

ITESO
DEPARTAMENTO DE MATEMÁTICAS Y FÍSICA

Asignatura: Ciencia de Datos e Inteligencia de Negocios

EXAMEN (Clustering y Clasificación)

Nombre: Eduardo Castillo Martínez

Lea detenidamente los reactivos y responda con claridad. Si se requiere hacer uso de más hojas para la realización de cálculos, es necesario que se adjunten a este cuando se haga entrega del examen.

1. (3 puntos) En un experimento se lograron identificar 5 variables importantes que eran las que determinaban el comportamiento del mismo. Después de hacer muchos experimentos se lograron recoger las muestras de diferentes condiciones de trabajo y se encuentran en el archivo *dataset_1.csv*. Determine cuantos grupos o patrones se encuentran en los datos recopilados y justifique su respuesta con código, figuras o mediciones.

```
5 @author: Edu
6 """
7 ### ejercicio 1
8 import numpy as np
9 import pandas as pd
10 import matplotlib.pyplot as plt
11 from scipy.cluster import hierarchy
12 from sklearn.cluster import KMeans
13 ###
14 data_file='dataset_1.csv'
15 data=pd.read_csv(data_file,header=0)
16
17 ### hierarchical clustering
18
19 k=hierarchy.linkage(data,'complete')
20 plt.figure(figsize=(25,10))
21 plt.title('dendograma completo')
22 plt.ylabel('distancia')
23 plt.xlabel('indice de la muestra')
24 dk=hierarchy.dendrogram(k)
25 plt.show()
26
27 ### gráfica de 'codo'
28 last=k[-10:,2]
29 last_rev=last[::-1]
30 idxs=np.arange(1,len(last_rev)+1)
31 plt.plot(idxs,last_rev)
32
33 aceleracion=np.diff(last)
34 aceleracion_rev=aceleracion[::-1]
35 plt.plot(idxs[:-1]+1,aceleracion_rev)
36 plt.show()
37
38 ### algoritmo kmeans
39
40 inercia = np.zeros(15)
41 for k in np.arange(1,15):
42     model = KMeans(n_clusters = k,
43                   init = "random",
44                   max_iter = 300,
45                   n_init = 10,
46                   n_jobs = 1)
47
48     model = model.fit(data)
49     inercia[k] = model.inertia_
50
51 plt.plot(np.arange(1,15), inercia[1:])
52
53 aceleration = np.diff(inercia)
54 plt.plot(np.arange(2,15),-1*aceleration[1:])
55 plt.show()
```

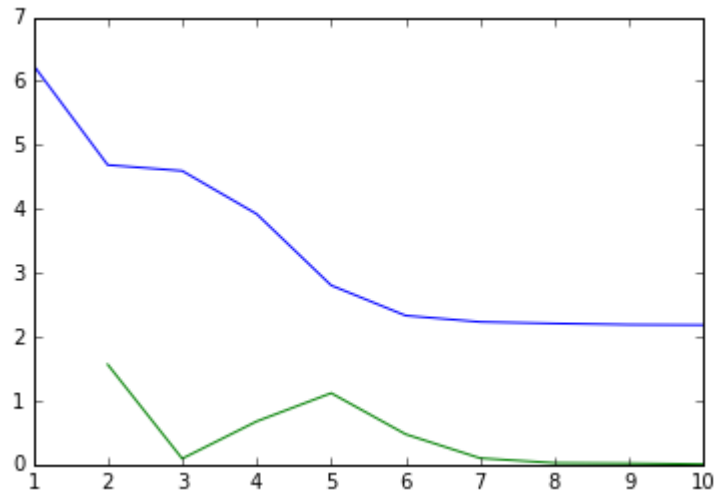


Gráfico de "codo" algoritmo hierarchical clustering.

En el gráfico de codo obtenido con hierarchical clustering(línea azul) y la aceleración (línea verde) se puede observar que hay puntos clave para 5 y 6 grupos, con esta información se puede optar por 5 o 6 grupos, más de 6 no tendría caso.

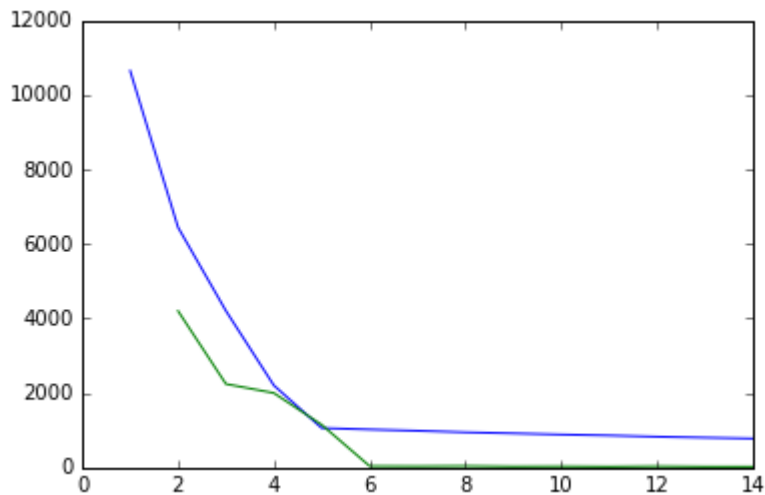


Gráfico de "codo" algoritmo k-means, centroides iniciados aleatoriamente.

En el gráfico de codo obtenido con k-means (línea azul) se puede observar que "codo" de la gráfica está en 5 grupos, basando la decisión final en la aceleración (línea verde) se agruparán los datos en 6 grupos.

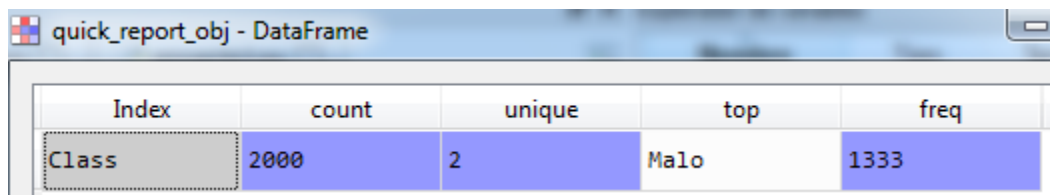
2. (3 puntos) En una fábrica de piezas automotrices detectaron que la calidad de las piezas que se producían podía evaluarse realizando las medidas de dos equipos dentro de la fábrica. Las categorías de calidad definidas son "Bueno", "Regular", "Malo". Después de realizar varias muestras lograron crear una base de datos con algunas muestras ya clasificadas (dataset_2.csv). Realicé el diseño de un clasificador que logré identificar las categorías especificadas en la base de datos. Entregue el código que se utilizó para el diseño del clasificador.

```
@author: Edu
"""
import pandas as pd
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures as plf
from sklearn.metrics import (confusion_matrix,
                             precision_score,
                             recall_score,
                             f1_score,
                             accuracy_score)

#%%
data_file='dataset_2.csv'
data=pd.read_csv(data_file,header=0,skip_blank_lines=True,parse_dates=False)
data=data.iloc[0:2000,0:3]

quick_report_obj=data.describe(include=['object']).transpose()

data2=data.iloc[:,0:2]
temp=pd.get_dummies(data['Class'])
data2=data2.join(temp.Regular) # 1= Regular, 0=Malo
### separar datos de de entranamiento y prueba
porcentaje_datos_m=.66666
ind=round(porcentaje_datos_m*len(data2.V1))
data_m=data2.iloc[0:ind,:]
data_p=data2.iloc[ind:len(data2.V1),:]
```



Index	count	unique	top	freq
Class	2000	2	Malo	1333

En las instrucciones dice que hay 3 diferentes categorías, pero en los datos de la base de datos "dataset_2.csv" solo existen 2 categorías, por lo que se convirtió la variable "Class" a variable tipo dummy, donde el "1" representa la clasificación "Regular" y el "0" representa la "Malo".

Se separaron los datos en dos, datos entrenamiento y datos de prueba.

```

31 ### entrenar regresion con datos de entrenamiento
32 x=data_m.iloc[:,0:2]
33 y=data_m.iloc[:,2:3]
34
35 xa= plf(degree=2).fit_transform(x)
36 reglog=linear_model.LogisticRegression(C=1)#crear el modelo, # la "c"
37 reglog.fit(xa,y)# entrenar el modelo
38
39 yg=reglog.predict(xa)
40 cm=confusion_matrix(y,yg)
41 print ('\t Accuracy: %1.3f' %accuracy_score(y,yg))
42 print ('\t Precision: %1.3f' %precision_score(y,yg))
43 print ('\t Recall: %1.3f' %recall_score(y,yg))
44 print ('\t F1: %1.3f' %f1_score(y,yg))
45
46 ### hacer pronósticos y comparar con datos de prueba
47 x1=data_p.iloc[:,0:2]
48 y1=data_p.iloc[:,2:3]
49
50 xa1= plf(degree=2).fit_transform(x1)
51
52 yg1=reglog.predict(xa1)
53 cm=confusion_matrix(y1,yg1)
54 print ('\t Accuracy: %1.3f' %accuracy_score(y1,yg1))
55 print ('\t Precision: %1.3f' %precision_score(y1,yg1))
56 print ('\t Recall: %1.3f' %recall_score(y1,yg1))
57 print ('\t F1: %1.3f' %f1_score(y1,yg1))
58

```

```

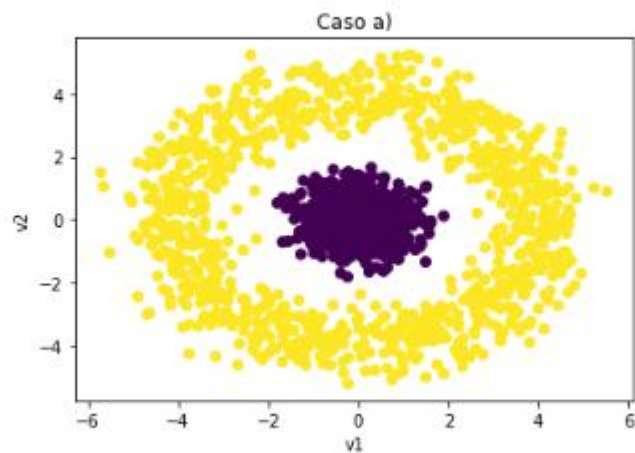
Accuracy: 0.963
Precision: 0.986
Recall: 0.908
F1: 0.946

```

Este fue en el resultado de los pronósticos en los datos de prueba, con la regresión logística entrenada con los datos de entrenamiento, como se observa el modelo tuvo un buen desempeño, teniendo una precisión al marcar las piezas como "Regular" del 98.6%, mientras que de las piezas que efectivamente resultaron ser "Regular" el modelo detectó el 90.8% de las piezas.

3. (5 puntos) A continuación se presentan dos conjuntos de datos a) y b), en los cuales se tienen 2 variables con 2000 muestras y existen 2 categorías o clases. Explique y justifique en cada caso ¿es posible hacer una reducción de las variables por medio del "PCA"? Y muestre en un gráfico, como serían los datos después de la reducción.

a. Base de datos "dataset_3a.csv"

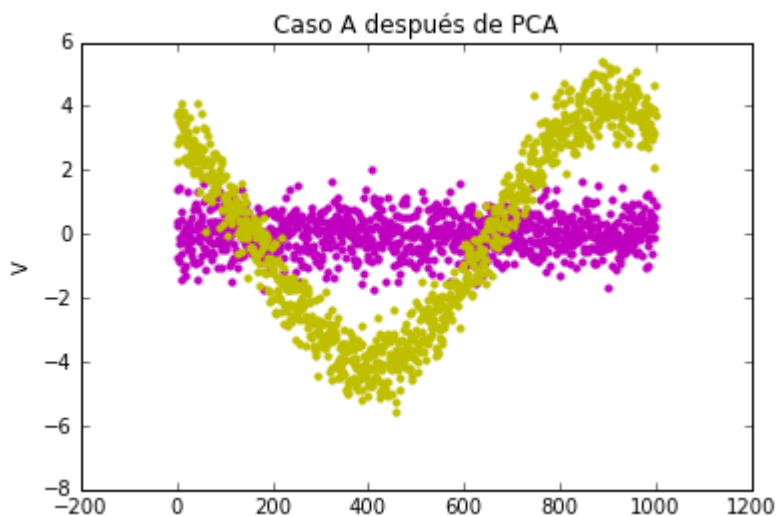


Para el caso A, estos son los valores propios que se obtienen:

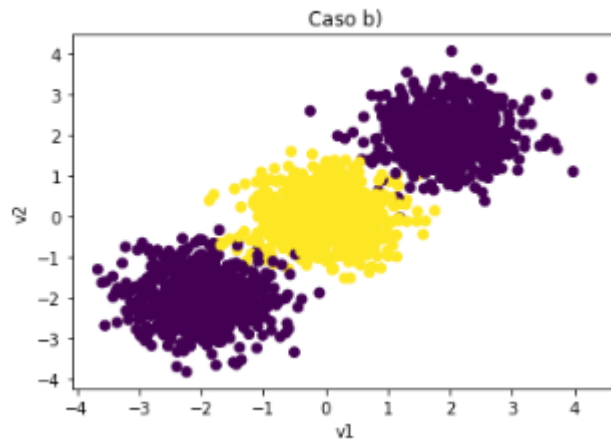
	0
0	4.480
1	4.305

Estos valores propios nos indican que de proyectar una de las variables, se perdería el 49.04% de la información, por lo que no es recomendable hacer una reducción por este método para estos datos.

De hacer la reducción por análisis de componentes principales, los datos quedarían de la siguiente manera:



b. Base de datos "dataset_3b.csv"

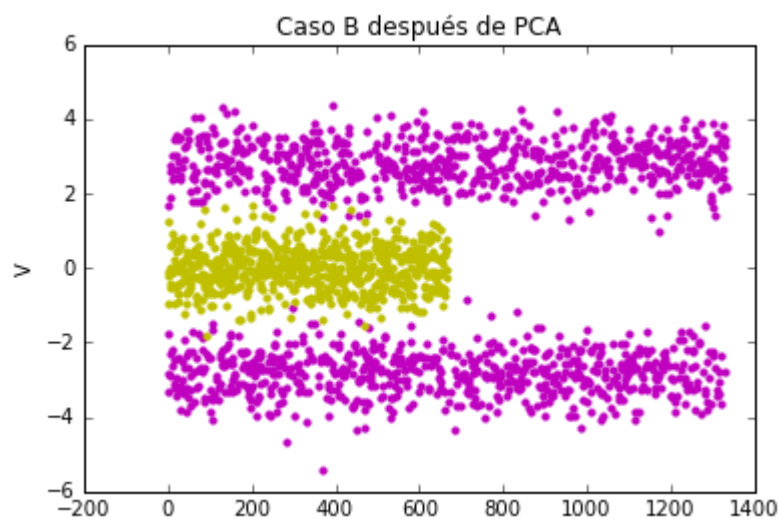


Para el caso B, estos son los valores propios que se obtienen:

	0
0	0.349
1	5.665

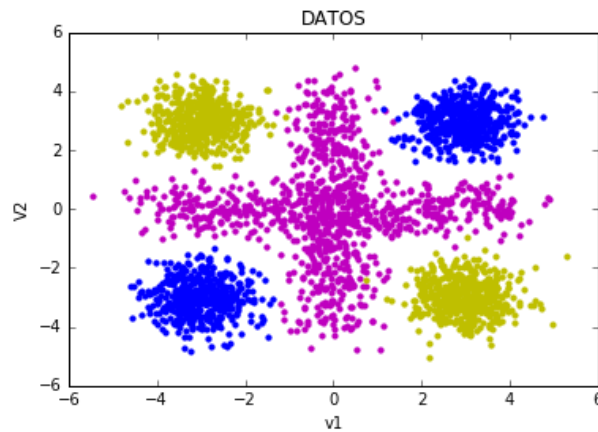
Estos valores propios nos indican que de proyectar una de las variables, se perdería tan solo el 5.81% de la información, por lo que para estos datos si es posible hacer una reducción por análisis de componentes principales.

De hacer la reducción por análisis de componentes principales, los datos quedarían de la siguiente manera:



4. (4 puntos) Aplique un clasificador del tipo regresión logística y un SVM, en la base de datos "dataset_4.csv". Indique cuál de los dos tiene un mejor resultado y porque creen que suceda esto. Justifique su respuesta con código, mediciones o figuras.

```
8 import pandas as pd
9 import numpy as np
10 from sklearn import linear_model
11 from sklearn import svm
12 import matplotlib.pyplot as plt
13 from sklearn.preprocessing import PolynomialFeatures as plf
14 from sklearn.metrics import (confusion_matrix,
15                               precision_score,
16                               recall_score,
17                               f1_score,
18                               accuracy_score)
19 ###
20 data_file='dataset_4.csv'
21 data=pd.read_csv(data_file,header=None)
22 data=data.iloc[0:3000,0:3]
23 data.columns=['V1','V2','Class']
24 unique_values_counts=pd.DataFrame(columns=['unique values'])
25 for v in list(data.columns.values):
26     unique_values_counts.loc[v]=[data[v].nunique()]
27
28 ### graficar datos
29
30 index=data['Class']==1
31 data_1=data.loc[index,:]
32 index=data['Class']==0
33 data_0=data.loc[index,:]
34 index=data['Class']==2
35 data_2=data.loc[index,:]
36
37 plt.figure()
38 plt.title('DATOS')
39 plt.ylabel('V2')
40 plt.xlabel('V1')
41 plt.scatter(data_2.V1,data_2.V2,color='m',s=10)
42 plt.scatter(data_1.V1,data_1.V2,color='y',s=10)
43 plt.scatter(data_0.V1,data_0.V2,color='b',s=10)
44 plt.show()
```



Haciendo un análisis gráfico de los datos, se puede prever que es muy posible que se necesite aplicar un kernel, dada la distribución de los mismos sería muy complicado hacer una separación lineal.

Se hizo la regresión logística tipo "one vs all" para poder clasificar las 3 distintas clases de los datos.

```

46 ### regresión logística one VS all
47
48 # primera regresión 1=1, 0=0, 2=0 #####
49 index=data['Class']!=1
50 data_1=data.iloc[:,2]
51 data_1[index]=0
52
53 x=data.iloc[:,0:2]
54 y=data_1
55 y=np.ravel(y)
56
57 xa=plf(degree=2).fit_transform(x)
58 reglog=linear_model.LogisticRegression(C=1)#crear el modelo, # La "C" e
59 reglog.fit(xa,y)# entrenar el modelo
60
61 yg=reglog.predict(xa)
62 cm=confusion_matrix(y,yg)
63 print ('\t Accuracy: %1.3f' %accuracy_score(y,yg))
64 print ('\t Precision: %1.3f' %precision_score(y,yg))
65 print ('\t Recall: %1.3f' %recall_score(y,yg))
66 print ('\t F1: %1.3f' %f1_score(y,yg))
67
68 # Segunda regresión 1=0, 0=1, 2=0 #####
69
70 data=pd.read_csv(data_file,header=None)
71 data=data.iloc[0:3000,0:3]
72 data.columns=['V1', 'V2', 'Class']
73
74 index=data['Class']!=0
75 index2=data['Class']==0
76 data_2=data.iloc[:,2]
77 data_2[index2]=1
78 data_2[index]=0
79
80 x2=data.iloc[:,0:2]
81 y2=data_2
82 y2=np.ravel(y2)
83

```



```

84 xa2=plf(degree=2).fit_transform(x2)
85 reglog2=linear_model.LogisticRegression(C=1)#crear el modelo, # l
86 reglog2.fit(xa2,y2)# entrenar el modelo
87
88 yg2=reglog2.predict(xa2)
89 cm2=confusion_matrix(y2,yg2)
90 print ('\t Accuracy: %1.3f' %accuracy_score(y2,yg2))
91 print ('\t Precision: %1.3f' %precision_score(y2,yg2))
92 print ('\t Recall: %1.3f' %recall_score(y2,yg2))
93 print ('\t F1: %1.3f' %f1_score(y2,yg2))
94
95 # tercera regresión 1=0, 0=0, 2=1 #####
96
97 data=pd.read_csv(data_file,header=None)
98 data=data.iloc[0:3000,0:3]
99 data.columns=['V1','V2','Class']
100
101 index=data['Class']!=2
102 index2=data['Class']==2
103 data_3=data.iloc[:,2]
104 data_3[index2]=1
105 data_3[index]=0
106
107 x3=data.iloc[:,0:2]
108 y3=data_3
109 y3=np.ravel(y3)
110
111 xa3=plf(degree=3).fit_transform(x3)
112 reglog3=linear_model.LogisticRegression(C=1)#crear el modelo, # l
113 reglog3.fit(xa3,y3)# entrenar el modelo
114
115 yg3=reglog3.predict(xa3)
116 cm3=confusion_matrix(y3,yg3)
117 print ('\t Accuracy: %1.3f' %accuracy_score(y3,yg3))
118 print ('\t Precision: %1.3f' %precision_score(y3,yg3))
119 print ('\t Recall: %1.3f' %recall_score(y3,yg3))
120 print ('\t F1: %1.3f' %f1_score(y3,yg3))
121

```

Polinomio 2°

Regresión 1=1, 0=0, 2=0

Accuracy: 0.998
Precision: 0.997
Recall: 0.997
F1: 0.997

Polinomio 2°

Regresión 1=0, 0=1, 2=0

Accuracy: 0.999
Precision: 0.999
Recall: 0.999
F1: 0.999

Polonomio 4°

Regresión 1=0,0=0,2=1

Accuracy: 0.998
Precision: 0.996
Recall: 0.997
F1: 0.997

Las regresiones se ajustan bien con polinomio de 2° grado a la hora clasificar los datos agrupados en las esquinas, para los datos agrupados en forma de "cruz" se requiere un polinomio dos grados mayor para poder que separe bien los datos, esto se esperaba por la peculiar forma de estos datos.

```
122 ### Vector Soporte SVM
123 X = data.iloc[:,2]
124 Y= data.iloc[:,2]
125
126 # crear y entrenar modelo svm
127 svc = svm.SVC(kernel='linear').fit(X,Y)
128 svc_poly = svm.SVC(kernel='poly',degree=3).fit(X,Y)
129 svc_rbf = svm.SVC(kernel='rbf').fit(X,Y)
130
131 # Crear un mesh para dibujar la frontera
132 h = 0.02
133 xx,yy = np.meshgrid(np.arange(4,8,h),np.arange(1.5,4.5,h))
134
135 # Dibujar las fronteras
136 titles = ['SVM Lineal', 'SVM Polinomial', 'SVM Radial']
137 for j,clf in enumerate((svc,svc_poly,svc_rbf)):
138     plt.subplot(2,2,j+1)
139     plt.subplots_adjust(wspace=0.4,hspace=0.6)
140     P = clf.predict(np.c_[xx.ravel(),yy.ravel()])
141     P = P.reshape(xx.shape)
142     plt.contourf(xx,yy,P,cmap=plt.cm.Paired,alpha =0.7)
143     plt.scatter(X.iloc[:,0],X.iloc[:,1],c=Y,)
144     plt.xlabel('V1')
145     plt.ylabel('V2')
146     plt.title(titles[j])
147
148     Yg =clf.predict(X)
149     cm = confusion_matrix(Y,Yg)
150     print(titles[j])
151     print('\t Accuracy: %1.3f' % accuracy_score(Y,Yg))
152     print('\t Precision: %1.3f' % precision_score(Y,Yg))
153     print('\t Recall: %1.3f' % recall_score(Y,Yg))
154     print('\t F1: %1.3f' % f1_score(Y,Yg))
155
156 plt.show()
```

SVM Lineal

Accuracy: 0.489
Precision: 0.507
Recall: 0.489
F1: 0.433

Polinomio 3°
SVM Polinomial

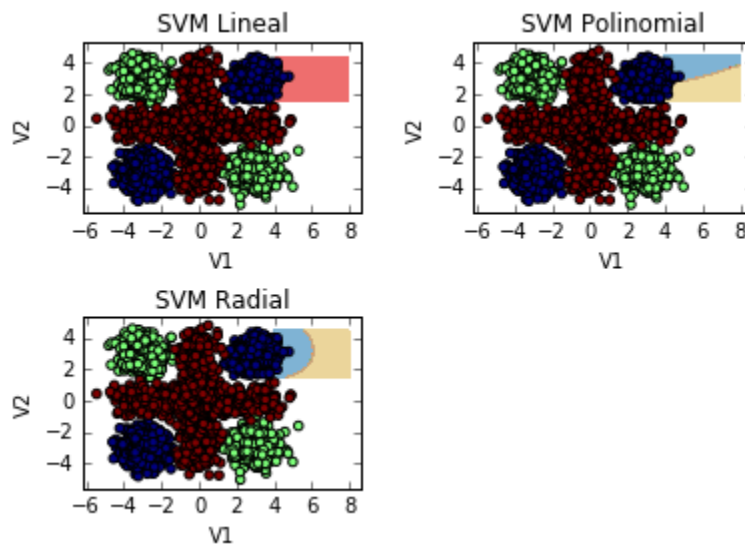
Accuracy: 0.472
Precision: 0.468
Recall: 0.472
F1: 0.430

Polinomio 4°
SVM Polinomial

Accuracy: 0.998
Precision: 0.998
Recall: 0.998
F1: 0.998

SVM Base Radial

Accuracy: 0.997
Precision: 0.997
Recall: 0.997
F1: 0.997



Para este conjunto de datos, se podrían utilizar tanto el modelo de regresión logística "one vs all" con polinomios de 2°, 2° y 4°, como también un modelo de máquina de vector soporte polinomial 4° o uno con kernel base radial. Dado la distribución de datos, es prácticamente imposible hacer una buena separación con la dimensión original de ellos, sin importar que modelo se elija, lo importante aquí es aplicar un Kernel adecuado para así encontrar un hyperplano que pueda hacer una buena separación.