SOCIAL INFORMATION NETWORKING PROJECT CODE (Victims of murder)

Name: Om Ashish Mishra

Registration Number: 16BCE0789

Slot: A2

# ML

Classification & Regression

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cross\_validation import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

from matplotlib.colors import ListedColormap

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

# Importing the dataset

dataset = pd.read\_csv('VICTIM\_OF\_MURDER\_0.csv')

X = dataset.iloc[:, [3,9]].values

y = dataset.iloc[:, 1].values

# Splitting the dataset into the Training set and Test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

# Feature Scaling

sc = StandardScaler()

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

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

# Fitting Logistic Regression to the Training set

classifier = LogisticRegression(random\_state = 0)

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

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Logistic Regression (Training set)')

plt.xlabel('Months')

plt.ylabel('Estimated Suicide Ratio')

plt.legend()

plt.show()

# Visualising the Test set results

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Logistic Regression (Test set)')

plt.xlabel('months')

plt.ylabel('Estimated suicide ratio')

plt.legend()

plt.show()

# Fitting Decision Tree Classification to the Training set

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

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Decision Tree Classification (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Decision Tree Classification (Test set)')

plt.xlabel('month')

plt.ylabel('Estimated suicide')

plt.legend()

plt.show()

# Fitting SVM to the Training set

classifier = SVC(kernel = 'linear', random\_state = 0)

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

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('SVM (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Visualising the Test set results

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('SVM (Test set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Fitting Naive Bayes to the Training set

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

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

plt.title('Naive Bayes (Training set)')

plt.xlabel('Month')

plt.ylabel('Estimated Suicide')

plt.legend()

plt.show()

# Visualising the Test set results

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612')))

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', 'blue','green','yellow','orange','pink','black','purple','silver','#FF0923','#672967','#452867','#C78234','#387612'))(i), label = j)

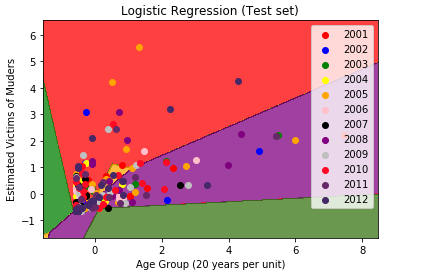
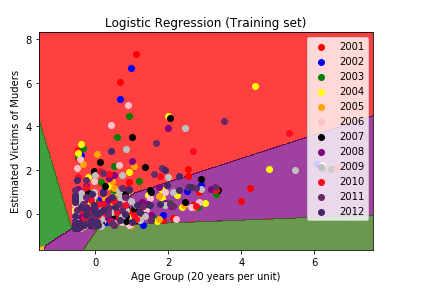
plt.title('Naive Bayes (Test set)')

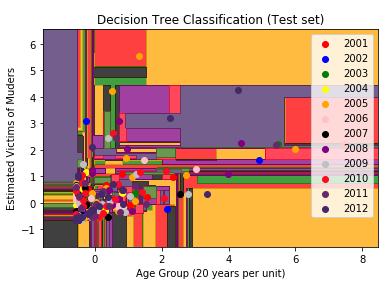
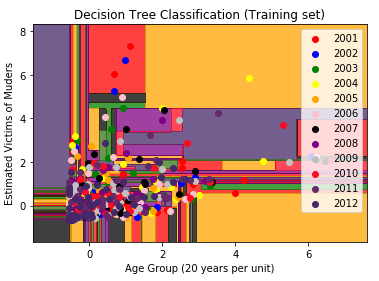
plt.xlabel('Month')

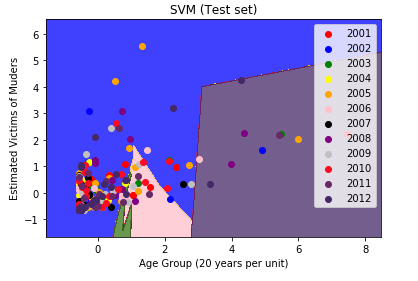
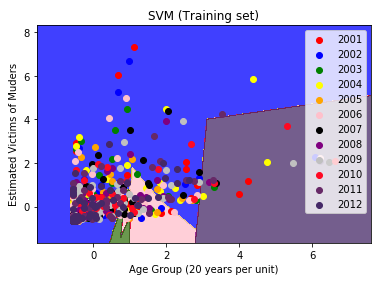
plt.ylabel('Estimated Suicide')

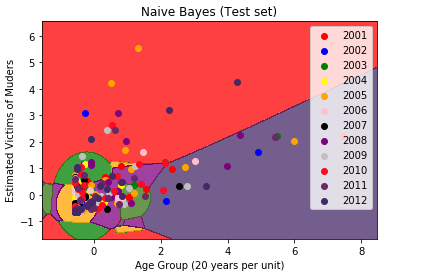
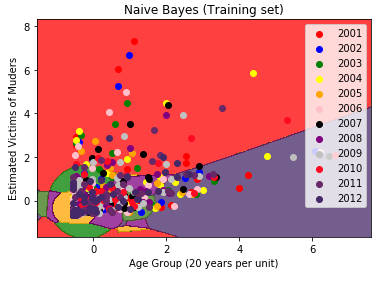
plt.legend()

plt.show()









Clustering:

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.cluster import KMeans

import scipy.cluster.hierarchy as sch

from sklearn.cluster import AgglomerativeClustering

# Importing the dataset

dataset = pd.read\_csv('VICTIM\_OF\_MURDER\_0.csv')

X = dataset.iloc[:, [3,9]].values

# Using the elbow method to find the optimal number of clusters

wcss = []

for i in range(1, 15):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 15), wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

# Fitting K-Means to the dataset

kmeans = KMeans(n\_clusters = 14, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

# Visualising the clusters

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1 - 2001')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2 - 2002')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3 - 2003')

plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4 - 2004')

plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5 - 2005')

plt.scatter(X[y\_kmeans == 5, 0], X[y\_kmeans == 5, 1], s = 100, c = 'pink', label = 'Cluster 6 - 2006')

plt.scatter(X[y\_kmeans == 6, 0], X[y\_kmeans == 6, 1], s = 100, c = 'orange', label = 'Cluster 7 - 2007')

plt.scatter(X[y\_kmeans == 7, 0], X[y\_kmeans == 7, 1], s = 100, c = 'purple', label = 'Cluster 8 - 2008')

plt.scatter(X[y\_kmeans == 8, 0], X[y\_kmeans == 8, 1], s = 100, c = 'black', label = 'Cluster 9 - 2009')

plt.scatter(X[y\_kmeans == 9, 0], X[y\_kmeans == 9, 1], s = 100, c = '#FD4567', label = 'Cluster 10 - 2010')

plt.scatter(X[y\_kmeans == 10, 0], X[y\_kmeans == 10, 1], s = 100, c = 'violet', label = 'Cluster 11 - 2011')

plt.scatter(X[y\_kmeans == 11, 0], X[y\_kmeans == 11, 1], s = 100, c = 'grey', label = 'Cluster 12 - 2012')

plt.scatter(X[y\_kmeans == 12, 0], X[y\_kmeans == 12, 1], s = 100, c = 'brown', label = 'Cluster 13 - 2013')

plt.scatter(X[y\_kmeans == 13, 0], X[y\_kmeans == 13, 1], s = 100, c = 'gold', label = 'Cluster 14 - 2014')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 50, c = 'yellow', label = 'Centroids')

plt.title('Clusters of Victims of Murder')

plt.xlabel('Number of Age')

plt.ylabel('Number of People')

plt.legend()

plt.show()

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('People')

plt.ylabel('Euclidean distances')

plt.show()

# Fitting Hierarchical Clustering to the dataset

hc = AgglomerativeClustering(n\_clusters = 14, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(X)

# Visualising the clusters

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1- 2001')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2 - 2002')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3 - 2003')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4 - 2004')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5 - 2005')

plt.scatter(X[y\_hc == 5, 0], X[y\_hc == 5, 1], s = 100, c = 'yellow', label = 'Cluster 6- 2006')

plt.scatter(X[y\_hc == 6, 0], X[y\_hc == 6, 1], s = 100, c = 'orange', label = 'Cluster 7 - 2007')

plt.scatter(X[y\_hc == 7, 0], X[y\_hc == 7, 1], s = 100, c = 'pink', label = 'Cluster 8 - 2008')

plt.scatter(X[y\_hc == 8, 0], X[y\_hc == 8, 1], s = 100, c = 'gold', label = 'Cluster 9 - 2009')

plt.scatter(X[y\_hc == 9, 0], X[y\_hc == 9, 1], s = 100, c = 'violet', label = 'Cluster 10 - 2010')

plt.scatter(X[y\_hc == 10, 0], X[y\_hc == 10, 1], s = 100, c = 'purple', label = 'Cluster 11 - 2011')

plt.scatter(X[y\_hc == 11, 0], X[y\_hc == 11, 1], s = 100, c = 'grey', label = 'Cluster 12 - 2012')

plt.scatter(X[y\_hc == 12, 0], X[y\_hc == 12, 1], s = 100, c = 'black', label = 'Cluster 13 - 2013')

plt.scatter(X[y\_hc == 13, 0], X[y\_hc == 13, 1], s = 100, c = 'brown', label = 'Cluster 14 - 2014')

plt.title('Clusters of Victims of Murder')

plt.xlabel('Number of Age')

plt.ylabel('Number of People')

plt.legend()

plt.show()

