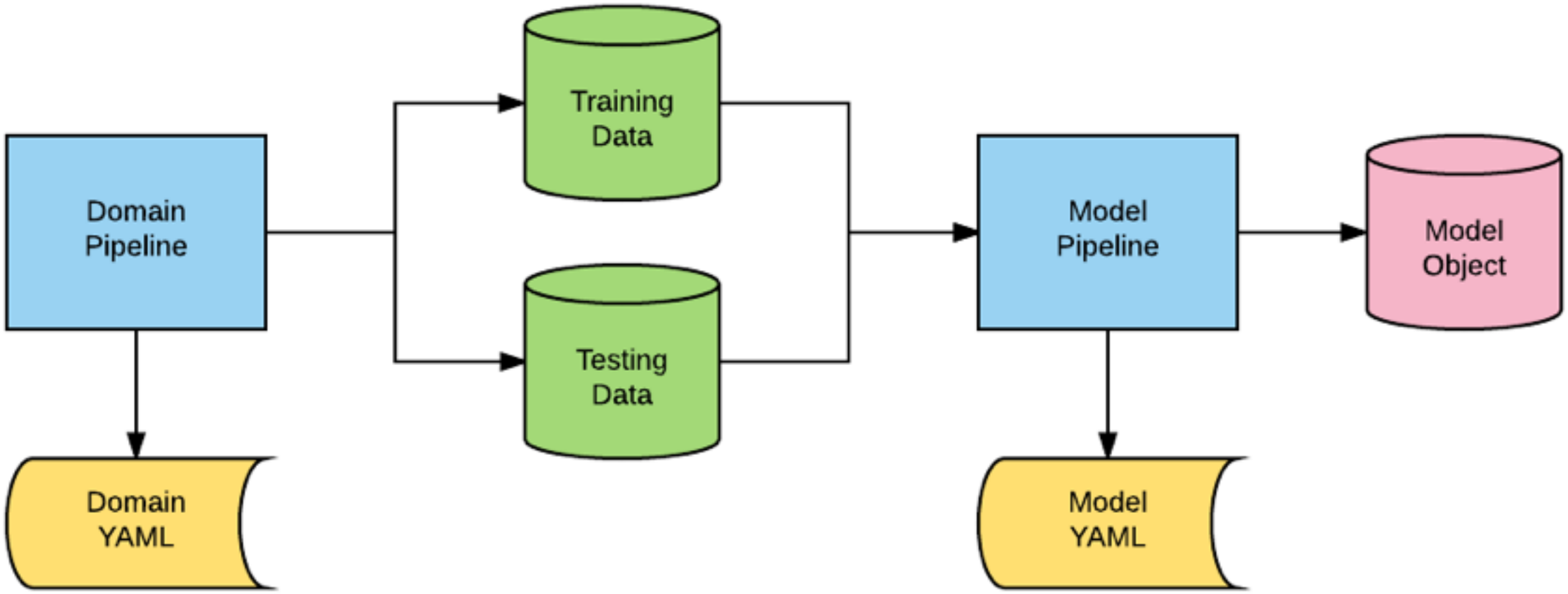
AlphaPy

A Data Science Pipeline in Python

The Framework



The Framework

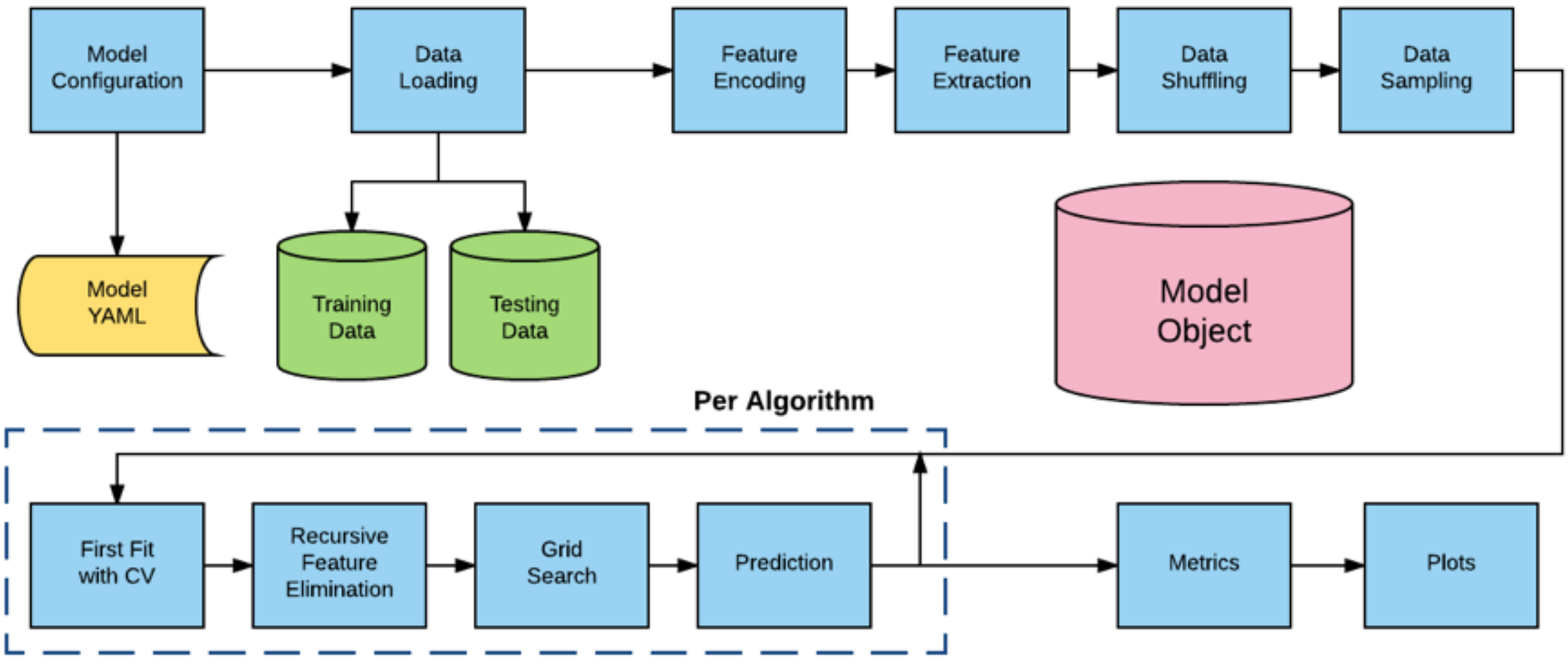
- The Model Pipeline is the common code that will generate a model for any classification or regression problem.

- The Domain Pipeline is the code required to generate the training and test data; it transforms raw data from a feed or database into canonical form.

- Both pipelines have configuration files using the YAML format.

- The output of the whole process is a Model Object, which is persistent; it can be saved and loaded for analysis.

Model Pipeline



Model Pipeline

```
[09/08/16 09:23:24] INFO         Running Model
[09/08/16 09:23:24] INFO         Loading Data
[09/08/16 09:23:24] INFO         Loading data from /Users/markconway/Projects/AlphaPy_Games/NBA/train.csv
[09/08/16 09:23:25] INFO         Found target won_on_spread in data frame
[09/08/16 09:23:25] INFO         Dropping target won_on_spread from data frame
[09/08/16 09:23:25] INFO         Loading Data
[09/08/16 09:23:25] INFO         Loading data from /Users/markconway/Projects/AlphaPy_Games/NBA/test.csv
[09/08/16 09:23:25] INFO         Found target won_on_spread in data frame
[09/08/16 09:23:25] INFO         Dropping target won_on_spread from data frame
[09/08/16 09:23:25] INFO         Original Feature Statistics
[09/08/16 09:23:25] INFO         Number of Training Rows    : 489
[09/08/16 09:23:25] INFO         Number of Training Columns : 117
[09/08/16 09:23:25] INFO         Number of Testing Rows     : 827
[09/08/16 09:23:25] INFO         Number of Testing Columns  : 117
[09/08/16 09:23:25] INFO         Unique Values for won_on_spread : [0 1]
[09/08/16 09:23:25] INFO         Unique Counts for won_on_spread : [259 230]
[09/08/16 09:23:25] INFO         Dropping Features: ['Unnamed: 0', 'index', 'season', 'date', 'away.team', 'awa
e', 'total_points', 'over', 'point_margin_game', 'cover_margin_game', 'lost_on_spread', 'under', 'overunder_ma
points']
[09/08/16 09:23:25] INFO         Removing Duplicate Columns
[09/08/16 09:23:25] INFO         Identifying Duplicate Columns
[09/08/16 09:23:25] INFO         Duplicate Columns ['home.ties', 'away.ties']
[09/08/16 09:23:25] INFO         Removed 2 Duplicate Columns
[09/08/16 09:23:25] INFO         Removing Constant Columns: ['delta.ties']
[09/08/16 09:23:25] INFO         Removed 1 Constant Features
[09/08/16 09:23:25] INFO         Original Features : Index(['line', 'over_under', 'home.point_win_streak', 'hom
       'home.losses', 'home.cover_margin_ngames_avg',
       'home.cover_margin_ngames', 'home.point_margin_streak',
       'home.under_streak', 'home.cover_margin_game',
       'home.point_margin_ngames', 'home.overunder_season',
       'home.cover_margin_season', 'home.cover_margin_season_avg',
       'home.over_streak', 'home.overunder_season_avg',
       'home.overunder_ngames_avg', 'home.overunder_streak',
       'home.days_since_previous_game', 'home.wins',
```

Components

- **scikit-learn**: machine learning in Python
- **pandas**: Python data analysis library
- **NumPy**: numerical computing package for Python
- **spark-sklearn**: scikit-learn integration package for Spark
- **skflow**: scikit-learn wrapper for Google TensorFlow
- **SciPy**: scientific computing library for Python
- **imbalanced-learn**: resampling techniques for imbalanced data sets
- **xgboost**: scaleable and distributed gradient boosting

Predictive Models

- *Binary Classification*: Classify elements of a given set into two groups.

- *Multiclass Classification*: Classify elements of a given set into greater than two or more groups.

- *Regression*: Predict real values of a given set.

Classification

- AdaBoost
- Extra Trees
- Google TensorFlow
- Gradient Boosting
- K-Nearest Neighbors
- Logistic Regression
- Support Vector Machine (including Linear)
- Naive Bayes (including Multinomial)
- Radial Basis Functions
- Random Forests
- XGBoost Binary and Multiclass

Regression

- Extra Trees
- Gradient Boosting
- K-Nearest Neighbor
- Linear Regression
- Random Forests
- XGBoost

Data Sampling

- Different techniques to handle unbalanced classes
- Undersampling
- Oversampling
- Combined Sampling (SMOTE)
- Ensemble Sampling

Feature Engineering

- Imputation
- Row Statistics and Distributions
- Clustering and PCA
- Standard Scaling (e.g., mean-centering)
- Interactions (n-way)
- Treatments (see below)

Treatments

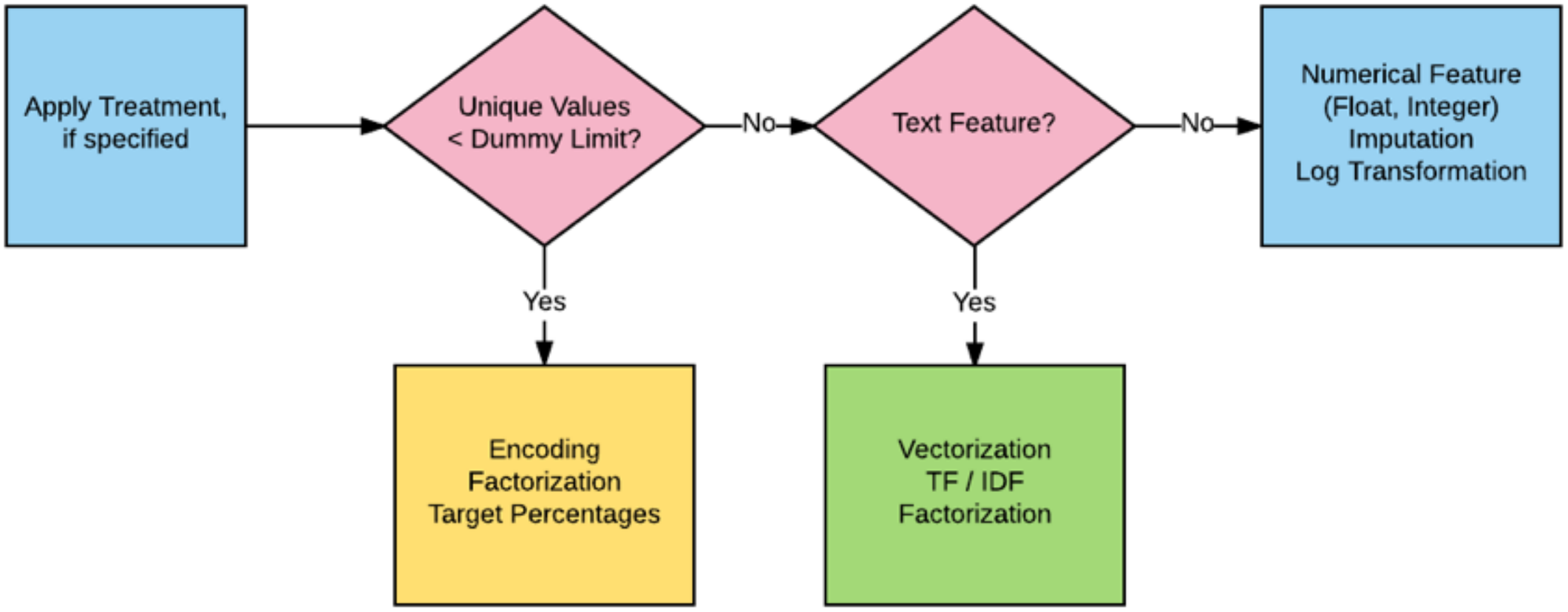
- Some features require special treatment, for example, a date column that is split into separate columns for month, day, and year.

- Treatments are specified in the configuration file with the feature name, the treatment function, and its parameters.

- In the following example, we apply a runs test to 6 features in the YAML file:

```
76
77  treatments:
78      ll                  : ['runs_test', ['all'], 18]
79      hh                  : ['runs_test', ['all'], 18]
80      hl                  : ['runs_test', ['all'], 18]
81      lh                  : ['runs_test', ['all'], 18]
82      lo                  : ['runs_test', ['all'], 18]
83      ho                  : ['runs_test', ['all'], 18]
84
```

Feature Transformation



Feature Processing

```
[09/12/16 12:52:23] INFO         Feature 18: v18 is a numerical feature of type float64 with 128999 unique values
[09/12/16 12:52:23] INFO         Feature 19: v19 is a numerical feature of type float64 with 128984 unique values
[09/12/16 12:52:23] INFO         Feature 20: v20 is a numerical feature of type float64 with 128945 unique values
[09/12/16 12:52:23] INFO         Feature 20: v20 is not normally distributed [p-value: 0.000000]
[09/12/16 12:52:24] INFO         Feature 21: v21 is a numerical feature of type float64 with 227529 unique values
[09/12/16 12:52:24] INFO         Feature 21: v21 is not normally distributed [p-value: 0.000000]
[09/12/16 12:52:24] INFO         Feature 22: v22 is a special treatment with 23420 unique values
[09/12/16 12:52:24] INFO         Applying function cvectorize to feature v22
[09/12/16 12:52:24] INFO         Function Call: cvectorize(df, 'v22', 2)
[09/12/16 12:52:31] INFO         Feature 22: v22 is a text feature [1:4] with 23420 unique values
[09/12/16 12:52:31] INFO         Feature 22: v22 => Factorization
[09/12/16 12:52:32] INFO         Feature 23: v23 is a numerical feature of type float64 with 127302 unique values
[09/12/16 12:52:33] INFO         Feature 24: v24 is a factor of type object with 5 unique values
[09/12/16 12:52:33] INFO         Encoding: Encoders.factorize
[09/12/16 12:52:33] INFO         Calculating target percentages
[09/12/16 12:52:34] INFO         Feature 25: v25 is a numerical feature of type float64 with 131388 unique values
[09/12/16 12:52:35] INFO         Feature 26: v26 is a numerical feature of type float64 with 129001 unique values
[09/12/16 12:52:36] INFO         Feature 27: v27 is a numerical feature of type float64 with 129000 unique values
[09/12/16 12:52:37] INFO         Feature 28: v28 is a numerical feature of type float64 with 128982 unique values
[09/12/16 12:52:38] INFO         Feature 29: v29 is a numerical feature of type float64 with 128999 unique values
[09/12/16 12:52:39] INFO         Feature 30: v30 is a factor of type object with 8 unique values
[09/12/16 12:52:39] INFO         Encoding: Encoders.factorize
[09/12/16 12:52:39] INFO         Calculating target percentages
```

Encoders

- Factorization
- One-Hot
- Ordinal
- Binary
- Helmert Contrast
- Sum Contrast
- Polynomial Contrast
- Backward Difference Contrast
- Simple Hashing

Feature Selection

- Univariate selection based on the percentile of highest feature scores

- Scoring functions for both classification and regression, e.g., ANOVA F-value or chi-squared statistic

- Recursive Feature Elimination (RFE) with Cross-Validation (CV) with configurable scoring function and step size

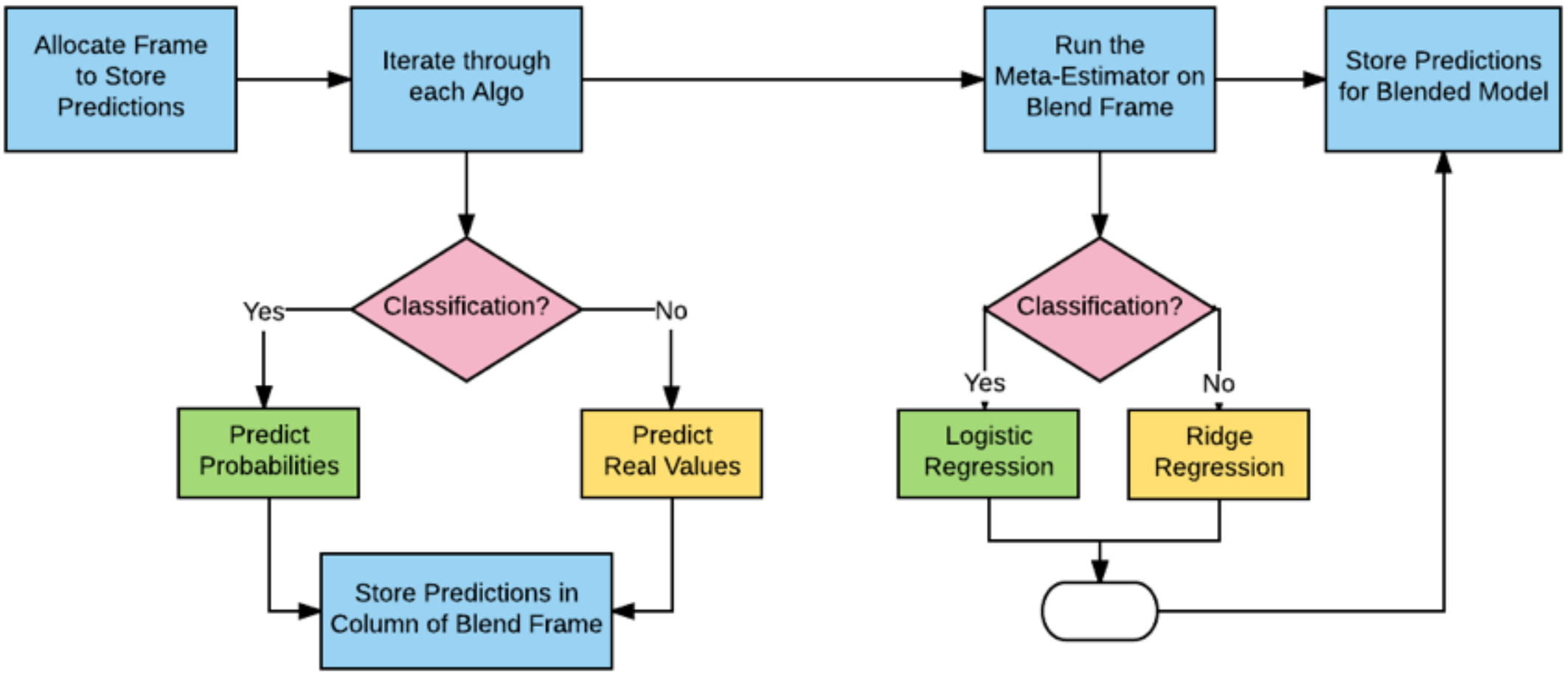
Grid Search

- Full or Randomized Distributed Grid Search with subsampling (Spark if available)

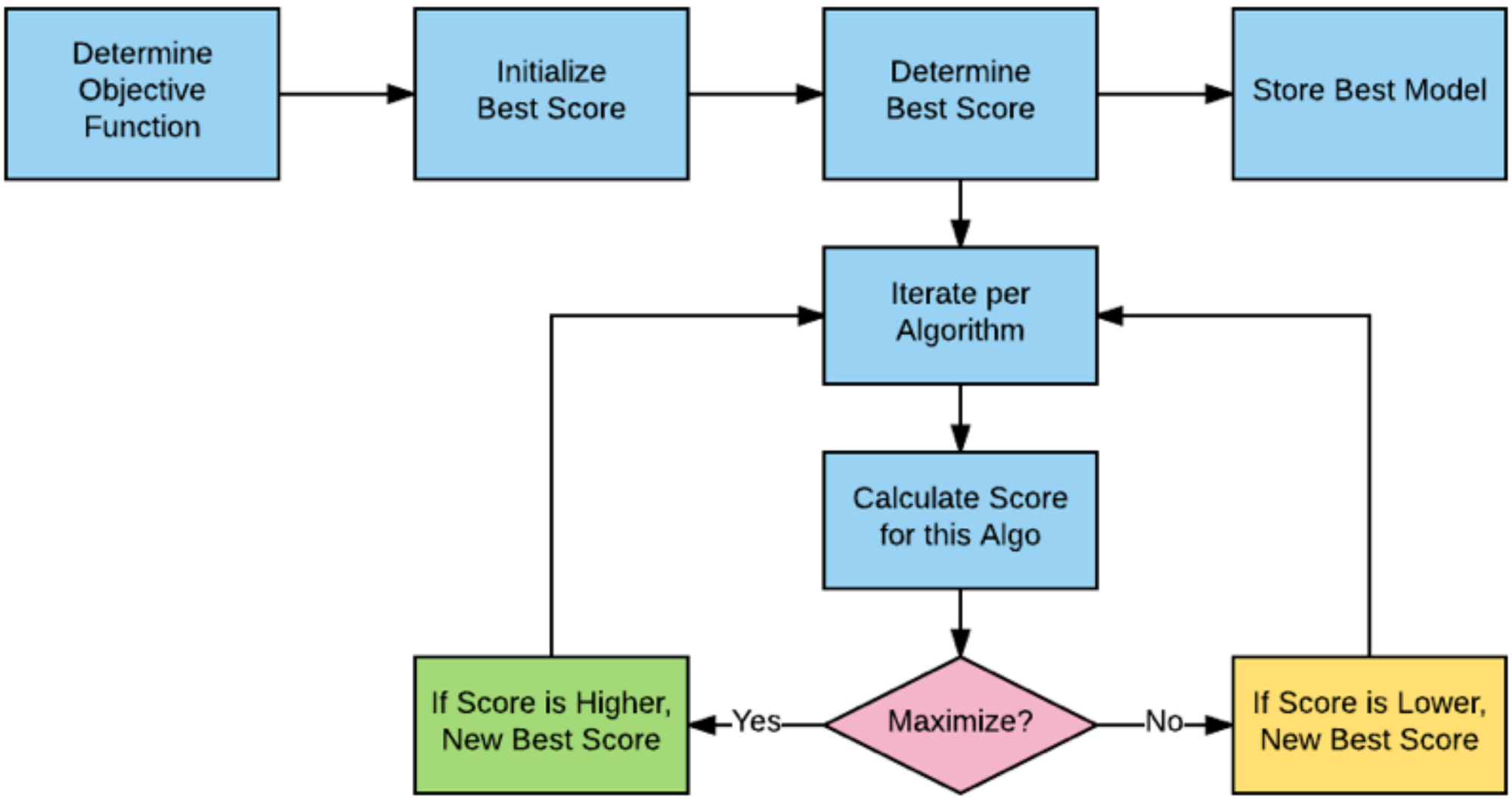
```
    # XGBoost Binary
    algo = 'XGB'
    model_type = ModelType.classification
    params = {"objective" : 'binary:logistic',
              "n_estimators" : n_estimators,
              "seed" : seed,
              "max_depth" : 5,
              "learning_rate" : 0.02,
              "min_child_weight" : 1.0,
              "subsample" : 0.7,
              "colsample_bytree" : 0.7,
              "nthread" : n_jobs,
              "silent" : True}
    est = xgb.XGBClassifier(**params)
    grid = {"n_estimators" : [21, 51, 101, 201, 501, 1001],
            "max_depth" : [5, 6, 7, 8, 9, 10, 11, 12, 15, 20],
            "learning_rate" : [0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.07, 0.1],
            "min_child_weight" : [1.0, 1.1],
            "subsample" : [0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
            "colsample_bytree" : [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]}
    scoring = False
    estimators[algo] = Estimator(algo, model_type, est, grid, scoring)
```

Ensembles

Blending and Stacking



Best Model Selection



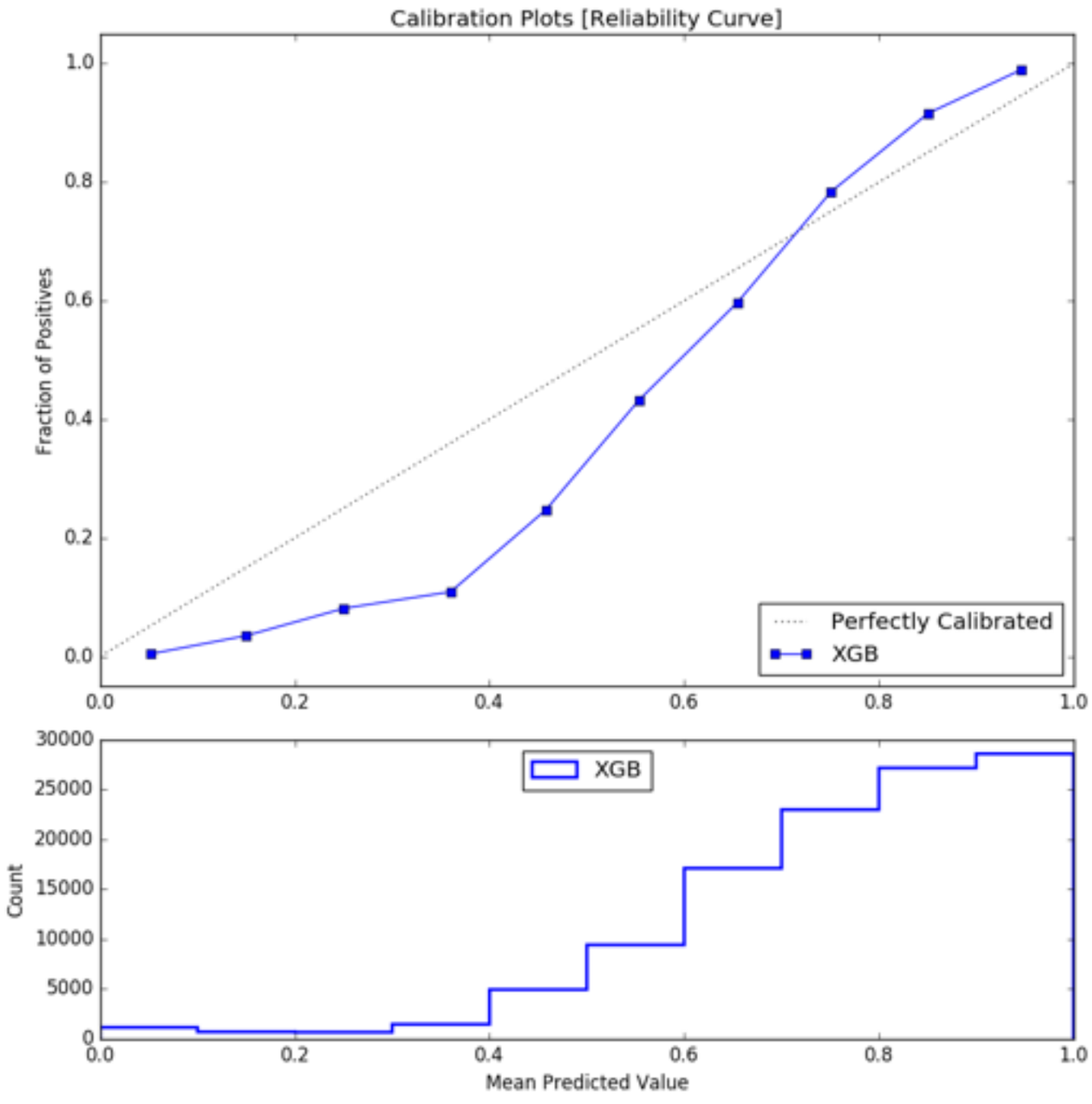
Model Evaluation

- Metrics
- Calibration Plot
- Confusion Matrix
- Learning Curve
- ROC Curve

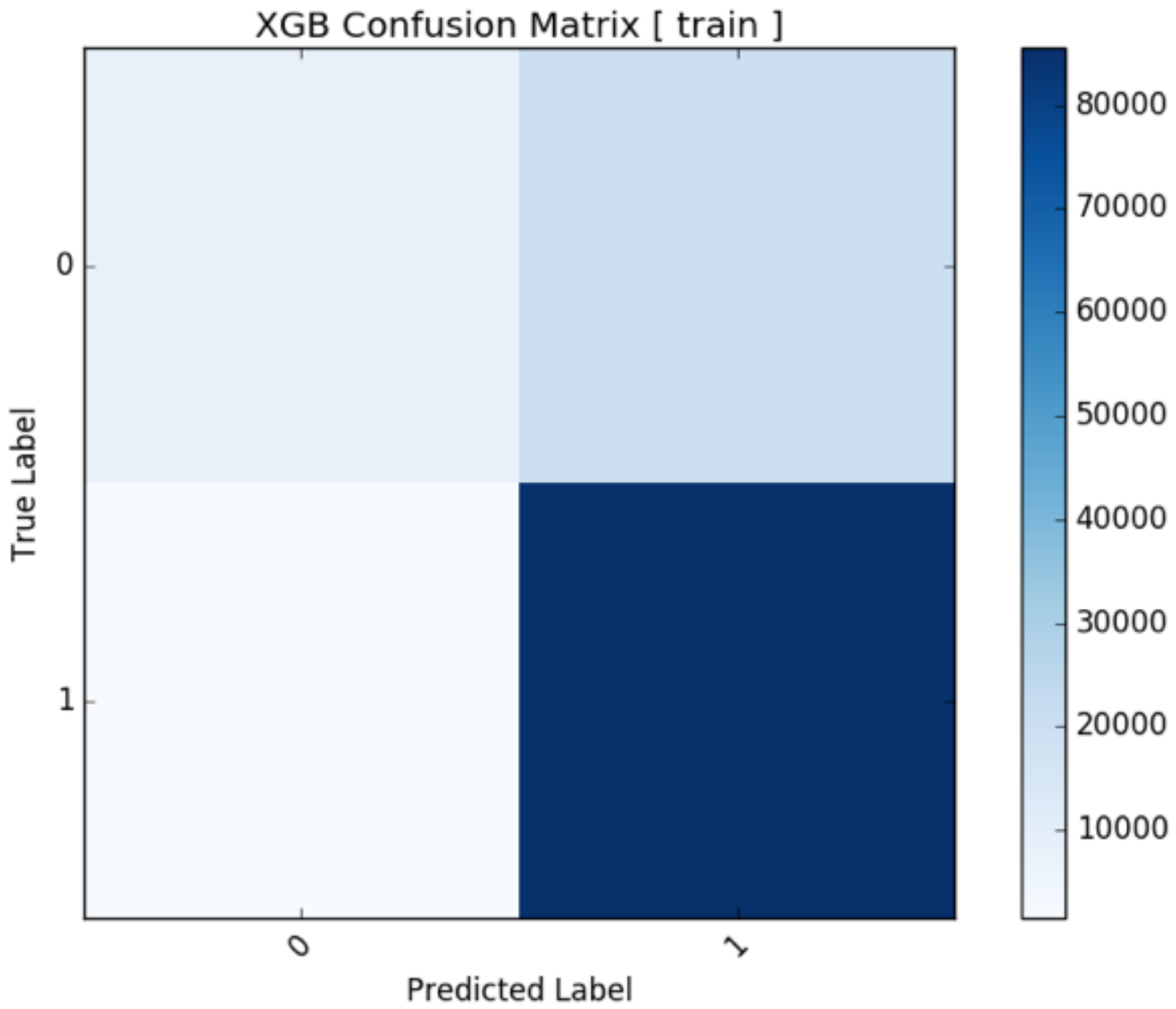
Metrics

```
[09/08/16 09:23:29] INFO         ----------------------------------------------------------------------
[09/08/16 09:23:29] INFO         Metrics for Partition: train
[09/08/16 09:23:29] INFO         ----------------------------------------------------------------------
[09/08/16 09:23:29] INFO         Algorithm: XGB
[09/08/16 09:23:29] INFO         accuracy: 0.993865030675
[09/08/16 09:23:29] INFO         adjusted_rand_score: 0.9755605847800342
[09/08/16 09:23:29] INFO         average_precision: 0.999905926116
[09/08/16 09:23:29] INFO         confusion_matrix: [[259   0]
 [  3 227]]
[09/08/16 09:23:29] INFO         explained_variance: 0.975524592916
[09/08/16 09:23:29] INFO         f1: 0.993435448578
[09/08/16 09:23:29] INFO         log_loss: 0.150030662974
[09/08/16 09:23:29] INFO         mean_absolute_error: 0.00613496932515
[09/08/16 09:23:29] INFO         mean_squared_error: 0.00613496932515
[09/08/16 09:23:29] INFO         median_absolute_error: 0.0
[09/08/16 09:23:29] INFO         precision: 1.0
[09/08/16 09:23:29] INFO         r2: 0.975373510156
[09/08/16 09:23:29] INFO         recall: 0.986956521739
[09/08/16 09:23:29] INFO         roc_auc: 0.999916065133
```

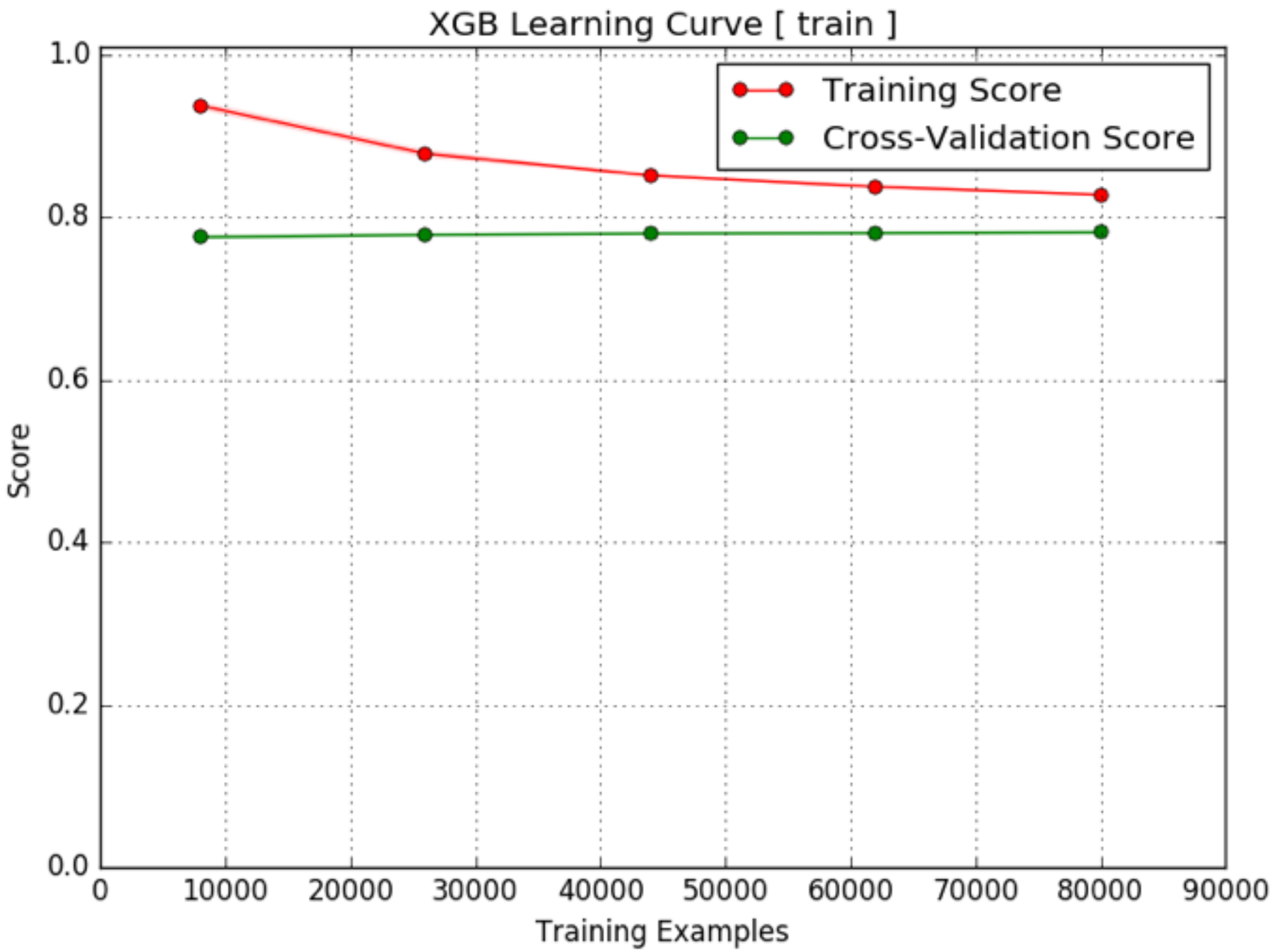
Calibration Plot



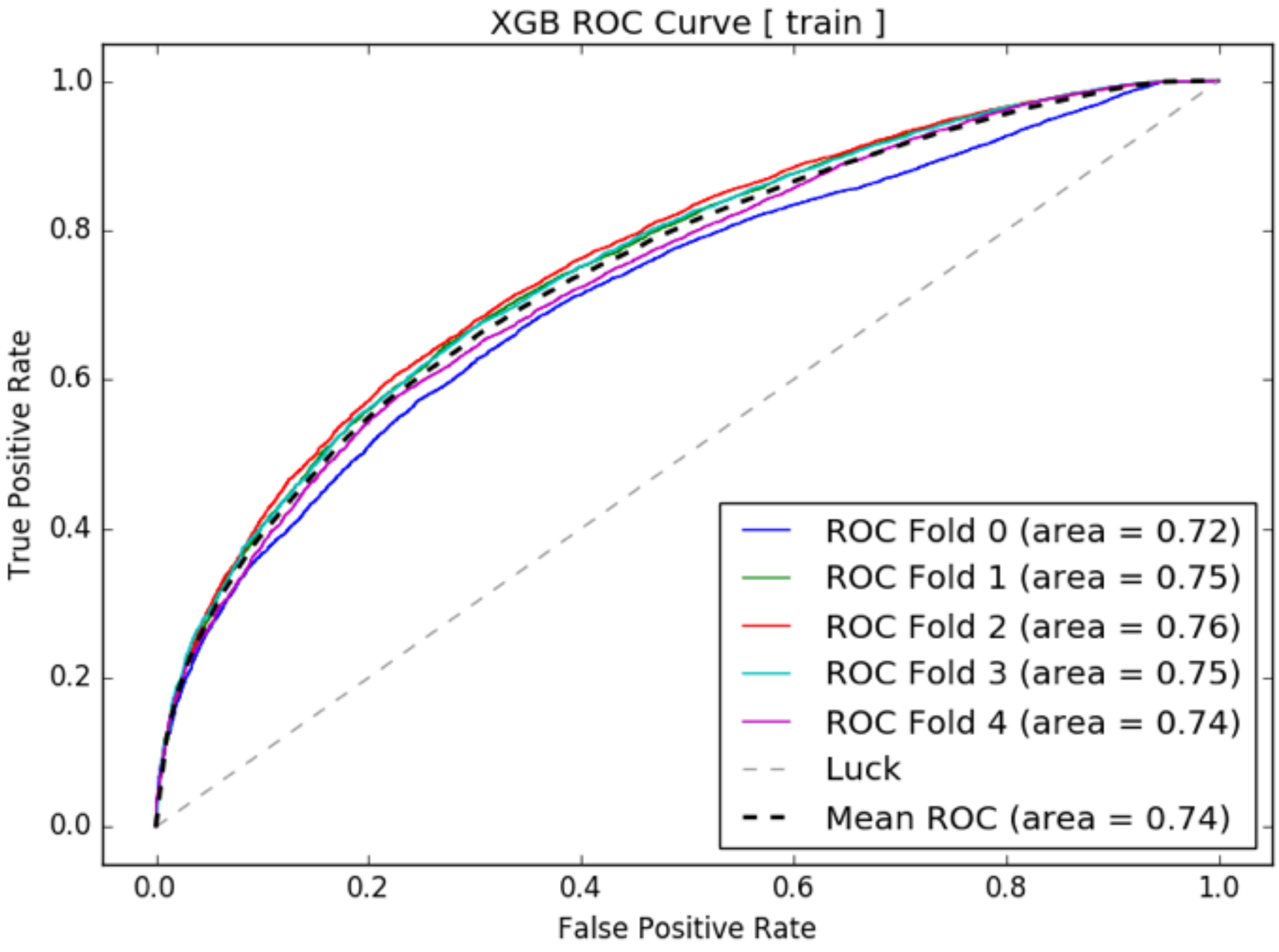
Confusion Matrix



Learning Curve



ROC Curve


XGB ROC Curve [train]
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Fold 0 (area = 0.72)
ROC Fold 1 (area = 0.75)
ROC Fold 2 (area = 0.76)
ROC Fold 3 (area = 0.75)
ROC Fold 4 (area = 0.74)
Luck
Mean ROC (area = 0.74)

Domains

- A domain encapsulates functionality that operates on specific kinds of schema or subject matter.

- The purpose of the Domain Pipeline is to prepare data for the Model Pipeline. For example, both market and sports data are time series data but must be structured differently for prediction.

- Market data consist of standard primitives such as *open*, *high*, *low*, and *close; t*he latter three are postdictive and cause data leakage. Leaders and laggards must be identified and possibly column-shifted, which is handled by the Model Pipeline.

- Sports data are typically structured into a match or game format after gathering team and player data.

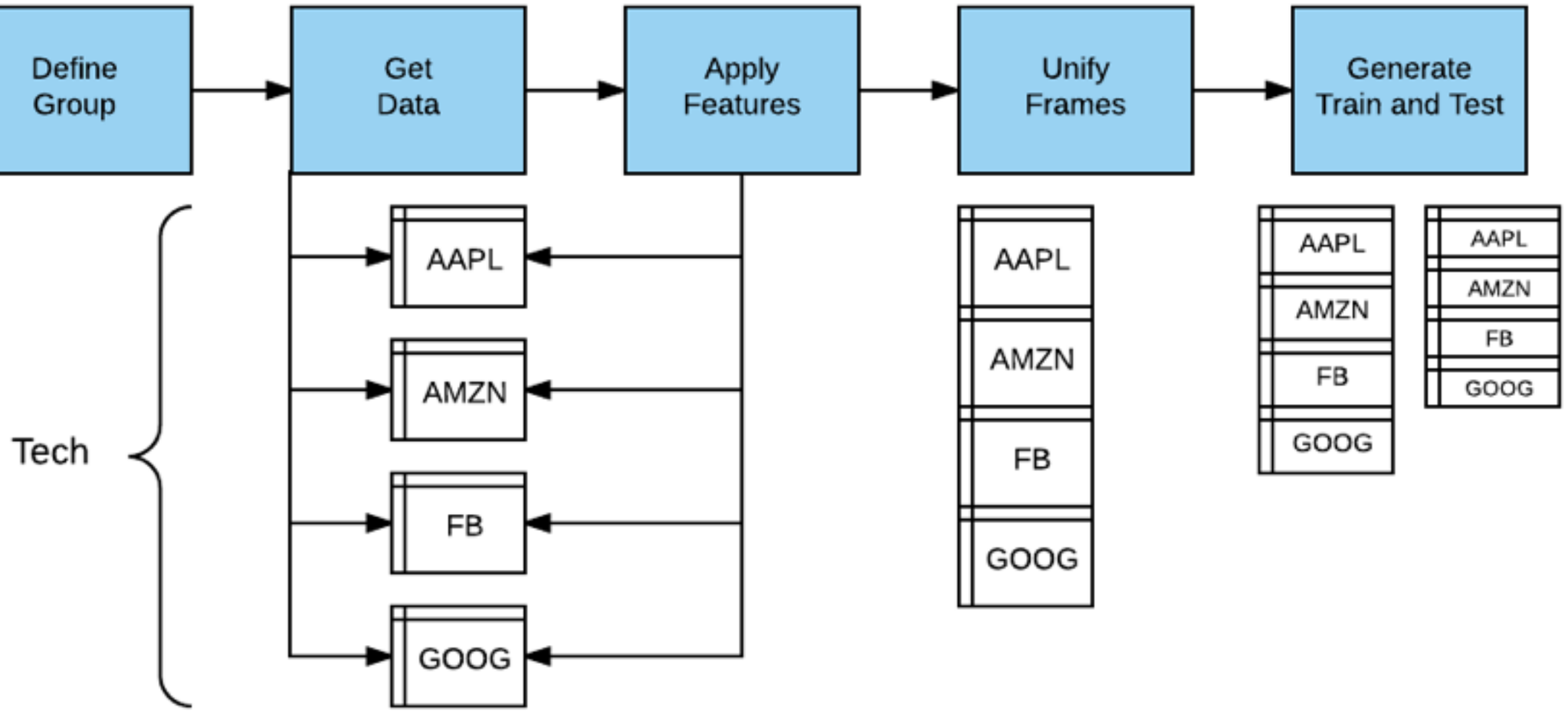
Market Domain

1. Suppose we have five years of history for a group of stocks, each stock represented by rows of time series data on a daily basis.

2. We want to create a model that predicts whether or not a stock will generate a given return over the next n days, where n = the forecast period.

3. The goal is to generate canonical training and test data for the model pipeline, so we need to apply a series of transformations to the raw stock data.

Market Pipeline



Feature Definition Language

- Along with treatments, we defined a Feature Definition Language (FDL) that would make it easy for data scientists to define formulas and functions.

- Features are applied to groups, so feature sets are uniformly applied across multiple frames.

- The features are represented by variables, and these variables map to functions with parameters.

FDL Example

- Suppose we want to use the 50-day moving average (MA) in our model, as we believe that it has predictive power for a stock's direction.

- The moving average function *ma* has two parameters: a feature (column) name and a time period.

- To apply the 50-day MA, we can simply join the function name with its parameters, separated by "_", or *ma_close_50*.

- If we want to use an alias, then we can define *cma* to be the equivalent of *ma_close* and get *cma_50*.

Stock Pipeline Example

- Develop a model to predict days with ranges that are greater than average.
- We will use both random forests and gradient boosting.
- Get daily data from Yahoo over the past few years to train our model.
- Define technical analysis features with FDL.

Stocks Configuration

Untitled — Edited

conda_alphapy.txt | (alphapy) localhost:AlphaPy_C | feature engineering.txt | game.yml | model.yml — AlphaPy_Games | stock.yml | model.yml — AlphaPy_Stocks | model.yml — BNP Claims | model.yml — Santander

```yaml
stock:
    forecast_period : 5
    fractal         : 1d
    leaders         : ['open', 'gap', 'gapbadown', 'gapbaup', 'gapdown', 'gapup']
    lookback_period : 1400
    predict_date    : 2016-01-01
    schema          : prices
    train_date      : 1900-01-01
    target_group    : tech

groups:
    etf  : ['dia', 'dust', 'edc', 'edz', 'eem', 'ewh', 'ewt', 'fas', 'faz',
            'gld', 'hyg', 'iwm', 'ivv', 'iwf', 'jnk', 'mdy', 'nugt', 'qqq',
            'sds', 'smh', 'spy', 'tbt', 'tlt', 'tna', 'tvix', 'tza', 'upro',
            'uvxy', 'vwo', 'vxx', 'xhb', 'xiv', 'xle', 'xlf', 'xlk', 'xlp',
            'xlu', 'xlv', 'xme']
    tech : ['aapl', 'adbe', 'amat', 'amgn', 'amzn', 'bidu', 'brcd', 'bvsn',
            'csco', 'ddd', 'emc', 'expe', 'fb', 'fit', 'fslr', 'goog',
            'intc', 'isrg', 'lnkd', 'msft', 'nflx', 'pcln', 'qcom', 'qqq',
            'tsla', 'twtr', 'yhoo']

features: ['abovema_close', 'adx', 'atr', 'bigdown', 'bigup', 'cma_5', 'cma_10',
           'cma_20', 'cma_50', 'diminus', 'diplus', 'gap', 'gapbadown', 'gapbaup',
           'gapdown', 'gapup', 'hc', 'hh', 'ho', 'hl', 'lc', 'lh', 'll', 'lo',
           'madelta', 'net', 'netdown', 'netup', 'nr_4', 'nr_7', 'nr_10', 'roi',
           'roi_2', 'roi_3', 'roi_4', 'roi_5', 'roi_10', 'roi_20', 'rr', 'rr_2',
           'rr_3', 'rr_4', 'rr_5', 'rr_6', 'rr_7', 'rrunder', 'rrover', 'rsi_8',
           'rsi_14', 'sepa', 'sepa_2', 'sepa_3', 'sepa_4', 'sepa_5', 'sepa_6',
           'sepa_7', 'sephigh', 'seplow', 'sepover', 'sepunder', 'trend', 'vma',
           'vmover', 'vmratio', 'vmunder', 'volatility', 'wr_4', 'wr_7', 'wr_10']

aliases:
    atr        : 'ma_truerange'
    cma        : 'ma_close'
    cmax       : 'highest_close'
    cmin       : 'lowest_close'
```

Line 2, Column 24 — Spaces: 4 — YAML

Stocks Configuration

```
65
66  variables:
67      abovema     : 'close > cma_50'
68      belowma     : 'close < cma_50'
69      bigup       : 'rrover & sepwide & netup'
70      bigdown     : 'rrover & sepwide & netdown'
71      madelta     : 'close - cma_50'
72      nr          : 'hlrange == rmin_7'
73      roihigh     : 'roi_5 >= 5'
74      roilow      : 'roi_5 < -5'
75      rrunder     : 'rr_1_10 < 1'
76      rrover      : 'rr_1_10 >= 1'
77      sep         : 'rixc - rixo'
78      sephigh     : 'sep >= 70'
79      seplow      : 'sep <= -70'
80      sepwide     : 'sephigh | seplow'
81      sepnothigh  : 'sep <= 30'
82      sepnotlow   : 'sep >= -30'
83      sepnarrow   : 'sepnothigh & sepnotlow'
84      trend       : 'rrover & sepwide'
85      vmover      : 'vmratio >= 1'
86      vmunder     : 'vmratio < 1'
87      volatility  : 'close / atr_10'
88      wr          : 'hlrange == rmax_4'
89      xmadown     : 'cma_20 < cma_50 & cma_20[1] > cma_50[1]'
90      xmaup       : 'cma_20 > cma_50 & cma_20[1] < cma_50[1]'
91
```

Model Configuration

- Data Section

```
6
7  data:
8      drop              : ['Unnamed: 0', 'date', 'open', 'high', 'low', 'close', 'adjclose', 'tag']
9      dummy_limit       : 100
10     features          : '*'
11     sampling          :
12         option        : False
13         method        : under_random
14         ratio         : 0.0
15     sentinel          : -1
16     separator         : ','
17     shuffle           : False
18     split             : 0.4
19     target            : rrover
20     target_value      : True
21     test              : test
22     test_labels       : True
23     train             : train
24
```

Model Configuration

- Model Section

```
25
26  model:
27      algorithms          : ['RF', 'XGB']
28      balance_classes     : True
29      calibration         :
30          option          : False
31          type            : sigmoid
32      cv_folds            : 5
33      estimators          : 501
34      feature_selection   :
35          option          : False
36          percentage      : 10
37          uni_grid        : [5, 10, 15, 20, 25]
38          score_func      : f_classif
39      grid_search         :
40          option          : False
41          iterations      : 100
42          random          : True
43          subsample       : True
44          sampling_pct    : 0.25
45      pvalue_level        : 0.01
46      rfe                 :
47          option          : False
48          step            : 10
49      scoring_function    : roc_auc
50      type                : classification
51
```

Model Configuration

- Features Section

```
51
52  features:
53      clustering          :
54          option          : False
55          increment       : 3
56          maximum         : 30
57          minimum         : 3
58      encoding            :
59          rounding        : 3
60          type            : factorize
61      genetic             :
62          option          : False
63          features        : 20
64      interactions        :
65          option          : True
66          poly_degree     : 2
67          sampling_pct    : 10
68      pca                 :
69          option          : True
70          increment       : 3
71          maximum         : 15
72          minimum         : 3
73          whiten          : False
74      text                :
75          ngrams          : 1
76          vectorize       : False
77
```

Model Configuration

- Other Sections

```
77
78  treatments:
79      ll                  : ['runs_test', ['all'], 18]
80      hh                  : ['runs_test', ['all'], 18]
81      hl                  : ['runs_test', ['all'], 18]
82      lh                  : ['runs_test', ['all'], 18]
83      lo                  : ['runs_test', ['all'], 18]
84      ho                  : ['runs_test', ['all'], 18]
85
86  pipeline:
87      number_jobs         : -1
88      seed                : 10231
89      verbosity           : 10
90
91  plots:
92      calibration         : True
93      confusion_matrix    : True
94      importances         : True
95      learning_curve      : True
96      roc_curve           : True
97
98  xgboost:
99      stopping_rounds     : 30
100
```

Start Stocks Pipeline

```
(alphapy) localhost:AlphaPy_Stocks markconway$ python ../AlphaPy/alpha_stock.py
None
[09/12/16 21:44:53] INFO        ********************************************************************************
[09/12/16 21:44:53] INFO        START STOCK PIPELINE
[09/12/16 21:44:53] INFO        ********************************************************************************
[09/12/16 21:44:53] INFO        Stock Configuration
[09/12/16 21:44:53] INFO        Getting Features
[09/12/16 21:44:53] INFO        Defining Groups
[09/12/16 21:44:53] INFO        Added: {'qqq', 'qcom', 'yhoo', 'amzn', 'brcd', 'ddd', 'goog', 'bidu', 'pcln', 'int
tsla', 'twtr', 'amgn', 'lnkd', 'bvsn', 'fb', 'csco', 'emc', 'adbe', 'fit', 'fslr', 'msft', 'aapl', 'isrg'}
[09/12/16 21:44:53] INFO        Added: {'edc', 'tvix', 'tbt', 'gld', 'vwo', 'dust', 'eem', 'mdy', 'xlf', 'edz', 'd
z', 'xle', 'qqq', 'spy', 'smh', 'ewt', 'hyg', 'iwm', 'jnk', 'sds', 'xiv', 'nugt', 'xlp', 'upro', 'xme', 'vxx', 'xl
k', 'xlu', 'iwf', 'tlt', 'tza'}
[09/12/16 21:44:53] INFO        Defining Aliases
[09/12/16 21:44:53] INFO        Defining Variables
[09/12/16 21:44:53] INFO        STOCK PARAMETERS:
[09/12/16 21:44:53] INFO        features        = ['abovema_close', 'adx', 'atr', 'bigdown', 'bigup', 'cma_5', 'cm
iminus', 'diplus', 'gap', 'gapbadown', 'gapbaup', 'gapdown', 'gapup', 'hc', 'hh', 'ho', 'hl', 'lc', 'lh', 'll', 'l
n', 'netup', 'nr_4', 'nr_7', 'nr_10', 'roi', 'roi_2', 'roi_3', 'roi_4', 'roi_5', 'roi_10', 'roi_20', 'rr', 'rr_2',
6', 'rr_7', 'rrunder', 'rrover', 'rsi_8', 'rsi_14', 'sepa', 'sepa_2', 'sepa_3', 'sepa_4', 'sepa_5', 'sepa_6', 'sep
epover', 'sepunder', 'trend', 'vma', 'vmover', 'vmratio', 'vmunder', 'volatility', 'wr_4', 'wr_7', 'wr_10']
[09/12/16 21:44:53] INFO        forecast_period = 1
[09/12/16 21:44:53] INFO        fractal         = 1d
[09/12/16 21:44:53] INFO        leaders         = ['open', 'gap', 'gapbadown', 'gapbaup', 'gapdown', 'gapup']
[09/12/16 21:44:53] INFO        lookback_period = 1400
[09/12/16 21:44:53] INFO        predict_date    = 2016-01-01
[09/12/16 21:44:53] INFO        schema          = prices
[09/12/16 21:44:53] INFO        target_group    = tech
[09/12/16 21:44:53] INFO        train_date      = 1900-01-01
```

Get Stock Prices

```
[09/12/16 21:44:53] INFO         Creating Model
[09/12/16 21:44:53] INFO         Calling Pipeline
[09/12/16 21:44:53] INFO         All Members: {'qqq', 'qcom', 'yhoo', 'amzn', 'brcd', 'ddd', 'goog', 'bidu', 'pcln',
pe', 'tsla', 'twtr', 'amgn', 'lnkd', 'bvsn', 'fb', 'csco', 'emc', 'adbe', 'fit', 'fslr', 'msft', 'aapl', 'isrg'}
[09/12/16 21:44:54] INFO         Getting qqq data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:54] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:54] INFO         Getting qcom data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:54] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:55] INFO         Getting yhoo data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:55] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:55] INFO         Getting amzn data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:55] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:55] INFO         Getting brcd data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:55] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:56] INFO         Getting ddd data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:56] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:56] INFO         Getting goog data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:56] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:56] INFO         Getting bidu data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:56] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:56] INFO         Getting pcln data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:56] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:57] INFO         Getting intc data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:57] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:57] INFO         Getting amat data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:57] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
[09/12/16 21:44:57] INFO         Getting nflx data from 2012-11-12 21:44:54.000196 to 2016-09-12 21:44:51.162443
[09/12/16 21:44:57] INFO         Starting new HTTP connection (1): ichart.finance.yahoo.com
```

Apply Variables

```
[09/12/16 21:45:02] INFO         Applying variable: abovema_close
[09/12/16 21:45:02] INFO         Applying variable: adx
[09/12/16 21:45:16] INFO         Applying variable: atr
[09/12/16 21:45:16] INFO         Applying variable: bigdown
[09/12/16 21:45:17] INFO         Applying variable: bigup
[09/12/16 21:45:17] INFO         Applying variable: cma_5
[09/12/16 21:45:17] INFO         Applying variable: cma_10
[09/12/16 21:45:17] INFO         Applying variable: cma_20
[09/12/16 21:45:17] INFO         Applying variable: cma_50
[09/12/16 21:45:17] INFO         Applying variable: diminus
[09/12/16 21:45:17] INFO         Applying variable: diplus
[09/12/16 21:45:17] INFO         Applying variable: gap
[09/12/16 21:45:17] INFO         Applying variable: gapbadown
[09/12/16 21:45:17] INFO         Applying variable: gapbaup
[09/12/16 21:45:17] INFO         Applying variable: gapdown
[09/12/16 21:45:17] INFO         Applying variable: gapup
[09/12/16 21:45:17] INFO         Applying variable: hc
[09/12/16 21:45:17] INFO         Applying variable: hh
[09/12/16 21:45:18] INFO         Applying variable: ho
[09/12/16 21:45:18] INFO         Applying variable: hl
[09/12/16 21:45:18] INFO         Applying variable: lc
[09/12/16 21:45:18] INFO         Applying variable: lh
[09/12/16 21:45:18] INFO         Applying variable: ll
[09/12/16 21:45:18] INFO         Applying variable: lo
```

Create Train/Test Files

```
[09/12/16 21:45:33] INFO         Writing data frame to /Users/markconway/Projects/AlphaPy_Stocks/Trend Days/train.csv
[09/12/16 21:45:37] INFO         Writing data frame to /Users/markconway/Projects/AlphaPy_Stocks/Trend Days/test.csv
[09/12/16 21:45:38] INFO         Loading Data
[09/12/16 21:45:38] INFO         Loading data from /Users/markconway/Projects/AlphaPy_Stocks/Trend Days/train.csv
[09/12/16 21:45:38] INFO         Found target rrover in data frame
[09/12/16 21:45:38] INFO         Dropping target rrover from data frame
[09/12/16 21:45:38] INFO         Loading Data
[09/12/16 21:45:38] INFO         Loading data from /Users/markconway/Projects/AlphaPy_Stocks/Trend Days/test.csv
[09/12/16 21:45:38] INFO         Found target rrover in data frame
[09/12/16 21:45:38] INFO         Dropping target rrover from data frame
[09/12/16 21:45:38] INFO         Original Feature Statistics
[09/12/16 21:45:38] INFO         Number of Training Rows    : 19102
[09/12/16 21:45:38] INFO         Number of Training Columns : 91
[09/12/16 21:45:38] INFO         Number of Testing Rows     : 4674
[09/12/16 21:45:38] INFO         Number of Testing Columns  : 91
[09/12/16 21:45:38] INFO         Unique Values for rrover : [0 1]
[09/12/16 21:45:38] INFO         Unique Counts for rrover : [11089  8013]
[09/12/16 21:45:38] INFO         Dropping Features: ['Unnamed: 0', 'date', 'open', 'high', 'low', 'close', 'adjclose', 'tag']
[09/12/16 21:45:38] INFO         Removing Duplicate Columns
[09/12/16 21:45:38] INFO         Identifying Duplicate Columns
[09/12/16 21:45:39] INFO         Duplicate Columns ['netdown', 'netup', 'rr_1_10']
[09/12/16 21:45:39] INFO         Removed 3 Duplicate Columns
[09/12/16 21:45:39] INFO         Removing Constant Columns: []
[09/12/16 21:45:39] INFO         Removed 0 Constant Features
```

Transform Features

```
[09/12/16 21:45:43] INFO        Feature 41: ll is a special treatment with 2 unique values
[09/12/16 21:45:43] INFO        Applying function runs_test to feature ll
[09/12/16 21:45:43] INFO        Function Call: runs_test(df, 'll', ['all'], 18)
[09/12/16 21:45:44] INFO        Feature 41: ll is a factor of type bool with 2 unique values
[09/12/16 21:45:44] INFO        Encoding: Encoders.factorize
[09/12/16 21:45:44] INFO        Calculating target percentages
[09/12/16 21:45:44] INFO        Feature 42: lo is a special treatment with 2 unique values
[09/12/16 21:45:44] INFO        Applying function runs_test to feature lo
[09/12/16 21:45:44] INFO        Function Call: runs_test(df, 'lo', ['all'], 18)
[09/12/16 21:45:44] INFO        Feature 42: lo is a factor of type bool with 2 unique values
[09/12/16 21:45:44] INFO        Encoding: Encoders.factorize
[09/12/16 21:45:44] INFO        Calculating target percentages
[09/12/16 21:45:44] INFO        Feature 43: madelta is a numerical feature of type float64 with 23764 unique values
[09/12/16 21:45:44] INFO        Feature 44: rmin_4 is a numerical feature of type float64 with 4459 unique values
[09/12/16 21:45:44] INFO        Feature 45: nr_4 is a factor of type bool with 2 unique values
[09/12/16 21:45:44] INFO        Encoding: Encoders.factorize
[09/12/16 21:45:44] INFO        Calculating target percentages
[09/12/16 21:45:44] INFO        Feature 46: rmin_7 is a numerical feature of type float64 with 3159 unique values
[09/12/16 21:45:44] INFO        Feature 47: nr_7 is a factor of type bool with 2 unique values
[09/12/16 21:45:44] INFO        Encoding: Encoders.factorize
[09/12/16 21:45:44] INFO        Calculating target percentages
[09/12/16 21:45:45] INFO        Feature 48: rmin_10 is a numerical feature of type float64 with 2497 unique values
[09/12/16 21:45:45] INFO        Feature 49: nr_10 is a factor of type bool with 2 unique values
[09/12/16 21:45:45] INFO        Encoding: Encoders.factorize
[09/12/16 21:45:45] INFO        Calculating target percentages
[09/12/16 21:45:45] INFO        Feature 50: roi is a numerical feature of type float64 with 23319 unique values
[09/12/16 21:45:45] INFO        Feature 51: roi_2 is a numerical feature of type float64 with 23440 unique values
[09/12/16 21:45:45] INFO        Feature 52: roi_3 is a numerical feature of type float64 with 23494 unique values
[09/12/16 21:45:45] INFO        Feature 53: roi_4 is a numerical feature of type float64 with 23519 unique values
[09/12/16 21:45:45] INFO        Feature 54: roi_5 is a numerical feature of type float64 with 23561 unique values
```

Fit Model

```
[09/12/16 21:45:46] INFO         Sample Weight for target rrover [True]: 1.383876
[09/12/16 21:45:46] INFO         Getting All Estimators
[09/12/16 21:45:46] INFO         Selecting Models
[09/12/16 21:45:46] INFO         Algorithm: RF
[09/12/16 21:45:46] INFO         Fitting Initial Model
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    7.4s
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    7.5s
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    7.4s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   36.0s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   36.0s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   35.8s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   35.9s
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 501 out of 501 | elapsed:  1.7min finished
[Parallel(n_jobs=-1)]: Done 501 out of 501 | elapsed:  1.7min finished
[Parallel(n_jobs=-1)]: Done 501 out of 501 | elapsed:  1.7min finished
[Parallel(n_jobs=-1)]: Done 501 out of 501 | elapsed:  1.7min finished
```

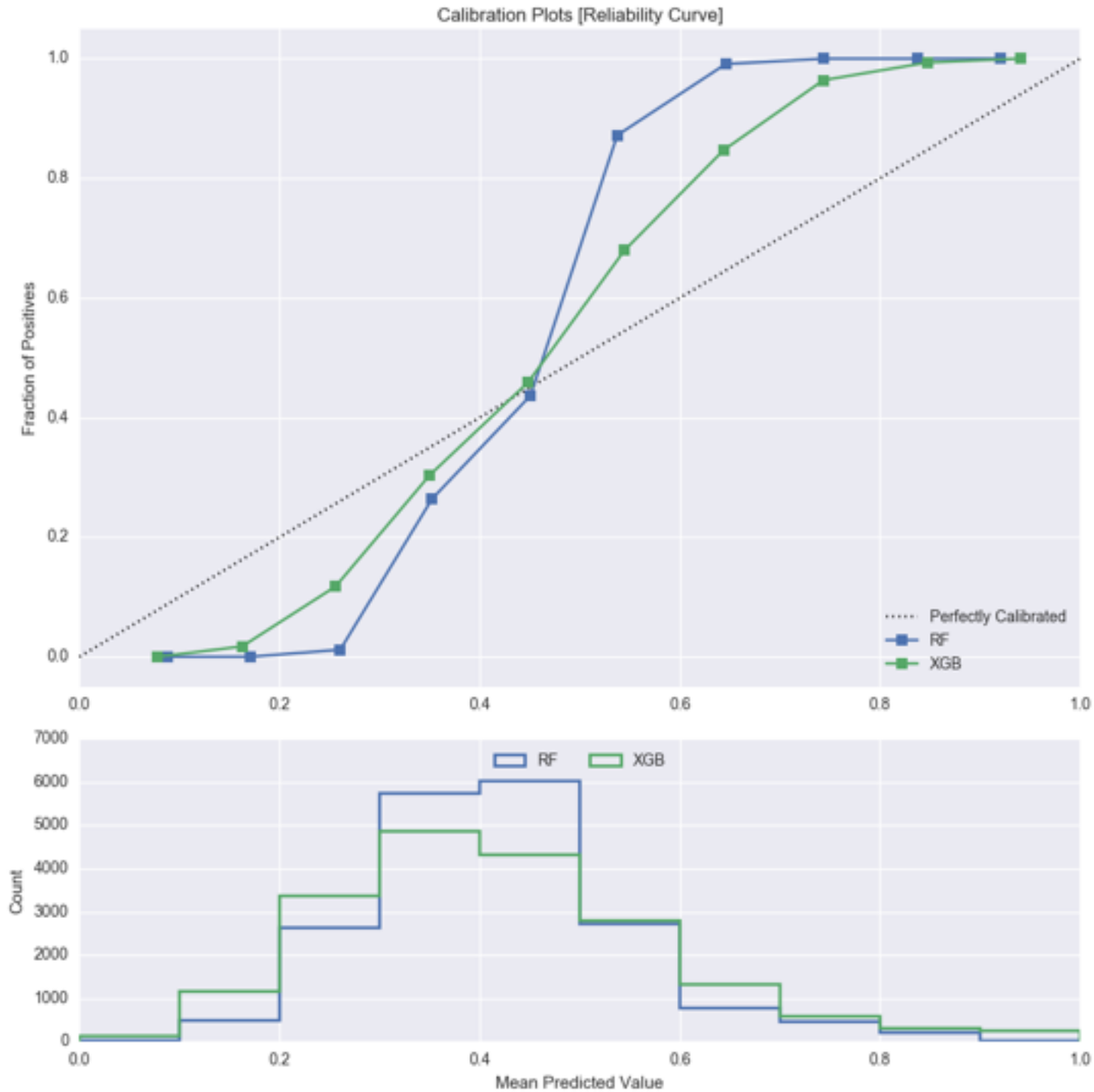
Cross-Validate

```
[09/12/16 21:48:01] INFO        Initial CV Scores for roc_auc: [ 0.65862458  0.66087472  0.659978    0.6515084   0.65447769]
[09/12/16 21:48:01] INFO        Final Model Predictions for RF
[09/12/16 21:48:01] INFO        Final Fitting
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    3.4s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   15.1s
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:   35.0s
[Parallel(n_jobs=-1)]: Done 501 out of 501 | elapsed:   39.6s finished
[09/12/16 21:48:41] INFO        Skipping Calibration
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.3s
[Parallel(n_jobs=4)]: Done 442 tasks      | elapsed:    0.7s
[Parallel(n_jobs=4)]: Done 501 out of 501 | elapsed:    0.8s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 442 tasks      | elapsed:    0.2s
[Parallel(n_jobs=4)]: Done 501 out of 501 | elapsed:    0.2s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.3s
[Parallel(n_jobs=4)]: Done 442 tasks      | elapsed:    0.6s
[Parallel(n_jobs=4)]: Done 501 out of 501 | elapsed:    0.7s finished
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    0.0s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 442 tasks      | elapsed:    0.2s
[Parallel(n_jobs=4)]: Done 501 out of 501 | elapsed:    0.2s finished
[09/12/16 21:48:44] INFO        Final Predictions Complete: 0:00:42.840783
[09/12/16 21:48:44] INFO        Algorithm: XGB
[09/12/16 21:48:44] INFO        Fitting Initial Model
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:  1.5min finished
[09/12/16 21:50:15] INFO        Initial CV Scores for roc_auc: [ 0.66949354  0.6756364   0.67205403  0.66401158  0.673068  ]
[09/12/16 21:50:15] INFO        Final Model Predictions for XGB
[09/12/16 21:50:15] INFO        Final Fitting
```

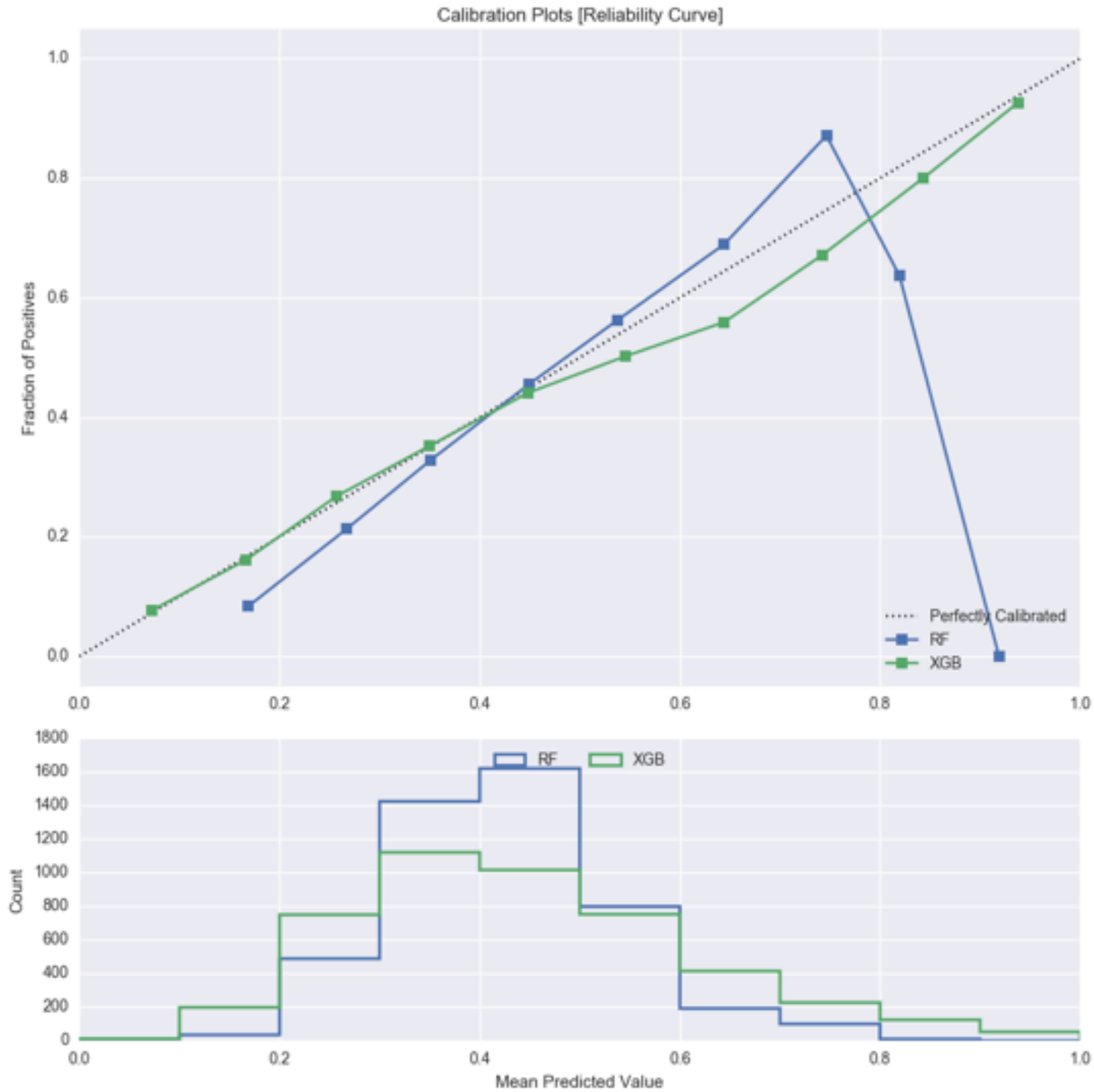
Generate Metrics

```
[09/12/16 21:51:08] INFO        ================================================================================
[09/12/16 21:51:08] INFO        Metrics for Partition: train
[09/12/16 21:51:08] INFO        --------------------------------------------------------------------------------
[09/12/16 21:51:08] INFO        Algorithm: RF
[09/12/16 21:51:08] INFO        accuracy: 0.762747356298
[09/12/16 21:51:08] INFO        adjusted_rand_score: 0.2701632733200826
[09/12/16 21:51:08] INFO        average_precision: 0.834716359198
[09/12/16 21:51:08] INFO        confusion_matrix: [[10733   356]
 [ 4176  3837]]
[09/12/16 21:51:08] INFO        explained_variance: 0.189950742524
[09/12/16 21:51:08] INFO        f1: 0.628707193184
[09/12/16 21:51:08] INFO        log_loss: 0.554747158381
[09/12/16 21:51:08] INFO        mean_absolute_error: 0.237252643702
[09/12/16 21:51:08] INFO        mean_squared_error: 0.237252643702
[09/12/16 21:51:08] INFO        median_absolute_error: 0.0
[09/12/16 21:51:08] INFO        precision: 0.915096589554
[09/12/16 21:51:08] INFO        r2: 0.0257257693465
[09/12/16 21:51:08] INFO        recall: 0.47884687383
[09/12/16 21:51:08] INFO        roc_auc: 0.853722916466
[09/12/16 21:51:08] INFO        --------------------------------------------------------------------------------
[09/12/16 21:51:08] INFO        Algorithm: XGB
[09/12/16 21:51:08] INFO        accuracy: 0.738613757722
[09/12/16 21:51:08] INFO        adjusted_rand_score: 0.223654885888894358
[09/12/16 21:51:08] INFO        average_precision: 0.783087046153
[09/12/16 21:51:08] INFO        confusion_matrix: [[9968 1121]
 [3872 4141]]
[09/12/16 21:51:08] INFO        explained_variance: 0.0117929025447
[09/12/16 21:51:08] INFO        f1: 0.623879472693
[09/12/16 21:51:08] INFO        log_loss: 0.5438330987
[09/12/16 21:51:08] INFO        mean_absolute_error: 0.261386242278
[09/12/16 21:51:08] INFO        mean_squared_error: 0.261386242278
[09/12/16 21:51:08] INFO        median_absolute_error: 0.0
[09/12/16 21:51:08] INFO        precision: 0.786963131889
[09/12/16 21:51:08] INFO        r2: -0.0733784716798
[09/12/16 21:51:08] INFO        recall: 0.516785224011
```

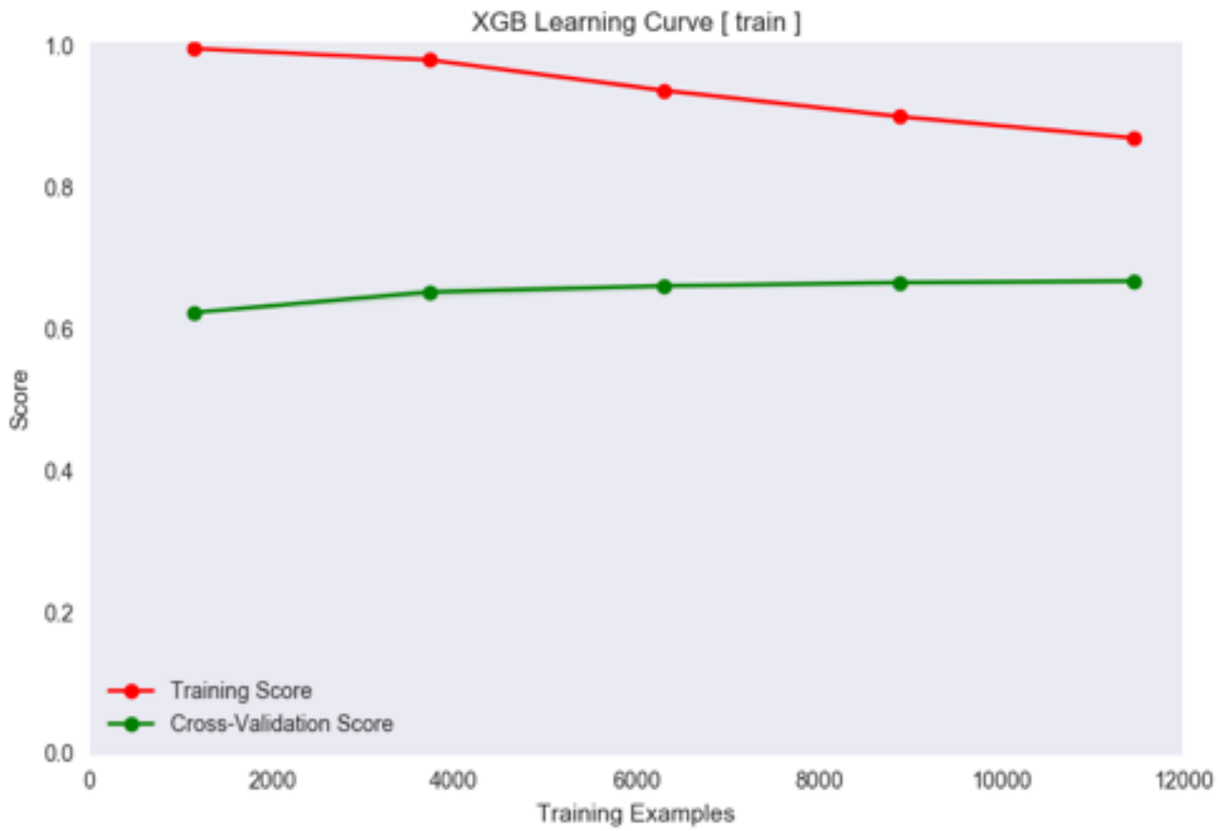
Calibration [Train]



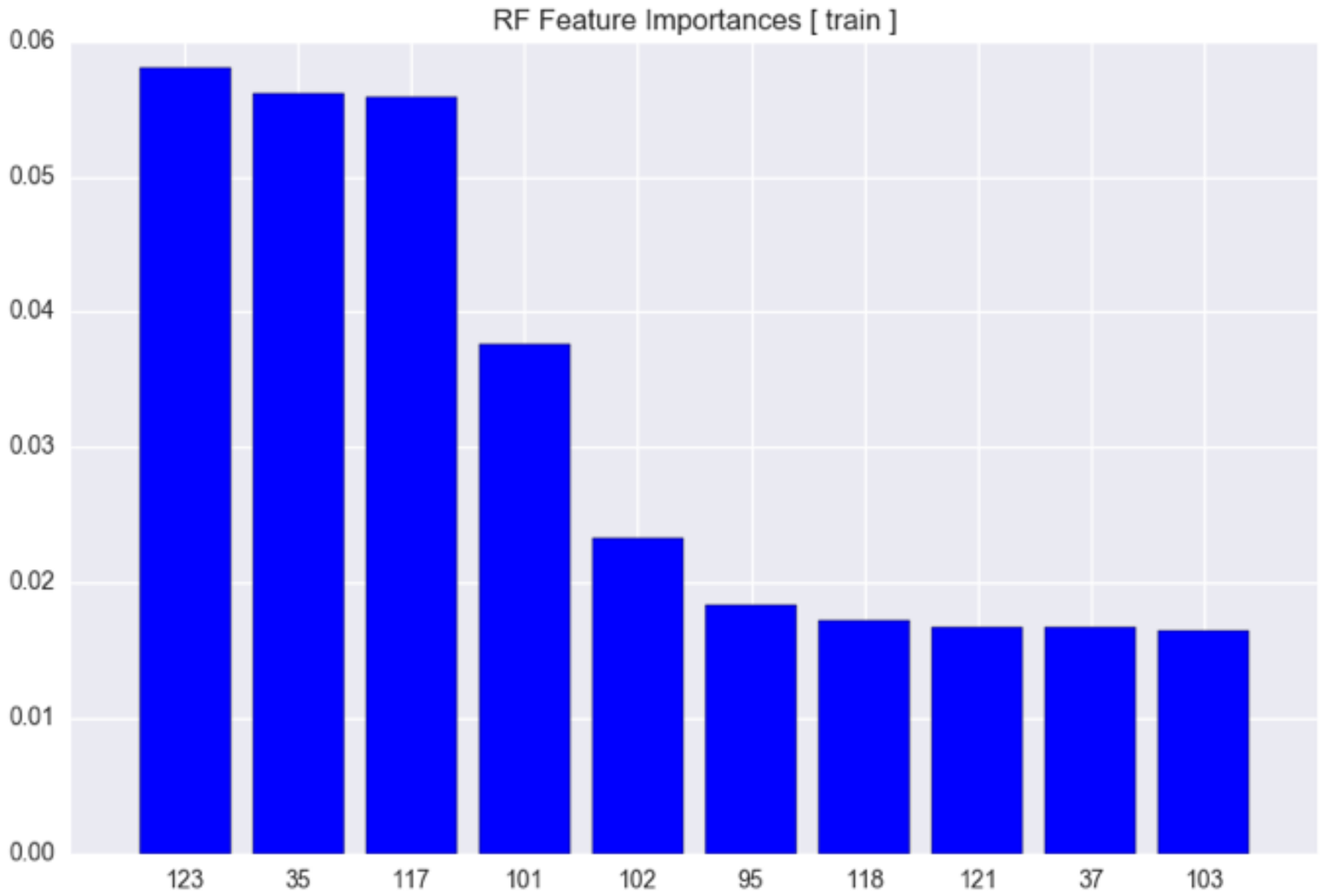
Calibration [Test]



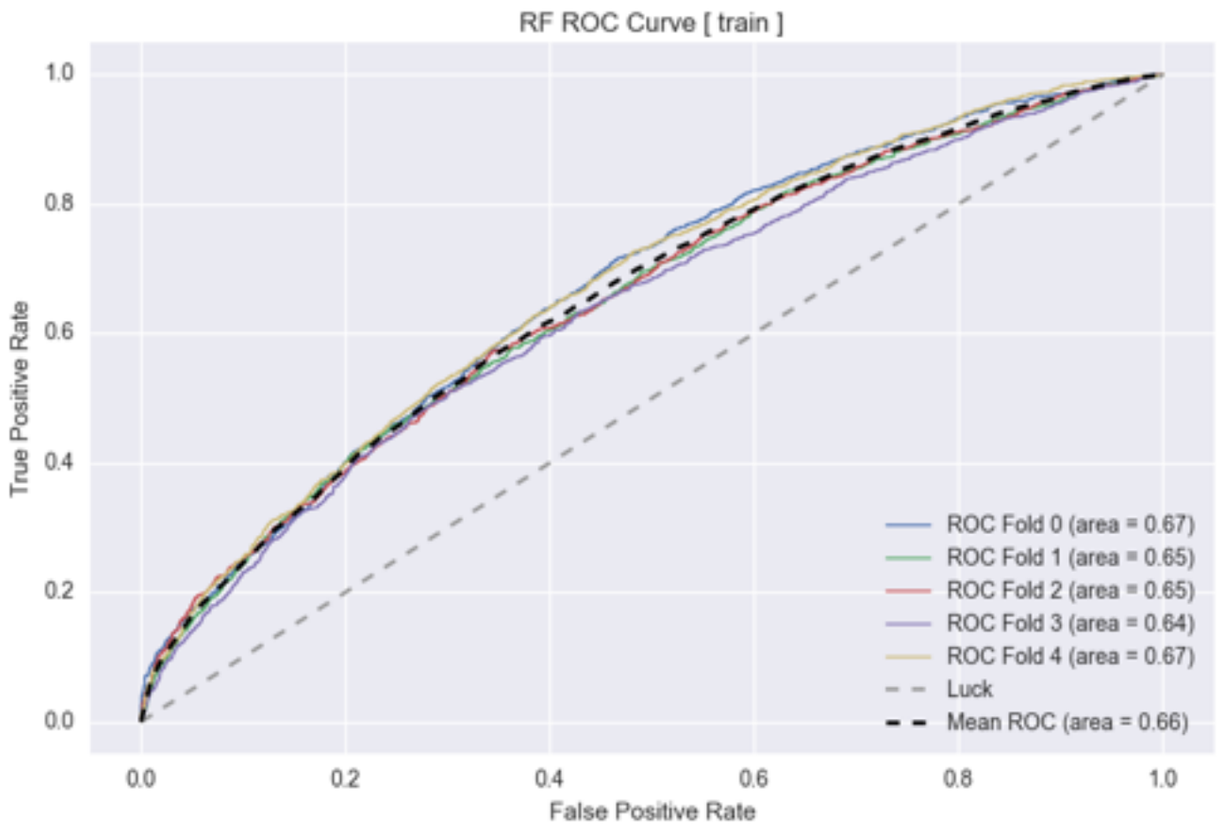
Learning Curves

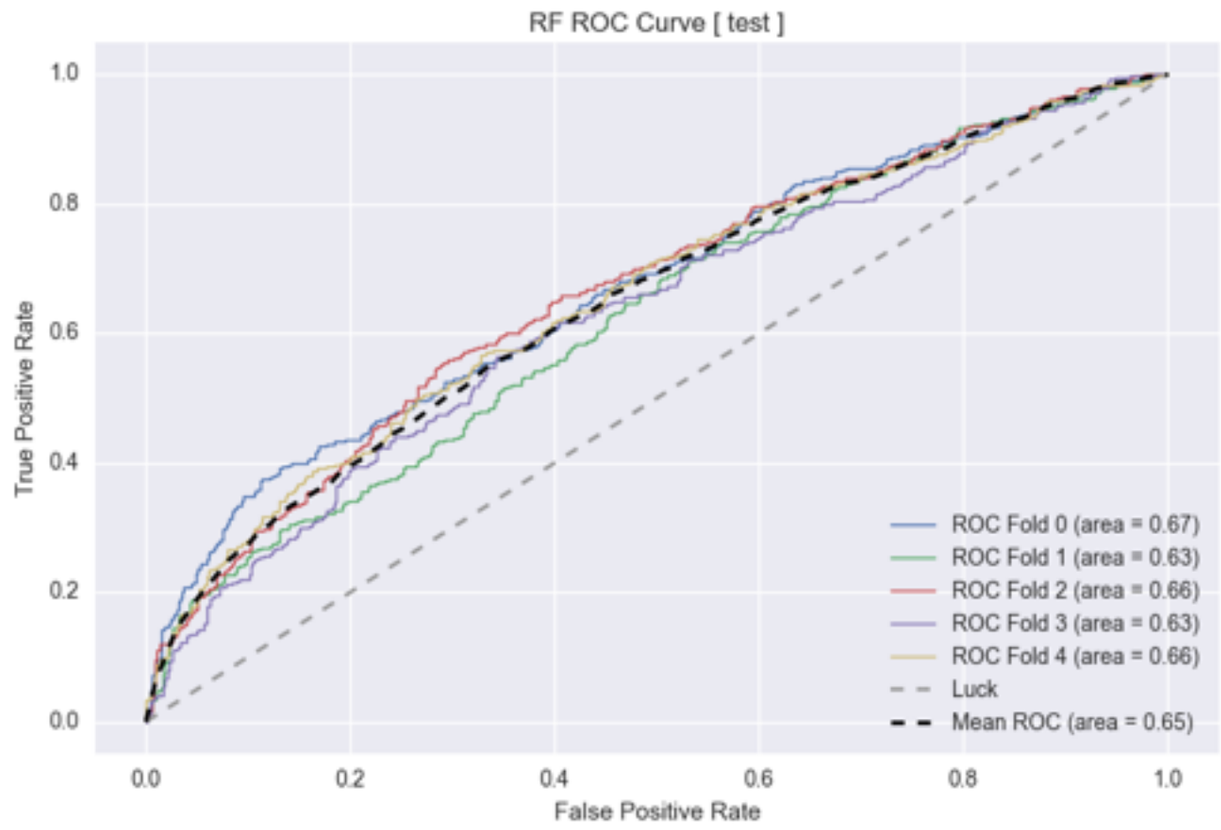


Feature Importances

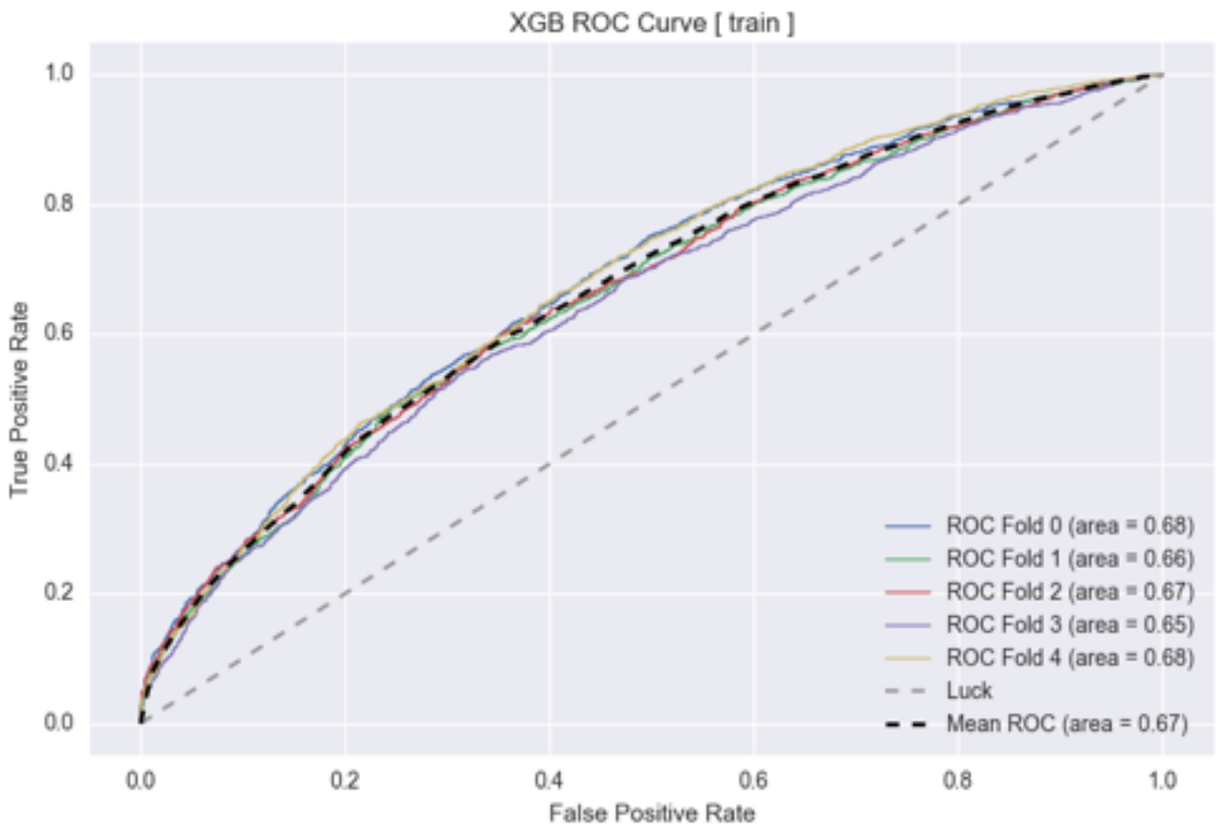


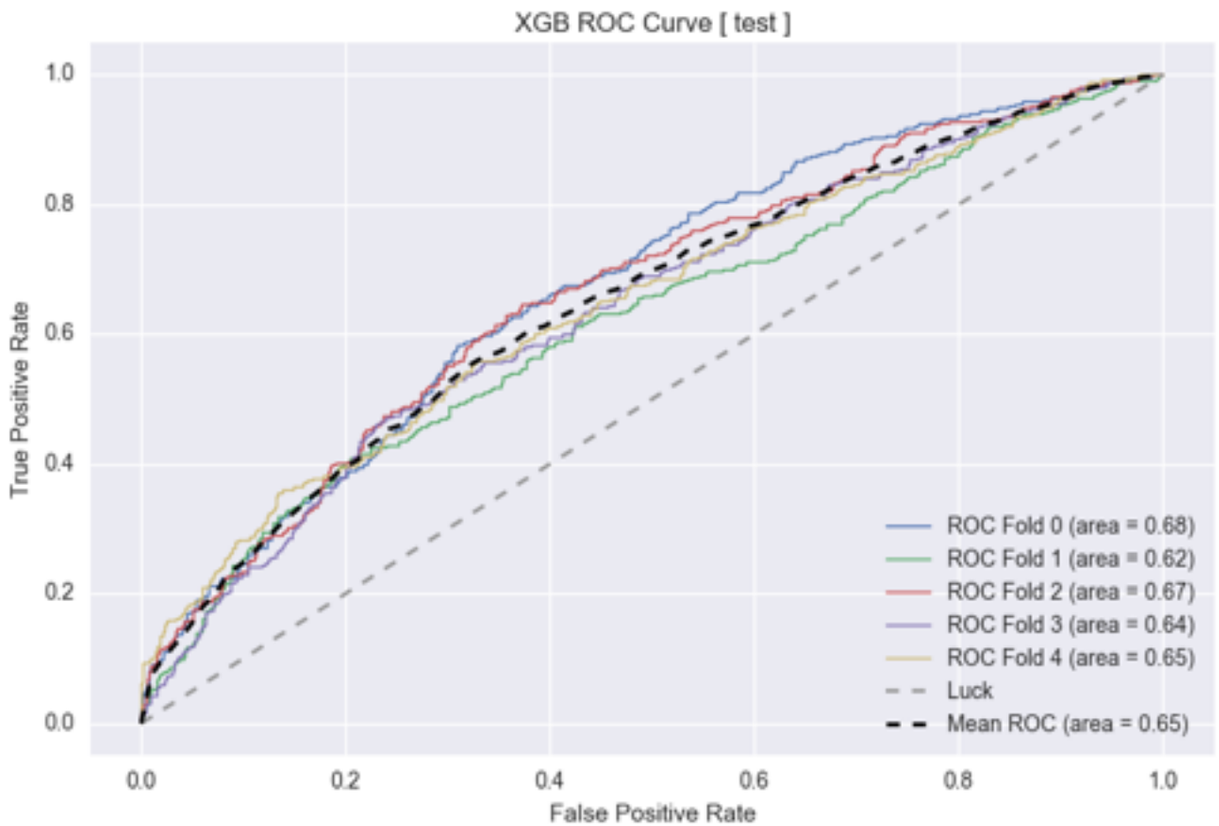
Random Forest ROC





XGBoost ROC


XGB ROC Curve [train]
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Fold 0 (area = 0.68)
ROC Fold 1 (area = 0.66)
ROC Fold 2 (area = 0.67)
ROC Fold 3 (area = 0.65)
ROC Fold 4 (area = 0.68)
Luck
Mean ROC (area = 0.67)


XGB ROC Curve [test]
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Fold 0 (area = 0.68)
ROC Fold 1 (area = 0.62)
ROC Fold 2 (area = 0.67)
ROC Fold 3 (area = 0.64)
ROC Fold 4 (area = 0.65)
Luck
Mean ROC (area = 0.65)

Results

- We have identified some features that predict large-range days. This is important for determining whether or not to deploy an automated system on any given day.
- Results are consistent across training and test data.
- The learning curves show that results may improve with more data.

Use Cases

- AlphaPy was created based on experimentation with many Kaggle competitions, so it can generate models for any classification or regression problem.

- AlphaPy has been designed to be a toolkit for all data scientists.

- The AlphaPy framework has specific pipelines for creating market and sports models.