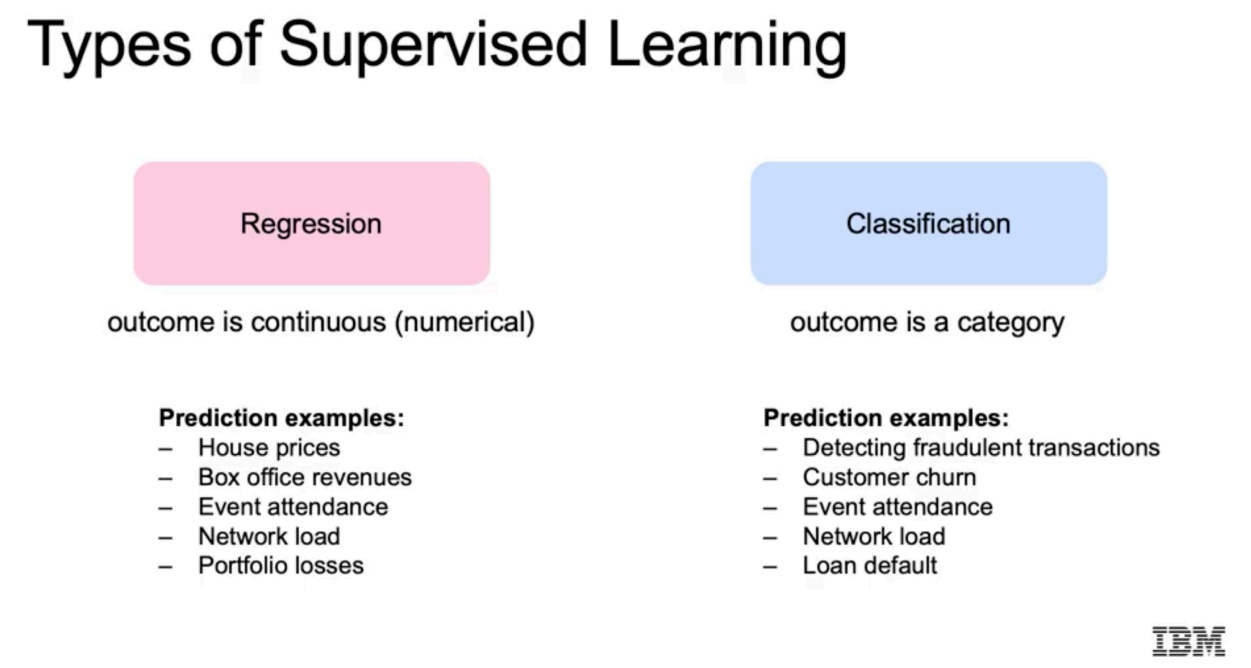
Supervised Machine Learning - Classification

Link: https://www.coursera.org/learn/supervised-machine-learning-classification/lecture/M9hsa/introduction-what-is-classification

By: Vadhna Samedy Hun

I. Introduction



Logistic vs. Linear Regression

$$P(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x + \varepsilon)}}$$

Logistic Function

$$P(x) = \frac{e^{(\beta_0 + \beta_1 x)}}{1 + e^{(\beta_0 + \beta_1 x)}}$$

$$\frac{P(x)}{1 - P(x)} = e^{(\beta_0 + \beta_1 x)}$$

Odds Ratio

IBM

Error Measurements

Accuracy = $\frac{TP + TN}{TP + FN + FP + TN}$

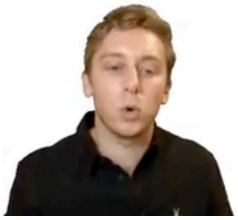
Precision = $\frac{TP}{TP + FP}$

Recall or Sensitivity = $\frac{TP}{TP + FN}$

Specificity = $\frac{TN}{FP + TN}$

F1 = $2 \frac{Precision * Recall}{Precision + Recall}$

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)



IBM

```
def evaluate_metrics(yt, yp):
    results_pos = {}
    results_pos['accuracy'] = accuracy_score(yt, yp)
    precision, recall, f_beta, _ = precision_recall_fscore_support(yt, yp)
    results_pos['recall'] = recall
    results_pos['precision'] = precision
    results_pos['f1score'] = f_beta
    return results_pos

sns.set_context('talk')
disp = ConfusionMatrixDisplay(confusion_matrix=cf,display_labels=l1_model.clas
disp.plot()
plt.show()
```

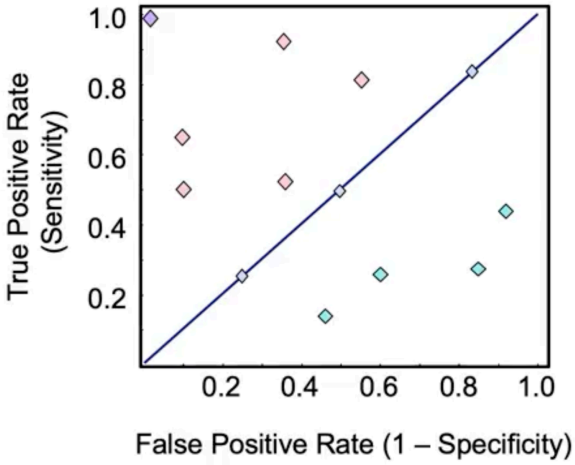
Imagine that we play a treasure hunt game.

— Sensitivity (Recall): How good are you at finding ALL the real treasures? It tells you how many real treasures you found out of all the ones that exist. For example: there are 10 and you found 8: 8/10=80%.

— Precision: How many of the things you picked were ACTUALLY treasures? It tell you how many of the things you thought were treasures were actually treasures.

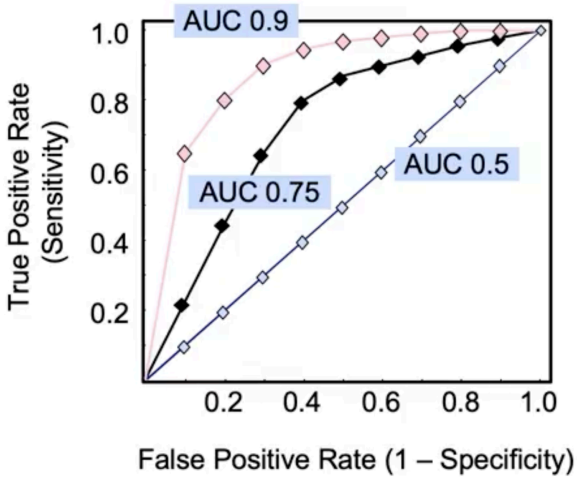
1.1. ROC and Precision-Recall Curve

Receiver Operating Characteristic (ROC)



IBM

Receiver Operating Characteristic (ROC)



Measures total area under ROC curve

IBM

Choosing the Right Approach

Which approach works best for choosing a classifier?

ROC Curve:
– Generally better for data with balanced classes.

Precision-Recall Curve:
– Generally better for data with imbalanced classes.

The right curve depends on tying results (true positives, true negatives, etc.) to outcomes (relative cost of false positive or false negative).

The curves compare classifiers generally (across possible decision thresholds), which may be less relevant to business objectives.

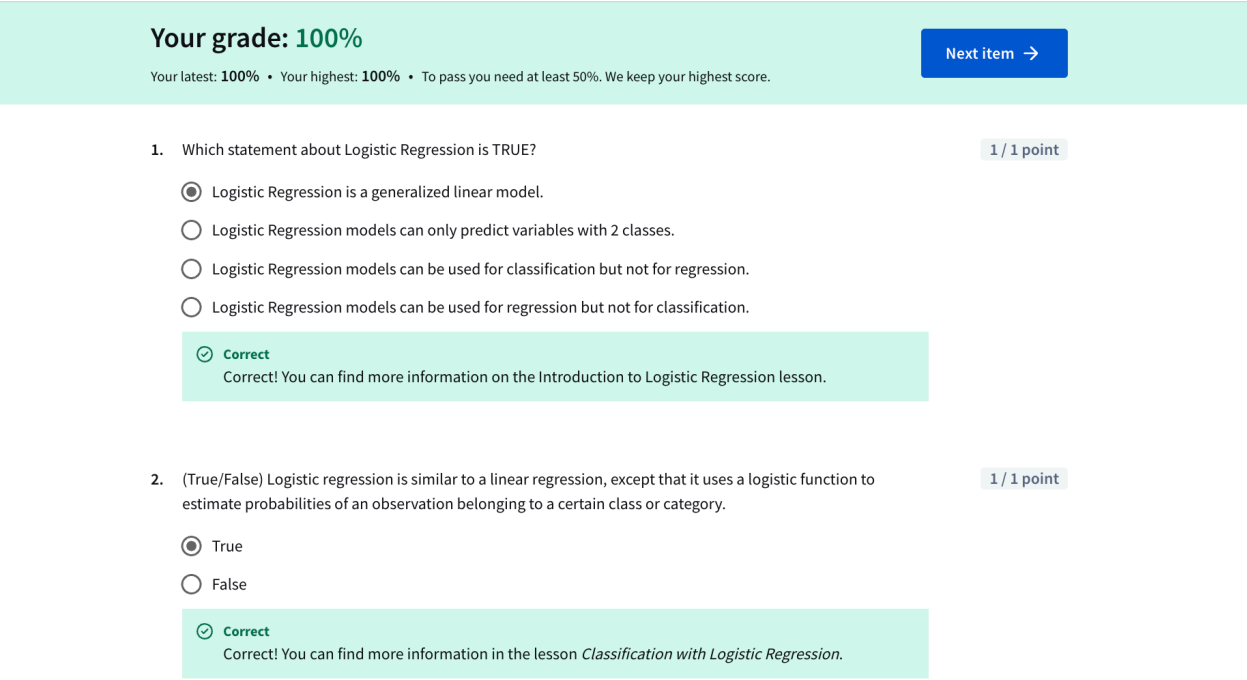
IBM

Classification Error Metrics: The Syntax

Code

```
# Import the desired error function
from sklearn.metrics import accuracy_score

# Calculate the error on the test and predicted data sets
accuracy_value = accuracy_score(y_test, y_pred)

# Lots of other error metrics and diagnostic tools:
from sklearn.metrics import precision_score, recall_score,
        f1_score, roc_auc_score,
        confusion_matrix, roc_curve,
        precision_recall_curve
```

IBM

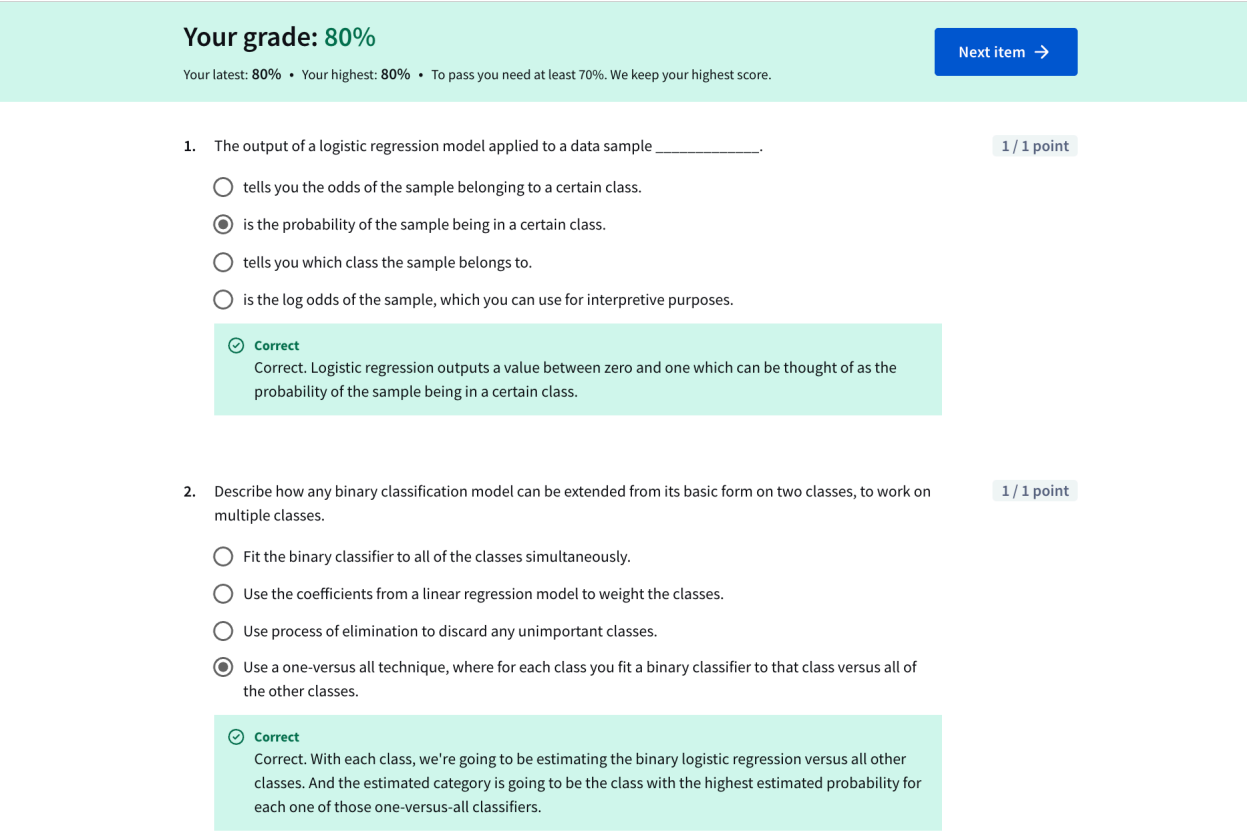


— Logistic Regression uses regression under the hood. And then it applies the logistic function to it to calculate a probability that the result belongs to a certain class.

```
# Extract and sort feature coefficients
def get_feature_coefs(regression_model, label_index, columns):
    coef_dict = {}
    for coef, feat in zip(regression_model.coef_[label_index, :], columns):
        if abs(coef) >= 0.01:
            coef_dict[feat] = coef
    # Sort coefficients
    coef_dict = {k: v for k, v in sorted(coef_dict.items(), key=lambda item: item[1])}
    return coef_dict

# Generate bar colors based on if value is negative or positive
def get_bar_colors(values):
```

```
    color_vals = []
    for val in values:
        if val <= 0:
            color_vals.append('r')
        else:
            color_vals.append('g')
    return color_vals

# Visualize coefficients
def visualize_coefs(coef_dict):
    features = list(coef_dict.keys())
    values = list(coef_dict.values())
    y_pos = np.arange(len(features))
    color_vals = get_bar_colors(values)
    plt.rcdefaults()
    fig, ax = plt.subplots()
    ax.barh(y_pos, values, align='center', color=color_vals)
    ax.set_yticks(y_pos)
    ax.set_yticklabels(features)
    # labels read top-to-bottom
    ax.invert_yaxis()
    ax.set_xlabel('Feature Coefficients')
    ax.set_title('')
    plt.show()
```

Quiz 1:



1. It gives you probabilities in respect with the number of classes you put in and then it chooses the maximum one for the result.
2. Under the hood, it automatically uses one versus all technique.

3. Which tool is most appropriate for measuring the performance of a classifier on unbalanced classes? 1 / 1 point

- ○ The Receiver Operating Characteristic (ROC) curve.
- ◉ The precision-recall curve.
- ○ The true positive rate.
- ○ The false positive rate.

Correct
Correct. The precision-recall curve displays the precision vs recall for different probability thresholds. Both precision and recall are focused on the positive class, which is normally the minority class in imbalanced classification problems.

4. (True/False) One of the requirements of logistic regression is that you need a variable with two classes. 1 / 1 point

- ○ True
- ◉ False

Correct
Correct! You can use a multinomial logistic regression if you have more than two classes. You can review the demo in lesson 2 of this module, in which you did a multinomial logistic to predict a target variable with more than two classes.

4. Although it uses one versus all by default, you can definitely tweak the algorithm for multiclass classification.

5. (True/False) The shape of ROC curves are the leading indicator of an overfitted logistic regression. 1 /

- ○ True
- ◉ False

Correct
Correct! Although overfitted models tend to have really high ROC curves with high values of area under the curve, a classification matrix or a measure like accuracy can be more reliable. Please review the lesson *Confusion Matrix, Accuracy, Specificity, Precision, and Recall.*

6. **Consider this scenario for Questions 3 to 7.** 1 /

You are evaluating a binary classifier. There are 50 positive outcomes in the test data, and 100 observations. Using a 50% threshold, the classifier predicts 40 positive outcomes, of which 10 are incorrect.

What is the classifier's Precision on the test sample?

- ○ 25%
- ○ 60%
- ◉ 75%
- ○ 80%

Correct
Correct! You can find more information in the lesson *Confusion Matrix, Accuracy, Specificity, Precision, and Recall.*

5. False.
6. Precision= TP/TP+FP = 30/(30+10) = 0.75.

7. **Consider this scenario for Questions 3 to 7.** 1 point

You are evaluating a binary classifier. There are 50 positive outcomes in the test data, and 100 observations. Using a 50% threshold, the classifier predicts 40 positive outcomes, of which 10 are incorrect.

What is the classifier's Recall on the test sample?

- ○ 25%
- ○ 60%
- ○ 75%
- ◉ 80%

Incorrect
Incorrect. Please review the lesson *Confusion Matrix, Accuracy, Specificity, Precision, and Recall.*

8. **Consider this scenario for Questions 3 to 7.** 1 / 1 point

You are evaluating a binary classifier. There are 50 positive outcomes in the test data, and 100 observations. Using a 50% threshold, the classifier predicts 40 positive outcomes, of which 10 are incorrect.

What is the classifier's F1 score on the test sample?

- ○ 50%
- ◉ 66.7%
- ○ 67.5%
- ○ 70%

Correct
Correct! You can find more information in the lesson *Confusion Matrix, Accuracy, Specificity, Precision, and Recall.*

7. Recall = TP / TP+FN = 30/(30+(50-30)) = 0.6 (I did not know why I chose 80%).
8. F1 = 2*(Precision * Recall)/(Precision + Recall) = 2*(0.6*0.8)/(0.6+0.8) = 66.7%.

9. **Consider this scenario for Questions 3 to 7.** 1 / 1 point

You are evaluating a binary classifier. There are 50 positive outcomes in the test data, and 100 observations. Using a 50% threshold, the classifier predicts 40 positive outcomes, of which 10 are incorrect.

Increasing the threshold to 60% results in 5 fewer positive predictions, all of which are correct. Which of the following statements about this new model (compared with the original model that had a 50% threshold) is TRUE?

- ○ The F1 score of the classifier would decrease.
- ○ The area under the ROC curve would decrease.
- ○ The F1 score of the classifier would remain the same.
- ◉ The area under the ROC curve would remain the same.

Correct
Correct! For more information, please review the lesson *ROC and Precision-Recall Curves.*

10. **Consider this scenario for Questions 3 to 7.** 1 point

You are evaluating a binary classifier. There are 50 positive outcomes in the test data, and 100 observations. Using a 50% threshold, the classifier predicts 40 positive outcomes, of which 10 are incorrect.

The threshold is now increased further, to 70%. Which of the following statements is TRUE?

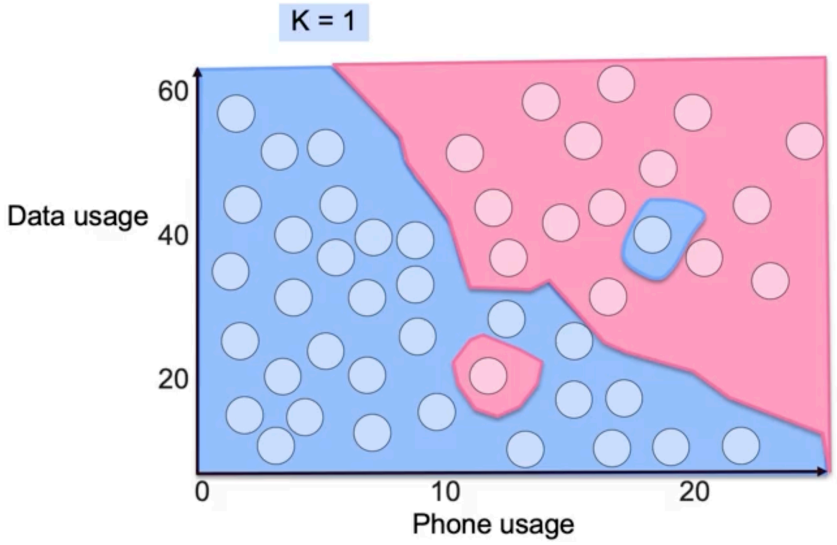
- ○ The Recall of the classifier would Increase.
- ○ The Precision of the classifier would decrease.
- ◉ The Recall of the classifier would increase or remain the same.
- ○ The Precision of the classifier would increase or remain the same.

Incorrect
InCorrect! For more information, please review the lesson *ROC and Precision-Recall Curves.*

9. ROC plots sensitivity (Recall) on the y-axis and (1-specificity) on the x-axis as the threshold increases, both decrease. Thus, the area under curve would remain the same.
10. I need to review that lesson.

II. K-Nearest Neighbors

K Nearest Neighbors Decision Boundary

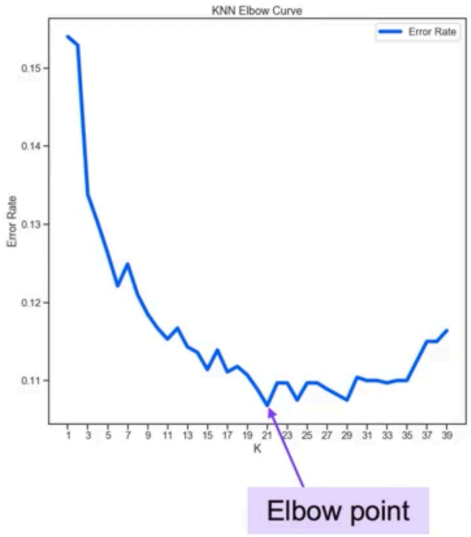


IBM

K Nearest Neighbors Decision Boundary

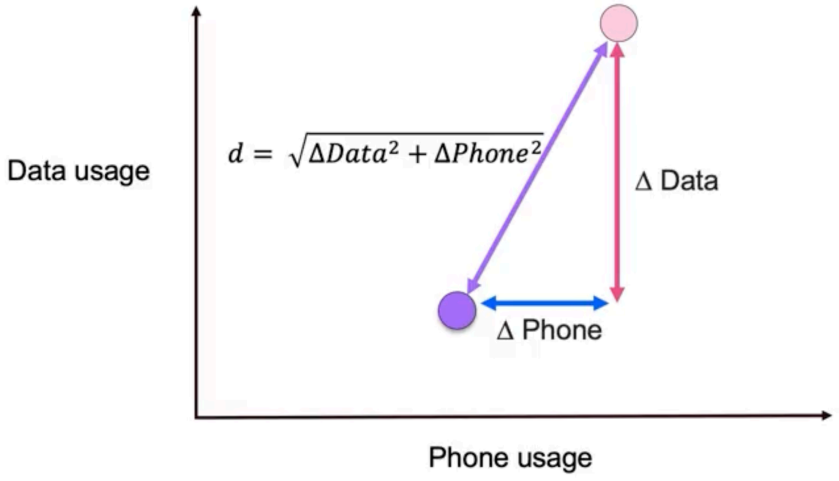
Choosing the right value for K

- KNN does not provide a 'correct' *K*
- The right value depends on which error metric is most important
- A common approach is to use an 'elbow method' approach
- This emphasizes kinks in a curve of the error rate as a function of K
- Beyond this point, the rate of improvement slows or stops



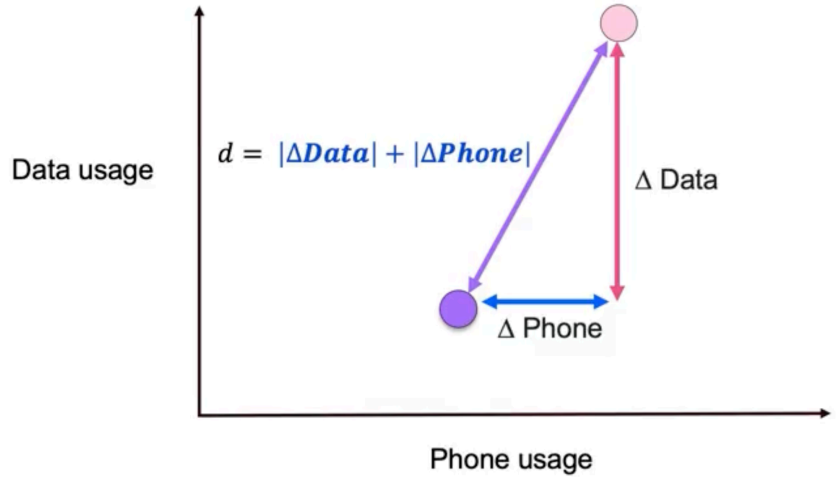
IBM

Euclidean Distance (L2)



IBM

Manhattan Distance (L1)



IBM

KNN Overview

Pros:
- Simple to implement (does not require estimation)
- Adapts well as new training data
- Easy to interpret

Cons:
- Slow to predict because many distance calculations
- Does not generate insight into data generating process (no model)
- Can require lots of memory if data set is large (or as it grows)
- When there are many predictors, KNN accuracy can break down due to curse of dimensionality

IBM

K Nearest Neighbors: The Syntax

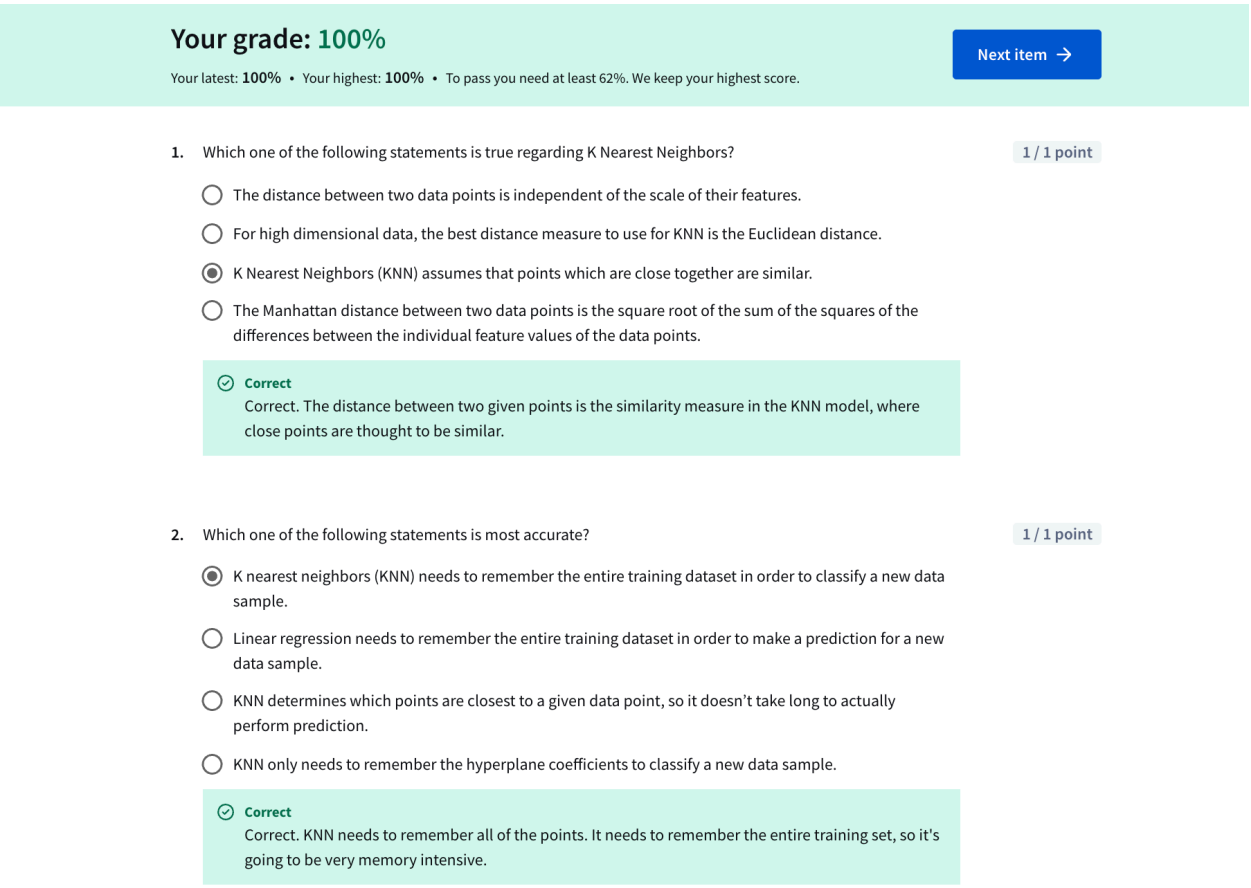
Code

```
# Import the class containing the classification method
from sklearn.neighbors import KNeighborsClassifier

# Create an instance of the class
KNN = KNeighborsClassifier(n_neighbors=3)

# Fit the instance on the data and then predict the expected value
KNN = KNN.fit(X_train, y_train)
y_predict = KNN.predict(X_test)

The fit and predict/transform syntax will show up throughout the course.
```

IBM

— K nearest neighbor methods are useful for classification. The elbow method is frequently used to identify a model with low K and low error rate.

Quiz 2:



— KNN works similar to group. It creates a decision boundary and then it classifies each point based on their distance to the decision boundary. And thus points that are nearby tend to be in the same class.

3. Which one of the following statements is most accurate about K Nearest Neighbors (KNN)? 1 / 1 point

- ○ KNN is an unsupervised learning method.
- ◉ KNN can be used for both classification and regression.
- ○ KNN is a regression model.
- ○ KNN is a classification model.

Correct
Correct. KNN is known as a classification model, but can also be used for regression. All you have to do is replace KNeighborsClassifier with KNeighborsRegressor.

4. (True/False) K Nearest Neighbors with large *k* tend to be the best classifiers. 1 / 1 point

- ○ True
- ◉ False

Correct
Correct! K Nearest Neighbors with high values of k might likely not generalize well with new data. A best practice is to use the elbow method to find a model with low k and high decrease in error.

5. When building a KNN classifier for a variable with 2 classes, it is advantageous to set the neighbor count k to an odd number. 1 / 1 point

- ◉ True
- ○ False

Correct
Correct! An odd neighbor count works as a tie breaker. It ensures there cannot be a tie in the number of n nearest neighbors for two given classes. You can find more information on the k nearest neighbor lesson.

6. The Euclidean distance between two points will always be shorter than or equal to the Manhattan distance. 1 / 1 point

- (●) True
- ( ) False

Correct
Correct! From trigonometry, you should realize that Euclidian distance is shorter than the Manhattan distance. You can review this on the K Nearest Neighbors lesson.

7. The main purpose of scaling features before fitting a k nearest neighbor model is to: 1 / 1 point

- (●) Ensure that features have similar influence on the distance calculation
- ( ) Break ties in case there is the same number of neighbors of different classes next to a given observation
- ( ) Ensure decision boundaries have roughly the same size for all classes
- ( ) Help find the appropriate value of k

Correct
Correct! You can find more information in the K Nearest Neighbor lesson.

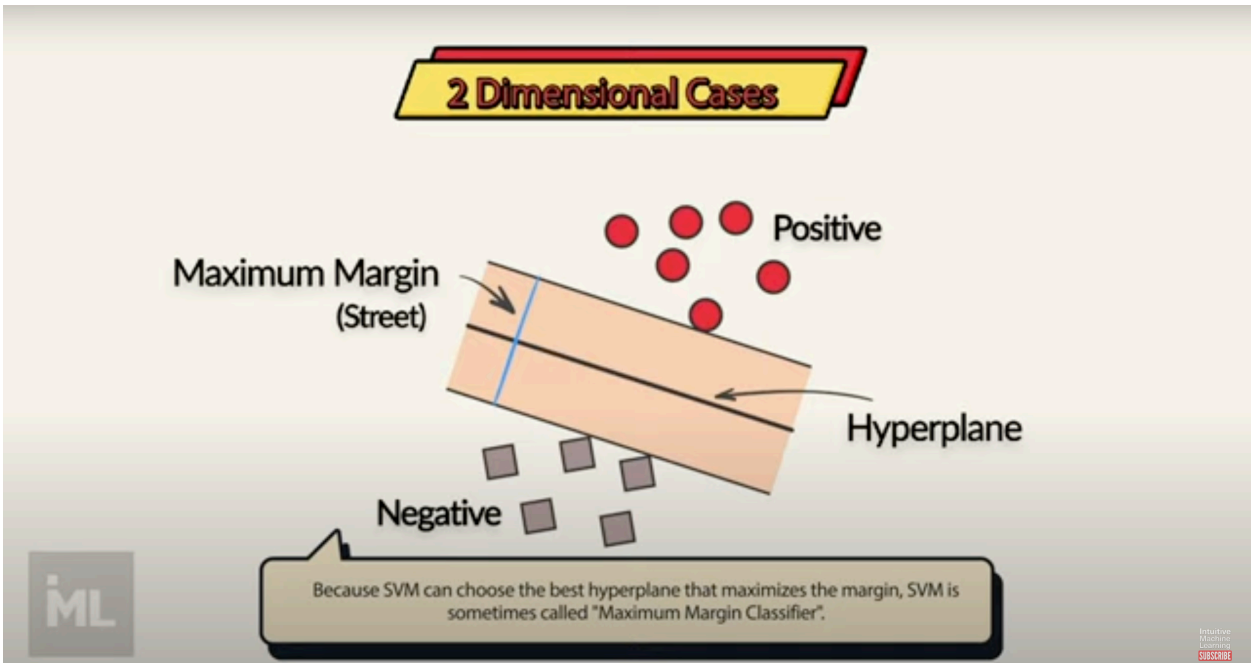
8. These are all pros of the k nearest neighbor algorithm EXCEPT: 1 / 1 point

- ( ) It is easy to interpret
- ( ) It adapt wells to new training data
- ( ) It is simple to implement as it does not require parameter estimation
- (●) It is sensitive to the curse of dimensionality

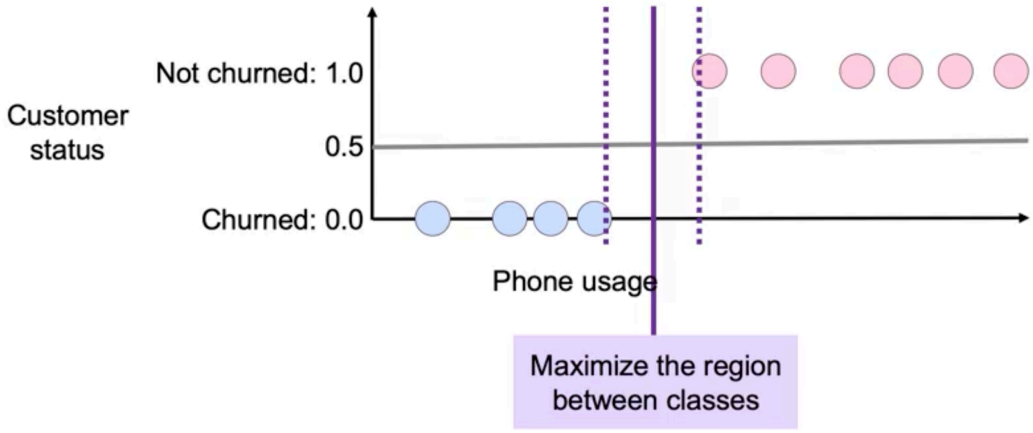
Correct
Correct! You can find more information in the K Nearest Neighbor lesson.

III. Support Vector Machine

SVM tries to find hyperplanes that have the maximum margin. The hyperplanes are determined by support vectors (data points have the smallest distance to the hyperplanes). Meanwhile, in order to reduce model variance, the SVM model aims to find the maximum possible margins so that unseen data will be more likely to be classified correctly



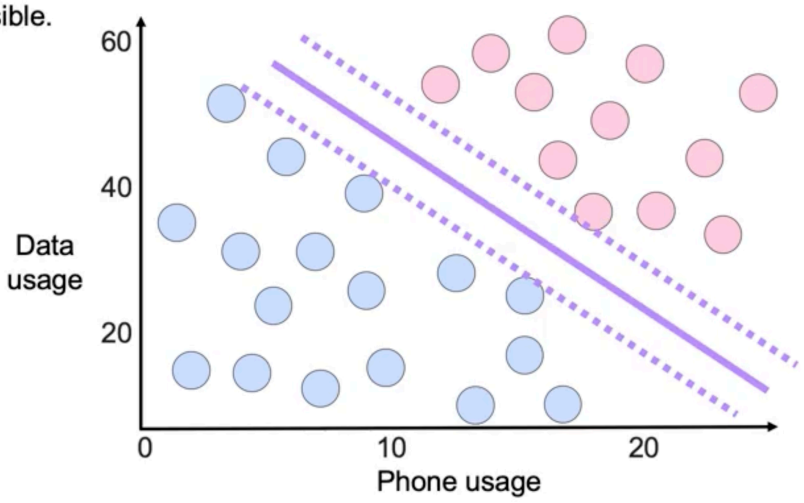
Support Vector Machines (SVM)



IBM

Classification with SVMs

And include the
largest boundary possible.

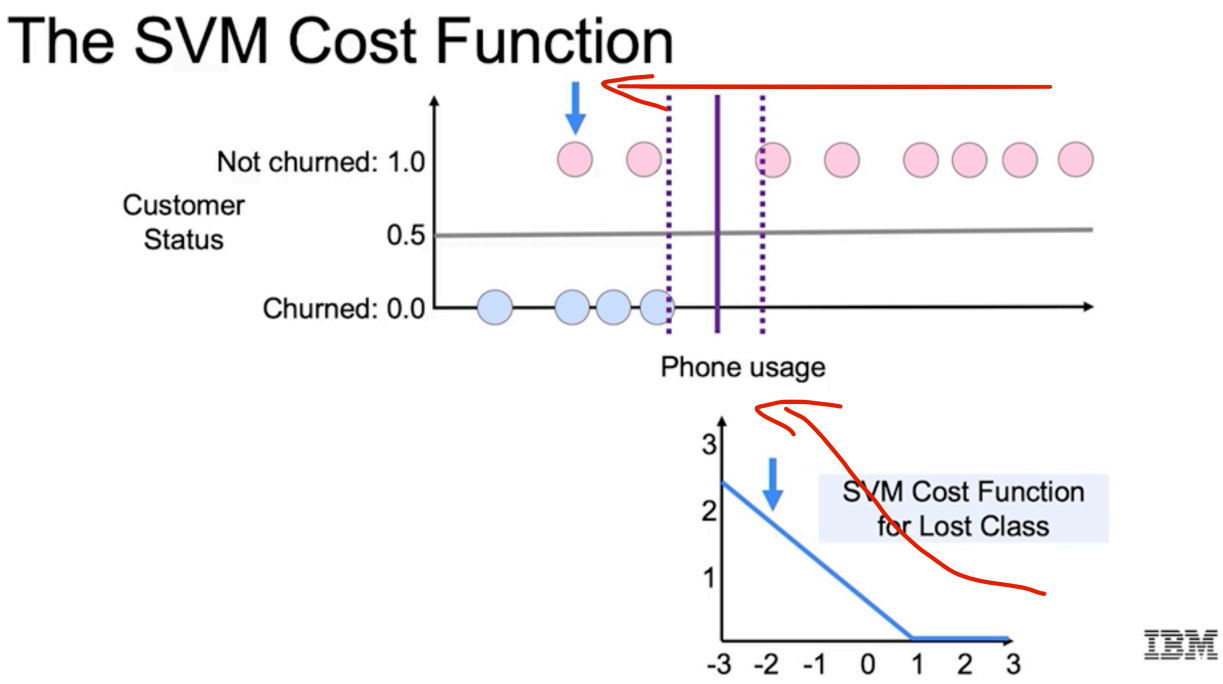


IBM

```
def plot_decision_boundary(estimator, X, y):
    estimator.fit(X, y)
    X_color = X.sample(300)
    y_color = y.loc[X_color.index]
    y_color = y_color.map(lambda r: 'red' if r == 1 else 'yellow')
    x_axis, y_axis = np.arange(0, 1, .005), np.arange(0, 1, .005)
    xx, yy = np.meshgrid(x_axis, y_axis)
    xx_ravel = xx.ravel()
    yy_ravel = yy.ravel()
    X_grid = pd.DataFrame([xx_ravel, yy_ravel]).T
    y_grid_predictions = estimator.predict(X_grid)
    y_grid_predictions = y_grid_predictions.reshape(xx.shape)

    fig, ax = plt.subplots(figsize=(10, 10))
    ax.contourf(xx, yy, y_grid_predictions, cmap=plt.cm.autumn_r, alpha=.3)
    ax.scatter(X_color.iloc[:, 0], X_color.iloc[:, 1], color=y_color, alpha=1)
    ax.set(
        xlabel=fields[0],
```

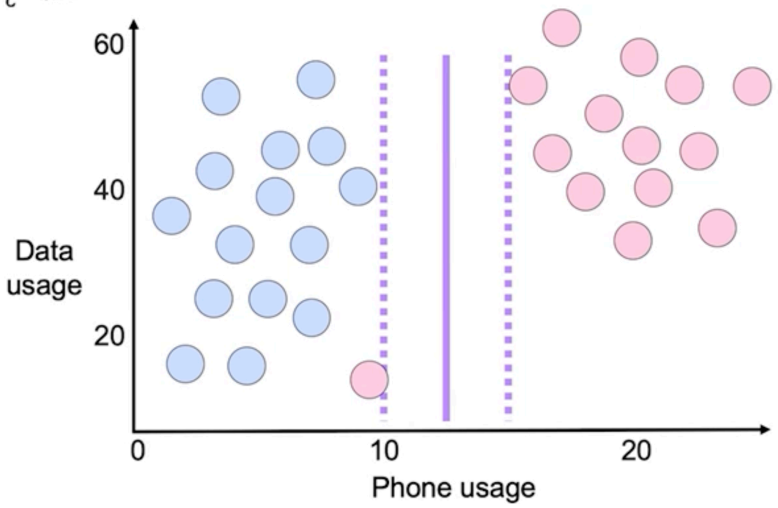
```
        ylabel=fields[1],
        title=str(estimator))
```

— SVM is a linear classifier. We know that they will not return probabilities, but rather return labels either one or zero, and those labels are decided by which side of a certain decision boundary they fall on.



Regularization in SVMs

$$J(\beta) = SVMCost(\beta) + \frac{1}{C}\sum_i \beta_i$$



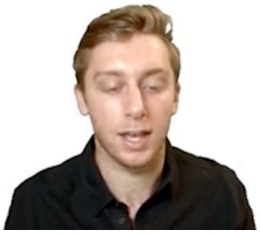
IBM

Linear SVM: The Syntax

Code

```
# Import the class containing the classification method
from sklearn.svm import LinearSVC

# Create an instance of the class
LinSVC = LinearSVC(penalty='l2', C=10.0)

# Fit the instance on the data and then predict the expected value
LinSVC = LinSVC.fit((X_train, y_train)
y_predict = LinSVC.predict(X_test)

Tune regularization parameters with cross-validation.

Use LinearSVM for regression.
```

regularization parameters

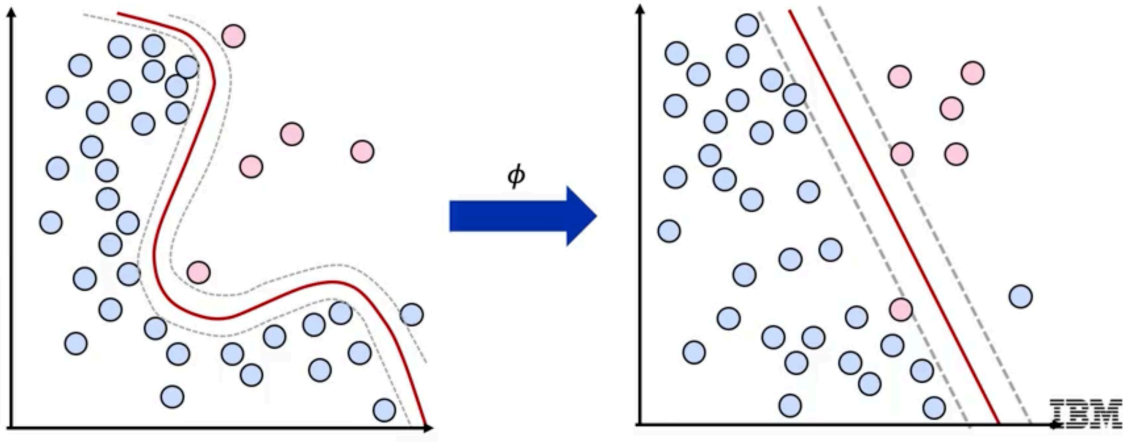


IBM

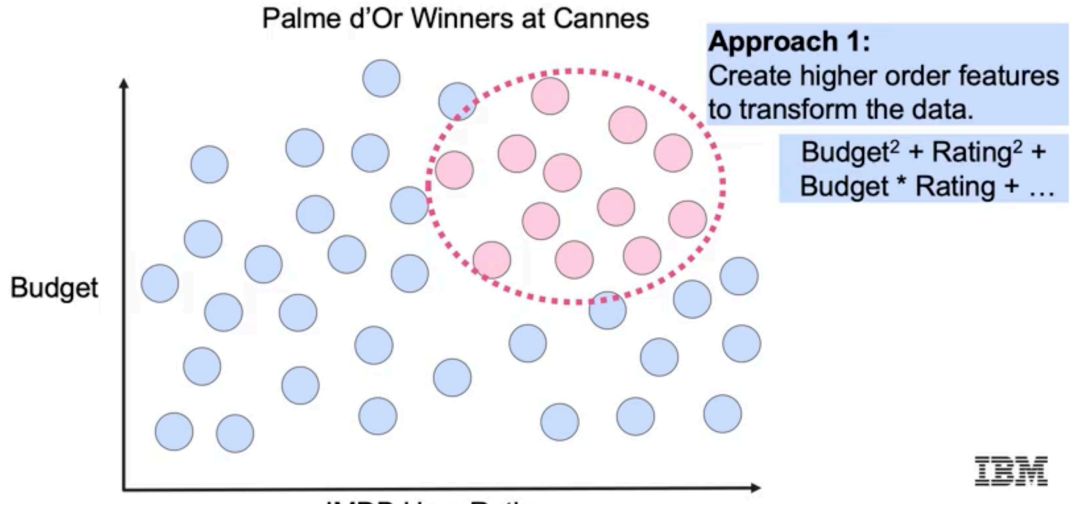
— A SVM model is a linear model that does not output predicted probabilities. Instead, it outputs labels determined by the decision boundary assigned to the region where an observation belongs. You can find more information in the lesson *Regularization in Support Vector Machines*.

Non-Linear Decision Boundaries with SVM

Non-linear data can be made linear with higher dimensionality



SVM Gaussian Kernel



SVMs with Kernels: The Syntax

Code

```
# Import the class containing the classification method
from sklearn.svm import SVC

# Create an instance of the class
rbfSVC = SVC(kernel='rbf', gamma=1.0, C=10.0)

# Fit the instance on the data and then predict the expected value
rbfSVC = rbfSVC.fit((X_train, y_train)
y_predict = rbfSVC.predict(X_test)

Tune kernel and associated parameters with cross-validation.
```

"C" is penalty associated with the error term

IBM

Machine Learning Workflow

Features	Data	Model Choice
- Many (~10K Features)	- Small (1K rows)	- Simple, Logistic or LinearSVC
- Few (<100 Features)	- Medium (~10k rows)	- SVC with RBF
- Few (<100 Features)	- Many (>100K Points)	- Add features, Logistic, LinearSVC, or Kernel Approx.

IBM

Faster Kernel Transformations: The Syntax

Code

```
# Import the class containing the classification method
from sklearn.kernel_approximation import Nystroem

# Create an instance of the class
NystroemSVC = Nystroem (kernel='rbf', gamma=1.0,
n_components=100)

# Fit the instance on the data and transform
X_train = NystroemSVC.fit_transform(X_train)
X_test = NystroemSVC.transform(X_test)

Tune kernel and associated parameters with cross-validation.
```

n_components is number of samples

IBM

Faster Kernel Transformations: The Syntax

Code

```
# Import the class containing the classification method
from sklearn.kernel_approximation import RBFsampler

# Create an instance of the class
rbfSample = RBFsampler (gamma=1.0, n_components=100)

# Fit the instance on the data and transform
X_train = rbfSample.fit_transform(X_train)
X_test = rbfSample.transform(X_test)

Tune kernel parameters and components with cross-validation.
```

parameter names are identical to previous

IBM

```
def plot_decision_boundary(estimator, X, y):
    estimator.fit(X, y)
    X_color = X.sample(300)
```

```
    y_color = y.loc[X_color.index]
    y_color = y_color.map(lambda r: 'red' if r == 1 else 'yellow')
    x_axis, y_axis = np.arange(0, 1, .005), np.arange(0, 1, .005)
    xx, yy = np.meshgrid(x_axis, y_axis)
    xx_ravel = xx.ravel()
    yy_ravel = yy.ravel()
    X_grid = pd.DataFrame([xx_ravel, yy_ravel]).T
    y_grid_predictions = estimator.predict(X_grid)
    y_grid_predictions = y_grid_predictions.reshape(xx.shape)

    fig, ax = plt.subplots(figsize=(10, 10))
    ax.contourf(xx, yy, y_grid_predictions, cmap=plt.cm.autumn_r, alpha=.3)
    ax.scatter(X_color.iloc[:, 0], X_color.iloc[:, 1], color=y_color, alpha=1)
    ax.set(
        xlabel=fields[0],
        ylabel=fields[1],
        title=str(estimator))
```

Quiz 3:

1. Select the TRUE statement regarding the cost function for SVMs: 1 / 1 point

- (●) SVMs use the Hinge Loss function as a cost function
- ( ) SVMs use same loss function as logistic regression
- ( ) SVMs use a loss function that penalizes vectors prone to misclassification
- ( ) SVMs do not use a cost function. They use regularization instead of a cost function.

Correct
Correct! You can find more information in the lesson *The Support Vector Machines Cost Function*.

2. Which statement about Support Vector Machines is TRUE? 1 / 1 point

- ( ) Support Vector Machine models can be used for regression but not for classification.
- ( ) Support Vector Machine models are non-linear.
- (●) Support Vector Machine models rarely overfit on training data.
- ( ) Support Vector Machine models can be used for classification but not for regression.

Correct
Correct! You can find more information in the lesson *Regularization in Support Vector Machines*.

3. (True/False) A large *c* term will penalize the SVM coefficients more heavily. 1 / 1 point

- ( ) True
- (●) False

Correct
Correct! You can find more information in the lesson Regularization in Support Vector Machines.

4. Regularization in the context of support vector machine (SVM) learning is meant to _____. 1 / 1 point

- ○ bring all features to a common scale to ensure they have equal weight
- ◉ lessen the impact that some minor misclassifications have on the cost function
- ○ encourage the model to ignore outliers during training
- ○ smooth the input data to reduce the chance of overfitting

Correct
Correct. In SVM, you have to come up with a way of optimizing to allow for some points to be misclassified within the process. This is where the regularization in SVM comes into play.

5. Support vector machines can be extended to work with nonlinear classification boundaries by _____. 1 / 1 point

- ◉ using the kernel trick
- ○ projecting the feature space onto a lower dimensional space
- ○ modifying the standard sigmoid function
- ○ incorporating polynomial regression

Correct
Correct. Support vector machines can be extended to non-linear classifiers using the kernel trick.

8. SVM with kernals can be very slow on large datasets. To speed up SVM training, which methods may you perform to map low dimensional data into high dimensional beforehand? 1 / 1 point

- [x] RBF Sampler

Correct
Correct. The RBF Sampler method can be used to map low dimensional data into high dimensional data.

- [ ] Linear SVC
- [ ] Regularization
- [x] Nystroem

Correct
Correct. The Nystroem method can be used to map low dimensional data into high dimensional data.

9. Concerning the Machine Learning workflow what model choice would you pick if you have "Few" features and a "Medium" amount of data? 1 / 1 point

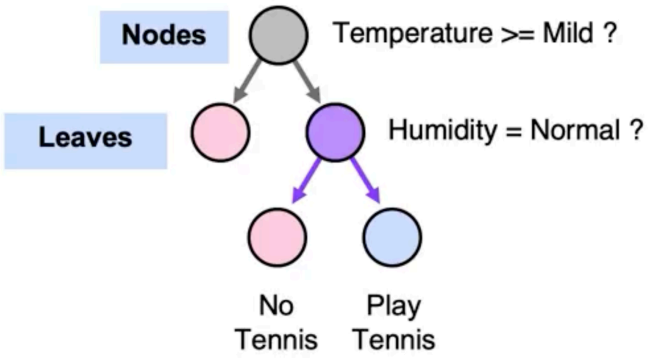
- (•) SVC with RBF
- ( ) LinearSVC, or Kernal Approximation
- ( ) Simple, Logistic or LinearSVC
- ( ) Add features, or Logistic

Correct
Correct. You would use SVC with RBF as your model with “Few” features and a “Medium” amount of data.

IV. Classifiers

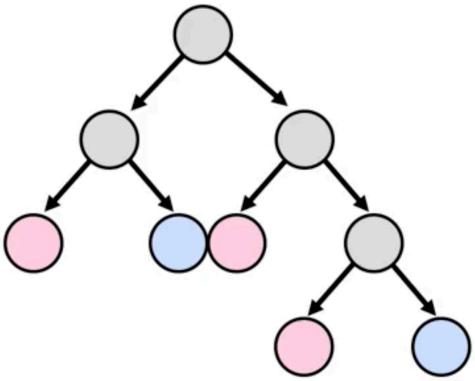
Introduction to Decision Trees

- Want to predict whether customers will play tennis based on temperature, humidity, wind, outlook
- Segment data based on features to predict result
- Trees that predict categorical results are **decision trees**



IBM

How Long to Keep Splitting?

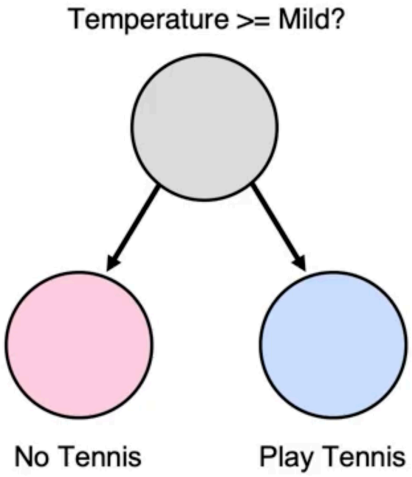


Until:

- Leaf node(s) are **pure** (only **one class** remains).

IBM

Building the Best Decision Tree



- Use **greedy search**: find the best split at each step.
- What defines the best split?
- One that maximizes the information gained from the split.
- How is information gain defined?

IBM

DecisionTreeClassifier: The Syntax

Code

```
# Import the class containing the classification method
from sklearn.tree import DecisionTreeClassifier

# Create an instance of the class
DTC = DecisionTreeClassifier (criterion='Gini',
max_features=10, max_depth=5)

# Fit the instance on the data and then predict the expected
value
DTC = DTC.fit(X_train, y_train)
y_predict = DTC.predict(X_test)

Tune parameters with cross-validation. Use
DecisionTreeRegressor for regression.
```

tree parameters

IBM

```
def measure_error(y_true, y_pred, label):
    return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
                      'precision': precision_score(y_true, y_pred),
```

```
                'recall': recall_score(y_true, y_pred),
                'f1': f1_score(y_true, y_pred)},
                name=label)

params_grid = {
   "max_depth" : range(1, dt.tree_.max_depth+1, 2),
   "max_features" : range(1, len(dt.feature_importances_)+1)
}

GR = GridSearchCV(DecisionTreeClassifier(random_state=42),
             param_grid=params_grid,
             scoring="accuracy",
             n_jobs=-1)
GR = GR.fit(X_train, y_train)
```

```
# Assuming 'data' is your DataFrame and you have already defined the feature c
feature_cols = [x for x in X.columns]  # Adjust this based on your dataset

# For the original decision tree (assuming `dt` is your trained model)
plt.figure(figsize=(12, 8))
plot_tree(dt, filled=True, feature_names=feature_cols, class_names=['Low', 'Med
plt.title('Decision Tree')
plt.savefig('wine_tree.png')  # Save the tree as an image
plt.show()
```

5.1. Random Forest

```
def get_accuracy(X_train, X_test, y_train, y_test, model):
   return  {"test Accuracy":metrics.accuracy_score(y_test, model.predict(X_test))
```

```
RF = RandomForestClassifier(
    oob_score=True,
    random_state=42,
    warm_start=True,
    n_jobs=-1
)

oob_list = list()
# number of trees
for n_trees in [15, 20, 30, 40, 50, 100, 150, 200, 300, 400]:

    # Use this to set the number of trees
    RF.set_params(n_estimators=n_trees)

    # Fit the model
    RF.fit(X_train, y_train)

    # Get the oob error
    oob_error = 1 - RF.oob_score_

    # Store it
    oob_list.append(pd.Series({'n_trees': n_trees, 'oob': oob_error}))

rf_oob_df = pd.concat(oob_list, axis=1).T.set_index('n_trees')

rf_oob_df
```

QUIZ 4:

Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 70%. We keep your highest score.

Next item →

1. These are all characteristics of decision trees, EXCEPT: 1 / 1 point

- ○ They can be used for either classification or regression
- ○ They split nodes into leaves
- ○ They segment data based on features to predict results
- ◉ They have well rounded decision boundaries

Correct
Correct! For more information please review the *Introduction to Decision Trees* lesson.

2. Decision trees used as classifiers compute the value assigned to a leaf by calculating the ratio: number of observations of one class divided by the number of observations in that leaf E.g. number of customers that are younger than 50 years old divided by the total number of customers. 1 / 1 point

How are leaf values calculated for regression decision trees?

- ◉ average value of the predicted variable
- ○ mode value of the predicted variable
- ○ weighted average value of the predicted variable
- ○ median value of the predicted variable

Correct
Correct! For more information please review the *Decision TreeSplit* lesson.

3. These are two main advantages of decision trees: 1 / 1 point

- ○ They output both parameters and significance levels
- ◉ They are very visual and easy to interpret
- ○ They do not tend to overfit and are not sensitive to changes in data
- ○ They are resistant to outliers and output scaled features

Correct
Correct! For more information please review the lesson Pros and Cons of Decision Trees.

4. How can you determine the split for each node of a decision tree? 1 / 1 point

- ○ Find the split that induces the largest entropy.
- ○ Randomly select the split.
- ◉ Find the split that minimizes the gini impurity.
- ○ Use a nonlinear decision boundary to find the best split.

Correct
Correct. The split for each node is determined by the split that minimizes the gini impurity or the entropy.

5. Which of the following describes a way to regularize a decision tree to address overfitting? 1 / 1 point

- ○ Increase the max depth.
- ○ Increase the number of branches.
- ◉ Decrease the max depth.
- ○ Reduce the information gain.

Correct
Correct. Decreasing the max depth can help to prune the tree. Any impure leaves can be assigned to the majority class.

6. What is a disadvantage of decision trees? 1 / 1 point

- ○ Scaling is required.
- ◉ They tend to overfit.
- ○ They can get too large.
- ○ They are difficult to interpret.

Correct
Correct. Decision trees tend to add high variance; that is, they tend to overfit.

7. What method can you use to minimize overfitting of a machine learning model? 1 / 1 point

- ○ Decrease the variance of your test data.
- ○ Choose the hyperparameters that maximize goodness of fit on your training data.
- ◉ Tune the hyperparameters of your model using cross-validation.
- ○ Increase the variance of your training data.

Correct
Correct. As you've done with other algorithms, you want to tune your hyperparameters using cross-validation. This means choosing the hyperparameters that maximize the estimated goodness of fit on unseen test data.

8. Concerning Classification algorithms, what is a characteristic of K-Nearest Neighbors? 1 / 1 point

- ◉ Training data is the model
- ○ The model is just parameters
- ○ Fitting can be slow
- ○ Prediction is fast

Correct
Correct. For K-Nearest Neighbors the training data is the model.

9. Concerning Classification algorithms, what are the characteristics of Logistic Regression? 1 / 1 point

- ◉ The model is just parameters, fitting can be slow, prediction is fast, and the decision boundary is simple and less flexible
- ○ The training data is the model, fitting is fast, prediction is fast, and the decision boundary is flexible
- ○ The model is just parameters, fitting is fast, prediction is fast, and the decision boundary is flexible
- ○ The training data is the model, fitting is fast, predicting class for new records can be slow, and the decision boundary is flexible

Correct
Correct.

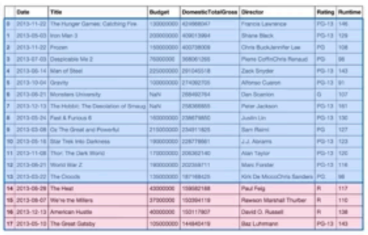
10. When evaluating all possible splits of a decision tree what can be used to find the best split regardless of what happened in prior or future steps? 1 / 1 point

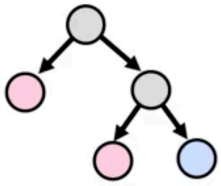
- ○ Regularization
- ◉ Greedy Search
- ○ Classification
- ○ Logistic regression

Correct
Correct. Greedy Search is used to find the best split regardless of what happened in prior or future steps.

V. Ensemble Based Methods

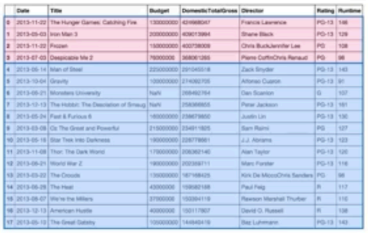
Bagging Error Calculations

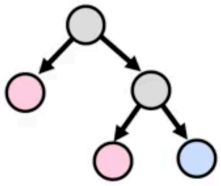




Same as decision trees:

- Easy to interpret and implement
- Heterogeneous input data allowed, no preprocessing required





Specific to bagging:

- Less variability than decision trees
- Can grow trees in parallel

IBM

BaggingClassifier: The Syntax

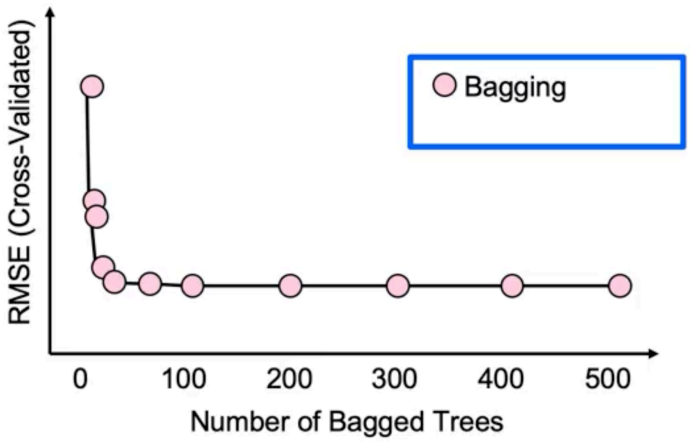
Code

```
# Import the class containing the classification method
from sklearn.ensemble import BaggingClassifier

# Create an instance of the class
BC = BaggingClassifier(n_estimators=50)

# Fit the instance on the data and then predict the expected value
BC = BC.fit(X_train, y_train)
y_predict = BC.predict(X_test)

Tune parameters with cross-validation. Use BaggingRegressor for regression.
```

IBM

Introducing More Randomness

- **Solution**:
  further de-correlate trees.
- Use random subset of features for each tree.
  - **Classification**: $\sqrt{m}$
  - **Regression**: $m/3$
- Called "**Random Forest**".



IBM

RandomForest: The Syntax

Code

```
# Import the class containing the classification method
from sklearn.ensemble import RandomForestClassifier

# Create an instance of the class
RC = RandomForestClassifier(n_estimators=50)

# Fit the instance on the data and then predict the expected value
RC = RC.fit(X_train, y_train)
y_predict = RC.predict(X_test)

Tune parameters with cross-validation. Use RandomForestRegressor for regression.
```

IBM

ExtraTreesClassifier: The Syntax

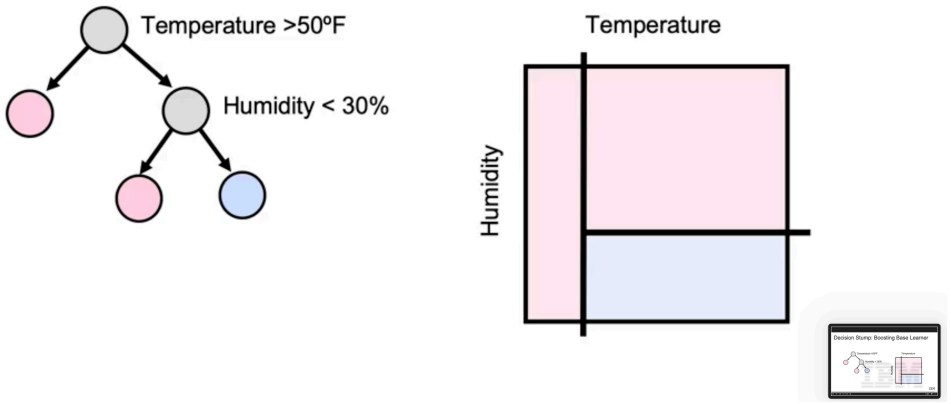
Code

```
# Import the class containing the classification method
from sklearn.ensemble import ExtraTreesClassifier

# Create an instance of the class
EC = ExtraTreesClassifier (n_estimators=50)

# Fit the instance on the data and then predict the expected value
EC = EC.fit(X_train, y_train)
y_predict = EC.predict(X_test)
```

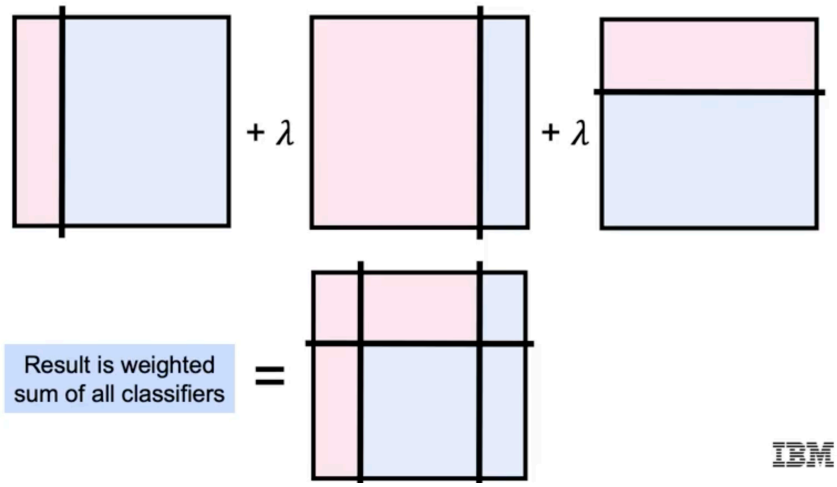
Tune parameters with cross-validation. Use **ExtraTreesRegressor** for regression.

IBM

Decision Stump: Boosting Base Learner



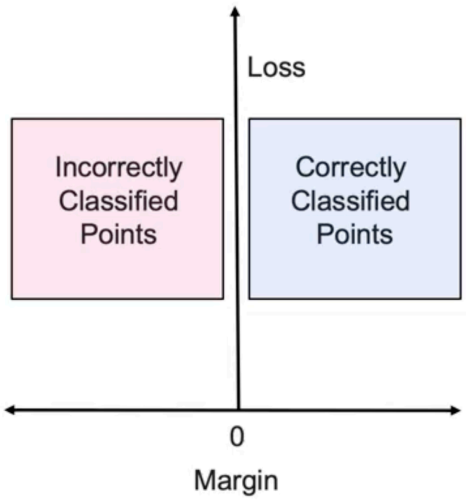
Overview of Boosting



IBM

Boosting Specifics

- Boosting utilizes different loss functions.
- At each stage, the margin is determined for each point.
- Margin is positive for correctly classified points and negative for misclassifications.
- Value of loss function is calculated from margin.

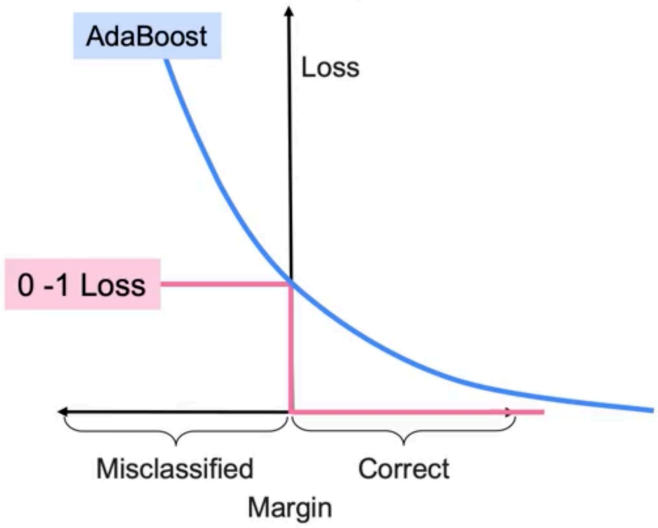


IBM

Gradient Boosting Loss Function

- Generalized boosting method that can use different loss functions.
- Common implementation uses binomial log likelihood loss function (**deviance**):

$$\log(1 + e^{(-margin)})$$



IBM

Bagging vs Boosting

Bagging	Boosting
– Bootstrapped samples	– Fit entire data set
– Base trees created independently	– Base trees created successively
– Only data points considered	– Use residuals from previous models
– No weighting used	– Up-weight misclassified points
– Excess trees will not overfit	– Beware of overfitting

IBM

GradientBoostingClassifier: The Syntax

Code

```
# Import the class containing the classification method
from sklearn.ensemble import GradientBoostingClassifier

# Create an instance of the class
GBC = GradientBoostingClassifier (learning_rate=0.1,
        max_features=1, subsample=0.5,
        n_estimators=200)

# Fit the instance on the data and then predict the expected value
GBC = GBC.fit(X_train, y_train)
y_predict = GBC.predict(X_test)

Tune with cross-validation. Use GradientBoostingRegressor for regression.
```

IBM

AdaBoostClassifier: The Syntax

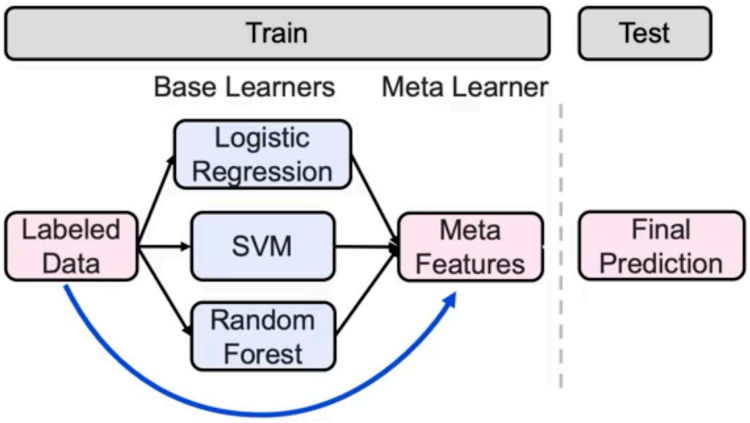
Code

```
# Import the class containing the classification method
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

# Create an instance of the class
ABC = AdaBoostClassifier (base_estimator=DecisionTreeClassifier(),
                          learning_rate=0.1, n_estimators=200)

# Fit the instance on the data and then predict the expected value
ABC = ABC.fit(X_train, y_train)
y_predict = ABC.predict(X_test)

Tune with cross-validation. Use AdaBoostRegressor for regression.
```

can also set max depth here

IBM

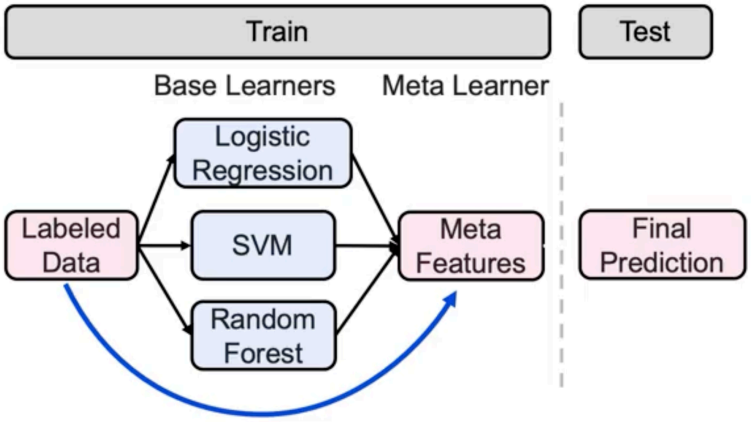
Stacking: Combining Classifiers



- Models of any kind can be combined to create a stacked model.
- Like bagging but not limited to decision trees.
- Output of base learners creates features, can recombine with data.

IBM

Stacking: Combining Classifiers



- Output of base learners can be combined via majority vote or weighted.
- Additional hold-out data needed if meta learner parameters are used.
- Be aware of increasing model complexity.
- The final prediction can be done by voting or with another model

IBM

VotingClassifier: The Syntax

Code

```
# Import the class containing the classification method
from sklearn.ensemble import VotingClassifier

# Create an instance of the class
VC = VotingClassifier(estimator_list)

# Fit the instance on the data and then predict the expected value
VC = VC.fit(X_train, y_train)
y_predict = VC.predict(X_test)

Use VotingRegressor for regression.

The StackingClassifier (or StackingRegressor) works similarly:
SC = StackingClassifier(estimator_list, final_estimator=LogisticRegression())
```

IBM

QUIZ:

Your grade: 80%

Your latest: **80%** • Your highest: **80%** • To pass you need at least 70%. We keep your highest score.

Next item →

1. The term *Bagging* stands for bootstrap aggregating. — 1 / 1 point

   - (•) True
   - ( ) False

   ✓ **Correct**
   Correct! You can find more information in the lesson: *Ensemble Based Methods and Bagging.*

2. This is the best way to choose the number of trees to build on a Bagging ensemble. — 1 point

   - ( ) Tune number of trees as a hyperparameter that needs to be optimized
   - (•) Choose a number of trees past the point of diminishing returns
   - ( ) Prioratize training error metrics over out of bag sample
   - ( ) Choose a large number of trees, typically above 100

   ✗ **Incorrect**
   Incorrect. Please review the lesson: *Ensemble Based Methods and Bagging.*

3. Which type of Ensemble modeling approach is NOT a special case of model averaging? 1 / 1 point

- ○ The Pasting method of Bootstrap aggregation
- ○ Random Forest methods
- ◉ Boosting methods
- ○ The Bagging method of Bootstrap aggregation

Correct
Correct! You can find more information in the lesson *Overview of Boosting.*

4. What is an ensemble model that needs you to look at out of bag error? 1 / 1 point

- ○ Stacking
- ○ Out of Bag Regression
- ◉ Random Forest
- ○ Logistic Regression.

Correct
Correct! You can find more information in the lesson *Random Forest.*

5. What is the main condition to use stacking as ensemble method? 1 / 1 point

- ○ Models need to be nonparametric
- ○ Models need to be parametric
- ○ Models need to output residual values for each class
- ◉ Models need to output predicted probabilities

Correct
Correct! You can find more information in the lesson *Stacking.*

6. This tree ensemble method only uses a subset of the features for each tree: 1 / 1 point

- ○ Bagging
- ◉ Random Forest
- ○ Stacking
- ○ Adaboost

Correct
Correct! This tree ensemble only uses a subset of the features for each tree. For more information, please review the Random Forest lesson.

7. Order these tree ensembles in order of most randomness to least randomness: 1 / 1 point

- ○ Random Forest, Random Trees, Bagging
- ◉ Random Trees, Random Forest, Bagging
- ○ Bagging, Random Forest, Random Trees
- ○ Random Forest, Bagging, Random Trees

Correct
Correct! Random Trees add one more degree of randomness than Random Forests and two more than Bagging. You can find more information in the Random Forest lesson.

8. This is an ensemble model that does not use bootstrapped samples to fit the base trees, takes residuals into account, and fits the base trees iteratively: 1 / 1 point

- ○ Random Forest
- ○ Bagging
- ◉ Boosting
- ○ Random Trees

Correct
Correct! These are all characteristics of boosting algorithms. You can find more information in the *Boosting* lesson.

9. When comparing the two ensemble methods Bagging and Boosting, what is one characteristic of Boosting? 1 / 1 point

- ○ Bootstraped samples
- ○ Only data points are considered
- ◉ Fits entire data set
- ○ No weighting used

Correct
Correct. With Boosting you can use the entire data set to train each of the classifiers

10. What is the most frequently discussed loss function in boosting algorithms? 1 point

- ○ Gradient Loss Function
- ○ AdaBoost Loss Function
- ◉ Gradient Boosting Loss Function
- ○ 0-1 Loss Function

Incorrect
Incorrect. Please review the *Adaboost and Gradient Boosting Overview*video.

VI. Unbalanced Classes

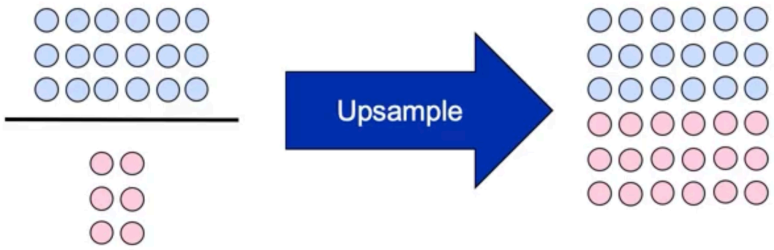
Unbalanced Classes

For unbalanced datasets, we can balance the size of the classes by either downsampling the larger class or upsampling the small one.

IBM

Unbalanced Classes

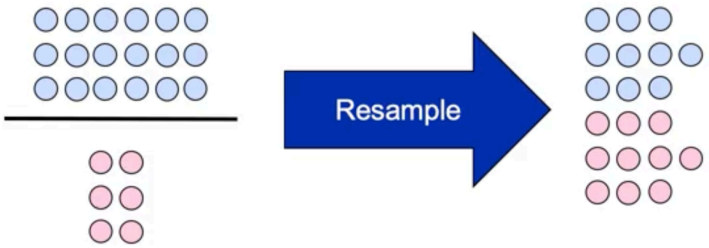
For unbalanced datasets, we can balance the size of the classes by either downsampling the larger class or upsampling the small one.



IBM

Unbalanced Classes

We can also do a mix of the two. In this example 6 < s < 19 for sampling. If we choose s = 10:



IBM

Unbalanced Classes

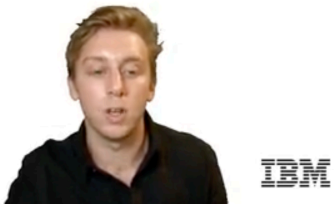
Steps for unbalanced datasets:

- Do a stratified test-train split
- Up or down sample the full dataset
- Build models

IBM

Stratified Sampling

- Train-test split, “stratify” option
- ShuffleSplit -> StratifiedShuffleSplit
- KFold -> StratifiedKFold -> RepeatedStratifiedKFold



QUIZ:

Your grade: 80%

Your latest: **80%** • Your highest: **80%** • To pass you need at least 70%. We keep your highest score.

Next item →

1. Which of the following statements about Downsampling is TRUE? — 1 point

- ○ Downsampling preserves all the original observations.
- ◉ Downsampling is likely to decrease Recall.
- ○ Downsampling results in excessive focus on the more frequently-occurring class.
- ○ Downsampling is likely to decrease Precision.

⊗ **Incorrect**
Incorrect. Please review the lesson *Upsampling and Downsampling.*

2. Which of the following statements about Random Upsampling is TRUE? — 1 point

- ○ Random Upsampling preserves all original observations.
- ◉ Random Upsampling generates observations that were not part of the original data.
- ○ Random Upsampling results in excessive focus on the more frequently-occurring class.
- ○ Random Upsampling will generally lead to a higher F1 score.

⊗ **Incorrect**
Incorrect. Please review the lesson *Upsampling and Downsampling.*

3. Which of the following statements about Synthetic Upsampling is TRUE? 1 / 1 point

- ○ Synthetic Upsampling uses fewer hyperparameters than Random Upsampling.
- ○ Synthetic Upsampling will generally lead to a higher F1 score.
- ◉ Synthetic Upsampling generates observations that were not part of the original data.
- ○ Synthetic Upsampling results in excessive focus on the more frequently-occurring class.

Correct
Correct! You can find more information in the lesson *Upsampling and Downsampling.*

4. What can help humans to interpret the behaviors and methods of Machine Learning models more easily? 1 / 1 point

- ○ Model Debug
- ◉ Model Explanations
- ○ Model Trust
- ○ Explanation Debug

Correct
Correct! Model explanations can help humans to interpret the behaviors and methods of Machine Learning models more easily

5. What type of explanation method can be used to explain different types of Machine Learning models no matter the model structures and complexity? 1 / 1 point

- ○ Model Explanations
- ◉ Model-Agnostic Explanations
- ○ Model Trust Explanations
- ○ Local Interpretable Model-Agnostic Explanations (LIME)

Correct
Correct! The Model-Agnostic explanation can be used to describe different types of Machine Learning models no matter the complexity while also having the same formats and presentations for model explanations?

6. What reason might a Global Surrogate model fail? 1 / 1 point

- ○ Single data instance groups
- ○ Single clusters in the data instance groups
- ○ Consistency between surrogate models and black-box models
- ◉ Large inconsistency between surrogate models and black-box models

Correct
Correct! A Global Surrogate model might fail if there is a large inconsistency between surrogate models and black-box models.

7. When working with unbalanced sets, what should be done to the samples so the class balance remains consistent in both the train and test set? 1 / 1 point

- ○ Use oversampling
- ◉ Stratify the samples
- ○ Use a combination of oversampling and undersampling
- ○ Apply weighted observations

Correct
Correct! You should stratify the samples so the class balance remains consistent in both the train and test set.

8. What approach are you using when trying to increase the size of a minority class so that it is similar to the size of the majority class? 1 / 1 point

- ○ Random Oversampling
- ○ Synthetic Oversampling
- ○ Undersampling
- ◉ Oversampling

Correct
Correct! You are oversampling when trying to increase the size of a minority class so that it is similar to the size of the majority class

9. What approach are you using when you create a new sample of a minority class that does not yet exist? 1 / 1 point

- ○ Oversampling
- ○ Random Oversampling
- ○ Weighting
- ◉ Synthetic Oversampling

Correct
Correct! Synthetic Oversampling is an approach used to create a new sample of a minority class that does not yet exist.

10. What intuitive technique is used for unbalanced datasets that ensures a continuous downsample for each of the bootstrap samples? 1 / 1 point

- ○ SMOTE
- ◉ Blagging
- ○ Upsampling
- ○ Downsampling

Correct
Correct! Blagging is an intuitive technique used for unbalanced datasets that ensures a continuous downsample for each of the bootstrap samples.