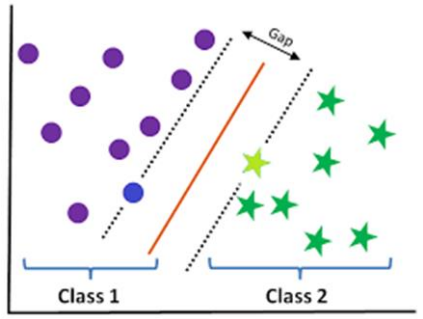
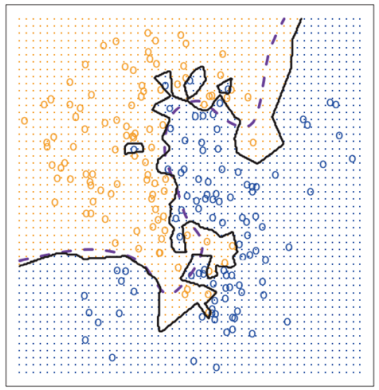
Intro to Classification

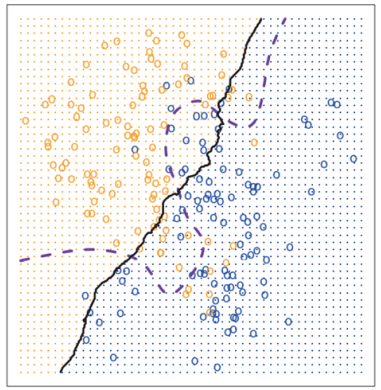
Intuition behind Classification

Identify whether a comment is positive or negative

Predict category





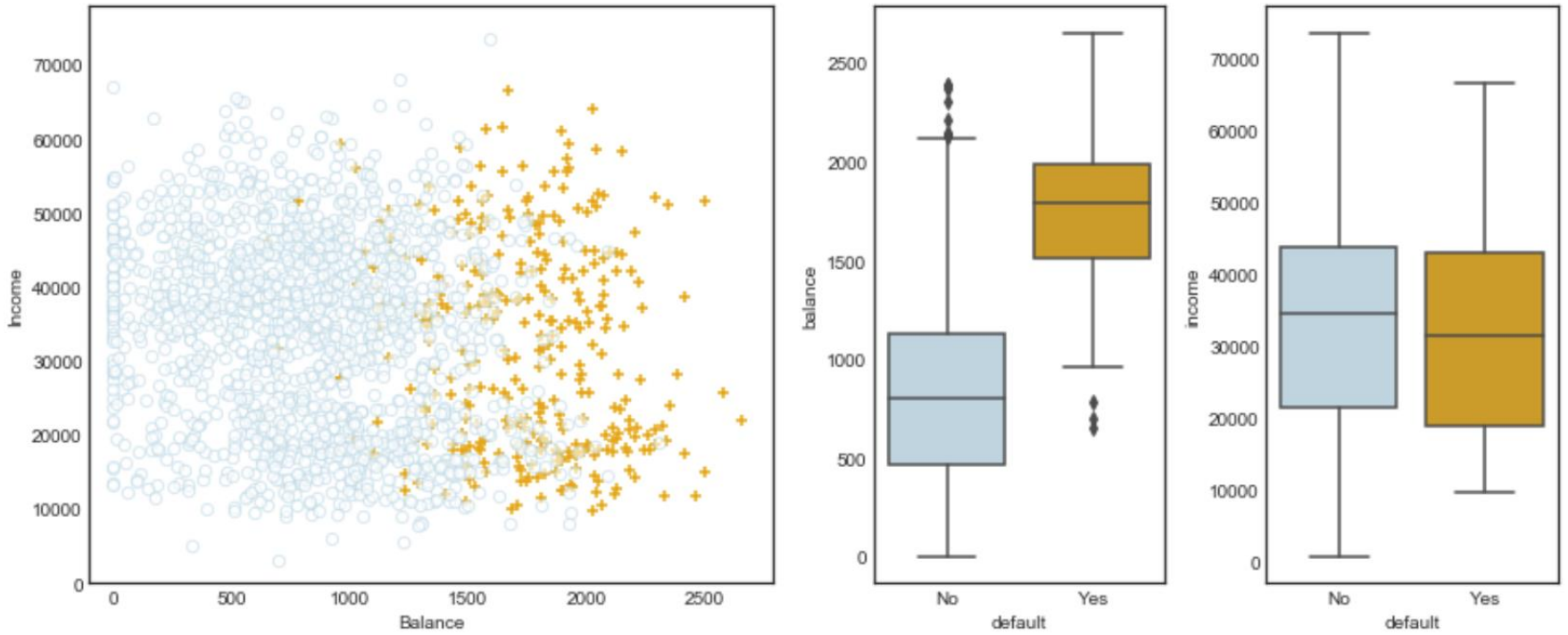


https://www.akkencloud.com/5-ways-handle-negative-comments-on-social-media/

Examples of classification problems

- Health Care:
  - A patient has a set of symptoms, those symptoms can be common in three different conditions, how can we find the right diagnosis?
- Banking (Fraud):
  - How do we determine whether an online transaction is fraudulent?
- Biology:
  - A Biologist wants to determine whether a certain type of mutation in the DNA can be decease-causing.

What's different from a Linear Regression?



Logistic vs. Linear Regression

Linear Regression

$$Expected\ Sales \sim B_0 + B_1(Budget)$$

We are no longer predicting numbers

Logistic Regression

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

Now we are predicting probabilities

Logistic vs. Linear Regression

Logistic Regression for Default dataset

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-10.651331	0.361169	-29.491287	3.723665e-191	-11.359208	-9.943453
balance	0.005499	0.000220	24.952404	2.010855e-137	0.005067	0.005931

A

B

Prediction whether a person has risk of default

C

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1{,}000}}{1+e^{-10.6513+0.0055\times 1{,}000}} = 0.00576,$$

D

A **Coefficients** cannot be interpreted as the ones in Linear Regression.

B **p-values:** we can still use the p-values to determine whether a variable has a important effect.

C Here are trying to predict the probability of default given certain conditions. In this case, it the persons balance.

D In this case, since P(X)<0.5, we are going to assume that the person has no risk of default.

How to predict using categorical variables

Logistic Regression for Default dataset

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-3.504128	0.070713	-49.554094	0.000000	-3.642723	-3.365532
student2	0.404887	0.115019	3.520177	0.000431	0.179454	0.630320

Prediction whether a person has risk of default

A B C D

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431,$$

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292.$$

A — In this example we are trying to predict the risk of default in two conditions.

B — In this case we are recoding the category student as 1 if the person is a student.

C — For the case when the person is not a student, we are assigning 0.

D — In this case, since P(X)<0.5, we are going to assume that the person has no risk of default.

Multiple Logistic Regression

Logistic Regression for Default dataset

	Coefficient	Std. error	Z-statistic	P-value
Intercept	−10.8690	0.4923	−22.08	<0.0001
balance	0.0057	0.0002	24.74	<0.0001
income	0.0030	0.0082	0.37	0.7115
student[Yes]	−0.6468	0.2362	−2.74	0.0062

TABLE 4.3. *For the* Default *data, estimated coefficients of the logistic regression model that predicts the probability of* default *using* balance, income, *and student status. Student status is encoded as a dummy variable* student[Yes], *with a value of* 1 *for a student and a value of* 0 *for a non-student. In fitting this model,* income *was measured in thousands of dollars.*

Prediction whether a person has risk of default

Case A: Person is student

$$\hat{p}(X) = \frac{e^{-10.869+0.00574\times1{,}500+0.003\times40-0.6468\times1}}{1+e^{-10.869+0.00574\times1{,}500+0.003\times40-0.6468\times1}} = 0.058.$$

Case B: Person is not student

$$\hat{p}(X) = \frac{e^{-10.869+0.00574\times1{,}500+0.003\times40-0.6468\times0}}{1+e^{-10.869+0.00574\times1{,}500+0.003\times40-0.6468\times0}} = 0.105.$$

$Balance = 1{,}500$

$Income = 40{,}000$

What happens when I want to predict more than two categories?

- In the real life we might find problems for which Logistic Regressions may not be a good fit. Some examples are:
  - We want to predict from ***more than two categories***.
  - A ***Linear approximation is not a good fit*** to our data.

- For those cases, we have other algorithms as options:
  - **KNN:** K-Nearest Neighbors
  - **LDA:** Linear Discriminant Analysis
  - **QDA**: Quadratic Discriminant Analysis
  - **Neural Networks**
  - **SVM:** Support Vector Machines.
  - **Tree-Based Methods:** Decision Trees, Random Forests, Boosted Machines.

In **Python**, the implementation of these algorithms is **very similar** to Logistic Regression, the main different with LR are the **packages/Libraries** used.

How to we evaluate the quality of the models?

Confusion Matrix for Default Data

		True default status		
		No	Yes	Total
Predicted default status	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

Accuracy = (9,644+81)/10,000 = 97.25%

Misclassification error rate: 2.75%

TABLE 4.4. *A confusion matrix compares the LDA predictions to the true default statuses for the* 10,000 *training observations in the* `Default` *data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for* 23 *individuals who did not default and for* 252 *individuals who did default.*

How to we evaluate the quality of the models?

Confusion Matrix for Default Data

		True default status		
		No	Yes	Total
Predicted default status	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

TABLE 4.4. *A confusion matrix compares the LDA predictions to the true default statuses for the 10,000 training observations in the* `Default` *data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for 23 individuals who did not default and for 252 individuals who did default.*

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Sometimes we are interested in having the details on the accuracy levels.

- Other metrics:
  - **TN**: True Negatives: ***99.76%***
  - **TP**: True Positives: ***24.32%*** Recall
  - **FN:** False Negatives: 252/333 = **75.68%**
  - **FP:** False Positives. 23/9667 = **0.34%**

As we can see in there, even though our global accuracy is 99.76%, our ability to identify default customers is very low.

How to we evaluate the quality of the models?

Confusion Matrix for Default Data

		True default status		
		No	Yes	Total
Predicted default status	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

TABLE 4.4. *A confusion matrix compares the LDA predictions to the true default statuses for the 10,000 training observations in the* `Default` *data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for 23 individuals who did not default and for 252 individuals who did default.*

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Recall: How many cases of the positive category the model was able to identify

$$Recall = \frac{TP}{Total\ \#\ of\ Positives\ in\ the\ data} = \frac{TP}{FN + TP}$$

Precision: among all the positive results, which one were actually positive. ***Ratio of positive in my predictions***.

$$Precision = \frac{TP}{Total\ \#\ of\ Predicted\ Positives} = \frac{TP}{FP + TP}$$

F1-Score: harmonic balance between precision and recall.

$$F1\ Score = 2 * \left(\frac{Precision * Recall}{Precision + Recall}\right)$$

Walk-through of concepts

Chapter 4 - Classification

In [3]:
```
# %load ../standard_import.txt
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

import sklearn.linear_model as skl_lm
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.metrics import confusion_matrix, classification_report, precision_score
from sklearn import preprocessing
from sklearn import neighbors

import statsmodels.api as sm
import statsmodels.formula.api as smf

%matplotlib inline
plt.style.use('seaborn-white')
```

Load dataset

In [4]:
```
# In R, I exported the dataset from package 'ISLR' to an Excel file
df = pd.read_excel('Data/Default.xlsx')

# Note: factorize() returns two objects: a label array and an array with the unique values.
# We are only interested in the first object.
df['default2'] = df.default.factorize()[0]
df['student2'] = df.student.factorize()[0]
df.head(3)
```

https://github.com/JWarmenhoven/ISLR-python

https://github.com/JWarmenhoven/ISLR-python/blob/master/Notebooks/Chapter%204.ipynb

http://faculty.marshall.usc.edu/gareth-james/ISL/

Lab

This lab on Linear Regression is a python adaptation of p. 109-119 of "Introduction to Statistical Learning with Applications in R" by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. Written by R. Jordan Crouser at Smith College for SDS293: Machine Learning (Spring 2016).

3.6.1 Importing Libraries

In [2]:
```
# Tells matplotlib to display images inline instead of a new window
%matplotlib inline

import numpy as np
import pandas as pd
import statsmodels.api as sm
```

https://github.com/jcrouser/islr-python

https://github.com/jcrouser/islr-python/blob/master/Lab%202%20-%20Linear%20Regression%20in%20Python.ipynb

http://faculty.marshall.usc.edu/gareth-james/ISL/