Logistic Regression

I received a dataset (*Social_Network_Ads.csv*) with 400 users showing their gender, age, estimated salary, and whether they purchased a specific vehicle that was on sale (where 1 = yes and 2 = no).

The models are using *age* and *estimated salary* in order to predict whether the individual purchased the vehicle. Therefore *User ID* and *Gender* are discarded.

Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

I split the data into a training (75%) and test set (25%). Thus 300 rows will be used to train the model, and 100 will be used to test the model. After running each specific model, I used a confusion matrix to determine how accurate the predictions were.

Logistic Regression

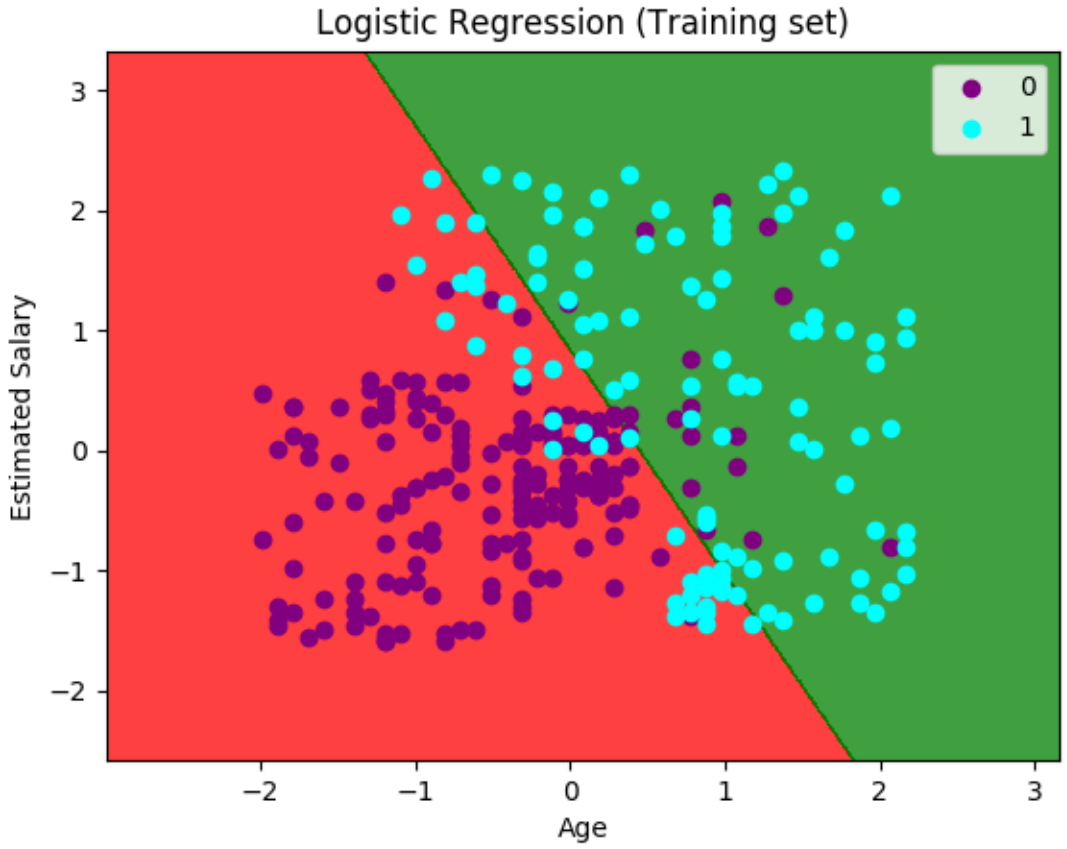
From confusion matrix below shows there was 89 correct predictions and 11 incorrect predictions giving us 89% accuracy.

```
# Making the confustion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, y_pred)
```

```
In [13]: cm
Out[13]:
array([[65,  3],
       [ 8, 24]])
```

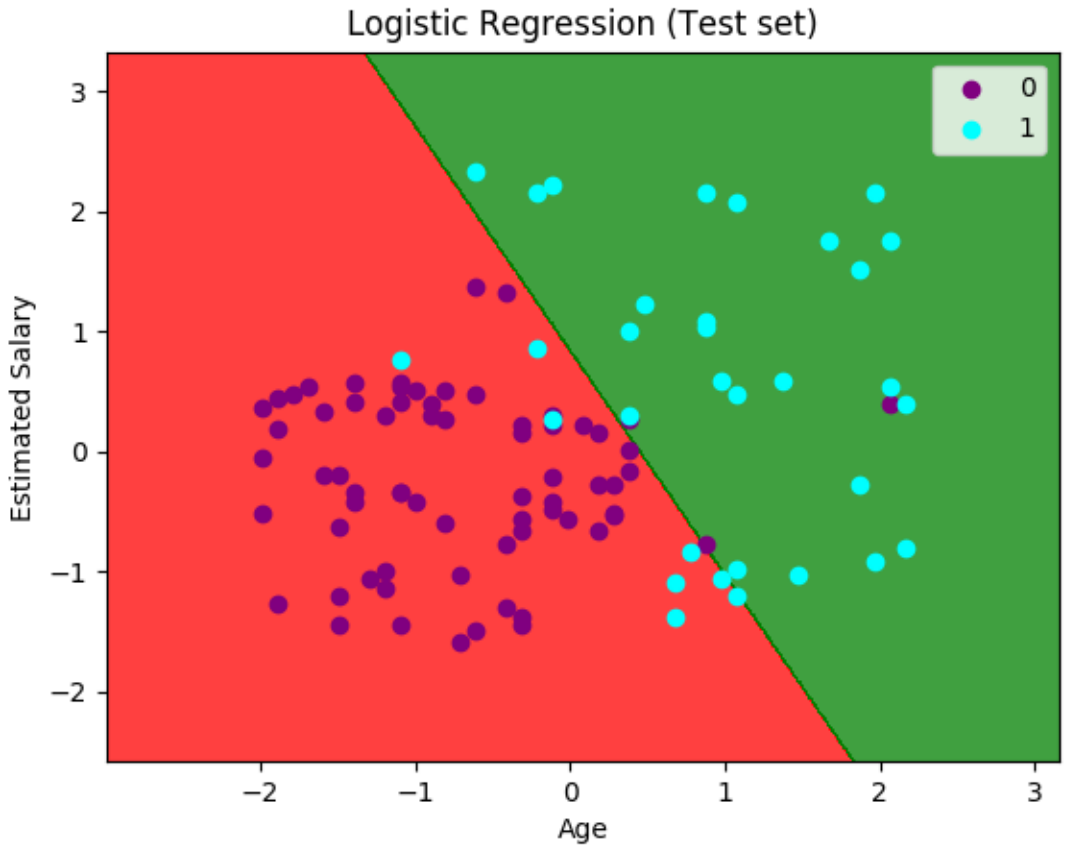
The diagram below shows the results of the **training set**.

- The points are the individuals in the dataset.
- The *purple points* are the training set observations where the dependent variable *purchased* is 0 (didn't buy the vehicle).
- The *cyan points* are the training set observations where the dependent variable *purchased* is 1 (bough the vehicle).
- The points within *red region* are the members our classifier will predict who won't buy the vehicle.
- The points within *green region* are the members our classifier will predict who will buy the vehicle.[1]



[1] These 5 points are true of all graphs in this document unless otherwise stated.

The diagram below shows the results of the **test set**.



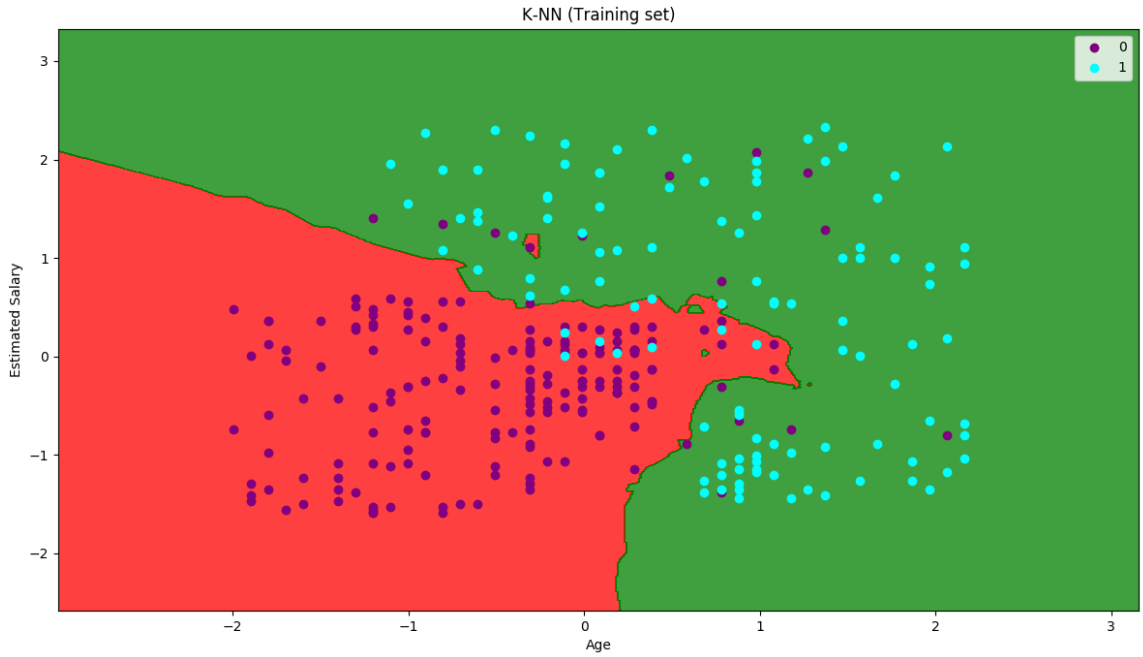
K-Nearest Neighbor

From confusion matrix below shows there was 93 correct predictions and 7 incorrect predictions giving us 93% accuracy.

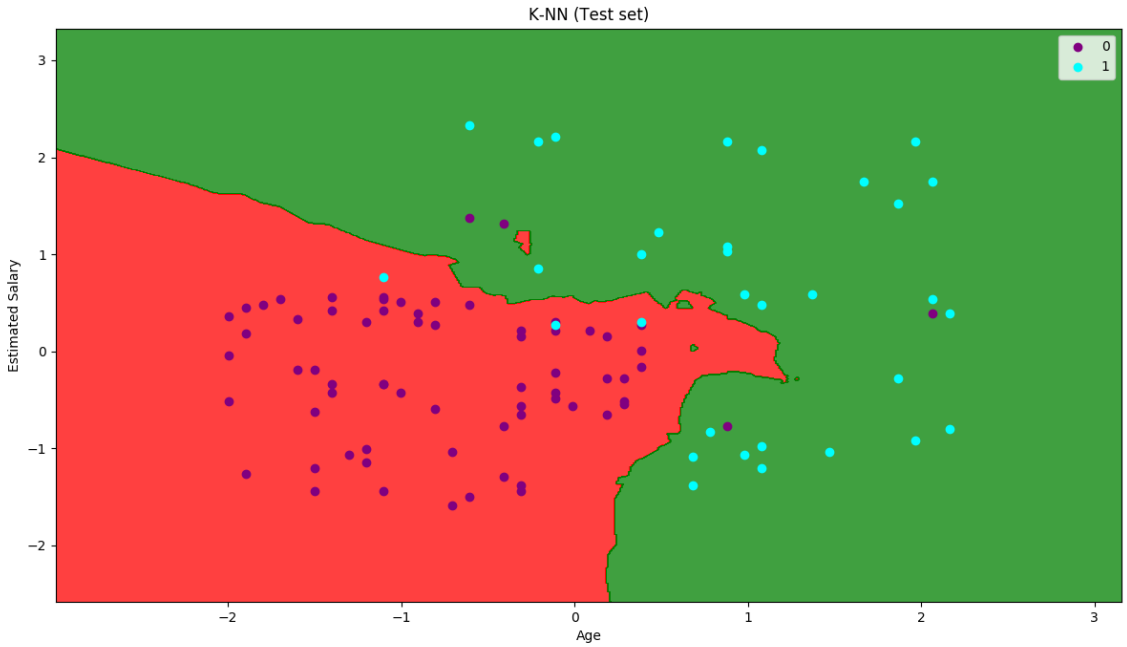
```
In [7]: from sklearn.metrics import confusion_matrix
   ...: cm = confusion_matrix(Y_test, y_pred)

In [8]: cm
Out[8]:
array([[64,  4],
       [ 3, 29]])
```

The diagram below shows the results of the **training set**.



The diagram below shows the results of the **test set**.



SVM

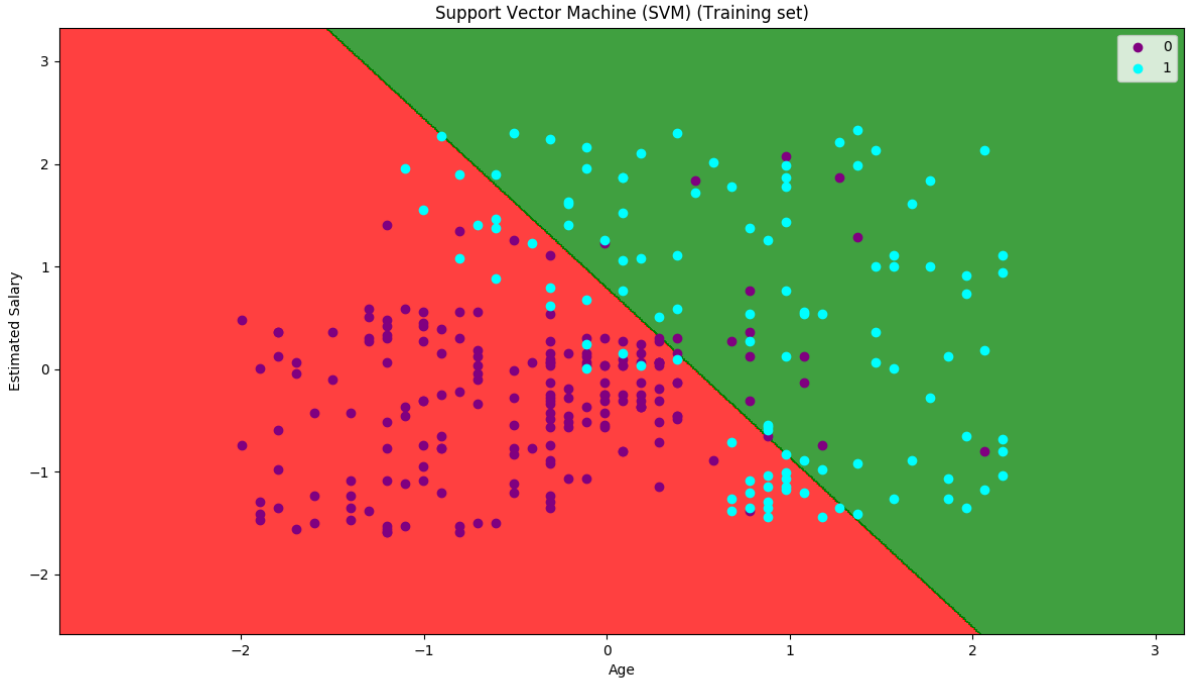
The SVM in this case is a linear classifier.

From confusion matrix below shows there was 90 correct predictions and 10 incorrect predictions giving us 90% accuracy.

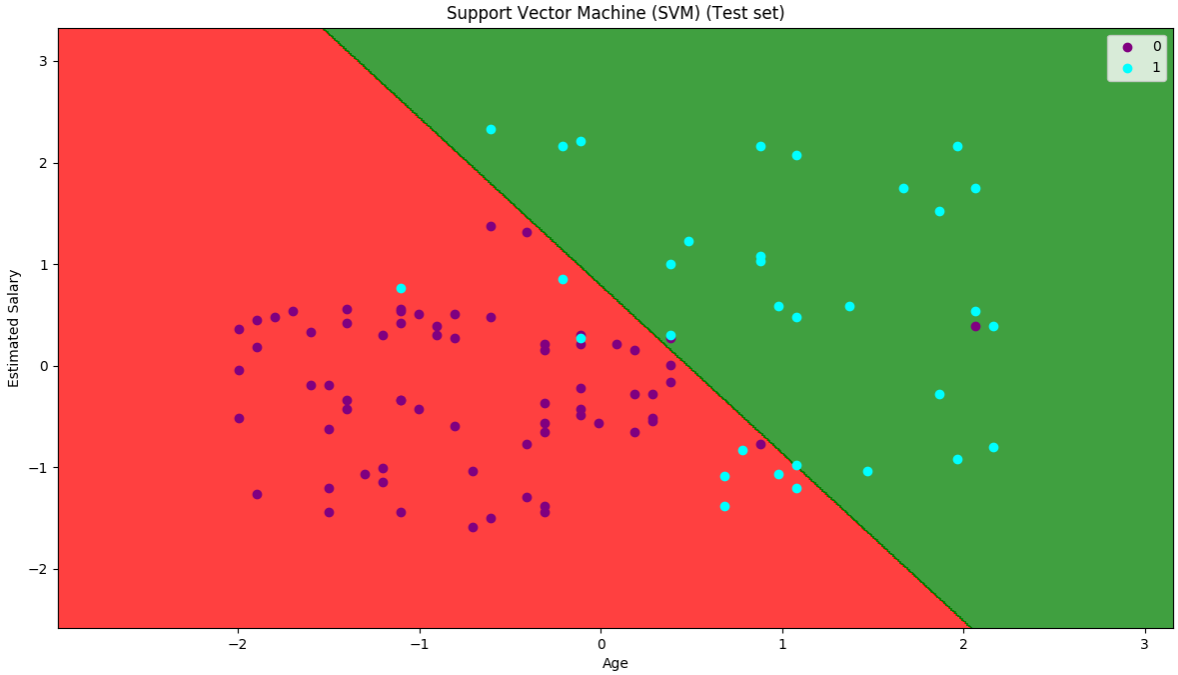
```
In [3]: from sklearn.metrics import confusion_matrix
   ...: cm = confusion_matrix(Y_test, y_pred)

In [4]: cm
Out[4]:
array([[66,  2],
       [ 8, 24]])
```

The diagram below shows the results of the **training set**.



The diagram below shows the results of the **test set**.



Kernel SVM

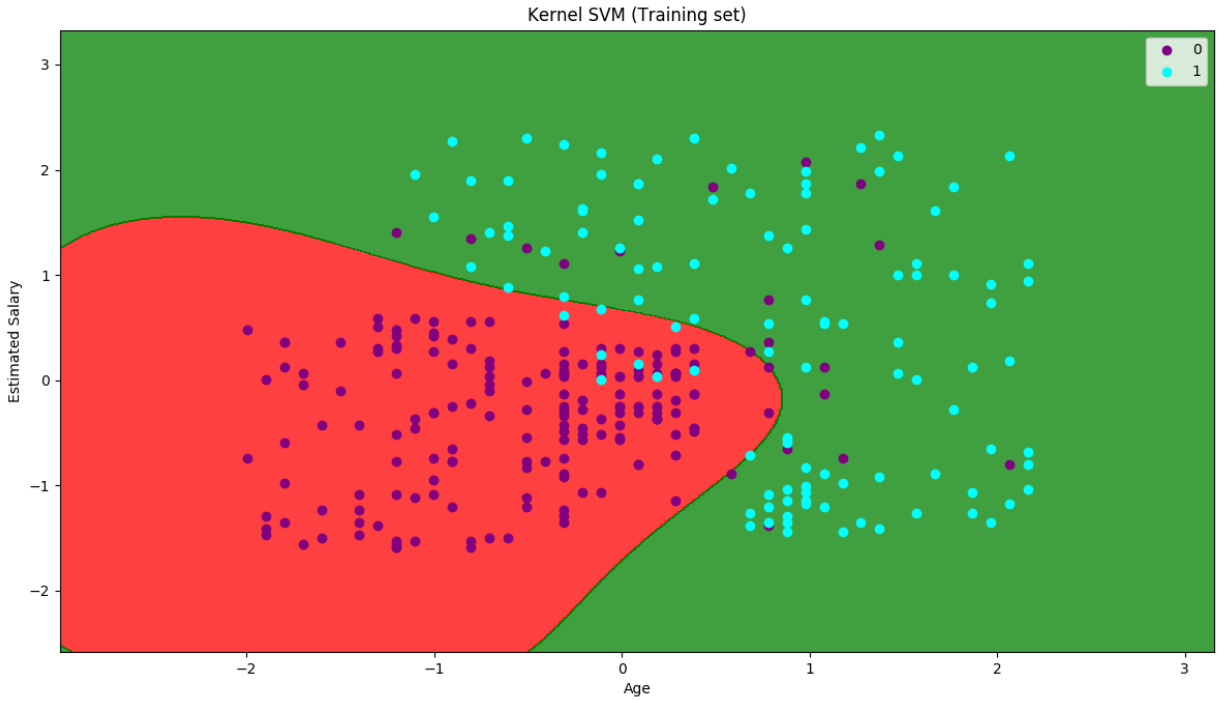
This version of SVM is a non linear classification.

From confusion matrix below shows there was 93 correct predictions and 7 incorrect predictions giving us 93% accuracy.

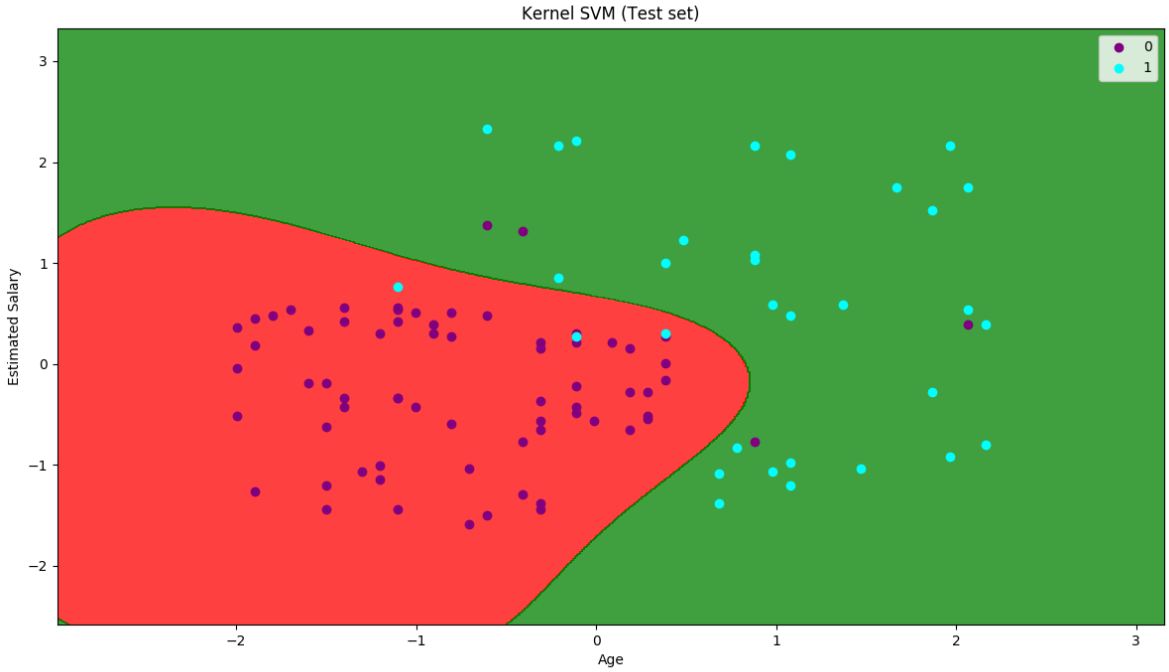
```
In [14]: from sklearn.metrics import confusion_matrix
    ...: cm = confusion_matrix(Y_test, y_pred)

In [15]: cm
Out[15]:
array([[64,  4],
       [ 3, 29]])
```

The diagram below shows the results of the **training set**.



The diagram below shows the results of the **test set**.



As you can see the Kernel SVM (nonlinear) is 3% more accurate than the linear version of SVM.