SOFTWARE IMAGING™

Whitepaper

Building Better UI Schneiderman's 8 Golden Rules

By James Brightman

8 Rules to bear in mind for your next UI project

Put simply, UI can make or break your product. Humans, for all our self-perceived intelligence, are in fact relatively shallow and simple creatures. We place a high value on things that look pleasing to the eye even if they perform below par. This is part of the reason Apple is a trillion-dollar company.

UI is the way in which humans and computers interface and as a developer, it's part of our job to streamline this interaction. While a product may offer amazing services and world-beating performance, if it is difficult to use or terrible to look at then people won't bother using it.

But with such a large array of material about UI design, it can be daunting to know where to even begin - Enter Bill Shneiderman and his 8 golden rules.

> **"People ignore design that ignores people."**
> **Frank Chimero**

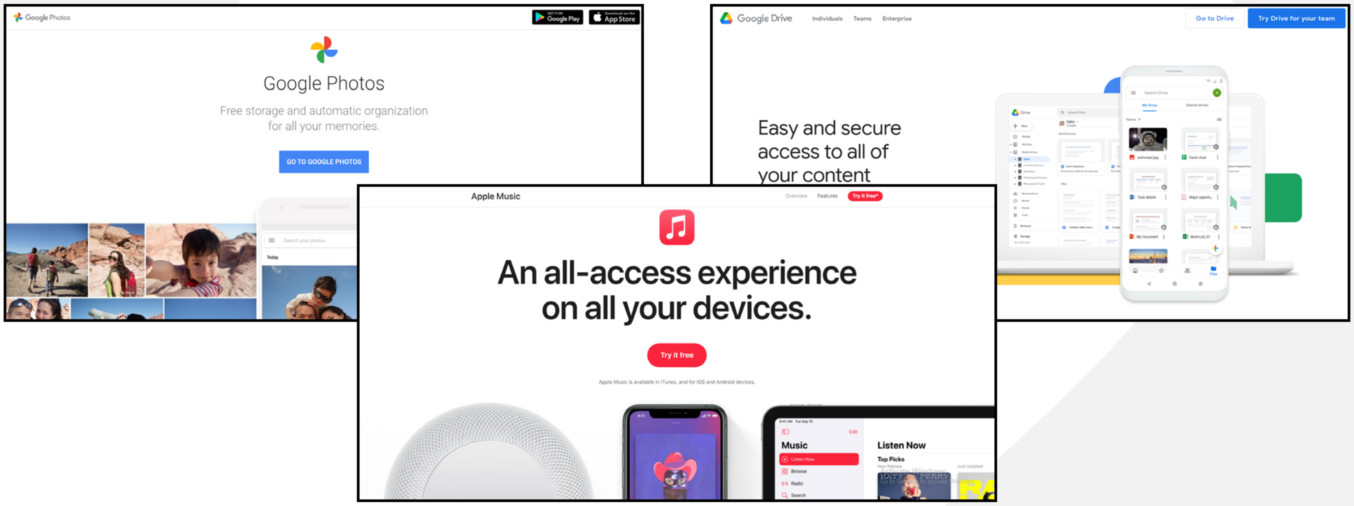
Schneiderman’s 8 Golden Rules

Schneiderman put forward a set of 8 principles that can be used to govern how designers and developers craft good interaction design in a system. These principles have the benefit of being extremely easy to understand with clear and practical applications. Let’s have a look at these rules and examine correct/incorrect applications of them in modern UI.

1. Strive for consistency

Have you ever noticed how all of Google’s product pages and logos look very similar? Or how clicking on the ‘X’ in a web browser closes the page? These are examples of consistent UI.

Striving for consistency is about allowing the user to follow an expected sequence of actions. To use the example of the ‘X’ button in a web browser — I expect that whenever I click this, the web page will be closed. If I click this and it occasionally does some other action, say maximize the page, this would be a severe break in the sequence of actions I expect of the UI.



Free storage and automatic organization for all your memories.

Easy and secure access to all of your content

An all-access experience on all your devices.

Apple Music is available in iTunes, and for iOS and Android devices.

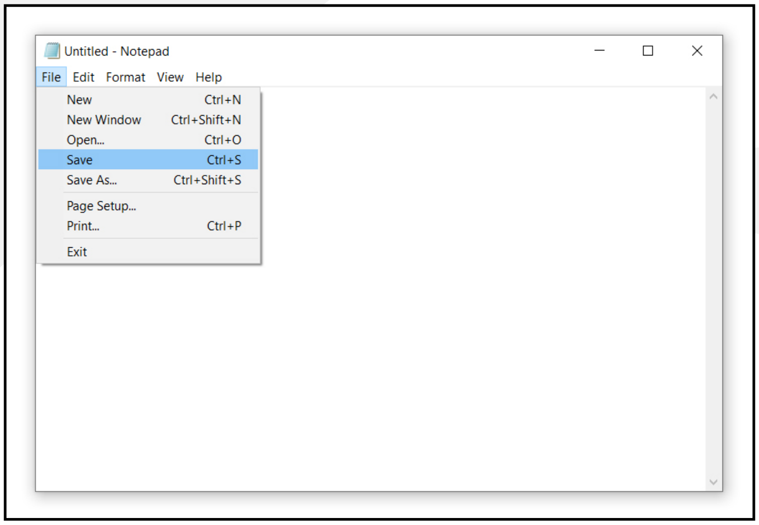
How do we know the 3rd UI isn’t from Google? Consistency! (that and an Apple logo…)

This is quite an obvious example of inconsistent UI behavior, this principle applies to more than just button actions. Error message format, hamburger menus, search bar positions, UI colours — all of these things need to remain consistent across the UI. This is what Google does brilliantly, which is why you know without looking at the URL that you are on a Google-branded page. Using UI libraries such as Material Design, or component-based UI design libraries such as React are a great way to strive for consistency.

The result of inconsistent UI is unlikely to cause the user to stop using the application immediately, but it has a cumulative negative effect on the users cognitive load.

2. Enable frequent users to use shortcuts

As a person gets more and more used to using an interface, their number of interactions increases, as should the pace of interaction. Essentially we want to be able to ‘reward’ users with a quicker interaction as they become more experienced with an interface. A great way to speed up interactions is with shortcuts. We use shortcuts to interact quickly with UI extremely frequently — when was the last time you manually saved a document instead of using Ctrl+S?

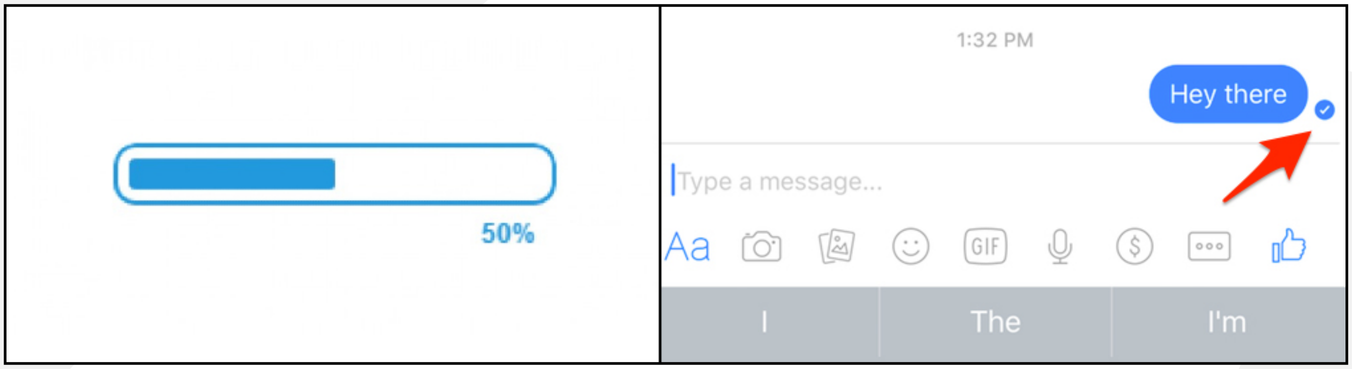


As we use the UI more, we learn more timesaving shortcuts

Shortcuts pertain not only to keyboard inputs, but also to allow users to access different areas of the system. For example, in a CRM system we may want to find information about a single newly added user. Instead of going through the entire system in search, suppose we have a ‘Recently Added Users’ button which allows us to go straight to the new user. The benefits of shortcuts to the user are vast as they allow for quicker and more efficient interaction with the application.

3. Offer informative feedback

When we perform an action, such as clicking a button or sending a message, we expect some feedback if that action is successful or not. The user needs to know if their task had been carried out. Informative feedback components normally take the form of loading icons, which tell us that our action has been received and is being processed. There are many different examples of informative feedback that we all see daily, such as read receipts on messenger apps, error popups on websites, and the loading bar when you have to run `npm install` for the 13th time that day (Ok maybe that last one is just me).



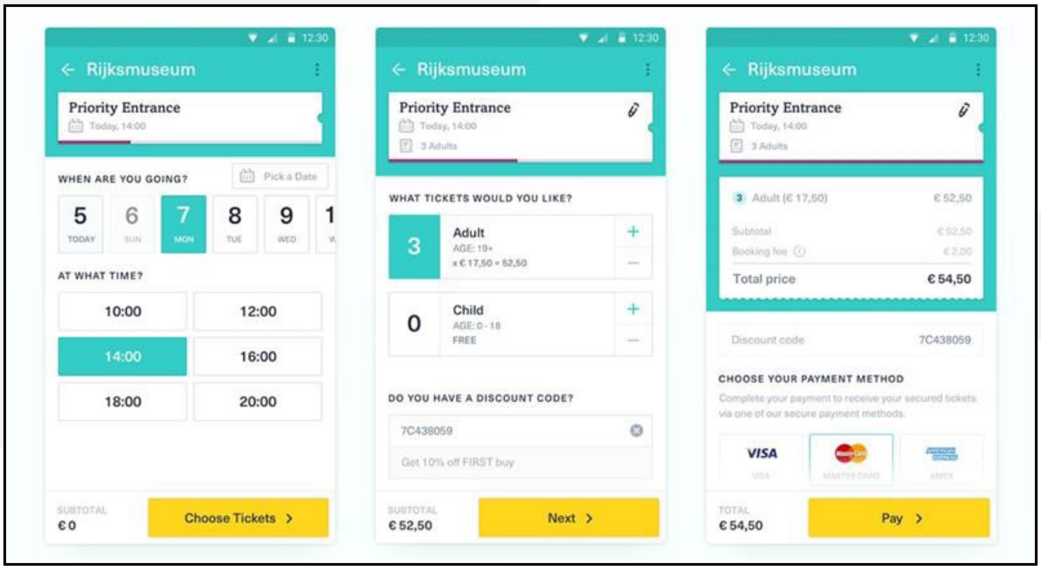
Hey there

Common examples of informative feedback

This isn’t to say that every single button should have a loading icon after you press it, or changing web pages needs to have a confirmation prompt. The important take away of this rule is to not leave the user in the dark whilst also not overloading them with excess information.

4. Design dialog to yield closure

If we can organize sequences of actions together with clear start and endpoints then we should strive to do so. It is beneficial for a user to feel like they are on a simple and straightforward path when trying to perform an action and is equally important that it is made clear when this action is finished. For example, let’s look at the sequence flow of buying an item online;



Complete your payment to receive your secured tickets via one of our secure payment methods.

Search for item > Add to basket > Checkout/Pay > Thank you message

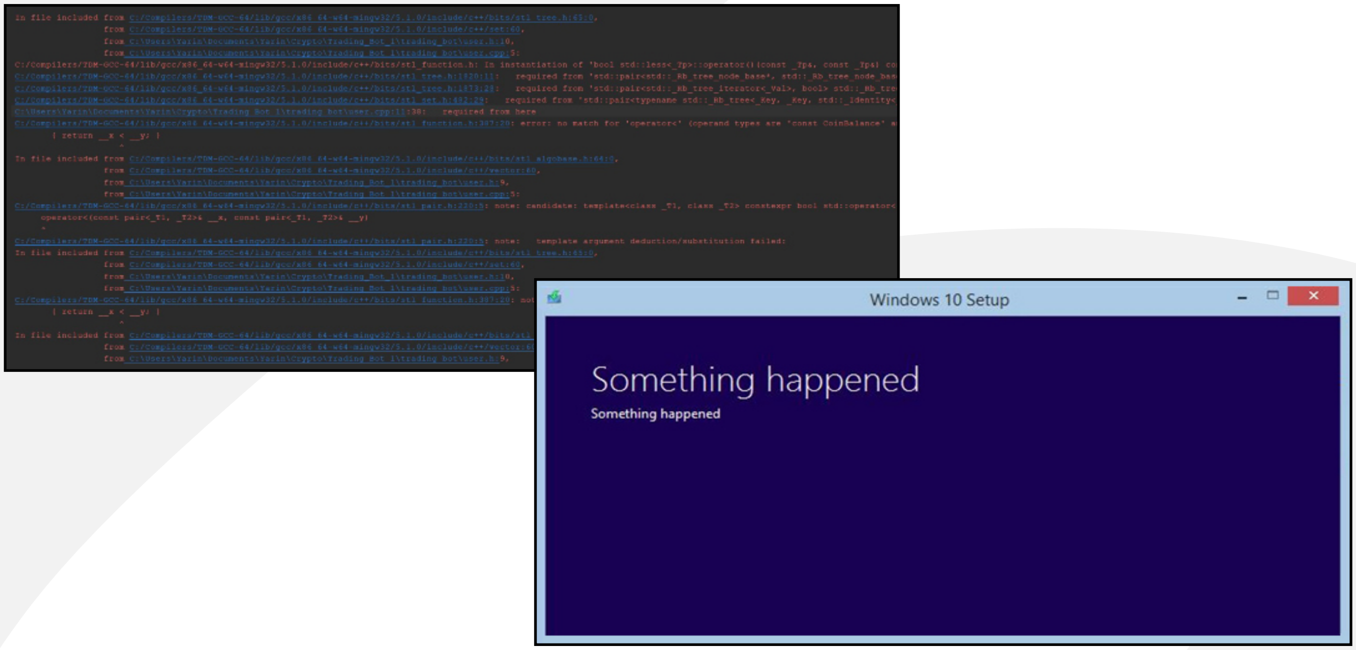
Good system design facilitates this logical flow of events and nudges the user into following this pattern. Try and swap any elements in this chain and it’s clear that it would not work.

Ending a sequence with some kind of informative feedback makes it clear to the user that the current sequence of actions has finished, there can be no confusion about if they have already paid for an item or not.

5. Offer simple error handling

Obviously, developers try and make sure that users do not run into errors, but errors are largely unavoidable. If we have to show the user an error then simple remember;

They’re users, not developers



```
In file included from C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_tree.h:65:0,
                 from C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/set:60,
                 from C:\Users\Yarin\Documents\Yarin\Crypto\Trading_Bot_1\trading_bot\user.h:10,
                 from C:\Users\Yarin\Documents\Yarin\Crypto\Trading_Bot_1\trading_bot\user.cpp:5:
C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_function.h: In instantiation of 'bool std::less<_Tp>::operator()(const _Tp&, const _Tp&) co
C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_tree.h:1820:11:   required from 'std::pair<std::_Rb_tree_node_base*, std::_Rb_tree_node_bas
C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_tree.h:1873:28:   required from 'std::pair<std::_Rb_tree_iterator<_Val>, bool> std::_Rb_tre
C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_set.h:482:29:   required from 'std::pair<typename std::_Rb_tree<_Key, _Key, std::_Identity<
C:\Users\Yarin\Documents\Yarin\Crypto\Trading_Bot_1\trading_bot\user.cpp:11:38:   required from here
C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_function.h:387:20: error: no match for 'operator<' (operand types are 'const CoinBalance' a
       { return __x < __y; }
                    ^

In file included from C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_algobase.h:64:0,
                 from C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/vector:60,
                 from C:\Users\Yarin\Documents\Yarin\Crypto\Trading_Bot_1\trading_bot\user.h:9,
                 from C:\Users\Yarin\Documents\Yarin\Crypto\Trading_Bot_1\trading_bot\user.cpp:5:
C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_pair.h:220:5: note: candidate: template<class _T1, class _T2> constexpr bool std::operator<
     operator<(const pair<_T1, _T2>& __x, const pair<_T1, _T2>& __y)
     ^

C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_pair.h:220:5: note:   template argument deduction/substitution failed:
In file included from C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_tree.h:65:0,
                 from C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/set:60,
                 from C:\Users\Yarin\Documents\Yarin\Crypto\Trading_Bot_1\trading_bot\user.h:10,
                 from C:\Users\Yarin\Documents\Yarin\Crypto\Trading_Bot_1\trading_bot\user.cpp:5:
C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl_function.h:387:20: not
       { return __x < __y; }
                    ^

In file included from C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/bits/stl
                 from C:/Compilers/TDM-GCC-64/lib/gcc/x86_64-w64-mingw32/5.1.0/include/c++/vector:6
                 from C:\Users\Yarin\Documents\Yarin\Crypto\Trading_Bot_1\trading_bot\user.h:9,
```

Something happened

Something happened

Try to strike a balance between these two extremes

While it may be interesting to us developers to know the HTTP return code of a failed API call, the vary majority of users have no need for this information. We should strive to display simple and pertinent error messages which not only detail the error but offer some steps to fix it.

Offering steps gives the user some power to try and find solution themselves instead of leaving them at a dead end, which your support team will likely thank you for.

6. Permit easy reversal of actions

Making mistakes is what makes us human. But undoing mistake is what makes us less anxious.

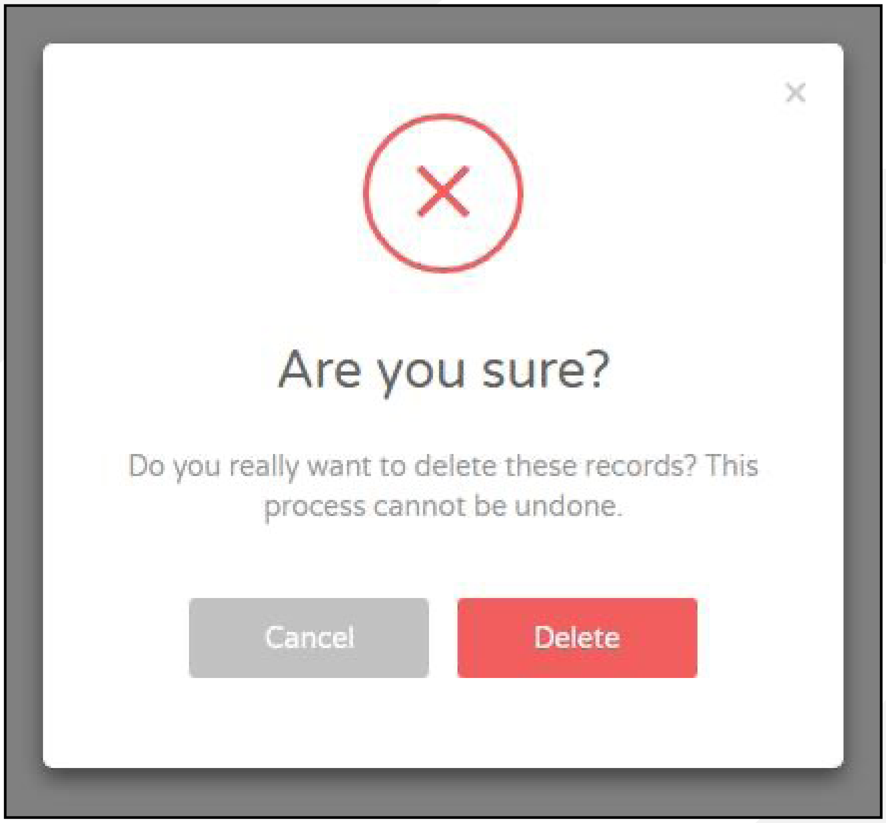
It’s important that the system can clearly show the user that mistakes aren’t the end of the world, and this is done through very common UI metaphors. We are all massively familiar with this golden rule in the form of the undo button (or if you follow rule 2, Ctrl + Z). This is a simple ‘whoops’ button that can undo our last action, no harm to foul.

We don’t always want to just go back in time - a redo button is equally important!

Schneiderman suggests that by actually allowing our users to make reversible mistakes, they gain a much better understanding of the system as they are not inhibited to explore and try new things, even if they cause errors. Therefore we can also use the lack of undoing UI metaphors to signify that a user needs to be more careful in this area of the system.

7. Support internal locus of control

What Schneiderman means by this is that the user has to be firmly in the driving seat whilst using the system. It should feel like all events are executed by the user and not being run silently in the background, as this undermines the users authority and trust in the actions of the system. We want to make sure that users are the initiators, not the responders.



Are you sure?

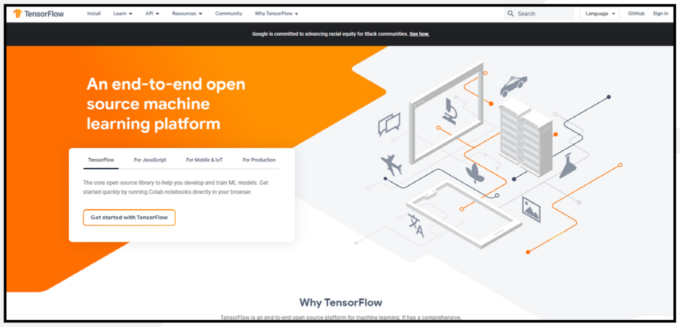
Do you really want to delete these records? This process cannot be undone.

Giving users the definitive decisions to make, such as permission dialogues, makes them feel like they have control over the system

8. Reduce short-term memory load

Unless you have a photographic memory, it is very unlikely that you can take in all the information immediately from a web page/app and recall it later. Schneiderman states that due to the limit of human short-term memory that displays be kept as simple as possible. We don't want to overload the user with all the possible features and attributes of the system immediately.

Tensorflow is a highly used Machine learning library with thounsands of functions and an almost unimaginable amount of complexity. Let's have a look at their homepage;



Google is committed to advancing racial equity for Black communities. See how.

An end-to-end open source machine learning platform

The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.

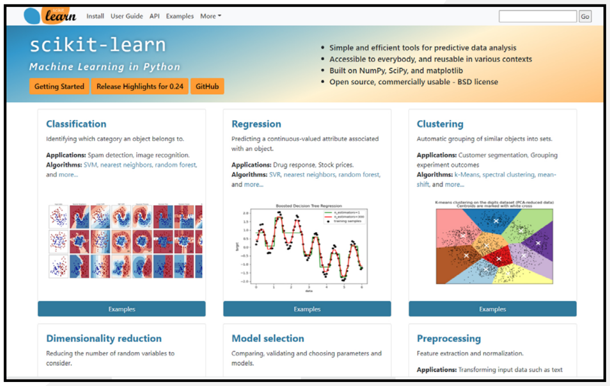
Why TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive

KISS — Keep It Simple Stupid!

The vast majority is space for graphics and simple text. It's a very clean and visually simplistic piece of UI design. Now let's look at a different ML library, Scikit-learn;

A much more busy and cluttered homepage


scikit learn
Install
User Guide
API
Examples
More
Go
Getting Started
Release Highlights for 0.24
GitHub
Boosted Decision Tree Regression
K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross
Examples
Examples
Examples

scikit-learn

Machine Learning in Python

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Dimensionality reduction

Reducing the number of random variables to consider.

Model selection

Comparing, validating and choosing parameters and models.

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text

It is staggeringly different to that of Tensorflow, with text everywhere, a myriad of colours and graphs. It's clear that the Scikit-learn UI places more load on short-term memory.

Just looking at each for 10 seconds, which one do you think would be easier to redraw from memory?

This isn't to say that Tensorflow > Scikit-learn with respect to what these frameworks are actually meant to do — however, if you shopping around for a new ML framework then the UI and the way they each present themselves plays a huge role in that decision.

In Conclusion

Schneiderman’s 8 golden rules aren’t the final word on UI and interface interaction design, but they are an extremely good set of rules to follow if you just want to get great informative, friendly and productive UI in place for a project.

In the land of confusing and half baked UI approaches, Schneiderman offers a full encompassing set of guidelines that essentially idiot-proof UI design (which is definitely something I look for).

Further reading

https://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html

https://www.cs.umd.edu/users/ben/goldenrules.html

About Software Imaging

Software Imaging is a software development company that creates innovative bespoke solutions and software products to solve businesses’ most challenging issues.

SOFTWARE IMAGING™

www.softwareimaging.com

Unipart House
Garsington Road, Oxford,
OX4 2PG, United Kingdom

Software Imaging Limited

Part of the Synapse Innovation group of companies