FINAL TERM PROJECT

Data Mining

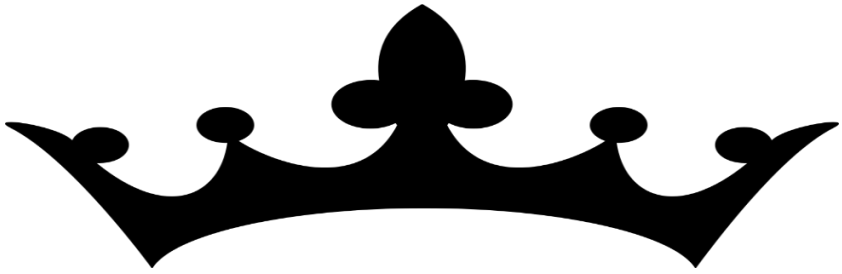
Supervised Data Mining – Classification

Category 3(Decision Trees)

Category 5(Naïve Bayes)

Software Tool - Category 10(Python)

https://scikit-learn.org/



Name: Jhona Davied D souza

UCID: jd655

Table of Contents

ABSTRACT

Goal

To evaluate the performance of classification algorithms namely Naïve Bayes and Decision Tree Classifier in predicting the value of independent variables

About the Dataset

The dataset consists of four fields namely User Id, Age, Estimated Salary and Purchased. The data is that of users who bought an SUV. The machine has to predict for given age and estimated salary whether a user will buy the SUV or not.

We first have to train the model on a training set which is 75 per cent of the whole data and then predict results on a test set of remaining 25 per cent data

Software Used

1. Anaconda Python(https://www.anaconda.com/distribution/)
2. IDE: Spyder from Anaconda
3. Dataset: https://www.kaggle.com/rakeshrau/social-network-ads

Libraries used

1. https://scikit-learn.org/

NAÏVE BAYES CLASSIFICATION ALGORITHM

Source Code

```
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 23 10:58:41 2019

@author: jhona
"""

# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
```

```
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Steps of Execution

1. Set the working directory where your program file and dataset will be stored

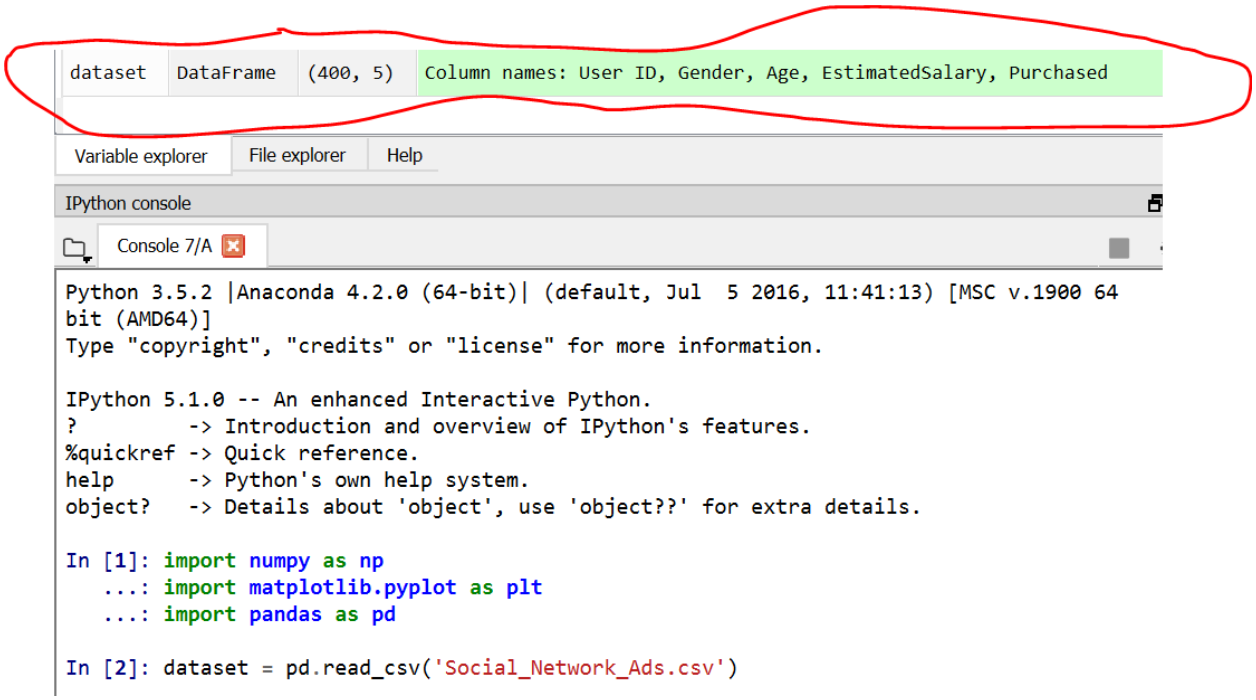
2. Import the libraries
   a) Numpy – math libraries
   b) Matplotlib- library used to plot charts
   c) Pandas – to import and manage datasets

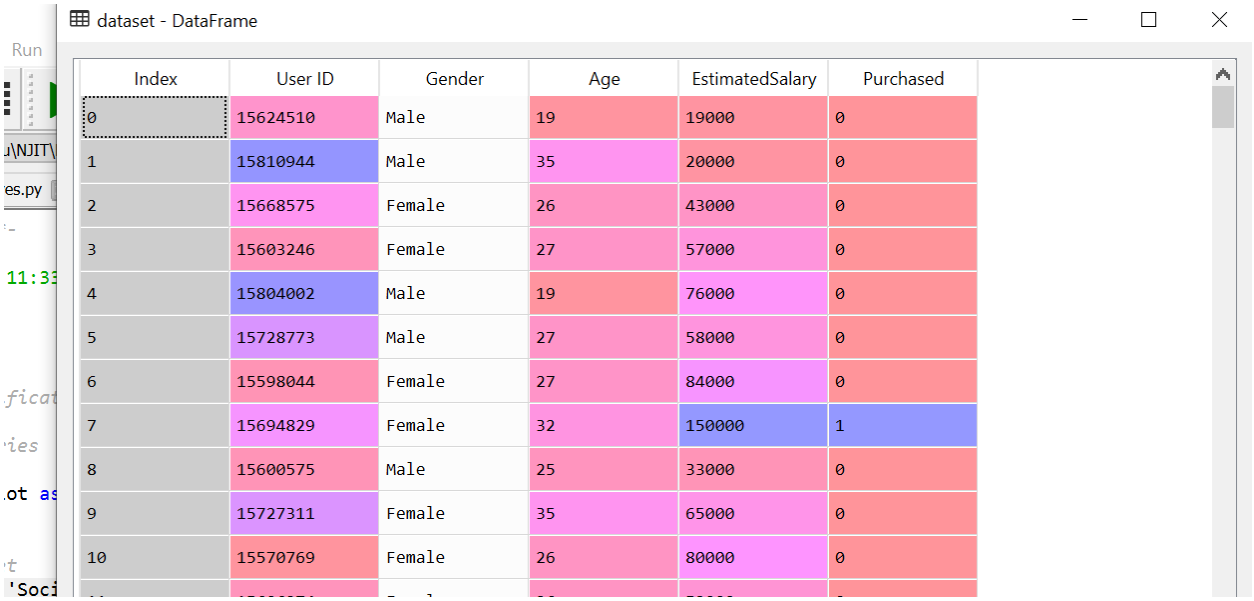
```
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul  5 2016, 11:41:13) [MSC v.1900 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: import pandas as pd
```

3. Import the dataset

   Use pandas to import the dataset

   a) Pd.read_csv('Social_Network_Ads.csv')





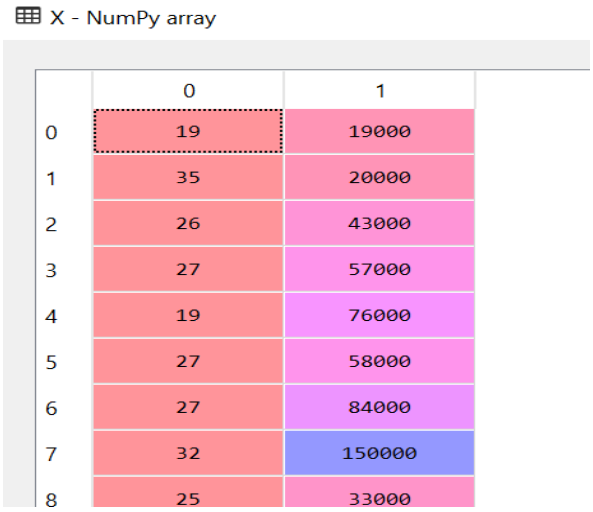
Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0

b) Distinguish matrix of features (independent variables) and dependent variables

a)

```
In [3]: X = dataset.iloc[:, [2, 3]].values
```

– The second and third column Age and Estimated salary will be the independent variables



	0	1
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000

b)

```
In [4]: y = dataset.iloc[:, 4].values
```

The fourth column Purchased will be the dependent variable

y - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0

X	int64	(400, 2)	array([[19, 19000], [35, 20000],
dataset	DataFrame	(400, 5)	Column names: User ID, Gender, Age, EstimatedSalary, Purchased
y	int64	(400,)	array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1… 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…

Variable explorer | File explorer | Help

IPython console

Console 11/A

```
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: import pandas as pd
   ...:

In [2]: dataset = pd.read_csv('Social_Network_Ads.csv')

In [3]: X = dataset.iloc[:, [2, 3]].values

In [4]: y = dataset.iloc[:, 4].values

In [5]:
```

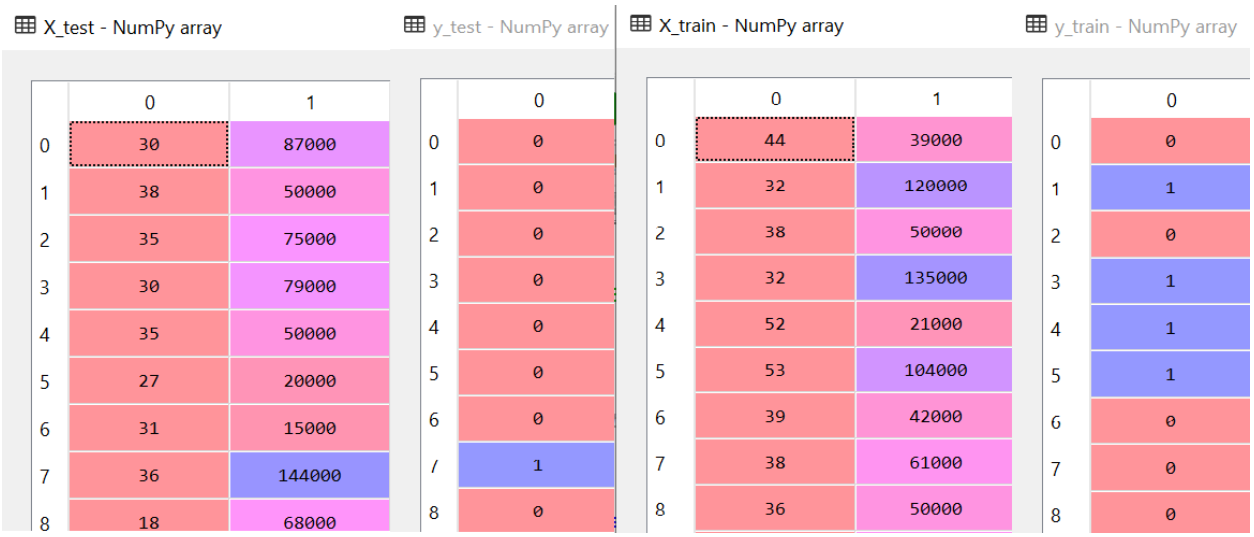
4. Splitting the dataset into training set and test set

   (Using the cross_validation library from sckit-learn)

```
In [5]: from sklearn.cross_validation import train_test_split
   ...: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)
```

   a) Create X_train, X_test, y_train, y_test

   b) Use train_test_split(X,y,test_size=0.25, random_state= 0) – 25 percent of data goes to test set remaining training set , random_state=0 gives same result everytime



X_test - NumPy array

	0	1
0	30	87000
1	38	50000
2	35	75000
3	30	79000
4	35	50000
5	27	20000
6	31	15000
7	36	144000
8	18	68000

y_test - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0

X_train - NumPy array

	0	1
0	44	39000
1	32	120000
2	38	50000
3	32	135000
4	52	21000
5	53	104000
6	39	42000
7	38	61000
8	36	50000

y_train - NumPy array

	0
0	0
1	1
2	0
3	1
4	1
5	1
6	0
7	0
8	0

5. Features Scaling

   Due to difference in scale of age and estimated salary the Euclidean distance won't be measured accurately and hence we need to use feature scaling

   We use the preprocessing library from sckit-learn.org and import Standard Scaler to fit_and_transform training set and test set

```
In [6]: from sklearn.preprocessing import StandardScaler
   ...: sc = StandardScaler()
   ...: X_train = sc.fit_transform(X_train)
   ...: X_test = sc.transform(X_test)
```

//for X_test we don't have to use fit since it is already fitted to the training set

X_train - NumPy array

	0	1
0	0.582	-0.887
1	-0.607	1.462
2	-0.013	-0.568
3	-0.607	1.897
4	1.374	-1.409
5	1.473	0.998
6	0.086	-0.800
7	-0.013	-0.249
8	-0.211	-0.568

X_test - NumPy array

	0	1
0	-0.805	0.505
1	-0.013	-0.568
2	-0.310	0.157
3	-0.805	0.273
4	-0.310	-0.568
5	-1.102	-1.438
6	-0.706	-1.583
7	-0.211	2.158
8	-1.993	-0.046

6. Fitting Naïve Bayes classifier to the training set

   Use the GaussianNB from Naïve Bayes library from sckit-learn.org
   Fit the Naïve Bayes classifier to X_train and Y_train

```
In [7]: from sklearn.naive_bayes import GaussianNB
   ...: classifier = GaussianNB()
   ...: classifier.fit(X_train, y_train)
Out[7]: GaussianNB()
```

7. Predicting the test set results

   Gets the vector of predictions y_pred

```
In [8]: y_pred = classifier.predict(X_test)
   ...:
```

Compare y_pred and y_test we see first 6 predictions are same

(For Naïve Bayes)

y_test - NumPy array y_pred - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	1
10	0

We observe first 8 predictions are correct but 9th prediction is 1(customer purchased the product) contrary to y_test

8. Making the confusion matrix

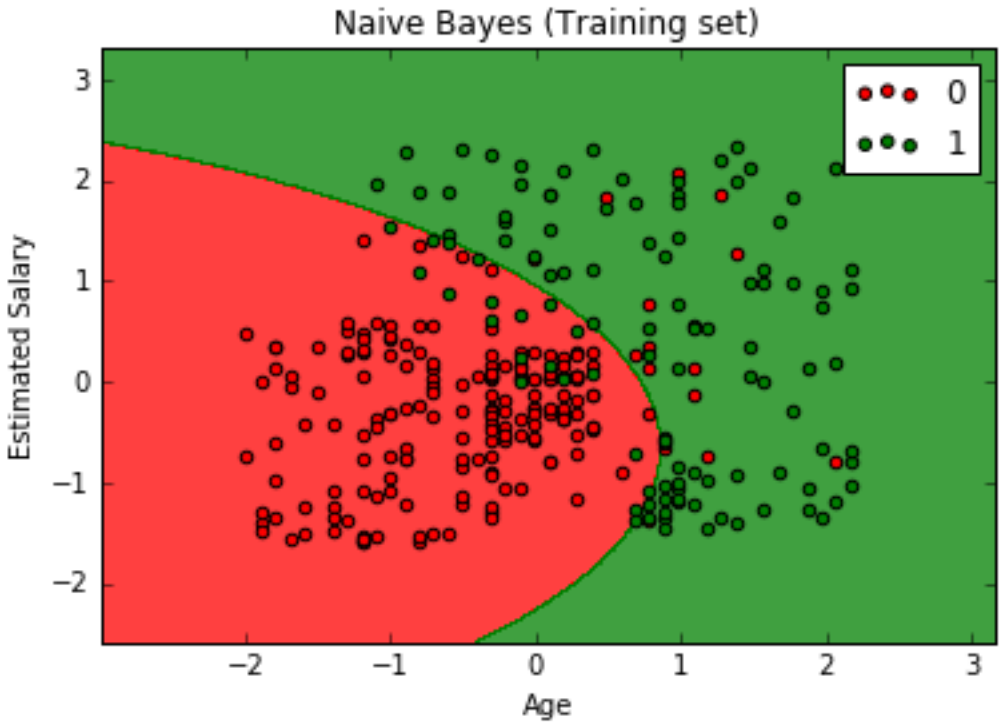
```
In [9]: from sklearn.metrics import confusion_matrix
   ...: cm = confusion_matrix(y_test, y_pred)
```

The confusion matrix evaluates the number of correct and incorrect predictions

```
In [10]: cm
Out[10]:
array([[65,  3],
       [ 7, 25]])
```

We have 65+25=90 correct predictions and 7+3=10 incorrect predictions which is quite good

9. Visualization of training set results



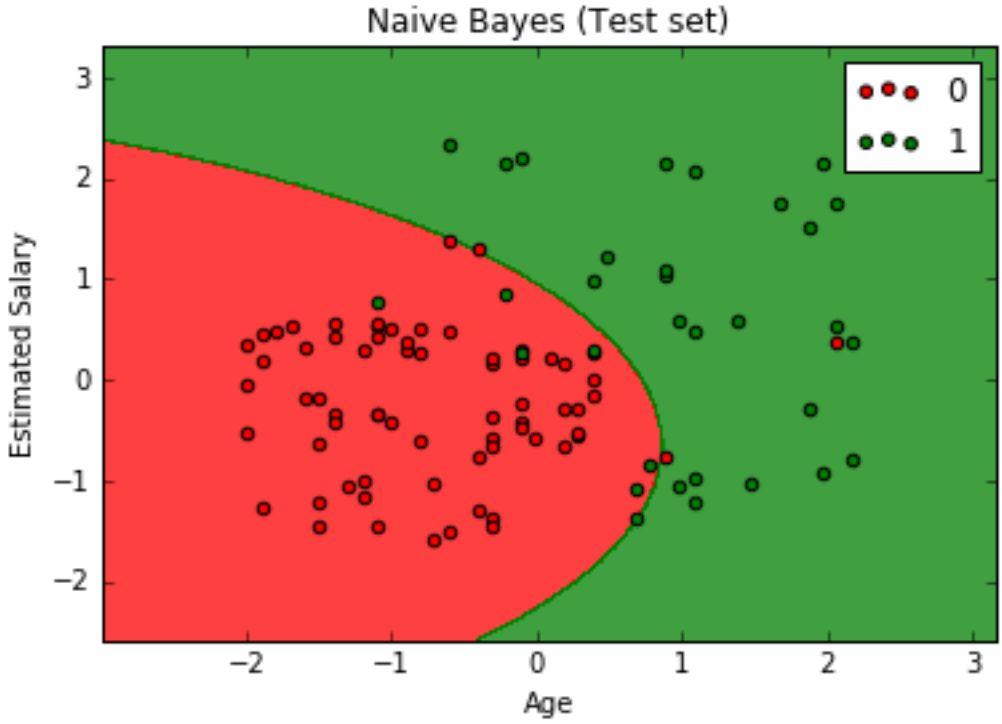
The **red** **points** represent the users who didn't buy the SUV, whereas the **green** **points** represent the users who bought the SUV.

The **red** **region** represents the users who didn't buy the SUV as predicted by the classifier and the **green** **region** represents the users who bought the SUV.

As you can see from the graph there are some green points inside the red region, these are the points that were predicted by the classifier as users who didn't buy the SUV but actually did buy the SUV

Whereas there are some red points inside the green region which represents the users that were predicted to buy the SUV but actually didn't buy it.

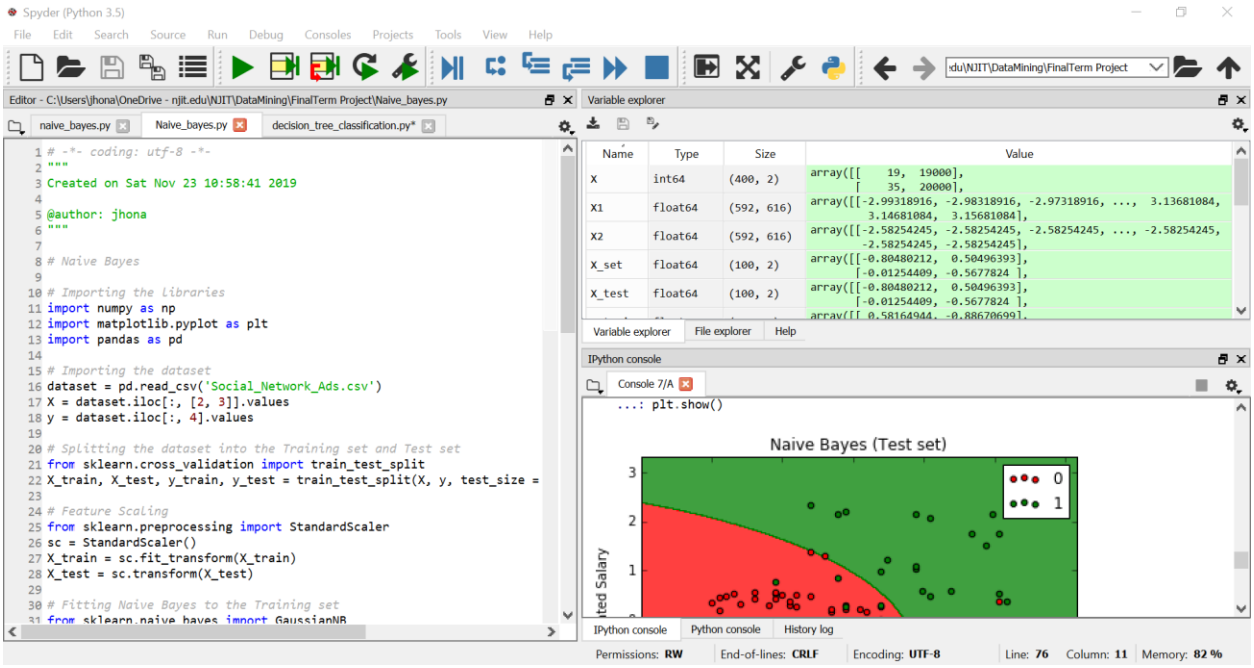
10. Visualizing the test set results



As you can see the test set results are pretty impressive, most of the red points are in the red region and the green points in the green region.

If you count the number of green points in the red region plus the number of red points in the green region you shall get a total of 10 which was the result given by the confusion matrix

Screenshot showing the running of Naïve Bayes



Decision Trees Classification

Source code

```
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 23 11:33:07 2019

@author: jhona
"""

# Decision Tree Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
```

```
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Steps of Execution

1. Set the working directory where your program file and dataset will be stored

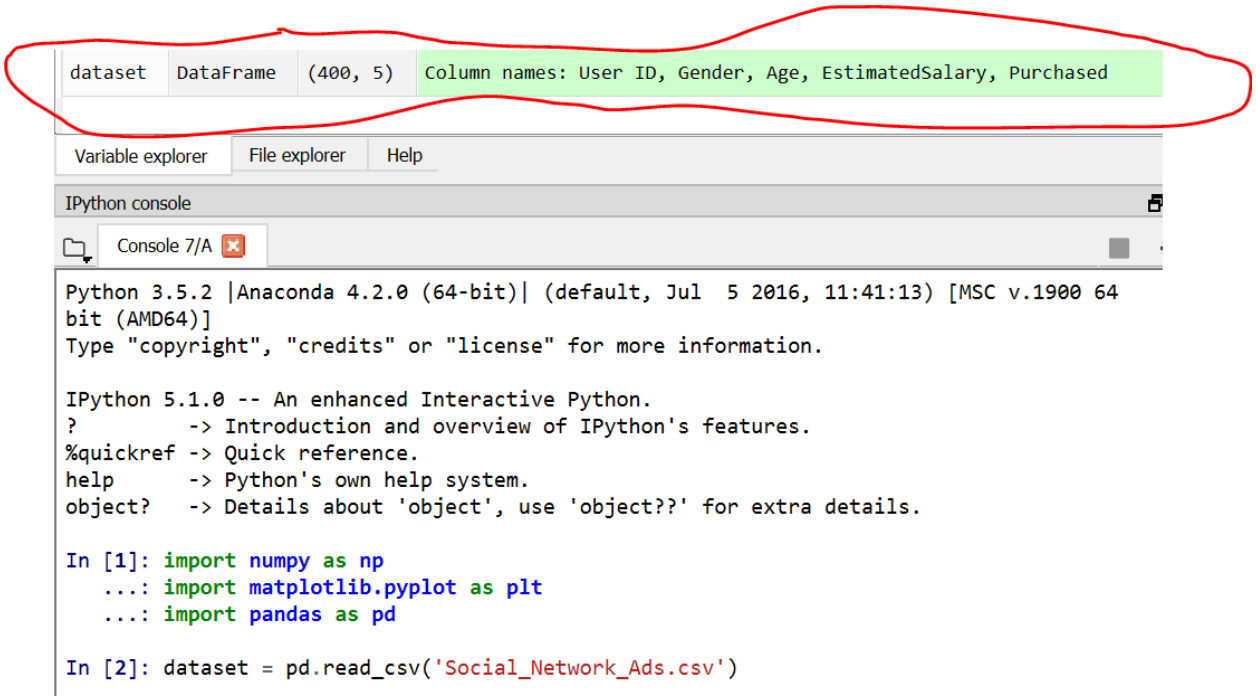
2. Import the libraries
   d) Numpy – math libraries
   e) Matplotlib- library used to plot charts
   f) Pandas – to import and manage datasets

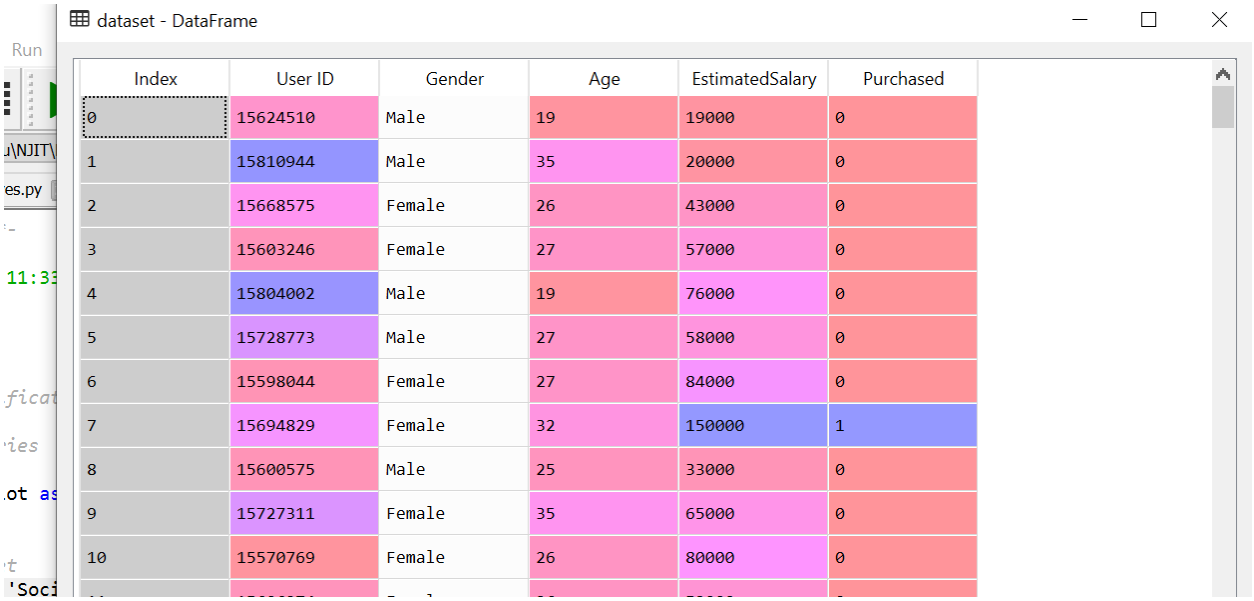
```
Python 3.5.2 |Anaconda 4.2.0 (64-bit)| (default, Jul  5 2016, 11:41:13) [MSC v.1900 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: import pandas as pd
```

3. Import the dataset

   Use pandas to import the dataset

   c) Pd.read_csv('Social_Network_Ads.csv')





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
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0

d) Distinguish matrix of features (independent variables) and dependent variables

c)

```
In [3]: X = dataset.iloc[:, [2, 3]].values
```

– The second and third column Age and Estimated salary will be the independent variables

X - NumPy array

	0	1
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
5	27	58000
6	27	84000
7	32	150000
8	25	33000

d)

```
In [4]: y = dataset.iloc[:, 4].values
```

The fourth column Purchased will be the dependent variable

y - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0

X	int64	(400, 2)	array([[19, 19000], [35, 20000],
dataset	DataFrame	(400, 5)	Column names: User ID, Gender, Age, EstimatedSalary, Purchased
y	int64	(400,)	array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1… 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…

Variable explorer | File explorer | Help

IPython console

Console 11/A

```
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: import pandas as pd
   ...:

In [2]: dataset = pd.read_csv('Social_Network_Ads.csv')

In [3]: X = dataset.iloc[:, [2, 3]].values

In [4]: y = dataset.iloc[:, 4].values

In [5]:
```

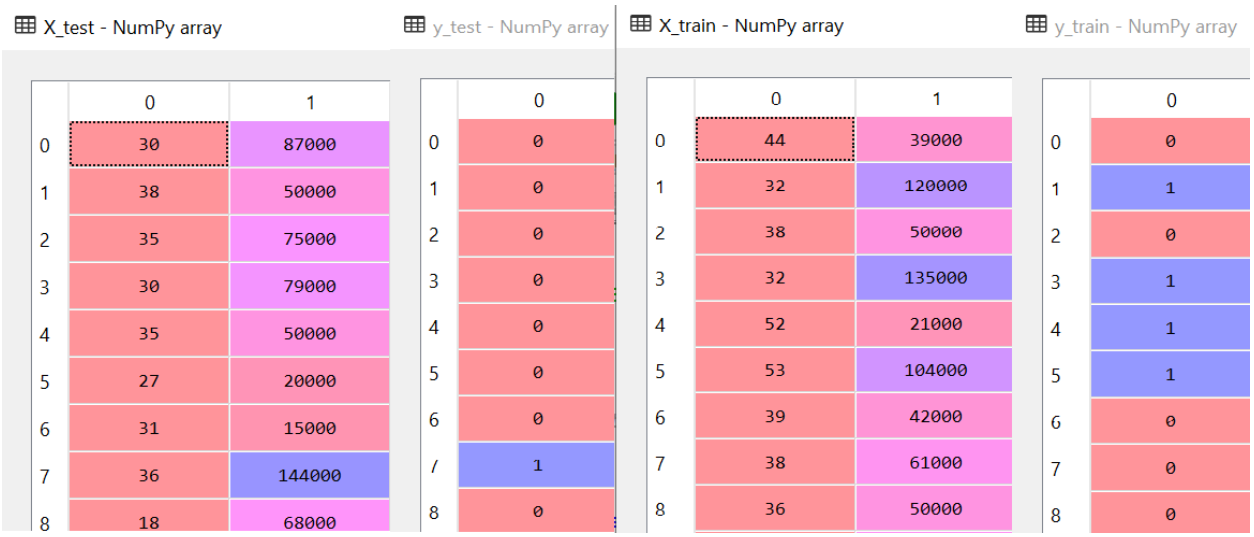
4. Splitting the dataset into training set and test set

   (Using the cross_validation library from sckit-learn)

```
In [5]: from sklearn.cross_validation import train_test_split
   ...: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)
```

   c) Create X_train, X_test, y_train, y_test

   d) Use train_test_split(X,y,test_size=0.25, random_state= 0) – 25 percent of data goes to test set remaining training set , random_state=0 gives same result everytime



X_test - NumPy array

	0	1
0	30	87000
1	38	50000
2	35	75000
3	30	79000
4	35	50000
5	27	20000
6	31	15000
7	36	144000
8	18	68000

y_test - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0

X_train - NumPy array

	0	1
0	44	39000
1	32	120000
2	38	50000
3	32	135000
4	52	21000
5	53	104000
6	39	42000
7	38	61000
8	36	50000

y_train - NumPy array

	0
0	0
1	1
2	0
3	1
4	1
5	1
6	0
7	0
8	0

5. Features Scaling

   Due to difference in scale of age and estimated salary the Euclidean distance won't be measured accurately and hence we need to use feature scaling

   We use the preprocessing library from sckit-learn.org and import Standard Scaler
   Fit_and_transform training set and test set

```
In [6]: from sklearn.preprocessing import StandardScaler
   ...: sc = StandardScaler()
   ...: X_train = sc.fit_transform(X_train)
   ...: X_test = sc.transform(X_test)
```

//for X_test we don't have to use fit since it is already fitted to the training set

X_train - NumPy array

	0	1
0	0.582	-0.887
1	-0.607	1.462
2	-0.013	-0.568
3	-0.607	1.897
4	1.374	-1.409
5	1.473	0.998
6	0.086	-0.800
7	-0.013	-0.249
8	-0.211	-0.568

X_test - NumPy array

	0	1
0	-0.805	0.505
1	-0.013	-0.568
2	-0.310	0.157
3	-0.805	0.273
4	-0.310	-0.568
5	-1.102	-1.438
6	-0.706	-1.583
7	-0.211	2.158
8	-1.993	-0.046

6. Fitting Decision Tree classifier to the training set

Import Decision tree classifier from library tree of sckit-learn.org
We use criterion = ‘entropy’ which makes the child nodes at the root more homogenous and gives better quality results than Naïve Bayes
Fit the classifier to the training set X_train and y_train

```
In [2]: from sklearn.tree import DecisionTreeClassifier
   ...: classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
   ...: classifier.fit(X_train, y_train)
Out[2]:
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=0, splitter='best')
```

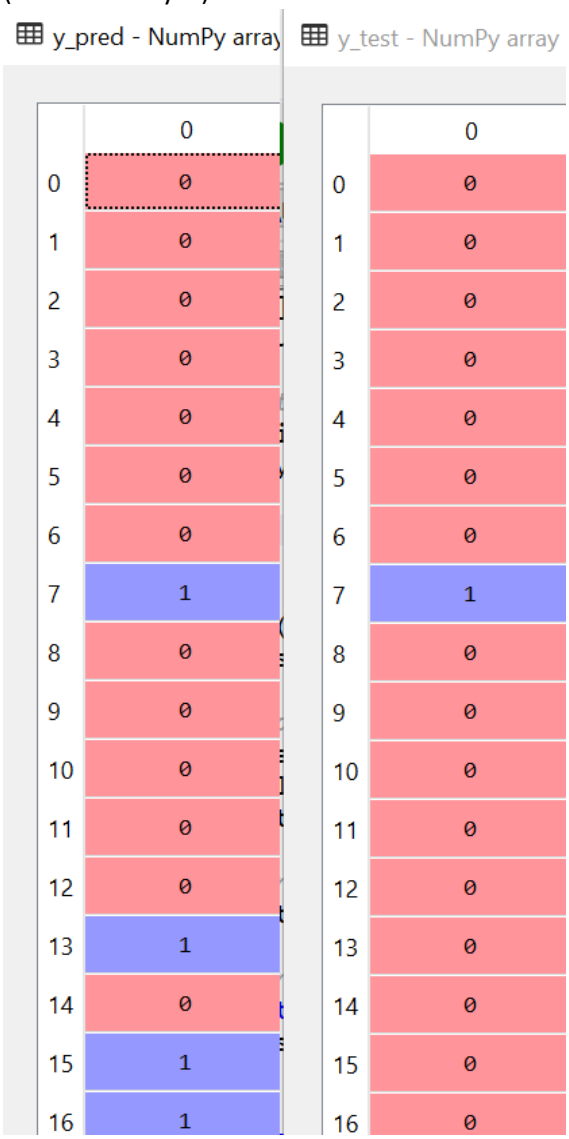
7. Predicting the test set results

Gets the vector of predictions y_pred

```
In [8]: y_pred = classifier.predict(X_test)
   ...:
```

Compare y_pred and y_test we see first 6 predictions are same

(For Naïve Bayes)



	y_pred	y_test
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	1	1
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	1	0
14	0	0
15	1	0
16	1	0

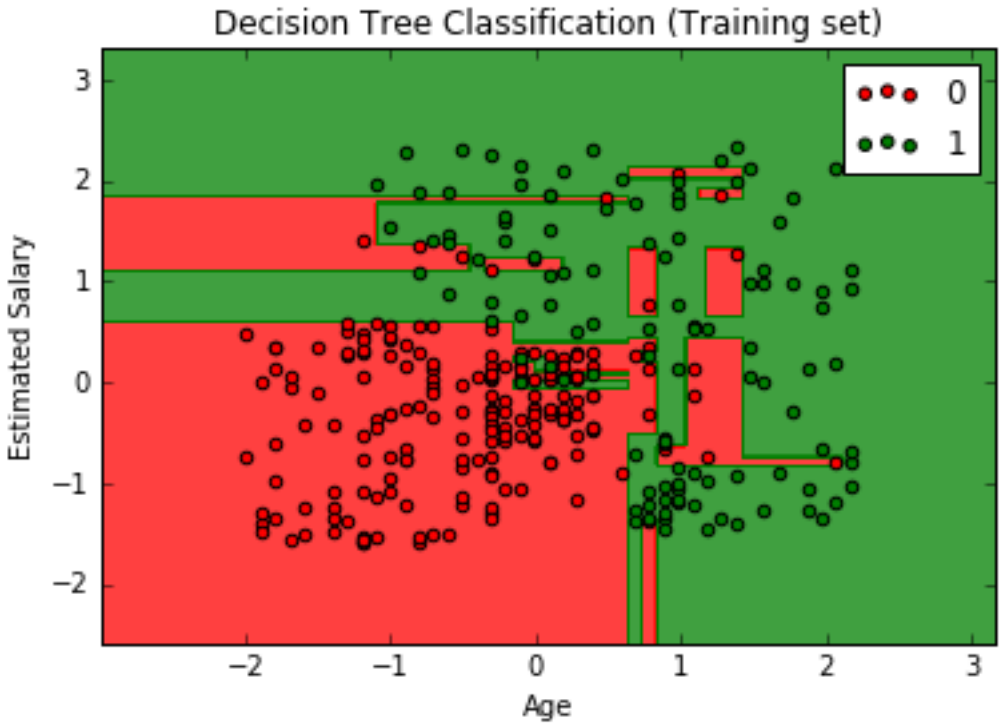
We observe till 12 all predictions are correct but at 13 there is one wrong prediction and so on there are more wrong predictions

8. Making the confusion matrix
   The confusion matrix evaluates the number of correct and incorrect predictions

```
In [4]: from sklearn.metrics import confusion_matrix
   ...: cm = confusion_matrix(y_test, y_pred)

In [5]: cm
Out[5]:
array([[62,  6],
       [ 3, 29]])
```

   We have 62+29=91 correct predictions and 6+3=9 incorrect predictions which is quite good compared to Naïve bayes which is 90 correct predictions and 10 incorrect predictions

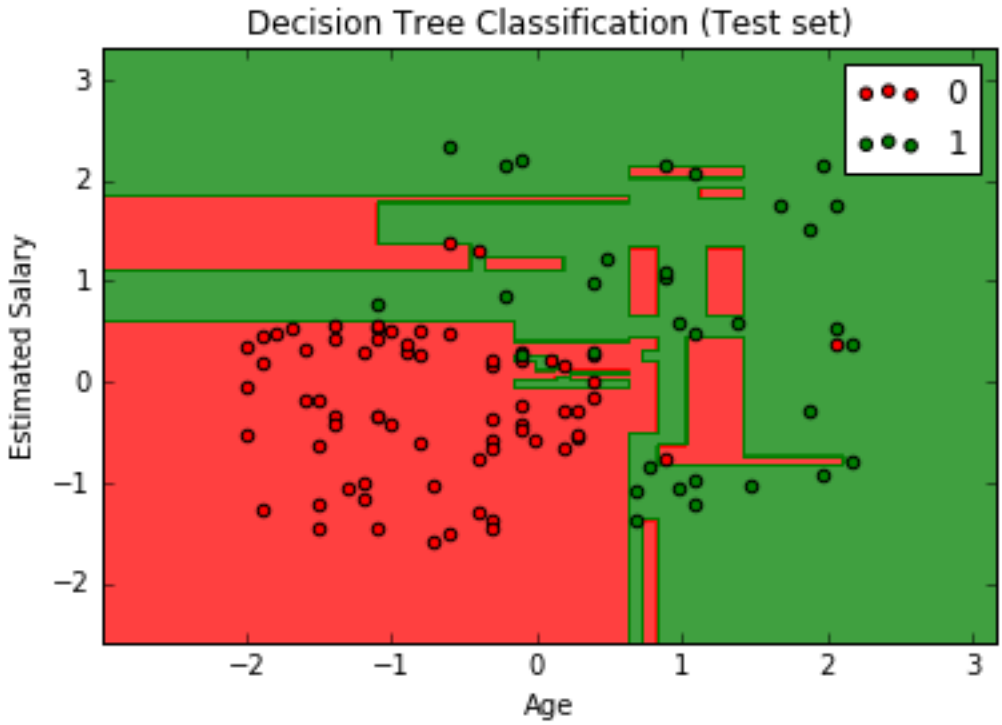
9. Visualization of training set results



This is more accurate than Naïve Bayes because it is trying to catch every user at the right place

The prediction boundary is only horizontal and vertical lines, it is making splits based on the conditions of independent variables i.e age and estimated salary

10. Visualizing the test set results



As we can see there are some green points in the red region and some red points in the green region

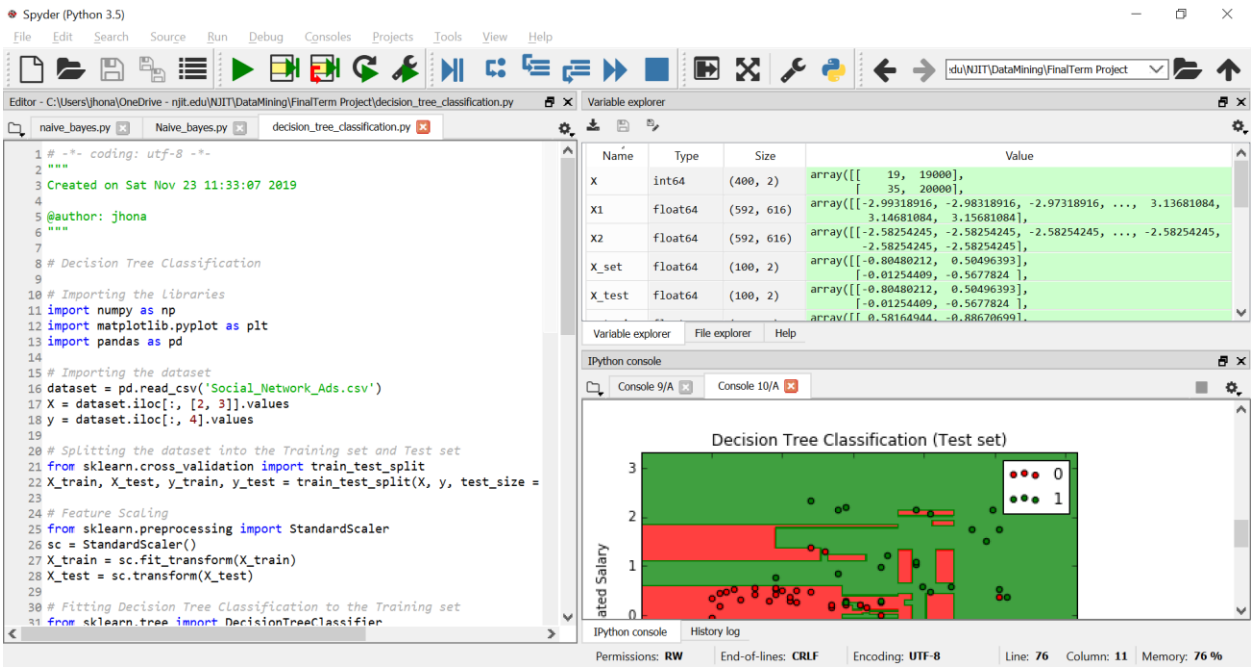
If we count the number of incorrect predictions we will get it to be 9

Also we see some red rectangles that don't seem to have any points in them.

But compared to the Naïve Bayes this seems to be comparatively better.

Screenshot showing the running of Decision Tree Classifier

Running of Program



Summary

1. We have observed two classification algorithms Naïve Bayes and Decision Trees Classification
2. We observe that Naïve Bayes has a curve which divides the points into two regions one where the user is supposed to buy the SUV and another region where the user doesn't buy the SUV
3. Though Naïve Bayes is better compared to linear regression where the graph is a straight line it has comparatively lesser accuracy as compared to Decision Trees
4. In Decision trees the graph gets broken into splits thus covering the points according to the predictions more accurately