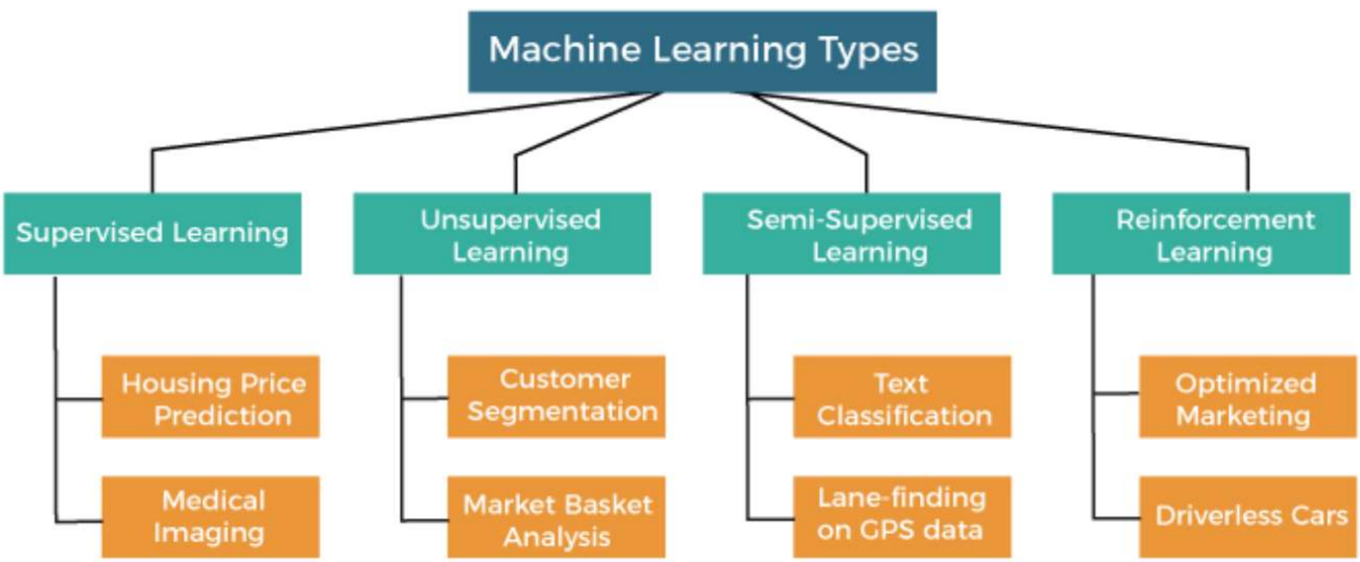
MACHINE LEARNING

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed.

- ***Descriptive analysis*** *tells what happened in your business in the past week, month or year, presenting it as numbers and visuals in reports and dashboards.*
- ***Diagnostic analysis*** *gives the reason why something happened.*
- ***Predictive analysis*** *determines the potential outcomes of present and past actions and trends.*
- ***Prescriptive analysis*** *offers decision support for the best course of action to get desired results.*

Types of Machine Learning :



1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning

Supervised Machine Learning:

Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

- **Classification**
- **Regression**

a) Classification

*Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "**Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.***

- ***Random Forest Algorithm***
- ***Decision Tree Algorithm***
- ***Logistic Regression Algorithm***
- ***Support Vector Machine Algorithm***

b) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

- ***Simple Linear Regression Algorithm***
- ***Decision Tree Algorithm***

UnSupervised Machine Learning:

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences.

Categories of UnSupervised Machine Learning

UnSupervised machine learning can be classified into two types of problems, which are given below:

- ***Clustering***
- ***Association***

a) Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups.

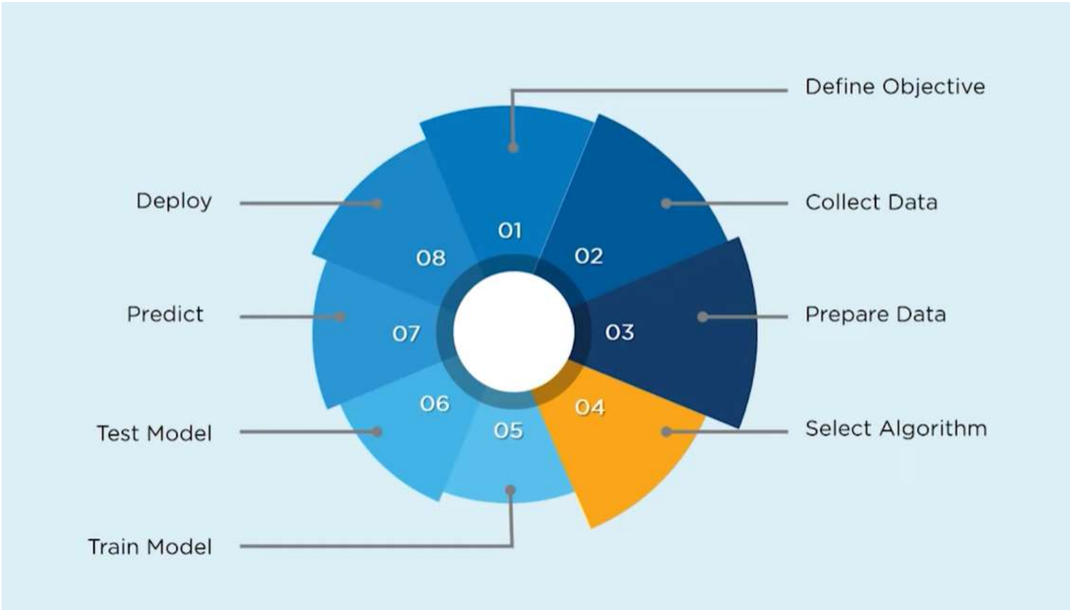
- ***K-Means Clustering algorithm***

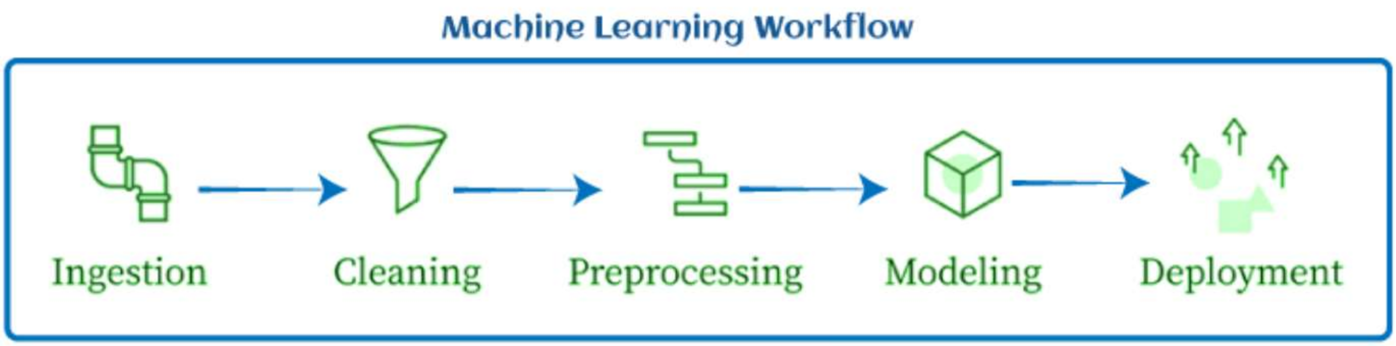
b) Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset.

- ***Apriori algorithm***

Machine Learning Pipeline Process :





All Machine Learning generic steps :

1.Importing the libraries

2.Importing the dataset

3.Taking care of missing Data

4.Encoding Categorical data

5.Encoding Independent and dependent variables

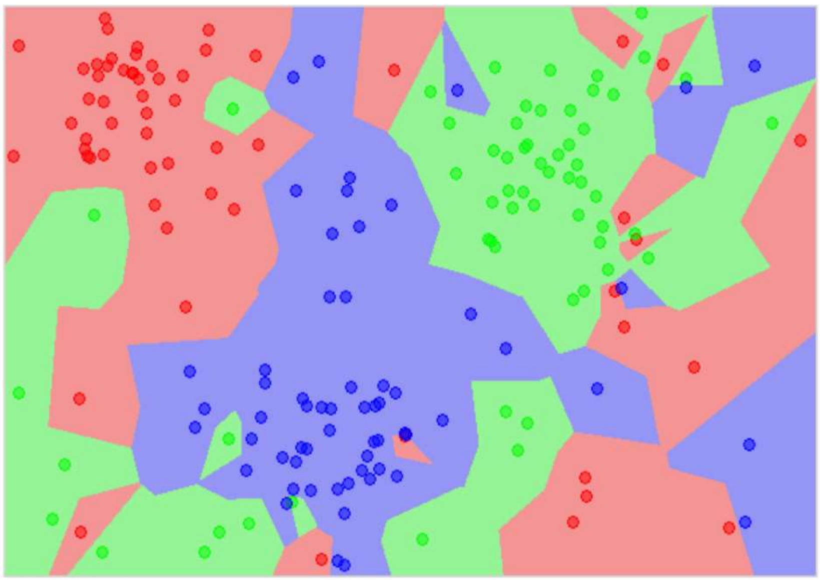
6.Splitting the dataset into Training and test Data

7. Feature Scaling.

k-Nearest Neighbours

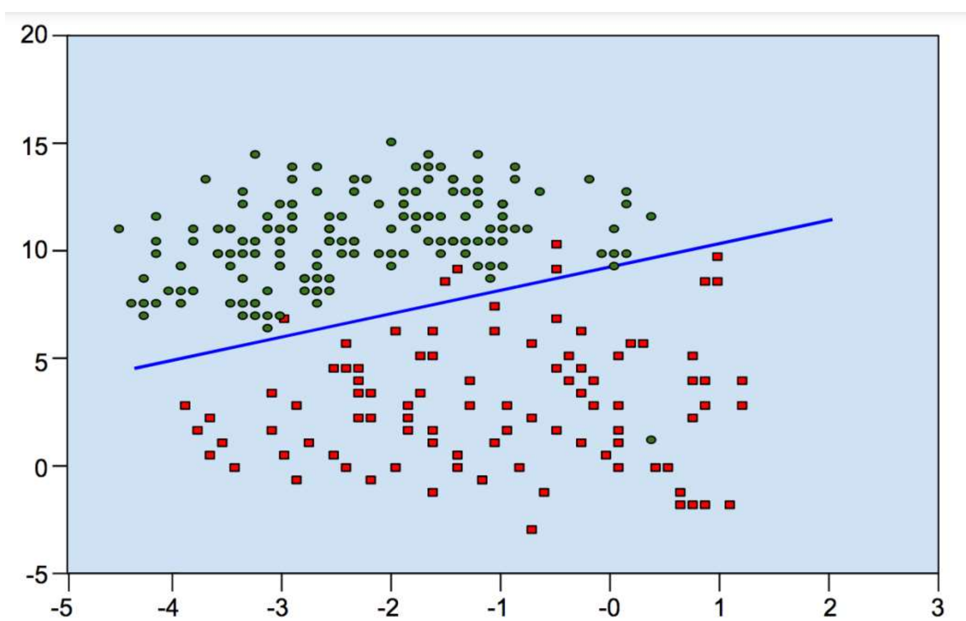
The k-Nearest Neighbours, which is simply called kNN is a statistical technique that can be used for solving for classification and regression problems.

The diagram shows three types of objects, marked in red, blue and green colors. When you run the kNN classifier on the above dataset, the boundaries for each type of object will be marked as shown below:



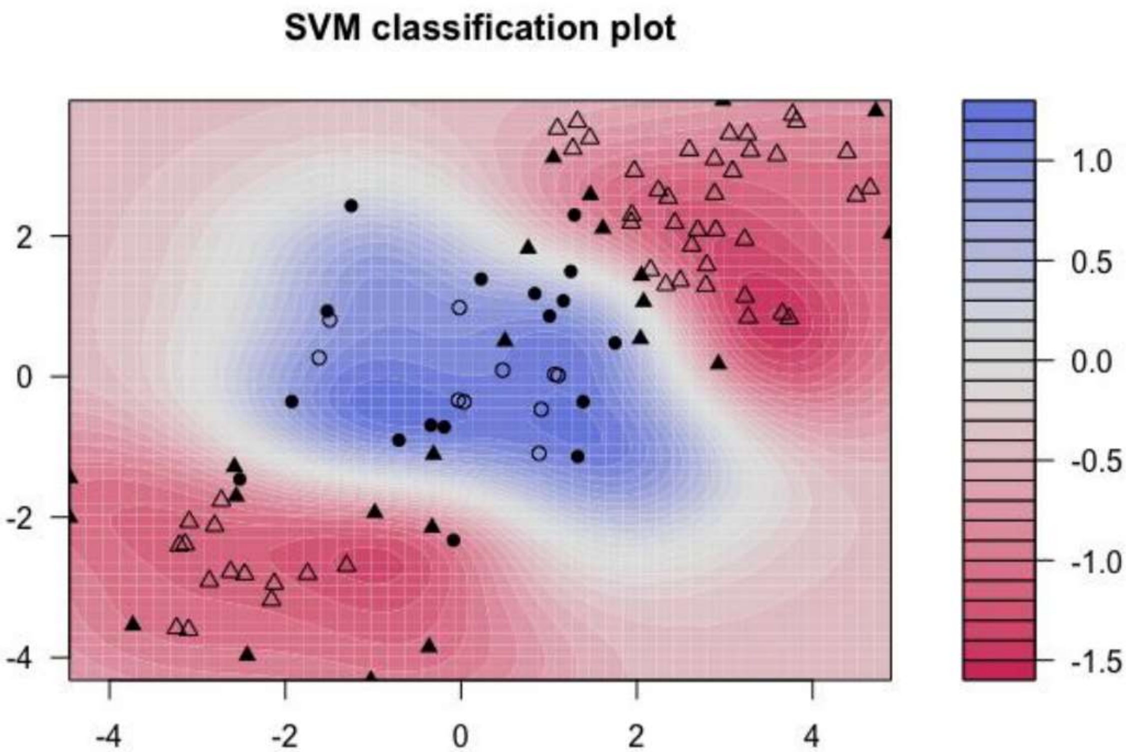
Logistic Regression

From the diagram, we can visually inspect the separation of red dots from green dots. You may draw a boundary line to separate out these dots. Now, to classify a new data point, you will just need to determine on which side of the line the point lies.



Support Vector Machines

The three classes of data cannot be linearly separated. The boundary curves are non-linear. In such a case, finding the equation of the curve becomes a complex job.



The Support Vector Machines (SVM) comes handy in determining the separation boundaries in such situations.

Few Workings on Machine Learning Algorithms :

In [35]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [36]:
```
dataset= pd.read_csv('Salary_Data.csv')
```

In [37]:
```
x = dataset.iloc[:,: -1].values
y = dataset.iloc[:, -1].values
```

In [38]:
```
from sklearn.model_selection import train_test_split
```

In [39]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1/3, random_state=0)
```

In [40]:
```
dataset
```

Out[40]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0

In [42]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
```

Out[42]: LinearRegression()

In [43]:
```
regressor
```

Out[43]: LinearRegression()

In [44]:
```
y_pred = regressor.predict(x_test)
```

In [45]:
```
plt.scatter(x_train,y_train,color='red')
plt.plot(x_train, regressor.predict(x_train),color='blue')
plt.title('Salary vs Experience (Training Set)')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()
```

In [16]:
```
#importing the libraries

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

In [17]:
```
#reading the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
x= dataset.iloc[:, :-1].values
y= dataset.iloc[:, -1].values
```

In [18]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.25,random_state=0)
```

In [19]:
```
print(x_train)
```

```
[[    44  39000]
 [    32 120000]
 [    38  50000]
 [    32 135000]
 [    52  21000]
 [    53 104000]
 [    39  42000]
 [    38  61000]
 [    36  50000]
 [    36  63000]
```

In [20]:
```
dataset
```

Out[20]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
...	...	...	...
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

400 rows × 3 columns

In [21]:
```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train =sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [22]:
```python
print(x_train)
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
 [ 1.37390747 -1.40858358]
 [ 1.47293972  0.99784738]
 [ 0.08648817 -0.79972756]
 [-0.01254409 -0.24885782]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]
 [-0.30964085 -1.29261101]
 [-0.30964085 -0.5677824 ]
 [ 0.38358493  0.09905991]
```

In [23]:
```python
print(x_test)
```

```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]
 [-0.80480212 -0.59677555]
 [-1.00286662 -0.42281668]
 [-0.11157634 -0.42281668]
 [ 0.08648817  0.21503249]
 [-1.79512465  0.47597078]
 [-0.60673761  1.37475825]
 [-0.11157634  0.21503249]
 [-1.89415691  0.44697764]
 [ 1.67100423  1.75166912]
 [-0.30964085 -1.37959044]
 [-0.30964085 -0.65476184]
 [ 0.8787462   2.15757314]
```

In [26]:
```
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)
```

Out[26]: LogisticRegression(random_state=0)

In [28]:
```
print(classifier.predict(sc.transform([[30,87000]])))
```

```
[0]
```

In [30]:
```
y_pred = classifier.predict(x_test)
```

In [31]:
```
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1))
```

```
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
```

In [32]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [33]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

In [34]:
```
print(cm)
```

```
[[65  3]
 [ 8 24]]
```

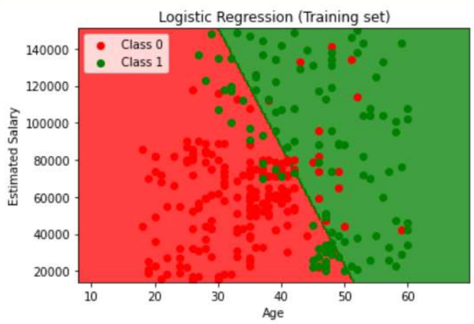
In [52]:
```
accuracy_score(y_test,y_pred)
```

Out[52]: 0.89

In [55]:

```
from matplotlib.colors import ListedColormap

x_set, y_set = sc.inverse_transform(x_train), y_train
x1, x2 = np.meshgrid(
    np.arange(start=x_set[:,0].min() - 10, stop=x_set[:,0].max() + 10, step=0.25),
    np.arange(start=x_set[:,1].min() - 1000, stop=x_set[:,1].max() + 1000, step=0.25))

plt.contourf(x1, x2, classifier.predict(sc.transform(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())  # Corrected y-axis limit line

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='Class ' + str(j))

plt.title('Logistic Regression (Training set)')  # Moved title before xlabel
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()  # Moved outside the loop
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you
intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you
intend to specify the same RGB or RGBA value for all points.
```



In [57]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier


dataset = pd.read_csv('Social_Network_Ads.csv')
x= dataset.iloc[:, :-1].values
y= dataset.iloc[:, -1].values
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.25,random_state=0)

sc = StandardScaler()
x_train =sc.fit_transform(x_train)
x_test = sc.transform(x_test)

classifier =DecisionTreeClassifier(random_state = 0)
classifier.fit(x_train, y_train)

print(classifier.predict(sc.transform([[30,87000]])))
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1))

from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
```

```
[0]
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]]
[[62  6]
 [ 4 28]]
```

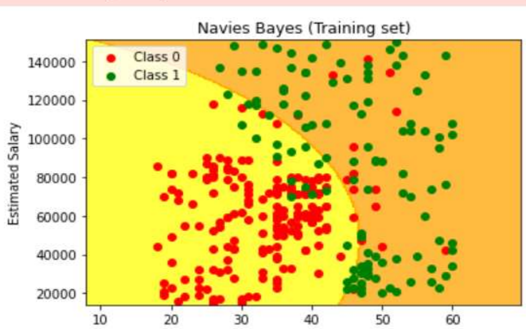
Out[57]: 0.9

In [58]:
```python
from matplotlib.colors import ListedColormap

x_set, y_set = sc.inverse_transform(x_train), y_train
x1, x2 = np.meshgrid(
    np.arange(start=x_set[:,0].min() - 10, stop=x_set[:,0].max() + 10, step=0.25),
    np.arange(start=x_set[:,1].min() - 1000, stop=x_set[:,1].max() + 1000, step=0.25))

plt.contourf(x1, x2, classifier.predict(sc.transform(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='Class ' + str(j))

plt.title('DecisionTree (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```

In [62]:
```python
from matplotlib.colors import ListedColormap

x_set, y_set = sc.inverse_transform(x_train), y_train
x1, x2 = np.meshgrid(
    np.arange(start=x_set[:,0].min() - 10, stop=x_set[:,0].max() + 10, step=0.25),
    np.arange(start=x_set[:,1].min() - 1000, stop=x_set[:,1].max() + 1000, step=0.25))

plt.contourf(x1, x2, classifier.predict(sc.transform(np.array([x1.ravel(), x2.ravel()]).T)).reshape(x1.shape),
             alpha=0.75, cmap=ListedColormap(('yellow', 'orange')))
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='Class ' + str(j))

plt.title('Navies Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

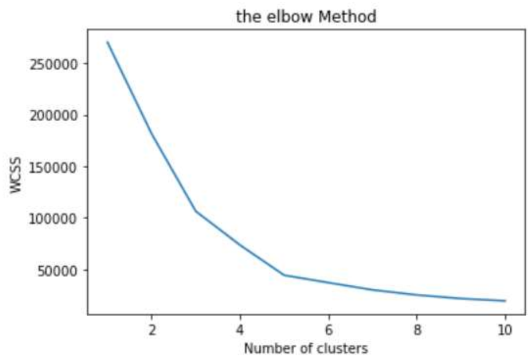
```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```



In [71]:
```
import pandas as pd
import matplotlib.pyplot as mtp
import numpy as np
```

In [68]:
```
dataset = pd.read_csv('Mall_Customers.csv')
x= dataset.iloc[:, [3,4]].values
```

In [69]:
```
from sklearn.cluster import KMeans
wcss =[]

for i in range(1,11):
    kmeans =KMeans(n_clusters = i ,init = 'k-means++',random_state =42)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title('the elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [72]:
```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
```

In [74]:
```
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second cluster
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third cluster
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth cluster
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth cluster
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```