Logistic Regression

April 24, 2019

1 Logistic Regression Project

In this project I worked with a fake advertising data set, indicating whether or not a particular internet user clicked on an Advertisement. I have tried to create a logistic regression model that would predict whether or not they will click on an ad based off the features of that user.

The data set had the following features:

- 'Daily Time Spent on Site': consumer time on site in minutes
- 'Age': cutomer 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

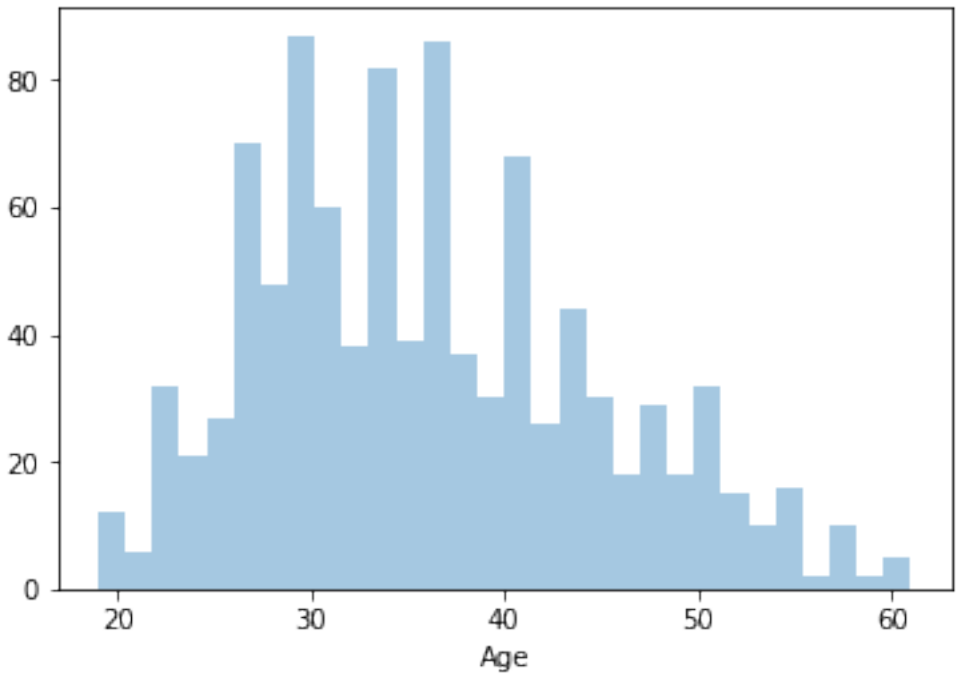
```
In [7]: import sklearn
        import numpy as np
        import seaborn as sns

In [2]: from sklearn.linear_model import LogisticRegression

In [3]: import pandas as pd

In [5]: data = pd.read_csv('advertising.csv')

In [6]: data.head()
```

```
Out[6]:    Daily Time Spent on Site  Age  Area Income  Daily Internet Usage  \
        0                     68.95   35     61833.90                256.09
        1                     80.23   31     68441.85                193.77
        2                     69.47   26     59785.94                236.50
        3                     74.15   29     54806.18                245.89
        4                     68.37   35     73889.99                225.58

                                Ad Topic Line           City  Male     Country  \
```

```
0     Cloned 5thgeneration orchestration     Wrightburgh     0     Tunisia
1     Monitored national standardization       West Jodi     1       Nauru
2       Organic bottom-line service-desk        Davidton     0  San Marino
3  Triple-buffered reciprocal time-frame  West Terrifurt     1       Italy
4           Robust logistical utilization    South Manuel     0     Iceland

             Timestamp  Clicked on Ad
0  2016-03-27 00:53:11              0
1  2016-04-04 01:39:02              0
2  2016-03-13 20:35:42              0
3  2016-01-10 02:31:19              0
4  2016-06-03 03:36:18              0
```

```
In [10]: sns.distplot(data['Age'],kde=False,bins=30)
```

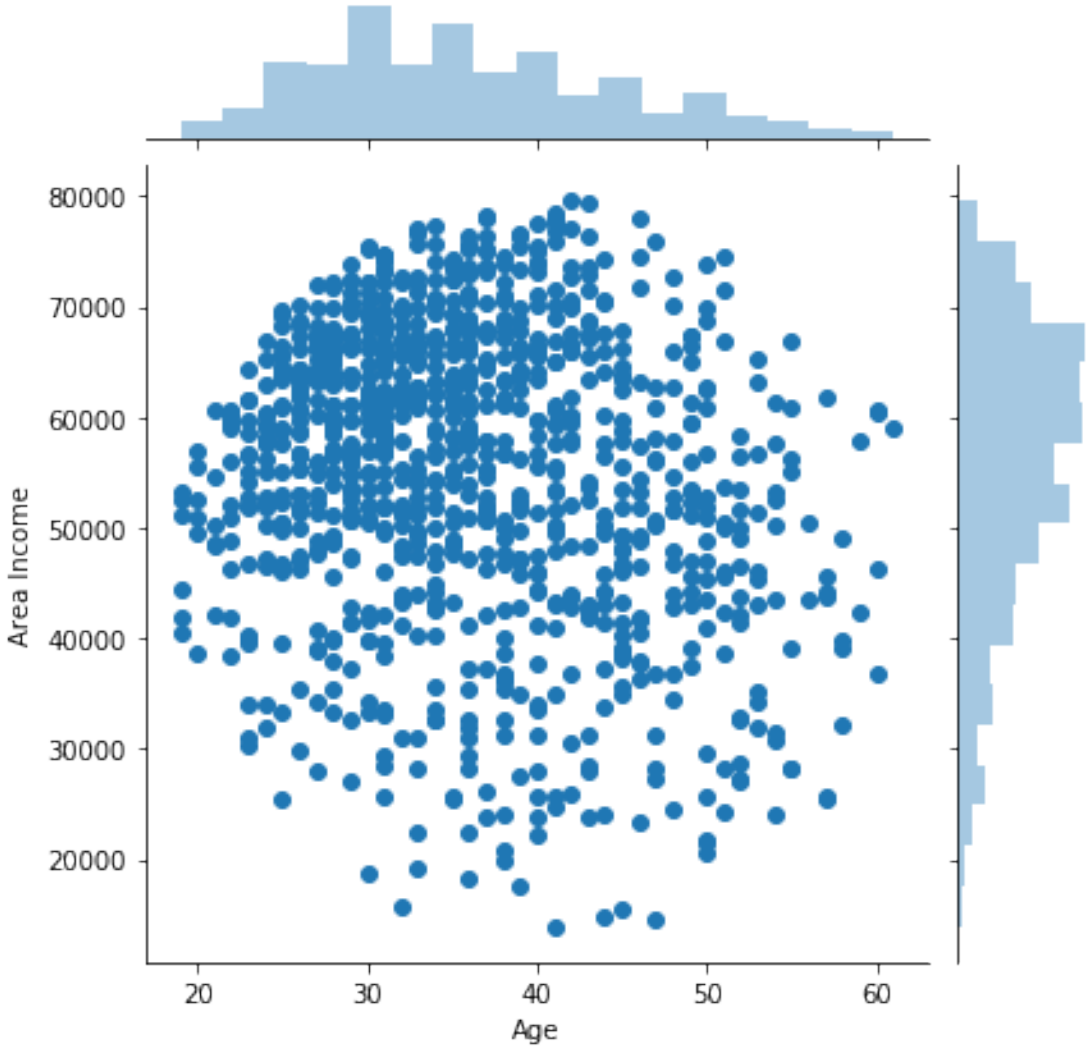
```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x23356bd16a0>
```



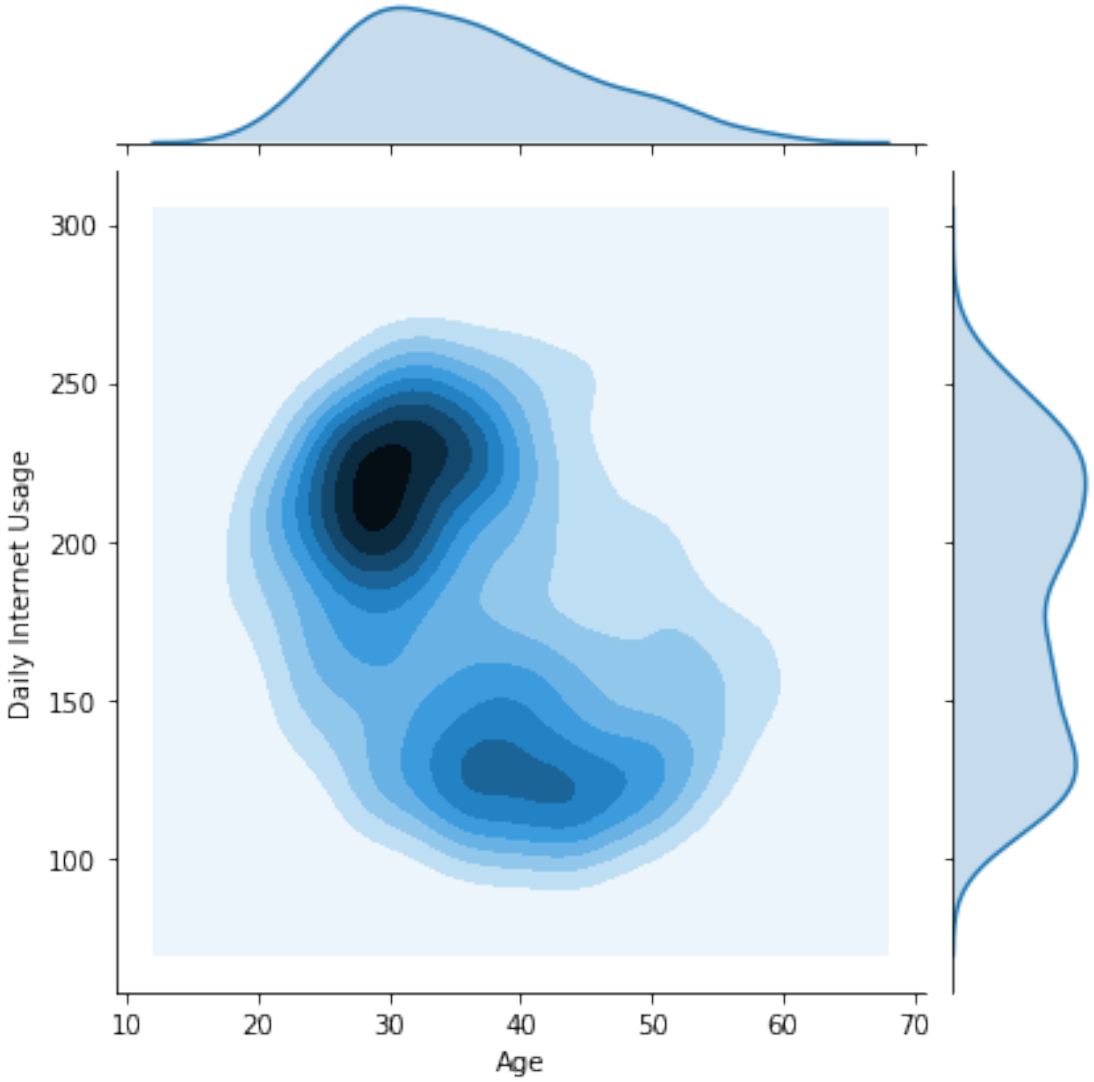

```
In [13]: sns.jointplot(data['Age'],data['Area Income'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a no
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
Out[13]: <seaborn.axisgrid.JointGrid at 0x233572e3cc0>
```

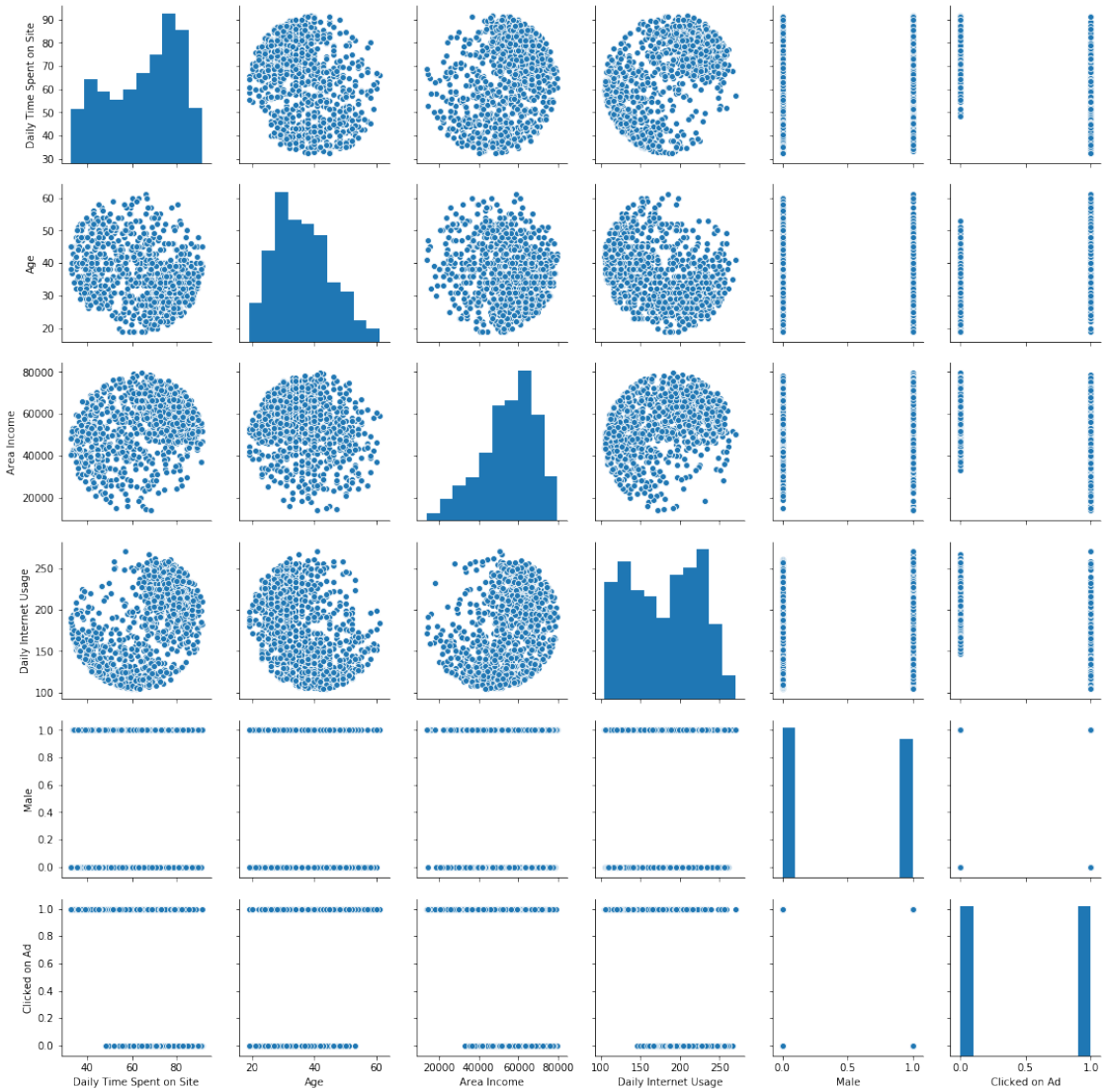



```
In [14]: sns.jointplot(data['Age'],data['Daily Internet Usage'],kind='kde')

C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a no
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval


Out[14]: <seaborn.axisgrid.JointGrid at 0x233573ca7f0>
```




```
In [15]: sns.pairplot(data)
```

```
Out[15]: <seaborn.axisgrid.PairGrid at 0x23357363748>
```



In [16]:
```
data.corr()
```

Out[16]:

	Daily Time Spent on Site	Age	Area Income
Daily Time Spent on Site	1.000000	-0.331513	0.310954
Age	-0.331513	1.000000	-0.182605
Area Income	0.310954	-0.182605	1.000000
Daily Internet Usage	0.518658	-0.367209	0.337496
Male	-0.018951	-0.021044	0.001322
Clicked on Ad	-0.748117	0.492531	-0.476255

	Daily Internet Usage	Male	Clicked on Ad
Daily Time Spent on Site	0.518658	-0.018951	-0.748117
Age	-0.367209	-0.021044	0.492531

```
Area Income                    0.337496  0.001322      -0.476255
Daily Internet Usage           1.000000  0.028012      -0.786539
Male                           0.028012  1.000000      -0.038027
Clicked on Ad                 -0.786539 -0.038027       1.000000
```

Performed various visualizations to find if there is any multi collinearity present in the independent variables

```
In [25]: data_x = data

In [27]: data_y = data_x.pop('Clicked on Ad')
```

Splitting the dataset into training and testing data

```
In [28]: from sklearn.model_selection import train_test_split

In [108]: train_x,test_x,train_y,test_y = train_test_split(data_x,
                                                           data_y, test_size=0.30,
                                                           random_state=101)
```

Using Logistic Regression from Scikit Learn and fitting it on training data and then using it to predict targets on test data

```
In [30]: from sklearn.linear_model import LogisticRegression

In [99]: logmodel = LogisticRegression()
         logmodel.fit(train_x,train_y)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning
  FutureWarning)


Out[99]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)

In [37]: predict_y = logmodel.predict(test_x)

In [102]: logmodel.coef_

Out[102]: array([[-4.45816498e-02,  2.18379839e-01, -7.63621393e-06,
                  -2.45264007e-02,  1.13334440e-03]])
```

In the following lines, I have created a confusion matrix to check the classification rate

```
In [38]: from sklearn.metrics import confusion_matrix

In [39]: cm =confusion_matrix(test_y,predict_y)
         cm
```

```
Out[39]: array([[149,   8],
                [ 15, 128]], dtype=int64)
```

In the following lines, I have played with pandas to add predicted target and actual target into dataframe and created a new variable which reports whether classification was correct or not.

```
In [40]: b = test_x
         b["Predicted Target"] = predict_y.reshape((-1,1))
         b["Actual Target"] = test_y
         b
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[40]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male
545	42.60	55	55121.65	168.29	0
298	80.39	31	66269.49	214.74	0
109	74.02	32	72272.90	210.54	0
837	55.77	49	55942.04	117.33	1
194	77.20	33	49325.48	254.05	1
605	64.67	51	24316.61	138.35	1
246	39.96	45	59610.81	146.13	1
693	65.15	29	41335.84	117.30	0
802	50.63	50	25767.16	142.23	0
406	56.34	50	68713.70	139.02	1
824	68.47	28	67033.34	226.64	0
977	54.97	31	51900.03	116.38	1
742	71.90	29	72203.96	193.29	1
341	72.23	48	52736.33	115.35	0
601	52.13	50	40926.93	118.27	1
319	62.06	44	44174.25	105.00	0
121	49.58	26	56791.75	231.94	0
299	65.80	25	60843.32	231.49	1
32	59.05	57	25583.29	169.23	1
269	79.15	26	62312.23	203.23	0
887	52.17	44	57594.70	115.37	1
668	52.56	31	33147.19	250.36	1
820	57.51	38	47682.28	105.71	0

754	76.87	36	72948.76	212.59	0
731	61.87	35	66629.61	250.20	1
225	78.70	30	53441.69	133.99	0
569	71.28	37	67307.43	246.72	1
56	65.19	36	75254.88	150.61	0
408	51.68	49	51067.54	258.62	0
734	56.57	26	56435.60	131.98	0
..	...	...	...	...	...
532	79.53	51	46132.18	244.91	0
866	86.58	32	60151.77	195.93	1
555	72.53	37	73474.82	223.93	0
716	39.96	35	53898.89	138.52	1
81	73.46	28	65653.47	222.75	1
756	56.93	37	57887.64	111.80	0
796	79.22	27	63429.18	198.79	1
612	75.19	31	33502.57	245.76	1
699	73.88	29	63109.74	233.61	0
915	34.04	34	40182.84	174.88	1
801	53.33	34	44275.13	111.63	1
729	79.60	28	56570.06	227.37	1
577	81.21	36	63394.41	233.04	0
260	74.65	28	67669.06	212.56	0
938	32.60	38	40159.20	190.05	0
389	63.88	38	19991.72	136.85	0
476	78.76	24	46422.76	219.98	1
202	35.65	40	31265.75	172.58	1
442	36.44	39	52400.88	147.64	1
113	46.13	46	37838.72	123.64	1
620	81.75	24	52656.13	190.08	1
550	75.80	36	71222.40	224.90	0
27	51.95	52	58295.82	129.23	0
720	75.71	34	62109.80	246.06	0
945	89.80	36	57330.43	198.24	0
847	88.82	36	58638.75	169.10	0
682	43.57	36	50971.73	125.20	1
884	70.92	39	66522.79	249.81	1
325	36.87	36	29398.61	195.91	0
482	69.42	25	65791.17	213.38	0

	Predicted Target	Actual Target
545	1	1
298	0	0
109	0	0
837	1	1
194	0	0
605	1	1
246	1	1
693	1	1

802	1	1
406	1	1
824	0	0
977	1	1
742	0	0
341	1	1
601	1	1
319	1	1
121	0	0
299	0	0
32	1	1
269	0	0
887	1	1
668	0	1
820	1	1
754	0	0
731	0	0
225	0	1
569	0	0
56	1	1
408	1	1
734	0	1
..	...	...
532	1	0
866	0	0
555	0	0
716	1	1
81	0	0
756	1	1
796	0	0
612	0	0
699	0	0
915	1	1
801	1	1
729	0	0
577	0	0
260	0	0
938	1	1
389	1	1
476	0	0
202	1	1
442	1	1
113	1	1
620	0	0
550	0	0
27	1	1
720	0	0
945	0	0

```
847                  0              0
682                  1              1
884                  0              0
325                  1              1
482                  0              0

[300 rows x 7 columns]
```

```
In [79]: test_y = test_y.tolist()
```

```
In [81]: correct =[]
         for i in range(0, len(test_y)):
             Yo = test_y[i];Ys= predict_y[i]
             if(Yo==Ys):
                 correct.append('Yes')
             else:
                 correct.append('No')
         b['Correct Prediction?'] = correct
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

```
In [113]: b.head()
```

```
Out[113]:      Daily Time Spent on Site  Age  Area Income  Daily Internet Usage  Male  \
          545                     42.60   55     55121.65                168.29     0
          298                     80.39   31     66269.49                214.74     0
          109                     74.02   32     72272.90                210.54     0
          837                     55.77   49     55942.04                117.33     1
          194                     77.20   33     49325.48                254.05     1

               Predicted Target  Actual Target Correct Prediction?
          545                 1              1                 Yes
          298                 0              0                 Yes
          109                 0              0                 Yes
          837                 1              1                 Yes
          194                 0              0                 Yes
```

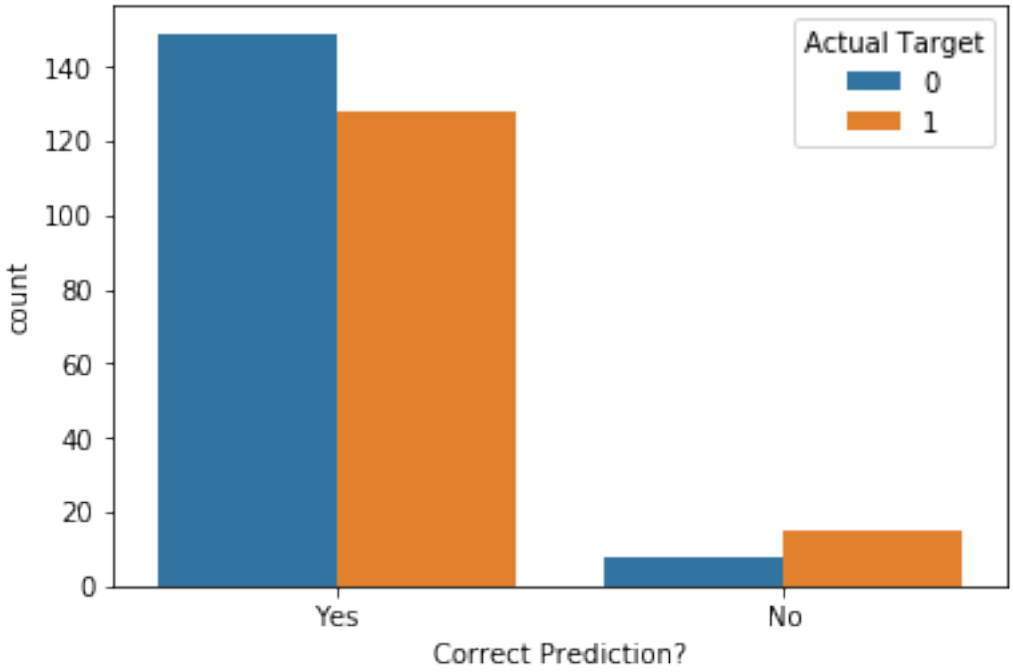
```
In [55]: from sklearn.metrics import accuracy_score
```

```
In [94]: accuracy = accuracy_score(test_y,predict_y)
```

```
In [95]: accuracy
```

```
Out[95]: 0.9233333333333333
```

```
In [115]: sns.countplot(b['Correct Prediction?'],hue= b['Actual Target'])

Out[115]: <matplotlib.axes._subplots.AxesSubplot at 0x2335e3a4cc0>
```



We have got an accuracy of 0.92 from the model created and the above visulaization shows how many correct predictions were made values 0 and 1 individually.

```
In [62]: from sklearn.neural_network import MLPClassifier as mlp

In [64]: clf = mlp()

In [65]: clf.fit(train_x,train_y)

Out[65]: MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                       beta_2=0.999, early_stopping=False, epsilon=1e-08,
                       hidden_layer_sizes=(100,), learning_rate='constant',
                       learning_rate_init=0.001, max_iter=200, momentum=0.9,
                       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
                       random_state=None, shuffle=True, solver='adam', tol=0.0001,
                       validation_fraction=0.1, verbose=False, warm_start=False)

In [88]: predict_y_2 = clf.predict(test_x)

In [92]: accuracy_mlp = accuracy_score(test_y,predict_y_2)

In [93]: print(accuracy)
```

```
0.5233333333333333
```

I have further compared this with MLP classifier model from neural networks which produced very less accuracy.