Naive Bayes Classifier

**Short note about Naïve Bayes** :

**In** [**machine learning**](https://en.wikipedia.org/wiki/Machine_learning)**, naive Bayes classifiers** are a family of simple [probabilistic classifiers](https://en.wikipedia.org/wiki/Probabilistic_classifier) based on applying [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with strong (naive) [independence](https://en.wikipedia.org/wiki/Statistical_independence) assumptions between the features.

A **Probabilistic classifier** is a [classifier](https://en.wikipedia.org/wiki/Statistical_classification) that is able to predict, given an observation of an input, a [probability distribution](https://en.wikipedia.org/wiki/Probability_distribution) over a [set](https://en.wikipedia.org/wiki/Set_(mathematics)) of classes, rather than only outputting the most likely class that the observation should belong to. Probabilistic classifiers provide classification that can be useful in its own right or when combining classifiers into [ensembles](https://en.wikipedia.org/wiki/Ensemble_classifier).

**classification** is the problem of identifying to which of a set of [categories](https://en.wikipedia.org/wiki/Categorical_data) (sub-populations) a new [observation](https://en.wikipedia.org/wiki/Observation) belongs, on the basis of a [training set](https://en.wikipedia.org/wiki/Training_set) of data containing observations (or instances) whose category membership is known.ex- ordinary" classifier-y=f(x) ,predicting the values of x based on y.

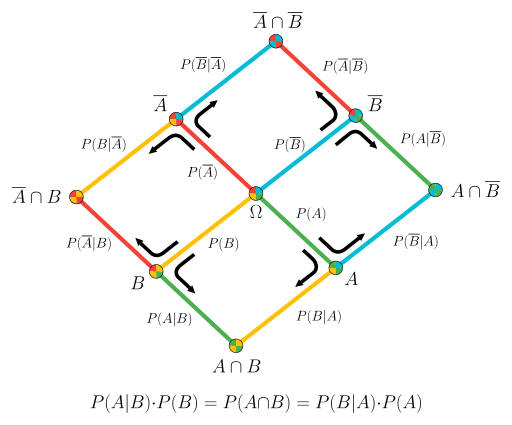
**Ensemble methods** use multiple learning algorithms to obtain better [predictive performance](https://en.wikipedia.org/wiki/Predictive_inference) than could be obtained from any of the constituent learning algorithms alone.

There are three types of Naive Bayes model under scikit learn library:

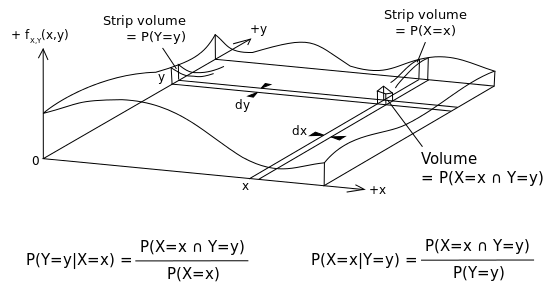
* [**Gaussian:**](http://scikit-learn.org/stable/modules/naive_bayes.html)It is used in classification and it assumes that features follow a normal distribution.
* [**Multinomial**](http://scikit-learn.org/stable/modules/naive_bayes.html)**:**It is used for discrete counts. For example, let’s say, we have a text classification problem. Here we can consider bernoulli trials which is one step further and instead of “word occurring in the document”, we have “count how often word occurs in the document”, you can think of it as “number of times outcome number x\_i is observed over the n trials”.
* [**Bernoulli**](http://scikit-learn.org/stable/modules/naive_bayes.html)**:**The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with ‘bag of words’ model where the 1s & 0s are “word occurs in the document” and “word does not occur in the document” respectively.

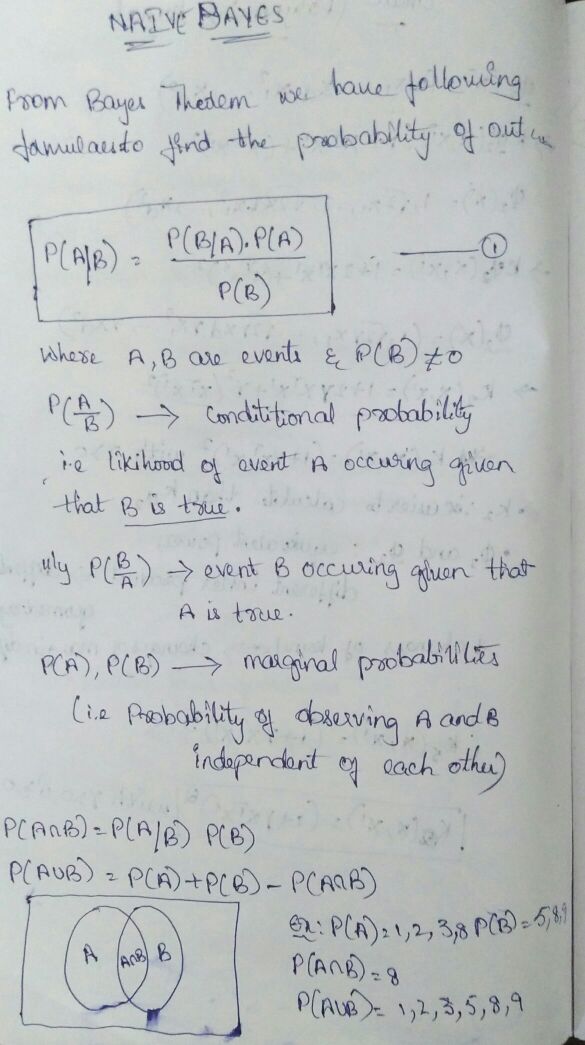
Decode Complex Algorithm

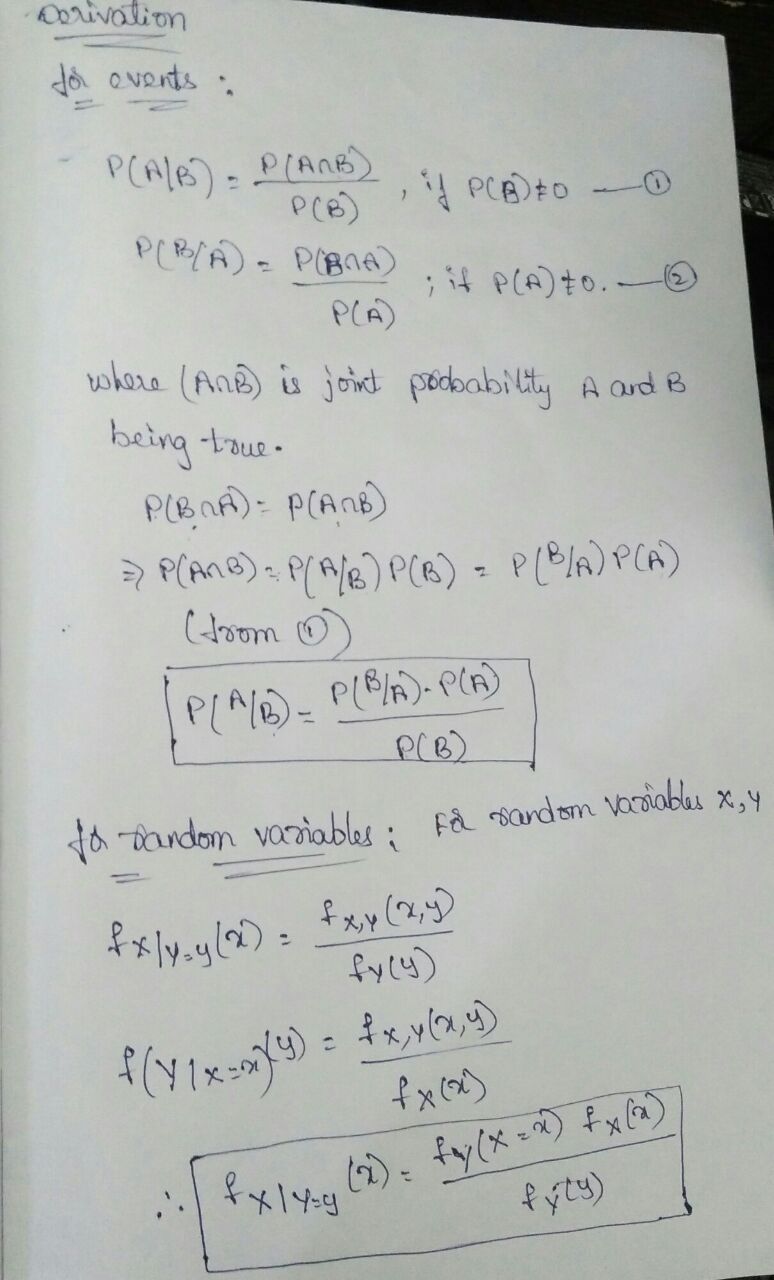
For Events-



Random Variables







**Step: 1**

D1= value rolled on [die](https://en.wikipedia.org/wiki/Dice) 1.  

D2= the value rolled on [die](https://en.wikipedia.org/wiki/Dice) 2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table 1** |  |  |  |  |  |  |  |
| **+** |  | **D2** |  |  |  |  |  |
|  |  | 1 | 2 | 3 | 4 | 5 | 6 |
| **D1** | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|  | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

**Dice Table**

**Step-2 : probability that D1 = 2?**

Table 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** | | **D2** | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** |
| **D1** | **1** | 2 | 3 | 4 | 5 | 6 | 7 |
| **2** | 3 | 4 | 5 | 6 | 7 | 8 |
| **3** | 4 | 5 | 6 | 7 | 8 | 9 |
| **4** | 5 | 6 | 7 | 8 | 9 | 10 |
| **5** | 6 | 7 | 8 | 9 | 10 | 11 |
| **6** | 7 | 8 | 9 | 10 | 11 | 12 |

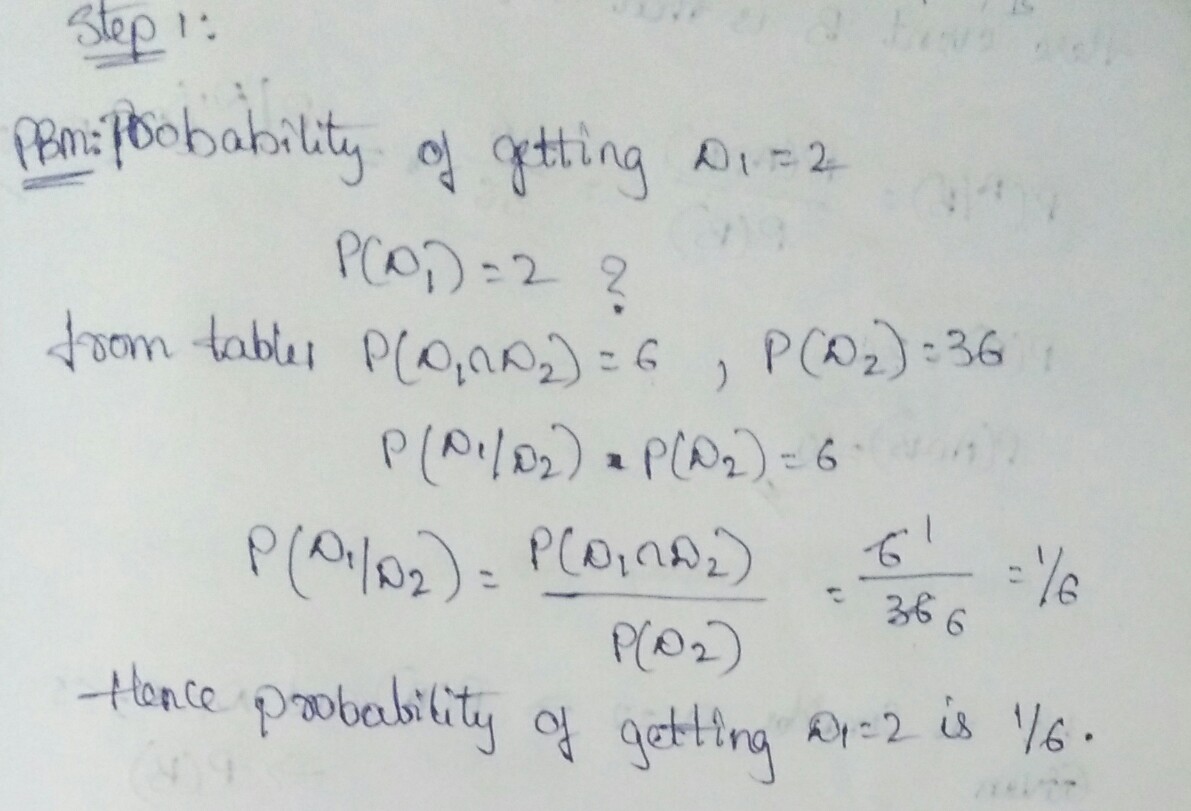
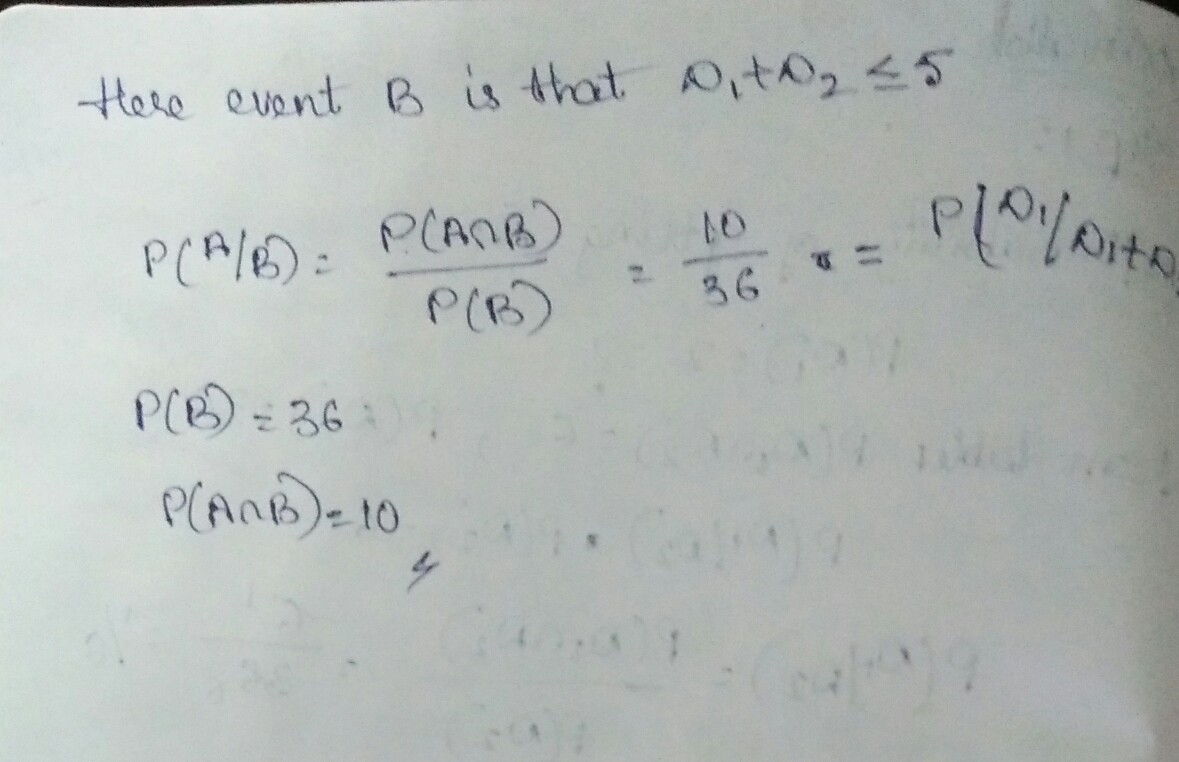
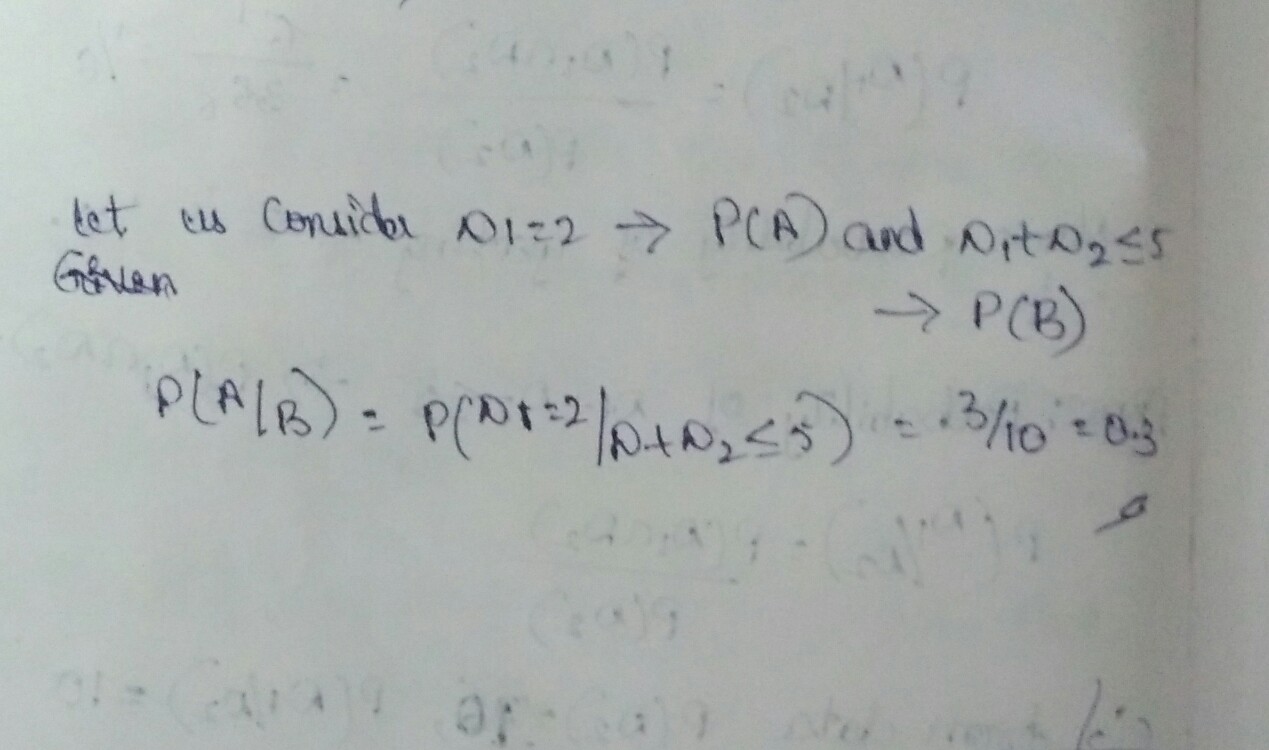
****

Table 2

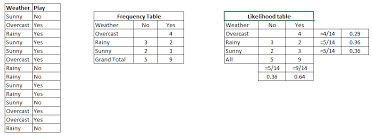
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **D2** | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** |
| **D1** | **1** | 2 | 3 | 4 | 5 | 6 | 7 |
| **2** | 3 | 4 | 5 | 6 | 7 | 8 |
| **3** | 4 | 5 | 6 | 7 | 8 | 9 |
| **4** | 5 | 6 | 7 | 8 | 9 | 10 |
| **5** | 6 | 7 | 8 | 9 | 10 | 11 |
| **6** | 7 | 8 | 9 | 10 | 11 | 12 |

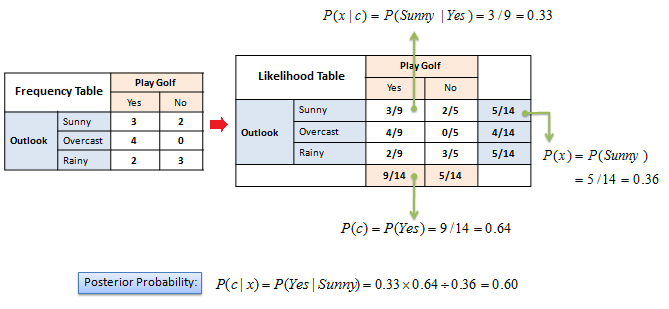
****

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table 3 | | | | | | | |
| **+** | | **D2** | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** |
| **D1** | **1** | 2 | 3 | 4 | 5 | 6 | 7 |
| **2** | 3 | 4 | 5 | 6 | 7 | 8 |
| **3** | 4 | 5 | 6 | 7 | 8 | 9 |
| **4** | 5 | 6 | 7 | 8 | 9 | 10 |
| **5** | 6 | 7 | 8 | 9 | 10 | 11 |
| **6** | 7 | 8 | 9 | 10 | 11 | 12 |

****

**Nayes Bias-Random Process**





**Use Cases of Naive Bayes algorithm**

**Real time Prediction**: Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.

**Multi class Prediction**: This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.

**Text classification/ Spam Filtering/ Sentiment Analysis**: Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)

**Recommendation System**: Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

**Python**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('C:\\Users\\Rama\\Desktop\\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)

#

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

#fitting classifier to training set

from sklearn.naive\_bayes import GaussianNB

classifier=GaussianNB

classifier.fit=(X\_train,y\_train)

#create your classifier here

#

# Fitting K-NN Regression to the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

#

# Predicting the Test set results

from sklearn.linear\_model import LinearRegression

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('Nayes Bias (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('Nayes Bias (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

