

Activity Report

On

Advertisement Prediction using Logistic Regression

For

(Data Analytics)

Submitted by

1930014 Devam Chakorborty

1930032 Ravi Prakash Mishra

1930033 Ravi Raj



**KALINGA INSTITUTE OF
INDUSTRIAL TECHNOLOGY (KIIT)**

— Deemed to be University U/S 3 of UGC Act, 1956 —

B. Tech in Electronics & Computer Science Engineering

School of Electronics Engineering

Kalinga Institute of Industrial Technology Deemed to be University

Bhubaneswar, India

November 2021

OBJECTIVE / AIM:

This project aims to use previous advertisement engagement data to create a model that will predict whether a user will click on an ad or not in the future. This would help the advertisers use the appropriate technique to increase their overall revenue through targeted advertising. As this is a binary, classifying decision, we will be using logistic regression to model this data.

THEORY

INTRODUCTION

The Online Advertising Market is expected to grow at a CAGR (**compound annual growth rate**) of 14.3% over the forecast period (2021 - 2026). The Covid-19 pandemic had a positive effect on the Online Advertising Market. People became more attracted onwards online portals and social networking sites, which resulted in increased online population and streaming.

This report is basically about the advertising data of a marketing agency for the development of an algorithm that predicts if a user will click on an advertisement or not. Given data consists of some variables like '**Daily Time Spent on Site**', '**Age**', '**Daily Internet Usage**', '**Area Income**', '**Ad Topic Line**', '**Timestamp**', '**Male**', '**Country**', '**City**' and '**Clicked on Ad**'.

SO, we will be looking at all other variables to accurately predict the value 'Clicked on Ad' variable. We will also perform some exploratory data analysis to see how 'Daily Time Spent on Site' in combination with 'Ad Topic Line' affects the user's decision to click on the ad.

Concept

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of the target or dependent variable is dichotomous, which means there would be only two possible classes. In simple words,

the dependent variable is binary having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no). The logistic regression equation has a very similar representation to linear regression. The difference is that the output value being modeled is binary in nature.

Mathematically, a logistic regression model predicts $P(Y=1)$ as a function of X . It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection, etc.

Types of Logistic Regression [1]

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into the following types –

- **Binary or Binomial**

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

- **Multinomial**

In such a kind of classification, the dependent variable can have 3 or more possible *unordered* types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

- **Ordinal**

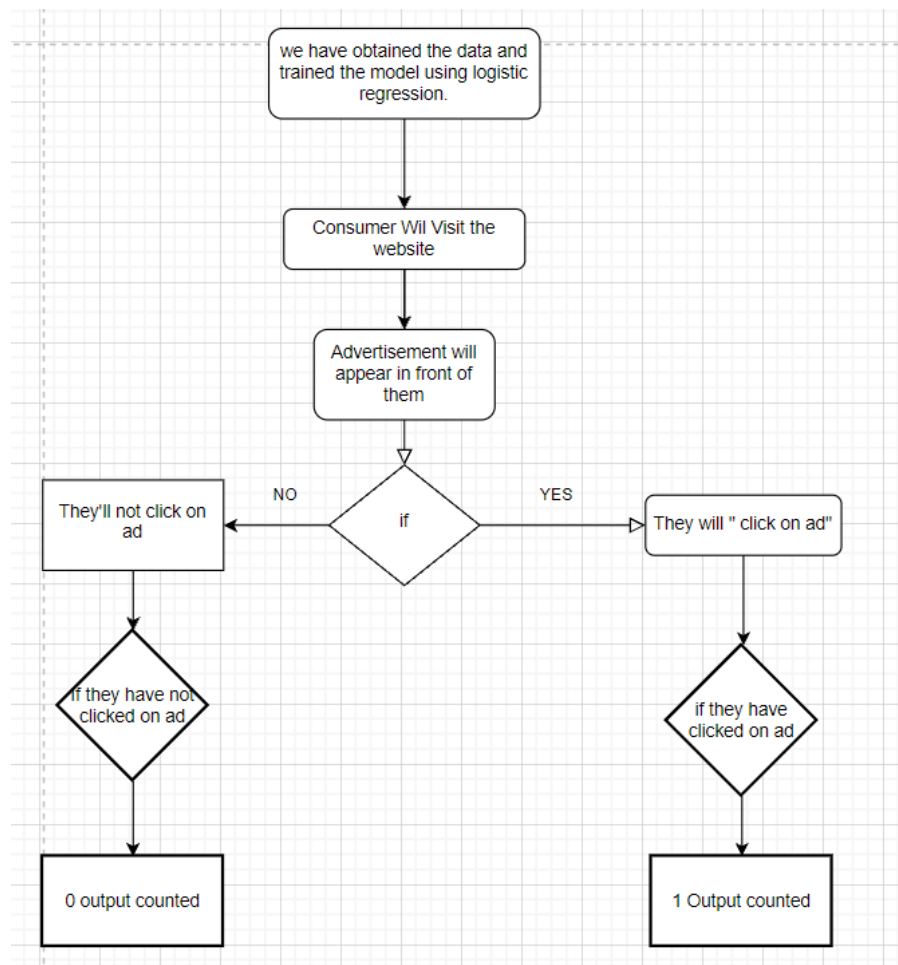
In such a kind of classification, the dependent variable can have 3 or more possible *ordered* types or the types having a **quantitative significance**. For example, these variables may represent “poor” or “good”, “very good”, “Excellent” and each category can have the scores like 0,1,2,3.

Logistic Regression Assumptions [2]

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same –

- In the case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multicollinearity in the model, which means the independent variables must be independent of each other.
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.

FLOWCHART

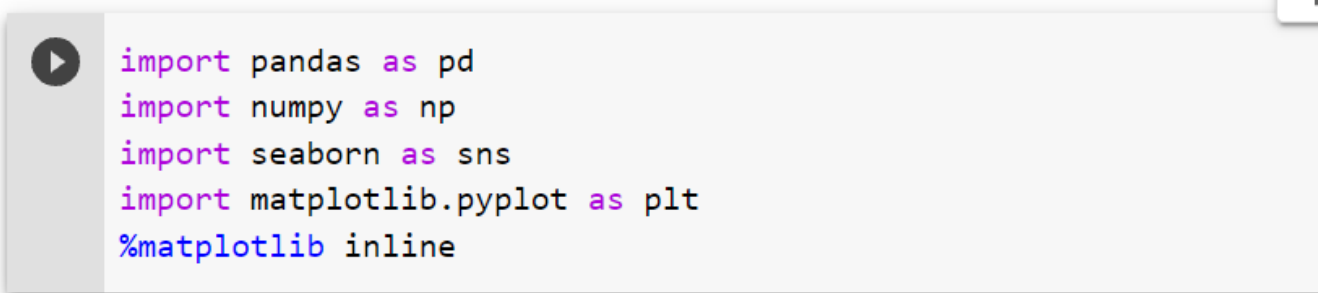


REQUIREMENTS

- A system with an installed **python environment** and some **libraries** to work upon it like:
 - padas,**
 - NumPy,**
 - seaborn,**
 - matplotlib.pyplot,**
 - sklearn.linear_model, etc.**
- **Data Set** “asvertising.csv” [\[3\]](#) (to work upon).

CODE

Now, Importing the libraries and retrieving Data,

A code editor snippet with a light gray background. On the left, there is a green checkmark and the text '1s'. Next to it is a black play button icon. The code is written in a monospaced font with syntax highlighting: 'import' is purple, 'pandas as pd' is black, 'numpy as np' is black, 'seaborn as sns' is black, 'matplotlib.pyplot as plt' is black, and '%matplotlib inline' is blue.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Loaded the data into variable “**ad_data**”, and viewed the dataset first 8 columns, the dataset is having few information regarding each column like

'Daily Time Spent on Site': consumer time on site in minutes
'Age': customer age in years
'Area Income': Avg. Income of geographical area of consumer
'Daily Internet Usage': Avg. minutes a day consumer is on the internet
'Ad Topic Line': Headline of the advertisement
'City': City of consumer
'Male': Whether or not consumer was male
'Country': Country of consumer
'Timestamp': Time at which consumer clicked on Ad or closed window
'Clicked on Ad': 0 or 1 indicated clicking on Ad

ADVERTISEMENT PREDICTION USING LOGISTIC REGRESSION

```
In [4]: ad_data = pd.read_csv('advertising.csv')
```

```
In [6]: ad_data.head(10)
```

Out[6]:

	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
5	59.99	23	59761.56	226.74	Sharable client-driven software	Jamieberg	1	Norway	2016-05-19 14:30:17	0
6	88.91	33	53852.85	208.36	Enhanced dedicated support	Brandonstad	0	Myanmar	2016-01-28 20:59:32	0
7	66.00	48	24593.33	131.76	Reactive local challenge	Port Jefferybury	1	Australia	2016-03-07 01:40:15	1

ad_data.info() info about data so that we must not miss, as all the 1000 entries are present and we need not do any data cleaning.

```
In [7]: ad_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Daily Time Spent on Site              1000 non-null   float64
1   Age                                   1000 non-null   int64
2   Area Income                           1000 non-null   float64
3   Daily Internet Usage                  1000 non-null   float64
4   Ad Topic Line                         1000 non-null   object
5   City                                  1000 non-null   object
6   Male                                  1000 non-null   int64
7   Country                               1000 non-null   object
8   Timestamp                             1000 non-null   object
9   Clicked on Ad                         1000 non-null   int64
dtypes: float64(3), int64(3), object(4)
memory usage: 78.2+ KB
```

Describe() function to gain insight into the ranges in which variables change

ADVERTISEMENT PREDICTION USING LOGISTIC REGRESSION

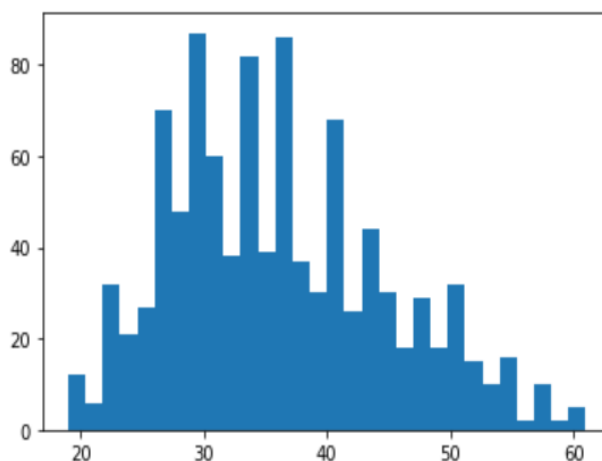
```
In [8]: ad_data.describe()
```

Out[8]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male	Clicked on Ad
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	65.000200	36.009000	55000.000080	180.000100	0.481000	0.500000
std	15.853615	8.785562	13414.634022	43.902339	0.499889	0.500025
min	32.600000	19.000000	13996.500000	104.780000	0.000000	0.000000
25%	51.360000	29.000000	47031.802500	138.830000	0.000000	0.000000
50%	68.215000	35.000000	57012.300000	183.130000	0.000000	0.500000
75%	78.547500	42.000000	65470.635000	218.792500	1.000000	1.000000
max	91.430000	61.000000	79484.800000	269.960000	1.000000	1.000000

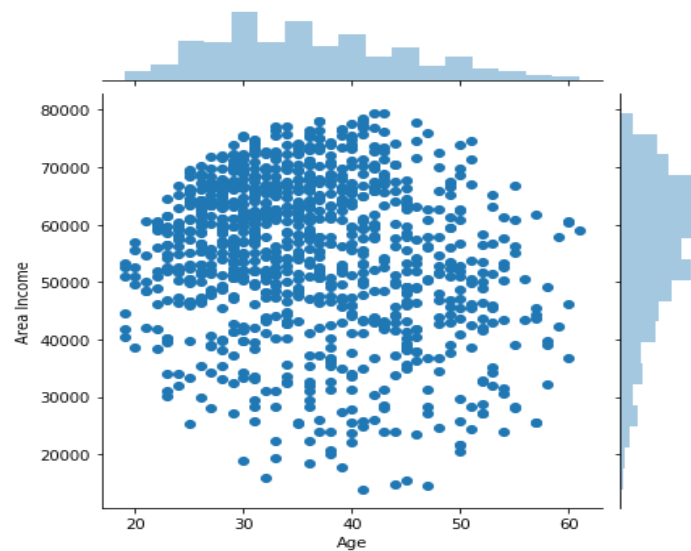
Now first, we'll **explore the data** to get a better understanding of what is happening. With Clicked on Ad being the variable in question, we'll be looking for the factors that could impact this the most.

```
In [9]: plt.hist(ad_data['Age'], bins = 30)
```



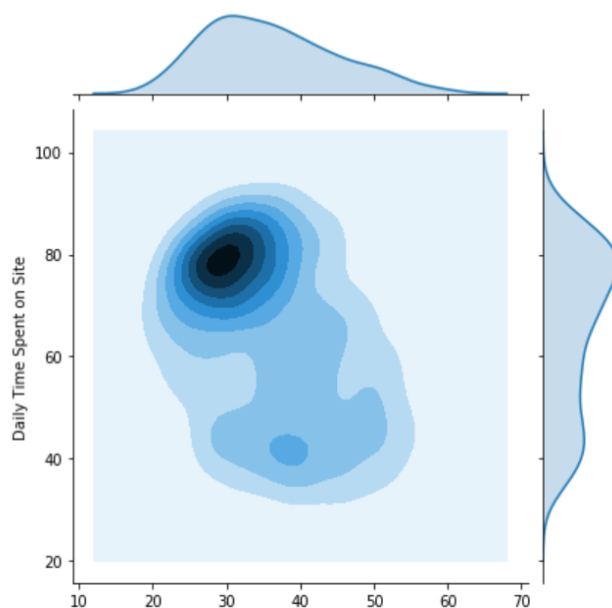
```
In [10]: sns.jointplot(ad_data['Age'], ad_data['Area Income'])
```

Out[10]: <seaborn.axisgrid.JointGrid at 0x1d836a9ce08>



We can see a **larger concentration in the top left**. Knowing there are more users of younger age, it makes sense that there would be a higher density on the left side of the graph. However, It looks like the majority of those younger people have an income above \$45,000.

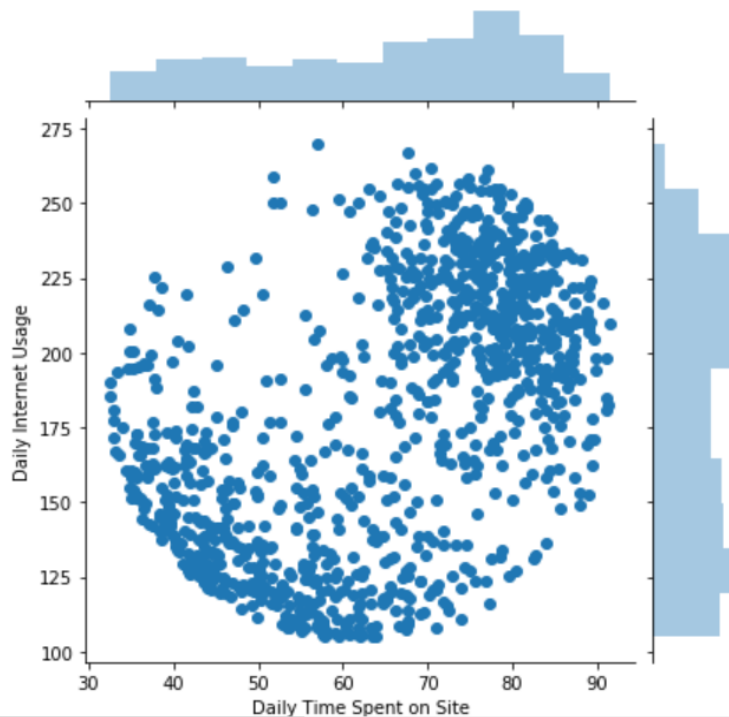
```
In [11]: #Daily Time Spent on Site is also another factor that could be important in determining what the user does.  
#Of course, Daily Time Spent on Site will also be very different given the user's age.  
sns.jointplot(ad_data['Age'],ad_data['Daily Time Spent on Site'], kind='kde')  
  
Out[11]: <seaborn.axisgrid.JointGrid at 0x1d836be5c08>
```



We see from this that most of these users are between 25 and 40 and spend over 60 minutes on the site each day, which means young ones are spending more time.


```
In [12]: #Is the daily time spent on site the same as the daily internet usage?  
sns.jointplot(ad_data['Daily Time Spent on Site'],ad_data['Daily Internet Usage'])
```

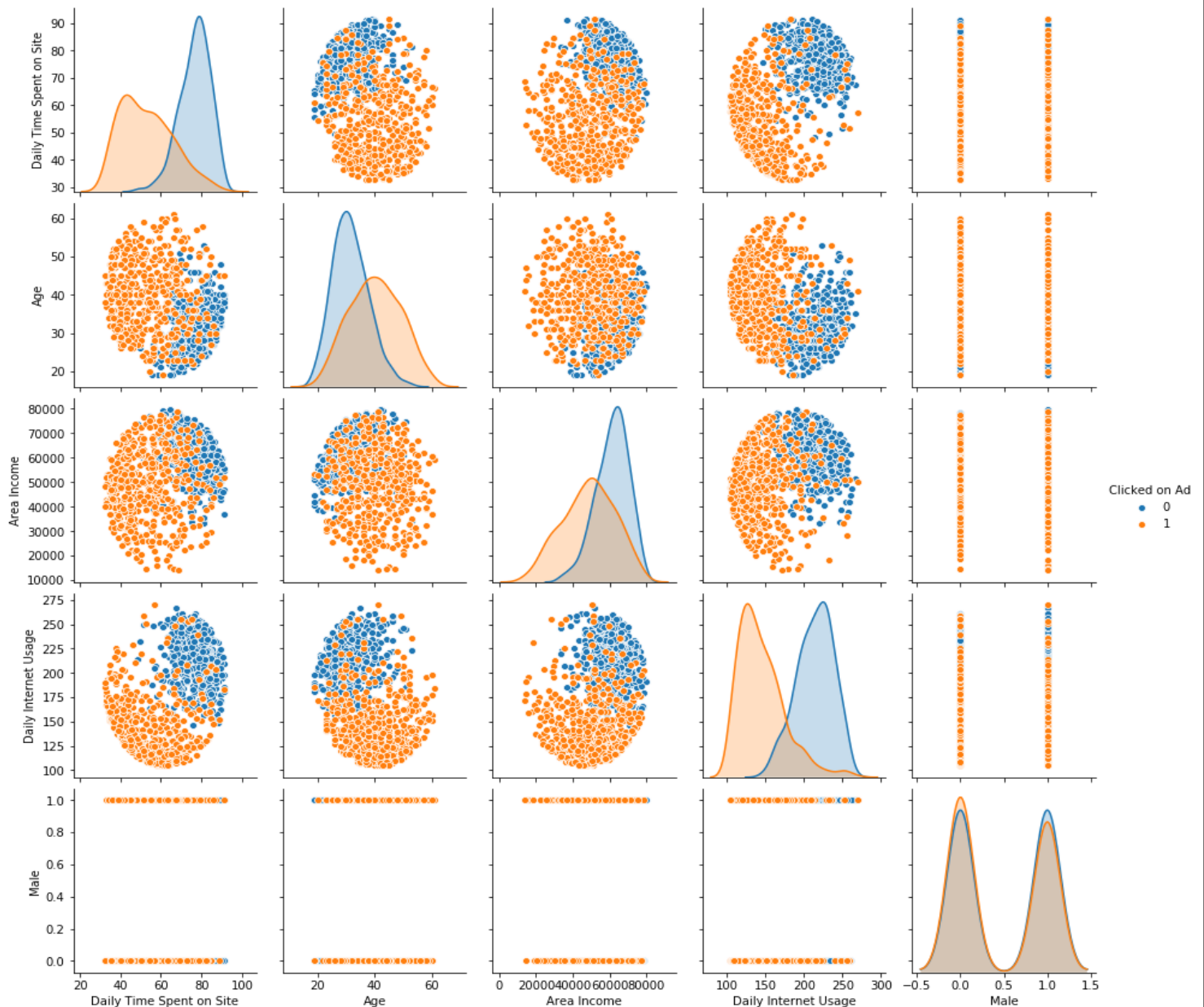
```
Out[12]: <seaborn.axisgrid.JointGrid at 0x1d836d2dec8>
```



With almost a circular dataset, users' usage is pretty spread out, with concentrations roughly being in high internet usage and high daily time spent on site and low internet usage and low daily time spent on site.

In the below figure, with a better understanding of what our users are like, let's plot them all together in regards to whether they clicked on the Ad or not to give us some visual understanding of what's happening.

ADVERTISEMENT PREDICTION USING LOGISTIC REGRESSION



What's clear from the above is that there is a higher concentration of people who clicked on the ad if they:

1. Spent less time on the Site
2. Spent less time on the Internet every day

Visually, these seem to be the identifying factors in whether they will click on an ad. Now I'll move on to performing a logistic regression to determine if these variables are as important as I hypothesize.

OBSERVATIONS/RESULTS

- **Building a Logistic Regression Model, Training and Predicting.**

It's time to split the data set into two, train the model, and test how it performs. We'll be dropping the **Ad Header** and **City** columns for the first model.

I'm assuming, for this relatively small data set, the *city will not be impactful* enough to make much of a difference in my analysis, though I will be keeping the country in as a dummy variable. We'll be changing the timestamp data into months.

After both our X data and y data are formatted correctly, I'll split the data into training and test data sets to cross-validate our model when we want to see how it performs.

```
In [24]: from sklearn.model_selection import train_test_split
```

```
In [25]: X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.30, random_state=101)
```

```
In [26]: from sklearn.linear_model import LogisticRegression
```

```
In [27]: logmodel1 = LogisticRegression(solver = 'lbfgs')
```

```
In [28]: logmodel1.fit(X1_train,y1_train)
```

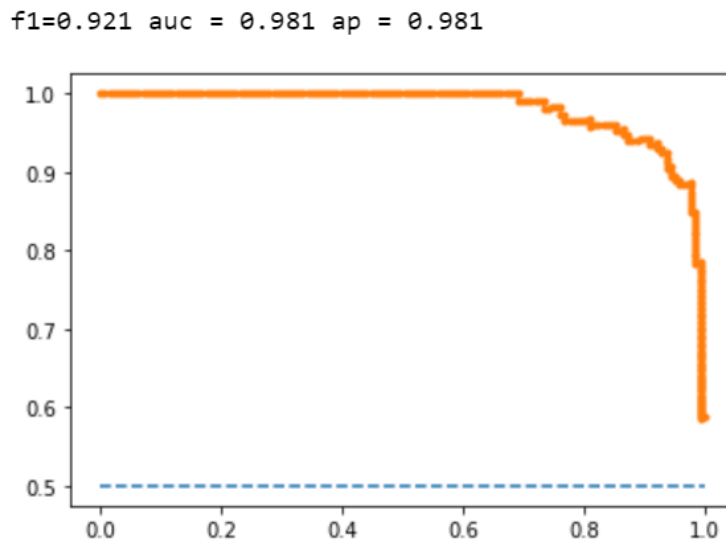
```
Out[28]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                             intercept_scaling=1, l1_ratio=None, max_iter=100,
                             multi_class='auto', n_jobs=None, penalty='l2',
                             random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                             warm_start=False)
```

```
In [29]: predict1 = logmodel1.predict(X1_test)
```

```
In [37]: #A relatively simple way is by looking at the classification report
#This gives you a quick, informative overview on how your model performs.
from sklearn.metrics import classification_report
print(classification_report(y1_test,predict1))
```

	precision	recall	f1-score	support
0	0.91	0.95	0.93	157
1	0.94	0.90	0.92	143
accuracy			0.93	300
macro avg	0.93	0.93	0.93	300
weighted avg	0.93	0.93	0.93	300

We can also draw a diagram of a **precision-recall curve** as,



DISCUSSION OF RESULTS

Precision, recall, and f1-score are used to understand how your data performs, especially on an imbalanced dataset. Our data is balanced, meaning the column to be classified has a 50/50 chance of being clicked on or not.

Precision is the ratio of correctly predicted positive outcomes to all positively predicted outcomes - it essentially tells us "How many people clicked after being labeled they would?".

The **recall** is the ratio of how many correctly predicted positive outcomes to how many people survived in total (thus including rows that the model labeled that they wouldn't click on the ad even though they did) - this answers the question "Of those who clicked on the ad, how many did we predict to do so?"

The **F1-score** is a weighted average of Precision and recall. It's harder to interpret but allows us to combine precision and recall and understand what the overall performance is to be used in comparison with other models.

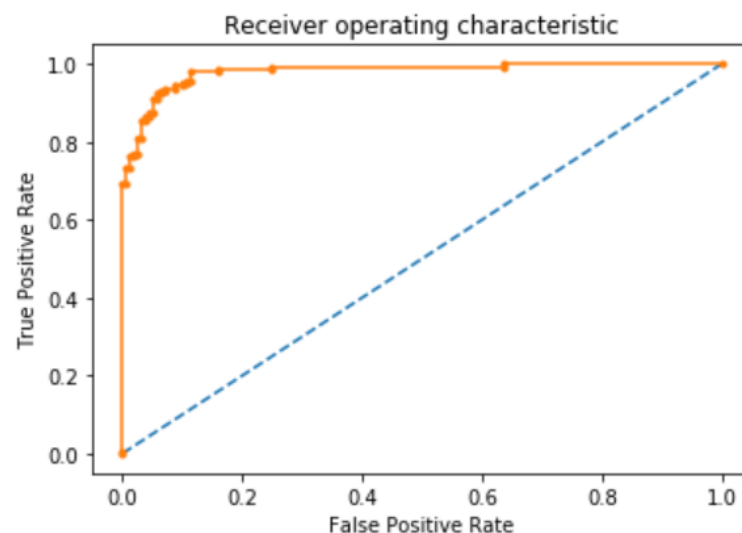
Remember that a completely random model would get 0.5 (50/50 chance) for these metrics, and a perfect model would be 1.0

It's also important to note that the classification report and the precision-recall curve metrics are different, even though they measure the same thing, due to how they are measured. The classification report only uses a binary yes/no outcome to measure these metrics, whereas the precision-recall curve breaks each row into a probability of being yes, causing the differences. Both are important to understand because both give slightly different answers depending on the question you are asking.

With this said, **anything over 0.9 is very good!** But we'll also want to see what else we can learn about our model

```
In [39]: ► ROCAUC(X1_test,y1_test,logmodel1)
```

AUC: 0.981



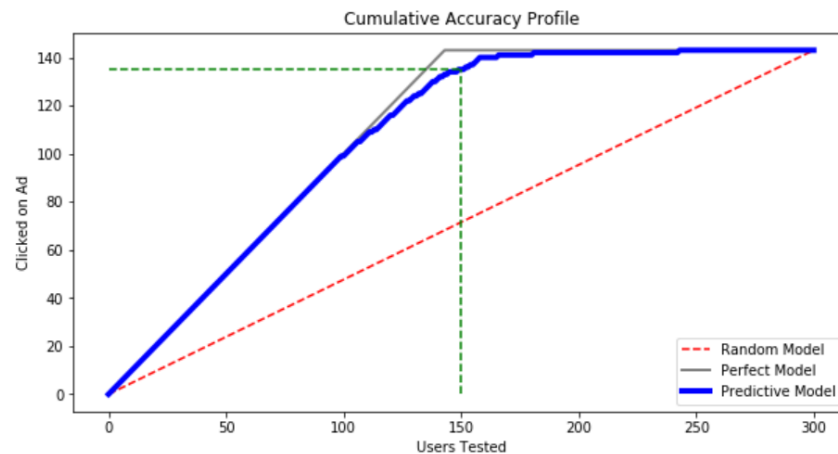
This is the **ROC curve**. The better the model, the larger the Area Under the Curve (AUC), and the farther away to the top left the model will be compared to the dotted line (a completely random model). Ours with an AUC of 0.98 is very good!

As we've observed so far, our model seems to be good. But how can we apply this?

Well, for starters, we can look at the **Cumulative Accuracy Profile (CAP)** Curve to help us understand how well if we predicted a dataset, we could determine if they would click on an ad or not.

```
In [41]: CAP(X1_test,y1_test,logmodel1)
```

```
Percentage of Total Positive Outcomes at 50%: 94.4055944055944%
Accuracy Ratio: 96.13380250322926
```



We see that our percentage of **total positive outcomes** at 50% is **94%**, which means that we can identify 94% of the people who will **click on our ad** by just applying our model to the top 50% of the users in our test data. Typically, anything **above 90% is an exceptional model**, meaning this model is very good. This is valuable information for budgeting for a given user set. To apply this concept, you could apply this model directly to each user coming in and only show the ad to users with a high probability rating or you could apply the model to a new data set of users and you could order the data by the probability that they'll click the ad and then use this information to choose which users you want to contact out of the list.

But what we're primarily interested in is what variables/attributes of users are important for them to click on the ad according to our model. For that, we'll create a coefficient report.

CONCLUSION:

In this analysis, we made a model using logistic regression to help us try to predict whether or not users will click on our ads. We made an excellent model (at least partly attributed to having a fictional dataset). We also saw that, by splitting the country into

a dummy variable, we were unable to use Recursive Feature Elimination and we saw the effects of that on our model.

We were able to interpret our model to help us understand what variables affect whether someone will click on an ad; namely, we found that an increase in Age had the largest impact on the odds of someone clicking our Ad.

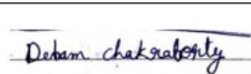
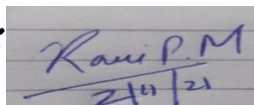
Finally, we noted along the way that our model could be used not only in predicting whether a future user will click on an ad or not, but also in other applications, from budgeting how much we should spend on advertising to identifying what kind of users we should target with our ad.

References

- [1] S. Swaminathan, "Towards Data Science," 15 March 2018. [Online]. Available: <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>.
- [2] IBM, "IBM," [Online]. Available: <https://www.ibm.com/topics/logistic-regression>.
- [3] "Kaggle," [Online]. Available: <https://www.kaggle.com>.

STUDENT SIGNATURES

Ravi Raj
01-11-21



SIGNATURE OF THE CONCERNED FACULTY