

# Taller 2 - Estudio Comparativo de Clasificadores

**Curso: Machine Learning y Deep Learning**

**Diplomado en Data Science**

Nombre: Dr. Ing. Rodrigo Salas (rodrigo.salas@uv.cl)

- Nombre integrante 1:Juan Pablo González Collao
- Nombre integrante 2:Pablo Omar Walters Barraza
- Nombre integrante 3:

Fecha de entrega: Lunes 11 de Julio 2022

A continuación coloque todos los toolbox que utilizó para el desarrollo de esta actividad.

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.svm import SVC
from scipy.stats import spearmanr
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram
from sklearn.cluster import AgglomerativeClustering
from sklearn.inspection import permutation_importance
from sklearn.ensemble import RandomForestClassifier
from scipy.cluster import hierarchy
#from google.colab import drive
#drive.mount('/content/drive')
```

# 1. Estudio Comparativo de Clasificadores (30 puntos)

En esta pregunta se realizará un estudio comparativo de los desempeños de los clasificadores para realizar una tarea en particular. Para realizar esta actividad, su equipo deberá escoger uno de los siguientes conjuntos de datos:

- [Parkinsons](#)
- [Wisconsin Breast Cancer](#)
- [Bank Marketing](#)
- [Student Performance](#)
- [Heart Disease Data Set](#)

Sólo se deberá escoger un solo conjunto de datos.

## 1.1. Realizar la lectura del archivo y guardar la información en un DataFrame de pandas

```
In [2]: df = pd.read_csv('C:\\\\Users\\\\pablo.walters\\\\Desktop\\\\clase\\\\taller 2\\\\breast cancer.csv')
#df=pd.read_csv('/content/drive/MyDrive/lab mod 4/Tarea 2/breast cancer.csv')
df["diagnosis"] = df["diagnosis"].replace("M", 0)
df["diagnosis"] = df["diagnosis"].replace("B", 1)

#df['diagnosis']=df[df['diagnosis']=="M"]==1
#df['diagnosis']=df[df['diagnosis']=="B"]==0
df
```

Out[2]:

	<b>id</b>	<b>diagnosis</b>	<b>radius_mean</b>	<b>texture_mean</b>	<b>perimeter_mean</b>	<b>area_mean</b>	<b>smoothness_m</b>
<b>0</b>	842302	0	17.99	10.38	122.80	1001.0	0.11
<b>1</b>	842517	0	20.57	17.77	132.90	1326.0	0.08
<b>2</b>	84300903	0	19.69	21.25	130.00	1203.0	0.10
<b>3</b>	84348301	0	11.42	20.38	77.58	386.1	0.14
<b>4</b>	84358402	0	20.29	14.34	135.10	1297.0	0.10
...	...	...	...	...	...	...	...
<b>564</b>	926424	0	21.56	22.39	142.00	1479.0	0.11
<b>565</b>	926682	0	20.13	28.25	131.20	1261.0	0.09
<b>566</b>	926954	0	16.60	28.08	108.30	858.1	0.08
<b>567</b>	927241	0	20.60	29.33	140.10	1265.0	0.11
<b>568</b>	92751	1	7.76	24.54	47.92	181.0	0.05

569 rows × 33 columns

In [3]: `df["diagnosis"].value_counts()`

Out[3]:

1	357
0	212
Name: diagnosis, dtype: int64	

In [4]:

```
df1=pd.read_csv('C:\\\\Users\\\\pablo.walters\\\\Desktop\\\\clase\\\\taller 2\\\\processed.csv')
df1 = df1[df1['thal'] != "?"]
df1 = df1[df1['ca'] != "?"]
df1['age']=df1['age'].astype(float)
df1['sex']=df1['sex'].astype(float)
df1['cp']=df1['cp'].astype(float)

df1['trestbps']=df1['trestbps'].astype(float)
df1['chol']=df1['chol'].astype(float)
df1['fbs']=df1['fbs'].astype(float)
df1['restecg']=df1['restecg'].astype(float)
df1['thalach']=df1['thalach'].astype(float)
df1['exang']=df1['exang'].astype(float)
df1['oldpeak']=df1['oldpeak'].astype(float)
df1['slope']=df1['slope'].astype(float)
df1['ca']=df1['ca'].astype(float)
df1['thal']=df1['thal'].astype(float)
df1['num']=df1['num'].astype(float)
df1["num"]=df1["num"].replace(4, 1).replace(3, 1).replace(2, 1)
df1
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0.0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	1.0
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1.0
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0.0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
297	57.0	0.0	4.0	140.0	241.0	0.0	0.0	123.0	1.0	0.2	2.0	0.0	7.0	1.0
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1.0
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	1.0
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	1.0
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1.0

297 rows × 14 columns

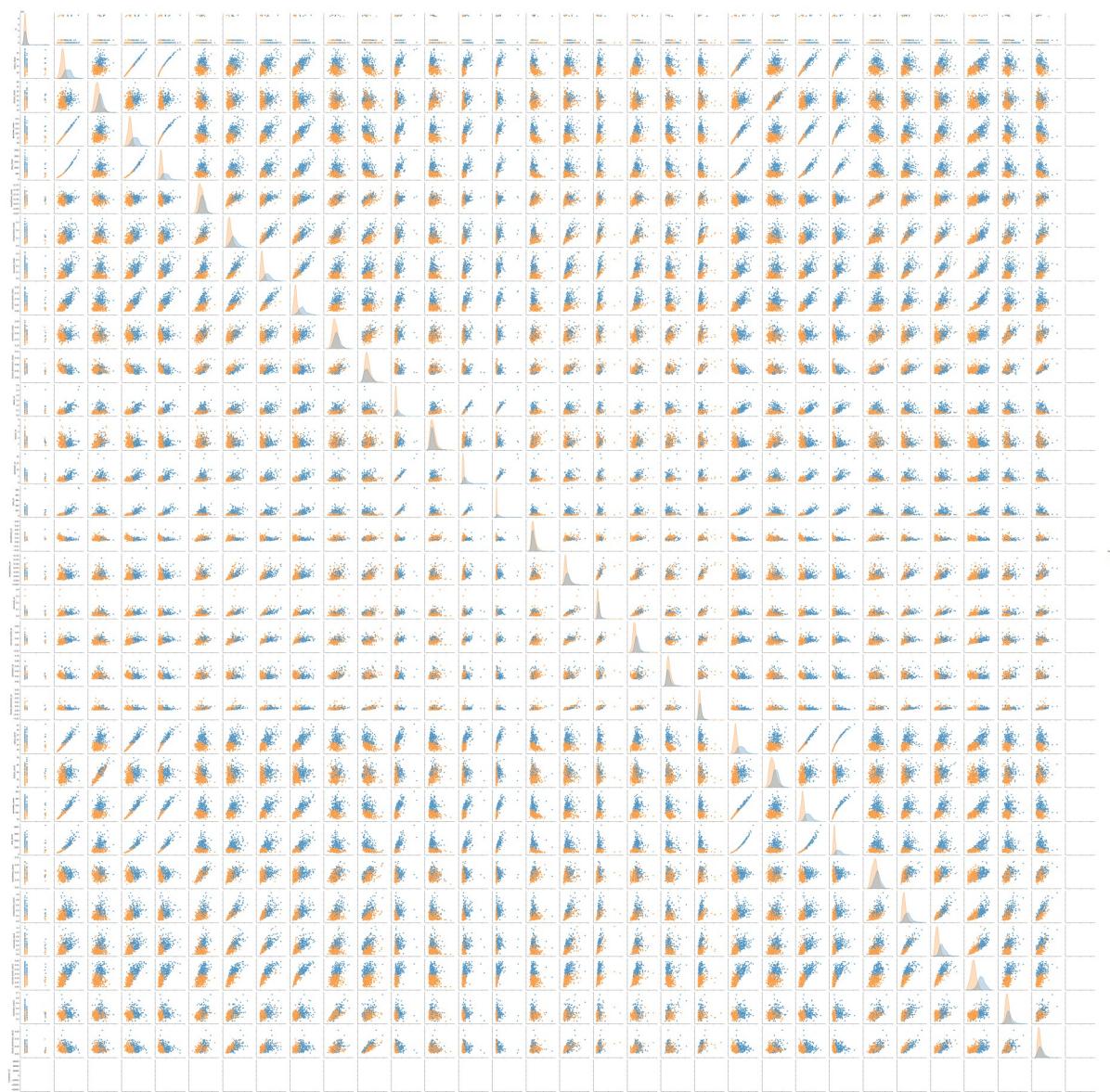
In [5]: `df1["num"].value_counts()`

Out[5]:

0.0	160
1.0	137
Name: num, dtype: int64	

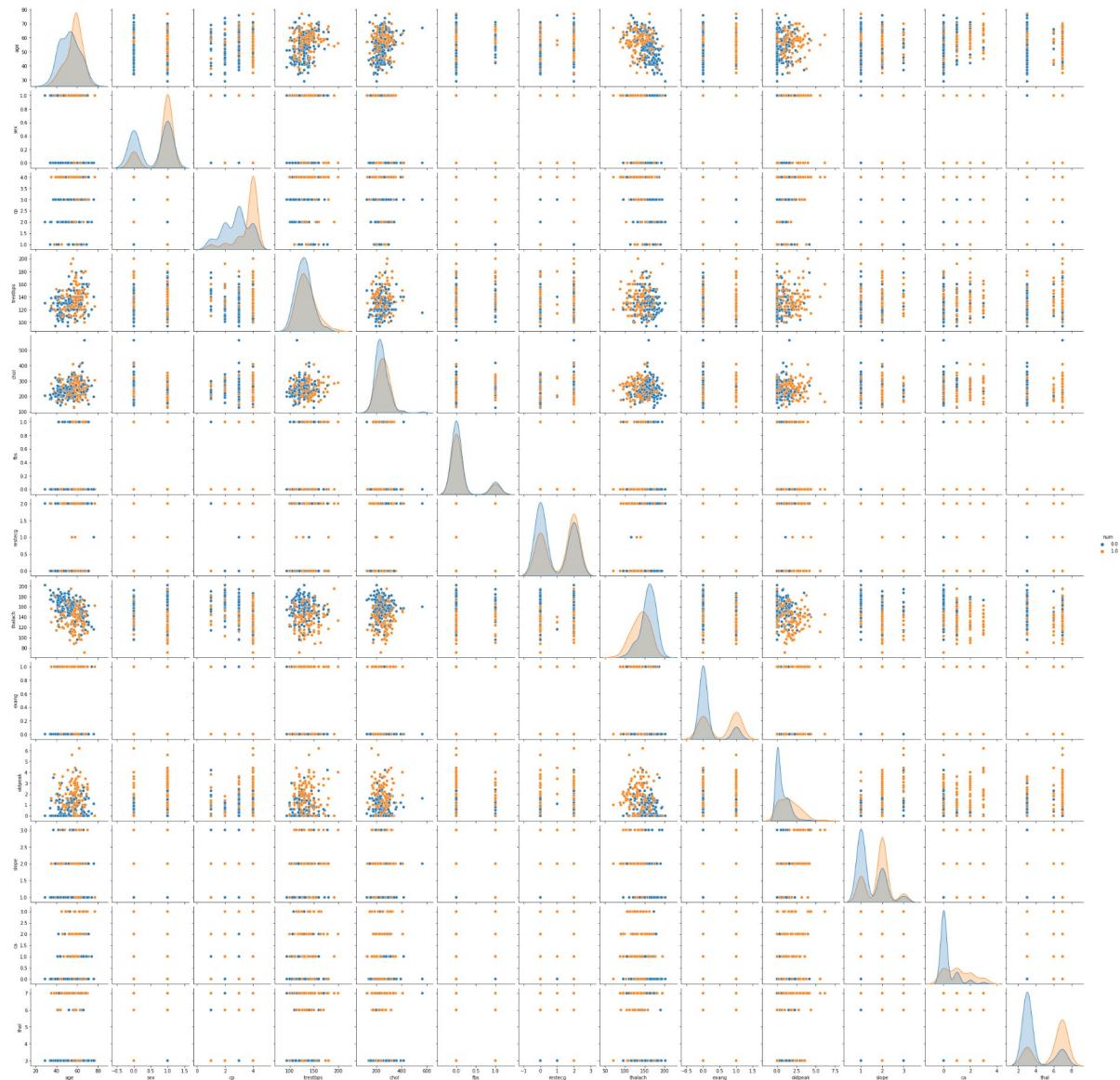
```
In [6]: sns.pairplot(df,hue='diagnosis')
```

```
Out[6]: <seaborn.axisgrid.PairGrid at 0x25a6ff1c070>
```



```
In [7]: sns.pairplot(df1,hue='num')
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x25a11430490>
```



**1.2.** Explicar brevemente cuál es el objetivo de este conjunto de datos. Indicar cuánto es la cantidad de datos, cuántas variables de entradas se tienen, cuántas son la cantidad de clases. Explicar brevemente el significado de la variable target. Explicar brevemente el significado de 5 variables de entrada.

### **Set de cancer de mama:**

El dataset se trata de una serie de observaciones relacionadas a las características del núcleo celular de diferentes muestras de tejido mamario. Las muestras de tejido mamario son obtenidas a través de la aspiración con aguja fina (FNA) y los datos relacionados a las distintas características morfológicas, son obtenidos a través de imágenes digitalizadas. El objetivo de esta información es caracterizar morfológicamente los núcleos celulares de muestras de tejido mamario con y sin cáncer, para generar un modelo que prediga la existencia de cáncer.

El dataset contiene 570 observaciones y 31 variables (incluyendo un ID y el target). Es decir, se tienen 29 variables de entrada para 2 clases, donde un 62,7% de las observaciones corresponde a tumores malignos y un 37,3% a tumores benignos (Benigno=0 y Maligno=1) A continuación, explicación breve de 5 variables de entrada:

- radius\_mean: corresponde al promedio del radio del núcleo celular, es decir distancias del centro a los puntos del perímetro.
- texture\_mean: corresponde a la desviación estándar de los valores de la escala de grises
- perimeter\_mean: corresponde al promedio del perímetro de los núcleos celulares
- concavity: severidad de las porciones cóncavas del contorno)
- concave points: número de porciones cóncavas del contorno

### **Set de enfermedades del cardiacas:**

El dataset consiste en la selección de un subconjunto de 14 variables de 76 variables originales (características y síntomas de pacientes) asociadas a un diagnóstico de enfermedad cardiaca a partir de una angiografía. El objetivo de esta información es predecir la enfermedad cardiaca, a partir de la observación de características y síntomas de pacientes.

El dataset contiene 298 observaciones y 14 variables (incluyendo el target).

Es decir, se tienen 13 variables de entrada para 5 clases, donde un 45,9% de las observaciones corresponde a algún grado de enfermedad cardiaca y un 54,1% a la inexistencia de enfermedad cardiaca. Cabe destacar, que para efectos del modelo, se consideran 2 clases (sano=0 y enfermo=1) A continuación, explicación breve de 5 variables de entrada:

- trestbps: tensión arterial en reposo (en mm Hg al ingreso hospitalario)
- chol: colesterol sérico en mg/dl
- fbs: glucemia en ayunas mayor o menor a 120 mg/dl. (mayor:1, menor: 0)
- restecg: resultados electrocardiográficos en reposo (normal:0, anormal: 1 y 2)
- thalach: frecuencia cardíaca máxima alcanzada

**1.3.** Separar el conjunto de datos en una matriz \$X\$ con las variables de entrada y en el target \$y\$

```
In [8]: target=df['diagnosis']
variables=df.drop(columns=["id","diagnosis","Unnamed: 32"])
```

```
In [9]: target1=df1['num']
variables1=df1.drop(columns="num")
```

**1.4.** Separar el conjunto de datos en conjunto de entrenamiento y en conjunto de test. Considerar el \$70\%\$ de los datos para entrenamiento. (Se sugiere no utilizar semilla de forma tal que las particiones sean aleatorias)

```
In [10]: ## Cancer de mama
X_training, X_test, y_training, y_test = train_test_split(variables, target, test_s
```

```
In [11]: ## Corazón
X_training1, X_test1, y_training1, y_test1 = train_test_split(variables1, target1,
```

**1.5.** Estandarizar los datos de entrada del conjunto de entrenamiento y test utilizando el "standard scaler". (El standar scaler se ajusta al conjunto de entranamiento y luego se transforman los datos del conjunto de entrenamiento y el de test)

```
In [12]: ## Cancer de mama
sc = StandardScaler()
sc.fit(X_training)

Z_train = sc.transform(X_training)
Z_test = sc.transform(X_test)
```

```
In [13]: ## Corazón
sc = StandardScaler()
sc.fit(X_training1)

Z_train1 = sc.transform(X_training1)
Z_test1 = sc.transform(X_test1)
```

```
In [48]: #Función para obtener parametros de desempeño
def desempeño(y_pred,y_real):
    desempeno=[]

    vp=confusion_matrix(y_pred, y_real)[0][0]
    fp=confusion_matrix(y_pred, y_real)[0][1]
    fn=confusion_matrix(y_pred, y_real)[1][0]
    vn=confusion_matrix(y_pred, y_real)[1][1]

    desempeno.append(accuracy_score(y_pred,y_real))
    sensibilidad=vp/(vp+fn)
    desempeno.append(sensibilidad)
    especificidad=vn/(vn+fp)
    desempeno.append(especificidad)
    p_negativo=vn/(vn+fn)
    desempeno.append(p_negativo)
    precision=vp/(vp+fp)
    desempeno.append(precision)
    t_error=(1-accuracy_score(y_pred,y_real))
    desempeno.append(t_error)
    medida_f=2*((precision*sensibilidad)/(precision+sensibilidad))
    desempeno.append(medida_f)

return desempeno
```

### 1.6. Aplicar los siguientes modelos de Scikit Learn:

- LDA
- QDA
- SVC Lineal
- SVC Radio Basal
- Random Forest
- Gradient Boosting
- Multilayer Perceptron

y obtener las siguientes métricas de desempeño:

- Accuracy (Exactitud)
- Sensibilidad
- Precisión
- Especificidad
- Tasa de Error
- Medida F

Realizar una tabla comparativa y concluir.

```
In [49]: classificacion=[]

## LDA

lda = LinearDiscriminantAnalysis(solver="svd")
y_lda = lda.fit(Z_train, y_training).predict(Z_test)
classificacion.append(desempeño(y_lda,y_test))

## QDA

qda = QuadraticDiscriminantAnalysis()
y_qda = qda.fit(Z_train, y_training).predict(Z_test)
classificacion.append(desempeño(y_qda,y_test))

## SVC Lineal

SVCL = SVC(kernel='linear', C=1E6).fit(Z_train,y_training)
y_svc_lineal=SVCL.predict(Z_test)
classificacion.append(desempeño(y_svc_lineal,y_test))

## SVC radio basal

SVCR = SVC(kernel='rbf', C=1E6).fit(Z_train, y_training)
y_svc_radial=SVCR.predict(Z_test)
classificacion.append(desempeño(y_svc_radial,y_test))

## Random Forest

RF = RandomForestClassifier().fit(Z_train, y_training)
y_rf=RF.predict(Z_test)
classificacion.append(desempeño(y_rf,y_test))

## Gradient Boosting

GB = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1).f
y_gb=GB.predict(Z_test)
classificacion.append(desempeño(y_gb,y_test))

## Multilayer Perceptron

MLP = MLPClassifier(max_iter=300).fit(Z_train, y_training)
y_mpl=MLP.predict(Z_test)
classificacion.append(desempeño(y_mpl,y_test))
```

```
In [50]: classificacion1=[]

## LDA

lda = LinearDiscriminantAnalysis(solver="svd")
y_lda1 = lda.fit(Z_train1, y_training1).predict(Z_test1)
classificacion1.append(desempeño(y_lda1,y_test1))
#print("LDA",confusion_matrix(y_lda1,y_test1))

## QDA

qda = QuadraticDiscriminantAnalysis()
y_qda1 = qda.fit(Z_train1, y_training1).predict(Z_test1)
classificacion1.append(desempeño(y_qda1,y_test1))

#print("QDA",confusion_matrix(y_qda1,y_test1))

## SVC Lineal

SVCL = SVC(kernel='linear', C=1E3).fit(Z_train1,y_training1)
y_svc_lineal1=SVCL.predict(Z_test1)
classificacion1.append(desempeño(y_svc_lineal1,y_test1))

#print("SVC Lineal",confusion_matrix(y_svc_lineal1,y_test1))

## SVC radio basal

SVCR = SVC(kernel='rbf', C=1E3).fit(Z_train1, y_training1)
y_svc_radial1=SVCR.predict(Z_test1)
classificacion1.append(desempeño(y_svc_radial1,y_test1))

#print("SVC radio basal",confusion_matrix(y_svc_radial1,y_test1))

## Random Forest

RF = RandomForestClassifier().fit(Z_train1, y_training1)
y_rf1=RF.predict(Z_test1)
classificacion1.append(desempeño(y_rf1,y_test1))

#print("Random Forest",confusion_matrix(y_rf1,y_test1))

## Gradient Boosting

GB = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1).fit(Z_train1, y_training1)
y_gb1=GB.predict(Z_test1)
classificacion1.append(desempeño(y_gb1,y_test1))

#print("Gradient Boosting",confusion_matrix(y_gb1,y_test1))

## Multilayer Perceptron

MLP = MLPClassifier(max_iter=300).fit(Z_train1, y_training1)
y_mpl1=MLP.predict(Z_test1)
classificacion1.append(desempeño(y_mpl1,y_test1))

#print("Multilayer Perceptron",confusion_matrix(y_mpl1,y_test1))
```

C:\Users\pablo.walters\Anaconda3\envs\Modulo4\lib\site-packages\sklearn\neural\_network\multilayer\_perceptron.py:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.  
warnings.warn(

```
In [51]: plt.rcParams["figure.figsize"] = [25, 5]
plt.rcParams["figure.autolayout"] = True

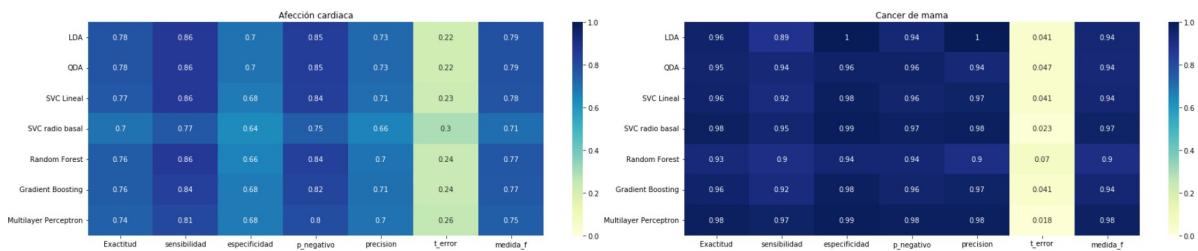
rty = pd.DataFrame(classificacion,index=['LDA','QDA','SVC Lineal','SVC radio basal'])
rty1=pd.DataFrame(classificacion1,index=['LDA','QDA','SVC Lineal','SVC radio basal'])

fig, (ax1, ax2) = plt.subplots(ncols=2)
fig.subplots_adjust(wspace=0.01)

sns.heatmap(rty1,vmax=1,vmin=0,annot=classificacion1, cmap="YlGnBu", ax=ax1).set_title('Afección cardiaca')
sns.heatmap(rty,vmax=1,vmin=0,annot=classificacion, cmap="YlGnBu", ax=ax2).set_title('Cancer de mama')

#ax2.yaxis.tick_right()

fig.subplots_adjust(wspace=0.001)
plt.show()
```



## Desempeño de Modelos

Respecto de la Exactitud, el desempeño de los modelos es bastante similar para ambos dataset. en el caso del cáncer de mama, cuyo dataset contiene mayor cantidad de variables y con mayor variabilidad, el mejor desempeño se obtuvo con el modelo Multilayer Perception, el cual tuvo el mejor rendimiento a excepción de las métricas especificidad y precisión. A diferencia de lo anterior, el dataset de afección cardiaca muestra un mejor desempeño con los modelos más simples (LDA y QDA), donde además, mostró mejor rendimiento en todas las métricas. Cabe destacar que se obtuvieron mejores desempeños en la aplicación de los modelos en el dataset que contiene una mayor cantidad de datos, variables y variabilidad de los datos. En el caso del dataset de afecciones cardiacas, se utilizaron variables de entrada nominales y ordinales.

## 2. Estudio de Variables Relevantes (30 puntos)

En esta pregunta se estudiarán las variables (Features) más relevantes para la predicción.

**2.1** Realizar un estudio de colinealidad. Para ello deberá obtener la matriz de correlación y aplicar un Jerárquical Clustering. Graficar la matriz de correlación y el dendograma. Concluir sobre los resultados.

In [18]:

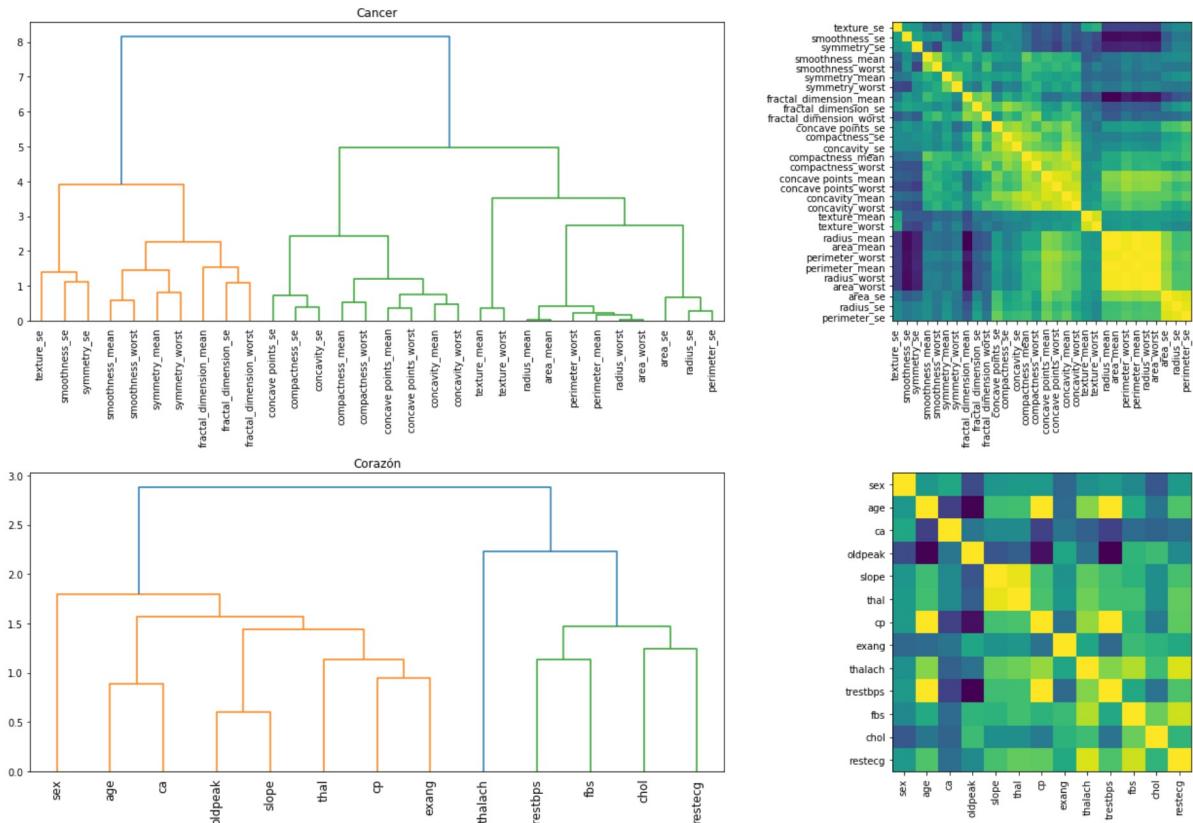
```

fig, ax = plt.subplots(2, 2, figsize=(20, 12))
corr = spearmanr(Z_train).correlation
corr_linkage = hierarchy.ward(corr)
dendro = hierarchy.dendrogram(corr_linkage, labels=X_training.columns, ax=ax[0][0],
ax[0][0].set_title("Cancer"))
dendro_idx = np.arange(0, len(dendro['ivl']))

ax[0][1].imshow(corr[dendro['leaves'], :][:, dendro['leaves']])
ax[0][1].set_xticks(dendro_idx)
ax[0][1].set_yticks(dendro_idx)
ax[0][1].set_xticklabels(dendro['ivl'], rotation='vertical')
ax[0][1].set_yticklabels(dendro['ivl'])
fig.tight_layout()

corr2 = spearmanr(Z_train1).correlation
corr_linkage2 = hierarchy.ward(corr2)
dendro1 = hierarchy.dendrogram(corr_linkage2, labels=X_training1.columns, ax=ax[1][0],
ax[1][0].set_title("Corazón"))
dendro_idx1 = np.arange(0, len(dendro1['ivl']))
ax[1][1].imshow(corr[dendro1['leaves'], :][:, dendro1['leaves']])
#ax[1][1].imshow(corr1[dendro1['leaves'], :][:, dendro1['leaves']])
ax[1][1].set_xticks(dendro_idx1)
ax[1][1].set_yticks(dendro_idx1)
ax[1][1].set_xticklabels(dendro1['ivl'], rotation='vertical')
ax[1][1].set_yticklabels(dendro1['ivl'])
plt.show()

```



## Clustering

En el dendograma del dataset cáncer de mama, haciendo un corte en el nivel 4, se pueden observar claramente 3 clúster bien diferenciados entre sí, y con una alta similitud interna, lo cual se comprueba en la observación de la matriz de correlaciones, sin embargo, se puede destacar que las variables dentro del clúster de mayor correlación son variables correlacionadas entre sí (área, radio y perímetro), lo que también ocurre con los demás clúster, pero en menor medida. Por lo tanto, estas variables entregan información redundante, por lo cual se podría seleccionar un features por cada clúster y obtener una similar exactitud.

Por otro lado, en el dendograma del dataset de afecciones cardiacas se pueden diferenciar claramente solo 3 cluster, haciendo un corte en el nivel 2, dada la baja disimilaridad entre las agrupaciones observadas. Cabe destacar que la variable thalach, que corresponde a la frecuencia cardiaca, es un clúster por sí mismo, ya que no está vinculado a otra variable.

**2.2** Aplicar el método de Random Forest al conjunto de datos de entrenamiento y obtener un gráfico de barras para el ranking de las variables más relevantes. Concluir respecto a las dos primeras variables seleccionadas.

```
In [27]: clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_training, y_training)
print("Accuracy Cancer",clf.score(Z_test, y_test))

clf1 = RandomForestClassifier(n_estimators=100, random_state=42)
clf1.fit(X_training1, y_training1)
print("Accuracy Corazón:",clf1.score(Z_test1, y_test1))

Accuracy Cancer 0.6549707602339181
Accuracy Corazón: 0.4888888888888889

In [28]: result = permutation_importance(clf, X_training, y_training, n_repeats=100)
result1 = permutation_importance(clf1, X_training1, y_training1, n_repeats=100)
```

```
In [29]: perm_sorted_idx = result.importances_mean.argsort()
tree_importance_sorted_idx = np.argsort(clf.feature_importances_)
tree_indices = np.arange(0, len(clf.feature_importances_)) + 0.5

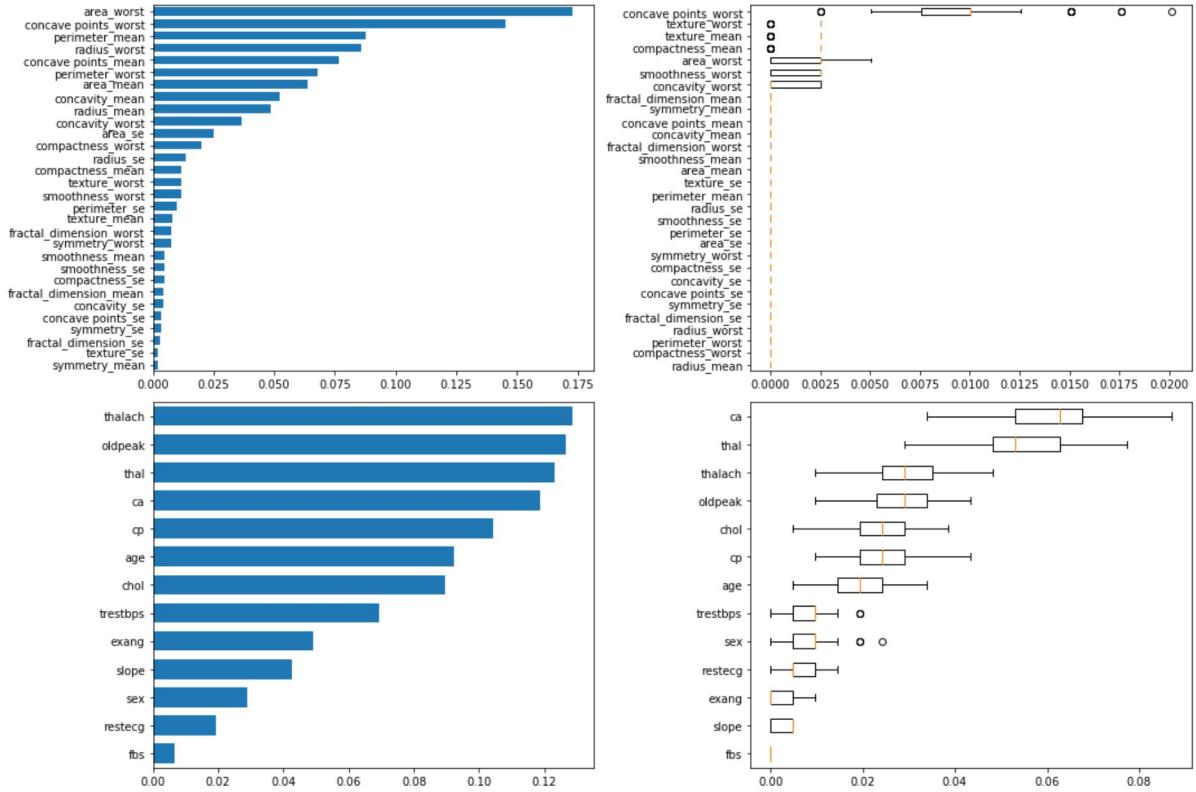
perm_sorted_idx1 = result1.importances_mean.argsort()
tree_importance_sorted_idx1 = np.argsort(clf1.feature_importances_)
tree_indices1 = np.arange(0, len(clf1.feature_importances_)) + 0.5

fig, ax = plt.subplots(2, 2, figsize=(15, 10))
ax[0][0].barh(tree_indices,clf.feature_importances_[tree_importance_sorted_idx], he
ax[0][0].set_yticklabels(X_training.columns[tree_importance_sorted_idx])
ax[0][0].set_yticks(tree_indices)
ax[0][0].set_ylim((0, len(clf.feature_importances_)))
ax[0][1].boxplot(result.importances[perm_sorted_idx].T, vert=False,
                  labels=X_training.columns[perm_sorted_idx])

print(clf1.feature_importances_)
ax[1][0].barh(tree_indices1,clf1.feature_importances_[tree_importance_sorted_idx1], he
ax[1][0].set_yticklabels(X_training1.columns[tree_importance_sorted_idx1])
ax[1][0].set_yticks(tree_indices1)
ax[1][0].set_ylim((0, len(clf1.feature_importances_)))
ax[1][1].boxplot(result1.importances[perm_sorted_idx1].T, vert=False, labels=X_trai

fig.tight_layout()
plt.show()
```

```
[0.09228843 0.02899791 0.10442151 0.06935471 0.08969668 0.00673601
 0.01942651 0.12870564 0.04903042 0.12674418 0.04268079 0.1187597
 0.12315751]
```



## Random Forest

Aplicando Random forest a ambos dataset, con 100 árboles, se obtiene un bajo desempeño en exactitud con valores cercanos al 50 y 60%.

Se puede observar en la gráfica de importancia de los Features de acuerdo al random forest, del cáncer de mama, que los primeros del ranking, no se encuentran en el mismo orden en la grafica de variabilidad (100 repeticiones). además, se observa una alta variabilidad de los Features de mayor importancia.

Lo anterior ocurre de forma similar, aplicando el randomforest al dataset de enfermedad cardiaca. A pesar de esta variabilidad, los modelos mantienen su exactitud, al volver a ejecutar el código, se puede observar un cambio en la importancia de los Features, sin cambiar de forma relevante su desempeño. Lo anterior da cuenta de que existen muchas variables que aportan información redundante al modelo, y que por lo tanto, se podría prescindir de algunas.

**2.3** Seleccionar las variables más relevantes según su criterio y volver a entrenar el random forest con las nuevas variables. Obtener las métricas de desempeño para el conjunto de Test. Concluir.

```
In [31]: clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_training[['concavity_mean',
                    'perimeter_worst', 'concave points_mean', 'area_worst',
                    'concave points_worst']], y_training)
print("Nueva Accuracy cancer", (clf.score(X_test[['concavity_mean',
                                                'perimeter_worst', 'concave points_mean', 'area_worst',
                                                'concave points_worst']], y_test)))

clf1 = RandomForestClassifier(n_estimators=100, random_state=42)
clf1.fit(X_training1[["cp","ca","thal","age","thalach"]], y_training1)
print("Nueva Accuracy corazón",clf1.score(X_test1[["cp","ca","thal","age","thalach"]])
```

Nueva Accuracy cancer 0.9239766081871345

Nueva Accuracy corazón 0.7666666666666667

### Random Forest aplicado a variables seleccionadas

Al aplicar Random forest considerando las variables mejores ranqueadas en orden de importancia, para ambos dataset, se obtienen exactitudes considerablemente mayores, aumentando del orden de 27 puntos porcentuales su desempeño.

**2.4.** Aplicar la técnica de PCA y obtener el gráfico de Biplot. Concluir sobre los resultados.

In [54]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA
import pandas as pd
from sklearn.preprocessing import StandardScaler

plt.figure(figsize=(15,10))

X = variables
y = target

# In general, it's a good idea to scale the data prior to PCA.
scaler = StandardScaler()
scaler.fit(X)
X=scaler.transform(X)
pca = PCA()
x_new = pca.fit_transform(X)

def myplot(score,coeff,X,labels=None):

    xs = score[:,0]
    ys = score[:,1]
    n = coeff.shape[0]
    scalex = 1.0/(xs.max() - xs.min())
    scaley = 1.0/(ys.max() - ys.min())
    plt.scatter(xs * scalex,ys * scaley, c = y)
    for i in range(n):

        plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = 'r',alpha = 0.5)
        if labels is None:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, X.columns[i], color = 'g')
        else:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, labels[i], color = 'g')
    plt.xlim(-.4,.75)
    plt.ylim(-.4,.75)
    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.grid()

#Call the function. Use only the 2 PCs.
myplot(x_new[:,0:2],np.transpose(pca.components_[0:2, :]),X_training)
plt.show()
```



```
In [24]: pca.components_
```

```
Out[24]: array([[ 2.18902444e-01,  1.03724578e-01,  2.27537293e-01,
   2.20994985e-01,  1.42589694e-01,  2.39285354e-01,
   2.58400481e-01,  2.60853758e-01,  1.38166959e-01,
   6.43633464e-02,  2.05978776e-01,  1.74280281e-02,
   2.11325916e-01,  2.02869635e-01,  1.45314521e-02,
   1.70393451e-01,  1.53589790e-01,  1.83417397e-01,
   4.24984216e-02,  1.02568322e-01,  2.27996634e-01,
   1.04469325e-01,  2.36639681e-01,  2.24870533e-01,
   1.27952561e-01,  2.10095880e-01,  2.28767533e-01,
   2.50885971e-01,  1.22904556e-01,  1.31783943e-01],
  [-2.33857132e-01, -5.97060883e-02, -2.15181361e-01,
   -2.31076711e-01,  1.86113023e-01,  1.51891610e-01,
   6.01653628e-02, -3.47675005e-02,  1.90348770e-01,
   3.66575471e-01, -1.05552152e-01,  8.99796818e-02,
   -8.94572342e-02, -1.52292628e-01,  2.04430453e-01,
   2.32715896e-01,  1.97207283e-01,  1.30321560e-01,
   1.83848000e-01,  2.80092027e-01,  -2.19866379e-01,
   -4.54672983e-02, -1.99878428e-01,  -2.19351858e-01,
   1.72304352e-01,  1.43593173e-01,  9.79641143e-02,
   -8.25723507e-03,  1.41883349e-01,  2.75339469e-01],
  [-8.53124284e-03,  6.45499033e-02, -9.31421972e-03,
   2.86995259e-02, -1.04291904e-01, -7.40915709e-02,
   2.73383798e-03, -2.55635406e-02, -4.02399363e-02,
   -2.25740897e-02,  2.68481387e-01,  3.74633665e-01,
   2.66645367e-01,  2.16006528e-01,  3.08838979e-01,
   1.54779718e-01,  1.76463743e-01,  2.24657567e-01,
   2.88584292e-01,  2.11503764e-01,  -4.75069900e-02,
   -4.22978228e-02, -4.85465083e-02, -1.19023182e-02,
   -2.59797613e-01, -2.36075625e-01, -1.73057335e-01,
   -1.70344076e-01, -2.71312642e-01, -2.32791313e-01],
  [ 4.14089623e-02, -6.03050001e-01,  4.19830991e-02,
   5.34337955e-02,  1.59382765e-01,  3.17945811e-02,
   1.91227535e-02,  6.53359443e-02,  6.71249840e-02,
   4.85867649e-02,  9.79412418e-02, -3.59855528e-01,
   8.89924146e-02,  1.08205039e-01,  4.46641797e-02,
   -2.74693632e-02,  1.31687997e-03,  7.40673350e-02,
   4.40733510e-02,  1.53047496e-02,  1.54172396e-02,
   -6.32807885e-01,  1.38027944e-02,  2.58947492e-02,
   1.76522161e-02, -9.13284153e-02, -7.39511797e-02,
   6.00699571e-03, -3.62506947e-02, -7.70534703e-02],
  [ 3.77863538e-02, -4.94688505e-02,  3.73746632e-02,
   1.03312514e-02, -3.65088528e-01,  1.17039713e-02,
   8.63754118e-02, -4.38610252e-02, -3.05941428e-01,
   -4.44243602e-02, -1.54456496e-01, -1.91650506e-01,
   -1.20990220e-01, -1.27574432e-01, -2.32065676e-01,
   2.79968156e-01,  3.53982091e-01,  1.95548089e-01,
   -2.52868765e-01,  2.63297438e-01, -4.40659209e-03,
   -9.28834001e-02,  7.45415100e-03, -2.73909030e-02,
   -3.24435445e-01,  1.21804107e-01,  1.88518727e-01,
   4.33320687e-02, -2.44558663e-01,  9.44233510e-02],
  [ 1.87407904e-02, -3.21788366e-02,  1.73084449e-02,
   -1.88774796e-03, -2.86374497e-01, -1.41309489e-02,
   -9.34418089e-03, -5.20499505e-02,  3.56458461e-01,
   -1.19430668e-01, -2.56032561e-02, -2.87473145e-02,
   1.81071500e-03, -4.28639079e-02, -3.42917393e-01,
   6.91975186e-02,  5.63432386e-02, -3.12244482e-02,
   4.90245643e-01, -5.31952674e-02, -2.90684919e-04,
   -5.00080613e-02,  8.50098715e-03, -2.51643821e-02,
   -3.69255370e-01,  4.77057929e-02,  2.83792555e-02,
```

-3.08734498e-02, 4.98926784e-01, -8.02235245e-02],  
[-1.24088340e-01, 1.13995382e-02, -1.14477057e-01,  
-5.16534275e-02, -1.40668993e-01, 3.09184960e-02,  
-1.07520443e-01, -1.50482214e-01, -9.38911345e-02,  
2.95760024e-01, 3.12490037e-01, -9.07553556e-02,  
3.14640390e-01, 3.46679003e-01, -2.44024056e-01,  
2.34635340e-02, -2.08823790e-01, -3.69645937e-01,  
-8.03822539e-02, 1.91394973e-01, -9.70993602e-03,  
9.87074388e-03, -4.45726717e-04, 6.78316595e-02,  
-1.08830886e-01, 1.40472938e-01, -6.04880561e-02,  
-1.67966619e-01, -1.84906298e-02, 3.74657626e-01],  
[-7.45229622e-03, 1.30674825e-01, -1.86872582e-02,  
3.46736038e-02, -2.88974575e-01, -1.51396350e-01,  
-7.28272853e-02, -1.52322414e-01, -2.31530989e-01,  
-1.77121441e-01, 2.25399674e-02, -4.75413139e-01,  
-1.18966905e-02, 8.58051345e-02, 5.73410232e-01,  
1.17460157e-01, 6.05665008e-02, -1.08319309e-01,  
2.20149279e-01, 1.11681884e-02, 4.26194163e-02,  
3.62516360e-02, 3.05585340e-02, 7.93942456e-02,  
2.05852191e-01, 8.40196588e-02, 7.24678714e-02,  
-3.61707954e-02, 2.28225053e-01, 4.83606666e-02],  
[-2.23109764e-01, 1.12699390e-01, -2.23739213e-01,  
-1.95586014e-01, 6.42472194e-03, -1.67841425e-01,  
4.05910064e-02, -1.11971106e-01, 2.56040084e-01,  
-1.23740789e-01, 2.49985002e-01, -2.46645397e-01,  
2.27154024e-01, 2.29160015e-01, -1.41924890e-01,  
-1.45322810e-01, 3.58107079e-01, 2.72519886e-01,  
-3.04077200e-01, -2.13722716e-01, -1.12141463e-01,  
1.03341204e-01, -1.09614364e-01, -8.07324609e-02,  
1.12315904e-01, -1.00677822e-01, 1.61908621e-01,  
6.04884615e-02, 6.46378061e-02, -1.34174175e-01],  
[ 9.54864432e-02, 2.40934066e-01, 8.63856150e-02,  
7.49564886e-02, -6.92926813e-02, 1.29362000e-02,  
-1.35602298e-01, 8.05452775e-03, 5.72069479e-01,  
8.11032072e-02, -4.95475941e-02, -2.89142742e-01,  
-1.14508236e-01, -9.19278886e-02, 1.60884609e-01,  
4.35048658e-02, -1.41276243e-01, 8.62408470e-02,  
-3.16529830e-01, 3.67541918e-01, 7.73616428e-02,  
2.95509413e-02, 5.05083335e-02, 6.99211523e-02,  
-1.28304659e-01, -1.72133632e-01, -3.11638520e-01,  
-7.66482910e-02, -2.95630751e-02, 1.26095791e-02],  
[ 4.14714866e-02, -3.02243402e-01, 1.67826374e-02,  
1.10169643e-01, -1.37021842e-01, -3.08009633e-01,  
1.24190245e-01, -7.24460264e-02, 1.63054081e-01,  
-3.80482687e-02, -2.53570194e-02, 3.44944458e-01,  
-1.67318771e-01, 5.16194632e-02, 8.42062106e-02,  
-2.06885680e-01, 3.49517943e-01, -3.42375908e-01,  
-1.87844043e-01, 2.50624789e-01, 1.05067333e-01,  
1.31572736e-02, 5.10762807e-02, 1.84598937e-01,  
1.43890349e-01, -1.97420469e-01, 1.85016760e-01,  
-1.17772055e-01, 1.57560248e-01, 1.18283551e-01],  
[ 5.10674568e-02, 2.54896423e-01, 3.89261058e-02,  
6.54375082e-02, 3.16727211e-01, -1.04017044e-01,  
6.56534798e-02, 4.25892667e-02, -2.88865504e-01,  
2.36358988e-01, -1.66879153e-02, -3.06160423e-01,  
-1.01446828e-01, -1.76792177e-02, -2.94710053e-01,  
-2.63456509e-01, 2.51146975e-01, -6.45875122e-03,  
3.20571348e-01, 2.76165974e-01, 3.96796652e-02,  
7.97974499e-02, -8.98773800e-03, 4.80886567e-02,

5.65148662e-02, -3.71662503e-01, -8.70345324e-02,  
-6.81253543e-02, 4.40335026e-02, -3.47316933e-02],  
[ 1.19672116e-02, 2.03461333e-01, 4.41095034e-02,  
6.73757374e-02, 4.55736020e-02, 2.29281304e-01,  
3.87090806e-01, 1.32138097e-01, 1.89933673e-01,  
1.06239082e-01, -6.81952298e-02, -1.68222383e-01,  
-3.78439858e-02, 5.60649318e-02, 1.50441434e-01,  
1.00401699e-02, 1.58783192e-01, -4.94026741e-01,  
1.03327412e-02, -2.40458323e-01, -1.37890527e-01,  
-8.01454315e-02, -9.69657077e-02, -1.01160611e-01,  
-2.05130344e-01, 1.22793095e-02, 2.17984329e-01,  
-2.54387490e-01, -2.56534905e-01, -1.72814238e-01],  
[ 5.95061348e-02, -2.15600995e-02, 4.85138123e-02,  
1.08308292e-02, 4.45064860e-01, 8.10105720e-03,  
-1.89358699e-01, -2.44794768e-01, 3.07388563e-02,  
-3.77078865e-01, 1.03474126e-02, -1.08493473e-02,  
-4.55237175e-02, 8.35707181e-02, -2.01152530e-01,  
4.91755932e-01, 1.34586924e-01, -1.99666719e-01,  
-4.68643826e-02, 1.45652466e-01, 2.31012813e-02,  
5.34307917e-02, 1.22193824e-02, -6.68546458e-03,  
1.62235443e-01, 1.66470250e-01, -6.67989309e-02,  
-2.76418891e-01, 5.35557351e-03, -2.12104110e-01],  
[ 5.11187749e-02, 1.07922421e-01, 3.99029358e-02,  
-1.39669069e-02, 1.18143364e-01, -2.30899962e-01,  
1.28283732e-01, 2.17099194e-01, 7.39617071e-02,  
-5.17975705e-01, 1.10050711e-01, -3.27527212e-02,  
8.26808881e-03, 4.60243656e-02, -1.85594647e-02,  
-1.68209315e-01, -2.50471408e-01, -6.20793442e-02,  
1.13383199e-01, 3.53232211e-01, -1.66567074e-01,  
-1.01115399e-01, -1.82755198e-01, -3.14993600e-01,  
-4.61258656e-02, 4.99560142e-02, 2.04835886e-01,  
1.69499607e-01, -1.39888394e-01, 2.56173195e-01],  
[-1.50583883e-01, -1.57841960e-01, -1.14453955e-01,  
-1.32448032e-01, -2.04613247e-01, 1.70178367e-01,  
2.69470206e-01, 3.80464095e-01, -1.64661588e-01,  
-4.07927860e-02, 5.89057190e-02, -3.45004006e-02,  
2.65166513e-02, 4.11532265e-02, -5.80390613e-02,  
1.89830896e-01, -1.25420649e-01, -1.98810346e-01,  
-1.57711497e-01, 2.68553878e-01, -8.15605686e-02,  
1.85557852e-01, -5.48570473e-02, -9.06533944e-02,  
1.45551659e-01, -1.53734861e-01, -2.15021948e-01,  
1.78141741e-01, 2.57894009e-01, -4.05556492e-01],  
[ 2.02924255e-01, -3.87061187e-02, 1.94821310e-01,  
2.55705763e-01, 1.67929914e-01, -2.03077075e-02,  
-1.59835337e-03, 3.45095087e-02, -1.91737848e-01,  
5.02252456e-02, -1.39396866e-01, 4.39630156e-02,  
-2.46356391e-02, 3.34418173e-01, 1.39595006e-01,  
-8.24647717e-03, 8.46167156e-02, 1.08132263e-01,  
-2.74059129e-01, -1.22733398e-01, -2.40049982e-01,  
6.93651855e-02, -2.34164147e-01, -2.73399584e-01,  
-2.78030197e-01, -4.03712272e-03, -1.91313419e-01,  
-7.54853164e-02, 4.30658116e-01, 1.59394300e-01],  
[ 1.46712338e-01, -4.11029851e-02, 1.58317455e-01,  
2.66168105e-01, -3.52226802e-01, 7.79413843e-03,  
-2.69681105e-02, -8.28277367e-02, 1.73397790e-01,  
8.78673570e-02, -2.36216532e-01, -9.85866201e-03,  
-2.59288003e-02, 3.04906903e-01, -2.31259943e-01,  
1.00474235e-01, -1.95485228e-04, 4.60549116e-02,  
1.87014764e-01, -5.98230982e-02, -2.16101353e-01,

5.83984505e-02, -1.88543592e-01, -1.42064856e-01,  
5.01551675e-01, -7.35745143e-02, -1.03907980e-01,  
7.58138963e-02, -2.78713843e-01, 2.35647497e-02],  
[-2.25384659e-01, -2.97886446e-02, -2.39595276e-01,  
2.73221894e-02, 1.64565843e-01, -2.84222358e-01,  
-2.26636013e-03, 1.54972363e-01, 5.88111647e-02,  
5.81570509e-02, -1.75883308e-01, -3.60098518e-02,  
-3.65701538e-01, 4.16572314e-01, 1.32600886e-02,  
2.42448176e-01, -1.26381025e-01, 1.21642969e-02,  
8.90392949e-02, -8.66008430e-02, -1.36613039e-02,  
7.58669276e-02, -9.08132490e-02, 4.10047202e-01,  
-2.34513845e-01, -2.02007041e-02, 4.57861197e-02,  
2.60229625e-01, -1.17250532e-01, 1.14944811e-02],  
[-4.96986642e-02, -2.44134993e-01, -1.76650122e-02,  
-9.01437617e-02, 1.71009601e-02, 4.88686329e-01,  
-3.33870858e-02, -2.35407606e-01, 2.60691555e-02,  
-1.75637222e-01, -9.08005031e-02, -7.16599878e-02,  
-1.77250625e-01, 2.74201148e-01, 9.00614773e-02,  
-4.61098220e-01, 6.69461742e-02, 6.88682942e-02,  
1.07385289e-01, 2.22345297e-01, -5.62690874e-03,  
3.00599798e-01, 1.10038577e-02, 6.00473870e-02,  
-1.29723903e-01, 2.29280589e-01, -4.64827918e-02,  
3.30223397e-02, -1.16759236e-01, -1.04991974e-01],  
[-6.85700057e-02, 4.48369467e-01, -6.97690429e-02,  
-1.84432785e-02, -1.19491747e-01, 1.92621396e-01,  
5.57175335e-03, -9.42381870e-03, -8.69384844e-02,  
-7.62718362e-02, 8.63867747e-02, 2.17071967e-01,  
-3.04950158e-01, 1.92587786e-01, -7.20987261e-02,  
-1.40386572e-01, 6.30479298e-02, 3.43753236e-02,  
-9.76995265e-02, 6.28432814e-02, 7.29389953e-03,  
-5.94440143e-01, -9.20235990e-02, 1.46790132e-01,  
1.64849237e-01, 1.81374867e-01, -1.32100595e-01,  
8.86081478e-04, 1.62708549e-01, -9.23439434e-02],  
[ 7.29289034e-02, 9.48006326e-02, 7.51604777e-02,  
9.75657781e-02, 6.38229479e-02, -9.80775567e-02,  
-1.85212003e-01, -3.11852431e-01, -1.84067326e-02,  
2.87868885e-01, -1.50274681e-01, 4.84569345e-02,  
1.59352804e-01, 6.42326151e-02, 5.05449015e-02,  
-4.52876920e-02, -2.05212693e-01, -7.25453753e-02,  
-8.46544307e-02, 2.44705083e-01, -9.62982088e-02,  
-1.11112024e-01, 1.72216251e-02, -9.69598236e-02,  
-6.82540931e-02, 2.96764124e-02, 4.60426186e-01,  
2.99840557e-01, 9.71448437e-02, -4.69471147e-01],  
[-9.85526942e-02, -5.54997454e-04, -4.02447050e-02,  
7.77727342e-03, -2.06657211e-02, 5.23603957e-02,  
3.24870378e-01, -5.14087968e-02, -5.12005770e-02,  
-8.46898562e-02, -2.64125317e-01, -8.73880467e-04,  
9.00742110e-02, 9.82150746e-02, -5.98177179e-02,  
9.10387102e-03, -3.87542329e-01, 3.51755074e-01,  
-4.23628949e-02, 8.57810992e-02, -5.56767923e-02,  
-8.92289971e-03, 6.33448296e-02, 1.90889625e-01,  
9.36901494e-02, -1.47920925e-01, 2.86433135e-01,  
-5.67527797e-01, 1.21343451e-01, 7.62533821e-03],  
[ 1.82579441e-01, -9.87867898e-02, 1.16648876e-01,  
-6.98483369e-02, -6.86974224e-02, 1.04135518e-01,  
-4.47410568e-02, -8.40276972e-02, -1.93394733e-02,  
1.33260547e-01, 5.58701567e-01, -2.42672970e-02,  
-5.16750385e-01, 2.24607172e-02, -1.56311888e-02,  
1.21777792e-01, -1.88205036e-01, 1.09668978e-01,

-3.22620011e-03, -7.51944193e-02, 1.56830365e-01,  
1.18484602e-01, -2.37113167e-01, -1.44063033e-01,  
1.09901386e-02, -1.86749953e-01, 2.88852570e-01,  
-1.07340243e-01, 1.43818093e-02, -3.78254532e-02],  
[ 1.92264989e-02, -8.47459309e-02, -2.70154137e-02,  
2.10040780e-01, -2.89548850e-02, -3.96623231e-01,  
9.69773167e-02, 1.86451602e-01, 2.45836949e-02,  
2.07221864e-01, 1.74930429e-01, -5.69864778e-02,  
-7.29276412e-02, -1.31850405e-01, -3.12107028e-02,  
-1.73164553e-01, -1.59399802e-02, 1.29546547e-01,  
1.95149333e-02, 8.41712034e-02, -7.07097238e-02,  
1.18189721e-01, -1.18034029e-01, 3.82899511e-02,  
4.79647647e-02, 6.24384938e-01, -1.15770341e-01,  
-2.63196337e-01, -4.52996243e-02, -2.80133485e-01],  
[-1.29476396e-01, -2.45566636e-02, -1.25255946e-01,  
3.62727403e-01, -3.70036864e-02, 2.62808474e-01,  
-5.48876170e-01, 3.87643377e-01, -1.60440385e-02,  
-9.74048386e-02, 4.99770798e-02, -1.12372419e-02,  
1.03653282e-01, -1.55304589e-01, -7.71755717e-03,  
-4.97276317e-02, 9.14549680e-02, -1.79419192e-02,  
-1.72678486e-02, 3.54889745e-02, -1.97054744e-01,  
3.64694332e-02, -2.44103670e-01, 2.31359525e-01,  
1.26024637e-02, -1.00463424e-01, 2.66853781e-01,  
-1.33574507e-01, 2.81842956e-02, 4.52048188e-03],  
[-1.31526670e-01, -1.73573093e-02, -1.15415423e-01,  
4.66612477e-01, 6.96899233e-02, 9.77487054e-02,  
3.64808397e-01, -4.54699351e-01, -1.51648349e-02,  
-1.01244946e-01, 2.12982901e-01, -1.00928890e-02,  
4.16915529e-02, -3.13358657e-01, -9.05215355e-03,  
4.65360884e-02, -8.42247975e-02, -1.11655093e-02,  
-1.99759830e-02, -1.20365640e-02, -1.78666740e-01,  
2.14106944e-02, -2.41031046e-01, 2.37162466e-01,  
-4.08535683e-02, -7.05054136e-02, -1.42905801e-01,  
2.30901389e-01, 2.27904438e-02, 5.99859979e-02],  
[ 2.11194013e-01, -6.58114593e-05, 8.43382663e-02,  
-2.72508323e-01, 1.47926883e-03, -5.46276696e-03,  
4.55386379e-02, -8.88309714e-03, 1.43302642e-03,  
-6.31168651e-03, -1.92223890e-01, -5.62261069e-03,  
2.63191868e-01, -4.20681051e-02, 9.79296328e-03,  
-1.53955481e-02, 5.82097800e-03, -2.90093001e-02,  
-7.63652550e-03, 1.97564555e-02, 4.12639581e-01,  
-3.90250926e-04, -7.28680898e-01, 2.38960316e-01,  
-1.53524821e-03, 4.86918180e-02, -1.76408967e-02,  
2.24756680e-02, 4.92048082e-03, -2.35621424e-02],  
[ 2.11460455e-01, -1.05339342e-02, 3.83826098e-01,  
-4.22794920e-01, -3.43466700e-03, -4.10167739e-02,  
-1.00147876e-02, -4.20694931e-03, -7.56986244e-03,  
7.30143287e-03, 1.18442112e-01, -8.77627920e-03,  
-6.10021933e-03, -8.59259138e-02, 1.77638619e-03,  
3.15813441e-03, 1.60785207e-02, -2.39377870e-02,  
-5.22329189e-03, -8.34191154e-03, -6.35724917e-01,  
1.72354925e-02, 2.29218029e-02, 4.44935933e-01,  
7.38549171e-03, 3.56690392e-06, -1.26757226e-02,  
3.52404543e-02, 1.34042283e-02, 1.14776603e-02],  
[-7.02414091e-01, -2.73661018e-04, 6.89896968e-01,  
3.29473482e-02, 4.84745766e-03, -4.46741863e-02,  
-2.51386661e-02, 1.07726530e-03, 1.28037941e-03,  
4.75568480e-03, 8.71109373e-03, 1.07103919e-03,  
-1.37293906e-02, -1.10532603e-03, 1.60821086e-03,

```
-1.91562235e-03,  8.92652653e-03,  2.16019727e-03,
-3.29389752e-04, -1.79895682e-03,  1.35643056e-01,
-1.02053601e-03, -7.97438536e-02, -3.97422838e-02,
-4.58327731e-03,  1.28415624e-02, -4.02139168e-04,
 2.28844179e-03, -3.95443454e-04, -1.89429245e-03]])
```

In [25]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA
import pandas as pd
from sklearn.preprocessing import StandardScaler

plt.figure(figsize=(15,10))

X = variables1
y = target1

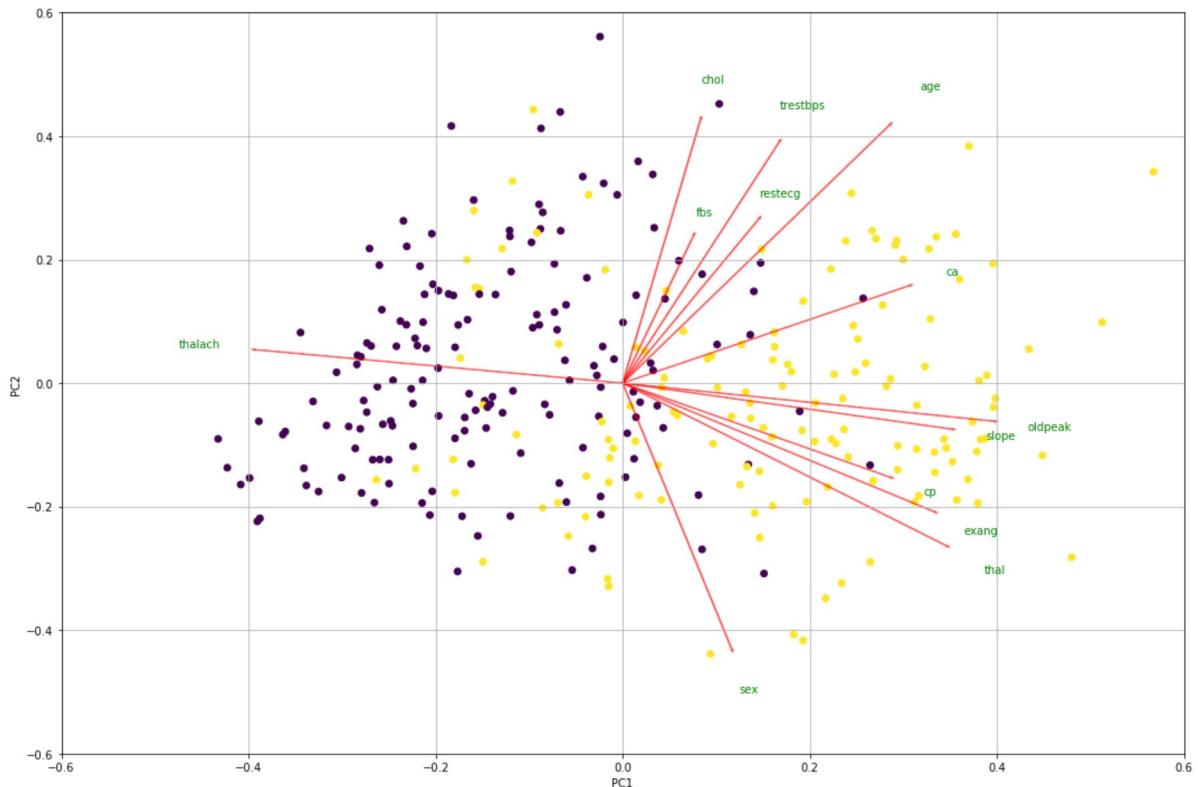
# In general, it's a good idea to scale the data prior to PCA.
scaler = StandardScaler()
scaler.fit(X)
X=scaler.transform(X)
pca = PCA()
x_new = pca.fit_transform(X)

def myplot(score,coeff,X,labels=None):

    xs = score[:,0]
    ys = score[:,1]
    n = coeff.shape[0]
    scalex = 1.0/(xs.max() - xs.min())
    scaley = 1.0/(ys.max() - ys.min())
    plt.scatter(xs * scalex,ys * scaley, c = y)
    for i in range(n):

        plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = 'r',alpha = 0.5)
        if labels is None:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, X.columns[i], color = 'g')
        else:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, labels[i], color = 'g')
    plt.xlim(-.6,.6)
    plt.ylim(-.6,.6)
    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.grid()

#Call the function. Use only the 2 PCs.
myplot(x_new[:,0:2],np.transpose(pca.components_[0:2, :]),X_training1)
plt.show()
```



```
In [26]: pca.components_
```

```
Out[26]: array([[ 0.28586758,  0.11680189,  0.28621211,  0.16784082,  0.08350419,
   0.07609387,  0.14593784, -0.39270583,  0.33313953,  0.39700637,
   0.3520241 ,  0.30641953,  0.34620842],
 [ 0.41866153, -0.431579 , -0.15253955,  0.39151404,  0.42820138,
  0.23985575,  0.26651507,  0.05414681, -0.2082855 , -0.06174634,
 -0.07449081,  0.15784224, -0.26344384],
 [-0.01261961,  0.48369265, -0.40887336,  0.31463151, -0.2600399 ,
  0.51477575,  0.07196171,  0.21882111, -0.19663643,  0.0454429 ,
  0.00149273,  0.0914224 ,  0.25484219],
 [-0.12419567, -0.2554797 , -0.32658925,  0.18685395, -0.19084194,
 -0.15145133,  0.000173282, -0.02090758, -0.11956303,  0.40946106,
 0.58672122, -0.40292598, -0.17142789],
 [ 0.33301471, -0.22056993, -0.03559127, -0.08463981, -0.40076951,
 0.21497083, -0.66229013, -0.30341471, -0.17926828, -0.02905481,
 -0.00673187,  0.21507271, -0.12336624],
 [-0.13706318, -0.12415037,  0.10476835,  0.4907731 ,  0.29293212,
 0.08406217, -0.55713847,  0.16833653,  0.30589048, -0.03499864,
 -0.09831845, -0.30932236,  0.28672087],
 [-0.20894989, -0.20719272,  0.2699967 , -0.07172239, -0.23630352,
 0.65404224,  0.23702123, -0.10906038,  0.34977143, -0.18173861,
 0.07847726, -0.2569787 , -0.23689505],
 [-0.32869113, -0.13528374,  0.03321693, -0.432931 ,  0.41035697,
 0.3508859 , -0.2089899 ,  0.28011342, -0.18170022,  0.30024537,
 0.23697419,  0.29864425,  0.05446431],
 [-0.25119348, -0.24877202,  0.44845459,  0.36336173, -0.4200732 ,
 -0.14381862,  0.08953717,  0.35828613, -0.16004309,  0.18949657,
 -0.11082038,  0.36570195, -0.02825065],
 [-0.12518455,  0.01019962, -0.4428583 ,  0.03297055, -0.04219898,
 -0.08321121, -0.07463762,  0.09239813,  0.63884822,  0.18838756,
 -0.1636798 ,  0.38034537, -0.38617433],
 [ 0.01953097,  0.53895838,  0.34113269,  0.17185439,  0.18867857,
 0.05519569, -0.17043459, -0.06083638, -0.15381371,  0.24964437,
 -0.02358603, -0.12136357, -0.62496031],
 [-0.60490431, -0.02300399, -0.12953711,  0.26132766,  0.14275303,
 -0.0222217 ,  0.01228126, -0.62631898, -0.21415943, -0.1897538 ,
 -0.03065266,  0.2288092 , -0.00520083],
 [ 0.03123119,  0.15025525,  0.06591852,  0.11444853,  0.02741043,
 -0.11587936, -0.10400283,  0.22559294,  0.09137349, -0.61145481,
 0.64318794,  0.26045956, -0.1391407 ]])
```

## PCA

Se puede observar en el gráfico Biplot una coincidencia entre las variables de mayor importancia según el randomforest y las variables cuyos vectores atraviesan una mayor variabilidad de los puntos en el hiperplano y por lo tanto, las variables que permiten la separación de clases.

En el caso de los datos de cancer, como existe correlación entre las variables de entrada al modelo, los vectores observados tienen magnitud y dirección similares entre sí. De acuerdo a la gráfica, se pueden destacar las siguientes variables: perimeter\_worst, concave points\_mean y area\_worst.

En el caso de los datos de cancer, donde existen menores variables de entrada, se pueden identificar claramente las variables de mayor relevancia para la separación de clases. se pueden destacar las siguientes variables: thalach, ca y cp.

### **Conclusión de la actividad:**

Al evaluar los distintos modelos (clasificadores y clustering) en el dataset de diferentes características, se puede concluir que el desempeño de los modelos no mejora necesariamente en función de su complejidad, es decir, existen modelos más adecuados para las características y objetivos que se tienen para el conjunto de datos. En este caso particular, los modelos más simples, tuvieron el mejor desempeño para el dataset de menores volumen de datos, variable y variabilidad.

En el caso de la data más compleja, se obtuvo desempeños altos para todos los modelos evaluados, destacando un modelo de clustering.

Al aplicar clustering jerárquico, y randomforest, se pudo identificar con claridad, las variables que representan un mayor peso para predicción del target.

Respecto del PCA, se pudo comprobar de forma gráfica la coincidencia de las variables de mayor importancia detectadas con el clustering. Si se hubiera aplicado inicialmente el PCA, se podría haber seleccionado las variables predictoras de mayor importancia para la aplicación de los modelos.