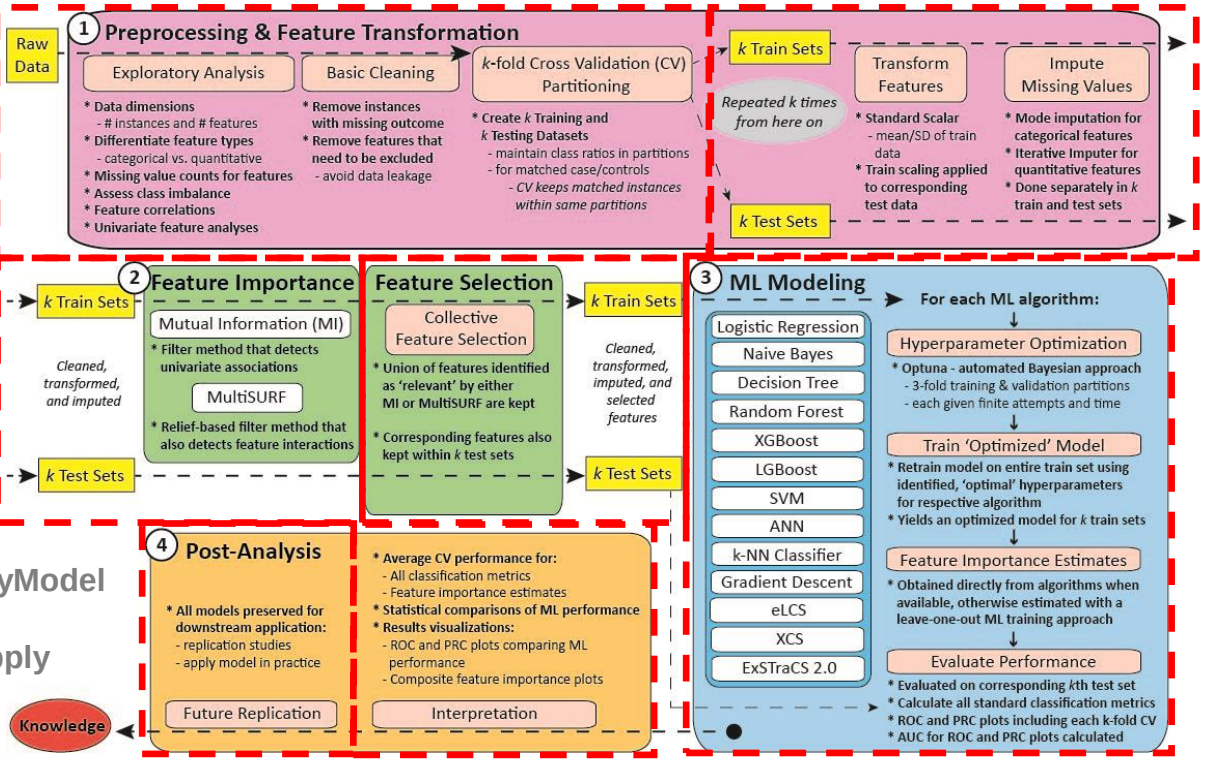
Phase 7: Data Compare

Phase 8: PDF ReportTrain

Phase 9: ApplyModel

Phase 10: PDF ReportApply

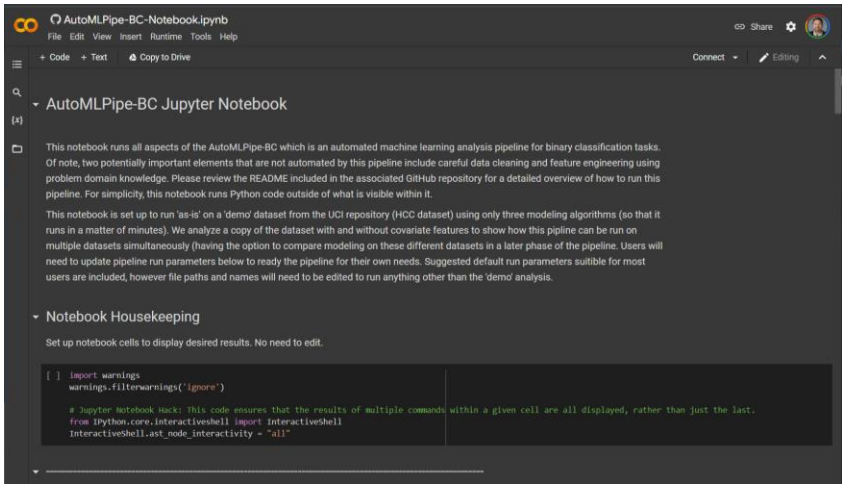
Phase 11: FileCleanup



AutoMLPipe-BC: Follow Along in Google Colab

Open This Link To VIEW (Previously Run) Notebook Code :

- https://colab.research.google.com/github/UrbsLab/AutoMLPipe-BC/blob/main/AutoMLPipe-BC-Notebook.ipynb



AutoMLPipe-BC: Code Base and Documentation on GitHub

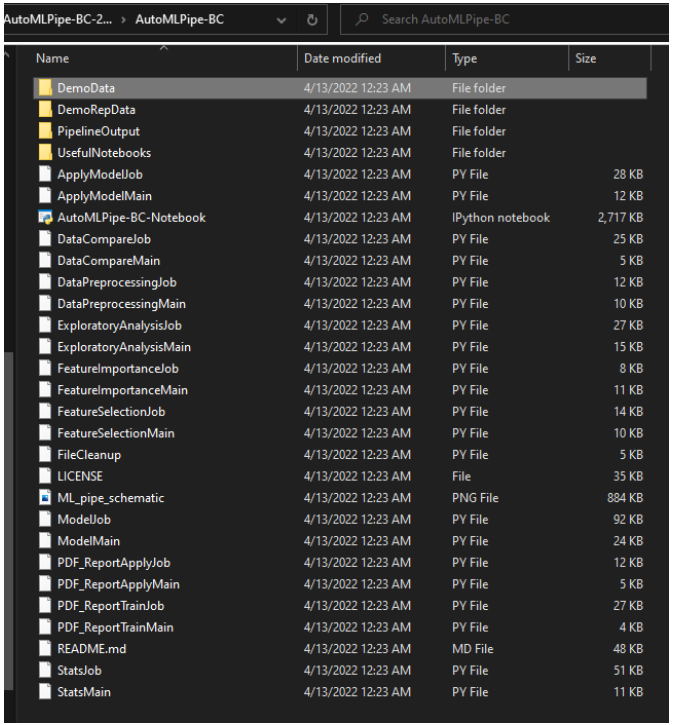
https://github.com/UrbsLab/AutoMLPipe-BC

Repository Includes:

- Python scripts for each phase
- Jupyter notebook for pipeline
- DEMO dataset folders
- Accessory Jupyter notebooks

README:

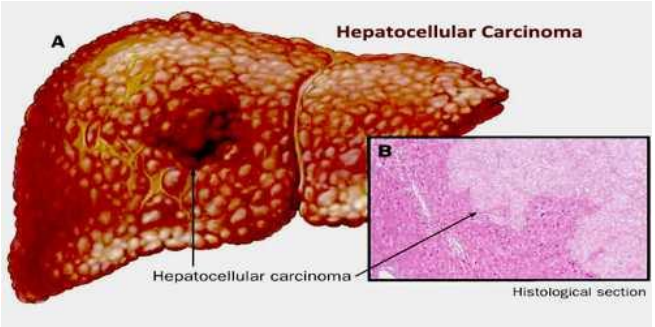
- Summary, Overview, Installation, Usage
- Troubleshooting



AutoMLPipe-BC: Demonstration using HCC Dataset from UCI

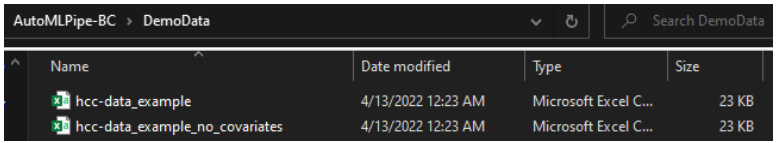
Hepatocellular Carcinoma (HCC) Survival Data

- AutoMLPipe-BC/DemoData/hcc-data_example.csv



- **Why This Dataset?**
  - It's small:
    - 165 Instances
    - 49 Features
  - Biomedical Classification Task (Survived or Deceased at 1 Year)
  - Exemplifies many data considerations/challenges
    - Mixed Feature Types (Categorical vs. Numeric)
    - Missing Data Values (~10%)
    - Class Imbalance (63 Deceased, 102 Survived)

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

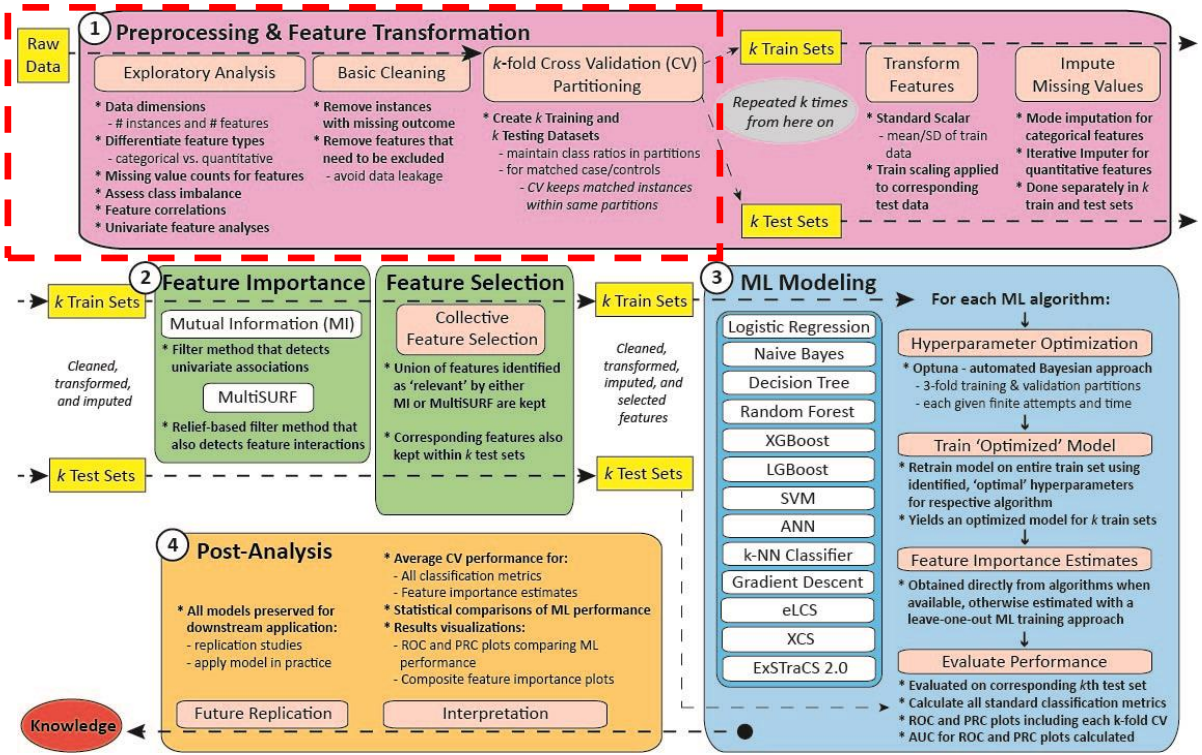


Cedars Sinai

Download Data: https://archive.ics.uci.edu/ml/datasets/HCC+Survival
https://www.assignmentpoint.com/science/medical/hepatocellular-carcinoma.html

AutoMLPipe-BC: Phase 1 – Exploratory Analysis

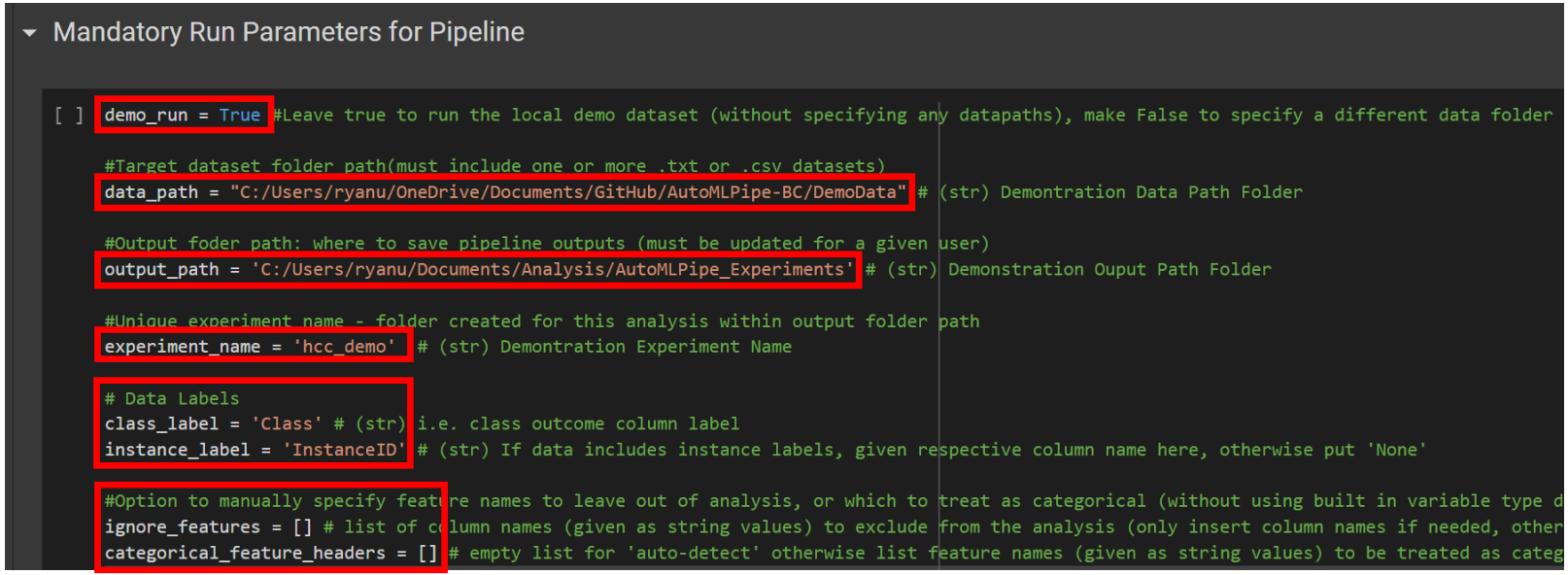
Phase 1
Exploratory Analysis



AutoMLPipe-BC: Phase 1 – Exploratory Analysis – Run Parameters

At minimum, users must check and set these pipeline run parameters

- In this pre-run Jupyter Notebook example from GitHub, output was sent to a local directory



AutoMLPipe-BC: Phase 1 – Exploratory Analysis – Run Parameters

Further run parameters with default value options…

Run Parameters for Phase 1: Exploratory Analysis

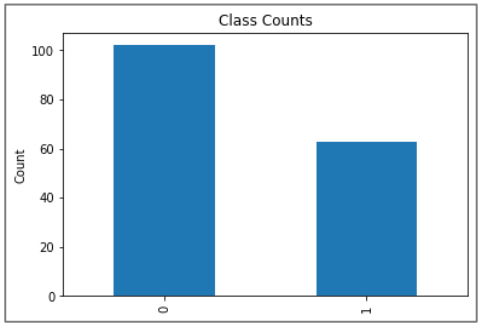
```
cv_partitions = 3 # (int, > 1) Number of training/testing data partitions to create - and resulting number of models
partition_method = 'S' # (str, S R or M) for stratified, random, or matched, respectively
match_label = 'None' # (str) Only applies when M selected for partition-method; indicates column label with matched in

categorical_cutoff = 10 # (int) Bumber of unique values after which a variable is considered to be quantitative vs cat
sig_cutoff = 0.05 # (float, 0-1) Significance cutoff used throughout pipeline
export_feature_correlations = 'True' # (str, True or False) Run and export feature correlation analysis (yields correl
export_univariate_plots = 'True' # (str, True or False) Export univariate analysis plots (note: univariate analysis st
topFeatures = 20 # (int) Number of top features to report in notebook for univariate analysis
random_state = 42 # (int) Sets a specific random seed for reproducible results
```

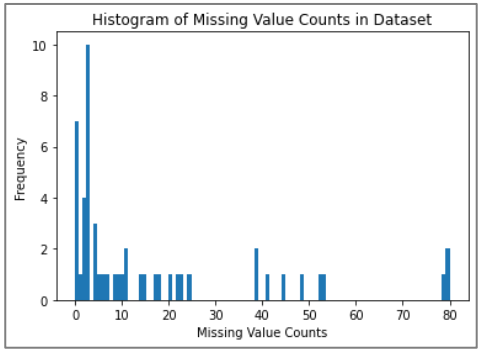
AutoMLPipe-BC: Phase 1 – Exploratory Analysis - Output

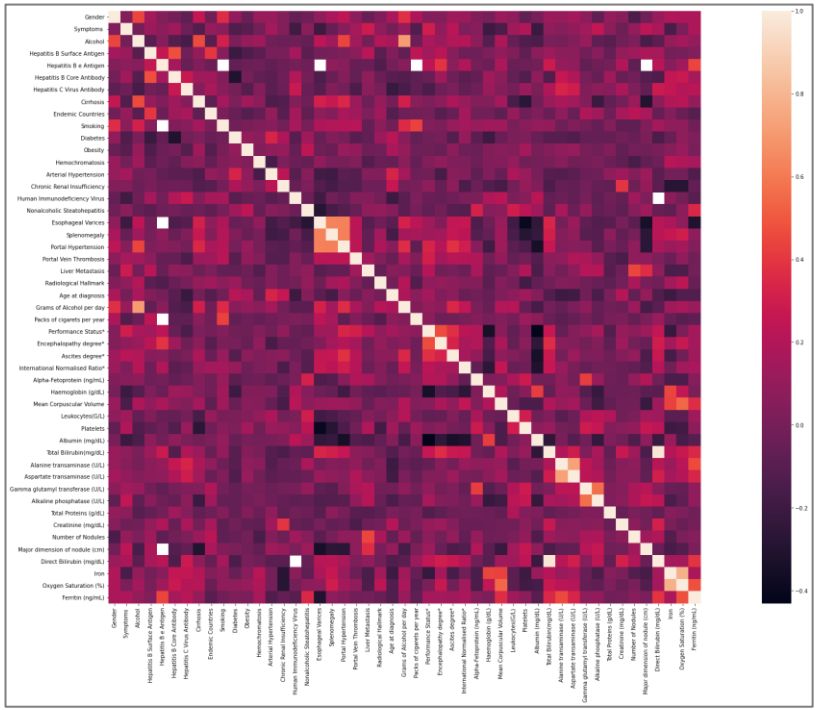
```
Data Counts: ----------------
Instance Count = 165
Feature Count = 49
    Categorical  = 27
    Quantitative = 22
Missing Count = 826
    Missing Percent = 0.10216450216450217
Class Counts: ----------------
0    102
1     63
```

```
################################################
Significant Univariate Associations:
Alkaline phosphatase (U/L): (p-val = 3.5403912777218106e-06)
Iron: (p-val = 3.5423866432400713e-06)
Alpha-Fetoprotein (ng/mL): (p-val = 1.0082106490397668e-05)
Haemoglobin (g/dL): (p-val = 1.3037749087103508e-05)
Performance Status*: (p-val = 3.2548676278782114e-05)
Oxygen Saturation (%): (p-val = 3.998706534072513e-05)
Albumin (mg/dL): (p-val = 5.159435074542993e-05)
Symptoms : (p-val = 0.0006092985105592953)
Aspartate transaminase (U/L): (p-val = 0.002884797765802902)
Liver Metastasis: (p-val = 0.002993588224869906)
Ascites degree*: (p-val = 0.0038134308539161175)
Ferritin (ng/mL): (p-val = 0.004446494113520735)
Portal Vein Thrombosis: (p-val = 0.011743041155542567)
Major dimension of nodule (cm): (p-val = 0.01569067499758109)
Age at diagnosis: (p-val = 0.01784161875604351)
Gamma glutamyl transferase (U/L): (p-val = 0.024388667167652392)
Total Proteins (g/dL): (p-val = 0.029520535772105137)
Encephalopathy degree*: (p-val = 0.03673986822541975)
International Normalised Ratio*: (p-val = 0.050292043910706546)
Total Bilirubin(mg/dL): (p-val = 0.0648371068893193)
```

```
topFeatures = 20
```



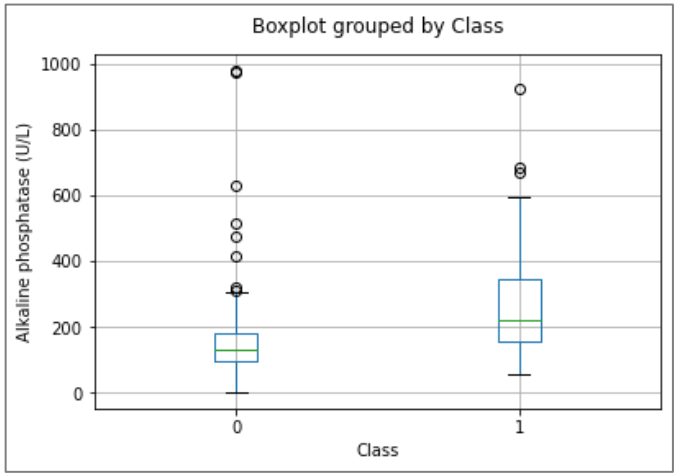


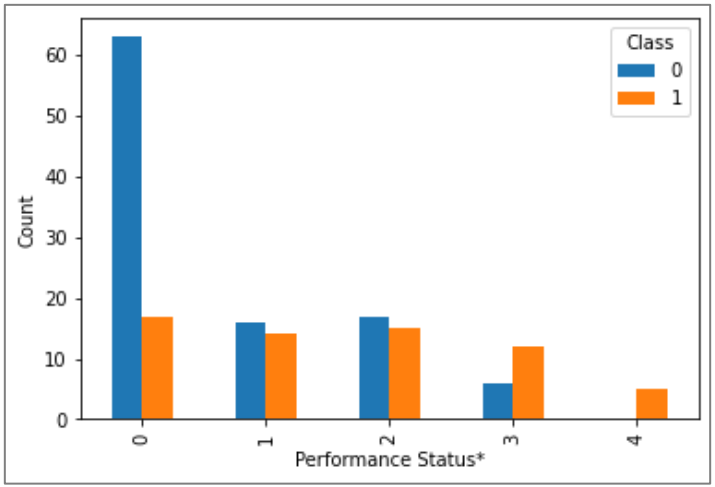


```
export_feature_correlations = 'True'
```

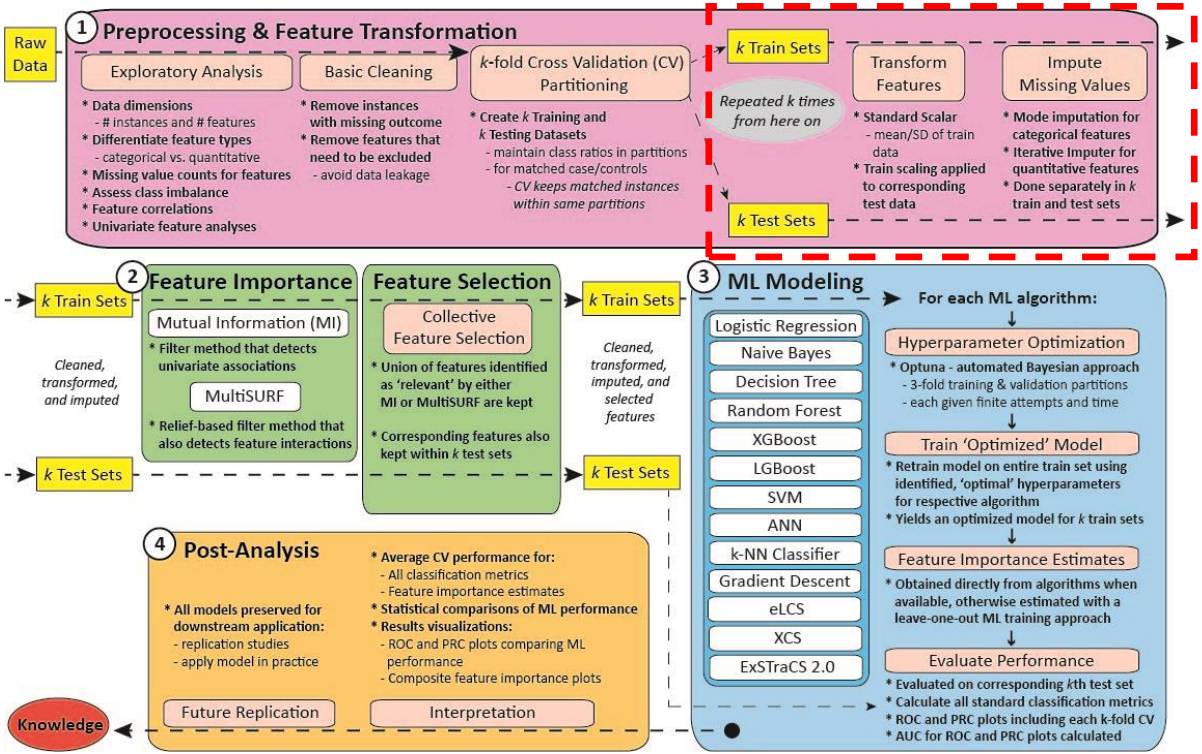
AutoMLPipe-BC: Phase 1 – Exploratory Analysis - Output

```
export_univariate_plots = 'True'
```





AutoMLPipe-BC: Phase 2 – Data Preprocessing



Phase 2
Data Preprocessing

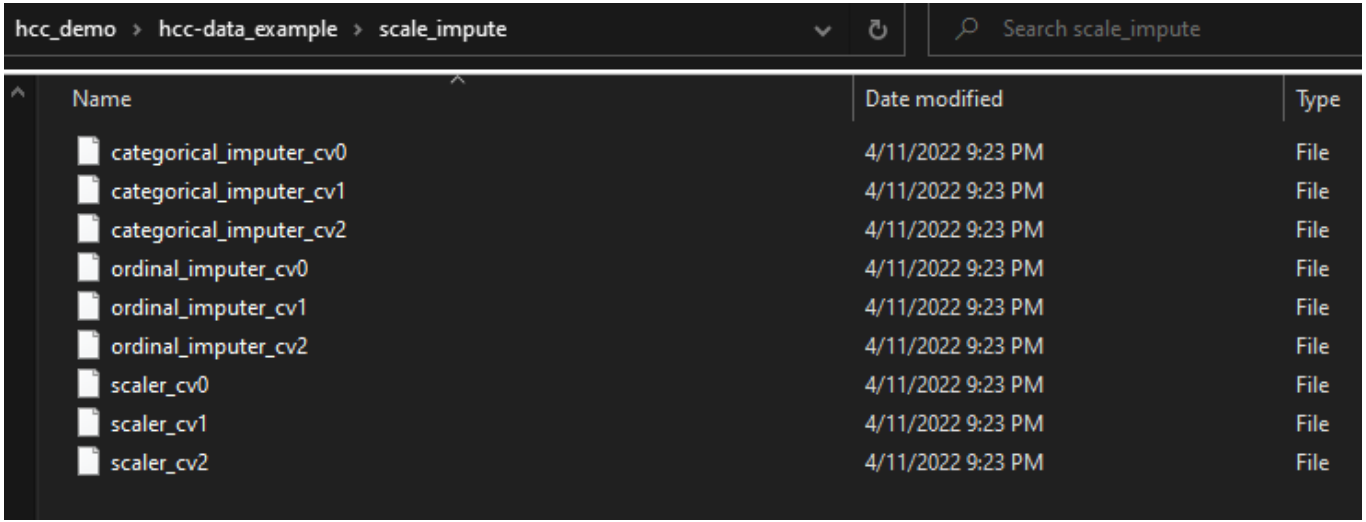
AutoMLPipe-BC: Phase 2 – Data Preprocessing – Run Parameters

Run Parameters for Phase 2: Data Preprocessing

```
[ ] scale_data = 'True' # (str, True or False) Perform data scaling?
    impute_data = 'True'  # (str, True or False) Perform missing value data imputation? (required for most ML algorithms
    overwrite_cv = 'True' # (str, True or False) Overwrites earlier cv datasets with new scaled/imputed ones
    multi_impute = 'True' # (str, True or False) Applies multivariate imputation to quantitative features, otherwise use
```

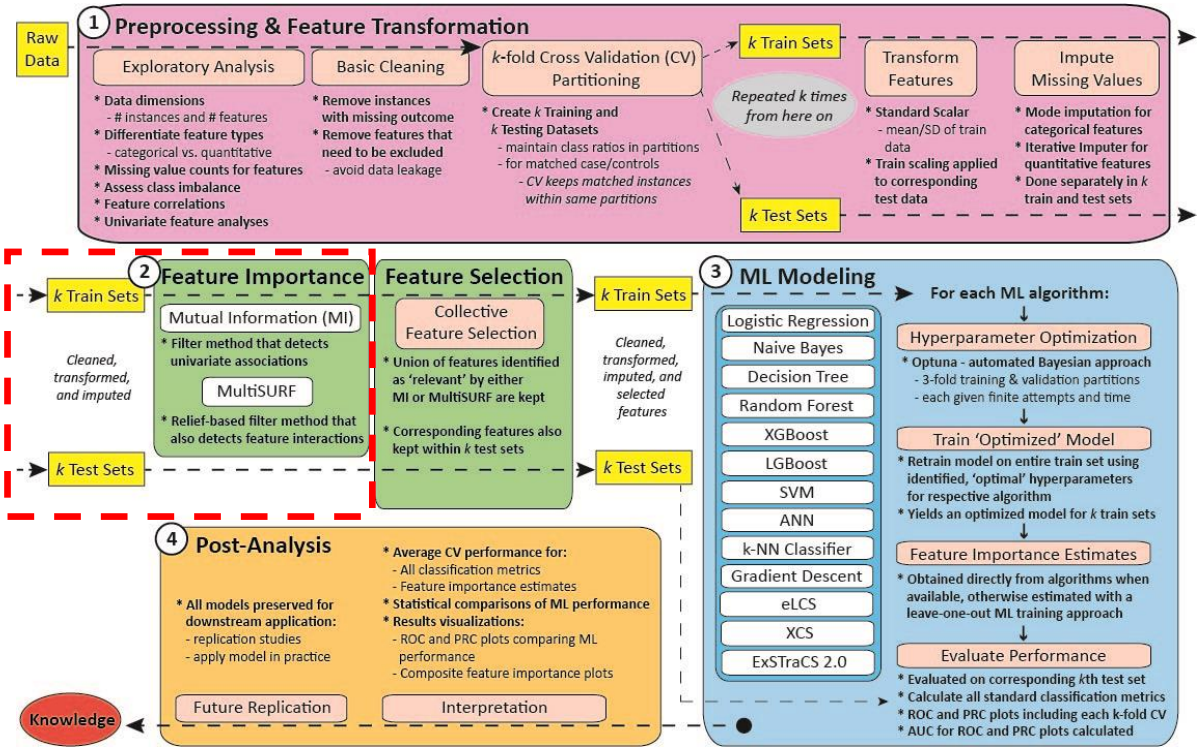
AutoMLPipe-BC: Phase 2 – Data Preprocessing – Output

Scaling and imputation information ‘pickled’ for future use in test or replication data



AutoMLPipe-BC: Phase 3 – Feature Importance

Phase 3
Feature Importance

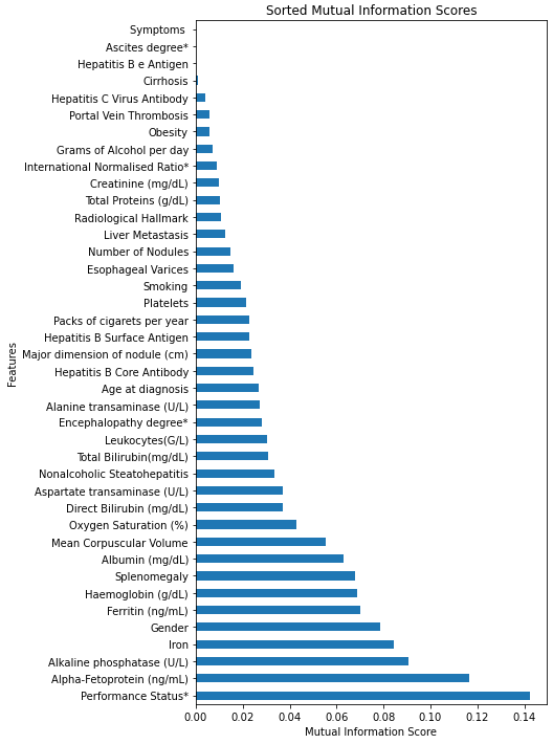


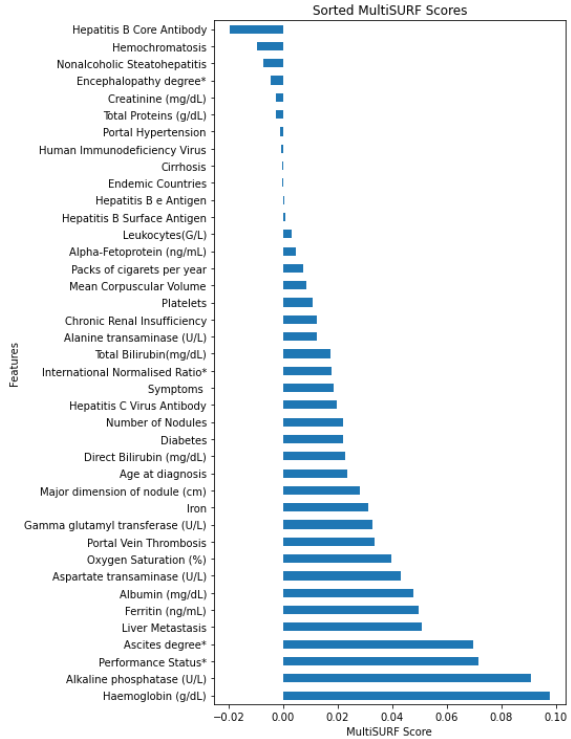
AutoMLPipe-BC: Phase 3 – Feature Importance – Run Parameters

Run Parameters for Phase 3: Feature Importance Evaluation

```
[ ] do_mutual_info = 'True' # (str, True or False) Do mutual information analysis
    do_multisurf = 'True' # (str, True or False) Do multiSURF analysis
    use_TURF = 'False' # (str, True or False) Use TURF wrapper around MultiSURF
    TURF_pct = 0.5 # (float, 0-1) Proportion of instances removed in an iteration (also dictates number of iterations)
    njobs = -1 # (int) Number of cores dedicated to running algorithm; setting to -1 will use all available cores
    instance_subset = 2000 # (int) Sample subset size to use with multiSURF
```

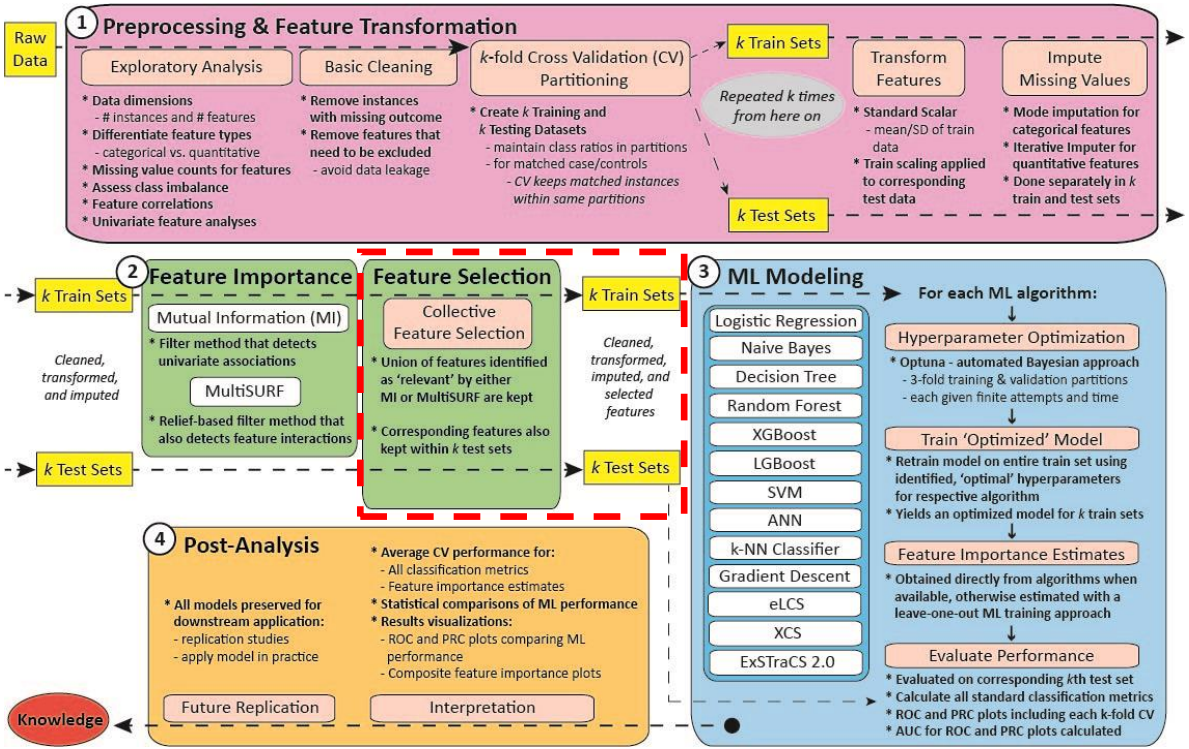
AutoMLPipe-BC: Phase 3 – Feature Importance - Output





AutoMLPipe-BC: Phase 4 – Feature Selection

Phase 4
Feature Selection



AutoMLPipe-BC: Phase 4 – Feature Selection – Run Parameters

Run Parameters for Phase 4: Feature Selection

```
[ ] max_features_to_keep = 2000 # (int) Maximum features to keep. 'None' if no max
    filter_poor_features = 'True' # (str, True or False) Filter out the worst performing features prior to modeling
    top_results = 40 # (int) Number of top features to illustrate in figures
    export_scores = 'True' # (str, True or False) Export figure summarizing average feature importance scores over c
```

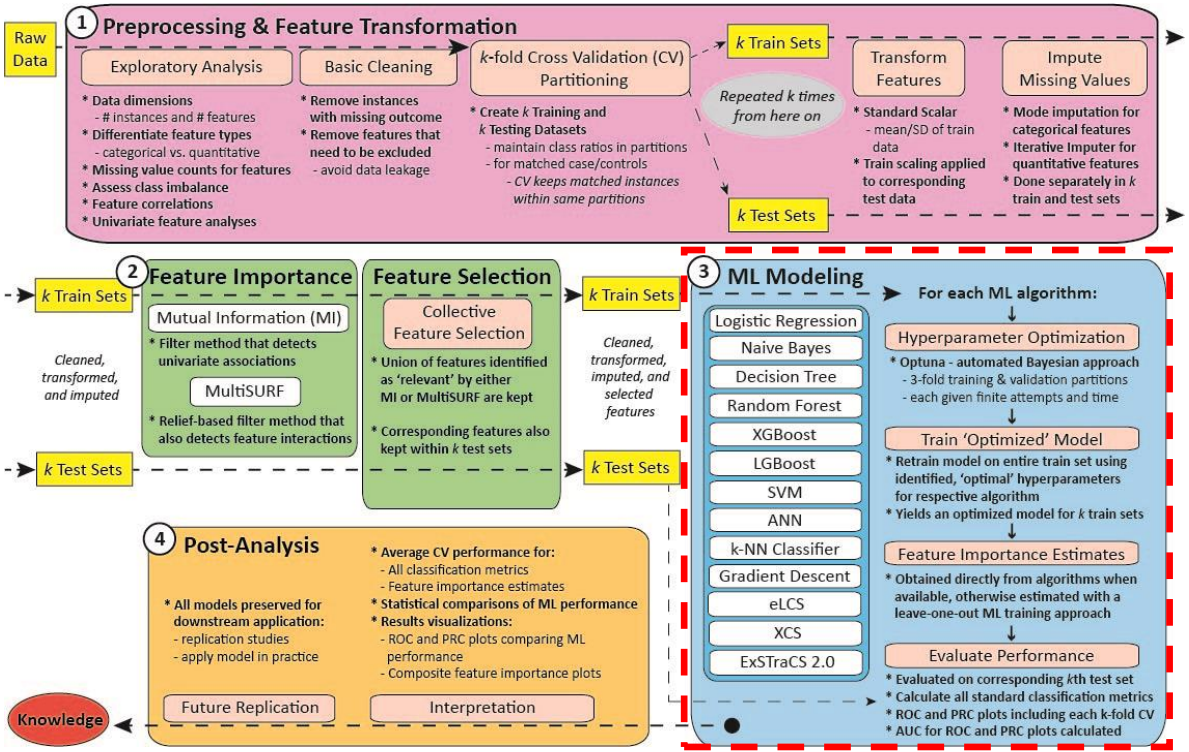
AutoMLPipe-BC: Phase 4 – Feature Selection - Output

Collective Feature Selection:

- Keep the union of informative features from all feature importance algorithms

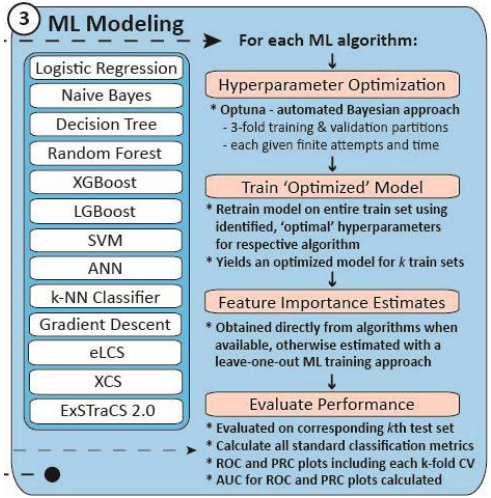
CV_Partition	Informative	Uninformative
0	39	10
1	34	15
2	38	11

AutoMLPipe-BC: Phase 5 – Model



Phase 5
Model

AutoMLPipe-BC: Phase 5 – Model – Run Parameters



Run Parameters for Phase 5: Modeling

```
#ML Model Algorithm Options (individual hyperparameter options can be adjusted below)
do_NB = 'True'          # (str, True or False) Run naive bayes modeling
do_LR = 'True'          # (str, True or False) Run logistic regression modeling
do_DT = 'True'          # (str, True or False) Run decision tree modeling
do_RF = 'False'         # (str, True or False) Run random forest modeling
do_GB = 'False'         # (str, True or False) Run gradient boosting modeling
do_XGB = 'False'        # (str, True or False) Run XGBoost modeling
do_LGB = 'False'        # (str, True or False) Run LGBoost modeling
do_SVM = 'False'        # (str, True or False) Run support vector machine modeling
do_ANN = 'False'        # (str, True or False) Run artificial neural network modeling
do_KN = 'False'         # (str, True or False) Run k-neighbors classifier modeling

# Newly implemented Rule-based ML Algorithm Options (Computationally expensive, so impractical to run hyperparameter sweep)
do_eLCS = 'False'       # (str, True or False) Run eLCS modeling (a basic supervised-learning learning classifier system)
do_XCS = 'False'        # (str, True or False) Run XCS modeling (a supervised-learning-only implementation of the best studied learning classifier system)
do_ExSTraCS = 'False'   # (str, True or False) Run ExSTraCS modeling (a learning classifier system designed for biomedical data mining)

#Other Analysis Parameters
training_subsample = 0  # (int) For long running algorithms, option to subsample training set (0 for no subsample) Limit Sample Size Used to train algorithms
use_uniform_FI = 'True' # (str, True or False) Overides use of any available feature importances estimate methods from models, instead using permutation_impor
primary_metric = 'balanced_accuracy' # (str) Must be an available metric identifier from (https://scikit-learn.org/stable/modules/model_evaluation.html#scorin

#Hyperparameter Sweep Options
n_trials = 100    # (int) Number of bayesian hyperparameter optimization trials using optuna
timeout = 300     # (int) Seconds until hyperparameter sweep stops running new trials (Note: it may run longer to finish last trial started)
export_hyper_sweep_plots = 'True' # (str, True or False) Export hyper parameter sweep plots from optuna

#Learning classifier system specific options (ExSTraCS, eLCS, XCS)
do_lcs_sweep = 'False'  # (str, True or False) Do LCS hyperparam tuning or use below params
nu = 1                  # (int, 0-10) Fixed LCS nu param
iterations = 200000     # (int, > data sample size) Fixed LCS # learning iterations param
N = 2000                # (int) > 500) Fixed LCS rule population maximum size param
lcs_timeout = 1200      # (int) Seconds until hyperparameter sweep stops for LCS algorithms (evolutionary algorithms often require more time for a single run)
```

AutoMLPipe-BC: Phase 5 – Model – Hyperparameters

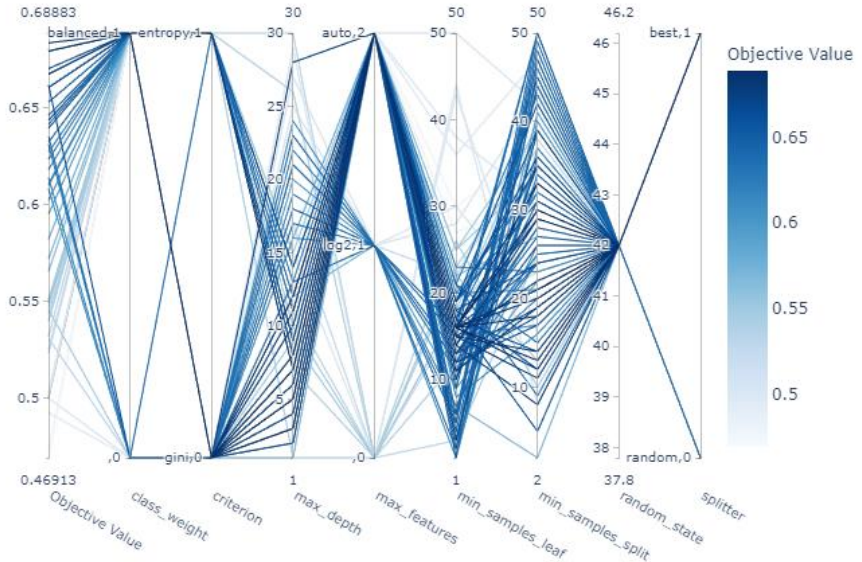
```
def hyperparameters(random_state,do_lcs_sweep,nu,iterations,N):
    param_grid = {}
    # Naive Bayes - no hyperparameters

    # Logistic Regression (Note: can take longer to run in data with larger instance spaces)
    # https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
    param_grid_LR = {'penalty': ['l2', 'l1'],'C': [1e-5, 1e5],'dual': [True, False],
                     'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
                     'class_weight': [None, 'balanced'],'max_iter': [10, 1000],
                     'random_state':[random_state]}

    # Decision Tree
    # https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decision%20tree%20classifier#sklearn.tree.DecisionTreeClassifier
    param_grid_DT = {'criterion': ['gini', 'entropy'],'splitter': ['best', 'random'],'max_depth': [1, 30],
                     'min_samples_split': [2, 50],'min_samples_leaf': [1, 50],'max_features': [None, 'auto', 'log2'],
                     'class_weight': [None, 'balanced'],
                     'random_state':[random_state]}

    # Random Forest
    # https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest#sklearn.ensemble.RandomForestClassifier
    param_grid_RF = {'n_estimators': [10, 1000],'criterion': ['gini', 'entropy'],'max_depth': [1, 30],
                     'min_samples_split': [2, 50],'min_samples_leaf': [1, 50],'max_features': [None, 'auto', 'log2'],
                     'bootstrap': [True],'oob_score': [False, True],'class_weight': [None, 'balanced'],
                     'random_state':[random_state]}
```

AutoMLPipe-BC: Phase 5 – Model - Output

```
hcc-data_example_CV0_NB_training_complete
Best trial:
  Value:  0.7253905514775081
  Params:
    penalty: l2
    dual: False
    C: 297.3042389323509
    solver: saga
    class_weight: balanced
    max_iter: 51.70191786366991
    random_state: 42
LogisticRegression(C=297.3042389323509, class_weight='balanced',
                   max_iter=51.70191786366991, random_state=42, solver='saga')
hcc-data_example_CV0_LR_training_complete
Best trial:
  Value:  0.6888292866553735
  Params:
    criterion: entropy
    splitter: best
    max_depth: 7
    min_samples_split: 33
    min_samples_leaf: 16
    max_features: auto
    class_weight: balanced
    random_state: 42
DecisionTreeClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=7, max_features='auto', min_samples_leaf=16,
                       min_samples_split=33, random_state=42)
hcc-data_example_CV0_DT_training_complete
```

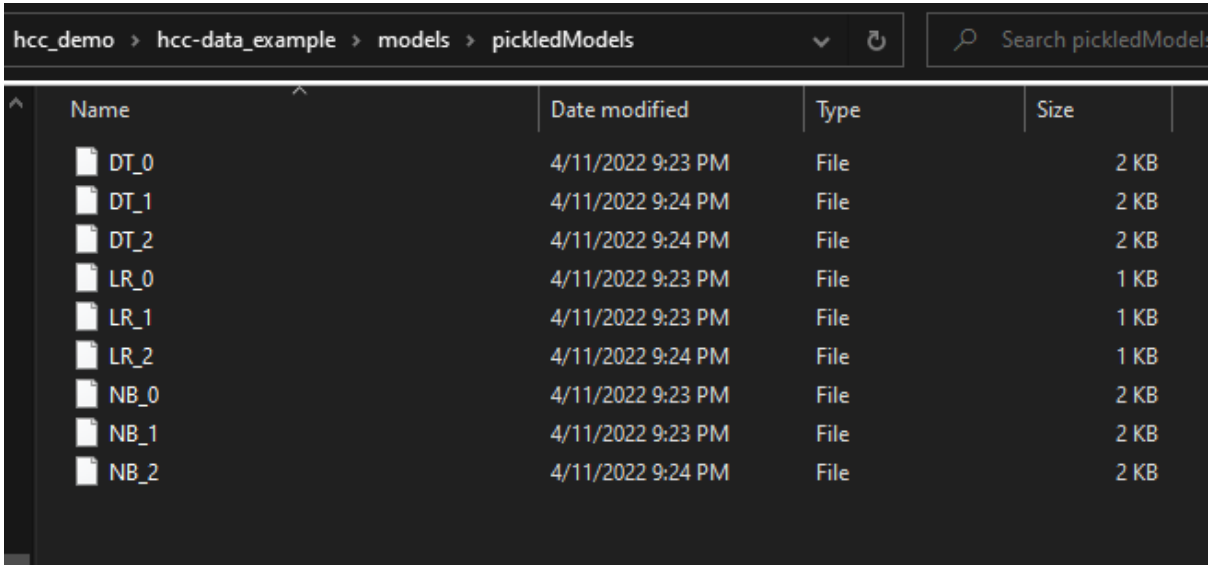
OPTUNA

Parallel Coordinate Plot

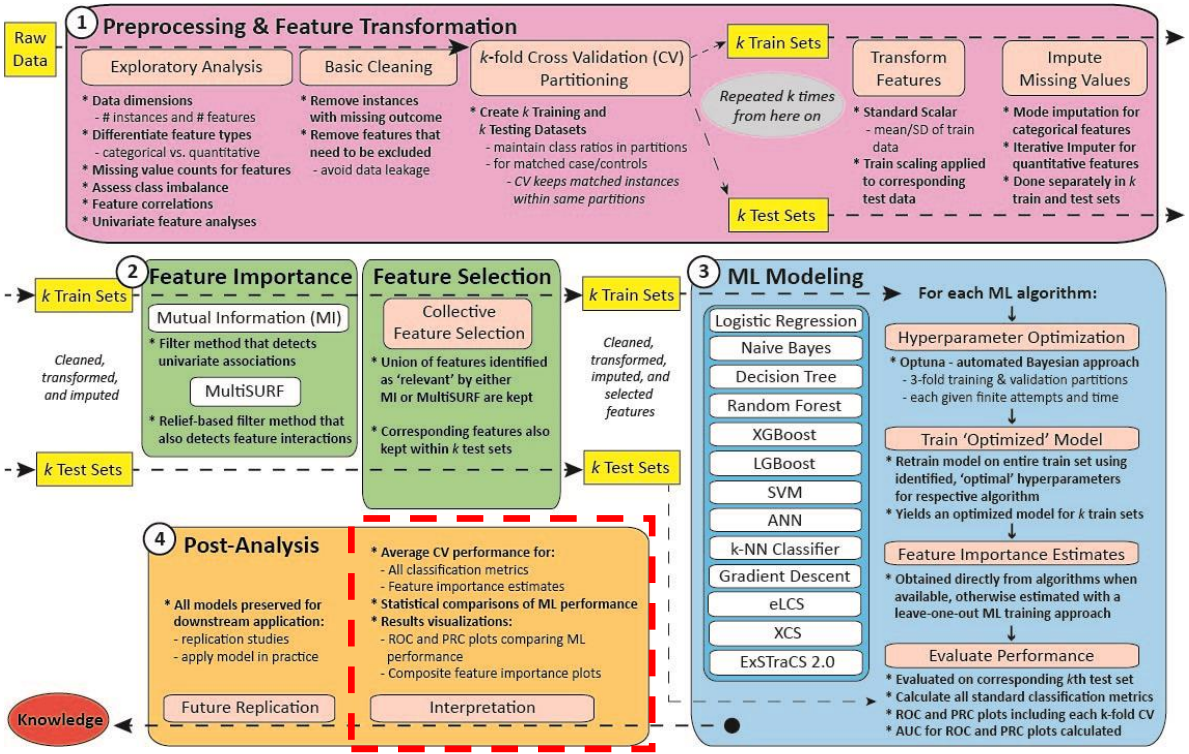


AutoMLPipe-BC: Phase 5 – Model - Output

Trained models are ‘pickled’ for storage and re-use



AutoMLPipe-BC: Phase 6 – Stats



Phase 6: Stats

AutoMLPipe-BC: Phase 6 – Stats – Run Parameters

Run Parameters for Phase 6: Statistics Summary and Figure Generation

```
[ ] plot_ROC = 'True'          # (str, True or False) Plot ROC curves individually for each algorithm including all CV results and averages
    plot_PRC = 'True'          # (str, True or False) Plot PRC curves individually for each algorithm including all CV results and averages
    plot_FI_box = 'True'       # (str, True or False) Plot box plot summaries comparing algorithms for each metric
    plot_metric_boxplots = 'True' # (str, True or False) Plot feature importance boxplots for each algorithm
```

AutoMLPipe-BC: Phase 6 – Stats – Output

DT_performance.csv

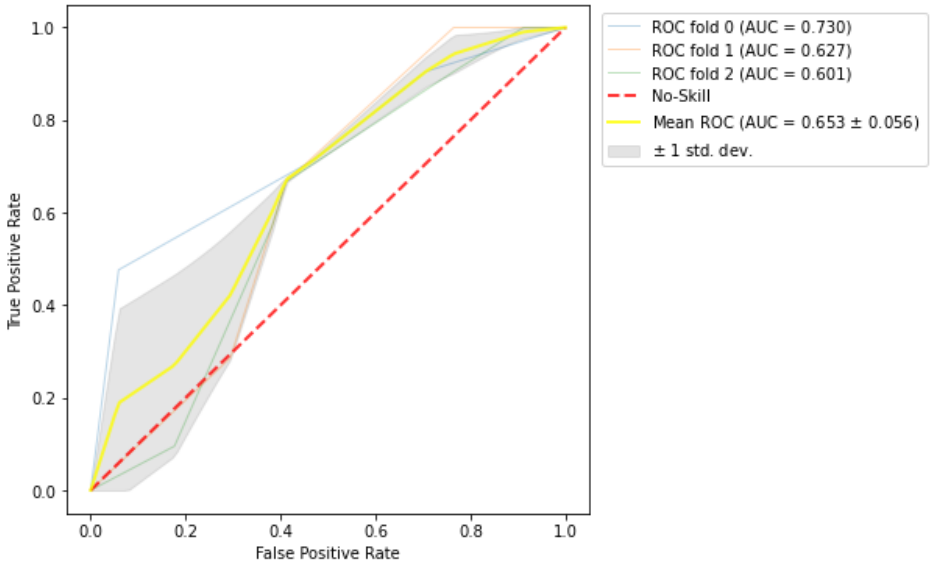
Balanced Accuracy	Accuracy	F1_Score	Sensitivity (Recall)	Specificity	Precision (PPV)	TP	TN	FP	FN	NPV	LR+	LR-	ROC_AUC	PRC_AUC	PRC_APS
0.621849	0.6	0.576923	0.714286	0.529412	0.483871	15	18	16	6	0.75	1.517857	0.539683	0.729692	0.720706	0.63256
0.627451	0.618182	0.571429	0.666667	0.588235	0.5	14	20	14	7	0.740741	1.619048	0.566667	0.627451	0.520897	0.446555
0.627451	0.618182	0.571429	0.666667	0.588235	0.5	14	20	14	7	0.740741	1.619048	0.566667	0.60084	0.429125	0.438991

Summary_performance_mean.csv

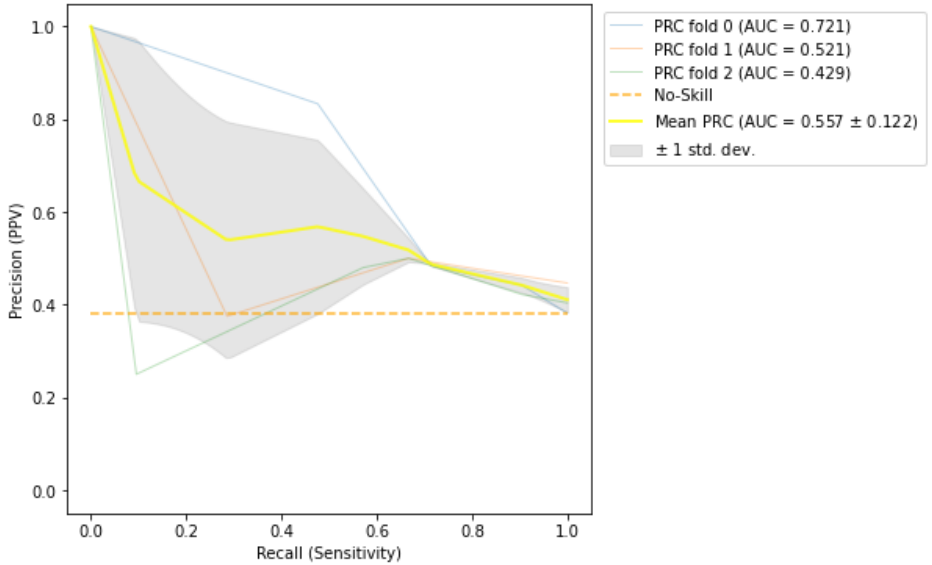
	Balanced Accuracy	Accuracy	F1_Score	Sensitivity (Recall)	Specificity	Precision (PPV)	TP	TN	FP	FN	NPV	LR+	LR-	ROC_AUC	PRC_AUC	PRC_APS
Naive Bayes	0.545051	0.557576	0.445243	0.492063	0.598039	0.578611	10.33333	20.33333	13.66667	10.66667	0.521926	3.670309	2.660253	0.676004	0.564472	0.552264
Logistic Regression	0.702148	0.70303	0.642512	0.698413	0.705882	0.595	14.66667	24	10	6.333333	0.791039	2.395863	0.428696	0.732493	0.593383	0.610367
Decision Tree	0.625584	0.612121	0.57326	0.68254	0.568627	0.494624	14.33333	19.33333	14.66667	6.666667	0.743827	1.585317	0.557672	0.652661	0.556909	0.506035

AutoMLPipe-BC: Phase 6 – Stats – Output

DT_ROC.png

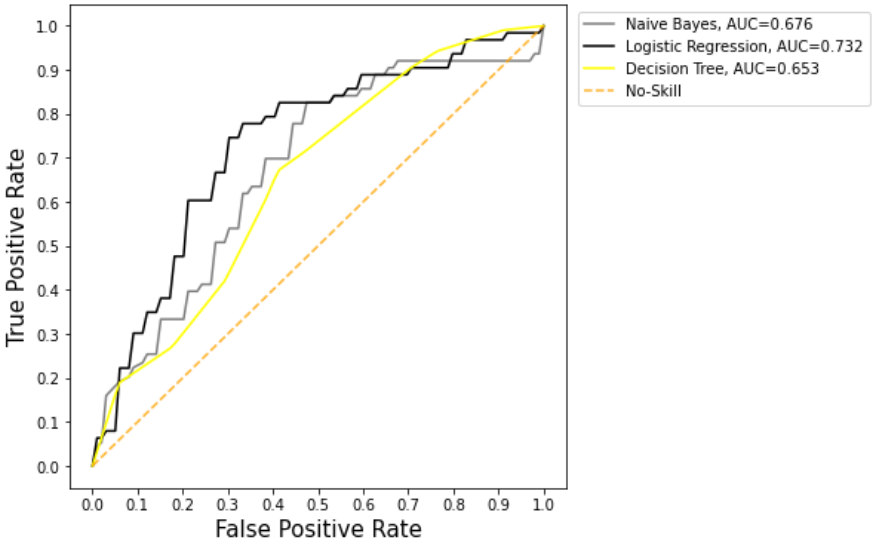


DT_PRC.png

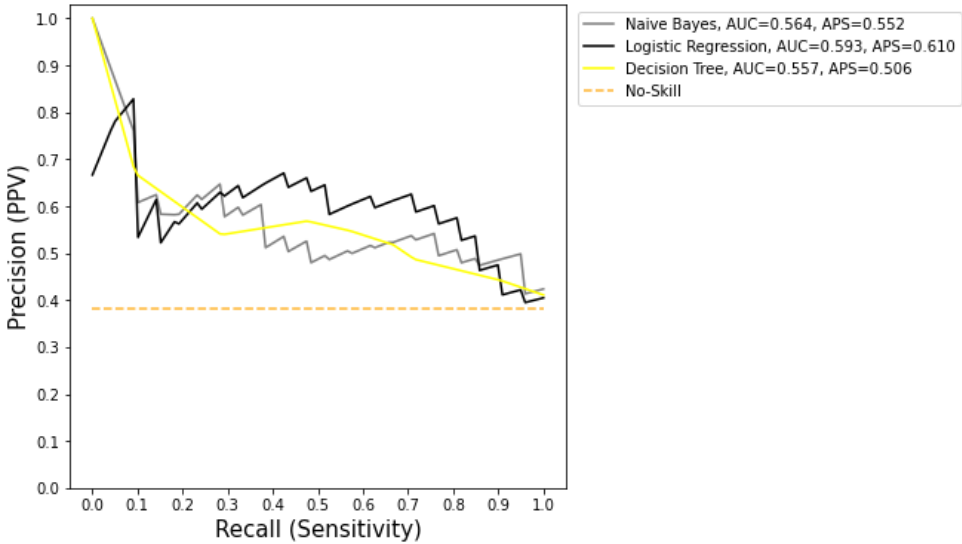


AutoMLPipe-BC: Phase 6 – Stats – Output

Summary_ROC.png

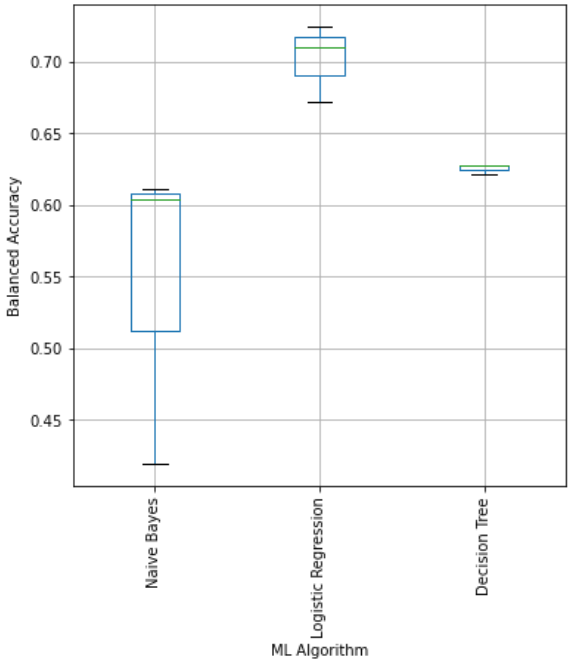


Summary_PRC.png

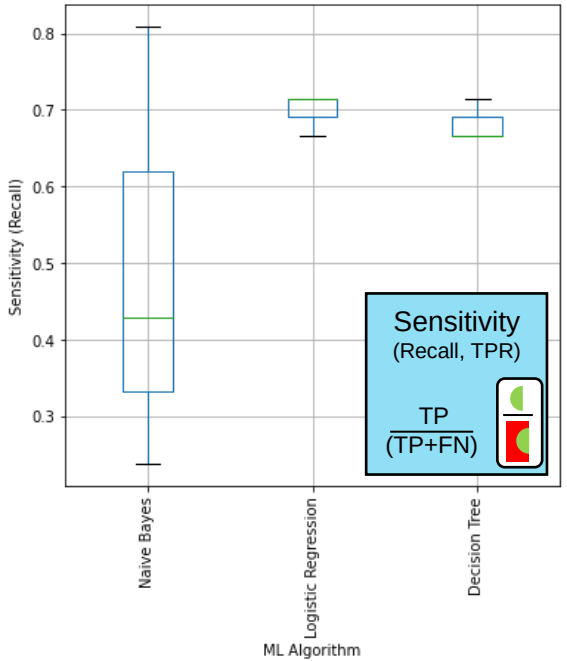


AutoMLPipe-BC: Phase 6 – Stats – Output

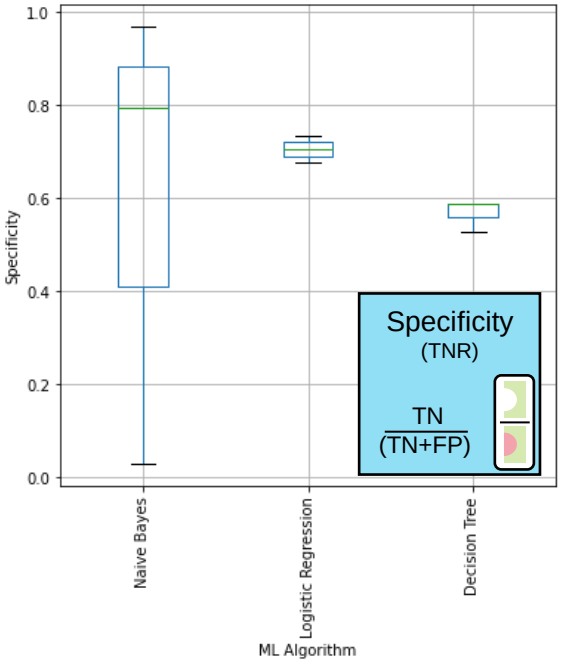
Compare_Balanced Accuracy.png



Compare_Sensitivity (Recall).png

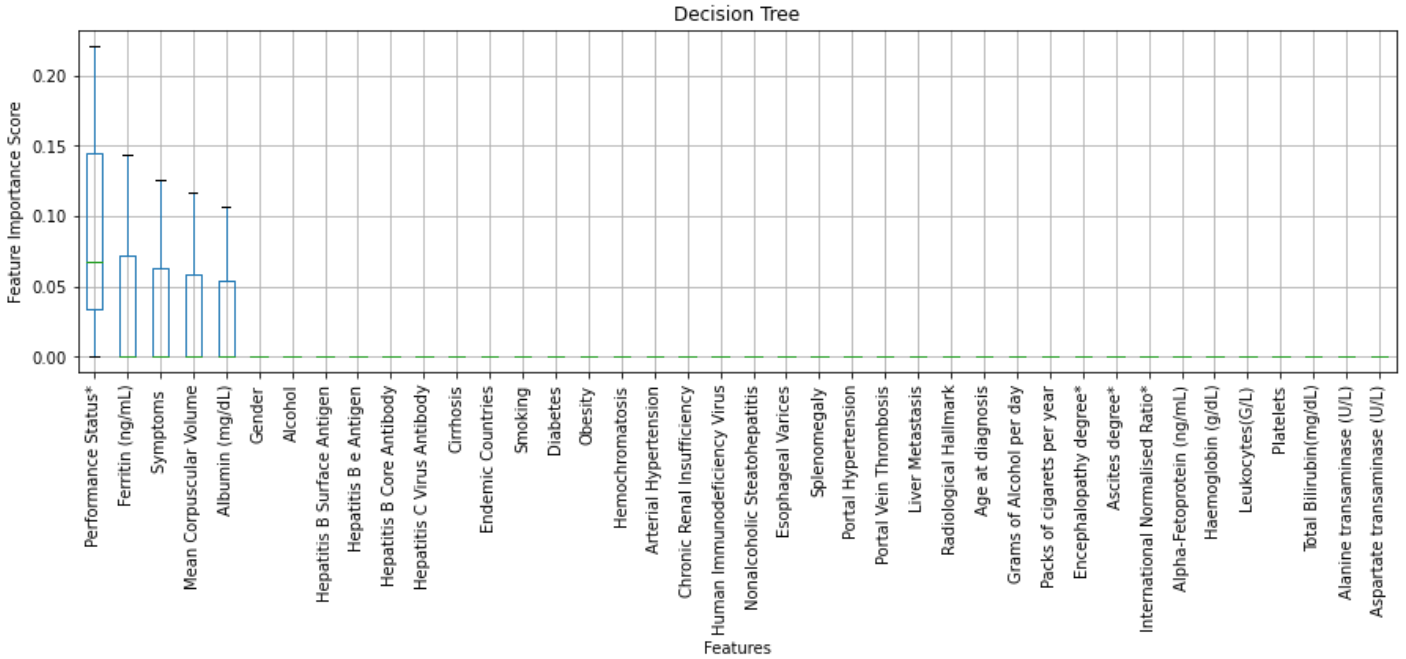


Compare_Specificity.png



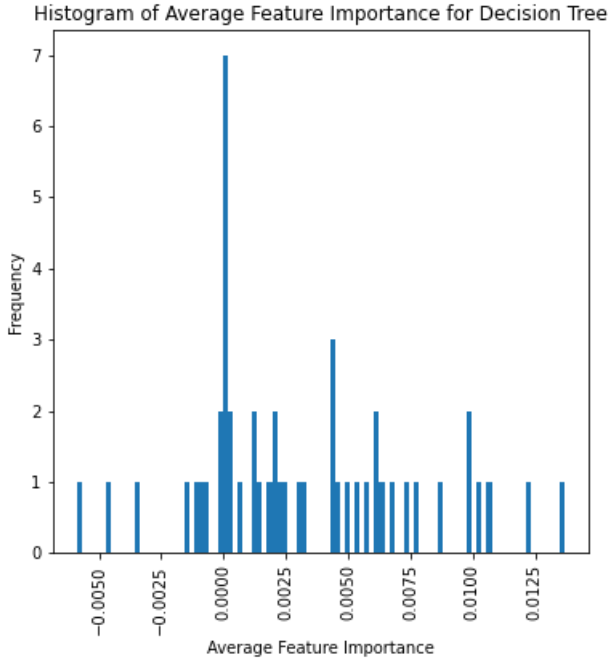
AutoMLPipe-BC: Phase 6 – Stats – Output

Decision Tree_boxplot.png



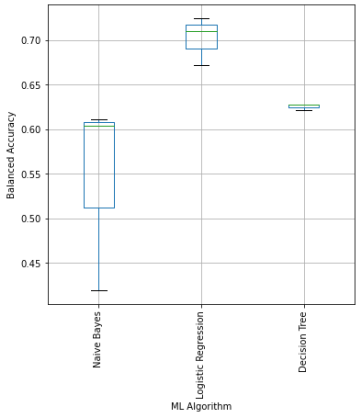
AutoMLPipe-BC: Phase 6 – Stats – Output

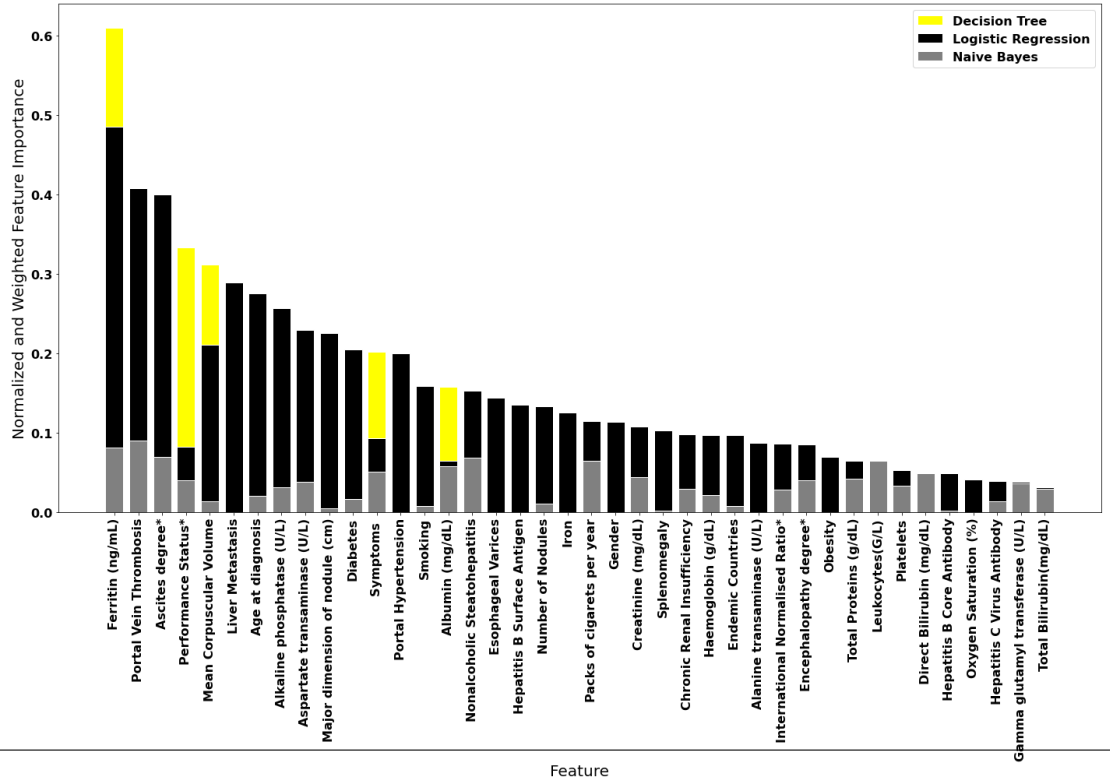
Decision Tree_histogram.png



AutoMLPipe-BC: Phase 6 – Stats – Output

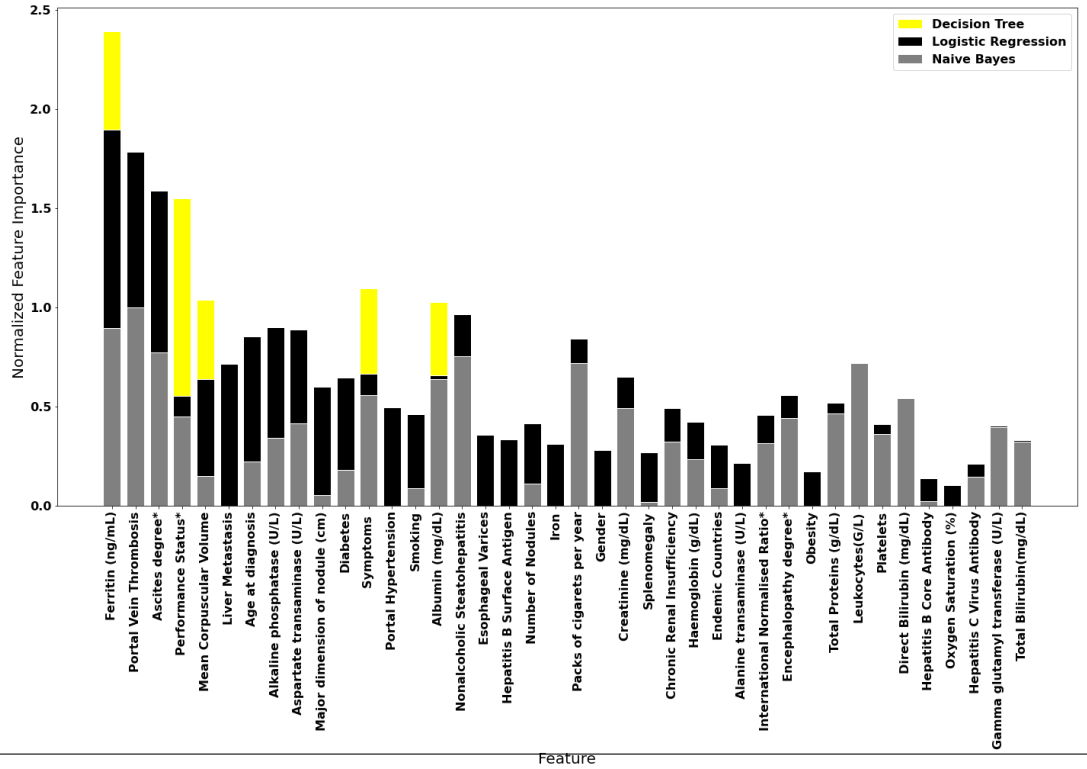
Compare_FI_Norm_Weight.png





AutoMLPipe-BC: Phase 6 – Stats – Output

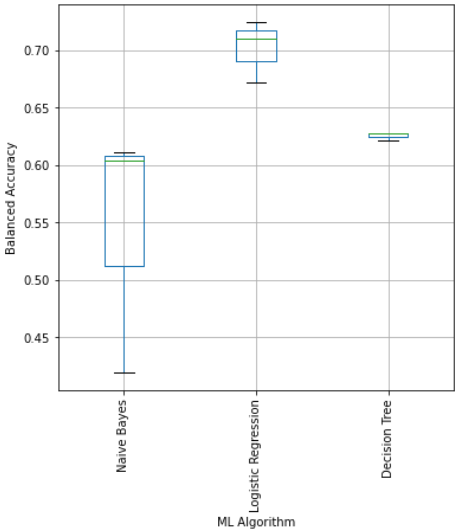
Compare_FI_Norm.png



AutoMLPipe-BC: Phase 6 – Stats – Output

KruskalWallis.csv

	Statistic	P-Value	Sig(*)
Balanced Accuracy	7.260504	0.02651	*
Accuracy	4.661064	0.097244	
F1_Score	7.260504	0.02651	*
Sensitivity (Recall)	0.857143	0.651439	
Specificity	2.420168	0.298172	
Precision (PPV)	2.778711	0.249236	
TP	0.857143	0.651439	
TN	2.420168	0.298172	
FP	2.420168	0.298172	
FN	0.857143	0.651439	
NPV	7.260504	0.02651	*
LR+	2.778711	0.249236	
LR-	7.260504	0.02651	*
ROC_AUC	1.688889	0.429796	
PRC_AUC	0.355556	0.837128	
PRC_APS	0.622222	0.732632	



MannWhitneyU_Balanced Accuracy.csv

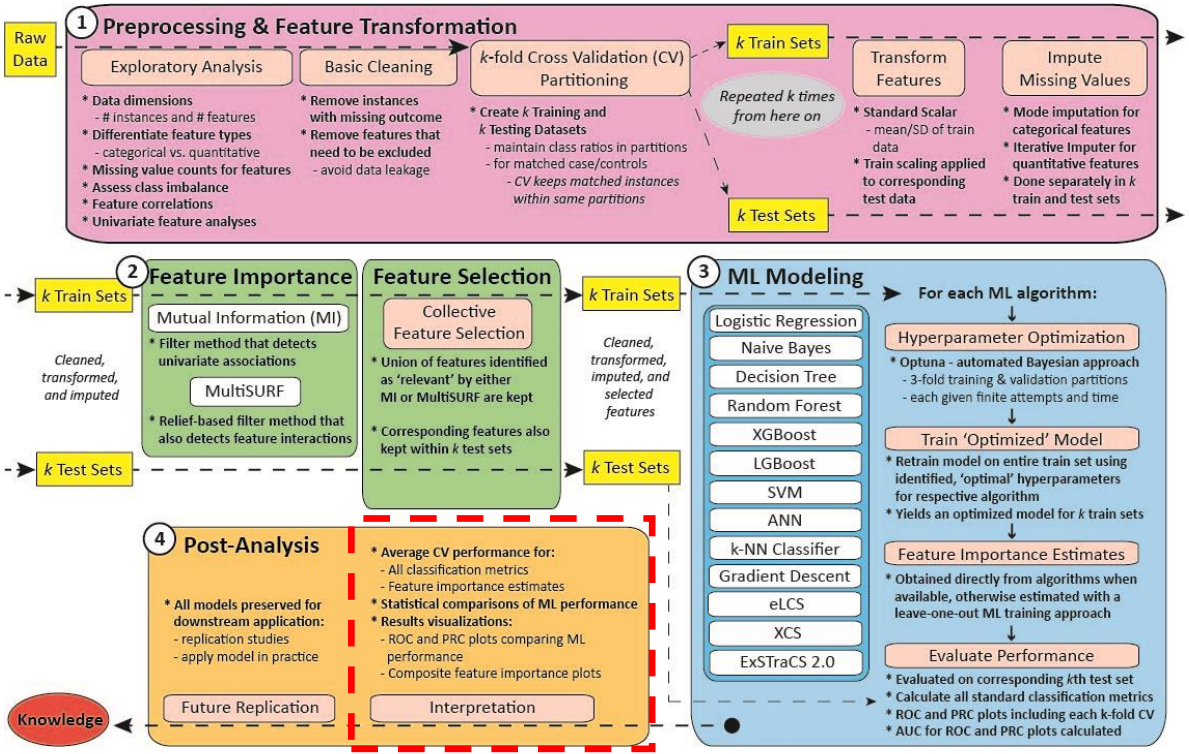
Algorithm 1	Algorithm 2	Statistic	P-Value	Sig(*)
Naive Bayes	Logistic Regression	0	0.040428	*
Naive Bayes	Decision Tree	0	0.038261	*
Logistic Regression	Decision Tree	0	0.038261	*

AutoMLPipe-BC: Phase 6 – Stats – Output

runtimes.csv

Pipeline Component	Time (sec)
Exploratory Analysis	5.457485
Preprocessing	0.26806
Mutual Information	0.295069
MultiSURF	3.038684
Feature Selection	1.584861
Naive Bayes	0.596135
Logistic Regression	8.108532
Decision Tree	4.941115
Stats Summary	15.46779

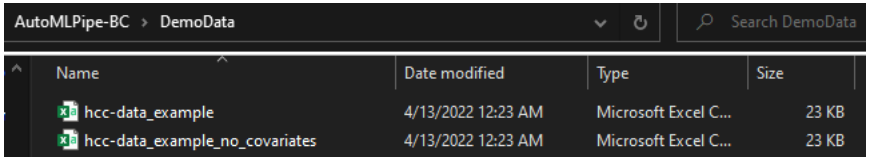
AutoMLPipe-BC: Phase 7 – Stats – Data Compare

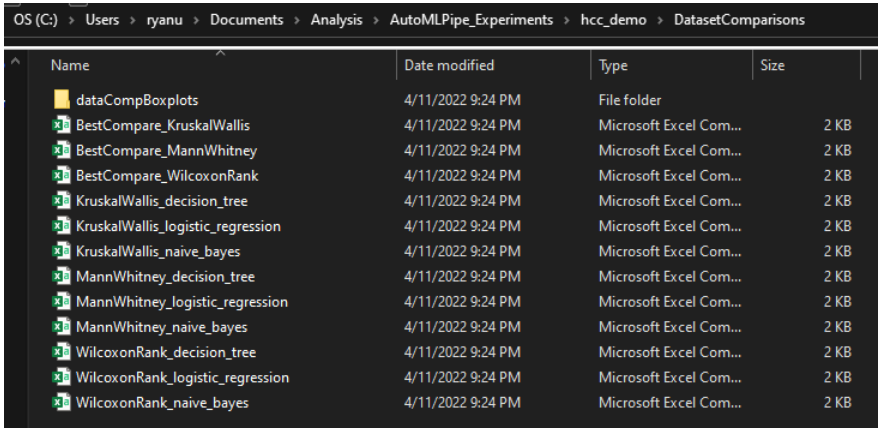


Phase 7: Data Compare

AutoMLPipe-BC: Phase 7 – Stats – Data Compare - Output

	Statistic	P-Value	Sig(*)	Mean_D1	Std_D1	Mean_D2	Std_D2
Balanced Accuracy	3.970588	0.046302	*	0.625584	0.003234	0.569094	0.008643
Accuracy	0.20202	0.653095		0.612121	0.010497	0.606061	0.063852
F1_Score	3.970588	0.046302	*	0.57326	0.003172	0.411998	0.141298
Sensitivity (Recall)	3.970588	0.046302	*	0.68254	0.027493	0.412698	0.244362
Specificity	0.441176	0.506555		0.568627	0.033962	0.72549	0.253579
Precision (PPV)	0.441176	0.506555		0.494624	0.009312	0.636508	0.315522
TP	3.970588	0.046302	*	14.33333	0.57735	8.666667	5.131601
TN	0.441176	0.506555		19.33333	1.154701	24.66667	8.621678
FP	0.441176	0.506555		14.66667	1.154701	9.333333	8.621678
FN	3.970588	0.046302	*	6.666667	0.57735	12.33333	5.131601
NPV	3.970588	0.046302	*	0.743827	0.005346	0.670106	0.014191
LR+	3.970588	0.046302	*	1.585317	0.058422	0.903319	0.790981
LR-	3.970588	0.046302	*	0.557672	0.015579	0.797792	0.051774
ROC_AUC	1.190476	0.275234		0.652661	0.068025	0.583567	0.042252
PRC_AUC	0.428571	0.512691		0.556909	0.149089	0.597772	0.013228
PRC_APS	0.047619	0.827259		0.506035	0.109639	0.464661	0.045786

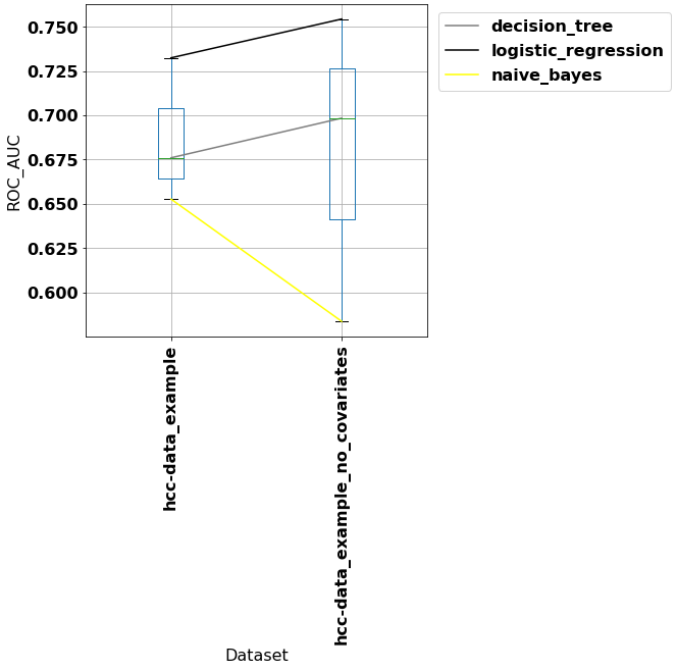




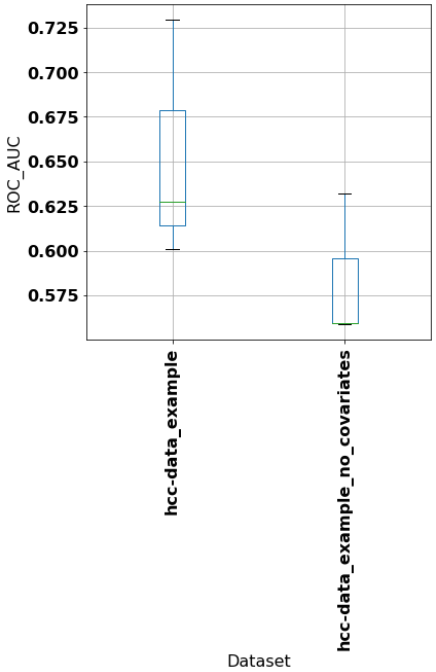
KruskalWallis_decision_tree.csv

AutoMLPipe-BC: Phase 7 – Stats – Data Compare - Output

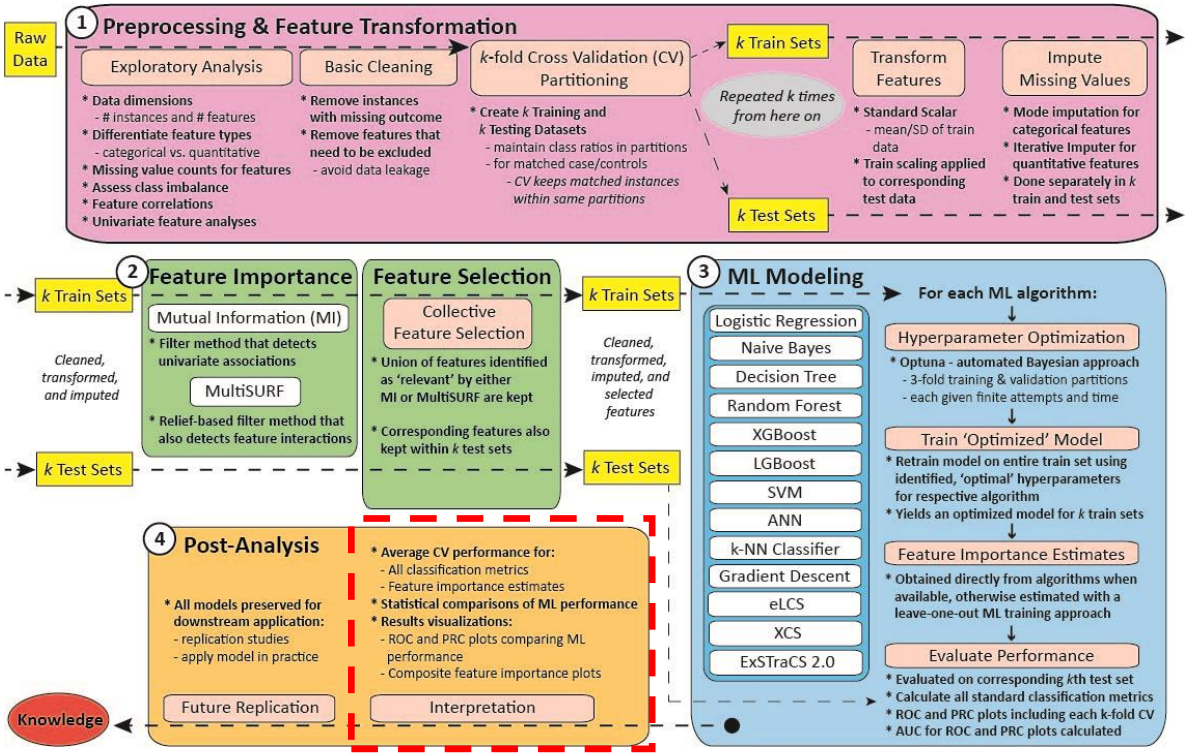
DataCompareAllModels_ROC_AUC.png



DataCompare_DT_ROC_AUC.png

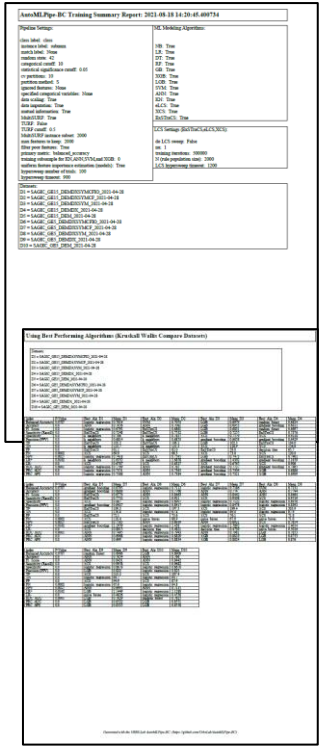


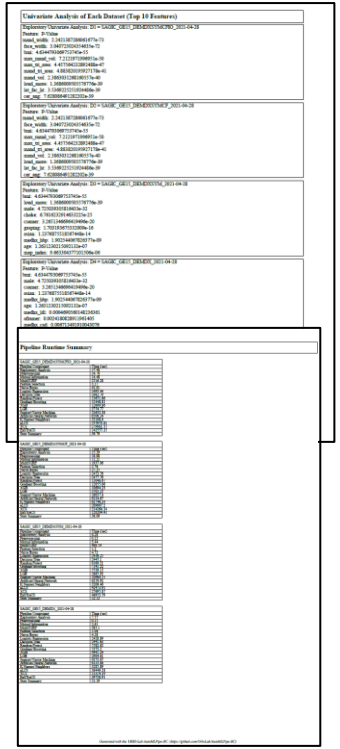
AutoMLPipe-BC: Phase 8 – PDF ReportTrain

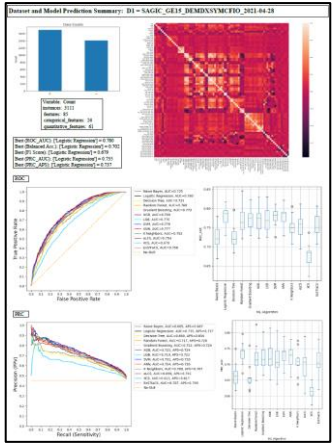


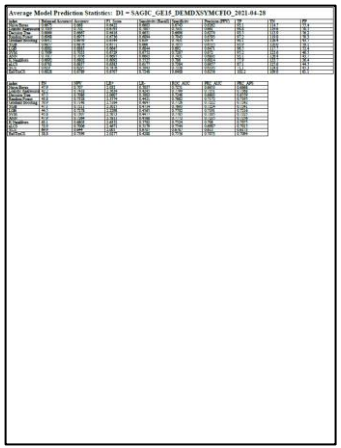
Phase 8: PDF ReportTrain

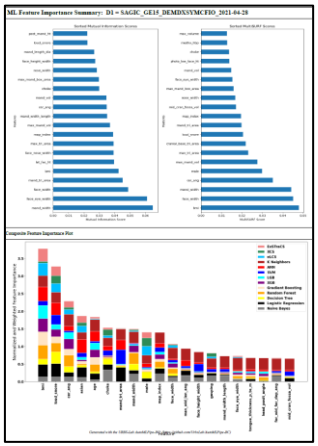
AutoMLPipe-BC: Phase 8 – PDF ReportTrain

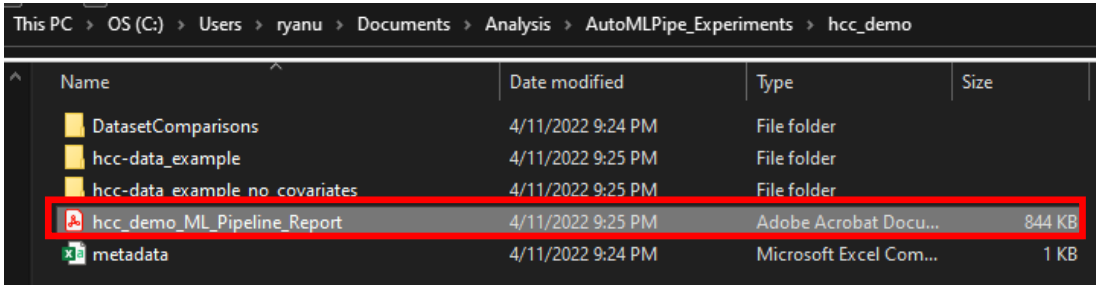




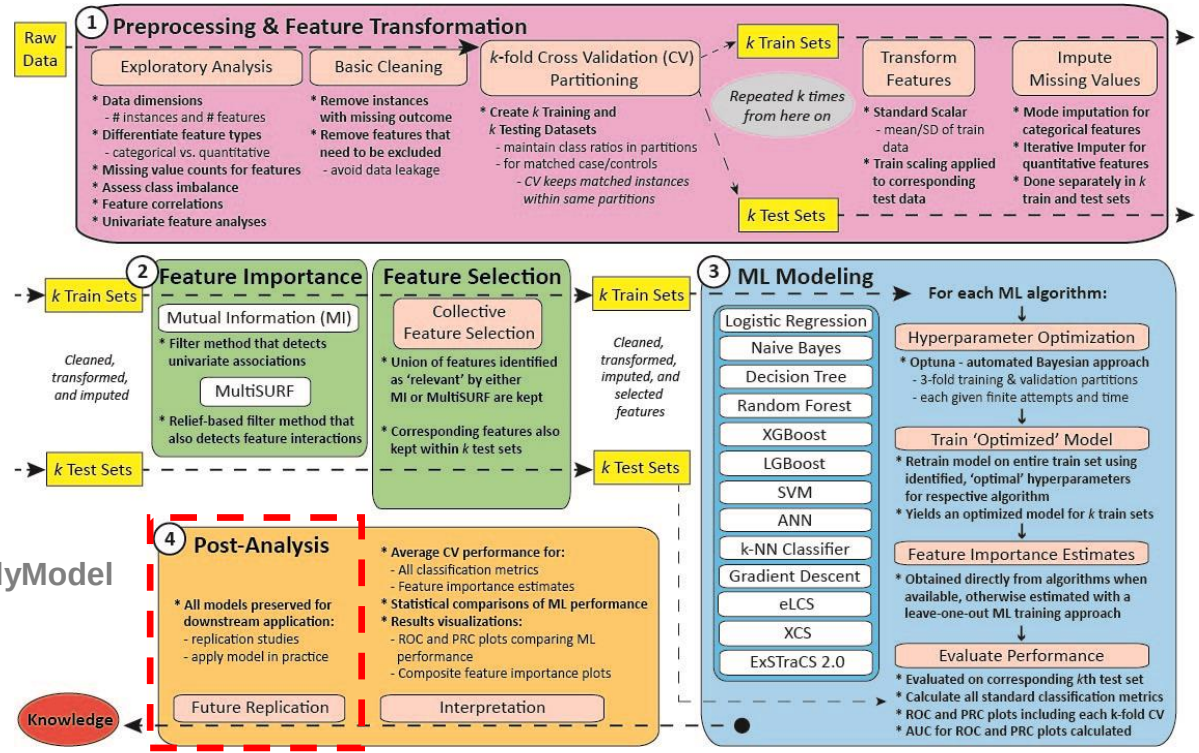








AutoMLPipe-BC: Phase 9 – ApplyModel



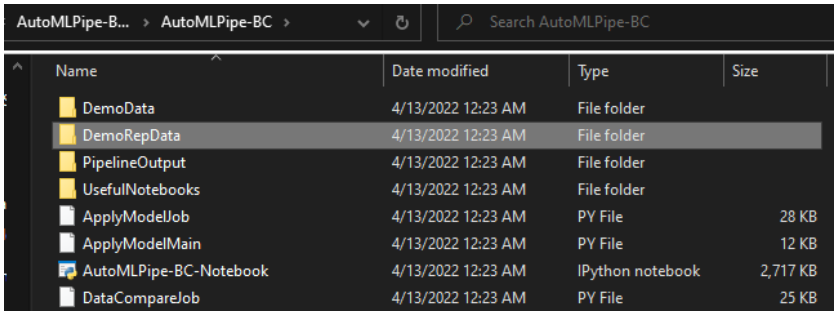
Phase 9: ApplyModel

AutoMLPipe-BC: Phase 9 – ApplyModel – Run Parameters

Run Parameters for Phase 10: Apply Models to Replication Dataset

An optional phase to apply all trained models from previous phases to a separate 'replication' dataset which will be used to evaluate models across all algorithms and CV splits. In this demo, we didn't have a separate replication dataset to use for the UCI HCC dataset evaluated. Thus here we use a copy of the original HCC dataset as a 'pretend' replication dataset to demonstrate functionality. The replication data folder can include 1 or more datasets that can be evaluated as separate replication data. The user also needs to

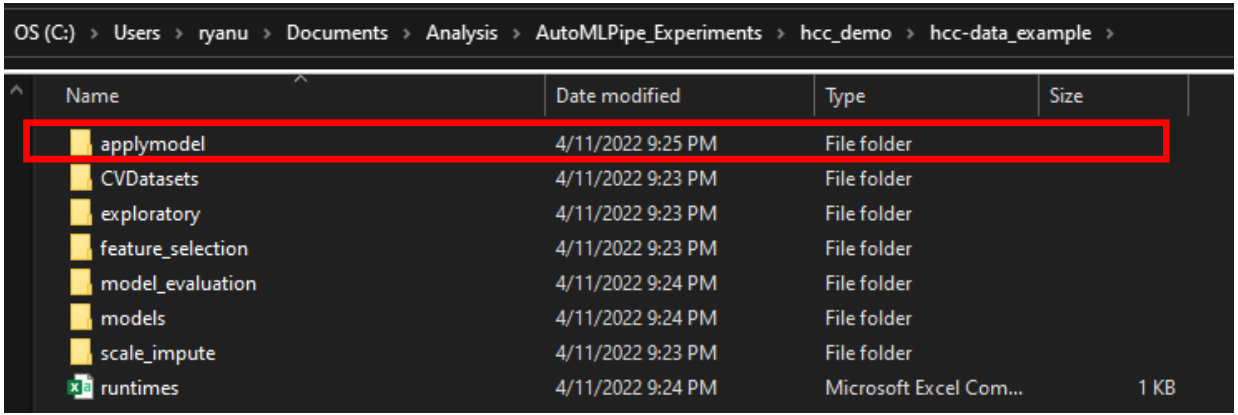
```
[ ] applyToReplication = True # (Boolean, True or False) Leave false unless you have a replication dataset handy to further evaluate
    rep_data_path = "C:/Users/ryanu/OneDrive/Documents/GitHub/AutoMLPipe-BC/DemoRepData" # (txt) Name of folder with replication Dat
    data_path_for_rep = "C:/Users/ryanu/OneDrive/Documents/GitHub/AutoMLPipe-BC/DemoData/hcc-data_example.csv" # (txt) Path and name
```



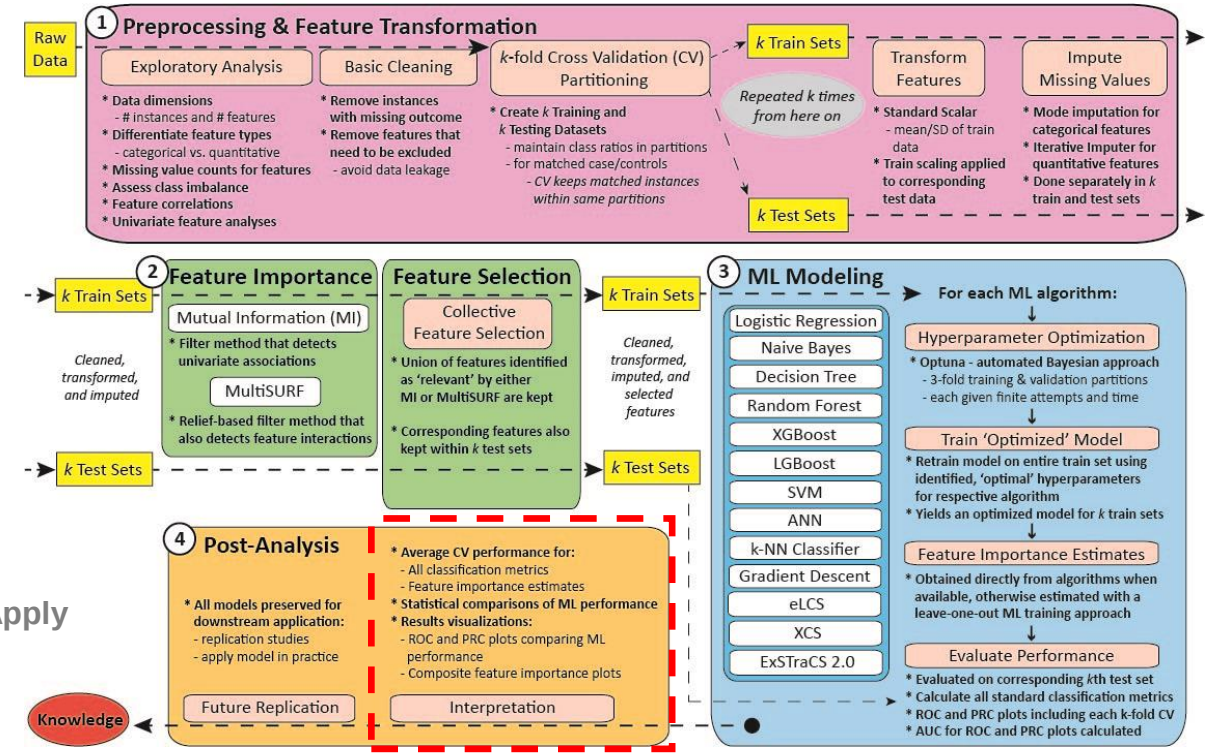
AutoMLPipe-BC: Phase 9 – ApplyModel - Output

New statistics, plots, and significance tests for…

- Exploratory analysis on target replication dataset
- All models (for each algorithm*CV partition) evaluated on same replication dataset

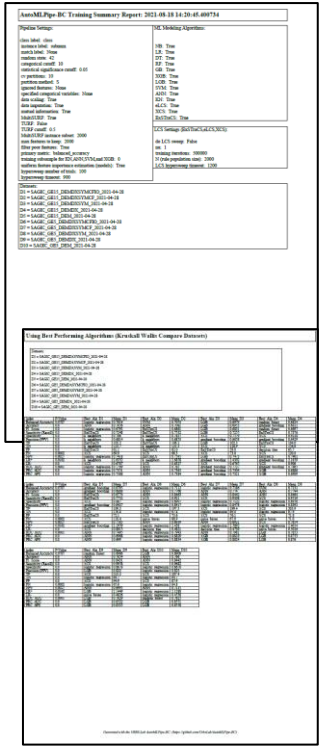


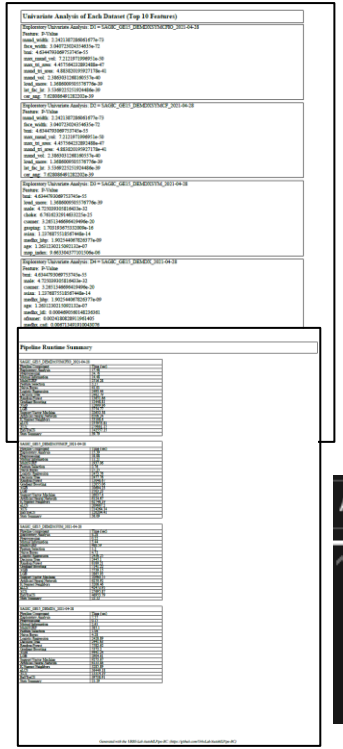
AutoMLPipe-BC: Phase 10 – PDF ReportApply

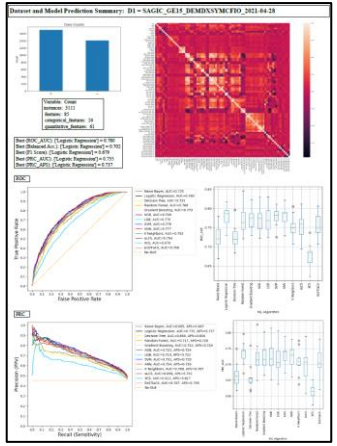


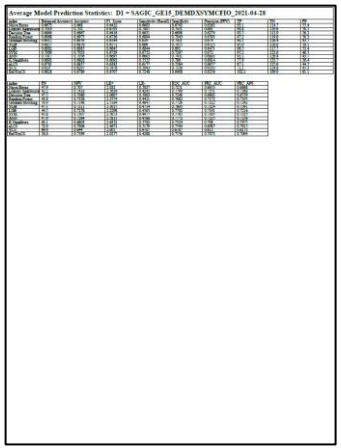
Phase 10:
PDF ReportApply

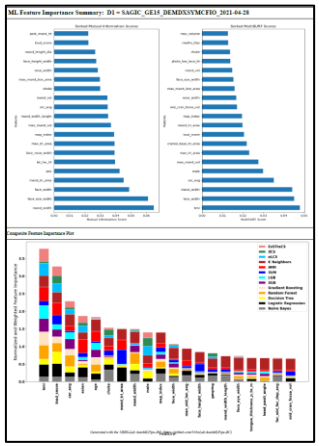
AutoMLPipe-BC: Phase 10 – PDF ReportApply

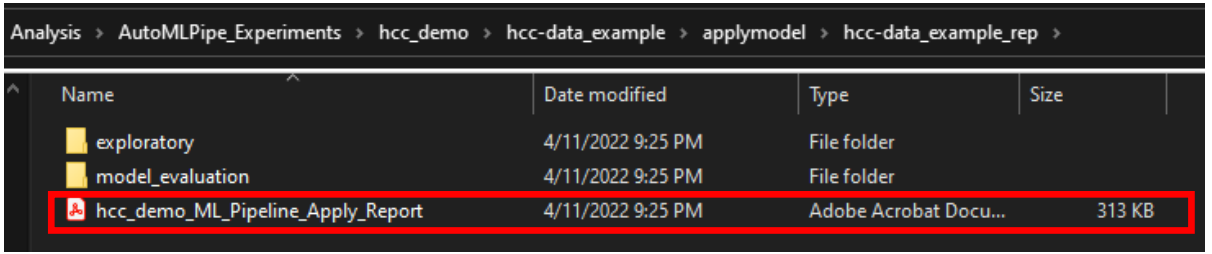




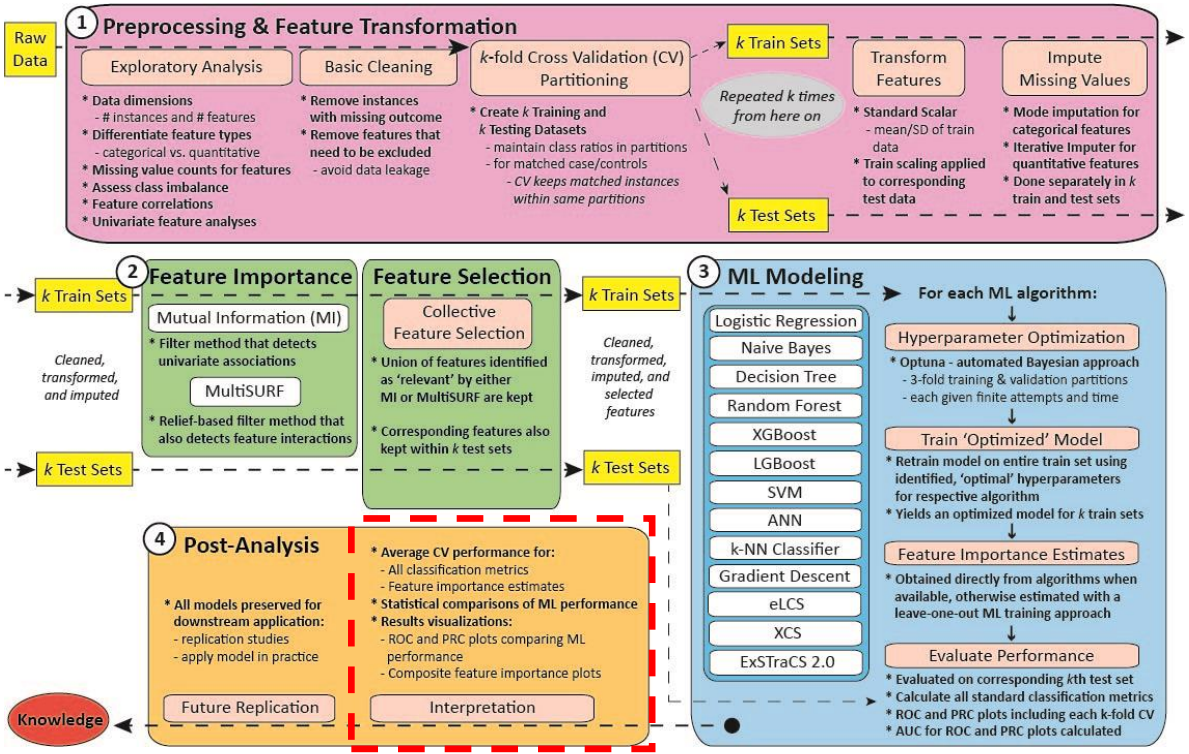








AutoMLPipe-BC: Phase 11 - FileCleanup



Phase 11: FileCleanup

AutoMLPipe-BC: Phase 11 – FileCleanup – Run Parameters

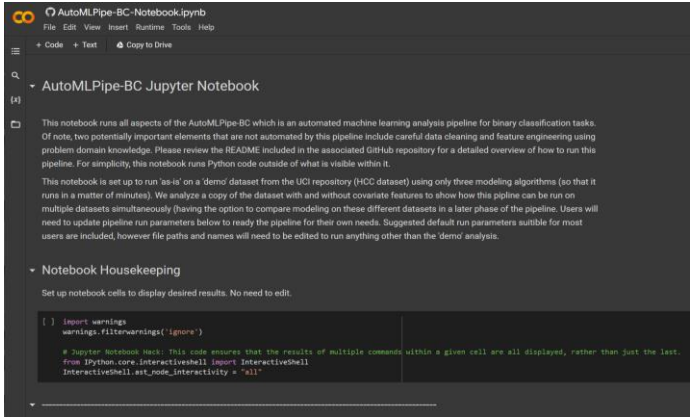
Run Parameters for Phase 11: File Cleanup

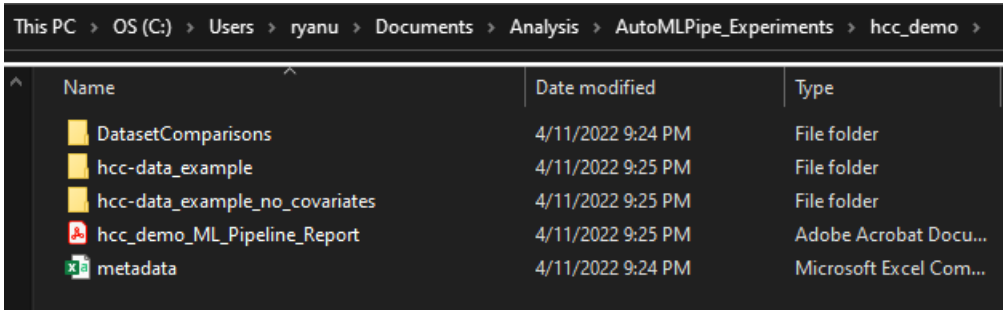
An optional phase to delete all unnecessary/temporary files generated by the pipeline.

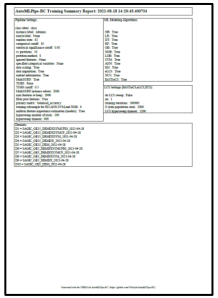
```
[ ]  del_time = 'True'  # (str, True or False) Delete individual run-time files (but save summary)
     del_oldCV = 'True' # (str, True or False) Delete any of the older versions of CV training and testing datasets not overwritten
```

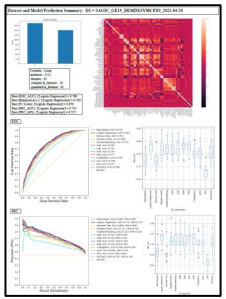
AutoMLPipe-BC: Output Summary

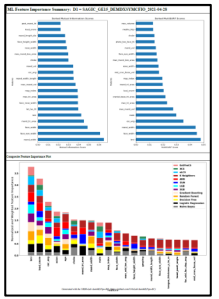
1. Notebook including results summary and visualization
2. Experiment output folder with all individual output files
3. Automatically generated PDF summary report





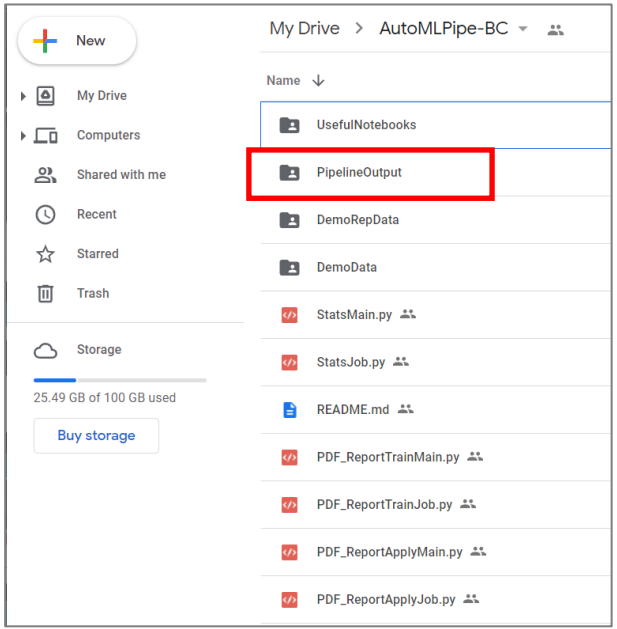






AutoMLPipe-BC: Want to View DEMO Output Files?

- https://drive.google.com/drive/folders/1h1Q82ecEbRvPYFXKcbfnJVKELGQKvT49?usp=sharing



AutoMLPipe-BC:
Run it Yourself with Google Collaboratory From Your Google Drive

Assumes: User has Google Account and Google Drive

1. Open Link (opens shared AutoMLPipe-BC folder):
   - https://drive.google.com/drive/folders/1h1Q82ecEbRvPYFXKcbfnJVKELGQKvT49?usp=sharing
2. Right click on folder ‘AutoMLPipe-BC’ at top of screen and select ‘Download’
   - Note: Folder is downloaded in zip format
3. Navigate to your download folder, right click on zipped file and choose ‘Extract all’
   - Note, file likely named something like ‘AutoMLPipe-BC-20220413T072201Z-001.zip’
4. Navigate to ‘My Drive’ base folder in your Google Drive, and copy the extracted folder called ‘AutoMLPipe-BC’ to ‘My Drive’
5. Navigate into the newly copied ‘AutoMLPipe-BC’ folder
6. Open ‘PipelineOutput’ folder and delete the subfolder ‘hcc_demo’ (You will need to do this anytime you want to re-run the demo of AutoMLPipe-BC)

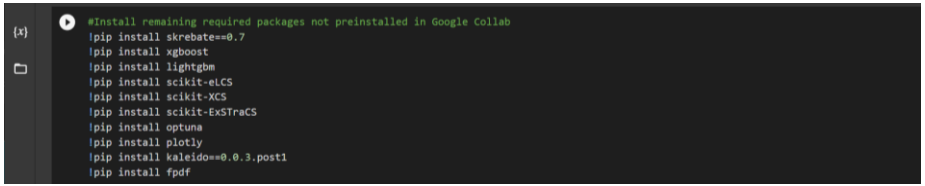
AutoMLPipe-BC:
Run it Yourself with Google Collaboratory From Your Google Drive

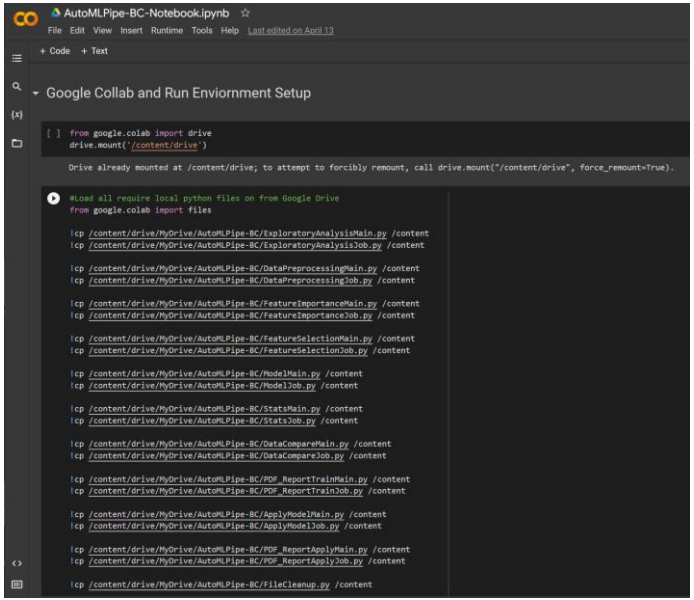
7. Navigate back to ‘AutoMLPipe-BC’ folder
8. Right click on ‘AutoMLPipe-BC-Notebook.ipynb’ (or double click if GC installed)
   - Open with… (and pick)
     - ‘Google Colaboratory’, (if already installed)
     - Or ‘Connect more apps’, then search for and install ‘Google Colaboratory’
9. At top of Google Colaboratory Notebook, open ‘Runtime’ menu and select ‘Run all’ (to run all code cells of the notebook)
   - Google Colaboratory will then try to mount the relevant google drive file (necessary to run the pipeline). Select your google drive account, and allow all permissions to mount drive.
10. Allow notebook to run to completion
11. In Google Drive navigate to ‘AutoMLPipe-BC/PipelineOutput/hcc_demo’ to view or download all pipeline output files
12. Enjoy!

AutoMLPipe-BC:
Run it Yourself with Google Collaboratory From Your Google Drive

Some extra stuff at the start of this notebook…

- Make files on your google drive accessible
- Loads in pipeline python files
- Installs other required packages





AutoMLPipe-BC:
Run it Yourself with Google Collaboratory From Your Google Drive

Phase 1: Exploratory Analysis

- Ensure output path correctly points to a location on your google drive

Mandatory Run Parameters for Pipeline

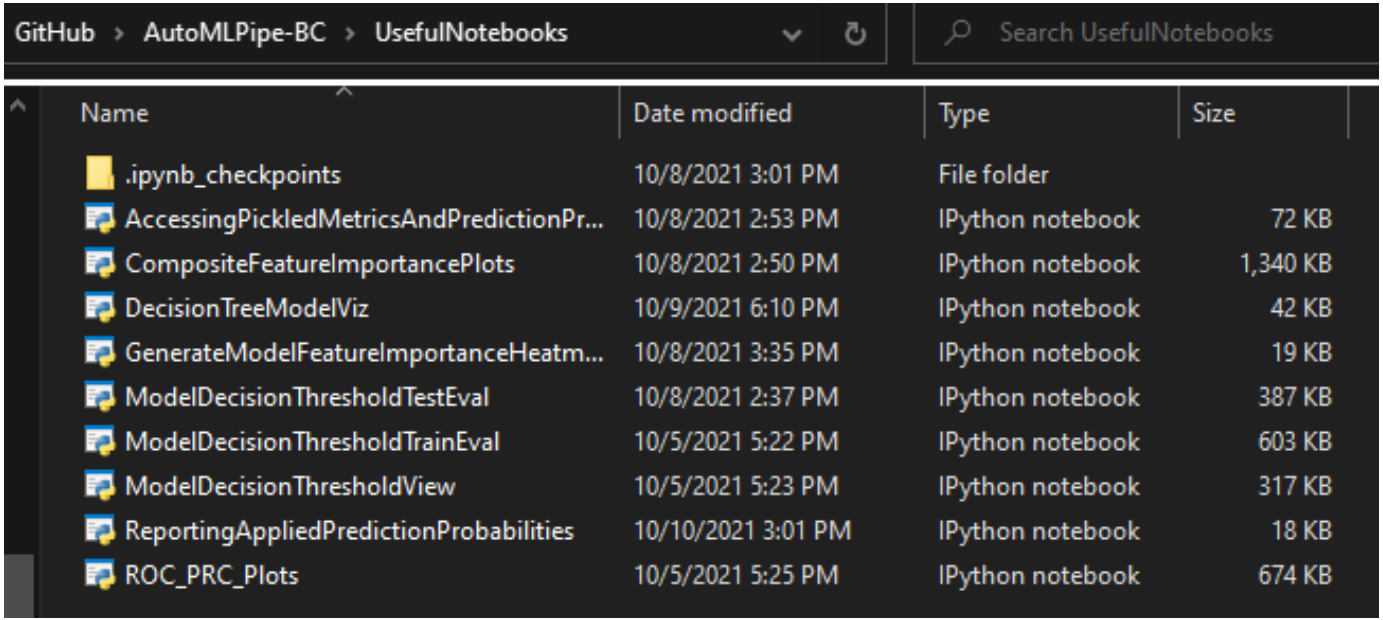
```
[ ]  demo_run = True #Leave true to run the local demo dataset (without specifying any datapaths), make False to specify a different data folder

     #Target dataset folder path(must include one or more .txt or .csv datasets)
     data_path = "C:/Users/ryanu/OneDrive/Documents/GitHub/AutoMLPipe-BC/DemoData" # (str) Demontration Data Path Folder (Ignored when demo_run =

     #Output foder path: where to save pipeline outputs (must be updated for a given user)
     output_path = '/content/drive/MyDrive/AutoMLPipe-BC/PipelineOutput' # (str) Demonstration Ouput Path Folder (Specific to Google Collab)

     #Unique experiment name - folder created for this analysis within output folder path
     experiment_name = 'hcc_demo'  # (str) Demontration Experiment Name

     # Data Labels
     class_label = 'Class' # (str) i.e. class outcome column label
     instance_label = 'InstanceID' # (str) If data includes instance labels, given respective column name here, otherwise put 'None'

     #Option to manually specify feature names to leave out of analysis, or which to treat as categorical (without using built in variable type d
     ignore_features = [] # list of column names (given as string values) to exclude from the analysis (only insert column names if needed, other
     categorical_feature_headers = [] # empty list for 'auto-detect' otherwise list feature names (given as string values) to be treated as categ
```

AutoMLPipe-BC: The way it was intended…

- **Detailed README**
  - Accessible to use for those with very basic Python experience
- **Run**
  - Local (serially)
    - Jupyter notebook
    - Command line
  - Computing Cluster (In Parallel)
- **PostHoc Jupyter Notebooks:**
  - Regenerate figures
  - Evaluate models using alternative decision thresholds
  - Show decision tree models

https://github.com/UrbsLab/AutoMLPipe-BC

AutoMLPipe-BC: Post-Hoc Jupyter Notebooks



AutoMLPipe-BC: Future Work

1. **Expand to multiclass and regression outcomes**
2. **Enhance modularity to make adding new algorithm options easy**
3. **Provide support for use on cloud computing platforms**
4. **Expand to include deep learning modeling options**
5. **Expand to include model explanation strategies (Shapley values/LIME)**

Want to Learn More?

Computational Biomedicine Workshop

What is Machine Learning? Demystifying Methods and Application

Tuesday, May 24, 2022
9:00 a.m. – 4:00 p.m.
Room NTC-B

Machine Learning (ML) is arguably one of the most exciting, rapidly changing, and easily misused/misunderstood fields of study today. The objective of this workshop is to offer a gentle introduction to ML in the broader context of biomedical data science. No experience with computer science or coding is necessary. This workshop will employ lecture, Q&A discussion, demonstration, and take-home resources to cover topics such as:

- The relationship between machine learning, statistics, and artificial intelligence
- Challenges in biomedical/clinical data analysis

Instructor:
Ryan J. Urbanowicz, PhD
Research Scientist II
(Assistant Professor Title Pending)
Department of Computational Biomedicine
Cedars-Sinai Medical Center

Questions?

Recommended Online Resources To Learn More About ML

- https://vas3k.com/blog/machine_learning/
- https://christophm.github.io/interpretable-ml-book/terminology.html
- https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/
- https://machinelearningmastery.com/data-preparation-techniques-for-machine-learning/
- https://www.analyticsvidhya.com/blog/2021/04/steps-to-complete-a-machine-learning-project/
- https://builtin.com/data-science/step-step-explanation-principal-component-analysis
- https://medium.datadriveninvestor.com/how-to-choose-the-right-machine-learning-algorithm-81449ee626b
- https://www.theinsaneapp.com/2021/02/types-of-machine-learning-algorithms.html
- https://www.tutorialspoint.com/data_mining/index.htm
- https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm
- https://www.analyticsvidhya.com/blog/2021/05/introduction-to-supervised-deep-learning-algorithms/
- https://www.analyticsvidhya.com/blog/2021/03/gradient-boosting-machine-for-data-scientists/
- https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3