Facultad de Ciencias, UNAM
Reconocimiento de patrones y aprendizaje automatizado
Tarea 2

Rubí Rojas Tania Michelle

15 de febrero de 2021

1. Cada una de las líneas en el documento `tarea2_docs.txt` lo vamos a considerar como un documento. Realiza la limpieza y preprocesamiento necesarios.

   SOLUCIÓN: Leemos el archivo **tarea_docs.txt** línea por línea (ya que cada una de éstas será considerada como un documento) y lo guardamos en un arreglo. Posteriormente, convertimos este arreglo en un `DataFrame` cuya única columna se llamará `'text'` y cada una de sus filas contendrá los distintos textos guardados.

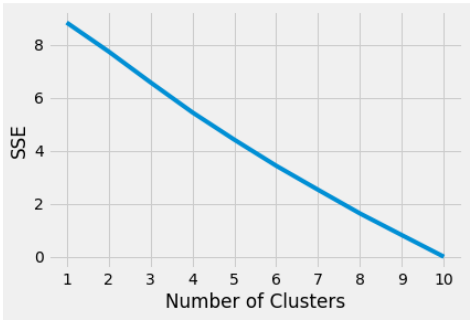
	text
0	El perro del doctor Dillon es muy astuto, es u...
1	¿El doctor Jekyll es el alter ego de Mr. Hyde ...
2	El maíz significa un principio vital y un elem...
3	En los organismos celulares desempeña diversas...
4	Según el modelo estándar (SM, de sus siglas en...
5	En comparación con lobos de tamaño equivalente...
6	La isla del tesoro (Treasure Island) es una no...
7	Y es que el maíz ha tenido un papel fundamenta...
8	El dogma central de la biología molecular es u...
9	Hasta la fecha, casi todas las pruebas experim...

   Luego, convertimos cada uno de los documentos en vectores usando TFIDF.

```
vectorizer = TfidfVectorizer(stop_words=get_stop_words('spanish'))
vectorizer.fit(df.text.values)
features = vectorizer.transform(df.text.values)
```

   Por otro lado, usaremos el algoritmo **K-means** para agrupar nuestros datos. De esta forma, para encontrar el valor adecuado de K realizamos una gráfica e intenamos hallar el *Elbow Curve*, para lo cual ejecutamos varios *k-means*, incrementamos a k en cada iteración y registramos el valor del SSE.

```
# A list holds the SSE values for each k
sse = []
for k in range(1, 11):
    kmeans = KMeans(init='k-means++', n_clusters=k)
    kmeans.fit(features)
    sse.append(kmeans.inertia_)
```



donde podemos notar que en $k = 5$ se dobla ligeramente la curva. Así, ejecutamos **K-means** con este valor, realizamos nuestras predicciones y modificamos el `DataFrame` de tal forma que agregamos la nueva clasificación encontrada.

	text	label
0	El perro del doctor Dillon es muy astuto, es u...	0
1	¿El doctor Jekyll es el alter ego de Mr. Hyde ...	0
2	El maíz significa un principio vital y un elem...	1
3	En los organismos celulares desempeña diversas...	4
4	Según el modelo estándar (SM, de sus siglas en...	2
5	En comparación con lobos de tamaño equivalente...	3
6	La isla del tesoro (Treasure Island) es una no...	0
7	Y es que el maíz ha tenido un papel fundamenta...	1
8	El dogma central de la biología molecular es u...	4
9	Hasta la fecha, casi todas las pruebas experim...	2

Como podemos observar, los textos se clasificaron en 5 grupos diferentes. Separamos cada texto en un `DataFrame` distinto para poder visualizar los resultados.

	text	label
2	El maíz significa un principio vital y un elem...	1
7	Y es que el maíz ha tenido un papel fundamenta...	1

	text	label
4	Según el modelo estándar (SM, de sus siglas en...	2
9	Hasta la fecha, casi todas las pruebas experim...	2

	text	label
5	En comparación con lobos de tamaño equivalente...	3

	text	label
3	En los organismos celulares desempeña diversas...	4
8	El dogma central de la biología molecular es u...	4

	text	label
0	El perro del doctor Dillon es muy astuto, es u...	0
1	¿El doctor Jekyll es el alter ego de Mr. Hyde ...	0
6	La isla del tesoro (Treasure Island) es una no...	0

Podemos observar que obtuvimos una muy buena agrupación, pues a mi consideración, sólo agrupó erróneamente un texto (y se encuentra en el grupo 3).

Contesta las siguientes preguntas:

- ¿En qué tópicos se pueden dividir?
  SOLUCIÓN: Se puede dividir en 5 diferentes tópicos:
  - Aquellos con la etiqueta 1, que corresponden a temas relacionados con el maíz en los pueblos indígenas y la cultura mexicana.
  - Aquellos con la etiqueta 2, que corresponden a temas relacionados con la física.
  - Aquel con la etiqueta 3, que corresponde a la descripción de lobos (el cual debería de estar relacionado con el texto que habla del perro del Doctor Dillon, pues hablan de cosas similares).
  - Aquellos con la etiqueta 4, que corresponden a temas de biología molecular.
  - Aquellos con la etiqueta 0, que corresponden a descripciones de libros (con un integrante erróneo).
- ¿Qué documentos habla de México y su cultura?
  SOLUCIÓN: Los documentos etiquetados con el número 1.

2. Descarga el dataset de datos de sonar de la siguiente liga: `https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/sonar/sonar.all-data`

   Mismo que deberás explorar y entender lo que hacen sus atributos.

   Lleva a cabo una clasificación utilizándo la regresión logística de los datos. Luego, efectúa una reducción de dimensión y haz tu clasificación otra vez. La comparación entre las clasificaciones la harás con las métricas que tú elijas.

   ¿Qué observas y por qué sucede esto?

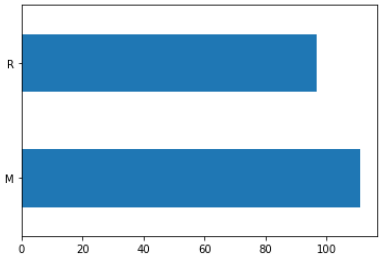
   SOLUCIÓN: **sonar.all-data** es un conjunto de datos que describe el sonido (o más bien, el chirrido) del sonar al rebotar sobre diferentes superficies. Las 59 variables de entrada son la fuerza de los retornos en ángulos diferentes y se encuentran en un rango de 0 a 1. La variable 60 de salida es un string **M** para Mina y **R** para Roca.

   El problema es de clasificación binaria, la cual requiere de un modelo para diferenciar rocas de cilindros metálicos.

   Descargamos el conjunto de datos y lo visualizamos.

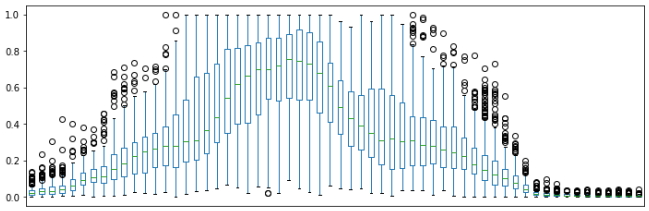
0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	56	57	58	59	60
0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.0027	0.0065	0.0159	0.0072	0.0167	0.0180	0.0084	0.0090	0.0032	R
0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.0084	0.0089	0.0048	0.0094	0.0191	0.0140	0.0049	0.0052	0.0044	R
0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.0232	0.0166	0.0095	0.0180	0.0244	0.0316	0.0164	0.0095	0.0078	R
0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.0121	0.0036	0.0150	0.0085	0.0073	0.0050	0.0044	0.0040	0.0117	R
0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.0031	0.0054	0.0105	0.0110	0.0015	0.0072	0.0048	0.0107	0.0094	R
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.0187	0.0346	0.0168	0.0177	0.0393	0.1630	0.2028	0.1694	0.2328	0.2684	...	0.0116	0.0098	0.0199	0.0033	0.0101	0.0065	0.0115	0.0193	0.0157	M
0.0323	0.0101	0.0298	0.0564	0.0760	0.0958	0.0990	0.1018	0.1030	0.2154	...	0.0061	0.0093	0.0135	0.0063	0.0063	0.0034	0.0032	0.0062	0.0067	M
0.0522	0.0437	0.0180	0.0292	0.0351	0.1171	0.1257	0.1178	0.1258	0.2529	...	0.0160	0.0029	0.0051	0.0062	0.0089	0.0140	0.0138	0.0077	0.0031	M
0.0303	0.0353	0.0490	0.0608	0.0167	0.1354	0.1465	0.1123	0.1945	0.2354	...	0.0086	0.0046	0.0126	0.0036	0.0035	0.0034	0.0079	0.0036	0.0048	M
0.0260	0.0363	0.0136	0.0272	0.0214	0.0338	0.0655	0.1400	0.1843	0.2354	...	0.0146	0.0129	0.0047	0.0039	0.0061	0.0040	0.0036	0.0061	0.0115	M

Veamos si la proporción de los datos en cada clase es equilibrada.

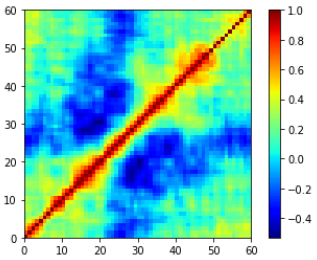


donde existen 111 datos para la clase Mina y 97 datos para la clase Roca. Como no hay mucha diferencia entre ambas clases, entonces no es necesario rebalancear de alguna manera el conjunto de datos.

Aunque los metadatos dicen que todos los predictores van de 0 a 1, todavía encontramos que hay varios de ellos que tienen valores relativamente menores en comparación con la mayoría. También encontramos que no hay ningún valor alto más grande que 1.



La matriz de correlación nos indica que hay alguna estructura en el órden de los atributos: el color rojo alrededor de la diagonal sugiere que los atributos que están uno al lado del otro están más correlacionados entre sí (es decir, cada variable tiene una gran correlación positiva con sus vecinos), las manchas azules sugieren alguna correlación negativa moderada entre los atributos que están más alejados unos de otros. Esto tiene sentido si el órden de los atributos se refiere al ángulo de los sensores para el chirrido del sonar.



Ahora bien, para el procesamiento de datos separamos al conjunto de datos en dos:

- X: el cual está conformado por las primeras 59 columnas del dataset original.
- y: el cual está conformado por la columna 60 del dataset original.

La gráfica de cajas nos sugiere que debemos estandarizar los datos de entrada, por lo que usamos la clase **StandardScaler** para lograr esto (recordándo que ésta elimina la media y escala los datos de forma que su varianza sea igual a 1).

La variable de salida (la columna 60) es un string **M** o **R**. Debemos convertirlos en valores enteros entre 0 y 1, y para lograr esto usamos la clase **LabelEncoder** (la cual modelará la codificación requerida usando todo el conjunto de datos a traves de la función `fit()` y luego aplicará la codificación para crear una nueva variable de salida usando la función `transform()`).

Para crear el modelo de regresión lineal, dividimos nuestro conjunto de datos ya modificado en entrenamiento (80 %) y prueba (20 %). Luego, simplemente creamos una instancia de `LogisticRegression` y lo entrenamos.

Los resultados obtenidos fueron los siguientes:

```
print('Accuracy of LR classifier on training set: {:.2f}'
     .format(logistic_regression.score(X_train, y_train)))
print('Accuracy of LR classifier on test set: {:.2f}'
     .format(logistic_regression.score(X_test, y_test)))
```

```
Accuracy of LR classifier on training set: 0.90
Accuracy of LR classifier on test set: 0.88
```

```
print("Mean Absolute Error: %.2f" % mean_absolute_error(y_test, y_pred))
# Error Cuadrado Medio
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error: %.2f" % np.sqrt(mean_squared_error(y_test, y_pred)))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 0.12
Mean squared error: 0.12
Root Mean Squared Error: 0.35
Variance score: 0.52
```

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.86	0.90	0.88	21
1	0.90	0.86	0.88	21
accuracy			0.88	42
macro avg	0.88	0.88	0.88	42
weighted avg	0.88	0.88	0.88	42

donde en particular podemos notar que

- Obtenemos un 90 % de precisión en el conjunto de entrenamiento y un 88 % de precisión en el conjunto de prueba.
- Obtenemos MSE de 0.12 (con un puntaje de varianza de 0.52).
- Obtenemos 86 % de precisión para los datos correspondientes a metales y 90 % de precisión para los datos correspondientes a las rocas.

Por otro lado, usaremos PCA para obtener la lista de atributos que tienen mayor varianza (mayor poder explicativo). Éstos serán los componentes principales. Realizamos un par de gráficas para poder determinar el número de estos componentes.

Donde obtenemos que con 43 componentes tenemos un 99 % de la varianza explicada (queremos una v.e. entre 95 % y 99 %, pero me ha gustado más al resultado con este valor). Así, creamos una instancia de PCA con un número de 43 componentes principales y la entrenamos, para después pasarle este nuevo conjunto X_pca al modelo de `LogisticRegression` y entrenarlo.

Los resultados obtenidos fueron los siguientes:

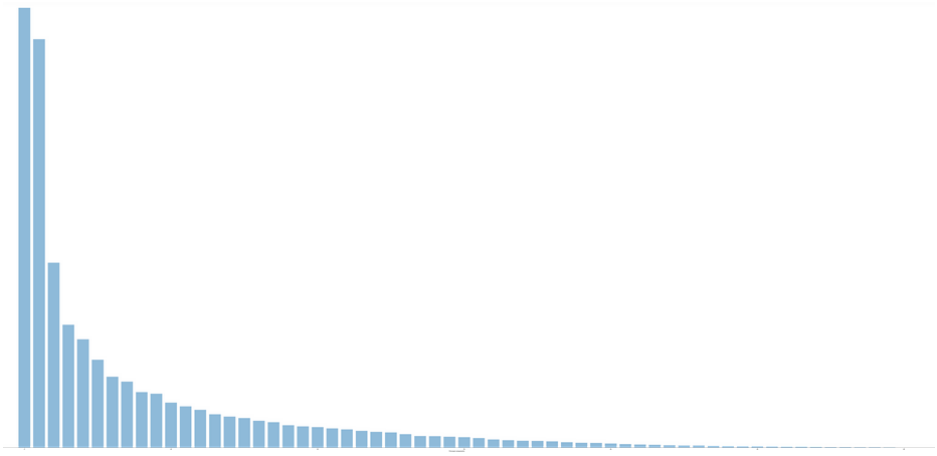


Figura 1: Variance Ratio vs Principal componentes

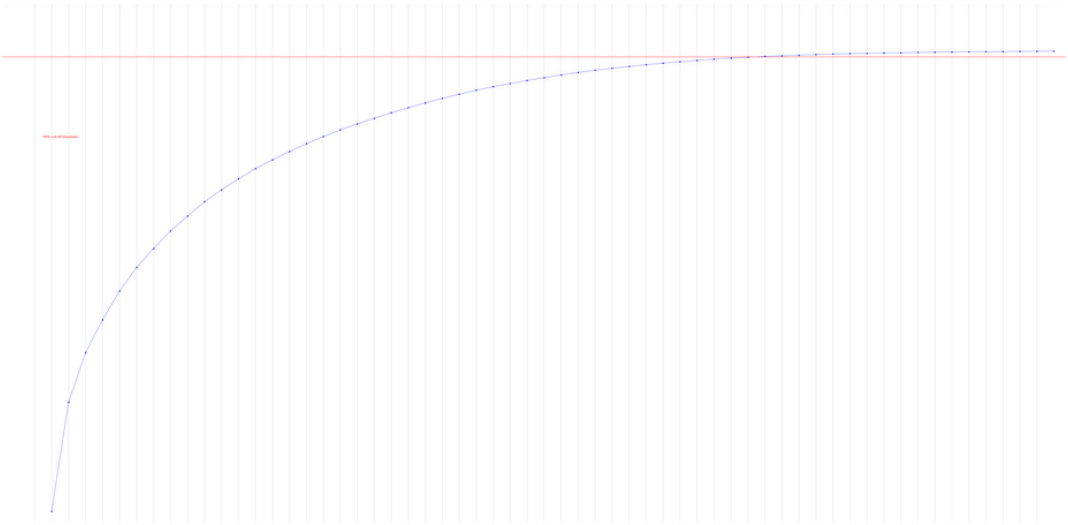


Figura 2: Cumulative variance vs Number of Components

```
print('Accuracy of LR classifier on training set: {:.2f}'
     .format(logistic_regression.score(X_train, y_train)))
print('Accuracy of LR classifier on test set: {:.2f}'
     .format(logistic_regression.score(X_test, y_test)))
```

```
Accuracy of LR classifier on training set: 0.90
Accuracy of LR classifier on test set: 0.88
```

```
print("Mean Absolute Error: %.2f" % mean_absolute_error(y_test, y_pred))
# Error Cuadrado Medio
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error: %.2f" % np.sqrt(mean_squared_error(y_test, y_pred)))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 0.12
Mean squared error: 0.12
Root Mean Squared Error: 0.35
Variance score: 0.52
```

	precision	recall	f1-score	support
0	0.86	0.90	0.88	21
1	0.90	0.86	0.88	21
accuracy			0.88	42
macro avg	0.88	0.88	0.88	42
weighted avg	0.88	0.88	0.88	42

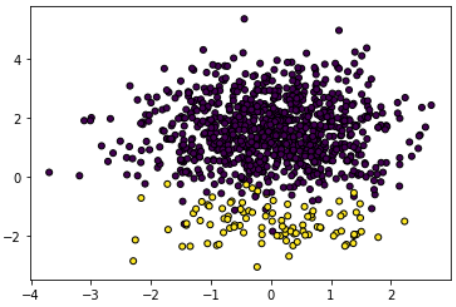
donde en particular podemos notar que

- Obtenemos un 90 % de precisión en el conjunto de entrenamiento y un 88 % de precisión en el conjunto de prueba.
- Obtenemos MSE de 0.12 (con un puntaje de varianza de 0.52).
- Obtenemos 86 % de precisión para los datos correspondientes a metales y 90 % de precisión para los datos correspondientes a las rocas.

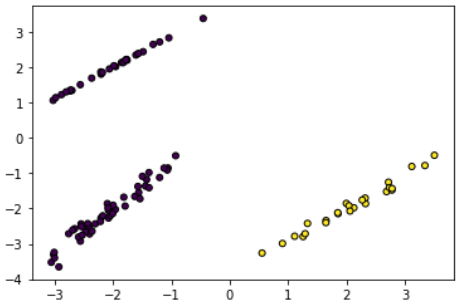
Así, podemos concluir que obtenemos la misma precisión que en el modelo anterior usando 43 componentes principales (16 columnas menos). Esto puede ser causado por el nivel de correlación que notamos al analizar los datos.

3. Diseña un experimento para determinar para qué umbral es posible tener un conjunto de datos no balanceado.

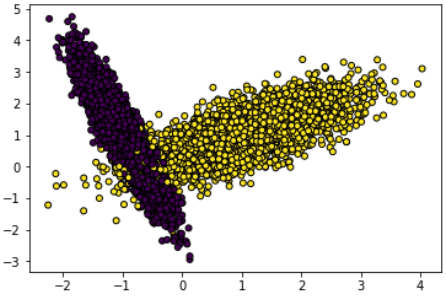
   Solución: Utilizamos la clase `make_classification`, la cual genera conjuntos de datos a partir de ciertos parámetros que nosotros podemos modificar. Al crear 9 distintos datsets, obtenemos las siguientes gráficas:



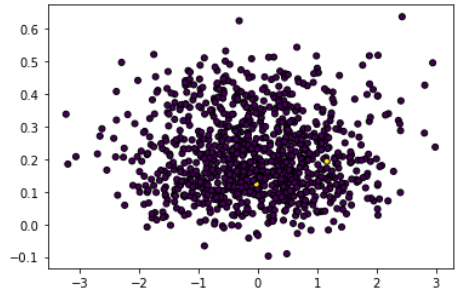
(a) `100 datos con {900, 100}`



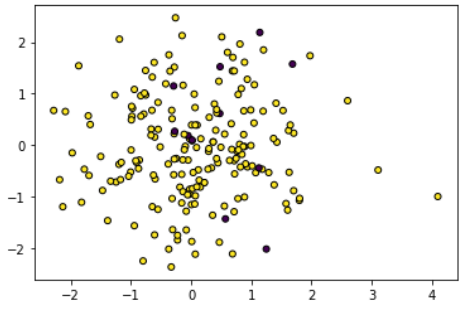
(b) `1000 datos con {75, 25}`



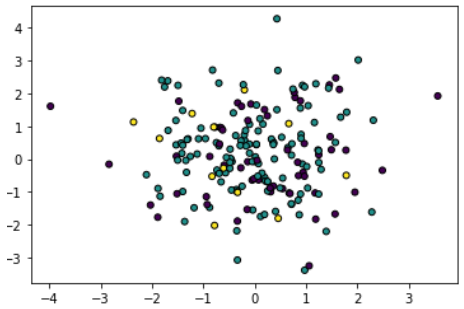
(a) `10000 datos con {6000, 4000}`



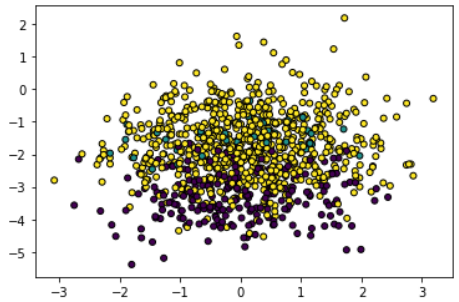
(b) `1000 datos con {997, 3}`



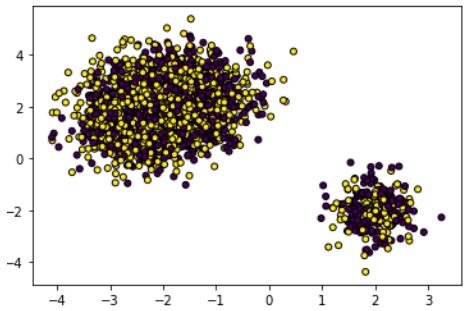
(a) `200 datos con {11, 189}`



(b) `200 datos con {57, 130, 13}`



(a) `1000 datos con {227, 47, 726}`



(b) `3575 datos con {1630, 1945}`

Ahora bien, al estar jugando con varios de los parámetros notamos que aquellos que verdaderamente influyen para que un conjunto de datos sea desbalanceado son **flip_y** y **weights** mientras que el resto de los parámetros sólo parecen influir en las características de los datos (más no en la distribución de valores en cada clase).

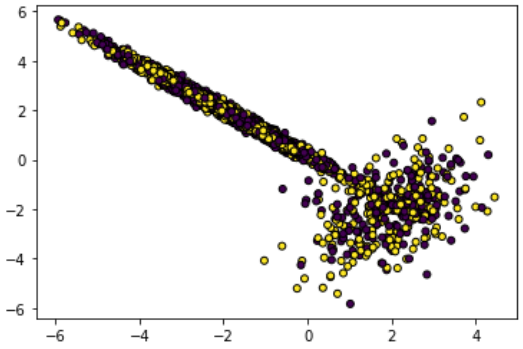
El parámetro **weight** sirve para poder asignar la proporción de valores que tendrá cada una de las clases. De esta forma, si queremos un conjunto de datos desbalanceado, debemos ajustar estos valores para que así sea. Por ejemplo, si tenemos 100 datos y queremos una proporción $60 - 40$ (y tenemos dos clases) entonces debemos escribir `weight[0.6,0.4]`.

La suma de las proporciones no debe ser necesariamente 1, y en algunos casos, cuando este valor es mayor a 1, se regresan más valores de lo que deberían (y en un experimento pudimos ver que todas las clases toman casi los mismos valores cuando esto pasa).

En particular, encontramos que si **flip_y** es mayor a 0 entonces lo que hace es ir aumentando los valores de las clases (como si se estuviera balanceando) poco a poco. Con un valor **flip_y** $<$ **0.5** realmente el conjunto sigue desbalanceado, pero conforme el valor se acerca a 1 entonces los valores de las clases se van *balanceando*. Esto último lo podemos ver con el siguiente conjunto

```
X8, y8 = make_classification(flip_y=1, n_informative=2, n_repeated=0,
                             n_clusters_per_class=1, n_samples=4000,
                             shuffle=True, n_features=2, n_redundant=0,
                             random_state=7, class_sep=2, n_classes=2,
                             weights=[0.1, 0.9])
```

donde podemos observar que a pesar de que las proporciones de los datos son `1:9`, gracias a que el valor de **flip_y** es igual a 1, las proporciones de las clases son totalmente diferentes (pues obtenemos que los valores son `{2065, 1935}`)



Creo además que el número de clusters también puede influir, pero realmente desvía los valores de las clases muy pero muy poquito, entonces no afecta mucho en el umbral que buscamos.

4. El dataset de `seatbelt.csv` representa una serie de datos temporales de diversos atributos que hacen referencia a la mortalidad asociada a los accidentes de tránsito en Gran Bretaña en el periodo de 1969 y 1984. La legislación para utilizar el cinturón de seguridad de manera obligatoria fue introducida el 31 de enero de 1983. Ajusta un modelo lineal generalizado para determinar la probabilidad de morir en un accidente de tránsito. Se espera que lleves a cabo la normalización o estandarización del dataset, la transformación de las variables que consideres pertinente y que propongas qué variables tienen más o menos importancia en la probabilidad de morir en un accidente de tránsito. Responde las siguientes preguntas:

   - ¿Volver obligatorio el cinturón de seguridad disminuyó la probabilidad de morir en algún accidente de tránsito?
   - ¿Qué otra conclusión puedes generar a partir del modelo que ajustaste?

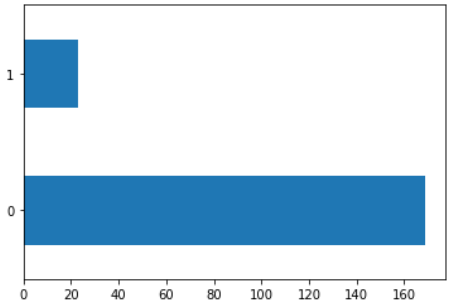
   Solución: Notemos que nuestro conjunto de datos se ve de la siguiente forma:

	DriversKilled	drivers	front	rear	kms	PetrolPrice	VanKilled	law
0	107	1687	867	269	9059	0.102972	12	0
1	97	1508	825	265	7685	0.102363	6	0
2	102	1507	806	319	9963	0.102062	12	0
3	87	1385	814	407	10955	0.100873	8	0
4	119	1632	991	454	11823	0.101020	10	0
...	...	...	...	...	...	...	...	...
187	96	1284	644	521	21626	0.114797	5	1
188	122	1444	643	429	20195	0.114093	7	1
189	120	1575	641	408	19928	0.116466	7	1
190	137	1737	711	490	18564	0.116026	4	1
191	154	1763	721	491	18149	0.116067	7	1

   el cual contiene 192 entradas y 8 atributos. Como la tarea principal del conjunto de datos se centrará en el antes y el después de la introducción de la legislación sobre el cinturón de seguridad, es conveniente dividir el conjunto de datos en dos:

   - Uno antes de la legislación (`law=0`) y,
   - Otro después de la legislación (`law=1`)

   Contando el número de entradas en cada uno de estos conjuntos obtenemos que hay 169 entradas para `law=0`, mientras que `law=1` tiene únicamente 23 entradas.



Luego, vemos la proporción de conductores fallecidos antes y después de que usar cinturón de seguridad fuera obligatorio.

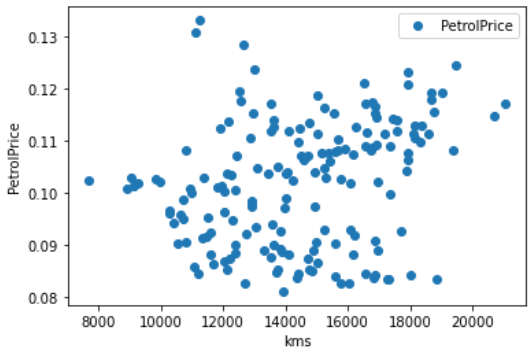
- Pasajeros al frente y detrás:
  - Sin ley, la probabilidad es de 0.7415.
  - Con ley, la probabilidad es de 0.6169.
- Pasajeros al frente:
  - Sin ley, la probabilidad es de 0.5084.
  - Con ley, la probabilidad es de 0.4319.
- Pasajeros detrás:
  - Sin ley, la probabilidad es de 0.2330.
  - Con ley, la probabilidad es de 0.3084.

Por lo tanto, podemos concluir que efectivamente disminuyó la probabilidad de morir en un accidente de auto. Podemos notar además que la probabilidad de morir siendo pasajero detrás aumentó con la legislación.

Por otro lado, para calcular la correlación y la regresión lineal entre dos factores se utilizó el conjunto de datos **sinley** (pues tiene la mayor cantidad de entradas), esto para que los datos no estuvieran sesgados por los diferentes factores.

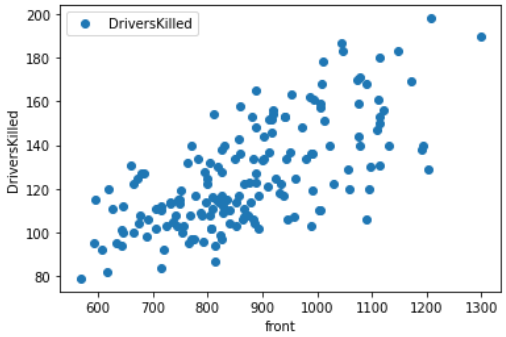
De esta forma, procedemos a calcular la correlación entre los atributos.

- Kilometraje vs Precio de la gasolina
  Obtenemos una correlación de 0.2454.

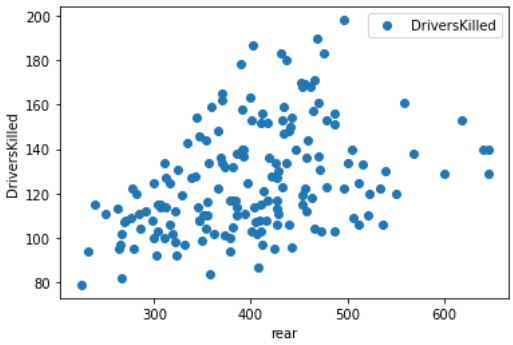


  Donde, a pesar de esperar que la correlación fuera negativa (pues a mayor precio de la gasolina, menos distancia se recorrería); se obtuvo un valor positivo.

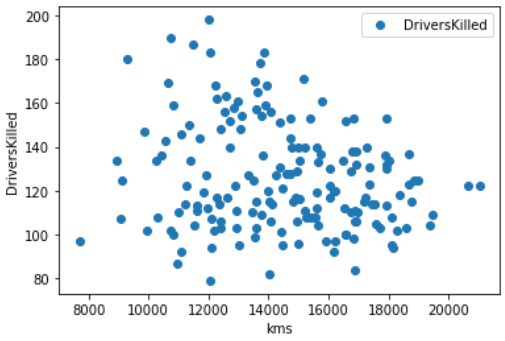
- Pasajeros al frente vs número de fallecidos.
  Obtenemos una correlación de 0.6808.



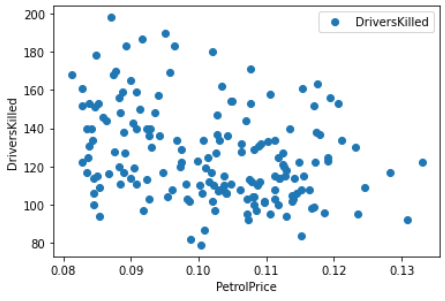
- Pasajeros en los asientos traseros vs número de fallecidos.
  Obtenemos una correlación de 0.6981.



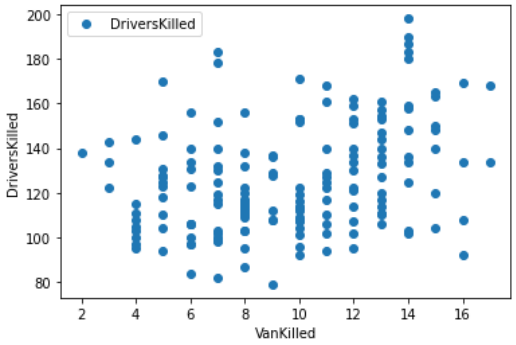
- Kilometraje vs número de fallecidos.
  Obtenemos una correlación de -0.1914.



- Precio de la gasolina vs número de fallecidos.
  Obtenemos una correlación de -0.3120.



- Número de furgonetas vs número de fallecidos.
  Obtenemos una correlación de 0.3382.



Notemos que las variables independientes parecen ser Kms, PetrolPrice, Front y Rear, mientras que la variable dependiente será DriversKilled.

Después, definimos dos nuevos conjuntos:

- X: correspondiente a las variables independientes.
- y: correspondiente a la variable dependiente.

Y separamos nuestro conjunto en prueba y entrenamiento en un $20 - 80$, respectivamente. Creamos una instancia del algoritmo `LinearRegression`, y lo entrenamos.

Los coeficientes obtenidos son:

	Coefficient
front	0.170932
rear	-0.107635
kms	0.000775
PetrolPrice	117.617962

Esto significa que,

- Cuando `DriverKilled` incrementa una unidad, entonces `front` incrementa 0.170932 unidades.
- Cuando `DriverKilled` incrementa una unidad, entonces `rear` disminuye -0.107635 unidades.
- Cuando `DriverKilled` incrementa una unidad, entonces `kms` incrementa 0.000775 unidades.
- Cuando `DriverKilled` incrementa una unidad, entonces `PetrolPrice` incrementa 117.617962 unidades.

Realizando nuestras predicciones, obtenemos los siguientes resultados:

	Actual	Predicted
138	102	118.157880
30	130	149.698408
119	178	148.096824
29	148	136.896331
143	137	134.645197
162	122	121.154351
165	137	136.965789
51	131	157.444694
105	140	119.056342
60	113	113.445734
15	102	130.249461
157	104	106.040791
133	95	93.698940
45	170	152.560420
68	144	155.337627
85	127	102.610783
24	134	139.199952

y visualizamos la precisión de nuestro modelo:

```
print("Mean Absolute Error: %.2f" % mean_absolute_error(y_test, y_pred))
# Error Cuadrado Medio
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error: %.2f" % np.sqrt(mean_squared_error(y_test, y_pred)))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 15.09
Mean squared error: 327.90
Root Mean Squared Error: 18.11
Variance score: 0.36
```

A partir de esto, podemos concluir que el modelo realiza predicciones considerablemente buenas.