Apuestas Seguras

Rec. de la Información y Rec. en la Web (2016)

Profesora:
Libertad Tansini

Estudiantes:
Santiago Bartesaghi
Carlos Huerta
Guillermo Rodriguez

Índice

Índice **2**

Introducción **3**

Problema **3**

Enfoque de la solución **3**

Diseño **4**

Implementación **5**

Módulos 5

Betfair 5

Interwetten - Bwin 5

Flujo de Aplicación 6

Marca Apuestas 6

Servidor Web 6

Funcionalidades y uso **7**

Evaluación y resultados **7**

Conclusiones **7**

Trabajo futuro **8**

Introducción

Las casas de apuestas son empresas que se dedican a ofrecer una cierta paga a los apostadores que acierten a determinado resultado de un evento. Por ejemplo se podría apostar que el Barcelona le va a ganar al Real Madrid y la casa de apuesta va a definir cuánto me va a pagar dependiendo en el monto apostado y la probabilidad de que el evento suceda.

Esta probabilidad es información estadística y posiblemente existan otros factores que afectan a la misma. A su vez es información propietaria de cada casa de apuesta por lo que cada una maneja sus propias probabilidades y pagas para cada evento/resultado.

Problema

En este caso nos enfocaremos únicamente en las casas de apuestas en la sección fútbol. El hecho de que cada casa de apuestas maneje su propios datos y pagas nos da el indicio que si varias casas de apuestas tuvieran probabilidades muy diferentes entre sí se podría encontrar una combinación de apuestas entre todas las casas de apuestas en las que podemos asegurarnos una ganancia apostando cierta cantidad de dinero a todos los posibles resultados del evento.

En BetBurguer[1] explican cómo saber si podemos asegurarnos tener ganancia apostando a un mismo evento en diferentes casas de apuestas. El problema se reduce a recuperar información que se encuentra dispersa en internet, procesar esta información e indicarle al usuario de una manera sencilla qué acciones debe tomar para poder verse beneficiado económicamente al realizar sus apuestas.

Enfoque de la solución

El enfoque de la solución que se plantea es entonces recuperar información de las casas de apuestas respecto a los eventos, todos los resultados con su correspondiente paga e indicarle al usuario final cuál es la manera más inteligente de realizar la apuesta. Esto significa mostrar en cuáles casas de apuesta podrá obtener la mayor ganancia para cierto resultado de un evento y en caso de existir combinaciones de apuestas en las que se asegure una ganancia, indicarle de qué manera debe realizar la apuesta para que esto suceda.

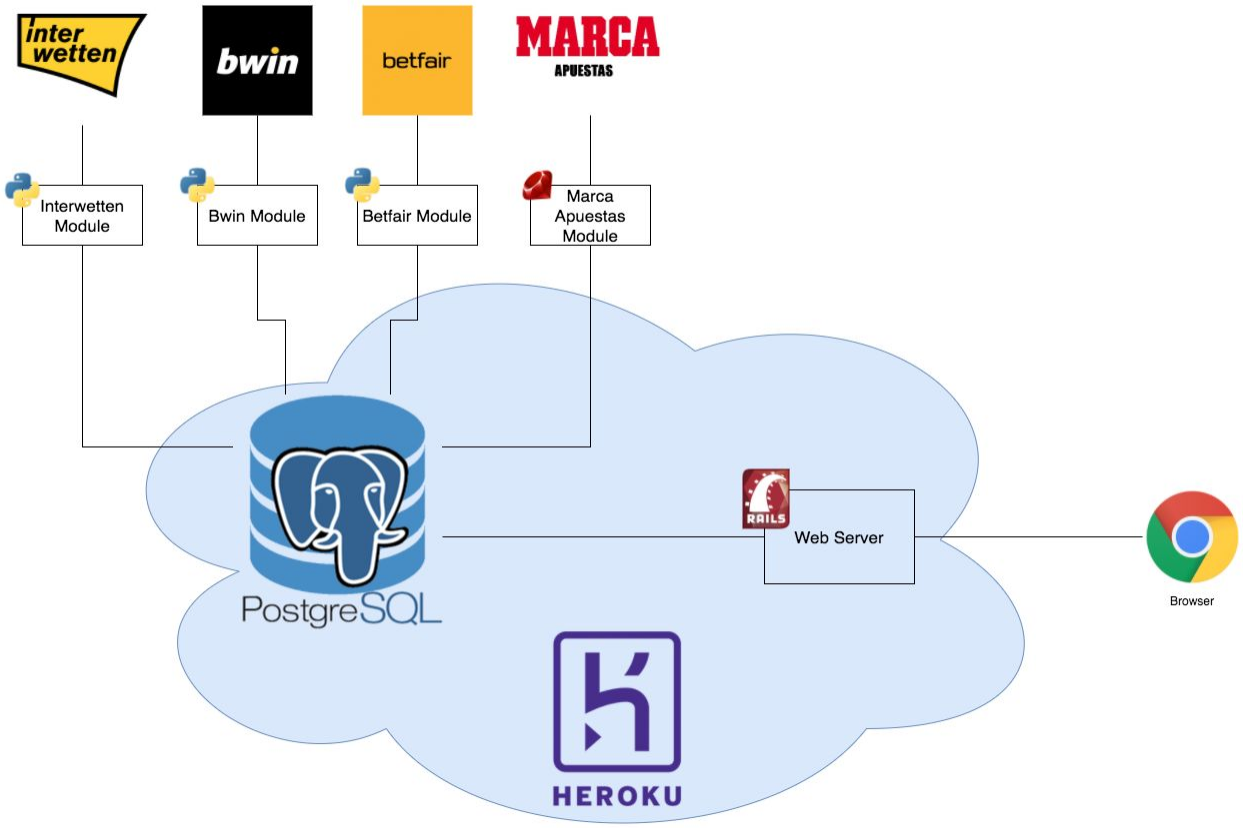
La mayor dificultad se encuentra en recuperar toda la información posible, ya que para poder encontrar apuestas seguras es necesario tener la mayor cantidad de casas de apuestas posibles, ya que muchas deben tener datos estadísticos parecidos y manejan probabilidades parecidas. En general las casas de apuestas coinciden en el resultado del evento, la diferencia es que difieren en la probabilidad de que cada evento pase y por lo tanto en la paga. Además

se agrega la dificultad de no disponer de muchas APIs, en general las casas de apuestas no proveen APIs y en caso de hacerlo, la mayoría son pagas.

Diseño

Se tienen varios componentes, posiblemente en distintos lenguajes de programación que se encargan de comunicarse con la API de una casa de apuestas o scrapear la página de la casa de apuestas. Estos componentes son los encargados de obtener la información, normalizarla utilizando un diccionario en formato JSON y almacenarla en una base de datos PostgreSQL hosteada en Heroku.

Por otro lado está servidor web en Ruby on Rails hosteado en Heroku el cual se encargará de obtener la información normalizada de la base de datos y procesará los datos para mostrarle al usuario final los eventos disponibles para apostar, en cuáles casas de apuesta podrá obtener la mayor ganancia y si existen combinaciones de apuestas en las que se asegura una ganancia.



Implementación

Módulos

Consisten 3 etapas:

1. Se scrapea o utiliza alguna API (en caso de existir) para obtener de una casa de apuestas, todas las apuestas disponibles de todos los partidos de todas las ligas de fútbol.
2. Se normalizan los nombres de los equipos para que coincidan entre los distintos nombres que tienen en las distintas casas de apuestas. Esto se hace utilizando un diccionario en formato JSON y buscando al equipo por clave, el valor es el nombre normalizado del equipo.
3. Se guarda toda la información de los equipos, partidos, y el partido con las pagas en esa casa de apuestas en la base de datos hosteada en Heroku.

Betfair

Para recabar los datos de betfair se utilizó una API que provee betfair [2]. Esta utiliza JSON-RPC como medio para enviar y recibir los datos.
El cliente se hizo en python y se utiliza psycopg2 [3] para comunicarse con la base de datos

Para realizar el login se necesito crear un usuario en el sitio, asociar una app key a dicho usuario y crear un certificado, estos datos son enviados una única vez cuando el cliente se conecta a la API.

Primero se consulta por todas las ligas existentes de fútbol y para cada liga se traen los ids de los mercados correspondientes a todos los partidos de dicha liga. Luego se consulta para cada partido de cada liga por el tipo de apuesta “match odds” (equipo 1, empate equipo2). Por último se guardan los datos en la base remota.

Si bien en un principio se pensó que disponer de la API iba a ser una ventaja, al final no resulto muy practico ya que las llamadas a la misma demoran bastante y a veces devuelve información errónea.

Interwetten - Bwin

La implementación para ambas casas de apuestas fue realizada en el lenguaje Python utilizando los módulos Scrapy (permite obtener la información de la web) y Psycopg2 (utilizado

para el almacenamiento en la base de datos). Ambas son consistentes y siguen la misma lógica.

Para cada casa de apuestas se creó un proyecto Scrapy que consiste en los siguientes componentes:

- Items.py: Clase Python que especifica un partido de la casa de apuesta.
- Spider.py: Encargado de Obtener la informacion de la web.
- Pipeline.py: Normaliza los datos obtenidos chequeando inconsistencia y almacena en la base de datos.

Flujo de Aplicación

Mediante el componente “Spider” se realiza el “scrapeo” de la web de la casa de apuesta solicitada, obteniendo para cada partido su fecha, cuadro local, cuadro visitante y las cuotas asociadas. En caso que la página web mantenga referencias a otras páginas mediante links (<a>), se realiza el scrapeo de las mismas. Luego de obtenida la información el módulo “Spider” es creada una instancia de la clase Items para cada partido hallado. Cuando una instancia de “Items” es creada el control de la misma se le pasa al módulo “Pipeline”. Este módulo realiza las validaciones pertinentes, normaliza los nombres de cuadros a una forma estándar y almacena la información en la base de datos.

Marca Apuestas

El script está hecho en Ruby. En este caso se scrapea la página de Marca Apuestas primero para obtener los links de todas las ligas de fútbol y luego se scrapea cada liga. Para scrapear se utiliza la gema Wombat [4] y se buscan los textos de tags HTML que tengan cierta clase CSS. Luego se procesan y normalizan los datos utilizando la librería estándar JSON para cargar el diccionario de nombres de equipos. Para guardar los datos en la BD se utiliza el ORM ActiveRecord [5].

Servidor Web

Es una aplicación hecha en Ruby on Rails hosteada en Heroku y que utiliza una base de datos PostgreSQL. Heroku es una plataforma como servicio (PaaS) que evita el costo de toda la configuración de infraestructura a las empresas/desarrolladores[6]. En la ruta base se encuentra el listado de todos los partidos con las mejores apuestas encontradas en las casas de apuestas que el sistema dispone. En la ruta /update_matches se vuelven a calcular las mejores apuestas para toda la información que se dispone en ese momento.

Funcionalidades y uso

El usuario final podrá encontrar en la página webir.herokuapp.com un listado de partidos con la información de la fecha, los equipos, y la paga de la casa de apuestas que ofrece mayor paga (mejor apuesta). Además se expone la columna Bet Security que muestra cuán buena es la combinación de apuestas, cuanto menor es el número, mejor. Y si el número es menor que 1 podemos encontrar una combinación de apuestas en la que nos aseguramos ganar. La lista está ordenada por el Bet Security de los partidos.

Además el usuario podrá cliquear el partido en el que desee apostar y se abrirá un modal. En ese modal puede ingresar cuánto dinero quiere apostar y el sistema le indicará la cantidad que debe apostar a cada cuadro en cuál casa de apuestas para garantizar la mayor ganancia segura posible.

Solo se muestran partidos que aún no hayan ocurrido porque los demás ya no tienen sentido. Como se manejan mucha cantidad de partidos se realiza una paginación de 100 partidos por página. El usuario puede navegar entre las páginas.

Por último se dispone de una búsqueda para poder encontrar equipos a los que se les quiere apostar. La búsqueda funciona de manera que se retornan los partidos en los que al menos 1 de los equipos contiene la palabra buscada en su nombre.

Evaluación y resultados

Como evaluación vemos que el usuario final puede verse beneficiado sin realizar esfuerzo alguno de encontrar apuestas seguras, y a pesar de que necesita un capital para invertir, es buen negocio si se dispone de él.

Por otro lado la API de Betfair no es del todo confiable ya que validando las pagas contra las de la página mismo vemos que no siempre coinciden. Garantizar que los datos mostrados son correctos es clave para la seguridad del usuario a la hora de apostar y evitar generarle pérdidas.

Conclusiones

Se considera que el sistema puede ser muy valioso si se siguen agregando casas de apuestas y se valida que la información mostrada es correcta. La ventaja del diseño es que permite agregar casas de apuestas sin tocar código ya existente, es simplemente agregar un módulo que obtenga la información de la casa de apuesta deseada, la normalice, y la guarde en la base de datos. Luego cliqueando refresh en la página se recalculan las mejores apuestas.

Por otro lado podrían separarse aún más las partes de scraping, normalización y guardado en la BD para mayor reutilización.

Trabajo futuro

Dado que es solo un prototipo la carga de datos se hace manual ejecutando programas/scripts pero esto podría automatizarse utilizando Cron[7] para ejecutar cada un día por ejemplo los programas/scripts que actualicen los datos (en caso de que estos cambien) y agreguen nuevos partidos a la base de datos.

La normalización hoy en día se hace con un diccionario, pero esto implica que cuando se agrega una nueva casa de apuestas hay que ir a mano al diccionario y agregar los equipos que vienen con un nombre diferente al que ya está en la base de datos. Ejemplo es que me puede venir Barcelona FC y en la base de datos el equipo es Barcelona. Otro trabajo futuro sería que cuando se busca si el equipo ya existe en la base de datos se utilice algún método de matcheo y si el nombre de equipo es por ejemplo un 90% parecido, se los considera que son el mismo.

Referencias

[1] Betburguer - How is it working https://www.betburger.com/help/how_is_it_working

[2] Betfair API Documentation
http://docs.developer.betfair.com/docs/display/1smk3cen4v3lu3yomq5qye0ni/Getting+Started

[3] Psycopg – PostgreSQL database adapter for Python http://initd.org/psycopg/docs/

[4] Github - Wombat https://github.com/felipecsl/wombat

[5] Github - ActiveRecord https://github.com/rails/rails/tree/master/activerecord

[6] Heroku - What is Heroku https://www.heroku.com/what

[7] Unix - Cron http://www.unixgeeks.org/security/newbie/unix/cron-1.html