Vashu Agarwal

E21CSEU0054

Lab5

In [22]:

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

# Task 1 : Importing the dataset


dataset = pd.read_csv('/Users/vashuagarwal/Downloads/BENNETT  thing
```

In [63]:

```
# Task 2 : print the names of the first 13 feature
dataset.head()
```

Out[63]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulpha
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0

In [64]:

```

dataset = dataset.dropna()
```

In [ ]:

In [ ]:

In [65]:

```
X = dataset.iloc[:, [2, 10]].values
y = dataset.iloc[:, 12].values
```

In [66]:

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split


# Task 3:  train the model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(4847, 2) (1616, 2) (4847,) (1616,)
```

In [ ]:

In [67]:

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [68]:

```
# Task 4: Fitting Naive Bayes to the Training set

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[68]:

```
GaussianNB()
```

In [69]:

```
# Task 5: Predicting the Test set results

y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[6 6 6 ... 6 7 6]
```

In [70]:

```
# Task 6 : Making the Confusion Matrix
from sklearn.metrics import confusion_matrix


cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[  1   0   3   5   0   0   0]
 [  1   1  14  29   0   0   0]
 [  1  10 142 379   4   0   0]
 [  0   2 101 609  13   0   0]
 [  0   0  21 240   5   0   0]
 [  0   0   3  30   1   0   0]
 [  0   0   0   1   0   0   0]]
```

In [71]:

```
# Task 7 : Making the Classification report

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           3       0.33      0.11      0.17         9
           4       0.08      0.02      0.03        45
           5       0.50      0.26      0.35       536
           6       0.47      0.84      0.60       725
           7       0.22      0.02      0.03       266
           8       0.00      0.00      0.00        34
           9       0.00      0.00      0.00         1

    accuracy                           0.47      1616
   macro avg       0.23      0.18      0.17      1616
weighted avg       0.42      0.47      0.39      1616


/Users/vashuagarwal/opt/anaconda3/lib/python3.8/site-packages/skle
arn/metrics/_classification.py:1221: UndefinedMetricWarning: Preci
sion and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [72]:

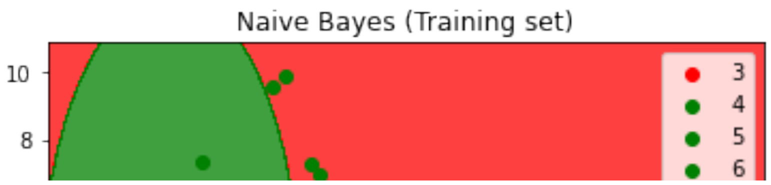
```
# Task 8 : Making the Classification accuracy score
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

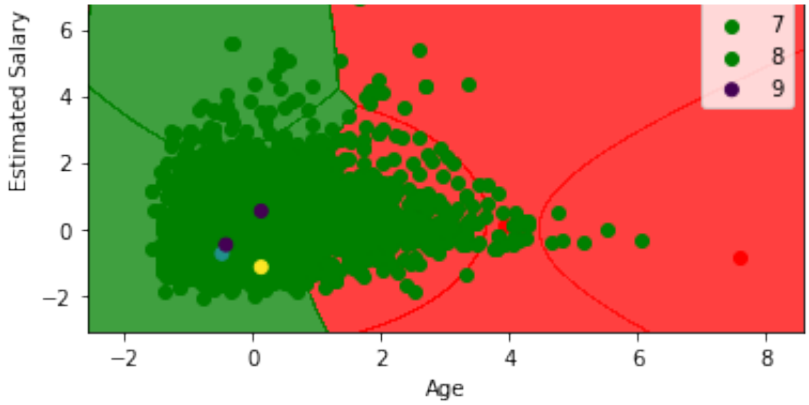
```
Accuracy :  0.46905940594059403
```

In [73]:

```
# 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
                     np.arange(start = X_set[:, 1].min() - 1, stop
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
             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()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
```



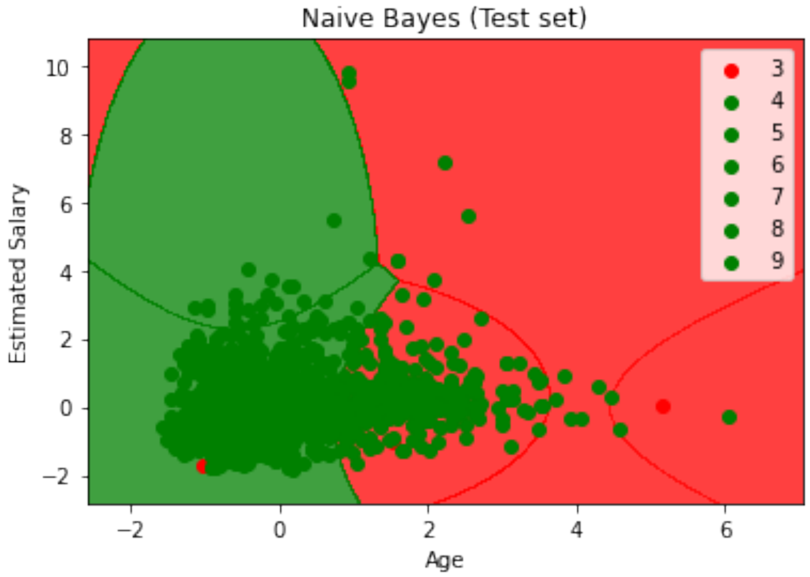


In [74]:

```
# 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
                     np.arange(start = X_set[:, 1].min() - 1, stop
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ra
             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()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D 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, whi
ch 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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
```



In [ ]:

In [ ]:

In [ ]: