Click-Through Prediction

Background

First: Interested in looking at why someone clicks on an ad
These are features of the person clicking

Second: Interested in why is an ad clicked on?
These are features of ad clicked on

What’s the data?

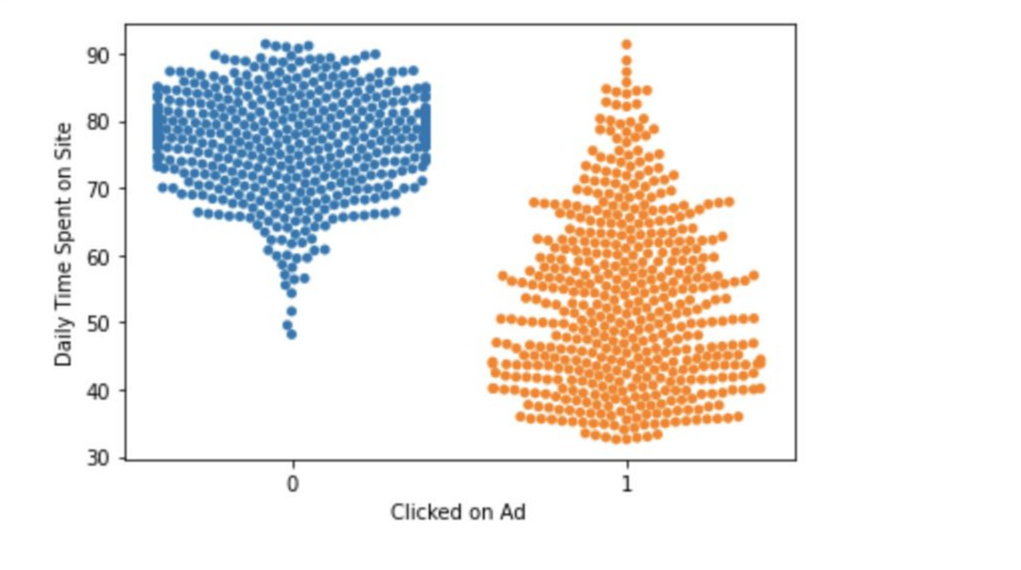
```
dtype='object')
```

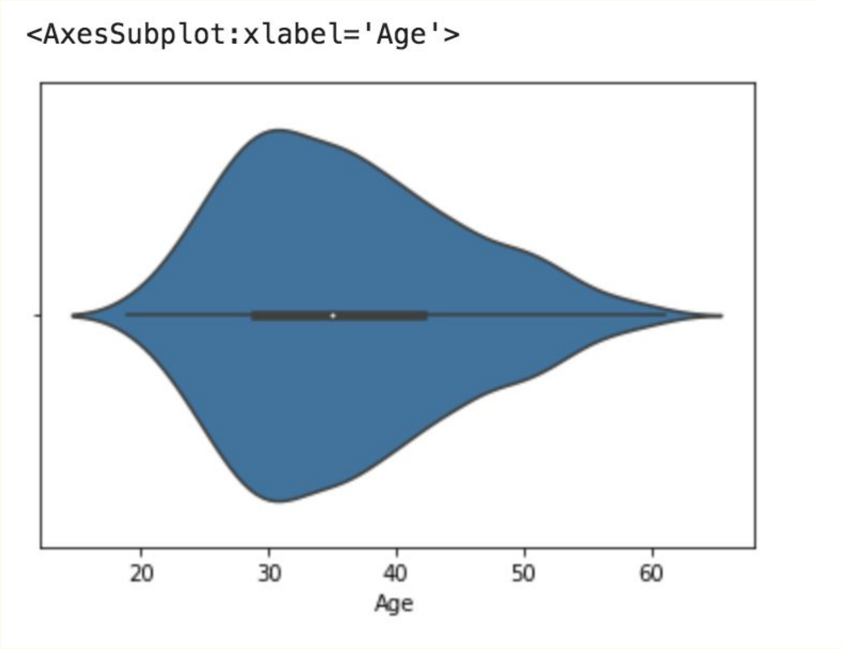
```
[4]: ad_data.head()
```

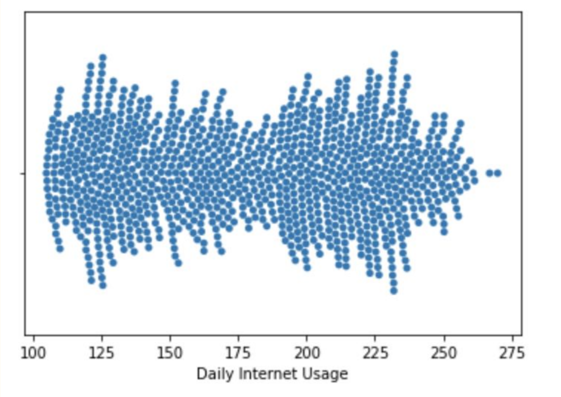
[4]:

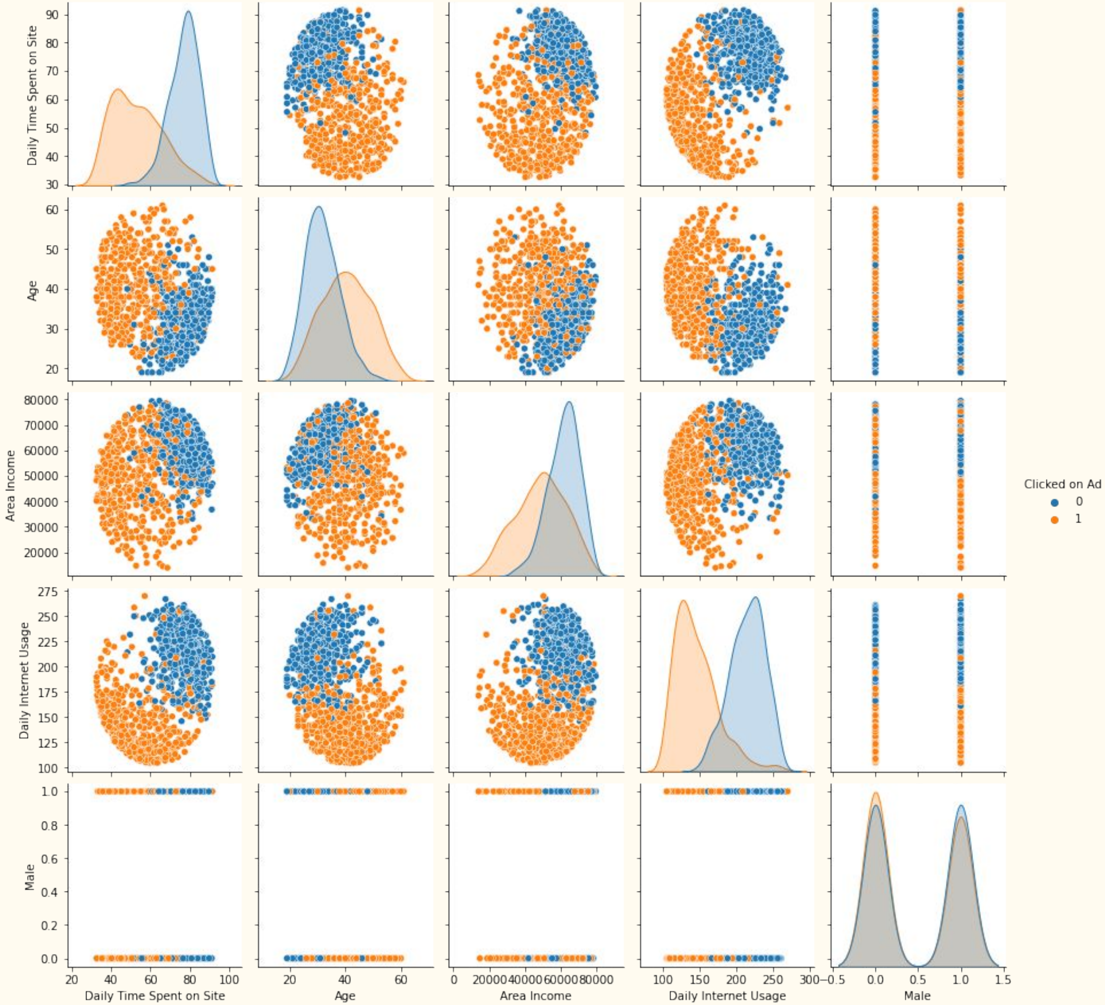
	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	0
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	0
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	0
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	0
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	0

EDA


Daily Time Spent on Site
90
80
70
60
50
40
30
0
1
Clicked on Ad

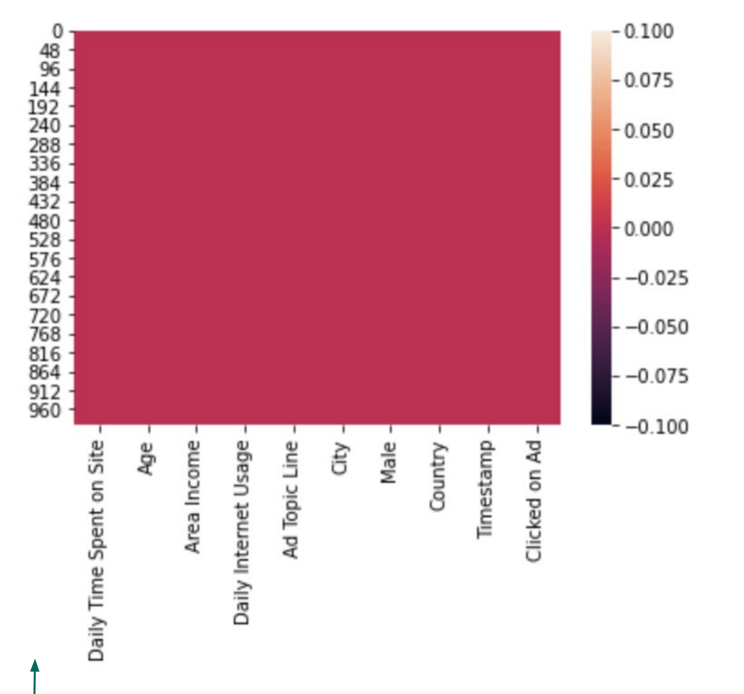

<AxesSubplot:xlabel='Age'>
20
30
40
50
60
Age

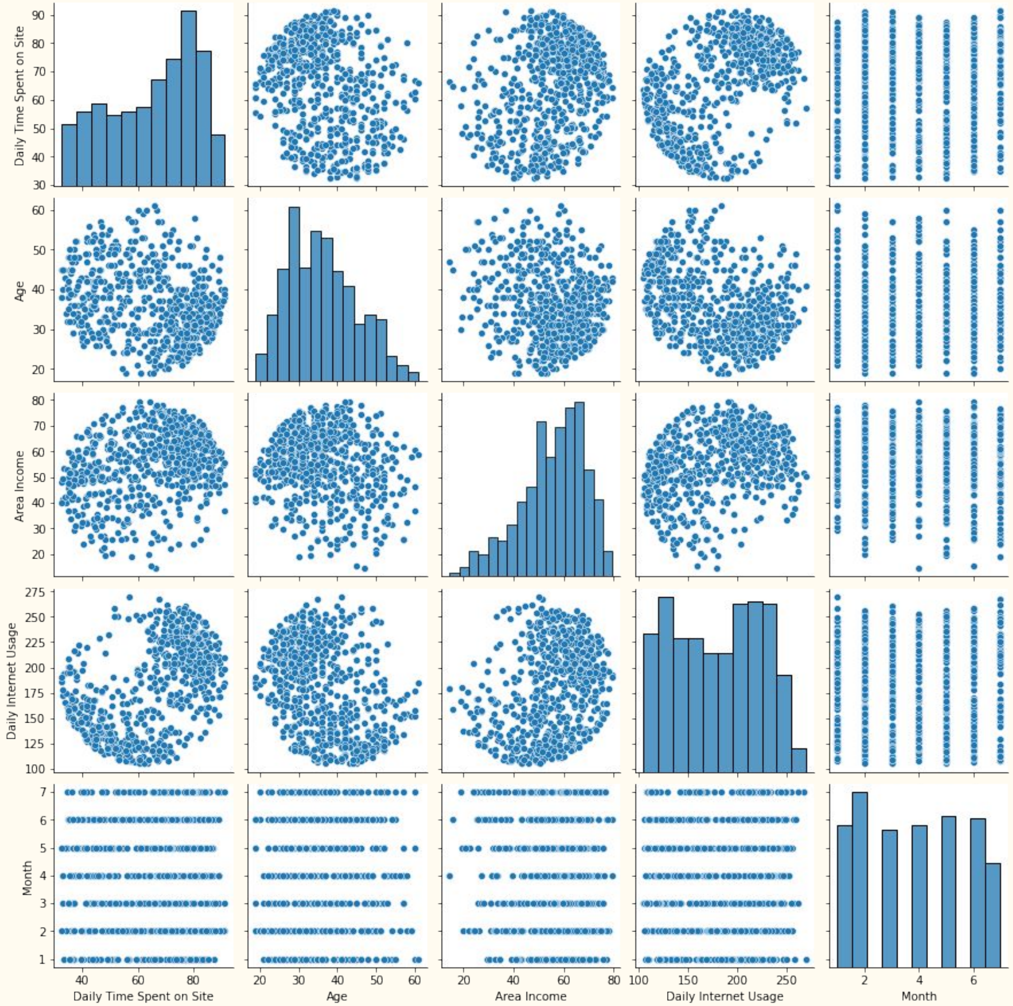

100
125
150
175
200
225
250
275
Daily Internet Usage



Doesn't look like a log regression would be as accurately predictive

Data seems to be fairly Gaussian, transformation isn't necessary





Determining if there are NAs

Looking at the training set features correlations

Feature Engineering

```
countries = x_train['Country'].to_list()
country_list = []
for name in countries:
    try:
        country = pc.countries.search_fuzzy(name)
        country_code = country[0].alpha_2
        country_list.append(country_code)
    except LookupError :
        country_list.append('None')

# column = ad_data['Country']
# get_continent(column)
```

```
1]: #print(country_list)
type(country_list)
Code_array= np.array(country_list)
x_train['Country Code'] = Code_array
```

```
2]: continent_list = []
for code in country_list:
    try:
        cont_code = pcc.country_alpha2_to_continent_code(code)
        continent_list.append(cont_code)
    except LookupError:
        continent_list.append('None')
```

```
3]: Continent_array = np.array(continent_list)
x_train['Continent Code'] = Continent_array
```

```
Age = x_train['Age']
x_train['Age Bins'] = pd.cut(Age, bins = [18, 30, 36, 43, 62], labels = ['youngest', 'younger-mid', 'older-mid', 'oldest'])
```

```
Income = x_train['Area Income']
x_train['Income Bins'] = pd.cut(Income, bins = [13, 47, 57, 65, 80], labels = ['low', 'low-mid', 'high-mid', 'high'])
```

Initial drops

	Sex	Time of Day	Month	Income Bins	Age Bins	Time_Online_Bins	Time_On_Site_Bins
212	Male	night	5	low-mid	youngest	most-time	more-time
692	Male	night	4	low	oldest	middle-time	middle-time

Maybe correlation from Time Online and Time on Site....

	Daily Time Spent on Site	Daily Internet Usage	Sex	Time of Day	Month	Income Bins	Age Bins
212	76.87	235.35	Male	night	5	low-mid	youngest
692	66.26	179.04	Male	night	4	low	oldest

Decision Tree

```
print("DTCLF Accuracy is:", dtclf.score(test_features_oc_ohe, test_labels))
#this is just to see if the score is the same between different methods
```

```
DTCLF Accuracy is: 0.9272727272727272
```

```
print("Depth of tree:", dtclf.get_depth())
print("Number of leaves:", dtclf.get_n_leaves())
print("Number of nodes:", dtclf.tree_.node_count)
#I can do better than this
```

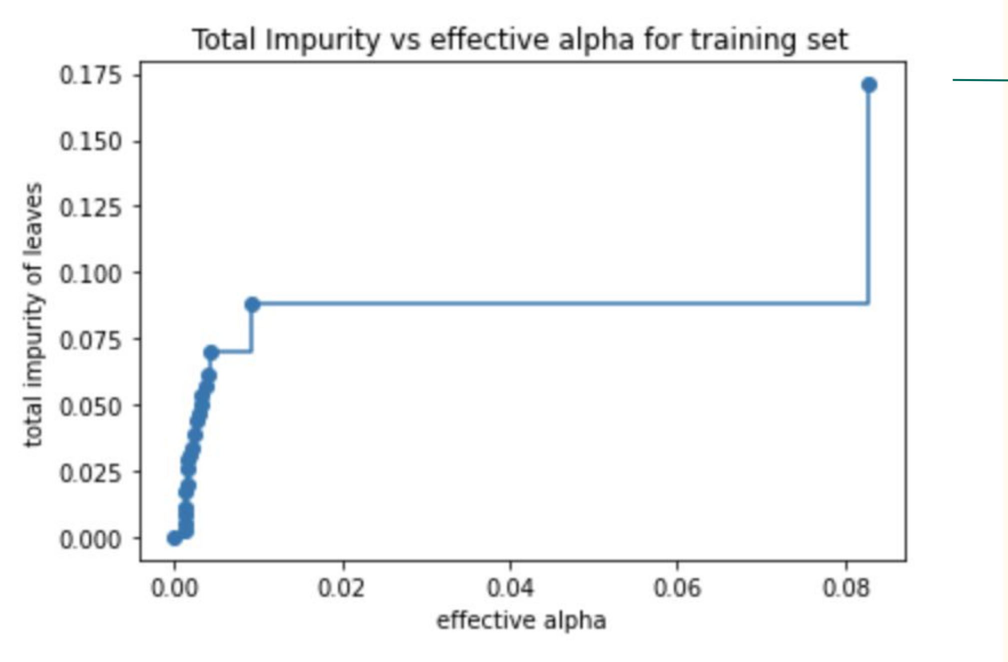
```
Depth of tree: 10
Number of leaves: 38
Number of nodes: 75
```

Pruning!

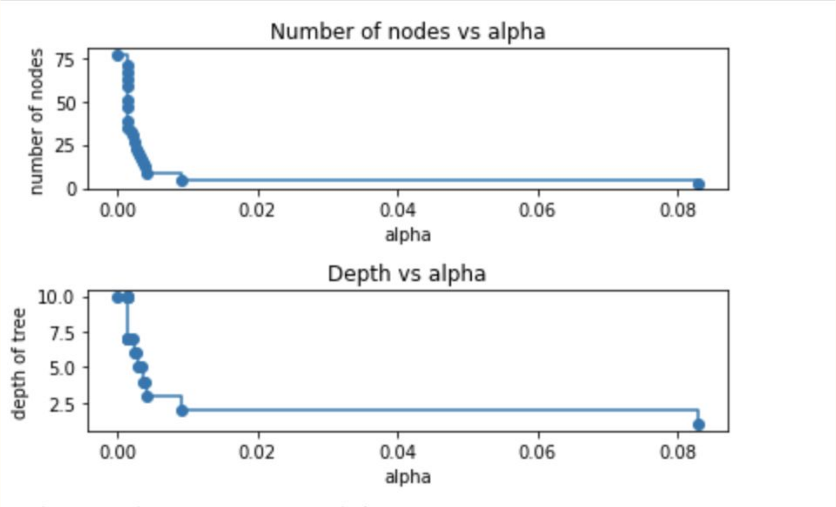
Chose to use minimal cost complexity to prune

Uses the "weakest link" method. This method takes the alpha values and determines effectiveness, then removes the node with the least effective alpha first. As alpha goes up, more of the tree is pruned so you get less nodes and more impurity.

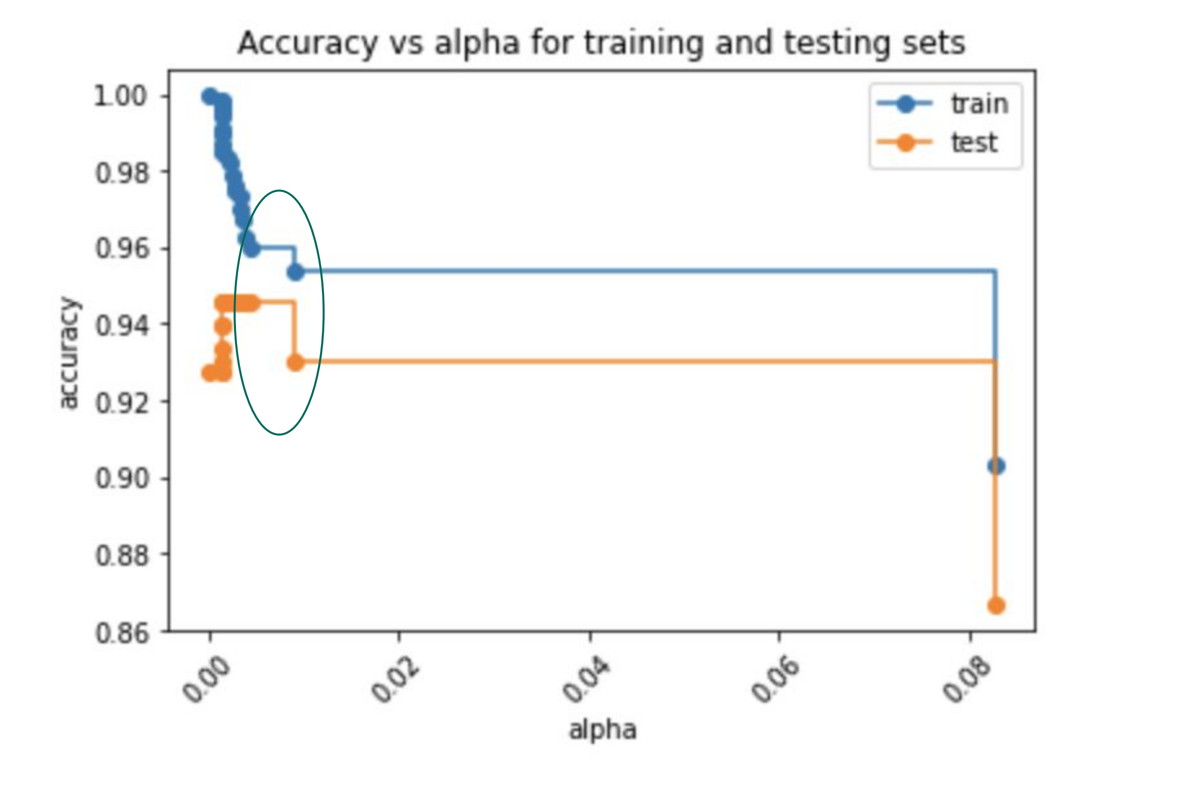
Got rid of the last alpha since that's the tree with only one node



After dropping the last alpha I retrained...



Select alpha by comparing train and test accuracy scores



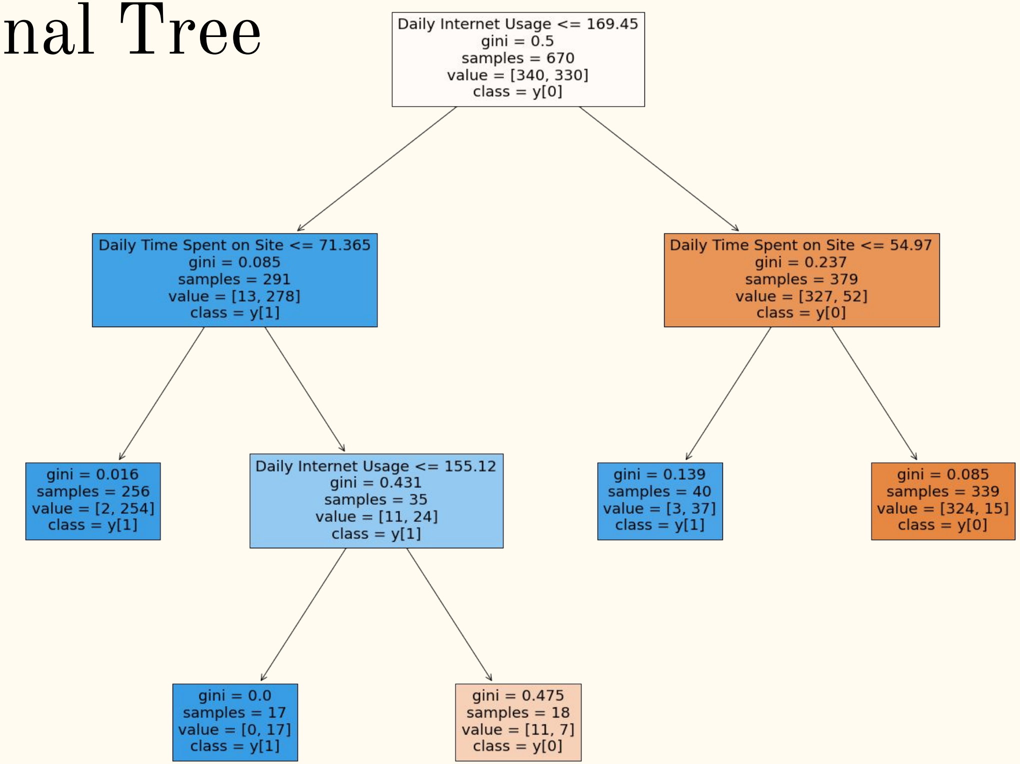
Want to find the alpha where the training set model accuracy is lowest and still maintains a high testing set model accuracy

Feature Importance

	Feature	Importance
1	Daily Internet Usage	0.787311
0	Daily Time Spent on Site	0.212689
9	Income Bins_low	0.000000
15	Age Bins_older-mid	0.000000
14	Age Bins_younger-mid	0.000000
13	Age Bins_youngest	0.000000
12	Income Bins_high	0.000000
11	Income Bins_high-mid	0.000000
10	Income Bins_low-mid	0.000000
8	Time of Day_night	0.000000
7	Time of Day_evening	0.000000
6	Time of Day_afternoon	0.000000
5	Time of Day_morning	0.000000
4	Sex_Male	0.000000
3	Sex_Female	0.000000
2	Month	0.000000
16	Age Bins_oldest	0.000000

Before getting rid of Time Online and Time on Site binned features, lots of other features had small importances

Final Tree




```
Accuracy with modified alpha is: 0.9454545454545454
Depth of tree with modified alpha is: 3
Number of leaves with modified alpha is: 5
```

Next steps

Use NLP, vectorization and KMeans to cluster Ad topic lines

Add in continent features

Maps to show where data comes from

Cross-Validation and other models to test for over-fitting of this one…the accuracy is pretty high, I want to make sure it's not inflated

Apply to bigger set!