Kaggle Higgs Boson

Machine Learning Challenge

Team Demibots

Bernard Ong

Nanda Rajarathinam

Venkatesh Subramanian

Deepak Khurana

V6 - August 2016

NYC Data Science Academy

Agenda

- **Introduction**
- **Strategy and Approach**
- **Exploratory Data Analysis and Feature Engineering**
- **Training and Cross-Validation**
- **Results and Findings**
- **Conclusion**

Introduction

Challenge Definition

Success Metrics

Challenge and Objectives

- Apply Machine Language to predict CERN's simulated particle collision events as either a Higgs Boson signal (s) or background (b) noise

- Learn how to strategize, optimize, and fine-tune models, algorithms, and parameters to achieve the best signal prediction possible

- Gain tactical working experience differentiating theory from actual practice

- Develop new skills in model ensembles and stacking techniques

Success Metrics

- **Achieve the highest Approximate Median Significance (AMS) Score in the Kaggle Private Leaderboard**

- **Achieve the Top 25% position placement in the Kaggle Private Leaderboard**

Strategy and Approach

Process
Feature Analysis and Engineering
Technology Stack
Models, Ensembles, and Stacking

Process

- **The machine learns, while we also learn how to learn**
  - Agile "chunking" process applied towards machine learning
  - Move fast, fail fast, wash-rinse-repeat = then harvest good results + detailed logs
- **Start with the basics to get the patterns and trends**
  - Treat data like it's alive, understand it's health and heartbeat
  - Get a feel of how the data behaves and reacts to changes
  - Increase complexity as the accuracy improves
- **Develop a solid build::create::train::evaluate::predict::score pipeline**
  - Make the iterative process fast, repeatable, and reliable
  - Agile "kanban" process applied towards multiple parallel production lines
  - Build in feature analysis and engineering to complement the development process

Feature Analysis & Engineering

- **Missingness Analysis and Imputation**
- **Correlation Charts and Multicollinearity**
- **Principal Component Analysis**
- **Density Charts**
- **5-PHI Fields**
- **Jet Fields**

Technology Stack

- **Python**
  - **For all modeling and algorithmic related work**
  - **Numpy, Pandas & Matplotlib.pyplot**
- **R**
  - **For all Exploratory Discovery Analysis (EDA) work and plots**
- **Key Python Libraries**
  - **Scikit Learn (Linear_Model.LogisticRegression, svm, DecisionTreeClassifier, Naive_Bayes.GaussianNB)**
  - **Scikit Learn Ensemble (GradientBoostingClassifier, RandomForestClassifier, AdaBoostClassifier)**
- **Special Libraries**
  - **XGBoost**
  - **Neural Network**

Models, Ensembles, Stacking

- **Naïve Bayes**
- **Logistic Regression**
- **Support Vector Machines**
- **Decision Tree, Extra Trees, Random Forest**
- **Gradient Boosting, AdaBoost, XGBoost**
- **Neural Networking**

EDA, Feature Analysis and Engineering
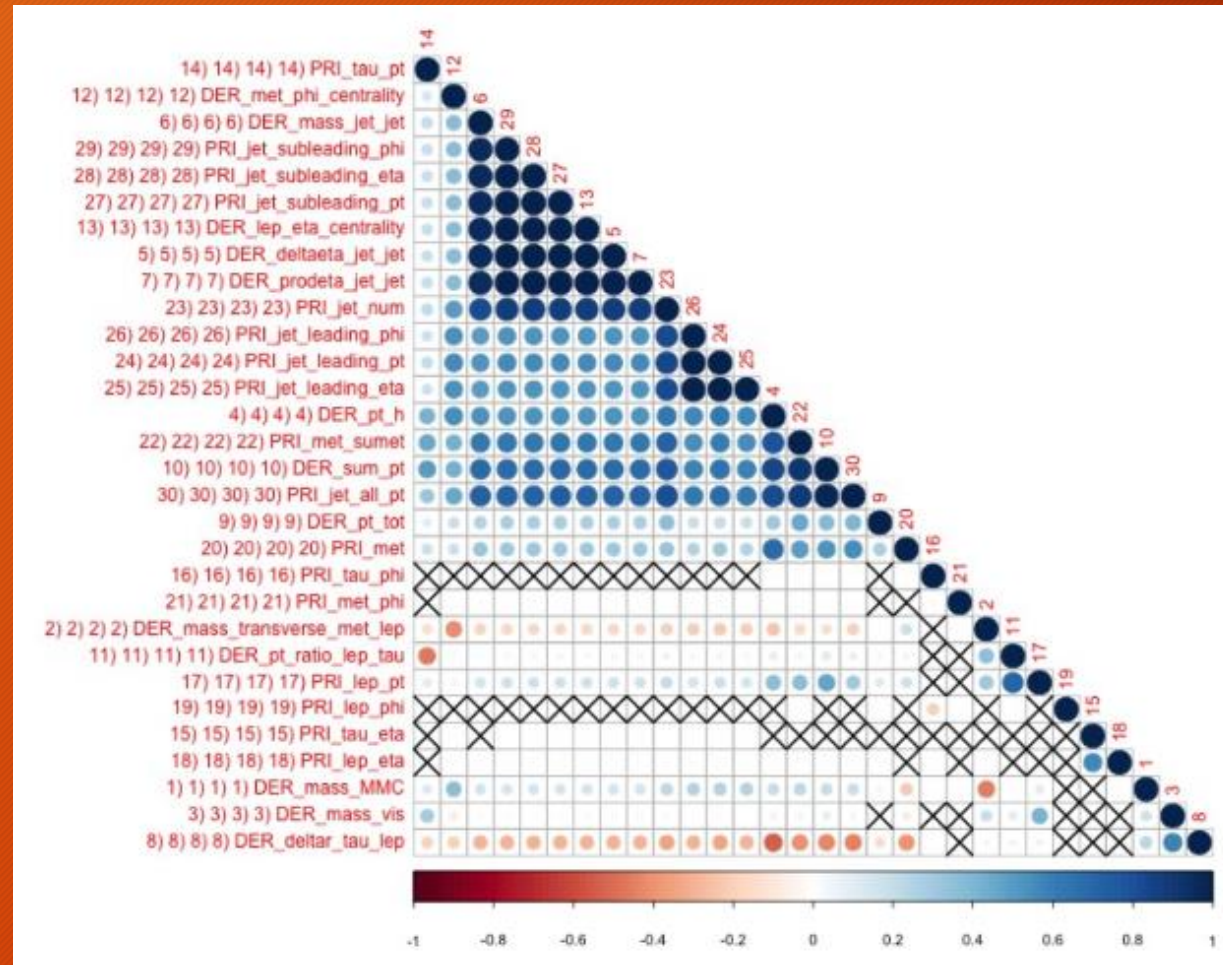
PCA Results

Correlation Results

Histograms

Imputation on "Missing" Data

Variable Importance Plot
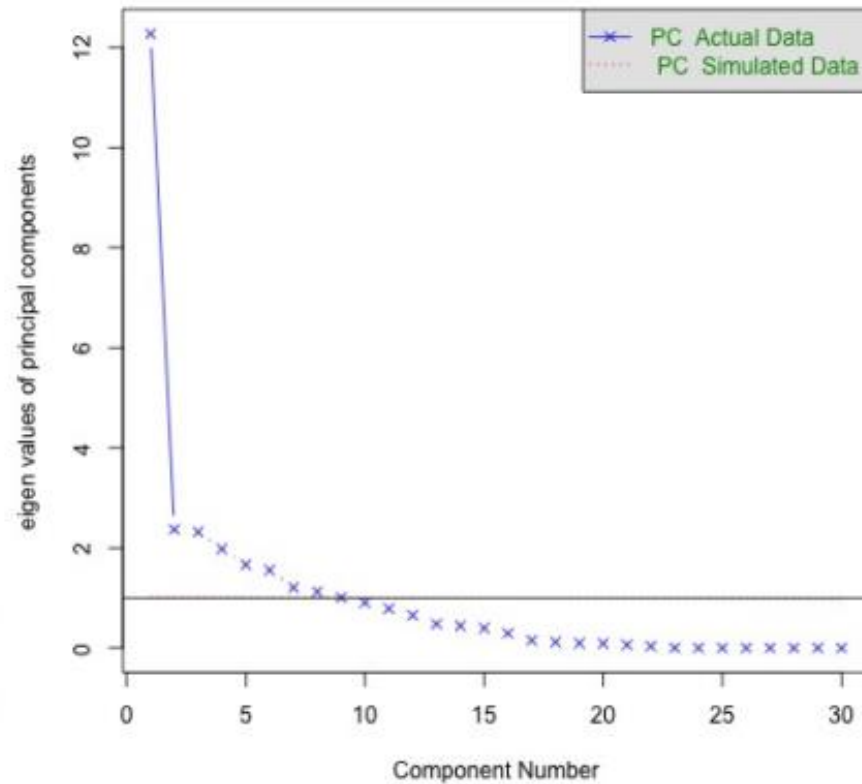
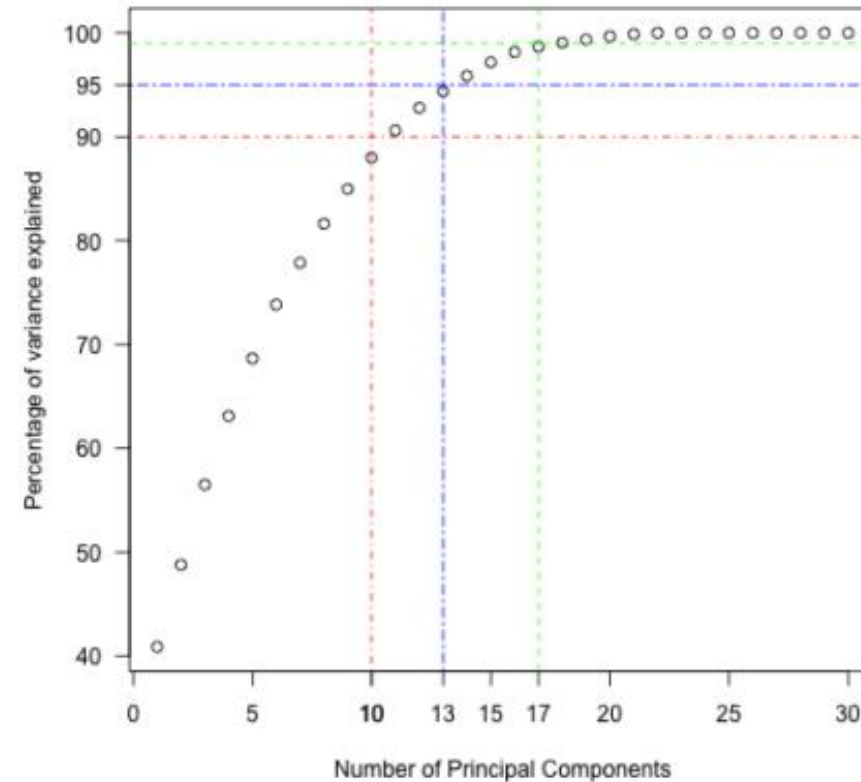Missing Values Plots

Features Correlation Plot

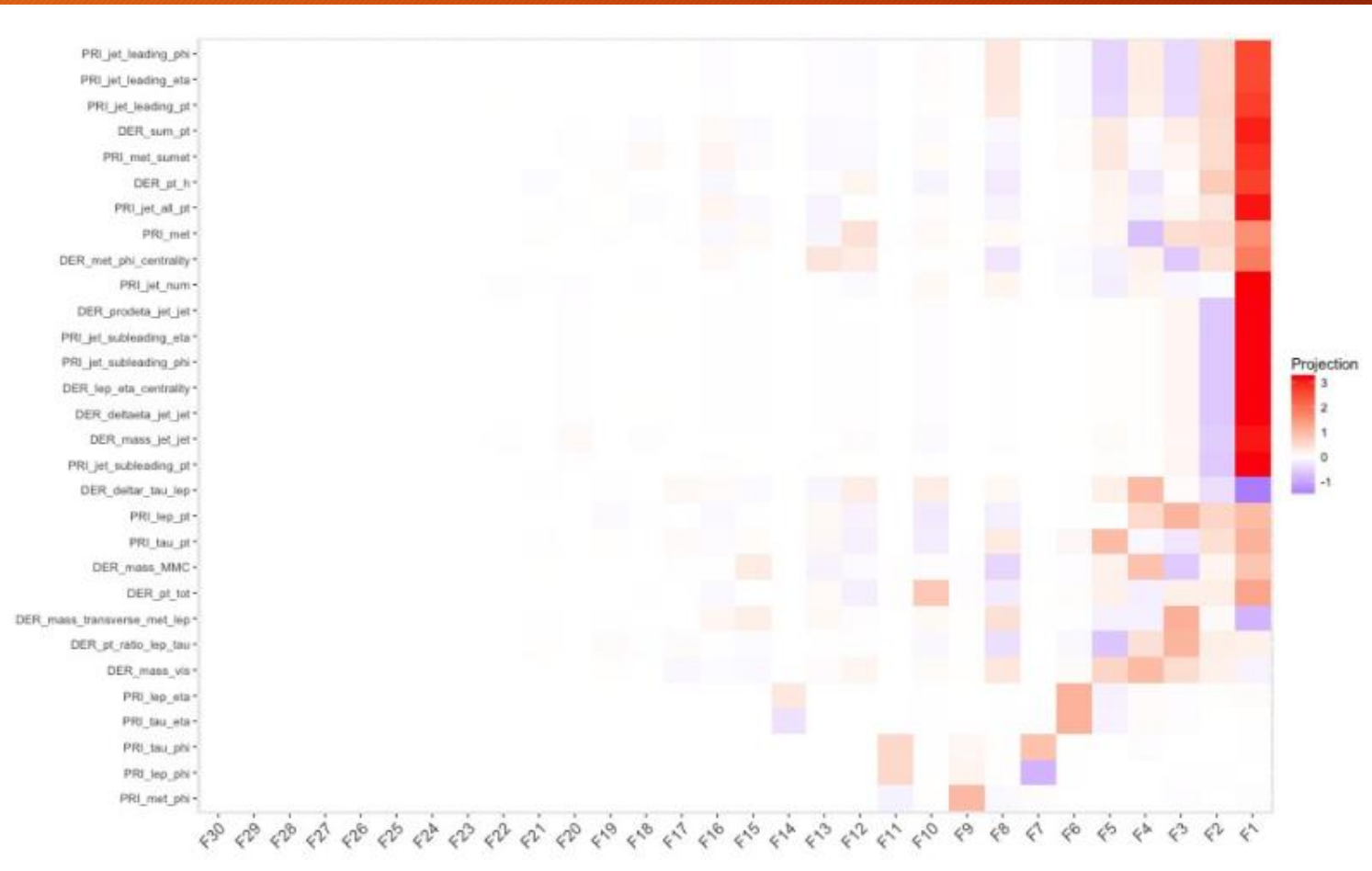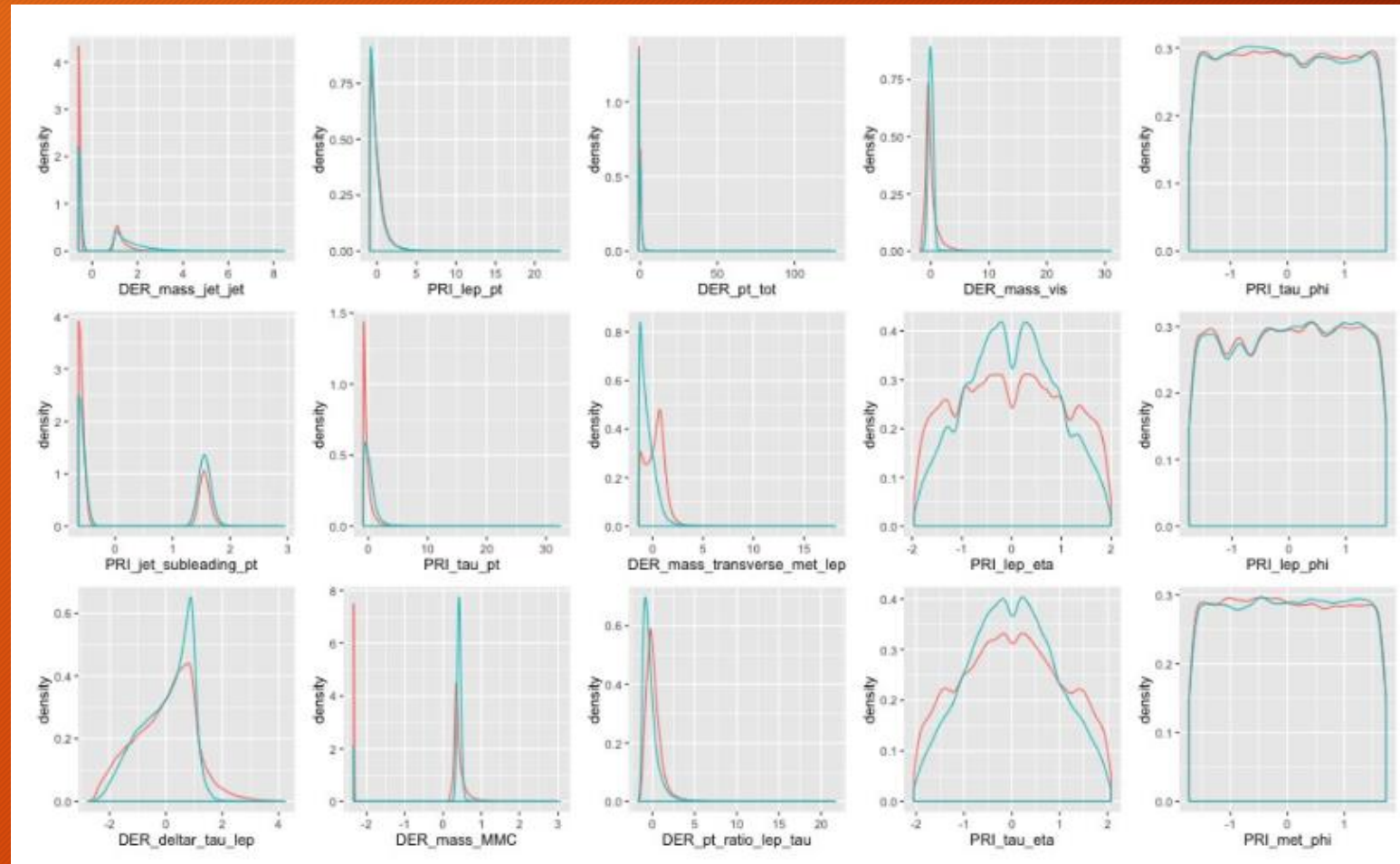# PCA Plot [1]

Parallel Analysis Scree Plots — PCA
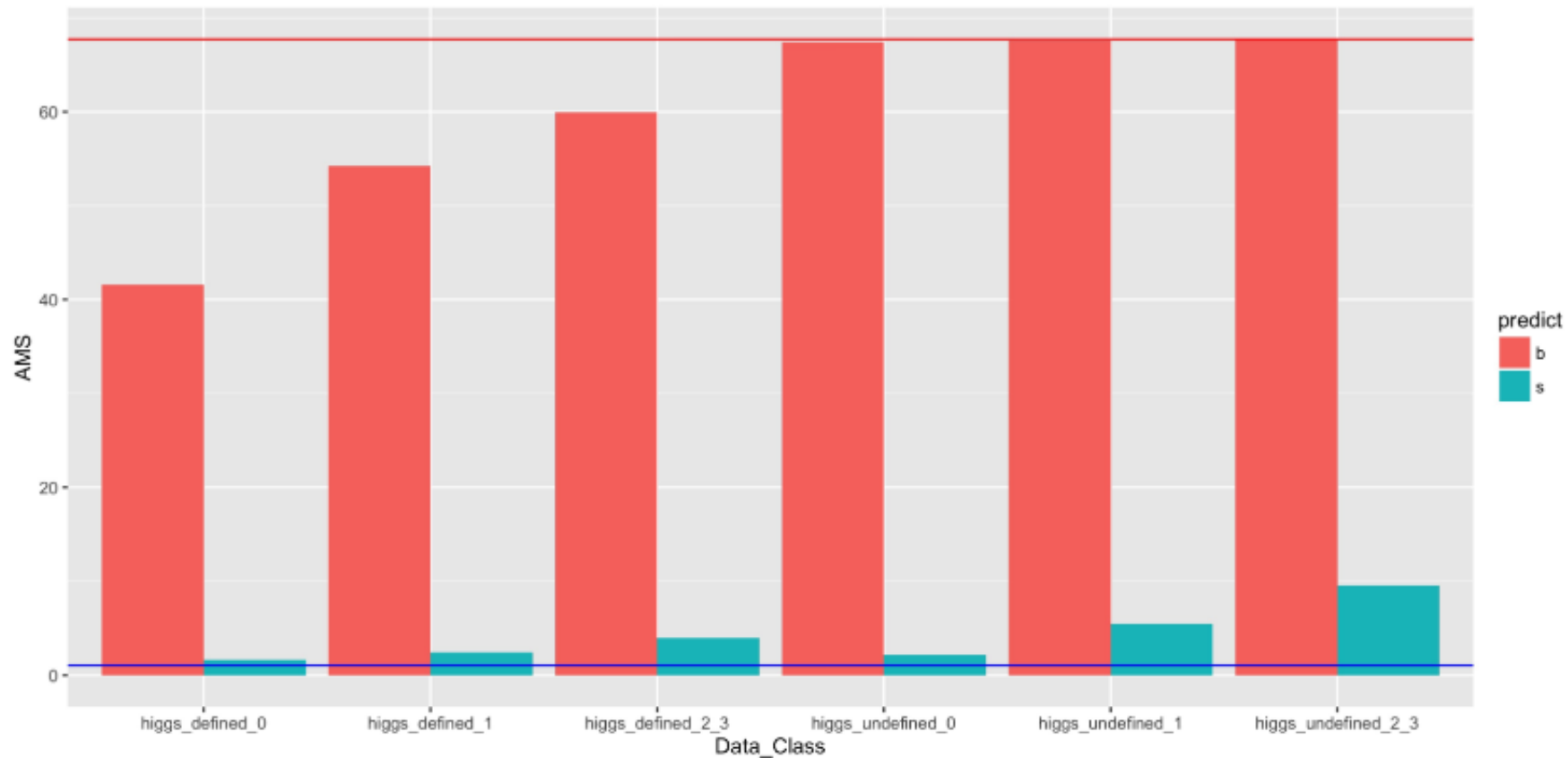
PCA Plot for Feature Reduction [2]

Least Influence Features
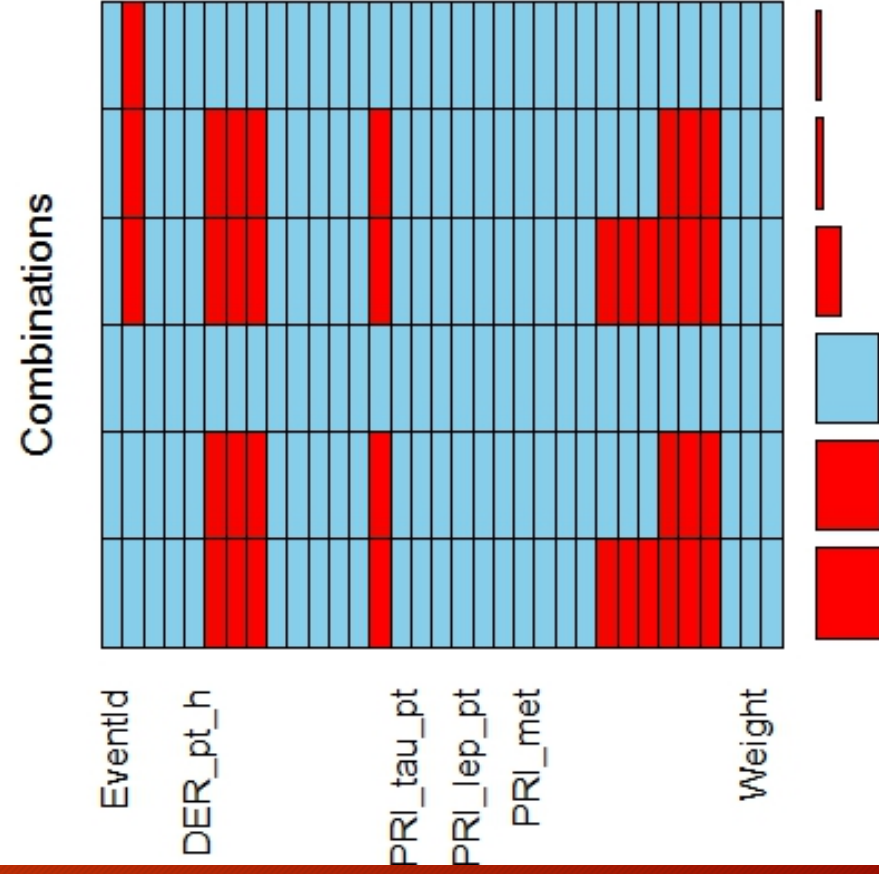
15% of Data No Influence on AMS

Imputation on "Missing" Data

V

- Zeros
- Mean
- Median
- Most Frequent
- "Do Nothing"

Missing Values Plots
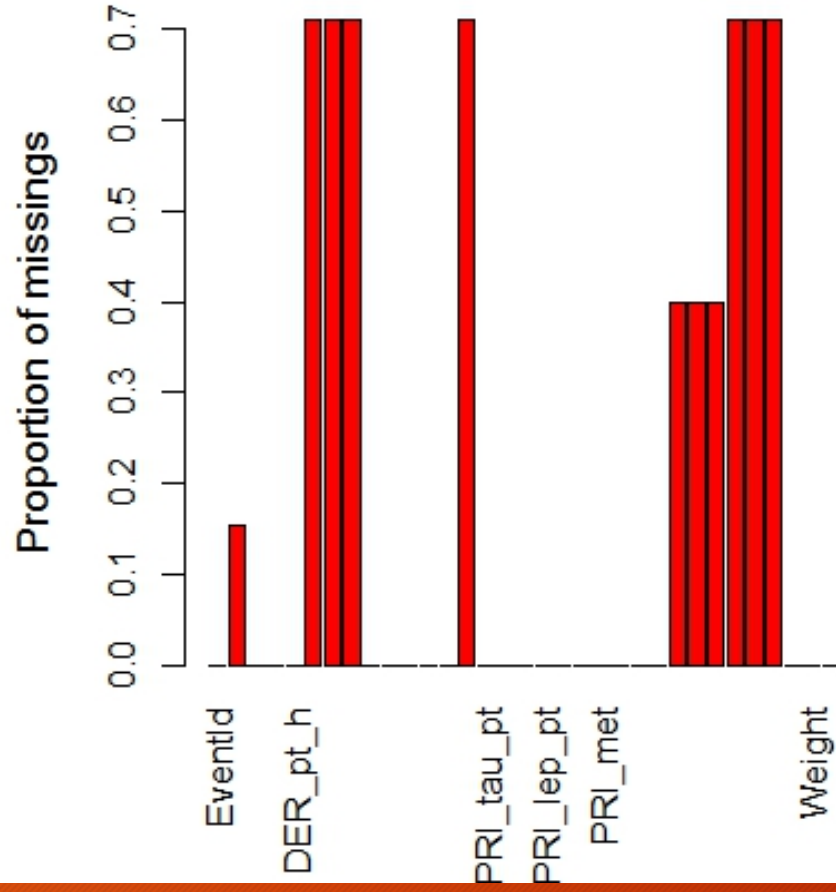
Training and Cross Validation

Parameters and Hyper-Parameters

Patterns and Trends

Tuning and Optimization

Naïve Bayes

All Input Data → Naïve Bayes Model → Parameter Tuning → Final Prediction → Local AMS Score = 1.00983
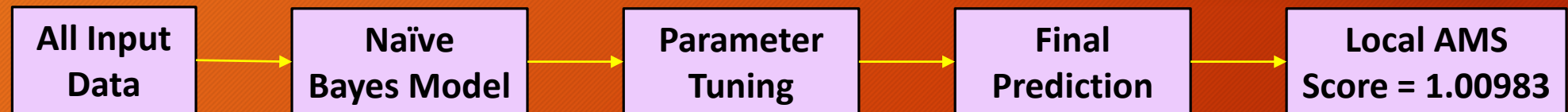
Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score
Single Model Algorithms							
Naïve Bayes	30	85%	none	False	False	3	0.99202
Naïve Bayes	30	86%	none	False	False	3	0.97608
Naïve Bayes	30	84%	none	False	False	3	1.00702
Naïve Bayes	30	83%	none	False	False	3	1.00969
Naïve Bayes	30	82%	none	False	False	3	1.00280
Naïve Bayes	30	83%	none	False	True	3	0.77996
Naïve Bayes	30	83%	none	True	False	3	1.00983
Naïve Bayes	30	85%	none	True	False	3	0.99227
Naïve Bayes	30	83%	zeros	True	False	3	0.88221
Naïve Bayes	30	83%	mean	True	False	3	0.92456
Naïve Bayes	30	83%	median	True	False	3	0.91496
Naïve Bayes	30	83%	most_frequent	True	False	3	0.88798

Logistic Regression

B

```
All Input Data  →  Logistic Regression Model  →  Parameter Tuning  →  Final Prediction  →  Local AMS Score = 1.81852
```
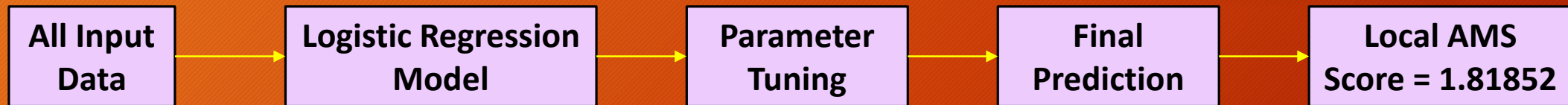
Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score
Logistic Regression	30	83%	none	True	False	3	1.78299
Logistic Regression	30	83%	none	False	False	3	1.76413
Logistic Regression	30	83%	none	True	True	3	1.55776
Logistic Regression	30	83%	zeros	True	False	3	1.81815
Logistic Regression	30	83%	mean	True	False	3	1.76026
Logistic Regression	30	83%	median	True	False	3	1.78949
Logistic Regression	30	83%	most_frequent	True	False	3	1.81852

Neural Networking Model

```
Dropped Set        Neural           Parameter        Final            Local AMS          Private LB AMS
Input Data    →    Network Model →  Tuning       →   Prediction   →   Score = 2.698  →   Score = 2.745
```
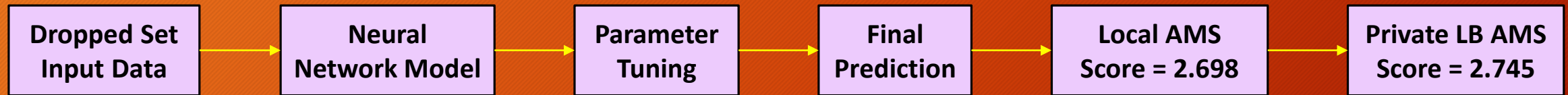
Neural Network	No. of Features	Hidden Layer	Neurons	Standardized	PCA	Xval Folds	AMS Score	Kaggle Position	Private LB AMS	Top %	
Neural Network	30	1	10	False	False	10	2.066		AMS = 2.107		Whole Data
Neural Network	17	1	10	False	False	10	1.587		AMS = 1.633		Removing The DER variables
Neural Network	17	1	20	False	False	10	1.622		AMS = 1.627		Removing The DER variables
Neural Network	25	1	10	False	False	10	2.306		AMS = 2.317		Removing The Phi's
Neural Network	25	1	20	False	False	10	2.188		AMS = 2.322		Removing The Phi's
Neural Network	25	1	30	False	False	10	2.218		AMS = 2.327		Removing The Phi's
Neural Network	29	1	10	False	False	10	2.552		AMS = 2.566		Grouping Based On Jet Value
Neural Network	29	1	20	False	False	10	2.309		AMS = 2.674		Grouping Based On Jet Value
Neural Network	29	1	30	False	False	10	2.698	Kaggle # 1253	AMS = 2.745	Top 70%	Grouping Based On Jet Value

Boosting, Random Forest, SVM
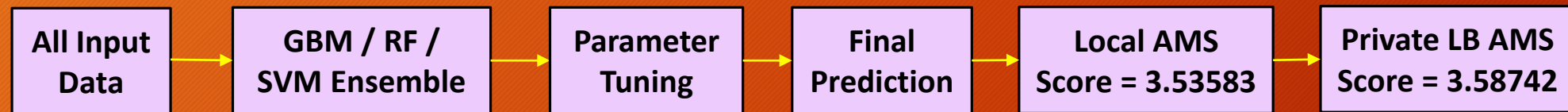
All Input Data → GBM / RF / SVM Ensemble → Parameter Tuning → Final Prediction → Local AMS Score = 3.53583 → Private LB AMS Score = 3.58742

Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score	Kaggle Position	Private LB AMS	Top %
Ensemble Models										
Gradient Boosting	30	83%	zeros	True	False	3	3.50757			
Gradient Boosting	30	83%	zeros	False	False	3	3.50790			
Gradient Boosting	30	85%	none	False	False	3	3.53583	Kaggle # 615	AMS = 3.58742	Top 35%
Gradient Boosting	30	83%	none	True	True	3	3.25130			
Gradient Boosting	30	83%	none	True	False	3	3.52030			
Gradient Boosting	30	83%	zeros	True	False	3	3.50760			
Gradient Boosting	30	85%	zeros	False	False	3	3.52930			
Gradient Boosting	30	83%	none	False	False	3	3.52010			
Gradient Boosting	30	85%	mean	False	False	3	3.53420			
Gradient Boosting	30	83%	mean	False	False	3	3.49640			
Gradient Boosting	30	83%	zeros	False	False	3	3.50790			
Gradient Boosting	30	83%	most_frequent	False	False	3	3.52242			
Random Forest	30	85%	none	False	False	10	2.82230	max depth = 7		
SVM	30	85%	none	False	False	3	2.54300			

XGBoost

All Input Data → XGBoost Ensemble → Parameter Tuning → Final Prediction → Local AMS Score = 3.60760 → Private LB AMS Score = 3.64051

Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score	Kaggle Position	Private LB AMS	Top %
XGBoost	30	85%	none	False	False	3	3.60760	Kaggle # 474	AMS = 3.64051	Top 27%
XGBoost	30	83%	none	False	False	3	3.58665			
XGBoost	30	85%	mean	False	False	3	3.55439			
XGBoost	30	85%	median	False	False	3	3.58718			
XGBoost	30	85%	zeros	False	False	3	3.59191			
XGBoost	30	85%	most_frequent	False	False	3	3.55665			
XGBoost	30	85%	none	True	False	3	3.58810			
XGBoost	30	85%	none	True	True	3	3.35131			
XGBoost	30	85%	none	False	True	3	3.42500			
XGBoost	30	85%	none	False	False	5	3.58605			
XGBoost	30	85%	none	False	False	10	3.64842	Overfitted	Same AMS	
XGBoost	30	85%	none	False	False	15	3.64190			
XGBoost	30	85%	none	False	False	20	3.64129			
XGBoost	30	85%	none	False	False	30	3.65412	Overfitted	Same AMS	

Drop the 5-phi Features
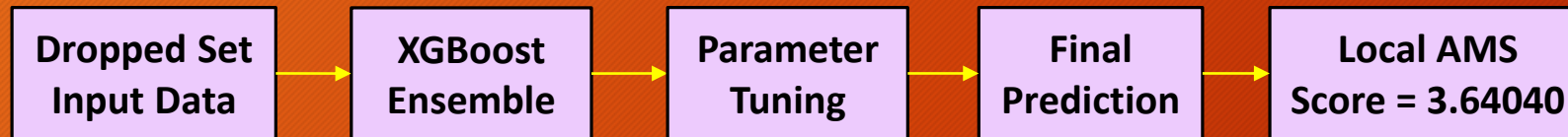
Dropped Set Input Data	→	XGBoost Ensemble	→	Parameter Tuning	→	Final Prediction	→	Local AMS Score = 3.64040

Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score	Kaggle Position	Private LB AMS	Top %
Feature Engineering - Remove 5 PHI based variables										
XGBoost	25	85%	none	False	False	10	3.64040			
SVM	13	85%	none	False	False	3	0.67110	- 12 DER vars		

Drop Features with >70% Missing Data
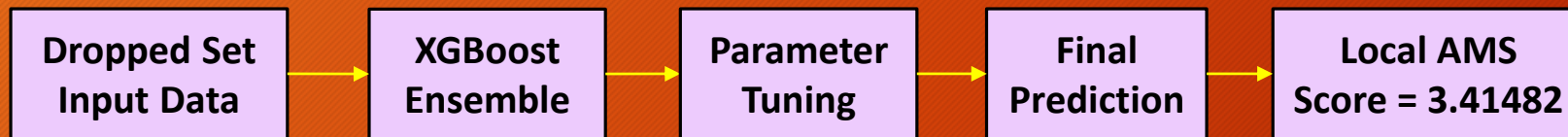
V

Dropped Set Input Data → XGBoost Ensemble → Parameter Tuning → Final Prediction → Local AMS Score = 3.41482

Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score	Kaggle Position	Private LB AMS	Top %
Feature Drops - Remove 7 Features Based on NA > 70% --- "DER_mass_transverse_met_lep", "DER_mass_vis","DER_pt_h","DER_deltaeta_jet_jet","DER_pt_tot","DER_sum_pt										
XGBoost	23	85%	none	False	False	3	3.38995			
XGBoost	23	83%	none	False	False	3	3.33877			
XGBoost	23	85%	none	True	False	3	3.41482			
XGBoost	23	85%	none	True	True	3	3.28738			
XGBoost	23	85%	none	False	True	3	3.28994			
XGBoost	23	85%	zeros	False	False	3	3.39736			
XGBoost	23	85%	mean	False	False	3	3.38502			
XGBoost	23	85%	median	False	False	3	3.36454			
XGBoost	23	85%	most_frequent	False	False	3	3.40686			

Polynomial Features

```
# ------------------------------------------
# Fields with high multicollinearity
# ------------------------------------------

- C01    # 25, 5     PRI_jet_leading_pt      DER_pt_h
- C02    # 23, 5     PRI_met_sumet           DER_pt_h
- C03    # 11, 5     DER_sum_pt              DER_pt_h
- C04    # 31, 5     PRI_jet_all_pt          DER_pt_h
- C05    # 23, 28    PRI_met_sumet           PRI_jet_subleading_pt
- C06    # 11, 28    DER_sum_pt              PRI_jet_subleading_pt
- C07    # 31, 28    PRI_jet_all_pt          PRI_jet_subleading_pt
- C08    # 23, 25    PRI_met_sumet           PRI_jet_leading_pt
- C09    # 11, 25    DER_sum_pt              PRI_jet_leading_pt
- C10    # 31, 25    PRI_jet_all_pt          PRI_jet_leading_pt
- C11    # 11, 23    DER_sum_pt              PRI_met_sumet
- C12    # 31, 23    PRI_jet_all_pt          PRI_met_sumet
- C13    # 31, 11    PRI_jet_all_pt          DER_sum_pt
- C14    # 4, 2      DER_mass_vis            DER_mass_MMC
- C15    # 18, 12    PRI_lep_pt              DER_pt_ratio_lep_tau
- C16    # 24, 10    PRI_jet_num             DER_pt_tot
```

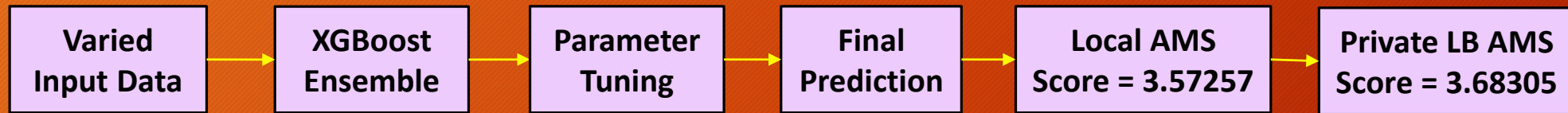**Engineered Input Data** → **XGBoost Ensemble** → **Parameter Tuning** → **Final Prediction** → **Local AMS Score = 3.57455** → **Private LB AMS Score = 3.64580**

Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score	Kaggle Position	Private LB AMS	Top %
Feature Engineering - Created Polynomial Params on multicollinear pairs features - tested with source features included										
XGBoost	78	85%	none	False	False	3	3.57455	Kaggle #453	AMS = 3.64580	Top 25%
XGBoost	78	85%	none	True	False	3	3.59784	Kaggle #462	AMS = 3.64441	Top 26%
XGBoost	78	85%	none	True	False	30	3.63050	Overfitted	Same AMS	
XGBoost	23	85%	none	False	False	3	3.38995			

XGBoost Hyper-Parm Tuning

```
┌──────────────┐   ┌──────────────┐   ┌──────────────┐   ┌──────────────┐   ┌──────────────────┐   ┌──────────────────┐
│   Varied     │ → │   XGBoost    │ → │  Parameter   │ → │    Final     │ → │   Local AMS      │ → │  Private LB AMS  │
│  Input Data  │   │  Ensemble    │   │   Tuning     │   │  Prediction  │   │ Score = 3.57257  │   │ Score = 3.68305  │
└──────────────┘   └──────────────┘   └──────────────┘   └──────────────┘   └──────────────────┘   └──────────────────┘
```

Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score	Kaggle Position	Private LB AMS	Top %	
XGBoost Hyper-Parameters Optimization and Finetuning											
XGBoost	30	85%	none	False	False	3	3.60760	objective = **binary:logitraw** / binary:logistic / reg:logistic			
XGBoost	30	85%	none	False	False	3	3.49675	objective = reg:linear			
XGBoost	30	85%	none	False	False	3	1.08533	objective = count:poisson			
XGBoost	30	85%	none	False	False	3	3.45721	objective = rank:pairwise			
XGBoost	30	85%	none	False	False	3	3.06477	eta = 0.01			
XGBoost	30	85%	none	False	False	3	3.45622	eta = 0.04			
XGBoost	30	85%	none	False	False	3	3.58646	Kaggle # 275	AMS = 3.65857	Top 15%	eta = 0.12 / max depth = 7
XGBoost	30	85%	none	False	False	3	3.54356	eta = 0.16	max depth = 7		
XGBoost	30	85%	none	False	False	3	3.48748	eta = 0.12	max depth = 9		
XGBoost	30	85%	none	False	False	3	3.54902	eta = 0.08	max depth = 9		
XGBoost	30	85%	none	False	False	3	3.54930	eta = 0.12	max depth = 7	subsample = 0.9	
XGBoost	30	85%	none	False	False	3	3.58646	eta = 0.12	max depth = 7	eval metric overfitted	
XGBoost	30	85%	none	False	False	3	3.58646	eta = 0.12	max depth = 7	eval metric overfitted	
XGBoost	78	85%	none	False	False	3	3.56471	eta = 0.12	max depth = 7	78 polynomial features	
XGBoost	78	85%	none	False	False	5	3.60254	eta = 0.12	max depth = 7	78 polynomial features	
XGBoost	25	85%	none	False	False	3	3.57257	Kaggle # 144	AMS = 3.68305	Top 8%	eta = 0.12 / max depth = 7 / Drop 5 phi features
XGBoost	25	85%	none	False	False	3	3.59074	eta = 0.08	max depth = 7	Drop 5 phi features	

Model Ensemble Stacking Tuning

BN
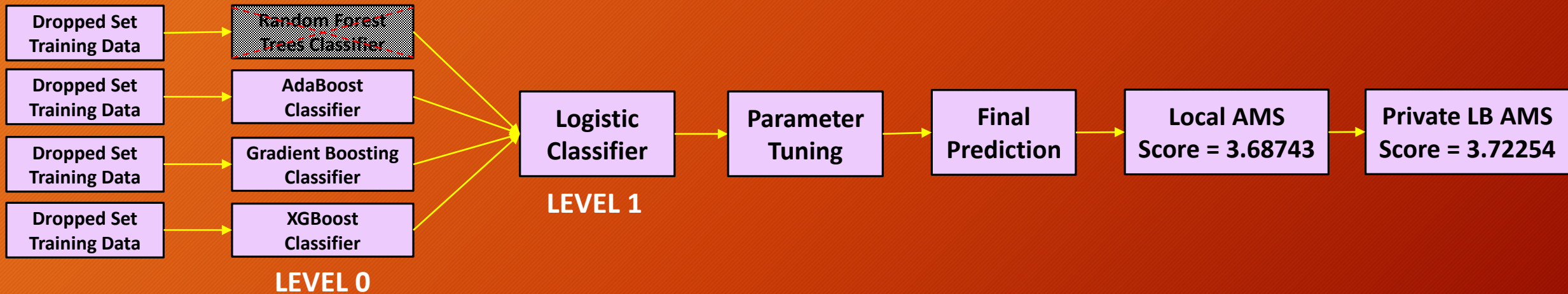
Dropped Set Training Data	→	Random Forest Trees Classifier
Dropped Set Training Data	→	AdaBoost Classifier
Dropped Set Training Data	→	Gradient Boosting Classifier
Dropped Set Training Data	→	XGBoost Classifier

LEVEL 0

Logistic Classifier → Parameter Tuning → Final Prediction → Local AMS Score = 3.68743 → Private LB AMS Score = 3.72254

LEVEL 1

Algorithm	No. of Features	Threshold	Imputer	Standardized	PCA	Xval Folds	AMS Score	Kaggle Position	Private LB AMS	Top %	
Model Ensemble Stacking											
Model Stacking	30	85%	none	False	False	3	0.58160	3% of training data			boost + xgboost -> logistic
Model Stacking	30	85%	none	False	False	3	2.01150	30% of training data			boost + xgboost -> logistic
Model Stacking	30	85%	none	False	False	3	3.65192	Kaggle # 127	AMS = 3.68787	Top 7%	boost + xgboost -> logistic
Model Stacking	30	85%	none	False	False	5	3.68160	Kaggle # 118	AMS = 3.69373	Top 7%	boost + xgboost -> logistic
Model Stacking	25	85%	none	False	False	5	3.67733	Kaggle # 107	AMS = 3.69600	Top 6%	boost + xgboost -> logistic / drop 5 PHI features / eta=0.08
Model Stacking	25	85%	none	False	False	5	3.68163	Kaggle # 99	AMS = 3.69919	Top 6%	boost + xgboost -> logistic / drop 5 PHI features / eta=0.08 / depth = 10
Model Stacking	25	85%	none	False	False	5	3.65121	Kaggle # 90	AMS = 3.70123	Top 5%	forest + adaboost + boost + xgboost -> logistic / drop 5 PHI features / eta=0.08 / depth = 10
Model Stacking	25	85%	none	False	False	5	3.68000	Kaggle # 45	AMS = 3.71929	Top 3%	boost + xgboost -> logistic / drop 5 PHI features / eta=0.08 / boost depth = 9
Model Stacking	25	85%	none	False	False	5	3.68989	Kaggle # 34	AMS = 3.72250	Top 2%	boost + xgboost -> logistic / drop 5 PHI features / eta=0.08 / boost depth = 9 / xgb depth = 7
Model Stacking	25	85%	none	False	False	5	3.68743	Kaggle # 34	AMS = 3.72254	Top 2%	adaboost + boost + xgboost -> logistic / drop 5 PHI features / eta=0.08 / boost depth = 9 / xgb depth = 7

Results and Findings

Interpretation

Discovery and Insights

Feature Analysis

- **Some variables are meaningless or cannot be computed; value is –999.0, which is outside the normal range of all variables**
  - Another reason to not impute -999.0
  - Imputation is a bad idea and wrong because –999.0 is not missing at all
  - -999.0 indicates nothing is meant to be there
- **Split training and test data into 6 categories according to level of undefined features**
- **PRI_jet_num is a categorical variable telling which of 4 {jets(1,2,3) and no (0) jet} event happened**
- **7 variables are missing more than 70 % of data in training set**
- **Last variables in PCA feature plots have least influence in their current form**
- **Weights and Labels are dependent on each other from chisquare test of dependence**
- **Signal events have low weights and Background events have high weights**
- **Signal events weight range is exclusively separated from background event weight range**
- **Signal seems to have only three weights Looks like they are looking for three channels of Higgs Boson**
- **Not worth spending time coming up with the formula It's already done for us**

Findings

- The Variable Importance plot as well as PCA showed that the 5 angle ('phi') variables seemed to be the least important of the 30 predictors. Although removal of the above variables did not result in a significant improvement of the AMS score using a single model, it did help increase the score quite a bit for the Ensemble model

- Ensemble models performed better than single models and stacking the Ensemble models provided the best result in achieving the final AMS score

- The final Stacking model consisted of AdaBoost, Gradient Boost, XGBoost as Level 0 classifiers followed by Logistic Regression as Level 1 classifier

- Stacking several models in the base layer does not always enhance the final result (AMS score). In our testing, three base classifiers (AdaBoost, Gradient Boost and XGBoost) provided almost equivalent results compared to two base classifiers

- Cross Validation plays a key role in testing the model. Based on our testing, the optimal number of folds for cross validation was determined as 5 since 3 folds seemed to underfit the model while 10 folds seemed to overfit the model

ROC / AUC Results Curve
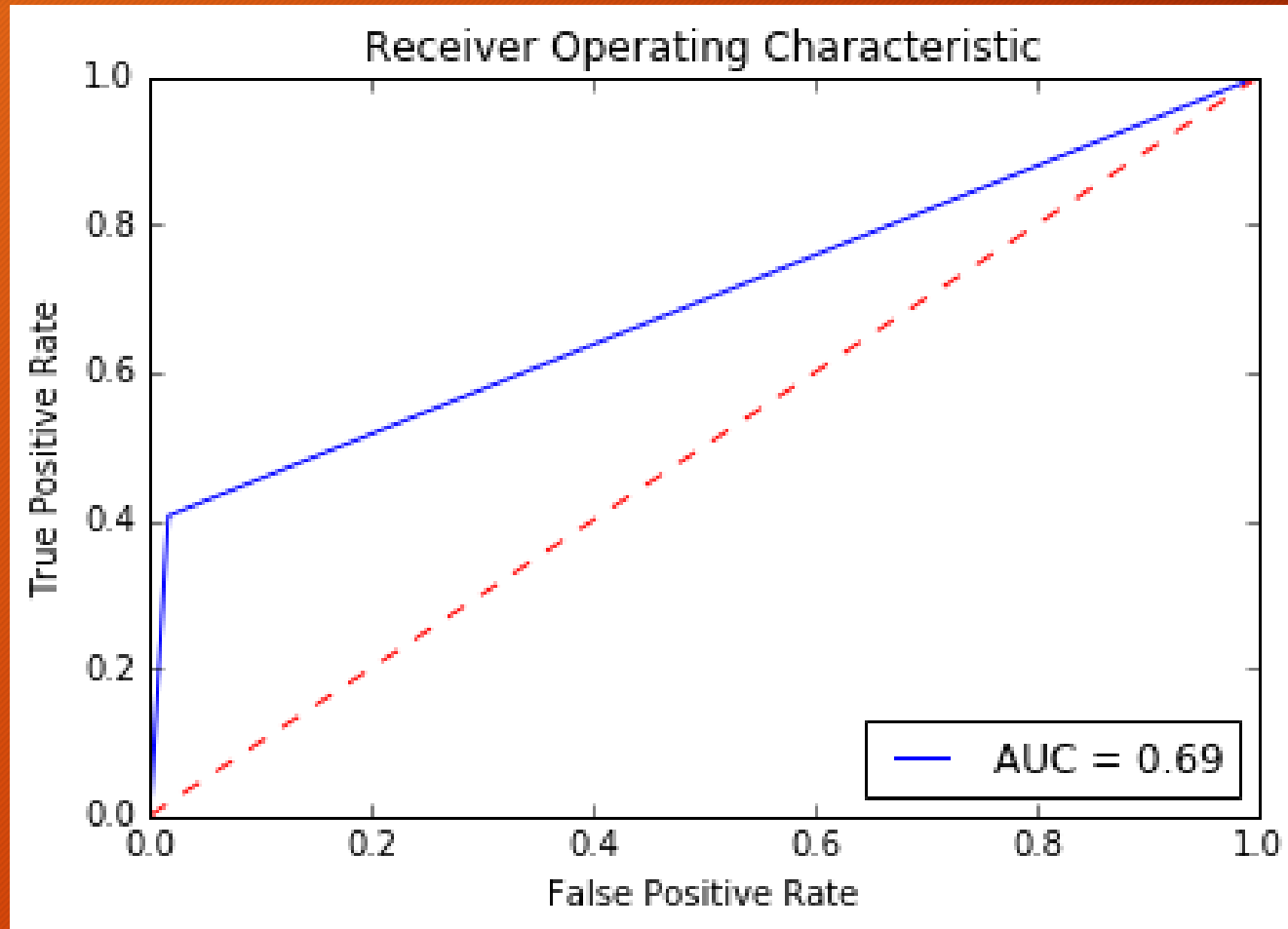
Sensitivity / Specificity Curve
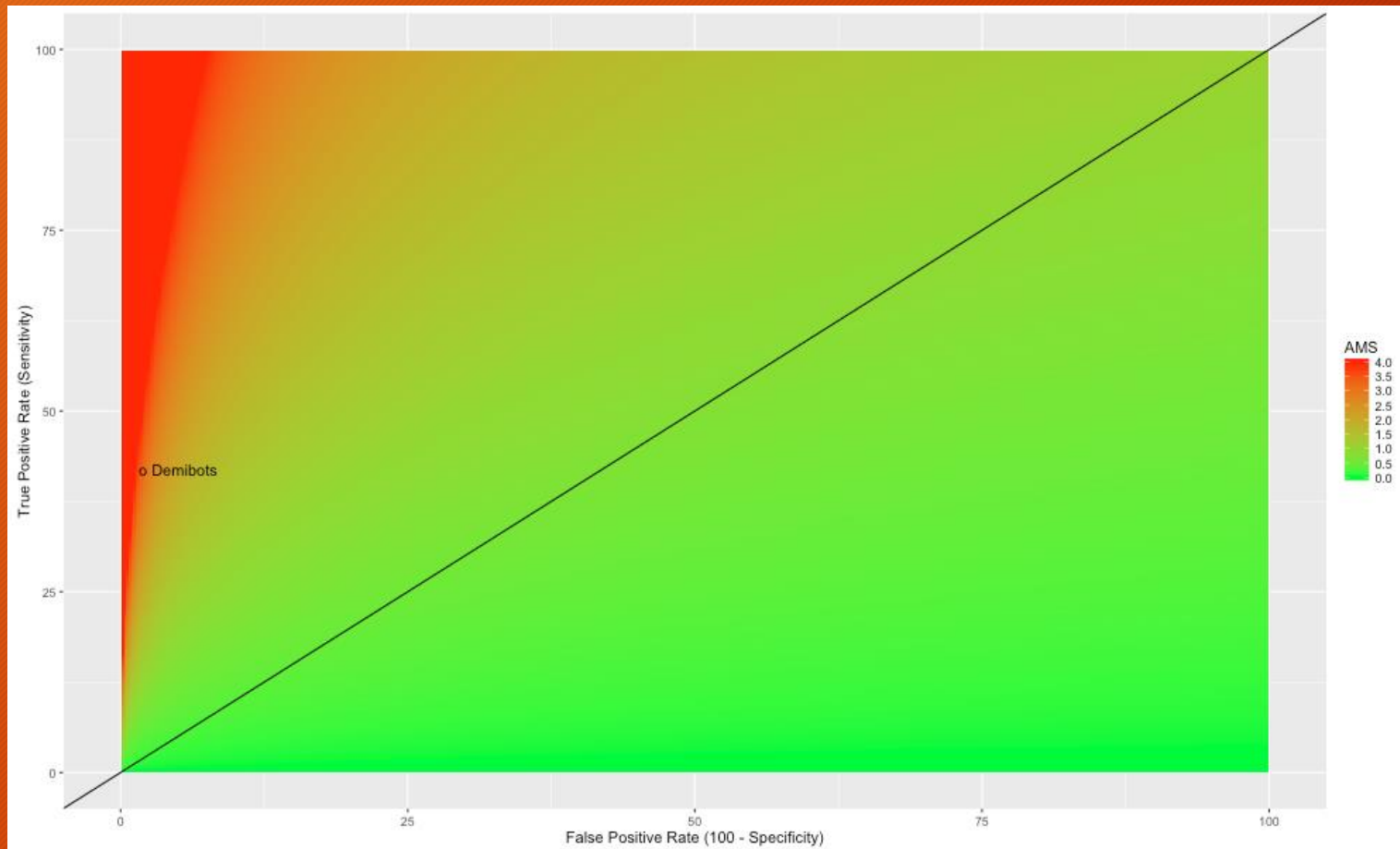
Lessons Learned

- An increase in the local AMS score based on the training data does not always translate to an increase in the private Kaggle Leaderboard score due to potential overfitting of the model

- Combination of (no. of models) x (no. of parameters) x (no. of features) x (no. of VABS) is complex and overwhelming. Disciplined process, pipelining, and automation are a must

- Methodical Tuning of the parameters is critical during testing (focusing on one parameter at a time). Random tuning of the parameters would make it difficult to keep track of the changes and may result in over-tuning that would end up decreasing the Kaggle score

- SVM is highly computation intensive and the features and parameters need to be carefully selected before the execution of the algorithm. The algorithm runs for several hours even with reduced features and data

- The maximum number of trees used need not be the same for all the models in a stacked classifier. Increasing the number of trees helped in improving the AMS score only up to a certain threshold. Any further increase in the trees resulted in decreasing the overall AMS score due to potential overfitting

Insights

- **Feature engineering (creating new features) did not really work that well – it was not the worth effort**

- **Ensembles and stacking clearly demonstrated significant improvements**

- **Hyper-parameters fine-tuning and optimization are very difficult and time consuming**
  - Each little change in hyper-parameters could decrease or increase accuracy significantly
  - A fine line exists balancing across effort vs computing time vs complexity vs accuracy

- **No clear strategy to deal with missing data except to just leave it alone**

- **Managing theoretical feature significance to improve prediction accuracy was very challenging in actual practice**

- **Prediction accuracy would have been even more challenging during a real competition due to the absence of the private leaderboard**

What We Would Have Liked to Do

BN

- **Find better techniques to auto-generate new features to increase prediction accuracy**
  - **Explore SVM in depth and try various options for tuning parameters and analyze effects on final score**
  - **Blending machine learning and physics (or business) models have a big potential for increased accuracy**
- **Build out our expertise in ensembles and aggregation strategies (VABS)**
  - **Inject more random seed sampling for base models**
- **Invest more time and effort to optimize our neural network frameworks**
  - **Enhance the current Ensemble model by including the Neural Network algorithm (Deep Learning) and test the predictive accuracy**
- **Use a Bayesian Optimizer tool to automate pipeline iterations to find best combination of hyper-parameters (given its complexity)**
  - **Use GridSearch to determine the best combination of parameters for the Tree based Boosting models**

Conclusion

Summary and Synopsis

Final Thoughts

Summary

- We have met our objectives in this Machine Learning Challenge, being able to apply various models, algorithms, and strategies to achieve relatively good predictions

- We have obtained the final private leaderboard score of **3.72254**, the 34th position

- We have a renewed sense of appreciation of what machine learning can do, and also the power behind the complexity

- We have learned and developed a lot of new skills from the project, but also realize how much more we still don't know

Winning or getting high scores in Kaggle does not necessarily equate to being a good Data Scientist.

Being great at machine learning is more than just depending on an individual's smarts and skills.

Knowing how to form a great team that works well together plays a major role in how one succeeds in the field of Data Science.

Final Thoughts

Final Private LB AMS Score

30	↑59	all your HADRON are belong to us		3.72654	65	Mon, 15 Sep 2014 15:14:34
31	↑33	mymo		3.72594	73	Sun, 14 Sep 2014 20:32:19 (-2.4d)
32	↓13	BlackMagic		3.72491	33	Sat, 23 Aug 2014 20:53:47 (-74.4d)
33	↑45	Alayne		3.72403	101	Mon, 08 Sep 2014 13:04:55 (-38.2d)
-		**Team Demibots (Bernard-Nanda-Venkat-Deepak)**	**Top 2%**	**3.72254**	-	**Sat, 27 Aug 2016 03:32:40** **Post-Deadline**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

34	↑49	Wei Wu	3.72235	45	Mon, 15 Sep 2014 14:35:18 (-0h)
35	↑201	Phil Culliton	3.72224	91	Mon, 15 Sep 2014 21:24:55
36	↓8	cherrybarry	3.72204	144	Mon, 15 Sep 2014 23:53:31 (-0.1h)
37	↑21	Carlos Fernandez	3.72180	108	Mon, 15 Sep 2014 16:20:02 (-0.2h)

Appendix

More Details :

XGBoost Details

More EDA Plots

Benefits of XGBoost

- **Regularization & Parallel Processing**
  - XGBoost implements regularization that helps in reducing overfitting. It also implements parallel processing that significantly expedites the model fitting process compared to GBM.
  - XGBoost utilizes OpenMP which can parallelize the code on a multithreaded CPU automatically. XGBoost has defined a data structure DMatrix to store the data matrix. This data structure will perform some preprocessing work on the data so that the latter iterations run faster.

- **High Flexibility**
  - XGBoost allow users to define custom optimization objectives and evaluation criteria. This adds a whole new dimension to the model and significantly enhances the flexibility

- **Handling Missing Values**
  - XGBoost has an in-built routine to handle missing values.
  - User is required to supply a different value than other observations and pass that as a parameter. XGBoost tries different things as it encounters a missing value on each node and learns which path to take for missing values in future.

- **Tree Pruning**
  - A GBM would stop splitting a node when it encounters a negative loss in the split. Thus it is more of a greedy algorithm.
  - XGBoost on the other hand makes the splits up to the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.
  - Another advantage is that sometimes a split of negative loss say -2 may be followed by a split of positive loss +10. GBM would stop as it encounters -2. But XGBoost will go deeper and it will see a combined effect of +8 of the split and keep both.

XGBoost Hyper-Parameters

- **General Parameters**
  - booster [default=gbtree]: The type of booster to be used. This value can be gbtree, gblinear or dart. gbtree and dart use tree based model while gblinear uses linear function.
  - silent [default=0]: 0 means printing running messages, 1 means silent mode.
  - nthread [default to maximum number of threads available if not set]: number of parallel threads used to run xgboost
  - num_pbuffer [set automatically by xgboost, no need to be set by user]: size of prediction buffer, normally set to number of training instances. The buffers are used to save the prediction results of last boosting step.
  - num_feature [set automatically by xgboost, no need to be set by user]: feature dimension used in boosting, set to maximum dimension of the feature

- **Tree Booster Parameters**
  - eta [default=0.3] : step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features. and eta actually shrinks the feature weights to make the boosting process more conservative.
  - max_depth [default=6] : maximum depth of a tree, increase this value will make model more complex / likely to be overfitting.
  - scale_pos_weight [default=0]: Control the balance of positive and negative weights, useful for unbalanced classes. A typical value to consider: sum(negative cases) / sum(positive cases)
  - subsample [default=1] : subsample ratio of the training instance. Setting it to 0.5 means that XGBoost randomly collected half of the data instances to grow trees and this will prevent overfitting.

- **Learning Task Parameters**
  - objective [default=reg:linear] Used "binary:logitraw" for this scenario. Logistic regression for binary classification that outputs score before logistic transformation
  - eval_metric : evaluation metrics for validation data, a default metric will be assigned according to objective(RMSE for regression, and error for classification, mean average precision for ranking)
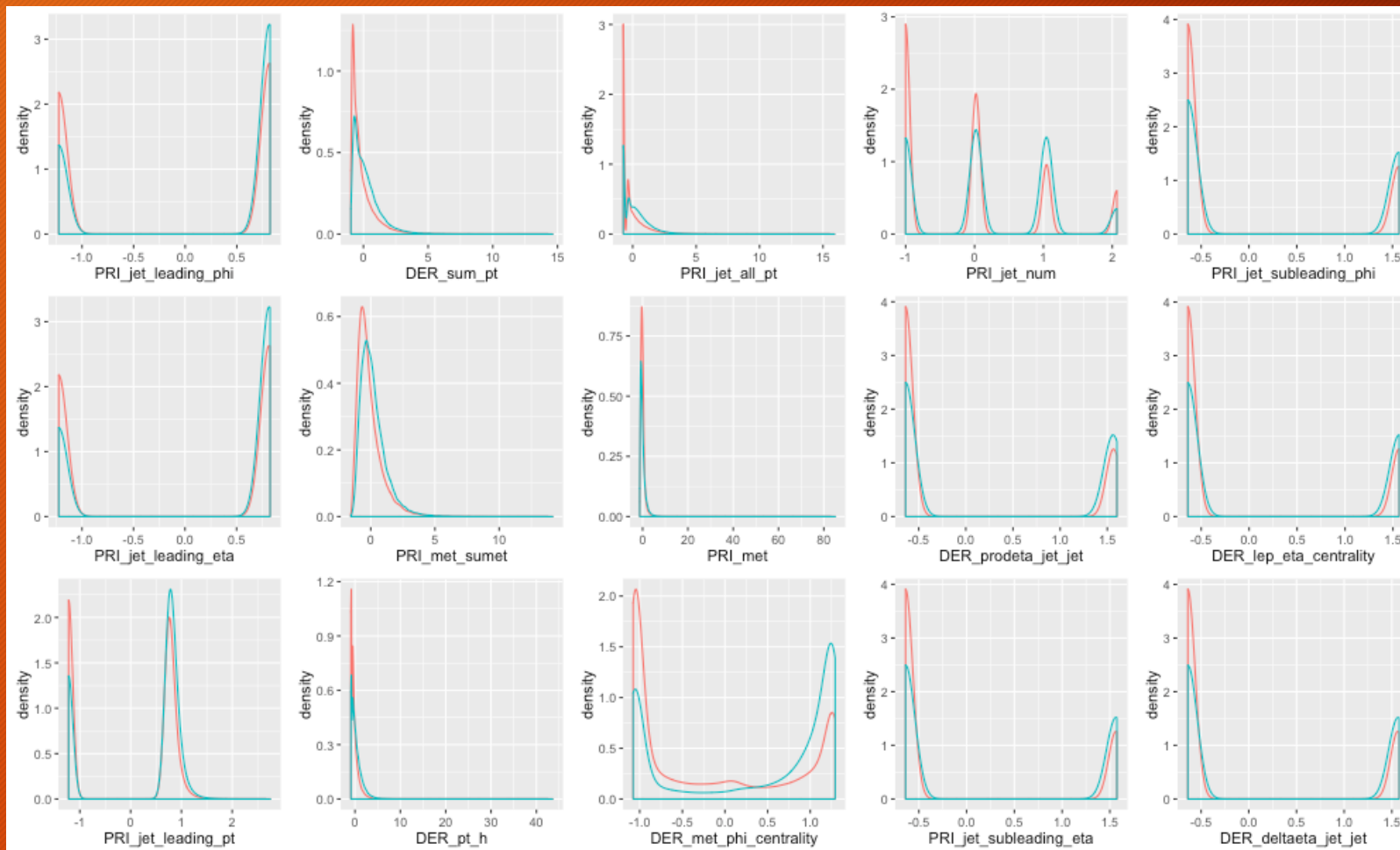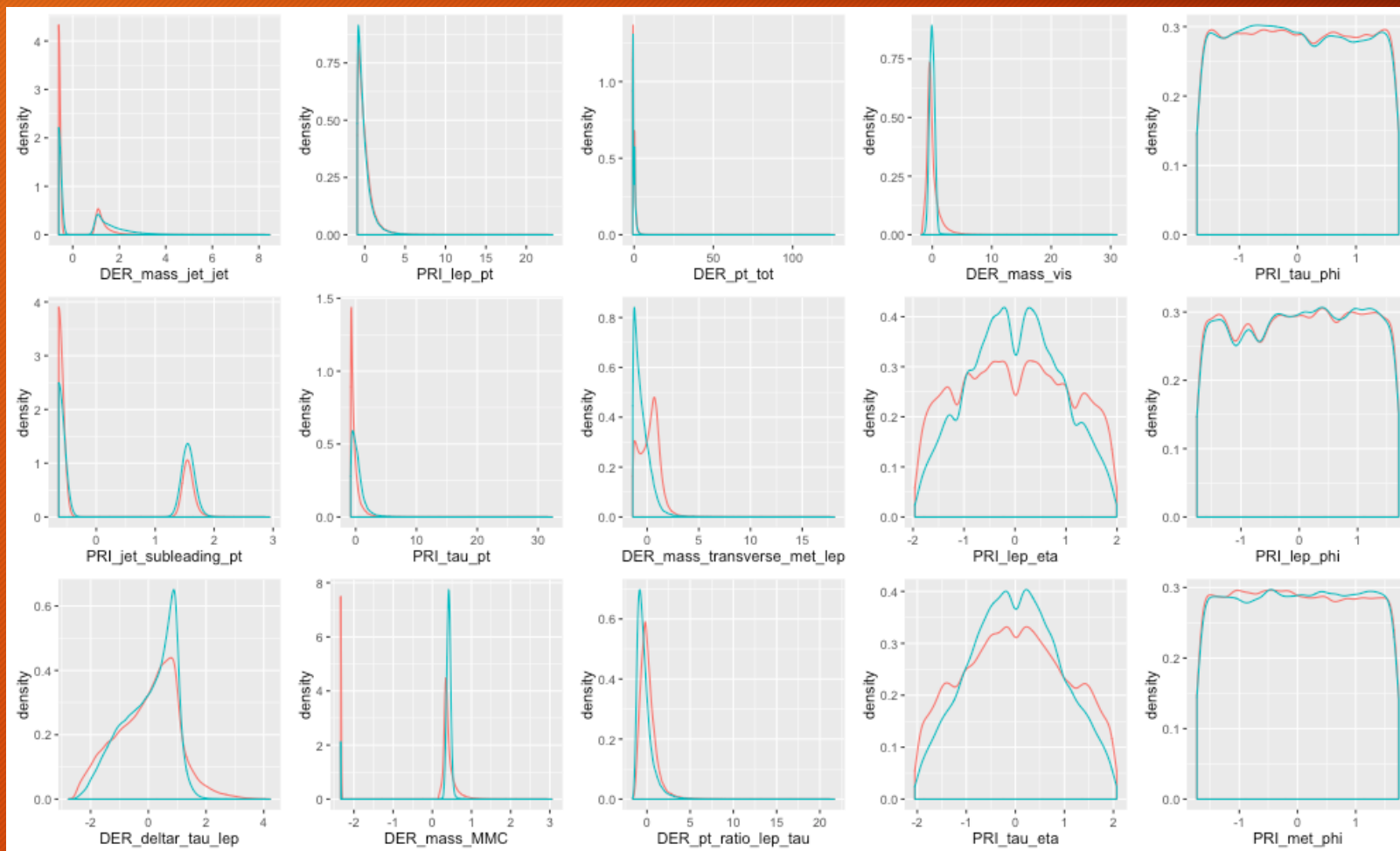
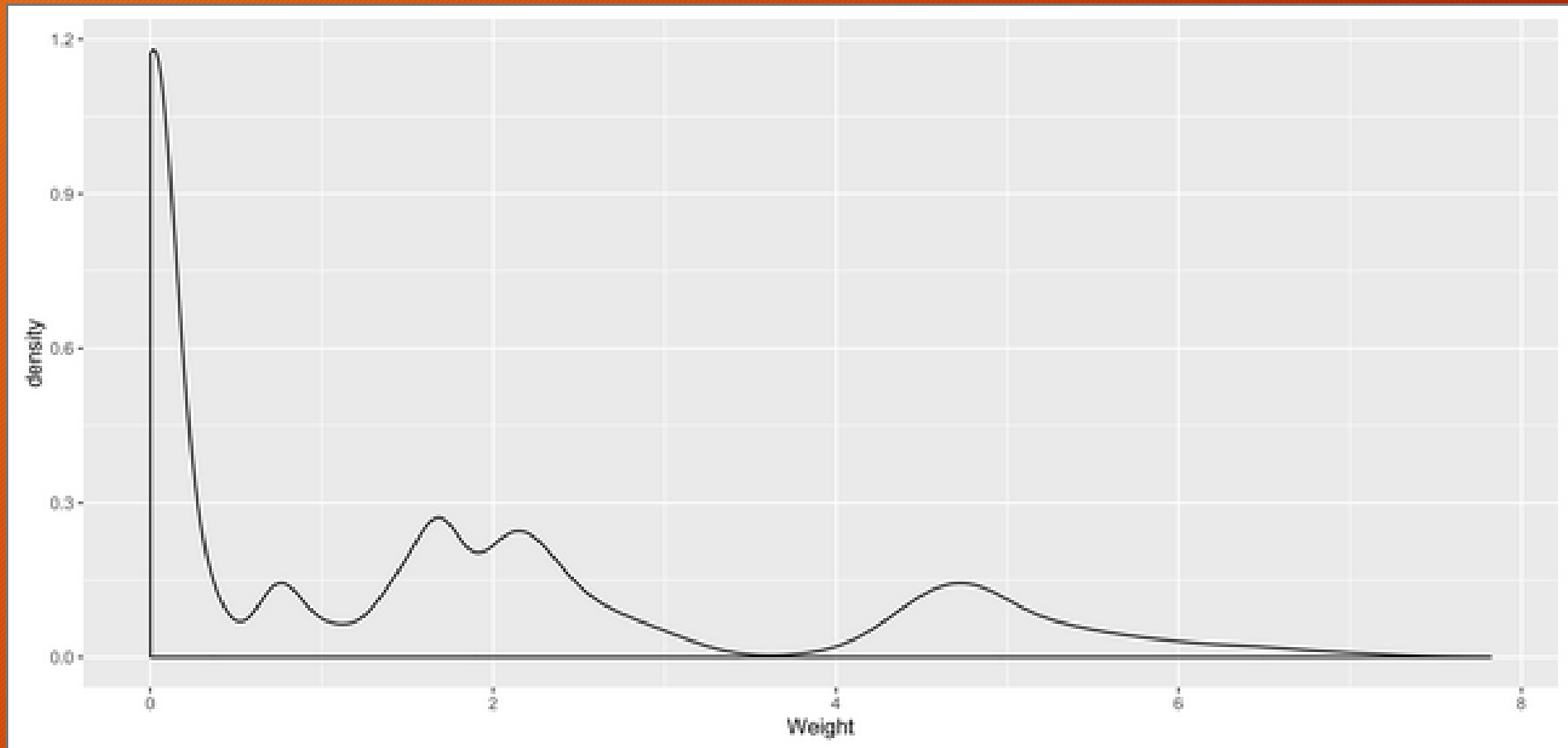# Histograms [1]
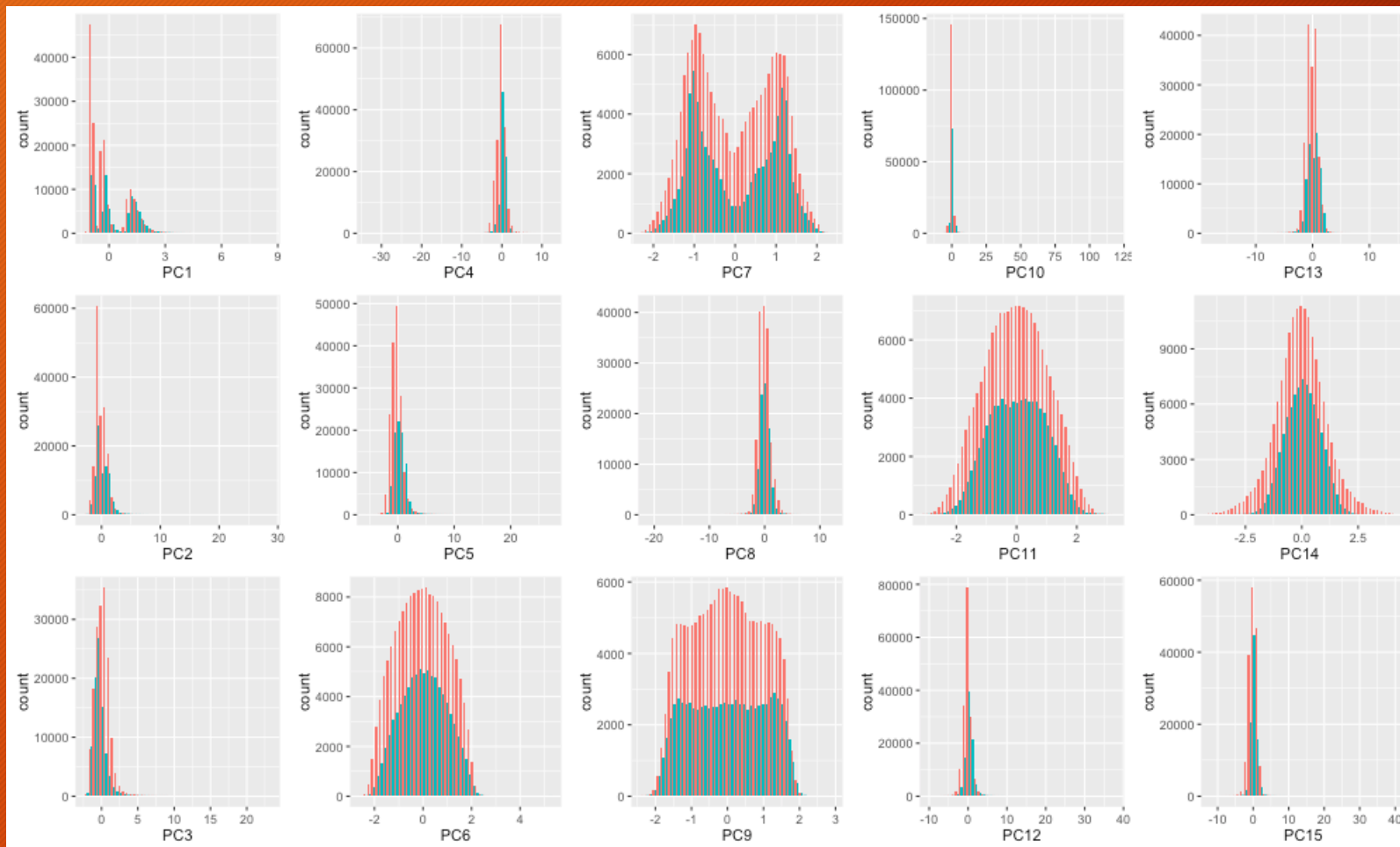
Histograms [2]

Density Plot [1]
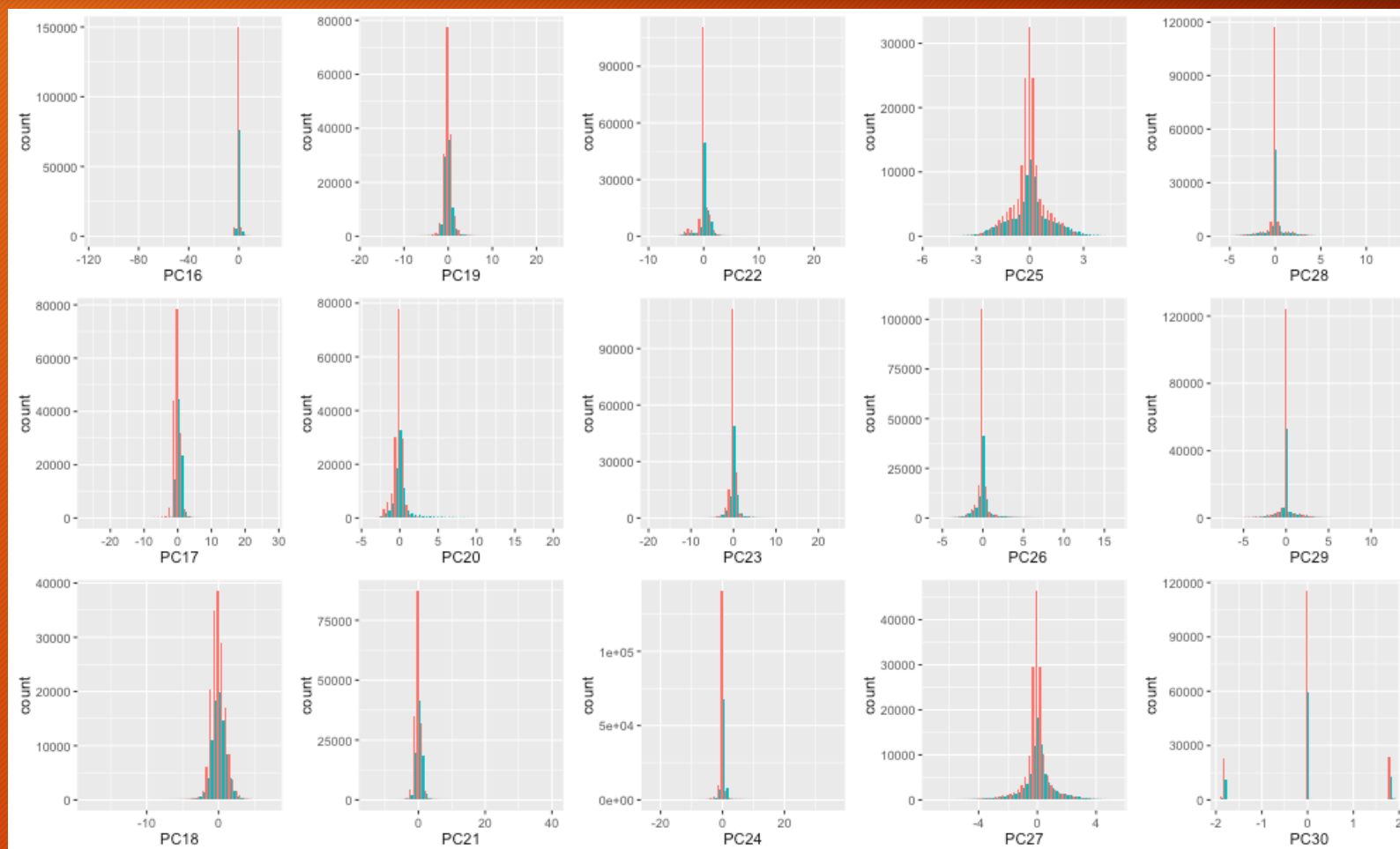
Density Plot 2

s & b Weight Density Plot

Variance Inflation Factors

VIF	Status of predictors
VIF = 1	Not correlated
1 < VIF < 5	Moderately correlated
VIF > 5 to 10	Highly correlated

	Variables	VIF
1	EventId	1.008405e+00
22	PRI_met_phi	1.030121e+00
17	PRI_tau_phi	1.087984e+00
20	PRI_lep_phi	1.099111e+00
16	PRI_tau_eta	1.405905e+00
19	PRI_lep_eta	1.416351e+00
10	DER_pt_tot	1.790023e+00
2	DER_mass_MMC	1.948378e+00
13	DER_met_phi_centrality	2.038035e+00
3	DER_mass_transverse_met_lep	2.782868e+00
32	Weight	2.808043e+00
4	DER_mass_vis	3.810547e+00
9	DER_deltar_tau_lep	4.156143e+00
21	PRI_met	4.399020e+00
12	DER_pt_ratio_lep_tau	6.172214e+00
23	PRI_met_sumet	6.217087e+00
5	DER_pt_h	1.313237e+01
7	DER_mass_jet_jet	3.899132e+01
24	PRI_jet_num	3.981534e+01
25	PRI_jet_leading_pt	3.135464e+03
28	PRI_jet_subleading_pt	4.729789e+03
26	PRI_jet_leading_eta	7.179173e+04
27	PRI_jet_leading_phi	7.235446e+04
8	DER_prodeta_jet_jet	1.589050e+05
29	PRI_jet_subleading_eta	1.797135e+05
30	PRI_jet_subleading_phi	2.218772e+05
6	DER_deltaeta_jet_jet	1.365005e+06
14	DER_lep_eta_centrality	2.235745e+06
15	PRI_tau_pt	1.433504e+09
18	PRI_lep_pt	1.558959e+09
31	PRI_jet_all_pt	2.853603e+10
11	DER_sum_pt	3.968489e+10
33	Label	NA

Variable Importance Plots