Trabajo Fin de Máster: Visualizador de conjuntos de datos

Javier Cano Montero

6 de julio de 2018

Resumen

Visualizador...

Índice general

Capítulo 1

Introducción

El universo que habitamos es tetradimensional, es decir, tiene tres dimensiones espaciales y una temporal. Esto nos limita a la hora de poder visualizar objetos (o datos) de mayor dimensión, debido a nuestra incapacidad física de poder imaginar más de tres dimensiones espaciales. Es uso habitual el realizar proyecciones de objetos tridimensionales a bidimensionales.

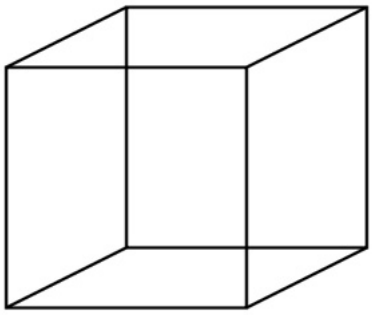

Figura 1.1: Cubo en 2 dimensiones

En Machine Learning la reducción de dimensionalidad es el proceso de reducir el número de variables de un conjunto a un subconjunto de variables manteniendo la mayor cantidad de información posible. La transformación de datos puede ser lineal o no lineal.

Capítulo 2

Objetivo

Crear una aplicación web que permita realizar visualizaciones de conjuntos de datos de alta dimensión tras aplicar redución de dimensionalidad. La aplicación debe presentar las siguientes características:

- Carga de ficheros desde el ordenador del usuario.
- Especificar características del dataset (variable target, separador, porcentaje de entrenamiento)
- Selección de las variables que se quieren visualizar.
- Selección de los algoritmos de reducción de dimensionalidad que se quieren utilizar.
- Visualizar gráficamente el conjunto de datos tras aplicar cada algoritmo de reducción de dimensionalidad.
- Para los algoritmos lineales mostrar cómo cada variable seleccionada es proyectada sobre las componentes principales.
- Para los algoritmos lineales calcular cómo de bien separa linealmente los datos el algoritmo de reducción de dimensionalidad.

Capítulo 3

Desarrollo

3.1. Aplicación web

Para crear la aplicación web existen varios frameworks para Python. Por familiaridad y las opciones que presta se decidió utilizar Flask.

Flask es un framework que permite crear una aplicación capaz de responder a llamadas HTTP con funciones Python usando decoradores. Estas funciones pueden devolver páginas web enteras o completar una parte de una plantilla hecha previamente. En este proyecto, se han usado ambas aproximaciones, la página de carga y preparación de datos es una web estática devuelta por una función; y el resto de la aplicación es una plantilla sobre la que se muestran las visualizaciones.

Además, y aunque no era un objetivo del proyecto, Flask permite crear webs responsive para la correcta visualización en dispositivos móviles.

3.2. Carga y preparación de datos

La pantalla de carga de datos permite subir un fichero a la aplicación web, especificar la variable objetivo, el separador del csv y el porcentaje de observaciones que se usarán para entrenamiento. Actualmente la aplicación solo acepta ficheros CSV.

Figura 3.1: Pantalla de carga de datos

El fichero se guarda en una carpeta local de la aplicación web para poder leerlo y transformarlo en un panel de datos de Python o Pandas.

Para mostrar el desarrollo de la aplicación se usará un dataset que se distribuyó en Kaggle bajo la licencia Creative Commons 4. El dataset es un conjunto de datos del juego FIFA 18 (https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset) que contiene las estadísticas de todos los jugadores disponibles. Entre las estadísticas aparecen la media general del jugador, la edad, el potencial, la aceleración, agresividad, estadísticas de portero, reacción, pases cortos y largos, potencia de disparo, etc... Para nuestros ejemplos utilizaremos la variable media general (Overall) como variable objetivo y el resto las utilizaremos para realizar la reducción de dimensionalidad con los algoritmos implementados en la aplicación.

3.2.1. Pandas

Pandas es la librería de creación de paneles de datos de Python. Entre sus funciones, permite leer ficheros CSV y su manipulación.

En el proyecto se ha usado Pandas para leer el fichero que se sube a la aplicación, usando el separador proporcionado por el usuario.

Una vez leídos los datos se realizan las siguientes operaciones:

1. Se realiza un muestreo estratificado si el usuario lo indica.
2. Se carga en una lista la variable objetivo del conjunto de datos y se elimina del panel.
3. Se transforman las variables a numéricas. Si se carga alguna variable categórica (que no sea la variable target) en el conjunto de datos la aplicación mostrará un error.
4. Se normalizan los datos (Media 0, desv. típica 1).

5. Se parten los datos en un conjunto de entrenamiento y pruebas. El porcentaje de observaciones que se incluyen en el conjunto de entrenamiento es especificado por el usuario en la pantalla de carga de datos.

3.3. Selección de variables y algoritmos de reducción de dimensionalidad

Una vez cargados y preparados los datos, se muestra la pantalla para la selección de variables y algoritmos de reducción de dimensionalidad.

En esta pantalla se mostrarán todas las variables del conjunto de datos menos la variable objetivo especificada por el usuario en la pantalla de carga de datos. El usuario podrá seleccionar cuáles de estas variables se pasarán al algoritmo de reducción de dimensionalidad.

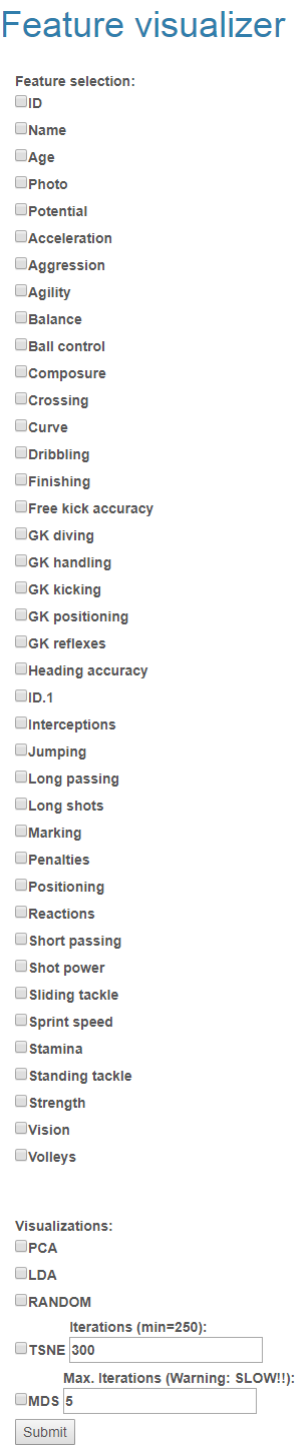

Figura 3.2: Pantalla de selección de variables y algoritmos

También se muestran los algoritmos de reducción de dimensionalidad que se pueden ejecutar sobre las variables seleccionadas del conjunto de datos.

3.3.1. Scikit-learn

Scikit-learn es la librería de machine learning más utilizada de Python. Entre los métodos que incluye se encuentran los que necesitamos para realizar reducción de dimensionalidad y los clasificadores lineales.

3.3.1.1. Algoritmos de reducción de dimensionalidad

Lineales

Estos algoritmos buscan un sistema de referencia de dimensión inferior al original sobre el que proyectar el conjunto de datos. El criterio para encontrar ese sistema de referencia depende del algoritmo.

En los algoritmos lineales, se puede observar cómo cada una de las variables forma parte de la nueva proyección. En estos algoritmos se obtienen componentes lineales, formadas por un escalar y un ángulo, que permiten explicar cómo cada variable es tomada en cuenta a la hora de crear el nuevo sistema de referencia.

Scikit-Learn ofrece una API muy similar para los algoritmos de reducción de dimensionalidad, por lo que se pueden ejecutar de forma similar:

1. **Crear objeto**. En el constructor especificamos que el espacio destino en el que proyectaremos los datos es de dimensión 2.
2. **Realizar fit**. Con los datos de entrenamiento, construimos el espacio de destino.
3. **Transformar train**. Proyectamos los datos de entrenamiento sobre el espacio de destino.
4. **Transformar test**. Proyectamos los datos de prueba sobre el espacio de destino.

Una vez se ha realizado la reducción de dimensionalidad del conjunto de datos, se pasan a la librería bokeh para representarlos gráficamente.

Principal Component Analysis Busca la proyección en la que los datos queden representados en términos de mínimos cuadrados. Para ello, se realiza una descomposición en autovalores y se calcula la matriz de covarianza. Los nuevos ejes sobre los que se proyectan los datos son las componentes principales. La primera componente principal esla transformación lineal que maximiza la varianza de los datos, la segunda es la segunda mayor combinación, etc...

Linear Discriminant Analysis Algoritmo de clasificación que generaliza el discriminante lineal de Fisher, un algoritmo que busca una combinación lineal que separe distintas clases de objetos o eventos. La combinación puede ser usada como un clasificador lineal y para realizar una reducción de dimensionalidad. LDA busca modelar la diferencia de la clase de datos, mientras PCA no la tiene en cuenta.

Proyección aleatoria Gaussiana Es una proyección en la que las componentes de proyección (escalar y ángulo) se toman al azar.

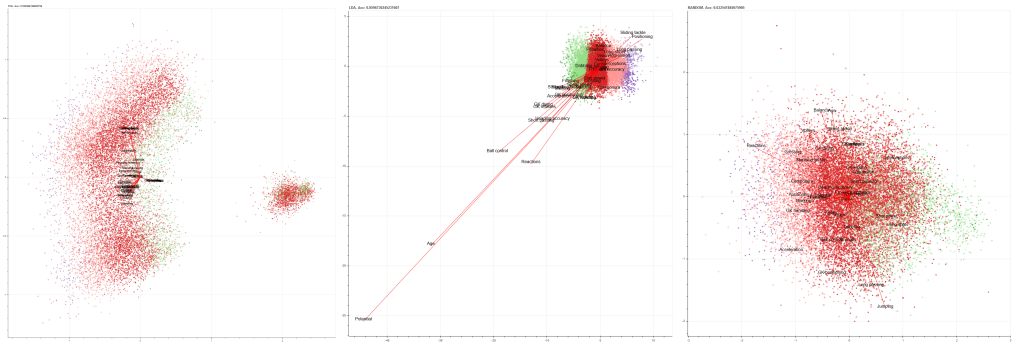

Figura 3.3: (1) PCA (2) LDA (3) RANDOM

En los métodos lineales, podemos ver cómo las distintas variables afectan a la construción de las componentes principales. En caso de PCA, como el criterio de construcción del espacio de dimensión reducida es maximizar la varianza, podemos ver que separa a los jugadores en dos tipos. Una gran nube centrada y otra pequeña que se centra en valores altos de la primera componente principal. Estudiando las componentes observamos que esta pequeña nube de jugadores tiene valores altos en las variables GK diving, GK handling, GK kicking, GK positioning y GK reflexes; es decir variables de portero (Goal Keeper). Como LDA tiene en cuenta la variable objetivo a la hora de crear el espacio de dimensión reducida no se observa esta separación. En la proyección aleatoria, como las componentes son elegidas al azar, no hay nada que resaltar.

No lineales

Multi-Dimensional Scaling Busca qué factores (dimensiones) subyacen bajo los datos obtenidos en el conjunto de datos. Muy utilizado en marketing y ciencias sociales. Construye una matriz de distancias con los datos y reconstruye en el espacio objetivo el conjunto de datos manteniendo las distancias.

T-Distributed Stochastic Neighbour Embeding (T-SNE) Observaciones cercanas en alta dimensión están próximos en el espacio objetivo. Tiene dos

etapas:

1. Construir una probabilidad de distribución sobre los puntos en alta dimensión
2. Define una probabilidad de distribución similar sobre el espacio objetivo.

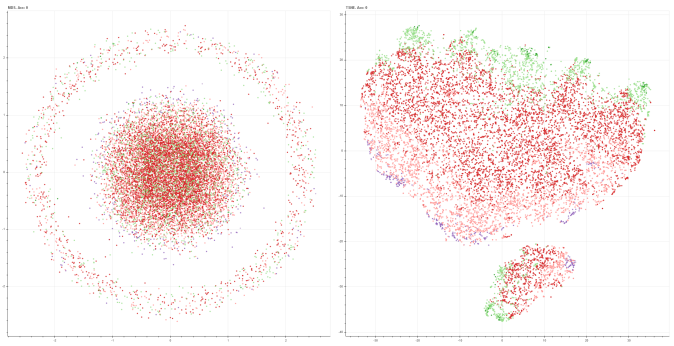

Figura 3.4: (1) MDS (2) T-SNE

En los métodos no lineales se puede observar que se forman dos nubes de puntos. Ambos casos son iguales que en PCA, los porteros forman su propia nube. Para verificarlo, puede resaltarse en la visualización de PCA a los porteros y observar cómo se resaltan en estas visualizaciones.

3.3.1.2. Clasificador lineal (SVM)

Para poder saber cómo de bien separan los algoritmos de reducción de dimensionalidad los datos, se ha optado por ejecutar una SVM con kernel lineal con los datos en dimensión reducida en los algoritmos lineales, excepto LDA porque ya es un algoritmo de clasificación. Los datos de entrenamiento de dimensión reducida se usan para entrenar la SVM y se predice el conjunto de datos de prueba. Con las predicciones se calcula el accuracy (acierto/total).

Una vez realizada la reducción de dimensionalidad y calculado el accuracy del clasificador lineal se crean las visualizaciones.

3.3.2. Bokeh

Bokeh es una librería para creación de visualizaciones en Python que permite interactividad e integrarse fácilmente con Flask. Para crear las visualizaciones se cargan los datos en una estructura de datos propia de Bokeh (ColumnDataSource) que permite vincular distintas gráficas, permitiéndonos resaltar puntos en una gráfica y que estos puntos queden resaltados en el resto de gráficas.

La aplicación recibe de la pantalla de selección de variables y algoritmos una lista con los algoritmos que tiene que representar, realiza la reducción de dimensionalidad y crea las figuras, que son incluídas en una lista. Si se desea visualizar un algoritmo de reducción de dimensionalidad lineal, también se extraen las componentes principales.

Finalmente, con Bokeh se crea un "contenedor" que incluye todas las figuras y se devuelve el renderizado de una plantilla con las gráficas incluídas. En las gráfica los datos de entrenamiento se muestran como cuadrados y los datos de test como círculos.

Capítulo 4

Caso de uso: Surtido retail

Para demostrar la utilidad de la herramienta se va a realizar un análisis del surtido de una cadena de hipermercados. El surtido se divide en una complicada jerarquía (sector, sección, categoría, familia, subfamilia...) en la que se puede clasificar cada producto. Además, tiene otras características como si se vende en mostrador, la marca, descripciones del producto, etc... Visualizar este conjunto de datos nos permitiría encontrar errores en la clasificación dentro de la jerarquía y proponer una nueva distribución para simplificar la compleja jerarquía y usarlo en proyectos como optimización de layouts, segmentación de clientes...

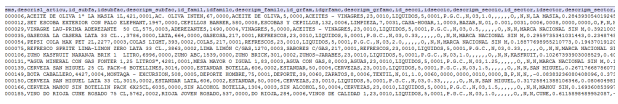

Figura 4.1: Muestra del dataset

El dataset contiene una gran cantidad de NAs, ya que no todos los productos tienen las mismas características, dispone de gran cantidad de variables categóricas y descripciones de los propios productos. Para poder realizar una visualización se optó por realizar un word2vec de cada producto. El word2vec nos permite representar un dataset de alta complejidad en un conjunto reducido de dimensiones (20 en nuestro caso). Los detalles de la construcción de este word2vec quedan fuera del alcance de este proyecto, ya que forma parte de los procesos internos de la empresa.

Para realizar las visualizaciones usaremos un muestreo estratificado del 10 % del dataset y como variable objetivo usaremos el identificador del sector del producto.

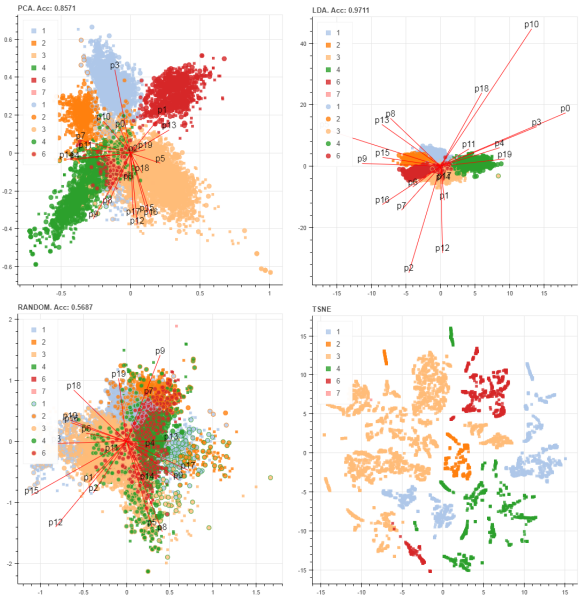

Figura 4.2: (1) PCA (2) LDA (3) RANDOM (4) TSNE

Se puede observar en las visualizaciones lineales que los sectores tienen cierto solape en algunos artículos. Este solape se debe a que muchos artículos similares pertenecen a distintos sectores. Se puede observar en T-SNE que artículos del mismo sector aparecen en posiciones opuestas en la representación, esto refuerza la hipótesis de que artículos similares pertenecen a sectores distintos. También se observa que en T-SNE aparecen distintos clusters. Cada uno de estos clusters pertenece a las categorías inferiores de la jerarquía de productos.

Capítulo 5

Conclusiones y trabajo futuro

Se ha creado una aplicación web que permite realizar visualizaciones de conjuntos de datos de alta dimensión tras aplicar redución de dimensionalidad para realizar exploraciones y poder caracterizar rápidamente visualizaciones de los métodos de reducción de dimensionalidad no lineales usando Bokeh sobre Flask.

En el futuro se propone implementar otros clasificadores no lineales e introducir mejoras en la experiencia de usuario como por ejemplo ser capaz de seleccionar en la visualización las variables que se van a introducir en siguientes visualizaciones.