Model Cost Benefit Analysis Framework – Level 2

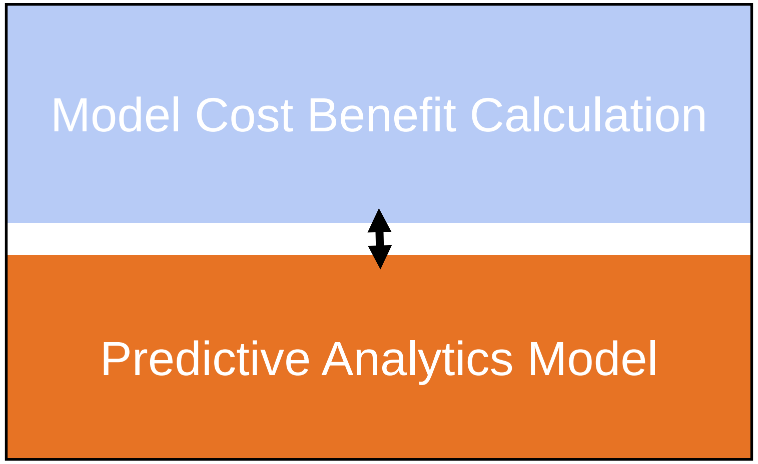
Model Cost-benefit Analysis Framework

Cluster Analysis, Social Network Analysis, etc.	*Regression Analysis*	*Classification Model*	**Example Model**
Descriptive Analytics Models	Predictive Analytics Models		**Type of Analytics**
Model Cost Benefit Calculation			**Level 1 Analysis**
Value of Insights Derived for Decision-making	Cost Regression	Cost Matrix Classification	**Level 2 Analysis**
	Instance Level Regression	Instance Level Classification	**Level 3 Analysis**

Model Cost Benefit Analysis Framework – Predictive Analytics

Three potential levels of analysis, depending on the problem context

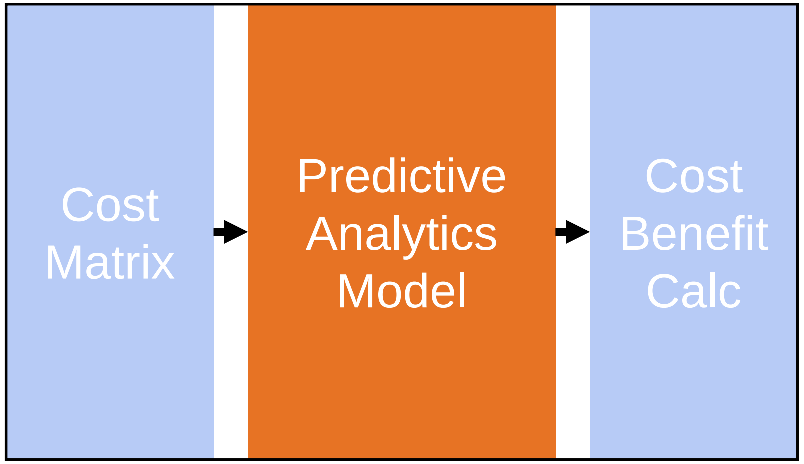
Lv 1 – Model Level



Treat model like a black box

Couple TP/TN/FP/FN with cost-benefit calculations

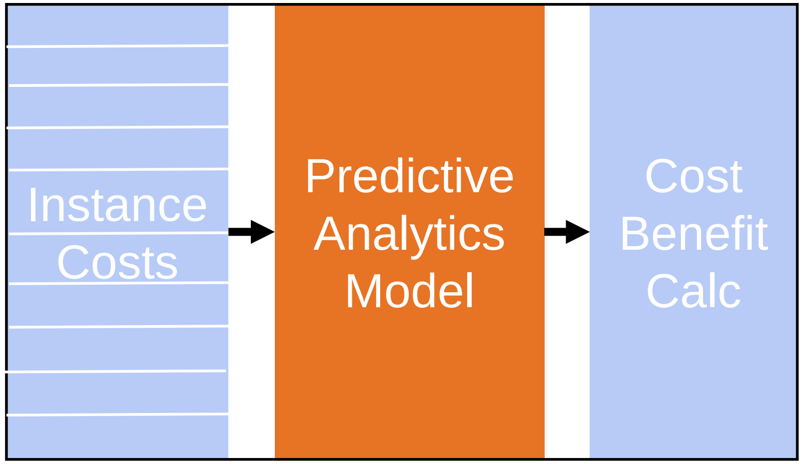
Lv 2 – Matrix Level



Input cost matrix ratios into model

Couple TP/TN/FP/FN with cost-benefit calculations

Lv 3 – Instance Level



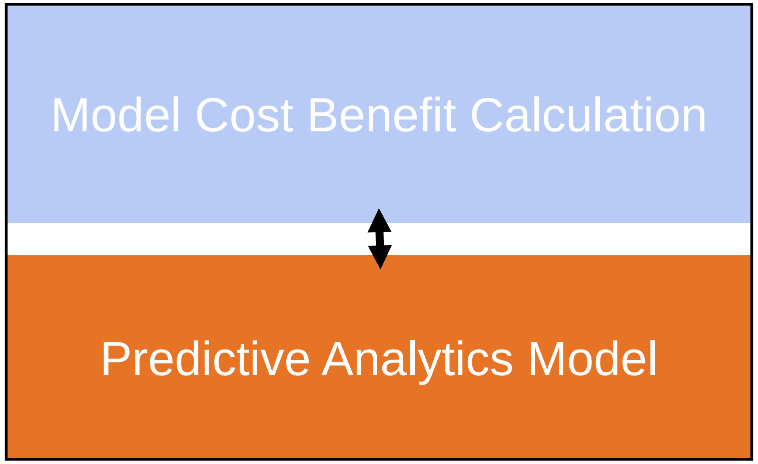
Input instance-level costs into model

Couple TP/TN/FP/FN with cost-benefit calculations

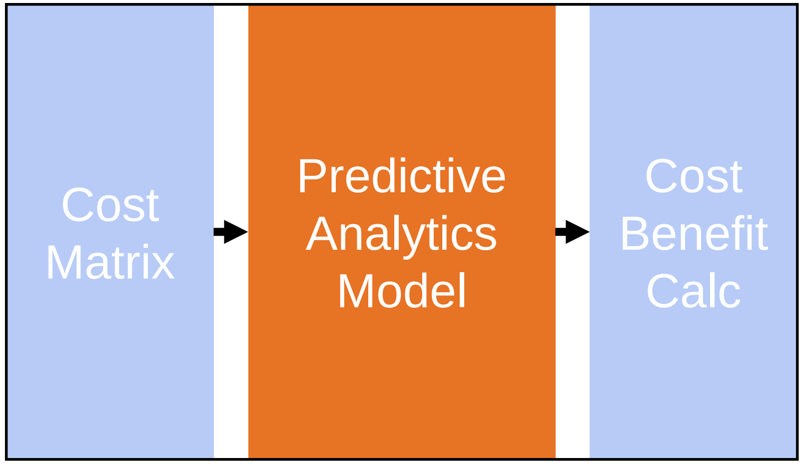
Model Cost Benefit Analysis Framework – Predictive Analytics

Three potential levels of analysis, depending on the problem context

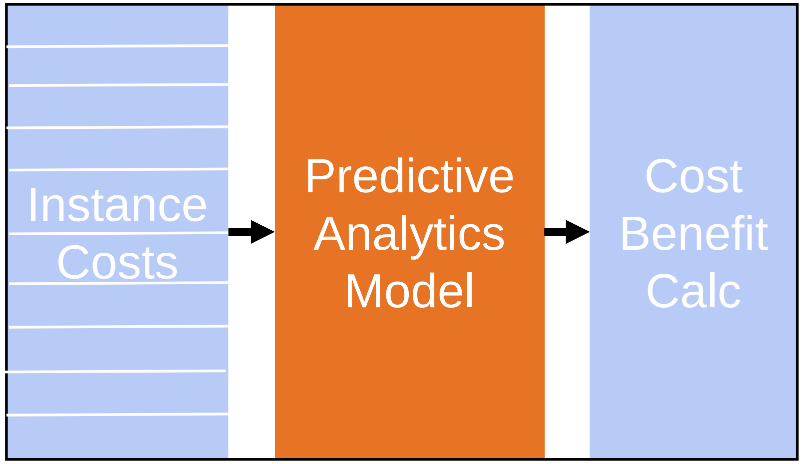
Lv 1 – Model Level



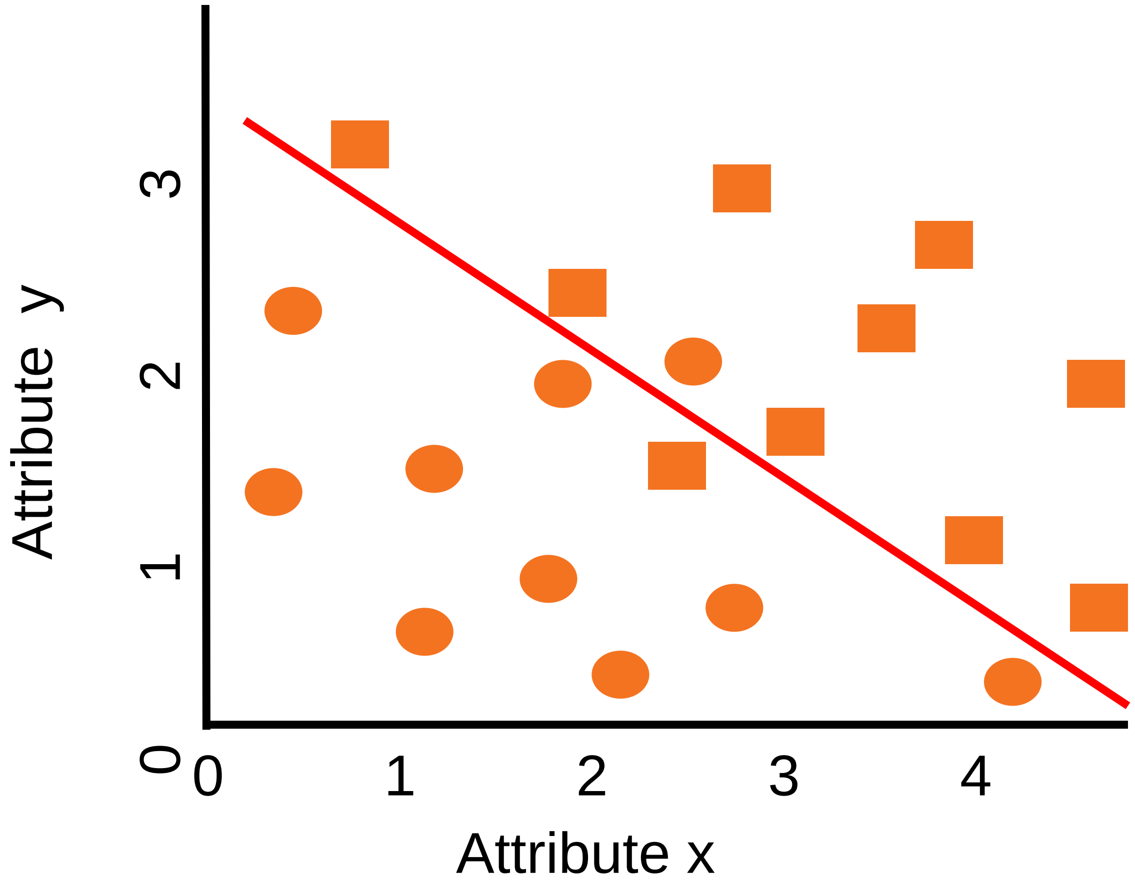
Lv 2 – Matrix Level



Lv 3 – Instance Level



Model Cost-benefit Analysis Framework –Motivation



Draw a single straight line that can best separate circles from squares (i.e., minimal error rate)

<u>Performance on Training Data:</u>

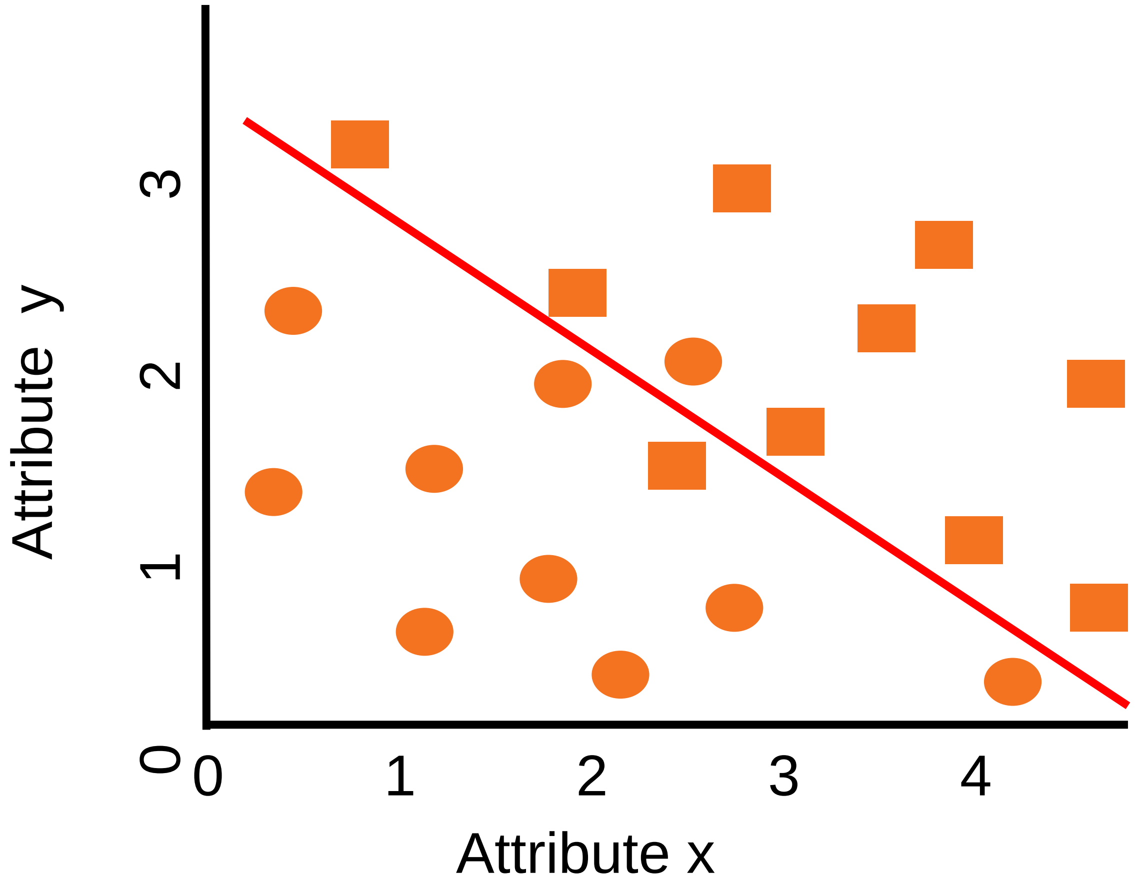
Accuracy = 18/20 = 0.90

Circle Recall = 9/10 = 0.90

Square Recall = 9/10 = 0.90

Cost of False Sq = 1; Cost of False Cr = 1

Model Cost-benefit Analysis Framework – Asymmetric Costs – Lvl 1



Draw a single straight line that can best separate circles from squares (i.e., minimal error rate)

Lv 1 Total Cost = 7

<u>Performance on Training Data:</u>

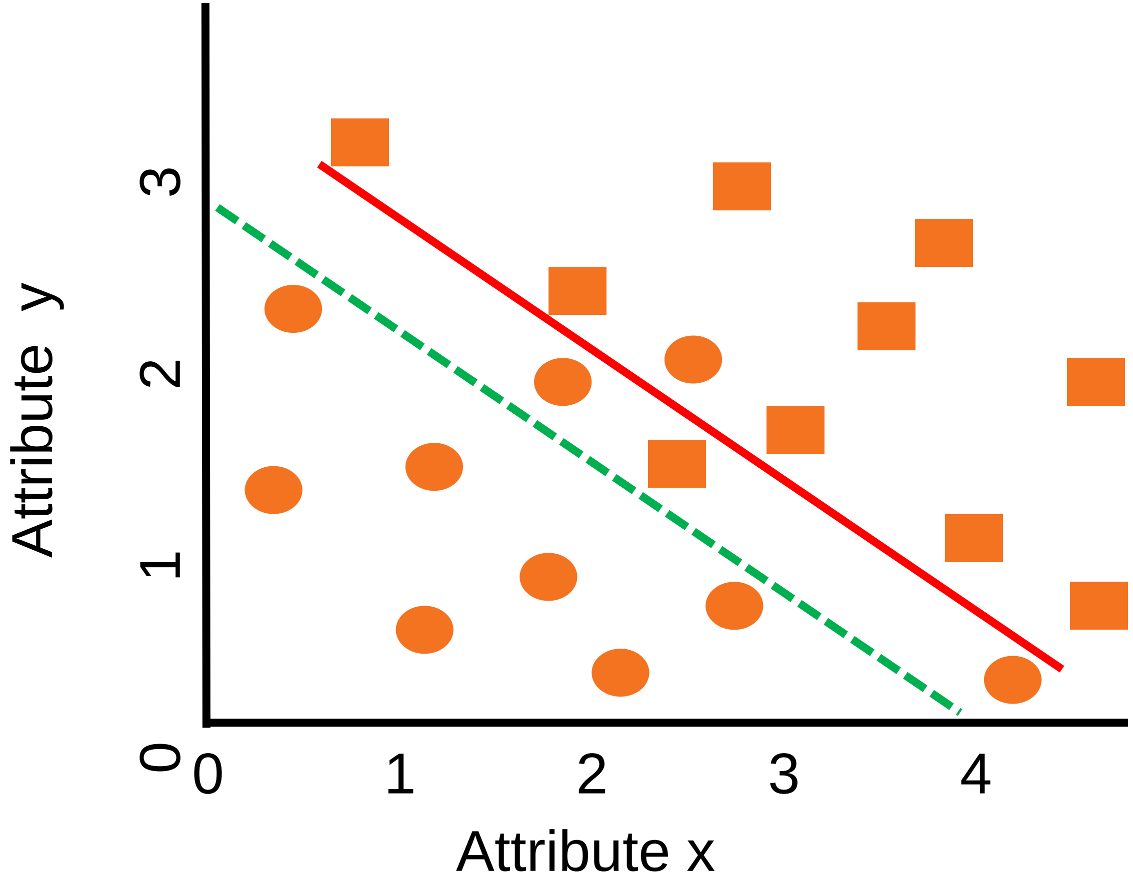
Accuracy = 18/20 = 0.90

Circle Recall = 9/10 = 0.90

Square Recall = 9/10 = 0.90

Cost of False Sq = 2; Cost of False Cr = 5

Model Cost-benefit Analysis Framework – Asymmetric Costs – Lvl 2



Draw a single straight line that can best separate circles from squares (i.e., minimal cost)

Performance on Training Data:
Accuracy = 17/20 = 0.85 0.90
Circle Recall = 7/10 = 0.70 0.90
Square Recall = 10/10 = 1.00 0.90
Cost of False Sq = 2; Cost of False Cr = 5

Lv 2 Total Cost = 6

Lv 1 Total Cost = 7

Model Cost-benefit Analysis Framework – Predictive Analytics Lv 2

Matrix level analysis example - **XGBoost**

Weight of FP….FP/FN

Lv 2 – Matrix Level

Cost Matrix → Predictive Analytics Model → Cost Benefit Calc

4. Train 3 Models with Different FP/FN Cost Ratios

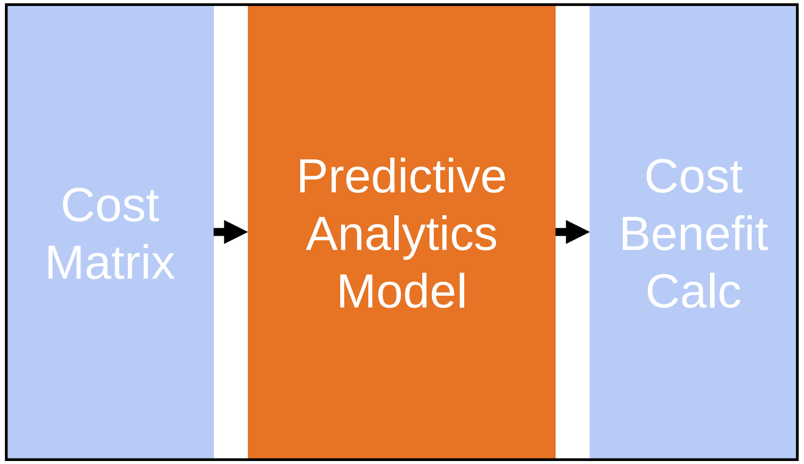
```
#define XGBoost parameter settings
depth=3
estimators=3
lr=0.3

# fit the unweighted model
clf = XGBClassifier(objective="binary:logistic", max_depth=depth, n_estimators=estimators, learning_ra
te=lr, n_jobs=16)
clf.fit(X, y)

# fit the weighted model 0.1
wclf = XGBClassifier(objective="binary:logistic", max_depth=depth, n_estimators=estimators, learning_r
ate=lr, n_jobs=16, scale_pos_weight=0.1)
wclf.fit(X, y)

# fit the weighted model 10
w2clf = XGBClassifier(objective="binary:logistic", max_depth=depth, n_estimators=estimators, learning_
rate=lr, n_jobs=16, scale_pos_weight=10)
w2clf.fit(X, y)
```

Model Cost-benefit Analysis Framework – Predictive Analytics Lv 2

Matrix level analysis example #2 - **SVM**

Ratio of FN to FP….FN : FP

Lv 2 – Matrix Level



4. Train 3 Models with Different FP/FN Cost Ratios

```
In [4]: # NOTE: Here, we are building three different SVM models. Pay special attention to the class_weight parameter
        # This parameter signifies the ratio of FP to FN costs.
        # In this particular context, we have "Cost of Misclassifying Blue : Cost of Misclassifying Orange"

        # fit the model and get the separating hyperplane
        clf = svm.SVC(kernel='linear', C=1.0)
        clf.fit(X, y)

        # fit the model and get the separating hyperplane using weighted classes
        wclf = svm.SVC(kernel='linear', class_weight={1: 10})
        wclf.fit(X, y)

        # fit the model and get the separating hyperplane using weighted classes
        w2clf = svm.SVC(kernel='linear', class_weight={1: 50})
        w2clf.fit(X, y)
```

?

VIRGINIA