Ingeniería Informática – Tecnologías Informáticas

Inteligencia Empresarial

2018-2019

Google Play Analytics

Daniel Moreno Soto

Francisco José Alonso Parejo

Francisco Jesús Belmonte Pintre

1. Introducción

Hay numerosos datasets disponibles, en plataformas como Kaggle, acerca de los datos sobre Apple App Store, pero no se encuentran fácilmente disponibles datasets sobre Google Play Store. iTunes App Store, en su web, ofrece una estructura perfectamente indexada que permite un scrapping sencillo. Por otra parte, Google Play Store utiliza una estructura basada en técnicas sofisticadas y modernas con JQuery que hacen del scrapping una tarea más difícil.

Los datasets sobre aplicaciones móviles, en la actualidad, tienen un gran potencial para realizar estudios de mercado y encontrar oportunidades de negocio. Estos estudios beneficiarían tanto a Google como a las empresas que se dedican a realizar estos estudios, puesto que conseguirían mayor audiencia.

En nuestro grupo de trabajo contamos con desarrolladores Android, a los que les sería de utilidad conocer las preferencias de los usuarios de aplicaciones móviles. Además, estos datos podrían generar ingresos de forma directa, ya que tienen un gran potencial y ofrecen significativas ventajas a las empresas que dispongan de ellos.

2. Sobre el dataset

El dataset se encuentra en forma de CSV. Contiene toda la información relevante acerca de las aplicaciones, dividida en 13 columnas, las cuales son:

1. App (String)
   Nombre de la aplicación.
2. Category (String)
   Categoría de la aplicación.
3. Rating (Decimal)
   Rating medio de la aplicación.
4. Reviews (Integer)
   Número de reviews de la aplicación.
5. Size (String)
   Tamaño de la aplicación.
6. Installs (String)
   Número de instalaciones de la aplicación.
7. Type (String)
   Tipo de la aplicación (de pago o gratis).
8. Price (String)
   Precio de la aplicación.
9. Content Rating (String)
   Grupo de edades a los que la aplicación va dirigida.

10. Genres (String)
    Género o géneros de la aplicación.

11. Last Updated (Date)
    Fecha en la que la aplicación fue actualizada por última vez en PlayStore.

12. **Current Ver (String)**
    Versión actual de la aplicación en PlayStore.

13. **Android Ver (String)**
    Versión mínima requerida de Android para la aplicación.

3. Trabajando sobre el dataset

Para establecer una primera toma de contacto con el dataset, se plantea obtener información básica con el primer fichero, como el número total de aplicaciones, el número total de posibles valores en el campo categoría, tipos, versiones de Android, etc.

Además, se obtiene la media de rating por categoría, la media de rating de todas las apps, entre otros aspectos.

A continuación. se procederá a describir el trabajo desempeñado durante las primeras semanas de trabajo, así como muestras del código desarrollado.

En primer lugar, se crea una clase "**App**" para poder generar un objeto por cada aplicación dentro del fichero. Los atributos de estos objetos vienen dados por los valores descritos anteriormente. Los tipos de las variables pueden diferir de los descritos anteriormente, puesto que el proyecto aún se encuentra en una versión temprana.

En segundo lugar, se crea un object "**FileIO**" que nos permita procesar el csv. Se obtiene un string por cada fila del fichero, es decir, un string por cada aplicación del dataset. Cada string separa los campos mediante comas, pero cada campo puede contener o no comas en su interior. Por ello, se hace uso de una expresión regular para poder separarlos.

Cabe destacar que algunos de los campos de las aplicaciones contienen el valor "NaN", por lo que, en una primera aproximación, se procede a descartar aquellas aplicaciones que contengan este valor en alguno de sus valores.

Este fichero devolverá un objeto RDD[App], que servirá para tratar todas las aplicaciones de manera más sencilla.

El objeto es el siguiente:

```
import org.apache.log4j.{Level, Logger}
import org.apache.spark.rdd.RDD
import org.apache.spark.sql.SparkSession

import scala.collection.mutable

object FileIO extends {

  val spark: SparkSession = SparkSession.builder()
    .appName("Spark GooglePlayStore Session")
    .master("local[2]")
    .getOrCreate()
  Logger.getRootLogger.setLevel(Level.ERROR)

  def readFile(filePath: String): RDD[App] = {
    val originalRDD = spark.sparkContext.textFile(filePath)
    val apps = originalRDD.map(row =>
row.split(",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)"))

    // Get the apps
    val appList = apps.filter(a => a.length == 13).filter(a =>
!a.contains("NaN")).map(a => new App(a(0), a(1), a(2).toDouble,
      a(3).toInt, a(4), a(5), a(6), a(7), a(8), a(9), a(10), a(11),
a(12)))

    appList
  }
}
```

Una vez tratados los datos, es decir, generada un conjunto con todo lo que interesa para el proyecto, se procede a generar un objeto que permita obtener información a partir de dicha lista.

El siguiente fichero (“**Main**”) contiene funciones que aportan información relevante acerca de las aplicaciones. Entre ellas encontramos funciones sobre el número de aplicaciones por instalaciones, el número de aplicaciones por categoría, el número total de aplicaciones para versiones Android concretas, etc.

A continuación, se muestra un fragmento del código del fichero mencionado, en el que se obtiene el número de apps en función de las versiones Android mínimas para las que están disponibles, así como el número de versiones mínimas distintas:

```
val numberOfAppsByAndroidVersion: RDD[(String, Int)] = appList.map(app
=> (app.androidVer, 1)).reduceByKey(_ + _)

val numberOfAndroidVersions: Long =
numberOfAppsByAndroidVersion.keys.count()
```

El problema mencionado durante el seguimiento, relacionado con el formato en el que se presentan el número de instalaciones, es resuelto utilizando dos vertientes: una que devuelva estas instalaciones en el formato inicial, es decir, en String; y otra que formatee las String iniciales en números enteros.

De esta manera se puede actuar de las dos formas posibles, en función de las necesidades.

4. Representación gráfica con Vegas-Viz

Vegas-Viz es una API de Scala para la visualización de datos estadísticos contenidos en datasets, sobre los que permite realizar diferentes filtros y transformaciones.

Para hacer uso de Vegas-Viz es necesario importar las siguientes dependencias:

```
scalaVersion := "2.11.8"

libraryDependencies ++= Seq(
  "org.vegas-viz" %% "vegas" % "0.3.9",
  "org.vegas-viz" %% "vegas-spark" % "0.3.9"
)
```

Se crea un nuevo fichero "Plot", el cual es el encargado de transformar los datos obtenidos anteriormente en gráficas, mediante Vegas-Viz. La función "plot_graph" es la encargada de generar gráficos de barra, mientras que en el main se le pasan los datos que se quieren representar. En el fichero, es necesario añadir la siguiente línea para importar la funcionalidad de Vegas-Viz:

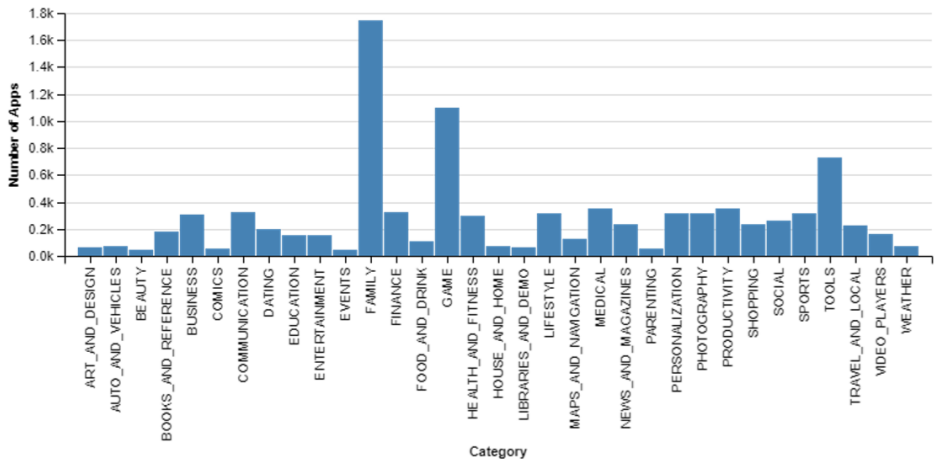
```
import vegas._
```

La función en la que se introducirán los datos, encargada de representarlos, es la siguiente:

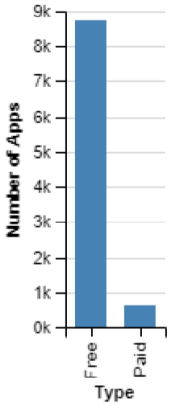
```
def plot_graph(dataMap: RDD[(String, Int)], k: String, v: String):
Unit = {
  val plot = Vegas(k + " " + v, width = 600.0, height = 400.0).
    withData(
      dataMap.collect().map(x => Map(k -> x._1, v -> x._2))
    ).
    encodeX(k, Nom).
    encodeY(v, Quant).
    mark(Bar)
  plot.window.show
}
```

Por último, se muestran algunos gráficos generados con el código anteriormente descrito.

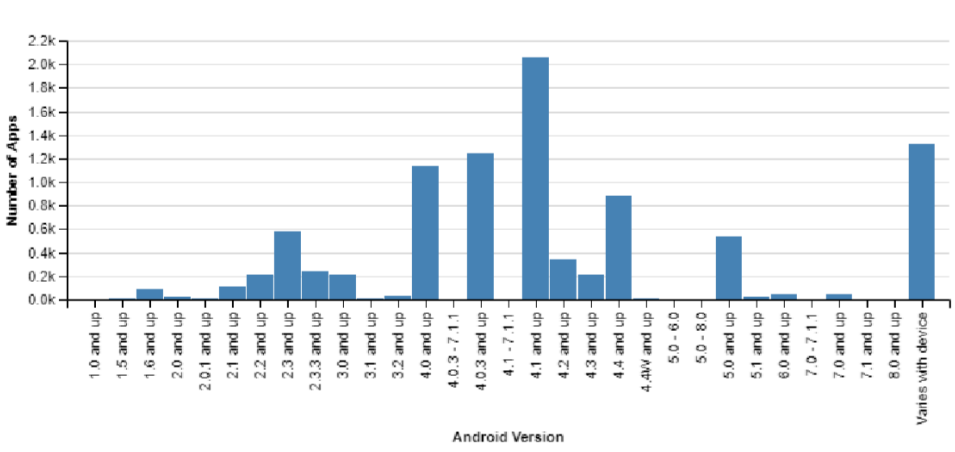
- `plot_graph(appsNumberByCategory, "Category", "Number of Apps")`



- `plot_graph(appsNumberByType, "Type", "Number of Apps")`



- `plot_graph(appsNumberByAndroidVers, "Android Version", "Number of Apps"`



5. Interfaz gráfica

Para ofrecer una visualización de los resultados de una manera más sencilla, se desarrolla una interfaz gráfica de usuario en Scala con Swing.

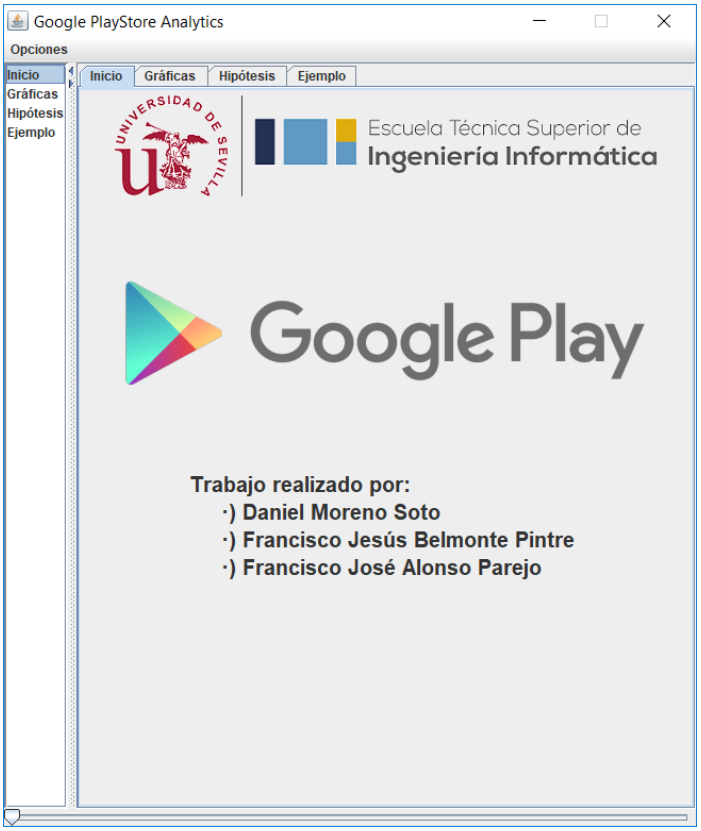
La aplicación cuenta con 4 pestañas:

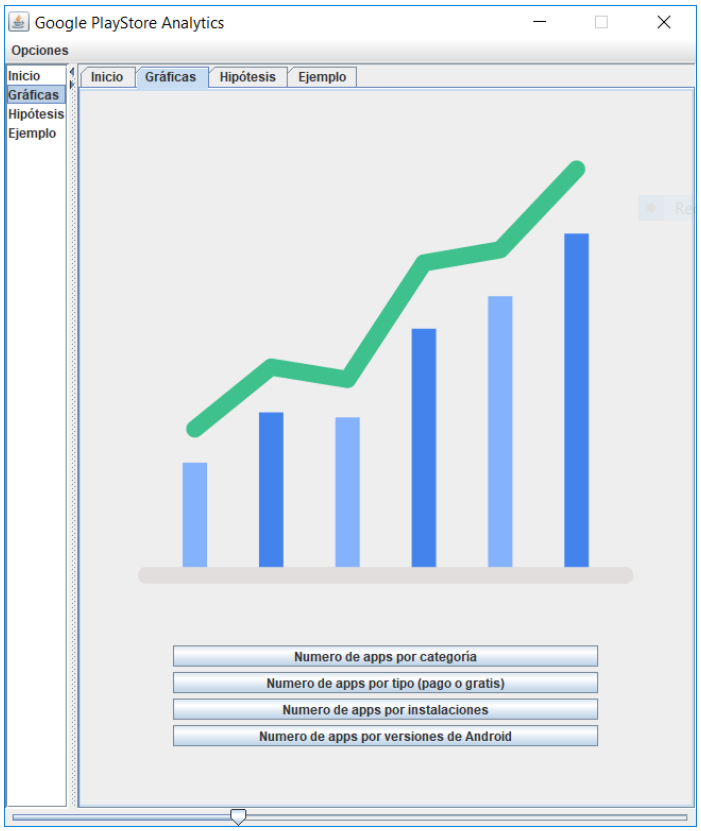
1. Inicio: en ella se visualizan los componentes del grupo.

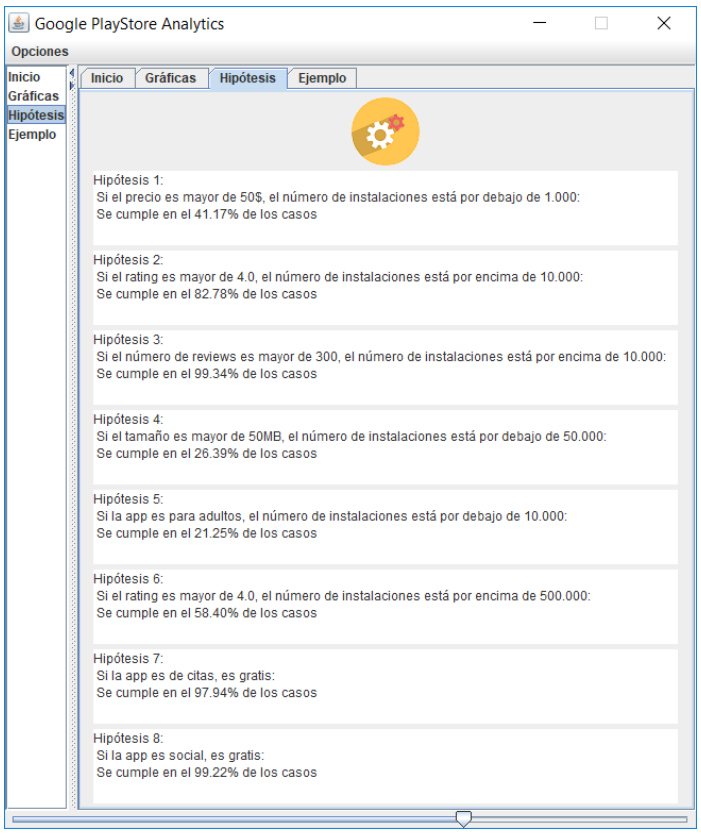
2. Gráficas: en esta pestaña se encuentran varios botones para visualizar las gráficas resultantes. Estas gráficas se muestran como ventanas emergentes.

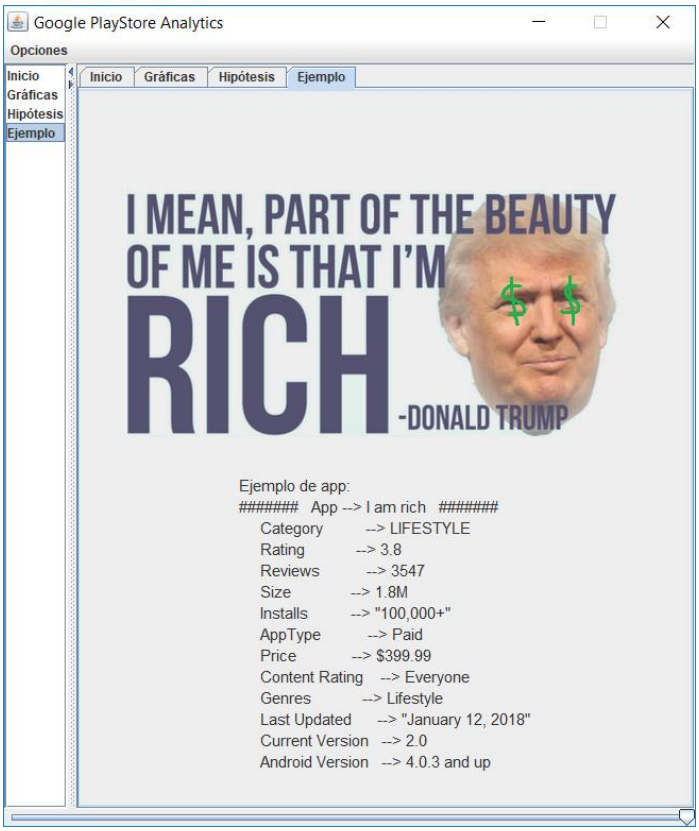
3. Hipótesis: se muestran las distintas hipótesis planteadas, así como el grado de confianza en las mismas.

4. Ejemplo: Se muestra un ejemplo de aplicación extraída del dataset, con el fin de ilustrar cómo están construidos los objetos App.




Google PlayStore Analytics
Opciones
Inicio
Gráficas
Hipótesis
Ejemplo
Inicio
Gráficas
Hipótesis
Ejemplo
Numero de apps por categoria
Numero de apps por tipo (pago o gratis)
Numero de apps por instalaciones
Numero de apps por versiones de Android


Google PlayStore Analytics
Opciones
Inicio
Gráficas
Hipótesis
Ejemplo
Inicio
Gráficas
Hipótesis
Ejemplo
Hipótesis 1:
Si el precio es mayor de 50$, el número de instalaciones está por debajo de 1.000:
Se cumple en el 41.17% de los casos
Hipótesis 2:
Si el rating es mayor de 4.0, el número de instalaciones está por encima de 10.000:
Se cumple en el 82.78% de los casos
Hipótesis 3:
Si el número de reviews es mayor de 300, el número de instalaciones está por encima de 10.000:
Se cumple en el 99.34% de los casos
Hipótesis 4:
Si el tamaño es mayor de 50MB, el número de instalaciones está por debajo de 50.000:
Se cumple en el 26.39% de los casos
Hipótesis 5:
Si la app es para adultos, el número de instalaciones está por debajo de 10.000:
Se cumple en el 21.25% de los casos
Hipótesis 6:
Si el rating es mayor de 4.0, el número de instalaciones está por encima de 500.000:
Se cumple en el 58.40% de los casos
Hipótesis 7:
Si la app es de citas, es gratis:
Se cumple en el 97.94% de los casos
Hipótesis 8:
Si la app es social, es gratis:
Se cumple en el 99.22% de los casos



6. Formulación de hipótesis

Dado que el planteamiento de este trabajo es cómo puede actuar una empresa que esté pensando en sacar una aplicación móvil al mercado, en tanto en cuanto a las decisiones que tomen entorno a la aplicación que quieren sacar, nos interesará detectar tendencias y patrones entre los atributos de los objetos.

Consideraremos los objetos como las aplicaciones móviles de la presente base de datos, esto son las entradas del .csv, y los atributos como las columnas del .csv, como pueden ser el tamaño, precio, categoría, etc.

Algo muy común en cualquier empresa es plantearse como influye el precio en función de las ventas.

Desde un punto de vista matemático, para razonar como de probable es que dicho patrón sea cierto en nuestro conjunto de datos, habría que contar cuántos de los objetos, en nuestro caso aplicaciones, cumplen que tienen un determinado precio "x", y de ese conjunto de aplicaciones filtrado contar cuantas aplicaciones cumplen con un determinado número de ventas "y".

Esto plantea la siguiente expresión, a la que denominaremos confianza de la implicación:

$$\text{confianza} = \frac{\sum_{i=0}^{|app|-1}(app[i].price == x \;\&\&\; app[i].sales == y)?\,1:0}{\sum_{i=0}^{|app|-1}(app[i].price == x)?\,1:0}$$

Visto de una manera menos formal: x => y equivaldría a calcular todos aquellos objetos que cumplen el atributo "x" y también cumplen el atributo "y", y dicha suma dividirla entre los objetos que cumplen el atributo "x".

La expresión anterior puede generalizarse a cualquier conjunto de atributos, no tiene por qué limitarse a un sólo atributo en cada lado de la implicación:

$$X \rightarrow Y$$

$$X \equiv z1, z2, \dots, zj$$

$$Y \equiv zj + 1, \dots, zk$$

$$i \in \{1, .., j, j + 1, .., k\}$$

En el numerador de la expresión de confianza, tendríamos el número de aplicaciones que cumple que el atributo 1 tiene el valor "z1" y el atributo 2 tiene el valor "z2", y así sucesivamente hasta el atributo "k", que tenga el valor "zk", dividido entre el número de aplicaciones que cumple que el atributo 1 tiene el valor "z1" y el atributo 2 tiene el valor "z2", y así sucesivamente hasta el atributo j, que tenga el valor "zj".

Veamos algunos ejemplos:

¿Si el precio de la aplicación es mayor que 50 dólares, entonces el número de instalaciones es menor que 1000?

Esto supone para una empresa determinar si el hecho de que una aplicación salga al mercado con un precio elevado, es indicativo de fracaso comercial.

Siguiendo la expresión de confianza anteriormente descrita, esta regla se cumple un 41,17% de las veces, lo cual es un porcentaje relativamente bajo y, por tanto, no podemos dar mucha validez a la implicación.

¿Si la valoración media de una aplicación está por encima de 4/5, entonces el número de instalaciones es mayor que 10.000?

Esto supone para una empresa determinar si el hecho de que una aplicación consiga una valoración media alta, se traducirá en muchas descargas de la misma.

Esta regla se cumple el 82,78% de las veces, lo cual es un indicativo de que existe un patrón evidente entre la calificación media y el número de descargas.

¿Si el número de reviews es mayor que 300, entonces el número de instalaciones es mayor que 10000?

Esta implicación viene a plantear si el hecho de que una aplicación tenga un número alto de valoraciones, implica un gran número de descargas. También puede verse como el hecho de que los consumidores vean atraída su curiosidad ante aquellos productos con muchas valoraciones.

Se trata de una apreciación interesante desde un punto de vista empresarial, puesto que, si se diese dicho caso, la empresa en cuestión que quisiese lanzar la aplicación tan sólo tendría que buscar la manera de obtener un número de valoraciones relativamente alto, lo cual es sumamente fácil y con un coste relativamente bajo

Esta regla se cumple el 99,34% de los casos, esto nos dice que hay una clara tendencia hacia ello, y es evidente que se da como una forma más de marketing online.

¿Si la valoración media de una aplicación está por encima de 4/5, entonces el número de instalaciones es mayor que 500.000?

Esta regla se cumple el 58,40% de los casos, muy cerca de la frontera de [0, 1]. Por lo general, aquellos casos que se encuentran cerca de la frontera son difíciles de evaluar en cuanto a la certeza o no de la regla. Los porcentajes, cuanto más extremos mejor.

¿Si el tamaño de una aplicación está por encima de los 50 megabytes, entonces el número de instalaciones es menor que 50.000?

Esto supone para una empresa determinar si el hecho de que una aplicación ocupe mucho espacio en un móvil, hará que los consumidores no respondan bien a las expectativas comerciales de la empresa.

Esta regla se cumple el 26,39% de las veces, no parece haber un patrón entre ambas afirmaciones y, en principio, el peso de una aplicación no tendría por qué asustar al comprador, haciendo que éste deje de comprarla.

¿Si la aplicación está destinada a un público adulto, entonces el número de ventas serán menor de 10.000 unidades?

Esto supone para una empresa determinar si cerrarse a un público adulto hará que la aplicación venda menos, y si por tanto es necesario hacerla para todos los públicos a fin de conseguir mejores ventas.

Esta regla se cumple un 21,25% de las veces, no parece haber un patrón entre ambas afirmaciones. Por lógica podríamos suponer que, si la aplicación es de calidad, véase un videojuego violento, pero con buena trama y mecánicas jugables, no debería hacer que tuviese malas ventas.

¿Si la aplicación tiene categoría "Dating", entonces la aplicación es gratuita?

Esto supone para una empresa que desea lanzar una aplicación de citas, si cobrar o no por ella.

Esta regla se cumple 97,94% de los casos, hay un claro patrón en dicha afirmación, y no sería buena idea cobrar por la aplicación.

Lo que se suele hacer en estos casos es enmascarar los pagos en concepto de membresías "Premium" que dan ciertos privilegios, pero lo que es la aplicación en sí suele ser gratuita.

¿Si la aplicación tiene categoría "Social", entonces la aplicación es gratuita?

Esta regla se cumple el 99,22% de los casos, igual que el anterior.

7. Conclusiones

En el presente proyecto se ha abordado la implementación de una herramienta de representación de datos y análisis de hipótesis planteadas para un conjunto de datos concreto sobre aplicaciones de google play store.

Un hecho que no se ha abordado a la hora de extraer conclusiones sobre ciertas hipótesis planteadas y que nos gustaría mejorar en una futura versión del trabajo, es el hecho de valorar como de representativo son los resultados que se obtienen de dichas hipótesis dentro del conjunto total de datos.

Supongamos lo siguiente:

$$X \rightarrow Y$$

Si decimos que dos objetos cumplen el concepto de la izquierda de la implicación, y de esos 2 objetos ambos cumplen el concepto de la derecha de la implicación, entonces diríamos que la probabilidad de certeza de la implicación es del 100%.

Sin embargo, si el conjunto total de objetos es de 50.000, entonces 2 objetos de 50.000 que existen en el conjunto de datos no es representativo, y por tanto la implicación es poco certera, por no decir nula.

Esto nos lleva a concluir que el hecho de que una implicación esté cerca del 100% de certeza se debe, o bien a que es algo evidente y que no requiere de un análisis de datos, o bien que es poco representativa.

Este problema es fácilmente abordable mediante un filtro por soporte, donde el soporte de la regla vendría definido por la siguiente expresión:

$$soporte = \frac{|(X \cup Y)'|}{|apps|}$$

$donde\ |(X \cup Y)'|\ son\ el\ número\ de\ aplicaciones\ que\ cumplen\ los\ atributos\ X\ e\ Y$

Otra cuestión a abordar sería la implementación de un algoritmo de extracción de la base STEM, esto son todas aquellas implicaciones entre atributos con un porcentaje de probabilidad del 100%, dicho de otra manera, quedarse sólo con aquellas implicaciones con un cien por cien de probabilidad.

Por último, mediante el uso de la biblioteca “MLib” introducida en la asignatura “Inteligencia Empresarial”, la implementación de un clasificador de regresión logística que busca un conjunto de pesos a modo de hiperplano bajo la combinación lineal:

$$\sum_i XiWi$$

donde el concepto de clase vendría definido por el número de ventas de la aplicación, esto es, dado el conjunto de entrenamiento formado por todos los ejemplos del .csv con el que se ha trabajado y la clase de cada ejemplo el número de ventas, hallar un hiperplano que, al introducir una nueva instancia, esto es una nueva aplicación que queramos sacar al mercado, nos la clasifique en función del número de ventas que el modelo prevé que va a obtener.

Podría implementarse como un clasificador de regresión logística binaria, que tendría 2 clases, la positiva y la negativa. La negativa equivaldría a tener menos de un determinado número de ventas, y la positiva equivaldría a tener igual o más de un determinado número de ventas. Como número de ventas podría usarse la media o la moda.

Otro planteamiento podría ser el de un clasificador de regresión logística multiclase con paradigma one vs rest, donde se tendrían "k" clasificadores, uno por cada clase y donde discretizariamos en varias clases el número de ventas, Cada clasificador enfrentaría a la clase i-esima frente al resto. Se devolvería la clase asociada al clasificador que obtenga mayor probabilidad.

Durante el transcurso de este proyecto hemos obtenido experiencia en el manejo de datasets, así como en el uso de la programación funcional en Scala para hacer esta tarea mucho más rápida y sencilla. Además, se han investigado librerías para la representación de datos de forma efectiva, entre las que seleccionamos Vegas-Viz para generar los gráficos, y Swing para el desarrollo de la interfaz de usuario.

Otro punto a destacar es la utilización de Spark, que permite el procesamiento paralelo de códig con el fin de mejorar el rendimiento de los análisis. Spark utiliza conjuntos de datos específicos, que son tratados de forma muy distinta a los demás conjuntos.

Se han encontrado dificultades a la hora de tratar estos conjuntos de datos (RDD) de Spark, además de otros específicos del dataset, que han sido comentados durante el proyecto.

En definitiva, este proyecto ha sido de gran utilidad para explorar nuevas tecnologías que son utilizadas por gran cantidad de desarrolladores, y que durante el transcurso de la carrera no son mencionadas. Además, ha permitido hacer uso de tecnologías alternativas, y poder investigar acerca de ellas.