Trabajo Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Desarrollo de un software para la búsqueda de apuestas seguras

Autor: José Antonio Rama Rico

Tutor: Jesús Iván Maza Alcañiz

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Desarrollo de software para la búsqueda de apuestas seguras

Autor:
José Antonio Rama Rico

Tutor:
Jesús Iván Maza Alcañiz

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado: Desarrollo de software para la búsqueda de apuestas seguras

Autor: José Antonio Rama Rico
Tutor: Jesús Iván Maza Alcañiz

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Agradecer en primer lugar a mi tutor Iván Maza, así como a mis compañerosde 4º, por la paciencia y la oportunidad de realizar este proyecto.

A toda la gente que me llevo de estos años, ya sea de la carrera, de la residencia o de bonitas casualidades. Por todos los momentos compartidos, por su ayuda, por las horas de estudio y las de no tanto estudio, por creer en mí y ayudarme a crecer. A Pablo y a Esteban, por estar conmigo durante todo el proceso de realización de este trabajo.

En especial agradecer a mi familia, por su apoyo incondicional sin el que no hubiera podido llegar donde estoy.

José Antonio Rama Rico
Sevilla 2022

Resumen

En el presente Trabajo Fin de Grado se realiza un estudio de los diferentes lenguajes de programación y librerías que se pueden utilizar para hacer scraping web y mi elección con el fin de realizar un programa informático que a través de scraping web recoja los datos de las cuotas de los partidos para buscar apuestas seguras en las mejores competiciones deportivas y las muestre por pantalla a partir de una interfaz gráfica.

Abstract

In this Final Degree Project, a study of the different programming languages and libraries that can be used to do web scraping and my choice is carried out in order to make a computer program that collects the data of the quotas through web scraping. of the matches to search for safe bets on the best sports competitions and display them on the screen from a graphical interface.

.

Índice

1 Introducción

1.1 Motivación y Contexto

Todo comenzó en 2º de carrera, estaba apostando a un partido de Champions League con Jesús Martinez, amigo de la infancia, y observamos la diversidad de cuotas que existía entre las diferentes casas de apuestas para el mismo partido y misma apuesta.

Esto nos llevó a pensar que podría existir alguna manera de utilizar esta diversidad para la disminución del riesgo que conllevaba cualquier apuesta o para maximizar el beneficio, ya que, podríamos elegir la cuota que más nos beneficiara.

Semanas más tarde, pensando sobre el tema, nos dimos cuenta que las cuotas de cada apuesta relacionada también estaban relacionadas entre sí, lo cual era obvio porque las cuotas están relacionadas con el tanto por ciento que la casa de apuesta estima que tiene la apuesta de convertirse en correcta.

En este momento se nos ocurrió investigar sobre la posibilidad de apostar en diferentes casas de apuestas cuotas relacionadas con la intención de no perder dinero o ganarlo independientemente del resultado del partido, por ejemplo, en un partido de futbol apostar en diferentes casas de apuestas todos los posibles resultados (1, x, 2) e independientemente del resultado tener un beneficio.

Al buscar esto en internet nos dimos cuenta que ya existía dicha investigación y que teóricamente funcionaba, se denominaban ‘surebet’.

A partir de esto se me ocurrió realizar un trabajo donde encuentre dichas apuestas seguras con la intención de investigar si realmente se puede sacar beneficio de ello.

1.2 Objetivos

El objetivo general de este Trabajo Técnico es encontrar apuestas seguras dentro de diferentes casas de apuestas, el análisis de dichas apuestas y su exposición en una interfaz gráfica donde se pueda observar todos los datos necesarios para aprovechar dicha información.

El objetivo general se podría dividir en los siguientes subobjetivos:

- Obtención de cuotas de diferentes casas de apuestas.
- Análisis de las cuotas para la obtención de apuestas seguras.
- Análisis de las apuestas seguras para la obtención de datos fundamentales para la ejecución de dicha apuesta y datos sobre su rendimiento.
- Exposición de datos actualizados en interfaz gráfica.

1.3 Estructura del documento

En este apartado paso a explicar la estructura que va a tener este trabajo a partir de este punto, una vez que hemos realizado la introducción del mismo.

A continuación, vamos podremos visualizar un apartado donde trataremos de explicar lo que es una cuota y como se calcularía, todo ello para poder entender lo que es una apuesta segura, como poder encontrarla y calcular sus características.

Cuando ya entendamos el concepto de apuesta segura pasaremos al siguiente apartado donde podremos ver las diferentes posibilidades que tenemos para realizar un programa que encuentre apuestas seguras. Empezando por los lenguajes de programación que podremos utilizar y terminando por las librerías más habituales a la hora de hacer scraping e interfaces gráficas dentro de Python.

Nuestro siguiente apartado trata de explicar paso a paso el programa informatico desarrollado. Lo explicaremos función por función en el orden en el cual se ejecuta el programa.

El siguiente apartado será ‘Modo de funcionamiento’ donde se expone como actúa el programa desde el punto de vista del usuario y por último, el apartado ‘Conclusiones y desarrollo futuro’ donde explicamos las conclusiones del programa y las posibles actualizaciones o mejores que podría tener el programa para un obtener un resultado más amplio o más específico.

2 Concepto de Apuestas Seguras

2.1 Introducción a las Cuotas

La cuota es el beneficio que recibes por cada euro apostado, es decir, la ganancia neta más la cantidad apostada.

Por ejemplo: Imagínate que has apostado 10€ por un equipo con una cuota de 1,3 y ese equipo gana. La cantidad de dinero que cobrarás de la casa de apuestas será 10€ x 1,3 = 13€, mientras tu beneficio será de 13€ – 10€ = 3€

Las casas de apuestas establecen sus cuotas en base a:

– El cálculo estadístico de probabilidades: dentro de este se tienen en cuenta distintas variables, como por ejemplo la trayectoria de los equipos, los jugadores lesionados, si el equipo juega con suplentes…

– El margen de beneficio que obtienen de las apuestas.

Con estos datos, calculan las cuotas finales que van a establecer. Las cuotas finales nunca podrán ser ni muy altas, ya que entonces perderían dinero, ni muy bajas, ya que si no casi nadie querría apostar.

Si quieres saber cómo se calculan las probabilidades de las cuotas gracias al método de las cantidades implícitas, echa un vistazo a nuestro ejemplo.

El método de las cantidades implícitas

Para hacerlo más fácil vamos a utilizar los mismos datos que en el ejemplo de las cantidades decimales. Cuota A =1,3; Cuota B= 2,8; Cuota AB =2,9

Calculamos la probabilidad simple = 1/cuota x 100

PA= 1/1,3 x 100 = 76,92%

PB= 1/2,8 x 100 = 35,71%

PAB= 1/2,9 x 100 = 34,48%

PA+PB+PAB = 76,92% + 35,71% + 34,48% = 147,11% >100%

El beneficio de las casas de apuestas se calcula como: (1/1,3) + (1/2,8) + (1/2,9) = 1,4711

Este resultado quiere decir que por cada 1,47€ que se hayan apostado, la casa de apuestas espera pagar 1€.

Los porcentajes de pagos de las casas de apuestas se calculan de la siguiente manera: 1/beneficio de las casas de apuestas x 100 = 1/14711 x 100 = 67,98%

Por cada 100€ apostados en el partido entre el equipo A y B, la casa de apuestas espera devolver 67,98€.

Debes tener en cuenta que las cantidades que estamos utilizando en el ejemplo son

completamente ficticias. Lo normal es que el porcentaje de pagos sea superior al 90%.

Ahora ya disponemos de datos suficientes para calcular la probabilidad implicita = probabilidad simple x porcentaje de pagos

PImplícitaA = 0,7692 x 0,6798 = 0,5229

PImplícitaB = 0,3571 x 0,6798 = 0,2427

PImplícitaAB = 0,3448 x 0,6798 = 0,2344

Al contrario que con las probabilidades simples, el resultado de sumar las probabilidades implícitas es del 100%, por ello, podemos tener en cuenta los datos obtenidos a la hora de realizar nuestras apuestas.

Y por último veremos porque fluctúan las cuotas, ya que, las casas de apuestas no deciden aumentar o disminuir las cuotas a su antojo, todo depende de las apuestas de los usuarios. Cuanta más gente apueste por un equipo, más bajas serán las cuotas para ese equipo y al revés.

Las cuotas indican la probabilidad de que el resultado deseado suceda, y, por tanto, pueden sufrir variaciones muy importantes. Dicho esto, lógicamente, cuanto más tiempo quede para se produzca el evento, mayores serán las cuotas.

Las casas de apuestas pueden modificar las cuotas en cualquier momento, pero no te preocupes, si ya habías realizado tu apuesta, ésta se mantendrá con la cuota del momento en cuestión.

Si tuviéramos que explicar cómo varían las cuotas de una manera resumida, diríamos que “a mayor riesgo, mayor ganancia”. Es por este motivo que cuanto más alta es la cuota, menos opciones tiene el equipo de ganar.

2.2 Surebet

Las apuestas seguras son parte del llamado sistema de arbitraje, un concepto cuyo origen está en los mercados financieros. Este sistema permite beneficiarse del desequilibrio entre dos o más mercados, de hecho, quizá hayas oído alguna vez el término arbitraje en los mercados en los que se realizan operaciones de intercambio de divisas, en ellos se pueden obtener beneficios por las diferencias en los tipos de cambio de las monedas.

Dicho lo cual, nos podríamos preguntar, ¿qué relación puede tener esto con las apuestas? La respuesta es... encontrando cuotas que te den margen de beneficio en diferentes resultados de un evento deportivo, puedes garantizarte una ganancia.

El sistema de arbitraje en las apuestas deportivas es más fácil de explicar que su equivalente en los mercados financieros. Aquí vamos a explicar los pasos que te acercarán a poder hacer una Surebet, o apuesta segura.

A través del arbitraje podrás conseguir una apuesta segura, ya que el sistema permite sacar provecho de la diferencia en las cuotas entre distintas casas de apuestas. Lo que predica la teoría es que podrás obtener un beneficio independientemente del resultado del partido, por el hecho de que las cuotas dejan un cierto margen que puedes aprovechar a tu favor.

La llegada de internet facilitó la posibilidad de hacer apuestas seguras, algo que solía estar reservado a apostantes profesionales.

Para conseguirlo, la estrategia a seguir es apostar a resultados contrapuestos en el partido cuando encuentres cuotas que te permitan ganar de forma segura una cantidad mayor que la que has invertido. Este es un método que solía estar reservado para los apostantes más expertos, pero la llegada de internet ha hecho que sea más sencillo y mucho menos llamativo hacer arbitraje en las apuestas deportivas.

Se debe saber que en la actualidad muchas casas de apuestas online reproducen las cuotas que se utilizan en los mercados asiáticos, a resultas de que en casi todas las páginas webs encontrarás cuotas parecidas. De ahí que se necesite invertir mucho tiempo y tener un buen conocimiento matemático para llevar esto a buen término. A día de hoy hay además algunas webs y softwares disponibles en la red que te pueden orientar en la dirección correcta.

Como decíamos, las apuestas seguras o apuestas de arbitraje se pueden conseguir cuando encuentras cuotas diferentes en algún mercado de apuestas, de manera que surge la oportunidad de apostar garantizándote un beneficio sin importar cómo termine el evento deportivo de que se trate. Esto es más fácil de decir que de hacer, ya que hay aspectos a tener en cuenta antes de ir detrás de una apuesta segura o de arbitraje.

Lo primero y más importante que debes saber es que muchas páginas de apuestas han prohibido expresamente las apuestas seguras, y actualmente ninguna casa de apuestas europea las permitirá. Aquí te mostramos un ejemplo de cómo una empresa puede expresar la prohibición en sus términos y condiciones: "Declaramos que una apuesta será parcial o totalmente invalidada si creemos que se ha producido arbitraje, y/o procederemos a cerrar la cuenta del usuario"

Empezaremos por decir que todas las casas de apuestas ganan dinero por el hecho de quedarse un margen de las cuotas que ofrecen, lo cual se entiende mejor poniendo un ejemplo: si el corredor de apuestas cree que hay un 50% de probabilidades de que se produzca un resultado en un evento deportivo, sus cuotas deberían ser 2.0, pero él las fija en 1.9 para obtener un margen de beneficio, esto no debería sorprender a nadie, ya que todos sabemos que las cuotas son fijadas en favor de la casa de apuestas (explicado en el apartado anterior). Prosiguiendo con nuestra explicación...

La página web de apuestas intentará equilibrar las inversiones que se han realizado en ambos lados de una apuesta (las apuestas a favor y en contra) y lo hará ajustando las cuotas, de manera que puedan ganar dinero sin importar el resultado del partido. Con las apuestas seguras, la lógica que se sigue es similar: se trata de apostar en dos páginas de apuestas diferentes que ofrezcan cuotas distintas en un mismo partido, aprovechándote de la diferencia en las cuotas para obtener un beneficio garantizado.

A continuación, vamos a explicarlo usando un partido en el que se puedan producir como mínimo dos resultados distintos, para lo cual usaremos el tenis: tenemos Jugador A y Jugador B, así como página de apuestas 1 y página de apuestas 2

	Jugador A	Jugador B
Página web 1	1.9	2.0
Página web 2	2.2	1.7

Esta sería una oportunidad excelente para llegar a una Apuesta Segura. Si juegas 100€ al jugador B a cuota 2.0, y 90€ al jugador A a cuota 2.2... tendrás un beneficio garantizado de 8€. Vamos a desglosarlo:

Apuestas 100€ al jugador B en la página web 1

Apuestas 90€ al jugador A en la página web 2

RESULTADO 1: Gana el jugador A

Pierdes los 100€ en la página de apuestas 1

Ganas tu apuesta de 90€ en la página 2, a cuota 2.2 = 198€

Por tanto, has perdido 100€ por un lado, pero por el otro has recuperado los 90€ que apostaste y además has ganado 108€; por por lo que tu beneficio son 8€

RESULTADO 2: Gana el jugador B

Pierdes los 90€ apostados en la página de apuestas 2

Ganas tu apuesta de 100€, a cuota 2.0, en la página 1; por lo que ganas 100€ y recuperas los 100€ apostados

Por tanto, dado que apostaste 190€ en total y has recuperado 200€, has conseguido 10€ de beneficio.

Si gana un jugador obtienes 10€, y 8€ si gana el otro... obtendrás beneficios gane quien gane.

Esto es sin embargo sólo un ejemplo; de hecho, puede ser realmente difícil encontrar casas de apuestas que ofrezcan cuotas tan dispares en un partido. Además, estamos hablando sólo de ganancias de 8 ó 10€ con la apuesta segura, habiendo tenido que invertir en total 190€. Aunque este caso puede parecer fácilmente conseguible, la realidad es que se requiere una gran inversión y muchas apuestas seguras para ganar una cantidad de dinero sustancial.

Pero, además, encontrar las cuotas requiere una importante inversión de tiempo. Aunque por simplicidad sólo hemos usado páginas web de apuestas 1 y 2; hubiese sido más realista decir, por ejemplo, "página de apuestas 14" y "página de apuestas 29"; es decir, lo más probable es que necesites consultar las cuotas en numerosas casas de apuestas para encontrar una oportunidad como la que hemos explicado. Teniendo también en cuenta que en la actualidad las cuotas son muy parecidas en la mayoría de webs de apuestas, hacer arbitraje es más difícil que lo que solía serlo.

Te podemos decir que muchos corredores de apuestas que operan bajo una cierta marca son en realidad parte de un grupo mayor de empresas, que en ocasiones puede englobar varias casas de apuestas y de ser así, tendrá acceso a cuotas en distintos sitios de apuestas. Esto implica que será más improbable que encuentres cuotas a resultados contrapuestos (en un evento deportivo) que puedan ser aprovechados para obtener una apuesta segura. E incluso si lo consigues, puede que la casa de apuestas lo sepa.

2.2.1 Calculo de Surebet, Beneficio esperado y Porcentaje de inversión por cuota

Podemos calcular si un cumulo de cuotas es acto para ser una apuesta segura de la siguiente manera:

Dividiendo el número 1 entre las distintas cuotas del mercado y luego sumando los resultados. Si la cifra que obtenemos es inferior a 1, es que las cuotas permiten hacer una surebet.

Ejemplo: Para un partido de fútbol encontramos las siguientes cuotas para el mercado 1X2:

- 1 @ 2,20
- X @ 3,60
- 2 @ 5,00

Realizamos el cálculo:

1/2,2 + 1/3,6 + 1/5 = **0,93 < 1**

Obtenemos un resultado menor que 1 (0,93), esto significará que hemos encontrado una surebet con la cual tendremos un beneficio seguro siempre que sepamos cuanto debemos invertir en cada cuota.

En los siguientes apartados vamos a ver cómo saber el beneficio esperado y conocer las cantidades que debemos invertir en cada cuota.

Una vez que hemos localizado una surebet habrá que calcular el beneficio esperado y el porcentaje que debemos invertir en cada cuota, lo cual lo realizaremos con los siguientes cálculos:

El beneficio se calcula a partir de la inversa del sumatorio que utilizamos para verificar si tenemos una surebet. Vamos a ayudarnos el ejemplo anterior para calcular su beneficio esperado:

Beneficio = 1/0,93 = 1,0725

Nuestro beneficio esperado es del 7,25%.

Una vez calculado el beneficio podremos calcular el porcentaje de inversión que debe ir destinado a cada apuesta de la siguiente forma:

Porcentaje a apostar = 1/cuota x beneficio

Vamos a calcular el porcentaje a apostar en nuestro ejemplo:

Cuota 1 = 1/2,2 x 1,07 = 48,75%

Cuota 2 = 1/3,6 x 1,07 = 29,79%

Cuota 3 = 1/5 x 1,07 = 21,45%

Es el momento de comprobar que el sistema funciona, vamos a invertir de forma ficticia 100€ para comprobar que independientemente del resultado del partido vamos a obtener un beneficio del 7,25% de la cantidad invertida.

Apostado 48,75€ en la primera cuota, 29,79€ en la segunda cuota y 21,45€ en la tercera.

- Si el partido lo gana el primer equipo obtendremos unas ganancias de 48,75 x 2,20 = 107,25€

- Si el partido acaba en empate obtendremos unas ganancias de 29,79 x 3,60 = 107,25€

- Si el partido lo gana el segundo equipo obtendremos unas ganancias de 21,45 x 5,00 = 107,25€

Podemos verificar que independientemente del resultado del partido tendremos un beneficio de 7,25 €, ya que, hemos invertido 100€ y siempre obtendremos 107,25€.

2.2.2 Riesgos de Surebet

A pesar de que las surebets son apuestas que matemáticamente no implican riesgo, sí que tienen asociadas circunstancias que pueden derivar en problemas. Recordemos que las surebets o apuestas seguras son errores de las casas, que se esmeran en no cometer y corregir rápidamente si se producen. Repasamos cuáles son estos riesgos:

- El primero de todos es que las casas no permiten esta manera de apostar y si la detectan, pueden anular las apuestas e incluso cancelar la cuenta de usuario.

- La casa de apuestas puede anular la apuesta si detecta un error grave en las cuotas. Esto puede dejarnos con una apuesta realizada en otra casa que no estará cubierta por la apuesta anulada.

- En el evento deportivo pueden producirse situaciones anómalas (suspensiones, lesiones, etc.) que lleven a la casa a anular la apuesta. Es importante tener en cuenta cómo resuelven las apuestas las casas ante estos hechos.

- Las cuotas se actualizan muy rápido. En el lapso de tiempo que nos puede llevar hacer dos apuestas para obtener una surebet, las cuotas pueden variar, esfumándose la surebet y dejándonos con una apuesta hecha.

A todos estos riesgos hay que añadir la realidad de apostar haciendo surebets. Normalmente una surebet deja una rentabilidad muy baja, por lo que para sacar verdadero partido hay que disponer de cantidades respetables de dinero en varias casas de apuestas y ponerlas en juego asumiendo los riesgos que se acaban de apuntar.

2.3 Aplicaciones Software Existentes

A la hora de desarrollar una aplicación real, es importante revisar el estado del arte de la tecnología, con el fin de crear un sistema actualizado y que cumpla con las especificaciones impuestas por suspredecesores.

En el momento actual existen numerosas opciones comerciales de sofwares de surebet, las más conocidas son: Betburger, Bethunter, ardmate, betwasp…

La función principal de estas páginas es la muestra por pantalla de las diferentes surebet que existen en el mercado según una serie de filtros que te permite aplicar el software.

Estos programas se pueden utilizar a partir de una suscripción mensual que oscilan entre 150€ y 350€ en función de la modalidad de suscripción escogida. Las modalidades de suscripción que suelen existir son entre poder ver las surebet antes de que empiecen los partidos, ver las surebet mientras están disputándose los partidos y ambas.

Las suscripciones de las surebet en directo son más caras, ya que, es más posible encontrar mejores

surebet porque las casas de apuesta tienen que cambiar las cuotas en función de los datos en vivo de dicho partido y esto hace que la varianza entre las cuotas de las diferentes casas de apuestas sea mayor.

Por otro lado, existen numerosas calculadoras de surebet, las cuales no te muestran las surebet existentes, sino que al introducir manualmente las cuotas de las diferentes apuestas el programa te calcula si son surebet y en caso de que si lo sean te calcula el beneficio esperado y la cantidad a apostar en cada apuesta.

3 Librerías Utilizadas

3.1 Scraping

3.1.1 Introducción

Los motores de búsqueda, como Google, utilizan desde hace tiempo los denominados rastreadores web o crawlers, que exploran Internet en busca de términos definidos por el usuario. Los rastreadores son tipos especiales de bots, que visitan una página web tras otra para generar asociaciones con términos de búsqueda y categorizarlos. El primer rastreador web se creó ya en 1993, cuando se presentó el primer motor de búsqueda: Jumpstation.

Entre estas técnicas de rastreo se incluye el web scraping o webharvesting, en nuestro caso veremos el web scraping.

El funcionamiento teórico del web scraping es extremadamente simple, en verdad, y funciona mediante dos partes: un rastreador web y un raspador web. El rastreador de telarañas es el caballo y el raspador es el carro. El rastreador conduce el raspador a través de Internet, donde extrae los datos solicitados.

- El rastreador

  Un rastreador web, al que generalmente llamamos “araña”, es una inteligencia artificial que navega por Internet para indexar y buscar contenido siguiendo enlaces y explorando, como una persona con demasiado tiempo libre. En muchos proyectos, primero «rastrea» la web o un sitio web específico para descubrir URL que luego pasa a su raspador.

- El raspador

  Un web scraper es una herramienta especializada diseñada para extraer datos de una página web de forma precisa y rápida. Los webs scrapers varían ampliamente en diseño y complejidad, según el proyecto. Una parte importante de cada raspador son los localizadores de datos (o selectores) que se utilizan para encontrar los datos que desea extraer del archivo HTML; por lo general, se aplica XPath, selectores CSS, regex o una combinación de ellos.

3.1.2 Lenguajes de programación

En este apartado vamos a mostrar y analizar los mejores lenguajes de programación orientados a web scraping.

- Node.js:

  Node.js es adecuado para sccraping en sitios web dinámicos y admite el scraping distribuido en Internet. Este lenguaje es útil para recopilar datos de sitios web básicos y avanzados.

- Python:

  Python es actualmente el lenguaje de codificación más popular. Es más eficaz para el web scraping debido a su función todo en uno para la recopilación de datos web.

  Se puede programar fácilmente sin ningún error porque es un lenguaje simple que es fácil de entender y usar.

- Ruby:

  Este es un lenguaje de código abierto con una interfaz fácil de usar. Es fácil de entender y aplicar.

  La característica única de Ruby es que trabaja conjunto a numerosos lenguajes; como Eiffel, Smalltalk, Perl, Lip, Ada, etc.

  Ruby está diseñado para estabilizar la programación funcional con la ayuda de la programación imperativa.

- C y C++:

  C y C++ son excelentes soluciones de ejecución, aunque sus servicios pueden ser costosos cuando se realizan tareas de web scraping y por lo tanto, no se recomiendan a menos que sea bajo una demanda específica enfocada solo a la extracción de datos.

En este trabajo he utilizado Python, ya que, era novato en el proceso de web scraping y Python me facilitó el aprendizaje por su simpleza.

3.1.3 Librerias de Scraping

A continuación, se realiza un estudio de las diferentes librerías que se pueden utilizar para el proceso de scraping dentro de python. Por último, se realizará una síntesis de las ventajas y desventajas de los sistemas descritos, justificandola elección a partir de la cual se desarrollará el programa.

- **Request:**

  Request es la biblioteca más básica para hacer web scraping. Request nos permite realizar solicitudes HTML al servidor del sitio web para recuperar los datos de una página web que es el primer paso y más importante para web scraping.

  Sin embargo, la biblioteca de solicitudes no analiza los datos HTML recuperados, para ello necesitaremos bibliotecas como lxml o beautiful Soup que hablaremos más delante de las mismas.

  - Ventajas:
    - Sencillo.
    - Autentificación básica/ implícita.
    - URL y dominios internacionales.
    - Solicitudes fragmentadas.
    - Soporte de proxy HTTP.
  - Desventajas:
    - Recupera solo el contenido estático de una página.
    - No se puede utilizar para analizar HTML.

- **Lxml:**

  Lxml es una biblioteca de Python de análisis de XML y HTML bastante rápida y de alto rendimiento. Combina la velocidad y la simplicidad. La combinación de request y lxml es muy común en el web scraping.

  - Ventajas:
    - Analizador bastante rápido.
    - Ligero.
    - Utiliza arboles de elementos.

        - Api pythonic.

    - Desventajas:

        - No funciona bien con HTML mal diseñado.

        - El aprendizaje no es apto para principiantes.

- **BeautifulSoup:**

    Es la biblioteca más utilizada para web scraping. Crea un árbol de aanálisis para analizar documentos HTML y XML. Convierte automáticamente los documentos a Unicode y los salientes a UTF-8.

    Es la librería mas sencilla y adecuada para principiantes. BeautifulSoup se suele combinar con request y también se puede combinar con lxml.

    - Ventajas:

        - Requiere pocas líneas de código.

        - Hay mucha documentación muy bien explicada.

        - Fácil de aprender para principiantes.

        - Robusto.

        - Funciona bien con HTML mal diseñado

    - Desventajas:

        - No funciona en páginas webs dinámicas.

- **Selenium:**

    Selenium es una biblioteca de Python creada originalmente para pruebas automatizadas de aplicaciones web y aunque su función en principio no fue el web scraping esto cambio con relativa rapidez.

    Selenium es un controlador web creado para la automatización de páginas web y esta función la hace muy especial, ya que, donde otras librerías no son capaces de ejecutarse en JavaScript, Selenium sobresale.

    Esta capacidad de ejecutarse en JavaScript en una página web le da a Selenium la posibilidad de extraer datos en páginas webs dinámicas. Carga y ejecuta JavaScript para cada página, lo que lo hace muy lento.

    - Ventajas:

        - Apto para principiantes.

        - Raspado web automatizado.

        - Ejecutable en páginas webs dinámicas.

    - Desventajas:

        - Muy lento.

        - Difícil de configurar.
        - Uso alto de CPU y memoria.
        - No es ideal para proyectos de gran escala.

- **Scrapy:**

    Scrapy no es solo una librería, es un marco de web scraping completo creado por los cofundadores de Acrapinghub. Es una solución de raspado de banda en toda regla que hace todo el trabajo pesado por ti.

    Scrapy tiene bots araña que pueden rastrear varios sitios web y extraer datos.

    Lo mejor de Scrapy es que es asíncrono. Puedes realizar múltiples solicitudes JTTP simultáneamente. Esto nos ahorraría mucho tiempo y aumenta la eficacia.

    Aunque Scrapy no puede manejar JavaScript como Selenium, puede compaginarlo con la librería Splash, un navegador de web liviano, para poder extraer datos de sitios web dinámicos.

    - Ventajas:
        - Asíncrono.
        - Excelente documentación explicativa.
        - Varios complementos.
        - Bajo uso de CPU y memoria.
        - Arquitectura bien diseñada.
    - Desventajas:
        - Difícil de aprender.
        - No apto para principiantes.
        - Extenso para trabajos fáciles.

Librería escogida:

En este trabajo he utilizado las librerías Request en combinación con BeutifulSoup para la extracción de datos, ya que, el sitio web donde extraigo datos es una web estática y no tenía la necesidad de utilizar Selenium al no ser una página web dinámica.

Por otro lado, me decanté por no escoger scrapy porque la dificultad de aprendizaje no merecía la pena en relación a la dimensión del programa y por otro lado la documentación de BeutifulSoup era bastante extensa y bien explicada.

3.2 Interfaz Gráfica

En este apartado vamos a ver las diferentes librerías para interfaces gráficas de Python y una explicación de cual hemos escogido para la realización del trabajo.

Vamos a hacer un repaso de las librerías más usadas de Python para interfaces gráficas:

- **Tkinter:**

  Tkinter es una biblioteca más famosa en el mundo del desarrollo de aplicaciones Python. Este marco GUI viene integrado con Python y no es necesario instalarlo por separado.

  La biblioteca se utiliza para crear aplicaciones GUI para computadoras y no es compatible con el desarrollo de aplicaciones móviles. El framework Tkinter es bueno para aquellos que son nuevos en Python y quieren aprender a desarrollar aplicaciones.

- **Kivy:**

  Este es el marco de interfaz gráfica de usuario de Python más preferido sobre el que se pueden construir aplicaciones móviles y de computadora. Es un marco Open GL y viene en forma de una biblioteca de terceros fabricada por la comunidad de Kivy.

  Esta biblioteca contiene varias funciones que pueden ayudar a los desarrolladores a crear aplicaciones sólidas para su sistema. Los sistemas operativos compatibles con esta biblioteca son Windows, Mac, Linux, Android, ios y Raspberry Pi. Es de ejecución muy rápida y contiene widgets interesantes que pueden hacer que la aplicación se vea más hermosa.

- **PyQT:**

  PyQT es una increíble biblioteca de Python que se utiliza para crear aplicaciones multiplataforma tanto para computadoras como para dispositivos móviles y se ejecuta en el marco Qt desarrollado por Nokia. Esta biblioteca viene en dos versiones que son de pago y gratuitas.

  El principal inconveniente de la versión gratuita es la limitación de las funciones que ofrece, pero sí, es adecuada para que un principiante comience con la versión gratuita. Los sistemas operativos que admite son Windows, Mac, Linux, Android y Zaurus.

- **PySide:**

  Esta es otra poderosa herramienta de desarrollo de GUI de Python que viene vinculada con el marco de Qt al igual que PyQT y ayuda a los desarrolladores a crear aplicaciones multiplataforma con facilidad.

  Esto también tiene una característica especial: se combina fácilmente con PyQT 4 y, por lo tanto, ayuda a los desarrolladores a cambiar de PyQT a PySide para crear mejores aplicaciones. Los sistemas operativos compatibles con esta biblioteca son Windows, Mac, Linux, Android y Maemo.

Librería escogida:

Para este trabajo he escogido la librería Tkinter ya que la interfaz gráfica que tenía que representar es muy sencilla y no tenía ningún tipo de experiencia en interfaces gráficas.

Esta opción era la óptima en mi situación y ha rendido a la altura de lo esperado. Ha sido un aprendizaje y puesta en marcha sin dificultad.

4 Solución desarrollada

En este capítulo se explica cómo se ha desarrollado todo programa, desde explicar cada función hasta las diferentes dificultades que he estado teniendo con sus respectivas soluciones.

4.1 Visión General

Este programa, como he explicado en otros apartados, tiene como objetivo la extracción de datos de una página web de comparación de cuotas en diferentes casas de apuestas para su posterior procesamiento con el objetivo de encontrar apuestas seguras y sus características.

Vamos a empezar con el sitio web que compara cuotas entre las diferentes casas de apuestas, esta página es 'http://www.elcomparador.com/', la cual compara cuotas de los eventos deportivos más importantes del mundo, ya sea, futbol de los países más importantes (España, Inglaterra, Italia, Francia, Subamerica, Norteamerica…), las mejores ligas de baloncesto (España, EEUU, competiciones europeas…), tenis y beisbol.

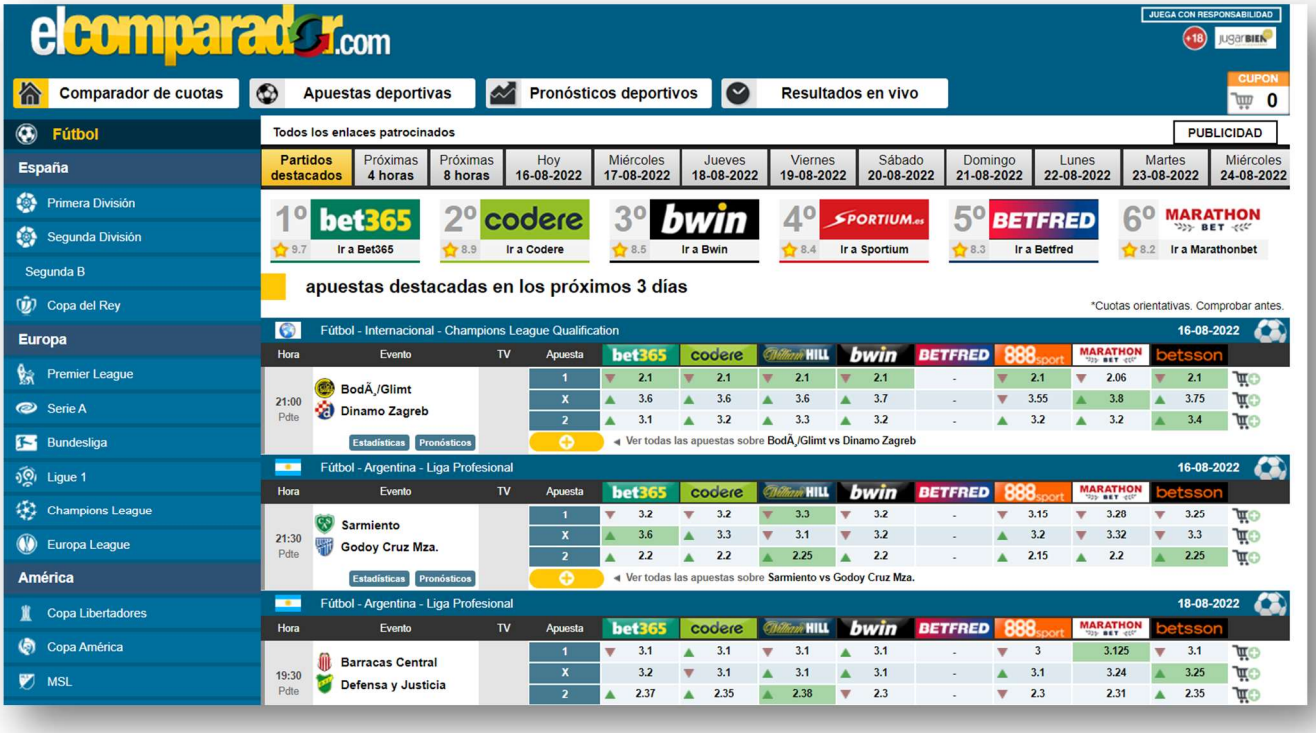

Figura 3.1 Página web 'elcomparador.com'

Como se puede observar en la figura 3.1 se comparan las cuotas de las diferentes apuestas que se pueden realizar entre 8 casas de apuestas (Bet365, Codere, WilliamHill, Bwin, Betfred, 888Sport, Marathon Bet y Betsson)

Una vez que sabemos de donde he de extraer los datos, vamos a ver superficialmente las diferentes funciones que he utilizado para dicha extracción y su posterior procesamiento, se pueden observar en la figura 3.2:

```
import ...
def equipos(link):...
def cuotas(link):...
def maximo(fila):...
def valido(cuotas, link):...
def cant_a_apostar(cuotas, cant_apuestas):...
def simplificar(cuotas, cant_apuestas, nombres, surebet, orden):...
def exportar_datos(porcentaje_a_apostar, beneficio, cuotas, nombres, partidos_seguros, cant_apuestas, orden):...
def casas_apuestas(orden):...
def fecha():...
def leer_fechas():...
def leer_datos(fecha):...
def final():...
def principal(link):...
def main():
```

Figura 3.2 Funciones utilizadas en el programa

- Función ‘equipos’: Se utiliza para la extracción de los nombres de los equipos.
- Función ‘cuotas’: Utilizada para la extracción de todas las cuotas de una competición.
- Función ‘maximo’: Su función es extraer la cuota máxima de cada apuesta (ver figura 3.3) e identificar a que casa de apuesta pertenece.

Apuesta	bet365	codere	William HILL	bwin	BETFRED	888sport	MARATHON BET	betsson
1	▼ 2.1	▼ 2.1	▼ 2.1	▼ 2.1	-	▼ 2.1	▼ 2.06	▼ 2.1
X	▲ 3.6	▲ 3.6	▲ 3.6	▲ 3.7	-	▼ 3.55	▲ 3.8	▲ 3.75
2	▲ 3.1	▲ 3.2	▲ 3.3	▲ 3.2	-	▲ 3.2	▲ 3.2	▲ 3.4

Figura 3.3 Recorte de página web ‘elcomparador.com’

- Función ‘valido’: Su función es determinar si en un partido en particular es viable una apuesta segura.
- Función ‘Cant_a_apostar’: En el supuesto caso en el que sea viable una apuesta segura en un partido se calcula la cantidad (%) que se debe de apostar a cada cuota y el beneficio (%) que obtendríamos en ese partido.
- Función ‘simplificar’: Regresa los valores de las cuotas y los nombres de los equipos que ya hemos validado que son apuestas seguras.
- Función ‘exportar_datos’: Se exportan los datos de los nombres de los equipos, cuotas, casas de apuestas, beneficio y la cantidad a apostar en un archivo txt.
- Función ‘casas_apuestas’: Convierte el orden que tiene asignado cada casa de apuesta a su nombre propio.

- Función ‘fecha’: Escribe la fecha en el documento txt donde se exportan los datos antes de cada ciclo de búsquedas.
- Función ‘leer_fechas’: Extrae la última fecha del archivo txt para la interfaz gráfica.
- Función ‘leer_datos’: Extrae los datos del archivo txt de la última búsqueda para su exposición en la interfaz gráfica.
- Función ‘final’: Escribe un espacio en el archivo txt para diferenciar entre búsquedas.
- Función ‘principal’: Es el programa principal donde se extraen datos y se analizan en cada una de las URL utilizadas.
- Función ‘main’: Se utiliza para ejecutar la función principal con las diferentes URL utilizadas.

4.2 Extracción de Datos

En este apartado vamos a explicar paso a paso cómo funciona nuestro programa empezando por la extracción de datos de la página web ‘http://www.elcomparador.com/’ para su posterior procesamiento.

En primer lugar, hay que hablar de la función ‘main’ (ver figura 3.4) que es la función que sustenta todas las demás.

```
def main():
    fecha()
    principal("http://www.elcomparador.com/futbol/europa/championsleague")
    principal("http://www.elcomparador.com/futbol/espa%C3%B1a/primeradivision")
    principal("http://www.elcomparador.com/futbol/espa%C3%B1a/segundadivision")
    principal("http://www.elcomparador.com/futbol/espa%C3%B1a/segundab")
    principal("http://www.elcomparador.com/futbol/espa%C3%B1a/copadelrey")
    principal("http://www.elcomparador.com/futbol/inglaterra/premierleague")
    principal("http://www.elcomparador.com/futbol/italia/seriea")
    principal("http://www.elcomparador.com/futbol/alemania/bundesliga1")
    principal("http://www.elcomparador.com/futbol/francia/ligue1")
    principal("http://www.elcomparador.com/futbol/europa/europaleague")
    principal("http://www.elcomparador.com/futbol/america/copalibertadores")
    principal("http://www.elcomparador.com/futbol/america/copaamerica")
    principal("http://www.elcomparador.com/futbol/usa/msl")
    principal("http://www.elcomparador.com/futbol/brasil/seriea")
    principal("http://www.elcomparador.com/baloncesto/espa%C3%B1a/acb")
    principal("http://www.elcomparador.com/baloncesto/espa%C3%B1a/leb")
    final()
    fechas = leer_fechas()
    global n_partidos
    datos = leer_datos(fechas[len(fechas) - 1])
    for i in arbol.get_children():
        arbol.delete(i)
    for i in range(n_partidos):
        arbol.insert("", END, text=datos[i * 12], values=(
            datos[i * 12 + 1], datos[i * 12 + 2], datos[i * 12 + 3], datos[i * 12 + 4], datos[i * 12 + 5],
            datos[i * 12 + 6], datos[i * 12 + 7], datos[i * 12 + 8], datos[i * 12 + 9], datos[i * 12 + 10],
            datos[i * 12 + 11]))
    n_partidos = 0
    raiz.after(200000, main)
```

Figura 3.4 Función ‘main’

Lo primero que ejecuta esta función es la función 'fecha' que como se puede observar en la figura 3.6 abre el archivo 'datos.txt' y 'fechas.txt' e introduce la fecha actual de búsqueda para tener constancia de que día y a qué hora hemos realizado dicha búsqueda.

Una vez hemos ejecutado la función 'fecha' ejecuta las funciones 'principal' con cada una de las URL que tiene la página web, en cada una de las URL están los partidos de una competición determinada como pudimos ver en la figura 3.4 de la página web 'http://www.elcomparador.com/'.

```
def principal(link):
    nombres = equipos(link)
    if nombres:
        max_cuotas, orden = cuotas(link)
        partidos_seguros, cant_apuestas = valido(max_cuotas, link)
        if len(partidos_seguros) > 0:
            cuotas_def, nombres_def, orden_def = simplificar(max_cuotas, cant_apuestas, nombres, partidos_seguros,
                                                             orden)
            porcentaje_a_apostar, beneficio = cant_a_apostar(cuotas_def, cant_apuestas)
            exportar_datos(porcentaje_a_apostar, beneficio, cuotas_def, nombres_def, partidos_seguros, cant_apuestas,
                           orden_def)
            global n_partidos
            n_partidos=n_partidos+len(partidos_seguros)
    return
```

Figura 3.5 Función 'principal'

En la función 'principal' lo primero que haremos es llamar a la función 'equipos' la cual nos retornará una lista con los nombres de los equipos de todos los partidos de esa URL como se observa en la figura 3.5.

```
def fecha():
    archivo = open("datos.txt", "a")
    archivo.write(datetime.now().strftime("%d-%m-%Y %H:%M:%S"))
    archivo.write("\n")
    archivo = open("fechas.txt", "a")
    archivo.write(datetime.now().strftime("%d-%m-%Y %H:%M:%S"))
    archivo.write(",")
```

Figura 3.6 Función 'fecha'

Aquí es donde entra en juego las bibliotecas Request y BeautifulSoup. Utilizaremos request para extraer todo el contenido del HTML del sitio web y una vez lo hayamos hecho podremos utilizar los comandos de BautifulSoup para buscar entre todo el HTML el fragmento que queramos, en este caso los nombres de los equipos de futbol.

Utilizamos el comando 'request.get(link)' para extraer en la variable 'r' todo el HTML del link de la página en cuestión y una vez lo hayamos hecho pasamos las entidades HTML a caracteres Unicode y lo guardamos en la variable 'soup', la cual utilizaremos para buscar el contenido especifico que deseemos con determinados comandos de

BeautifulSoup que paso a definir.

Los comandos más utilizados son ‘find(‘a’,{‘b’ : ’c’ })’ y ‘find_all(‘a’,{‘b’ : ’c’ })’ donde ‘a’ es la etiqueta, ‘b’ es el nombre del atributo y ‘c’ es el valor del atributo ‘b’. También se ejecuta si solo se indica el nombre del atributo y se elimina todo los demás. El comando ‘find’ busca el primer elemento que cumpla con las exigencias de dentro del paréntesis mientas que find_all busca todos los elementos que cumplan con las especificaciones del paréntesis. Existen muchos más comandos de búsqueda en BeautifulSoup pero estos son los más comunes e importantes.

Al finalizar el proceso de búsqueda obtendremos una lista ‘equipos’ con todos los elementos donde se sitúan los nombres de los equipos, ahora lo último que tenemos que hacer es extraer el contenido de los elementos en forma de texto. Para ello utilizamos un ‘for’ que dentro de la lista ‘equipos’ que vaya iterando cada elemento de la lista. A estos elementos de la lista le extraeremos el contenido en forma de texto con ‘get_text()’ y lo insertaremos en la lista ‘nombre_equipos’ que finalmente será lo que la función ‘equipos’ retornará.

```
def equipos(link):
    r = requests.get(link)
    soup = BeautifulSoup(r.text, "lxml")
    equipos = soup.find_all('span', {'class': 'equipo'})
    nombre_equipo = []
    cont = 0
    if equipos:
        for i in equipos:
            nombre_equipo.insert(cont, i.get_text())
            cont = cont + 1
    return nombre_equipo
```

Figura 3.7 Función ‘equipos’

Quisiera mencionar que he incluido un ‘if’ antes del buche ‘for’ por si en un link de alguna competición aún no hubiese ningún partido registrado con vacaciones o algún otro motivo y al no recoger ningún nombre me saltase un error en el buche ‘for’ posterior.

Volviendo otra vez a la función ‘principal’, ya teniendo una lista con todos los nombres de los equipos, vemos en la figura 3.7 que añadimos un ‘if’ por si la lista ‘equipos’ estuviese vacía para que en ese caso no siguiese con el proceso porque daría errores.

Justo después llamamos a la función ‘cuotas’ a la cual le pasaremos el link con el que estamos trabajando con la finalidad de extraer la máxima cuota de cada apuesta.

Como se pues observar en la figura 3.8 lo primero que hacemos es extraer del html todas las cuotas con el mismo método que utilizamos para extraer los nombres con la diferencia que en este caso queremos extraer dos valores distintos de etiquetas, ya que, el HTML está diseñado de tal manera que las filas de datos de cuotas estaban divididas entre los nombres ‘pares’ e ‘impares’. Por este motivo la solución a este problema fue extraer los dos valores de etiqueta en la misma lista de Python.

En esta lista se fueron guardando los datos de los dos valores de etiqueta indistintamente por orden que se lo fuese encontrando, de esta manera pude conseguir que la variable 'total_cuotas' tuviese todas las cuotas con un orden especifico, teniendo primero la primera final de impares seguido de la primera fila de pares seguido a su vez de la segunda fila de impares y así sucesivamente.

```
def cuotas(link):
    r = requests.get(link)
    soup = BeautifulSoup(r.text, "lxml")
    total_cuotas = soup.find_all("div", {'class': ['par', 'impar']})
    total_casas_aux = soup.find("div", {'id': 'contenedor_evento'})
    total_casas = total_casas_aux.find_all(class_='logos_lleno')
    cuotas_fila = []
    max = []
    orden = []
    for i in range(int(len(total_cuotas) / len(total_casas))):
        for t in range(len(total_casas)):
            cuotas_fila.insert(t, total_cuotas[t + (len(total_casas) * i)].get_text())
        m, n = maximo(cuotas_fila)
        max.insert(i, m)
        orden.insert(i, n)
        cuotas_fila.clear()

    return max, orden
```

Figura 3.8 Función 'cuotas'

Una vez tengo la lista con todas las cuotas, ahora necesitaba el dato de cuantos valores tiene cada fila para saber que cuotas son de cada apuesta. Para ello no me quedaba otra que extraer los datos de las casas de apuesta, ya que, hay la misma cantidad de cuotas que de casas de apuesta como se observa en la figura 3.9.

Hora	Evento	TV	Apuesta	bet365	codere	William HILL	bwin	BETFRED	888sport	MARATHON BET	betsson
21:00 Pdte	Celtic – Real Madrid		1	-	-	▼ 5.8	▼ 5.75	-	-	▼ 5.45	-
	Estadísticas		X	-	-	▲ 4.2	▼ 4.33	-	-	▼ 4.333	-
			2	-	-	▼ 1.53	▲ 1.53	-	-	▲ 1.61	-
			⊕	◂ Ver todas las apuestas sobre Celtic vs Real Madrid							
21:00 Pdte	RasenBallsport Leipzig – Shakhtar Donetsk		1	1.3	-	1.29	▼ 1.3	-	-	▲ 1.3	-
	Estadísticas		X	5.75	-	5.5	▲ 5.75	-	-	▼ 6.1	-
			2	8	-	▲ 11	▲ 9	-	-	▼ 9.7	-
			⊕	◂ Ver todas las apuestas sobre RasenBallsport Leipzig vs Shakhtar Donetsk							
Fútbol - Champions League Grp. G											06-09-202
Hora	Evento	TV	Apuesta	bet365	codere	William HILL	bwin	BETFRED	888sport	MARATHON BET	betsson
18:45 Pdte	Borussia Dortmund – FC Copenhaguen		1	-	-	▼ 1.35	▼ 1.35	-	-	▲ 1.38	-
	Estadísticas		X	-	-	5	5.5	-	-	▼ 5.5	-
			2	-	-	▲ 8.5	▲ 7.5	-	-	▲ 7.8	-
			⊕	◂ Ver todas las apuestas sobre Borussia Dortmund vs FC Copenhaguen							
21:00 Pdte	Sevilla – Manchester City		1	5.75	-	▼ 6	-	-	-	▲ 6.5	-
	Estadísticas		X	4.33	-	▲ 4.33	-	-	-	▲ 4.65	-
			2	1.5	-	1.5	-	-	-	▼ 1.5	-
			⊕	◂ Ver todas las apuestas sobre Sevilla vs Manchester City							

Figura 3.9 Recorte de página web 'elcomparador.com'

A la hora de extraer el número de casas de apuestas no lo pude hacer de la forma convencional, buscando un valor de etiqueta que fuese común en todos ellos, ya que, en un mismo sitio web hay una fila de casas de apuestas por cada día de partidos como muestro en la figura 3.9.

Para solucionar este problema cree una variable auxiliar llamada 'total_casas_aux' donde extraer el primer contenedor de partidos que encontrase, ya que, en cada contenedor existen una única fila de casas de apuestas. Al obtener esta variable pude buscar dentro de ella los valores de las etiquetas de las casas de apuestas y extraerlo dentro de la variable 'total_casas'.

Una vez tengo la cantidad de casas de apuestas que hay para cada apuesta hago un buche que me recorra el total de filas (apuestas), ya que, la división del total de cuotas entre la cantidad de casas de apuesta para cada apuesta me da el número de total de filas (apuestas).

Y dentro de dicho bucle hice otro bucle que me recorriera cada cuota dentro de una misma fila, en este bucle relleno una lista auxiliar con todas las cuotas de una misma fila para enviar esta lista a una función que me encuentre el máximo valor y su posición (casa de apuesta que le corresponde).

Al tener la cuota máxima de cada apuesta la inserto en otra lista donde guardaré las cuotas máximas de cada apuesta. Por otro lado, guardaré en otra lista la posición (casa de apuesta a la que pertenece la cuota máxima) de cada apuesta. Estas dos listas 'orden' y 'max' será lo que devuelva esta función.

Paso a explicar la función que utilizo para encontrar el valor máximo y su posición, a esta función la llamé 'maximo' y se puede observar en la figura 3.10.

```
def maximo(fila):
    max = 0
    orden = 0
    for i in range(0, len(fila), 1):
        if fila[i] == '-':
            fila[i] = 1
        if float(fila[i]) > float(max):
            max = fila[i]
            orden = i
    return max, orden
```

Figura 3.10 Función 'maximo'

Esta función es muy sencilla, simplemente hago un bucle que recorra la fila de cuotas y cambie los '-' por un '1' porque en algunos casos no aparece ninguna cuota de una casa de apuesta y el sitio web pone un '-' en su lugar y el hecho de poner un '1' no cambia nada el máximo porque nunca será un máximo una cuota de valor '1'. Una vez que realizamos este cambio ya solo hay que encontrar el máximo de cada fila y retornarlo junto con su posición.

Una vez he llegado aquí ya tengo una lista con los nombres de los equipos de cada

partido, otra lista con las cuotas máximas de cada apuesta en cada partido y otra lista con la posición de cada cuota máxima que indicará más tarde la casa de apuesta a la que corresponde cada cuota máxima de cada apuesta.

Y con esto se acabaría la parte de extraer datos del sitio web y empezaríamos con el análisis de dichos datos para indicar que partidos son apuestas seguras y determinar las características de las mismas.

4.3 Análisis de Datos

Actualmente ya disponemos de todas las cuotas máxima de cada apuesta, la posición de dicha apuesta, que indicará más adelante la casa que corresponde a la cuota máxima de la apuesta y por último el nombre de todos los equipos del sitio web que estemos usando.

Ahora vamos a utilizar la función 'valido' para determinar qué partidos son actos para realizar una apuesta segura y cuales no lo son. Vamos a explicar los pasos a seguir para determinar dicha característica.

```
def valido(cuotas, link):
    r = requests.get(link)
    soup = BeautifulSoup(r.text, "lxml")
    total_opciones_aux = soup.find("div", {'class': 'borde_inferior'})
    total_opciones = total_opciones_aux.find_all("div", {'class': 'apuesta'})
    prob = 0
    posicion_sure_bet = []
    cant_partidos = int(len(cuotas) / len(total_opciones))
    for i in range(0, cant_partidos, 1):
        for n in range(len(total_opciones)):
            prob = prob + 1 / float(cuotas[n + i * len(total_opciones)])
        if prob < 1:
            posicion_sure_bet.append(i)
        prob = 0

    return posicion_sure_bet, len(total_opciones)
```

Figura 3.11 Función 'valido'

Como se puede observar en la figura 3.11 a la función 'valido' le pasamos el link del sitio web y la lista de las cuotas máximas de cada apuesta. Utilizamos el link para extraer el número de apuesta en cada partido, ya que, no serán las mismas cantidades de apuestas posibles en baloncesto que en futbol, por ejemplo, en baloncesto no existe la posibilidad de que el partido finalice en forma de empate y en el futbol sí por lo tanto el futbol tendrá 3 apuestas posibles (gana equipo local, empate y gana equipo visitante) mientras que en baloncesto solo existen 2 posibles apuestas (gana equipo local y gana equipo visitante).

Al intentar extraer el número de apuesta posible me encontré con el mismo problema que a la hora de encontrar la cantidad de casas de apuestas disponibles. El problema era que si buscaba dentro del HTML el valor de etiquita que tiene las diferentes apuestas me encontrará la totalidad de apuestas que existen para todos los partidos mientras que yo necesito saber la cantidad de apuestas que existen para cada partido de dicho sitio web.

Para solucionar dicho problema realice la misma operación que en el anterior problema. Busqué dentro del HTML el primer contenedor de un único partido utilizando el comando 'find' en vez de utilizar el comando 'find_all' y guardé dicho contenedor en una variable llamada 'total_opciones_aux'.

Una vez realizado esta operación utilizare la variable auxiliar 'total_opciones_aux' para buscar dentro de ella los valores de las etiquetas donde se encuentran las diferentes apuestas que existen para dicho contenedor. De esta manera obtengo una variable con la cantidad de apuestas de un único partido, que es lo que necesitaba.

Todavía me falta determinar si los partidos son apuestas seguras, para ello utilizaré dos bucles, una de ellos que recorra los partidos existentes y otro que este dentro del anterior que recorra las apuestas existentes en dichos partidos. Para obtener la cantidad de partidos solo hay que dividir la cantidad de cuotas que tengo en la lista de cuotas máximas entre la cantidad de apuestas que tiene cada partido.

Para determinar si un partido es acto para realizar una apuesta segura he de sumar los inversos de las cuotas y que esta suma sea menor que 1 como se mostró en el apartado 1.4.

Como se puede observar en la figura 3.11 he realizado la suma de los inversos de cada cuota en cada partido y una vez tengo el resultado de cada partido compruebo si el resultado del sumatorio es menor que 1. En los casos donde el resultado sea menor que 1 inserto en la lista 'posicion_sure_bet' la posición del partido que es acto para realizar la apuesta segura. Es decir, si en el link que estoy pasando a la función 'valido' existen 5 partidos y resulta que solo hay un partido que es acto para ser apuesta segura y este partido es el tercero, se añadirá a la lista 'posición_sure_bet' el número 2 y si el partido acto para ser apuesta segura fuese el primero se añadiría un 0.

Retornamos la lista que contiene la posición de las apuestas seguras y la cantidad de apuestas que existen en cada partido, ya que, nos hará falta más adelante.

Seguimos en la función 'principal', que recordamos que se puede observar en la figura 3.5. Ahora realizamos un 'if' que nos compruebe si en un link determinado existen partidos actos para apuesta segura comprobando si el tamaño de la lista 'posicion_sure_bet' es mayor a 0. Si fuese mayor que 0 es que existe alguna apuesta segura y procederíamos a calcular sus características y si su tamaño es igual a 0 directamente cambiaríamos de link en la función 'principal' para buscar partidos actos para apuesta segura en otra competición.

Si hubiese algún partido seguro pasaríamos a utilizar la función 'simplificar' a la cual pasaremos las listas de las cuotas máximas de cada apuesta, la lista de los nombres de los equipos, la variable que indica la cantidad de apuestas que tiene cada partido, la lista de las posiciones de los partidos que son apuestas seguras y la lista que indica la posición que tiene cada cuota máxima, es decir, a que casa de apuesta pertenece cada cuota máxima.

En esta función, que se pude observar en la figura 3.12, tiene como finalidad retornar una lista donde aparezcan las cuotas máximas de los partidos seguros, otra lista donde aparezcan los nombres de los equipos de los partidos que son apuesta segura y otra lista donde aparezcan las posiciones de las cuotas máximas de los partidos seguros que corresponden a las casas de apuestas de las cuotas máximas de la primera lista.

```
def simplificar(cuotas, cant_apuestas, nombres, surebet, orden):
    cuotas_def = []
    nombres_def = []
    orden_def = []
    for i in surebet:
        for n in range(i * cant_apuestas, i * cant_apuestas + cant_apuestas, 1):
            cuotas_def.append(cuotas[n])
            orden_def.append(orden[n])
        for m in range(i * 2, i * 2 + 2, 1):
            nombres_def.append(nombres[m])
    return cuotas_def, nombres_def, orden_def
```

Figura 3.12 Función ‘simplificarr’

Con esta función ya conoceríamos los partidos que son seguros, sus cuotas, las casas de apuestas de cada cuota y los nombres de los partidos seguros. Ya solo nos faltaría conocer el beneficio que obtendríamos con cada apuesta segura y la cantidad de dinero en tanto por cierto que debemos de apostar en cada apuesta para que la apuesta segura se realice correctamente.

```
def cant_a_apostar(cuotas, cant_apuestas):
    sumatorio = 0
    porcentaje_a_apostar = []
    beneficio = []
    for i in range(int(len(cuotas) / cant_apuestas)):
        for n in range(cant_apuestas):
            sumatorio = sumatorio + 1 / float(cuotas[i * cant_apuestas + n])
        for m in range(cant_apuestas):
            porcentaje_a_apostar.append(round(((1 / float(cuotas[i * cant_apuestas + m])) / sumatorio) * 100, 2))
        beneficio.append(round((1 / sumatorio) * 100 - 100, 2))
        sumatorio = 0
    return porcentaje_a_apostar, beneficio
```

Figura 3.13 Función ‘cant_a_apostar’

Para ello utilizaremos la función ‘cant_a_apostar’ a la cual le pasaremos la lista donde guardamos las cuotas máximas de los partidos seguros y la variable que nos indica la cantidad de apuestas que tiene cada partido.

Como se puede observar en la figura 3.13 para realizar el cálculo del beneficio esperado y de la cantidad a apostar en cada apuesta me apoyo en un buche que me recorra cada partido y dentro de él otros dos buches que me recorran cada apuesta dentro

del partido.

El primer buche que me recorre las apuestas dentro del partido lo utilizo para el cálculo del sumatorio de las inversas de las cuotas, ya que, la inversa de este sumatorio me indica el beneficio esperado.

El siguiente buche de apuestas dentro de un partido lo utilizo para calcular cual es la cantidad que se debe de apostar en cada apuesta. La cantidad que se debe apostar en cada apuesta es la división entre la inversa de la cuota entre el sumatorio del primer buche o lo que es lo mismo, la inversa de cada cuota por el beneficio esperado.

La manera que se calcula el beneficio esperado y la cantidad a apostar en cada cuota aparece explicado en el apartado 1.4.1.

Al llegar a este paso ya disponemos de todas las características necesarias para definir la apuesta segura y solo nos faltará exportar las características a un archivo .txt y enseñar dicha información en la interfaz gráfica.

4.4 Exportar datos e Interfaz gráfica

Actualmente ya hemos extraído todos los datos necesarios del HTML y hemos procesado estos datos con la intención de determinar que partidos son actos para realizar una apuesta segura y que características tiene dicha apuesta segura.

```
def exportar_datos(porcentaje_a_apostar, beneficio, cuotas, nombres, partidos_seguros, cant_apuestas, orden):
    archivo = open("datos.txt", "a")

    for i in range(len(partidos_seguros)):
        for n in range(2):
            archivo.write(nombres[n + i * 2])
            archivo.write(",")
        archivo.write(str(beneficio[i]))
        archivo.write("%,")
        for n in range(cant_apuestas):
            archivo.write(str(cuotas[n + i * cant_apuestas]))
            archivo.write(",")
            archivo.write(casas_apuestas(orden[n + i * cant_apuestas]))
            archivo.write(",")
        for n in range(cant_apuestas):
            archivo.write(str(porcentaje_a_apostar[n + i * cant_apuestas]))
            archivo.write("%,")
        archivo.write(";\n")
```

Figura 3.14 Función ‘exportar_datos’

El siguiente paso en la función ‘principal’ (figura 3.5) es la función ‘exportar_datos’ cuya finalidad es la extracción de todos los datos generados sobre los partidos seguros al archivo .txt donde anteriormente introdujimos la fecha a la cual se inició la búsqueda de dichos datos.

En la figura 3.14 podemos observar que la función ‘exportar_datos’ se le pasa la lista de los porcentajes a apostar en cada cuota, la lista de los beneficios de cada partido, la lista de las cuotas de cada partido, la lista de los nombres de cada partido, la variable de la cantidad de partidos seguros, la variable que indica la cantidad de apuestas que hay en cada partido y por último la lista que indica que casa de apuesta corresponde a cada cuota.

```
24-02-2022 02:38:55
Real Sociedad,Osasuna,0.06%, 1.88,Betsson, 3.5,Bet365, 5.5,Bwin,53.22%,28.59%,18.19%,;
Valencia,Athletic Bilbao,0.15%, 2.77,Marathon Bet, 3.3,Betsson, 2.99,Marathon Bet,36.16%,30.35%,33.5%,;
Borussia Monchengladbach,Wolfsburg,1.18%, 2.12,Marathon Bet, 3.75,Betsson, 4,Bwin,47.73%,26.98%,25.29%,;
Eintracht Frankfurt,Bayern Munich,0.46%, 7.5,Betfred, 5.8,Marathon Bet, 1.45,Marathon Bet,13.39%,17.32%,69.28%,;
_
```

Figura 3.15 Recorte archivo ‘datos.txt’

La función ‘exportar_datos’ consiste en un bucle que recorre cada partido seguro escribiendo de cada partido el nombre de los equipos, el beneficio, la cuota y casa de apuesta de cada apuesta disponible y por último el porcentaje a apostar en cada apuesta, en dicho orden y separado por comas como se indica en la figura 3.15.

Dentro de la función ‘exportar_datos’ nos ayudamos de la función ‘casas_apuestas’ que tiene como finalidad interpretar las posiciones de las casas de apuestas de la lista ‘orden’ como los nombres de las casas de apuestas que corresponden a dichas posiciones como se puede observar en la figura 3.16.

```
def casas_apuestas(orden):
    if orden == 0:
        casa = "Bet365"
    if orden == 1:
        casa = "Codere"
    if orden == 2:
        casa = "William Hill"
    if orden == 3:
        casa = "Bwin"
    if orden == 4:
        casa = "Betfred"
    if orden == 5:
        casa = "888Sport"
    if orden == 6:
        casa = "Marathon Bet"
    if orden == 7:
        casa = "Betsson"
    return casa
```

Figura 3.16 Función ‘casas_apuestas’

Por último, se hará un sumatorio global de los partidos seguros que estamos exportando, ya que, nos será de ayuda a la hora de administrar los datos a la interfaz gráfica.

Con ello habríamos realizado con éxito toda la función 'principal' y a través de la función 'main' recorreremos dicha función 'principal' hasta haber realizado este proceso en todas las competiciones que vamos a analizar cómo se puede observar en la figura 3.4.

Tras exportar todos los datos de todas las competiciones al archivo .txt usaremos dentro de la función 'main' la función 'final' que básicamente escribe un '-' al final de todos los datos exportados para diferenciar las diferentes búsquedas.

El siguiente proceso a realizar es la lectura del archivo .txt para exponer los datos en la interfaz gráfica. Para ello lo primero que necesito es la fecha de la última búsqueda y para ello utilizo la función 'leer_fechas' que básicamente me devuelve una lista con todas las fechas que existen dentro de mi archivo .txt.

Ahora para conseguir todos los datos de la última búsqueda utilizo la función 'leer_datos' (figura 3.17) a la cual le paso la última fecha que tengo registrada, es decir, de la lista que me ha retornado la función 'leer_fechas' escoger el último elemento.

```
def leer_datos(fecha):
    archivo = open("datos.txt", "r")
    n = archivo.read().find(fecha)
    with open("datos.txt") as f:
        content = f.read()
    i = n + len(fecha) + 1
    datos_aux = []
    datos = []
    while content[i] != "-":
        j = i
        while content[j] != ";":
            j += 1
        datos_aux = content[i:j].split(",")
        datos_aux.pop(12)
        for n in range(12):
            datos.append(datos_aux[n])
        i = j + 2
    return datos
```

Figura 3.17 Función 'leer_datos'

En esta función lo primero que hago es abrir el archivo .txt y buscar la posición donde se encuentra la fecha que le he pasado a esta función y una vez que lo encuentro recorro el contenido del archivo .txt desde la fecha escogida hasta el '-' que ya comentamos anteriormente que lo utilizaríamos para diferenciar entre búsquedas.

Dentro de este buche utilizaremos otro que nos recorra los caracteres entre los símbolos ';' que ya, al exportar los datos al archivo .txt escribimos un ';' al final de cada partido. Y, por último, guardaremos los caracteres que existen entre ',' en los elementos

de la lista ‘datos’.

De tal manera que al final de este proceso obtenemos una lista llamada ‘datos’ que contiene todos los datos de la última búsqueda y cada dato estará en una celda distinta de la lista.

```
for i in range(n_partidos):
    arbol.insert("", END, text=datos[i * 12], values=(
        datos[i * 12 + 1], datos[i * 12 + 2], datos[i * 12 + 3], datos[i * 12 + 4], datos[i * 12 + 5],
        datos[i * 12 + 6], datos[i * 12 + 7], datos[i * 12 + 8], datos[i * 12 + 9], datos[i * 12 + 10],
        datos[i * 12 + 11]))
```

Figura 3.18 Recorte de la función ‘main’

Por último, utilizamos un bucle que nos recorra todos los partidos seguros que hemos encontrado en la última búsqueda como podemos ver en la figura 3.18. De este modo podemos introducir cada elemento en las filas de la tabla que hemos creado en la interfaz gráfica.

Por último, nos encontramos con el problema de repetir todo el proceso a cada x tiempo quedando el programa abierto. Para ello utilizamos el comando ‘after (a,b)’ donde ‘a’ determina el tiempo que va a existir entre cada ciclo en segundos y ‘b’ es la función que va a repetirse.

De tal manera nuestro comando será ‘after (20000, main)’ para repetir a cada 20000 milisegundos la función ‘main’, ya que, no existen tanta variación de cuotas como para repetir el ciclo en menos tiempo. Lo podemos ver en la figura 3.4.

```
raiz = Tk()
raiz.title("SureBet")
raiz.geometry('1400x230')

arbol = ttk.Treeview(raiz, columns=(
    "EQUIPO 1", "EQUIPO 2", "BENEFICIO", "CUOTA 1", "CASA DE APUESTAS 1", "CUOTA 2", "CASA DE APUESTAS 2",
    "CUOTA 3",
    "CASA DE APUESTAS 3", "CANTIDAD A APOSTAR 1", "CANTIDAD A APOSTAR 2", "CANTIDAD A APOSTAR 3"))
```

Figura 3.19 Recorte de la parte de interfaz gráfica

En cuanto a la interfaz gráfica vamos a trabajar con Tkinter como explicamos en apartados anteriores. La intención es generar una tabla dentro de la interfaz que se vaya actualizando a medida que se efectúen búsquedas de partidos actos para apuestas seguras.

Para ello empezamos creando la interfaz incluyendo el título que será ‘SureBet’ y las dimensiones de la interfaz. Los comandos se pueden observar en la figura 3.19.

En las siguientes líneas crearemos la tabla con el comando '.Treeview(a , columns=(b))' donde 'a' es el nombre de tu interfaz y 'b' son los nombres de las columnas que generan la tabla separados por comas y entre comillas.

Para terminar de elaborar la tabla solamente hay que definir el nombre del encabezado de sus columnas y definir sus dimensiones y orientación además de cerrar la tabla con el comando '.pack()'. Podemos observar cómo se definen en la figura 3.20.

```
arbol.column("#0", minwidth=0, width=100, anchor=CENTER)
arbol.column("#1", minwidth=0, width=100, anchor=CENTER)
arbol.column("#2", minwidth=0, width=70, anchor=CENTER)
arbol.column("#3", minwidth=0, width=70, anchor=CENTER)
arbol.column("#4", minwidth=0, width=130, anchor=CENTER)
arbol.column("#5", minwidth=0, width=70, anchor=CENTER)
arbol.column("#6", minwidth=0, width=130, anchor=CENTER)
arbol.column("#7", minwidth=0, width=70, anchor=CENTER)
arbol.column("#8", minwidth=0, width=130, anchor=CENTER)
arbol.column("#9", minwidth=0, width=160, anchor=CENTER)
arbol.column("#10", minwidth=0, width=160, anchor=CENTER)
arbol.column("#11", minwidth=0, width=160, anchor=CENTER)

arbol.heading("#0", text="EQUIPO 1")
arbol.heading("#1", text="EQUIPO 2")
arbol.heading("#2", text="BENEFICIO")
arbol.heading("#3", text="CUOTA 1")
arbol.heading("#4", text="CASA DE APUESTAS 1")
arbol.heading("#5", text="CUOTA 2")
arbol.heading("#6", text="CASA DE APUESTAS 2")
arbol.heading("#7", text="CUOTA 3")
arbol.heading("#8", text="CASA DE APUESTAS 3")
arbol.heading("#9", text="CANTIDAD A APOSTAR 1")
arbol.heading("#10", text="CANTIDAD A APOSTAR 2")
arbol.heading("#11", text="CANTIDAD A APOSTAR 3")
arbol.pack()
```

Figura 3.20 Recorte de la parte de interfaz gráfica

5 Modo de Funcionamiento

En este apartado vamos a tratar lo correspondiente al análisis de resultados y al modo de funcionamiento del programa, es decir, como responde el programa al ser ejecutado y a los resultados que podemos esperarnos del mismo.

Al ejecutar el programa lo primero que se ejecuta es la interfaz gráfica, pero aparecerá vacía ya que todavía no se ha realizado la búsqueda de partidos actos para apuesta segura y por lo tanto solo aparece la interfaz gráfica que a su vez contiene la tabla de resultados vacía como se puede apreciar en la figura 4.1.

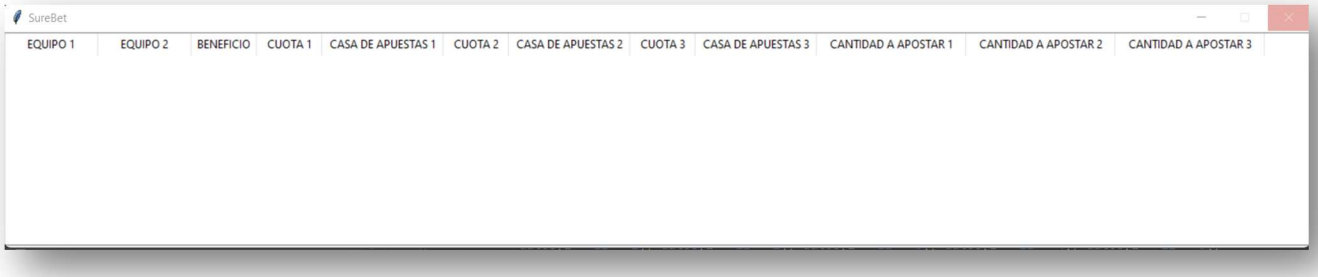

Figura 4.1 Inicio de la interfaz gráfica

A terminar de mostrar la interfaz gráfica empieza la función 'main' y con ello la búsqueda de partidos actos para realizar una apuesta segura. Esta búsqueda tarda una media unos 9 segundos aproximadamente entre que busca todos los datos, los analiza, los exporta a un archivo txt y por último recoge los datos del archivo txt y los muestra en la tabla de la interfaz gráfica.

Al terminar la búsqueda podemos visualizar algo parecido a la figura 4.2.

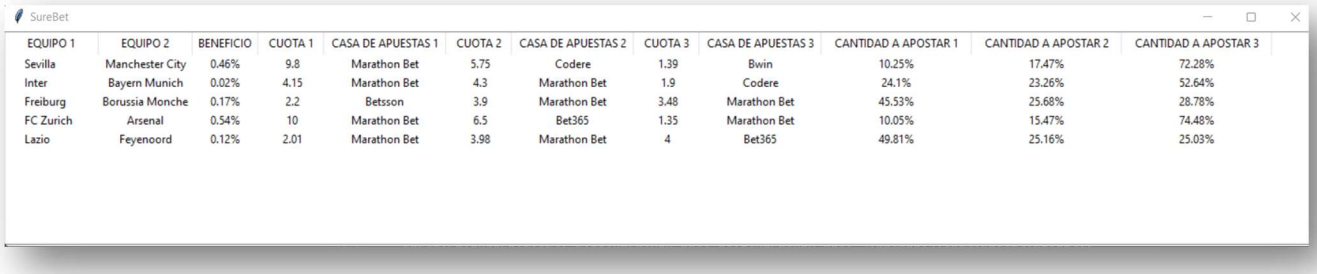

Figura 4.2 Resultados de la interfaz gráfica

Mientras no se cierre el programa la búsqueda de partidos actos para realizar una apuesta segura seguirá realizándose a cada 200 segundos. He determinado que éste sea el tiempo de espera entre búsquedas ya que solo habrá cambios de partidos seguros cuando algún partido haya acabado, algún partido se pongo como nuevo en el sitio web o por el contrario alguna cuota sea modificada por alguna casa de apuestas.

Por experiencia los casos expuestos anteriormente no suelen ocurrir en periodos cortos de tiempo y por ello una espera de 200 segundos me parece suficiente.

Una vez realizado las búsquedas podemos ver las búsquedas históricas en el archivo txt llamado ‘datos’ como puede apreciarse en la figura 4.3.

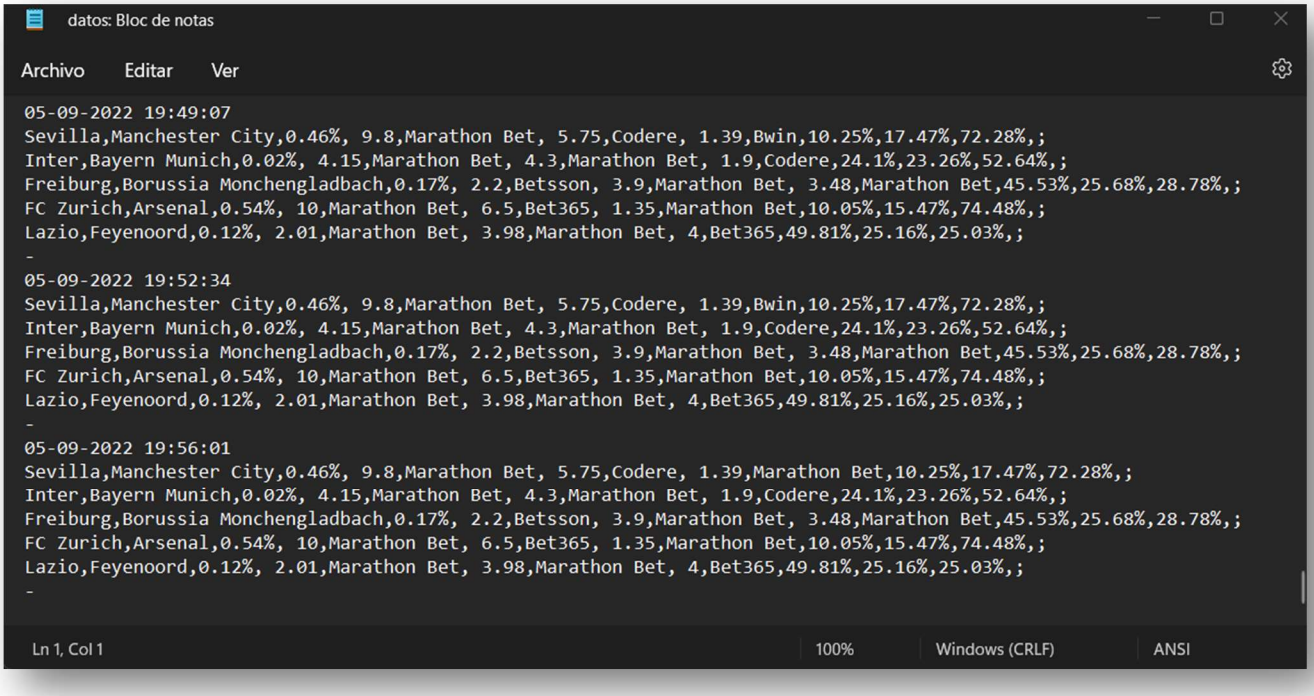

Figura 4.3 Históricos sacados de datos.txt

6 Conclusiones y desarrollo futuro

Tras la finalización del programa podemos recoger las conclusiones del mismo desde un punto de vista estructural como de un punto de vista de resultados al igual que podemos intuir los aspectos en los cuales se puede mejorar el programa. Vamos a empezar por las conclusiones y más adelante explicamos como podríamos mejorar el programa.

- CONCLUSIONES:

  - Con los datos acumulados desde mediados de febrero hasta principios de septiembre recogidos intermitentemente puedo afirmar que todos los días que he buscado partidos que sean actos para ser apuestas seguras he encontrado mínimo un resultado. Lo que me indica que la existencia de las apuestas seguras es real y recurrente, ya que, he tenido una media de 2 resultados por búsqueda.

  - Basado en los mismos resultados de la conclusión anterior puedo estimar la media de los beneficios de los partidos seguros en torno a 0,5% siendo los menores de 0,03€ y los mayores de 1,03%.

Estas son las conclusiones en el caso de buscar partidos actos para apuestas seguras en 8 casas de apuestas, en apuestas de ganar/perder/empatar y en las competiciones más famosas. En el caso de que ampliemos horizontes y tengamos en cuenta más casas de apuestas, mas tipos de apuestas y más competiciones las conclusiones se multiplicarían.

En cuanto al desarrollo a futura del programa podemos orientarlo en dos vertientes diferentes. Por un lado, mejorar el programa desde un punto de vista tecnológico y estadístico y por otro lado desde un punto de vista mas empresarial en busca de beneficios.

- Tecnológico y Estadístico:

  - El programa se puede mejorar claramente buscando partidos seguros entre mas competiciones y mas deportes o cualquier actividad que tenga opción a apostar en ella. Como por ejemplo el beisbol, rugby, petanca, esport o cualquiera que se nos ocurra. La variedad es inmensa.

  - También se puede ampliar el programa buscando entre otros tipos de apuesta, ya que, dentro de un partido hay inmensidad de tipos de apuesta como por ejemplo en el futbol que dentro de cada partido puedes apostar cuantos goles van a marcar o cuantas tarjetas va a haber. Nos valdrá cualquier apuesta que conlleve la totalidad de resultados posibles, es decir, podríamos hacerlo en las apuestas que son ‘mas de 2,5 goles’ y ‘menos de 2,5 goles’ porque no hay ningún resultado que no ocurra en estas dos apuestas. Sim embargo no podría hacerlo en apuestas que sean del tipo ‘van a meter 1 gol’, ‘van a meter 2 goles’…ya que, nunca podrás obtener la totalidad de resultados posibles entre las apuestas que existen.

- Podría hacer un control estadístico de los resultados obtenidos a partir del histórico de resultados pudiendo obtener cual es la competición que mas o mejores partidos seguros encuentra o cuales son las casas de apuestas que mas o mejores partidos seguros encuentra.

- Empresarial:

  Podemos buscar un beneficio del programa utilizando los resultados para realizar una apuesta segura y obtener el beneficio, pero para ello tenemos que tener en cuenta que las casas de apuestas conocen este método y trabajan para su detención. Por lo tanto nuestros esfuerzos deberían estar bajo esta premisa y buscar formas de operación a través de las cuales los trabajadores de las casas de apuestas no se den cuenta de tus intenciones para ello podemos hacer diferentes maniobras:

  - Utilizar diferentes nombres de usuario que se correspondan a diferentes personas que no tengan nada que ver unas con otras para que las operaciones parezcan que no son compartidas.

  - Realizar un programa informático que mueva automáticamente el dinero entre cuentas para que las cuentas en las diferentes casas de apuestas siempre tengan el saldo necesario.

  - Utilizar cuentas bancarias de terceros para que no vinculen a las personas que se están transfiriendo dinero entre ellas.

  - Hacer apuestas sin decimales, ya que, esto sería algo que los trabajadores de las casas de apuestas verían como raro. Para ello habría que modificar el programa para que los resultados de las cantidades a apostar sean números enteros y recalcular los beneficios.

Índice de Figuras

Bibliografía

[1] "¿Qué Es El Web Scraping?". IONOS Digital Guide, 2020, https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-el-web-scraping/.

[2] "¿Qué Es El Web Scraping Y Para Qué Se Utiliza?". Ciberseguridad, 2022, https://ciberseguridad.com/guias/recursos/web-scraping/#%C2%BFQue_es_el_web_scraping. Accessed 7 Sept 2022.

[3] "El Tutorial Definitivo De Numpy Para Principiantes En Ciencia De Datos | Datapeaker". Datapeaker, 2022, https://datapeaker.com/big-data/el-tutorial-definitivo-de-numpy-para-principiantes-en-ciencia-de-datos/. Accessed 7 Sept 2022.

[4] "Beautiful Soup Documentation — Beautiful Soup 4.9.0 Documentation". Crummy.Com, 2022, https://www.crummy.com/software/BeautifulSoup/bs4/doc/#calling-a-tag-is-like-calling-find-all. Accessed 7 Sept 2022.

[5] "Requests: HTTP For Humans™ — Requests 2.28.1 Documentation". Requests.Readthedocs.Io, 2022, https://requests.readthedocs.io/en/latest/. Accessed 7 Sept 2022.

[6] "Comparar Cuotas De Apuestas". Elcomparador.Com, 2022, http://www.elcomparador.com/. Accessed 7 Sept 2022.

[7] "Se Han Encontrado Apuestas Seguras / Surebet - Apuestas Profesionales". Es.Surebet.Com, 2022, https://es.surebet.com/surebets. Accessed 7 Sept 2022.

[8] "Surebet ¿Qué Es Y Como Encontrarla? | Blog Marathonbet". Blog Marathonbet, 2022, https://apuestas.marathonbet.es/glosario-apuestas/que-es-surebet-en-apuestas-deportivas/. Accessed 7 Sept 2022.

[9] Lafuente, Ainhoa. "Qué Es El Web Scraping - Aukera". Aukera, 2019, https://aukera.es/blog/web-scraping/. Accessed 7 Sept 2022.

[10]"Web Scraping With Ruby". Scrapingbee.Com, 2022, https://www.scrapingbee.com/blog/web-scraping-ruby/. Accessed 7 Sept 2022.

[11]“Python, Real. "Beautiful Soup: Build A Web Scraper With Python – Real Python". Realpython.Com, 2022, https://realpython.com/beautiful-soup-web-scraper-python/. Accessed 7 Sept 2022.

[12] "Web Scraping En Node.Js Para Humanos Usando Scrape-It". Medium, 2020, https://medium.com/@carboleda/web-scraping-en-node-js-para-humanos-usando-scrape-it-3c469bcb786. Accessed 7 Sept 2022.

[13] "The Ultimate Guide To Web Scraping With C++". Webscrapingapi, 2021, https://www.webscrapingapi.com/c-web-scraping/. Accessed 7 Sept 2022.

[14] "Requests And Responses — Scrapy 2.6.2 Documentation". Docs.Scrapy.Org, 2022, https://docs.scrapy.org/en/latest/topics/request-response.html. Accessed 7 Sept 2022.

[15] "Lxml - Processing XML And HTML With Python". Lxml.De, 2022, https://lxml.de/. Accessed 7 Sept 2022.

[16] "Beautiful Soup Documentation — Beautiful Soup 4.9.0 Documentation". Crummy.Com, 2022, https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Accessed 7 Sept 2022.

[17] "Beautiful Soup Documentation — Beautiful Soup 4.4.0 Documentation". Beautiful-Soup-4.Readthedocs.Io, 2022, https://beautiful-soup-4.readthedocs.io/en/latest/. Accessed 7 Sept 2022.

[18] "Web Scraping Con Python. Guía De Inicio De Beautiful Soup". J2LOGO, 2020, https://j2logo.com/python/web-scraping-con-python-guia-inicio-beautifulsoup/. Accessed 7 Sept 2022.

[19] "Beautifulsoup - Aprende Python". Aprende Python, 2022, https://aprendepython.es/pypi/scraping/beautifulsoup/. Accessed 7 Sept 2022.

[20] "Guide To Parsing HTML With Beautifulsoup In Python". Stack Abuse, 2020, https://stackabuse.com/guide-to-parsing-html-with-beautifulsoup-in-python/. Accessed 7 Sept 2022.

[21] "Beautifulsoup". DEV Community 👩‍💻👨‍💻, 2022, https://dev.to/t/beautifulsoup. Accessed 7 Sept 2022.

[22] "Selenium". Selenium, 2022, https://www.selenium.dev/. Accessed 7 Sept 2022.

[23] "Selenium With Python — Selenium Python Bindings 2 Documentation". Selenium-Python.Readthedocs.Io, 2022, https://selenium-python.readthedocs.io/. Accessed 7 Sept 2022.

[24] "Automatiza Tareas Dinámicas En La Web Con Selenium Webdriver - Bravent". Bravent, 2022, https://www.bravent.net/automatiza-tareas-dinamicas-en-la-web-con-selenium-webdriver/. Accessed 7 Sept 2022.

[25] "Selenium: Definition, How It Works And Why You Need It". Browserstack, 2022, https://www.browserstack.com/selenium. Accessed 7 Sept 2022.

[26] "Scrapy | A Fast And Powerful Scraping And Web Crawling Framework". Scrapy.Org, 2022, https://scrapy.org/. Accessed 7 Sept 2022.

[27] "¿Qué Es Scrapy? | Keepcoding Tech School". Keepcoding Tech School, 2022, https://keepcoding.io/blog/que-es-scrapy/. Accessed 7 Sept 2022.

[28] "Github - Scrapy/Scrapy: Scrapy, A Fast High-Level Web Crawling & Scraping Framework For Python.". Github, 2022, https://github.com/scrapy/scrapy. Accessed 7 Sept 2022.

[29] datos, El. "Extracción De Datos De Sitios Web Con Scrapy (I): Recopilando Información De Productos De Zara". El Mundo De Los Datos, 2021, https://elmundodelosdatos.com/extraccion-datos-sitios-web-productos-zara/. Accessed 7 Sept 2022.

[30] Ramirez, Luis. "Tutorial De Web Scraping Python En Español; Scrapy". Luisramirez.Dev, 2022, https://luisramirez.dev/tutorial-de-web-scraping-python-en-espanol-scrapy/. Accessed 7 Sept 2022.

[31] "Scrapy Tutorial". Tutorialspoint.Com, 2022, https://www.tutorialspoint.com/scrapy/index.htm. Accessed 7 Sept 2022.

[32] "Tkinter — Interface De Python Para Tcl/Tk — Documentación De Python - 3.10.7". Docs.Python.Org, 2022, https://docs.python.org/es/3/library/tkinter.html. Accessed 7 Sept 2022.

[33] Python, Recursos. "Introducción A Tcl/Tk (Tkinter) - Recursos Python". Recursos Python, 2021, https://recursospython.com/guias-y-manuales/introduccion-a-tkinter/. Accessed 7 Sept 2022.

[34] Python, Real. "Python GUI Programming With Tkinter – Real Python". Realpython.Com, 2022, https://realpython.com/python-gui-tkinter/. Accessed 7 Sept 2022.

[35] "Python - GUI Programming (Tkinter)". Tutorialspoint.Com, 2022, https://www.tutorialspoint.com/python/python_gui_programming.htm. Accessed 7 Sept 2022.

[36] "¿Qué Es Tkinter? | Keepcoding Tech School". Keepcoding Tech School, 2022, https://keepcoding.io/blog/que-es-tkinter/. Accessed 7 Sept 2022.

[37] "Empezando Por Lo Básico — Documentación De Guia Tkinter - 0.1.1". Guia-Tkinter.Readthedocs.Io, 2022, https://guia-tkinter.readthedocs.io/es/develop/chapters/5-basic/5.1-Intro.html. Accessed 7 Sept 2022.

[38] "Kivy: Cross-Platform Python Framework For NUI". Kivy.Org, 2022, https://kivy.org/. Accessed 7 Sept 2022.

[39] "Github - Kivy/Kivy: Open Source UI Framework Written In Python, Running On Windows, Linux, Macos, Android And Ios". Github, 2022, https://github.com/kivy/kivy. Accessed 7 Sept 2022.

[40] Toro, Luigys, and Luigys Toro. "Kivy: Un Framework Para Python Que Permite Desarrollar Aplicaciones De Manera Rápida". Desde Linux, 2017, https://blog.desdelinux.net/kivy-framework-para-python/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+UsemosLinux+%28Usemos+Linux%29. Accessed 7 Sept 2022.

[41] Martins, Samuel. "Build An Android Application With Kivy Python Framework - Logrocket Blog". Logrocket Blog, 2022, https://blog.logrocket.com/build-android-application-kivy-python-framework/. Accessed 7 Sept 2022.

[42] "Kivy Tutorial - Geeksforgeeks". Geeksforgeeks, 2020, https://www.geeksforgeeks.org/kivy-tutorial/. Accessed 7 Sept 2022.

[43] Python, Real. "Python And Pyqt: Building A GUI Desktop Calculator – Real Python". Realpython.Com, 2022, https://realpython.com/python-pyqt-gui-calculator/. Accessed 7 Sept 2022.

[44] "Learn Python Pyqt | Learn Python Pyqt". Pythonpyqt.Com, 2022, https://pythonpyqt.com/. Accessed 7 Sept 2022.

[45] "Pyside". Pypi, 2015, https://pypi.org/project/PySide/. Accessed 7 Sept 2022.

[46] "Pyqt Vs Pyside (Spanish)". DEV Community 👷‍♂️, 2019, https://dev.to/amigosmaker/pyqt-vs-pyside-spanish-2kmg. Accessed 7 Sept 2022.