Website Redesign

February 1, 2022

1 Which version of the website should you use?

by Scott Schmidt

1.1 Background

An early-stage startup in Germany has been working on a redesign of the landing page. The team believes a new design will increase the number of people who click through and join your site. They have been testing the changes for a few weeks and now they want to measure the impact of the change and need you to determine if the increase can be due to random chance or if it is statistically significant.

1.2 The data

The team assembled the following file:

Redesign test data

- "treatment" - "yes" if the user saw the new version of the landing page, no otherwise.
- "new_images" - "yes" if the page used a new set of images, no otherwise.
- "converted" - 1 if the user joined the site, 0 otherwise.

The control group is those users with "no" in both columns: the old version with the old set of images.

1.3 Report

1. Analyze the conversion rates for each of the four groups: the new/old design of the landing page and the new/old pictures.
2. Can the increases observed be explained by randomness? Are the results statistical significance?
3. Which version of the website should they use?

1.4 Summary

Using Area Under the Curve (AUC) as a result metric: * The logistic regression is 0.5059. * Decision Tree AUC is 0.5071

In short, there does not seem to be a correlation that the new website design and images has improved in customers joining the website. A 50% AUC is essentially equilivalent to a coin flip, and the data appears to be random. There is definitely not a strong statistical significance that is above.

[24]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, confusion_matrix, roc_auc_score

#CLEANING THE DATA:
df = pd.read_csv('./data/redesign.csv')
df = df.replace(to_replace = ['yes','no'],value = [1,0])

#FIND MISSiNG VALUES:
def findNA():
    findNA=df.isnull().sum().sort_values(ascending=False)
    print("Missing values per column: ")
    print(findNA)

df = df.dropna() #there is no missing data

#SPLIT AND TEST DATA:
y_data = df['converted']
x_data = df.drop('converted', axis = 1)

from sklearn.model_selection import train_test_split
x_training_data, x_test_data, y_training_data, y_test_data =␣
 ↪train_test_split(x_data, y_data, test_size = 0.3, random_state=42)
```

1.5 Correlations and Heatmap

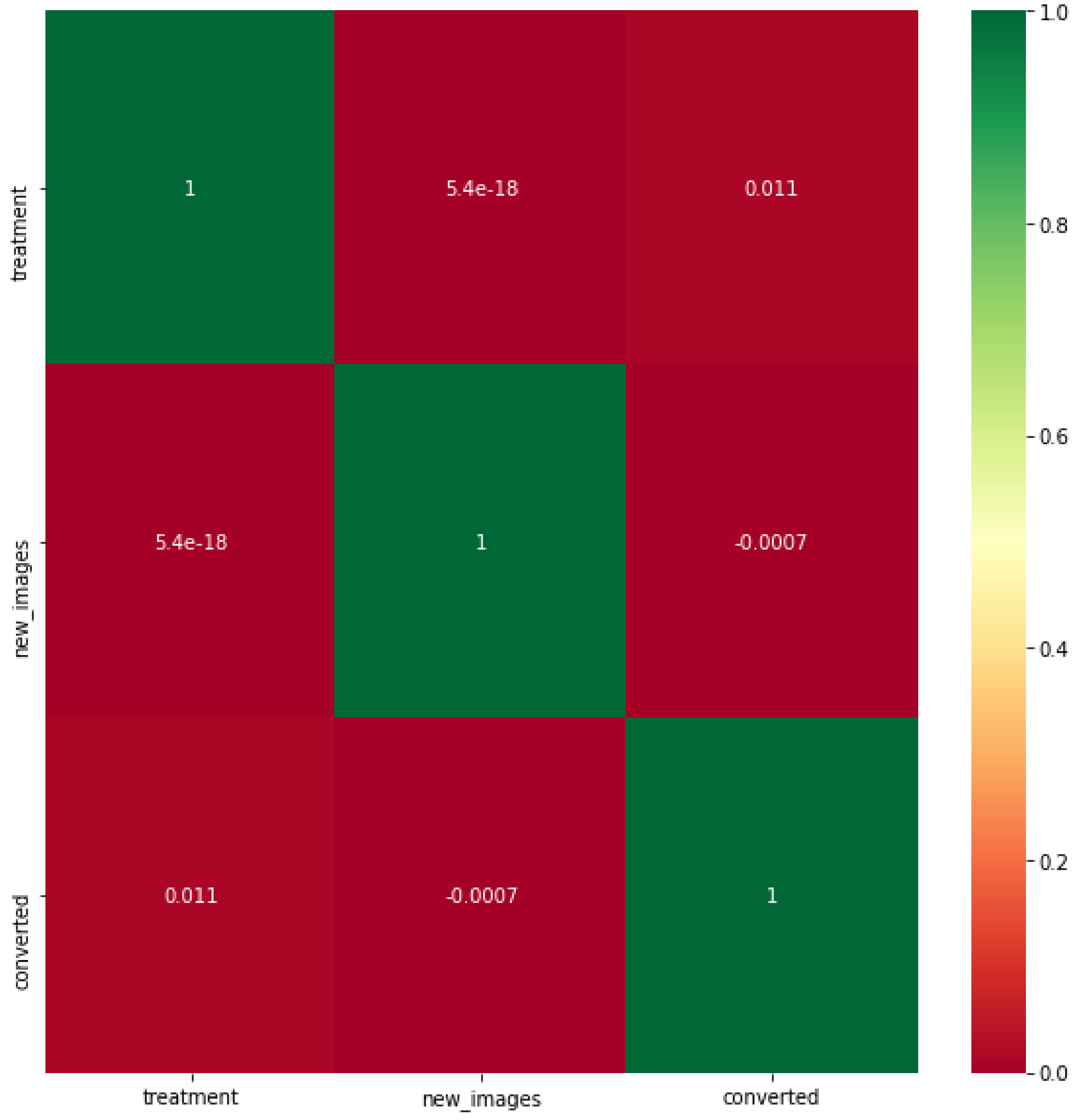
There appears to be no correlation between the new website and images.

[9]:
```
#FIND INDIVIDUAL CORRELATIONS:
treatmentCorr=df['treatment'].corr(df['converted'])
new_imagesCorr=(df['new_images'].corr(df['converted']))
print(round(treatmentCorr, 4) , " is the treatment correlation.")
print(round(new_imagesCorr, 4) , " is the new images correlation.")

#Find Multicollinearity using heat map:
plt.figure(figsize=(10,10))
heat = sns.heatmap(df.corr(),annot=True,cmap="RdYlGn")
print(heat)
```

```
0.0111  is the treatment correlation.
-0.0007  is the new images correlation.
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```



1.6 Logistic Regression

[16]:
```
from sklearn.linear_model import LogisticRegression # model algorithm
lr = LogisticRegression() #Logistic regression defaults to L2

#Train the model and create predictions
lr.fit(x_training_data, y_training_data)
predictions = lr.predict(x_test_data)

#use model to predict probability that given y value is 1:
```

```
y_pred_proba = lr.predict_proba(x_test_data)[::,1]

auc = round( metrics.roc_auc_score(y_test_data, y_pred_proba), 4)
print("Logistic model AUC is: ", auc)  # AUC is:  0.5059
```

```
Logistic model AUC is:  0.5059
```

Using logistic regression, the AUC was 0.5059. This does not show correlation between the new website. There is definitely not a strong signiciant correlation, above 95%.

1.7 Decision Tree

[25]:
```
from sklearn.tree import DecisionTreeRegressor

from sklearn.model_selection import train_test_split
x_training_data, x_test_data, y_training_data, y_test_data =␣
 ↪train_test_split(x_data, y_data, test_size = 0.3, random_state=42)

#FITTING DECISION TREE FOR BEST MAX_DEPTH:
train_error = {}
for i in range(2, 50, 5):
    model_1 = DecisionTreeRegressor(max_leaf_nodes=i)
    model_1.fit(x_training_data, y_training_data)
    #error=mean_absolute_error(y_test_data, y_pred_proba)
    error=mean_absolute_error(y_test_data, model_1.predict(x_test_data)).
 ↪round(4)
    train_error[i]=error

#MAKE PREDICTION:
min_error=min(train_error, key=train_error.get) #Best value for least error:
tree=DecisionTreeRegressor(max_leaf_nodes=min_error)
tree.fit(x_training_data, y_training_data)

#use model to predict probability that given y value is 1:
y_pred_proba = tree.predict(x_test_data)

auc = round( metrics.roc_auc_score(y_test_data, y_pred_proba), 4 )
print("Decision Tree AUC is: ", auc)

def printReports():
    print(classification_report(y_test_data, y_pred_proba))
    print(confusion_matrix(y_test_data, y_pred_proba))
    metrics.mean_absolute_error(y_test_data, y_pred_proba)
    metrics.mean_squared_error(y_test_data, y_pred_proba)
    np.sqrt(metrics.mean_squared_error(y_test_data, y_pred_proba))
```

```
Decision Tree AUC is:  0.5071
```