Optimizing web design using Machine Learning

- Vamsri Krishna S, Sathya Krishnan Suresh

Thoughts on the article: sat articles

Abstract:
Most of the websites' design that we use today are static and it is the same for every user. A considerable amount of analysis goes into the process of deciding a design but the design need not cover the website's entire user base. The design of a website might look good for some user while for another user it might be difficult to find an important information on the website because of it's design. We wanted to come up with a solution using Machine Learning to make the designs of websites dynamic so that design of a website is customized for a particular user and he/she will be able to use the website with ease.

Ideation:

To customize a website's design for each user, the website must make minor changes(eg-changing the font size, font style, position of important information, color, etc) for the first 'n' times the user logs in. The website monitors the time taken by the user to reach the information that he/she is looking for. Using the response time along with some other information that the user might have given while creating an account like age, etc the website should be able to

find a design from the already available designs that the user is comfortable with.

To implement the above idea, we designed a landing page of a website that you would expect a lot of people to visit-Hospital's website. Then we made changes to the color and positions of some information in the design to monitor the response time. Totally 25 different designs were used. You can take a look at the designs by clicking here.

Data Collection Process:

We sat down with the each of the data contributors and showed them the photos of the 25 websites designed one by one. Their goal was to reach the button called 'View Test Results'. The button was placed randomly and the color of the websites was also changed randomly. We calculated the time it took for each of them to reach the target. Other features like their gender, age, position, device used were also noted down.

Description of the data collected:

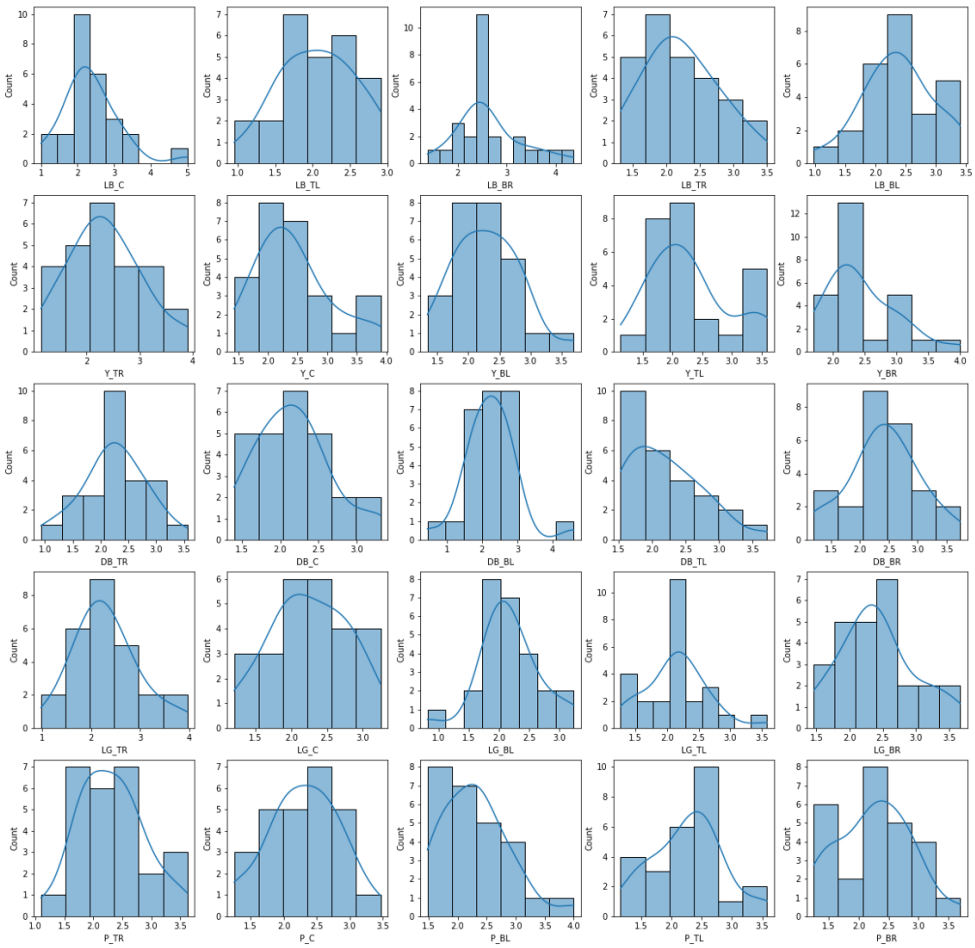
The data used here was collected from our friends and family members. Let's look at the data collected.

ID	AGE	DISTANCE	DEVICE	LB_C	LB_TL	LB_BR	LB_TR	LB_BL	Y_TR	Y_C	...	LG_BR	P_TR	P_C	P_BL	P_TL	P_BR	GLASSES	POSITION	GENDER	AVG_TIME
1	42	88	LP	2.39	1.50	3.90	3.21	2.23	2.45	2.98	...	2.42	2.45	2.40	2.80	2.91	1.61	NO	LAPTOP ON LAP	FEMALE	2.43
2	19	91	LP	2.35	2.07	2.46	1.62	2.27	2.07	1.43	...	1.46	2.32	2.50	2.03	2.06	2.06	YES	LAPTOP ON LAP	MALE	1.99
3	49	77	LP	2.31	2.86	3.21	2.63	3.12	2.64	2.24	...	2.57	2.75	3.47	2.40	2.51	2.79	NO	LAPTOP ON LAP	MALE	2.69
4	34	93	LP	2.30	0.95	2.35	1.37	0.98	1.73	1.81	...	2.05	1.69	1.49	1.77	1.59	1.67	NO	LAPTOP ON LAP	FEMALE	1.79
5	19	90	LP	3.37	2.29	2.79	2.59	2.37	3.06	2.53	...	2.98	1.87	2.91	2.28	2.41	2.63	NO	LAPTOP ON LAP	MALE	2.49
6	19	86	LP	1.90	1.90	2.42	1.32	1.98	1.81	1.95	...	1.95	2.00	2.01	1.74	1.48	1.45	NO	LAPTOP ON LAP	MALE	2.07
7	19	93	LP	1.48	2.16	2.09	1.66	1.88	2.04	1.77	...	2.35	1.66	1.99	2.29	2.49	2.89	NO	LAPTOP ON LAP	MALE	2.09
8	19	90	LP	3.32	2.35	3.27	2.61	3.18	2.55	2.71	...	3.24	3.62	2.62	2.94	2.04	2.47	NO	LAPTOP ON LAP	MALE	2.72
9	19	91	LP	1.98	2.42	2.55	2.15	2.05	2.31	1.90	...	2.44	2.39	2.51	1.48	2.60	2.61	NO	LAPTOP ON LAP	MALE	2.11
10	19	100	LP	1.98	1.66	2.44	2.27	2.80	2.11	2.06	...	2.39	2.49	2.83	2.43	2.47	2.55	NO	LAPTOP ON LAP	MALE	2.18

The dataset consists of 32 features. The first three features are standard features that you would always want to have when you design a website-Age, Distance and Device. The next 25 columns consist of time taken by the user to reach the target column for each design. The next four columns inform us whether the user wears spectacles, position of the device when the user was using the website, gender, and the average time of all readings.

Work done on the dataset:

Though the volume of the data collected is small we were still able to do extract some useful information from them. The distribution of response time is something that would give us a look into the level of comfort that each user has for a particular design. Take a look at the following plot.



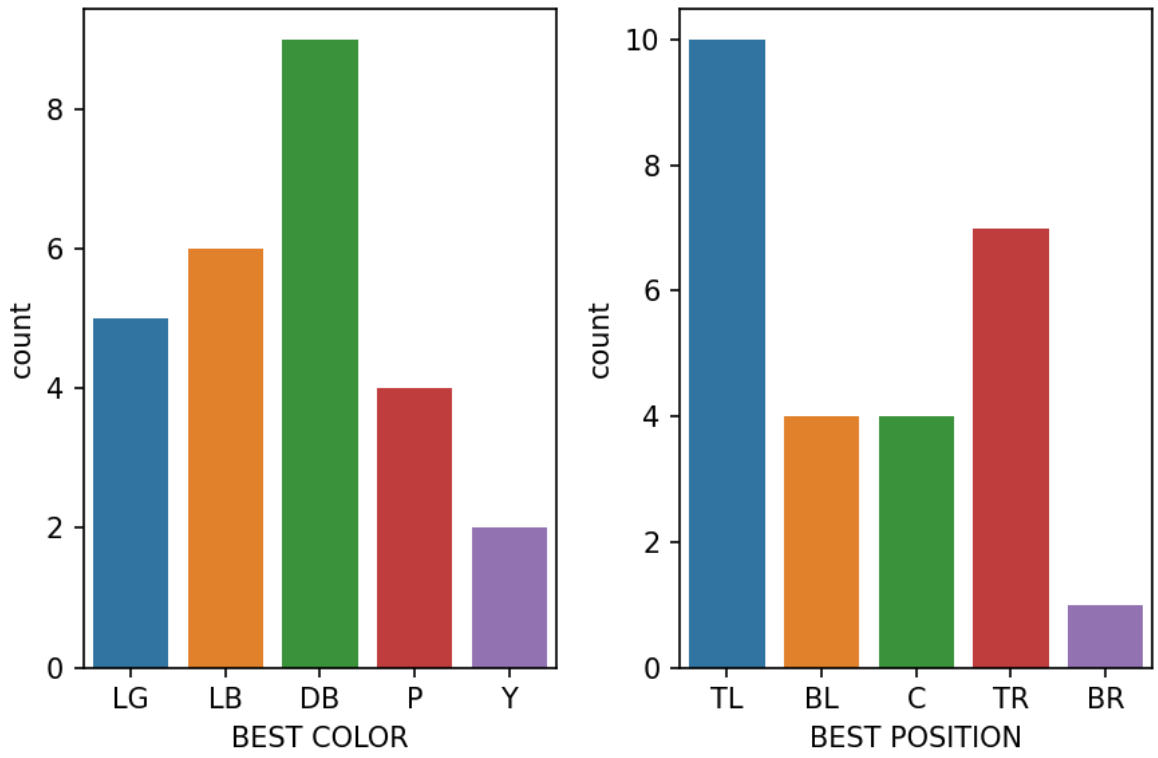
Distribution plot of response time

The above plot shows the distribution of response time for various designs. First let's analyze the variation of response time with respect to the colors used. For this, look at the plot row-wise. Most of them have normal distribution indicating that some designs have the same 'ease factor' or 'comfort factor' independent of the other features. One interesting observation here is the response time for the color *pink*. The distribution is not normalized and it is all over the place. Even though the other colors have one or two plots in

which the distribution is not normalized but it is more evident for ‘pink’ color.

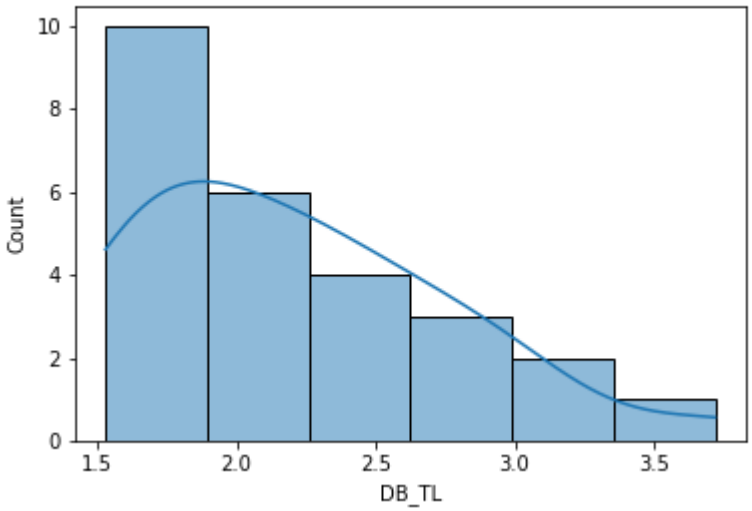
Now let’s analyze the plot with respect to the positions used. For this, look at the plot column-wise. The distribution plots of the response time when the button is in the **center** and in the **top-right** position is distributed normally and almost everyone seems to be able to identify the button quickly. But when the plot for the remaining positions are distributed randomly. This is very important for this study because it aims on each individual user’s preferred design.

By taking the average of response time for each color and position for each user we found their preferred color and position. The following plot shows the count for each color and position.



Count plot

From the above plot, we can infer that ‘dark blue’ is the color that most users prefer. The low value for ‘pink’ color is in line with the inference that we had obtained from the previous plot. We can confirm this by taking a look at distribution plot of ‘DB_TL’. The plot is given below and you can clearly see that the distribution is skewed left indicating that for this combination people were able to reach the target location sooner. Some of them even told us that they **liked dark blue** color.



Distribution of response time for ‘DB_TL’ design.

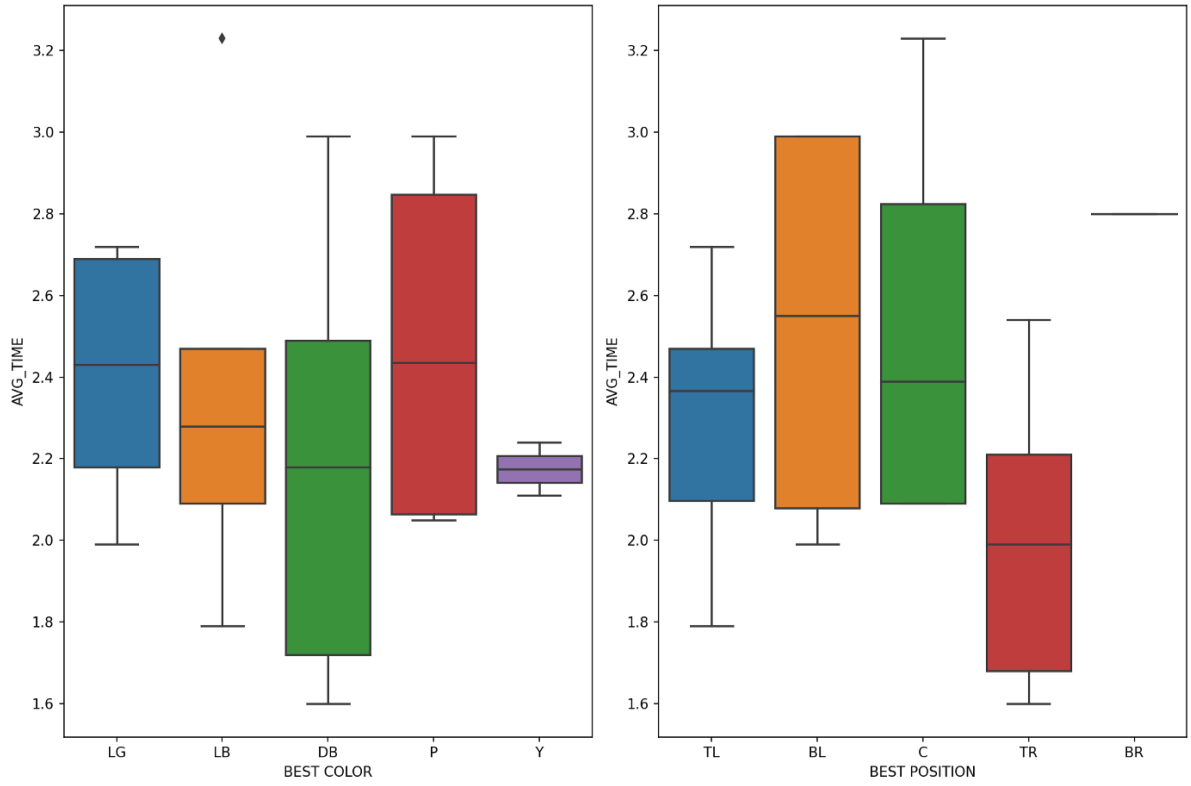
After looking at the information obtained we wanted to know the count of samples that preferred these two in combination but we found only two and their features are very similar. Both of them are male, do not wear glasses and are of the same age. This is interesting because this once again tells us that each user needs a design that is customized for them alone.

```
df[(df['BEST COLOR']=='DB')&(df['BEST POSITION']=='TL')]
```
✓ 0.1s

	AGE	DISTANCE	DEVICE	LB_C	LB_TL	LB_BR	LB_TR	LB_BL	Y_TR	Y_C	...	P_C	P_BL	P_TL	P_BR	GLASSES	POSITION	GENDER	AVG_TIME	BEST COLOR	BEST POSITION
4	19.0	90.0	LP	3.37	2.29	2.79	2.59	2.37	3.06	2.53	...	2.91	2.28	2.41	2.63	NO	LAPTOP ON LAP	MALE	2.49	DB	TL
9	19.0	100.0	LP	1.98	1.66	2.44	2.27	2.80	2.11	2.06	...	2.83	2.43	2.47	2.55	NO	LAPTOP ON LAP	MALE	2.18	DB	TL

2 rows × 34 columns

The inferences obtained so far have been from the combination of color and position, so let's take a look at them separately. The first plot is in line with the inferences we had been accumulating but in the second plot, the response time for *'TR'(top-right) is less than that of 'TL'(top-left)* and 'BR'(bottom-right) is not even in the picture. This suggests that the response time for 'TL' is small only when it is used with 'DB'(dark blue). You can draw a lot of inferences from this plot but we leave that to the reader.



Difficulty in developing a good model:

We have used [scikit-learn library] here for using the various classification models available. There are two target columns here- color and position. Basic classification models like Logistic Regression, SVC, RandomForest, DecisionTree, KNeighborsClassifier. The accuracy of various models are shown below.

```
from sklearn.metrics import accuracy_score
models = [LogisticRegression(),SVC(),KNeighborsClassifier(),DecisionTreeClassifier(),RandomForestClassifier()]
def validate_models(models,X_train,y_train,X_test,y_test):
    ind = []
    scores = []
    for model in models:
        model = model.fit(X_train,y_train)
        preds = model.predict(X_test)
        acc_score = accuracy_score(y_test,preds)
        scores.append([acc_score])
        ind.append(str(model))
    vdf = pd.DataFrame(index=ind,columns=['SCORE1'],data=scores)
    return vdf
print(validate_models(models=models,X_train=X_train_sc,y_train=y_train,X_test=X_test_sc,y_test=y_test))
```

✓ 0.2s

	SCORE1
LogisticRegression()	0.333333
SVC()	0.333333
KNeighborsClassifier()	0.000000
DecisionTreeClassifier()	0.000000
RandomForestClassifier()	0.333333

The accuracy score of every model is legit bad. But there is no need to be alarmed by that fact. This is an expected result. There are two reasons for that. One is that the volume of data collected is very small. The second reason is you cannot expect a simple model with no hyperparameter tuning to understand the complex pattern of this small dataset. The response times vary by 0.01 seconds in most of the cases and in some samples they have the same value but different preferred color and design. To get good accuracy scores there have to be a minimum of a million samples and one certainly has to use neural networks probably with two or more activation layers to understand the complexity of the underlying pattern. You can take a look at the full code here on Feb 15.

Solution Model:

After reading the above section you might think that it would be very difficult to come up with a solution and yes, performing micro-segmentation to customize the design for each user is a tough job but we have come up with one idea that might be feasible for groups

of users. First, we try to cluster the test samples available. The reason for this is to separate the huge dataset into separate similar segments which will be easier to analyse and test out the different models on them. Once we have gained enough insights about each segment. we will try to pair the segments like we do in Agglomerative clustering. And then we try out the different models that we had already developed for the small segments that we had previously analysed. If the models perform well on the combined segments, then we can combine two two-segments and continue our testing. If it does not perform well, we back propagate. This is somewhat similar to the 'feed-forward' and 'back-propagation' concept in neural networks and that's why we had recommended a neural network model earlier.

Future of this idea:

We searched extensively for a website that had already implemented this idea, but we could find none. We only found a couple of small articles. There are many companies that use AI and ML to develop new colours and fonts. This idea can only be implemented by websites that have millions of users like Instagram, Twitter. Because they have the manpower to collect the data and analyse them thoroughly. Once the color and positions have been automated, then the font-style, font-size and all other things can be automated. Customizing the website for individual users will increase their amount of usage of the website and you can expect a lot of new users.

Conclusion:

In this article we talked big and implemented a little but we will try

to come up with more analysis in the future as the size of the dataset we collect grows. We hope you had as much fun as we had while working on this article. If you found this article useful clap, leave a comment and subscribe.

Review by Dr.K.Indira , Director, ES Group of Institutions, Villupuram:
Optimizing web design using Machine Learning — a very innovative work looking at the prospective of dynamic web design. The dataset features chosen are opt and close to the work chosen. Though the accuracy score is not to the mark as indicated by the authors, improvement can be achieved by increasing dataset size and fine analysis between the parameters chosen. The term 'n' times for the users to login needs more clarity and scope for future betterment is very extensive.