Welcome to the course!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Before we get to XGBoost...

- Need to understand the basics of
  - Supervised classification
  - Decision trees
  - Boosting

Supervised learning

- Relies on labeled data
- Have some understanding of past behavior

Supervised learning example

- Does a specific image contain a person's face?

- Training data: vectors of pixel values
- Labels: 1 or 0

Supervised learning: Classification

- Outcome can be binary or multi-class

Binary classification example

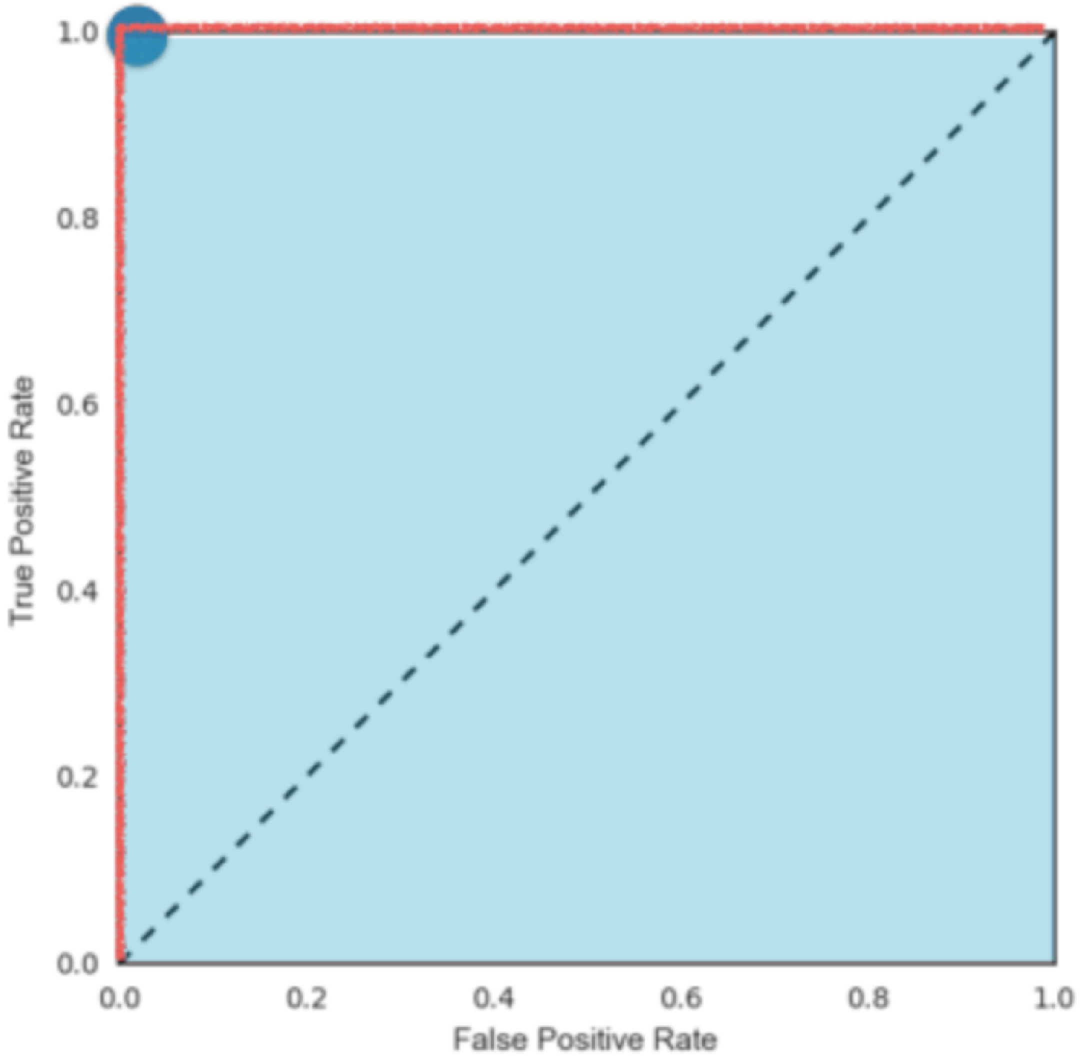
- Will a person purchase the insurance package given some quote?

Multi-class classification example

- Classifying the species of a given bird

AUC: Metric for binary classification models

- Area under the ROC curve (AUC)
- Larger area under the ROC curve = better model



Accuracy score and confusion matrix

- Confusion matrix

	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

- Accuracy

$$\frac{tp + tn}{tp + tn + fp + fn}$$

Review

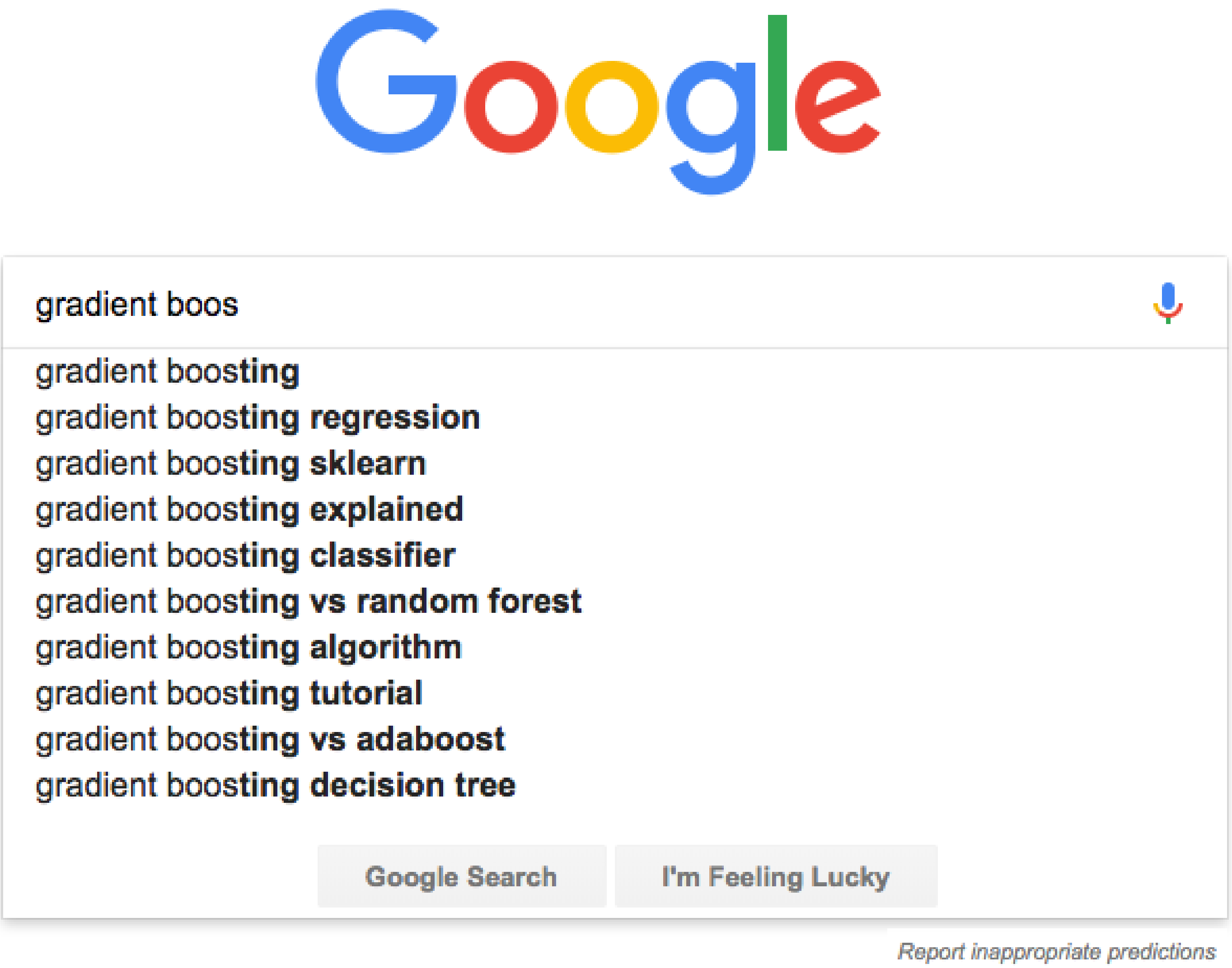
- Supervised Learning with scikit-learn

Other supervised learning considerations

- Features can be either numeric or categorical
- Numeric features should be scaled (Z-scored)
- Categorical features should be encoded (one-hot)

Ranking

- Predicting an ordering on a set of choices



Recommendation

- Recommending an item to a user
- Based on consumption history and profile
- Example: Netflix

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Introducing XGBoost

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

What is XGBoost?

- Optimized gradient-boosting machine learning library
- Originally written in C++
- Has APIs in several languages:
  - **Python**
  - R
  - Scala
  - Julia
  - Java

What makes XGBoost so popular?

- Speed and performance
- Core algorithm is parallelizable
- Consistently outperforms single-algorithm methods
- State-of-the-art performance in many ML tasks

Using XGBoost: a quick example

```python
import xgboost as xgb
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
class_data = pd.read_csv("classification_data.csv")

X, y = class_data.iloc[:,:-1], class_data.iloc[:,-1]
X_train, X_test, y_train, y_test= train_test_split(X, y,
        test_size=0.2, random_state=123)
xg_cl = xgb.XGBClassifier(objective='binary:logistic',
        n_estimators=10, seed=123)
xg_cl.fit(X_train, y_train)

preds = xg_cl.predict(X_test)
accuracy = float(np.sum(preds==y_test))/y_test.shape[0]

print("accuracy: %f" % (accuracy))
```

```
accuracy: 0.78333
```

Let's begin using XGBoost!

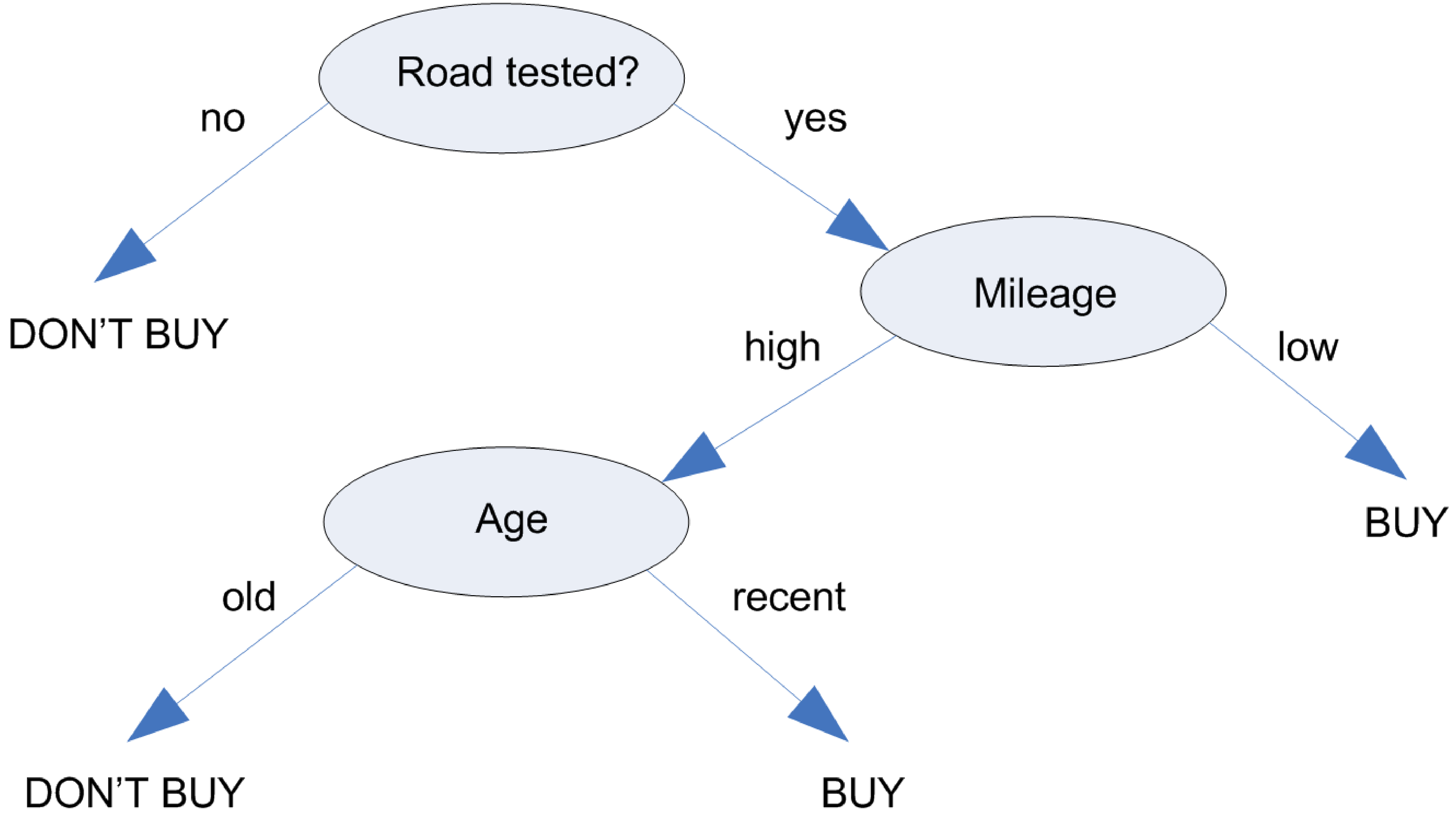
EXTREME GRADIENT BOOSTING WITH XGBOOST

What is a decision tree?

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Visualizing a decision tree



[1] https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/nodes_treebuilding.htm

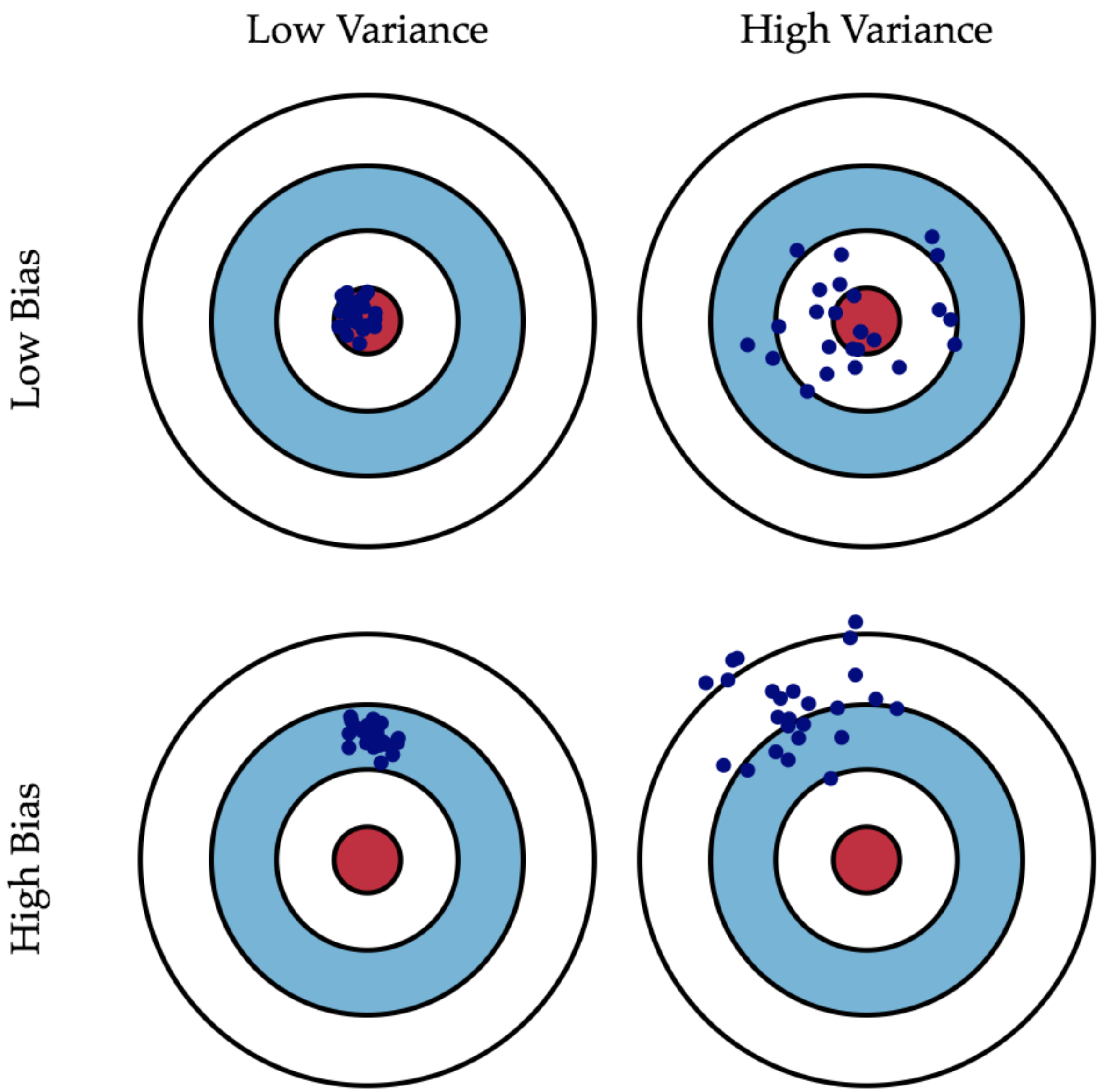
Decision trees as base learners

- Base learner - Individual learning algorithm in an ensemble algorithm
- Composed of a series of binary questions
- Predictions happen at the "leaves" of the tree

Decision trees and CART

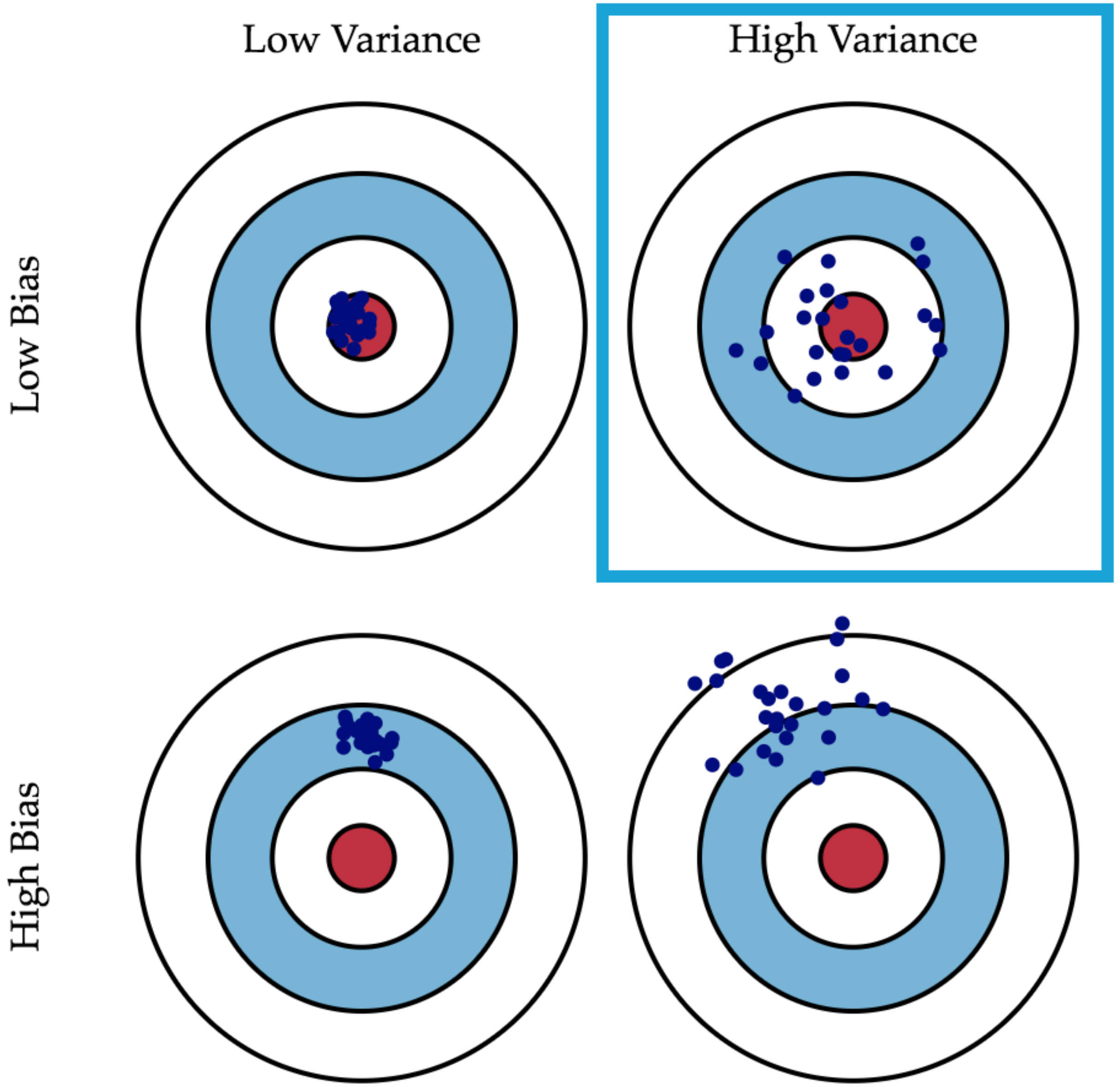
- Constructed iteratively (one decision at a time)
  - Until a stopping criterion is met

Individual decision trees tend to overfit



[1] http://scott.fortmann-roe.com/docs/BiasVariance.html

Individual decision trees tend to overfit



[1] http://scott.fortmann-roe.com/docs/BiasVariance.html

CART: Classification and Regression Trees

- Each leaf **always** contains a real-valued score
- Can later be converted into categories

Let's work with some decision trees!

EXTREME GRADIENT BOOSTING WITH XGBOOST

What is Boosting?

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Boosting overview

- Not a specific machine learning algorithm
- Concept that can be applied to a set of machine learning models
  - "Meta-algorithm"
- Ensemble meta-algorithm used to convert many weak learners into a strong learner

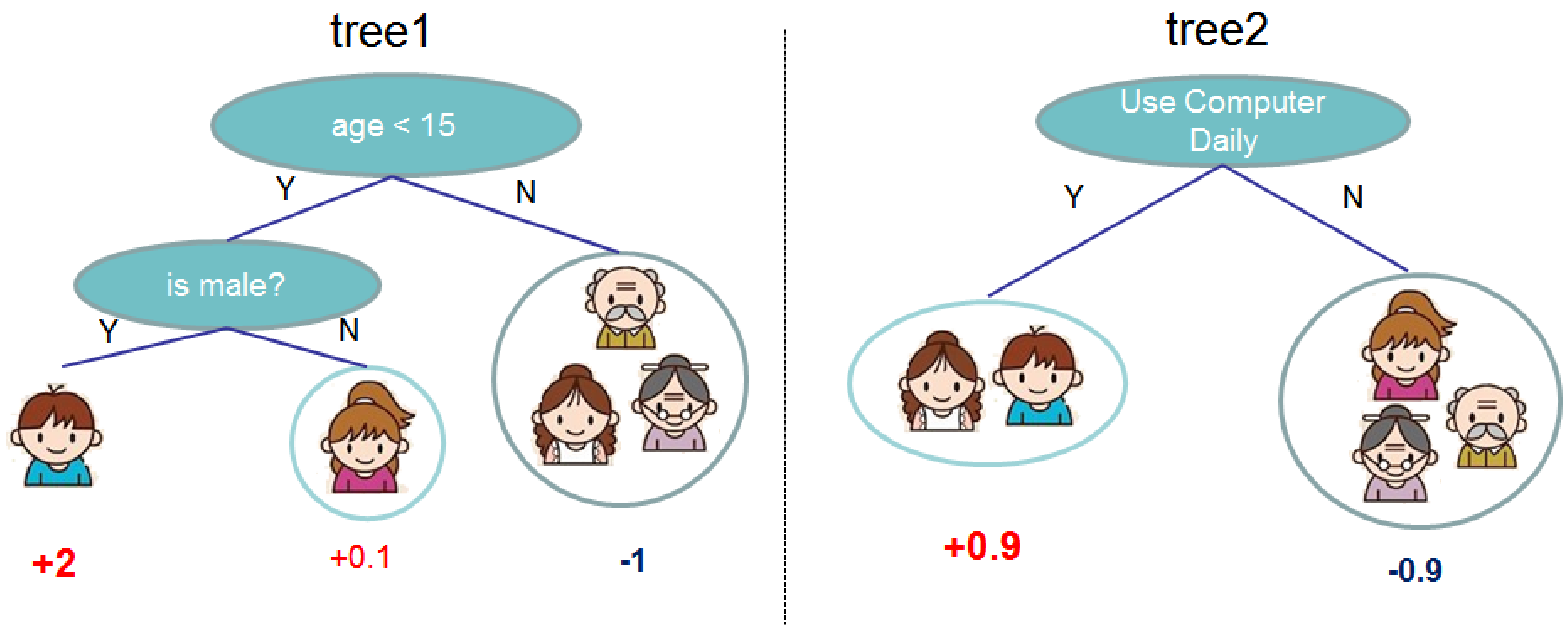
Weak learners and strong learners

- Weak learner: ML algorithm that is slightly better than chance
  - Example: Decision tree whose predictions are slightly better than 50%
- Boosting converts a collection of weak learners into a strong learner
- Strong learner: Any algorithm that can be tuned to achieve good performance

How boosting is accomplished

- Iteratively learning a set of weak models on subsets of the data
- Weighing each weak prediction according to each weak learner's performance
- Combine the weighted predictions to obtain a single weighted prediction
- ... that is much better than the individual predictions themselves!

Boosting example



f() = 2 + 0.9= 2.9 f()= -1 - 0.9 = -1.9

[1] https://xgboost.readthedocs.io/en/latest/model.html

Model evaluation through cross-validation

- Cross-validation: Robust method for estimating the performance of a model on unseen data
- Generates many non-overlapping train/test splits on training data
- Reports the average test set performance across all data splits

Cross-validation in XGBoost example

```
import xgboost as xgb
import pandas as pd
churn_data = pd.read_csv("classification_data.csv")
churn_dmatrix = xgb.DMatrix(data=churn_data.iloc[:,:-1],
        label=churn_data.month_5_still_here)
params={"objective":"binary:logistic","max_depth":4}
cv_results = xgb.cv(dtrain=churn_dmatrix, params=params, nfold=4,
        num_boost_round=10, metrics="error", as_pandas=True)
print("Accuracy: %f" %((1-cv_results["test-error-mean"]).iloc[-1]))
```

```
Accuracy: 0.88315
```

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

When should I use XGBoost?

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

When to use XGBoost

- You have a large number of training samples
  - Greater than 1000 training samples and less 100 features
  - The number of features < number of training samples
- You have a mixture of categorical and numeric features
  - Or just numeric features

When to NOT use XGBoost

- Image recognition
- Computer vision
- Natural language processing and understanding problems
- When the number of training samples is significantly smaller than the number of features

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

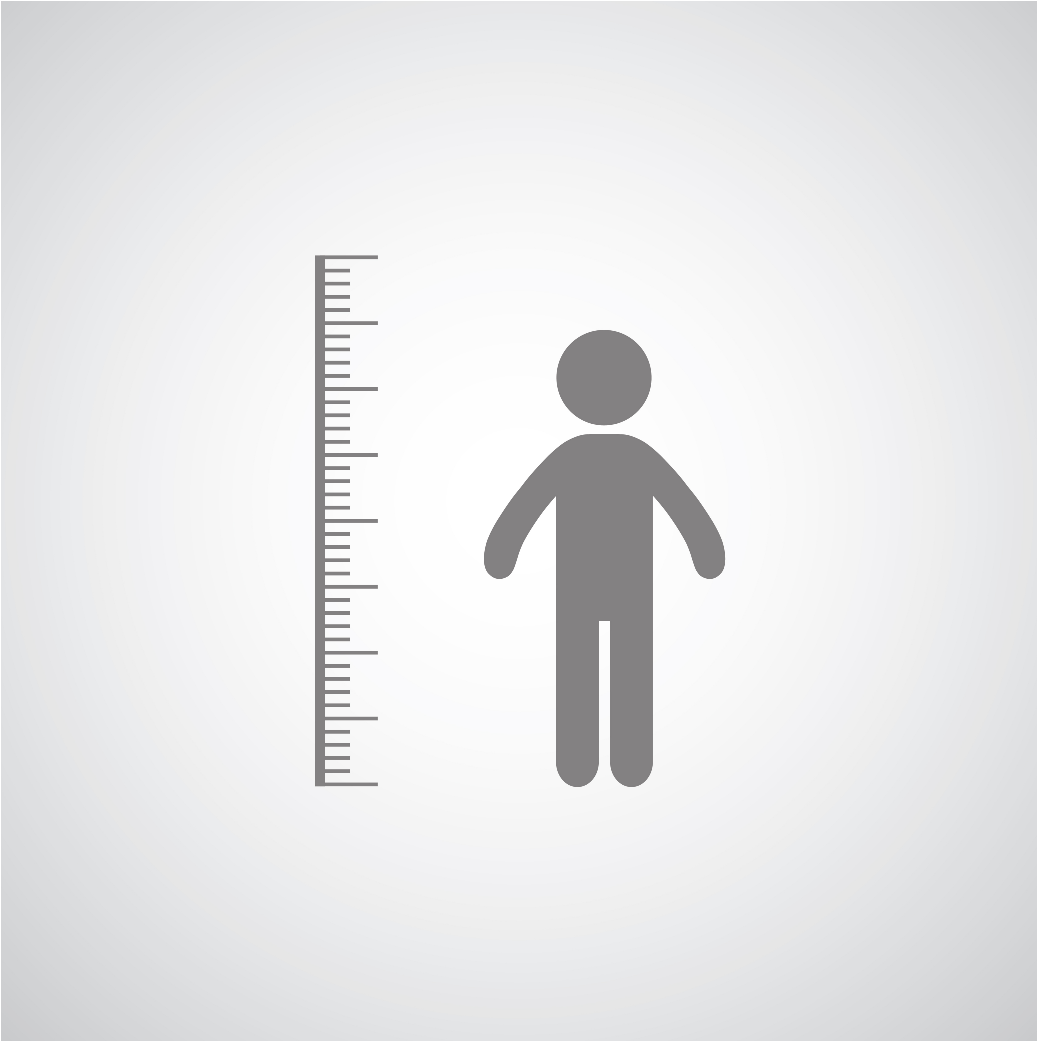
Regression review

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Regression basics

- Outcome is real-valued



Common regression metrics

- Root mean squared error (RMSE)
- Mean absolute error (MAE)

Computing RMSE

Actual	Predicted
10	20
3	8
6	1

Computing RMSE

Actual	Predicted	Error
10	20	-10
3	8	-5
6	1	5

Computing RMSE

Actual	Predicted	Error	Squared Error
10	20	-10	100
3	8	-5	25
6	1	5	25

- Total Squared Error: 150
- Mean Squared Error: 50
- Root Mean Squared Error: 7.07

Computing MAE

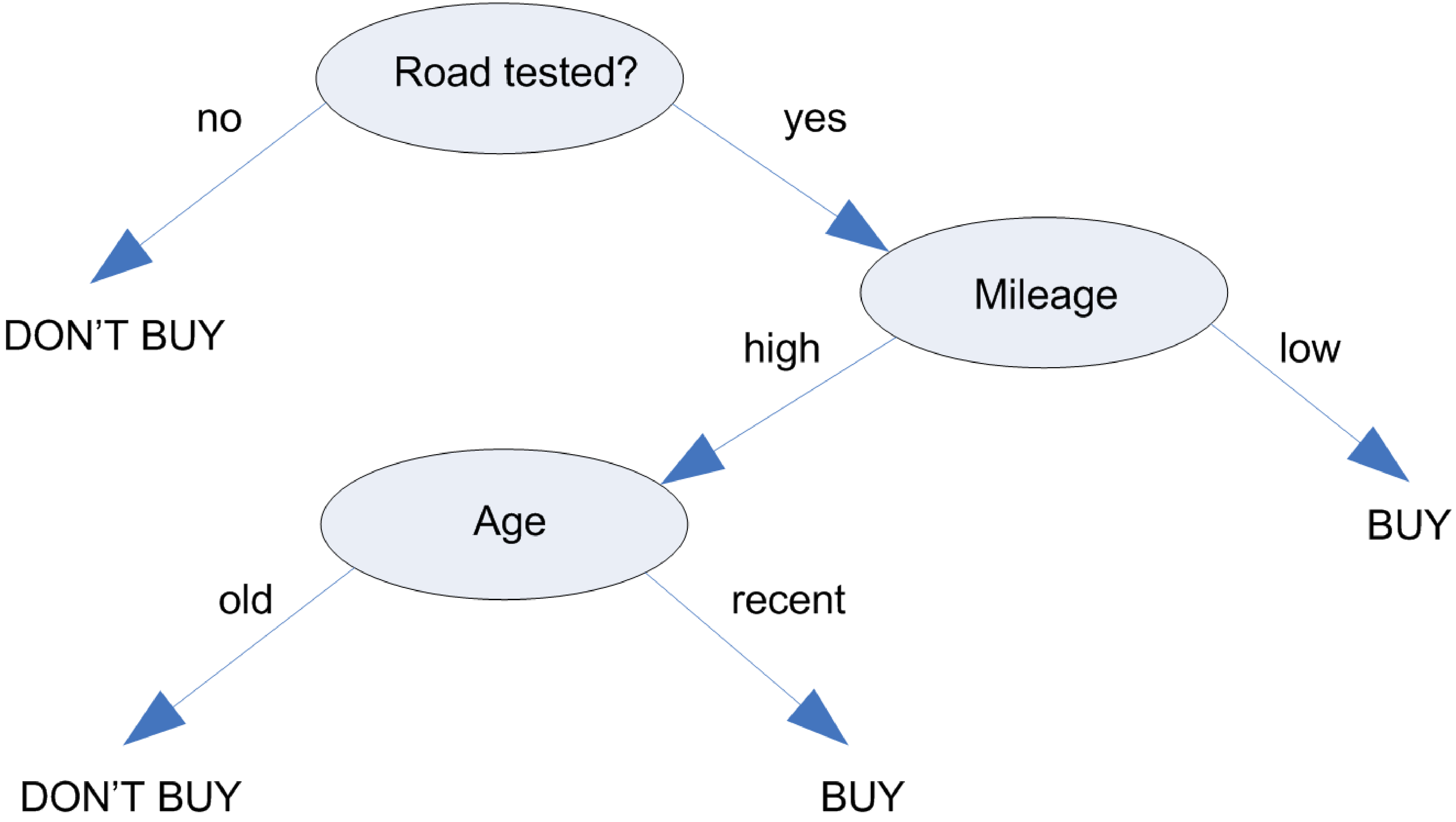
Actual	Predicted	Error
10	20	-10
3	8	-5
6	1	5

- Total Absolute Error: 20
- Mean Absolute Error: 6.67

Common regression algorithms

- Linear regression
- Decision trees

Algorithms for both regression and classification



[1] https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/nodes_treebuilding.htm

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Objective (loss) functions and base learners

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Objective Functions and Why We Use Them

- Quantifies how far off a prediction is from the actual result
- Measures the difference between estimated and true values for some collection of data
- Goal: Find the model that yields the minimum value of the loss function

Common loss functions and XGBoost

- Loss function names in xgboost:
  - reg:linear - use for regression problems
  - reg:logistic - use for classification problems when you want just decision, not probability
  - binary:logistic - use when you want probability rather than just decision

Base learners and why we need them

- XGBoost involves creating a meta-model that is composed of many individual models that combine to give a final prediction
- Individual models = base learners
- Want base learners that when combined create final prediction that is **non-linear**
- Each base learner should be good at distinguishing or predicting different parts of the dataset
- Two kinds of base learners: tree and linear

Trees as base learners example: Scikit-learn API

```
import xgboost as xgb
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

boston_data = pd.read_csv("boston_housing.csv")
X, y = boston_data.iloc[:,:-1],boston_data.iloc[:,-1]

X_train, X_test, y_train, y_test= train_test_split(X, y, test_size=0.2,
                                                        random_state=123)
xg_reg = xgb.XGBRegressor(objective='reg:linear', n_estimators=10,
                                                  seed=123)
xg_reg.fit(X_train, y_train)

preds = xg_reg.predict(X_test)
```

Trees as base learners example: Scikit-learn API

```
rmse = np.sqrt(mean_squared_error(y_test,preds))

print("RMSE: %f" % (rmse))
```

```
RMSE: 129043.2314
```

Linear base learners example: learning API only

```
import xgboost as xgb
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

boston_data = pd.read_csv("boston_housing.csv")

X, y = boston_data.iloc[:,:-1],boston_data.iloc[:,-1]

X_train, X_test, y_train, y_test= train_test_split(X, y, test_size=0.2,
                                                        random_state=123)
DM_train = xgb.DMatrix(data=X_train,label=y_train)
DM_test =  xgb.DMatrix(data=X_test,label=y_test)
params = {"booster":"gblinear","objective":"reg:linear"}
xg_reg = xgb.train(params = params, dtrain=DM_train, num_boost_round=10)

preds = xg_reg.predict(DM_test)
```

Linear base learners example: learning API only

```
rmse = np.sqrt(mean_squared_error(y_test,preds))

print("RMSE: %f" % (rmse))
```

```
RMSE: 124326.24465
```

Let's get to work!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Regularization and base learners in XGBoost

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Regularization in XGBoost

- Regularization is a control on model complexity
- Want models that are both accurate and as simple as possible
- Regularization parameters in XGBoost:
  - gamma - minimum loss reduction allowed for a split to occur
  - alpha - l1 regularization on leaf weights, larger values mean more regularization
  - lambda - l2 regularization on leaf weights

L1 regularization in XGBoost example

```python
import xgboost as xgb
import pandas as pd
boston_data = pd.read_csv("boston_data.csv")
X,y = boston_data.iloc[:,:-1],boston_data.iloc[:,-1]
boston_dmatrix = xgb.DMatrix(data=X,label=y)
params={"objective":"reg:linear","max_depth":4}
l1_params = [1,10,100]
rmses_l1=[]
for reg in l1_params:
    params["alpha"] = reg
    cv_results = xgb.cv(dtrain=boston_dmatrix, params=params,nfold=4,
                        num_boost_round=10,metrics="rmse",as_pandas=True,seed=123)
    rmses_l1.append(cv_results["test-rmse-mean"].tail(1).values[0])
print("Best rmse as a function of l1:")
print(pd.DataFrame(list(zip(l1_params,rmses_l1)), columns=["l1","rmse"]))
```

```
Best rmse as a function of l1:
        l1          rmse
    0    1  69572.517742
    1   10  73721.967141
    2  100  82312.312413
```

Base learners in XGBoost

- Linear Base Learner:
  - Sum of linear terms
  - Boosted model is weighted sum of linear models (thus is itself linear)
  - Rarely used
- Tree Base Learner:
  - Decision tree
  - Boosted model is weighted sum of decision trees (nonlinear)
  - Almost exclusively used in XGBoost

Creating DataFrames from multiple equal-length lists

- `pd.DataFrame(list(zip(list1,list2)),columns=["list1","list2"]))`
- `zip` creates a `generator` of parallel values:
  - `zip([1,2,3],["a","b""c"])` = `[1,"a"],[2,"b"],[3,"c"]`
  - `generators` need to be completely instantiated before they can be used in `DataFrame` objects
- `list()` instantiates the full generator and passing that into the `DataFrame` converts the whole expression

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Why tune your model?

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Untuned model example

```
import pandas as pd
import xgboost as xgb
import numpy as np
housing_data = pd.read_csv("ames_housing_trimmed_processed.csv")
X,y = housing_data[housing_data.columns.tolist()[:-1]],
        housing_data[housing_data.columns.tolist()[-1]]
housing_dmatrix = xgb.DMatrix(data=X,label=y)
untuned_params={"objective":"reg:linear"}
untuned_cv_results_rmse = xgb.cv(dtrain=housing_dmatrix,
        params=untuned_params,nfold=4,
        metrics="rmse",as_pandas=True,seed=123)
print("Untuned rmse: %f" %((untuned_cv_results_rmse["test-rmse-mean"]).tail(1)))
```

```
Untuned rmse: 34624.229980
```

Tuned model example

```
import pandas as pd
import xgboost as xgb
import numpy as np
housing_data = pd.read_csv("ames_housing_trimmed_processed.csv")
X,y = housing_data[housing_data.columns.tolist()[:-1]],
     housing_data[housing_data.columns.tolist()[-1]]
housing_dmatrix = xgb.DMatrix(data=X,label=y)
tuned_params = {"objective":"reg:linear",'colsample_bytree': 0.3,
    'learning_rate': 0.1, 'max_depth': 5}
tuned_cv_results_rmse = xgb.cv(dtrain=housing_dmatrix,
     params=tuned_params, nfold=4, num_boost_round=200, metrics="rmse",
     as_pandas=True, seed=123)
print("Tuned rmse: %f" %((tuned_cv_results_rmse["test-rmse-mean"]).tail(1)))
```

```
Tuned rmse: 29812.683594
```

Let's tune some models!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Tunable parameters in XGBoost

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Common tree tunable parameters

- **learning rate:** learning rate/eta
- **gamma:** min loss reduction to create new tree split
- **lambda:** L2 reg on leaf weights
- **alpha:** L1 reg on leaf weights
- **max_depth:** max depth per tree
- **subsample:** % samples used per tree
- **colsample_bytree:** % features used per tree

Linear tunable parameters

- **lambda:** L2 reg on weights
- **alpha:** L1 reg on weights
- **lambda_bias:** L2 reg term on bias
- You can also tune the number of estimators used for both base model types!

Let's get to some tuning!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Review of grid search and random search

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Grid search: review

- Search exhaustively over a given set of hyperparameters, once per set of hyperparameters
- Number of models = number of distinct values per hyperparameter multiplied across each hyperparameter
- Pick final model hyperparameter values that give best cross-validated evaluation metric value

Grid search: example

```
import pandas as pd
import xgboost as xgb
import numpy as np
from sklearn.model_selection import GridSearchCV
housing_data = pd.read_csv("ames_housing_trimmed_processed.csv")
X, y = housing_data[housing_data.columns.tolist()[:-1]],
       housing_data[housing_data.columns.tolist()[-1]
housing_dmatrix = xgb.DMatrix(data=X,label=y)
gbm_param_grid = {'learning_rate': [0.01,0.1,0.5,0.9],
                  'n_estimators': [200],
                  'subsample': [0.3, 0.5, 0.9]}
gbm = xgb.XGBRegressor()
grid_mse = GridSearchCV(estimator=gbm,param_grid=gbm_param_grid,
            scoring='neg_mean_squared_error', cv=4, verbose=1)
grid_mse.fit(X, y)
print("Best parameters found: ",grid_mse.best_params_)
print("Lowest RMSE found: ", np.sqrt(np.abs(grid_mse.best_score_)))
```

```
Best parameters found: {'learning_rate': 0.1,
'n_estimators': 200, 'subsample': 0.5}
Lowest RMSE found:  28530.1829341
```

Random search: review

- Create a (possibly infinite) range of hyperparameter values per hyperparameter that you would like to search over
- Set the number of iterations you would like for the random search to continue
- During each iteration, randomly draw a value in the range of specified values for each hyperparameter searched over and train/evaluate a model with those hyperparameters
- After you've reached the maximum number of iterations, select the hyperparameter configuration with the best evaluated score

Random search: example

```python
import pandas as pd
import xgboost as xgb
import numpy as np
from sklearn.model_selection import RandomizedSearchCV
housing_data = pd.read_csv("ames_housing_trimmed_processed.csv")
X,y = housing_data[housing_data.columns.tolist()[:-1]],
      housing_data[housing_data.columns.tolist()[-1]]
housing_dmatrix = xgb.DMatrix(data=X,label=y)
gbm_param_grid = {'learning_rate': np.arange(0.05,1.05,.05),
                  'n_estimators': [200],
                  'subsample': np.arange(0.05,1.05,.05)}
gbm = xgb.XGBRegressor()
randomized_mse = RandomizedSearchCV(estimator=gbm, param_distributions=gbm_param_grid,
                        n_iter=25, scoring='neg_mean_squared_error', cv=4, verbose=1)
randomized_mse.fit(X, y)
print("Best parameters found: ",randomized_mse.best_params_)
print("Lowest RMSE found: ", np.sqrt(np.abs(randomized_mse.best_score_)))
```

```
Best parameters found: {'subsample': 0.6000000000000009,
'n_estimators': 200, 'learning_rate': 0.20000000000000001}
Lowest RMSE found: 28300.2374291
```

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Limits of grid search and random search

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Grid search and random search limitations

- Grid Search
  - Number of models you must build with every additional new parameter grows very quickly
- Random Search
  - Parameter space to explore can be massive
  - Randomly jumping throughout the space looking for a "best" result becomes a waiting game

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Review of pipelines using sklearn

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Pipeline review

- Takes a list of named 2-tuples (name, pipeline_step) as input
- Tuples can contain any arbitrary scikit-learn compatible estimator or transformer object
- Pipeline implements fit/predict methods
- Can be used as input estimator into grid/randomized search and cross_val_score methods

Scikit-learn pipeline example

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
names = ["crime","zone","industry","charles","no","rooms",
        "age", "distance","radial","tax","pupil","aam","lower","med_price"]

data = pd.read_csv("boston_housing.csv",names=names)

X, y = data.iloc[:,:-1], data.iloc[:,-1]
rf_pipeline = Pipeline[("st_scaler",
                StandardScaler()),
                ("rf_model",RandomForestRegressor())]

scores = cross_val_score(rf_pipeline,X,y,
scoring="neg_mean_squared_error",cv=10)
```

Scikit-learn pipeline example

```
final_avg_rmse = np.mean(np.sqrt(np.abs(scores)))

print("Final RMSE:", final_avg_rmse)
```

```
Final RMSE: 4.54530686529
```

Preprocessing I: LabelEncoder and OneHotEncoder

- `LabelEncoder` : Converts a categorical column of strings into integers
- `OneHotEncoder` : Takes the column of integers and encodes them as dummy variables
- Cannot be done within a pipeline

Preprocessing II: DictVectorizer

- Traditionally used in text processing
- Converts lists of feature mappings into vectors
- Need to convert DataFrame into a list of dictionary entries
- Explore the **scikit-learn documentation**

Let's build pipelines!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Incorporating xgboost into pipelines

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Scikit-learn pipeline example with XGBoost

```
import pandas as pd
import xgboost as xgb
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
names = ["crime","zone","industry","charles","no","rooms","age",
        "distance","radial","tax","pupil","aam","lower","med_price"]
data = pd.read_csv("boston_housing.csv",names=names)
X, y = data.iloc[:,:-1], data.iloc[:,-1]
xgb_pipeline = Pipeline[("st_scaler", StandardScaler()),
                        ("xgb_model",xgb.XGBRegressor())]
scores = cross_val_score(xgb_pipeline, X, y,
                         scoring="neg_mean_squared_error",cv=10)
final_avg_rmse = np.mean(np.sqrt(np.abs(scores)))
print("Final XGB RMSE:", final_avg_rmse)
```

```
Final RMSE: 4.02719593323
```

Additional components introduced for pipelines

- `sklearn_pandas` :
  - `DataFrameMapper` - Interoperability between `pandas` and `scikit-learn`
  - `CategoricalImputer` - Allow for imputation of categorical variables before conversion to integers
- `sklearn.preprocessing` :
  - `Imputer` - Native imputation of numerical columns in scikit-learn
- `sklearn.pipeline` :
  - `FeatureUnion` - combine multiple pipelines of features into a single pipeline of features

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Tuning xgboost hyperparameters in a pipeline

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Tuning XGBoost hyperparameters in a pipeline

```
import pandas as pd
   ...: import xgboost as xgb
   ...: import numpy as np
   ...: from sklearn.preprocessing import StandardScaler
   ...: from sklearn.pipeline import Pipeline
   ...: from sklearn.model_selection import RandomizedSearchCV
names = ["crime","zone","industry","charles","no",
   ...: "rooms","age", "distance","radial","tax",
   ...: "pupil","aam","lower","med_price"]
data = pd.read_csv("boston_housing.csv",names=names)
X, y = data.iloc[:,:-1],data.iloc[:,-1]
xgb_pipeline = Pipeline[("st_scaler",
   ...: StandardScaler()), ("xgb_model",xgb.XGBRegressor())]
gbm_param_grid = {
   ...:     'xgb_model__subsample': np.arange(.05, 1, .05),
   ...:     'xgb_model__max_depth': np.arange(3,20,1),
   ...:     'xgb_model__colsample_bytree': np.arange(.1,1.05,.05) }
randomized_neg_mse = RandomizedSearchCV(estimator=xgb_pipeline,
   ...: param_distributions=gbm_param_grid, n_iter=10,
   ...: scoring='neg_mean_squared_error', cv=4)
randomized_neg_mse.fit(X, y)
```

Tuning XGBoost hyperparameters in a pipeline II

```
print("Best rmse: ", np.sqrt(np.abs(randomized_neg_mse.best_score_)))
```

```
Best rmse: 3.9966784203040677
```

```
print("Best model: ", randomized_neg_mse.best_estimator_)
```

```
Best model:  Pipeline(steps=[('st_scaler', StandardScaler(copy=True,
with_mean=True, with_std=True)),
('xgb_model', XGBRegressor(base_score=0.5, colsample_bylevel=1,
       colsample_bytree=0.95000000000000029, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=8, min_child_weight=1, missing=None,
       n_estimators=100, nthread=-1, objective='reg:linear', reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=0, silent=True,
       subsample=0.90000000000000013))])
```

Let's finish this up!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Final Thoughts

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

What We Have Covered And You Have Learned

- Using XGBoost for classification tasks
- Using XGBoost for regression tasks
- Tuning XGBoost's most important hyperparameters
- Incorporating XGBoost into sklearn pipelines

What We Have Not Covered (And How You Can Proceed)

- Using XGBoost for ranking/recommendation problems (Netflix/Amazon problem)
- Using more sophisticated hyperparameter tuning strategies for tuning XGBoost models (Bayesian Optimization)
- Using XGBoost as part of an ensemble of other models for regression/classification

Congratulations!

EXTREME GRADIENT BOOSTING WITH XGBOOST